International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

CloudCost AI: Intelligent Cost & Resource Optimizer

Arpitha R B
Department of Computer Applications St Francis College
arpithabhandare88@gmail.com

Ayesha Taj
Department of Computer Applications St Francis College
ayeshaa231626@gmail.com

Biju Francina
Department of Computer Applications St Francis College
francinabiju30@gmail.com

***Abstract*—Organizations that use cloud services often pay for idle or oversized virtual machines (VMs) and waste energy through poor resource utilization. Cost tools built by cloud vendors generally analyze only their own platform, so simple solutions that work across several clouds are needed. We introduce CloudCost AI, a machine-learning framework that lowers cost and improves resource use on three major clouds: Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). We built a synthetic dataset of 28,800 records from 60 VMs monitored at 15-minute intervals, containing VM configuration, CPU and memory utilization, network and disk I/O, latency, throughput, workload pattern, pricing, and cost. A Random Forest classifier recommends whether a VM should be scaled up, scaled down, or left unchanged, with an accuracy of about 94.6%. An XGBoost regression model predicts VM cost with an R^2 of about 0.9998, and K-Means clustering groups usage into four patterns: steady, idle with low cost, idle with high cost, and overloaded. The framework also provides a Cloud Health Score, AI-based recommendations, estimated savings, budget alerts, priority ranking, and a carbon-footprint estimate, all presented through an interactive 12-page Streamlit dashboard. Because the dataset is synthetic, the results show the feasibility of the approach; validation on real cloud data is planned as future work.**

***Index Terms*—multi-cloud computing, cloud cost optimization, machine learning, resource optimization, cloud sustainability**

I. Introduction

A. Background

Cloud computing is now at the center of how most organizations run their IT infrastructure. Companies no longer need to purchase and look after their own physical servers; they can rent computing power and adjust it as demand changes. Yet this flexibility has a drawback: many organizations are billed for cloud servers, known as virtual machines (VMs), that sit idle or are bigger than their work requires [1]. The problem grows once a company uses more than one cloud provider, because following usage on Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) at the same time is not easy.

Cloud billing is usage-based, so every unused or oversized VM adds to the monthly bill without delivering any value. Idle resources also keep consuming electricity in the data center, which links cost waste directly to energy use and carbon emissions. Reducing waste therefore benefits both the budget and the environment.

Each cloud provider offers its own tools for this purpose, for example Cost Explorer on AWS, Cost Management on Azure, and the Recommender on Google Cloud, and these work reasonably well. The issue is that each tool is built to look only at its own platform. Therefore, if a team uses two or three clouds together, which is common today, there is no single place to see the full picture. Spotting waste or planning a budget then becomes guesswork rather than something backed by data. What is missing is a lightweight tool that brings all of this together in one view.

Machine learning offers a way to close this gap. By learning patterns from historical resource data, it can classify VMs by their condition, estimate costs, and group VMs with similar usage, which supports more adaptive resource management than fixed thresholds and manual monitoring [5], [6].

B. Problem Statement

Organizations running workloads on more than one cloud lack a single, easy-to-use system that can (i) detect idle or wrongly sized VMs across providers, (ii) predict the cost of each VM, (iii) group VMs by how they are actually used, and (iv) turn these findings into clear, prioritized actions for non-expert users. Manual monitoring is slow and error-prone, and provider-specific tools cannot give a combined multi-cloud view.

C. Objectives

This work aims to:

- build a machine-learning pipeline that recommends whether a VM should be scaled up, scaled down, or left unchanged;
- predict VM cost accurately from usage and configuration data;
- discover and label usage patterns so that results are easy to act on;
- provide supporting insights, namely a Cloud Health Score, estimated savings, budget alerts, priority ranking, and a carbon-footprint estimate; and
- deliver all of the above through one interactive dashboard covering AWS, Azure, and GCP.

D. Contributions

This paper makes four contributions. First, we present CloudCost AI, a unified framework that combines classification (Random Forest), regression (XGBoost), and clustering (K-Means) for multi-cloud VM optimization. Second, we describe a labelled usage-pattern scheme (Steady, Idle–Low Cost, Idle–High Cost, and Overloaded) that converts clusters into actionable categories. Third, we add decision-support modules (health score, savings estimation, budget alerts, priority ranking, and carbon-footprint estimation) on top of the model outputs. Fourth, we implement a 12-page interactive Streamlit dashboard, including Explainable AI (SHAP) and What-if Optimizer pages, and report its results on a synthetic multi-cloud dataset.

E. Paper Organization

The remaining sections are arranged as shown here. Section II covers earlier studies, Section III presents the method, Section IV explains how the system was built,

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

Section V analyzes the findings, and Section VI closes with conclusions and future directions.

II. Related Work

A. Background Concepts

Four ideas are central to this work. *Over-provisioning* means that a VM has more CPU, memory, or storage than its workload needs, which wastes money. *Under-provisioning* means the opposite: the resources are too small, which can increase response time, reduce throughput, and hurt the application [5], [6]. *Right-sizing* is the process of matching a VM's size to its real usage, and it is a widely used way to bring down cloud spending [1]. *Idle resources* are VMs that stay switched on but do very little work; they still incur charges and consume energy.

Traditional cost management relies on manual monitoring, fixed thresholds, and periodic reviews. These methods can find some waste, but they respond slowly when workloads change. For example, a VM that is underused in one period may need more resources during a sudden rise in demand, and reducing resources without checking memory, latency, or workload variability can harm performance. Data-driven methods that learn from usage history are therefore attractive, and they are the basis of the studies discussed next.

B. Existing Studies

A fair amount of work already exists on cloud cost and resource optimization using machine learning, and it helped shape our approach.

Aldossary and Djemame [2] built a framework that predicts the total cost of VM auto-scaling by combining resource usage with power-consumption data. Tested on a cloud testbed, it reduced costs by up to 25% compared with earlier approaches. It is a solid piece of work; however, it stops at cost prediction, with no clustering of usage patterns and no dashboard through which users can interact with the results.

Closer to our work, Bhaskaran [3], in an M.Sc. thesis, built an AI-driven framework that brings together cost prediction, task scheduling, and fault detection in one visualization dashboard, using Linear Regression, Random Forest, and XGBoost on synthetic workloads. We found this to be the closest match to our approach, as both use similar models and a dashboard-first design. We go further by adding K-Means clustering of usage patterns, a Cloud Health Score, a carbon-footprint estimate, and a single framework designed to cover AWS, Azure, and GCP together.

Canali and Lancellotti [4] took a different angle, proposing an automated way to cluster VMs purely by correlations in their resource usage, without needing to know which service each VM is running. They tested this on a real enterprise data center, which adds credibility. We borrow the same general idea in our K-Means step, but we go a step further by giving each cluster a clear, practical label (Steady, Idle–Low Cost, Idle–High Cost, and Overloaded) so that the output is something a user can act on immediately.

Deochake [1] presented a wide-ranging review of cloud cost optimization strategies, covering pricing models and resource allocation techniques, backed by real case studies. It is useful background reading and confirms how significant the problem is, but it does not build or test a predictive system. Machine learning is also widely used for cloud resource management in general; the surveys in [5] and [6] review how learning-based methods are applied to tasks such as workload prediction, scheduling, and resource allocation, and they highlight the need for practical, deployable tools.

Table I compares the features reported for the cited works with CloudCost AI.

TABLE I
Feature Comparison with Related Work

Work	Cost pred.	Cluster-ing	Dash-board	Health / carbon
[2] Aldossary	Yes	–	–	–
[3] Bhaskaran	Yes	–	Yes	–
[4] Canali	–	Yes	–	–
[1] Deochake	Review	–	–	–
CloudCost AI	Yes	Yes	Yes	Yes

"Yes" means the feature is described in the cited work; "–" means it is not reported there. [1] is a review and does not build a system.

The results of the cited studies also depend on their datasets, evaluation methods, and deployment environments, so an improvement reported in one study cannot automatically be assumed in another. Our work is therefore presented as a prototype evaluated on simulated data, and we do not claim better results than the earlier studies; our contribution is the integrated design.

Taken together, most existing work tackles only one piece of the puzzle (cost prediction, clustering, or a general survey) rather than bringing prediction, clustering, health scoring, and multi-cloud support into a single, easy-to-use framework. This gap motivated us to build CloudCost AI.

III. Proposed System and Methodology

A. Dataset

Since we did not have access to live cloud billing APIs, we built a synthetic dataset designed to look and behave like real multi-cloud usage data. It contains 28,800 records and 18 columns covering 60 VMs monitored every 15 minutes across AWS, Microsoft Azure, and GCP. Each record captures VM configuration, CPU and memory utilization, network and disk I/O, latency, throughput, workload pattern, pricing, and cost. This gives 480 records per VM, which is about five days of monitoring. Table II summarizes the dataset.

TABLE II
Summary of the Synthetic Dataset

Property	Value
Total records	28,800
Columns	18
Virtual machines	60
Sampling interval	15 minutes
Records per VM	480 (about 5 days)
Cloud providers	AWS, Azure, GCP
Target labels	scale_up, scale_down, no_action

The 18 columns can be grouped into the categories shown in Table III. These attributes describe both the configuration of a VM and its observed behavior. CPU and memory usage show how much of the allocated capacity is used, network and disk activity describe the workload, latency and throughput represent performance, and the pricing and cost fields support the analysis of spending. The workload-pattern field records usage styles such as idle and steady. Each record also carries a pre-assigned target label (scale_up, scale_down, or no_action), which is used to train and test the Random Forest classifier.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

TABLE III
ATTRIBUTE GROUPS IN THE DATASET

Group	Examples
Identity	VM ID, cloud provider, region, VM type
Configuration	vCPU count, RAM (GB)
Utilization	CPU, memory, overall utilization
Network and disk	network I/O, disk I/O
Performance	latency, throughput
Workload	workload pattern (e.g., steady, idle)
Pricing and cost	price per hour, usage cost
Target	scaling action label

B. System Architecture

At a high level, the system works in a simple sequence. Raw VM data goes through cleaning and preprocessing and is then passed to the three machine-learning models, and the outputs are merged into one final report. This report powers the dashboard, so users can see predictions, recommendations, and analytics in one place rather than moving between tools. Fig. 1 shows the overall architecture, which has five blocks:

- *Data Layer:* the synthetic multi-cloud dataset (28,800 records, 60 VMs, 15-minute intervals) from AWS, Azure, and GCP.
- *Data Preprocessing:* data cleaning (missing values and noise), feature engineering (cost, usage, and metadata), normalization (feature scaling), and an 80%/20% train–test split.
- *Machine-Learning Models:* Random Forest for scaling action and health scoring, XGBoost for cost prediction, and K-Means for usage patterns.
- *Optimization Modules:* Health Score, Recommendations, Cost Savings, Budget Alerts, Priority Ranking, and Carbon Footprint.
- *Streamlit Dashboard:* interactive visualizations, real-time insights, and a combined view of AWS, Azure, and GCP.

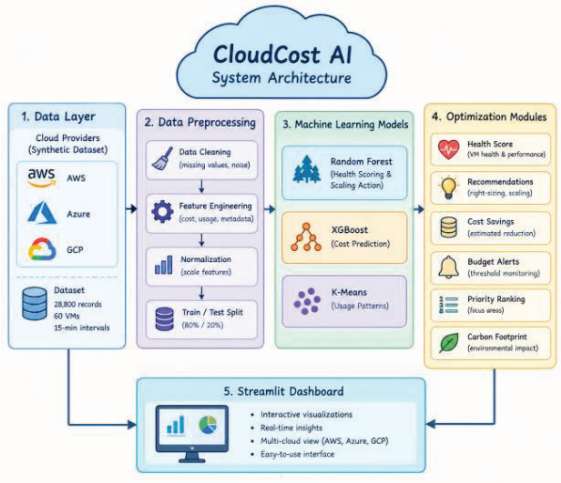


Fig. 1. System architecture of CloudCost AI.

Algorithm 1 summarizes the complete workflow in plain steps.

Algorithm 1: CloudCost AI workflow

1. Load the multi-cloud VM dataset (28,800 records, 60 VMs).
2. Clean the data, create features, encode categories, and normalize numeric values.
3. Split the data into 80% training and 20% testing sets.
4. Train Random Forest (scaling action) and XGBoost (cost); apply K-Means with K = 4.
5. Evaluate: accuracy for Random Forest, R^2 for XGBoost.
6. Compute the health score, savings, priority, budget status, and carbon score.
7. Merge all outputs into one final report (24 columns per record).
8. Show the report through the Streamlit dashboard pages.

The architecture deliberately separates data preparation, model execution, and result presentation. This separation makes each part easier to understand and test, and it provides a base for future improvements such as updated datasets, additional evaluation metrics, and integration with real cloud monitoring services.

C. Data Preprocessing

Raw telemetry rarely arrives in a form that models can use directly. In CloudCost AI, the preprocessing stage first cleans the data by handling missing values and noise. Feature engineering then derives cost, usage, and metadata features from the raw columns, for example an overall utilization value that combines CPU and memory usage. Numeric features are normalized so that attributes with large ranges (such as network I/O) do not dominate attributes with small ranges (such as utilization percentages), and categorical fields such as provider, region, VM type, and workload pattern are converted into numeric form. Before this step the dataset is inspected for structure, data types, missing values, duplicate records, and the distribution of key attributes; the prepared version contains 28,800 records with no missing values and no duplicates. Timestamp and VM identifier fields are excluded from the model features because they are not predictive inputs. Finally, the data are split into 80% for training and 20% for testing, so that every model is evaluated on records it did not see during training. The classification target is the scaling-action label, and the cost model uses the cost column as its target.

D. Machine-Learning Models

Three models are used, each with a different job. Together they answer three practical questions: what should be done with a VM, how much will it cost, and how is it being used?

1) Random Forest Classifier: A Random Forest builds many decision trees on random samples of the data and combines their votes, which makes it robust to noise and less prone to overfitting than a single tree. In CloudCost AI it predicts whether a VM should be scaled up, scaled down, or left unchanged (no action), based on usage patterns. The predicted action is the class chosen by most of the trees. The model achieved an accuracy of around 94.6%.

2) XGBoost Regression: XGBoost is a gradient-boosted tree method in which each new tree corrects the errors of the earlier ones. It handles non-linear relationships between usage, configuration, and cost well. In CloudCost AI it estimates what a VM will cost, using how the VM is

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

used and how it is set up. The model achieved an R^2 of around 0.9998.

3) K-Means Clustering: K-Means is an unsupervised method that groups similar records without using labels. It assigns each record to the nearest of K cluster centers and updates the centers until the groups stop changing. We use K = 4 to group VMs into four usage categories, summarized in Table IV.

TABLE IV
USAGE PATTERN CATEGORIES

Category	Meaning
Steady	stable usage in a healthy range
Idle–Low Cost	very low usage and low spending
Idle–High Cost	very low usage but high spending
Overloaded	usage so high that performance may suffer

E. Evaluation Metrics

Classification performance is measured by accuracy, the share of correctly predicted actions among all predictions. Regression performance is measured by the coefficient of determination (R^2), which shows how closely the predicted costs match the actual costs. An R^2 close to 1 means the predictions explain almost all of the variation in cost.

F. Decision-Support Modules

On top of the three models, five modules turn raw predictions into information that a user can act on:

- *Cloud Health Score:* an overall score (out of 100) that summarizes how well each VM's resources are being used.
- *AI Recommendations:* plain-language advice derived from the model outputs, for example that an underutilized VM should be downsized.
- *Estimated Savings:* the savings that can be achieved if the recommendations are implemented.
- *Budget Alerts:* alerts raised whenever spending is likely to exceed a user-defined budget; warning and critical thresholds can be set by the user.
- *Priority Ranking:* a ranking of VMs by the urgency of the action required, so that attention goes first to the most important resources.

In addition, a Carbon Footprint module estimates the CO_2 impact of each resource from its CPU usage and the carbon intensity of its region, and a What-if Optimizer lets users change VM settings and immediately see the predicted cost and recommended action.

Table V summarizes how each module is used in the system.

TABLE V
SUMMARY OF DECISION-SUPPORT FEATURES

Feature	What the user gets
Cloud Health Score	score out of 100 and a status for each VM
AI Recommendations	action, priority, and a written explanation
Estimated Savings	possible reduction in cost
Budget Alerts	safe, warning, or critical budget status
Priority Ranking	order in which VMs should be handled
Carbon Footprint	carbon score and estimated CO_2 impact
What-if Optimizer	new cost and action for changed settings
Explainable AI	features that raise or lower a predicted cost

G. Interpretation of Model Outputs

The Random Forest output is one of three resource-management classes. Table VI explains what each class means for a VM. A scale-down prediction suggests that a VM has more capacity than its workload needs, a scale-up prediction suggests that more resources should be considered, and no action means that the learned patterns do not support either change.

TABLE VI
MEANING OF THE PREDICTED ACTIONS

Action	Meaning
scale_down	capacity is higher than the workload needs
scale_up	more resources should be evaluated
no_action	current configuration appears suitable

These outputs are decision-support recommendations. Before a resource change is applied in a real environment, administrators should consider workload requirements, service-level objectives, availability, and the possible effects of resizing. The system does not modify live cloud resources. Cluster membership alone also does not decide whether a VM should be resized; it provides additional context. For example, a group with low utilization deserves a closer look for over-provisioning, whereas a group with high utilization needs closer performance monitoring.

The reliability of the predicted cost depends on the quality and representativeness of the training data. Because the current dataset is simulated, the cost predictions should be read as experimental results within that dataset and not as estimates of real cloud invoices.

IV. IMPLEMENTATION

A. Development Environment

Model development and data processing were carried out in Google Colab, with the data stored in Google Drive. The Streamlit dashboard was developed and tested locally in Visual Studio Code, with a main application file and a separate page file for each dashboard page, connected through a sidebar navigation menu. All pages read from one final report file, which merges the model outputs and the decision-support results into 24 columns per record, so that every page shows consistent numbers.

B. Tools and Technologies

The entire system was written in Python. The tools used are listed in Table VII. Streamlit was chosen mainly because it allowed us to turn our Python scripts quickly into an interactive web application without building a separate front end.

TABLE VII
TOOLS AND TECHNOLOGIES USED

Purpose	Tool
Programming language	Python
Data handling	Pandas, NumPy
Machine learning	scikit-learn, XGBoost
Explainability	SHAP
Visualization	Matplotlib, Plotly
Dashboard	Streamlit
Model development	Google Colab

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

C. Development Workflow

The project followed a step-by-step pipeline: data understanding, exploratory data analysis (EDA), preprocessing, Random Forest training, XGBoost training, K-Means clustering, health scoring, dashboard development, and testing. The backend (models, health score, recommendations, budget alerts, savings estimation, priority ranking, and carbon-footprint score) was completed and verified first. The dashboard pages were then built on top of the verified outputs and checked by comparing the numbers shown on each page with the final report.

D. Dashboard Design

The dashboard has nine core pages: Dashboard Overview, Resources, Idle Resource Detection, Cost Prediction, Usage Patterns, AI Recommendations, Cloud Health Score, Budget Alerts, and Reports and Analytics. Three more advanced pages were added once the core system was working: an Explainable AI page, a What-if Optimizer, and a Carbon Footprint page. A login page controls access to the dashboard. Table VIII lists the pages and the model or module that powers each one.

TABLE VIII
DASHBOARD PAGES AND THEIR SOURCES

Page	Source
Dashboard Overview	all outputs
Resources	dataset
Idle Resource Detection	Random Forest
Cost Prediction	XGBoost
Usage Patterns	K-Means
AI Recommendations	model outputs
Cloud Health Score	health score
Budget Alerts	cost and budget
Reports and Analytics	final report
Explainable AI	SHAP on XGBoost
What-if Optimizer	XGBoost + Random Forest
Carbon Footprint	CPU usage and region

For usability, all monetary values on the dashboard are shown in Indian rupees (converted from US dollars at a rate of 83 rupees per dollar), and timestamps are shown in Indian Standard Time. The interface uses a dark theme with card-style summaries, expandable sections, and charts on every page so that results can be read quickly. Each page focuses on one aspect of cloud resource monitoring, so users can explore the outputs without reading any code. The pages are described below.

1) Dashboard Overview and Resources: The Dashboard Overview gives a consolidated summary of the cloud environment: the number of monitored VMs, estimated cost, average CPU and memory usage, and how many VMs are healthy, in warning state, critical, or candidates for scaling down (Fig. 2). The Resources page lists the available VM records with their configuration.

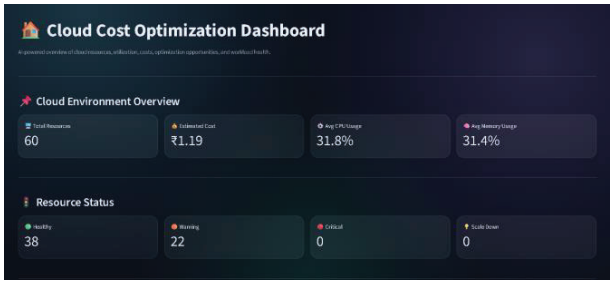


Fig. 2. Dashboard Overview.

2) Budget Alerts and Reports and Analytics: Budget Alerts compares the configured budget with the estimated spending and raises a warning or critical alert when the user-defined thresholds are reached (Fig. 3). Reports and Analytics collects the results in one place for further examination of usage and cost.



Fig. 3. Cloud Budget Alert.

3) Idle Resource Detection, Cost Prediction, and Usage Patterns: Idle Resource Detection uses the Random Forest model to flag VMs that may be over-provisioned or underused. Cost Prediction shows the costs estimated by XGBoost so that users can relate usage to expenditure. Usage Patterns shows the K-Means groups so that different utilization behaviors can be compared (Fig. 4).

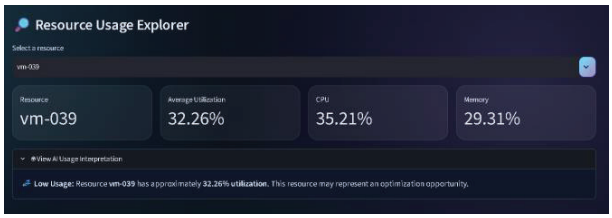


Fig. 4. Usage Pattern (K-Means).

4) AI Recommendations and Cloud Health Score: AI Recommendations lists, for each VM, the suggested action, its priority, and a plain-language explanation. Cloud Health Score summarizes the condition of each resource as a score out of 100 with a status, so that VMs needing attention are easy to spot (Fig. 5).



Fig. 5. Cloud Health Score: (a) score overview and (b) detailed breakdown.

5) Explainable AI: This page uses SHAP (SHapley Additive exPlanations) to show how each input feature pushes a predicted cost up or down, which makes the model easier to trust (Fig. 6).

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

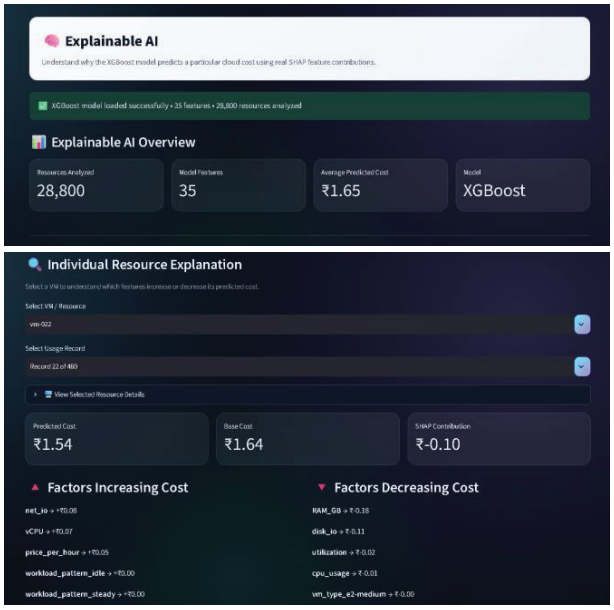


Fig. 6. Explainable AI (SHAP): (a) prediction explanation and (b) feature importance.

6) What-if Optimizer: Users move sliders to change VM settings such as vCPU, RAM, utilization, and price and immediately see the predicted cost and recommended action. This lets them test alternative configurations before making a real decision (Fig. 7).

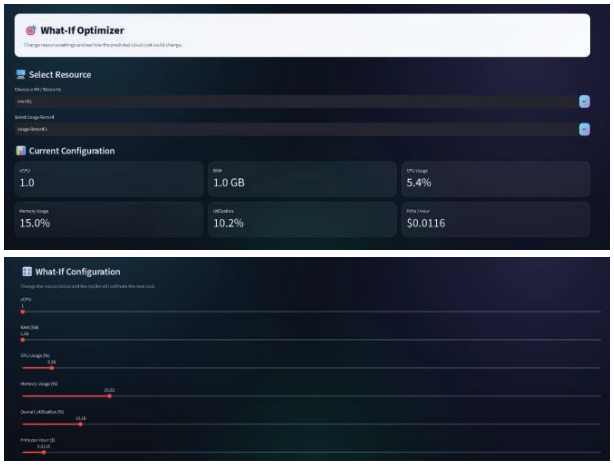


Fig. 7. What-if Optimizer page: (a) VM settings input and (b) predicted cost and recommended action.

7) Carbon Footprint: This page estimates the carbon impact of resource usage from CPU usage and region-related information and highlights resources with higher impact, adding an environmental view to cost optimization (Fig. 8).

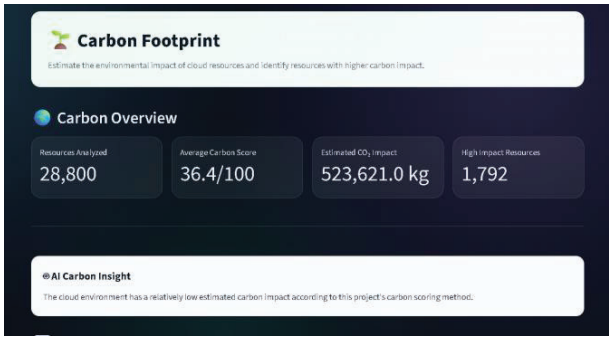


Fig. 8. Carbon Footprint page.

E. Testing and Validation

Testing followed the same order as development. First, the backend was run from start to finish in a clean notebook, and it produced a complete final report with 24 columns per record, which holds the model outputs, health score, recommendations, savings, priority, budget status, and carbon values. Second, each dashboard page was opened and its numbers were compared with the final report to confirm that the page and the report agree. Third, the Random Forest model was run in two separate environments (Google Colab and a local Visual Studio Code setup) by different team members, and both runs gave near-identical results with only tiny natural variation, which is expected for a model that uses random sampling.

Interactive elements were also tested by changing inputs and checking the outputs. Moving the sliders of the What-if Optimizer changes the predicted cost and the recommended action, changing the budget or thresholds on the Budget Alerts page changes the alert status, and selecting a different VM updates the health score, recommendation, and SHAP explanation.

F. Typical User Workflow

A typical session with the dashboard has five steps. The user first signs in and opens the Dashboard Overview to see how many VMs are healthy, in warning state, or critical. Next, the user checks Idle Resource Detection and Usage Patterns to find underused or overloaded VMs. The user then reads the AI Recommendations and Priority Ranking to decide where to act first and uses the Cost Prediction and What-if Optimizer pages to compare options. Budget Alerts show whether spending stays within the chosen limit. Finally, Reports and Analytics and the Carbon Footprint page summarize cost and environmental impact for review.

V. Results and Discussion

A. Experimental Setup

All experiments were run on the synthetic dataset described in Section III. The data were split into 80% for training and 20% for testing. The Random Forest classifier was evaluated by accuracy on the test set, and the XGBoost regressor by the R^2 score between predicted and actual cost. K-Means was applied with four clusters, and each cluster was given a descriptive label so that the result can be read directly. The models were developed with scikit-learn and XGBoost in Python, and every result in this section comes from the same final run that feeds the dashboard.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

B. Model Performance

On the classification side, the Random Forest model reached approximately 94.6% accuracy in predicting whether a VM should be scaled up, scaled down, or left unchanged. For cost prediction, the XGBoost model performed very well, with an R^2 of approximately 0.9998, indicating that its predictions align closely with the actual costs in our dataset. The K-Means step split the VMs into four clear usage groups, labelled Steady, Idle with Low Cost, Idle with High Cost, and Overloaded. Table IX and Fig. 9 summarize the model performance.

TABLE IX
PERFORMANCE OF THE MACHINE-LEARNING MODELS

Model	Task	Result
Random Forest	Scaling decision	94.6% accuracy
XGBoost	Cost prediction	$R^2 \approx 0.9998$
K-Means	Usage patterns	4 clusters

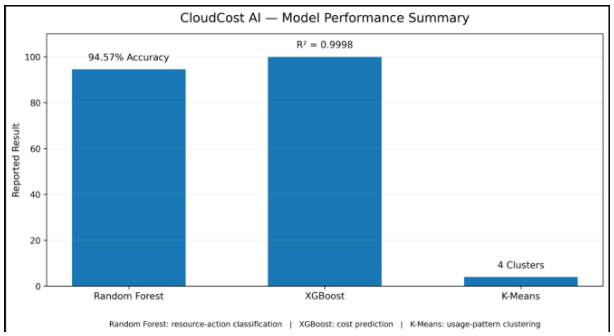


Fig. 9. Model performance summary of CloudCost AI.

C. Dashboard Findings

The dashboard turns the model outputs into readable findings. In the Dashboard Overview (Fig. 2), the 60 monitored VMs show an average CPU usage of 31.8% and an average memory usage of 31.4%. Of the 60 VMs, 38 are classified as Healthy and 22 as Warning, with none in the Critical state. Table X lists the main indicators. Average utilization of about one-third indicates that, in this dataset, a large share of the provisioned capacity is not being used, which is exactly the situation in which scaling down can reduce cost.

TABLE X
KEY INDICATORS SHOWN ON THE DASHBOARD

Indicator	Value
Monitored VMs	60
Average CPU usage	31.8%
Average memory usage	31.4%
Healthy VMs	38
Warning VMs	22
Critical VMs	0
Average carbon score	36.4 / 100

The Budget Alerts page (Fig. 3) lets a user set a monthly budget together with warning and critical thresholds, and then shows current spending, remaining budget, and the percentage used, along with a written insight. The Resource Usage Explorer (Fig. 4) shows, for a selected VM, its average, CPU, and memory utilization with a short interpretation; for example, a VM at about 32% utilization is flagged as a possible optimization opportunity.

Table XI collects a few VM-level examples taken from the dashboard pages to show the kind of output a user sees.

TABLE XI
EXAMPLES OF VM-LEVEL DASHBOARD OUTPUTS

VM	Observed output
vm-003	GCP, asia-south1, n1-standard-2; 11.49% average utilization; scale_down; Medium priority; health 62.7
vm-039	35.21% CPU, 29.31% memory, 32.26% average utilization; flagged as low usage
vm-045	health score 85/100, status Healthy; 28.93% average utilization
vm-022	predicted cost Rs 1.54 against a base cost of Rs 1.64 (SHAP)

The Cloud Health Score page (Fig. 5) gives each VM a score out of 100 and a status; for example, a VM with a score of 85/100 and 28.9% average utilization is labelled Healthy. The AI Recommendations page (Fig. 10) lists, for each VM, the recommended action, priority, and health score, together with a plain-language explanation. For instance, a GCP VM with 11.5% average utilization receives a scale-down recommendation with Medium priority, because it appears underutilized.

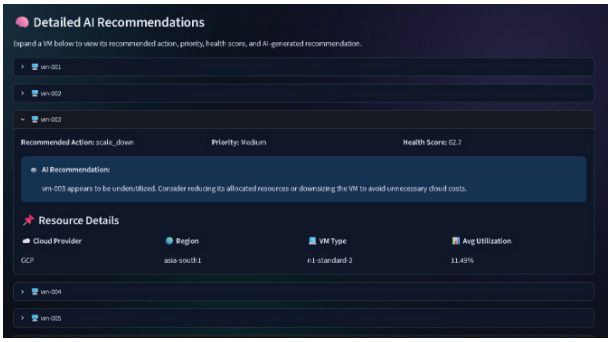


Fig. 10. AI-based VM recommendation and resource optimization.

The Explainable AI page (Fig. 6) uses SHAP to show why the cost model predicts a particular value. For a selected usage record, the page lists a base cost, the predicted cost, and the features that push the cost up (for example, network I/O, vCPU count, and price per hour) or down (for example, RAM and disk I/O). This helps users trust the prediction instead of treating the model as a black box. The What-if Optimizer (Fig. 7) lets users change vCPU, RAM, utilization, and price values and immediately see the estimated cost and recommended action.

Finally, the Carbon Footprint page (Fig. 8) summarizes the environmental side of the analysis. For the 28,800 usage records, it reports an average carbon score of 36.4/100, an estimated CO_2 impact of about 523,621 kg, and 1,792 high-impact records, and it concludes that the estimated impact is relatively low under the project's carbon-scoring method. These values come from the project's own scoring method applied to synthetic data and should be read as relative indicators, not as measured emissions.

D. Analysis of Usage Patterns

The four K-Means groups describe how VMs behave over time. Steady VMs show stable usage and need no urgent attention. Idle VMs with low cost waste little money, but they still occupy resources and may be candidates for shutdown or downsizing. Idle VMs with high cost are the most valuable to find, because the same low usage is combined with high spending, so correct sizing gives the largest savings. Overloaded VMs run close to their limits and may need more capacity to avoid slow responses. Labeling the clusters in this

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 11, CONFLUENCE - 2026

way turns an abstract grouping into a statement that a non-expert user can act on.

The two supervised models complement the clusters. The Random Forest output tells the user what action is suggested for each record, while the XGBoost output shows how much each VM is expected to cost. When these are read together with the cluster label, a user can, for example, see that a VM is idle, expensive, and recommended for scale-down, which is a strong reason to act first on that VM.

E. Discussion

Using the Cloud Health Score and the savings estimates, these results give users a quick way to identify which VMs are wasting money and what to do about it. Compared with provider-specific tools, CloudCost AI offers one interface for AWS, Azure, and GCP, adds usage-pattern labels that are easy to understand, and links cost to sustainability through the carbon-footprint estimate. Compared with the related work in Section II, it combines prediction, clustering, scoring, explainability, and a what-if simulator in a single framework.

F. Limitations

These findings have practical value for three groups of users. Small teams without a dedicated cloud-cost analyst can get a clear list of what to fix first. Managers can use the budget alerts and savings estimates to plan spending. Sustainability-minded teams can see which resources carry the highest estimated carbon impact. In all cases the system supports a human decision; it does not act on its own.

We want to be clear about the limits of this study. First, the dataset is synthetic; therefore, these numbers show that the approach works, not how it would perform on noisy real-world data. A result as high as 0.9998 is fairly typical on clean synthetic data and would likely decrease somewhat once the system is tested against real cloud usage. Second, the data cover only about five days of monitoring per VM, so long-term and seasonal patterns are not captured. Third, accuracy alone can hide differences between classes, so a detailed per-class analysis would give a fuller picture on real data. Fourth, the carbon-footprint figures are estimates based on CPU usage and regional carbon intensity rather than measured energy use. Fifth, because the system does not connect to live cloud billing APIs, it cannot yet apply its recommendations automatically.

VI. Conclusion and Future Work

In this study, we built CloudCost AI, a machine-learning framework that brings together scaling prediction, cost prediction, usage clustering, and several supporting features in a single dashboard for multi-cloud cost and resource optimization. On our synthetic dataset, it performed well, with 94.6% accuracy for scaling decisions and an R^2 of 0.9998 for cost prediction. The main feature that sets it apart from existing single-cloud tools is that it is designed from the ground up to work across AWS, Azure, and GCP together through one interface.

The work shows that three standard machine-learning methods, each with a clear job, can be combined with simple decision-support modules to produce advice that a non-expert can follow. Classification tells the user what to do, regression tells the user what it will cost, and clustering explains how each VM is used. Explainable AI and the What-if Optimizer add trust and the ability to test options, while the carbon-footprint page links cost savings to a lower environmental impact. All results come from a simulated dataset, so they should be read as evidence that the approach is feasible and not as a promise of performance in production.

In future work, we plan to:

- test the system on real cloud usage data, and compare it with provider tools, to see how it performs outside a synthetic setting;
- connect the dashboard to live cloud billing and monitoring APIs so that data update automatically;
- extend the What-if Optimizer and Explainable AI modules with more scenarios;
- add resource-usage forecasting so that scaling decisions can be made before problems occur; and
- report a more detailed evaluation, including per-class results, on larger and longer datasets.

References

[1] S. Deochake, “Cloud cost optimization: A comprehensive review of strategies and case studies,” arXiv:2307.12479, 2023.

[2] M. Aldossary and K. Djemame, “Performance and energy-based cost prediction of virtual machines auto-scaling in clouds,” in *Proc. 44th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Prague, Czech Republic, 2018, pp. 502–509, doi: 10.1109/SEAA.2018.00086.

[3] R. Bhaskaran, “AI-driven cloud optimization: Enhancing cost prediction, resource scheduling and fault resilience in cloud environments,” M.Sc. thesis, School of Computing, National College of Ireland, Dublin, Ireland, 2025.

[4] C. Canali and R. Lancellotti, “Automated clustering of virtual machines based on correlation of resource usage,” *J. Commun. Softw. Syst.*, vol. 8, no. 4, pp. 102–109, Dec. 2012, doi: 10.24138/jcomss.v8i4.164.

[5] S. Goodarzy, M. Nazari, R. Han, E. Keller, and E. Rozner, “Resource management in cloud computing using machine learning: A survey,” in *Proc. 19th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, 2020, pp. 811–816, doi: 10.1109/ICMLA51294.2020.00132.

[6] T. Khan, W. Tian, G. Zhou, S. Ilager, M. Gong, and R. Buyya, “Machine learning (ML)-centric resource management in cloud computing: A review and future directions,” *J. Netw. Comput. Appl.*, vol. 204, Art. no. 103405, 2022, doi: 10.1016/j.jnca.2022.103405.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)