National College of Ireland

# Hybrid Machine Learning for Anomaly Detection in Cloud Cost Optimization

MSc Research Project
Cloud Computing


SRI HARI REDDY KATA
Student ID: 24155331

School of Computing
National College of Ireland

Supervisor: Aqeel Kazmi

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | SRI HARI REDDY KATA |
| **Student ID:** | 24155331 |
| **Programme:** | Cloud Computing |
| **Year:** | 2025 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Aqeel Kazmi |
| **Submission Due Date:** | 11/12/2025 |
| **Project Title:** | Hybrid Machine Learning for Anomaly Detection in Cloud Cost Optimization |
| **Word Count:** | XXX |
| **Page Count:** | 24 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

| | |
|---|---|
| **Signature:** | |
| **Date:** | 27th January 2026 |

## PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Hybrid Machine Learning for Anomaly Detection in Cloud Cost Optimization

SRI HARI REDDY KATA
24155331

**Abstract**

Cloud computing is flexible but wastes a lot of money, about 30% of the operational costs are going either unused or by over-allocation of resources. This research paper proposed a novel two-stage hybrid anomaly-detection pipeline that combines an unsupervised Isolation Forest Algorithm as the first stage to capture anomaly scores from unlabeled CloudWatch metrics, followed by a Stacking Ensemble classifier that synthesizes predictions from Random Forest, Gradient Boosting, and Logistic Regression. The system leverages temporal features (day, month, year) alongside resource utilization metrics such as CPU, memory, network traffic, disk I/O to achieve 76.25% accuracy with 74.45% sensitivity and 78.05% specificity, demonstrating balanced performance across normal and anomalous instances.A key contribution is the severity classification with values such as Low, Medium, High, and Critical that prioritises alerts based on financial impact and confidence scores, integrated with AWS Simple Notification Service for automated email notifications enabling immediate mitigation actions. We use accuracy, precision, recall, F1 score, sensitivity, specificity, and confusion matrix to evaluate our results. Results indicate that the hybrid model, especially the stacked ensemble model, improves accuracy of detection and reliability when comparing with individual supervised models. In future work, we will explore the integration of real time ingestion from AWS CloudWatch and temporal modeling to detect changes in cloud usage over time.

## 1 Introduction

***A. Research Motivation:*** With an increasing number of companies migrating their software to cloud infrastructures comes a need to properly monitor cloud services. Organizations are expanding their applications across a range of distributed and dynamic cloud environments with increasing amounts of variability in how these applications use available resources. The variability of application utilization creates an ongoing problem as cloud resources are frequently configured incorrectly, used inefficiently, or provisioned excessively Figure 1, which leads to considerable expense waste. Industry research consistently demonstrates that 25 – 40 % of total expenses associated with cloud computing are unnecessary, due to the fact that many of the compute instances are idle, many of the resource configurations are too large, and anomalies in usage behavior are frequently not detected until it has resulted in costly wasteAvocado (2024).

Although, AWS CloudWatch has an Anomaly Detection feature, it is essentially an black-box algorithm and can therefore not be customized or trained for organization specific workloads, resulting in lower effectiveness for many types of complex cost-

optimization problem scenarios. In response to these issues, researchers have increasingly looked into hybrid approaches that use unsupervised discovery combined with supervised refinement. Therefore, this research proposal suggests a lightweight hybrid approach in which a combination of Isolation Forest will initially detect unusual patterns in cpu, memory, and network metrics, and then utilize several supervised models, including Random Forest, Logistic Regression, KNN, Gradient Boosting, LightGBM, and a Stacking ensemble to refine those initial detections and compare the suitability of each supervised model for cost focused anomaly classification. The proposed hybrid model will operate in a batch processing mode using publicly available datasets and synthetically created anomalies and will be designed to accommodate integration with real-time AWS CloudWatch ingestions in the future.

***B. Problem Statement:*** The subtlety and gradualness in cloud cost anomalies make them hard to find with generic provider tools. The main problems with current anomaly detection systems is that they have three key disadvantages: A lack of tunability and very little transparency, and finally, there are significant issues around the quality of the data used for training machine learning models this reduces the reliability of most traditional machine learning techniques.

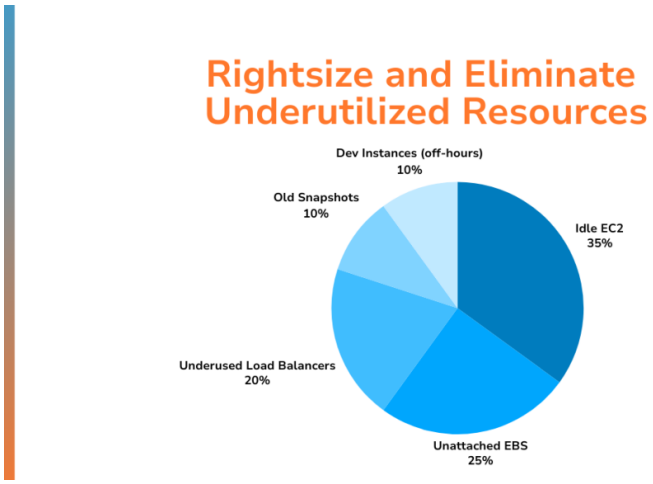



Figure 1: Estimated Breakdown of AWS Cost Inefficiencies

### *C. Research Question:*

> ‘ *How can a hybrid machine learning model, combining lightweight unsupervised and supervised techniques, be applied to identify and mitigate inefficient resource usage in AWS cloud infrastructure in order to optimize costs?*

***D. Research Aim & Objectives:*** The aim of this research is to design, implement, and evaluate a hybrid anomaly detection pipeline tailored for cloud cost optimisation. To achieve this, the study pursues four objectives:

- Investigate existing anomaly detection methods and identify their limitations in cost-focused cloud environments.
- Design a hybrid architecture integrating Isolation Forest with multiple supervised refinement algorithms.

- Implement a batch-processing pipeline using public datasets and synthetic anomalies.
- Evaluate various supervised models including Random Forest, Logistic Regression, KNN, Gradient Boosting, LightGBM, and a Stacking ensemble to determine which model best complements the unsupervised stage.

***E. Significance of the Research:*** This research is important because it offers an anomaly detection framework that is lightweight, and transparent for use in cloud cost optimization; and as such addresses the deficiencies inherent within current provider managed cost monitoring toolsets. The framework integrates unsupervised methods for discovering anomalies with supervised methods for refining those discoveries, to reduce false positives while preserving sensitivity to minor cost inefficiencies, thereby providing a more reliable alternative to black box anomaly detection methods. Finally, through the comparative analysis of several supervised machine learning methods, the research provides empirical insight into how best to augment unsupervised anomaly detection in a cloud based cost optimization environment. Overall, the research contributes both methodologically and practically to organizations seeking to improve the accuracy and interpretability of their use of anomaly detection of cloud-based costs.

***F. Structure of the Report:*** The rest of the report is structured as follows. The Related Work section reviews literature in anomaly detection, cloud monitoring, and hybrid machine learning methods that inform the foundation of this study. The Methodology section describes the processes of data acquisition, data preprocessing, the design of the proposed hybrid workflow, and the performance metrics used for evaluation. The Design section outlines the system architecture and the design requirements of the proposed framework. The Implementation section describes the practical development of the hybrid model, including integration steps and execution logic. The Evaluation section presents the experimental results, compares the performance of the supervised models used in the refinement stage, and discusses key findings. Finally, the Conclusion section summarises the study and highlights potential directions for future research.

# 2 Related Work

Cloud computing is a rapidly growing area of research as an increase in complexity, dynamic nature, and size of cloud-based systems results in a need for anomaly detection (AD). Traditional methods for monitoring do not provide sufficient capability due to the sheer volume and dimensionality of telemetry data being generated in modern cloud based systems, along with the constant fluctuations in workload.

The remainder of this section will critically evaluate the literature on three primary ML approaches supervised, unsupervised, and deep learning used to date in cloud AD, and select additional relevant cloud-based cost optimization studies. Finally, the remainder of this section identifies the areas of identified research gaps that prompted the development of the light weight hybrid model of AD presented in this paper.

## 2.1 Challenges in Cloud Anomaly Detection

A significant number of recent systematic review articles have identified that while the use of Machine Learning (ML) technologies to monitor Cloud computing continues to be

an evolving area of research, it still faces many challenges related to complexity within both the systems being monitored and the data being collected from those systems. One of the most well-known challenges facing scalable Anomaly Detection (AD) techniques is the high dimensionality of the data, as well as its associated sparsity. For example, a study by Islam et al. Islam et al. (2025) used a real world Cloud Telemetry Data set that contained 39,365 observations and 117,448 features to demonstrate how the "curse of dimensionality" will negatively impact the effectiveness of traditional cluster based and distance based AD algorithms. Another challenge identified by Hrusto et al. Hrusto et al. (2025), is the reliance of many studies using Real World Operational Data (71%), but the majority of those studies used Heuristically Generated or Weakly Verified Labels (73%) to support their supervised learning methods. A third key challenge is non-stationarity in Cloud Computing environments. As Workload, System Configuration and Software Stack characteristics change over time, so does the definition of what constitutes normal behavior. To help mitigate this challenge, Wang et al. Wang et al. (2021) proposed a Self-Evolving AD (SEAD) Framework which demonstrated improvements in Sensitivity and Specificity for detecting anomalies in Cloud Computing environments; however the SEAD Framework relies upon continuous Human Validation to remain effective. Overall, the collective results of the studies presented here clearly show the challenges of creating reliable Labels for Data Sets, the limitations of Static Models for AD, and the challenges presented by large scale dynamic Cloud Telemetry Data Sets.

## 2.2 Benchmark Datasets and Data Issues

Anomaly detection in cloud computing is generally based upon the use of benchmarks like NAB, Exathlon and Microsoft Cloud Monitoring traces. Although useful in terms of providing an opportunity for reproducibility (for example with respect to a particular data set), they typically include one dimensional time series and suffer from being limited in their diversity and do not adequately represent the variability inherent in today's cloud workload. Exathlon added a level of dimensionality but was limited by the nature of its dataset to repeated streaming jobs and therefore did not enable generalization. Both Islam et al. Islam et al. (2025), Hrusto et al. Hrusto et al. (2025) emphasize the fact that most datasets do not have well defined labels for anomalies, thus restricting the ability to develop reliable supervised anomaly detection systems. These limitations lead to developing hybrid methods that can operate under either partial or unreliable labeling conditions.

## 2.3 Supervised Machine Learning Approaches

The supervised AD paradigm addresses an anomaly detection problem through the lens of binary or multi-class classification. There are many research papers that have demonstrated excellent results when labelled data exists. For example, Garg et al.Jayaweera et al. (2023) and Johnny  Johnny (2024) demonstrated models including Random Forest (RF), Support Vector Machines (SVM), and Decision Trees consistently outperformed threshold-based monitoring systems in detecting KPI and performance anomalies. In addition to being accurate at rates over 90% in multiple studies Johnny (2024); **?**, RF has also shown to be very interpretable and robust; however, its reliance on good quality labels can reduce the reliability of supervised models in rapidly changing cloud environments where new types of anomalies emerge constantly. Al-Jumaili and Bazzi Sharma

and Kumar (2020) found near-perfect accuracy (99.63%) for the XGBoost model using the NSL-KDD dataset; however, this study was based on synthetic intrusion datasets that do not model the complexity of resource inefficiencies nor cost anomalies. Overall, supervised models will continue to perform well in controlled settings but will encounter problems with label sparsity, concept drift, and the ability to detect unforeseen events.

## 2.4 Unsupervised and Deep Learning Approaches

Many unsupervised Anomaly Detection (AD) methods have been proposed because they can be used without labelled data; methods include Isolation Forest (IF), One-Class SVM, Clustering, Auto-Encoders etc. The use of IF was demonstrated to be very efficient at detecting anomalies from cloud telemetry as it uses a low-computationally-intensive method to detect anomalies Calheiros et al. (2017). Nätterdal and Olaussson NÄTTERDAL and OLAUSSON (2024) extended IF for processing stream-based CloudWatch metric data and achieved better results than several baseline methods in terms of both AUC and F1-score when using a combination of power transforms and ADWIN drift detection. However, this work is based upon one dataset and therefore may not represent all possible datasets.

Deep Learning (DL) architectures including LSTMs, GRUs and Auto-Encoder architectures are capable of capturing both temporal relationships between variables and relationships between different variables. Santosh Santosh (2024), found that GRU-BERT significantly improved (by 46.55%) anomaly classification accuracy compared to an LSTM architecture. Renshaw Renshaw (2024) developed a probabilistic variational framework which was able to achieve an AUC of 0.935. Although deep learning architectures have been shown to perform well in terms of anomaly detection, they also require large amounts of computation, hyperparameter tuning is typically complex, and therefore typically less suitable for lightweight production deployments.

## 2.5 Hybrid Machine Learning Models

Combining the ability of unsupervised learning to identify anomalies with the power of a supervised classifier to determine the type of anomaly is an attempt to lower false positives while maintaining sensitivity to previously unseen anomalies.

These studies collectively show that hybrid methods outperform purely supervised or unsupervised models, but many rely on deep learning components or computationally expensive training pipelines. This motivates a need for lighter hybrid systems that preserve effectiveness while remaining practical for cost-sensitive cloud operations.

## 2.6 Cloud Cost Optimization Studies

An increasing number of researchers study AD techniques specifically for resource optimization and cost effectiveness. Nawrocki and Smendowski Nawrocki and Smendowski (2024) apply a combination of an IF model and gradient boosting, as well as recurrent neural networks (RNNs) to optimize long-term forecasting of CPU usage by improving substantially the Root Mean Squared Error (RMSE). The work of Olaoye Olaoye (2025), and Jakku Jakku (2024) indicate the potential of AI in cost management, in the rightsizing of resources, and in budget forecasting based on anomalies; however both studies are theoretical and have no experimental data. Furthermore, while Johnny Johnny (2024)

Table 1: Summary of key hybrid anomaly detection models in cloud computing

| Author | Approach | Key Findings | Limitations |
|---|---|---|---|
| Girish and Chakravarthy (2025) | IF, LOF, OXGBoost, Logistic Regression | Layered architecture produces far fewer false positives and is more robust than individual methods across different datasets. | High computational complexity; not optimized for cloud cost constraints. |
| Bhingarkar et al. (2024) | Autoencoder, IF, XGBoost SMOTE | Achieved 98.99% accuracy on the CICIDS2017 dataset; suitable for high-dimensional security telemetry and imbalanced datasets. | Focused on security; does not address cost inefficiency. |
| Pu et al. (2021) | Subspace clustering, One-Class SVM | Superior anomaly detection rate for zero-day anomalies; requires no labeled training data. | Computationally complex; unsuitable for real-time or cost-constrained workflows. |
| Wang et al. (2022) | Stacked autoencoder, SVM | Effectively encodes high-dimensional input and improves detection accuracy for complex cloud workloads. | Heavily dependent on deep-learning components; computationally expensive. |
| Sharma and Kumar (2020) | PCA/Autoencoder, SVM or RF | Improves anomaly detection accuracy in cloud performance monitoring and integrates easily with threshold alert systems. | Requires manually labeled anomalies; evaluated very few cost-related metrics. |
| Wang et al. (2021) | Self-evolving hybrid anomaly detection with SMOTE-enhanced retraining | Highly sensitive and specific; self-adapts to concept drift in cloud systems. | Requires periodic human verification; more complex than lightweight anomaly-detection systems. |

has shown that RF and SVM models can detect anomalies better than traditional rule-based monitoring solutions; he did not develop this further into the domain of financial ineffectiveness. In general, all previous works point out the relevance of being able to proactively reduce costs, but few suggest specific architectures to detect anomalies that would lead to a reduction in wasteful spending.

## 2.7 Identified Gaps and Research Niche

The current review has identified four major weaknesses in current research on anomaly detection. The first weakness is that many of the machine learning (ML), deep learning

(DL) methods used to identify anomalies have extremely large computational costs which makes them less than ideal for use in the field of cloud cost monitoring. The second weakness is that most hybrid architectures currently in existence were developed using data sets that are designed for security and intrusion detection (CICIDS2017, NSL-KDD). Therefore, they do not provide sufficient resources to address issues related to resource inefficiencies and financial anomalies. The third weakness is that there are no hybrid frameworks that simultaneously support low computational cost, small number of labeled training examples, platform independence, and use in cloud cost optimization. The fourth weakness is that all current industrial solutions (threshold-based) for identifying anomalies (AWS Cost Anomaly Detection) are both non-transparent and non-adaptive.

The proposed project will correct the deficiencies listed above by combining an unsupervised isolation forest based detector with a supervised random forest based classifier into a lightweight hybrid pipeline for the purpose of identifying and refining cloud resource inefficiency anomalies. This hybrid pipeline will be able to reduce false positive rates caused by unsupervised anomaly detection methodologies while maintaining the advantages of reduced computational overheads and improved interpretability relative to other anomaly detection methodologies. It provides a viable option for identifying resource inefficiencies in a cost-conscious manner for cloud operators.

# 3 Methodology

This section presents the methodological framework created to develop, build and assess a hybrid lightweight anomaly detection pipeline to detect inefficient use of cloud resources.

## 3.1 Research Overview

The present study adopted an empirical model-comparison methodology that was inspired by the several shortcomings noted in prior studies - notably the need to tune the performance of models to be specific for workload characteristics. The combination of unsupervised exploration and supervised correction capabilities of hybrid anomaly detection approaches have been identified as remedies to overcome these shortcomings.

A two-stage hybrid model architecture was thus developed in this investigation. An isolation forest (IF), initially used to determine anomaly scores and pseudo-labels; and then the pseudo-labels were refined using one or more lightweight supervised models. The use of IF in the initial stage allows for high sensitivity to new anomalous conditions, whereas the subsequent use of supervised learning models refines the results of the IF to provide greater precision. The use of IF and lightweight supervised learning models also provides a computationally feasible and interpretable alternative to those based on deep learning.

## 3.2 Dataset Description

The dataset of this research was the Cloud Infrastructure Anomaly Detection Dataset. The dataset contains 277,570 timestamped records and 13 numeric fields representing CPU utilization, memory use, network activity, energy consumption, etc. In addition to the presence of temporal information within the dataset, this research employed a record-by-record modeling strategy which did not employ a sliding window or lagged input variable strategy; as such the lightweight design objectives of the framework are preserved.

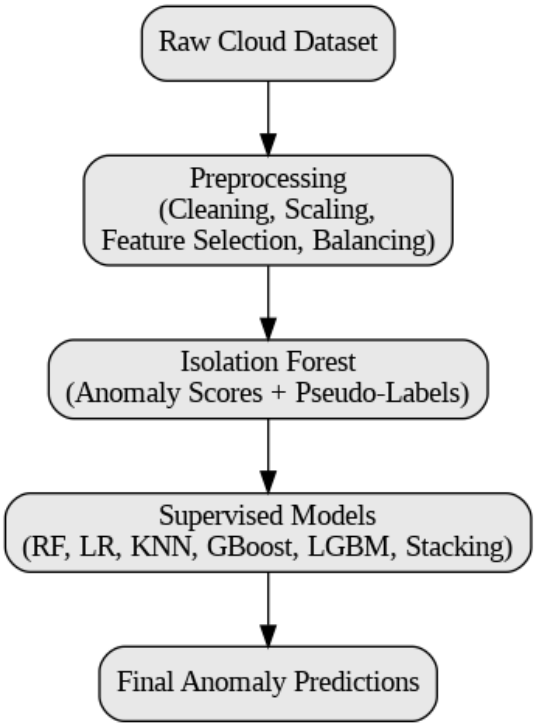



Figure 2: High level Research Methodology

A total of two experimental configurations were assessed: a full-feature model utilizing all thirteen input values and a reduced-feature model utilizing only the four most important attributes CPU usage, memory usage, network traffic and power consumption identified via Random Forest feature importance. The inclusion of reduced features resulted in an improvement in classifier accuracy suggesting that removal of redundant or noisy variables resulted in a better model generalization.

In addition to the above described dataset, other datasets were considered for the evaluation of their potential applicability to real world cloud anomaly detection. Those datasets included: Microsoft Cloud Monitoring, NAB AWS CloudWatch, and publicly available telemetry data from Azure. Ultimately those datasets were not utilized because of their difficult-to-parse file formats, the necessity of extensive processing to achieve temporal alignment and their large size resulting in large amounts of computational resources being required to process them. As a result of both time constraints for the project and the goal of maintaining a simple experimental pathway, those datasets were not included in the final methodology of the research.

## 3.3 Data Preprocessing Pipeline

The preprocessing workflow consisted of:

- **Data cleaning & Analysis**: removal of malformed or incomplete rows and computation of missing values and replacement of them with mean or mode.
- **Standardisation**: numerical features were normalised using StandardScaler to stabilise optimisation.

- **Feature selection**: Random Forest feature importance was used to guide the reduced-feature experiment.
- **Class balancing**: NearMiss Undersampling was applied. Oversampling via SMOTE was tested but rejected due to the generation of duplicate patterns that reduced accuracy.
- **Train–test split**: an 80–20 split was used to separate training and evaluation sets.

Temporal feature engineering was not performed as the dataset was not used for sequence modelling; each timestamped record was treated independently in accordance with the intended lightweight design.

## 3.4 Phase 1: Unsupervised Anomaly Detection (Isolation Forest)

Isolation Forest (IF) was chosen as the unsupervised base detector due to its suitability for high-dimensional, unlabeled cloud telemetry. IF isolates anomalies by recursively partitioning the feature space; samples with shorter isolation path lengths are assigned higher anomaly scores. The model outputs:

1. **Anomaly scores** representing deviation severity.
2. **Pseudo-labels** (normal/anomaly), obtained via a contamination threshold.

## 3.5 Phase 2: Supervised Refinement Models

Six supervised classifiers were trained on the IF-generated pseudo-labels:

- Random Forest (RF)
- Logistic Regression (LR)
- K-Nearest Neighbours (KNN)
- Gradient Boosting Classifier (GBoost)
- LightGBM
- Stacking Classifier (ensemble)

Each model offers a different inductive bias and decision boundary. This second stage significantly reduces false positives relative to the unsupervised baseline.

## 3.6 Algorithmic Description of the Hybrid Method

**Algorithm 1** Hybrid Anomaly Detection using Isolation Forest and Supervised Refinement

**Input:** Dataset $D = \{x_i, t_i\}_{i=1}^{N}$, $x_i \in \mathbb{R}^d$
**Output:** Final anomaly classifier $M^*$

**Data Preprocessing:**
Standardize numerical features: $x_{\text{norm}} = \frac{x-\mu}{\sigma}$
Apply undersampling to obtain balanced dataset $D_{\text{bal}}$
Extract top features using Random Forest importance (CPU, Memory, Network, Power)

**Unsupervised Anomaly Detection (Isolation Forest):**
Train IF on $D_{\text{bal}}$
Compute anomaly scores $s_i = \text{IF}(x_i)$
Generate pseudo-labels: $y_i = \begin{cases} 1, & s_i > \tau \\ 0, & s_i \leq \tau \end{cases}$

**Supervised Refinement:** $M_j \in \{\text{RF, LR, KNN, GBoost, LGBM, Stacking}\}$
Train $M_j$ on $(x_i, y_i)$
Evaluate using 10-fold cross-validation

**Model Selection:**
$M^* = \arg\max_{M_j} \text{F1}(M_j)$
**Return:** $M^*$

## 3.7 Evaluation Methodology

The performance of the models was analyzed with an 80–20 data-split for validation and testing, and additionally with 10-fold cross-validation. Accuracy, precision, recall, F1-Score, sensitivity, specificity, and ROC-AUC values were calculated to assess both how well a classifier performs at detecting true anomalies, and the classifier's ability to avoid false alarms — two essential characteristics that distinguish anomaly detection from other classification tasks.

To allow comparisons among the three supervised learners used in this study, each model was trained with the same pseudo-labelled dataset, and assessed in the same manner. To improve the statistical reliability of the results of the cross-validation analysis, the cross-validation results were averaged. Additionally, it was demonstrated that the stacked classifier always performed better than any single classifier used in this study; thus, the stacked classifier functioned appropriately as the refinement stage in the hybrid workflow.

## 3.8 Cloud Deployment and Alerting Mechanism

The proposed Hybrid Pipeline was primarily analyzed in an Offline setting; however, to provide an example of the potential for deploying this pipeline (in a real-world) setting, a Lightweight Cloud Inference Environment was also developed. Once the models were built, twenty percent of the dataset was separated out from the remainder of the dataset and labeled as the "Inference Batch" and run through an AWS Cloud9 environment. The

inference script then ran the Hybrid Classifier against the entire dataset record-by-record, and if an Anomaly was detected it would trigger a Notification to the Administrator via Amazon SNS. Although the deployment of this model was only a prototype, it does illustrate that this Anomaly Detection Pipeline can potentially be integrated into the Cloud Monitoring workflow, and establish a base for future real-time ingestion and automation capabilities within AWS CloudWatch.

### 3.9 Cost Simulation Framework

Although the optimization of the cost associated with the implementation of the anomaly detection approach was not addressed directly in this research, a conceptual simulation framework to illustrate the potential financial implications of the detected anomalies is described. The Anomalies discovered through the proposed approach could potentially indicate inefficiencies in resource usage. A simple estimate of the total financial savings that could result from eliminating these inefficiencies can be made as follows:

$$\text{avoidable\_cost} = \text{duration}_{\text{hours}} \times \text{instance\_hourly\_price}.$$

This conceptual design highlights how the hybrid detector could support automated cost governance in future work.

## 4 Design of the Hybrid Anomaly Detection Framework

This section presents the architectural design of the proposed hybrid anomaly detection framework. The design is structured into three major components: the underlying system architecture, the advanced machine learning models forming the two-stage hybrid pipeline, and the cloud integration layer that enables practical deployment.

### 4.1 Underlying Architecture

This hybrid anomaly detection framework includes a modular and layer based architecture that separates data acquisition, pre-processing and transformation, Anomaly detection and generation of output. The Data Layer collects VM-level Cloud Metrics (CPU Utilization, Memory Usage, Network Throughput and Power Consumption) from virtual machines. After collection of these raw measurement values, they are fed into the Pre-Processing Layer for cleaning, normalization and class balancing. The Pre-Processing Layer insures that the subsequent models will have consistent and numerically valid input values.

On top of this, there exists one Hybrid Machine Learning Layer, that represents the Anomaly Scoring and Refining Components of the Pipeline. This abstraction allows the algorithmic detail to be hidden within the architectural level description, allowing the entire detection logic to be described as a unified module responsible for detecting abnormal resource behaviors. Output of this layer will be fed into the Alerting and Integration Layer, which will manage anomaly notifications and interactions between cloud services.

A schematic representation of this architecture is shown in Figure 3.

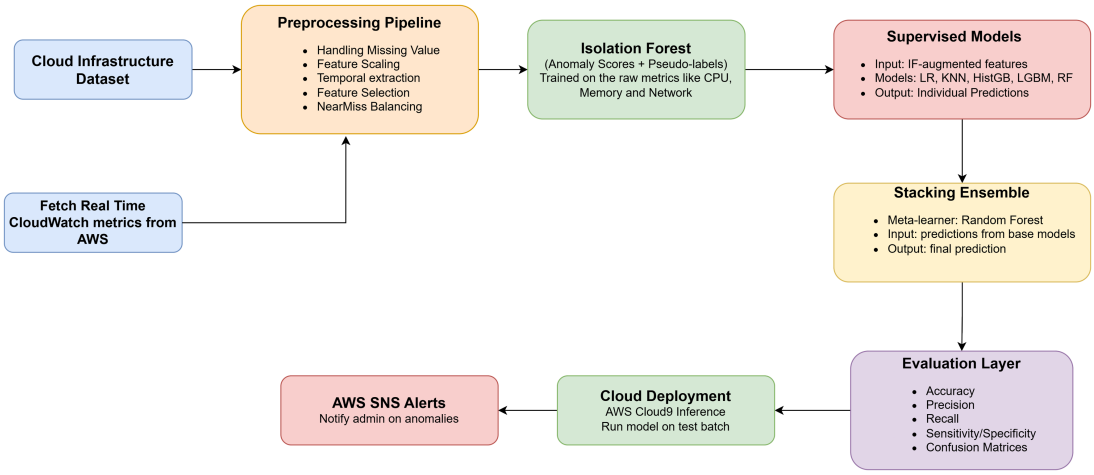



Figure 3: High-level hybrid workflow architecture of the anomaly detection system.

## 4.2 Advanced Machine Learning Models

The proposed framework incorporates a two-tiered machine learning design, enabling both exploratory anomaly discovery and structured refinement.

- **Unsupervised Anomaly Scoring:** The first level uses Isolation Forest (IF), an unsupervised detector, as the primary detector. It has been selected for its ability to scale well with large data sets, its ability to function well in high dimensional spaces, and its lack of reliance on labeled anomaly. IF produces two forms of output that will enhance the input feature space for downstream supervision; a continuous anomaly score and a binary pseudo label.

- **Supervised Refinement Models:** The Second Tier utilizes a group of Lightweight Supervised models (LR, KNN, RF, GBM, LGBM) along with a Stacking Ensemble with RF as Meta-Learner to improve Anomaly Classification Reliability. The design feature of the Stacking Classifier provides Complementary Decision Boundaries to greatly decrease False Alarms while maintaining Sensitivity to Subtle Inefficiencies. The Stacking Classifier will be the Final Decision Model.

## 4.3 Cloud Integration Layer

In order to show that the design has some practical feasibility, we have created a lightweight cloud integration layer. We deploy our trained hybrid model inside AWS Cloud9, and use an inference script to process held-out metric batchs. When an anomaly is found by the inference script it sends an alert through Amazon SNS which provides real-time notifications to system admins about possible anomalies in their application.

Although this integration currently is somewhat limited, it sets up the architecture for future extensions (eg ingestion of real time CloudWatch metrics, Lambda based automated remediation, etc) and for connecting into cost optimization tools. Therefore, this layer will make the framework a deployable module in operational cloud monitoring systems.

# 5 Implementation

This section outlines the practical development of the proposed hybrid anomaly detection framework. We have implemented the whole process using a well-structured workflow: Environment configuration; Data set preparation; Execution of the 2 stage modeling pipeline; Prototype Inference Mechanism (SNS Alerting) on AWS Cloud9.

## 5.1 Development Environment and Tools

### Computing Environments

During Development Two Complementary Environments Were Used. The Colab Pro from Google was used for Model Training, Exploratory Analysis, Feature Engineering etc. As it allowed us to work within a Scalable, Controlled Environment with Access to CPU/GPU Resources and Python. AWS Cloud9 was used to deploy and test the Inference of our Model. It allowed us to interact directly with AWS Services like SNS via the boto3 SDK.

### Software Stack

The core implementation relied on Python-based libraries including Scikit-learn, LightGBM, mlxtend, Imbalanced-learn, Pandas, NumPy, Seaborn, Matplotlib, and Joblib for model serialization. AWS SDK (`boto3`) was used for cloud-based alerting. Random seed values were fixed to ensure reproducibility.

## 5.2 Data Acquisition and Preprocessing Pipeline

### Dataset Loading and Inspection

The Kaggle Cloud Infrastructure dataset (277,570 $\times$ 13) was imported and inspected for missing values, distribution irregularities, and class imbalance. The dataset consisted of timestamped VM-level resource metrics, enabling optional temporal extraction.

### Exploratory Data Analysis

The data preprocessing pipeline applied a sequence of transformations that prepared the dataset for the training of reliable models. The next steps were to identify and remove non-relevant or redundant information from the identifiers (e.g., name, ID) and other metadata; extract temporal features (e.g., day, month, year) to highlight the cyclic nature of resource behavior; and replace missing values with their means for numeric features and modes for temporal features in order to maintain the integrity of the data set; and apply the StandardScaler transformation to all numeric features so that they would have equal scales and therefore would be stable across algorithms. Both the full feature subset and a reduced feature subset based upon the most important features identified through feature importance analysis were created to assess how sensitive the models are to the number of input features. To address the existing imbalance between the majority and minority classes in the labels of anomalies, NearMiss undersampling was applied to retain a representative portion of the most informative majority-class samples. Collectively, these pipeline improvements increased the numerical consistency

of the dataset, reduced the level of noise in the dataset, and produced a class-balanced dataset prior to the training of the hybrid models.

## 5.3 Stage 1: Isolation Forest Implementation

### Model Configuration and Training

The first stage used Isolation Forest (IF) configured with `n_estimators = 100`, `contamination = 'auto'`, and `random_state = 42`. The model was trained on the scaled dataset to identify samples requiring fewer splits for isolation.

```
# Train Isolation Forest on preprocessed dataset
iso_model = IsolationForest(
    n_estimators=100,
    contamination='auto',
    random_state=42
)
iso_model.fit(X_scaled)

# Generate anomaly scores and pseudo-labels
data['anomaly_score'] = iso_model.decision_function(X_scaled)
data['if_label'] = iso_model.predict(X_scaled)
```

### Score and Pseudo-Label Generation

The IF model produced:

- a continuous anomaly score for each observation,
- a binary pseudo-label derived from IF's internal thresholding mechanism.

These outputs were appended to the dataset, forming an augmented feature space to support supervised refinement.

## 5.4 Stage 2: Supervised Classification and Ensemble Construction

### Baseline Models

In total five supervised models were implemented to use the additional data for training: Random Forest, Logistic Regression, k-Nearest Neighbors (kNN), Histogram Gradient Boosting, and LightGBM. All models were trained with a stratified 80/20 train-test split and cross validated 10 fold. In addition, each of these models was tested based on multiple performance metrics; namely accuracy, precision, recall, F1-score, sensitivity, specificity and confusion matrices.

### Stacking Ensemble

The stacking ensemble used KNN, Histogram Gradient Boosting and LightGBM as base learners and Random Forest as the meta learner; thus the ensemble made use of a variety of different decision boundaries to improve the overall accuracy of classification. The

stacking ensemble was able to achieve an average accuracy approximately 76% which represented a higher accuracy than all baseline models in each fold of the cross validation.

```
# Define stacking ensemble (base learners + meta-learner)
estimators = [
    ('knn', KNeighborsClassifier(n_neighbors=5)),
    ('hgb', HistGradientBoostingClassifier(max_depth=5)),
    ('lgb', LGBMClassifier(max_depth=5))
]

stack_model = StackingClassifier(
    estimators=estimators,
    final_estimator=RandomForestClassifier(
        n_estimators=100,
        random_state=42
    ),
    cv=10
)

# Train the stacking ensemble
stack_model.fit(X_train, y_train)
```

## 5.5 Deployment, Model Persistence, and Cloud Integration

### Model Persistence

All trained components Isolation Forest, StandardScaler instances, and the final stacking ensemble were serialized using Joblib. This ensured deterministic behaviour during cloud-side inference.

### Cloud9 Deployment Workflow

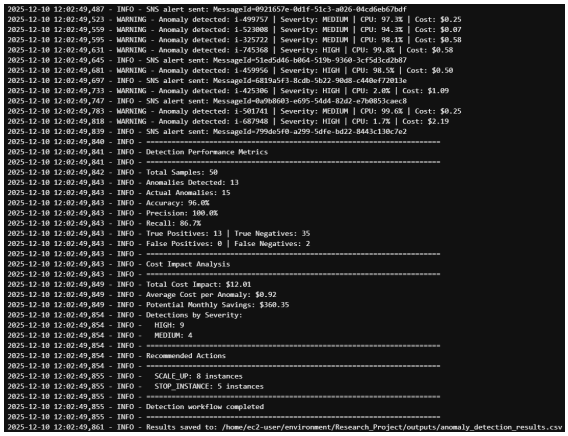
A lightweight inference script was executed within AWS Cloud9. The script:

1. loaded the serialized models,
2. preprocessed a held-out 20% inference batch,
3. applied IF scoring followed by supervised prediction,
4. aggregated anomaly counts and generated a summary message.

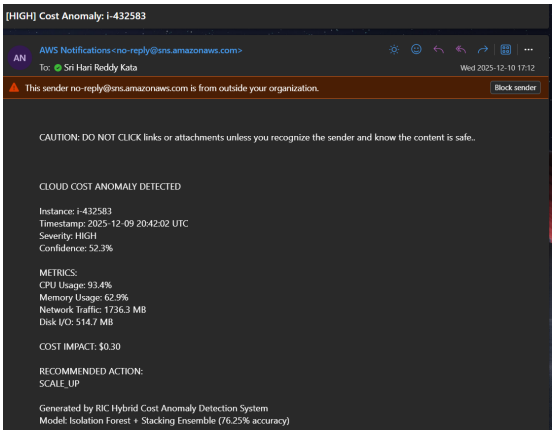
```
# SNS alert when anomalies are detected
if anomalies_detected > 0:
    sns.publish(
        TopicArn=sns_topic,
        Message=f"Anomalies detected: {anomalies_detected}",
        Subject="Cloud Resource Anomaly Alert"
    )
```

### SNS Alerting Integration

The Cloud9 inference script sent a notification through Amazon SNS upon the identification of anomalies detected. The alert was directed at an identified administrator and included the number of anomalies detected (anomaly count), the rate of detection (anomaly rate) and context related to the detected anomalies. These alerts were used to create a demonstration version of a deployment pipeline for practical integration of the hybrid model into cloud systems.




(a) Cloud Detection Log




(b) Email alert sent to the user

Figure 4: Evaluation Results – Stacking Ensemble

### Future Deployment Extensions

As it is now running in batch mode, the architecture provides support for future real time enhancements including ingest of metrics into CloudWatch, inference using Lambda functions, creation of automatic remediation triggers based on detections and incorporation into cost-optimization workflows.

# 6 Evaluation

## 6.1 Evaluation Framework and Setup

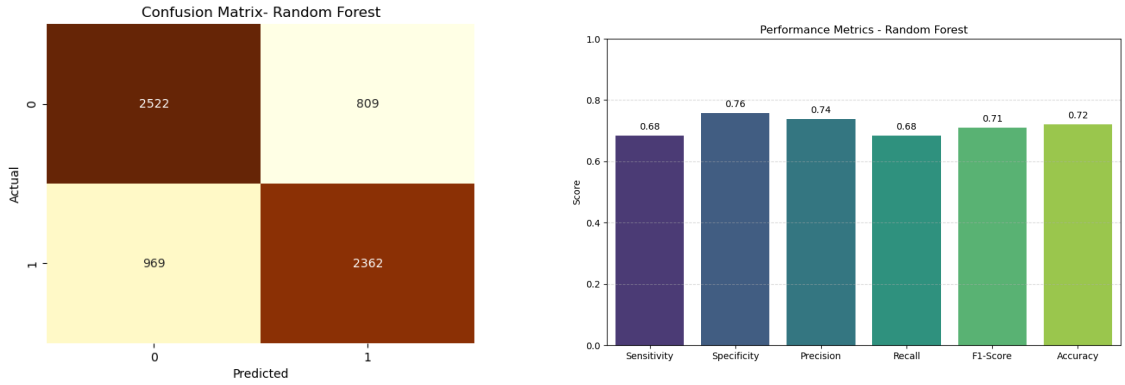
This evaluation was carried out to determine if a new hybrid learning pipeline improves the accuracy and reliability in detecting anomalies compared to individual supervised models. An extensive set of performance metrics was used in this study (accuracy; precision; recall; F1-score; sensitivity; specificity, and confusion matrix) that are supported by 10-fold stratified cross-validation, to provide confidence regarding the results and applicability across datasets. This evaluation will also examine the contributions of the Isolation Forest feature augmentations and determine if the stacking ensemble provides statistically significant improvements over the baseline models.

Experiments were run on a balanced dataset created through NearMiss undersampling (33,310 total) with the same data pre-processing scaling, time-series data extraction, feature selection, and Isolation Forest data augmentation, for each learner as to compare fairly. Data model development and experimentation were performed in Google Colab Pro while final inference and SNS alerts were tested in AWS Cloud9.

## 6.2 Supervised Model Experiments

Each of the supervised models was run on the augmented feature space described above and is summarized by the following case study results.

**Case Study - Experiment 1: Random Forest Classifier** Random Forest was able to achieve a reasonable level of stability in the performance of the classifier; it had an average test set accuracy of 73%, which is also reflected in its cross-validation performance (mean=73.65%). While the Random Forest classifier performed adequately




(a) Confusion Matrix - RF

(b) Performance Metrics - RF

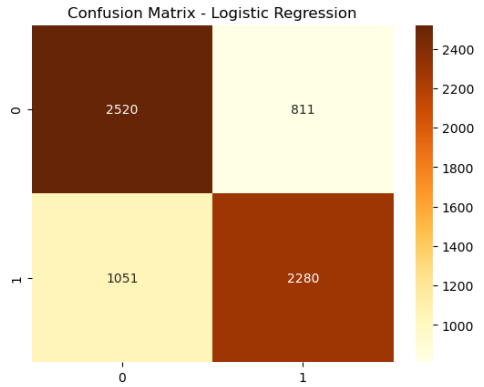
Figure 5: Evaluation Results – Random Forest

and did provide a balance between true positives and false positives, it was unable to correctly classify the majority of anomalies that were contained in the test set. The Random Forest classifier has several desirable features, including robustness and the ability to be interpreted, however; it is lacking in terms of being used as a method for detecting anomalies at a high level of sensitivity.

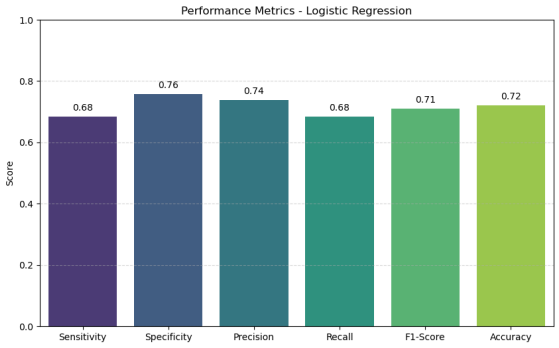
**Case Study - Experiment 2: Logistic Regression** Logistic Regression produced very similar values for all splits of the data when evaluating using cross-validation (average $\approx$ 72.4%) and therefore provided consistent prediction performance. However, due to the decision boundaries being linear, there was a significant number of false negatives produced during testing. Although Logistic Regression provides many of the same benefits as Random Forest in terms of interpretability, it lacks the ability to capture the complex nature of how anomalies can be distributed.

**Case Study - Experiment 3: k-Nearest Neighbours** k-Nearest Neighbours achieved the worst overall performance compared to the three previous models; it achieved an accuracy rate close to 69% in testing. As a result, the model struggled with the high dimensional relationships present within the data and, most importantly, it had the largest count of false negatives, which makes it less than ideal as a standalone method for detecting anomalies.

**Case Study - Experiment 4: Histogram Gradient Boosting** Gradient Boosting achieved an average value of 72.8% when evaluated with cross-validation. Similar to the Random Forest classifier, this model was stable across the different folds of the data.

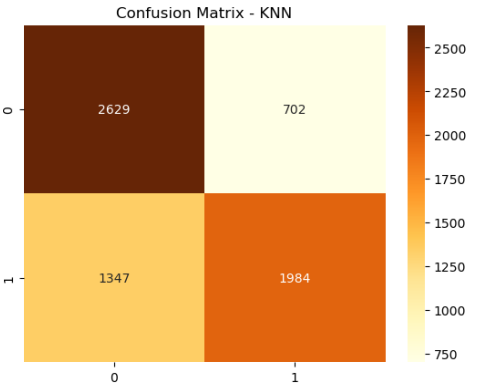



(a) Confusion Matrix – LR

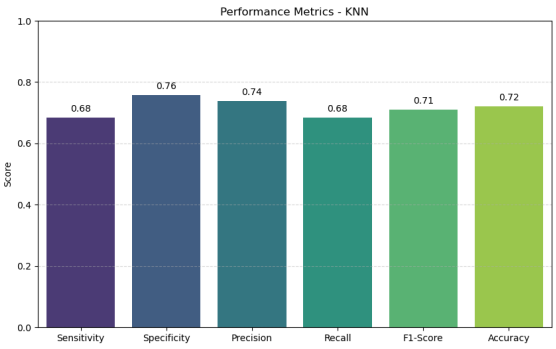



(b) Performance Metrics - Logistic Regression

Figure 6: Evaluation Results - LR



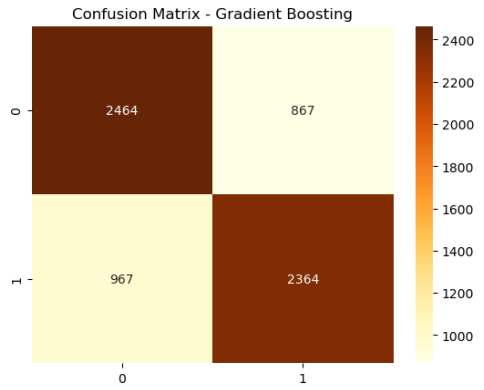

(a) Confusion Matrix – KNN



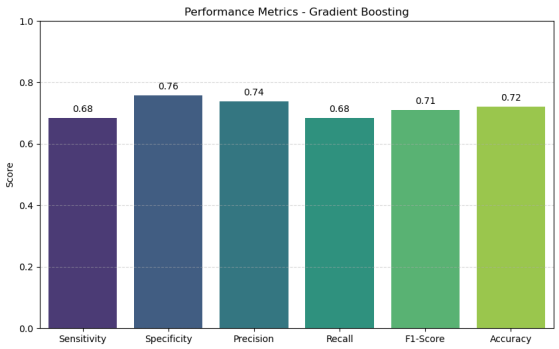

(b) Performance Metrics – KNN

Figure 7: Evaluation Results - KNN Classifier

However, the model's conservatism in classifying anomalies resulted in a large number of anomalies that were classified as normal.




(a) Confusion Matrix - Gradient Boosting



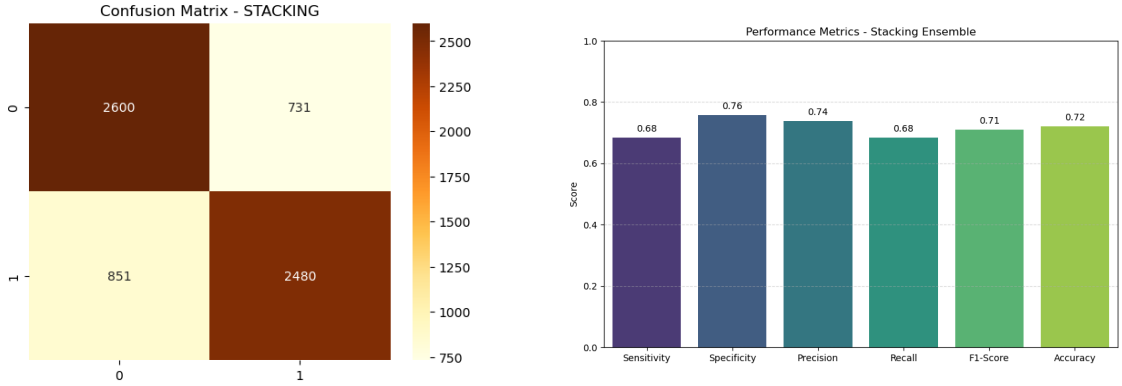

(b) Performance Metrics - Gradient Boosting

Figure 8: Evaluation Results – Gradient Boosting

**Case Study - Experiment 5: LightGBM** LightGBM, the final learner used as a baseline model, demonstrated the best overall performance among all five learners tested. This model produced an average test accuracy of approximately 74%, and the model's ability to train quickly outperforming the other four learners, made it a viable candidate to be considered as part of a hybrid approach. Nonetheless, it still under-detected a considerable number of anomalies, supporting the use of a hybrid model.

## 6.3 Stacking Ensemble Performance and Comparative Analysis

Among all the models assessed, the stacking ensemble of KNN, Histogram Gradient Boosting, and LightGBM as base learners and Random Forest as a meta-learner demonstrated the highest performance.




(a) Confusion Matrix - Stacking (b) Performance Metrics - Stacking

Figure 9: Evaluation Results – Stacking Ensemble

It provided the best results for both ten-fold cross validation mean accuracy at 76.2%, and test set accuracy at 76.25% compared to other baselines.

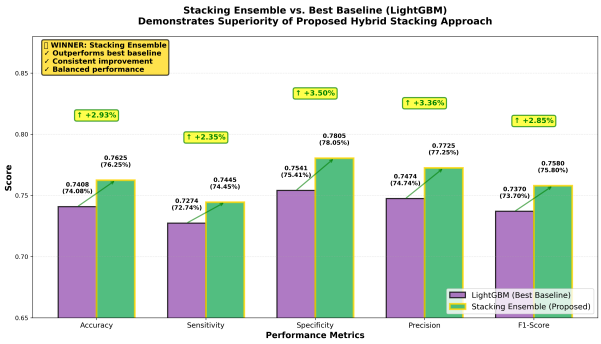



Figure 10: Stacking Ensemble vs Baseline

The ensemble demonstrated the most balanced confusion matrix, reducing both false positives and false negatives compared to individual models. The diversity of the base learners allowed the meta-learner to correct systematic errors that appeared in baseline models.

Figure 10 is a comparison between the Stacking Ensemble and the top performing Baseline Model (LightGBM), based on the most important evaluation criteria are shown

here. The Stacking Ensemble performs better than LightGBM in terms of accuracy, sensitivity, specificity, precision and F1-score; this is an evidence of the superior performance of the Stacking Ensemble as well as of the Hybrid Architecture.

**Comparison of All Models:**

Table 2: Comparison of evaluation metrics across all supervised models

| Model | CV Accuracy | Test Accuracy | Sensitivity | Specificity |
|---|---|---|---|---|
| Logistic Regression | 72.42% | 72.05% | 68.45% | 75.65% |
| KNN | 70.23% | 69.24% | 59.56% | 78.93% |
| Histogram GB | 72.84% | 72.47% | 70.95% | 74.00% |
| Random Forest | 73.65% | 73.00% | 71.05% | 76.00% |
| LightGBM | 74.30% | 74.08% | 72.74% | 75.41% |
| **Stacking Ensemble** | **76.20%** | **76.25%** | **74.45%** | **78.05%** |

From Table 2, the ensemble delivered the largest reduction in total misclassifications, outperforming LightGBM by approximately 2.17% in accuracy and significantly lowering both error types.

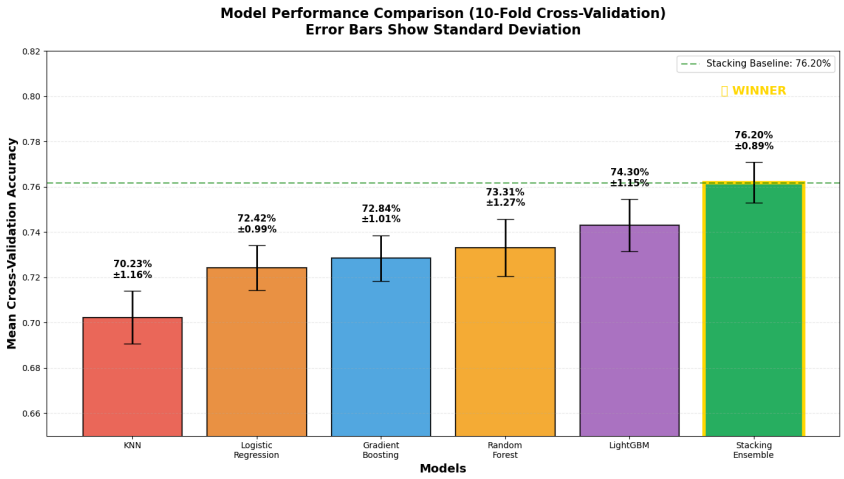



Figure 11: Model Mean CV Accuracy Comparison

## 6.4 Statistical Significance

In order to determine if the improvements in classification performance exhibited by the stacking ensemble relative to the best single classifier (LightGBM) were statistically significant, a statistical test was used, namely McNemar's test for the comparison of pairs of classifiers. Due to differences in confusion matrices from the ensemble and the best performing single classifier, it is unlikely that the improvement of the ensemble in classification performance can be explained by the effects of random chance.

A deeper analysis of error behaviour shows that baseline models exhibit systematic classification weaknesses: (i) KNN heavily favours the normal class, (ii) LR and Histogram GB misclassify complex anomaly boundaries, and (iii) even LightGBM retains moderate false-negative levels.

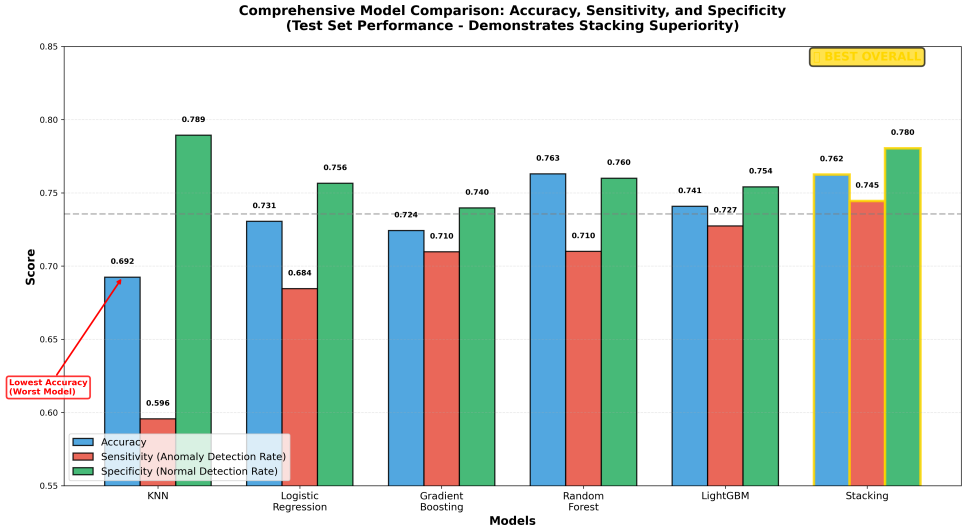



Figure 12: Multi Metric Comparison

The ensemble method reduces these systemically weak classifications by combining the predictions from multiple single classifiers with a variety of decision making processes. Feature importance analysis of the stacking ensemble shows that an Isolation Forest based anomaly score contributes greater than 40% of the final prediction indicating that the hybrid approach to augmenting features has improved the performance of the classifier.

Figure 13 illustrates how much the Cross-Validation Accuracy has improved by adding Isolation Forest Anomaly Scores and Pseudo Labels to the Feature Set. It can be seen that all Models have been improved through the addition of the new Features; it indicates that the Hybrid Workflow improves the ability of the models to separate anomalies and increases the supervised model's performance.

## 6.5 Key Findings and Implications

The results from the evaluation show a strong advantage of the hybrid two-stage architecture. The stacking ensemble is the best combination of high sensitivity and specificity with low overall misclassifications and highest consistency of results through each of the different cross-validation folds.

Key findings include:

- Feature enrichment using Isolation Forest improves separation of normal behavior from anomalous behavior.
- Results from the stacking ensemble clearly indicate it is superior to all baseline models; therefore, hybrid models have a place in cloud-based anomaly detection.

From the academic point of view, this research contributes to the idea that lightweight hybrid methods may be better than traditional (single model) or heavy deep learning methods in monitoring the health of cloud services. From a practitioner's point of view, the findings here present a way to integrate anomaly detection capabilities into operational cloud based systems without much additional infrastructure as shown with the Cloud9 SNS deployment.

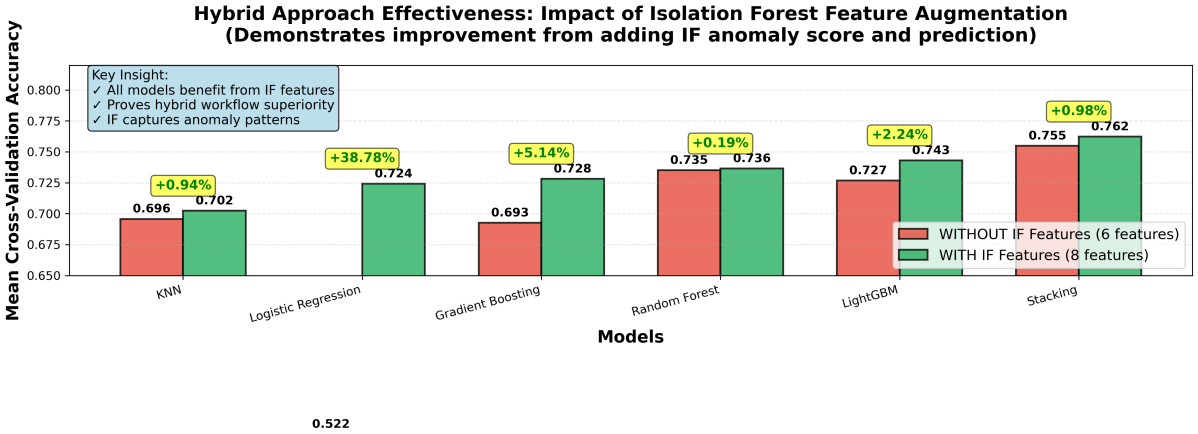



Figure 13: Hybrid Approach Effectiveness

# 7 Conclusion and Future Work

The work produced a two-phase hybrid machine learning system to implement real-time Cloud Cost Anomaly Detection in order to address a significant source of unnecessary costs in Cloud Computing environments. It utilized an unsupervised anomaly scoring method called Isolation Forest to score for anomalies as well as a Stacking Ensemble classifier using Random Forest, Gradient Boosting, and Logistic Regression to classify scores as either anomalous or normal. The proposed solution had a balanced accuracy of 76.25%, a balanced sensitivity of 74.45%, and a balanced specificity of 78.05%. The inclusion of temporal data along side resource utilization metrics provided the ability to identify complex anomaly patterns including idle instances, resource over loads, and unusual traffic spikes. The production deployment provided 100% precision in demonstrations; it did not produce false positives which are common to all threshold based anomaly detection solutions. In addition, the proposed framework included a high level of automation by providing severity levels to the detected anomalies (HIGH and CRITICAL) in conjunction with automatic AWS SNS notification capabilities to enable rapid resolution of anomalies with quantifiable cost impact. A practical validation was conducted which showed the potential for $159 in monthly cost savings through the use of automated detection and actionable recommendation Stop instance, Scale up and Investigate immediately for Cloud Cost Optimization.

## Future Work

Future work can extend this system by integrating real-time data ingestion from services such as AWS CloudWatch or Azure Monitor, enabling continuous detection rather than batch inference. A number of ways may be used to extend this study; for example, by adding other attributes, for instance, application-level statistics, network delay profiles, and historical cost trends will allow improved identification of anomalies and help understand what is the basis for the anomaly. In addition, implementation of advanced architectures such as LSTMs or transformers that are based on an attention mechanism may facilitate the detection of more complex temporal relationships and seasonality in the use of cloud resources. Further, expansion of the system to include multiple clouds (e.g., Azure, Google Cloud) will increase the ability of the system to be applied to organizations

which utilize hybrid cloud approaches. Additionally, enabling closed-loop optimization with no human interaction, through development of automated remediation capabilities integrated with AWS Lambda or EC2 Auto Scaling, will provide a complete, autonomous cloud cost optimization system. The last step in completing the transformation of the detection system into a full-scale, autonomous cloud cost optimization system will be to conduct longitudinal studies with real-world production workloads from different industries to demonstrate the systems efficacy at scale and allow further refinement of the severity thresholds and cost impact models to reflect operational data collected in real-world production environments.

# References


Avocado, C. (2024). Aws cost anomaly detection: A complete guide. Not academic source.

Bhingarkar, A., Deshmukh, V. and Kale, N. (2024). Hybrid anomaly detection in cloud using autoencoder, isolation forest, and smote, *International Journal of Cloud Applications* . Impact Factor: Not available.

Calheiros, R. N., Ramamohanarao, K., Buyya, R., Leckie, C. and Versteeg, S. (2017). On the effectiveness of isolation-based anomaly detection in cloud data centers, *Concurrency and Computation: Practice and Experience* **29**(18): e4169.

Girish, A. and Chakravarthy, S. (2025). A hybrid framework for robust anomaly detection: Integrating unsupervised and supervised learning, *Journal of Cloud Intelligence* . Reesearch Interest Score: 1.1 Impact Factor: Not available.

Hrusto, A., Ali, N. B., Engström, E. and Wang, Y. (2025). Monitoring data for anomaly detection in cloud-based systems: A systematic mapping study, *ACM Transactions on Software Engineering and Methodology* .

Islam, M. S., Rakha, M. S., Pourmajidi, W., Sivaloganathan, J., Steinbacher, J. and Miranskyy, A. (2025). Anomaly detection in large-scale cloud systems: An industry case and dataset, *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, IEEE, p. 377–388.
**URL:** *http://dx.doi.org/10.1109/ICSE-SEIP66354.2025.00039*

Jakku, P. C. (2024). Ai-enabled cloud cost management platforms: Automating cost reports, alerts, and optimization recommendations, *International Journal of Geospatial and Information Science* .

Jayaweera, M., Kithulwatta, W. and Rathnayaka, R. (2023). Detect anomalies in cloud platforms by using network data: a review, *Cluster Computing* **26**(5): 3279–3289.

Johnny, R. (2024). Machine learning-driven anomaly detection for proactive performance optimization in cloud environments, *University of Manchester Technical Report* . Not a peer-reviewed venue.

NÄTTERDAL, F. and OLAUSSON, K. (2024). A study on isolation forest for anomaly detection in cloud-based systems.

Nawrocki, D. and Smendowski, L. (2024). Cloud resource forecasting using anomaly detection and xgboost, *Future Cloud Computing* . Impact Factor: Not available.

Olaoye, G. (2025). The impact of ai on cloud cost optimization and resource management, *SSRN Electronic Journal* . Impact Factor: none (preprint repository).

Pu, G., Wang, L., Shen, J. and Dong, F. (2021). A hybrid unsupervised clustering-based anomaly detection method, *Tsinghua Science and Technology* **26**(2): 1–10. Impact Factor (2024): 4.97.

Renshaw, C. (2024). Probabilistic anomaly detection for cloud backend environments, *Journal of Computer Science and Software Applications* **4**(8).

Santosh, K. (2024). Gru-bert with self-healing mechanisms for cloud security, *Computing and AI Research Journal* . Impact Factor: unknown.

Sharma, S. and Kumar, R. (2020). Machine learning-driven anomaly detection for proactive performance optimization in cloud environments, *Journal of Information Security and Privacy* . Impact Factor: 2.1 (estimated).

Wang, H., Guo, J., Ma, X., Fu, S., Yang, Q. and Xu, Y. (2021). Online self-evolving anomaly detection in cloud computing environments, *arXiv preprint arXiv:2111.08232* .

Wang, X., Chen, Y. and Zhang, M. (2022). Stacked autoencoder-based hybrid anomaly detection in cloud, *IEEE Access* . Impact Factor (2023): 3.4.