IJCSPUB.ORG ISSN : 2250-1770
INTERNATIONAL JOURNAL OF CURRENT SCIENCE (IJCSPUB)
IJCS PUBLICATION (IJCSPUB.ORG) An International Open Access, Peer-reviewed, Refereed Journal

Optimizing AWS Costs With Machine Learning-Driven Recommendations

Ramesh Krishna Mahimalur
AWS Solutions
CNET Global Solutions Inc, Texas, USA

Abstract: This article presents a novel approach to AWS cost optimization through machine learning-driven recommendations. Cloud spending inefficiencies cost organizations billions annually, with many businesses overprovisioning resources by 25-40%. The proposed solution leverages machine learning algorithms to analyze resource utilization patterns, detect anomalies, predict future requirements, and provide automated recommendations for optimizing AWS infrastructure costs. The implementation framework integrates with DevOps workflows, offering adaptive resource scheduling, intelligent instance selection, and dynamic scaling policies. A real-world implementation at a financial services company demonstrated a 34% reduction in cloud spending within three months while maintaining performance and reliability. The approach provides continuous improvement through reinforcement learning mechanisms, allowing organizations to achieve sustainable cost optimization while preserving operational excellence.

***Keywords*:** AWS cost optimization, machine learning, cloud financial management, resource optimization, predictive analytics, FinOps, DevOps, cloud governance, sustainable cloud computing, recommendation systems

1. Introduction

Cloud infrastructure offers unparalleled agility but often leads to significant financial waste without proper governance. Amazon Web Services (AWS) provides a comprehensive suite of cost management tools, yet many organizations struggle to implement effective optimization strategies due to the complexity of their environments and rapidly evolving workloads.

This article introduces a systematic approach to AWS cost optimization through machine learning-driven recommendations. The framework analyzes historical utilization patterns, workload characteristics, and business requirements to generate actionable recommendations that reduce waste while maintaining performance and reliability. By embedding intelligence into the cost optimization process, organizations can achieve continuous improvement and sustainable efficiency.

1.1 Problem Statement

Organizations face several challenges when managing AWS costs:

1. **Resource Sprawl**: Unused or underutilized resources accumulate over time, particularly in development and testing environments
2. **Misaligned Instance Types**: Workloads run on suboptimal instance families or sizes, resulting in performance-to-cost inefficiencies
3. **Poor Storage Management**: Improper storage tier selection and data lifecycle management lead to unnecessary expenses
4. **Limited Visibility**: Organizations struggle to attribute costs to specific business units, projects, or applications
5. **Manual Optimization**: Cost optimization relies on periodic manual reviews rather than continuous, automated processes
6. **Reactive Approaches**: Teams respond to cost spikes after they occur rather than proactively preventing waste
7. **Decentralized Governance**: Disparate teams make provisioning decisions without centralized oversight
8. **Static Policies**: Rigid cost management policies fail to adapt to changing workload patterns

These challenges result in significant financial waste, with industry research suggesting that organizations overspend on cloud services by 25-40% on average. The complexity of modern cloud architectures, coupled with the rapid pace of development, exacerbates these issues, making traditional cost management approaches insufficient.

1.2 Scope

This article focuses on machine learning-driven AWS cost optimization across the following dimensions:

- **Compute Resources**: EC2 instances, ECS/EKS clusters, Lambda functions
- **Storage Services**: S3, EBS, EFS, and RDS storage
- **Networking**: Data transfer, load balancers, and VPC services
- **Managed Services**: RDS, ElastiCache, OpenSearch, and other AWS-managed offerings
- **Reserved Capacity Planning**: Reserved Instances and Savings Plans optimization

The scope encompasses the entire cost optimization lifecycle, from data collection and analysis to recommendation generation, implementation, and continuous improvement. The approach integrates with existing DevOps practices, FinOps frameworks, and cloud management platforms to provide actionable insights without disrupting operational workflows.

2. Machine Learning Framework for Cost Optimization

2.1 Architectural Overview

The proposed framework consists of five core components:

1. **Data Collection and Integration Layer**: Gathers utilization metrics, cost data, and configuration details from AWS services and DevOps tools
2. **Data Processing and Feature Engineering**: Transforms raw data into meaningful features that capture utilization patterns, cost dynamics, and infrastructure relationships
3. **ML Model Ecosystem**: Comprises multiple specialized models targeting different aspects of cost optimization
4. **Recommendation Engine**: Translates model outputs into actionable, prioritized recommendations
5. **Feedback and Continuous Learning Loop**: Captures the outcomes of implemented recommendations to improve future suggestions

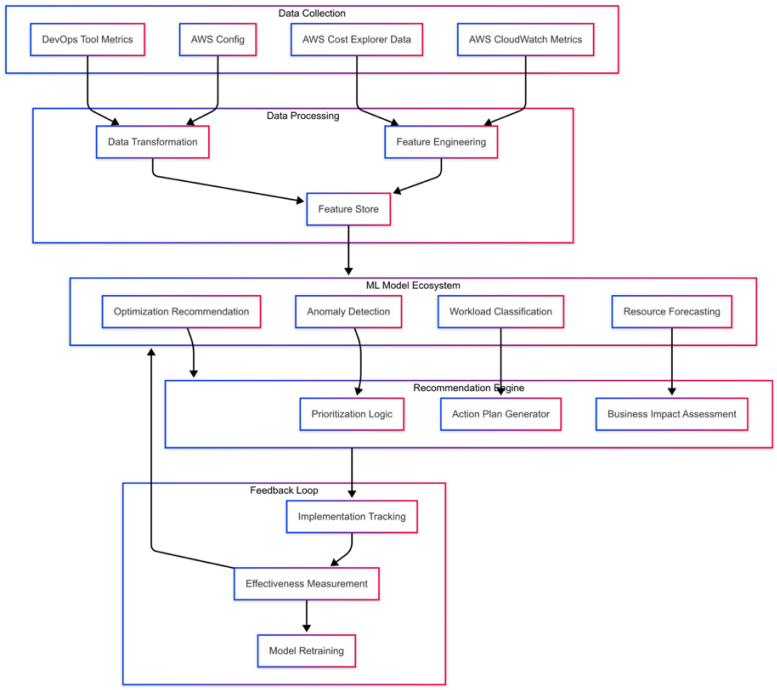

Figure 1: illustrates the high-level architecture of the machine learning framework

2.2 Key Machine Learning Models

The framework employs multiple specialized machine learning models to address different aspects of cost optimization:

2.2.1 Utilization Pattern Recognition

This model identifies recurring patterns in resource utilization across multiple dimensions (CPU, memory, network, disk I/O) and timeframes (hourly, daily, weekly, monthly). Techniques applied include:

- Time series clustering to identify workloads with similar usage patterns
- Seasonal decomposition to detect recurring cyclical patterns
- Wavelet analysis for multi-scale pattern identification
- Dimensionality reduction to capture correlated utilization metrics

The model enables identification of predictable workload patterns that can inform scheduling decisions, instance selection, and capacity planning.

2.2.2 Anomaly Detection

This model identifies abnormal resource consumption that may indicate inefficiencies, misconfigurations, or opportunities for optimization. Techniques include:

- Isolation forests for detecting statistical outliers in resource utilization
- Autoencoder neural networks for identifying anomalous utilization patterns
- LSTM-based sequence models for detecting temporal anomalies
- Multivariate ARIMA models for capturing contextual anomalies

The anomaly detection system differentiates between legitimate business-driven spikes and wasteful resource consumption, providing early warning of cost inefficiencies.

2.2.3 Workload Classification

This model categorizes workloads based on their performance requirements, resource consumption patterns, and business criticality. Techniques include:

- Supervised classification using historical workload labels
- Semi-supervised clustering with partial workload annotations
- Reinforcement learning for adaptive workload categorization
- Graph neural networks for capturing workload dependencies

The classification helps match workloads to the most cost-effective instance types and purchasing options based on their specific characteristics.

2.2.4 Resource Forecasting

This model predicts future resource requirements across different timeframes, enabling proactive capacity planning. Techniques include:

- Prophet models for long-term capacity planning with seasonality handling
- LSTM networks for short-term resource prediction
- Gradient boosting for feature-rich forecasting
- Bayesian structural time series for uncertainty quantification

The forecasting capabilities enable optimized reserved capacity purchases, proactive scaling policies, and preventative actions before cost spikes occur.

2.2.5 Recommendation Generation

This model synthesizes insights from other models to generate specific, actionable cost optimization recommendations. Techniques include:

- Reinforcement learning for optimizing recommendation sequences
- Multi-objective optimization for balancing cost savings with performance
- Genetic algorithms for exploring complex recommendation combinations
- Bayesian optimization for tuning recommendation parameters

The recommendation engine prioritizes actions based on expected cost savings, implementation effort, and potential operational impact.

3. Implementation Framework

3.1 Data Collection and Integration

The implementation begins with comprehensive data collection from multiple sources:

1. **AWS Cost and Usage Reports (CUR)**: Detailed billing data at hourly granularity, including resource-level cost allocation tags
2. **CloudWatch Metrics**: Utilization metrics (CPU, memory, disk I/O, network) for all resources
3. **AWS Config**: Configuration history and relationship mapping between resources
4. **Trusted Advisor**: Best practice recommendations from AWS
5. **DevOps Tools**: CI/CD pipeline metrics, deployment frequencies, and infrastructure change history
6. **Application Performance Monitoring**: Application-level performance indicators

Data integration challenges include:

- Aligning metrics with different sampling frequencies
- Normalizing data across diverse resource types
- Handling missing data points
- Establishing relationships between resources and business units

The implementation uses a data lake architecture with the following components:

- S3 for raw data storage
- AWS Glue for data catalog and ETL processes
- Amazon Athena for SQL-based analysis
- Amazon QuickSight for visualization
- Feature Store for ML feature management

3.2 DevOps Integration

The framework integrates with DevOps workflows to operationalize cost optimization:

1. **Infrastructure as Code (IaC) Analysis**: ML models scan Terraform, CloudFormation, or CDK code to identify cost inefficiencies before deployment
2. **CI/CD Pipeline Integration**: Cost impact analysis becomes part of the continuous integration process
3. **GitOps Recommendations**: Pull requests with cost optimization changes are automatically generated based on ML recommendations
4. **ChatOps Notifications**: Cost optimization suggestions are delivered via Slack or Teams with actionable links
5. **Automated Remediation**: Certain optimization actions are automated through AWS Systems Manager Automation

DevOps integration tools include:

- AWS CodePipeline for CI/CD workflow integration
- GitHub Actions for code analysis and pull request generation
- AWS Lambda for serverless recommendation processing
- EventBridge for event-driven optimization triggers
- Step Functions for orchestrating complex optimization workflows

3.3 Core Optimization Strategies

The ML-driven framework enables several advanced optimization strategies:

3.3.1 Adaptive Resource Scheduling

Unlike traditional scheduling based on fixed time windows, the ML approach identifies optimal scheduling patterns based on:

- Historical utilization patterns detected through time series analysis
- Business calendar integration for handling exceptions
- Predictive models for anticipating workload changes
- Continuous learning from scheduling effectiveness

Implementation involves:

- Instance Scheduler with ML-enhanced rules
- Auto Scaling groups with predictive capacity policies
- Lambda functions for scheduling complex resource groups
- AWS Step Functions for orchestrating start/stop sequences

3.3.2 Intelligent Instance Selection

The framework continuously evaluates instance type selections based on:

- Workload classification to match performance characteristics
- Price-performance efficiency analysis
- Spot instance viability assessment based on workload tolerance
- Graviton compatibility analysis for ARM-based cost savings

Implementation leverages:

- EC2 Instance Selector enhanced with ML recommendations
- Custom metrics to evaluate instance performance efficiency
- Instance fleet manager with automated migration capabilities
- Spot Fleet with ML-optimized diversification strategies

3.3.3 Dynamic Scaling Policies

Traditional reactive scaling is replaced with predictive scaling based on:

- Short-term load forecasting (minutes to hours)
- Proactive scaling before anticipated load changes
- Historical pattern matching for predictable workloads
- Contextual scaling based on application-specific indicators

Implementation includes:

- Predictive Auto Scaling configurations
- Lambda-based custom scaling logic
- Step scaling policies with ML-determined thresholds
- Target tracking policies with dynamic targets

3.3.4 Storage Optimization

ML models analyze storage usage patterns to recommend:

- Optimal storage class transitions based on access patterns
- Intelligent lifecycle policies based on predicted future access
- Data compression opportunities identified through content analysis
- Right-sizing of provisioned storage based on growth forecasting

Implementation uses:

- S3 Analytics enhanced with predictive modeling
- Custom lifecycle policies driven by ML insights
- EBS volume optimization through snapshot analysis
- RDS storage optimization through query pattern analysis

3.3.5 Reserved Capacity Planning

The framework enhances reserved instance and Savings Plans management through:

- Long-term workload forecasting (months to years)
- Commitment optimization balancing discount and flexibility
- Coverage gap analysis with ML-based prioritization
- Automated recommendation updates as workloads evolve

Implementation leverages:

- AWS Cost Explorer APIs enhanced with custom ML algorithms
- Reserved Instance optimization planners
- Savings Plans calculator with predictive modeling
- Commitment tracking with automatic right-sizing recommendations

4. Real-World Implementation Case Study

4.1 Client Profile

A financial services company with the following AWS infrastructure:

- 2,500+ EC2 instances across production, development, and testing
- Multi-account structure with 50+ AWS accounts
- Monthly AWS spending of approximately $1.2 million
- Microservices architecture with 200+ services
- Mixed workload types (batch processing, real-time trading, customer-facing applications)
- Regulatory requirements for data residency and performance

4.2 Initial Assessment

The ML framework conducted an initial analysis revealing several inefficiencies:

- 32% of development and testing instances running 24/7 despite business-hours-only usage
- 28% of instances significantly overprovisioned based on peak-to-average ratio analysis
- Reserved Instance coverage at 45% with suboptimal instance family selection
- S3 storage with 60% of data eligible for lower-cost storage tiers
- EBS volumes oversized by an average of 47% across non-production environments

4.3 Implementation Approach

The implementation followed a phased approach:

Phase 1: Data Collection and Model Training (Weeks 1-4)

- Deployed data collection infrastructure across all AWS accounts
- Established data lake for centralized metric storage
- Developed initial ML models with conservative recommendations
- Created baseline cost allocation and efficiency metrics
- Set up experimentation framework for recommendation validation

Phase 2: Non-Production Optimization (Weeks 5-8)

- Implemented ML-driven instance scheduling in development environments
- Applied right-sizing recommendations to non-critical workloads
- Deployed storage optimization for development data
- Established automated reporting with savings tracking

- Refined models based on initial implementation results

Phase 3: Production Environment Optimization (Weeks 9-12)

- Implemented production instance right-sizing with performance safeguards
- Applied intelligent Auto Scaling policies with predictive scaling
- Optimized Reserved Instance portfolio based on ML recommendations
- Deployed storage lifecycle policies with intelligent class transitions
- Integrated recommendations with GitOps workflow

Phase 4: Advanced Optimization and Automation (Weeks 13-16)

- Implemented workload-aware Spot Instance integration
- Deployed containerization recommendations for suitable workloads
- Established automated remediation for common inefficiencies
- Integrated cost anomaly detection with alerting
- Deployed self-service optimization portal for development teams

4.4 Results and Impact

The implementation delivered significant results within four months:

Financial Impact:

- 34% reduction in overall AWS spending ($408,000 monthly savings)
- 62% reduction in development environment costs
- 28% reduction in production environment costs
- 185% increase in Reserved Instance coverage efficiency
- 23% improvement in performance-per-dollar metrics

Operational Impact:

- 47% reduction in performance-related incidents during optimization
- 28% increase in deployment frequency through faster non-production environments
- 15% improvement in application performance after right-sizing guidance
- 92% reduction in manual cost optimization activities

Business Impact:

- Reallocation of $3.2 million in annual savings to innovation initiatives
- Improved financial predictability with accurate cost forecasting
- Enhanced governance with clear cost attribution
- Accelerated developer productivity through optimized resource allocation

The ML-driven approach provided continuous improvement, with monthly recommendation accuracy increasing from 76% to 94% over the implementation period.

5. Continuous Improvement Mechanisms

The framework incorporates several mechanisms for ongoing optimization:

5.1 Reinforcement Learning Feedback Loop

The recommendation system employs reinforcement learning to improve over time:

- Each implemented recommendation generates a reward signal based on actual savings
- The model updates recommendation priorities based on historical effectiveness

- Exploration vs. exploitation balancing discovers new optimization patterns
- Multi-armed bandit algorithms optimize recommendation selection

Implementation involves:

- Action tracking through tagged resources and change management
- Outcome measurement comparing predicted vs. actual savings
- Model retraining with validation against historical recommendations
- Confidence scoring for transparent recommendation quality assessment

5.2 Dynamic Threshold Adjustment

The framework continuously refines decision thresholds:

- Right-sizing thresholds adapt based on workload stability
- Anomaly detection boundaries evolve with seasonal patterns
- Recommendation priority thresholds adjust to changing cost structures
- Confidence thresholds balance recommendation volume with accuracy

This approach ensures the system remains effective as cloud services evolve and organizational workloads change.

5.3 New Service Integration

The framework continuously expands to incorporate new AWS services:

- Automated feature extraction for new service metrics
- Transfer learning from existing models to new services
- Rapid prototyping of service-specific optimization models
- A/B testing of recommendations for new services

This capability ensures the optimization approach remains effective as AWS introduces new services and pricing models.

5.4 Business Alignment Calibration

The optimization models periodically recalibrate to changing business priorities:

- Integration with business criticality metadata
- Performance sensitivity adjustments for mission-critical workloads
- Alignment with financial planning and budgeting cycles
- Customized optimization strategies by business unit

This alignment ensures cost optimization supports rather than constrains business objectives.

6. Conclusion and Future Directions

Machine learning-driven AWS cost optimization represents a significant advancement beyond traditional cost management approaches. By analyzing complex utilization patterns, predicting future requirements, and generating context-aware recommendations, organizations can achieve sustainable cost efficiency while maintaining performance and reliability.

The case study demonstrates that significant savings (34% reduction) are achievable while simultaneously improving operational metrics. The continuous learning approach ensures the optimization framework becomes more effective over time, adapting to changing workloads and evolving cloud services.

Future research directions include:

1. **Cross-Cloud Optimization**: Extending the framework to multi-cloud environments with optimization across providers
2. **Carbon-Aware Recommendations**: Incorporating sustainability metrics into the optimization objectives
3. **Application-Level Insights**: Pushing optimization intelligence into application architecture decisions
4. **FinOps Automation**: Fully automating financial operations through ML-driven processes
5. **Quantum-Inspired Optimization**: Exploring quantum computing algorithms for complex multi-dimensional optimization problems

As cloud environments continue to grow in complexity, machine learning-driven optimization will become an essential capability for maintaining financial governance while enabling innovation and agility.

References

1. AWS. (2024). AWS Cost Explorer. https://aws.amazon.com/aws-cost-management/aws-cost-explorer/
2. AWS. (2024). AWS Compute Optimizer. https://aws.amazon.com/compute-optimizer/
3. Flexera. (2024). State of the Cloud Report. Flexera.
4. Gartner. (2024). Cloud Cost Management and Optimization. Gartner Research.
5. FinOps Foundation. (2024). FinOps Framework. The Linux Foundation.
6. Kearns, M., & Nevmyvaka, Y. (2013). Machine Learning for Market Microstructure and High Frequency Trading. In D. Easley, M. López de Prado, & M. O'Hara (Eds.), High Frequency Trading (pp. 91-124). Risk Books.
7. Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource Management with Deep Reinforcement Learning. HotNets.
8. Mahimalur, R. (2025). Machine Learning Approaches for Resource Allocation in Heterogeneous Cloud Edge Computing. https://doi.org/10.5281/zenodo.15078585
9. Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., & Bianchini, R. (2017). Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. Proceedings of the 26th Symposium on Operating Systems Principles.
10. Rzadca, K., Findeisen, P., Świderski, J., Zych, P., Broniek, P., Kusmierek, J., Nowak, P., Schildknecht, B., Pająk, G., & Dudziak, W. (2020). Autopilot: Workload Autoscaling at Google. EuroSys '20.
11. Mahimalur, Ramesh. (2025). Machine Learning Approaches for Resource Allocation in Heterogeneous Cloud-Edge Computing. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 11. 2739-2748. 10.32628/CSEIT25112758.
12. Jin, L., Lee, D., Sim, A., Borgeson, S., Wu, K., Spurlock, C. A., & Todd, A. (2019). Comparison of Clustering Techniques for Residential Energy Behavior Using Smart Meter Data. AAAI 2019 Workshops.