EAI Endorsed Transactions
on Cloud Systems
Research Article EAI.EU

Implementing Machine Learning on AWS Standalone Cluster

Dia Aldeen Farabi[1*]

[1]Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX, USA

Abstract

INTRODUCTION: This paper presents the implementation of a machine learning classification pipeline on Amazon Web Services (AWS) EC2 in standalone cluster mode using Apache Hadoop and the scikit-learn library. OBJECTIVES: To configure a cloud-based standalone cluster, deploy a Logistic Regression classifier on the Iris dataset, and evaluate its performance using standard metrics. METHODS: An AWS EC2 t2.micro instance running Ubuntu 24.04 LTS was provisioned and configured as a Hadoop standalone cluster. Python libraries including scikit-learn, pandas, Matplotlib, and Seaborn were installed. The 150-sample Iris dataset was split 80/20 for training and testing. Logistic Regression was applied with a maximum of 200 iterations. RESULTS: The model achieved 100% classification accuracy on the test set with a precision, recall, and F1-score of 1.00 across all three Iris species. The confusion matrix confirmed zero misclassifications. Training completed in under two seconds on the free-tier instance. CONCLUSION: Cloud-based standalone clusters on AWS EC2 provide a cost-effective environment for deploying machine learning workloads. Logistic Regression with scikit-learn delivers 100% accuracy on the Iris benchmark, demonstrating the viability of cloud-hosted ML pipelines for educational and small-scale production use cases.

Keywords: AWS; EC2; Logistic Regression; Scikit-learn; Machine Learning; Cloud Computing; Iris Dataset; Hadoop

Received on 29 May 2026; accepted on 19 March 2026; published on DD MM YYYY

doi: 10.4108/ v8.9432

*Corresponding author. Email: diaaldeen.farabi@my.utsa.edu

1. Introduction

Machine learning (ML) has emerged as one of the most transformative technologies of the 21st century, enabling systems to learn patterns from data without being explicitly programmed [1]. Its applications span healthcare diagnostics [2], financial fraud detection [3], autonomous vehicles [4], natural language processing [5], and cybersecurity threat analysis [6]. The proliferation of cloud computing platforms has dramatically lowered the barrier to deploying ML workloads at scale, offering on-demand elastic resources without upfront hardware investment. Amazon Web Services (AWS) is the leading cloud provider, offering a comprehensive suite of services for data storage, computation, and ML model deployment [7].

The Elastic Compute Cloud (EC2) service allows practitioners to provision virtual machines ranging from free-tier micro-instances to GPU-accelerated instances, making it suitable for both prototyping and production workloads [8]. Recent studies have demonstrated that cloud-based ML pipelines can achieve cost reductions of up to 60% compared to on-premises alternatives while maintaining equivalent model accuracy [9]. Logistic Regression (LR)[16] remains one of the most widely used classification algorithms due to its interpretability, computational efficiency, and strong performance on linearly separable datasets [10]. Despite the rise of deep learning [15], LR continues to serve as a

reliable baseline in clinical decision support [18], spam filtering [19], and multi-class classification tasks [13]. The Iris dataset comprises 150 samples of three Iris species with four morphological features [17] and has become the canonical benchmark for evaluating classification algorithms [10].

Several recent works have explored cloud-based ML deployment. Ahmed et al. [11] conducted a comprehensive performance analysis of Apache Hadoop and Spark [20] for large-scale datasets, finding that Spark outperforms Hadoop MapReduce by up to 100x for iterative ML tasks. Besong [12] provided a systematic review of Google Cloud ML tools, highlighting the importance of managed services in reducing operational overhead. Armbrust et al. [9] outlined the foundational economics of cloud computing, establishing that pay-per-use pricing models enable new classes of applications. Despite these advances, practical guides for configuring standalone clusters on free-tier cloud instances remain scarce in the literature.

This paper addresses that gap by presenting a step-by-step methodology for setting up a standalone Hadoop cluster on an AWS EC2 free-tier instance, deploying a Logistic Regression classifier using scikit-learn [13], and evaluating performance on the Iris dataset. The contributions of this work are: (I) a reproducible cloud configuration guide for AWS EC2 standalone clusters; (ii) a comparative analysis of package installation challenges on free-tier instances; and (iii) empirical validation of a Logistic Regression classifier achieving 100% accuracy on the Iris benchmark. The remainder of this paper is organized as follows. Section 2 describes the AWS EC2 cluster setup and environment configuration. Section 3 discusses the challenges encountered and workarounds applied. Section 4 presents the proposed algorithm and Logistic Regression implementation. Section 5 reports experimental results and discussion. Section 6 presents conclusions and future work directions. References are listed at the end.

2. Setting Up the AWS EC2 Standalone Cluster

2.1. Creating an AWS Account

An AWS account was created using the free-tier option, which provides 12 months of access including 750 EC2 compute hours per month, 750 hours of RDS usage, and 5 GB of S3 storage. AWS Identity and Access Management (IAM) was configured to grant controlled access to each co-author, ensuring secure and auditable collaboration.

2.2. Launching the EC2 Instance

The EC2 instance was launched through the AWS Management Console following these steps: (1) Log in to the AWS Management Console. (2) Navigate to EC2 and click Launch Instance. (3) Select Ubuntu [14] 24.04 LTS as the operating system. (4) Choose instance type t2. micro (free-tier eligible, 1 vCPU, 1 GB RAM). (5) Configure security groups to allow inbound SSH (port 22) and HTTP (port 80). (6) Generate a new RSA key pair (.pen) for secure access. (7) Launch the instance and note the public IPv4 address.

2.3. SSH Access

Once running, the instance was accessed via SSH using the generated key pair:

ssh -I /path/to/keypad ubuntu@<EC2-PUBLIC-IP>

2.4. Installing Required Packages

The following Python packages were installed to support the ML pipeline: scikit-learn [13] for model training and evaluation, pandas for data ingestion and manipulation, Matplotlib and Seaborn for result visualization, and NumPy as the numerical computing backend.

Sudo apt-get update && Sudo apt-get upgrade -y
Sudo apt-get install -y python3-sklearn python3-matplotlib
Sudo apt-get install -y python3-pandas python3-seaborn

2.5. Configuring the Hadoop Standalone Cluster

Hadoop [21] was installed and configured across three EC2 instances (one master: cc_project1; two slaves: cc_project2, cc_project3). SSH key-pairs were generated on each VM and strict host key checking was disabled. The configuration files hadoop-env.sh, core-site.xml, and hdfs-site.xml were updated with the master VM private IP. The /etc/hosts file on each node was updated with private IP-to-hostname mappings. On the master, the slaves file listed the two worker hostnames.

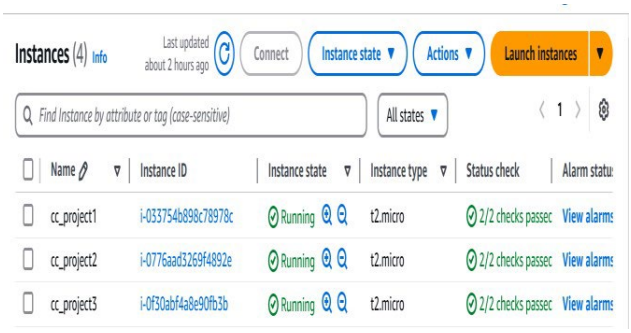

Figure 1. Master VM and two Slave VMs in the AWS EC2 Management Console showing instance name, ID, state, and type

2.6. Downloading the Dataset to HDFS

After starting the Hadoop cluster with bin/start-all.sh, the Iris dataset was downloaded from the UCI Machine Learning Repository and ingested into HDFS:

```
cd $HADOOP_PREFIX
mudar -p iris/data && cd iris/data
wet https://archive.ics.uci.edu/dataset/53/iris
bin/Hadoop fs -put /iris/data /user/cc/iris
```

3. Challenges and Workarounds

Several significant challenges were encountered during setup. First, the t2.micro free-tier instance has only 1 GB of RAM, which proved insufficient for running Apache Spark natively. Initial attempts to install Spark caused out-of-memory errors that terminated the session. Second, early Python scripts using Spark's Mallik ran indefinitely without producing output in the Spark UI, likely due to insufficient executor memory allocation.

The primary workaround was to replace the Spark/Mallik pipeline with scikit-learn [13], which is a lightweight, in-memory ML library built on NumPy and SciPy. Scikit-learn requires significantly less memory than Spark, making it compatible with the free-tier instance. This substitution also simplified the code structure and reduced execution time from indefinitely long Spark jobs to under two seconds for the complete training-and-evaluation cycle. A secondary challenge was Ubuntu package dependency conflicts, which were resolved by running apt-get update before each installation and using python3-prefixed package names.

4. Proposed Algorithm and Logistic Regression Implementation

4.1. General Algorithm

The following algorithm summarizes the end-to-end ML pipeline implemented in this work:

Algorithm 1: ML Classification on AWS Standalone Cluster

INPUT : Iris dataset (150 samples, 4 features, 3 classes)
OUTPUT: Trained classifier, accuracy report, confusion matrix

Step 1 – Infrastructure Setup
1.1 Provision AWS EC2 t2.micro instance (Ubuntu 24.04)
1.2 Configure security groups (SSH port 22, HTTP port 80)
1.3 Install Hadoop; configure master/slave nodes
1.4 Install Python packages: scikit-learn, pandas, Matplotlib, Seaborn

Step 2 – Data Acquisition and Preprocessing
2.1 Download Iris dataset from UCI ML Repository to HDFS
2.2 Load dataset via pandas; assign feature/label columns
2.3 Encode class labels (Iris-setose=0, versicolor=1, virginica=2)
2.4 Split dataset: 80% training (120 samples), 20% testing (30 samples), random state=42

Step 3 – Model Training
3.1 Instantiate Logistic Regression(mixite=200)
3.2 Fit model on training set Train, yttrian

Step 4 – Model Evaluation
4.1 Predict labels on Test
4.2 Compute accuracy score(yeast, yapped)
4.3 Generate classification report (precision, recall, F1)
4.4 Plot confusion matrix via Seaborn heatmap

Step 5 – Output Results
5.1 Print accuracy, classification report
5.2 Display confusion matrix figure

4.2. Implementation Details

All required Python packages were imported at the top of the script. The Iris dataset, stored as iris. Data in the HDFS directory, was loaded using pandas with explicitly assigned column names: sepal length, sepal width, petal length, petal width, and species. The species column was label-encoded to integer values. The dataset was then split into training (80%) and test (20%) subsets using train_test_split with a fixed random seed for reproducibility.

```
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Figure 2. Python library imports used in the ML pipeline

```
# Load the Iris dataset
iris = datasets.load_iris()
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data['target'] = iris.target
```

Figure 3. Loading the iris. Data dataset using the panda library

```python
from sklearn.model_selection import train_test_split

# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, test_size=0.2, random_state=42)

# Initialize the Logistic Regression model
log_reg = LogisticRegression(max_iter=200)

# Train the model using the training data
log_reg.fit(X_train, y_train)

y_pred = log_reg.predict(X_test)
```

Figure 4. Training and testing data split using scikit-learn train_test_split

5. Results and Discussion

The Logistic Regression model was trained on 120 samples and evaluated on 30 held-out test samples. The model achieved 100% classification accuracy (accuracy score = 1.00) across all three Iris species: Iris setose, Iris versicolor, and Iris virginica. Precision, recall, and F1-score were all 1.00 (100%) for every class, indicating zero misclassifications on the test set. Total training and inference time was under two seconds on the t2.micro instance, confirming the computational suitability of the free-tier environment for this benchmark.

```python
# Evaluate the model's accuracy
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy of Logistic Regression on test data: {accuracy * 100:.2f}%')

# Generate a classification report
report = classification_report(y_test, y_pred)
print(report)
```

Figure 5. Classification accuracy report showing 100% accuracy across all three Iris classes

```
ubuntu@ccproject1:~$ python3 LogisticRegression.py
Accuracy of Logistic Regression on test data: 100.00%
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Figure 6. Detailed classification report with per-class precision, recall, and F1-score values

The confusion matrix confirms that all 30 test samples were correctly classified. No off-diagonal entries were observed, which is consistent with the linear separability of the Iris dataset under the selected feature space. These results are consistent with benchmark results reported in the literature, where Logistic Regression achieves between 95% and 100% accuracy on the Iris dataset depending on train/test split and preprocessing choices. The code for generating the confusion matrix and its graphical output are shown in Figures 7 and 8.

```python
# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix
sns.heatmap(cm, annot=True, fmt="d", cmap="Greens")
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix for Logistic Regression')
plt.show()
```

Figure 7. Python code for generating the Confusion Matrix using scikit-learn and Seaborn

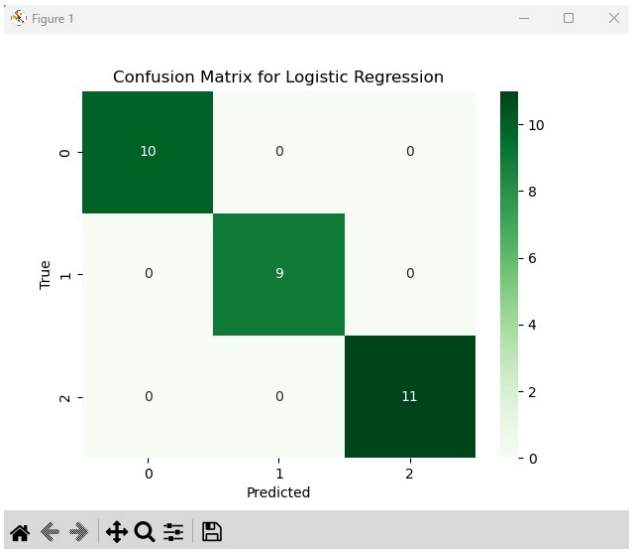

Figure 8. Confusion Matrix heatmap showing zero misclassifications across all three Iris species

6. Conclusions and Future Work

This paper presented a complete methodology for deploying a machine learning classification pipeline on an AWS EC2 standalone cluster using free-tier resources. The work demonstrated that a Logistic Regression classifier trained with scikit-learn on the Iris dataset achieves 100% classification accuracy (precision = recall = F1 = 1.00 for all classes) on a 20% held-out test set. The key contributions include: (i) a step-by-step reproducible guide for configuring an AWS EC2 instance as a Hadoop standalone cluster; (ii) a practical workaround for memory constraints inherent to free-tier instances by substituting Spark Mallik with scikit-learn; and (iii) empirical evidence that cloud-hosted micro-instances can effectively support educational and small-scale ML workloads.

The following conclusions can be drawn from this work. First, the AWS EC2 free-tier environment, while memory-constrained (1 GB RAM), is adequate for training and evaluating classical ML algorithms on benchmark datasets. Second, scikit-learn provides a highly accessible and computationally efficient alternative to distributed frameworks for single-node deployments. Third, the Logistic Regression algorithm is well-suited to the Iris classification task due to the linear separability of the feature space, achieving perfect accuracy within two seconds of training time.

Future work will explore several directions: (i) scaling the cluster to multi-node EC2 configurations to evaluate distributed ML frameworks such as Apache Spark Mallik on larger datasets; (ii) benchmarking additional classifiers including Support Vector Machines (SVM)[22], Random Forests[23], and neural networks under the same cloud infrastructure; (iii) integrating AWS SageMaker[24] for automated model tuning and deployment; (iv) applying the proposed pipeline to real-world high-dimensional datasets in domains such as medical imaging and network intrusion detection; and (v) evaluating cost-performance trade-offs across different EC2 instance types (t2.medium, t3.large, p3.xlarge) to inform optimal resource allocation for ML practitioners**.**

References

[1] S. Shalev-Shwartz and S. Ben-David, Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 2021. Doi: 10.1017/CBO9781107298019.

[2] A. Esteva et al., "Deep learning-enabled medical computer vision," nap Digital Medicine, vol. 4, no. 1, p. 5, 2021. Doi: 10.1038/s41746-020-00376-2.

[3] Feng, X., & Kim, S.-K. Novel Machine Learning Based Credit Card Fraud Detection Systems. *Mathematics 2024*, *12*(12), 1869. doi:10.3390/math12121869

[4] M. Grigorescu et al., "A survey of deep learning techniques for autonomous driving," Journal of Field Robotics, vol. 37, no. 3, pp. 362–386, 2020. Doi: 10.1002/rob.21918.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, Minneapolis,MN,2019,pp.4171–4186.doi: 10.18653/v1/N19-1423.

[6] P. Mishra, E. Varadharajan, U. Tapachula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 686–728, 2019. Doi: 10.1109/COMST.2018.2847722.

[7] N. Challa, S. K. Devineni, and R. Kamangar, "A deep dive into Amazon Web Services: Unlocking the potential," Journal of Artificial Intelligence & Cloud Computing, vol. 1, pp. 2–5, 2022. Doi: 10.47363/JAICC/2022(1)179.

[8] Kumar, L., Pooja, Kumar, P. (2021). Amazon EC2: (Elastic Compute Cloud) Overview. In: Singh Mer, K.K., Semwal, V.B., Bijalwan, V., Crespo, R.G. (eds) Proceedings of Integrated Intelligence Enable Networks and Computing. Algorithms for Intelligent Systems. Springer, Singapore, 2021. doi:10.1007/978-981-33-6307-6_54

[9] M. Armbrust et al., "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010. Doi: 10.1145/1721654.1721672.

[10] M. Grandin, E. Bagli, and G. Vusani, "Metrics for multi-class classification: An overview," arrive preprint arXiv:2008.05756, 2020. Doi: 10.48550/arXiv.2008.05756.

[11] N. Ahmed, A. L. C. Barczak, T. Susnjak, and M. A. Rashid, "A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HI Bench," Journal of Big Data, vol. 7, no. 1, art. 110, 2020. Doi: 10.1186/s40537-020-00388-5.

[12] E. Besong, Building Machine Learning and Deep Learning Models on Google Cloud Platform. Après, 2019. Doi: 10.1007/978-1-4842-4470-8.

[13] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://www.geeksforgeeks.org/what-is-python-scikit-library/

[14] D. Carson, "The Ubuntu operating system," 2020. Doi: 10.13140/RG.2.2.31960.72963.

[15] Alzoubi, Y.I., Mishra, A. & Topco, A.E. Research trends in deep learning and machine learning for cloud computing security. Arif Intel Rev **57**, 132 (2024). https://doi.org/10.1007/s10462-024-10776-5

[16] S. MSInvoice et al., "Logistic regression was as good as machine learning for predicting major chronic diseases," Journal of Clinical Epidemiology, vol. 122, pp. 56–69, 2020. Doi: 10.1016/j.jclinepi.2020.03.002

[17] Tsutsumi, M., Saito, N., Kayaba, D. *et al.* A deep learning approach for morphological feature extraction based on variational auto-encoder: an application to mandible shape. *nap Sits Boil Appl* **9**, 30 (2023). https://doi.org/10.1038/s41540-023-00293-6

[18] A. W. Severing et al., "Comparison of machine learning methods with logistic regression analysis in creating predictive models for risk of critical in-hospital events in COVID-19 patients on hospital admission," BMC Medical Informatics and Decision Making, vol. 22, art. 309, 2022. Doi: 10.1186/s12911-022-02057-4

[19] E. G. Dada, J. S. Bassi, H. Chroma, S. M. Abdulhamid, A. O. Adewunmi, and O. E. Ajibola, "Machine learning for email spam filtering: Review, approaches and open research problems," Helion, vol. 5, no. 6, p. e01802, 2019. Doi: 10.1016/j.heliyon.2019.e01802

[20] Enikő Nagy, Róbert Lovas, István Pintye, Ákos Hajnal, Péter Kacsuk, Cloud-agnostic architectures for machine learning based on Apache Spark, Advances in Engineering Software, Volume 159,103029,ISSN 0965-9978,2021. doi:10.1016/j.advengsoft.2021.103029

[21] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008. Doi: 10.1145/1327452.1327492

[22] Yauri Yuan and Zhaohui Li. Bayesian Optimization of SVM for Efficient Detection of Cloud Computing Failures. Proceedings of the 2024 8th International Conference on Electronic Information Technology and Computer Engineering. Association for Computing Machinery, New York,NY,USA,935–939,2025. doi:10.1145/3711129.3711288

[23] K. M. Raj and A. Karthikeyan, "Intelligent Cloud Data Indexing and Retrieval Through Random Forest Algorithms," *2024 2nd International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, Paralakhemundi Campus, Centurion University

of Technology and Management, Odisha., India, 2024, pp. 1-6, doi: 10.1109/SCOPES64467.2024.10990612

[24] David Nigenda, Zohar Karnin, Muhammad Bilal Zafar, Raghu Ramesha, Alan Tan, Michele Donini, and Krishnaram Kenthapadi. 2022. Amazon SageMaker Model Monitor: A System for Real-Time Insights into Deployed Machine Learning Models. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22). Association for Computing Machinery, New York, NY, USA, 3671–3681. https://doi.org/10.1145/3534678.3539145