AI-Powered Data Pipeline for E-commerce Warehouse Demand Forecasting

# Implementation Steps

Data Ingestion Using Terraform and Python

Data Processing Using AWS Glue & Lambda

**Machine Learning with Amazon SageMaker (for demand forecasting)**

Model Integration & Optimisation

Deployment & Scaling



Table of Contents

[Implementation Steps 1](#_Toc192944781)

[Block 3: Machine Learning with Amazon SageMaker 3](#_Toc192944782)

[Objective: 3](#_Toc192944783)

[Verify Processed Data Availability 3](#_Toc192944784)

[Create an Amazon SageMaker Notebook Instance 4](#_Toc192944785)

[Load & Explore Data in SageMaker 4](#_Toc192944786)

[3. Load the processed data from S3 5](#_Toc192944787)

[4. Visualise basic statistics 5](#_Toc192944788)

[Prepare Data for Model Training 6](#_Toc192944789)

[1. Convert timestamp to determine and create new features 6](#_Toc192944790)

[2. Select features and target variable 6](#_Toc192944791)

[3. Save the prepared dataset as CSV for SageMaker 6](#_Toc192944792)

[4. Checked the dataset was successfully written to S3 6](#_Toc192944793)

[Train a Demand Forecasting Model in SageMaker 6](#_Toc192944794)

[1. Set up and SageMaker session 6](#_Toc192944795)

[Deploy the Model as an Endpoint 7](#_Toc192944796)

[Automate predictions using lambda 8](#_Toc192944797)

[Issues Encountered 10](#_Toc192944798)

[Summary of Key Deliverables 11](#_Toc192944799)

# Step 3: Machine Learning with Amazon SageMaker

## Objective:

* Prepare and structure data for machine learning.
* Train a demand forecasting model using Amazon SageMaker.
* Deploy the model and generate predictions for warehouse stock optimisation.

## Verify Processed Data Availability

1. Confirm all the processed data is still available in S3



1. Preview the sample



## Create an Amazon SageMaker Notebook Instance

Go to the AWS SageMaker Console → Notebook Instances → Click Create notebook instance → Enter Notebook Name → Choose an IAM Role: Select “Create a new role” and grant access to you ingestion bucket – choose desired instance type ( ml.t3.medium (or ml.m5.large for larger datasets) → create notebook instance.



## Load & Explore Data in SageMaker

* 1. Open a Jupiter notebook and create a new python notebook



* 1. Install all the necessary libraries



* 1. Load the processed data from S3



* 1. Visualise basic statistics



## Prepare Data for Model Training

1. Convert timestamp to determine and create new features



1. Select features and target variable



1. Save the prepared dataset as CSV for SageMaker



1. Checked the dataset was successfully written to S3



## Train a Demand Forecasting Model in SageMaker

1. Set up and SageMaker session



1. Choose an ML algorithm: The following script sets up the training job and chooses XGBoost, a powerful model for forecasting.





## Deploy the Model as an Endpoint

1. Deploy the trained model



1. Test the deployed model



## Automate predictions using lambda

1. Create a new AWS Lambda Function



1. Attach necessary IAM permissions



1. Add the Lambda Function code



1. Test the Lambda Function via the AWS console by creating a test event



# Issues Encountered

**Problem 1:** Processed Data Not Accessible from SageMaker**.** When attempting to load the processed dataset from Amazon S3, SageMaker could not access the data, causing a permissions error.
**Resolution 1:** Updated IAM role policies to explicitly grant SageMaker permissions to read from the S3 bucket and verified S3 bucket policies to ensure they allowed access from SageMaker’s execution role.

**Problem 2**: SageMaker Training Job Failing Due to Instance Type Limits. The initially chosen instance type (ml.t3.medium) ran out of memory when training the demand forecasting model.
**Resolution 2:** Upgraded to a **larger instance type (ml.m5.large)** for better performance and used Amazon SageMaker Spot Instances to reduce costs while using a more powerful instance.

**Problem 3:** Model Deployment Endpoint Timing Out. The deployed SageMaker endpoint was timing out when making inference requests. **Resolution 3:** Adjusted endpoint configurations to allow higher request timeouts, optimised the model by reducing payload size for faster response times and tested with sample requests using the AWS SDK (Boto3) before integrating with the application.

**Problem 4:** Lambda Function Could Not Invoke SageMaker Endpoint. AWS Lambda was unable to invoke the SageMaker endpoint due to missing permissions.
**Resolution 4:** Updated IAM permissions to allow Lambda to invoke the SageMaker runtime endpoint and verified the Lambda function by testing with a manually created test event in the AWS Console.

# Summary of Key Deliverables

1. **IAM Role Management:** Ensuring proper permissions for SageMaker to access S3, Lambda, and endpoint invocations.
2. **Data Preprocessing:** Addressing **formatting issues** before passing data into the ML model.
3. **Resource Optimisation:** Using **larger instance types & Spot Instances** for training efficiency.
4. **SageMaker Endpoint Optimisation:** Fixing timeout issues by tuning request limits & payload sizes.