MODEL TRAINING & EVALUATION FOR FINANCIAL FRAUD DETECTION WITH AMAZON SAGEMAKER & CONTROL-M

Model training and evaluation are fundamental in payment fraud detection because the effectiveness of a machine learning (ML)-based fraud detection system depends on its ability to accurately identify fraudulent transactions while minimizing false positives. Given the high volume, speed, and evolving nature of financial fraud, properly trained and continuously evaluated models are essential for maintaining accuracy and efficiency. Fraud detection requires a scalable, automated, and efficient approach to analyzing vast transaction datasets and identifying fraudulent activities.

This blog presents an ML-powered fraud detection pipeline built on Amazon Web Services (AWS) solutions—Amazon SageMaker, Amazon Redshift, Amazon EKS, and Amazon Athena—and orchestrated using Control-M to ensure seamless automation, scheduling, and workflow management in a production-ready environment. The goal is to train three models—logistic regression, decision tree, and multi-layer perceptron (MLP) classifier across three vectors—precision, recall, and accuracy. The results will help decide which model can be promoted into production.

While model training and evaluation is the outcome, the training and evaluation is part of the larger pipeline. In this blog, the represented pipeline integrates automation at every stage, from data extraction and preprocessing to model training, evaluation, and result visualization. By leveraging Control-M's orchestration capabilities, the workflow ensures minimal manual intervention, optimized resource utilization, and efficient execution of interdependent jobs.

Process Flow:

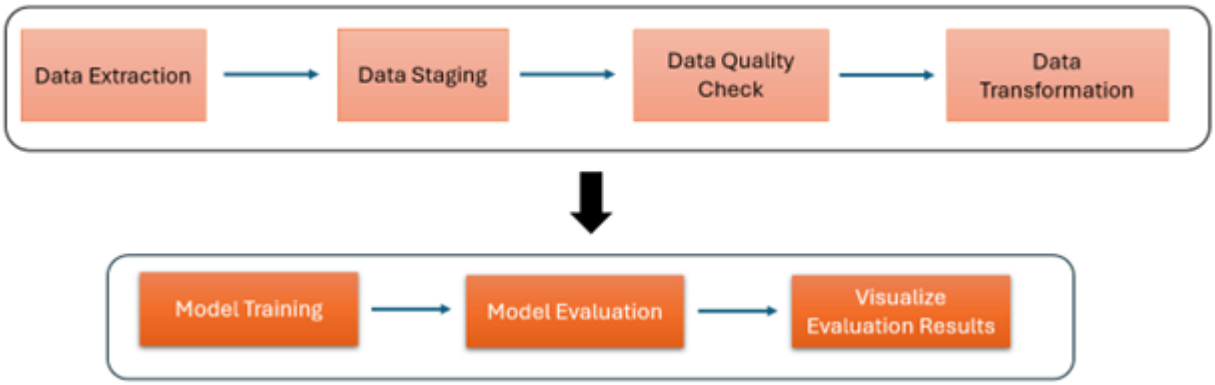


Figure 1. The end-to-end pipeline.

Key architectural highlights include:

- Automated data extraction and movement from Amazon Redshift to Amazon S3 using Control-M Managed File Transfer (MFT)
- Orchestrated data validation and preprocessing with AWS Lambda and Kubernetes (Amazon EKS)
- Automated model training and evaluation in Amazon SageMaker with scalable compute resources
- Scheduled performance monitoring and visualization using Amazon Athena and QuickSight
- End-to-end workflow orchestration with Control-M, enabling fault tolerance, dependency management, and optimized scheduling

In production environments, manual execution of ML pipelines is not feasible due to the complexity of handling large-scale data, model retraining cycles, and continuous monitoring. By integrating Control-M for workflow orchestration, this solution ensures scalability, efficiency, and real-time fraud detection while reducing operational overhead. The blog also discusses best practices, security considerations, and lessons learned to help organizations build and optimize their fraud detection systems with robust automation and orchestration strategies.

Amazon SageMaker:

The core service in this workflow is Amazon SageMaker, AWS's fully managed ML service, which enables rapid development and deployment of ML models at scale. We've automated our ML workflow using Amazon SageMaker Pipelines, which provides a powerful framework for orchestrating complex ML workflows. The result is a fraud detection solution that demonstrates the power of combining AWS's ML capabilities with its data processing and storage services. This approach not only accelerates development but also ensures scalability and reliability in production environments.

Dataset Overview

The dataset used for this exercise is sourced from Kaggle, offering an excellent foundation for evaluating model performance on real-world-like data.

The Kaggle dataset used for this analysis provides a synthetic representation of financial transactions, designed to replicate real-world complexities while integrating fraudulent behaviors. Derived from the PaySim simulator, which uses aggregated data from financial logs of a mobile money service in an African country, the dataset is an invaluable resource for fraud detection and financial analysis research.

The dataset includes the following features:

- **step:** Time unit in hours over a simulated period of 30 days.
- **type:** Transaction types such as CASH-IN, CASH-OUT, DEBIT, PAYMENT, and TRANSFER.
- **amount:** Transaction value in local currency
- **nameOrig:** Customer initiating the transaction.
- **oldbalanceOrg/newbalanceOrig:** Balance before and after the transaction for the initiating customer.
- **nameDest:** Recipient of the transaction.
- **oldbalanceDest/newbalanceDest:** Balance before and after the transaction for the recipient.
- **isFraud:** Identifies transactions involving fraudulent activities.
- **isFlaggedFraud:** Flags unauthorized large-scale transfers.

Architecture

The pipeline has the following architecture and will be orchestrated using Control-M.

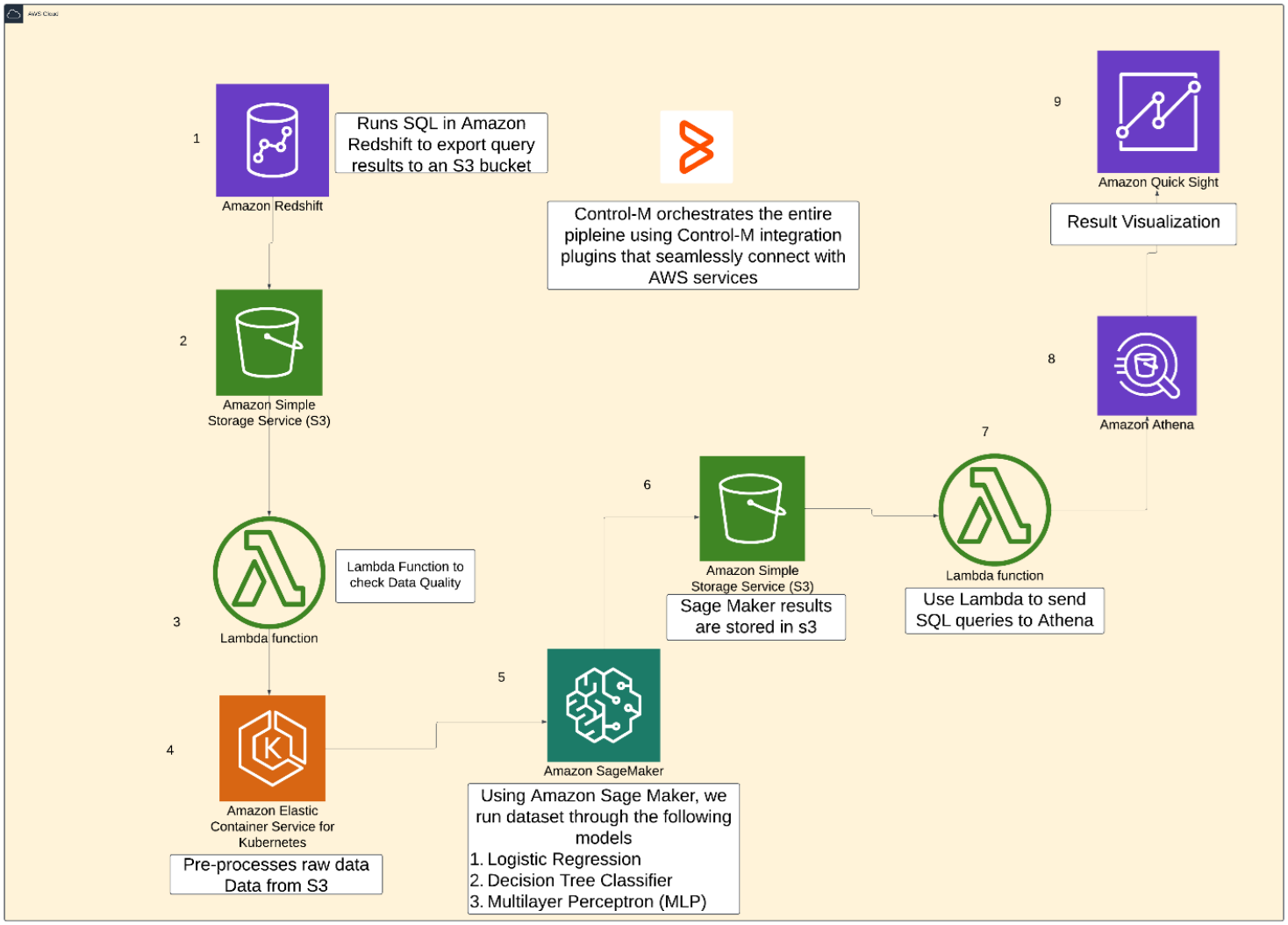


Figure 2. Pipeline archetecture.

Note: All of the code artifacts used are available at this link.

Control-M Integration Plug-ins Used in This Architecture

To orchestrate this analysis pipeline, we leverage Control-M integration plug-ins that seamlessly connect with various platforms and services, including:

1. **Control-M for SageMaker**:
   - Executes ML model training jobs on Amazon SageMaker.
   - Enables integration with SageMaker to trigger training jobs, monitor progress, and retrieve outputs.
2. **Control-M for Kubernetes**:
   - Executes Python scripts for data processing and normalization within an Amazon EKS environment.
   - Ideal for running containerized jobs as part of the data preparation process.
3. **Control-M Managed File Transfer**:
   - Facilitates file movement between Amazon S3 buckets and other storage services.
   - Ensures secure and automated data transfers to prepare for analysis.
4. **Control-M Databases**:
   - Enables streamlined job scheduling and execution of SQL scripts, stored procedures, and database management tasks across multiple database platforms, ensuring automation and consistency in database operations.
5. **Control-M for AWS Lambda:**
   - Enables seamless scheduling and execution of AWS Lambda functions, allowing users to automate serverless workflows, trigger event-driven processes, and manage cloud-based tasks efficiently.
   - Ensures **orchestration, monitoring, and automation** of Lambda functions within broader enterprise workflows, improving operational efficiency and reducing manual intervention

AWS Services used in this Architecture

1. **Amazon S3:** Amazon S3 is a completely managed Object Storage service.
2. **Amazon SageMaker:** Amazon SageMaker is a fully managed ML service.
3. **Amazon EKS:** Amazon Elastic Kubernetes Service (Amazon EKS) is a fully managed Kubernetes service that enables you to run Kubernetes seamlessly in both AWS Cloud and on-premises data centers.
4. **Amazon Redshift:** Amazon Redshift is a popular Cloud Data Warehouse provided by AWS.

Setting up the Kubernetes environment

The Amazon EKS Kubernetes environment is central to the pipeline's data preprocessing stage. It runs Python scripts to clean, normalize, and structure the data before it is passed to the ML models. Setting up the Kubernetes environment involves the following steps:

- **Amazon EKS Cluster Setup**:

    - Use Terraform to create an Amazon EKS cluster or set up your Kubernetes environment in any cloud provider.
    - Ensure the Kubernetes nodes can communicate with the cluster and vice versa.
- **Containerized Python Script**:
    - Build and push a container image to the Amazon Elastic Container Registry (ECR) for the preprocessing script.
    - Deploy a Kubernetes Job to execute the script.

For a detailed guide on setting up a Kubernetes environment for similar workflows, refer to this blog, where we described the Kubernetes setup process step by step.

For a comprehensive walkthrough on setting up Snowflake in similar pipelines, please refer to this blog.

Workflow summary

1. **Redshift_Unload Job**

- **Action:** Executes a SQL script (copy_into_s3.sql) to extract structured data from **Amazon Redshift** and store it in **Amazon S3**.
- **Purpose:** Moves raw data into an accessible S3 bucket for subsequent transformations.
- **Next Step:** Triggers S3_to_S3_Transfer to move data from the warehouse bucket to the processing bucket.

2. **S3_to_S3_Transfer Job**

- **Action:** Uses **Control-M's Managed File Transfer (MFT)** to move the dataset from the sam-sagemaker-warehouse-bucket to the bf-sagemaker bucket.
- **Purpose:** Ensures the data is available in the right location for preprocessing and renames it to Synthetic_Financial_datasets_log.csv.
- **Next Step:** Triggers the Data_Quality_Check job.

3. **Data_Quality_Check Job**

- **Action:** Runs an **AWS Lambda function** (SM_ML_DQ_Test) to validate the dataset.
- **Purpose:** Ensures the CSV file contains at least **5 columns** and **more than 1000 rows**, preventing corrupt or incomplete data from entering the ML pipeline.
- **Next Step:** Triggers EKS-Preprocessing-job for data transformation.

4. **EKS-Preprocessing-Job**

- **Action:** Executes a Kubernetes **job** to clean and transform the dataset stored in **Amazon** S3.
- **Purpose:**
    - Runs a **Python script (main.py) inside a container** to process Synthetic_Financial_datasets_log.csv
    - Generates a **cleaned and structured dataset** (processed-data/output.csv).
- **Configuration Details:**
    - **Image:** new-fd-repo stored in **Amazon ECR**
    - **Environmental variables:** Defines S3 input/output file locations
    - **Resource allocation:** Uses 2Gi memory, 1 CPU (scales up to 4Gi memory, 2 CPUs)

    - **IAM permissions:** Uses a Kubernetes **service account** for S3 access
    - **Logging & cleanup:** Retrieves logs and deletes the job upon completion
- **Next step:** Triggers the **Amazon SageMaker training job**.

5. **Amazon SageMaker_TE_Pipeline**

- **Action:** Runs the TrainingAndEvaluationPipeline in **Amazon SageMaker**.
- **Purpose:**
    - Trains and evaluates **multiple ML models** on the preprocessed dataset (processed-data/output.csv).
    - Stores trained model artifacts and evaluation metrics in an **S3 bucket**.
    - Ensures **automatic resource scaling** for efficient processing.
- **Next step:** Triggers Load_Amazon_Athena_Table to store results in Athena for visualization.

6. **Load_Amazon_Athena_Table Job**

- **Action:** Runs an **AWS Lambda function (athena-query-lambda)** to load the evaluation metrics into **Amazon Athena**.
- **Purpose:**
    - Executes a **SQL query** to create/update an **Athena table (evaluation_metrics)**.
    - Allows **QuickSight** to query and visualize the model performance results.

How the steps are connected

1. **Redshift S3 Transfer:** Data is extracted from **Amazon Redshift** and moved to **Amazon S3**.
2. **Data validation and preprocessing:** The **data quality check** ensures clean input before transformation using **Kubernetes**.
3. **ML Training: Amazon SageMaker** trains and evaluates multiple ML models.
4. **Athena and QuickSight integration:** The model evaluation results are queried through **Athena**, enabling real-time **visualization in Amazon QuickSight.**
5. **Final outcome:** A streamlined, automated ML workflow that delivers a **trained model** and **performance insights** for further decision-making.

This detailed workflow summary ties each step together while emphasizing the critical roles played by the Kubernetes preprocessing job and the Amazon SageMaker training pipeline.

Control-M workflow definition

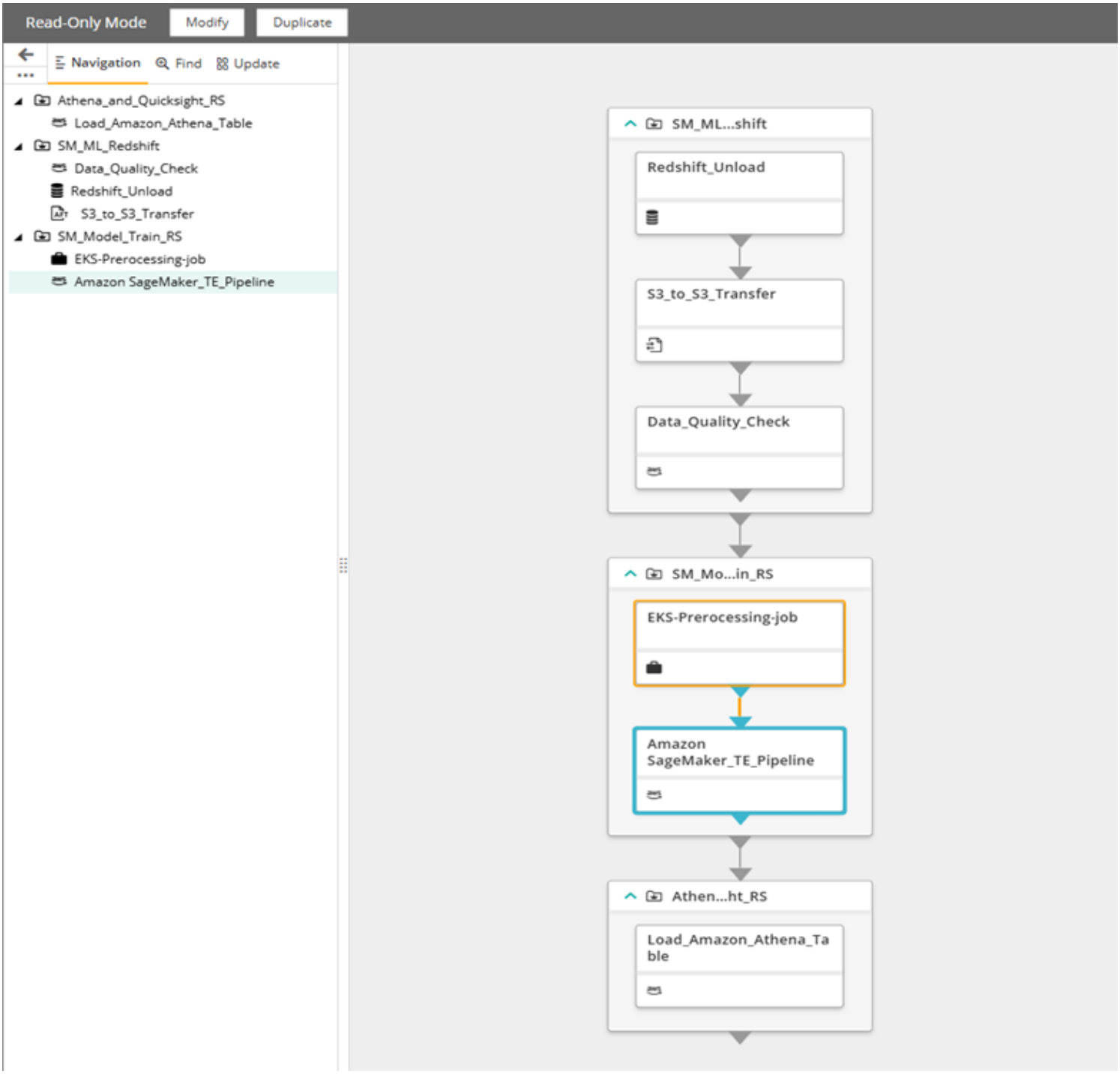


Figure 3. Control-M workflow definition.

In the next section we will go through defining each of these jobs. The jobs can be defined using a drag-and-drop, no-code approach in the Planning domain of Control-M, or they can be defined as code in JSON. For the purposes of this blog, we will use the as-code approach.

Amazon Redshift and file transfer workflows

Redshift_Unload Job

Type: Job:Database:SQLScript

Action: Executes a SQL script in Amazon Redshift to unload data from Redshift tables into an S3 bucket.

Description: This job runs a predefined SQL script (copy_into_s3.sql) stored on the Control-M agent

to export structured data from Redshift into Amazon S3. The unloaded data is prepared for subsequent processing in the ML pipeline.

Dependencies: The job runs independently but triggers the Copy_into_bucket-TO-S3_to_S3_MFT-262 event upon successful completion.

Key configuration details: Redshift SQL script execution

SQL script:

```
UNLOAD ('SELECT * FROM SageTable')
TO 's3://sam-sagemaker-warehouse-bucket/Receiving Folder/Payments_RS.csv'
IAM_ROLE 'arn:aws:iam::xyz:role/jogoldbeRedshiftReadS3'
FORMAT AS csv
HEADER
ALLOWOVERWRITE
PARALLEL OFF
DELIMITER ','
MAXFILESIZE 6GB; -- 1GB max per file
```

Event handling

- Events to trigger:
  - Copy_into_bucket-TO-S3_to_S3_MFT-262 Signals that the data has been successfully unloaded to S3 and is ready for further processing or transfers.

See an example below:

```
"Redshift_Unload" : {
"Type" : "Job:Database:SQLScript",
"ConnectionProfile" : "ZZZ-REDSHIFT",
"SQLScript" : "/home/ctmagent/redshift_sql/copy_into_s3.sql",
"Host" : "<<host details>>",
"CreatedBy" : "<<creator’s email>>",
"RunAs" : "ZZZ-REDSHIFT",
"Application" : "SM_ML_RS",
"When" : {
"WeekDays" : ,
"MonthDays" : ,
"DaysRelation" : "OR"
},
"eventsToAdd" : {
"Type" : "AddEvents",
"Events" :
}
}
```

S3_to_S3_Transfer job

Type: Job:FileTransfer

Action: Transfers a file from one S3 bucket to another using Control-M Managed File Transfer (MFT).

Description: This job moves a dataset (Payments_RS.csv000) from sam-sagemaker-warehouse-bucket to bf-sagemaker, renaming it as Synthetic_Financial_datasets_log.csv in the process. This prepares the data for further processing and validation.

Dependencies: The job waits for Copy_into_bucket-TO-S3_to_S3_MFT-262 to ensure that data has been successfully exported from Redshift and stored in S3 before initiating the transfer.

Key configuration details:

- **Source bucket:** sam-sagemaker-warehouse-bucket
  - **Source path:** /Receiving Folder/Payments_RS.csv000
- **Destination bucket: bf-sagemaker**
  - **Destination path:** /temp/
  - **Renamed file:** Synthetic_Financial_datasets_log.csv at the destination.
- **Connection profiles:** Uses the MFTS3 profile for both the source and destination S3 buckets.
- **File watcher:** Monitors the source file for readiness with a minimum detected size of 200 MB.

Event handling

- **Events to wait for:**
  - Copy_into_bucket-TO-S3_to_S3_MFT-262 Ensures data has been exported from Redshift to S3 before transferring it to another S3 bucket.
- **Events to trigger:**
  - S3_to_S3_MFT-TO-Data_Quality_Check Notifies the next step that the dataset is ready for validation.
  - SM_ML_Snowflake_copy-TO-SM_Model_Train_copy Signals the beginning of the model training process using the processed data.

See an example below:

```
"S3_to_S3_Transfer" : {
"Type" : "Job:FileTransfer",
"ConnectionProfileSrc" : "MFTS3",
"ConnectionProfileDest" : "MFTS3",
"S3BucketNameSrc" : "sam-sagemaker-warehouse-bucket",
"S3BucketNameDest" : "bf-sagemaker",
"Host" : : "<<host details>>",
"CreatedBy" : : "<<creator’s email>>",
"RunAs" : "MFTS3+MFTS3",
"Application" : "SM_ML_RS",
"Variables" : ,
"FileTransfers" : ,
"When" : {
"WeekDays" : ,
"MonthDays" : ,
```

```
"DaysRelation" : "OR"
},
"eventsToWaitFor" : {
"Type" : "WaitForEvents",
"Events" :
},
"eventsToAdd" : {
"Type" : "AddEvents",
"Events" :
},
"eventsToDelete" : {
"Type" : "DeleteEvents",
"Events" :
}
},
"eventsToAdd" : {
"Type" : "AddEvents",
"Events" :
}
}
```

Data_Quality_Check job

Type: Job:AWS Lambda

Action: Executes an AWS Lambda function to perform a data quality check on a CSV file.

Description: This job invokes the Lambda function **SM_ML_DQ_Test** to validate the structure and integrity of the dataset. It ensures that the CSV file has at least **5 columns** and contains more than **1,000 rows** before proceeding with downstream processing. The job logs execution details for review.

Dependencies: The job waits for the event **S3_to_S3_MFT-TO-Data_Quality_Check**, ensuring that the file transfer between S3 buckets is complete before running data validation.

Key configuration details:

- **Lambda function name:** SM_ML_DQ_Test
- **Execution environment:**
  - **Host:** Runs on ip-172-31-18-169.us-west-2.compute.internal
  - **Connection profile:** JOG-AWS-LAMBDA
  - **RunAs:** JOG-AWS-LAMBDA
- **Validation criteria:**
  - The CSV file **must have at least 5 columns**.
  - The CSV file **must contain more than 1,000 rows**.
- **Logging:** Enabled (Append Log to Output: checked) for debugging and validation tracking.

Event handling:

- **Events to wait for:**

    - The job waits for **S3_to_S3_MFT-TO-Data_Quality_Check** to confirm that the dataset has been successfully transferred and is available for validation.
- **Events to delete:**
    - The event **S3_to_S3_MFT-TO-Data_Quality_Check** is deleted after processing to ensure workflow continuity and prevent reprocessing.

See an example below:

```
"Data_Quality_Check" : {
      "Type" : "Job:AWS Lambda",
      "ConnectionProfile" : "JOG-AWS-LAMBDA",
      "Append Log to Output" : "checked",
      "Function Name" : "SM_ML_DQ_Test",
      "Parameters" : "{}",
      "Host" : : "<<host details>>",
      "CreatedBy" : : "<<creator's email>>",
      "Description" : "This job performs a data quality check on CSV file to
make sure it has at least 5 columns and more than 1000 rows",
      "RunAs" : "JOG-AWS-LAMBDA",
      "Application" : "SM_ML_RS",
      "When" : {
        "WeekDays" : ,
        "MonthDays" : ,
        "DaysRelation" : "OR"
      },
      "eventsToWaitFor" : {
        "Type" : "WaitForEvents",
        "Events" :
      },
      "eventsToDelete" : {
        "Type" : "DeleteEvents",
        "Events" :
      }
    }
```

Amazon SageMaker: Model training and evaluation workflows

EKS-Preprocessing-Job

Type: Job:Kubernetes

Action: Executes a Kubernetes job on an Amazon EKS cluster to preprocess financial data stored in an S3 bucket.

Description: This job runs a containerized Python script that processes raw financial datasets stored in **bf-sagemaker**. It retrieves the input file **Synthetic_Financial_datasets_log.csv**, applies necessary transformations, and outputs the cleaned dataset as **processed-data/output.csv**. The Kubernetes job ensures appropriate resource allocation, security permissions, and logging for monitoring.

Dependencies: The job runs independently but triggers the **sagemaker-preprocessing-job-TO-AWS_SageMaker_Job_1-751-262** event upon completion, signaling that the processed data is ready for model training in SageMaker.

Key configuration details:

Kubernetes job specification

- **Image:** 623469066856.dkr.ecr.us-west-2.amazonaws.com/new-fd-repo
- **Command execution:** Runs the following script inside the container:

bash

CopyEdit

python3 /app/main.py -b bf-sagemaker -i Synthetic_Financial_datasets_log.csv -o processed-data/output.csv

- **Environment variables:**
  - S3_BUCKET: bf-sagemaker
  - S3_INPUT_FILE: Synthetic_Financial_datasets_log.csv
  - S3_OUTPUT_FILE: processed-data/output.csv

Resource allocation

- **Requested resources:** 2Gi memory, 1 CPU
- **Limits:** 4Gi memory, 2 CPUs
- **Volume mounts:** Temporary storage mounted at /tmp

Execution environment

- **Host:** Runs on mol-agent-installation-sts-0
- **Connection profile:** MOL-K8S-CONNECTION-PROFILE for EKS cluster access
- **Pod logs:** Configured to retrieve logs upon completion (Get Pod Logs: Get Logs)
- **Job cleanup:** Deletes the Kubernetes job after execution (Job Cleanup: Delete Job)

Event handling:

- **Events to trigger:**
  - **sagemaker-preprocessing-job-TO-AWS_SageMaker_Job_1-751-262** Signals that the preprocessed data is ready for **SageMaker model training**.

See an example below:

```
"EKS-Prerocessing-job" : {
      "Type" : "Job:Kubernetes",
      "Job Spec Yaml" : "apiVersion: batch/v1\r\nkind: Job\r\nmetadata:\r\n
name: s3-data-processing-job\r\nspec:\r\n  template:\r\n    spec:\r\n
serviceAccountName: default  # Ensure this has S3 access via IAM\r\n
containers:\r\n      - name: data-processing-container\r\n        image:
623469066856.dkr.ecr.us-west-2.amazonaws.com/new-fd-repo\r\n        command:
\r\n        env:\r\n        - name: S3_BUCKET\r\n          value: \"bf-
sagemaker\"\r\n        - name: S3_INPUT_FILE\r\n          value:
```

```
\"Synthetic_Financial_datasets_log.csv\"\r\n        - name:
S3_OUTPUT_FILE\r\n          value: \"processed-data/output.csv\"\r\n
resources:\r\n          requests:\r\n            memory:
\"2Gi\"\r\n            cpu: \"1\"\r\n          limits:\r\n            memory:
\"4Gi\"\r\n            cpu: \"2\"\r\n        volumeMounts:\r\n        - name:
tmp-storage\r\n          mountPath: /tmp\r\n      restartPolicy:
Never\r\n      volumes:\r\n      - name: tmp-storage\r\n        emptyDir:
{}\r\n\r\n",
      "ConnectionProfile" : "MOL-K8S-CONNECTION-PROFILE",
      "Get Pod Logs" : "Get Logs",
      "Job Cleanup" : "Delete Job",
      "Host" : : "<<host details>>",
      "CreatedBy" : : "<<creator's email>>",
      "RunAs" : "MOL-K8S-CONNECTION-PROFILE",
      "Application" : "SM_ML_RS",
      "When" : {
        "WeekDays" : ,
        "MonthDays" : ,
        "DaysRelation" : "OR"
      },
      "eventsToAdd" : {

        "Type" : "AddEvents",
        "Events" :
      }
    }
```

Amazon SageMaker_TE_Pipeline job

Type: Job:Amazon Sagemaker

Action: Executes an **Amazon Sagemaker training and evaluation pipeline** to train ML models using preprocessed financial data.

Description: This job runs the **TrainingAndEvaluationPipeline**, which trains and evaluates ML models based on the preprocessed dataset stored in **bf-sagemaker**. The pipeline automates model training, hyperparameter tuning, and evaluation, ensuring optimal performance before deployment.

Dependencies: The job waits for the event **sagemaker-preprocessing-job-TO-AWS_SageMaker_Job_1-751-262**, ensuring that the preprocessing job has completed and the cleaned dataset is available before training begins.

Key configuration details:

- **SageMaker pipeline name:** TrainingAndEvaluationPipeline
- **Execution environment:**
  - **Host:** Runs on prodagents
  - **Connection profile:** MOL-SAGEMAKER-CP for SageMaker job execution
  - **RunAs:** MOL-SAGEMAKER-CP

- **Pipeline parameters:**
  - **Add parameters:** unchecked (defaults used)
  - **Retry pipeline execution:** unchecked (will not automatically retry failed executions)

Event handling:

- **Events to wait for:**
  - **sagemaker-preprocessing-job-TO-AWS_SageMaker_Job_1-751-262** Ensures that the preprocessed dataset is available before initiating training.
- **Events to delete:**
  - **sagemaker-preprocessing-job-TO-AWS_SageMaker_Job_1-751-262** Removes dependency once training begins.
  - **SM_ML_Snowflake-TO-AWS_SageMaker_Job_1** Cleans up previous event dependencies.

See an example below:

```
"Amazon SageMaker_TE_Pipeline" : {
      "Type" : "Job:AWS SageMaker",
      "ConnectionProfile" : "MOL-SAGEMAKER-CP",
      "Add Parameters" : "unchecked",
      "Retry Pipeline Execution" : "unchecked",
      "Pipeline Name" : "TrainingAndEvaluationPipeline",
      "Host" : : "<<host details>>",
      "CreatedBy" : : "<<creator's email>>",
      "RunAs" : "MOL-SAGEMAKER-CP",
      "Application" : "SM_ML_RS",
      "When" : {
        "WeekDays" : ,
        "MonthDays" : ,
        "DaysRelation" : "OR"
      },
      "eventsToWaitFor" : {
        "Type" : "WaitForEvents",
        "Events" :
      },
      "eventsToDelete" : {
        "Type" : "DeleteEvents",
        "Events" :
      }
    }
```

Load_Amazon_Athena_Table job

Type: Job:AWS Lambda

Action: Executes an **AWS Lambda function** to load evaluation results into an **Amazon Athena table** for further querying and visualization.

Description: This job triggers the Lambda function **athena-query-lambda**, which runs an Athena SQL query to create or update a table containing ML evaluation metrics. The table enables seamless integration with **Amazon QuickSight** for data visualization and reporting.

Dependencies: The job waits for the event **SM_Model_Train_copy-TO-Athena_and_Quicksight_copy**, ensuring that the SageMaker training and evaluation process has completed before loading results into Athena.

Key configuration details:

- **Lambda function name:** athena-query-lambda
- **Execution environment:**
  - **Host:** Runs on airflowagents
  - **Connection Profile:** JOG-AWS-LAMBDA
  - **RunAs:** JOG-AWS-LAMBDA
- **Athena table purpose:**
  - Stores **ML model evaluation results**, including accuracy, precision, and recall scores.
  - Enables easy querying of performance metrics through **SQL-based analysis**.
  - Prepares data for **visualization in Amazon QuickSight**.

Event handling:

- **Events to wait for:**
  - **SM_Model_Train_copy-TO-Athena_and_Quicksight_copy** Ensures that the SageMaker training and evaluation process has completed before updating Athena.
- **Events to delete:**
  - **SM_Model_Train_copy-TO-Athena_and_Quicksight_copy** Cleans up the event dependency after successfully loading data.

See an example below:

```
"Load_Amazon_Athena_Table" : {
      "Type" : "Job:AWS Lambda",
      "ConnectionProfile" : "JOG-AWS-LAMBDA",
      "Function Name" : "athena-query-lambda",
      "Parameters" : "{}",
      "Append Log to Output" : "unchecked",
      "Host" : "airflowagents",
      "CreatedBy" : "michael_oladugba@bmc.com",
      "RunAs" : "JOG-AWS-LAMBDA",
      "Application" : "SM_ML_RS",
      "When" : {
        "WeekDays" : ,
        "MonthDays" : ,
        "DaysRelation" : "OR"
      }
    },

    "eventsToWaitFor" : {
      "Type" : "WaitForEvents",
```

```
      "Events" :
    },
    "eventsToDelete" : {
      "Type" : "DeleteEvents",
      "Events" :
    }
  }
```

WORKFLOW EXECUTION:

Training and evaluation steps in Amazon SageMaker

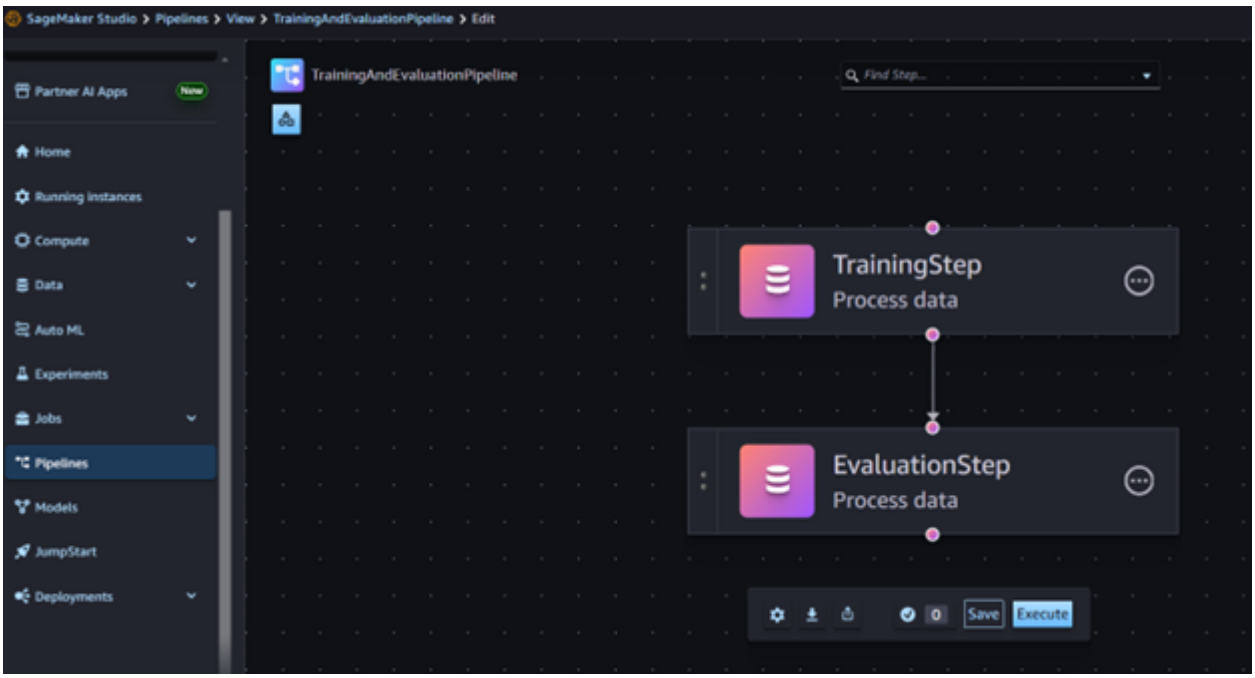


Figure 4. Amazon SageMaker training and evaluation steps.

Pipeline execution logs in CloudWatch:

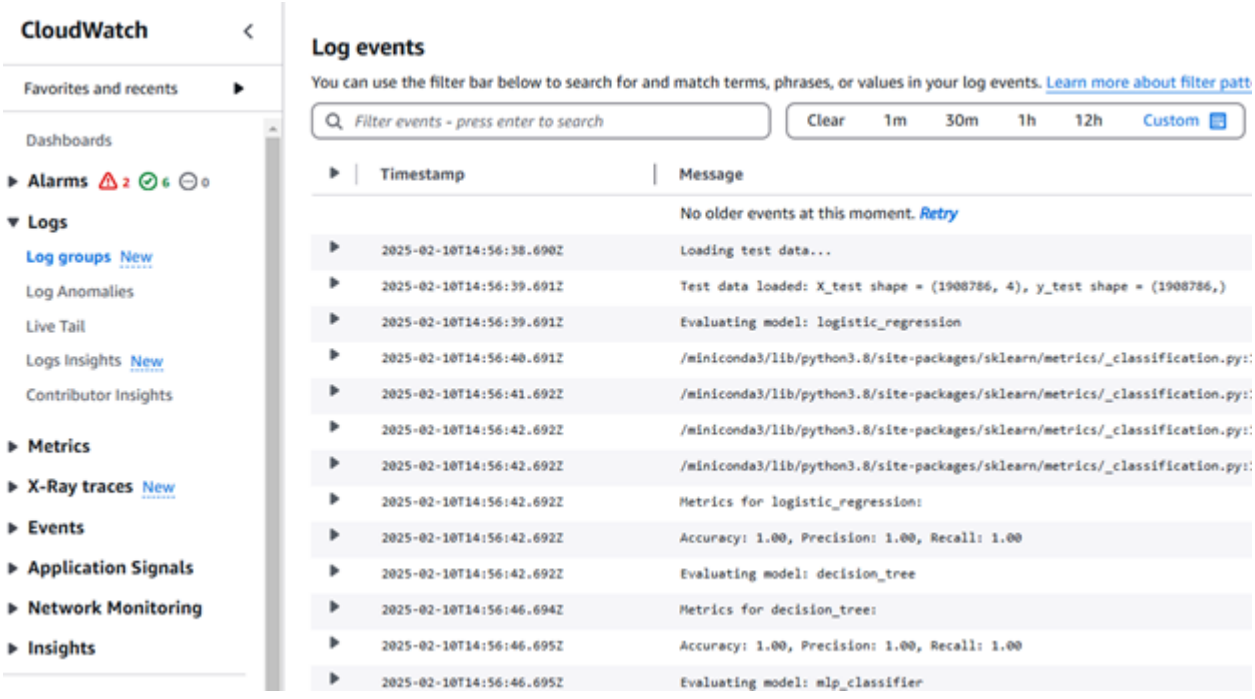


Figure 5. CloudWatch execution logs.

Workflow execution in Control-M

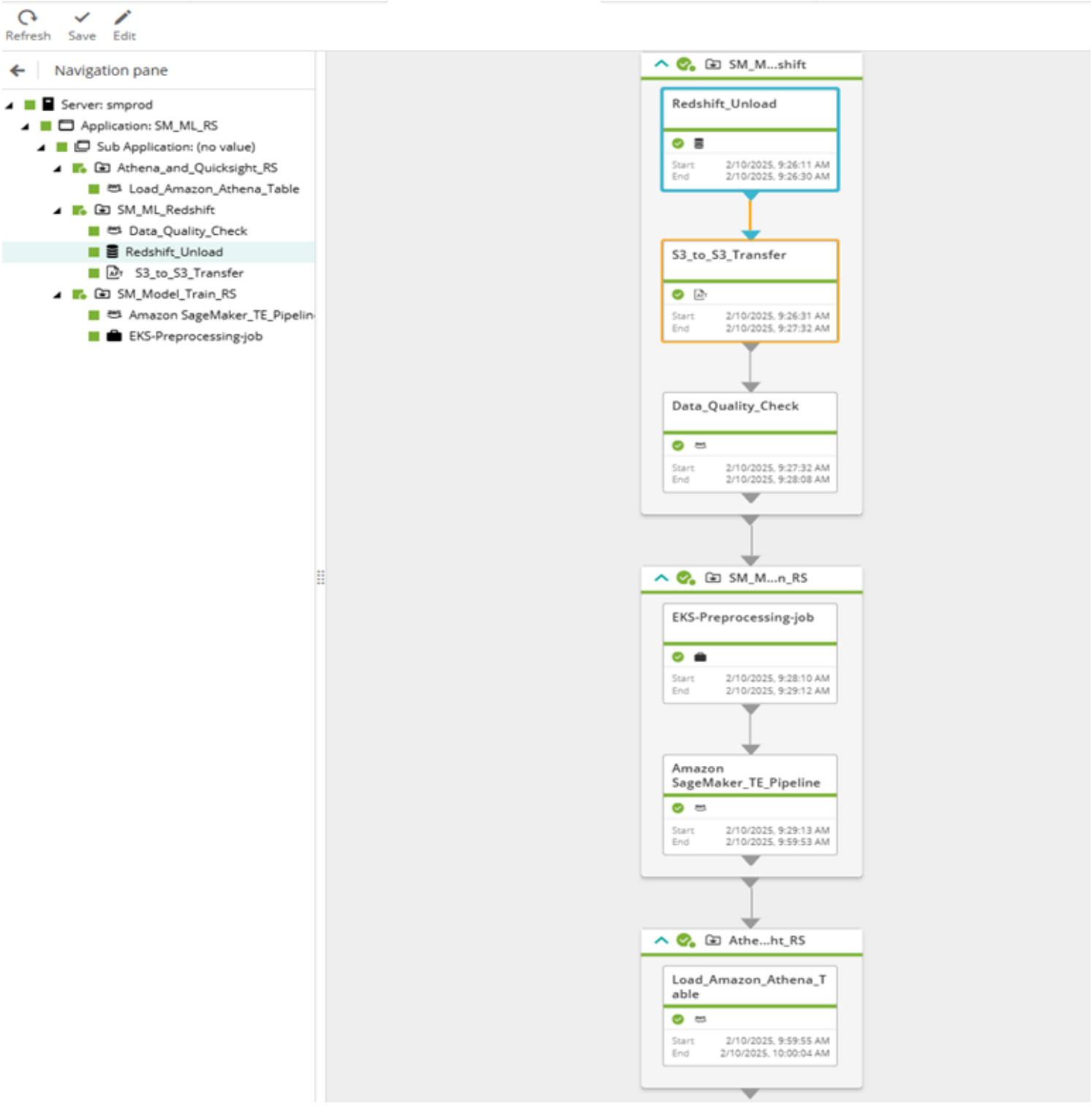


Figure 6. Control-M workflow execution.

The role of Amazon SageMaker:

To analyze the dataset and identify patterns of fraud, we will run the data through three ML models that are available in Amazon SageMaker: **logistic regression**, **decision tree classifier**, and **multi-layer perceptron (MLP)**. Each of these models offers unique strengths, allowing us to evaluate their performance and choose the best approach for fraud detection.

1. Logistic regression: Logistic regression is a linear model that predicts the probability of an

event (e.g., fraud) based on input features. It is simple, interpretable, and effective for binary classification tasks.
2. Decision tree classifier: A decision tree is a rule-based model that splits the dataset into branches based on feature values. Each branch represents a decision rule, making the model easy to interpret and well-suited for identifying patterns in structured data.
3. Multi-layer perceptron: An MLP is a type of neural network designed to capture complex, non-linear relationships in the data. It consists of multiple layers of neurons and is ideal for detecting subtle patterns that may not be obvious in simpler models.

By running the dataset through these models, we aim to compare their performance and determine which one is most effective at detecting fraudulent activity in the dataset. Metrics such as accuracy, precision, and recall will guide our evaluation.

Trainmodels.py:

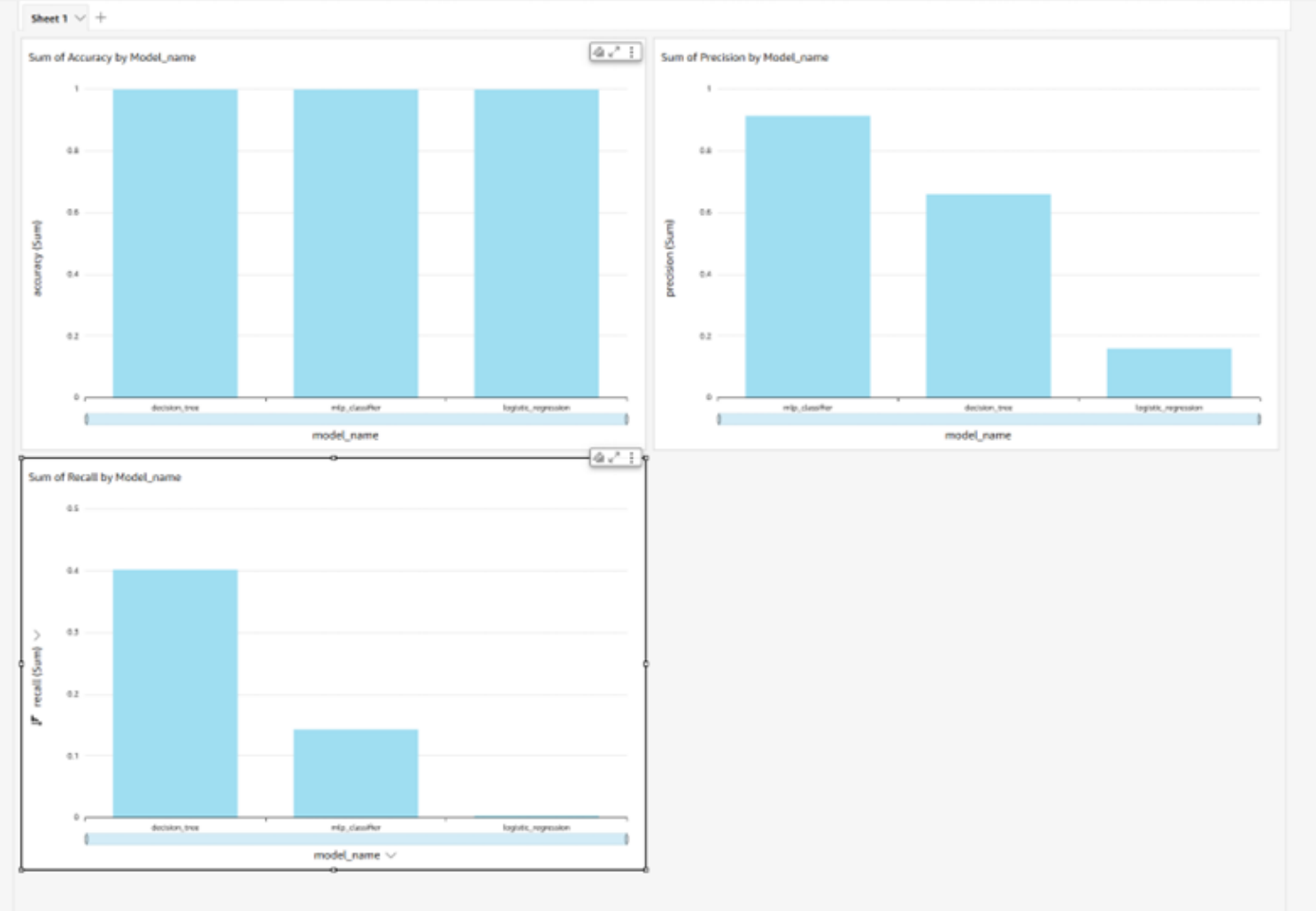
This script processes data to train ML models for fraud detection. It begins by validating and loading the input dataset, ensuring data integrity by handling missing or invalid values and verifying the target column isFraud. The data is then split into training and testing sets, which are saved for future use. The logistic regression, decision tree classifier, and MLP are trained on the dataset, with the trained models saved as .pkl files for deployment or further evaluation. The pipeline ensures robust execution with comprehensive error handling and modularity, making it an efficient solution for detecting fraudulent transactions.

Evaluatemodels.py:

This script evaluates ML models for fraud detection using a test dataset. It loads test data and the three pre-trained models to assess their performance. For each model, it calculates metrics such as accuracy, precision, recall, classification report, and confusion matrix. The results are stored in a JSON file for further analysis. The script ensures modularity by iterating over available models and robustly handles missing files or errors, making it a comprehensive evaluation pipeline for model performance.

Results and outcomes

Model evaluation results in Amazon QuickSight.



Model evaluation results in Amazon QuickSight

The decision tree classifier model shows the most balanced performance with respect to precision and recall, followed by the MLP. Logistic regression performs poorly in correctly identifying positive instances despite its high accuracy.

Summary

Building an automated, scalable, and efficient ML pipeline is essential for combating fraud in today's fast-evolving financial landscape. By leveraging AWS services like Amazon SageMaker, Redshift, EKS, and Athena, combined with Control-M for orchestration, this fraud detection solution ensures seamless data processing, real-time model training, and continuous monitoring.

A key pillar of this workflow is Amazon SageMaker, which enables automated model training, hyperparameter tuning, and scalable inference. It simplifies the deployment of ML models, allowing organizations to train and evaluate multiple models—logistic regression, decision tree classifier, and MLP—to determine the most effective fraud detection strategy. Its built-in automation for training, evaluation, and model monitoring ensures that fraud detection models remain up-to-date, adaptive, and optimized for real-world transactions.

The importance of automation and orchestration cannot be overstated—without it, maintaining a production-grade ML pipeline for fraud detection would be cumbersome, inefficient, and prone to delays. Control-M enables end-to-end automation, ensuring smooth execution of complex workflows, from data ingestion to model training in Amazon SageMaker and evaluation in Athena.

This reduces manual intervention, optimizes resource allocation, and improves overall fraud detection efficiency.

Moreover, model training and evaluation remain at the heart of fraud detection success. By continuously training on fresh transaction data within Amazon SageMaker, adapting to evolving fraud patterns, and rigorously evaluating performance using key metrics, organizations can maintain high fraud detection accuracy while minimizing false positives.

As fraudsters continue to develop new attack strategies, financial institutions and payment processors must stay ahead with adaptive, AI-driven fraud detection systems. By implementing a scalable and automated ML pipeline with Amazon SageMaker, organizations can not only enhance security and reduce financial losses but also improve customer trust and transaction approval rates.