Report on Azure Data Factory and Machine Learning Pipeline Implementation

Step 1: Creating or Configuring Azure Data Lake Storage Gen2

Objective: Establish a centralized storage for raw data files, enabling hierarchical namespace for better organization and integration with Azure services like Data Factory and Machine Learning.

Actions Taken:

Navigated to the Azure Portal and created a new Storage Account with "Azure Data Lake Storage Gen2".

Configured the account with standard settings: Selected a subscription, resource group, and region.

Set up access controls: Assigned roles such as "Storage Blob Data Contributor" And service principal for read/write operations.

Created containers (root container for raw data) to organize files. This storage acts as the "data lake".

Validation:
Tested connectivity by uploading a sample file via the Azure Portal's Storage Explorer.

Outcome: A secure, scalable storage ready for data ingestion. This step ensures compliance with Azure's data governance features.

Step 2: Configuring Datasets for Source and Sink

Objective: Define the input (source) and output (sink) datasets for the data pipeline.
The source is an HTTP-based CSV file, and the sink is the Data Lake Storage for persisting the data as "output.csv".

Actions Taken:

In Azure Data Factory Studio, created a new Dataset for the source:
Typed: "DelimitedText" linked to an HTTP linked service.
Configured the linked service with the HTTP endpoint raw web URL from github hosting the CSV file.

Specified schema: Delimited text with comma separators, first row as header, etc.

Created a Dataset for the sink:

Type: "DelimitedText2" linked to Azure Data Lake Storage Gen2 linked service.

Configured the file path to output as "output.csv" in a container (/raw/output.csv).

Validation:
Used the "Preview data" feature in ADF Studio to sample the source dataset, confirming the CSV structure (columns, rows).

Outcome: Datasets ready for pipeline integration, bridging external HTTP data to Azure storage.

Step 3: Creating and Configuring the Data Pipeline

Objective: Build a pipeline to copy data from the HTTP source to the Data Lake sink.

Actions Taken:

In ADF Studio, created a new Pipeline named "pipeline-data".
Added a "Copy data" activity (named "Copy data1").
Source: Linked to "IngestCSVToLake" (HTTP/DelimitedText1 dataset). Configured any necessary mappings or transformations .

Sink: Linked to "DelimitedText2" (Data Lake dataset), outputting to "output.csv".

Settings: Enabled "Quote all text" if needed for CSV integrity; set "Max concurrent connections" for performance; configured "Block size (MB)" for efficient data transfer.

No additional activities like transformations were added, keeping it a simple ingestion pipeline.

Validation:

Ran "Validate all" in the pipeline editor to check for configuration errors.

Debugged the pipeline: Triggered a debug run to simulate execution, monitoring for issues like authentication failures or data truncation.

Published the pipeline after validation.

Outcome: A functional pipeline that ingests CSV data from HTTP and stores it in Azure Data Lake Gen2 as "output.csv".
This data is now accessible via path in azure.

Step 4: Executing and Monitoring the Pipeline

Objective: Run the pipeline to populate the data lake.

Actions Taken:
Triggered the pipeline manually via "Add trigger".
Monitored execution in the "Monitor" tab, checking activity runs for success (green status).

Validation:
Verified output in Storage Explorer: Confirmed "output.csv" exists with the expected row count and content.
Checked logs for any warnings (e.g., data type mismatches).

Outcome: Data successfully ingested, ready for ML consumption.

Section 2: Machine Learning Pipeline in Azure Machine Learning

With data in Azure Data Lake Gen2, i transitioned to Azure Machine Learning Studio to build a regression-based ML pipeline. The pipeline uses Designer (no-code/low-code interface) to process data, train models, score, and evaluate.

Step 1: Setting Up Azure Machine Learning Workspace and Dataset

Objective: Prepare the ML environment and register the ingested data.
Actions Taken:
Created or accessed an AML Workspace in the Azure Portal.
Registered the dataset: In ML Studio, created a new Dataset from the Data Lake path ("output.csv" via Azure Blob Storage).

Type: Tabular (for CSV).

Configured path to Gen2 storage using a datastore.

Validation:
Previewed the dataset in ML Studio to ensure columns and data types are correct.

Outcome: Dataset integrated into AML, usable as input for pipelines.

Step 2: Creating the ML Pipeline in Designer

Objective: Design a pipeline for data preprocessing, model training, and evaluation.
Actions Taken:
In ML Studio's Designer, created a new pipeline draft.

Added modules:

Input: The registered dataset ("output.csv").

Normalize Data: Applied normalization (ZScore) to scale features for better model performance.
Split Data: Divided the dataset into train/test sets (80/20 split, random seed for reproducibility).

Branching for Multiple Models:

Path 1: Linear Regression → Train Model (using train data) → Score Model (on test data).

Path 2: Boosted Decision Tree Regression → Train Model → Score Model.

Evaluate Model: Compared scored results from both models, computing metrics like MAE, RMSE.

Connected modules in a flow: Dataset → Normalize → Split → Train/Score branches → Evaluate.

Configured parameters: For models, set hyperparameters (learning rate for Boosted Tree, regularization for Linear).

Validation:

Used "Validate" in Designer to check connections and configurations.

Outcome: A visual pipeline ready for execution.

Step 3: Submitting and Running the Pipeline

Objective: Execute the pipeline to train and evaluate models.

Actions Taken:

Submitted the pipeline as a job, selecting a compute target (CPU cluster).

Monitored progress in the "Jobs" tab; status changed to "Completed" (green checkmark).

Validation:
Reviewed child jobs and logs for errors.

Outcome: Trained models with outputs like scored datasets and evaluation metrics.

Step 4: Reviewing Metrics and Outputs

Objective: Analyze model performance.

Actions Taken:

In the "Evaluate Model" module's output:
Viewed metrics:
Coefficient of Determination (R^2): 0.8528977 (Linear), 0.9074591 (Boosted).
Mean Absolute Error (MAE): 0.1839895.
Root Mean Squared Error (RMSE): 0.3120005.
Relative Absolute Error (RAE): 0.3533780.
Relative Squared Error (RSE): 0.09254092.

Compared models: Boosted Decision Tree outperformed Linear Regression based on higher R^2 and lower errors.

Optionally, created custom charts in the metrics view.

Validation:

Ensured metrics align with expectations ($R^2 > 0.8$ indicates good fit).

Outcome: Insights into model accuracy, with potential for deployment or iteration.

Summary

i have implemented an end-to-end workflow: Ingesting CSV data via Azure Data Factory into Data Lake Gen2, then building an ML pipeline in Azure ML to train regression models. The Data Factory pipeline focused on simple data copying, while the ML pipeline emphasized preprocessing and evaluation. Total steps ensure data flows securely from external sources to actionable insights.