Report: Time Savings and Reproducibility Using Dagster ML Pipeline

---

1. Introduction

Reproducibility has long been an issue for machine learning workflows built around traditional Jupyter notebooks. Due to code cells being modified, execution orders being rearranged, or having different versions of their data, notebooks will often produce inconsistent results when they are run after a certain period of time. This makes finding errors difficult and leads to longer execution times overall. To overcome these issues, this project created a reproducible and consistent end-to-end machine learning pipeline using **Dagster** (a new data orchestration framework).

---

2. Pipeline Overview

Machine learning pipelines based on the **Superstore dataset** are built to perform specific operations (ops) and are organized in an orderly/defined manner. The stages of the pipeline include the following:

1. Load the dataset
2. Data Preprocessing (date conversion, target variables creation, missing values handling)
3. Exploratory Data Analysis with visualizations
4. Train-Validate-Test Split dataset
5. Model training and evaluation
6. Result Visualization

The following three machine learning models were evaluated:

- Decision Tree Classifier
- Random Forest Classifier
- Logistic Regression

Dagster ensures that you have a clear picture of what comes after one op and creates dependencies to map the flow of data through the entire pipeline.

---

3. Reproducibility and Reruns Using Dagster

In order to demonstrate reproducibility and controlled reruns of the same pipeline, a single pipeline was run multiple times on different input data sets, and all runs had identical logic. This means that no matter what changes are made in the data, there will be no need to change or reorder code, and training the model can be reliably reproduced with Dagster's orchestration of the pipeline.

- **First run:** Using the original Superstore dataset
- **Second run:** Using the modified Superstore dataset

The pipeline code itself was **not modified**. The only thing that was different about each run was the input dataset. For each run, Dagster automatically created a separate execution and recorded which data was used in each run. This means we can safely rerun the same pipeline with different data, without breaking it or having to manually rerun any notebook cell.

A record of each run and its associated detailed step-level execution information will be displayed in the Dagster UI along with the corresponding log and output files.

---

4. Time Comparison: Dagster vs Traditional Notebook Reruns

Rerunning an entire Jupyter notebook to modify data is the standard way of doing things in a Jupyter notebook. This entails having to go through all the cells in order with their original input prior to executing them again. For larger databases and multiple Models this can add many minutes to performance time.

- **Original database pipeline** runtime was around **40 seconds**.

- **Modified database pipeline** was around **44 seconds**.

With Dagster, the overall time is decreased due to:

- The lack of need to rerun cells manually.
- The ability to order the execution of pipelines without having to do it manually.
- Only rerun the portions of your pipeline that have changed when you rerun that part of your notebook.

Based on our estimations, the average amount of time to rerun every cell in a Jupyter notebook is anywhere between **3–5 minutes**, while using the Dagster toolset has been found to save you time and make your process more efficient and reliable.

---

5. Conclusion

Dagster makes it possible to create machine-learning workflows that are reproducible, trustworthy, and efficient. In the course of this project, we took an unstructured notebook-style ML process and created a formal pipeline that allows for easier and much faster reruns. The ability to track execution, visualize pipelines, and maintain a history of each run makes Dagster an excellent tool for real-world machine learning and data science projects that require reproducibility.