Peyton Bailey

D602 – Task 2

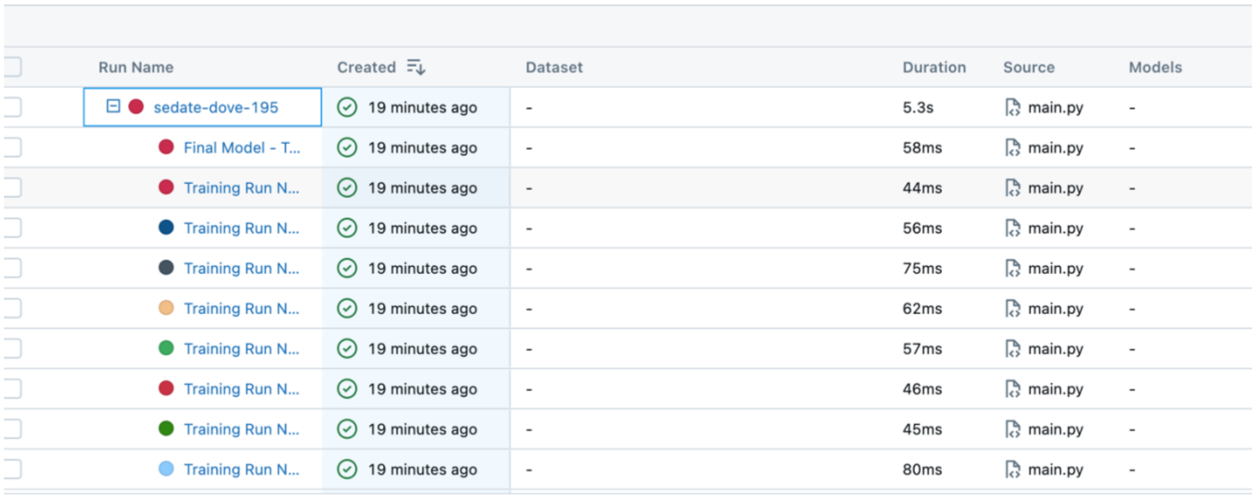
July 28th, 2025

The overall goal of this task was to design and implement a data pipeline using a regression model previously developed by another analyst. The data import and cleaning scripts needed to be created, and the model training/testing script needed to be modified and edited. The first script I wrote for this pipeline was the data import script. I used the template from the poly_regressor script to determine which columns I needed to download. I used the pd.read_csv function within my script and specified 'skip' on the bad lines. This ensured that the data was all uniform and dropped rows that had missing values. I also ran a DVC command to create a metafile for the dataset.

The second script was the data cleaning script. I created a data frame called airline_data, imported directly from the processed_data.csv produced from the import script. Then, I filtered the data to show only departures from LAX. I filtered again for columns that were necessary to keep. Then, the columns were renamed, and data types were adjusted to match the required format to run correctly in the poly_regressor script. One challenge I encountered was negative numbers in my "Departure Delay" column. This represented early departures; negative integers would cause errors during the model training script. I replaced all negative numbers with a zero, using ".clip (lower=0)".

The poly_regressor script was by far the most challenging, due to the complexity and the fact that it was pre-written code that needed to be fixed. Gaining an overall understanding of what was being performed in the script took me some time. The hyper-tuning of the parameters (alpha, order) was logged, and the combination that received the best score was fitted to the training and validated on the test data. One major hurdle was understanding that running the pipeline through the main script and running on the command line are two completely different processes. I had multiple errors with the run ID not aligning with the environment ID. I learned that running MLFlow through the command line sets its experiment ID, which will conflict with an MLFlow start experiment command within the script.

The main script was created to call on the other scripts to run together as a successful pipeline. One challenge I encountered in my scripts was not wrapping them into a primary function that could be called from the main script. This was something I went back and fixed after much troubleshooting. The image below shows the results of a successful run pipeline in the MLFlow UI.



Run Name	Created	Dataset	Duration	Source	Models
⊟ ● sedate-dove-195	19 minutes ago	-	5.3s	main.py	-
● Final Model - T...	19 minutes ago	-	58ms	main.py	-
● Training Run N...	19 minutes ago	-	44ms	main.py	-
● Training Run N...	19 minutes ago	-	56ms	main.py	-
● Training Run N...	19 minutes ago	-	75ms	main.py	-
● Training Run N...	19 minutes ago	-	62ms	main.py	-
● Training Run N...	19 minutes ago	-	57ms	main.py	-
● Training Run N...	19 minutes ago	-	46ms	main.py	-
● Training Run N...	19 minutes ago	-	45ms	main.py	-
● Training Run N...	19 minutes ago	-	80ms	main.py	-