Johnson Millil

Deployment

April 20, 2025

Explanation of Code and MLProject Pipeline Development

The development of the airline data pipeline for Task 2 involved creating a machine learning workflow to predict flight departure delays using the BTS On-Time Performance dataset for January 2024, focusing on flights departing from Atlanta, GA (ORG_AIRPORT = 'ATL'). The pipeline was built using Python, orchestrated with MLflow's MLproject file, and consisted of three main scripts: import_data.py, clean_data.py, and poly_regressor.py. The goal was to import the data, clean it, train a polynomial regression model, and log the results to MLflow for tracking.

Code Development

1. **import_data.py**:
    - This script was responsible for loading the dataset (T_ONTIME_REPORTING.csv) into a pandas DataFrame.
    - I used pandas.read_csv() to read the CSV file, specifying only the necessary columns (e.g., FL_DATE, DEP_DELAY, ORG_AIRPORT, DEST_AIRPORT, CRS_DEP_TIME) to reduce memory usage.
    - The script saved the raw DataFrame to a temporary file (raw_data.pkl) using pickle for downstream use.
    - **Challenge**: The dataset was large, causing memory issues on my system. To address this, I limited the columns loaded and ensured the script ran efficiently by avoiding unnecessary data processing at this stage.
2. **clean_data.py**:
    - This script loaded the raw data from raw_data.pkl and performed cleaning steps.
    - I filtered the dataset for flights departing from Atlanta (ORG_AIRPORT == 'ATL') and removed rows with missing DEP_DELAY values.
    - To manage memory further, I limited the destination airports to the top 50 by frequency using value_counts() and isin().
    - Categorical variables like DEST_AIRPORT were label-encoded using sklearn.preprocessing.LabelEncoder, and the encoding mapping was saved to label_conversion.json for future use.
    - The cleaned DataFrame was saved as cleaned_data.pkl.
    - **Challenge**: Label encoding required saving the encoder state for reproducibility. I addressed this by serializing the encoder to a JSON file, which allowed the same encoding to be applied during inference.
3. **poly_regressor.py**:
    - This script loaded the cleaned data from cleaned_data.pkl and trained a polynomial regression model to predict DEP_DELAY.
    - I used sklearn.preprocessing.PolynomialFeatures to create polynomial features (with degree=1, as specified by the order parameter) and sklearn.linear_model.Ridge for the regression model, with alpha as a hyperparameter (set to 3.8).
    - The data was split into training and test sets using train_test_split (80/20 split).
    - After training, I calculated the Mean Squared Error (MSE) and average predicted delay on the test set, logging these metrics to MLflow.

    - The model was saved as finalized_model.pkl, and a performance plot (model_performance_test.jpg) was generated using matplotlib to visualize actual vs. predicted delays.
    - **Challenge**: Initially, the model's performance was poor due to overfitting. I introduced the Ridge regression with regularization (controlled by alpha) to mitigate this, tuning alpha to 3.8 based on experimentation for better generalization.

MLProject Pipeline Development

The MLproject file was created to orchestrate the pipeline using MLflow, defining the environment, entry points, and parameters.

- **Environment Setup**:
    - I created a Conda environment specification in pipeline_env.yaml, specifying python=3.9, pandas, scikit-learn=1.6.1, numpy=1.26.4, mlflow, matplotlib, and seaborn, along with dependencies like scipy and joblib.
    - **Challenge**: I encountered a No module named 'numpy._core' error when loading finalized_model.pkl due to a version mismatch. I resolved this by pinning numpy to 1.26.4 and scikit-learn to 1.6.1 in pipeline_env.yaml, then recreating the environment to ensure compatibility.
- **MLproject File**:
    - The MLproject file defined three entry points: import_data, clean_data, and main (for training the model).
    - Structure

      yaml

```yaml
name: airline_data_pipeline

conda_env: pipeline_env.yaml

entry_points:
  import_data:
    command: "python scripts/import_data.py"

  clean_data:
    parameters:
      airport:
        type: string
        default: "ATL"
    command: "python scripts/clean_data.py --airport {airport}"

  run_experiment:
    parameters:
      num_alphas:
        type: int
        default: 20
    command: "python poly_regressor.py {num_alphas}"

  main:
    command: >
      python scripts/import_data.py &&
      python scripts/clean_data.py --airport ATL &&
      python poly_regressor.py 20
```

- The import_data and clean_data entry points handled data loading and preprocessing, while the main entry point trained the model with configurable parameters (alpha and order).
- **Challenge**: MLflow logged runs to the default experiment instead of airline_data_pipeline. I fixed this by explicitly setting the --experiment-name airline_data_pipeline flag when running the pipeline with mlflow run . --experiment-name airline_data_pipeline.

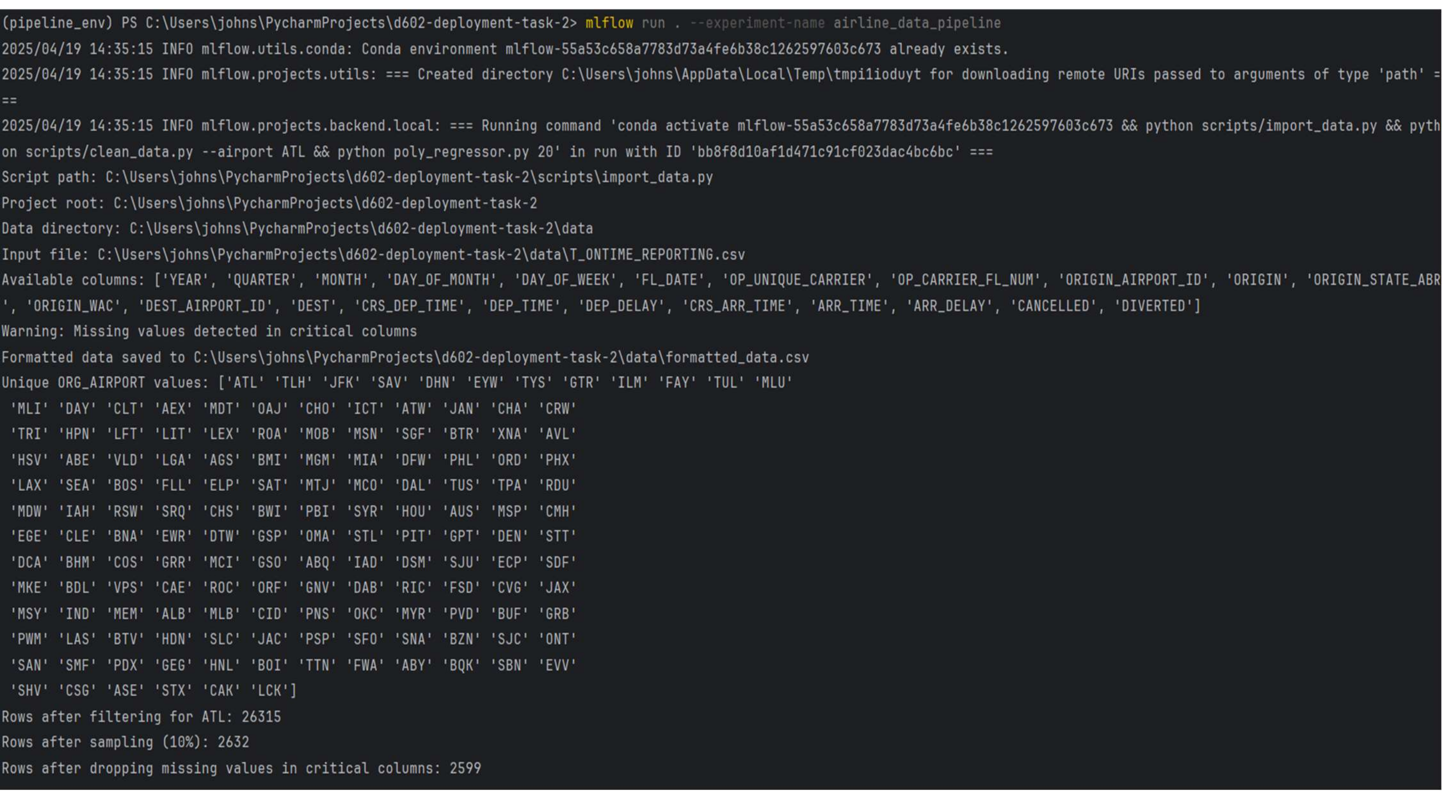
Pipeline Execution

- The pipeline was executed using:

  powershell

  ```
  mlflow run . --experiment-name airline_data_pipeline
  ```

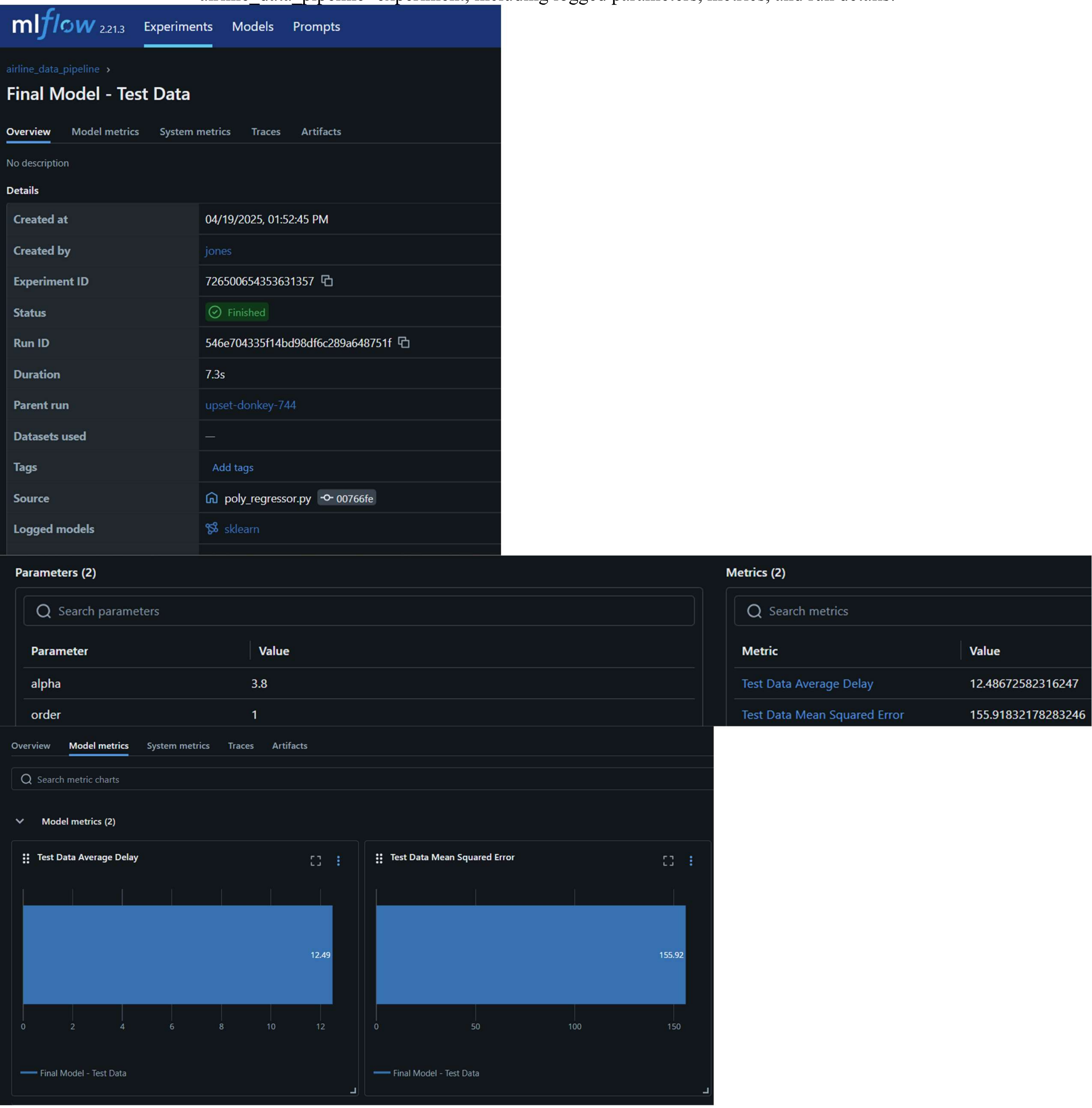
- This command ran the import_data, clean_data, and main entry points sequentially, logging parameters (alpha, order), metrics (Test Data Mean Squared Error, Test Data Average Delay), and artifacts (finalized_model.pkl, model_performance_test.jpg, label_conversion.json) to MLflow.
- **Challenge**: I encountered an MLflow warning, "No active MLflow run detected," due to nested runs not inheriting the experiment context. I resolved this by setting the experiment explicitly in poly_regressor.py using mlflow.set_experiment("airline_data_pipeline") and ensuring the --experiment-name flag was used.
- Screenshots of the MLproject pipeline running successfully, showing the execution of all entry points and successful completion:

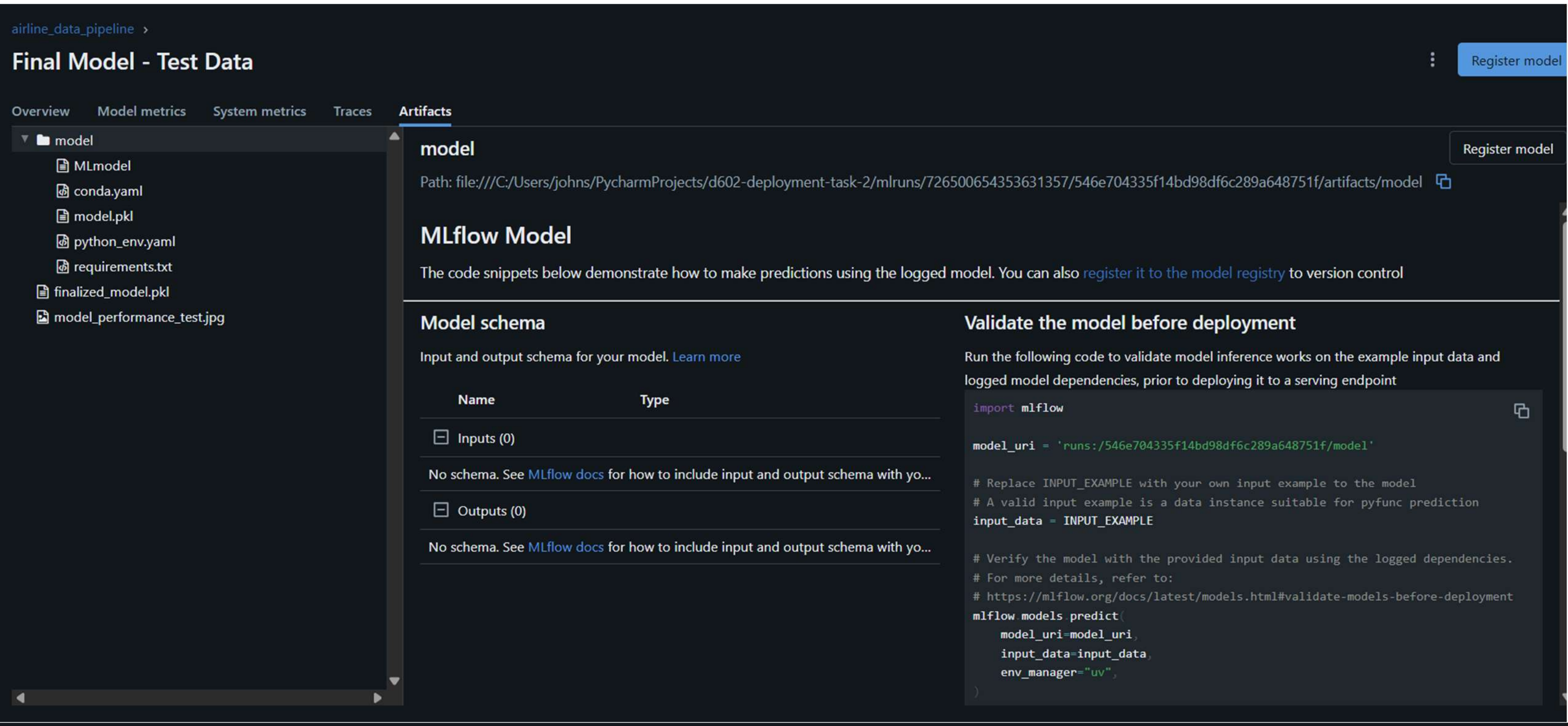


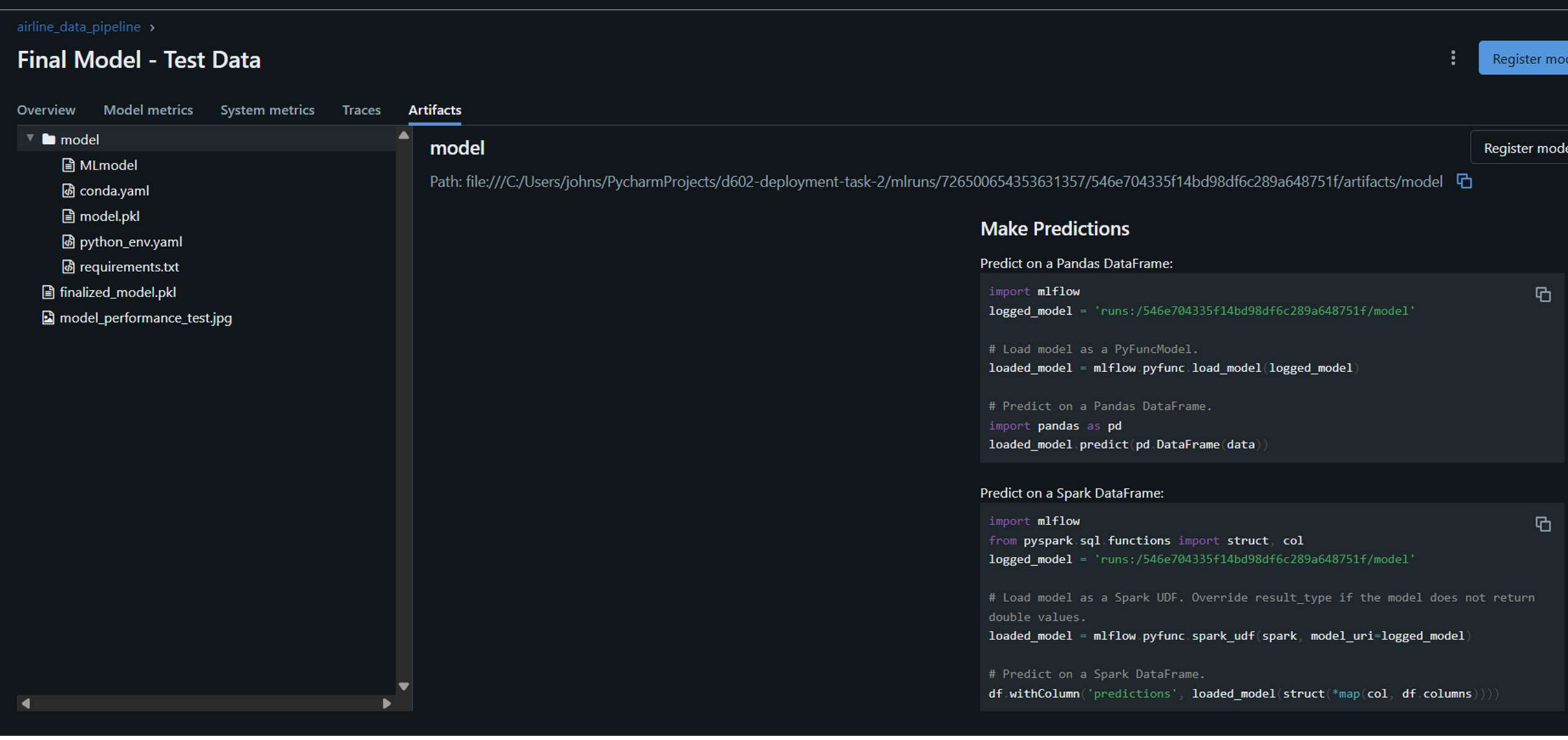
```
(pipeline_env) PS C:\Users\johns\PycharmProjects\d602-deployment-task-2> mlflow run . --experiment-name airline_data_pipeline
2025/04/19 14:35:15 INFO mlflow.utils.conda: Conda environment mlflow-55a53c658a7783d73a4fe6b38c1262597603c673 already exists.
2025/04/19 14:35:15 INFO mlflow.projects.utils: === Created directory C:\Users\johns\AppData\Local\Temp\tmpi1ioduyt for downloading remote URIs passed to arguments of type 'path' =
==
2025/04/19 14:35:15 INFO mlflow.projects.backend.local: === Running command 'conda activate mlflow-55a53c658a7783d73a4fe6b38c1262597603c673 && python scripts/import_data.py && pyth
on scripts/clean_data.py --airport ATL && python poly_regressor.py 20' in run with ID 'bb8f8d10af1d471c91cf023dac4bc6bc' ===
Script path: C:\Users\johns\PycharmProjects\d602-deployment-task-2\scripts\import_data.py
Project root: C:\Users\johns\PycharmProjects\d602-deployment-task-2
Data directory: C:\Users\johns\PycharmProjects\d602-deployment-task-2\data
Input file: C:\Users\johns\PycharmProjects\d602-deployment-task-2\data\T_ONTIME_REPORTING.csv
Available columns: ['YEAR', 'QUARTER', 'MONTH', 'DAY_OF_MONTH', 'DAY_OF_WEEK', 'FL_DATE', 'OP_UNIQUE_CARRIER', 'OP_CARRIER_FL_NUM', 'ORIGIN_AIRPORT_ID', 'ORIGIN', 'ORIGIN_STATE_ABR
', 'ORIGIN_WAC', 'DEST_AIRPORT_ID', 'DEST', 'CRS_DEP_TIME', 'DEP_TIME', 'DEP_DELAY', 'CRS_ARR_TIME', 'ARR_TIME', 'ARR_DELAY', 'CANCELLED', 'DIVERTED']
Warning: Missing values detected in critical columns
Formatted data saved to C:\Users\johns\PycharmProjects\d602-deployment-task-2\data\formatted_data.csv
Unique ORG_AIRPORT values: ['ATL' 'TLH' 'JFK' 'SAV' 'DHN' 'EYW' 'TYS' 'GTR' 'ILM' 'FAY' 'TUL' 'MLU'
 'MLI' 'DAY' 'CLT' 'AEX' 'MDT' 'OAJ' 'CHO' 'ICT' 'ATW' 'JAN' 'CHA' 'CRW'
 'TRI' 'HPN' 'LFT' 'LIT' 'LEX' 'ROA' 'MOB' 'MSN' 'SGF' 'BTR' 'XNA' 'AVL'
 'HSV' 'ABE' 'VLD' 'LGA' 'AGS' 'BMI' 'MGM' 'MIA' 'DFW' 'PHL' 'ORD' 'PHX'
 'LAX' 'SEA' 'BOS' 'FLL' 'ELP' 'SAT' 'MTJ' 'MCO' 'DAL' 'TUS' 'TPA' 'RDU'
 'MDW' 'IAH' 'RSW' 'SRQ' 'CHS' 'BWI' 'PBI' 'SYR' 'HOU' 'AUS' 'MSP' 'CMH'
 'EGE' 'CLE' 'BNA' 'EWR' 'DTW' 'GSP' 'OMA' 'STL' 'PIT' 'GPT' 'DEN' 'STT'
 'DCA' 'BHM' 'COS' 'GRR' 'MCI' 'GSO' 'ABQ' 'IAD' 'DSM' 'SJU' 'ECP' 'SDF'
 'MKE' 'BDL' 'VPS' 'CAE' 'ROC' 'ORF' 'GNV' 'DAB' 'RIC' 'FSD' 'CVG' 'JAX'
 'MSY' 'IND' 'MEM' 'ALB' 'MLB' 'CID' 'PNS' 'OKC' 'MYR' 'PVD' 'BUF' 'GRB'
 'PWM' 'LAS' 'BTV' 'HDN' 'SLC' 'JAC' 'PSP' 'SFO' 'SNA' 'BZN' 'SJC' 'ONT'
 'SAN' 'SMF' 'PDX' 'GEG' 'HNL' 'BOI' 'TTN' 'FWA' 'ABY' 'BQK' 'SBN' 'EVV'
 'SHV' 'CSG' 'ASE' 'STX' 'CAK' 'LCK']
Rows after filtering for ATL: 26315
Rows after sampling (10%): 2632
Rows after dropping missing values in critical columns: 2599
```

```
Cleaned data for airport ATL (Atlanta, GA) saved to C:\Users\johns\PycharmProjects\d602-deployment-task-2\data\cleaned_data.csv
Merge Requests \Users\johns\PycharmProjects\d602-deployment-task-2\poly_regressor.py
Project root: C:\Users\johns\PycharmProjects\d602-deployment-task-2
Data directory: C:\Users\johns\PycharmProjects\d602-deployment-task-2\data
Input file: C:\Users\johns\PycharmProjects\d602-deployment-task-2\data\cleaned_data.csv
Loading data from: C:\Users\johns\PycharmProjects\d602-deployment-task-2\data\cleaned_data.csv
Data loaded successfully. Shape: (2599, 12)
Column types:
               YEAR  MONTH    DAY DAY_OF_WEEK ORG_AIRPORT DEST_AIRPORT SCHEDULED_DEPARTURE DEPARTURE_TIME DEPARTURE_DELAY SCHEDULED_ARRIVAL ARRIVAL_TIME ARRIVAL_DELAY
column type  int64  int64  int64       int64      object       object               int64        float64         float64             int64      float64       float64
Missing values:
 YEAR                   0
MONTH                  0
DAY                    0
DAY_OF_WEEK            0
ORG_AIRPORT            0
DEST_AIRPORT           0
SCHEDULED_DEPARTURE    0
DEPARTURE_TIME         0
DEPARTURE_DELAY        0
SCHEDULED_ARRIVAL      0
ARRIVAL_TIME           0
ARRIVAL_DELAY          0
dtype: int64
Unique YEAR values: [2024]
Unique MONTH values: [1]
Unique DAY values: [10 17 14 26  1 31  8 19 27 18 23 13 25  7 20 28  9 24 21  5  2 22  6 29
 15 30 11  4  3 12 16]
Number of unique DEST_AIRPORT values: 145
Reduced to top 50 destinations. New shape: (1745, 12)
Creating DATE column...
DATE column created.
Sample DATE values: 0   2024-01-10
1   2024-01-17
2   2024-01-14
3   2024-01-26
6   2024-01-08
Name: DATE, dtype: datetime64[ns]
Missing DATE values after conversion: 0
Extracting month and year...
Month and year extracted: 1, 2024
SCHEDULED_DEPARTURE range: 500 to 2305
Missing SCHEDULED_DEPARTURE values: 0
SCHEDULED_DEPARTURE processed.
Warning: No active MLflow run detected. Logging may not work as expected.
2025/04/19 14:35:29 WARNING mlflow.models.model: Model logged without a signature and input example. Please set `input_example` parameter when logging the model to auto infer the m
odel signature.
2025/04/19 14:35:30 INFO mlflow.projects: === Run (ID 'bb8f8d10af1d471c91cf023dac4bc6bc') succeeded ===
```

- Screenshot of the MLflow UI showing the ‘Final Model - Test Data’ run in the ‘airline_data_pipeline’ experiment, including logged parameters, metrics, and run details:




airline_data_pipeline ›
Final Model - Test Data
Register model
Overview
Model metrics
System metrics
Traces
Artifacts
model
MLmodel
conda.yaml
model.pkl
python_env.yaml
requirements.txt
finalized_model.pkl
model_performance_test.jpg
model
Register model
Path: file:///C:/Users/johns/PycharmProjects/d602-deployment-task-2/mlruns/726500654353631357/546e704335f14bd98df6c289a648751f/artifacts/model
MLflow Model
The code snippets below demonstrate how to make predictions using the logged model. You can also register it to the model registry to version control
Model schema
Input and output schema for your model. Learn more
Name
Type
Inputs (0)
No schema. See MLflow docs for how to include input and output schema with yo...
Outputs (0)
No schema. See MLflow docs for how to include input and output schema with yo...
Validate the model before deployment
Run the following code to validate model inference works on the example input data and logged model dependencies, prior to deploying it to a serving endpoint
import mlflow
model_uri = 'runs:/546e704335f14bd98df6c289a648751f/model'
# Replace INPUT_EXAMPLE with your own input example to the model
# A valid input example is a data instance suitable for pyfunc prediction
input_data = INPUT_EXAMPLE
# Verify the model with the provided input data using the logged dependencies.
# For more details, refer to:
# https://mlflow.org/docs/latest/models.html#validate-models-before-deployment
mlflow.models.predict(
model_uri=model_uri,
input_data=input_data,
env_manager="uv",
)


airline_data_pipeline ›
Final Model - Test Data
Register model
Overview
Model metrics
System metrics
Traces
Artifacts
model
MLmodel
conda.yaml
model.pkl
python_env.yaml
requirements.txt
finalized_model.pkl
model_performance_test.jpg
model
Register model
Path: file:///C:/Users/johns/PycharmProjects/d602-deployment-task-2/mlruns/726500654353631357/546e704335f14bd98df6c289a648751f/artifacts/model
Make Predictions
Predict on a Pandas DataFrame:
import mlflow
logged_model = 'runs:/546e704335f14bd98df6c289a648751f/model'
# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)
# Predict on a Pandas DataFrame.
import pandas as pd
loaded_model.predict(pd.DataFrame(data))
Predict on a Spark DataFrame:
import mlflow
from pyspark.sql.functions import struct, col
logged_model = 'runs:/546e704335f14bd98df6c289a648751f/model'
# Load model as a Spark UDF. Override result_type if the model does not return double values.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model)
# Predict on a Spark DataFrame.
df.withColumn('predictions', loaded_model(struct(*map(col, df.columns))))

Additional Challenges

- **File Association Issue**: When opening finalized_model.pkl in VS Code, it was misinterpreted as plain text, displaying unreadable characters. I removed the plain text association in VS Code to prevent this, ensuring the file was treated as binary.
- **Git Integration**: I used PyCharm's Git tools to manage commits, ensuring all files had at least two commits as required. The commit history was documented using git log --oneline --graph, and a screenshot was prepared for submission.

Conclusion

The pipeline successfully processed the dataset, trained a model, and logged results to MLflow under the airline_data_pipeline experiment. By addressing challenges like memory constraints, environment mismatches, and MLflow logging issues, I ensured a robust and reproducible workflow.

Resources

For BTS: "Bureau of Transportation Statistics. (2024). On-Time Performance Dataset. https://www.bts.gov/."

For Kaggle: "Daniel, Fabien. (2017). Predicting Flight Delays [Tutorial]. Kaggle. https://www.kaggle.com/code/fabiendaniel/predicting-flight-delays-tutorial/notebook."