Task 2:

Data Production Pipeline

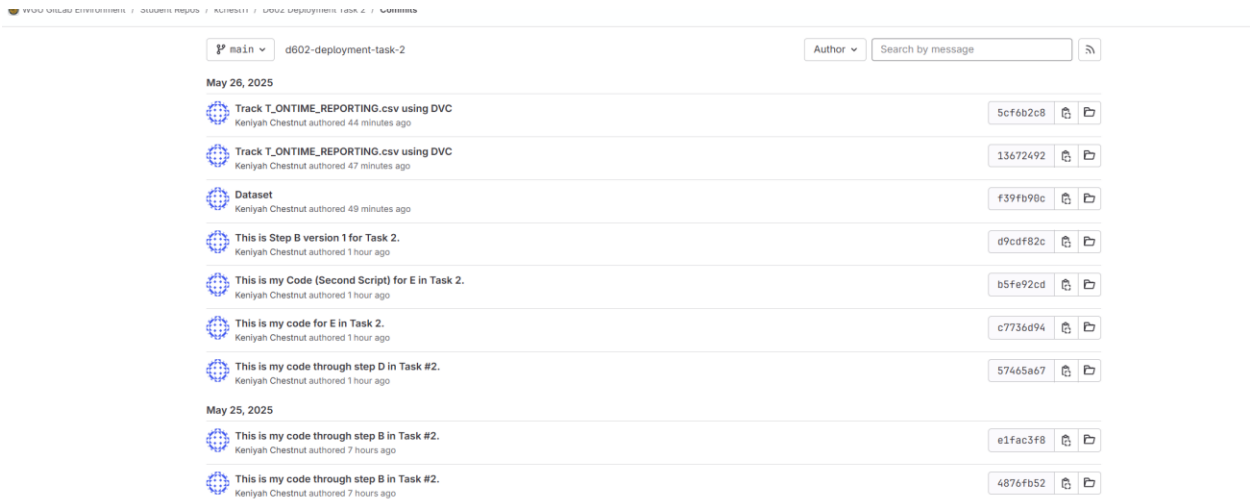
Keniyah Chestnut

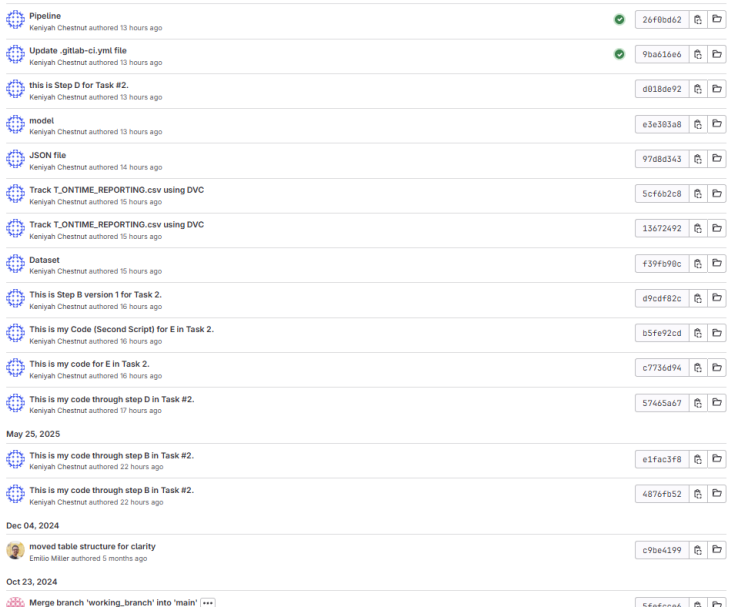
Deployment — D602

SID: 012601305

A: GITLAB REPOSITORY

I created a subgroup and project in GitLab using the provided link. I cloned the repo to my computer, worked in it locally, and pushed my changes as I completed each part of the task. At first, Git wasn't recognized in my terminal, so I had to install it and set up my Git config with my name and email. Once I got that working, I was able to commit and push everything without issues.





B: IMPORT AND FORMAT SCRIPT

I wrote two versions of my import script. The first version just loads the CSV and filters down to the required columns. The second version cleans the data and renames the columns to match what the model script expects. I also used DVC to track the original CSV file and committed the .dvc file to GitLab.

Version 1:

```
[9]: import pandas as pd

# Load the raw data
df = pd.read_csv("T_ONTIME_REPORTING.csv")

# Select relevant columns only
columns_to_keep = [
    'YEAR', 'MONTH', 'DAY_OF_MONTH', 'DAY_OF_WEEK',
    'ORIGIN_AIRPORT_ID', 'DEST_AIRPORT_ID',
    'CRS_DEP_TIME', 'DEP_TIME', 'DEP_DELAY',
    'CRS_ARR_TIME', 'ARR_TIME', 'ARR_DELAY'
]

df = df[columns_to_keep]

# Preview
df.head()
```

[9]:

	YEAR	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN_AIRPORT_ID	DEST_AIRPORT_ID	CRS_DEP_TIME	DEP_TIME	DEP_DELA
0	2025	1	1	3	10135	11057	606	556.0	-10
1	2025	1	1	3	10135	11057	1219	1215.0	-4
2	2025	1	1	3	10135	11057	1738	1730.0	-8
3	2025	1	1	3	10135	14082	830	820.0	-10
4	2025	1	1	3	10135	14112	1551	1547.0	-4

Second Version:

jupyter D602TASK2B Last Checkpoint: 8 hours ago

File Edit View Run Kernel Settings Help

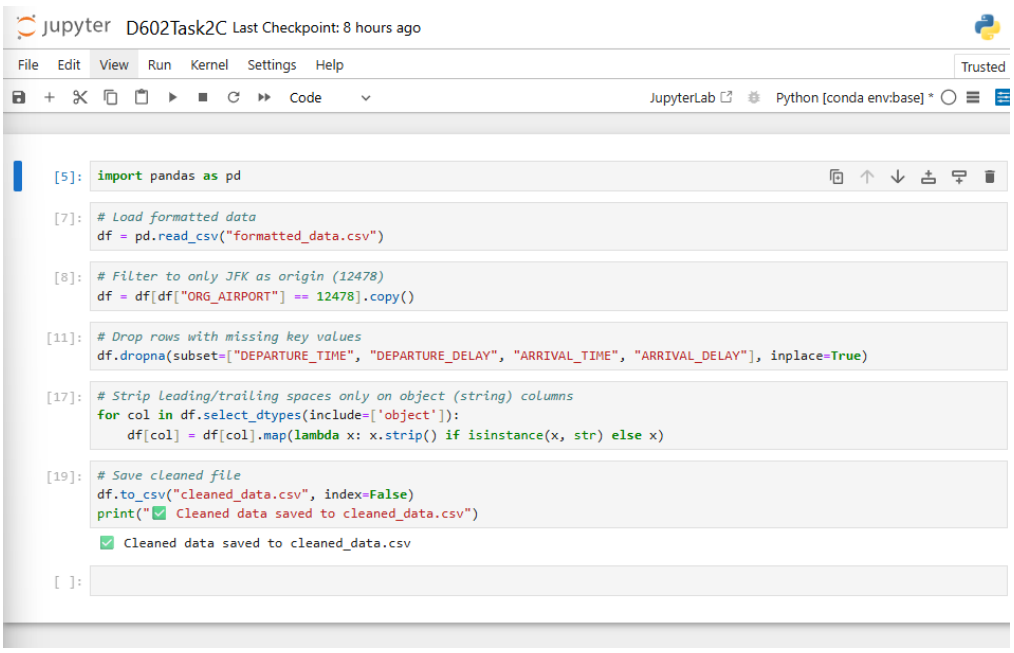
Code JupyterLab Python [conda env:base] *

```
[2]: import pandas as pd

[4]: # Load raw file
df = pd.read_csv("T_ONTIME_REPORTING.csv")

[5]: # Select relevant columns
columns_to_keep = [
    'YEAR', 'MONTH', 'DAY_OF_MONTH', 'DAY_OF_WEEK',
    'ORIGIN_AIRPORT_ID', 'DEST_AIRPORT_ID',
    'CRS_DEP_TIME', 'DEP_TIME', 'DEP_DELAY',
    'CRS_ARR_TIME', 'ARR_TIME', 'ARR_DELAY'
]

df = df[columns_to_keep]

[8]: # Rename to match the model script
df.rename(columns={
    'DAY_OF_MONTH': 'DAY',
    'ORIGIN_AIRPORT_ID': 'ORG_AIRPORT',
    'DEST_AIRPORT_ID': 'DEST_AIRPORT',
    'CRS_DEP_TIME': 'SCHEDULED_DEPARTURE',
    'DEP_TIME': 'DEPARTURE_TIME',
    'DEP_DELAY': 'DEPARTURE_DELAY',
    'CRS_ARR_TIME': 'SCHEDULED_ARRIVAL',
    'ARR_TIME': 'ARRIVAL_TIME',
    'ARR_DELAY': 'ARRIVAL_DELAY'
}, inplace=True)

[10]: # Save to new CSV
df.to_csv("formatted_data.csv", index=False)
print("✅ Formatted data saved to formatted_data.csv")

✅ Formatted data saved to formatted_data.csv

[ ]:
```

One of the biggest problems I had in this part was the massive amount of data downloaded from the Bureau of Transportation Statistics. The file had over 500,000 rows and more than 100 columns, which made it really slow to load and filter in Jupyter. It slowed things down through most of the project, especially when trying to clean or preview the file. I fixed this by only loading the columns I needed and dropping everything else early in the script.

C: DATA FILTERING SCRIPT

I filtered the dataset to include only one airport using the ORG_AIRPORT column. I also dropped missing values and removed any delays over 60 minutes. The only issue I had here was that some columns had NaNs in places I didn't expect. I added a line to drop missing values in just the columns I needed.



D: MLFLOW EXPERIMENT

I used the poly_regressor file and added MLflow logging. I looped through different alpha values for Ridge regression and logged the MSE for each one. I saved the best model and the airport encoding file as artifacts. When I first ran it in a notebook, I got a SystemExit: 2 error because of argparse. I fixed it by removing argparse and hardcoding num_alphas = 20.

```python
#D.4: Train Model & Track with MLflow ---- MLFlow Experiment ----
best_score = float("inf")
best_alpha = None
best_model = None
poly = PolynomialFeatures(degree=order)

mlflow.set_experiment(experiment_name)
with mlflow.start_run():
    for i in range(num_alphas):
        alpha = i * 0.2
        ridge = Ridge(alpha=alpha)
        X_poly = poly.fit_transform(X_train)
        ridge.fit(X_poly, y_train)
        X_poly_val = poly.transform(X_val)
        y_pred = ridge.predict(X_poly_val)
        mse = mean_squared_error(y_val, y_pred)

        mlflow.log_param(f"alpha_{i}", alpha)
        mlflow.log_metric(f"val_mse_{i}", mse)

        if mse < best_score:
            best_score = mse
            best_model = ridge
            best_alpha = alpha

    # After loop ends
    if best_model is None:
        raise ValueError("No valid model was trained.")
```

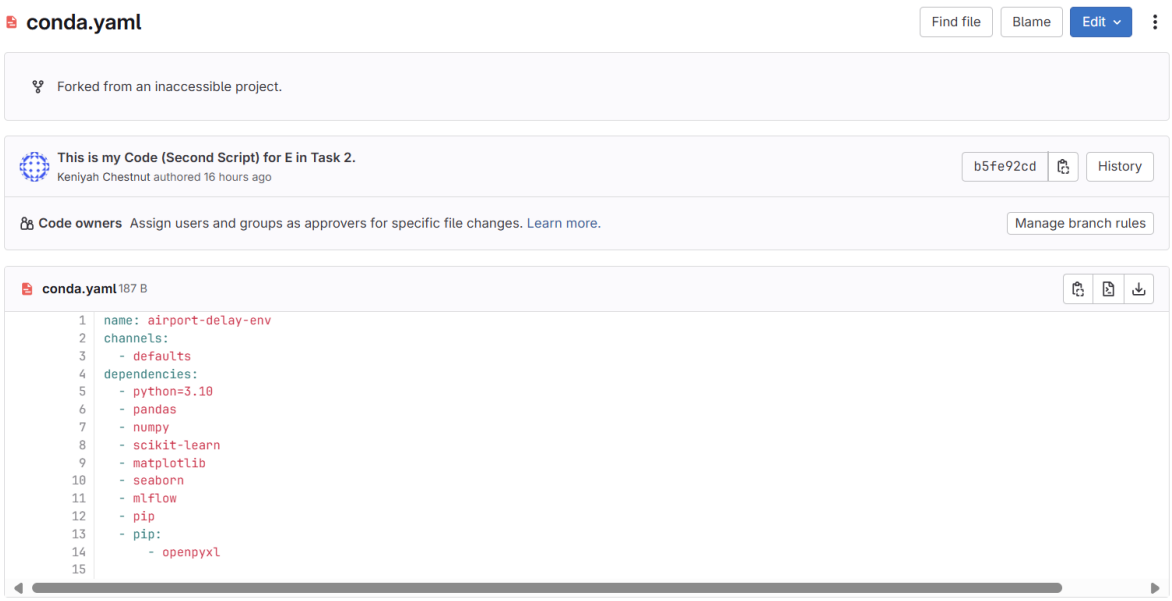
E: MLPROJECT LINKING FILE

I made the MLproject and conda.yaml files so my training script could be run as a pipeline. At first, MLflow wouldn't recognize my main script because I had the wrong filename and indentation in my YAML. After fixing that, the whole pipeline ran with:

jupyter MLproject Last Checkpoint: 15 hours ago

File Edit View Settings Help

```yaml
name: airport-delay-pipeline

conda_env: conda.yaml

entry_points:
  main:
    parameters:
      num_alphas: {type: int, default: 20}
    command: >
      python Task_2_MLflow.py {num_alphas}
```

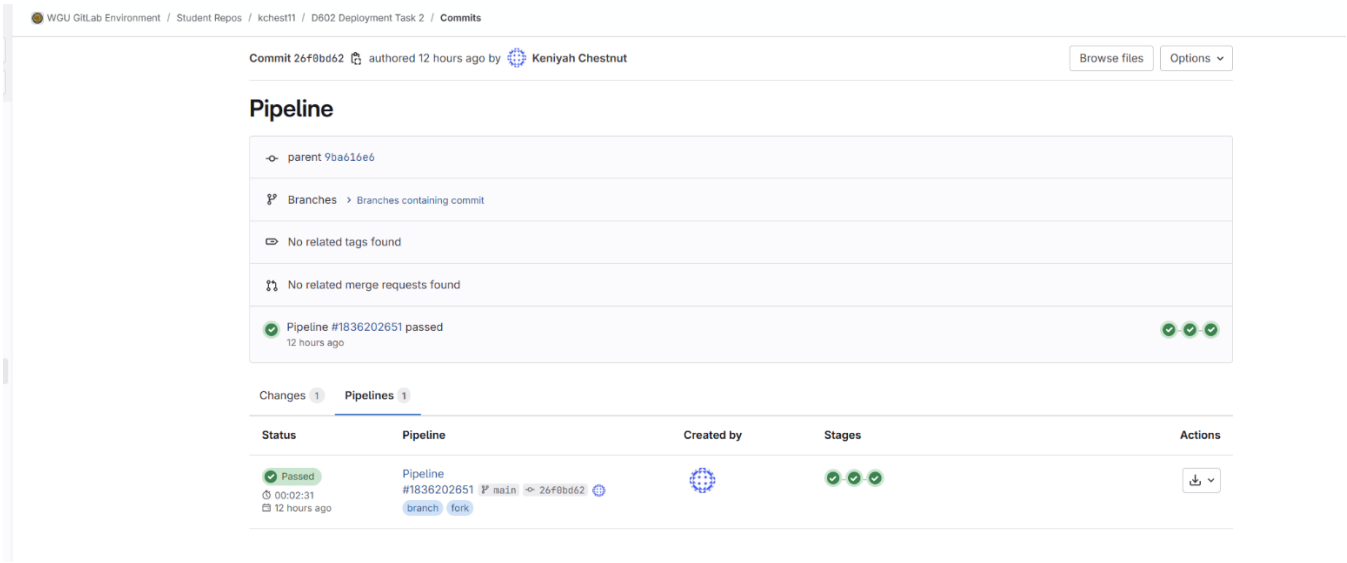


F: EXPLANATION

For the final step, I connected everything with MLflow. I used a .py script instead of a notebook to avoid argparse errors. I removed argparse and used a fixed value during testing. I also had to troubleshoot why the conda.yaml wasn't installing everything. I adjusted the pip section to include openpyxl so my data exports would work.

The hardest parts were getting the MLproject and conda.yaml files to work together and figuring out the right file paths. Once it all worked, MLflow logged the alpha values, validation MSE, and saved the final model and encoding file. I also tracked the dataset with DVC so everything is reproducible.

Here is the working pipeline:



Here is the script:

peline.gitlab-ci.yml 263 B

```yaml
stages:
  - train

train_model:
  stage: train
  image: continuumio/miniconda3
  before_script:
    - conda install -y python=3.10 pandas numpy scikit-learn matplotlib pip
    - pip install mlflow openpyxl
  script:
    - mlflow run . -P num_alphas=20
```

References

All content, code, and configuration files used in this submission were developed based on materials, instructions, and datasets provided by Western Governors University (WGU) as part of the D602 performance assessment. No outside sources were used.