

## D602 Task 2

By Eric Williams

Below I will provide screenshots, explanations, and problems I had along the way for each phase of this project.

### **B:IMPORT AND FORMAT SCRIPT**

For the first task of downloading the necessary information, it took some time to figure out precisely which columns would be needed for the project. Then I was able to find this template in the code which explains which columns would be needed:

*Outputs:*

\* log file named "polynomial\_regression.txt" containing information about the model training process

\* MLFlow experiment named with current date containing model training runs, one for each value of the Ridge regression penalty

```
| YEAR | MONTH | DAY | DAY_OF_WEEK | ORG_AIRPORT | DEST_AIRPORT | SCHEDULED_DEPARTURE | DEPARTURE_TIME | DEPARTURE_DELAY | SCHEDULED_ARRIVAL |  
|:-----|:-----|:-----|:-----|:-----|:-----|:-----|:-----|:-----|:-----|  
| integer | integer | integer | integer | string | string | integer | integer | integer | integer |
```

The next problem I faced was the massive amount of data downloaded from the Bureau website. This became a problem to deal with throughout the project--there were over half a million columns of data and over 100 columns, so the load times were long.

	Year	Quarter	Month	DayofMonth	DayOfWeek	FlightDate	Reporting_Airline	DOT_ID_Reporting_Airline	IATA_CODE_Reporting_Airline	Tail_Number	...	Div4Tai
0	2024	1	1	8	1	2024-01-08	9E	20363	9E	N485PX	...	
1	2024	1	1	9	2	2024-01-09	9E	20363	9E	N912XJ	...	
2	2024	1	1	10	3	2024-01-10	9E	20363	9E	N918XJ	...	
3	2024	1	1	11	4	2024-01-11	9E	20363	9E	N490PX	...	
4	2024	1	1	12	5	2024-01-12	9E	20363	9E	N915XJ	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
547266	2024	1	1	2	2	2024-01-02	UA	19977	UA	N411UA	...	
547267	2024	1	1	2	2	2024-01-02	UA	19977	UA	N426UA	...	
547268	2024	1	1	2	2	2024-01-02	UA	19977	UA	N77520	...	
547269	2024	1	1	2	2	2024-01-02	UA	19977	UA	N446UA	...	
547270	2024	1	1	2	2	2024-01-02	UA	19977	UA	N849UA	...	

547271 rows × 110 columns

Next I identified the columns I needed to keep. The column names weren't exactly the same and needed to be adjusted:

```
# Specify the columns to keep
columns_to_keep = [
    'Year', 'Month', 'DayofMonth', 'DayOfWeek',
    'OriginAirportID', 'DestAirportID', 'DestAirportSeqID',
    'CRSDepTime', 'DepTime', 'DepDelay', 'DepDelayMinutes',
    'CRSArrTime', 'ArrTime', 'ArrDelay', 'ArrDelayMinutes'
]
















# Filter the DataFrame to keep only the specified columns
df_filtered = df[columns_to_keep]
df_filtered
```

```
#Selecting the relevant columns and create a copy
df_filtered = df_filtered[['Year', 'Month', 'DayofMonth', 'DayOfWeek',
    'OriginAirportID', 'DestAirportID',
    'DestAirportSeqID', 'CRSDepTime',
    'DepTime', 'DepDelay', 'DepDelayMinutes',
    'CRSArrTime', 'ArrTime', 'ArrDelay', 'ArrDelayMinutes']].copy()

#Renaming the columns
df_filtered.rename(columns={
    'Year': 'YEAR',
    'Month': 'MONTH',
    'DayofMonth': 'DAY',
    'DayOfWeek': 'DAY_OF_WEEK',
    'OriginAirportID': 'ORG_AIRPORT',
    'DestAirportID': 'DEST_AIRPORT',
    'DestAirportSeqID': 'DEST_AIRPORT_SEQ_ID',
    'CRSDepTime': 'SCHEDULED_DEPARTURE',
    'DepTime': 'DEPARTURE_TIME',
    'DepDelay': 'DEPARTURE_DELAY',
    'CRSArrTime': 'SCHEDULED_ARRIVAL',
    'ArrTime': 'ARRIVAL_TIME',
    'ArrDelay': 'ARRIVAL_DELAY'
}, inplace=True)
```

I uploaded the cleaning program into GitLab with multiple updates to demonstrate the progression of my code:

Nov 01, 2024

	Replace D602_Task_2_cleaning_A.ipynb ... Eric Williams authored 1 day ago	f1fbef68		
	Replace D602_Task_2_cleaning_A.ipynb ... Eric Williams authored 1 day ago	8e5f2d72		
	Replace D602_Task_2_cleaning_A.ipynb ... Eric Williams authored 1 day ago	238c8a4b		
	Replace D602_Task_2_cleaning_A.ipynb ... Eric Williams authored 1 day ago	87e0e825		
	Initial File to clean airport data Eric Williams authored 1 day ago	2236c884		

I also ran a DVC command to create a metafile for my dataset and uploaded it to GitLab:

The file has been successfully created.

ewil708-main-patch-...

d602-deployment-task-2 / filtered\_dataset.csv.dvc

F

Forked from an inaccessible project.

Upload New File

...

Eric Williams authored just now

Code owners

Assign users and groups as approvers for specific file changes. [Learn more.](#)

filtered\_dataset.csv.dvc

107 B

Blame

Edit

1

outs:

2

- md5: 01f48581a6028abfd62f3f0f4942f69a

3

size: 573270

4

hash: md5

5

path: filtered\_dataset.csv

6

C:DATA FILTERING SCRIPT

Next I filtered by the JFK airport. Luckily the reduced size of this data was much easier to deal with:

#Filtering rows where DEST\_AIRPORT\_ID is 12478  
df\_filtered = df\_filtered[df\_filtered['DEST\_AIRPORT'] == 12478].copy()  
  
df\_filtered

	YEAR	MONTH	DAY	DAY_OF_WEEK	ORG_AIRPORT	DEST_AIRPORT	DEST_AIRPORT_SEQ_ID	SCHEDULED_DEPARTURE	DEPARTURE_TIME	DEPARTURE_DELAY	DepT
72	2024	1	6	6	11433	12478	1247805	2115	2107.0	-8.0	
120	2024	1	8	1	12451	12478	1247805	1228	1227.0	-1.0	
121	2024	1	9	2	12451	12478	1247805	1228	1235.0	7.0	
122	2024	1	10	3	12451	12478	1247805	1228	1235.0	7.0	
123	2024	1	11	4	12451	12478	1247805	1228	1233.0	5.0	
...	...	...	...	...	...	...	...	...	...	...	...
487506	2024	1	7	7	10693	12478	1247805	1745	1840.0	55.0	
487683	2024	1	8	1	13930	12478	1247805	1845	1853.0	8.0	
487726	2024	1	7	7	13930	12478	1247805	1403	1520.0	77.0	
487902	2024	1	11	4	13930	12478	1247805	710	722.0	12.0	
488167	2024	1	13	6	13930	12478	1247805	710	722.0	12.0	

9467 rows × 15 columns

I then performed two additional cleaning processes: deleting missing data (likely for cancelled flights), as well as stripping empty leading and trailing spaces:

```
# Check for missing values by column
missing_values = df_filtered.isnull().sum()

# Display the number of missing values for each column
print(missing_values)
```

```
YEAR                0
MONTH               0
DAY                 0
DAY_OF_WEEK         0
ORG_AIRPORT         0
DEST_AIRPORT        0
DEST_AIRPORT_SEQ_ID 0
SCHEDULED_DEPARTURE 0
DEPARTURE_TIME      277
DEPARTURE_DELAY      277
DepDelayMinutes     277
SCHEDULED_ARRIVAL    0
ARRIVAL_TIME        292
ARRIVAL_DELAY       318
ArrDelayMinutes     318
dtype: int64
```

```
# Drop rows with missing values for the specified columns
df_filtered.dropna(subset=['DEPARTURE_TIME', 'DEPARTURE_DELAY', 'ARRIVAL_TIME', 'ARRIVAL_DELAY'], inplace=True)
```

```
# Check for missing values by column
missing_values = df_filtered.isnull().sum()

# Display the number of missing values for each column
print(missing_values)
```

```
YEAR                0
MONTH               0
DAY                 0
DAY_OF_WEEK         0
ORG_AIRPORT         0
DEST_AIRPORT        0
DEST_AIRPORT_SEQ_ID 0
SCHEDULED_DEPARTURE 0
DEPARTURE_TIME      0
DEPARTURE_DELAY      0
DepDelayMinutes     0
SCHEDULED_ARRIVAL    0
ARRIVAL_TIME        0
ARRIVAL_DELAY       0
ArrDelayMinutes     0
dtype: int64
```

```
#Dropping spaces before and after each cell
df_filtered = df_filtered.applymap(lambda x: x.strip() if isinstance(x, str) else x)

df_filtered
```



I then exported and saved the cleaned dataset

	YEAR	MONTH	DAY	DAY_OF_WEEK	ORG_AIRPORT	DEST_AIRPORT	DEST_AIRPORT_SEQ_ID	SCHEDULED_DEPARTURE	DEPARTURE_TIME	DEPARTURE_DELAY	D
72	2024	1	6	6	11433	12478	1247805	2115	2107.0	-8.0	
120	2024	1	8	1	12451	12478	1247805	1228	1227.0	-1.0	
121	2024	1	9	2	12451	12478	1247805	1228	1235.0	7.0	
122	2024	1	10	3	12451	12478	1247805	1228	1235.0	7.0	
123	2024	1	11	4	12451	12478	1247805	1228	1233.0	5.0	
...	...	...	...	...	...	...	...	...	...	...	...
487506	2024	1	7	7	10693	12478	1247805	1745	1840.0	55.0	
487683	2024	1	8	1	13930	12478	1247805	1845	1853.0	8.0	
487726	2024	1	7	7	13930	12478	1247805	1403	1520.0	77.0	
487902	2024	1	11	4	13930	12478	1247805	710	722.0	12.0	
488167	2024	1	13	6	13930	12478	1247805	710	722.0	12.0	



9149 rows × 15 columns

```
#Saving filtered dataset
df_filtered.to_excel(r'C:\Users\18014\Desktop\filtered_dataset.xlsx', index=False)
```

I also uploaded two separate updates for this coding file into GitLab to show two more progressions of my code:

 Replace D602\_Task\_2a.ipynb   
Eric Williams authored 5 hours ago

9ff31302  

 Upload New File   
Eric Williams authored 5 hours ago

9b2fe19f  

## D:MLFLOW EXPERIMENT

Going through another person's code to try and fix their problems and make it functional was difficult. There were a couple of small changes that had to be made. First, I brought all the code into Jupyter notebook and broke it into chunks to see what was functional and what needed to be changed. Here I had to uncomment a few lines and change the filename to import the cleaned data:

```
# set up the argument parser
parser = argparse.ArgumentParser(description='Parse the parameters for the polynomial regression')
parser.add_argument('num_alphas', metavar='N', type=int, help='Number of Lasso penalty increments')
order = 1
num_alpha_increments = 20

# Uncomment the two lines above and comment the line below to run this script from the command prompt or as part of an
# MLFlow pipeline
num_alphas = 20

# configure logger
logname = "polynomial_regression.txt"
logging.basicConfig(filename=logname,
                    filemode='w',
                    format='%(asctime)s %(levelname)s %(message)s',
                    datefmt='%H:%M:%S',
                    level=logging.DEBUG)
logging.getLogger('matplotlib.font_manager').disabled = True
logging.info("Flight Departure Delays Polynomial Regression Model Log")

# read the data file
df = pd.read_csv(r"cleaned_data.csv")
tab_info=pd.DataFrame(df.dtypes).T.rename(index={0:'column type'})
```

Because I had already changed the column names to match with the code in the template document, much of the code was functional. Much of the code in the middle remains untouched.

Here is my code finishing the to-do list, running the test, calculating the MSE and average delay time, and the plots I generated:

```
# TO DO: create an MLflow run within the current experiment that logs the following as artifacts, parameters,
# or metrics, as appropriate, within the experiment:
# 1. The informational log files generated from the import_data and clean_data scripts
# 2. the input parameters (alpha and order) to the final regression against the test data
# 3. the performance plot
# 4. the model performance metrics (mean squared error and the average delay in minutes)

mlflow.start_run(experiment_id = experiment.experiment_id, run_name = "Final Model - Test Data")
# YOUR CODE GOES HERE

# 1. The informational log files generated from the import_data and clean_data scripts
mlflow.log_artifact("polynomial_regression.txt")

# 2. The input parameters (alpha and order) to the final regression against the test data
alpha = 20
order = 1
mlflow.log_param("alpha", alpha)
mlflow.log_param("order", order)

X_test_poly = poly.transform(X_test)
predictions = ridgereg.predict(X_test_poly)
mse = mean_squared_error(Y_test, predictions)
average_delay = predictions.mean()

# 3. The performance plot
plt.figure()
plt.plot(Y_test, label='Actual', alpha=0.7)
plt.plot(predictions, label='Predicted', alpha=0.7)
plt.title('Model Performance')
plt.xlabel('Samples')
plt.ylabel('Delay')
plt.legend()
plt.savefig("performance_plot.png")
mlflow.log_artifact("performance_plot.png")

mlflow.end_run()

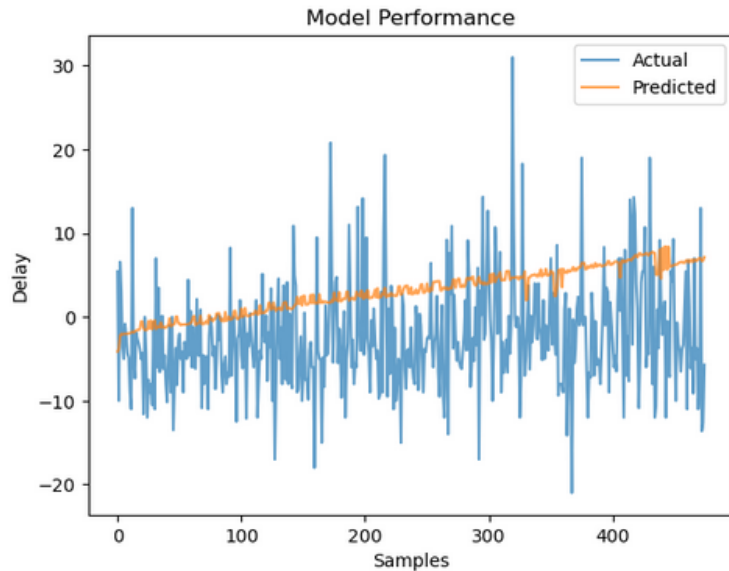
logging.shutdown()

# 4. The model performance metrics (mean squared error and the average delay in minutes)

# Log metrics
mlflow.log_metric("mean_squared_error", mse)
mlflow.log_metric("average_delay_minutes", average_delay)

print(mse)
print(average_delay)

72.20985906801079
3.030863234272947
```



### **E:MLPROJECT LINKING FILE**

Creating the pipeline was perhaps the most difficult part of the process for me due to the difficulty of finding the necessary code and the persistent errors I had to update. I ran about 20 attempts before the pipeline ran successfully. I found a few problems with my earlier work that needed to be changed. For example, the YAML file was having trouble with my Jupyter files, so I converted them to Python files and reuploaded them for the pipeline. The pipeline also couldn't find the source for the data until I uploaded the datasets to GitLab and changed the sourcing to be from a local path rather than from my desktop. Then I started to receive errors that the pipeline couldn't read the python files and the different packages, so I had to specify the packages and language that was being used. Finally after troubleshooting all the errors, the pipeline was successful in merging the above python files.

## Update .gitlab-ci.yml file

parent [73ed3cf4](#)


Branches [ewil708-main-patch-90813](#)

No related tags found

No related merge requests found

Pipeline [#1524734864](#) passed with stage in 1 minute and 32 seconds

Changes [1](#) Pipelines [1](#)

Status	Pipeline	Created by	Stages
<div>Passed</div> <div>00:01:32</div> <div>3 hours ago</div>	<div>Update .gitlab-ci.yml file</div> <div>#1524734864 <a href="#">ewil708-main-patch-90813</a></div> <div><a href="#">bf9dc76a</a></div> <div>latest fork</div>		<div>✓</div>

Here is the script below:

```

1 image: python:3.9
2
3 stages:
4   - run
5
6 run_scripts:
7   stage: run
8   script:
9     - pip install pandas numpy seaborn matplotlib scikit-learn mlflow
10    - echo "Running analysis script..."
11    - python D602_Task_2_final.py
12    - echo "Running data cleaning script..."
13    - python D602_Task_2_cleaning_final.py
14
15
16

```

The script essentially says to run python, to install the needed packages, to run the script of the data cleaning process, and to run the script for the MLFlow and Polyregression project.



## Sources

No sources were used besides official WGU course materials.