D602 Deployment Write Up

By: Kevin Sandoval

Part A:

For Part A of the performance assessment, I followed the steps shown in the video titled “Create your GitLab course specific branch OLD 1” found in the D602 Course Resources page. I then created a branch for each time I wanted to submit the code I was working on. For parts B and C I erroneously labeled a branch as a final version, but then had to make additional changes. Because of this, the dates for the branch creation is important to note as I created branches after I named one with “final” in the name. Apologies for any inconvenience that may cause and I have learned to adjust the naming convention of my branches and files in the future. I also followed the link to the Bureau of Transportation Statistics and downloaded all data from California in the month of June, 2024.

Part B:

For Part B of the performance assessment, I used code that I created in D206 for importing data from a CSV file. I then edited the names of columns and checked to make sure that all the column names matched what was shown in the poly\_regressor\_Python\_1.0.0.ipynb file that I accessed from GitLab. After ensuring correctness I exported the edited dataset under the name “imported\_data.” I then created a logger that would created a text file showing a confirmation that the exporting the file to a CSV file was successful. I found the template for the logger also in the poly\_regressor\_Python\_1.0.0.ipynb file. I did not encounter many challenges in this portion of the PA.

Part C:

For Part C of the performance assessment, I imported the dataset I created called “imported\_data” and named it “cleaned\_data” to match naming conventions. As part of the data cleaning process I checked the dataset for duplicate values and missing values. I also added loggers at the start and throughout my code that wrote to a text file to see that everything was running smoothly.

I found that there were no duplicates but there were missing values in the columns “DEPARTURE\_TIME”, “DEPARTURE\_DELAY”, “ARRIVAL\_TIME”, and “ARRIVAL\_DELAY”. I chose to drop the rows that had missing values in “DEPARTURE\_TIME” and “ARRIVAL\_TIME” as those values can not be created or guessed. I was left with some missing values in “ARRIVAL\_DELAY” however and so I chose to create the values from the information that I did have. Before I started, I changed the other columns that had missing values to type int so I could work on the columns more easily. Since the values I did have were in int form, creating the missing values was not as simple as subtracting the time as that would have given erroneous numbers. For example, all values in “DEPARTURE\_TIME” ranged from 0 to 2400 but the 10s place was never more than 6 as the integer is representing time. So 11:37AM would be shown as 1137and 5:15PM would be shown as 1715. Because of this I needed to change these missing values to datetime format and then try and subtract. I decided to create a new dataframe that contained the missing values. After these edits I realized that the function I was using was delta, which did calculate the difference between the scheduled time and the actual arrival time, but in some cases it created a value that indicated that the flight was actually 10 hours early, when in fact it was 14 hours late. To combat this I added 1440 to the negative values as 1440 is the total amount of minutes in a day. The calculation was in minutes to match the convention to the other non missing values. After I got all missing values I then used the .loc function to add in the created missing values to the correct position based on their index. I added a logger to in these steps to ensure that everything was running correctly.

As a final data cleaning step I checked the departure delay and arrival delay for any outliers. I decided to exclude some of the outliers found in the departure delay but chose to keep those in the arrival delay.

Finally, I edited the “ARRIVAL\_DELAY” column to be of type int to match what was shown in the poly regressor file. I then exported the dataframe to a CSV file named “cleaned\_data” and added a logger to state that the export was successful.

Part D:

For Part D, the first thing I did was edit the path for calling in the dataset. I then ran the workbook to ensure that everything would run correctly. Afterwards I saw the steps at the bottom of the file and implement another MLFlow experiment to capture the features mentioned. I used the template provided in one of the cells above that created an MLFlow experiment. I changed the variables and to match the new testing dataframe rather than the training. I also created the log file to see that everything was running correctly. One problem I did run into was in the argument parser. There was a comment added in that mentioned uncommenting the new lines above and commenting the line below to run the script with an MLFlow pipeline. I could not get the code to work properly and kept receiving the error: num\_alpha\_increments = args[0]

~~~~^^^

TypeError: 'Namespace' object is not subscriptable

In order for the model to run I had to keep those lines of code commented. I did not change the code from the download at all.

Part E:

Based on the example found on the MLFlow website, I was able to create an MLProject file and add in the necessary dependencies. I checked with the YAML file and the code I have been using to ensure that everything match up properly.

Part F:

I tried to run “mlflow run <https://gitlab.com/wgu-gitlab-environment/student-repos/ksando10/d602-deployment-task-2.git>” which is the path to my forked repository on GitLab. However I do not have access to edit the main branch, and the mlflow command checks the main branch so I was unable to perform the run in this way. When I called each .py function manually to where they are saved in my files however, the project runs perfectly as seen in the screenshot below. The warnings are for potential deprecation of code that was included in the poly regressor file, not code that I wrote myself. I wrote the command using the && separator as recommended by Dr. Sewell in his webinar for Task 2.

I created an actual final branch called “Final\_Branch” in my forked repository. That branch contains all the necessary files for the pipeline including the coding files, the text files that get created, the DVC image file, the datasets used, the model performance graph, the MLProject file, and the YAML files.

A screenshot of a computer

Description automatically generated

G.

Deza, A (September 16th, 2022). *MLOps Essentials: Model Development and Integration.* Retrieved January 8th, 2024,From <https://www.linkedin.com/learning/mlops-tools-mlflow-and-hugging-face>

Dr. Sewell, W (n.d). *D602 Webinar # 3.* Retrieved December 27th, 2024,From D602 Course Search

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=4bdc60ba-a36a-4e4e-8172-b21900d735f5>

DVCorg (Sep. 30th, 2020), *Versioning Data with DVC (Hands-on Tutorial).* Retrieved January 10th, 2024, From <https://www.youtube.com/watch?v=kLKBcPonMYw&t=83s>

Mlflow (n.d.) *mlfow* Retrieved January 16th, 2024 From <https://mlflow.org/docs/latest/python_api/mlflow.html#mlflow.start_run>

Sherin, A (n.d). *How to Run MlFlow Experiment* Retrieved January 8th, 2024,From <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=6025520f-a6cc-4fcd-9234-b25300068d11>

WGU. (n.d). *Create your GitLab course specific branch OLD 1.* Retrieved December 29th, 2024 From <https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=b28d1026-3889-48de-881f-b22100338f5c>