UDACITY

Return to Classroom

DISCUSS ON STUDENT HUB

A Dynamic Risk Assessment System

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Udacity Student

You have done an amazing job in this project and it will open a huge door in your future work. MLOps is a very important topic on every field of machine learning. If you want models into production and reliable pipelines, you are on the right path!

As a reference, I would like to share some articles and resources to boost your learning:

MLOps: Continuous delivery and automation pipelines in machine learning

MLOps: What It Is, Why it Matters, and How To Implement It (from a Data Scientist Perspective)

MLOps Core

10 Amazing MLOps Learning Resources

MLOps-Reducing the technical debt of Machine Learning

I really hope you enjoy the project as we mentors did and we are looking forward your next step!

Data Ingestion

- The ingestion.py script should perform this step.
- Every file contained in the data folder needs to be read into Python.
- All files should be compiled into a pandas data frame and written to a csv file called "finaldata.csv". De-dupe the compiled data frame before saving.
- Store the ingestion record in a file called "ingestedfiles.txt".

Data Ingestion

Great job! 😄

You have loaded the the paths using the config file using the *ingestion.py* script and performed the ingestion step.

Training, Scoring, and Deployment

Students should write training.py to accomplish this. The model should be saved in the pickle format.

Training, Scoring, and Deployment

training.py

Perfect! 😄

Your *training.py* script trains the model and save in the pickle format.

Students should write a scoring script in the scoring.py starter file. Scoring should be performed using the F1 score.

Training, Scoring, and Deployment

scoring.py

You have written the *scoring.py* script using the F1 score.

You can read more about the F1 Score and other metrics.

The scoring.py script should write the F1 score to a .txt file called latestscore.txt.

Training, Scoring, and Deployment

scoring.py > latestscore.txt

The *scoring.py* script write the F1 score to the *latestscore.txt*.

The deployment.py script should copy the trained model, the F1 score, and the ingested file record to a production deployment directory.

Training, Scoring, and Deployment

deployment.py

Great! 😉

You have copied the files that were created in the previous steps.

Diagnostics

- The diagnostics.py script should perform this step.
- Timing should be checked for both data ingestion and training in seconds.
- Summary statistics (means, medians, and modes) should be checked for each numeric column.
- Students will create a function for making predictions based on the deployed model and a dataset.

Diagnostic

diagnostic.py

Your *diagnostic.py* script is perfect! 😉

[✔] - Timing for data ingestion and training
[✔] - Summary Statistics
[✔] - Prediction

- The diagnostics.py script should perform this step.
- Data integrity should be checked by measuring the percentage of NA values in each of the numeric dataset's columns.

Diagnostic

diagnostic.py - Data Integrity

Great! 😄

You have checked the data integrity in your *diagnostic.py* script.

- The diagnostics.py script should perform this step.
- All modules in requirements.txt need to have their latest versions and currently installed versions checked.

Diagnostic

diagnostic.py - Dependencies

Awesome! 😉

You have implemented all necessary functions.

Reporting

The app.py script will perform this section.

- An endpoint for scoring needs to provide model scores based on test datasets and models (found in /testdata/).
- An endpoint for summary statistics needs to provide summary statistics for the ingested data (found in the directory specified by the output_folder_path in config.json)
- An endpoint for diagnostics needs to provide diagnostics for the ingested data (found in the directory specified by the output_folder_path in config.json). The diagnostics should include timing, dependency checks, and missing data checks.
- An endpoint for model predictions needs to return predictions from the deployed model (found in the directory specified in the prod_deployment path in config.json) for an input dataset (passed to the endpoint as an input)

Students will create a function in reporting.py that generates a confusion matrix that shows the accuracy of the model on test data (found in /testdata/).

Reporting

app.py

Great job! 😄

Your application is running without issues and returning HTTP 200 status

In apicalls.py, call API's to get the model predictions, accuracy score, summary statistics, and diagnostics that are returned by the API endpoints. The apicalls.py script needs to combine these API outputs and write the combined outputs to the workspace, to a file called apireturns.txt.

Reporting

app.py - Prediction

Your *app.py* is able to predict using the saved model.

Process Automation

- The fullprocess.py script will perform this section.
- Check for the presence of non-ingested data in the /sourcedata/ folder
- Check for whether the most recent model performs better than the previously deployed model

Process Automation

Process Automation - fullprocess.py

You have created the ML pipeline in order to achieve a better F1 score. Great job in this part of your project.

😄

- In deployment.py, copy the model from its initial location to the final production deployment directory. The initial location is specified in the output_folder_path of config.json. The production deployment directory is specified in the prod_deployment_path of config.json.

- Students need to set up a cron job that regularly runs fullprocess.py.
- The fullprocess.py script should call the deployment.py script only under certain conditions: when there is new data ingested, AND when there is model drift.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START