Module 4: Mini Project-1 Part-B

Bike Rental Prediction

Continuous Delivery

For this project, we will build a GitHub Actions workflow to automate model training, testing, package building, api dockerizing, and docker image pushing steps for the bike rental count prediction system. Please refer to Module 4 - AST 2 for this mini-project.

Part B [Mini-project Session - 8 July 2023]

Step 1: Ensure to go through the previous mini-project [Bike Rental Prediction - Continuous Integration]

Step 2: Download and understand project folder in your local system: (1 points)

2.1 Download the given project folder '*bikeshare_project*' on to your system
2.2 Open it in VS Code, and understand the project structure and code organization

Step 3: Perform training, testing, and package building steps on your system: (1 point)

3.1 Open the terminal in VS Code, and create a virtual environment
3.2 Activate the virtual environment, and install the necessary dependencies (for training and testing both)
3.3 Execute the "*train_pipeline.py*" script to train the image classification model on bike rental dataset
3.4 Run the "*predict.py*" script to generate predictions
3.5 Run the test cases by executing the "*pytest*" command in the terminal (debug if issue persists)
3.6 Run the "*build*" command to create distributable files (.tar, .whl, etc)

Step 4: Run FastAPI application on your system: (1 point)

4.1 Copy the wheel file (.whl) generated in previous step and paste it inside "*bikeshare_model_api*" directory
4.2 Move into the api folder "*bikeshare_model_api*" and Install fastapi dependencies using "*requirements.txt*" (.whl file will be used to install the functionalities of the model into the FastAPI project)

4.3 From inside the api folder, execute the "*app/main.py*" script to deploy the model and get server url
4.4 Access the application and test its prediction, then stop the application

Step 5: Dockerize the FastAPI application on your system: (2 points)

5.1 Create a Dockerfile to dockerize "*bikeshare_model_api*" application
5.2 Exit from virtual environment, and start Docker
5.3 Using the Dockerfile create a docker image
5.4 Using the docker image start a new container, and check if the application is running
5.5 On successful run, push the docker image to DockerHub

Step 6: On your GitHub account, create a new repository, & add DockerHub credentials within its secrets: (1 point)

6.1 Create a new repository to store files related to this mini-project
6.2 Get the access token from your DockerHub account settings
6.3 Add the docker username and access token to the secrets of this repository

Step 7: Clone the remote GitHub repository in your system & add project files:

7.1 Clone the remote repository in your system
7.2 Add the downloaded project folder "*bikeshare_project*" along with your Dockerfile to this cloned repository
[Note: Do not include the below files to the repo:

- cache files (*__pychache__*, etc),
- distributable files (*.whl*, etc), and
- trained model (*.pkl* file)

These need to be generated during the GitHub actions workflow.**]**

7.3 Finally, push the changes into the remote GitHub repository

Step 8: Create a GitHub Actions workflow to automate the steps for model training, testing, package building, api dockerizing, and pushing the docker image: (4 points)

8.1 Create a GitHub Actions workflow to automate the steps for model training, testing, package building, api dockerizing, and docker image pushing
8.2 Add them as different jobs in the workflow
8.3 Add the jobs to run sequentially (debug if issue persists)
8.4 Access the running workflow pipeline

8.5 On successful run, access the pushed docker image on your DockerHub account