HCL MaxAI Workbench Machine Learning as a Code User Guide

Contents

Chapter 1. Overview

MLaC (Machine Learning as a Code) is a python code corresponding to each ML pipeline stage

User can either directly use that code for model training or can create a docker container and deploy/execute using docker compose, kubernet or kubeflow. This components code is generated using the MaxAI Workbench. Users will have to train the model in MaxAI Workbench and based upon the configuration, the code will be generated which can further be used to retrain the model in production environment. Based upon the configurations (Basic and Advance), automatic code will be generated to provide the facility for running the code as a pipeline, on the local machine or on the cloud.

Pipeline Components

- **DataIngestion**: Read the data from specified location and save it for the later use.
- **DataTransformation**: Data cleaning and transformation is performed.
- **FeatureEngineering**: Feature reduction or feature selection techniques applied.
- **ModelTraining**: Train models based on the algorithms selected and Feature Engineering options and create separate folder for each of the model to run them in the parallel.
- **Prediction**: Best model is selected based on the score and perform prediction on the provided data.

Supported Platforms

- Red Hat Linux

Chapter 2. Code Generation

1. Click the required use case **Publish** button.
2. Select the MLaC option from packages in the available options.

   Note: Code Generation is currently available for 'Classification', 'Regression' and ' Timeseries' Problem Type only (only ML algorithms are supported for Classification and Regression and MLP and LSTM is supported for Timeseries).

3. Click the **Generate Code** button for generating code.
4. Success message will be generated after code generation process finished. The Path of the generated code will be displayed.
   Example: `\HCLTech\AION\data\target\AI0034\1\publish\MLaC`
5. Each component has the following files:
   - Docker file: Required for creating the Docker image.
   - `Requirements.txt`: Required for installing the python packages in Docker.
   - Script: This script will be executed as a part of Machine Learning pipeline.
   - Configuration: Configuration required to execute script.

Chapter 3. Generate Container (Local)

Generate Container (Local)

1. User must create an image for each of the components present in the MLaC folder (every component has a docker file present which will help to build the Docker image).
2. Once all the images have been created user can run each container (using the images) one by one in the following sequence: **Model Monitoring > Data Ingestion > Data Transformation > feature engineering > Model Training > model Registry > modelServing**.
3. You will see the status of each step can be seen in the shell.

Detailed Steps to Create a Container

1. Train the model.
2. Go to the MLac generate code and click **Generate the code**.
3. Copy the path.
4. Open MaxAI shell and go the path: `\HCLT\AION\target\AION_375_1\publish\`
5. Run the command python MLaC\run_pipeline.py.
6. Basically we are creating persistent volume and intermediate files will be stored in the file (raw.data, traindata, etc).
7. : `Model Monitoring: {“status”:” success”,” DataFilepath:” pipeline executing first time”}`
8. **Example**: `Data Ingestion: {: {“status”:” success”,” DataFilepath”: rawData.dat”,” Records”:130}`
9. In data ingestion it will read data from UI and created a new file.
10. Similarly, **DataTransformation > FeatureEngineering > ModelTraining > XGBClassifier_MLBased > ModelRegistry > ModelServing**.
11. Open putty goes to the location path.
12. Check the docker images are present and delete it one by one using the command Docker rmi -f (forcefully) and image ID.
13. Go the Model Monitoring path.
14. Run the command Docker build -t modelmonitoring: y1.
15. It will copy and all the Docker file images, Docker build -t model monitoring:y1(you can change as per convenience) . (docker build -t modelmonitoring:y1 .)
16. Another example after ModelMonitoring, `cd DataIngestion <docker build -t dataingestion:y1>`.
17. Similarly, **DataTransformation > FeatureEngineering > ModelTraining > ModelRegistry > ModelServing**.
18. Check all images or build or not by using Docker images as command.
19. Build the Docker image using this command so that we can use the container.
20. To execute the each component one by one, please issue run docker command.

Steps to Run the Container

After checking Docker images are present:

a. Run command `Docker run -it --rm -v /home/ubuntu/aion:/aion (model montoring Image id) 416398138 python code.py`.
b. It will create a container and it will map the folder.
c. Follow the steps for Remaining (docker images) as **Data Ingestion > Data Transformation > Feature Engineering > Model Training > Model Registry > Model**.