Assignment 2

Topics

- Containers and Virtualization
- Programming Best Practices

Assignment Overview

You have developed a model to classify clouds (in the sky, not AWS) into one of two types based on features generated from cloud images. You are happy with the model's performance and want to move it into production so anyone can classify the clouds they see. This will require moving the notebook you developed into scripts and taking the steps to make it reproducible across environments. The notebook can be found here

- load config from local file
- clean/normalize the data
- train a simple supervised model
- capture some performance metric in a chart
- save the key "artifacts" to the disk
  - metadata/config
  - training data
  - trained model object
  - metrics/charts
- Upload the artifacts directory to S3

All the steps should be composed in a single `pipeline.py` script and make use of functions defined in your python package. E.g. data-cleaning, model-training, and other key steps must be defined in composable functions following the best-practices covered in this unit's lectures (unit-testing, exception handling, logging, coding style, etc.).

Steps

1. Pull out all configurations necessary for developing and evaluating the model in `clouds.ipynb` into a YAML configuration file. The choice of what things need to be configured is part of what will be graded in the assignment. An example is included in the `config` directory, but is incomplete and should be expanded to include appropriate config.

2. Move the code developed in `clouds.ipynb` into modular functions within modules/scripts that will enable the reproducible execution of each step of the model development.
    - Each step should take as input the artifacts created by the previous step (except for the data acquisition step).
    - Each step's artifacts should be saved to an appropriate file.
    - The choice of what functions and Python modules/scripts to create is your own but should follow the lessons learned in the course.
3. Add relevant unit tests for the functions generated for 3.1: Additional features, transformations, and interactions. Write one "happy path" unit test and one "unhappy path" unit test for each feature.
4. Make it so that the entire pipeline can be reproduced using a singe command (after the necessary image has been built). The choice of what files to create for this step is part of what will be graded in the assignment.
5. Include any additional files necessary to create a reproducible model pipeline.

Technical Requirements

- Write a Dockerfile that runs the pipeline and saves the desired artifacts to the local filesystem.
- Include instructions in the README which include any required environment variables and relevant commands for building and running the Docker image
- Write python modules that perform the data cleaning and model training functions presented in the jupyter notebook
- Compose all these steps in a single `pipeline.py` file
- Write unit tests for the functions in the python modules (at least 5 total tests)
- Include instructions in the README on how to run the tests (tests should also run in a Docker container)
- All python code should comply with PEP8 and pass `pylint` linting (`.pylintrc` included in this repository)
- The S3 Bucket used for uploading pipeline artifacts should be set in the config file *or* overridden as an environment variable. IF the bucket cannot be set without modifying your code, you will lose points.