CS 643, Cloud Computing - Programming Assignment 2

Goal: The purpose of this individual assignment is to learn how to develop parallel machine learning (ML) applications in Amazon AWS cloud platform. Specifically, you will learn: (1) how to use Apache Spark to train an ML model in parallel on multiple EC2 instances; (2) how to use Spark's MLlib to develop and use an ML model in the cloud; (3) How to use Docker to create a container for your ML model to simplify model deployment. In this assignment, you are allowed to use ChatGPT (or other AI copilots) for help with programming. If you do use such tools, you should report how you used them and how useful their code and suggestions have been.

Description: You have to build a wine quality prediction ML model in Spark over AWS. The model must be trained in parallel on multiple EC2 instances. Then, you need to save and load the model in an application that will perform wine quality prediction; this application will run on one EC2 instance. The assignment must be implemented in Java, Scala, or Python on Ubuntu Linux. The details of the assignment are presented below:

- Input for model training: we share 2 datasets with you for your ML model. Each row in a dataset is for one specific wine, and it contains some physical parameters of the wine as well as a quality score. Both datasets are available in Canvas, under Programming Assignment 2.
    - TrainingDataset.csv: you will use this dataset to train the model in parallel on multiple EC2 instances.
    - ValidationDataset.csv: you will use this dataset to validate the model and optimize its performance (i.e., select the best values for the model parameters).
- Input for prediction testing: TestDataset.csv. We will use this file, which has a similar structure with the two datasets above, to test the functionality and performance of your prediction application. Your prediction application should take such a file as input from the local host. Specifically, your application should take the path to the test file as a command line parameter. For your testing, you will use the validation file.
- Output: The output of your application will be a measure of the prediction performance, specifically the F1 score, which is available in MLlib.
- Model Implementation: You have to develop a Spark application that uses MLlib to train for wine quality prediction using the training dataset. You will use the validation dataset to check the performance of your trained model and to potentially tune your ML model parameters for best performance. You should start with a simple classification model (e.g., logistic regression) from MLlib, but you can try multiple ML models to see which one leads to better performance. For classification models, you can use 10 classes (the wine scores are from 1 to 10). Note: there will be extra-credit for the top 3 applications/students in terms of prediction performance (see below under grading).

  Docker container: You have to build a Docker container for your prediction application. In this way, the prediction model can be quickly deployed across many different environments. You can learn more about Docker in Module 12 (slides are already posted in Canvas). This is an example of the Docker command, which TA will run on his PC to test your Docker container:

  `>sudo Docker run yourDockerImage testFilePath` (you may use the '-v' parameter of the 'docker run' command to map part of the host file system on the container's file system)

- The model training is done in parallel on 4 EC2 instances. You can choose any AWS services for Spark parallel training, including EMR, Flintrock, etc, and you don't have to build a Docker container for training.
- The prediction without Docker is done on a single EC2 instance. After this is done, you shall build a Docker container, and deploy it on an EC2 instance.

Submission: You will submit in Canvas, under Programming Assignment 2, a text/Word/pdf file that contains:

- A link to your code in GitHub. The code includes the code for parallel model training and the code for the prediction application.
- A link to your container in Docker Hub.

This file must also describe step-by-step how to set-up the cloud environment and run the model training and the application prediction. For the application prediction, you should provide instructions on how to run it with and without Docker. Finally, the file should describe what code was generated by ChatGPT/copilots (if any), what code you wrote from scratch, and what code you adapted from the one generated by ChatGPT/copilots. You should also describe your overall experience with generating code using ChatGPT/copilots (if you used these tools).

Grading:

- Parallel training implementation	– 40 points
- Single machine prediction application	– 20 points
- Docker container for prediction application	– 20 points
- Description of the process of building the application, including experience with ChatGPT/Copilots	– 20 points
- Extra-credit for top 3 prediction performance	– 20 points