CS643 - Assignment 2

Sai Himaja Kinthada (sk3355@njit.edu)

This project involves creating a Python application that utilizes the PySpark interface. The application runs on an Amazon Web Services (AWS) Elastic MapReduce (EMR) cluster. Its main goal is to train a machine learning model in parallel on EC2 instances to predict wine quality using publicly accessible data. After training, the model is used to predict wine quality. Docker is used to produce a container image for the trained machine learning model, simplifying the deployment process.

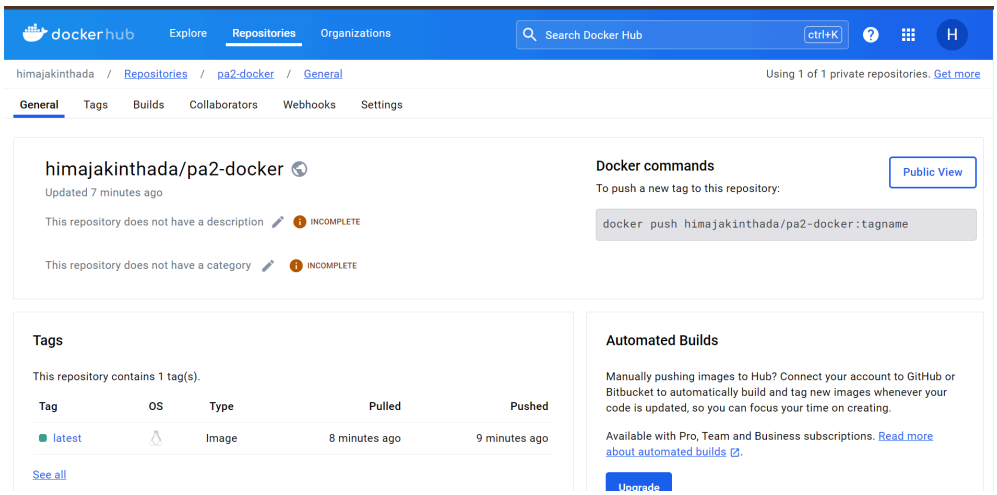
The primary Python source files in this project are:

1. **winequalityprediction.py**: Reads the training dataset from S3 and trains the model in parallel on an EMR Spark cluster. Once trained, the model can be executed on provided test data via S3. The program stores the trained model in the S3 bucket.

2. **winequalitytestdataprediction.py**: Loads the trained model and executes it on a given test data file. This program prints the F1 score as a metric for the accuracy of the trained model.

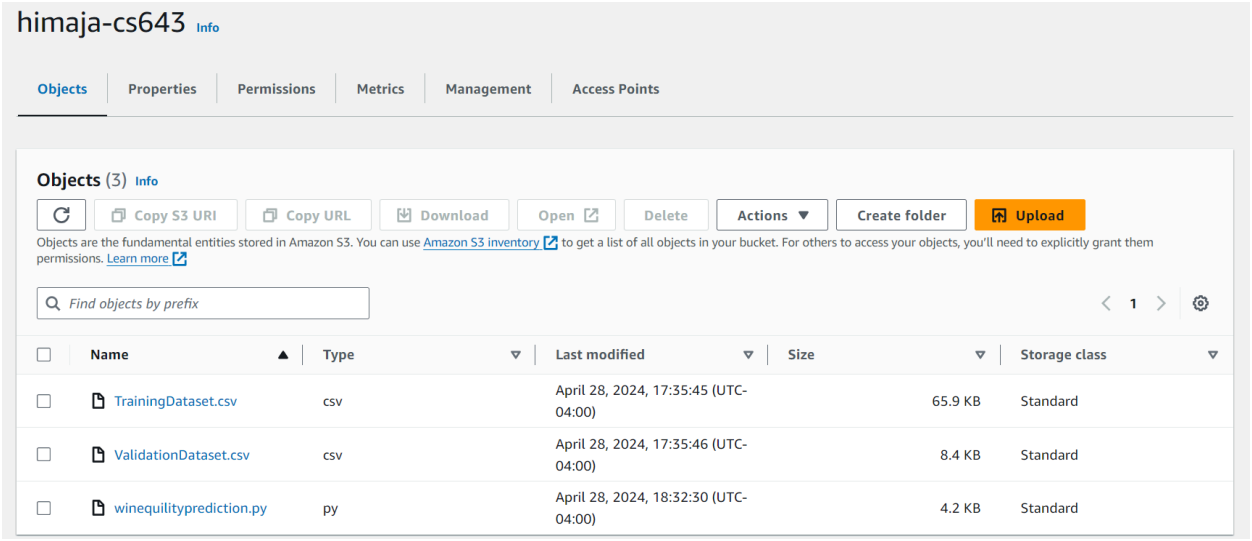
Github : https://github.com/himajakinthada28/CS643-PA2

Docker: https://hub.docker.com/r/himajakinthada/pa2-docker

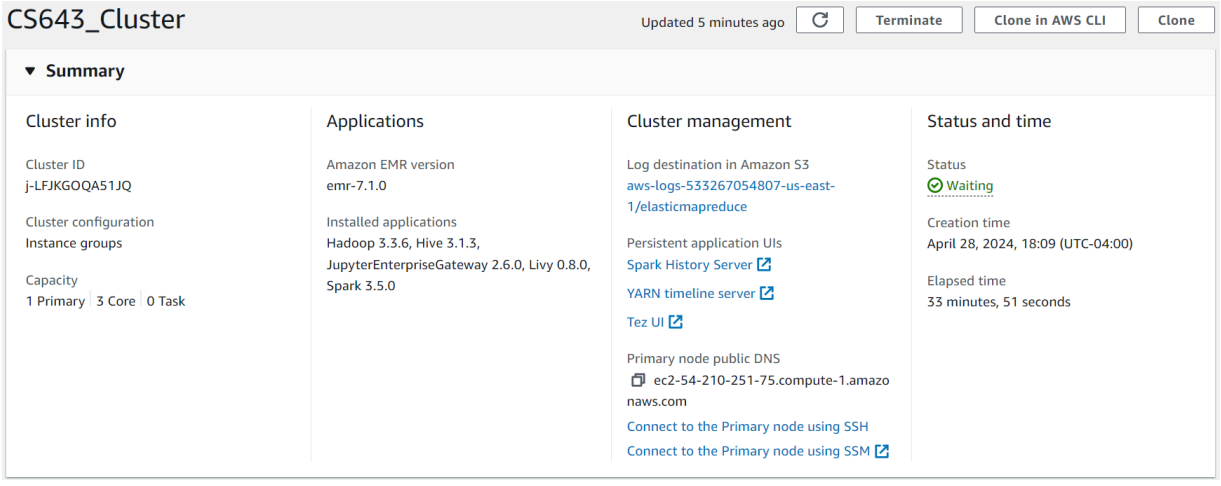


Steps to launch an EMR cluster and run the ML model without Docker

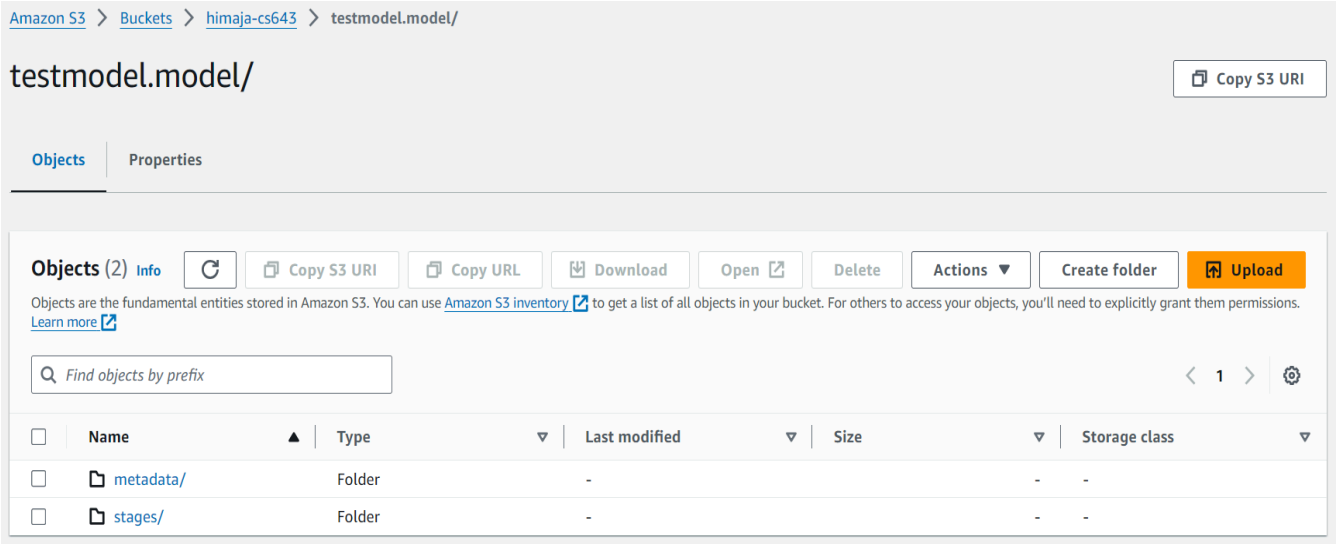
1) Create an s3 bucket and upload the winequilityprediction.py, TrainingDataset.csv and ValidationDataset.csv



2) Create an EMR cluster
   a) 3 core nodes (slave) and 1 primary node (master)
   b) m5.xlarge type of instances
   c) Check to configure Hadoop and spark in the instance
   d) Create a keypair to do ssh login in the primary instance
   e) Use default EMR roles in the IAM section
   f) Edit the inbound rules of the attached security group and allow ssh from everywhere (0.0.0.0/0)
   g) Wait for the status to show “waiting” for the cluster setup



3) Do SSH login to the primary instance using the security key-pair created in the above step.
4) Install numpy in the instance (pip install numpy)
5) Start the spark job using the command:
   spark-submit s3://himaja-cs643/winequilityprediction.py
6) Make sure to modify the path of the model output in the file winequilityprediction.py as per the s3 bucket name
7) Once the job completes, it will show the weighted F1 score and store the model in the s3 bucket with the name (**testmodel.model**).



F1 Score output on the terminal:

```
PipelineModel_3818802b2029
Test Accuracy1 of wine prediction model =  0.96875
Weighted f1 score of wine prediction model =  0.9547916666666667
[hadoop@ip-172-31-82-232 ~]$
```

Steps to run the ML model with Docker

We will install docker in a separate EC2 instance and try to build the image there itself. To accomplish this we need to put the github repo in the EC2 instance.

Launch an EC2 instance with "t2.medium" , "linux AMI" and allow SSH, HTTP and HTTPS traffic in the security group.

Steps to copy the Dockerfile from local to EC2 instance:

1) Go to local directory where the github repo folder is saved and do right click, Open in terminal
2) Open an scp connection with the EC2 instance using the below command
   scp -i "key_pair file path" <github repo folder> <EC2 instance DNS>:/home/ec2-user

3) Once the command is executed the file will be copied to the instance

```
[ec2-user@ip-172-31-59-24 ~]$ ls
cloudComputing-master
```

Steps to download docker in the EC2 instance:

Execute the below commands sequentially in the EC2 instance:

1) sudo yum update -y
2) sudo yum install docker -y
3) sudo service docker start
4) sudo usermod -a -G docker ec2-user

Use command **"docker –version"** to verify the installation

```
[ec2-user@ip-172-31-59-24 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:03:08 ago on Mon Apr 29 00:02:23 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                 Repository    Size
================================================================================
Installing:
 docker                  x86_64        25.0.3-1.amzn2023.0.1   amazonlinux   44 M
Installing dependencies:
 containerd              x86_64        1.7.11-1.amzn2023.0.1   amazonlinux   35 M
 iptables-libs           x86_64        1.8.8-3.amzn2023.0.2    amazonlinux  401 k
 iptables-nft            x86_64        1.8.8-3.amzn2023.0.2    amazonlinux  183 k
 libcgroup               x86_64        3.0-1.amzn2023.0.1      amazonlinux   75 k
 libnetfilter_conntrack  x86_64        1.0.8-2.amzn2023.0.2    amazonlinux   58 k
 libnfnetlink            x86_64        1.0.1-19.amzn2023.0.2   amazonlinux   30 k
 libnftnl                x86_64        1.2.2-2.amzn2023.0.2    amazonlinux   84 k
 pigz                    x86_64        2.5-1.amzn2023.0.3      amazonlinux   83 k
 runc                    x86_64        1.1.11-1.amzn2023.0.1   amazonlinux  3.0 M

Transaction Summary
================================================================================
Install  10 Packages

Total download size: 83 M
Installed size: 313 M
Downloading Packages:
(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm      4.7 MB/s | 401 kB     00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm       5.9 MB/s | 183 kB     00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm            3.4 MB/s |  75 kB     00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm 1.8 MB/s |  58 kB     00:00
(5/10): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64.rpm      1.4 MB/s |  30 kB     00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm           2.2 MB/s |  84 kB     00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm                 3.5 MB/s |  83 kB     00:00
(8/10): runc-1.1.11-1.amzn2023.0.1.x86_64.rpm               25 MB/s | 3.0 MB     00:00
(9/10): containerd-1.7.11-1.amzn2023.0.1.x86_64.rpm         50 MB/s |  35 MB     00:00
(10/10): docker-25.0.3-1.amzn2023.0.1.x86_64.rpm            39 MB/s |  44 MB     00:01
--------------------------------------------------------------------------------
Total                                                       70 MB/s |  83 MB     00:01
```

```
[ec2-user@ip-172-31-59-24 ~]$ docker --version
Docker version 25.0.3, build 4debf41
[ec2-user@ip-172-31-59-24 ~]$
```

Build the docker image using the command where the Dockerfile is present:

docker build -t cs643-docker .

If you face permission issues, then use the below command to resolve it:

=> sudo chmod 666 /var/run/docker.sock

Now login to Docker hub to push the image**.**

Command: docker login -u himajakinthada

```
[ec2-user@ip-172-31-59-24 ~]$ docker login -u himajakinthada
Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Push the image using the below commands:

Command:

docker tag pa2-docker himajakinthada/pa2-docker

docker push himajakinthada/pa2-docker

Now pull the image from the docker hub on the machine where you want to run the Docker image.

Command:

docker pull himajakinthada/pa2-docker

```
[ec2-user@ip-172-31-59-24 ~]$ docker pull himajakinthada/pa2-docker
Using default tag: latest
latest: Pulling from himajakinthada/pa2-docker
Digest: sha256:b85beb3bff12f6f558fdc27b70dde7d36e74b3abf7736b31066cf89de847ad54
Status: Image is up to date for himajakinthada/pa2-docker:latest
docker.io/himajakinthada/pa2-docker:latest
[ec2-user@ip-172-31-59-24 ~]$
```

Now run the Docker run command to execute the image and see the results.

Command:

docker run -v C:\Users\shash\OneDrive\Desktop\cs643- pa2\data\csv pa2-docker ValidationDataset.csv