Tyler Hayes

New Jersey Institute of Technology

Spring 2025

SP25-CS643852

Module 07 Assignment 01:

Programming Assignment 2

URL to DockerHub: https://hub.docker.com/repository/docker/tewq/wine-predictor-cli/general

URL to GitHub: https://github.com/TyHysNJIT/SP25-CS643852-TylerHayes-WinePredictor

---

Amazon Web Services Setup

Start five EC2 instances:

- Initialize them using the *22.04 Ubuntu AMI*
- Create a new SSH key and save it via the configuration screen.
- Build a security group which will be used to allow only your SSH (via "My IP"), and communication between the machines on the necessary ports.
- For the instance type, select "*r6a.2xlarge*".
- Configure 32 GiB of storage for each root volume
- I named these instances as so:
    - CS643852-M3-A4 Instance Training 1
    - CS643852-M3-A4 Instance Training 2
    - CS643852-M3-A4 Instance Training 3
    - CS643852-M3-A4 Instance Training 4
    - CS643852-M3-A4 Instance Testing 1
- Connect to the Training 1 instance and generate an SSH key by running:
    - *ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa*
- Run this for each worker EC2 instance by replacing the private IP each time:
    - *cat ~/.ssh/id_rsa.pub | ssh -i /home/ubuntu/CS643852-M3-A4.pem ubuntu@<WORKERS_PRIVATE_IP> 'mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys'*
        - *This will securely copy our new SSH key from the Training 1 instance into the authorized keys for the other EC2 instances.*

Install Dependencies:

- Run these commands on all EC2 instances, including the testing instance:

```
sudo apt update && sudo apt install -y openjdk-11-jdk scala git
wget https://dlcdn.apache.org/spark/spark-3.5.5/spark-3.5.5-bin-hadoop3.tgz
tar -xvf spark-3.5.5-bin-hadoop3.tgz
sudo mv spark-3.5.5-bin-hadoop3 /opt/spark
echo 'export SPARK_HOME=/opt/spark' >> ~/.bashrc
echo 'export PATH=$PATH:$SPARK_HOME/bin' >> ~/.bashrc
source ~/.bashrc
```

This will refresh our package list, install the Java development kit, install Scala, and install Git all at once. Then it will download Spark with Hadoop 3 bundled in already. From there, we just 'unzip' the file and declare some environment variables for Spark configuration later on.

Establish a 'Master' node:

- Pick one instance to be the master node. For this guide I will be using Training 1, but in practice any of these EC2 instances can serve as the master if desired.
- Update $SPARK_HOME/conf/spark_env.sh:

```
export SPARK_MASTER_HOST=<MASTER_PRIVATE_IP>
```

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

- Update $SPARK_HOME/conf/slaves
    - This should contain a row with one private IP address for each worker EC2 instance.

You should now be able to launch spark via $SPARK_HOME/sbin/start-all.sh and stop via $SPARK_HOME/sbin/stop-all.sh.

Create an IAM Role:

- We will need to create an IAM role to grant the EC2 instances access to the datasets, as well as a way to write the final model somewhere.
    - To do this, we can go to the IAM page
        - Select "New Role"
        - Select "AWS Service" for entity type
        - Select "EC2 Instance" for Use Case
        - Add AmazonS3FullAccess from the permissions/policies.
        - Click next
        - Define the role name
        - Click create role
    - Assign this role to all of the EC2 instances, granting them access to the S3 storage.

Modeling

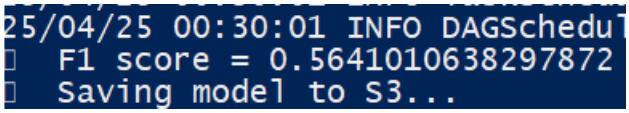
Compile & send the .java Code:

Note: My code is hard-coded to use the pointer to my S3 storage. As a result, you would need to change this in your version to match whatever origin file-store you are using.

- Install Java Developer Kit & Maven
- If your present working directory is the 'WinePredictor' folder, run '*mvn clean compile assembly:single -DskipTests*' to compile the training code with the dependencies using Maven.
- Use scp to transmit the compiled .jar file to your master node:
    - **scp -i "C:\my_key.pem" "C:\ wine-predictor-1.0-SNAPSHOT-jar-with-dependencies.jar" ^ ubuntu@<MASTER_PUBLIC_IP>:~/**

Execute the .jar by submitting a Spark job:

- spark-submit --class com.hvel.WinePredictor --master spark://<MASTER_PRIVATE_IP>:7077 --packages org.apache.hadoop:hadoop-aws:3.3.4,com.amazonaws:aws-java-sdk-bundle:1.11.1026 wine-predictor-1.0-SNAPSHOT-jar-with-dependencies.jar
    - This will train the model on the 'TrainingDataset.csv' file and test it on the 'ValidationDataset.csv' file.
    - The model will output the F1 score in the console based on the validation set:



- Your model should now also be saved in S3.

Predicting

Using Docker

To make a prediction using Docker, we will need to install Docker on our EC2 instance and add ourselves to the group. We can run these commands on our 'Testing 1' machine:

```
sudo apt update
sudo apt install -y docker.io
sudo usermod -aG docker $USER
newgrp docker  # to refresh group without logout
```

You will either need to compile the .jar yourself and run it directly, or you can use Docker. If you compile the .jar directly you will need to SCP it into the same folder as the saved-model folder.

You can pull the latest docker image and run it by using these commands:

```
docker pull tewq/wine-predictor-cli
docker run tewq/wine-predictor-cli
```

In the below examples I will be calling this from it's local Docker name on my machine, which I SCP'd over to the EC2 instance since I had not published it to Dockerhub yet. This is the same Docker container, though.

You can make predictions by calling the docker run function on the wine predictor container. By passing in the features as arguments/parameters to the function call, you can get the model to make a prediction about the wine quality.

```
ubuntu@ip-172-31-14-65:~/wine-app$ docker run --rm wine-predictor-cli 8.9 0.22 0.48 1.8 0.077 29 60 0.9968 3.39
+----------+
|prediction|
+----------+
|5.0       |
+----------+
```

Using Java Directly

If we have compiled the command line interface code for this approach, we can either SCP it over and call the file with our same predictions. We simply move the 'saved-model' file into the target folder for the CLI code, and then we can run the jar directly with our arguments as so:

```
ubuntu@ip-172-31-14-65:~/wine-app/target$ java -jar wine-predictor-cli-1.0-jar-with-dependencies.jar 8.9 0.22 0.4
.077 29 60 0.9968 3.39 0.53 9.4
+----------+
|prediction|
+----------+
|5.0       |
+----------+
```