CS 643(Sec 852)
Prog Assignment 2

Shah Rahim
sr278
sr278@njit.edu

Background:

This application is used to train and predict wine tasting csv files.

1. The training application **TrainingApplication.java** is in charge of taking in the TrainingDataset.csv, removing any bad characters like the quotation marks from the csv and then using the csv dataset to train a model
2. The prediction application **PredictionApplication.java** loads the trained model and takes in a csv validation dataset as input and then runs the model providing an **F1 score**

Machine Learning Model used:

This project chose to use the **RandomForestClassificationModel**.

Github Repository:

The repository for this project containing all source code can be found here:

https://github.com/sr278-njit/apache-spark-predictor

Docker Hub Repository:

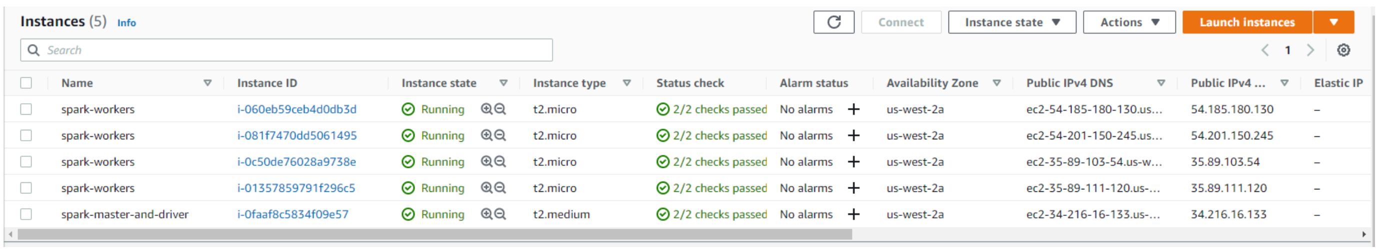
The repository for this project containing the docker image is:

https://hub.docker.com/repository/docker/sr278/cloud-computing

Apache Spark Setup:

We will first look up how to set up an Apache spark environment to run these applications. You will need the following hardware specs:

1. You will need **4** t2.micro EC2 instances that will be used as the workers
2. You will need **1** t2.medium EC2 instance to run the Master Spark node. This EC2 instance will also run the Training and Prediction applications

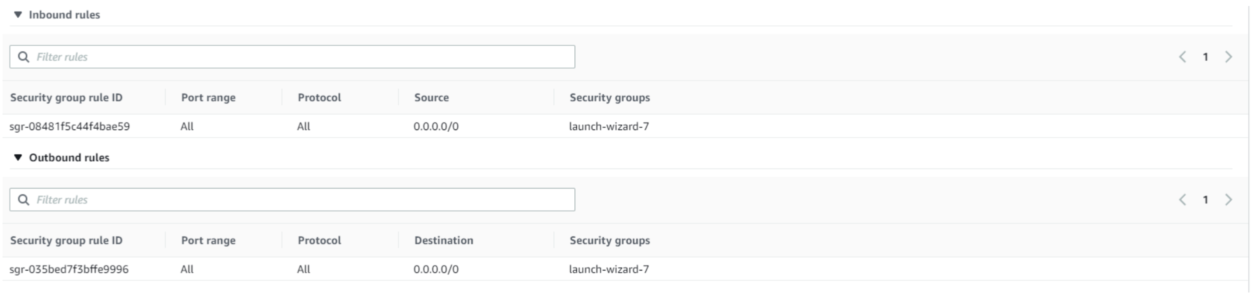


Instances (5) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
spark-workers	i-060eb59ceb4d0db3d	Running	t2.micro	2/2 checks passed	No alarms	us-west-2a	ec2-54-185-180-130.us...	54.185.180.130	–
spark-workers	i-081f7470dd5061495	Running	t2.micro	2/2 checks passed	No alarms	us-west-2a	ec2-54-201-150-245.us...	54.201.150.245	–
spark-workers	i-0c50de76028a9738e	Running	t2.micro	2/2 checks passed	No alarms	us-west-2a	ec2-35-89-103-54.us-w...	35.89.103.54	–
spark-workers	i-01357859791f296c5	Running	t2.micro	2/2 checks passed	No alarms	us-west-2a	ec2-35-89-111-120.us-...	35.89.111.120	–
spark-master-and-driver	i-0faaf8c5834f09e57	Running	t2.medium	2/2 checks passed	No alarms	us-west-2a	ec2-34-216-16-133.us-...	34.216.16.133	–

Security Groups:

Please make sure the security groups are open to **ALL** traffic for inbound and outbound rules. This will ensure there are no discrepancies regarding networking:



Inbound rules

Security group rule ID	Port range	Protocol	Source	Security groups
sgr-08481f5c44f4bae59	All	All	0.0.0.0/0	launch-wizard-7

Outbound rules

Security group rule ID	Port range	Protocol	Destination	Security groups
sgr-035bed7f3bffe9996	All	All	0.0.0.0/0	launch-wizard-7

Installing Spark/Docker packages:

1. On all 5 EC2 instances, please ssh into them as user **ubuntu**@EC2-public-ip-address
2. Please copy over the **spark-setup.sh** file from the github repository under /home/ubuntu
3. Change the permissions on the script to **755(execute)**
4. Run the script as root:

   sudo /home/ubuntu/spark-setup.sh

   All of the packages needed will be installed. Please run these steps on all 5 EC2 instances

Running Master Spark Node:

1. Please copy over the **start-stop-spark.sh** shell script to all 5 EC2 instances under /home/ubuntu
2. Change the permissions on the script to **755(execute)**
3. To run the Master, please run the script like so(make sure the specify the 'master' arg):

   sudo /home/ubuntu/start-stop-spark.sh master

   This will begin the Master Spark instance

4. Make sure the process is running:

```
ubuntu@ip-172-31-43-245:~/apache-spark-predictor$ ps -ef | grep spark
root       5885       1  0 Apr27 ?        00:00:00 sudo nohup ./spark-class org.apache.spark.deploy.master.Master --ip ip-172-31-43-245 --port 7077 --webui-port 8090
root       5886    5885  0 Apr27 ?        00:02:59 /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /opt/spark/conf/:/opt/spark/jars/* -Xmx1g org.apache.spark.deploy.master.Master --ip ip-172-31-43-245 --port 7077 --webui-port 8090
ubuntu    95197   55206  0 02:45 pts/1    00:00:00 grep --color=auto spark
ubuntu@ip-172-31-43-245:~/apache-spark-predictor$
```

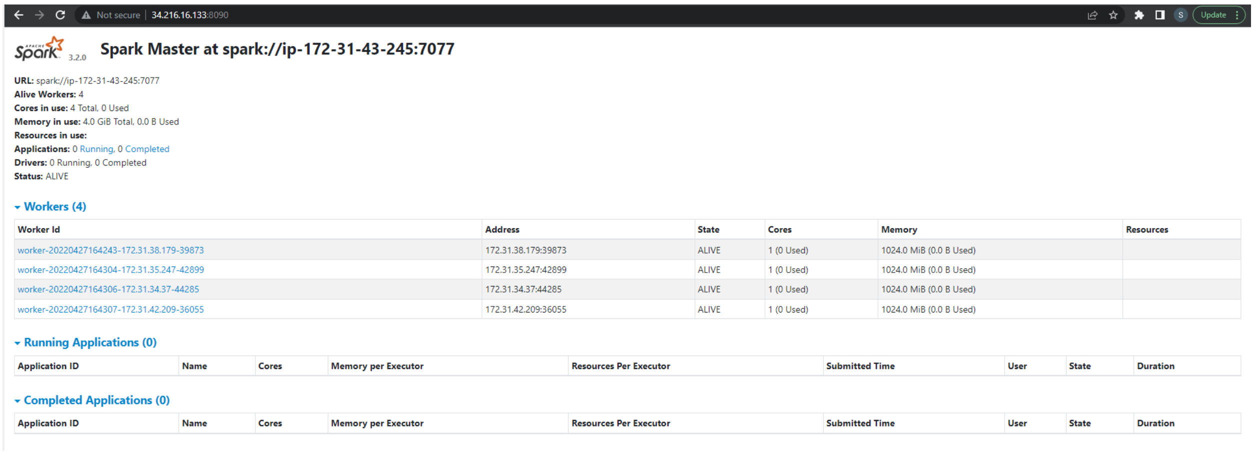
Running Worker Spark Nodes:

1. Make sure the **start-stop-spark.sh** script has been copied to **ALL** nodes and the execute permissions have been set
2. Grab the private IP address of the master. For example the Master private IP in running this project was **ip-172-31-43-245**
3. Run the following command on all Worker nodes(make sure to use your Masters private IP):

   ./start-stop-spark.sh worker spark://ip-172-31-43-245:7077

   The port 7077 is because the Spark Master is listening on the default port 7077 for Workers to communicate with them

4. Validate all workers are registered with the master. Go to the **Master's** public IP address on port 8090. You will see the Master UI with the Workers registered. For example:



Spark Master at spark://ip-172-31-43-245:7077

URL: spark://ip-172-31-43-245:7077
Alive Workers: 4
Cores in use: 4 Total, 0 Used
Memory in use: 4.0 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (4)

Worker Id	Address	State	Cores	Memory	Resources
worker-20220427164243-172.31.38.179-39873	172.31.38.179:39873	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20220427164304-172.31.35.247-42899	172.31.35.247:42899	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20220427164306-172.31.34.37-44285	172.31.34.37:44285	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20220427164307-172.31.42.209-36055	172.31.42.209:36055	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Running Training Application:

Make sure to be ssh'd into the **Master node**. You will need to clone the github repository under **/home/ubuntu**

1. Clone the github repository under **/home/ubuntu**

```
ubuntu@ip-172-31-43-245:~$ git clone https://github.com/sr278-njit/apache-spark-predictor.git
Cloning into 'apache-spark-predictor'...
remote: Enumerating objects: 49, done.
remote: Counting objects: 100% (49/49), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 49 (delta 4), reused 45 (delta 4), pack-reused 0
Unpacking objects: 100% (49/49), 29.58 KiB | 2.11 MiB/s, done.
```

2. Make sure all source code is present:

```
ubuntu@ip-172-31-43-245:~$ cd apache-spark-predictor
ubuntu@ip-172-31-43-245:~/apache-spark-predictor$ ls
Dockerfile  README.md  TrainingDataset.csv  ValidationDataset.csv  pom.xml  predictor-start.sh  setup-spark.sh  src
start-stop-spark.sh  target
```

3. You will need to set the **TRAINING_DATASET** environment variable so that the TrainingApplication.java class can read in the csv dataset. If you like, please rerun the app using ValidationDataset.csv as the TRAINING_DATASET.

   Note the TrainingDataset.csv file is already provided in the repository:

   export TRAINING_DATASET=/home/ubuntu/apache-spark-predictor/TrainingDataset.csv

   NOTE:

   The application removes the unnecessary quotes from the csv files and writes a new file in the same path as **Sanitized_<TRAINING_DATASET>**

4. To ensure we are building the latest source code, please run the following command within the repository where the **pom.xml** file is under **/home/ubuntu/apache-spark-predictor**:

```
ubuntu@ip-172-31-43-245:~/apache-spark-predictor$ mvn clean install exec:java
-Dexec.mainClass="net.njit.apache.spark.trainer.TrainingApplication" -e
```

5. This command will freshly download the packages needed to run the application. Please note it might take upto a minute to get everything downloaded. Once the dependencies are resolved, you will see the log outputs. The highlighted parts are important as they show how the model performed with the F1 score and accuracy. :

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
```

```
22/04/29 03:18:58 INFO SparkContext: Running Spark version 3.2.0
22/04/29 03:18:58 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using
builtin-java classes where applicable
22/04/29 03:18:58 INFO ResourceUtils:
==============================================================
22/04/29 03:18:58 INFO ResourceUtils: No custom resources configured for spark.driver.
22/04/29 03:18:58 INFO ResourceUtils:
==============================================================
22/04/29 03:18:58 INFO SparkContext: Submitted application: trainer
22/04/29 03:18:58 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores
-> name: cores, amount: 1, script: , vendor: , memory -> name: memory, amount: 1024, script: , vendor: ,
offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount:
1.0)
22/04/29 03:18:58 INFO ResourceProfile: Limiting resource is cpu
22/04/29 03:18:58 INFO ResourceProfileManager: Added ResourceProfile id: 0
22/04/29 03:18:58 INFO SecurityManager: Changing view acls to: ubuntu
22/04/29 03:18:58 INFO SecurityManager: Changing modify acls to: ubuntu
22/04/29 03:18:58 INFO SecurityManager: Changing view acls groups to:
22/04/29 03:18:58 INFO SecurityManager: Changing modify acls groups to:
22/04/29 03:18:58 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users
with view permissions: Set(ubuntu); groups with view permissions: Set(); users  with modify permissions:
Set(ubuntu); groups with modify permissions: Set()
22/04/29 03:18:59 INFO Utils: Successfully started service 'sparkDriver' on port 37515.
22/04/29 03:18:59 INFO SparkEnv: Registering MapOutputTracker
22/04/29 03:18:59 INFO SparkEnv: Registering BlockManagerMaster
22/04/29 03:18:59 INFO BlockManagerMasterEndpoint: Using
org.apache.spark.storage.DefaultTopologyMapper for getting topology information
22/04/29 03:18:59 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
22/04/29 03:18:59 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
22/04/29 03:18:59 INFO DiskBlockManager: Created local directory at
/tmp/blockmgr-eaf68cb0-16d6-4a0a-826b-4b9cdec263f5
22/04/29 03:18:59 INFO MemoryStore: MemoryStore started with capacity 409.2 MiB
22/04/29 03:18:59 INFO SparkEnv: Registering OutputCommitCoordinator
22/04/29 03:18:59 INFO Utils: Successfully started service 'SparkUI' on port 4040.
22/04/29 03:18:59 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at
http://ip-172-31-43-245.us-west-2.compute.internal:4040
22/04/29 03:18:59 INFO Executor: Starting executor ID driver on host
ip-172-31-43-245.us-west-2.compute.internal
22/04/29 03:18:59 INFO Utils: Successfully started service
'org.apache.spark.network.netty.NettyBlockTransferService' on port 38641.
22/04/29 03:18:59 INFO NettyBlockTransferService: Server created on
ip-172-31-43-245.us-west-2.compute.internal:38641
22/04/29 03:18:59 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for
block replication policy
22/04/29 03:18:59 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver,
ip-172-31-43-245.us-west-2.compute.internal, 38641, None)
22/04/29 03:18:59 INFO BlockManagerMasterEndpoint: Registering block manager
ip-172-31-43-245.us-west-2.compute.internal:38641 with 409.2 MiB RAM, BlockManagerId(driver,
ip-172-31-43-245.us-west-2.compute.internal, 38641, None)
22/04/29 03:18:59 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver,
ip-172-31-43-245.us-west-2.compute.internal, 38641, None)
22/04/29 03:18:59 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver,
ip-172-31-43-245.us-west-2.compute.internal, 38641, None)
TRAINING_DATASET value is: /home/ubuntu/apache-spark-predictor/TrainingDataset.csv
```

```
Loaded dataset: /home/ubuntu/apache-spark-predictor/Sanitized_TrainingDataset.csv
Training RandomForestClassificationModel with dataset:
/home/ubuntu/apache-spark-predictor/Sanitized_TrainingDataset.csv
Gathering RandomForestClassificationModel summary for dataset:
/home/ubuntu/apache-spark-predictor/Sanitized_TrainingDataset.csv
Prediction Model Accuracy is: 0.6849100860046912
Prediction Model F1 score is: 0.6610944518195154
```

6. Essentially, the RandomForestClassificationModel was able to train the model using the dataset with an **F1** score of 66% and **Accuracy** of 68%

   NOTE:

   The following java code:

   ```
   model.write().overwrite().save("/home/ubuntu/model/randomForest");
   ```

   Writes the model to **/home/ubuntu/model/randomForest**. This will be used by the PredictionApplication.java file.

Running Prediction Application:

This Prediction application runs almost exactly the same way as the TrainingApplication.java program. Only difference is, this runs in a docker container. The image we will be using can be found here:

https://hub.docker.com/layers/205611107/sr278/cloud-computing/spark-predictor-v1.0/images/sha256-33ec3113ec5f0b0fe4ea09f44d240bf9402cf96f37c0688d0b5d0f85ae9885e3?context=repo

We have already installed docker in previous steps when we ran the **setup-spark.sh** script. Here are the steps to run the Prediction Application.

1. Verify docker is present on the machine:

   ```
   ubuntu@ip-172-31-43-245:~/apache-spark-predictor$ docker --version
   Docker version 20.10.12, build 20.10.12-0ubuntu2~20.04.1
   ```

2. Make sure you have the validation dataset file on the Master node. Remember, we are running the Training application, Prediction application and the Master Spark node on the same t2.medium EC2 instance.

3. Run the following docker command:

   sudo docker run -t -v /home/ubuntu/apache-spark-predictor/ValidationDataset.csv:/usr/ValidationDataset.csv -v /home/ubuntu/model/randomForest:/usr/model/randomForest -e MODEL_PATH=/usr/model/randomForest -e VALIDATION_DATASET=/usr/ValidationDataset.csv --net=host -i sr278/cloud-computing:spark-predictor-v1.0

   NOTE:

   /home/ubuntu/apache-spark-predictor/ValidationDataset.csv:/usr/ValidationDataset.csv

   This part makes sure the validation dataset is present in the docker container. It can be any dataset as long as the paths are valid

   /home/ubuntu/model/randomForest:/usr/model/randomForest

   This part makes sure the trained model is accessible by the docker container

   MODEL_PATH=/usr/model/randomForest

   This part sets the environment variable needed by the Prediction application to load the model.

   VALIDATION_DATASET=/usr/ValidationDataset.csv

   This part sets the environment variable needed by the Prediction application to load the dataset.

   --net=host

   This part opens ip[the docker containers network to the host.

   sr278/cloud-computing:spark-predictor-v1.0

   This is the image we will be running. The Docker file can be found in the project repository source code.

4. Make sure you use the correct paths where your dataset file is and set the right environment variables in the docker run command. After running the command, the container will freshly install the dependencies which will take upto a minute. The output will look like:

```
Loading RandomForestClassificationModel from: /usr/model/randomForest
ytes result sent to driver
22/04/29 03:42:09 INFO TaskSetManager: Finished task 0.0 in stage 7.0 (TID 7) in 54 ms on
ip-172-31-43-245.us-west-2.compute.internal (executor driver) (1/1)
22/04/29 03:42:09 INFO TaskSchedulerImpl: Removed TaskSet 7.0, whose tasks have all completed, from pool
22/04/29 03:42:09 INFO DAGScheduler: ResultStage 7 (collect at treeModels.scala:549) finished in 0.069 s
22/04/29 03:42:09 INFO DAGScheduler: Job 4 is finished. Cancelling potential speculative or zombie tasks for
this job
22/04/29 03:42:09 INFO TaskSchedulerImpl: Killing all running tasks in stage 7: Stage finished
22/04/29 03:42:09 INFO DAGScheduler: Job 4 finished: collect at treeModels.scala:549, took 0.922956 s
…
```

```
22/04/29 03:42:09 WARN SparkSession$Builder: Using an existing SparkSession; some spark core
configurations may not take effect.
VALIDATION_DATASET value is: /usr/ValidationDataset.csv
Loaded dataset: /usr/Sanitized_ValidationDataset.csv
Prediction Model Accuracy is: 0.53125
Prediction Model F1 score is: 0.5080882352941176
```

This run basically loads the dataset, removes any unnecessary quote characters. It then loads the model saved from the Training application and runs a prediction performance test. The **F1** score is 51% which is not the best.

5. To run this app outside of docker, please review the **predictor-start.sh** script which contains all you need to run the Prediction Application.