CS643 Programming Assignment #2: Wine Quality Predictions

Tavaris Walton
08/2/2024
CS643

GitHub Link: https://github.com/tavwalt/CS643-AWS-ProgAssgn-2
DockerHub Link: https://hub.docker.com/r/tavwalt2/my-pyspark-app

Abstract:

The objective of this final programming assignment is to create an Apache Spark MLlib application to train a machine learning model in parallel on a cluster composed of four workers and one master. The document will showcase a step-by-step process on how to set up the cluster namely EC2 instances, and docker images. Additionally, the **parallel training steps are specified** as well as the steps to run the prediction application on both a **single machine without docker** and through downloading the docker image on a machine, instantiating a container, and **running the container on a single machine**. The code can be found on GitHub and the image can be found on Docker Hub. My implementation utilizes *Apache Spark with PySpark* and *Python3* for this programming assignment.

Training Setup:

On the AWS Management Console, navigate to **Services** **"EC2"** Launch Instances Launch Instances. Enter ***5*** for # of instances. Select the "***Ubuntu Server 22.04 LTS (HVM),EBS General Purpose (SSD)***." Select the ***t2.large*** type for memory will help when running Docker and more. Select ***Create a new key pair*** and name it ***EC2-A-KeyPair***. Hit ***Download key pair.*** Under ***Network Settings -> Security groups (Firewall),*** check ***Allow SSH traffic from [Anywhere 0.0.0.0/0]***. For ***Configure storage***, configure it from ***8*** GiB to ***16*** GiB (to install and configure modules/packages on the EC2 instances). Keep all the rest of the default options and click ***Launch instance.*** Then ***View all instances***. You will see a **Pending** status for the ***Instance State*** of the EC2 Instances.

Note: The t2-large instances will help with memory. And for security reasons SSH ip should always be specific and never just open to the public.

Run the following command to set the correct permissions for the .pem file you downloaded:

```
$ chmod 400 EC2-A-KeyPair.pem
```

To connect to your EC2 instance (after it has started running), run the following command in your terminal (replacing <YOUR_INSTANCE_PUBLIC_DNS> with the "Public IPv4 DNS" attribute of the EC2 instance):

$ ssh -i "EC2-A-KeyPair.pem" ubuntu@<YOUR_EC2_INSTANCE_PUBLIC_IPV4_ADDRESS>

> **NB:** Make sure you can **SSH** into your machines, as that is one of the requirements needed for **behind-the-scenes communication**

Apache Spark Setup

```
$sudo apt-get update
$sudo apt-get upgrade
$sudo apt-get install -y curl vim wget software-properties-common
ssh net-tools ca-certificates
$sudo apt install -y default-jre      {installing a default java}
$sudo apt install net-tools
$sudo mkdir -p /opt/spark
$wget https://dlcdn.apache.org/spark/spark-3.5.1/spark-3.5.1-
bin-hadoop3.tgz
$sudo tar -zxvf spark-3.5.1-bin-hadoop3.tgz
$sudo mv spark-3.5.1-bin-hadoop3 /opt/spark
```

Run these pip install commands before proceeding:

```
$sudo apt install python3-pip
$pip install numpy
$pip install pandas
$pip install quinn
$pip install pyspark
$pip install findspark

$sudo apt-get install ipython3
$pip3 install jupyter
$pip3 install py4j
$sudo apt install openjdk-19-jdk openjdk-19-jre
$sudo apt-get install scala
```

NB: $ **`python3 --3version`** used to check if python3 was installed on Linux/Ubuntu
NB: $ **`java -version`** used to check if java was install/availability on Linux/Ubuntu
NB: $ **`scala -version`** used to check if scala was install/availability on Linux/Ubuntu

Preparing Your Environment

`Navigate to $`**`vi ~/.bashrc`**` to Open environmental variable and add info at the bottom of file:`

```
export JAVA_HOME=/usr/lib/jvm/java-19-openjdk-amd64
export PATH=$PATH:JAVA_HOME/bin
export PYSPARK_PYTHON=python3
export SPARK_HOME=/opt/spark/spark-3.5.1-bin-hadoop3
export PATH=$PATH:$SPARK_HOME/bin
export PATH=$PATH:$SPARK_HOME/sbin
export PYTHONPATH=$SPARK_HOME/python:$PYTHONPATH
```

*When finished Press **ESC** then type **:wq!** Press **ENTER** to save*

***Reboot** machine
*Type `$spark-shell` to test if it was installed successfully

Add additional Environmental variables

ENTER values for [Master]

$ vi /opt/spark/spark-3.5.1-bin-hadoop3/conf/spark-defaults.conf.template

Remove the # signs

```
spark.master                  spark://master:7077
~~spark.executor.memory          6g~~
spark.eventLog.enabled              true
spark.eventLog.dir            file:///opt/spark/tmp
spark.serializer          org.apache.spark.serializer.KryoSerializer
spark.driver.memory          ~~2g or~~ 5g
spark.executor.extraJavaOptions  -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two three"
```

When finished Press **ESC** then type **:wq!** Press **ENTER** to save

***Reboot** machine*

ENTER values for [Worker nodes 1-4]

a. Edit TWO files **BEFORE** starting each of your **Worker {1-4}:**

```
$ cd /opt/spark/spark-3.5.1-bin-hadoop3/conf
$vi spark-env.sh
     export SPARK_PUBLIC_DNS=External-IP-addressOFMaster
     export SPARK_WORKER_WEBUI_PORT=8081
```

$ vi /opt/spark/spark-3.5.1-bin-hadoop3/conf/spark-defaults.conf.template

```
spark.master    spark://Internal-IP-addressOFMaster:7077
spark.eventLog.enabled              true
spark.serializer          org.apache.spark.serializer.KryoSerializer
spark.executor.memory        5g
spark.executor.extraJavaOptions  -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two three"
```

Security Groups in AWS Management Console:

- Go to the AWS Management Console.
- Navigate to "EC2" and then "Security Groups."
- Find the security group associated with your EC2 instance.
- Edit **inbound** rules to allow traffic on the port used by [***Spark Master only***] **(default is 8080 Web UI & 7077 for Workers to connect):**
    - Type: Custom TCP Rule
        - Protocol: TCP
        - Port Range: **8080**
        - Source: You can specify `0.0.0.0/0` to allow access from anywhere, or restrict it to your IP address for security.
    - Type: Custom TCP Rule
        - Protocol: TCP
        - Port Range: **7077 -** The port on which the Spark master listens for incoming connections from Spark workers and client applications.
        - Source: You can specify `0.0.0.0/0` to allow access from anywhere, or restrict it to your IP address for security.

[***For Spark Master only***]

- Edit **inbound** rules to allow traffic on the port used by **Jupyter Notebook (default is 8888):**
    - Type: Custom TCP Rule
    - Protocol: TCP
    - Port Range: 8888
    - Source: You can specify `0.0.0.0/0` to allow access from anywhere, or restrict it to your IP address for security.

[***For Spark Worker only***]

- Edit **inbound** rules to allow traffic on the port used by **Workers nodes for (SPARK_WORKER_WEBUI_PORT)**
    - Type: Custom TCP Rule
    - Protocol: TCP
    - Port: 7078 - The port on which Spark workers register themselves with the master and communicate with it.
    -
    - Source: You can specify `0.0.0.0/0` to allow access from anywhere, or restrict it to your IP address for security.

SKIP TO PAGE [7] TO START APPLICATION FIRST!

Will be needed to view Running Applications in the **"Training: Prediction without Docker and Prediction with Docker"** section:

How to obtain the Public IP Address of Your EC2 Instance?

- Go to the AWS Management Console.
- Navigate to "EC2" and then "Instances."
- Find your EC2 instance and note its public IP address or DNS name.

- ### How to view/access the [spark Master] Web UI?

  - Open a web browser and enter the **Public IPv4 DNS** or address name of your EC2 instance followed by the port number used by Spark Master.

    For example: http://ec2-54-147-145-250.compute-1.amazonaws.com:8080

- ### How to view/access the [spark Workers 1-4] Web UI?

  - Open a web browser and enter the **Public IPv4 DNS** or address name of your EC2 instance followed by the port number used by Spark Master.

    For example: http://ec2-100-27-204-125.compute-1.amazonaws.com:7081

Training: Prediction without Docker:

Step1:

SSH into your EC2 instance

```
$ ssh -i "EC2-A-KeyPair.pem" ec2-user@<YOUR_EC2_INSTANCE_PUBLIC_IPV4_ADDRESS>
```

Step2 - Start Master and Work engines:

b. Start **Master**

```
$ start-all.sh
```
(used to start Master and a Worker on same machine)

OR

```
$ start-master.sh
```
(used to start Master machine only!)

c. Edit TWO files **BEFORE** starting each of your **Worker {1-4}:**

```
$ vi /opt/spark/spark-3.5.1-bin-hadoop3/conf/spark-env.sh
          export SPARK_PUBLIC_DNS=External-IP-addressOFMaster
          export SPARK_WORKER_WEBUI_PORT=8081
```

This port is used to access the Spark web interfaces for monitoring and managing the cluster.

```
$ vi /opt/spark/conf/spark-defaults.conf
     spark.master   spark://Internal-IP-addressOFMaster:7077
```

d. Now Start each **Worker {1-4}**

```
$ /opt/spark/spark-3.5.1-bin-hadoop3/sbin/start-worker.sh spark://Internal-IP-addressOFMaster:7077
```

NB: Every time you restart **any** AWS-EC2 instance you will have a different **Internal** and/or **External** IP-address

Step3 - Download:

Download all files on your EC2-Master Instance

Perform a pull down from the GitHub repository:

```
$ git clone https://github.com/tavwalt/CS643-AWS-ProgAssgn-2
```

Step4 – Run code in Terminal:

1. Navigate to the folder that *contains the downloads* for [CS643-AWS-ProgAssgn-2] that has **all** files
   a. Edit the .csv **Paths** shown in red below in the `TavarisTrain.py` and `TavarisPredictions.py` files:

      ## Load Training Dataset. Pull data, make header and 'inferSchema' so column has integer values(or appropirate values)

train_df = spark.read.format('csv').options(header='true', inferSchema='true', sep=';').load('/home/ubuntu/CS643-AWS-ProgAssgn-2/tavarisTrainingDataset.csv')

validation_df = spark.read.format('csv').options(header='true', inferSchema='true', sep=';').load('/home/ubuntu/CS643-AWS-ProgAssgn-2/tavarisValidationDataset.csv')

#Used at the end for Second set of Prediction values
train_RandForest = spark.read.format('csv').options(header='true', inferSchema='true', sep=';').load('/home/ubuntu/CS643-AWS-ProgAssgn-2/tavarisTrainingDataset.csv')

validation_RandForest = spark.read.format('csv').options(header='true', inferSchema='true', sep=';').load('/home/ubuntu/CS643-AWS-ProgAssgn-2/tavarisValidationDataset.csv')

Disclaimer

NB: I REMOVED ALL THE DOUBLE QUOTES (“”””) **FROM EACH COULUMN IN THE .CSV FILES MANUALLY; BECAUSE THE *Apache Spark Mlib* PROGRAM DOES NOT READ THE INPUT DATA IF QUOTES** (“”””) **ARE PRESNET.**

→IF YOU NEED TO TEST MY WORK, PLEASE REMOVE YOUR **QUOTES** (“”””) AND ADD YOUR PATHS

2. **Start/Run code** in terminal:
Run on Master Only
$spark-submit --master spark://Internal-ip-address:7077 TavarisTrain.py

```
Example:
$spark-submit --master spark://172.32.29.55:7077 TavarisTrain.py
$spark-submit --master spark://172.32.29.55:7077 TavarisPredictions.py
```

- How to view/access the [spark Master] Web UI?
  - Open a web browser and enter the **Public IPv4 DNS** or address name of your EC2 instance followed by the port number used by Spark Master.

    For example: http://ec2-54-147-145-250.compute-1.amazonaws.com:8080

- How to view/access the [spark Workers 1-4] Web UI?

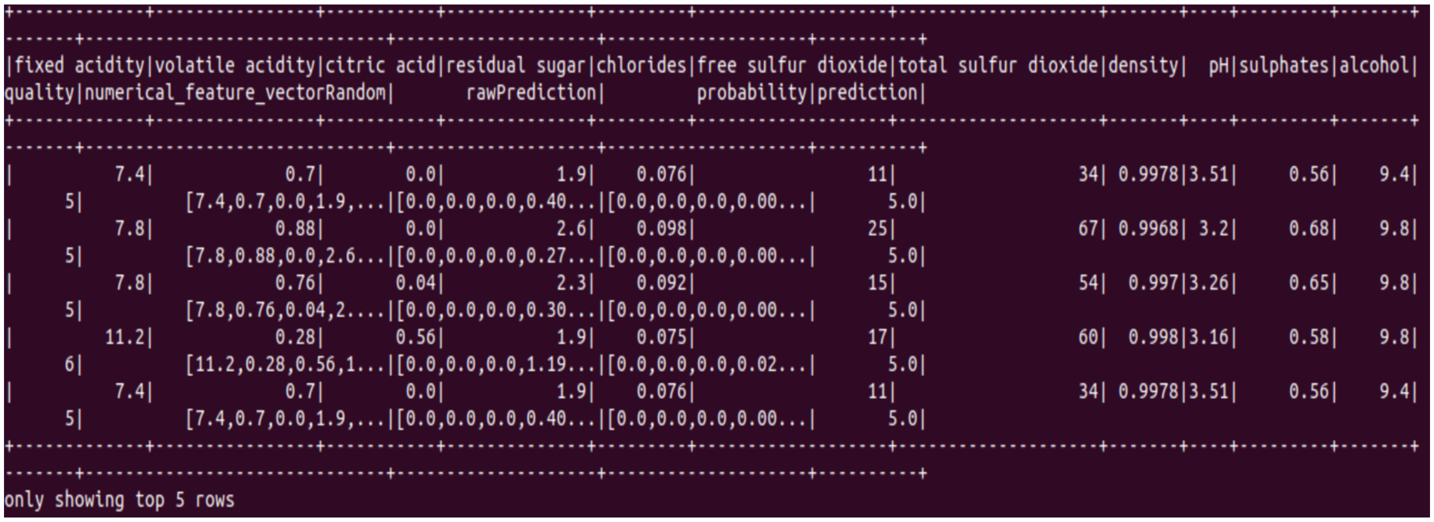
  - Open a web browser and enter the **Public IPv4 DNS** or address name of your EC2 instance followed by the port number used by Spark Master.

    For example: http://ec2-100-27-204-125.compute-1.amazonaws.com:7081

Snapshot from CMD

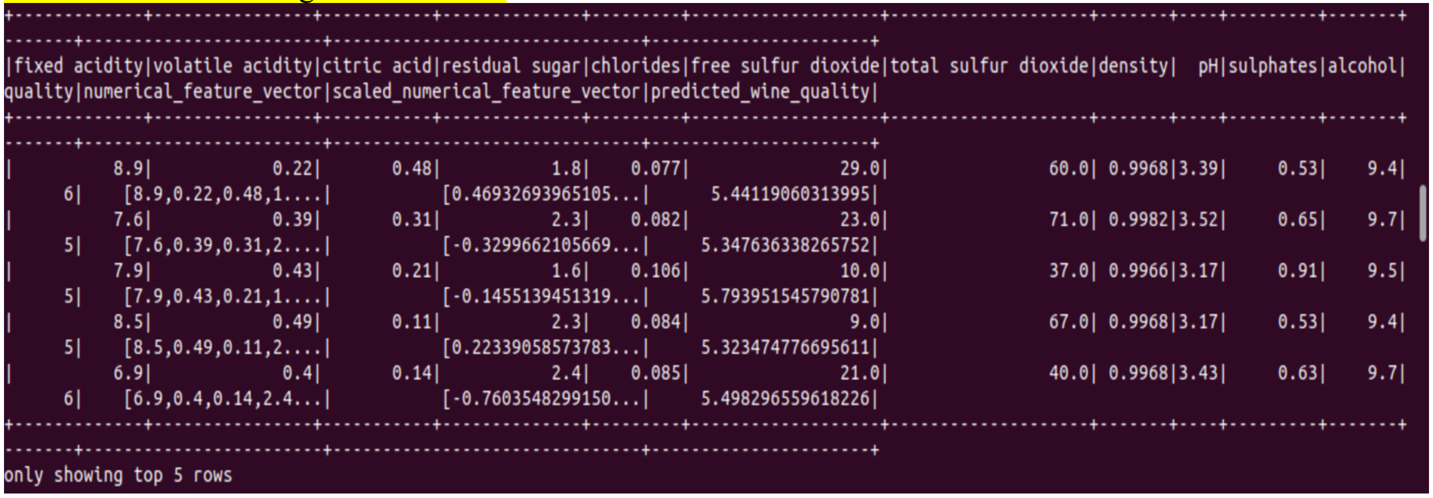
After running the *TavarisTraining.py* script, here is the output result received from my code:

F1 Score for RandomForestClassifier Model:



```
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------+------------------------------+--------------------+--------------------+----------+
|fixed acidity|volatile acidity|citric acid|residual sugar|chlorides|free sulfur dioxide|total sulfur dioxide|density|  pH|sulphates|alcohol|
quality|numerical_feature_vectorRandom|       rawPrediction|         probability|prediction|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------+------------------------------+--------------------+--------------------+----------+
|          7.4|             0.7|        0.0|           1.9|    0.076|                 11|                  34| 0.9978|3.51|     0.56|    9.4|
      5|          [7.4,0.7,0.0,1.9,...|[0.0,0.0,0.0,0.40...|[0.0,0.0,0.0,0.00...|       5.0|
|          7.8|            0.88|        0.0|           2.6|    0.098|                 25|                  67| 0.9968| 3.2|     0.68|    9.8|
      5|          [7.8,0.88,0.0,2.6...|[0.0,0.0,0.0,0.27...|[0.0,0.0,0.0,0.00...|       5.0|
|          7.8|            0.76|       0.04|           2.3|    0.092|                 15|                  54|  0.997|3.26|     0.65|    9.8|
      5|          [7.8,0.76,0.04,2....|[0.0,0.0,0.0,0.30...|[0.0,0.0,0.0,0.00...|       5.0|
|         11.2|            0.28|       0.56|           1.9|    0.075|                 17|                  60|  0.998|3.16|     0.58|    9.8|
      6|          [11.2,0.28,0.56,1...|[0.0,0.0,0.0,1.19...|[0.0,0.0,0.0,0.02...|       5.0|
|          7.4|             0.7|        0.0|           1.9|    0.076|                 11|                  34| 0.9978|3.51|     0.56|    9.4|
      5|          [7.4,0.7,0.0,1.9,...|[0.0,0.0,0.0,0.40...|[0.0,0.0,0.0,0.00...|       5.0|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------+------------------------------+--------------------+--------------------+----------+
only showing top 5 rows
```

F1 Score for Linear Regression Model:



```
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------+------------------------+-------------------------------+----------------------+
|fixed acidity|volatile acidity|citric acid|residual sugar|chlorides|free sulfur dioxide|total sulfur dioxide|density|  pH|sulphates|alcohol|
quality|numerical_feature_vector|scaled_numerical_feature_vector|predicted_wine_quality|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------+------------------------+-------------------------------+----------------------+
|          8.9|            0.22|       0.48|           1.8|    0.077|               29.0|                60.0| 0.9968|3.39|     0.53|    9.4|
      6|    [8.9,0.22,0.48,1....|           [0.46932693965105...|      5.44119060313995|
|          7.6|            0.39|       0.31|           2.3|    0.082|               23.0|                71.0| 0.9982|3.52|     0.65|    9.7|
      5|    [7.6,0.39,0.31,2....|           [-0.3299662105669...|     5.347636338265752|
|          7.9|            0.43|       0.21|           1.6|    0.106|               10.0|                37.0| 0.9966|3.17|     0.91|    9.5|
      5|    [7.9,0.43,0.21,1....|           [-0.1455139451319...|     5.793951545790781|
|          8.5|            0.49|       0.11|           2.3|    0.084|                9.0|                67.0| 0.9968|3.17|     0.53|    9.4|
      5|    [8.5,0.49,0.11,2....|           [0.22339058573783...|     5.323474776695611|
|          6.9|             0.4|       0.14|           2.4|    0.085|               21.0|                40.0| 0.9968|3.43|     0.63|    9.7|
      6|     [6.9,0.4,0.14,2.4...|           [-0.7603548299150...|     5.498296559618226|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+
-------+------------------------+-------------------------------+----------------------+
only showing top 5 rows
```

Linear Regression Model has a higher score than the RandomForestClassifier Model; I used **Linear Regression** for my prediction application.

Step5 [Optional] – Run code with "**Jupyter Notebook**" a web IDE**:**

1. In terminal type
   a. **`$pip3 install jupyter`**
   b. `$ jupyter notebook --generate-config` (will be default config)
   c. `$ mkdir cert`
   d. `$ cd cert`
   e. `$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout mycert.pem -out mycert.pem`
   f. `$ sudo chown $USER:$USER /home/ubuntu/cert/mycert.pem`
   g. `$ sudo chmod -R a+rw /home/ubuntu/cert/mycert.pem`
   h. `$ cd .jupyter`

   i. `$ vi jupyter_notebook_config.py` (edit config file to add info below)

```
c = get_config()  #noqa
c.NotebookApp.certfile = u'/home/ubuntu/cert/mycert.pem'
c.NotebookApp.ip = '*'
c.NotebookApp.port = 8888
c.NotebookApp.open_browser = False
```

2. Close out of your SSH instance and SSH again into your EC2. Run the following code to configure Jupyter with a password (of your choice when prompted) and start up the Jupyter Notebook on the EC2 instance.

   a. `$ jupyter notebook password` (optional)

   b. `$ jupyter notebook`

3. **Copy** the encrypted URL link in your terminal and replace `<YOUR_INSTANCE_PUBLIC_DNS>`. Then **Paste** it in your browser
   For Example
   `https://<YOUR_INSTANCE_PUBLIC_DNS>:8888/tree?token=72231f56b989181d0747f4348edb45bc7dbe1454c7fe18a5`

4. Create a new create a new Python 3 notebook by:

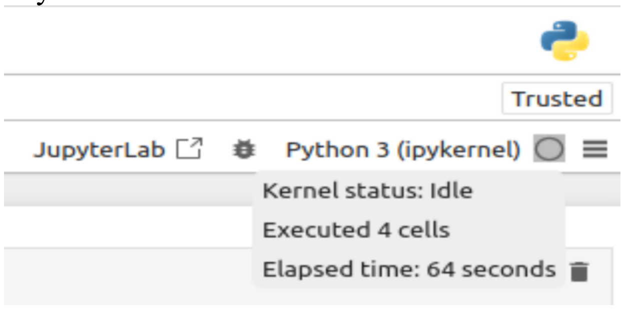
   a. Selecting **New** "Python 3 (ipykernel)"

   b. Open hyperlink [`https://github.com/tavwalt/CS643-AWS-ProgAssgn-2.git`](https://github.com/tavwalt/CS643-AWS-ProgAssgn-2.git)

   c. Open *"TavarisTrain.py"* or *"TavarisPredictions.py"* and **copy** code

   d. Paste it in the new `untitled`***.ipynb*** file you created and **RUN** it to see output

i. Pay attention to the *kernel status:idle* time



Kernel status: Idle
Executed 4 cells
Elapsed time: 64 seconds

You will see several things printed as the code reads the data, trains your model, makes the *predictions of wine quality for the Dataset* and finally outputs the F1 score.

The highest F1 score I observed (using the tavarisValidationDataset) for the predictions comes from the **Linear Regression model**

Partial Text OUTPUT:

Showing 5 rows for F1 Score for **Linear Regression Model #1**

```
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+-------+--
----------------------+-------------------------------+----------------------+
|fixed acidity|volatile acidity|citric acid|residual sugar|chlorides|free sulfur dioxide|total sulfur dioxide|density|
pH|sulphates|alcohol|quality|numerical_feature_vector|scaled_numerical_feature_vector|predicted_wine_quality|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+-------+--
----------------------+-------------------------------+----------------------+
|          8.9|            0.22|       0.48|           1.8|    0.077|               29.0|                60.0| 0.9968|3.39|     0.53|    9.4|      6|
[8.9,0.22,0.48,1....|       [0.46932693965105...|  5.44119060313995|
|          7.6|            0.39|       0.31|           2.3|    0.082|               23.0|                71.0| 0.9982|3.52|     0.65|    9.7|      5|
[7.6,0.39,0.31,2....|       [-0.3299662105669...| 5.347636338265752|
|          7.9|            0.43|       0.21|           1.6|    0.106|               10.0|                37.0| 0.9966|3.17|     0.91|    9.5|      5|
[7.9,0.43,0.21,1....|       [-0.1455139451319...| 5.793951545790781|
|          8.5|            0.49|       0.11|           2.3|    0.084|                9.0|                67.0| 0.9968|3.17|     0.53|    9.4|      5|
[8.5,0.49,0.11,2....|       [0.22339058573783...| 5.323474776695611|
|          6.9|             0.4|       0.14|           2.4|    0.085|               21.0|                40.0| 0.9968|3.43|     0.63|    9.7|      6|
[6.9,0.4,0.14,2.4...|       [-0.7603548299150...| 5.498296559618226|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+-------+--
----------------------+-------------------------------+----------------------+
only showing top 5 rows
```

Random Forest Model #2

```
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+-------+--
----------------------------+--------------------+--------------------+----------+
|fixed acidity|volatile acidity|citric acid|residual sugar|chlorides|free sulfur dioxide|total sulfur dioxide|density|
pH|sulphates|alcohol|quality|numerical_feature_vectorRandom|       rawPrediction|         probability|prediction|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+----+---------+-------+-------+--
----------------------------+--------------------+--------------------+----------+
|          7.4|             0.7|        0.0|           1.9|    0.076|                 11|                  34| 0.9978|3.51|     0.56|    9.4|      5|
[7.4,0.7,0.0,1.9,...|[0.0,0.0,0.0,0.45...|[0.0,0.0,0.0,0.00...|       5.0|
|          7.8|            0.88|        0.0|           2.6|    0.098|                 25|                  67| 0.9968| 3.2|     0.68|    9.8|      5|
[7.8,0.88,0.0,2.6...|[0.0,0.0,0.0,0.23...|[0.0,0.0,0.0,0.00...|       5.0|
|          7.8|            0.76|       0.04|           2.3|    0.092|                 15|                  54|  0.997|3.26|     0.65|    9.8|      5|
[7.8,0.76,0.04,2....|[0.0,0.0,0.0,0.26...|[0.0,0.0,0.0,0.00...|       5.0|
|        11.2|           0.28|       0.56|           1.9|    0.075|                17|                60|  0.998|3.16|     0.58|    9.8|      6|
[11.2,0.28,0.56,1...|[0.0,0.0,0.0,0.56...|[0.0,0.0,0.0,0.01...|       5.0|
|         7.4|            0.7|        0.0|           1.9|    0.076|                11|                34| 0.9978|3.51|     0.56|    9.4|      5|
[7.4,0.7,0.0,1.9,...|[0.0,0.0,0.0,0.45...|[0.0,0.0,0.0,0.00...|       5.0|
+------------+---------------+-----------+--------------+---------+------------------+--------------------+-------+----+---------+-------+-------+--
---------------------------+--------------------+--------------------+----------+
only showing top 5 rows
```

Prediction1 - F1 Score **Linear Regression** is more accurate

```
[Row(quality=6, predicted_wine_quality=5.44119060313995),
 Row(quality=5, predicted_wine_quality=5.347636338265752),
 Row(quality=5, predicted_wine_quality=5.793951545790781),
 Row(quality=5, predicted_wine_quality=5.323474776695611),
 Row(quality=6, predicted_wine_quality=5.498296559618226),
 Row(quality=5, predicted_wine_quality=5.3481426676556065),
 Row(quality=5, predicted_wine_quality=5.486218264921513),
 Row(quality=5, predicted_wine_quality=5.793951545790781),
 Row(quality=5, predicted_wine_quality=5.008904406332718),
 Row(quality=6, predicted_wine_quality=5.364021784970281),
 Row(quality=5, predicted_wine_quality=5.162782473716847),
 Row(quality=6, predicted_wine_quality=5.393723015290125),
 Row(quality=5, predicted_wine_quality=5.294634154178206),
 Row(quality=6, predicted_wine_quality=5.238249607904794),
 Row(quality=5, predicted_wine_quality=5.171470113929214),
 Row(quality=6, predicted_wine_quality=5.294234062849296),
 Row(quality=6, predicted_wine_quality=5.603236600007109),
 Row(quality=7, predicted_wine_quality=5.774372450204788),
 Row(quality=4, predicted_wine_quality=4.338242730996639),
Row(quality=5, predicted_wine_quality=5.8400597210465754
```

Prediction with Docker using Python:

A. Creating Docker Image for Prediction Application

1. Initialize docker on your ec2-instance using these steps.

2. Create a **new folder** and create **Docker file**.

```
$ mkdir myDocker
$ touch Dockerfile
```

3. Navigate to the folder where the Dockerfile is saved. This folder should also have several files:
   - Dockerfile (contains the instruction needed to make image)
   - dockerPredictions.py
   - tavarisTrainingDataset.csv
   - tavarisValidationDataset.csv
   - **requirements.txt** (edit this file to import dependencies outside of spark)
   - **java-19-openjdk-amd64** (library)
   - **spark-3.5.1-bin-hadoop3** (library)

```
$ cd /home/ubuntu/myDocker
```

4. Build this docker image with:

```
$ sudo docker build -t my-pyspark-app .
```

5. Verify the docker image has been created after it has been built

```
$ sudo docker images
```

6. Run this image build to test it:

```
$ sudo docker run my-pyspark-app
```

7. Login into Docker from terminal with
   - `$ sudo docker login   (enter your account info)`

8. Create a tag for your image and add your DockerHub username to it

```
$ sudo docker tag my-pyspark-app tavwalt2/my-pyspark-app:1.0.0
```

9. Verify the docker images again

```
$ sudo docker images
```

10. Push this image to Docker Hub with:

```
$ sudo docker push tavwalt2/my-pyspark-app:1.0.0
```

11. It can then be seen on Docker Hub via the link below
    - https://hub.docker.com/r/tavwalt2/my-pyspark-app
    - Use command to pull it **`$ docker pull tavwalt2/my-pyspark-app`**

FYI: If you run your code Spark-Code in Java using Maven use this!

When running as a **Stand-Alone** machine (meaning Master & Work as all-in-one)

```
$spark-submit --class CS643 --master spark://Internal-ip-address:7077
TavarisTrain.jar
```

When running in **Deploy-Mode Cluster** machine (Master with attached worker nodes)

```
$spark-submit --class CS643 --master spark://Internal-ip-address:7077 --
deploy-mode cluster TavarisTrain.jar
```