**Neel Patel – nap48 CS643**

**Project 2 Wine Quality Prediction - Readme**

## Code locations:

* Zip file submission on Canvas
* Github public repo:
* Dockerhub: (was not successful, the commands required to make a docker image are provided below)

# Learner Lab setup

1. A screenshot of a computer

   Description automatically generatedOpen learner Lab and click [Start Lab]
2. A screenshot of a computer

   Description automatically generatedOpen [AWS Details] and save the AWS CLI keys as well as the PEM/PPK file to ssh into EC2 instances

# EMR instance creation

* Find out what exact server are you on (In my case us-east-1e)
  + Open AWS Cloud Shell (terminal inside aws website)
  + Run the following command, it will list you whatever instances are available to you at that point.
    - aws ec2 describe-instance-type-offerings --location-type "availability-zone" --filters Name=location,Values=**us-east-1e** --region **us-east-1** --query "InstanceTypeOfferings[\*].[InstanceType]" --output text | sort
* Go to <https://us-east-1.console.aws.amazon.com/emr/home?region=us-east-1#/clusters>
* Create Cluster with following settings:
  + Name and Applications:
    - Ensure that the latest Amazon EMR release is selected
    - Spark
  + Cluster Configuration:
    - Primary, Core, and Task are all seet to c3.xlarge (worked for us-east-1e)
      * This instance type should be listed as part of your output from the earlier query. Might have to try different instances regardless.
  + Cluster Scaling and Provisioning:
    - Core = 1 ; Task = 3
  + Networking:
    - Open EC2 security groups firewall
    - Primary Node = ElasticMapReduce-Primary (or Create) (Might be called -Master later)
    - Core/Task nodes = ElasticMapReduce-Core (or Create) (Might be called -Slave or -Worker later)
  + Security configuration:
    - Amazon EC2 key pair: vockey
  + Identity and Access Management:
    - Service Role: EMR\_DefaultRole
    - Instance Profile: EMR\_EC2\_DefaultRole
* A screenshot of a computer

  Description automatically generatedThis will create total 5 special EC2 instances for the cluster, wait until Status for cluster says Waiting and Status Details says Ready to run steps to proceed.

# EMR Inbound port rules:

* Go to EC2 instances page --> Security Groups --> find the Master EMR rule
  + Add the following ports (22 and 4040) with the settings shown below:
  + A screenshot of a computer

    Description automatically generated

# S3 location

* Search for S3 in the AWS site --> Create S3 bucket --> provide name --> Upload files
* Upload the TrainingDataset.csv and ValidationDataset.csv
* As part of code execution, the models will be saved in this bucket in the models/ directory.
  + Subsequent executions will overwrite the models in s3 appropriately.

# Setting up EMR

* Open terminal and SSH into the Master node using ‘hadoop’ as the user and the PEM key
* Run the following commands:
  + *sudo yum update*
    - update all current packages
  + *pip install pyspark findspark boto3 numpy pandas scikit-learn datetime*
    - install relevant packages for program
  + *sudo yum install git -y*
    - install git to clone program
  + *git version*
    - check git version
  + *python -V*
    - check python version (if it doesn’t exist, install it)

# Developing on EMR instance (Visual Studio Code)

* Assuming Visual Studio Code is installed, Open the program and install the Remote-SSH extension.
* Click Bottom left Remote window button --> Connect to Host --> Add new SSH Host
  + Enter “ssh -i “<path to pem key>” hadoop@ec2<address>”
  + Edit the config file so the Path is accurate (Ctrl Shift P for menu)
* Once configured, open Remote Explorer Tab and connect to instance. You can now code and save files.

# Copying/Running completed project code to EMR

* Assuming you are in the ~ directory..
* Run the following commands:
  + git clone <https://github.com/neelapatel/CS643-WinePrediction>
  + spark-submit –master yarn CS643-WinePrediction/WineTraining.py
  + spark-submit –master yarn CS643-WinePrediction/WineTesting.py
* Note: WineTraining.py will create models on the S3 bucket that will need to be deleted on subsequent runs, therefore it is best to use your OWN S3 bucket with TrainingDataset.csv and ValidationDataset.csv

# Docker:

* *sudo service docker start*
* *cd CS643-WinePrediction*
* *touch Dockerfile*
* *nano Dockerfile* 
  + add the required syntax, dockerfile included on github
* *sudo docker login*
* *sudo docker build -t neelapatel/cs643-wineprediction .*
* testing:
  + *sudo docker run -it neelapatel/cs643-wineprediction*
* push:
  + *sudo docker push neelapatel/cs643-wineprediction*

# Code Execution Notes:

* The code runs assuming you are using the exact instructions documented in this readme.
* WineTraining and WineTesting.py will only use files stored in the neel-cs643 s3 bucket. If bucket does not exist, code will need to be modified with the appropriate changes, and MUST include the TrainingDataset.csv and ValidationDataset.csv