CS643 Programming Assignment 2

Name: Stephen Crane
Date: 8/7/2025

GitHub Link: https://github.com/stevethepirate88/CS643_Project2
DockerHub Link: https://hub.docker.com/repository/docker/stevethepirate/wineproject/general

Background

I have tried so many ways to set this up on AWS, and each one proved more futile than the last. The life I could have lived, the songs I could have written, the love I could have had in the time I spent trying to get this to run on AWS.

First I tried running this on AWS EMR. That sounds reasonable, right? It automatically builds the clusters for me, and I just need to feed it a .py file I've cleverly written. No problem, right? Sounds easy? Oh, what a fool I was. After many attempts to just initially deploy it, I finally found myself with a reliable system for getting a cluster running. Apparently our student accounts limit what I can do with IAMS so that was part of my problem there. But when I finally got it to the point where it could run a file? I was given the generic Error with exit code 13:

https://gist.github.com/stevethepirate88/ea6cd4e6dad85372b4cbcd5296313b92

What is exit code 13? It can be anything, really. Not enough memory, maybe? Unlikely. They were pretty beefy. Was it an error in my code? Likely, but the exact error or location of that error is EMR's alone to know. It is not knowledge meant for us mere mortals.

Here is a non-exhaustive list of links I used to try and troubleshoot my issues:

- Tutorial: Getting started with Amazon EMR
- Write a Spark Application
- Container exited with a non-zero exit code 13
- Amazon S3 Access Grants with Amazon EMR
- Run commands and scripts on an Amazon EMR Cluster
- Using Python libraries with EMR Serverless
- Executing a Python script on aws cloud for big data analytics
- Common errors when running jobs

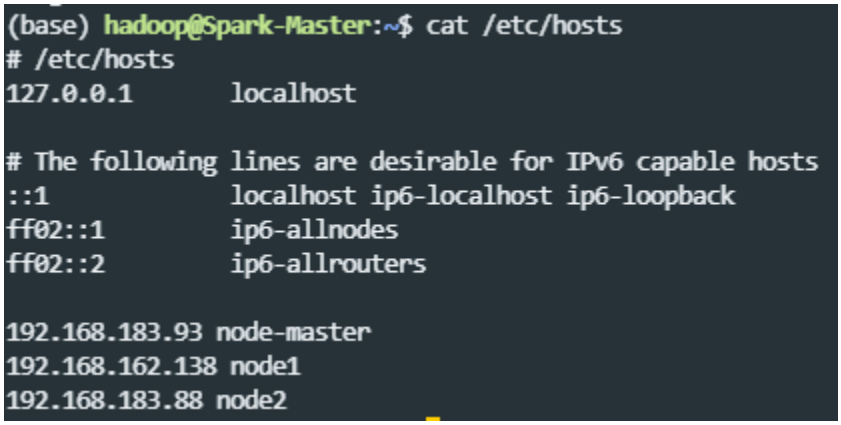
There is more. Bottom line, after many days of bashing my head against this, I decided you know what? Maybe EMR can just remain a mystery to me. I can do things the hard way. I can build my own Spark cluster. There was just one issue: I was fast running out of time, and trying

to figure out AWS's permissions and the ticking clock on the lab as well as any number of other issues that would develop… I had to find another way.

Thus, I fell back on cloud hosting knowledge I have spent the past 10 years dealing with: Linode/Akamai! I abandoned AWS for Linode for this project, and was able to get the cluster set up! And my program ran! Huzzah! Which now leads me to:

Setup

To set up my cluster I deployed 1 x 8GB Shared Linode as the Spark-Master. I then deployed 2x 24GB High Memory Linodes as Spark-Node1 and Spark-Node2 respectively. All 3 of those Linodes were assigned private IP addresses:

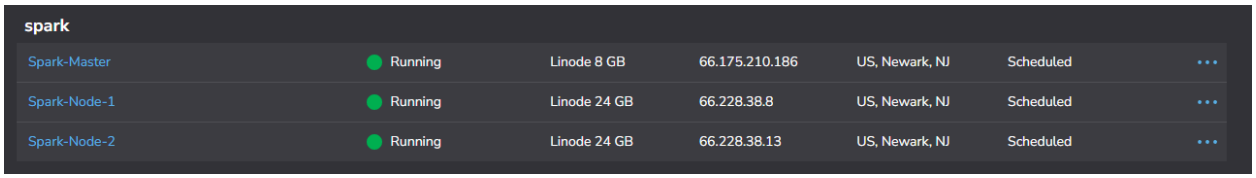


From there, I followed the following guides in order:

- Set up and secure a Linode
  - Linode provides direct, root access to servers, so setting up security is a ground-up matter
  - I logged in as root to each of the servers, secured SSH with a keypair, and installed fail2ban
  - I set individual hostnames on each Linode as well for easy tracking
  - I created a limited user, *hadoop* on each Linode as well and set it to passwordless sudo
- How to Install and Set Up a 3-Node Hadoop Cluster
  - This covers the basics of setting up Hadoop between 3 Linodes
  - The software installed and configured is Hadoop, HDFS, and YARN
  - The 8GB Shared Linode acted as the Master node in this cluster
- Running Spark on Top of a Hadoop YARN Cluster
  - This guide covers the installation of Spark as well as the configurations
  - Because I used relatively large Linodes, I didn't need to worry super much about directly configuring the memory allocation

- Introduction to PySpark
    - This covers the installation and configuration of Miniconda, Scala, and PySpark

With my 3 Linodes set up, I was ready to go:



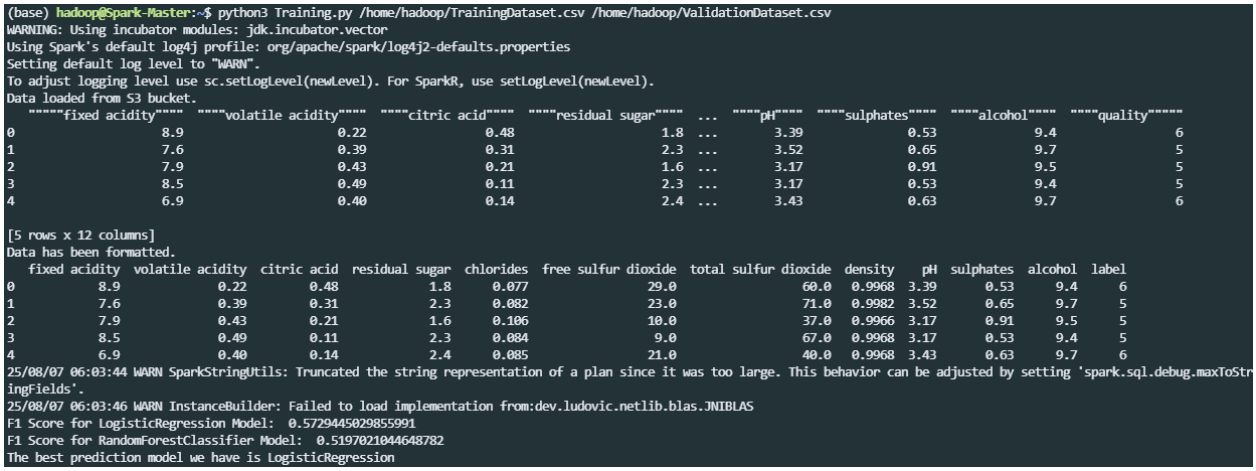
The Scripts

Training.py

Before running the script, it is important to run the following command to make sure there is an appropriate folder in place in hdfs:

hdfs dfs mkdir training

This will create the folder that training.py is going to put the model into.



The syntax for this is *python3 Training.py <path/to/trainingSet.csv> <path/to/validatorSet.csv>*

It tests two different models: LogisticRegression and RandomForestClassifier. It then creates the model out of the one with the highest F1 score and outputs the model to hdfs. In the above case, LogisticRegression had the higher F1 score.

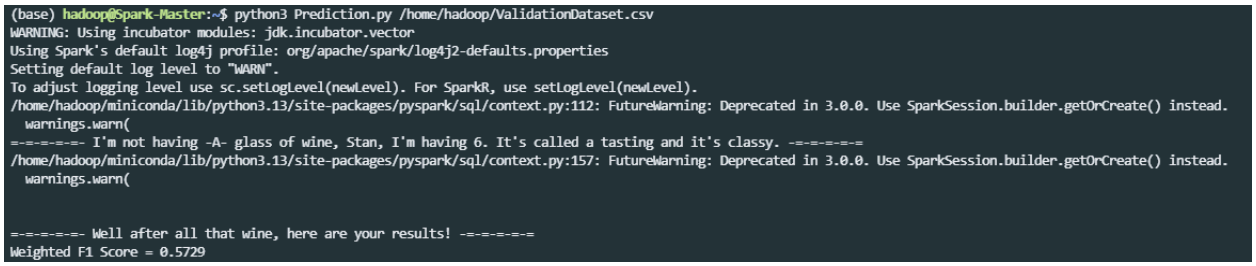
The paths to the two local files need to be absolute paths on the local machine. I've tried setting this up to use S3, but it's just too much for my poor little brain to handle.

Prediction.py without Docker

This script takes a variable and runs it against the existing WineModel in the hdfs.

The syntax is simple:

python3 Prediction.py <path/to/local/file.csv>



You should get an output similar to the above to let you know the weighted F1 score based on the model.

Prediction.py with Docker

To run the Prediction script with Docker, you just need to run the script with the following syntax:

sudo docker run -v <path/to/local/directory/with/files.csv>:/app/Validation stevethepirate/wineproject

If you have a .csv file in the local directory, the docker image will read it and let you know the weighted F1 score.