# CS643
## CLOUD COMPUTING

Name : Manohar
UCID: mk2224

TiTle: Wine Quality Predictions

Git Hub Link: https://github.com/Makmanohar/cloudPA2

Docker link: https://hub.docker.com/repository/docker/manohar0987/pa2docker/general

## Abstract:

Predicting wine quality is a common problem in the wine industry, where many factors affect wine quality, such as grape varieties, winemaking techniques, and environmental factors. In recent years, to solve this problem, machine learning methods have been applied to predict wine quality based on various characteristics.Python is a popular language for data analysis and machine learning, making it an excellent choice for predicting wine quality. In this project, the use Python and machine learning algorithms to predict the quality of wine based on its chemical properties.The dataset used in this project is the "Wine Quality" dataset, which contains the chemical properties and quality classifications of various red and white wines. The goal of this project is to create a machine learning model that is accurately predict the quality of wine based on its chemical characteristics.

## Training Setup:

To launch instances on the AWS Management Console, follow these steps:

1. Navigate to Services -> EC2 -> Launch Instances -> Launch Instances.
2. Enter 5 for the number of instances.
3. Select the "Ubuntu Server 20.04 LTS... ami-04505e74c0741db8d" Amazon Machine Image.
4. Choose the t2.large type, which is useful for Docker purposes, as it allows for fewer problems relating to running out of memory.

5. Create a new key pair and name it "ProgAssgn2". Download the key pair.
6. Under Network Settings -> Security groups (Firewall), check "Allow SSH traffic from [Anywhere 0.0.0.0/0]".
7. For Configure storage, configure it from 8 GiB to 16 GiB to install and configure modules/packages on the EC2 instances.
8. Keep all the rest of the default options and click "Launch instance".
9. View all instances. You will see a Pending status for the Instance State of the EC2 instances.

After your EC2 instance has started running, you can connect to it by running the following command in your terminal, replacing <YOUR_INSTANCE_PUBLIC_DNS> with the "Public IPv4 DNS" attribute of the EC2 instance:

$ ssh -i /Users/manoharkolla/Downloads/manu.pem
ubuntu@ec2-54-237-139-191.compute-1.amazonaws.com

Now, Perform getting the GitHub Repository

$ git clone https://github.com/Makmanohar/cloudPA2

Afet getting the information from the GitHub I am using bellow command

Parallal Implementation:

$ python3 /home/ubuntu/cloudPA2/training.py Validation Dataset.csv
TrainingDataset.csv

The above command is used for the Parallal implentation

Implementation without Docker:

We must install anaconda using cords forge command to run the Jupiter on the ssh

$ Jupiter notebook password
$jupyter notebook —no-browser

First we should create untitled ipynb in the first command line test.py which is given bellow:

#import findspark
#findspark.init()

```
#findspark.find()

#Loading the libraries
import pyspark
from pyspark.mllib.tree import RandomForest, RandomForestModel
from pyspark.mllib.util import MLUtils
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.mllib.regression import LabeledPoint
from pyspark.sql.functions import col
from pyspark.mllib.linalg import Vectors
from pyspark import SparkContext, SparkConf
from pyspark.sql.session import SparkSession
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.ml import Pipeline
import sys
import pandas as pd
import numpy as np
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score


#Starting the spark session
conf = pyspark.SparkConf().setAppName('winequality').setMaster('local')
sc = pyspark.SparkContext(conf=conf)
spark = SparkSession(sc)

path  = sys.argv[1]
#loading the validation dataset
val = spark.read.format("csv").load(path, header = True , sep=";")
val.printSchema()
val.show()


#changing the 'quality' column name to 'label'
for col_name in val.columns[1:-1]+["""quality"""]:
    val = val.withColumn(col_name, col(col_name).cast('float'))
val = val.withColumnRenamed("""quality""", "label")
```

```
#getting the features and label seperately and converting it to numpy array
features =np.array(val.select(val.columns[1:-1]).collect())
label = np.array(val.select('label').collect())

#creating the feature vector
VectorAssembler = VectorAssembler(inputCols = val.columns[1:-1] , outputCol =
'features')
df_tr = VectorAssembler.transform(val)
df_tr = df_tr.select(['features','label'])

#The following function creates the labeledpoint and parallelize it to convert it into RDD
def to_labeled_point(sc, features, labels, categorical=False):
    labeled_points = []
    for x, y in zip(features, labels):
        lp = LabeledPoint(y, x)
        labeled_points.append(lp)
    return sc.parallelize(labeled_points)

#rdd converted dataset
dataset = to_labeled_point(sc, features, label)

#loading the model from s3
RFModel = RandomForestModel.load(sc, "/winepredict/trainingmodel.model/")

print("model loaded successfully")
predictions = RFModel.predict(dataset.map(lambda x: x.features))

#getting a RDD of label and predictions
labelsAndPredictions = dataset.map(lambda lp: lp.label).zip(predictions)

labelsAndPredictions_df = labelsAndPredictions.toDF()
#cpnverting rdd ==> spark dataframe ==> pandas dataframe
labelpred = labelsAndPredictions.toDF(["label", "Prediction"])
labelpred.show()
labelpred_df = labelpred.toPandas()


#Calculating the F1score
F1score = f1_score(labelpred_df['label'], labelpred_df['Prediction'], average='micro')
print("F1- score: ", F1score)
```

```
print(confusion_matrix(labelpred_df['label'],labelpred_df['Prediction']))
print(classification_report(labelpred_df['label'],labelpred_df['Prediction']))
print("Accuracy" , accuracy_score(labelpred_df['label'], labelpred_df['Prediction']))

#calculating the test error
testErr = labelsAndPredictions.filter(
    lambda lp: lp[0] != lp[1]).count() / float(dataset.count())
print('Test Error = ' + str(testErr))
```

Implementation with Docker:

Creating Docker Image for Prediction Application
Initialize docker on your ec2-instance using these steps.

1.Add the user ubuntu to the ‘docker’ group
$ sudo usermod -aG docker $USER

2.Verify user ubuntu has been added to the docker group
$ groups

3.ubuntu adm dialout cdrom floppy sudo audio dip video plugdev netdev lxd docker
Login with the follow command:
$ docker login

4.Proceed to enter your credentials

5. Navigate to the folder where the Dockerfile is saved. This folder should also have the model, the Prediction.py application, and the dataset to utilize for prediction.
  $ cd /home/ubuntu/CS643-AWS-ProgAssgn-2/

6. Build this docker image with:
$ docker build -t cc362/aws-cs643-progassgn-2 .

7. Verify the docker image has been created after it has been built $ docker images

8. For running we run the bellow code

$ docker run -v /home/ubuntu/cloudPA2/:/data manohar0987/pa2docker:latest