1. Start the EMR cluster
   1. Create an EMR cluster with name “wineQualityTesting”
   2. In Applications, choose Spark
   3. Choose number of Instances as 6 (1 Master and 5 cores)
   4. Provide it with your SSH (.pem) key and roles to create the cluster.
   5. Wait till the cluster reaches in “Waiting” State
2. Create a S3 Bucket with name “wineQualityBucket” and add the files to the bucket.
3. Check the versions of the all the installed components.
   1. To provide EMR an access to the AWS CLI, open the SSH Role in the master and then execute the SSH command provided by the AWS
   2. Run the command, to download all the files in the EC2 locally

aws s3 cp s3://wineQualityBucket . --recursive

**Model Code (in Spark):**

1. Import and Initialize findSpark. Establish a connection to Spark using SparkContext
   1. import findspark

findspark.init()

from pyspark import SparkContext

conf = SparkConf().setAppName("Wine Quality Prediction").setMaster("local[4]")

sc = SparkContext(conf=conf)

1. Import pySpark and initialize Spark Session.
   1. import pyspark

from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()

1. To load the Training and Validation Data set file in Memory.read the file Training and Validation Data set, write the following commands.
   1. defTrain = spark.read.format('csv').options(header='true', inferSchema = 'true', delimiter = ';').csv("content/sample\_data/TrainingDataset.csv")
   2. defTest = spark.read.format('csv').options(header='true', inferSchema = 'true', delimiter = ';').csv("content/sample\_data/ ValidationDataset.csv")
2. To create a Vector of features to predict the values execute the following Lines

featureColumns = [ col for col in defTrain.columns if (col != '""""quality"""""')]

from pyspark.mllib.linalg import Vectors

from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(inputCols = featureColumns, outputCol='features')

dataDF = assembler.transform(defTrain)

dataDF.printSchema()

1. Split the Training Dataset into 70:30 datasets. It checks the Root mean square error

splitValue = 0.7

trainingDF, testDF = defTrain.randomSplit([splitValue, 1 - splitValue])

1. To perform RandomForest Classification on the given data and develop the model.  
   Import regression module from ml.regression and provide the constructor with parameters.

It calibrates the model for maximum accuracy by checking its RMS error

from pyspark.ml.evaluation import RegressionEvaluator

from pyspark.ml.classification import RandomForestClassifier

from pyspark.ml import Pipeline

rf = RandomForestClassifier(**featuresCol='features', labelCol='""""quality"""""', numTrees=150, maxBins=484, maxDepth=29, minInstancesPerNode=10, seed=34**)

rfPipeline = Pipeline(stages=[assembler, rf])

rfPipelineModel = rfPipeline.fit(trainingDF)

evaluator = RegressionEvaluator(labelCol='""""quality"""""', predictionCol="prediction", metricName="rmse")

rfTrainingPredictions = rfPipelineModel.transform(defTrain)

rfTestPredictions = rfPipelineModel.transform(testDF)

print("Random Forest RMSE on traning data = %g" % evaluator.evaluate(rfTrainingPredictions))

print("Random Forest RMSE on test data = %g" % evaluator.evaluate(rfTestPredictions))

1. To perform metrics evaluation, Import MulticlassMetrics module from ml.evaluation

It calculates the accuracy of the code, measure the F1 score based on the labels.

Provide the Test data file to the model, on which accuracy can be measured.

cm = metrics.confusionMatrix().toArray()

print(cm)

print("(Parameter, Precision, Recall, Accuracy, F1 Score)")

for i in range(3, 9):

if metrics.precision(i) != 0:

print(i, ", ", round(metrics.precision(i), 2), ", ", round(metrics.recall(i), 2), ", ", round(metrics.accuracy, 2), ", ", round(metrics.fMeasure(float(i), 1.0), 2))