**//Complete program of ALS prediction with images in dataframe scala – spark-shell**

import org.apache.spark.ml.evaluation.RegressionEvaluator

import org.apache.spark.ml.recommendation.ALS

import org.apache.spark.mllib.recommendation.Rating

//Created a case class of scala

case class Rating(userId: Int, movieId: Int, rating: Float, timestamp: Long)

//created a function which return case class object

def parseRating(str: String): Rating = {

val fields = str.split(",")

assert(fields.size == 4)

Rating(fields(0).toInt, fields(1).toInt, fields(2).toFloat, fields(3).toLong)

}

//Created RDD through SparkContext

val ratrawdata = sc.textFile("/user/Test/ratings.csv")

//Filtered first data to apply parse function

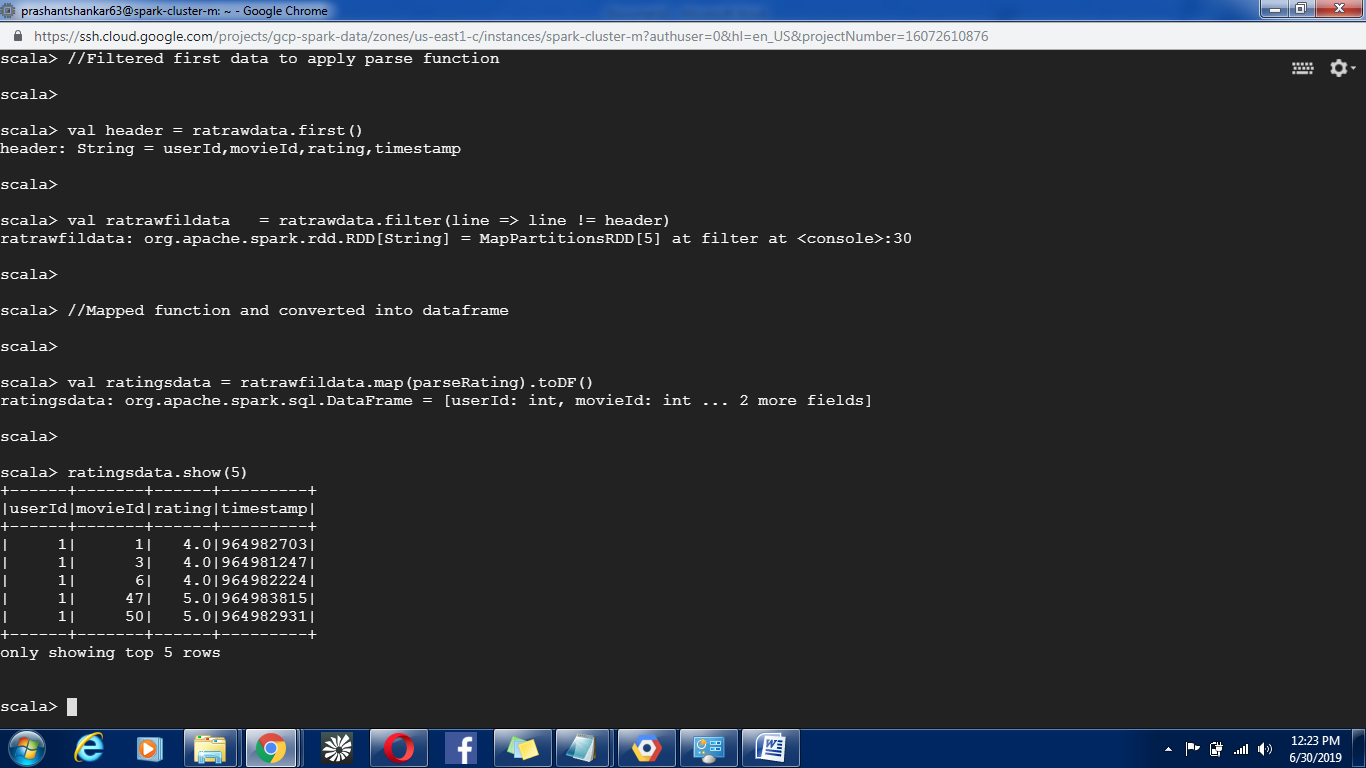
val header = ratrawdata.first()

val ratrawfildata = ratrawdata.filter(line => line != header)

//Mapped function and converted into dataframe

val ratingsdata = ratrawfildata.map(parseRating).toDF()

**PIC 1: ratingsdata.show(5)**

****

//Splitting data into Training and Testing data set

val Array(training, test) = ratingsdata.randomSplit(Array(0.8, 0.2))

//Build the recommendation model using ALS on the training data

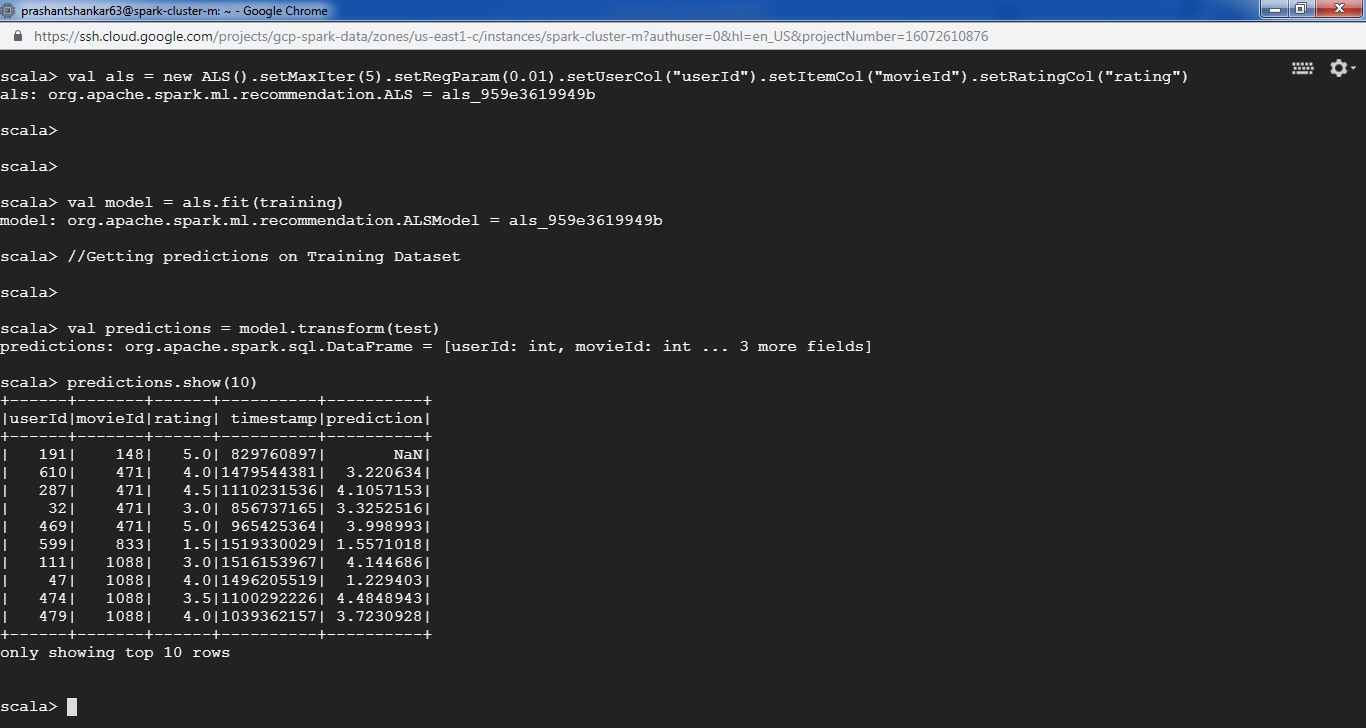
val als = new ALS().setMaxIter(5).setRegParam(0.01).setUserCol("userId").setItemCol("movieId").setRatingCol("rating")

val model = als.fit(training)

//Getting predictions on Training Dataset

val predictions = model.transform(test)

predictions.show(10)

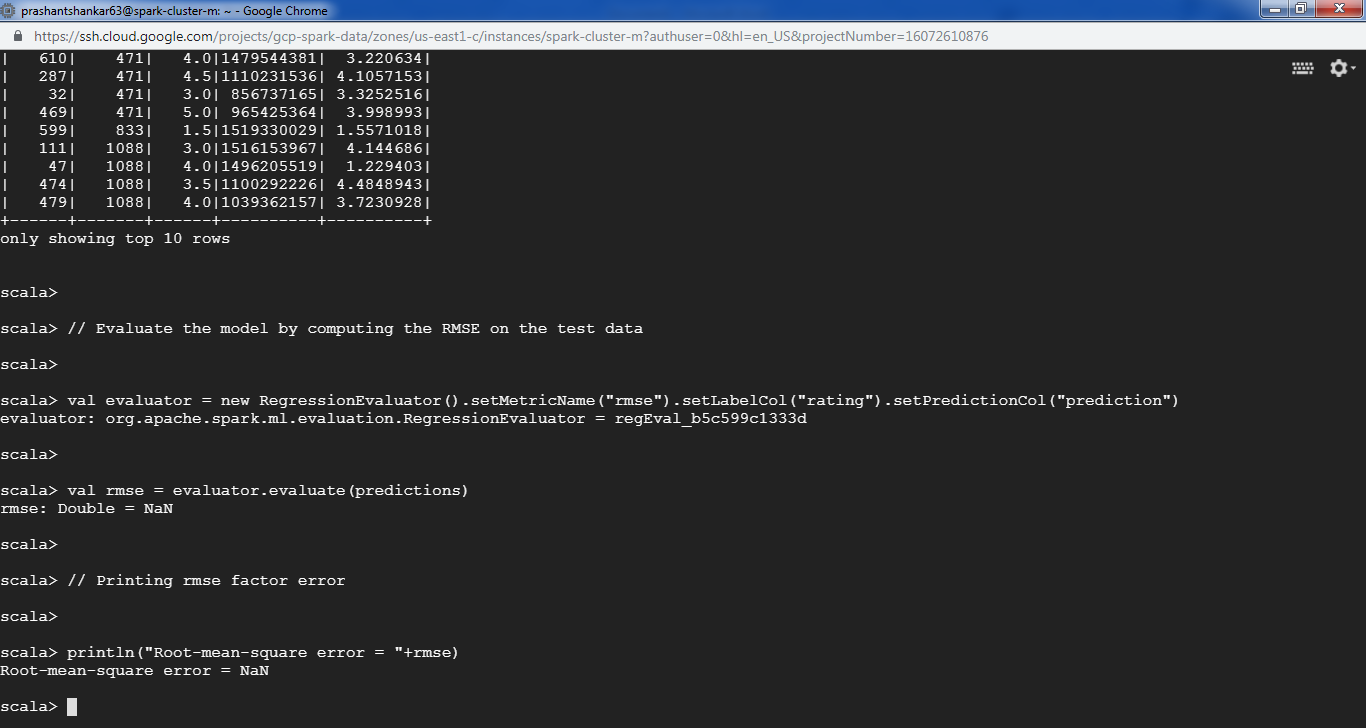


// Evaluate the model by computing the RMSE on the test data

val evaluator = new RegressionEvaluator().setMetricName("rmse").setLabelCol("rating").setPredictionCol("prediction")

val rmse = evaluator.evaluate(predictions)

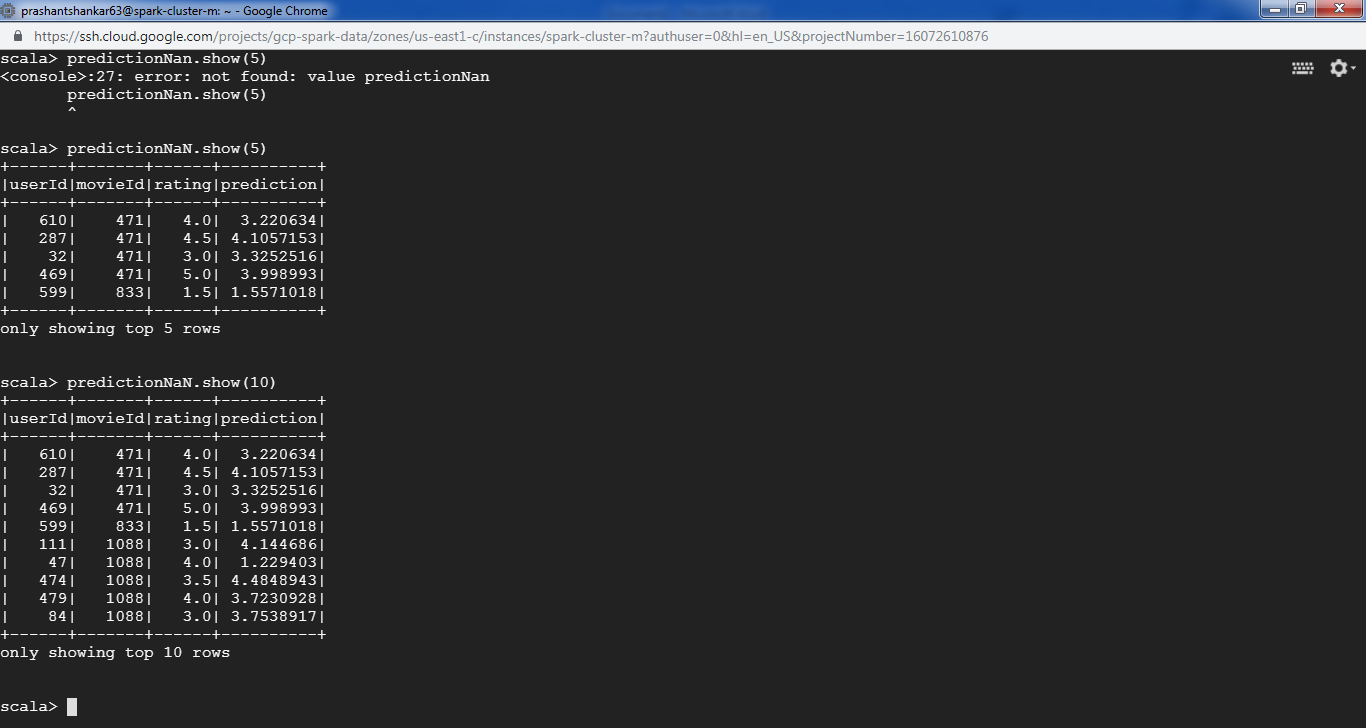
// Printing rmse factor error

println("Root-mean-square error = "+rmse)

//Trying to filter the data without rmse error

val predictionNaN = predictions.select("userId","movieId","rating","prediction").filter("prediction != 'NaN'")

predictionNaN.show(10)

****

+------+-------+------+----------+ |userId|movieId|rating|prediction|+------+-------+------+----------+| 610| 471| 4.0| 3.220634|| 287| 471| 4.5| 4.1057153|| 32| 471| 3.0| 3.3252516|| 469| 471| 5.0| 3.998993|| 599| 833| 1.5| 1.5571018|| 111| 1088| 3.0| 4.144686|| 47| 1088| 4.0| 1.229403|| 474| 1088| 3.5| 4.4848943|| 479| 1088| 4.0| 3.7230928|| 84| 1088| 3.0| 3.7538917|+------+-------+------+----