**Linear Regression: Spark & Scala**

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

import org.apache.spark.ml.regression.LinearRegression

import org.apache.spark.ml.tuning.{ParamGridBuilder, TrainValidationSplit}

import org.apache.log4j.\_

import org.apache.spark.ml.feature.VectorAssembler

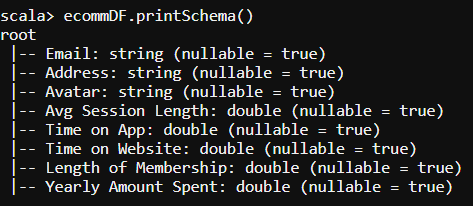
import org.apache.spark.mllib.linalg.Vectors

Logger.getLogger("org").setLevel(Level.ERROR)

val data = spark.read.option("header","true").option("inferSchema","true").format("csv").load("/user/edureka\_292003/EcommerceCustomers.csv")

val ecommDF = data

ecommDF.printSchema()



val ecommDF1 = ecommDF.select(ecommDF("Yearly Amount Spent").as("label"),$"Avg Session Length",$"Time on App",$"Time on Website",$"Length of Membership")

val assembler = new VectorAssembler().setInputCols(Array("Avg Session Length","Time on App","Time on Website","Length of Membership")).setOutputCol("features")

val ecommDF2 = assembler.transform(ecommDF1).select($"label",$"features")

val lr = new LinearRegression()

val lrModel = lr.fit(ecommDF2)

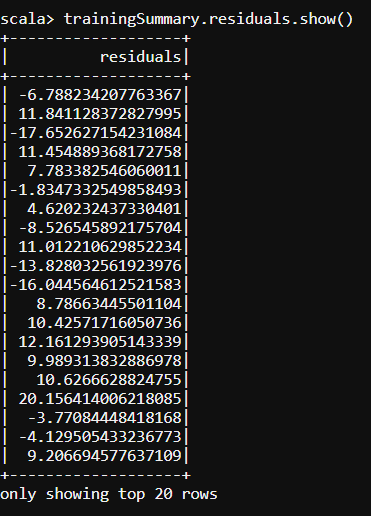
println(s"Coefficients: ${lrModel.coefficients} Intercept: ${lrModel.intercept}")

val trainingSummary = lrModel.summary

println(s"numIterations: ${trainingSummary.totalIterations}")

println(s"objectiveHistory: ${trainingSummary.objectiveHistory.toList}")

trainingSummary.residuals.show()



println(s"RMSE: ${trainingSummary.rootMeanSquaredError}")

println(s"MSE: ${trainingSummary.meanSquaredError}")

println(s"r2: ${trainingSummary.r2}")

