**Logistic Regression: Spark & Scala**

// Import the needed libraries

import org.apache.spark.mllib.evaluation.MulticlassMetrics

import org.apache.spark.mllib.classification.{LogisticRegressionWithLBFGS, LogisticRegressionModel}

import org.apache.spark.mllib.regression.LabeledPoint

import org.apache.spark.mllib.linalg.{Vector, Vectors}

// Transform each qualitative data in the data set into a double numeric value

def getDoubleValue( input:String ) : Double = {

var result:Double = 0.0

if (input == "P") result = 3.0

if (input == "A") result = 2.0

if (input == "N") result = 1.0

if (input == "NB") result = 1.0

if (input == "B") result = 0.0

return result

}

// Read data into memory in - lazy loading

val data = sc.textFile("hdfs://nameservice1/user/edureka\_292003/Qualitative\_Bankruptcy.txt")

data.count()

// Prepare data for the logistic regression algorithm

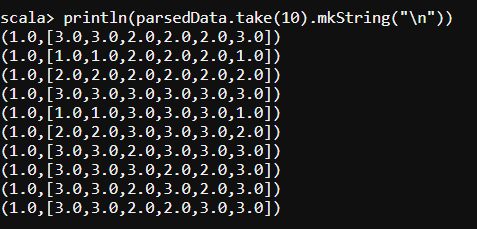
val parsedData = data.map{line =>

val parts = line.split(",")

LabeledPoint(getDoubleValue(parts(6)), Vectors.dense(parts.slice(0,6).map(x => getDoubleValue(x))))

}

println(parsedData.take(10).mkString("\n"))



// Split data into training (60%) and test (40%)

val splits = parsedData.randomSplit(Array(0.6, 0.4), seed = 11L)

val trainingData = splits(0)

val testData = splits(1)

// Train the model

val model = new LogisticRegressionWithLBFGS().setNumClasses(2).run(trainingData)

// Evaluate model on training examples and compute training error

val labelAndPreds = testData.map { point =>

val prediction = model.predict(point.features)

(point.label, prediction)

}

val trainErr = labelAndPreds.filter(r => r.\_1 != r.\_2).count.toDouble / testData.count

println("Training Error = " + trainErr)

