***LOAN DEFAULT PREDICTION***

**1.PROBLEM STATEMENT:**

*-> To build a logistic regression model in a big data environment for a bank.*

*-> In order to predict,whether the client will default on a loan or not.*

**2.ABSTRACT:**

*-> Banks provide loan to clients in exchange for the promise of repayment.*

*-> Some might default on the loans; unable to repay them due to some reason.*

*-> The bank maintains insurance to reduce their risk of loss in the event of default*

*-> The insured amount may cover all or just some part of the loan amt.*

*For this assignment, the bank wants to predict which client will default on their loans*

*based on their financial information.*

**3.IMPORTANT FEATURES:**

*->TARGET*

*->NAME\_CONTRACT\_TYPE*

*->FLAG\_OWN\_CAR*

*->FLAG\_OWN\_REALTY*

*->AMT\_INCOME\_TOTAL*

*->AMT\_CREDIT*

*->AMT\_ANNUITY*

*->NAME\_INCOME\_TYPE*

*->FLAG\_MOBILE*

*->FLAG\_CONT\_MOBILE*

*->CNT\_FAM\_MEMBERS*

*->REGION\_RATING\_CLIENT*

*->REG\_REGION\_NOT\_LIVE\_REGION*

*->REG\_REGION\_NOT\_WORK\_REGION*

*->total\_doc*

*->Safe\_clients*

**4.UNWANTED FEATURES:**

*->CODE\_GENDER*

*->CNT\_CHILDREN*

*->NAME\_EDUCATION\_TYPE*

*->NAME\_FAMILY\_STATUS*

*->NAME\_HOUSING\_TYPE*

*->DAYS\_BIRTH*

*->DAYS\_EMPLOYED*

*->DAYS\_EMPLOYED*

*->FLAG\_EMP\_PHONE*

*->FLAG\_WORK\_PHONE*

*->FLAG\_PHONE*

*->OCCUPATION\_TYPE*

*->REGION\_RATING\_CLIENT\_W\_CITY*

*->ORGANIZATION\_TYPE*

**5.PRE-PROCESSING:**

**NOTE:**  In this pre-processing step , we had just combined 10 columns into a 2 main important features (total\_doc , Safe\_clients).

*-> STEP 1: DROPPED UNWANTED COLUMNS*

*-> STEP 2: FEATURE EXTRACTION.*

*-> STEP 3: REMOVED NULL VALUES.*

*-> STEP 4: REMOVED DUPLICATES.*

*-> STEP 5: LABEL ENCODING.*

*6.BUILDING LOGISTIC REGRESSION MODEL*

*STEPS:*

**-> Import the needed libraries**

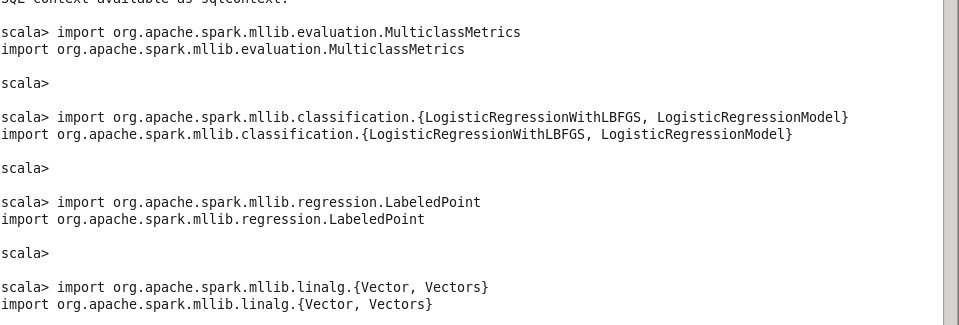
*import org.apache.spark.mllib.evaluation.MulticlassMetrics*

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

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

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

**IMPLEMENTATION OUTPUT:**



**// 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 == "0") result = 0.0*

*if (input == "1") result = 1.0*

*if (input == "2") result = 2.0*

*if (input == "3") result = 3.0*

*if (input == "4") result = 4.0*

*if (input == "5") result = 5.0*

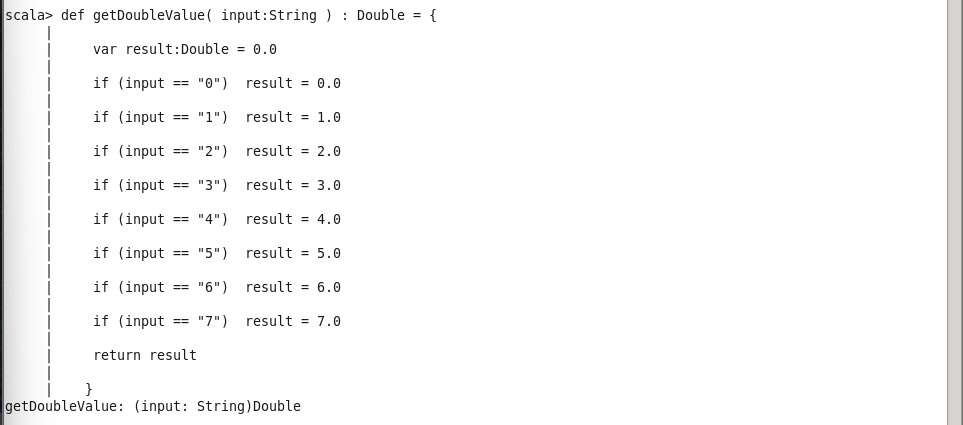
*if (input == "6") result = 6.0*

*if (input == "7") result = 7.0*

*return result*

*}*

**IMPLEMENTATION:**



**// Read data into memory in - lazy loading**

*val data = sc.textFile("/user/cloudera/loan\_application.csv")*

*data.count() //*

*\*

**// Prepare data for the logistic regression algorithmquit**

*val parsedData = data.map{line =>*

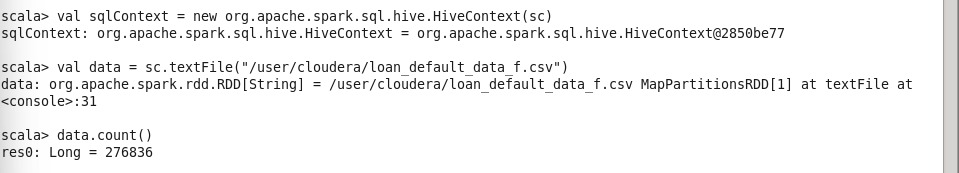
*val parts = line.split(",")*

*LabeledPoint(getDoubleValue(parts(25)), Vectors.dense(parts.slice(0,25).map(x =>getDoubleValue(x))))*

*}*

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

**IMPLEMENTATION:**



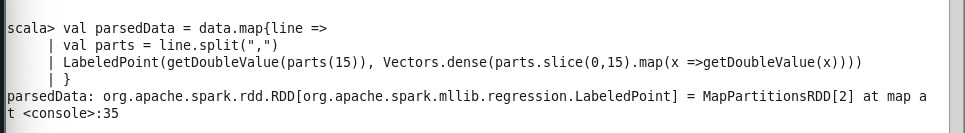
**// Split data into training (80%) and test (20%)**

*val splits = parsedData.randomSplit(Array(0.8, 0.2), seed = 11L)*

*val trainingData = splits(0)*

*val testData = splits(1)*

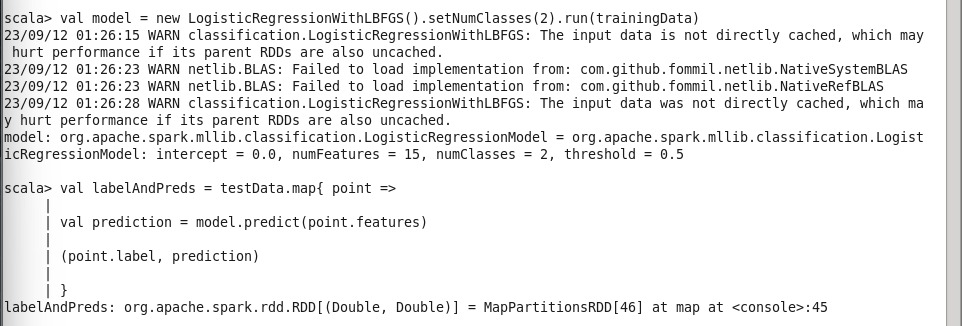
**IMPLEMENTATION:**



**// Train the model**

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

**IMPLEMENATION:**



**// Evaluate model on training examples and compute training error**

*val labelAndPreds = testData.map{ point =>*

*val prediction = model.predict(point.features)*

*(point.label, prediction)*

*}*

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

*println("Test Error = " + testErr)*

**IMPLEMENTATION:**

