Tecnológico Nacional de México

INSTITUTO TECNOLÓGICO CAMPUS TIJUANA

ING. EN SISTEMAS COMPUTACIONALES

Subdirección Académica

Departamento de Sistemas y Computación

BDD-1704 TI9A - 6:00pm-7:00pm

ASIGNATURA:

Datos Masivos

SEMESTRE:

Septiembre- Enero 2020

Tarea:

Práctica#2 Análisis de código:

Regresión Logística

MAESTRO:

JOSE CHRISTIAN ROMERO HERNANDEZ

23/11/2020

// Importar librería con una sesión Spark con la librería de Regresión logística

import org.apache.spark.ml.classification.LogisticRegression

import org.apache.spark.sql.SparkSession

// Utilizar el código de reporte de errores

import org.apache.log4j.\_

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

// Crear una sesión Spark

val spark = SparkSession.builder().getOrCreate()

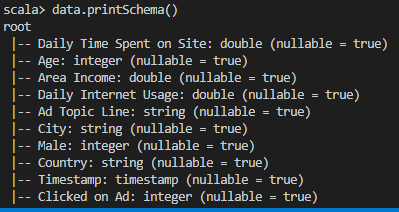
// Utilizar el objeto Spark para leer el archivo csv “Advertising”

val data = spark.read.option("header","true").option("inferSchema", "true").format("csv").load("C:/Users/DELL/Desktop/BigData-master/Spark\_LogisticRegression/advertising.csv")

// Imprimir el esquema del Dataframe

data.printSchema()

// Resultado



///////////////////////

/// Despliegue los datos /////

/////////////////////

//imprimir una linea de ejemplo

data.head(1)

//Resultado



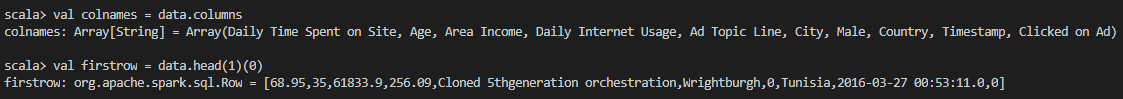
//Variable que contiene los datos de las columnas

val colnames = data.columns

//Variable que contiene primer renglón

val firstrow = data.head(1)(0)

//Resultado



/\*Ciclo para imprimir datos del

primer renglón en relación a los nombres asignados en las columnas\*/

println("\n")

println("Example data row")

for(ind <- Range(1, colnames.length)){

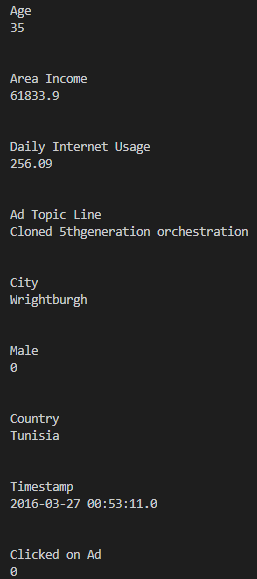
println(colnames(ind))

println(firstrow(ind))

println("\n")

}

//Resultado



////////////////////////////////////////////////////

//// Preparar el DataFrame para Machine Learning ////

//////////////////////////////////////////////////

// Crear una nueva columna llamada "Hour" desde Timestamp que contiene "Hour of the click"

val timedata = data.withColumn("Hour",hour(data("Timestamp")))

// Renombrar la columna "Clicked on Ad" a "label"

// Toma las siguientes columnas como features "Daily Time Spent on Site", "Age", "Area Income", "Daily Internet Usage", "Timestamp", "Male"

val logregdata = timedata.select(data("Clicked on Ad").as("label"), $"Daily Time Spent on Site", $"Age", $"Area Income", $"Daily Internet Usage", $"Hour", $"Male")

// Importar librerias VectorAssembler y Vectors

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

import org.apache.spark.ml.linalg.Vectors

// Crear un nuevo objeto VectorAssembler llamado “assembler” para features

val assembler = (new VectorAssembler() .setInputCols(Array("Daily Time Spent on Site", "Age","Area Income","Daily Internet Usage","Hour","Male")).setOutputCol("features"))

// Utilizar un randomSplit para crear 70/30 split test y train data

val Array(training, test) = logregdata.randomSplit(Array(0.7, 0.3), seed = 12345)

///////////////////////////////

// Configure un Pipeline ///////

/////////////////////////////

// Importar librería Pipeline

import org.apache.spark.ml.Pipeline

// Crear un objeto LogisticRegression llamado “lr”

val lr = new LogisticRegression()

// Crear un nuevo pipeline con los elementos: assembler, lr

val pipeline = new Pipeline().setStages(Array(assembler, lr))

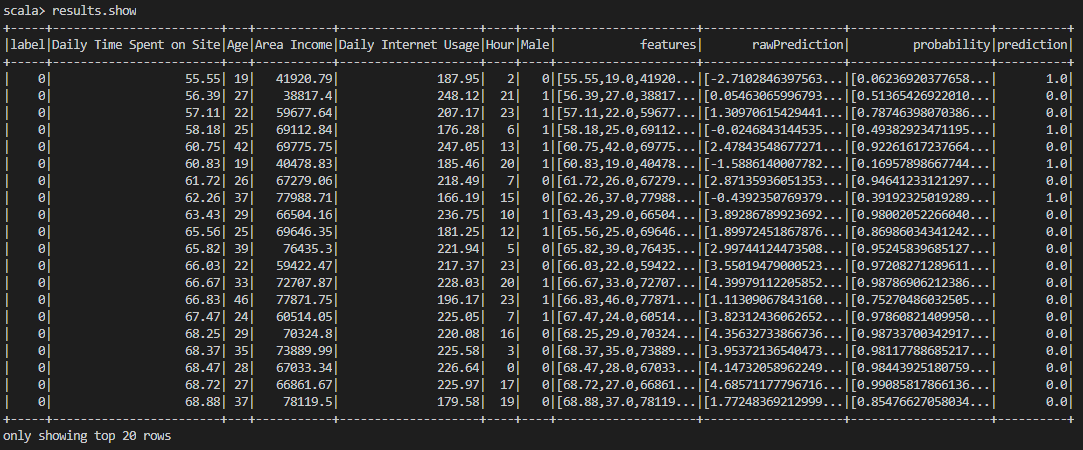
// Ajustar el pipeline para el conjunto training.

val model = pipeline.fit(training)

// Tomar los resultados en el conjunto Test utilizando transform

val results = model.transform(test)

// Resultado



////////////////////////////////////

//// Evaluacion del modelo /////////////

//////////////////////////////////

// Para métricas y evaluación importar la librería MulticlassMetrics

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

// Convertir los resultados del test a RDD utilizando .as y .rdd

val predictionAndLabels = results.select($"prediction",$"label").as[(Double, Double)].rdd

// Inicializar un objeto MulticlassMetrics

val metrics = new MulticlassMetrics(predictionAndLabels)

// Imprimir la matriz Confusion y metricas

println("Confusion matrix:")

println(metrics.confusionMatrix)

metrics.accuracy

// Resultado

