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#1 RegresionLineal

MAESTRO:

JOSE CHRISTIAN ROMERO HERNANDEZ

23/11/2020

// Importar librerías

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

import org.apache.spark.sql.SparkSession

import spark.implicits.\_

// Opcional: Utilice el siguiente código para configurar errores

import org.apache.log4j.\_

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

// Inicie una simple Sesión Spark

val spark = SparkSession.builder().appName("Practice1").master("local")getOrCreate()

// Utilice Spark para el archivo csv Clean-Ecommerce.

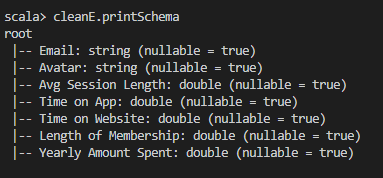
val path = "C:/Users/DELL/Desktop/BigData-master/Spark\_LogisticRegression/Clean-Ecommerce.csv"

// Imprima el schema en el DataFrame

val cleanE = spark.read.option("header", "true").option("inferSchema","true")csv(path)

clean.printSchema

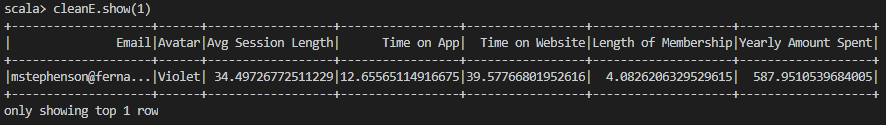
//Resultado



//Imprime un renglón del ejemplo del DataFrame

cleanE.show(1)

//Resultado



**Configuración del DataFrame para Machine Learning:**

// Importe VectorAssembler y Vectors

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

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

**Transforme el data frame para que tome la forma de ("label","features")**

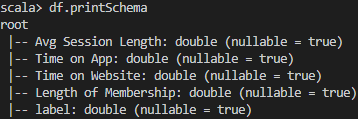
**Pasos**

// 1) Renombre la columna Yearly Amount Spent como "label". También de los datos // tome solo la columna numérica,

// Deje todo esto como un nuevo DataFrame que se llame df

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

//Resultado



// 2) Que el objeto assembler convierta los valores de entrada a un vector

// Utilice el objeto VectorAssembler para convertir la columnas de entradas del df

// a una sola columna de salida de un arreglo llamado "features"

// Configure las columnas de entrada de donde se supone que leemos los valores.

// Llamar a esto nuevo assambler.

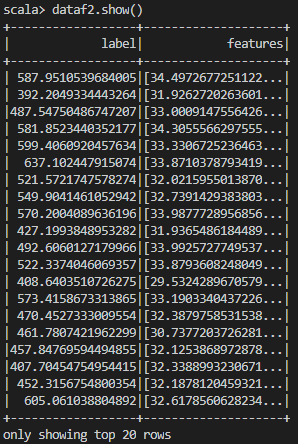
val assembler = new VectorAssembler().setInputCols(Array("Avg Session Length","Time on App","Time on Website","Length of Membership")).setOutputCol("features")

// 3) Utilice el assembler para transformar nuestro DataFrame a dos columnas: label and features

val dataf2 = assembler.transform(df).select($"label", $"features")

dataf2.show()

//Resultado



**Regresión Lineal**

// Crear un objeto para modelo de regresión lineal.

val lr = new LinearRegression()

// Ajuste el modelo para los datos y llame a este modelo lrModelo

val lrModel = lr.fit(output)

// Resuma el modelo sobre el conjunto de entrenamiento imprima la salida de algunas metricas

// Utilize metodo .summary de nuestro modelo para crear un objeto

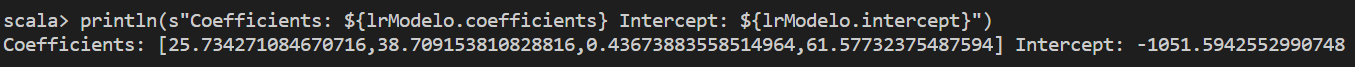
// llamado trainingSummary

val trainingSummary = lrModel.summary

// Imprima the coefficients y intercept para la regresión lineal

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

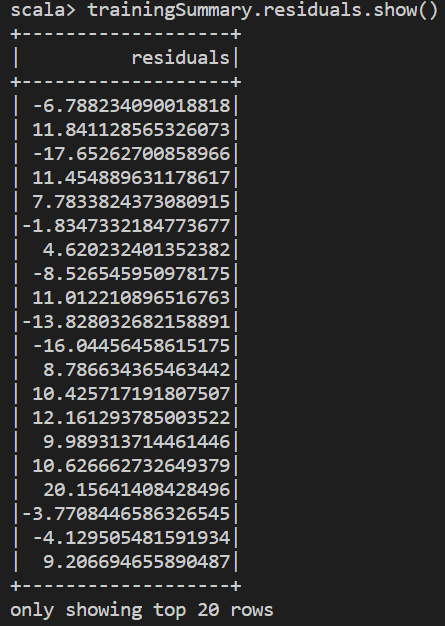
//Resultado



// Muestre los valores de residuales, el RMSE, el MSE, y R^2.

trainingSummary.residuals.show()

//Resultado



//Valores RMSE

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

//Resultado



//Valor R^2.

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

//Resultado