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:

Explicación de lo que yo entendí de los códigos:

CorrelationExample,

ChiSquareTestExample y

SummarizerExample

MAESTRO:

JOSE CHRISTIAN ROMERO HERNANDEZ

24/11/2020

***CorrelationExample.scala***

//importamos las librerias de spark matrix, vectors, correlation y row.

import org.apache.spark.ml.linalg.{Matrix, Vectors}

import org.apache.spark.ml.stat.Correlation

import org.apache.spark.sql.Row

//importamos la session de spark

import org.apache.spark.sql.SparkSession

//creamos un objeto llamado CorrelationExample el cual contendra todos los procedimientos para obtener la correlacion

object CorrelationExample{

//definimos un main con un arreglo de tipo string el cual contendra la variabe spark y la sesion anteriormente importada

def main(args: Array[String]): Unit = {

//creamos una variable inmutable llamada spark que contendra la sesion de spark y le daremos el nombre de CorrelationExample

val spark = SparkSession.builder.appName("CorrelationExample").getOrCreate()

//importamos la varieble spark impicitamente

import spark.implicits

//creamos otra variable llamada date el cual va a contener los datos en forma de vectores con los cuales vamos a obtener la correlacion

val data = Seq(

Vectors.sparse(4, Seq((0, 1.0), (3, -2.0))),

Vectors.dense(4.0, 5.0, 0.0, 3.0),

Vectors.dense(6.0, 7.0, 0.0, 8.0),

Vectors.sparse(4, Seq((0, 9.0), (3, 1.0)))

)

// se crea un dataframe al cual se le asigna el valor de una tupla llamada tupla1

// el dataframe contiene una columna llamada features

val df = data.map(Tuple1.apply).toDF("features")

//a un valor de tipo Fila llamado coeficiente1 de una matrix se le asigna el valor de la correlacion de pearson aplicada en el dataframe

//aplicado a la columna features

val Row(coeff1: Matrix) = Correlation.corr(df, "features").head

//a un valor de tipo Fila llamado coeficiente2 de una matrix se le asigna el valor de la correlacion de pearson aplicada en el dataframe

//aplicado a la columna features

val Row(coeff2: Matrix) = Correlation.corr(df, "features", "spearman").head

println(s"Spearman correlation matrix:\n $coeff2")// se imprime

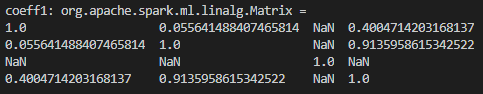
//se detiene la ejecucion

spark.stop()

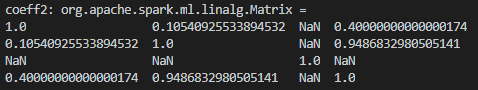
}

}

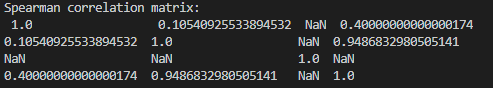
//Resultado coeficiente 1



//Resultado coeficiente 2



//Resultado correlación de matrix spearman



***ChiSquareTestExample.scala***

//cargamos las librerias vector, vectors y ChiSquareTest

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

import org.apache.spark.ml.stat.ChiSquareTest

//importamos la session de spark

import org.apache.spark.sql.SparkSession

//creamos un objeto llamado ChiSquareTestExample el cual contendra todos los procedimientos para obtener los valores de ChiSquare

object ChiSquareTestExample{

//se crea un main que contendra un arreglo de tipo string y la variable spark

def main(args: Array[String]): Unit = {

/\*se crea la variable spark cargandole la sesion de spark anteriormente importada y la asignaremos un nombre el cual es

ChiSquareTestExample\*/

val spark = SparkSession.builder.appName("ChiSquareTestExample").getOrCreate()

//importamos la variable spark implicitamente

import spark.implicits

//creamos una variable lamda date la cual va a tener los datos que utilizaremos en el programa en forma de vectores

val data = Seq(

(0.0, Vectors.dense(0.5,10.0)),

(0.0, Vectors.dense(1.5,20.0)),

(1.0, Vectors.dense(1.5,30.0)),

(0.0, Vectors.dense(3.5,30.0)),

(0.0, Vectors.dense(3.5,40.0)),

(1.0, Vectors.dense(3.5,40.0))

)

// se crea un dataframe al cual se le asigna el valor date en las columnas label y features

val df = data.toDF("label","features")

//se crea un valor chi al que se le aplica chisquare mediantes las librerias al dataframe

val chi = ChiSquareTest.test(df, "features", "label").head

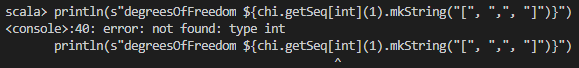
println(s"pValues = ${chi.getAs[Vector](0)}")

//Resultado



println(s"degreesOfFreedom ${chi.getSeq[Int](1).mkString("[", ",", "]")}")

//Resultado con error y corrección



nota: aqui me equivoque en el int, tiene que ser Int en vez de int



println(s"statistics ${chi.getAs[Vector](2)}")// se imprime

//Resultado



spark.stop()

}

}

***SummarizerExample.scala***

//importamos las librerias vector, vectors y summarizer

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

import org.apache.spark.ml.stat.Summarizer

//importamos la session de spark

import org.apache.spark.sql.SparkSession

//creamos un objeto llamado SummarizerExample el cual contendra todos los los procedimientos

object SummarizerExample{

//creamos un main que contendra un arreglo de tipo entero y contendra este la variable saprk

def main(args: Array[String]): Unit = {

//creamos la variable spark, le cargamos la sesion.

val spark = SparkSession.builder.appName("SummarizerExample").getOrCreate()

//importamos la variable spark de forma implicita

import spark.implicits

//importar Summarizer

import Summarizer.\_

//valor data contendrá dos vectores

val data = Seq(

(Vectors.dense(2.0, 3.0, 5.0), 1.0),

(Vectors.dense(4.0, 6.0, 7.0), 2.0)

)

//el valor df para hacer referencia a features y weight

val df = data.toDF("features", "weight")

//dos valores para media y varianza;

//selec las metricas (Mean,Variance)

val (meanVal, varianceVal) = df.select(metrics("mean", "variance").summary($"features", $"weight").as("summary")).select("summary.mean", "summary.variance").as[(Vector, Vector)].first()

//método summary que se lo aplicamos a features y weight y le damos un alias y seleccionaremos el sumary con su respectiva métrica y le damos la forma en que deseamos imprimir

//mandamos a la impresión para mostrar los valores de la mean y varianza

println(s"with weight: mean = ${meanVal}, variance = ${varianceVal}")

//segundo valor sin usar el método summary

val (meanVal2, varianceVal2) = df.select(mean($"features"), variance($"features")).as[(Vector, Vector)].first()

//imprimimos la media de la variable 2

println(s"without weight: mean = ${meanVal2}, sum = ${varianceVal2}")

spark.stop()

}

}

//Resultado

