TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLOGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

SEMESTRE FEBRERO-JUNIO 2022

MATERIA:

Datos masivos.

UNIDAD 2

Practica 1

DOCENTE:

JOSE CHRISTIAN ROMERO HERNANDEZ

INTEGRANTES:

López Higuera Saúl Alfredo #18210493

Ramos Rivera Manuel Isaí #17212931

Tijuana BC 04 de mayo del 2022

```
//CORRELATION

package org.apache.spark.examples.mllib

import org.apache.spark.{SparkConf, SparkContext}
// $example on$
import org.apache.spark.mllib.linalg._
import org.apache.spark.mllib.stat.Statistics
import org.apache.spark.rdd.RDD
object CorrelationsExample {

def main(){

    val conf = new SparkConf().setAppName("CorrelationsExample")
    val sc = new SparkContext(conf)

    // $example on$
    val seriesX: RDD[Double] = sc.parallelize(Array(1, 2, 3, 3, 5))  // a
series
    // must have the same number of partitions and cardinality as seriesX
    val seriesY: RDD[Double] = sc.parallelize(Array(11, 22, 33, 33, 555))

    // compute the correlation using Pearson's method. Enter "spearman" for
Spearman's method. If a
    // method is not specified, Pearson's method will be used by default.
    val correlation: Double = Statistics.corr(seriesX, seriesY, "pearson")
    println(s"Correlation is: $correlation")

    val data: RDD[Vector] = sc.parallelize(
    Seq(
        Vectors.dense(1.0, 10.0, 100.0),
        Vectors.dense(2.0, 20.0, 200.0),
        Vectors.dense(5.0, 33.0, 366.0))
    )  // note that each Vector is a row and not a column

    // calculate the correlation matrix using Pearson's method. Use
"spearman" for Spearman's method
    // If a method is not specified, Pearson's method will be used by
default.
    val correlMatrix: Matrix = Statistics.corr(data, "pearson")
    println(correlMatrix.toString)
    // $example off$

    sc.stop()
    }
```

```
}

//HYPOTHESIS TESTING

import org.apache.spark.{SparkConf, SparkContext}
// $example on$
import org.apache.spark.mllib.linalg._
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.stat.Statistics
import org.apache.spark.mllib.stat.test.ChiSqTestResult
import org.apache.spark.rdd.RDD
// $example off$

object HypothesisTestingExample {

def main() {

    val conf = new SparkConf().setAppName("HypothesisTestingExample")
    val sc = new SparkContext(conf)

    // $example on$
    // a vector composed of the frequencies of events
    val vec: Vector = Vectors.dense(0.1, 0.15, 0.2, 0.3, 0.25)

    // compute the goodness of fit. If a second vector to test against is
not supplied
    // as a parameter, the test runs against a uniform distribution.
    val goodnessOfFitTestResult = Statistics.chiSqTest(vec)
    // summary of the test including the p-value, degrees of freedom, test
statistic, the method
    // used, and the null hypothesis.
    println(s"$goodnessOfFitTestResult\n")

    // a contingency matrix. Create a dense matrix ((1.0, 2.0), (3.0, 4.0),
(5.0, 6.0))
    val mat: Matrix = Matrices.dense(3, 2, Array(1.0, 3.0, 5.0, 2.0, 4.0,
6.0))

    // conduct Pearson's independence test on the input contingency matrix
    val independenceTestResult = Statistics.chiSqTest(mat)
    // summary of the test including the p-value, degrees of freedom
    println(s"$independenceTestResult\n")

    val obs: RDD[LabeledPoint] =
    sc.parallelize(
```

```
        Seq(
        LabeledPoint(1.0, Vectors.dense(1.0, 0.0, 3.0)),
        LabeledPoint(1.0, Vectors.dense(1.0, 2.0, 0.0)),
        LabeledPoint(-1.0, Vectors.dense(-1.0, 0.0, -0.5)
        )
        )
    ) // (label, feature) pairs.

    // The contingency table is constructed from the raw (label, feature)
pairs and used to conduct
    // the independence test. Returns an array containing the
ChiSquaredTestResult for every feature
    // against the label.
    val featureTestResults: Array[ChiSqTestResult] =
Statistics.chiSqTest(obs)
    featureTestResults.zipWithIndex.foreach { case (k, v) =>
    println(s"Column ${(v + 1)} :")
    println(k)
    }  // summary of the test
    // $example off$

    sc.stop()
    }
}

//SUMMARIZER

import org.apache.spark.{SparkConf, SparkContext}
// $example on$
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.stat.{MultivariateStatisticalSummary,
Statistics}
// $example off$
object SummaryStatisticsExample {

def main() {
    // $example on$
    val observations = sc.parallelize(
    Seq(
        Vectors.dense(1.0, 10.0, 100.0),
        Vectors.dense(2.0, 20.0, 200.0),
        Vectors.dense(3.0, 30.0, 300.0)
        )
    )
```

```
    // Compute column summary statistics.
    val summary: MultivariateStatisticalSummary =
Statistics.colStats(observations)
    println(summary.mean)  // a dense vector containing the mean value for
each column
    println(summary.variance)  // column-wise variance
    println(summary.numNonzeros)  // number of nonzeros in each column
    // $example off$

    sc.stop()
    println("hello world")
    }
}
```