TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:

Enero - Junio 2021

CARRERA:

Ingeniería en Sistemas Computacionales

MATERIA:

Datos Masivos

TÍTULO:

Práctica-Basic Statics

UNIDAD A EVALUAR:

Unidad -2

ALUMNO: JUAN ANTONIO ACEVES ZAMORA

NO. CONTROL:16210502

NOMBRE DEL DOCENTE :

JOSE CHRISTIAN ROMERO HERNANDEZ

Correlation

Empezamos con esta librería para tener acceso a matrices locales y Métodos de fábrica para Vector.

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

Librería para usar el método de correlación

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

Permite acceder a un valor de una fila a través del acceso genérico por ordinal, así como el acceso primitivo

```
import org.apache.spark.sql.Row
```

Crea vectores densos y dispersos a partir de sus valores, dentro de la matriz

```
val data = Seq(
   (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 extraen los datos de nuestra matriz y se crea un dataframe respecto a las características

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

Se crea la matriz de correlación Pearson usando el dataframe que acabamos de crear y le pedimos los primeros valores con head

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

Imprimimos el resultado

```
println(s"Pearson correlation matrix:\n $coeff1")
```

Se crea la matriz de correlación Spearman usando el dataframe que acabamos de crear y le pedimos los primeros valores con head

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

Imprimimos el resultado

```
println(s"Spearman correlation matrix:\n $coeff2")
```

Hypothesis testing

Se hace uso de la siguiente librería para aplicar métodos a vectores

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

También se utiliza la librería de chiSquare para realizar los cálculos necesarios

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

Se crea la siguiente secuencia de vectores densos

```
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))
)
```

Creación del dataframe a partir del conjunto de vectores anterior

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

Se toman los primeros valores del dataframe previamente creado

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

Inicia con las partes de la prueba, se buscarán los valores de p

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

Después se buscarán los grados de libertad del modelo

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

Por último se extraerán ciertos valores de un vector determinado todo en base a la función chi cuadrado

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

Summarizer

Importación de librerías necesarias, en este uso de vectores y el propio summarizer

```
import spark.implicits._
import Summarizer._
```

Se crea un conjunto de vectores o secuencia

```
val data = Seq(
  (Vectors.dense(2.0, 3.0, 5.0), 1.0),
  (Vectors.dense(4.0, 6.0, 7.0), 2.0)
)
```

Creación del dataframe a partir de los vectores

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

Se hace uso de la librería summarizer para obtener la media y la varianza de algunos datos en el dataframe solicitado

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

Se imprimen las variables trabajadas anteriormente

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

Se repite el procesos con 2 nuevas variables

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

Impresión de variables

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