TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

NOMBRE DE LOS ALUMNOS:

RAYMUNDO HIRALES LAZARENO (N. CONTROL: 17212339)

PAULA ANDREA RAMOS VERDIN (N. CONTROL: 18210721)

Carrera: Ingeniería Informática

MATERIA: Datos Masivos

PROFESOR: JOSE CHRISTIAN ROMERO HERNANDEZ

Practica Evaluatoria U2

FECHA: 23/05/22

Introducción

En el presente documento se expondrá a detalle la práctica evaluatoria de la unidad 2, donde veremos la utilización de un dataframe y el uso de scripts para su manejo.

Desarrollo

//RAYMUNDO HIRALES LAZARENO - 17212339 //PAULA ANDREA RAMOS VERDIN - 18210721 //Exam Unit 2 - 22/03/22

```
import org.apache.spark.sql.types.DoubleType import
org.apache.spark.ml.classification.MultilayerPerceptronClassifier import
org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator import
org.apache.spark.ml.feature.IndexToString import org.apache.spark.ml.feature.StringIndexer import
org.apache.spark.ml.feature.VectorIndexer import org.apache.spark.ml.feature.VectorAssembler import
org.apache.spark.ml.feature.IndexToString import org.apache.spark.ml.Pipeline
```

//1-.Cargar el dataframe iris val iris=spark.read.format("csv").option("header","true").load("iris.csv")

```
//Limpiar el dataframe val df = iris.withColumn("sepal_length",
$"sepal_length".cast(DoubleType)).withColumn("sepal_width",
$"sepal_width".cast(DoubleType)).withColumn("petal_length",
$"petal_length".cast(DoubleType)).withColumn("petal_width", $"petal_width".cast(DoubleType))
```

//2-.¿cual es el nombre de las columnas? df.columns

```
res0: Array[String] = Array(sepal_length, sepal_width, petal_length, petal_width, species)
```

//3-.¿Como es el esquema? df.printSchema()

```
scala> df.printSchema()
root
 |-- sepal_length: double (nullable = true)
 |-- sepal_width: double (nullable = true)
 |-- petal_length: double (nullable = true)
 |-- petal_width: double (nullable = true)
 |-- species: string (nullable = true)
```

//4-.Imprimir las primeras 5 columnas df.show(5)

```
scala> df.show(5)
+------------+-----------+------------+-----------+-------+
|sepal_length|sepal_width|petal_length|petal_width|species|
+------------+-----------+------------+-----------+-------+
|         5.1|        3.5|         1.4|        0.2| setosa|
|         4.9|        3.0|         1.4|        0.2| setosa|
|         4.7|        3.2|         1.3|        0.2| setosa|
|         4.6|        3.1|         1.5|        0.2| setosa|
|         5.0|        3.6|         1.4|        0.2| setosa|
+------------+-----------+------------+-----------+-------+
only showing top 5 rows
```

//5-. Usa el metodo describe () para aprender mas sobre los datos del DataFrame df.describe().show()

```
scala> df.describe().show()
+-------+------------------+-------------------+------------------+-------------------+---------+
|summary|      sepal_length|        sepal_width|      petal_length|       petal_width|  species|
+-------+------------------+-------------------+------------------+-------------------+---------+
|  count|               150|                150|               150|                150|      150|
|   mean| 5.843333333333335| 3.0540000000000007|3.7586666666666693|1.1986666666666672|     null|
| stddev|0.8280661279778637|0.43359431136217375| 1.764420419952262|0.7631607417008414|     null|
|    min|               4.3|                2.0|               1.0|                0.1|   setosa|
|    max|               7.9|                4.4|               6.9|                2.5|virginica|
+-------+------------------+-------------------+------------------+-------------------+---------+
```

//6-.Haga la transformación pertinente para los datos categoricos los cuales serán nuestras etiquetas a clasificar.

val assembler = new VectorAssembler().setInputCols(Array("sepal_length", "sepal_width", "petal_length", "petal_width")).setOutputCol("features") val features = assembler.transform(df)

val indexerL = new StringIndexer().setInputCol("species").setOutputCol("indexedLabel").fit(features) val indexerF = new VectorIndexer().setInputCol("features").setOutputCol("indexedFeatures").setMaxCategories(4).fit(features)

val splits = features.randomSplit(Array(0.6, 0.4)) val training = splits(0) val test = splits(1)

val layers = Array[Int](4, 5, 5, 3)

//7-.Construya el modelo de clasificación y explique su arquitectura.

val trainer = new MultilayerPerceptronClassifier().setLayers(layers).setLabelCol("indexedLabel").setFeaturesCol("indexedFeatures").setBlockSize(128).setSeed(System.currentTimeMillis).setMaxIter(200) val converterL = new IndexToString().setInputCol("prediction").setOutputCol("predictedLabel").setLabels(indexerL.labels) val pipeline = new Pipeline().setStages(Array(indexerL, indexerF, trainer, converterL))

val model = pipeline.fit(training)

//8-.Imprima los resultados del modelo

val predictions = model.transform(test)

val evaluator = new MulticlassClassificationEvaluator().setLabelCol("indexedLabel").setPredictionCol("prediction").setMetricName("accuracy") val accuracy = evaluator.evaluate(predictions) println("Error = " + (1.0 - accuracy))

```
scala> val accuracy = evaluator.evaluate(predictions)
accuracy: Double = 0.6764705882352942

scala> println("Error = " + (1.0 - accuracy))
Error = 0.32352941176470584
```

Conclusión

Se puede decir que lo mas importante de esta practica es el amplio uso que podemos darle a los scripts de scala para el manejo de dataframes a simple vista estos comandos pueden parecer de lo más complicado pero en realidad es una herramienta sencilla, intuitiva y practica, resulto bastante interesante y enriquecedor esta practica evaluatoria unidad 2.

link de youtube