EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

**INSTITUTO TECNOLÓGICO DE TIJUANA**

**SUBDIRECCIÓN ACADÉMICA**

**DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN**

**SEMESTRE FEBRERO- JUNIO 2021**

**TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN**
**ING. INFORMÁTICA**

**Datos Masivos**

# EXAMEN Unidad 2

**Aceves Zamora Juan Antonio**
**Briseño Cota Raúl Omar**

**Profe. JOSE CHRISTIAN ROMERO HERNANDEZ**

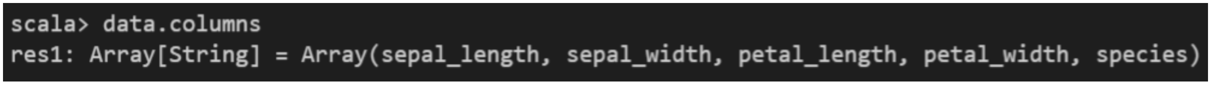
# Código

1. Cargar en un dataframe Iris.csv que se encuentra en https://github.com/jcromerohdz/iris, elaborar la liempieza de datos necesaria para ser procesado por el siguiente algoritmo (Importante, esta limpieza debe ser por medio de un script de Scala en Spark) .

```
//Carga de librerías
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.classification.MultilayerPerceptronClassifier
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.types.IntegerType
import org.apache.spark.ml.feature.StringIndexer
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.linalg.Vectors
import org.apache.spark.ml.feature.VectorIndexer
import org.apache.spark.ml.feature.IndexToString

//Creación de sesión de Spark
val spark = SparkSession.builder().getOrCreate()

//Carga de dataframe
val data = spark.read.option("header",
"true").option("inferSchema","true")csv("C:/Users/brise/Documents/Github/iri
s/iris.csv")

//Limpieza de valores nulos
data.na.drop()
```

2. ¿Cuáles son los nombres de las columnas?

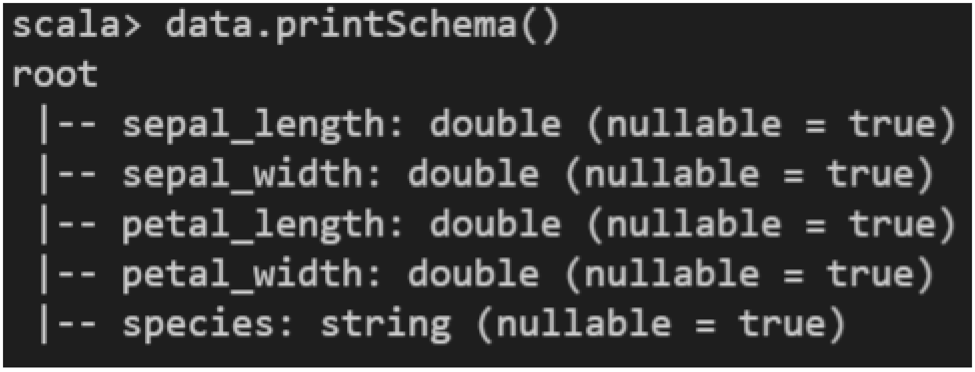
```
//Mostrar columnas
data.columns
```



```
scala> data.columns
res1: Array[String] = Array(sepal_length, sepal_width, petal_length, petal_width, species)
```

3. ¿Cómo es el esquema?

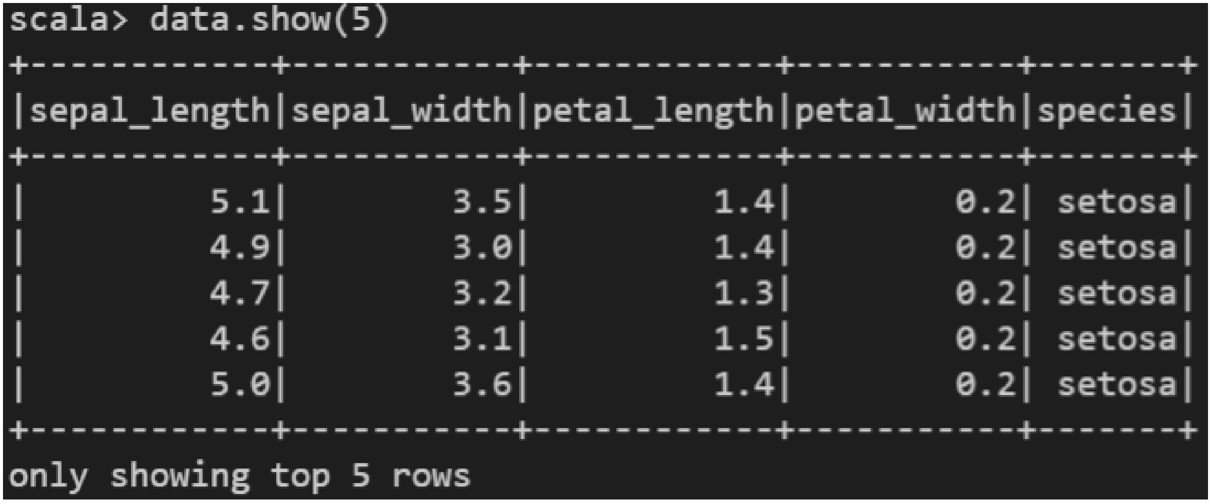
```
//Impresión de esquema
data.printSchema()
```



```
scala> data.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. Imprime las primeras 5 columnas.

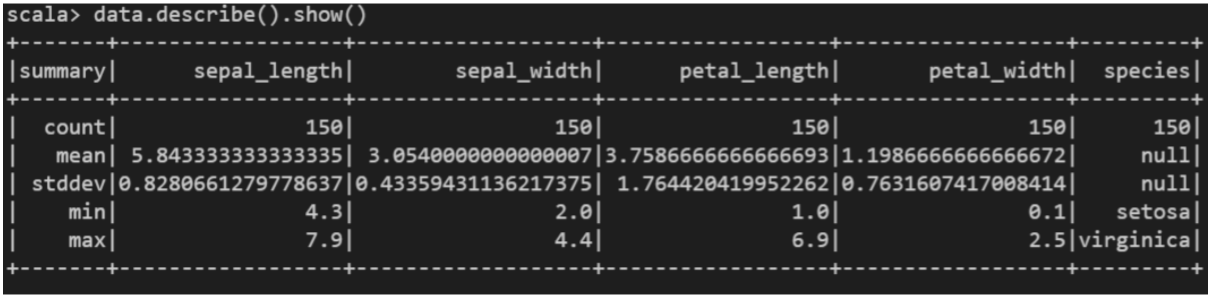
```
//Mostrar las primeras 5 filas
data.show(5)
```



```
scala> data.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.

```
data.describe().show()
```

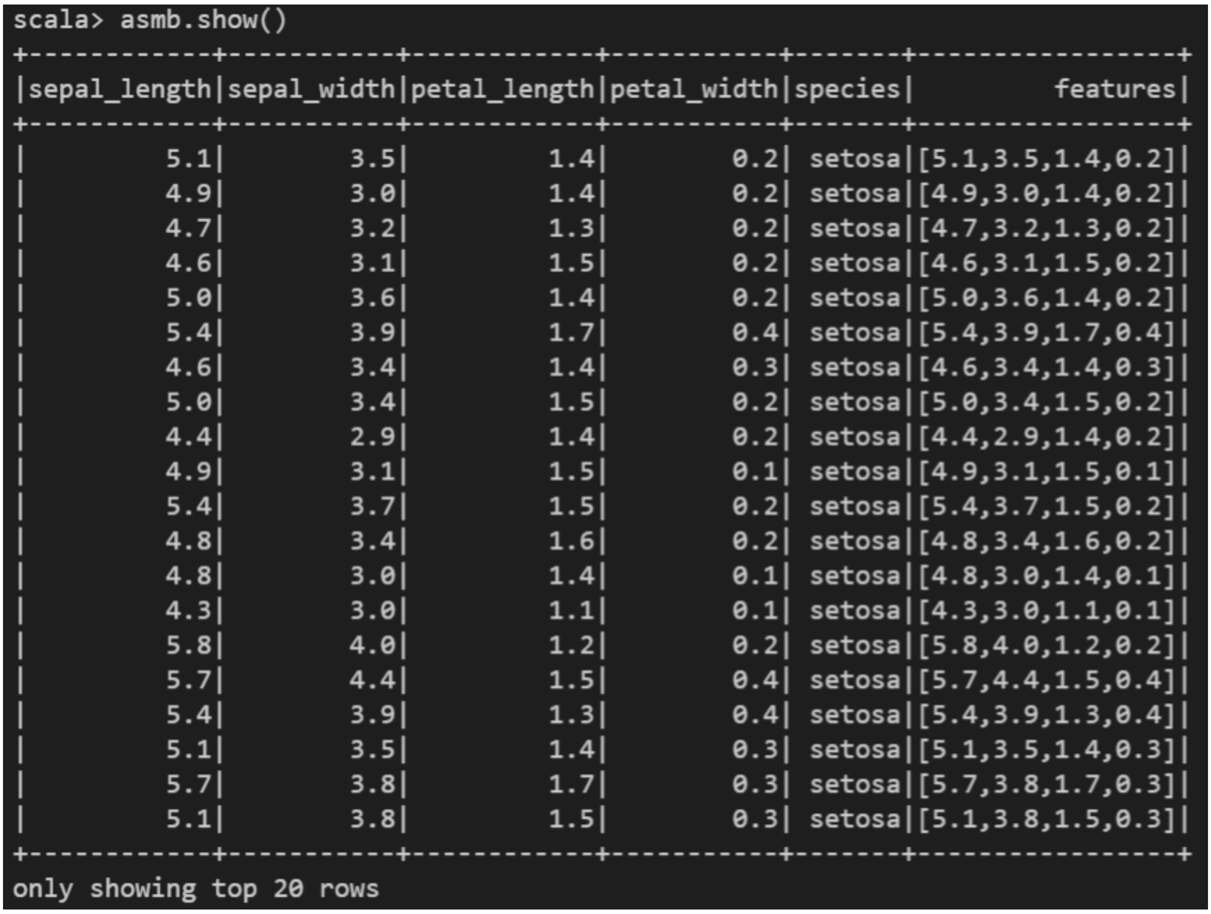


```
scala> data.describe().show()
+-------+------------------+------------------+------------------+------------------+---------+
|summary|      sepal_length|       sepal_width|      petal_length|       petal_width|  species|
+-------+------------------+------------------+------------------+------------------+---------+
|  count|               150|               150|               150|               150|      150|
|   mean| 5.843333333333335| 3.054000000000007|3.7586666666666693|1.1986666666666672|     null|
| stddev|0.8280661279778637|0.4335943113621737| 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|
+-------+------------------+------------------+------------------+------------------+---------+
```

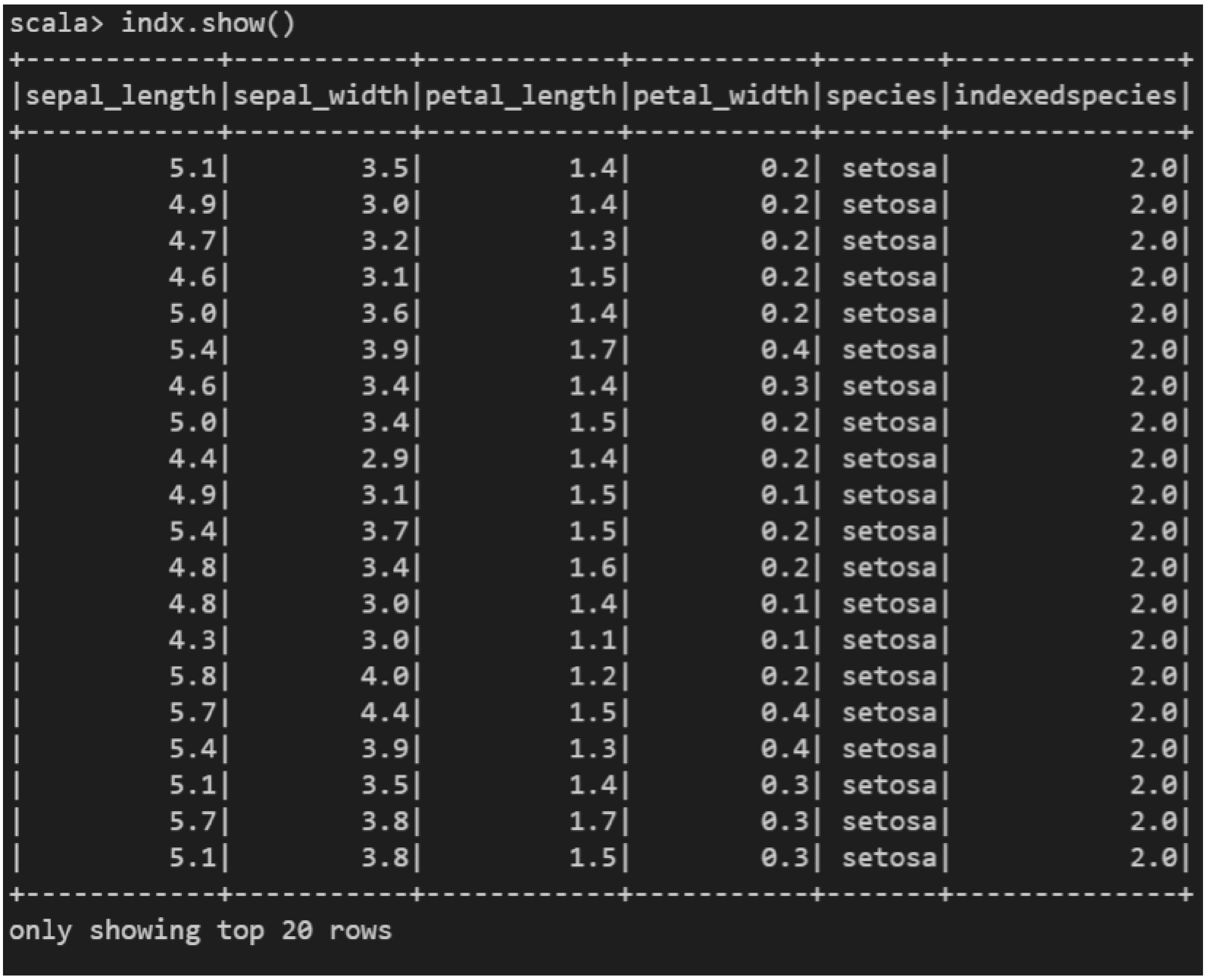
6. Haga la transformación pertinente para los datos categoricos los cuales serán nuestras etiquetas a clasificar.

```
val assembler = new
VectorAssembler().setInputCols(Array("sepal_length","sepal_width","petal_len
gth","petal_width")).setOutputCol("features")

val asmb = assembler.transform(data)

asmb.show()

val labelIndexer = new
StringIndexer().setInputCol("species").setOutputCol("indexedspecies").fit(da
ta)

val lblInd = new
StringIndexer().setInputCol("species").setOutputCol("indexedspecies")

val indx = lblInd.fit(data).transform(data)

indx.show()
```

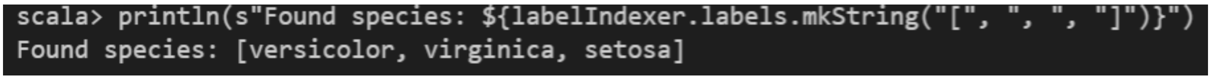
```
println(s"Found species: ${labelIndexer.labels.mkString("[", ", ", "]")}")

val indexed =
labelIndexer.transform(data).withColumnRenamed("indexedSpecies", "label")

val features = assembler.transform(indexed)

features.show()

val featureIndexer = new
StringIndexer().setInputCol("label").setOutputCol("indexedSpecies").fit(indexed)

val splits = features.randomSplit(Array(0.6, 0.4), seed = 1234L)

val train = splits(0)

val test = splits(1)

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



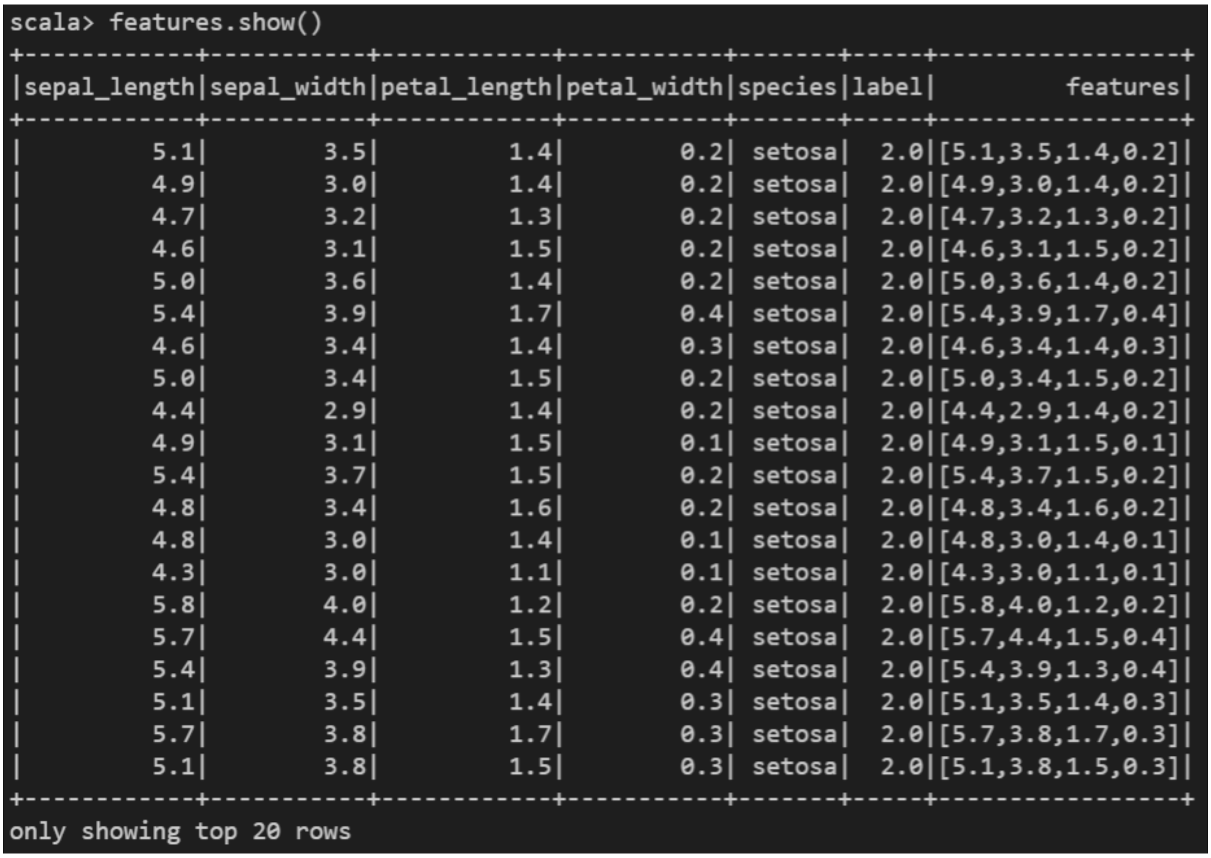
```
scala> asmb.show()
+------------+-----------+------------+-----------+-------+-----------------+
|sepal_length|sepal_width|petal_length|petal_width|species|         features|
+------------+-----------+------------+-----------+-------+-----------------+
|         5.1|        3.5|         1.4|        0.2| setosa|[5.1,3.5,1.4,0.2]|
|         4.9|        3.0|         1.4|        0.2| setosa|[4.9,3.0,1.4,0.2]|
|         4.7|        3.2|         1.3|        0.2| setosa|[4.7,3.2,1.3,0.2]|
|         4.6|        3.1|         1.5|        0.2| setosa|[4.6,3.1,1.5,0.2]|
|         5.0|        3.6|         1.4|        0.2| setosa|[5.0,3.6,1.4,0.2]|
|         5.4|        3.9|         1.7|        0.4| setosa|[5.4,3.9,1.7,0.4]|
|         4.6|        3.4|         1.4|        0.3| setosa|[4.6,3.4,1.4,0.3]|
|         5.0|        3.4|         1.5|        0.2| setosa|[5.0,3.4,1.5,0.2]|
|         4.4|        2.9|         1.4|        0.2| setosa|[4.4,2.9,1.4,0.2]|
|         4.9|        3.1|         1.5|        0.1| setosa|[4.9,3.1,1.5,0.1]|
|         5.4|        3.7|         1.5|        0.2| setosa|[5.4,3.7,1.5,0.2]|
|         4.8|        3.4|         1.6|        0.2| setosa|[4.8,3.4,1.6,0.2]|
|         4.8|        3.0|         1.4|        0.1| setosa|[4.8,3.0,1.4,0.1]|
|         4.3|        3.0|         1.1|        0.1| setosa|[4.3,3.0,1.1,0.1]|
|         5.8|        4.0|         1.2|        0.2| setosa|[5.8,4.0,1.2,0.2]|
|         5.7|        4.4|         1.5|        0.4| setosa|[5.7,4.4,1.5,0.4]|
|         5.4|        3.9|         1.3|        0.4| setosa|[5.4,3.9,1.3,0.4]|
|         5.1|        3.5|         1.4|        0.3| setosa|[5.1,3.5,1.4,0.3]|
|         5.7|        3.8|         1.7|        0.3| setosa|[5.7,3.8,1.7,0.3]|
|         5.1|        3.8|         1.5|        0.3| setosa|[5.1,3.8,1.5,0.3]|
+------------+-----------+------------+-----------+-------+-----------------+
only showing top 20 rows
```



```
scala> indx.show()
+------------+-----------+------------+-----------+-------+--------------+
|sepal_length|sepal_width|petal_length|petal_width|species|indexedspecies|
+------------+-----------+------------+-----------+-------+--------------+
|         5.1|        3.5|         1.4|        0.2| setosa|           2.0|
|         4.9|        3.0|         1.4|        0.2| setosa|           2.0|
|         4.7|        3.2|         1.3|        0.2| setosa|           2.0|
|         4.6|        3.1|         1.5|        0.2| setosa|           2.0|
|         5.0|        3.6|         1.4|        0.2| setosa|           2.0|
|         5.4|        3.9|         1.7|        0.4| setosa|           2.0|
|         4.6|        3.4|         1.4|        0.3| setosa|           2.0|
|         5.0|        3.4|         1.5|        0.2| setosa|           2.0|
|         4.4|        2.9|         1.4|        0.2| setosa|           2.0|
|         4.9|        3.1|         1.5|        0.1| setosa|           2.0|
|         5.4|        3.7|         1.5|        0.2| setosa|           2.0|
|         4.8|        3.4|         1.6|        0.2| setosa|           2.0|
|         4.8|        3.0|         1.4|        0.1| setosa|           2.0|
|         4.3|        3.0|         1.1|        0.1| setosa|           2.0|
|         5.8|        4.0|         1.2|        0.2| setosa|           2.0|
|         5.7|        4.4|         1.5|        0.4| setosa|           2.0|
|         5.4|        3.9|         1.3|        0.4| setosa|           2.0|
|         5.1|        3.5|         1.4|        0.3| setosa|           2.0|
|         5.7|        3.8|         1.7|        0.3| setosa|           2.0|
|         5.1|        3.8|         1.5|        0.3| setosa|           2.0|
+------------+-----------+------------+-----------+-------+--------------+
only showing top 20 rows
```



```
scala> println(s"Found species: ${labelIndexer.labels.mkString("[", ", ", "]")}")
Found species: [versicolor, virginica, setosa]
```



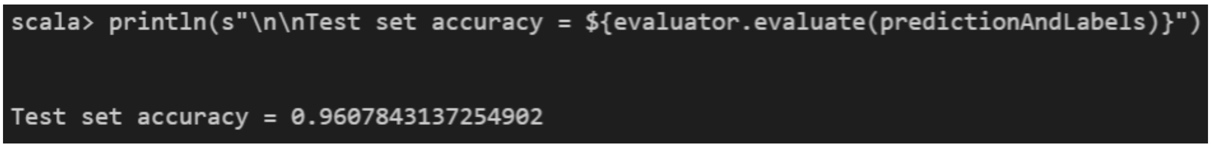
```
scala> features.show()
+------------+-----------+------------+-----------+-------+-----+-----------------+
|sepal_length|sepal_width|petal_length|petal_width|species|label|         features|
+------------+-----------+------------+-----------+-------+-----+-----------------+
|         5.1|        3.5|         1.4|        0.2| setosa|  2.0|[5.1,3.5,1.4,0.2]|
|         4.9|        3.0|         1.4|        0.2| setosa|  2.0|[4.9,3.0,1.4,0.2]|
|         4.7|        3.2|         1.3|        0.2| setosa|  2.0|[4.7,3.2,1.3,0.2]|
|         4.6|        3.1|         1.5|        0.2| setosa|  2.0|[4.6,3.1,1.5,0.2]|
|         5.0|        3.6|         1.4|        0.2| setosa|  2.0|[5.0,3.6,1.4,0.2]|
|         5.4|        3.9|         1.7|        0.4| setosa|  2.0|[5.4,3.9,1.7,0.4]|
|         4.6|        3.4|         1.4|        0.3| setosa|  2.0|[4.6,3.4,1.4,0.3]|
|         5.0|        3.4|         1.5|        0.2| setosa|  2.0|[5.0,3.4,1.5,0.2]|
|         4.4|        2.9|         1.4|        0.2| setosa|  2.0|[4.4,2.9,1.4,0.2]|
|         4.9|        3.1|         1.5|        0.1| setosa|  2.0|[4.9,3.1,1.5,0.1]|
|         5.4|        3.7|         1.5|        0.2| setosa|  2.0|[5.4,3.7,1.5,0.2]|
|         4.8|        3.4|         1.6|        0.2| setosa|  2.0|[4.8,3.4,1.6,0.2]|
|         4.8|        3.0|         1.4|        0.1| setosa|  2.0|[4.8,3.0,1.4,0.1]|
|         4.3|        3.0|         1.1|        0.1| setosa|  2.0|[4.3,3.0,1.1,0.1]|
|         5.8|        4.0|         1.2|        0.2| setosa|  2.0|[5.8,4.0,1.2,0.2]|
|         5.7|        4.4|         1.5|        0.4| setosa|  2.0|[5.7,4.4,1.5,0.4]|
|         5.4|        3.9|         1.3|        0.4| setosa|  2.0|[5.4,3.9,1.3,0.4]|
|         5.1|        3.5|         1.4|        0.3| setosa|  2.0|[5.1,3.5,1.4,0.3]|
|         5.7|        3.8|         1.7|        0.3| setosa|  2.0|[5.7,3.8,1.7,0.3]|
|         5.1|        3.8|         1.5|        0.3| setosa|  2.0|[5.1,3.8,1.5,0.3]|
+------------+-----------+------------+-----------+-------+-----+-----------------+
only showing top 20 rows
```

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

```
val trainer = new
MultilayerPerceptronClassifier().setLayers(layers).setBlockSize(128).setSeed
(1234L).setMaxIter(100)

val model = trainer.fit(train)

val result = model.transform(test)

val predictionAndLabels = result.select("prediction", "label")

val evaluator = new
MulticlassClassificationEvaluator().setMetricName("accuracy")
```

8. Imprima los resultados del modelo

```
println(s"\n\nTest set accuracy =
${evaluator.evaluate(predictionAndLabels)}")
```



```
scala> println(s"\n\nTest set accuracy = ${evaluator.evaluate(predictionAndLabels)}")


Test set accuracy = 0.9607843137254902
```

Link del video: https://youtu.be/hfmM-PgDZp8
Link de GitHub: https://github.com/rulom24/DatosMasivos.git