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

Examen:

Práctica Evaluatoria

MAESTRO:

JOSE CHRISTIAN ROMERO HERNANDEZ

Equipo:

Marco Antonio Rodriguez Medrano

Aide Ceballos Bobadilla

INTRODUCCIÓN:

Buenos días, tardes o noches querido profesor, este documento es una colaboración entre, el alumno: Marco Antonio Rodriguez Medrano y a mi compañera de equipo la alumna: Aide Ceballos Bobadilla, se nos pidió hacer una práctica evaluatoria la cual consiste en aplicar nuestros conocimientos sobre Machine Learning vistos en clase, pero por la actual pandemia que nos tiene aislados.

Tuvimos que trabajar remotamente con la ayuda del software TeamViewer y la plataforma meet , los cuales nos ayudaron a primero conectarnos vía remota y trabajar los dos en una misma computadora y el segundo a tener un diálogo (pair coding), el cual consiste en que un miembro del equipo tomará el rol de programador y el segundo tomará el rol de asesor del programador y después de un tiempo se cambiarán los roles los miembros del equipo.

DESARROLLO:

**//importamos librerías necesarias incluidas las librerías de Machine Learning Mllib**

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator

import org.apache.spark.ml.classification.MultilayerPerceptronClassifier

import org.apache.spark.sql.SparkSession

import org.apache.spark.ml.feature.StringIndexer

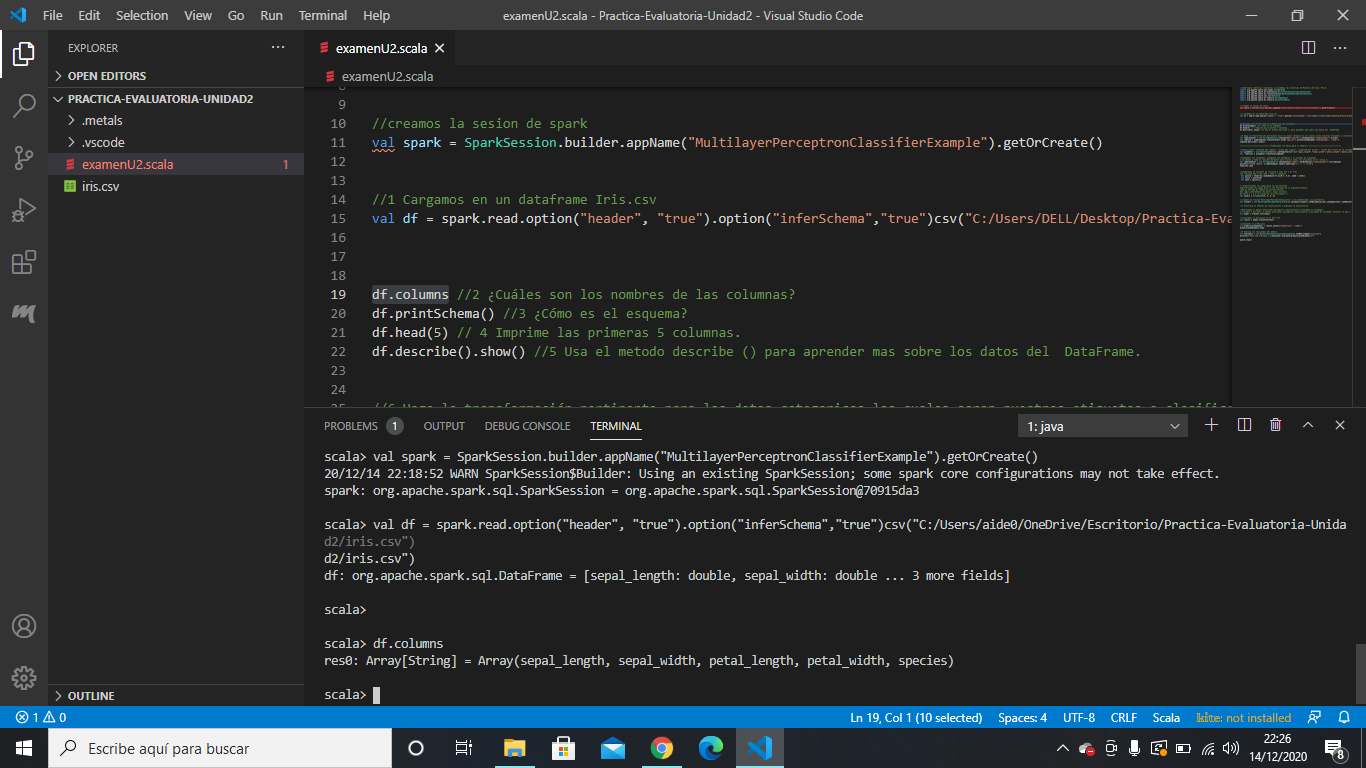
import org.apache.spark.ml.feature.VectorAssembler

**//Creamos la sesion de spark**

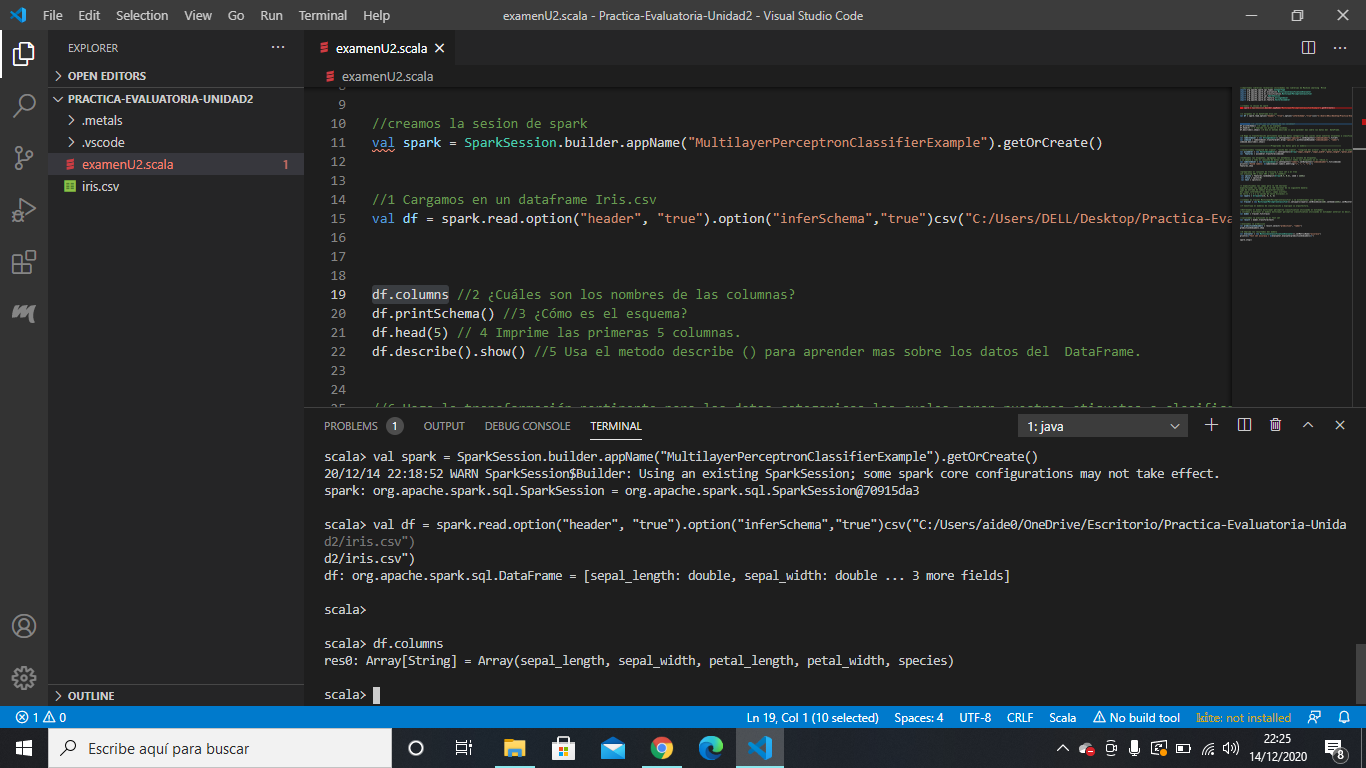
val spark = SparkSession.builder.appName("MultilayerPerceptronClassifierExample").getOrCreate()

**//1 Cargamos en un dataframe Iris.csv**

val df = spark.read.option("header", "true").option("inferSchema","true")csv("C:/Users/DELL/Desktop/Practica-Evaluatoria-Unidad2/iris.csv")

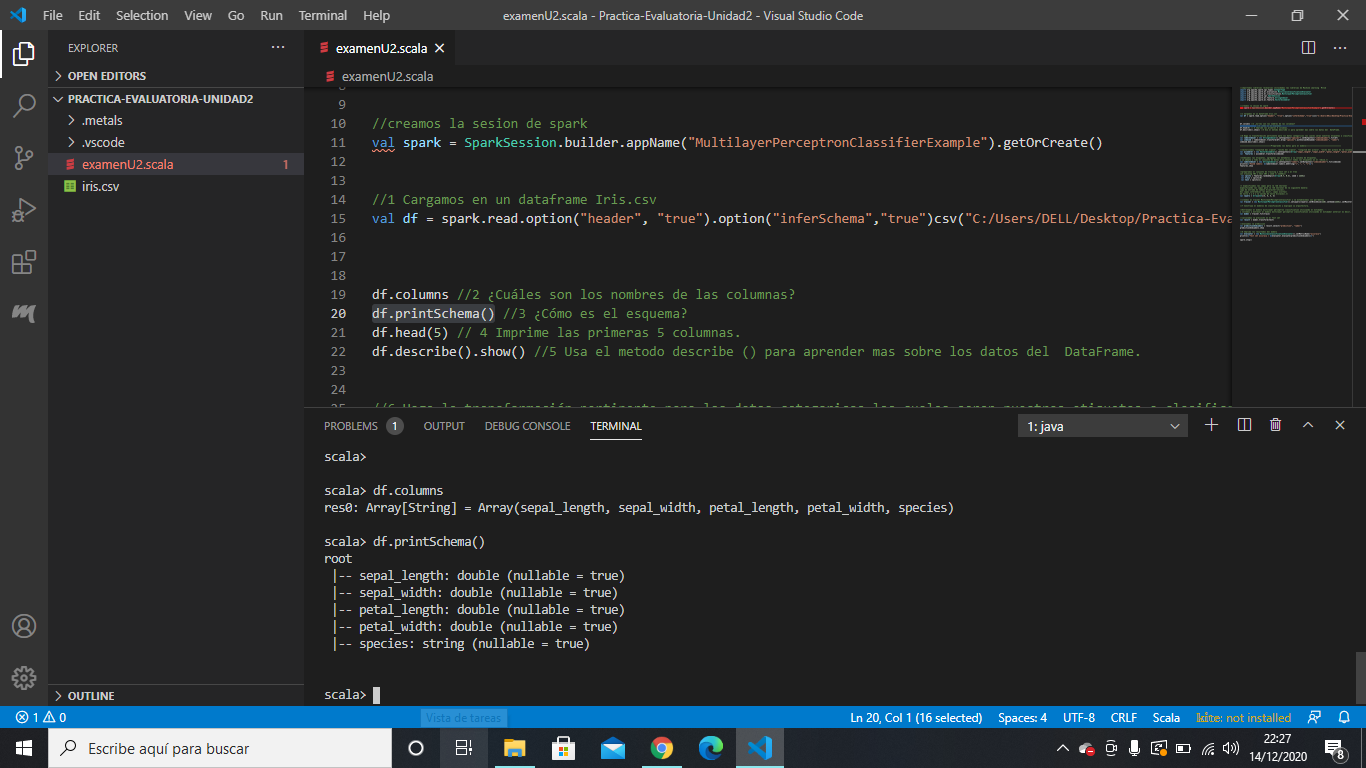


**//2 ¿Cuáles son los nombres de las columnas?**

df.columns

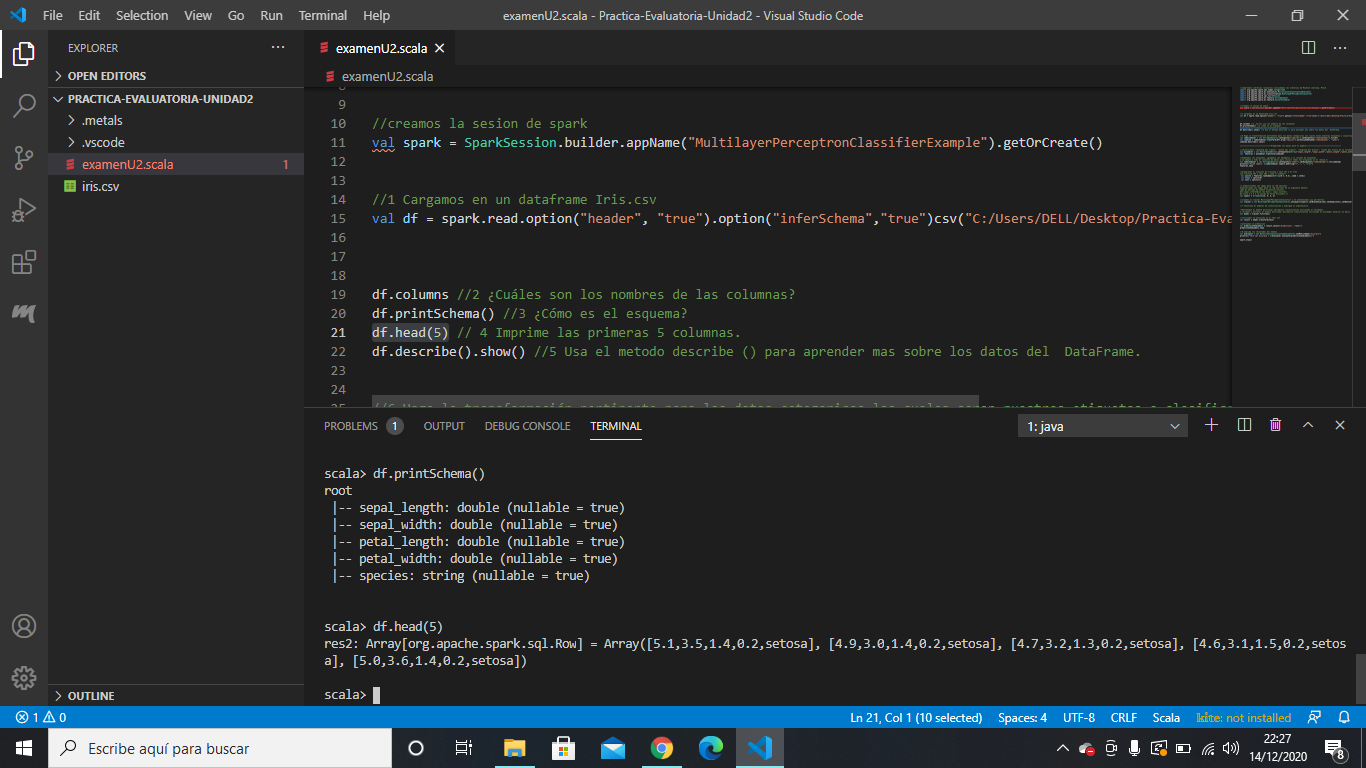
**//3 ¿Cómo es el esquema?**

df.printSchema()



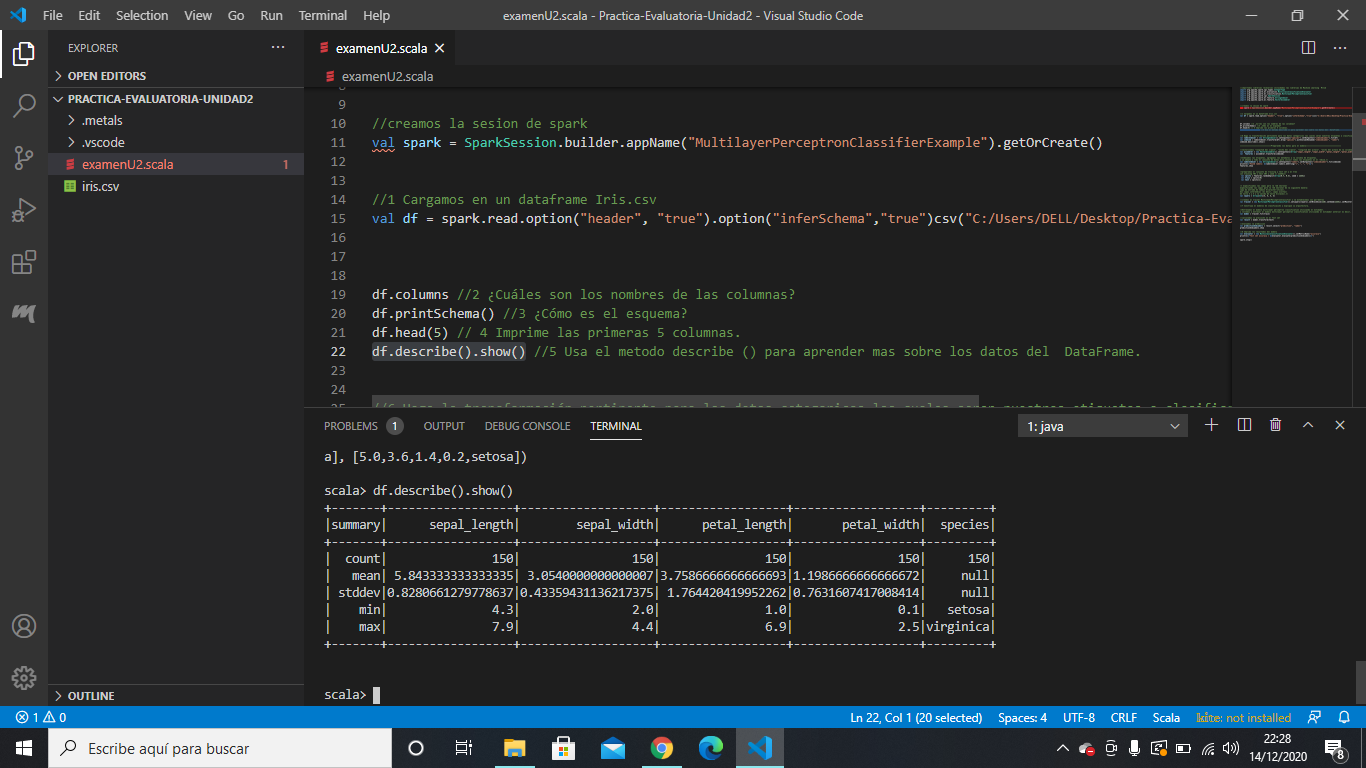
**// 4 Imprime las primeras 5 columnas.**

df.head(5)



**//5 Usa el método describe () para aprender más sobre los datos del DataFrame.**

df.describe().show()

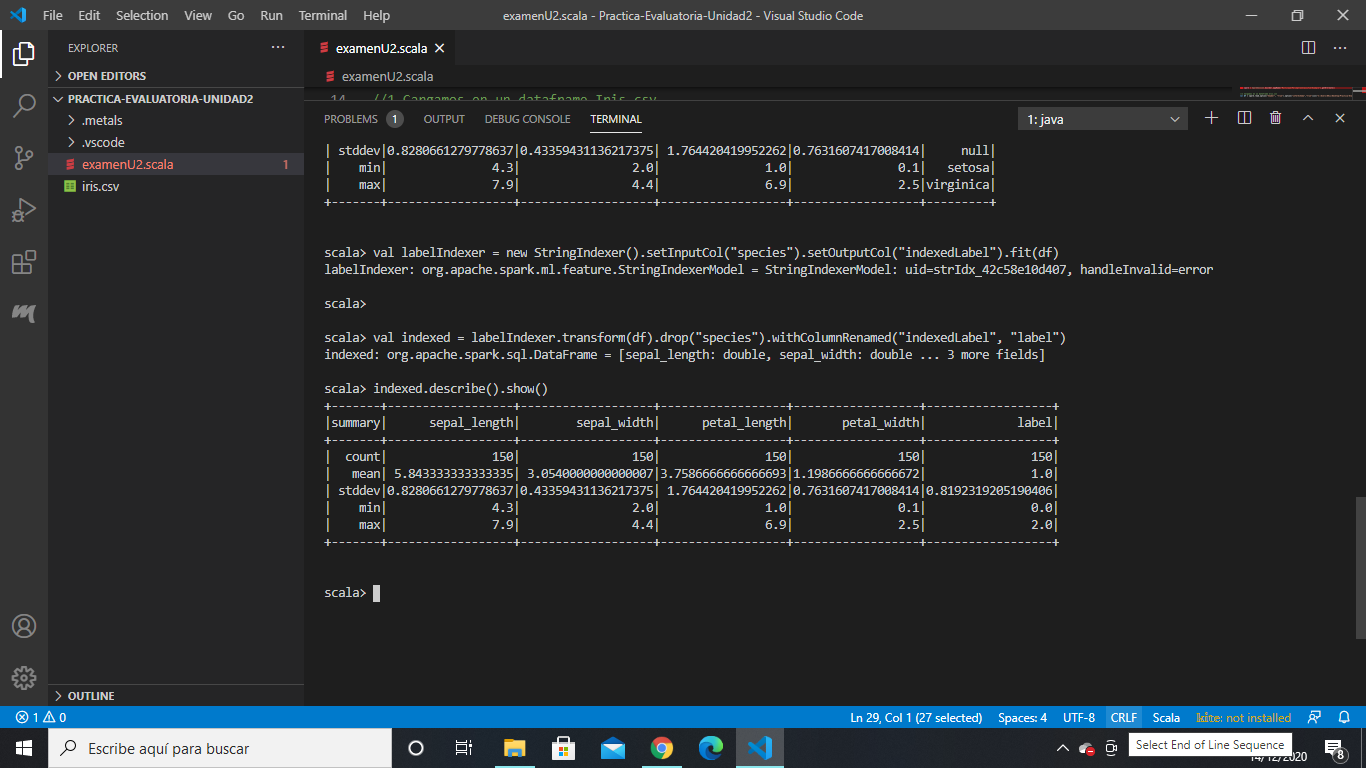


**//6 Haga la transformación pertinente para los datos categoricos los cuales seran nuestras etiquetas a clasificar.**

val labelIndexer = new StringIndexer().setInputCol("species").setOutputCol("indexedLabel").fit(df)

val indexed = labelIndexer.transform(df).drop("species").withColumnRenamed("indexedLabel", "label")

indexed.describe().show()



**///////////////////////////////Preparando los datos para el modelo///////////////////////////////////**

**//vectorizamos "longitud del sépalo", "ancho del sépalo", "longitud del pétalo", "ancho del pétalo de la columna features"**

val assembler = new

VectorAssembler().setInputCols(Array("sepal\_length","sepal\_width","petal\_length","petal\_width")).setOutputCol("features")

val features = assembler.transform(indexed)

**/\*Indexamos las etiquetas, agregamos los metadatos a la columna de etiquetas.**

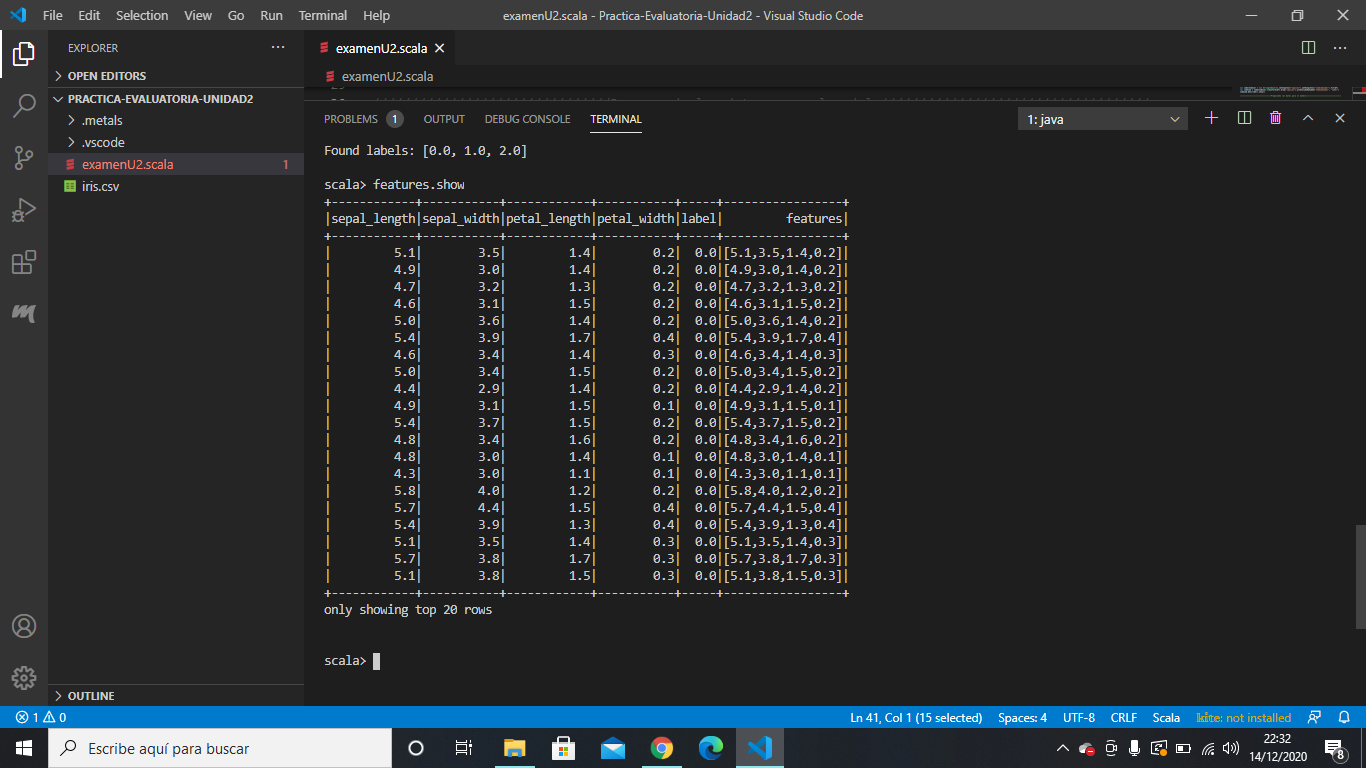
**y se adapta a todo el conjunto de datos para incluir todas las etiquetas en el índice.\*/**

val labelIndexer = new

StringIndexer().setInputCol("label").setOutputCol("indexedLabel").fit(indexed)

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

features.show



**/\*preparamos el cunjunto de training y test set y el tren**

**con training 70% y test 30% y seed => 12345L\*/**

val splits = features.randomSplit(Array(0.7, 0.3), seed = 1234L)

val train = splits(0)

val test = splits(1)

**/\* especificamos las capas para la red neuronal**

**especificando las capas para la red neuronal de la siguiente manera:**

**capa de entrada de tamaño 4(características),**

**dos capas intermedias (es decir, capas ocultas)**

**de talla 5 y 4 y la salida de talla 3(clases).\*/**

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

**//creamos el trainer MultilayerPerceptronClassifier y le establecemos sus parámetros**

val trainer = new

MultilayerPerceptronClassifier().setLayers(layers).setBlockSize(128).setSeed(1234L).setMaxIter(100)

**//7 Construya el modelos de clasificación y explique su arquitectura.**

**/\*Entrenamos el modelo multilayer perceptron classification utilizando el estimador.**

**y volvemos a entrenar al modelo multilayer perceptron classification utilizando el estimador anterior es decir, (train) \*/**

val model = trainer.fit(train)

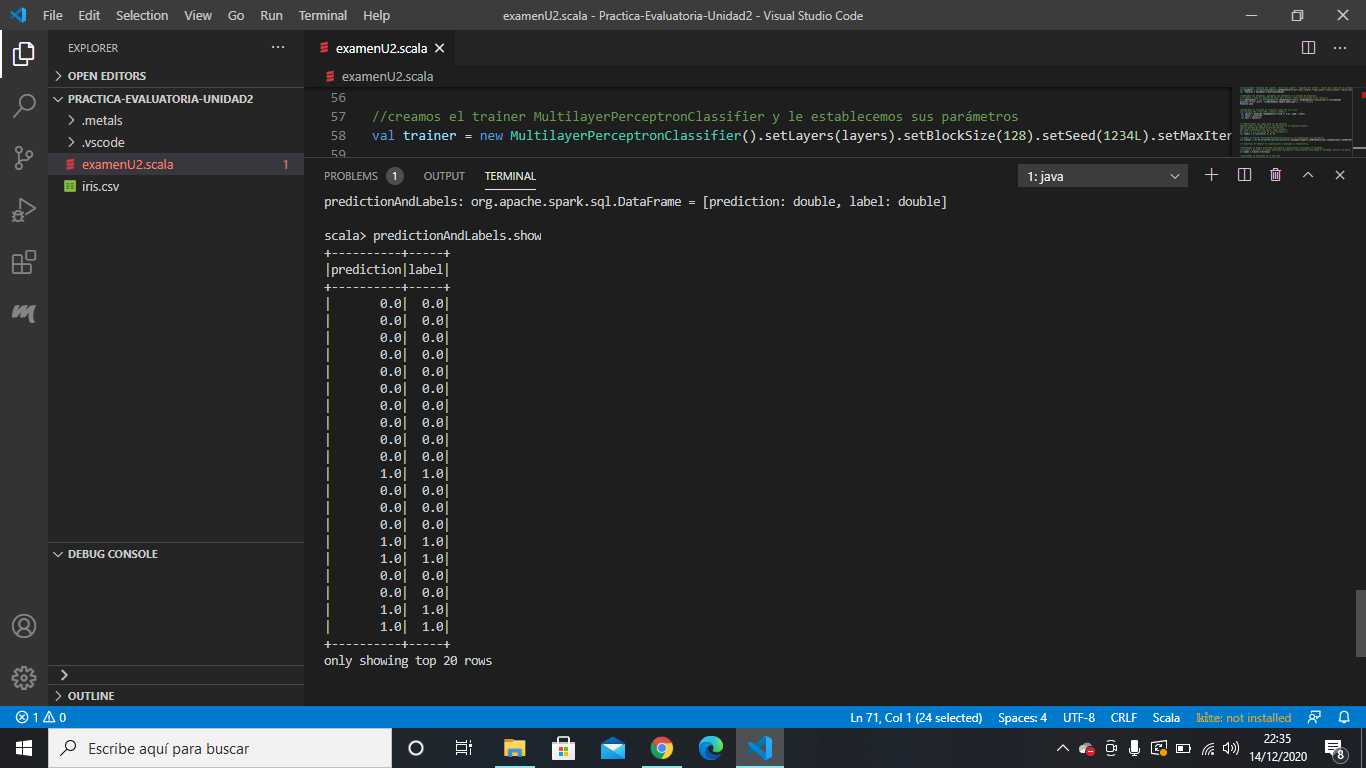
**//calculamos la precisión en el Test set**

val result = model.transform(test)

**//evaluamos la predicción**

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

predictionAndLabels.show

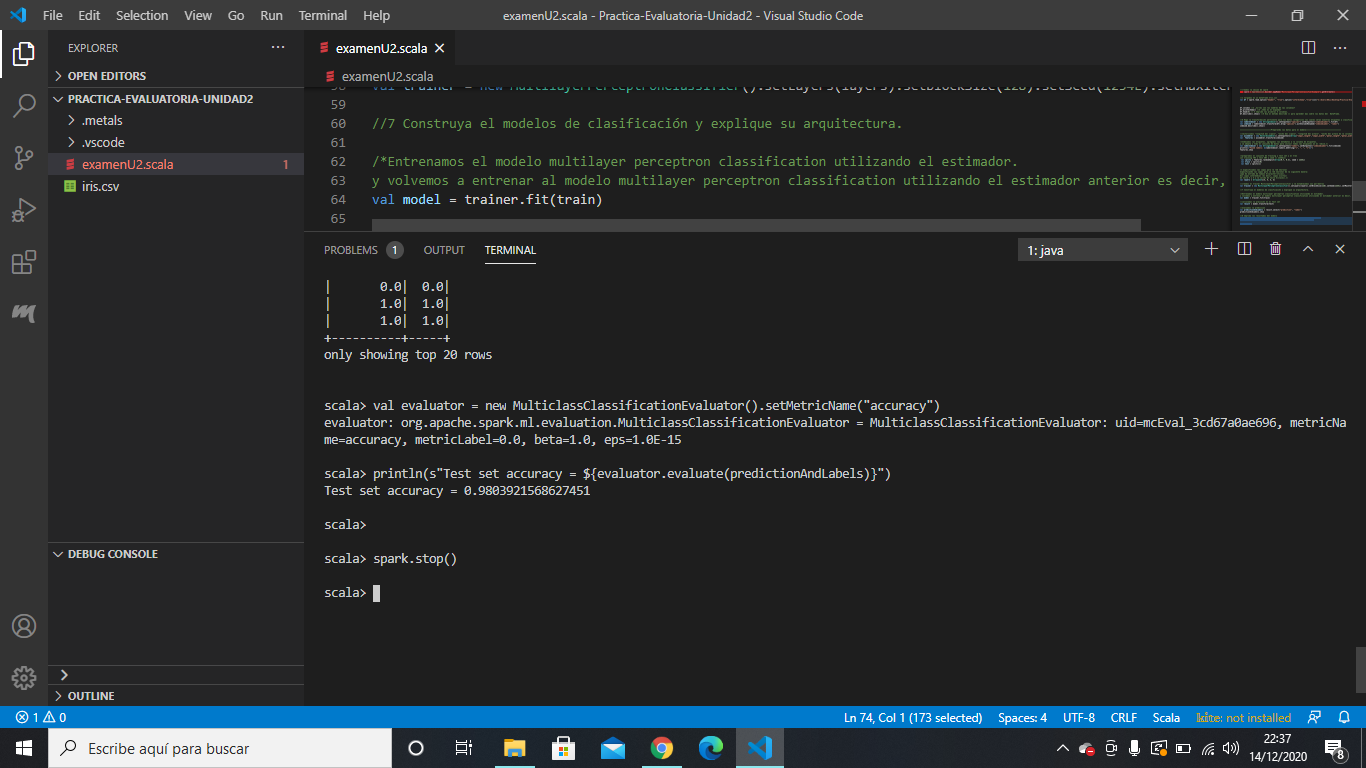


**//8 Imprima los resultados del modelo**

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

println(s"Test set accuracy = ${evaluator.evaluate(predictionAndLabels)}")

spark.stop()



CONCLUSIÓN:

-Ceballos Bobadilla Aide 15211282

Mi conclusión respecto a esta práctica, es que a través de diferentes herramientas de spark nos es posible utilizar esta tecnología para la representación, manipulación y control sobre un conjunto de datos , lo cual es muy beneficioso, ya que estas herramientas de agrupación nos permiten ordenar la información que tengamos y realizar reglas que nos ayuden a ordenar la información que vaya acumulandoce con el fin de que esta tenga una sentido y sea de utilidad.

-Rodriguez Medrano Marco Antonio 17210635

Mi conclusión con respecto a esta práctica evaluatoria, es que gracias a las múltiples herramientas del lenguaje apache spark podemos manejar eficientemente cualquier tipo de dato ya sea para obtener la precisión sobre algo en particular, ordenar los datos en grupos dependiendo su tipo, etc. pero en esta práctica utilizamos la librería Mllib de Spark entre otras pudimos crear el código anterior.