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

Tarea:

Explicación de lo que yo entendí del código:

MAESTRO:

JOSE CHRISTIAN ROMERO HERNANDEZ

24/11/2020

//importamos las librerias pipeline, DecisionTreeClassificationModel, DecisionTreeClassifier, MulticlassClassificationEvaluator, IndexToString, StringIndexer y VectorIndexer

import org.apache.spark.ml.Pipeline

import org.apache.spark.ml.classification.DecisionTreeClassificationModel

import org.apache.spark.ml.classification.DecisionTreeClassifier

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

import org.apache.spark.ml.feature.{IndexToString, StringIndexer, VectorIndexer}

//importamos la sesion spark

import org.apache.spark.sql.SparkSession

//creamos un objeto llamado DecisionTreeClassificationExample el cual contendra todos los metodos

object DecisionTreeClassificationExample {

//creamos un main con un arreglo de tipo string y va a contener la variable spark

def main(args: Array[String]): Unit = {

//creamos la variable spark y le asignamos el nombre DecisionTreeClassificationExample

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

// Cargamos los datos almacenados en formato LIBSVM como un DataFrame.

val data = spark.read.format("libsvm").load("C:/Users/DELL/Desktop/Ejercicio equipo#2 Decision tree classifier/sample\_libsvm\_data.txt")

// Etiquetas de índice, agregando metadatos a la columna de etiquetas.

// Encajar en todo el conjunto de datos para incluir todas las etiquetas en el índice.

val labelIndexer = new StringIndexer().setInputCol("label").setOutputCol("indexedLabel").fit(data)

// Identifica automáticamente características categóricas e indexalas.

val featureIndexer = new VectorIndexer().setInputCol("features").setOutputCol("indexedFeatures").setMaxCategories(4).fit(data)

// las entidades con> 4 valores distintos se tratan como continuas.

// Divida los datos en conjuntos de prueba y entrenamiento (el 30% se reserva para probar).

val Array(trainingData, testData) = data.randomSplit(Array(0.7, 0.3))

// Entrenamiento de un modelo de árbol de decisión.

val dt = new DecisionTreeClassifier().setLabelCol("indexedLabel").setFeaturesCol("indexedFeatures")

// Convertir etiquetas indexadas a etiquetas originales.

val labelConverter = new IndexToString().setInputCol("prediction").setOutputCol("predictedLabel").setLabels(labelIndexer.labels)

// Indexadores de cadena y árbol en una tubería.

val pipeline = new Pipeline().setStages(Array(labelIndexer, featureIndexer, dt, labelConverter))

// Modelo de tren. Esto también ejecuta los indexadores.

val model = pipeline.fit(trainingData)

// Hacer predicciones.

val predictions = model.transform(testData)

// Seleccione filas de ejemplo para mostrar.

predictions.select("predictedLabel", "label", "features").show(5)

// Seleccione (predicción, etiqueta verdadera) y calcule el error de prueba.

val evaluator = new MulticlassClassificationEvaluator().setLabelCol("indexedLabel").setPredictionCol("prediction").setMetricName("accuracy")

val accuracy = evaluator.evaluate(predictions)

//imprimimos los test error

println(s"Test Error = ${(1.0 - accuracy)}")

// creamos una variable llamda treeModel el cual va a ejecutar el modelo DecisionTreeClassificationModel

val treeModel = model.stages(2).asInstanceOf[DecisionTreeClassificationModel]

//se imprime elmodelo anteriormente creado

println(s"Learned classification tree model:\n ${treeModel.toDebugString}")

spark.stop()

}

}

// Resultados

