

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

ENERO - JUNIO 2020

INGENIERÍA INFORMÁTICA

MATERIA

DATOS MASIVOS

CATEDRÁTICO:

JOSE CHRISTIAN ROMERO HERNANDEZ

UNIDAD 2

# NAIVE BAYES

ALUMNO

Betzibeth Magdalena Hernandez Marin

López Valencia Luis Daniel

//Importamos las liberrias a utilizar

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

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

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

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

import org.apache.spark.ml.linalg.Vectors

//Cargamos nuestro dataset

val data = spark.read.format("csv").option("inferSchema","true").option("header","true").csv("iris.csv")

//Se limpia nuestra base de datos para eliminar valores erroneos o nulos

val Clean = data.na.drop()

//Muestra nombre de las columnas

Clean.columns

//Muestra la estructura del dataframe

Clean.printSchema()

//Muestra las 5 primeras columnas

Clean.show(5)

//Da información general sobre los datos del dataframe

Clean.describe().show

// Generamos un vector donde se almacenaran las caracteristicas del dataset a evaluar y se guardan mediante la nueva columna features

val assembler = new VectorAssembler().setInputCols(Array("sepal\_length","sepal\_width","petal\_length","petal\_width")).setOutputCol("features")

// Transformamos los datos utlizando nuestro dataset

val featureSet = assembler.transform(Clean)

featureSet.show()

//Transformamos los valores string de la columna species a datos numericos para poder utilizarlo

val labelIndexer = new StringIndexer().setInputCol("species").setOutputCol("label")

val dataindex = labelIndexer.fit(featureSet).transform(featureSet)

//muestra el indice de species en columna llamada label

dataindex.show()

// Se dividen los datos en datos de entrenamiento y datos de prueba 60% para entrenamiento y 40% para prueba

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

//toma el valor del 60% de los datos

val train = splits(0)

//toma el valor del 40% de los datos

val test = splits(1)

//Establecemos las capas de nuestra red neuronal 4 de entrada, 5 y 4 capa intermedia o capa oculta y salida de 3.

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

// Hacemos el entrenamiento de datos aplicando nuestro algoritmo multilayerPerceptron

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

//se entrena el modelo con los datos asignados

val model = trainer.fit(train)

//se testea con el modelo ya entrenado

val result = model.transform(test)

// seleccionamos prediction y label para guardarlos en la variable predictionandlabels

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

//mostramos el contenido de la variable

predictionAndLabels.show()

//Se calcula la eficacia del modelo

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

//Imprimimos el resultado de la predicción.

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