  
第12章 大數據分析範例－鳶尾花分類



### 讀入資料檔

|  |  |
| --- | --- |
| 指令 | 說明 |
| 進入Ubuntu環境 | |
| start-all.sh | 啟動hadoop，若已經啟動，則不需要再執行此指令 |
| cd ~ | 回到家目錄 |
| wget http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data | 下載鳶尾花資料集 |
| hdfs dfs -mkdir /iris | 在hdfs上建一目錄 |
| hdfs dfs -put ~/iris.data /iris | 將資料集上傳hdfs |
| hdfs dfs -ls /iris | 查看hdfs資料 |
| 進入spark-shell環境 | |
| spark-shell | 進入spark-shell |
| val lines = sc.textFile("hdfs://master:9000/iris/iris.data").filter(x=> !x.isEmpty).map(x=>x.split(","))  lines.count()  lines.first | 讀入文本資料，去除空列，以逗號分割 |
| case class IrisData(sepalLength: Double, sepalWidth: Double, petalLength: Double, petalWidth: Double, species: String) | 建立IrisData類別 |
| val irisDF = lines.map(x=>IrisData(x(0).toDouble,x(1).toDouble,x(2).toDouble,x(3).toDouble,x(4))).toDF() | 將讀入的資料轉成IrisData資料框格式 |
| irisDF.printSchema | irisDF的資料框綱要 |
| irisDF.show | 顯示前 20 筆資料 |
| irisDF.count | 列印筆數 |
| 建立 label 欄位 | |
| import org.apache.spark.ml.feature.StringIndexer | 匯入必要的程式庫 |
| val strInder = new StringIndexer().setInputCol("species").setOutputCol("label").fit(irisDF) | 重編species欄位為數值索引 |
| strInder.transform(irisDF).show | 放入irisDF資料集show出來 |
| strInder.transform(irisDF).orderBy("label").groupBy("label").count.show | 統計各品種筆數 |
| strInder.labels | 查看原來品種資料（按索引順序） |
| 建立features 欄位 | |
| import org.apache.spark.ml.feature.VectorAssembler  val vecAssemer = new VectorAssembler().setInputCols(Array("sepalLength", "sepalWidth", "petalLength", "petalWidth")).setOutputCol("features") | 將所有花瓣和花萼資料併入向量欄位 |
| vecAssemer.transform(irisDF).show | 先用 irisDF 資料集來轉換並查看 |
| 品種索引數值轉換回字串品種名稱 | |
| import org.apache.spark.ml.feature.IndexToString  val indToSpecies = new IndexToString().setInputCol("prediction").setOutputCol("predSpecies").setLabels(strInder.labels) | 品種索引數值轉換回字串品種名稱 |

### 切割資料集

|  |  |
| --- | --- |
| 指令 | 說明 |
| val Array(irisTrainSet, irisTestSet) = irisDF.randomSplit(Array(0.8, 0.2)) | 切割訓練和測試資料集 |
| irisTrainSet.count  irisTestSet.count  irisTrainSet.groupBy("species").count.show  irisTestSet.groupBy("species").count.show | 查看分割後資料集 |

### 模式設定與訓練

|  |  |
| --- | --- |
| 指令 | 說明 |
|  模式設定 | |
| import org.apache.spark.ml.classification.{RandomForestClassifier, RandomForestClassificationModel} | 匯入必要程式庫 |
| val rfc = new RandomForestClassifier().setImpurity("gini").setMaxDepth(3).setNumTrees(5).setMaxBins(5) | 建立隨機森林分類器物件 |
| 模式訓練 | |
| import org.apache.spark.ml.Pipeline  val pipeline = new Pipeline().setStages(Array(strInder, vecAssemer, rfc, indToSpecies)) | 建立管線 |
| val pipeLineModel = pipeline.fit(irisTrainSet) | 開始訓練 |

### 使用評價器（evaluator）計算準確度

|  |  |
| --- | --- |
| 指令 | 說明 |
| val trainPred = pipeLineModel.transform(irisTrainSet)  val testPred = pipeLineModel.transform(irisTestSet) | 計算預測值 |
| trainPred.select("label", "features", "prediction", "predSpecies").show(10)  testPred.select("label", "features", "prediction", "predSpecies").show(10) | 查看實際值和預測值 |
| import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator  val evaluator = new MulticlassClassificationEvaluator().setLabelCol("label").setPredictionCol("prediction").setMetricName("precision") | 計算準確度的物件 |
| evaluator.evaluate(trainPred)  evaluator.evaluate(testPred) | 計算準確度 |

### 交叉驗證

|  |  |
| --- | --- |
| 指令 | 說明 |
| import org.apache.spark.ml.tuning.ParamGridBuilder  val param = new ParamGridBuilder().addGrid(rfc.impurity, Array("gini", "entropy")).addGrid(rfc.maxDepth, Array(3,5)).addGrid(rfc.numTrees, Array(5,7)).addGrid(rfc.maxBins, Array(5,10,20)).build() | 參數網格物件（各種超參數組合） |
| import org.apache.spark.ml.tuning.CrossValidator  val crossValidator = new CrossValidator().setEstimator(pipeline).setEstimatorParamMaps(param).setNumFolds(10).setEvaluator(evaluator) | 交叉驗證器物件 |
| val crossValidatorModel = crossValidator.fit(irisTrainSet) | 進行交叉驗證 |
| val bestModel = crossValidatorModel.bestModel  val bestTrainPred = bestModel.transform(irisTrainSet)  val bestTestPred = bestModel.transform(irisTestSet) | 計算預測值 |
| evaluator.evaluate(bestTrainPred)  evaluator.evaluate(bestTestPred) | 計算準確度 |
| crossValidatorModel.getEstimatorParamMaps  crossValidatorModel.avgMetrics  val bestParam=crossValidatorModel.getEstimatorParamMaps.zip(crossValidatorModel.avgMetrics).maxBy(\_.\_2).\_1  println("最佳組合是：\n"+bestParam) | 找出最佳組合超參數 |

### 使用最佳模式來預測新值

|  |  |
| --- | --- |
| 指令 | 說明 |
| irisDF.describe("sepalLength", "sepalWidth", "petalLength", "petalWidth").show | 計算資料集統計值 |
| val newFeatures=sc.makeRDD(Array(Array(7.9,4.4,6.9,2.5),Array(4.3,2,1,0.1)))  val newIrisDF = newFeatures.map(a =>IrisData(a(0), a(1), a(2), a(3), "")).toDF | 建立最大和最小鳶尾花數值資料框 |
| val newPred = bestModel.transform(newIrisDF) | 進行預測 |
| newPred.select("features","predSpecies").show | 顯示出預測的品種 |