**巨量資料分析與應用 期末報告**

基本資料

班級：商資三甲 分組編號：16

組長：學號C109193104姓名鄭怡芳

組員：學號C109193110姓名張珈薰

題目

汽車數據品牌分類

--資料分析目的說明--

|  |
| --- |
| 本組利用car.csv，將汽車參數，如油耗(mpg)、汽缸(cylinders)、發動機尺寸(cubicinches)、最大功率(hp)、汽車重量磅(weightlbs)、最快加速到60公里/時(time-to-60)，等欄位分析資料為哪一款汽車品牌(brand)。 |

資料集描述

資料集個數：1 總筆數：262

資料集1名稱：car1 資料集1筆數：262

檔名1：cars1.csv

來源：https://www.kaggle.com/datasets/abineshkumark/carsdata

|  |  |
| --- | --- |
| **欄位名稱** | **欄位描述（包括編碼格式）** |
| **油耗(mpg)** | 每加倫汽油能跑的英里數(Double) |
| **汽缸(cylinders)** | 讓汽車活塞運作的容器(Double) |
| **發動機尺寸(cubicinches)** | 汽車的引擎尺寸(Double)，立方英尺 |
| **最大功率(hp)** | 最大可以出多少馬力(Double) |
| **汽車重量磅(weightlbs)** | 汽車的實際重量(Double) |
| **最快加速到60公里/時(time-to-60)** | 每秒最快加速到60公里/時(Double) |
| **製造年份(year)** | 1971年~1983年的製造年份(Double) |
| **汽車品牌(brand)** | 品牌地區(String)，例如: 美規(US.)、歐規(Europe.)、日規(Japan.) |

分析策略說明

|  |
| --- |
| 說明通過哪些步驟或流程？使用哪些資料集、哪些欄位、用何種方法來分析？  使用多元分類(Multiclass Classification)。   1. 進入Ubuntu作業系統，並啟動hadoop。 2. 開啟firefox，輸入資料集網址，下載car.csv資料集。 3. 將資料集拉進家目錄。 4. 並將car.csv內的五筆含空值資料及第一行刪除。 5. 再刪除製造年份(year)一整列欄位。 6. 將資料集名稱改為cars1.csv。 7. 建立hdfs資料夾，名稱為car1。 8. 將資料集上傳至car1資料夾。 9. 進入spark-shell環境。 10. 讀入資料，留下x不為空值資料，並用,切割出資料。 11. 查看資料是否有成功讀入，確認讀入筆數為256筆，並讀取第一筆資料。 12. 建立carIndex為分類索引號，抽出第六個欄位，去重複排序，並配上索引號映射。 13. 將索引前後對調並做Map，建立分類名稱對照表。 14. 匯入LabeledPoint格式和特徵值相關程式庫。 15. 每一行資料品種及特徵抓進去，並建立LabeledPoint類別。 16. 切割資料集，將資料用 7:2:1 的比例，分成訓練集、驗證集與測試集，並用168隨機分割。 17. 匯入隨機森林分類的程式庫。 18. 設定隨機森林分類參數，使用gini公式，最大深度為3，建立5棵樹及5個分支。 19. 開始訓練分類器，拿carTrainSet訓練3種品牌(分類數目)。 20. 設定一個getPrecision方法，計算出模型的準確度。 21. 開始使用驗證集與訓練集測試，獲得預測準確率為0.83及0.79。 22. 寫一個利用隨機森林分類參數，所訓練模型準確度的方法。 23. 用var變數存取每一次更高的最佳準確度。 24. 開始跑迴圈，計算所有超參數組合的模式和準確度。 25. 最後得出最佳預測準確率為0.89及測試資料集準確率0.82。 26. 輸入最大及最小的預測資料，並輸出預測結果是否正確。 27. 查看隨機森林的決策樹(7棵樹權重)。 |

程式碼

* 請使用Consolas字型

|  |  |
| --- | --- |
| **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  https://www.kaggle.com/datasets/abineshkumark/carsdata | 下載car資料集 |
| hdfs dfs -mkdir /car1 | 在hdfs上建一目錄 |
| hdfs dfs -put ~/cars1.csv /car1 | 將資料集上傳hdfs |
| hdfs dfs -ls /car1 | 查看hdfs資料 |
| **進入spark-shell環境** | |
| spark-shell | 進入spark-shell |
| val lines = sc.textFile("hdfs://master:9000/car1/cars1.csv").filter(x=> !x.isEmpty).map(x=>x.split(",")) | 讀入資料 |
| lines.count() | 查看總共幾筆資料 |
| lines.first | 查看第一筆資料 |
| val carIndex=lines.map(line=>line(6)).distinct.sortBy(x=>x).zipWithIndex.collectAsMap | 建立分類索引號 |
| carIndex | 查看分類索引號 |
| val carNameMap=carIndex.toArray.map(x=>(x.\_2,x.\_1)).toMap | 建立分類名稱對照表 |
| carNameMap | 查看分類名稱對照表 |
| import org.apache.spark.mllib.regression.LabeledPoint | 匯入LabeledPoint類別程式庫 |
| import org.apache.spark.mllib.linalg.Vectors | 匯入LabeledPoint類別程式庫 |
| val carLP=lines.map(line=>new LabeledPoint(carIndex(line.last).toDouble, Vectors.dense(line.init.map(x=>x.toDouble)))) | 建立LabeledPoint類別 |
| carLP.first | 查看LabeledPoint類別第一筆資料 |
| carLP.first.label | 查看LabeledPoint類別第一個標籤分類 |
| carLP.first.features | 查看LabeledPoint類別第一筆特徵 |
| val Array(carTrainSet, carValidSet, carTestSet) = carLP.randomSplit(Array(0.7, 0.2, 0.1),168) | 資料集隨機分割 |
| carTrainSet.count | 查看carTrainSet資料筆數 |
| carValidSet.count | 查看carValidSet資料筆數 |
| carTestSet.count | 查看carTestSet資料筆數 |
| import org.apache.spark.mllib.tree.RandomForest | 匯入隨機森林分類成市庫 |
| val impurity="gini" | 設定隨機森林分類參數 |
| val maxDepth=3 | 設定隨機森林分類參數 |
| val numTrees=5 | 設定隨機森林分類參數 |
| val maxBins=5 | 設定隨機森林分類參數 |
| val carModel = RandomForest.trainClassifier(carTrainSet, 3, Map[Int, Int](), numTrees, "auto" , impurity, maxDepth, maxBins) | 開始訓練 |
| def getPrecision(model:org.apache.spark.mllib.tree.model.RandomForestModel, dataset:org.apache.spark.rdd.RDD[org.apache.spark.mllib.regression.LabeledPoint])={  val precision = dataset.map { x =>  val pred = model.predict(x.features)  if (x.label == pred) 1.0 else 0.0  }.mean()  precision  } | 計算模式預測準確度 |
| println("預測準確率＝" + getPrecision(carModel, carValidSet)) | 輸出carValidSet預測準確度 |
| println("預測準確率＝" + getPrecision(carModel, carTrainSet)) | 輸出carTrainSet預測準確度 |
| println("預測準確率＝" + getPrecision(carModel, carTestSet)) | 輸出carTestSet預測準確度 |
| def modelAndPrecision(numTrees:Int, impurity:String, maxDepth:Int, maxBins:Int)={  val carModel = RandomForest.trainClassifier(carTrainSet, 3, Map[Int, Int](), numTrees, "auto" , impurity, maxDepth, maxBins)  val precision=getPrecision(carModel, carValidSet)  println(f"Precision:$precision%2.4f <== numTrees:$numTrees%2d, impurity:$impurity%8s, maxDepth:$maxDepth%2d, maxBins:$maxBins%2d")  (precision, carModel)  } | 模式訓練與計算準確度副程式 |
| var bestPres=Double.MinValue | 開始跑迴圈，計算所有超參數組合的模式和準確度 |
| var bestModel:org.apache.spark.mllib.tree.model.RandomForestModel=null  for(maxBins<-Array(5,10,20); maxDepth<-Array(3,5); numTrees<-Array(5,7); impurity<-Array("gini", "entropy")) {  val (tempPres, tempModel)=modelAndPrecision(numTrees, impurity, maxDepth, maxBins)  if(tempPres > bestPres) {  bestPres=tempPres  bestModel=tempModel  }  } | 開始跑迴圈，計算所有超參數組合的模式和準確度 |
| println("最佳預測準確率＝" + bestPres) | 計算模式準確度 |
| println("測試資料集準確率＝" + getPrecision(bestModel, carTestSet)) | 計算測試資料集的準確度 |
| val newFeatures=Array(Vectors.dense(31.9,4.0,89.0,71.0,1925.0,14.0),Vectors.dense(18.0,4.0,121.0,112.0,2933.0,15.0)) | 輸入欲預測的資料 |
| newFeatures.foreach(x=>println(x + " 預測為： " +carNameMap(bestModel.predict(x).toInt))) | 輸出預測結果 |
| println("隨機森林決策樹" + bestModel.toDebugString) | 查看隨機森林的決策樹s |

HardCopy執行畫面

方法：選好要拷貝的執行結果畫面，按Alt-PrintScreen，出現gnome-screenshot畫面後，點選「複製到剪貼簿(O)」，再到本檔案中貼上。

注意：執行結果超過一個畫面是必然的，所以上下面畫面間的資料應該重複，而不可以缺少。

<<<以下為範例畫面，可以刪除>>>





































