**Distracted Driver Detection Analysis**

1. **參與者資訊:**

101308016 風管四 周俊川

101701016 應數四 杜靖愷

103304042 統計二 劉乃慈

1. **摘要:**

大數據分析如今紅紅火火，但真的要討論到如何實現的時候，往往卻不是那麼地簡單，數據科學與大數據分析這門課介紹給我們知道Spark這個分析工具，我們非常幸運可以在起飛的時候有機會了解這個工具。

以下將說明我們如何運用Spark的ML來實現Multilayer Perceptron Neural Network，這之前我們也會示範利用Python可以有效地處理數據。

1. **情境說明**

許多車輛意外原始於駕駛者的分心。他們可能一邊駕駛一邊滑著演書，或是一邊轉頭跟後座的朋友談話。根據統計資料顯示，高達五分之一的車禍原因是駕駛者的分心。為了改善這個狀況，我們利用監視器中拍攝到的畫面，試著分析出駕駛者現在的駕駛狀況，他是否正專注於其他事物，是否正在從事其他事情......，並希望藉由分析結果警告有高事故危險性的駕駛者，讓他們能夠提高警覺。

**Input：**

模型訓練集為如以下所示的jpg檔22424張，已知其駕駛狀態分類



**Output：**預測的駕駛者狀態，分為下列10種狀況

c0: 安全駕駛者(safe driving)

c1: 正在用右手打字駕駛者(texting – right)

c2: 正在用右手講電話的駕駛者(talking on the phone – right)

c3: 正在用左手打字駕駛者(texting – left)

c4: 正在用左手講電話的駕駛者(talking on the phone – left)

c5: 正在操作收音機的駕駛者(operating the radio)

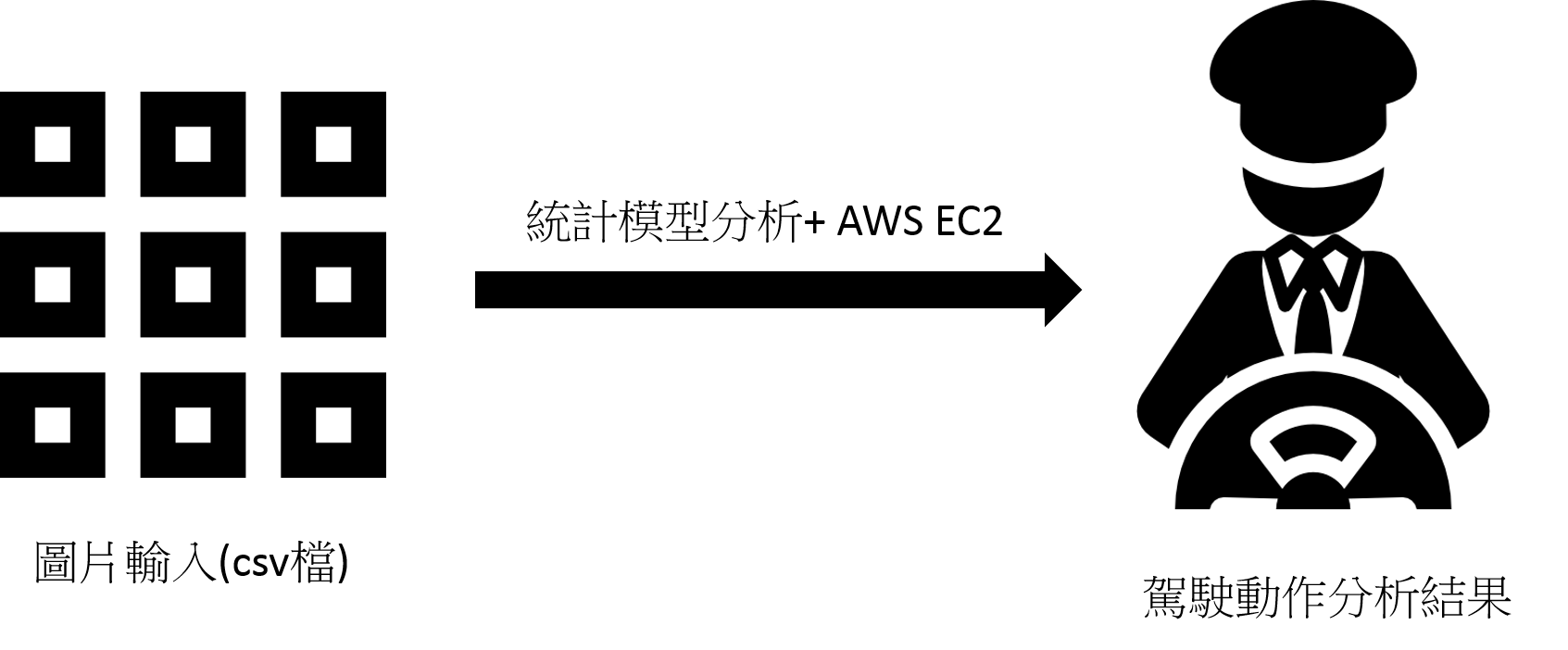
c6: 正在喝飲料的駕駛者(drinking)

c7: 正在向後座拿東西的駕駛者(reaching behind)

c8: 正在化妝的駕駛者(hair and makeup)

c9: 正在談話駕駛者(talking to passenger)

**情境表示圖像:**

****

1. **資料集來源與處理**
   1. 資料來源、

kaggle 網站( <https://www.kaggle.com/c/state-farm-distracted-driver-detection>)

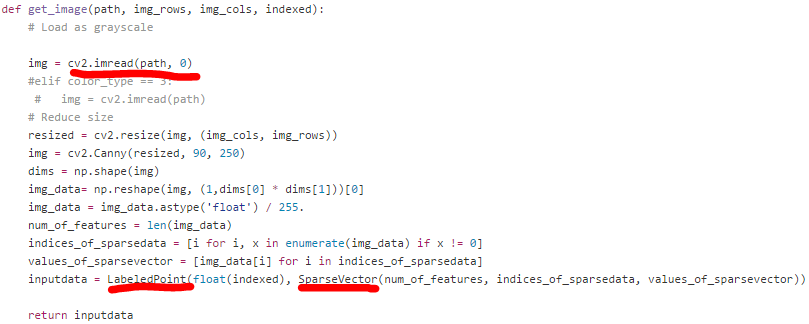
* 1. 使用工具
     1. Pyspark
     2. Ipython
     3. Github
     4. Python套件：joblib, opencv, numpy
  2. 資料前處理

為了讓資料量減少，以便於後續分析執行速度，在執行前，我們對資料進行了三個動作的處理:彩色轉黑白、pixel的減少、轉成 sparsevector型式，之後我們利用python套件joblib來進行平行化讀取資料的動作，以下就各點進行說明。

資料輸入時為彩色格式，且分別有(紅，綠，藍) 三原色的數值。為了使資料量減少，直覺上來說，圖片是彩色或黑白的並不會影響分析結果，因此我們使用了python裡的opencv套件讀取圖片，然後把圖片轉成greyscale的模式。

另外我們將照片pixel由原本的480 x 640，降為48 x 64，因為1個pixel對我們來說就是一個feature，這樣可以讓我們由307200個features（如果用RGB的圖片格式則更多!）直接降到3072個。

最後，我們將每張圖片的pixel 轉為1 x 3072 python中的numpy array，再轉為spark Mlilb 中的其中一個資料格式sparsevector，表示為SparseVector(3072,{11:1.0,……})，代表總共有3072個feature，並且第11個feature值為1(黑)。因此在降低資料量的同時，也有助於後續匯入spark中進行分析。



上面的程式碼是資料前處理的實現。

由於圖片有2萬多筆，如果直接載入需要花10分鐘，用joblib的話將所有圖片經過上述轉化後完全載入只要50秒。



紅色框起來的這一段程式碼是平行化的實現，這一段幫我們節省 了不少時間

* 1. 特徵值的選取策略

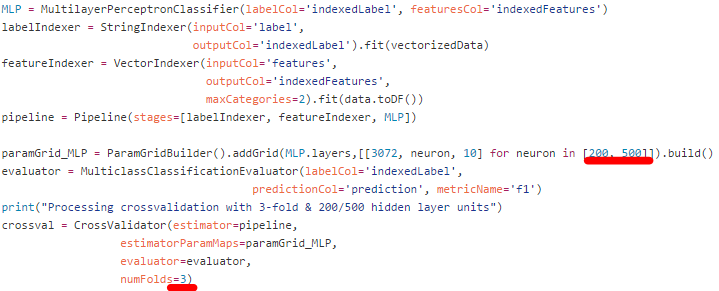
特徵值方面，我們採用的策略就是把圖片降維成3072，因為這個用肉眼最極限可以判斷圖片中司機駕駛狀態的pixel大小。

* 1. 分析模型演算法的挑選

我們採用的模型是Multilayer perceptron neural network（MLP）,要注意雖然MLP的名字有perceptron的字眼但使用的學習函數仍然是sigmoid function。

在spark的ML中其實用於classification的模型本來就不像scikit-learn的套件那麼豐富，而且我們的題目是multiclass以及我們使用的語言是pyspark，所以我們可以使用的模型就剩下decision tree, random forest以及MLP，所以我們的方式就是先隨機抽取1000張圖片，然後用這1000張圖片去split 800張來訓練，200張來做validation，decision tree只有約58%的準確率，random forest更糟糕只有22%（我們tree用了100棵），而MLP就已經有78%，所以我們就果斷選擇MLP作為我們分析的模型。

* 1. 建模、參數與超級參的優化處理

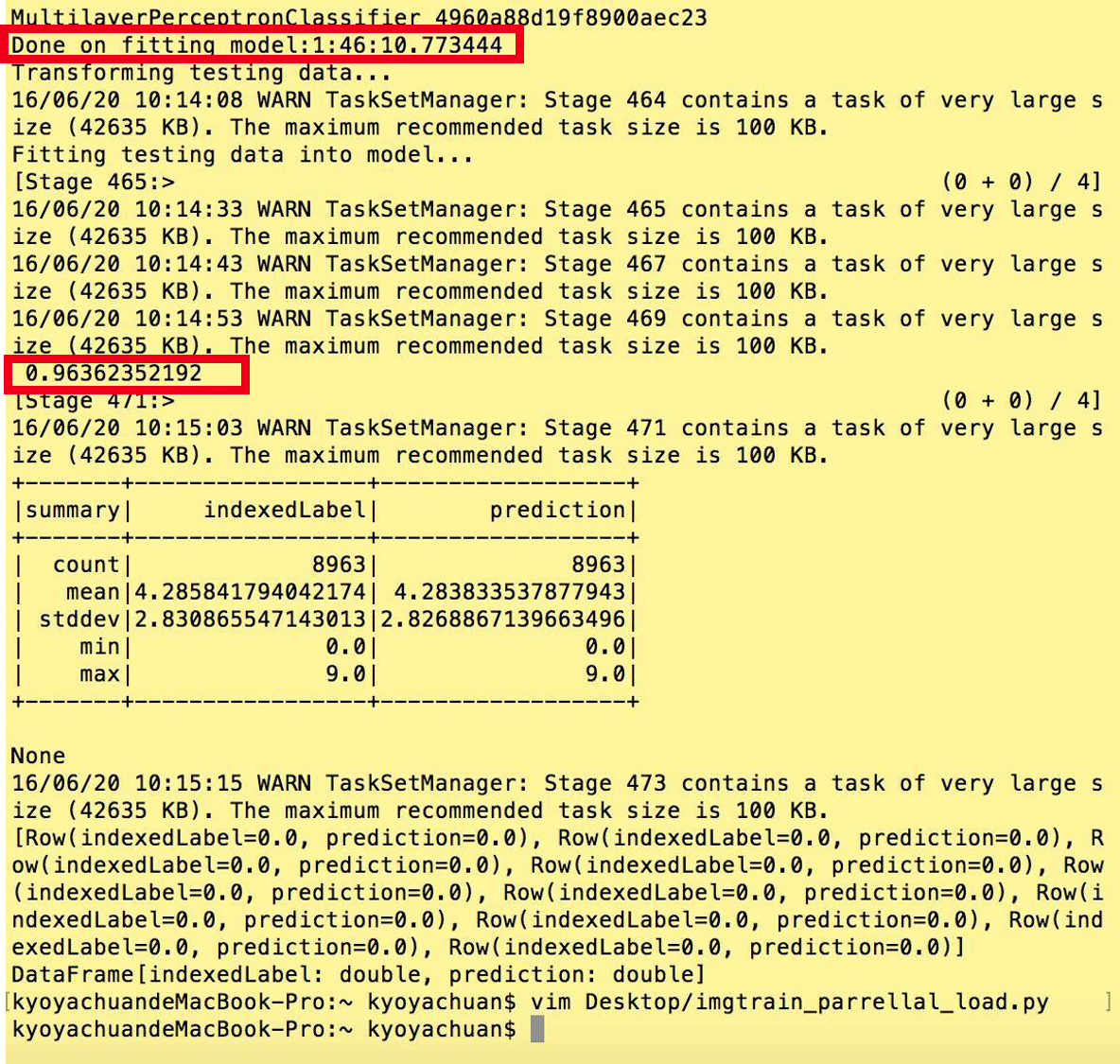
****

在最後的結果上，我們利用spark的ml中的Pipeline, ParamGridBuilder, MulticlassClassificationEvaluator和Crossvalidator來做hypertuning。

我們一開始做crossvalidation的時候是都用3個layers，3072的input unit，10個output unit以及20,50,100,200的hidden unit去做tuining，最後的結果發現200的表現最好，之後再用200和500個hidden unit去跑，而crossvalidation用k-fold，k等於3。

1. **評量結果**

下圖為執行結果，以最佳模型進行最後的資料測試後，我們總共執行了1個小時46分10秒，準確率高達96%

****

1. **應用的系統平台說明與Spark函式庫的使用**

在本次的專案中，我們使用的工具有PySpark、Slack、Ipython以及github。我們利用PySpark中的Mllib和Ml進行資料分析，並利用Slack 進行組內溝通，Ipython作為主要的編寫環境，最後使用github 進行專案程序管理。

參考：<http://spark.apache.org/docs/latest/ml-classification-regression.html>

1. **系統效能（含記憶體使用量與運算效能）**

在最後的執行分析的部分，我們用spark submit的方式去執行.py，給予spark的內存是8gb（即使6gb也會遇到heapspace的問題）。

1. **分工及Github**

|  |  |
| --- | --- |
| 情境設計 | 乃慈 |
| 程式撰寫 | 靖愷（前處理,ml），俊川（spark submit執行細節,平行化） |
| 上台報告、書面報告 | 乃慈，靖愷 |

以下是我們的github鏈接

<https://github.com/kyoyachuan/datascienceProject>