# 資料探勘期末報告—第九組

鄭璟翰
國立中山大學資工系 113 級
b093040003@student.nsysu.edu.tw

郭晏涵
國立中山大學資工系 113 級
b093040024@student.nsysu.edu.tw

## 1. 簡介

隨著大數據的時代來臨，各種感應裝置普及，資料的蒐集已經也比以往還要方便許多，這也導致了資料的數量與維度都大幅上升，而醫學上的數據也是如此。然而過往的分類技術往往需要先透過訓練資料所產生的分類器，這樣的分類器在面對訓練資料中所沒有的標籤時，並沒有辦法正確分類。因此這次期末報告就採用「先分類後分群」的方法來進行：先依照訓練資料訓練出分類器，對於分類器不確定的資料先分類到unknown，蒐集所有unknown的資料後再進行分群，當作為分類結果。

這次期末報告我們所選擇的資料集是Gene Expression Cancer RNA-Seq Dataset，每一筆資料總共有大約兩萬筆有關基因的資料；訓練資料中有三種分類結果，而測試資料當中多加入了兩種分類結果。分析的方法則是透過python實作兩種分類器K-Nearest Neighbor、Random Forest(透過scikit-learn完成)以及兩種分群方法K-Means、DBSCAN來分析資料集，因此總共會產生六種分類結果，包括兩種分類器加上兩種分類器乘上兩種分群方法。

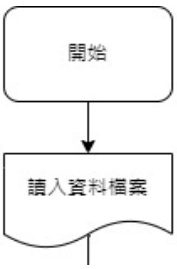
流程圖如右**圖1.**。

## 2. 資料整理及輸出

不論在進行分類分群之前，同樣需要對資料進行處理都需要事先進行資料整理、輸入等步驟，我們這個部分整理於 tool.py 這個檔案之中。當需要進行與分類分群無關的處理時，就會透過呼叫 tool.py 來獲得需要的資料以及輸出分類結果、統計畫面。這個部分大多和期中報告類似，因此這裡不再重複；惟有統計及輸出結果的部分稍有不同：因為這次選用的資料集分類結果有五種，而非陰性陽性，沒辦法再使用 Recall 和 Precision 分析，只能使用 Accuracy 來呈現分類結果。

## 3. 分類與分群結果

透過 2 資料整理，得到各筆資料的內容(data)以及罹患何種癌症(label)，依據所得到的資料進行分類與分群的演算法後，就可以得到最後結果。



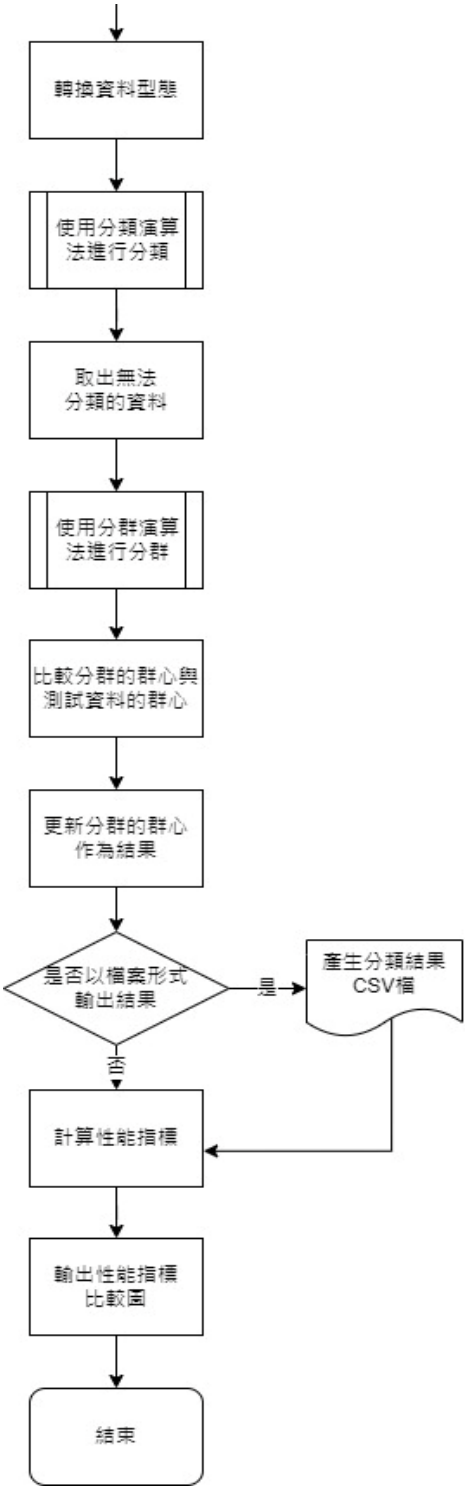




**圖1. 程式流程圖**

### 3.1 分類器(KNN、Random Forest)

分類器的部分採用期中報告中的 KNN 和 Random Forest，選擇這兩者的原因是這兩種的分類器容易取得無法分類的資料。

KNN 分類器是藉由找出距離最小 K 筆資料的方法是依照距離分類，因此可以就由直接判斷和其他的距離來判斷是否為無法分類的資料。藉由計算訓練資料中所有點的距離取平均得知任兩點的平均距離約為 200，因此我們設定將最小 K 筆資料距離總和若大於 200 * K + 100，則判斷為未知的分類項目；測試時以 K=9 作為範例，所以前 9 筆的距離總和大於 1900 則設定為未知資料。使用 KNN 進行訓練的缺點為需要進行的計算量過大，因為一筆資料中有兩萬多維，且每筆測試資料要和每一筆訓練資料進行運算，導致整體的計算量非常大。依此次的測試與訓練集測量，進行一次 KNN 分類大約需要花一分鐘。

Random Forest 分類器是從訓練資料中選取子資料讓 Decision Tree 來訓練，最後由各個 Decision Tree 來進行投票。因為有投票的過程，各筆資料的分類結果就可以機率的方式展現，因此就可以將機率過低的結果視為未知分類。藉由 scikit-learn 實現的話則可以依照 predict_pro 所回傳的機率表來來進行分類。若該筆資料之最高機率的分類項目沒有超過 0.7，則將該筆資料設定為未知的分類項目。

另外期末報告沒有選到的分類器有Decision Tree、Linear classifier(SGD)、MLP。沒有選用的Decision Tree的原因Decision Tree是橡樹的結構一樣從根到葉來決定分類結果，因此沒辦法判斷單一筆資料他的分類情況；MLP則是因為訓練的資料維度過高，導致其中的參數受到過多的變數影響而導致準確率低；Linear classifier則是因為線性分類導致準確率非常低，因此不採用。

其他關於各分類器的詳細敘述請見期中報告。

### 3.2 分群(Clustering)

進行完第一階段分類的訓練後，接著對未知的項目進行分群的演算法。分群是一種對於沒有標籤的資料進行分門別類的演算法，屬於非監督式學習，因為不需要其他輔助的標籤，因此對於未知分類標籤的資料非常適合。這裡總共實作兩種分群的演算法，分別是 K-Means 以及 DBSCAN 後面將詳細介紹這兩種演算法。

完成分群演算法後，我們可以得到各筆資料被分類在哪一個群心中。接著要求使用者輸入未知的分類標籤，藉由統計測試資料的標籤與各個分群重疊的情況，來判斷各個分群實際上對應到哪一個類別，最後再將該群更新為該項類別，即可完成全部分群的部分。

**A. K-Means**

K-Means 是一種將資料依照平均距離分割成 K 類別分群方法，其中 K 是 hyper parameter。一般情況下，很難得知 K 要設定為多少才能得知結果，但在這次的測試的資料集中，我們可以得知未知的分類標籤總共有兩個，因此可以設定 K=2。

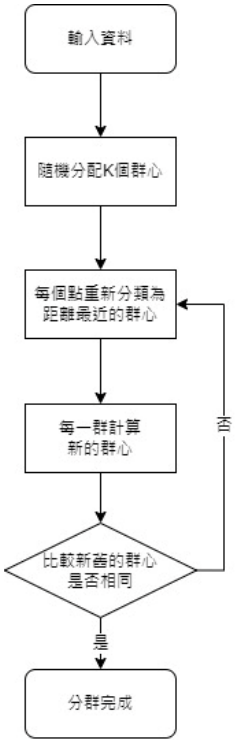



圖2. K-Means流程圖

```
KMeans(K, data):
  for i from 1 to K:
    center.push(data[random number])
  end-for
  while Dif(last-center, center):
    for i from 1 to #data:
      label[i]=minDist(center, data[i])
    end-for
    last-center = center
    for i from 1 to #label:
      center[i]=average(data[labe[i]])
    end-for
  end-while
  return center
end-process
```

表1. K-Means虛擬碼

K-Means 演算法詳細過程如**圖 2.**流程圖及**表 1.** 虛擬碼所示，可以發現直到群心收斂時，都需要不斷的更新群心，因此 K-Means 並不適合資料呈現環狀分佈的情況；而實際上執行這次測試時的資料集時，大約只要三到四次的計算就可以收斂群心。需要特別注意隨機分配 K 個群心時若選到相同的點，可能會有一個分類的結果為 0，導致分類結果錯誤。

另外從以上的流程圖及虛擬碼可以得知 K-Means 是將所有點都納入計算新群心的考量中，因此 K-Means 適合當各個未知標籤之間距離較為大、相同標籤的資料較為聚集的資料分佈；若資料的分佈當中雜音(不規則的資料點)很多的話，K-Means 很容易受到影響。

**B. DBSCAN**

DBSCAN，全名為 Density-based spatial clustering of applications with noise，是一種基於密度來將資料進行分群的演算法。給定 esp(高密度資料的範圍)以及 minPts(組成高密度區的最小點數量)後，DBSCAN 能把周圍的點組合成同一群，並標記出位於低密度區域的局外點(稱為噪音)。

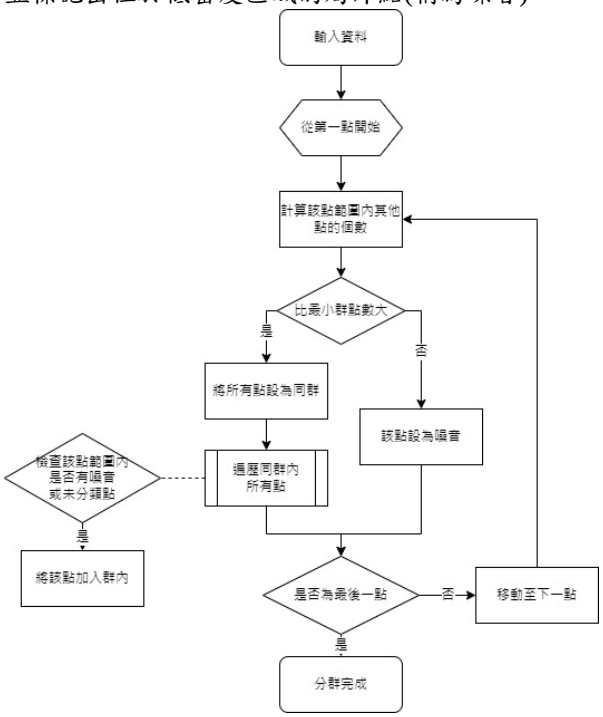



圖3. DBSCAN流程圖

```
DBSCAN(data, eps, minPts,count = 0):
  for point in Data:
    if point.label == unclassifiy:
      continue
    end-if
    neighbors=findNei(data,point,eps)
    if #neighbors < minPts:
      point.label = noise
      continue
    end-if
    point.label=count

    for n in neighbors:
      if point.label==noise:
        point.label=count
      end-if
      if point.label!=unclassifiy:
        continue
      end-if
      point.label=count
      subnei=findNei(data,point,eps)
      if len(subnei) >= minPts:
        for i in neighbors:
          data[i].label=count
        end-for
      end-if
    end-for

    count++
  end-for
  return list(point.label)
end-process
```

表2. DBSCAN虛擬碼

DBSCAN 演算法詳細過程如**圖 3.**流程圖及**表 2.**虛擬碼所示，和 K-Means 相比，可以發現 DBSCAN 並不需要事先傳入分類的個數，只需要先設定組成 minPts 和 esp 就可以依照密度進行分類，因為在先前分類器 KNN 時計算過資料之間的平均距離為 200，所以這裡將 esp 設定為 200，minPts 則設為 5。另外從以上的流程圖及虛擬碼可以得知對於周圍沒有其他資料的資料點設定為噪音，因此和 K-Means 相比 DBSCAN 適合資料集中的分佈，比較不受到噪音的影響。

## 4. 結果分析與結論

```
Input 1 label: COAD
Input 2 label: PRAD
KNN
            classifier: 32%
accuracy of Kmeans: 92%
            DBSCAN: 92%
Random Forest
            classifier: 34%
accuracy of Kmeans: 98%
            DBSCAN: 97%
Output result as a csv file?(Y/N) Y

KNN result.csv create!

RF result.csv create!
```

圖4. 執行結果

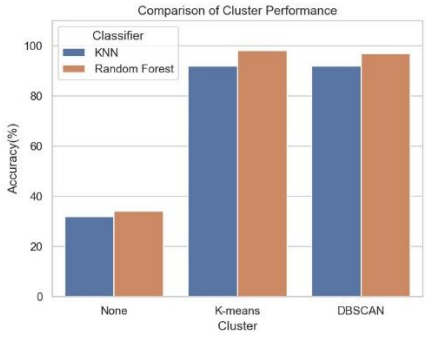



圖5. 分類分群結果

透過圖 4.及圖 5.的實驗結果可以發現：

(1)、在只有進行 classifier 的情況下，分類器的準確度都只有 1/3，因此可以推測約六成的測試資料都是訓練資料中所沒有的分類。

(2)、六種結果的準確度以 random forest 加上 K-Means 的準確率最高；但以時間複雜度而言，KNN 所要耗費的時間太多，所以最有效率的方法為 random forest 加上 clustering。

(3)、比較兩種分群 K-Means 的準確率稍微高一些，但我們認為是因為 DBSCAN 會將距離較遠的資料判定為噪音，才會導致準確率稍微降低。

## 參考文獻


[1]. Wikipedia DBSCAN
https://en.wikipedia.org/wiki/DBSCAN

[2]. Wikipedia k-means clustering
https://en.wikipedia.org/wiki/K-means_clustering