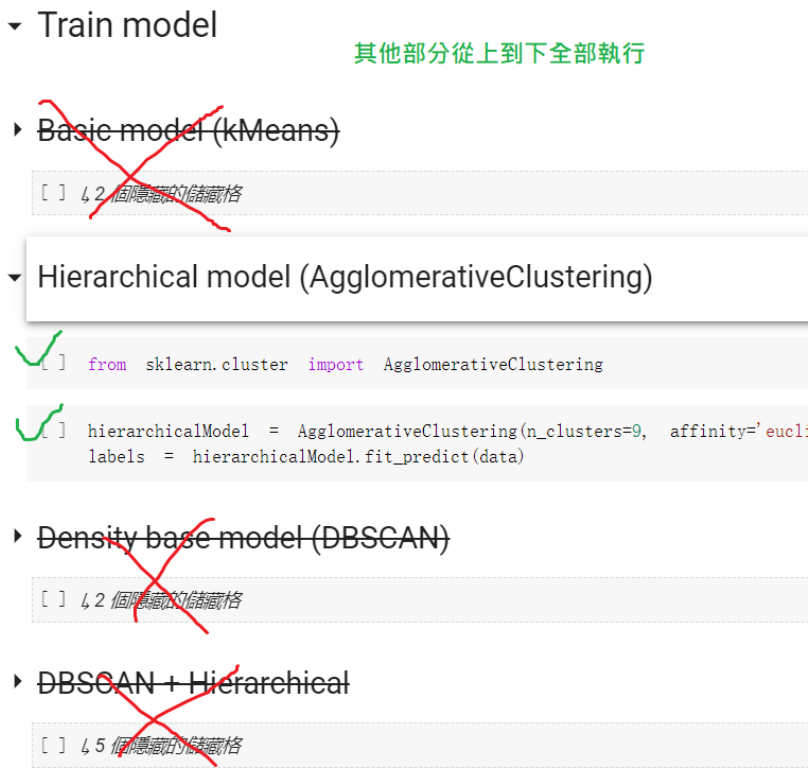
四資工三甲 B10715029 陳彥瑋

Data Science HW2

1. 執行程式

程式網址：
https://colab.research.google.com/drive/1mx4pEHMsoA3hvMuIVJ4WQfVriSX3-ucQ

除了標題被劃掉的部分之外，其餘執行（也就是 Train model 那邊只執行 Hierarchical model 的部分，其餘全部執行）。



2. 演算法簡介

Load file into colab workspace

```
[ ]  !gdown --id "1gLpMZ_sdjaSk6C5RkqQiLEVyvAD9GZv9" --output "data.csv"
     !gdown --id "1gUSSIntugJwDu_pAUdEI9GghRUSEMCbv" --output "test.csv"
```

```
Downloading...
From: https://drive.google.com/uc?id=1gLpMZ_sdjaSk6C5RkqQiLEVyvAD9GZv9
To: /content/data.csv
100% 370k/370k [00:00<00:00, 44.9MB/s]
Downloading...
From: https://drive.google.com/uc?id=1gUSSIntugJwDu_pAUdEI9GghRUSEMCbv
To: /content/test.csv
100% 5.49k/5.49k [00:00<00:00, 9.41MB/s]
```

```
[ ]  !ls
```

```
data.csv  sample_data  test.csv
```

一開始先從雲端硬碟載入資料到 colab，並且確認檔案存在。

Preprocessing

```
[ ]  import pandas as pd
     import numpy as np
```

```
[ ]  data = pd.read_csv("data.csv")
     data.head()
```

	id	feature1	feature2	feature3	feature4	feature5	feature6	feature7	featu
0	0	86	140	9	0.0	0.0	5.444444	4.768726	3.055
1	1	34	93	9	0.0	0.0	0.555556	0.272165	0.388
2	2	134	148	9	0.0	0.0	0.111111	0.172133	0.055
3	3	199	144	9	0.0	0.0	0.333333	0.516398	0.333
4	4	197	236	9	0.0	0.0	2.444444	6.829628	3.333

```
[ ]  from sklearn import preprocessing
```

```
[ ]  scaler = preprocessing.MinMaxScaler()
     scaler.fit(data)
     data = scaler.transform(data)
```

前處理的部分是把欄位做 z-score 標準化，把數值範圍壓縮到 0 ~ 1 之間，以避免因為某些欄位的值變化量較大而導致在算距離時會有較大影響力的問題。

Train model

~~Basic model (kMeans)~~

```
[ ] 4 個隱藏的儲藏格
```

Hierarchical model (AgglomerativeClustering)

```
[ ] from sklearn.cluster import AgglomerativeClustering
```

```
[ ] hierarchicalModel = AgglomerativeClustering(n_clusters=9, affinity='euclidean', memory="./model_cache/", linkage="ward")
    labels = hierarchicalModel.fit_predict(data)
```

~~Density base model (DBSCAN)~~

```
[ ] 4 個隱藏的儲藏格
```

~~DBSCAN + Hierarchical~~

```
[ ] 5 個隱藏的儲藏格
```

在模型的部分，我有實驗過各種的模型但效果最好的是 Hierarchical model，參數為所需 9 個 cluster、歐式距離計算、以要合併的 cluster 之間會有最小 variance 作為合併的選擇，各式模型和參數的實驗可以參考 3. 試驗 的部分。

在訓練完模型之後，我先找出每個資料點所屬的分類，並儲存在 labels 變數裡，以方便做資料點是否為同一群的判斷。

Apply model

```
[ ] cmp = pd.read_csv("test.csv")
```

```
[ ] result = []
    for i in range(cmp.shape[0]):
        x1 = cmp.iloc[i]["0"]
        x2 = cmp.iloc[i]["1"]

        result.append(labels[int(x1)]==labels[int(x2)] * 1)
```

這個部分是讀入 test.csv 檔案，並且根據檔案內要比較的資料點判斷是否為同一群，同一群則儲存為 1，否則為 0（Python 裡 True 和 False 在與數字做運算時可以視為數字的 1 和 0）。

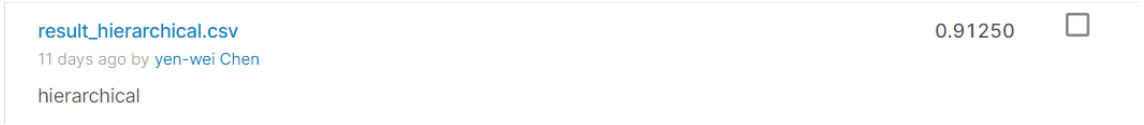
Store result

```
[ ]  with  open("result.csv",  'w')  as  fh:
         fh.write("index,ans"+"\n")
         for  i  in  range(cmp.shape[0]):
             fh.write(str(i)+","+str(int(result[i]))+"\n")
```

```
[ ]  from  google.colab  import  files
     files.download('result.csv')
```

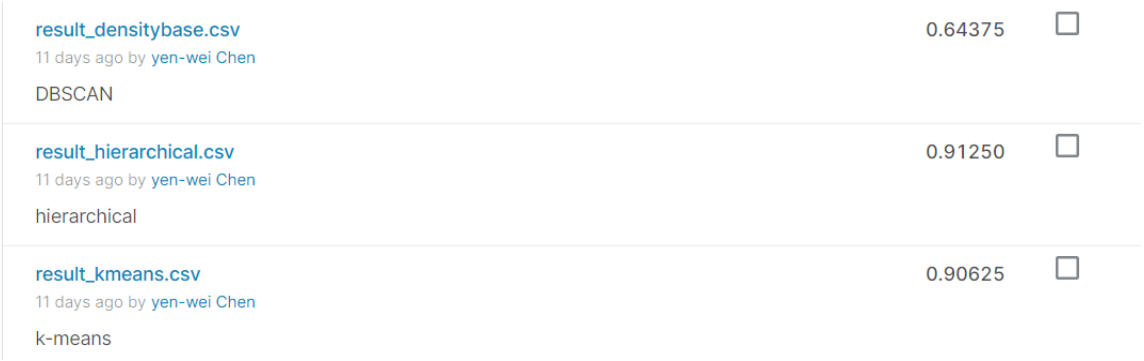
這邊的部分是把結果寫入 reult.csv，並且執行下載。

結果如下：



3. 試驗

一開始，我測試了上課有提到的 3 種不同類型的模型，結果如下。



可以看到 hierarchical 和 k-means 的結果還不錯，但是 DBSCAN 結果不太好。

我在這邊判斷有可能是因為 DBSCAN 的輸出結果會有 outlier 標籤的原因，導致在比較時雖然距離很遠，但是因為同樣是 outlier 的關係，所以會被分到同一個代表 outlier 的標籤 “-1”。因此，為了解決這個問題，我又額外設計了 DBSCAN 和 hierarchical base model 混和的方式，概念大概是用 DBSCAN 做初步的判斷，但是對於 outlier 則另外使用 hierarchical base model 的分類方法強制分類到大的 cluster 裡，程式如下：

~~DBSCAN + Hierarchical~~

```
[ ] from sklearn.cluster import AgglomerativeClustering
    from sklearn.cluster import DBSCAN
```

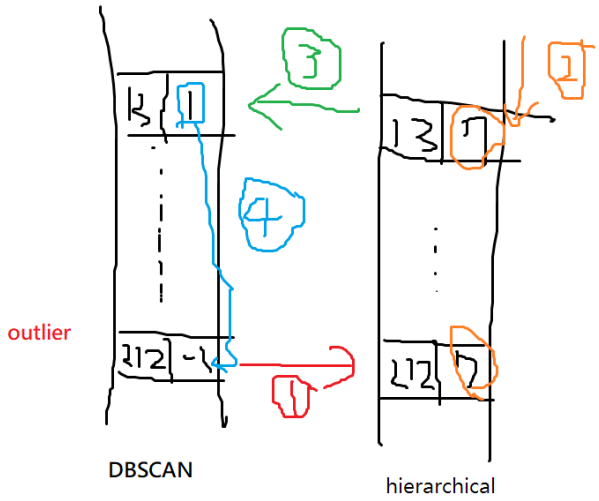
```
[ ] hierarchicalModel = AgglomerativeClustering(n_clusters=9, affinity='euclidean', memory="./model_cache/", linkage="ward")
    labels_sup = hierarchicalModel.fit_predict(data)
```

```
[ ] densityModel = DBSCAN(eps=0.5, min_samples=5, metric='euclidean')
    labels_main = densityModel.fit_predict(data)
```

```
assert (labels.shape[0] == labels_sup.shape[0])

labels = labels_main.copy()

for index in range(labels.shape[0]):
    if(labels[index] == -1):
        group_sup = labels_sup[index]

        same_group_i = 0
        for sup_index in range(labels_sup.shape[0]):
            if (labels_sup[sup_index] == group_sup and sup_index != index):
                same_group_i = sup_index
                break

        print(index, same_group_i)
        labels[index] = labels[same_group_i]
```

```
[ ] labels.tolist().count(-1)
```

labels_sup 是 hierarchical model 的標籤結果，labels_main 是 DBSCAN 的標籤結果。在重標記的程式部分首先先找出 outlier 的標記，並且找出這個資料點在 hierarchical model 的分類且儲存到 group_sup 變數中，接著再找到跟 group_sup 一樣分類的資料點的位置 same_group_i，最後就是抓取這個位置在 DBSCAN 的分類標籤並覆蓋 outlier 的分類。可以參考以下的圖：



至於為什麼要進行圖中 3 的轉換則是因為 DBSCAN 和 hierarchical model 的分類標籤可能不同，不能直接用 hierarchical model 的標籤來標記。

執行結果如下，其實即使做了這樣的轉換，結果仍然差不多，推測可能 outlier 標籤影響不大，反而只是 DBSCAN 不適合此作業任務，因此接下來放棄 DBSCAN 的改良：

result_dbscan_plus_hierarchical.csv 11 days ago by yen-wei Chen DBSCAN + Hierarchical model as support to classifier the outlier	0.64375	☐
result_densitybase.csv 11 days ago by yen-wei Chen DBSCAN	0.64375	☐

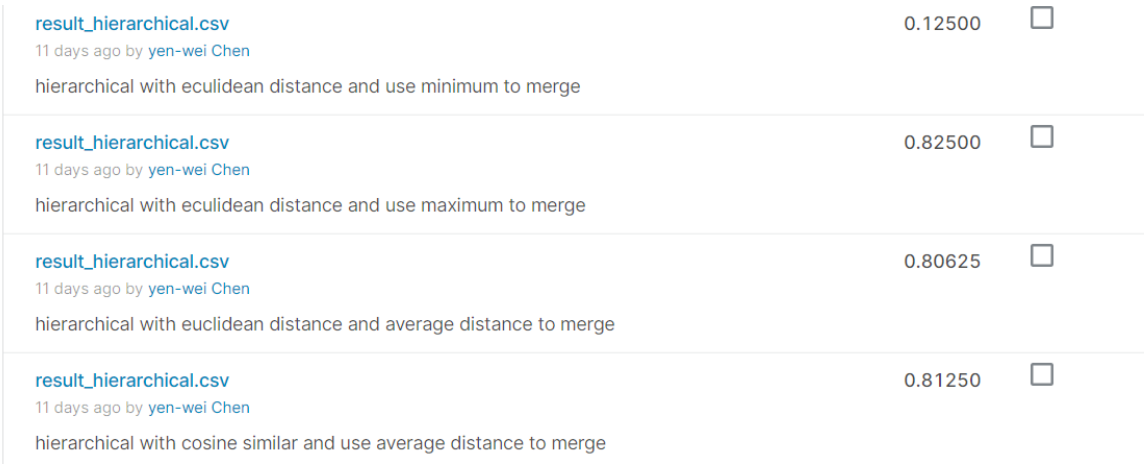
由於剩下二者之中 hierarchical model 的結果最佳，所以主要進行這個模型的改良。

首先我調整不同所需分類數量的影響（原始值是 9），發現無論是太高或太低對於模型的影響都是不良的（尤其是太低），最佳值大概是在 7 ~ 9 左右，推測太低會造成因為分類的選擇不多所以會勉強地把不相近的資料點分類在一起，而太高可能是因為種類太多而把相近的點分到不同的群。

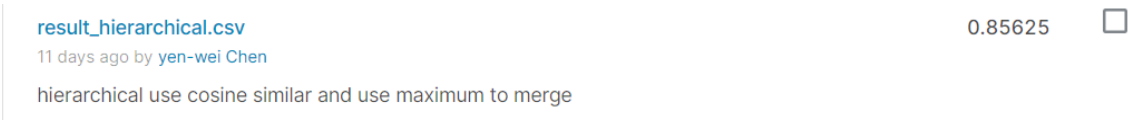
result_hierarchical.csv 11 days ago by yen-wei Chen hierarchical with 7 cluster	0.91250	☐
result_hierarchical.csv 11 days ago by yen-wei Chen hierarchical with 8 cluster	0.91250	☐
result_hierarchical.csv 11 days ago by yen-wei Chen hierarchical with 11 cluster	0.90625	☐
result_hierarchical.csv 11 days ago by yen-wei Chen hierarchical with 13 cluster	0.89375	☐
result_hierarchical.csv 11 days ago by yen-wei Chen hierarchical with 5 cluster	0.82500	☐

接著是調整不同的距離計算方式和整合方式（原始值是 euclidean + ward，ward 的部分是 2. 演算法簡介 – Train model 部分所說的最小 variance 的方式），發現在都是歐式距離計算下，準確率排行為 ward（0.91250，結果圖在演算法部分）> max（0.82500）> average（0.80625）> min（0.12500）。

而在都是以 average 為整合方式的情況下，可以發現 cosine similarity（0.81250）的計算方式會比歐式距離（0.80625）來的好。



接下來我試試看把兩個結果的最好的部分結合起來，看看能不能突破原本的模型，但是因為 cosine similarity 似乎沒有支援 ward，所以我挑選的是 cosine similarity + max 的方法，結果如下：



結果為 0.85625，仍然輸給原先的 歐式距離 + ward（0.91250），不過結果相較於其他的結合方式人人高出了不少。