

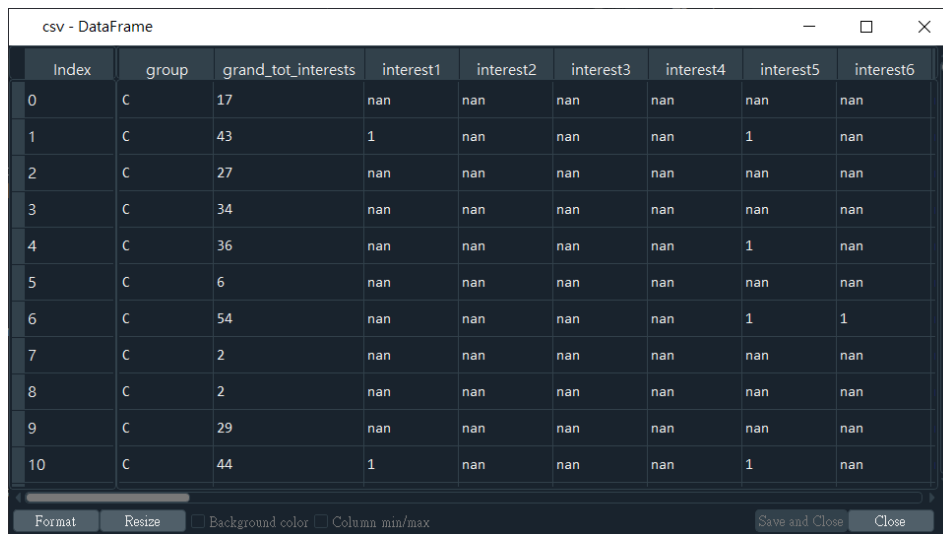
Introduction to Machine Learning

Homework 4 Dimensionality Reduction & Clustering

1. Data pre-processing :

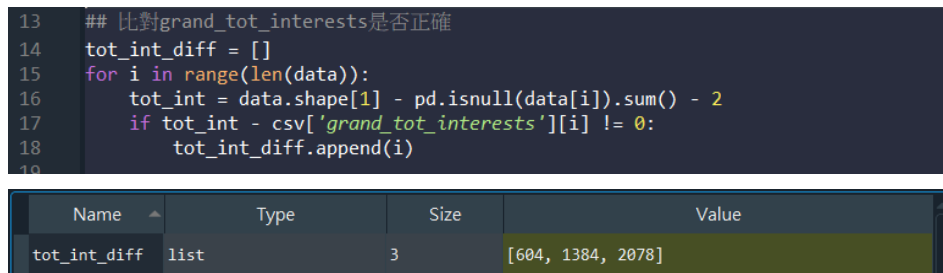
首先載入 csv 檔，部分內容顯示於圖一。接著整理資料是否正確，發現在 index 為 604、1384 及 2078 時出現錯誤，原因是相較於其他 index 的 interest 都是 1 或 nan，這三行卻是 2 跟 nan，因此造成在統計 grand_tot_interests 數量時出錯，程式碼與結果顯示於圖二。

圖一、



Index	group	grand_tot_interests	interest1	interest2	interest3	interest4	interest5	interest6
0	C	17	nan	nan	nan	nan	nan	nan
1	C	43	1	nan	nan	nan	1	nan
2	C	27	nan	nan	nan	nan	nan	nan
3	C	34	nan	nan	nan	nan	nan	nan
4	C	36	nan	nan	nan	nan	1	nan
5	C	6	nan	nan	nan	nan	nan	nan
6	C	54	nan	nan	nan	nan	1	1
7	C	2	nan	nan	nan	nan	nan	nan
8	C	2	nan	nan	nan	nan	nan	nan
9	C	29	nan	nan	nan	nan	nan	nan
10	C	44	1	nan	nan	nan	1	nan

圖二、



```
13 ## 比對grand_tot_interests是否正確
14 tot_int_diff = []
15 for i in range(len(data)):
16     tot_int = data.shape[1] - pd.isnull(data[i]).sum() - 2
17     if tot_int - csv['grand_tot_interests'][i] != 0:
18         tot_int_diff.append(i)
19
```

Name	Type	Size	Value
tot_int_diff	list	3	[604, 1384, 2078]

修改完 grand_tot_interests 之後，為了讓 data 能丟入 PCA、LDA 或 K-Means 等模型裡訓練，必須將資料內容全部轉為數值，且為了排除不同資料彼此間數據的落差，必須再將資料做 normalization。

首先將所有的 nan 以數值 0 代替，再來找出 group 的所有種類，發現共有 C、P、R、I 四種 group，並從 1 開始 label 到 4，接著將資料做標準化，由於所有 interest 的值都是 0 或 1，只剩 grand_tot_interests 的可能數值介於 0 到 217 之間(因為總共有 217 種 interest)，因此只須對它進行 normalize，也就是全部除以最大的可能值 217，其意義代表在所有 interest 中，有喜歡的 interest 的比例。過程和結果分別顯示於圖三、圖四。

圖三、

```

20  ## nan填0，其他填1
21  data_int = np.array(data[:,2:])
22  for i in range(data_int.shape[0]):
23      for j in range(data_int.shape[1]):
24          if str(data_int[i,j]) == str(np.nan):
25              data_int[i,j] = 0
26          else:
27              data_int[i,j] = 1
28
34  ## 找出group種類並label
35  group_type = []
36  group_type.append(csv['group'][0])
37  for i in range(len(csv)):
38      if group_type[-1] != csv['group'][i]:
39          group_type.append(csv['group'][i])
40  group_map = {"C": 1, "P": 2, "R": 3, "I": 4}
41  data_group = np.array(csv['group'].map(group_map))
42

```

Name	Type	Size	Value
group_type	list	4	['C', 'P', 'R', 'I']

圖四、

csv1 - DataFrame								
Index	group	grand_tot_interests	interest1	interest2	interest3	interest4	interest5	interest6
0	1	0.078341	0	0	0	0	0	0
1	1	0.198157	1	0	0	0	1	0
2	1	0.124424	0	0	0	0	0	0
3	1	0.156682	0	0	0	0	0	0
4	1	0.165899	0	0	0	0	1	0
5	1	0.0276498	0	0	0	0	0	0
6	1	0.248848	0	0	0	0	1	1
7	1	0.00921659	0	0	0	0	0	0
8	1	0.00921659	0	0	0	0	0	0
9	1	0.133641	0	0	0	0	0	0
10	1	0.202765	1	0	0	0	1	0

另外，我整理了所有 interest 在 6340 個 sample 中的出現機率，發現大部分都小於 10%，也就是大部分 sample 在這些 interest 的值都是 0，那麼這些資料對於後續的 Dimensionality reduction 和 Clustering 就沒有幫助了，因此我將出現機率小於 10% 和大於 90% 的 interest 丟掉(大於 90% 代表大部分的值都是 1，所以也沒有用處)。程式碼以及過程顯示於圖五、圖六，而圖七為最後前處理完的資料。

圖五、

```

51  ## 去掉太多或太少 NaN 的 interest
52  int_count = pd.isnull(csv).sum()
53  int_nan_prob = int_count / len(csv)
54  int_useful = int_nan_prob[(0.1 < int_nan_prob) & (int_nan_prob < 0.9)]
55  j = 0
56  csv2 = pd.concat([csv1_group, csv1_tot_int], axis=1)
57  csv2.columns = csv.columns[:2]
58  for i in range(len(csv1)):
59      if csv1.columns[i] == int_useful.index[j]:
60          csv2 = pd.concat([csv2, csv1[csv1.columns[i]]], axis=1)
61          j = j + 1
62          if j >= len(int_useful):
63              break

```

圖六、

每項 interest 中出現 nan 的機率	機率大於 0.1 且小於 0.9 的 interest																																																				
<div>int_nan_prob - Series</div> <table> <tr><th>Index</th><th>0</th></tr> <tr><td>interest1</td><td>0.843375</td></tr> <tr><td>interest2</td><td>0.999842</td></tr> <tr><td>interest3</td><td>0.994479</td></tr> <tr><td>interest4</td><td>0.996057</td></tr> <tr><td>interest5</td><td>0.874132</td></tr> <tr><td>interest6</td><td>0.464669</td></tr> <tr><td>interest7</td><td>0.999842</td></tr> <tr><td>interest8</td><td>0.985331</td></tr> <tr><td>interest9</td><td>0.947476</td></tr> <tr><td>interest10</td><td>0.999842</td></tr> <tr><td>interest11</td><td>0.972397</td></tr> <tr><td>interest12</td><td>0.287224</td></tr> </table>	Index	0	interest1	0.843375	interest2	0.999842	interest3	0.994479	interest4	0.996057	interest5	0.874132	interest6	0.464669	interest7	0.999842	interest8	0.985331	interest9	0.947476	interest10	0.999842	interest11	0.972397	interest12	0.287224	<div>int_useful - Series</div> <table> <tr><th>Index</th><th>0</th></tr> <tr><td>interest1</td><td>0.843375</td></tr> <tr><td>interest5</td><td>0.874132</td></tr> <tr><td>interest6</td><td>0.464669</td></tr> <tr><td>interest12</td><td>0.287224</td></tr> <tr><td>interest15</td><td>0.701577</td></tr> <tr><td>interest16</td><td>0.288486</td></tr> <tr><td>interest21</td><td>0.354259</td></tr> <tr><td>interest40</td><td>0.718612</td></tr> <tr><td>interest41</td><td>0.897792</td></tr> <tr><td>interest43</td><td>0.852208</td></tr> <tr><td>interest44</td><td>0.785174</td></tr> <tr><td>interest47</td><td>0.116877</td></tr> </table>	Index	0	interest1	0.843375	interest5	0.874132	interest6	0.464669	interest12	0.287224	interest15	0.701577	interest16	0.288486	interest21	0.354259	interest40	0.718612	interest41	0.897792	interest43	0.852208	interest44	0.785174	interest47	0.116877
Index	0																																																				
interest1	0.843375																																																				
interest2	0.999842																																																				
interest3	0.994479																																																				
interest4	0.996057																																																				
interest5	0.874132																																																				
interest6	0.464669																																																				
interest7	0.999842																																																				
interest8	0.985331																																																				
interest9	0.947476																																																				
interest10	0.999842																																																				
interest11	0.972397																																																				
interest12	0.287224																																																				
Index	0																																																				
interest1	0.843375																																																				
interest5	0.874132																																																				
interest6	0.464669																																																				
interest12	0.287224																																																				
interest15	0.701577																																																				
interest16	0.288486																																																				
interest21	0.354259																																																				
interest40	0.718612																																																				
interest41	0.897792																																																				
interest43	0.852208																																																				
interest44	0.785174																																																				
interest47	0.116877																																																				

圖七、

csv2 - DataFrame

Index	group	grand_tot_interests	interest1	interest5	interest6	interest12	interest15	interest16
0	1	0.078341	0	0	0	1	0	1
1	1	0.198157	1	1	0	1	0	0
2	1	0.124424	0	0	0	1	0	1
3	1	0.156682	0	0	0	1	0	1
4	1	0.165899	0	1	0	1	0	1
5	1	0.0276498	0	0	0	0	0	0
6	1	0.248848	0	1	1	1	1	1
7	1	0.00921659	0	0	0	0	0	0
8	1	0.00921659	0	0	0	0	0	0
9	1	0.133641	0	0	0	1	0	1
10	1	0.202765	1	1	0	1	0	1

2. Dimensionality reduction :

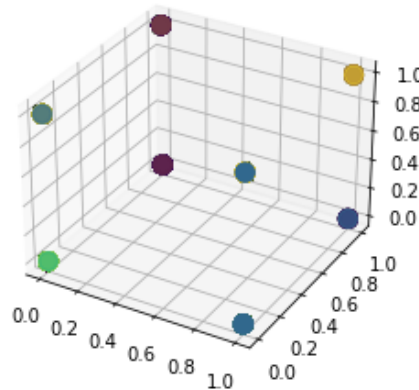
降維主要分為特徵選擇(feature selection)與特徵提取(feature extraction)兩種方法。首先我用特徵選擇的 subset selection，卻發現完全無法從三維空間中分辨出資料的種類與分布，原因在於資料過於單純，數值不是 0 就是 1，使得在三維空間中，可能的點就只有 8 個(2^3)，因此由圖八可看出所有 6340 個樣本皆落在這八個點當中，無法看出資料的分布。

特徵選擇行不通之後，我換使用特徵提取的 PCA，圖九、圖十為使用 PCA 方法將資料投影至三維空間、二維平面。圖中可看出 PCA 比起 subset selection 在可視化資料的分布上好了許多，但各種 group 的資料卻都混在一起，我認為可能是因為 PCA 不論資料的 label，只是盲目的找出最大變異量的投影向量，但這樣的投影對資料的

區分作用並不大，反而可能使資料點擠在一起無法區分，因此 PCA 對於此 dataset 並不是有效的降維方法。

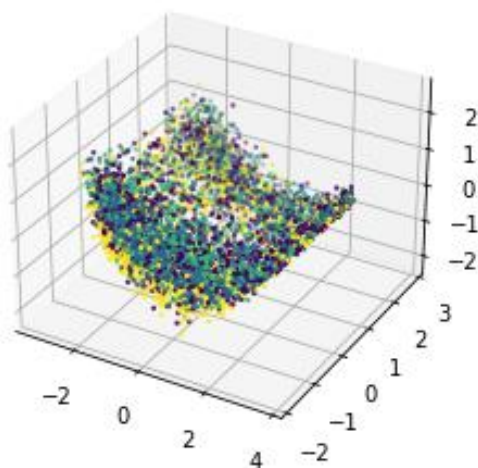
圖八、

Subset selection (component = 3)



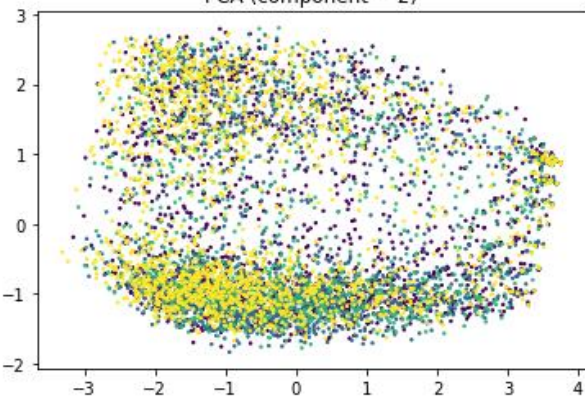
圖九、

PCA (component = 3)



圖十、

PCA (component = 2)



因此最後，我選擇使用特徵提取的 LDA 來做降維，程式碼顯示於圖十一。LDA 是一種監督式學習，它在投影後會使不同 group 的分散量拉到最大，相較於 PCA，LDA 沒有丟掉 group 的資訊，因此降維後能讓不同的 group 分離，有助於後續 clustering 的進行。

LDA 降維後最多可生成 label-1 維的子空間，也就是三維，資料投影至三維、二維、一維的結果分別顯示於圖十二、圖十三、圖十四。可看出 LDA 在三維空間中的分布(圖十二)比起 PCA(圖九)好上許多，能輕易分辨出不同 group 以及彼此之間的距離；而投影到二維平面後，LDA(圖十三)也優於 PCA(圖十)；然而當 LDA 降到一維時，會因為丟掉過多維度的資訊，而導致全部資料又擠在一塊(圖十四)。

圖中紫色點代表 group C、藍色代表 group P、綠色代表 group R、黃色代表 group I，可看出 LDA 在三維及二維的投影下呈現相似的趨勢，group C 和 group I 較能從群體中被分辨出來、而 group P 和 group R 則混在一起不易分辨，因此從圖中的分布情形可大致推測出，後續的 clustering 可能會比較適合分成 3 到 4 群。

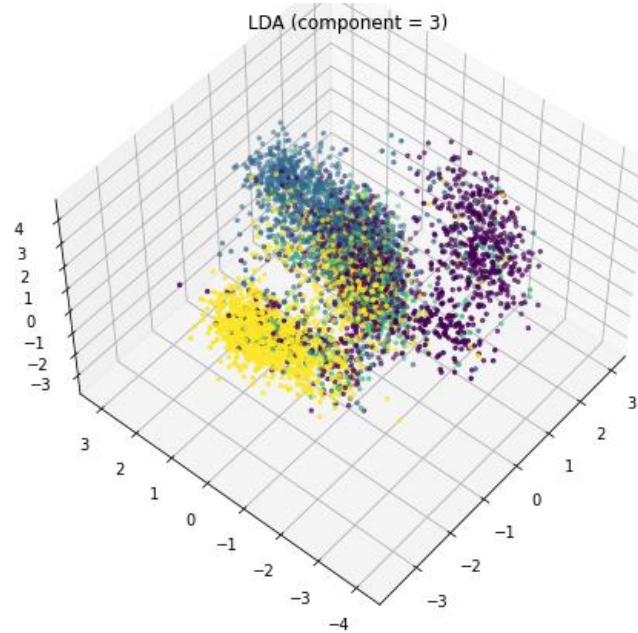
圖十一、

```

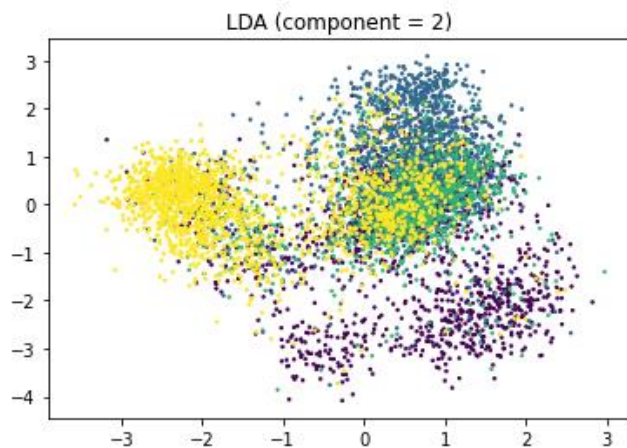
112 LDA_3 = LinearDiscriminantAnalysis(n_components=3)
113 LDA_3_feature = LDA_3.fit_transform(X, y)
114 fig = plt.figure(figsize=(6,6))
115 ax = fig.add_subplot(111, projection='3d')
116 ax.set_title('LDA (component = 3)')
117 ax.scatter(LDA_3_feature[:,0], LDA_3_feature[:,1], LDA_3_feature[:,2], s=6, c=y)
118 ax.view_init(elev=50,azim=220)
119 ax.dist = 8
120 plt.show()
121
122 #%% LDA (component=2)
123 LDA_2 = LinearDiscriminantAnalysis(n_components=2)
124 LDA_2_feature = LDA_2.fit_transform(X, y)
125 plt.figure()
126 plt.title('LDA (component = 2)')
127 plt.scatter(LDA_2_feature[:,0], LDA_2_feature[:,1], s=2, c=y)
128 plt.show()
129
130 #%% LDA (component=1)
131 LDA_1 = LinearDiscriminantAnalysis(n_components=1)
132 LDA_1_feature = LDA_1.fit_transform(X, y)
133 plt.figure()
134 plt.title('LDA (component = 1)')
135 plt.scatter(LDA_1_feature[:,0], np.zeros(len(data1)), c=data_group)
136 plt.show()

```

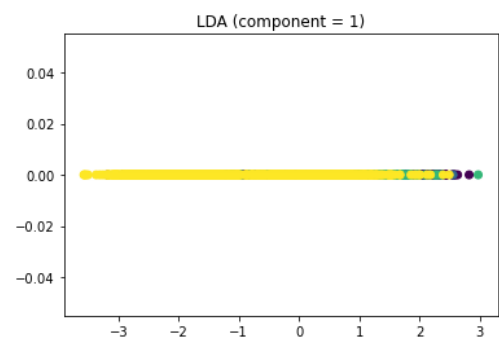
圖十二、



圖十三、



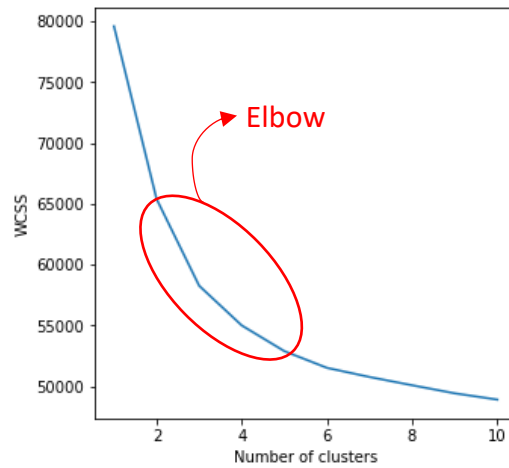
圖十四、



3. Clustering：

我選擇 K-means 方法做 clustering，原因是簡單快速。首先我透過計算 WCSS 並作圖，得知群數 K 大概落在 2 到 5 之間，如圖十五所示，而這也和前面 LDA 降維後的分布圖所顯示出的可能群數符合。

圖十五、

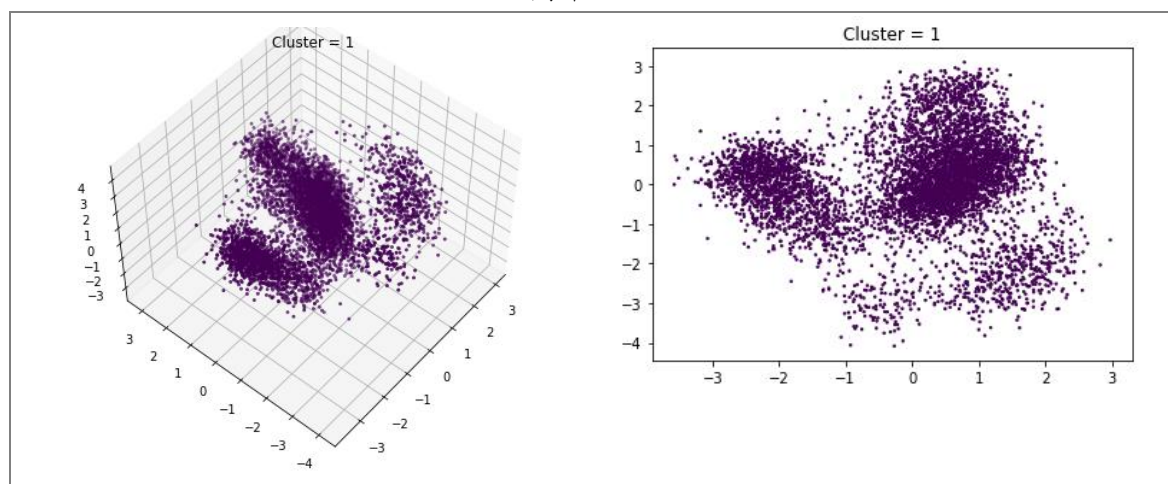


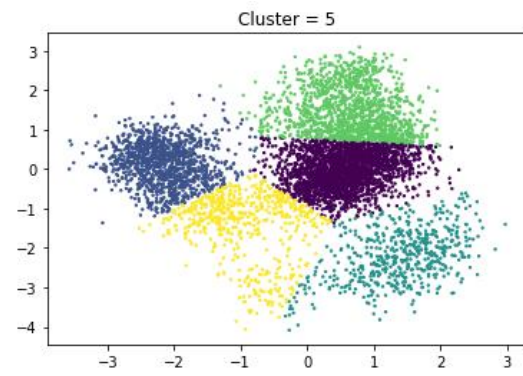
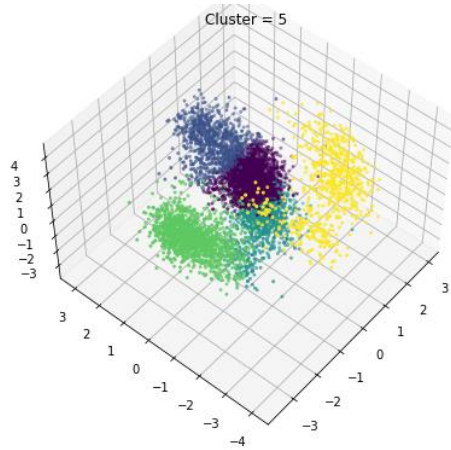
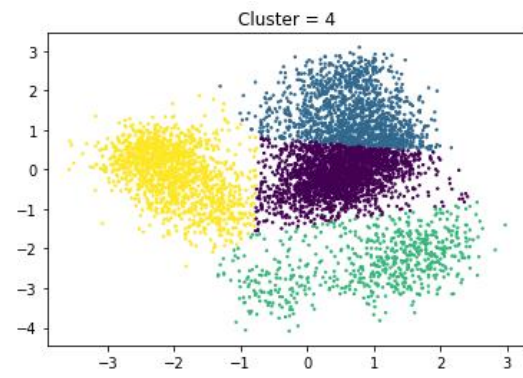
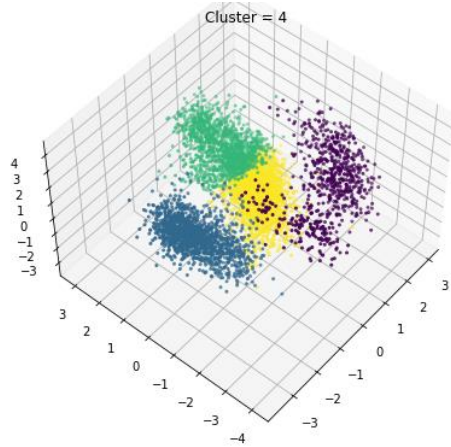
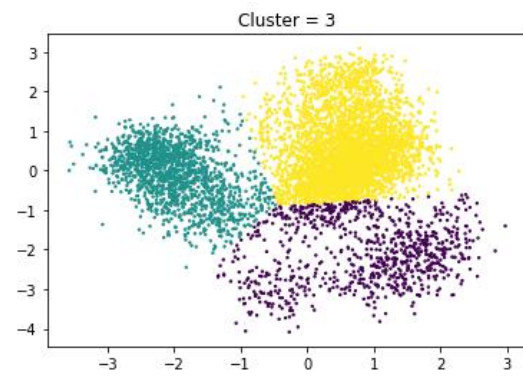
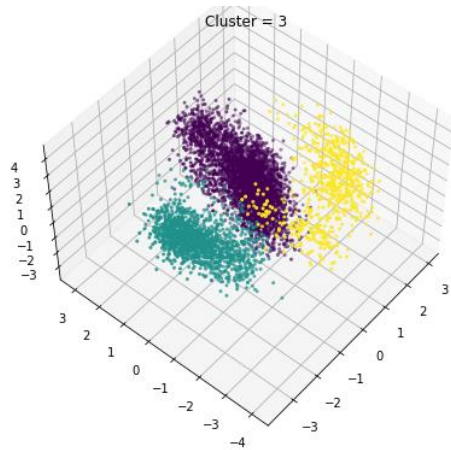
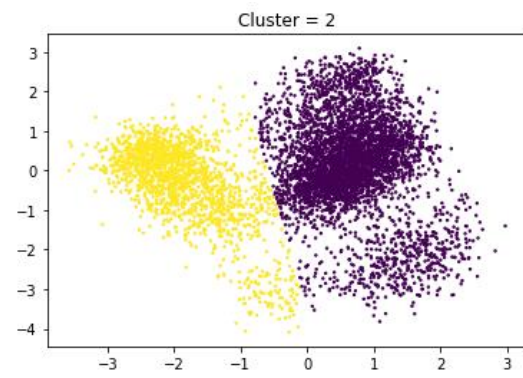
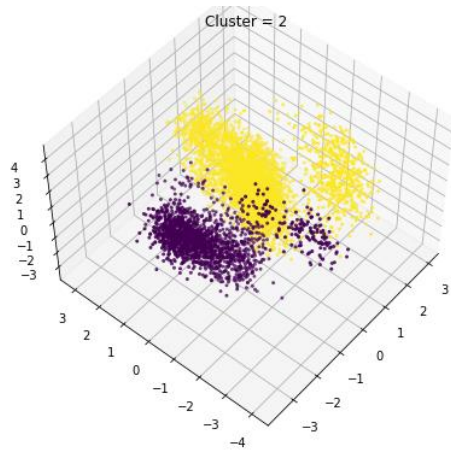
接著我對 LDA 降維後的資料(已拿掉 label 資訊)作 K-means 聚類，K 值從 1 到 5，程式碼顯示於圖十六，結果顯示於圖十七，從中可發現在 K 為 2 到 4 時，分群分的最合理。

圖十六、

```
158  ### K-means (3D)
159  for i in range(1,6):
160      fig = plt.figure()
161      ax = Axes3D(fig, elev=50, azim=220)
162      labels = KMeans(n_clusters=i, init='k-means++').fit(LDA_3_feature).labels_
163      ax.scatter(LDA_3_feature[:,0], LDA_3_feature[:,1], LDA_3_feature[:,2], c=labels, s=4)
164      ax.set_title('Cluster = %d' % (i))
165      ax.dist = 8
166      plt.show()
167
168  ### K-means (2D)
169  for i in range(1,6):
170      plt.figure()
171      labels = KMeans(n_clusters=i, init='k-means++').fit(LDA_2_feature).labels_
172      plt.scatter(LDA_2_feature[:,0], LDA_2_feature[:,1], c=labels, s=2)
173      plt.title('Cluster = %d' % (i))
174      plt.show()
175
```

圖十七、

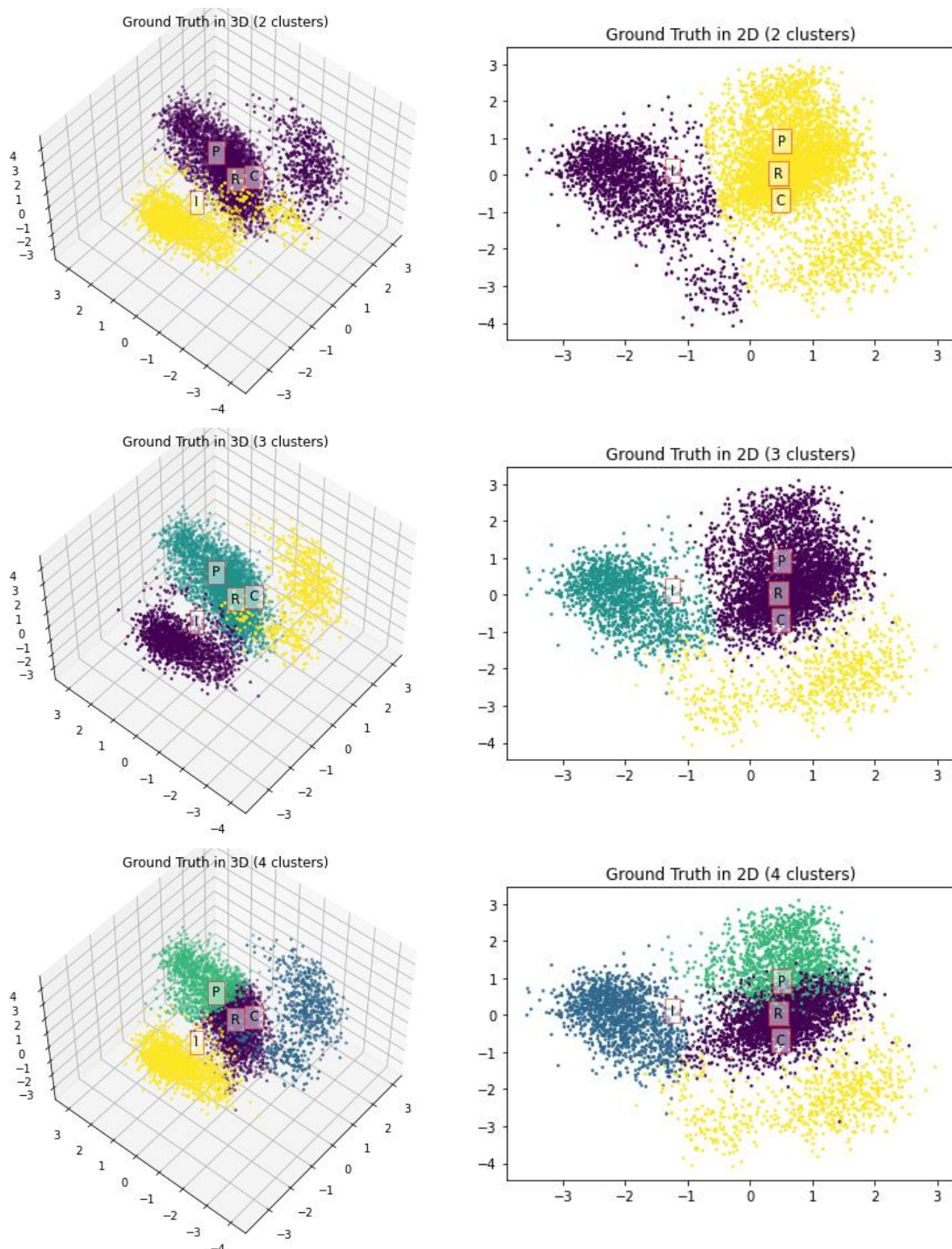




4. Result interpretation :

我針對社群聚類結果最好的幾種做進一步的分析，如圖十八所示，K-means 在 $K = 2, 3, 4$ 時的 ground truth (注意：圖中顏色僅用來分辨資料，不代表資料的 label)。實際上 group I 的中心離其它 group 最遠，再來是 group C (在三維空間中比較容易看出)，而 group P 和 group R 的中心彼此距離最近；做完 K-means 之後，group I 能最早被分辨出來，接著是 group C，最後是 group P 和 group R，因此可得知 K-means 聚類方法在 K 為 4 時能有效的將 group 分辨出來；另外也能看出 group P 和 group R 可能是相似的群體，有著相似的 interest 分布，group C 和他們不太一樣，但仍有著一定的關係，而 group I 的 interest 分布則跟另外三群差最多。

圖十八、



最後針對聚類後的結果與實際的 group 分布做比較，如圖十九所示。實際資料的分布有些雜亂，可看到「萬綠叢中一點紅」的畫面，也就是群體中心附近資料點的 label 和群體的不一樣，我認為可能的原因有三：

1. 在做問卷調查時，可能在 group A 之中訪問到來自 group B 的人，或是在整理資料時 label 出錯，導致 group A 資料之中有零星 group B 的情況。
2. 我在做資料前處理時，把可能有用的 interest 丟掉了(像是那些 nan 出現的機率太高或太低的 interest)，使得資料的資訊不夠完整。
3. 在 LDA 降維的過程中造成 group 邊緣資料的移動以及相對位置的改變。

而經過聚類後的資料分布因為 K-means 本身的演算法特性，使得群體中心附近的資料都能很一致的 label，僅少數邊緣資料點的 label 會比較雜亂。

兩者大致上群體分布趨勢相同，僅差別在於部分位於群體中心附近的資料 label 和群體本身不相同，而造成此情形的原因就是上述的三種，因此能得出結論：若資料本身無誤，且不同類的資料差異不會太小時，K-means 幾乎能成功的將資料正確分群，而少數的錯誤來自 LDA 降維時的資訊遺失、以及資料本身的群體邊緣所造成的誤差。最後，LDA 降維加上 K-means 聚類，確實能有效的對此 dataset 進行聚類分析。

圖十九、

