Data Science - Hw2

B10615045 陳尚富

環境建置：

macOs Catelina。使用 jupyter notbook 執行程式。

如何執行程式：

打開 B10615045_hw2.ipynb，然後 run all cell。即會產生結果 submit.csv。

程式架構：

首先，讀入題目給訂的 csv 檔案。

```
Data = pd.read_csv('data.csv')
Test = pd.read_csv('test.csv')
```

接著依照題目給的資料集說明，認為 feature1 和 feature3 最應該丟棄，接著測試了一次發現成果不好，又因為認為圖像辨識中 RGB 轉為灰階後 RGB 相關數值和明度、飽和度應該就不太重要，所以就丟掉了 feature11-13 和 feature17-19。發現結果還是沒有太好，所以丟棄 feature4-5，理由是因為 line extraction 結果在 data 中是 0、1 經過正規化的結果，而我覺得這樣正規化後的值，原本應該一樣的資料就會變得不同，導致可能在辨識上會受到影響。

```
# Drop the RPG stuff, used grey scale
Test = Test.drop(['index'], axis=1)
ID = Data['id']

# feature45 = pd.DataFrame(Data, columns = ['feature4','feature5'])

Data = Data.drop(['id',
#                   'feature1','feature2',
                  'feature3',
                   'feature4','feature5',
                   'feature11','feature12','feature13',
#                   'feature14','feature15','feature16',
                   'feature17', 'feature18','feature19'
                  ],axis = 1)

Data
```

	feature1	feature2	feature6	feature7	feature8	feature9	feature10	feature14	feature15	feature16
0	97	123	1.555555	1.096294	13.222221	108.740740	52.814816	-14.444445	33.888890	-19.444445
1	37	189	1.388889	1.485184	5.722223	23.885176	30.000000	-8.666667	19.000000	-10.333333
2	148	169	1.055555	0.774073	1.833333	2.611110	19.592592	-13.444445	-2.777778	16.222221
3	95	57	1.833333	3.411111	2.111111	1.718517	26.296297	-4.888889	24.444445	-19.555555
4	200	189	1.111111	1.088662	3.166667	1.005541	49.666668	-13.666667	33.666668	-20.000000
...	...	...	...	...	...	...	...	...	...	...

查看資料中是否有 Nan 的資料。沒有，所以就不用對 Nan 做處理。

```
Data.isnull().sum()
```

```
feature1     0
feature2     0
feature6     0
feature7     0
feature8     0
feature9     0
feature10    0
feature14    0
feature15    0
feature16    0
dtype: int64
```

對資料集做標準化，因為通過計算資料集中樣本的相關統計訊息，對每個特徵進行 centering and scaling，使資料集呈現標準常態分佈。對大多數機器學習而言能夠有較好的結果。

Normalize

```
X= np.asarray(Data)
from sklearn.preprocessing import StandardScaler, Normalizer, scale

scale = StandardScaler()
X = scale.fit_transform(X)
# normalizer = Normalizer().fit(X)
# X = normalizer.transform(X)
```

原本要使用 PCA 將資料及降維，只取資料集中較為重要的特徵，但是發現效果不彰，所以最終將其移除。

Reducing the dimensions of the data

```
# pca = PCA(n_components = 9)
# X = pca.fit_transform(X)
# X = pd.DataFrame(X)
# # X.columns = ['P1', 'P2']

# X.head(2)
```

使用 K means 演算法，來對此資料集進行非監督式學習。分群數設定為 5，理由是使用 silhouette_score 計算後，發現分群為 2~7 的結果較佳，然後再經過使用平台測試後，最終分群為五的效果較好。

```
kmean = KMeans(n_clusters=5, random_state = 0).fit(X)
pred = kmean.labels_
print(pred)
```

```
[2 2 1 ... 0 3 1]
```

跑過所有 test 的資料，判斷倆倆是否為同一群，是輸出 1、否輸出 0。

Match if they were same cluster

```
result = np.array([],dtype=int)

for index,value in enumerate(Test.values):
#     print(index,value)
    if pred[value[0]] == pred[value[1]]:
        result = np.append(result,1)
#         print('yes')
    else:
        result = np.append(result,0)
#         print('no')
```

最後，輸出結果。

輸出 submit.csv

```
def output_csv(result, filename):
    df = pd.DataFrame(result)
    df.to_csv(filename, header = ['ans'], index_label = 'index')

output_csv(result, "submit.csv")
```

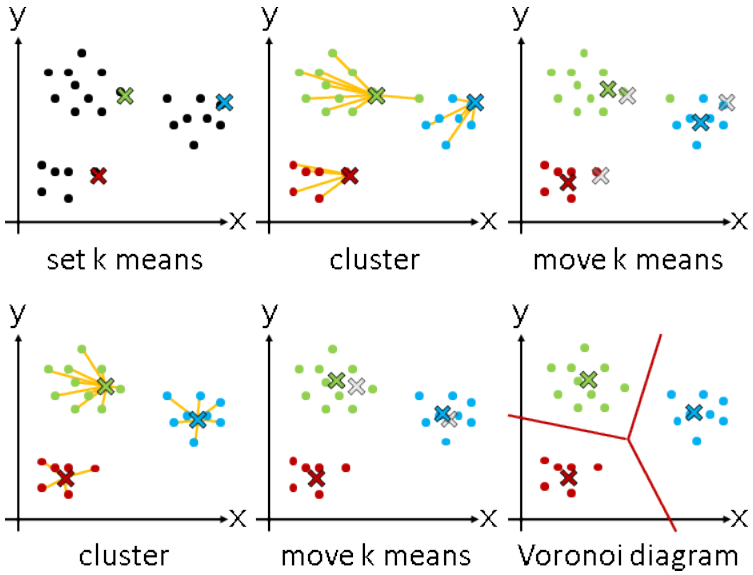
演算法流程：

k-means 是一種非監督式學習，且是一種 clustering 常用的方法。簡單形容 k-means 的話，就是物以類聚。

K-means 的主要算法：

1. 要先決定要分 k 群，接著隨機選 k 個點作為初始群心。
2. 將資料集中的資料，分群到最靠近自己的群心內。
3. 重新計算該群的群心。

重複 2.和 3.，直到群心不在移動時，k-means 結束。



（圖片出處：http://www.csie.ntnu.edu.tw/~u91029/Fitting.html）