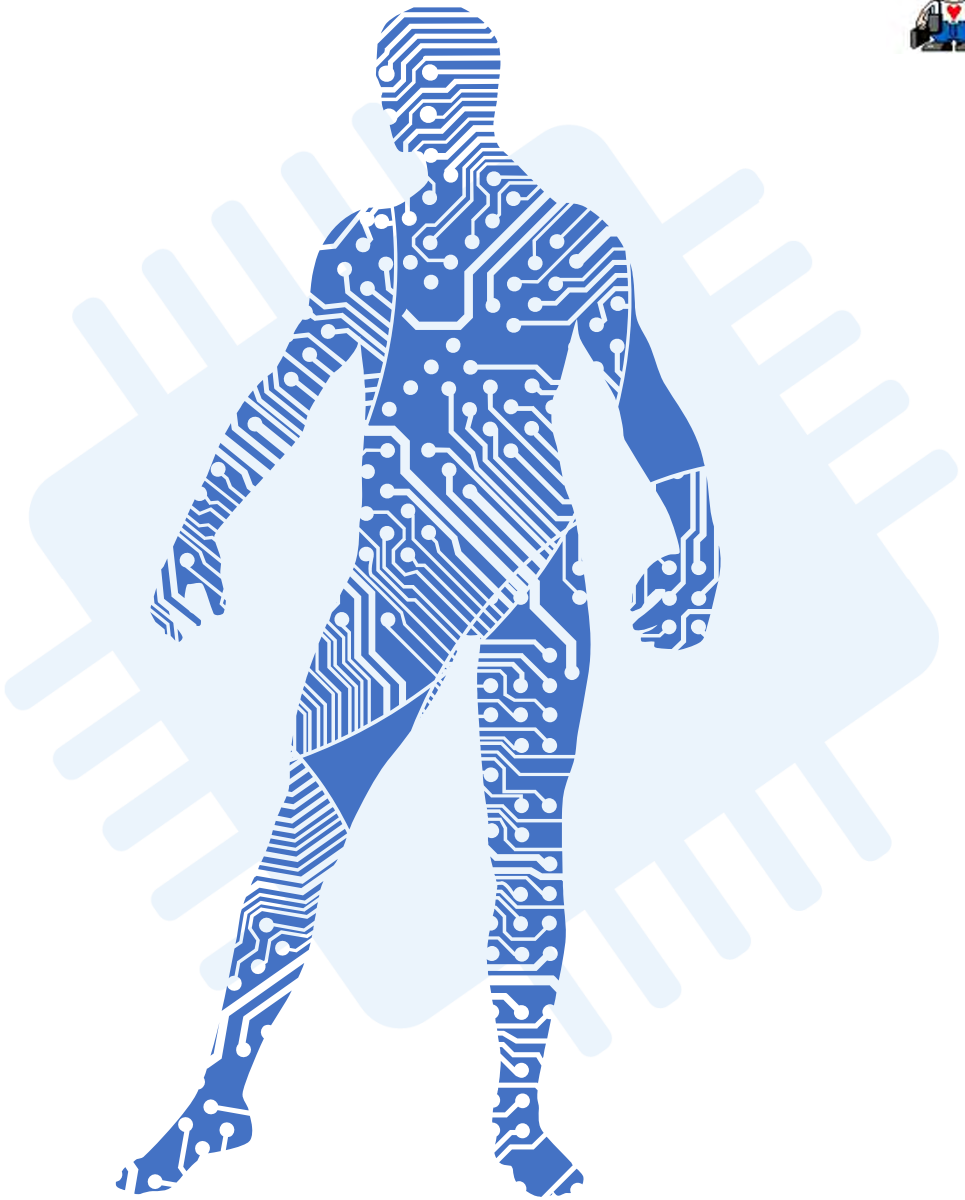
機器學習

第 14 章 K 平均集群法
(K-Means Clustering)

講師：紀俊男

本章大綱

- 原理解說
- 資料前處理
- 實作 K 平均法
- 將結果視覺化
- 本章總結



AI 原理解説

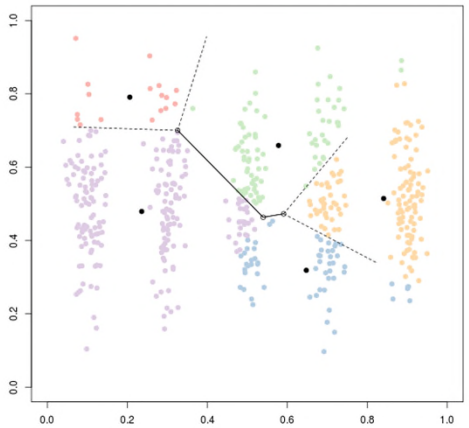
何謂「集群演算法（Clustering）」

Genre	Age	Income (k$)	Spending	
Male	19	15	39	
Male	21	15	81	
Female	20	16	6	
Female	[illegible]	[illegible]	77	
Female	[illegible]	[illegible]	40	
Female	22	17	76	
Female	35	18	6	
Female	23	18	94	
Male	64	19	3	

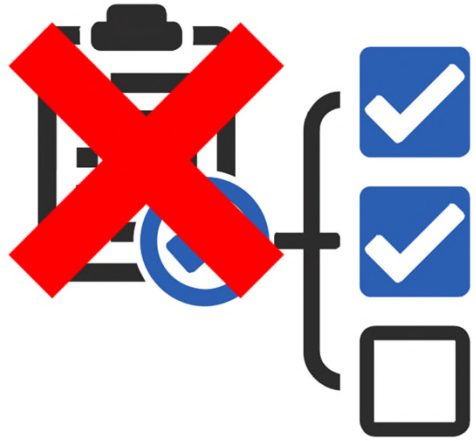
自變數 X

?

只有自變數 X，
沒有應變數 Y 的問題

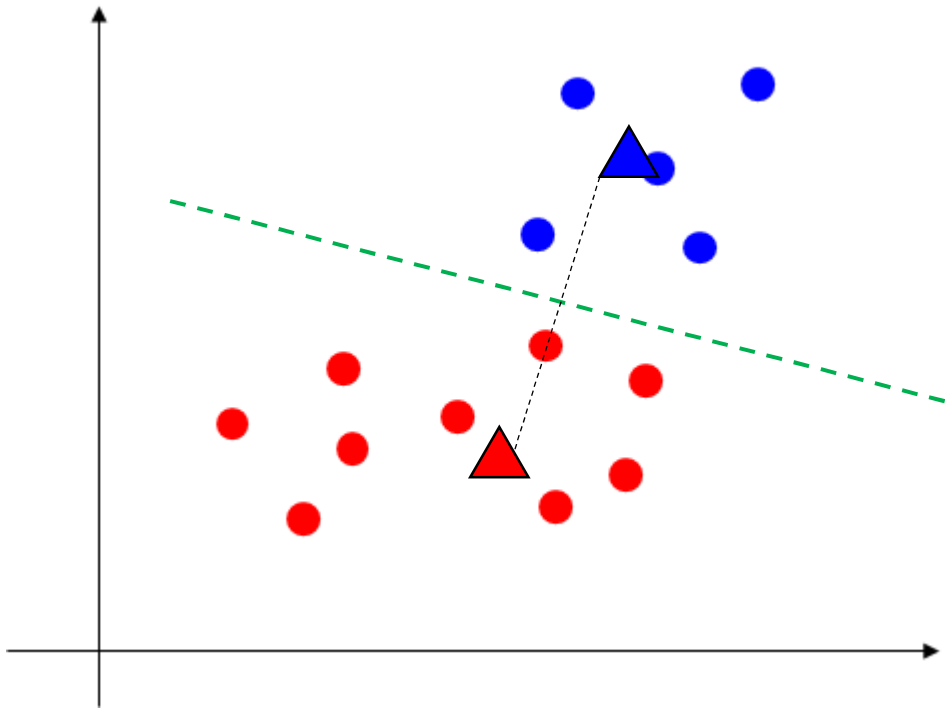


利用相似度（距離）
慢慢找到所屬群體



又稱為「無監督式學習」
（沒有老師、沒有答案）

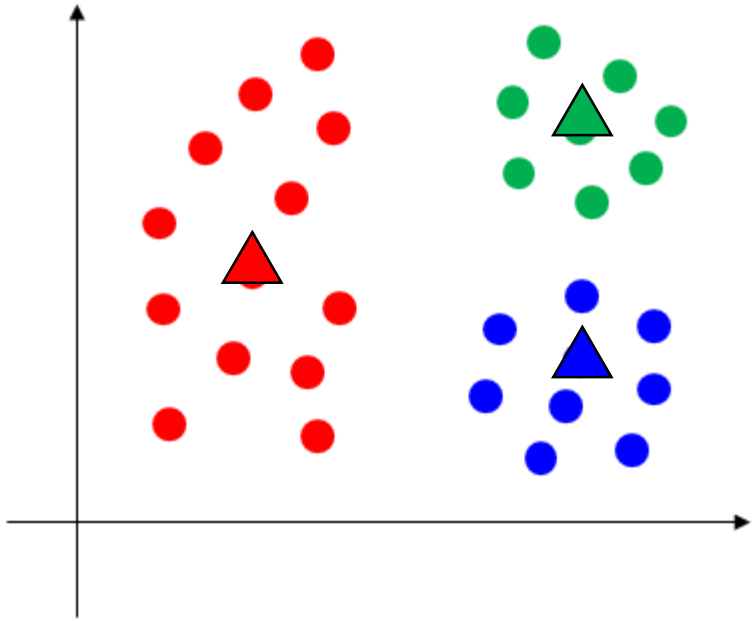
何謂「K 平均法（K-Means）」

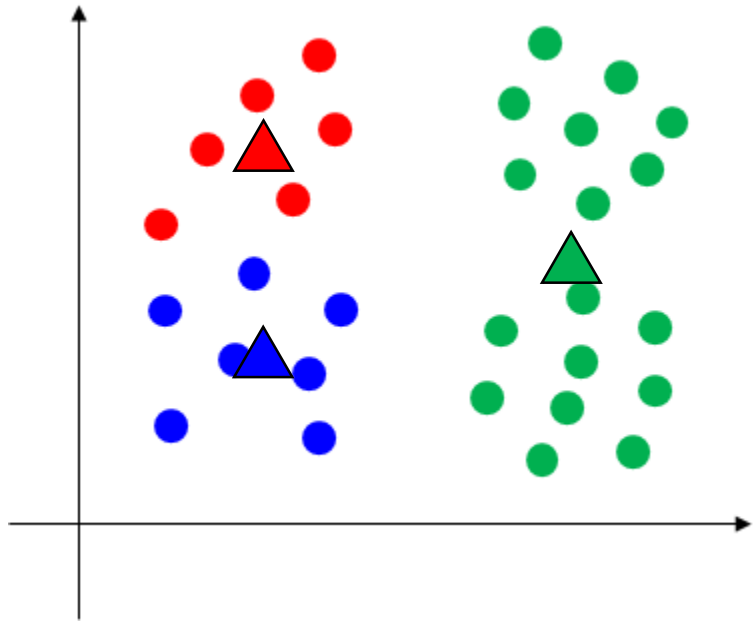


1. 假設要分 K 群，任選 K 個中心點。
2. 對所有樣本點，看距離哪個中心點近，分給該中心點所屬群。
3. 重新計算各群中心點。
4. 重複步驟（2）~（3）。
5. 若樣本點所屬群沒有變化，流程結束。

中心點隨機初始化陷阱

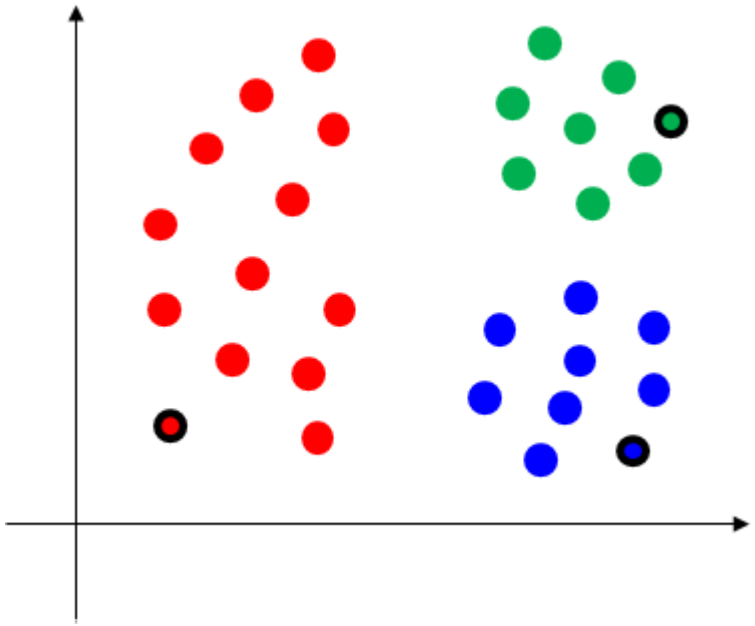
- **中心點**選擇**不同**，有可能導致**不同**的集群**結果**





中心點隨機初始化陷阱

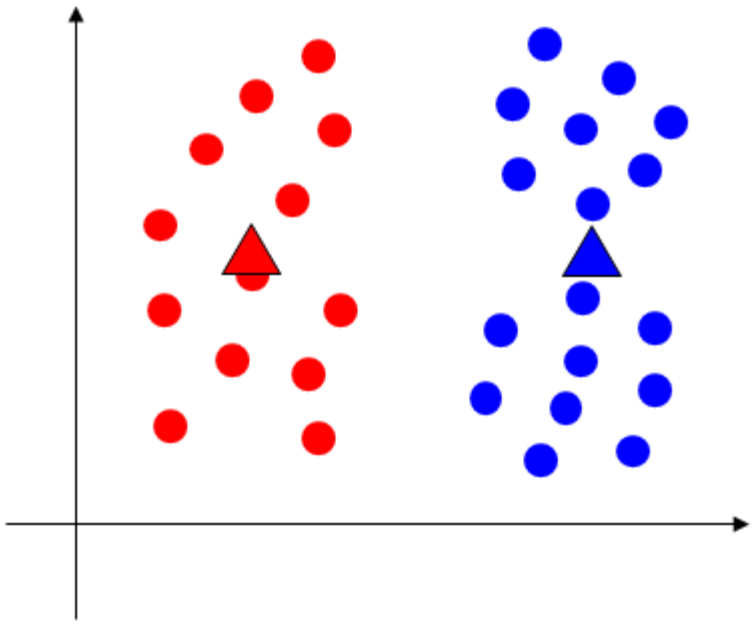
- 解決方法：**K-Means++ 演算法**
  - 挑選「彼此盡量遠離」的初始化中心點



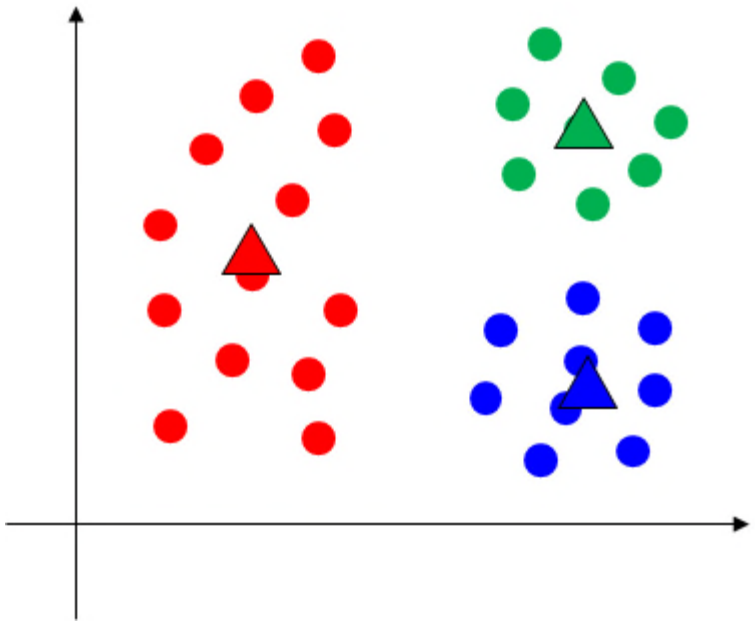
1. 任意選擇 K 個樣本點當中心點。
2. 其它樣本點，看距離哪個**中心點**近，分給該中心點所屬群。
3. 重新計算各群中心點。找各群中，離**舊中心點最遠**的樣本點，當**新中心點**。
4. 重複步驟（2）~（3）。
5. 若樣本點所屬群沒有變化，**開始一般的** K-Means 流程。

如何選擇正確的 K 值

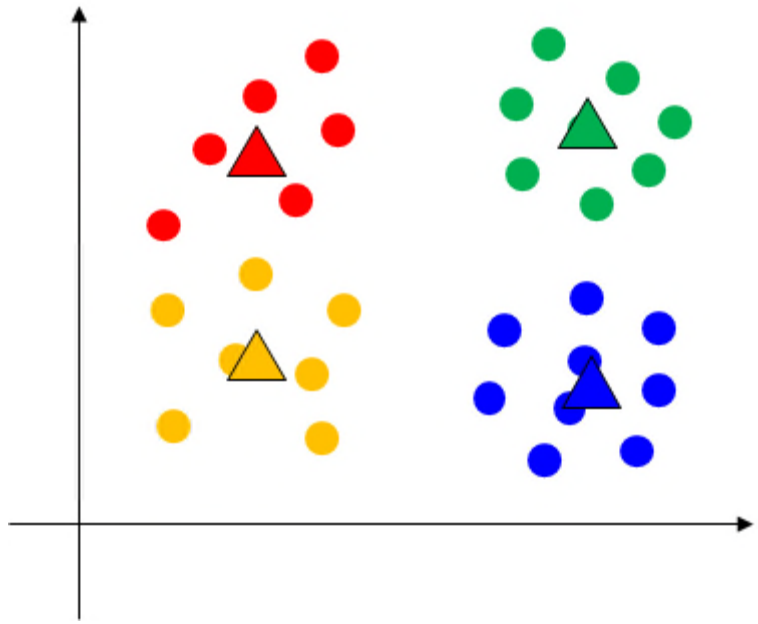
- 起始 **K 值**不同，最後答案會**不一樣**



K = 2



K = 3

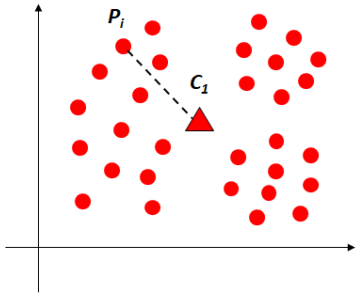


K = 4

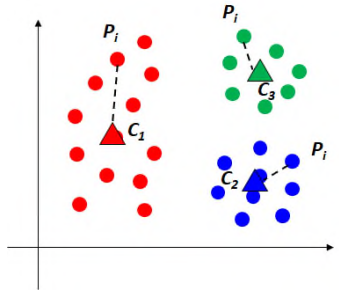
如何選擇正確的 K 值

- 解決方法：使用「**群內平方和**」（Within Cluster Sum of Square）

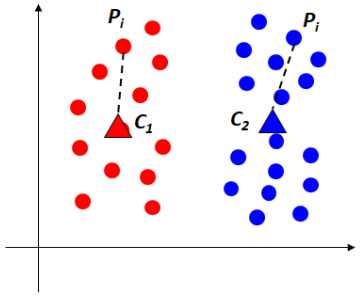
$$WCSS = \sum_{P_i \text{ in } Cluster_1} distance(P_i, C_1)^2 + \sum_{P_i \text{ in } Cluster_1} distance(P_i, C_1)^2 + \sum_{P_i \text{ in } Cluster_1} distance(P_i, C_1)^2 \ldots$$



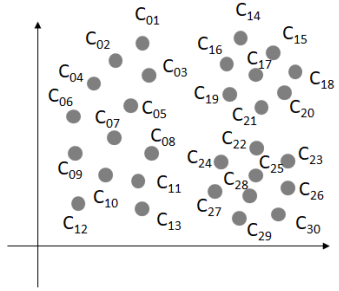
$$WCSS = \sum_{P_i \text{ in } Cluster_1} distance(P_i, C_1)^2$$



$$WCSS = \sum_{P_i \text{ in } Cluster_1} distance(P_i, C_1)^2 + \sum_{P_i \text{ in } Cluster_1} distance(P_i, C_1)^2 + \sum_{P_i \text{ in } Cluster_1} distance(P_i, C_1)^2$$



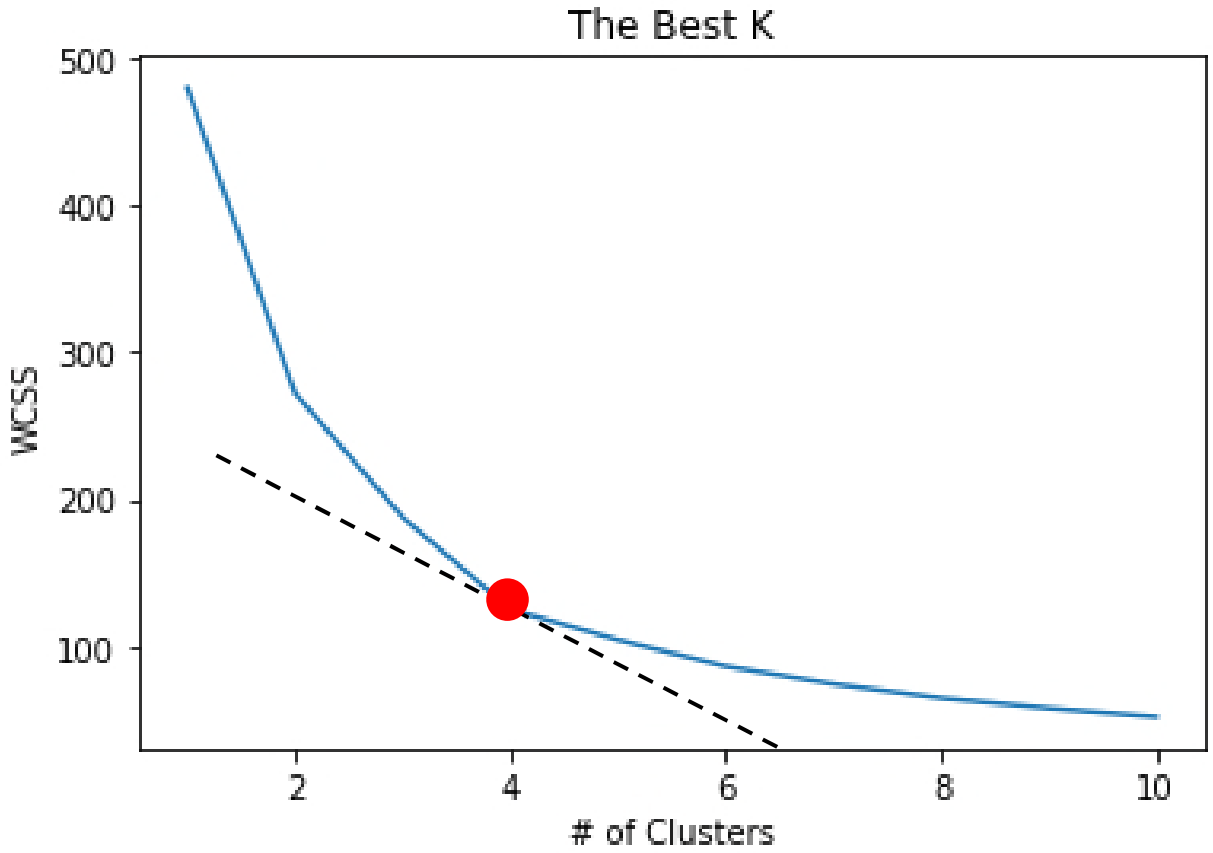
$$WCSS = \sum_{P_i \text{ in } Cluster_1} distance(P_i, C_1)^2 + \sum_{P_i \text{ in } Cluster_1} distance(P_i, C_1)^2$$



$$WCSS = 0$$

如何選擇正確的 K 值

- 最佳 K 值：下圖**紅點**處（依照「凸邊形優化（Convex Optimization）」定理）



資料前處理

下載與瀏覽資料集

- 依照講師指示，**下載**並**瀏覽**資料集

Mall_Customers.csv

	A	B	C	D	E
1	ID	Gender	Age	Income (k$)	Spending
2	1	Male	19	15	39
3	2	Male	21	15	81
4	3	Female	20	16	6
5	4	Female	23	16	77
6	5	Female	31	17	40
7	6	Female	22	17	76
8	7	Female	35	18	6
9	8	Female	23	18	94
10	9	Male	64	19	3

- **ID**：客戶代號
- **Gender**：性別
- **Age**：年齡
- **Income(K$)**：收入
- **Spending**：花費指數（0~100%）

目的：利用「年齡、收入...」
--> 將客戶分成若干群

資料前處理

撰寫程式碼

```python
import HappyML.preprocessor as pp

# Load Dataset
dataset = pp.dataset(file="Mall_Customers.csv")

# Decomposition
X = pp.decomposition(dataset, x_columns=[1, 2, 3, 4])

# One-Hot Encoding
X = pp.onehot_encoder(ary=X, columns=[0], remove_trap=True)

# Feature Scaling (for PCA Feature Selection)
X = pp.feature_scaling(fit_ary=X, transform_arys=X)

# Feature Selection (PCA)
from HappyML.preprocessor import PCASelector

selector = PCASelector()
X = selector.fit(x_ary=X, verbose=True, plot=True).transform(x_ary=X)
```

資料前處理流程：

1. 載入資料
2. 取得自變數
3. 處理缺失資料

   （無缺失資料）
4. 類別資料數位化

   （+去除虛擬變數陷阱）
5. 特徵縮放（PCA 必須）
6. 特徵選擇（使用 PCA）

隨堂練習：資料前處理

- 請撰寫下列程式碼，並予以執行，完成「**資料前處理**」的步驟：

```python
import HappyML.preprocessor as pp

# Load Dataset
dataset = pp.dataset(file="Mall_Customers.csv")

# Decomposition
X = pp.decomposition(dataset, x_columns=[1, 2, 3, 4])

# One-Hot Encoding
X = pp.onehot_encoder(ary=X, columns=[0], remove_trap=True)

# Feature Scaling (for PCA Feature Selection)
X = pp.feature_scaling(fit_ary=X, transform_arys=X)

# Feature Selection (PCA)
from HappyML.preprocessor import PCASelector

selector = PCASelector()
X = selector.fit(x_ary=X, verbose=True, plot=True).transform(x_ary=X)
```

實作 K 平均法

使用「標準函式庫」實作

程式碼

載入必要套件

```
1  from sklearn.cluster import KMeans
2  import time
3
4  kmeans = KMeans(n_clusters=4, init="k-means++", random_state=int(time.time()))
5  Y_pred = kmeans.fit_predict(X)
```

集群個數（K 值）

初始中心點演算法（"random" 或 "k-means++"）

產生物件本身

訓練 & 預測

執行結果

Y_pred

	0
0	1
1	1
2	2
3	1
4	2
5	1
6	2
7	1
8	2
9	1

隨堂練習：使用「標準函式庫」實作

- 請輸入下列程式碼，並執行看看，是否能取得客戶的**集群結果**：

```
1  from sklearn.cluster import KMeans
2  import time
3
4  kmeans = KMeans(n_clusters=4, init="k-means++", random_state=int(time.time()))
5  Y_pred = kmeans.fit_predict(X)
```

以「標準函式庫」找出最佳的 K 值

- 程式碼

```
1  from sklearn.cluster import KMeans
2  import time
3
4  # Find Best K
5  wcss = []
6  for i in range(1, 11):
7      kmeans = KMeans(n_clusters=i, init="k-means++", random_state=int(time.time()))
8      kmeans.fit(X)
9      wcss.append(kmeans.inertia_)
10
11 # Draw WCSS for each K
12 import matplotlib.pyplot as plt
13
14 plt.plot(range(1, 11), wcss)
15 plt.title("The Best K")
16 plt.xlabel("# of Clusters")
17 plt.ylabel("WCSS")
18 plt.show()
```

計算分成 1~10 群的 WCSS 值

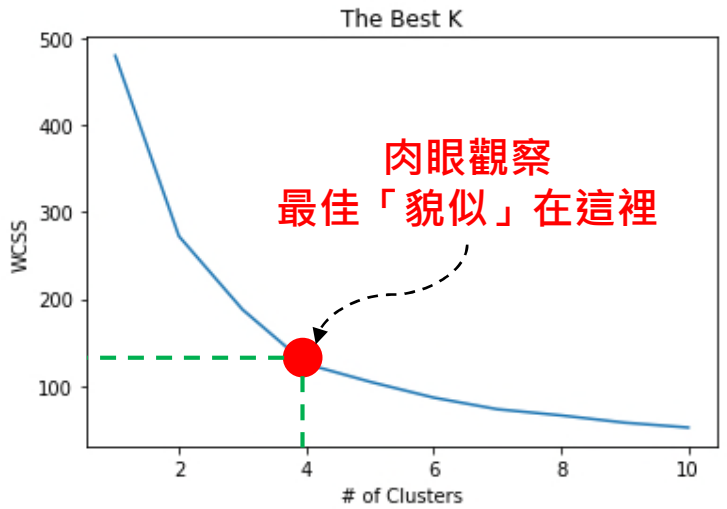
儲存 1~10 的 WCSS →

擬合這次 K=i 的結果

將這次的 WCSS 存起來 →

可取得這次的 WCSS 值

繪製所有的 WCSS 值



隨堂練習：以「標準函式庫」找出最佳的 K 值

- 請輸入下列程式碼，並執行看看，然後以「**肉眼**」選擇最佳的 K 值：

```
1  from sklearn.cluster import KMeans
2  import time
3
4  # Find Best K
5  wcss = []
6  for i in range(1, 11):
7      kmeans = KMeans(n_clusters=i, init="k-means++", random_state=int(time.time()))
8      kmeans.fit(X)
9      wcss.append(kmeans.inertia_)
10
11 # Draw WCSS for each K
12 import matplotlib.pyplot as plt
13
14 plt.plot(range(1, 11), wcss)
15 plt.title("The Best K")
16 plt.xlabel("# of Clusters")
17 plt.ylabel("WCSS")
18 plt.show()
```

使用「快樂版函式庫」實作

- 程式碼解說（１）：　/HappyML/clustering.py

引入必要套件

類別的成員變數

建構函數

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time

from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler


class KMeansCluster:
    __cluster = None
    __best_k = None
    __max_k = None
    __strategy = None
    __random_state = None
    __centroids = None

    def __init__(self, best_k="auto", max_k=10, random_state=int(time.time())):
        if type(best_k) is int:
            self.__strategy = "fixed"
            self.best_k = best_k
        else:
            self.__strategy = "auto"
            self.best_k = 8

        self.__max_k = max_k
        self.__random_state = random_state

        self.__cluster = KMeans(n_clusters=self.best_k, init="k-means++", random_state=self.__random_state)
```

- best_k = “auto” 或 整數
  - “auto”：交由本類別決定
  - 整數：自行指定
- max_k = 整數
  - 最佳 K 值尋找範圍的上限
  - 最佳 K 值 = 1 ~ max_k

使用「快樂版函式庫」實作

- 程式碼解說（２）：

/HappyML/clustering.py

cluster 的 setter & getter

best_k 的 setter & getter

centroids（中心點）的 getter

```
31      @property
32      def cluster(self):
33          return self.__cluster
34
35      @cluster.setter
36      def cluster(self, cluster):
37          self.__cluster = cluster
38
39      @property
40      def best_k(self):
41          return self.__best_k
42
43      @best_k.setter
44      def best_k(self, best_k):
45          if (type(best_k) is int) and (best_k >= 1):
46              self.__best_k = best_k
47          else:
48              self.__best_k = 1
49
50      @property
51      def centroids(self):
52          return self.__centroids
```

使用「快樂版函式庫」實作

- 程式碼解說（3）：

/HappyML/clustering.py

計算 1~max_k 的所有 WCSS 值

找尋最佳的 K 值（X 前進一格、Y 前進不足一格時停止）

verbose = True 印出找到的最佳 K 值

把所有的 WCSS 值繪製出來，並把最佳 K 值也點描出來

使用最佳 K 值去擬合，並保存所有中心點資訊

預測各樣本點的所屬群 →

```
54      def fit(self, x_ary, verbose=False, plot=False):
55          if self.__strategy == "auto":
56              wcss = []
57              for i in range(1, self.__max_k+1):
58                  kmeans = KMeans(n_clusters=i, init="k-means++", random_state=self.__random_state)
59                  kmeans.fit(x_ary)
60                  wcss.append(kmeans.inertia_)
61
62              scaler = MinMaxScaler(feature_range=(0, len(wcss)-1))
63              wcss_scaled = scaler.fit_transform(np.array(wcss).reshape(-1, 1)).ravel()
64              for i in range(1, wcss_scaled.shape[0]):
65                  if (wcss_scaled[i-1]-wcss_scaled[i]) < 1:
66                      break
67              self.best_k = i
68
69              if verbose:
70                  print("The best clusters = {}".format(self.best_k))
71
72              if plot:
73                  plt.plot(range(1, len(wcss)+1), wcss, color="blue")
74                  plt.scatter(x=self.best_k, y=wcss[self.best_k], color="red")
75                  plt.title("The Best Cluster")
76                  plt.xlabel("# of Clusters")
77                  plt.ylabel("WCSS")
78                  plt.show()
79
80          # Fit the Model
81          self.cluster = KMeans(n_clusters=self.best_k, random_state=self.__random_state)
82          self.cluster.fit(x_ary)
83          self.__centroids = self.cluster.cluster_centers_
84
85          return self
86
87      def predict(self, x_ary, y_column="Result"):
88          return pd.DataFrame(self.cluster.predict(x_ary), index=x_ary.index, columns=[y_column])
```

使用「快樂版函式庫」實作

- 呼叫範例

1

2

```python
from HappyML.clustering import KMeansCluster

cluster = KMeansCluster()
Y_pred = cluster.fit(x_ary=X, verbose=True, plot=True).predict(x_ary=X, y_column="Customer Type")

# Optional, Attach the Y_pred to Dataset & Save as .CSV file
dataset = pp.combine(dataset, Y_pred)
dataset.to_csv("Mall_Customers_answers.csv", index=False)
```

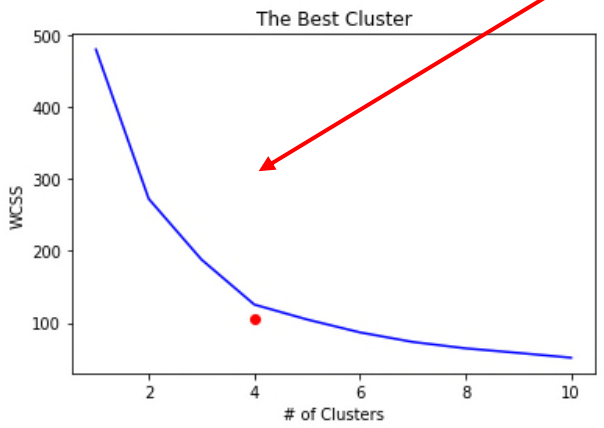
3 訓練 & 預測

4 將 Y_pred 結果，附加到 dataset 尾部

5 將 dataset 輸出成 .CSV 檔

- 執行結果

```
The best clusters = 4
```



Y_pred

Index	Customer Type
0	3
1	3
2	1
3	3
4	1
5	3
6	1
7	3
8	1
9	3

	ID	Gender	Age	Income (k$)	Spending (1-100)	Customer Type
0	1	Male	19	15	39	0
1	2	Male	21	15	81	0
2	3	Female	20	16	6	1
3	4	Female	23	16	77	0
4	5	Female	31	17	40	1
5	6	Female	22	17	76	0
6	7	Female	35	18	6	1
7	8	Female	23	18	94	0

ID	Gender	Age	Income (k$)	Spending (1-100)	Customer Type
1	Male	19	15	39	0
2	Male	21	15	81	0
3	Female	20	16	6	3
4	Female	23	16	77	0
5	Female	31	17	40	1
6	Female	22	17	76	0
7	Female	35	18	6	3

隨堂練習：使用「快樂版函式庫」實作

- 請先將先前使用 **標準函式庫** 製作的 K 平均法**註解**掉。

- 輸入下列程式，並執行看看：

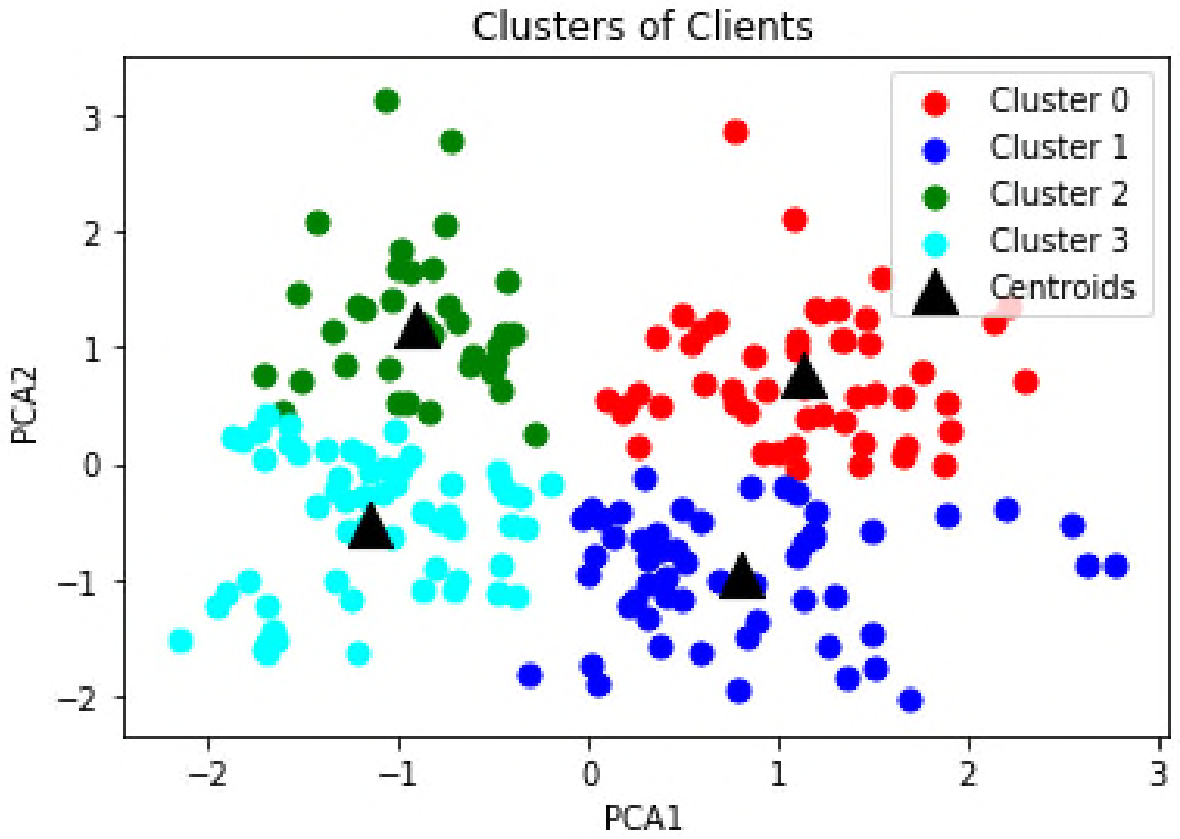
```
from HappyML.clustering import KMeansCluster

cluster = KMeansCluster()
Y_pred = cluster.fit(x_ary=X, verbose=True, plot=True).predict(x_ary=X, y_column="Customer Type")

# Optional, Attach the Y_pred to Dataset & Save as .CSV file
dataset = pp.combine(dataset, Y_pred)
dataset.to_csv("Mall_Customers_answers.csv", index=False)
```

- 觀察下列輸出：
  - 最佳 K 值、Y_pred、dataset 尾部、輸出的 .CSV 檔

將結果視覺化

使用「標準函式庫」視覺化

- 先看**執行結果**



使用「標準函式庫」視覺化

- 程式碼

引入必要套件

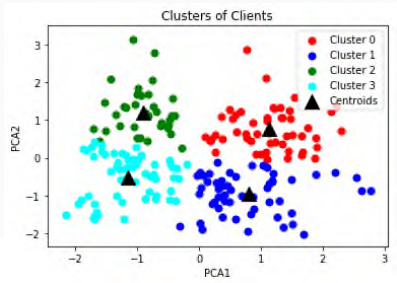
將 DataFrame 轉成 NDArray

繪製四個群
樣本點所在位置

繪製四個群
的中心點

繪製標題、軸線標籤
與圖示

```python
import matplotlib.pyplot as plt

# Draw Samples
Y_array = Y_pred.values.ravel()
plt.scatter(X.iloc[Y_array==0, 0], X.iloc[Y_array==0, 1], s=50, c="red", label="Cluster 0")
plt.scatter(X.iloc[Y_array==1, 0], X.iloc[Y_array==1, 1], s=50, c="blue", label="Cluster 1")
plt.scatter(X.iloc[Y_array==2, 0], X.iloc[Y_array==2, 1], s=50, c="green", label="Cluster 2")
plt.scatter(X.iloc[Y_array==3, 0], X.iloc[Y_array==3, 1], s=50, c="cyan", label="Cluster 3")

# Draw Centroids
centroids = cluster.centroids
plt.scatter(centroids[:, 0], centroids[:, 1], s=200, c="black", marker="^", label="Centroids")

# Labels & Legends
plt.title("Clusters of Clients")
plt.xlabel("PCA1")
plt.ylabel("PCA2")
plt.legend(loc="best")
plt.show()
```



隨堂練習：使用「標準函式庫」視覺化

- 請輸入下列程式碼，並執行看看，是否能將集群結果**視覺化**：

```
1  import matplotlib.pyplot as plt
2
3  # Draw Samples
4  Y_array = Y_pred.values.ravel()
5  plt.scatter(X.iloc[Y_array==0, 0], X.iloc[Y_array==0, 1], s=50, c="red", label="Cluster 0")
6  plt.scatter(X.iloc[Y_array==1, 0], X.iloc[Y_array==1, 1], s=50, c="blue", label="Cluster 1")
7  plt.scatter(X.iloc[Y_array==2, 0], X.iloc[Y_array==2, 1], s=50, c="green", label="Cluster 2")
8  plt.scatter(X.iloc[Y_array==3, 0], X.iloc[Y_array==3, 1], s=50, c="cyan", label="Cluster 3")
9
10 # Draw Centroids
11 centroids = cluster.centroids
12 plt.scatter(centroids[:, 0], centroids[:, 1], s=200, c="black", marker="^", label="Centroids")
13
14 # Labels & Legends
15 plt.title("Clusters of Clients")
16 plt.xlabel("PCA1")
17 plt.ylabel("PCA2")
18 plt.legend(loc="best")
19 plt.show()
```

使用「快樂版函式庫」視覺化

- 程式碼解說（１）：

/HappyML/model_drawer.py

引入調色盤（Color Map）物件→

如果傳入的自變數，
不是只有兩個特徵，則退出

將 DataFrame 轉成 NDArray →

找出應變數 Y 究竟有幾種答案 →

```
1  import matplotlib.cm as cm
2
3  def cluster_drawer(x, y, centroids, title="", font='Arial Unicode MS'):
4      # Check for x has only two columns
5      if x.shape[1] != 2:
6          print("ERROR: x must have only two features to draw!!")
7          return None
8
9      # Change y from DataFrame to NDArray
10     y_ndarray = y.values.ravel()
11
12     # Get how many classes in y
13     y_unique = np.unique(y_ndarray)
```

使用「快樂版函式庫」視覺化

- 程式碼解說（２）：

/HappyML/model_drawer.py

幾種 Y 就有幾種色 →

繪製樣本點

繪製中心點

設定中文字形

繪製標題、軸線標籤、與圖示

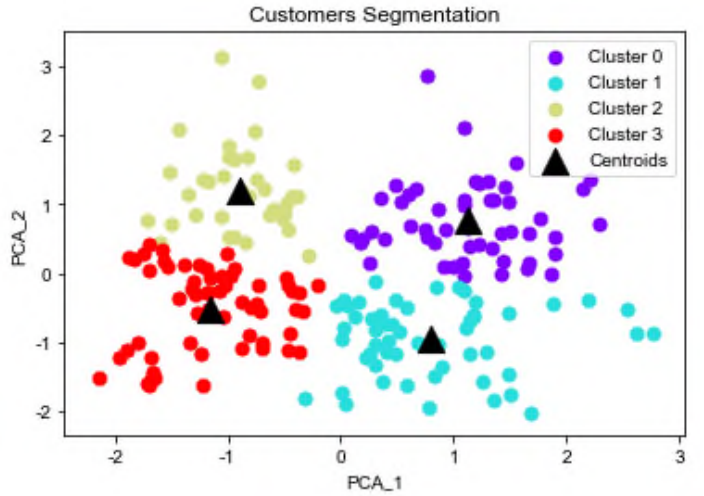
```
15      # Iterate all classes in y
16      colors = cm.rainbow(np.linspace(0, 1, len(y_unique)))
17      for val, col in zip(y_unique, colors):
18          plt.scatter(x.iloc[y_ndarray==val, 0], x.iloc[y_ndarray==val, 1], s=50, c=col, label="Cluster {}".format(val))
19
20      # Draw Centroids
21      plt.scatter(centroids[:, 0], centroids[:, 1], s=200, c="black", marker="^", label="Centroids")
22
23      # Labels & Legends
24      # for showing Chinese characters
25      plt.rcParams['font.sans-serif']=[font]
26      plt.rcParams['axes.unicode_minus'] = False
27
28      plt.title(title)
29      plt.xlabel(x.columns[0])
30      plt.ylabel(x.columns[1])
31      plt.legend(loc="best")
32      plt.show()
```

使用「快樂版函式庫」視覺化

- 呼叫範例

1
```
import HappyML.model_drawer as md
```
2
```
md.cluster_drawer(x=X, y=Y_pred, centroids=cluster.centroids, title="Customers Segmentation")
```

- 執行結果



隨堂練習：使用「快樂版函式庫」視覺化

- 請先將先前使用 **標準函式庫** 視覺化的程式碼**註解**掉。

- 輸入下列程式，並執行看看：

```
1  import HappyML.model_drawer as md
2
3  md.cluster_drawer(x=X, y=Y_pred, centroids=cluster.centroids, title="Customers Segmentation")
```

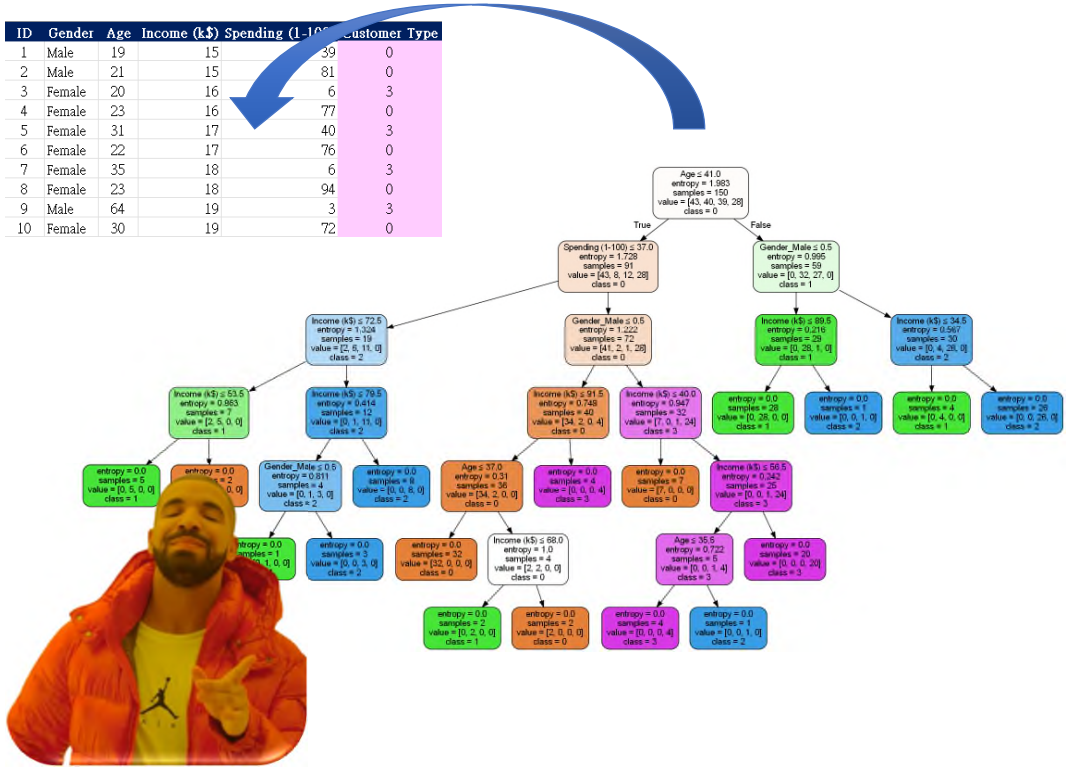
AI 分析集群結果

如何分析集群結果

- 用「肉眼」分析集群原因

ID	Gender	Age	Income (k$)	Spending (1-100)	Customer Type
1	Male	19	15	39	0
2	Male	21	15	81	0
3	Female	20	16	6	3
4	Female	23	16	77	0
5	Female	31	17	40	3
6	Female	22	17	76	0
7	Female	35	18	6	3
8	Female	23	18	94	0
9	Male	64	19	3	3
10	Female	30	19	72	0

- 利用「決策樹」分析集群原因



原始程式碼解說

決策樹的
前處理程式碼

決策樹模型
訓練＆預測

```python
# In[] Preprocessing
import HappyML.preprocessor as pp

# Load Dataset
dataset = pp.dataset(file="Mall_Customers_answers.csv")

# Decomposition
X, Y = pp.decomposition(dataset, x_columns=[1, 2, 3, 4], y_columns=[5])

# One-Hot Encoding
X = pp.onehot_encoder(ary=X, columns=[0], remove_trap=True)

# Feature Selection
from HappyML.preprocessor import KBestSelector
selector = KBestSelector(best_k="auto")
X = selector.fit(x_ary=X, y_ary=Y, verbose=True, sort=True).transform(x_ary=X)

# Split Training / Testing Set
X_train, X_test, Y_train, Y_test = pp.split_train_test(x_ary=X, y_ary=Y)

# In[] Decision Tree
from HappyML.classification import DecisionTree

classifier = DecisionTree()
Y_pred = classifier.fit(X_train, Y_train).predict(X_test)
```

原始程式碼解說

決策樹效能
（對資料集之解釋程度）

```
21 # Performance
22 from HappyML.performance import KFoldClassificationPerformance
23
24 K = 10
25 kfp = KFoldClassificationPerformance(x_ary=X, y_ary=Y,
26         classifier=classifier.classifier, k_fold=K)
27
28 print("----- Decision Tree Classification -----")
29 print("{} Folds Mean Accuracy: {}".format(K, kfp.accuracy()))
30 print("{} Folds Mean Recall: {}".format(K, kfp.recall()))
31 print("{} Folds Mean Precision: {}".format(K, kfp.precision()))
32 print("{} Folds Mean F1-Score: {}".format(K, kfp.f_score()))
```

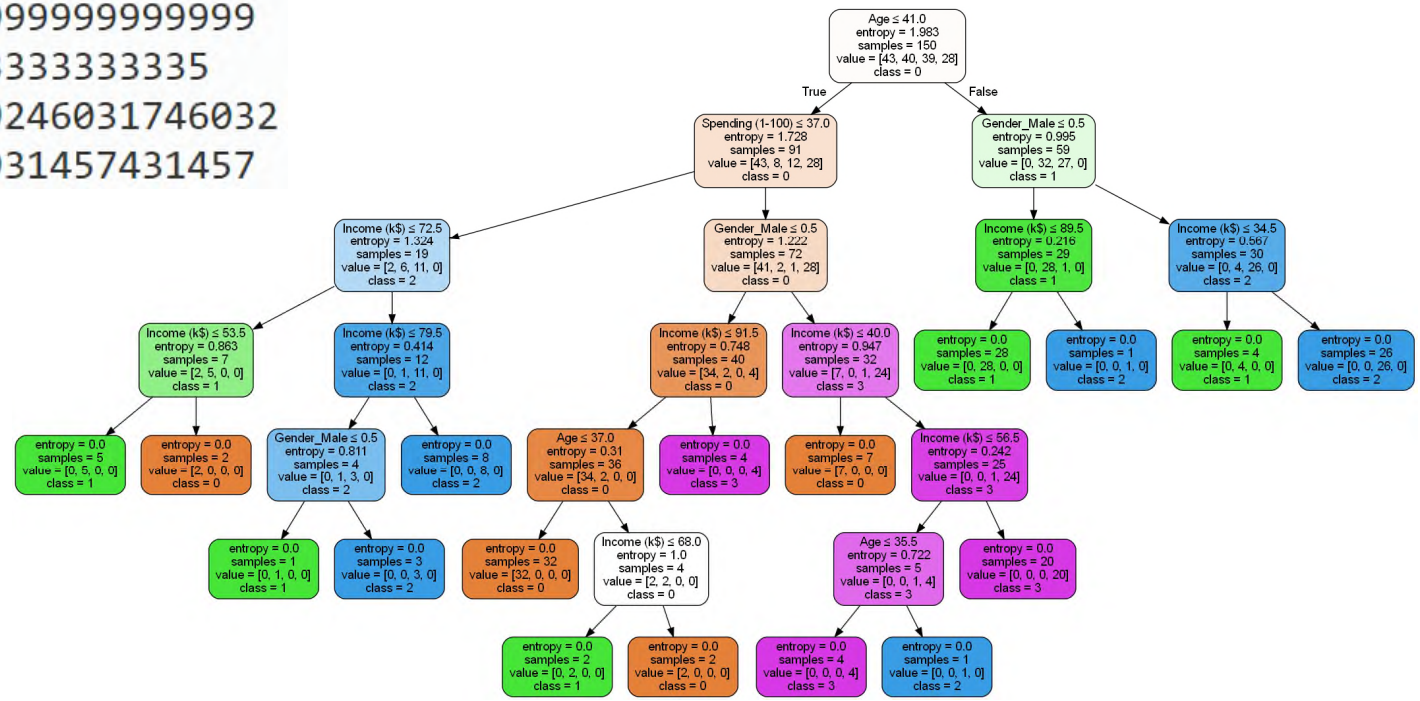
決策樹繪製

```
33
34 # Visualization
35 GRAPHVIZ_INSTALL = "C:/Program Files/Graphviz/bin"
36 import HappyML.model_drawer as md
37 from IPython.display import Image, display
38
39 graph = md.tree_drawer(classifier=classifier.classifier,
40     feature_names=X_test.columns, target_names="0123",
41     graphviz_bin=GRAPHVIZ_INSTALL)
42 display(Image(graph.create_png()))
```

原始程式碼解說

- 執行結果

```
Number of Features Selected: 4
----- Decision Tree Classification -----
10 Folds Mean Accuracy: 0.8999999999999999
10 Folds Mean Recall: 0.8920833333333335
10 Folds Mean Precision: 0.9119246031746032
10 Folds Mean F1-Score: 0.8869931457431457
```



隨堂練習：以「決策樹」分析集群結果

- 請撰寫下列程式碼，繪製出決策樹，並且分析集群原因（1）：

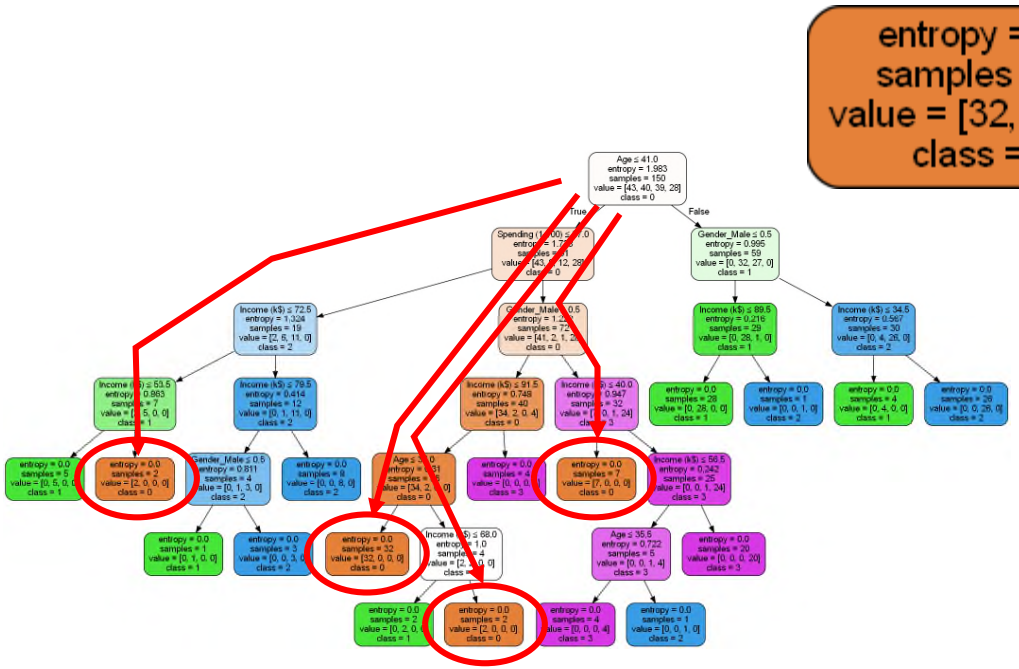
```
# Decomposition
X, Y = pp.decomposition(dataset, x_columns=[1, 2, 3, 4], y_columns=[5])

# One-Hot Encoding
X = pp.onehot_encoder(ary=X, columns=[0], remove_trap=True)

# Feature Selection
from HappyML.preprocessor import KBestSelector
selector = KBestSelector(best_k="auto")
X = selector.fit(x_ary=X, y_ary=Y, verbose=True, sort=True).transform(x_ary=X)

# Split Training / Testing Set
X_train, X_test, Y_train, Y_test = pp.split_train_test(x_ary=X, y_ary=Y)

# Decision Tree
from HappyML.classification import DecisionTree

classifier = DecisionTree()
Y_pred = classifier.fit(X_train, Y_train).predict(X_test)
```

隨堂練習：以「決策樹」分析集群結果

- 請撰寫下列程式碼，繪製出決策樹，並且分析集群原因（2）：

```
# Performance
from HappyML.performance import KFoldClassificationPerformance

K = 10
kfp = KFoldClassificationPerformance(x_ary=X, y_ary=Y,
        classifier=classifier.classifier, k_fold=K)

print("----- Decision Tree Classification -----")
print("{} Folds Mean Accuracy: {}".format(K, kfp.accuracy()))
print("{} Folds Mean Recall: {}".format(K, kfp.recall()))
print("{} Folds Mean Precision: {}".format(K, kfp.precision()))
print("{} Folds Mean F1-Score: {}".format(K, kfp.f_score()))

# Visualization
GRAPHVIZ_INSTALL = "C:/Program Files/Graphviz/bin"
import HappyML.model_drawer as md
from IPython.display import Image, display

graph = md.tree_drawer(classifier=classifier.classifier,
    feature_names=X_test.columns, target_names="0123",
    graphviz_bin=GRAPHVIZ_INSTALL)
display(Image(graph.create_png()))
```

以「決策樹」分析集群結果



entropy = 0.0
samples = 32
value = [32, 0, 0, 0]
class = 0

Types	Gender	Age	Income (k$)	Spending (1-100)	Sample
0		~41	53.5~72.5	~37	2
0	F	~37	~91.5	37~	32
0	F	~41	68~91.5	37~	2
0	M	~41	~40	37~	7

Class = 0

女性、年輕、高薪、常客

以「決策樹」分析集群結果

- 完整分析結果

Types	Gender	Age	Income (k$)	Spending (1-100)	Sample	
0		~41	53.5~72.5	~37	2	
0	F	~37	~91.5	37~	32	高薪年輕女性、常客
0	F	~41	68~91.5	37~	2	
0	M	~41	~40	37~	7	
1		~41	~53.5	~37	5	
1	F	~41	72.5~79.5	~37	1	
1	F	~41	~68	37~	2	
1	F	41~	~89.5	~37	28	高薪熟女、稀客
1	M	41~	~34.5		4	
2	M	~41	72.5~79.5	~37	3	
2		~41	79.5~	~37	8	
2	M	35.5~41	40~56	37~	1	
2	F	41~	89.5~		1	
2	M	41~	34.5~	37~	26	小資熟男、常客
3	F	~41	91.5~	37~	4	
3	M	~41	56~	37~	20	中產年輕男性、常客

本次目標

課後作業：信用卡客戶集群分析

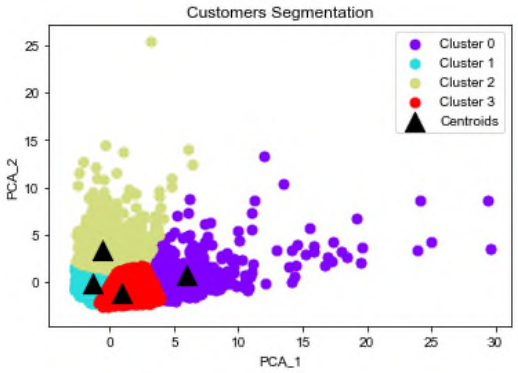
- 資料集說明
  - 請下載資料集 **CreditCards.csv**（資料原始出處： https://is.gd/sJd4hy ）
  - 欄位名稱解說如下：
    - **CUST_ID**：客戶編號
    - **BALANCE**：帳戶餘額
    - **BALANCE_FREQUENCY**：餘額更新頻率（0~100%）
    - **PURCHASES**：購物金額
    - **ONEOFF_PURCHASES**：單筆最大購物金額
    - …

- 題目要求
  - 請利用 K-Means 演算法，將客戶集群。
  - 請將客戶集群結果，以圖表繪製出來。
  - 將客戶集群結果，附加到原始 dataset 後方。
  - 對附加了集群結果的 dataset，重新做資料前處理。
  - 前處理時，使用 KBestSelector 挑選「顯著性高」的特徵。
  - 前處理完畢後，使用「決策樹」執行「監督式學習」，並用 10-Folds 交叉驗證，說明你的模型與真實結果有多貼近。
  - 最後繪製出決策樹。說明這樣的集群結果，最重要的決定因素是哪個？

課後作業：信用卡客戶集群分析

- 提示
  - 本資料集有「**缺失資料**」，請記得先補齊，再執行運算。

- 輸出結果




```
----- Decision Tree Classification -----
10 Folds Mean Accuracy: 0.9191101183921022
10 Folds Mean Recall: 0.8945332130323873
10 Folds Mean Precision: 0.9055827011793317
10 Folds Mean F1-Score: 0.8990512580449794
```

本章總結

- 何謂「集群演算法（Clustering）」
  - 只有**自變數 X**，沒有**應變數 Y** 的問題。
  - 利用**相似度（距離）**，慢慢找到所屬群體。
  - 又稱為「**無監督式學習**」。

- K-Means 理論
  - K 平均法**集群流程**。
  - 中心點隨機初始化陷阱解決方法：**K-Means++ 演算法**。
  - 選擇正確的 K 值的方法：取最佳的 **WCSS 值**。

- K-Means 相關套件
  - 演算法本身：sklearn.cluster.KMeans