

# 課程: Python程式語言應用專題報告

## 主題: 投資人投資組合-集群法應用

學號: 168

姓名: 李明昌

日期: 2024年10月3日

```
In [1]: # 注意:  
# 1. 本研究使用 Jupyter notebook 撰寫, 版本如下:  
# !jupyter --version
```

```
In [2]: # 2. 將 Jupyter notebook 轉換為 py 方法如下:  
# !jupyter nbconvert --to script report_168_alan.ipynb
```

```
In [3]: # 3. 下載舊版 Anaconda  
# 下載最新版 Anaconda: https://www.anaconda.com/download  
# 如果 Jupyter notebook 使用時有異常, 可以下載舊版 Anaconda: https://repo.anaconda.com/archive/  
# 範例: Anaconda3-2024.02-1-Windows-x86_64.exe, https://repo.anaconda.com/archive/
```

### 4.報告規範:

[https://github.com/rwepa/python\\_data\\_scientist/blob/main/report\\_readme.txt](https://github.com/rwepa/python_data_scientist/blob/main/report_readme.txt)

## 1.商業理解

```
In [4]: # 研究目的: 探討投資人的投資組合集群分析-使用集群法  
# 資料來源: Machine Learning and Data Science Blueprints for Finance, 2020.  
# 資料名稱: investor_portfolio.csv  
# 資料網址: https://github.com/rwepa/DataDemo/blob/master/investor_portfolio.csv  
# 報告名稱: report_168_alan.ipynb
```

研究的目標是建立一個機器學習模型，使用非監督式學習的 K-means 集群法。根據人口統計、承擔風險的能力和意願變數對投資人進行集群分析。

## 2.資料理解

資料理解包括以下主題：資料匯入、資料摘要、探索性資料分析、資料視覺化、資料清理、資料合併、特徵選擇、資料轉換。其中資料匯入與資料摘要為必需主題。本研究使用 Python 程式語言 (McKinney, 2010) 並參考RWEPA 網站資料 (Lee, 2024)。Lee, 2024)。

## 2.1 資料匯入

```
In [5]: # 載入 Python 模組
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas import read_csv, set_option
import seaborn as sns

# 載入 scikit-learn 集群法模組
from sklearn import cluster
from sklearn.cluster import KMeans

# 載入計算側影係數
from sklearn import metrics
```

```
In [6]: # 載入資料
df = pd.read_csv('investor_portfolio.csv')
```

```
In [7]: # 檢視資料
df
```

```
Out[7]:
```

	ID	AGE	EDUC	MARRIED	KIDS	LIFEC	OCCAT	RISK	HHOUSE	WSAVE
0	1	3	2	1	0	2	1	3	1	
1	2	4	4	1	2	5	2	3	0	
2	3	3	1	1	2	3	2	2	1	
3	4	3	1	1	2	3	2	2	1	
4	5	4	3	1	1	5	1	2	1	
...	...	...	...	...	...	...	...	...	...	
3861	3862	3	1	1	1	3	1	4	0	
3862	3863	3	1	1	1	3	1	4	0	
3863	3864	5	1	1	0	5	1	4	0	
3864	3865	2	4	1	7	3	1	3	1	
3865	3866	3	4	2	0	1	2	2	1	

3866 rows × 13 columns



## 2.2 資料摘要

```
In [8]: # 資料物件
type(df)
```

```
Out[8]: pandas.core.frame.DataFrame
```

```
In [9]: # 資料型態
df.dtypes # 資料皆為整數
```

```
Out[9]: ID          int64
AGE          int64
EDUC         int64
MARRIED      int64
KIDS         int64
LIFECL       int64
OCCAT        int64
RISK         int64
HHOUSES      int64
WSAVED       int64
SPENDMOR     int64
NWCAT        int64
INCCL        int64
dtype: object
```

```
In [10]: # 資料摘要
pd.set_option('display.precision', 2)
df.describe(include='all')
```

Out[10]:

	ID	AGE	EDUC	MARRIED	KIDS	LIFECL	OCCAT	RISK	HH
count	3866.00	3866.00	3866.00	3866.00	3866.00	3866.00	3866.00	3866.00	3866.00
mean	1933.50	3.11	2.91	1.35	0.94	3.70	1.74	3.04	3.04
std	1116.16	1.51	1.07	0.48	1.25	1.62	0.93	0.88	0.88
min	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
25%	967.25	2.00	2.00	1.00	0.00	3.00	1.00	2.00	2.00
50%	1933.50	3.00	3.00	1.00	0.00	3.00	1.00	3.00	3.00
75%	2899.75	4.00	4.00	2.00	2.00	5.00	3.00	4.00	4.00
max	3866.00	6.00	4.00	2.00	8.00	6.00	4.00	4.00	4.00

欄位說明:  
[https://github.com/rwepa/DataDemo/blob/master/README.md#investor\\_portfoliocsv](https://github.com/rwepa/DataDemo/blob/master/README.md#investor_portfoliocsv)

- 三大屬性:
- 1.人口統計屬性:6個
  - 2.財務屬性:4個
  - 3.行為屬性:2個
- 欄位名稱:

ID : 編號

AGE : 年齡, 6個類別, 1:35歲以下(不含), 2:35-45, 3:45-55, 4:55-65, 5:65-75, 6:75歲以上(含)【人口統計屬性】

EDUC : 教育程度, 1:高中以下, 4:大學【人口統計屬性】

MARRIED : 是否已婚, 1:已婚, 2:未婚【人口統計屬性】

KIDS : 子女數【人口統計屬性】

LIFECL : 生命週期, 6個類別, 1:35歲以下,未婚,無子女; 6:55歲以上,無工作【人口統計屬性】

OCCAT : 職業(occupation category), 1:管理職, 4:失業【人口統計屬性】

HHOUSE : 自有房屋, 1:有, 0:沒有【財務屬性】

WSAVED : 支出與收入類別, 3個類別, 1:支出>收入, 2:支出=收入, 3:支出<收入【財務屬性】

NWCAT : 淨值類別(net worth category), 5個類別, 1:淨值低於25百分位數, 5:淨值高於90百分位數【財務屬性】

INCCL : 收入類別(income category), 5個類別, 1:收入低於1萬元, 5:收入超過10萬元【財務屬性】

RISK : 願意承受風險的程度, 1:願意承受風險的程度最高, 4:願意承受風險的程度最低【行為屬性】, 本屬性可考量為反應變數。

SPENDMOR: 支出偏好, 5個類別, 5:最高支出偏好【行為屬性】

## 2.3 探索性資料分析(Exploratory Data Analysis, EDA)

```
In [11]: # 資料列數與行數  
df.shape # 3866*13
```

```
Out[11]: (3866, 13)
```

```
In [12]: # 欄位名稱  
df.columns
```

```
Out[12]: Index(['ID', 'AGE', 'EDUC', 'MARRIED', 'KIDS', 'LIFECL', 'OCCAT', 'RISK',  
              'HHOUSES', 'WSAVED', 'SPENDMOR', 'NWCAT', 'INCCL'],  
              dtype='object')
```

```
In [13]: # 檢查NA值  
df.isnull().sum() # 所有變數皆沒有NA值
```

```
Out[13]: ID          0
         AGE          0
         EDUC          0
         MARRIED       0
         KIDS          0
         LIFECL        0
         OCCAT         0
         RISK          0
         HHOUSES       0
         WSAVED        0
         SPENDMOR       0
         NWCAT         0
         INCCL         0
         dtype: int64
```

```
In [14]: # 顯示前5筆
         df.head()
```

```
Out[14]:
```

	ID	AGE	EDUC	MARRIED	KIDS	LIFECL	OCCAT	RISK	HHOUSES	WSAVED	SPE
0	1	3	2	1	0	2	1	3	1	1	
1	2	4	4	1	2	5	2	3	0	2	
2	3	3	1	1	2	3	2	2	1	2	
3	4	3	1	1	2	3	2	2	1	2	
4	5	4	3	1	1	5	1	2	1	3	

```
In [15]: # 顯示後5筆
         df.tail()
```

```
Out[15]:
```

	ID	AGE	EDUC	MARRIED	KIDS	LIFECL	OCCAT	RISK	HHOUSES	WSAVEI	
3861	3862	3	1	1	1	3	1	4	0		
3862	3863	3	1	1	1	3	1	4	0		
3863	3864	5	1	1	0	5	1	4	0		
3864	3865	2	4	1	7	3	1	3	1		
3865	3866	3	4	2	0	1	2	2	1		

## 2.4 資料視覺化

```
In [16]: # 計算相關係數
         mydf = df.drop(['ID'], axis=1) # axis=1 表示行
         mydf
```

Out[16]:

	AGE	EDUC	MARRIED	KIDS	LIFECL	OCCAT	RISK	HHOUSES	WSAVED	SPE
0	3	2	1	0	2	1	3	1	1	
1	4	4	1	2	5	2	3	0	2	
2	3	1	1	2	3	2	2	1	2	
3	3	1	1	2	3	2	2	1	2	
4	4	3	1	1	5	1	2	1	3	
...	...	...	...	...	...	...	...	...	...	...
3861	3	1	1	1	3	1	4	0	2	
3862	3	1	1	1	3	1	4	0	2	
3863	5	1	1	0	5	1	4	0	3	
3864	2	4	1	7	3	1	3	1	3	
3865	3	4	2	0	1	2	2	1	2	

3866 rows × 12 columns



In [17]:

```
pd.set_option('display.precision',4)
correlation = mydf.corr()
correlation
```

Out[17]:

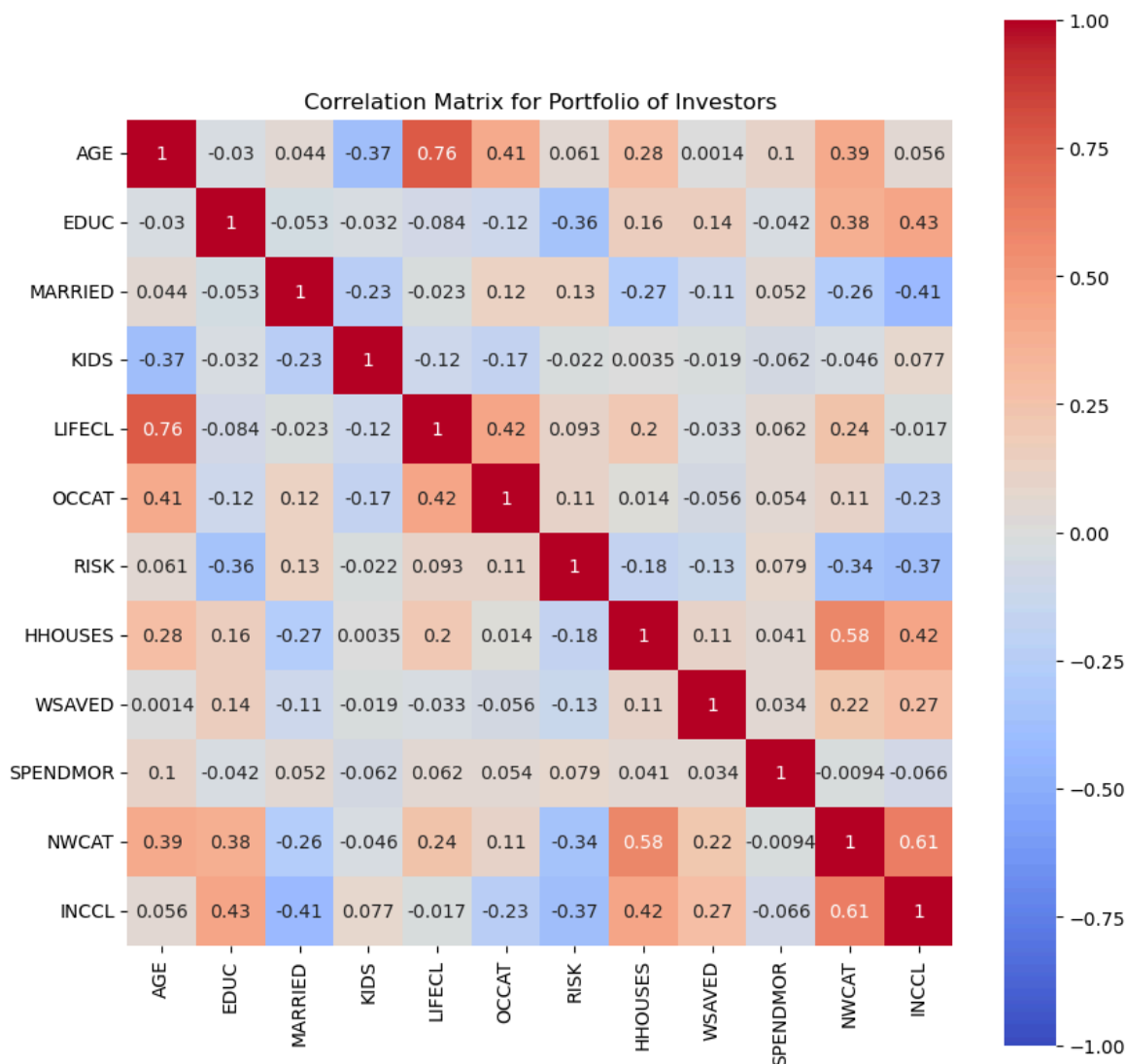
	AGE	EDUC	MARRIED	KIDS	LIFECL	OCCAT	RISK	HHOUSES	WSAVED	SPENDMOR	NWCAT	INCCCL
AGE	1.0000	-0.0299	0.0438	-0.3713	0.7636	0.4127	0.0605	0.2841	0.0014	0.1009	0.3906	0.0561
EDUC	-0.0299	1.0000	-0.0528	-0.0322	-0.0844	-0.1162	-0.3572	0.1622	0.1431	-0.0422	0.3819	0.4318
MARRIED	0.0438	-0.0528	1.0000	-0.2326	-0.0230	0.1193	0.1344	-0.2698	-0.1064	0.0517	-0.2640	-0.4144
KIDS	-0.3713	-0.0322	-0.2326	1.0000	-0.1168	-0.1670	-0.0223	0.0035	-0.0191	-0.0621	-0.0464	0.0771
LIFECL	0.7636	-0.0844	-0.0230	-0.1168	1.0000	0.4213	0.0928	0.1976	-0.0328	0.0625	0.2441	-0.0174
OCCAT	0.4127	-0.1162	0.1193	-0.1670	0.4213	1.0000	0.1070	0.0140	-0.0557	0.0536	0.1134	-0.2300
RISK	0.0605	-0.3572	0.1344	-0.0223	0.0928	0.1070	1.0000	-0.1775	-0.1326	0.0787	-0.3409	-0.3747
HHOUSES	0.2841	0.1622	-0.2698	0.0035	0.1976	0.0140	-0.1775	1.0000	0.1070	0.0787	-0.3409	-0.3747
WSAVED	0.0014	0.1431	-0.1064	-0.0191	-0.0328	-0.0557	-0.1326	0.1070	1.0000	0.0787	-0.3409	-0.3747
SPENDMOR	0.1009	-0.0422	0.0517	-0.0621	0.0625	0.0536	0.0787	0.0787	0.0787	1.0000	0.5833	0.4171
NWCAT	0.3906	0.3819	-0.2640	-0.0464	0.2441	0.1134	-0.3409	0.5833	0.5833	0.5833	1.0000	0.4171
INCCCL	0.0561	0.4318	-0.4144	0.0771	-0.0174	-0.2300	-0.3747	0.4171	0.4171	0.4171	0.4171	1.0000



In [18]:

```
# seaborn galley: https://seaborn.pydata.org/examples/index.html
plt.figure(figsize=(10,10))
plt.title('Correlation Matrix for Portfolio of Investors')
sns.heatmap(correlation, square=True, vmin = -1, vmax = 1, center = 0, annot=True)
```

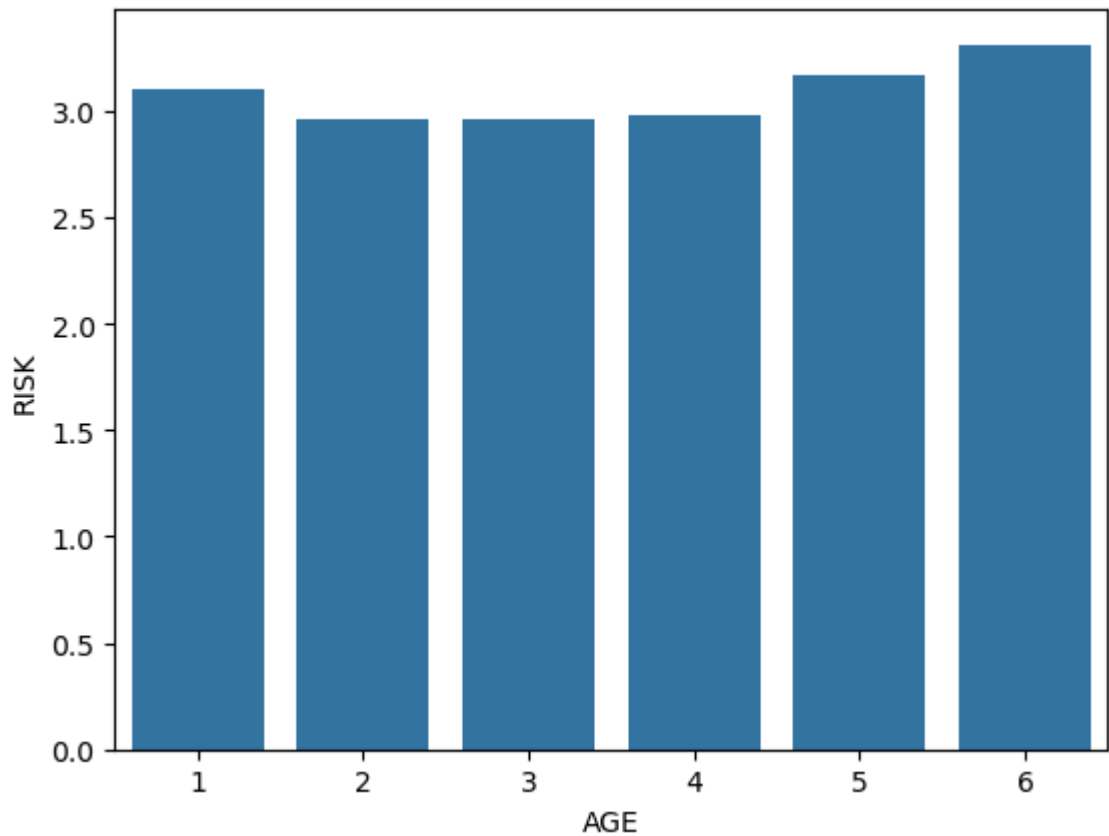
Out[18]: <Axes: title={'center': 'Correlation Matrix for Portfolio of Investors'}>



上方相關係數圖顯示以下之特質: 1.風險承受能力(RISK)與淨值類別(NWCAT)與)呈現負相關。例: RISK為1之願意承受風險的程度最高者，其淨值類別(NWCAT)為較高5。2.風險承受能力(RISK)與INCCL(收入類別相關)呈現負相關。3.風險承受能力(RISK)與孩子數目 (KIDS)呈現負相關。直觀。

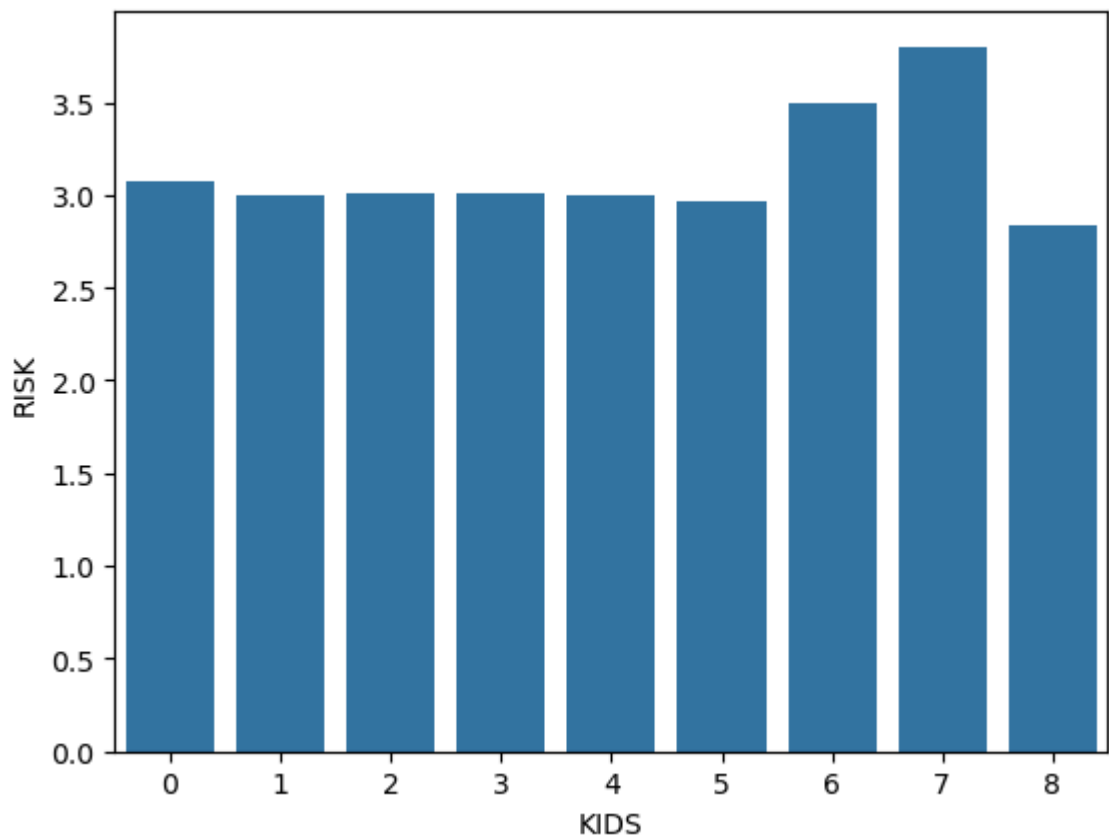
```
In [19]: # 平均風險承受能力(RISK)VS年齡(AGE)長條圖
# 結果顯示風險承受能力先下降再提升
mydf = df
mydf["AGE"] = mydf["AGE"].astype("category")
sns.barplot(x='AGE', y='RISK', data=mydf, errorbar=None)
```

Out[19]: <Axes: xlabel='AGE', ylabel='RISK'>



```
In [20]: # 平均風險承受能力(RISK)VS子女數(KIDS)長條圖
# 結果顯示子女數(KIDS)約7個時達平均風險承受能力最大值(平均風險承受能力愈低)
mydf["KIDS"] = mydf["KIDS"].astype("category")
sns.barplot(x='KIDS', y='RISK', data=mydf, errorbar=None)
```

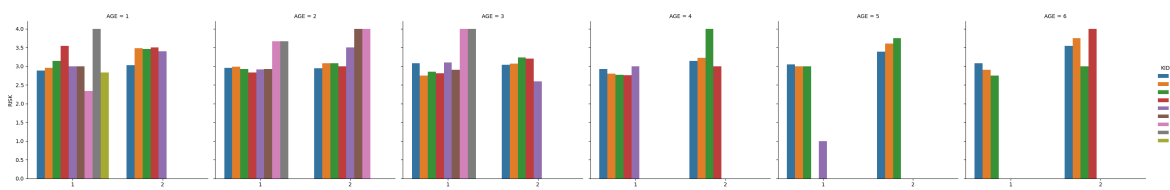
```
Out[20]: <Axes: xlabel='KIDS', ylabel='RISK'>
```





```
In [21]: # 說明下圖之結果
mydf["MARRIED"] = mydf["MARRIED"].astype("category")
#sns.catplot(x="MARRIED", y="RISK", hue="KIDS", col="AGE", data=mydf, kind="bar")
sns.catplot(x="MARRIED", y="RISK", hue="KIDS", col="AGE", data=mydf, kind="bar",
```

```
Out[21]: <seaborn.axisgrid.FacetGrid at 0x20f92ef5430>
```



### 3.資料準備

資料準備主要工作是將資料隨機區分為二大類：訓練集 ( train dataset ) 與測試集 ( test dataset ) 為主。建立型模可以先考慮要使用非監督式學習方法，資料暫先不用區分訓練集與測試集。

### 4.建立模型

建立模型方法包括推論統計、機器學習、深度學習與生成式學習等方法。本研究使用非監督式學習 K-means 集群法。

```
In [22]: # 先刪除沒有使用的ID變數
# 因資料的差異性不大，且具有相似值，因此先不用進行資料轉換
X=df.drop(['ID'], axis=1)
X # 3866 × 12
```

```
Out[22]:
```

	AGE	EDUC	MARRIED	KIDS	LIFECL	OCCAT	RISK	HHOUSE	WSAVED	SPE
0	3	2	1	0	2	1	3	1	1	
1	4	4	1	2	5	2	3	0	2	
2	3	1	1	2	3	2	2	1	2	
3	3	1	1	2	3	2	2	1	2	
4	4	3	1	1	5	1	2	1	3	
...	...	...	...	...	...	...	...	...	...	...
3861	3	1	1	1	3	1	4	0	2	
3862	3	1	1	1	3	1	4	0	2	
3863	5	1	1	0	5	1	4	0	3	
3864	2	4	1	7	3	1	3	1	3	
3865	3	4	2	0	1	2	2	1	2	

3866 rows × 12 columns



## 4.1 K-means 集群法

K-means 是一種迭代聚類算法，用於將  $n$  個數據點劃分為  $k$  個聚類。算法的目標是最小化所有點與其所屬聚類中心之間的平方距離之和，流程如下：

1. 初始化集群中心：首先隨機選擇  $K$  個數據點作為初始聚類中心。
2. 分配樣本到最近的集群中心：對於每個樣本，計算其與每個集群中心的距離，將樣本分配到距離最近的集群中心所對應的集群。(第1次迭代)
3. 更新集群中心：對每個集群，計算其所有樣本的平均值，將平均值作為新的集群中心。(第2次迭代, 第3次迭代....)
4. 重複步驟2和步驟3：重複執行步驟2和步驟3，直到集群心不再變化或達到預定的迭代次數。
5. 結果：最終的集群中心即為集群結果，每個樣本被分配到對應的一個集群。由於初始集群中心的隨機性，演算法可能收斂到不同的解。集群內(inner)距離平和將達到最小值，集群間(between)距離平和將達到最大值。斂到不同的解。

```
In [23]: # K-means 集群法示範

# https://github.com/rwepa/r\_data\_scientist/blob/main/r\_kmeans\_animation.gif

# R 程式碼: https://github.com/rwepa/r\_data\_scientist/blob/main/r\_kmeans\_animation
```

評估模型集群個數的方法包括:

1. 陡坡圖 ( Scree plot )
2. 側影係數 ( Silhouette coefficient )

## 4.2 方法1 群內距離平方和陡坡圖 ( Scree plot )

繪製群內距離平方和陡坡圖 (Sum of square errors within clusters, 簡稱 SSE) · 降低最明顯之  $K$  值為集群數。

群內距離平方和陡坡圖又稱為手肘法 ( Elbow method )

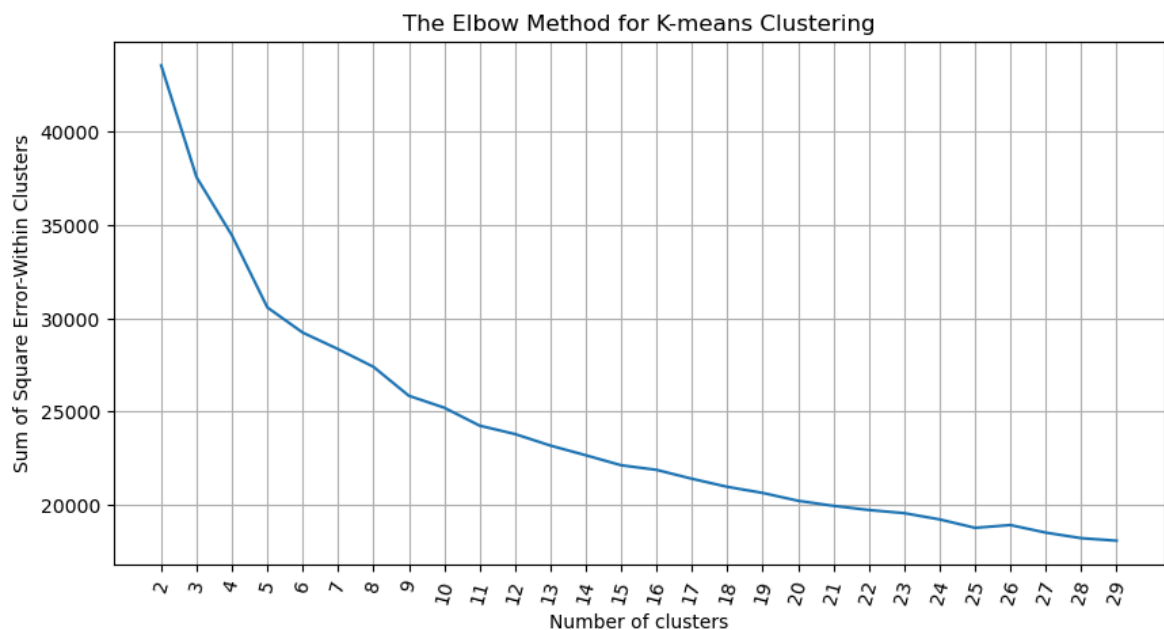
```
In [24]: # 計算群內 SSE
distorsions = []
max_loop=30

for k in range(2, max_loop):
    k_means = KMeans(n_clusters=k)
    k_means.fit(X)
    distorsions.append(k_means.inertia_)

distorsions
```

```
Out[24]: [43571.11802589656,
37573.44812460114,
34453.00814540151,
30591.932011554854,
29240.93857059658,
28354.82556578546,
27403.581446774857,
25851.62367190421,
25211.696035379602,
24249.10321006414,
23798.744142458094,
23180.669612245532,
22662.502475017336,
22125.19785965468,
21878.33166261029,
21400.35199415213,
20967.939638634867,
20640.957319814213,
20219.692037899153,
19947.35245234775,
19724.51505016489,
19556.639260722874,
19221.14029457558,
18769.33093021637,
18917.407669122786,
18512.29714933016,
18214.575225852444,
18078.67271989965]
```

```
In [25]: # 群內距離平方和陡坡圖
fig = plt.figure(figsize=(10, 5))
plt.plot(range(2, max_loop), distortions)
plt.xticks([i for i in range(2, max_loop)], rotation=75)
plt.xlabel("Number of clusters")
plt.ylabel("Sum of Square Error-Within Clusters")
plt.title("The Elbow Method for K-means Clustering")
plt.grid(True)
```



## 4.3 方法2 使用側影係數 ( Silhouette coefficient )

- 側影係數又稱為輪廓分數 ( Silhouette Score ) 。
- 在機器學習與數據挖掘領域，輪廓指的是一種反映數據聚類結果一致性的方法。
- 側影係數衡量一個點與它自己的集群 ( 凝聚力 ) 相比於其他集群 ( 分離 ) 的相似程度。
- 側影係數值的範圍在 1 到 -1 之側影係數越高，表示該點越適合在該集群之中。
- 如果側影係數的輪值廓大部分是負值，則可能是建立了太多或太少的集群。

參考[https://en.wikipedia.org/wiki/Silhouette\\_\(clustering\)](https://en.wikipedia.org/wiki/Silhouette_(clustering))

步驟1 計算集群內平均距離  $a(i)$  。

步驟2 計算最鄰近集群間平均距離  $b(i)$  。

步驟3 計算每個資料點 $i$ 側影係數：

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

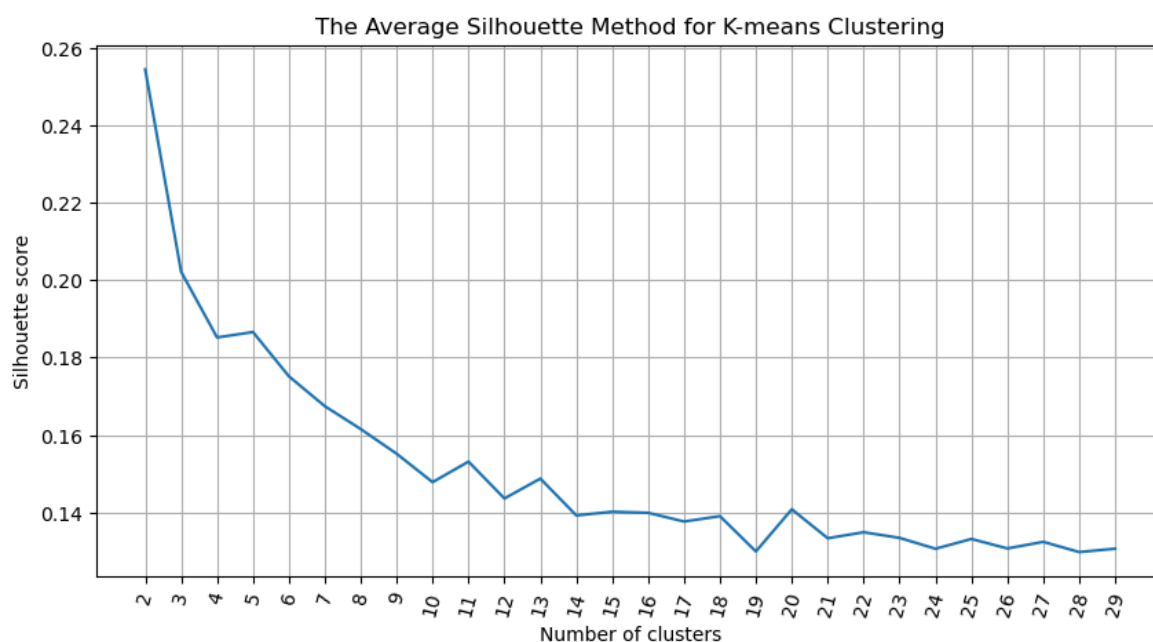
$$\text{上式可以以改寫為：} s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \text{ 且 } -1 \leq s(i) \leq 1 \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

```
In [26]: # 使用 metrics.silhouette_score 函數計算側影係數
silhouette_score = []
for k in range(2, max_loop):
    kmeans = KMeans(n_clusters=k, random_state=10, n_init=10)
    kmeans.fit(X)
    silhouette_score.append(metrics.silhouette_score(X, kmeans.labels_, rand

silhouette_score
```

```
Out[26]: [0.2544089264150241,
0.20215644186231574,
0.18520483889712674,
0.186603624753214,
0.17517140818856253,
0.16746166685459618,
0.16155207527816903,
0.1551748437010208,
0.14784508431917154,
0.15315189978565352,
0.1436352684912865,
0.14877494266375477,
0.13925334511512985,
0.14021996843343293,
0.13993697049851966,
0.13768471192693238,
0.13906710967733774,
0.12995709613601597,
0.1408399034974352,
0.13336220014342093,
0.13491820920833048,
0.13345852780628376,
0.13065159608112403,
0.13318099404221184,
0.1307303614993667,
0.13243683876201995,
0.12979897661650927,
0.13067306278318006]
```

```
In [27]: # 側影係數圖
fig = plt.figure(figsize=(10, 5))
plt.plot(range(2, max_loop), silhouette_score)
plt.xticks([i for i in range(2, max_loop)], rotation=75)
plt.xlabel("Number of clusters")
plt.ylabel("Silhouette score")
plt.title("The Average Silhouette Method for K-means Clustering")
plt.grid(True)
```

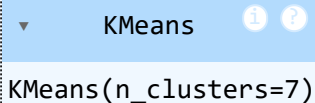


觀察上面二個圖形，最佳集群數約K=7。

## 4.4 K-means 集群預測

```
In [28]: # 設定集群數  
nclust=7
```

```
In [29]: # 建立 k-means 集群法且 K=7  
k_means = cluster.KMeans(n_clusters=nclust)  
k_means.fit(X)
```

```
Out[29]:  KMeans  
KMeans(n_clusters=7)
```

```
In [30]: # 預測分群結果, 使用 predict 函數  
target_labels = k_means.predict(X)  
target_labels
```

```
Out[30]: array([5, 1, 2, ..., 3, 2, 2])
```

## 5. 評估與測試

```
In [31]: # K-means 集群-評估  
print("km", metrics.silhouette_score(X, k_means.labels_, metric='euclidean'))  
  
km 0.16268933856065337
```

## 6. 佈署應用與結論

```
In [32]: # 將變數轉換為原來整數, 本例使用 int64.  
X["MARRIED"] = X["MARRIED"].astype("int64")  
X["KIDS"] = X["KIDS"].astype("int64")  
X["AGE"] = X["AGE"].astype("int64")  
X.dtypes
```

```
Out[32]: AGE          int64  
EDUC          int64  
MARRIED       int64  
KIDS          int64  
LIFECL        int64  
OCCAT         int64  
RISK          int64  
HHOUSES       int64  
WSAVED        int64  
SPENDMOR      int64  
NWCAT         int64  
INCCL         int64  
dtype: object
```

```
In [33]: # 繪製各集群重要變數的長條圖  
cluster_output = pd.concat([pd.DataFrame(X), pd.DataFrame(k_means.labels_, column  
cluster_output
```

Out[33]:

	AGE	EDUC	MARRIED	KIDS	LIFECL	OCCAT	RISK	HHOUSES	WSAVED	SPE
0	3	2	1	0	2	1	3	1	1	
1	4	4	1	2	5	2	3	0	2	
2	3	1	1	2	3	2	2	1	2	
3	3	1	1	2	3	2	2	1	2	
4	4	3	1	1	5	1	2	1	3	
...	...	...	...	...	...	...	...	...	...	...
3861	3	1	1	1	3	1	4	0	2	
3862	3	1	1	1	3	1	4	0	2	
3863	5	1	1	0	5	1	4	0	3	
3864	2	4	1	7	3	1	3	1	3	
3865	3	4	2	0	1	2	2	1	2	

3866 rows × 13 columns

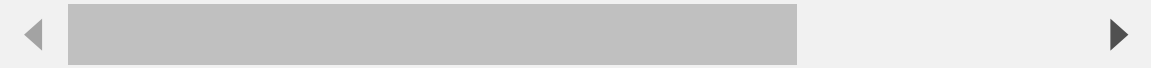


In [34]:

```
# 計算7個集群之變數平均值
output=cluster_output.groupby('cluster').mean()
output
```

Out[34]:

	AGE	EDUC	MARRIED	KIDS	LIFECL	OCCAT	RISK	HHOUSES	WSAV
cluster									
0	1.9106	2.4674	1.3147	2.1285	3.3371	1.4283	3.3613	0.5326	2.25
1	4.6559	3.3498	1.2452	0.3156	5.3460	2.0475	2.8574	0.9468	2.63
2	2.4529	3.5394	1.1342	1.5652	2.7406	1.4439	2.4787	0.9432	2.71
3	4.8821	2.0701	1.5949	0.1846	5.6598	2.4410	3.5846	0.6821	2.23
4	4.5995	3.4877	1.2452	0.3678	5.3243	2.0463	2.7166	0.9319	2.56
5	1.9663	2.8972	1.6166	0.0015	1.3850	1.5169	3.1457	0.4601	2.40
6	1.9127	2.4198	1.2925	2.0259	3.2925	1.4198	3.2783	0.5047	2.21

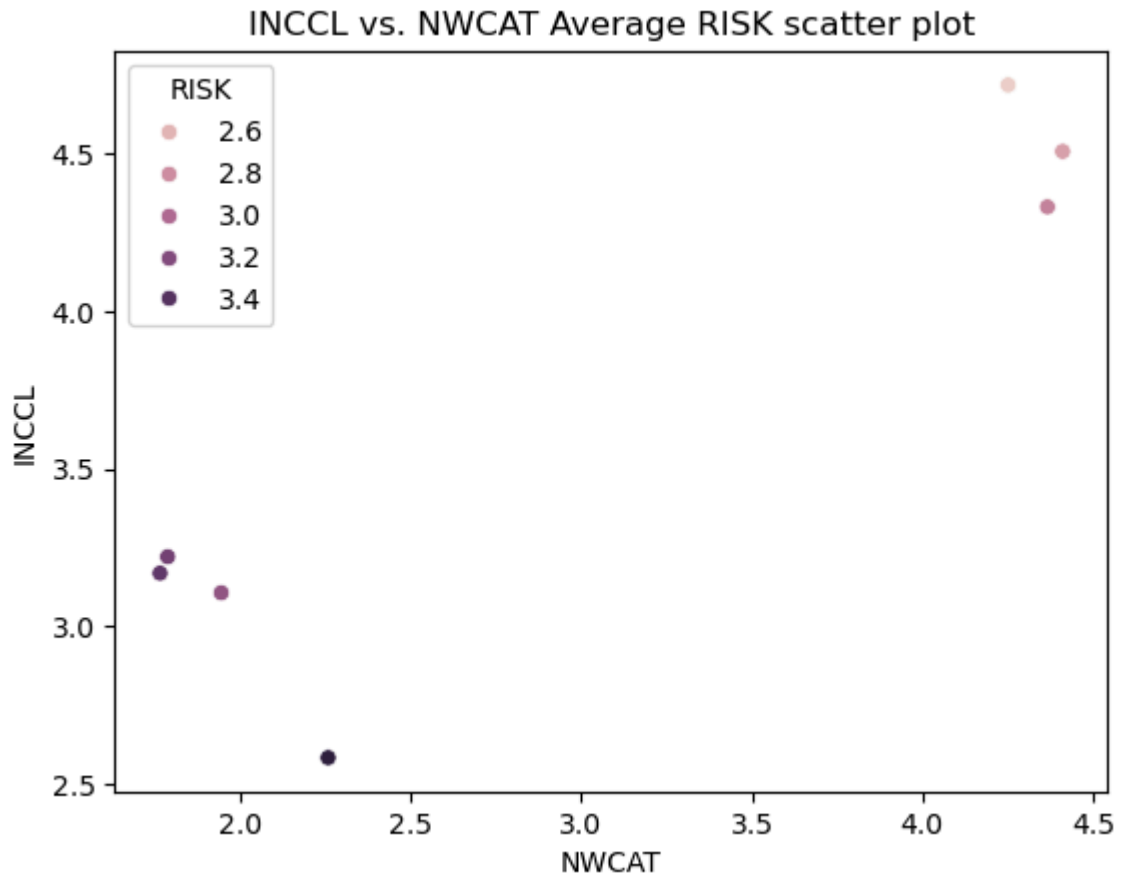


In [35]:

```
# 繪製集群散佈圖
ax = sns.scatterplot(data=output, x="NWCAT", y="INCCL", hue="RISK")
ax.set_title('INCCL vs. NWCAT Average RISK scatter plot')

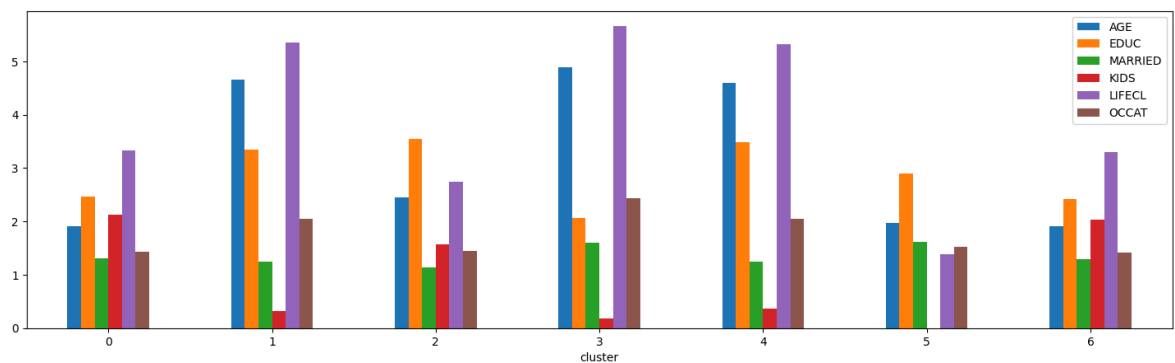
# 結果顯示不同 RISK 的散佈圖。
```

Out[35]: Text(0.5, 1.0, 'INCCL vs. NWCAT Average RISK scatter plot')



In [36]: `# 依據人口統計特徵(6個變數)繪製長條圖`  
`output[['AGE', 'EDUC', 'MARRIED', 'KIDS', 'LIFECL', 'OCCAT']].plot.bar(rot=0, figsize=`

Out[36]: `<Axes: xlabel='cluster'>`



結論: 1.針對集群 0 與 1，集群 0 的平均年齡 ( AGE ) 較低，但集群 1 有較高平均教育程度 ( EDUC )。2.就婚姻 ( MARRIED ) 和子女數量 ( KIDS ) 而言，這兩個集是相似的。3.根據人口統計屬性，集群 0 中的個體平均具有更高的風險承能力。

In [37]: `# end`

In [ ]: