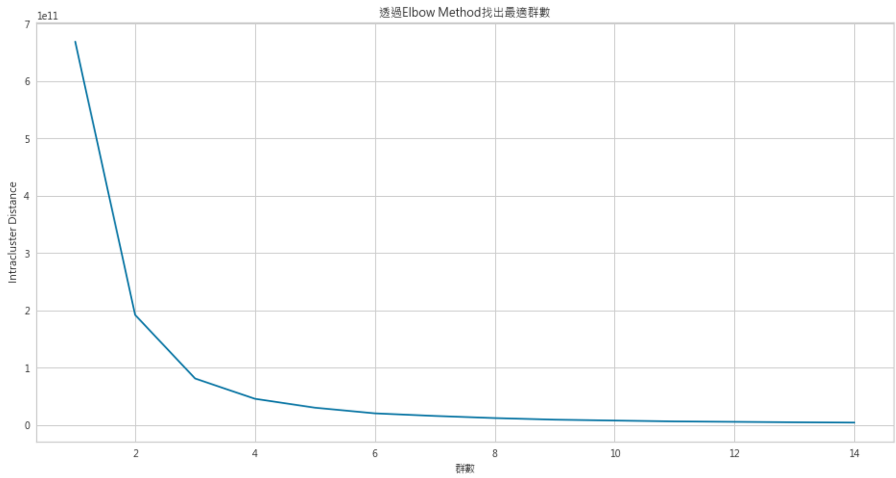
ML Assignment 2 – Clustering

309709022 陳政廷

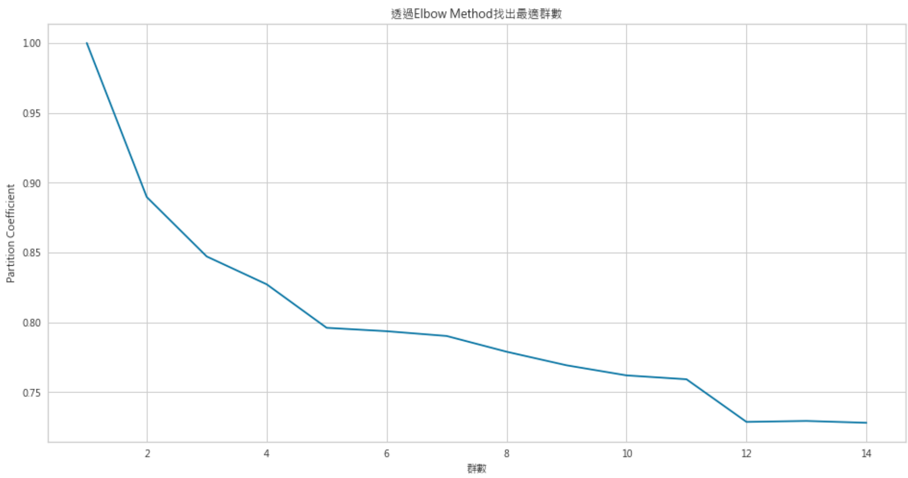
A. Diamonds(程式碼附在最後)

1) Flat – K-Means and Fuzzy C-Means Clustering Algorithms

K-Means：透過 elbow method 找出最佳群數為 3

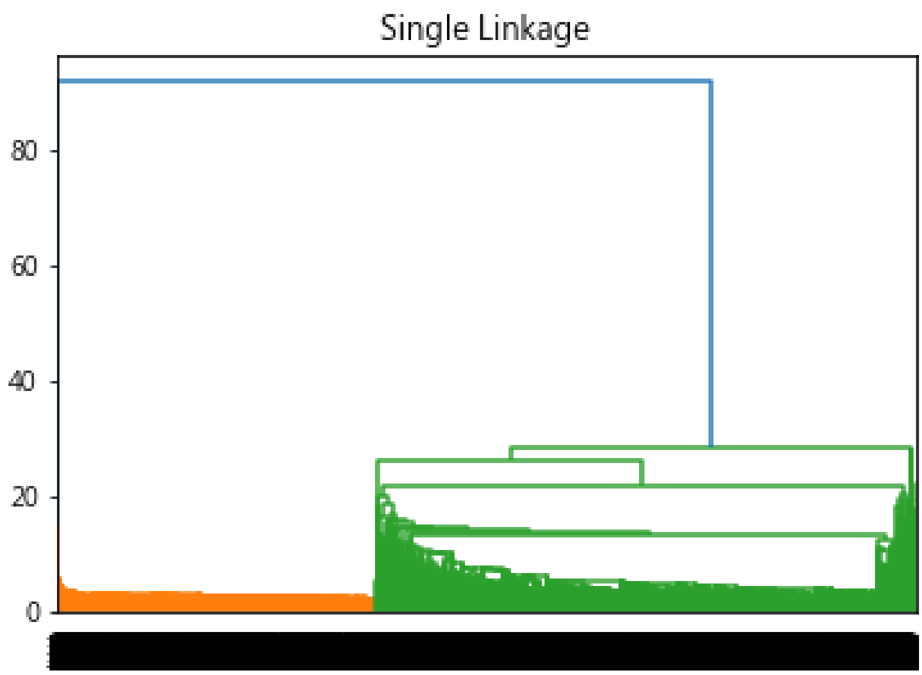


Fuzzy C-Means：透過 elbow method 找出最佳群數為 5



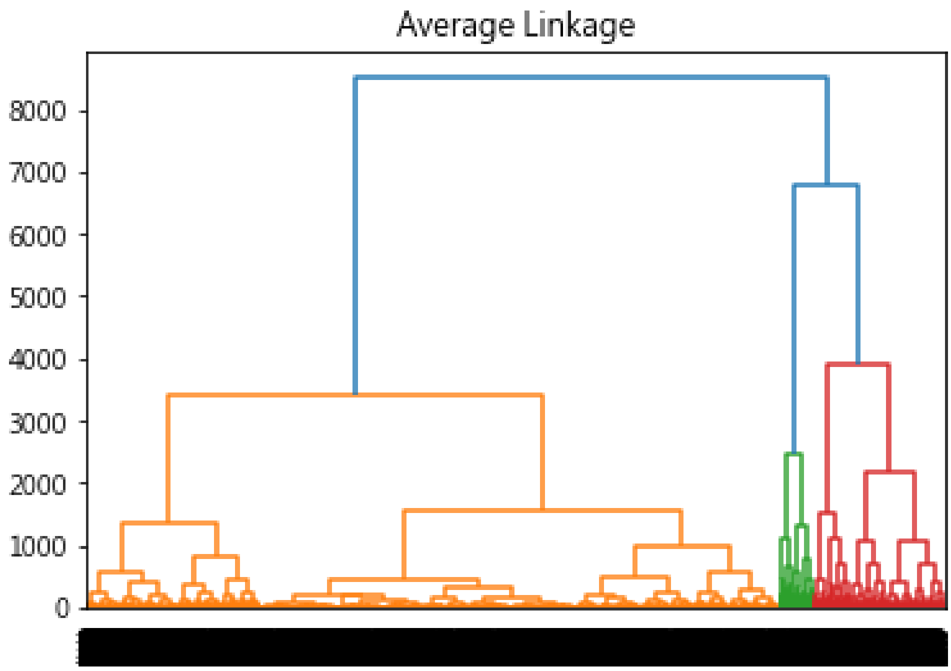
2) Single Linkage, Complete Linkage, and Average Linkage Clustering Algorithms

Single Linkage：由於無法從 Dendrogram 看出(無法看到左邊狀況)，故引用 Silhouette Coefficient，其算法如下所示，整體介於-1~1 之間且數值越大代表分的狀況越好，故根據其結果分為 2 群

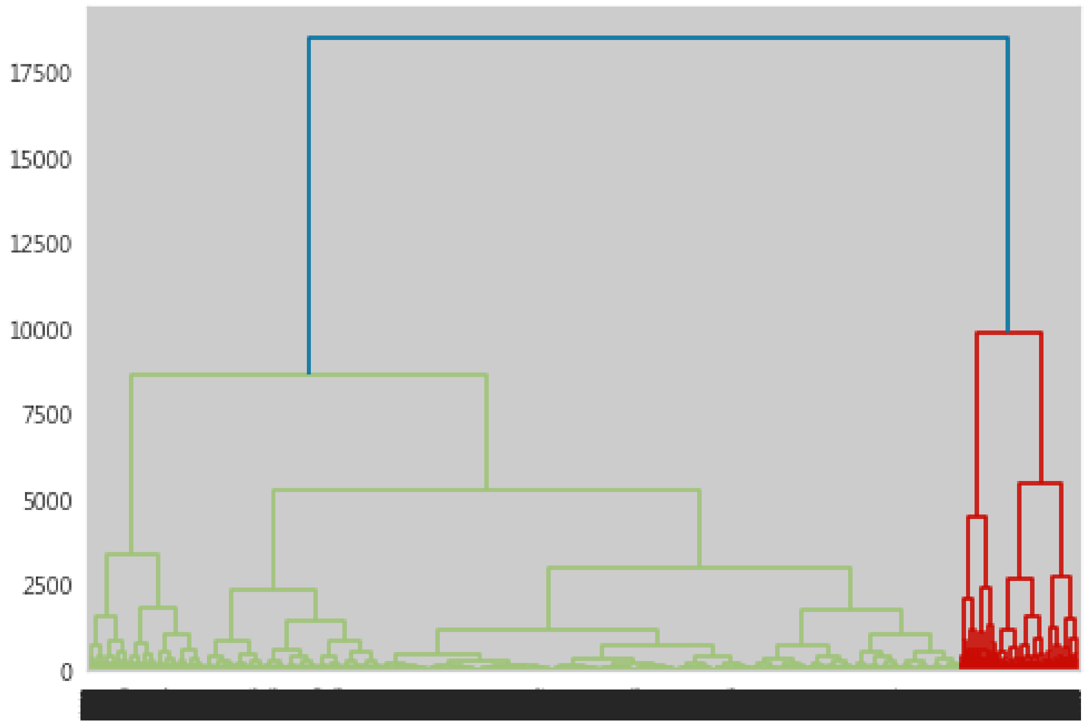


群數	silhouette_score
2	0.337358
3	0.294777
4	0.286415
5	0.287363
6	0.287256
7	0.315677
8	0.307199
9	0.069631
10	0.066485

Average Linkage：根據 Dendrogram 結果分為 5 群



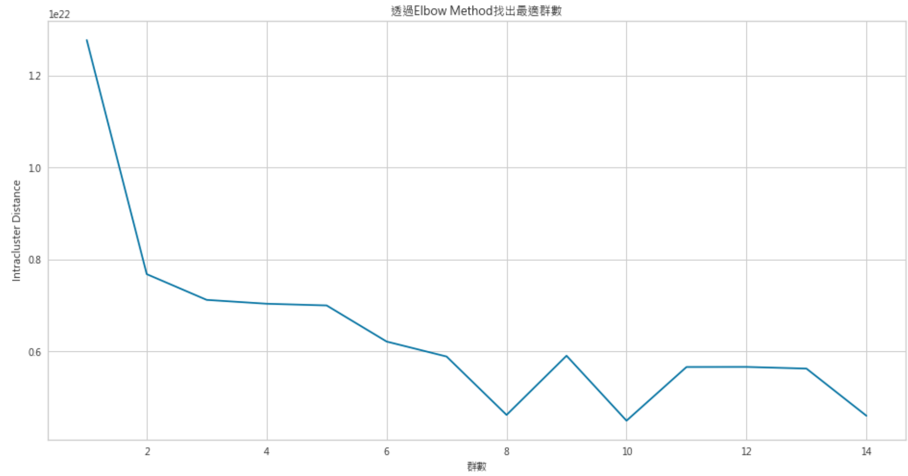
Complete Linkage：根據 Dendrogram 結果分為 4 群



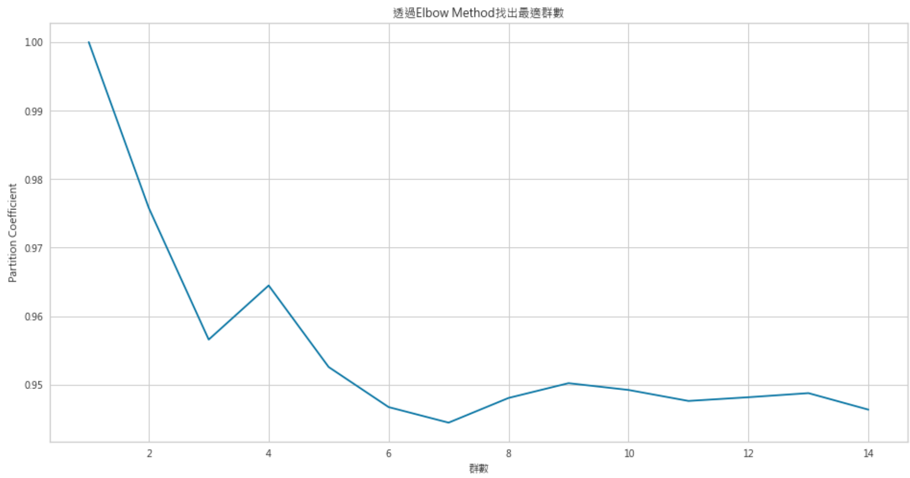
B. Bankruptcy

1) Flat – K-Means and Fuzzy C-Means Clustering Algorithms

K-Means：根據 elbow method 分為 2 群

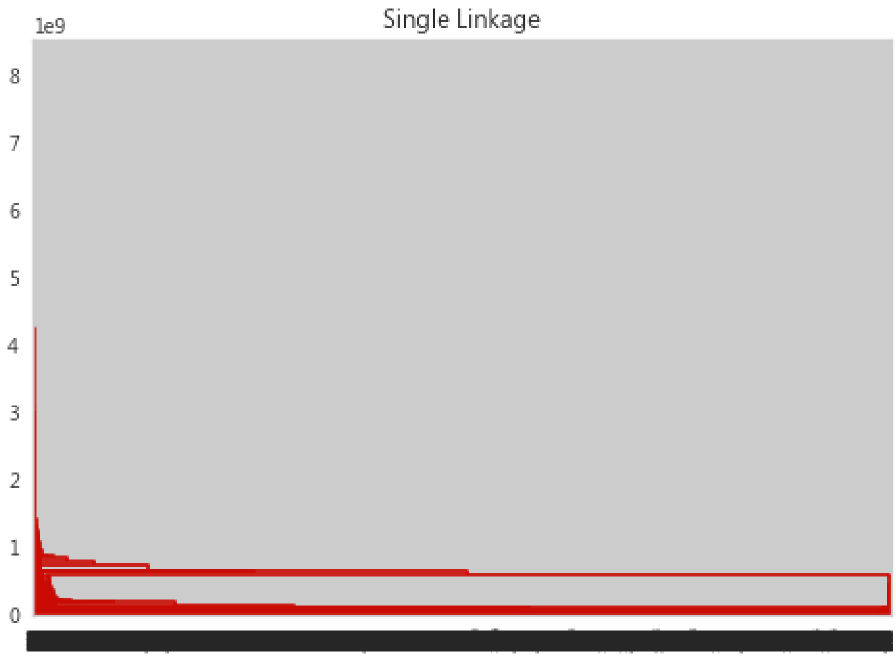


Fuzzy-C-Means：根據 elbow method 分為 3 群



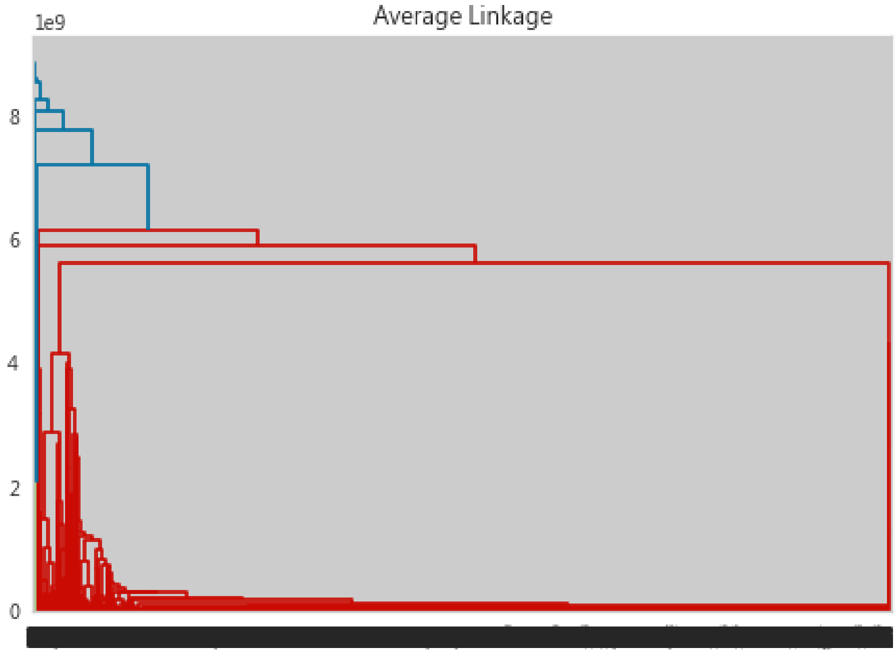
2) Single Linkage, Complete Linkage, and Average Linkage Clustering Algorithms

Single Linkage：由於 Dendrogram 結果無法判別故由 Silhouette Coefficient 結果分為 4 群



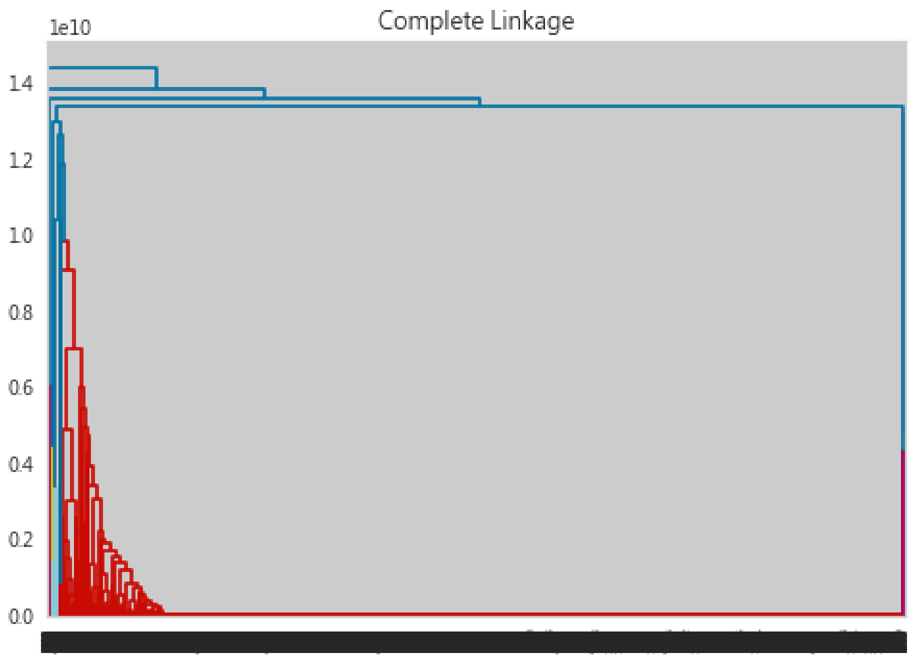
群數	silhouette_score
2	0.937375
3	0.935834
4	0.936411
5	0.934745
6	0.934645
7	0.921423
8	0.920229
9	0.920839
10	0.882032

Average Linkage：由於 Dendrogram 結果無法判別故由 Silhouette Coefficient 結果分為 4 群



群數	silhouette_score
2	0.924911
3	0.924472
4	0.923546
5	0.923474
6	0.922410
7	0.921860
8	0.920150
9	0.920075
10	0.914120

Complete Linkage：由於 Dendrogram 結果無法判別故由 Silhouette Coefficient 結果分為 7 群



群數	silhouette_score
2	0.939426
3	0.936074
4	0.923652
5	0.927134
6	0.927703
7	0.928568
8	0.926538
9	0.926975
10	0.922751

C.程式碼

```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 28 00:20:14 2022

@author: Tim Chen
"""

#################
#  載入套件&data #
#################
import numpy as np
import pandas as pd
import skfuzzy as fuzz
from sklearn import cluster
from sklearn import metrics
from kneed import KneeLocator
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as sch
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import AgglomerativeClustering
plt.rcParams['font.sans-serif'] = ['Microsoft JhengHei']
plt.rcParams['axes.unicode_minus'] = False


#%%
#################
# diamonds(reg) #
#################

#  讀取資料
reg_data = pd.read_csv("diamonds_trainingset.csv", encoding= "utf-8-sig")
reg_data = reg_data[[i for i in reg_data.columns if "Unnamed" not in i]]
reg_data.dtypes

#  將類別資料欄為轉為 one hot
color_dummy = pd.get_dummies(reg_data["color"],prefix = "color_")
```

```
reg_data = reg_data.drop(columns = ["color"])
reg_data = pd.concat([reg_data, color_dummy], axis = 1)
reg_cluster = reg_data.copy()

###########
# K-Means #
###########

# 透過 Elbow method：分別查看每個資料集被分成 1~15 群的狀況，來決定最佳分群數
kmeans_distortions = []
kmeans_label_ls = []
K = range(1,15)
for k in K:
    kmeanModel = cluster.MiniBatchKMeans(n_clusters = k,random_state = 0,max_iter =
1000)
    kmeanModel.fit(reg_cluster)
    kmeans_distortions.append(kmeanModel.inertia_)
    kmeans_label_ls.append(kmeanModel.labels_)
plt.figure(figsize = (16,8))
plt.plot(K, kmeans_distortions, 'bx-')
plt.xlabel('群數')
plt.ylabel('Intracluster Distance')
plt.title('透過 Elbow Method 找出最適群數')
plt.savefig("./Kmeans 決定最佳分群數(diamonds).png")
plt.show()
kn = KneeLocator([i for i in range(1, 15)], kmeans_distortions, curve = 'convex', direction =
'decreasing')
print("根據 elbow method 找出的最適群數為  ", kn.knee)

# 輸出上述分群結果
reg_data["K-Means_label"] = kmeans_label_ls[kn.knee-1]


#################
# Fuzzy C-Means #
#################

# 透過 Elbow method：分別查看每個資料集被分成 1~15 群的狀況，來決定最佳分群數
```

```
cmeans_coefficient = []
cmeans_label_ls = []
C = range(1,15)
for c in C:
    cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(reg_cluster.to_numpy().transpose(),
                                                                                   c, 2, error = 0.005, maxiter =
1000, init = None, seed = 0)
    cmeans_coefficient.append(fpc)
    cmeans_label_ls.append(np.argmax(u, axis=0))

plt.figure(figsize = (16,8))
plt.plot(C, cmeans_coefficient, 'bx-')
plt.xlabel('群數')
plt.ylabel('Partition Coefficient')
plt.title('透過 Elbow Method 找出最適群數')
plt.savefig("./Fuzzy-C-means 決定最佳分群數(diamonds).png")
plt.show()
kn = KneeLocator([i for i in range(1, 15)], cmeans_coefficient, curve = 'convex', direction =
'decreasing')
print("根據 elbow method 找出的最適群數為  ", kn.knee)

# 輸出上述分群結果
reg_data["C-Means_label"] = cmeans_label_ls[kn.knee-1]



##################
# Single Linkage #
##################

# 透過樹狀圖查看最佳分群數
dis = sch.linkage(reg_cluster, metric = "euclidean", method = "single")
sch.dendrogram(dis)
plt.title("Single Linkage")
plt.savefig("./Single_Linkage_dendrogram(diamonds).png")
plt.show()

# 由於無法從圖片判斷，故在此導入 silhouette_score
score_ls =[]
```

```
num_cls = []
for s in range(2,11):
    model = AgglomerativeClustering(n_clusters = s, linkage="single").fit(reg_cluster)
    num_cls.append(s)
    score_ls.append(metrics.silhouette_score(reg_cluster, model.labels_,metric='euclidean'))
silhouette_df = pd.DataFrame({"群數":num_cls, "silhouette_score":score_ls})

# 使用 Single Linkage 分群
single_linkage = AgglomerativeClustering(n_clusters = 2, linkage="single").fit(reg_cluster)
reg_data["Single_linkage_label"] = single_linkage.labels_


###################
# Average Linkage #
###################

# 透過樹狀圖查看最佳分群數
dis = sch.linkage(reg_cluster, metric = "euclidean", method = "average")
sch.dendrogram(dis)
plt.title("Average Linkage")
plt.savefig("./Average_Linkage_dendrogram(diamonds).png")
plt.show()

# 使用 Average Linkage 分群
average_linkage = AgglomerativeClustering(n_clusters = 5,
linkage="average").fit(reg_cluster)
reg_data["Average_linkage_label"] = average_linkage.labels_


####################
# Complete Linkage #
####################

# 透過樹狀圖查看最佳分群數
dis = sch.linkage(reg_cluster, metric = "euclidean", method = "complete")
sch.dendrogram(dis)
plt.title("Complete Linkage")
plt.savefig("./Complete_Linkage_dendrogram(diamonds).png")
```

```
plt.show()

#  使用 Complete Linkage 分群
complete_linkage = AgglomerativeClustering(n_clusters = 4,
linkage="complete").fit(reg_cluster)
reg_data["Complete_linkage_label"] = average_linkage.labels_

#  輸出結果
reg_data.to_csv("diamonds_trainingset_cluster.csv", encoding="utf-8-sig")



#%%
###################
# bankruptcy(cls) #
###################
cls_data = pd.read_csv("bankruptcy_trainingset.csv", encoding= "utf-8-sig")
cls_data = cls_data[[i for i in cls_data.columns if "Unnamed" not in i]]
cls_data.dtypes
cls_cluster = cls_data.copy()

###########
# K-Means #
###########

#  透過 Elbow method：分別查看每個資料集被分成 1~15 群的狀況，來決定最佳分群數
kmeans_distortions = []
kmeans_label_ls = []
K = range(1,15)
for k in K:
    kmeanModel = cluster.MiniBatchKMeans(n_clusters = k,random_state = 0,max_iter =
1000)
    kmeanModel.fit(cls_cluster)
    kmeans_distortions.append(kmeanModel.inertia_)
    kmeans_label_ls.append(kmeanModel.labels_)
plt.figure(figsize = (16,8))
plt.plot(K, kmeans_distortions, 'bx-')
plt.xlabel('群數')
plt.ylabel('Intracluster Distance')
```

```
plt.title('透過 Elbow Method 找出最適群數')
plt.savefig("./Kmeans 決定最佳分群數(bankruptcy).png")
plt.show()
kn = KneeLocator([i for i in range(1, 15)], kmeans_distortions, curve = 'convex', direction =
'decreasing')
print("根據 elbow method 找出的最適群數為 ", kn.knee)

# 輸出上述分群結果
cls_data["K-Means_label"] = kmeans_label_ls[kn.knee-1]


#################
# Fuzzy C-Means #
#################

# 透過 Elbow method：分別查看每個資料集被分成 1~15 群的狀況，來決定最佳分群數
cmeans_coefficient = []
cmeans_label_ls = []
C = range(1,15)
for c in C:
    cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(cls_cluster.to_numpy().transpose(),
                                                      c, 2, error = 0.005, maxiter =
1000, init = None, seed = 0)
    cmeans_coefficient.append(fpc)
    cmeans_label_ls.append(np.argmax(u, axis=0))

plt.figure(figsize = (16,8))
plt.plot(C, cmeans_coefficient, 'bx-')
plt.xlabel('群數')
plt.ylabel('Partition Coefficient')
plt.title('透過 Elbow Method 找出最適群數')
plt.savefig("./Fuzzy-C-means 決定最佳分群數(bankruptcy).png")
plt.show()
kn = KneeLocator([i for i in range(1, 15)], cmeans_coefficient, curve = 'convex', direction =
'decreasing')
print("根據 elbow method 找出的最適群數為 ", kn.knee)

# 輸出上述分群結果
```

```
cls_data["C-Means_label"] = cmeans_label_ls[kn.knee-1]


##################
# Single Linkage #
##################

# 透過樹狀圖查看最佳分群數
dis = sch.linkage(cls_cluster, metric = "euclidean", method = "single")
sch.dendrogram(dis)
plt.title("Single Linkage")
plt.savefig("./Single_Linkage_dendrogram(bankruptcy).png")
plt.show()

# 由於無法從圖片判斷，故在此導入 silhouette_score
score_ls =[]
num_cls = []
for s in range(2,11):
    model = AgglomerativeClustering(n_clusters = s, linkage="single").fit(cls_cluster)
    num_cls.append(s)
    score_ls.append(metrics.silhouette_score(cls_cluster, model.labels_,metric='euclidean'))
silhouette_df = pd.DataFrame({"群數":num_cls, "silhouette_score":score_ls})

# 使用 Single Linkage 分群
single_linkage = AgglomerativeClustering(n_clusters = 4, linkage="single").fit(cls_cluster)
cls_data["Single_linkage_label"] = single_linkage.labels_


###################
# Average Linkage #
###################

# 透過樹狀圖查看最佳分群數
dis = sch.linkage(cls_cluster, metric = "euclidean", method = "average")
sch.dendrogram(dis)
plt.title("Average Linkage")
plt.savefig("./Average_Linkage_dendrogram(bankruptcy).png")
plt.show()
```

```
# 由於無法從圖片判斷，故在此導入 silhouette_score
score_ls =[]
num_cls = []
for s in range(2,11):
    model = AgglomerativeClustering(n_clusters = s, linkage="average").fit(cls_cluster)
    num_cls.append(s)
    score_ls.append(metrics.silhouette_score(cls_cluster, model.labels_,metric='euclidean'))
silhouette_df = pd.DataFrame({"群數":num_cls, "silhouette_score":score_ls})

# 使用 Average Linkage 分群
average_linkage = AgglomerativeClustering(n_clusters = 2, linkage="average").fit(cls_cluster)
cls_data["Average_linkage_label"] = average_linkage.labels_


####################
# Complete Linkage #
####################

# 透過樹狀圖查看最佳分群數
dis = sch.linkage(cls_cluster, metric = "euclidean", method = "complete")
sch.dendrogram(dis)
plt.title("Complete Linkage")
plt.savefig("./Complete_Linkage_dendrogram(bankruptcy).png")
plt.show()

# 由於無法從圖片判斷，故在此導入 silhouette_score
score_ls =[]
num_cls = []
for s in range(2,11):
    model = AgglomerativeClustering(n_clusters = s, linkage="complete").fit(cls_cluster)
    num_cls.append(s)
    score_ls.append(metrics.silhouette_score(cls_cluster, model.labels_,metric='euclidean'))
silhouette_df = pd.DataFrame({"群數":num_cls, "silhouette_score":score_ls})

# 使用 Complete Linkage 分群
complete_linkage = AgglomerativeClustering(n_clusters = 7,
linkage="complete").fit(cls_cluster)
```

```
cls_data["Complete_linkage_label"] = complete_linkage.labels_

cls_data.to_csv("bankruptcy_trainingset_cluster.csv",encoding="utf-8-sig")
```