國立臺東專科學校
National Taitung Junior College

自動化油耗資料分群系統

指導教授:廖強棋
學生:楊庭睿、謝信宇、莊東霖、陳威宇

摘要

油耗資料分群分析對多方面皆有很大的應用，因此我們使用資料探勘其中的分群演算法”K-Means”和”K-Medoids”，並且還使用了手肘法來選擇分群數量當作參考，我們使用Anaconda中的Spyder來撰寫程式碼，以自動執行Python的K-Means和K-Medoids演算法，最後使用這些功能來讓我們獲得自動化油耗資料的分群分析結果。

壹、研究動機

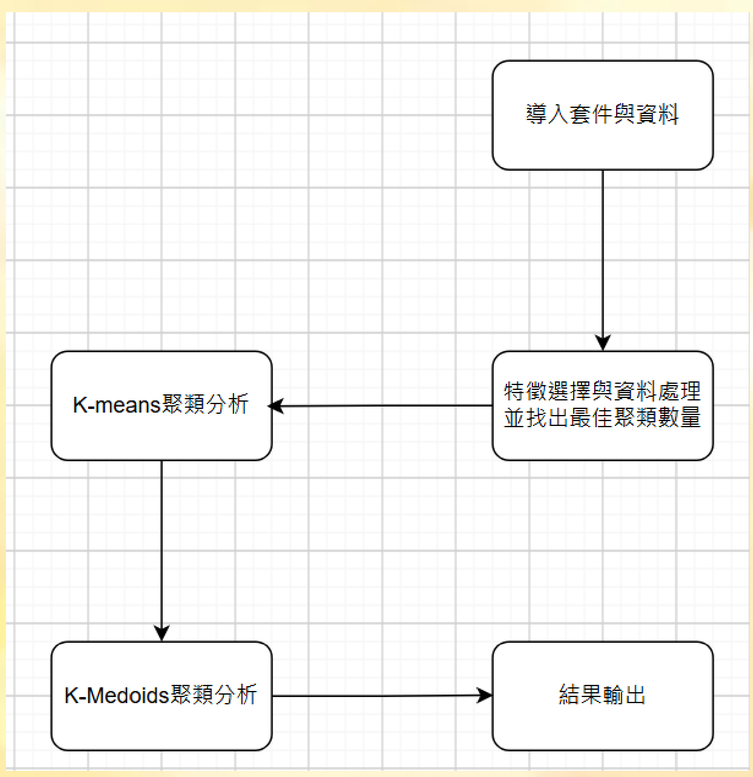
油耗資料分群分析對車輛維護與保養、駕駛人員的行為分析、環境影響分析……等，皆有很大的幫助，而Weka是一般用來進行資料探勘的軟體，我們發現使用者在Weka上設定分群時，可能會因為不熟悉各種操作而發生了一些小錯誤，導致分群的結果不會是所需要的甚至是產生錯誤，因此我們嘗試製作出能夠降低操作上產生的錯誤，同時也能讓使用者能更快且更容易取得所需要的正確分群結果，藉由自動化的方式來做內部的設置讓使用者可以省去在Weka上設定的時間，同時也可降低使用者在設定時出錯導致出乎預料的錯誤發生，讓使用者可以更專注於數據分析無需顧慮設定上是否有錯誤而導致分析結果不正確。

貳、作品優勢

以Python語法來實現自動化的使用K-Means與K-Medoids演算法，透過數據分析的結果可以得知龐大的資料中各個資料的關聯性，若設定中出現錯誤產出的結果也會接連的出現錯誤，導致無法獲得所需的結果，因此創建了自動化的功能也能避免使用者自行使用而產生的各種問題。

參、程式展示

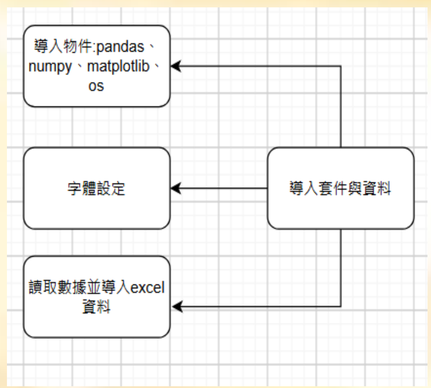
(圖四~九)我們使用手肘法來選擇最佳聚類數量，然後再使用K-Means與K-Medoids演算法來進行聚類分析，並透過Matplotlib來繪製最後輸出的圖表。
(圖十)python自動抓取excel中的資料所產生的變數。



圖一、程式運作流程圖

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn_extra.cluster import KMedoids
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_sc
from tabulate import tabulate
import os

# 設置環境變量以避免 KMeans 的內存泄漏警告
os.environ['OMP_NUM_THREADS'] = '1'

# 設定中文顯示和負號正常顯示
plt.rcParams['font.family'] = ['SimHei']  # 使用黑體,確保系統已安裝該字體
plt.rcParams['axes.unicode_minus'] = False  # 讓負號正常顯示

# 讀取數據
data = pd.read_excel(r'C:\Users\user\Downloads\車輛油耗資料(全)(1).xlsx')
```

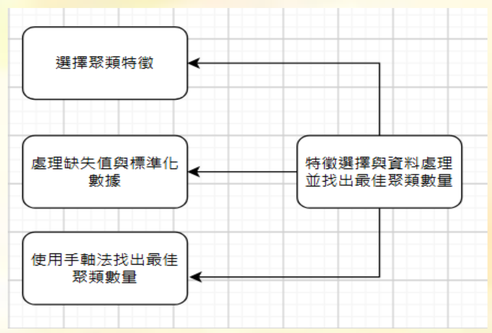
圖二、導入套件與資料



圖三、導入套件與資料示意圖

```python
# 選擇聚類的特徵
features = ['1 月油量', '1月里程數']
additional_features = ['噸位', '排氣量(C.C.)', '平均燃油效率(km/L)', '總里程數', '總用油量']
all_features = features + additional_features

# 處理缺失值
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(data[features])
additional_data = imputer.fit_transform(data[additional_features])

# 標準化數據
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

inertia = []
K = range(1, 10)
for k in K:
    kmeans = KMeans (n_clusters=k, random_state=5)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(8, 4))
plt.plot(K, inertia, 'bx-')
plt.xlabel('聚類數量',fontsize=12)
plt.ylabel('內部平方和(Inertia)', fontsize=12)
plt.title('手肘法選擇最佳聚類數量',fontsize=14)
plt.grid()
plt.show()
```

圖四、找出最佳聚類數量



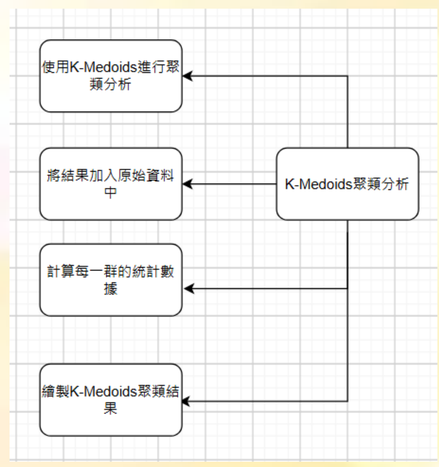
圖六、找出最佳聚類數量示意圖

```python
# 選擇 6 個聚類
optimal_clusters = 6

# K-Means 聚類
kmeans = KMeans(n_clusters=optimal_clusters, init='random', n_init=10, random_state=5)
kmeans.fit(X_scaled)
predictions = kmeans.predict(X_scaled)

# K-Means 聚類結果
kmeans_silhouette = silhouette_score(X_scaled, predictions)
kmeans_ch_score = calinski_harabasz_score(X_scaled, predictions)
kmeans_db_score = davies_bouldin_score(X_scaled, predictions)

print(f"K-Means 聚類評估指標:")
print(f"輪廓係數 (Silhouette Score): {kmeans_silhouette:.2f}")
print(f"Calinski-Harabasz 指數: {kmeans_ch_score:.2f}")
print(f"Davies-Bouldin 指數: {kmeans_db_score:.2f}\n")

# 將聚類結果加入原始資料中
data['KMeans_Cluster'] = predictions

# 計算每一群的統計數據
kmeans_summary = data.groupby('KMeans_Cluster').agg(
    平均重量=('噸位', 'mean'),
    平均排氣量=('排氣量(C.C.)', 'mean'),
    平均燃油效率=('平均燃油效率(km/L)', 'mean'),
    平均總行駛里程=('總里程數', 'mean'),
    總油量=('總用油量', 'sum')
).reset_index()

# 將統計數據四捨五入到小數點後三位
kmeans_summary = kmeans_summary.round(3)

print("K-Means 聚類各群統計數據:")
print(tabulate(kmeans_summary, headers='keys', tablefmt='pretty', showindex=False))

# 繪製 K-Means 聚類結果
plt.figure(figsize=(10, 6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=predictions, cmap='viridis', alpha=0.6)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
            s=125, c='red', marker='X', label='中心點')
plt.title('KMeans 聚類', fontsize=16)
plt.xlabel('1月油量 (標準化)', fontsize=14)
plt.ylabel('1月里程數 (標準化)', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend()
plt.grid()
plt.show()
```

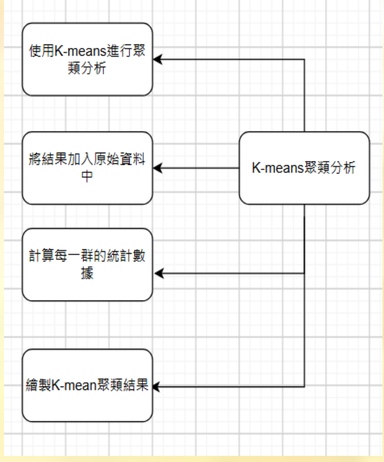
圖五、K-Means

```python
# K-Medoids 聚類
kmedoids = KMedoids(n_clusters=optimal_clusters, random_state=5)
kmedoids.fit(X_scaled)
medoid_predictions = kmedoids.predict(X_scaled)

# K-Medoids 聚類結果
kmedoids_silhouette = silhouette_score(X_scaled, medoid_predictions)
kmedoids_ch_score = calinski_harabasz_score(X_scaled, medoid_predictions)
kmedoids_db_score = davies_bouldin_score(X_scaled, medoid_predictions)

print(f"K-Medoids 聚類評估指標:")
print(f"輪廓係數 (Silhouette Score): {kmedoids_silhouette:.2f}")
print(f"Calinski-Harabasz 指數: {kmedoids_ch_score:.2f}")
print(f"Davies-Bouldin 指數: {kmedoids_db_score:.2f}\n")

# 將聚類結果加入原始資料中
data['KMedoids_Cluster'] = medoid_predictions

# 計算每一群的統計數據
kmedoids_summary = data.groupby('KMedoids_Cluster').agg(
    平均重量=('噸位', 'mean'),
    平均排氣量=('排氣量(C.C.)', 'mean'),
    平均燃油效率=('平均燃油效率(km/L)', 'mean'),
    平均總行駛里程=('總里程數', 'mean'),
    總油量=('總用油量', 'sum')
).reset_index()

# 將統計數據四捨五入到小數點後三位
kmedoids_summary = kmedoids_summary.round(3)

print("K-Medoids 聚類各群統計數據:")
print(tabulate(kmedoids_summary, headers='keys', tablefmt='pretty', showindex=False))

# 繪製 K-Medoids 聚類結果
plt.figure(figsize=(10, 6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=medoid_predictions, cmap='viridis', alpha=0.6)
plt.scatter(X_scaled[kmedoids.medoid_indices_, 0], X_scaled[kmedoids.medoid_indices_, 1],
            s=125, c='red', marker='X', label='中心點')
plt.title('KMedoids 聚類', fontsize=16)
plt.xlabel('1月油量 (標準化)', fontsize=14)
plt.ylabel('1月里程數 (標準化)', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend()
plt.grid()
plt.show()
```

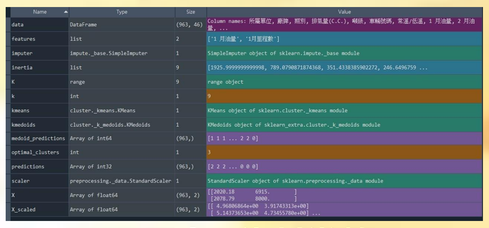
圖七、K-Medoids



圖八、K-Medoids示意圖



圖九、K-Means示意圖



Name	Type	Size	Value
data	DataFrame	(963, 46)	Column names: 所屬單位, 廠牌, 噸別, 排氣量(C.C.), 噸位, 車輛類別, 常溫/低溫, 1 月油量, 2 月油量, ...
features	list	2	['1 月油量', '1月里程數']
imputer	impute._base.SimpleImputer	1	SimpleImputer object of sklearn.impute._base module
inertia	list	9	[1925.9999999999998, 789.8790871874368, 351.4338385902272, 246.6496759 ...
K	range	9	range object
k	int	1	9
kmeans	cluster._kmeans.KMeans	1	KMeans object of sklearn.cluster._kmeans module
kmedoids	cluster._k_medoids.KMedoids	1	KMedoids object of sklearn_extra.cluster._k_medoids module
medoid_predictions	Array of int64	(963,)	[1 1 1 ... 2 2 0]
optimal_clusters	int	1	6
predictions	Array of int32	(963,)	[2 2 2 ... 0 0 0]
scaler	preprocessing._data.StandardScaler	1	StandardScaler object of sklearn.preprocessing._data module
X	Array of float64	(963, 2)	[[2020.18 6915.] [2479.79 5000.] [[4.56886864e+00 3.91743113e+00] [5.14373655e+00 4.73455780e+00] ...
X_scaled	Array of float64	(963, 2)	

圖十、資料變數

國立臺東專科學校
National Taitung Junior College

自動化油耗資料分群系統

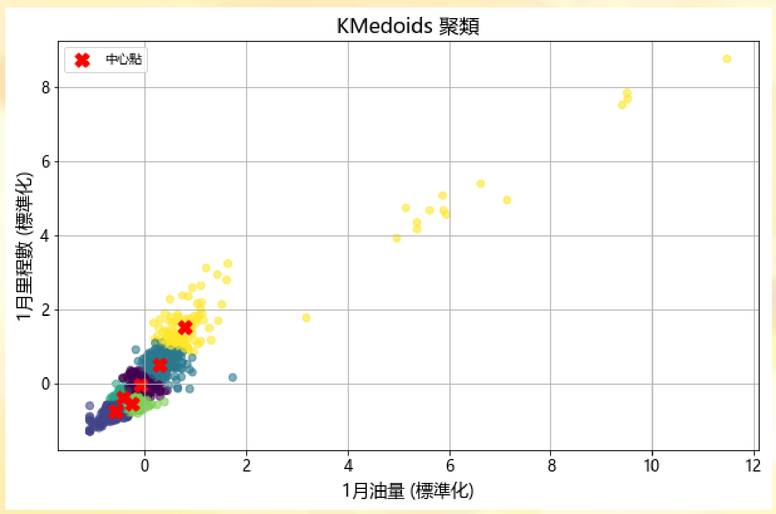
指導教授:廖強棋
學生:楊庭睿、謝信宇、莊東霖、陳威宇

肆、結果分析

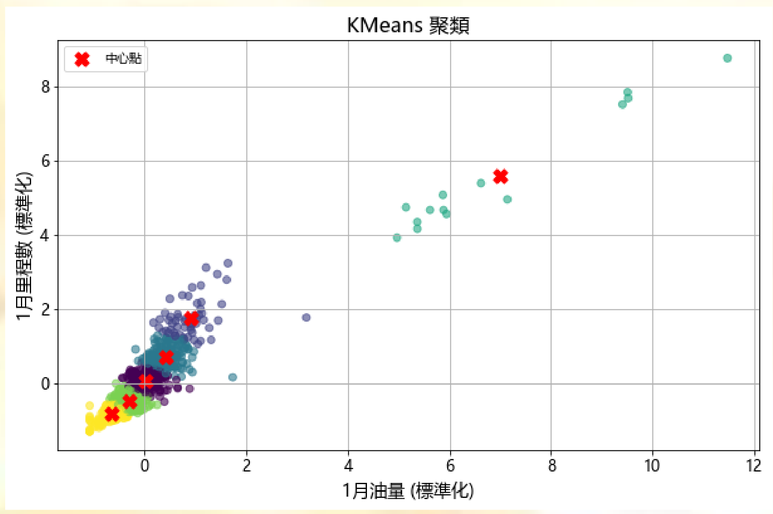
我們所使用的演算法是K-Means和K-Medoids他們演算法的執行內容如下:

K-Means是先將資料分成幾類並給每個群類一個中心點(虛擬點)，然後計算每筆資料與每個中心點之間的距離，與最近的中心點設為一個分群再用每筆資料的座標來給每一群計算出一個新的中心點，用來取代舊的中心點，重複計算，直到中心點停止變動。(圖十二)

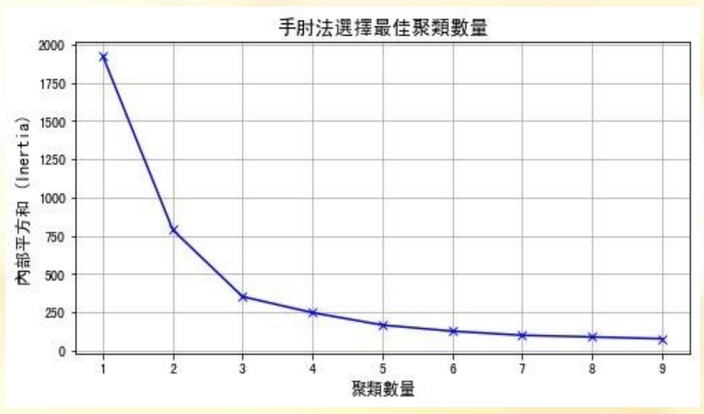
K-Medoids與K-Means做法大致相同，差別在於K-Medoids使用的中心點為實體樣本點，以及計算新的中心點時，是使用所有最小距離和的樣本點當作新中心點。(圖十一)



圖十一、K-Medoids 聚類



圖十二、K-Means 聚類



圖十三、手肘法

```
K-Means 聚類評估指標：
輪廓係數 (Silhouette Score): 0.40
Calinski-Harabasz 指數: 1852.78
Davies-Bouldin 指數: 0.71

K-Means 聚類各群組對應的中心點：
群組 0 對應的中心點是: [-0.304632   -0.46752344]
群組 1 對應的中心點是: [0.40577806 0.70584946]
群組 2 對應的中心點是: [0.01378236 0.05777746]
群組 3 對應的中心點是: [0.8963772  1.75513458]
群組 4 對應的中心點是: [-0.64966785 -0.8309875 ]
群組 5 對應的中心點是: [6.98870749 5.58444276]
K-Means 聚類各群統計數據：
```

KMeans_Cluster	平均重量(噸)	平均排氣量(c.c.)	平均燃油效率(km/L)	平均總行駛里程(公里)	總油量(公升)
0.0	3.562	2975.592	5.024	109768.264	5066352.652
1.0	3.945	3143.754	6.09	43631.447	414639.21
2.0	3.714	3010.408	5.427	29777.73	870134.216
3.0	17.214	7774.857	3.185	89054.429	393929.93
4.0	3.591	2944.836	4.507	14547.866	1089168.246
5.0	3.699	2875.737	4.65	11267.034	426726.47

圖十四、K-Means結果輸出

```
K-Medoids 聚類評估指標：
輪廓係數 (Silhouette Score): 0.28
Calinski-Harabasz 指數: 247.52
Davies-Bouldin 指數: 0.97

K-Medoids 聚類各群組對應的中心點：
群組 0 對應的中心點是: [-0.09725689 -0.0514581 ]
群組 1 對應的中心點是: [-0.58017124 -0.76390044]
群組 2 對應的中心點是: [0.2823428  0.48701576]
群組 3 對應的中心點是: [-0.43993066 -0.3737893 ]
群組 4 對應的中心點是: [-0.25389234 -0.5417329 ]
群組 5 對應的中心點是: [0.78572826 1.50522087]
K-Medoids 聚類各群統計數據：
```

KMedoids_Cluster	平均重量(噸)	平均排氣量(c.c.)	平均燃油效率(km/L)	平均總行駛里程(公里)	總油量(公升)
0.0	3.548	2968.631	4.955	119640.379	4921219.879
1.0	3.68	2881.031	4.621	11373.357	486510.83
2.0	3.726	3024.541	5.268	27249.68	944448.329
3.0	3.609	2925.131	5.108	15686.475	329295.39
4.0	3.587	2945.122	4.167	13880.575	600660.766
5.0	5.693	3747.303	5.566	47435.566	978815.53

圖十五、K-Medoids結果輸出

第零群 這群的總行駛里程以及總油耗是六群最高的，表明這一群可能是經常進行長途的物流配送的車輛，而重量及排氣量屬於中等範圍，應該是中型貨卡。
第一群 這群的行駛里程較低，可能是屬於短程運送，日常配送。
第二群 這群燃油效率偏高且行駛里程適中，可能屬於中長途的運輸車輛，排氣量也相對較高、總油耗量較低，說明需要更高的動力來應對長途與高負載的運輸。
第三群 總里程與燃油效率偏中等水平，可能是日常多用途的車輛，而總油耗較低說明可能使用頻率與負載不高。
第四群 燃油效率最低可是總油耗卻不低，可能屬於耗油量大的車型，適合運載較重貨物且適合行走於崎嶇地形(如山路)的車輛，行程里程也不算高 ，可能是用於特定用途，非日常廣泛使用，得知這群可能為運載重型貨物或者有著越野用途的車輛。
第五群 重量與排氣量明顯高於其他群，說明這群為大型運輸車輛，可是燃油效率卻是最高的，可能是經過優化設計的新型節能車輛，中高的里程跟油耗，適合於專業用途與大型項目。

伍、分群分析後的應用

1.車輛維護與保養

透過聚類分群的結果來大致查看車輛的使用強度（油耗與里程的關聯），從而引導車隊或是車主進行車輛的保養，進而提升車輛的壽命也能預防車輛故障等問題。

2. 駕駛人員的行為分析

油耗與里程數的數據能夠反映出駕駛人員行駛的習慣。聚類分析能夠將不同駕駛習慣的車主分類，例如:經常在行駛中瞬間加速的駕駛可能會出現在高油耗群體中。

這樣分類能夠幫助駕駛人員了解自己的駕駛習慣，而改善駕駛習慣，使得降低車輛的油耗，達到更好的節省油耗。

3. 環境影響分析

藉由各種不同車輛群體的油耗模式，來評估車輛對環境的影響，尤其是大型車隊。透過聚類分群識別出油耗較高的群體，車隊可以採取措施來降低二氧化碳的排放量。

陸、結論

透過Anaconda中的Spyder環境使用Python語法來執行K-Means演算法，我們成功地實現了自動化對大數據的聚類演算法分群分析，並減少使用者在設定方面出現各種錯誤的可能性，對於不熟悉使用Weka的使用者可以有著更簡單的動作，減少操作上的不便，未來可以再更進一步發展新增其他的演算法，以滿足各方面的需求。

致謝

**感謝財團法人車輛研究測試中心(ARTC)
提供實際資料供我們研究分析，
財團法人車輛研究測試中心(ARTC)官網:https://www.artc.org.tw
與QR-code(如右圖)。**