台北捷運人流分析

Taipei-MRT-Analysis

台北捷運

製作人：甯文駿

摘要

本作品旨在分析**台北捷運各站的進出站人流**，並進一步利用機器學習方法對捷運站點進行**群集分類**，以辨識出不同使用模式之捷運站類型。研究分為兩大部分：

首先，在**《MRT_Analysis_IN OUT.ipynb》**中，透過整理台北捷運進出站流量資料，計算各站**每日平均進站與出站人數**，並進行**視覺化分析**，觀察尖峰時段與使用強度的分布情形。此階段著重於呈現各捷運站人流的時間性差異，並藉由**熱力圖**提供初步的洞察。

其次，在**《MRT_Analysis_KMeans.ipynb》**中，應用 **K-means 分群演算法**對捷運站進行群集分析。藉由將站點依據其人流模式分類，可辨識出如通勤型、觀光型與住宅型等不同功能導向的**站點類型**。此方法有助於交通規劃與營運策略的制定，如針對特定類型站點進行資源配置或尖峰時段人流調度。

資料來源

臺北市資料大平臺-臺北捷運各站分時進出量統計

連結：https://data.taipei/dataset/detail?id=63f31c7e-7fc3-418b-bd82-b95158755b4d

使用資料：臺北捷運每日分時各站 OD 流量統計資料_202406

1.MRT_Analysis_IN OUT.ipynb

1.1 讀取資料，做缺失值檢查和其餘處理

先讀取資料(202406)可看到資料集的內容為 8096760 rows × 5 columns，檔案大小約為 310MB，且資料完整並無缺失值的存在，接著在對欄位名稱、資料型態做轉換和新增。

```
# 資料來源：臺北市資料大平臺-臺北捷運各站分時進出量統計
# https://data.taipei/dataset/detail?id=63f31c7e-7fc3-418b-bd82-b95158755b4d
# 使用資料：臺北捷運每日分時各站OD流量統計資料_202406
df = pd.read_csv('臺北捷運每日分時各站OD流量統計資料_202406.csv')
df #  8096760 rows × 5 columns
```

	日期	時段	進站	出站	人次
0	2024-06-01	0	松山機場	松山機場	0
1	2024-06-01	0	松山機場	中山國中	0
2	2024-06-01	0	松山機場	南京復興	0
3	2024-06-01	0	松山機場	忠孝復興	0
4	2024-06-01	0	松山機場	大安	0
...	...	...	...	...	...
8096755	2024-06-30	23	新北產業園區	徐匯中學	0
8096756	2024-06-30	23	新北產業園區	三和國中	0
8096757	2024-06-30	23	新北產業園區	三重國小	0
8096758	2024-06-30	23	新北產業園區	迴龍	2
8096759	2024-06-30	23	新北產業園區	丹鳳	0

8096760 rows × 5 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8096760 entries, 0 to 8096759
Data columns (total 5 columns):
 #   Column  Dtype 
---  ------  ----- 
 0   日期      object
 1   時段      int64 
 2   進站      object
 3   出站      object
 4   人次      int64 
dtypes: int64(2), object(3)
memory usage: 308.9+ MB
```

```python
# 欄位名稱轉換
df.columns = ['Date', 'Time', 'Entry', 'Exit', 'People']
df = df[df['People'] != 'Null']

# 資料型態轉換
df["People"] = pd.to_numeric(df["People"], downcast='integer')
df["Entry"] = df["Entry"].astype('category')
df["Exit"] = df["Exit"].astype('category')

# 新增欄位進出站
df['From_to'] = df['Entry'].str.cat(df['Exit'], sep='_')

# 欄位轉換和新增欄位
df['Date'] = pd.to_datetime(df['Date'] ,errors='coerce')
df['Weekday'] = df['Date'].dt.weekday # 0:星期一, 1:星期二, 2:星期三, 3:星期四, 4:星期五, 5:星期六, 6:星期日
df["Weekday"] = pd.to_numeric(df["Weekday"], downcast='integer')
df['Weeknum'] = df['Date'].dt.isocalendar().week
df["Weeknum"] = pd.to_numeric(df["Weeknum"], downcast='integer')
df.head()
```

	Date	Time	Entry	Exit	People	From_to	Weekday	Weeknum
0	2024-06-01	0	松山機場	松山機場	0	松山機場_松山機場	5	22
1	2024-06-01	0	松山機場	中山國中	0	松山機場_中山國中	5	22
2	2024-06-01	0	松山機場	南京復興	0	松山機場_南京復興	5	22
3	2024-06-01	0	松山機場	忠孝復興	0	松山機場_忠孝復興	5	22
4	2024-06-01	0	松山機場	大安	0	松山機場_大安	5	22

1.2 設計 def()函數來計算進出站分時人數和一週內的平均人數

df_to_pivot(df)

作用：

把原始的資料轉換為「樞紐分析表（pivot table）」，以方便後續分析。

```python
# 轉換為樞紐分析表 (Pivot Table)
def df_to_pivot(df):
    pv = df.pivot_table(index=['Date', 'From_to'], columns='Time', values='People')
    return pv
```

weekday_series(df, weekday, station, output, by_thousand)

作用：

計算特定捷運站在指定星期幾的每個時段平均進出人數。

```python
#  進出站分時人數
def weekday_series(df, weekday, station, output, by_thousand):
    # 進出站判斷
    if output == 'in':
        df_weekday = df[(df['Weekday'] == weekday) & (df['Entry'] == station)].copy()
    elif output == 'out':
        df_weekday = df[(df['Weekday'] == weekday) & (df['Exit'] == station)].copy()

    # 進出站人數
    pv_weekday = df_to_pivot(df_weekday).dropna()
    weekday_count = len(pv_weekday.index.get_level_values(0).unique())

    # 以千人為單位
    if by_thousand:
        result_weekday = round(pv_weekday.sum(numeric_only=True) / weekday_count / 1000, 1)
    else:
        result_weekday = pv_weekday.sum(numeric_only=True) / weekday_count

    return result_weekday
```

```python
    # 測試 weekday_series 函數
    try:
        weekday_series_output = weekday_series(df, 0, '台北車站', 'in', True)
        print('weekday_series_output:' , weekday_series_output)
    except Exception as e:
        print('Error:', e)
```

```
weekday_series_output: Time
0      1.3
1      0.0
5      0.0
6      1.3
7      5.3
8     11.2
9      7.9
10     6.4
11     6.5
12     6.8
13     7.3
14     7.0
15     7.3
16     8.1
17    11.0
18    13.0
19    10.0
20     8.0
21     8.0
22     7.7
23     2.8
dtype: float64
```

station_sum_up(df, station, output, by_thousand)

作用：

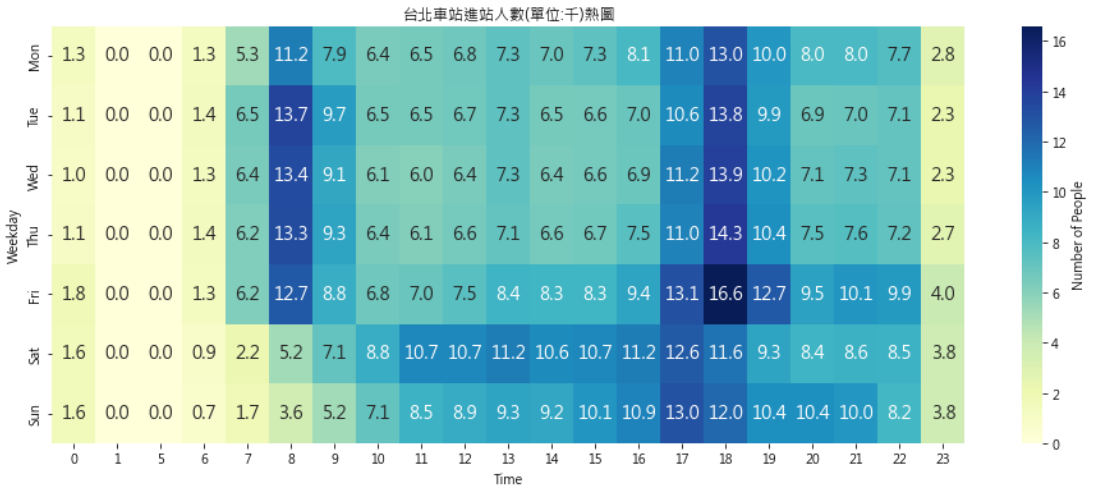
計算一個站點在一週內每天的每個時段平均人數（共 7 筆）。

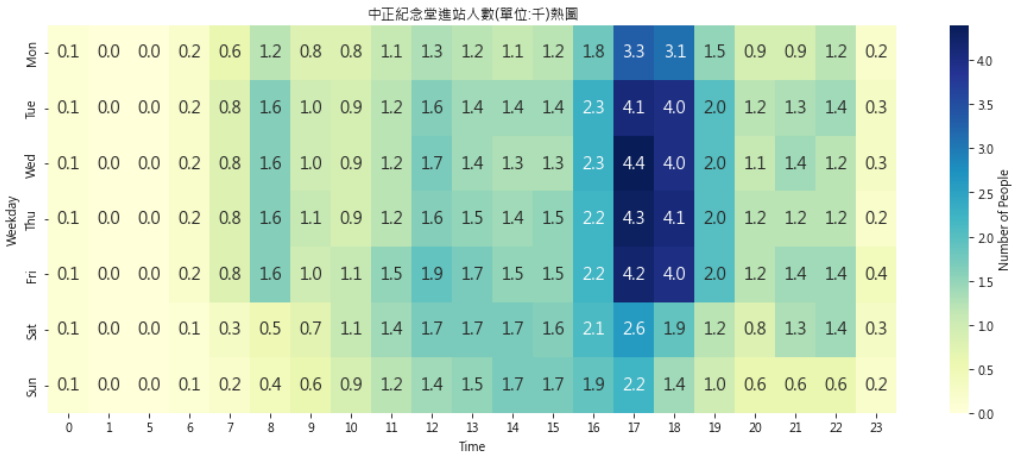
```python
#  一週內的平均人數
def station_sum_up(df, station, output, by_thousand):
    sys.stdout.write('\rprocessing {}, output format: {}'.format(station, output))
    weekday_list = []
    # 星期一到星期日
    for i in range(7):
        series = weekday_series(df, i, station, output, by_thousand)
        weekday_list.append(series)

    # 合併七天的資料
    result = pd.concat(weekday_list, axis=1).T
    result['Weekday'] = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
    result = result.set_index('Weekday')

    return result
```

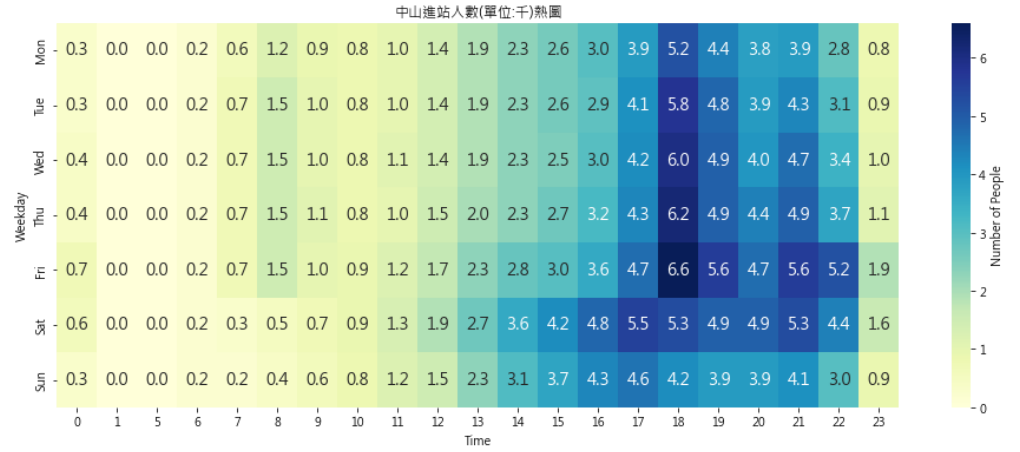
1.3 輸入車站名稱，輸出進/出站的人流熱力圖

進佔人流熱力圖

```python
# 車站名稱列表
stations = ['台北車站', '中正紀念堂', '中山']

for station in stations:
    # 使用 station_sum_up 函示計算車站的進站人數
    station_data = station_sum_up(df, station, 'in', True)

    # 繪製進站熱力圖
    plt.figure(figsize=(16, 6))
    sns.heatmap(station_data, annot=True, fmt='.1f', cmap='YlGnBu', cbar_kws={'label': 'Number of People'} ,annot_kws={'size': 14}
    plt.title(f'{station}進站人數(單位:千)熱圖')
    plt.ylabel('Weekday')
    plt.xlabel('Time')
    plt.show()
```

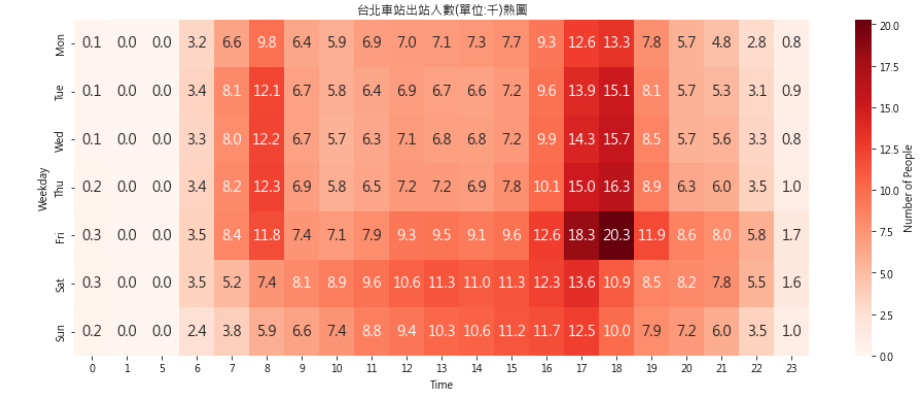


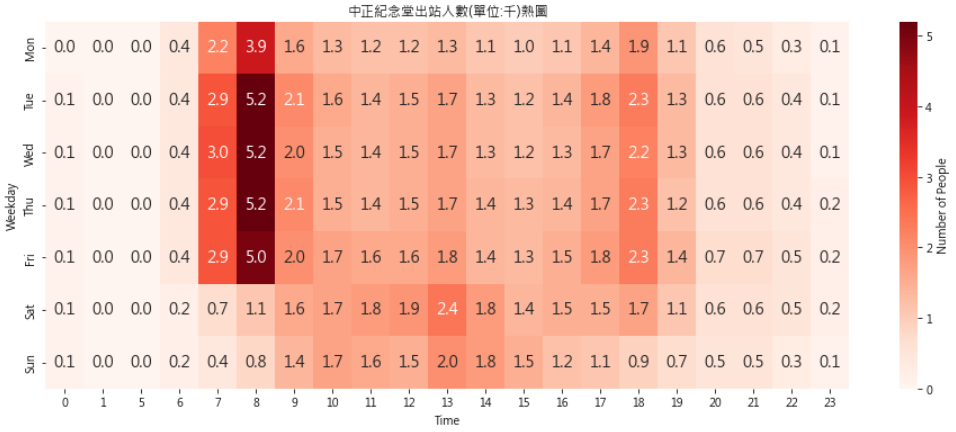


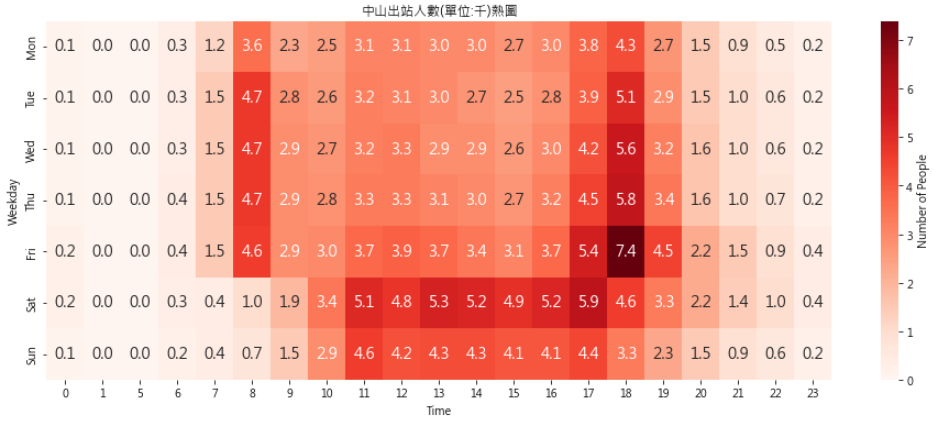


出站人流熱力圖

```python
stations = ['台北車站', '中正紀念堂', '中山']

for station in stations:
    # 使用 station_sum_up 函示計算車站的進站人數
    station_data = station_sum_up(df, station, 'out', True)

    # 繪製出站熱力圖
    plt.figure(figsize=(16, 6))
    sns.heatmap(station_data, annot=True, fmt='.1f', cmap='Reds', cbar_kws={'label': 'Number of People'} ,annot_kws={'size': 14})
    plt.title(f'{station}出站人數(單位:千)熱圖')
    plt.ylabel('Weekday')
    plt.xlabel('Time')
    plt.show()
```







1.4 以某站為例做簡單分析

以中正紀念堂站為例，分析人流高峰

- 平日
  - **進站人流：17:00 到 19:00** 達到最高峰，對應到**下班時段**。
  - **出站人流：7:00 到 9:00** 達到最高峰，對應到**上班時段**，反映出**工作日早晨的通勤需求**。

- 假日
  - 進站：10:00 到 12:00。
  - 出站：15:00 到 17:00。

2.MRT_Analysis_KMeans.ipynb

2.1 讀取資料

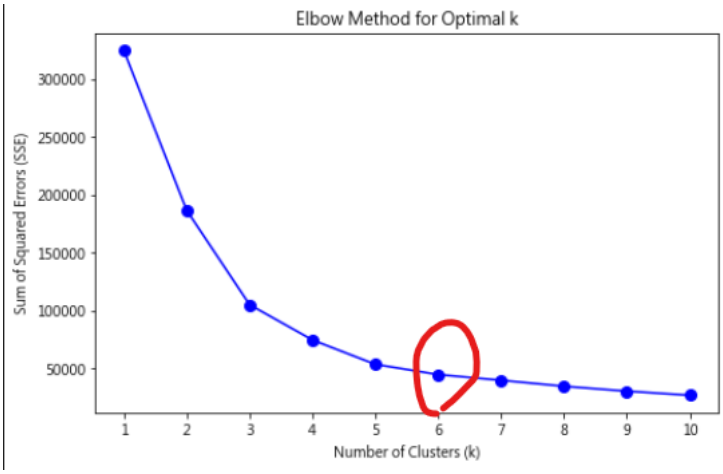
```
# 出站資料
df_cluster = pd.read_csv('TAIPEI_MRT_output.csv')
df_cluster
```

	station	0_0	0_1	0_5	0_6	0_7	0_8	0_9	0_10	0_11	...	6_14	6_15	6_16	6_17	6_18	6_19	6_20	6_21	6_22	6_23
0	松山機場	0.0	0.0	0.0	0.8	1.5	2.2	1.5	1.3	1.2	...	1.6	1.8	1.6	1.3	0.7	0.5	0.5	0.4	0.3	0.1
1	中山國中	0.2	0.0	0.0	0.5	2.6	7.6	4.1	1.8	1.6	...	2.2	2.4	2.6	3.5	2.9	2.3	2.1	2.3	1.7	0.6
2	南京復興	0.3	0.0	0.0	2.2	11.2	36.8	16.1	7.2	6.8	...	7.2	6.3	7.7	9.4	7.4	5.3	4.2	4.0	3.5	1.2
3	忠孝復興	0.3	0.0	0.0	1.7	6.0	12.5	9.8	10.2	10.5	...	15.0	14.4	14.5	15.2	11.9	8.4	5.7	4.6	2.8	1.2
4	大安	0.3	0.0	0.0	1.3	9.4	16.1	7.4	4.3	4.0	...	4.9	4.2	4.1	4.7	4.2	3.0	2.7	2.8	1.9	0.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
103	徐匯中學	0.4	0.0	0.0	0.3	1.0	0.9	0.8	0.8	1.0	...	2.2	2.6	3.5	4.0	3.9	3.6	3.7	3.3	3.1	1.6
104	三和國中	0.5	0.0	0.0	0.3	1.1	1.2	1.0	0.8	0.9	...	2.0	2.7	3.1	3.6	3.8	3.5	3.3	3.3	3.5	1.4
105	三重國小	0.5	0.0	0.0	0.3	0.8	1.0	0.9	1.0	1.2	...	2.4	2.5	3.0	3.8	3.9	3.3	3.3	3.8	3.2	1.5
106	迴龍	0.4	0.0	0.0	0.5	2.0	1.5	1.2	0.9	1.0	...	1.6	2.1	2.5	2.8	2.8	2.8	2.7	2.7	2.6	1.5
107	丹鳳	0.4	0.0	0.0	0.4	1.0	0.9	0.6	0.5	0.6	...	1.5	1.8	2.0	2.5	2.9	2.2	2.2	2.3	2.7	1.2

108 rows × 148 columns

2.2 使用 KMeans 分群演算法，加上肘部圖來決定要分幾群

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

sse = []
K = range(1, 11)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df_cluster.drop(columns=['station']))
    sse.append(kmeans.inertia_)

# 繪製肘部圖
plt.figure(figsize=(8, 5))
plt.plot(K, sse, 'bo-', markersize=8)
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Sum of Squared Errors (SSE)')
plt.title('Elbow Method for Optimal k')
plt.xticks(K)
plt.show()
```



2.3 進行分群並繪製該群站點的人流熱力圖

k_means_df(df, num_clusters)

作用：

使用 KMeans 演算法將捷運站分群。

```python
from sklearn.cluster import KMeans

def k_means_df(df, num_clusters):
    # 去除非數值欄位
    df_numeric = df.drop(columns=['station'])
    # KMeans 分群
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(df_numeric)
    # 新增分群欄位
    df['cluster'] = kmeans.labels_
    return df
```

k_means_list(km_df)

作用：

把每個站的時段人流資料整理為「每日一列、時段為欄」，並按照分群結果分類存入 dictionary。

```python
def k_means_list(km_df):
    group_dict = {}
    for station in km_df.index:
        # 取得站名與分群
        station_group = km_df.loc[station, 'cluster']
        # 移除非數值欄位並轉換為列表
        old_list = km_df.loc[station].drop(labels=['station', 'cluster']).tolist()
        # 每21小時一組重新整理
        new_list = []
        week_list = []
        for number in old_list:
            if len(week_list) < 21:
                week_list.append(number)
            else:
                new_list.append(week_list)
                week_list = [number]
        new_list.append(week_list)

        # 組成資料框
        result_df = pd.DataFrame(new_list, columns=[0,1,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23],
                                 index=['Mon','Tue','Wed','Thu','Fri','Sat','Sun'])
        station_dict = {km_df.loc[station, 'station']: result_df}
        group_dict.setdefault(station_group, []).append(station_dict)

    return group_dict
```

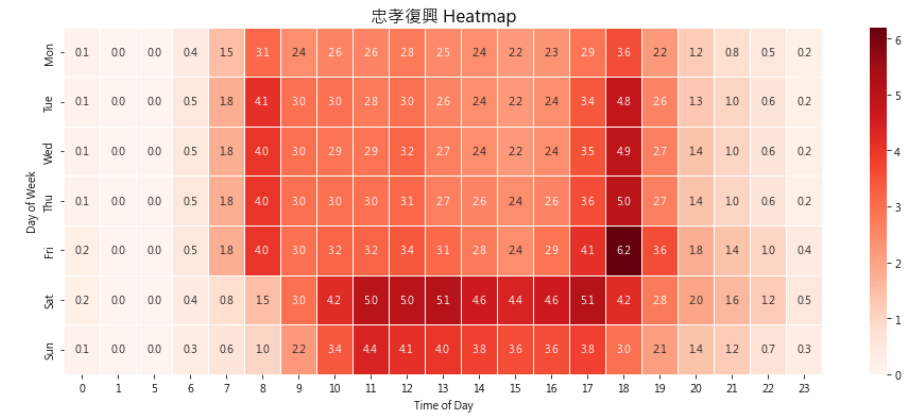
根據先前的肘部圖結果，決定分成六群來做後續分析。

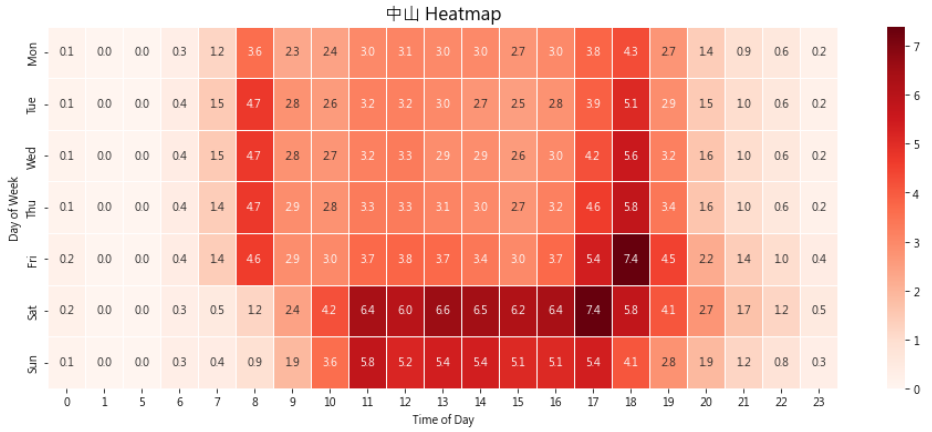
```python
# 使用函数進行分群並整理
k = 6
df_cluster_grouped = k_means_df(df_cluster, num_clusters=k)
grouped_data = k_means_list(df_cluster_grouped)
grouped_data[0]
```

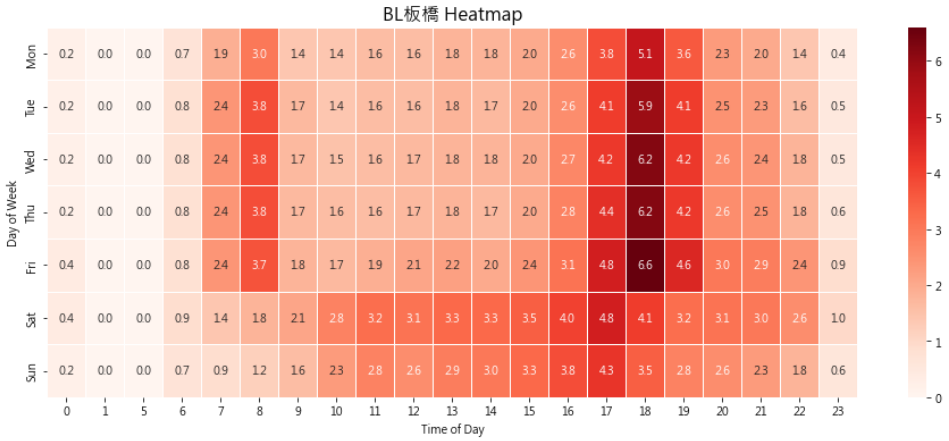
```python
cluster_counts = df_cluster_grouped.groupby('cluster')['station'].apply(list).reset_index()
cluster_counts['station_count'] = cluster_counts['station'].apply(len)

for index, row in cluster_counts.iterrows():
    print(f"There are {row['station_count']} stations belonging to cluster {row['cluster']}:")
    print(", ".join(row['station']))
    print("\n")
```

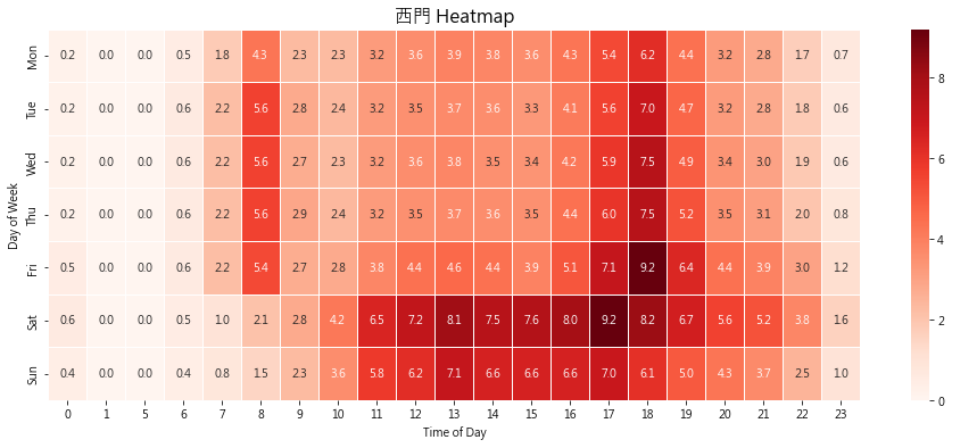
```
There are 21 stations belonging to cluster 0:
G大坪林, 公館, 頂溪, 永安市場, 0景安, 南勢角, 圓山, 劍潭, 士林, 芝山, 石牌, 淡水, 海山, 亞東醫院, 府中, 新埔, 江子翠, 龍山寺, 永春, 南港, 松山


There are 1 stations belonging to cluster 1:
台北車站


There are 19 stations belonging to cluster 2:
大安, 西湖, 港墘, 南港展覽館, 古亭, 中正紀念堂, 台大醫院, 雙連, 民權西路, 善導寺, 忠孝新生, 忠孝敦化, 國父紀念館, 台北101/世貿, 信義安和, 台北小


There are 60 stations belonging to cluster 3:
松山機場, 中山國中, 科技大樓, 六張犁, 麟光, 辛亥, 萬芳醫院, 萬芳社區, 木柵, 動物園, 大直, 劍南路, 文德, 內湖, 大湖公園, 葫洲, 東湖, 南港軟體園


There are 3 stations belonging to cluster 4:
南京復興, 市政府, 松江南京


There are 4 stations belonging to cluster 5:
忠孝復興, 中山, BL板橋, 西門
```

以 cluster 5 為例，繪製人流熱力圖。


忠孝復興 Heatmap
Day of Week
Mon 0.1 0.0 0.0 0.4 1.5 3.1 2.4 2.6 2.6 2.8 2.5 2.4 2.2 2.3 2.9 3.6 2.2 1.2 0.8 0.5 0.2
Tue 0.1 0.0 0.0 0.5 1.8 4.1 3.0 3.0 2.8 3.0 2.6 2.4 2.2 2.4 3.4 4.8 2.6 1.3 1.0 0.6 0.2
Wed 0.1 0.0 0.0 0.5 1.8 4.0 3.0 2.9 2.9 3.2 2.7 2.4 2.2 2.4 3.5 4.9 2.7 1.4 1.0 0.6 0.2
Thu 0.1 0.0 0.0 0.5 1.8 4.0 3.0 3.0 3.0 3.1 2.7 2.6 2.4 2.6 3.6 5.0 2.7 1.4 1.0 0.6 0.2
Fri 0.2 0.0 0.0 0.5 1.8 4.0 3.0 3.2 3.2 3.4 3.1 2.8 2.4 2.9 4.1 6.2 3.6 1.8 1.4 1.0 0.4
Sat 0.2 0.0 0.0 0.4 0.8 1.5 3.0 4.2 5.0 5.0 5.1 4.6 4.4 4.6 5.1 4.2 2.8 2.0 1.6 1.2 0.5
Sun 0.1 0.0 0.0 0.3 0.6 1.0 2.2 3.4 4.4 4.1 4.0 3.8 3.6 3.6 3.8 3.0 2.1 1.4 1.2 0.7 0.3
0 1 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Time of Day
0 1 2 3 4 5 6


中山 Heatmap
Day of Week
Mon 0.1 0.0 0.0 0.3 1.2 3.6 2.3 2.4 3.0 3.1 3.0 3.0 2.7 3.0 3.8 4.3 2.7 1.4 0.9 0.6 0.2
Tue 0.1 0.0 0.0 0.4 1.5 4.7 2.8 2.6 3.2 3.2 3.0 2.7 2.5 2.8 3.9 5.1 2.9 1.5 1.0 0.6 0.2
Wed 0.1 0.0 0.0 0.4 1.5 4.7 2.8 2.7 3.2 3.3 2.9 2.9 2.6 3.0 4.2 5.6 3.2 1.6 1.0 0.6 0.2
Thu 0.1 0.0 0.0 0.4 1.4 4.7 2.9 2.8 3.3 3.3 3.1 3.0 2.7 3.2 4.6 5.8 3.4 1.6 1.0 0.6 0.2
Fri 0.2 0.0 0.0 0.4 1.4 4.6 2.9 3.0 3.7 3.8 3.7 3.4 3.0 3.7 5.4 7.4 4.5 2.2 1.4 1.0 0.4
Sat 0.2 0.0 0.0 0.3 0.5 1.2 2.4 4.2 6.4 6.0 6.6 6.5 6.2 6.4 7.4 5.8 4.1 2.7 1.7 1.2 0.5
Sun 0.1 0.0 0.0 0.3 0.4 0.9 1.9 3.6 5.8 5.2 5.4 5.4 5.1 5.1 5.4 4.1 2.8 1.9 1.2 0.8 0.3
0 1 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Time of Day
0 1 2 3 4 5 6 7


BL板橋 Heatmap
Day of Week
Mon 0.2 0.0 0.0 0.7 1.9 3.0 1.4 1.4 1.6 1.6 1.8 1.8 2.0 2.6 3.8 5.1 3.6 2.3 2.0 1.4 0.4
Tue 0.2 0.0 0.0 0.8 2.4 3.8 1.7 1.4 1.6 1.6 1.8 1.7 2.0 2.6 4.1 5.9 4.1 2.5 2.3 1.6 0.5
Wed 0.2 0.0 0.0 0.8 2.4 3.8 1.7 1.5 1.6 1.7 1.8 1.8 2.0 2.7 4.2 6.2 4.2 2.6 2.4 1.8 0.5
Thu 0.2 0.0 0.0 0.8 2.4 3.8 1.7 1.6 1.6 1.7 1.8 1.7 2.0 2.8 4.4 6.2 4.2 2.6 2.5 1.8 0.6
Fri 0.4 0.0 0.0 0.8 2.4 3.7 1.8 1.7 1.9 2.1 2.2 2.0 2.4 3.1 4.8 6.6 4.6 3.0 2.9 2.4 0.9
Sat 0.4 0.0 0.0 0.9 1.4 1.8 2.1 2.8 3.2 3.1 3.3 3.3 3.5 4.0 4.8 4.1 3.2 3.1 3.0 2.6 1.0
Sun 0.2 0.0 0.0 0.7 0.9 1.2 1.6 2.3 2.8 2.6 2.9 3.0 3.3 3.8 4.3 3.5 2.8 2.6 2.3 1.8 0.6
0 1 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Time of Day
0 1 2 3 4 5 6



從**西門站的出站人流熱力圖**可以看到，**平日**的出站人流**高峰落在晚間 6 點左右**，而**假日**更為明顯，從**早上 11 點開始，人潮便絡繹不絕直至晚間 7-8 點左右**。

2.4 填補空缺的 2-4 時段欄位

由於台北捷運的人流資料中，並無 2-4 時段的資料(從熱力圖即可看出)，因此我將新增欄位做填補，以利後續作圖。

```
#  填補空缺的 2-4 時段
group_ids = range(7)
for group_id in group_ids:
    for suffix in range(2, 5):  # 從 _2 到 _4
        column_name = f"{group_id}_1"
        new_column_name = f"{group_id}_{suffix}"
        insert_position = df_cluster_grouped.columns.get_loc(column_name) + suffix - 1
        df_cluster_grouped.insert(insert_position, new_column_name, 0)
```

```
df_cluster_grouped_re = df_cluster_grouped
df_cluster_grouped_re.set_index('station', inplace=True)
df_cluster_grouped_re
```

station	0_0	0_1	0_2	0_3	0_4	0_5	0_6	0_7	0_8	0_9	...	6_15	6_16	6_17	6_18	6_19	6_20	6_21	6_22	6_23	cluster
松山機場	0.0	0.0	0	0	0	0.0	0.8	1.5	2.2	1.5	...	1.8	1.6	1.3	0.7	0.5	0.5	0.4	0.3	0.1	3
中山國中	0.2	0.0	0	0	0	0.0	0.5	2.6	7.6	4.1	...	2.4	2.6	3.5	2.9	2.3	2.1	2.3	1.7	0.6	3
南京復興	0.3	0.0	0	0	0	0.0	2.2	11.2	36.8	16.1	...	6.3	7.7	9.4	7.4	5.3	4.2	4.0	3.5	1.2	4
忠孝復興	0.3	0.0	0	0	0	0.0	1.7	6.0	12.5	9.8	...	14.4	14.5	15.2	11.9	8.4	5.7	4.6	2.8	1.2	5
大安	0.3	0.0	0	0	0	0.0	1.3	9.4	16.1	7.4	...	4.2	4.1	4.7	4.2	3.0	2.7	2.8	1.9	0.8	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
徐匯中學	0.4	0.0	0	0	0	0.0	0.3	1.0	0.9	0.8	...	2.6	3.5	4.0	3.9	3.6	3.7	3.3	3.1	1.6	3
三和國中	0.5	0.0	0	0	0	0.0	0.3	1.1	1.2	1.0	...	2.7	3.1	3.6	3.8	3.5	3.3	3.3	3.5	1.4	3
三重國小	0.5	0.0	0	0	0	0.0	0.3	0.8	1.0	0.9	...	2.5	3.0	3.8	3.9	3.3	3.3	3.8	3.2	1.5	3
迴龍	0.4	0.0	0	0	0	0.0	0.5	2.0	1.5	1.2	...	2.1	2.5	2.8	2.8	2.8	2.7	2.7	2.6	1.5	3
丹鳳	0.4	0.0	0	0	0	0.0	0.4	1.0	0.9	0.6	...	1.8	2.0	2.5	2.9	2.2	2.2	2.3	2.7	1.2	3

108 rows × 169 columns

2.5 設計 def()函數來繪製 cluster 的時間折線圖

line_graph_xlabel()

作用：

回傳 0~167 的列表，用作折線圖的 x 軸（每小時為一個點，代表整週）。

```
def line_graph_xlabel():
    return list(range(168))  #  x 軸標籤列表（0~167），對應一周 168(24x7) 小時
```

cluster_mean_dict(df)

作用：

對每個 cluster 內的所有站點資料取平均，回傳 {cluster: 平均值 Series} 字典，用於後面畫「平均曲線」。

```
# 計算每個cluster內數據列的平均值
def cluster_mean_dict(df):
    result_dict = {}
    for cluster in df['cluster'].unique():
        result_dict[cluster] = df[df['cluster'] == cluster].iloc[:, :-1].mean()
    return result_dict
```

cluster_group_dict_list_format(df)

作用：

依照 cluster 將各站的資料分組，每個值是一個 list，內含 {站名: 對應數值 Series}，方便後面逐站畫圖。

```
# Key: cluster, Value: DataFrame
def cluster_group_dict_list_format(df):
    group_dict = {}
    for station in df.index:
        station_group = df.loc[station, 'cluster']
        station_dict = {station: df.loc[station][:-1]}  # 只保留數據列
        group_dict.setdefault(station_group, []).append(station_dict)
    return group_dict
```

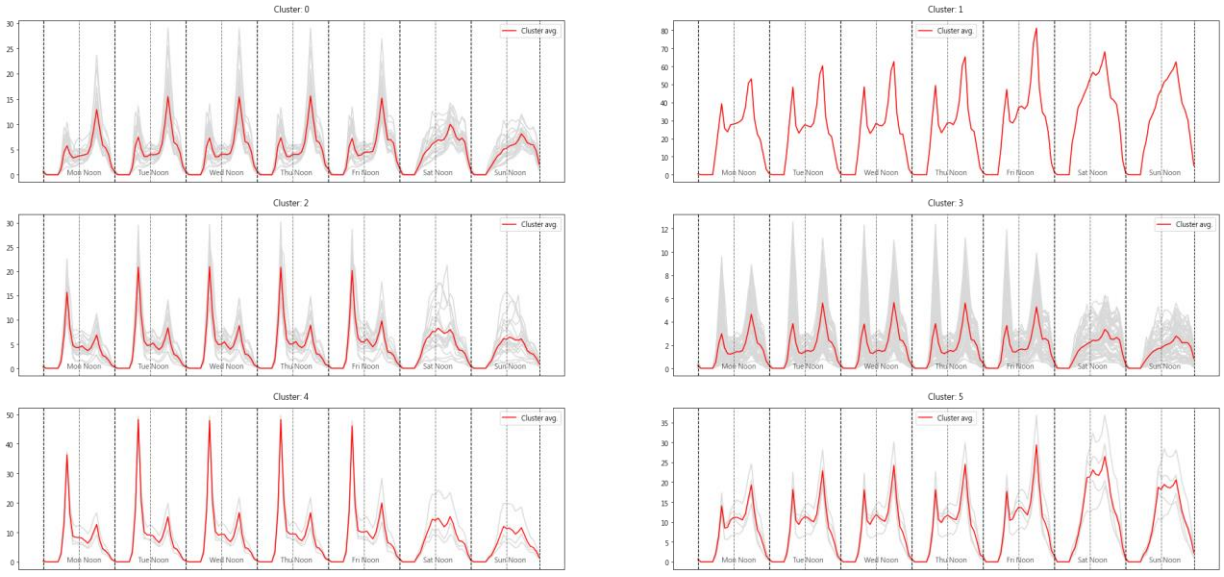
line_graph(group_dict, cluster_mean_dict, mean_color)

作用：視覺化函式，用來畫出每個 cluster 的折線圖。

```python
#  繪製cluster的折線圖
def line_graph(group_dict, cluster_mean_dict, mean_color):
    x_labels = line_graph_xlabel()

    # 計算圖表行數
    heigh = (len(group_dict) + 1) // 2

    fig, ax = plt.subplots(heigh, 2, figsize=(16 * 2, 5 * heigh))
    fig.subplots_adjust(hspace=0.2)

    for cluster in group_dict:
        # 設置子圖位置
        location_c = cluster % 2
        location_r = cluster // 2

        ax[location_r, location_c].set_title(f'Cluster: {cluster}', y=1.03)

        # 繪製每個車站的數據
        for station_dict in group_dict[cluster]:
            for station, values in station_dict.items():
                # 確保 values 的長度與 x_labels 一致
                if len(values) == len(x_labels):
                    ax[location_r, location_c].plot(x_labels, values, color='#d9d9d9', label='_nolegend_')
                else:
                    # 如果長度不一致，則處理方式可以是打印警告，或是填充至168
                    print(f"Skipping {station} due to length mismatch: {len(values)} vs {len(x_labels)}")

        # 繪製聚類平均值
        if cluster in cluster_mean_dict:
            ax[location_r, location_c].plot(x_labels, cluster_mean_dict[cluster], color=mean_color, label='Cluster avg.')

        # 添加垂直線
        weekday = ['Mon Noon', 'Tue Noon', 'Wed Noon', 'Thu Noon', 'Fri Noon', 'Sat Noon', 'Sun Noon']
        for i in range(0, 168, 24):
            ax[location_r, location_c].axvline(x=i, color='k', linestyle='--', linewidth=1.2)
            ax[location_r, location_c].text(i + 8, 0, weekday[i // 24], color=(0.3, 0.3, 0.3))

        # 最右側的垂直線
        ax[location_r, location_c].axvline(x=167, color='k', linestyle='--', linewidth=1)

        # 為每隔 12 小時添加淺色垂直線
        for i in range(12, 168, 24):
            ax[location_r, location_c].axvline(x=i, color=(0.45, 0.45, 0.45), linestyle='--', linewidth=1)

        # 關閉 x 軸標籤
        ax[location_r, location_c].tick_params(axis='x', which='both', bottom=False, top=False, labelbottom=False)
        ax[location_r, location_c].legend()
    plt.show()
```

2.6 分析各群的結果並加以解釋

```python
cluster_mean = cluster_mean_dict(df_cluster_grouped_re)
line_graph(line_group_dict, cluster_mean, 'r')
```



- 紅色線是每個 Cluster 內，站點的人流平均變化，概括了整個 Cluster 內車站的平均流量模式，代表該類型車站的整體特徵。
- 灰色是該 Cluster 全部站點，展示了該 Cluster 內每個車站每小時的流量變化趨勢。
- 如果灰色線條非常集中，且緊跟紅色線條，則說明分群的劃分效果較好，該 Cluster 內的車站數據模式相似。

Cluster 0

- 車站：G 大坪林, 公館, 頂溪, 永安市場, O 景安, 南勢角, 圓山, 劍潭, 士林, 芝山, 石牌, 淡水, 海山, 亞東醫院, 府中, 新埔, 江子翠, 龍山寺, 永春, 南港, 松山。
- 特徵：數值週期性波動顯著，尤其是在週一到週五的白天 (Noon) 時段有尖峰，週末趨於平緩。
- 解釋：可能代表****工作日交通流量明顯增加的站點****，週末人流量較低且波動較小。

Cluster 1

- 車站：台北車站。
- 特徵：數值波動幅度大且尖峰明顯分散在一周****各時間段****，週五和週六晚間顯示更高的數值。
- 解釋：可能代表商業或娛樂活動密集 的站點，例如週末晚間活動增加。

Cluster 2

- 車站：大安, 西湖, 港墘, 南港展覽館, 古亭, 中正紀念堂, 台大醫院, 雙連, 民權西路, 善導寺, 忠孝新生, 忠孝敦化, 國父紀念館, 台北 101/世貿, 信義安和, 台北小巨蛋, 南京三民, 行天宮, 東門。
- 特徵：週期性波動明顯，數值波峰出現在週一到週五的白天 (Noon)，週末數值趨於平緩且波峰較低。
- 解釋：與 Cluster 0 類似，可能代表****工作日通勤為主的站點****，但整體數值更低。

Cluster 3

- 車站：松山機場, 中山國中, 科技大樓, 六張犁, 麟光, 辛亥, 萬芳醫院,

萬芳社區，木柵，動物園，大直，劍南路，文德，內湖，大湖公園，葫洲，東湖，南港軟體園區，小碧潭，新店，新店區公所，七張，景美，萬隆，台電大樓，小南門，明德，嗎哩岸，奇岩，北投，新北投，復興崗，忠義，關渡，竹圍，紅樹林，頂埔，永寧，土城，後山埤，昆陽，象山，大安森林公園，北門，輔大，新莊，0頭前庄，先嗇宮，三重，菜寮，台北橋，大橋頭站，中山國小，蘆洲，三民高中，徐匯中學，三和國中，三重國小，迴龍，丹鳳。

- 特徵：數值波動頻繁但幅度較小，整體平緩，週末數值有所增加。
- 解釋：可能代表活動較少且穩定的站點，例如**住宅區或偏遠站點**。

Cluster 4

- 車站：南京復興，市政府，松江南京。
- 特徵：數值波動幅度較大，**週一到週五的白天（Noon）峰值非常高**，週末數值明顯下降。
- 解釋：可能代表高峰期特別明顯的站點，如**商務區或辦公中心**，**平日通勤流量大**。

Cluster 5

- 車站：忠孝復興，中山，BL板橋，西門。
- 特徵：數值波動呈現明顯的**雙峰結構**，週一到週五白天和晚上都有顯著峰值，週末也存在活動增加。
- 解釋：可能代表多元活動站點，如結合商務和娛樂功能的地區，**活動時間更廣泛**。

Cluster 總結

- Cluster 0 和 Cluster 2：主要代表**通勤需求高**的站點，波動集中在工作日白天。
- Cluster 1 和 Cluster 5：**週末活動顯著**，代表娛樂或商業活動集中的站點。
- Cluster 3：**整體活動較低**，代表穩定或低需求的站點。
- Cluster 4：高峰期流量顯著，通常是商務或**高流量的工作站點**。

3. GitHub

連結：https://github.com/c110156247/Taipei-MRT-Analysis