머신러닝 기반 서울시 내 도심형 물류센터 최적입지 선정

A Study on the Optimal Location Selection for Micro-Fullfilment-center on Seoul using Machine Learning

강대경 (201800302), 김 찬 (201801158), 신 용 (201802033), 김나연 (201904193)

ABSTRACT

The role of the distribution center has changed to a competitive element of a company that can create new added value in the connection between consumers' distributors and consumers.

The research is meaningful in that it extracts variables that can affect the location selection of urban logistics centers at a time when the current speed of delivery is highlighted as a company's competitiveness and presents guidelines for the location selection of urban logistics centers through data mining techniques. This analysis studied whether gas stations in Seoul will enter urban logistics centers in the future. The co-prosperity of gas stations, consumers, and distributors is an expected effect. Previous studies have shown that the factors of falling sales at gas stations are an increase in demand for electric vehicles and an increase in crude oil prices, while retailers have found that fast delivery is slowing down to competitiveness.

First, clustering was conducted by selecting variables that could affect the demand for logistics centers, such as official land prices, delivery efficiency, and productive population by region in Seoul, and four clusters were interpreted from the perspective of consumers. The interpreted cluster was agreed to determine the priority of the area where the urban logistics center was needed. The areas that most need the distribution center were Dongdaemun, Yangcheon, Dongjak, Songpa, Guro, Seodaemun, and Yeongdeungpo. Second, individual publicly announced land prices, area, distance from logistics hubs, population, number of schools, and size of commercial districts were selected as variables and the location score of the dependent variable was calculated as a measurement theory that calculates the importance of variables using the Analytical Hierarchy Process. Location suitability was categorized as very suitable, suitable, unsuitable, and very unsuitable by segmenting location scores, and analysis through classification techniques and multi-classification neural networks showed that gas stations most suitable for urban logistics centers could be classified.

Keywords: Machine Learning, Micro fullfilment center, Gas station,

Optimal Location Selection, Last-One mile

CONTENTS

1. Introduction

The MFC(Micro-Fulfillment-Center) is a small urban distribution center. While distribution centers are generally located outside the city. But MFCs are in the city. The role of the distribution center has changed from the connection between distributors and consumers to a competitive element of companies that can create new added value. To add more, the time between ordering and delivery can be greatly reduced. if the product is stored in the MFC in advance by predicting consumer preferences. It can provide faster delivery services to consumers. And MFC is related to Last-One Mile. The research is meaningful in that it extracts variables that can affect the location selection of MFC at a time when the current speed of delivery is highlighted as a company's competitiveness and presents guidelines for the location selection of MFC through data mining techniques. Recently, gas stations which have been reduced in demand due to electric vehicles, have been converted into micro fulfillment centers. This analysis studied whether gas stations in Seoul will enter MFC in the future.

2. Background of Study

In this study, the concept of urban distribution centers will be limited to located in metropolitan city so that retailers can supply distribution services quickly and in a timely. Therefore, the Micro-Fulfillment-Center can transport various services to necessary places in a timely, creating new demand for retailer and gas stations.

The study to determine the optimal location was mainly focused on optimization techniques based on mathematical models. Such research aims to minimize the distance to demand areas based on specific facilities.

Although there are quite a few models for determining location, there are typically Set Covering Model, Maximum Covering Location Model, P-median, and P-center. These models estimate charging demand as Point Demand.

First, as a model that assumes point demand, the Set Covering Model is a model that determines the minimum number and location of facilities required to meet all demand within a predefined distance, and many studies have used it to determine the number and location of hydrogen and other alternative fuel vehicles.

The Maximum Covering Location Model is a model that determines the location of facilities that meet the maximum demand when the number of facilities is being given.

P-median is a model aimed at minimizing the distance between facilities and demand points, and was used in a study to determine the location of charging stations that minimize the distance between hydrogen charging facilities and demand points.

P-center is a model that aims to reduce the maximum distance between facilities and demand points and is often used in optimal location studies along with P-median.

After reviewing the previous study, we found that the optimal selection using machine learning models has many limitations.

"A Study on the Location of Small Urban Distribution Centers in Seoul(김은재, 유건호, 이보라, 황세원 (2021))" used the MCLP algorithm to determine the optimal location of urban logistics centers.

"Optical Location Selection for Hydrogen Refueling Stations on a Highway using Machine Learning(조재혁, 김성수 (2021))" only confirmed the applicability of machine learning models.
Therefore, a study that analyzes the optimal location of urban distribution centers using machine learning, It is a meaningful study.

Author(Year)	조재혁, 김성수 (2021)	김은재, 유건호, 이보라, 황세원 (2021)	김수환 류준형(2020)	Team3 – Data mining [강대경 (201800302), 김 찬 (201801158), 신 용 (201802033), 김나연 (201904193)](2022)
Goal	Optimal Location Selection for Hydrogen Refueling Stations on a Highway using Machine Learning	A Study on the Location of Small Urban Distribution Centers in Seoul	A Machine Learning based Methodology for Selecting Optimal Location of Hydrogen Refueling Stations	A Study on the Optimal Location Selection for Micro-Fulfilment-center on Seoul using Machine Learning
Solution Approach	Binary Classification	MCLP	Clustering(k-medoid)	Clustering & Classification & ANN
Use Geographic Variable?	X	O	O	O
Given the number of hubs	O	X	X	O
Hub Capacity	X	X	O	O
Consider_Fulfillment_center_fixed cost	X	X	O	O
			Team3 – Data mining	

3. Research Design

3.1. Data collection and preprocessing process

We have collected data for making two types of dataset. One is for Clustering and the other one is for Classification. In Clustering, Our first objective was Finding Districts where need Microfulfillment centers.

In [14]:
```
df_logistic
```
Out[14]:

	District	Total_logis
0	강남구	2439367
1	서초구	2165093
2	서대문구	2575393
3	강서구	2289581

In [9]:
```
# subtract population 20~40
pop2040 = df_pop[df_pop.항목 =='계'][['동별', '항목', '20~24세', '25~29세'
pop2040
```
Out[9]:
```
동별
종로구      69930
중구       61688
용산구      112398
성동구      136671
```

In [24]:
```
df_sales_store
```
Out[24]:

	시군구명	분기당_매출_금액	점포수
0	강남구	1.186065e+13	186288
1	강동구	2.749058e+12	78406
2	강북구	1.482913e+12	48726
3	강서구	2.842142e+12	76835

In [37]:
```
# 구별 공시지가(단위: 천)
estate = df_estate[df_estate['시도명']=='서울특별시
estate
```
Out[37]:
```
시군구명
강남구       17385000
강동구        5986000
강북구        3705000
```

To choose district We extracted features(TOTAL_LOGIS, POP2040, TOTAL_SALES, NUM_STORE, ESTATE, etc.) by district from each files.

	운행년월 (DRIVEN_YM)	주소(구) (ADDR)	운행거리(평균) (DRIVEN_AVR)	운행거리(총거리) (DRIVEN_SUM)	운행대수 (DRIVEN_CNT)	운행_총시간 (DRIVEN_TIME)	경유지_총건수(VIA_CNT)
0	202011	동대문	10.706256	52579.968458	7122	109180824	402530
1	202006	용산구	12.758485	85058.221296	3671	66316795	375971
2	202007	구로구	12.307220	76255.048379	5280	79544164	299260

In [32]:
```
# 경유지 개수 / 운행대수로 변경 -> n대 차량이 k개 경유지 돌면 k/n 값이 클수록 운행효율에 정비례
df_subcjdata['driven_eff'] = (df_subcjdata.DRIVEN_SUM / df_subcjdata.DRIVEN_TIME) * (df_subcjdata.VIA_CNT / df_subcjdata.DRIVEN_CNT)
```
In [33]:
```
df_subcjdata.drop(['DRIVEN_AVR', 'DRIVEN_SUM', 'DRIVEN_CNT', 'DRIVEN_TIME', 'VIA_CNT'], axis=1, inplace=True)
```
In [34]:
```
df_subcjdata
```
Out[34]:

	ADDR	driven_eff
0	동대문구	0.027219
1	용산구	0.131360
2	구로구	0.054334
3	노원구	0.026526

And then create feature 'DRIVEN_EFF' by calculating (DRIVEN_SUM/DRIVEN_TIME) * (VIA_CNT/DRIVEN_CNT). We thought that operation efficiency is proportional to the speed of operation and the number of transit points and inversely proportional to the number of operations. Therefore, We assumed that Areas with high operational efficiency have high delivery efficiency. Below is Dataset for Clustering.

In [45]:
```python
# Final dataset for using first step.
df_table
```

Out[45]:

	DISTRICT	DRIVEN_EFF	TOTAL_LOGIS	POP2040	TOTAL_SALES	NUM_STORE	ESTATE
0	동대문구	0.027219	2400374	166805	4.447564e+12	66655	5198000
1	용산구	0.131360	2250955	112398	6.805869e+12	62840	10079000
2	구로구	0.054334	2442377	184583	2.806535e+12	69651	4109000
3	노원구	0.026526	2507172	215172	1.834499e+12	57391	3911000
4	서초구	0.159699	2165093	185381	6.456707e+12	117682	11141000

Dataset for Clustering		
Variables	Name	Description
Independent Variable	DRIVEN_EFF	Operational efficiency
	TOTAL_LOGIS	Delivery volume by region
	POP2040	Number of Producible Populations
	TOTAL_SALES	Off-line commercial sales
	NUM_STORE	Size of offline commercial area
	ESTATE	Officially assessed reference land price

In Classification, Our second goal is find optimal location of gas station for MFC in district and make a model classified whether it is optimal location well. Similar to the previous data collection, we extracted features(GAS_STATION, AREA, ESTATE, etc..) by gas station. A feature 'ESTATE' used here is a different concept from the 'ESTATE' used in Clustering. In Clustering, the concept of 'ESTATE' is Officially assessed reference land price. However, 'ESTATE' used in classification means Officially assessed individual land price.

```python
from haversine import haversine
from tqdm import notebook

distance_compare_list = []
for i in notebook.tqdm(range(df_DC.shape[0])):
    for j in range(df_gas.shape[0]):
        DC = (df_DC.Latitude[i], df_DC.Longitude[i])
        oil = (df_gas.LATITUDE[j], df_gas.LONGITUDE[j])
        result = haversine(DC, oil, unit='km')
        distance_compare_list.append(result)
```

GAS_STATION	DIST_EASTDC	DIST_WESTDC	DIST_KOREADC
현대오일뱅크(주) 직영소월길주유소	8.970251	13.202286	11.678968
선익상사(주) 동자동주유소	9.568763	12.593357	11.494239
현대오일뱅크㈜ 직영갈월동주유소	9.713120	12.454393	11.199948

And a feature 'DIST_####DC' is Distance from #### terminal(East, West, Korea). To calculate the distance between of them, we use haversine package.

```
locations = []
for addr in addrs:
    url = 'https://dapi.kakao.com/v2/local/search/address.json?query={}'.format(addr)
    headers = {"Authorization": "KakaoAK [redacted]"}
    place = requests.get(url, headers = headers).json()['documents']
    locations.append(place)
```

To gather the feature, we used Kakao API, and sort out information related to location address such as Latitude, Longitude, District, and Dong. And then merge all of them by gas station in seoul.

Using the QGIS program, the number of populations, the number of schools, the number of apartments, and the number of markets with a radius of 1000M based on gas stations were calculated. The following is the code for calculating the number of populations based on gas stations.

```
from sklearn.neighbors import KNeighborsClassifier

oilbank = pd.read_excel('서울시주유소_위경도.xlsx')
oilbank.rename(columns = {'LATITUDE': '위도', 'LONGITUDE':'경도'}, inplace = True)
x_train = df[['위도', '경도']]
y_train = df['wkt_geom']

neigh = KNeighborsClassifier(n_neighbors = 1)
neigh.fit(x_train, y_train)

x_test = oilbank.loc[:, ['위도', '경도']]
labels = neigh.predict(x_test)
oilbank['wkt_geom'] = labels
import haversine

df_LEFT_JOIN['points'] = df_LEFT_JOIN[['위도', '경도']].values.tolist()
oilbank['points'] = oilbank[['위도', '경도']].values.tolist()
m = 1000
for i in range(len(oilbank)):
    print(i)
    oil_point = oilbank.loc[i, 'points']
    df_LEFT_JOIN['m'] = df_LEFT_JOIN['points'].apply(lambda x: haversine.haversine(oil_point, x, unit = 'm'))
    population = df_LEFT_JOIN[df_LEFT_JOIN['m'] <= m+1]['인구'].sum()
    oilbank.loc[i, 'population'] = population
result = oilbank.loc[:, ['GAS_STATION', '위도', '경도', 'DISTRICT', 'DONG', 'wkt_geom', 'population']]
result.to_excel('주유소 100m 반경 안 인구수 최종 파일.xlsx',index=False)
result
```

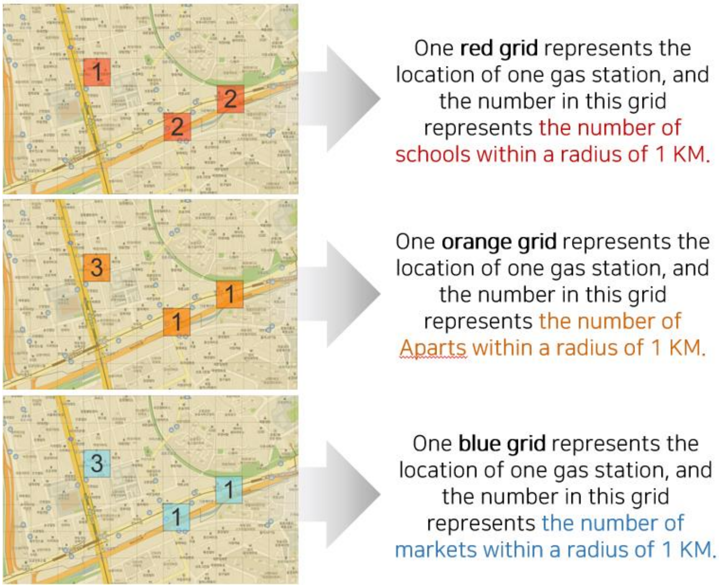
If you change the number to another number instead of the number 1000 in the part where it says "m=1000", the code was created so that the number of populations within the radius of the number entered based on the gas station could be obtained. In this project, the number of schools, apartments, and markets were all conducted based on a radius of 1000m, that is, 1km, so the number of populations was also set at 1000m. This creates an Excel file in which the number of populations with a radius of 1000m per gas station is obtained, and the Excel file is retrieved from the QGIS program and combined with the location of the gas station indicated by the grid. Therefore, it can be visualized as follows.

The following is part of the code for calculating the number of schools, populations, and apartments based on gas stations.

```python
import haversine
m = 1000
oilbank_df['point'] = oilbank[['LATITUDE', 'LONGITUDE']].values.tolist()
lst = ['school', 'apart', 'market']
for stat in lst:
    globals()[f'{stat}_df']['point'] = globals()[f'{stat}'][['Latitude', 'Longitude']].values.tolist()
    for i in range(len(oilbank)):
        oil_point = oilbank_df.loc[i, 'point']

        globals()[f'{stat}_df']['stat_per_m'] = globals()[f'{stat}_df']['point'].apply(lambda x: haversine.haversine(x, oil_point, unit = 'm'))
        oilbank_df.loc[i, stat] = len(globals()[f'{stat}_df'][globals()[f'{stat}_df']['stat_per_m'] <= m+1])
```

Through this, the number of schools, populations, and apartments with a radius of 1000M can be obtained based on the latitude and longitude of the gas station. This code creates columns for schools, populations, and apartments in one file, so if you create each file and change it to a SHP file, you can open it in one LAYER in the QGIS program. And if you change the label properties of each of these LAYERs, you can get each number based on a radius of 1000M in one grid containing a gas station.



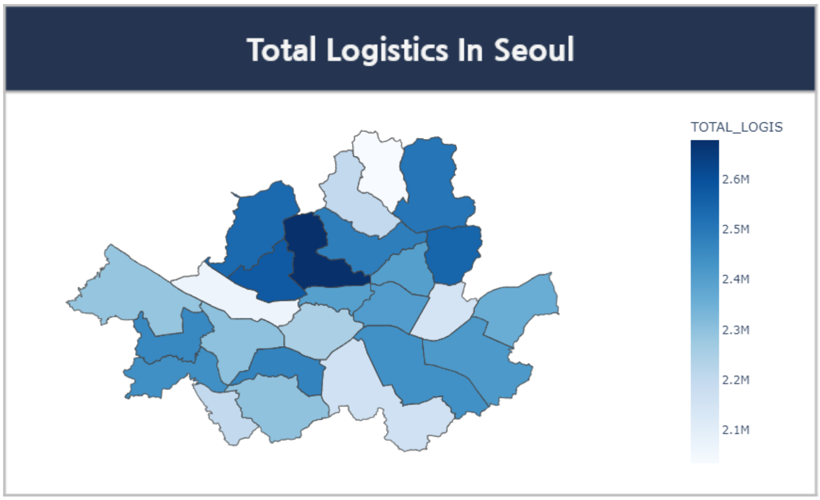
We were able to get the gas station area SHP file. The area of the gas station could be obtained by opening this SHP file in the QGIS program and executing a command to find the area in the 'Property Table>Field Calculator'. However, although the area of each gas station was obtained, it was difficult to combine the area into the previous pretreatment file because the name of the gas station in AREA and the name of the gas station that previously performed pretreatment were different. Therefore, based on the location of the gas station, it was possible to unify the name of the gas station and make all the data into one file by using the 'Vector>Geographic Information Processing Tool>Dissolve' in the QGIS program.

Below table is our final dataset for classification.

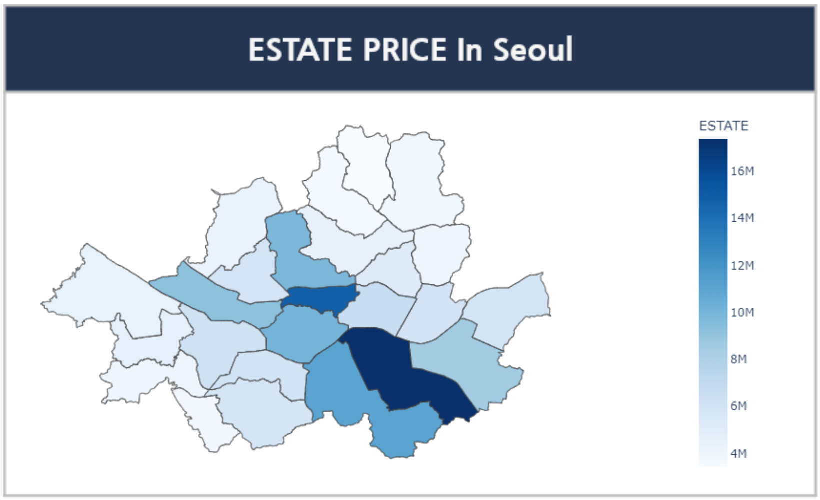
	GAS_STATION	DISTRICT	DONG	LATITUDE	LONGITUDE	AREA	ESTATE	DIST_EASTDC	DIST_WESTDC	DIST_KOREADC	POP1000	SCHOOL	APART	MARKET
0	현대오일뱅크(주) 직영소월길주유소	용산구	후암동	37.554409	126.977735	245.0	8280000.0	8.970251	13.202286	11.678968	15660	5	1	58
1	선익상사(주) 동자동주유소	용산구	동자동	37.550201	126.972418	711.0	18850000.0	9.568763	12.593357	11.494239	25467	9	2	6
2	현대오일뱅크㈜ 직영갈월동주유소	용산구	갈월동	37.547029	126.972228	700.0	15050000.0	9.713120	12.454393	11.199948	26924	9	1	22

Dataset for Classification		
Variables	Name	Description
Independent Variable	GAS_STATION	Naming of Gas Station
	AREA	Area of Gas Station
	ESTATE	Officially assessed individual land price;
	DIST_EASTDC	Distance from EastDC;
	DIST_WESTDC	Distance from WestDC;
	DIST_KOREADC	Distance from KoreaDC;
	POP1000	Population within 1km radius of Gas station
	SCHOOL	The number of school within 1km radius of Gas station
	APART	The number of apartment within 1km radius of Gas station
	MARKET	The number of market within 1km radius of Gas station

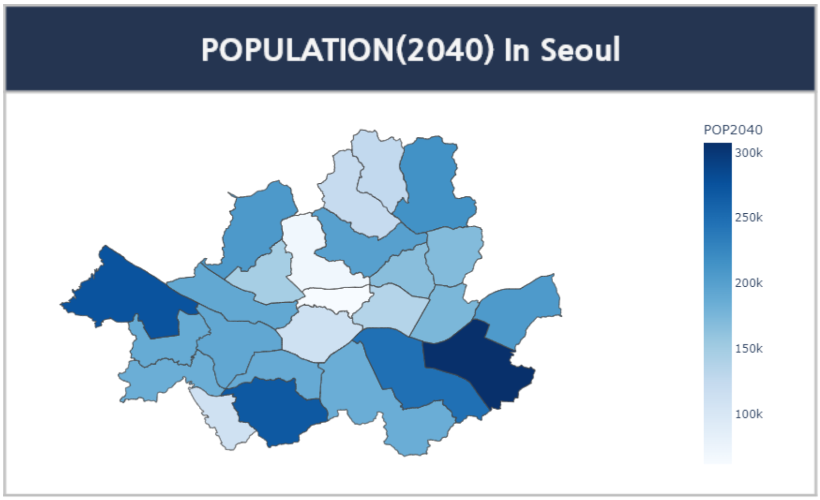
3.2 EDA



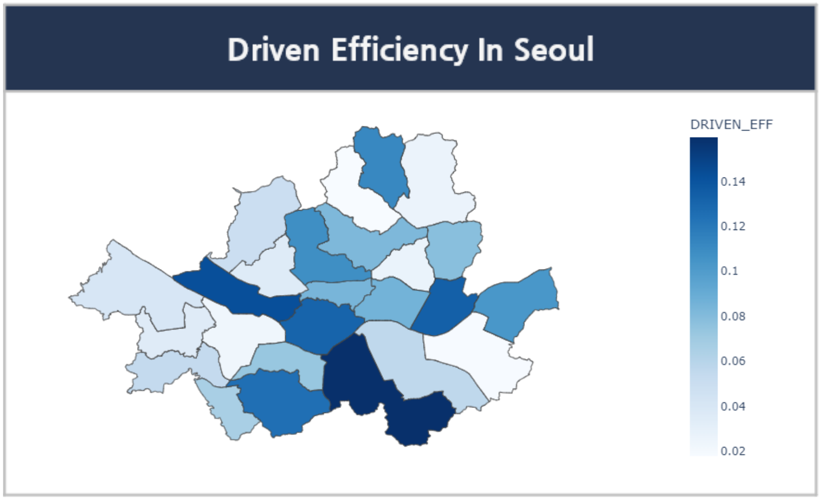
The darker the color, the higher the logistics volume. The dark part is Jongno-Gu and the bright part is Dobong-Gu. Therefore, there are a lot of logistics in Jongno-Gu, but Dobong-Gu does not.



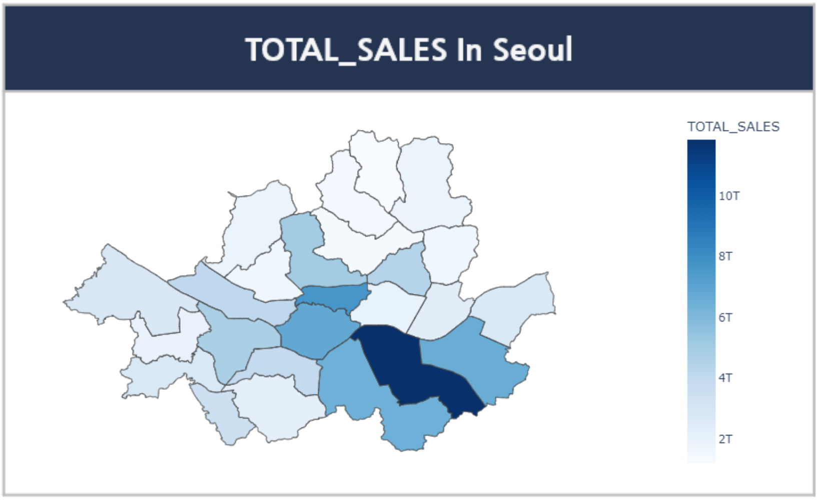
The darker the color, the higher the Land price. The dark part is Gangnam-Gu and the bright part is Dobong-Gu. Therefore, land price is especially high in Gangnam-Gu, and Dobong-Gu is lowest. and there is little difference of estate price among district-Dobong, Gangbuk, Geumcheon, Nowon, Jungnang, Guro-where is lower estate price.



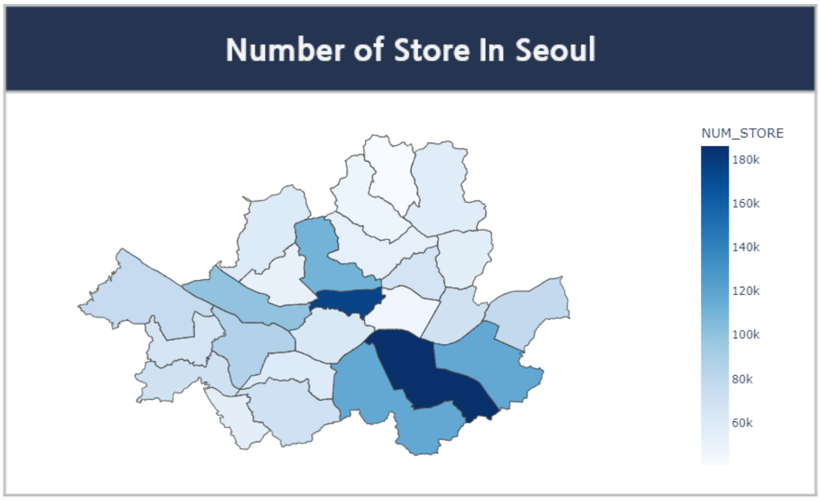
The darker the color, the higher the Population. The dark part is Songpa-Gu and the bright part are Jung-Gu and Jongno-Gu. Therefore, People live in Songpa-gu the most. and Jung-Gu & Jongno-Gu are lower population.



The darker the color, the higher the driven operation efficiency. The dark part is Seocho-Gu and the bright part are Gangbuk-Gu and Songpa-Gu. Therefore, Operation Efficiency is lowest in Gangbuk-Gu and Songpa-Gu, and highest in Seocho-Gu.



The darker the color, the higher the total sales. The dark part is Gangnam-Gu and the bright part is Dobong-Gu. Therefore, Total sales is especially high in G angnam-Gu, and Dobong-Gu is lowest.



The darker the color, the higher the number of stores. The dark part is Gangnam-Gu and the bright part is Dobong-Gu. Therefore, Total sales is especially high in Gangnam-Gu, and Dobong-Gu is lowest.

4. Data Analysis

4.1 Clustering

To choose optimal location where MFC is needed, we used four clustering algorithms-K-means, K-medoid, AGNES, GMM. We did standard scaling and doing PCA before clustering. Below is after doing StandardScaler.

In [6]:
```
# StandardScalaer
from sklearn.preprocessing import StandardScaler
s_scaler = StandardScaler()
s_scaled = s_scaler.fit_transform(df_scaling)
pd.DataFrame(s_scaled, columns=col_name)
```

Out[6]:

	DRIVEN_EFF	TOTAL_LOGIS	POP2040	TOTAL_SALES	NUM_STORE	ESTATE
0	-1.136150	0.227904	-0.197550	0.290781	-0.345691	-0.461560
1	1.346835	-0.703058	-1.124559	1.234272	-0.450148	0.939167
2	-0.489647	0.489605	0.105360	-0.365747	-0.263659	-0.774076
3	-1.152681	0.893314	0.626548	-0.754631	-0.599345	-0.830897

And then, we needed the number of Principal Component.

```
In [8]: print('Explained variance ratio :', pca.explained_variance_ratio_)
        pca_ratio = pd.DataFrame({'Explained variance':pca.explained_variance_,
                    'Explained variance ratio':pca.explained_variance_ratio_},
                   index=np.array([f"pca{num+1}" for num in range(s_scaled.shape[1])]))
        pca_ratio['Cumulative ratio'] = pca_ratio['Explained variance ratio'].cumsum()
        pca_ratio

        Explained variance ratio : [0.4736976  0.25002054 0.16547079 0.08416489 0.02014256 0.00650362]
Out[8]:
```

	Explained variance	Explained variance ratio	Cumulative ratio
pca1	2.960610	0.473698	0.473698
pca2	1.562628	0.250021	0.723718
pca3	1.034192	0.165471	0.889189
pca4	0.526031	0.084165	0.973354
pca5	0.125891	0.020143	0.993496
pca6	0.040648	0.006504	1.000000

According to upper image, we selected up to PCA3 because that point means the number of PC that Explained variance is more than 0.7 and cumulative ratio is over 80%. So, we concluded up to PC3 is suitable.

```
In [9]: pca = PCA(n_components=3)
        values_pca = pca.fit_transform(s_scaled)
        principalDf = pd.DataFrame(data=values_pca, columns = ['PC1', 'PC2', 'PC3'])
        print('Explained variance ratio :', pca.explained_variance_ratio_)

        Explained variance ratio : [0.4736976  0.25002054 0.16547079]

In [10]: principalDf
Out[10]:
```

	PC1	PC2	PC3
0	-0.484454	-0.823491	0.285771
1	1.235587	1.702674	0.624046
2	-0.893321	-0.611083	0.128440
3	-1.456038	-1.414561	-0.074750

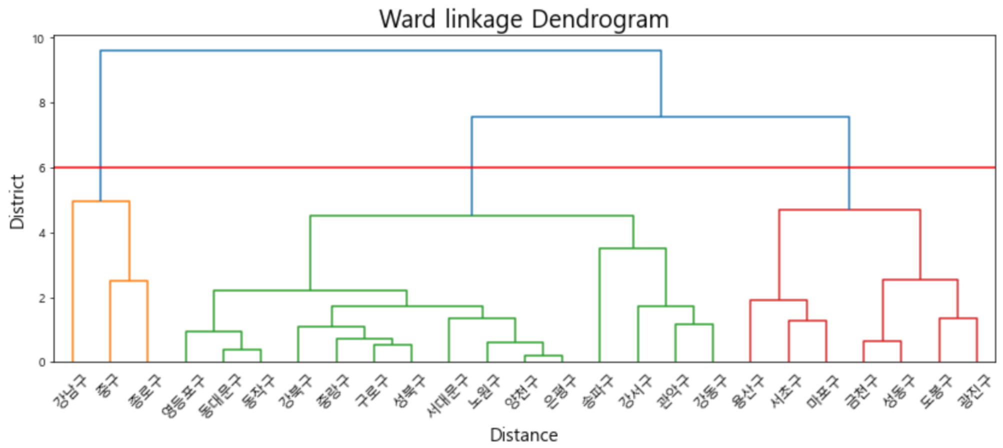
And then we adopted several algorithms and have struggled to interpret each clustering result.

4.1.1 AGNES

First, we did agglomerative hierarchical clustering. Using Euclidean distance and Ward linkage, we thought 3 is good for the number of clustering. The reason of using ward linkage is the result of trial and error. Indeed, Ward linkage is best among the several linkage(single, complete, average, etc..)

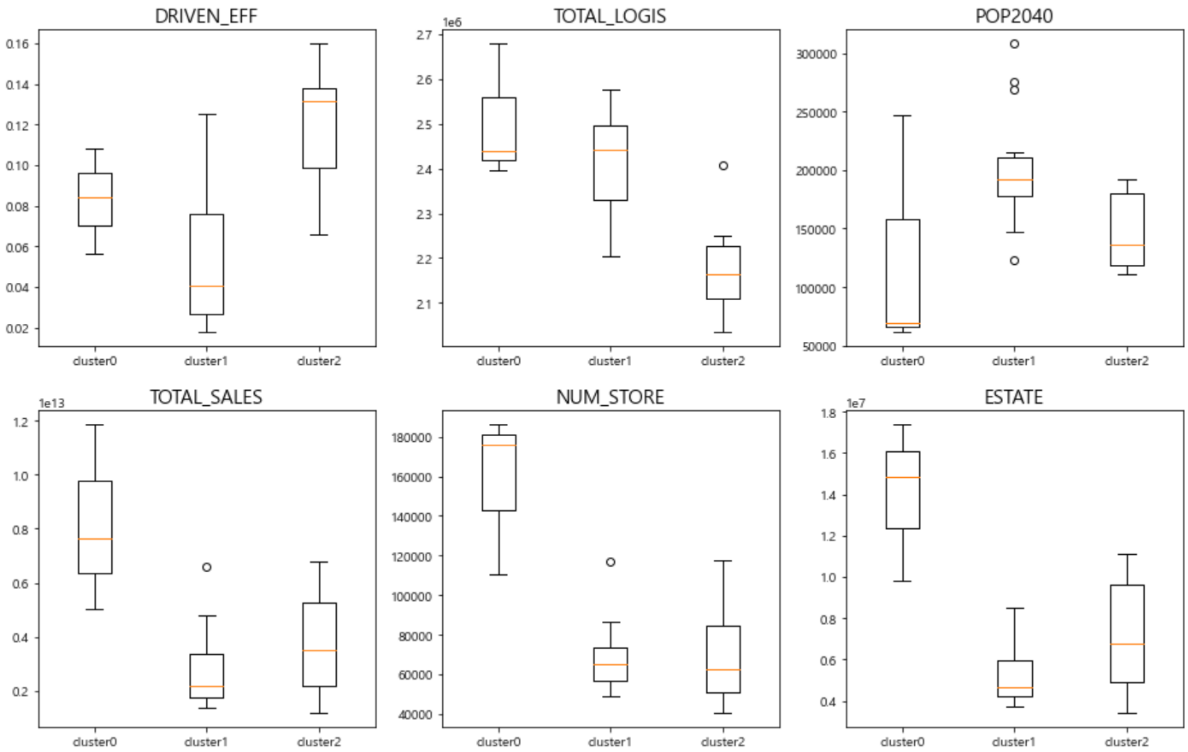
```python
In [12]: from scipy.cluster.hierarchy import linkage, dendrogram, fcluster

plt.figure(figsize=(14, 5))
ax = plt.subplot(111)
dendrogram(linkage(values_pca, method='ward', metric = 'euclidean'), labels = df_table['DISTRICT'].values, leaf_font_size=12)
plt.axhline(y = 6, color = 'red')
plt.title('Ward linkage Dendrogram', size=20)
plt.xlabel('Distance', size=15)
plt.ylabel('District', size=15)
plt.show()
```



- **Cluster0** : 강남구, 중구, 종로구
- **Cluster1** : 영등포구, 동대문구, 동작구, 강북구, 중랑구, 구로구, 성북구, 서대문구, 노원구, 양천구, 은평구, 송파구, 강서구, 관악구, 강동구
- **Cluster2** : 용산구, 서초구, 마포구, 금천구, 성동구, 도봉구, 광진구

Using this result, each cluster was interpreted by performing a boxplot.



- **Cluster0**
  - Highest logistics, superior consumption, and large commerce
  - Un-tact consumption is expected to be small due to high demand for logistics, but large commercial districts
  - Because of the small population, it is difficult to expect an increase in demand
  - The officially assessed reference land price is the largest -> **Not suitable for MFC location**
- **Cluster1**
  - Logistics volume and population are good
  - Because it is a small commerce area and has high offline consumption, it can be expected that un-tact consumption will be large
  - Low officially assessed reference land price and operation efficiency ->

Suitable for MFC location

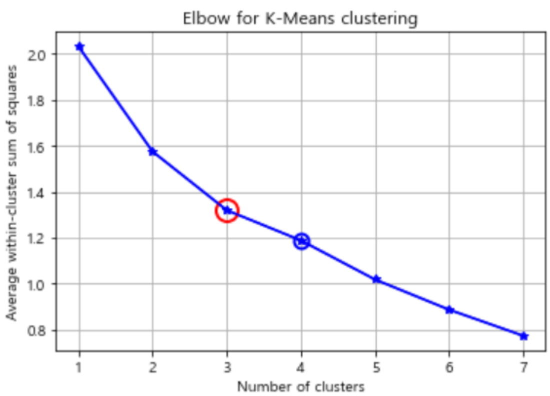
- **cluster2**
  - Low officially assessed reference land price
  - However, the logistics service rate is already high because the logistics volume is the lowest and the operation efficiency is the highest. -> **Not suitable for MFC location**

Therefore, Cluster1 is best suitable for MFC location.

4.1.2 K-means

In K-means, we had to determine the number of clustering “K”. To solve that problem, we used “Elbow method”.

```
In [22]: elbow_method(values_pca)
```



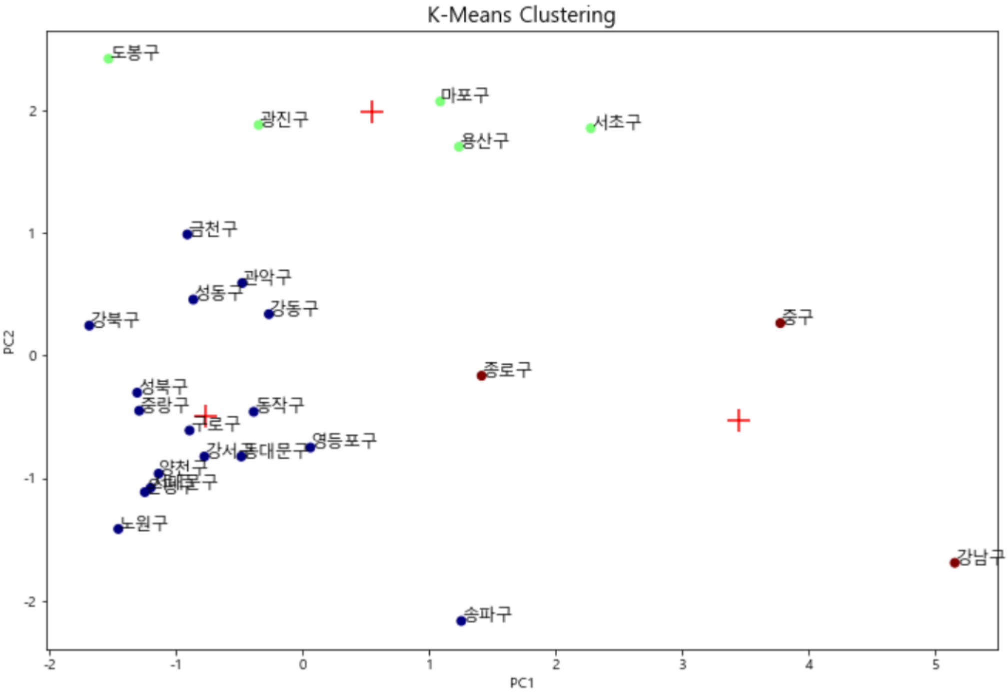
We determined the number k=3. Because when draw a line from where the number of clusters is 3 to 7, the last overlapping point is 3. And this point is like elbow. Also, because we had set the number of clusters in agglomerative dendrogram, we thought k=3 is the best.

```
In [23]: kmeans = KMeans(init='k-means++', n_clusters=3, random_state=100) # Opitmal K = 3
         kmeans.fit(values_pca)

Out[23]: KMeans(n_clusters=3, random_state=100)
```

And visualize the result of k-means. A Blue color is Cluster0, Green color is

Cluster1, and Brown color is Cluster2.

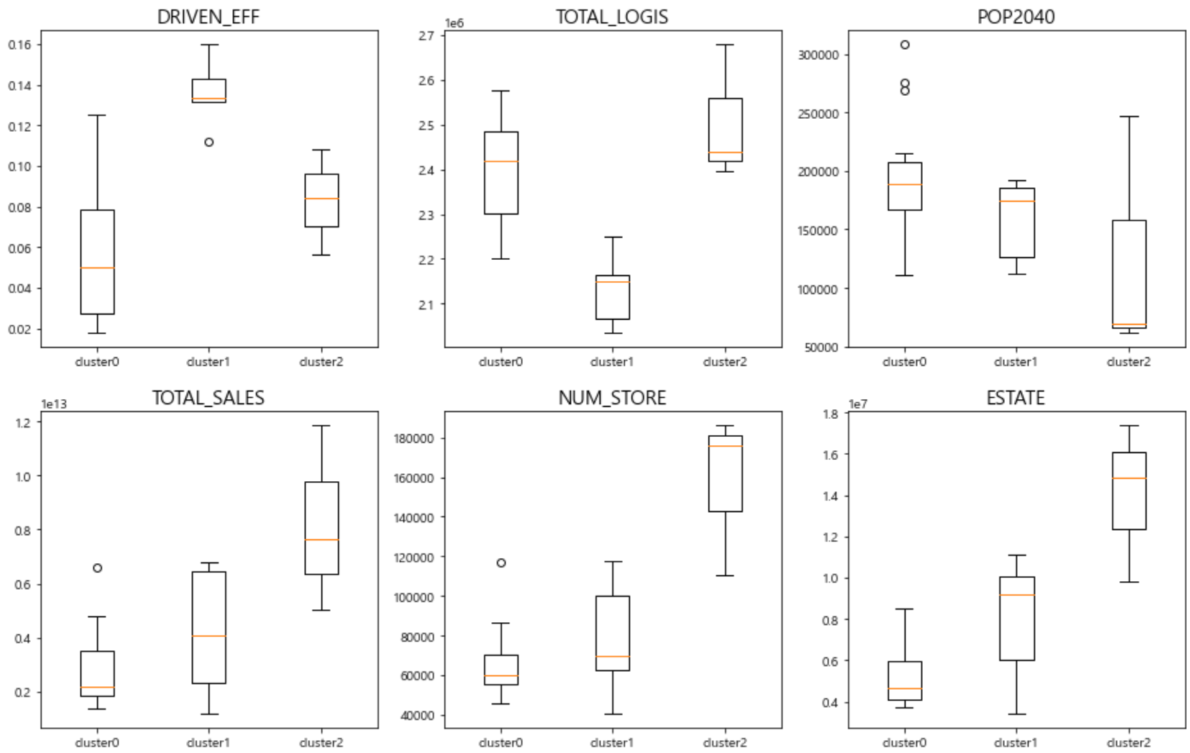


- **Cluster0** : 동대문구, 구로구, 노원구, 성북구, 강북구, 금천구, 영등포구, 중랑구, 관악구, 양천구, 서대문구, 송파구, 강서구, 강동구, 성동구, 동작구, 은평구

- **Cluster1** : 도봉구, 광진구, 마포구, 용산구, 서초구

- **Cluster2** : 강남구, 중구, 종로구

As before, we used EDA with boxplot to select which cluster is best.



- **Cluster0**
  - ➢ Logistics volume and population are good
  - ➢ Although the population is large, the size of the commercial district is small, so you can expect an increase in un-tact consumption
  - ➢ Lowest officially assessed reference land price and operation efficiency -> **<u>Suitable for MFC location</u>**
- **Cluster1**
  - ➢ Lowest logistics
  - ➢ Since the operation efficiency is the highest, it could be said that the logistics service rate is already high -> **<u>MFC location do not needed</u>**
- **Cluster2**
  - ➢ Highest logistics

- the highest commercial district and consumption power relative to the population
- But, officially assessed reference land price is so high -> **Not suitable for MFC location**

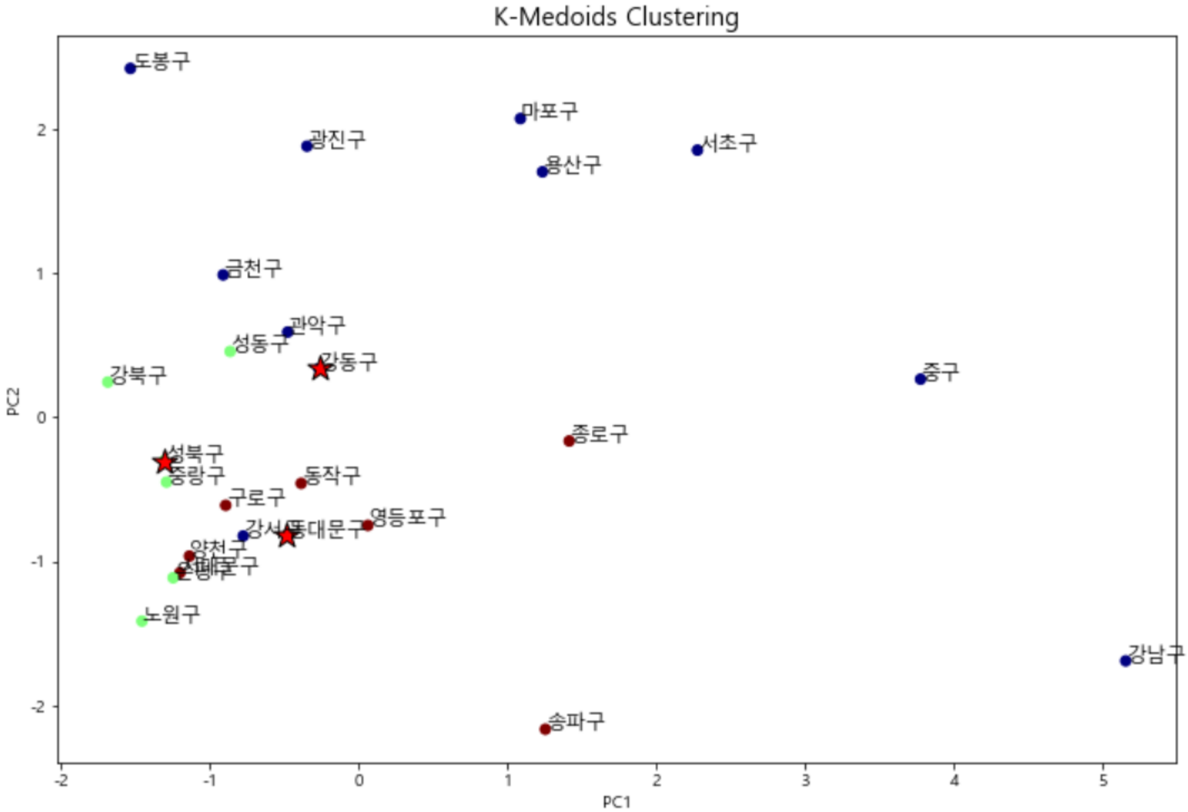
Therefore, Cluster0 is best suitable for MFC location.

4.1.3 K-medoids

We have learned that K-means is sensitive to outliers. So we did K-medoids instead of K-means. We have already set K=3 and we have kept it here.

```
In [36]: from sklearn_extra.cluster import KMedoids
         kmedoids = KMedoids(n_clusters=3, random_state=100).fit(values_pca)
```

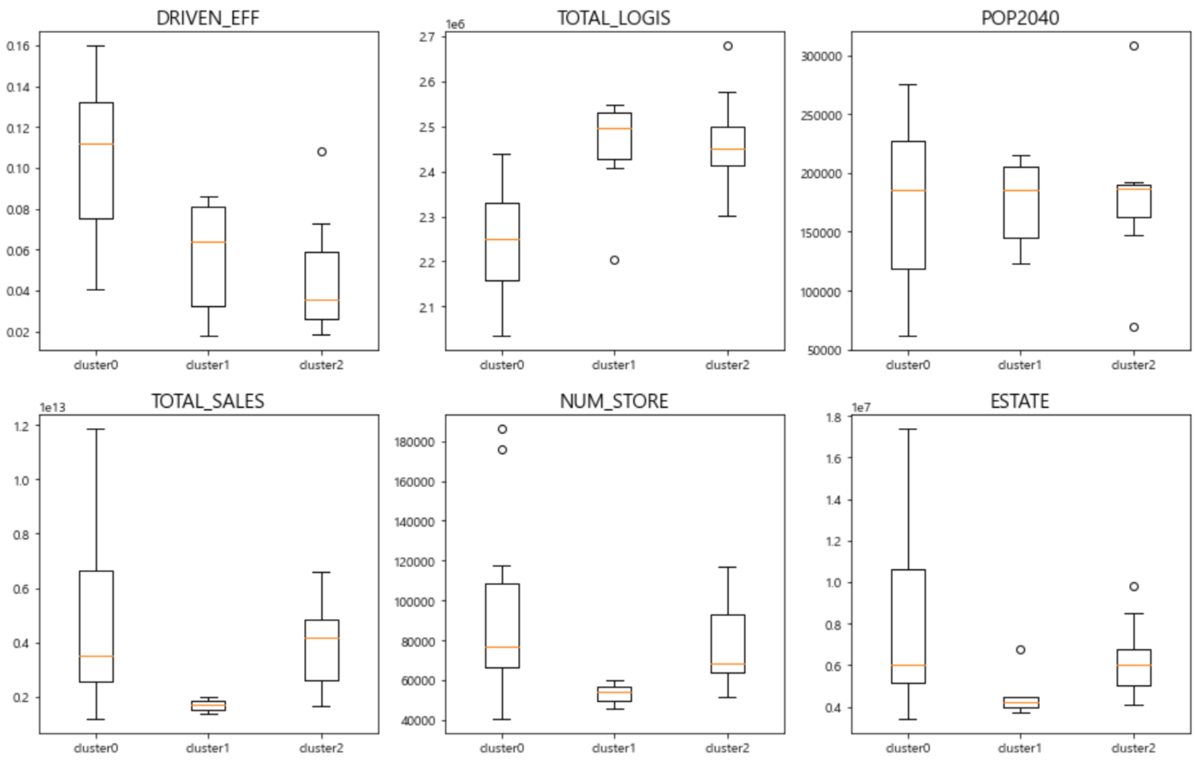
Visualize the result of k-medoids. A Blue color is Cluster0, Green color is Cluster1, and Brown color is Cluster2.



- **Cluster0** : 용산구, 서초구, 마포구, 도봉구, 금천구, 관악구, 광진구, 강남구, 중구, 강서구, 강동구
- **Cluster1** : 노원구, 성북구, 강북구, 중랑구, 성동구, 은평구
- **Cluster2** : 동대문구, 구로구, 영등포구, 양천구, 서대문구, 송파구, 종로구, 동작구

Each cluster is represented by a one of the objects in the cluster. It seems to be low performance than k-means. Looked like low intra-class similarity but didn't ignore it because something meaningful could be hidden.

As before, we used EDA with boxplot to select which cluster is best.



- **Cluster0**
  - ➢ Un-tact consumption is expected to be small due to the largest population and the largest offline consumption and commercial district
  - ➢ Logistics services are currently running well because of the smallest logistics volume but high operational efficiency
  - ➢ The officially assessed reference land price and its width are the largest -> **<u>Not suitable for MFC location</u>**
- **Cluster1**
  - ➢ Logistics volume and population are good
  - ➢ Small commercial district size, so we can expect un-tact consumption
  - ➢ Lowest the officially assessed reference land price and operation efficiency is not good -> **<u>MFC location is not bad</u>**
- **Cluster2**

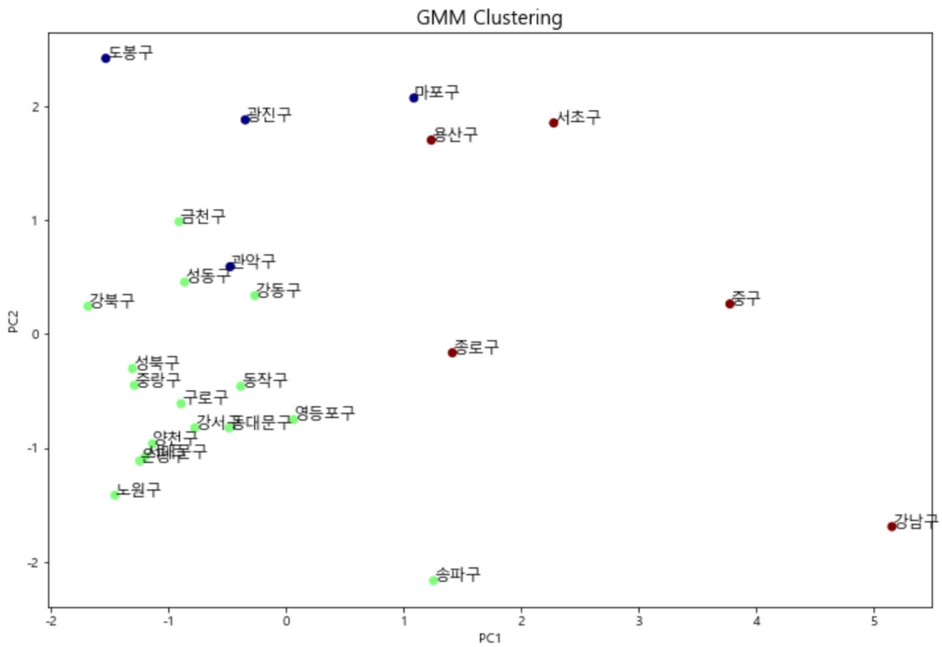
- ➢ Logistics volume and population are good
- ➢ Although the officially assessed reference land price is higher than Cluster1, the operation efficiency is the lowest and sales and commercial area size are larger than Cluster1 -> **More suitable for MFC location than Cluster1**

Therefore, Cluster2 is best suitable for MFC location.

4.1.4 GMM

We didn't learned about Gaussian Mixture model, but it is one of the most popular techniques for clustering. We performed clustering on the assumption that the dataset of the analysis targets was generated by a combination of data with Gaussian distributions. And we also wondered how the results were different.

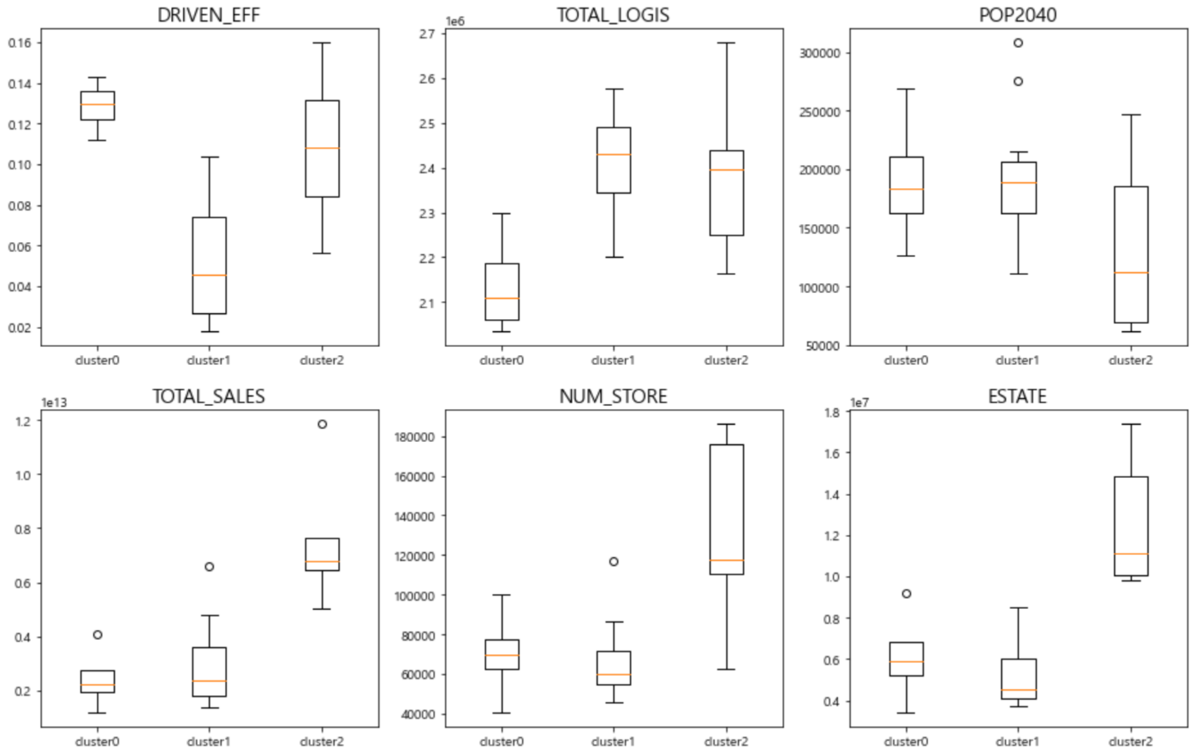
```
In [45]: gmm = GaussianMixture(n_components=3, random_state=100)
         gmm.fit(values_pca) # GMM 클러스터링 수행
```



- **Clutser0** : 마포구, 도봉구, 광진구, 관악구
- **Cluster1** : 동대문구, 구로구, 노원구, 성북구, 강북구, 금천구, 영등포구, 중

랑구, 양천구, 서대문구, 송파구, 강서구, 강동구, 성동구, 동작구, 은평구

- **Cluster2** : 용산구, 서초구, 강남구, 중구, 종로구



- **Cluster0**
  - ➢ Low officially assessed reference land price and average population
  - ➢ Since the operation efficiency is the highest and lowest logistics, it could be said that the logistics service rate is already high -> **<u>Not suitable for MFC location</u>**
- **Cluster1**
  - ➢ Logistics volume and Population is good
  - ➢ It can be expected to consume un-tact consumption because commercial area size is small
  - ➢ The officially assessed reference land price and operation efficiency is

low -> **<u>Suitable for MFC location</u>**

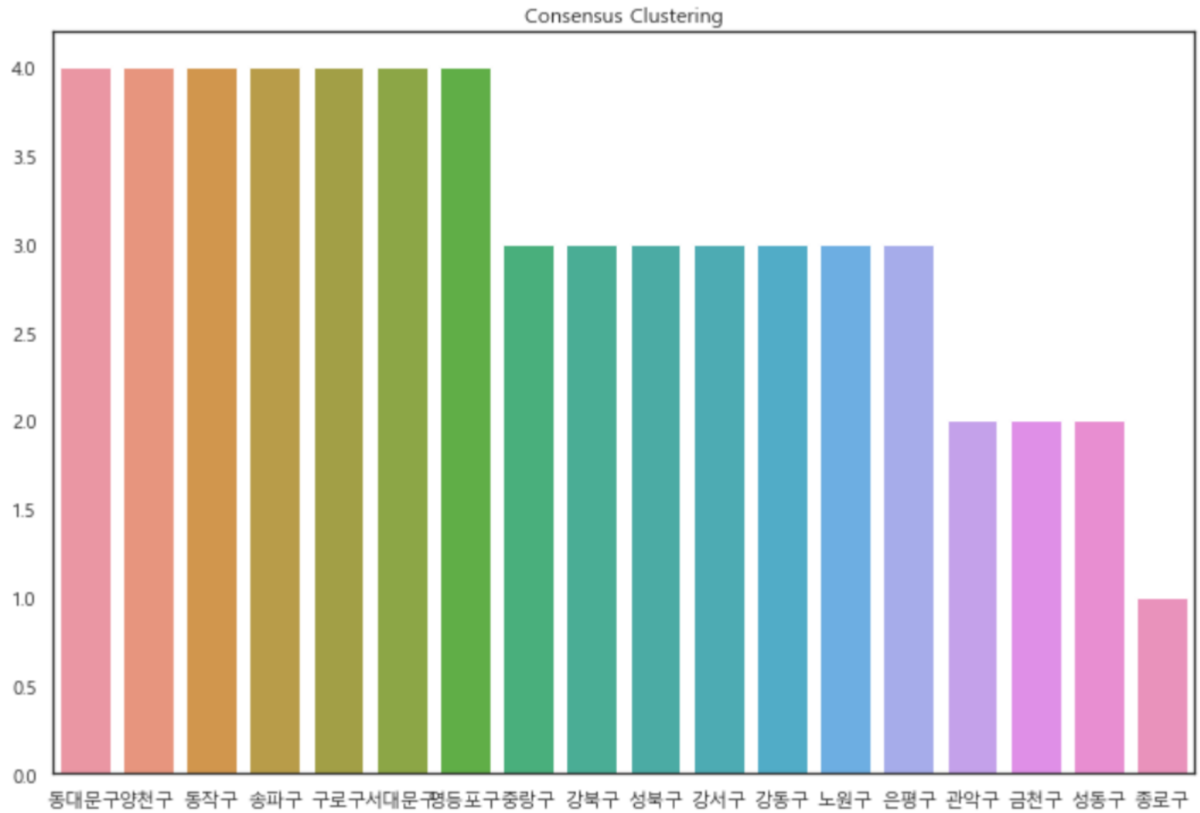
- **Cluster2**

  ➢ Logistics volume is good but Inconsistent demand due to wide range

  ➢ Operational efficiency is high and the officially assessed reference land price is too high -> **<u>Not suitable for MFC location</u>**

Therefore, Cluster1 is best suitable for MFC location.

We could not declare what clustering algorithm is more suitable. We thought there is no specially better the other one. So, we did consensus clustering about 4 algorithms.



In this case, y-label is Rank means if it is more counted, more suitable for MFC location. Therefore, These districts(동대문구, 양천구, 동작구, 송파구, 구로구, 서대문구, 영등포구) are more needed Micro-Fulfillment center. They are prior to the other districts. The table below is the result of the collection.

Rank(Candidate District where will be installed MFC)

1st	2nd	3rd	4rd
Dongdaemun	Jungnang	Gwanak	Jongno
Yangcheon	Gangbuk	Geumcheon	
Dongjak	Seongbuk	Seongdong	
Songpa	Gangseo		
Guro	Gangdong		
Seodaemun	Nowon		
Yeongdeungpo	Eunpyeong		

4.2 Classification

Until now, we had figured out that what districts are needed Micro-fulfillment center. So, we will now create a model to classify the appropriate gas station as an MFC. To know what gas station is suitable, we had calculated a score. According to a paper, the factors that determine the logistic center are land price, size of area, size of the commerce, distance to supplier, population of area, etc,,. Using this factor, we had to calculate score to do so we needed weight for each column. A AHP method is good for getting weight, and then we discussed about what column is more meaningful. The scale is as follows:

척도	정의
1	Equal importance
3	Moderate importance
5	Strong importance
7	Very strong importance
9	Absolute importance
2,4,6,8	Intermediate values of the above values

We got a lot of debates, and finally we get a weight of each feature. Right side is the result of AHP method.

Pairwise Comparison Matrix

	AREA	ESTATE	DISTANCE	POPULATION	SCHOOL	APARTMENT	MARKET
AREA	1	1/2	3	1/3	9	7	5
ESTATE	2	1	3	3	9	7	6
DISTANCE	1/3	1/3	1	1/3	9	5	3
POPULATION	3	1/3	3	1	9	6	5
SCHOOL	1/9	1/9	1/9	1/9	1	1/3	1/4
APARTMENT	1/7	1/7	1/5	1/6	3	1	1
MARKET	1/5	1/6	1/3	1/5	4	1	1

AREA	0.198
ESTATE	0.327
DISTANCE	0.124
POPULATION	0.244
SCHOOL	0.020
APARTMENT	0.040
MARKET	0.048

And then get a score with equation : SCORE = (AREA*0.198) - (ESTATE*0.327) - (DIST_EASTDC*0.0413) - (DIST_WESTDC*0.0413) - (DIST_KOREADC*0.0413) + (POP1000*0.244) + (SCHOOL*0.020) + (APART*0.040) + (MARKET *0.048)

	AREA	ESTATE	DIST_EASTDC	DIST_WESTDC	DIST_KOREADC	POP1000	SCHOOL	APART	MARKET	SCORE
0	0.043427	0.189831	0.318940	0.438800	0.443422	0.253190	0.357143	0.018519	0.983051	0.013764
1	0.169609	0.432974	0.341411	0.418207	0.436269	0.411800	0.642857	0.037037	0.101695	-0.037691
2	0.166631	0.345562	0.346831	0.413507	0.424874	0.435364	0.642857	0.018519	0.372881	0.008770
3	0.250571	0.236987	0.349078	0.411788	0.450253	0.489771	0.357143	0.037037	0.067797	0.053482
4	0.117620	0.305306	0.395409	0.373522	0.403165	0.442966	0.642857	0.111111	0.338983	0.016702

And using quct() function, generate categorical variable "SCORE_CAT".

```
In [11]: df['SCORE_CAT'] = pd.qcut(dfdf.SCORE ,4, labels=['매우부적합', '부적합', '적합', '매우적합'])
         df
```

	AREA	ESTATE	DIST_EASTDC	DIST_WESTDC	DIST_KOREADC	POP1000	SCHOOL	APART	MARKET	SCORE_CAT
0	245.0	8280000.0	8.970251	13.202286	11.678968	15660	5	1	58	부적합
1	711.0	18850000.0	9.568763	12.593357	11.494239	25467	9	2	6	매우부적합
2	700.0	15050000.0	9.713120	12.454393	11.199948	26924	9	1	22	부적합
3	1010.0	10330000.0	9.772973	12.403577	11.855362	30288	5	2	4	적합
4	519.0	13300000.0	11.006969	11.272070	10.639303	27394	9	6	20	부적합

For doing classification, we split the data with train_test_split.(train:test = 7:3)

In [12]:
```
feature_cols = df.columns[[5,6,7,8,9,12,13,14,15]]
# define X and y
X = df[feature_cols]
y = df.SCORE_CAT
```

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1113)
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(328, 9) (328,)
(141, 9) (141,)
```

And use models K-Nearest Neighbors Classifier, Logistic Regression, Decision Tree, RandomForest, GradientBoosting Classifier.

<Decision Tree>

In [16]:
```
clf = DecisionTreeClassifier(random_state=0).fit(X_train, y_train)
predicted=clf.predict(X_test)
print ('Confusion Matrix :')
print(confusion_matrix(y_test, predicted))
accuracy_score_clf = 'Accuracy Score :', accuracy_score(y_test, predicted)
print(accuracy_score_clf)
print ('Report : ')
print (classification_report(y_test, predicted))
```

```
Confusion Matrix :
[[35  0  9  1]
 [ 0 33  0  2]
 [10  0 16  4]
 [ 0  5  4 22]]
('Accuracy Score :', 0.75177304964539)
Report :
              precision    recall  f1-score   support

       매우부적합       0.78      0.78      0.78        45
        매우적합       0.87      0.94      0.90        35
         부적합       0.55      0.53      0.54        30
          적합       0.76      0.71      0.73        31

    accuracy                           0.75       141
   macro avg       0.74      0.74      0.74       141
weighted avg       0.75      0.75      0.75       141
```

<Logistic Regression>

```
In [18]: lr = LogisticRegression(C=20, max_iter=1000, random_state=0).fit(X_train, y_train)
predicted=lr.predict(X_test)
print ('Confusion Matrix :')
print(confusion_matrix(y_test, predicted))
accuracy_score_lr = 'Accuracy Score :',accuracy_score(y_test, predicted)
print(accuracy_score_lr)
print ('Report : ')
print (classification_report(y_test, predicted))
```

```
Confusion Matrix :
[[31  1 12  1]
 [ 0 29  3  3]
 [ 0  3 21  6]
 [ 1 12  5 13]]
('Accuracy Score :', 0.6666666666666666)
Report :
              precision    recall  f1-score   support

      매우부적합       0.97      0.69      0.81        45
       매우적합       0.64      0.83      0.73        35
        부적합       0.51      0.70      0.59        30
         적합       0.57      0.42      0.48        31

    accuracy                           0.67       141
   macro avg       0.67      0.66      0.65       141
weighted avg       0.70      0.67      0.67       141
```

<K-Nearest Neighbors>

KNN is needed to scaler. So, we transform data with MinMaxScaler before modeling.

```
In [19]: scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
In [20]: knn = KNeighborsClassifier(n_neighbors=11).fit(X_train_scaled, y_train)
predicted=knn.predict(X_test_scaled)
print ('Confusion Matrix :')
print(confusion_matrix(y_test, predicted))
accuracy_score_knn = 'Accuracy Score :',accuracy_score(y_test, predicted)
print(accuracy_score_knn)
print ('Report : ')
print (classification_report(y_test, predicted))
```

```
Confusion Matrix :
[[34  1  9  1]
 [ 0 26  1  8]
 [ 3  1 15 11]
 [ 0  5  7 19]]
('Accuracy Score :', 0.6666666666666666)
Report :
              precision    recall  f1-score   support

      매우부적합       0.92      0.76      0.83        45
       매우적합       0.79      0.74      0.76        35
        부적합       0.47      0.50      0.48        30
         적합       0.49      0.61      0.54        31

    accuracy                           0.67       141
   macro avg       0.67      0.65      0.66       141
weighted avg       0.70      0.67      0.68       141
```

<Random Forest>

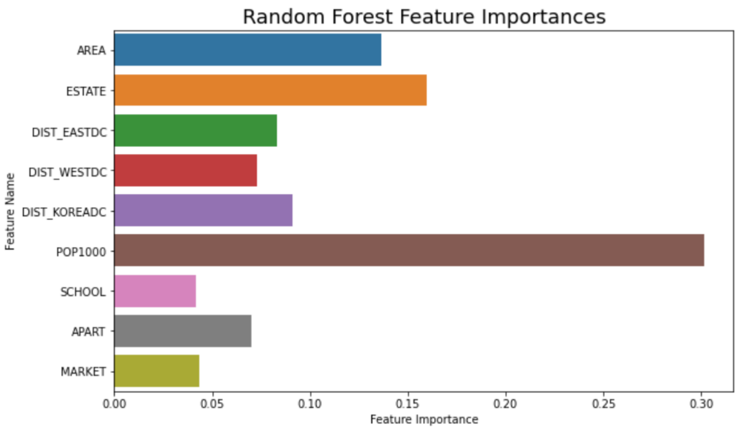
In [21]:
```
forest = RandomForestClassifier(max_depth = 12, random_state=0).fit(X_train, y_train)
predicted=forest.predict(X_test)
print ('Confusion Matrix :')
print(confusion_matrix(y_test, predicted))
accuracy_score_forest = 'Accuracy Score :',accuracy_score(y_test, predicted)
print(accuracy_score_forest)
print ('Report : ')
print (classification_report(y_test, predicted))
```

```
Confusion Matrix :
[[37  0  8  0]
 [ 0 31  2  2]
 [ 4  0 19  7]
 [ 2  7  2 20]]
('Accuracy Score :', 0.7588652482269503)
Report :
              precision    recall  f1-score   support

       매우부적합       0.86      0.82      0.84        45
        매우적합       0.82      0.89      0.85        35
         부적합       0.61      0.63      0.62        30
          적합       0.69      0.65      0.67        31

    accuracy                           0.76       141
   macro avg       0.74      0.75      0.74       141
weighted avg       0.76      0.76      0.76       141
```

In [23]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='Feature Importance', y='Feature Name', data=pd.DataFrame([(i, j) for i, j in
plt.title('Random Forest Feature Importances', fontsize=18)
plt.show()
```



The Population is most important features in Random Forest. 2^{nd} is Estate, and 3^{rd} is Area.

<GradientBoost Classifier>

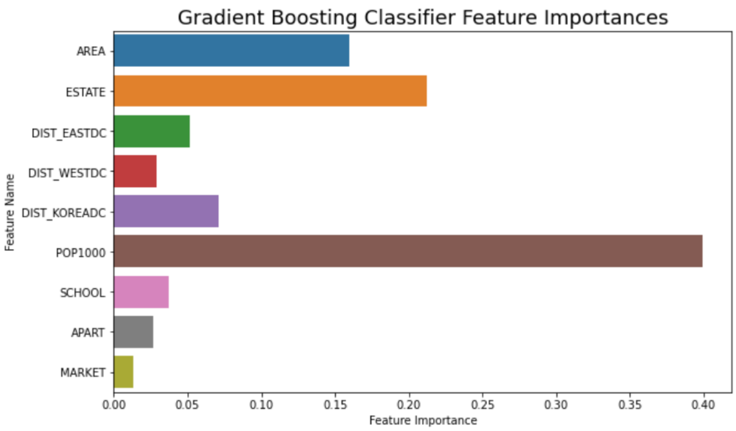
```
In [24]: gb = GradientBoostingClassifier(max_depth=7, random_state=0).fit(X_train, y_train)
         predicted=gb.predict(X_test)
         print ('Confusion Matrix :')
         print(confusion_matrix(y_test, predicted))
         accuracy_score_gb = 'Accuracy Score :',accuracy_score(y_test, predicted)
         print(accuracy_score_gb)
         print ('Report : ')
         print (classification_report(y_test, predicted))
```

```
Confusion Matrix :
[[38  0  7  0]
 [ 0 29  1  5]
 [ 5  0 19  6]
 [ 0  4  5 22]]
('Accuracy Score :', 0.7659574468085106)
Report :
              precision    recall  f1-score   support

       매우부적합       0.88      0.84      0.86        45
        매우적합       0.88      0.83      0.85        35
         부적합       0.59      0.63      0.61        30
          적합       0.67      0.71      0.69        31

    accuracy                           0.77       141
   macro avg       0.76      0.75      0.75       141
weighted avg       0.77      0.77      0.77       141
```

```
In [26]: plt.figure(figsize=(10, 6))
         sns.barplot(x='Feature Importance', y='Feature Name', data=pd.DataFrame([(i, j) for i, j in
         plt.title('Gradient Boosting Classifier Feature Importances', fontsize=18)
         plt.show()
```

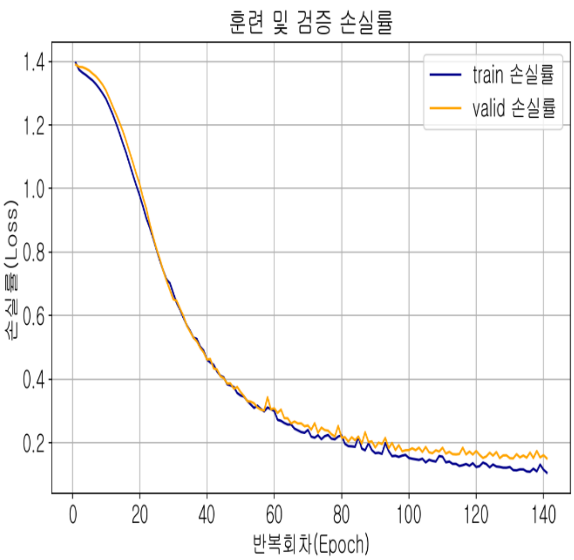


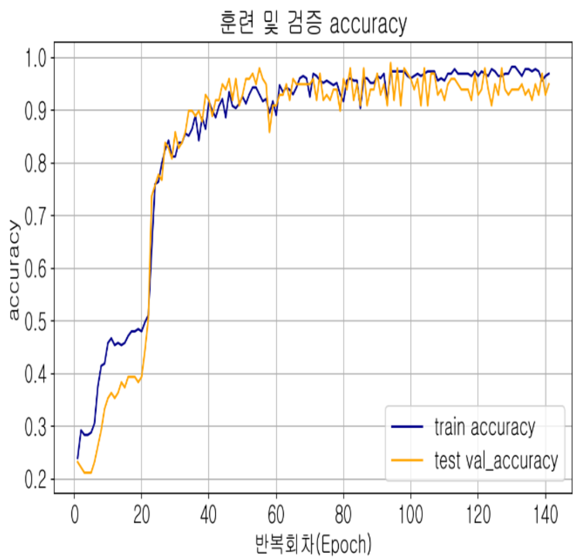
Population is most important in Gradient Boosting too. 2nd is Estate, and 3rd is Area.

4.3 Multi-Class_Classification Artificial Neural Network

We used multi-classification neural network technology to predict four classes. As a result of observing the loss rate and accuracy, it was trained well without overfitting. The final loss rate is 12 percent and the final accuracy is 95 percent. For the result of binary classification is 1 - Loss rate = precision, but

our result is appropriate because it is multiple classification.






```
let_evaluate = neural_model.evaluate(train_scaled, train_target)
print('최종 손실률 : %f, 최종 accuracy : %f' % (let_evaluate[0], let_evaluate[1]))
```

```
8/8 [==============================] - 0s 4ms/step - loss: 0.1029 - sparse_categorical_accuracy: 0.9738
최종 손실률 : 0.102898, 최종 accuracy : 0.973799
```

Lastly, we had compared the performance with accuracy.

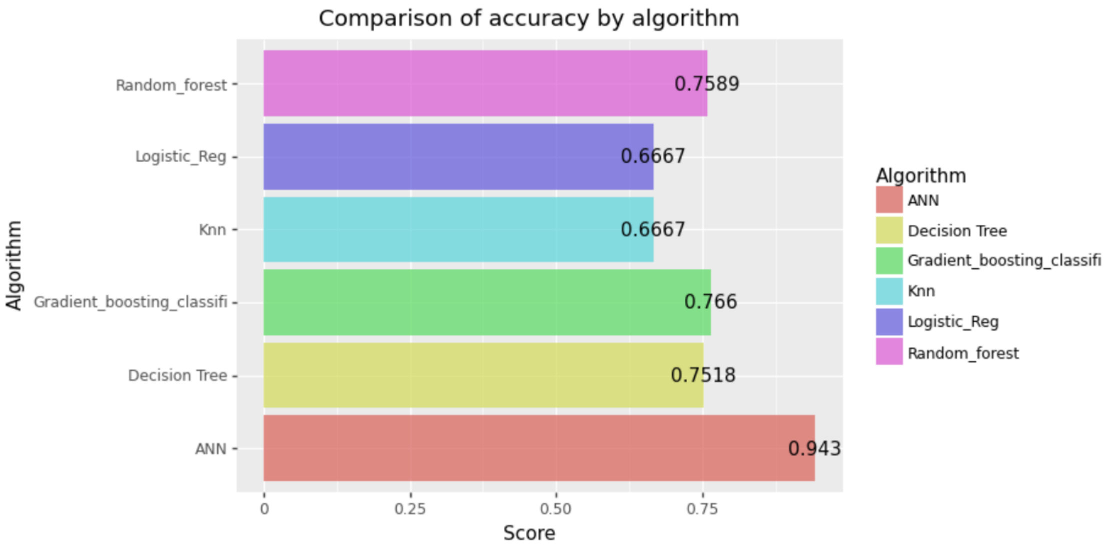
In [47]:
```
import plotnine
from plotnine import *
```

In [48]:
```
df_accuracy = pd.DataFrame({
    'Algorithm' : ['Logistic_Reg',  'Knn', 'Decision Tree', 'Random_forest','Gradient_boosting_classifi', 'ANN'],
    'Score': [round(accuracy_score_lr[1],4),round(accuracy_score_knn[1],4),  round(accuracy_score_clf[1], 4),
             round(accuracy_score_forest[1], 4),
            round(accuracy_score_gb[1], 4),
             round(let_evaluate[1], 3)]})
df_accuracy
```

Out[48]:

	Algorithm	Score
0	Logistic_Reg	0.6667
1	Knn	0.6667
2	Decision Tree	0.7518
3	Random_forest	0.7589
4	Gradient_boosting_classifi	0.7660
5	ANN	0.9430

```
In [49]: p = ggplot(df_accuracy, aes(x = 'Algorithm',
                             y = 'Score',
                             fill = 'Algorithm',)) + geom_col(alpha = 0.7)
p + coord_flip() + geom_text(aes(label = 'Score')) + ggtitle('Comparison of accuracy by algorithm')
```



Understandably, ANN is the best classification model among them with accuracy score 94.3%.

Finally, we have the MFC demand region (the region where MFC is most neede d) as a result of clustering. And using neural network classification, we selected the best gas station for MFC location.

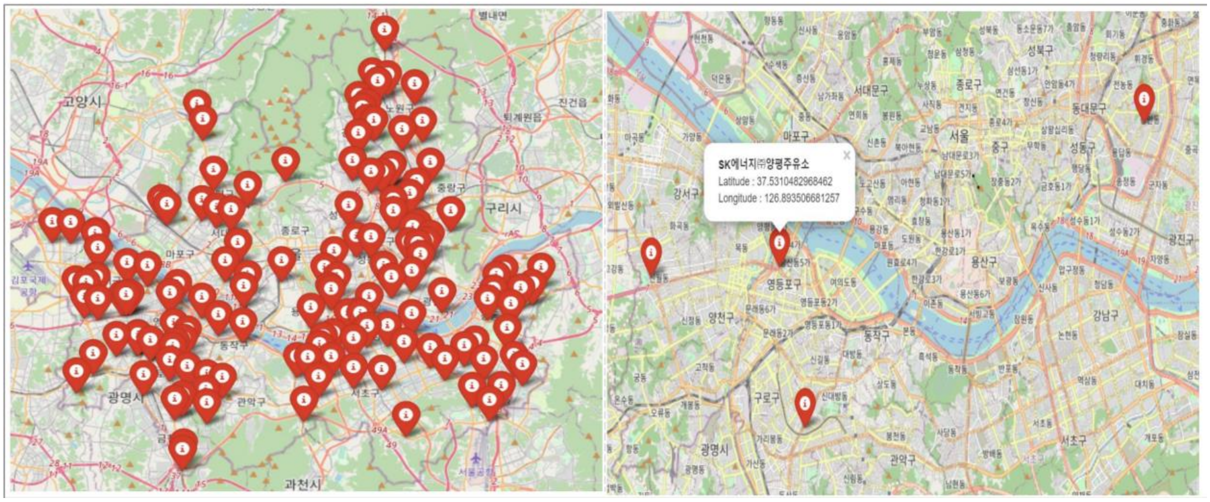
	GAS_STATION	DISTRICT	DONG	SCORE_CAT	Actual_Location_Score	Predicted_value_by_ANN	SCORE
49	삼미상사㈜ 장안킹셀프주유소	동대문구	장안동	매우적합	1	1	0.119815
52	삼영주유소	동대문구	장안동	매우적합	1	1	0.108965
55	배봉로주유소	동대문구	전농동	매우적합	1	1	0.084279
60	대성산업㈜청량리주유소	동대문구	청량리동	매우적합	1	1	0.091912
147	에이치지 가로공원주유소	양천구	신월동	매우적합	1	1	0.113783
149	동일석유㈜ 개나리주유소	양천구	신월동	매우적합	1	1	0.143189
165	양천구주유소	양천구	목동	매우적합	1	1	0.118080
235	SK에너지㈜양평주유소	영등포구	양평동3가	매우적합	1	1	0.160287
236	SK에너지㈜ 기린주유소	영등포구	양평동4가	매우적합	1	1	0.145911
243	㈜정수에너지개발	영등포구	신길동	매우적합	1	1	0.106042
247	㈜엠에스주유소	영등포구	대림동	매우적합	1	1	0.088108
249	(주)대청에너지	영등포구	대림동	매우적합	1	1	0.115939
251	현대오일뱅크㈜직영신대방셀프주유소	동작구	신대방동	매우적합	1	1	0.146005

However, several gas stations were selected in the same region. Therefore, one optimal location per region was selected using a group-by based on the regio n.

	DISTRICT	SCORE	GAS_STATION	DONG	SCORE_CAT	Actual_Location_Score	Predicted_value_by_ANN
0	동대문구	0.119815	삼미상사㈜ 장안킹셀프주유소	장안동	매우적합	1	1
1	동작구	0.146005	현대오일뱅크㈜직영신대방셀프주유소	신대방동	매우적합	1	1
2	양천구	0.143189	동일석유㈜ 개나리주유소	신월동	매우적합	1	1
3	영등포구	0.160287	SK에너지㈜양평주유소	양평동3가	매우적합	1	1

The results of map visualization are as follows.

Therefore, among of 469 gas stations, 141 MFCs were selected, and the four m ost suitable gas stations for MFCs were selected.



4.4 Association Rule

We want to know which variables affect the cluster group the most. Association rule is indicating how often an event occurs together and how much it is related to each other. The indicators of association rules are usually support, confidence, and lift. Other indicators exist, but we used them with the indicators I learned in class. Since our dataset for using the association rule is numerical data, it was categorized through binning. The numerical data were changed into binary variables, which are greater or smaller than the average.

This is 9 numerical attributes.

	GAS_STATION	AREA	ESTATE	DIST_EASTDC	DIST_WESTDC	DIST_KOREADC	POP1000	SCHOOL	APART	MARKET
1	현대오일뱅크(...	245.00	8280000	8.97025	13.2023	11.679	15660	5	1	58
2	선익상사(주) 동...	711.00	18850000	9.56876	12.5934	11.4942	25467	9	2	6
3	현대오일뱅크㈜...	700.00	15050000	9.71312	12.4544	11.1999	26924	9	1	22
4	서계주유소	1010.00	10330000	9.77297	12.4036	11.8554	30288	5	2	4
5	㈜영원에너지 ...	519.00	13300000	11.007	11.2721	10.6393	27394	9	6	20
6	㈜신태성주유소	447.00	12560000	11.3075	10.9692	10.7474	29460	6	6	5
7	현대오일뱅크㈜...	1090.00	5880000	12.4011	9.80518	11.5668	19479	1	2	1
8	한국석유공업㈜...	729.00	20900000	11.5979	11.2026	9.31318	12041	1	3	5
9	(주)중앙에너비...	365.00	19850000	7.84822	14.9818	8.67607	16531	0	0	2
10	한남제3한강주...	764.00	24360000	7.88949	15.0662	8.44207	15595	0	0	1

Changed into binary attributes through binning (greater or smaller than the average)

	Name	Distribution	Mean
N	AREA		805.6831
N	ESTATE		9526646.06
N	DIST_EASTDC		11.4242
N	DIST_WEST...		15.2233
N	DIST_KORE...		13.2437
N	POP1000		28901.19
N	SCHOOL		2.82
N	APART		4.99
N	MARKET		4.12

```
C area_cat := 0 if AREA >= 805.6831 else 1
C estate_cat := 0 if ESTATE > 9526646.06 else 1
C east_cat := 0 if DIST_EASTDC > 11.4242 else 1
C west_cat := 0 if DIST_WESTDC > 15.2233 else 1
C korea_cat := 0 if DIST_KOREADC > 13.2437 else 1
C pop_cat := 0 if POP1000 > 28901 else 1
C sch_cat := 0 if SCHOOL > 2 else 1
C apart_cat := 0 if APART > 4 else 1
C mkt_cat := 0 if MARKET > 3 else 1
```

After binning

	GAS_STATION	area_cat	estate_cat	east_cat	west_cat	korea_cat	pop_cat	sch_cat	apart_cat	mkt_cat	SCORE_CAT
1	현대오일뱅크(...	Low	Low	Short	Short	Short	Low	High	Low	High	부적합
2	선익상사(주) 동...	Low	High	Short	Short	Short	Low	High	Low	High	매우부적합
3	현대오일뱅크㈜...	Low	High	Short	Short	Short	Low	High	Low	High	부적합
4	서계주유소	High	High	Short	Short	Short	High	High	Low	High	적합
5	㈜영원에너지 ...	Low	High	Short	Short	Short	Low	High	High	High	부적합
6	㈜신태성주유소	Low	High	Short	Short	Short	High	High	High	High	부적합
7	현대오일뱅크㈜...	High	Low	Far	Short	Short	Low	Low	Low	Low	부적합
8	한국석유공업㈜...	Low	High	Far	Short	Short	Low	Low	Low	High	매우부적합
9	(주)중앙에너비...	Low	High	Short	Short	Short	Low	Low	Low	Low	매우부적합
10	한남제3한강주...	Low	High	Short	Short	Short	Low	Low	Low	Low	매우부적합

The most frequent items

Itemsets	Support	%
estate_cat=Low	322	68.66

➔ It was divided into larger or smaller than mean, but we think the reason why the value is so large is because there are a lot of outliers.

Association Rule

Min support: 10%   Min confidence: 65%

- Area = Low, Population = Low, Apart = Low → Score = 매우 부적합 (Lift: 2.865)
- Population = Low, School = Low, Apart = Low → Score = 매우 부적합 (Lift: 2.663)
- Area = High, Estate = Low, Population = High → Score = 매우 적합 (Lift: 2.991)

Min support: 20% Min confidence: 40%

- Estate = Low, Population = High → Location = 매우 적합 (Lift: 2.121)
- Population = Low, Apart = Low → Location = 매우 부적합 (Lift: 2.292)

➔ Through the association rules, it tends to be that the larger area, the larger population, and the lower official land price, the more suitable for MFC location. It is similar to Random Forest and Gradient Boosting results.

5. Conclusion

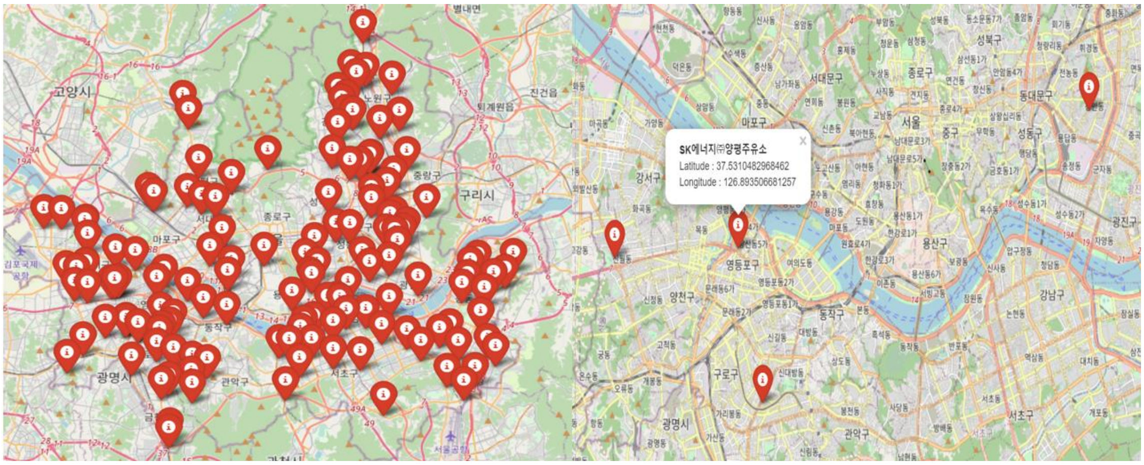
5.1. Project Conclusion

Our project can make 2 main conclusions. First, it is a conclusion about the area where MFC is needed. Select a suitable location for entering the MFC through 4 clustering methods. After that, make a consensus with 4 methods. The area selected with the most frequency is where MFC entry is most needed. This is the rank of the districts where MFC location is required.

Rank(Candidate District where will be installed MFC)

1st	2nd	3rd	4rd
Dongdaemun	Jungnang	Gwanak	Jongno
Yangcheon	Gangbuk	Geumcheon	
Dongjak	Seongbuk	Seongdong	
Songpa	Gangseo		
Guro	Gangdong		
Seodaemun	Nowon		
Yeongdeungpo	Eunpyeong		

Among the “141” gas station in Seoul, only “4” gas stations that are most suitable for Micro fulfillment center



5.2. Expectation effectiveness

- SEOUL

There is a possibility that the introduction of MFC in Seoul on a trial basis will cause innovation in the logistics industry. → Recession caused by COVID-19, can revitalize the economy

- CUSTOMER

As the logistics delivery process decreases, various products can be received faster than now.

- GAS STATION

Even those who have difficulty building their own fulfillment into the city at a low cost and can increase the competitiveness in the market

- LOGISTICS COMPANY

Provides high delivery service at low cost while gaining advantage in the delivery market.

5.3. Limitation

- Logistics data is only available by district, so logistics variables are not considered in the classification model.

- Considering only the internal conditions of Seoul when analyzing the location.

- There may be other meaningful variables that we did not think.

- We don't use MCLP, so we can't find the best place where covers all demands.

- Our project can't consider coverage.

Dataset Explanation

데이터 명	링크
개별공시지가_2022년 Land Value by public announcement in 2022	https://data.seoul.go.kr/dataList/OA-1180/F/1/datasetView.do
서울시 주유소 현황: Status of Oil station in Seoul	https://www.data.go.kr/data/15098386/fileData.do
주민등록인구(연령별_동별)_2022 : Total population data	https://data.seoul.go.kr/dataList/10727/S/2/datasetView.do
자치구단위 월별 착지 데이터 : Monthly delivery data	http://175.193.201.33/data/selectSampleData.do?r_id=P213&sample_data_seq=327&tab_type=&file_id=&sch_text=cj%EB%8C%80%ED%95%9C%ED%86%B5%EC%9A%B4&sch_order=U&currentPage=1
서울시_상권_매출액 : Commercial sales data in Seoul	https://data.seoul.go.kr/dataList/OA-15572/S/1/datasetView.do
서울시 건축물대장 법정동 코드정보 : Code information of address in seoul	https://data.seoul.go.kr/dataList/OA-15410/S/1/datasetView.do

서울시 우리마을가게 상권분석서비스(상권영역) : Commercial sales Area data	https://data.seoul.go.kr/dataList/OA-15560/S/1/datasetView.do
CJ택배 운행량 월별 통계 : CJ delivery service monthly data	http://175.193.201.33/data/selectSampleData.do?r_id=P213&sample_data_seq=326&tab_type=&file_id=&sch_text=cj%EB%8C%80%ED%95%9C%ED%86%B5%EC%9A%B4&sch_order=U&currentPage=1
물류터미널정보_221001: Distibution Center information	https://www.nlic.go.kr/nlic/fmTerminal0010.action

References

- "CJ대한통운, SK 주유소를 물류거점으로 쓴다."(2021.08.27) 한겨레

- "가구 픽업부터 도심물류거점까지… 주유소 활용 나선 이유는?"

  (2021-10-06) UPI뉴스

- "CJ대한통운, SK에너지, 주유소를 도심 물류센터로"(2021.08.26) 조선비즈

- 영남권 물류센터 입지 선정 최적화에 관한 연구(2017) 윤관호, 차연두, 여기태

- 마이크로 풀필먼트 센터 도입방향 : 지역 중소 유통업체를 중심으로(2022)

  박종헌, 권구포, 김정환

- GIS 및 혼합정수계획법을 이용한 공동물류센터 입지선정 (2016)

  유한솔, 김남석

- GIS를 활용한 대전시 주유소 입지와 판매권역 분석 (2004)