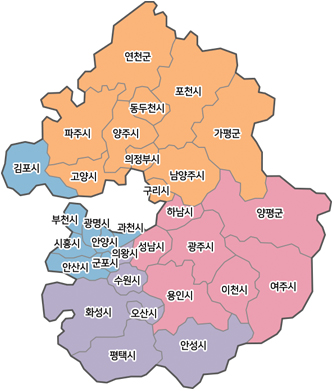
**기말 대체 과제**

2015313098 홍민기

**1. 해결하고자 하는 문제에 대한 설명**

근 10년간 서울에만 존재하던 많은 인프라들, 예를 들어 쇼핑몰이나 직장, 거주지 등이 주변 경기권으로 확장되었다. 또한, 이에 맞추어 경기권에서의 교통 통행량 또한 증가하였다. 이러한 부대 시설들을 이용하기 위해서 이동하는 차량이다. 각 지역은 고유의 특징을 가지고 있다. 어떠한 곳은 쇼핑을 위한 통행량이 많다거나 또는 학교를 가기 위한 통행량, 그리고 출근 등의 통행량이 많이 드러나는 곳이 존재한다. 그리고 이는 경기도교통정보 센터에 기록되어 데이터로 남는다. 이러한 데이터를 이용해 교통 통행량에서 각 지역들의 특징을 도출하여 비슷한 지역끼리 묶고 그 지역의 특성을 파악해보고자 한다.



**2. 데이터에 대한 설명**

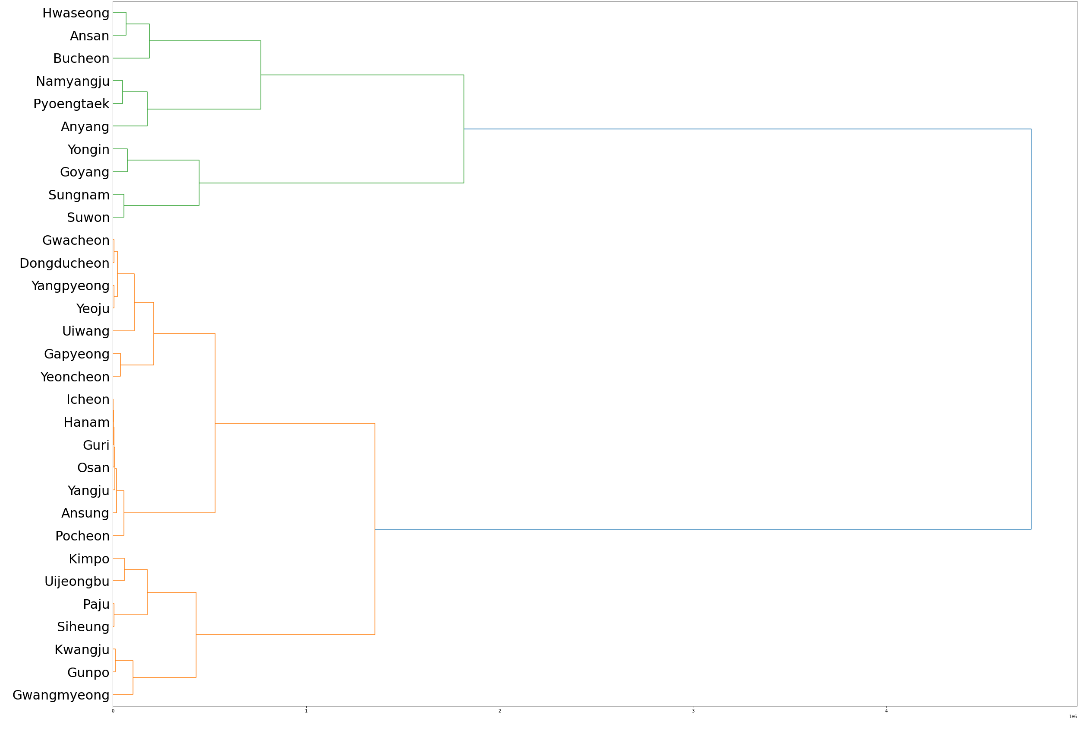
 경기도교통정보센터에 기록 되어있는 통행발생/도착량 통계 중 가장 최근 데이터에 해당하는 2016년도 데이터를 사용하여 분석하였다. 또한 발생과 도착에 해당하는 데이터 중, 도착에 해당하는 데이터들 (출근도착, 등교도착, 학원도착, 업무도착, 쇼핑도착, 여가오락 친교도착, 귀가도착, 기타도착)을 사용하였다. 또한 시, 군 단위로 분류하여 구 단위의 소분류 통계는 제외하였다. 각 열은 ‘시 구분, 출근도착, 등교도착, 학원도착, 업무도착, 쇼핑도착, 여가오락친교도착, 귀가도착, 기타도착, 총목적도착’ 으로 이루어져 있으며 행은 31개의 경기도의 시로 구성되어 있다. 각 데이터는 2016년 일 평균 교통 통계량으로 총목적도착 항목은 그 시의 도착 교통 통계량의 총합이다. 이 데이터를 정리하여 csv 파일로 편집하여 ‘traffic.csv’ 라는 데이터셋을 구성하였다.

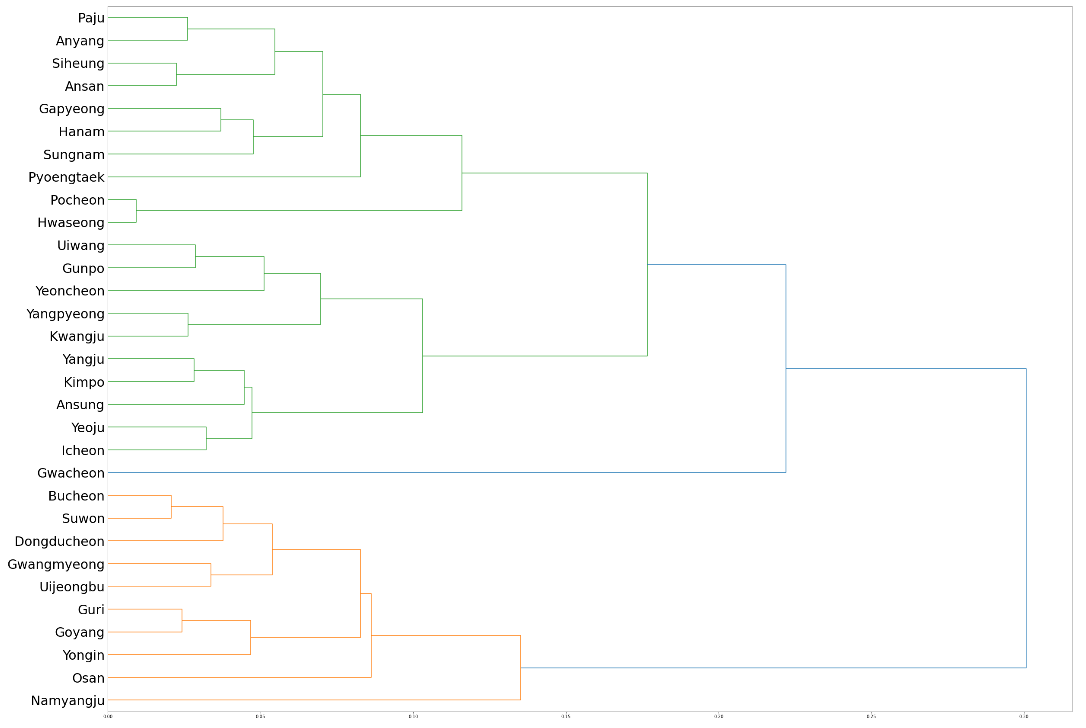
사이트 주소 :   
<https://gits.gg.go.kr/gtdb/web/trafficDb/newzone/HB003/2/IN_OUT/1_IN_OUT_TRIP.do>

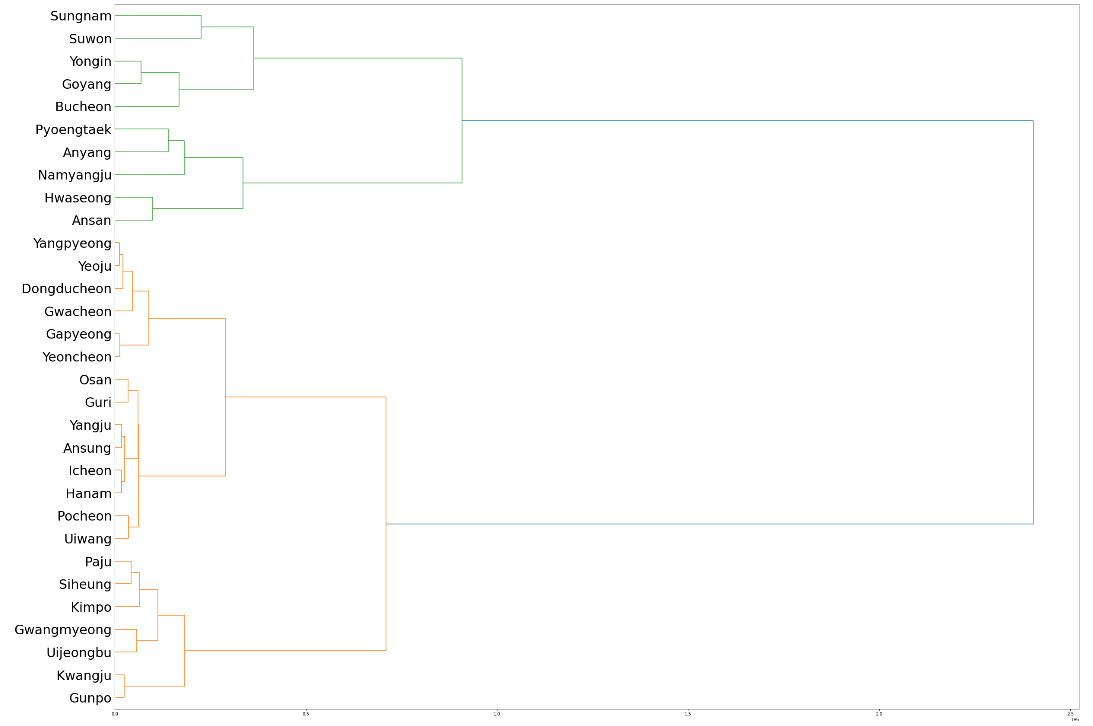
**3. 문제를 해결하기 위해 적용한 방법**

이 데이터를 분석하기 위해서 Hierarchy Clustering 알고리즘을 사용하였다. 데이터 레이블이 없는 같은 특성을 갖는 다항목의 데이터를 분석하기 위해 활용하기 가장 좋은 알고리즘은 Hierarchy Clustering 알고리즘이기 때문이다. Scipy 라이브러리가 제공하는 Hierarchy Clustering 알고리즘을 통해 Dendrogram을 통해 각 도시가 가지는 데이터 유사도를 찾아보고 분석한다. Scipy에서 Dendrogram Label에 한글을 지원하지 않아 국문 도시명을 영어로 직접 입력하였으며 데이터는 각 항목을 총목적도착으로 나누어 퍼센티지를 이용하고 더불어 총목적도착 값을 포함한 데이터, 퍼센티지를 이용하되 총목적도착 값이 포함되지 않은 데이터, 그리고 마지막으로 퍼센티지가 아닌 있는 그대로의 숫자 데이터, 이 세 가지를 통해 분석해보았다.

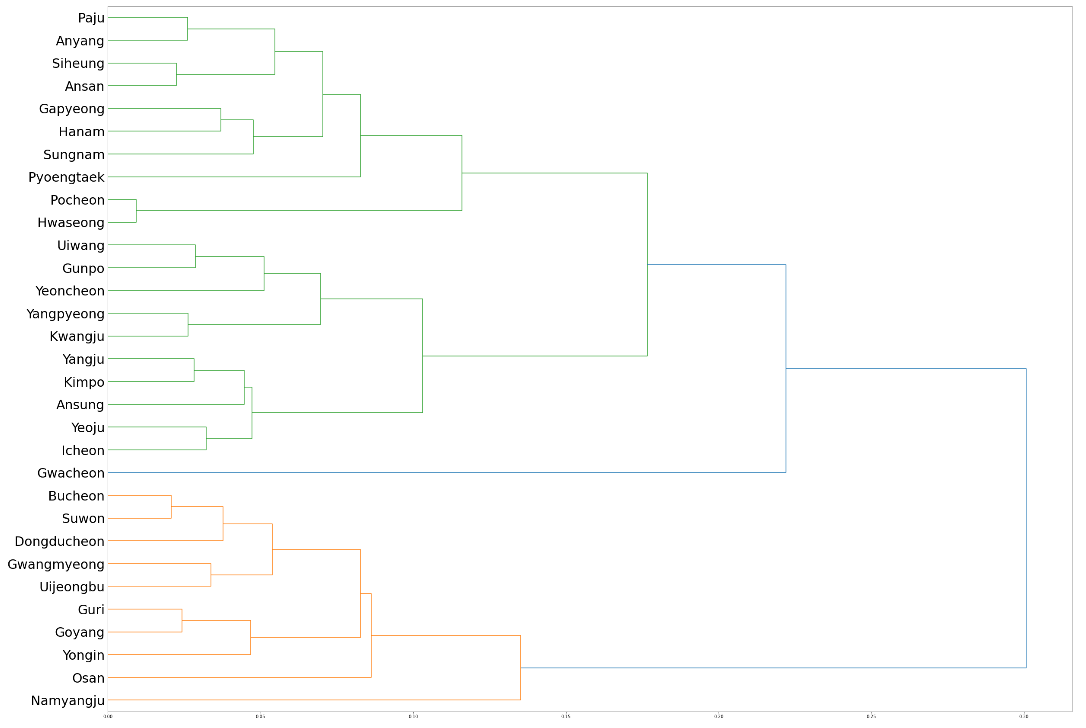
**4. 위의 방법을 적용하여 얻은 결과**

 총목적도착을 포함한 퍼센티지 데이터 이용 ↓

 총목적도착을 포함하지 않은 퍼센티지 데이터 이용 ↓

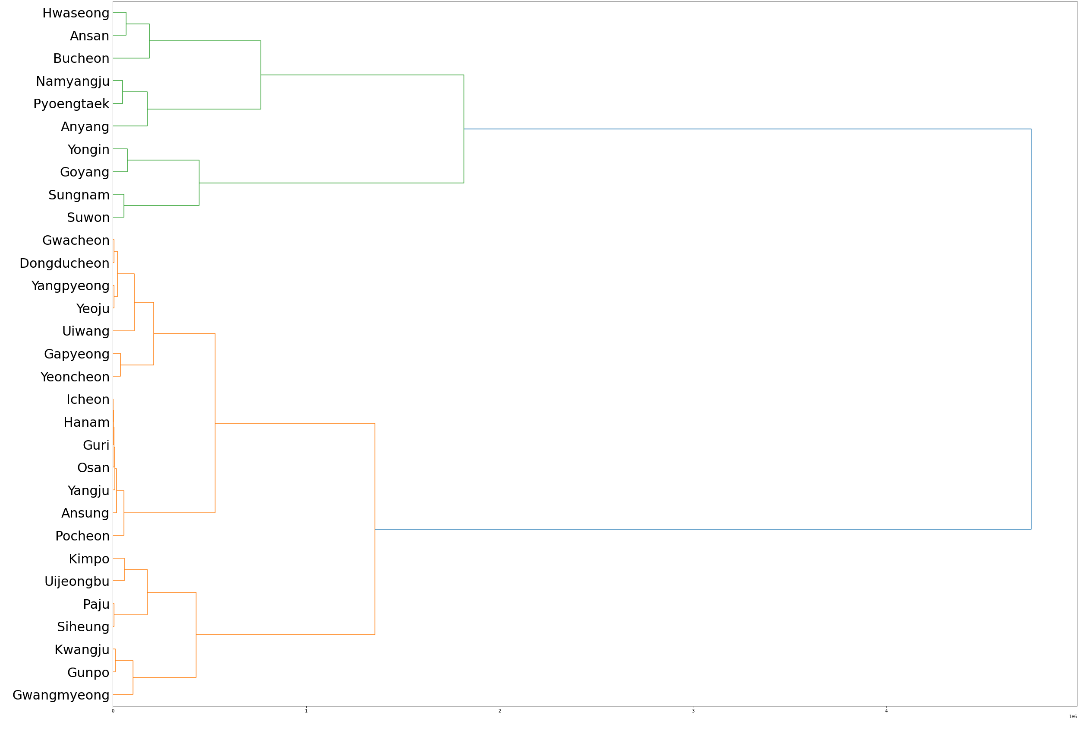
 퍼센티지를 사용하지 않은 데이터 이용 ↓

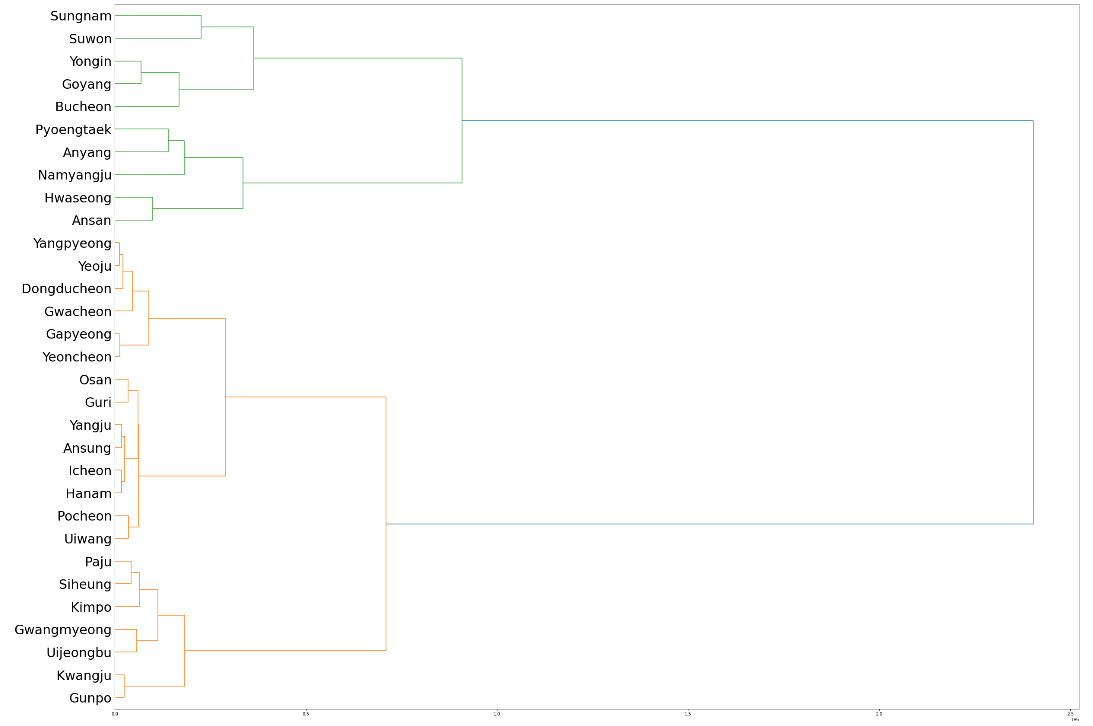
**5. 결과에 대한 해석**

가장 기대했던 데이터 분석 값은 총목적도착을 포함하지 않은 퍼센티지를 이용한 데이터 분석 값이었다. 이유는 총목적도착에 대한 비율로 나타낸다면 총목적도착량에 비례하는 이용 통계를 가질 수 있고 따라서 도시의 크기에 관계없이 그 도시가 가지는 주요 기능에 대해 분석할 수 있을 것 같았기 때문이다.

위는 분석한 데이터를 적당한 크기로 나누어 본 것이다. 6개정도의 클러스터로 잘라보았다. 기타를 제외한 7개의열 값이 입력 데이터에 존재했기 때문에 그것을 근거로 6~7개의 클러스터로 나누어 보았다. 결과값을 보면 과천 데이터의 값만 동떨어져 있는 것을 볼 수 있는데 이는 과천 데이터의 값이 다른 데이터에 비해 귀가도착의 비율이 낮기 때문이라고 판단되며 총목적도착을 이용하지 않는 것은 도시의 크기를 고려하지 않은 데이터의 도출이라고 판단된다.

다음은 퍼센티지에 총목적도착을 포함한 데이터에 대한 분석 결과이다.

이번에는 6개의 cluster로 나누어 보았다. 이번 데이터를 분석해본 결과 총목적도착에 지나치게 의존적인 데이터를 보여주었다.

마지막 데이터인 퍼센티지를 사용하지 않은 데이터의 결과이다.

이번에는 7개의 Cluster로 분리해보았다. 총목적도착 통계량에 따라 분명히 달라지나 이전의 퍼센티지를 사용한 총목적도착 통계량에 따른 유사성을 크게 보여주지 않는다. 이 클러스터를 지도에서 나타내 보았다. 각 지역들을 살펴보면 가까운 거리에 존재하는 도시가 많이 보이는 것을 볼 수 있다. 수원-성남이나 김포-시흥-파주-광명-군포, 양평-여주 등 비슷한 위치에 있는 근접 도시들이 같은 Cluster안에 분류된 것을 확인할 수 있다.

이로부터 우리는 각 지역들의 구체적인 특징을 잡지는 못했으나 지역의 위치에 따라서 교통 통계량이 유사성을 보이고 비슷한 위치에 있는 도시들은 비슷한 특징을 갖는 다는 것을 알 수 있다.

**6. 코드**

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

import seaborn as sn

traffic = pd.read\_csv('traffic.csv')

traffic\_percent = pd.read\_csv('traffic.csv')

traffic\_percent["출근도착"] = traffic\_percent["출근도착"].div(traffic\_percent["총목적도착"])

traffic\_percent["등교도착"] = traffic\_percent["등교도착"].div(traffic\_percent["총목적도착"])

traffic\_percent["학원도착"] = traffic\_percent["학원도착"].div(traffic\_percent["총목적도착"])

traffic\_percent["업무도착"] = traffic\_percent["업무도착"].div(traffic\_percent["총목적도착"])

traffic\_percent["쇼핑도착"] = traffic\_percent["쇼핑도착"].div(traffic\_percent["총목적도착"])

traffic\_percent["여가오락친교도착"] = traffic\_percent["여가오락친교도착"].div(traffic\_percent["총목적도착"])

traffic\_percent["귀가도착"] = traffic\_percent["귀가도착"].div(traffic\_percent["총목적도착"])

traffic\_percent["기타도착"] = traffic\_percent["기타도착"].div(traffic\_percent["총목적도착"])

#tdf1 : 총목적도착 + percentage, tdf2 : percentage only, tdf3 : no percentage

tdf1 = traffic\_percent[["출근도착", "등교도착", "학원도착", "업무도착", "쇼핑도착", "여가오락친교도착", "귀가도착", "총목적도착"]]

tdf2 = traffic\_percent[["출근도착", "등교도착", "학원도착", "업무도착", "쇼핑도착", "여가오락친교도착", "귀가도착"]]

tdf3 = traffic[["출근도착", "등교도착", "학원도착", "업무도착", "쇼핑도착", "여가오락친교도착", "귀가도착"]]

#SciPy에서 한글 지원을 하지 않는 관계로 영문명 리스트 입력

labelList=['Suwon', 'Sungnam', 'Uijeongbu', 'Anyang', 'Bucheon', 'Gwangmyeong', 'Pyoengtaek', 'Dongducheon',

'Ansan', 'Goyang', 'Gwacheon', 'Guri', 'Namyangju', 'Osan', 'Siheung', 'Gunpo', 'Uiwang', 'Hanam',

'Yongin', 'Paju', 'Icheon', 'Ansung', 'Kimpo', 'Hwaseong', 'Kwangju', 'Yangju', 'Pocheon', 'Yeoju',

'Yeoncheon', 'Gapyeong', 'Yangpyeong']

from scipy.cluster.hierarchy import linkage, dendrogram

cluster1 = linkage(tdf1, 'ward')

plt.figure(figsize=(40,30))

dendrogram(cluster1,

orientation='right',

labels = labelList,

leaf\_rotation=0,

leaf\_font\_size=30,

)

plt.show()

from scipy.cluster.hierarchy import linkage, dendrogram

cluster1 = linkage(tdf1, 'ward')

plt.figure(figsize=(40,30))

dendrogram(cluster1,

orientation='right',

labels = labelList,

leaf\_rotation=0,

leaf\_font\_size=30,

)

plt.show()

from scipy.cluster.hierarchy import linkage, dendrogram

cluster2 = linkage(tdf2, 'ward')

plt.figure(figsize=(40,30))

dendrogram(cluster2,

orientation='right',

labels = labelList,

leaf\_rotation=0,

leaf\_font\_size=30,

)

plt.show()

from scipy.cluster.hierarchy import linkage, dendrogram

cluster3 = linkage(tdf3, 'ward')

plt.figure(figsize=(40,30))

dendrogram(cluster3,

orientation='right',

labels = labelList,

leaf\_rotation=0,

leaf\_font\_size=30,

)

plt.show()