주성분/군집분석 과제

빅데이터통계기법 과제 03

2021.04.26(빅데이터통계기법)

202050135 우주지질학과 홍지민

PCA 실습

- Python에선 sklearn이란 라이브러리에서 PCA기법을 지원합니다.
- 다만, scatter_matrix나 로딩 벡터를 그리기 위해서 다른 라이브러리가 필요합니다.
- R에서 USArrests 데이터를 csv파일로 다운 받고 분석을 진행 했습니다.

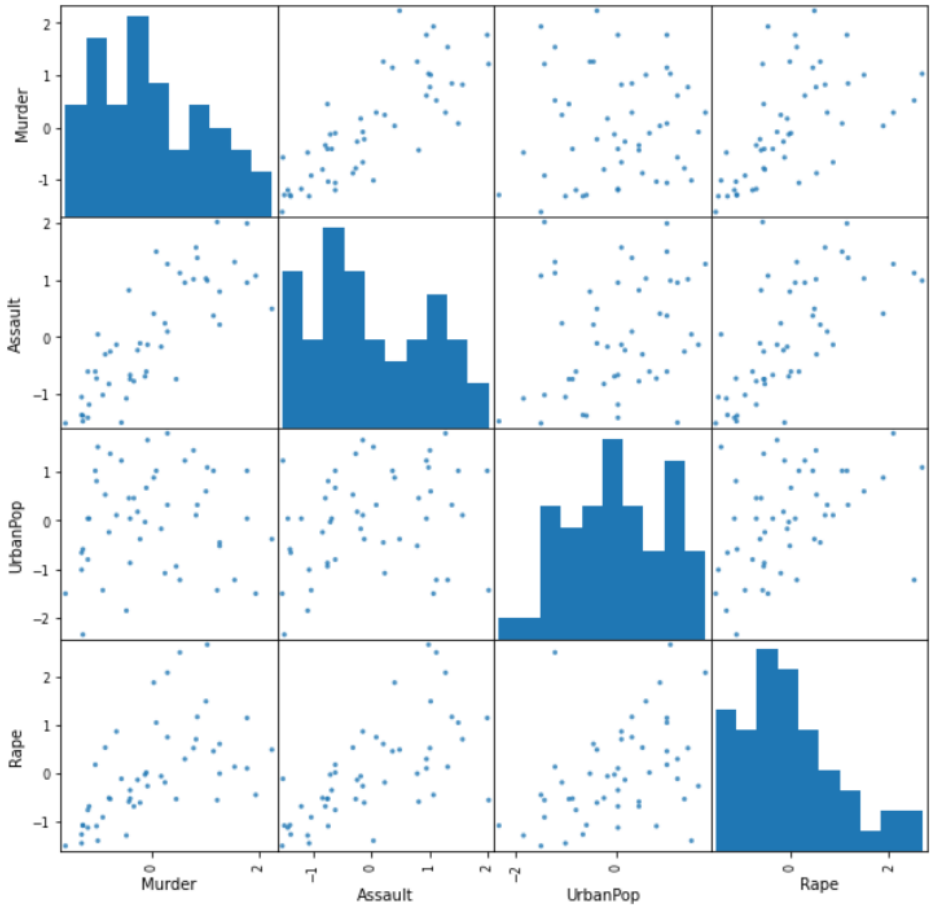
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from pandas.plotting import scatter_matrix
```

- 수학을 쉽게 다루는 라이브러리
- 데이터프레임을 다루는 라이브러리
- 그림을 그리는 라이브러리
- PCA 및 정규화 라이브러리
- scatter_matrix를 생성하기 위한 라이브러리

PCA 실습

```
raw = pd.read_csv('USArrests.csv',index_col=[0])

standard_data = StandardScaler().fit_transform(raw.values)
data = pd.DataFrame(standard_data,columns=raw.columns,index=raw.index)

scatter_matrix(data,marker='o',s=10,alpha=.8,figsize=(10,10))
plt.show()
```

- R에서 다운한 데이터 파일 변수 지정
- 데이터 표준화(정규화)
- 다시 데이터 프레임으로 만들고 plot



- UrbanPop를 제외한 나머지 3개의 변수에서 양의 상관관계가 보입니다.
- 특히, Murder와 Assault에 해당하는 변수는 강한 양의 상관관계로 보입니다.
- PCA를 실시한다면 하나의 성분에 Murder, Assault, Rape가 기여도가 높고, 다른 성분에 UrbanPop의 기여도가 높을 것 같다고 예상합니다.

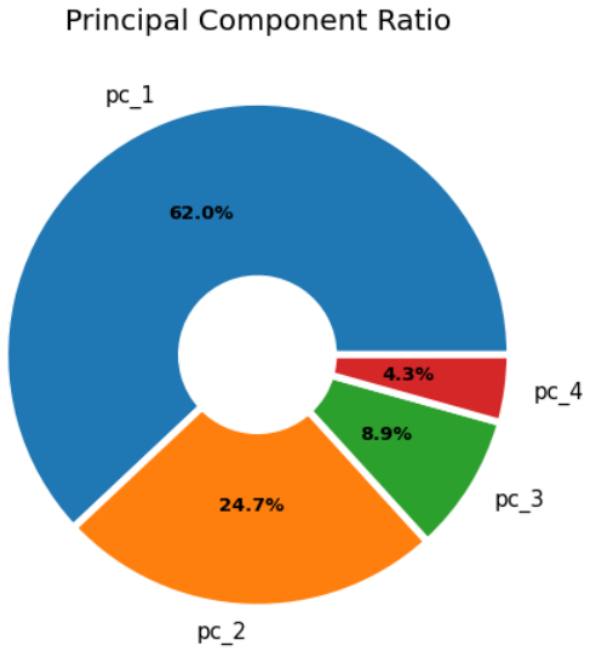
PCA 실습

```
pca = PCA()
column=['pc_1','pc_2','pc_3','pc_4']
coms = pca.fit_transform(standard_data)
pca_df = pd.DataFrame(coms,columns=column)
```

- PCA 실시 후 컬럼명 설정 및 데이터 프레임 설정

```
ratio = pca.explained_variance_ratio_
_,_,text = plt.pie(ratio,labels=column,autopct='%.1f%%',₩
        wedgeprops={'width': 0.7, 'edgecolor': 'w', 'linewidth': 5},textprops={'fontsize':15})
plt.setp(text, size=13, weight="bold")
plt.title("Principal Component Ratio")
plt.show()
```

- PIE 그래프를 통해 각 성분이 얼마나 많은 데이터를 설명하는지 분석
- 그래프 그리기



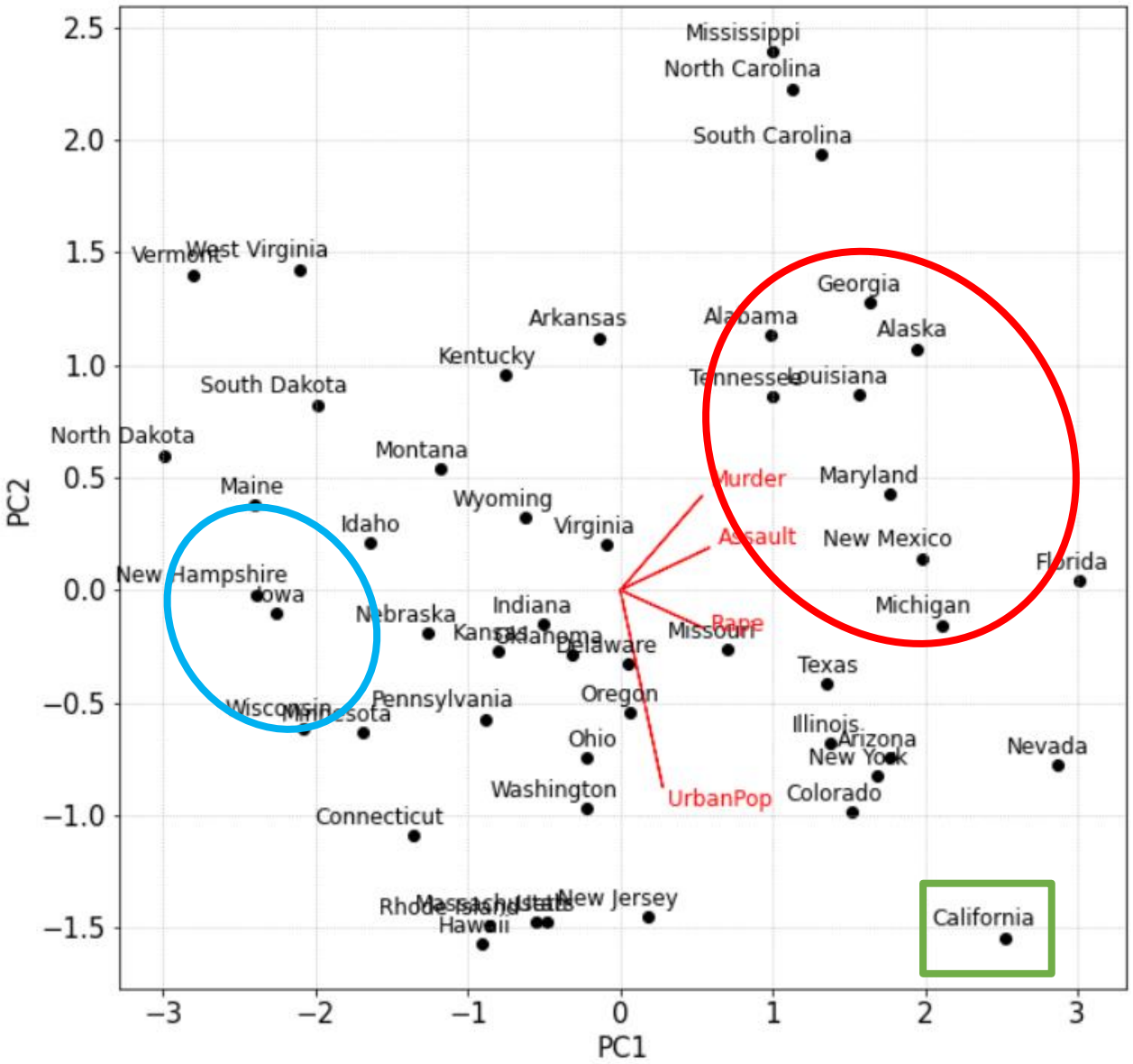
- 성분 1이 62%, 성분 2가 24.7%로 2차원 그래프로 84.7%의 데이터를 설명할 수 있습니다.
- 3차원 그래프를 그린다면 93.6%로 더욱 많은 데이터를 설명할 수 있을 것 같습니다.
- 다만, 한눈에 보기엔 2차원 그래프가 좋으므로 2개의 성분을 선택해 2차원 그래프를 그렸습니다.

PCA 실습

```
fig, ax = plt.subplots(figsize=(10,10))
ax.scatter(pca_df['pc_1'],pca_df['pc_2'],color='black')
for i, txt in enumerate(data.index):
    ax.annotate(txt, (pca_df['pc_1'][i]+0.2,pca_df['pc_2'][i]+0.05),₩
                size=10, xytext=(0,0), ha='right', textcoords='offset points',fontsize=12)
ax.set_xlabel("PC1",fontsize=15);ax.set_ylabel("PC2",fontsize=15)
ax.grid(linestyle=':')

x_vector = pca.components_[0]
y_vector = pca.components_[1]
for j in range(len(x_vector)):
  ax.arrow(0,0,x_vector[j],y_vector[j],color='r')
  ax.text(x_vector[j]*1.1,y_vector[j]*1.1,"{}".format(data.columns[j]),color='red',fontsize=12)
plt.setp([ax.get_xticklabels(),ax.get_yticklabels()],fontsize=15)
plt.show()
```

- 2개의 성분으로 나타낸 데이터를 산포도로 plot
- 도시 이름 plot 및 그래프 설정
- 성분 1,2의 로딩 벡터를 나타내기 위해서 선택
- 로딩 벡터 그린 후 각각의 벡터가 어떤 변수를 나타내는지 plot

PCA 실습



- 성분 1,2로 나타낸 산포도 그래프입니다.
- 예상대로 범죄에 해당하는 변수들이 성분 1에 많이 기여를 했습니다.
- 인구수는 성분 2에 기여도가 높습니다.
- 빨간 지역은 범죄가 많은 도시입니다.
- 파란 지역이 보다 안전한 도시라고 생각됩니다.
- 캘리포니아는 다른 범죄보다 강간과 관련된 범죄가 심한 도시입니다.

```
pca.components_
```

```
array([[ 0.53589947,  0.58318363,  0.27819087,  0.54343209],   pc 1의 기여도
       [ 0.41818087,  0.1879856 , -0.87280619, -0.16731864],   pc 2의 기여도
       [-0.34123273, -0.26814843, -0.37801579,  0.81777791],
       [ 0.6492278 , -0.74340748,  0.13387773,  0.08902432]])
```

군집분석 실습

- Python에선 sklearn에서 클러스터링 방법을 제공합니다..
- 다만, 하지만 덴드로그램을 그리기 위해 추가적인 라이브러리도 Import
- PCA 실습에서 이용한 데이터를 그대로 이용 했습니다.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage
```

- 거리군집, 밀집군집, 계층군집 라이브러리 import
- 덴드로그램을 위한 scipy 라이브러리 import

군집분석 실습

```
raw = pd.read_csv('USArrests.csv',index_col=[0])
pca_df = pd.read_csv('pca_df.csv',index_col=[0])

standard_data = StandardScaler().fit_transform(raw.values)
data = pd.DataFrame(standard_data,columns=raw.columns,index=raw.index)

kmeans = KMeans(n_clusters=3)
kmeans.fit(data)
```

- 앞에서 사용한 데이터 파일 변수 지정
- 데이터 표준화
- 거리기반 군집을 3개로 설정하고 fit

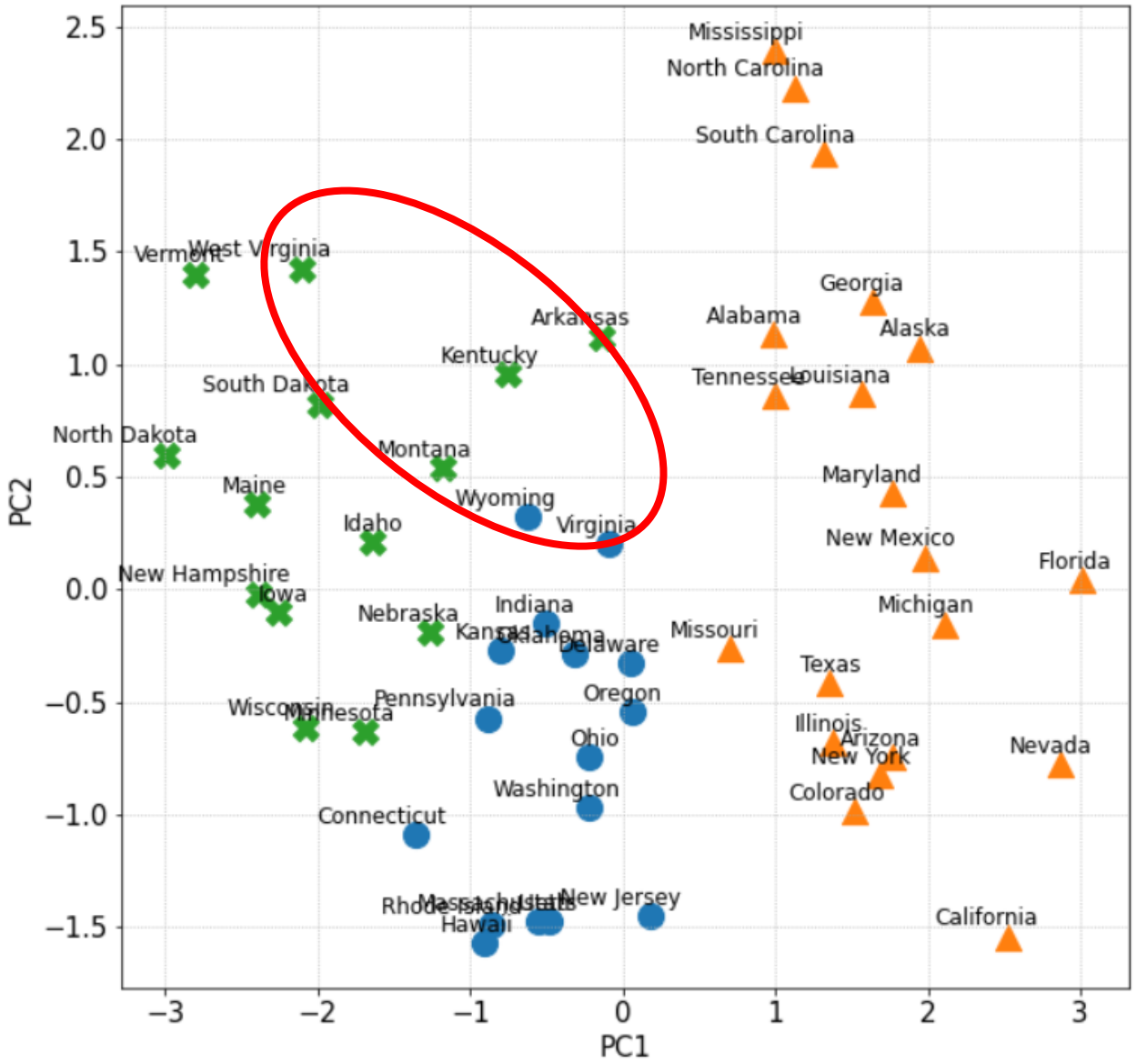
```
pca_df['cluster'] = kmeans.labels_

f1 = pca_df[pca_df['cluster'] == 0]
f2 = pca_df[pca_df['cluster'] == 1]
f3 = pca_df[pca_df['cluster'] == 2]
```

- 각각을 라벨링 후 군집에 따라 필터링

```
i = 0
for f in [f1,f2,f3]:
  if i == 0:
    s= 'o'
  elif i == 1:
    s = '^'
  elif i == 2:
    s = 'X'
  else:
    s = 'p'
  ax.scatter(f['pc_1'],f['pc_2'],marker=s,s=200)
```

- 반복문을 돌면서 산포도 plot
- 'i' 변수를 통해 군집마다 마커를 변경

군집분석 실습



- 거리기반 군집분석이므로 가까운 것이 군집화
- 노란색 삼각형은 살인, 강도, 강간의 범죄가 높은 지역을 뜻합니다.
- 파란색 동그라미는 살인, 강도의 범죄는 낮은 지역이지만 강간은 상대적으로 높은 지수를 나타내고 있습니다.
- 초록 엑스는 다른 지역에 비해서 범죄율이 낮은 도시를 나타냅니다.
- 빨간색 동그라미 지역은 살인, 강도가 높은 도시임에도 분류가 제대로 되지 않은 것으로 생각됩니다. (로딩 벡터상에서)

군집분석 실습

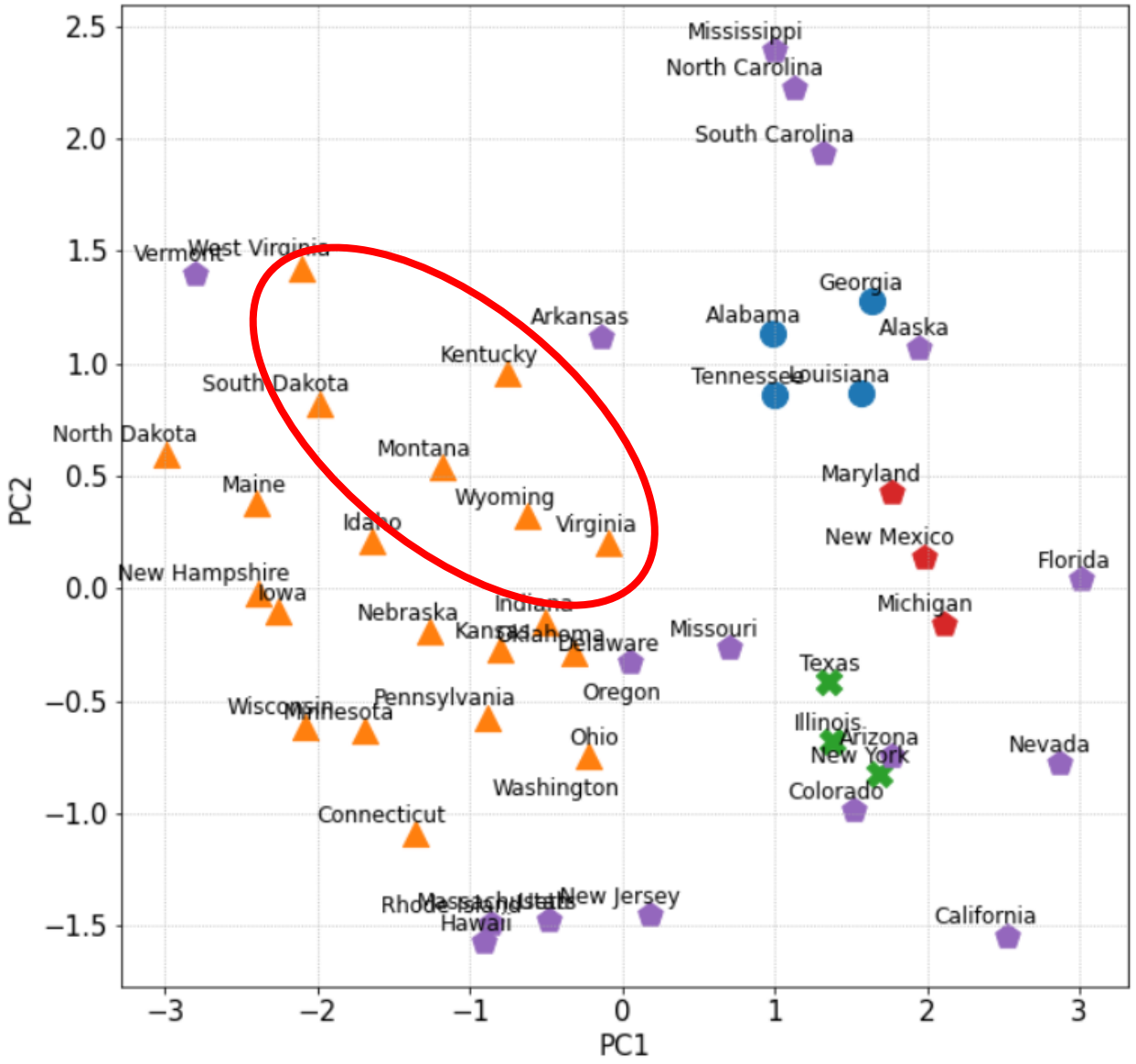
```
dbscan = DBSCAN(eps=0.9,min_samples=3)
pca_df['dbs_cluster'] = dbscan.fit_predict(data)
```

- eps는 입실론 반경을 가지는 원형의 영역입니다. (이 영역내에 최소 데이터의 수를 충족)
- min_samples는 위의 최소 데이터의 수에 해당

```
c1 = pca_df[pca_df['dbs_cluster'] == 0]
c2 = pca_df[pca_df['dbs_cluster'] == 1]
c3 = pca_df[pca_df['dbs_cluster'] == 2]
c4 = pca_df[pca_df['dbs_cluster'] == 3]
c5 = pca_df[pca_df['dbs_cluster'] == -1]
```

- 각각을 라벨링 후 군집에 따라 필터링

군집분석 실습

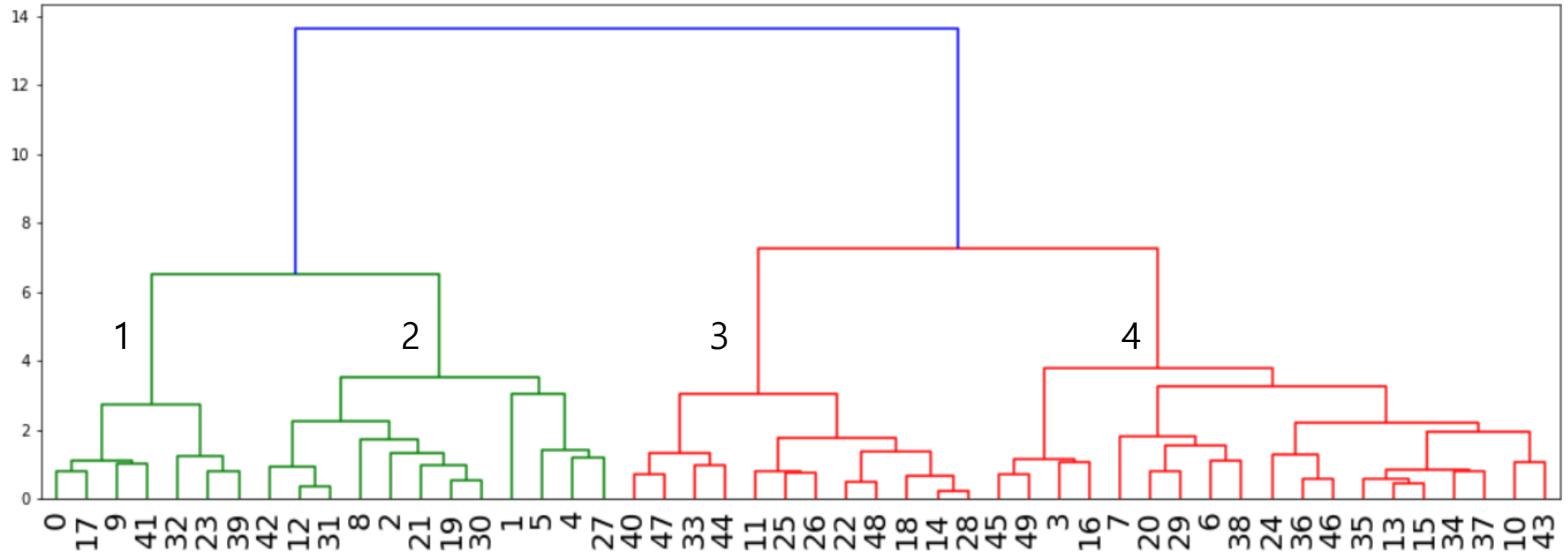


- 밀도기반 군집분석이므로 밀집된 것부터 군집화
- 우선 보라색 오각형은 아웃라이어로 알고리즘 상 이상치로 판단한 점들입니다.
- 파란색 동그라미는 살인, 강도의 범죄가 높은 도시입니다.
- 빨간 오각형은 전체적인 범죄가 많이 발생하는 도시로 판단합니다.
- 초록색 엑스는 특히 강간이 많이 발생하는 도시로 생각됩니다.
- 노란색 삼각형은 타도시에 비해서 범죄수가 낮은 도시로 생각됩니다.
- 역시나 빨간색 동그라미 지역은 살인, 강도가 높음에도 분류가 잘 되지 않습니다.

군집분석 실습

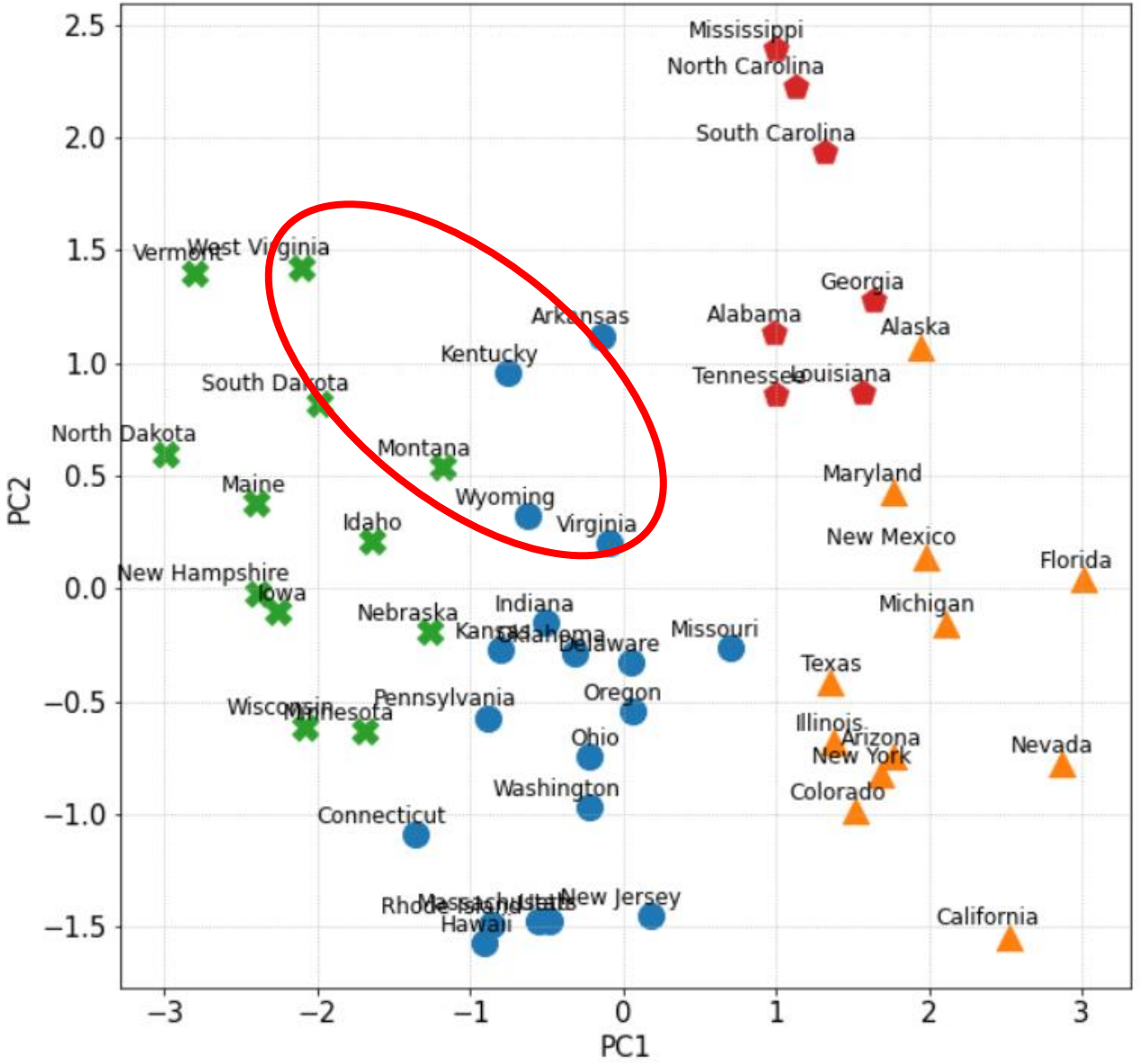
```
agg_cluster = linkage(data, method='ward')

plt.figure(figsize=(18,6))
dendrogram(agg_cluster,leaf_rotation=90,leaf_font_size=20)
plt.show()
```

- 계층분석 실시
- 덴드로그램 작성



- 크게 4개의 군집으로 나누는 것이 좋을 것 같다고 판단 했습니다.

군집분석 실습



- 계층기반 군집화이므로 서로의 군집을 만들며 군집화 했습니다.
  (metho가 여러가지가 있었습니다)
- 빨간색 오각형은 살인, 강도의 범죄가 많은 도시
- 노란색 삼각형은 전체적인 범죄가 많은 도시
  (특히, 강간이 많은 것 같습니다.)
- 파란색은 다른 범죄에 비해 강간이 높은 도시
- 녹색 엑스는 상대적으로 범죄가 없는 도시
- 여전히 빨간색 동그라미 지역에서 좋은 분류를 보여주지 못하고 있습니다.
  (PCA에서 손실된 정보의 이유...(?))
  - 다른 기법이 필요할 것 같습니다.
- 그나마 계층기반 군집화가 제일 좋은 성능을 보이는 것으로 생각됩니다.