활동 분석을 통한 비만 예방 프로젝트

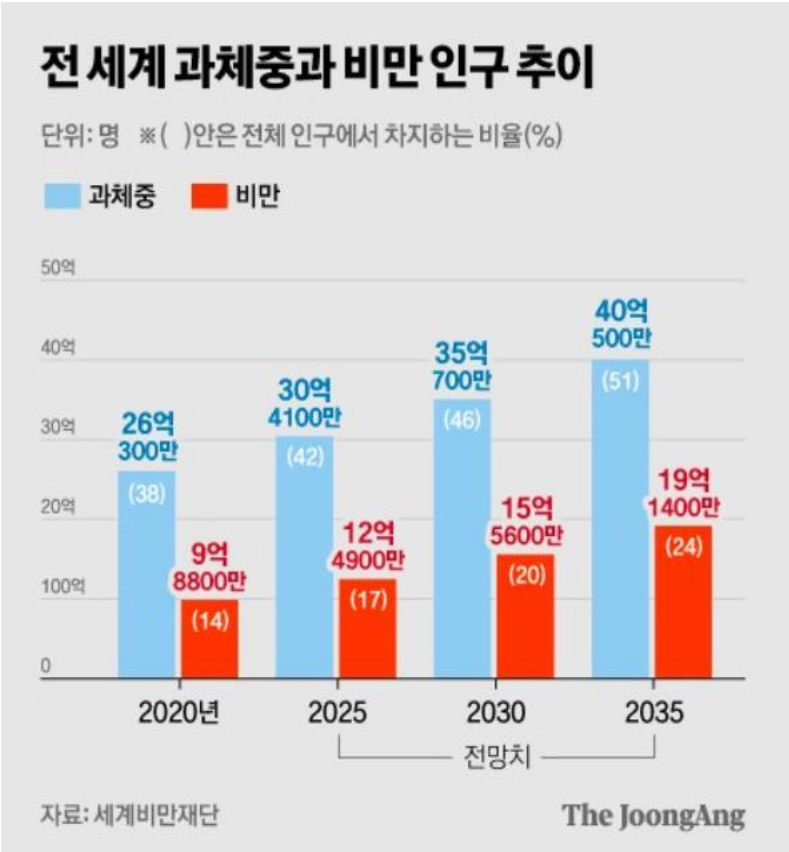
박누리

목차

프로젝트 주제 및 선정 배경, 기획의도

세계보건기구(WHO) 및 기타 보건 단체에 따르면, 전 세계적으로 비만 유병률이 계속해서 상승하고 있습니다. 2016년 WHO 보고서에 따르면 세계 성인 인구의 39%가 과체중이며, 그 중 13%가 비만 상태

WHO에 따르면 2019년에는 전 세계적으로 만 5세 이하의 4세 미만 어린이 중 4%가 비만 상태였으며, 어린이와 청소년 중 18세 미만의 약 1.6억명이 비만 상태



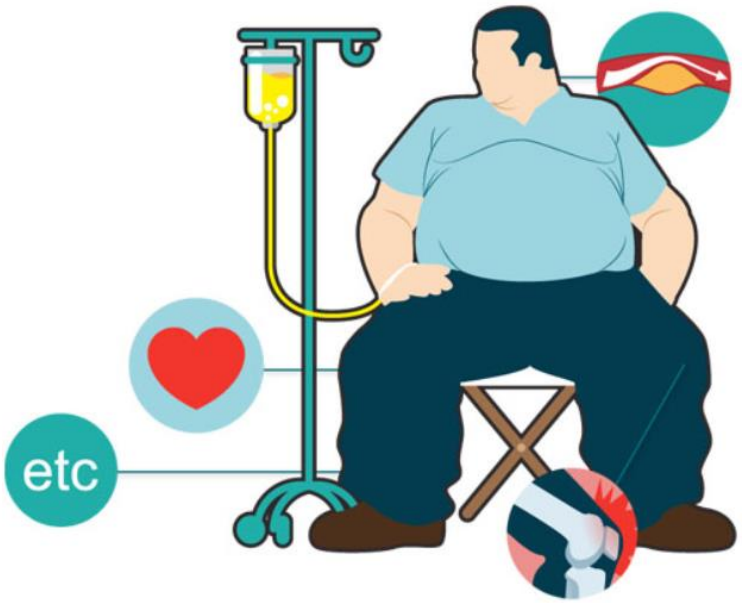
프로젝트 주제 및 선정
배경, 기획의도

비만은 2 형 당뇨병, 고혈압, 심혈관 질환, 관절염, 일부 종류의 암, 수면 무호흡 증후군, 간 질환 등을 유발하는 건강문제의 원인

의료비, 약비, 입원비, 휴가일수 감소 등으로 인해 비만은 경제적 부담을 초래

우울, 스트레스, 자존감 저하 및 정신적 건강 문제를 유발하여 개인의 심리적 안녕에 부정적인 영향을 미침

비만으로 인한 사회적 압박과 편견은 비만인 개인의 삶의 질을 저하시킬 수 있음



프로젝트 내용

- 프로젝트의 주된 목적은 스마트폰의 센서 데이터를 기반으로 인간의 다양한 일상 활동(ADL)을 자동으로 인식하는 시스템을 개발
- K-means clustering과 PCA(주성분 분석) 모델을 구현하여 인간의 활동 라벨을 분류하고 예측을 수행함

활용방안 및 기대효과

- 활용방안
  - 건강관리 및 모니터링 : 활동 패턴을 분석하여 신체 활동 부족, 수면 부족, 스트레스 수준 등을 파악
  - 운동 및 피트니스 애플리케이션 : 걸음 수, 칼로리 소모, 운동의 종류 및 강도를 모니터링하여 개인 맞춤형 운동 계획을 수립
  - 비만 관리 및 식습관 개선 : 사용자의 식습관 및 활동 패턴을 모니터링하여 건강한 라이프스타일 촉진
- 기대효과
  - 건강향상 : 비만 관리, 신체 활동 촉진, 올바른 운동 자세 유지 등의 효과
  - 사전 경고 및 응급 조치 : 사용자의 행동 패턴을 기반으로 사전 경고 및 응급 조치 시스템을 개발하여 안전을 강화
  - 사용자 경험 개선 : 개인 맞춤형 건강 정보 및 운동 조언을 받을 수 있음

프로젝트 절차 및 구조

활용 장비 및 재료

K-means clustering

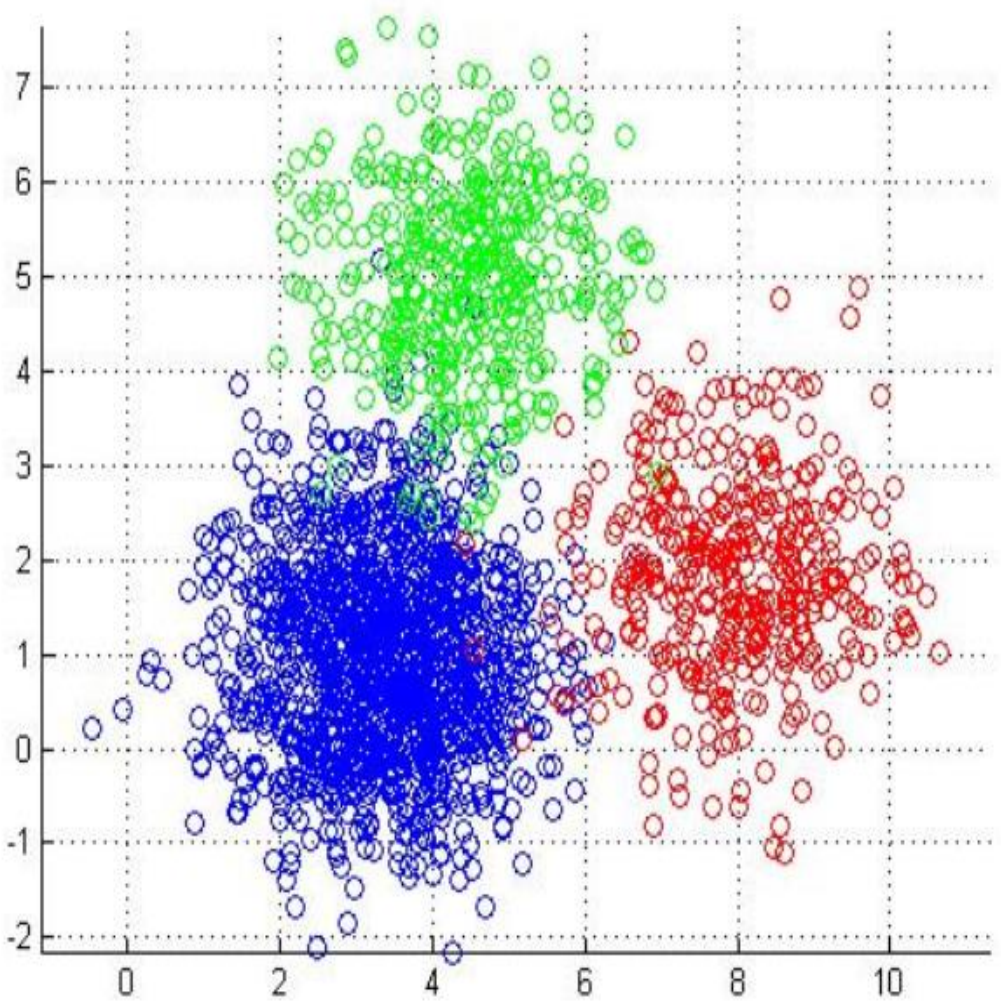
의미 : K-means Clustering의 목적은 유사한 데이터 포인트끼리 그루핑 하여 패턴을 찾아내는 것

Step 1 . 얼마나 많은 클러스터가 필요한지 K 결정 & 초기 Centroid 선택

Step 2.
모든 데이터를 순회하며 각 데이터마다 가장 가까운 Centroid가 속해있는 클러스터로 assign

Step 3 . Centroid를 클러스터의 중심으로 이동

Step 4 . 클러스터에 assign 되는 데이터가 없을 때까지 스텝 2, 3을 반복

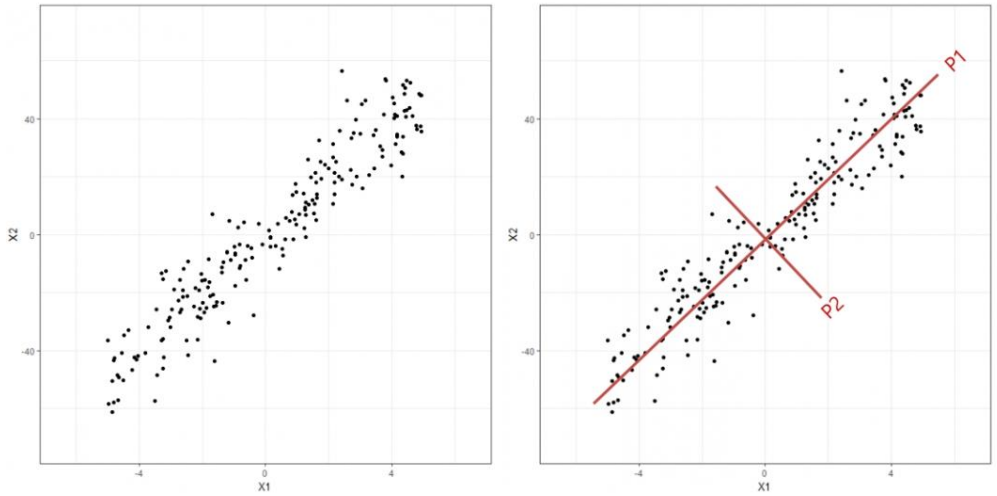


PCA(주성분 분석)

- 주성분 분석(Principal Component Analysis, PCA)은 가장 널리 사용되는 차원 축소 기법 중 하나로, 원 데이터의 분포를 최대한 보존하면서 고차원 공간의 데이터들을 저차원 공간으로 변환

- PCA는 기존의 변수를 조합하여 서로 연관성이 없는 새로운 변수, 즉 주성분(principal component, PC)들을 만들어 낸다

- 첫 번째 주성분 PC1이 원 데이터의 분포를 가장 많이 보존하고, 두 번째 주성분 PC2가 그 다음으로 원 데이터의 분포를 많이 보존하는 식



Why K-means clustering & PCA?

차원 감소 (PCA):

- 인간 자세 데이터는 보통 다양한 요소와 센서 데이터로 구성되며, 고차원 데이터일 수 있음. 고차원 데이터를 처리하면 연산 및 시각화가 어려울 수 있음.
- PCA는 데이터의 주성분을 추출하여 데이터의 차원을 줄일 수 있음. 일련의 주성분은 원래 데이터의 변동성을 최대한 보존하면서 차원을 감소시킴. 이를 통해 데이터를 시각화하거나 모델링에 활용하기 쉬워짐.

군집화 (K-means):

- 인간 자세 데이터를 군집화하면 유사한 자세를 가진 데이터를 같은 군집으로 묶어 분석할 수 있음.
- K-means 군집화는 데이터를 K개의 군집으로 나누는 비지도 학습 알고리즘으로, 유사한 데이터 포인트끼리 묶어주는 역할을 함.
- 이를 통해 특정 인간 자세의 군집을 찾아내거나 다양한 자세를 비교하는 데 유용함.

데이터 수집 및 라이브러리

	rn	activity	tBodyAcc.mean.X	tBodyAcc.mean.Y	tBodyAcc.mean.Z	tBodyAcc.std.X	tBodyAcc.std.Y	tBodyAcc.std.Z	tBodyAcc.mad.X	tBodyAcc.mad.Y	...	fBodyBodyGyroJerkMag.meanFreq	fBodyBodyGyroJer
0	7	STANDING	0.279	-0.0196	-0.1100	-0.997	-0.967	-0.983	-0.997	-0.966	...	0.146	
1	11	STANDING	0.277	-0.0127	-0.1030	-0.995	-0.973	-0.985	-0.996	-0.974	...	0.121	
2	14	STANDING	0.277	-0.0147	-0.1070	-0.999	-0.991	-0.993	-0.999	-0.991	...	0.740	
3	15	STANDING	0.298	0.0271	-0.0617	-0.989	-0.817	-0.902	-0.989	-0.794	...	0.131	
4	20	STANDING	0.276	-0.0170	-0.1110	-0.998	-0.991	-0.998	-0.998	-0.989	...	0.667	

5 rows × 563 columns

```
import time
t0 = time.time()

import seaborn as sns
import matplotlib.pyplot as plt
from tqdm import tqdm
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import random
from subprocess import check_output
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import KMeans
from sklearn.metrics import homogeneity_score, completeness_score, v_measure_score, adjusted_rand_score, adjusted_mutual_info_score, silhouette_score
```

```
dataset = pd.read_csv("train.csv")

dataset.head()
```

데이터 전처리 과정

```
['STANDING' 'SITTING' 'LAYING' 'WALKING' 'WALKING_DOWNSTAIRS'
 'WALKING_UPSTAIRS']
rn                                      0
activity                                0
tBodyAcc.mean.X                         0
tBodyAcc.mean.Y                         0
tBodyAcc.mean.Z                         0
                                       ..
angle.tBodyGyroMean.gravityMean         0
angle.tBodyGyroJerkMean.gravityMean     0
angle.X.gravityMean                     0
angle.Y.gravityMean                     0
angle.Z.gravityMean                     0
Length: 563, dtype: int64
```

```python
# 데이터 세트에서 특성을 추출하고 'rn' 및 'activity' 열을 제거하여 features를 만듭니다.
features=dataset.drop(['rn', 'activity'], axis = 1)
# 'activity' 열을 labels로 저장하여 데이터 포인트의 실제 레이블을 보유합니다.
labels=dataset['activity']

# 'activity' 열에서 고유한 레이블을 추출하여 Labels_keys에 저장하고, 이를 NumPy 배열로 변환하여 Labels를 만듭니다.
Labels_keys = labels.unique().tolist()
Labels = np.array(Labels_keys)
print(Labels)
# 각 레이블과 해당 인덱스를 매핑하는 딕셔너리 dict를 생성합니다.
dict = {}
for i in range(len(Labels)):
    dict[Labels[i]]=i
# 데이터 세트에서 누락된 값의 합계를 출력하여 데이터의 누락된 값 여부를 확인합니다.
print(dataset.isnull().sum())
```

모델링

- K-Means 클러스터링 알고리즘을 구현한 kmeans 함수를 정의합니다. 이 함수는 주어진 데이터와 클러스터 수에 따라 K-Means 클러스터링을 수행합니다.
- K-Means 알고리즘을 여러 번 반복하여 중심(centroid)을 업데이트하고, 클러스터에 데이터 포인트를 할당합니다. 이 과정은 수렴할 때까지 반복됩니다.
- K-Means 알고리즘은 클러스터링 결과로 각 데이터 포인트에 대한 클러스터 레이블을 반환합니다.

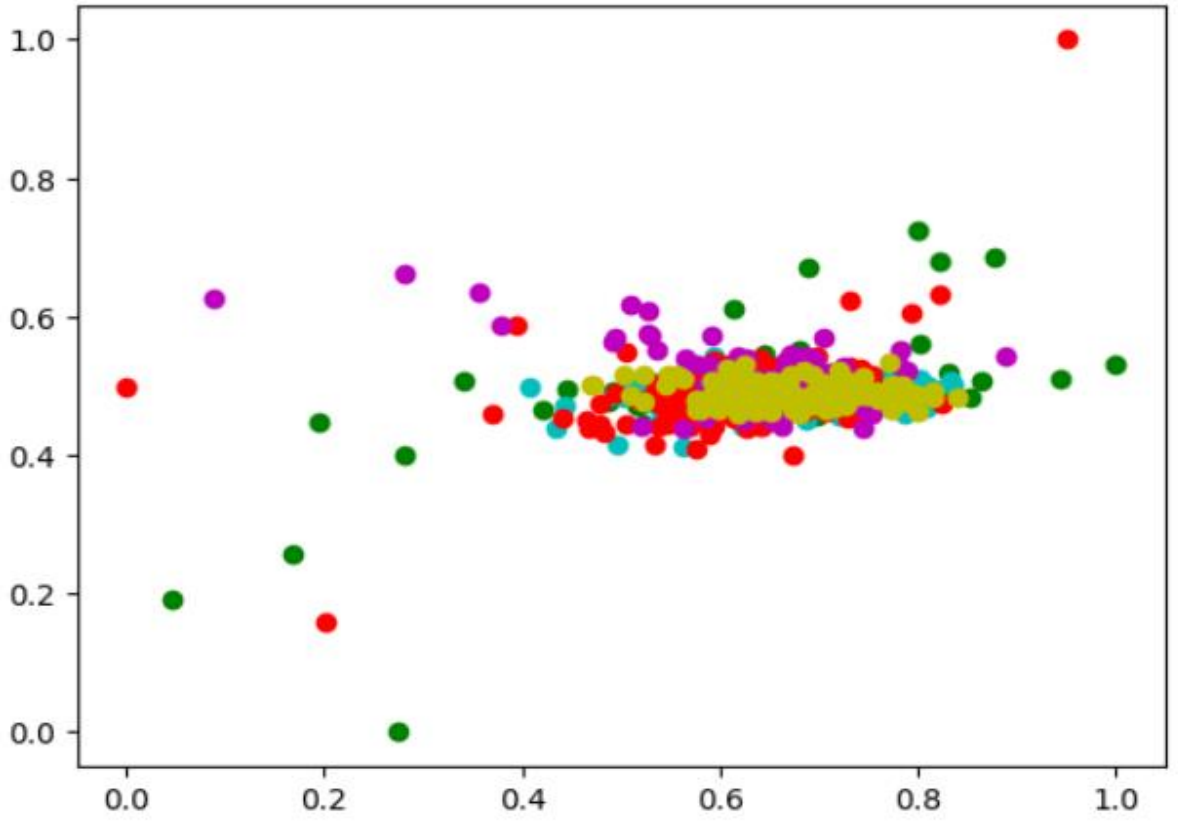
```
# K-Means 클러스터링 알고리즘을 구현한 kmeans 함수를 정의합니다. 이 함수는 주어진 데이터와 클러스터 수에 따라 K-Means 클러스터링을 수행합니다.
def kmeans(x,n_cluster):

    m=len(x[0])
    n=len(x)
    pred=np.zeros((n), dtype=int)

    kcase=0
    # K-Means 알고리즘을 여러 번 반복하여 중심(centroid)을 업데이트하고, 클러스터에 데이터 포인트를 할당합니다. 이 과정은 수렴할 때까지 반복됩니다.
    while(True):
        kcase+=1
        print("kcase =",kcase)
        if(kcase>1):
            centre = np.zeros((n_cluster,m) )
            n_points= np.zeros(n_cluster)
            for i in range(n):
                n_points[pred[i]]+=1
                centre[pred[i]]+=x[i]
            for i in range(n_cluster):
                if(n_points[i]>0):
                    centre[i]/=n_points[i]

                flag=False
```

```
            for i in range(n):
                d=[np.linalg.norm(x[i]-centre[j]) for j in range(n_cluster)]
                minj=np.argmin(d)
                if(pred[i]!=minj):
                    pred[i]=minj
                    flag=True
            if(flag==False):
                break
        else:
            centre = np.zeros((n_cluster,m) )

            for j in range(m):
                minj = min(x[:, j])
                maxj = max(x[:, j])
                rangej = float(maxj-minj)
                centre[: ,j]=minj+rangej*np.random.rand(n_cluster)

            for i in range(n):
                d=[np.linalg.norm(x[i]-centre[j]) for j in range(n_cluster)]
                minj=np.argmin(d)
                pred[i]=minj

    # K-Means 알고리즘은 클러스터링 결과로 각 데이터 포인트에 대한 클러스터 레이블을 반환합니다.
    return pred
```

클러스터링 결과 시각화

```
pred=kmeans(x,n_cluster)
for i in range(n_cluster):
    plt.scatter(x[np.where(pred==i)][:,0], x[np.where(pred==i)][:,1], c=colors[i])
```

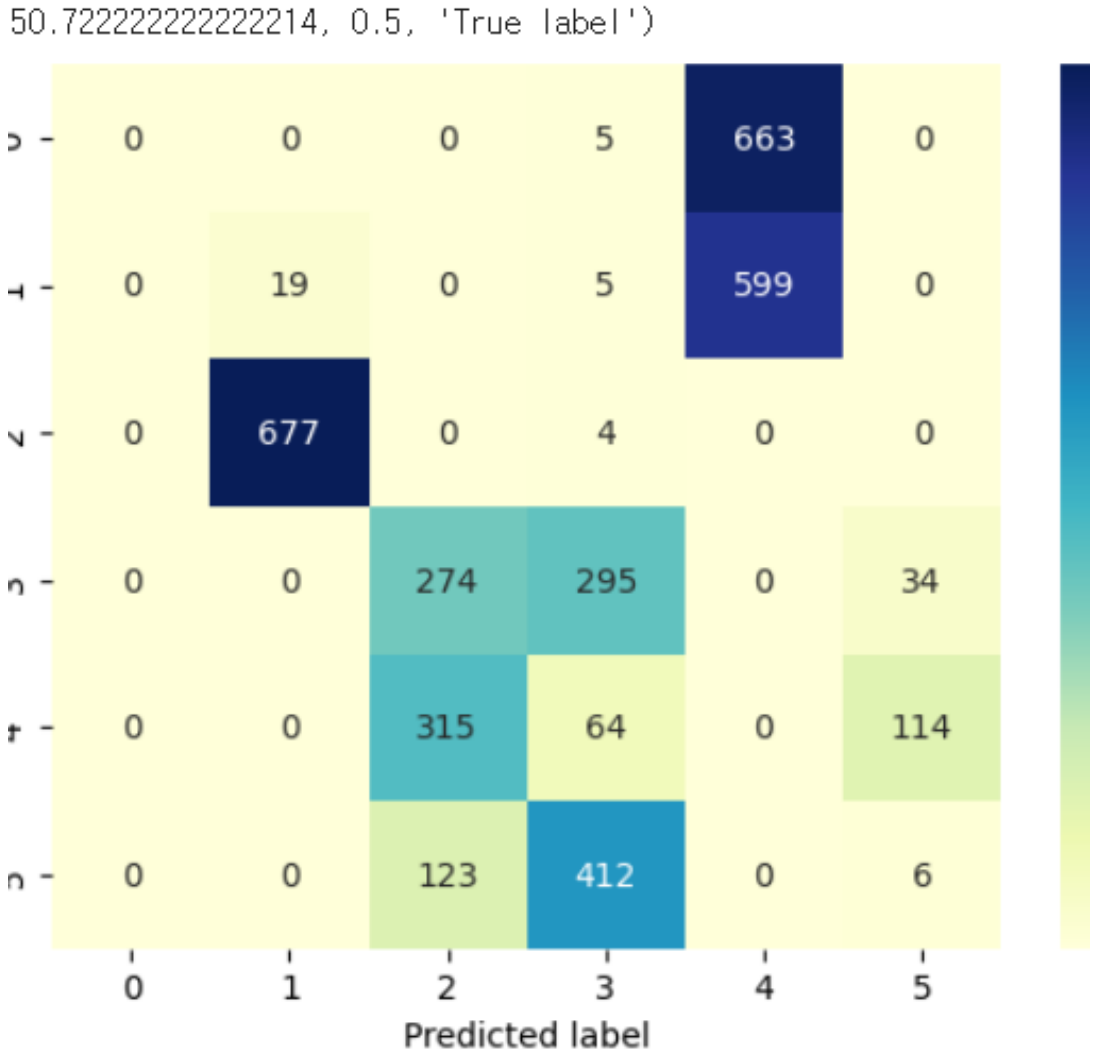
클러스터링 결과 시각화



모델 결과 시각화 및 평가

- 클러스터링 결과와 실제 레이블 간의 교차 테이블(crosstab)을 생성하고, 이를 hmap 배열에 저장하는 작업을 수행

- 클러스터와 실제 레이블 간의 crosstab 시각적으로 나타내는 작업



PCA 수행

```
def cov_mat(x):
    m = x.shape[0]
    x = x - np.mean(x, axis=0)
    return 1 / m * np.matmul(x.T, x)

def PCA(x, n_components):
    cov_matrix = cov_mat(x)
    eigval, eigvec = np.linalg.eig(cov_matrix)

    idx = eigval.argsort()[::-1]
    eigvec = eigvec[:, idx]
    eigvec = eigvec[:, :n_components]

    ans = np.matmul(x, eigvec)
    return ans
```

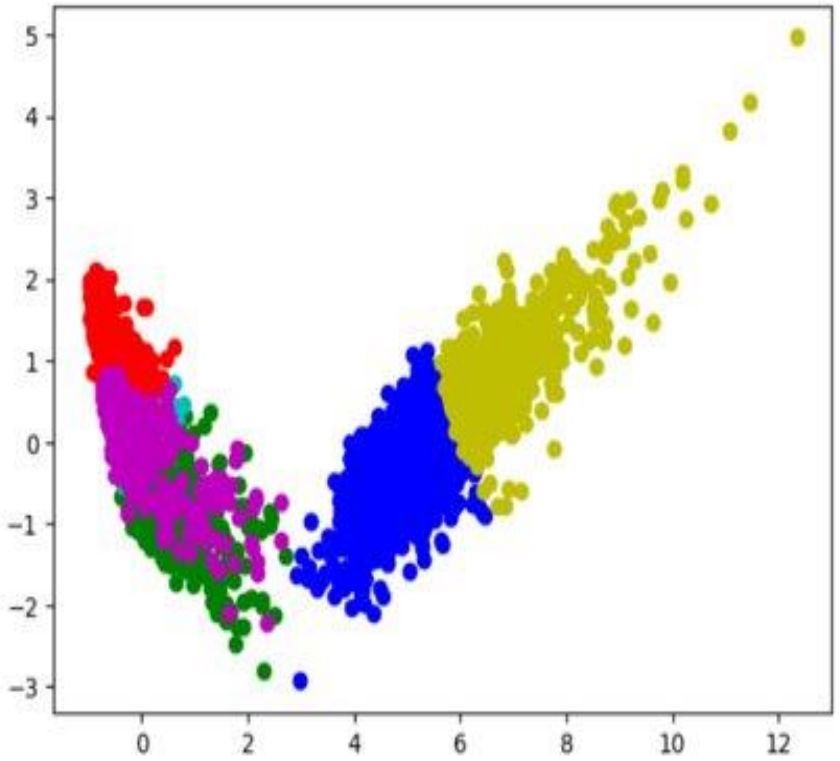
주성분 분석(PCA)을 수행하는 함수를 정의

```
x_pca = PCA(x, 10)
```

주어진 데이터 x(주성분 분석을 적용하려는 데이터)에 대해 PCA를 수행

```
pred_pca=kmeans(x_pca,n_cluster)
```

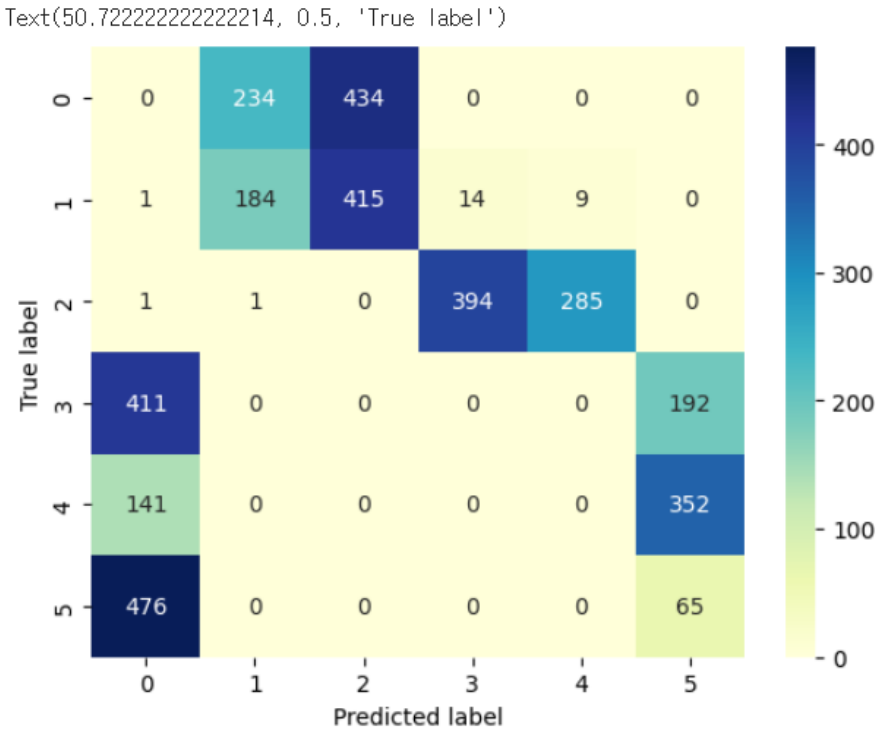
PCA로 변환된 데이터 x_pca에 K-Means 클러스터링을 적용한 결과를 저장



실제 레이블과 예측 레이블 간 교차표

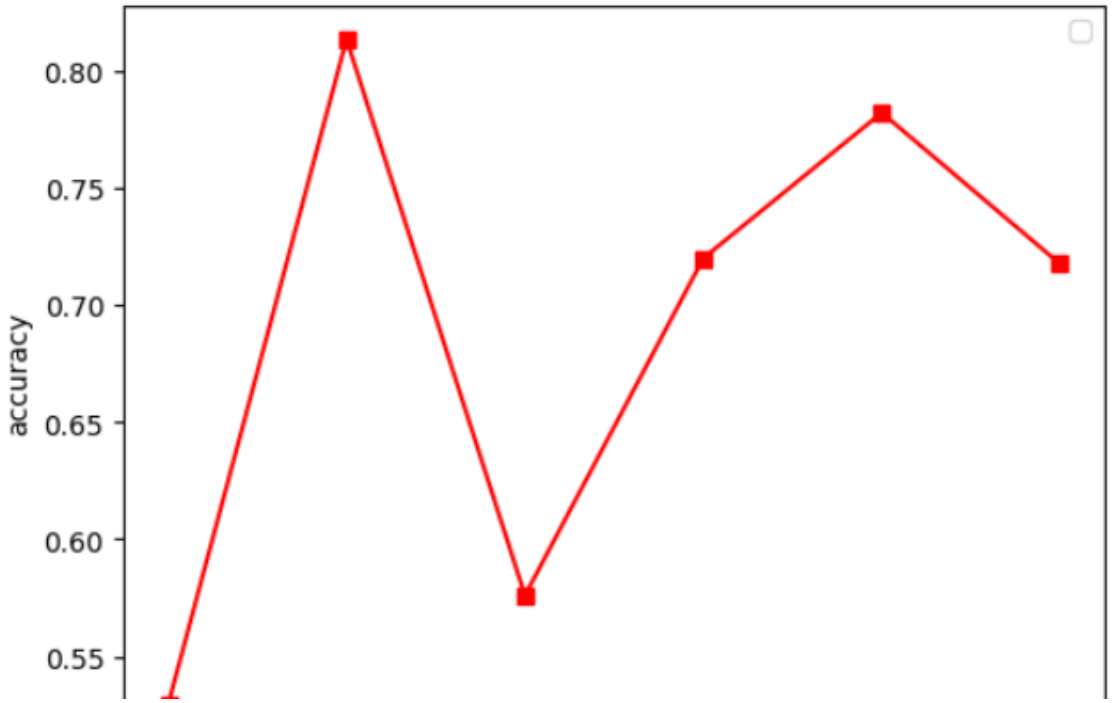
- 제 레이블(y)과 클러스터링 결과로 얻은 예측 레이블(pred_pca) 간의 교차표(crosstab)를 생성
- 교차표는 각 실제 레이블과 예측 레이블 간의 일치 및 불일치 수를 나타냄
- 교차표인 hmap을 열람하기 쉽게 시각화

```
[[  0. 234. 434.   0.   0.   0.]
 [  1. 184. 415.  14.   9.   0.]
 [  1.   1.   0. 394. 285.   0.]
 [411.   0.   0.   0.   0. 192.]
 [141.   0.   0.   0.   0. 352.]
 [476.   0.   0.   0.   0.  65.]]
```



정확도 평가

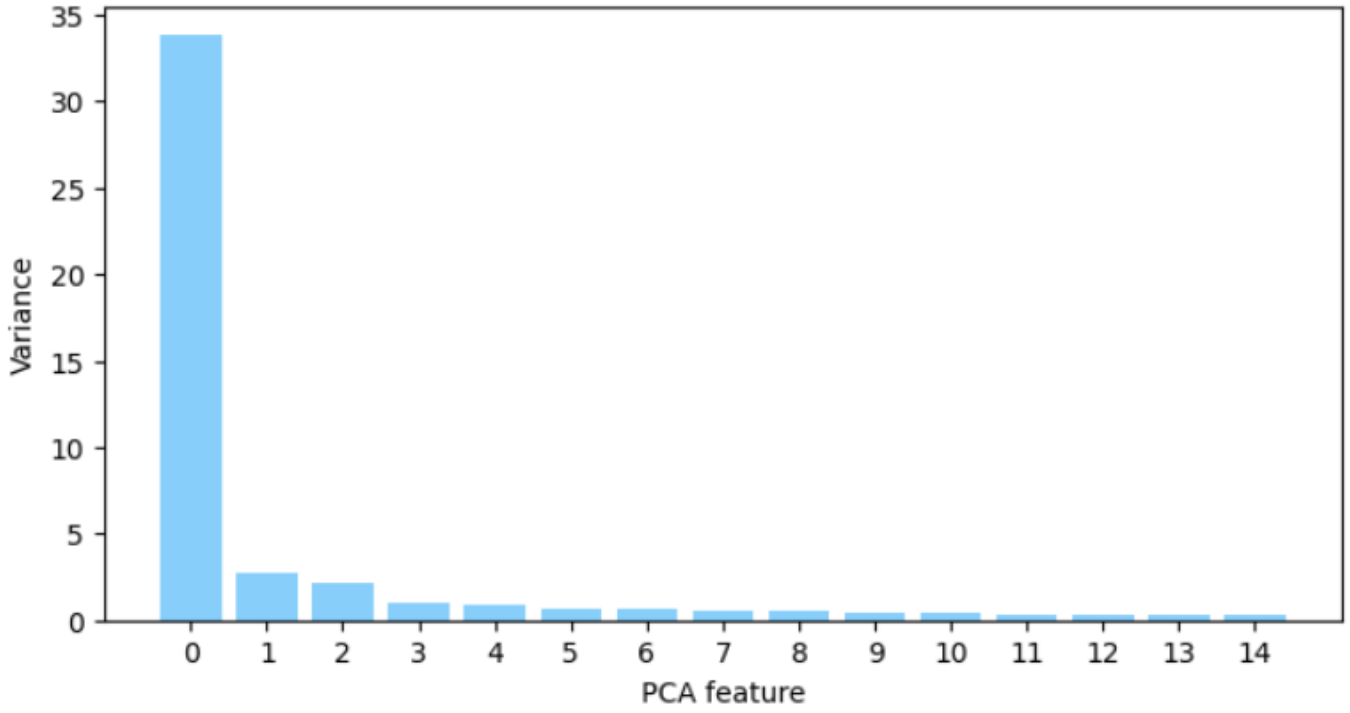
- 클러스터링 결과를 기반으로 각 클러스터의 정확도를 계산
- 클러스터 수 k에 따른 정확도를 선 그래프로 시각화




```
acc=np.zeros(n_cluster)
for _k in range(n_cluster):
    TN = TP = FN = FP = 0
    for i in range(n_cluster): #predict
        for j in range(n_cluster): #labels
            val=hmap[j][i]
            if (i==_k and j==_k): TN=TN+val
            if (i==_k and j!=_k): FN=FN+val
            if (i!=_k and j==_k): FP=FP+val
            if (i!=_k and j!=_k): TP=TP+val
    acc[_k] = (TP+TN)/(TP+TN+FP+FN)
```

클러스터링 성능 측정 지표 & 주성분 평가

- 클러스터링 성능을 측정하는 다양한 지표를 출력(지표에는 이너샤, 호모모지니어스성, 완결성, V-메져, ARI, AMI, 실루엣점수가 포함)
- 다른 방식으로 K- means 클러스터링 수행 후 주성분 분석 수행
- 그래프는 주성분 분석을 통해 어떤 주성분이 데이터의 분산을 가장 잘 설명하는지 확인하는 데 사용됨
- 그래프에서 설명된 분산이 가장 큰 주성분을 선택하여 데이터 차원을 줄이는 데 도움



클러스터링 성능 평가 및 출력

- 데이터 프레임을 주성분 개수가 1인 PCA로 축소하는 작업을 수행. 주성분 개수가 1인 경우, 데이터는 1차원 공간으로 축소되므로 K-평균 클러스터링은 이 1차원 데이터에서 두 개의 클러스터로 데이터를 분할하려고 시도할 것.

- 데이터 프레임을 주성분 개수가 2인 PCA로 축소하는 작업을 수행. 주성분 개수가 2인 경우, 데이터는 2차원 공간으로 축소되므로 K-평균 클러스터링은 이 2차원 데이터에서 두 개의 클러스터로 데이터를 분할하려고 시도할 것.

Shape of the new Data df: (3609, 1)

orig_label	0	1
clust_label		
0	1970	0
1	2	1637

inertia	homo	compl	v-meas	ARI	AMI	silhouette
11485	0.994	0.994	0.994	0.998	0.994	0.828

- 주성분 개수가 1인 경우

Shape of the new Data df: (3609, 2)

orig_label	0	1
clust_label		
0	1970	0
1	2	1637

inertia	homo	compl	v-meas	ARI	AMI	silhouette
21264	0.994	0.994	0.994	0.998	0.994	0.748

- 주성분 개수가 2인 경우

자체 평가

- **잘한 점**

(1) 다양한 접근 방식

(2) 시각화 자료를 많이 활용

- **보완해야 할 점**

(1) 시간 계획

(3) 타겟 구체화 부족