Capstone 2.
신체활동 추적 AI 솔루션

1조
이지호, 김●●, 강●●, 이●●

1. 문제범위 설정-4W frame

1 문제 범위 설정 — 2 데이터 획득 — 3 데이터 탐색 — 4 모델링 — 5 평가 — 6 배포

👤 Who (누구를 위한 프로젝트인가)

- 스마트폰 또는 웨어러블 기기를 사용하는 일반 사용자
- 특히 **일상 행동 변화가 건강·안전과 직결되는 사용자**

❓ Why (왜 이 프로젝트를 하는가)

- 스마트폰과 웨어러블 기기는 다양한 센서를 통해 사용자의 움직임을 수집
- 센서 데이터는 고차원·고복잡 구조로 그대로 사용하기 어려움
- 차원축소를 통해 핵심 정보만 남기면서도 행동 인식 성능을 유지할 수 있는지를 검증

📦 What (무엇을 해결하는가)

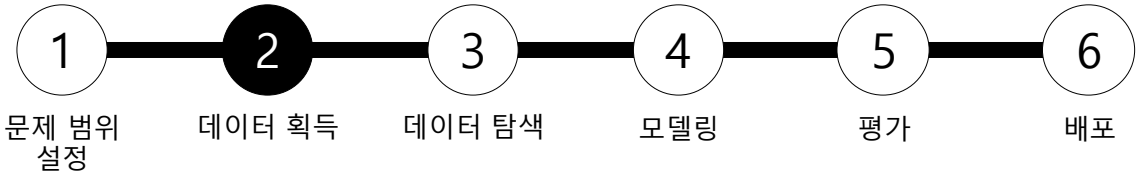
센서 데이터로부터 추출된 **561차원의 고차원 feature**를 대상으로 차원축소 기법을 적용하여,

- 데이터 복잡도를 줄이고
- 계산 효율을 개선하며
- 행동 분류 성능을 유지 또는 향상시키는 **경량 행동 인식 모델 가능성**을 탐구

🌍 Where (어디에 활용될 수 있는가)

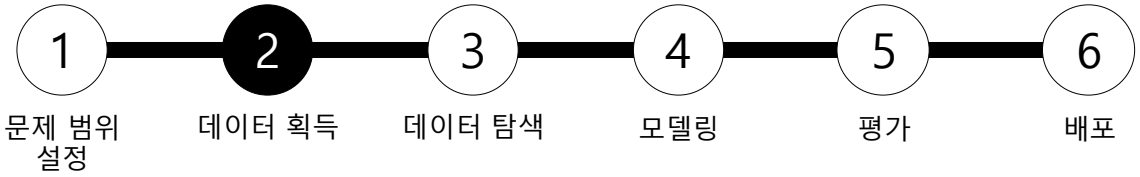
- 스마트폰·웨어러블 기반 행동 인식 시스템
- 낙상 감지, 이상 행동 탐지 등 **헬스케어·실버케어**
- 사용자 행동 기반 **스마트홈 자동화**
- 모바일 환경에서의 실시간 AI 서비스

2. 데이터 획득(스터디)



실험방법	19~48세 사이 성인 30명 대상, 허리 벨트에 갤럭시S2 고정하여 데이터 수집
수집데이터	가속도(accelerometer) – 3축 (X, Y, Z), 자이로스코프(gyro) – 3축 (X, Y, Z), 샘플링 속도: 50Hz (초당 50회 측정)
전처리 과정	활동영상 촬영 → 데이터를 2.56초 길이의 슬라이딩 윈도우로 자름 → 각 윈도우는 128개의 연속 측정값 포함 → 가속도에서 중력(gravity)와 신체 움직임(body motion)을 분리 → 각 윈도우에서 561개의 통계적 특징(feature) 을 추출
데이터 구성	샘플(record, 한 행 샘플) 포함 정보 : 정규화된 561개의 센서 기반 feature, 해당 샘플의 활동(Activity) 레이블, 샘플을 수집한 참여자의 ID(rn) → 즉, **한 사람(rn)이 특정 시간구간 (Window) 동안 한 행동(Activity)을 561개의 수치(feature)로 표현한 데이터**
6가지 활동 (class)	걷기 (WALKING), 계단 오르기 (WALKING_UPSTAIRS), 계단 내려가기 (WALKING_DOWNSTAIRS), 앉기 (SITTING), 서 있기 (STANDING), 누워 있기 (LAYING)

2-1. 팀원별 역할

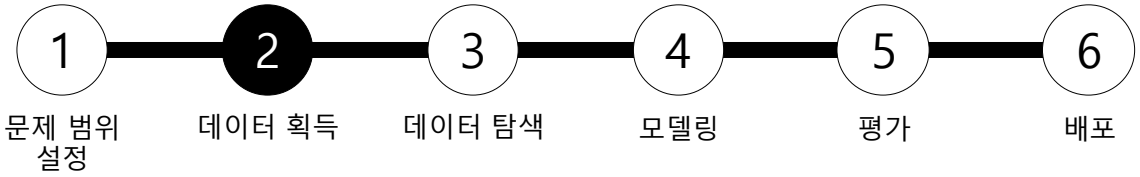


모든 실험 과정을 동일하게 진행하되 개인적으로 추가 실험한 부분이 있으면 이를 결과에 반영하는 방식

팀원명	공통	개인 역할
이지호	Elbow 통한 비지도 학습 통해 clustering Clustering 결과를 가지고 k-means 수행 PCA 차원축소 실험 및 시각화 t-SNE 시각화	프로젝트 매니징, 자료 취합, 의견 수렴, 비지도 학습 시 k=5일 경우의 특성 도출 및 시각화
김●●		고차원 PCA 수행을 통해 성능 확인
강●●		지도학습(activity 라벨)기반으로 비지도학습과의 성능 비교
이●●		비지도학습 k=6일 경우 특성 도출 및 시각화, 원본 데이터 vs PCA 이후 데이터 fitting 시간 차이 확인

1 → 프로젝트 요구사항 이행 표시

2-2. Timeline



12월 15일(월)		12월 16일(화)	
시간	**Task**	**시간**	**Task**
12:00 ~ 12:30	• 프로젝트 Overview	9:00 ~ 12:30	• PCA 차원축소 테스트 • k-means 와의 성능 비교 및 fitting 시간 비교
14:00 ~ 14:30	• 데이터셋 study • 4w 도출	14:00 ~ 15:00	• t-SNE 테스트 및 시각화
14:30 ~ 15:00	• 데이터 탐색 &전처리 • Elbow 통한 비지도 clustering	15:00 ~ 17:00	• 결과 취합 및 논의
15:00 ~ 16:00	• Elbow 기반 optima_k 도출 후 k-means 수행 • Clustering 기준 확인을 위해 도출된 cluster 별 특성 확인 • 각자 k값 별 테스트를 진행해보고 시사점 도출 • k=5, k=6 의 차이점을 inertia 지표들을 통해 확인	12월 17일(수)	
		시간	**Task**
		9:00 ~ 10:00	• 발표 전 프로젝트 최종 점검

2-3. 사용한 Tool과 Python Library

- 사용한 Tool

시각화, 데이터 분석 : Notebook(Jupyter, Colab 등)

- 사용한 Python Library

🐼 **pandas**: 표 형태 데이터(DataFrame) 생성·전처리·분석
🔢 **NumPy**: 다차원 배열 기반 수치 연산 처리
⚖️ **StandardScaler**: 평균 0·분산 1 기준 피처 스케일링
🎯 **KMeans**: 데이터를 K개의 군집으로 분할하는 비지도 학습 알고리즘
📊 **Silhouette Score**: 군집 간 분리도와 응집도를 통한 군집 품질 평가
📉 **PCA**: 주요 성분 기반으로 고차원 데이터를 저차원으로 축소
🧠 **t-SNE**: 고차원 데이터를 국소적 이웃 구조를 보존하며 2D·3D로 시각화
🧩 **Homogeneity Score**: 각 군집이 단일 클래스만 포함하는지 평가
🧷 **Completeness Score**: 각 클래스가 하나의 군집에 잘 모였는지 평가
🧿 **V-measure Score**: Homogeneity와 Completeness의 조화 평균 지표
🔁 **ARI**: 군집 결과와 실제 라벨 간 일치도를 보정하여 평가
🤝 **AMI**: 상호정보량 기반 군집 일치도를 보정하여 평가
🌲 **RandomForestClassifier**: 다수의 결정트리를 앙상블한 지도학습 분류 모델
✂️ **train_test_split**: 데이터셋을 학습·테스트 데이터로 분리
✅ **accuracy_score**: 전체 예측 중 정답 비율 계산
📝 **classification_report**: Precision·Recall·F1-score를 클래스별로 요약
🧱 **confusion_matrix**: 실제값과 예측값 분포를 행렬로 표현
📉 **matplotlib**: 기본 그래프 및 시각화 생성
🎨 **seaborn**: 통계 기반 고급 시각화 라이브러리

3. 데이터 탐색

1 문제 범위 설정 — 2 데이터 획득 — 3 데이터 탐색 — 4 모델링 — 5 평가 — 6 배포

train.csv 데이터는 총 563개의 열과 3,609개 행으로 이루어져 있음

1

```
df = pd.read_csv('train.csv')
```

```
df.head()
```

	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	...	fBodyBc
0	7	STANDING	0.279	-0.0196	-0.1100	-0.997	-0.967	-0.983	-0.997	-0.966	...	
1	11	STANDING	0.277	-0.0127	-0.1030	-0.995	-0.973	-0.985	-0.996	-0.974	...	
2	14	STANDING	0.277	-0.0147	-0.1070	-0.999	-0.991	-0.993	-0.999	-0.991	...	
3	15	STANDING	0.298	0.0271	-0.0617	-0.989	-0.817	-0.902	-0.989	-0.794	...	
4	20	STANDING	0.276	-0.0170	-0.1110	-0.998	-0.991	-0.998	-0.998	-0.989	...	

5 rows × 563 columns

```
df.shape
```

```
(3609, 563)
```

```
df.describe()
```

	rn	tBodyAcc.mean.X	tBodyAcc.mean.Y	tBodyAcc.mean.Z	tBodyAcc.std.X	tBodyAcc.std.Y	tBodyAcc.std.Z	tBodyAcc.mad.X	tBodyAcc.mad.Y	tBodyAcc
count	3609.000000	3609.000000	3609.000000	3609.000000	3609.000000	3609.000000	3609.000000	3609.000000	3609.000000	3609
mean	5152.430590	0.274544	-0.017415	-0.109195	-0.608457	-0.506265	-0.614482	-0.634634	-0.521660	-0
std	2975.767839	0.063589	0.042589	0.056218	0.439157	0.501627	0.399514	0.413194	0.485282	0
min	7.000000	-0.521000	-1.000000	-0.926000	-1.000000	-0.999000	-1.000000	-1.000000	-0.999000	-1
25%	2570.000000	0.262000	-0.025200	-0.122000	-0.992000	-0.976000	-0.979000	-0.993000	-0.976000	-0
50%	5158.000000	0.277000	-0.017200	-0.109000	-0.939000	-0.812000	-0.844000	-0.946000	-0.816000	-0
75%	7727.000000	0.287000	-0.011000	-0.098000	-0.254000	-0.051700	-0.283000	-0.306000	-0.084500	-0
max	10281.000000	0.693000	1.000000	1.000000	1.000000	0.980000	1.000000	1.000000	0.988000	1

8 rows × 562 columns

```
df.shape
```

```
(3609, 563)
```

563개의 열은 크게 rn(사용자 ID), activity(활동), 561개의 센서별 데이터로 나누어짐

```
missing = df.isna().sum()
missing_cols = missing[missing > 0].index.tolist()
print(missing_cols)
```

```
[]
```

2

별도의 결측치는 없어서 처리하지 않고 넘어감

3. 데이터 탐색

1 문제 범위 설정 — 2 데이터 획득 — 3 데이터 탐색 — 4 모델링 — 5 평가 — 6 배포

Activity가 나누어져 있지만, 비지도 학습을 통해 k-means를 위한 적정 분류 기준을 도출하기 위해 센서데이터 외의 열을 모두 제거하고 Elbow 기법을 수행하고 최적 k(optimal_k)를 5 혹은 6으로 설정해보기로 함

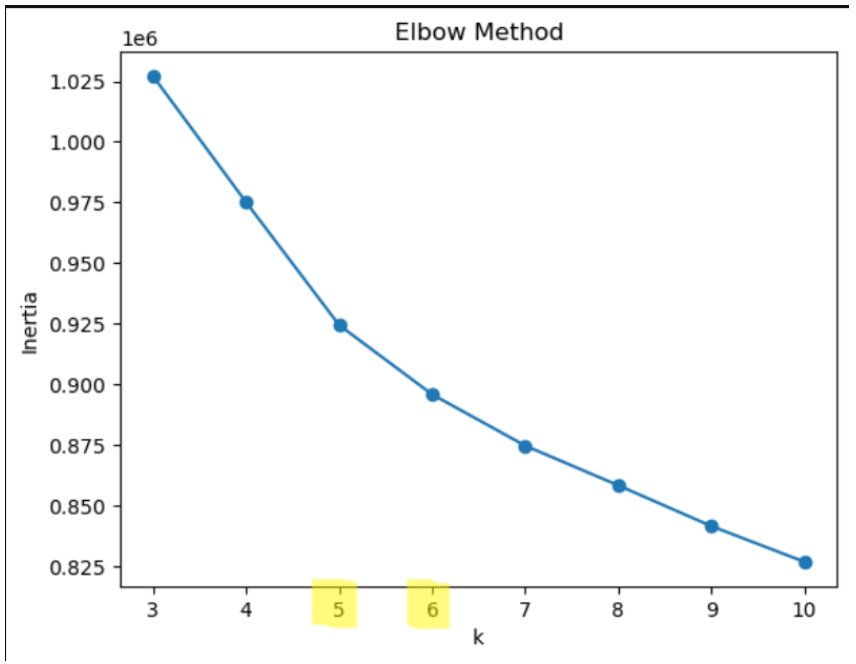
```python
# label 컬럼 제거
X = df.drop(columns=['activity','rn'])
```

3

```python
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X) # 정규화(스케일링) 수행
```

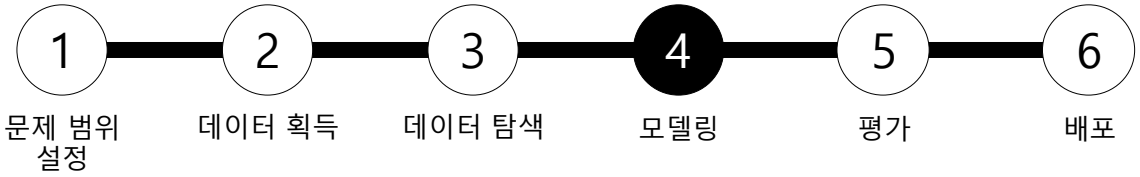
```python
# Elbow 수행 → 가장 '꺾이는 부분' 찾는 과정 ? 아니면 평평해지는 부분 찾는 과정? 논의 필요

inertias = []
K = range(3, 11)

for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    inertias.append(kmeans.inertia_)

plt.plot(K, inertias, marker='o')
plt.xlabel("k")
plt.ylabel("Inertia")
plt.title("Elbow Method")
plt.show()
```

5



⚠ 이 과정을 각자 실행해봤을 때 rn(사용자 ID) 열 제거 여부가 결과에 영향을 미침을 확인, 최종 논의 끝에 센서 데이터 외의 열은 모두 제거하는 것으로 합의

4. 모델링(k-means)



optimal-k를 5(혹은 6)으로 설정 후 k-means 를 수행, 분류된 cluster 별 특성을 찾기 위해 다양한 시도를 해봄

```
optimal_k = 5   # 꺾이는 부분을 찾아서 넣기 (이것이 우리가 판단하는 class의 숫자가 될 것임)

kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
clusters = kmeans.fit_predict(X_scaled)
```

4

```
df["cluster"] = clusters
print(df["cluster"].value_counts().sort_index())
```

```
cluster
0     718
1     659
2    1252
3     105
4     875
Name: count, dtype: int64
```

4. 모델링(k-means)

1 문제 범위 설정 — 2 데이터 획득 — 3 데이터 탐색 — 4 모델링 — 5 평가 — 6 배포

6

optimal-k를 5(혹은 6)으로 설정 후 k-means 를 수행, 분류된 cluster 별 특성을 찾기 위해 다양한 시도를 해봄

1. Top feature를 뽑아본다.

```
# 클러스터별 Top features를 뽑는다.

top_features = {}

for c in range(optimal_k):
    temp = cluster_z[cluster_z["cluster"] == c].drop(columns=["cluster"]).T
    temp.columns = ["z"]

    top_features[c] = temp.reindex(
        temp["z"].abs().sort_values(ascending=False).head(10).index
    )

top_features
```

```
{0:                                z
 tGravityAcc.arCoeff.Y.1    -1.096401
 tGravityAcc.arCoeff.Y.2     1.064288
 fBodyAccMag.meanFreq       -1.038569
 tGravityAcc.arCoeff.Y.3    -1.007262
 fBodyAccMag.skewness        0.957137
 tGravityAcc.arCoeff.Z.1    -0.949846
 tGravityAcc.arCoeff.Y.4     0.921645
 tGravityAcc.arCoeff.Z.2     0.911020
 fBodyBodyGyroMag.meanFreq  -0.906453
 tBodyGyroMag.arCoeff1      -0.903945,
 1:                                 z
 tBodyAccJerkMag.std         1.385165
 tBodyAccJerkMag.mad         1.383932
 fBodyBodyAccJerkMag.mean    1.380024
```

컬럼	의미
BodyAcc	신체 가속도(몸 움직임)
Gyro	회전/방향 변화
Jerk	갑작스러운 움직임
Mag	움직임의 크기
entropy	움직임의 불규칙성
energy	전체 활동 에너지

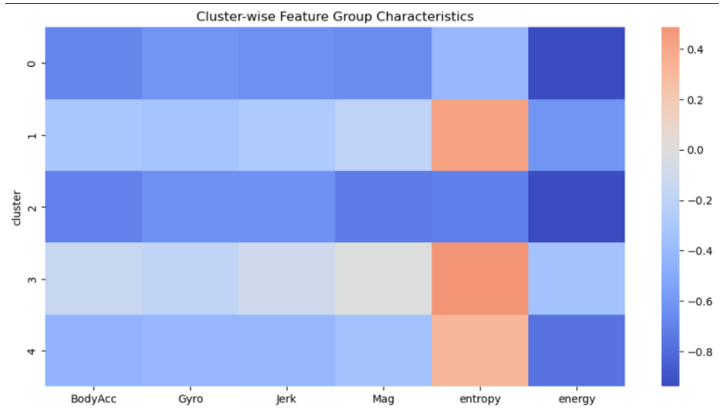
2. 그룹화를 해서 그룹별 수치를 본다.

```
# 그룹화를 해서 뭔가 의미있는걸 본다

group_summary = {}

for group in ["BodyAcc", "Gyro", "Jerk", "Mag", "entropy", "energy"]:
    cols = [c for c in X.columns if group in c]
    group_summary[group] = cluster_summary[cols].mean(axis=1)

group_df = pd.DataFrame(group_summary)
group_df["cluster"] = cluster_summary["cluster"]
group_df
```

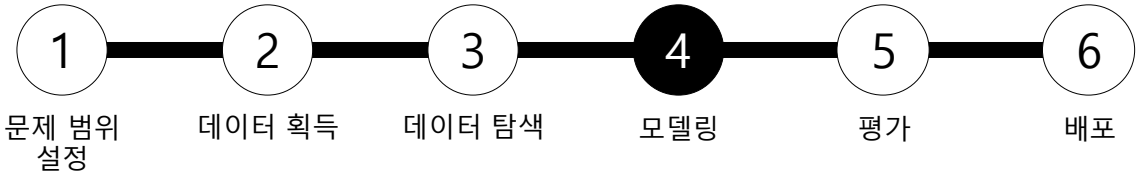
	BodyAcc	Gyro	Jerk	Mag	entropy	energy	cluster
0	-0.687136	-0.611410	-0.636337	-0.658987	-0.413031	-0.931327	0
1	-0.311002	-0.339539	-0.279911	-0.196753	0.426384	-0.615200	1
2	-0.709596	-0.638991	-0.638012	-0.724858	-0.714335	-0.939973	2
3	-0.148115	-0.183111	-0.103672	-0.007065	0.486828	-0.348515	3
4	-0.443829	-0.423999	-0.414432	-0.345435	0.326546	-0.762035	4

3. 시각화를 해본다.

```
# 시각화를 해서 본다

import seaborn as sns

plt.figure(figsize=(12,6))
sns.heatmap(
    group_df.set_index("cluster"),
    cmap="coolwarm",
    center=0
)
plt.title("Cluster-wise Feature Group Characteristics")
plt.show()
```



4. 모델링(k-means)



6

optimal-k를 6으로 설정 후 k-means 를 수행한 결과, cluster별로 다음과 같은 특성이 나타남

Cluster	움직임 크기 (BodyAcc·Gyro)	불규칙성 (Entropy)	Jerk 특성	중력 영향	주요 해석
0	낮음 ~ 중간	**높음**	변화 빈번	낮음	저강도이지만 전환·변화 잦은 동적 상태
1	매우 낮음	낮음	거의 없음	중간	안정적이고 변화 없는 정적 상태
2	낮음	**높음**	분산 큼 (std, mad)	낮음	리듬·보폭 변화가 큰 동적 활동
3	낮음	중간	미세 존재	중간	균형 유지를 위한 준정적 상태
4	**최저**	**최저**	없음	**매우 큼**	중력 기준 변화 없는 완전 정적 상태
5	**최저**	**최저**	없음	**매우 큼**	Cluster 4와 의미상 중복

k=6 클러스터링 결과, 동적 활동은 불규칙성과 jerk 기반으로 의미 있게 구분되었으나, 정적 활동 영역에서 Cluster 4와 5가 행동적으로 중복되어 과분할이 발생하였다고 해석함 → 실제로 가장 의미있는 k는 k=5 라고 결론내림

⚠ 최적 optimal-k를 6으로 설정했던 이유는, 동일 조건에서 각자 실행한 elbow에서 적정 k를 그래프가 완만해지는 수치인 6으로 보기로 합의하였기 때문

4. 모델링(k-means)

1 문제 범위 설정 — 2 데이터 획득 — 3 데이터 탐색 — 4 모델링 — 5 평가 — 6 배포

6 7

optimal-k 5와 6의 차이를 지표로 확인해 봄

```python
compare_df = pd.DataFrame({
    "k=5": {
        "inertia": kmeans_5.inertia_,
        "silhouette": silhouette_score(X_scaled, labels_k5)
    },
    "k=6": {
        "inertia": kmeans_6.inertia_,
        "silhouette": silhouette_score(X_scaled, labels_k6)
    }
})

# 구조 비교 지표 (k=5 기준 vs k=6)
structure_df = pd.DataFrame({
    "k=5 vs k=6": {
        "homogeneity": homogeneity_score(labels_k5, labels_k6),
        "completeness": completeness_score(labels_k5, labels_k6),
        "v_measure": v_measure_score(labels_k5, labels_k6),
        "ARI": adjusted_rand_score(labels_k5, labels_k6),
        "AMI": adjusted_mutual_info_score(labels_k5, labels_k6)
    }
})

print("=== 내부 지표 비교 (k=5 vs k=6) ===")
display(compare_df)

print("\n=== 군집 구조 유사도 지표 (k=5 기준 → k=6 비교) ===")
display(structure_df)
```

=== 내부 지표 비교 (k=5 vs k=6) ===

	k=5	k=6
inertia	924561.365400	895968.713509
silhouette	0.136163	0.112380

=== 군집 구조 유사도 지표 (k=5 기준 → k=6 비교) ===

	k=5 vs k=6
homogeneity	0.858887
completeness	0.747201
v_measure	0.799161
ARI	0.698129
AMI	0.798799

- k=5가 k=6이 될 경우 기존 k=5 군집 구조를 대부분 유지하면서 일부 클러스터를 추가로 분할
- homogeneity가 높아 기존 구조를 크게 깨뜨리지는 않았으나, completeness와 ARI를 통해 일부 군집이 과도하게 세분화 됨을 확인
- k 증가로 군집이 세밀해졌지만, 군집 간 경계는 오히려 불분명해질 가능성 존재

지표	무엇을 보는 지표인가	값이 높을 때 의미	값이 낮을 때 의미
Inertia	각 데이터가 속한 클러스터 중심까지의 거리 합	X 의미 없음 (항상 k 증가 시 감소)	X 의미 없음
Silhouette Score	클러스터 내부 응집도 + 클러스터 간 분리도	군집이 잘 분리됨	군집이 서로 겹침
Homogeneity	비교대상 클러스터가 기존 클러스터의 하나의 군집에서 왔는지	기존 군집 구조를 존중	기존 군집이 섞임
Completeness	기존 클러스터 군집 하나가 비교대상에서도 하나로 유지되는지	불필요한 분할 없음	과도한 분할 발생
V-measure	Homogeneity + Completeness의 균형	전체 구조가 유사	구조 차이가 큼
ARI(Adjusted Rand Index)	두 군집 결과가 얼마나 동일한지	거의 같은 군집 결과	다른 군집 구조
AMI(Adjusted Mutual Information)	두 군집이 공유하는 정보량	정보 구조가 유사	정보 구조가 다름

4. 모델링(k-means)

6

비지도 학습 결과(k=6) 와 실제 activity와의 관계를 확인했을 때, 센서 패턴만으로 행동 구분이 되는 경향 확인

activity	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
cluster						
0	0	0	0	26	75	4
1	106	158	192	0	0	2
2	0	0	0	248	311	97
3	1	0	0	329	107	438
4	554	22	0	0	0	0
5	20	443	476	0	0	0

- **Cluster 0** → 대부분 WALKING_DOWNSTAIRS
- **Cluster 1** → LATING + SITTING + STANDING
- **Cluster 2** → WALKING + WALKING_DOWNSTAIRS
- **Cluster 3** → WALKING + WALKING_DOWNSTAIRS + WALKING_UPSTAIRS
- **Cluster 4** → 대부분 LAYING
- **Cluster 5** → SITTING + STANDING

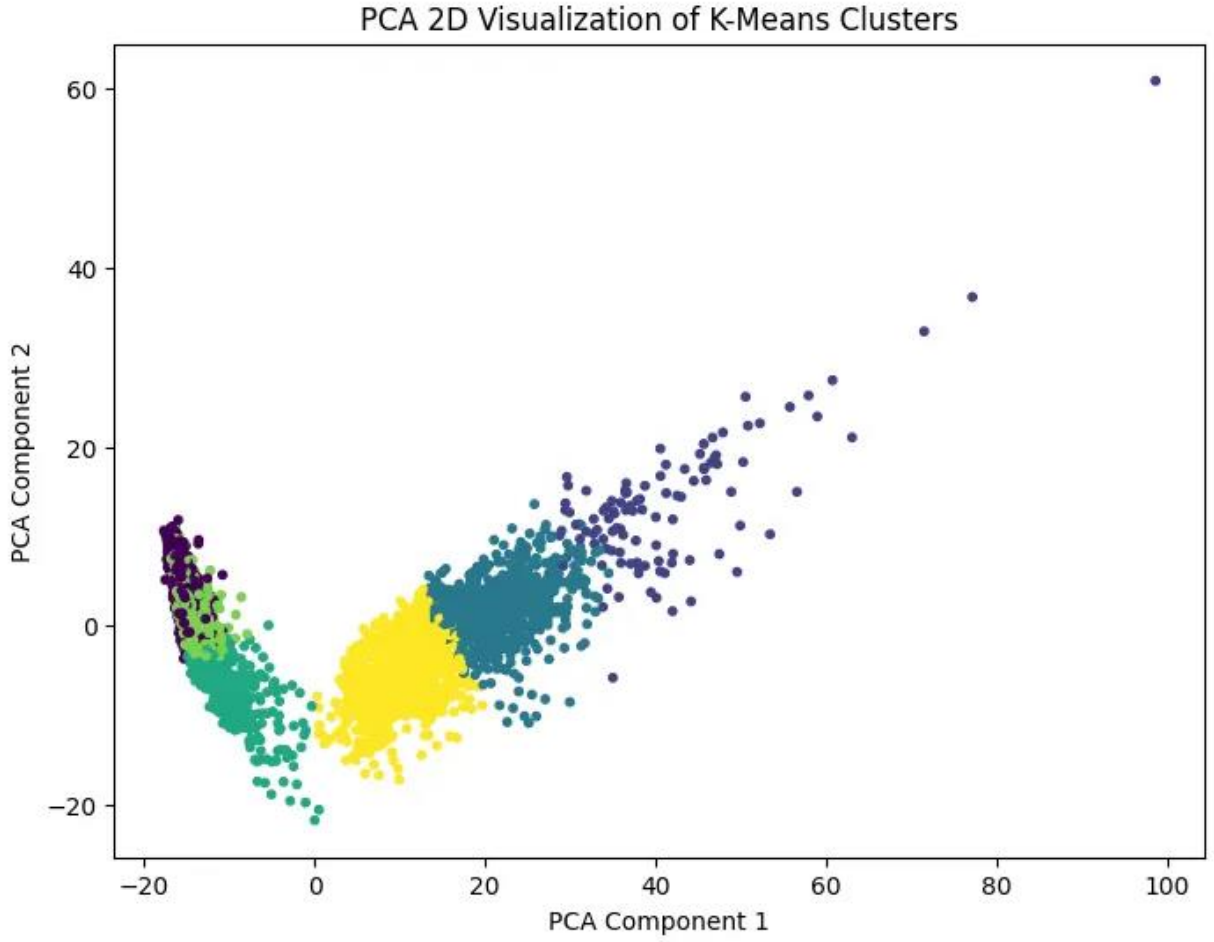
👉 즉, 실제 **라벨을 안 알려줘도 센서 패턴만으로 행동이 구분됨을 확인**

4. 모델링(PCA)

1 문제 범위 설정 — 2 데이터 획득 — 3 데이터 탐색 — 4 모델링 — 5 평가 — 6 배포

8

PCA 차원 축소를 진행하고 시각화를 진행하였음



Activity	의미	정적/동적	센서 구분 난이도
LAYING	누워 있음	정적	⭐ 쉬움
SITTING	앉아 있음	정적	⭐⭐ 어려움
STANDING	서 있음	정적	⭐⭐ 어려움
WALKING	걷기	동적	⭐ 쉬움
WALKING_UPSTAIRS	계단 오르기	동적	⭐⭐ 보통
WALKING_DOWNSTAIRS	계단 내려가기	동적	⭐⭐ 보통

🔍**행동 클래스 의미 PCA / 군집 결과와 연결해서 보면**

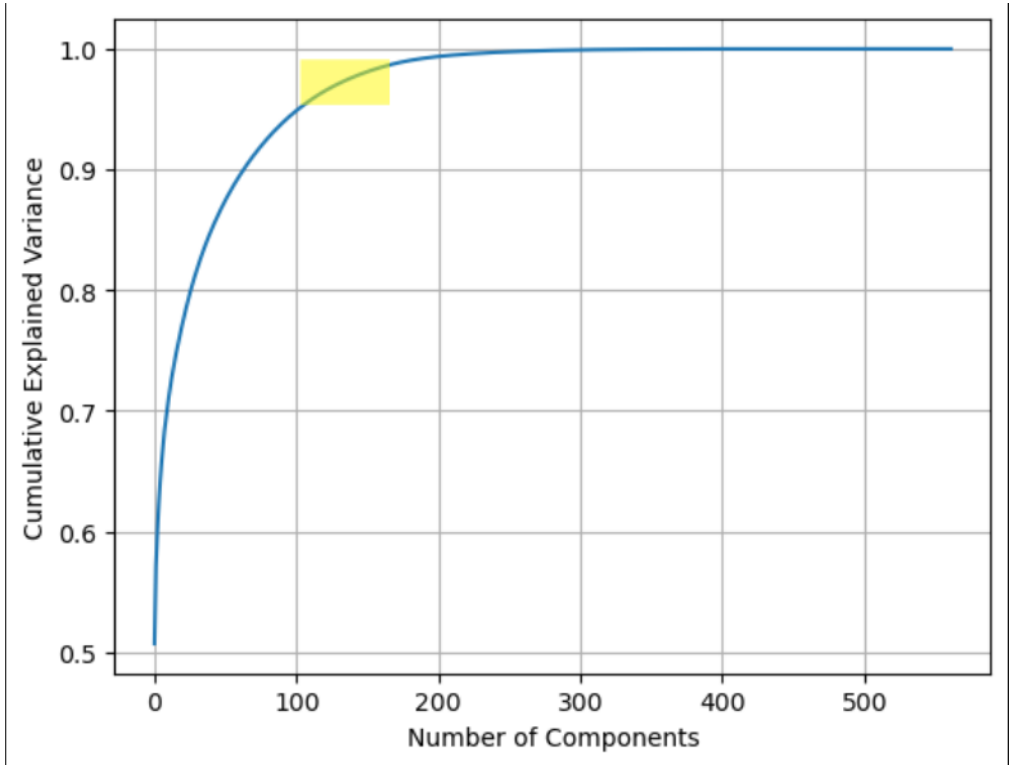
- **LAYING**
  - PCA에서 가장 독립적인 영역
  - 군집에서도 단독 cluster로 나오는 경우 많음
- **SITTING ↔ STANDING**
  - PCA에서 강하게 겹침
  - K-Means에서도 같은 군집에 묶이기 쉬움
  - 👉 정상적인 결과 (센서 한계)
- **WALKING 계열 3종**
  - PCA에서 오른쪽 방향으로 퍼짐
  - 동적 행동 그룹으로 자연스럽게 묶임

5. 평가

8 9

차원 설정에 따라 분류 성능과 설명력이 달라지는 것을 확인함

차원	Random Forest (정확도 점수)	Explained Variance (설명력 점수)
2	0.983380(98.33%)	0.573014
3	0.983380(98.33%)	0.600617
5	0.987535(98.75%)	0.643224
20	0.984765(98.47%)	0.770302
50	0.977839(97.78%)	0.873802
100	0.969529(96.95%)	0.948402
150	0.962604(96.26%)	0.980902
200	0.955679(95.56%)	0.993673



랜덤포레스트를 통한 분류 정확도 점수와, 모델의 설명력 점수를 종합하여 최적 성능을 발휘할 수 있는 차원은 약 100~150차원 정도임을 확인하였음

5. 평가

추가

K-means를 수행하고 PCA를 수행한 것과 K-means 없이 PCA를 수행했을 때 분류 성능에 차이가 있음을 확인함

차원	K-means를 수행 후 PCA 수행	K-means 수행 없이 데이터를 바로 차원축소
2	0.983380(98.33%)	0.533651(53.36%)
3	0.983380(98.33%)	-
5	0.987535(98.75%)	0.838205(83.82%)
20	0.984765(98.47%)	0.907546(90.75%)
50	0.977839(97.78%)	0.932699(93.26%)
100	0.969529(96.95%)	0.938137(93.81%)
150	0.962604(96.26%)	0.933379(93.37%)
200	0.955679(95.56%)	0.942896(94.28%)

- K-means clustering을 거친 후 PCA를 수행하면 이미 기계적으로 분류된 데이터를 기반으로 차원축소를 수행하며 PC1 만으로 class 분류가 가능(재현)해지기 때문에 2차원에서도 분류 정확도가 높게 나오는 것을 확인함
- 하지만 K-means를 거치지 않은 상태에서 activity 라벨만 가지고 분류했을 때는 차원을 늘려가면서 정확도가 올라가는 것을 확인할 수 있음

5. 평가

비지도 학습 clustering vs **지도학습 k-means** vs **PCA 후 분류 성능**을 다양한 방법으로 측정해 봄

분류	Random Forest(정확도 점수)	inertia	silhouette
비지도학습 (k=5)	0.9640	924561.365400	0.136163
비지도학습 (k=6)	0.9432	895968.713509	0.112380
지도학습 (activity 알 때)	0.9626	(Micro F1:0.9596)	(Weighted F1:0.0611)
PCA 2D	0.8421	74639.923622	0.114247

K-means 최초 데이터를 여러 번 fit한 시간과 PCA 이후 fit 한 시간을 비교함

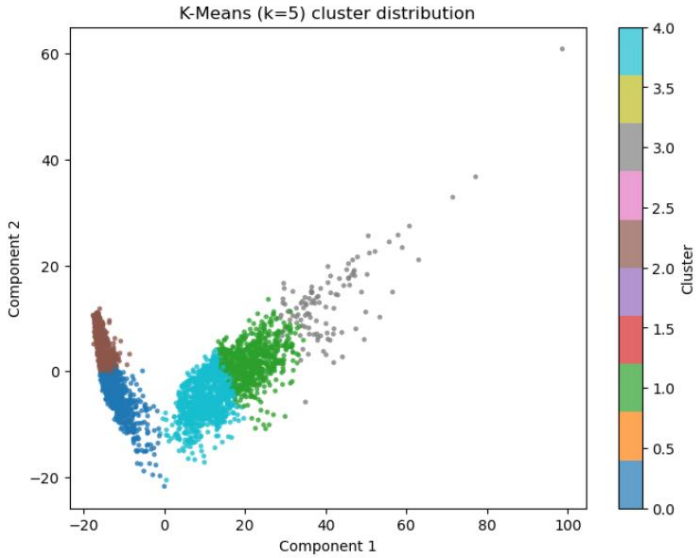
10

분류	10회 fitting 소요 시간(초)
최초 데이터 K-means	1.26
PCA 2D	0.24
PCA 3D	0.67

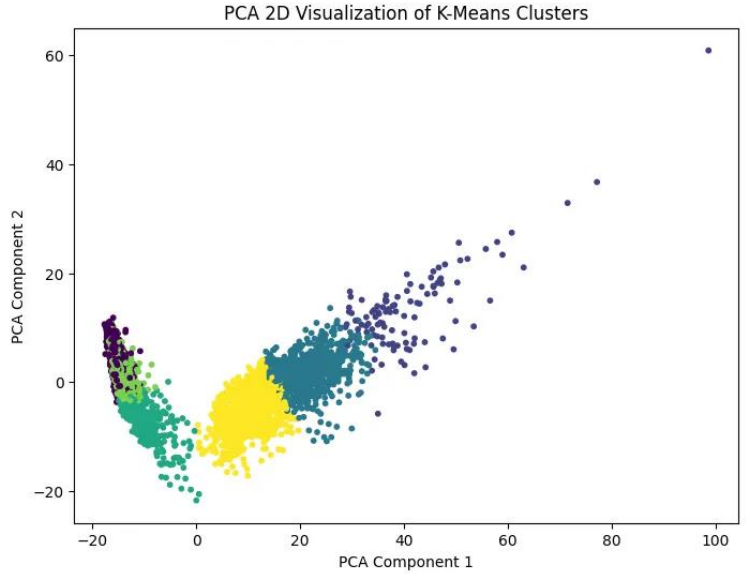
5. 평가

11

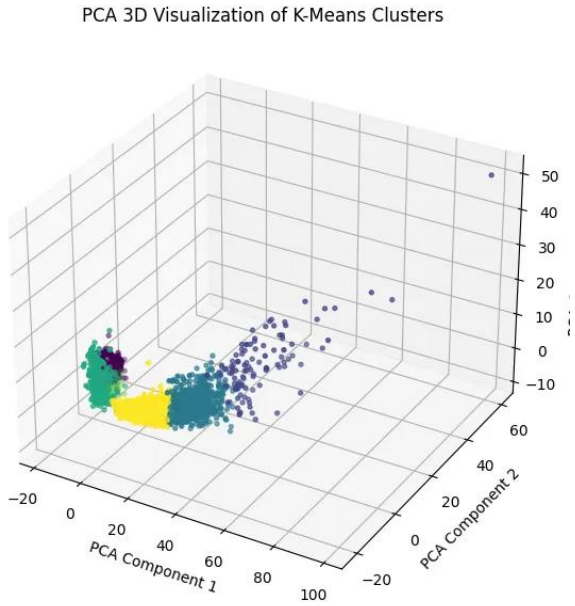
PCA 2D, 3D 시각화 결과 및 t-SNE 시각화 결과 비



K=5일때 2차원 분포그래프



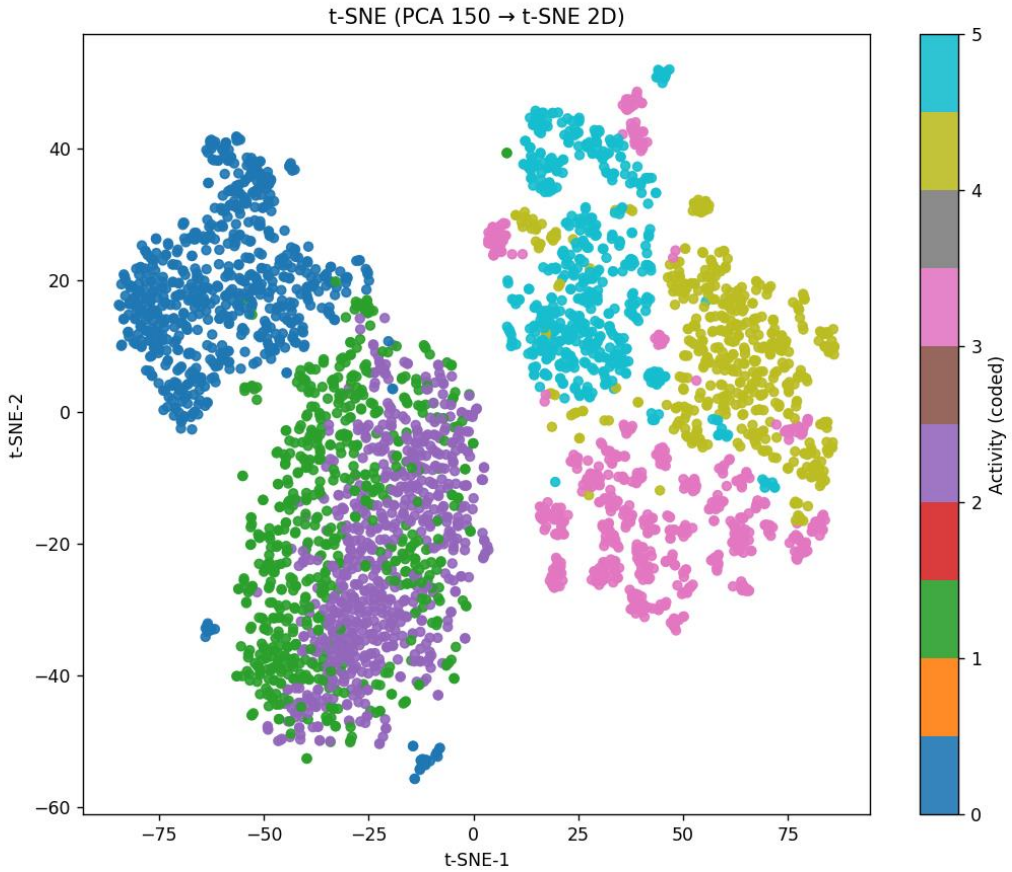
K=6인 상태에서 2차원 그래프



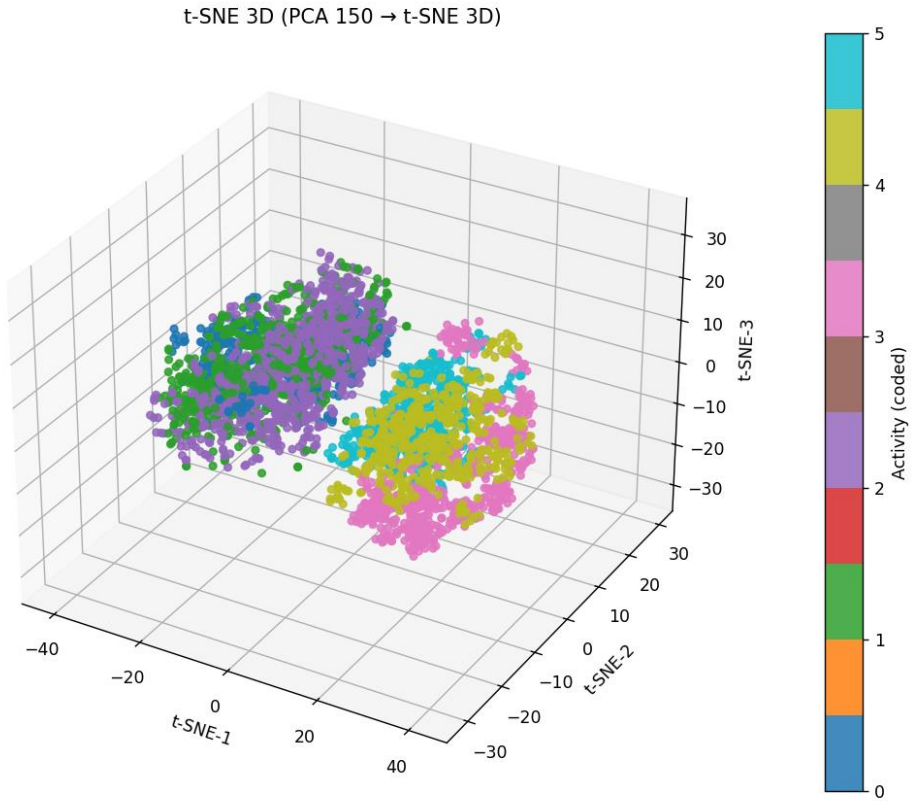
K=6인 상태에서 3차원 그래프

5. 평가

PCA 2D, 3D 시각화 결과 및 t-SNE 시각화 결과 추가



150차원 t-SNE 2차원 그래프



150차원 t-SNE 3차원 그래프

⚠ t-SNE 2차원 시각화 수행 시, PCA 2차원 시각화보다 군집별로 잘 분류됨을 확인, 오히려 3차원 표현을 했을 때는 알아보기 힘든 경향이 존재함

6. 교훈 및 한계와 아쉬운 점

- 각자 실험을 하는 것은 모두 고른 경험을 할 수 있다는 점에서 나쁘지 않았으나, k-means 수행여부, y값 설정 등의 조건을 동일하게 진행 후 실험을 했으면 취합 과정에서 혼란이 덜하지 않았을까 생각함
- 변수를 저장할 때 필요에 따라 이름을 분류해야 한다는 점을 느꼈음
- 시간이 좀 더 있었다면 3D 그래프를 좀 더 다양한 시각에서 볼 수 있었을 것 같음 → 이게 코드로 가능하다는 것을 나중에 확인했음