



Day53; 20221121

| | |
|--|----|
| | 날짜 |
| | 유형 |
| | 태그 |

GitHub - u8yes/AI

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

<https://github.com/u8yes/ai>

u8yes/AI



1 Contributor 0 Issues 0 Stars 0 Forks

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/fdce5b33-a860-487b-bfda-ea57ccb74a2d/02_%EB%A8%B8%EC%8B%A0%EB%9F%AEC%EB%8B%9D_%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/5c9ff938-417a-4f11-b553-977a4a053ef4/03_%EC%84%A0%ED%98%95_%ED%9A%8C%EA%B7%80%EB%B6%84%EC%84%9D.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/41625688-b71d-4419-8158-83e55c5ac01b/06-3_XGBoost.ipynb

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/74cf757c-7241-43c5-8e53-a0ce9d36db79/07_k-Means-Clustering.ipynb

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e70b5c26-deaf-4a1b-848e-7ec17f936562/08_PrincipalComponentAnalysis\(PCA\).ipynb](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e70b5c26-deaf-4a1b-848e-7ec17f936562/08_PrincipalComponentAnalysis(PCA).ipynb)

부스팅

3가지 분류기의 검증 결과, 정확도가 다음과 같다고 가정하면

| | 분류기1 | 분류기2 | 분류기3 |
|-----|------|------|------|
| 정확도 | 0.4 | 0.5 | 0.95 |

- 세 분류기의 정확도

| | 분류기1 | 분류기2 | 분류기3 |
|-----|------|------|------|
| 분류값 | 남자 | 남자 | 여자 |

- 세 분류기의 분류값(하드보팅)

이 경우 가중 투표는 다음과 같이 진행.

- 하드 보팅

- ✓ 남자 : $0.4 + 0.5 = 0.95$
- ✓ 따라서 두 분류기가 남자라고 예측했음에도 가중 투표 결과, 사진의 인물을 여자로 분류.

- 소프트 보팅

- ✓ 남자: $0.4 \cdot 0.7 + 0.5 \cdot 0.8 + 0.95 \cdot 0.1 = 0.775$
- ✓ 여자: $0.4 \cdot 0.3 + 0.5 \cdot 0.2 + 0.95 \cdot 0.9 = 1.075$.
- ✓ 소프트 보팅의 결과, 여자로 최종 결론 도출.

| | 분류기1 | 분류기2 | 분류기3 |
|-----|--------------------|--------------------|--------------------|
| 분류값 | 남자: 0.7 여자: 0.3 | 남자: 0.8 여자: 0.2 | 남자: 0.1 여자: 0.9 |

- 세 분류기의 분류값(소프트보팅)

gradient

미국·영국 ['grɛdiənt] ↗ 영국식 ↗

(명사)

1 (특히 도로·철도의) 경사도

a steep gradient ↗

급경사도

2 (두 지역간 온도·기압 등의) 변화도[증감률]

[영어사전 결과 더보기](#)

descent

미국·영국 [dɪ'sent] ↗ 영국식 ↗

(명사)

1 내려오기, 내려가기, 하강, 강하 (↔ascent)

The plane began its **descent** to Heathrow. ↗

비행기가 히스로 공항으로 하강하기 시작했다.

2 내리막 (↔ascent)

There is a gradual **descent** to the sea. ↗

바다까지는 완만한 내리막 경사가 져 있다[완만한 내리막길이다].

3 혈통, 가문, 가계 (=ancestry)

to be of Scottish **descent** ↗

스코틀랜드 혈통이다

영어사전 다른 뜻 2

XGBoost(eXtreme Gradient Boosting)

- 과적합 방지가 가능한 규제가 포함되어 있다.
- CART(Classification And Regression Tree)를 기반으로 함. 분류와 회귀 둘 다 가능하다.
- 조기종료(early stopping)을 제공한다.
- 양상을 부스팅의 특징인 가중치 부여를 경사하강법(gradient descent)으로 한다.

```
2] : # xgboost 설치 : dos command 창에서  
# conda install -c anaconda py-xgboost  
  
# XGBoost 버전 확인  
import xgboost  
  
print(xgboost.__version__)
```

1.5.0

```
2] : # 위스콘신 유방암 예측  
import pandas as pd  
from sklearn.datasets import load_breast_cancer  
from sklearn.model_selection import train_test_split  
import warnings  
warnings.filterwarnings("ignore") # warning을 무시해줄.  
# 다 무시하는게 아니라 검은색으로 출력되게끔 해줄.
```

```
[4]: dataset = load_breast_cancer()
X_features = dataset.data
y_label = dataset.target

cancer_df = pd.DataFrame(data=X_features, columns=dataset.feature_names)
cancer_df['target'] = y_label

cancer_df.head()
```

[4]:

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | mean fractal dimension | ... | worst texture | worst perimeter | worst area | worst smoothness | worst con |
|---|-------------|--------------|----------------|-----------|-----------------|------------------|----------------|---------------------|---------------|------------------------|-----|---------------|-----------------|------------|------------------|-----------|
| 0 | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.3001 | 0.14710 | 0.2419 | 0.07871 | ... | 17.33 | 184.60 | 2019.0 | 0.1622 | |
| 1 | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.0869 | 0.07017 | 0.1812 | 0.05667 | ... | 23.41 | 158.80 | 1956.0 | 0.1238 | |
| 2 | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.1974 | 0.12790 | 0.2069 | 0.05999 | ... | 25.53 | 152.50 | 1709.0 | 0.1444 | |
| 3 | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.2414 | 0.10520 | 0.2597 | 0.09744 | ... | 26.50 | 98.87 | 567.7 | 0.2098 | |
| 4 | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.1980 | 0.10430 | 0.1809 | 0.05883 | ... | 16.67 | 152.20 | 1575.0 | 0.1374 | |

5 rows × 31 columns

악성, 양성 둘 다 '암'임.

malignant

미국·영국[mə'lɪgnənt] ↗ 영국식 ↗

(형용사)

1 악성의 (→benign), (↔non-malignant)

malignant cells ↗

악성 세포

2 악의에 찬 (=malevolent)

[영어사전 결과 더보기](#)

benign

미국·영국[brɪ'nɪŋ] ↗ 영국식 ↗

1 상냥한, 유순한

2 양성의 (↔malignant)

[영어사전 다른 뜻 2](#)

```
[7]: print(dataset.target_names, '\n')
print(cancer_df['target'].value_counts())
```

['malignant' 'benign']

1 357

0 212

Name: target, dtype: int64

양성과 악성은 우리 몸에 생기는 종양(멍울)을 구별할 때 나오는 용어로, 종양은 세포가 비정상적으로 증식해 덩어리를 형성한 것을 말한다. 이는 양성종양과 악성종양으로 나뉘는데, 악성종양은 ‘암’을 말하며, 양성은 근종(근육층의 일부가 과다하게 증식)·선종(선상피에서 발생)·지방종(정상적 지방조직이 있는 피부 아래 조직에서 많이 발생)·섬유종(섬유 결합조직으로 구성되는 종양) 등을 말한다.

양성종양은 지방이나 신경세포 등이 과도하게 증식해 덩어리가 된 것으로, 커지는 속도가 느릴 뿐더러 일정한 크기 이상 자라지 않는다는 특징이 있다. 또한 암처럼 다른 조직에 전이되지 않으며, 쉽게 치유할 수 있다. 양성종양은 인체에 해는 거의 없으나, 주요 기관에 압박을 가할 경우에는 문제가 될 수 있으므로 이에 따른 치료가 필요하다.

반면 악성종양은 쉽게 말해 ‘암’을 뜻한다. 악성종양은 자라나는 속도가 매우 빠르며, 혈관이나 림프관 등을 통해 신체 다른 조직으로 전이되는 특징이 있다. 따라서 조기에 발견해 치료가 이뤄져야 하며, 치료 후에도 재발 가능성이 매우 높으므로 추적 관찰이 중요하다.

```
: # cancer_df에서 feature를 DataFrame과 Label을 Series 객체 추출
# 맨 마지막 컬럼이 Label임. Feature를 DataFrame은 cancer_df의 첫번째 컬럼에서 맨 마지막 두번째 컬럼까지를 :
# -1 줄라이싱으로
X_features = cancer_df.iloc[:, :-1]
y_label = cancer_df.iloc[:, -1]

# 전체 데이터 중 80%는 학습용 데이터, 20%는 테스트용 데이터 추출(test_size=0.2). # seed = random_state
X_train, X_test, y_train, y_test = train_test_split(X_features, y_label,
                                                    test_size=0.2, random_state=156)

print(X_train.shape, X_test.shape)
(455, 30) (114, 30)
```

요즘 가장 많이 정확도가 높은 알고리즘으로 유명해지고 있다.

정상적으로 warning이 뜸.

사이킷런 wrapper XGBoost의 적용

```
: # 사이킷런 wrapper XGBoost 클래스의 XGBClassifier 임포트
from xgboost import XGBClassifier

xgb_wrapper = XGBClassifier(n_estimators=400, learning_rate=0.1, max_depth=3) # 최적의 투닝은 n_estimators = 적용이 좋은 의사결정트리 갯수
# 의사결정트리가 양상을 가장 적중이 좋다. # 부스팅은 XGBoost가 의사결정트리 알고리즘이 가장 적중이 좋다.
# learning_rate=0.1를 적용은 꼭, gradient_descent를 적용. # 의사결정트리처럼 max_depth 최대 깊이 지정해줄.
# 마치 하이퍼 파라미터처럼 적용해주는 과정에 있음

xgb_wrapper.fit(X_train, y_train)

w_preds = xgb_wrapper.predict(X_test) # test값을 넣어서 예측해줌.
w_pred_proba = xgb_wrapper.predict_proba(X_test)[:,1]
```

[11:09:31] WARNING: ..\src\learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

```
: from sklearn.metrics import accuracy_score # 정확도
from sklearn.metrics import precision_score, recall_score # 정밀도, 재현율
from sklearn.metrics import f1_score

def get_clf_eval(y_test, pred=None, pred_proba=None):
    accuracy = accuracy_score(y_test, pred)
    precision = precision_score(y_test, pred)
    recall = recall_score(y_test, pred)
    f1 = f1_score(y_test, pred)
    print('정확도: {:.4f}, 정밀도: {:.4f}, 재현율: {:.4f}, F1: {:.4f}'.format(accuracy, precision, recall, f1))

: get_clf_eval(y_test, w_preds)
```

정확도: 0.9737, 정밀도: 0.9744, 재현율: 0.9870, F1: 0.9806

```
In [16]: # 조기 종료(early stopping)을 제공.

xgb_wrapper = XGBClassifier(n_estimators=400, learning_rate=0.1, max_depth=3)
evals = [(X_test, y_test)]
xgb_wrapper.fit(X_train, y_train, early_stopping_rounds=100, eval_metric='logloss', eval_set=evals, verbose=True)

ws100_preds = xgb_wrapper.predict(X_test) # test값을 넣어서 예측해줌.
ws100_pred_proba = xgb_wrapper.predict_proba(X_test)[:,1]
```

```
[293]: validation_0-logloss:0.08615
[294]: validation_0-logloss:0.08620
[295]: validation_0-logloss:0.08622
[296]: validation_0-logloss:0.08631
[297]: validation_0-logloss:0.08618
[298]: validation_0-logloss:0.08626
[299]: validation_0-logloss:0.08613
[300]: validation_0-logloss:0.08618
[301]: validation_0-logloss:0.08605
[302]: validation_0-logloss:0.08602
[303]: validation_0-logloss:0.08610
[304]: validation_0-logloss:0.08598
[305]: validation_0-logloss:0.08606
[306]: validation_0-logloss:0.08597
[307]: validation_0-logloss:0.08600
[308]: validation_0-logloss:0.08600
[309]: validation_0-logloss:0.08588
[310]: validation_0-logloss:0.08592
[311]: validation_0-logloss:0.08595
```

```
get_clf_eval(y_test, ws100_preds, ws100_pred_proba)
```

정확도: 0.9649, 정밀도: 0.9620, 재현율: 0.9870, F1: 0.9744

```

: # 조기 종료(early stopping)을 제공.
xgb_wrapper = XGBClassifier(n_estimators=400, learning_rate=0.1, max_depth=3)
evals = [(X_test, y_test)]
xgb_wrapper.fit(X_train, y_train, early_stopping_rounds=10, eval_metric='logloss', eval_set=evals, verbose=True)
# logloss 로그 손실값

ws10_preds = xgb_wrapper.predict(X_test) # test값을 넘어서 예측해줌.
ws10_pred_proba = xgb_wrapper.predict_proba(X_test)[:,1]
# 더 이상 손실이 줄어들지 않을 때 400 셋팅값을 채우지 않고 멈추게 됨.
# early_stopping_rounds=10 최소 10번까지는 해보라는 것.

[0]: validation_0-logloss:0.61352
[1]: validation_0-logloss:0.54784
[2]: validation_0-logloss:0.49425
[3]: validation_0-logloss:0.44799
[4]: validation_0-logloss:0.40911
[5]: validation_0-logloss:0.37498
[6]: validation_0-logloss:0.34571
[7]: validation_0-logloss:0.32053
[8]: validation_0-logloss:0.29721
[9]: validation_0-logloss:0.27799
[10]: validation_0-logloss:0.26030
[11]: validation_0-logloss:0.24604
[12]: validation_0-logloss:0.23156
[13]: validation_0-logloss:0.22005
[14]: validation_0-logloss:0.20857
[15]: validation_0-logloss:0.19999
[16]: validation_0-logloss:0.19012
[17]: validation_0-logloss:0.18182
[18]: validation_0-logloss:0.17473
[19]: validation_0-logloss:0.16766

```

```

: get_clf_eval(y_test, ws10_preds, ws10_pred_proba)

정확도: 0.9561, 정밀도: 0.9615, 재현율: 0.9740, F1: 0.9677

```

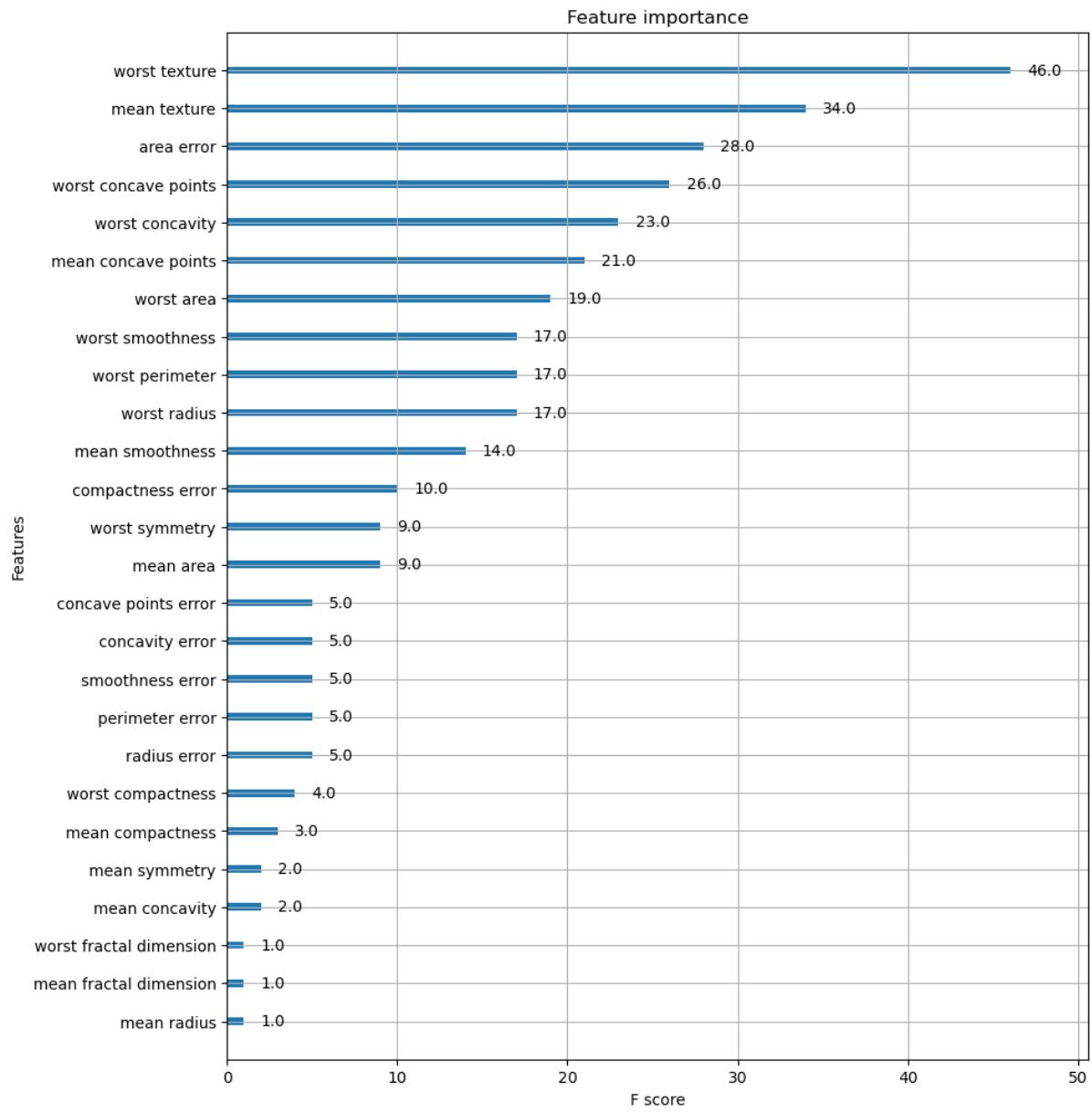
```

]: from xgboost import plot_importance
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(10, 12))
plot_importance(xgb_wrapper, ax=ax) # ax에서 반환해주는 축값

]: <AxesSubplot:title={'center':'Feature importance'}, xlabel='F score', ylabel='Features'>

```



비지도 학습

- 군집화(Clustering)
- 주성분 분석(Principal Component Analysis, PCA)

지도 학습

- k-최근접 이웃(k-Nearest Neighbor, kNN)
 - 서포트 벡터머신(Support Vector Machine, SVM)
 - 의사결정 트리(Decision Tree)
 - 나이브 베이즈(Naïve Bayes)
 - 앙상블(Ensemble)
-

군집화

기본은 무작위로 설정

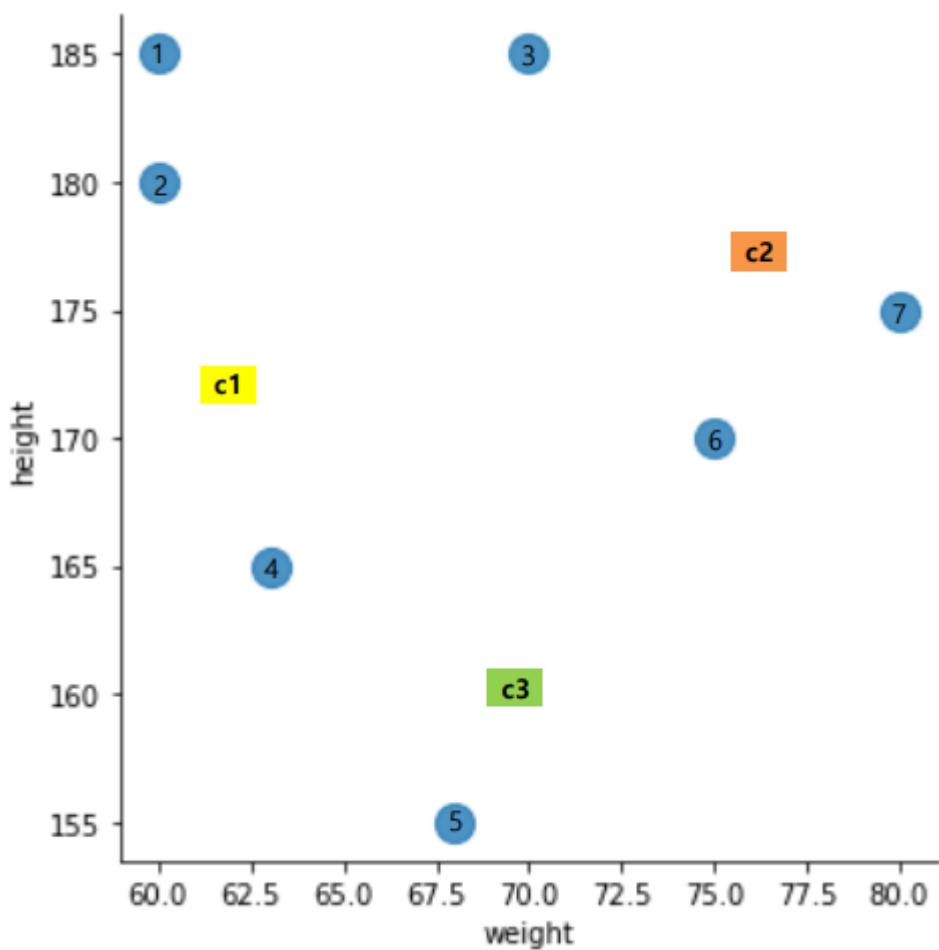
수치화된 데이터

무작위로 설정함.

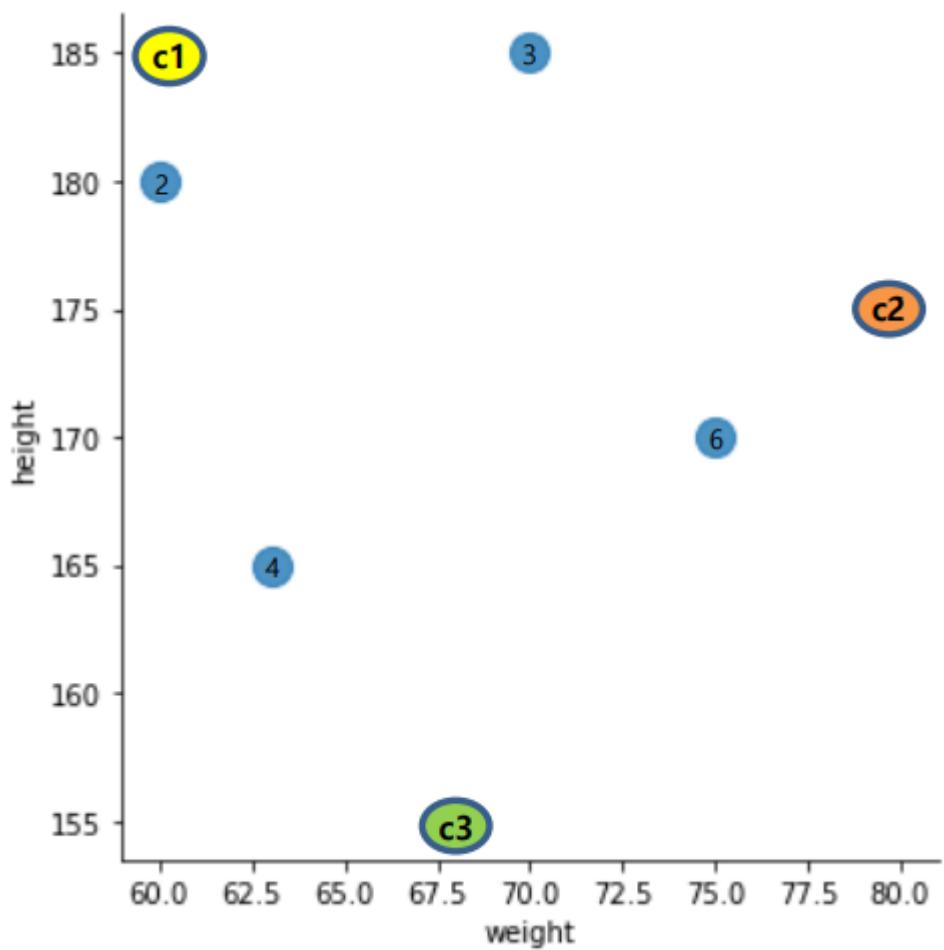
but 항상 무작위를 설정하는 것은 아니다.

경우에 따라 최초 중심을 k 평균 모델에 부여할 수도 있음.

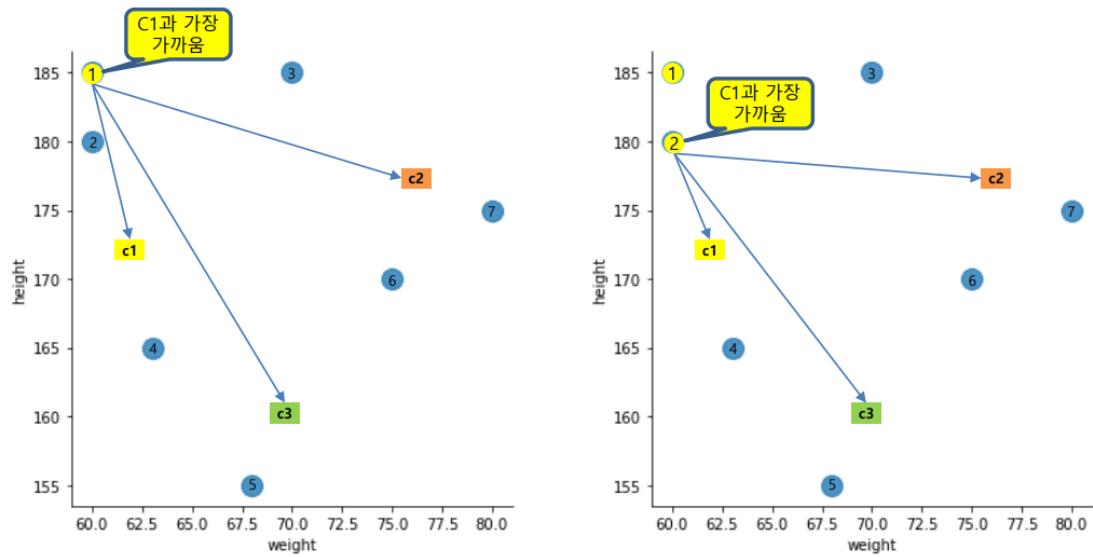
무작위로 클러스터 c1, c2, c3가 잡힘



클러스터의 최초 중심 설정 – kmean++의 중심 정하기.

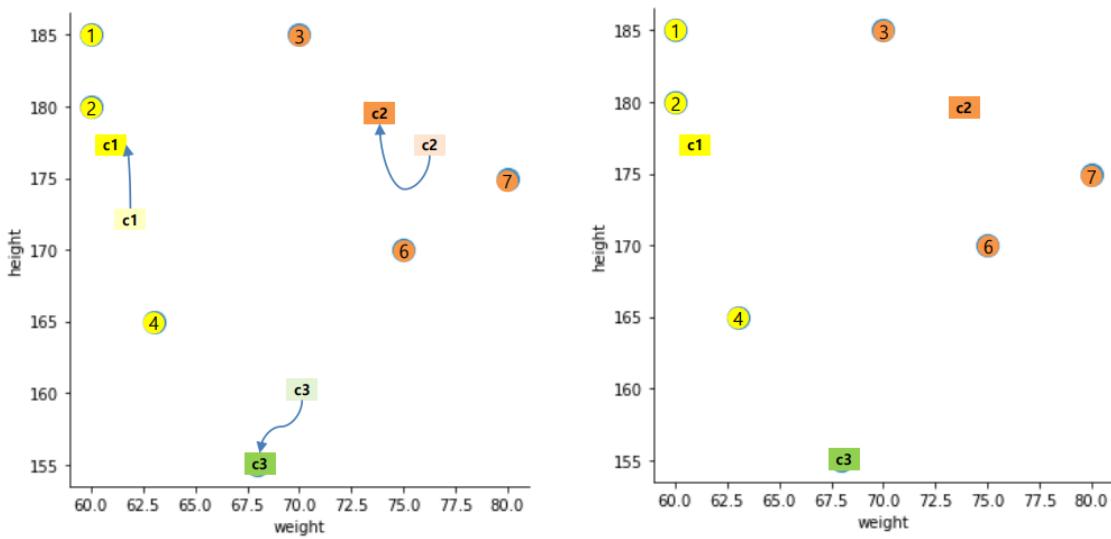


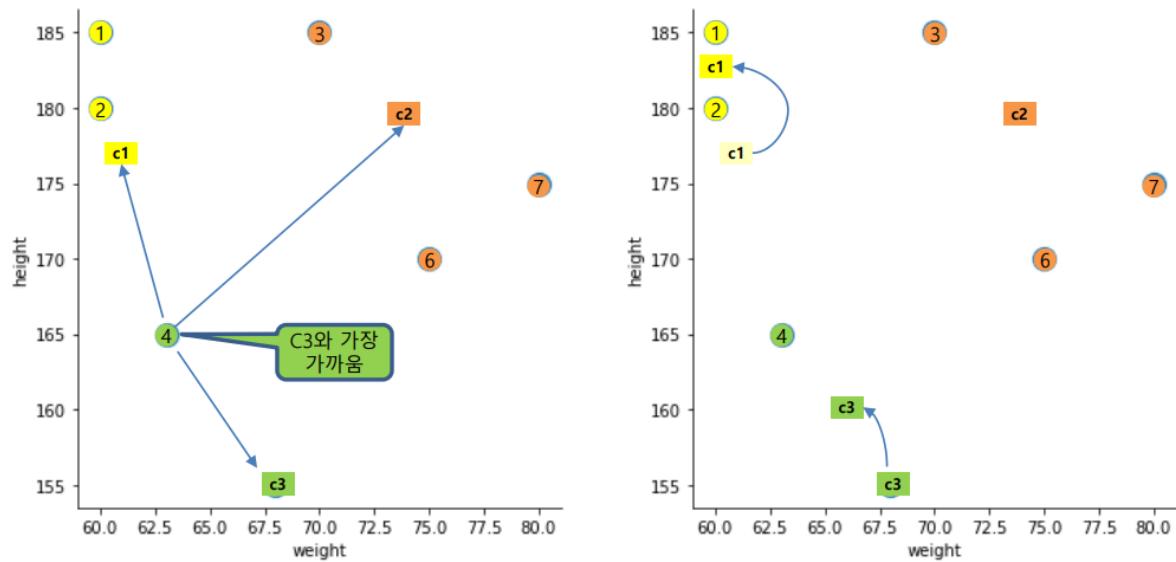
- 표준 k 평균 알고리즘에 따라 무작위로 클러스터의 중심을 설정하고 각 (4) 데이터를 가장 가까운 클러스터에 지정하는 과정.



k 평균 알고리즘

- 데이터 순회가 완료되면 각 (5) 클러스터 중심을 클러스터에 속한 데이터들의 가운데 위치로 변경.





1, 2가 c1

3, 6, 7가 c2

4, 5가 c3

| 학생 | 키 | 몸무게 |
|----|-----|-----|
| 1 | 185 | 60 |
| 2 | 180 | 60 |
| 3 | 185 | 70 |
| 4 | 165 | 63 |
| 5 | 155 | 68 |
| 6 | 170 | 75 |
| 7 | 175 | 80 |

```
] : import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

np.random.seed(5)
```

데이터 만들기

- 사람들의 키와 몸무게 데이터를 만들어 보도록 하겠습니다.

```
] : df = pd.DataFrame(columns=['height','weight'])
```

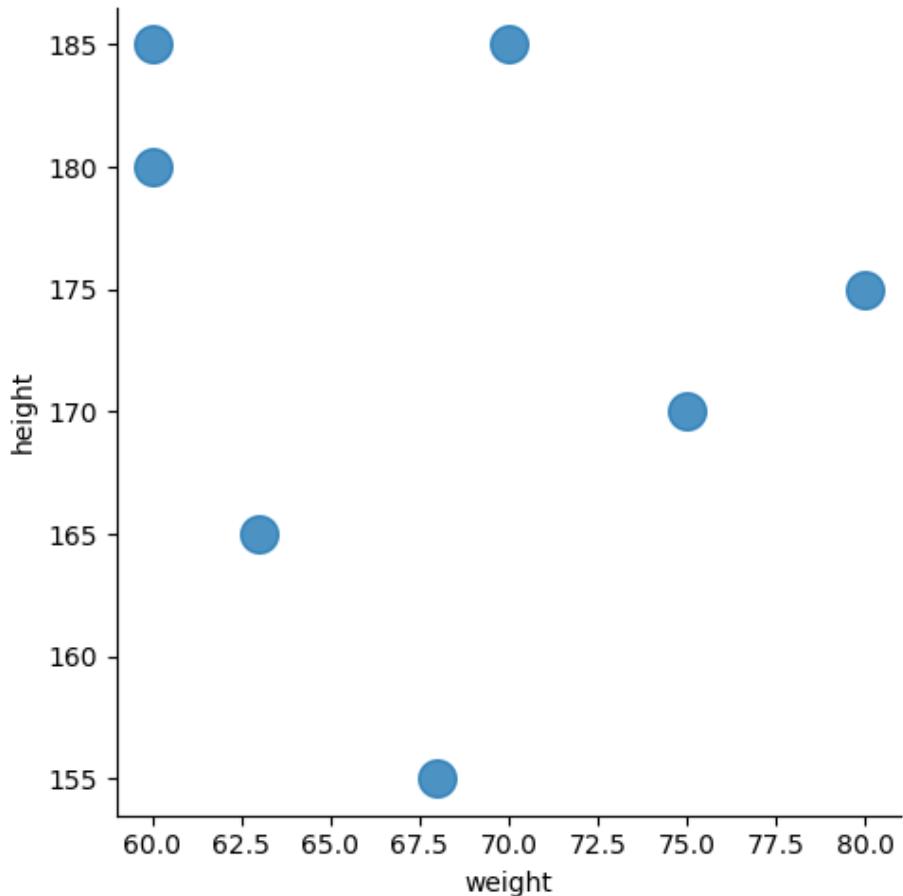
```
df.loc[0] = [185, 60]
df.loc[1] = [180, 60]
df.loc[2] = [185, 70]
df.loc[3] = [165, 63]
df.loc[4] = [155, 68]
df.loc[5] = [170, 75]
df.loc[6] = [175, 80]
```

```
df
```

```
] :
```

| | height | weight |
|---|--------|--------|
| 0 | 185 | 60 |
| 1 | 180 | 60 |
| 2 | 185 | 70 |
| 3 | 165 | 63 |
| 4 | 155 | 68 |
| 5 | 170 | 75 |
| 6 | 175 | 80 |

```
[2]: # 히트맵 시각화  
sns.lmplot(y='height', x='weight', data=df, fit_reg=False, scatter_kws={'s': 200})  
[2]: <seaborn.axisgrid.FacetGrid at 0x28361d24148>
```



k-평균 군집화

- sklearn의 kmean 라이브러리에 데이터를 활용하여, 데이터를 군집화합니다.

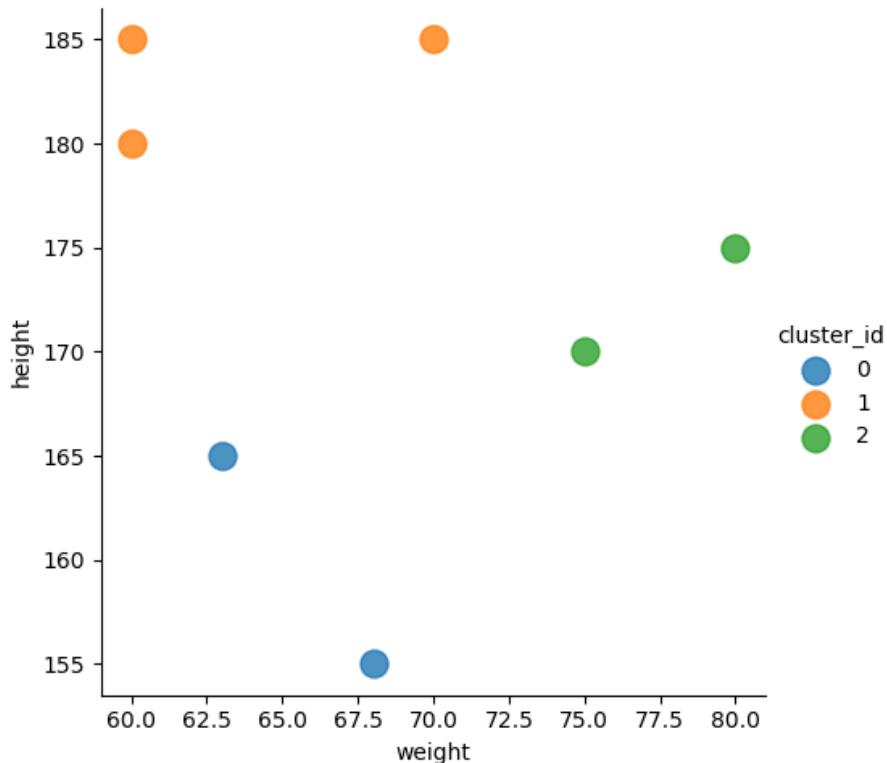
```
29]: data_points = df.values
kmeans = KMeans(n_clusters=3).fit(data_points)

30]: kmeans.cluster_centers_
30]: array([[160.          ,  65.5        ],
       [183.33333333,  63.33333333],
       [172.5         ,  77.5        ]])

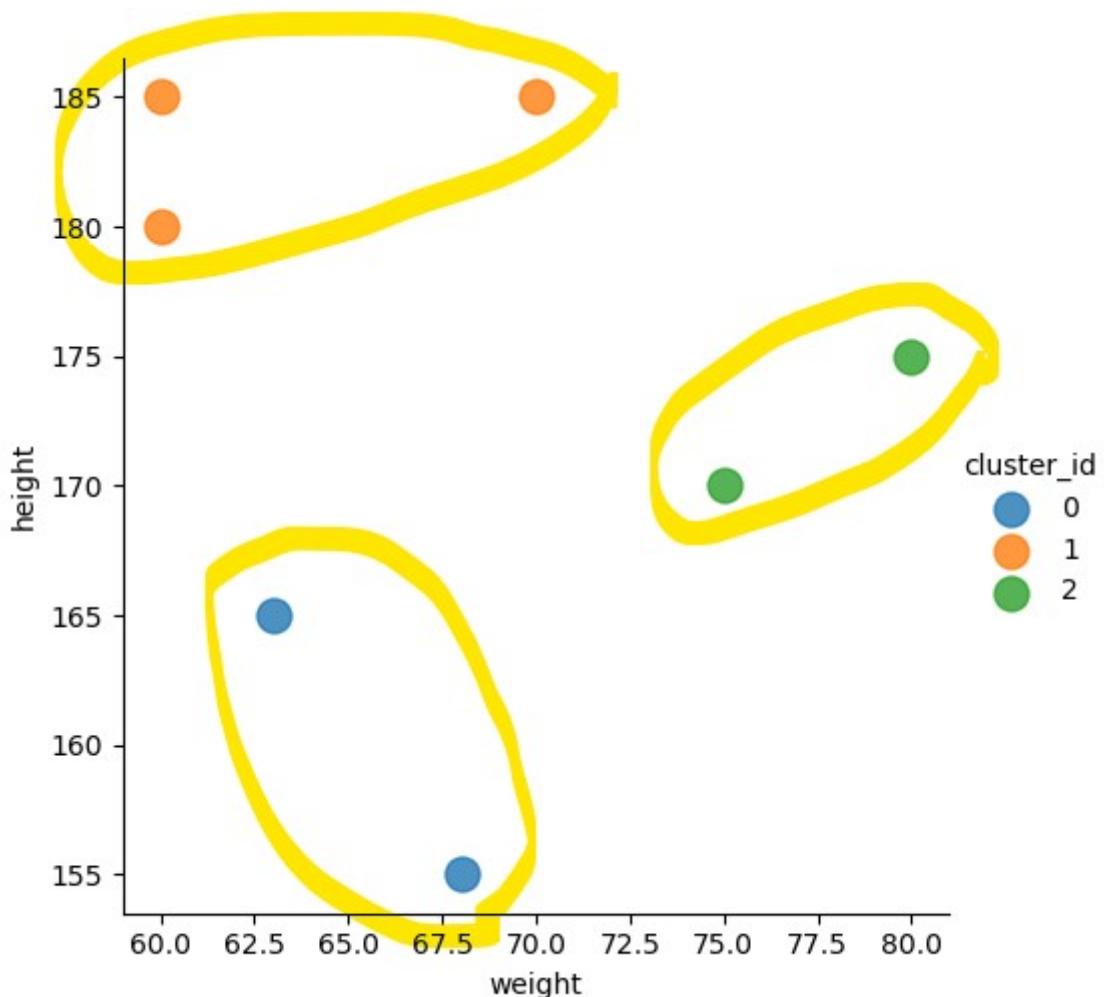
38]: # 데이터가 어느 군집에 소속되어 있는지 데이터프레임 'cluster_id' 컬럼에 저장한다.
df['cluster_id'] = kmeans.labels_
df

38]:
   height  weight  cluster_id
0      185      60          1
1      180      60          1
2      185      70          1
3      165      63          0
4      155      68          0
5      170      75          2
6      175      80          2
```

```
[1]: # k-means 군집 시각화  
sns.lmplot(x='weight', y='height', data=df, fit_reg=False, scatter_kws={'s':150}, hue='cluster_id')  
[1]: <seaborn.axisgrid.FacetGrid at 0x2836420f348>
```



PDF 모양과 다르게 그려졌으니 참고할 것(아래
그림이 정확함)



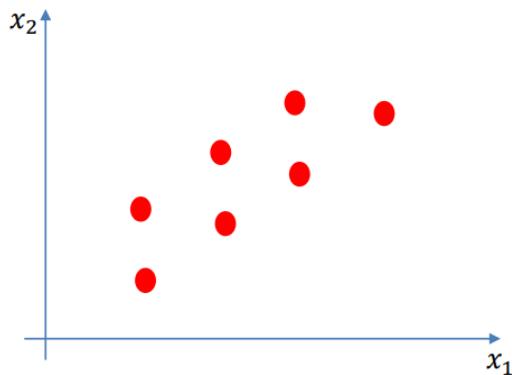
주성분분석 - 비지도 학습

아래는 R의 강의 PDF

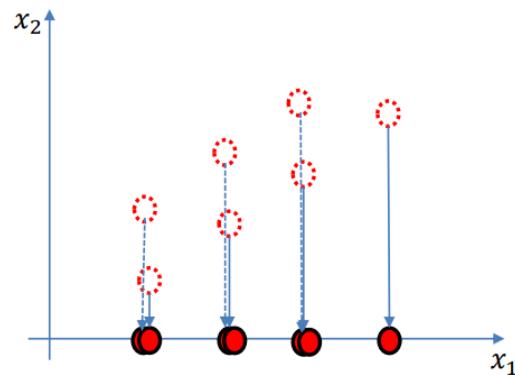
<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/01bfd480-4065-4e3b-b35e-8ccd06235d18/14.%EC%9A%94%EC%9D%B8%EB%B6%84%EC%84%9D.pdf>

주성분 분석(Principal Component Analysis, PCA)

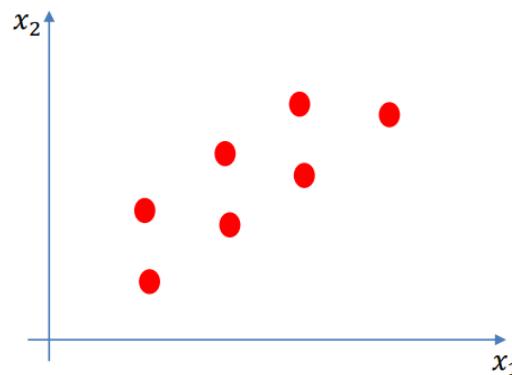
- 고차원의 데이터를 저차원의 데이터로 차원 축소하는 알고리즘.
 - 주로 고차원의 데이터를 3차원 이하의 데이터로 바꿔서 시각화하는 데 많이 사용.
 - 유용한 정보만 살려서 적은 메모리에 저장하거나 데이터의 노이즈를 줄이고 싶을 때도 사용.
-
- 주성분 분석의 특징
 - 데이터의 분산을 최대한 유지하면서 저차원으로 데이터를 변환하는데 있음.
 - 분산을 유지하는 이유는 데이터의 고유한 특성을 최대한 유지하기 위함.



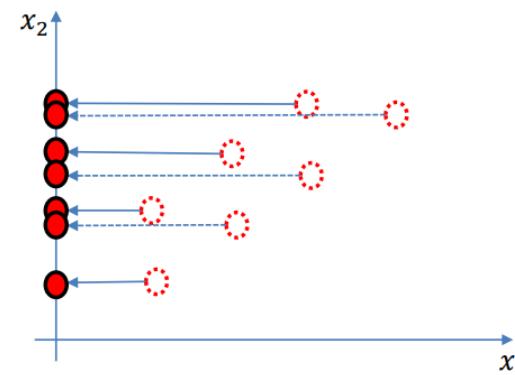
- 2차원 데이터포인트 분포도



- 2차원 데이터를 x_1 축에 투영시켰을 때의 시각화



- 2차원 데이터포인트 분포도



- 2차원 데이터를 x_2 축에 투영시켰을 때의 시각화

5차원이라는 것은 5개의 feature를 가지고 있다는 것

데이터가 5차원 데이터이고,

plot이 항상 실행되게 해줌

%matplotlib inline

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

np.random.seed(5)
```

식습관 데이터를 차원축소시켜서 시각화하기

- 고차원 데이터를 1차원 데이터로 줄여서 시각화해보는 실습.

```
: import pandas as pd
```

데이터 획득

- 주성분 분석을 실습하기 위한 데이터를 만들어보겠습니다.
- 사람들의 하루 식습관 데이터를 만든 후, 주성분 분석을 사용하여 데이터를 시각화.
- 먼저 칼로리, 아침, 점심, 저녁, 운동횟수, 그리고 체형이라는 특징을 가진 데이터 프레임을 생성.

```

]: df = pd.DataFrame(columns=['calory', 'breakfast', 'lunch', 'dinner', 'exercise', 'body_shape'])

]: # 10명의 가공 데이터를 만들
df.loc[0] = [1200, 1, 0, 0, 2, 'Skinny']
df.loc[1] = [2800, 1, 1, 1, 1, 'Normal']
df.loc[2] = [3500, 2, 2, 1, 0, 'Fat']
df.loc[3] = [1400, 0, 1, 0, 3, 'Skinny']
df.loc[4] = [5000, 2, 2, 2, 0, 'Fat']
df.loc[5] = [1300, 0, 0, 1, 2, 'Skinny']
df.loc[6] = [3000, 1, 0, 1, 1, 'Normal']
df.loc[7] = [4000, 2, 2, 2, 0, 'Fat']
df.loc[8] = [2600, 0, 2, 0, 0, 'Normal']
df.loc[9] = [3000, 1, 2, 1, 1, 'Fat']
df

```

| | calory | breakfast | lunch | dinner | exercise | body_shape |
|---|--------|-----------|-------|--------|----------|------------|
| 0 | 1200 | 1 | 0 | 0 | 2 | Skinny |
| 1 | 2800 | 1 | 1 | 1 | 1 | Normal |
| 2 | 3500 | 2 | 2 | 1 | 0 | Fat |
| 3 | 1400 | 0 | 1 | 0 | 3 | Skinny |
| 4 | 5000 | 2 | 2 | 2 | 0 | Fat |
| 5 | 1300 | 0 | 0 | 1 | 2 | Skinny |
| 6 | 3000 | 1 | 0 | 1 | 1 | Normal |
| 7 | 4000 | 2 | 2 | 2 | 0 | Fat |
| 8 | 2600 | 0 | 2 | 0 | 0 | Normal |
| 9 | 3000 | 1 | 2 | 1 | 1 | Fat |

데이터 전처리

- 주성분 분석을 위해 데이터 전처리 과정이 필요.
- 우선 가공 데이터에 데이터 특징과 클래스가 함께 있으므로, 데이터의 특징만으로 구성된 X 데이터프레임을 생성.

```
i: X = df[['calory', 'breakfast', 'lunch', 'dinner', 'exercise']]  
X
```

```
i:  
   calory  breakfast  lunch  dinner  exercise  
0     1200         1      0       0        2  
1     2800         1      1       1        1  
2     3500         2      2       1        0  
3     1400         0      1       0        3  
4     5000         2      2       2        0  
5     1300         0      0       1        2  
6     3000         1      0       1        1  
7     4000         2      2       2        0  
8     2600         0      2       0        0  
9     3000         1      2       1        1
```

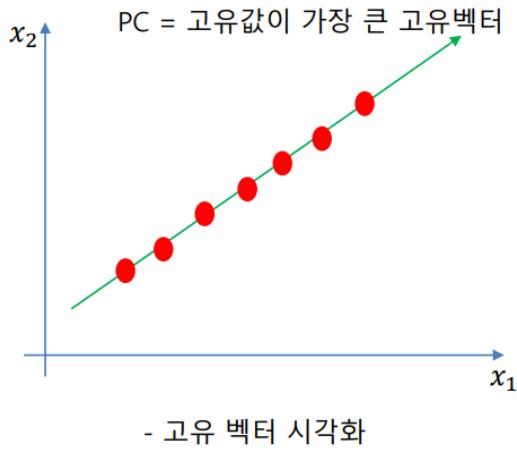
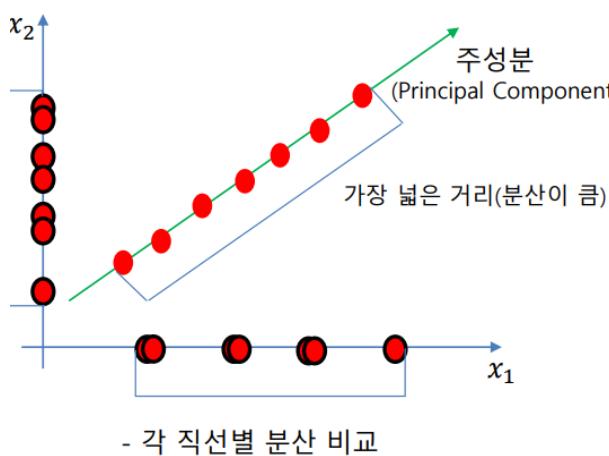
```
: Y = df['body_shape']  
Y  
: 0    Skinny  
1    Normal  
2     Fat  
3    Skinny  
4     Fat  
5    Skinny  
6    Normal  
7     Fat  
8    Normal  
9     Fat  
Name: body_shape, dtype: object
```

데이터 표준화

- 데이터의 특징들이 수치이기는 하지만, 각각의 범위가 다르므로, 범위를 통일시켜줍니다.
- 데이터를 표준화함으로써, 머신러닝 알고리즘이 범위가 넓은 데이터에 중점을 두는 현상이 줄고, 정확도가 높아집니다.

```
: from sklearn.preprocessing import StandardScaler # 수치 표준화시킴.  
x_std = StandardScaler().fit_transform(X)  
x_std  
  
: array([[ -1.35205803,  0.        , -1.3764944 , -1.28571429,  1.        ,  
         [ 0.01711466,  0.        , -0.22941573,  0.14285714,  0.        ,  
         [ 0.61612771,  1.29099445,  0.91766294,  0.14285714, -1.        ,  
         [-1.18091145, -1.29099445, -0.22941573, -1.28571429,  2.        ,  
         [ 1.89972711,  1.29099445,  0.91766294,  1.57142857, -1.        ,  
         [-1.26648474, -1.29099445, -1.3764944 ,  0.14285714,  1.        ,  
         [ 0.18826125,  0.        , -1.3764944 ,  0.14285714,  0.        ,  
         [ 1.04399418,  1.29099445,  0.91766294,  1.57142857, -1.        ,  
         [-0.15403193, -1.29099445,  0.91766294, -1.28571429, -1.        ,  
         [ 0.18826125,  0.        ,  0.91766294,  0.14285714,  0.        ]])
```

분산이 가장 큰 차원은 수학적으로 공분산 행렬(covariance matrix)에서 고유값(eigen value)이 가장 큰 고유 벡터(eigen vector)임.



공분산 행렬(covariance matrix)에서 고유값(eigen value)

고유 벡터(eigen vector)

eigen-

미국·영국 [aɪgən] ↗

((연결형)) [고유의, 독특한]의 뜻

[영어사전 결과 더보기](#)

공분산 행렬 구하기

- 주성분 분석을 하기 위해 가장 먼저 특성들의 공분산 행렬을 구해야 합니다.
- 공분산 행렬(covariance matrix), 고유값(eigen value)

```
] : import numpy as np  
]  
]: features = x_std.T  
covariance_matrix = np.cov(features)  
print(covariance_matrix)  
  
[[ 1.11111111  0.88379717  0.76782385  0.89376551 -0.93179808]  
 [ 0.88379717  1.11111111  0.49362406  0.81967902 -0.71721914]  
 [ 0.76782385  0.49362406  1.11111111  0.40056715 -0.76471911]  
 [ 0.89376551  0.81967902  0.40056715  1.11111111 -0.63492063]  
 [-0.93179808 -0.71721914 -0.76471911 -0.63492063  1.11111111]]
```

고유값(eigen value)과 고유벡터(eigen vector) 구하기

```
] : eig_vals, eig_vecs = np.linalg.eig(covariance_matrix)  
print(eig_vecs, '\n')  
print(eig_vals, '\n')  
  
[[-0.508005 -0.0169937 -0.84711404  0.11637853  0.10244985]  
 [-0.44660335 -0.36890361  0.12808055 -0.63112016 -0.49973822]  
 [-0.38377913  0.70804084  0.20681005 -0.40305226  0.38232213]  
 [-0.42845209 -0.53194699  0.3694462   0.22228235  0.58954327]  
 [ 0.46002038 -0.2816592  -0.29450345 -0.61341895  0.49601841]]  
  
[4.0657343  0.8387565  0.07629538  0.27758568  0.2971837 ]
```

첫번째 벡터의 값

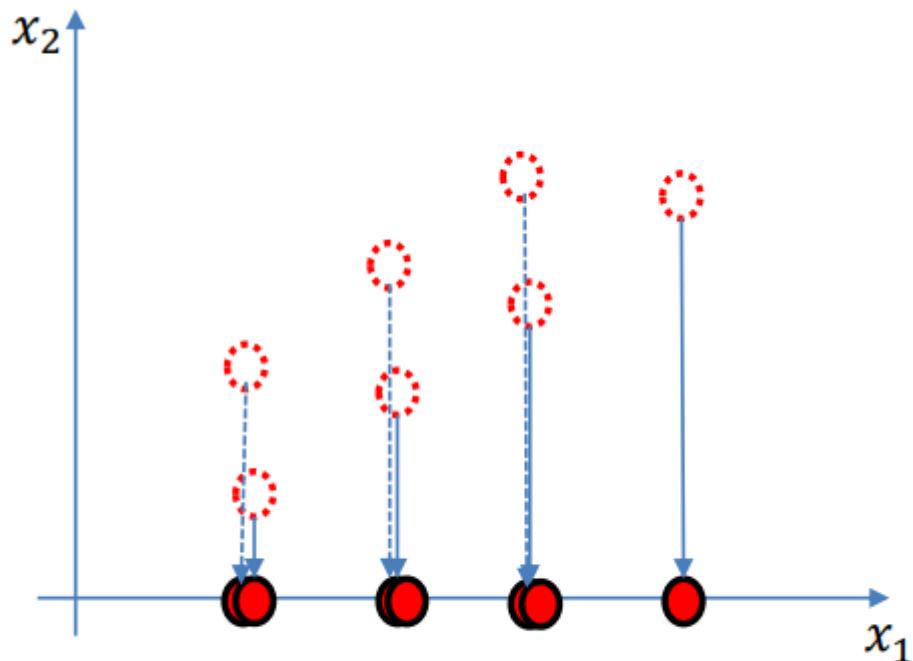
```
] : # 가장 큰 고유벡터의 정보 합유량을 찾는다.  
eig_vals[0] / sum(eig_vals) # 가장 큰 고유 벡터가 73% 정보를 합유 # 5차원 -> 가장 큰 고유벡터(1차원)로 차원 축소.  
]  
0.7318321731427543
```

5차원 데이터를 고유벡터로 사영(=투영)시키기

- a벡터를 b벡터에 사용할 때 공식 = $\text{dot}(a,b) / \text{magnitude of } b$

```
projected_X = x_std.dot(eig_vecs.T[0]) / np.linalg.norm(eig_vecs.T[0])  
projected_X
```

```
array([ 2.22600943,  0.0181432 , -1.76296611,  2.73542407, -3.02711544,  
       2.14702579,  0.37142473, -2.59239883,  0.39347815, -0.50902498])
```



- 2차원 데이터를 x_1 축에 투영시켰을 때의 시각화

시각화

- 1차원으로 축소된 데이터를 시각화하기 위해서, 판다스 데이터프레임에 데이터를 담습니다.
- 주성분(PC1)을 x축으로 하고, 1차원 데이터이므로, y축은 0으로 통일시킵니다.
- 데이터의 쉬운 이해를 위해 클래스를 데이터 마지막 컬럼에 포함시킵니다.

```
[2]: result = pd.DataFrame(projected_X, columns=['PC1'])
result['y-axis'] = 0.0
result['label'] = Y
result
```

```
[2]:
```

| | PC1 | y-axis | label |
|---|-----------|--------|--------|
| 0 | 2.226009 | 0.0 | Skinny |
| 1 | 0.018143 | 0.0 | Normal |
| 2 | -1.762966 | 0.0 | Fat |
| 3 | 2.735424 | 0.0 | Skinny |
| 4 | -3.027115 | 0.0 | Fat |
| 5 | 2.147026 | 0.0 | Skinny |
| 6 | 0.371425 | 0.0 | Normal |
| 7 | -2.592399 | 0.0 | Fat |
| 8 | 0.393478 | 0.0 | Normal |
| 9 | -0.509025 | 0.0 | Fat |

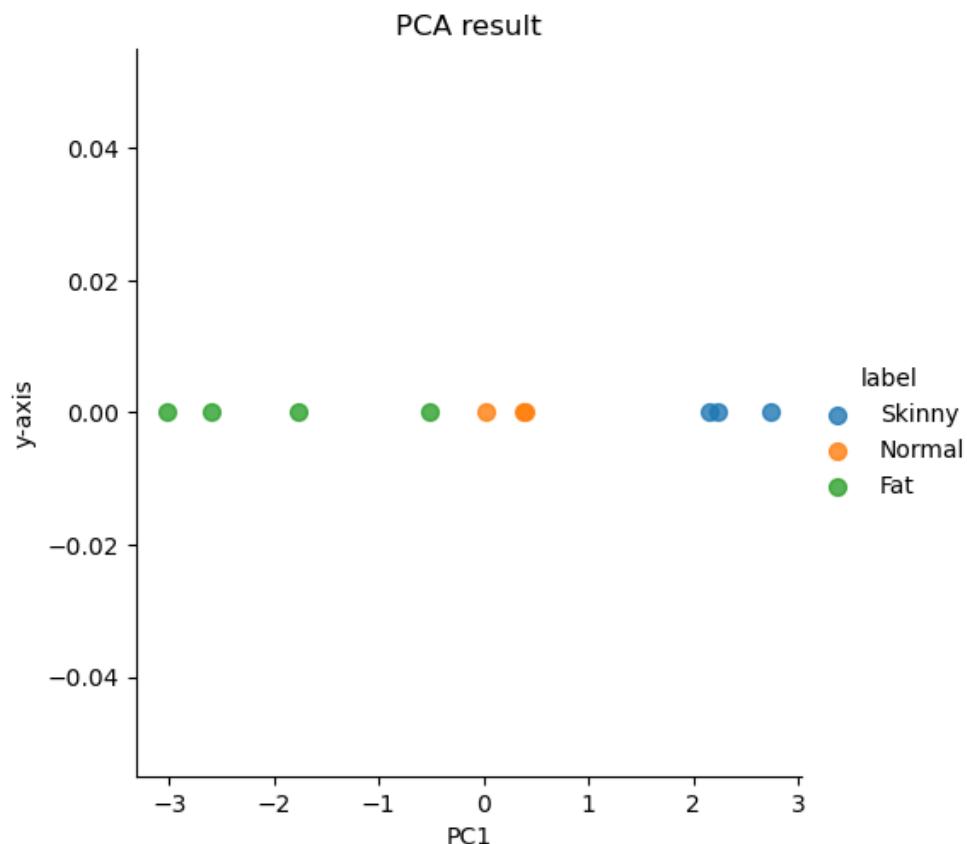
```

: import matplotlib.pyplot as plt
: import seaborn as sns

sns.lmplot(x='PC1', y='y-axis', data=result, fit_reg=False, scatter_kws={'s':50}, hue='label')

plt.title('PCA result')
: Text(0.5, 1.0, 'PCA result')

```



decomposition

(영사)

the decomposition of organic waste ⌂

유기물 폐기물의 분해

[영어사전 결과 더보기](#)

활용형

동사 [decompose](#)

scikit-learn을 사용한 주성분 분석 구현

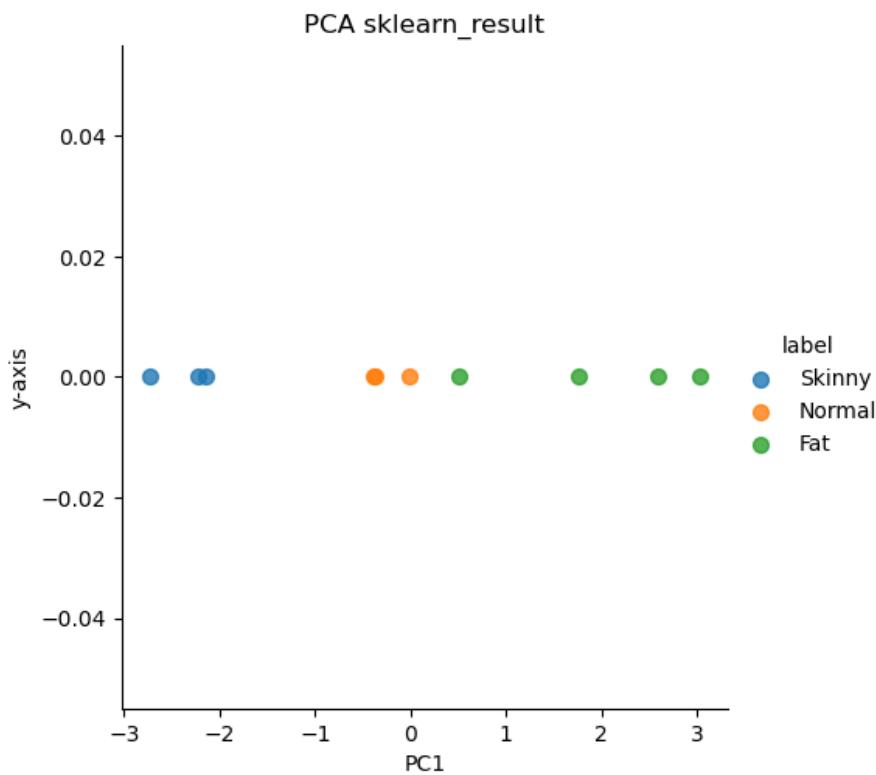
```
[41]: from sklearn.decomposition import PCA  
  
pca = PCA(n_components=1)  
sklearn_pca_x = pca.fit_transform(x_std)
```

```
[48]: sklearn_result = pd.DataFrame(sklearn_pca_x, columns=['PC1'])  
sklearn_result['y-axis'] = 0.0  
sklearn_result['label'] = Y  
sklearn_result
```

```
[48]:
```

| | PC1 | y-axis | label |
|---|-----------|--------|--------|
| 0 | -2.226009 | 0.0 | Skinny |
| 1 | -0.018143 | 0.0 | Normal |
| 2 | 1.762966 | 0.0 | Fat |
| 3 | -2.735424 | 0.0 | Skinny |
| 4 | 3.027115 | 0.0 | Fat |
| 5 | -2.147026 | 0.0 | Skinny |
| 6 | -0.371425 | 0.0 | Normal |
| 7 | 2.592399 | 0.0 | Fat |
| 8 | -0.393478 | 0.0 | Normal |
| 9 | 0.509025 | 0.0 | Fat |

```
[1]: sns.lmplot(x='PC1', y='y-axis', data=sklearn_result, fit_reg=False, scatter_kws={'s':50}, hue='label')  
[2]: plt.title('PCA sklearn_result')
```



선형 회귀분석(Linear Regression Analysis)

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/d50e77c7-e9df-4253-b8fe-eb4d615723a6/03_%EC%84%A0%ED%98%95_%ED%9A%8C%EA%B7%80%EB%B6%84%EC%84%9D.pdf

Galton은 모든 데이터는 평균이 있더라.(회귀)

선형 회귀 분석(Linear Regression Analysis)

- 선형성이라는 기본 가정이 충족된 상태에서 독립변수와 종속변수의 관계를 설명하거나 예측하는 통계방법

선형성 - 1차 함수

선형 회귀 분석(Linear Regression Analysis)

$$y = Wx + b$$

- x : 독립 변수
- y : 종속 변수
- W : 직선의 기울기(가중치 : weight)
- b : y 절편(bias)

1차함수는 결국 기울기와 절편을 찾는 것이다. 찾아가는 과정이 인공지능의 원리이다.

국가자격 종목별 상세정보 | Q-net

국가기술자격, 원서접수 및 시험관련 정보 원서접수, 시험일정, 합격자 발표, 정보안내! 원서접수, 합격자발표, 시험일정, 시험정보 안내, 통계자료 등 수험자들에게 보다 양질의 서비스를 제공하기 위해서 최선을

❸ <http://www.q-net.or.kr/crf005.do?id=crf00505&gSite=Q&gId=&jmCd=1320&examInstiCd=1>

시험현황통계와 자격취득현황통계



국가기술자격통계 홈으로 이동 ►

회귀분석은 연속형을 기반으로 하는 분석이다.

x라는 데이터를 가지고 w기울기, b절편을 찾아내면 함수가 정의된다.

그 함수는 모든 x라도 매칭되는 y를 찾아낼 수 있다.

예측값은 y이다.

fit()이 바로 w기울기, b절편(bias)를 찾는 것이다.

첫 x값을 알려줬을 때 기울기는 무제한이다.