

비지도 학습

조교 이동은

목차

- 비지도 학습의 종류
- 비지도 학습의 도전과제
- 데이터 전처리와 스케일 조정
 - 여러가지 전처리 방법
 - 데이터 변환 적용하기
 - 훈련 데이터와 테스트 데이터의 스케일을 같은 방법으로 조정하기
 - 지도학습에서 데이터 전처리 효과
- 차원축소 특성추출 / Manifold 학습
 - PCA
 - t-SNE
- 군집
 - K – means
 - DBSCAN

비지도 학습의 종류

- 비지도 학습 : 알고있는 출력값이나 정보 없이 학습알고리즘을 가르쳐야 하는 모든 종류의 머신러닝

(답이 없으며 입력 데이터 만으로 데이터에서 의미있는 내용을 추출해야)

- 종류 : 비지도 변환 & 군집화

- 비지도 변환 : 데이터를 새롭게 표현 / 지도 학습을 위한 데이터 전처리에 사용
- 군집화 : 데이터를 비슷한 것끼리 그룹으로 묶는 것

비지도 학습의 도전과제

- 어려운 점 : 알고리즘에 대한 평가
 - 정답이 없는 데이터에 적용하기 때문에 올바른 출력인지 알 수 없음
 - 모델이 잘하고 있는지 직접 결과를 확인해야하는 경우가 多
- 전처리 과정에서의 활용
 - 비지도 학습의 결과로 새롭게 표현된 데이터를 사용해 학습하면 지도 학습의 정확도가 좋아지기도 하며, 메모리와 시간 절약

데이터 스케일 조정

열(특성)의 통계치 이용

- StandardScaler
 - 각 특성의 평균을 0, 분산을 1로 변경하여 모든 특성을 같은 크기로
 - 최솟값과 최댓값 크기를 제한하지 않음
 - $X - X_{\text{mean}} / \text{std}$
- RobustScaler
 - 평균과 분산대신 median과 사분위값을 사용하여 같은 scale을 갖도록
 - 이상치에 영향을 받지 않음
 - $X - q_2 / q_3 - q_1$
- MinMaxScaler
 - 모든 특성이 0과 1사이에 위치하도록 데이터를 변경
 - $\text{MinMaxScaler} = X - X_{\text{min}} / X_{\text{max}} - X_{\text{min}}$

데이터 변환 적용하기

- MinMaxScaler

모든 특성이 0과 1사이에 위치하도록 데이터를 변경

- scaler.fit(X_train)
- scaler.transform(X_train)

```
from sklearn.preprocessing import MinMaxScaler  
  
scaler = MinMaxScaler()
```

** Test Set에 적용 --?

```
scaler.fit(X_train)
```

```
MinMaxScaler(copy=True, feature_range=(0, 1))
```

```
# transform data  
X_train_scaled = scaler.transform(X_train)
```

데이터 변환 적용하기

- `scaler.transform(X_test)`
- 각 특성 별 최소값과 최대값 출력

```
# transform test data
```

```
X_test_scaled = scaler.transform(X_test)
```

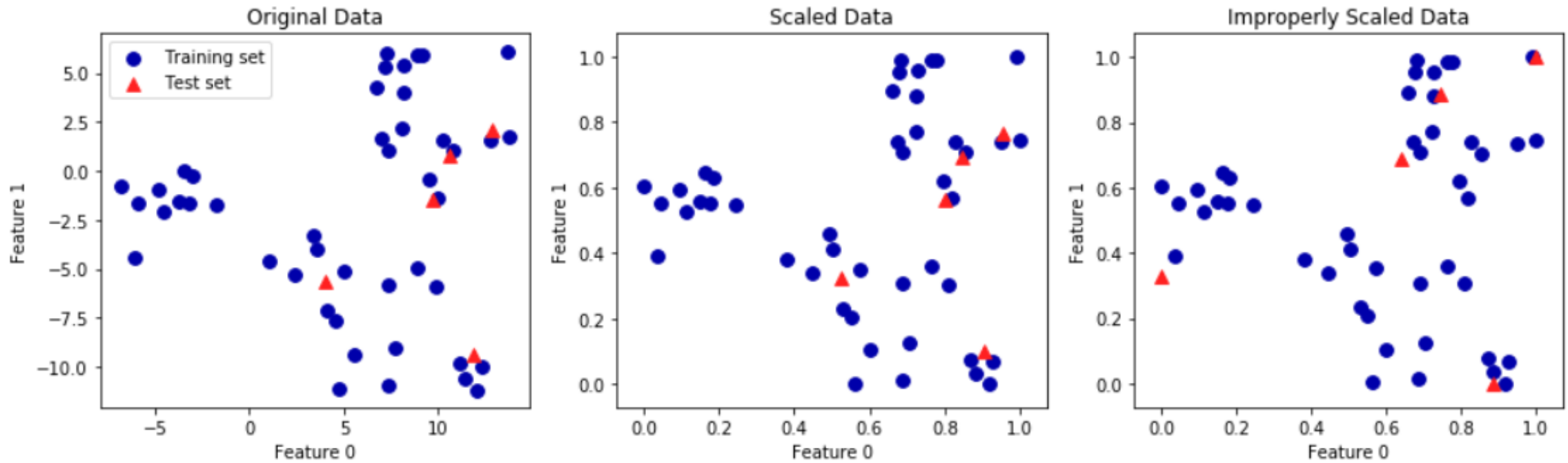
```
per-feature minimum after scaling:
```

```
[ 0.034  0.023  0.031  0.011  0.141  0.044  0.      0.      0.154 -0.006  
 -0.001  0.006  0.004  0.001  0.039  0.011  0.      0.     -0.032  0.007  
  0.027  0.058  0.02   0.009  0.109  0.026  0.      0.     -0.     -0.002]
```

```
per-feature maximum after scaling:
```

```
[0.958 0.815 0.956 0.894 0.811 1.22  0.88  0.933 0.932 1.037 0.427 0.498  
 0.441 0.284 0.487 0.739 0.767 0.629 1.337 0.391 0.896 0.793 0.849 0.745  
 0.915 1.132 1.07  0.924 1.205 1.631]
```

- Scale 조정된 Test set의 최소값과 최대값은 0 과 1이 아니다 !?
- `scaler.transform(X_test)` #X_train으로 fit된 상태
 - 모든 스케일모델은 항상 훈련 세트와 테스트 세트에 같은 변환을 적용
 - 훈련 세트의 최솟값을 빼고 훈련세트의 범위로 나눈다.



모든 Scale 은 항상 훈련세트와 테스트 세트에 같은 변환을 적용해야
`scaler.transform(X_test)`에서 transfor메서드는 `X_train`으로 fit된 , 즉 train 데이터의
 최솟값을 빼고 훈련 세트의 범위(Max - Min)로 나눠준다

MinMaxScaler = $X - X_{min} / X_{max} - X_{min}$

```
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```


► `from sklearn.preprocessing import StandardScaler`
`scaler = StandardScaler()`
`X_scaled = scaler.fit(X_train).transform(X_train)`
`X_scaled_2 = scaler.fit_transform(X_train)`

지도학습을 위한 데이터 전처리(스케일조정)

```
from sklearn.svm import SVC

X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target,
                                                    random_state=0)

svm = SVC(C=100)
svm.fit(X_train, y_train)
print("Test set accuracy: {:.2f}".format(svm.score(X_test, y_test)))
```

Test set accuracy: 0.63

```
# preprocessing using 0-1 scaling
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# learning an SVM on the scaled training data
svm.fit(X_train_scaled, y_train)

# scoring on the scaled test set
print("Scaled test set accuracy: {:.2f}".format(
    svm.score(X_test_scaled, y_test)))
```

Scaled test set accuracy: 0.97

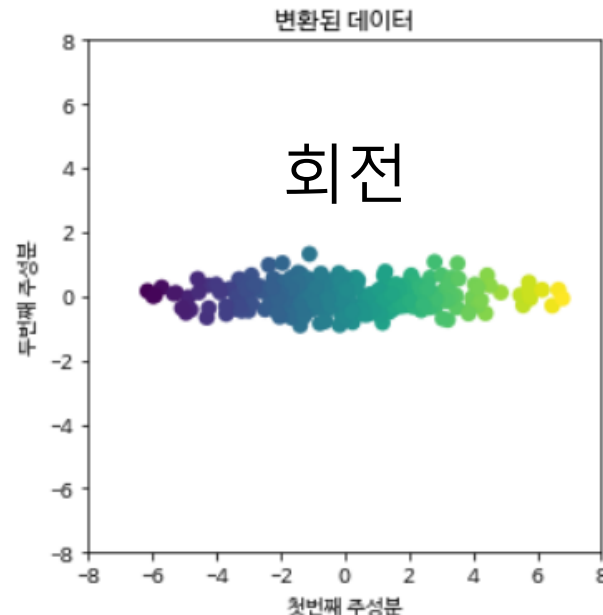
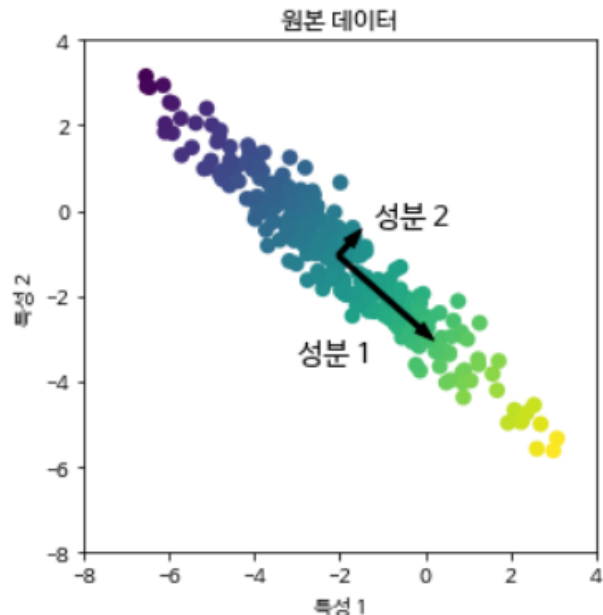
```
# preprocessing using zero mean and unit variance scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# learning an SVM on the scaled training data
svm.fit(X_train_scaled, y_train)

# scoring on the scaled test set
print("SVM test accuracy: {:.2f}".format(svm.score(X_test_scaled, y_test)))
```

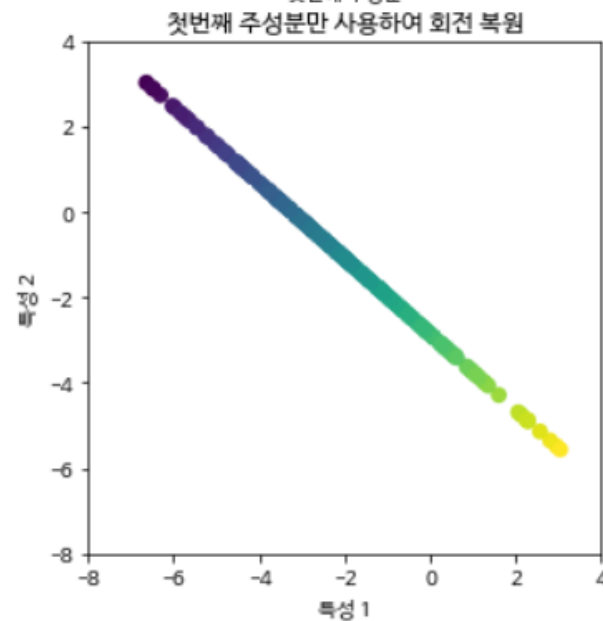
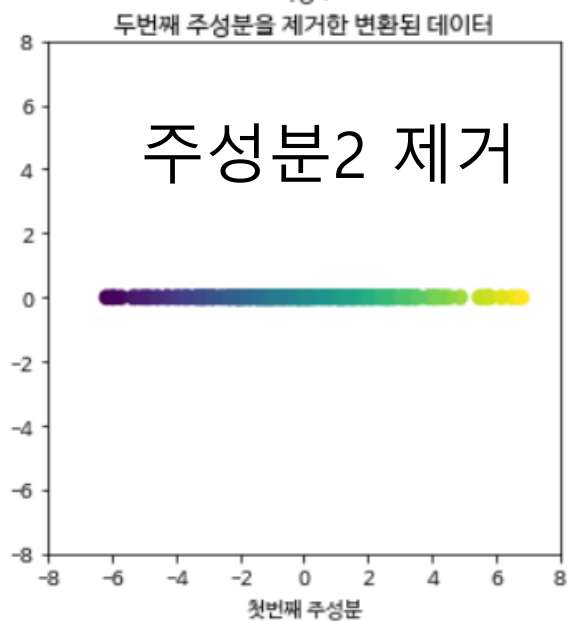
SVM test accuracy: 0.96

주성분 분석(PCA) – Principal Component Analysis

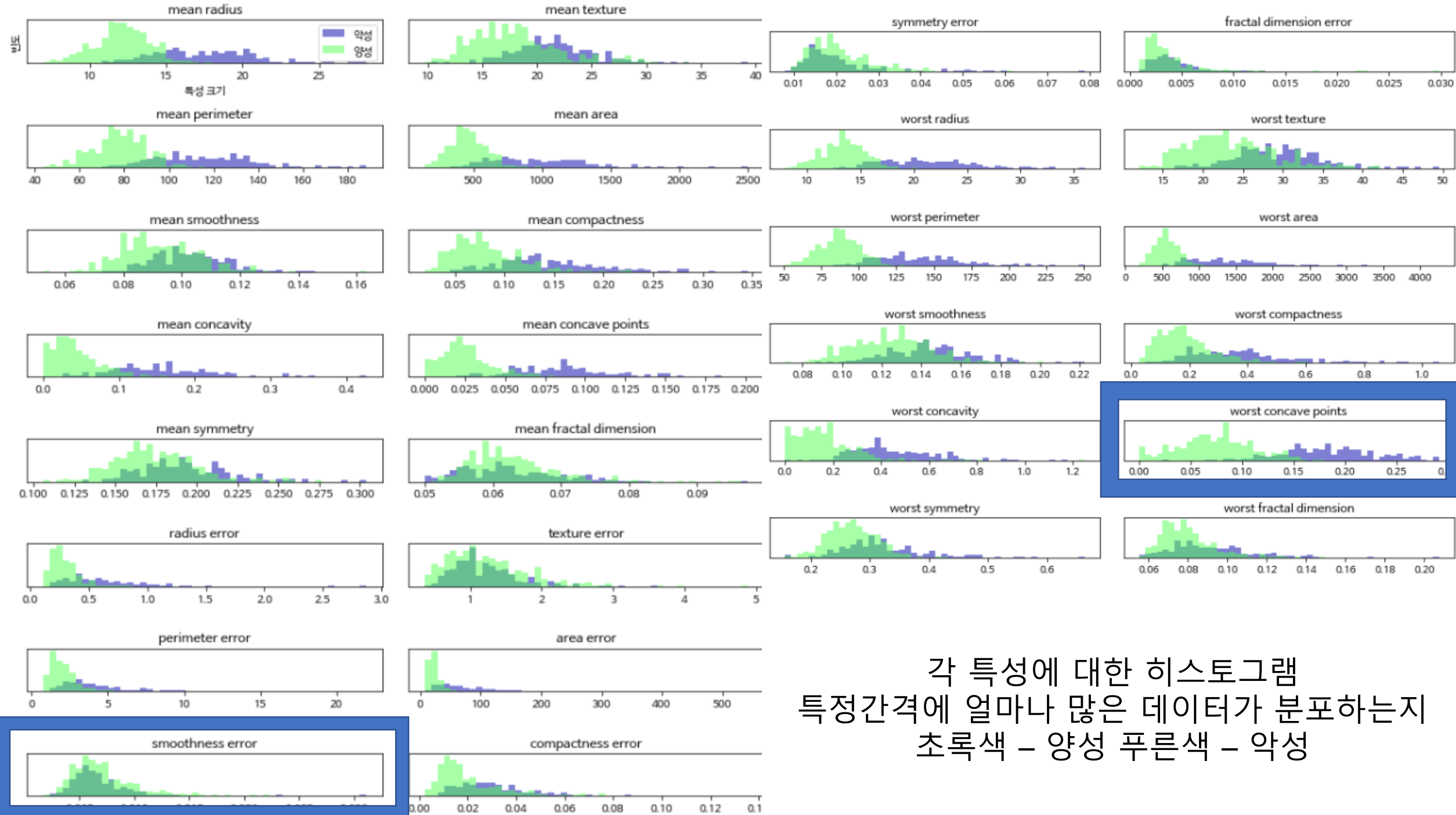


성분1
성분2 : 성분1과 직각인 방향

* 주성분만 남기고 차원 축소



주성분 1만 담고있는 특성1,2



각 특성에 대한 히스토그램
특정간격에 얼마나 많은 데이터가 분포하는지
초록색 - 양성 푸른색 - 악성

```
In [16]: from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()

scaler = StandardScaler()
scaler.fit(cancer.data)
X_scaled = scaler.transform(cancer.data)
```

PCA를 적용하기 전에 StandardScaler를 사용하여
특성의 분산이 1이 되도록 데이터의 스케일을 조정

```
In [17]: from sklearn.decomposition import PCA
# 데이터의 처음 두 개 주성분만 유지시킵니다
pca = PCA(n_components=2)
# 유방암 데이터로 PCA 모델을 만듭니다
pca.fit(X_scaled)

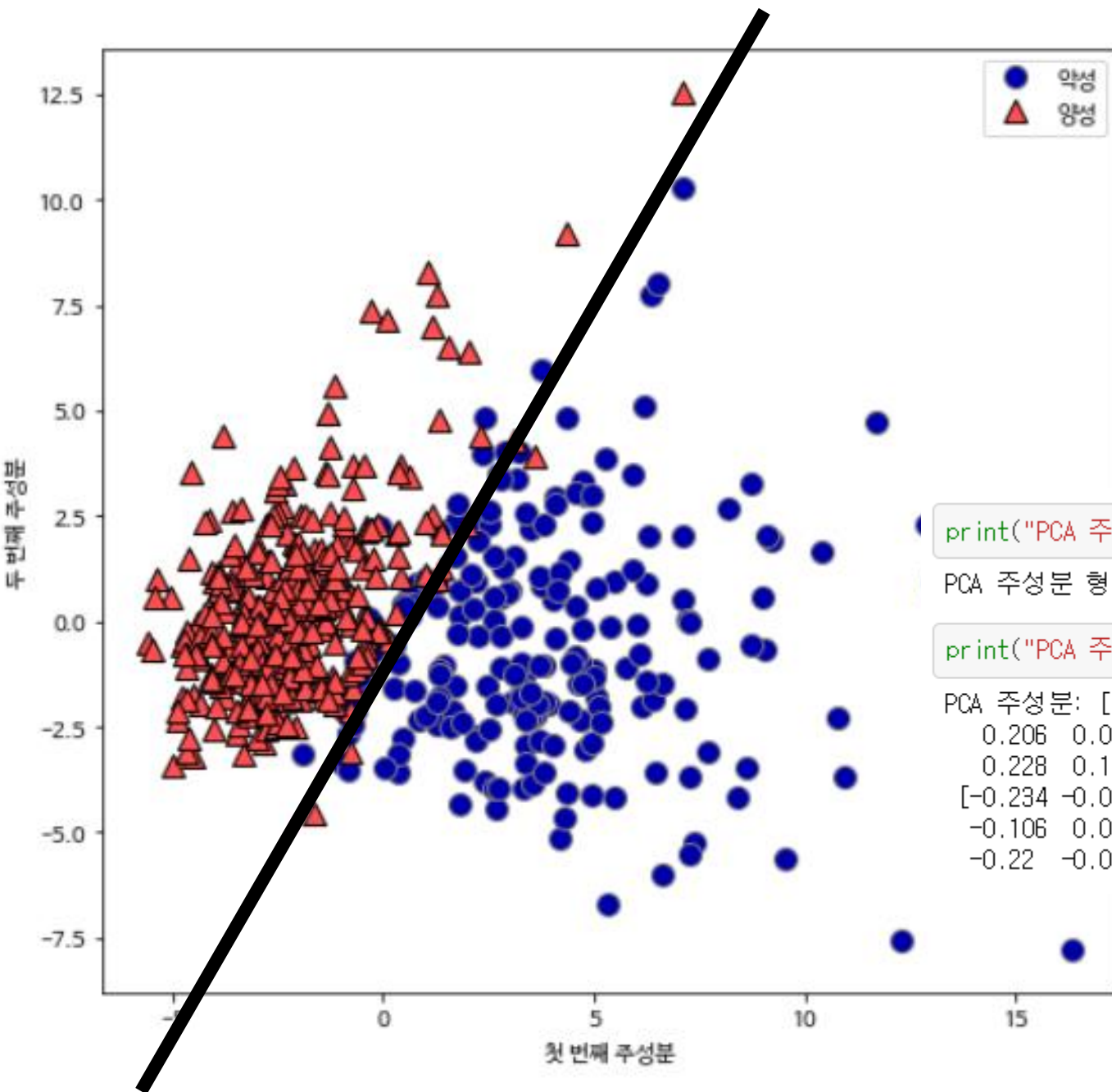
# 처음 두 개의 주성분을 사용해 데이터를 변환합니다
X_pca = pca.transform(X_scaled)
print("원본 데이터 형태:", str(X_scaled.shape))
print("축소된 데이터 형태:", str(X_pca.shape))
```

PCA 객체를 생성 -> fit 메서드로 주성분을 찾고 ->
transform 메서드로 데이터를 회전 및 차원의 축소

PCA 객체를 생성시 n_components 을 통해 얼마나 많은
주성분을 유지할지
기본값은 모든 주성분 저장

```
원본 데이터 형태: (569, 30)
축소된 데이터 형태: (569, 2)
```

```
In [18]: # 클래스를 색깔로 구분하여 처음 두 개의 주성분을 그래프로 나타냅니다.
plt.figure(figsize=(8, 8))
mglearn.discrete_scatter(X_pca[:, 0], X_pca[:, 1], cancer.target)
plt.legend(["악성", "양성"], loc="best")
plt.gca().set_aspect("equal")
plt.xlabel("첫 번째 주성분")
plt.ylabel("두 번째 주성분")
```



PCA 단점: 그래프의 두 축을 해석하기 어렵다.

components_ 속성에 주성분이 저장

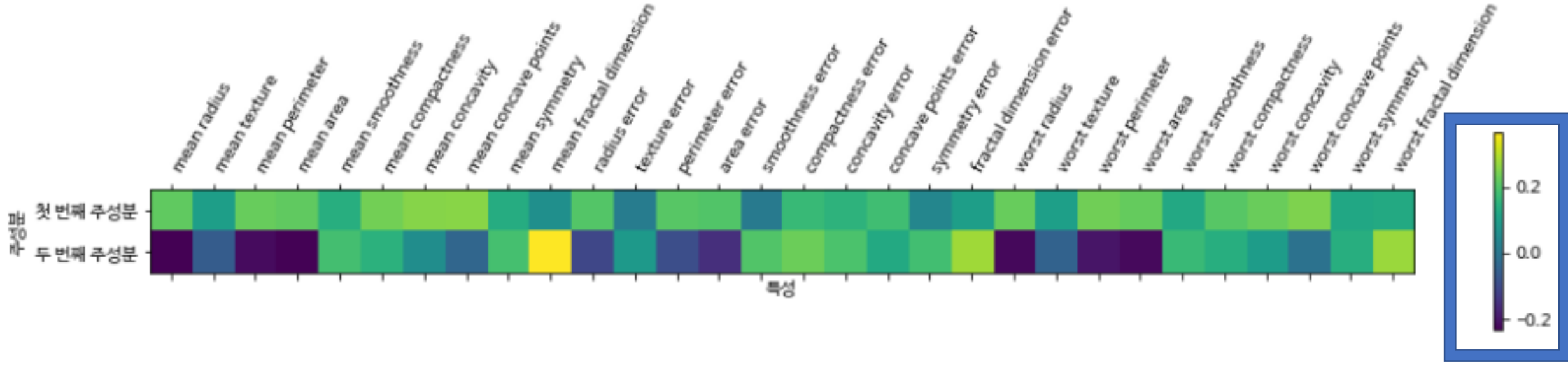
- 각 행은 주성분 하나씩을 나타내며 중요도에 따라 정렬되어 있음.
- 열은 원본데이터의 특성에 대응하는 값

```
print("PCA 주성분 형태:", pca.components_.shape)
```

PCA 주성분 형태: (2, 30)

```
print("PCA 주성분:", pca.components_)
```

PCA 주성분: [[0.219 0.104 0.228 0.221 0.143 0.239 0.258 0.261 0.138 0.064
 0.206 0.017 0.211 0.203 0.015 0.17 0.154 0.183 0.042 0.103
 0.228 0.104 0.237 0.225 0.128 0.21 0.229 0.251 0.123 0.132]
 [-0.234 -0.06 -0.215 -0.231 0.186 0.152 0.06 -0.035 0.19 0.367
 -0.106 0.09 -0.089 -0.152 0.204 0.233 0.197 0.13 0.184 0.28
 -0.22 -0.045 -0.2 -0.219 0.172 0.144 0.098 -0.008 0.142 0.275]]



PCA - 특성추출 (고유얼굴 특성)

- PCA를 이용한 LFW(Labeled Faces in the Wild) 데이터셋의 얼굴 이미지에서 특성을 추출
- 인터넷에서 내려받은 유명 인사들의 얼굴 이미지들로 2000년 초반 이후의 정치인, 가수, 배우, 운동선수들의 얼굴을 포함합니다.



Alejandro Toledo	39	Alvaro Uribe	35	Amelie Mauresmo	21
Andre Agassi	36	Angelina Jolie	20	Ariel Sharon	77
Arnold Schwarzenegger	42	Atal Bihari Vajpayee	24	Bill Clinton	29
Carlos Menem	21	Colin Powell	236	David Beckham	31
Donald Rumsfeld	121	George Robertson	22	George W Bush	530
Gerhard Schroeder	109	Gloria Macapagal Arroyo	44	Gray Davis	26
Guillermo Coria	30	Hamid Karzai	22	Hans Blix	39
Hugo Chavez	71	Igor Ivanov	20	Jack Straw	28
Jacques Chirac	52	Jean Chretien	55	Jennifer Aniston	21
Jennifer Capriati	42	Jennifer Lopez	21	Jeremy Greenstock	24
Jiang Zemin	20	John Ashcroft	53	John Negroponte	31
Jose Maria Aznar	23	Juan Carlos Ferrero	28	Junichiro Koizumi	60
Kofi Annan	32	Laura Bush	41	Lindsay Davenport	22
Lleyton Hewitt	41	Luiz Inacio Lula da Silva	48	Mahmoud Abbas	29
Megawati Sukarnoputri	33	Michael Bloomberg	20	Naomi Watts	22
Nestor Kirchner	37	Paul Bremer	20	Pete Sampras	22
Recep Tayyip Erdogan	30	Ricardo Lagos	27	Roh Moo-hyun	32
Rudolph Giuliani	26	Saddam Hussein	23	Serena Williams	52
Silvio Berlusconi	33	Tiger Woods	23	Tom Daschle	25
Tom Ridge	33	Tony Blair	144	Vicente Fox	32
Vladimir Putin	49	Winona Ryder	24		

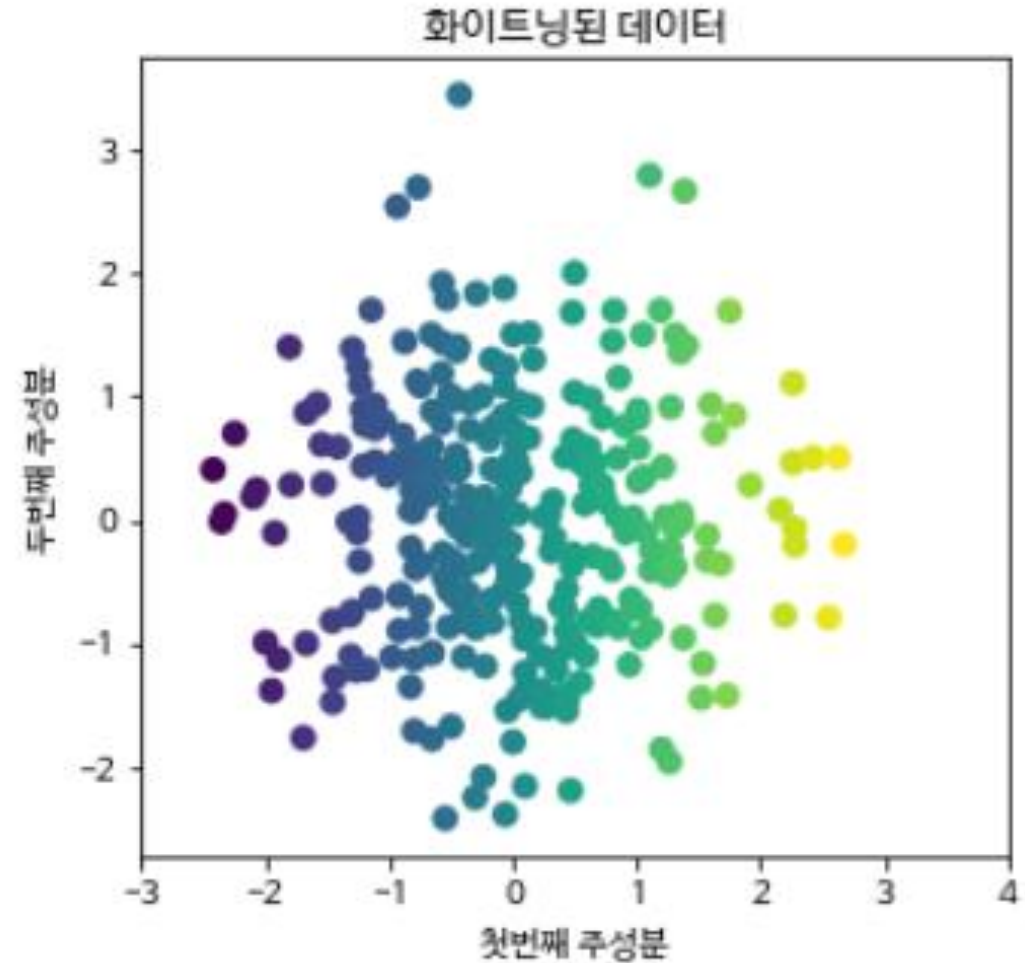
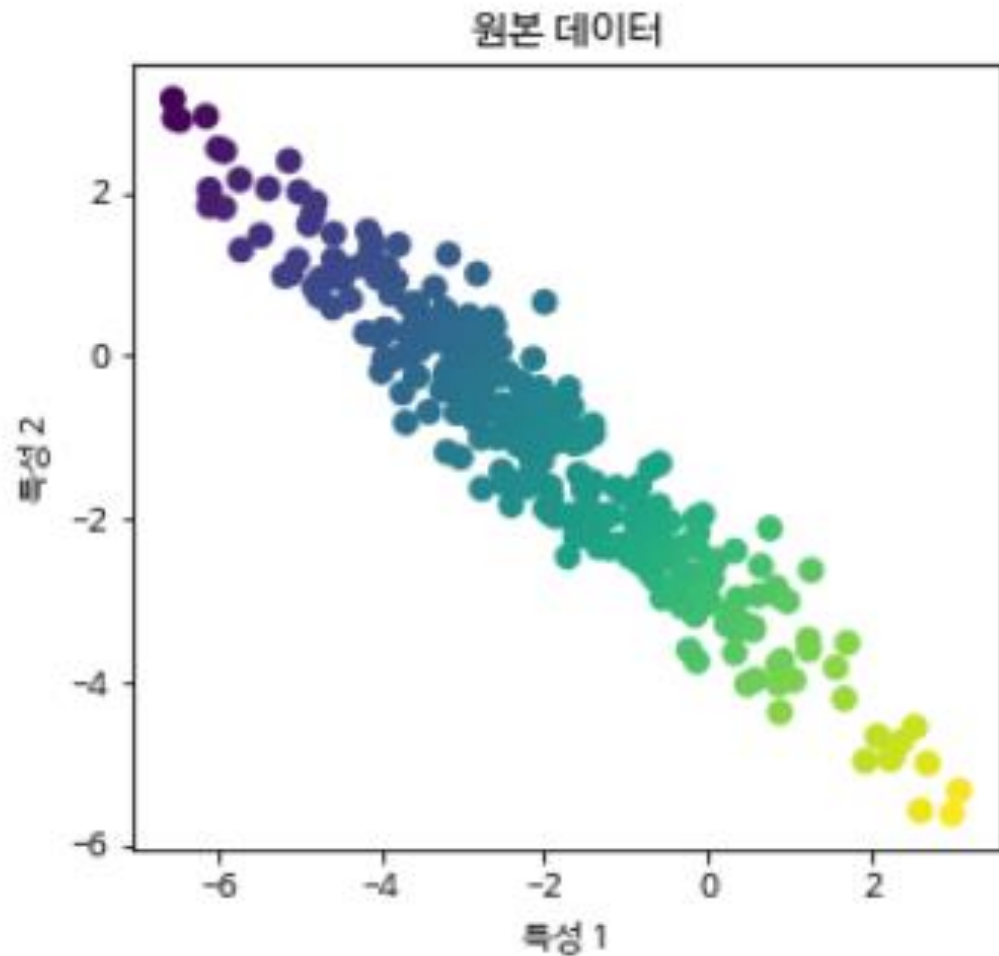
얼굴인식 : 새로운 얼굴 이미지가 데이터베이스에 있는 기존 얼굴 중 하나에 속하는지 찾는 작업

1-Nearest Neighbors Classifier

```
: from sklearn.neighbors import KNeighborsClassifier
# 데이터를 훈련 세트와 테스트 세트로 나눕니다
X_train, X_test, y_train, y_test = train_test_split(
    X_people, y_people, stratify=y_people, random_state=0)
# 이웃 개수를 한 개로 하여 KNeighborsClassifier 모델을 만듭니다
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
print("1-최근접 이웃의 테스트 세트 점수: {:.2f}".format(knn.score(X_test, y_test)))
```

1-최근접 이웃의 테스트 세트 점수: 0.23

PCA_whitening (PCA + StandardScaler)



```
pca = PCA(n_components=100, whiten=True, random_state=0).fit(X_train)
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

```
print("X_train_pca.shape:", X_train_pca.shape)
```

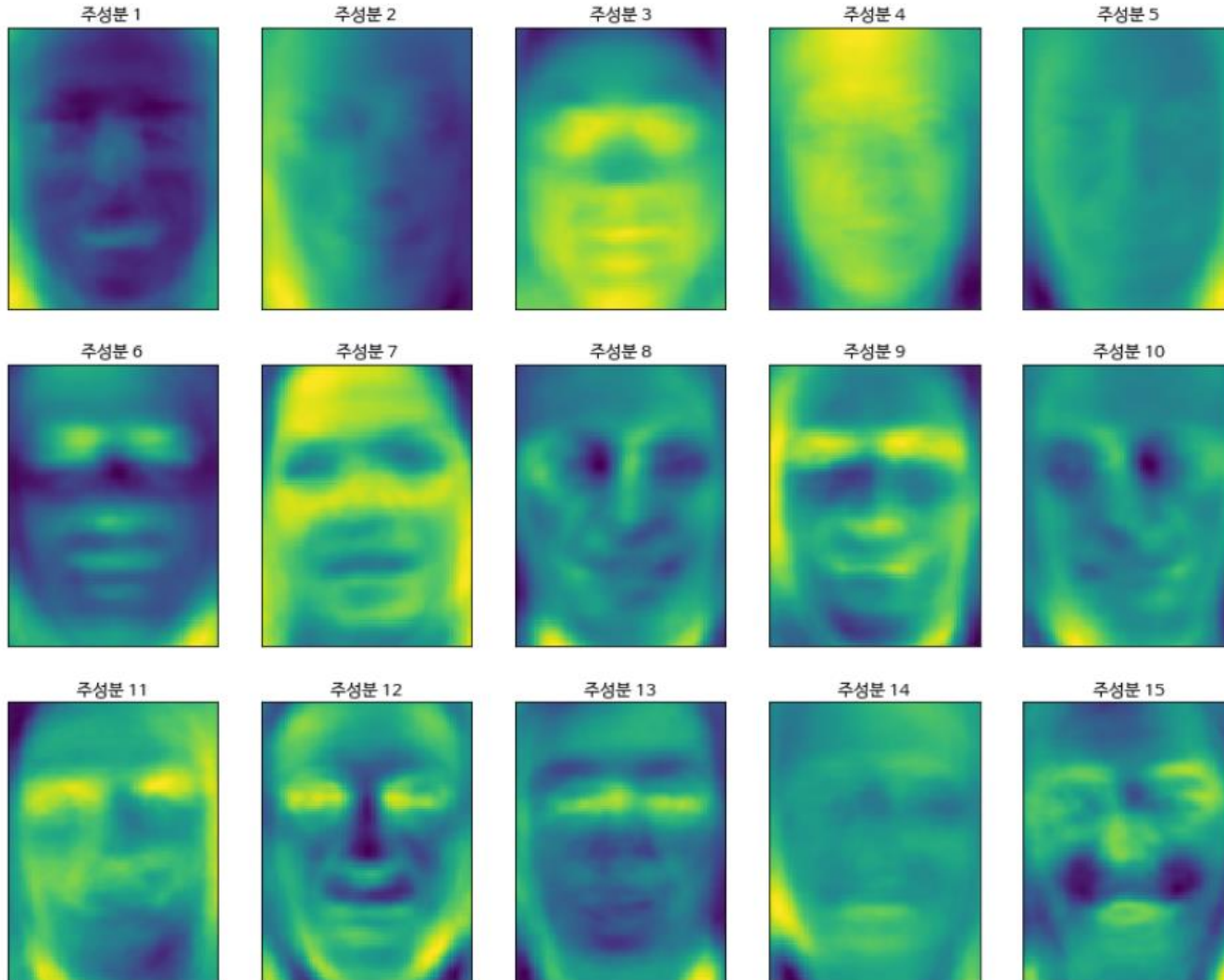
X_train_pca.shape: (1547, 100)

```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train_pca, y_train)
print("테스트 세트 정확도: {:.2f}".format(knn.score(X_test_pca, y_test)))
```

테스트 세트 정확도: 0.31

23% -> 31%

주성분 확인

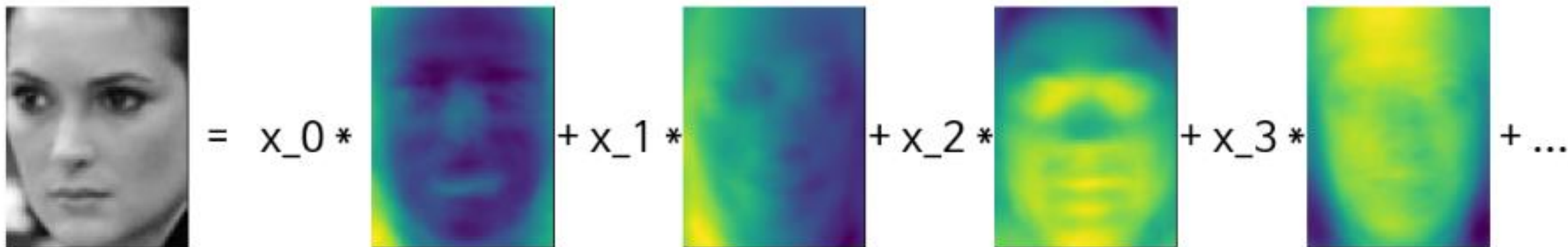


주성분 예측

주성분1 : 얼굴과 배경의 명암차이
주성분2 : 오른쪽과 왼쪽의 조명차이
→ 원본 픽셀 자체를 사용하는 것 보다는
의미가 있지만 사람이 얼굴을 인식하는
방법과는 차이가 있음

나이, 성별, 표정, 머리 모양 같이 픽셀의
강도로는 표현하기 어려운 속성을 사용

이미지를 주성분의 가중치의 합으로 분해한 pca 구성도


$$\text{Image} = x_0 * \text{PC}_0 + x_1 * \text{PC}_1 + x_2 * \text{PC}_2 + x_3 * \text{PC}_3 + \dots$$

주성분 개수에 따른 세 얼굴 이미지의 재구성

