빅데이터분석 실습

비지도학습 기초
주성분 분석
데이터 사이언스 전공
담당교수: 곽철완

강의 내용

- 비지도학습 종류
- 데이터 전처리와 스케일 조정
- 데이터 변환 적용
- 학습용, 평가용 데이터 스케일을 같은 방법으로 조정
- 지도 학습에서 데이터 전처리 효과
- 주성분 분석(PCA)

1. 비지도학습 종류

- **비지도 변환 unsupervised transformation**
  - 데이터를 새롭게 표현하여 다른 알고리즘이 원래 데이터보다 쉽게 해석할 수 있도록 만드는 알고리즘
  - 피처가 많은 고차원 데이터를 피처 수를 줄이면서 중요한 피처만 포함한 데이터로 표현하는 방법
    - 차원축소 dimensionality reduction : 시각화를 위해 데이터세트를 2차원으로 변경
  - 데이터를 구성하는 단위나 성분 파악
    - 텍스트 문서에서 주제 추출

▪ 군집 알고리즘

- 데이터를 비슷한 것끼리 그룹으로 묶는 것
- 소셜 미디어 사이트에 올린 사진들을 동일 사람으로 묶기 위해서 사용

▪ 비지도 학습의 특징

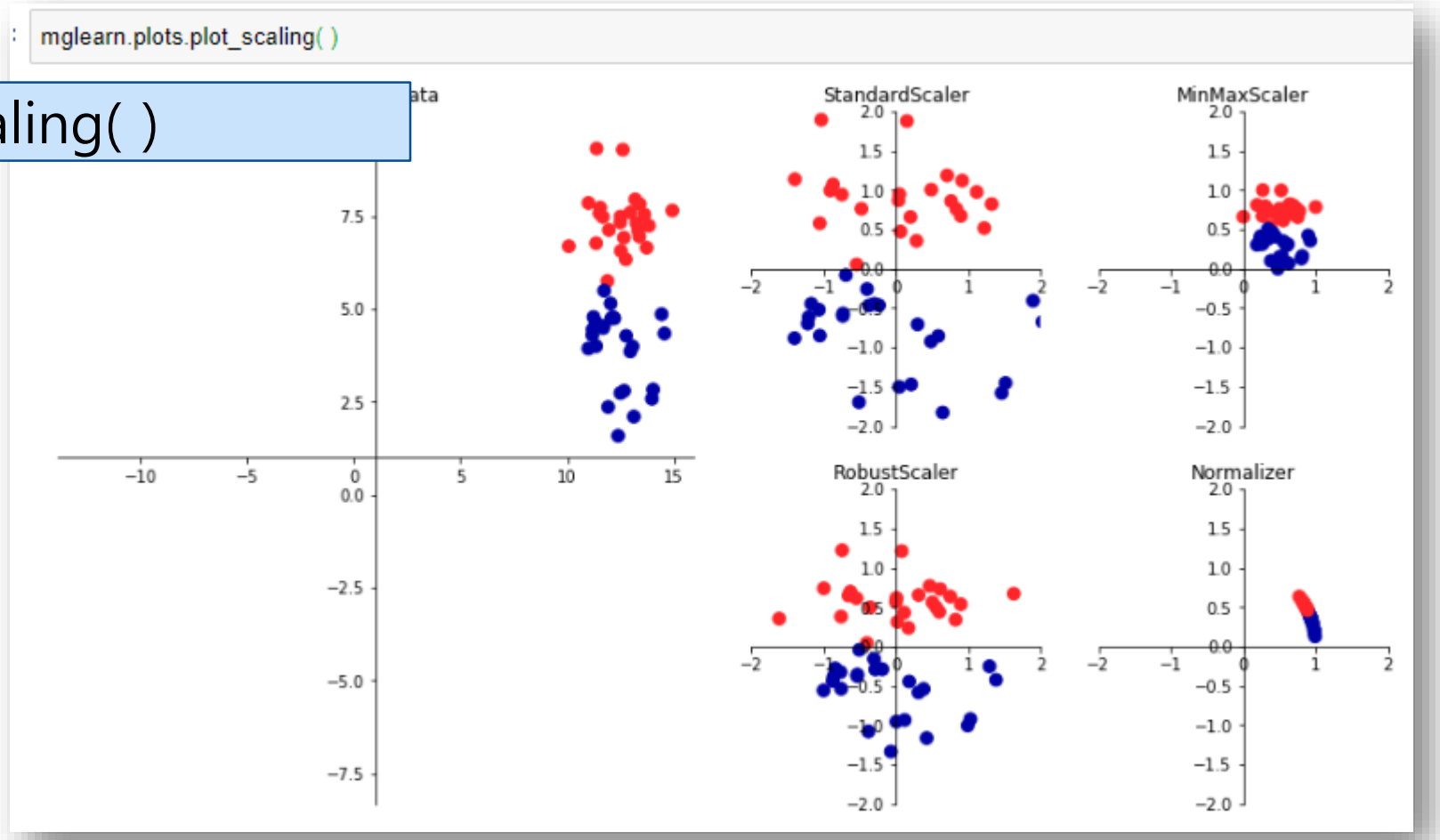
- 비지도 학습의 가장 어려운 점은 알고리즘이 무엇을 학습했는지 평가하는 것
  - 정해진 답이 없기 때문에 무엇이 올바른 결과인지 모른다
  - 군집 알고리즘이 얼굴 사진을 앞면과 옆면으로 분류할 수 있는데, 우리가 원하는 방법이 아니다 → 그러나 알고리즘에 우리가 원하는 것을 알려줄 수 없다
- 이러한 이유로 비지도 학습은 알고리즘은 데이터 사이언티스트가 데이터를 잘 이해하고 싶을 때 탐색적 분석 단계에서 많이 사용한다
  - 지도 학습의 전처리 단계에서도 사용한다

2. 데이터 전처리와 스케일 조정

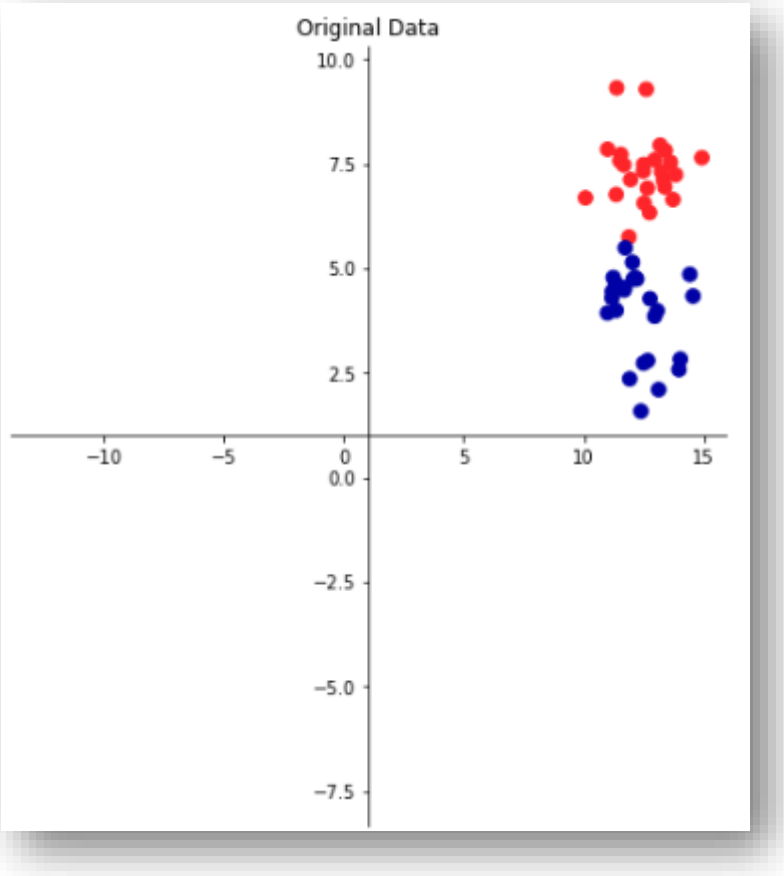
▪ 전처리 방법

◦ 피처 값 조정 사례

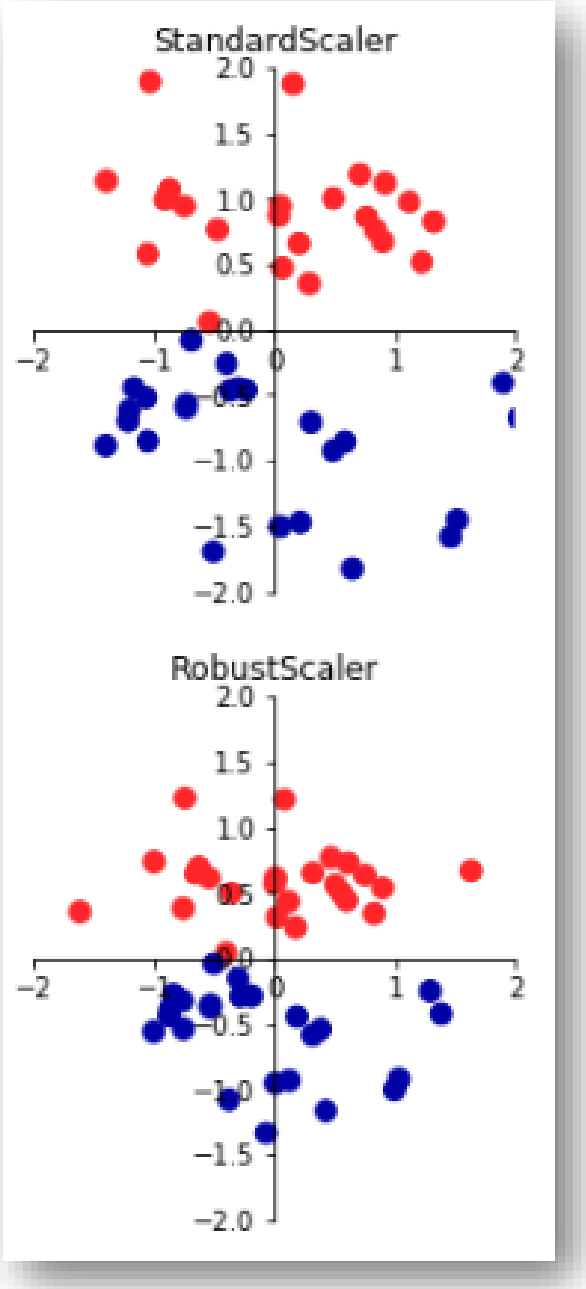
```
mglearn.plots.plot_scaling( )
```



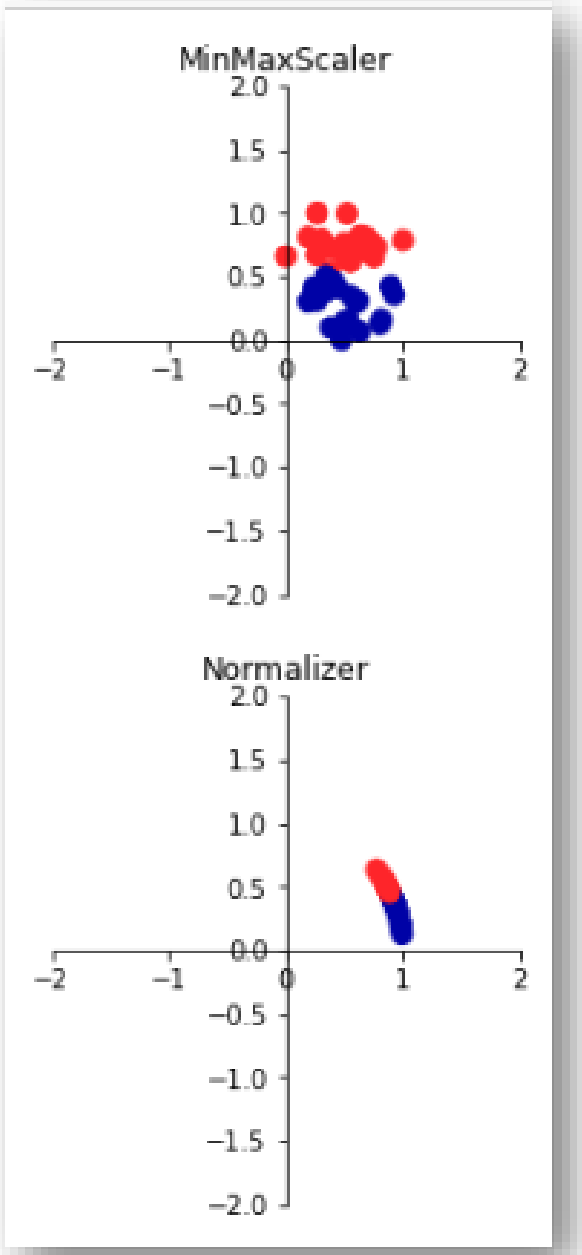
- 오른쪽 그림은 2개의 피처를 인위적으로 만든 이진 분류 데이터 세트이다
  - 첫번째 피처(x 축 값)는 10~15사이
  - 두번째 피처(y 축 값)는 1~9 사이



- 기준에 따라 4가지로 변환(정규화)
  - StandardScaler
    - 각 피처의 평균을 0, 분산을 1로 변경하여 모든 피처를 정규화시킨다(z-값)
  - RobustScaler
    - StandardScaler와 비슷하지만 평균과 분산 대신에 중간 값과 사분위 값을 사용(이상치에 영향을 받지 않는다)



- MinMaxScaler
  - 모든 피처가 0~1 사이에 위치하도록 데이터를 변경한다
  - x 축의 0~1 사이, y 축의 0~1 사이에 위치한다
- Normalizer
  - 피처 벡터의 유클리디안 길이가 1이 되도록 데이터 포인트를 조정한다(지름이 1인 원)
  - 각 데이터 포인트가 다른 비율로 조정된다
  - 데이터의 방향만 중요할 때 사용한다



3. 데이터 변환 적용

■ 적용 절차

- 사용 데이터 세트: cancer 데이터
- 커널 SVM 적용
- MinMaxScaler 사용

- 학습용과 평가용 데이터 세트로 구분
  - load_breast_cancer 데이터 세트 import
  - train_test_split 함수 import
  - 데이터 세트를 cancer 이름으로 저장

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
cancer = load_breast_cancer( )

X_train, X_test, y_train, y_test = train_test_split(cancer.data,
cancer.target, random_state=1)

print(X_train.shape)
print(X_test.shape)
```

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
cancer = load_breast_cancer( )

X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, random_state=1)

print(X_train.shape)
print(X_test.shape)
```

```
(426, 30)
(143, 30)
```

- 학습용 데이터 세트 크기는 행 426 열 30(피처 수)
- 평가용 데이터 세트 크기는 행 143, 열 30(피처 수)

- preprocessing 모듈에서 MinMaxScaler import후 객체 생성

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

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

- fit 메서드를 적용하여 학습용 데이터 훈련
  - MinMaxScaler의 fit 메서드는 각 피처의 최소값과 최대값 계산

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

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

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

- 학습 결과를 적용하여 학습 데이터 스케일 조정
  - transform 메서드 사용

```
X_train_scaled = scaler.transform(X_train)

print(“변환된 후 크기: “, X_train_scaled.shape)
print(“스케일 조정 전 피처별 최소값:\n”, X_train.min(axis=0))
print(“스케일 조정 전 피처별 최대값:\n”, X_train.max(axis=0))
print(“스케일 조정 후 피처별 최소값:\n”, X_train_scaled.min(axis=0))
print(“스케일 조정 후 피처별 최대값:\n”, X_train_scaled.max(axis=0))
```


```
X_train_scaled = scaler.transform(X_train)

print("변환된 후 크기: ", X_train_scaled.shape)
print("스케일 조정 전 피처별 최소값:\n", X_train.min(axis=0))
print("스케일 조정 전 피처별 최대값:\n", X_train.max(axis=0))
print("스케일 조정 후 피처별 최소값:\n", X_train_scaled.min(axis=0))
print("스케일 조정 후 피처별 최대값:\n", X_train_scaled.max(axis=0))
```

```
변환된 후 크기:  (426, 30)
스케일 조정 전 피처별 최소값:
 [6.981e+00 9.710e+00 4.379e+01 1.435e+02 5.263e-02 1.938e-02 0.000e+00
 0.000e+00 1.060e-01 5.024e-02 1.153e-01 3.602e-01 7.570e-01 6.802e+00
 1.713e-03 2.252e-03 0.000e+00 0.000e+00 9.539e-03 8.948e-04 7.930e+00
 1.202e+01 5.041e+01 1.852e+02 7.117e-02 2.729e-02 0.000e+00 0.000e+00
 1.566e-01 5.521e-02]
스케일 조정 전 피처별 최대값:
 [2.811e+01 3.928e+01 1.885e+02 2.501e+03 1.634e-01 2.867e-01 4.268e-01
 2.012e-01 3.040e-01 9.575e-02 2.873e+00 4.885e+00 2.198e+01 5.422e+02
 3.113e-02 1.354e-01 3.960e-01 5.279e-02 6.146e-02 2.984e-02 3.604e+01
 4.954e+01 2.512e+02 4.254e+03 2.226e-01 9.379e-01 1.170e+00 2.910e-01
 5.774e-01 1.486e-01]
스케일 조정 후 피처별 최소값:
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0.]
스케일 조정 후 피처별 최대값:
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1.]
```

◦ SVM 적용을 위해 평가용 데이터 세트 변환

```
X_test_scaled = scaler.transform(X_test)

print(“스케일 조정 후 피처별 최소값:\n”, X_test_scaled.min(axis=0))
print(“스케일 조정 후 피처별 최대값:\n”, X_test_scaled.max(axis=0))
```


```
X_test_scaled = scaler.transform(X_test)

print("스케일 조정 후 피처별 최소값:\n", X_test_scaled.min(axis=0))
print("스케일 조정 후 피처별 최대값:\n", X_test_scaled.max(axis=0))
```

```
스케일 조정 후 피처별 최소값:
 [ 0.0336031   0.0226581   0.03144219  0.01141039  0.14128374  0.04406704
  0.          0.          0.1540404  -0.00615249 -0.00137796  0.00594501
  0.00430665  0.00079567  0.03919502  0.0112206   0.          0.
 -0.03191387  0.00664013  0.02660975  0.05810235  0.02031974  0.00943767
  0.1094235   0.02637792  0.          0.         -0.00023764 -0.00182032]
스케일 조정 후 피처별 최대값:
 [0.9578778  0.81501522 0.95577362 0.89353128 0.81132075 1.21958701
 0.87956888 0.9333996  0.93232323 1.0371347  0.42669616 0.49765736
 0.44117231 0.28371044 0.48703131 0.73863671 0.76717172 0.62928585
 1.33685792 0.39057253 0.89612238 0.79317697 0.84859804 0.74488793
 0.9154725  1.13188961 1.07008547 0.92371134 1.20532319 1.63068851]
```

- 조정한 평가용 세트의 최소값과 최대값이 학습용 데이터 세트와 달리 0~1 이 아니다
  - 항상 학습용과 평가용 데이터 세트가 같은 변환을 적용해야 한다
  - (여기서는) transform 메서드는 평가용 데이터 세트의 최소값 범위를 사용하지 않고 학습용 데이터 세트의 최소값을 빼고 훈련용 데이터 세트의 범위로 나누어져 최대값과 최소값 범위가 다르다

4. 학습용과 평가용 데이터 세트 스케일을 같은 방법으로 조정

▪ 평가용 데이터 세트에 최소값과 범위 사용

◦ make_blobs 함수를 이용하여 무작위 데이터 세트 생성한다

```
from sklearn.datasets import make_blobs
X, _ = make_blobs(n_samples=50, centers=5, random_state=4, cluster_std = 2)
```

◦ 학습용과 평가용 데이터 세트로 나눈다

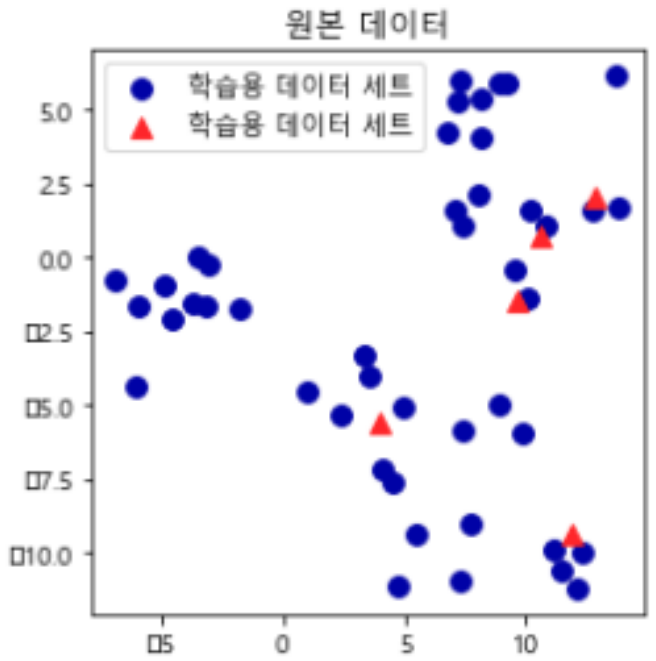
```
X_train, X_test = train_test_split(X, random_state=5, test_size=.1)
```

- 원본 산점도 그리기
  - plt.subplots( ) 함수
    - fig = plt.figure( ) 와 axes = fig.subplots( ) 합한 것
    - 화면을 너비 13, 높이 4를 만들어 1 * 3으로 나누기
  - axes[0] : 첫번째 그림

```
fig, axes = plt.subplots(1, 3, figsize = (13, 4))
axes[0].scatter(X_train[:,0], X_train[:, 1], c=mglearn.cm2.colors[0],
          label = "학습용 데이터 세트", s=60)
axes[0].scatter(X_test[:,0], X_test[:, 1], marker='^',
          c=mglearn.cm2.colors[1], label = "학습용 데이터 세트", s=60)
axes[0].legend(loc='upper left')
axes[0].set_title("원본 데이터")
```

```
fig, axes = plt.subplots(1, 3, figsize = (13, 4))
axes[0].scatter(X_train[:,0], X_train[:, 1], c=mglearn.cm2.colors[0], label = "학습용 데이터 세트", s=60)
axes[0].scatter(X_test[:,0], X_test[:, 1], marker='^', c=mglearn.cm2.colors[1], label = "학습용 데이터 세트", s=60)
axes[0].legend(loc='upper left')
axes[0].set_title("원본 데이터")
```

```
Text(0.5, 1.0, '원본 데이터')
```



◦ MinMaxScaler 사용해 스케일 조정

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

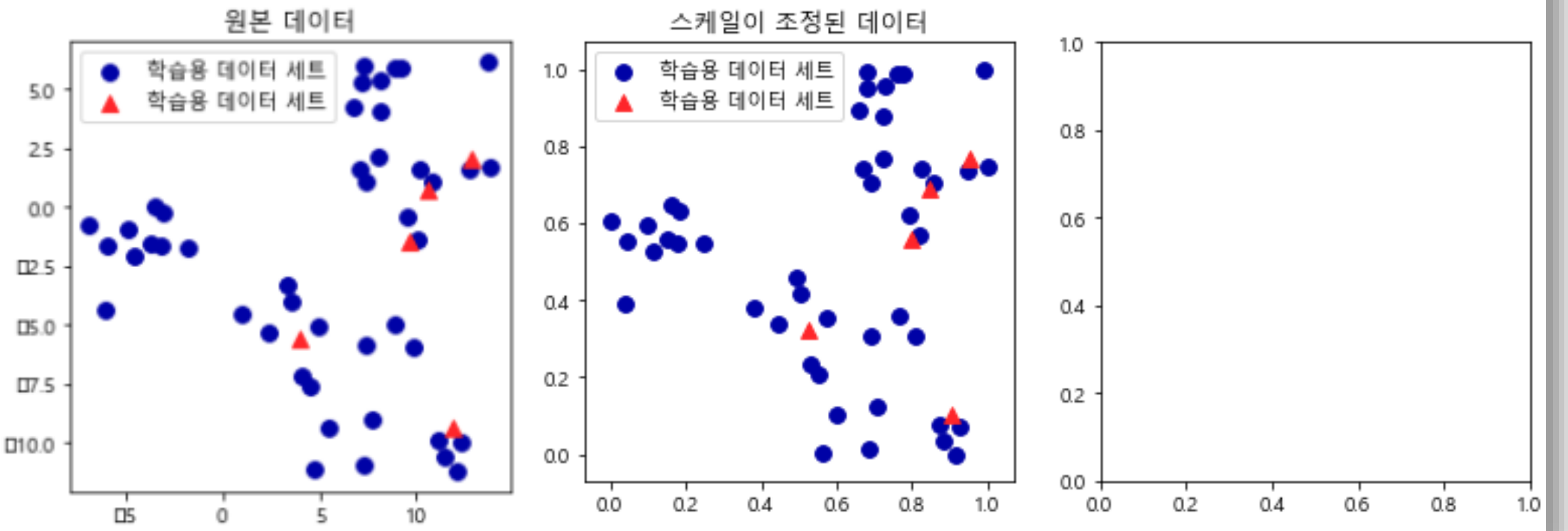

```
scaler = MinMaxScaler( )
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

◦ 스케일이 조정된 산점도 그리기

```
fig, axes = plt.subplots(1, 3, figsize = (13, 4))
axes[1].scatter(X_train_scaled[:,0], X_train_scaled[:, 1], c=mglearn.cm2.colors[0],
        label = "학습용 데이터 세트", s=60)
axes[1].scatter(X_test_scaled[:,0], X_test_scaled[:, 1], marker='^',
        c=mglearn.cm2.colors[1], label = "학습용 데이터 세트", s=60)
axes[1].legend(loc='upper left')
axes[1].set_title("스케일이 조정된 데이터")
```

```
fig, axes = plt.subplots(1, 3, figsize = (13, 4))
axes[1].scatter(X_train_scaled[:,0], X_train_scaled[:, 1], c=mglearn.cm2.colors[0],
        label = "학습용 데이터 세트", s=60)
axes[1].scatter(X_test_scaled[:,0], X_test_scaled[:, 1], marker='^',
        c=mglearn.cm2.colors[1], label = "학습용 데이터 세트", s=60)
axes[1].legend(loc='upper left')
axes[1].set_title("스케일이 조정된 데이터")
```

```
Text(0.5, 1.0, '스케일이 조정된 데이터')
```

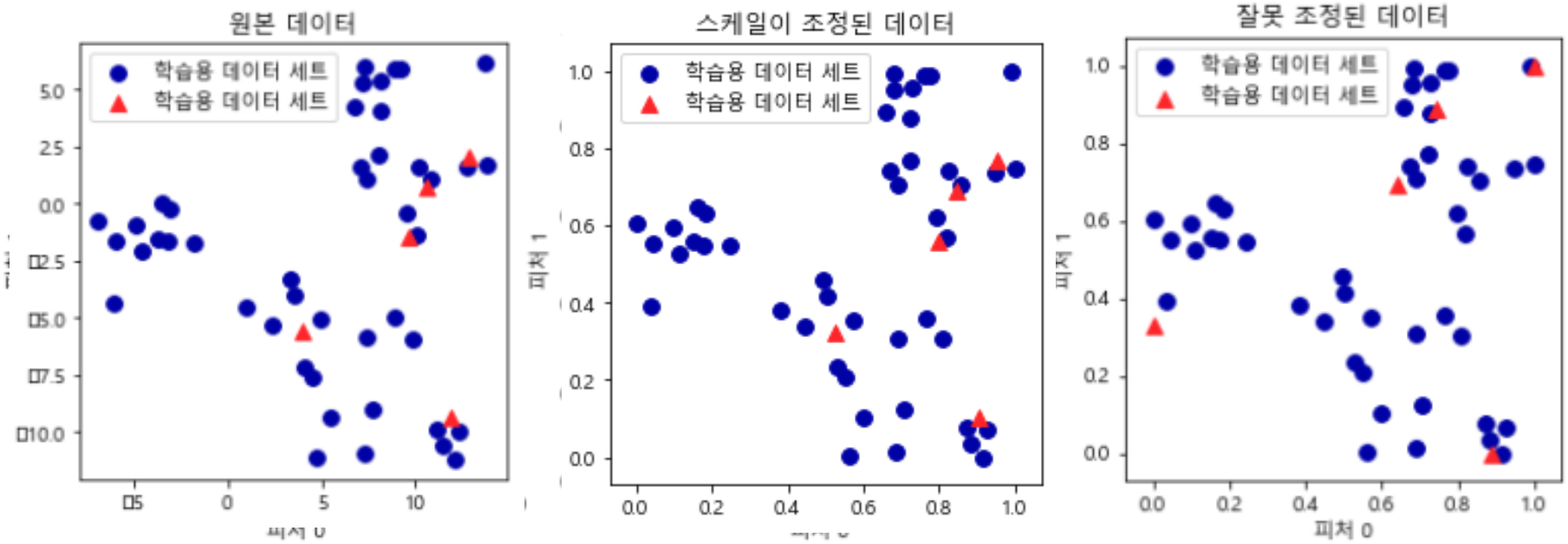


◦ <잘못된 방법> 잘못 조정된 산점도 그리기

```
test_scaler = MinMaxScaler( )
test_scaler.fit(X_test)
X_test_scaled_badly = test_scaler.transform(X_test)
fig, axes = plt.subplots(1, 3, figsize = (13, 4))
axes[2].scatter(X_train_scaled[:,0], X_train_scaled[:, 1], c=mglearn.cm2.colors[0],
        label = "학습용 데이터 세트", s=60)
axes[2].scatter(X_test_scaled_badly[:,0], X_test_scaled_badly[:, 1], marker='^',
        c=mglearn.cm2.colors[1], label = "학습용 데이터 세트", s=60)
axes[2].legend(loc='upper left')
axes[2].set_title("잘못 조정된 데이터")
for ax in axes:
ax.set_xlabel("피처 0")
ax.set_ylabel("피처 1")
```

```
test_scaler = MinMaxScaler( )
test_scaler.fit(X_test)
X_test_scaled_badly = test_scaler.transform(X_test)
fig, axes = plt.subplots(1, 3, figsize = (13, 4))
axes[2].scatter(X_train_scaled[:,0], X_train_scaled[:, 1], c=mglearn.cm2.colors[0],
        label = "학습용 데이터 세트", s=60)
axes[2].scatter(X_test_scaled_badly[:,0], X_test_scaled_badly[:, 1], marker='^',
        c=mglearn.cm2.colors[1], label = "학습용 데이터 세트", s=60)
axes[2].legend(loc='upper left')
axes[2].set_title("잘못 조정된 데이터")
for ax in axes:
    ax.set_xlabel("피처 0")
    ax.set_ylabel("피처 1")
```

- 원본 데이터 그림의 눈금과 조정된 데이터 그림 눈금이 다르다
- 스케일이 조정된 데이터 그림과 잘못 조정된 데이터 그림의 차이점 확인이 필요하다



5. 지도 학습에서 데이터 전처리 효과

▪ SVM 분석에서 MinMaxScaler 효과

◦ 비교를 위해 원본 데이터로 SVM 모델 만들기

```
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("평가용 데이터 세트 정확도: {:.2f}".format(svm.score(X_test, y_test)))
```

```
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("평가용 데이터 세트 정확도: {:.2f}".format(svm.score(X_test, y_test)))
```

```
평가용 데이터 세트 정확도: 0.63
```

◦ MinMaxScaler 사용해 데이터 스케일 조정 후 SVM 모델 학습

```
scaler = MinMaxScaler( )
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
svm.fit(X_train_scaled, y_train)
print("스케일 조정된 평가용 데이터 세트 정확도:{:.2f}"
                        .format(svm.score(X_test_scaled, y_test)))
```

```
scaler = MinMaxScaler( )
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
svm.fit(X_train_scaled, y_train)
print("스케일 조정된 평가용 데이터 세트 정확도:{:.2f}".format(svm.score(X_test_scaled, y_test)))
```

```
스케일 조정된 평가용 데이터 세트 정확도:0.97
```

◦ scikit-learn 에서 제공하는 도구 사용 방법

```
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)
svm.fit(X_train_scaled, y_train)
print("StandardScaler로 조정된 평가용 데이터 세트 정확도:{:.2f}"
                              .format(svm.score(X_test_scaled, y_test)))
```

```
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)
svm.fit(X_train_scaled, y_train)
print("StandardScaler로 조정된 평가용 데이터 세트 정확도:{:.2f}".format(svm.score(X_test_scaled, y_test)))
```

```
StandardScaler로 조정된 평가용 데이터 세트 정확도:0.96
```

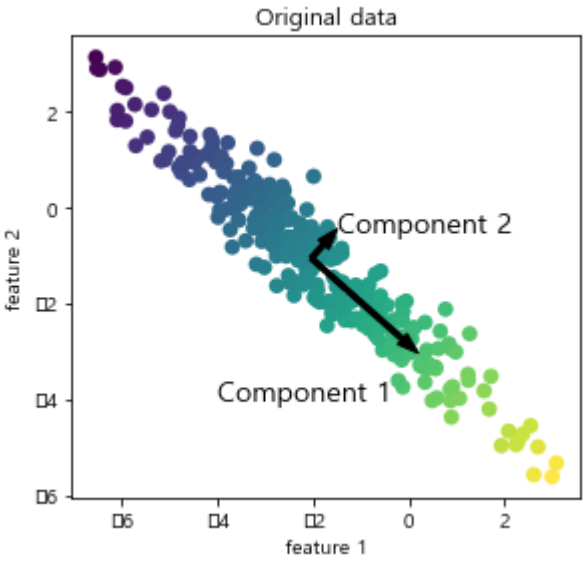
6. 주성분 분석(PCA)

▪ 특징

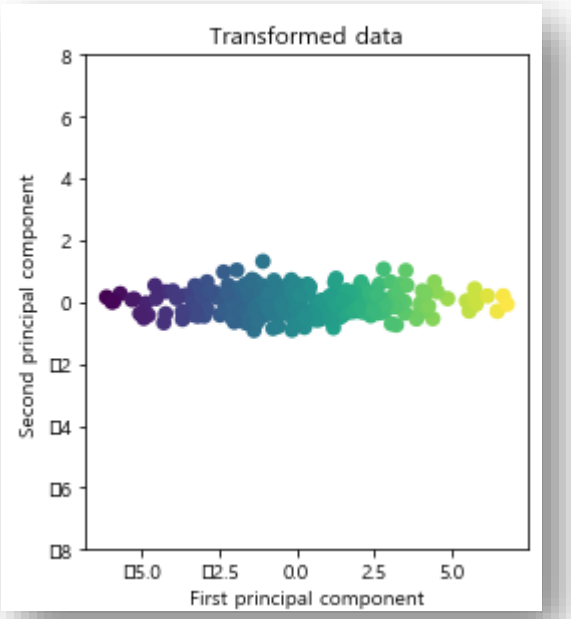
- 피처들이 통계적으로 상관관계가 없도록 데이터 세트를 회전시키는 기술이다

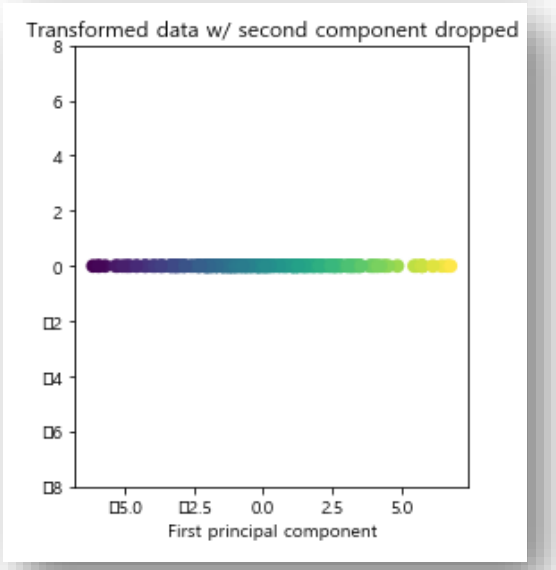
```
mglearn.plots.plot_pca_illustration( )
```

  - 오른쪽 그림에서 알고리즘은 먼저 component 1(성분 1), 분산이 가장 큰 방향을 찾는다
    - 이 방향이 이 데이터에서 가장 많은 정보를 담고 있는 방향이다(피처들의 상관관계가 가장 큰 방향)
  - 다음으로 첫번째 방향과 직각인 방향 중에서 가장 많은 정보를 담은 방향을 찾는다

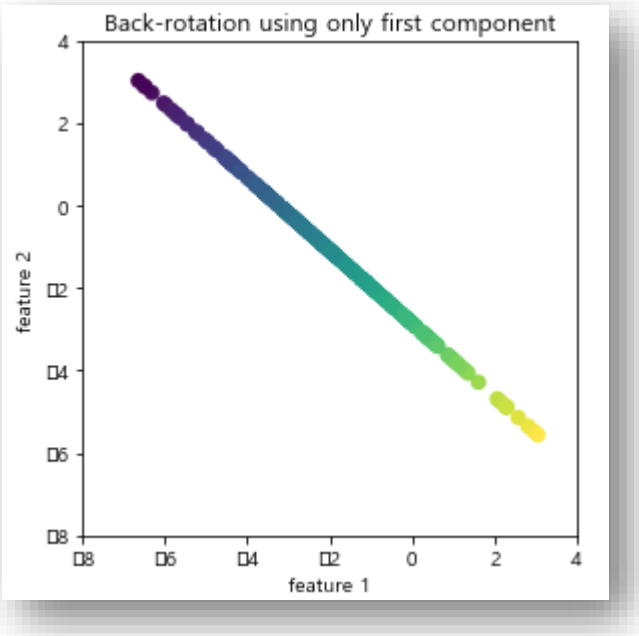


- 주성분1을 x 축과 평형하도록 회전했다
  - 회전하기 전에 데이터 평균을 그래프 가운데 맞추었다
  - x축과 y축이 서로 연관관계가 없으므로 변환된 데이터는 주성분 1과 주성분 2 사이에 상관관계가 없다
- 아래 그림은 2번째 주성분을 제거한 것이다
  - 2차원 데이터 세트이 1차원 데이터 세트로 차원이 감소한다
  - 이는 가장 유용한 방향을 찾아 첫번째 주성분만 유지한다





- 데이터에 다시 평균을 더하고 반대로 회전시킨다
- 데이터 포인트가 원래 공간에 있지만 첫번째 주성분의 정보만 담고있다
  - 이 방법은 데이터에서 노이즈를 제거하거나 주성분에서 유지되는 정보를 시각화하는데 사용된다



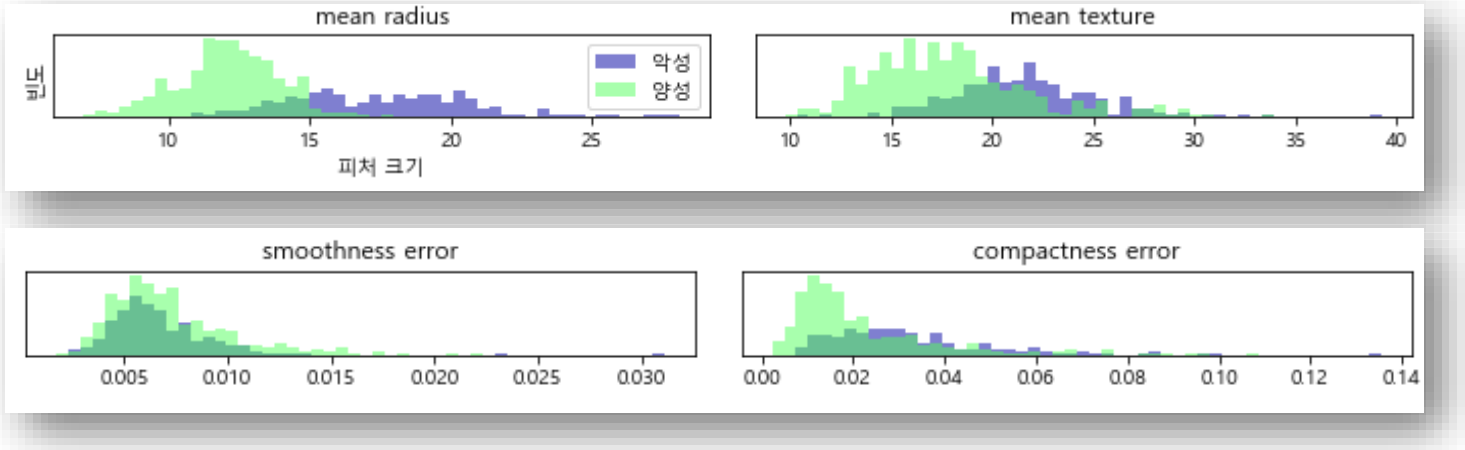
cancer 데이터 세트 시각화하기

- PCS가 가장 많이 적용되는 분야는 고차원 데이터 세트의 시각화이다
- cancer 데이터 세트는 피처가 30개가 있어서 산점도 행렬을 그리려면 30 x 14.5 = 435개의 산점도를 그려야 한다
- 쉬운 방법은 양성과 악성 두 클래스에 대한 히스토그램을 그리는 것이다

```
fig, axes = plt.subplots(15, 2, figsize = (10, 20))
malignant = cancer.data[cancer.target == 0]
benign = cancer.data[cancer.target ==1]

ax = axes.ravel( )

for i in range(30):
        _, bins = np.histogram(cancer.data[:, i], bins=50)
        ax[i].hist(malignant[:, i], bins=bins, color=mglearn.cm3(0), alpha=.5)
        ax[i].hist(benign[:, i], bins=bins, color=mglearn.cm3(2), alpha=.5)
        ax[i].set_title(cancer.feature_names[i])
        ax[i].set_yticks(( ))
ax[0].set_xlabel("피처 크기")
ax[0].set_ylabel("빈도")
ax[0].legend(["악성", "양성"], loc="best")
fig.tight_layout( )
```



- 각 그래프는 악성 양성 그래프를 겹쳐 놓은 것이다
- 피처 중 악성과 양성이 명확하게 분리되어 있으면, 피처로 유용하다
- 그러나, 피처 간의 상호작용이나 상호작용이 악성 혹은 양성과 어떤 관련이 있는지 알려주지 않는다
- PCA를 통하여 2개의 주성분을 찾고 2차원 공간에 하나의 산점도로 표시한다

- PCA 적용 전에 StandardScaler 적용

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer( )

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

- PCA 적용

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(X_scaled)

X_pca = pca.transform(X_scaled)
print("원본 데이터 형태:", str(X_scaled.shape))
print("축소된 데이터 형태:", str(X_pca.shape))
```

◦ 주성분을 2개로 지정하였다

◦ 적용 결과 피처가 2개로 축소되었다

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(X_scaled)

X_pca = pca.transform(X_scaled)
print("원본 데이터 형태:", str(X_scaled.shape))
print("축소된 데이터 형태:", str(X_pca.shape))
```

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

- 처음 2개 주성분을 그린다

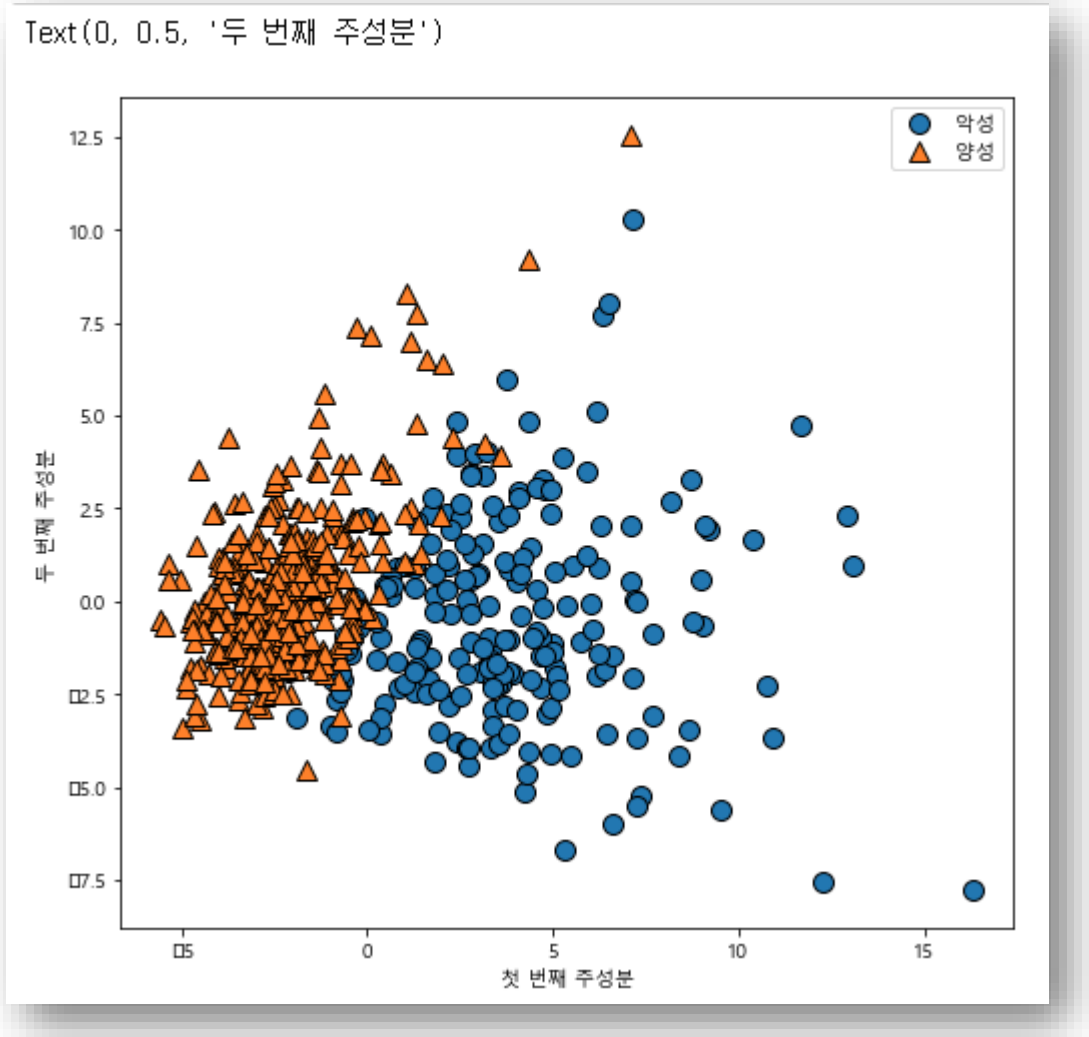
```
plt.figure(figsize=(8, 8))
mglearn.discrete_scatter(X_pca[:, 0], X_pca[:, 1], cancer.target)
plt.legend(["악성", "양성"], loc="best")
plt.gca( ).set_aspect("equal")
plt.xlabel("첫 번째 주성분")
plt.ylabel("두 번째 주성분")
```

x축 y축 비율을 동일하게 만들기


```
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")
```

- PCA는 비지도 학습이므로, 단순히 데이터에 있는 상관관계만 고려한다
- 데이터 포인트는 클래스 정보(악성, 양성)를 이용하여 구분하였다

- 그러나, PCA 단점은 주성분 즉, x 축과 y 축을 해석하기 쉽지 않다
- 주성분은 여러 피처들이 결합된 형태이다



- PCA 주성분 형태

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

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

```
PCA 주성분 형태:  (2, 30)
```

- PCA 주성분

print("PCA 주성분 : ", pca.components_)

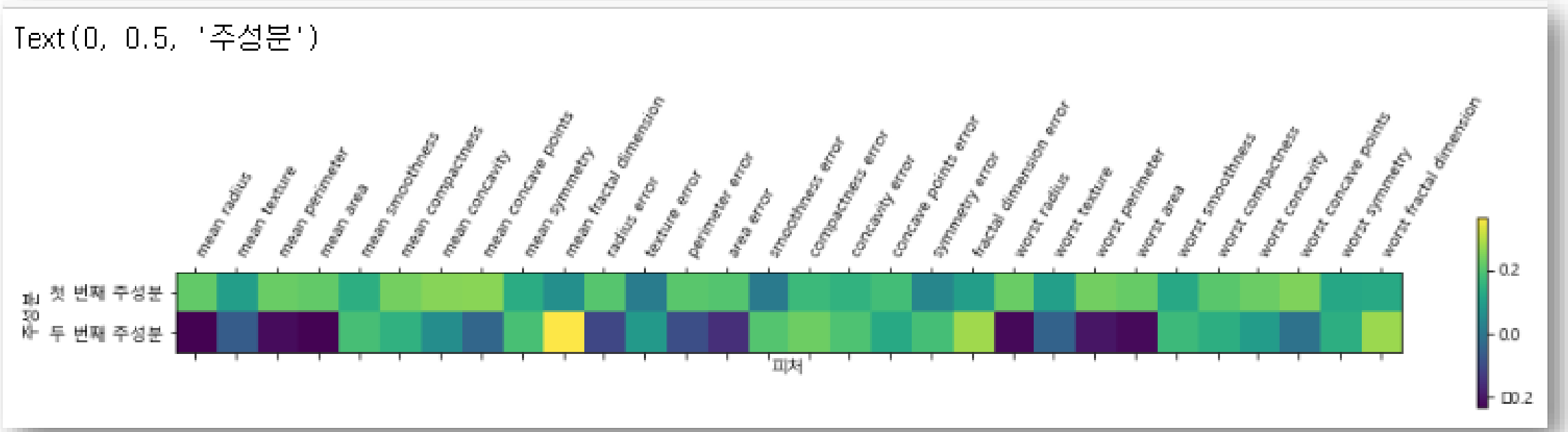
```
print("PCA 주성분: ", pca.components_)
```

```
PCA 주성분:  [[ 0.21890244  0.10372458  0.22753729  0.22099499  0.14258969
   0.25840048  0.26085376  0.13816696  0.06436335  0.20597878  0.01742803
```

- 히트맵으로 그리기
  - matshow: 평면으로 그리기

```
plt.matshow(pca.components_, cmap='viridis')
plt.yticks([0, 1], ["첫 번째 주성분", "두 번째 주성분"])
plt.colorbar( )
plt.xticks(range(len(cancer.feature_names)),
                    cancer.feature_names, rotation=60, ha='left')
plt.xlabel("피처")
plt.ylabel("주성분")
```


```
plt.matshow(pca.components_, cmap='viridis')
plt.yticks([0, 1], ["첫 번째 주성분", "두 번째 주성분"])
plt.colorbar( )
plt.xticks(range(len(cancer.feature_names)), cancer.feature_names, rotation=60, ha='left')
plt.xlabel("피처")
plt.ylabel("주성분")
```



- 첫 번째 주성분의 모든 피처 부호가 같다
  - 모든 피처 사이에 공통의 상호관계가 있다
  - 한 피처의 값이 커지면 다른 피처 값들도 같이 높아질 것이다
- 두 번째 주성분은 부호가 섞여 있다: 그래서 의미를 설명하기 쉽지 않다

▪ 고유얼굴(eigenface) 피처 추출

- PCA는 피처 추출에도 이용
- 피처 추출은 원본 데이터 표현보다 분석에 적합한 표현을 찾을 수 있다는 가정에서 출발한다
- 이미지 데이터 세트는 피처 추출에 좋은 사례이다
  - 이미지는 적색, 녹색, 청색(RGB)의 강도가 기록된 픽셀로 구성됨
  - 이미지는 수 천개의 픽셀로 이루어지며 함께 모여 있을 때 의미있음
- PCA를 이용하여 LAW 데이터 세트의 얼굴 이미지에서 피처를 추출하는 모델을 만든다
  - 이 데이터 세트는 2000년 초반의 정치인, 가수, 배우, 운동선수의 얼굴이 포함되어 있음

◦ 이미지 크기를 줄이고 흑백 이미지를 사용하였음

```
from sklearn.datasets import fetch_lfw_people
people = fetch_lfw_people(min_faces_per_person=20, resize=0.7)
image_shape = people.images[0].shape

fig, axes = plt.subplots(2, 5, figsize=(15, 8),
        subplot_kw={'xticks' : ( ), 'yticks' : ( )})
for target, image, ax in zip(people.target, people.images, axes.ravel( )):
        ax.imshow(image)
        ax.set_title(people.target_names[target])
```

```
from sklearn.datasets import fetch_lfw_people
people = fetch_lfw_people(min_faces_per_person=20, resize=0.7)
a = people.images[0].shape

fig, axes = plt.subplots(2, 5, figsize=(15, 8), subplot_kw={'xticks' : ( ), 'yticks' : ( )})
for target, image, ax in zip(people.target, people.images, axes.ravel( )):
                    ax.imshow(image)
                    ax.set_title(people.target_names[target])
```

```
Downloading LFW metadata: https://ndownloader.figshare.com/files/5976012
Downloading LFW metadata: https://ndownloader.figshare.com/files/5976009
Downloading LFW metadata: https://ndownloader.figshare.com/files/5976006
Downloading LFW data (~200MB): https://ndownloader.figshare.com/files/5976015
```

◦ LFW 데이터 세트에는 62명의 얼굴을 찍은 이미지 3,023개가 있으며 각 이미지는 87 x 65 픽셀이다

```
print("이미지 수와 각 이미지 크기:", people.images.shape)
print("클래스 수:", len(people.target_names))
```

```
print("이미지 수와 각 이미지 크기:", people.images.shape)
print("클래스 수:", len(people.target_names))
```

```
이미지 수와 각 이미지 크기: (3023, 87, 65)
클래스 수: 62
```

◦ 데이터 세트에 포함된 인명 확인

```
counts = np.bincount(people.target)
for i, (count, name) in enumerate(zip(counts, people.target_names)):
        print("{0:25}{1:3}" .format(name, count), end='      ')
        if (i +1) % 3 ==0:
            print( )
```


```
counts = np.bincount(people.target)
for i, (count, name) in enumerate(zip(counts, people.target_names)):
    print("{0:25}{1:3}" .format(name, count), end='      ')
    if (i +1) % 3 ==0:
        print( )
```

◦ Colin Powell과 George W Bush가 다른 사람보다 많다

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		

◦ 사람마다 50개 이미지만 선택

```
mask = np.zeros(people.target.shape, dtype=np.bool)
for target in np.unique(people.target):
        mask[np.where(people.target == target)[0][:50]] = 1

X_people = people.data[mask]
y_people = people.target[mask]

X_people = X_people / 225.
```

```
mask = np.zeros(people.target.shape, dtype=np.bool)
for target in np.unique(people.target):
    mask[np.where(people.target == target)[0][:50] ]= 1

X_people = people.data[mask]
y_people = people.target[mask]

X_people = X_people / 225.
```

◦ KNeighborsClassifier

```
from sklearn.neighbors import KNeighborsClassifier

X_train, X_test, y_train, y_test = train_test_split(X_people, y_people,
                    stratify=y_people, random_state=0)
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
print("1-최근접 이웃의 테스트 점수: {:.2f}".format(knn.score(X_test, y_test)))
```

```
from sklearn.neighbors import KNeighborsClassifier

X_train, X_test, y_train, y_test = train_test_split(X_people, y_people, stratify=y_people, random_state=0)
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
print("1-최근접 이웃의 테스트 점수: {:.2f}".format(knn.score(X_test, y_test)))
```

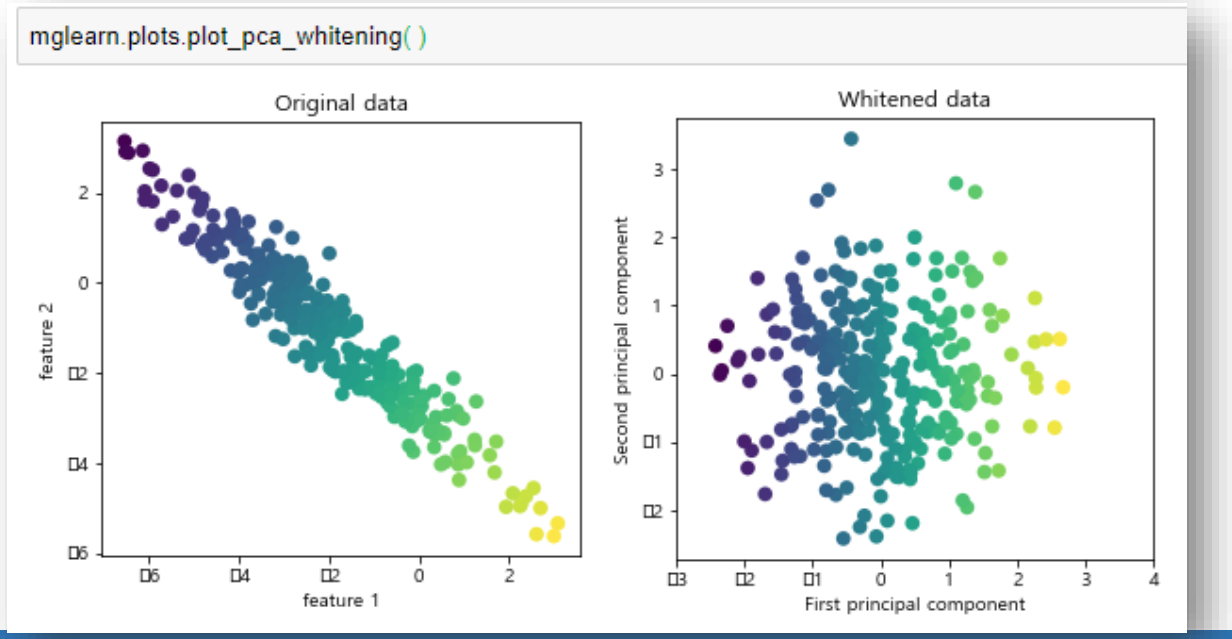
```
1-최근접 이웃의 테스트 점수: 0.23
```

- 정확도가 23%이다. 62가지로 분류할 때(무작위인 경우 1/62 = 1.6%) 보다 높지만 올바른 인식은 ¼에 불과하다
- 그래서 PCA가 필요하다

- PCA 적용 방법
  - 일반적으로 픽셀을 이용해 이미지를 비교할 때, 각 픽셀의 회색톤 값을 다른 이미지에서 동일한 위치에 있는 픽셀 값과 비교한다
    - 하지만, 이 방식은 사람 얼굴 이미지를 인식하는 방법과 많이 다르고, 픽셀을 있는 그대로 비교하는 방식으로는 얼굴의 특징을 찾기 힘들다
  - 그래서 PCA를 적용한다

- PCA 화이트닝 whitening
  - 화이트닝 옵션은 주성분의 스케일이 같아지도록 조정한다
  - 마치 StandardScaler를 적용하는 것과 같다

```
mglearn.plots.plot_pca_whitening( )
```



- PCA를 이용한 학습
  - 학습 데이터세트를 PCA를 이용하여 주성분 100를 추출하도록 학습
  - 그 후, 학습 데이터세트와 평가 데이터세트를 변환시킨다

```
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(“변환된 학습 데이터세트 크기:”, X_train_pca.shape)
```

```
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("변환된 학습 데이터세트 크기:", X_train_pca.shape)
```

```
변환된 학습 데이터세트 크기: (1547, 100)
```

- 다시 최근접 이웃 분류기 적용

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

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

```
kNN을 이용한 평가용 데이터 세트 정확도: 0.31
```

  - 31% 로 증가하여, PCA가 데이터를 잘 표현한다고 볼 수 있다

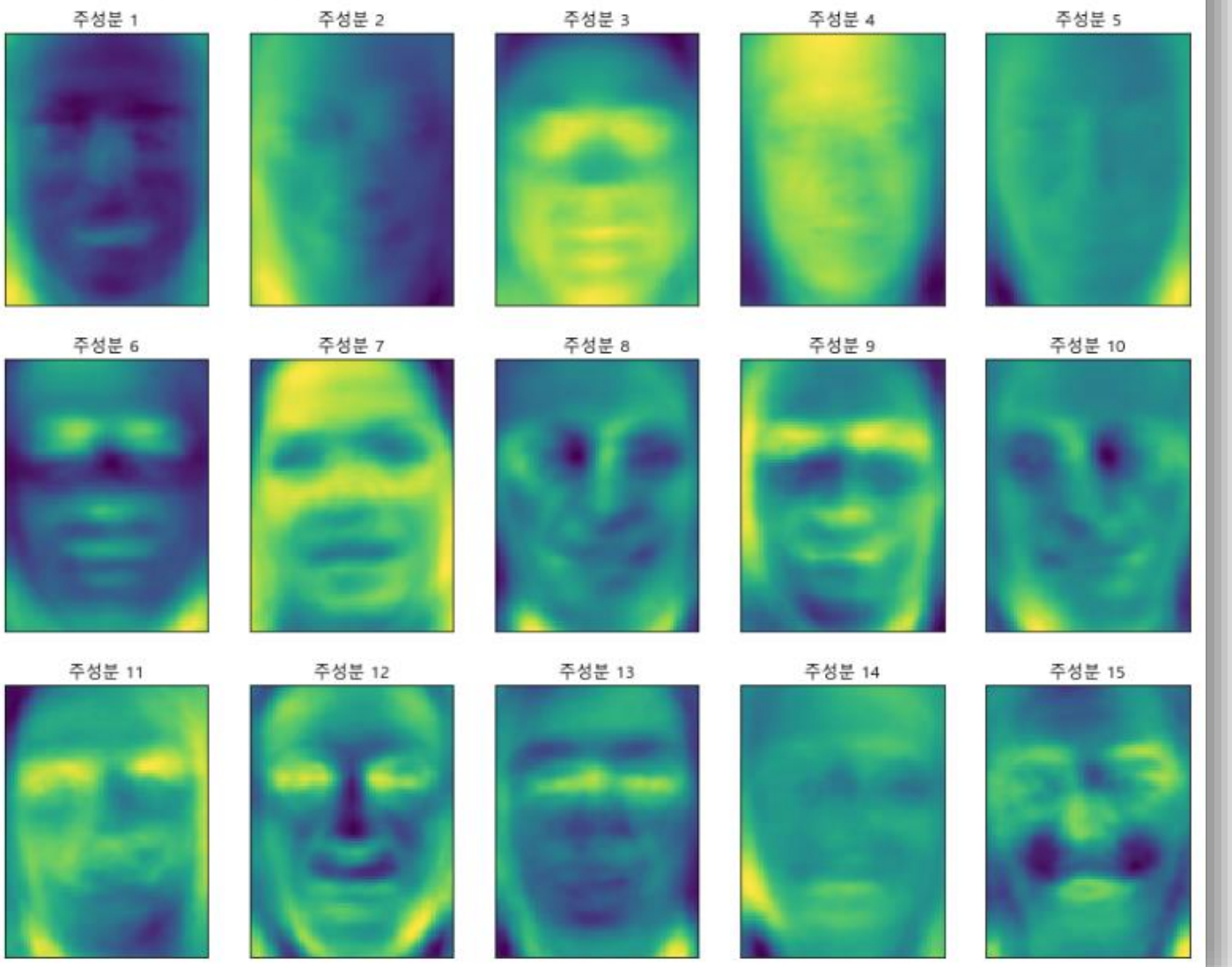
- 주성분 확인

```
print(“pca.components_.shape:”, pca.components_.shape)
```

```
print("pca.components_.shape:", pca.components_.shape)
pca.components_.shape: (100, 5655)
```

```
fig, axes = plt.subplots(3, 5, figsize=(15, 12),
                         subplot_kw={‘xticks’: ( ), ‘yticks’ : ( )})
for i, (component, ax) in enumerate(zip(pca.components_, axes.ravel( ))):
        ax.imshow(component.reshape(image_shape), cmap=‘viridis’)
        ax.set_title(“주성분 {}”.format(i + 1)))
```

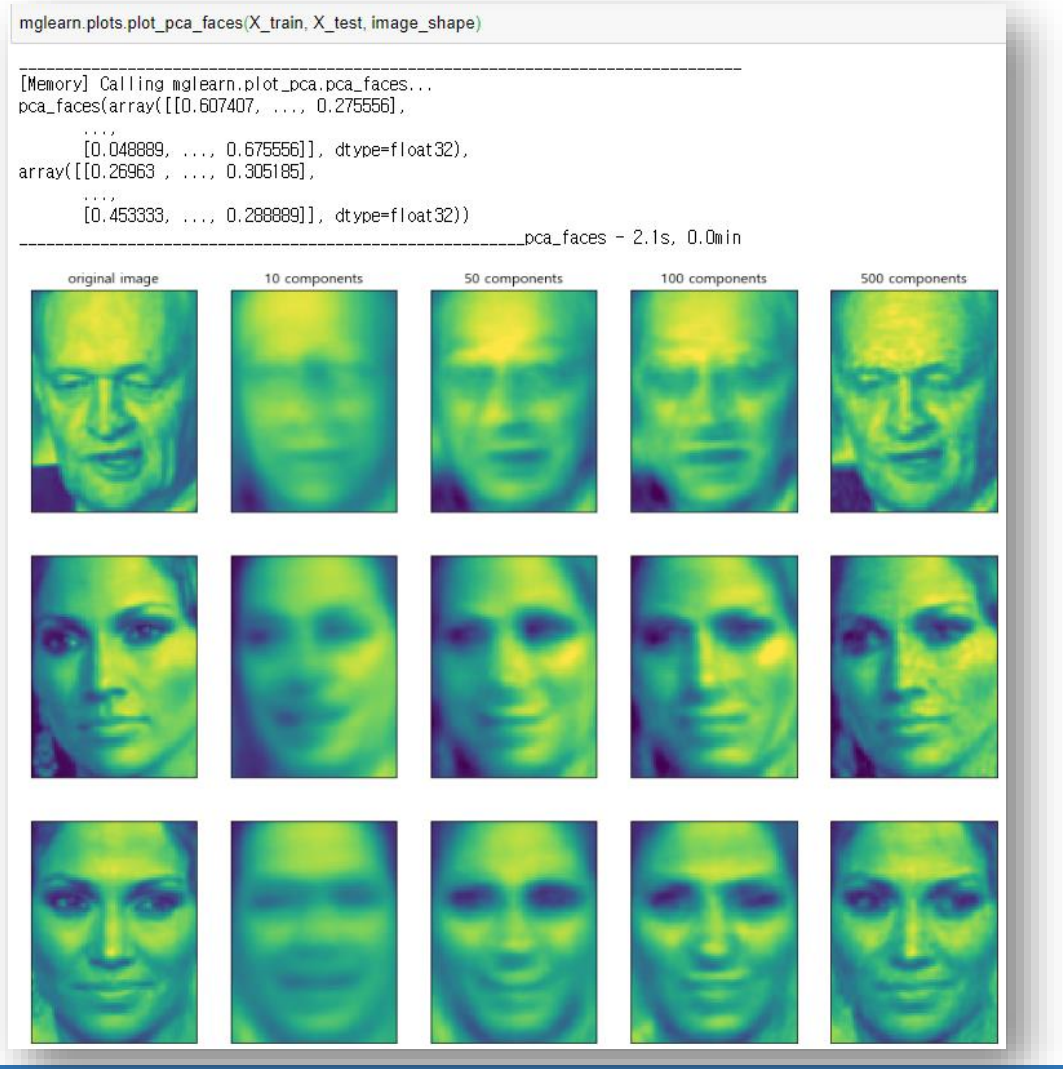
```
fig, axes = plt.subplots(3, 5, figsize=(15, 12),
                         subplot_kw={'xticks': ( ), 'yticks': ( )})
for i, (component, ax) in enumerate(zip(pca.components_, axes.ravel())):
    ax.imshow(component.reshape(image_shape), cmap='viridis')
    ax.set_title("주성분 {}".format((i + 1)))
```



- 주성분 해석
  - 첫번째 주성분은 얼굴과 배경의 명암 차이를 기록한 것으로 보인다
  - 두번째 주성분은 오른쪽과 왼쪽 조명의 차이를 보이고 있는 것 같다
  - 하지만, 이 해석은 얼굴 인식과 거리가 멀다
- PCA 모델은 픽셀을 기반으로 하므로, 얼굴의 배치나 조명이 두 이미지의 유사성 판단에 영향을 미친다
- 하지만, 사람은 얼굴의 유사성을 판단할 때, 성별, 나이, 표정, 머리 모양 등 픽셀의 강도로 판단하기 어려운 속성들을 사용한다 → 알고리즘이 사람의 방식과 다르다

mglearn.plots.plot_pca_faces(X_train, X_test, image_shape)

- 주성분 수를 기준으로 얼굴 이미지를 재구성한 결과이다
- 주성분을 많이 사용할 수록 이미지가 상세해 진다
  - 주성분을 픽셀 수만큼 사용하면 이미지를 완벽하게 재구성할 수 있다



요약

- 비지도학습 종류
- 데이터 전처리와 스케일 조정
- 데이터 변환 적용
- 학습용, 평가용 데이터 스케일을 같은 방법으로 조정
- 지도 학습에서 데이터 전처리 효과
- 주성분 분석(PCA)