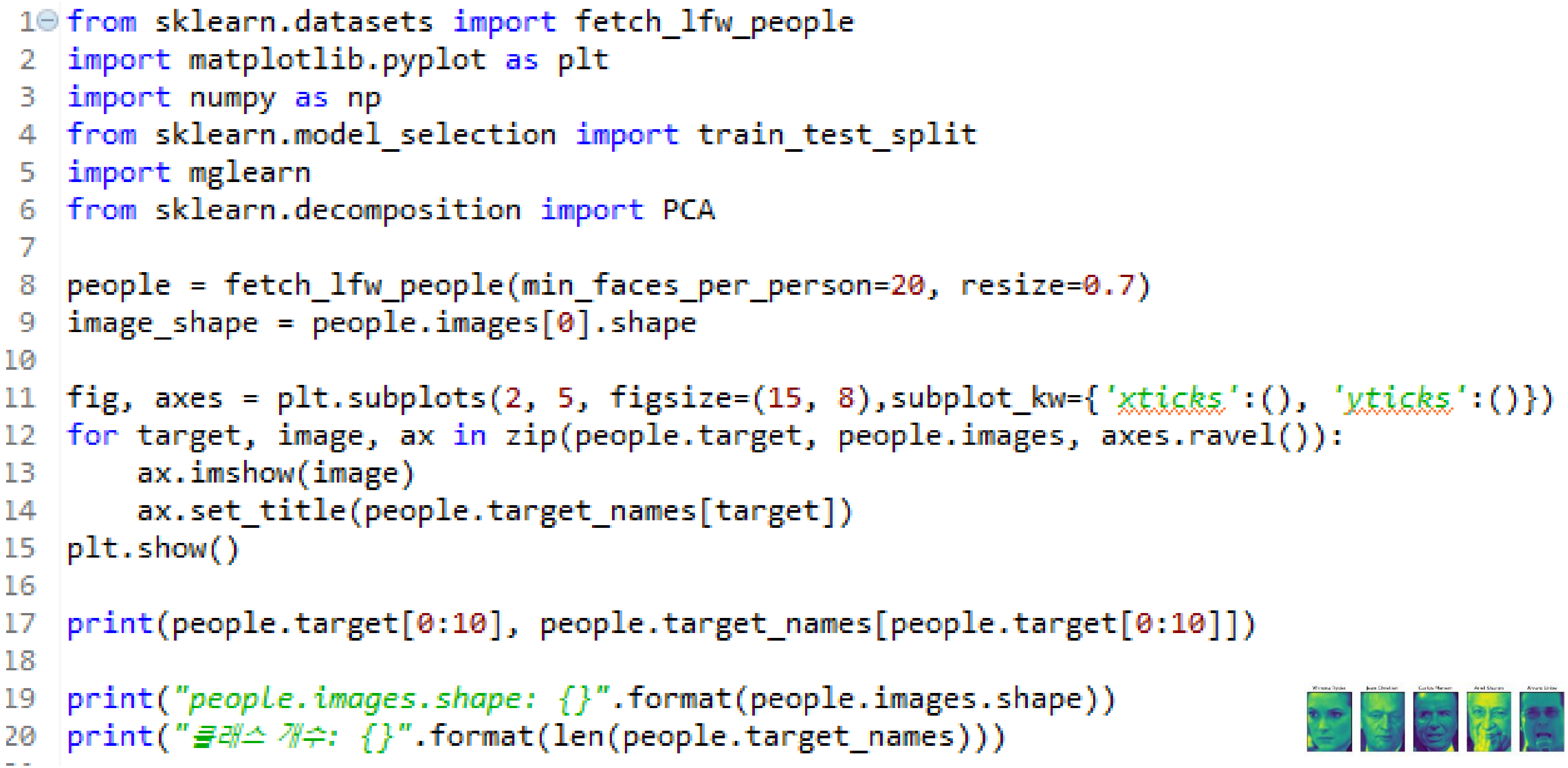
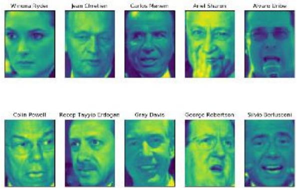
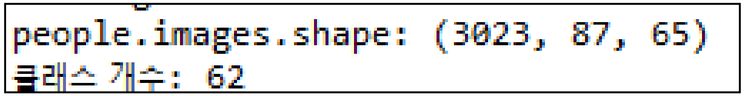
특성추출 예제 – 데이터 다운로드



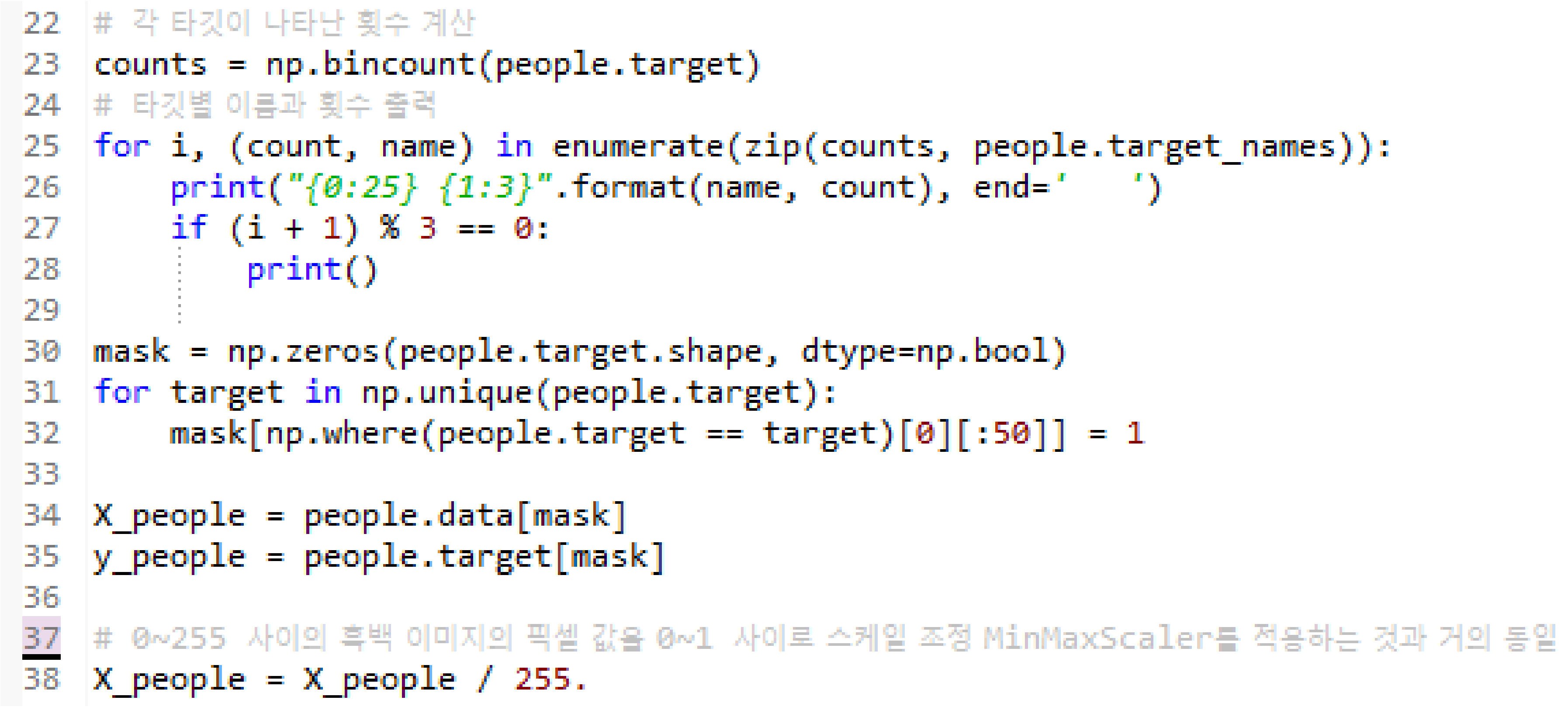
```python
from sklearn.datasets import fetch_lfw_people
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
import mglearn
from sklearn.decomposition import PCA

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])
plt.show()

print(people.target[0:10], people.target_names[people.target[0:10]])

print("people.images.shape: {}".format(people.images.shape))
print("클래스 개수: {}".format(len(people.target_names)))
```



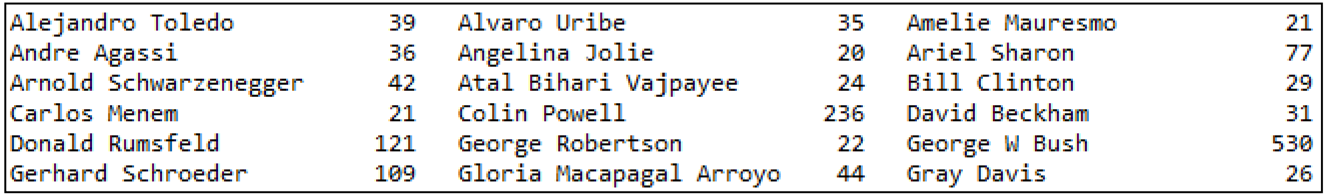


```
people.images.shape: (3023, 87, 65)
클래스 개수: 62
```

특성추출 예제 – 데이터 탐색

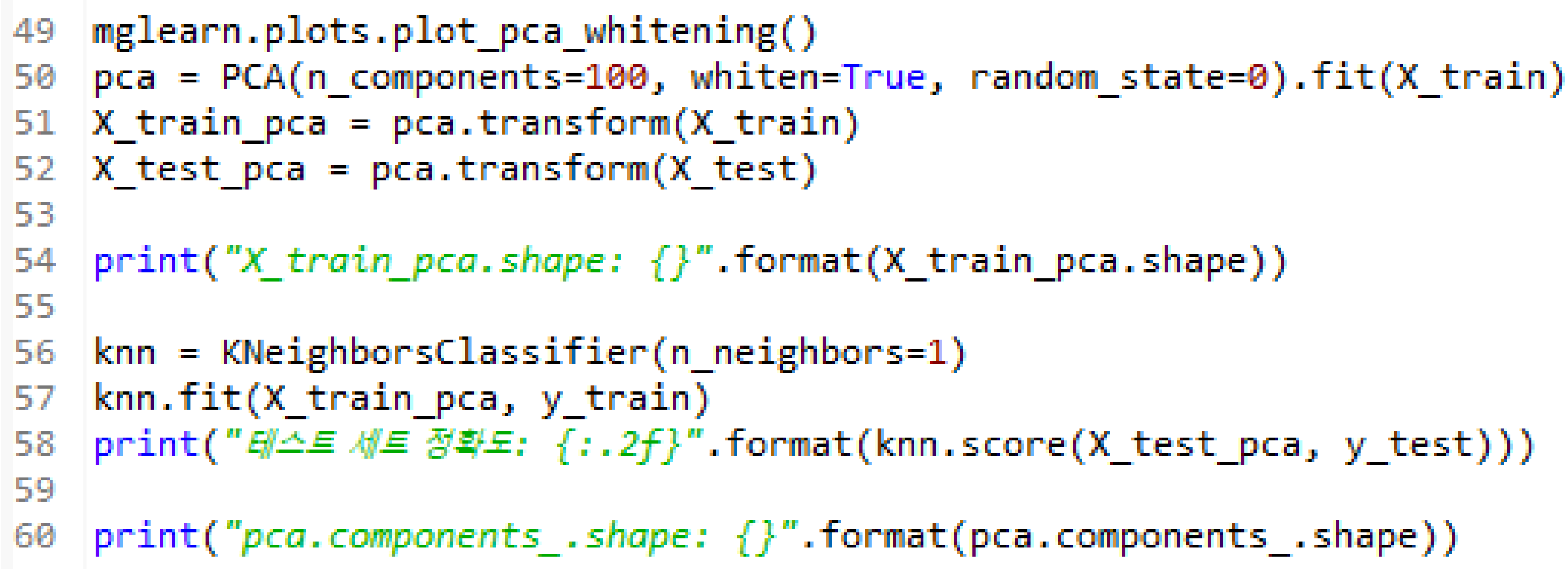


```
# 각 타깃이 나타난 횟수 계산
counts = np.bincount(people.target)
# 타깃별 이름과 횟수 출력
for i, (count, name) in enumerate(zip(counts, people.target_names)):
    print("{0:25} {1:3}".format(name, count), end='   ')
    if (i + 1) % 3 == 0:
        print()

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]

# 0~255 사이의 흑백 이미지의 픽셀 값을 0~1 사이로 스케일 조정 MinMaxScaler를 적용하는 것과 거의 동일
X_people = X_people / 255.
```

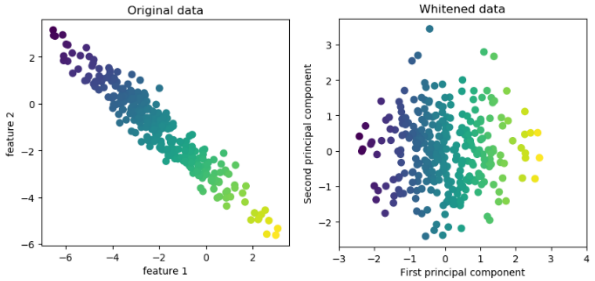


```
Alejandro Toledo           39   Alvaro Uribe               35   Amelie Mauresmo            21
Andre Agassi               36   Angelina Jolie             20   Ariel Sharon               77
Arnold Schwarzenegger      42   Atal Bihari Vajpayee       24   Bill Clinton               29
Carlos Menem               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
```

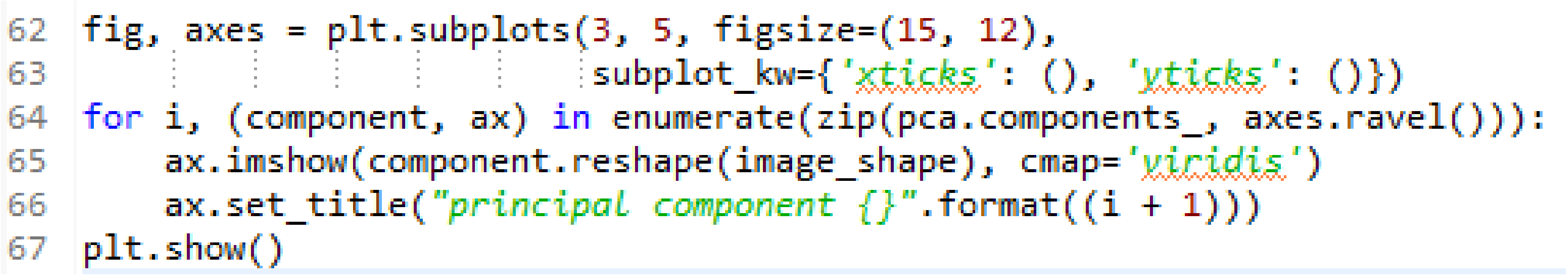
특성추출 – whitening



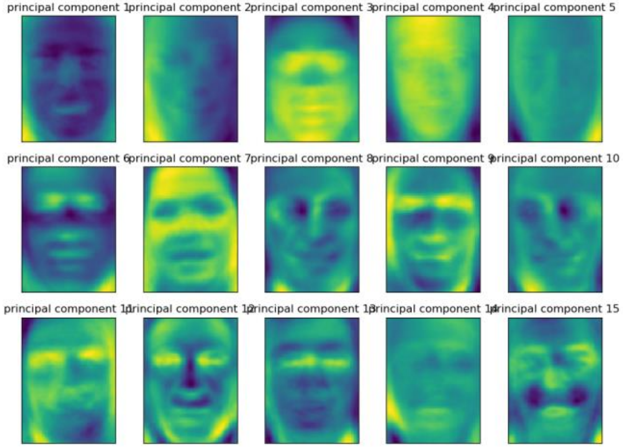
```
mglearn.plots.plot_pca_whitening()
pca = PCA(n_components=100, whiten=True, random_state=0).fit(X_train)
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)

print("X_train_pca.shape: {}".format(X_train_pca.shape))

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)))

print("pca.components_.shape: {}".format(pca.components_.shape))
```



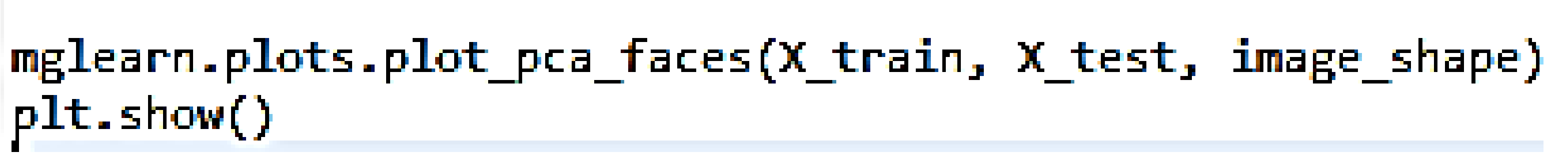
특성추출예제 – 각 주성분 시각화



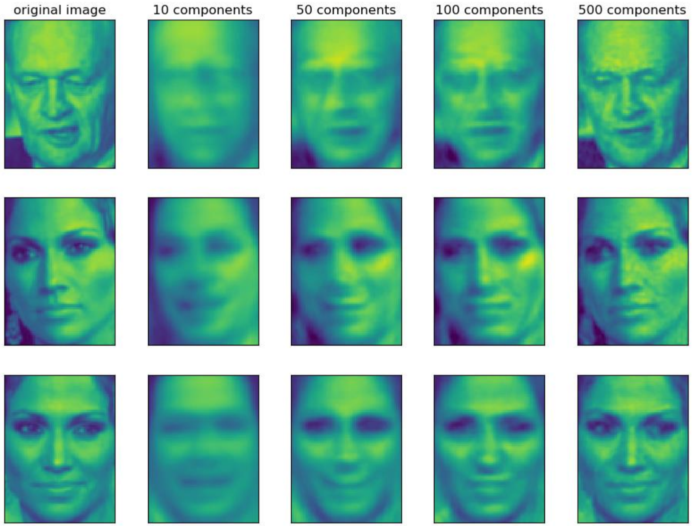
```
fig, axes = plt.subplots(3, 5, figsize=(15, 12),
                         subplot_kw={'xticks': (), 'yticks': ()})
for i, (component, ax) in enumerate(zip(pca.components_, axes.ravel())):
    ax.imshow(component.reshape(image_shape), cmap='viridis')
    ax.set_title("principal component {}".format((i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



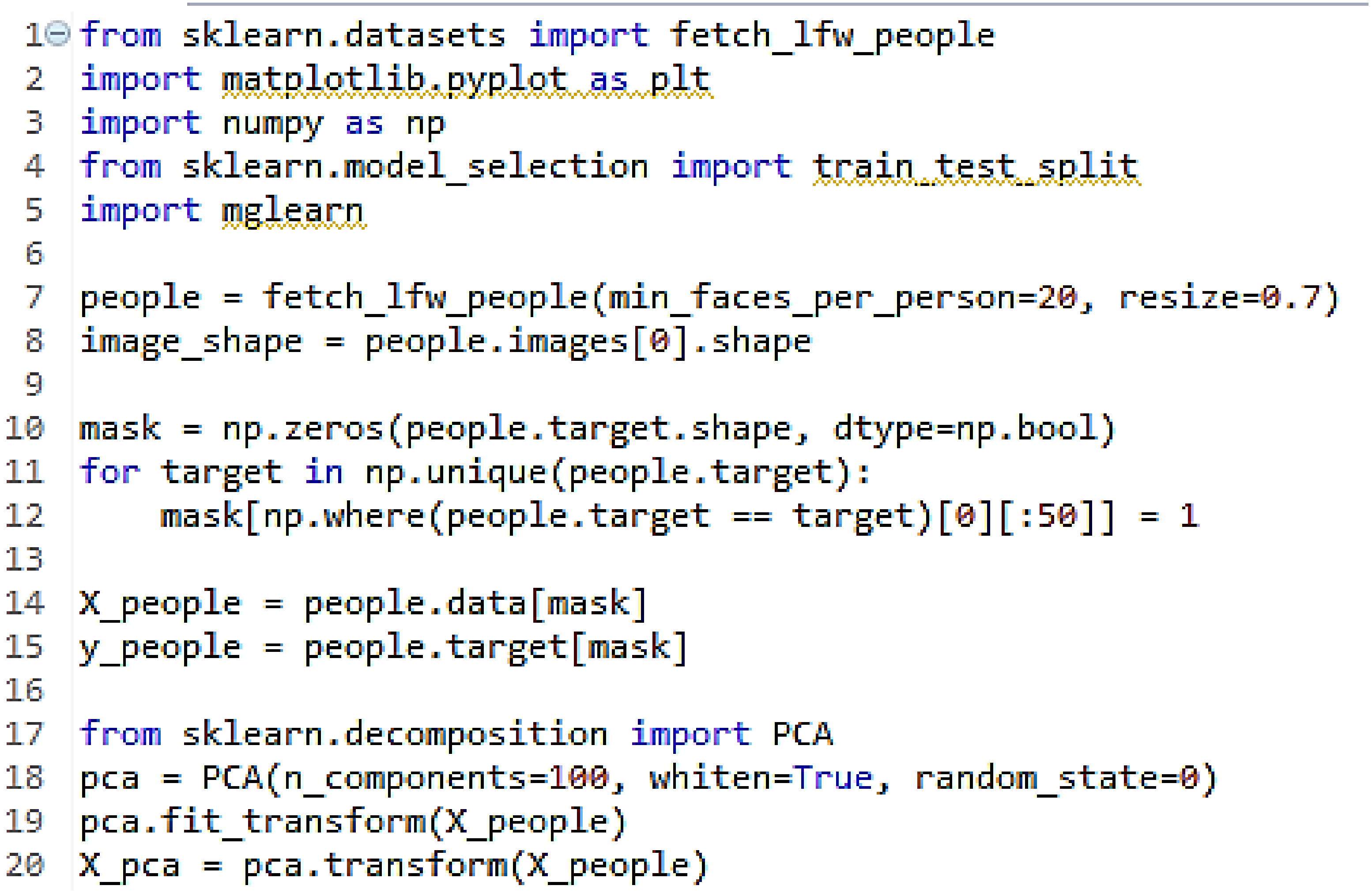
특성추출예제 – 주성분을 사용해 원본이미지 재구성



```
mglearn.plots.plot_pca_faces(X_train, X_test, image_shape)
plt.show()
```

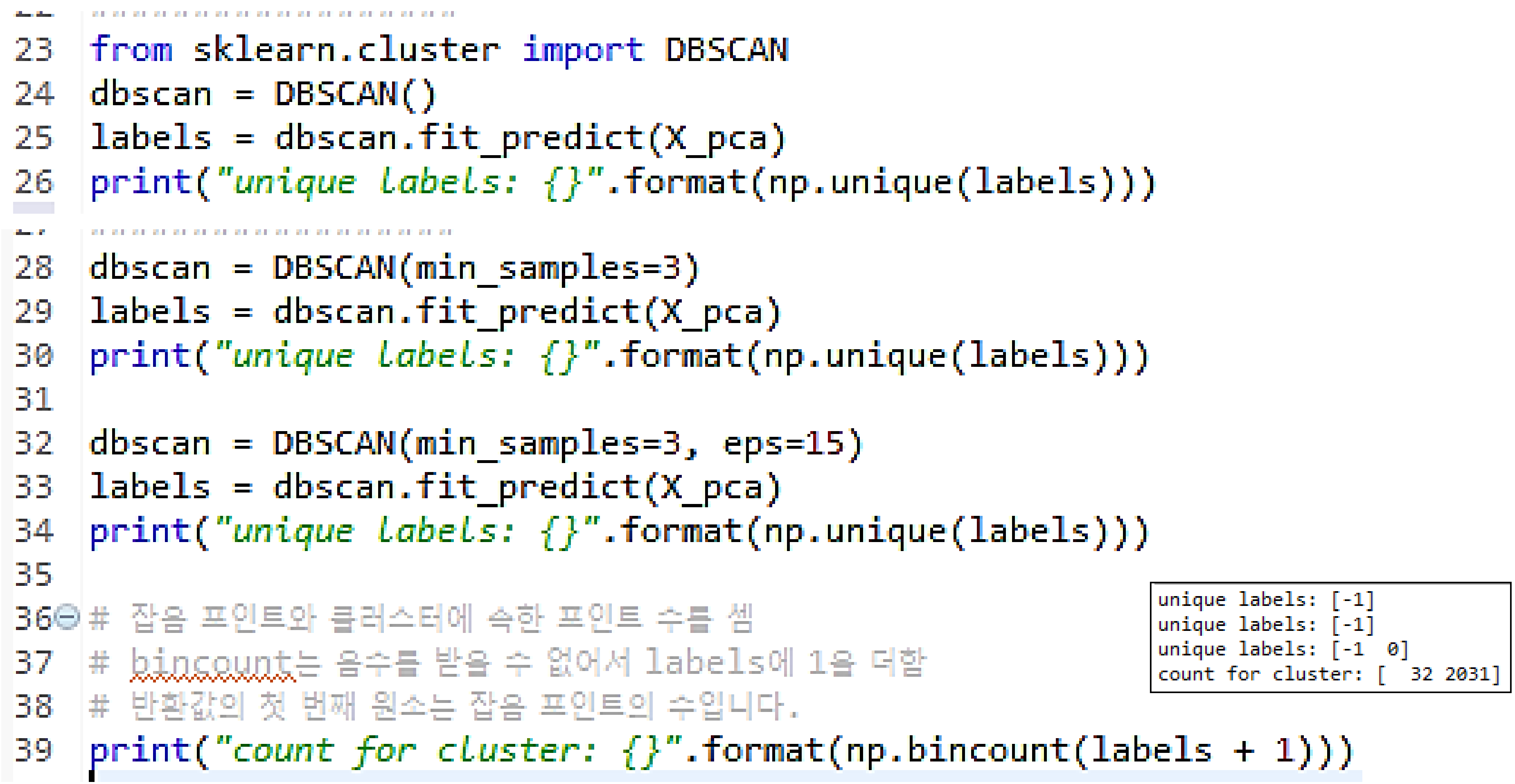


군집 알고리즘 예제



```python
from sklearn.datasets import fetch_lfw_people
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
import mglearn

people = fetch_lfw_people(min_faces_per_person=20, resize=0.7)
image_shape = people.images[0].shape

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]

from sklearn.decomposition import PCA
pca = PCA(n_components=100, whiten=True, random_state=0)
pca.fit_transform(X_people)
X_pca = pca.transform(X_people)
```

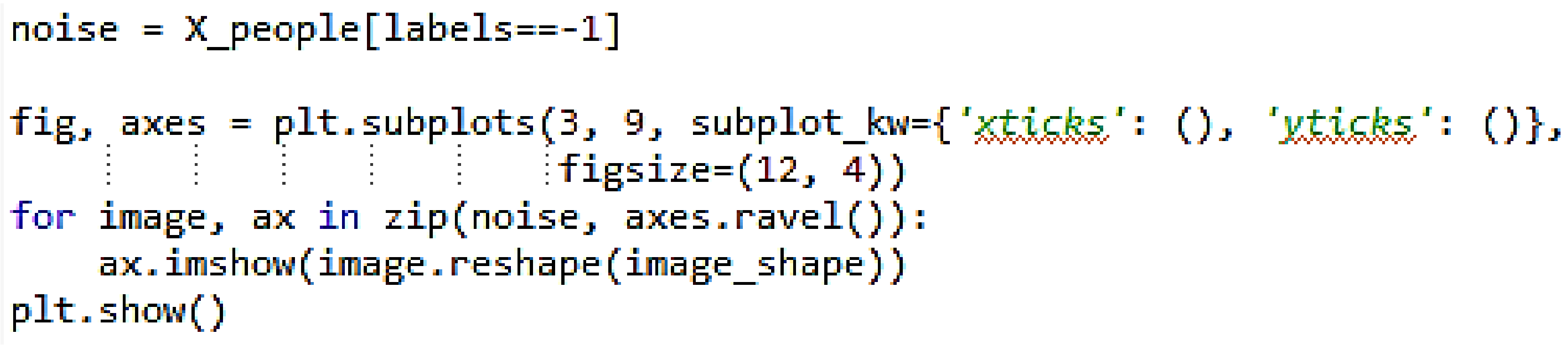
군집 알고리즘 예제 – DBSCAN



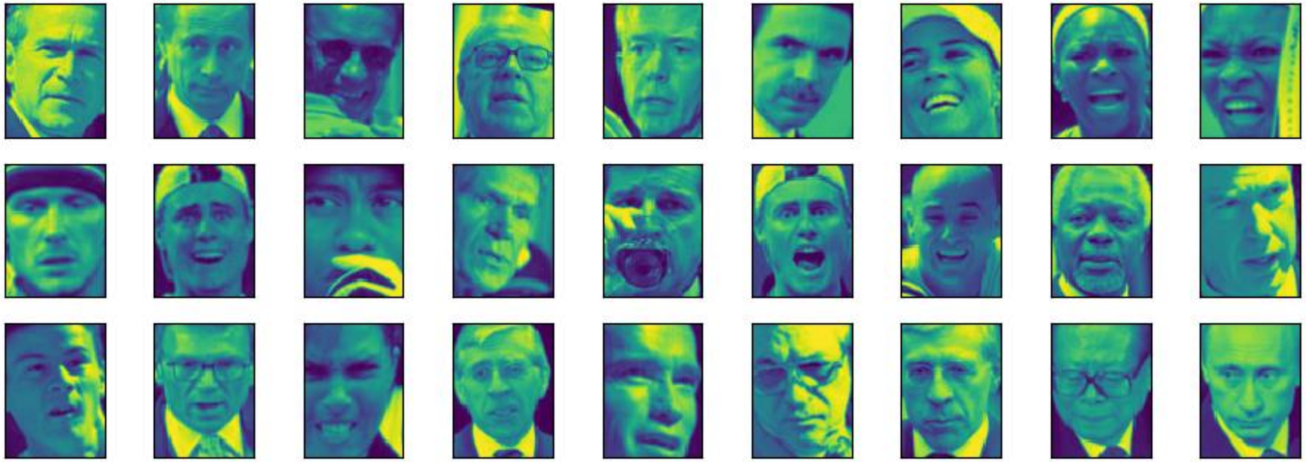
```python
from sklearn.cluster import DBSCAN
dbscan = DBSCAN()
labels = dbscan.fit_predict(X_pca)
print("unique labels: {}".format(np.unique(labels)))

dbscan = DBSCAN(min_samples=3)
labels = dbscan.fit_predict(X_pca)
print("unique labels: {}".format(np.unique(labels)))

dbscan = DBSCAN(min_samples=3, eps=15)
labels = dbscan.fit_predict(X_pca)
print("unique labels: {}".format(np.unique(labels)))

# 잡음 포인트와 클러스터에 속한 포인트 수를 셈
# bincount는 음수를 받을 수 없어서 labels에 1을 더함
# 반환값의 첫 번째 원소는 잡음 포인트의 수입니다.
print("count for cluster: {}".format(np.bincount(labels + 1)))
```

```
unique labels: [-1]
unique labels: [-1]
unique labels: [-1  0]
count for cluster: [  32 2031]
```

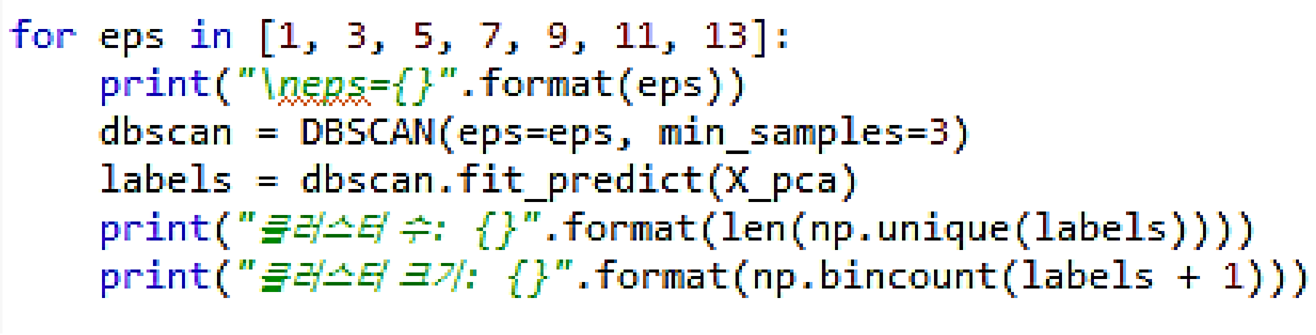
군집 알고리즘 예제 – DBSCAN



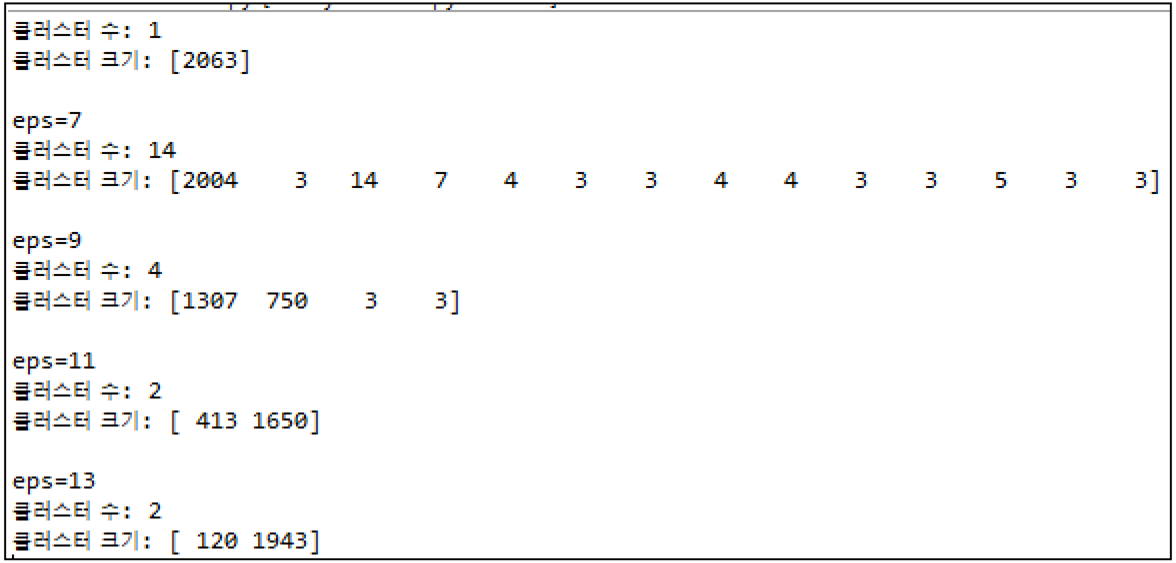
```
noise = X_people[labels==-1]

fig, axes = plt.subplots(3, 9, subplot_kw={'xticks': (), 'yticks': ()},
                         figsize=(12, 4))
for image, ax in zip(noise, axes.ravel()):
    ax.imshow(image.reshape(image_shape))
plt.show()
```



군집 알고리즘 예제 – DBSCAN

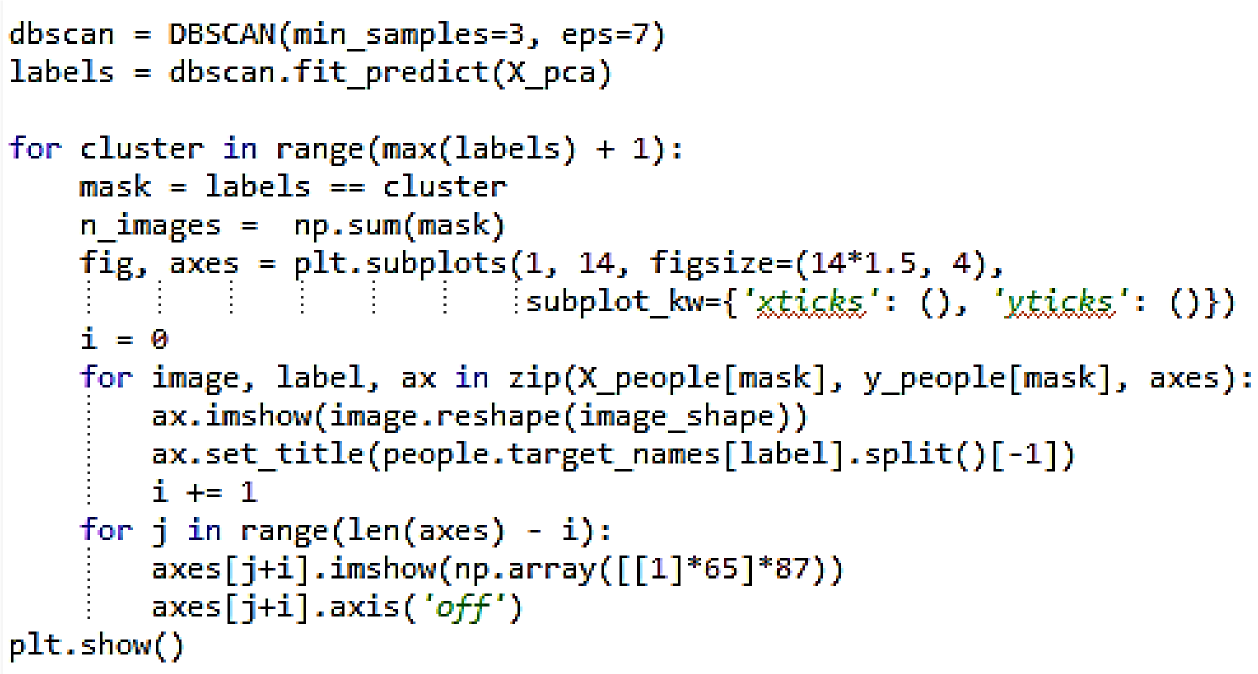


```
for eps in [1, 3, 5, 7, 9, 11, 13]:
    print("\neps={}".format(eps))
    dbscan = DBSCAN(eps=eps, min_samples=3)
    labels = dbscan.fit_predict(X_pca)
    print("클러스터 수: {}".format(len(np.unique(labels))))
    print("클러스터 크기: {}".format(np.bincount(labels + 1)))
```

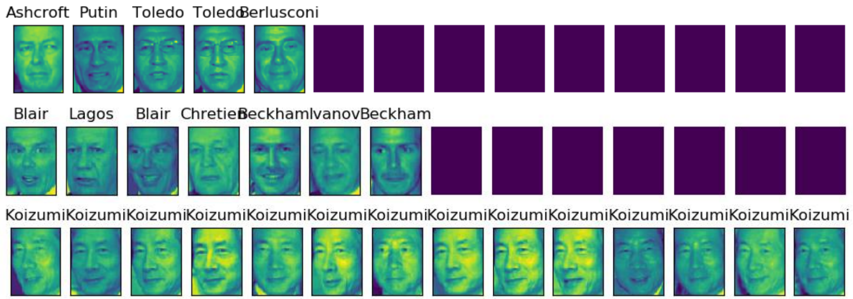


```
클러스터 수: 1
클러스터 크기: [2063]

eps=7
클러스터 수: 14
클러스터 크기: [2004    3   14    7    4    3    3    4    4    3    3    5    3    3]

eps=9
클러스터 수: 4
클러스터 크기: [1307  750    3    3]

eps=11
클러스터 수: 2
클러스터 크기: [ 413 1650]

eps=13
클러스터 수: 2
클러스터 크기: [ 120 1943]
```

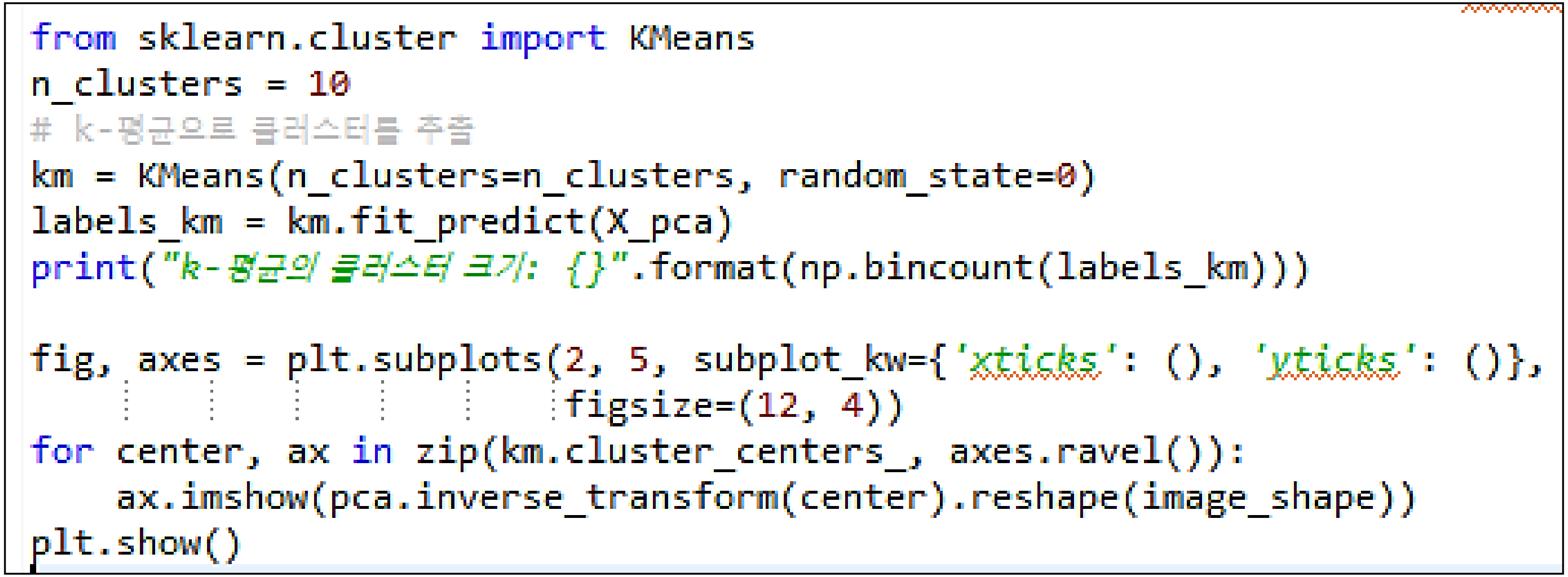
군집 알고리즘 예제 – DBSCAN



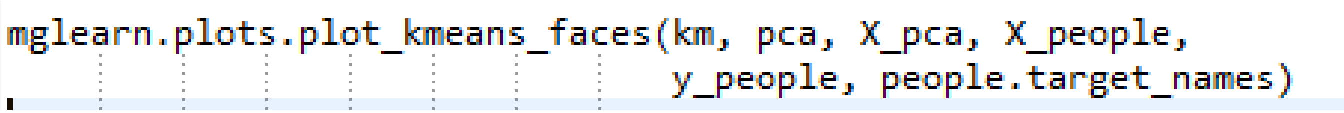
```python
dbscan = DBSCAN(min_samples=3, eps=7)
labels = dbscan.fit_predict(X_pca)

for cluster in range(max(labels) + 1):
    mask = labels == cluster
    n_images =  np.sum(mask)
    fig, axes = plt.subplots(1, 14, figsize=(14*1.5, 4),
                             subplot_kw={'xticks': (), 'yticks': ()})
    i = 0
    for image, label, ax in zip(X_people[mask], y_people[mask], axes):
        ax.imshow(image.reshape(image_shape))
        ax.set_title(people.target_names[label].split()[-1])
        i += 1
    for j in range(len(axes) - i):
        axes[j+i].imshow(np.array([[1]*65]*87))
        axes[j+i].axis('off')
plt.show()
```



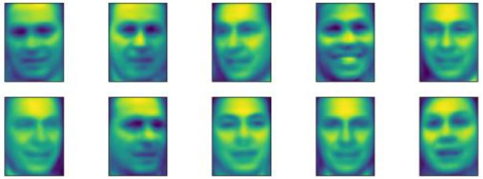
군집 알고리즘 예제 – kmeans

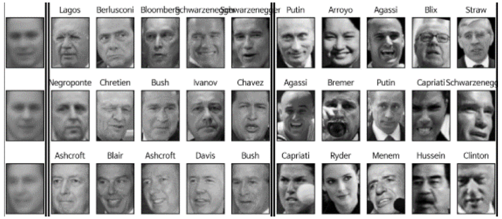


```
from sklearn.cluster import KMeans
n_clusters = 10
# k-평균으로 클러스터를 추출
km = KMeans(n_clusters=n_clusters, random_state=0)
labels_km = km.fit_predict(X_pca)
print("k-평균의 클러스터 크기: {}".format(np.bincount(labels_km)))

fig, axes = plt.subplots(2, 5, subplot_kw={'xticks': (), 'yticks': ()},
                         figsize=(12, 4))
for center, ax in zip(km.cluster_centers_, axes.ravel()):
    ax.imshow(pca.inverse_transform(center).reshape(image_shape))
plt.show()
```

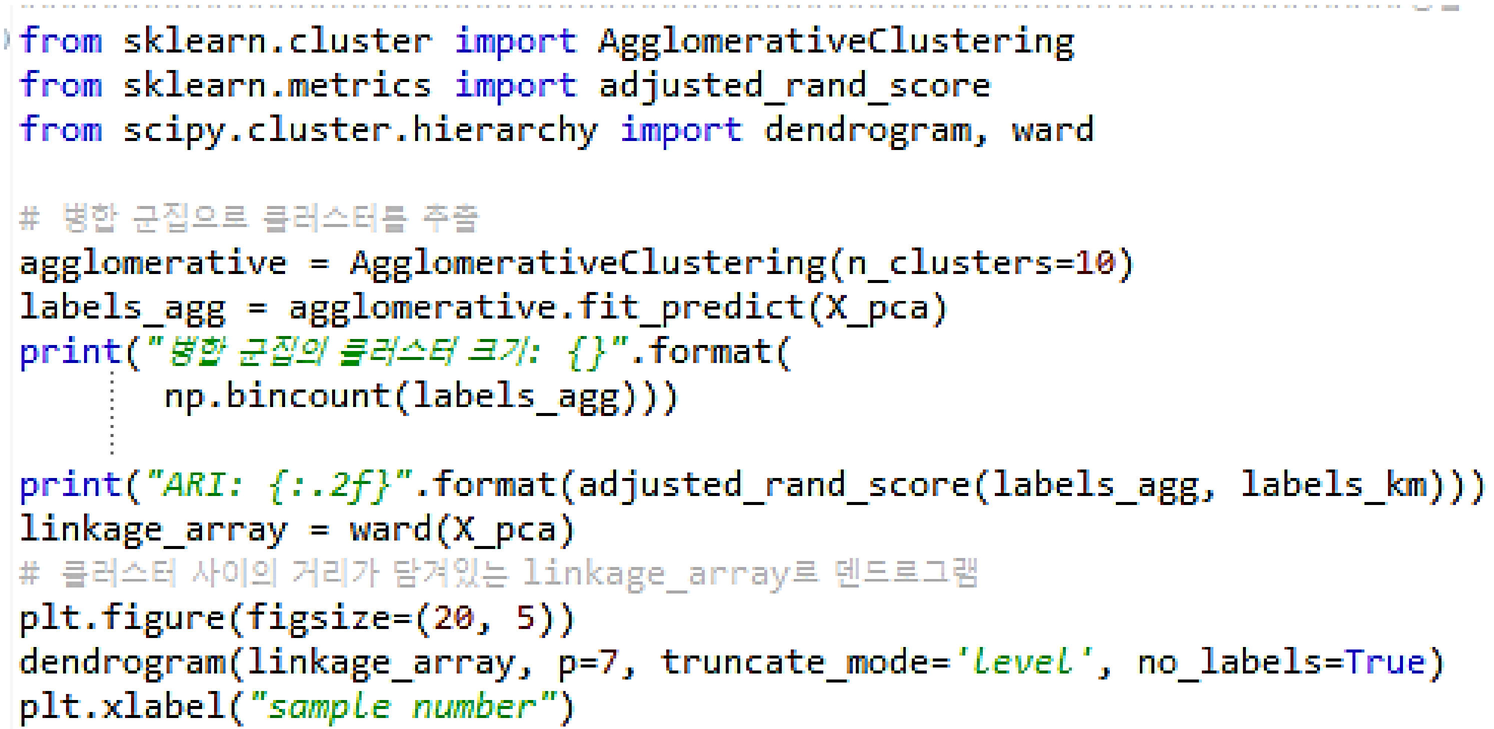


```
mglearn.plots.plot_kmeans_faces(km, pca, X_pca, X_people,
                                y_people, people.target_names)
```

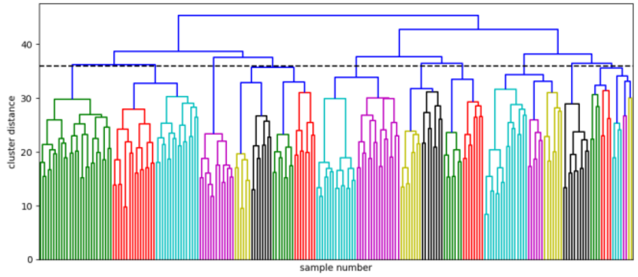




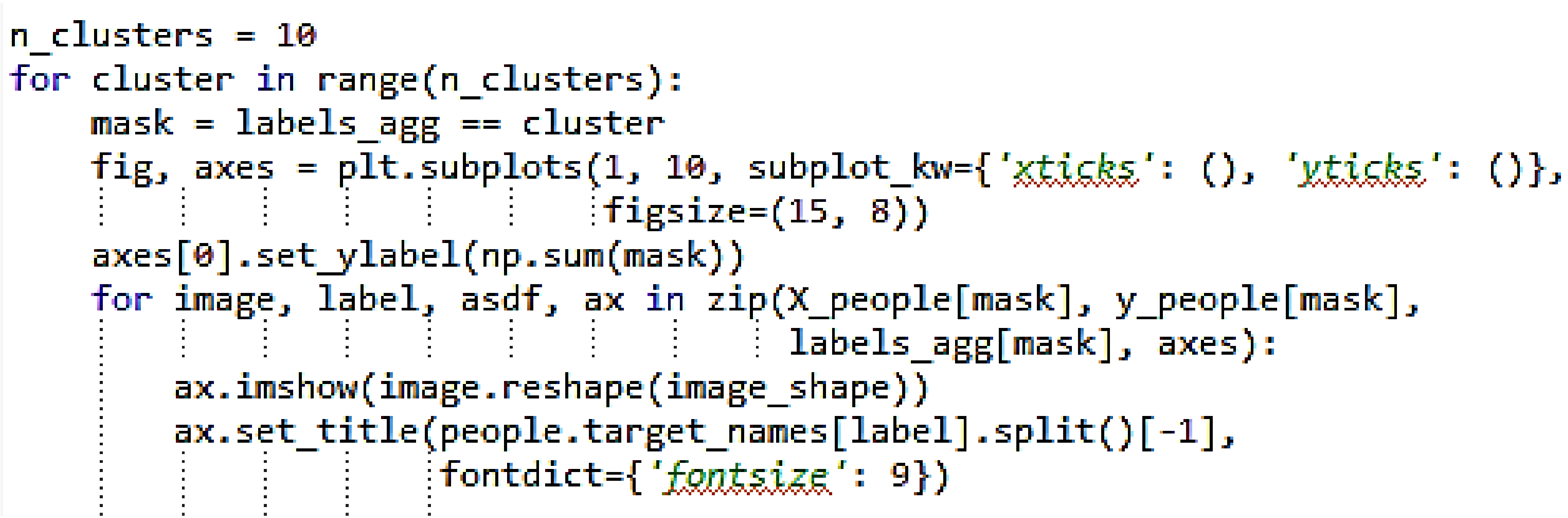
군집 알고리즘 예제 – AgglomerativeClustering



```
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import adjusted_rand_score
from scipy.cluster.hierarchy import dendrogram, ward

# 병합 군집으로 클러스터를 추출
agglomerative = AgglomerativeClustering(n_clusters=10)
labels_agg = agglomerative.fit_predict(X_pca)
print("병합 군집의 클러스터 크기: {}".format(
       np.bincount(labels_agg)))

print("ARI: {:.2f}".format(adjusted_rand_score(labels_agg, labels_km)))
linkage_array = ward(X_pca)
# 클러스터 사이의 거리가 담겨있는 linkage_array로 덴드로그램
plt.figure(figsize=(20, 5))
dendrogram(linkage_array, p=7, truncate_mode='level', no_labels=True)
plt.xlabel("sample number")
plt.ylabel("cluster distance")
ax = plt.gca()
bounds = ax.get_xbound()
ax.plot(bounds, [36, 36], '--', c='k')
plt.show()
```



군집 알고리즘 예제 – AgglomerativeClustering



```
n_clusters = 10
for cluster in range(n_clusters):
    mask = labels_agg == cluster
    fig, axes = plt.subplots(1, 10, subplot_kw={'xticks': (), 'yticks': ()},
                             figsize=(15, 8))
    axes[0].set_ylabel(np.sum(mask))
    for image, label, asdf, ax in zip(X_people[mask], y_people[mask],
                                      labels_agg[mask], axes):
        ax.imshow(image.reshape(image_shape))
        ax.set_title(people.target_names[label].split()[-1],
                     fontdict={'fontsize': 9})
```