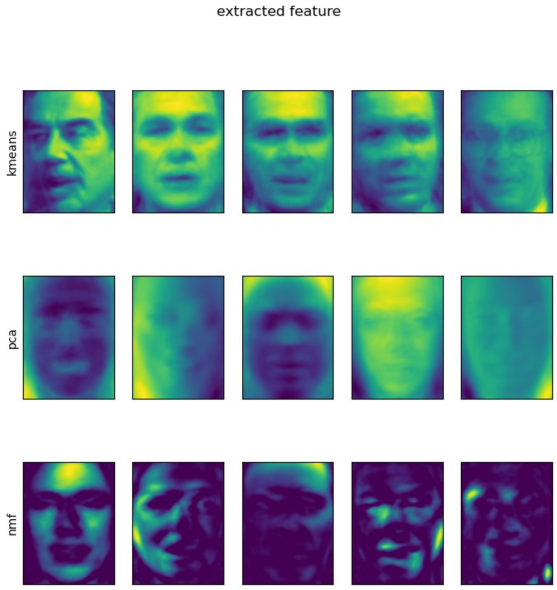
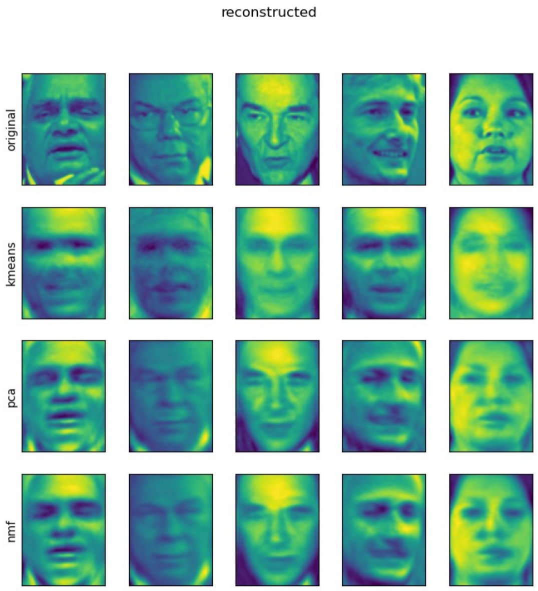
Q1-5. K-Means Image Reconstruction

```
import numpy as np
import os

import matplotlib as mpl
import matplotlib.pyplot as plt

import sklearn
from sklearn.datasets import fetch_lfw_people
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.decomposition import NMF

##########################################
### Image Reconstruction using K-Means ###
##########################################

### Prepare image dataset ##########################################################################################
# Load Labeled Faces in the Wild (LFW) people dataset
people = fetch_lfw_people(min_faces_per_person=20, resize=0.7)

image_shape = people.images[0].shape    # Acquire the shape of the image

# Collect the images with matching target names
# Filter out the images with mismatched names
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]        # Use the image data with matching target names
y_people = people.target[mask]      # Use the target names that match the images

X_people = X_people / 255   # Scale the grayscale image values between 0 and 1

# Split the dataset between training dataset and test dataset
X_train, X_test, y_train, y_test = train_test_split(X_people, y_people, stratify=y_people, random_state=42)

### Preapre K-Means clusterer for clustering, PCA and NMF for dimensionaltiy reduction ##############################################################
nmf = NMF(n_components=100, random_state=0).fit(X_train)           # NMF that reduces current image dataset into the dataset with 100 features
pca = PCA(n_components=100, random_state=0).fit(X_train)           # PCA that reduces current image dataset into the dataset with 100 features
kmeans = KMeans(n_clusters=100, random_state=0).fit(X_train)       # K-Means clustering that clusters current image dataset under 100 clusters

X_reconstructed_pca = pca.inverse_transform(pca.transform(X_test))          # Reconstruct the dataset using 100 features from PCA
X_reconstructed_kmeans = kmeans.cluster_centers_[kmeans.predict(X_test)]    # Reconstruct the dastset based on the clusters from K-Means
X_reconstructed_nmf = np.dot(nmf.transform(X_test), nmf.components_)        # Reconstruct the dataset using 100 features from NMF

### Display the image reconstruction results of PCA, NMF, K-Means ##############################################################################

# Display the image reconstruction results based on PCA features and NMF features
fig, axes = plt.subplots(3, 5, figsize=(8, 8), subplot_kw={'xticks': (), 'yticks': ()})
fig.suptitle("extracted feature")
for ax, comp_kmeans, comp_pca, comp_nmf in zip(
        axes.T, kmeans.cluster_centers_, pca.components_, nmf.components_):
    ax[0].imshow(comp_kmeans.reshape(image_shape))
    ax[1].imshow(comp_pca.reshape(image_shape), cmap='viridis')
    ax[2].imshow(comp_nmf.reshape(image_shape))

axes[0, 0].set_ylabel("kmeans")
axes[1, 0].set_ylabel("pca")
axes[2, 0].set_ylabel("nmf")

# Display the image reconstruction results based on clusters of K-Means
fig, axes = plt.subplots(4, 5, subplot_kw={'xticks': (), 'yticks': ()},
                         figsize=(8, 8))
fig.suptitle("reconstructed")
for ax, orig, rec_kmeans, rec_pca, rec_nmf in zip(
        axes.T, X_test, X_reconstructed_kmeans, X_reconstructed_pca,
        X_reconstructed_nmf):

    ax[0].imshow(orig.reshape(image_shape))
    ax[1].imshow(rec_kmeans.reshape(image_shape))
    ax[2].imshow(rec_pca.reshape(image_shape))
    ax[3].imshow(rec_nmf.reshape(image_shape))

axes[0, 0].set_ylabel("original")
axes[1, 0].set_ylabel("kmeans")
axes[2, 0].set_ylabel("pca")
axes[3, 0].set_ylabel("nmf")

plt.show()
```





- Grayscale 이미지의 Pixel 값에 K-Means Clustering 을 적용하여 각 Pixel 의 명암 정보가 Cluster 를 중심으로 재정리될 수 있음 .

- 각 Cluster 는 이미지 내의 명암 정보를 모아놓았기 때문에 사용하는 Cluster 의 개수에 따라 이미지의 표현력이 결정됨 . 이와 같이 K-Means 를 얼굴에 대한 Feature Extraction 으로 사용할 수 있음 . 이는 차원 축소를 통해 핵심 Feature 를 추출하는 PCA 와 유사한 효과를 가짐 .

- 더 많은 Cluster 를 사용할수록 이미지를 더욱 선명하게 표현할 수 있으며 , 이는 더 많은 Feature 로 이미지를 표현하는 것과 같음 .