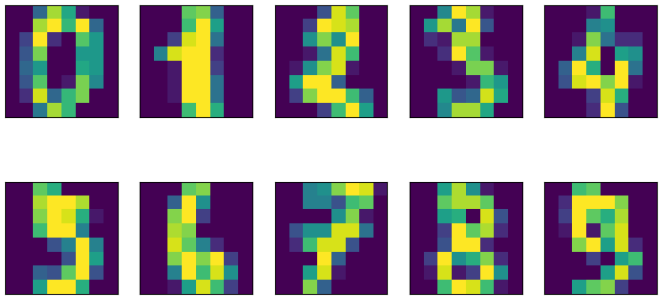
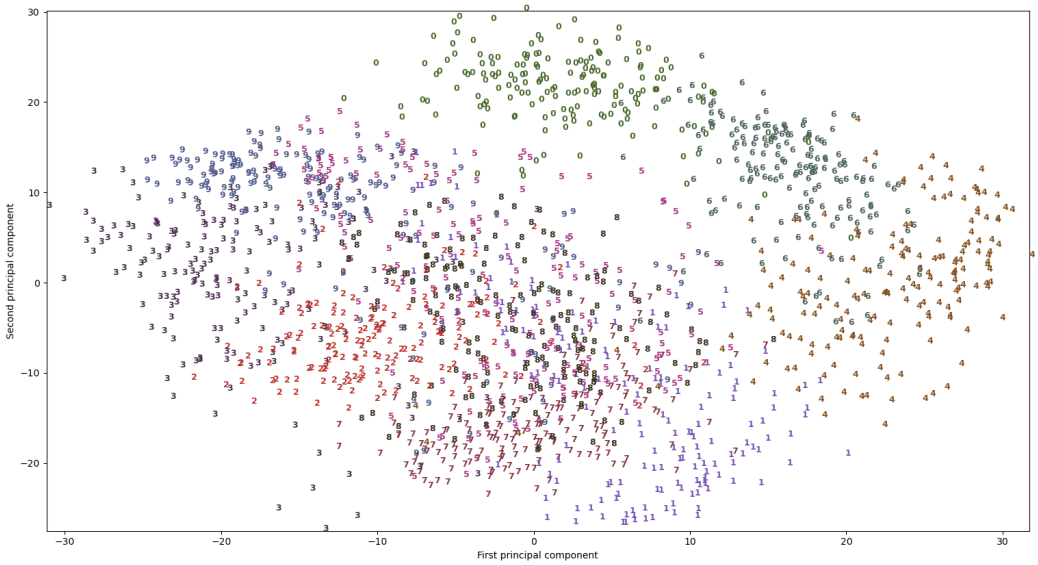
Q2-3. Manifold Learning with t-SNE

```
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import mglearn

from sklearn.decomposition import PCA
from sklearn.datasets import load_digits
from sklearn.manifold import TSNE

##########################################################################################################################################
### Load Digits Dataset ##################################################################################################################
##########################################################################################################################################
digits = load_digits()          # Digits 데이터셋 준비

# 데이터셋에서 10 개의 데이터를 출력함
fig, axes = plt.subplots(2, 5, figsize=(10, 5), subplot_kw={'xticks':(), 'yticks': ()})
for ax, img in zip(axes.ravel(), digits.images):
    ax.imshow(img)

print(digits.images.shape)  # Digits 데이터셋의 데이터 이미지의 형태를 출력함

##########################################################################################################################################
### Principle Component Analysis of Digits Dataset #######################################################################################
##########################################################################################################################################
pca = PCA(n_components=2)   # 2 개의 상위 Principle Component 를 생성하는 PCA 객체 준비
pca.fit(digits.data)        # Digits 데이터셋에 대해 2 개의 상위 Principle Component 를 생성함

digits_pca = pca.transform(digits.data)     # Principle Component 를 이용하여 데이터셋을 재구성함

# PCA 기반으로 재구성된 데이터셋에 대한 분포를 그래프로 그림
colors = ["#476A2A", "#7851B8", "#BD3430", "#4A2D4E", "#875525",
          "#A83683", "#4E655E", "#853541", "#3A3120", "#535D8E"]
plt.figure(figsize=(10, 10))
plt.xlim(digits_pca[:, 0].min(), digits_pca[:, 0].max())    # X 축의 범위를 정의함
plt.ylim(digits_pca[:, 1].min(), digits_pca[:, 1].max())    # Y 축의 범위를 정의함

# 1st Principle Component 와 2nd Principle Component 로 Digits 데이터셋을 재구성하여 그래프로 그림
for i in range(len(digits.data)):
    plt.text(digits_pca[i, 0], digits_pca[i, 1], str(digits.target[i]),
             color = colors[digits.target[i]],
             fontdict={'weight': 'bold', 'size': 9})

plt.xlabel("First principal component")
plt.ylabel("Second principal component")
plt.show()

##########################################################################################################################################
### t-SNE (t-Stochastic Neighbor Embedding) of Digits Dataset ############################################################################
##########################################################################################################################################
tsne = TSNE(random_state=42)    # t-SNE Feature 를 생성하는 t-SNE 객체 준비

digits_tsne = tsne.fit_transform(digits.data)   # t-SNE 로 데이터셋을 재구성함

plt.figure(figsize=(10, 10))
plt.xlim(digits_tsne[:, 0].min(), digits_tsne[:, 0].max() + 1)  # X 축의 범위를 정의함
plt.ylim(digits_tsne[:, 1].min(), digits_tsne[:, 1].max() + 1)  # Y 축의 범위를 정의함

# t-SNE Feature 로 Digits 데이터셋을 재구성하여 그래프로 그림
for i in range(len(digits.data)):
    plt.text(digits_tsne[i, 0], digits_tsne[i, 1], str(digits.target[i]),
             color = colors[digits.target[i]],
             fontdict={'weight': 'bold', 'size': 9})
plt.xlabel("t-SNE feature 0")
plt.xlabel("t-SNE feature 1")
plt.show()
```

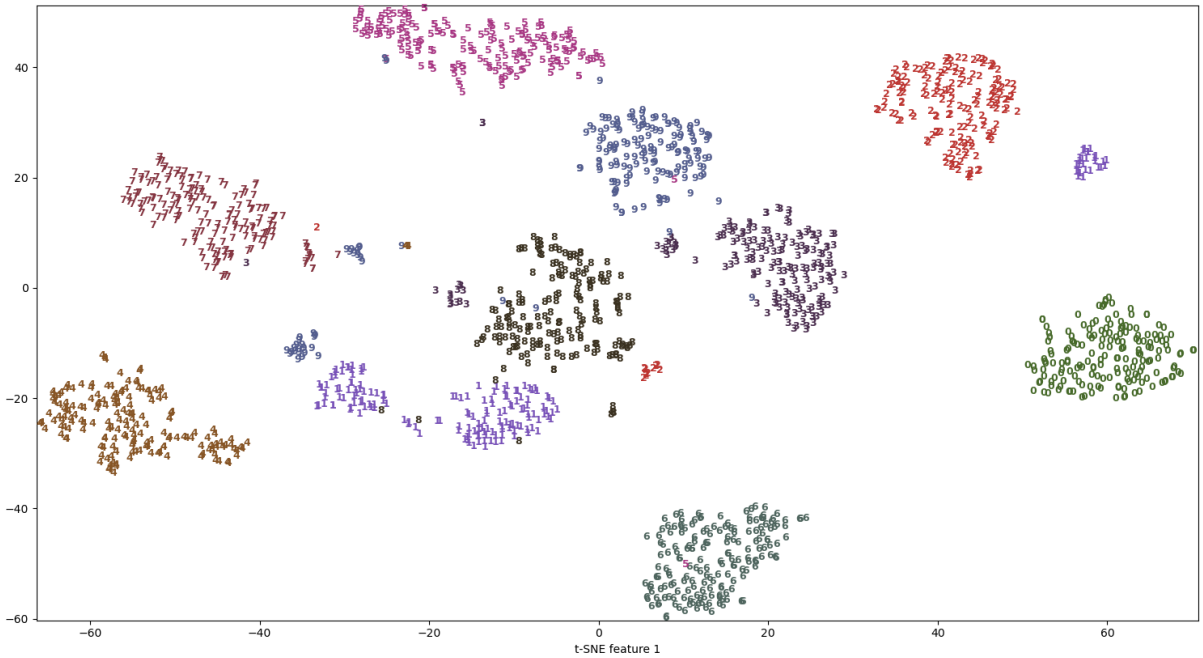


Digits 데이터셋의 클래스별 이미지



PCA 로 재구성된 Digits 데이터셋 분포

1st PC 와 2nd PC 로 재구성된 데이터셋의 분포는
Classification 에 적합하지 않기 때문에 다른 차원 축소 기법이 요구됨



t-SNE 로 재구성된 Digits 데이터셋 분포

t-SNE 로 재구성된 데이터셋 분포가
PCA 로 재구성했을 때보다 Classification 에 효과적임