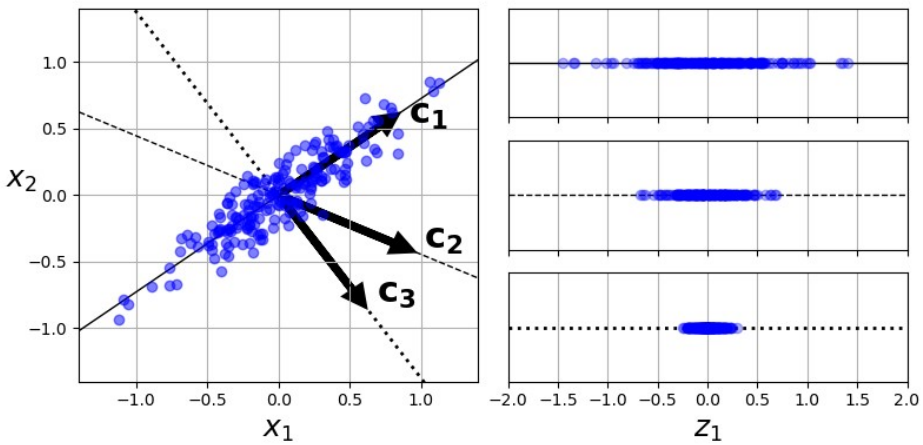
Q1-3. PCA (Principle Component Analysis)

```
import sys

import sklearn

import numpy as np
import os

import matplotlib.pyplot as plt
from matplotlib import gridspec

from sklearn.datasets import make_swiss_roll

# 그래프 결과를 저장한 경로 정의
PROJECT_ROOT_DIR = '.'
CHAPTER_ID = 'dim_reduction'
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, 'images', CHAPTER_ID)
os.makedirs(IMAGES_PATH, exist_ok=True)

# 그래프 결과를 저장하는 함수
def save_fig(fig_id, tight_layout=True, fig_extension='png', resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + '.' + fig_extension)
    print('Save Image ', fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)

############################################################################################################################################
#### Prepare the random dataset ############################################################################################################
############################################################################################################################################
angle = np.pi / 5   # 36 도
stretch = 5
m = 200

np.random.seed(3)
X = np.random.randn(m, 2) / 10

X = X.dot(np.array([[stretch, 0], [0, 1]]))                                 #  벡터에 변수를 곱함으로서 벡터의 위치를 늘리는 효과를 가짐
X = X.dot([[np.cos(angle), np.sin(angle)], [-np.sin(angle), np.cos(angle)]])    #  벡터에 Rotation Matrix( 회전 행렬 ) 을 곱함으로서 벡터를 회전시킴

############################################################################################################################################
### Prepare principle components and Re-Organize the dataset ###############################################################################
############################################################################################################################################
u1 = np.array([np.cos(angle), np.sin(angle)])                           # 36 도로 배치된 Unit Vector u1 / Principle Component  후보군 1
u2 = np.array([np.cos(angle - 2 * np.pi/6), np.sin(angle - 2 * np.pi/6)])   # -24 도로 배치된 Unit Vector u2 / Principle Component  후보군 2
u3 = np.array([np.cos(angle - np.pi/2), np.sin(angle - np.pi/2)])           # -54 도로 배치된 Unit Vector u3 / Principle Component  후보군 3

X_proj1 = X.dot(u1.reshape(-1, 1))      # 데이터셋을 u1  벡터에 투영시킴
X_proj2 = X.dot(u2.reshape(-1, 1))      # 데이터셋을 u2  벡터에 투영시킴
X_proj3 = X.dot(u3.reshape(-1, 1))      # 데이터셋을 u3  벡터에 투영시킴

############################################################################################################################################
### Plot the original dataset and potential principle components ###########################################################################
############################################################################################################################################
plt.figure(figsize=(8,4))
plt.subplot2grid((3,2), (0, 0), rowspan=3)

plt.plot([-1.4, 1.4], [-1.4*u1[1]/u1[0], 1.4*u1[1]/u1[0]], "k-", linewidth=1)   # u1  벡터를 그림
plt.plot([-1.4, 1.4], [-1.4*u2[1]/u2[0], 1.4*u2[1]/u2[0]], "k--", linewidth=1)  # u2  벡터를 그림
plt.plot([-1.4, 1.4], [-1.4*u3[1]/u3[0], 1.4*u3[1]/u3[0]], "k:", linewidth=2)   # u3  벡터를 그림

plt.plot(X[:, 0], X[:, 1], "bo", alpha=0.5)     # 원본 데이터셋을 그래프로 그림

plt.axis([-1.4, 1.4, -1.4, 1.4])
plt.arrow(0, 0, u1[0], u1[1], head_width=0.1, linewidth=5, length_includes_head=True, head_length=0.1, fc='k', ec='k')  # u1  벡터를 화살표를 그림
plt.arrow(0, 0, u2[0], u2[1], head_width=0.1, linewidth=5, length_includes_head=True, head_length=0.1, fc='k', ec='k')  # u2  벡터를 화살표를 그림
plt.arrow(0, 0, u3[0], u3[1], head_width=0.1, linewidth=5, length_includes_head=True, head_length=0.1, fc='k', ec='k')  # u3  벡터를 화살표를 그림

plt.text(u1[0] + 0.1, u1[1] - 0.05, r"$\mathbf{c_1}$", fontsize=22)
plt.text(u2[0] + 0.1, u2[1], r"$\mathbf{c_2}$", fontsize=22)
plt.text(u3[0] + 0.1, u3[1], r"$\mathbf{c_3}$", fontsize=22)

plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$x_2$", fontsize=18, rotation=0)
plt.grid(True)

############################################################################################################################################
### Plot the distribution of Re-Organized dataset ##########################################################################################
############################################################################################################################################

# u1 에 투영된 데이터셋의 분포를 그래프로 그림
plt.subplot2grid((3,2), (0, 1))
plt.plot([-2, 2], [0, 0], "k-", linewidth=1)

plt.plot(X_proj1[:, 0], np.zeros(m), "bo", alpha=0.3)   # u1 에 투영된 데이터셋을 그래프로 그림

plt.gca().get_yaxis().set_ticks([])
plt.gca().get_xaxis().set_ticklabels([])
plt.axis([-2, 2, -1, 1])
plt.grid(True)

# u2 에 투영된 데이터셋의 분포를 그래프로 그림
plt.subplot2grid((3,2), (1, 1))
plt.plot([-2, 2], [0, 0], "k--", linewidth=1)

plt.plot(X_proj2[:, 0], np.zeros(m), "bo", alpha=0.3)   # u2 에 투영된 데이터셋을 그래프로 그림

plt.gca().get_yaxis().set_ticks([])
plt.gca().get_xaxis().set_ticklabels([])
plt.axis([-2, 2, -1, 1])
plt.grid(True)

# u3 에 투영된 데이터셋의 분포를 그래프로 그림
plt.subplot2grid((3,2), (2, 1))
plt.plot([-2, 2], [0, 0], "k:", linewidth=2)

plt.plot(X_proj3[:, 0], np.zeros(m), "bo", alpha=0.3)   # u3 에 투영된 데이터셋을 그래프로 그림

plt.gca().get_yaxis().set_ticks([])
plt.axis([-2, 2, -1, 1])
plt.xlabel("$z_1$", fontsize=18)
plt.grid(True)

save_fig("pca_best_projection_plot")
plt.show()
```



차원 축소에 사용할 Unit Vector **각 Unit Vector 별로 Projection 된 데이터셋의 분포**