第一次作品：主成分分析實作

學號：410978002

姓名：謝元皓

作品目標：透過python將主成分概念視覺化表現出來

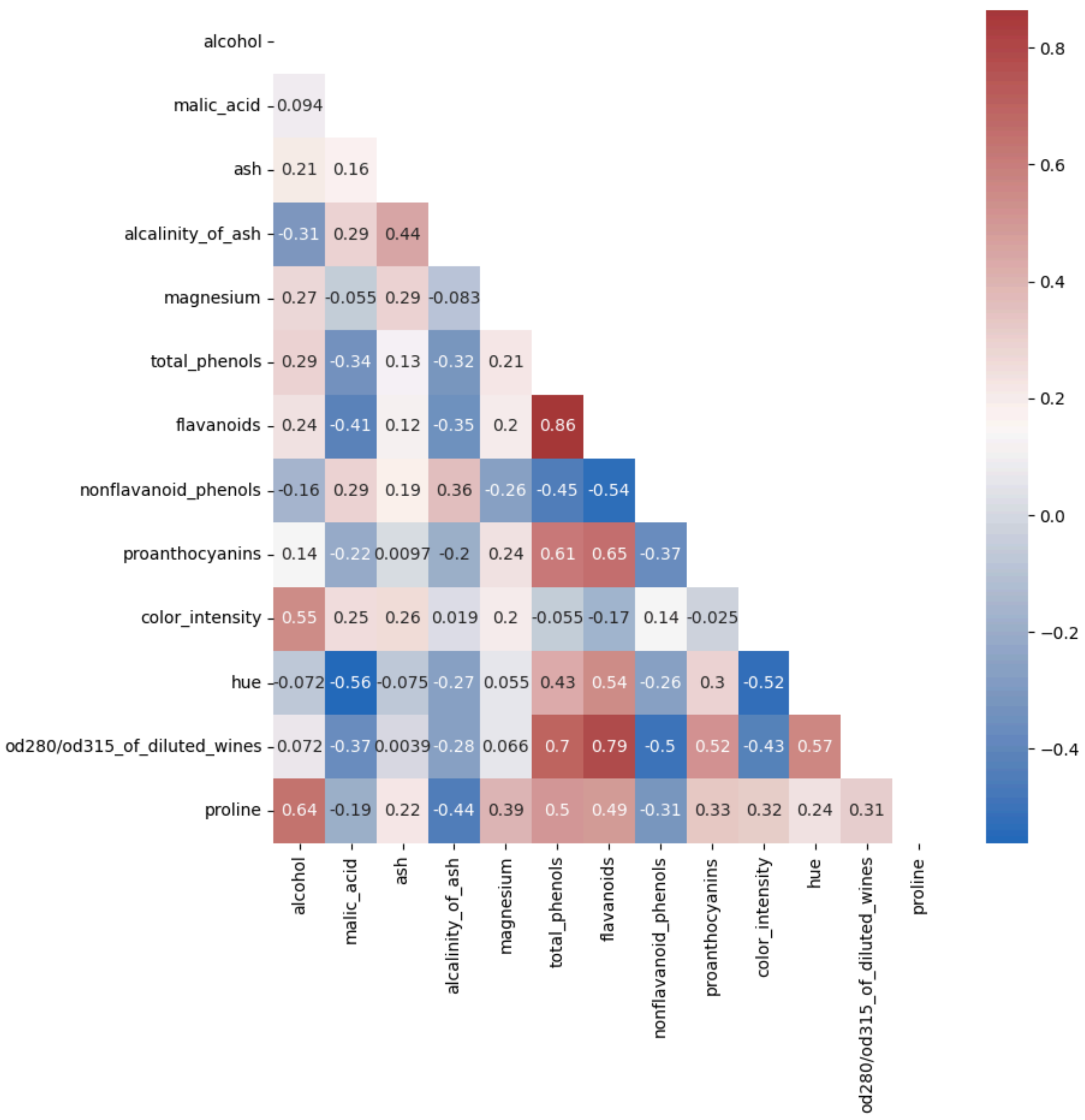
第 1 題：

有一組資料來自義大利某個地區三個紅酒製造商所產的紅酒，資料內容包括的 178 支紅酒的 13 種化學成分。利用這組資料回答下列問題：

(1) 繪製變數間的相關係數圖，以觀察變數間是否存在相關性。

- 在sklearn套件中下載紅酒資料
- 造出變數間的相關矩陣
- 由藍色到紅色依序表示相關程度由低到高

In [ ]:
```python
import pandas as pd
import numpy as np
import seaborn
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
# Load the wine dataset
X, y = load_wine(return_X_y=True, as_frame=True)
labels = X.columns
# Plot the correlation matrix
plt.figure(figsize = (9, 9))
R = X.corr()
mask = np.triu(np.ones_like(R, dtype=bool)) # diagonal mask
seaborn.heatmap(R, annot=True, mask = mask, cmap='vlag')
plt.show()
```



結論:

- 由圖形中可以看出，Flavanoids和Total_Phenols此兩變數最高度相關
- 相關係數矩陣為對稱矩陣，因此僅呈現下半部。

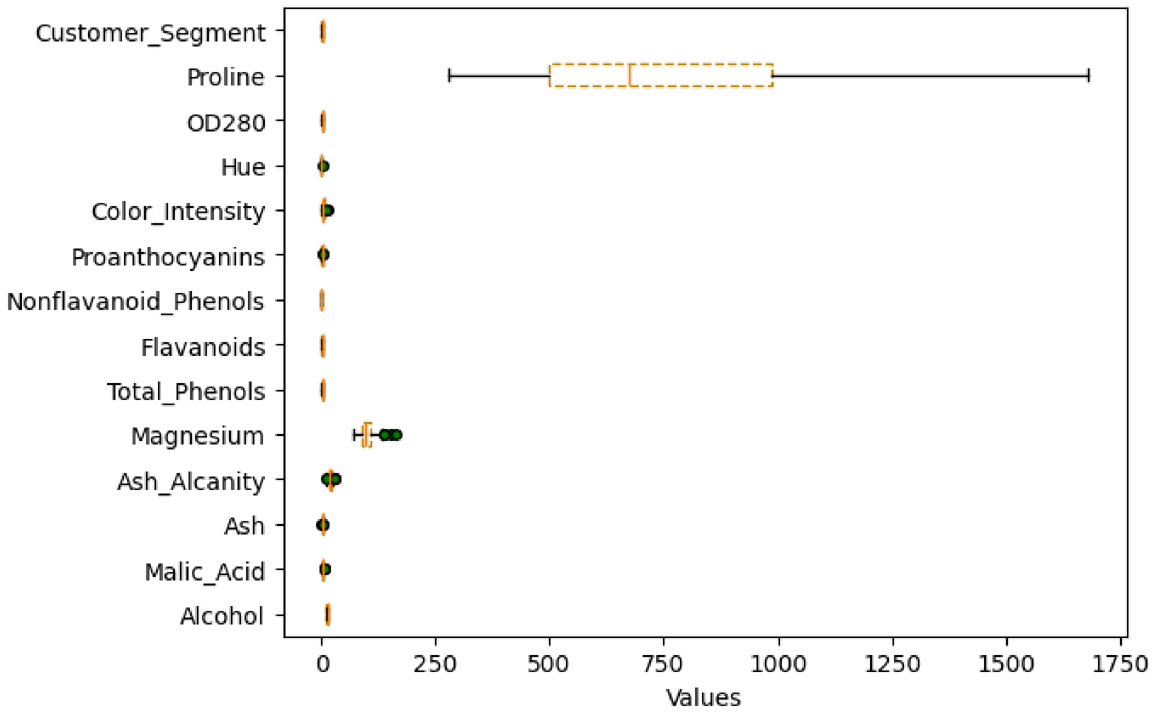
---

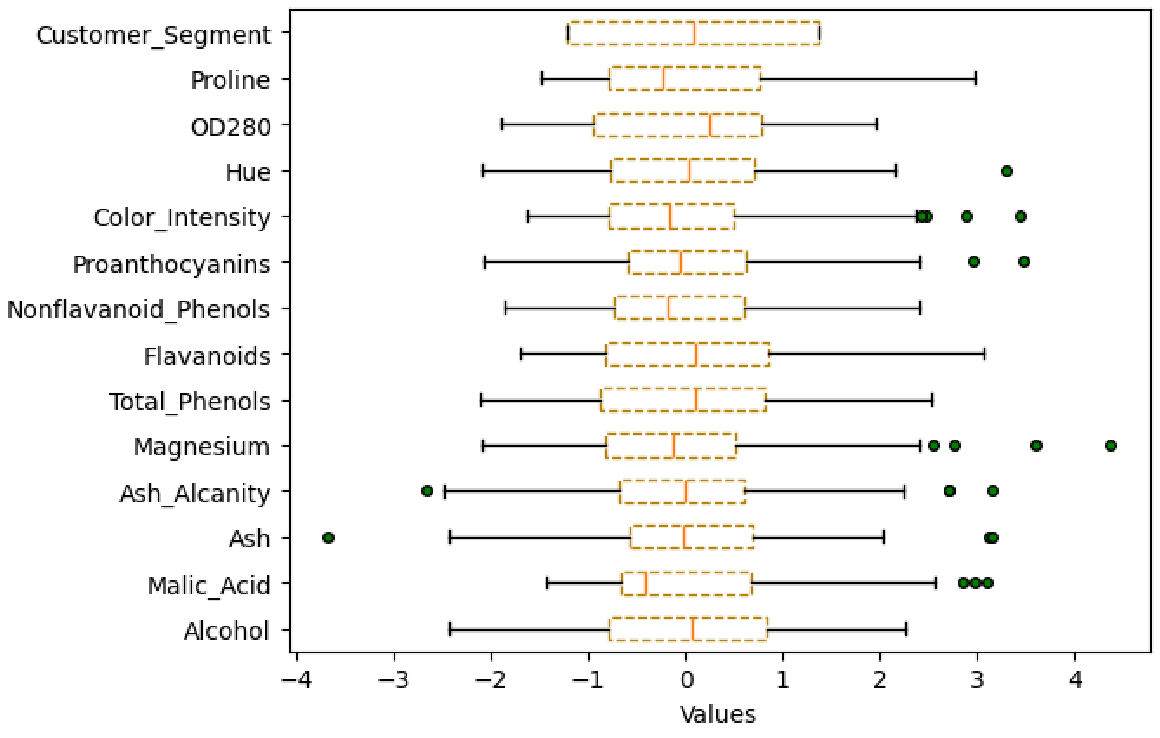
(2)繪製含每個化學成分變數盒鬚圖，分別畫為標準化與標準化。

- 資料未標準化之前為極度不平衡資料，可看出唯獨Proline 值特別大，因此需要做資料前處理
- 標準化後才不會受原始變數的數字大小所影響
- ratings為未標準化資料矩陣，ratings_為標準化資料矩陣

In [ ]:
```
df = pd.read_excel('wine.xlsx')
ratings = np.array(df)
categories = df.columns
fig, ax = plt.subplots()
boxprops = dict(linestyle = '--', linewidth = 1, \
color = 'darkgoldenrod')
flierprops = dict(marker='o', markerfacecolor = 'green',
markersize = 4, linestyle = 'none')
```

```
ax.boxplot(ratings, boxprops = boxprops, \
flierprops = flierprops, \
labels = categories, vert = False)
ax.set_xlabel('Values')
plt.show()

df = pd.read_excel('wine.xlsx')
ratings = np.array(df)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(ratings)
ratings_ = scaler.transform(ratings)
categories = df.columns
fig, ax = plt.subplots()
boxprops = dict(linestyle = '--', linewidth = 1, \
color = 'darkgoldenrod')
flierprops = dict(marker='o', markerfacecolor = 'green',
markersize = 4, linestyle = 'none')
ax.boxplot(ratings_, boxprops = boxprops, \
flierprops = flierprops, \
labels = categories, vert = False)
ax.set_xlabel('Values')
plt.show()
```





(3) 進行主成分分析，繪製特徵值由大而小的分布與 scree plot。

- 利用numpy指令cov計算樣本共變異數矩陣 S_x
- 比較後發現S_X與S_X-formula一致
- ratings_ is a 178 by 13 data matrix

In [ ]:
```python
import numpy as np
Sx = np.cov(ratings_.T, bias=False)
N = ratings_.shape[0]
mu_x = ratings_.mean(axis = 0)
Tmp = ratings_ - mu_x
Sx_formula = Tmp.T @ Tmp / (N - 1)
#print(Sx_formula)
```

接著進一步對共變異矩陣S_X進行特徵值與特徵向量分析，取得由大而小排列的特徵值及相對應特徵向量，最後再將特徵值與特徵向量合併回到原來的樣本共變異矩陣。

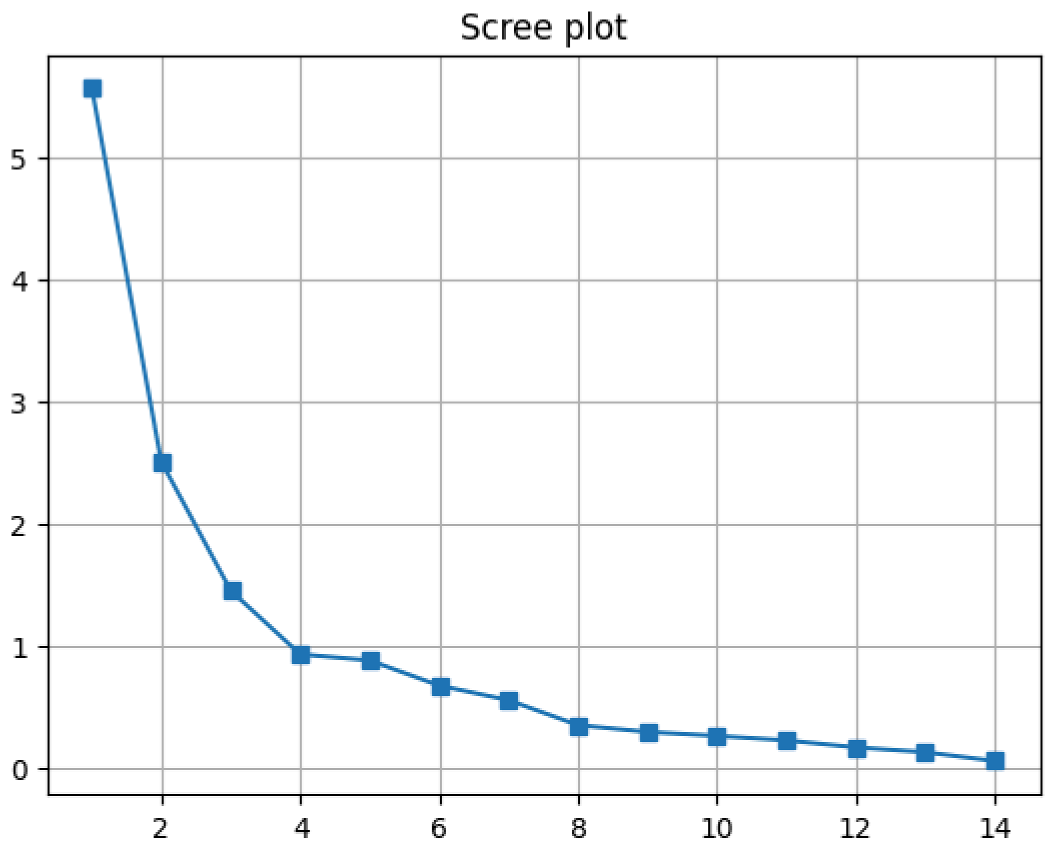
In [ ]:
```python
from numpy.linalg import eig
from numpy.linalg import inv
w, v = eig(Sx)
idx = np.argsort(-w) #sort eigenvalues in descending order
# idx = np.argsort(w)[::-1]
eigvals = w[idx]
eigvecs = v[:, idx]
Sigma_x = eigvecs @ np.diag(eigvals) @ eigvecs.T
```

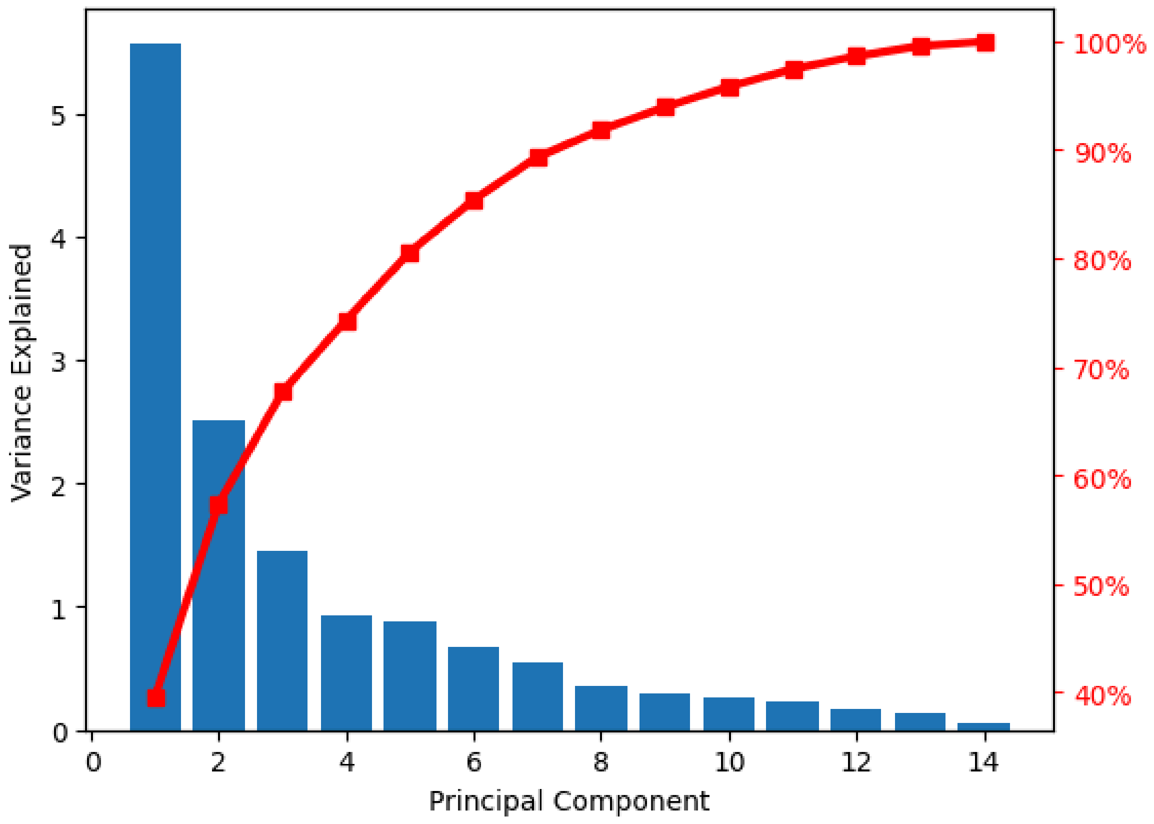
將特徵分布印出觀察以下兩圖

- Screen plot 主成分由大到小的分布
- Pareto plot 加入累積變異的比例

In [ ]:
```python
from matplotlib.ticker import PercentFormatter
plt.figure()
```

```
x = np.arange(1, 1+len(eigvals))
plt.plot(x, eigvals, marker='s')
plt.title('Scree plot')
plt.grid(True)
plt.show()
fig, ax = plt.subplots()
x = np.arange(1, 1+len(eigvals))
ax.bar(x, eigvals)
ax2 = ax.twinx()
ax2.plot(x, eigvals.cumsum()/eigvals.sum()*100, \
marker='s', color='red', lw=3)
ax2.tick_params(axis='y', colors='red')
ax2.yaxis.set_major_formatter(PercentFormatter())
ax.set_xlabel('Principal Component')
ax.set_ylabel('Variance Explained')
plt.show()
```

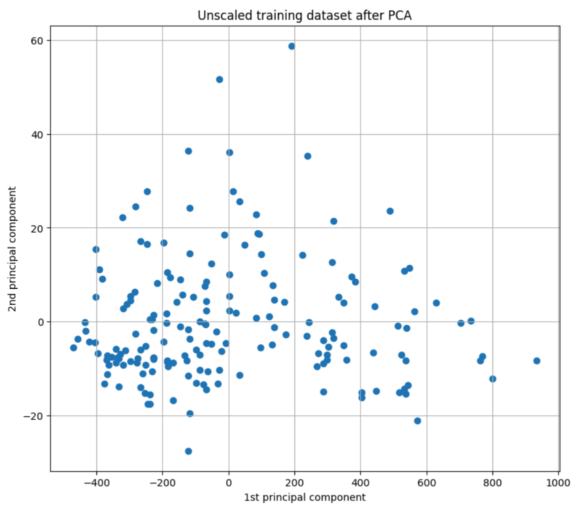


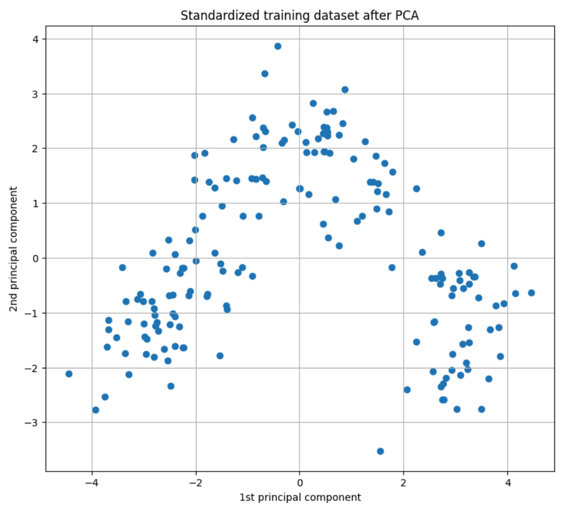


(4) 利用主成分分析取得前兩項成分，並繪製其散布圖

- 左圖為資料不做標準化，取前兩大主成分得到的散佈圖。
- 右圖為資料標準化後，取兩個主成分得到的散佈圖。

In [ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# 對未標準化的資料做PCA
pca = PCA().fit(ratings)
Z = pca.transform(ratings)
# 對標準化的資料做PCA
scaler = StandardScaler()
scaler.fit(ratings_)
ratings_ = scaler.transform(ratings_)
pca = PCA().fit(ratings_)
Z1 = pca.transform(ratings_)
# 做圖
fig, ax = plt.subplots(1, 2, figsize = (20, 8))
ax[0].scatter(Z[:,0], Z[:,1])
ax[0].set_xlabel('1st principal component')
ax[0].set_ylabel('2nd principal component')
ax[0].set_title("Unscaled training dataset after PCA")
ax[0].grid(True)
ax[1].scatter(Z1[:,0], Z1[:,1])
ax[1].set_xlabel('1st principal component')
ax[1].set_ylabel('2nd principal component')
ax[1].grid(True)
ax[1].set_title("Standardized training dataset after PCA")
plt.show()
```





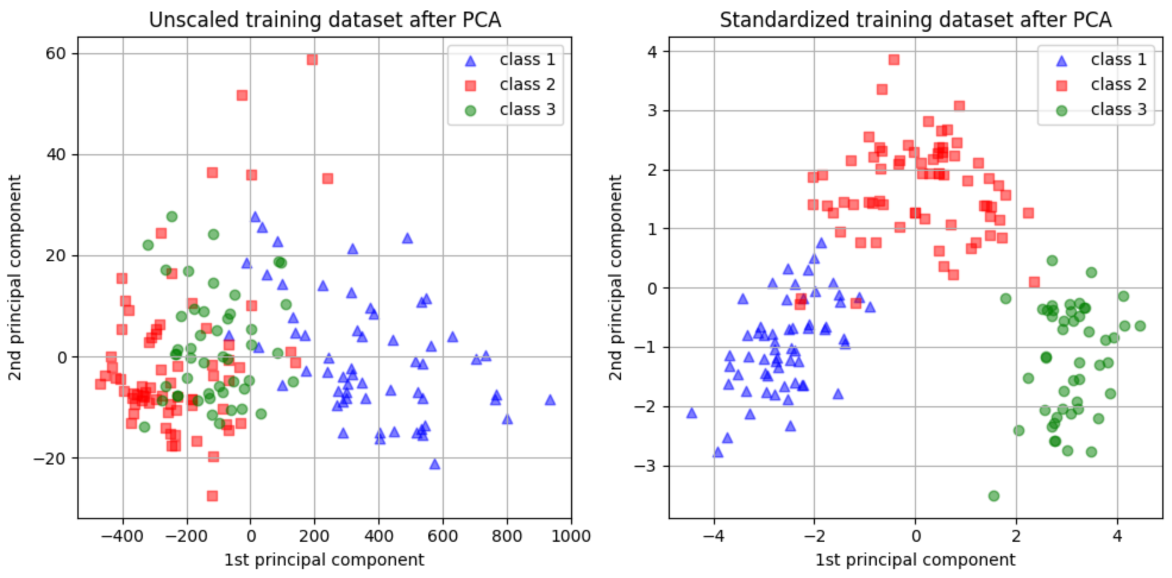
由右圖可以看出兩個主成分包含三個群組，並往下將此三組以顏色區隔

---

(5) 再依據每個資料的標籤，為每個在散布圖上的資料點依據標籤塗上顏色

- 左圖為資料不做標準化，得到標籤顏色分類的散佈圖。
- 右圖為資料標準化後，得到標籤顏色分類的的散佈圖。

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# 對未標準化的資料做PCA
pca = PCA().fit(ratings)
Z = pca.transform(ratings)
# 對標準化的資料做PCA
scaler = StandardScaler()
scaler.fit(ratings_)
ratings_ = scaler.transform(ratings_)
pca = PCA().fit(ratings_)
Z1 = pca.transform(ratings_)
# 做圖
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(10, 5))

target_classes = range(0, 3)
colors = ("blue", "red", "green")
markers = ("^", "s", "o")

for target_class, color, marker in zip(target_classes, colors, markers):
    ax1.scatter(
        x=Z[y == target_class, 0],
        y=Z[y == target_class, 1],
        color=color,
        label=f"class {target_class +1}",
        alpha=0.5,
        marker=marker,
    )

    ax2.scatter(
        x=Z1[y == target_class, 0],
        y=Z1[y == target_class, 1],
        color=color,
        label=f"class {target_class +1}",
        alpha=0.5,
        marker=marker,
    )

ax1.set_title("Unscaled training dataset after PCA")
ax2.set_title("Standardized training dataset after PCA")

for ax in (ax1, ax2):
    ax.set_xlabel("1st principal component")
    ax.set_ylabel("2nd principal component")
    ax.legend(loc="upper right")
    ax.grid()

_ = plt.tight_layout()
```



結論:

由比較圖可以看出資料是否做標準化相當重要，未做標準化主成分分析後並不能明確分群;

做完標準化之後，才能明顯區分來自不同酒莊的酒。

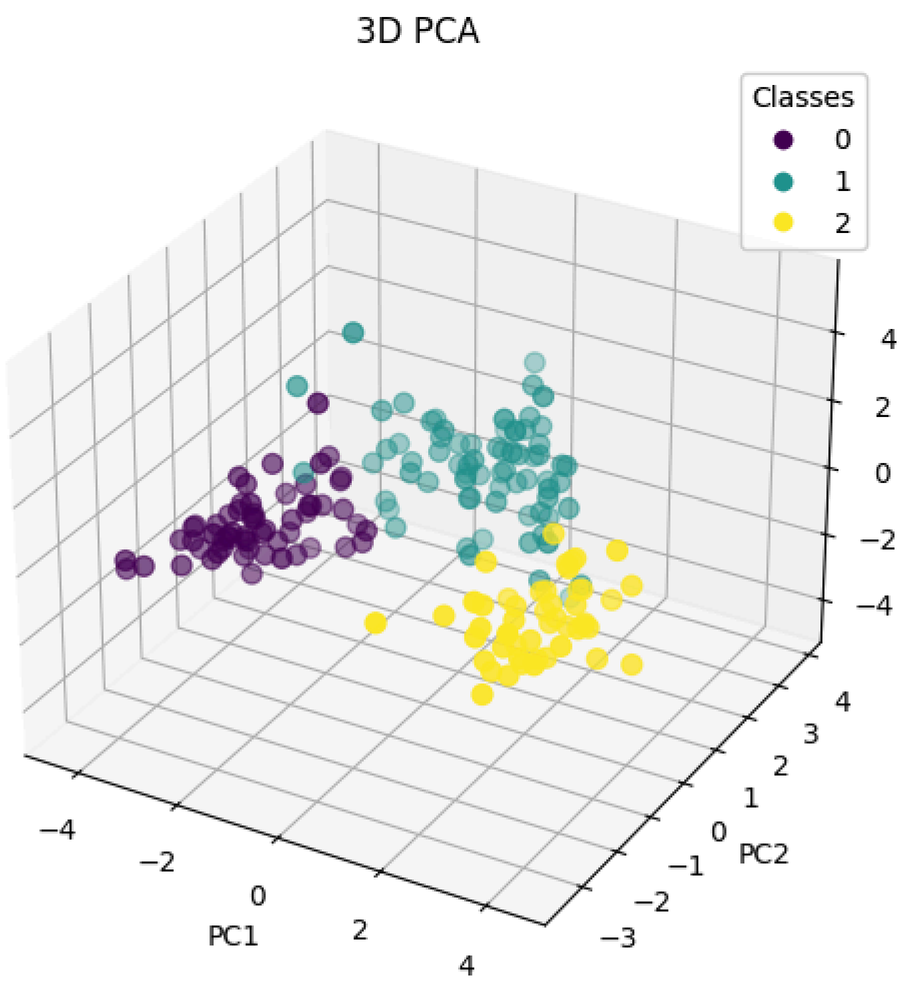
---

(6) 採取三個主成分，並繪製立體圖:

- 此圖為原始資料未做標準化得出的結果

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.datasets import load_wine
from sklearn.decomposition import PCA

# Load the wine dataset
X, y = load_wine(return_X_y=True)

# unscale data Perform PCA
pca = PCA(n_components=3)
Z1 = pca.fit_transform(X)
```

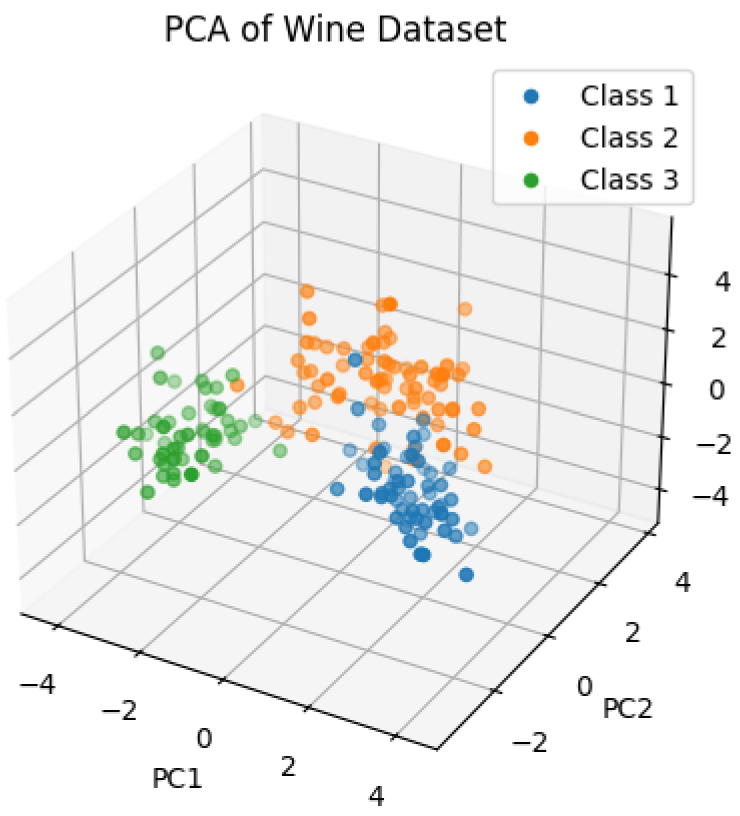
```
#scale data preform PCA
scaler = StandardScaler()
scaler.fit(ratings_)
ratings_ = scaler.transform(ratings_)
pca = PCA().fit(ratings_)
Z1 = pca.transform(ratings_)

# Plot 3D figure
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

# Scatter plot
scatter = ax.scatter(Z1[:,0], Z1[:,1], Z1[:,2], c=y, cmap='viridis', s=50)
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
ax.set_title('3D PCA')

# Legend
legend = ax.legend(*scatter.legend_elements(), title='Classes')
ax.add_artist(legend)
plt.show()
```



可以觀察到，第一個主成份的級距與另外兩座標軸相差極大，因此將資料標準化後再次觀察

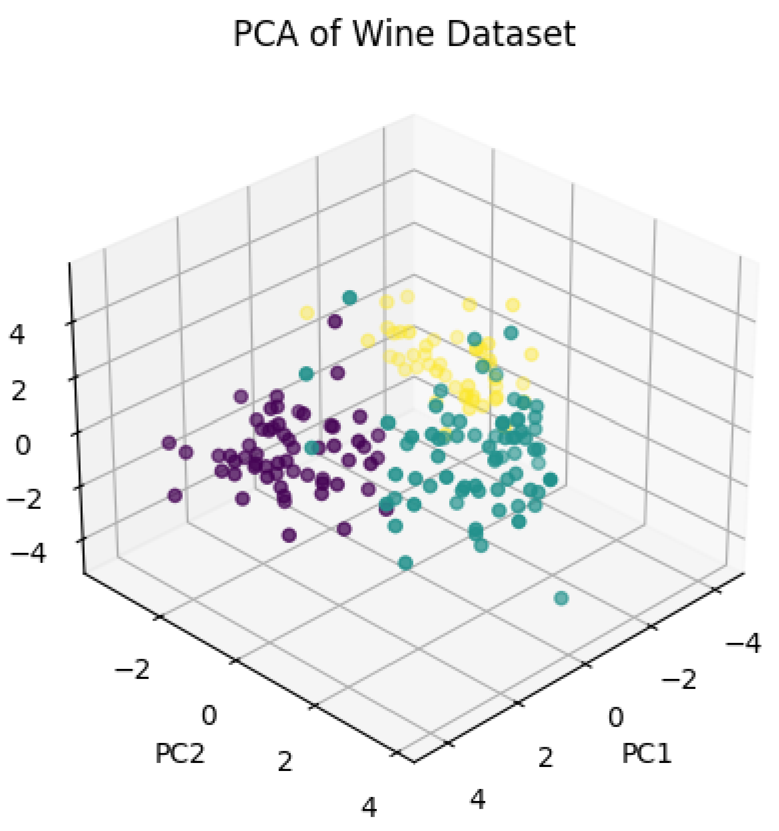
In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

```python
from sklearn.datasets import load_wine
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Load the wine dataset
X, y = load_wine(return_X_y=True)

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Perform PCA
pca = PCA(n_components=3)
Z1 = pca.fit_transform(X_scaled)

# Create a 3D plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plot each class separately
for i in np.unique(y):
    ax.scatter(Z1[y == i, 0], Z1[y == i, 1], Z1[y == i, 2], label=f'Class {i+1}'

ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
ax.set_title('PCA of Wine Dataset')
ax.legend()
plt.show()
```



做完資料標準化後，此時三個主成份級距接近，較能觀察出資料分群的概況，並嘗試轉化圖形角度，找出最適合看出分群概況的3D立體圖

In [ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.datasets import load_wine
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Load the wine dataset
X, y = load_wine(return_X_y=True)

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Perform PCA
pca = PCA(n_components=3)
Z1 = pca.fit_transform(X_scaled)

# Plot 3D scatter plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Scatter plot
ax.scatter(Z1[:, 0], Z1[:, 1], Z1[:, 2], c=y, cmap='viridis', marker='o')

# Set labels and title
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
ax.set_title('PCA of Wine Dataset')


# Rotate the plot for better viewing angle
ax.view_init(elev=30, azim=45)
plt.show()
```



上圖的角度最能明確觀察到分群的狀況

(7) Z_1 與 Z_2 都是從原變數組合而成的新變數，可否從 Z_1 與 Z_2 的組成係數，看出原變數哪個比較重要？哪個比較不重要？若再與原變數間的相關係數圖對照，是否透露相同的訊息。

```
In [ ]: import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.decomposition import PCA
        pca = PCA(n_components=1).fit(X) # 取 第 一 個 特 徵 向 量
        pca = PCA().fit(ratings_) # 進 行 主 成 分 分 析
        #print(pca.explained_variance_ratio_) # 共 變 異 矩 陣 特 徵 值 佔 比
        # print(pca.explained_variance_) # 共 變 異 矩 陣 的 特 徵 值
        # print(pca.components_) # 共 變 異 矩 陣 的 特 徵 向 量
        eigvals = pca.explained_variance_
        #print(eigvals)
        eigvecs = pca.components_.T # by column [v1 v2]
        print(eigvecs)
```

```
[[-0.13632501 -0.48416087 -0.20740081 -0.08191848 -0.25089415 -0.13517139
  -0.09269887 -0.42154435 -0.45019071  0.31127983 -0.22154641 -0.26411262
  -0.05610645  0.09062605]
 [ 0.22267638 -0.22359095  0.08879606  0.46988824 -0.18860015 -0.59841948
   0.3743698  -0.08757556 -0.00602569 -0.32592413  0.06839251  0.1192121
   0.06675544 -0.02522531]
 [-0.00225793 -0.31585588  0.62610236 -0.24984122 -0.0935236  -0.10799983
  -0.16708856  0.17208034  0.26249446 -0.12452347 -0.49452428 -0.04502305
  -0.19201787 -0.00163582]
 [ 0.22429849  0.01161574  0.6119896   0.07199322  0.0465675   0.08811224
  -0.26872469 -0.41324857 -0.11863342  0.15716811  0.47461722 -0.06131271
   0.20007784 -0.09536107]
 [-0.12463016 -0.30055143  0.13098458 -0.16321412  0.77833048 -0.14483831
   0.32957951  0.14881189 -0.25253628  0.12773363  0.07119731  0.06116074
   0.05829909  0.02230075]
 [-0.35926404 -0.06711983  0.14650775  0.19098521 -0.14466563  0.14809748
  -0.03789829  0.36343884 -0.40637354 -0.30772263  0.29740957 -0.30087591
  -0.35952714 -0.25303779]
 [-0.39071171  0.00131345  0.15096275  0.14461667 -0.11200553  0.06247252
  -0.06773223  0.175405   -0.09091933 -0.14044    -0.03219187 -0.05001396
   0.59834288  0.60190917]
 [ 0.2670012  -0.0269887   0.16997551 -0.32801272 -0.43257916  0.25868639
   0.61111195  0.23075135 -0.15912282  0.24054263  0.12200984  0.04266558
   0.06403952  0.08223093]
 [-0.2790625  -0.04122256  0.14987959  0.46275771  0.0915882   0.46627764
   0.42292282 -0.3437392   0.26578679  0.10869629 -0.23292405 -0.09334264
  -0.11013538 -0.05864198]
 [ 0.08931829 -0.52978274 -0.1372663   0.07211248 -0.0462696   0.42525454
  -0.18613617  0.04069617 -0.07526459 -0.21704255  0.01972448  0.59795428
   0.15917751 -0.17882114]
 [-0.27682265  0.27790735  0.08532854 -0.43466618 -0.02986657 -0.01565089
   0.19204101 -0.48362564 -0.21241681 -0.50966073 -0.06140493  0.25774292
  -0.04923091 -0.02258256]
 [-0.35052618  0.16277625  0.16620436  0.15672341 -0.14419358 -0.21770365
  -0.0785098   0.06865116 -0.08426484  0.45570504  0.06646166  0.61109218
  -0.32941979  0.13509216]
 [-0.26951525 -0.36605886 -0.12668685 -0.2557949  -0.08440794 -0.0665655
   0.0542037  -0.11146671  0.54490539 -0.04620802  0.55130818 -0.07268036
  -0.17322892  0.21604362]
 [ 0.39366953 -0.00569041  0.00121795  0.12246373  0.15758395  0.20033864
  -0.05938234 -0.07179553 -0.16236882 -0.19899373  0.01444169  0.01575769
  -0.49224318  0.66904528]]
```

結論:

由上結果可以看出，透過觀察組成的係數，可以清楚理解原變數在建構Z_1 Z_2新變數時的重要性。係數的絕對值越大，則表示原變數的貢獻越大；相反的，如果組成係數趨近於零，則表示原始變數幾乎沒貢獻，在數據解釋中較不重要。而相關係數可以更直觀的理解變數之間的關係；因此，綜合考慮兩點才能更更全面的理解數據。

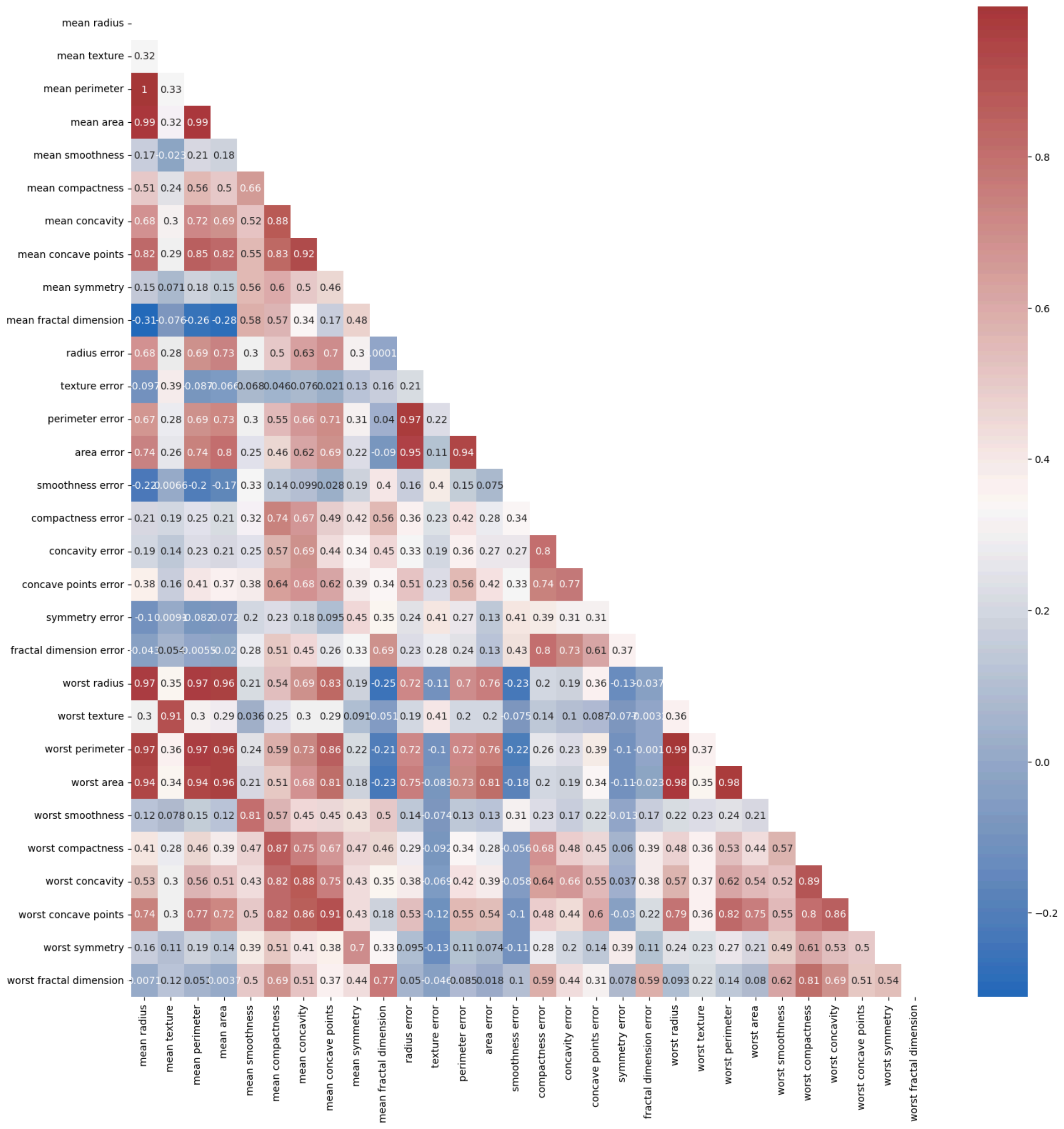
第 2 題：

有一組關於乳癌患者腫瘤的影像量測資料，資料內容包括30個變數，樣本數為 569 位患者，區分為兩個群組，分別是 Malignant（惡性腫瘤）與 Benign（良性腫瘤）。利用這組資料回答下列問題：

(1) 繪製變數間的相關係數圖，以觀察變數間是否存在相關性。

- 在sklearn套件中下載breast_cancer資料
- 由下圖可知，方框越紅代表越最高度相關

In [ ]:
```python
import pandas as pd
import numpy as np
import seaborn
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
# Load the breast cancer dataset
X, y = load_breast_cancer(return_X_y=True, as_frame=True)
labels = X.columns
# Plot the correlation matrix
plt.figure(figsize = (18, 18))
R = X.corr()
mask = np.triu(np.ones_like(R, dtype=bool)) # diagonal mask
seaborn.heatmap(R, annot=True, mask = mask, cmap='vlag')
plt.show()
```



結論:

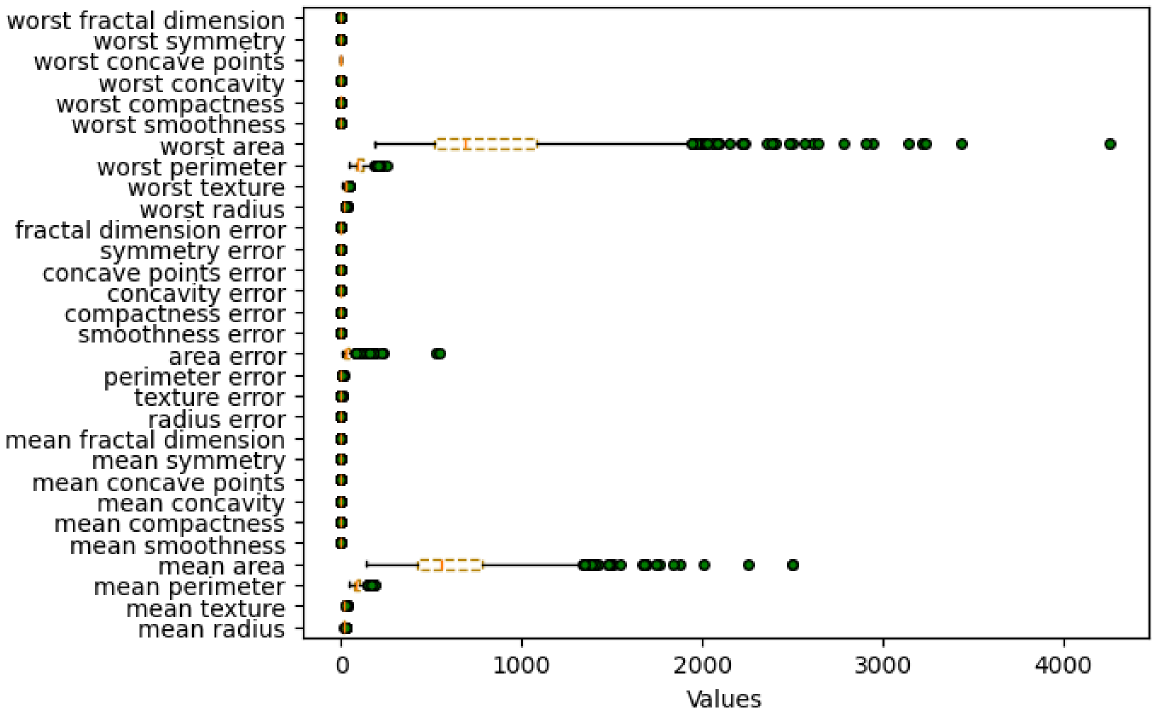
- 變數很多，故須將圖片放大才能清楚呈現變數之間的相關性
- 相關係數矩陣為對稱矩陣，因此僅呈現下半部。

---

(2)繪製含每個化學成分變數盒鬚圖，觀察每個變數的 scaling，分別畫為標準化與標準化。

- 下圖為未標準化的資料
- breast_canser_data_array為未標準化資料矩陣

In [ ]:
```
from sklearn.datasets import load_breast_cancer
import numpy as np
breast_canser = load_breast_cancer(as_frame=True)
breast_canser_data =  breast_canser.data
breast_canser_target = breast_canser.target
breast_canser_data_array = np.array(breast_canser_data)
breast_canser_data_cate = breast_canser_data.columns
fig, ax = plt.subplots()
boxprops = dict(linestyle = '--', linewidth = 1, \
color = 'darkgoldenrod')
flierprops = dict(marker='o', markerfacecolor = 'green',
markersize = 4, linestyle = 'none')
ax.boxplot(breast_canser_data_array
, boxprops = boxprops, \
flierprops = flierprops, \
labels = breast_canser_data_cate
, vert = False)
ax.set_xlabel('Values')
plt.show()
```

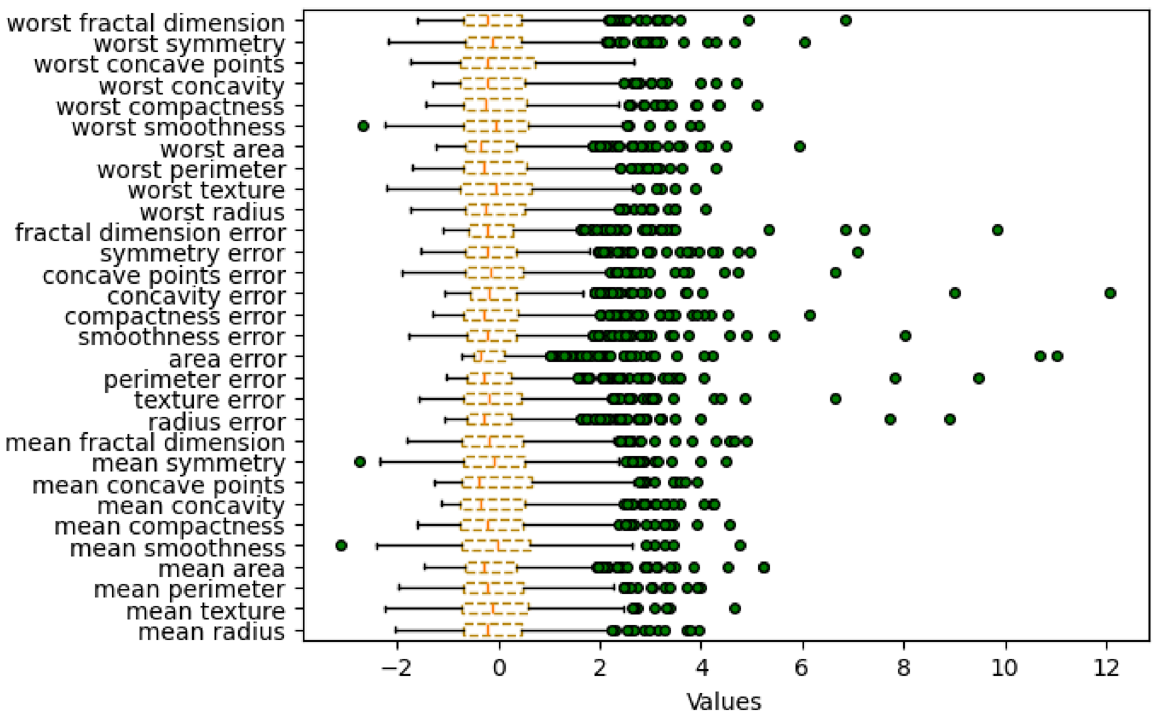


結論:

- 未標準化的資料，明顯發現需要標準化才能有相似的級距
- 可以觀察到 worst area 與mean area 這兩項變數的值特別不平衡

下圖為已標準化資料的盒鬚圖

In [ ]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(breast_canser_data_array)
breast_canser_data_array_ = scaler.transform(breast_canser_data_array)
breast_canser_data_cate = breast_canser_data.columns
fig, ax = plt.subplots()
boxprops = dict(linestyle = '--', linewidth = 1, \
color = 'darkgoldenrod')
flierprops = dict(marker='o', markerfacecolor = 'green',
markersize = 4, linestyle = 'none')
ax.boxplot(breast_canser_data_array_, boxprops = boxprops, \
flierprops = flierprops, \
labels = breast_canser_data_cate, vert = False)
ax.set_xlabel('Values')
plt.show()
```

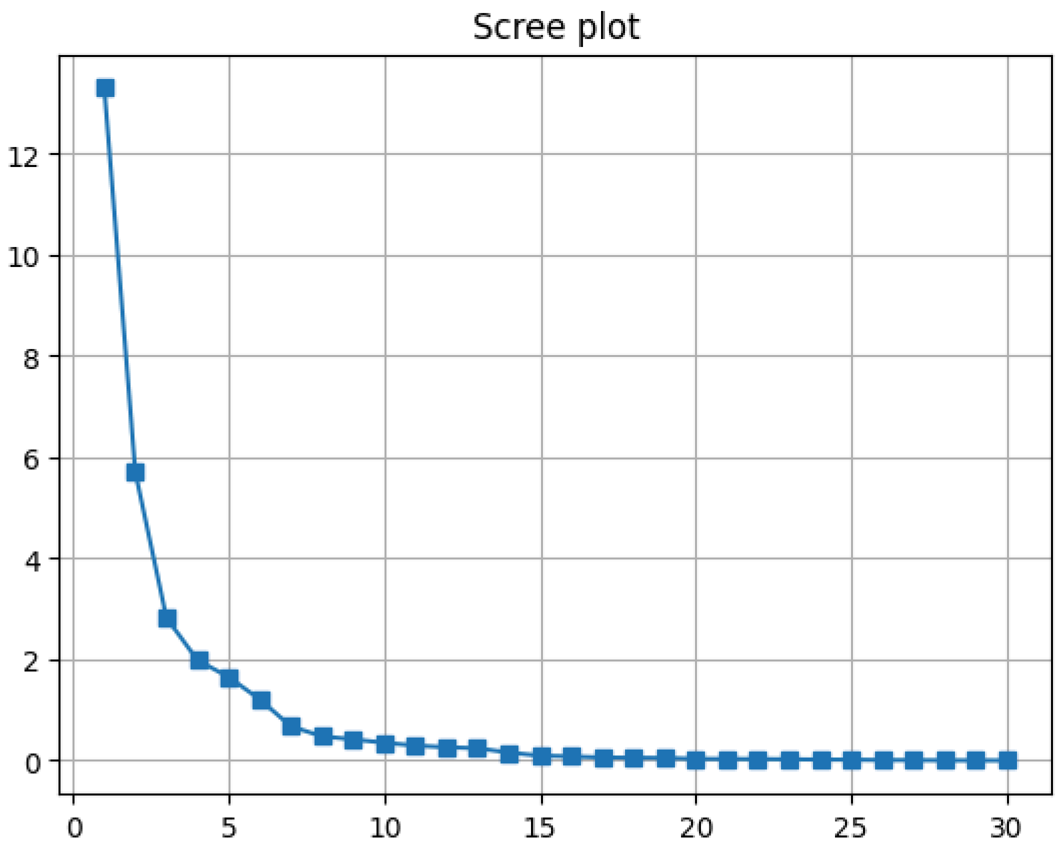


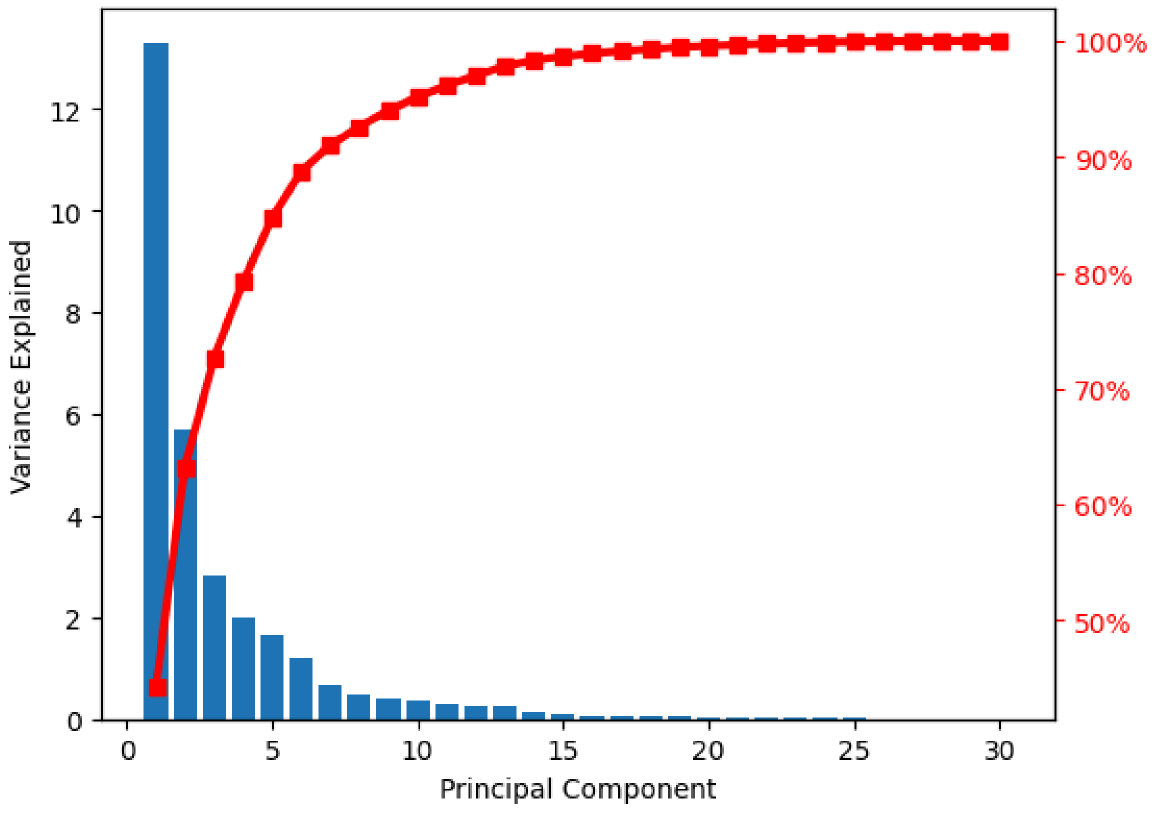
結論:在資料分析前都需要資料前處理，以確保不會有極端不平衡的狀況出現

(3) 進行主成分分析，繪製特徵值由大而小的分布與 scree plot與累積百分比的 pareto plot。

In [ ]:
```
import numpy as np
from numpy.linalg import eig
from numpy.linalg import inv
from matplotlib.ticker import PercentFormatter
Sx = np.cov(breast_canser_data_array_.T, bias=False)
w, v = eig(Sx)
idx = np.argsort(-w) #sort eigenvalues in descending order
# idx = np.argsort(w)[::-1]
eigvals = w[idx]
eigvecs = v[:, idx]
```

```
#開始作主成分分析圖
plt.figure()
x = np.arange(1, 1+len(eigvals))
plt.plot(x, eigvals, marker='s')
plt.title('Scree plot')
plt.grid(True)
plt.show()
fig, ax = plt.subplots()
x = np.arange(1, 1+len(eigvals))
ax.bar(x, eigvals)
ax2 = ax.twinx()
ax2.plot(x, eigvals.cumsum()/eigvals.sum()*100, \
marker='s', color='red', lw=3)
ax2.tick_params(axis='y', colors='red')
ax2.yaxis.set_major_formatter(PercentFormatter())
ax.set_xlabel('Principal Component')
ax.set_ylabel('Variance Explained')
plt.show()
```





結論:

可以觀察到約採用五個主成分，其解釋變異量就高達85%，因此這是一筆適合用PCA維度縮減的資料檔。

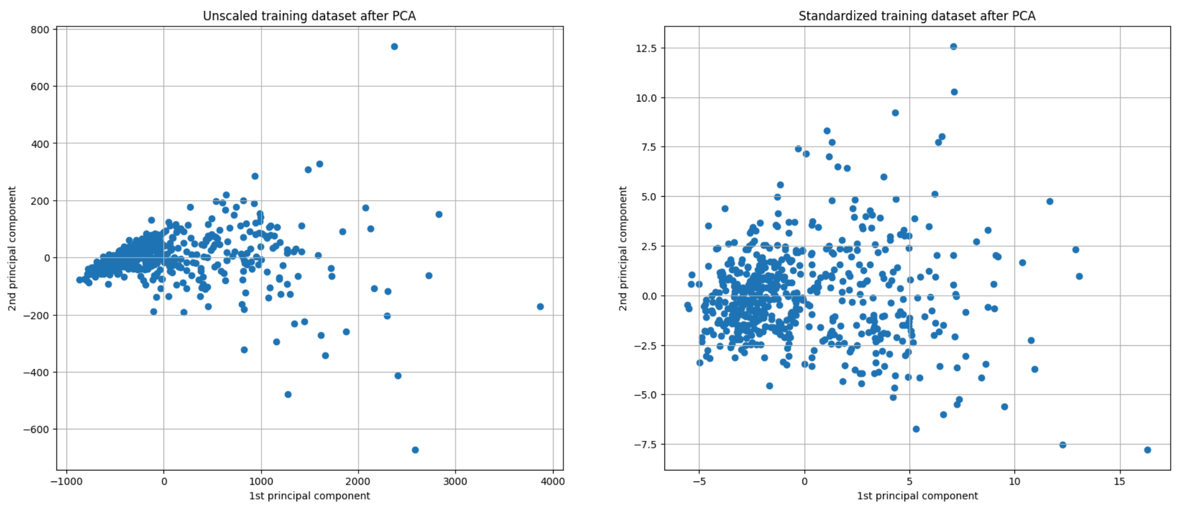
---

(4) 利用主成分分析取得前兩項成分，並繪製其散布圖

- 左圖為資料不做標準化，取前兩大主成分得到的散佈圖。
- 右圖為資料標準化後，取兩個主成分得到的散佈圖。

```
In [ ]:
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# 對未標準化的資料做PCA
pca = PCA().fit(breast_canser_data_array)
Z = pca.transform(breast_canser_data_array)
# 對標準化的資料做PCA
scaler = StandardScaler()
scaler.fit(breast_canser_data_array_)
ratings_ = scaler.transform(breast_canser_data_array_)
pca = PCA().fit(breast_canser_data_array_)
Z1 = pca.transform(breast_canser_data_array_)
# 做圖
fig, ax = plt.subplots(1, 2, figsize = (20, 8))
ax[0].scatter(Z[:,0], Z[:,1])
ax[0].set_xlabel('1st principal component')
ax[0].set_ylabel('2nd principal component')
ax[0].set_title("Unscaled training dataset after PCA")
ax[0].grid(True)
ax[1].scatter(Z1[:,0], Z1[:,1])
```

```
ax[1].set_xlabel('1st principal component')
ax[1].set_ylabel('2nd principal component')
ax[1].grid(True)
ax[1].set_title("Standardized training dataset after PCA")
plt.show()
```



結論:

標準化後兩個主成分的級距較一致。

較第一題不太一樣的是，標準化後的資料進行主成分分析仍然觀察不太到點分群的狀況。

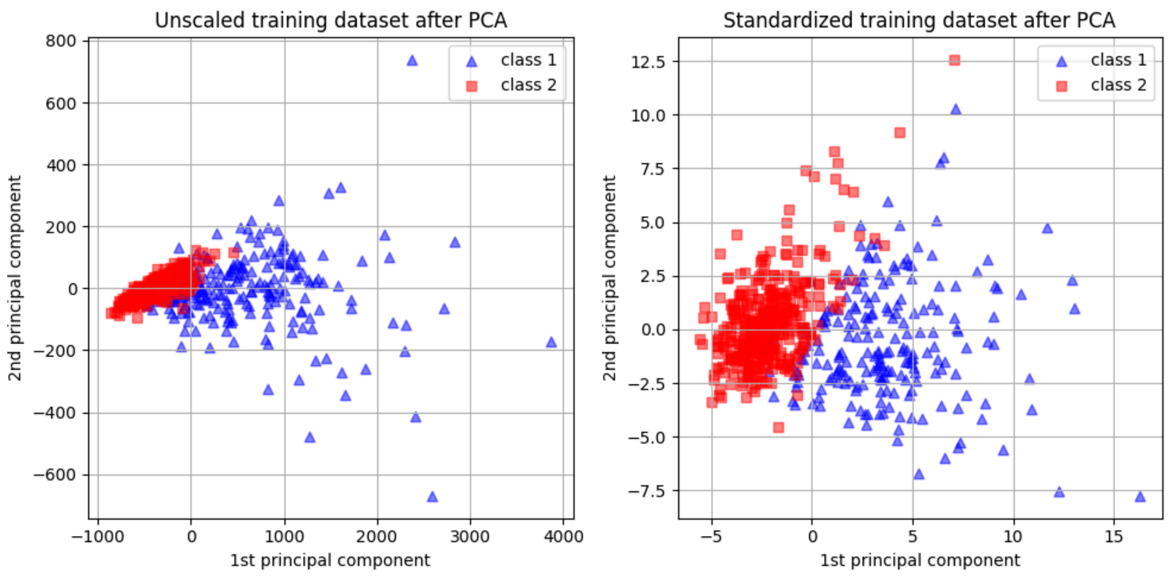
因次必須往下將CLASSES以不同顏色標出方可觀察。

---

(5) 再依據每個資料的標籤，為每個在散布圖上的資料點依據標籤塗上顏色

- 左圖為資料不做標準化，得到標籤顏色分類的散佈圖。
- 右圖為資料標準化後，得到標籤顏色分類的的散佈圖。

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# 做圖
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(10, 5))

target_classes = range(0, 3)
colors = ("blue", "red")
markers = ("^", "s")

for target_class, color, marker in zip(target_classes, colors, markers):
    ax1.scatter(
        x=Z[y == target_class, 0],
        y=Z[y == target_class, 1],
        color=color,
        label=f"class {target_class+1}",
        alpha=0.5,
        marker=marker,
    )

    ax2.scatter(
        x=Z1[y == target_class, 0],
        y=Z1[y == target_class, 1],
        color=color,
        label=f"class {target_class+1}",
        alpha=0.5,
        marker=marker,
    )

ax1.set_title("Unscaled training dataset after PCA")
ax2.set_title("Standardized training dataset after PCA")

for ax in (ax1, ax2):
    ax.set_xlabel("1st principal component")
    ax.set_ylabel("2nd principal component")
    ax.legend(loc="upper right")
    ax.grid()

_ = plt.tight_layout()
```



結論:

做完標準化後，兩個類別的資料分散較平均，兩個主成分的級距也較為相似。

而未做標準化的資料分配不平均，兩個主成分的級距也不相似

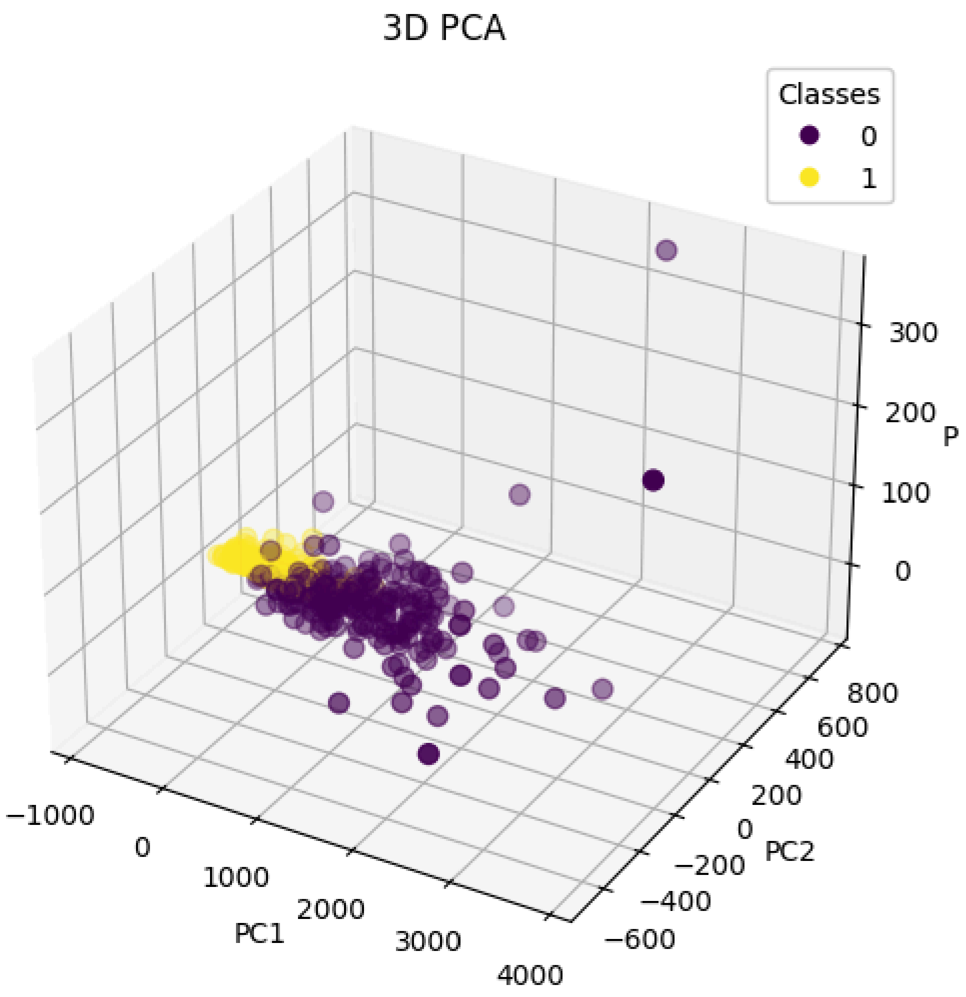
---

(6) 採取三個主成分，並繪製立體圖:

- 此圖為原始資料未做標準化得出的結果

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.decomposition import PCA
from sklearn.datasets import load_breast_cancer
# Load the breast cancer dataset
X, y = load_breast_cancer(return_X_y=True, as_frame=True)
# unscale data Perform PCA
pca = PCA(n_components=3)
Z1 = pca.fit_transform(X)
# Plot 3D figure
```

```
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

# Scatter plot
scatter = ax.scatter(Z1[:,0], Z1[:,1], Z1[:,2], c=y, cmap='viridis', s=50)
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
ax.set_title('3D PCA')

# Legend
legend = ax.legend(*scatter.legend_elements(), title='Classes')
ax.add_artist(legend)
plt.show()
```



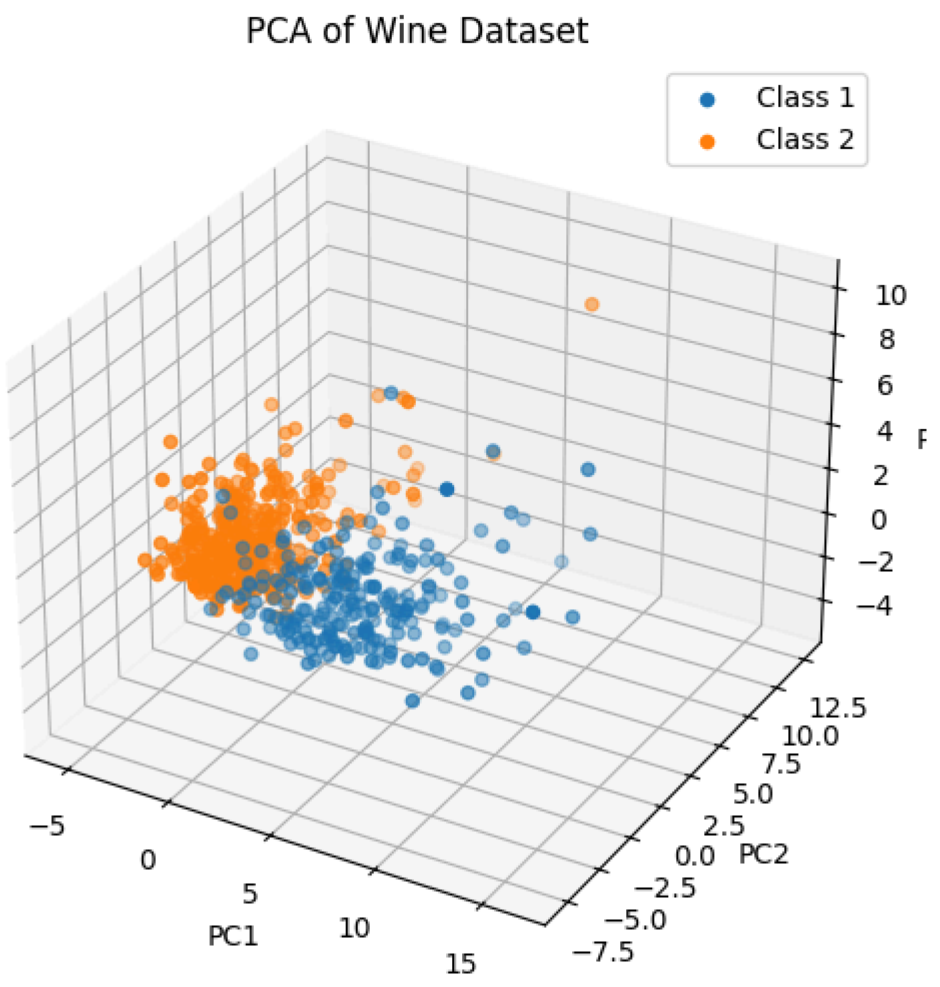
發現三個主成分的級距都不一致，也不易觀察到分群的情況

- 標準化後得出的結果

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.datasets import load_breast_cancer
# Load the breast cancer dataset
X, y = load_breast_cancer(return_X_y=True, as_frame=True)
```

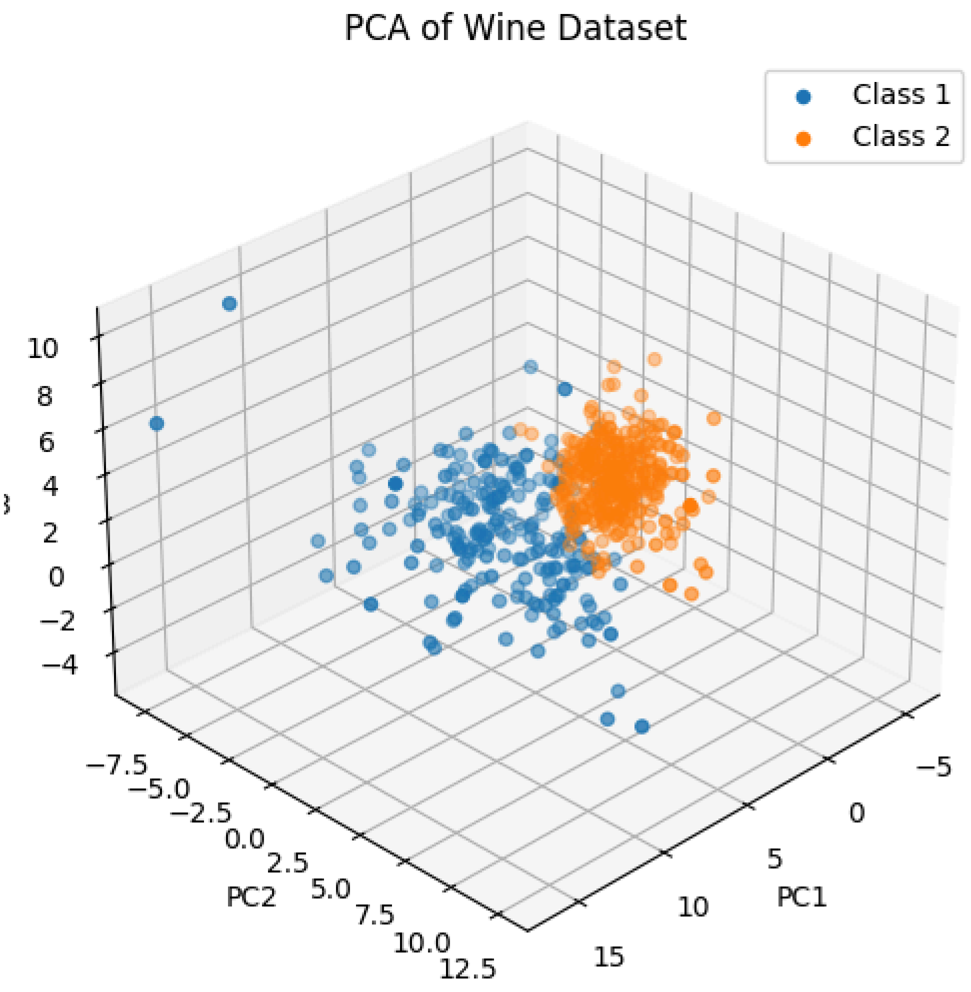
```
#scale data preform PCA
scaler = StandardScaler()
scaler.fit(breast_canser_data_array_)
ratings_ = scaler.transform(breast_canser_data_array_)
pca = PCA(n_components=3).fit(breast_canser_data_array_)
Z1 = pca.transform(breast_canser_data_array_)

# Create a 3D plot
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

# Plot each class separately
for i in np.unique(y):
    ax.scatter(Z1[y == i, 0], Z1[y == i, 1], Z1[y == i, 2], label=f'Class {i+1}'

ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
ax.set_title('PCA of Wine Dataset')
ax.legend()
plt.show()
```



級距較一致，也較能區分兩個類別的資料點

- 轉換角度得出的結果

```
In [ ]: from sklearn.datasets import load_breast_cancer
        # Load the breast cancer dataset
        X, y = load_breast_cancer(return_X_y=True, as_frame=True)

        #scale data preform PCA
        scaler = StandardScaler()
        scaler.fit(breast_canser_data_array_)
        ratings_ = scaler.transform(breast_canser_data_array_)
        pca = PCA(n_components=3).fit(breast_canser_data_array_)
        Z1 = pca.transform(breast_canser_data_array_)

        # Create a 3D plot
        fig = plt.figure(figsize=(8, 6))
        ax = fig.add_subplot(111, projection='3d')

        # Plot each class separately
        for i in np.unique(y):
            ax.scatter(Z1[y == i, 0], Z1[y == i, 1], Z1[y == i, 2], label=f'Class {i+1}'

        ax.set_xlabel('PC1')
        ax.set_ylabel('PC2')
        ax.set_zlabel('PC3')
        ax.set_title('PCA of Wine Dataset')
        ax.legend()
        ax.view_init(elev=30, azim=45)
        plt.show()
```



結論:

因此未來在做主成分分析時，需要先將資料做標準化動作，然後觀察取幾個主成分最能滿足

維持原數據類別間的最大變異量，同時達到維度縮減的目的。

(7) Z_1 與 Z_2 都是從原變數組合而成的新變數，可否從 Z_1 與 Z_2 的組成係數，看出原變數哪個比較重要？哪個比較不重要？若再與原變數間的相關係數圖對照，是否透露相同的訊息。

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
pca = PCA(n_components=1).fit(X) # 取 第 一 個 特 徵 向 量
pca = PCA().fit(breast_canser_data_array_) # 進 行 主 成 分 分 析
#print(pca.explained_variance_ratio_) # 共 變 異 矩 陣 特 徵 值 佔 比
# print(pca.explained_variance_) # 共 變 異 矩 陣 的 特 徵 值
# print(pca.components_) # 共 變 異 矩 陣 的 特 徵 向 量
eigvals = pca.explained_variance_
#print(eigvals)
eigvecs = pca.components_.T # by column [v1 v2]
print(eigvecs)
```

```
[[ 2.18902444e-01 -2.33857132e-01 -8.53124284e-03  4.14089623e-02
   3.77863538e-02  1.87407904e-02 -1.24088340e-01 -7.45229622e-03
  -2.23109764e-01  9.54864432e-02  4.14714866e-02  5.10674568e-02
   1.19672116e-02  5.95061348e-02  5.11187749e-02 -1.50583883e-01
   2.02924255e-01  1.46712338e-01 -2.25384659e-01 -4.96986642e-02
  -6.85700057e-02  7.29289034e-02 -9.85526942e-02  1.82579441e-01
   1.92264989e-02 -1.29476396e-01 -1.31526670e-01  2.11194013e-01
   2.11460455e-01 -7.02414091e-01]
 [ 1.03724578e-01 -5.97060883e-02  6.45499033e-02 -6.03050001e-01
  -4.94688505e-02 -3.21788366e-02  1.13995382e-02  1.30674825e-01
   1.12699390e-01  2.40934066e-01 -3.02243402e-01  2.54896423e-01
   2.03461333e-01 -2.15600995e-02  1.07922421e-01 -1.57841960e-01
  -3.87061187e-02 -4.11029851e-02 -2.97886446e-02 -2.44134993e-01
   4.48369467e-01  9.48006326e-02 -5.54997454e-04 -9.87867898e-02
  -8.47459309e-02 -2.45566636e-02 -1.73573093e-02 -6.58114593e-05
  -1.05339342e-02 -2.73661018e-04]
 [ 2.27537293e-01 -2.15181361e-01 -9.31421972e-03  4.19830991e-02
   3.73746632e-02  1.73084449e-02 -1.14477057e-01 -1.86872582e-02
  -2.23739213e-01  8.63856150e-02  1.67826374e-02  3.89261058e-02
   4.41095034e-02  4.85138123e-02  3.99029358e-02 -1.14453955e-01
   1.94821310e-01  1.58317455e-01 -2.39595276e-01 -1.76650122e-02
  -6.97690429e-02  7.51604777e-02 -4.02447050e-02  1.16648876e-01
  -2.70154137e-02 -1.25255946e-01 -1.15415423e-01  8.43382663e-02
   3.83826098e-01  6.89896968e-01]
 [ 2.20994985e-01 -2.31076711e-01  2.86995259e-02  5.34337955e-02
   1.03312514e-02 -1.88774796e-03 -5.16534275e-02  3.46736038e-02
  -1.95586014e-01  7.49564886e-02  1.10169643e-01  6.54375082e-02
   6.73757374e-02  1.08308292e-02 -1.39669069e-02 -1.32448032e-01
   2.55705763e-01  2.66168105e-01  2.73221894e-02 -9.01437617e-02
  -1.84432785e-02  9.75657781e-02  7.77727342e-03 -6.98483369e-02
   2.10040780e-01  3.62727403e-01  4.66612477e-01 -2.72508323e-01
  -4.22794920e-01  3.29473482e-02]
 [ 1.42589694e-01  1.86113023e-01 -1.04291904e-01  1.59382765e-01
  -3.65088528e-01 -2.86374497e-01 -1.40668993e-01 -2.88974575e-01
   6.42472194e-03 -6.92926813e-02 -1.37021842e-01  3.16727211e-01
   4.55736020e-02  4.45064860e-01  1.18143364e-01 -2.04613247e-01
   1.67929914e-01 -3.52226802e-01  1.64565843e-01  1.71009601e-02
  -1.19491747e-01  6.38229479e-02 -2.06657211e-02 -6.86974224e-02
  -2.89548850e-02 -3.70036864e-02  6.96899233e-02  1.47926883e-03
  -3.43466700e-03  4.84745766e-03]
 [ 2.39285354e-01  1.51891610e-01 -7.40915709e-02  3.17945811e-02
   1.17039713e-02 -1.41309489e-02  3.09184960e-02 -1.51396350e-01
  -1.67841425e-01  1.29362000e-02 -3.08009633e-01 -1.04017044e-01
   2.29281304e-01  8.10105720e-03 -2.30899962e-01  1.70178367e-01
  -2.03077075e-02  7.79413843e-03 -2.84222358e-01  4.88686329e-01
   1.92621396e-01 -9.80775567e-02  5.23603957e-02  1.04135518e-01
  -3.96623231e-01  2.62808474e-01  9.77487054e-02 -5.46276696e-03
  -4.10167739e-02 -4.46741863e-02]
 [ 2.58400481e-01  6.01653628e-02  2.73383798e-03  1.91227535e-02
   8.63754118e-02 -9.34418089e-03 -1.07520443e-01 -7.28272853e-02
   4.05910064e-02 -1.35602298e-01  1.24190245e-01  6.56534798e-02
   3.87090806e-01 -1.89358699e-01  1.28283732e-01  2.69470206e-01
  -1.59835337e-03 -2.69681105e-02 -2.26636013e-03 -3.33870858e-02
   5.57175335e-03 -1.85212003e-01  3.24870378e-01 -4.47410568e-02
   9.69773167e-02 -5.48876170e-01  3.64808397e-01  4.55386379e-02
  -1.00147876e-02 -2.51386661e-02]
 [ 2.60853758e-01 -3.47675005e-02 -2.55635406e-02  6.53359443e-02
  -4.38610252e-02 -5.20499505e-02 -1.50482214e-01 -1.52322414e-01
  -1.11971106e-01  8.05452775e-03 -7.24460264e-02  4.25892667e-02
   1.32138097e-01 -2.44794768e-01  2.17099194e-01  3.80464095e-01
   3.45095087e-02 -8.28277367e-02  1.54972363e-01 -2.35407606e-01
  -9.42381870e-03 -3.11852431e-01 -5.14087968e-02 -8.40276972e-02
   1.86451602e-01  3.87643377e-01 -4.54699351e-01 -8.88309714e-03
  -4.20694931e-03  1.07726530e-03]
 [ 1.38166959e-01  1.90348770e-01 -4.02399363e-02  6.71249840e-02
  -3.05941428e-01  3.56458461e-01 -9.38911345e-02 -2.31530989e-01
   2.56040084e-01  5.72069479e-01  1.63054081e-01 -2.88865504e-01
   1.89933673e-01  3.07388563e-02  7.39617071e-02 -1.64661588e-01
  -1.91737848e-01  1.73397790e-01  5.88111647e-02  2.60691555e-02
  -8.69384844e-02 -1.84067326e-02 -5.12005770e-02 -1.93394733e-02
   2.45836949e-02 -1.60440385e-02 -1.51648349e-02  1.43302642e-03
  -7.56986244e-03  1.28037941e-03]
 [ 6.43633464e-02  3.66575471e-01 -2.25740897e-02  4.85867649e-02
  -4.44243602e-02 -1.19430668e-01  2.95760024e-01 -1.77121441e-01
  -1.23740789e-01  8.11032072e-02 -3.80482687e-02  2.36358988e-01
   1.06239082e-01 -3.77078865e-01 -5.17975705e-01 -4.07927860e-02
   5.02252456e-02  8.78673570e-02  5.81570509e-02 -1.75637222e-01
  -7.62718362e-02  2.87868885e-01 -8.46898562e-02  1.33260547e-01
   2.07221864e-01 -9.74048386e-02 -1.01244946e-01 -6.31168651e-03
   7.30143287e-03  4.75568480e-03]
 [ 2.05978776e-01 -1.05552152e-01  2.68481387e-01  9.79412418e-02
  -1.54456496e-01 -2.56032561e-02  3.12490037e-01  2.25399674e-02
   2.49985002e-01 -4.95475941e-02 -2.53570194e-02 -1.66879153e-02
  -6.81952298e-02  1.03474126e-02  1.10050711e-01  5.89057190e-02
  -1.39396866e-01 -2.36216532e-01 -1.75883308e-01 -9.08005031e-02
   8.63867747e-02 -1.50274681e-01 -2.64125317e-01  5.58701567e-01
   1.74930429e-01  4.99770798e-02  2.12982901e-01 -1.92223890e-01
   1.18442112e-01  8.71109373e-03]
 [ 1.74280281e-02  8.99796818e-02  3.74633665e-01 -3.59855528e-01
  -1.91650506e-01 -2.87473145e-02 -9.07553556e-02 -4.75413139e-01
  -2.46645397e-01 -2.89142742e-01  3.44944458e-01 -3.06160423e-01
  -1.68222383e-01 -1.08493473e-02 -3.27527212e-02 -3.45004006e-02
   4.39630156e-02 -9.85866201e-03 -3.60098518e-02 -7.16599878e-02
   2.17071967e-01  4.84569345e-02 -8.73880467e-04 -2.42672970e-02
  -5.69864778e-02 -1.12372419e-02 -1.00928890e-02 -5.62261069e-03
  -8.77627920e-03  1.07103919e-03]
 [ 2.11325916e-01 -8.94572342e-02  2.66645367e-01  8.89924146e-02
  -1.20990220e-01  1.81071500e-03  3.14640390e-01 -1.18966905e-02
   2.27154024e-01 -1.14508236e-01 -1.67318771e-01 -1.01446828e-01
  -3.78439858e-02 -4.55237175e-02  8.26808881e-03  2.65166513e-02
  -2.46356391e-02 -2.59288003e-02 -3.65701538e-01 -1.77250625e-01
  -3.04950158e-01  1.59352804e-01  9.00742110e-02 -5.16750385e-01
  -7.29276412e-02  1.03653282e-01  4.16915529e-02  2.63191868e-01
  -6.10021933e-03 -1.37293906e-02]
 [ 2.02869635e-01 -1.52292628e-01  2.16006528e-01  1.08205039e-01
  -1.27574432e-01 -4.28639079e-02  3.46679003e-01  8.58051345e-02
   2.29160015e-01 -9.19278886e-02  5.16194632e-02 -1.76792177e-02
   5.60649318e-02  8.35707181e-02  4.60243656e-02  4.11532265e-02
   3.34418173e-01  3.04906903e-01  4.16572314e-01  2.74201148e-01
   1.92587786e-01  6.42326151e-02  9.82150746e-02  2.24607172e-02
  -1.31850405e-01 -1.55304589e-01 -3.13358657e-01 -4.20681051e-02
  -8.59259138e-02 -1.10532603e-03]
 [ 1.45314521e-02  2.04430453e-01  3.08838979e-01  4.46641797e-02
  -2.32065676e-01 -3.42917393e-01 -2.44024056e-01  5.73410232e-01
  -1.41924890e-01  1.60884609e-01  8.42062106e-02 -2.94710053e-01
   1.50441434e-01 -2.01152530e-01 -1.85594647e-02 -5.80390613e-02
   1.39595006e-01 -2.31259943e-01  1.32600886e-02  9.00614773e-02
  -7.20987261e-02  5.05449015e-02 -5.98177179e-02 -1.56311888e-02
  -3.12107028e-02 -7.71755717e-03 -9.05215355e-03  9.79296328e-03
   1.77638619e-03  1.60821086e-03]
```

```
[ 1.70393451e-01  2.32715896e-01  1.54779718e-01 -2.74693632e-02
  2.79968156e-01  6.91975186e-02  2.34635340e-02  1.17460157e-01
 -1.45322810e-01  4.35048658e-02 -2.06885680e-01 -2.63456509e-01
  1.00401699e-02  4.91755932e-01 -1.68209315e-01  1.89830896e-01
 -8.24647717e-03  1.00474235e-01  2.42448176e-01 -4.61098220e-01
 -1.40386572e-01 -4.52876920e-02  9.10387102e-03  1.21777792e-01
 -1.73164553e-01 -4.97276317e-02  4.65360884e-02 -1.53955481e-02
  3.15813441e-03 -1.91562235e-03]
[ 1.53589790e-01  1.97207283e-01  1.76463743e-01  1.31687997e-03
  3.53982091e-01  5.63432386e-02 -2.08823790e-01  6.05665008e-02
  3.58107079e-01 -1.41276243e-01  3.49517943e-01  2.51146975e-01
  1.58783192e-01  1.34586924e-01 -2.50471408e-01 -1.25420649e-01
  8.46167156e-02 -1.95485228e-04 -1.26381025e-01  6.69461742e-02
  6.30479298e-02 -2.05212693e-01 -3.87542329e-01 -1.88205036e-01
 -1.59399802e-02  9.14549680e-02 -8.42247975e-02  5.82097800e-03
  1.60785207e-02  8.92652653e-03]
[ 1.83417397e-01  1.30321560e-01  2.24657567e-01  7.40673350e-02
  1.95548089e-01 -3.12244482e-02 -3.69645937e-01 -1.08319309e-01
  2.72519886e-01  8.62408470e-02 -3.42375908e-01 -6.45875122e-03
 -4.94026741e-01 -1.99666719e-01 -6.20793442e-02 -1.98810346e-01
  1.08132263e-01  4.60549116e-02  1.21642969e-02  6.88682942e-02
  3.43753236e-02 -7.25453753e-02  3.51755074e-01  1.09668978e-01
  1.29546547e-01 -1.79419192e-02 -1.11655093e-02 -2.90093001e-02
 -2.39377870e-02  2.16019727e-03]
[ 4.24984216e-02  1.83848000e-01  2.88584292e-01  4.40733510e-02
 -2.52868765e-01  4.90245643e-01 -8.03822539e-02  2.20149279e-01
 -3.04077200e-01 -3.16529830e-01 -1.87844043e-01  3.20571348e-01
  1.03327412e-02 -4.68643826e-02  1.13383199e-01 -1.57711497e-01
 -2.74059129e-01  1.87014764e-01  8.90392949e-02  1.07385289e-01
 -9.76995265e-02 -8.46544307e-02 -4.23628949e-02 -3.22620011e-03
  1.95149333e-02 -1.72678486e-02 -1.99759830e-02 -7.63652550e-03
 -5.22329189e-03 -3.29389752e-04]
[ 1.02568322e-01  2.80092027e-01  2.11503764e-01  1.53047496e-02
  2.63297438e-01 -5.31952674e-02  1.91394973e-01  1.11681884e-02
 -2.13722716e-01  3.67541918e-01  2.50624789e-01  2.76165974e-01
 -2.40458323e-01  1.45652466e-01  3.53232211e-01  2.68553878e-01
 -1.22733398e-01 -5.98230982e-02 -8.66008430e-02  2.22345297e-01
  6.28432814e-02  2.44705083e-01  8.57810992e-02 -7.51944193e-02
  8.41712034e-02  3.54889745e-02 -1.20365640e-02  1.97564555e-02
 -8.34191154e-03 -1.79895682e-03]
[ 2.27996634e-01 -2.19866379e-01 -4.75069900e-02  1.54172396e-02
 -4.40659209e-03 -2.90684919e-04 -9.70993602e-03  4.26194163e-02
 -1.12141463e-01  7.73616428e-02  1.05067333e-01  3.96796652e-02
 -1.37890527e-01  2.31012813e-02 -1.66567074e-01 -8.15605686e-02
 -2.40049982e-01 -2.16101353e-01 -1.36613039e-02 -5.62690874e-03
  7.29389953e-03 -9.62982088e-02 -5.56767923e-02  1.56830365e-01
 -7.07097238e-02 -1.97054744e-01 -1.78666740e-01  4.12639581e-01
 -6.35724917e-01  1.35643056e-01]
[ 1.04469325e-01 -4.54672983e-02 -4.22978228e-02 -6.32807885e-01
 -9.28834001e-02 -5.00080613e-02  9.87074388e-03  3.62516360e-02
  1.03341204e-01  2.95509413e-02  1.31572736e-02  7.97974499e-02
 -8.01454315e-02  5.34307917e-02 -1.01115399e-01  1.85557852e-01
  6.93651855e-02  5.83984505e-02  7.58669276e-02  3.00599798e-01
 -5.94440143e-01 -1.11112024e-01 -8.92289971e-03  1.18484602e-01
  1.18189721e-01  3.64694332e-02  2.14106944e-02 -3.90250926e-04
  1.72354925e-02 -1.02053601e-03]
[ 2.36639681e-01 -1.99878428e-01 -4.85465083e-02  1.38027944e-02
  7.45415100e-03  8.50098715e-03 -4.45726717e-04  3.05585340e-02
 -1.09614364e-01  5.05083335e-02  5.10762807e-02 -8.98773800e-03
 -9.69657077e-02  1.22193824e-02 -1.82755198e-01 -5.48570473e-02
 -2.34164147e-01 -1.88543592e-01 -9.08132490e-02  1.10038577e-02
 -9.20235990e-02  1.72216251e-02  6.33448296e-02 -2.37113167e-01
 -1.18034029e-01 -2.44103670e-01 -2.41031046e-01 -7.28680898e-01
  2.29218029e-02 -7.97438536e-02]
[ 2.24870533e-01 -2.19351858e-01 -1.19023182e-02  2.58947492e-02
 -2.73909030e-02 -2.51643821e-02  6.78316595e-02  7.93942456e-02
 -8.07324609e-02  6.99211523e-02  1.84598937e-01  4.80886567e-02
 -1.01160611e-01 -6.68546458e-03 -3.14993600e-01 -9.06533944e-02
 -2.73399584e-01 -1.42064856e-01  4.10047202e-01  6.00473870e-02
  1.46790132e-01 -9.69598236e-02  1.90889625e-01 -1.44063033e-01
  3.82899511e-02  2.31359525e-01  2.37162466e-01  2.38960316e-01
  4.44935933e-01 -3.97422838e-02]
[ 1.27952561e-01  1.72304352e-01 -2.59797613e-01  1.76522161e-02
 -3.24435445e-01 -3.69255370e-01 -1.08830886e-01  2.05852191e-01
  1.12315904e-01 -1.28304659e-01  1.43890349e-01  5.65148662e-02
 -2.05130344e-01  1.62235443e-01 -4.61258656e-02  1.45551659e-01
 -2.78030197e-01  5.01551675e-01 -2.34513845e-01 -1.29723903e-01
  1.64849237e-01 -6.82540931e-02  9.36901494e-02  1.09901386e-02
  4.79647647e-02  1.26024637e-02 -4.08535683e-02 -1.53524821e-03
  7.38549171e-03 -4.58327731e-03]
[ 2.10095880e-01  1.43593173e-01 -2.36075625e-01 -9.13284153e-02
  1.21804107e-01  4.77057929e-02  1.40472938e-01  8.40196588e-02
 -1.00677822e-01 -1.72133632e-01 -1.97420469e-01 -3.71662503e-01
  1.22793095e-02  1.66470250e-01  4.99560142e-02 -1.53734861e-01
 -4.03712272e-03 -7.35745143e-02 -2.02007041e-02  2.29280589e-01
  1.81374867e-01  2.96764124e-02 -1.47920925e-01 -1.86749953e-01
  6.24384938e-01 -1.00463424e-01 -7.05054136e-02  4.86918180e-02
  3.56690392e-06  1.28415624e-02]
[ 2.28767533e-01  9.79641143e-02 -1.73057335e-01 -7.39511797e-02
  1.88518727e-01  2.83792555e-02 -6.04880561e-02  7.24678714e-02
  1.61908621e-01 -3.11638520e-01  1.85016760e-01 -8.70345324e-02
  2.17984329e-01 -6.67989309e-02  2.04835886e-01 -2.15021948e-01
 -1.91313419e-01 -1.03907980e-01  4.57861197e-02 -4.64827918e-02
 -1.32100595e-01  4.60426186e-01  2.86433135e-01  2.88852570e-01
 -1.15770341e-01  2.66853781e-01 -1.42905801e-01 -1.76408967e-02
 -1.26757226e-02 -4.02139168e-04]
[ 2.50885971e-01 -8.25723507e-03 -1.70344076e-01  6.00699571e-03
  4.33320687e-02 -3.08734498e-02 -1.67966619e-01 -3.61707954e-02
  6.04884615e-02 -7.66482910e-02 -1.17772055e-01 -6.81253543e-02
 -2.54387490e-01 -2.76418891e-01  1.69499607e-01  1.78141741e-01
 -7.54853164e-02  7.58138963e-02  2.60229625e-01  3.30223397e-02
  8.86081478e-04  2.99840557e-01 -5.67527797e-01 -1.07340243e-01
 -2.63196337e-01 -1.33574507e-01  2.30901389e-01  2.24756680e-02
  3.52404543e-02  2.28844179e-03]
[ 1.22904556e-01  1.41883349e-01 -2.71312642e-01 -3.62506947e-02
 -2.44558663e-01  4.98926784e-01 -1.84906298e-02  2.28225053e-01
  6.46378061e-02 -2.95630751e-02  1.57560248e-01  4.40335026e-02
 -2.56534905e-01  5.35557351e-03 -1.39888394e-01  2.57894009e-01
  4.30658116e-01 -2.78713843e-01 -1.17250532e-01 -1.16759236e-01
  1.62708549e-01  9.71448437e-02  1.21343451e-01  1.43818093e-02
 -4.52996243e-02  2.81842956e-02  2.27904438e-02  4.92048082e-03
  1.34042283e-02 -3.95443454e-04]
[ 1.31783943e-01  2.75339469e-01 -2.32791313e-01 -7.70534703e-02
  9.44233510e-02 -8.02235245e-02  3.74657626e-01  4.83606666e-02
 -1.34174175e-01  1.26095791e-02  1.18283551e-01 -3.47316933e-02
 -1.72814238e-01 -2.12104110e-01  2.56173195e-01 -4.05556492e-01
  1.59394300e-01  2.35647497e-02  1.14944811e-02 -1.04991974e-01
 -9.23439434e-02 -4.69471147e-01  7.62533821e-03 -3.78254532e-02
 -2.80133485e-01  4.52048188e-03  5.99859979e-02 -2.35621424e-02
  1.14776603e-02 -1.89429245e-03]]
```

結論:

由上結果可以看出，透過觀察組成的係數，可以清楚理解原變數在建構Z_1 Z_2新變數時的重要性。係數的絕對值越大，則表示原變數的貢獻越大；相反的，如果組成係數趨近於零，則表示原始變數幾乎沒貢獻，在數據解釋中較不重要。而相關係數可以更直觀的理解變數之間的關係；因此，綜合考慮兩點才能更更全面的理解數據。