

第一次作品：主成分分析應用於群組資料之探討

學號：411078064

姓名：謝意盛

作品目標：

參考資料：

1. 汪群超 Chun-Chao Wang 老師的講義網址

載入套件：

```
In [122]:  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from matplotlib.ticker import FuncFormatter  
import matplotlib.lines as mlines  
from matplotlib.colors import ListedColormap  
import seaborn as sns  
from sklearn.preprocessing import StandardScaler  
from sklearn.decomposition import PCA
```

第 1 題：

有一組資料來自義大利某個地區的三家紅酒製造商 (Wine.xlsx)，資料內容記錄了 178 支紅酒的 13 種化學成分。本題利用這組資料回答下列 8 項問題：

(1) 匯入資料，觀察資料的基本屬性。

```
In [2]:  
# 路徑  
path = r"C:\Users\Sheng\Documents\NTPU_Bachelor\Year_4\Y4_Sem_2\ShallowML\ClassData"  
# 讀取資料  
df_wine = pd.read_excel(path + r'\Wine.xlsx')  
df_wine.head()  
# 取出反應變數 y  
y = df_wine.iloc[:, -1].values  
# 移除最後一欄  
df_wine1 = df_wine.iloc[:, :-1]  
  
# 顯示資料特性  
# df.info()  
print('【樣本數】:', df_wine1.shape[0])  
print('【欄位數】:', df_wine1.shape[1])  
display(df_wine1.head())
```

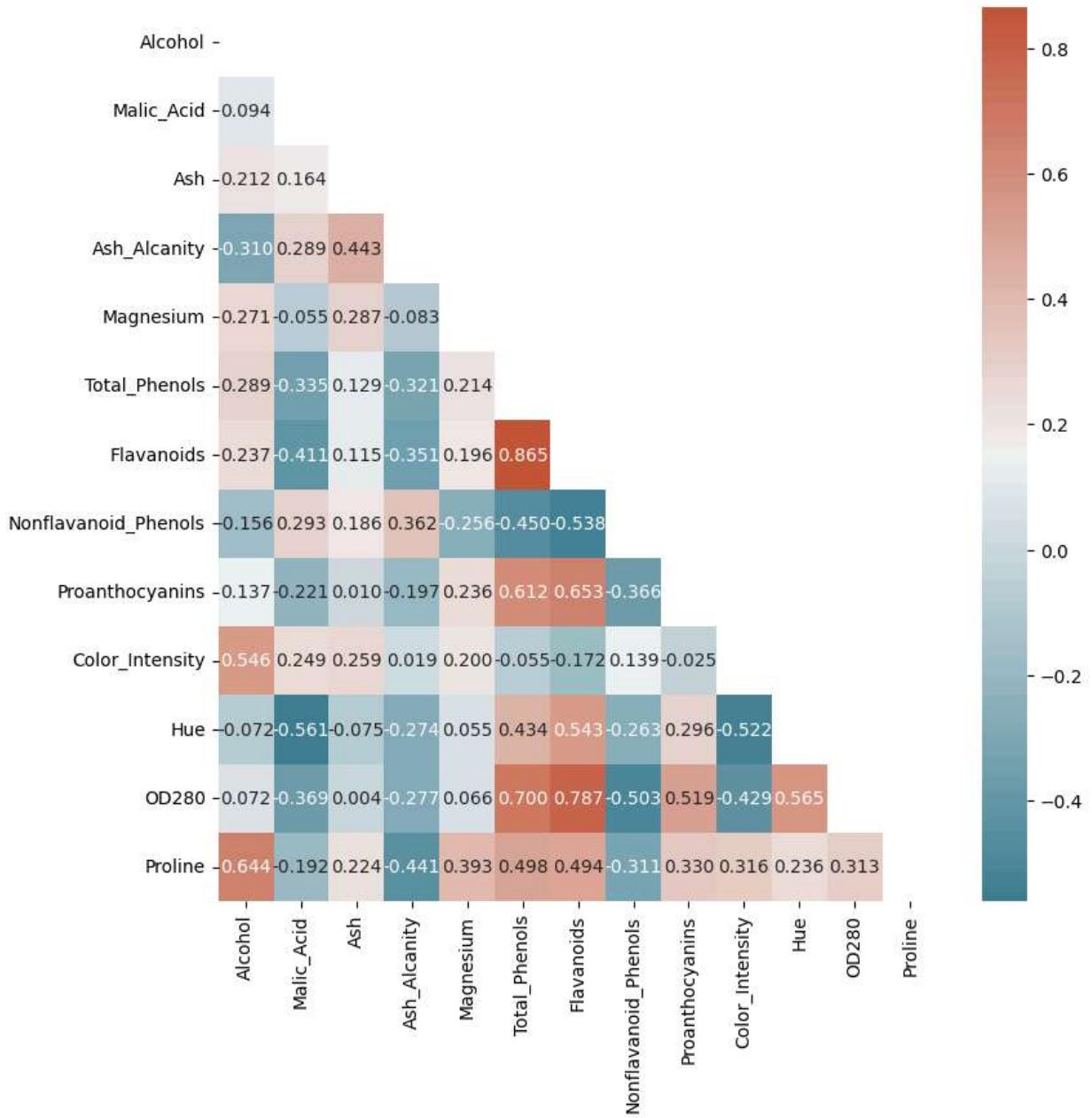
【樣本數】：178
【欄位數】：13

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phen
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0



(2) 利用 `seaborn` 套件裡的 `heatmap` 指令繪製出變數間的相關係數圖，以觀察變數間是否存在相關性。

```
In [3]: # Plot the correlation matrix
plt.figure(figsize = (9, 9))
# 計算相關係數
R = df_wine1.corr()
# np.ones_like(R, dtype = bool): 生成一個和 R 一樣大小的矩陣，元素全為 True
# np.triu(): 上三角矩陣(np.tril 則是下三角矩陣)
mask = np.triu(np.ones_like(R, dtype = bool)) # diagonal mask
# annot = True: 在 heatmap 上標註相關係數
# mask: 隱藏上三角(或下三角)
cmap = sns.diverging_palette(220, 20, as_cmap=True)
sns.heatmap(R, annot = True, mask = mask, cmap = cmap, fmt = ".3f")
plt.show()
```



注意事項與討論：

- 上圖呈現的是變數間的相關係數圖。由於相關係數矩陣為對稱矩陣 (Symmetric Matrix)，因此僅顯示其下三角矩陣 (Lower Triangular Matrix)。
- 從圖中可見，紅色越深代表變數間的正相關性越高，而藍色越深則代表負相關性越高。例如，**Flavanoids** 與 **Total_Phenols** 及 **OD280** 之間呈現高度正相關，相關係數皆大於 0.7；而與 **Nonflavanoid_Phenols** 及 **Malic_Acid** 則呈現負相關，相關係數皆小於 -0.4。
- 另一個例子是 **Color_Intensity** 與 **Alcohol** 之間存在正相關，相關係數約為 0.546，而與 **Hue** 則呈現負相關，相關係數為 -0.522。
- 從相關係數矩陣的觀察結果來看，部分變數之間確實存在較強的相關性，因此可考慮使用主成分分析 (PCA) 進行降維，以減少變數間的多重共線性並提升分析的效能。

(3) 繪製一張含有每種化學成分（變數）的盒鬚圖（Boxplot），觀察每個變數的數值範圍（Scaling），以作為是否需要標準化的參考。同時，再繪製一張標準化後的盒鬚圖，並與原始數據進行比較。

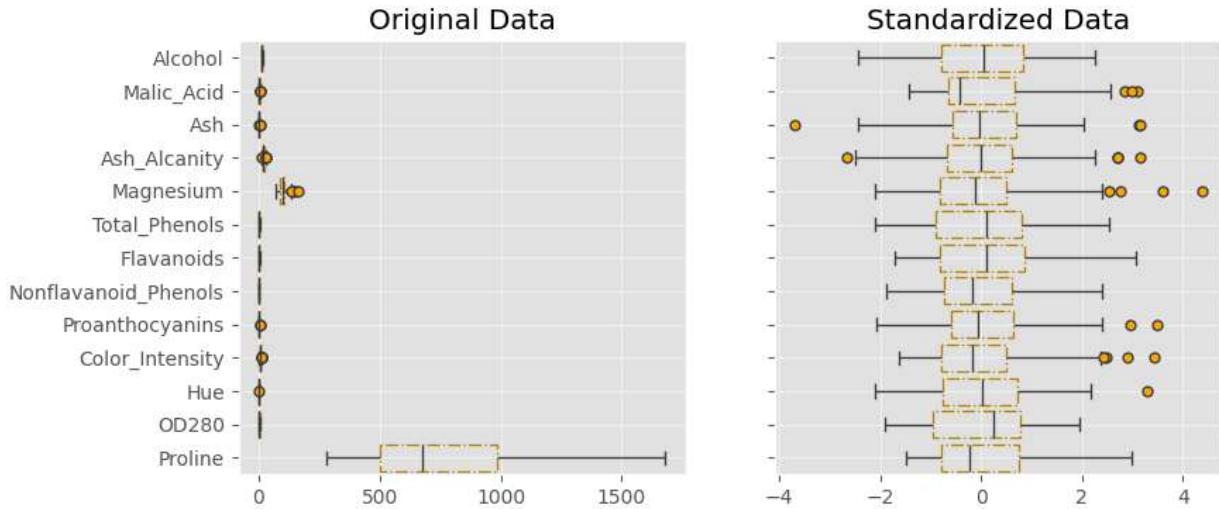
```
In [4]: # 將資料進行標準化
columns = df_wine1.columns
scaler = StandardScaler() # create a StandardScaler object
X_std = scaler.fit_transform(df_wine1) # standardize the data
df_X_std = pd.DataFrame(X_std, columns=columns)

In [60]: # 繪製 boxplot
plt.style.use('ggplot')
fig, ax = plt.subplots(1, 2, figsize=(9, 4), sharey=True)
boxprops = dict(linestyle='-.', linewidth=1, facecolor='none', edgecolor='darkgoldenrod')
flierprops = dict(marker='o', markerfacecolor='orange', markersize=5, linestyle='none')

# 未標準化
sns.boxplot(data=df_wine1, ax=ax[0], orient='h', boxprops=boxprops, flierprops=flierprops)
ax[0].grid(True, alpha=0.5)
ax[0].set_title('Original Data')

# 標準化
sns.boxplot(data=df_X_std, ax=ax[1], orient='h', boxprops=boxprops, flierprops=flierprops)
ax[1].grid(True, alpha=0.5)
ax[1].set_title('Standardized Data')

plt.show()
```



注意事項與討論：

- 左圖顯示的是原始資料中各化學成分的盒鬚圖，而右圖則是標準化後的盒鬚圖。
- 從左圖可以明顯看出，各化學成分的數值範圍並不一致，尤其是 **Proline**，其數值範圍遠高於其他變數，導致整體數據的 scaling 被拉長。因此，為了避免 **Proline** 在後續分析中對結果產生過大影響，應對資料進行標準化，以降低潛在誤差。
- 標準化後的右圖顯示，各化學成分的數值範圍趨於一致，使後續分析更加穩定，避免因數據 scaling 差異而產生誤差。

(4) 對原始資料與標準化後的資料分別進行主成分分析 (PCA)，並繪製 Scree Plot (顯示特徵值由大到小的分布) 以及 Pareto Plot (顯示每個主成分解釋的變異量比例及其累積變異量比例)，觀察標準化是否會影響主成分的分析結果。

In [6]:

```
# PCA
# 未標準化的資料
pca = PCA() # create a PCA object
X_pca = pca.fit(df_wine1) # apply PCA to the standardized data
Z = pca.transform(df_wine1) # get the new data matrix Z
eigenvalues = pca.explained_variance_
eigenvectors = pca.components_

# 標準化後的資料
pca_std = PCA() # create a PCA object
X_pca_std = pca_std.fit(X_std) # apply PCA to the standardized data
Z_std = pca_std.transform(X_std) # get the new data matrix Z
eigenvalues_std = pca_std.explained_variance_
eigenvectors_std = pca_std.components_
```

In [66]:

```
# 繪製 scree plot 和 pareto plot
plt.style.use('ggplot')

def plots(eigenvalues, title):
    # 計算主成分數量
    pca_range = np.arange(1, len(eigenvalues) + 1)
    # 計算解釋變異量
    cum_var_exp = np.cumsum(eigenvalues / eigenvalues.sum()) * 100

    fig, ax = plt.subplots(1, 2, figsize=(10, 4))
    # Scree Plot
    ax[0].plot(pca_range, eigenvalues, 'o-', alpha=0.9, color='gray')
    ax[0].set_title('Scree Plot')
    ax[0].set_xlabel('Principal Component ($z_i$)')
    ax[0].set_ylabel('Eigenvalue')

    # Pareto Plot
    ax2 = ax[1].twinx() # 創建共享 x 軸的第二個 y 軸
    ax[1].bar(pca_range, eigenvalues, alpha=0.6, align='center', label='Individual exp')
    ax[1].set_title('Pareto Plot')
    ax[1].set_xlabel('Principal Component ($z_i$)')
    ax[1].set_ylabel('Variance Explained')
    if title == 'PCA: Original Data':
        # 使用 FuncFormatter 來格式化 y 軸標籤，例如1e2
        ax[1].yaxis.set_major_formatter(FuncFormatter(lambda x, _: '{:.0e}'.format(x)))
    # 畫出累積解釋變異量，用百分比表示
    ax2.plot(pca_range, cum_var_exp, 'o-', alpha=0.9, color='gray')
    ax2.set_ylabel('Cumulative Variance Explained')

    if title == 'PCA: Original Data':
        # 使用 FuncFormatter 來格式化 y 軸標籤
        ax2.yaxis.set_major_formatter(FuncFormatter(lambda y, _: '{:.1f}%'.format(y)))
    else:
        ax2.yaxis.set_major_formatter(FuncFormatter(lambda y, _: '{:.0f}%'.format(y)))

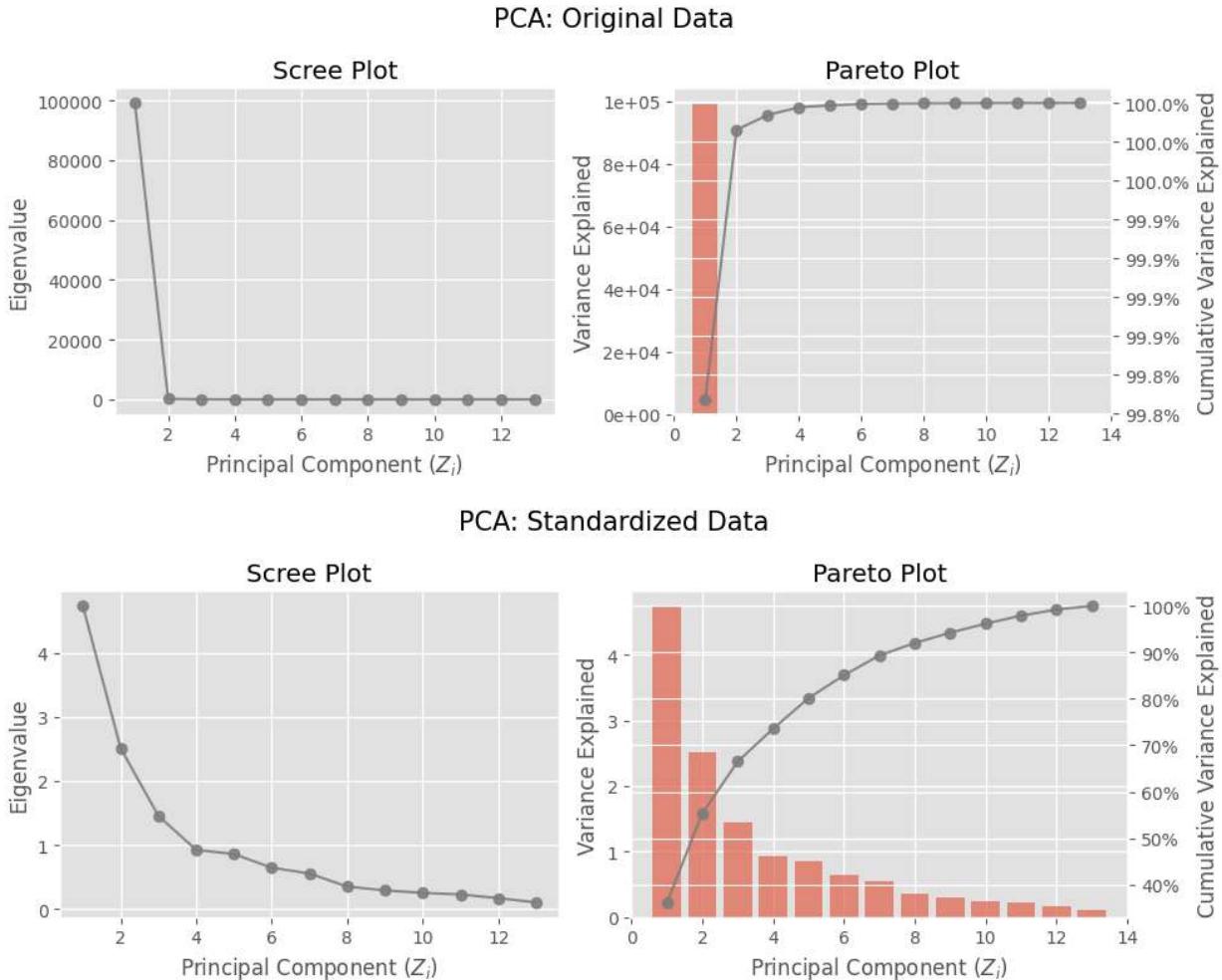
    plt.suptitle(title, fontsize=15)
    plt.tight_layout()
    plt.show()

# 未標準化
```

```

plots(eigenvalues, 'PCA: Original Data')
# 標準化
plots(eigenvalues_std, 'PCA: Standardized Data')

```



注意事項與討論：

- 上圖為原始資料進行 PCA 後的結果，其中 **Scree Plot** (左上) 顯示特徵值的分布，**Pareto Plot** (右上) 則顯示各主成分的變異量比例與累積變異量比例；下圖則為標準化後資料進行 PCA 的對應結果，分別為 **Scree Plot** (左下) 與 **Pareto Plot** (右下)。
- 從上圖可以看出，**Scree Plot** 中第一與第二主成分的特徵值差距顯著，而 **Pareto Plot** 顯示第一主成分解釋的變異量比例遠高於其他主成分，導致變異分配極不均衡，說明未標準化的資料會影響 PCA 的結果，使得部分主成分的影響被放大或縮小。
- 下圖則顯示標準化後的 PCA 分析結果較為合理，表明標準化後的 PCA 具有較佳的解釋性。在 **Scree Plot** 中，前三個主成分的特徵值變化較為陡峭，而從第四主成分開始趨於平緩；**Pareto Plot** 則顯示前三個主成分的累積變異量比例已接近 70%，表示這三個主成分已能解釋大部分資料的變異。因此，可判斷該資料取前三個主成分即可有效代表原始資料的結構，同時大幅降低維度，減少資訊流失並提升後續分析的效率。

(5) 資料中的每支酒都有標籤 (df_wine 中的 **Customer Segment**)，表示其來自的酒莊。現在假設不考慮此標籤，先利用主成分分析 (PCA) 擷取前兩個主成分 (Z_1 和 Z_2)，並繪製其散布

圖，觀察該散布圖是否能夠顯示出三個明顯的群組。請注意：資料是否經過標準化可能會影響結果，試著比較標準化與非標準化情況下的差異。

In [194...]

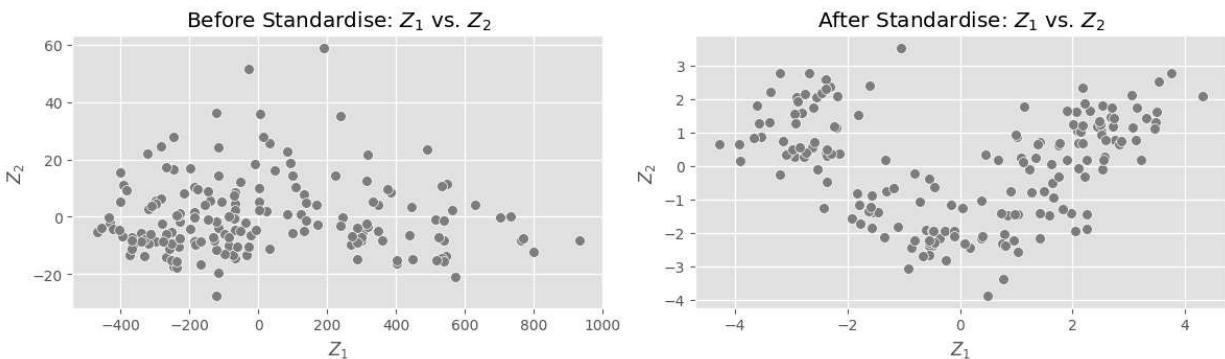
```
plt.style.use('ggplot')
fig, ax = plt.subplots(1, 2, figsize=(12, 4))

ax[0].scatter(Z[:, 0], Z[:, 1], edgecolors='w', color='gray', s=50)
ax[0].set_xlabel('$Z_1$')
ax[0].set_ylabel('$Z_2$')
ax[0].set_title('Before Standardise: $Z_1$ vs. $Z_2$')

ax[1].scatter(Z_std[:, 0], Z_std[:, 1], edgecolors='w', color='gray', s=50)
ax[1].set_xlabel('$Z_1$')
ax[1].set_ylabel('$Z_2$')
ax[1].set_title('After Standardise: $Z_1$ vs. $Z_2$')

plt.suptitle('Distribution of Data After PCA', fontsize=15)
plt.tight_layout()
plt.show()
```

Distribution of Data After PCA



注意事項與討論：

- 左圖為未標準化資料的前兩個主成分所繪製的散佈圖。可以看出，數據點之間的分佈並無明顯區隔，無法直接劃分出群組，顯示未標準化的數據可能會影響 PCA 的判斷。
- 右圖則為標準化後的前兩個主成分散佈圖。與左圖相比，數據點呈現出較明顯的分佈模式，整體趨勢呈 V 字形，可清楚觀察並劃分出三個群組，說明標準化確實有助於 PCA 結果的判讀。

(6) 根據每筆資料的標籤（已在第 1 小題中存入 `y` 變數），為散佈圖上的資料點填上對應的顏色，並觀察其分佈是否與前一小題的判斷相符。

In [197...]

```
plt.style.use('ggplot')
# 獲取唯一的類別
unique_classes = np.unique(y)

# 創建顏色映射
base_cmap = plt.colormaps['Accent']
cmap = ListedColormap(base_cmap(np.linspace(0, 1, len(unique_classes)))))

fig, ax = plt.subplots(1, 2, figsize=(12, 4))
```

```

# Before Standardise
scatters = ax[0].scatter(Z[:, 0], Z[:, 1], c=y, edgecolors='w', cmap=cmap, s=50)
ax[0].set_xlabel('$Z_1$')
ax[0].set_ylabel('$Z_2$')
ax[0].set_title('Before Standardise: $Z_1$ vs. $Z_2$')

# 查出 scatters 使用的顏色和對應的類別
colors = cmap(np.linspace(0, 1, len(unique_classes)))
labels = [f'Class {cls}' for cls in unique_classes]

# 創建圖例項目
legend_patches = [mlines.Line2D([], [], color=colors[i], marker='o', linestyle='None',
                                label=labels[i]) for i in range(len(unique_classes))]

# 添加圖例
legend = ax[0].legend(handles=legend_patches, loc='upper right', fontsize=11)
legend.get_frame().set_alpha(0.5) # 設置圖例框的透明度

# After Standardise
scatters_std = ax[1].scatter(Z_std[:, 0], Z_std[:, 1], c=y, edgecolors='w', cmap=cmap,
                             ax[1].set_xlabel('$Z_1$')
                             ax[1].set_ylabel('$Z_2$')
                             ax[1].set_title('After Standardise: $Z_1$ vs. $Z_2$')

# 查出 scatters_std 使用的顏色和對應的類別
colors_std = cmap(np.linspace(0, 1, len(unique_classes)))
labels_std = [f'Class {cls}' for cls in unique_classes]

# 創建圖例項目
legend_patches_std = [mlines.Line2D([], [], color=colors_std[i], marker='o', linestyle='None',
                                     label=labels_std[i]) for i in range(len(unique_classes))]

# 添加圖例
legend = ax[1].legend(handles=legend_patches_std, loc='lower right', fontsize=11)
legend.get_frame().set_alpha(0.5) # 設置圖例框的透明度

plt.suptitle('Distribution of Data After PCA', fontsize=15)
plt.tight_layout()
plt.show()

```



注意事項與討論：

- 左圖為未標準化資料的前兩個主成分散佈圖。加入標籤後可以更直觀地觀察到，資料點之間並未形成明確的分群，即使進行了PCA，仍無法有效區分群組，這再次說明標準化對PCA的重要性。

- 右圖為標準化資料的前兩個主成分散佈圖。與左圖相比，加入標籤後可以明顯看出資料點已被清楚劃分為三個群組，進一步證明標準化有助於 PCA 在群組辨識上的效果。
- 綜合第 3、4、5、6 小題的結果，可以明顯看出標準化與否對 PCA 影響重大。因此，在進行 PCA 之前，應先檢視資料的數值範圍（scaling），判斷是否需要標準化，以避免後續分析產生誤差。

(7) 當採用三個主成分時，可繪製三維散佈圖，以觀察群組的區分效果是否更佳。請嘗試旋轉視角，以獲得最佳辨識效果，並檢視在三維空間中，資料點是否能夠更清楚地分群。

In [190...]

```
# 獲取唯一的類別
unique_classes = np.unique(y)

# 創建顏色映射
base_cmap = plt.colormaps['Accent']
cmap = ListedColormap(base_cmap(np.linspace(0, 1, len(unique_classes)))))

# 改畫立體圖，要利用三個主成分
plt.style.use('default')
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
sc = ax.scatter(Z_std[:, 0], Z_std[:, 1], Z_std[:, 2], c=y, edgecolors='w', cmap=cmap,
                 ax.set_xlabel('$Z_1$')
                 ax.set_ylabel('$Z_2$')
                 ax.set_zlabel('$Z_3$')
                 ax.set_title('3D Plot: Distribution of Data After PCA', fontsize=15)

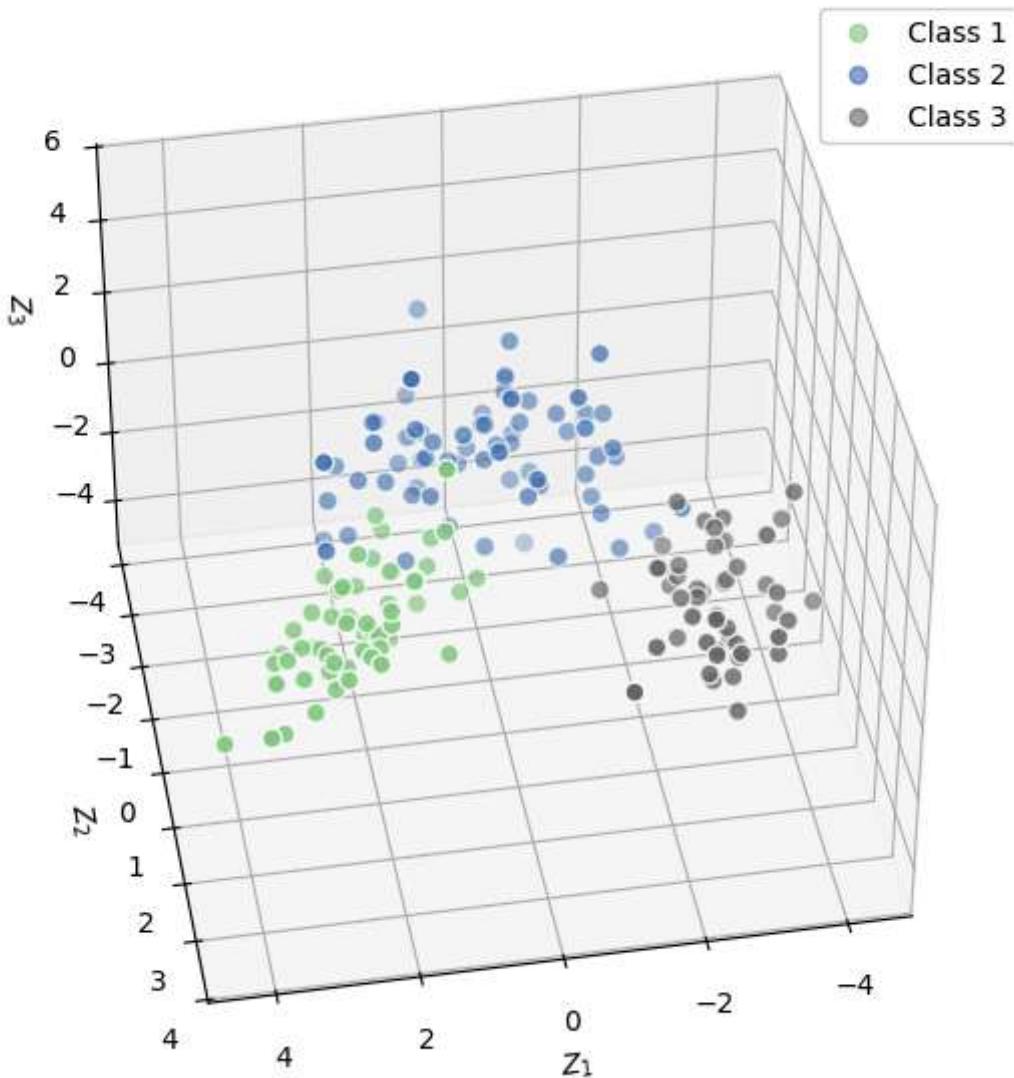
# 調整觀察角度
ax.view_init(elev=40, azim=80) # 例如，仰角 40 度，方位角 80 度

# 創建圖例項目
legend_elements, _ = sc.legend_elements(prop="colors", alpha=0.6)
labels = [f'Class {cls}' for cls in unique_classes]

# 添加圖例
legend = ax.legend(handles=legend_elements, labels=labels, loc='upper right', fontsize=10)
legend.get_frame().set_alpha(0.9) # 設置圖例框的透明度

plt.tight_layout()
plt.show()
```

3D Plot: Distribution of Data After PCA



注意事項與討論：

- 上圖為標準化後資料的前三個主成分所繪製的三維散佈圖（3D Scatter Plot）。從圖中可以清楚看出，資料點在三維空間中已能有效區分，說明僅使用前三個主成分即可完整呈現資料的分佈情況，進一步驗證了第4小題的判斷。
- 這證明在更高維度的空間中，可以更清晰地觀察資料點的分佈，從而減少資訊遺失，提升對資料結構的辨識能力。

(8) 觀察並判斷原變數 X_i 與主成分 Z_i 之間的關係：

主成分 Z_1 和 Z_2 是由原始變數 X_i 的線性組合所形成的新變數。我們可以透過前兩個特徵向量 (eigenvectors) 的係數，來分析各原變數對於這兩個主成分的貢獻程度，判斷哪些變數影響較大，哪些影響較小。此外，可以繪製特徵向量與原變數對應的熱圖 (heatmap)，以更直觀地觀察不同變數在各主成分中的權重分佈。

進一步地，若將此結果與第 2 小題的相關係數圖對照，可比較兩者是否提供相似的資訊。例如，若某些變數在相關係數圖中彼此高度相關，它們可能在 PCA 中對相同的主成分具有較高的權重。此題先不考慮理論推導，而僅從圖像觀察與直覺分析，看看是否能發現有趣的模式或趨勢。

原始資料 X_{std} : $\{X_{std}\} =$

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}$$

$\quad \quad \quad m = 178, n = 13$

$$Z_1 = v_{11}x_{11} + v_{12}x_{12} + \cdots + v_{1n}x_{1n}$$

\vdots

$$Z_{13} = v_{m1}x_{m1} + v_{12}x_{12} + \cdots + v_{113}x_{113}$$

eigenvector =

1) 將 eigenvectors 做資料前處理。

```
In [230]: eigenvectors_df = pd.DataFrame(eigenvectors_std[:2], columns = df_wine1.columns, index=display(eigenvectors_df))
```

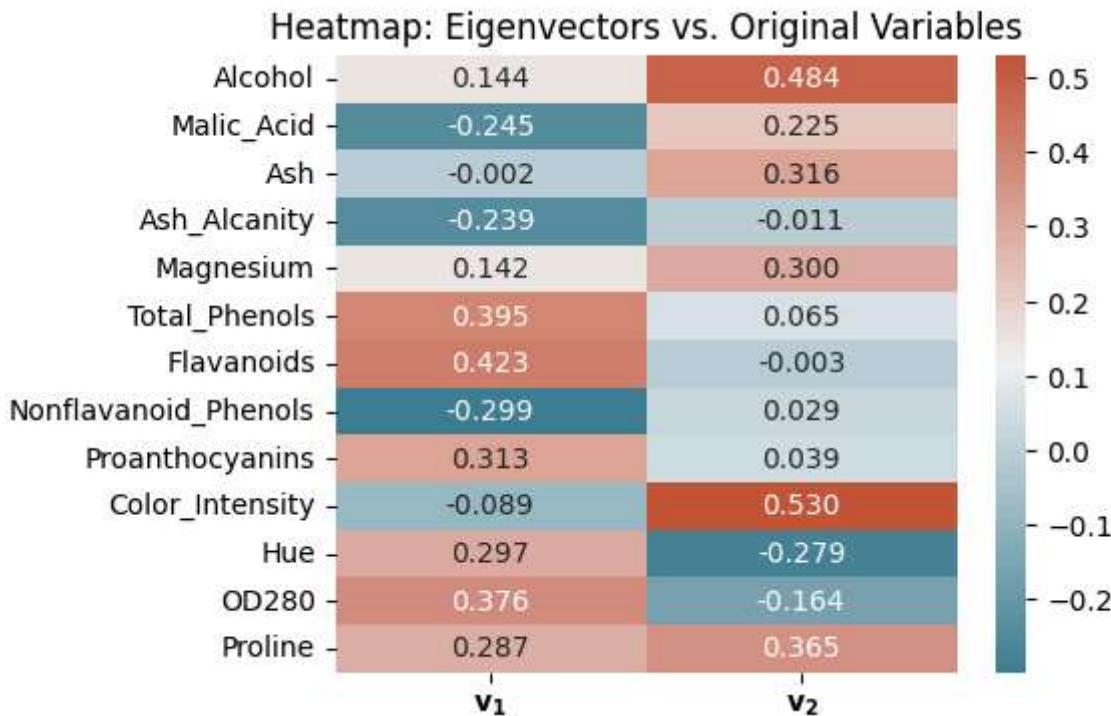
	v ₁	v ₂
Alcohol	0.144329	0.483652
Malic_Acid	-0.245188	0.224931
Ash	-0.002051	0.316069
Ash_Alcanity	-0.239320	-0.010591
Magnesium	0.141992	0.299634
Total_Phenols	0.394661	0.065040
Flavanoids	0.422934	-0.003360
Nonflavanoid_Phenols	-0.298533	0.028779
Proanthocyanins	0.313429	0.039302
Color_Intensity	-0.088617	0.529996
Hue	0.296715	-0.279235
OD280	0.376167	-0.164496
Proline	0.286752	0.364903

2) 繪製特徵向量與原變數對應的熱圖 (heatmap)，以及第 2 小題原變數之間的相關係數熱圖。

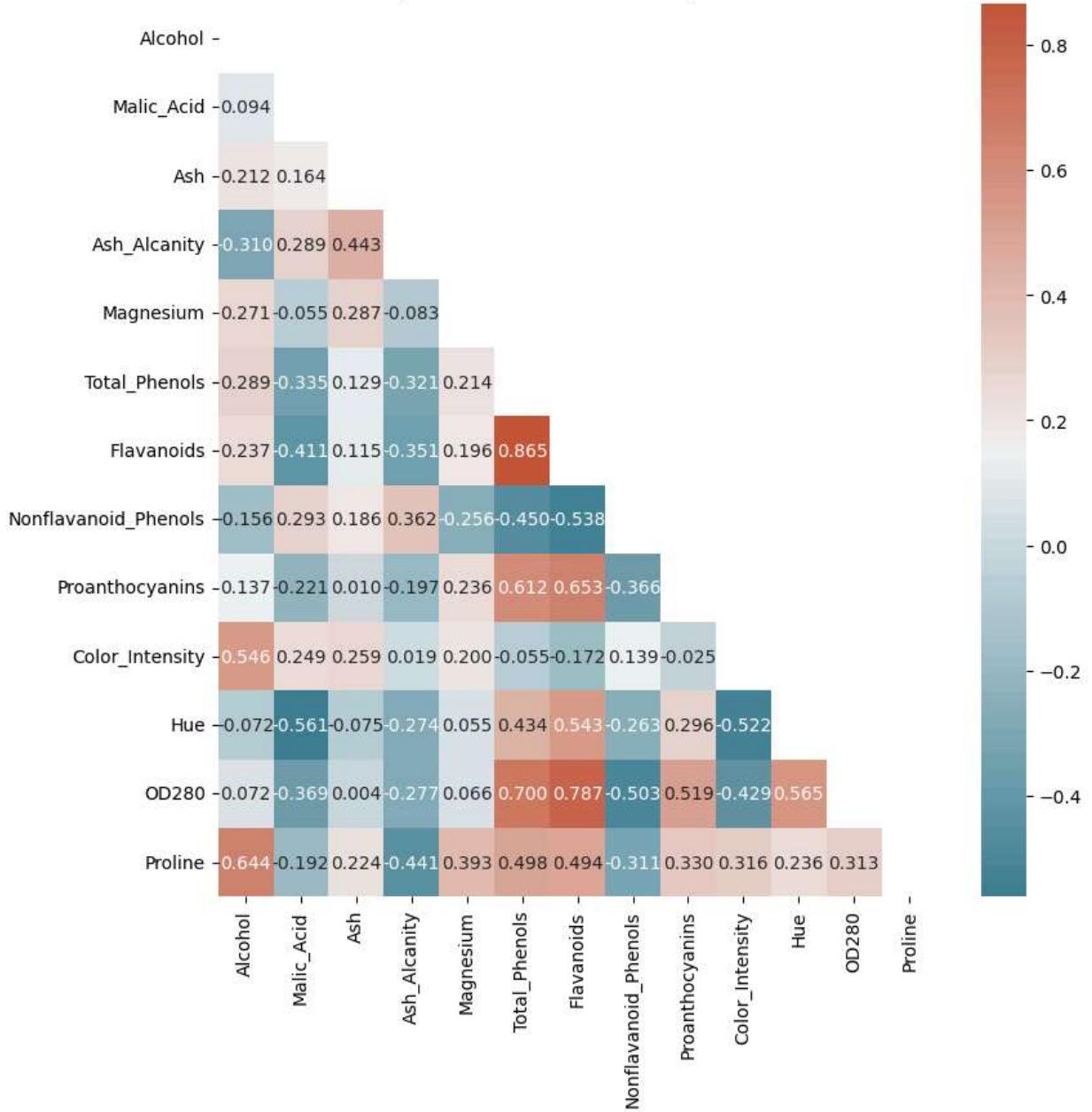
In [231...]

```
# 繪製特徵向量與原始變數的熱圖
plt.figure(figsize = (5, 4))
cmap = sns.diverging_palette(220, 20, as_cmap=True)
sns.heatmap(eigenvectors_df, annot = True, cmap = cmap, fmt = ".3f")
plt.title('Heatmap: Eigenvectors vs. Original Variables')
plt.show()

# 原變數間的相關係數熱圖
plt.style.use('default')
plt.figure(figsize = (9, 9))
# 計算相關係數
R = df_wine1.corr()
# np.ones_like(R, dtype = bool): 生成一個和 R 一樣大小的矩陣 · 元素全為 True
# np.triu(): 上三角矩陣 (np.tril 則是下三角矩陣)
mask = np.triu(np.ones_like(R, dtype = bool)) # diagonal mask
# annot = True: 在 heatmap 上標註相關係數
# mask: 隱藏上三角 (或下三角)
cmap = sns.diverging_palette(220, 20, as_cmap=True)
sns.heatmap(R, annot = True, mask = mask, cmap = cmap, fmt = ".3f")
plt.title('Heatmap: Correlation Matrix of Original Variables')
plt.show()
```



Heatmap: Correlation Matrix of Original Variables



注意事項與討論：

- 上圖為