I. About this homework:

1. 本次 Programming 作業共有四個問題。此報告將針對前三題的 coding 以及其相關概念做說明，最後第四題，會單以文字回答，以下為本次繳交之作業清單及 HW1.py 中所使用的 dependencies。
2. Submitted files:

```
HW1_110061543
├── HW1.py
├── README.txt
├── test.csv
├── train.csv
└── Wine.csv
```

3. Installed dependencies
   - pandas
   - numpy
   - matplotlib
   - sklearn

II. Question 1: Splitting wine.csv into training data and test data.

1. Wine.csv 中的資料已經根據 Type0, 1, 2 排序好了，所以直接根據 rows 分成 3 個 dataframes (type1, type2, type3)，並從三個 dataframes 中各隨機取 20 筆資料，將這 60 筆資料存成 test.csv，剩下的 423 資料存成 train.csv。

```
7   ###################### part 1 ######################
8   # Read the CSV file into a DataFrame
9   df = pd.read_csv('Wine.csv')
10
11  # Split the DataFrame into 3 types ,and sample 20 from each of them
12  type1 = df.iloc[2:176]
13  type2 = df.iloc[177:381]
14  type3 = df.iloc[382:]
15  type1_sample = type1.sample(n=20)
16  type2_sample = type2.sample(n=20)
17  type3_sample = type3.sample(n=20)
18
19  # Write test.csv & train.csv files respectively
20  sampled_df = pd.concat([type1_sample, type2_sample, type3_sample])
21  sampled_df.to_csv('test.csv', index=False)
22  df = df.drop(sampled_df.index)
23  df.to_csv('train.csv', index=False)
```

III. Question 2: Evaluating the posterior probabilities and accuracy rate

1. 第二部分要訓練 classifier 然後用 test data 來算準確率。首先用 pandas 將 train.csv 跟 test.csv 轉成 numpy arrays，x_train 用來表示 feature values，而 y_train 用來表示 labels (x_test, y_test 亦同)。

```
25  ###################### part 2 ######################
26  # Load the training and test data and convert to np arrays for training
27  train_df = pd.read_csv('train.csv')
28  test_df = pd.read_csv('test.csv')
29  x_train, y_train = train_df.iloc[:, 1:].values, train_df.iloc[:, 0].values # values of features, target
30  x_test, y_test = test_df.iloc[:, 1:].values, test_df.iloc[:, 0].values
31  #print(y_train)
```

2. 第二部份主要為兩個 functions，”fit”用來計算 training data 中每個 label 的 priors, means, standard deviations；接下來用”fit”計算好的資料，用”predict”來預測 test data 的 predicted labels，最後跟 test data 的 labels 來比較準確率。

   (1) 在 function “fit”中，先根據每個 labels 計算 priors 後，再計算 means, standard deviations，因為有多個 labels 資料，因此 priors 跟 means, standard deviations 分別存入兩個 dict，以便 function “predict”使用。

```
33  # Get labels and calculate the priors, means ,and stds
34  def fit(x, y): # "x" for x_train, "y" for y_train
35      # Priors
36      labels = np.unique(y)
37      n_samples = len(y)
38      priors = {}
39      for label in labels:
40          priors[label] = np.sum(y == label) / n_samples
41
42      # Mean & stds
43      parameters = {}
44      for label in labels:
45          label_data = x[y == label]
46          means = np.mean(label_data, axis=0)
47          stds = np.std(label_data, axis=0)
48          parameters[label] = list(zip(means, stds))
49
50      return labels, priors, parameters
```

   (2) 根據下圖 Maximum A Posteriori probability (MAP)的公式[2]，Function “predict” 除了使用 priors, means, standard deviations 外，還需要計算 likelihood，因前面”fit” 以計算好大部分所需數值，因此另外寫一個 function “likelihood”，用於使用”predict”計算 MAP。

$$p(w_i|x) = \frac{p(w_i)p(x|w_i)}{p(x)}$$

(Note: $p(w_i|x) = \frac{p(w_i,x)}{p(x)} = \frac{p(w_i)\frac{1}{p(w_i)}p(w_i,x)}{p(x)} = \frac{p(w_i)p(x|w_i)}{p(x)}$)

$p(w_i)$: 事前機率 (priori probability)

$p(x|w_i)$: 概似函數 (likelihood function)或是類別條件密度函數(class-conditional density function)

$p(x) = \sum_i^L p(w_i)p(x|w_i)$: 全機率

(Maximum A Posteriori probability)

```python
# Make predictions on the test data
def predict(x, labels, priors, parameters):
    y_pred = []
    for data in x:
        posteriors = []
        for label in labels:
            likelihoods = []
            for i, (mean, sigma) in enumerate(parameters[label]):
                likelihoods.append(likelihood(data[i], mean, sigma))
            likelihood_product = np.product(likelihoods)
            posterior = priors[label] * likelihood_product
            posteriors.append(posterior)
        posterior_sum = np.sum(posteriors)
        posteriors /= posterior_sum
        label_idx = np.argmax(posteriors)
        y_pred.append(labels[label_idx])
    return np.array(y_pred)

def likelihood(x, mean, sigma):
    return np.exp(-((x - mean) ** 2 / (2 * sigma ** 2))) / (sigma * np.sqrt(2 * np.pi))
```

(3) 最後計算 accuracy，根據 traing data 訓練出來的 classfier，拿來預測 test data 的 label，並跟 test data 做比較來計算準確率(截圖放在下章節跟 PCA 圖比較)。

```python
labels, priors, parameters = fit(x_train, y_train)
y_pred = predict(x_test, labels, priors, parameters)

#  Calculate the accuracy
correct_predictions = 0
for i in range(len(y_test)):
    if y_pred[i] == y_test[i]:
        correct_predictions += 1
accuracy = correct_predictions / len(y_test)
print(f'Accuracy rate: {accuracy:.2%}')
```

IV. Question 3: Plotting the visualized result of testing data using PCA

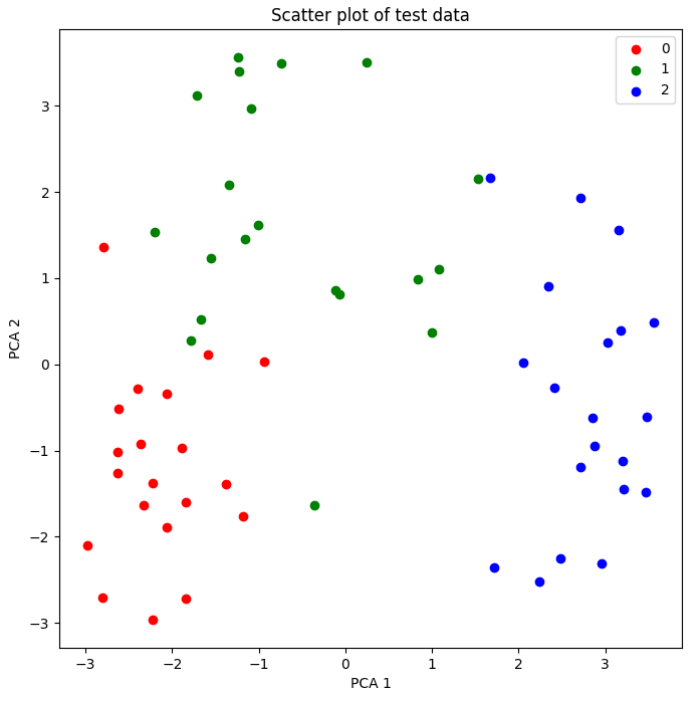
1. 這裡使用 Principal Component Analysis (PCA)的主要原因是要將多維度的資訊降成低維度，以便觀察 plots。使用 PCA，首先要將資料標準化，接著使用 PCA 降成所需維度(此題降成二階)，最後畫出 PCA 圖。

```python
###################### part 3 ######################
# Load the data and preprocess it with standardization ,then use PCA
def plot_scatter(x, y, title):
    labels = np.unique(y)
    colors = ['r', 'g', 'b']
    # Standardize the features
    x_std = StandardScaler().fit_transform(x)

    # Perform PCA to reduce the dimensionality
    pca = PCA(n_components=2)
    x_pca = pca.fit_transform(x_std)

    # Plot the scatter plot
    fig = plt.figure(figsize=(8, 8))
    ax = fig.add_subplot(1, 1, 1)
    for label, color in zip(labels, colors):
        ax.scatter(x_pca[y==label, 0], x_pca[y==label, 1], color=color, label=label)
    ax.set_xlabel('PCA 1')
    ax.set_ylabel('PCA 2')

    ax.set_title(title)
    ax.legend()
    plt.show()

plot_scatter(x_test, y_pred, 'Scatter plot of test data')
```

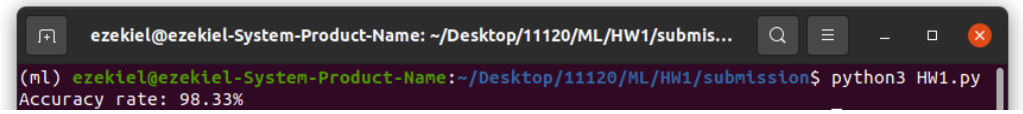
2. 這裡呈現上題需要之準確率與其 plot，因為準確率要求 95%以上，因此擷取準確率=96.67%, 98.33%, 100%做比較，可以發現準確率越高資料分類越清楚。

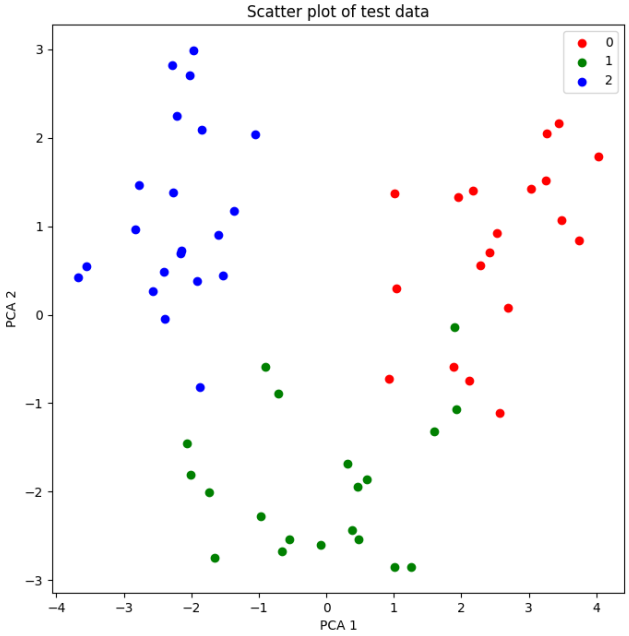
   (1) 96.67%

```
ezekiel@ezekiel-System-Product-Name: ~/Desktop/11120/ML/HW1/submis...
(ml) ezekiel@ezekiel-System-Product-Name:~/Desktop/11120/ML/HW1/submission$ python3 HW1.py
Accuracy rate: 96.67%
```

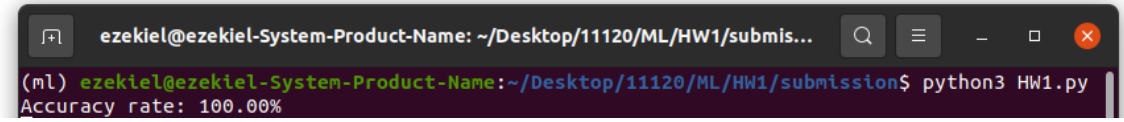


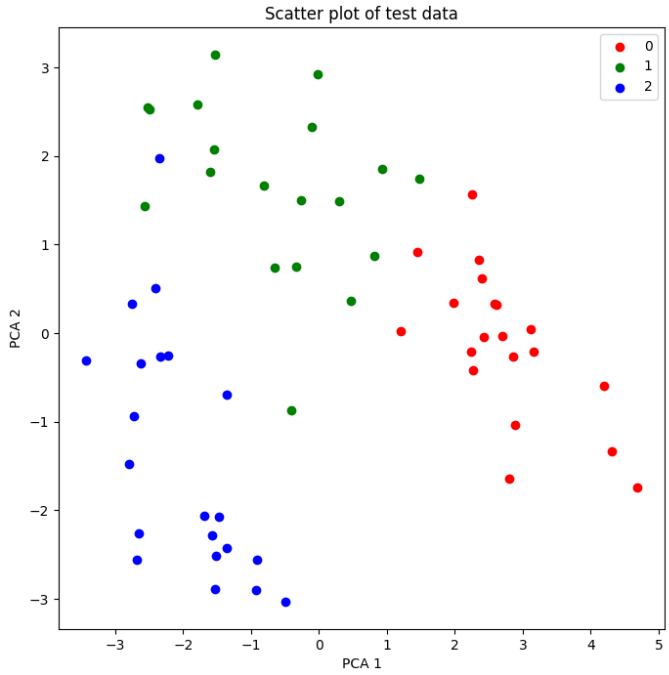
(2) 98.33%


ezekiel@ezekiel-System-Product-Name: ~/Desktop/11120/ML/HW1/submis...
(ml) ezekiel@ezekiel-System-Product-Name:~/Desktop/11120/ML/HW1/submission$ python3 HW1.py
Accuracy rate: 98.33%


Scatter plot of test data
0
1
2
PCA 2
PCA 1

(3) 100%


ezekiel@ezekiel-System-Product-Name: ~/Desktop/11120/ML/HW1/submis...
(ml) ezekiel@ezekiel-System-Product-Name:~/Desktop/11120/ML/HW1/submission$ python3 HW1.py
Accuracy rate: 100.00%


Scatter plot of test data
0
1
2
PCA 2
PCA 1

V. Question 4: The effect of prior distribution on the posterior probabilities

我認為這題應該是要講述 Class imbalance problem，本題作業每次都取三個 type 各 20 筆資料來放入 test.csv，因此每次 prior 的計算對於後續 posterior 不會有太大的影響，但實際上 prior 的分布應會影響 classifier，如果有一個 type 的資料特別多，會造成此類別的 prior 分布比較高，因此在訓練時，很有可能造成 classifier 會偏好此類別，使準確率降低。

VI. Conclusion

1. 本次作業主要練習 Maximum A Posteriori probability (MAP)，使用的葡萄酒資料分為 3 types 跟 16 features。除了撰寫 MAP 的所有流程，我們最後用 PCA 降低了資料維度，來輸出比較好觀察的圖形，而 PCA 的資料降維的功能，應在其他地方大有用處；最後一部分討論了 Class imbalance problem，雖在這裡只做討論，但在實作上非常重要。
2. Learning list:
   - Maximum A Posteriori probability (MAP)
   - Principal Component Analysis (PCA)
   - Class imbalance problem

VII. References

[1] https://archive.ics.uci.edu/ml/datasets/Wine

[2] https://reurl.cc/NqxM3p