機器學習 Machine Learning

劉昆琳 111061528 HW1 03/27/2023

Part 2. Programming assignment :

1.

第一題要求將資料集 wine.csv 中三種葡萄酒的類別進行切割,從每種類別隨機挑選出 20 筆測試資料,故使用 random.sample()這個函示來實現切割訓練集及測試集,並儲存成 train.csv 與 test.csv,如附檔。

2.

貝氏分類器假設每個特徵都是相互獨立的,並且每個特徵對結果的影響是獨立的。它是一種監督學習方法,其基本思想是利用已知類別的訓練數據集,通過計算特徵的條件概率和各類別的先驗概率,來推斷未知樣本所屬的類別,在此題分類問題中,要找的就是給定特徵下事件發生機率最高的目標$P(y_i|X_i,D)$,其中$X_i =< x_1, x_2, \dots, x_n >$代表各個特徵,$y_i$代表著所有結果中的其中一種,而發生機率最高的$P(y_i|X_i,D)$裡的$y_i$,則稱之為$y^*$,也就是分類器最後判定的類別,故寫成數學式子則為:

$$\boldsymbol{y^* = \arg\max_{y_i \in y} P(y_i|x_1, x_2, \dots, x_n)}$$

根據條件機率,可得以下式子,且由於假設每個特徵都是相互獨立的,故$P(x_1, x_2, \dots, x_n)$在每個類別中都是一樣的,故可以消除:

$$\boldsymbol{y^* = \arg\max_{y_i \in y} \frac{P(x_1, x_2, \dots, x_n|y_i)P(y_i)}{P(x_1, x_2, \dots, x_n)} = \arg\max_{y_i \in y} P(y_i) \prod_j P(x_j|y_i)}$$

由上式可得知,僅需知道$P(y_i)$、$P(x_1, x_2, \dots, x_n|y_i)$,就可以完成貝氏分類器的建模,而根據題目可以得知,特徵皆是連續的變數,且為常態分布,故須以樣本資料的平均值跟標準差來計算機率,算式改寫如下:

$$\boldsymbol{y^* = \arg\max_{y_i \in y} P(y_i) \prod_j^n N(x_j|u_{jy_i}, \sigma_{jy_i}^2)}$$

根據以上對貝氏分類器的基本了解，設計一副程式 Bayes_calprob(x,y)，其中輸入的 x、y 代表訓練集的特徵與類別，而函式會回傳 class_priors 與 likelihood，其中 class_priors 計算每個類別出現的機率，對應前面公式所提到$P(y_i)$，由於每項特徵都是高斯分布，故須先計算每個類別下特徵的平均值跟標準差並儲存在 likelihood，才可透過高斯函數的公式回推機率。

有了 Bayes_calprob(x,y)計算出來的 class_priors 與 likelihood，就可以依照貝氏分類器進行建模，並針對測試集的特徵進行分類，但此時這邊要注意一點，就是當訓練集樣本數目夠多時，使用貝氏分類器是可行的辦法，能有效避免過擬和的問題，但從前面推導的公式可以發現當特徵數很多的時候，相乘之下所算出的機率值會非常小，導致最後出來的後驗機率趨近於 0 造成儲存問題，故透過取 log 的方式把原本很小的訊號拉高，則公式可被表達成:

$$y^* = \arg\max_{y_i \in y}\left(\boldsymbol{log}\big(P(y_i)\big) + \sum_{j}^{n}\left(-0.5 \times \boldsymbol{log}\left(2\pi\sigma_{jy_i}^2\right) - \frac{(x_j - u_{jy_i})^2}{2\sigma_{jy_i}^2}\right)\right)$$

以上功能皆寫成副程式 predict(X, class_priors, likelihood)來求得後驗概率並分類測試集資料，其中 X 為輸入的測試資料，class_priors 與 likelihood 皆是透過 Bayes_calprob(x,y)所計算出來的；經過上述的方法與流程，最後測試集的準確率大多數都有大於 95%，下圖為 Bayes_calprob(x,y)、predict(X, class_priors, likelihood)與測試集的準確率，如圖 1 及圖 2。

```python
def Bayes_calprob(x,y):
    # P(c|x) = P(x|c)P(c)/P(x)
    # P(c|x) : Posteriori probability
    # P(x|c) : Likelihood functions
    # P(c) : class prior probability
    # P(x) : predictor prior probability

    classes = list(set(y))
    features = x.shape[1]
    class_priors = [0 for i in range(len(classes))]
    likelihood = dict()

    for c in classes:
        # 計算每個類別出現的機率 => 對應至P(c)
        class_priors[c] = list(y).count(c) / len(y)

        # 因為每項特徵是Gaussian distribution，所以計算在每個類別下特徵的平均值跟標準差 => 對應至P(x|c)
        # => 才可以透過公式回推機率函數
        likelihood[c] = np.vstack([np.mean(x[y == c],axis = 0),np.std(x[y == c],axis = 0)])

    return class_priors, likelihood
```

圖 1、函式 Bayes_calprob(x,y)

```python
def predict(X, class_priors, likelihood):
    preds = []
    prob = []
    for x in X:
        class_probs = []
        for c in range(len(class_priors)):
            # 當特徵數很多的時候，相乘之下所算出的機率值會非常小 => 故取Log
            log_class_prob = np.log(class_priors[c]) # => P(c)
            log_class_likelihood = -0.5 * np.log(2 * np.pi * np.square(likelihood[c][1])) - \
            0.5 * np.square(x - likelihood[c][0]) / np.square(likelihood[c][1]) # => P(x|c)
            class_probs.append(log_class_prob + np.sum(log_class_likelihood))
        prob.append(class_probs)
        preds.append(np.argmax(class_probs))
    return preds, prob
```

```python
class_priors, likelihood = Bayes_calprob(x_train,y_train)
preds, prob = predict(x_test, class_priors, likelihood)
print('Acc :',sum(preds == y_test)/len(y_test))
```

```
Acc : 0.9666666666666667
```

圖 2、函式 predict(X, class_priors, likelihood)與測試集的準確率

3.

主成分分析算法(PCA)是最常用的線性降維方法，它的目標是通過某種線性投影，將高維的數據映射到低維的空間中，並期望在所投影的維度上數據的信息量最大，也就是方差最大，因為這樣能最大限度的將資料分開，以此使用較少的數據維度，同時保留住較多原數據點的特性。

PCA 降維為盡量保持特徵貢獻度的情況下，透過降低特徵維度以減少計算成本。一般來說，特徵數愈多，愈容易得到好的預測效果，然而，當特徵過多時，也容易對我們的運算造成龐大的負荷，例如今天要預測股價未來漲或跌，同時搜集了股價、市值、營收、股利當作特徵，會發現市值與營收是有正向關係而股價與股利也有一個正向關係，但終究這四個特徵仍是不同的，因此若透過特徵選擇來減少特徵，終究會丟失一些重要資訊，而 PCA 降維裡可以想像它把營收跟市值合併成一個特徵，股價跟股利合併成一個特徵，剩下兩個合併而成的因子。

將提供的 13 項特徵，畫成熱力圖與分佈圖，看一下每個特徵之間的相關性，如圖 3、4，從圖中可以看出，Total phenols、Flavanoids 有非常強的相關性，所以再利用機器學習等方法預測類別時，這幾項特徵提供的幫助是一樣的，反而會增加電腦的運算量，所以此時就可以將這 2 項特徵，合成成 1 項特徵即可，而 PCA 主要就是協助找出特徵之間相似度高的特徵，透過合併已達到降維的目的。

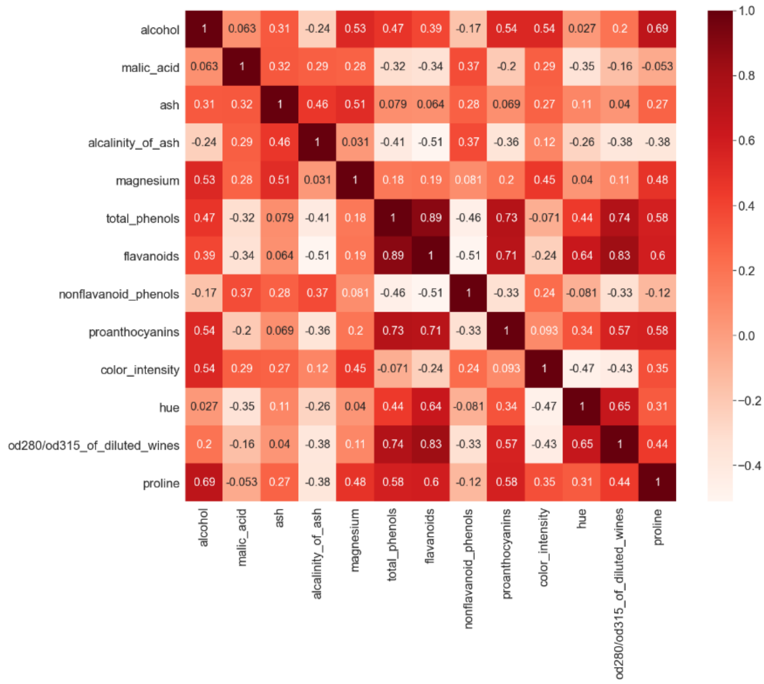

圖 3、各項特徵的相關性

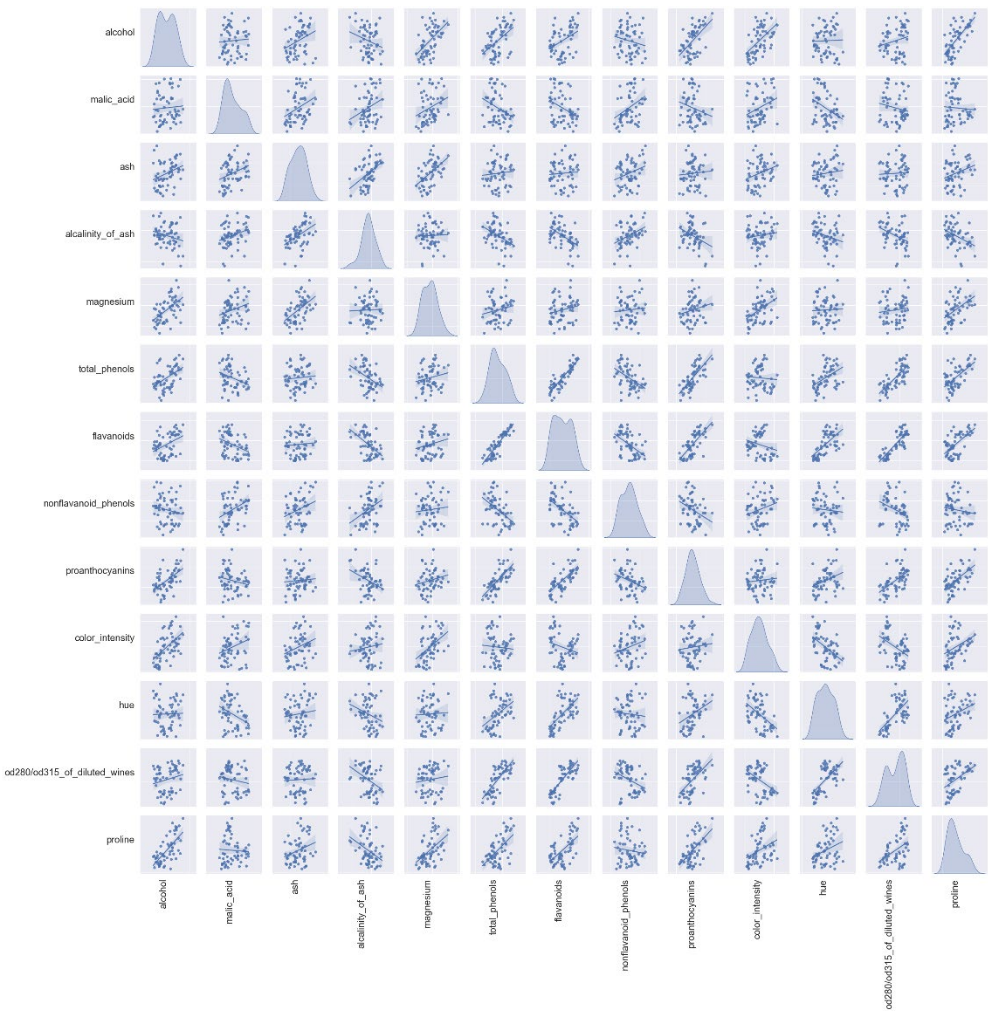

圖 4、特徵兩兩之間的關係

Python 的 sklearn 函式庫提供大量常見的機器學習演算法和許多實用的資料集合，故此次作業將利用其提供的 PCA 進行實驗，首先必須先將特徵正規化，因為每組特徵可能會因為單位的不同或數字大小的代表性不同，造成各自變化的程度不一，進而影響統計分析的結果，之後將 component 設成 2，代表希望留下的特徵維度，最後利用 pca.explained_variance_ratio_觀察降維後的每項特徵提供多少貢獻度，並將這些貢獻度總合起來，看看總共保留了多少特徵貢獻度，將結果繪製成圖 5，可以發現兩個主成分之和的貢獻度僅 0.566，且將 2 維的資料分布繪製出來，如圖 6，會發現紅綠兩個類別的分布重疊到了，這可能是因為兩個主成分的貢獻度不足以代表整個資料集所導致的。

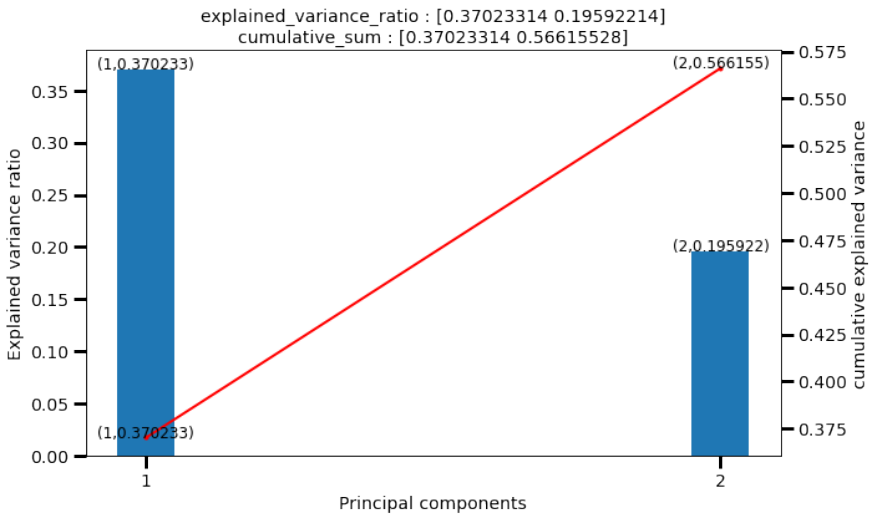

圖 5、explained variance ratio and cumulative sum

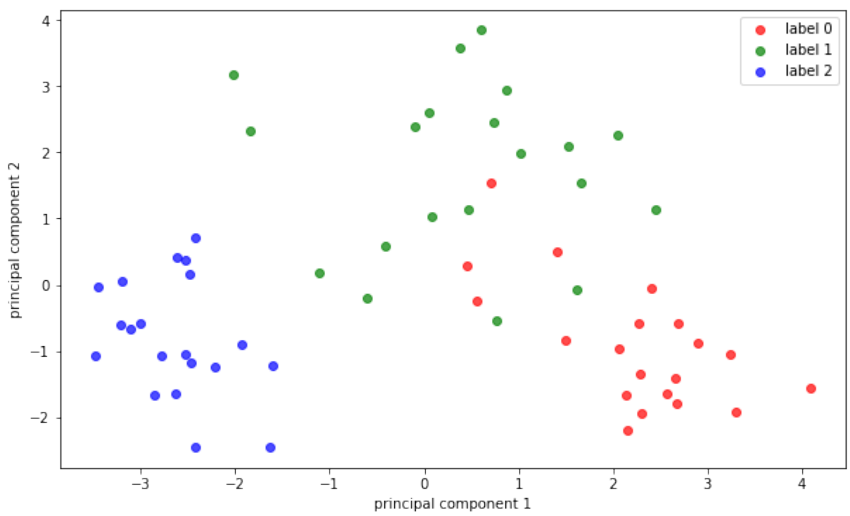

圖 6、降維後的資料分布

綜合以上的結果，分析 PCA 的優缺點:

優點：

- 降低資料的複雜性，識別最重要的多個特徵。
- 達到降維目的，避免高維的計算量大。
- 以方差衡量信息的無監督學習，不受樣本標籤限制。
- 由於協方差矩陣對稱，因此 k 個特徵向量之間兩兩正交，也就是各主成分份間正交，正交就肯定線性不相關，可消除原始數據成分間的相互影響。

缺點：

- 主成分解釋其含義往往具有一定的模糊性，不如原始樣本完整。
- 對降維最終得到數目，不能很好地估計。
- 貢獻率小的主成分往往可能含有對樣本差異的重要信息，也就是可能對於區分樣本的類別更有用。
- 對於維度之間非線性的依賴關係不能得到很好的結果。

4.

從前面的介紹可以得知，在貝氏定理中，後驗概率(Posterior Probability)是觀察到某些特徵後對類別的概率，而後驗概率的計算依賴於兩個重要因素：先驗概率(Prior Probability)和似然函數(Likelihood Function)，先驗概率是在觀察到任何特徵之前，對每個類別的概率，似然函數表示給定某個類別時，觀察到特徵的概率。

而學生認為先驗概率對於後驗概率的影響是小的，假設類別 0、1 的資料皆充足，但類別 2 僅有少少的幾筆，此時貝式分類器仍不會將測試集中的類別 2 分類至類別 0、1，因為似然函數的關係，為了印證此假設，學生特別做了一個實驗，就是設定先驗概率的初始值，從皆是 1/3 開始，因為這樣代表 3 個類別的資料皆分布均勻，之後慢慢提升類別 0、1 的先驗概率，並降低類別 2，最後值將將公式中的先驗概率拿掉，如表 1，可以發現，整體的準確率並沒有大幅的下降，

皆維持 9 成以上，這也說明了先驗概率對於後驗概率的影響是小的，但先驗概率應要避免為零，因為為零就代表訓練集內沒有該類別的資料，故就不要肖想測試集中的資料能被貝氏分類器分到該類別，如此一來就會導致結果很差，如下表中的[0.5, 0.5, 0]。

情況(每個類別出現的機率)	準確率(%)
[1/3, 1/3, 1/3]	96.67
[0.4, 0.4, 0.2]	98.33
[0.45, 0.45, 0.1]	96.67
[0.49, 0.49, 0.02]	96.67
[0.495, 0.495, 0.01]	98.33
先驗概率這項拿掉	93.33
[0.5, 0.5, 0]	**65**

參考文獻

[1] https://pyecontech.com/2020/02/27/bayesian_classifier/

[2] https://www.mropengate.com/2015/06/ai-ch14-3-naive-bayes-classifier.html

[3]https://roger010620.medium.com/%E8%B2%9D%E6%B0%8F%E5%88%86%E9%A1%9E%E5%99%A8-naive-bayes-classifier-%E5%90%ABpython%E5%AF%A6%E4%BD%9C-66701688db02

[4] https://medium.com/chung-yi/ml%E5%85%A5%E9%96%80-%E4%BA%8C%E5%8D%81-pca%E9%99%8D%E7%B6%AD-9ad3a09782ad

[5] https://ithelp.ithome.com.tw/articles/10267685