## Part 2. Programming assignment :

2.

因在切割數據做為訓練集和測試集時具有隨機性，所以每次分類的準確度不盡相同，但平均準確度大於 95%。

```
Accuracy rate:  95.0 %
Accuracy rate:  95.0 %
Accuracy rate:  96.67 %
Accuracy rate:  96.67 %
Accuracy rate:  91.67 %
Accuracy rate:  98.33 %
Accuracy rate:  98.33 %
Accuracy rate:  98.33 %
Accuracy rate:  90.0 %
Accuracy rate:  100.0 %
Accuracy rate:  93.33 %
Accuracy rate:  93.33 %
Accuracy rate:  95.0 %
Accuracy rate:  95.0 %
Accuracy rate:  96.67 %
Accuracy rate:  93.33 %
Accuracy rate:  93.33 %
Accuracy rate:  96.67 %
Accuracy rate:  96.67 %
Accuracy rate:  91.67 %
Accuracy rate:  98.33 %
Accuracy rate:  98.33 %
Accuracy rate:  96.67 %
Accuracy rate:  95.0 %
Accuracy rate:  93.33 %
Accuracy rate:  98.33 %
Accuracy rate:  91.67 %
Accuracy rate:  98.33 %
Accuracy rate:  98.33 %
Accuracy rate:  86.67 %
-----------------------------------
平均準確度：   95.33 %
```

關於後驗機率(posterior probability)，我是參考維基百科的「單純貝氏分類器」[1]上的實例，並將其計算過程改為符合這次作業的需求，計算過程如下：

先使用 numpy.mean 和 numpy.var 分別計算三個 target 中各 features 的平均值($\mu$)與變異數($\sigma^2$)，

再計算三個 target 的事前機率(priori probability)：

Ex：$P(target0) = \frac{target0\text{ 在訓練集中的數量}}{\text{訓練集資料總數}}$

再使用公式計算三個 target 中各 features 的概似函數(likelihood)：

Ex：$p(alcohol \mid target0) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(\frac{-(alcohol-\mu)^2}{2\sigma^2}\right)}$

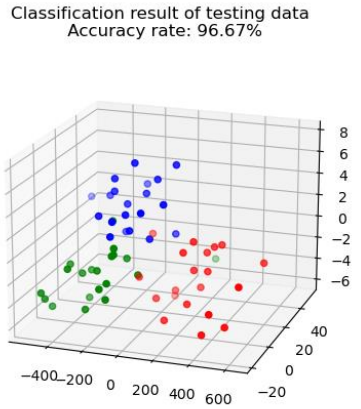
即可求得三個 target 的後驗機率：

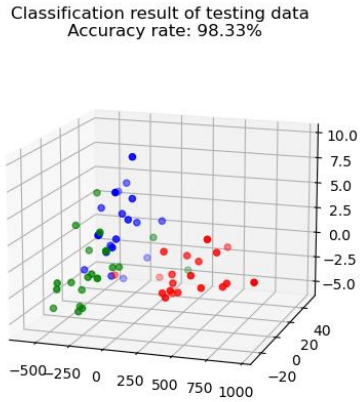
Ex：posterior (target0) =

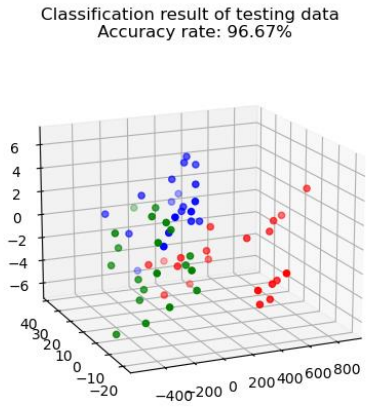
$$\frac{P(target0)p(alcohol \mid target0)p(maliacid \mid target0)p(ash \mid target0)\ldots}{evidence}$$

3.

以下是我實作分類結果可視化的其中三張節錄：



此分類結果可視化的 figures 是使用 sklearn.decomposition 中的 PCA function 來產生，PCA 是 Principal Component Analysis 的縮寫，是一種非監督式降維的演算法，且我在前幾週的系統理論課程正好有上到。

PCA 的主要目的是特徵擷取，將擁有高維的 features 降維到低維度，同時又可大部分地保存資料特性，盡量減小降維造成的訊息損失。

以下是 PCA function 在我的 code 中應用[2]的狀況：

```
df_test_list = df_test.values.tolist() # 將測試集的表單轉為 list
pca = PCA(n_components = 3) # 將資料降為三維
pca.fit(df_test_list) # fit 出特徵映射後使資料變異量最大的投影向量
test_pca = pca.transform(df_test_list)
ax = plt.figure().add_subplot(projection = '3d') #將 figure 設定為三維
ax.scatter(test_pca[:20,0], test_pca[:20,1], test_pca[:20,2],color = 'r')
ax.scatter(test_pca[20:40,0], test_pca[20:40,1], test_pca[20:40,2],color = 'g')
ax.scatter(test_pca[40:60,0], test_pca[40:60,1], test_pca[40:60,2],color = 'b')
plt.title('Classification result of testing data \n Accuracy rate: ' +
str(round(c/60*100,2)) + '%')
```

## 4.

在本次分類的實作上，三個 target 的事前機率大致分別為 0.3, 0.4, 0.1，在乘到後驗機率時會稍微影響到後驗機率的數值，進而影響分類結果。

也就是說，當訓練集中的某個 target 資料較多時，該 target 就有更高的事前機率，即其它的概似機率不需太高即可辨認這個 target，反之，若 target 有較低的事前機率，則需要有更多的其它概似機率來證明它可能發生。

Reference：

[1] 單純貝氏分類器, 維基百科
https://zh.wikipedia.org/zh-tw/%E6%9C%B4%E7%B4%A0%E8%B4%9D%E5%8F%B6%E6%96%AF%E5%88%86%E7%B1%BB%E5%99%A8

[2] PCA 主成分分析的可视化(Python), 知乎
https://zhuanlan.zhihu.com/p/523732469