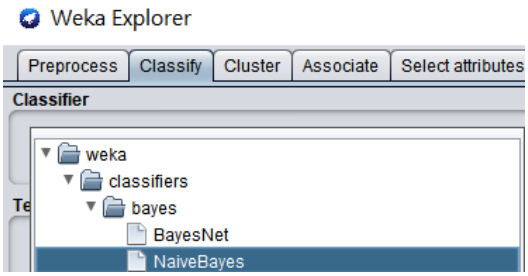
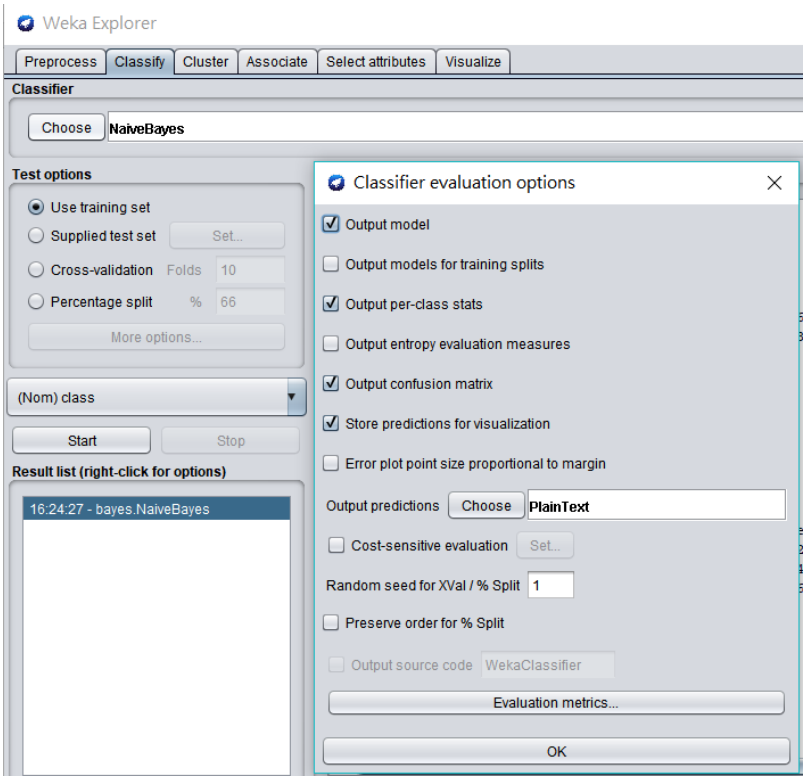
1. 重要步驟截圖及說明

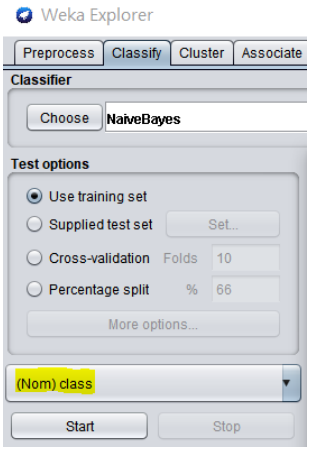
i. 打開 weka 的 Explore 之後，在 Classify Panel 中的 Classifier 選取「NaiveBayes」。



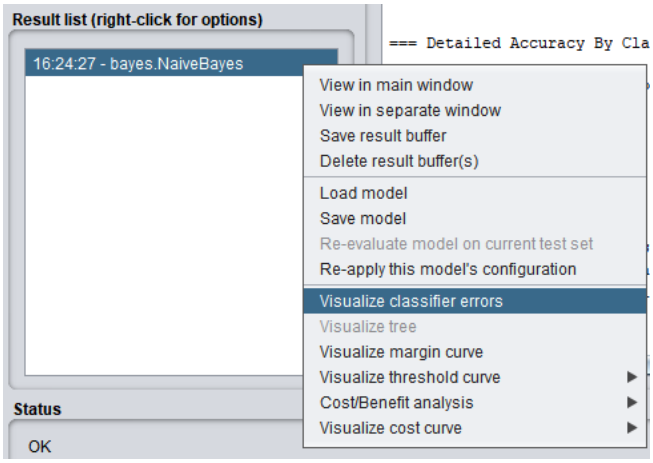
ii. 在 Test options 中勾選「Use training set」，並在點選「More options」後，在「Output predictions 欄位選擇「PlainText」。



iii. 在螢光筆處選擇「(Nom)class」，並點選「Start」開始分析。



iv. 若要利用 Visualize Classifier Errors，則右鍵 Result list 中執行的結果，並選取「Visualize Classifier Errors」。



(a) 錯誤率為 23.6979%

```
Incorrectly Classified Instances       182               23.6979 %
```

由題，「Test dataset instances 被分類到 tested_negative class，但實際上屬於 tested_positive class」，代表 Confusion matrix 中的「false negative (FN)」，由圖可知總共有 79 筆 Test dataset instances 被分類到 FN，可得百分比為 79/768 * 100% = 10.2864%

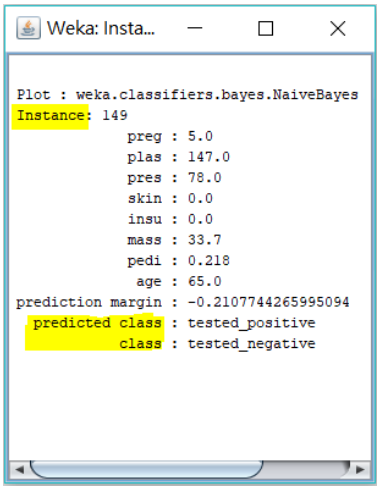
```
=== Confusion Matrix ===

   a   b   <-- classified as
 421  79 |   a = tested_negative
 103 165 |   b = tested_positive
```

(b) 欄位 error 出現 “+” 代表欄位 actual 被錯誤地預測為欄位 predicted 的情況；以 inst7 為例，代表 tested_positive 被錯誤地預測為 tested_negative。

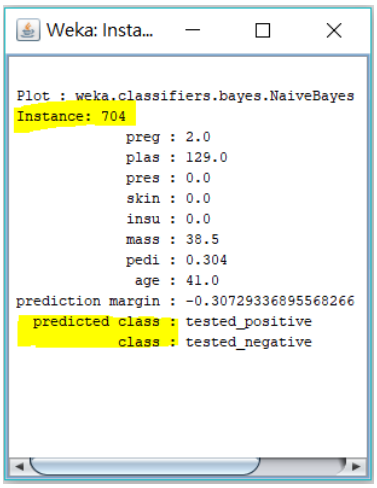
```
Time taken to build model: 0 seconds

=== Predictions on training set ===

    inst#     actual  predicted error prediction
        1 2:tested_positive 2:tested_positive       0.678
        2 1:tested_negative 1:tested_negative       0.981
        3 2:tested_positive 2:tested_positive       0.815
        4 1:tested_negative 1:tested_negative       0.987
        5 2:tested_positive 2:tested_positive       1
        6 1:tested_negative 1:tested_negative       0.941
        7 2:tested_positive 1:tested_negative   +   0.974
```

(c)

I. 若點取左上角的藍色方框，可得預測錯誤的資料點的詳細資訊（如下圖）

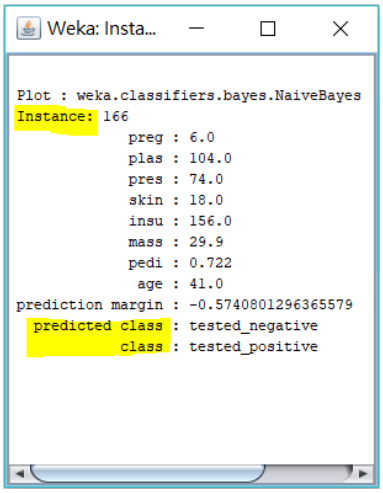


可由 instance 得知為第 149 筆資料，預測的 class 為 tested_positive，但實際上為 tested_negative，屬於 Confusion matrix 中的 FN。



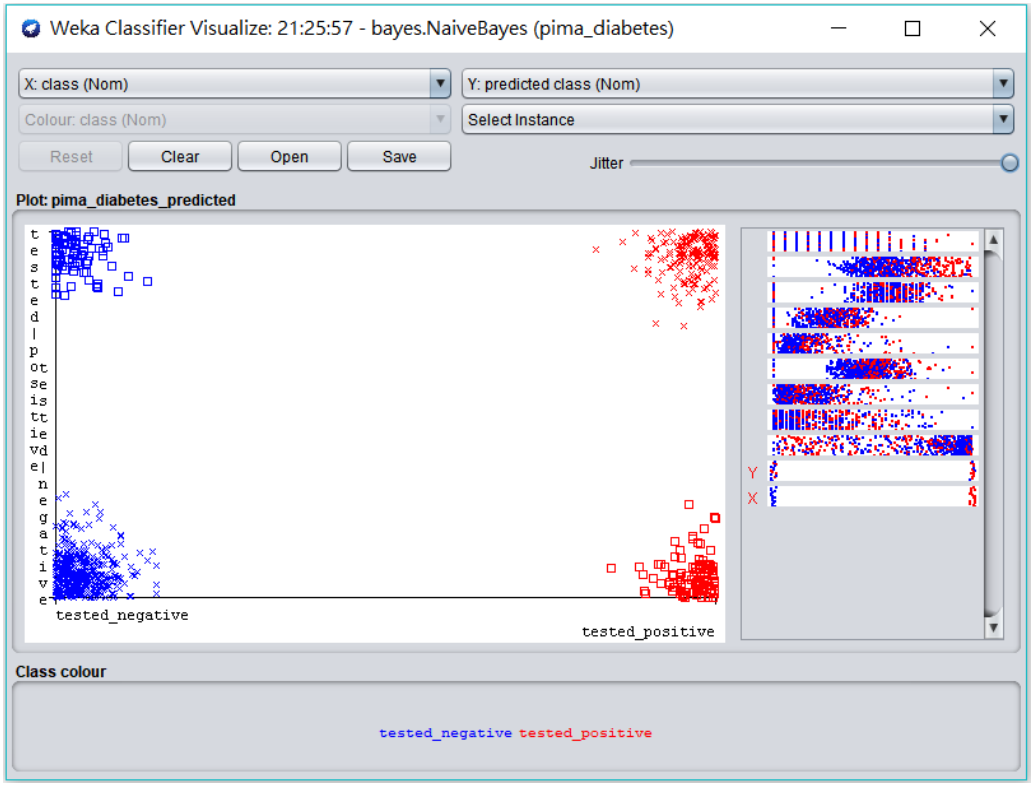
可由 instance 得知為第 704 筆資料，預測的 class 為 tested_positive，但實際上為 tested_negative，屬於 Confusion matrix 中的 FN。

II. 若點取右上角的紅色方框，可得預測錯誤的資料點的詳細資訊（如下圖）



可由 instance 得知為第 166 筆資料，預測的 class 為 tested_ negative，但實際上為 tested_ positive，屬於 Confusion matrix 中的 FP。

(d)



由(c)可知，當 x 軸代表 class(Nom)、y 軸代表 predicted class(Nom)時，左上角的資料代表 Confusion matrix 中的 FN，右下角代表 Confusion matrix 中的 FP；左下角的資料皆為 tested_negative，因此屬於 Confusion matrix 中的 TN，而右上角的資料皆為 tested_positive，因此屬於 Confusion matrix 中的 TP。

2.

(a) 由圖所示， preg 平均值為 3.845052，plas 平均值為 120.894531，pres 平均值為 68.105469，skin 平均值為 20.536458，insu 平均值為 79.799479，mass 平均值為 31.992578，pedi 平均值為 0.471876，age 平均值為 33.240885

	preg	plas	pres	skin	insu	mass	pedi	age
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885

(b)

i. 開啟 Anaconda Navigator 的 Jupyter。

ii. 將 Naïve Bayes_weather_example.ipynb 與 diabetes.csv 上傳到 Jupyter，並開啟 Naïve Bayes_weather_example.ipynb。

iii. 將 weather 改成 diabetes。

```
In [8]: import pandas as pd
        #讀取CSV檔案
        data = pd.read_csv('diabetes.csv')
```

iv. 填入所需的 x:input 和 y:output。

切分input 和output

```
#x:input
x=data.loc[:,['preg','plas','pres','skin','insu','mass','pedi','age']]
#y:output
y=data.loc[:,['class']]
```

v. 由於其餘屬性皆為數字，所以只將 class 轉為數字 label、並將屬性合併變成 list

sklearn: Naive Bayes Classifier

```
from sklearn import preprocessing
#將屬性轉為數字label
le = preprocessing.LabelEncoder()

#將class轉為數字label
#class: tested_negative: 0 ,tested_positive: 1
Y_class_label=le.fit_transform(y['class'])

#將屬性合併
#變成list
feature=list(zip(x.preg, x.plas,x.pres,x.skin,x.insu,x.mass,x.pedi,x.age))

#轉成array
import numpy as np
features=np.asarray(feature)
```

vi. 變更參數名稱

```
## 訓練模型：訓練集
```

```
#Import Gaussian Naive Bayes 模型 (高斯樸素貝氏)
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()

# 訓練集訓練模型
# model.fit(x, y)
model.fit(features, Y_class_label)
```

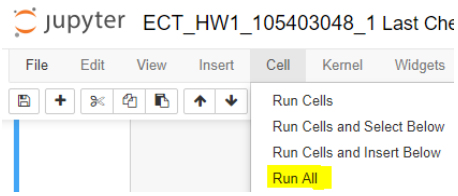
測試集測試模型

```
expected = Y_class_label
predicted = model.predict(features)
from sklearn import metrics
print(metrics.classification_report(expected, predicted))
```

vii. 依(e)題，變更最後的預測參數

```
# 當'preg'=2, 'plas'=1, 'pres'=0, 'skin'=0, 'insu'=2, 'mass'=1,'pedi=2', 'age'=20時，最終的 output class 為何
predicted= model.predict([[2,1,0,0,2,1,2,20]])

print ("Predicted Value:", predicted)
```

viii. 按下 Run all 執行所有 cell 的程式，即可得所有 cell 之 output。



(c) Precision(P) = $\frac{TP}{TP+FP}$，表示的是預測為正的樣本中有多少是真正的正樣本。

Recall(R) = $\frac{TP}{TP+FN}$ ，表示的是樣本中的正例有多少是被預測正確了。

F1-Score = $\frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}$ ， F1 值即 Precision 和 Recall 的調和均值。

測試集測試模型

```
expected = Y_class_label
predicted = model.predict(features)
from sklearn import metrics
print(metrics.classification_report(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       0.80      0.84      0.82       500
           1       0.68      0.62      0.64       268

   micro avg       0.76      0.76      0.76       768
   macro avg       0.74      0.73      0.73       768
weighted avg       0.76      0.76      0.76       768
```

(d) 共有 421 筆資料屬於 TN，79 筆資料屬於 FP，103 筆資料屬於 FN，165 筆資料屬於 TP。

```
print(metrics.confusion_matrix(expected, predicted))
```

```
[[421  79]
 [103 165]]
```

(e) 結果為 1，代表 tested_positive

```
# 當'preg'=2, 'plas'=1, 'pres'=0, 'skin'=0, 'insu'=2, 'mass'=1,'pedi=2', 'age'=20時，最終的 output class 為何
predicted= model.predict([[2,1,0,0,2,1,2,20]])

print ("Predicted Value:", predicted)
```

```
Predicted Value: [1]
```

(f) Python 相較於 Weka 而言，應用較為廣泛，可以再 cell 中執行更多的預測情況，如題(e)所示；但 Weka 提供較佳的用戶者體驗，操作介面較簡單且直覺。