# Assignment_01

> 106403551 呂晟維 這次作業是用**notebook轉成pdf**的方式繳交，如果助教覺得這樣不好閱讀，之後就改用**word**當編輯工具。辛苦了**!!**

1. 用 Weka 軟體對 mushrooms.arff 利用 Naïve Bayes 進行 Supervised learning 選擇 ”Use training set”, 設定 Attribute: type 為 Output, 在過程中對重要步驟截圖加以說明並回答以下問題：

   - (a) 解釋 Classifier Output, Test data 的錯誤率為多少？有多少 Test dataset instances 被分類到有毒 poisonous 但實際上屬於可食用的 edible 請利用 Confusion matrix 解釋。(25%)
   - (b) 在 Output predictions 結果中 "+" 代表的意義為何，請截圖並解釋。(10%)
   - (c) 請使用 Visualize Classifier Errors, 解釋此圖與 Confusion matrix之間的關係。(15%)

2. 用 python 對 mushrooms.csv 進行 Supervised learning 中的 Naïve Bayes 分析 並回答以下問題：

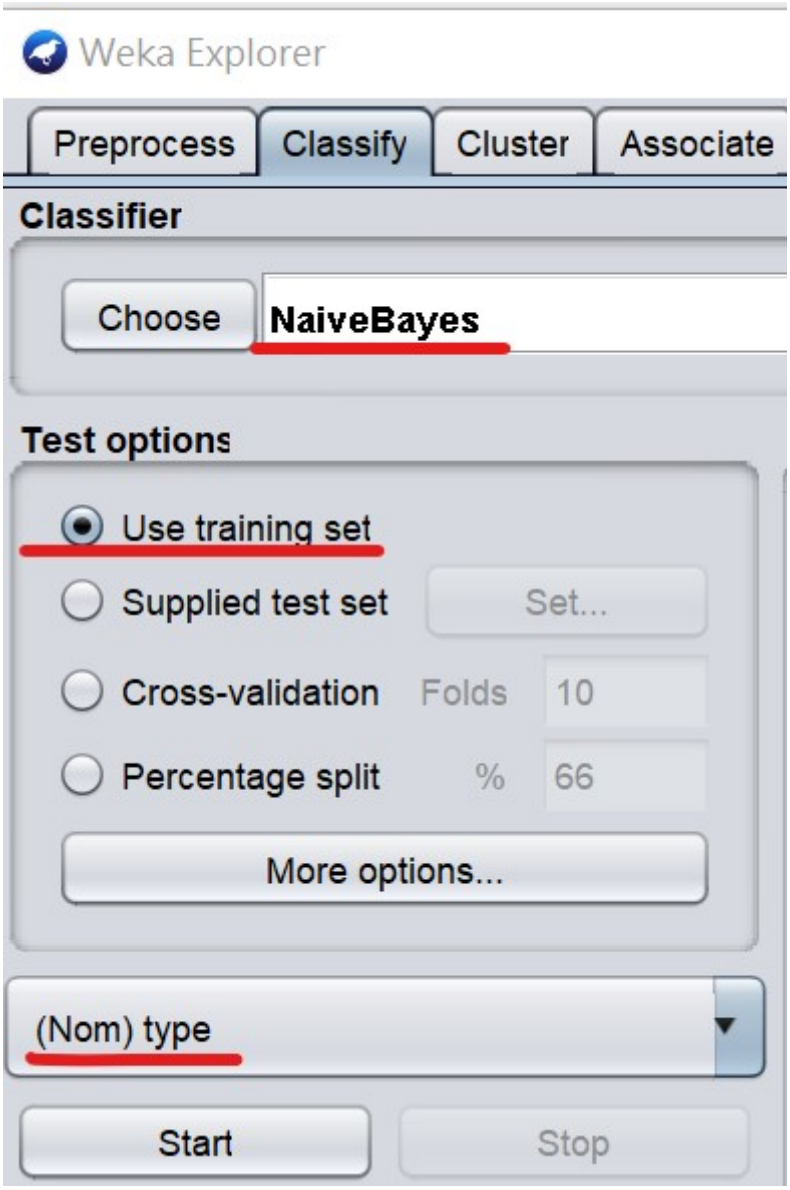
   - (a) 在過程中對所有重要程式步驟進行截圖並 加以說明，越詳盡越好。 (15%)
   - (b) 請問 mushrooms 資料集中共有多少 instance? 是否包含空值的欄位(null)? (10%)
   - (c) 請問欄位 stalk_color_above_ring 有幾種不同的 value? (5%)
   - (d) 請利用 metrics.confusion_matrix() 呈現出混淆矩陣，並截圖加以說明。 (10%)
   - (e) 請利用 metrics.classification_report 列出模型的準確率並與 Weka 的結果比較何者較高? (10%)

繳交期限 : 3/25 (三) 中午 12:00
請轉檔為 PDF 格式 , 檔名為 ECT_HW1_學號_版本.pdf並同時附上 python 的 .ipynb 檔，命名格式同上。

# Weka

## (a) 錯誤率 & Confusion matrix

Weka的Input和參數設定如下：




我們使用training set來作為testing data來跑Naïve Bayes方法，使用weka跑出來的部分result如下：

```
=== Summary ===

Correctly Classified Instances        7984               98.2767 %
Incorrectly Classified Instances       140                1.7233 % <-- 錯誤率
Kappa statistic                          0.9655
Mean absolute error                      0.0222
Root mean squared error                  0.1209
Relative absolute error                  4.439  %
Root relative squared error             24.2041 %
Total Number of Instances             8124

=== Confusion Matrix ===

    a    b   <-- classified as
 3822   94 |    a = poisonous
   46 4162 |    b = edible
```

由Incorrectly Classified Instances得知，其中錯誤率為**1.7233 %**，8124筆資料中共有140個錯誤。正確率是98.2767 %，有7984筆正確的預測。

再透過Confusion Matrix的結果判斷，當香菇為可食用的edible，但是被誤分類為有毒的poisonous的情況是Matrix中的左下角的區塊，數量有**46**個。

Confusion Matrix詳細的解說在第c小題會介紹。

# (b) 在 Output predictions 結果中 "+" 代表的意義為何，請截圖並解釋。

勾選Output predictions為Plane text形式，部分的output結果如下：

> 附註 1:p的意思是label是p(有毒的)，轉成數值代號1。

```
=== Predictions on training set ===

    inst#     actual  predicted error prediction
        1        1:p        1:p       0.985
        2        2:e        2:e       1
        3        2:e        2:e       1
        4        1:p        1:p       0.972
        5        2:e        2:e       1
        6        2:e        2:e       1
        7        2:e        2:e       1
        8        2:e        2:e       1
        9        1:p        1:p       0.973
       10        2:e        2:e       1
      ...
     4107        1:p        2:e   +   1
      ...
     4277        2:e        1:p   +   0.906
      ...
```

"+"號代表在Testing的過程中有error，會在error欄位顯示。

可以發現前10筆的資料都預測正確，一直到第4107筆測試資料才出現第一個Prediction error，他的真實值是1:p，是有毒的香菇，但是Naïve Bayes模型卻預測錯誤成2:e可食用的，屬於FN(false negative)的錯誤。然後第4277筆測試資料出現第二個Prediction error，他的真實值是2:e，是有可食用的香菇，但模型預測他為有毒的，屬於FP(false positive)的錯誤。

附註：預測錯誤都是false類型。

# (c) 請使用 Visualize Classifier Errors, 解釋此圖與 Confusion matrix之間的關係。

Confusion Matrix的是一個Table，表示Classerifier預測的效能。我們也可以透過圖示(visual classifier errors)來看到所有分布在2維平面上的預測值和真實值的關係，將虛擬的演算法用圖像化的方式呈現他的效能。

比較一下Confusion Matrix和Visualized的輸出，如果將原本的Confusion Matrix轉換一下位置/方向，讓xy軸跟Visualized error的一樣。

原本

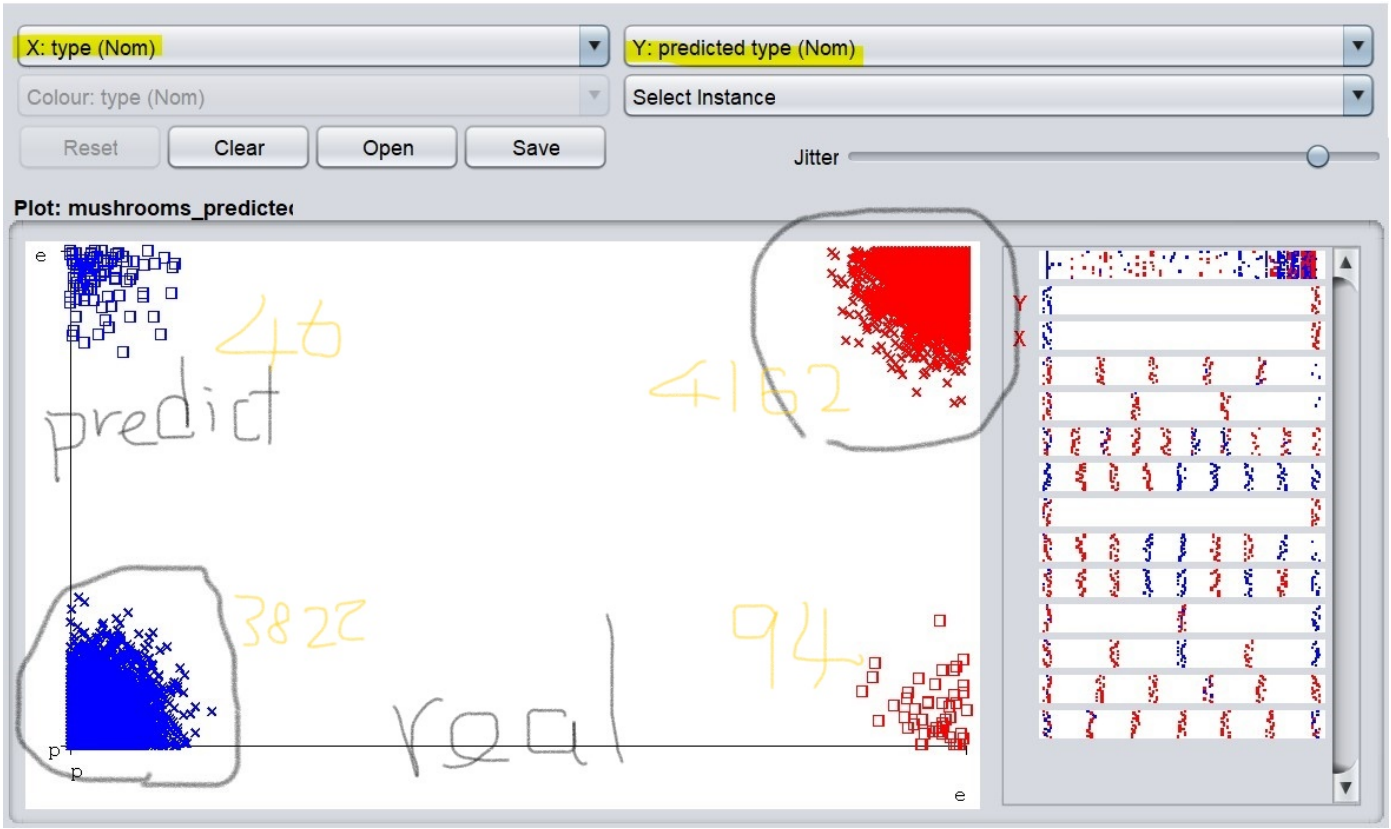
```
      p    e   <-- classified as
   3822   94 |    p = poisonous
     46 4162 |    e = edible

  (Predicted)
    TP   FN | (Actual)
    FP   TN |
```

調整後

```
   (Actual)
     p    e
    FP   TN | e (Predicted)
    TP   FN | p  p = positive
    --------
    46 4162 | e
  3822   94 | p
```

`Visualized error` 正確的預測 是用黑筆框起來的2塊區域，左下角是TP、有上角是TN，True開頭的都是正確的預測，使用叉叉來表示。而方形則是錯誤的預測。由於我們將Matrix的xy軸方向顛倒了，數值都可以直接對上Matrix的圖示。




# Python

由於第a小題的內容比較多，我放到最後面說明~

## (b) 請問 **mushrooms** 資料集中共有多少 **instance?** 是否包含空值的欄位(null)? (10%)

這邊使用 `dataFrame.info` 看有幾筆資料、每個欄位的資料型別是什麼(int, float..)、有無空值(null)的存在、佔據多少記憶體。

資料集中共有**8123**筆資料，沒有包含空值的欄位。

```
>>> data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123              <-- 有8123筆instance
Data columns (total 12 columns):                 <-- 有12種attribute
type                        8124 non-null object   <-- 通通都沒有空值
cap_shape                   8124 non-null object
cap_surface                 8124 non-null object
cap_color                   8124 non-null object
odor                        8124 non-null object
stalk_shape                 8124 non-null object
stalk_color_above_ring      8124 non-null object
stalk_color_below_ring      8124 non-null object
ring_number                 8124 non-null object
ring_type                   8124 non-null object
population                  8124 non-null object
habitat                     8124 non-null object
dtypes: object(12)
memory usage: 761.8+ KB
```

## (c) 請問欄位 stalk_color_above_ring 有幾種不同的 value? (5%)

這邊使用 `dataFrame.describe` 看資料的平均值、分佈情況、是否有資料傾斜Skew的問題。

欄位 `stalk_color_above_ring` 有9個unique的value，代表在此欄位8124筆資料中只有9種不同的值，代表9種不同的顏色。眾數是w(white)白色，出現4464次。

```
In [4]: data.describe()
```

Out[4]:

| | type | cap_shape | cap_surface | cap_color | odor | stalk_shape | stalk_color_above_ring | talk_color_below_ring | ring_numb |
|---|---|---|---|---|---|---|---|---|---|
| count | 8124 | 8124 | 8124 | 8124 | 8124 | 8124 | 8124 | 8124 | 81 |
| unique | 2 | 6 | 4 | 10 | 9 | 2 | 9 | 9 | |
| top | e | x | y | n | n | t | w | w | |
| freq | 4208 | 3656 | 3244 | 2284 | 3528 | 4608 | 4464 | 4384 | 74 |

## (d) 請利用 metrics.confusion_matrix() 呈現出混淆矩陣，並截圖加以說明。(10%)

由於官方文件並沒有說明postive和negative的位置，所以我們參考維基百科來做解釋。

得知預測正確(True)的有3296+3423=6719筆資料，預測錯誤(False)的有912+493=1405筆資料。其中左上角的3296筆是True Positive，實際有毒預測也有毒；右上角的912筆是False Negative，實際有毒但是預測可以吃，剩下以此類推。

```
(Predicted)
   p     n
 True False | p (Actual)
False  True | n
-----------
  TP    FN  | p
  FP    TN  | n

p = positive = posinious = 1
n = negative = edible = 0
```

程式碼和輸出：

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

[[3296  912]
 [ 493 3423]]
```

## (e) 請利用 metrics.classification_report 列出模型的準確率並與 Weka 的結果比較何者較高? (10%)

函數用於顯示主要分類指標的文本報告，在報告中顯示每個類的精確度，召回率，F1值等信息。我們得知python的精確度為0.83，沒有weka的0.98高。但是由於是使用Training set來做測試weka的準確率太高反而有overfitting的問題。

```
>>> from sklearn import metrics
>>> print(metrics.classification_report(expected, predicted))

              precision    recall  f1-score   support

           0       0.87      0.78      0.82      4208 <-- edible
           1       0.79      0.87      0.83      3916 <-- posinious

    accuracy                           0.83      8124 <-- 準確率
   macro avg       0.83      0.83      0.83      8124
weighted avg       0.83      0.83      0.83      8124
```

## (a) 在過程中對所有重要程式步驟進行截圖並 加以說明，越詳盡越好。 (15%)

於python檔有詳盡的註解和官方文件的連結，這裡就只講解流程了。

### 1. 前置處理資料

首先使用pandas讀入csv檔案，並切分成input和output兩個dataFrame，input是1~11欄，output是第0欄。因為Naive Bayes需要將nominal的資料轉成數值資料(numeric)，所以要再進一步整理資料，因此使用 `le.fit_transform()` 方法。我們也可以透過 `le.inverse_transform()` 來看標籤對應的值是甚麼，得知0是e(edible)，1是p(posinous)。

```
#讀取CSV檔案 轉成dataFrame
import pandas as pd
data = pd.read_csv('mushrooms.csv')

#x:input
x = data.iloc[:,1:12]
#y:output
y = data.loc[:,['type']]
```

處理label

```
from sklearn import preprocessing

#將屬性轉為數字label，le.fit_transform()是將文字轉乘標籤
le = preprocessing.LabelEncoder()

columns = []
for c in x:
    encoded = le.fit_transform(data[c])
    columns.append(encoded)

#play: edible: 0 ,posinous: 1
Y_type_label=le.fit_transform(y.type)

#將屬性合併，使用zip()一次iterate多個陣列，封裝成8124筆data
#變成list
feature=list(zip(columns[0], columns[1], columns[2], columns[3],
                 columns[4], columns[5], columns[6], columns[7],
                 columns[8], columns[9], columns[10]))
#轉成array
import numpy as np
features=np.asarray(feature)
```

不知道le是如何轉換成label的，所以inverse來看一下，得知0是e(edible)，1是p(posinous)。

```
list(le.inverse_transform([0, 0, 1]))
```

**2.** 訓練模型

第二步開始Train model了。假設數據的分布是高斯樸素貝氏，高斯樸素貝氏就是常態分佈的意思，使用 `fit()` 方法來train模型。

```
#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(features, Y_type_label)
```

**3.** 測試模型

要測試模型用predict()方法來餵資料，這邊使用training data來當testing data，因此把features餵進去。要印出測試的結果有2種方法，第一種是 `classification_report()` 方法，查看整理的訓練結果，包含準確度和f1-score等統計數據；第二種是 `confusion_matrix()` 方法，輸出的形式跟weka一樣，可以查看實際值和預測值的關係，和正確錯誤的數量。

```
expected = Y_type_label
predicted = model.predict(features)

from sklearn import metrics
print(metrics.classification_report(expected, predicted))

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