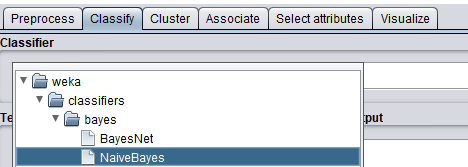
ETC HW1

**Q1.**

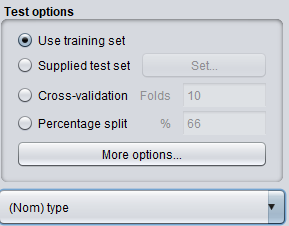
依照題意:

1. 選擇Naïve Bayes 進行 Supervised learning



* 這裡代表選擇Naïve Bayes的方式來做分類

1. 選擇 Use training set , 設定 Attribute: type 為 Output,

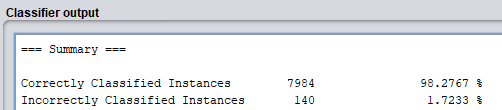


* 「Use training set」: 代表我用這個training set建完model之後，也用這個training set當作testing set來做testing。通常不會這樣做，而是選擇下面的Percentage split來決定多少%要做training，多少%做testing。
* 「Attribute: type為Output」: 代表我把屬性「type」當作output值，也就是貝式定理P(H|E)中的H(hypothesis)。

Ex: 假設H = yes/no , E = sunny/rainy/cloudy, 這裡的H 就代表我所關心的output是yes/no去做某件事情，在E的條件下。

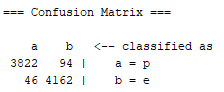
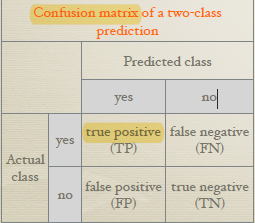
**(a)**

1. Test data錯誤率:



* 由圖可知錯誤率為1.7233%

1. 有多少Test dataset instances被分類到有毒的(poisonous)但實際上屬於可食用的(edible)?用Confusion matrix解釋。

如圖所示，a代表p(有毒的)，b代表e(可食用的)，配合右圖可得知以下資訊:

「預測有毒 + 實際有毒」 = 3822 🡪 預測正確

「預測可食用 + 實際有毒」 = 94 🡪 預測錯誤

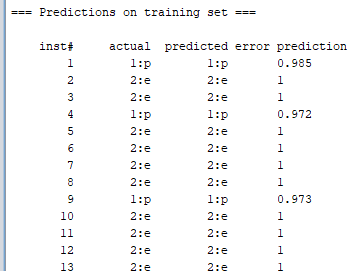
「預測有毒 + 實際可食用」 = 46 🡪 預測錯誤

「預測可食用 + 實際可食用」 = 4162 🡪 預測正確

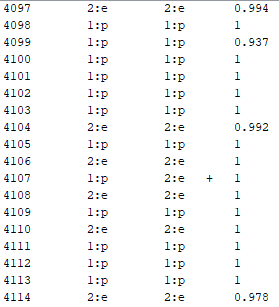
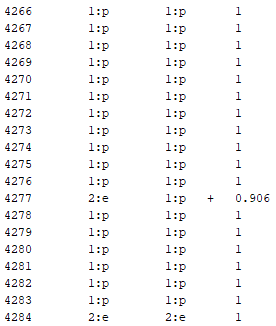
* 依照題意「預測有毒 + 但實際可食用」 = 46(units)

**(b)**

1. output prediction中的“+”代表甚麼?



依此圖可看出，第二個和第三個欄位分別為「實際值」、「預測值」

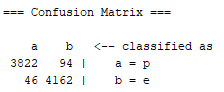
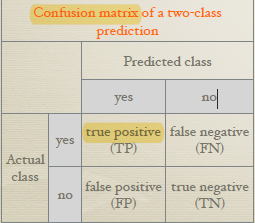
 

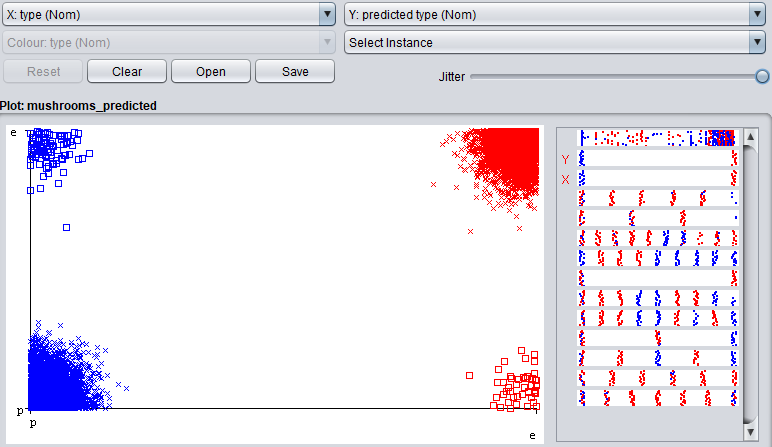
繼續觀察資料可發現，當實際值 = 預測值，並不會出現“+”；但實際值 ≠ 預測值的時候“+”就出現了，因此可推斷“+”代表預測錯誤。

**(C)**

1. 使用 Visualize Classifier Errors, 解釋此圖與 Confusion matrix

之間的關係。

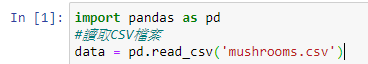


首先，Visualize Classifier Errors的圖最上方可得知，X軸為實際值、Y軸為預測值；再者，觀察圖的四周可發現左下角為p，左上、右下為e，因此可以得知左下為pp(預測p且實際p) = Confusion matrix中的TP(3822)，以此類推右上角為TN(4162)、左上角為FN(94)、右下角為FP(46)。仔細觀察數字更可以發現與圖片上的密度分布相符。順帶一提:

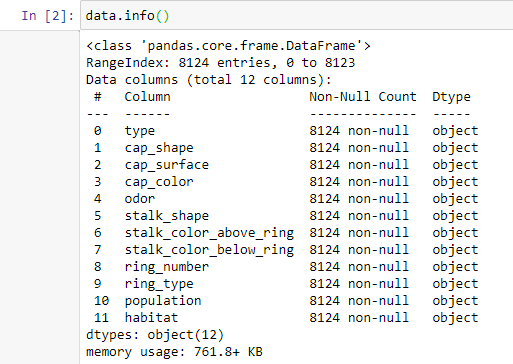
Visualize Classifier Errors圖轉90度就跟Confusion matrix一樣了。

**Q2.**

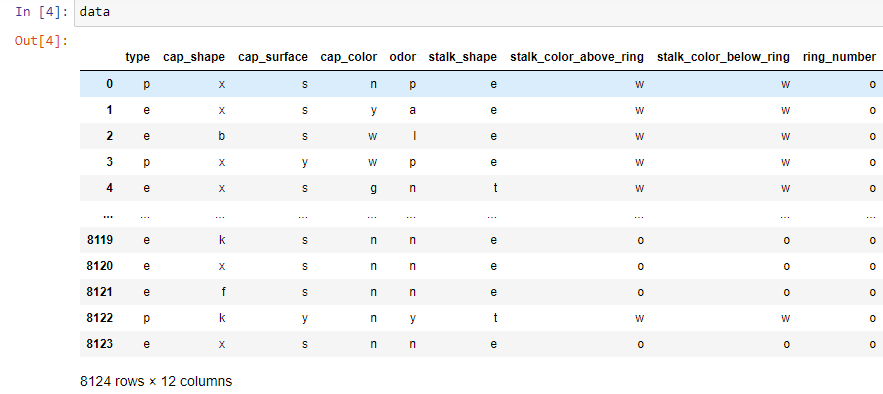
**(a)**



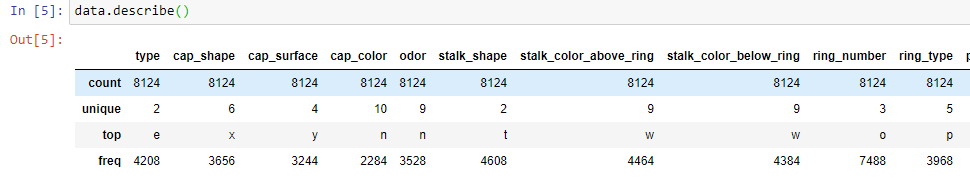
* 先把檔案讀進來



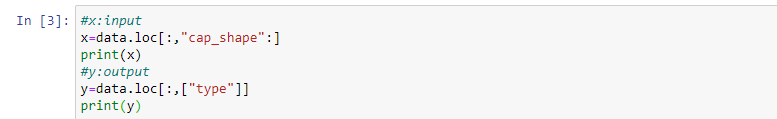
* 然後觀察看看資料大致樣貌，可得知有8124個instance、有12種attribute、沒有null的欄位、data type都為object、記憶體花了多少……



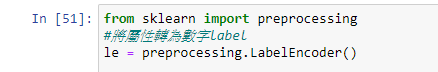
* 這步驟是為了觀察各attribute的value是什麼形式、有哪些可能值，由此可知為nominal。



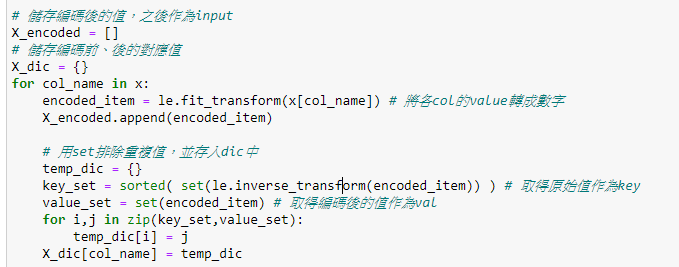
* 用describe()來查看統計資料。由上述可知，nominal不需要mean、std等參數，因此未出現這些參數是正常的。這裡有一點要注意，「unique」代表的意義相等於「Weka中的Distinct」而非unique，可從以下截圖佐證: 



* loc函數前半段處理row，後半段處理col，因此可得知我取得了所有的row(instance)並切割除了「type」以外的所有屬性放入x中當作input；「type」屬性則是放入y當作output。

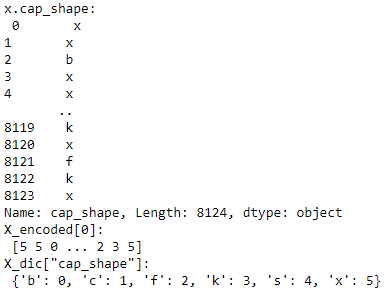


* 放入model訓練前，先把value編碼成數字，所以用了一個LableEncoder()，他可以把數字、英文這種東西按照順序編碼成0~n，其中a < b < … < z；0 < 1 < ……

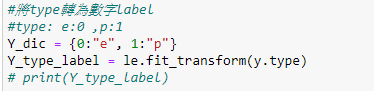


* 我把編碼後的結果存入X\_encoded，並建立一個dictionary(X\_dic)，用來方便確認各個value編碼後的結果為何。

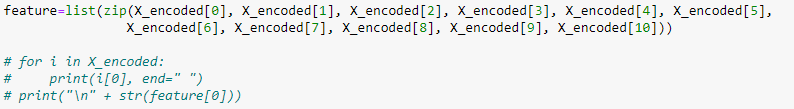




* 由上圖可看出，編碼後前3碼為[5,5,0]對應dic中的[x,x,b]與原始數據相同，之所以key為原始值是為了之後input方便使用，predict時會再次說明。

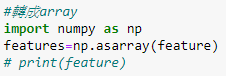


* 這部分與x相同，只是改成y且只有一個「type」屬性，因此不加贅述

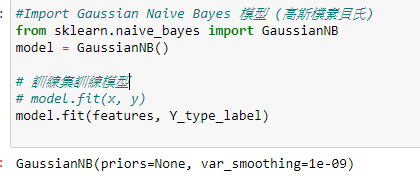


* 用zip把各個編碼過的數據壓縮起來，壓縮格式為「相同index的元素壓成一個元組」，最後再轉成list如下圖所示(只取一個元素當作範例):

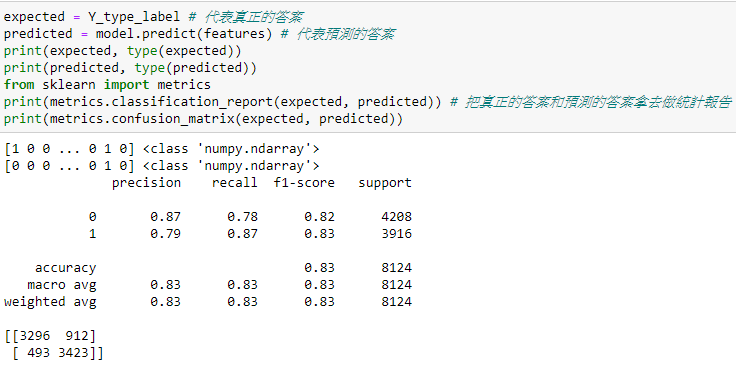
 



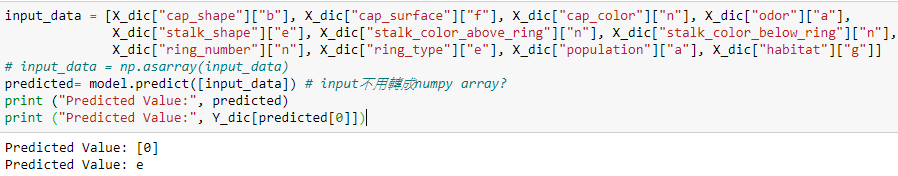
* 轉成numpy array，為了給之後的model使用，而且也多了許多運算可用



* Load一個已經建好的model，用model.fit()把data丟進去訓練



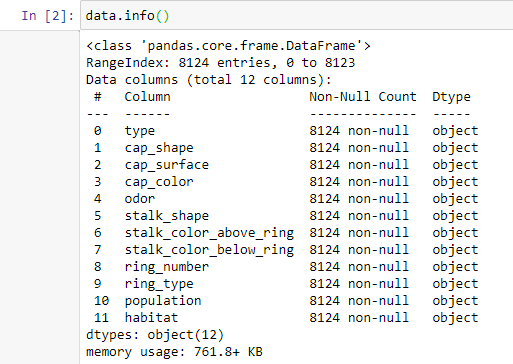
* Model訓練好後，就開始預測了。Expected為真正的答案，predicted為預測出來的答案，metrics.classification\_report(expected, predicted)可以幫我們做分析報告。Precision代表精確度，0.87為預測0正確的機率 = (3296)/(3296+493) = 0.869 ≒ 0.87。Accuracy代表整體的準確度 = 0.87\*(4208)/8124 + 0.79\*(3916)/8124 = 0.83… ≒ 0.83。



* 最後我們可以實際input一組data來預測一個output，這裡我之前特別做的dictionary就是為了在此方便了解自己各個attribute選擇了哪個value作為input。把這個input\_data放入model.predict()中就可以得到預測的答案了。

**(b)**

1. mushrooms 資料集中共有多少instance?



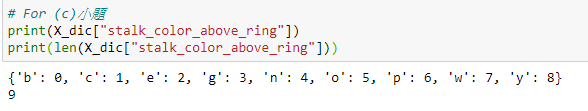
* 由此圖可以看出有8124個instance

2. 是否包含空值(null)的欄位?

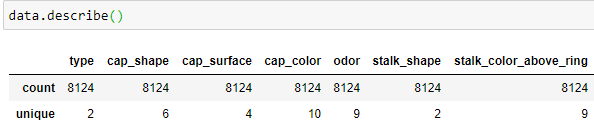
* 同樣由上圖可看出沒有空值(null)的欄位

**(c)**

1. stalk\_color\_above\_ring 有幾種不同的value?



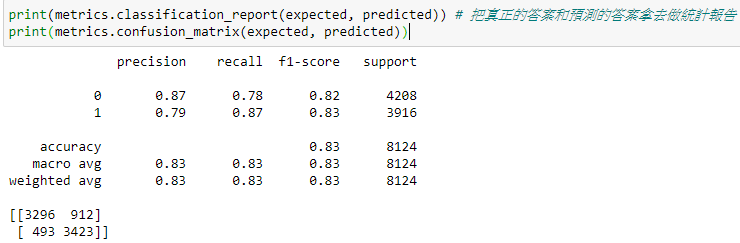
* 如圖所示，用我之前整理的dictionary可以找出有9種不同的value



* 也可以用內建的describe()找出有9種不同的value，但這樣無法得知這些value分別為何，在input的時候不好處理。

**(d)**

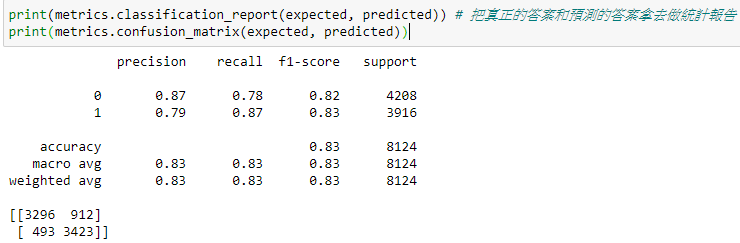
1. 用metrics.confusion\_matrix ( )呈現出混淆矩陣，並截圖加以說明。

根據report的結果，可以推論出confusion\_matrix 的row代表實際值、col代表預測值。所以3296代表TP[0,0]、912代表TN[0,1]、493代表FP[1,0]、3423代表FN[1,1]；在此 e = 0, p = 1。舉例:

0.87代表預測0實際為0的機率 = (3296)/(3296+493) = 0.869 ≒ 0.87

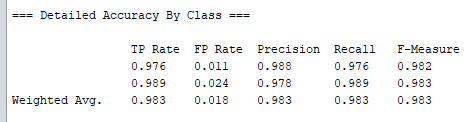
**(e)**

1. 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?



* 準確率(這裡用加權平均) = 0.87\*(4208)/8124 + 0.79\*(3916)/8124 = 0.83… ≒ 0.83。

In Weka:



* 可以看出Weka的0.983 > 0.83