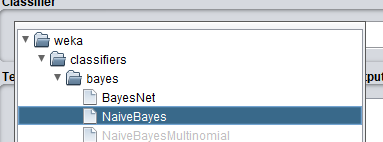
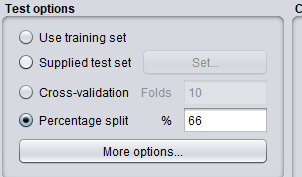
ECT HW6

**Weka Part:**

前置處理:



* 依題目要求使用NaiveBayes

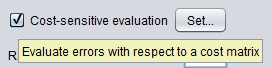


* 依題目要求設定Percentage Split = 66%

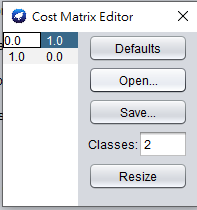
**(a)**



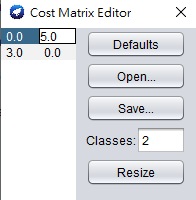
* 點選More Options



* 如圖所示，將框框打勾，並點擊「Set…」



* 會跑出以上的圖片，把Cost Matrix設定成題目要求的數值



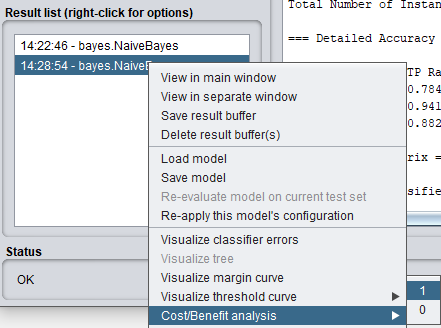
* 如圖所示，雙擊之後便可以修改成所需的數值



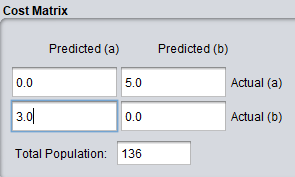
Confusion Matrix:  Cost Matrix: 

* 兩個數值分別如上圖所示，必須配合Confusion Matrix來解釋
* For Total Cost
* 將兩個Matrix相乘，即可得到Total Cost
* 0\*40 + 5\*11 + 3\*5 + 0\*80 = 55+15 = 70
* For Average Cost
* 純粹Total Cost/ 個數 = 700 / (40+11+5+80) = 70 / 136 ≒ 0.5147

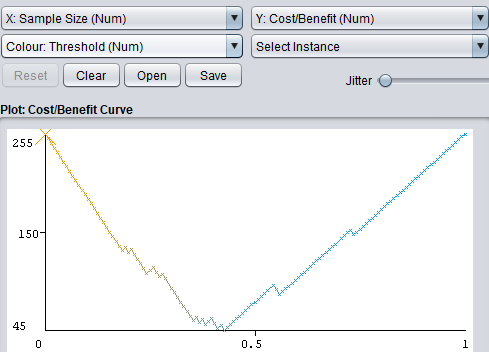
**(b)**



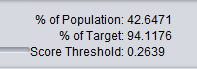
* 在Result list的紀錄終點擊右，找到Cost/Benefit analysis並依照題目要求，選擇class = 1



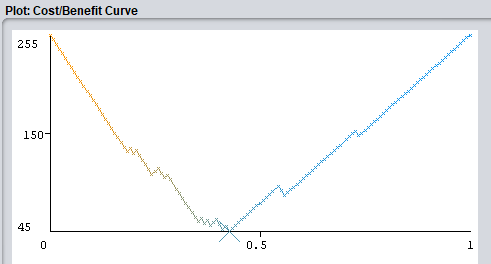
* 依照題目要求，設Cost Matrix如上圖
* 因題目要求要找最佳的sample size rate，在此先定義何謂最佳的sample size Rate。
* 根據Cost/Benefit analysis評分方式，「Gain」越高就代表越好，因此需要調整sample size rate找到Gain最高的數值



* 可以把Y軸設為Cost/Benefit，找cost較低的部分，幫助尋找gain的最大值。

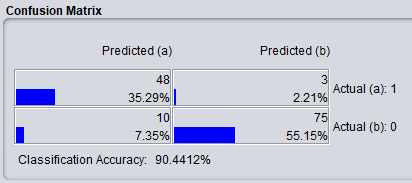
 

* 在sample size rate = 42.6471%時找到最大的Gain = 210
* 因此42.6471%為最佳的sample size rate



* 此時的確在Cost的低點，證明剛剛的分析是有用的

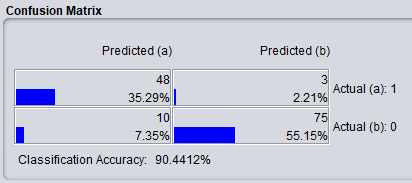
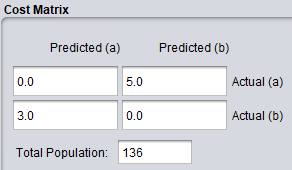
**(c)**



* Confusion Matrix如上圖所示



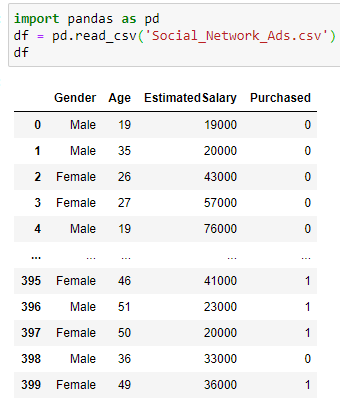
* Cost = 45



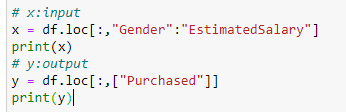
* 必須配合這兩張圖來驗證Cost，將兩個Matrix相乘即可得到Cost
* Cost = 0\*48 + 5\*3 + 3\*10 + 0\*75 = 15 + 30 = 45

**Python Part:**

前置處理:



* 先讀取資料，並觀看資料格式

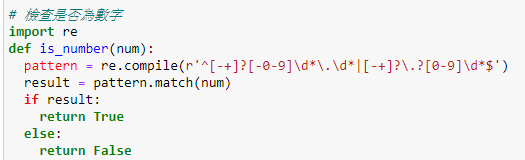


* 將資料分成input、output

**(d)**



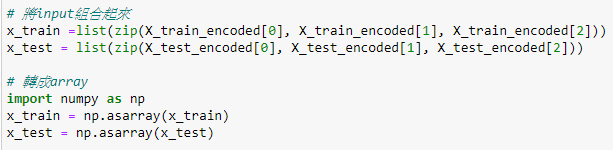
* 依照題目要求，設test\_size = 0.33，random\_state = 1，把資料切割成training、testing data
* 若是直接fit進model訓練會產生以下錯誤
* 
* 因為其中有nominal的值無法處理，因此要做預處理



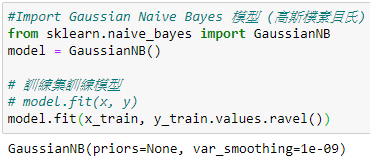
* 等等要處理的只有nominal的部分，因此先寫個函數來判斷是否為數字



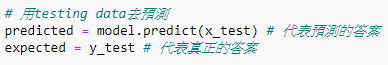
* 對每一個欄位的第一個值進行判斷是否為數字。在此的方法為先把第一個欄位的值變成String，再用我寫的函數去判斷。
* 不是數字 : 用LabelEncoder()編碼後放入list中
* 是數字 : 直接放入list中



* 因為要放進model中訓練時input要變成一個一個的數組，因此用zip把每個數值組合起來，並且轉換成array來當作input。



* 建立模型，並把training data fit進去訓練

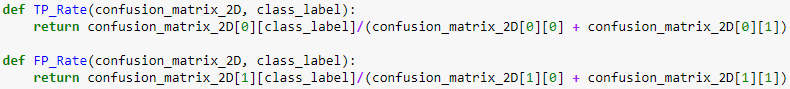


* 使用predict()函數，把testing data的x\_test放進去進行預測，並把實際答案y\_test放進expected變數中。

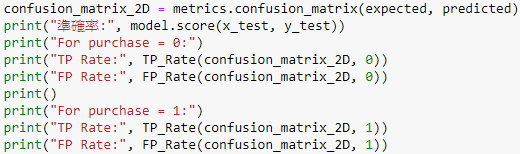




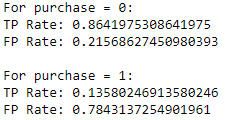
* 使用內建的score()函數找出model的準確率



* 因為題目只有說找TP Rate / FP Rate，並未提及是針對哪一個Class Label做處理，因此我寫了一個函數，參數為Confusion Matrix和Class Label，讓使用者可以自己決定要針對特定的class計算。



* 先用內建函數confusion\_matrix()取得matrix，再用自定義函數去計算purchase = 0 和 purchase = 1時分別的TP Rate / FP Rate



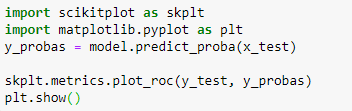
* 結果如圖所示



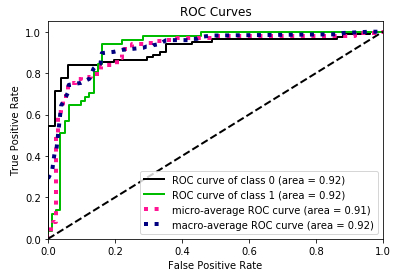


* 為求保險，我把實際的Confusion Matrix印出來算一次
* For purchase = 0
* TP Rate = 70 / (70+11) = 0.8641975308641975
* FP Rate = 11 / (11+40) = 0.21568627450980393
* For purchase = 1
* TP Rate = 11 / (70+11) = 0.13580246913580246
* FP Rate = 40 / (11+40) = 0.7843137254901961

**(e)**

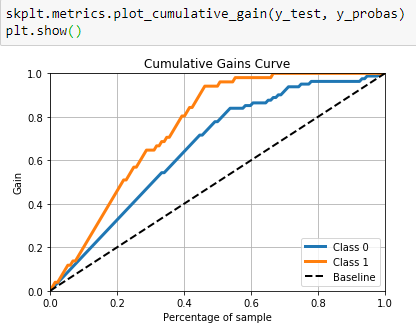
****

* 使用scikitplot中的plot\_roc()來進行繪製，依照文檔需求，第一個參數為y\_test(y\_label的答案)、第二個參數為y\_probas，代表每個x\_test估計出來的機率所形成的vector。
* y\_probas可以使用內建函數predict\_proba()來取得

****

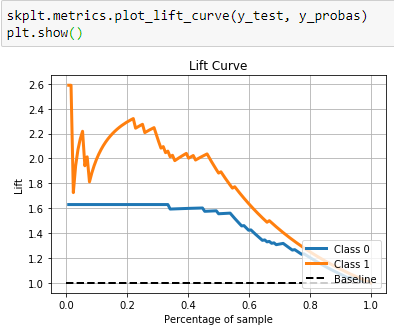
* 結果如上圖所示
* AUC部分可分為4種答案，依照上課所學主要關注純粹ROC curve的部分
* Class(purchase) = 0時 : AUC = 0.92
* Class(purchase) = 1時 : AUC = 0.92
* Micro/Macro average : AUC = 0.91 和 0.92

**(f)**



* 使用內建函數plot\_cumulative\_gain()，繪製出lift chart。在此Y軸雖說標示為Gain，但其實它就是TP Rate。

**(g)**



* 使用內建函數plot\_lift\_curve()，繪製出lift curve