HW6 Cost-Benefit Analysis

Weka 部分

使用 weka 對 Social_Network_Ads.csv 進行 naive bayes 分析，選擇 percentage split 66% ，過程中對所有重要步驟進行截圖並加以說明，越詳盡好：

- (a) 使用 cost-sensitive-learning，將 cost matrix 設定如下圖，列出 total cost 及 average cost，截圖並詳細說明該數字是如何計算出來的 (20%)

	Prediced (a)	Prediced (b)
Actual (a)	0.0	5.0
Actual (b)	3.0	0.0

- (b) 對購買者（purchase = 1）進行 cost/benefit analysis，cost matrix 一樣設定如下圖，說明最佳的 sample size rate 是多少 ?截圖並詳細說明 (20%)
- (c) 承上題，在最佳的 sample size rate 情況下，混淆矩陣長怎樣 ? cost 為多少 ?截圖並詳細說明該數字是如何計算出來的 (10%)

(a) 使用 cost-sensitive-learning，將 cost matrix 設定如下圖，列出 total cost 及 average cost，截圖並詳細說明該數字是如何計算出來的 (20%)

	Prediced (a)	Prediced (b)
Actual (a)	0.0	5.0
Actual (b)	3.0	0.0

操作步驟如下:

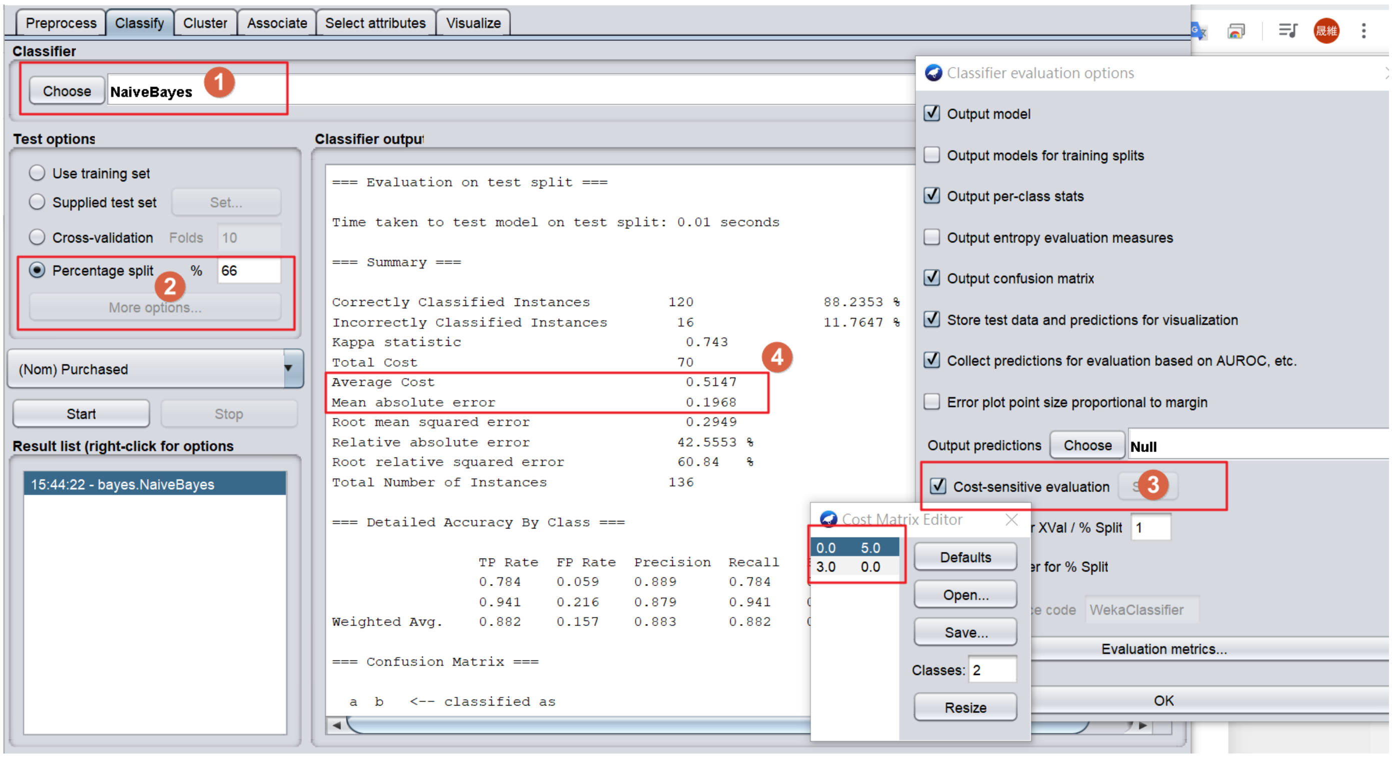
1. 選擇 naive bayes 分析
2. 選擇資料 percentage split 66% > 將 200 筆原始資料分成 264 筆訓練資料和 136 筆測試資料。
3. More options > 選擇 cost-sensitive evaluation > 填上如上表的 cost 表格
4. 執行後可以看到 Total Cost 和 Average Cost:

```
=== Summary ===
Total Cost                                70
Average Cost                               0.5147

=== Confusion Matrix ===
  a  b   <-- classified as
 40 11 |  a = 1
  5 80 |  b = 0
```

5. 算法:

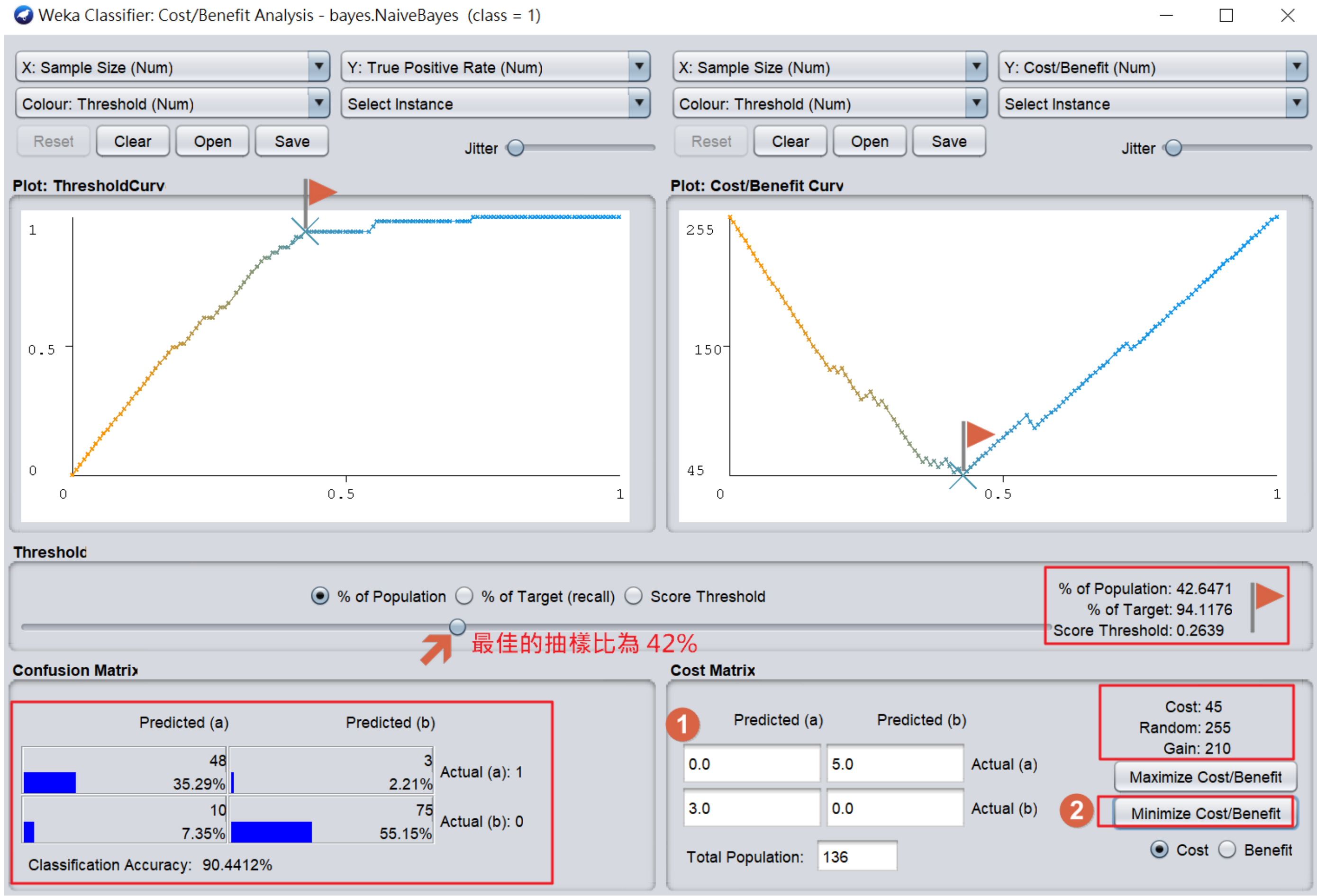
- `Total Cost = Confusion Matrix 內積 Cost Matrix = 0*40 + 5*11 + 3*5 + 0*80 = 70`
- `Average Cost = Total Cost / Count of Test data = 70/136 = 0.5147`



(b) 對購買者（purchase = 1）進行 cost/benefit analysis，cost matrix 一樣設定如下圖，說明最佳的 sample size rate 是多少 ?截圖並詳細說明 (20%)

操作:

- 對結果右鍵 cost/benefit analysis > purchase = 1
- 填好 Cost 矩陣
- 按下 Minimize Cost
- 結果顯示最佳抽樣比為 Sample size rate = 42%，在此抽樣比率之下，有蒐集到 94% 的潛在顧客 (會購買的)
- 算法: 94% of target = Recall = TP rate = 48/48+3 (利用左下的混淆矩陣)
- Lift chart 顯示在最佳抽樣比 42% 之下，lift = 2.2；Threshold = 0.2639 表示機率為 0.2639 以上的顧客都是要行銷的對象。



(c) 承上題，在最佳的 sample size rate 情況下，混淆矩陣長怎樣 ？cost 為多少 ？截圖並詳細說明該數字是如何計算出來的 (10%)

上圖左架角的即為在最佳的 sample size rate (42%)下的 Confusion Matrix，此時 Cost 為 45。

	Prediced (a)	Prediced (b)
Actual (a)	48	3
Actual (b)	10	75

Cost 的算法:

- Cost = Confusion Matrix 內積 Cost Matrix = `10*3 + 3*5 = 45`
- Random = 全部抽樣的 Cost = `85 * 3 = 255`
- Gain = Random - Cost

Python 部分

使用 python 對 Social_Network_Ads.csv 進行 naive bayes 分析，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：

- (d) 設test_size = 0.33，random_state = 1，進行 naive bayes 分析後，列出準確率及TP Rate/FP Rate (10%)
- (e) 繪出 ROC Curve 並計算出 AUC (20%)
- (f) 繪出 lift chart (又稱Cumulative Gain Chart) (X軸: sample size rate; Y軸: TP rate) (10%)
- (g) 繪出 lift curve (X軸: sample size rate; Y軸: Lift) (10%)

PS: 可以直接看 .ipynb 檔，因為以下是直接截圖貼上的

(d) 設test_size = 0.33，random_state = 1，進行 naive bayes 分析後，列出準確率及TP Rate/FP Rate (10%)

讀入 csv 並把 nom 型態轉成數值型態，因 naive bayes 只接受數值資料

In [1]:
```
import pandas as pd

df = pd.read_csv('Social_Network_Ads.csv')
df.head(5)
```

Out[1]:

	Gender	Age	EstimatedSalary	Purchased
0	Male	19	19000	0
1	Male	35	20000	0
2	Female	26	43000	0
3	Female	27	57000	0
4	Male	19	76000	0

Map string to int

[(筆記) pandas的map、apply、applymap](#)

In [2]:
```
gender_mapping = {'Male': 1, 'Female': 0}
ser_map = df['Gender'].map(gender_mapping)
df['Gender'] = ser_map

df.head(5)
```

Out[2]:

	Gender	Age	EstimatedSalary	Purchased
0	1	19	19000	0
1	1	35	20000	0
2	0	26	43000	0
3	0	27	57000	0
4	1	19	76000	0

切分 train(0.67) test(0.33) 資料再訓練模型

In [3]:
```
features = df[['Gender', 'Age', 'EstimatedSalary']]
label = df['Purchased']
print('Total records:', len(df))
```

```
Total records: 400
```

In [4]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, label, test_size=0.33, random_state=1)

print('Train Set Size:', len(X_train))
print('Test Set Size:', len(X_test))
```

```
Train Set Size: 268
Test Set Size: 132
```

In [5]:
```
#Import Gaussian Naive Bayes 模型 (高斯樸素貝氏)
from sklearn.naive_bayes import GaussianNB

model = GaussianNB()
model.fit(X_train, y_train)
```

Out[5]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

Traing 的準確度

Training Precision

In [6]:
```
model.score(X_train, y_train)
```
Out[6]:
```
0.8955223880597015
```

In [7]:
```
from sklearn import metrics

y_train_predict = model.predict(X_train)
print(metrics.classification_report(y_train, y_train_predict))
```
```
              precision    recall  f1-score   support

           0       0.90      0.95      0.92       176
           1       0.89      0.79      0.84        92

    accuracy                           0.90       268
   macro avg       0.89      0.87      0.88       268
weighted avg       0.90      0.90      0.89       268
```

Testing 的準確度和 TP Rate/FP Rate

Testing Precision

In [8]:
```
model.score(X_test, y_test)
```
Out[8]:
```
0.8333333333333334
```

In [9]:
```
expected = y_test
predicted = model.predict(X_test)
print(metrics.classification_report(expected, predicted))
```
```
              precision    recall  f1-score   support

           0       0.86      0.86      0.86        81
           1       0.78      0.78      0.78        51

    accuracy                           0.83       132
   macro avg       0.82      0.82      0.82       132
weighted avg       0.83      0.83      0.83       132
```

Confusion matrix & TP/FP rate of Testing

In [25]:
```
cm = metrics.confusion_matrix(expected, predicted)
print(cm)
```
```
[[70 11]
 [11 40]]
```

In [26]:
```
tp = cm[0, 0]
fp = cm[1, 0]
tn = cm[1, 1]
fn = cm[0, 1]
tp_rate = tp/(tp+fn)
fp_rate = fp/(fp+tn)
print("TP rate:", tp_rate)
print("FP rate:", fp_rate)
```
```
TP rate: 0.8641975308641975
FP rate: 0.21568627450980393
```

(e) 繪出 ROC Curve 並計算出 AUC (20%)

ROC Curve

繪出 ROC Curve 並計算出 AUC = `0.82` (可是圖上寫 `0.92`)

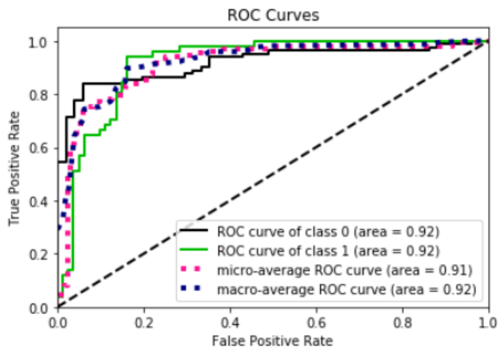
- Receiver Operating Characteristic (ROC)
- Metrics Module (API Reference) 所有的圖都在這兒

```
$ pip install scikit-plot
```

In [22]:
```
import scikitplot as skplt

y_probas = model.predict_proba(X_test)
skplt.metrics.plot_roc(y_test, y_probas)
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x1a6635351c8>



In [21]:
```
sklearn.metrics.roc_auc_score(y_test, predicted)
```

Out[21]: 0.8242556281771968

(f) 繪出 lift chart (又稱Cumulative Gain Chart) (X軸: sample size rate; Y軸: TP rate) (10%)

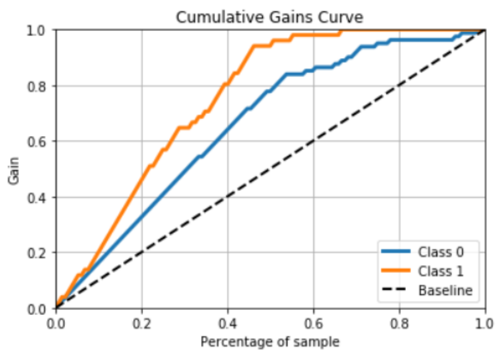
Lift Chart

繪出 lift chart (又稱Cumulative Gain Chart)

- X軸: sample size rate
- Y軸: TP rate

In [12]:
```
skplt.metrics.plot_cumulative_gain(y_test, y_probas)
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1a66324c788>



(g) 繪出 lift curve (X軸: sample size rate; Y軸: Lift) (10%)

Lift Curve

繪出 lift curve

- X軸: sample size rate
- Y軸: Lift

In [13]:
```
skplt.metrics.plot_lift_curve(y_test, y_probas)
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1a6632cb408>