Batch evaluation may be carried out using the “classifier performance” tool (p.358 manual), which can be accessed through the EM wizard by clicking on a data cell.

The forum has a few entries on this, namely:

1. https://forum.hugin.com/index.php/topic,148.msg309/topicseen.html#msg309 (Testing BBN model).
2. https://forum.hugin.com/index.php/topic,193.msg452/topicseen.html#msg452 (Analysis Wizard : Predicted value?)
3. https://forum.hugin.com/index.php?action=profile;area=showposts;u=3 (How to classify data).

Here is an example with “Chest Clinic.”

I will use Bronchitis as the class variable.

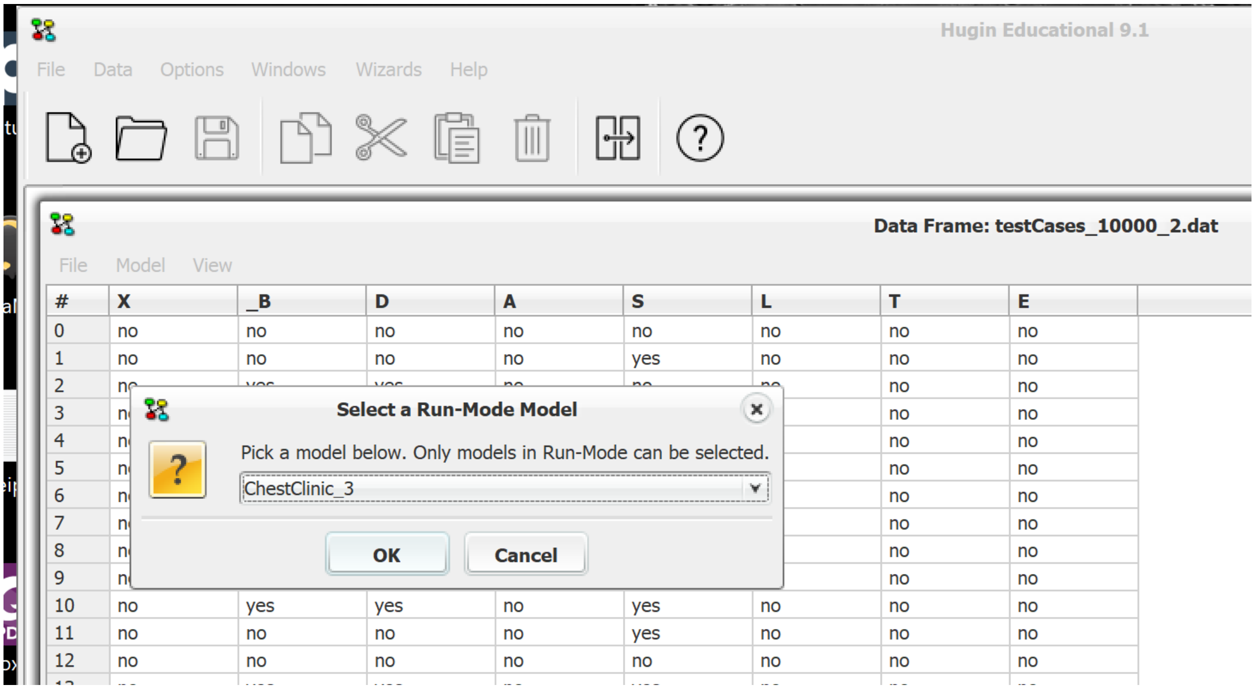
I started by generating a test case file (testCases_10000.dat) by simulation (in Run Mode: File->Cases->Simulate Cases): 10000 cases, no missing values, MCAR. Since the cases are generated from the model itself, accuracy should be very good.

I then follow the steps in (3) above:

> You need to perform a number of steps to pre-compute probabilities before doing classification.
>
> 1) load the network (and switch to run-mode).
> 2) load data
> 3) select your network as the 'run mode model'
> 4) add colum(s) for collecting beliefs
> 5) process entire data file in batch - this yields numbers in the colums configured in step 4.
> 6) start the classification
>
> for step 1-4 follow the description in the 'Data Frame' manual page
> http://download.hugin.com/webdocs/manuals/Htmlhelp/descr_dataframe.html
> (current version:
> https://download.hugin.com/webdocs/manuals/GUI/pages/Manual/UsingData/DataFrame.html?highlight=data%20frame).
>
> and lastly follow the steps described in the 'Data classifier performance' manual page
> http://download.hugin.com/webdocs/manuals/Htmlhelp/descr_data_classifier_performance.html

After loading data, I renamed the column of the class (target) variable by prepending an underscore, as suggested in the manual.

I select the Run-Mode model from the Data Frame, as shown below:



#	X	_B	D	A	S	L	T	E
0	no	no	no	no	no	no	no	no
1	no	no	no	no	yes	no	no	no
2	no	yes	yes	no	no	no	no	no
3	n						no	no
4	n						no	no
5	n						no	no
6	n						no	no
7	n						no	no
8	n						no	no
9	n						no	no
10	no	yes	yes	no	yes	no	no	no
11	no	no	no	no	yes	no	no	no
12	no	no	no	no	no	no	no	no

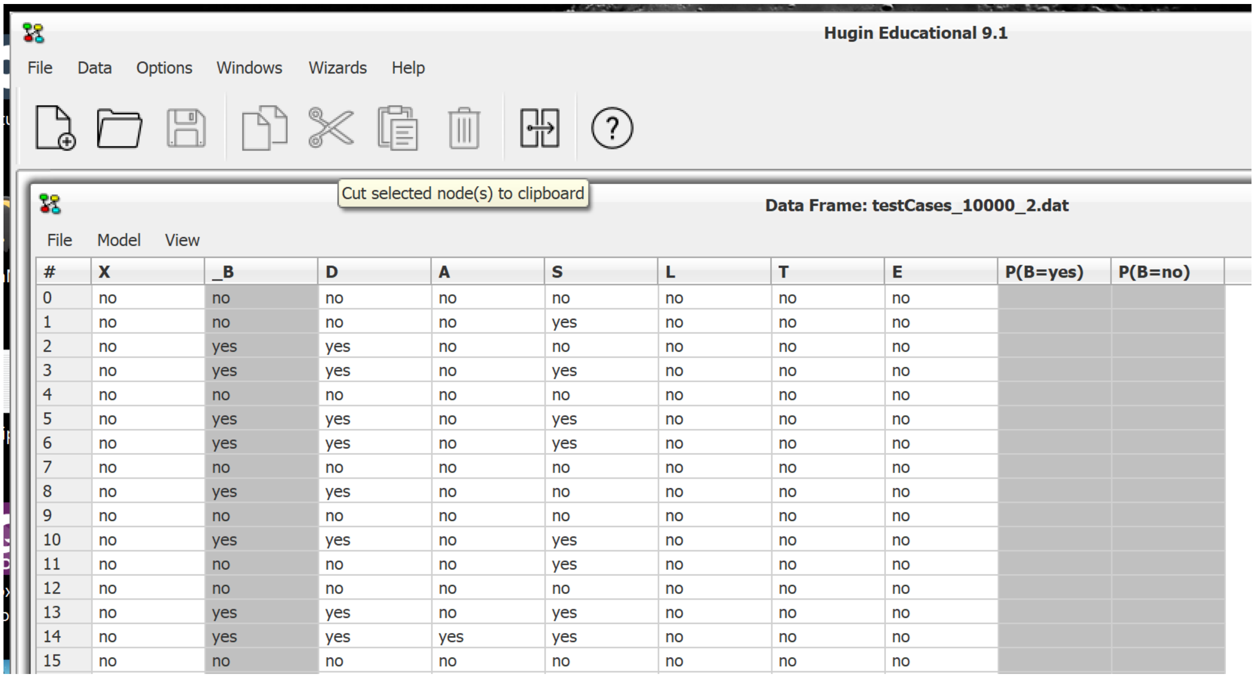
Select a Run-Mode Model

Pick a model below. Only models in Run-Mode can be selected.

ChestClinic_3

OK Cancel

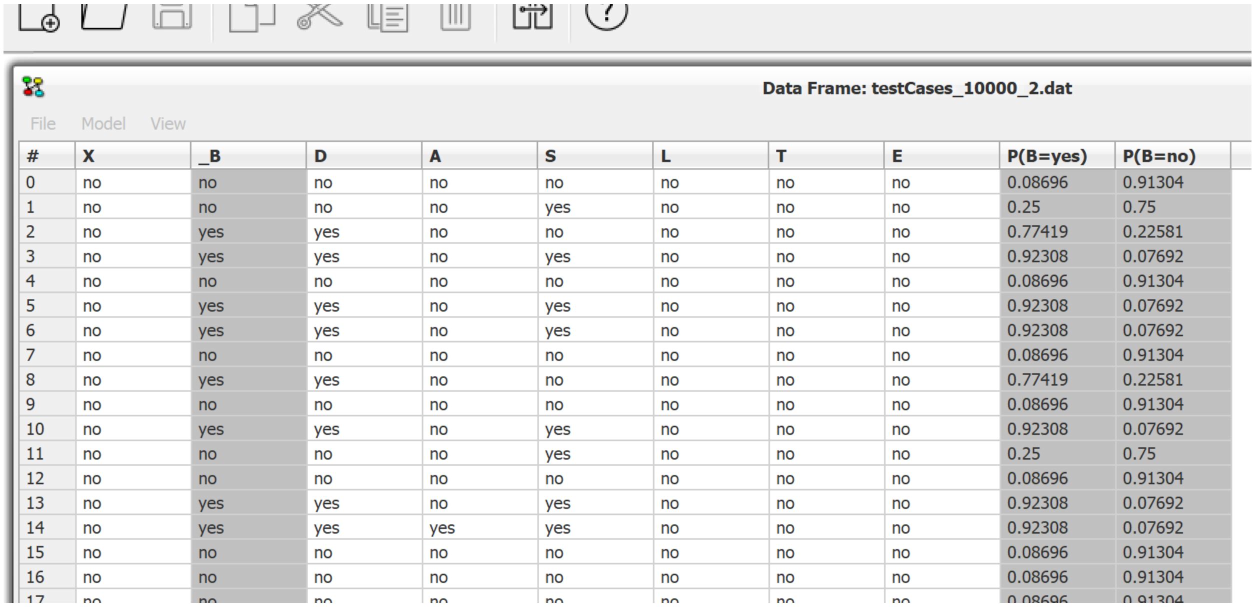
I pick B as the monitor note, with two states: yes and no. New columns show up in the data frame, as the manual says at https://download.hugin.com/webdocs/manuals/GUI/pages/Manual/UsingData/DataFrame.html?highlight=data%20frame:



#	X	_B	D	A	S	L	T	E	P(B=yes)	P(B=no)
0	no	no	no	no	no	no	no	no		
1	no	no	no	no	yes	no	no	no		
2	no	yes	yes	no	no	no	no	no		
3	no	yes	yes	no	yes	no	no	no		
4	no	no	no	no	no	no	no	no		
5	no	yes	yes	no	yes	no	no	no		
6	no	yes	yes	no	yes	no	no	no		
7	no	no	no	no	no	no	no	no		
8	no	yes	yes	no	no	no	no	no		
9	no	no	no	no	no	no	no	no		
10	no	yes	yes	no	yes	no	no	no		
11	no	no	no	no	yes	no	no	no		
12	no	no	no	no	no	no	no	no		
13	no	yes	yes	no	yes	no	no	no		
14	no	yes	yes	yes	yes	no	no	no		
15	no	no	no	no	no	no	no	no		

I do batch propagation for all cases. To do this, right click on any row header and select propagate all cases.

This populates the probabilities of P(B=”yes”) and P(B=”no”), as below:

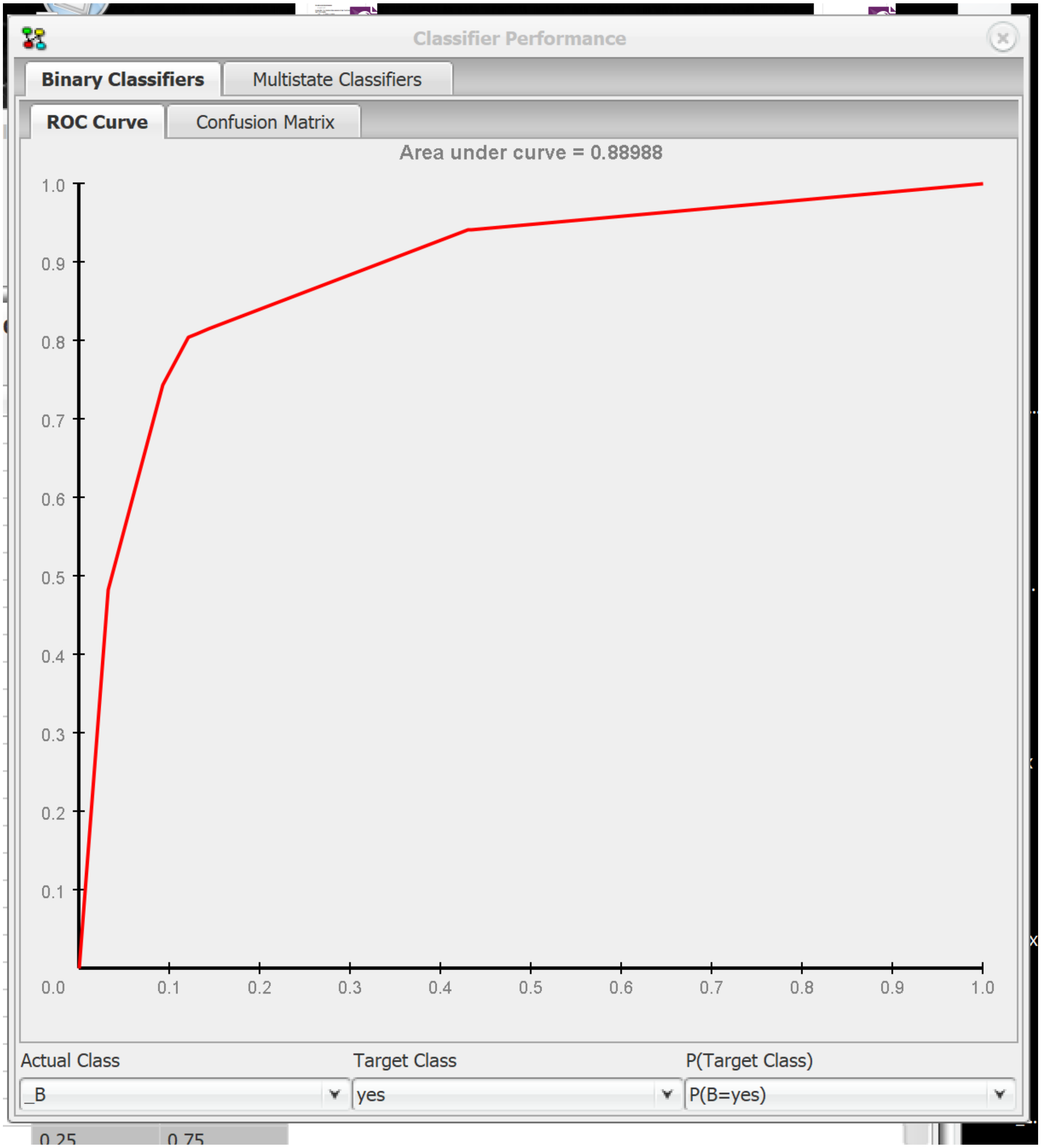


Data Frame: testCases_10000_2.dat

#	X	_B	D	A	S	L	T	E	P(B=yes)	P(B=no)
0	no	no	no	no	no	no	no	no	0.08696	0.91304
1	no	no	no	no	yes	no	no	no	0.25	0.75
2	no	yes	yes	no	no	no	no	no	0.77419	0.22581
3	no	yes	yes	no	yes	no	no	no	0.92308	0.07692
4	no	no	no	no	no	no	no	no	0.08696	0.91304
5	no	yes	yes	no	yes	no	no	no	0.92308	0.07692
6	no	yes	yes	no	yes	no	no	no	0.92308	0.07692
7	no	no	no	no	no	no	no	no	0.08696	0.91304
8	no	yes	yes	no	no	no	no	no	0.77419	0.22581
9	no	no	no	no	no	no	no	no	0.08696	0.91304
10	no	yes	yes	no	yes	no	no	no	0.92308	0.07692
11	no	no	no	no	yes	no	no	no	0.25	0.75
12	no	no	no	no	no	no	no	no	0.08696	0.91304
13	no	yes	yes	no	yes	no	no	no	0.92308	0.07692
14	no	yes	yes	yes	yes	no	no	no	0.92308	0.07692
15	no	no	no	no	no	no	no	no	0.08696	0.91304
16	no	no	no	no	no	no	no	no	0.08696	0.91304
17	no	no	no	no	no	no	no	no	0.08696	0.91304

I run “Classifier Performance” (by right clicking on one of the columns with numbers).

I choose _B as actual class, yes as target class, P(B=yes) as P(Target Class). I get the following ROC curve:



Area under curve = 0.88988

Actual Class: _B

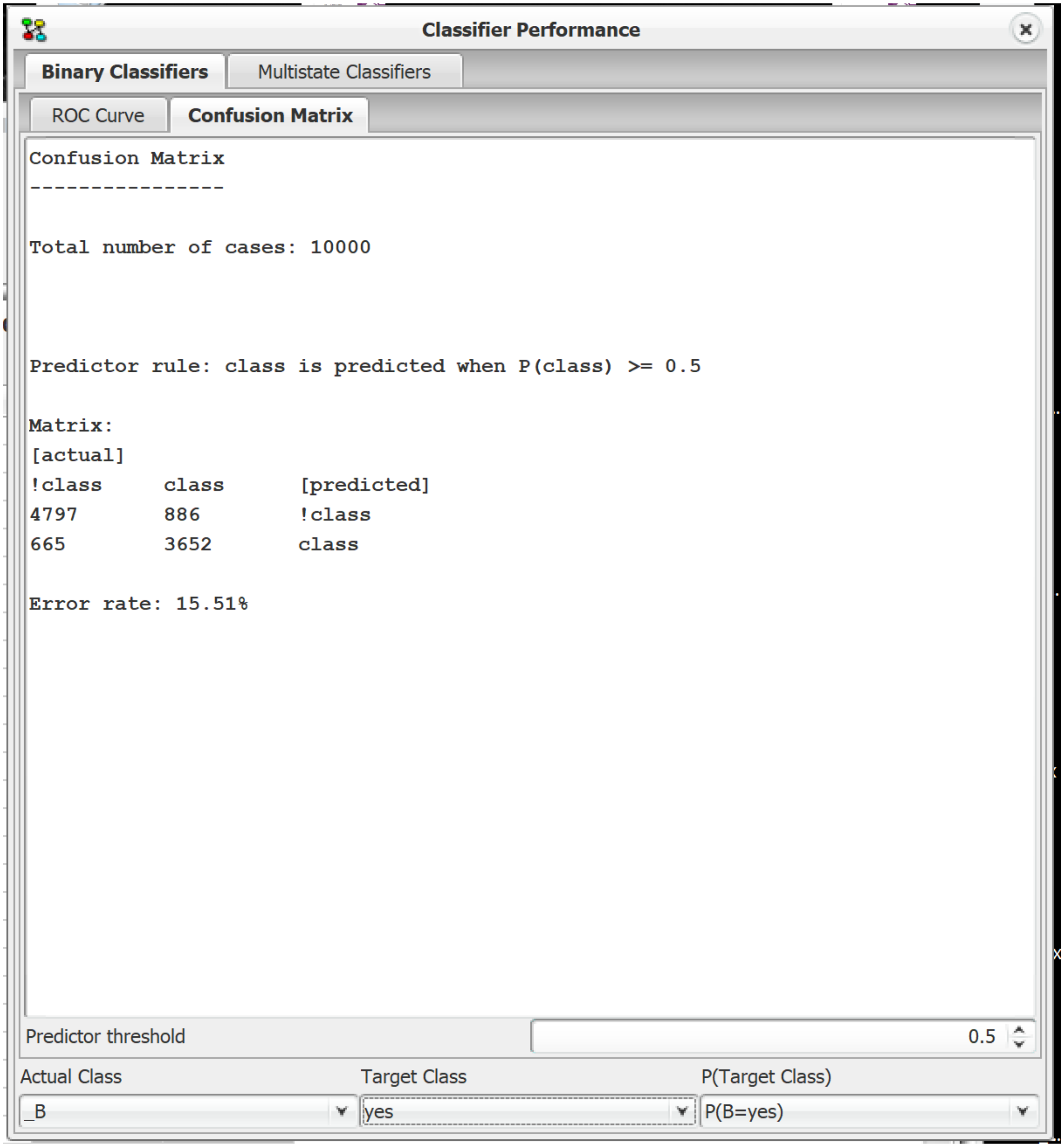
Target Class: yes

P(Target Class): P(B=yes)

The ROC curve shows how one can trade off true positive rate (in the Y axis, also known as recall or sensitivity: number of true positives / (number of true positives + number of false positives)) and false positive rate (in the X axis: number of false positives / (number of true negatives + number of false positives)); this is also 1 – specificity, where specificity is number of true negatives / (number of true negatives + number of false positives)).

The result is quite good.

I can also get the confusion matrix:



```
Confusion Matrix
----------------

Total number of cases: 10000



Predictor rule: class is predicted when P(class) >= 0.5

Matrix:
[actual]
!class     class      [predicted]
4797       886        !class
665        3652       class

Error rate: 15.51%
```

Predictor threshold: 0.5

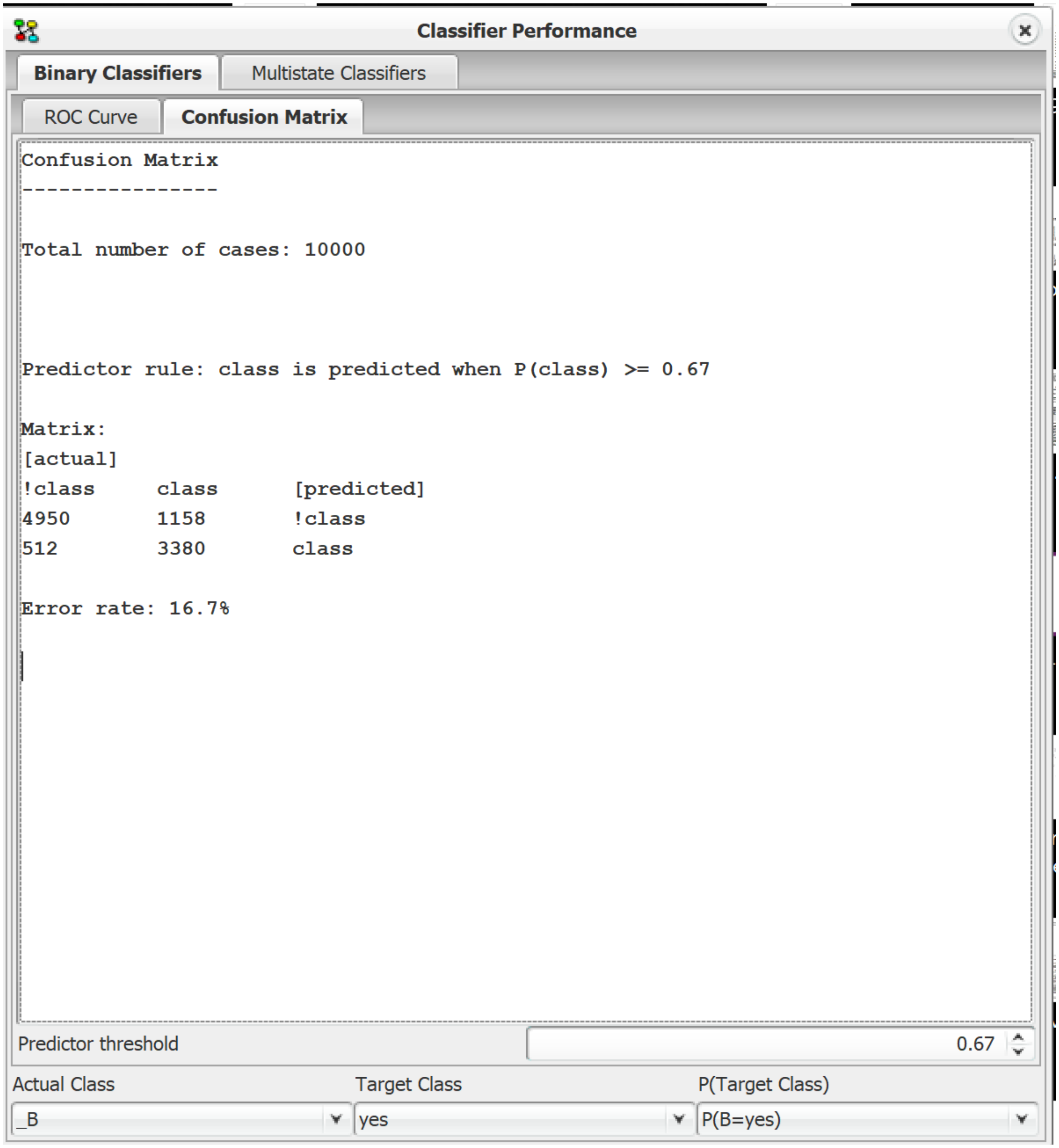
Actual Class: _B

Target Class: yes

P(Target Class): P(B=yes)

Again, the result is good.

Changing the threshold to a higher value increases the number of false negatives and decreases the number of false positives:



```
Confusion Matrix
----------------


Total number of cases: 10000



Predictor rule: class is predicted when P(class) >= 0.67

Matrix:
[actual]
!class     class      [predicted]
4950       1158       !class
512        3380       class

Error rate: 16.7%
```

Predictor threshold 0.67

Actual Class: _B

Target Class: yes

P(Target Class): P(B=yes)

The ROC curve is built by varying the threshold from 0-1. For a very low threshold, there is very high recall (=TPR=sensitivity), but low specificity (equivalently, high false positive rate). For a very high threshold, there is very high specificity (equivalently, low false positive rate) but low recall (=TPR=sensitivity).