# Weka – VotedPerceptron & Attribute Transformation (1)

**Lab6 (in-class):** 5 DIC 2016 -13:15-15:00 (CHOMSKY)

ACKNOWLEDGEMENTS: INFORMATION, EXAMPLES AND TASKS IN THIS LAB COME FROM SEVERAL WEB SOURCES.

## Learning objectives

In this lab you are going to:

- use the perceptron: weka implementation: *VotedPerceptron*
- transforming attributes: *NumericToBinary*, *StringToWordVector*
- use a metaclassifier: *FilteredClassifier*

### Task 1: Warming-up: An *ad interim* summary [max time 10 min]

In our previous lab, we saw that J48, IBk and NaiveBayes have good (J48 and IBk) or at least sort of decent performance (NaiveBayes) on the junk dataset. Some of the different behaviors are statistically significant (see Appendix to this document). However, it remains hard to decide which classifier is the "best" classifier. "Best" according to which evaluation measure? Your disappointed faces showed that this was a quite hard dilemma ☺

We learned that Accuracy can be a tricky metric, and it is often unstable and non-representative, if not confirmed by other metrics. K statistics is a kind of accuracy corrected by chance removal. IR metrics (P/R/F) give us a better idea of the counts of TP/TN/FP/FN. ROC curves are thresholds that depict the performance of classifiers regardless their class distribution: ROC curves plot the true positive rate (hit rate) on the vertical axis against the false positive rate (false alarms) on the horizontal axis, which means that if the shape of the curves is closer to the y axis and approaches the top left corner, the performance of a classifier is robust.

Table1 (see Task 2) shows the performance of our classifiers on the junk dataset. Namely, *Acc, k stat, averaged P/R/F and averaged AUC* tell us that J48 and IBk have consistent behavior, although the small numerical variations that we can observe betw the 2 classifier are statistically significant (J48 appears to be more robust than IBk on this dataset).
The interpretation of the performance of the NaiveBayes is more problematic: *Acc, k stat, averaged P/R/F* are just decent, but averaged AUC is higher wrt previous classifiers with an outstanding 0.96 of the NaiveBayes –k. (*NaiveBayes Multinomial* seems to perform worse than both NaïveBayes and NaiveBayes –k, see Table 1). One possible interpretation of these results is that traditional metrics (ie. Acc, k stat, averaged P/R/F) show that this implementation of NB is less robust because of its class-conditional independence inductive bias that might not be ideal for this learning problem. But since ROC curves are not so much influenced by class distribution, NB –k performs better according to this metric. If we compare the 4 curves on the junk class, we see only small

differences. The curves look good (**but see discussion in the appendix**) since they are all placed very close to the upper left corner.

There is no ultimate answer[1] on the best metric: it depends on the purpose of the classification. Before making any final decision think of the inductive biases underlying the mathematical models and compute the cost/benefit analysis (**read Appendix very carefully**). Then, whatever decision you make, motivate your choice with strong arguments ☺. In the academic world, especially for publications, always compute the metrics that are used in your domain (to allow comparisons with other researchers' results), then suggest new metrics and new interpretations if you think traditional metrics are not adequate to explain a classifier's behavior.

## Task 2: Voted Perceptron [max time 10 min]

Today we are going to use a weka implementation of the perceptron called "VotedPerceptron" (see Daume', Section 3.6).

**Att.:** J48, IBK and Naïve Bayes can deal with both nominal and numeric data (read the tooltips when you hover on the classifier's name).
Linear classifiers can be pickier. Check always how the classifiers have been implemented. If you have problems with data types, apply the transformations that you have learned in Lecture 6. In this task we will see some of the advantages of transforming attributes.

- Upload the junk dataset
- Go to the Classify tab
- Select function→VotedPerceptron (there is no simple Perceptron implementation in Weka).

If you read the tooltip, you will see that this implementation can handle nominal, numeric, and binary attributes (but theoretically, remember what Joakim said in the lecture). Good for us ! However this implementation can only handle **binary** classes. This means that if we try to apply this classifier to the iris dataset, that has 3 classes, it will not work and it will appear greyed out.

Q1: Fill up the table below with the required values. How does VotedPerceptron perform on this dataset?

Change the iteration parameter to 100. Run the classifier. Write down the performance.
Change the iteration parameter to 1000. Run the classifier. Write down performance.

Q2: How does the performance vary in relation to the number of iterations?

[1] Read this post to get a realistic picture of the "best performance dilemma":
< http://stats.stackexchange.com/questions/68893/area-under-curve-of-roc-vs-overall-accuracy >

| Junk dataset < http://stp.lingfil.uu.se/~santinim/ml/2016/Datasets/junk.arff > | | | | | | |
|---|---|---|---|---|---|---|
| **Classifier** | **Acc** | **k-statistc** | **Avg. P** | **Avg. R** | **Avg. F** | **Avg. AUC** |
| J48, default | 92.97 | 0.85 | 0.93 | 0.93 | 0.93 | 0.93 |
| IBk, default | 90.78 | 0.80 | 0.90 | 0.90 | 0.90 | 0.90 |
| NaïveBayes, default | 79.28 | 0.59 | 0.84 | 0.79 | 0.79 | 0.93 |
| NaïveBayes, -k[2] | 76.37 | 0.54 | 0.84 | 0.76 | 0.76 | 0.96 |
| NaiveBayes Multinomial | 79.09 | 0.56 | 0.79 | 0.79 | 0.79 | 0.84 |
| | | | | | | |
| VotedPerceptron, default | 49.03 | 0.12 | 0.75 | 0.49 | 0.40 | 0.58 |
| VotedPerceptron, 100 iterations | | | | | | |
| VotedPerceptron, 1000 iterations | | | | | | |
| | | | | | | |
| VotedPerceptron, default (*binarized dataset*) | 92.93 | 0.85 | 0.92 | 0.92 | 0.92 | 0.92 |
| VotedPerceptron, 100 iterations (*binarized*) | | | | | | |
| VotedPerceptron, 1000 iterations (*binarized*) | | | | | | |
| **Table 1. Evaluation metrics and performance of a number of classifiers on the junk dataset** | | | | | | |

## Task 3: Voted Perceptron, binarized dataset [max time 10 min]

Go to the Preprocess tab.
The junk dateset contains attribute values expressed as numeric data, namely real and integer[3]. The class attribute is nominal.
Choose Filter→unsupervised→attribute→NumericToBinary[4].
You will see that the suffix "binarized" is appended to the name of the binarized attribute.

Go to the Classify tab. Select function→VotedPerceptron again. Run the classifier with default parameters.

Q3: Fill up the table above with the required values. How does VotedPerceptron perform on the binarized dataset?

[2] -k is a parameter that overrides the assumption of the normal distribution. –k corresponds to "*useKernelEstimator* -- Use a kernel estimator for numeric attributes rather than a normal distribution." In statistics, kernel density estimation (KDE) is a non-parametric way to estimate the probability density function of a random variable. Kernel density estimation is a fundamental data smoothing problem where inferences about the population are made, based on a finite data sample (wikipedia). Kernel is a difficult concpet to understand and it has different applications. We do not go into details in this course, but you will probably come across something called "kernel trick" in the future (if you curious, read here: < http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html > ). So stay alert ☺

[3] Remember: An integer is a number without a fractional part. Integers may also be negative. A real number is simply a number with a fractional part
< https://www.eskimo.com/~scs/cclass/mathintro/sx1.html>

[4] NumericToBinary converts all numeric attributes into binary ones: Nonzero values become 1. Weka Filters are listed and described in Witten et al (2011: 432-445).

## Task 4: Data with String Attributes [max time 20 min]

The StringToWordVector filter assumes that the document text is stored in an attribute of type String, ie a nominal attribute without a pre-specified set of values. In the filtered data, this is replaced by a fixed set of numeric attributes, and the class attribute is put at the beginning, as the first attribute. To perform document classification, first create an ARFF file with a string attribute that holds the document's text—declared in the header of the ARFF file using @attribute document string, where document is the name of the attribute. A nominal attribute is also needed to hold the document's classification.

Download and explore the following datasets:

*Reuters Corn*
Training set: < http://stp.lingfil.uu.se/~santinim/ml/2016/Datasets/ReutersCorn-train.arff >
Test set: < http://stp.lingfil.uu.se/~santinim/ml/2016/Datasets/ReutersCorn-test.arff >
*Reuters Grain*
Training set: < http://stp.lingfil.uu.se/~santinim/ml/2016/Datasets/ReutersGrain-train.arff >
Test set: < http://stp.lingfil.uu.se/~santinim/ml/2016/Datasets/ReutersGrain-test.arff >

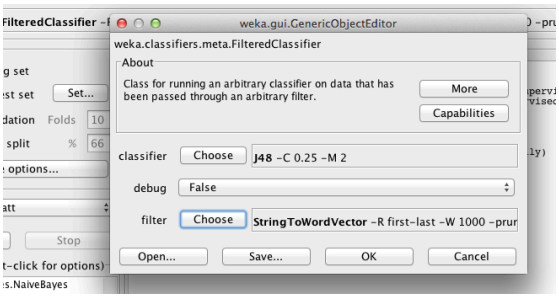
These are standard collections of *newswire articles* that are widely used for evaluating document classifiers. ReutersCorn-train.arff and ReutersGrain-train.arff are training sets derived from this collection; ReutersCorn-test.arff and ReutersGrain-test.arff are corresponding test sets. The actual documents in the corn and grain data are the same; only the labels differ. In the first dataset, articles concerning corn-related issues have a class value of 1 and the others have 0; the aim is to build a classifier that identifies "corny" articles. In the second, the labeling is performed with respect to grain-related issues; the aim is to identify "grainy" articles.

Build classifiers for the two training sets by applying a metalearner [5] called **FilteredClassifier** with StringToWordVector using (1) J48 and (2) NaiveBayesMultinomial, evaluating them on the corresponding test set in each case.

Follow these steps:

- Uplaod the ReutersCorn in the Preprocess tab.
- Go to the Classify tab.
- Choose meta→FilteredClassifier.
- Choose J48 with default parameters as classifier.
- Choose the filter: Unsupervised→attribute→StringToVectors to transform your string data. This filter converts a string attribute to a vector that represents word occurrence frequencies.

[5] A metalearner takes simple classifiers and turn them into more powerful learners. It has more advanced features wrt a "regular" classifier. In our task, if you filter the data in the Preprocess tab, and then try to invoke J8 the normal way, you will see that J48 is greyed out. It means that additional transformations or adjustments would be required. Instead, wehn using the FilteredClassifier metalearner, it is possible to deal with string attributes directly. Read more Witten et al. (2011: 443).




- Click OK.
- In the Classify panel, choose the matching Test set.
- Run the classifier.
- Write down the performance
- Repeat for the Grain dataset. Then apply the Naïve Bayes Multinomial to the dataset.

Fill up the following table:

| | Corn | | | | Grain | | | |
|---|---|---|---|---|---|---|---|---|
| | Acc | k stat | Avg F | AUC | Acc | k stat | Avg F | AUC |
| J48 | | | | | | | | |
| NB Multinomial | | | | | | | | |

Q4: What is the performance in the four scenarios? Based on the results, which classifier would you choose?

Q5: Which of the two classifiers used above produce the best AUC for the two Reuters datasets? Compare this to the outcome for percent correct. What do the different outcomes mean?

## Task 5: ROC Curves and AUC [max time 10 min]

For the Reuters dataset that produced the most extreme difference in Task 4, look at the ROC curves for class 1. Make a very rough estimate of the area under each curve, and explain it in words

# Appendix

## Classifiers Comparison

### Statistical significance

If we take J48 as baseline for statistical significance, we can notice that there are statistically significant differences. For example, if we take *AUC as Comparison field*, we see that NaïveBayes –k performs statistically better than J48. See Fig. 1.

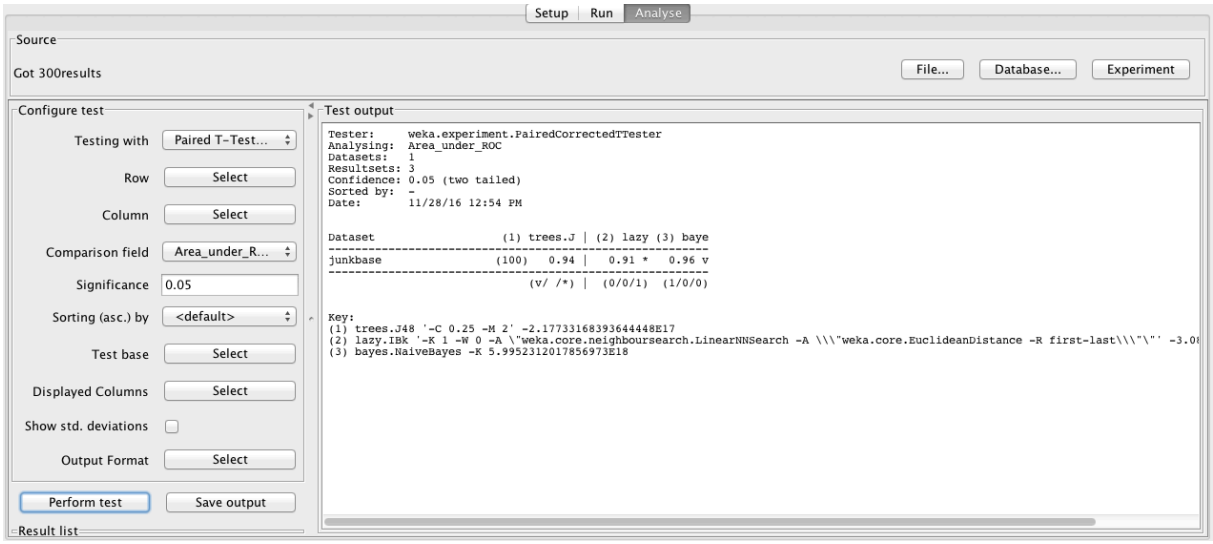



**Fig. 1. Statistical significance of the area under the curve.**

### Comparison of 3 ROC Curves on the junk datast (J48, IBk and NB)

See set up in Fig. 2.

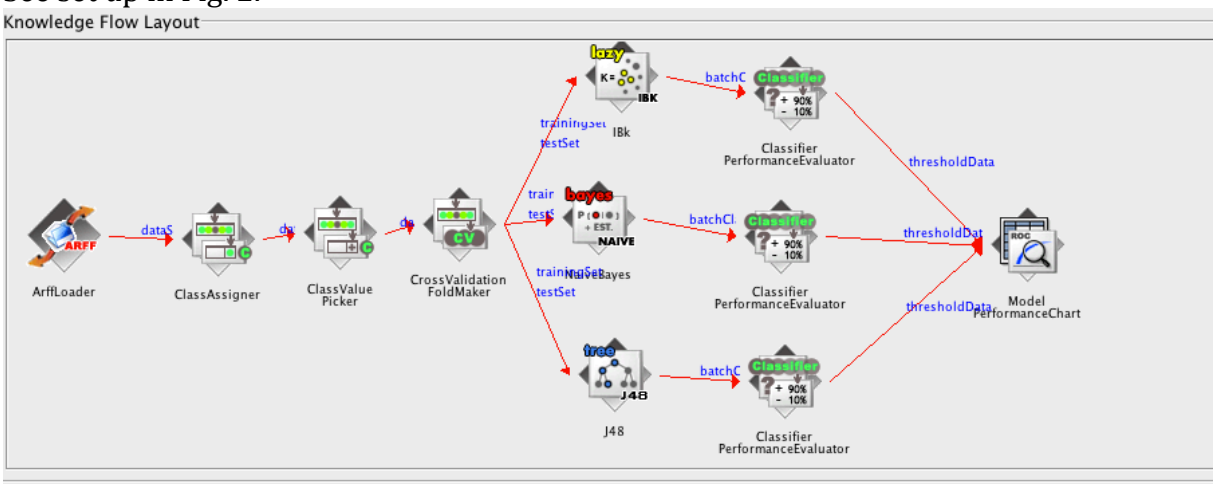



**Fig. 2. Setup to compare 3 ROC curves: J48, IBk and NaiveBayes.**

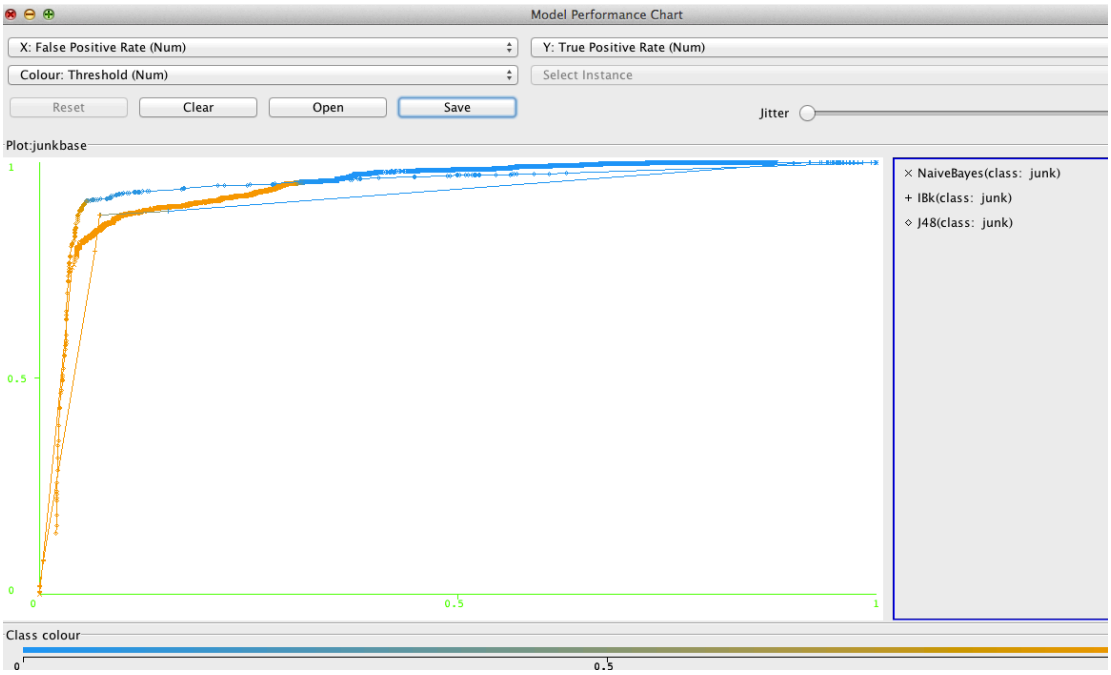



**Fig. 3. Three 3 ROC curves: J48, IBk and NaiveBayes.**

Remember: The color indicated the threshold: blue corresponds to lower threstholds, Each instance in the ROC curve has a threshold value of class (in this case 0.5). If the instance highly belongs to this class, its threshold will be close to 1 (see the color line on the bottom), so it will have red colour, the higher threshold of instance the dark colour it will have in the ROC curve. In Fig. 3, we compare the junk class of J48, IBK and NB. It is hard to visually assess (in this case) the size of the area under the curve, but we can see that the middle curve (J48) has the best curve: longer stretch of orange (so this means values much higher than the threshold 0.5).

If we add Naïve Bayes –k to the setup (see Fig. 4):

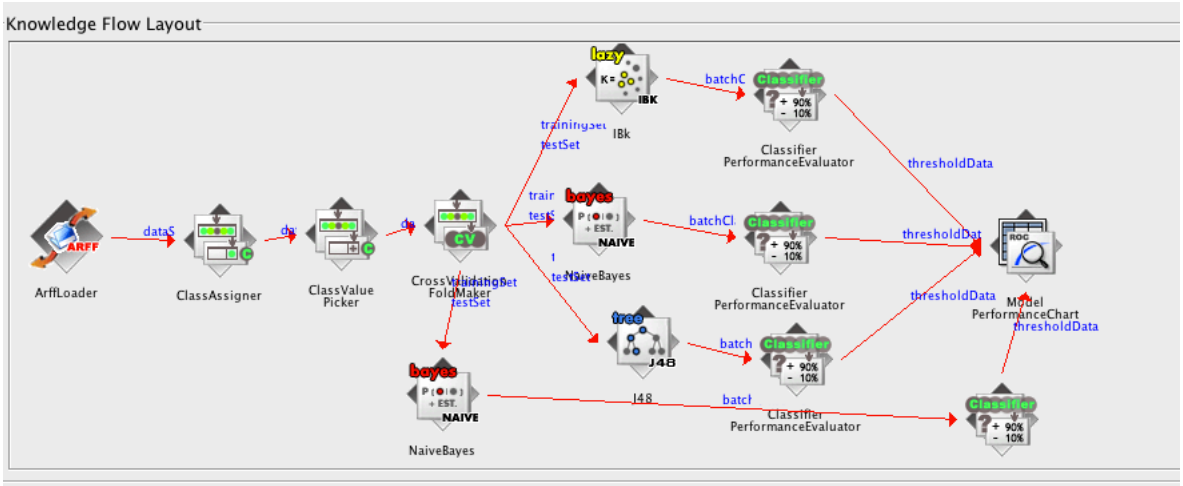



Fig. 4. **Setup to compare 4 ROC curves: J48, IBk, NaiveBayes and NaiveBayes -k.**

We get the following visualization:

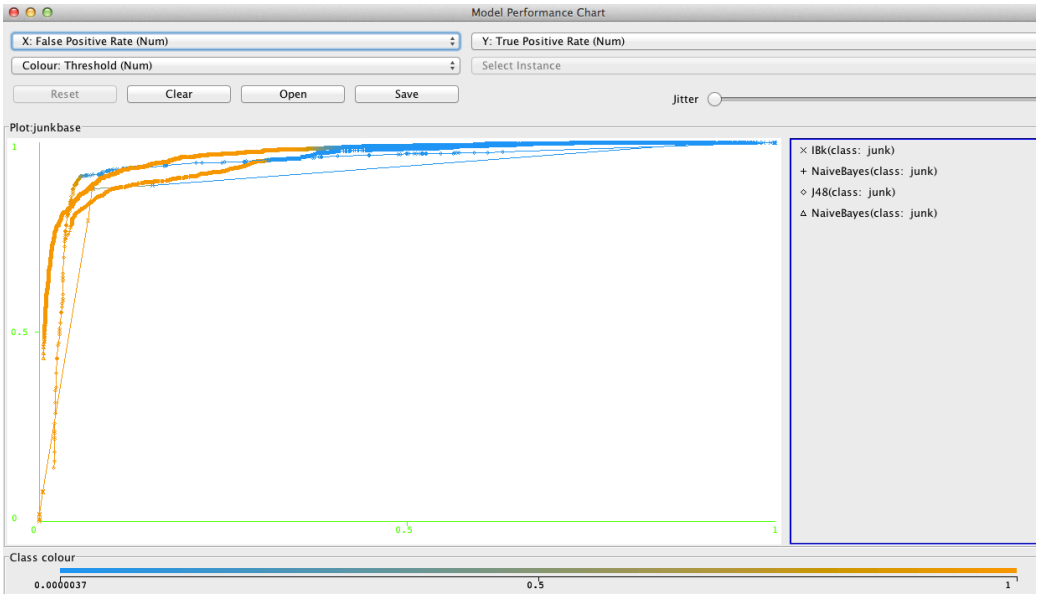



**Fig. 5. Four ROC curves: J48, IBk, NaiveBayes and NaiveBayes -k.**

Now it is clear that Naïve Bayes –k has a better ROC curve, closer to the upper-left corner and with a longer stretch of orange.
Weka uses the Mann Whitney statistic to calculate the AUC[6].

## Cost-Sensitive Evaluation: Cost/Benefit Analysis

Have you run the Cost/Benefit Analysis (cf. also Task 6, Lab4 and Task 5, Lab5) on the junk dataset? Remember the cost is NOT taken into account in the measures that we used in Table 1. Let's do it now.

Let's assume that we think that NaiveBayes –k performs better than the other classifiers in Table 1 because it shows the highest averaged AUC and the best ROC curve. Let's perform a cost/benefit analysis of this classifier.

Run NaiveBayes –k on the junk dataset. Right-click on the Result list and choose Cost/Benefit analysis, *junk* class.

You will see the following window:

[6] Read more here if you are interested (this is not a requirement for this course):
< https://weka.wikispaces.com/ROC+curves >
< https://weka.wikispaces.com/Area+under+the+curve >

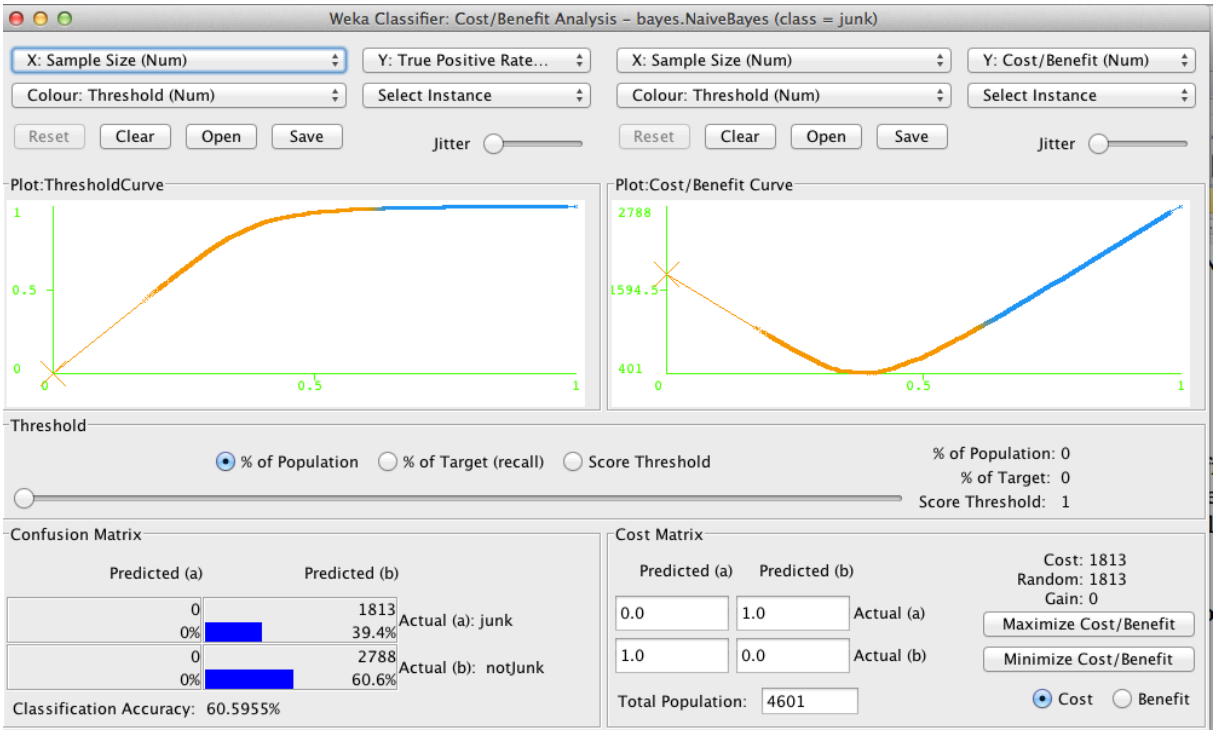



**Fig. 6. This is what you see when you open this window on the junk dataset the first time. Look at the crosses: they are at the beginning of the curves. You will notice that the values of this confusion matrix on the LHS of the window differs from the "regular" confusion matrix shown in the Result panel.**

Look attentively at the window for the Cost/Benefit Analysis. It consists of several panels. The left part of the window contains the Plot: ThresholdCurve frame with the Threshold Curve (called also the **Lift curve**). The Threshold curve looks very similar to the ROC curve. In both of them the axis Y corresponds to the true positive rate. However, in contrast to the ROC curve, *the axis X in the Threshold curve corresponds to the part of selected instances* (the "Sample Size"). In other words, the Threshold curve depicts the dependence of the part of "junk" emails retrieved in the course of predicting selected from the whole dataset (ie only those selected for which the estimated probability of being junk exceeds the chosen threshold). The value of the threshold can be modified interactively by moving the slider in the Threshold frame of the Cost/Benefit Analysis window. The confusion matrix for the current value of the threshold is shown in the Confusion Matrix frame at the left bottom corner of the window. The values of the confusion matrix changes when you move the slider.

Click on the Minimize Cost/Benefit button at the right bottom corner of the window. You will see that the values of the confusion matrix change and that the crosses change their position on the curves (see Fig 7).

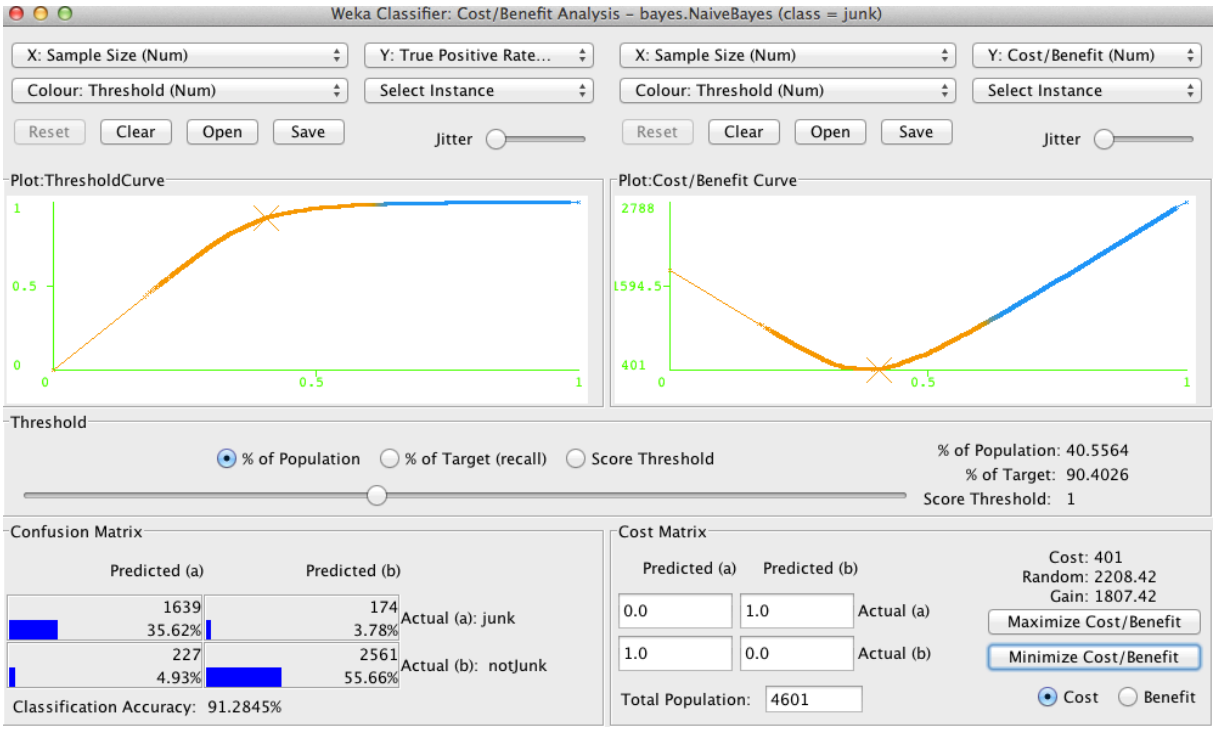



**Fig. 7. This is what you see when you press "Minimize Cost/Benefit".**

Why? Let us take a look at the right side of the window first. Its right bottom corner contains the *Cost Matrix* frame. The left part of the frame contains the Cost matrix itself. Its four entries indicate the cost one should pay for decisions taken on the base of the classification model. The cost values are expressed in the table in abstract units, however in the case studies they can be considered in money scale, for example, in US Dollars or GBPounds or EUROS or any other currency. The left bottom cell of the Cost matrix defines the cost of false positives. Its default value is 1 unit. In the case of spam[7], a **false positive** corresponds to the *mean price* one should pay when the classifier puts a notjunk message into the junk category (false alarm). The right top cell of the Cost matrix defines the cost of false negatives. Its default value is 1 unit. In the case of junk, a false negative corresponds to when the classifer places junk into the notjunk category. It is also taken by default that one should not pay price for correct decision taken using the classification model. It is clear that all these settings can be changed in order to match the real situation taking place in the process of problem at hand. In order to find the threshold corresponding to the minimum cost, it is sufficient to press the button Minimize Cost/Benefit. The current value of the cost is compared by the program with the cost of selecting the same number of instances at random. The difference between the values of the cost function between the random selection and the current value of the cost is called Gain, indicated at the right side of the frame. Unfortunately, the current version of the Weka software does not provide the means of automatic maximization of the Gain function. However, this can easily be done interactively by moving the slider in the Threshold frame of the Cost/Benefit Analysis window.
---the end---

[7] A normal spam filter that categorizes mail into a spam and a ham (notspam) category produces a false positive when it puts a ham (notspam) message into the spam category and a false negative when it places spam into the ham (notspam) category.