Tutorial on Classification

Igor Baskin and Alexandre Varnek

Introduction

The tutorial demonstrates possibilities offered by the Weka software to build classification models for SAR (Structure-Activity Relationships) analysis. Two types of classification tasks will be considered – two-class and multi-class classification. In all cases protein-ligand binding data will analyzed, ligands exhibiting strong binding affinity towards a certain protein being considered as “active” with respect to it. If it is not known about the binding affinity of a ligand towards the protein, such ligand is conventionally considered as “nonactive” one. In this case, the goal of classification models is to be able to predict whether a new ligand will exhibit strong binding activity toward certain protein biotargets. In the latter case one can expect that such ligands might possess the corresponding type of biological activity and therefore could be used as ‘’hits” for drug design. All ligands in this tutorial are described by means of an extended set of MACCS fingerprints, each of them comprising 1024 bits, the “on” value of each of them indicating the presence of a certain structural feature in ligand, otherwise its value being “off”.

Part 1. Two-Class Classification Models.

1. Data and descriptors.

The dataset for this tutorial contains 49 ligands of Angeotensin-Converting Enzyme (ACE) and 1797 decoy compounds chosen from the DUD database. The set of "extended" MACCS fingerprints is used as descriptors.

2. Files

The following file is supplied for the tutorial:

- ace.arff – descriptor and activity values

3. Exercise 1: Building the Trivial model ZeroR

In this exercise, we build the trivial model ***ZeroR***, in which all compounds are classified as “nonactive”. The goal is to demonstrate that the *accuracy* is not a correct choice to measure

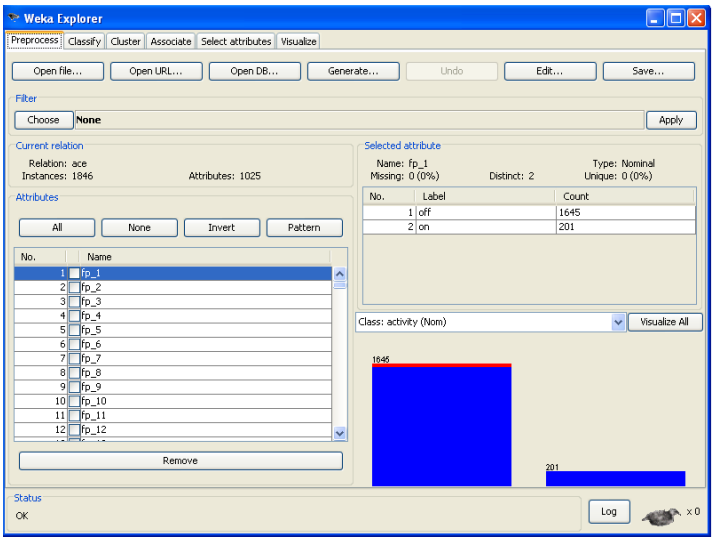
the performance of classification for unbalanced datasets, in which the number of "nonactive" compounds is much larger than the number of "active" ones.

Step by step instructions

Important note for Windows users: During the installation, the ARFF files should be associated with Weka.

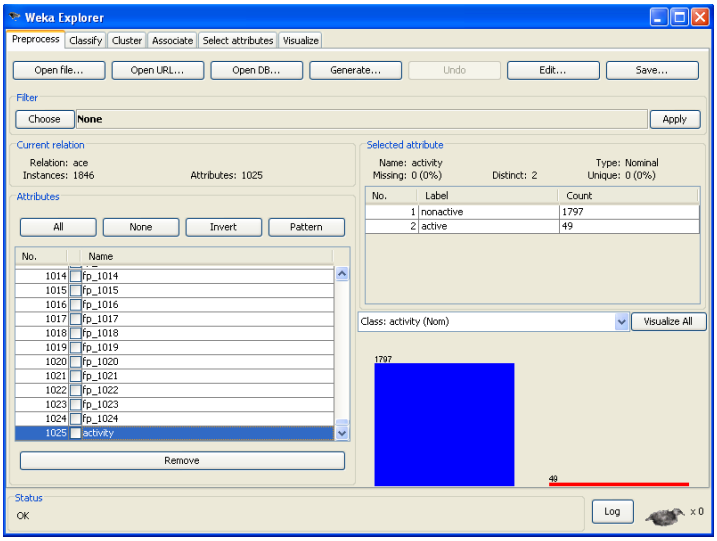
In the starting interface of Weka, click on the button ***Explorer***.

- In the ***Preprocess*** tab, click on the button ***Open File***. In the file selection interface, select the file ace.arff.



The dataset is characterized in the ***Current relation*** frame: the name, the number of instances (compounds), the number of attributes (descriptors + activity/property). We see in this frame that the number of compounds is 1846, whereas the number of descriptors is 1024, which is the number of attributes (1025) minus the activity field. The ***Attributes*** frame allows user to modify the set of attributes using *select* and *remove* options. Information about the selected attribute is given in the ***Selected attribute*** frame in which a histogram depicts the attribute distribution. One can see that the value of the currently selected descriptor fp_1 (the first bit in the corresponding fingerprint) is "on" in 201 compounds and "off" in 1645 compounds in the dataset.

- Select the last attribute "activity" in the ***Attributes*** frame.

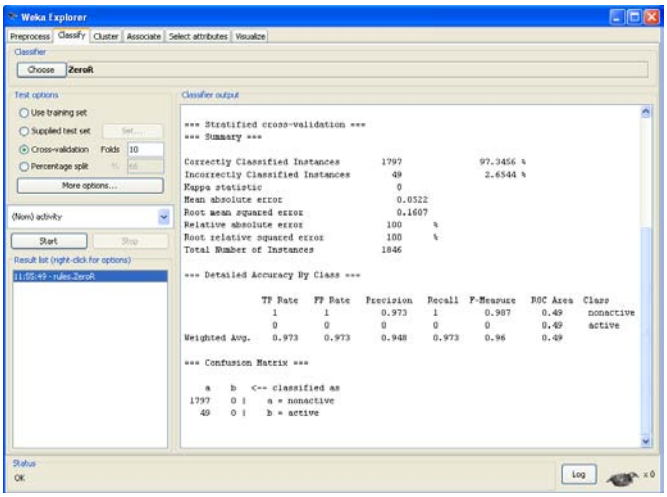


One can read from the ***Selected attribute*** frame that there are 1797 nonactive and 49 active compounds in the dataset. Nonactive compounds are depicted by the blue color whereas active compounds are depicted by the red color in the histogram.

- Click on the tab ***Classify***.

The ***ZeroR*** method is already selected by default. For assessing the predictive performance of all models to be built, the 10-fold cross-validation method has also be specified by default.

- Click on the ***Start*** button to build a model.



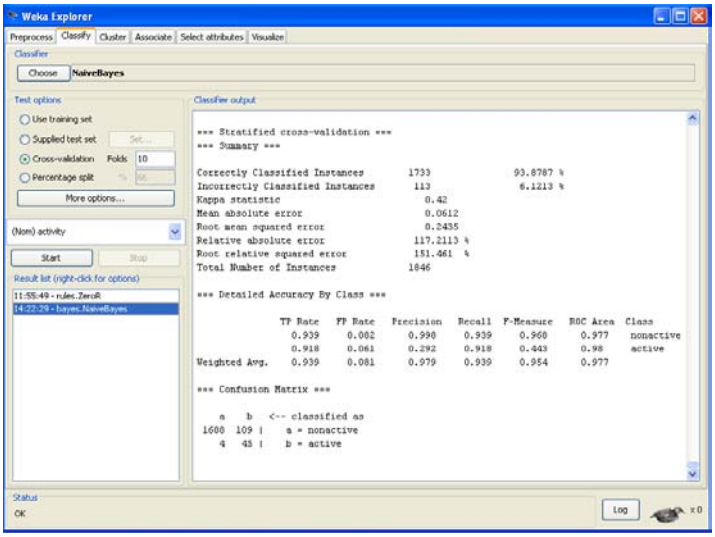
The predictive performance of the model is characterized in the right-hand ***Classifier output*** frame. The Confusion Matrix for the model is presented at the bottom part of the ***Classifier output*** window. It can be seen from it that all compounds have been classified as “nonactive”. It is clear that such

trivial model is useless and it cannot be used for discovering “active” compounds. However, pay attention that the accuracy of the model (Correctly Classifieds Instances) of this trivial model is very high: 97.3456 %. This fact clearly indicates that the accuracy cannot be used for assessing the usefulness of classification models built using unbalanced datasets. For this purpose a good choice is to use the “Kappa statistic”, which is zero for this case. “Kappa statistic” is an analog of correlation coefficient. Its value is zero for the lack of any relation and approaches to one for very strong statistical relation between the class label and attributes of instances, i.e. between the class of biological activity of chemical compounds and the values of their descriptors. Another useful statistical characteristic is “ROC Area”, for which the value near 0.5 means the lack of any statistical dependence.

4. Exercise 2: Building the Naïve Bayesian Model

In this exercise, we build a Naïve Bayesian model for predicting the ability of chemical compounds to bind to the Angeotensin-Converting Enzyme (ACE). The goal is to demonstrate the ability of Weka to build statistically significant classification models for predicting biological activity of chemical compounds, as well as to show different ways of assessing the statistical significance and usefulness of classification models.

- In the ***classifier*** frame, click ***Chose***, then select the *NaiveBayes* method from the *bayes* submenu.
- Click on the ***Start*** button to build a model.

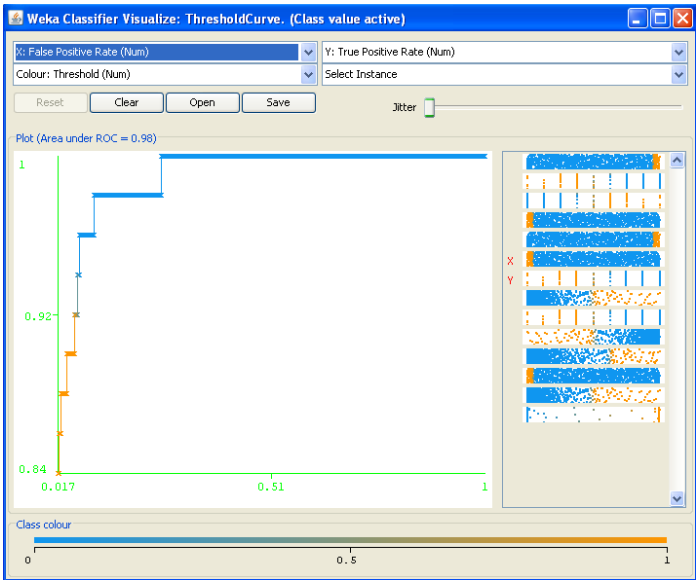


Although the accuracy of the model became lower (93.8787 % instead of 97.3456 %), its real statistical significance became much stronger. This follows from the value of the ‘’Kappa statistic” 0.42, which indicates the existence of moderate statistical dependence. It can be analyzed using the

"Confusion Matrix" at the bottom of the ***Classifier output*** window. So, there are 45 true positive, 1688 true negative, 109 false positive, and 4 false negative, and 109 false positive compounds. It is because of the considerable number of false positive that the value of recall for "active" compounds 0.292 is rather low. Nonetheless, the model exhibits an excellent value of "ROC Area" for "active" compounds 0.98. This indicates that this Naïve Bayesian model could very advantageously be used for discovering biologically active compounds through virtual screening. This can clearly be shown by analyzing ROC and Cost/Benefit plots.

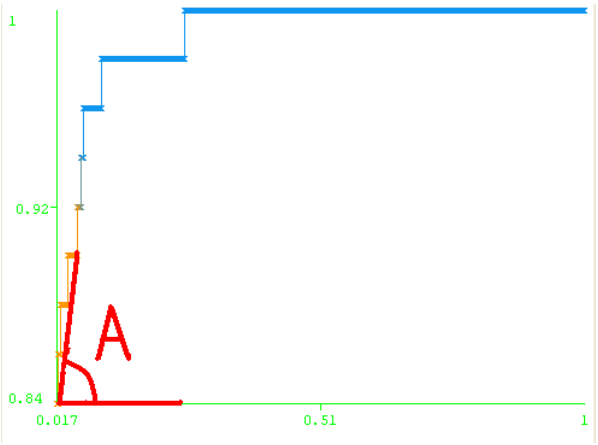
The Naïve Bayes method provides probabilistic outputs. This means that Naïve Bayes models can assess the value of the probability (varying from 0 to 1) that a given compound can be predicted as "active". By moving the threshold from 0 to 1 and imposing that a compound can be predicted as "active" if the corresponding probability exceeds the current threshold, one can build the ROC (Receiver Operating Characteristic) curve.

- Visualize the ROC curve by clicking the right mouse button on the model type bayes.NaiveBayes in the ***Result list*** frame and selecting the menu item *Visualize threshold curve / active*.



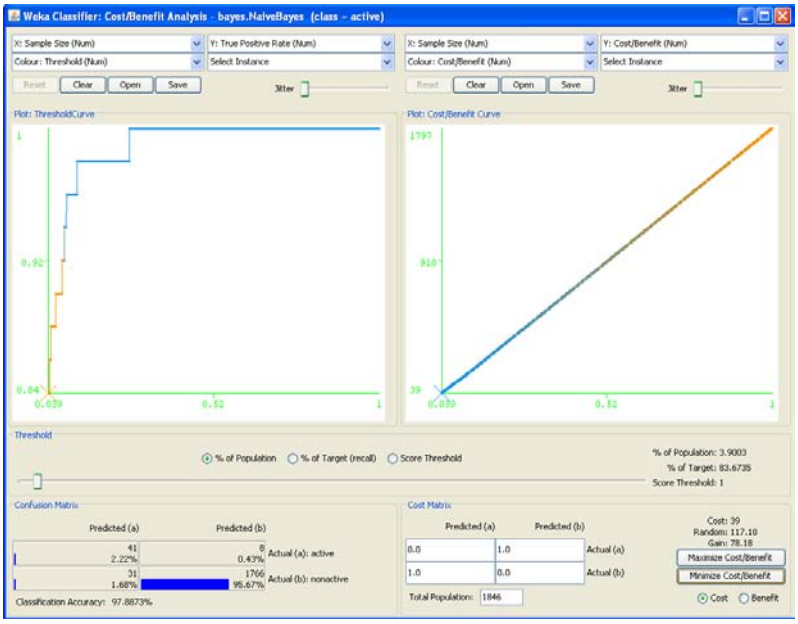
The ROC curve is shown in the ***Plot*** frame of the window. The axis X in it corresponds to the false positive rate, whereas its axis Y corresponds to the true positive rate. The color depicts the value of the threshold. The "colder" (closer to the blue) color corresponds to the lower threshold value. All compounds with probability of being "active" exceeding the current threshold are predicted as "active". If such prediction made for a current compound is correct, then the corresponding compound is true positive, otherwise it is false positive. If for some values of the threshold the true positive rate greatly exceeds the false positive rate (which is indicated by the angle *A* close to 90 degrees), then the classification model with such threshold can be used to extract selectively "active"

compounds from its mixture with the big number of “nonactive” ones in the course of virtual screening.

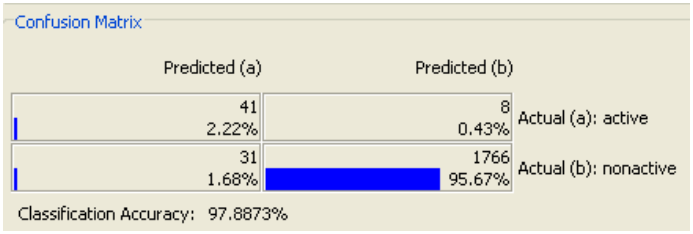


In order to find the optimal value of the threshold (or the optimal part of compounds to be selected in the course of virtual screening), one can perform the cost/benefit analysis.

- Close the window with the ROC curve.
- Open the window for the cost/benefir analysis by clicking the right mouse button on the model type bayes.NaiveBayes in the ***Result list*** frame and selecting the menu item *Cost/Benefit analysis / active*.
- Click on the ***Minimize Cost/Benefit*** button at the right bottom corner of the window.

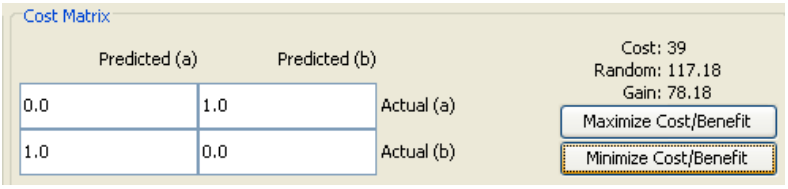


Consider attentively the window for the Cost/Benefit Analysis. It consists of several panels. The left part of the window contains the ***Plot: ThresholdCurve*** frame with the Threshold Curve (called also the Lift curve). The Threshold curve looks very similar to the ROC curve. In both of them the axis 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 "active" compounds retrieved in the course of virtual screening upon the part of compounds selected from the whole dataset used for screening. Remind that only those compounds are selected in the course of virtual screening, for which the estimated probability of being "active" 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.



Pay attention that the confusion matrix for the current value of the threshold sharply differs from the previously obtained one. In particular, the classification accuracy 97.8873 % is considerably higher than the previous value 93.8787 %, the number of false positives has greatly decreased from 109 to 31, whereas the number of false negatives has increased from 4 to 8. Why is this happening?

In order to give an answer to this question and explain the corresponding phenomenon, let us take a look at the right side of the window. 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 EUROs. The left bottom cell of the Cost matrix defines the cost of false positives. Its default value is 1 unit. In the case of virtual screening this corresponds to the mean price one should pay in order to synthesize (or purchase) and test a compound wrongly predicted by the model as "active". The right top cell of the Cost matrix defines the cost of false negatives. Its default value is 1 unit. In the case of virtual screening this corresponds to the mean price one should pay for "throwing away" very useful compound and losing profit because of the wrong prediction taken by the classification model. It is also taken by default that one should not pay price for correct decisions

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 drug design.

The overall cost corresponding to the current value of the threshold is indicated at the right side of the frame. Its current value is 39 (cost of 31 false positives and 8 false negatives). In order to find the threshold corresponding to the minimum cost, it is sufficient to press the button ***Minimize Cost/Benefit***. This explains the afore-mentioned difference in confusion matrices. The initial confusion matrix corresponds to the threshold 0.5, whereas the second confusion matrix results from the value of the threshold found by minimizing the cost function.

The current value of the cost is compared by the program with the cost of selecting the same number of instances at random. Its value 117.18 is indicated at the right side of the frame. The difference between the values of the cost function between the random selection and the current value of the cost is called Gain. Its current value 78.18 is also indicated at the right side of the frame. In the context of virtual screening, the Gain can be interpreted as the profit obtained by using the classification model instead of random selection of the same number of chemical compounds. 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 current model corresponds to the minimum value of the Cost function. Read the values for the current threshold from the right side of the ***Threshold*** frame.

So, the current model (with the threshold obtained by minimizing the cost) specifies that it is optimal to select 3.9003 % of compounds in the course of virtual screening, and this ensures retrieving of 83.6735 % of active compounds.

- Close the window with the Cost/Benefit Analysis.

Exercise: Find the model corresponding to the maximum Gain.

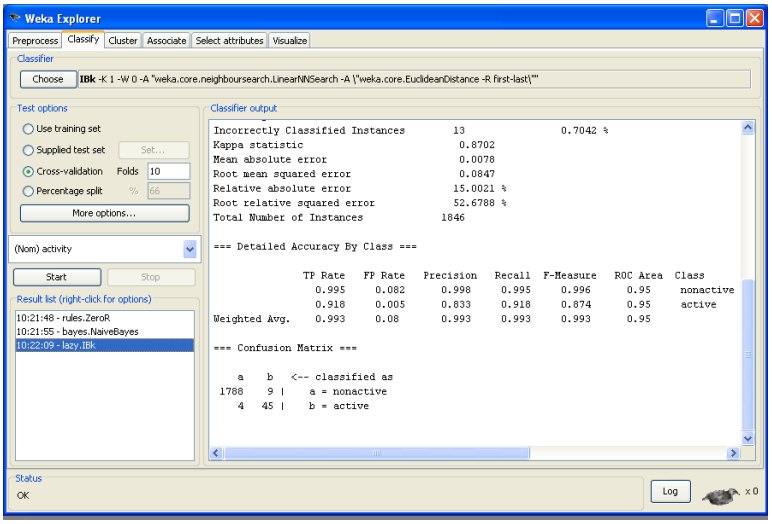
5. Exercise 3: Building the Nearest Neighbors Models (k-NN)

In this exercise, we build k-NN models for predicting the ability of chemical compounds to bind to the Angeotensin-Converting Enzyme (ACE). The goal is to learn how to use instance-based (lazy) methods.

- In the ***classifier*** frame, click ***Chose***, then select the *IBk* method from the *lazy* submenu.

The lazy submenu contains a group of methods, in which the training phase is almost omitted – it actually amounts to memorizing all instances from the training set. Instead of it, all main calculations are delayed to the test phase. That is why such methods are sometimes called lazy, instance-based and memory-based. The price for this "laziness" is however rather high – computations at the test phase are very intensive, and that is why such methods work very slowly during prediction, especially for big training sets. So, the abbreviation *IBk* means that this is an Instance-Based method based on k neighbours. The default value of *k* is 1. So, build an 1-NN model

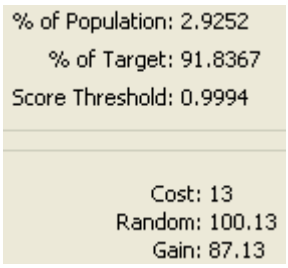
- Click on the ***Start*** button to build a 1-NN model.



One can see that the 1 Nearest Neighbour model is statistically much stronger than the previous Naïve Bayes one. In particular, the number of Incorrectly Classified Instances has decreased from 113 to 13, whereas the Kappa statistic has increased from 0.42 to 0.8702. Nonetheless, the ROC Area became slightly smaller in comparison with the Naïve Bayes model. Now perform the Cost/Benefit Analysis for the 1-NN model.

- Click the right mouse button on the model type *lazy.IBk* in the ***Result list*** frame and selecting the menu item *Cost/Benefit analysis / active*.

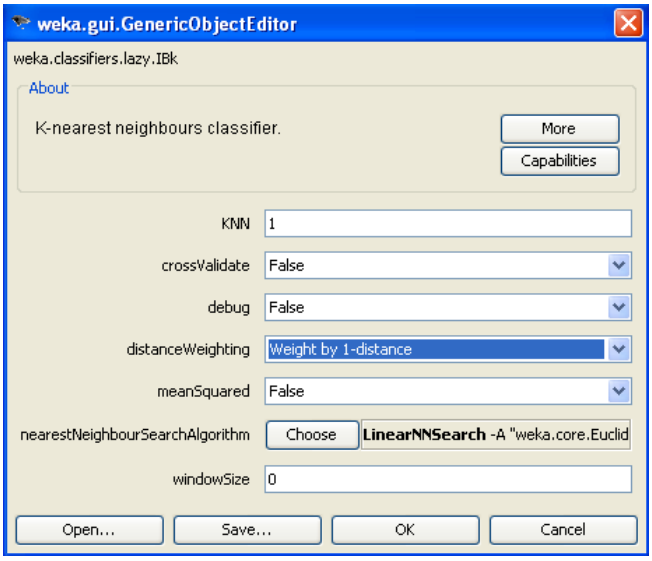
- Click on the ***Minimize Cost/Benefit*** button at the right bottom corner of the window.



One can see that the Cost became considerably lower (13 vs 39), and the Gain became higher (87.13 vs 78.18). It can also be checked that the initial 1-NN model corresponds to the lowest Cost and the

highest Gain. It can also be seen that when using the 1-NN classifier in virtual screening it is sufficient to select 2.9252 % of compounds in order to retrieve 91.8367 % "active" ones. Can this result be further improved? Yes, this can be carried out by using the weighted modification of the k-NN method.

- Close the window with the Cost/Benefit Analysis.
- Click with the left mouse button on the word *IBk* in the ***Classifier*** frame. The window for setting options for the k-NN method pops up.
- Change the option *distanceWeighting* to *Weight to 1-distance*.



- Click on the ***OK*** button.
- Click on the ***Start*** button.

One can see that the ROC Area has increased from 0.95 to 0.977, although the accuracy of prediction and the Kappa statistics have not changed.

- Repeat the Cost/Benefit analysis.

So, after the introduction of weighting, the Cost became lower (11 vs 13), the Gain became slightly higher (87.24 vs 87.13), and now it is sufficient to screen 2.8169 % (instead of 2.9252 %) of

compounds in order to retrieve the same number of the “active” ones. So, some moderate improvement has been achieved.

So, the Nearest Neighbours approach appeared to be considerably more efficient than the Naïve Bayes for predicting the ability of chemical compounds to bind to the Angeotensin-Converting Enzyme (ACE) using the set of "extended" MACCS fingerprints as descriptors. The question arises: is the Nearest Neighbours approach always more efficient than the Naïve Bayes? The answer is: no. In this case the exceptionally high performance of the 1-NN method can be explained by the fact that the MACCS fingerprints have specially been optimized so as to provide high performance of retrieving “active” compounds in similarity search, which is actually 1-NN. With other sets of descriptors, the results might be different.

Exercise: Build two 3-NN models (with and without weighting) for the same dataset and analyze their relative performances in comparison with the corresponding 1-NN models. Hint: the kNN option in the k-NN parameters window should be changed.

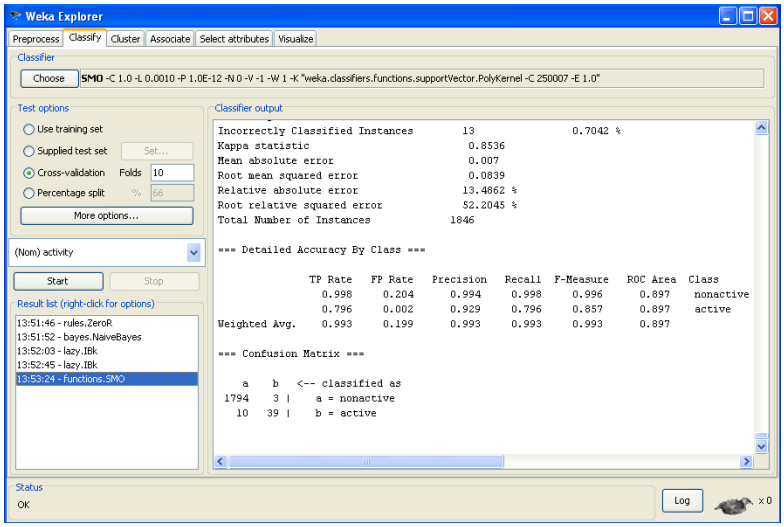
6. Exercise 4: Building the Support Vector Machine Models

In this exercise, we build Support Vector Machine (SVM) models for predicting the ability of chemical compounds to bind to the Angeotensin-Converting Enzyme (ACE). The goal is to learn the possibilities offered by the Weka software for that.

- In the ***classifier*** frame, click ***Chose***, then select the *SMO* method from the *functions* submenu.

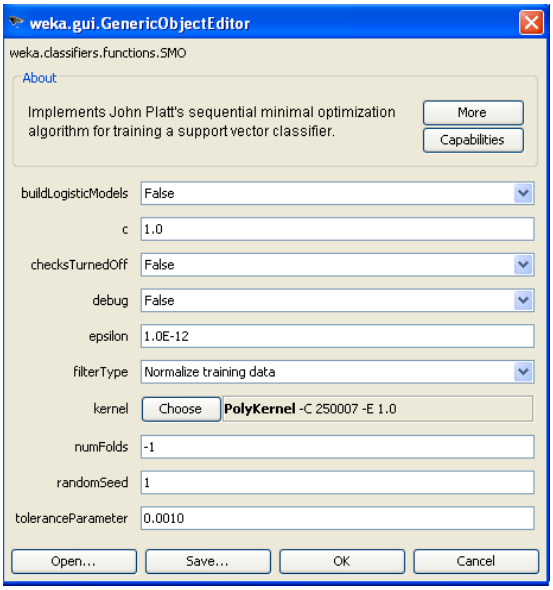
The Weka software implements John Platt’s Sequential Minimal Optimization (SMO) algorithm for training a support vector classifier, and this explains abbreviation *SMO* used in Weka for this methods.

- Click on the ***Start*** button.

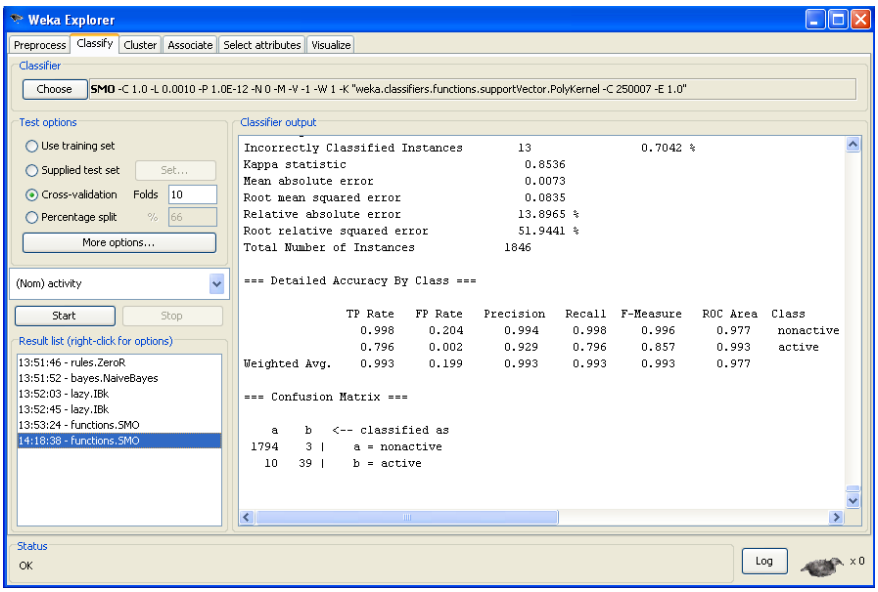


We have obtained very good model with a small number of misclassification errors (13) and rather high value of the Kappa statistic 0.8536. The only thing that became worse in comparison with previous models is ROC Area. This can however be easily explained by the fact that original SVM method is not probabilistic, and only a single optimal value of threshold (which is in the case of the standard SVM approach the distance between the separating hyperplane and the coordinate origin) is provided. Without such freely moving threshold, it would not be possible to perform virtual screening based on ranking chemical compounds and adjusting threshold for selection. This results in the relatively bad value of ROC Area. Nonetheless this can be improved by using a special modification of the original SVM approach, which assigns probability value to each prediction. Since the algorithm for assigning probability values to SVM predictions is based on the use of logistic functions, such models are called in Weka *Logistic Models*.

- Click with the left mouse button on the word *SOM* in the ***Classifier*** frame. The window for setting options for the SVM method pops up.

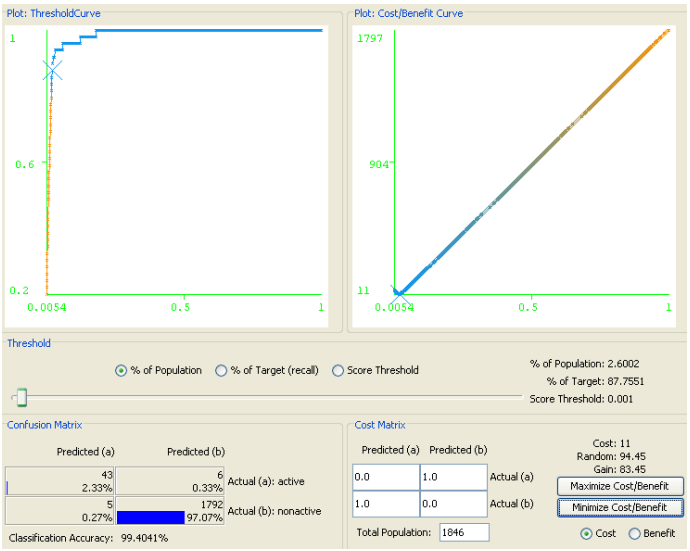


- Change the option *buildLogisticModels* to True.
- Click on the ***OK*** button.
- Click on the ***Start*** button.



Although the accuracy of prediction has not changed, the ROC Area became very high – 0.993 for "active" compounds. For such probabilistic variant of the SVM method, good ROC curves can be built, and the Cost/benefit analysis can easily be performed.

- Click the right mouse button on the model type *functions.SMO* in the ***Result list*** frame and select the menu item *Cost/Benefit analysis / active*.
- Click on the ***Minimize Cost/Benefit*** button at the right bottom corner of the window.

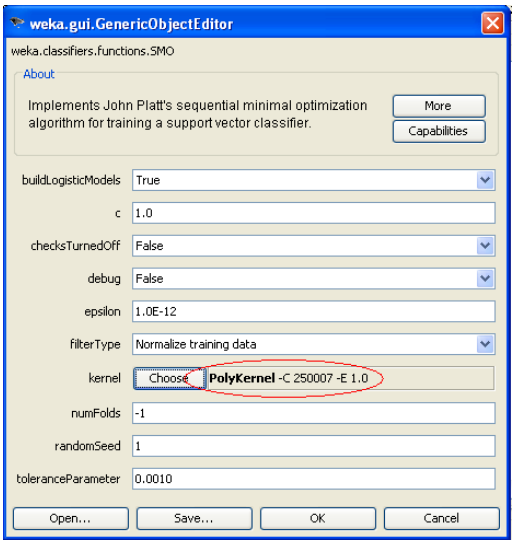


One can see that the value of the Cost function is low (11), whereas the Gain is rather high (83.45). In order to retrieve 87.7551 % of active compounds in the course of virtual screening, it is possible to select only 2.6002 % of compounds. The Threshold Curve at the left side of the window also demonstrates very good performance of the probabilistic SVM approach in virtual screening.

- Close the window with the Cost/Benefit Analysis.

The obtained results can further be improved. All these models have been built using the linear kernels chosen by default. Such kernels take into account only individual impacts of descriptors (in this case fingerprint bits), but do not consider their interaction (in this case, interaction of features corresponding to different fingerprint bits). All binary interactions of features can be depicted using the quadratic kernels. Let us build a new probabilistic SVM model for the quadratic kernel.

- Click with the left mouse button on the word *SOM* in the ***Classifier*** frame. The window for setting options for the SVM method pops up.

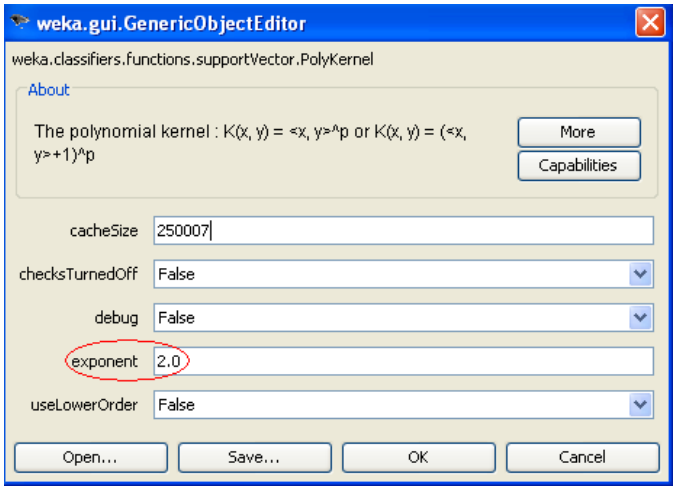


Now change the kernel from the linear to the quadratic one. All of them are particular cases of the polynomial kernel with different exponents (one for the linear, and two to the quadratic kernel). Therefore, in order to obtain the quadratic kernel, it is sufficient to set the exponent parameter of the polynomial kernel to 2, and it is not necessary to change the type of kernel.

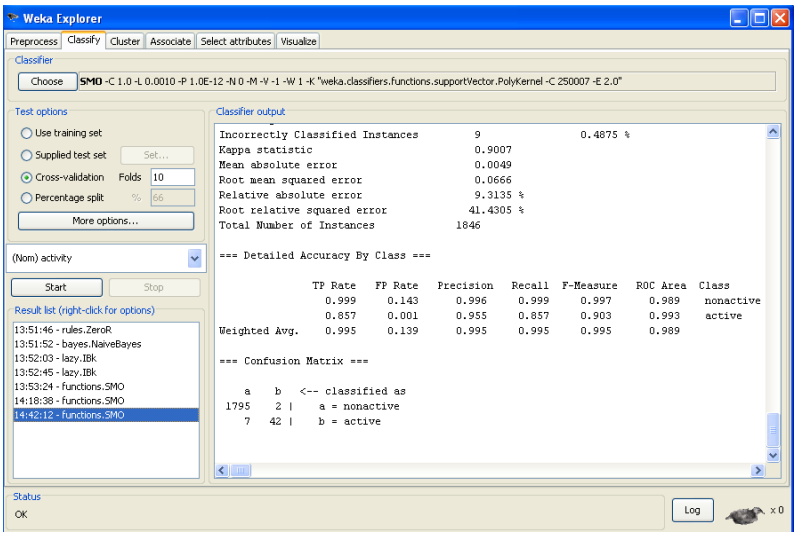
- Click with left mouse button on the ***PolyKernel*** word near the *kernel* label.

A new window with parameters of the polynomial kernel pops up.

- Change the value of the *exponent* option from 1.0 to 2.0



- Click on the ***OK*** button to close the window with polynomial kernel options.
- Click on the ***OK*** button to close the window with SVM options.
- Click on the ***Start*** button.



So, all statistical parameters have substantially improved in comparison with the case of the linear kernel. In particular, the number of misclassification errors has dropped from 13 to 9, the value of the Kappa statistics has raised from 0.8536 to 0.9007.

- Perform the Cost/Benefit analysis.

The Cost has fallen even further from 9 to 7, in comparison with the linear kernel. With the quadratic kernel, it is sufficient to select only 2.2752 % of compounds in order to retrieve 85.7143 % of “active” ones.

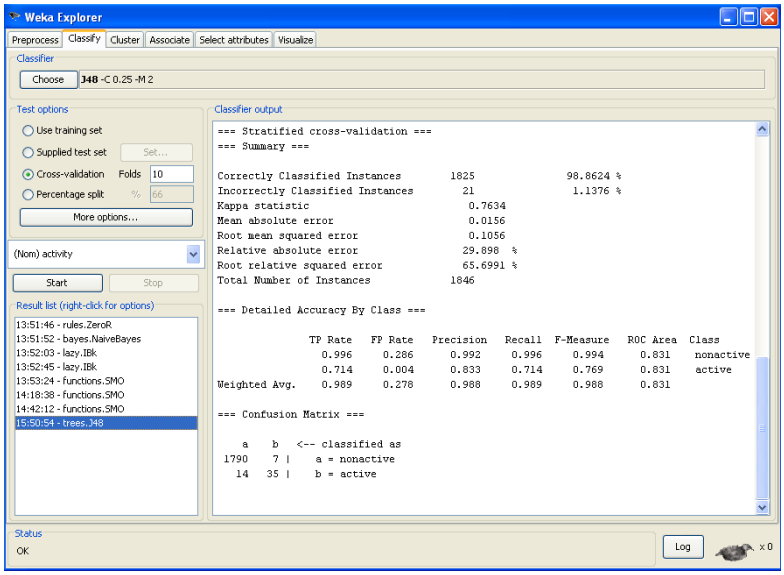
So, the transition to the quadratic kernel from the linear one has lead to substantial improvement of SVM classification models. This means that it is important to consider not just individual features coded by fingerprints, but also nonlinear interaction between them. Unfortunately, very popular Tanimoto similarity measure does not take this into account.

Exercise: Rebuild probabilistic SVM model with quadratic kernel for different values of parameter C (trade-off between errors and model complexity). Trye the values 0.1, 0.5, 2, 10. Can any improvement be achieved in comparison with the use of its default value 1?

7. Exercise 5: Building a Classification Tree Model

In this exercise, we build a classification tree model (using the C4 method named in Weka as J48) for predicting the ability of chemical compounds to bind to the Angeotensin-Converting Enzyme (ACE). The goal is to learn the possibilities offered by the Weka software to build and visualize classification trees.

- In the ***classifier*** frame, click ***Choose***, then select the *J48* method from the *trees* submenu.
- Click on the ***Start*** button.



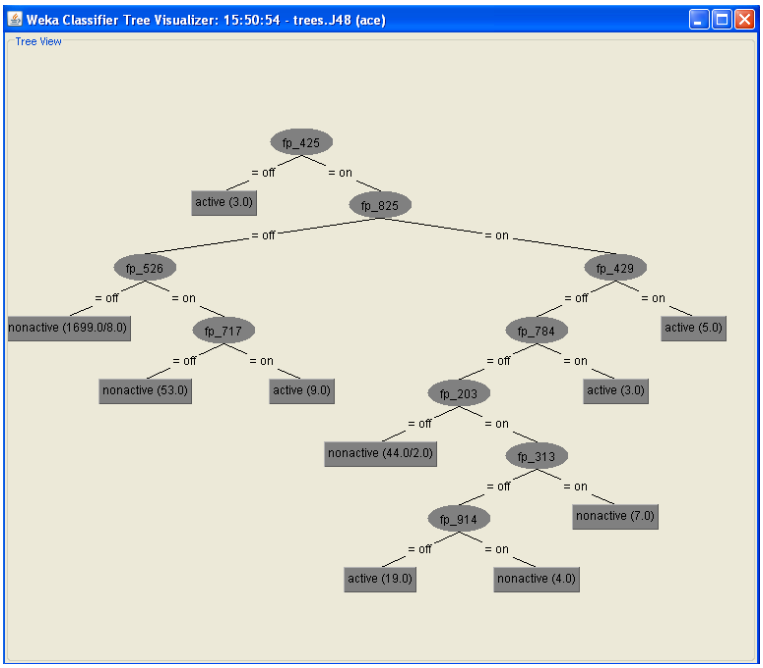
The statistical parameters of the *J48* model appears not high, especially in comparison with previously considered methods. Nonetheless, the main strength of individual classification trees

stems not from high statistical significance of models, but from their interpretation ability. In order to visualize the classification tree in the text mode, scroll the text field in the ***Classifier output*** frame up.

```
fp_425 = off: active (3.0)
fp_425 = on
|   fp_825 = off
|   |   fp_526 = off: nonactive (1699.0/8.0)
|   |   fp_526 = on
|   |   |   fp_717 = off: nonactive (53.0)
|   |   |   fp_717 = on: active (9.0)
|   fp_825 = on
|   |   fp_429 = off
|   |   |   fp_784 - off
|   |   |   |   fp_203 = off: nonactive (44.0/2.0)
|   |   |   |   fp_203 = on
|   |   |   |   |   fp_313 = off
|   |   |   |   |   |   fp_914 = off: active (19.0)
|   |   |   |   |   |   fp_914 = on: nonactive (4.0)
|   |   |   |   fp_313 = on: nonactive (7.0)
|   |   |   fp_784 = on: active (3.0)
|   |   fp_429 = on: active (5.0)
```

In order to obtain more usual representation of the same tree, do the following

- Click the right mouse button on the model type *trees.J48* in the ***Result list*** frame and select the menu item *Visualize tree*.
- Resize a new window with graphical representation of the tree
- Clock with the right mouse button to the space in this screen, and in the popup menu select the item *Fit to screen*.



The ***Tree View*** graphical diagram can be used to visualize decision trees. It contains two type of nodes, ovals and rectangles. Each oval contains a query of the sort: does chemical structure contains a feature depicted by the specified fingerprint bit number. If the answer is “yes”, then the node connected with the previous one with the “= on” branch is queried next. Otherwise, the “= off” branch is activated. The tree top node is queried the first. The “leaves” of the tree, depicted by rectangular, contain final decisions, whether the current compound is active or not.

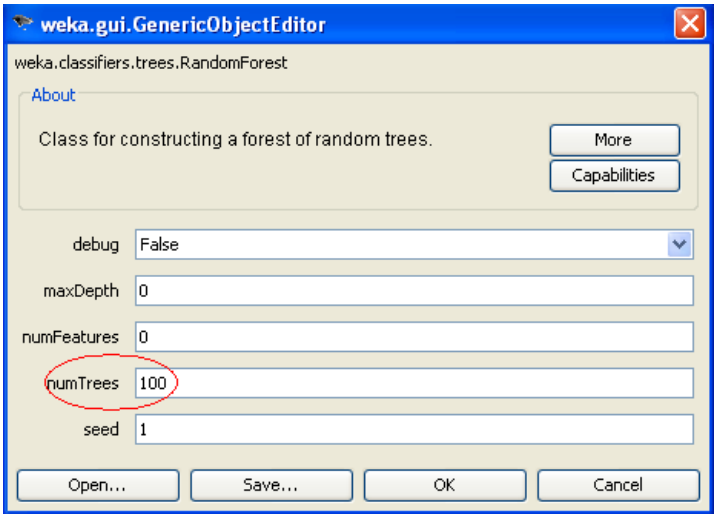
Exercise: Build the ROC curve and perform the Cost/Benefit analysis of the J48 model.

8. Exercise 6: Building a Random Forest Model

In this exercise, we build a Random Forest model for predicting the ability of chemical compounds to bind to the Angeotensin-Converting Enzyme (ACE). The goal is to learn the possibilities offered by the Weka software to build Random Fore models.

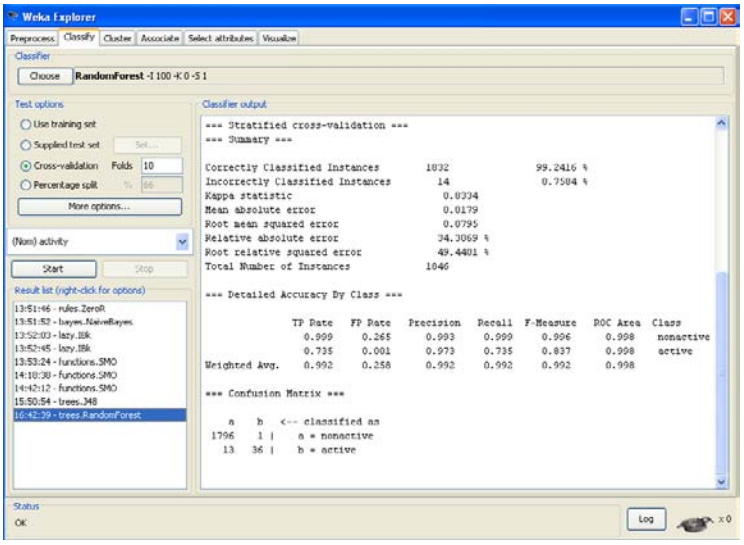
Although models built using individual decision trees are not very strong from statistical point of view, they can largely be improved by applying ensemble modeling. In the latter case, an ensemble of several models is built instead of a single one, and prediction of the ensemble model is made as a consensus of predictions made by all its individual members. The most widely used method based on the ensemble modeling is Random Forest, which has recently become very popular in chemoinformatics.

- In the ***classifier*** frame, click ***Choose***, then select the *J48* method from the *trees* submenu.
- Click with the left mouse button on the word *RandomForest* in the ***Classifier*** frame. The window for setting options for the Random Forest method pops up.
- Change the value of the *numTrees* option from 10 to 100



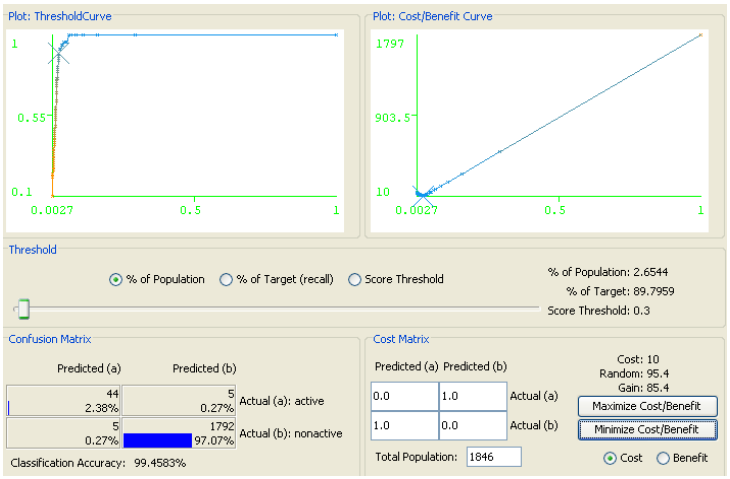
We have changed the default number of trees 10 in ensemble to 100

- Click on the ***OK*** button to close the window with the Random Forest options.
- Click on the ***Start*** button.



The resulting model is rather strong. Although its classification accuracy and the value of Kappa statistics are worse than for the SVM model, the ROC Area appears to be very high. This means that it can advantageously be applied in virtual screening. Indeed, perform the Cost/Benefir Analysis of the model.

- Click the right mouse button on the model type *trees.RandomForest* in the ***Result list*** frame and select the menu item *Cost/Benefit analysis / active*.
- Click on the ***Minimize Cost/Benefit*** button at the right bottom corner of the window.



Very good Cost/Benefit parameters are observed. The Cost is rather low (10), the Gain is rather high (85.4), it is sufficient to select only 2.6544 % of compounds in order to retrieve 89.7959 % of “active” ones.

Exercise: Study the dependence of the Kappa statistic and ROC Area upon the number of trees in ensemble. Try 10, 20, 30, 40, 50, 100, 200 trees.

Part 2. Milti-Class Classification Models.

1. Data and descriptors.

The dataset for this tutorial contains 3961 ligands to 40 different protein biotargets and 3127 decoy compounds chosen from the DUD database []. The extended set of MACCS fingerprints is used as descriptors.

2. Files

The following file is supplied for the tutorial:

- dud.arff – descriptor and activity values

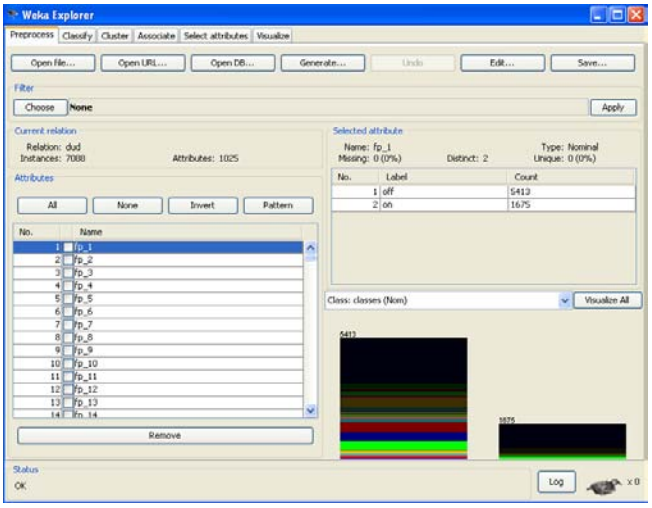
3. Exercise 7: Building the Naïve Bayes Model

In this exercise we will show how the Naïve Bayes method implemented in the Weka software can be apply for building a multi-class model capable of predicting affinity to 40 pharmaceutically important protein biotargets. In this case the output attribute is called “classes” and it can take 41 different values: the names of biotargets and “none” for the lack of affinity.

Step by step instructions

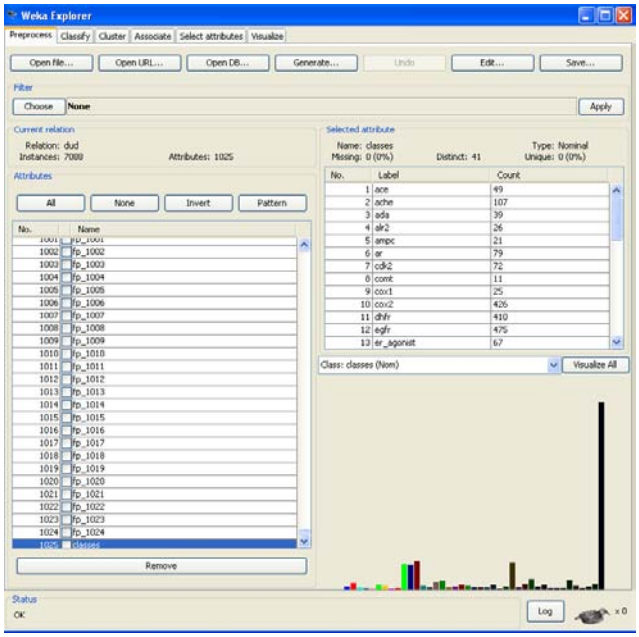
Important note for Windows users: During installation, the ARFF files should have been associated with Weka. In this case, it is highly recommended to locate and double click on the file dud.arff, and to skip the following three points.

- In the starting interface of Weka, click on the button ***Explorer***.
- In the ***Preprocess*** tab, click on the button ***Open File***. In the file selection interface, select the file dud.arff.



The dataset is characterized in the ***Current relation*** frame: the name (*dud*), the number of instances (compounds), the number of attributes (descriptors + activity/property). We see in this frame that the number of compounds is 7088, whereas the number of descriptors is 1024, which is the number of attributes (1025) minus the "classes" field. The ***Attributes*** frame allows user to modify the set of attributes using *select* and *remove* options. Information about the selected attribute is given in the ***Selected attribute*** frame in which a histogram depicts the attribute distribution. One can see that the value of the currently selected descriptor fp_1 (the first bit in the corresponding fingerprint) is "on" in 1675 compounds and "off" in 5413 compounds in the dataset.

- Select the last attribute "classes" in the ***Attributes*** frame.



One can read from the ***Selected attribute*** frame the list of different classes (40 types of biotargets and 'none' for decoys) and the number of compounds belonging to each of the classes (i.e. the number of ligands strongly binding to the corresponding protein). Compounds binding to different

biotargets are depicted with different colors in the histogram. The last black color corresponds to "decoys".

- Click on the tab ***Classify***.
- In the ***classifier*** frame, click ***Chose***, then select the *NaiveBayes* method from the *bayes* submenu.
- Click on the ***Start*** button to build a model.

All information concerning the predictive performance of the resulting model can be extracted from the text field in the right-hand ***Classifier output*** frame. Consider first the global statistics.

```
Correctly Classified Instances        5766               81.3488 %
Incorrectly Classified Instances      1322               18.6512 %
Kappa statistic                          0.7688
```

We can see that for 81.3488 % of ligands the corresponding biotargets have been correctly predicted. The value of the Kappa statistic 0.7688 means that statistical significance of the model is rather high. Therefore, it can be applied in "target fishing", *i.e.* in prediction of putative biological targets for a given compound.

Consider individual statistic for each of the targets.

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.918	0.003	0.682	0.918	0.783	0.999	ace
0.86	0.011	0.554	0.86	0.674	0.988	ache
0.897	0.001	0.795	0.897	0.843	0.999	ada
0.654	0.003	0.486	0.654	0.557	0.982	alr2
0.857	0.002	0.621	0.857	0.72	0.997	ampc
0.456	0.001	0.783	0.456	0.576	0.991	ar
0.611	0.004	0.595	0.611	0.603	0.979	cdk2
0.455	0.002	0.294	0.455	0.357	0.98	comt
0.4	0.016	0.081	0.4	0.135	0.967	cox1
0.847	0.004	0.933	0.847	0.888	0.99	cox2
0.956	0	0.997	0.956	0.976	1	dhfr
0.857	0.004	0.936	0.857	0.895	0.98	egfr
0.776	0.008	0.495	0.776	0.605	0.987	er_agonist
1	0.001	0.907	1	0.951	1	er_antagonist
0.858	0.03	0.333	0.858	0.48	0.982	fgfr1
0.705	0.002	0.896	0.705	0.789	0.986	fxa
0.975	0	1	0.975	0.987	1	gart
0.923	0.001	0.906	0.923	0.914	0.991	gpb
0.679	0.001	0.93	0.679	0.785	0.946	gr
0.645	0.001	0.851	0.645	0.734	0.992	hivpr
0.651	0.019	0.172	0.651	0.272	0.962	hivrt
0.829	0	0.935	0.829	0.879	0.982	hmga
0.973	0	0.947	0.973	0.96	1	hsp90
0.744	0.005	0.627	0.744	0.681	0.989	inha
0.867	0.009	0.163	0.867	0.274	0.933	mr
0.714	0	0.972	0.714	0.824	0.998	na
0.89	0.003	0.955	0.89	0.921	0.993	p38
0.829	0	0.906	0.829	0.866	0.989	parp
0.75	0.006	0.629	0.75	0.684	0.977	pde5
0.506	0.005	0.699	0.506	0.587	0.957	pdgfrb
0.86	0	1	0.86	0.925	0.981	pnp
0.906	0.003	0.786	0.906	0.842	0.984	ppar_gamma
0.593	0	1	0.593	0.744	0.927	pr
1	0.001	0.8	1	0.889	1	rxr_alpha
0.939	0.001	0.838	0.939	0.886	0.998	sahh
0.038	0.003	0.231	0.038	0.065	0.954	src
0.153	0.007	0.18	0.153	0.165	0.972	thrombin
0.864	0.001	0.731	0.864	0.792	0.997	tk
0.388	0.006	0.306	0.388	0.342	0.986	trypsin
0.659	0.016	0.341	0.659	0.45	0.941	vegfr2
0.875	0.018	0.975	0.875	0.922	0.99	none
0.813	0.01	0.864	0.813	0.828	0.986	

For each of the targets, several statistical characteristics are presented: True Positive (TP) rate, False Positive (FP) rate, Precision, Recall, F-Measure, and ROC Area. One can see that different targets are characterized by rather different performance of recognition. For example, models for *dhfr* and *gart* ligands are very strong, whereas those for *pr*, *hivrt* and *hivpr* are not so good. For each of the targets, individual ROC curves can be built and Cost/Benefit analysis can be performed.

Exercise: Perform *Cost/Benefit* analysis for the *ace* target and compare its results with the case of the two-class classification Naïve Bayes model obtained in Exercise 2

4. Exercise 8: Building a Joint Classification Tree for 40 Targets

In this exercise we will show how decision trees can be applied for building a multi-class model capable of predicting affinity to 40 pharmaceutically important protein biotargets and how the resulting decision tree can be visualized.

- In the ***classifier*** frame, click ***Choose***, then select the *J48* method from the *trees* submenu.
- Click on the ***Start*** button.

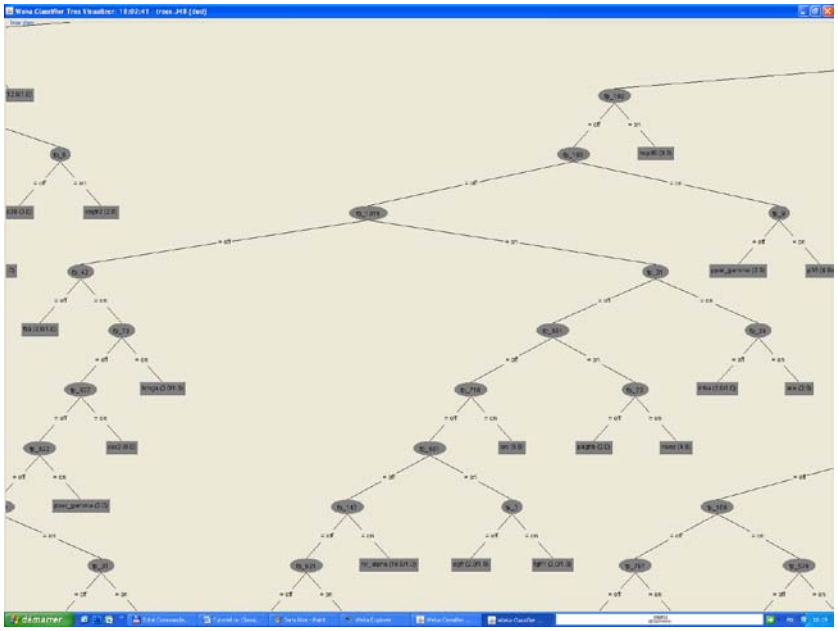
The global statistics of the multi-class classification model is as follows:

```
Correctly Classified Instances        6046               85.2991 %
Incorrectly Classified Instances      1042               14.7009 %
Kappa statistic                          0.813
```

So, the classification tree model is characterized with better statistical characteristics than the Naïve Bayes one (compare with the previous exercise). Now visualize the joint classification tree.

- Click the right mouse button on the model type *trees.J48* in the ***Result list*** frame and select the menu item *Visualize tree*.
- Resize a new window with graphical representation of the tree
- Clock with the right mouse button to the space in this screen, and in the popup menu select the item *Auto Scale*.

The graphical representation of the tree appears to be very big and cannot accommodate into the window. So use scroll bars to scroll it inside the window.



5. Exercise 9: Building the Multi-Class Random Forest Model

In this exercise we will show how the Random Forest method can be applied for building a multi-class model capable of predicting affinity to 40 pharmaceutically important protein biotargets.

- In the ***classifier*** frame, click ***Choose***, then select the *J48* method from the *trees* submenu.
- Click with the left mouse button on the word *RandomForest* in the ***Classifier*** frame. The window for setting options for the Random Forest method pops up.
- The global statistics of the multi-class classification model is as follows:

```
Correctly Classified Instances        6298               88.8544 %
Incorrectly Classified Instances       790               11.1456 %
Kappa statistic                          0.8559
```

We can see that the Random Forest method provides the strongest multi-class classification models.

Exercise: Perform Cost/Benefit analysis for the *ace* target and compare its results with the case of the two-class classification Random Forest model obtained in Exercise 6.

Appendix

1. Notes for Windows

On Windows, Weka should be located on the usual program launcher, in a folder *Weka-version (e.g., **weka-3-6-2***).

It is recommended to associate Weka to ARFF files. Thus, by double clicking an ARFF, Weka/Explorer will be launched and the default directory for loading and writing data will be set to the same directory as the loaded file. Otherwise, the default directory will be Weka directory.

If you want to change the default directory for datasets in Weka, proceed as follows:

- Extract from the java archive weka.jar, the weka/gui/explorer/Explorer.props file. It can be done using an archive program such as WinRAR or 7-zip.
- Copy this file in your home directory. To identify your home directory, type the command echo %USERPROFILE% in a DOS command terminal.
- Edit the file Explorer.props with WordPad.
- Change the line InitialDirectory=%c by InitialDirectory=C:/Your/Own/Path

If you need to change the memory available for Weka in the JVM, you need to edit the file RunWeka.ini or RunWeka.bat in the installation directory of Weka (root privilege may be required). Change the line maxheap=128m by maxheap=1024m. You cannot assign more than 1.4Go to a JVM because of limitations of Windows.

2. Notes for Linux

To launch Weka, open a terminal and type:

java -jar /installation/directory/weka.jar.

If you need to assign additional memory to the JVM, use the option -Xm*MemorySize*m, replacing *MemorySize* by the required size in megabytes. For instance to launch Weka with 1024 Mo, type:

java -jar -Xm512m /installation/directory/weka.jar.