Feature Selection with Decision Tree Criterion

Krzysztof Grąbczewski and Norbert Jankowski
Department of Computer Methods
Nicolaus Copernicus University
Toruń, Poland
kgrabcze,norbert@phys.uni.torun.pl

Abstract

Classification techniques applicable to real life data are more and more often complex hybrid systems comprising feature selection. To augment their efficiency we propose two feature selection algorithms, which take advantage of a decision tree criterion. Large computational experiments have been done to test the possibilities of the two methods and to compare their results with other techniques of similar goal.

1. Introduction

Effective and versatile classification can not be achieved by single classification algorithms. They must be hybrid complex models comprising a feature selection stage. Similarly to other data analysis tasks, also in feature selection it is very beneficial to take advantage of different kinds of algorithms.

One of the great features of decision tree algorithms is that they inherently estimate a suitability of features for separation of objects representing different classes. This facility can be directly exploited for the purpose of feature selection.

After presenting the *Separability of Split Value* criterion, we derive two algorithms of ranking features. The methods are applied to a number of publicly available datasets and their results compared to those obtained with some most commonly used feature selection techniques. At the end we present a number of conclusions and future perspectives.

2. SSV based algorithms

One of the most efficient heuristics used for *decision tree* construction is the *Separability of Split Value* (SSV) criterion [3].Its basic advantage is that it can be applied to both continuous and discrete features in such a manner that the estimates of *separability* can be compared regardless the substantial difference in types.

The *split value* is defined differently for continuous and symbolic features. For continuous features it is a real number and for symbolic ones it is a subset of the set of alternative values of the feature. The *left side* (LS) and *right side* (RS) of a split value s of feature f for a given dataset D are defined as:

$$\mathsf{LS}_{s,f}(D) = \begin{cases} \{x \in D : f(x) < s\} & \text{if } f \text{ is continuous} \\ \{x \in D : f(x) \notin s\} & \text{otherwise} \end{cases}$$
$$\mathsf{RS}_{s,f}(D) = D - \mathsf{LS}_{s,f}(D)$$

where $f(x)$ is the f's feature value for the data vector x. The definition of the *separability of split value* s is:

$$\mathsf{SSV}(s,f,D) = 2 \cdot \sum_{c \in C} |\mathsf{LS}_{s,f}(D_c)| \cdot |\mathsf{RS}_{s,f}(D - D_c)| - \sum_{c \in C} \min(|\mathsf{LS}_{s,f}(D_c)|, |\mathsf{RS}_{s,f}(D_c)|)$$

where C is the set of classes and D_c is the set of data vectors from D assigned to class $c \in C$.

Originally the criterion was used to construct decision trees with recursive splits. In such an approach, current split is selected to correspond to the best split value among all possible splits of all the features (to avoid combinatorial explosion in the case of discrete features with numerous symbols the analysis is confined to a relatively small number of subsets). The subsequent splits are performed as long as any of the tree nodes can be sensibly split. Then the trees are pruned to provide possibly highest generalization (an inner cross validation is used to estimate the best pruning parameters).

2.1. SSV criterion for feature selection

The SSV criterion has been successfully used not only for building classification trees. It proved efficient in data type conversion (from continuous to discrete [2] and in the

opposite direction [4]) and as the discretization part of feature selection methods, which finally rank the features according to such indices like Mutual Information [1].

Decision tree algorithms are known for the ability of detecting the features that are important for classification. Feature selection is inherent there, so it is not so necessary at the data preparation phase. Inversely: the trees' capabilities can be used for feature selection.

Feature selection based on the SSV criterion can be designed in different ways. From the computational point of view, the most efficient one is to create feature ranking on the basis of the maximum SSV criterion values calculated for each of the features, for the whole training dataset. The cost is the same as when creating decision stubs (single-split decision trees). However, when the classification task is a multiclass one, separabilities of single splits can not reflect the actual virtue of the features. Sometimes two or three splits may be necessary to prove that the feature can accurately separate different classes. Thus it is reasonable to reward such features, but some penalty must be introduced, when multiple splits are being analyzed. These ideas brought the following algorithm:

Algorithm 1 (Separate feature analysis)

- ► ***Input:*** *Training data (a sample X of input patterns and their labels Y).*
- ◄ ***Output:*** *Features in decreasing order of importance.*

1. *For each feature f*
   (a) *$T \leftarrow$ the* SSV *decision tree built for one-dimensional data (feature f only).*
   (b) *$n \leftarrow$ the number of leaves in T.*
   (c) *For $i = n - 1$ downto 1*
      i. *$s_i \leftarrow \frac{\mathsf{SSV}(T)}{\log(2+i)}$, where $\mathsf{SSV}(T)$ is the sum of the* SSV *values for all the splits in T.*
      ii. *Prune T by deleting a node N of minimum error reduction.*
   (d) *Define the rank $\mathcal{R}(f)$ of feature f as the maximum of the s_i values.*
2. *Return the list of features in decreasing order of $\mathcal{R}(f)$.*

Another way of using SSV for feature ranking is to create a single decision tree and *read* feature importance from it. The filter we have used here is the algorithm 2.

Algorithm 2 (Feature selection from single SSV tree)

- ► ***Input:*** *Training data $\langle X, Y \rangle$.*
- ◄ ***Output:*** *Features in decreasing order of importance.*

1. *$T \leftarrow$ the* SSV *decision tree built for $\langle X, Y \rangle$.*
2. *For each non-leaf node N of T, $G(N) \leftarrow$ the classification error reduction of node N.*
3. *$\mathcal{F} \leftarrow$ the set of all the features of the input space.*
4. *$i \leftarrow 0$*
5. *While $\mathcal{F} \neq \emptyset$ do:*
   (a) *For each feature $f \in \mathcal{F}$ not used by T define its rank $\mathcal{R}(f) \leftarrow i$. Remove these features from $\mathcal{F}$.*
   (b) *Prune T by deleting all the final splits of nodes N for which $G(N)$ is minimal.*
   (c) *Prune T by deleting all the final splits of nodes N for which $G(N) = 0$.*
   (d) *$i \leftarrow i + 1$*
6. *Return the list of features in decreasing order of $\mathcal{R}(f)$.*

This implements a „full-featured" filter – the decision tree building algorithm selects the splits locally, i.e. with respect to the splits selected in earlier stages, so that the features occurring in the tree, are complementary. It is important to see that despite this, the computational complexity of the algorithm is very attractive (contrary to wrapper methods of feature selection).

The ranking generated by the algorithm 2 can be used for feature selection either by dropping all the features of rank 0 or by picking a given number of top-ranked features.

In some cases the full classification trees use only a small part of the features. It does not allow to select any number of features – the maximum is the number of features used by the tree, because the algorithm gives no information about the ranking of the rest of the features.

3. Algorithms used for the comparison

Numerous entropy based methods and many other feature relevance indices are designed to deal with discrete data. Their results strongly depend on the discretization method applied before calculation of feature ranks [1]. This makes reliable comparisons of ranking algorithms very difficult, so here we compare the results obtained with methods, which do not suffer from such limitations. We have chosen two simple but very successful methods based on the *Pearson's correlation coefficient* and a *Fisher–like criterion*. Another two methods being the subject of the comparisons belong to the Relief family.

The Pearson's correlation coefficient detects linear correlation. Thus, used for feature selection it may provide poor results in the case of nonlinear dependencies.Nevertheless in many applications it seems optimal because of very good results and computational simplicity.

The Fisher-like criterion is the *F-score* defined as

$$\frac{m_0 - m_1}{s_0 + s_1}, \tag{1}$$

where m_i is the mean value of the feature calculated for the elements of i-th class, and s_i is the corresponding standard

deviation. Such criterion can be used only in the case of binary classification. For multiclass data our estimate of feature relevance was the maximum of the "one class vs the rest" ranks.

The two representatives of the Relief family of algorithms are the basic Relief method [6] and its extension by Kononenko known as ReliefF [7]. The basic algorithm randomly selects m vectors and for each (vector V) of them finds the nearest *hit* H and *miss* M (neighbors belonging to the same and different class respectively) – then the weight of each feature f is changed according to

$$W_f \leftarrow W_f - \frac{1}{m}\mathrm{diff}_f(V,H) + \frac{1}{m}\mathrm{diff}_f(V,M), \quad (2)$$

where diff_f, for numerical feature f, is the normalized difference between the values of f and for nominal f it is the truth value of the equality of given values.

The ReliefF algorithm differs from the basic version in that it takes k nearest hits and k nearest misses of each class in a generalization of formula (2). All our calculations used $k = 10$.

The m parameter was selected to be less than or equal to 500. For smaller datasets we used each training vector once instead of m randomly selected vectors.

Most of the calculations time was spent on running the Relief methods, which are most complex of the six methods being analyzed – all the complexities are given in table 1.

Method	Complexity
CC	fn
F-score	fn
Relief	$m(n+f)$
ReliefF	$m(n\log k + fk)$
SSV1	$fn\log n$
SSV2	$fn\log n$

Table 1. Feature selection methods complexities (n – vectors count, f – features count)

4. Testing methodology

The algorithms resulting in a feature ranking can be compared only in the context of particular classifiers. Different kinds of classification learning methods may broaden the scope of the analysis, so we have applied *Naive Bayesian Classifier*, k *Nearest Neighbors* (kNN) method with Euclidean and Manhattan distance metrics and *Support Vector Machines* (SVM) with Gaussian and linear kernels (all our SVMs were trained with parameters C=1 and bias=0.1).

All datasets were standardized before the tests, however it must be pointed out, that none of the feature ranking methods tested here is sensitive to data scaling – the standardization affects only kNN and SVM classification methods. Moreover, Naive Bayes and SSV based selections directly deal with symbolic data, so in such cases, the standardization does not concern them at all.

Drawing reliable conclusions requires appropriate testing strategy and appropriate analysis of the results. The major group of our tests were 10-fold cross-validation (CV) tests repeated 10 times to obtain good approximation of the average result. In each fold of the CV tests we ran six feature ranking algorithms: CC based ranking, F-score based ranking, basic Relief, ReliefF, and two our methods presented as algorithms 1 (SSV1) and 2 (SSV2). The six rankings were analyzed to determine all the feature subsets generated by selecting a number of top-ranked features. Then for each such subset the classification algorithm was applied and results collected (these are the results averaged by the 10 repetitions of 10-fold CV). Such strategy may require enormous amount of calculations. For example: when we tested SVM models on features selected from a 60 features dataset, we needed to solve about 320 classification tasks at each CV pass (there are 6*60 = 360 possible selections – some feature subsets are always repeated). If we deal with a 3-class task we need to build 3 SVM models for each CV pass ("one class vs the rest" technique), which implies 3*320 = 960 SVM runs. We repeat 10 times the 10-fold CV, so finally we need to build almost 100 000 SVM models – such calculations took more than 100 hours of (one half of) a 3.6GHz Pentium 4 HT.

To determine the statistical significance of the differences in the results we used the t-test (or paired t-test) methodology with $p = 0.05$.

5. Results

To make thorough analysis possible, feature selection methods must be applied to datasets defined in a space of not too large dimensionality, but on the other hand too small dimensionality would make feature selection tests baseless. Thus we decided to use the higher-dimensional classification tasks data available in the UCI repository [8]. Table 2 presents the datasets we have used for the tests, however because of the space limits we discuss just a representative part of the results – the figures for all the datasets can be seen at http://www.phys.uni.torun.pl/∼kgrabcze/papers/05-sel-figures.pdf.

The datasets distributed in separate training and test files had been merged and CV tests performed for the whole data.

The figures present average classification accuracies (10 repetitions of 10-fold CV) obtained with different classi-

dataset	features	vectors	classes
chess	36	3196	2
glass	9	214	6
image segmentation	19	2310	7
ionosphere	34	351	2
mushroom	22	8124	2
promoters	57	106	2
satellite image	36	6435	6
splice	60	3190	3
thyroid	21	7200	3

Table 2. Datasets summary

fiers trained on data after given number of top-ranked features were selected (the rankings were generated for each of the CV folds independently).

Ionosphere data (Figure 1). The results obtained with Naive Bayesian classifier for the ionosphere data are unique. The SSV2 algorithm selecting all the features used by SSV decision trees leads to much higher accuracy (89.87%) than obtainable with the other methods. Please notice that the SSV2 algorithm can sensibly rank only the features which occur in the tree, thus we can not reliably select more features than there are in the tree – the figures reflect this fact by constant values of the SSV2 series for the numbers of features exceeding the sensible maximum.

5NN classifier with Euclidean metric gives maximum accuracy (89.95%) when 4 features are selected with CC or F-Score, but a very high result (88.32%) can be obtained for just 2 features selected with SSV2 (the accuracy is significantly better than for the whole dataset – 84.65%). 5NN with Manhattan distance yields maximum 91.11% accuracy for 10 features selected with SSV2.

It is interesting that the SSV1 selection with SVM classifier works poorly for small numbers of features, but is one of the fastest to reach the accuracy level of SVM trained in the whole data. Although feature selection can not significantly improve the accuracy, it may be very precious because of the dimensionality reduction it offers.

Promoters data (Figure 2). This is an example of a dataset for which almost each selection method improves the accuracy of each classifier and the improvement is very significant. The exceptions are the combinations of the Naive Bayes classifier with the Relief and SSV2 feature selection methods.

Please, notice that the maximum accuracy is obtained usually for the number of features between 3 and 5, so the selection is crucial here.

Splice data (Figure 3). The conclusions for this dataset are very similar to those of the promoters data. Feature selection provides very significant improvement of 5NN and

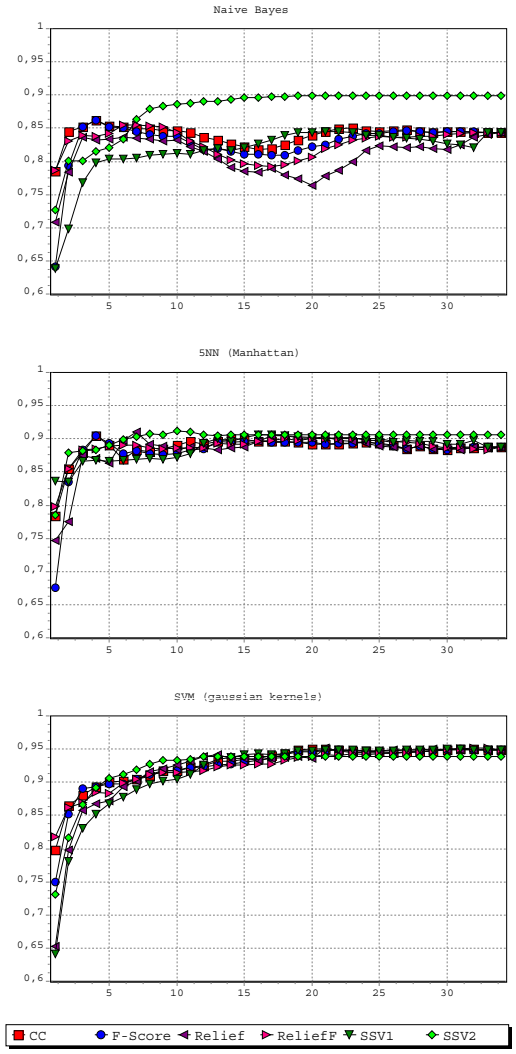

Figure 1. Results for the ionosphere data.

SVM classifiers. For Naive Bayes we get a slight improvement but also statistically significant – the best results are within the range of 30-40 features and are obtained with ReliefF (96.2% accuracy with standard deviation of 0.14%), and SSV1 (96.11% $\pm$ 0.14%), while with no selection we achieve 95.66% $\pm$ 0.08%. The results show less variance than in the case of promoters (this is due to the larger training data samples). Hence even small differences can be statistically significant here.

Other data sets (Figure 4). Image segmentation and thyroid data are the examples of classification tasks, where decision trees show best adaptation abilities. It is consistent with the fact that for these datasets the SSV2 selector is most useful also for other classification algorithms.

5NN applied to image segmentation achieves its best

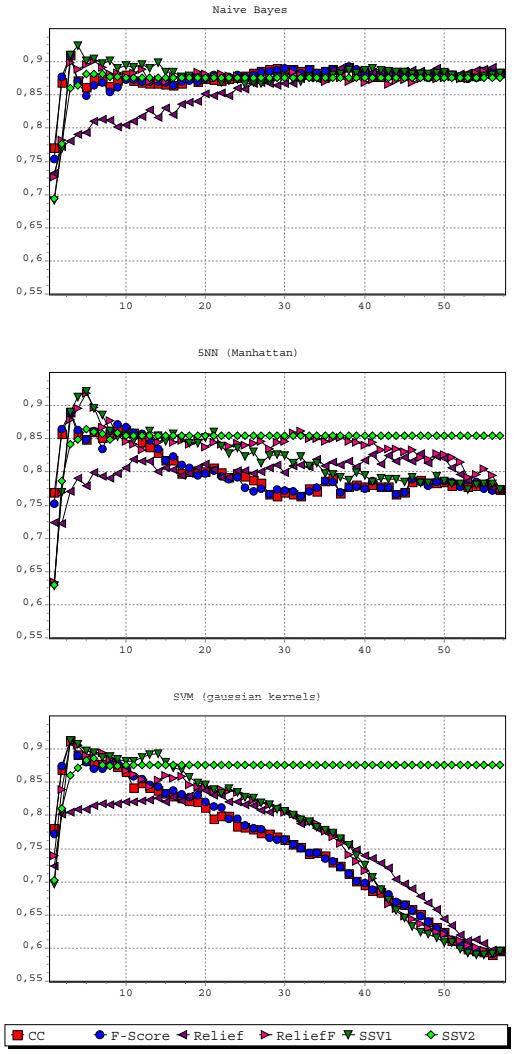

Figure 2. Results for the promoters data.

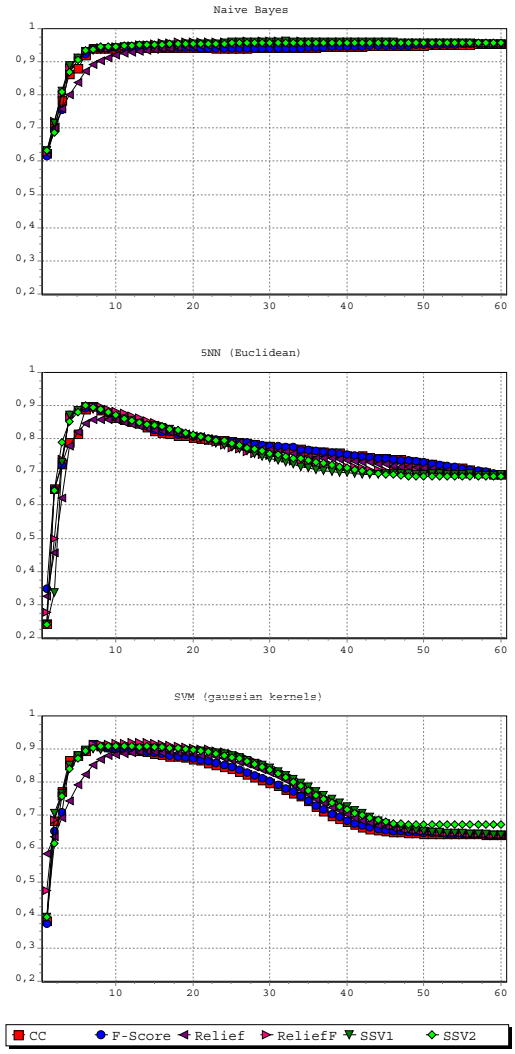

Figure 3. Results for the splice data.

result (96.43% $\pm$ 0.22%) after selecting 11 features with SSV2, which is significantly better result than with any other feature selection method, we have tested. 3 features are enough to get 96.28% $\pm$ 0.13%.

The plot showing the results of 5NN with Euclidean measure, applied to the thyroid data shows clearly the advantage of SSV2 selector (its ability to detect dependencies between features) – selecting 3 features yields a high accuracy of very high stability (98.54% $\pm$ 0.04%), specific for decision trees in the case of this dataset.

The remaining two plots of figure 4 demonstrate some interesting negative results. For the satellite image data CC based selection loses any comparisons, though proves very valuable in many other tests. It is also an exceptional dataset, where ReliefF performs worse than Relief. For the chess data, SSV1 selection methods is not in the same league with others. Also the F-Score and CC methods can not improve the accuracy of SVM while SSV2, Relief and ReliefF offer significant improvements.

6. Conclusions

We have presented two feature selection methods based on the SSV criterion and experimentally confirmed that they are a very good alternative to the most popular methods. On our way we have shown several examples of spectacular improvement obtained by means of feature selection.

Feature selection algorithms can not be estimated without the context of final models. We must realize that different classifiers may require different feature selectors to do their best, however in general we observe that for a particular dataset, a feature selectors performing good (or bad)

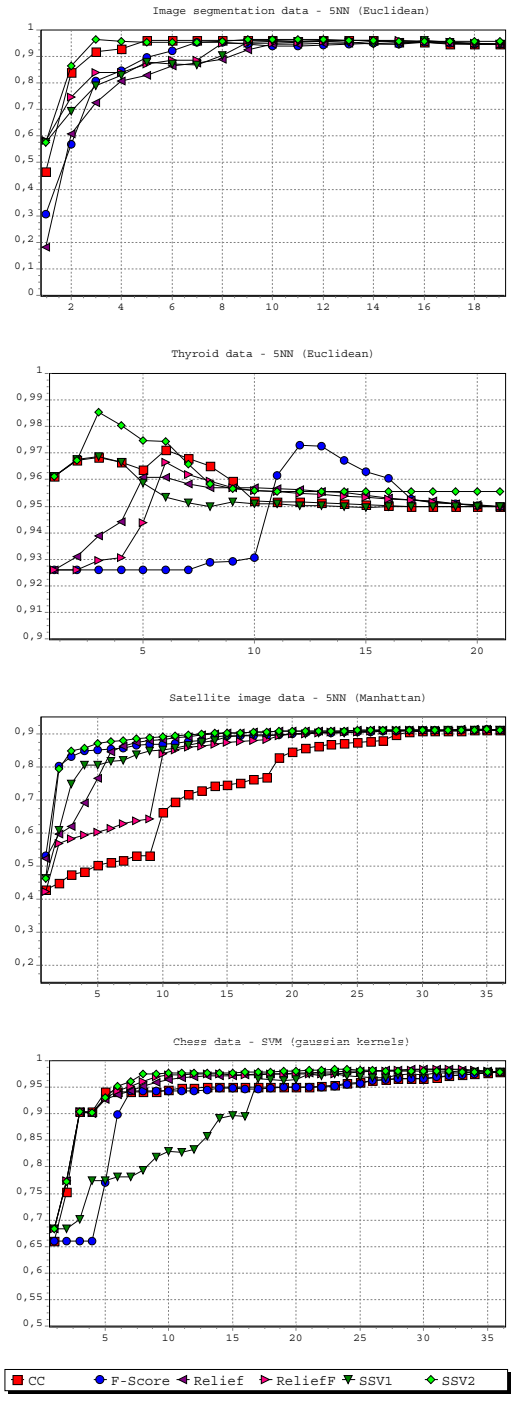

Figure 4. Other interesting results.

for some classifier, perform good (or bad respectively) with other classifiers. In accordance with the no-free-lunch theorem, there is no single feature selection method performing very good for any dataset. Thus effective feature selection requires a validation stage to confirm methods usability in a given context. Such validation can be expensive, so we must be careful. To reduce the complexity of the search for optimal feature sets we may find some regularities and use them as heuristics (for example, we see that SSV2 is usually very good feature selector for the datasets, where decision tree algorithms provide high accuracies).

Further research. The comparison presented here treats each of the feature selection methods separately. It is a natural step forward to think about efficient algorithms to combine the results of different feature selection methods. Some simple methods have already been used [5] but deeper meta-level analysis should bring many interesting conclusions and more versatile feature selection methods.

Acknowledgements. The research is supported by the Polish Ministry of Science – a grant for years 2005–2007.

References

[1] W. Duch, J. Biesiada, T. Winiarski, K. Grudziński, and K. Grąbczewski. Feature selection based on information theory filters and feature elimination wrapper methods. In *Proceedings of the Int. Conf. on Neural Networks and Soft Computing (ICNNSC 2002)*, Advances in Soft Computing, pages 173–176, Zakopane, 2002. Physica-Verlag (Springer).

[2] K. Grąbczewski. SSV criterion based discretization for Naive Bayes classifiers. In *Proceedings of the 7th International Conference on Artificial Intelligence and Soft Computing*, Zakopane, Poland, June 2004.

[3] K. Grąbczewski and W. Duch. A general purpose separability criterion for classification systems. In *Proceedings of the 4th Conference on Neural Networks and Their Applications*, pages 203–208, Zakopane, Poland, June 1999.

[4] K. Grąbczewski and N. Jankowski. Transformations of symbolic data for continuous data oriented models. In *Artificial Neural Networks and Neural Information Processing – ICANN/ICONIP 2003*, pages 359–366. Springer, 2003.

[5] K. Grąbczewski and N. Jankowski. Mining for complex models comprising feature selection and classification. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature extraction, foundations and Applications*. Springer, 2005.

[6] K. Kira and L. A. Rendell. A practical approach to feature selection. In *ML92: Proceedings of the ninth international workshop on Machine learning*, pages 249–256, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[7] I. Kononenko. Estimating attributes: Analysis and extensions of RELIEF. In *European Conference on Machine Learning*, pages 171–182, 1994.

[8] C. J. Merz and P. M. Murphy. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.