# Feature Selection for high-dimensional data with *RapidMiner*

Benjamin Schowe
Technical University of Dortmund
Artificial Intelligence Group
benjamin.schowe@tu-dortmund.de

**Abstract**

The number of recorded feature has grown exponentially over the last years. In the bioinformatics domain datasets with hundreds of thousands of features are no more unusual. Extracting knowledge from such huge heaps of data demands for new methods. Traditional methods that are applicable for low dimensional data like wrapper feature selection can no longer handle the growing number of features. We present an extension to *RapidMiner* containing feature selection and classification algorithms suited for the high-dimensional setting overcoming the limitations of traditional feature selection approaches.

## 1 Introduction

High-dimensional data are a challenge for data miners for several reasons.

- The more complex models are, the harder they are to comprehend and communicate. It is much easier to tell a biologist that from some tens of thousands of genes these 10 are important to the disease he is tackling than to explain the influence of the 10.000th feature in a Support Vector Machine [5].
- The number of examples $n$ to comprehensively describe a $p$-dimensional space grows exponentially in $p$ [2].
- With high dimensionality often comes high variance, challenging the stability and such meaning of feature selections[13, 8].
- More dimensions mean more data. More data means longer runtime. Instead of sampling an losing information this can be tackled from the

feature side by neglecting those feature which contain no information concerning the learning task at hand.

For large $p$ conventional wrapper selection methods like *Forward-* or *Backward-Selection* [9] or evolutionary methods [12] are computationally infeasible. Furthermore, they tend to overfit to the used learning scheme [14]. This can improve the performance of the learner used for feature subset evaluation, but is not useful for presenting the set of "solely relevant" features to practitioners or clients. In this paper we present an extension to *RapidMiner*, which provides feature selection algorithms suitable for the high-dimensional and high-volume (large $p$, large $n$) setting.

This paper is built up as follows. Section 4 describes feature selection with linear models. Some very fast filter approaches are shown in Sections 2 and 3. Then Section 5 shows how one can get more stable selections in *RapidMiner*. We show some experiments illustrating the benefits of feature selection in Section 7 and conclude in Section 8.

**Notation** Operator names are printed in bold face. Throughout the paper we will be using the term feature as synonym to attribute and variable. Let $X$ denote the set of all features and $x \in X$ a single feature. The index $i$ allows instance-wise indexation such that $x_i$ means the values of feature $x$ of the $i$th example. In contrast index $j$ stands for feature-wise indexation such that $x_j$ is the $j$th feature. Last, $\mathbf{x}_i$ denotes the $p$-dimensional vector of the $i$th example.

# 2 Filter Methods

The fastest way for feature selection is most probably ranking the features with some statistical test and selecting the $k$ features with the highest score or those with a score greater than some threshold $t$ [5, 14] . Such univariate filters do not take into account feature interaction, but they allow a first inspection of the data and sometimes provide reasonable results. And, most of all, they are fast. Univariate filter methods usually work in $\Theta(n \cdot p)$.

## 2.1 Significance Analysis for Microarrays

For the very high-dimensional problem of analyzing microarray-data [18] suggests scoring the genes with the SAM statistic or relative difference $d(x)$ which is defined as

$$d(x) = \frac{\bar{x}_+ - \bar{x}_-}{s_0 + \sqrt{\frac{1/n_x + 1/n_-}{n_+ + n_- - 2}\left(\sum_{i+}(x_i - \bar{x}_+)^2 + \sum_{i-}(x_i - \bar{x}_-)^2\right)}} \tag{1}$$

$i_{+/-}$ denotes the indices and $\bar{x}_{+/-}$ denotes the mean of all examples belonging to the positive/negative class and $s_0$ is a small correctional parameter controlling the influenece of variance. This function is implemented in the **Weight by SAM**-operator.

## 2.2 Welch test

Another statistical test for measuring significant differences between the mean of two classes, the Welch-test, is defined as

$$w(x) = \frac{\bar{x}_+ - \bar{x}_-}{\sqrt{\frac{\sum_{i+(x_i-\bar{x}_+)^2}}{n_+} + \frac{\sum_{i+(x_i-\bar{x}_-)^2}}{n_-}}} \tag{2}$$

The **Weight by Welch-test**-operator[1] computes for each feature a $p$-value for the two-sided, two-sample Welch-test. It does not assume subpopulation variances are equal. Degrees of freedom are estimated from the data.

## 2.3 Other filters

Whether a scoring function is applicable to a feature depends on whether the feature and label are numerical or nominal. For a continuous feature $X$ and a nominal label $Y$ with $C$ classes the F-test score is defined as

$$F(x,y) \quad = \quad \frac{(n-C)\sum_c n_c(\bar{x_c}-\bar{x})^2}{(C-1)\sum_c (n_c-1)\sigma_c^2} \tag{3}$$

with per-class-variance $\sigma_c^2$ and $n_c$ the number of examples in class $c \in \{1,..,C\}$. It reflects the ratio of the variance between classes and the average variance inside theses classes.

A linear dependency between two numerical features can be scored by Pearson's linear correlation

$$R(x,y) \quad = \quad \frac{Cov(x,y)}{\sqrt{Var(x)\cdot Var(y)}} \tag{4}$$

respectively its estimate

$$r(x,x) \quad = \quad \frac{\sum_i (x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_i (x_i-\bar{x})^2 \sum_i (y_i-\bar{y})^2}}. \tag{5}$$

[1]The operator **Weight by Welch-Test** has been implemented by Miriam Bützken.

A possible measure for the dependency between two nominal features is the *mutual information*

$$MI(x,y) = \sum_{l,m} P(x_l, y_m) \log_2 \frac{P(x_l, y_m)}{P(x_l)P(y_m)} \tag{6}$$

where the $x_l$ and $y_m$ are the possible nominal values of $x$ and $y$.

The operator **Weight by Maximum Relevance** scores features according to those three above mentioned functions. It automatically chooses the function which matches the types of features involved. One has to note for datasets with mixed nominal and numerical features that those functions map to different scales. So it is wise to first transform all features to the same domain [3, 7].

# 3 Hybrid Multivariate Methods

A group of new algorithms has come up to bridge the gap between fast but univariate filters on the one hand, and slow but multivariate wrappers on the other hand. Their goal is to find a subset of features which is highly predictive with no or a minimum of redundant information.

## 3.1 Minimum Redundancy Maximum Relevance

The *correlation based feature selection* (CFS) [7] and *minimum Redundancy Maximum Relevance feature selection* [3] perform a *sequential forward search* with a correlation based or information theoretic measure in the evaluation step. They iteratively add to a set $F$ the best feature according to a quality criterion $Q$:

$$F_{j+1} = F_j \cup \arg\max_{x \in X \setminus F_j} Q(f) \tag{7}$$

where $Q$ is either the difference

$$Q_{MID} = Relevance(x,y) - \frac{1}{j} \sum_{x' \in F_j} Redundancy(x,x') \tag{8}$$

or the ratio between relevance and average pairwise redundancy of $x$ given the already selected features $x' \in F_j$:

$$Q_{MIQ} = \frac{Relevance(x,y)}{\frac{1}{j} \sum_{x' \in F_j} Redundancy(x,x')} \tag{9}$$

The *Relevance*$(\cdot,\cdot)$ and *Redundancy*$(\cdot,\cdot)$ functions automatically map to linear correlation (eq. 5), the F-test-score (eq. 3) or the mutual information (eq. 6) depending on the types of features involved (nominal/numerical). The operator **Select by MRMR / CFS** implements this algorithm. Additionally, it has the possibility to give stabilized selection results by applying a fast ensemble technique [15]. It repeats the selection process $e$ times to decrease the results variance. Runtime is prevented from being multiplied by $e$ by splitting the relevance and redundancy measures into blocks and clever use of caching.

The $Q_{MID}$ and $Q_{MIQ}$ criteria can also be used with other search strategies. The operator **Performance (MRMR)** allows to evaluate any feature set inside a **Optimize Selection**-loop. But in contrast to the single criteria MID and MIQ and those in the **Performance (CFS)**-operator, the **Performance (MRMR)** also deliver relevance and redundancy as two separate criteria allowing for multi-objective optimization of the feature subset selection. To speed up the computation, a **MRMR-Cache**-object can be created via the **MRMR Cache Creator** directly before the optimization loop. If two features are compared in different iterations of the loop their correlation or mutual information has only to be computed once. This provides significant speed-up.

# 4 Selection with Linear Models

For linear models with standardized features the absolute values of the entries in the coefficient vector $\boldsymbol{\beta}$ reflect a feature's importance. From a feature selection point of view those features with zero entries are not selected. If the linear model does not provide such zero entries one could discard those features with very small absolute values.

## 4.1 Recursive Feature Elimination

The $L_2$-regularized *Support Vector Machine* (SVM) [19], which minimizes

$$argmin_{\boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{\beta}||_2^2 + C\sum_{i=1}^{n}\xi_i \qquad s.t. \qquad y_i(\langle\boldsymbol{\beta}, \mathbf{x_i}\rangle + \boldsymbol{\beta}_0) \geq 1 - \xi_i, \qquad (10)$$

tends to distribute the weight equally among correlated features. Hence, setting all small values to zero simultaneously could eliminate two important correlated features which had to share their influence due to their correlation. A technique for tackling this problem is *Recursive Feature Elimination* (SVM-RFE) [6]. Instead of discarding all features with small influence $\boldsymbol{\beta}_j < t$ at once or discarding all but the largest $|\boldsymbol{\beta}_j|$, SVM-RFE works in an iterative way:

1. A linear SVM is trained on all remaining features, yielding $\boldsymbol{\beta}$

| Rank | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1st round | $x_1$ | $x_{14}$ | $x_5$ | $x_{15}$ | $x_{16}$ | $x_9$ | $x_{10}$ | $x_7$ | $x_{17}$ | $x_{20}$ | $x_3$ | $x_8$ | $x_{19}$ | $x_{11}$ | $x_{12}$ | $x_4$ | $x_2$ | $x_{13}$ | $x_{18}$ | $x_6$ |
| 2nd round | $x_1$ | $x_5$ | $x_{14}$ | $x_{16}$ | $x_{15}$ | $x_8$ | $x_{19}$ | $x_9$ | $x_{10}$ | $x_{17}$ | $x_3$ | $x_{11}$ | $x_7$ | $x_{20}$ | | | | | | |
| 3rd round | $x_1$ | $x_{14}$ | $x_5$ | $x_{15}$ | $x_{16}$ | $x_8$ | $x_{19}$ | $x_9$ | $x_{10}$ | | | | | | | | | | | |
| 4th round | $x_1$ | $x_5$ | $x_{14}$ | $x_{16}$ | $x_8$ | $x_{15}$ | | | | | | | | | | | | | | |
| 5th round | $x_1$ | $x_{14}$ | $x_{16}$ | $x_5$ | | | | | | | | | | | | | | | | |

Table 1: Changing ranks of features $x_j$ in each iteration of SVM-RFE

2. The fraction $r$ or fixed number $c$ of features with smallest $|\boldsymbol{\beta}_j|$ is discarded.

3. If only $k$ features are left finish, else goto 1.

This recursive feature elimination scheme can make use of any multivariate model producing a $\boldsymbol{\beta}_j$ for each feature.

Table 1 shows an example run clarifying the algorithms behavior and benefit. For example $x_8$ would have been discard if only $|\beta_8|$ of the first SVM run had been considered. But as it does not belong the fraction of features removed in the first round stays in the feature set. In the first round perhaps some features were removed which share some information with $x_8$. As this information is now only covered by $x_8$ it receives a higher ranking. It turns out that $x_8$ is a rather important features and results among the top six features.

Due to its popularity in the bioinformatics community and good applicability to the $p \gg n$-scenario we have implemented two RFE operators in *RapidMiner*. The operator **Select by Recursive Feature Elimination with SVM** is a hard wired implementation of SVM-RFE with a linear *jMySVM* inside. The $C$ parameter is fixed for all iterations. If one wants to optimize the $C$ parameter for each round inside RFE or make use of another multivariate weighting, one can use the operator **Recursive Feature Elimination**. It contains a subprocess which can be filled with any (chain of) operator(s) producing an *Attribute Weights* object.

## 4.2 Least Angle Regression

The $L_1$-regularized *Least Absolute Selection and Shrinkage Operator* (LASSO) [16] yields a sparser coefficient vector $\boldsymbol{\beta}$ with many zero entries. The optimization function of the LASSO is

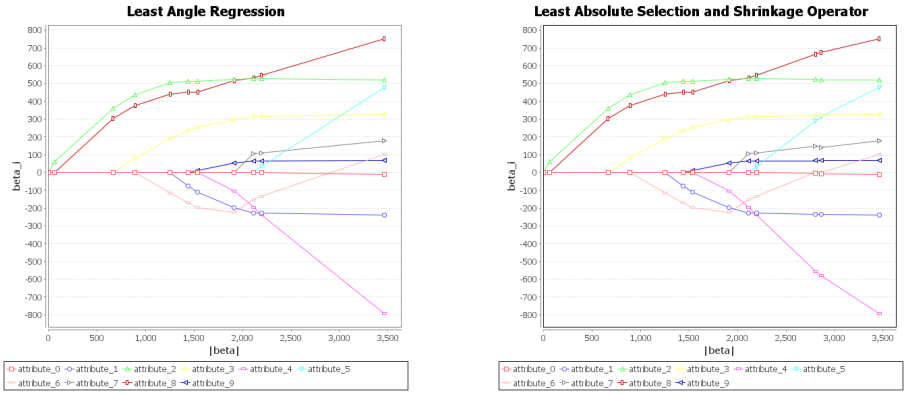
$$min_{\boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2 \tag{11}$$

under the condition that

$$\sum_{j=1}^{p} |\boldsymbol{\beta}_j| \leq t \tag{12}$$

As there is no offset or bias $\boldsymbol{\beta}_0$ the input data have to be standardized to zero mean.

In [4] the *Least Angle Regression* (LARS) algorithm was introduced which provides a stepwise regression model. With some small modifications it also delivers a LASSO solution. Starting with $\boldsymbol{\beta} = \mathbf{0}$, LARS stepwise increases those coefficients whose features have in each iteration the highest correlation with the target until all are non-zero. The **LARS - Least Angle Regression**-operator can deliver such full unconstrained ($t = 0$) solutions and any constrained solution on the iteration process. Fig. 1 shows the development of the coefficients for the unconstrained LARS and the slightly different unconstrained LASSO solution. Unconstrained LARS-models can later be converted




(a) LARS (b) LASSO

Figure 1: Development of the coefficients in the unconstrained ($t = 0$) LARS and LASSO solutions on the Diabetes dataset

into constrained solutions and deliver any $\boldsymbol{\beta}$ for any choice $t$. The operator **LARS - Change Model Parameters** can be used to either change the $t$ parameter or the maximum number of features with non-zero entries in the coefficient vector, see Fig. 2.

# 5 Ensemble-Methods

When selecting a sub-set of features the main goals are to enhance the classification performance or to keep a good performance with a smaller and clearer

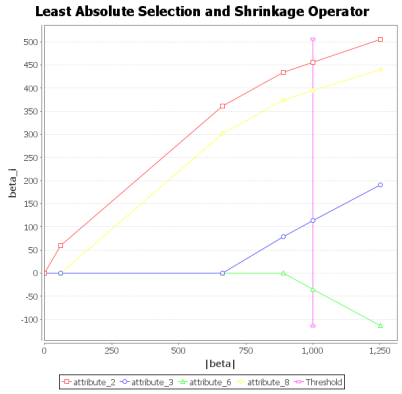



Figure 2: Constrained LASSO solution, $t = 1000$, on the Diabetes dataset

set of features. But most often it is also necessary to provide a feature set which is stable [13, 10, 11, 15]. If small variations in the input data result in major changes in the selected feature sets it is hard to present such a result as the *final* solution.

The stability of a feature selection method can be measured by the similarity of the resulting feature subset generated by a feature selection method on different data drawn from the same basic population. Different stability indices are available for this similarity of sets. The *Jaccard index* of two feature-sets as used by [13] is defined as

$$S_J(F_a, F_b) = \frac{|F_a \cap F_b|}{|F_a \cup F_b|}. \tag{13}$$

A measure very similar to the Jaccard Index was proposed by [10] which we will refer to as Kuncheva's index $S_K$. For two feature subsets of size $k$ it is defined as

$$S_K(F_a, F_b) \quad = \quad \frac{|F_a \cap F_b| \frac{k^2}{p}}{k - \frac{k^2}{p}} \tag{14}$$

where $S_K(F_a, F_b) \in [-1, 1]$. Its advantage over the Jaccard index is that it also regards the size $p$ of the whole feature set and the number $k$ of selected features.

When one wants to analyze the stability of a feature selection scheme, one can put it inside the **Feature Selection Stability Validation**. It repeats the inner selection process on variations (bootstraps or cross-validation-like subsets) of the *Example Set* and compares the resulting feature sets or -weights by Jaccard index and Kuncheva's index or linear correlation.

It has be shown [13, 8, 11, 15] that applying ensemble-methods to the feature selection problem can benefit selection stability and classification performance. For this reason the **Ensemble Feature Selection** meta-operator can be filled with any feature selection scheme. The inner process is the repeatedly applied to variations of the input data, similar to the above operator. The resulting *Attribute Weights* of each iteration of the inner process are then combined to a final *Attribute Weights*-object. There are three ways for combining *Attribute Weights*. The *top-k* method counts, how of a feature was selected or ranked among the top $k$ features. Then the $k$ features with the highest count are returned. The user can also define a threshold to the minimum number of iterations in which a feature has to be selected. The *geq_w* method works similar, but counts how often a feature received a weight $\geq w$. Last, the *accumulate_weights* option simply adds up the weights over all iterations.

# 6 Utility

This section contains some helpful operators which do not implement any particular feature selection or model building algorithm. They serve as shorthands for otherwise lengthy sub-processes or macros or add missing features.

**Select top k features** In order to derive a feature subset from an importance measure like the above mentioned *SAM* score or F-test score one can define the desired size of the subset and only choose the top scored features. For useful application of weighting schemes inside a **Wrapper-Validation** the **Select top k features**-operator takes an *Attribute Weights*-object and sets the top-ranked weights to one and all others to zero. The user can either define a fixed size $k$ of the subset or choose the top $p$ percent.

**Log performance** To get more stable performance estimates in *RapidMiner* one can repeat processes inside a **Loop and Average**. Sadly, this operator allows to log only on single performance value. To overcome this drawback the **Log Performance**-operator attached after a **Loop and Average** can log arbitrary entries in a *Performance Vector*-object along with their mean, variance, standard deviation and number of samples.

**Weights 2 Ranking** Naturally, every feature weighting scheme provides a ranking of the features. This operator just sorts a features according to their weight an replaces the weight with the position in the ordered list. See Table 2 for an example. If also negative weights can mean high influence, e.g. as generated by an SVM, it is possible to use the absolute weights as a sorting criterion.

| Attribute | Weight | Rank |
|---|---|---|
| Outlook | 0.247 | 2 |
| Temperature | 0.178 | 3 |
| Humidity | 1.118 | 1 |
| Wind | 0.048 | 4 |

Table 2: The feature weights and the resulting ranks for the Golf dataset.

**Rank by Selection** Some feature selection operators provide the user with a final set of selected features. Most often the features are selected in an iterative fashion which implies a ranking. In order to extract this implicit ranking, one can place the selection scheme inside the **Rank by Selection** operator. This operator then repeatedly executes its sub-process for selecting $1..k$ features.

**Replace Missing Values (with offset)** This adds a useful feature to *RapidMiner*'s own **Replace Missing Values** operator. The added functionality is that a constant offset can be added to the replenished values. This is for example helpful in miRNA analysis. The value of each miRNA stands for the days it took to reach a certain level of growth. If after $n$ days the experiment has to be finished, there might be probes which have not yet reached that threshold. The number of days needed to reach the threshold is thus unknown. When analyzing this dataset in *RapidMiner* simply replacing the missing values by the maximum ($n$, in this case) does not pay respect to the nature of the experiment. In this case it is useful to add an offset to distinguish the maximum values from the unknown values.

**Nearest Centroid** A very fast and simple classification method is implemented in the **Nearest Centroid**-operator. After selecting the most important features, the centroids for each class are calculated. Each new instance is then classified as belonging to the class of the nearest centroid. This is an un-shrunken variant of the *nearest shrunken centroid method* [17]

# 7 Experiments

Just to visualize the benefits of feature selection Fig. 3 compares a rather complicated *Random Forest* model trained on all features to the two very simple learners *Naive Bayes* and *1-Nearest-Neighbor* each combined with the *MRMR feature selection*. Horizontal lines show accuracy on all features without selection.

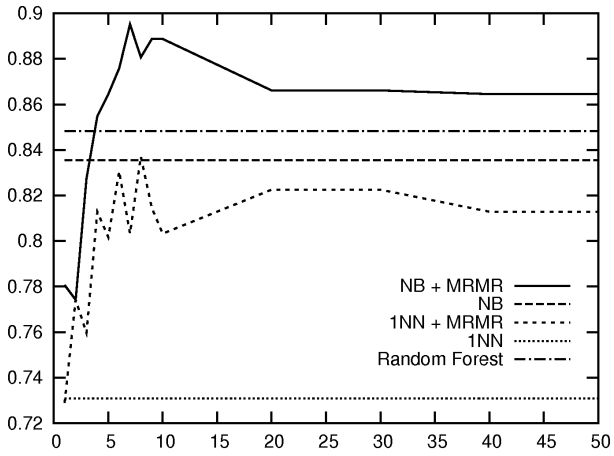
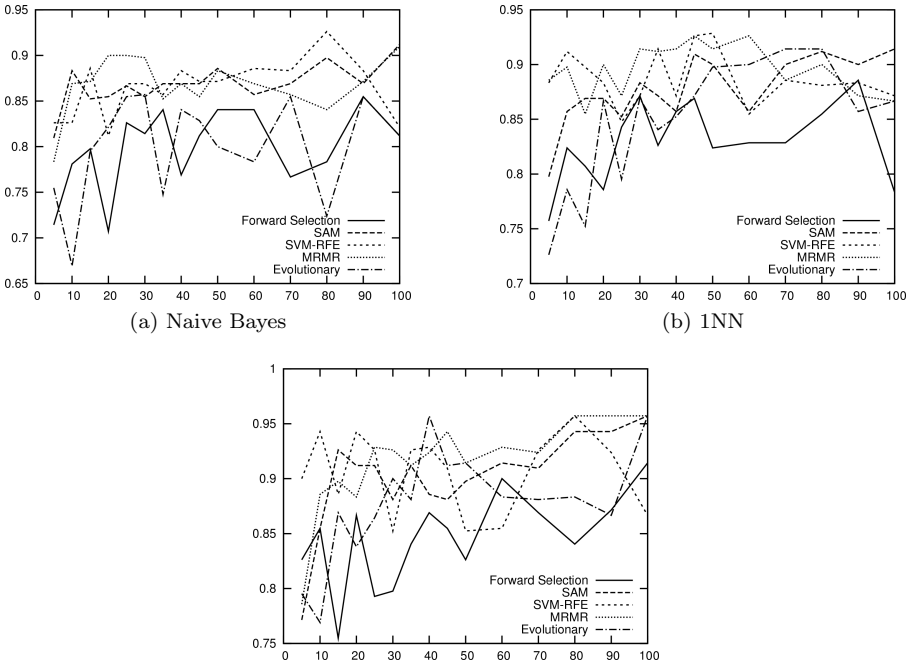



Figure 3: Simple learners combined with a good selection provided better results than the more sophisticated Random Forest on the colon dataset ($n = 62, p = 2000$). x-axis: number of selected feature, y-axis: accuracy

To exemplarily point out the usefulness of filter- and embedded approaches we also compared a selection by SAM-statistics, SVM-RFE and MRMR to two wrapper approaches. First, a forward selection wrapper utilizes a ten-fold cross-validation with a Naive Bayes learner to evaluate feature sets. We used the *RapidMiner*-operator **Forward Selection**. Second, we used and evolutionary wrapper selection with the same cross-validation scheme inside - operator **Optimize Selection (Evolutionary)**. These experiments were conducted on a microRNA-expression dataset with 67 examples and 302 features. It can be seen from Fig. 4 that the wrapper approaches were outperformed by our operators in terms of classification accuracy most of the time. Furthermore, our presented operators needed much less runtime than the wrapper approaches, cf. Fig. 5.

The stabilizing effect of applying ensemble methods to features selection can be seen in Fig. 6. We measured the stability of resulting feature sets for MRMR and a ten-fold ensemble of MRMR. That dataset used is the *colon* dataset [1] with $p = 2000$ features and $n = 62$ examples.




(a) Naive Bayes

(b) 1NN

(c) SVM

Figure 4: Accuracy ($y$-axis) of the different feature selection methods evaluated with different learners dependent on the number of selected features ($x$-axis).

# 8 Conclusion

We presented an extension to *RapidMiner* which delivers implementations of algorithms well suited for very high-dimensional data. The extension contains[2] operators with new feature selection methods, meta-schemes for enhancing existing algorithms and the *Least Angle Regression* algorithm which delivers sparse models. The operators for feature selection and sparse models are useful when practitioners need small and interpretable models. The algorithms in our extension are faster than traditional wrapper approaches. Besides the speedup the classification performance was also enhanced. And we increased the stability of feature selection methods by applying ensemble methods. All these achievements are reasonable for overcoming the curse of dimensionality.

[2]This paper regards version 1.0.6 of the *Feature Selection Extension*.

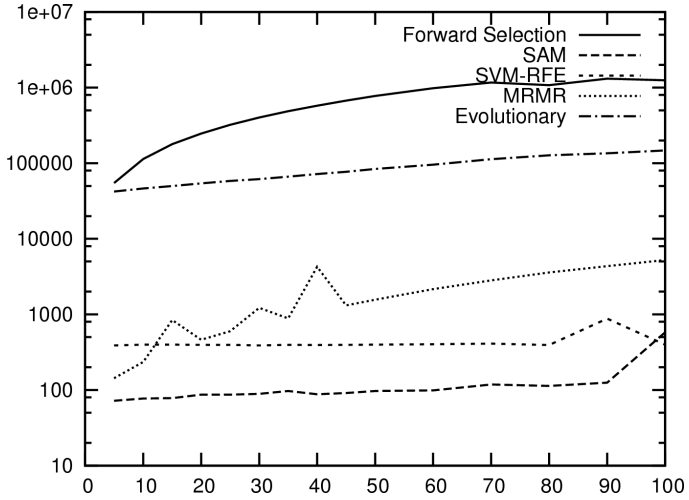



Figure 5: Runtime ($y$-axis, log-scale) of the different selection methods dependent on the number of selected features ($x$-axis).

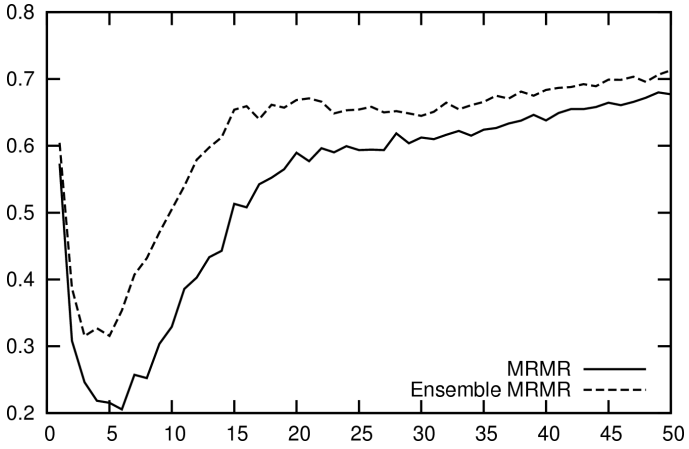



Figure 6: Stability measured by Kuncheva's index ($y$-axis) of the MRMR and an emsembled version of MRMR dependent on the number of selected features ($x$-axis).

## Acknowledgements

Part of the work on this paper has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project C1. Visit the website `http://sfb876.tu-dortmund.de` for more details.

## References

[1] U. Alon, N. Barkai, D. A. Notterman, K. Gishdagger, S. Ybarradagger, D. Mackdagger, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States of America*, 96(12):6745–6750, June 1999.

[2] R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, NJ, 1961.

[3] C. H. Q. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. In *CSB*, pages 523–529. IEEE Computer Society, 2003.

[4] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407, 2004.

[5] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR Special Issue on Variable and Feature Selection*, 3:1157 – 1182, 2003.

[6] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[7] M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *ICML*, pages 359–366, 2000.

[8] A. Kalousis, J. Prados, and M. Hilario. Stability of feature selection algorithms: a study on high-dimensional spaces. *Knowledge and Information Systems*, 12:95–116, 2007. 10.1007/s10115-006-0040-8.

[9] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, 1997.

[10] L. I. Kuncheva. A stability index for feature selection. In V. Devedzic, editor, *IASTED International Conference on Artificial Intelligence*

*and Applications, part of the 25th Multi-Conference on Applied Informatics, Innsbruck, Austria, February 12-14, 2007*, pages 421–427. IASTED/ACTA Press, 2007.

[11] N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4), 2010.

[12] I. Mierswa and M. Wurst. Information preserving multi-objective feature selection for unsupervised learning. In M. K. et al., editor, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2006)*, pages 1545–1552, New York, USA, 2006. ACM Press. accepted for publication.

[13] Y. Saeys, T. Abeel, and Y. V. de Peer. Robust feature selection using ensemble feature selection techniques. In W. Daelemans, B. Goethals, and K. Morik, editors, *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II*, volume 5212 of *Lecture Notes in Computer Science*, pages 313–325. Springer, 2008.

[14] Y. Saeys, I. n. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, August 2007.

[15] B. Schowe and K. Morik. Fast-ensembles of minimum redundancy feature selection. In M. R. Oleg Okun and G. Valentini, editors, *Supervised and Unsupervised Ensemble Methods and their Applications - SUEMA 2010, ECML/PKDD 2010 Workshop*, pages 11–22, 2010.

[16] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58(1):267–288, 1996.

[17] R. Tibshirani, T. Hastie, Narasimhan, and G. Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 99:6567–6572, May 2002.

[18] V. G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 98(9):5116–5121, April 2001.

[19] V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.