# Monte Carlo feature selection for supervised classification: A Statistical supplement

Michał Dramiński[1], Alvaro Rada-Iglesias[2], Stefan Enroth[3], Claes Wadelius[2], Jacek Koronacki[1] ⋆, Jan Komorowski[4,5] ⋆,⋆⋆

[1] Institute of Computer Science, Polish Acad. Sci, Ordona 21, PL-01-237 Warsaw, Poland
[2] Department of Genetics and Pathology, Rudbeck Laboratory, Uppsala University
[3] The Linnaeus Centre for Bioinformatics, Uppsala University and The Swedish University for Agricultural Sciences, Box 758, SE-751-24 Uppsala, Sweden
[4] The Linnaeus Centre for Bioinformatics, Uppsala University,SE-751 24 Uppsala, Sweden
[5] Interdisciplinary Centre for Mathematical and Computer Modelling, Warsaw University, Poland

**Abstract.** In this supplement statistical correctness of the procedure described in our paper *Monte Carlo feature selection for supervised classification* is assessed for two examples studied in that paper.

## 1 Statistical correctness of the MCFS procedure

Let us perform the validation steps of the procedure, following their brief description given in the paper referred to in the Abstract. We shall confine ourselves to describing the first of them for the leukemia data only (for a similar analysis regarding the imputed lymphoma data, based on $\mathrm{cRI}_{gk}$, see [Dramiński *et al.*, 2004]). The first validation step consists of two experiments.

In the first experiment, 15,000 subsets of genes, each with 2,000 randomly selected genes, were drawn from the set of all 7,129 genes. For each such subset of features we thus have 38 samples from two classes, AML and AL. Each such set of 38 samples was randomly split 30 times into a training set of 24 and a test set of 14 samples. Finally, 450,000 trees were constructed on all the 450,000 training sets obtained, and their weighted accuracies were stored. The choice of the number of features for each sample was rather arbitrary, as the only requirement was to include sufficient randomness into the whole experiment.

The second experiment was started with 50 random permutations of the class labels of the samples. Then, for each permutation, the experiment was run similarly to the former one. The only difference was that only 300 subsets

⋆ these authors contributed equally
⋆⋆ to whom correspondence should be addressed

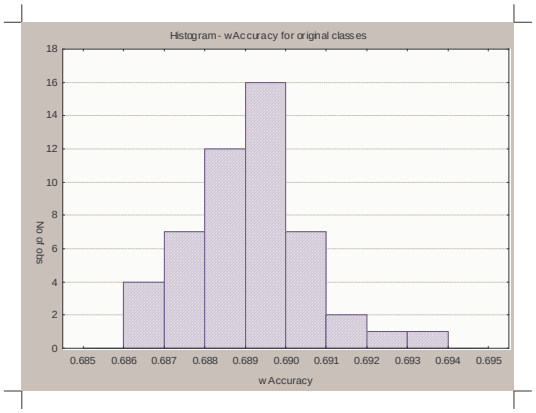


**Fig. 1.** *wAcc* for true classes: leukemia data

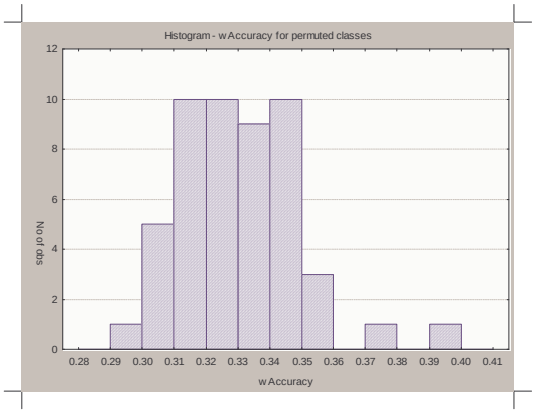


**Fig. 2.** *wAcc* for permuted classes: leukemia data

of 2,000 genes were randomly selected (that number of subsets multiplied by the number of permutations results in the same overall number, i.e. 15,000 of sets of samples, each with 2,000 features.).Hence, 450,000 trees were constructed and their *wAcc*'s stored.

Results of both experiments are summarized by histograms in Fig's. 1, 2, 3. It follows from them that the leukemia data can be considered informative.

The second validation step begins with splitting the set of all $d$ genes, ranked according to $\mathrm{RI}_{g_k}$ in the main step of the procedure, into two subsets: that of the $2b$ relatively most important genes and that of the remaining $d - 2b$ genes. We set $b = 100$ for the leukemia data and $b = 45$ for the imputed lymphoma data. The second validation step consists again of two experiments.

In the first experiment, $b$ genes are randomly drawn 3,000 times from the set of the $2b$ relatively most important ones. Thus, for each such set of the $b$

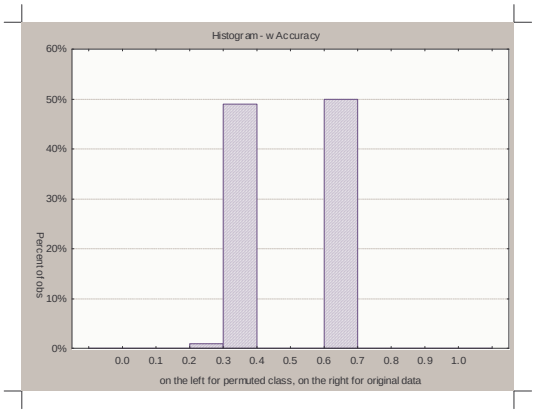


**Fig. 3.** *wAcc* for true and permuted classes: leukemia data

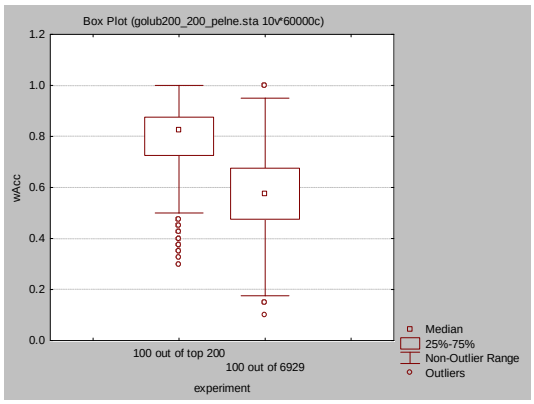


**Fig. 4.** *wAcc* for 100-feature samples: leukemia data

genes, one obtains a set of $n$ samples, each sample with just $b$ features. Now, each such set of samples is split 10 times at random into a training and a test set (as usual, each time 66% of samples are drawn at random for training in such a way as to preserve proportions of the classes from the complete set of training data). The 30,000 weighted accuracies thus obtained are stored.

The second experiment is performed in the same way, except that each of the 3,000 sets of $b$ genes is drawn from the set of the remaining $d - 2b$ genes.

The boxplots for *wAcc*, which resulted from our experiments, are given in Fig. 4 and Fig. 5. In this way, and for the leukemia and imputed lymphoma data, the claim that most important genes have been found by the main step of the procedure has been confirmed.

For reasons explained at the end of Section 2.2 of the paper referred to in the Abstract, we suggest performing an additional confirmatory step, also outlined there (see the diagram in Fig. 6). For the leukemia data, 20 ALL

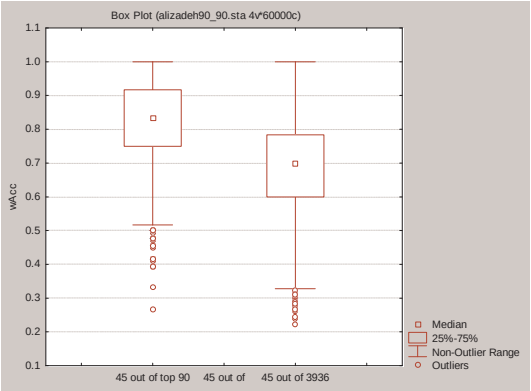


**Fig. 5.** *wAcc* for 45-feature samples: lymphoma data

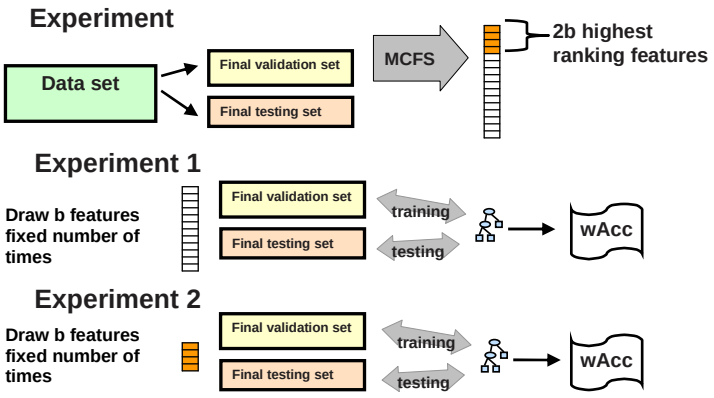


**Fig. 6.** Block diagram for the confirmatory step; MCFS (Monte Carlo Feature Selection) is an acronym for the main step of the procedure

samples and 8 AML samples were drawn at random to the final validation set. For the lymphoma data, the final validation set comprised 26 DLCL samples, 7 FL samples and 9 CLL samples. In the former case, for the main step of the procedure we set $m = 200$, $s = 10,000$ and $t = 10$. In the latter, we chose $m = 100$,$s = 70,000$ and $t = 20$. In the second validation step, we used, respectively, $b = 100$ and $b = 45$ (as already earlier), and, respectively again, 10,000 and 100,000 sets of $b$ genes were drawn in both experiments. The results obtained for the final test sets, not used in the main step in any way, are summarized by means of boxplots in Fig.'s 7 and 8. Thus, the earlier obtained rankings that were based on all samples may safely be claimed significant.

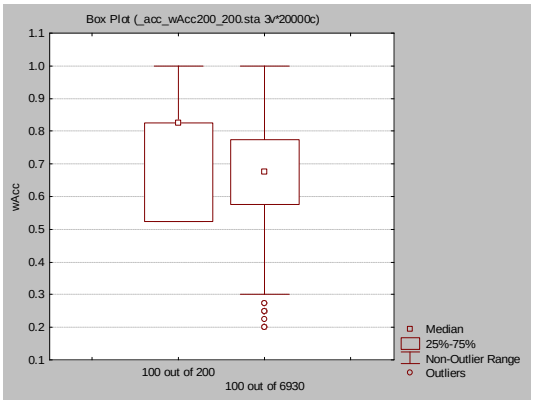


**Fig. 7.** Confirmatory step: leukemia data (in the left boxplot, upper inner hinge coincides with the median)

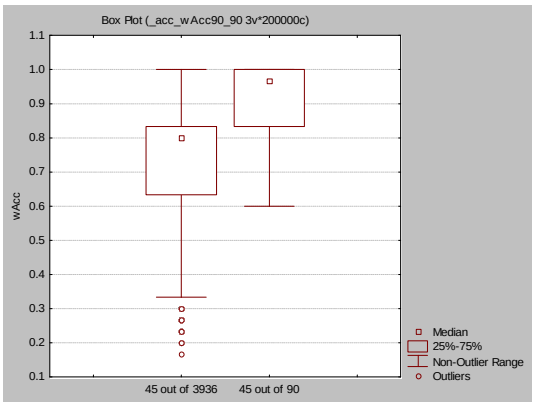


**Fig. 8.** Confirmatory step: lymphoma data

## 2 Monte Carlo Feature Selection (MCFS) algorithm with a rule based classifier

The original MCFS algorithm relies on using a decision tree classifier. In this section, we study the MCFS algorithm with rule-based, the so-called ADX, classifier in lien of the decision tree. ADX classifier has been proposed and implemented by Michał Dramiński [Dramiński, 2004].

### 2.1 Definitions

Let $D$ denote a database. Each row in a database is an event $e$ (sometimes called an object), what means that $D$ is a set of events/objects. Let $|D|$ denote the number of events in $D$. Each column in database $D$ corresponds to

one attribute of events. Attribute can be nominal such as color (e.g., possible values of color are: blue, red, green, etc.) or ordered such as height represented by an ordered set of values/levels (e.g., small, medium and high), or numerical such as height or weight measured, respectively, in inches and pounds. Attributes describe events in $D$. All possible combinations of values of attributes describing an event form a domain. We assume that $D$ includes a special attribute called decision attribute d which determines class of each event. Thus, class is a value of $d$ and each event in $D$ has a defined class. We assume that the decision attribute has nominal values.
Selector $s$ is an expression that describes set of events using definition of one attribute. For example: color=blue; or weight>80. Each simple selector consists of a name of attribute, its value and an operator relating the two. More advanced/complex selector can include a list of nominal or ordered values (e.g. color=[blue, red]) or a range of numeric values (e.g. weight=(70;80] ). Both mentioned cases can be written as a set of simple selectors suitably combined into one advanced selector: "color=[blue, red]" is the same as "color=blue or color=red", "weight=(70;80]" is equivalent to "weight>70 and weight≤80". Selector $s$ without any condition imposed on the attribute is called general and this kind of s will be presented as "attribute=*" (e.g. color=* means any color).
Let complex $c$ denote a set of selectors and let length $n$ of the complex denote the number of simple or advanced selectors contained in the complex. For example complex $c = \langle s_1, s_2, s_3 \rangle$ has length 3. Complex is understood as a conjunction of selectors. Coverage *cov* of selector $s$ is the number of events which satisfy condition in $s$ (and are called covered events) divided by $|D|$. Coverage of any general selector is equal to 1.0. Coverage of $c$ is the number of covered events (events which satisfy complex $c$, i.e., condition in $s_1$ and in $s_2$ and in $s_3$ and so on) divided by $|D|$. Positive set $D^p$ for a given class is the set of events whose decision attribute has the value corresponding to this class. During creation of rules for given value $v$ of decision attribute $d$, all events that have $v$ for $d$ are called positive. All other events are called negative $D^n$. For a given class, positive coverage *pcov* of a given complex is the coverage on the subset of events which have the considered class. Negative coverage *ncov* of the complex is the coverage on the subset of events whose class is different from the considered class. By definition, strong rules determine classes, and hence their negative coverage is equal to 0. A complex which describes just one class can be called a strong rule (for that class). It means that if such a complex has *ncov* equal to 0 it implies only one class of events. We can say that probability of this class for such a rule is 1. If complex has $ncov > 0$ and $pcov > ncov$ for a given class (value $v$ of decision attribute $d$) it still describes this class but with some probability lesser then 1. This probability is usually called confidence or consistency of a rule. For strong rule confidence always equals 1. Set of complexes (disjunction of them) that describe one class we

call a ruleset. Set of rulesets that completely describe all classes (all values of decision attribute) we call a ruleset family.

### 2.2 The ADX algorithm — creation of a ruleset for one class

The general idea of ADX algorithm is based on the assumption that conjunction of selectors cannot have a bigger *cov* than minimal *cov* for each of these selectors. The main goal of the algorithm is to find the best ruleset (set of complexes such that each of them has possibly large *pcov* and possibly small *ncov* for each of the classes). If such rulesets exist, classification can be realized by measuring similarity of new unlabeled event to each ruleset (set of complexes for each class). Class which is most similar to the considered event will be chosen as a classifier decision. We assume that world is not perfect and probably input data either. It means that we can have in the data: ambiguities, noise, and unknown values.

**Finding selectors base** For a given class, for each value of each attribute in $D$, the algorithm finds *pcov* and *ncov*. It means that in the first step, the algorithm builds a set of contingency tables from $a_1$ to $a_m$ (where $m$ denotes the number of all attributes with decision attribute $d$ excluded).

At this moment a new parameter can be defined. With this parameter, to be called $minSelectorPosCoverage$ minimal acceptable positive coverage for a selector is determined. All selectors with smaller positive coverage than this parameter are deleted. All selectors that satisfy the condition determined by $minSelectorPosCoverage$ are the basis for the next steps. Notice that in this step only simple selectors are selected. Size of selector base (number of selectors that satisfy $minSelectorPosCoverage$) affects complexity of later calculations.

**Creation of candidates** Complexes whose quality has not been evaluated yet are called candidates. Creation of such complexes with length 2 is very simple. These are all possible selectors' pairs excluding pairs where both selectors are based on the same attribute. Notice that complex (of length 2) $c_1 = \langle s_1, s_2 \rangle$ equals $c_2 = \langle s_2, s_1 \rangle$, because each complex consists of conjunction of selectors. There is an important issue about creation of complexes longer than 2. To create a new complex candidate, the algorithm uses two complexes with length smaller by 1, which can be called parents. Parent complexes need to have common part of length $n-1$ (where $n$ denotes the length of parents complexes). For example complex $c_1 = \langle s_1, \boldsymbol{s_2}, \boldsymbol{s_3} \rangle$ and $c_2 = \langle \boldsymbol{s_2}, \boldsymbol{s_3}, s_5 \rangle$ can be used to create $c_3 = \langle s_1, \boldsymbol{s_2}, \boldsymbol{s_3}, s_5 \rangle$. Creation of complexes with length 2 is based on simple selectors. Complexes with length 3 are based on complexes with length 2 and so on.

**Estimation of candidates' quality** After creation of each candidates set there is a need to separately estimate quality of each newly created complex. During the estimation process, for each complex candidate there is calculated positive and negative coverage (*pcov* and *ncov*) on a training set. This can be done by calculation of how many positive and negative events are covered by the complex considered. The best complexes become parents and are used to create in the next iteration complex candidates 1 selector longer. Some quality measure $Q$ is used to compare complexes and it is defined as follows $Q = (pcov - ncov)(1 - ncov)^u$, where $u$ equals to 1 for estimation of candidates' process and $u = 1/2$ for later final selection. Complexes that do not cover any positive event ($pcov = 0$) are useless and are deleted.

**Selection of parents** Selection is a process during which ADX selects complexes (using estimation measures *pcov* and *ncov*) as parents from which next candidates are created. The set of selected complexes will be used in the next iteration as a parent set. However, parameter $searchBeam$ is used to limit the number of complexes to be a parent in the next step. This parameter uses measures $Q$ to select candidates for the next step. After the evaluation process, selection of a set of best complexes is performed that can be used to create next complexes of length greater by 1. Complexes which were not selected to be parents are stored for final selection. Parameter $searchBeam$ controls the scope of exploration but also affects learning time. With increasing $searchBeam$ exploration grows also but, unfortunately, learning time grows too. Once the set of parents is selected new candidates can be created (complexes that are 1 selector longer). The algorithm can be stopped when the set of new candidates is empty (if there are no parents that have common part of $n - 1$ length).

**Merging of complexes and final selection of ruleset** After creation of all possible complexes, the algorithm decreases the number of them and improves their quality. If complexes are based on the same attribute set and only one selector has different value then it is possible to merge such complexes. Instead of old complexes the new one is added. For example: $\langle A = 1, B = 3, C = 4\rangle \oplus \langle A = 1, B = 3, C = 7\rangle \Longrightarrow \langle A = 1, B = 3, C = [4, 7]\rangle$, where new selector is $C = [4, 7]$. For the resulting complex *pcov* and *ncov* are the sums of corresponding coverages of deleted complexes. The main criterion that allows for merging of two complexes is the increase of $Q$ for the outcome complex compared to any component complexes.

In final selection step - from the whole set of stored rules we have to select the most appropriate set that can be used for later prediction. Selection of final rules is based on measure $Q$ and selecting fixed number of rules $finalBeam$.

First of all, complexes are sorted by $Q$ and starting from the highest $Q$, they are added one by one to the final set. After adding a complex it is

retained in the set, if its inclusion does not decrease the value of $Q_r$. Measure $Q_r = (S^p - S^n)\sqrt{1 - S^n}$ is very similar to $Q_{u=1/2}$ where $S^p$ and $S^n$ are sums of suitable factors over all rules in the final ruleset. The factors $S^p$ and $S^n$ are sums of scores (to be described in the next section) obtained on random subset of training events – $S^p$ is a sum for positive events $e$ and $S^n$ for negative ones.

$$S^p = \sum_{e \in D^p} S(e) \tag{1}$$

$$S^n = \sum_{e \in D^n} S(e) \tag{2}$$

The idea is to select complexes that maximize score for positive events and minimize it for the negative set – both for randomly selected subset of training events.

### 2.3 Classification in ADX

For prediction, the ADX algorithm uses the set of final rules, obtained from final selection. For each of the rulesets, subset of rules that cover an event are selected and the score measure is calculated for each such subset. The highest score determines the class label. The score measure combines *pcov*, *ncov* and *prob* of rules that cover the event, where *prob* denotes probability of positive class occurrence for a given rule. The score measure can be defined in any of the following ways:

1. $$S_0 = \frac{\frac{p-n}{|r|}}{\frac{P-N}{|R|}} \tag{3}$$

2. $$S_1 = \frac{p-n}{P-N} \tag{4}$$

3. $$S_2 = \frac{p}{P} * \frac{N}{n} \tag{5}$$

4. $$S_3 = \frac{p}{P} * [1 - \frac{n}{N}] \tag{6}$$

5. $$S_4 = 1 - \prod_r (1 - prob(r)) \tag{7}$$

6. $$S_5 = \frac{\sum_r prob(r)}{r} \tag{8}$$

7.

$$S_6 = \begin{cases} S_5 \\ \frac{P}{p} \text{ if } \bigwedge_c S_5^c = 1 \end{cases}$$

8.

$$S_7 = \begin{cases} S_4 \\ \frac{P}{p} \text{ if } \bigwedge_c S_4^c = 1 \end{cases}$$

where:

- $p$ – denotes sum *pcov* of rules that cover tested event
- $n$ – denotes sum *ncov* of rules that cover tested event
- $P$ – denotes sum *pcov* of rules for a given class
- $N$ – denotes sum *ncov* of rules for a given class
- $r$ – denotes subset of rules that cover an event
- *prob* – probability of occurrence of positive class if the rule covers the event
- $c$ – is a class label

The experiments have proven that the most universal measure is $S_6$ and gives very good and stable classification results.

### 2.4 Integration of ADX into MCFS

The original version of MCFS is based on decision tree classifier but if any other classifier meets two simple criteria there is no problem to integrate it:

- It works at least as fast as a decision tree - because we would like to train/test thousands of classifiers in reasonable time.
- It is possible to propose $RI_{gk}$ measure for such classifier.

Experiment show that the first criterion is definitely fulfilled by ADX classifier. To fulfill the second criterion the following measure $RI_{gk}$ has to be introduced:

$$\mathrm{RI}_{g_k} = \sum_{\sigma=1}^{st} (wAcc)^u \sum_{r_{g_k}(\sigma)} \mathrm{Q}(r_{g_k}(\sigma)) \left(cov(r_{g_k}(\sigma))\right)^v , \tag{9}$$

The proposed measure is similar to $RI$ for the decision tree classifier. In the above formula $r_{g_k}(\sigma)$ denotes the rule that contains selector based on $g_k$ (rule plays the same role as nodes in $RI$). Instead of Information Gain for a tree node we can now use $Q$ of the rule. Coverage of the rule plays the role of the fraction of events tested in a particular node of the decision tree. Note that from ADX we can obtain separate sets of rules, each of them pertaining to a different class. Therefore it is possible to construct separate rankings of genes for different classes.

### 2.5 Top ranking genes by two implementations of the MCFS algorithm

It is interesting to see if the rankings of genes provided by two different implementations of the MCFS algorithm, one with decision trees and another with ADX rule based classifiers, can be considered overlapping to a sufficiently large extent. For our two example data sets (Alizadeh and Golub), we have found that the groups (of whatever but equal size, from tens to hundreds) of top ranking genes obtained by the two procedures, overlap in about 50% for the Alizadeh et al. and about 75% for the Golub et al. data. This result is illustrated in tables 1 and 2 for the sets of 45 and 90 top ranking features. Given that the ranking is made between thousands of genes, the overlap can be considered reasonably high.

| algorithm | J48 | ADX |
|---|---|---|
| J48 | x | 21 |
| ADX | 41 | x |

**Table 1.** Overlap of top rankings obtained from c4.5 and ADX (top right 45 top features, bottom left 90 top features) – Alizadeh et al. data

| algorithm | J48 | ADX |
|---|---|---|
| J48 | x | 33 |
| ADX | 65 | x |

**Table 2.** Overlap of top rankings obtained from c4.5 and ADX (top right 45 top features, bottom left 90 top features) – Golub et al. data

## References

[Dramiński *et al.*, 2004] Dramiński, M., Koronacki, J., Ćwik, J. and Komorowski, J. (2004) Monte Carlo Gene Screening for Supervised Classification. In: *Current Issues in Data and Knowledge Engineering*, B. De Baets, R. de Caluwe, G. de Tr, J. Fodor, J. Kacprzyk, S. Zadrozny (eds.), Exit, Warsaw.

[Dramiński, 2004] Dramiński M. (2004) ADX Algorithm: A brief Description of a Rule Based Classifier. Proceedings of the New Trends in Intelligent Information Processing and WebMining IIS'2004 Symposium, Zakopane, Poland, Springer-Verlag.

## A Top ranging genes for leukemia data MCFS+J48

Ranging of the top 200 highly estimated genes obtained of MCFS using decision tree (leukemia data):

1. X95735_at
2. M31166_at
3. M27891_at
4. M55150_at
5. D88422_at
6. M23197_at
7. M98399_s_at
8. U50136_rna1_at
9. M21551_rna1_at
10. M27783_s_at
11. M54995_at
12. U02020_at
13. M77142_at
14. M81933_at
15. X70297_at
16. Y12670_at
17. U46499_at
18. M83652_s_at
19. U46751_at
20. L09209_s_at
21. M16038_at
22. D14874_at
23. M84526_at
24. M92287_at
25. M31523_at
26. X62654_rna1_at
27. M31303_rna1_at
28. D49950_at
29. U22376_cds2_s_at
30. J05243_at
31. D26308_at
32. X90858_at
33. J04615_at
34. X74262_at
35. L08177_at
36. U37055_rna1_s_at
37. X87613_at
38. M80254_at
39. X62320_at
40. J04990_at

41. L47738_at
42. M29540_at
43. M91432_at
44. U85767_at
45. M24400_at
46. M96326_rna1_at
47. L05148_at
48. M11722_at
49. U12471_cds1_at
50. U82759_at
51. X59417_at
52. X04085_rna1_at
53. HG4321-HT4591_at
54. U16954_at
55. M22324_at
56. U41813_at
57. HG2981-HT3127_s_at
58. X14008_rna1_f_at
59. M12959_s_at
60. M62762_at
61. X06182_s_at
62. M81695_s_at
63. M63138_at
64. D87076_at
65. M28130_rna1_s_at
66. D14664_at
67. X17042_at
68. D38073_at
69. M57731_s_at
70. AFFX-HUMTFRR/M11507_3_at
71. U09087_s_at
72. HG627-HT5097_s_at
73. D87742_at
74. X85116_rna1_s_at
75. Y00787_s_at
76. M69043_at
77. L08246_at
78. M58297_at
79. D26156_s_at
80. HG1612-HT1612_at
81. L09717_at
82. M22960_at
83. M13452_s_at
84. X07743_at
85. U62136_at

86. X15949_at
87. AF009426_at
88. D10495_at
89. M25897_at
90. Z48501_s_at
91. M31158_at
92. J03589_at
93. X51521_at
94. U38846_at
95. X16546_at
96. M31211_s_at
97. U73737_at
98. M86406_at
99. X58431_rna2_s_at
100. X62535_at
101. HG2855-HT2995_at
102. M83667_rna1_s_at
103. M63379_at
104. U97105_at
105. X74801_at
106. L34600_at
107. HG2379-HT3996_s_at
108. J04621_at
109. AF012024_s_at
110. D38128_at
111. U67963_at
112. U79734_at
113. M95678_at
114. Z69881_at
115. U41767_s_at
116. M29194_at
117. Z49194_at
118. L42572_at
119. L38608_at
120. M20203_s_at
121. J03930_at
122. U72621_at
123. X52142_at
124. U32944_at
125. J03801_f_at
126. M28209_at
127. U20998_at
128. M19507_at
129. U90902_at
130. D38522_at

131. L20941_at
132. L19779_at
133. M95178_at
134. X52056_at
135. M83221_at
136. M13792_at
137. X77533_at
138. HG4582-HT4987_at
139. M26708_s_at
140. M28170_at
141. L25931_s_at
142. HG2788-HT2896_at
143. L41870_at
144. U43292_at
145. X61587_at
146. U00802_s_at
147. U88629_at
148. S82470_at
149. L13278_at
150. S82185_at
151. U53225_at
152. M80899_at
153. U19878_at
154. X66533_at
155. HG4755-HT5203_s_at
156. M25809_at
157. L11669_at
158. U40369_rna1_at
159. HG3494-HT3688_at
160. M68891_at
161. X64364_at
162. U72936_s_at
163. M29696_at
164. M75715_s_at
165. L28821_at
166. AFFX-HUMTFRR/M11507_M_at
167. M93056_at
168. U29175_at
169. X16901_at
170. X80907_at
171. X17094_at
172. M61853_at
173. HG3454-HT3647_at
174. X66401_cds1_at
175. U61836_at

176. M32304_s_at
177. X63753_at
178. X83490_s_at
179. Z15115_at
180. U90552_s_at
181. X57579_s_at
182. Z32765_at
183. M86873_s_at
184. M29971_at
185. L13329_at
186. D31887_at
187. U58034_at
188. D88378_at
189. U31342_at
190. L20321_at
191. Z18948_at
192. M20642_s_at
193. L49219_f_at
194. U77396_at
195. M19045_f_at
196. D86967_at
197. AF005043_at
198. D82346_at
199. Y00339_s_at
200. J04027_at

## B Top ranging genes for lymphoma data MCFS+J48

Ranging of the top 200 highly estimated genes obtained of MCFS using decision tree (lymphoma data - not imputed data):

1. GENE1622X
2. GENE1602X
3. GENE1613X
4. GENE1553X
5. GENE530X
6. GENE1610X
7. GENE1647X
8. GENE653X
9. GENE1606X
10. GENE2426X
11. GENE622X
12. GENE2402X
13. GENE1661X
14. GENE598X

15. GENE2668X
16. GENE669X
17. GENE588X
18. GENE537X
19. GENE639X
20. GENE2553X
21. GENE542X
22. GENE1673X
23. GENE685X
24. GENE454X
25. GENE844X
26. GENE640X
27. GENE2368X
28. GENE1607X
29. GENE1635X
30. GENE1605X
31. GENE2404X
32. GENE1632X
33. GENE584X
34. GENE834X
35. GENE1603X
36. GENE1648X
37. GENE849X
38. GENE642X
39. GENE1662X
40. GENE524X
41. GENE694X
42. GENE617X
43. GENE771X
44. GENE1672X
45. GENE1611X
46. GENE651X
47. GENE620X
48. GENE464X
49. GENE2589X
50. GENE626X
51. GENE689X
52. GENE1637X
53. GENE646X
54. GENE563X
55. GENE2356X
56. GENE586X
57. GENE2373X
58. GENE2097X
59. GENE2391X

60. GENE760X
61. GENE1644X
62. GENE1599X
63. GENE2360X
64. GENE3497X
65. GENE2340X
66. GENE2547X
67. GENE236X
68. GENE447X
69. GENE712X
70. GENE650X
71. GENE717X
72. GENE1625X
73. GENE616X
74. GENE2240X
75. GENE2554X
76. GENE2190X
77. GENE2345X
78. GENE1617X
79. GENE631X
80. GENE816X
81. GENE636X
82. GENE2324X
83. GENE625X
84. GENE802X
85. GENE2244X
86. GENE1537X
87. GENE459X
88. GENE2270X
89. GENE655X
90. GENE977X
91. GENE1192X
92. GENE641X
93. GENE3621X
94. GENE1619X
95. GENE728X
96. GENE1676X
97. GENE2357X
98. GENE2346X
99. GENE632X
100. GENE729X
101. GENE1623X
102. GENE2374X
103. GENE649X
104. GENE538X

105. GENE2364X
106. GENE675X
107. GENE659X
108. GENE647X
109. GENE528X
110. GENE1731X
111. GENE611X
112. GENE3792X
113. GENE812X
114. GENE1612X
115. GENE1975X
116. GENE638X
117. GENE2109X
118. GENE508X
119. GENE2271X
120. GENE765X
121. GENE770X
122. GENE1507X
123. GENE768X
124. GENE1747X
125. GENE788X
126. GENE734X
127. GENE292X
128. GENE2096X
129. GENE738X
130. GENE1615X
131. GENE2253X
132. GENE1583X
133. GENE633X
134. GENE735X
135. GENE663X
136. GENE2110X
137. GENE531X
138. GENE455X
139. GENE3704X
140. GENE2370X
141. GENE2310X
142. GENE1295X
143. GENE546X
144. GENE2378X
145. GENE2403X
146. GENE709X
147. GENE1656X
148. GENE838X
149. GENE733X

150. GENE2113X
151. GENE681X
152. GENE473X
153. GENE1204X
154. GENE2166X
155. GENE786X
156. GENE1220X
157. GENE457X
158. GENE1616X
159. GENE539X
160. GENE2221X
161. GENE3770X
162. GENE2293X
163. GENE710X
164. GENE541X
165. GENE2076X
166. GENE742X
167. GENE2108X
168. GENE682X
169. GENE740X
170. GENE2321X
171. GENE2251X
172. GENE1631X
173. GENE3880X
174. GENE543X
175. GENE303X
176. GENE593X
177. GENE691X
178. GENE714X
179. GENE1600X
180. GENE2214X
181. GENE2328X
182. GENE1674X
183. GENE648X
184. GENE1732X
185. GENE3635X
186. GENE307X
187. GENE3384X
188. GENE713X
189. GENE1636X
190. GENE2339X
191. GENE3787X
192. GENE2429X
193. GENE703X
194. GENE884X

195. GENE3786X
196. GENE1748X
197. GENE529X
198. GENE676X
199. GENE2548X
200. GENE2424X

## C Top ranging genes for leukemia data MCFS+ADX

Ranging of the top 200 highly estimated genes obtained of MCFS using rule based classifier (leukemia data):

1. X95735_at
2. M55150_at
3. M31166_at
4. M27891_at
5. M77142_at
6. D88422_at
7. X70297_at
8. U46499_at
9. U50136_rna1_at
10. M27783_s_at
11. M91432_at
12. L09209_s_at
13. M23197_at
14. M16038_at
15. Y12670_at
16. D14874_at
17. U22376_cds2_s_at
18. M92287_at
19. X62654_rna1_at
20. U02020_at
21. X74262_at
22. M81933_at
23. M31523_at
24. U41813_at
25. M12959_s_at
26. M28130_rna1_s_at
27. U85767_at
28. J03930_at
29. M84526_at
30. U62136_at
31. X59417_at
32. X90858_at
33. U12471_cds1_at

34. X04085_rna1_at
35. U46751_at
36. U09087_s_at
37. J04615_at
38. M83652_s_at
39. U73737_at
40. L47738_at
41. X80907_at
42. X74801_at
43. X07743_at
44. M54995_at
45. J05243_at
46. L27584_s_at
47. Y00787_s_at
48. U32944_at
49. M21551_rna1_at
50. M63138_at
51. D38073_at
52. M31303_rna1_at
53. M11722_at
54. AF009426_at
55. D10495_at
56. U38846_at
57. L42572_at
58. AFFX-HUMTFRR/M11507_3_at
59. M31158_at
60. X52142_at
61. Z69881_at
62. X15949_at
63. HG4321-HT4591_at
64. D26156_s_at
65. M28170_at
66. X85116_rna1_s_at
67. L41870_at
68. X06182_s_at
69. D38522_at
70. M31211_s_at
71. X62320_at
72. U29175_at
73. M29540_at
74. Y11710_rna1_at
75. D26308_at
76. M63488_at
77. M96326_rna1_at
78. X58431_rna2_s_at

79. X61587_at
80. U65928_at
81. M80254_at
82. U26266_s_at
83. HG2810-HT2921_at
84. Z49194_at
85. HG1612-HT1612_at
86. Z15115_at
87. M58297_at
88. X57579_s_at
89. D86479_at
90. X66533_at
91. J03801_f_at
92. L28821_at
93. X62535_at
94. M98399_s_at
95. S50223_at
96. S82185_at
97. U41767_s_at
98. D63880_at
99. J03589_at
100. X66401_cds1_at
101. L08246_at
102. X77533_at
103. AF012024_s_at
104. U72936_s_at
105. U31342_at
106. M19045_f_at
107. U02493_at
108. X14008_rna1_f_at
109. M54915_s_at
110. HG2855-HT2995_at
111. U72621_at
112. M94633_at
113. U35451_at
114. L38608_at
115. M55040_at
116. M23178_s_at
117. M29696_at
118. U90546_at
119. U82759_at
120. U73960_at
121. L07648_at
122. M24349_s_at
123. M22960_at

124. M62762_at
125. X59350_at
126. L13278_at
127. D82346_at
128. U20998_at
129. U84487_at
130. U05259_rna1_at
131. D14664_at
132. S79854_at
133. X17042_at
134. M83667_rna1_s_at
135. M24400_at
136. D49950_at
137. U27460_at
138. U16307_at
139. M25897_at
140. X98261_at
141. U59321_at
142. U26173_s_at
143. X76648_at
144. D88270_at
145. U67963_at
146. M81695_s_at
147. U79274_at
148. D38128_at
149. M74088_s_at
150. D43950_at
151. Y13896_at
152. M29194_at
153. U66838_at
154. U28413_at
155. X63469_at
156. U31556_at
157. AC002115_cds4_at
158. M22324_at
159. M95678_at
160. D86983_at
161. X16546_at
162. U47928_at
163. M63379_at
164. S82470_at
165. J04990_at
166. HG2788-HT2896_at
167. D43682_s_at
168. M15059_at

169. X87613_at
170. U90902_at
171. X64072_s_at
172. U90552_at
173. M69043_at
174. X54326_at
175. M86406_at
176. Z19002_at
177. X66899_at
178. Z68747_at
179. X97748_s_at
180. M63438_s_at
181. L25931_s_at
182. U28833_at
183. U94836_at
184. M37435_at
185. U28758_s_at
186. L10386_at
187. D87742_at
188. HG4332-HT4602_at
189. U00802_s_at
190. M19507_at
191. X83490_s_at
192. HG4316-HT4586_at
193. M65214_s_at
194. D80001_at
195. D83785_at
196. X63753_at
197. U49020_cds2_s_at
198. U50733_at
199. M60527_at
200. M13792_at

## D Top ranging genes for lymphoma data MCFS+ADX

Ranging of the top 200 highly estimated genes obtained of MCFS using rule based classifier (lymphoma data):

1. GENE1622X
2. GENE1618X
3. GENE1617X
4. GENE1602X
5. GENE659X
6. GENE1744X
7. GENE1619X

8. GENE1637X
9. GENE1636X
10. GENE1616X
11. GENE1644X
12. GENE1702X
13. GENE1662X
14. GENE1647X
15. GENE1661X
16. GENE653X
17. GENE766X
18. GENE1610X
19. GENE622X
20. GENE530X
21. GENE1553X
22. GENE1632X
23. GENE735X
24. GENE1613X
25. GENE1634X
26. GENE1648X
27. GENE1635X
28. GENE712X
29. GENE1643X
30. GENE834X
31. GENE1645X
32. GENE1607X
33. GENE1641X
34. GENE1625X
35. GENE1649X
36. GENE1640X
37. GENE1627X
38. GENE588X
39. GENE2368X
40. GENE1663X
41. GENE2402X
42. GENE675X
43. GENE598X
44. GENE1603X
45. GENE633X
46. GENE710X
47. GENE669X
48. GENE620X
49. GENE625X
50. GENE1633X
51. GENE1753X
52. GENE1731X

53. GENE1595X
54. GENE1609X
55. GENE689X
56. GENE1608X
57. GENE721X
58. GENE586X
59. GENE641X
60. GENE1646X
61. GENE1650X
62. GENE1660X
63. GENE1639X
64. GENE651X
65. GENE642X
66. GENE1638X
67. GENE616X
68. GENE1606X
69. GENE639X
70. GENE1693X
71. GENE1674X
72. GENE399X
73. GENE631X
74. GENE491X
75. GENE1599X
76. GENE1596X
77. GENE1692X
78. GENE473X
79. GENE655X
80. GENE2110X
81. GENE647X
82. GENE626X
83. GENE1204X
84. GENE1631X
85. GENE648X
86. GENE1651X
87. GENE738X
88. GENE844X
89. GENE1295X
90. GENE531X
91. GENE1652X
92. GENE636X
93. GENE589X
94. GENE1698X
95. GENE711X
96. GENE1730X
97. GENE1549X

98. GENE611X
99. GENE1696X
100. GENE1671X
101. GENE2395X
102. GENE1623X
103. GENE765X
104. GENE1653X
105. GENE698X
106. GENE1514X
107. GENE2403X
108. GENE1629X
109. GENE558X
110. GENE1537X
111. GENE716X
112. GENE1539X
113. GENE555X
114. GENE652X
115. GENE896X
116. GENE2404X
117. GENE537X
118. GENE524X
119. GENE2426X
120. GENE717X
121. GENE650X
122. GENE638X
123. GENE685X
124. GENE2555X
125. GENE1615X
126. GENE2391X
127. GENE977X
128. GENE236X
129. GENE1656X
130. GENE2668X
131. GENE682X
132. GENE542X
133. GENE719X
134. GENE508X
135. GENE709X
136. GENE2244X
137. GENE646X
138. GENE727X
139. GENE1545X
140. GENE1516X
141. GENE1624X
142. GENE527X

143. GENE2339X
144. GENE697X
145. GENE1684X
146. GENE649X
147. GENE660X
148. GENE2554X
149. GENE676X
150. GENE534X
151. GENE1612X
152. GENE595X
153. GENE1614X
154. GENE2400X
155. GENE1223X
156. GENE635X
157. GENE1697X
158. GENE2354X
159. GENE563X
160. GENE1578X
161. GENE446X
162. GENE1600X
163. GENE1655X
164. GENE654X
165. GENE681X
166. GENE587X
167. GENE1523X
168. GENE775X
169. GENE585X
170. GENE2190X
171. GENE1736X
172. GENE1673X
173. GENE725X
174. GENE506X
175. GENE2373X
176. GENE2345X
177. GENE1679X
178. GENE771X
179. GENE2346X
180. GENE657X
181. GENE783X
182. GENE1605X
183. GENE565X
184. GENE734X
185. GENE645X
186. GENE868X
187. GENE1581X

188. GENE694X
189. GENE619X
190. GENE695X
191. GENE568X
192. GENE538X
193. GENE490X
194. GENE2392X
195. GENE464X
196. GENE786X
197. GENE529X
198. GENE1654X
199. GENE520X
200. GENE632X