5. Variable selection

*Variable selection* can be a part of processing algorithm design, especially for decision trees. Nonetheless, here variable selection is employed in order to find most important or useful variables for various data mining tasks such as classification and clustering. The other good reason is to recognize less important or poor variables that are rather unnecessary, irrelevant or even distracting for the afore-mentioned goals. These can be deleted from the data set.

Particularly, removing poor variables is important when we have to reduce the high dimension of a data set such as hundreds, thousands or even more. Such are necessary to reduce to limit running times for building computational models.
5.1 Simple technique

Pearson correlation can be used to calculate dependencies between quantitative variables $X$ and $Y$ for $n$ cases in a data set utilizing the means of $X$ and $Y$.

\[
 r(X, Y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sum_{i=1}^{n} (y_i - \bar{y})^2}}
\]
If a correlation coefficient between two variables are, say, higher than 0.6 or 0.7, one of them could be left out the data set and, thus, to reduce its dimension and to reduce redundancy.

However, this Pearson correlation coefficient is only able to compute linear correlations.

Nonetheless, it is not perhaps always wise to delete a variable in this way, since we do not know, whether such a variable is important or less important for a data mining task, e.g. classification.
An “indefinite thought”: If two variables with high correlation occurred to be important, might it even impair classification results if one of them were deleted? Some machine learning method might be sensitive to such change in the data. Then removing a highly correlating variable might be even questionable. In any case, it is a good principle to remove such variables that we know beforehand or can show to be poor or less important for a data mining task.
5.2 Principles of variable selection

We may divide variable selection techniques into two general types. One group includes *open loop* or *filter techniques*. The other includes *closed loop* or *wrapper techniques*.

The filter techniques are based mostly on selecting variables through the use of between-class separability criteria. These techniques do not consider the effect of selected variables on the performance of an entire processing algorithm, for example, a classifier, because the variable selection criterion does not involve predictive evaluation for reduced data sets with selected variable subsets only.
Instead, open loop or filter techniques select, for example, such variables for which the resulting reduced data set has maximal between-class separability, usually defined based on between-class covariances. The ignoring of the effect of a selected variable subset on the performance of the predictor like a classifier (lack of feedback from the predictor) is a weak side of filter techniques. On the other hand, these are computationally less expensive. Fig. 5.1 depicts the principle of filter techniques.
Fig. 5.1 The principle of open loop or filter variable selection techniques.
Closed loop or wrapper techniques are based on variable selection using predictor performance – thus providing processing feedback – as a criterion of variable subset selection.

Wrapper techniques generally provide better selection of a variable subset, since they fulfill the ultimate goal and criterion of optimal variables selection by providing the best prediction. Fig. 5.2 depicts the principle.
Fig. 5.2 The principle of closed loop or wrapper variable selection techniques.
A procedure for optimal variable selection contains:

- **Variable selection criterion** \( J \) that allows us to judge whether one subset of variables is better than another (evaluation method)

- Systematic **search procedure** that allows us to search through candidate subsets of variables and includes the initial state of the search and stopping criteria.

A search procedure selects a variable subset from among possible subsets of variables. The goodness of this subset is evaluated with the variable selection (optimality judgement) criterion.
An ideal search procedure would implement an exhaustive search through all possible subsets of variables. In fact, this approach is the only method that ensures finding an optimal solution. In practice, the large number of variables makes an exhaustive search unfeasible. To reduce computational complexity, simplified non-exhaustive search methods are used. Consequently, these methods usually provide only a suboptimal solution.
5.3 Variable selection criteria

There are such variable selection criteria as based on Minimum Concept Description, Inconsistency Count and Interclass Separability. We first consider a criterion called Mutual Information (MI) measure of data sets, based on entropy.

For two variables, MI can be considered to provide a reduction of uncertainty about one variable given the other one. Let us treat MI for classification for a given data set \( L \) consisting of \( p \)-dimensional vectors \( x \) with labeled classes \( \{c_1, \ldots, c_C\} \).
Mutual information criterion

MI for the classification problem is the reduction of uncertainty about classification given a subset of variables. It can be seen as the suitability of the variable subset $S$ for the classification.

If we consider initially only probabilistic knowledge about classes, the uncertainty is measured by entropy as

$$E(c) = - \sum_{i=1}^{C} P(c_i) \log_2 P(c_i)$$

where $P(c_i)$ is the a priori probability of class $c_i$ which may be estimated on the basis of the data set.
Entropy $E(c)$ is the expected amount of information needed for class prediction (classification). It is maximal when a priori probabilities $P(c_i)$ are equal. The uncertainty about class prediction can be reduced by knowledge about $\mathbf{x}$ formed with variables from a subset $S$, characterizing recognised cases or objects and their class membership.

The *conditional entropy* $E(c|x)$, a measure of uncertainty, given vector data $\mathbf{x}$, is defined as:

$$E(c|x) = -\sum_{i=1}^{C} P(c_i|x) \log_2 P(c_i|x)$$
The \textit{conditional entropy} $E$, given the subset $S$ of variables, is defined for discrete variables as:

$$E(c|S) = - \sum_{\text{all } x} P(x) \left( \sum_{i=1}^{C} P(c_i|x) \log_2 P(c_i|x) \right)$$

Using $P(c_i|x) = \frac{P(c_i,x)}{P(x)}$ we obtain:

$$E(c|S) = - \sum_{\text{all } x} P(x) \left( \sum_{i=1}^{C} \frac{P(c_i,x)}{P(x)} \log_2 \frac{P(c_i,x)}{P(x)} \right)$$

$$= - \sum_{\text{all } x} \sum_{i=1}^{C} P(c_i,x) \log_2 \frac{P(c_i,x)}{P(x)}$$
For continuous variables, the outer sum should be replaced by an integral and the probabilities $P(x)$ by the probability density function $p(x)$:

$$E(c|S) = -\int_{\text{all } x} p(x) \left( \sum_{i=1}^{C} P(c_i|x) \log_2 P(c_i|x) \right)$$

Using Bayes’s rule,

$$P(c_i|\mathbf{x}) = \frac{p(x|c_i)P(c_i)}{p(x)}$$
The probabilities $P(c_i|x)$ difficult to estimate can then be replaced by $p(x)$ and $p(x|c_i)$. The initial uncertainty, based on a priori probabilities $P(c_i)$, might decrease given knowledge about $x$. The mutual information $MI(c,S)$ between the classification and the variable set $S$ is measured by a decrease in uncertainty about the prediction of classes, given knowledge about vectors $x$ formed from variables $S$:

$$J(S) = MI(c,S) = E(c) - E(c|S)$$
Since for discrete variables we can derive the equation

\[ J(S) = MI(c, S) = \sum_{\text{all } x} \sum_{i=1}^{C} P(c_i, x) \log_2 \frac{P(c_i, x)}{P(x)P(c_i)} \]

the mutual information is a function of \( c \) and \( x \). If they are independent, the mutual information is equal to zero (knowledge of \( x \) does not improve class prediction).

The mutual information criterion is difficult to use in practice due to difficulties and inaccuracy of estimating conditional probabilities. Continuous variables should be discretized somehow.
These problems surface when the dimensionality of variables is high and the number of cases small. For low-dimensional data, application of the mutual information criterion can be used to choose the best variable subset.

In the simplified application, a greedy algorithm adds one most-informative variable at a time. The added variable is chosen as that which has the maximal mutual information with a class and minimal mutual information with already selected variables. The method does not solve the redundancy problem between groups of variables.
Inconsistency count criterion

Another criterion for variable subset evaluation for *discrete variable* data sets is the *inconsistency count*. Let us consider a given variable subset S and the reduced data set $L_S$ with all $n$ cases. Every case consists of vector $x$ and class $c$ constituted with $m<p$ variables from subset $S$. The inconsistency criterion $J(L_S)$ for data set $L_S$ can be defined as the ratio of all inconsistency counts divided by the number of cases. Two cases, i.e., $m$-dimensional vectors and their classes $(x^k, c^k)$ and $(x^l, c^l)$ are inconsistent if the vectors are identical $x^k = x^l$, but different associated classes $c^k \neq c^l$. (Of course, their original vectors of $p$ variables need not to be identical.)
For the identical vectors $\mathbf{x}^k$ we calculate the inconsistency count $I_k$ of all inconsistent cases for the matching vectors minus the largest number of cases in one of the classes from this set of inconsistent cases. For instance, if there are $q_1$ consistent cases for $\mathbf{x}^k$ in class $c_1$, $q_2$ in $c_2$, ..., and $q_C$ in $c_C$, then $\max\{q_i\}$ is computed and subtracted from the sum of those all. Thus, $I_k$ is obtained as follows:

$$I_k = \sum_{i=1}^{C} q_i - \max_{i=1,\ldots,C} \{q_i\}$$
The inconsistency rate criterion is defined for a reduced data set $S$ as a ratio of sum of all inconsistency counts and a number of all cases $n$ in the data set.

$$ J(S) = \frac{\sum_{\text{all inconsistent vectors}} I_k}{n} $$

Remember that this criterion is only used for discrete variables.
5.4 Search methods

Given a large number of variables constituting a data set, the number of possible variable subsets evaluated by using exhaustive search-based variable selection could be very high, too high to be computationally feasible. For \( p \) variables, \( 2^p \) (including an empty subset) can be formed. For \( p=12 \) variables, the number of subsets is 4096. For \( p=100 \), the number of subsets is greater than \( 10^{30} \), which makes exhaustive search unrelizable. If, for some reason, we are searching for a variable subset consisting of exactly \( m \) variables, we obtain the number of possible subsets:

\[
\binom{p}{m} = \frac{p!}{(p-m)!m!}
\]
For a small number of variables, the exhaustive search could be possible and could guarantee an optimal solution.

**Algorithm: Variable selection based on exhaustive search**

A (learning) data set $L$ with labeled classes is given with $p$ variables $x_1, V=\{x_1, x_2, ..., x_p\}$. A variable selection criterion $J$, e.g., mutual information with a defined computation procedure based on a limited-size data set $T_V$ is used.

1. Set $j=1$ (a counter of the variable subset number).
2. Select a distinct subset of variables $S_j \subseteq V$ (with the number of elements $1 \leq |S_j| \leq p$).
3. For a selected variable subset $S_j$, compute selection criterion $J(S_j)$. 
(4) \( j = j + 1 \)

(5) If \( j \leq 2^p \), continue from step (2); otherwise, go to the next step.

(6) Select an optimal subset \( \hat{S}_{opt} \) with a maximal value of the selection criterion:

\[
J(\hat{S}_{opt}) \geq J(S^j), \quad j = 1, 2, \ldots, 2^p
\]

A sequence of generated distinct variable subset \( S^j \) is not important for the above algorithm. For example, for the three variables \( V = \{x_1, x_2, x_3\} \) one can generate the exhaustive collection of \( 2^3 = 8 \) variable subsets (in practice, the empty subset excluded):

\[
\{ \}, \{x_1\}, \{x_2\}, \{x_3\}, \{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}
\]
The preceding description directly shows that the algorithm produces time complexity $O(2^p)$ possible to execute for small $p$ values only.

There are various variable selection methods that utilise different variable selection criteria. For instance, *Branch and bound* algorithm uses trees where nodes represent different subsets of variables. It is not necessary to explore all nodes possible in the tree, but the process can be made by pruning some branches with suitable computational criteria.
5.5 Variable selection methods with criterion functions

There are also such as *Variable selection with individual variable ranking*, *Variable selection by stepwise forward search*, *Variable selection by stepwise backward search* and *Probabilistic (Monte Carlo) method for variable selection*. The methods, except the third one, are presented in the following.

One of the simplest variable selection procedures is based on first evaluating the individual predictive power of each variable alone, then ranking such evaluated variables, and eventually choosing the best first $m$ variables.
The criterion for an individual variable could be of either filter or wrapper process. The method assumes that variables are \textit{independent} from each other and that the final selection criterion can be obtained as a \textit{sum or product of criteria evaluated for each variable independently}. Because these conditions are infrequently satisfied, the method does not guarantee an optimal selection. A single variable alone may have very low predictive power, but together with another may provide substantial predictive power.
A decision concerning how many best $m$-ranked variables ought to be chosen for the final variable subset could be made on the basis of experience from using another search procedure. Here, one could select the minimal number $\hat{m}$ of best-ranked variables that guarantee a performance better than or equal to a predefined threshold according to a defined criterion $J_{\text{ranked}}$. 
Algorithm: Variable selection with individual variable ranking

A data set \( L \) given with \( n \) cases of labeled classes consists of \( p \) variables \( V=\{x_1, x_2, ..., x_p\} \). There are a variable evaluation criterion \( J_{\text{single}} \) with a defined procedure for its computation based on a limited-size data set \( L_S \) and an evaluation criterion \( J_{\text{ranked}} \) for a final collection of \( m \) ranked variables.

(1) Set \( j=0; \)

(2) Set \( j=j+1 \), and choose a variable \( x_j \).
(3) Evaluate the predictive power of a single variable \( x_j \) alone by calculating the criterion \( J_{\text{single}}(x_j) \).

(4) If \( j < p \), continue from step (2); otherwise, go to the next step.

(5) Rank all \( p \) variables according to the value of computed criterion \( J_{\text{single}} \):

\[
x_a, x_b, \ldots, x_m, \ldots, x_r, J_{\text{single}}(x_a) \geq J_{\text{single}}(x_b), \text{ etc.}
\]

(5) Find the minimal number of first-ranked \( \hat{m} \) variables according to criterion \( J_{\text{ranked}} \).

(6) Select the first \( \hat{m} \) best-ranked variables as a final subset of selected variables.
In principle, the result to be given by the preceding algorithm would be optimal. Nevertheless, the former requirements (p. 194) about independence and joint effect of criteria evaluated are hardly ever entirely valid in real applications.

To reduce the computational burden associated with an exhaustive search, several suboptimal methods have been proposed from which one is *Sequential suboptimal forward variable selection* algorithm.
The task is to select the best $m$-variable subset from $p$, $m<p$, variables constituting the original data set. Fig. 5.3 depicts an example in the tree form of finding an $m=3$ variable subset from $p=4$. Fig. 5.4 gives a graph mapping to the variable space of 4 variables.

A forward selection search begins from evaluations of single variables. For each, a variable selection criterion $J$ is computed and the variable of the maximal value of the performance criterion is chosen for the next step of the search: a ”winner”, root of a subtree. Next, one variable is added to the selected ”winner”, forming all possible two-variable subsets. Each subset of two variables is evaluated, and those giving the maximal increase of criterion are taken. The procedure continues until the best $m$-variable subset.
Fig. 5.3 A sequential forward variable selection search.
Fig. 5.4 Variable space in the form of the graph of 4 weather data set variables.
Algorithm: Variable selection by stepwise forward search

A data set \( L \) given with \( n \) cases of labeled classes consists of \( p \) variables \( V=\{X_1, X_2, \ldots, X_p\} \). There are a number \( m \) of variables in the resulting subset of the best variables and evaluation criterion \( J \) with a defined procedure for its computation based on a limited-size data set \( L_S \).

1. Set an initial "winner" variable subset as an empty set \( S=\{ \} \).
2. Set \( j=1 \).
(3) Form all possible \( p-j+1 \) subsets, with a total of \( j \) variables, that contain a winning \( j-1 \) variable subset \( S_{\text{winner},j-1} \) from the previous step, with one new variable added.

(4) Evaluate the variable selection criterion for each variable subset formed in step \( j \). Select as a winner a subset \( S_{\text{winner},j} \) with a larger increase \( \Delta \) of the performance criterion \( J \) as compared to the maximal criterion value (for the winner subset \( S_{\text{winner},j-1} \)) from the previous step.
(5) If $j=m$, then stop. The winner $S_{\text{winner},j}$ subset in step $j$ is the final selected subset of $m$ variables. Otherwise, set $j=j+1$ and continue from step 3.

The forward selection algorithm provides a suboptimal solution, because it does not examine all possible subsets.

The algorithm assumes that the number of variables $m$ in a resulting subset is known requiring exactly $m$ steps.
An alternative stopping criterion is on the basis of a defined threshold $\varepsilon$ of maximal performance increase for two consecutive steps, i.e., the stopping point is reached when the following criterion increase condition is satisfied:

$$J = J(S_{\text{winner},j}) - J(S_{\text{winner},j-1}) < \varepsilon$$

*Backward selection* is similar to forward selection, but it applies a reversed procedure by starting $p$ variables and discarding these one by one.
The forward and backward search methods can be combined, allowing them to cover more variable subsets through increased computation, and thereby to find better suboptimal variable sets.

Applying Monte Carlo techniques variable selection methods have been developed involving probabilistic search. These random search methods can be used for both filter and wrapper variable selection algorithms.

Probabilistic algorithms are straightforward to implement and guarantee finding the best subset of variables, if a required number of random trials will be performed. They provide satisfactory results for highly correlated variables.
Algorithm: Probabilistic (Monte Carlo) method of variable selection

A data set $L$ given with $n$ cases of labeled classes consists of $p$ variables $V=\{X_1, X_2, ..., X_p\}$. There are a variable subset selection criterion $J$ with a defined procedure for its computation based on a limited-size data set $L_S$ and a maximum number of random subset trials $max\_runs$.

1. Set initially the best-variable subset as equal to an original $p$-variable set $S_{best}=V$. Compute the value of the criterion $J(S_0)=J(L)$ for the entire data set $L$. 
(2) Set \( j = 1 \) (a search trial number).

(3) From all possible \( 2^p \) variable subsets, select randomly a distinct subset of variables \( S_j \) (with number of variables \( 1 \leq m_j \leq p \)).

(4) Create a reduced data set \( L_{S,j} \) with all \( n \) cases constituted with \( m_j \) variables from a subset \( S_j \).

(5) Compute the value of criterion \( J(L_{S,j}) \) for the data set \( L_{S,j} \).

(6) If \( J(L_{S,j}) > J(L_{S,j-1}) \), then set \( S_{best} = L_{S,j} \) and continue from step (7). Otherwise, continue from step (3).
(7) Set $j=j+1$. If $j>max\_runs$, then stop. Otherwise, continue from step (3).

There exists a version of the preceding Monte Carlo method that utilizes the filter technique and inconsistency rate criterion.
5.6 Evaluation of variable importance

If we are able to assess importance of each variable, it is possible to sort them and perhaps weight them giving larger weights for the most important variables and smaller for the least important ones. We can also reduce dimensionality by leaving out those less useful.

A means to assess variable importance is to calculate how much variables of a data set have influence on separation of cases into classes. This, of course, means that the class labels of all cases are known. On the other hand, if they were not known, clustering methods could be applied to form clusters to be used as classes.
In the following a method using nearest neighbor searching is presented for the evaluation of variable importance. We call it Scatter method.\textsuperscript{17}

The main idea of the present method is straightforward. Starting from a randomly selected case the algorithm searches for the nearest neighbor to be the new current case, then it traverses each case one by one in the same way. For every move it checks whether the class of nearest case is from a different class compared with that of the current case. These class changes are counted.

The less there are such changes, the better the cases are separated into different classes. On the other hand, the more changes there are from a class to another, the weaker "concentration" of cases is in the data set. If the data are almost random, i.e., randomly located subject to any areas of classes, there are a great number of those changes. Naturally, such a number also depends on the numbers of cases \( n \) and classes \( C \). In addition, class distribution, i.e., how many cases there are in each class, affects.
The separation of cases into classes can be counted on the basis of individual variables as one-dimensional or all variables in the entire \( p \)-dimensional variable space. To measure separation between classes, a value called separation power from (approximately) \([0,1)\) is computed: the greater separation power, the better classification ability.

Let the number of cases be \( n \), that of classes \( C \), \( 2 \leq C < n \), in data set \( L \).
Scatter algorithm
(1) Preprocessing:
(1.1) Normalize all variable values variable by variable in data set $L$ into scale $[0,1]$.
(1.2) Initialize an empty list $A$ for the class labels of traversed cases.
(1.3) Take a random initial case $a$ from $L$.
(1.4) Search for the nearest case $b$ for $a$ according to the Euclidean distance. If $b$ is not unique, choose randomly from the nearest cases.
(1.5) Insert the class label of case $a$ to the end of list $A$.
(1.6) Remove case $a$ from $L$.
(1.7) Set $a=b$.
(1.8) If $L$ is not empty, return to step (1.4).

(2) Compute class label changes:
(2.1) Initialize change counter $v=0$ and index $i=1$.
(2.2) Take the class labels $l_i$ and $l_{i+1}$ located in the positions $i$ and $i+1$ of $A$.
(2.3) If $l_i \neq l_{i+1}$, then set $v=v+1$. 
(2.4) Set $i = i + 1$.

(2.5) If $i < n$, return to step (2.2).

(3) Compute a scatter value:

(3.1) Compute the theoretical maximum $w$ of changes:

i. Search for $s_{\text{max}}$, the maximum of the class sizes $|c_1|, |c_2|, \ldots, |c_C|$.

ii. If $s_{\text{max}}$ is not unique (several classes of maximum size), then set $w = n - 1$ and go to step (3.2).

iii. Set $m = n - s_{\text{max}}$. % $m$ is the size of jointly all others than the largest class.

iv. If $s_{\text{max}} > m$, then set $w = 2m$ else set $w = n - 1$. 
(3.2) Assign the scatter value: \( s = \frac{v}{w} \) which is from \((0,1]\).

(4) Compute a statistical baseline \( z \):

(4.1) Prepare a simulated data with the class label distribution of the original data set, but random numbers as variable values. Repeat steps (1)-(3) for several times, say 50, with simulated data sets and calculate scatter values and their mean \( z \).

(5) Compute separation power \( f \): Set \( f = z - s \).
Note that we could still insert an outer, main loop for the entire algorithm so that it would be repeated, e.g., 10 times and a mean of the 10 separation power values is computed. This is necessary, because the starting case is chosen randomly and resulting separation power may depend slightly on that choice.

Note also that other distance measures than Euclidean metric could be applied to its nearest neighbor searching.

Scatter values come from interval $[(C-1)/w,1]$, since there is at least 1 change, between the two classes of a data set. Time complexity of the algorithm is $O(n^2)$. 
Example 1

There are four different ways how Scatter algorithm can be utilized as follows.

Let us recall Vertigo data set from Chapter 2, where Table 2.1 showed its main 38 variables. To exemplify the use of Scatter algorithm we return back to Vertigo with the data of 815 patients from six disease classes. There were approximately 11% missing values first imputed with the class modes of the binary variables and the class medians of the other variables.

(1) The first way to use Scatter algorithm was to run it for the entire data set as described exactly in the preceding algorithm repeating it 10 times.
A mean and standard deviation of scatter values were 0.518±0.011. (Henceforth, standard deviations are not given because they were small.) Separation power was 0.243 on the average, a fairly great value between minimum 0.006 and average baseline value (with random data) 0.761 obtained. This showed that classification is a sensible task for Vertigo data set.

(2) The second way to use Scatter algorithm is to run all the data classwise, in other words, by putting each class to oppose its “counter class” of all other classes.
This produced means of separation powers 0.45, 0.296, 0.186, 0.167, 0.097 and 0.194 for the six classes of vestibular schwannoma, benign positional vertigo, Menière’s disease, sudden deafness, traumatic vertigo and vestibular neuritis. These predicted that the first class would be the easiest to classify and the fifth the most difficult. The sizes of the six classes were 16%, 18%, 38%, 5%, 8% and 15% of 815 cases.
(3) The third way to apply Scatter algorithm is to run its variables one by one for the entire data. This means that the algorithm is executed separately for each of \( p \) variables. Fig. 5.5 presents the separation power results gained.\(^{18}\) There are clear differences between the variables.

(4) The fourth way is to run the algorithm for single variables and for each class vs. the other classes. Results for this way are illustrated in Fig. 5.6.\(^{18}\) This denotes that importances of variables depend on the classes. For instance, variable 2 is important for the first, third and sixth classes.

Fig. 5.5 Separation powers of 38 variables of Vertigo data set. Variables 1, 2, 3, 4, 7, 11, 14 and 37 received the highest values subject to the entire data.
Fig. 5.6 Separation powers of single variables for different classes: vestibular schwannoma (VS), benign positional vertigo (BPV), Menière’s disease (MD), sudden deafness (SD), traumatic vertigo (TV) and vestibular neuritis (VN).
Fig. 5.6 predicted that few variables would be important for the small fourth and fifth classes. The reason may be medical and, particularly, their scarcity in Vertigo data, only 5% and 8% of cases as the smallest ones. Still, some variables were very important for them. Especially variable 14 was such for the fourth class of sudden deafness, and it is known to be essential for diagnosing this disorder. Variable 21 was very important for the fifth of traumatic vertigo. In Fig. 5.5 it was not among the highest, since this disease represented 8% of cases only.
Note that if a data set comprises only two classes, this yields that as to ways (1)-(4) to run Scatter algorithm (1) and (2) as well as (3) and (4) are identical, because the "counterclass" of each class contains only one class.
Example 2

Let us look back at the variables of female urinary incontinence in Table 3.1 from Chapter 3. Altogether, there were 13 variables after removing three. Eight of 13 variables were originally binary, and the other five were binarized so that it was possible to compute mutual information for all. Missing values were imputed as previously, with modes of the variables. For this special data, binarization was very natural, because the medical experts used their threshold values seen very good in practice to divide values of each variable into the two opposite categories.
The main data set of 529 patients was employed for running both mutual information and Scatter algorithm. In Table 5.1 the comparison of the results is given. The class distribution was skewed, when the largest disease class included 61% of all cases, then three other disease classes 26%, 6% and 3% and the fifth class of the normal (healthy) 3%.

Variables 13, 4 and 2 in this order are the most important. Variables 1, 3, 11 and 12 are the least important. For all 13 variables, Spearman ranking order correlation was 0.87, highly positive with p equal to 0.0001.
Table 5.1 Descending importance order of variables according to Mutual information and Scatter algorithm. The variables reported by the medical experts in advance to be the most important (variables 2, 4, 10 and 13) are green and the least important (variables 1, 11 and 12) are red.

<table>
<thead>
<tr>
<th>Variable</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutual information</td>
<td><strong>13</strong></td>
<td>4</td>
<td>2</td>
<td>9</td>
<td><strong>10</strong></td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td><strong>11</strong></td>
<td>3</td>
<td>1</td>
<td><strong>12</strong></td>
</tr>
<tr>
<td>Scatter algorithm</td>
<td><strong>13</strong></td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td><strong>10</strong></td>
<td>9</td>
<td>7</td>
<td>6</td>
<td><strong>12</strong></td>
<td>3</td>
<td>1</td>
<td><strong>11</strong></td>
</tr>
</tbody>
</table>