

Chapter 2

All Relevant Feature Selection Methods and Applications

Witold R. Rudnicki, Mariusz Wrzesień and Wiesław Paja

Abstract All-relevant feature selection is a relatively new sub-field in the domain of feature selection. The chapter is devoted to a short review of the field and presentation of the representative algorithm. The problem of all-relevant feature selection is first defined, then key algorithms are described. Finally the Boruta algorithm, under development at ICM, University of Warsaw, is explained in a greater detail and applied both to a collection of synthetic and real-world data sets. It is shown that algorithm is both sensitive and selective. The level of falsely discovered relevant variables is low—on average less than one falsely relevant variable is discovered for each set. The sensitivity of the algorithm is nearly 100 % for data sets for which classification is easy, but may be smaller for data sets for which classification is difficult, nevertheless, it is possible to increase the sensitivity of the algorithm at the cost of increased computational effort without adversely affecting the false discovery level. It is achieved by increasing the number of trees in the random forest algorithm that delivers the importance estimate in Boruta.

Keywords All-relevant feature selection · Strong and weak relevance · Feature importance · Boruta · Random forest

W.R. Rudnicki (✉)
Interdisciplinary Centre for Mathematical and Computational Modelling,
University of Warsaw, Pawińskiego 5A, 02-106 Warsaw, Poland
e-mail: W.Rudnicki@icm.edu.pl

M. Wrzesień · W. Paja
Faculty of Applied IT, University of Information Technology and Management,
Sucharskiego 2, 35-225 Rzeszów, Poland
e-mail: mwrzesien@wsiz.rzeszow.pl

W. Paja
e-mail: wpaja@wsiz.rzeszow.pl

2.1 Introduction

The usual goal of feature selection in machine learning is to find the best set of features that allows one to build useful models of studied phenomena. The chapter is devoted to a different application of feature selection process, where building a machine learning model is merely a tool for extracting all features that are relevant for a problem. The relevance is considered in a broad sense—it is sufficient for a feature to be declared relevant, when it is useful for building a machine learning model of the problem under scrutiny at some context. One may ask why this goal is *relevant* at all? Why should anyone be interested in this type of relevance?

Let us firstly describe a toy problem that illustrates a need for the all-relevant feature selection in an artificially transparent setting. Let us construct a system containing 100 objects described with one hundred real-valued variables X_1, \dots, X_{100} , and one binary decision variable D . The descriptive variables X_1 and X_2 are drawn from a normal distribution $N(0, 1)$. The value of the decision variable is determined from values of these variables in the following manner. It is one (**TRUE**) if both variables have the same sign and is zero (**FALSE**) if their signs differ. The descriptive variables X_3, \dots, X_{10} are obtained as a linear combination of X_1 and X_2 , and normalised to $N(0, 1)$. The variables X_{11}, \dots, X_{100} , are drawn from a normal distribution $N(0, 1)$. Finally the indexes of the variables are randomly permuted. The goal of the researcher is to determine which variables are responsible for the value of a decision variable.

There is a very easy path to the solution of this problem. One could take a classifier that is able to rank feature importance and select two most important features. Unfortunately this path may lead us astray, as displayed in Fig. 2.1, that shows the ranking of feature importance for our toy problem returned by a random forest (RF) [2] classifier. Here, for clarity, variable indexes are not permuted.

The toy problem is simple enough that it can be solved directly by a brute force approach. It is sufficient to build 4,950 models including two variables to find one that gives perfect classification and hence is the most likely to be built on two variables used to generate the model. However, for real life problems a number of descriptive variables may be much larger, connections between these variables and decision may be more complicated, measurements are subject to noise. Moreover, one does not know beforehand how many variables influence decision. Finally, while for our toy problem the model based on two variables used to generate the model usually gives best results, this is not guaranteed to work in a general case. Hence the brute force approach will not work in most cases.

As an example of a real-life application we may consider deciphering connection between gene expression levels in humans with some medical condition. In this case a number of variables is roughly twenty thousands, it is not known how many genes are involved and how, and last but not least—measurements are subject both to normal variability and experimental error. Analysis of such problem can be split into two separate tasks: determination which variables are connected in some way with the decision variable, and then identification of those variables that are responsible for

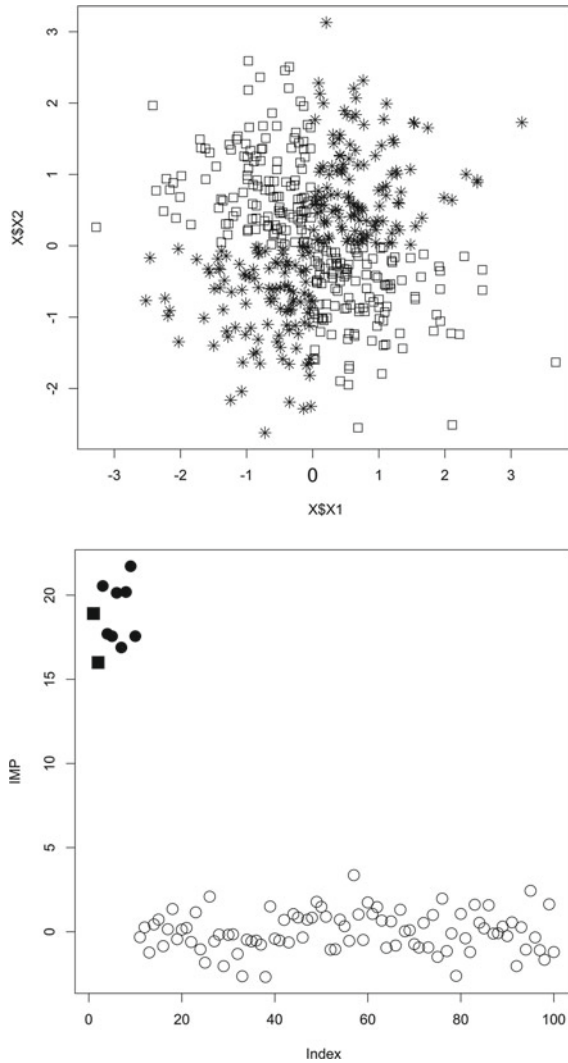


Fig. 2.1 The illustration for the toy problem. The *upper panel* shows projection of the system on the plane (X_1X_2) . The *lower panel* shows the importance of variables in random forest classifier. *Solid squares* correspond to variables used to generate the decision variable, *solid circles* correspond to combinations of X_1 and X_2 and *open circles* correspond to random variables. It is clear that importance of variables obtained from random forest can be used to discern informative and non-informative features. Nevertheless, the importance ranking does not allow to detect the variables used for generation of the decision variable

a value of the decision variable. The first task can be tackled using the all-relevant feature selection approach. The solution of the second task, which generally is much

harder, should be easier when all relevant variables are identified and hence the number of variables is reduced.

For example in our toy problem a perfect algorithm for all-relevant feature selection should find that 10 variables out of 1,000 are somehow connected with decision variable, therefore the number of models tested in the brute force approach can be reduced to 45. In a medical problem of a researcher studying a connection between gene expression and a medical condition, a number of genes to consider may be reduced from multiple thousands to hundreds or tens, or maybe even a handful of variables. A domain specific knowledge can be then applied to build a model of a problem under scrutiny.

2.1.1 Definitions

Up to this point the notion of relevance was used without definition, instead we relied on its intuitive understanding. However, it has been already observed by Kohavi and John [7] that there are several definitions of relevance that may be contradictory and misleading. They proposed that two degrees of relevance (strong and weak) are required to encompass all notions that are usually associated with this term. In their approach the relevance is defined in the absolute terms, with the help of an ideal Bayes classifier.

Definition 1 A feature X is *strongly relevant* when removal of X alone from the data always results in deterioration of the prediction accuracy of the ideal Bayes classifier.

Definition 2 A feature X is *weakly relevant* if it is not strongly relevant and there exists a subset of features S , such that the performance of ideal Bayes classifier on S is worse than the performance on $S \cup \{X\}$.

Definition 3 A feature X is *irrelevant* if it is neither strongly nor weakly relevant.

One should note, that an information system might be constructed in such a way, that there are no strongly relevant attributes. Indeed, it is easy to notice that the toy system described above does not contain strongly relevant attributes.

Another useful notions were introduced by Nilson et al. [13], who used concepts of weakly and strongly relevant features to define formally two problems of feature selection. A *minimal optimal problem* in feature selection has the goal to find the minimal set of attributes giving the best possible classifier. The other is an *all relevant problem*, where one is interested in finding all strongly and weakly relevant attributes.

Definition 4 (*Minimal optimal problem*) Find a set of attributes consisting of all strongly relevant attributes and such subset of weakly relevant attributes, that all remaining weakly relevant attributes contain only redundant information.

Definition 5 (*All-relevant problem*) Find all strongly relevant and all weakly relevant attributes.

It has been shown by Nilsson and co-workers, that exact solution of the all relevant problem requires an exhaustive search, which is intractable for all but smallest systems.

The relevance defined earlier is a qualitative notion—a feature can either be relevant or irrelevant. It is also an objective property of the system under scrutiny, independent from the classifier used for building a model. This notion is distinct from *importance of variable*, that is a quantitative and classifier-dependent measure of the contribution of a variable to a model of the system. One can use various measures of importance of variable, provided that they satisfy the simple condition—the importance of relevant variables should be higher than importance of irrelevant ones. A useful and intuitive measure of importance was introduced by Breiman in random forest (RF) classification algorithm [2].

Definition 6 (*Importance of a variable*) is the loss of the classification accuracy of the model that was built using this variable, when the information on the variable's value is withdrawn.

A final concept that will be used often enough in the current chapter to deserve a mention in this section is a *contrast variable*.

Definition 7 (*Contrast variable*) is such descriptive variable that does not carry information on the decision variable by design.

It is added to the system in order to discern relevant and irrelevant variables. It may be obtained by drawing from theoretically justified probability distribution e.g. normal or uniform; it may be also obtained from real variables by random permutation of their values between objects. Application of contrast variables for feature selection was first proposed by Stoppiglia et al. [15] and then independently by Tuv et al. [17], and Rudnicki et al. [14].

One may notice, that any all-relevant feature selection algorithm is a special type of classification algorithm. It assigns variables to two classes: *relevant* or *non relevant*. Hence the performance of the algorithms can be measured using the same quantities that are used for estimation of ordinary classifiers. Two measures are particularly useful for estimation of performance: sensitivity S and positive predictive value PPV . Sensitivity S is measured as

$$S = TP / (TP + FN), \quad (2.1)$$

where TP is a number of truly relevant features recognised by an algorithm, FN is a number of truly relevant features that are not recognised by an algorithm and FP is a number of non relevant features that are incorrectly recognised as relevant. Positive predictive value PPV is measured as

$$PPV = TP / (TP + FP). \quad (2.2)$$

2.1.2 Algorithms for All-Relevant Feature Selection

There are two issues that are non-existent for the *minimal optimal* problem, but are very important for the *all relevant* one. The first one is detection of weakly relevant attributes that can be completely obscured by other attributes, the second one is discerning between weakly but truly relevant variables from those that are only seemingly relevant due to random fluctuations.

The concepts of strong and weak relevance, and consequently also the problem of *all relevant* feature selection, are defined in a context of a perfect classifier that is able to use all available information. Yet, in real-world applications one is restricted to use imperfect classification algorithms, that are not capable of using all information present in the information system, and this may influence the outcome of the feature selection algorithm. In particular, an algorithm may not be able to find and use some of the relevant features. In many cases this will not disturb solution of the *minimal optimal* problem, provided that final predictions of a classifier are sufficiently accurate; yet it will significantly decrease a sensitivity of an *all relevant* feature selection. Hence a classification algorithm used in *all relevant* feature selection should be able to detect weak and redundant attributes.

Algorithms that may be used for finding all the relevant features [3, 6, 9, 14, 17] are designed around ensembles of decision trees, either using the random forest algorithm [2] or an algorithm specially tailored for the task. The choice of decision trees as base learners is due to their flexibility and relative robustness, when multiple redundant features are present in the data set. Moreover, the estimate of the variable importance is easily obtained for tree-based ensembles.

The second issue, namely discerning between the truly and randomly relevant attributes arises because the analysis is performed for finite size samples. This gives a chance for random correlations to emerge and significantly influence the results. The probability of such an event increases with the decreasing number of objects; the effect is also boosted by overall large number of attributes, which in addition increases chances for random interactions between features. This issue is handled by introducing ‘contrast variables’ which are used as a reference. A statistical test is performed that compares the importance of original variables with that of contrast variables.

Contrast variables have been used to find all relevant variables by four independent groups. Tuv et al. [17] in ACE algorithm used ensembles of shallow classification trees and iterative procedure in which the influence of the most important variables on decision was removed in order to reveal variables of secondary importance. In each step only these variables that were more important in the statistical test than the 75th percentile of contrast variables were deemed important.

Rudnicki et al. [14] introduced Boruta algorithm that used the importance estimate from the random forest. The algorithm started by establishing initial ranking of variables in random forest. Then the algorithm performed an iterative procedure in which the least important variables were consecutively turned into contrast variables by permuting their values between objects. Then the threshold level was increased by a predefined step and procedure was repeated until the self-consistence was achieved.

The procedure was carried out until the importance of all contrast variables was lower than that of the unperturbed variables.

Huynh-Thu et al. [6] independently proposed a procedure that aimed at the same goal from another end. In this approach the procedure starts similarly from establishing the ranking of importance from RF. Then the algorithm estimates the importance of noninformative variables by turning all variables into contrast variables. In the following steps the algorithm iteratively introduces back the informative variables into the information system and computes the importance of both i informative variables (the original most important variables) and $N - i$ noninformative variables.

Dramiński et al. [4] introduced the MCFS algorithm to improve a feature ranking obtained from an ensemble of decision trees. It was constructed in such a way that eliminated known bias of random forest towards variables with fewer number of values. The algorithm was later extended for use as an all-relevant feature selection algorithm [3] by introducing a comparison of the importance of variables with the maximal importance obtained from a set where all variables were uninformative.

The second version of Boruta [9] was introduced to improve computational efficiency and used a different heuristic procedure. In this version the original dataset is extended with random contrast variables. For each original attribute a 'shadow' attribute is generated by randomly permuting its values. Then, for each attribute, it is tested whether its importance is higher than the maximal importance achieved by a contrast attribute. In order to obtain statistically significant results this procedure is repeated several times, with contrast variables generated independently for each iteration. After each iteration, the algorithm checks how many times the importance of tested attributes is higher (or lower) than that of the highest ranked contrast variable. Once this number is significantly higher than allowed under hypothesis of equality with importance of highest random contrast, the attribute is deemed relevant and not tested further. On the other hand, if this number is significantly lower than allowed under the same hypothesis, then the attribute is deemed irrelevant and permanently removed from the data set. The corresponding contrast variables can be either retained or removed from the dataset; the former choice increases precision of the result, whereas the other greatly improves computational efficiency. The algorithm is terminated when either the relevance of all attributes is established or until predefined number of steps is executed. The result of the algorithm is the assignment of each variable to one of three classes—relevant, irrelevant, unresolved (or tentative). The final decision about the unresolved (tentative) attributes is left to the user.

All these algorithms are quite similar to each other: they are based on the ensemble of trees, they use similar measures of importance and use contrast variables to discern relevant and non relevant attributes. They differ mostly in implementation of the statistical test as well as in performance. The current study is devoted to detailed analysis of performance of Boruta algorithm for a family of synthetic data sets with varying number of truly relevant variables and total number of variables, and hence varying difficulty. The difficulty is measured as the error level of the random forest classifier built on the truly relevant variables. The small scale tests performed by us have shown that results of these algorithms are also similar, hence we believe that the analysis performed for single algorithm will be relevant also for other algorithms.

2.1.3 *Random Forest*

The random forest algorithm is used in the current work both as a classifier and as an engine for the feature selection algorithm, hence we give below a short summary of its most important qualities. It is designed as an ensemble of weak classifiers that combine their results during the final classification of each object. Individual classifiers are built as classification trees. Each tree is constructed using different bootstrap sample of the training set, roughly $1/3$ of objects is not used for building a tree. At each step of the tree construction a different subset of attributes is randomly selected and a split is performed using an attribute which leads to a best distribution of data between nodes of the tree.

Each object has not been used by roughly $1/3$ of trees. This object is called ‘out of bag’ (OOB) for these trees, and they are the OOB trees for this object. One may perform (OOB) error estimate by comparing the classification of the ensemble of the OOB trees for each object with the true decision. The OOB object can be used also for estimation of variables’ importance using following procedure. For each tree all its’ OOB objects are classified and the number of votes for a correct class is recorded. Then values of the variable under scrutiny are randomly permuted across objects, the classification is repeated and the number of votes for a correct class is again recorded. The importance of the variable for the single tree can be then defined as a difference between a number of correct votes cast in original and permuted system, divided by number of objects. The importance of the variable under scrutiny is then obtained by averaging importance measures for individual trees. The implementation of random forest in R library [11] is used in Boruta and also was used for classification tasks.

2.2 Testing Procedure

Boruta algorithm is a wrapper on the random forest, hence it is likely that quality of feature selection depends on the quality of random forest model. Therefore in the first step of the testing procedure we performed a series of tests of the random forest algorithm itself on synthetic data sets. Then the performance of the all-relevant feature selection algorithm was examined on the selected synthetic data sets as well as on few real-world data sets.

2.2.1 *Data Sets*

Synthetic data sets were constructed as variants of the well known hypercube problem. In this problem a set of points are generated in corners of D -dimensional hypercube, each coordinate of the corner is either $+1$ or -1 . The corners of the hypercube were assigned to one of two classes using two methods. The first one relies on random process. Corners of a hypercube are numbered $1, \dots, 2^D$, then a random sample of length $2^{(D-1)}$ is drawn from the range $(1, \dots, 2^D)$ and corners with these numbers are

assigned to class 1; the remaining corners assigned to class 2. The second method is deterministic. Corners with odd number of -1 coordinates are assigned to class 1 and the remaining corners are assigned to class 2. The points were generated using three methods. In the first one, the points are generated from multidimensional Gaussian distribution with mean zero and standard deviation one and assigned to the nearest corner. In the second method, the multidimensional uniform distribution spanned on $(-1, 1)$ interval was used instead of Gaussian. In the third one, points were drawn from 2^D multidimensional Gaussian distributions with standard deviation 0.1, each centred on the respective corner of the hypercube. Then two classes of additional features were added to each data set. Features from the first class were obtained as a linear combination of original variables. Features from the second class were drawn randomly from the normal distribution. As a result we obtain the data set described with three types of features. The *generative features* are the original variables used to define the value of decision variable. The *combination features* are obtained as linear combinations of generative features and hence they are also connected with decision variable. These two sets of features are by definition relevant. One should note, however, that features of both types are weakly relevant—it is possible to replace any of the features with combination of other features. The remaining variables are *random features*—they are not connected with decision variable.

Multiple data sets with varying numbers of generative, combination and random features, as well as varying number of objects were generated using four combinations of the class assignment and point distributions methods, resulting in four series of data sets. The first series, denoted as *NORM* used the deterministic class assignment and single Gaussian in a centre for generation of data points. The second one, denoted as *UNI* used deterministic class assignment and uniform distribution of points, the third one used random class assignment and uniform distribution of points. The last series was obtained using random class assignment and Gaussians centered on corners of the hypercube for points generation. Two last series were generated with functions *mlbench.xor* and *mlbench.hypercube* from the *mlbench* package [10] in R [16], with default parameters for data dispersion and are denoted as *XOR* and *HYPER*.

In addition to analysis of synthetic data sets the relevance of the variables was examined for four recently published data sets deposited in the UCI repository [1]: MicroMass, QSAR biodegradation (Q-b) [12], Türkiye Student Evaluation Data Set (TSE) [5] and Amazon Commerce Reviews Set (ACRS).

2.2.2 Classification

The tests of classification accuracy for random forest were performed for four series of data sets described earlier. However, the data sets used for survey of classification results were simpler than those used for feature selection. The number of generative variables varied between 2 and 8, the number of combination features was either zero or two times the number of original features and the number of objects varied between 100 and 2,000. The systems were not extended with random features—the goal of this survey was to find the region of parameter space that is feasible for

classification in the best settings, without additional noise from random features. The random forest implementation in R [11] was used to perform classification, using the default parameters: 500 trees in ensemble and the number of variables used for split generation set at the square root of the total.

2.2.3 Feature Selection

Two series of data sets were selected for further analysis with Boruta feature selection algorithm implemented in a package in R [8]. In higher dimensions both functions using deterministic class assignment generated data sets that were too difficult for the random forest algorithm, hence only sets generated with the help of two functions from *mlbench* package that were based on random were used in feature selection test.

The result of the classification testing has shown that the quality of models depends monotonically on the number of objects—the OOB classification error decreases with increasing number of objects. This relationship was universal, but in some cases the number of objects required to obtain model of good quality was very high. This is especially true for the high dimensional problems. Therefore to reduce the number of variable parameters we fixed the number of objects at single value 500, that allowed us to scan a wide range of difficulties for numbers of variables varying between 50 and 10,000.

The tests were performed for the following grid of parameters describing data sets: $N_{gen} = (2, 3, 4, 5)$ generative variables $\times N_{comb} = (5, 10, 20, 50, 100, 200, 500)$ combination variables $\times N_{all} = (50, 100, 200, 500, 1,000, 2,000, 5,000, 10,000)$ all variables, where $N_{all} = N_{gen} + N_{comb} + N_{rand}$ (and N_{rand} is a number of random variables). Obviously, the grid points corresponding to negative number of random variables were not explored. The number of variable parameters in the test is four, therefore generation of every possible combination is neither feasible nor interesting. Data sets that are either very easy or very difficult are not interesting for further analysis. For the purpose of this work the data set was considered easy when the OOB estimate of the classification error of random forest model is below 2% and it is considered hard when the OOB error is above 30%. Therefore only a subset of possible datasets within the range of parameters was generated and tested. For each number of generative variables the initial test system was generated that comprised of 500 objects with 5 combination features and 50 random features. Then the number of objects, combination features and random features was varied until either easy or hard region of the parameter space was found.

Additionally, the influence of number of trees in the forest on the feature selection procedure was examined. To this end, the entire procedure was repeated using random forest classifiers obtained with three different numbers of trees, namely 500, 1,000 and 2,000.

Despite fixing the number of objects and examining only two set series of data sets, the number of possible combinations was still too high to be practical, hence not all of the possible grid points were examined. The set of combinations examined is given in Table 2.1.

Table 2.1 Data sets generated for the all relevant feature selection analysis

# objects	# generative variables	# combination variables	Total variables	# Trees
500	(2, 3, 4, 5)	(5, 10, 20, 50)	100	(500, 1,000, 2,000)
500	(2, 3, 4, 5)	(5, 10, 20, 50, 100)	200	(500, 1,000, 2,000)
500	(2, 3, 4, 5)	(5, 10, 20, 50, 100, 200)	500	(500, 1,000, 2,000)
500	(2, 3, 4, 5)	(5, 10, 20, 50, 100, 200)	1,000	(500, 1,000, 2,000)
500	(2, 3, 4, 5)	(5, 10, 20, 50, 100, 200)	2,000	(500, 1,000, 2,000)
500	(2, 3, 4, 5)	(5, 10, 20, 50, 100, 200)	5,000	(500, 1,000, 2,000)

In contrast with the synthetic data sets, information on true relevance of variables is unknown for real-world data sets. Therefore, we can measure directly neither sensitivity nor PPV of the algorithm. However, we can estimate the PPV using contrast variables, by measuring how many of them algorithm deems relevant. To this end, we generate contrast variables as ‘shadows’ of original variables, which are obtained by copying values of original variables and randomly permuting them between objects. Each variable is accompanied by a shadow variable. The system extended in this way is then analysed with the Boruta algorithm. Then the PPV estimate is obtained as

$$PPV^* = \frac{N_{relevant}(X_{original})}{N_{relevant}(X_{original}) + N_{relevant}(X_{contrast})}, \quad (2.3)$$

where PPV^* denotes approximate PPV , $N_{relevant}(X_{original})$ and $N_{relevant}(X_{contrast})$ are respectively a number of original and contrast variables that algorithm has deemed relevant. Entire analysis was repeated five times to check robustness of the results. Boruta algorithm assigns variables to three classes: (*Confirmed*, *Tentative*, *Rejected*). One can treat the *Tentative* class either as relevant or irrelevant, hence two measures of PPV^* were used, PPV_c^* and PPV_t^* that differed in the assignment of the *Tentative* variables. The former assigns them to irrelevant, whereas latter to relevant class.

2.3 Results and Discussion

Four series of datasets were generated using small variations of the same approach, nevertheless, the results differ significantly for these sets. Two series of synthetic data sets generated with deterministic class assignment were generally difficult to classify with random forest algorithm. The classification results for these sets were satisfactory (OOB error less then 30 %) only for low dimensional problems (2 and 3). The problems of higher dimensionality were solvable only when large number of objects was available. Therefore further analysis for synthetic sets was performed for two remaining series.

2.3.1 Classification

The results of the classification survey are in general agreement with intuitive expectations, see Fig. 2.2. Increasing the dimensionality of the problem makes it more difficult, adding noise to the problem makes it more difficult, and increasing the number of objects helps in building better models.

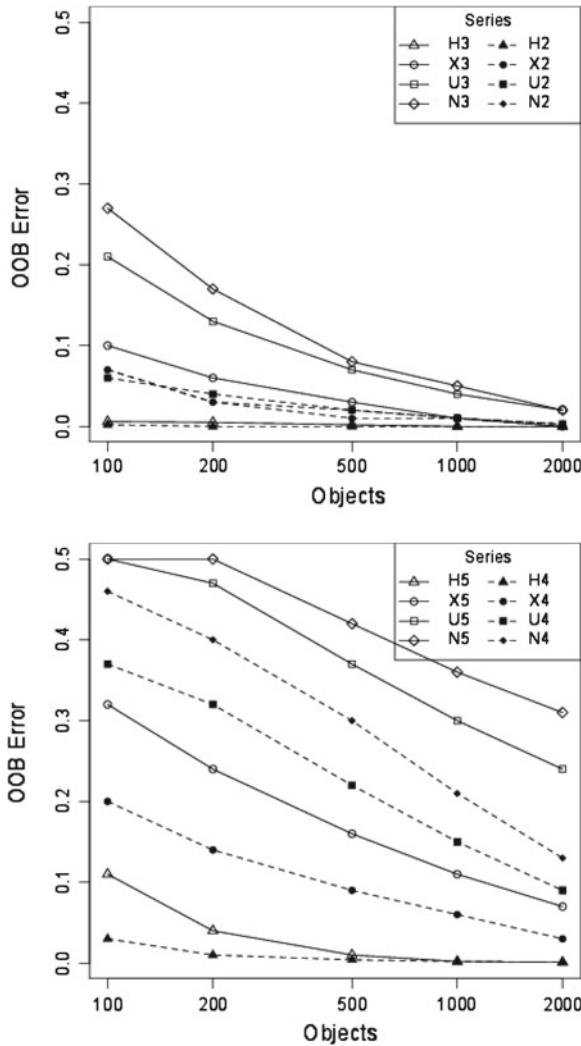


Fig. 2.2 Classification results for four variants of hypercube problem. *Top* 2D and 3D models. *Bottom* 4D and 5D models. Labels for series are constructed from the first letter of the series name and dimension of the problem, for example X3 denotes three dimensional data sets from XOR series, H5 denotes five dimensional data sets from HYPER series etc

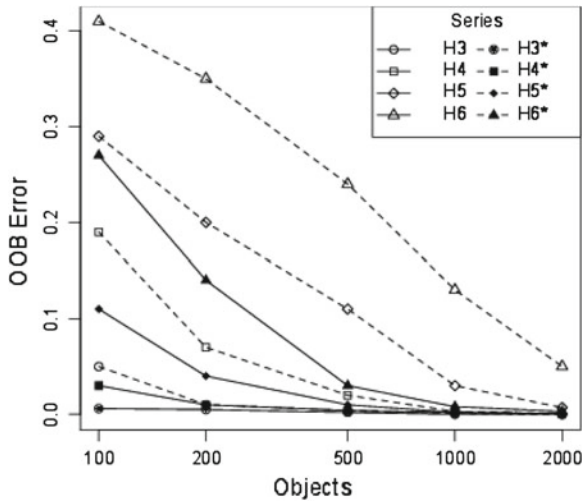


Fig. 2.3 The OOB error for two series of HYPER data sets, one with points described with generative variables only, and the other with additional combination features. The number of combination features is two times the number of generative features. The datasets with combination features are marked with a ‘*’

One result that may be less intuitive is that introduction of features that are linear combinations of original variables may improve the classification. These features in some cases may form lower dimensional subspace that allows to separate clusters located originally in corners of the hypercube. This is not universal, but observed for the last series. Hence presence of the combination features in the data set may facilitate transition of a problem that is formally N -dimensional to easier $(N-k)$ -dimensional one. The effect is displayed in Fig. 2.3. The classification error is significantly lower for series with original generative features augmented with linear combinations. This result shows that relationship between importance and true relevance may not be straightforward.

2.3.2 Feature Selection

In line with expectations the results of the feature selection are correlated with the results of the classifications. It is difficult to identify important features for data sets that are difficult to classify and relatively easy for those that are easy to classify. This is clearly visible in Table 2.2 that collects the overall results of the survey of the synthetic data sets. For the XOR series the sensitivity is very high for easy 2-dimensional data sets and drops to 25 % for hard 5-dimensional data sets. On the other hand, the level of false discovery is uniformly low—the expected value of false positive discovery is 0.3. It means that on average only 3 falsely relevant variables

Table 2.2 The cumulative results for the XOR and HYPER series of data sets

DIM	XOR					HYPER				
	TP	FP	FN	Sensitivity (%)	PPV (%)	TP	FP	FN	Sensitivity (%)	PPV (%)
2	51.2	0.3	0.2	98	98	–	–	–	–	–
3	42.1	0.2	8.2	81	97	50.5	0.1	1.3	97	99.7
4	37.0	0.3	15.0	65	99	52.2	0.1	0.7	96	99.6
5	14.2	0.3	36.2	23	97	47.7	0.1	5.3	91	99.0

The average number of false positive, false negative, sensitivity and PPV were computed for the entire range of parameters

Table 2.3 Cumulative results for four dimensional data set

Ntotal	Mean TP	Mean FP	Mean FN	Mean sensitivity (%)	Mean PPV (%)
100	24.5	0.7	0.8	91.7	97.4
200	40.1	0.5	0.9	90.2	98.7
500	61.6	0.1	6.6	81.5	99.8
1,000	52.4	0.3	15.7	53.6	99.4
2,000	48.2	0.1	19.9	59.7	99.9
5,000	29.7	0.1	42.6	33.8	99.6
10,000	25.6	0.1	57.2	34.2	99.6

The average number of true and false positive, false negative, sensitivity and PPV are displayed for varying number of total variables. The averaging was performed over variable number of combination variables

should be expected in 10 runs of Boruta algorithm. Both sensitivity and PPV are very high for the sets in the HYPER series, hence deeper analysis is devoted to the more difficult XOR series.

The four-dimensional data sets are examined in closer detail in the Table 2.3, where the results for a range of total number of variables is presented. It is clear that the sensitivity of the algorithm drops with increasing number of variables, in line with the number of false positive discoveries.

The drop in sensitivity with increasing number of variables is expected behaviour. When the number of variables is large, the chance for a variable to be included in a tree in few first splits is diminished, hence the impact of individual variable is a subject to larger variability when compared with systems with a small number of variables. Therefore it is more difficult to discern relevant variables with lesser impact from random ones. This effect can be circumvented by increasing a number of trees in the system; see Table 2.4, where cumulative data for all four-dimensional sets is presented as well as a more detailed analysis of a five-dimensional set.

Another interesting effect is presented in Table 2.5. Systems with different number of relevant variables have variable behaviour of sensitivity when the number of variables is increasing. For example, when the total number of variables is 500, the sensitivity is 100 % for a system with 54 relevant variables, whereas it is 87 % for a system with 204 relevant variables. When the number of random variables is

Table 2.4 Change of sensitivity as a function of a number of trees in Boruta

Ntree	Average for 4D systems				Average for 5D systems			
	TP	FN	Sensitivity (%)	PPV (%)	TP	FN	Sensitivity (%)	PPV (%)
100	–	–	–	–	40.2	164.8	19.6	100
200	–	–	–	–	70.2	134.8	34.2	100
500	34.9	21.1	57.8	99.7	123.6	81.4	60.3	100
1,000	39.9	16.0	63.5	99.5	157.6	47.4	76.9	100
2,000	43.4	12.5	68.5	98.9	180.4	24.6	88.0	99.8
5,000	–	–	–	–	189.6	15.4	92.5	99.5
10,000	–	–	–	–	193.2	11.8	94.2	99.4
20,000	–	–	–	–	195.6	9.4	95.4	99.4

The average results for all 4-dimensional systems examined with Boruta using 500, 1,000, and 2,000 trees are shown in the left panel. The more detailed inspection of results for sets described with 5 generative, 200 combination and 1,000 total variables is presented in the right panel. Average results for five instances are presented for Boruta using 100 to 20,000 trees

Table 2.5 Results of feature selection presented for two series of 4-dimensional data sets for varying total number of variables in the system

Ncomb	Ntotal	Mean TP	Mean FP	Mean FN	Mean sensitivity (%)	Mean PPV (%)
50	100	54	0.0	0.0	100.0	100.0
	200	54	0.7	0.0	100.0	98.8
	500	54	0.0	0.0	100.0	100.0
	1,000	46.7	1.3	7.3	86.4	97.2
	2,000	52.0	0.3	2.0	96.3	99.4
	5,000	21.7	0.0	32.3	40.1	100.0
	10,000	10.7	0.0	43.3	19.8	100.0
200	500	176.7	0.0	27.3	86.6	100.0
	1,000	173.3	0.0	30.7	85.0	100.0
	2,000	145.7	0.0	58.3	71.4	100.0
	5,000	112.7	0.0	91.3	55.2	100.0
	10,000	84.7	0.0	119.3	41.5	100.0

The average number of true and false positive, false negative, sensitivity and PPV are displayed. The averaging was performed over Random Forest models built from 500, 1,000, and 2,000 trees

increased, the sensitivity for the system with 54 relevant variables drops faster than for the system with 204 ones, reaching 20 % when total number of variables arrives at 10,000, whereas the sensitivity for the system with 204 relevant features is still 40 % at this point.

This effect is most likely due to the method for generation of splits in random forest algorithm. The subset of variables is randomly selected from all variables and split is performed for the variable that produces the best split. When the number of relevant variables is large in comparison with the sample size, the variables with low

importance are rarely selected and hence their apparent importance is similar to that of random variables. When the number of relevant variables is small, but not very small, then there is a good chance that one or two relevant variables will be selected at each step. In this case the truly relevant variables have the highest chance to be selected and hence their apparent importance is high. Finally when the number of variables is very small in comparison with the number of total variables the chance of truly relevant variable being included in the sample is small and this again decreases the apparent importance of relevant variables in comparison with random ones, and hence decreases sensitivity.

The systematic survey of range of synthetic data sets generated with varying parameters shows that the results of Boruta algorithm are robust. While the sensitivity may be low for systems described with very large number of variables, nevertheless, the variables that are reported as relevant are relevant with very high probability.

2.3.2.1 Real-World Data Sets

Boruta algorithm has been also applied to four real-world data sets recently deposited in the UCI repository (see Table 2.6). In this case only the false discovery ratio could be estimated since the true relevance of the attributes is unknown. In two cases of the sets described with small number of attributes nearly all attributes were deemed relevant.

The level of false discovery was very low. In all cases the PPV_c^* was 100 %—not a single false discovery was made with the strict definition of relevance. With the more relaxed definition, accommodating also Boruta's tentative class as relevant, some false discoveries were reported for QSAR biodegradation data set. Nevertheless, even in this case the expected value of false discovery was 0.4 and PPV_t^* was 98.9 %. Therefore we may assume that nearly all features identified by Boruta as relevant are truly so. The case of QSAR biodegradation data set could suggest that variables assigned by Boruta to tentative class, bear higher risk of being false positive.

Table 2.6 Results for the real-world data sets from the UCI repository

Dataset	Data		Original			Contrast			PPV_c^*	PPV_t^*
	Instances	Variables	Conf	Tent	Rej	Conf	Tent	Rej	(%)	(%)
Q-b	1,055	41	36.2	0.8	4.0	0.0	0.4	40.6	100.0	98.9
TES	5,820	33	30.0	1.0	1.0	0.0	0.0	32.0	100.0	100.0
MM-500	931	1,300	293	66	941	0	0	1,300	100	100
MM-1000	931	1,300	363	58	879	0	0	1,300	100	100
ACRS	1,500	10,000	220	84	9,696	0.0	0.0	10,000.0	100.0	100.0

The MicroMass data set was analysed with Random Forest runs with 500 and 1,000 trees that are described as MM-500 and MM-1000, respectively. The number of variables marked as confirmed (Conf), tentative (Tent) and rejected (Rej) is reported for original and contrast variables. The PPV_c^* was computed according to Eq. 2.3 counting as relevant only these variables with *confirmed* status, for PPV_t^* also variables with *tentative* status were taken into account

Nevertheless, in the case of MicroMass data set all attributes deemed tentative by Boruta using 500 trees, were later deemed confirmed by Boruta using 1,000 trees, without any false positive hits. This suggests that when the number of variables deemed tentative is large, it is quite likely that most of them are truly relevant and Boruta run with larger number of trees is required.

2.4 Conclusions

As it was demonstrated in the chapter, the all-relevant feature selection algorithms are capable of discerning between relevant and non relevant variables. The Boruta algorithm, which was used as a representative algorithm of the class, was examined on a wide range of synthetic problems and several recently published real-world data sets. Algorithm works particularly well for systems for which good quality models may be obtained by means of random forest classification algorithm. The sensitivity of the algorithm is close to 100 % for such systems. The sensitivity of Boruta can be improved by utilising random forest with larger number of decision trees. The level of false discoveries is very low for all data sets examined, therefore *all relevant* feature selection is suitable for generation of robust knowledge.

The main factor limiting analysis with Boruta algorithm is time of computations. The single iteration of the random forest algorithm can take several hours for larger systems. The algorithm in the best case requires at least time equivalent to 30 random forest iterations to complete, hence entire analysis may take more then one CPU-week. The random forest is computationally demanding and its implementation in R, while very useful, is not very efficient for large problems. In particular, while the random forest is trivially parallel its implementation is strictly sequential. This limits application of the algorithm for analysis of truly large datasets described with tens or even hundreds thousands variables and thousands of objects.

Acknowledgments Computations were partially performed at the Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw, Poland, grant G34-5. Authors would like to thank Mr. Rafał Niemiec for technical help.

References

1. Bache, K., Lichman, M.: UCI machine learning repository. <http://archive.ics.uci.edu/ml> (2013)
2. Breiman, L.: Random forests. *Mach. Learn.* **45**, 5–32 (2001)
3. Draminski, M., Kierczak, M., Koronacki, J., Komorowski, J.: Monte Carlo feature selection and interdependency discovery in supervised classification. In: Koronacki, J. (ed.) *Advances in Machine Learning II*. SCI, vol. 263, pp. 371–385. Springer (2010)
4. Draminski, M., Rada-Iglesias, A., Enroth, S., Wadelius, C., Koronacki, J., Komorowski, J.: Monte Carlo feature selection for supervised classification. *Bioinformatics* **24**(1), 110–117 (2008)

5. Gunduz, N., Fokoue, E.: UCI machine learning repository. <http://archive.ics.uci.edu/ml/datasets/Turkiye+Student+Evaluation> (2013)
6. Huynh-Thu, V.A., Wehenkel, L., Geurts, P.: Exploiting tree-based variable importances to selectively identify relevant variables. In: JMLR: Workshop and Conference Proceedings, vol. 4, pp. 60–73 (2008)
7. Kohavi, R., John, G.: Wrappers for feature subset selection. *Artif. Intell.* **97**(1), 273–324 (1997)
8. Kursa, M.B., Rudnicki, W.R.: Feature selection with the Boruta package. *J. Stat. Softw.* **36**(11), 1–13 (2010)
9. Kursa, M.B., Jankowski, A., Rudnicki, W.R.: Boruta—a system for feature selection. *Fundam. Inform.* **101**(4), 271–285 (2010)
10. Leisch, F., Dimitriadou, E.: mlbench: machine learning benchmark problems. R package version 2.1–1 (2010)
11. Liaw, A., Wiener, M.: Classification and regression by random forest. *R News* **2**(3), 18–22 (2002). <http://CRAN.R-project.org/doc/Rnews/>
12. Mansouri, K., Ringsted, T., Ballabio, D., Todeschini, R., Consonni, V.: Quantitative structure-activity relationship models for ready biodegradability of chemicals. *J. Chem. Inf. Model.* **53**(4), 867–878 (2013)
13. Nilsson, R., Peña, J.M., Björkegren, J., Tegnér, J.: Detecting multivariate differentially expressed genes. *BMC Bioinform.* **8**, 150 (2007)
14. Rudnicki, W.R., Kierczak, M., Koronacki, J., Komorowski, J.: A statistical method for determining importance of variables in an information system. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H., Slowinski, R. (eds.) *Rough Sets and Current Trends in Computing*, vol. 4259/2006, pp. 557–566. Springer, Berlin/Heidelberg (2006)
15. Stoppiglia, H., Dreyfus, G., Dubois, R., Oussar, Y.: Ranking a random feature for variable and feature selection. *J. Mach. Learn. Res.* **3**(7–8), 1399–1414 (2003)
16. Team, R.C.: R: A language and environment for statistical computing. R foundation for statistical computing, Vienna, Austria (2012). <http://www.R-project.org/>
17. Tuv, E., Borisov, A., Torkkola, K.: Feature selection using ensemble based ranking against artificial contrasts. In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pp. 2181–2186. IEEE (2006)

Feature Selection for Data and Pattern Recognition

Stańczyk, U.; Jain, L.C. (Eds.)

2015, XVIII, 355 p. 74 illus., 20 illus. in color., Hardcover

ISBN: 978-3-662-45619-4