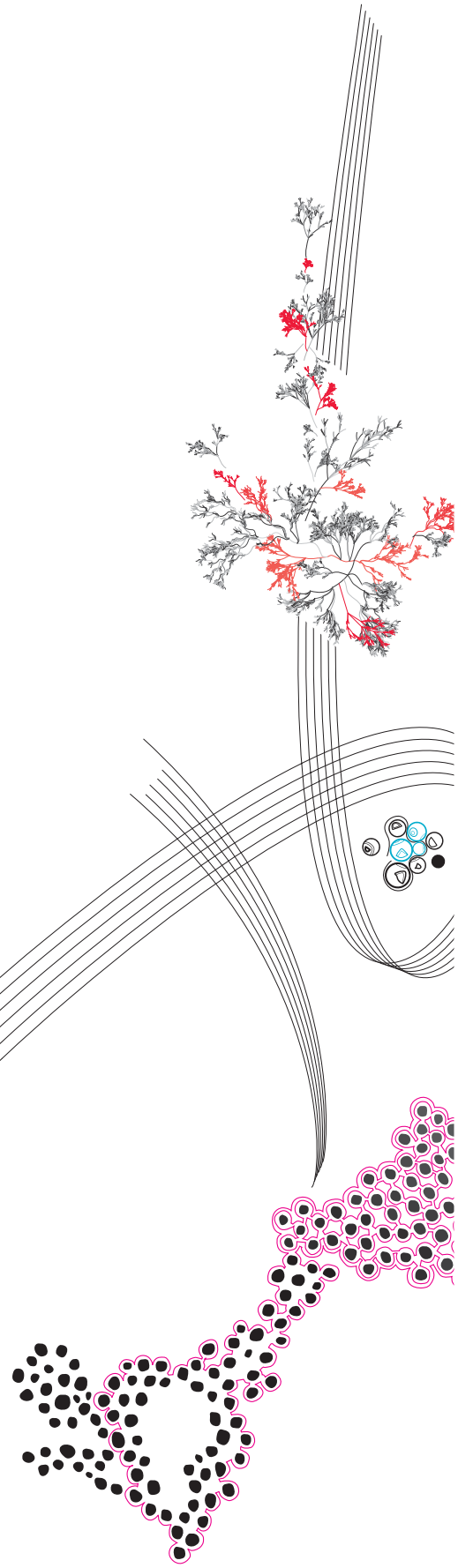

BSc Thesis Applied Mathematics

Analysis of random forest algorithms

Janiek Smulders

Supervisor: M.N.M. van Lieshout

July, 2021

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Analysis of random forest algorithms

Janiek L. Smulders*

July, 2021

Abstract

Random forests have many different implementations in R-packages. This study aims to analyse the performance of different random forests and to provide guidelines on which R-package to use. The R-packages studied in this paper are **extraTrees**, **party**, **randomForestSRC**, **ranger**, **RLT**, **RRF** and **KnowGRRF**. Only regression problems are considered in this study. The analysis is done by comparing the R-packages to **randomForest** regarding the mean squared error, the runtime and the variable importance. This is done by testing the R-packages on different types of data. Based on the computations in this research it can be concluded that **RLT** is advised to use for numerical data to obtain the lowest MSE. In all other cases **ranger** is suggested to use as it has a significantly lower runtime. Furthermore, the mean decrease in accuracy found in **randomForest** or the unbiased mean decrease found in **party** are recommended methods to use for obtaining the variable importance.

Keywords: random forests, regression, R.

*Email: j.l.smulders@student.utwente.nl

Contents

1 Introduction **3**

2 R-packages information **4**
2.1 Random forest algorithm 4
2.2 Performance measurements 4
2.3 R-packages 5
2.3.1 randomForest 5
2.3.2 extraTrees 5
2.3.3 party 6
2.3.4 randomForestSRC 6
2.3.5 ranger 6
2.3.6 RLT 7
2.3.7 RRF 8
2.3.8 KnowGRRF 9

3 Method **11**
3.1 Categorical data 11
3.2 Numerical data 12
3.3 Categorical and numerical data combined 14
3.4 Correlated data 14

4 Results **16**
4.1 Results on the categorical data sets 17
4.2 Results on the numerical data sets 18
4.3 Results on the data sets with categorical and numerical variables 20
4.4 Results on the data sets with correlated variables 20
4.5 Results on frequency distribution of the runtime 21
4.6 Results on variable importance 24

5 Conclusion **28**

A Alternative R-packages **33**

B Supplementary results on variable importance **35**

1 Introduction

Regression problems are presented in the form $Y = \beta X + e$ where Y is a vector consisting of n variables, X is an $n \times p$ matrix consisting of n observations of p predictor variables, β is a vector consisting of p unknown parameters and e represents the error terms. The error terms are all mutually independent and have an expected value of 0. Several methods are available to approximate the relation between the predictor variables and the response variable. The method of least squares tries to approximate the solution by minimizing the sum of the squares of the residuals. The residuals are the difference between the actual value, Y, and the predicted value of the model. However, this procedure may fail when for example the number of variables is very high. Then the random forest algorithm can be used to solve the model. Random forest is an ensemble method that combines a number of decision trees. Every decision tree gives a prediction and the random forest algorithm uses this to obtain a final prediction [1]. There are already multiple algorithms available in R. However, it is not always clear which R-package one should choose to obtain the best results when having a certain type of data set. The aim of this study is to investigate the performance of the R-packages **extraTrees** [16], **party** [11], **randomForestSRC** [10], **ranger** [18], **RLT** [21], **RRF** [4] and **KnowGRRF** [8] compared to **randomForest** [2], which will be used as a benchmark. The approach will be to analyse the performance of the R-packages on different types of data sets. To provide a framework for the research, the following questions can be asked in order determine for what conditions the R-packages are an improvement of **randomForest**:

1) Which R-packages perform well regarding the mean squared error?
2) Which R-packages are preferred concerning the runtime?
3) Which R-package contains the most accurate method to identify the variable importance?

2 R-packages information

This section will first explain the random forest algorithm in general. In addition, the measurements which are used in this paper to assess the performance of the R-packages will be discussed. Lastly, all the R-packages studied in this paper are examined.

2.1 Random forest algorithm

The data has n observations and p predictor variables. The algorithm will grow t trees on the data and the random forest will produce output $\hat{\mathrm{Y}}$ as the predicted value. The general steps taken in every algorithm are the following [23]:
Step 1: Draw t new data samples from the data
Step 2: Select for every data sample the variables for the trees to be grown on
Step 3: Grow a tree on every data sample
Step 4: Take the average of all the results from the trees
The steps are also shown in Figure 1.

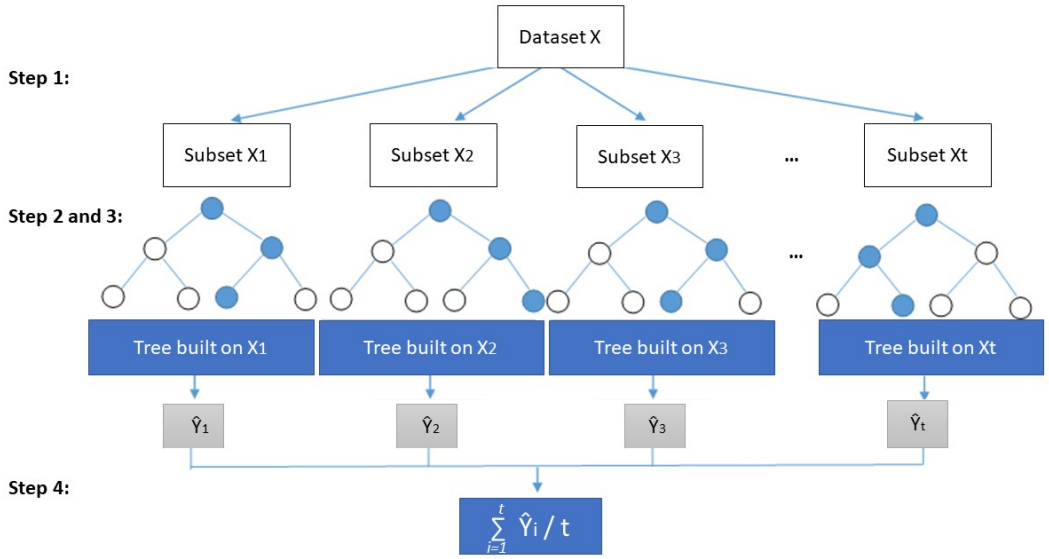

FIGURE 1: Random forest algorithm

2.2 Performance measurements

The performance of the algorithms will be based on 3 aspects, namely the mean squared error (MSE), the runtime and the variable importance (VI). Firstly, the MSE is the average squared difference between the predicted value and the actual value:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\mathrm{Y}_i - \hat{\mathrm{Y}}_i)^2 \tag{1}$$

where Y_i is the actual value of the response which can be found in the data sets and $\hat{Y}_i$ is the predicted value given by the random forest for observation $i \in \{1, 2, \ldots, n\}$. Secondly, the runtime of the random forest is the time the algorithm requires to grow the forest. Lastly, the VI evaluates the importance of a variable for the model.

2.3 R-packages

In the following subsections all the R-packages analysed in this paper will be examined. In section 2.3.1 the benchmark algorithm will be explained focusing on the construction of the algorithm and the variable importance. In the subsequent subsections various R-packages are discussed in which the differences with the benchmark will be highlighted regarding the algorithm and the variable importance. Refer to Appendix A to see the alternative R-packages which have been considered.

2.3.1 randomForest

The R-package **randomForest** will be used in this study as the benchmark. In **randomForest** new samples are created with bagging. Bagging means that each new sample is drawn with replacement from the original data set and therefore many trees have different samples to be trained on. All the new samples created by bagging will have n observations which is the same size as the original data set X. In addition, the observations which are not in the newly generated sample form the test set of the new sample. It is said that the test set consists of the out-of-bag (OOB) observations. Then a tree is grown on the new sample using random feature selection. With random feature selection is meant that at each node of a tree, a small group of variables to split on is selected at random. For this the parameter $mtry$ is used to denote the number of variables selected at random and for regression the default setting is equal to $p/3$ variables. Now, from this group of variables, the best binary split is chosen. In this case, the best split is the one which gives the largest reduction in the MSE as defined in Equation 1 [3]. So it first determines the best cutting threshold for the variables and then chooses the best variable to split on. The algorithm will stop splitting nodes when the minimum number of observations in a terminal node is reached, for regression the default setting is 5. Lastly, every tree gives an estimate of the response variable and then an overall prediction is made by the average of all the estimates [1, 13].

The variable importance
The R-package **randomForest** has 2 methods to measure the VI. The first one is the mean decrease in accuracy. This is measured by first computing the MSE on the OOB data for every tree and taking the average over all the trees. Then the same is done again but with the mth variable permuted. Then the VI for the mth variable is the percentage of increase between the first one and the second one. The higher the increase, the more important the variable is considered. The second one is the mean decrease in node impurity. This is measured by computing for a variable the difference between residual sum of squares (RSS) before and after a split. RSS is defined as $\sum_{i=1}^{n}(\mathrm{Y}_i - \hat{\mathrm{Y}}_i)^2$. This is summed up over all splits over all trees. The higher this number will be, the more important the variable is considered.

2.3.2 extraTrees

The R-package **extraTrees** stands for extremely randomized trees. Compared to **randomForest** it has 2 significant differences. Firstly, **extraTrees** chooses the cut at each node randomly. Like in **randomForest**, it first chooses a random subset of $mtry$ variables. However, at each node **extraTrees** chooses the cut uniformly randomly, while **randomForest** chooses the best cutting threshold for the variable. After the cutting threshold has been fixed, the feature with the biggest gain is chosen, which is similar to **randomForest**

[17]. Secondly, **extraTrees** samples without replacement and uses therefore the complete original dataset. So it does not perform bootstrapping like in **randomForest**. The aim of **extraTrees** is to achieve a faster computation time compared to **randomForest** while having a similar MSE [7].

The variable importance
The R-package **extraTrees** does not have a method to measure the VI.

2.3.3 party

The R-package **party** includes a function called *cforest* which means conditional inference forest. This is an unbiased forest consisting of conditional inference trees which are called ctrees. The main difference with **randomForest** is that a significance test is used to select splitting variables rather than selecting the variable that maximizes the decrease in MSE. So ctrees apply a significance test to determine whether there exists a statistically significant association between the predictor variables and the response variables. If this is the case, the predictor with the highest association with the response variable is chosen to split on [20, 12]. It is expected that *cforest* is not biased towards variables with many cut points and also has a better performance on data with correlated variables than **randomForest**.

The variable importance
The R-package **party** has 2 different procedures to measure the VI. The first one is the mean decrease in accuracy as in **randomForest**. The second one is the unbiased mean decrease in accuracy. This method adjusts for correlations between predictor variables. This is done by permuting within a grid determined by the covariates that are associated to the variable of interest [11].

2.3.4 randomForestSRC

The R-package **randomForestSRC** (**RFSRC**) stands for a fast unified Random Forests for Survival, Regression and Classification. This R-package provides many options for the user on how to build the random forest. The main difference with **randomForest** is that **RFSRC** generates the bootstraps without replacement. The bootstraps will have a size of 0.632 times the original data size. So it is expected that **randomForestSRC** will be faster than **randomForest**.

The variable importance
The VI computed in **randomForestSRC** has 3 different methods to measure the VI. The first one is mean decrease in accuracy as in **randomForest**. However, as **randomForestSRC** does not perform bootstrapping, it will not have OOB data to obtain the VI as in **randomForest**. So in this case it will use the out of sample data to obtain the VI. The second one works similarly to the mean decrease in accuracy, except now the variable will not be permuted. Every time a split is made on this variable, it will randomly choose to go to the right or left daughter node. The third one also works similarly to mean decrease in accuracy except now the variable will not be permuted, but every time a split is made on this variable, it will go to the opposite daughter node.

2.3.5 ranger

The name **ranger** comes from the words 'RANdom forest GeneRator' and it is a fast implementation of **randomForest**. A significant difference between **ranger** and **ran-**

domForest is the technique used for selecting the variable set and a splitting variable at the nodes. Normally, all values of the *mtry* variables need to be evaluated as splitting candidates. However, for high dimensional data which has many variables, this is not very efficient. In **ranger** 2 different splitting algorithms are used. The first algorithm sorts the values of the features beforehand and retrieves them by their index. In the second one, the raw values are retrieved and sorted while splitting. The first one should be used for nodes containing many observations and the second one for nodes containing less observations. Furthermore, in **ranger**, drawing the *mtry* variables as potential splitting variables in each node is done by sampling without replacement. In this way, copies of the original data is prevented which is more memory efficient. In addition, node information is saved in simple data structures and whenever possible it tries to free memory early [19]. This is all done with the aim to obtain a faster random forest. Hence, it is expected that **ranger** will be faster than **randomForest**.

The variable importance
In **ranger** there are 3 options to measure the VI. The first two methods to measure the VI are the same as in **randomForest**. The third one is unbiased mean decrease in node impurity. It is unbiased in terms of the number of categories and category frequencies. This method achieves this by identifying 2 parts in the node impurity. The first part is the reduction in node impurity directly related to the predictor variable and the second part is the reduction in node impurity solely related to the structure of the predictor variable. The unbiased node impurity method only measures the first part for the variable importance [15].

2.3.6 RLT

RLT means Reinforcement Learning Trees which uses a different method for selecting the splitting variable and it also has an option of making high dimensional splits. It chooses the splitting variable that gives the greatest gain in the future rather than the greatest gain from the immediate split, which makes it more efficient to make high dimensional cuts [22].

At the first node of a tree, the split will be made on the variable with the highest variable importance. After the first split has been made, the algorithm finds the qth variable with the lowest importance and puts this one and all variables with a lower variable importance in the muted set. Also, it finds the rth variable with the highest importance and puts this one and all variables with higher variable importance than this one in the protected set. To determine a splitting variable at the subsequent nodes in a tree, only the variable importance of variables which are not in the muted set are considered. Now there are 2 options to make a split:

- one dimensional split: the variable with the highest variable importance is chosen. The threshold for the cut point is chosen uniformly randomly.
- k dimensional split: the split is made on a linear combination of k variables, namely βX where β is a vector of length k and X consists of k variables. The vector β is chosen according to [22]. Only variables that have a minimal percentage α of the maximum variable importance in the current node are considered. Then similarly to a one dimensional split, the cutting threshold is chosen uniformly randomly.

Now the set of muted variables is updated for the daughter nodes. This is done by adding the variables with the lowest variable importance to the set of muted variables. Now after

every split, the protected set is updated at every node by adding the splitting variable from that node. The muted set is updated by finding the qth variable with the lowest importance among the variables that are not in the muted set and also not in the protected set. Then this variable and all the variables with a lower variable importance than that one, are put in the muted set.

The aim is to choose r such that all informative variables will be in the protected set. The number q can be tuned by choosing a certain percentage of the number of nonmuted variables at each internal node. It is expected that **RLT** works well for data sets with many uninformative variables and works less well for data sets in which all variables are important.

The variable importance
The VI is computed for every variable in every node of a tree for the variables which are not in the muted set. The VI is obtained with the use of **randomForest** with the mean decrease of accuracy. The VI for the variables in the muted set equals 0.

2.3.7 RRF

RRF denotes Regularized Random Forest which is based on the **randomForest** R-package. The main difference between those is that features are selected with a regularization framework in the random forest. This is done by first assigning a gain to each feature. Then the features that have not already been selected before will be penalized. Due to this, less features are selected and a compact high quality subset of features is created.

The algorithm keeps track of a feature set F which is initially empty at the first node of the first tree. Let $\text{gain}(X_{j,v})$ be the measure of selecting feature X_j at node v. The highest gain of a feature will be selected to split the node on. Now in a regularized forest the gain of choosing feature X_j will be penalized if it is not in feature set F:

$$\text{gain}_{RRF} = \begin{cases} \text{gain}(X_j, v), & \text{if } (X_j, v) \in F \\ \lambda \text{gain}(X_j, v) & \text{if } (X_j, v) \notin F \end{cases}$$

where $\lambda \in (0, 1]$ is called the penalty factor. Once a feature that is not in F is chosen, it will be added to F and the set will be continued to use in the next leaves of the tree and also in the next trees of the forest [6]. This procedure is also displayed in Figure 2.

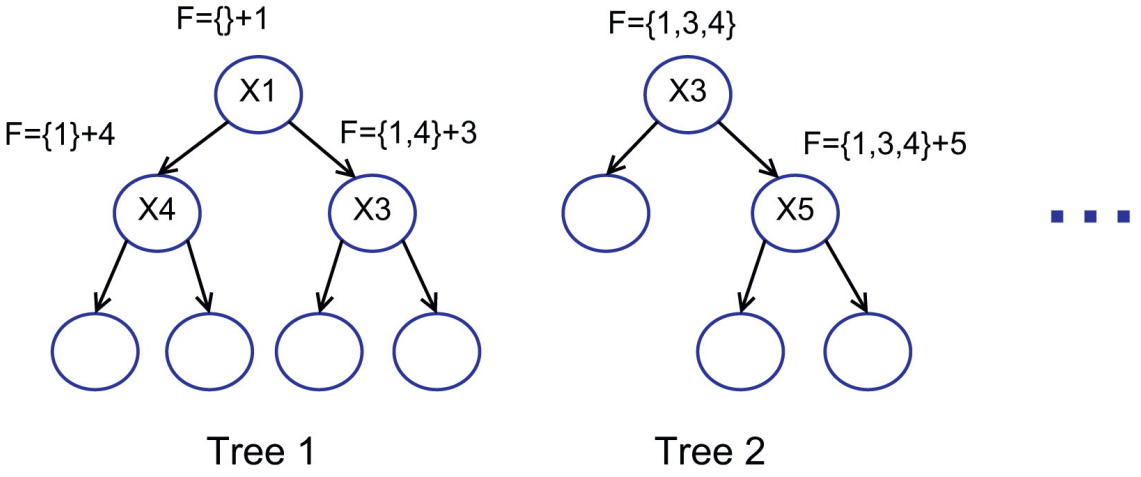

FIGURE 2: The feature selection procedure in RRF [5]

As **RRF** will identify the most important features and is less likely to select noise features to split a node on, it is expected that it will work well for data sets with a combination of informative and uninformative features.

There is also an option in the R-package to make a guided regularized random forest (GRRF). This also makes use of the importance scores obtained by **randomForest** of features to guide the RRF. Here the penalty factor is not the same for every feature:

$$\text{gain}_{GRRF} = \begin{cases} \text{gain}(X_j, v), & \text{if } (X_j, v) \in F \\ \lambda_j \text{gain}(X_j, v) & \text{if } (X_j, v) \notin F \end{cases}$$

where $\lambda_j = (1-\gamma) + \gamma \frac{\text{Importance}_j}{\max_{j=1}^{P} \text{Importance}_j}$ where $\gamma \in [0,1]$ is called the importance coefficient and Importance_j denotes the variable importance of predictor variable j [5].

The variable importance
The VI computed by the R-package **RRF** has exactly the same 2 methods as in **randomForest**.

2.3.8 KnowGRRF

KnowGRRF denotes Knowledge-Based Guided Regularized Random Forest. It differs from **randomForest** by feature selection. First, in this case the gain is defined by:

$$\text{gain}_{Know-GRRF} = \begin{cases} \text{gain}(X_j, v), & \text{if } (X_j, v) \in F \\ \lambda_i \text{gain}(X_j, v) & \text{if } (X_j, v) \notin F \end{cases}$$

where $\lambda_i = \text{score}_j^{\delta}$ and $\text{score}_j \in [0,1]$ and $\delta \in [0, \infty)$ is the tuning parameter. The score_j indicates the importance of predictor j. The importance can be obtained by combining a set of priors from a number of domains for every predictor [9]. However, in this study the importance of features will be used just like in **GRRF** and hence $\text{score}_j = \frac{\text{Importance}_j}{\max_{j=1}^{P} \text{Importance}_j}$ is obtained. The number of features selected is quite sensitive to the tuning parameter δ and when a higher value for δ is set, it is likely to choose less features. Thus, compared to **RRF**, is expected to work better on data sets with just a few informative variables and a

lot of uninformative variables.

The variable importance
As **KnowGRRF** defines what variables to use and then employs **randomForest** to grow the forest, the VI is also retrieved from **randomForest**.

3 Method

In this section the different data sets on which the R-packages are tested are described. Firstly, categorical data and numerical data will be discussed. Then data with categorical and numerical covariates is examined. Lastly, data with correlated features is described.

3.1 Categorical data

To generate data sets with categorical variables, the categories of the variables $X_{i,j}$ are sampled with replacement using the function *sample*. In this function, the number of classes and the probability of the occurrence of a class can be set. The subscripts for $X_{i,j}$ describes the i^{th} observation of the j^{th} variable. The response variable Y_i is determined by $Y_i = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \cdots + \beta_p X_{i,p} + e_i$ where e_i is the error term and β_k with $k \in \{0, 1, \ldots, p\}$. The error term is drawn from a uniform distribution, namely $\mathcal{U}(0, 5)$.

A small data set of 1 variable is generated to give a visualisation of a tree using categorical data set. The predictor variable $X_{i,1}$, is generated with 3 classes 'A', 'B' and 'C' for 100 observations. The variable is drawn from a discrete distribution where all classes are equally likely to be drawn. The response variable Y_i is determined by $Y_i = \beta_0 + \beta_1 X_{i,1} + e_i$ where $\beta_0 = 1$ and $\beta_1 = \begin{cases} 2, & \text{if } X_{i,1} \text{ is class A} \\ 9, & \text{if } X_{i,1} \text{ is class B} \\ 20, & \text{if } X_{i,1} \text{ is class C} \end{cases}$.

Figure 3 shows a ctree from the R-package **party** of the data just described. The white circles in the figure display which predictor was chosen to make the split at each node and the p-values of the significance test. In the edges, the classes chosen for the split are shown. Also, in the gray leaves of the tree the number n represents the total number of observations in that specific leaf and the number y denotes the estimate in that specific leaf.

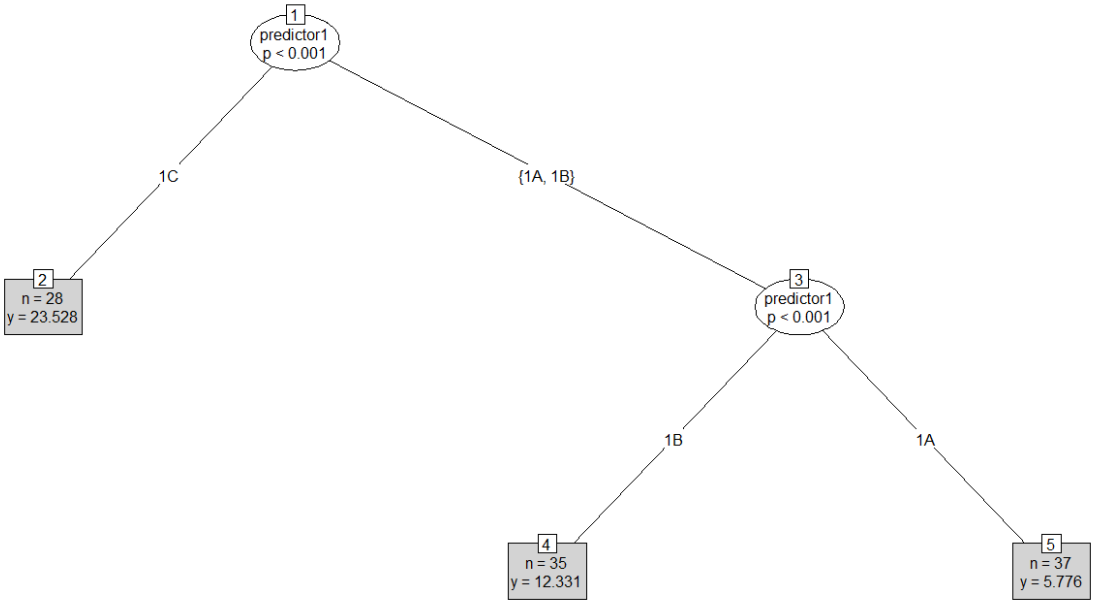

FIGURE 3: A ctree of 1 predictor variable with 3 classes

Several data sets with different characteristics are created to explore the effect on the performance of the R-packages. This is done by varying the number of variables, the number of classes, the number of informative variables and the occurrence of classes. An overview of the data sets is given in Table 1. In data set AE, the first 2 variables are taken to be informative and the other variables are redundant. In data set AF, the occurrence of a certain class is higher than the other two classes for all variables.

TABLE 1: Information about categorical data sets

Name of data set	Number of variables	Number of classes per variables	Number of informative variables	Probability of occurrence of a class
AA	16	all 3 classes	all informative	all equally likely
AB	8	all 3 classes	all informative	all equally likely
AC	8	all 20 classes	all informative	all equally likely
AD	8	4 with 3 classes and 4 with 20 classes	all informative	all equally likely
AE	8	all 3 classes	2 informative	all equally likely
AF	8	all 3 classes	all informative	5/6, 1/12, 1/12

3.2 Numerical data

For the numerical data several data sets are generated, each representing a different aspect. The variables in the numerical data are drawn from a normal distribution $X_{i,j} \sim \mathcal{N}(\mu, \sigma^2)$ with mean μ, drawn from uniform distribution $\mu \sim \mathcal{U}(5, 20)$ and variance σ^2 drawn from uniform distribution $\sigma^2 \sim \mathcal{U}(0, 5)$. The noise e_i in the response variable is introduced by drawing e_i from a uniform distribution $e_i \sim \mathcal{U}(0, 1)$. In Table 2 the characteristics of the generated data sets is displayed. In the last column, the different aspects that are explored in the data sets is specified.

TABLE 2: Information about numerical data sets

Name of data set	Number of variables	Response variable	Aspect of data set
BA	5	$Y_i = 5 + 2X_{i,1} + 4X_{i,2} + X_{i,3}$ $+3X_{i,4} + 2.5X_{i,5} + e_i$	Linear and only informative variables
BB	5	$Y_i = 5 + 2X_{i,1} + 400X_{i,2} + X_{i,3}$ $+3X_{i,4} + 2.5X_{i,5} + e_i$	Linear, only informative variables and one variable with higher importance
BC	5	$Y_i = X_{i,1}^2 + e_i$	Higher-order and only first variable informative
BD	5	$Y_i = X_{i,1}X_{i,2} + e_i$	Interaction and only first and second variable informative
BE	5	$Y_i = X_{i,1}X_{i,2}X_{i,3}X_{i,4}X_{i,5} + e_i$	Interaction and only informative variables
BF	25	$Y_i = 5 + 2X_{i,1} + 4X_{i,2} + X_{i,3}$ $+3X_{i,4} + 2.5X_{i,5} + 2X_{i,6} + 4X_{i,7}$ $+X_{i,8} + 3X_{i,9} + 2X_{i,10} + 4X_{i,11}$ $+X_{i,12} + 3X_{i,13} + 2X_{i,14} + 4X_{i,15}$ $+X_{i,16} + 3X_{i,17} + 2X_{i,18} + 4X_{i,19}$ $+X_{i,20} + 3X_{i,21} + 2X_{i,22} + 4X_{i,23}$ $+X_{i,24} + 3X_{i,25} + e_i$	Linear and only informative variables
BG	25	$Y_i = 5 + 2X_{i,1} + 400X_{i,2} + X_{i,3}$ $+3X_{i,4} + 2.5X_{i,5} + 2X_{i,6} + 4X_{i,7}$ $+X_{i,8} + 3X_{i,9} + 2X_{i,10} + 4X_{i,11}$ $+X_{i,12} + 3X_{i,13} + 2X_{i,14} + 4X_{i,15}$ $+X_{i,16} + 3X_{i,17} + 2X_{i,18} + 4X_{i,19}$ $+X_{i,20} + 3X_{i,21} + 2X_{i,22} + 4X_{i,23}$ $+X_{i,24} + 3X_{i,25} + e_i$	Linear, only informative variables and one variable with higher importance
BH	25	$Y_i = 5 + 2X_{i,1} + 4X_{i,2} + e_i$	Linear and only first and second variable informative
BI	50	$Y_i = 5 + 2X_{i,1} + 4X_{i,2} + e_i$	Linear and only first and second variable informative

To give a visualisation of what happens in a random forest, a tree from *cforest* was extracted which was run on data set BA with 100 observations. Figure 4 shows on which predictor the split was made and the threshold for the split.

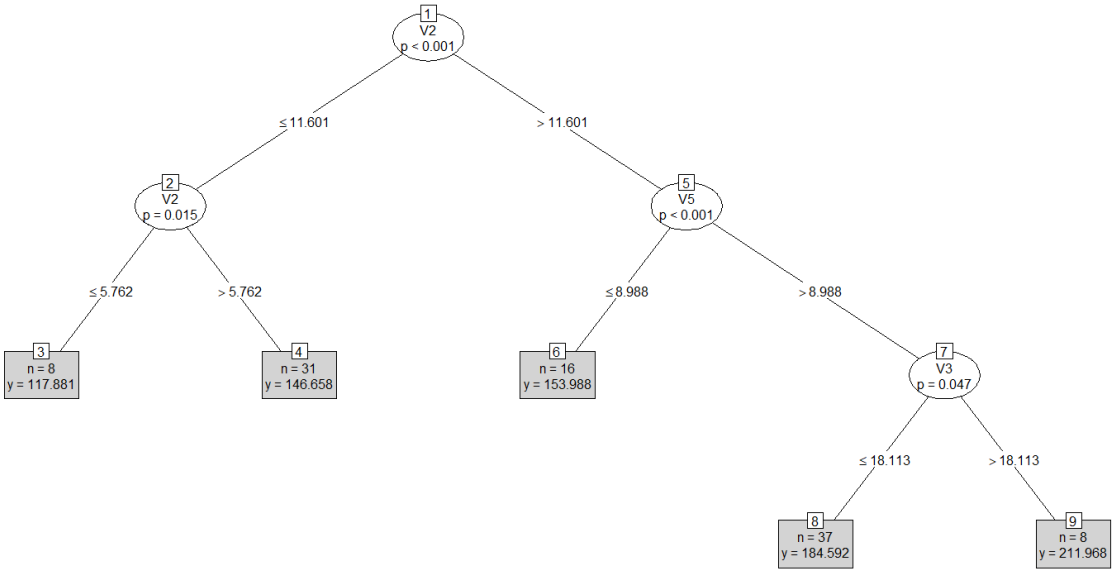

FIGURE 4: A ctree on data set BA

3.3 Categorical and numerical data combined

The numerical variables are drawn from a normal distribution $X_{i,j} \sim \mathcal{N}(\mu, \sigma^2)$ with mean μ drawn from uniform distribution $\mu \sim \mathcal{U}(5, 20)$ and variance σ^2 drawn from uniform distribution $\sigma^2 \sim \mathcal{U}(0, 5)$. The categorical variables are drawn from a discrete distribution where the occurrence of a class is equally likely. All categorical variables are generated with 3 classes. The response variable Y is determined by $Y = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \cdots + \beta_{16} X_{i,16} + e_i$ where e_i is drawn from a uniform distribution $\mathcal{U}(0, 5)$ and β_k with $k \in \{0, 1, \ldots, 16\}$ are defined for certain values. An overview of the data sets is given in Table 3:

TABLE 3: Information about numerical and categorical data sets

Name of data set	Number of numerical variables	Number of categorical variables
CA	8	8
CB	14	2
CC	2	14

3.4 Correlated data

To create correlated variables in the data sets the relation between various variables is specified. This is done by using a correlation matrix which displays the correlation between variables. A data set is generated by sampling $n \times p$ values from $\mathcal{N}(0, 1)$. All the data sets consist of $p = 25$ variables and of $n = 1200$ observations. Then using the function *genCorData* from the R-package **simstudy** and specifying the correlation matrix, 200 observations are generated such that these are correlated to train the random forest. Then the other 1000 observations which are not correlated are used to test the random forest. The values in the correlation matrices generated are defined as:

- Correlation matrix 1: $\sigma_{i,j} = 0.6^{|i-j|}$.

- Correlation matrix 2: $\begin{cases} \sigma_{i,j} = 0.4 & \text{if } i \neq j \\ \sigma_{i,j} = 1 & \text{if } i = j \end{cases}$.
- Correlation matrix 3: $\begin{cases} \sigma_{i,j} = 0.7 & \text{if } i \neq j \\ \sigma_{i,j} = 1 & \text{if } i = j \end{cases}$.

The first correlation matrix could be interpreted as for example correlation between alleles in biomedical data sets where alleles closer to each other are more correlated than ones further. The second and third correlation matrix could be interpreted as a correlation between variables in econometric data where all variables are equally correlated. The following table displays the data sets generated with certain characteristics:

TABLE 4: Information about data sets with correlated variables

Name of data set	Correlation matrix used	Response variable
DA	1	$\mathrm{Y}_i = 5X_{i,3} + 3X_{i,15} + e_i$
DB	1	$\mathrm{Y}_i = X_{i,1} + X_{i,2} + ... + X_{i,24} + X_{i,25} + e_i$
DC	1	$\mathrm{Y}_i = X_{i,2}X_{i,3} + X_{i,6}X_{i,7} + X_{i,12}X_{i,24} + e_i$
DD	2	$\mathrm{Y}_i = 5X_{i,3} + 3X_{i,15} + e_i$
DE	2	$\mathrm{Y}_i = X_{i,1} + X_{i,2} + ... + X_{i,24} + X_{i,25} + e_i$
DF	2	$\mathrm{Y}_i = X_{i,2}X_{i,3} + X_{i,6}X_{i,7} + X_{i,12}X_{i,24} + e_i$
DG	3	$\mathrm{Y}_i = 5X_{i,3} + 3X_{i,15} + e_i$
DH	3	$\mathrm{Y}_i = X_{i,1} + X_{i,2} + ... + X_{i,24} + X_{i,25} + e_i$
DI	3	$\mathrm{Y}_i = X_{i,2}X_{i,3} + X_{i,6}X_{i,7} + X_{i,12}X_{i,24} + e_i$

4 Results

This section will give the results of the 7 different R-packages with different kind of data sets which have been discussed. The R-package **randomForest** will be used as benchmark, so that the other R-packages can be compared to it. To assess the performance of the analyzed R-packages on the different data sets, the R-packages will be compared to the benchmark on 3 aspects. Firstly, the MSE is measured for the random forests and compared with the MSE of **randomForest**. The MSE is chosen as evaluation measure over the OOB error because not all the random forests can be evaluated with the OOB error. This is due to the fact that some random forests do not have OOB data. The MSE will be measured by first training the random forests on 200 observations and then it will be tested for 1000 observations. Secondly, the runtime of the R-packages is regarded. The runtime of each R-package will be kept track off and will be compared to the runtime of **randomForest** on the same data sets. This is done with the use of the R-package **microbenchmark** which can measure the execution time of R expressions [14]. The mean of the MSE and the mean of the runtime will be taken of 100 runs. The results of this is given in the first 4 sections displayed in tables to give a clear overview of the results per type of data set. In addition, a few graphs which display the diversity of the runtimes will be presented which will be obtained using the R-package **microbenchmark**. This results is shown in Section 4.5. The last aspect to assess the performance is variable importance. For this, several tables which show the variable importance of different R-packages on data sets is shown in Section 4.6.

The settings of the R-packages are as following:

- *mtry*, the number of variables which can be chosen from at a node is set to $p/3$ (rounded down). An exception is **RLT** as the framework for reinforcement learning trees of this R-package will consider all the variables and does not have the parameter *mtry*.
- *ntree*, the number of trees grown is set to 500, except for **RLT** where it is set to 100 considering the runtime.
- *nodesize*, the minimum number of observations in a terminal node is set to 5 for all R-packages. However, in **RLT** there is no *nodesize* parameter, but *nmin* is used instead. The parameter *nmin* denotes the minimum number of observations needed in an internal node to perform a split. It is recommended to set this to twice the desired size of the terminal node, which in this case is 10 [21]. In addition, **party** does not have the parameter *nodesize*. In *cforest*, a split is not made on a node if the significance test is rejected.

Also, the following extra parameter settings for specific R-packages are used:

- For the penalty factor λ in the R-package **RRF**, 2 values have been chosen to contrast and compare the effects, namely $\lambda = 0.5$ and $\lambda = 0.8$. Now $\mathbf{RRF_{0.5}}$ and $\mathbf{RRF_{0.8}}$ denote a regularized random forest with a penalty factor of 0.5 and 0.8, respectively. Furthermore, a guided regularized random forest (GRRF) can also be grown in the R-package. For the importance coefficient γ, the values 0.4 and 0.5 are chosen for $\mathbf{GRRF_{0.4}}$ and $\mathbf{GRRF_{0.6}}$, respectively.
- For the R-package **RLT** a percentage of the number of nonmuted variables at each node for the number of newly muted variables, called the muting percentage, is set to 0.2 and 0.5. There is also an option to make a high dimensional split. A one dimensional split and a split of a linear combination of 2 variables are chosen.

The parameters α and r which were discussed in Section 2.3.6 is set to the default setting. So 4 different settings are tried, namely $\mathbf{RLT_{1,0.2}}$, $\mathbf{RLT_{1,0.5}}$, $\mathbf{RLT_{2,0.2}}$ and $\mathbf{RLT_{2,0.5}}$, where the first subscript denotes the number of variables the split is made on and the second subscript denotes the muting percentage.

- The tuning parameter δ in the R-package **KnowGRRF** will be obtained by minimizing the akaike information criterion (AIC) for every dataset. This is done by using the BFGS quasi-Newton method [9]. The AIC asseses the quality of a model compared to other models. Then **randomForest** is used to grow the forest using the features selected by **KnowGRRF**. If **KnowGRRF** chooses all the variables, the performance will be equal to **randomForest**.

In the tables the dash is used if the random forest does not work for this type of data. For example, **extraTrees** and **RLT** cannot handle categorical data as these R-packages choose the threshold for splitting on a variable uniformly randomly, which is not possible for categorical data.

4.1 Results on the categorical data sets

TABLE 5: MSE and runtime in seconds for the data sets AA, AB and AC per R-package

	AA MSE	AA runtime	AB MSE	AB runtime	AC MSE	AC runtime
randomForest	200.897	0.3076	74.170	0.2412	448.248	0.2028
extraTrees	-	-	-	-	-	-
KnowGRRF	196.856	0.1603	61.095	0.2231	453.020	0.1990
party	266.730	0.5497	123.396	0.5803	386.823	1.6166
RFSRC	219.658	0.1168	90.702	0.1162	474.878	0.1623
ranger	188.142	0.0382	72.039	0.0393	473.842	0.0335
$\mathbf{RLT_{1,0.2}}$	-	-	-	-	-	-
$\mathbf{RLT_{1,0.5}}$	-	-	-	-	-	-
$\mathbf{RLT_{2,0.2}}$	-	-	-	-	-	-
$\mathbf{RLT_{2,0.5}}$	-	-	-	-	-	-
$\mathbf{RRF_{0.5}}$	196.856	0.2946	71.023	0.2365	569.766	0.1792
$\mathbf{RRF_{0.8}}$	197.174	0.2903	71.075	0.2359	570.263	0.1869
$\mathbf{GRRF_{0.4}}$	197.189	0.2980	71.013	0.2376	570.667	0.1959
$\mathbf{GRFF_{0.6}}$	197.311	0.2979	71.074	0.2233	570.468	0.1919

TABLE 6: MSE and runtime in seconds for the data sets AD, AE and AF per R-package

	AD MSE	AD runtime	AE MSE	AE runtime	AF MSE	AF runtime
randomForest	252.419	0.3223	13.051	0.2279	21.870	0.0734
extraTrees	-	-	-	-	-	-
KnowGRRF	252.958	0.3282	6.458	0.0669	20.414	0.0633
party	223.148	2.8946	23.776	0.5345	38.461	0.2081
RFSRC	243.414	0.1735	16.134	0.1057	19.895	0.0574
ranger	234.928	0.0460	13.690	0.0381	21.369	0.0153
$\mathbf{RLT_{1,0.2}}$	-	-	-	-	-	-
$\mathbf{RLT_{1,0.5}}$	-	-	-	-	-	-
$\mathbf{RLT_{2,0.2}}$	-	-	-	-	-	-
$\mathbf{RLT_{2,0.5}}$	-	-	-	-	-	-
$\mathbf{RRF_{0.5}}$	245.086	0.3114	12.607	0.2187	21.437	0.0696
$\mathbf{RRF_{0.8}}$	245.022	0.3052	12.545	0.2249	21.429	0.0720
$\mathbf{GRRF_{0.4}}$	244.498	0.3133	12.504	0.2232	21.465	0.0706
$\mathbf{GRFF_{0.6}}$	244.940	0.3065	12.486	0.2156	21.486	0.0716

From Tables 5 and 6 can be seen that regarding the MSE, there is no clear outstanding R-package. Only **party** performs slightly better on data set AC and AD, but performs worse on the other data sets. Focusing on the runtime, it is noticeable that **ranger** is always the fastest.

4.2 Results on the numerical data sets

TABLE 7: MSE and runtime in seconds for the data sets BA, BB and BC per R-package

	BA MSE	BA runtime	BB MSE	BB runtime	BC MSE	BC runtime
randomForest	252.655	0.1065	922 178	0.0910	3350.59	0.0946
extraTrees	251.067	0.0312	989 325	0.0353	4070.12	0.0285
KnowGRRF	252.655	0.1065	922 178	0.0910	3350.59	0.0946
party	400.872	0.1654	1 718 636	0.1440	6042.87	0.1653
RFSRC	300.386	0.0633	1 179 557	0.0590	3798.30	0.0580
ranger	252.622	0.0292	930 594	0.0210	3348.09	0.0242
$\mathbf{RLT_{1,0.2}}$	243.539	10.8776	92 052	5.8588	214.53	14.0109
$\mathbf{RLT_{1,0.5}}$	352.697	7.6101	76 451	3.1843	195.16	6.1655
$\mathbf{RLT_{2,0.2}}$	108.998	11.2874	93 877	6.0063	208.19	14.1161
$\mathbf{RLT_{2,0.5}}$	266.011	8.5546	76 841	3.2595	179.84	5.7584
$\mathbf{RRF_{0.5}}$	252.402	0.0967	921 878	0.0842	3338.04	0.0932
$\mathbf{RRF_{0.8}}$	252.444	0.1103	922 427	0.0786	3358.77	0.0940
$\mathbf{GRRF_{0.4}}$	251.896	0.1047	922 803	0.0812	3362.41	0.0947
$\mathbf{GRFF_{0.6}}$	252.009	0.1043	926 006	0.0873	3345.02	0.0903

TABLE 8: MSE and runtime in seconds for the data sets BD, BE and BF per R-package

	BD MSE	BD runtime	BE MSE	BE runtime	BF MSE	BF runtime
randomForest	2494.634	0.0965	$4.091 \cdot 10^{10}$	0.0887	3466.773	0.5955
extraTrees	2629.600	0.0313	$4.067 \cdot 10^{10}$	0.0267	3513.309	0.1739
KnowGRRF	525.480	0.0860	$4.091 \cdot 10^{10}$	0.0887	3430.239	0.4239
party	4027.188	0.1503	$5.473 \cdot 10^{10}$	0.1481	3857.866	0.5194
RFSRC	2841.943	0.0525	$4.234 \cdot 10^{10}$	0.0493	3556.546	0.1794
ranger	2485.349	0.0228	$4.094 \cdot 10^{10}$	0.0227	3471.565	0.0939
$\mathbf{RLT_{1,0.2}}$	505.085	6.1141	$4.752 \cdot 10^{10}$	6.1647	3739.234	34.7281
$\mathbf{RLT_{1,0.5}}$	479.624	4.9774	$6.114 \cdot 10^{10}$	4.9430	3883.996	31.4323
$\mathbf{RLT_{2,0.2}}$	346.853	6.3825	$3.393 \cdot 10^{10}$	6.2752	3210.999	36.6481
$\mathbf{RLT_{2,0.5}}$	302.075	5.3093	$5.235 \cdot 10^{10}$	5.4803	3523.321	35.1311
$\mathbf{RRF_{0.5}}$	2492.779	0.0943	$4.089 \cdot 10^{10}$	0.0858	3471.317	0.5672
$\mathbf{RRF_{0.8}}$	2483.017	0.0893	$4.085 \cdot 10^{10}$	0.0868	3471.020	0.5753
$\mathbf{GRRF_{0.4}}$	2497.537	0.0878	$4.084 \cdot 10^{10}$	0.0857	3471.291	0.5560
$\mathbf{GRFF_{0.6}}$	2498.397	0.0941	$4.076 \cdot 10^{10}$	0.0869	3467.286	0.5845

TABLE 9: MSE and runtime in seconds for the data sets BG, BH and BI per R-package

	BG MSE	BG runtime	BH MSE	BH runtime	BI MSE	BI runtime
randomForest	474 216	0.3415	89.657	0.4315	101.719	0.9255
extraTrees	368 954	0.1127	80.612	0.1397	96.717	0.3119
KnowGRRF	470 486	0.2391	23.016	0.1127	21.965	0.1267
party	987 641	0.2957	142.891	0.3668	133.697	0.7191
RFSRC	573 353	0.1057	99.490	0.1676	95.790	0.2827
ranger	477 804	0.0489	90.000	0.0808	102.208	0.1327
$\mathbf{RLT_{1,0.2}}$	79 399	25.2492	28.159	27.7455	22.080	31.6840
$\mathbf{RLT_{1,0.5}}$	71 402	19.3870	24.430	20.4363	18.206	20.1803
$\mathbf{RLT_{2,0.2}}$	80 152	25.2727	17.177	28.0479	11.333	32.2992
$\mathbf{RLT_{2,0.5}}$	73 307	19.2715	13.900	20.7216	8.491	19.9887
$\mathbf{RRF_{0.5}}$	474 077	0.3325	89.854	0.4204	99.806	0.8975
$\mathbf{RRF_{0.8}}$	478 431	0.3304	89.786	0.4186	101.784	0.8953
$\mathbf{GRRF_{0.4}}$	477 913	0.3333	89.886	0.4196	101.472	0.8891
$\mathbf{GRFF_{0.6}}$	473 703	0.3330	89.253	0.4141	96.027	0.9166

It is clear from Tables 7, 8 and 9 that **RLT** has overall the lowest MSE when using the correct parameter settings. Regarding the runtime, **ranger** has overall the lowest runtime.

4.3 Results on the data sets with categorical and numerical variables

TABLE 10: MSE and runtime in seconds for the data sets CA, CB and CC per R-package

	CA MSE	CA runtime	CB MSE	CB runtime	CC MSE	CC runtime
randomForest	1154.432	0.5078	2058.861	0.5253	318.661	0.4228
extraTrees	-	-	-	-	-	-
KnowGRRF	1100.597	0.3421	2003.556	0.3636	318.661	0.4228
party	1520.923	0.6910	2443.505	0.5956	418.888	0.7273
RFSRC	1270.628	0.1661	2174.462	0.1795	344.183	0.1421
ranger	1132.658	0.0684	2056.985	0.0839	308.744	0.0530
$\mathbf{RLT_{1,0.2}}$	-	-	-	-	-	-
$\mathbf{RLT_{1,0.5}}$	-	-	-	-	-	-
$\mathbf{RLT_{2,0.2}}$	-	-	-	-	-	-
$\mathbf{RLT_{2,0.5}}$	-	-	-	-	-	-
$\mathbf{RRF_{0.5}}$	1150.285	0.4771	2062.325	0.5116	312.400	0.4267
$\mathbf{RRF_{0.8}}$	1150.843	0.4839	2062.794	0.4964	312.396	0.4150
$\mathbf{GRRF_{0.4}}$	1148.941	0.4863	2057.981	0.5144	312.232	0.4241
$\mathbf{GRFF_{0.6}}$	1149.188	0.4887	2060.693	0.4824	312.637	0.4046

From Table 10 it can be seen that the MSE is quite similar for all R-packages and that the runtime is much lower for **ranger** compared to the other R-packages.

4.4 Results on the data sets with correlated variables

TABLE 11: MSE and runtime in seconds for data sets DA, DB and DC per R-package

	DA MSE	DA runtime	DB MSE	DB runtime	DC MSE	DC runtime
randomForest	8.196	0.3915	8.145	0.3362	4.686	0.5056
extraTrees	7.889	0.1356	7.171	0.1029	4.245	0.1212
KnowGRRF	6.924	0.1508	14.884	0.2015	3.983	0.1049
party	10.513	0.3593	11.853	0.3052	4.682	0.3360
RFSRC	8.646	0.1533	9.210	0.1092	4.541	0.1382
ranger	8.190	0.0771	8.127	0.0509	4.689	0.0593
$\mathbf{RLT_{1,0.2}}$	1.698	37.8587	11.952	19.0354	3.421	18.0506
$\mathbf{RLT_{1,0.5}}$	1.337	24.6933	13.764	13.8167	3.276	12.9269
$\mathbf{RLT_{2,0.2}}$	0.741	36.8546	9.247	19.4601	3.268	17.9437
$\mathbf{RLT_{2,0.5}}$	0.495	28.0792	11.992	14.2328	3.162	12.9636
$\mathbf{RRF_{0.5}}$	8.182	0.3774	8.157	0.3235	4.692	0.4971
$\mathbf{RRF_{0.8}}$	8.209	0.3812	8.163	0.3230	4.694	0.4966
$\mathbf{GRRF_{0.4}}$	8.192	0.3819	8.150	0.3196	4.690	0.4937
$\mathbf{GRRF_{0.6}}$	8.165	0.3806	8.168	0.3229	4.686	0.4948

TABLE 12: MSE and runtime in seconds for data sets DD, DE and DF per R-package

	DD MSE	DD runtime	DE MSE	DE runtime	DF MSE	DF runtime
randomForest	6.776	0.3450	7.058	0.3366	4.846	0.5158
extraTrees	8.819	0.1037	4.260	0.1019	4.047	0.1265
KnowGRRF	0.922	0.0899	13.272	0.2370	4.716	0.1757
party	7.715	0.3083	12.503	0.2981	5.142	0.3565
RFSRC	7.131	0.1062	8.313	0.1061	4.871	0.1400
ranger	6.752	0.0496	7.085	0.0495	4.842	0.0597
$\mathbf{RLT_{1,0.2}}$	2.244	15.7706	15.533	22.3365	3.185	26.2374
$\mathbf{RLT_{1,0.5}}$	1.915	10.8121	20.867	16.7434	3.260	20.3467
$\mathbf{RLT_{2,0.2}}$	0.532	15.9826	11.732	22.7296	3.004	26.3262
$\mathbf{RLT_{2,0.5}}$	0.288	10.7058	17.216	17.3598	3.024	20.2786
$\mathbf{RRF_{0.5}}$	6.761	0.3263	7.157	0.3189	4.826	0.5052
$\mathbf{RRF_{0.8}}$	6.803	0.3222	7.124	0.3197	4.826	0.5071
$\mathbf{GRRF_{0.4}}$	6.762	0.3231	7.110	0.3139	4.833	0.5056
$\mathbf{GRRF_{0.6}}$	6.771	0.3211	7.319	0.3209	4.834	0.5078

TABLE 13: MSE and runtime in seconds for data sets DG, DH and DI per R-package

	DG MSE	DG runtime	DH MSE	DH runtime	DI MSE	DI runtime
randomForest	9.465	0.7771	9.224	0.3252	7.635	1.1538
extraTrees	14.265	0.2357	3.418	0.0981	6.011	0.3015
KnowGRRF	1.197	0.2036	17.961	0.2281	7.775	0.5337
party	8.592	0.7156	17.570	0.2842	9.501	0.9485
RFSRC	9.465	0.2063	9.576	0.1022	8.668	0.2707
ranger	9.497	0.1060	9.398	0.0485	7.587	0.1244
$\mathbf{RLT_{1,0.2}}$	3.744	36.5478	23.399	22.1106	4.243	55.3419
$\mathbf{RLT_{1,0.5}}$	3.209	26.1168	32.312	16.5422	4.521	40.2478
$\mathbf{RLT_{2,0.2}}$	0.895	40.4316	16.828	22.4248	3.542	53.9144
$\mathbf{RLT_{2,0.5}}$	0.414	27.9544	26.387	16.8341	3.840	40.7548
$\mathbf{RRF_{0.5}}$	9.318	0.7281	9.373	0.3117	7.572	1.1383
$\mathbf{RRF_{0.8}}$	9.436	0.7209	9.348	0.3126	7.576	1.1229
$\mathbf{GRRF_{0.4}}$	9.549	0.7291	9.426	0.3084	7.559	1.1320
$\mathbf{GRRF_{0.6}}$	9.418	0.7348	9.668	0.3109	7.575	1.1388

From Tables 11, 12 and 13 it can be noticed that **RLT** performs well on data set DA, DC, DD, DF, DG and DI regarding the MSE. These are the data sets containing several uninformative variables. Considering the runtime, it is clear that **ranger** has the lowest compared to other R-packages.

4.5 Results on frequency distribution of the runtime

In this section the frequency distribution of the runtimes on the data sets is considered. All the R-packages are run 100 times on the data set BB and BF and the frequency distribution of the runtimes are displayed in Figure 5 and Figure 6.

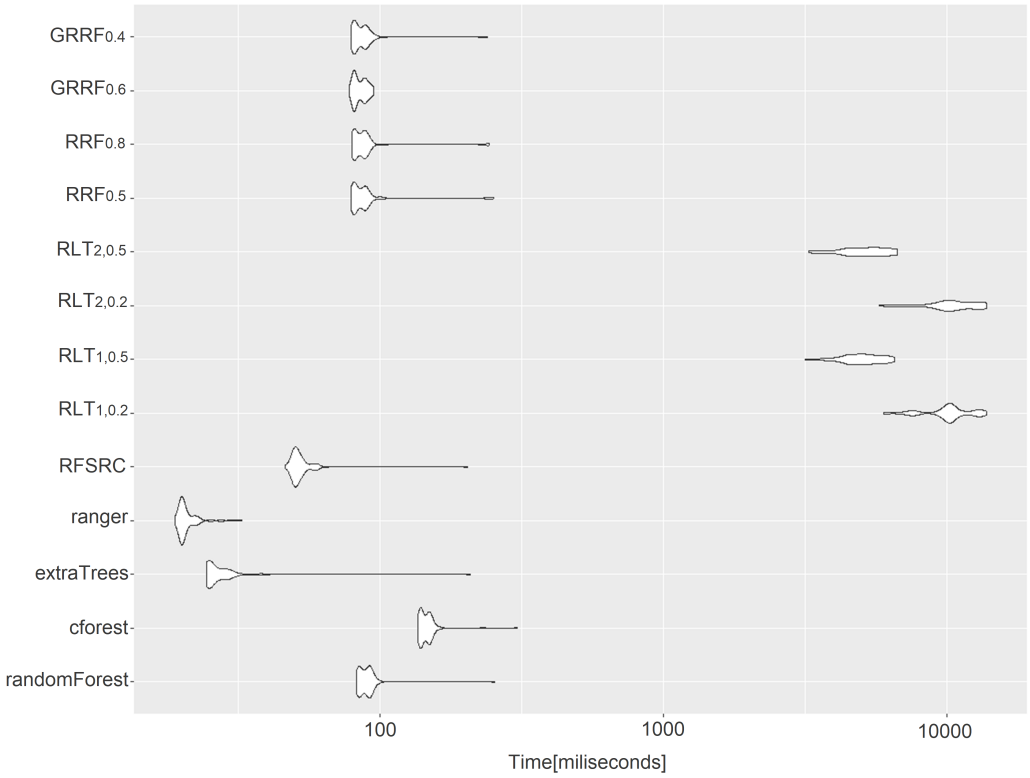

FIGURE 5: Runtimes of all R-packages on data set BB

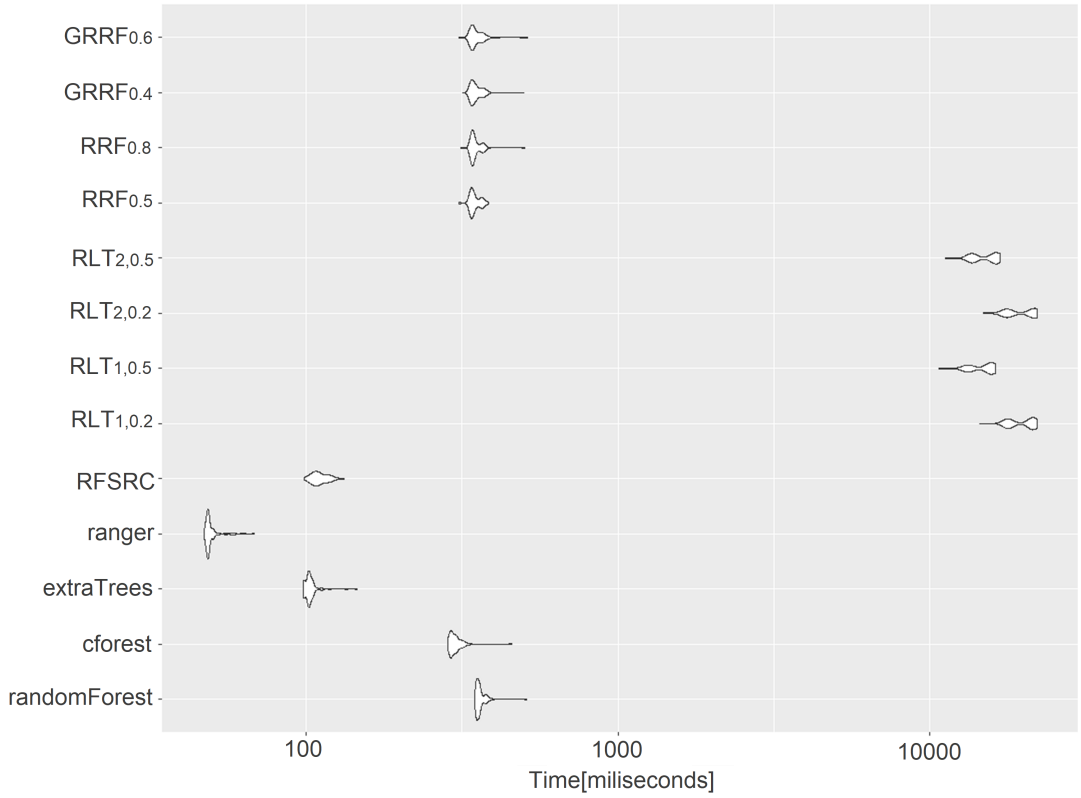

FIGURE 6: Runtimes of all R-packages on data set BF

From Figure 5 and Figure 6 it is noticeable that the variety in the runtimes of **RLT** is more scattered while the variety of the runtimes of the other algorithms is more concentrated and with a few outliers. This could be explained by the fact that **RLT** considers all variables as splitting candidates, except for the ones that are in the muted set whereas the other R-packages select *mtry* random potential splitting candidates and then choose the optimal split. This is also tested and demonstrated in Figure 7 and Figure 8 where **ranger** is run 1000 times on data set BB and BF, respectively. In Figure 7 *mtry* values of 1 and 5 are used and in Figure 8 *mtry* values of 1, 8 and 25 are used to see the difference in the time of the outliers. The R-package **ranger** was chosen to evaluate this, as this R-package had been noticed to be the fastest.

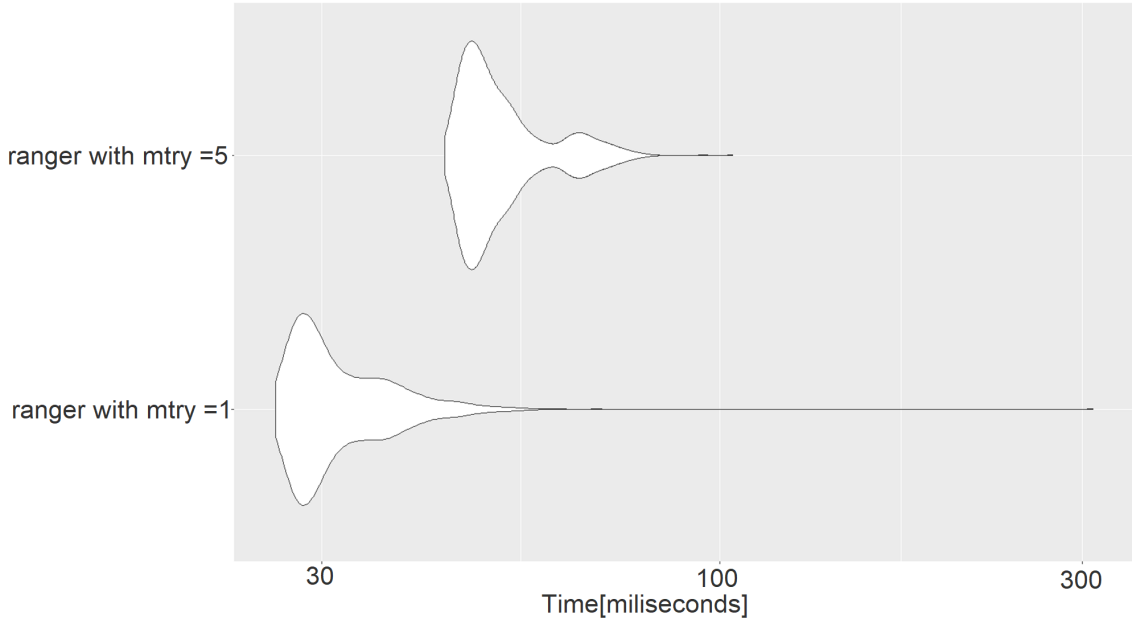

FIGURE 7: Runtimes of **ranger** on data set BB

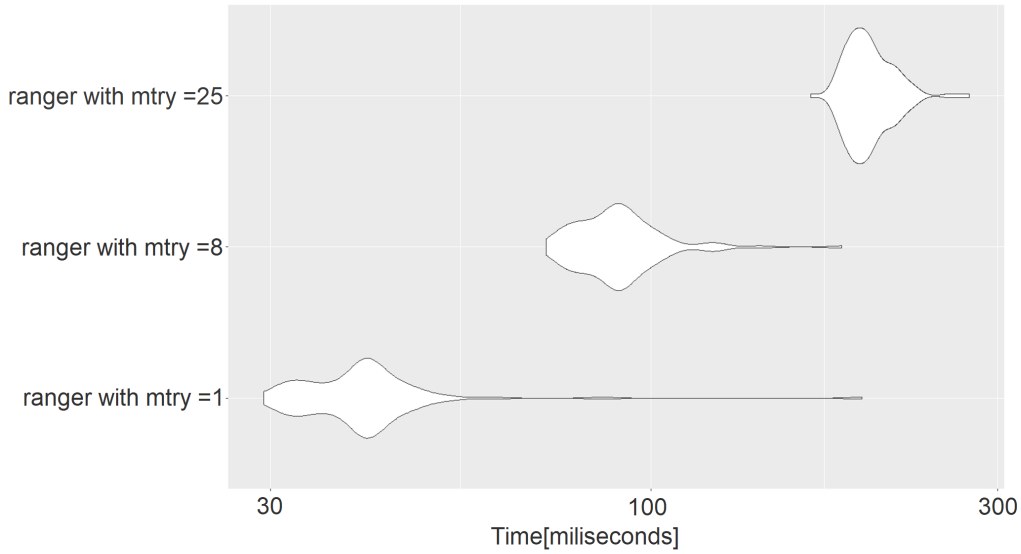

FIGURE 8: Runtimes of **ranger** on data set BF

4.6 Results on variable importance

This section shows the results concerning the variable importance. The measures for VI per R-package has already been discussed in Section 2.3. The methods for measuring VI considered are:

- Accuracy permutation which is the mean decrease in accuracy using permutation. This is in the R-packages **randomForest**, **party**, **randomForestSRC**, **ranger** and **RRF**.

- Node impurity which is the mean decrease in node impurity. This method is accessible in the R-packages **randomForest**, **ranger** and **RRF**.
- Accuracy random which is the mean decrease in accuracy using random assignment instead of permutation. This is available in the R-package **randomForestSRC**.
- Accuracy anti which is the mean decrease in accuracy using opposite assignment instead of permutation. This one is also available in the R-package **randomForestSRC**.
- Unbiased node impurity which is the unbiased mean decrease in node impurity. This can be retrieved from the R-package **ranger**.
- Unbiased accuracy permutation which is the unbiased mean decrease in accuracy using permutation. This method can be found in the R-package **party**.

For every data set the variable importance of every method is obtained and presented in the following tables. The values are obtained by adding the VI over 20 runs and then normalizing them by the maximum. This is done in order to make it easier to compare the importance of the variables. In Tables 14, 15 and 16 the VI for data sets AE, BH and DD are displayed, respectively. The results for the other data sets for the VI can be found in Appendix B. In the first column, the informative variables are marked in bold.

Table 14: Variable Importance on data set AE per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	100	100	100	100	100	100
2	74.24	45.09	39.08	35.25	36.59	48.06
3	−0.63	7.30	0.75	0.65	0.17	0.10
4	−3.30	6.50	0.01	−0.04	−0.88	−0.40
5	−4.27	6.58	−0.02	−0.03	−1.00	−0.12
6	0.47	7.54	1.05	0.88	0.79	−0.07
7	1.76	7.71	1.02	0.66	0.16	0.56
8	1.72	8.22	1.55	1.51	1.71	0.62

In Table 14 it can be noticed that all methods correctly identify the most important variables. In addition, accuracy permutation clearly identifies the second variable as more important than the other methods. Moreover, node impurity assigns compared to the other methods relatively high variable importance to the uninformative variables.

TABLE 15: Variable Importance on data set BH per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	47.97	23.12	24.34	21.59	19.84	17.15
2	100	100	100	100	100	100
3	−0.20	3.00	1.52	1.18	0.70	−0.04
4	−0.81	2.06	0.97	0.71	−0.74	−0.04
5	2.23	3.71	3.51	3.55	1.74	0.03
6	−1.25	2.60	1.65	1.37	−0.06	−0.18
7	−1.30	2.19	0.96	0.86	−0.81	−0.18
8	−0.14	2.61	1.68	1.52	−0.03	−0.09
9	−1.79	3.15	2.75	2.40	0.70	−0.03
10	−0.17	2.78	1.26	1.13	0.11	−0.04
11	−0.95	1.78	0.55	0.41	−1.23	−0.08
12	−0.80	2.04	0.61	0.46	−1.10	−0.07
13	−0.40	2.47	1.11	0.90	−0.36	−0.03
14	−0.73	2.24	0.71	0.50	−0.65	0.09
15	−2.45	3.13	2.92	2.52	0.61	−0.12
16	1.07	2.54	0.83	0.59	−0.21	0.03
17	−1.05	2.23	0.72	0.53	−0.58	−0.01
18	−2.02	2.04	0.55	0.38	−0.95	−0.17
19	−1.56	3.63	2.32	1.85	1.17	−0.15
20	1.70	3.00	2.19	1.97	−0.01	0.03
21	0.38	4.05	2.52	2.16	1.94	0.05
22	−0.01	2.34	1.15	0.88	−0.75	−0.04
23	0.73	2.25	1.19	0.84	−0.64	0.04
24	0.09	3.27	2.49	2.18	0.65	−0.05
25	−0.22	1.95	0.84	0.60	−1.14	−0.12

From Table 15 it is clear that all methods identify the 2 most important variables correctly. Accuracy permutation identifies the first variable significantly more important than the other methods. In addition, unbiased accuracy permutation clearly identifies the uninformative variables, as these have an assigned value very close to 0.

Table 16: Variable Importance on data set DD per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	14.61	10.30	3.70	4.04	17.51	0.60
2	11.31	3.15	1.42	1.00	4.81	0.41
3	100	100	100	100	100	100
4	4.80	2.09	1.08	0.75	3.43	0.16
5	9.27	4.25	1.44	1.25	8.66	0.38
6	4.63	2.24	0.96	0.69	4.38	0.13
7	13.55	5.98	1.71	1.37	10.99	0.53
8	6.50	3.52	1.78	1.24	6.81	0.35
9	4.96	2.05	0.82	0.61	3.55	0.06
10	13.37	8.68	4.12	3.34	15.19	1.03
11	11.46	5.84	2.04	1.81	11.88	0.52
12	10.14	7.40	2.77	2.27	13.23	0.77
13	6.31	2.25	1.06	0.69	4.39	0.17
14	10.30	6.76	3.31	2.88	12.57	0.89
15	55.33	44.31	28.90	26.82	51.23	24.26
16	2.99	1.36	0.50	0.33	1.95	0.02
17	11.98	8.27	4.01	3.42	13.56	0.45
18	14.93	7.47	3.10	2.55	14.34	1.60
19	0.80	1.40	0.35	0.24	1.20	0.04
20	3.86	2.17	0.94	0.63	3.16	0.10
21	12.10	5.19	2.20	1.86	10.32	0.31
22	9.05	3.36	1.77	1.31	7.13	0.41
23	8.43	3.10	1.70	1.14	5.71	0.11
24	11.03	4.78	2.77	2.58	9.27	0.43
25	8.67	4.65	1.91	1.53	9.52	0.81

From Table 16 it is noticeable that accuracy permutation assigns the highest value to the 2 informative variables. Moreover, unbiased accuracy permutation has again assigned values very close to 0 for all the uninformative variables compared to the other methods.

5 Conclusion

The aim of this study is investigate the performance of the random forests in the R-packages **extraTrees**, **party**, **randomForestSRC**, **ranger**, **RLT**, **RRF** and **KnowGRRF** and to provide guidelines on which R-package to use. The performance of the R-packages is measured on 3 aspects, namely the MSE, the runtime and the VI. Regarding the MSE, the R-package **RLT** works well on numerical data, but the runtime of **RLT** is significantly higher than the other R-packages. However, the muting percentage should be tuned appropriately to the data set. In case of many uninformative variables, the muting percentage can be chosen higher and in case of many informative variables, it should be kept low. On data sets with categorical variables, the R-package **ranger** would be a suitable option, as the MSE is similar to the other R-packages and the runtime is significantly faster. Focusing solely on the runtime of the R-packages, it can be concluded from the computations that the R-package **ranger** is preferred independent of the type of data. In addition, regarding the frequency distribution of the runtimes, it can be concluded that a lower value of $mtry$ increases the chance of having higher outliers in the runtime. This may result from the fact that a lower value for $mtry$ will give more randomness and increases the chance of selecting very poor splitting features making the runtime longer. Regarding the VI, several methods have been analysed, namely accuracy permutation, node impurity, accuracy random, accuracy anti, unbiased node impurity and unbiased accuracy permutation. From this, it can be concluded that the accuracy permutation and the unbiased accuracy permutation identify the important variables most accurately. The former clearly identifies the most important variables while the latter clearly identifies the true noise variables. Accuracy permutation can be found in many R-packages whereas unbiased accuracy permutation is only available in the R-package **party**. The reason why accuracy permutation performs better than node impurity, could be attributed to the fact that accuracy permutation obtains its result from a global effect over the whole tree, whereas node impurity acquires this from local points at every node in the trees.

Below is given a concise conclusion for every analysed R-package.

- **extraTrees**: compared to **randomForest**, this R-package is always faster on the data sets analyzed. This can be explained by the fact that **extraTrees** chooses the cut threshold randomly while **randomForest** chooses the best cut which is computationally more expensive. However, the performance does not show any significant difference. The MSE is in every run quite close to the MSE of **randomForest**.

- **party**: compared to **randomForest**, this R-package always performs worse, except on data set AC and data set AD. These are the categorical data sets with variables with many classes. As for the runtime, it is in general much slower than **randomForest**. The higher runtime could be caused by performing a significance test to split the nodes. The results for the MSE, could perhaps be attributed to the settings for the unbiased trees being not optimally set.

- **randomForestSRC**: compared to **randomForest**, this R-package is always faster. As for the MSE of **randomForestSRC**, it performs slightly worse than **randomForest**. These results could be explained by the fact that **randomForest** samples the data sets without replacement, which is computationally less expensive.

- **ranger**: compared to **randomForest**, this R-package is always significantly faster. It is also the fastest R-package compared to all the other R-packages. The perfor-

mance is approximately the same as **randomForest**. The fastness of the algorithm can be explained by all the methodological decisions made to speed up the algorithm.

- **RRF**: compared to **randomForest**, the performance of this R-package and all optional parameters is very similar. No significant improvement of MSE and runtime on the data sets are made. An option why no significant improvements is made, could be that the parameters were not chosen optimally.

- **RLT**: on data sets with numerical variables it performs overall better compared to **randomForest**. This can be explained by the fact that **RLT** considers all variables for splitting while *mtry* has not yet be chosen optimally for **randomForest**. It is noticeable that on data set BA which only has informative variables, the performance gets worse when more variables are muted. Similarly, on data set BH which has not only informative variables, the performance gets better when more variables are muted. Furthermore, in both cases the performance gets better when the split is made on a linear combination of 2 variables instead of only one variable. This is effect is more prominent on datasets where the response variable is predicted by a linear relation.

- **KnowGRRF**: compared to **randomForest**, the MSE of the R-package on the data sets is improved, especially when only the correct features are chosen. The improvement is most prominent on data sets with a lot of uninformative variables. In addition, the runtime is also often faster, which can be explained by the fact that the random forest is grown on less variables.

It might be interesting for future research to also look at other R-packages with different random forest algorithms. Another alternative is to look at the computational complexity of the algorithms or to use different kinds of data, for example using the sine function or logarithm.

Acknowledgements

I want to thank my supervisor M.N.M. van Lieshout for taking the time to meet me every week and her valuable guidance. In addition, I want to thank the people close to me for their support.

References

[1] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct. 2001.

[2] L. Breiman and A. Cutler. *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*, r-package version 54.6-14 edition, Mar. 2018.

[3] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. USA: Wadsworth, Belmont, CA, 1984.

[4] H. Deng and X. Guan. *RRF: Regularized Random Forest*, r-package version 1.9.1 edition, July 2019.

[5] H. Deng and G. Rungeri. Gene selection with guided regularized random forest. *Pattern Recognition*, 46(12):3483–3489, Dec. 2013.

[6] H. Deng and G. Rungeri. Feature selection via regularized trees. *The 2012 International Joint Conference on Neural Networks*, Jun. 2012.

[7] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63:3–42, Mar. 2006.

[8] X. Guan and L. Liu. *KnowGRRF: Knowledge-Based Guided Regularized Random Forest*, r-package version 1.0 edition, Mar. 2019.

[9] X. Guan, G. Runger, and L. Liu. Dynamic incorporation of prior knowledge from multiple domains in biomarker discovery. *BMC Bioinformatics*, 21(2):3–14, Mar. 2020.

[10] H. Ishwaran H and U. Kogalur. *randomForestSRC: Fast Unified Random Forests for Survival, Regression, and Classification (RF-SRC)*, r-package version 2.11.0 edition, Mar. 2021.

[11] T. Hothorn, K. Hornik, C. Strobl, and A. Zeileis. *party: A Laboratory for Recursive Partytioning*, r-package version 1.3-7 edition, Mar. 2021.

[12] T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, Sep. 2006.

[13] A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, Dec. 2002.

[14] O. Mersmann. *Accurate Timing Functions*, r-package version 1.4-7 edition, Sep. 2019.

[15] S. Nembrini, I. R. König, and M. N. Wright. The revival of the gini importance? *Bioinformatics*, 34(21):3711–3718, May 2018.

[16] J. Simm and I. Magrans de Abril. *extraTrees: Extremely Randomized Trees (ExtraTrees) Method for Classification and Regression*, r-package version 1.0.5 edition, Feb. 2015.

[17] J. Simm, I. Magrans de Abril, and M. Sugiyama. *Tree-Based Ensemble Multi-Task Learning Method for Classification and Regression*, 2014.

[18] M.N. Wright, S. Wager, and P. Probst. *ranger: A Fast Implementation of Random Forests*, r-package version 0.2-3 edition, Jan. 2020.

[19] M.N. Wright and A. Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, Mar. 2017.

[20] R. Xia. Comparison of random f comparison of random forests and cforests and cforest: Variable importance measures and prediction accuracies. *All Graduate Plan B and other Reports. 1255*, 2009.

[21] R. Zhu. *RLT: Reinforcement Learning Trees*, r-package version 3.2.2 edition, Aug. 2018.

[22] R. Zhu, D. Zeng, and M.R. Kosorok. Reinforcement learning trees. *Journal of the American Statistical Association*, 110(512):1770–1784, Feb. 2015.

[23] A. Ziegler and I. R. Konig. Mining data with random forests: current options for real-world applications. *WIREs Data Mining Knowl Discov*, 4(1):55–63, Dec. 2013.

A Alternative R-packages

1. R-package **adabag**. This R-package contains the Adaboost algorithm, which stands for adaptive boosting. This R-package has not been selected, as it was beyond the scope of this research.
   See: https://CRAN.R-project.org/package=adabag

2. R-package **blockForest**. This R-package uses block-structured covariate data for prediction. It was not selected as it was beyond the scope of this research.
   See: https://CRAN.R-project.org/package=blockForest

3. R-package **drf**. It estimates the multivariate conditional distribution based on their joint conditional distribution. This R-package was not selected as it was relatively a new method and not yet well documented.
   See: https://CRAN.R-project.org/package=drf

4. R-package **grf**. It gives as output an estimate of the predictive distribution. This R-package was therefore not selected as it would be complicated to compare to **randomForest**.
   See: https://CRAN.R-project.org/package=grf estimation of predictive distributions

5. R-package **h2o**. This R-package is an interface for the 'H2O' Open Source Machine Learning Platform. It also contains a function for random forest. This R-package is not selected as random forests are not considered to be the main point of this R-package.
   See: https://CRAN.R-project.org/package=h2o

6. R-package **hyperSMURF**. This random forest handles highly imbalanced data by oversampling the minority class and undersampling the majority class. This R-package was not selected as it cannot be used for regression tasks.
   See: https://CRAN.R-project.org/package=hyperSMURF

7. R-package **iRF**. This R-package grows feature weighted random forests. This R-package was not selected as it was beyond the scope of this research.
   See: https://CRAN.R-project.org/package=iRF

8. R-package **JRF**. This R-package contains joint random forest for estimating multiple related networks. This R-package was not chosen as it is not commonly used.
   See: https://CRAN.R-project.org/package=JRF

9. R-package **LongituRF**. It contains a random forest constructed for high-dimensional longitudinal data. This R-package was not selected as it was too data specific.
   See: https://CRAN.R-project.org/package=LongituRF

10. R-package **obliqueRF**. A random forest that consists of oblique decisions trees. This R-package was not selected as it could not be used for regression tasks.
    See: https://CRAN.R-project.org/package=obliqueRF

11. R-package **orf**. This R-package is similar to **randomForest**, but it can also take into account ordering information of the categorical outcome variable. This R-package was not chosen as it was beyond the scope of this research.
    See: https://CRAN.R-project.org/package=orf

12. R-package **quantreg**. This algorithm gives the full conditional distribution of a response variable. Therefore, this R-package was not selected as the final estimate would be complicated to compare to **randomForest**.
    See: https://CRAN.R-project.org/package=quantreg

13. R-package **RandomForestGLS**. This R-package is an extension of random forests for the case of dependent error processes. This R-package is less widely used.
    See: https://CRAN.R-project.org/package=RandomForestsGLS

14. R-package **randomUniformForest**. The forest is constructed by unpruned trees. The cut points at each node is selected using the continuous uniform distribution. This R-package was not selected as it is less commonly used.
    See: https://CRAN.R-project.org/package=randomUniformForest

15. R-package **Rborist**. It is an optimized form of **randomForest** as it is faster. This R-package was not selected as the construction of the algorithm was not well documented.
    See: https://CRAN.R-project.org/package=Rborist

16. R-package **rFerns**. It build a random ferns model of the data. The model is based on extending the naïve Bayes classifier. This R-package was not selected as it can only be used for classification tasks.
    See: https://CRAN.R-project.org/package=rFerns

17. R-package **RGF**. This R-package is an interface to a Python implementation of regularized greedy forests. This R-package was not selected as it was beyond the scope of this study.
    See: https://CRAN.R-project.org/package=RGF

18. R-package **rotationForest**. This method obtains its predicted value using feature extraction. This R-package was not chosen as it only works for classification.
    See: https://CRAN.R-project.org/package=rotationForest

19. R-package **trtf**. In this R-package a transformation is grown using transformation trees. It can detect distributional changes and it gives as output value an estimation of the conditional distribution function. This R-package was not chosen, as the final estimate would be difficult to compare to **randomForest**.
    See: https://CRAN.R-project.org/package=trtf

20. R-package **wsrf**. It implements an alternative variable weighting method for variable subspace selection. This R-package was not selected as it can only be used for classification.
    See: https://CRAN.R-project.org/package=wsrf

B Supplementary results on variable importance

TABLE 17: Variable Importance on data set AA per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	44.29	43.21	25.26	18.16	31.67	27.57
2	16.05	22.66	6.12	3.73	8.97	6.45
3	4.84	17.81	2.64	1.66	2.56	2.35
4	43.08	40.49	23.91	17.52	32.92	24.13
5	41.61	38.55	20.66	13.46	28.67	20.22
6	13.42	21.77	5.41	3.34	11.96	3.08
7	67.41	68.96	46.33	37.06	67.00	79.70
8	9.83	22.09	3.87	2.89	11.63	3.89
9	100	100	100	100	100	100
10	27.64	29.33	11.37	7.61	16.86	13.16
11	5.62	18.95	2.60	1.65	3.58	2.06
12	12.50	18.69	4.40	2.52	2.14	1.24
13	29.24	30.38	13.02	8.50	14.76	13.56
14	6.17	19.45	2.83	1.51	2.39	1.34
15	8.93	18.75	2.52	1.36	2.55	0.92
16	43.63	50.22	25.37	22.41	47.45	38.76

TABLE 18: Variable Importance on data set AB per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	97.91	97.84	100	100	97.72	65.82
2	37.27	39.45	22.39	17.71	19.00	14.66
3	12.84	27.82	8.80	5.91	2.53	4.22
4	96.05	88.58	93.45	85.84	84.54	68.75
5	91.76	82.63	81.35	69.82	74.71	55.14
6	46.10	46.12	32.25	26.10	33.37	18.24
7	100	100	98.06	90.02	100	100
8	10.58	30.18	9.39	7.40	11.99	5.47

TABLE 19: Variable Importance on data set AC per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	96.97	98.01	100	100	6.80	65.74
2	16.61	64.88	22.30	18.64	1.19	−1.63
3	68.27	84.82	56.41	53.02	14.47	43.02
4	100	100	88.16	88.52	60.81	100
5	54.67	86.55	56.81	57.66	100	41.71
6	67.96	81.86	56.57	55.27	41.65	45.85
7	50.90	80.39	43.42	42.05	30.76	8.85
8	87.33	88.63	71.57	73.22	76.37	28.40

TABLE 20: Variable Importance on data set AD per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	94.86	31.97	80.16	85.32	91.06	75.37
2	37.91	16.15	37.68	32.07	32.84	30.07
3	1.95	9.26	2.39	1.55	1.71	1.15
4	100	33.97	100	100	100	100
5	91.24	96.77	38.71	29.63	26.22	86.09
6	16.51	69.57	14.70	9.95	0.06	18.39
7	65.05	92.28	28.49	22.24	29.27	38.70
8	86.73	100	27.25	21.85	21.00	56.58

TABLE 21: Variable Importance on data set AF per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	81.43	70.56	98.01	93.08	68.20	66.53
2	36.29	19.12	17.94	13.97	10.82	8.40
3	8.38	11.95	9.33	6.68	4.66	1.32
4	83.94	100	100	100	100	85.44
5	73.09	39.51	55.09	45.84	33.75	37.51
6	54.73	36.41	47.47	42.73	35.99	27.93
7	100	91.78	84.49	78.69	90.18	100
8	9.70	5.55	5.54	4.23	−1.38	1.32

TABLE 22: Variable Importance on data set BA per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	34.13	38.66	34.60	33.37	9.46	13.06
2	100	100	100	100	100	100
3	23.21	38.60	21.68	19.94	13.69	12.49
4	74.92	67.91	59.84	58.97	52.40	48.87
5	51.01	50.20	39.94	39.55	26.47	27.69

TABLE 23: Variable Importance on data set BB per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	0.95	18.12	7.02	5.82	1.62	0.58
2	100	100	100	100	100	100
3	−4.27	17.94	10.72	11.07	1.66	−1.15
4	−5.69	18.44	15.72	16.12	1.93	−1.83
5	0.49	17.49	8.21	8.23	0.86	0.12

TABLE 24: Variable Importance on data set BC per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	100	100	100	100	100	100
2	1.45	16.87	9.26	8.41	1.56	0.01
3	−3.38	16.80	9.09	8.08	1.55	0.27
4	1.14	15.06	7.20	6.17	−1.22	0.48
5	−1.30	17.50	8.79	7.51	2.86	−0.11

TABLE 25: Variable Importance on data set BD per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	100	100	100	100	100	80.60
2	91.63	90.83	97.22	97.64	89.00	100
3	−1.83	33.83	20.62	18.92	5.40	1.39
4	−6.67	37.00	37.86	40.53	11.52	−2.96
5	1.35	33.63	17.35	15.09	3.99	1.72

Table 26: Variable Importance on data set BE per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	74.83	71.33	74.11	72.19	41.63	44.99
2	61.55	76.33	84.72	82.68	52.93	59.04
3	100	100	100	100	100	100
4	80.45	80.25	80.03	78.07	58.72	61.97
5	70.05	67.82	62.44	59.51	34.60	56.90

Table 27: Variable Importance on data set BF per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	−2.93	21.65	4.53	1.84	−7.72	−1.63
2	88.22	94.39	100	100	89.34	50.58
3	−17.23	21.17	7.09	3.08	−5.40	−1.58
4	10.06	35.57	37.00	30.45	14.03	−1.30
5	54.61	58.75	48.32	39.95	48.48	29.81
6	13.76	37.71	22.65	16.52	19.31	6.18
7	38.25	49.35	43.47	35.49	33.07	9.54
8	1.02	25.40	8.19	4.48	−0.11	0.18
9	97.75	86.31	72.28	75.42	79.41	45.05
10	37.56	39.28	20.02	17.26	19.83	10.79
11	100	100	86.80	84.93	100	100
12	5.77	43.17	30.99	25.08	24.05	−1.89
13	33.37	33.13	15.86	9.96	8.94	7.64
14	11.40	40.50	20.58	12.94	18.38	4.10
15	40.15	41.69	34.21	27.51	23.69	3.10
16	5.47	21.28	5.48	2.09	−6.34	0.60
17	53.08	57.36	43.48	34.32	39.54	15.92
18	26.81	36.61	22.03	15.89	16.85	11.59
19	41.52	57.58	28.06	22.35	44.59	29.29
20	5.79	24.98	10.49	5.52	−0.66	1.20
21	59.49	56.68	41.74	37.06	43.60	24.65
22	44.71	52.47	40.27	33.62	40.95	26.13
23	47.48	45.96	30.26	23.69	27.73	17.54
24	6.06	24.88	8.50	4.80	−4.63	−0.02
25	61.16	61.05	36.40	29.54	54.08	33.23

TABLE 28: Variable Importance on data set BG per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	0.66	2.43	0.83	0.78	0.55	0.03
2	100	100	100	100	100	100
3	−0.59	2.94	1.40	1.19	0.82	−0.15
4	−0.69	1.60	0.59	0.43	−0.71	−0.14
5	0.44	1.73	1.50	1.48	−0.42	−0.07
6	−0.72	1.87	1.11	0.96	−0.20	0.07
7	−0.64	1.58	0.67	0.56	−0.67	−0.10
8	−0.43	2.03	1.05	0.93	−0.22	−0.09
9	−2.54	3.24	3.94	3.51	1.00	−0.07
10	1.14	3.14	1.53	1.49	1.17	0.05
11	−0.85	1.44	0.47	0.45	−0.86	−0.06
12	−0.43	1.69	0.87	0.79	−0.54	−0.04
13	−0.90	1.70	0.64	0.55	−0.68	0.01
14	0.38	1.71	0.70	0.58	−0.58	0.10
15	−1.94	1.82	1.40	1.31	−0.38	−0.05
16	0.26	1.87	0.66	0.49	−0.37	0.03
17	1.26	2.31	0.97	0.91	0.21	0.10
18	−1.69	1.41	0.47	0.37	−0.74	−0.21
19	−1.60	2.63	1.53	1.24	0.60	−0.16
20	0.15	2.53	2.08	1.93	0.54	−0.01
21	−1.03	2.27	1.83	1.58	0.12	−0.00
22	0.09	1.99	0.66	0.58	−0.12	−0.07
23	0.25	1.56	0.56	0.43	−0.96	0.03
24	−0.13	1.86	1.34	1.08	−0.44	0.00
25	−0.34	1.56	0.48	0.38	−0.91	−0.01

TABLE 29: Variable Importance on data set BI per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	42.38	18.55	17.50	15.50	17.09	16.17
2	100	100	100	100	100	100
3	0.81	1.74	0.54	0.45	0.25	0.03
4	-1.27	0.97	0.20	0.13	-0.51	-0.04
5	-0.02	1.13	0.62	0.55	-0.37	-0.04
6	-1.13	1.04	0.31	0.19	-0.69	0.04
7	-0.25	1.27	0.74	0.58	-0.20	-0.02
8	1.03	1.29	0.52	0.45	-0.16	-0.01
9	1.31	1.87	2.03	1.73	0.40	0.10
10	-0.98	1.49	0.74	0.66	-0.03	-0.01
11	-0.40	1.12	0.25	0.16	-0.19	0.05

Continued on next page

Table 29 – continued from previous page

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
12	0.02	0.92	0.10	0.06	-0.70	-0.03
13	0.38	1.52	0.81	0.66	0.48	-0.00
14	-0.36	1.28	0.35	0.31	-0.36	0.03
15	-0.74	2.16	2.25	1.75	0.48	-0.14
16	3.58	2.18	2.46	1.76	0.58	-0.01
17	0.68	1.71	0.89	0.74	0.35	0.04
18	0.04	0.93	0.20	0.15	-0.42	-0.01
19	-2.03	2.24	1.81	1.34	0.84	-0.22
20	-0.96	0.86	0.11	0.07	-0.69	-0.02
21	0.17	1.56	0.58	0.42	0.07	0.06
22	-1.07	1.07	0.39	0.28	-0.39	-0.02
23	-0.00	3.36	1.80	1.53	2.47	0.06
24	-0.15	1.28	0.43	0.31	-0.22	-0.00
25	-0.80	0.97	0.16	0.13	-0.49	-0.05
26	-0.63	1.18	0.37	0.26	-0.27	0.00
27	-0.40	0.90	0.26	0.17	-0.59	-0.02
28	-0.11	1.08	0.62	0.53	-0.42	0.01
29	-0.73	0.76	0.18	0.13	-0.76	-0.02
30	-0.79	1.92	1.65	1.21	0.75	-0.03
31	0.13	1.21	0.38	0.32	-0.45	0.02
32	-1.09	1.47	0.96	0.72	0.00	0.00
33	-1.47	1.21	0.44	0.36	-0.30	-0.07
34	-0.98	0.93	0.20	0.14	-0.87	-0.01
35	-0.16	1.16	0.55	0.42	-0.41	-0.06
36	-0.06	1.58	0.52	0.47	0.13	-0.03
37	-0.37	1.54	0.36	0.27	0.47	-0.02
38	0.25	1.30	0.51	0.39	-0.39	-0.01
39	0.15	1.26	0.38	0.27	-0.45	-0.03
40	-0.33	0.94	0.26	0.19	-0.71	-0.06
41	0.22	1.77	1.27	1.00	0.33	-0.01
42	-0.37	1.20	0.33	0.23	-0.34	-0.08
43	-1.54	0.95	0.37	0.26	-0.76	-0.02
44	1.64	2.79	0.67	0.56	1.97	0.16
45	2.62	2.17	1.26	1.17	1.28	0.20
46	0.14	1.17	0.32	0.22	-0.33	0.08
47	-0.71	0.97	0.28	0.20	-0.80	0.00
48	-1.13	1.70	0.85	0.72	0.38	-0.02
49	-0.18	1.56	0.32	0.22	0.08	0.02
50	0.15	1.45	0.53	0.42	-0.13	0.01

TABLE 30: Variable Importance on data set CA per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	7.75	8.17	2.24	1.51	3.38	0.79
2	−2.23	4.68	0.02	−0.00	−0.42	−0.14
3	1.87	4.98	0.69	0.59	0.60	−0.02
4	21.54	17.54	12.67	12.07	21.81	5.35
5	6.09	6.84	2.50	2.09	4.42	1.37
6	6.88	6.53	1.99	1.48	2.95	1.47
7	1.47	5.69	0.59	0.45	0.83	0.12
8	0.22	4.89	0.05	0.09	−0.66	−0.10
9	46.78	44.35	31.71	26.26	35.53	18.79
10	−1.79	14.32	5.70	3.88	1.00	−0.07
11	20.31	22.99	14.93	11.04	11.03	4.93
12	18.42	23.27	12.57	9.01	9.72	3.65
13	63.77	63.23	46.49	42.41	58.38	46.92
14	13.39	21.46	11.69	8.85	9.56	2.76
15	100	100	100	100	100	100
16	17.93	24.05	11.73	8.52	10.94	2.59

TABLE 31: Variable Importance on data set CB per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	−4.41	5.84	−0.41	−0.32	−0.36	−0.92
2	7.87	9.32	2.85	2.14	5.85	2.85
3	37.10	44.40	26.62	18.91	32.58	19.44
4	1.44	21.81	5.10	3.58	4.15	−0.03
5	28.25	37.03	19.52	13.78	21.41	13.85
6	26.42	39.02	27.06	21.11	24.07	12.38
7	35.32	41.37	21.97	15.74	27.37	18.89
8	25.17	35.69	21.25	16.43	23.84	10.29
9	62.08	77.45	56.32	50.84	71.25	65.62
10	1.53	19.71	10.34	6.84	0.40	1.07
11	47.61	66.15	54.89	45.27	59.60	32.80
12	11.39	27.13	11.18	7.84	13.12	4.02
13	75.93	83.04	63.17	55.52	79.86	78.21
14	0.51	20.51	5.25	3.51	−0.89	−0.40
15	100	100	100	100	100	100
16	30.46	40.87	19.29	14.77	32.13	27.19

TABLE 32: Variable Importance on data set CC per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	19.82	12.78	9.36	7.89	12.60	8.06
2	8.42	8.19	2.62	1.81	3.52	1.76
3	1.53	5.79	0.83	0.53	1.49	−0.22
4	36.79	25.20	24.14	24.17	28.34	16.40
5	9.08	7.52	2.61	1.88	2.94	1.19
6	9.55	9.19	4.14	3.08	6.52	1.68
7	14.74	10.71	6.35	5.44	9.70	6.25
8	−1.30	4.44	0.02	−0.01	−0.88	−0.11
9	46.74	26.56	30.31	28.76	28.44	30.18
10	4.81	6.46	1.71	1.36	0.95	0.49
11	0.48	5.57	0.62	0.38	0.61	−0.11
12	0.21	4.86	0.23	0.12	−0.04	0.03
13	4.79	6.18	1.59	0.98	2.11	0.43
14	3.95	5.80	1.52	1.05	1.86	0.69
15	100	100	100	100	100	100
16	3.84	15.41	6.52	4.61	1.39	0.61

TABLE 33: Variable Importance on data set DA per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	12.57	9.73	3.73	3.37	14.58	1.13
2	28.13	26.43	15.38	15.91	33.10	7.61
3	100	100	100	100	100	100
4	20.49	16.01	8.48	7.46	22.34	4.08
5	6.95	6.38	2.55	2.12	9.77	0.51
6	0.50	2.50	0.73	0.49	1.69	−0.07
7	0.28	1.87	0.54	0.33	0.14	−0.02
8	−1.64	1.73	0.47	0.33	−0.51	−0.04
9	0.10	2.04	0.50	0.30	−0.02	−0.03
10	0.16	2.37	0.58	0.37	0.82	0.01
11	4.22	2.77	1.33	0.90	1.20	0.02
12	2.22	2.50	0.84	0.56	0.98	−0.04
13	6.48	3.63	1.65	1.12	2.64	0.31
14	26.20	12.14	6.73	5.34	15.25	3.12
15	65.59	31.78	29.79	24.84	34.36	33.41
16	10.25	4.49	1.74	1.22	4.83	0.25
17	6.20	4.87	2.57	1.87	5.06	0.17
18	1.90	2.43	0.98	0.64	1.19	0.06
19	−0.24	1.96	0.70	0.61	0.49	−0.08
20	−0.15	1.58	0.46	0.31	−0.44	−0.03
21	−0.59	1.44	0.34	0.21	−0.87	−0.02
22	−0.23	1.64	0.40	0.25	−0.55	−0.00
23	−0.07	1.99	0.43	0.30	0.11	0.02
24	−0.87	1.64	0.45	0.30	−0.61	0.01
25	−1.73	1.46	0.22	0.12	−0.96	−0.04

TABLE 34: Variable Importance on data set DB per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	43.62	28.66	29.07	20.47	24.20	9.64
2	56.60	34.14	38.52	27.90	32.87	11.44
3	71.61	58.19	60.10	44.30	58.26	21.16
4	57.22	39.00	38.31	27.23	38.14	10.68
5	82.87	73.38	69.83	58.47	72.66	34.06
6	78.65	65.24	60.57	48.43	66.43	27.68
7	56.65	37.53	31.99	23.34	36.37	9.89
8	69.81	51.28	46.44	36.08	52.44	33.87
9	52.90	36.41	31.61	22.86	35.58	11.62
10	77.47	64.15	58.67	50.92	65.52	22.01
11	62.70	48.22	39.21	31.62	49.45	14.46
12	95.95	95.68	92.52	88.67	92.56	39.09
13	54.84	36.10	28.31	22.01	41.20	9.90
14	94.66	93.82	79.51	74.44	92.59	28.42
15	100	100	100	100	100	100
16	53.20	34.59	26.61	21.29	38.78	5.04
17	64.49	41.72	37.03	27.53	43.38	10.16
18	77.36	50.74	51.26	40.24	51.08	18.30
19	71.99	62.08	53.88	46.59	61.19	23.24
20	58.88	40.66	43.66	30.84	39.41	13.23
21	79.47	63.15	62.57	47.10	63.10	33.76
22	79.31	57.90	57.66	43.40	56.36	32.31
23	29.84	18.10	14.99	10.05	11.69	1.81
24	60.22	35.24	34.44	24.15	32.26	10.97
25	36.04	22.80	19.17	13.45	16.81	3.01

TABLE 35: Variable Importance on data set DC per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	3.92	12.54	6.76	4.85	8.81	11.05
2	84.63	77.52	72.04	64.74	78.97	66.59
3	100	100	100	100	100	100
4	21.89	24.39	19.75	18.27	24.65	9.98
5	13.73	21.16	14.25	11.67	20.31	−1.11
6	68.54	50.58	26.97	19.90	49.96	42.91
7	71.60	49.83	26.86	18.90	49.52	27.65
8	8.68	12.41	3.78	2.70	7.95	8.18
9	−1.72	6.49	2.05	1.34	−0.86	1.72
10	2.32	9.49	4.61	3.56	2.84	7.96
11	−6.25	10.56	3.29	2.27	4.41	−2.80
12	41.87	27.47	17.58	13.11	23.81	15.57
13	−3.97	7.01	1.85	1.22	−1.27	−1.18
14	3.84	8.32	1.64	1.11	1.28	−0.32
15	2.13	11.33	3.21	2.15	3.47	4.79
16	0.02	8.26	1.59	1.02	0.19	−2.08
17	4.96	9.84	2.28	1.83	2.53	−1.85
18	7.73	7.63	1.09	0.71	−2.03	−0.04
19	4.87	8.88	3.91	2.87	1.45	−3.47
20	13.89	12.29	2.91	2.10	7.52	21.38
21	−4.58	9.55	4.05	2.59	1.94	−2.75
22	−6.90	7.40	2.99	1.92	−1.09	−3.86
23	−6.90	16.51	14.76	12.21	12.92	−1.36
24	4.72	18.85	13.62	10.68	14.30	9.70
25	8.94	7.78	2.09	1.34	−1.59	6.40

TABLE 36: Variable Importance on data set DE per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	69.15	58.24	57.39	58.15	68.86	29.93
2	43.73	8.93	13.55	8.05	14.11	7.89
3	74.11	39.70	47.66	32.93	54.01	46.92
4	52.98	12.51	16.66	10.51	20.01	8.20
5	73.71	36.59	40.88	30.81	48.77	39.58
6	67.11	18.96	26.34	17.32	28.29	23.45
7	66.58	22.95	28.61	20.06	34.02	18.68
8	58.85	26.86	34.40	24.92	41.58	25.16
9	62.55	20.14	26.07	16.96	30.12	23.34
10	83.02	40.15	50.53	33.32	48.93	46.05
11	72.23	55.56	52.89	46.86	68.81	45.15
12	100	100	100	100	100	100
13	47.90	11.18	15.95	9.30	16.64	6.94
14	74.35	27.27	36.91	24.19	35.83	24.08
15	90.97	86.46	81.95	79.50	90.18	79.48
16	45.45	10.73	14.47	9.41	18.30	7.15
17	70.21	28.71	34.84	21.81	36.97	16.36
18	74.15	40.04	44.87	33.59	55.62	40.43
19	59.33	16.66	24.17	15.81	25.68	17.91
20	45.57	10.06	15.22	9.75	18.09	5.63
21	79.55	36.69	50.38	35.74	51.99	50.65
22	69.68	24.07	33.12	22.92	33.45	30.89
23	64.26	22.47	29.58	19.13	30.92	16.40
24	64.35	35.16	39.51	33.11	48.12	30.01
25	78.61	52.07	56.36	46.26	66.00	62.41

TABLE 37: Variable Importance on data set DF per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	14.02	14.05	22.19	18.78	13.97	3.60
2	75.02	68.79	90.51	78.68	73.83	53.16
3	81.08	93.95	100	100	94.82	100
4	22.41	14.43	13.73	11.05	13.95	8.62
5	2.98	15.52	21.33	18.78	17.41	5.68
6	45.74	30.70	20.87	14.01	31.08	19.12
7	61.20	39.21	37.35	28.82	46.47	27.85
8	21.25	27.51	36.31	30.03	29.46	12.93
9	29.13	15.32	11.04	7.60	15.10	19.81
10	49.93	52.19	47.47	44.36	61.28	46.37
11	46.00	44.70	53.63	44.64	51.82	32.17
12	81.51	52.17	59.08	46.13	57.81	42.96
13	8.20	18.90	18.66	16.69	20.73	17.63
14	24.37	13.85	9.19	6.31	13.05	9.41
15	37.19	23.37	17.55	14.15	25.77	47.94
16	18.89	15.76	16.57	12.62	14.72	3.08
17	55.90	62.20	47.41	36.54	71.04	22.77
18	33.92	17.75	28.62	23.95	19.47	9.24
19	20.53	22.58	29.75	23.54	25.34	6.71
20	24.90	13.96	19.32	16.98	13.87	39.28
21	23.51	21.70	25.48	21.24	26.01	8.41
22	5.69	9.29	7.87	6.48	5.94	0.75
23	25.41	29.97	36.64	35.06	33.35	9.28
24	100	100	93.77	75.61	100	58.93
25	2.69	22.06	26.69	24.48	26.66	19.96

TABLE 38: Variable Importance on data set DG per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	28.69	22.55	7.85	9.53	34.17	1.03
2	14.41	3.31	1.38	1.20	9.91	0.44
3	100	100	100	100	100	100
4	15.06	2.34	0.61	0.98	5.78	0.56
5	19.46	3.81	1.80	1.20	10.27	1.11
6	15.06	3.45	1.51	1.16	10.35	0.29
7	25.02	9.61	3.58	3.02	19.66	1.99
8	13.97	3.30	1.73	1.10	8.59	1.42
9	16.00	4.07	1.62	1.28	11.09	0.58
10	21.92	6.45	4.09	2.81	16.10	3.20
11	19.47	7.26	2.97	2.77	18.46	1.73
12	21.69	10.67	3.57	3.32	22.92	1.67
13	15.75	3.10	1.74	1.23	10.04	0.75
14	20.37	6.42	4.83	3.34	14.86	1.66
15	58.89	47.65	29.51	27.70	60.71	32.43
16	11.32	2.82	1.37	1.12	8.78	0.16
17	22.07	7.19	3.53	2.34	15.78	1.24
18	25.11	7.01	4.49	3.01	16.72	3.45
19	9.55	1.22	0.47	0.34	3.77	0.15
20	15.36	4.58	1.91	1.68	13.10	0.51
21	23.08	10.95	5.77	4.83	23.29	1.61
22	15.02	2.87	1.48	1.00	8.54	0.64
23	20.35	6.61	3.48	2.52	16.84	1.89
24	19.46	5.49	3.71	2.58	15.16	1.75
25	17.99	7.37	3.96	3.23	18.12	1.72

TABLE 39: Variable Importance on data set DH per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	72.40	52.87	56.14	55.04	67.68	23.48
2	48.19	8.41	14.16	9.01	18.53	8.04
3	68.15	26.37	36.68	26.85	44.63	58.48
4	73.28	8.10	16.69	9.28	16.56	13.87
5	84.99	24.72	38.11	21.47	35.95	46.93
6	66.02	13.58	22.23	13.37	28.33	21.51
7	70.46	13.42	18.90	11.82	24.69	20.27
8	72.64	10.25	22.00	11.95	20.26	27.47
9	70.13	15.43	25.00	15.55	29.41	26.14
10	77.73	19.44	32.16	20.10	35.08	61.90
11	77.05	40.16	52.93	40.89	61.06	81.25
12	100	100	100	100	100	100
13	65.74	9.09	17.95	9.89	17.83	12.70
14	65.38	23.91	33.04	23.90	37.54	19.17
15	87.45	72.84	75.98	71.11	83.42	89.17
16	59.50	32.74	35.39	31.97	48.63	13.03
17	85.64	18.79	31.64	17.25	27.24	20.45
18	96.89	25.11	44.21	23.96	36.12	64.12
19	65.59	9.02	17.50	10.06	19.69	25.12
20	47.57	13.30	16.78	11.94	26.81	8.55
21	73.70	36.16	47.52	36.80	54.07	50.48
22	64.48	11.04	20.15	11.62	22.85	20.49
23	82.47	29.54	43.72	26.13	41.39	53.53
24	58.80	22.38	29.72	20.72	41.26	33.53
25	81.30	31.63	49.13	33.35	48.45	79.70

TABLE 40: Variable Importance on data set DI per method

Variable	Accuracy permutation	Node impurity	Accuracy random	Accuracy anti	Unbiased node impurity	Unbiased accuracy permutation
1	31.57	11.46	34.36	36.96	16.29	6.35
2	63.36	35.24	68.78	71.01	48.21	40.49
3	68.47	51.50	70.75	75.75	69.26	100
4	29.72	11.67	32.89	29.76	20.43	11.63
5	36.53	19.74	36.95	35.79	28.95	31.73
6	58.18	20.95	43.56	39.66	34.09	35.84
7	91.86	49.18	46.58	43.47	59.64	43.83
8	42.14	25.17	42.30	40.49	33.15	29.85
9	41.14	15.38	22.90	21.31	19.56	32.14
10	63.87	50.88	78.11	82.98	63.64	62.96
11	60.40	27.12	25.21	23.38	41.83	84.20
12	93.24	31.71	48.24	44.37	38.76	48.34
13	32.50	14.35	42.64	40.84	26.48	29.42
14	36.94	40.25	48.20	46.57	51.66	13.63
15	45.57	17.32	26.20	26.64	27.43	96.39
16	43.12	19.16	40.52	36.30	29.31	11.84
17	72.30	51.01	45.31	37.98	60.33	47.99
18	56.37	23.31	54.16	51.61	38.53	51.84
19	28.78	12.22	33.49	34.04	19.06	15.19
20	24.12	8.27	29.07	29.80	20.70	9.47
21	42.93	32.57	52.47	53.30	46.42	23.18
22	45.05	14.61	35.58	34.25	27.00	12.13
23	66.69	57.14	71.28	71.03	69.58	62.96
24	100	100	100	100	100	52.09
25	48.57	37.39	55.64	55.43	52.31	37.31