Università degli Studi di Milano - Bicocca

FACOLTÀ DI ECONOMIA

Corso di Dottorato di Ricerca in Statistica e Applicazioni XXV ciclo

Machine Learning Methods for Feature Selection and Rule Extraction in Genome-wide Association Studies (GWASs).

Discussant:
Stefano Nembrini

Matricola 734383

Supervisor:
Prof. Paola Zuccolotto

Co-supervisors:
Dott. Cristian Pattaro
Dott. Marco Sandri

Academic Year 2011 – 2012

Contents

List of Figures

List of Tables

Chapter 1

Preliminary Concepts

1.1 Introduction

The genetics and heredity of human complex traits have been studied for more than a century now, and many genes have been found to be implicated in such traits.

Recent advances in biotechnology, in particular in microarray technology, allow hundreds of thousands of measurements of a biosample, making genome-wide association studies (GWAS) possible. These were conceived with the purpose to study the association between common genetic variation and complex human traits using high-throughout platforms which measured hundreds of thousands of common single-nucleotide polymorphisms (SNPs).

Thanks to these studies many novel genetic loci associated with complex traits have been successfully identified primarily using a univariate regression-based approach.

In biological systems, proteins, DNA, RNA, and metabolites frequently interact to perform their biological function, and to respond to environmental factors.

The traditional regression-based methods usually provide a cozy framework to work upon, in that they are quite well established in the genetic field and produce a fairly understandable output, but they are limited to address the complex interaction among the hundreds of thousands of SNPs and their environment.

Many human traits and diseases display a high heritability, showing a

strong genetic element connected with such traits and diseases. The genetic studies in human populations have estimated the heritability and identified genetic loci linked to these human traits and diseases.

Although over 2000 genetic variants have been successfully identified associated with human diseases by GWAS using univariate approaches, these findings have been able to explain only a small portion of the heritability of most complex diseases (e.g. Chronic kidney disease (CKD)).

Many explanations of this *missing heritability* have been suggested (Manolio *et al.*, 2009):

1. Larger numbers of variants of smaller effects should be included in GWAS studies.
   In the last few years, a number of consortia have been created with the purpose of discovering novel genetic variants, this has brought a growth in sample sizes capable of detecting novel loci associated with human diseases, yet this loci - having small effects - contribute to explain additional variability to a limited extent.

2. Rarer variants (possibly with larger effects), that are inadequately detected by available current arrays, should be included.
   'Next-generation' sequencing (NGS) technologies - allowing full DNA sequencing - will facilitate the comprehensive catalogue of variants with minor allele frequency (MAF) that should account for larger variations of the study outcomes.

3. Structural variants that contribute to the basis of the disease, but are incompletely assessed by commercial SNP genotyping arrays, should be accounted for.
   Several approaches have been developed for taking into account such structural variation, e.g. integrating analysis of copy number variants into GWAS.

4. Low power to detect gene-gene and gene-environment interactions.

The purpose of this study focuses on this latter and consists in looking deeper into already available data, focusing on the use of different statistical models hopefully capable of detecting such interactions, with the prospect of explaining some additional variability or of giving a better insight of the underlying mechanisms.

The main reason is that, although most complex diseases involve multiple genes and their interactions, existing studies are mainly based on limited single-locus approaches, detecting SNPs essentially according to their marginal associations with phenotypes (Li *et al.*, 2011).

Rather than functioning linearly and independently, genes and their protein products function as interactive complexes in biological pathways, networks, and systems. The genetic architecture involves these higher order genetic interactions and their relationship with the environmental factors (Moore and Dunlap, 2010).

While on the one hand, the large amount of genotypic and phenotypic measurements on large samples from GWAS allows us to better understand the genetic architecture of complex diseases, on the other, large datasets (even millions of predictors and thousands of individuals) present hard computational challenges both in data management as well as in statistical analysis. To search for additional genetics effects of complex diseases and to improve the prediction of diseases, some alternative methods have been proposed to complement the classic statistical genetic methods, e.g. *tree ensembles*, which have recently gained popularity in GWAS due to their ability to capture the complex interaction effects that classical regression models lack of, e.g. random forests (Breiman, 2001) and Gradient Boosting Machines (Friedman, 2002).

The use of such methods is not new. Successful applications were conducted in various studies (see Chang *et al.* (2008), Bureau *et al.* (2005), Jiang *et al.* (2009), Sun *et al.* (2007)). For instance Lunetta *et al.* (2004) use random forests as an exploratory tool for genomic studies: they suggest the use of random forest when little is known about the genetic architecture of the trait (i.e. response) at hand. If interactions among SNPs exist, variable importance measures provided by the method will reflect this. If this is the case, they claim the random forest approach will outperform a univariate ranking method (Fisher Exact test in this case) when one is interested in screening large numbers of SNPs among a large number of irrelevant SNPs. It is however important to note that, in the before mentioned studies, the dimensions of the problem at hand were restricted, for example the number of samples ranged from a hundred to around 2000, while the number of SNPs ranged from 200 up to 100000, but in this latter case, the number of samples was extremely small (around a hundred), such was due to the big

computational demands of the algorithm for large datasets.

It was not until recent years, that RF were able to be applied to full datasets, i.e. the same dataset to which univariate techniques are applied. For instance (Goldstein *et al.*, 2010) apply random forest to a multiple sclerosis (MS) case-control dataset ($n \cong 3000$, $p \cong 300000$), and show that such method can provide results that can be compatible with previous studies based on univariate rankings, yet they are able to detect new SNPs associated with the outcome of interest.

Data from GWAS studies usually show common issues, these are shown below along with the benefits ensemble methods provide:

1. **Multicollinearity**
   SNPs are mostly independent, but SNPs close to each other may be strongly correlated. Ensemble learning methods are usually able to build good predictors even when multicollinearity is present (Liakhovitski *et al.*, 2010).

2. **Large number of variables (hundreds of thousands) compared to that of observations (some thousands)**.
   Ensemble learning methods have emerged as a tool to analyze high-dimensional biomedical data, where this problem is quite common (Díaz-Uriarte and De Andres, 2006).

3. **Large number of observed covariates, but small number of informative covariates**.
   Having to deal with a great amount of variables requires some kind of preselection: the huge number of predictors can be usually tackled through some dimension reduction technique like the combinatorial partitioning method (Nelson *et al.*, 2001), restricted partition method (Culverhouse *et al.*, 2004), set association (Wille *et al.*, 2003), multifactor dimensionality reduction (Hahn *et al.*, 2003) or a variable screening procedure (Culverhouse *et al.*, 2004; Fan and Lv, 2008). Ensemble methods have shown stable performance and insensitivity to uninformative predictors, nevertheless, in order to provide more accurate results, a modification of the classic RF algorithm was proposed by Amaratunga *et al.* (2008),by the name *Enriched Random Forest*,

i.e. a random forest where the contribution of trees whose nodes are populated by non-informative features is reduced. It is well known that the reduction of noisy variables can convey great improvements in the predictive ability. For instance, since Random Forests has been demonstrated to be robust if the signal-to-noise ratio is not extremely small, a prefiltering procedure may reduce the impact of the non informative predictors by reducing the search space.

4. **Necessity to select relevant genes**.
   Usual approaches use univariate rankings of gene relevance and arbitrary thresholds to identify relevant genes (Hindorff *et al.*, 2009): this approach clearly lacks a global view. Variable selection performed by an ensemble method is achieved according to multivariate rankings, i.e. the ranking of a gene is performed according to the whole set of genes.

5. **Gene-gene, gene-environment interactions and complex relationships**
   These algorithms allow dropping usual linear assumptions and are particularly suited to discover higher order and non-linear effects, but they often referred to as *black boxes* - and therefore criticized - in that their outcome may be difficult to understand. In order to help interpret their results, we think a *rule extraction* method may come in handy, which is a technique meant for improving the interpretability of these methods. These techniques can either build a transparent model (Barakat and Diederich, 2004; Johansson *et al.*, 2010), thus easily interpreted, or simplify the interpretability of an ensemble method by selecting just a few nodes from a large initial large number of nodes (Meinshausen, 2010), which is a desirable outcome in the biological and medical field.

6. **High computational burden due to a large number of covariates.**
   The packages that implement ensemble algorithms are usually unable to handle high-dimensional data, or not optimized for large datasets. The Random Jungle package, on the other hand, was specifically created for the fast analysis of GWAS data and will be extremely useful in our work, specially in the SNP screening phase (Schwarz *et al.*, 2010).

7. **Variables can be continuous, categorical (ordinal and nominal) and binary.**
   Ensemble algorithms are meant for dealing with all these types of variables.

8. **Clustered observations.**
   Studies can be based on related individuals, i.e. observations are not independent. This can be solved running a mixed linear model, and taking the residuals of that model as the new outcome. This is further analyzed later on.

1.2 Definitions

In machine learning, there exists a use of terms that may differ from those commonly used in statistics. Therefore, some of these are summarized in table 1.1.

Table 1.1: Machine Learning Terms

term	synonyms
dependent variable	decision attribute, outcome, response, target (variable)
independent variables	predictors, attributes, features, predictor variables
dataset	(information) system

In this section, some definitions can be found, that can be useful to understand some biological terms used in the present work (King *et al.*, 2006; Moore and Dunlap, 2010).

Character/Trait/Phenotype.
Any detectable phenotypic property of an organism; synonymous with phenotype, trait. The observable characteristics of a cell or an organism, such as its size and shape, its metabolic functions, and its behavior. The genotype is the underlying basis of the phenotype, and the term is commonly used to describe the effect a particular gene produces, in comparison with its mutant alleles. Some genes control the behavior of the organism, which in turn generates an artefact outside the body.

Complex disease.
A complex disease is caused by a combination of genetic, environmental, and lifestyle factors, most of which have not yet been identified.

Gene.
The definition of a gene changes as more of its properties are revealed. In the classical literature it is defined as a hereditary unit that occupies a specific position (locus) within the genome or chromosome; a unit that has one or more specific effects upon the phenotype of the organism; and a unit that recombines with other such units.

Genetic Architecture.
It refers to the complete genetic basis, which includes various types of genetic effects, underlying a given trait.

GWAS.
A genome-wide association study is an examination of the association between genetic variation across a whole genome and a given trait.

Locus (plural loci).
A locus is the position that a gene occupies in a chromosome or within a segment of genomic DNA.

Nucleotide.
It is one of the monomeric units from which DNA or RNA polymers are constructed, consisting of a purine or pyrimidine base, a pentose, and a phosphoric acid group. The nucleotides of DNA are deoxyadenylic acid, thymidylic acid, deoxyguanilic acid, and deoxycytidylic acid. The corresponding nucleotides of RNA are adenylic acid, uridylic acid, guanylic acid, and cytidylic acid.

Single-nucleotide polymorphisms.

Single-nucleotide polymorphisms (SNPs) are small variations in DNA sequence in which at any given position a single nucleotide is replaced by one of the other three nucleotides.

1.3 Model Selection

Statistical learning plays an important role in many areas of science when we need to learn from data, specially in the fields of statistics, data mining, and artificial intelligence (Friedman *et al.*, 2001). Due to the huge improvements both in data collection and in computational performances, in the last few decades we have seen many innovative techniques used to solve large-scale data problems, such as kernel methods, neural networks, support vector machines and ensemble methods. In such scenario, one is usually interested in learning from data as to be able to build models which will prove useful for the prediction of future outcomes. In the next pages we follow the work of Guyon *et al.* (2010) to present a unified approach to model selection.

The problem of statistical learning is often decomposed into the tasks of fitting parameters to some training data, and then choosing the best model according to some criteria, which is known as model selection. By this name, one usually refers to a set of techniques used to select a model, that best explains the data at hand or best predicts future data. From a purely statistical point of view, we are concerned with supervised learning. The goal of supervised learning is to predict a target variable y from a domain $\mathcal{Y}$, which may be either continuous or categorical. The former case yields a regression problem, while the latter yields a classification problem. The predictions are produced using a p-dimensional variable $\mathbf{x}$ from a domain $\mathcal{X}$, and the data pairs $(\mathbf{x}, y)$, independent and identically distributed, are drawn from an unknown, but fixed, probability distribution $P(\mathbf{x}, y)$. A number of n pairs are drawn from that distribution and form the training data $D = \{\mathbf{x}_i, y_i\}_1^n$, while $\mathbf{X} = [x_{ij}], i = 1, \ldots n, j = 1, \ldots p$ is a $n \times p$-dimensional design matrix whose rows are the training observations and whose columns are the features. Finally, we denote by $\mathbf{y}$ the a $n \times 1$-dimensional column

vector containing the target values y_i. The supervised learning problem can be formulated as follows (Guyon *et al.*, 2010):

- **Function approximation (induction) methods** seek a function f (called model or learning machine) belonging to a class of models $\mathcal{F}$, which closely matches a target function, i.e. minimizes a specified risk function or maximizes some utility function. The goal is to minimize an expected risk $R(f) = \int \mathcal{L}(f(\mathbf{x}), y)\, dP(\mathbf{x}, y)$ also called generalization error, where $\mathcal{L}(f(\mathbf{x}), y)$ is a loss function measuring the discrepancy between $f(\mathbf{x})$ and y. Since $P(\mathbf{x}, y)$ is unknown, only estimates of $R(f)$ can be computed, called *evaluation functions* or *estimators*.

- **Bayesian and ensemble methods** make predictions according to model averages that are convex combinations of models $f \in \mathcal{F}$, i.e. that belong to the convex closure of the model class $\mathcal{F}^*$. Bayesian methods approximate $E_f(y|\mathbf{x}) = \int_{f \in F} f(\mathbf{x})\, dP(f)$, an expectation taken over a class of models $\mathcal{F}$, using an unknown probability distribution $P(f)$ over the models; starting from a *prior* distribution which is translated into a *posterior* once data have been taken into account. Ensemble methods approximate $E_D(f(\mathbf{x}, D))$, where $f(\mathbf{x}, D)$ is a function from the model class $\mathcal{F}$, trained with m examples and $E_D(\cdot)$ is the mathematical expectation over all training sets of size m. The idea behind ensembles is to generate a variety of functions, providing different perspectives over the problem at hand, then creating a consensus through the ensemble itself.

A single model can be parametrized as $f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta})$, where $\boldsymbol{\alpha}$ are the parameters, while $\boldsymbol{\theta}$ are the hyper-parameters of the model (Guyon *et al.*, 2010). The latter can include: indicators of the presence/absence of features, choice of pre/postprocessing, choice of the algorithm (e.g. linear models, neural networks, kernel methods, etc.), algorithm parameters (e.g. number of layers in a neural network). The tuning of model parameters is referred to as the *first level of inference*: data are split in several subsets of size m_{tr} for the purpose of training the models. The *validation sample* of size m_{va} is that part of the data used to adjust the hyper-parameters $\boldsymbol{\theta}$ at the *second level of inference*. Lastly, m_{te} is the number of samples used to evaluate the

final model. Thus the empirical estimates of the expected risk are called: training error ($R_{tr}(f)$), validation error ($R_{va}(f)$), and test error ($R_{te}(f)$).

There is a clear connection between the problem of model selection and that of performance prediction. Performance prediction is the problem of estimating the expected risk or generalization error $R(f)$. Model selection is the problem of adjusting the capacity of the models to the training data available in order to avoid either under-fitting or over-fitting. Solving the performance prediction problem would also solve the model selection problem, but model selection is an easier problem. What is common among the various views of model selection is the idea of multiple levels of inference, each level corresponding to one set of parameters or hyper-parameters.

Consider a two-level case for a model class $f(\mathbf{x};\boldsymbol{\alpha},\boldsymbol{\theta})$ parameterized by one set of parameters and one set of hyperparameters. From a frequentist point of view, one creates a hierarchy of optimization problems:

$$f^{**} = \text{argmin}_{\theta} R_2(f^*, D), \text{ such that } f^* = \text{argmin}_{\alpha} R_1(f, D) \tag{1.1}$$

More generally, we define a multi-level inference problem as a learning problem organized into a hierarchy of learning problems. Formally, consider a machine learning toolkit which includes a choice of learning machines $\mathcal{A}[\mathcal{B}, R]$ where $\mathcal{B}$ is a model space of functions $f(\mathbf{x};\boldsymbol{\theta})$ of parameters $\boldsymbol{\theta}$ and R is an evaluation function (e.g. a risk function). $\mathcal{A}[\mathcal{B}, R]$ can be thought of not as a procedure but as an *object*, as intended in the object oriented programming context, equipped with a method *train*, which processes data according to a training algorithm

$$f^{**} = \text{train}(\mathcal{A}[\mathcal{B}, R_2], D) \tag{1.2}$$

This framework embodies the second level of inference of equation (1.1). The solution f^{**} belongs to $\mathcal{B}^*$, the convex closure of $\mathcal{B}$. To implement the first level of inference, $\mathcal{B}$ can be considered as a learning machine itself - and not just a model space. Its model space $\mathcal{F}$ includes functions $f(\boldsymbol{x};\boldsymbol{\alpha},\boldsymbol{\theta})$ of variable parameters $\boldsymbol{\alpha}$($\boldsymbol{\theta}$ is fixed), which are adjusted by the train method of $\mathcal{B}$:

$$f^* = \text{train}(\mathcal{B}[\mathcal{F}, R_1], D) \tag{1.3}$$

The solution f^* belongs to $\mathcal{F}^*$, the convex closure of $\mathcal{F}$. The method train of $\mathcal{A}$ should call the method train of $\mathcal{B}$ as a subroutine, because of the nested nature of the learning problem of equation (1.1). Note that for fixed values of $\boldsymbol{\theta}$, the problem of learning a can be formulated as a convex optimization problem, with a single unique solution, for which powerful mathematical programming packages are available, while the overall optimization of $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ is nonconvex.

After assessing the importance of model selection inside a statistical learning perspective, it is of interest to outline that ensemble methods (e.g. random forest, and boosted trees) can be seen as a way of avoiding such model selection by averaging among models rather than choosing a single model. On the one hand, model selection algorithms are the best choice in application where model simplicity and insight of the phenomenon are the aim of the analysis, on the other hand, ensembles - especially after the latest computational power growth - have proved to be cutting edge solution for prediction.

1.4 (Variable) Relevance

As stated above, model selection algorithms are chosen when interpretability is a key aspect to be taken into account, and can be agreed on that they should be used whenever possible, especially when the number of features is small. The extensive enhancement in data storage has allowed a dramatic increase in the number of features that can be included, resulting in the great dilemma *interpretability vs. performance*. Therefore, there has been a large increase in the study of *relevant features*, particularly in the data mining and machine learning areas for feature subset selection. Such subset should contain features good enough to describe the training data as well as to predict future cases. Following Bell and Wang (2000), we provide some formalizations of the concept of relevance.

Gärdenfors (1978) states that - when judging the probability of a statement r, the relevance relation is defined with the aid of a given probability measure P as: p is relevant to r on prior evidence e iff $P(r|p\&e) \neq P(r|e)$,

while p is irrelevant to r on evidence e iff $P(r|p\&e) = P(r|e)$. This means that if the probability of statement r is changed by adding p to e, p is said to be relevant on the evidence e - otherwise it is declared irrelevant. Moreover, relevance in this sense can be divided into both *positive* and *negative* relevance, where p is positively relevant to r on the evidence e $P(r|p\&e) > P(r|e)$ and negatively relevant if $P(r|p\&e) < P(r|e)$.

According to the work of Pearl (1988), irrelevance can be identified with conditional independence, while relevance comes from the negation of irrelevance. The properties of conditional independence are the following:

Let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ be three disjoint sets of variables. If $\mathfrak{F}(\mathcal{X}, \mathcal{Z}, \mathcal{Y})$ stands for the relation $\mathcal{X}$ is independent of $\mathcal{Y}$ given $\mathcal{Z}$ in some probability distribution p, then $\mathfrak{F}$ must satisfy the following conditions:

- Symmetry: $\mathfrak{F}(\mathcal{X}, \mathcal{Z}, \mathcal{Y}) \Longleftrightarrow \mathfrak{F}(\mathcal{Y}, \mathcal{Z}, \mathcal{X})$;
- Decomposition: $\mathfrak{F}(\mathcal{Y}, \mathcal{Z}, \mathcal{X} \cup \mathcal{W}) \Rightarrow \mathfrak{F}(\mathcal{X}, \mathcal{Z}, \mathcal{Y})$ and $\mathfrak{F}(\mathcal{X}, \mathcal{Z}, \mathcal{W})$;
- Weak Union: $\mathfrak{F}(\mathcal{Y}, \mathcal{Z}, \mathcal{X} \cup \mathcal{W}) \Rightarrow \mathfrak{F}(\mathcal{Z}, \mathcal{X} \cup \mathcal{W}, \mathcal{Y})$;
- Contraction: $\mathfrak{F}(\mathcal{X}, \mathcal{Z}, \mathcal{Y})$ and $\mathfrak{F}(\mathcal{X}, \mathcal{Z} \cup \mathcal{Y}, \mathcal{W}) \Rightarrow \mathfrak{F}(\mathcal{X}, \mathcal{Z}, \mathcal{Y} \cup \mathcal{W})$.

The main contribution in the definition of relevance within a machine learning context is due to John *et al.* (1994) and Kohavi (1994). Their main definition differentiates between strong and weak relevance. Let $\mathcal{S}_i$ be the set of all features except X_i, i.e. $\mathcal{S}_i = \{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m\}$, and let Y be a decision attribute not in $\mathcal{S}_i$. Denote by s_i a value-assignment to all features $\mathcal{S}_i$, then X_i is strongly relevant to Y iff there exist some x_i, y, and s_i with $P(X_i = x_i, S_i = s_i) > 0$ such that

$$P(Y = y|S_i = s_i, X_i = x_i) \neq P(Y = y|S_i = s_i). \tag{1.4}$$

A feature X_i is weakly relevant to Y iff it is not strongly relevant, and there exists a subset of features $S_i^{'}$ of S_i for which there exists some x_i, y, and $s_i^{'}$ with $P\left(X_i = x_i, S_i^{'} = s_i^{'}\right) > 0$ such that

$$P\left(Y = y|X_i = x_i, S_i^{'} = s_i^{'}\right) \neq P\left(Y = y|S_i^{'} = s_i^{'}\right). \tag{1.5}$$

These definitions imply that X_i is strongly relevant if the probability of the outcome Y (given all features) can change if the knowledge about the value of X_i is eliminated, and thus, it cannot be removed without performance degradation of the predicting algorithm. On the other hand, X_i is weakly relevant if it is not strongly relevant, but can sometimes contribute to prediction accuracy.

All the above definitions provide a more qualitative definition of relevance. In the following lines, we provide a more quantitative definition of relevance which complies with the commonsense meaning that is usable for the purpose of variable selection.

According to Bell and Wang (2000), the relevance of a variable with respect to another (target) variable one is defined in terms of information theory as the mutual information between the two variables relative to the entropy of the target variable, i.e. the relative reduction of uncertainty of one variable due to the knowledge of another.

First, let us recall two basic notions of information theory: entropy and mutual information (Cover and Thomas, 2006). The entropy of a random variable X with a probability mass function $p(x)$ is defined by

$$H(X) = -\sum_{x} p(x) \log_2 p(x) \tag{1.6}$$

The entropy is a measure of the average uncertainty in the random variable. While entropy is the uncertainty of a single random variable, conditional entropy $H(X|Y)$ is the entropy of a random variable conditional on the knowledge of another random variable. The reduction in uncertainty due to another random variable is called the mutual information.

For two random variables X and Y the mutual information is defined

as:

$$I(X,Y) = H(X) - H(X|Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)}. \tag{1.7}$$

Such quantity measures the dependence between the two random variables. It is symmetric in X and Y, always non-negative, and is equal to zero iff X and Y are independent.

More generally, given three sets of variables $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ (where a single variable X, Y, or Z is regarded as a singleton set of variables) with a joint probability distribution p, let $I(\mathcal{X};\mathcal{Y},|\mathcal{Z})$ be the mutual information between $\mathcal{X}$ and $\mathcal{Y}$ given $\mathcal{Z}$, and let $H(\mathcal{X}|\mathcal{Y})$ be the entropy of of $\mathcal{X}$ given $\mathcal{Y}$. If $H(\mathcal{Y}|\mathcal{Z}) \neq 0$, then the variable relevance of $\mathcal{X}$ to $\mathcal{Y}$ given $\mathcal{Z}$, denoted $r_p(\mathcal{X},\mathcal{Y}|\mathcal{Z})$, is defined as

$$r_p(\mathcal{X},\mathcal{Y}|\mathcal{Z}) = \frac{I(\mathcal{X},\mathcal{Y}|\mathcal{Z})}{H(\mathcal{Y}|\mathcal{Z})} = \frac{H(\mathcal{Y}|\mathcal{Z}) - H(\mathcal{Y}|\mathcal{X},\mathcal{Z})}{H(\mathcal{Y}|\mathcal{Z})} \tag{1.8}$$

If $H(\mathcal{Y}|\mathcal{Z}) = 0$, then $r_p(\mathcal{X},\mathcal{Y}|\mathcal{Z}) = 0$.

In order to define the problem of Feature Subset Selection (FSS) in the context of machine learning, we remind that $\mathcal{X} = \{X_1, X_2, \ldots, X_p\}$ is the set of features and Y the decision attribute. Given a dataset $D(\mathcal{X},Y)$, the learning task of the algorithm is to deduce the (unknown) structure between $\mathcal{X}$ and Y so that this relation can be used to predict future cases. This relationship can be referred to as *learning information*, whose natural measure is the mutual information, defined as $I(\mathcal{X},Y)$ (Cover and Thomas, 2006). A good feature subset should retain the learning information contained in the dataset.

For a given feature subset $\Pi \subseteq \mathcal{X}$, if $I(\Pi,Y) = I(\mathcal{X},\mathcal{Y})$, Π is said to preserve the learning information and is defined a *sufficient feature subset* (SFS). Therefore, removing all other features $\Sigma = \mathcal{X} \setminus \Pi$ will not result in a loss of learning information, i.e. Y is conditionally independent of Σ given Π.

Given a specific dataset, there may be more than one SFS, although they

may not be equally good for prediction purposes. To solve this multiplicity problem, we can introduce the empirical principle known as Occam's razor, also known as the *principle of parsimony*. In machine learning, it can be formulated as: given two models that are both consistent with the training data, the simpler one should perform better on future examples. In other words, Occam's razor is a way to build (hopefully) optimal generalizers (Wolpert, 1990). In order to define model simplicity, one can take advantage of some measure of simplicity (or complexity), e.g. the number of neurons in a neural network, the number of leaf nodes in a CART, or the number of degrees of freedom in a linear model. The above mentioned measures are all model dependent, and change depending on the algorithm used. In order to achieve a generalized definition of FSS that is not model-dependent, Bell and Wang (2000) suggest choosing Shannon's entropy measure as a simplicity measure in the sense of the encoding length measure seen in (Schweitzer, 1995). Therefore, $\Pi \subseteq \mathcal{X}$ is an occam-optimal feature subset iff Π is an SFS and there is no Σ such that $H(\Sigma, Y) < H(\Pi, Y)$, i.e. Π is an SFS that minimizes the joint entropy of the features and the decision attribute that at the same time preserves mutual information. Such conditions can be restated in terms of relevance as $I(\Pi, Y) = I(\mathcal{X}, Y) \iff r(\Pi, Y) = r(\mathcal{X}, Y)$ for any $\Pi \subseteq \mathcal{X}$, so any Π which preserves learning information in fact also maximizes the relevance $r(\mathcal{X}, Y)$. Let Π and Σ be SFSs. By definition, we have $I(\Pi, Y) = I(\Sigma, Y) = I(\mathcal{X}, \mathcal{Y})$, and then

$$H(\Pi; Y) \leq H(\Sigma; Y) \iff H(\Pi) \leq H(\Sigma) \iff \frac{I(\Pi; Y)}{H(\Pi)} \geq \frac{I(\Sigma; Y)}{H(\Sigma)} \quad (1.9)$$

Therefore, an occam-optimal FS Π would be the SFS which maximizes the relevance $r(\mathcal{X}, Y)$. An important question is that an occam-optimal FS may not be unique, due to fact that redundant features - as measured by the average inter-correlation between features - can act as surrogates and replace each other in a feature subset. Moreover, finding this optimal FS is a typical NP-hard problem, hence one must turn to some heuristic algorithm that can provide (hopefully) quasi-optimal solutions.

In real life cases, it is usually desirable to obtain the smallest feature subset that gives the best prediction. Another question arises if we would like to understand the underlying mechanism of the phenomenon at hand - as it

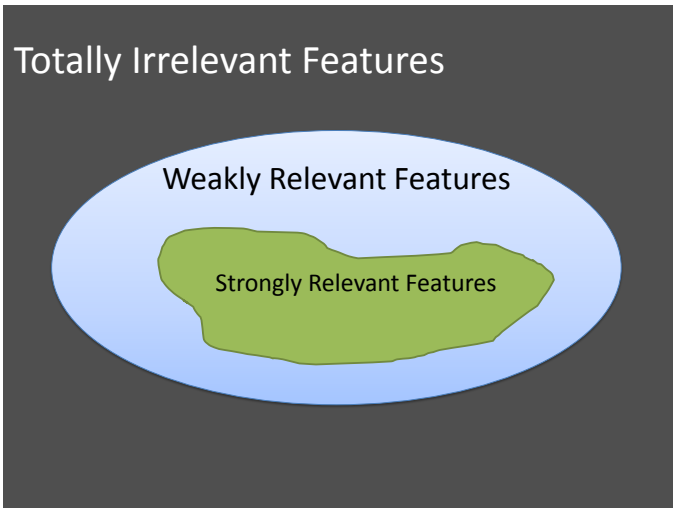

Figure 1.1: A view of feature set relevance

is in the case of genetic studies, where one wants to discover all possible SNPs that map to genes which are related to the outcome of interest (typically a disease). For the above reason, it can be useful to group together weakly (redundant and non-redundant) and strongly relevant features, yielding a so-called *all-relevant problem*, thus, focusing on the removal of irrelevant ones.

1.5 Feature-Selection Algorithms

Algorithms that deal with datasets containing large numbers of irrelevant attributes are generally referred to as feature-selection Algorithms. These can be grouped into three classes: filters, wrappers, and embedded methods (Kohavi and John, 1997).

1.5.1 Filters

The simplest type of feature selection algorithms are filters. These methods include a broad class of techniques that aim at the reduction of the model space $\mathcal{F}$ before training the learning machine, and can be seen as some kind of preprocessing step. Some of the methods that are encompassed in this class can be Wilcoxon tests, t-tests, ANOVA methods, χ^2-tests, Spearman's rank correlation coefficient, Pearson's correlation coefficient, and mutual information measures.

This filtering can be thought both on a univariate scale as well on a multivariate scale. In this latter case, one can include methods like the FOCUS algorithm (Almuallim and Dietterich, 1991), which exhaustively examines all the subsets of features and selects the minimal one that is sufficient to distinguish between two classes in the training data, or the Relief algorithm (Kira and Rendell, 1992), which uses an approach based on the K-nearest-neighbor algorithm to assign an importance measure to a feature from randomly sampled subsets of the training set.

As for other filtering approaches, one could perform some dimensionality reduction, such as Principal Components Analysis or a Clustering Method, in order to select some variables to feed the inductive algorithm.

This approach works well in domains where there is little interaction among relevant attributes, in the opposite case such method can lead a relevant feature in isolation to look no more useful that an irrelevant one Blum and Langley (1997). The main drawback of this approach is that it ignores the effect of the selected feature subset on the performance of the induction algorithm whatsoever, in the sense that the selection process and the induction algorithm are performed on two different levels.

1.5.2 Wrappers

Wrapper methods consider learning machines as black boxes capable of internally tuning their parameters $\boldsymbol{\alpha}$ given some data D and some hyperparameters $\boldsymbol{\theta}$. Wrappers use either a search algorithm or a sampling algorithm (e.g. Monte-Carlo Markov Chain methods) to explore the hyperparameter space and an evaluation function or a posterior probability to assess the performance of the trained machine, and finally select either one single (best) model or create an ensemble to make predictions. In this case, feature selection algorithm performs a search for a good subset using the induction algorithm itself as a part of the evaluation function. For frequentist approaches, the most commonly used evaluation function is the cross-validation estimator (for bagging methods, the bootstrap estimator is a natural choice). Two of the most famous wrappers are the sequential forward selection and backward elimination. The former begins the search with the empty set of features, while the latter begins with the full set of features.

The main (and straight) drawback of wrappers over filters is the former's

computational demands. But this approach has to be preferred over the former, in case we are dealing with features that interact to some extent.

1.5.3 Embedded Methods

Embedded methods are similar to wrappers in the sense that they need an evaluation function and a search strategy to explore the hyper-parameter space. In contrast to filter and wrappers, embedded methods do not separate the learning from the feature selection process.

These methods have been drawing some attention in the machine learning field lately. Among these, one can include the LASSO (Tibshirani, 1996), or the elastic net (Zou and Hastie, 2005). Other examples include decision trees, which select relevant features using top-down, hierarchical partitioning schemes, where the output is a model that uses only a subset of features, i.e. those that appear in the nodes of the tree. Among these, we find CART (Breiman *et al.*, 1984), CHAID (Kass, 1980), and C4.5 (Quinlan, 1993).

When the underlying phenomenon at hand presents interactions between features, it is strongly advised to avoid filters. Therefore, all the methods used in this work as variable selection tools can be encompassed in the definition of wrappers around the random forest function, which uses the CART recursive partitioning function as base learner. Such methods are described later on.

Chapter 2

Ensemble Generation

In the following chapter, we present an algorithm for the generic ensemble generation as proposed in Friedman *et al.* (2003), which provides a general framework in which one can include classic ensemble methods like bagging, random forests, and gradient boosting.

From previous section we recall that the purpose of supervised learning is to predict the values of a response variable y using the known joint values of a covariate set $\mathbf{x} = (x_1, x_2, \ldots, x_p)$. Predictions take the form $\hat{y} = F(\mathbf{x})$, where $F(\mathbf{x})$ is a function that maps the input variables $\mathbf{x}$ to a value of the output variable y. The purpose of supervised learning is to produce an accurate mapping, and lack of accuracy is defined by the prediction risk

$$R(F) = E_{\mathbf{x}y} L(y, F(\mathbf{x})) \tag{2.1}$$

where $L(y, \hat{y})$ is a loss function that characterizes the cost of predicting a value $\hat{y}$ when the true value is y, and the expected value is over the joint distribution $(\mathbf{x}, y)$ of all variables for the data to be predicted. In this case, the optimal mapping function is the one that minimizes the prediction risk

$$F^*(\mathbf{x}) = \underset{F(\mathbf{x})}{\arg\min} E_{\mathbf{x}y} L(y, F(\mathbf{x})) \tag{2.2}$$

In the supervised learning case, one is given a collection of observed

cases $\{\mathbf{x}_i, y_i\}_1^N$, where for each observation the values of all the variables have been determined. Thus, an approximation $F(\mathbf{x})$ to $F^*(\mathbf{x})$ is obtained by applying a learning method to the training data, which are often regarded as a random sample of some probability distribution.

Ensemble methods have the following structural form

$$F(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m f_m(\mathbf{x}) \tag{2.3}$$

where M is the size of the ensemble and $f_m(\mathbf{x})$ is a different function of the input variables $\mathbf{x}$ derived from the training data, called base learner. The prediction provided by the ensemble is a linear combination of the predictions given by each base learner, with $\{a_m\}_0^M$ being the parameters that define the particular linear combination. Ensembles differ in the choice of particular base learners, how they are derived from the data, and how the parameters $\{a_m\}_1^M$ are obtained. Popular base learners include, for example, multivariate spline functions (Friedman, 1991), where the parameters are the knots on the corresponding variables, and CARTs (Breiman *et al.*, 1984), where the parameters are the splitting variables, the values defining the partition of the covariate space, and the values assigned to the terminal nodes. Given a set of base learners $\{f_m(\mathbf{x})\}_1^M$, the parameters of the linear combination are obtained by a regularized linear regression on the training data

$$\{\hat{a}_m\}_0^M = \arg\min_{\{a_m\}_0^M} \sum_{i=1}^{N} L\left(y_i, a_0 + \sum_{m=1}^{M} a_m f_m(\mathbf{x}_i)\right) + \lambda \cdot \sum_{m=1}^{M} |a_m|. \tag{2.4}$$

The first term in (2.4) measures the prediction risk on the training data, while the second is a regularization term that penalizes the coefficients of the base learners. The amount of regularization is given by $\lambda \geq 0$. Usually larger values of λ produce more overall shrinkage, with a larger number of parameters being set to zero (Tibshirani, 1996). The value of λ should be the one minimizing the future prediction risk based on future samples, which is usually determining via cross-validation. The base learners $\{f_m(\mathbf{x})\}_1^M$ are randomly generated by means of the perturbation sampling technique accurately described in (Friedman *et al.*, 2003) and presented later in this

chapter, and are taken to be a simple function of the covariate set $\mathbf{x}$ indexed by a set of parameters $\{\mathbf{p} = p_1, p_2, \ldots\}$, as

$$f_m(\mathbf{x}) = f(\mathbf{x}; \mathbf{p}_m) \tag{2.5}$$

where $\mathbf{p}_m$ represents a set of parameters indexing a specific function $f_m(\mathbf{x})$ from a the parametrized class $f(\mathbf{x}; \mathbf{p}_m)$.

Friedman *et al.* (2003) make a connection to numerical integration, and view the mapping function $F(\mathbf{x})$ in the form

$$F(\mathbf{x}, a) = a_0 + \int_P a(\mathbf{p}) f(\mathbf{x}; \mathbf{p}) \, d\mathbf{p}, \tag{2.6}$$

where $f(\mathbf{x}; \mathbf{p})$ is the single base learner, indexed by a set of parameters $\mathbf{p} = (p_1, p_2, \ldots, p_K)$ and $a(\mathbf{p})$ is its corresponding coefficient in the linear model (2.6).

Because most useful base learners $f(\mathbf{x}; \mathbf{p})$ involve several parameters, the integration problem in (2.6) is high dimensional. To solve this integral, numerical quadrature is carried out as in (2.3) and (2.4). The evaluation points $\mathbf{p} = (p_1, p_2, \ldots, p_M)$ are drawn at random from some probability distribution $r(\mathbf{p})$ defined on the parameter space $p \in P$. One way to do so is to choose $r(\mathbf{p})$ to be constant so that each point is equally likely to be selected at each of the M draws. Thus, equation (2.6) is computed through Monte Carlo integration and can be approximated by the average of the function evaluated at a set of evaluation points $\mathbf{p} = (p_1, p_2, \ldots, p_M)$. On the other hand, it could be useful to recognize that certain values of these evaluation points have more impact on the accuracy of the integral being estimated, and therefore, it should be advisable to sample more frequently around these *important* values.

In Monte Carlo integration, point importance can be measured in single or group fashion. In the former case, the relevance of each point is determined without taking into account other points being evaluated in the quadrature rule, while in the latter, importance is assigned to groups of points together, rather than separately to individually sampled points. The former situation requires pure Monte Carlo methods, while the latter requires quasi Monte Carlo techniques. This typically depends on the type

of ensemble being used: parallel ensembles like Bagging or Random Forest require single importance sampling, while other methods like Gradient Boosting take advantage of group importance sampling.

Given a single evaluation point $\mathbf{p} \in P$, this idea of importance is formally defined by its lack of relevance:

$$J(\mathbf{p}) = \min_{\alpha_0,\alpha} E_{\mathbf{x}y} L(y, \alpha_0 + \alpha f(\mathbf{x};\mathbf{p})) \tag{2.7}$$

If a single base learner were to be selected (e.g. a tree), it would be the global minimizer

$$\mathbf{p}^* = \arg\min_{\mathbf{p}\in P} J(\mathbf{p}) \tag{2.8}$$

This value is unlikely for this optimal single point to produce as accurate results as one involving many evaluation points, but this is the usual praxis, e.g. when a single tree or linear regression model is used. The assumption is that a collection of evaluation points, each having small values of $J(\mathbf{p})$, will produce an integration rule with higher accuracy than that provided by the best single point rule (or even than that of a similar sized collection of evaluation points sampled with constant probability).

As to apply Monte Carlo sampling as stated above, a sampling probability $r(\mathbf{p})$ that gives high probability to points $\mathbf{p} \in P$ providing smaller values of $J(\mathbf{p})$. That is

$$r(\mathbf{p}) = g(J(\mathbf{p})) \tag{2.9}$$

where $g(\cdot)$ is a monotonically decreasing function of its argument. The density $r(\mathbf{p})$ will be centered at or near $\mathbf{p}^*$ and have correspondingly decreasing values for points increasingly distant from $\mathbf{p}^*$. Introducing randomness in the selection of the evaluation points produces less optimal values with $J(\mathbf{p}) \geq J(\mathbf{p})^*$. The relevant distance measure is

$$d(\mathbf{p}, \mathbf{p}^*) = J(\mathbf{p}) - J(\mathbf{p}^*). \tag{2.10}$$

The sampling distribution is characterized by a scale parameter defined as

$$\sigma = \int_P d(\mathbf{p}, \mathbf{p}^*)\, r(\mathbf{p})\, d\mathbf{p}. \tag{2.11}$$

A σ with a large value means a large spread of the base learners $f(\mathbf{x}; \mathbf{p})$ with many possible irrelevant cases, resulting in a decrease in the accuracy of the integration rule. Very large values can be assimilated to the naive case of Monte Carlo sampling with constant probability.
A σ with a small value means that each sampled point provides little information added to that provided by other nearby points, and the base learners $f(\mathbf{x}; \mathbf{p})$ perform quite similarly, while the extreme case of $\sigma = 0$ indicates that the integration rule provides the same accuracy as the best single point. The value of σ in real life cases depends on M and varies depending on the joint distribution of $(\mathbf{x}_i, y_i)_1^N$ and the choice of the base learners $f(\mathbf{x}; \mathbf{p})$. (Breiman, 2001) noticed that the performance of random forest depended empirically on the strength of the individual trees in the forest and the correlation between them. This concept stands for every ensemble, and is connected with the width of the sampling distribution $r(\mathbf{p})$. The strength of a base learner $f(\mathbf{x}; \mathbf{p})$ is directly related to its partial importance, i.e. inversely related to $J(\mathbf{p})$.

1. An ensemble composed of strong base learners $\{f(\mathbf{x}; \mathbf{p})\}_1^M$ all having $J(\mathbf{p}_m) \simeq J(\mathbf{p}^*)$ is known to perform poorly, because the sampling distribution with a small width σ leads to an ensemble of base learners all having a similar strength to the strongest one $f(\mathbf{x}; \mathbf{p}^*)$, providing similarly highly correlated predictions;

2. An ensemble consisting of very weak base learners $J(\mathbf{p}_m) \gg J(\mathbf{p}^*)$ is also known to perform quite poorly, because its corresponding sampling distribution with large width produces highly diverse base learners, most of which have large values of $J(\mathbf{p})$ and less correlated predictions.

3. The best results come from an ensemble of moderately strong base learners whose predictions are not too correlated, which corresponds to finding a good correlation-strength trade-off for the ensemble mem-

bers. This can be achieved by seeking an appropriate width for an importance sampling distribution.

2.1 Perturbation Sampling Methods

Because finding a suitable sampling distribution $r(\mathbf{p})$ on the parameter space $\mathbf{p} \in P$ may be hard, the process can be indirectly approximated by repeatedly modifying or perturbing some aspect of the problem at hand in a random way. The magnitude of these perturbations then affects the width σ of the corresponding sampling distribution $r(\mathbf{p})$. This heuristic used to simulate the process of sampling from $r(\mathbf{p})$ is called perturbation sampling. The aspects of the problem that can undertake a perturbation are the following:

1. *Perturbations of the loss function.*
   A first choice can be the repeated perturbation of the loss criterion by

$$L_m(y, \hat{y}) = L(y, \hat{y}) + \eta \cdot l_m(y, \hat{y}) \tag{2.12}$$

   where $l_m(y, \hat{y})$ is a different randomly constructed function of its arguments. The sample points are obtained by

$$\left\{ \mathbf{p}_m = \underset{\alpha_0, \alpha, \mathbf{p} \in P}{\arg\min} E_{\mathbf{x}y} L_m(y, \alpha_0 + \alpha f(\mathbf{x}; \mathbf{p})) \right\}_1^M . \tag{2.13}$$

2. *Perturbations of the argument of the loss function.*
   An option over the perturbation of the loss criterion is the modification of the argument of such function, for example taking

$$\left\{ \mathbf{p}_m = \underset{\alpha_0, \alpha, \mathbf{p} \in P}{\arg\min} E_{\mathbf{x}y} L(y, \alpha_0 + \alpha f(\mathbf{x}; \mathbf{p}) + \eta g_m(\mathbf{x})) \right\}_1^M \tag{2.14}$$

   where $g_m(\mathbf{x})$ is a different randomly constructed function of $\mathbf{x}$. This is for instance the approach used in Gradient Boosting (Friedman, 2001).

In these two cases, the expected size of the perturbation is induced by the value of η, which controls the width of the corresponding sampling distribution.

3. *Perturbations of the joint distribution of variables.*
   Another approach is that of modifying the joint distribution of variables by random reweighing

$$\{\mathbf{x}, y\}_m = \{\mathbf{x}, y\} \cdot [w_m \{\mathbf{x}, y\}]^\eta \tag{2.15}$$

   where $w_m \{\mathbf{x}, y\}$ is a different randomly constructed (weighting) function of $\{\mathbf{x}, y\}$, as it happens to be in the Adaboost algorithm (Freund and Schapire, 1995). Thus, the evaluation points $\{\mathbf{p}_m\}_1^M$ are the solution to

$$\left\{\mathbf{p}_m = \underset{\alpha_0, \alpha, \mathbf{p} \in P}{\arg\min} E_{\{\mathbf{x}y\}_m} L\left(y, \alpha_0 + \alpha f\left(\mathbf{x}; \mathbf{p}\right)\right)\right\}_1^M. \tag{2.16}$$

   As in the two previous cases, the expected size of the perturbation is induced by the value of η, which controls the width of the corresponding sampling distribution.

4. *Perturbations of the search algorithm.*
   One other option is to randomly modify the search algorithm used to solve $\mathbf{p}^* = \underset{\mathbf{p} \in P}{\arg\min} J\left(\mathbf{p}\right)$.
   Examples of this approach can be found for instance in randomized trees (Dietterich, 2000), where the optimal spit in each node is chosen uniformly at random among k best splits, as well as in the random forest algorithm, where the optimal split is chosen among a set of randomly chosen subset of predictor variables or in the Extra-Trees algorithm of (Geurts *et al.*, 2006) that splits nodes by choosing cut-points fully at random, instead of choosing the best split for each predictor variable as in the random forest algorithm. For instance, for randomized trees, the width σ of the sampling distribution $r\left(\mathbf{p}\right)$ depends on the value of k, while for random forests, the size of the subsets inversely controls the width of the sampling distribution.

The generic ensemble generation algorithm (see Algorithm 1) starts the ensemble F_0 with some constant (line 1), it could be zero or another suitable

Algorithm 1: Generic Ensemble Generation

1 $F_0(\mathbf{x}) = \arg\min_{\alpha} \sum_{i=1}^{N} L(y_i, \alpha)$;
2 **for** $m = 1$ *to* M **do**
3 $\quad \mathbf{p}_m = \arg\min_{\mathbf{p}} E_{i \in S_m(\eta)} L(y_i, F_{m-1}(\mathbf{x}) + f(\mathbf{x}_i; \mathbf{p}))$;
4 $\quad f_m(\mathbf{x}) = f_m(\mathbf{x}; \mathbf{p}_m)$;
5 $\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot f_m(\mathbf{x})$;
6 Ensemble $= \{f_m(\mathbf{x})\}_1^M$.

constant, then at each iteration a new base learner is created and added to the collection (line 3). $S_m(\eta)$ represents a different subsample of data of size $\eta < N$ randomly drawn at random without replacement from the original data $S_m(\eta) \subset \{\mathbf{x}_i, y_i\}_1^N$, with small values of η creating more diverse base learners $\{f_m(\mathbf{x})\}_1^M$ as well as reducing computational times. At each step, the memory function

$$F_{m-1}(\mathbf{x}) = F_0(\mathbf{x}) + \nu \cdot \sum_{k=1}^{m-1} f_k(\mathbf{x}) \tag{2.17}$$

contains partial information about the previously built base learners $\{f_k(\mathbf{x})\}_1^{m-1}$, and ν is a shrinkage parameter $0 \leq \nu \leq 1$. Setting $\nu = 0$ creates a parallel ensemble, because it causes each base learner to be generated independently of previously generated base learners, while $\nu = 1$ maximizes their influence. Thus, setting ν to intermediate values $0 < \nu < 1$ varies the degree to which base learners are created with regards to the others previously created. Several known ensemble methods can be seen as special cases of Algorithm 1, for instance:

- Bagging is obtained by using a squared loss function $L(y, \hat{y}) = (y - \hat{y})^2$, setting $\nu = 0$, and choosing S_m to be a bootstrap sample of data. Random Forest introduces additional dispersion by randomizing the tree generation algorithm. Friedman *et al.* (2003) show that choosing a bootstrap sample of data is roughly equivalent to setting $\eta = N/2$, as opposed to the original algorithm of Bagging and Random Forest. In both cases, the coefficients are set to $a_0 = 0$ and $\left\{a_m = \frac{1}{M}\right\}_1^M$ so that predictions are simple averages of those included in the base learners

in the whole ensemble.

- Adaboost uses exponential loss $L(y, \hat{y}) = \exp(-y \cdot \hat{y})$ for $y \in \{-1, 1\}$, and is equivalent to setting $\nu = 1$ and $\eta = N$. Predictions are obtained as the sign of the final memory function $F_M(\mathbf{x})$, $\hat{y} = \text{sign}(F_M(\mathbf{x}))$.

- Gradient Boosting is a generalization of the Adaboost algorithm for both regression and classification: it uses using a variety of loss functions (e.g., least squares, absolute-deviations, logistic, etc.), and $\nu = 0.1$ and $\eta = N/2$, while predictions are given by $F_M(\mathbf{x})$.

- MART (Multiple Additive Regression Trees) is the implementation of Gradient Boosting when the base learners are trees. When using trees as base learners, one has the option of controlling the size of each tree, namely selecting the number of terminal nodes J, which corresponds to setting the interaction depth allowed for predictor variables. For instance, main effects are obtained setting $J = 2$, while two-interaction effects are included by letting $J = 3$.

It is of interest to note that all these ensembles can be implemented using R: namely Bagging is implemented in the `ipred` package, Random Forest in the `randomForest` package, while Adaboost, Gradient Boosting, and MART in the `gbm` package.

2.2 Why Ensembles Work

Dietterich (1997) gives three reasons to understand why ensembles can improve performance and why it may be difficult or impossible for single classifiers to provide similarly well.

- *Large hypothesis space.* The first reason may be found in the fact that machine-learning algorithms work by searching a space of possible hypotheses for the most accurate hypothesis (i.e. the hypothesis that best approximates the unknown function f). The training data might not provide enough information for choosing a single best base learner. Machine learning algorithms consider very large hypothesis spaces, so - even after the elimination of many hypotheses, there may be many others left, i.e. there may be many learners performing equally well on the training data. From these set of remaining hypotheses, one

can construct an ensemble of classifiers, and combining them may be a better choice.

- *The search process may be imperfect.* Another problem arises when the learning algorithms used are sub-optimal, which is often due to the fact that the problem at hand is usually NP-hard, thus practical ways to solve problems are carried out through heuristics. Thus, running the search algorithms with a slightly different training sample or injected noise may be useful to find a different (suboptimal) hypothesis. Therefore, ensembles can be seen as a way to compensate for imperfect search processes.

- *The hypothesis space being searched might not contain the true target function.* A third reason may be due to the fact that our hypothesis space might not contain the true function f, but it may include several equally good approximations of f. Thus, taking weighted combinations of these approximations, one might be able to represent classifiers that lie outside the search hypothesis space, with ensembles giving some good approximation of it. For instance, if the true boundary between two classes is a diagonal line, using a single decision tree (whose boundaries are line segments parallel to the coordinate axes) cannot lead to a good result, while a good approximation can be achieved by combining a set of decision trees.

In the present work we propose the use of ensembles based on trees, focusing on random forests. Such algorithm is an example of a parallel ensemble, meaning that it can be easily parallelized, which is a key feature considering the huge amount of variables we need to deal with.

Chapter 3

Classification and Regression Trees (CART)

CART (Breiman *et al.*, 1984) are a top-down algorithm that iteratively splits the training data into disjoint subsets according to a sequence of conditions involving a single covariate at a time. The algorithm starts with the whole dataset $D = \{y_i, \mathbf{x}_i\}_1^N$ (called root), browses all the input space $\mathcal{X}$ in search of a covariate, along with its cutting point c, that meets some criterion, and splits such data into two disjoint subsets (called daughter nodes), each defining a region R_j, $j = 1, 2, \ldots, J$ of the input space $\mathcal{X}$. Such nodes are treated in a similar fashion, i.e. split according to a single covariate, along with its cutting point c, meeting a given criterion. The algorithm keeps on partitioning the training data until some stopping rule is met, e.g. the minimum number of observations to include in either child node, and terminal nodes (called leaves) are obtained.

In each terminal node a simple model (e.g. a constant) is fitted. Thus, each terminal node can be expressed as:

$$\hat{y} = \sum_{j=1}^{J} \hat{c}_j I_{\hat{R}_j}(\mathbf{x}) \tag{3.1}$$

where I is the indicator function which equals 1 if the i-th variable is used to split R_j and 0 otherwise.

The resulting model can be represented as a binary tree, where observations satisfying the above mentioned condition at each step are assigned to

the left branch, and others to the right branch (see Figure 3.1).

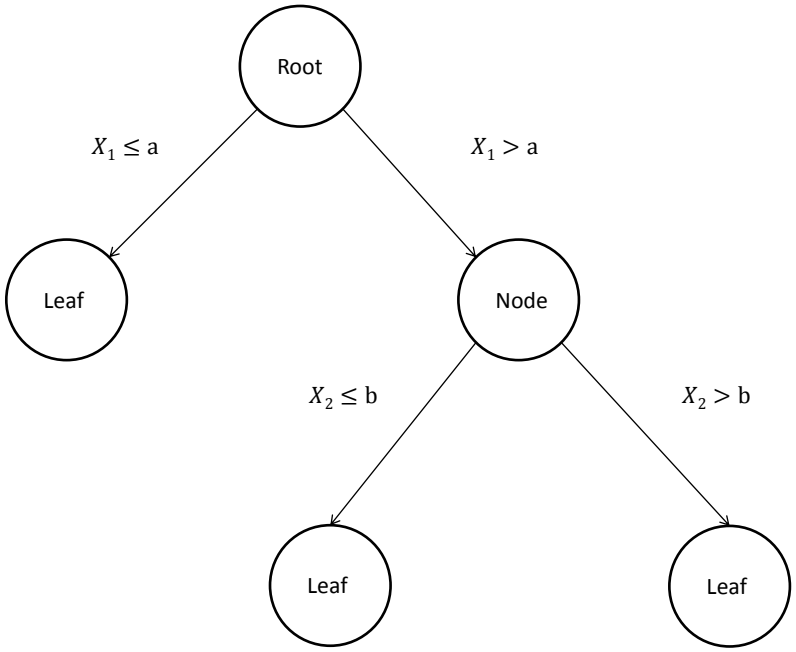

Figure 3.1: A CART example: The root is split according to X_1 and its cutting point a into left and right node. Right node is again split according to variable X_2 and cutting point b.

Because all these regions are disjoint, every possible input $\mathbf{x}$ belongs in a single one, and the tree model can be though of as the sum of all these regions. Trees accommodate different loss functions quite easily. For example in the regression case, two of the most used are the squared loss and the absolute loss. In the former case, the optimal constant $\hat{c}_j$ is the mean, while in the letter it is the median of the data points within region R_j. In the binary classification case, where p is the proportion in the second class, the most used measures are the following:

- the misclassification error: $1 - \max(1, 1-p)$;
- the Gini index: $2p(1-p)$;
- the cross entropy or deviance: $-p\log p - (1-p)\log(1-p)$.

If we choose the squared loss, the search problem, i.e. finding the tree

with the lowest prediction risk, can be formulated as:

$$\left\{\hat{c}_j, \hat{R}_j\right\}_1^J = \underset{\{c_j,R_j\}_1^J}{\arg\min} \sum_{i=1}^{N} \left(y_i - \hat{y}_i\right)^2 = \underset{\{c_j,R_j\}_1^J}{\arg\min} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{J} c_j I_{R_j}\left(\mathbf{x}_i\right)\right)^2 . \quad (3.2)$$

To solve, one searches over the space of all possible constants and region minimize the selected loss function. Because unrestricted optimization with respect to $\{R_j\}_1^J$ is very difficult, such partitions can be chosen to be axis-parallel. Similarly, because the joint optimization with respect to $\{R_j\}_1^J$ and $\{c_j\}_1^J$ is NP-complete (Hyafil and Rivest, 1976), a greedy approach is adopted, i.e. optimization is carried out at each step according to a single covariate. Standard implementation of CART algorithm are available in the packages `tree` and `rpart` in R.

3.1 Impurity Reduction

Let $(Y, \mathbf{X})$: $\Omega \to \left(D_Y \times D_{X_1} \times \ldots \times D_{X_p}\right) \equiv \mathcal{D}$ be a vector random variable defined on a probability space $(\Omega, \mathcal{F}, P)$, where $\mathbf{X} = \{X_1, \ldots, X_p\}$ is a set of covariates and Y is the response variable. The binary recursive partitioning nature of CART provides a hierarchical partitioning of the domain $\mathcal{X}$ into J disjoint (hyper-)rectangles (i.e. regions) $R_j \subset \mathcal{D}$, $j = 1, 2, \ldots, J$. Each rectangle is generated in $\mathcal{D}$ by splitting a parent rectangle in two parts through a binary split of the domain of a given covariate X_i. The impurity reduction yielded in R_j by X_i at the cutpoint s, is given by

$$d_{ij}^s = \Delta H_Y\left(X_i, R_j\right) = p_j \cdot \left\{H_Y - \left(p_{jL} H_{Y|X_i \le s} + p_{jR} H_{Y|X_i > s}\right)\right\}, \quad (3.3)$$

where $p_j = P\left(R_j\right)$, $p_{jL} = \left(X_i \le s | R_j\right)$, and $p_{jR} = \left(X_i > s | R_j\right)$. H_Y, $H_{Y|X_i \le s}$, and $H_{Y|X_i > s}$ are the heterogeneity indexes of Y in the j-th rectangle and in the left and right splits of R_j. Let d_{ij} be the maximum heterogeneity reduction provided by X_i in the j-th rectangle, for all possible cutpoints $s \in D_{X_i}|R_j$:

$$d_{ij} = \max_{s \in D_{X_i|R_j}} d_{ij}^s. \quad (3.4)$$

In CARTs, in a given R_j the splitting variable X_i, along with its cutting point s, is chosen so that the heterogeneity of Y is maximally reduced. Heterogeneity indexes for CART are the Gini index, and variance in case of regression. In the regression case d_{ij}^s can be rewritten as:

$$d_{ij}^s = \Delta\sigma_Y^2\left(X_i, R_j\right) = p_j \cdot \left\{\sigma_Y^2 - \left(p_{jL}\sigma_{Y|X_i\leq s}^2 + p_{jR}\sigma_{Y|X_i>s}^2\right)\right\}, \tag{3.5}$$

where σ_Y^2, $\sigma_{Y|X_i\leq s}^2$, and $\sigma_{Y|X_i>s}^2$ are the variances of Y in R_j and in the left and right splits. When a tree t is built with sample size N, these quantities are estimated by:

$$\hat{d}_{ij}^s = \widehat{\Delta H_Y}\left(X_i\right) = \frac{n_j}{N} \cdot \left\{\widehat{H_Y} - \left(\frac{n_{jL}}{N}\widehat{H}_{Y|X_i\leq s} + \frac{n_{jR}}{N}\widehat{H}_{Y|X_i>s}\right)\right\}, \tag{3.6}$$

where $\widehat{H_Y}$, $\widehat{H}_{Y|X_i\leq s}$, and $\widehat{H}_{Y|X_i>s}$ are the estimated heterogeneities of Y in the j-th rectangle and in the left and right splits, while n_j, n_{jL}, and n_{jR} are the sample size in node j and in the left and right splits. Similarly, d_{ij} is estimated by:

$$\widehat{d}_{ij} = \max_{s\in S_{ij}} \widehat{d}_{ij}^s \tag{3.7}$$

where S_{ij} is the set of available cutpoints for variable X_i at node j. In the regression case, d_{ij}^s is estimated using the sample variance $\widehat{\sigma}^2$:

$$\begin{aligned}
\widehat{d}_{ij}^s = \Delta\widehat{\sigma}_Y^2\left(X_i\right) &= \frac{n_j}{N}\left\{\widehat{\sigma}_Y^2 - \left(\frac{n_{jL}}{N}\widehat{\sigma}_{Y|X_i\leq s}^2 + \frac{n_{jR}}{N}\widehat{\sigma}_{Y|X_i>s}^2\right)\right\} \\
&= \frac{n_j}{N}\left\{\frac{DEV_{total}\left(j\right)}{n_j} - \frac{DEV_{within}\left(jL, jR\right)}{n_j}\right\} \\
&= N^{-1}DEV_{between}\left(jL, jR\right),
\end{aligned} \tag{3.8}$$

where DEV_{within} and $DEV_{between}$ are the within-node and between-node deviances.

3.2 CART Properties

The binary recursive partitioning in CARTs feature the following characteristics.

1. **Capability of dealing with irrelevant features**. As quickly mentioned before, CARTs belong to the class of feature selection algorithms known as embedded methods. This is due to the fact that at every node, the input space is inspected and, at the end of the procedure, only a limited number of variables are selected in the tree.

2. **Data do not need preprocessing**. CARTs can handle numeric, binary, and categorical variables without the need of preprocessing (e.g. scaling), because splits are not affected by monotonic transformations.

3. **Scalability**. CARTs are very fast to compute, with approximate time complexity of $O\left(pN\log N\right)$ (Seni and Elder, 2010).

4. **Capability of dealing with missing values**. If the predictor variables have missing values, CART handles them via a mechanism called surrogate splits. This method exploits the correlation between predictors to try and ease the problem of missingness. When considering a predictor for a split, only the observations for which that predictor is not missing are used. When the primary (best) predictor - and its corresponding splitting point - is selected, a list of surrogate variables - along with their split points - can be created. The first surrogate variable is the predictor - along with its split - that best mimics the primary split. The second surrogate variable is the predictor - along with its split - that does second best, and so on.

5. **Interpretability**. The main charm of CARTs is their output that makes the model clearly interpretable.

3.3 CART Limitations

Although CARTs show desirable properties, they suffer from some limitations, some of which are as follows:

1. **Lack of smoothness**. A first limitation of CARTs is the lack of smoothness of the prediction surface, which is particularly true in the

regression case where the underlying function to be approximated is expected to be smooth.

2. **Difficulty capturing additive structure**. The binary partitioning nature of CART may need many splits to recreate such structure, and thus needing a dataset with a large sample size.

3. **Instability**. The greedy search strategy causes high variance, which means that small change in the data, e.g. due to sampling fluctuations, can cause big changes in the resulting model.

Chapter 4

Random Forests

In the following chapter, we present the Random Forest algorithm along with its main characteristics.

4.1 Random Forest Algorithm

Random forests (RF) are a well-established machine learning tool, first introduced by (Breiman, 2001). They are an ensemble method that uses classification and regression trees as base learners (Breiman *et al.*, 1984). The main drawback of CARTs is the fact that they produce results that are data dependent, i.e. tiny changes in the input data may result in completely different trees. In order to correct such behavior, Breiman (1996) proposed bagging, i.e. an ensemble of CARTs grown on different bootstrap samples of the dataset, the resulting aggregation being a more stable classifier. In bagging, the variance reduction is limited by the high correlation between trees. This latter issue was resolved by Breiman adopting the random subspace method by (Ho, 1998), i.e. instead of searching the whole feature space for the optimal split, only a small (random) subset of features is selected.

The main charm of Random Forests is the fact that they provide good prediction accuracy, importance of predictor variables accounting for both main effects and interactions, internal estimation of error (out-of-bag error rates) working as a suitable surrogate for cross validation, robustness to noise, and the fact that predictors are allowed to be related to the outcome in a nonlinear fashion.

The algorithm works as follows:

1. Let $D = \{y_i, \mathbf{x}_i\}_1^n$ be the training data, containing a set of predictor variables $\mathbf{X}$ with dimensions $n \times p$, and $n \times 1$ response vector $\mathbf{y}$, where p is the number of predictor variables and n is the sample size.

2. A boostrap sample D^* with size n is drawn with replacement from the original sample D. Note that a fraction of the observations are not sampled, these are referred to as *out-of-bag* observations, and are used for prediction purposes.

3. A classification or regression tree t is grown to its largest extent using D^*. At each node a random subset is drawn (without replacement) from the p predictor variables in order to determine the best split. The number of variables randomly selected is held constant during the whole procedure.

4. For each tree t, the *out-of-bag* prediction error is computed.

5. The previous steps are repeated to grow a prespecified number of trees, T. Results are aggregated in the forest, i.e. samples are predicted as simple averages in the regression case, while they are predicted through a majority vote in the classification case.

6. The overall performance of the forest is computed: in the regression case - as the amount of variance explained= $1 - \left(MSE_{OOB}/\hat{\sigma}_y^2\right)$, where $MSE_{OOB} = n^{-1} \sum_1^n \left(y_i - \hat{y}_i^{OOB}\right)^2$; while - in the classification case - the misclassification error rate is computed.

As suggested by Liaw and Wiener (2002), default parameters work just fine for most problems. On the other hand, it is advisable to tune the number of variables randomly selected at each split in order to obtain the smallest error rate, and to set the number of trees T large enough for the error to stabilize.

Another characteristic of random forest is the calculation of proximities between observations, i.e. for the observations assigned to the same node in a tree their pairwise proximity is increased by one. Once the forest has been created, a proximity matrix is obtained, which allows the identification of outliers, and missing values imputation.

4.1.1 Generalization Erorr

In his seminal paper, Breiman (2001) discusses bounds on the generalization error. It depends on the correlation between single trees and the prediction error provided by each one of them. To describe this in a formal way, let us give some theoretical definitions.

Given a random variable $(Y, \mathbf{X})$, where Y is categorical and $\mathbf{X}$ is a set of categorical or numeric variables, with realization $(y, \mathbf{x})$, a classifier $h(\mathbf{x})$ is a statistical tool capable of predicting y given the input $\mathbf{x}$. A random classifier $h(\mathbf{x}, \Theta)$ is a classifier whose prediction about y depend on both the input $\mathbf{x}$ and on a random vector Θ from a known distribution. A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(\mathbf{x}, \Theta_k), k = 1, \ldots\}$ where $\{\Theta_k\}$ are independent and identically distributed random vectors from a known distribution, each predicting the values of y given $\mathbf{x}$. By definition a RF consists of an infinite number of classifiers, but from an operational point of view a finite set of classifiers $\{h(\mathbf{x}, \Theta_1), \ldots, h(\mathbf{x}, \Theta_k)\}$ is used. The random forest prediction for y corresponds to the most popular prediction of the collection of the classifiers. RF for regression are formed by growing trees depending on a random vector Θ from a known distribution, such that the tree prediction $h(\mathbf{x}, \Theta)$ takes on numerical values instead of class labels. The mean-squared generalization error for any numerical predictor $h(\mathbf{x})$ is

$$E_{\mathbf{X},Y}(Y - h(\mathbf{X}))^2. \tag{4.1}$$

The random forest predictor is formed by taking the average over k trees $\{h(\mathbf{x}, \Theta_k)\}$.

Theorem 1. *As the number of trees in the forest goes to infinity, almost surely*

$$E_{\boldsymbol{X},Y}(Y - av_k h(\boldsymbol{X}, \Theta_k))^2 \to E_{\boldsymbol{X},Y}(Y - E_\Theta h(\boldsymbol{X}, \Theta))^2. \tag{4.2}$$

Proof. see Appendix I in Breiman (2001). □

The right hand side of the equation is the generalizing error of the forest,

defined as $PE^*(forest)$. Let us define the average generalization error of a tree as:

$$PE^*(tree) = E_\Theta E_{\mathbf{X},Y}\left[Y - h(\mathbf{X},\Theta)\right]^2. \tag{4.3}$$

Theorem 2. *Assume that for all Θ, $EY = E_{\boldsymbol{X}} h(\boldsymbol{X},\Theta)$. Then*

$$PE^*(forest) \leq \bar{\rho} PE^*(tree) \tag{4.4}$$

where $\bar{\rho}$ is the weighted correlation between the residuals $Y - h(\boldsymbol{X},\Theta)$ and $Y - h(\boldsymbol{X},\Theta')$, where Θ and Θ' are independent.

Proof.

$$\begin{aligned} PE^*(forest) &= E_{\mathbf{X},Y}\left[Y - h(\mathbf{X},\Theta)\right]^2 \\ &= E_\Theta E_{\Theta'} E_{\mathbf{X},Y}\left(Y - h(\mathbf{X},\Theta)\right)\left(Y - h\left(\mathbf{X},\Theta'\right)\right) \end{aligned} \tag{4.5}$$

the term on the right is a covariance and can be written as:

$$E_\Theta E_{\Theta'}\left(\rho\left(\Theta,\Theta'\right) sd(\Theta)\, sd\left(\Theta'\right)\right) \tag{4.6}$$

where $sd = \sqrt{E_{\mathbf{X},Y}\left(Y - h(\mathbf{X},\Theta)\right)^2}$. The weighted correlation is defined as:

$$\bar{\rho} = E_\Theta E_{\Theta'}\left(\rho\left(\Theta,\Theta'\right) sd(\Theta)\, sd(\Theta)\right) / \left(E_\Theta sd(\Theta)\right)^2. \tag{4.7}$$

Then

$$PE^*(forest) = \bar{\rho}\left(E_\Theta sd(\Theta)\right)^2 \leq \bar{\rho} PE^*(tree). \tag{4.8}$$

□

As a result, the requirements for an accurate regression forest are: low correlation between residuals and low error trees. Thus, the randomization process needs to aim at low correlation. Although in Breiman's words *The*

generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large, some works show that the random forest classifier is not universally consistent (Biau *et al.*, 2008; Traskin, 2007). This result is nonetheless important because it indicates that random forest does not overfit as more trees are grown, with more trees being generally better than fewer trees because the true value of the generalization error will be more accurately approximated.

Similarly, Lin and Jeon (2006) derive lower bounds for the mean-squared error for a regression forest under random splitting by drawing analogies with adaptive nearest-neighbor methods. Meinshausen (2006) proved consistency for quantile regression forests, Biau *et al.* (2008) proved consistency of random forest for classification under the assumption of random splitting, while Ishwaran and Kogalur (2010) proved uniform consistency of random survival forests.

4.1.2 Implementation and Tuning Parameters

The random forest algorithm is implemented in the `randomForest` package in R. Although there is a large number of tuning parameters, random forests are known to work just fine in practice with default parameters. The parameters that may undergo some tuning are the following:

1. *Node Size*: The number of observations the terminal nodes. Although in a single CART, a very small number can lead to overfitting, the idea in random forest is to grow trees to their largest extent, in order to have as little bias as possible. In a similar way to Bagging, the large variance is mitigated through the averaging process over all trees.
   In the R implementation of random forests, default sample sizes are one for classification and five for regression.

2. *Number of Trees*: A random forest can easily grow some hundreds to some thousands trees. In practice, the default value of `ntree=500` is a good compromise. It is strongly advisable to grow a number of trees which is large enough for the OOB error to converge, while larger ntree values lead to slightly more stable values of variable importances (Díaz-Uriarte and De Andres, 2006).

3. *Number of randomly selected predictors*: The number of randomly selected variables at each split is maybe the most important parameter in random forest. As already mentioned, it affects the ability of the ensemble to create more diverse base learners. Usually - although maybe surprising - very few predictors can be sampled at each split, and - with a sufficient number of trees - each predictor will be able to contribute. It is generally recommended to tune the number of randomly selected predictors (`mtry` in the R implementation), in order to obtain the smallest OOB error. This can be achieved through the `tuneRF` function in the `randomForest` package in R, alhtough one must bear in mind to be careful not to overtune the algorithm and introduce the overfitting that the algorithm should prevent. Alternatively, one can use the `train` function in the `caret` package, which does the same but exploiting cross-validation.

4. *The size(s) of sample to be drawn*: As a default, the probability of an observation being drawn in the bootstrap sample is equal to $1 - (1 - 1/n)^n \approx .632$, so that on average 1/3 of training data is not used. These are referred to as *out-of-bag* sample, and this is the part of the observations used for prediction purposes. This can be modified using the parameter `sampsize` in the `randomForest` package in R.

Different implementations are present in different packages, e.g. in the STATISTICA software, a given percentage of the dataset is heldout for testing purposes (default `Random test data proportion`=30%), and the subsample proportion to be used for drawing the bootstrap samples can be tuned (default `Subsample proportion`=50%), while default the default parameter for the number of predictors randomly sampled is set to $\log_2(p+1)$.

It is worth stating that the before mentioned variations of random forests can be briefly described as follows:

- Random Survival Forest is a modification for the analysis of right-censored survival data (implemented in the `randomSurvivalForest` package in R),

- Quantile Regression Forest is a generalization of the random forest algorithm for quantile regression (implemented in the `quantregForest` package in R).

4.2 Variable Importance Measures (VIMs)

Random Forests are frequently treated as a *black-box* model, in the sense that they can be seen in terms of input-output without any knowledge of its internal workings, mainly because of the large number of trees involved, as opposed to linear models or CART. However, they provide some metrics that aid in interpretation. A key feature of RF is that the importance of each predictor can be measured. The standard VIMs are the Gini importance (aka resubstitution measure) and the permutation importance.

4.2.1 Gini Importance

The Gini importance of a predictor variable X_i is the so-called total decrease in node impurity (TDNI). It is obtained as the summation of the impurity decrease of all nodes in a forest, where predictor variable X_i is used for splitting. Let d_{ij} be the decrease in the heterogeneity index allowed by variable X_i at node $j \in J$. The Gini importance for variable X_i for a single tree t is given by:

$$g_{(X_i;t)} = \sum_{j \in J} d_{ij} I_{ij} \tag{4.9}$$

where I is the indicator function which equals 1 if the i-th variable is used to split R_j and 0 otherwise. This quantity measures the reduction in the heterogeneity of the outcome variable given by X_i each time that particular predictor variable is used to define a partitioning of the data. In the case of regression, the heterogeneity is defined in terms of deviance, while in the case of classification, various measures can be used: the misclassification error, the Gini index, and the cross-entropy or deviance. For instance in the case of two class prediction, if p is the proportion in the second class, these quantities are defined as $1 - \max(p, 1-p)$, $2p(1-p)$, and $-p\log p - (1-p)\log(1-p)$.

For the whole RF ensemble the Gini importance is given by the average of $g(X_i;t)$ over the set of T trees:

$$g_{(X_i)} = \frac{1}{T} \sum_{t=1}^{T} g_{(X_i,t)} \tag{4.10}$$

4.2.2 Permutation Importance

This forecasting measure uses the OOB observations. The first version of this measure is called unscaled permutation importance, which is the mean decrease of accuracy for a prediction variable. This measure is computed for predictor X_i for each tree t in the forest: compute the prediction accuracy A_t using the OOB observations, then the observations of X_i are shuffled, and the prediction accuracy A_t^* is recomputed using the OOB observations. The quantity $(A_t - A_t^*)$ is then averaged over all trees in the forest, and gives the unscaled (or raw) permutation importance of predictor X_i

$$\bar{d}_{(X_i)} = \frac{1}{T}\sum_{t=1}^{T}(A_t - A_t^*) \tag{4.11}$$

Shuffling an important predictor then leads to a decrease in accuracy, as a result a positive measure follows. Sometimes it is possible that the accuracy slightly increases when a predictor is randomized, as a result a negative measure follows. Negative permutation importance measures can be treated as no decline in accuracy. The terms A_t and A_t^* are not generally defined in (4.11): one could use forecasting errors, the percentage of cases forecasted incorrectly, or some other measure (see for instance Breiman (2002)). Currently, the standard measures are the proportion (or percentage) of cases misclassified for classification, and the increase in mean square error (MSE) for regression.

A slightly different version of this measure is the scaled permutation importance (called z-score), which is calculated by diving the mean decrease in accuracy by its standard error

$$z_{(X_i)} = \frac{\bar{d}_{(X_i)}}{\sqrt{\frac{s^2}{T}}} \tag{4.12}$$

where s^2 is the sample variance estimator of the quantity $(A_t - A_t^*)$ over all the trees in the forest. The s^2 estimator is defined as

$$s^2 = \frac{1}{T}\sum_{t=1}^{T}(A_t - A_t^*)^2 - \bar{d}^2_{(X_i)}. \tag{4.13}$$

The permutation measure, defined as Measure 1 by Breiman (2002) as opposed to the Gini measure called Measure 4, was thought of a solution to the well-known biased conveyed by the *in-sample* Gini measure.

4.2.3 Bias and Conditional Permutation Variable Importance Measure

The Gini importance is well known to show a bias for predictors providing more splits, and this is analyzed later on in this work. A lot of attention has been drawn on the permutation VIM as being a possible source of a similar bias. Strobl *et al.* (2007a) noticed that if predictor variables are of different type, e.g. different scales of measurement, different numbers of categories, while there is no effect in the mean values of the distributions of the permutation importance measures, i.e. close to zero for uninformative variables, there are notable differences in the variance of these values. This means that in a single run of the random forest algorithm, one could be induced to a sever over- or underestimation of the importance of variables that have more categories, even though they are no more or less informative than the other variables. For this reason, Strobl *et al.* (2007b) provide an unbiased variable selection for random forests within the framework of condition inference trees (Hothorn *et al.*, 2006), in this case the deviation of the permutation importance measure over the simulation runs does not increase substantially with the number of categories or scale of measurement of the predictors. That is implemented through the `cforest_unbiased` function in the R package `party`. The implementation of the permutation importance in the `cforest` function is slightly different from that in the `randomForest` function, in that the randomization is carried out not shuffling the values of the variable of interest, but randomly assigning observations to child nodes of splits in such variable, as explained in (Hapfelmeier *et al.*, 2011) in order to allow for missing values in the explanatory variables.

Another source of bias in the permutation VIM is due to the correlation among predictors. Nicodemus and Malley (2009) show that - under the null model - permutation VIM obtained with RF with the Gini index as the splitting criterion show larger standard deviation for uncorrelated predictors, while on average it provides an unbiased estimate of the importance of a

predictor. In this case conditional inference forests were more reliable. This again indicates that it is advisable to run the algorithm multiple times to obtain a measure of central tendency and variability for VIMs. Therefore the authors advise that one should avoid using Gini VIMs because they show a substantial bias under predictor correlation, while permutation based VIM is unbiased, although the one provided by RF implementing Gini based splits show a larger variability, and are therefore less reliable.

On the other hand Strobl *et al.* (2008) showed that permutation importance overestimates the importance of correlated predictor variables, and identify two possible causes for this bias: (i) a preference for the selection of correlated predictors in early splits in the tree building process, and (ii) the particular permutation scheme employed in the computation of the permutation VIM. This bias is particularly important in cases where spurious correlation between the predictors and the outcome is present. A solution to this problem is the conditional permutation importance: in this case, a single predictor variable is again shuffled - as in the regular permutation VIM - but within groups of observations of other predictor variables. This permutation scheme allows to preserve the correlation structures among predictor variables and detects any spurious correlation among predictors.

This being said, Nicodemus *et al.* (2010) provide some power-case simulations to support the fact that permutation VIM provides larger measures for correlated predictors, and propose the use of the conditional permutation VIM. In both cases, they measure some bias. In the case of the permutation VIM they detect higher measures for correlated predictors, while as for the conditional VIM, they detect larger measures for uncorrelated predictors, although according to them, the larger variability of the conditional VIM, makes the inflation less pronounced.

The conditional VIM is available in the `party` package though the function `varimp(obj,conditional = TRUE)`, but according to the authors it is extremely computationally intensive: Nicodemus *et al.* (2010) state they were able to compute (un)scaled VIMs on the full set of observations (n=2000),

whereas they were only able to calculate the conditional VIM on a subset (n = 500) of observations. For this reason, they suggest the use of the permutation VIM in large-scale screening studies, such as genome wide association studies, for which the original permutation VIM may be better suited to identify regions of markers containing influential predictors, because in this case correlation is usually a consequence of physical proximity of loci and thus may help localize causal variants; while they suggest one should use the conditional permutation VIM in small studies where the aim is to uncover spurious correlations and identify a set of truly influential predictors among a set of correlated ones.

4.2.4 Permutation Importance by Meng

In GWAS studies, it is still not known how and to what extent Linkage Disequilibrium (LD, i.e. correlation) between non-causal SNPs and true risk SNPs influences the ability of RF to correctly identify the true risk SNPs. According to Meng *et al.* (2009), the correlation would cause the importance of each risk SNP that has non-causal SNPs correlated with it to be reduced. Meng *et al.* (2009) proposed a revised RF algorithm:

1. change the tree-building algorithm by building each tree in an RF only with SNPs in Linkage Equilibrium (LE, i.e. uncorrelated),

2. modify the permutation importance measure as follows

$$\bar{d}_{(X_i)}^{\texttt{MENG}} = \frac{1}{T_{X_i}} \sum_{t=1}^{T} (A_t - A_t^*) \tag{4.14}$$

   which means that the decrease in accuracy is not averaged over all T trees in the RF, but only over T_{X_i}, i.e. over the total number of trees in which variable X_i appears.

The authors found out that novel permutation importance in combination of the revised method was sometimes inflated (as the number of SNPs in LD with the causal variant increased), and suggest the use of the revised importance measure in combination with the original RF for most stable performance when the genetic model and the number of SNPs in LD with risk SNPs are unknown.

4.3 The Proximity Matrix

In addition to variable importance measures and partial dependence plots (analyzed later in this work), which show the importance of each predictor, and provide a description of how each predictor is related to the response, the random forest algorithm provides some additional features. It can be useful to determine the extent to which observations tend to be predicted similarly. This can be carried out thought the *proximity matrix*, which is constructed as follows:

Algorithm 2: Proximity Matrix Construction

1 **for** *Each each tree* **do**
2 Grow a tree as usual;
3 Drop observations down the tree;
4 For all possible pairs of cases, if a pair ends in the same terminal node, increase their proximity by one;
5 Normalize by dividing by the number of trees

Note that the data used to compute the proximity matrix (line 3) can either be all the observations or only the OOB data. The result is an $n \times n$ matrix with each cell showing the proportion of trees for which each pair of observations is placed in the same terminal node. The higher that proportion, the more proximate these observations are. This proximity matrix can be used for three basic applications.

4.3.1 Clustering Data

The proximity matrix can be treated as a similarity matrix and a multidimensional scaling can be applied. Usual plots of the observations in the first two dimensions can be used as an exploratory tool, and one can see if the data tend to cluster in the space defined by predictors.

4.3.2 Imputing Missing Values

There are two ways in which random forest can impute missing data. The first and quick method (`na.roughfix` in R) relies on a measure of location. For numerical predictors, the median of the available values is used, for categorical predictors, the mode category of the available data is used. For

small amounts of missing values, this method can suffice, considering the computational requests of the second.
The second method exploits the proximity matrix. First, missing values are imputed through the rough method mentioned above, then the weighted average of the values of the nonmissing cases for that variable is used (numerical variable), or - for categorical predictors - the inputed value is the most common nonmissing value for the variable, with the frequencies weighted by proximity. The step using proximity values is then iterated several times, usually four to six times.

Two important remarks about this procedure must be made: first, it could be computationally too demanding for some datasets, and imputed values tend to make OOB estimates a bit too optimistic. Therefore, one must consider whether imputing missing values should be carried out using one method or the other.

4.3.3 Detecting Outliers

The proximity matrix can also be used to detect outliers. The basic idea is that outliers are observations whose proximities are small. Currently the procedure is implemented in R through the `outlier` function, but only for a categorical response variable, for which outliers are defined within its categories.

Algorithm 3: Outlier Detection in Random Forest

1 **for** *Each each category of the response* **do**
2 For each observation, compute the inverse of the sum of the squared proximities with all of the other observations in the same outcome class (unstandardized values);
3 Compute the median and mean absolute deviation around the median of these values;
4 Subtract the median and divide by the mean absolute deviation to obtain the standardized values (negative values are set to 0);

A large value will indicate that on average the proximities of a given observation are small, i.e. that observation may be considered an outlier. Although random forest is known to be robust against outliers, it can be

advisable to drop the outliers from the training data, specially in case the overall number of observations is modest (e.g. less than 100).

Chapter 5

Variable Selection Methods

In the following pages, we present three methods that can be generally used for variable selection purposes, although their focus in this work is that of SNP selection for GWAS studies. All can be encompassed in the definition of wrappers around the random forest algorithm.

5.1 Random Forest Recursive Feature Elimination

Díaz-Uriarte and De Andres (2006) suggest the use of Random Forests for the selection of relevant genes, which is a common task in gene expression studies. In this scenario, one is usually interested in one of the following objectives:

1. To identify relevant genes for further research, which involves obtaining a (probably) large set of genes that are related to the outcome of interest, and this set should include genes performing similar biological functions, which can be highly correlated.

2. To obtain the smallest possible set of genes that provide good predictive performance, which could be used for diagnostic purposes in clinical practice (thus, highly correlated genes should not be selected).

The aim of RF-RFE is the latter. The main advantage of this method is that it returns a very small set of genes that retain a high predictive accuracy. Díaz-Uriarte and De Andres (2006) compare this method to others used in the field (e.g. Support Vector Machines, Nearest Neighbor, Diagonal Linear

Discriminant Analysis, and Shrunken Centroids) and both simulations and the use on real data set show that it is competitive.

Variable selection with microarray data can lead to many solutions that are equally good from the point of view of prediction capabilities, but that share few common genes. This issue is referred to as the multiplicity/lack of stability/lack of uniqueness problem, and is not necessarily a bad thing if the purpose of the analysis is solely focused on prediction. This unfortunately shifts the relevance away from the selection of particular genes, and, as emphasized in (Somorjai *et al.*, 2003), one must bear in mind the medical/biological interpretability of the problem at hand. One method that does not evaluate the stability of its results leads to a false sense of trust on the interpretability of the output obtained.

Loosely speaking, the algorithm iteratively fits a random forest eliminating a fraction of the least important variables - ranked according to the permutation importance measure - at each step. After fitting all the forests, the OOB error is inspected, and one selects the number of genes providing the smallest prediction error. Note that because of the iterative nature of this algorithm, the OOB error is biased down, and cannot be used to asses the overall error rate, therefore Díaz-Uriarte and De Andres (2006) suggest the use of the bootstrap to assess prediction error rates. The reasons why the OOB error should be avoided are similar to those leading to *selection bias* (Ambroise and McLachlan, 2002), although Díaz-Uriarte and De Andres (2006) suggest that using error rates is not necessarily a bad procedure from the point of view of selecting the final set of genes, as suggested in (Braga-Neto *et al.*, 2004), where it is shown that - in certain cases - resubstitution performs as well as cross-validation, and a priori it is not clear if it is less appropriate than cross-validation for the purpose of ranking feature sets.

This is coherent with what Ambroise and McLachlan (2002) suggest, namely, in order to assess and correct for selection bias, one can either perform a crossvalidation or apply the bootstrap external to the selection process. Thus, a resampling scheme has to be included in the algorithm, and

after all the forests have been fitted, one selects the solution with the smallest number of genes whose error rate is within u standard errors: setting $u = 0$ results in selecting the set of genes that leads to the smallest error rate, while setting $u = 1$ is similar to the common one s.e. rule suggested by Breiman (2001) and leads to solutions with fewer genes.

Ambroise and McLachlan (2002) recommend 10-fold rather than leave-one-out cross-validation, and concerning the bootstrap, suggest using the so-called .632+ bootstrap (Efron and Tibshirani, 1997) error estimate designed to handle overfitted prediction rules. The .632+ bootstrap method uses an average of the resubstitution error (the error when a classifier is applied to the training data) and the error on samples not used to train the predictor (the leave-one-out bootstrap error) weighted by a quantity that reflects the amount of overfitting. Díaz-Uriarte and De Andres (2006) opted for the use of the .632+ bootstrap method to obtain an honest estimate of the error rate, as to evaluate the stability of the variable selection procedure in the original sample: they investigate the frequency with which the genes selected in the original sample appear among the genes selected in the bootstrap samples. They report that the genes selected in the original sample are seldom selected in more than 50% of the bootstrap samples, which results are not affected by changes in the parameters, i.e. the number of trees to be grown (between 2000 and 5000), the number of randomly selected predictors, or the node size, while the use of the one s.e. rule can lead, in some cases, to a more stable solution. The number of variables to be dropped at each iteration can be adjusted to modify the resolution of the number of variables selected: smaller values yield a finer resolution in the examination of the number of genes, but are more computationally demanding. The procedure devised by Díaz-Uriarte and De Andres (2006) is implementable through the R package `varSelRF` - although only classification is allowed. It additionally provides the use of selection probability plots to evaluate the stability and confidence on the selection of relevant genes as proposed in (Pepe *et al.*, 2003). If the task is to rank genes and to select the top genes for further study, one can quantify the degree of confidence of choosing the gth gene among the top k:

$$P_g(k) = P\left[\text{gene } g\text{ranked in the top } k\right] = P\left[\text{Rank}(g) \leq k\right] \tag{5.1}$$

The value of $P_g(k)$ may be of particular interest for k equal to a predetermined number of genes to be selected.

The R package `caret` proposes a general framework for variable selection through recursive feature elimination. First, the algorithm fits the model with all predictors. Each predictor is ranked using its importance to the model. Let $S_1 > S_2 > \ldots > S_s$ be an ordered sequence indicating the number of predictors to retain. At each iteration of feature selection, the S_i top raked predictors are retained, the model is refit and performance is assessed. The value of S_i with the best performance is determined and the top S_i predictors are used to fit the final model. This procedure is described in Algorithm 4

Algorithm 4: Recursive Feature Elimination

```
1  Tune/train the model on the training set using all predictors;
2  Calculate model performance;
3  Calculate variable importance or rankings;
4  for Each subset size S_i,i = 1,...,s do
5      Keep the S_i most important variables;
6      [Optional] Pre-process the data;
7      Tune/train the model on the training set using S_i predictors;
8      Calculate model performance;
9      [Optional] Recalculate the rankings for each predictor;
10 Calculate the performance profile over the S_i predictors;
11 Determine the appropriate number of predictors;
12 Determine the final ranks of each predictor;
13 Fit the final model based on the optimal S_i predictors.
```

The algorithm has an optional step (line 9) where the predictor rankings are recomputed on the model on the reduced feature set. Svetnik *et al.* (2004) showed that, for random forest models, there was a decrease in performance when the rankings were recomputed at every step, as well as that this reranking procedure was more prone to overfitting. However, in other cases, when the initial rankings are not good (e.g. linear models with highly collinear predictors), recalculation can slightly improve performance. One potential issue of such algorithm is the overfitting to the predictor set such

that the wrapper procedure could focus on nuances of the training data that are not found in future samples (i.e. overfitting to predictors and samples). For example, suppose a very large number of uninformative predictors were collected and one such predictor randomly correlated with the outcome. The RFE algorithm would give a good rank to this variable and the prediction error (on the same data set) would be lowered. It would take a different test/validation to find out that this predictor was uninformative. This was referred to as *selection bias* by Ambroise and McLachlan (2002). In the current RFE algorithm, the training data is being used for at least three purposes: predictor selection, model fitting and performance evaluation. Unless the number of samples is large, especially in relation to the number of variables, one static training set may not be able to fulfill these needs.

Since feature selection is part of the model building process, resampling methods (e.g. cross-validation or the bootstrap) should factor in the variability caused by feature selection when calculating performance. To get performance estimates that incorporate the variation due to feature selection, it is suggested that the steps in Algorithm 4 be encapsulated inside an outer layer of resampling (e.g. 10-fold crossvalidation). Algorithm 5 shows a version of the algorithm that uses resampling.

Granitto *et al.* (2006) and Svetnik *et al.* (2004) applied this recursive method in various fields, even outside genetics, e.g. pharmaceutical and agroindustrial, suggesting cross-validation as the proper resampling method. While Svetnik *et al.* (2004) suggest the use of repeated (twenty times) 5-fold cross validation, Pang *et al.* (2012) go for 10 times 10-fold cross validation, while Granitto *et al.* (2006) opt for 100 times leave-group-out cross validation/random subsampling, i.e. they recursively split the data into train/test set with a 75%/25% proportion.
We have opted for a 10-fold cross-validation - which shows a good tradeoff between bias and variability - and repeated it ten times in order to have the same conditions shown in Granitto *et al.* (2006), i.e. instead of selecting genes on the original sample - like in Díaz-Uriarte and De Andres (2006) - and then inspecting their inclusion probabilities in the resampling, we in-

Algorithm 5: Recursive Feature Elimination incorporating resampling

```
for Each Resampling Iteration do
    Partition data into training and test/held back set via resampling;
    Tune/train the model on the training set using all predictors;
    Predict the held back samples;
    Calculate variable importance or rankings;
    for Each subset size S_i,i = 1,...,s do
        Keep the Si most important variables;
        [Optional] Pre-process the data;
        Tune/train the model on the training set using S_i predictors;
        Predict the held back samples;
        [Optional] Recalculate the rankings for each predictor;
Calculate the performance profile over the S_i using the heldback samples;
Determine the appropriate number of predictors;
Estimate the final list of predictors to keep in the final model;
Fit the final model based on the optimal S_i using the original training set.
```

spect those directly in the resampling: we check how many times (out of 100) a particular SNP is selected in the optimal solution using the training set. We have preferred repeated cross-validation over repeated subsampling because - generally speaking - in the latter some observations may be selected more than once while some others may be included in multiple resamplings (i.e. the tests sets may overlap). Kim (2009) suggest the use of repeated cross-validation estimator, for it outperforms the non-repeated one by reducing the variability of the estimator. Such results apply to large samples (i.e. $n \sim 1000$), while in the opposite case, different resampling approaches may prove more suitable (see Berrar *et al.* (2006)).

5.2 Feature Selection with the `Boruta` Package

To some abstract extent, variable selection can be done by simply looking at the permutation importance obtained after a single run of the random forest algorithm. It has become normal practice to compute random forest variable importances and to retain the first k variables and to rerun the algorithm in order to delete the least important/noisy variables. Very optimistically, assuming the raw scores are independent from tree to tree, we can compute a straightforward estimate of the standard error. Following Breiman's advice - backed by his empirical studies using many different types of data sets, which show that a good case can be made for independence - one could compute a z-score by dividing the raw score by the estimated standard error.

Because the individual raw scores are computed from T independent bootstrap samples, a simple test for the relevance of variable X_i could be constructed based on the central limit theorem for the mean importance $\overline{d}_{(Xi)}$. If each individual variable importance $\overline{d}_{(Xi)}$ has standard deviation s, the mean importance from T replications has standard error $s/\sqrt{T}$. Hence, under the null hypothesis of zero variable importance, the z-score is asymptotically standard normal: which means that in practice, whenever $z_{(Xi)}$ exceeds the α-quantile of the standard normal distribution, the null hypothesis of zero importance would be rejected. Simulation studies (Strobl and Zeileis, 2008), show that the power of the test against the null hypothesis of zero importance does indeed increase with the relevance of the predictor variable (as expected), yet it increases with the number of trees grown in the forest, i.e. the power increases according to a tuning parameter that

can be increased at the user's will. Moreover, the authors show that the power decreases with increasing sample size. Conversely, the unscaled score for a given variable increase with the relevance of the predictor variable and with the sample size as expected, while there is no effect of the number of trees on the average importance. The reason for such behavior is reckoned to be a result induced by the scaling process, which induces a dependence on the number of trees but at the same time inverts the dependence on the sample size. To support this, Rudnicki *et al.* (2006) have observed that, even if all attributes are completely random and thus not related to the outcome variable, relative strong correlations between some attributes and the outcome variable may appear by chance. As a consequence, these noisy attributes get z-scores that suggest strong, non random dependence on the decision attribute. Unfortunately, being Breiman's distributive assumption on z-scores false, one needs some reference which can help identify truly important attributes from non important ones. The solution proposed through the Boruta algorithm arises from the spirit of random forest, i.e. adding more randomness to the system. Essentially, the algorithm creates a randomized copy of the original system and consequently builds a classifier for this extended system. The importance of a single attribute is assessed in comparison with those of the randomized ones. Only variables whose importance is systematically higher than randomized variables are considered important (Kursa *et al.*, 2010). The procedure is shown in Algorithm 6

The information system is always extended by at least 5 shadow attributes (line 3), even if the number of attributes in the original set is lower than 5: the values of the replicated variables are randomly permuted across objects (line 4). The added attributes are randomized so that the correlations between the replicated attributes and the outcome variable is random by design. A predefined number of random forest runs on the extended information system is performed and the z-scores are gathered, the replicated variables are randomized before each run so that the random part of the system is different for each forest (line 5). For all attributes a statistical test is performed (line 7): the null hypothesis is that importance of the single attribute is equal to the MIRA. The test is a two-sided equality test: the hypothesis may be rejected either when the importance of the attribute is significantly higher or lower than MIRA. For each attribute, one can compute how many times the importance of the attribute was higher

Algorithm 6: Boruta Algorithm

1 Set all attributes as *undetermined*;
2 **while** *importance is undetermined for all the predictors, or algorithm does not reach a predefined number of iterations* **do**
3 Extend the dataset by adding copies of all predictor variables;
4 Shuffle such variables (shadow attributes) in order to remove the correlation with the response variable;
5 Run a random forest on the extended dataset and gather the z-scores;
6 Find the maximal importance (z-score) of all randomized attributes (MIRA), and then assign a hit to every attribute that scored better than MIRA;
7 Count the number of hits collected for each attribute and then perform a binomial test: if the null hypothesis of equality is rejected, declare the predictor variables which have importance significantly higher (lower) than MIRA as *important* (*unimportant*), otherwise leave *undetermined*;
8 Remove shadow attributes.

than MIRA (a hit is recorder for the variable). The expected number of hits for N runs is $E(N) = 0.5N$ with standard deviation $S = \sqrt{0.25N}$: i.e. a binomial distribution with $p = q = 0.5$. A variable is deemed important (i.e. accepted) when the number of hits is significantly higher than the expected value (the converse is true for unimportant variables) (line 7). The procedure is repeated for a previously defined number of times or until all attributes are either rejected or conclusively deemed important. In case the algorithm reaches an end before all attributes have been confirmed or rejected, they are left tentative, which means they can be ignored in further analysis, or the number of maximal iterations can be increased in order to solve such doubtful cases.

In practice this algorithm is preceded by a warm-up phase during which the attributes are compared to the fifth, third, and second best shadow attribute, and the test for rejection is performed at the end of each initial round, while the test for confirmation is not. Such procedure is introduced to cope with the high fluctuations of z-scores when the number of attributes is large at the beginning of the whole procedure.

This procedure is implemented in the `Boruta` package in R, and has been already applied to genetic data (Kursa and Rudnicki, 2011) and seems to be quite flexible in the sense that - although as default z-scores are used - various types of variable importances from different algorithms can be used: e.g. the authors show how to implement variable importances from the package `rferns` that implements random ferns, i.e. an ensemble of Naive Bayes classifiers. It is interesting to note that in an early version of the algorithm (see Rudnicki *et al.* (2006)), the comparison of each variable importance with that of random attributes was performed through a t-test, and that was justified by the fact that - if the number of iterations in the algorithm is sufficiently large - the averages could be assumed to be normally distributed. Nevertheless, neither a given number of iterations was suggested nor was a test for normality requested for the distribution of each variable importance. It seems quite reasonable that the authors shifted from a parametric to a non parametric approach. The authors claim that the time complexity in real cases is approximately $O(p * n)$, where p is the number of variables, while n is the number of observations. We have noticed that this method is actually highly scalable, namely we have tried with some datasets, and even with $p \cong 4000$ and $n = 1000$ the whole procedure is feasible on a single

CPU (AMD V 1.4 processor, 2.3 GHz) within a few hours of computation. Such is a great result, considering the current need for parallel processing due to extremely CPU consuming task that are requested in a resampling or even in a recursive scenario: so being able to select relevant attributes on an ordinary machine like a notebook, where standard analyses can be carried out in few hours or even overnight. In most cases the default parameters of `randomForest` can be used, and can suffice in most cases since random forest performance has a rather weak dependence on its parameters. In different scenarios, one can tune the parameters `mtry` and `ntree` in order to obtain the minimum OOB error.

5.3 Gini VIM Correction Procedure

As mentioned above, one of the two random forest VI measures is the total heterogeneity reduction produced by a covariate on the response obtained by adding all the decreases of the heterogeneity in all the tree nodes where such covariate is selected. It has been known for long now that this measure - along all other so-called total decrease in node impurity measures (TDNI) (e.g. entropy) - is biased in favor of variables that offer more (potential) cutting points: e.g. have more distinct numeric values or less missing values.

As already seen, variable relevance, or importance, results as the summation, over the set J of nonterminal nodes of the tree t, of the heterogeneity reductions due to the splits made by that variable along the whole tree. Let d_{ij} be the decrease in heterogeneity allowed by X_i at the node j in J. X_i is used to split at a node j if the decrease it provides is greater than any provided by the other covariates. Thus, the variable importance of X_i for the t-th tree is defined as:

$$\widehat{\mathrm{VI}}_{X_i}(t) = \sum_{j \in J} d_{ij} I_{ij} \tag{5.2}$$

where I_{ij} is the indicator function which equals 1 if the i-th variable is used to split R_j, and 0 otherwise. Dobra and Gehrke (2001) state that a split criterion is unbiased if it is only based on the strength of the dependency between the predictor variable X_i (i.e. its importance) selected and the response variable, regardless of other characteristics of X_i. In the case of a binary response variable Y which can take on the values $Y = 0$ and $Y = 1$,

let $\mathbf{x} = (X_1, \ldots, X_p)$ denote the random vector of continuous predictors, and $D = \{y_i, \mathbf{x}_i\}_1^N$ is a sample of N independent and identically distributed observations of Y and X. For a given cutting point s due to variable X_i at a given node j, the following contingency table can be specified (as seen in Sandri and Zuccolotto (2008)):

Table 5.1: Contingency table obtained by splitting on variable X_i at node j.

	L	R	
	$X_i \leq s$	$X_i > s$	Σ
$Y = 0$	n_0	$N_0 - n_0$	N_0
$Y = 1$	n_1	$N_1 - n_1$	N_1
Σ	N_L	$N_R = N - N_L$	N

where N is the sample size at node j, N_L and N_R the number of units in the left and right nodes after splitting, while N_0 and N_1 are the number of units with response $Y = 0$ and $Y = 1$ inside the node. In this special case, the empirical Gini Index (Breiman 1984) is defined as $\hat{G} = 2\hat{p}(1 - \hat{p})$, $\hat{p} = N_1/N$ and the Gini gain, i.e. the impurity reduction obtained by splitting through variable X_i at cutpoint s, which defines the difference in impurity before and after splitting, is defined as:

$$\widehat{\Delta G} = \hat{G} - \left(\frac{N_L}{N}\hat{G_L} + \frac{N_R}{N}\hat{G}_R\right) \tag{5.3}$$

where $\hat{G_L}$ and $\hat{G_R}$ are the Gini indexes for the left and right nodes. Strobl *et al.* (2007b) indicate three important sources of selection bias imputable to such measure:

1. Estimation effects, namely:

   - The empirical Gini index underestimates the true index by a factor equal to $\frac{N-1}{N}G$ as shown below:

$$\begin{aligned} \mathrm{E}(G_j) &= \mathrm{E}\left(2\frac{N_1}{N}\left(1 - \frac{N_1}{N}\right)\right) \\ &= 2p(1-p)\,2\frac{N}{N}(1-p)) \\ &= \frac{N-1}{N}G \end{aligned} \tag{5.4}$$

Which means that the empirical Gini index is a negatively biased estimator and its bias is equal to Bias $\left(\hat{G}\right)$ = -G/N.

- The variance of $\hat{G}$ can be written as:

$$\text{Var}\left(\hat{G}\right) = 4\frac{G}{N}\left(\frac{1}{2}G\right) + O\left(\frac{1}{N^2}\right). \tag{5.5}$$

The variance of the empirical Gini index depends on the true index and increases when G moves away from its maximum value 1/2 or from its minimum value and for small sample sizes. These two effects lead to a preference of variables with many missing values.

2. A multiple comparison effect. The common problem of multiple comparison refers to the inflation of type I errors connected to the number of statistical tests performed. In the case of split selection, a type I error occurs when a variable is selected for splitting even if it is not informative. This effect leads to a preference of variables with many possible splits: with more values for numerical variables (i.e. less missing values and/or less ties), with more categories for categorical/ordinal variables. For instance, for numerical variables with no ties the possible cutpoints to be evaluated is equal to $N - 1$, a categorical variable with k categories has $2^k - 1$ possible cutpoints, while an ordinal variable with k levels offers at most $k - 1$ splitting points.

Before introducing the correction procedure, let us define the notion of (un)informative splits. Let us suppose that the predictor space $\mathcal{X}$ has been recursively split into J rectangles. If X_i and Y are stochastically independent, they continue to be independent in each rectangle R_j, while if some association exists, they could be dependent or conditionally independent in a given R_j. This means that uninformative covariates (i.e. stochastically independent of Y) remain uninformative in each subset of the sample space. Informative covariates, on the other hand, can either continue to be informative or become uninformative in a given R_j. Let us consider growing a tree using a sample N with at least one covariate associated with Y. In this case the heterogeneity reductions on informative covariates will be typically greater than those provided by uninformative ones, and thus, an informative covariate will be chosen as splitting variable. This can be defined as

an *informative split*. On the other hand, when within a node there are no informative covariates, only uninformative covariates and/or informative covariates which have become uninformative can be chosen as a splitting variable. This can be defined as an *uninformative split*.

In informative splits, the heterogeneity reduction of a splitting variable is directly connected to its importance, while in an uninformative split it is due to chance. The importance estimation of a given variable X_i can be expresses as:

$$\widehat{\mathrm{VI}}_i(t) = \sum_{j \in J_I} \hat{d}_{ij} \cdot I_{ij} + \sum_{j \in J_U} \hat{d}_{ij} \cdot I_{ij} = \hat{\mu}_i(t) + \varepsilon_i(t) \tag{5.6}$$

where J_I and J_U are the sets of nodes where informative and uninformative splits take place, $\hat{\mu}_i(t)$ is the part of importance measure due to informative splits and thus directly connected to the true importance of X_i, while $\varepsilon_i(t)$ is a noise term due to uninformative splits and thus a bias source.

Let us now describe the correction procedure proposed by Sandri and Zuccolotto (2008) based on the use of pseudocovariates. Consider a sample of the data $(\boldsymbol{Y}, \boldsymbol{X})$, where $\boldsymbol{Y}$ is the response variable of dimension $N \times 1$ and $\boldsymbol{X}$ is the set of covariates of size $N \times p$. We permute the rows of $\boldsymbol{X}$ obtaining another set of variables $\boldsymbol{Z} = \{Z_1, \ldots, Z_p\}$ - called pseudocovariates - which are added to the original set of p covariates $\boldsymbol{X}$. The addition of $\boldsymbol{Z}$ does not affect informative splits, for they only compete in uninformative splits. Let $\widehat{\mathrm{VI}}_{X_i}(\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{Z})$ be the measure of the importance of X_i which derives from a tree-based ensemble on the extended dataset $(\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{Z})$. The following algorithm is based on the assumption that for each (un)informative covariate X_i, there is a corresponding pseudovariable Z_i, which has the same probability of winning the competition in uninformative splits. For the above

reasons, after a large enough number of replications R, the quantity

$$\widehat{\mathrm{VI}}^{*}_{X_i} = \frac{1}{R}\sum_{r=1}^{R}\left(\widehat{\mathrm{VI}}^{(r)}_{X_i}(\boldsymbol{Y},\boldsymbol{X},\boldsymbol{Z}) - \widehat{\mathrm{VI}}^{(r)}_{Z_i}(\boldsymbol{Y},\boldsymbol{X},\boldsymbol{Z})\right) \tag{5.7}$$

can be used as an unbiased VIM for X_i. The algorithm for the correction of TDNI measures in tree-based ensembles is the following.

Algorithm 7: TDNI measure correction Algorithm

```
for R times do
    Create a set of pseudocovariates Z either permuting the values of
    a single variable X_i or permuting the rows of X ;
    Run a tree-based ensemble on the extended dataset (Y, X, Z);
    Compute variable importance measures for each X_i and Z_i;
Compute the unbiased importance measure VI*_{X_i} for each predictor
as in equation (5.7);
```

While one either decides to permute each X_i or to permute the rows of $\boldsymbol{X}$, the pseudocovariates are stochastically independent of Y and each Z_i has the same distribution of its corresponding X_i, while in the latter case the sample multiple relationships existing among the p variables in $\boldsymbol{X}$ are left unchanged when creating the corresponding pseudovariables. According to simulation studies, Sandri and Zuccolotto (2008) suggest this latter.

Chapter 6

Selection of Clinical Variables

6.1 Prediction Purposes and Dataset Description

The dataset used in this work comes from a study carried out by Pattaro *et al.* (2007). The MICROS study is a population-based survey on three small, isolated villages, characterized by: old settlement, small number of founders, high endogamy rates, slow/null population expansion. The dataset we have used is a result of screening questionnaires, clinical measurements, blood and urine samples, and DNA collection. We have focused our analysis on the prediction of the glomerular filtration rate (GFR).
Such measure is accepted as the best overall measure of kidney function (Stevens *et al.*, 2006), and its variation is a crucial determinant of renal function. GFR is usually estimated through equations which include variables such as age, sex, race, and body size, in addition to serum creatinine, as surrogates for muscle mass. Some of these include the Cockcroft-Gault formula (Cockcroft and Gault, 1976), the Modification of Diet in Renal Disease (MDRD) study equation (Levey *et al.*, 1999), and the CKD-EPI (Chronic Kidney Disease Epidemiology Collaboration) formula (Levey *et al.*, 2009). In the MICROS study, GFR based on serum creatinine was estimated through the MDRD Study equation.

Some observations had to be removed because of missing values on the response variable, while missing values of clinical variables were imputed using the quick method relying on a measure of location (`na.roughfix` in

the `randomForest` package), while missing values of SNPs had previously been resolved through imputation methods using HapMap (Johnson *et al.*, 2008; Gibbs *et al.*, 2003). The resulting dataset has 1201 observations, 68 clinical variables, and ~ 300000 genetic variables (i.e. SNPs) [1].

The following analyses are carried out for clinical variables only, the same variable selection methods will be carried out on genetic variables as well, and then the results put together in order to judge the joint performance of a random forest classifier. Tables 6.2 and 6.3 show the clinical variables used in this section. Note that all variables are numeric except the following categorical variables: DM (cat:2), sex (cat:2), and village (cat:3). Genetic variables are single-nucleotide polymorphisms (SNPs), which are ordered categorical variables taking on values 0-1-2. Namely, the homozygous dominant genotype is coded by 0, the heterozygous genotype by 1, and the homozygous recessive genotype by 2.

Table 6.1: Datasets used throughout this work

dataset	dimensions		variables
X	$n \times p$	$p = 290319$	SNPs
X_1	$n \times p_1$	$p_1 = 2048$	SNPs
X_2	$n \times p_2$	$p_2 = 50$	SNPs
X_3	$n \times q$	$q = 68$	clinical
X_4	$n \times q_1$	$q_1 = 28$	clinical
X_5	$n \times (p_2 + q_1)$	$(p_2 + q_1) = 78$	clinical+SNPs

As shown in Table 6.1, the datasets in this work can be described as follows (all contain $n = 1201$ samples):

1. X is the dataset containing all the SNPs to be used in the screening phase;

2. X_1 is the dataset containing the SNPs selected after the screening procedure, and used in the feature selection algorithms;

3. X_2 is the dataset containing the union of the SNPs resulting from the

[1] Note that the dataset featured individuals related through a multigenerational pedigree: therefore a linear mixed model using a kinship matrix (as explained in the Appendix) was used, and the new response variable is the residuals from such fit.

feature selection algorithms;

4. X_3 is the dataset containing the clinical variables used in the feature selection algorithms;

5. X_4 is the dataset containing the union of the clinical variables resulting from the feature selection algorithms;

6. X_5 is the dataset containing both the clinical variables and SNPs resulting from the feature selection algorithms.

Table 6.2: List of clinical variables 1:30

symbol	description
age	age
basophil	basophils
bmi	body mass index
calcium	calcium
chlor	chloride
cholest	cholesterol
dbp	diastolic bloodpressure
DM	diabetes mellitus
eosinoph	eosinophils
ferritin	ferritin
ft4	free thyroxine
ggt	gamma-glutamyltransferase
glucose	glucose
got	glutamic oxaloacetic transaminase
gpt	glutamate pyruvate transaminase
hct	hematocrit
hdl	high-density lipoprotein
height	height
hgb	hemoglobin
inr	international normalized ratio
kalium	potassium
ldl	low-density lipoprotein
lymphoz	lymphocytes
mch	mean corpuscular hemoglobin
mchc	mean corpuscular hemoglobin concentration
mcv	mean corpuscular volume
monoz	monocytes
mpv	mean platelet volume
natrium	sodium
neutroph	neutrophil granulocytes

Table 6.3: List of clinical variables 31:68

symbol	description
neutroph	neutrophil granulocytes
pct	procalcitonin
pdw	platelet distribution width
plt	platelet hematocrit
pttime	activated partial thromboplastin time
pttratio	activated partial thromboplastin time ratio
pttsec	prothrombintime
rbc	red blood cell count
rdw	red blood cell distribution width
rrpuls	pulse pressure during RR interval
sbp	systolic bloodpressure
schmitz_adiponectin	adiponectin
schmitz_albumin	albumin
schmitz_apo_ai	apolipoprotein A1
schmitz_apo_b	apolipoprotein B
schmitz_apo_ci	apolipoprotein C1
schmitz_apo_cii	apolipoprotein C2
schmitz_apo_ciii	apolipoprotein C3
schmitz_apo_e	apolipoprotein E
schmitz_cicp	procollagen I C-Terminal Propeptide
schmitz_crp	c-reactive protein
schmitz_ictp	carboxyterminalTelopeptide of Type I Collagen
schmitz_ige	immunoglobulin E
schmitz_igf_1	insulin-like growth factor1
schmitz_iron	iron
schmitz_leptin	leptin
schmitz_nt_probnp	n-terminal pro b-type natriuretic peptide
schmitz_opg	osteoprotegerin
schmitz_srankl	soluble receptor activator of nuclear factor kappa-B ligand
schmitz_stfr	soluble transferrin receptor
schmitz_transferrin	transferrin
schmitz_urea	urea
sex	sex
trigly	triglycerides
tsh	thyroid-stimulating hormone
uric_acid	uric acid
village	village
wbc	white blood cells
weight	weight

6.2 Random Forest Recursive Feature Elimination

In the following section, we used the feature selection algorithm described in section 5.1 to dataset X_3 as explained in Table 6.1. We have iteratively fitted a random forest dropping a single variable (least ranking) at each iteration until only two were retained (in order to retain a multivariate approach). Such procedure was feasible due to the size of the feature space, which is already quite small (i.e. 68 variables). This procedure was encapsulated inside a ten-fold cross validation which was repeated ten times. A single iteration took around 16 minutes on a single CPU. Subsequently, the root mean squared error (RMSE) was computed for each subset solution along with its correspondent standard deviation.

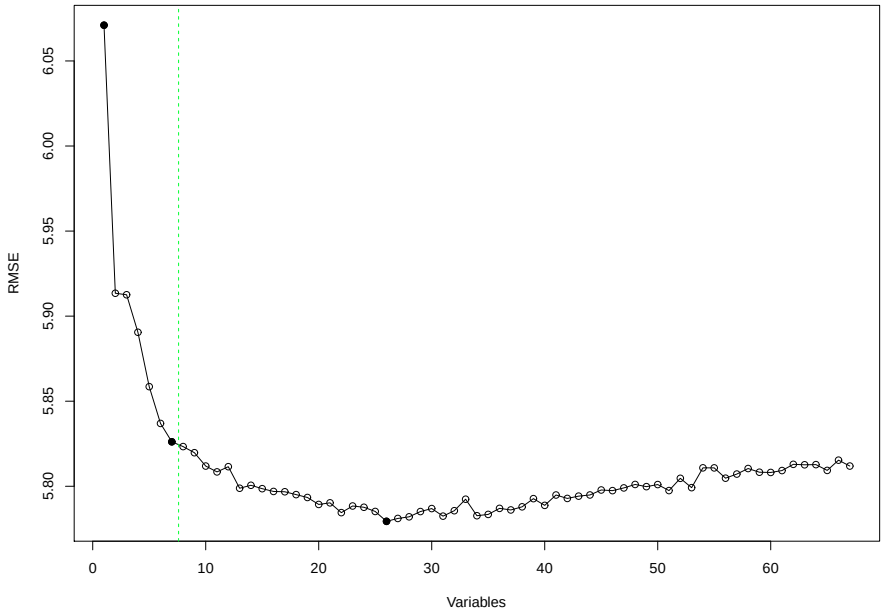

Figure 6.1: RMSE for RF-RFE obtained for a 10-fold cross-validation repeated ten times. At each step a single variable is dropped.

The smallest value of RMSE (5.779, see Figure 6.2) is reached at 27 variables, but moving within one standard deviation we can obtain the smallest subset available (i.e. two variables). Taking the union of all the two-variable solutions, we obtain a subset of four variables, namely age,

schmitz_nt_probnp, schmitz_urea, and uric_acid, with inclusion probability of 100%, 65%, 34%, and 1% respectively. This result falls within the framework of aggressive variable selection, as proposed in (Díaz-Uriarte and De Andres, 2006). Although the 1 s.e. approach is indeed a consolidated practice, and usually provides a good approach to standard variable selection, there are cases in which it provides too restrictive a solution, as it is in the current case, where a single variable (i.e. age) has got an effect on GFR so big that - in terms of prediction error - we would not be able move to a richer solution in terms of number of variables selected. Inspecting the error plot in Figure 6.2, one could argue that up to eight variables there is a consistent gain in error, and after that *elbow* the addition of one single variable does not provide a reduction big enough to justify the inclusion of other variables.

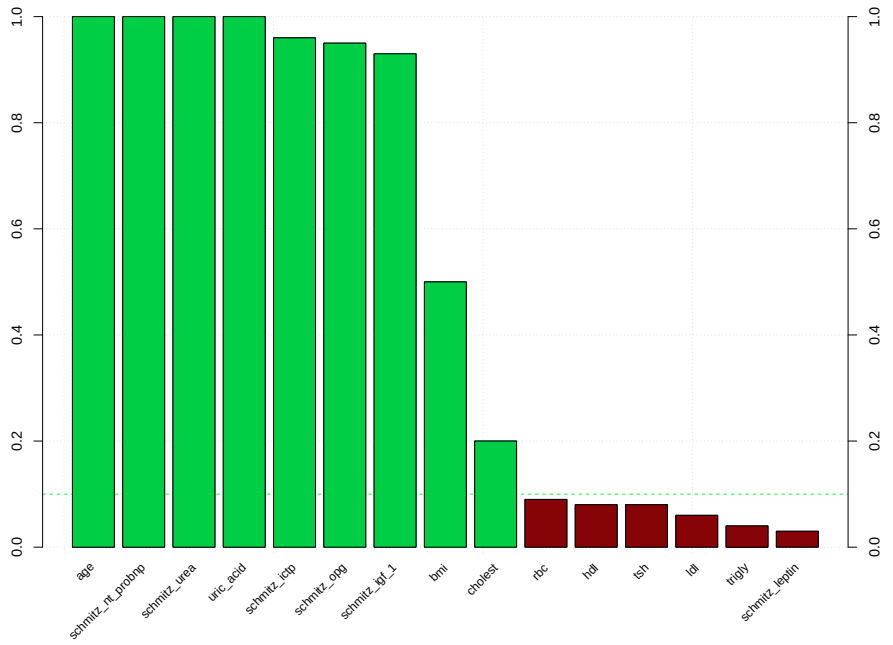

Figure 6.2: Inclusion Probability for Clinical Variables

Thus the solution with eight variables was selected. Subsequently, we computed the selection frequency of each of the 68 variables, i.e. how many times (out of 100) each clinical variable was selected in the eight-variable solution. The barplot in Figure 6.2 summarizes the inclusion prob-

ability of the following variables being selected: age (100%), schmitz_urea (100%), schmitz_nt_probnp (100%), uric_acid (100%), schmitz_ictp (96%), schmitz_opg (95%), schmitz_igf_1 (93%), bmi (50%), cholest (20%), rbc (9%), hdl (8%), tsh (8%), ldl (6%), trigly (4%), schmitz_leptin (3%). Selecting a threshold at 10% (i.e. we select as important variables that in the resampling have been selected among the first six more than ten out of 100 times) we can discard: rbc (9%), hdl (8%), tsh (8%), ldl (6%), trigly (4%), schmitz_leptin (3%). The procedure allows to eliminate 86.76% of initial variables.

6.3 Gini VIM Correction Procedure

In the following section, we use the feature selection algorithm described in section 5.3 to dataset X_3 as explained in Table 6.1. The plot of Figure 6.3 shows a Gini correction procedure carried out with 1000 iterations: we have identified a group of three highly important (dark green: age [not shown in plot due to its large VIM value], schmitz_nt_probnp, schmitz_urea,), a group of three middle important (green: schmitz_igf_1, schmitz_opg, uric_acid), and a group of four slightly important clinical variables (light green: ft4, bmi, cholest, schmitz_ictp).

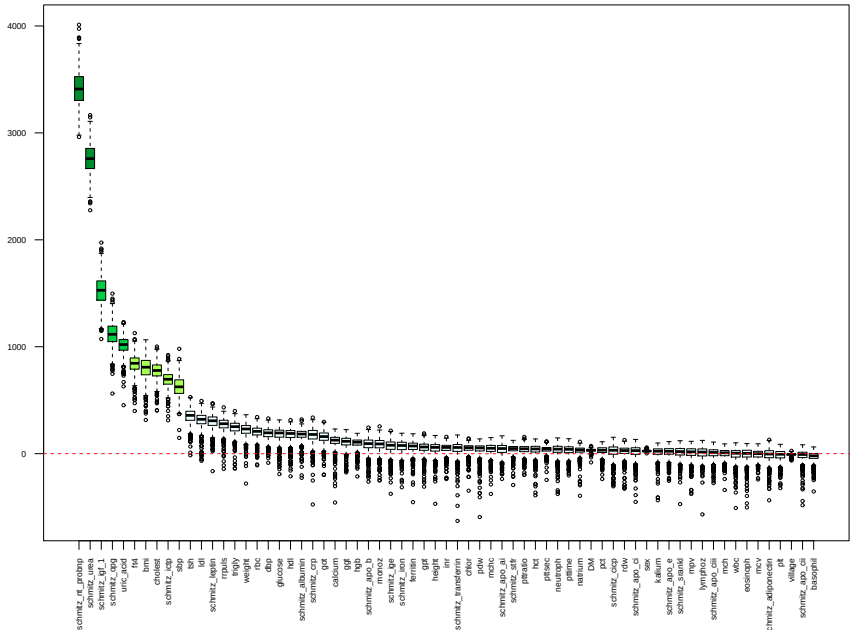

Figure 6.3: Gini Correction for clinical variables

The resulting subset is small (eleven variables) with more than 83% of variables removed.

6.4 Feature Selection with the `Boruta` Package

In the following section, we apply the feature selection algorithm described in section 5.2 to dataset X_3 as explained in Table 6.1.

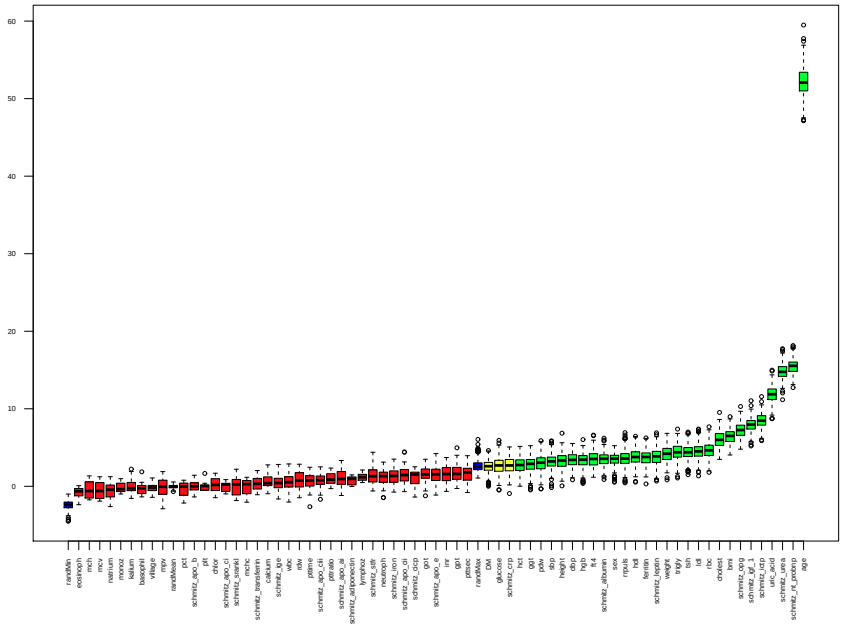

Figure 6.4: Boruta Selection for clinical variables

The plot 6.4 shows the distribution of the z-scores obtained within a run of the Boruta algorithm. `maxRuns` (i.e. the number of Random Forest to produce) had to be increased to 400 in order to resolve doubtful cases. Nonetheless, three variables are left as tentative (DM, glucose, schmitz_crp) because they did not seem to be resolved increasing the parameter `maxRuns`. The resulting subset encompasses a total of 28 variables.

6.5 Results

Table 6.4 shows the results of the selection of clinical variables, displaying how many times (out of three) a variable was selected along with each of the methods used. The table columns show the variable names, the name of each feature selection algorithm used (i.e. Boruta, RF-RFE, Gini Correction) indicating whether a given variable was selected by the corresponding method (1=selected, 0=not selected), # indicates the number of feature selection algorithms that selected a given variable (ranging from 1 to 3).

Table 6.4: Summary of the selection of clinical variables

variable name	#	Boruta	RF-RFE	Gini Correction
age	3	1	1	1
bmi	3	1	1	1
cholest	3	1	1	1
schmitz_ictp	3	1	1	1
schmitz_igf_1	3	1	1	1
schmitz_nt_probnp	3	1	1	1
schmitz_opg	3	1	1	1
schmitz_urea	3	1	1	1
uric_acid	3	1	1	1
ft4	2	1	0	1
sbp	2	1	0	1
dbp	1	1	0	0
ferritin	1	1	0	0
ggt	1	1	0	0
hct	1	1	0	0
hdl	1	1	0	0
height	1	1	0	0
hgb	1	1	0	0
ldl	1	1	0	0
pdw	1	1	0	0
rbc	1	1	0	0
rrpuls	1	1	0	0
schmitz_albumin	1	1	0	0
schmitz_leptin	1	1	0	0
sex	1	1	0	0
trigly	1	1	0	0
tsh	1	1	0	0
weight	1	1	0	0

Further in this work, these clinical variables will be used, along with the SNPs selected in the following sections, in order to verify the prediction performance of Random Forest using clinical variables with the addition of SNPs.

Chapter 7

Selection of SNPs

In the following lines, we provide a framework that takes advantage of a multiple use of the random forest algorithm as both a screening phase as well as a multiple variable selection process.

We propose the use of a two-step procedure based on random forest.

1. A screening phase:

   (a) Apply a single run of random forest with the Random Jungle package, using the $n \times p$ dataset X.

   (b) Collect three different importance measures (VIM) for each SNP, namely the Gini, permutation, and Meng VIM.

   (c) Select a prespecified number of SNPs according to each ordered VIM, with or without the help of a screeplot. The number of variables to retain depends on both computational capabilities at hand and on the desired number of overall subset.

   (d) Aggregate the results of each VIM, i.e. we take the union of the three sets of variables, and create the $n \times p_1$ dataset X_1 ($p_1 \ll p$) to be processed in the following phase.

2. A feature selection phase:

   (a) Perform finer feature selection with algorithms with a manageable subset of p_1 variables deriving from the screening phase.

(b) Select a smaller subset of p_2 informative variables ($p_2 \ll p_1$) by means of three feature selection algorithms.

(c) Compare results to standard GWA studies.

7.1 Screening Phase

Firstly we have applied Random Jungle to the original dataset X.

Since it is advisable to tune the number of variables randomly selected at each node, we have performed various random forests with `mtry` = $\lceil p/10 \cdot (1, \ldots, 9) \rceil$, where $\lceil \cdot \rceil$ denotes the largest integer, with a relatively small number of trees, i.e. `ntree` = 5000. For each `mtry` value, we recorded the values of accuracy and error, as shown in Table 7.1.

Table 7.1: `mtry` values

percentage	mtry	accuracy	error
10%	29032	0.0702769	0.9297231
20%	58064	0.0709461	0.9290539
30%	87096	0.0724675	0.9275325
40%	116128	0.0703509	0.9296491
50%	145160	0.0724536	0.9275464
60%	174192	0.0724203	0.9275797
70%	203224	0.0702812	0.9297188
80%	232256	0.0703163	0.9296837
90%	261288	0.0713772	0.9286228

The `mtry` value providing the best solution in terms of prediction error is $p/10 \cdot 3$. Thus, this value was used to produce a random forest with `ntree` = 50000 (accuracy=0.0707269, error=0.929273). The values of the prediction error show that `ntree` = 5000, even if small compared to the number of SNPs, was already large enough for the error rates to stabilize. Such results are compatible with those provided by Goldstein *et al.* (2010). As already stated, larger values of `ntree` are however advisable in order to obtain more reliable variable importance measures (Díaz-Uriarte and De Andres, 2006).

The preliminary tuning of random forests took around fifteen days using 28 CPUs and ~ 15 GB of RAM on a machine with 32 CPUs and 64GB of RAM, while the random forest with $p/10 \cdot 3$ and `ntree` = 50000 took around twelve days on the same machine.

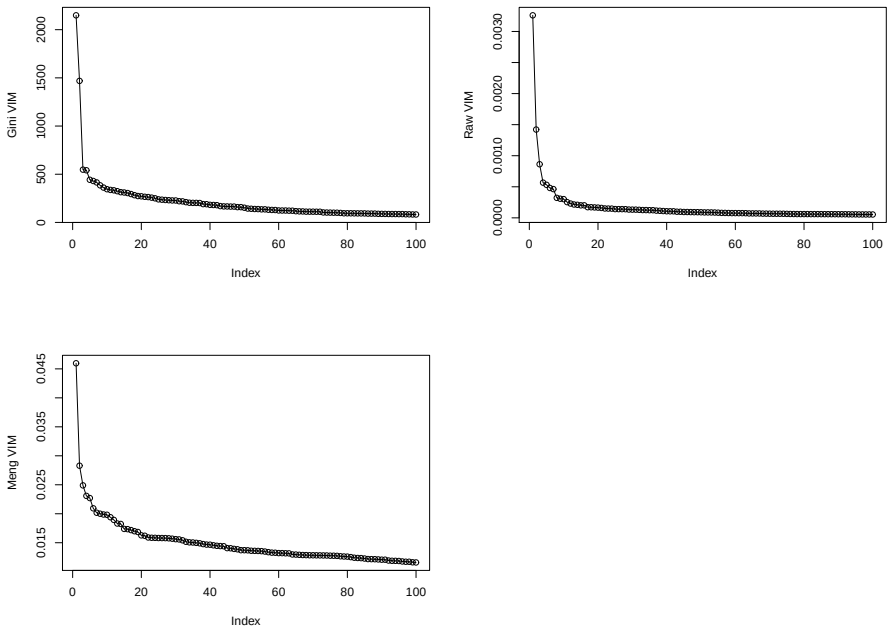

Figure 7.1: Plots showing variable importances for the highest ranking 100 variables: Gini VIM (top left), Permutation VIM (top right), and Meng Permutation VIM (bottom).

Through a visual inspection of the screeplots in Figure 7.1 of the ordered variable importance measures, we considered that as for the Raw VIM an elbow was noticed around 10 SNPs, as for the Gini VIM around 15, while for the Meng VIM around 20 (see Figure 7.1).

In order for this phase to act as a screening procedure, we retain 1000 SNPs for each VIM, resulting in a final subset of SNPs equal to $p_1 = 2013$, due to the fact that the three subset were not disjoint. This screening phase can be seen as a two-step variable selection procedure as explained in Xin and Zhu (2012), i.e. ranking and thresholding. The authors are of the opinion that ranking is more important than thresholding. From a decision-theoretic point of view, once the variables are ranked, the choice of the decision threshold has more to do with ones prior belief of how sparse the model is likely to be. To this subset, we decided to add another 35 SNPs that were found to be associated in previous studies and could be relevant (Pattaro *et al.*, 2010, 2012; Böger *et al.*, 2011), the final subset being $p_1 = 2048$.

7.2 Feature Selection Phase

7.2.1 Random Forest Recursive Feature Elimination

In the following section, we used the feature selection algorithm described in section 5.1 to dataset X_1 as explained in Table 6.1. We have iteratively fitted a random forest dropping 20% of (least ranking) SNPs at each iteration until only two SNPs were retained (in order to maintain a multivariate approach), such fraction was chosen because it provides a good compromise between refinement of results and computational costs (Díaz-Uriarte and De Andres, 2006). This procedure was encapsulated inside a ten-fold cross validation which was repeated ten times, as suggested in Granitto *et al.* (2006). Each of the 100 iterations took around 50 minutes on a single CPU. Subsequently the root mean squared error (RMSE) for each subset solution was computed, along with the correspondent standard deviation.

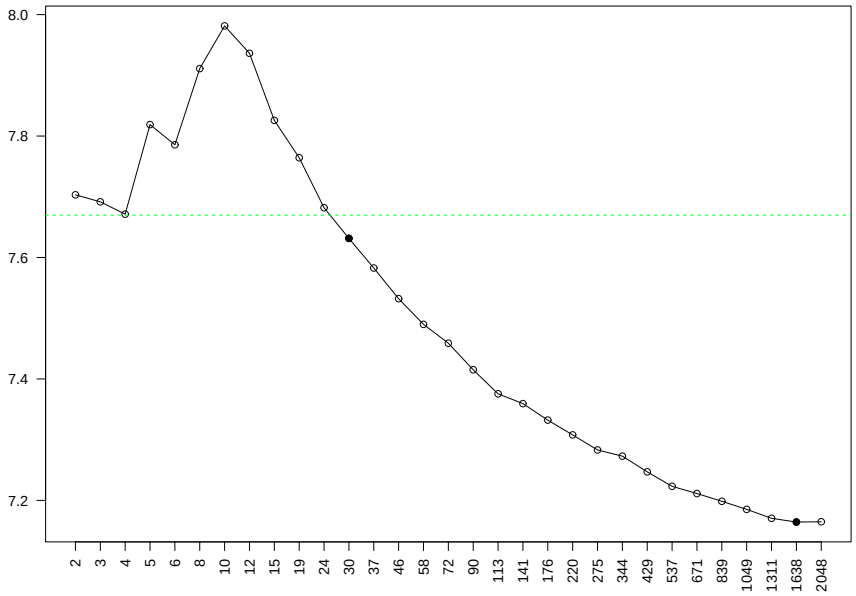

Figure 7.2: RMSE for RF-RFE obtained for a 10-fold cross-validation repeated ten times. At each step 20% of SNPs are dropped: on the x-axis the number of SNPs used is shown.

The smallest value of RMSE (7.164, see Figure 7.2) was reached at 1638 SNPs, but moving within one standard deviation we could select 30 SNPs. Within the 30-SNPs solution, the inclusion frequency of each SNP was computed and only those SNPs that were selected more than 10 out of 100 resamplings were retained. That allowed us to retain 24 SNPs of the initial 2048. This result falls within the framework of aggressive variable selection, as proposed in (Díaz-Uriarte and De Andres, 2006), for it allows us to remove 98.53% of variables and to retain a small subset.

The barplot 7.3 summarizes the inclusion probability of the first 35 mostly selected SNPs - note that 946 variables were selected in the 24-variable solution, most of which have an inclusion probability of less than 5%. As we did for clinical variables, we retain the SNPs that were selected in more than 10 out of 100 resamplings, and that allowed us to retain 24 of the initial 2048 variables.

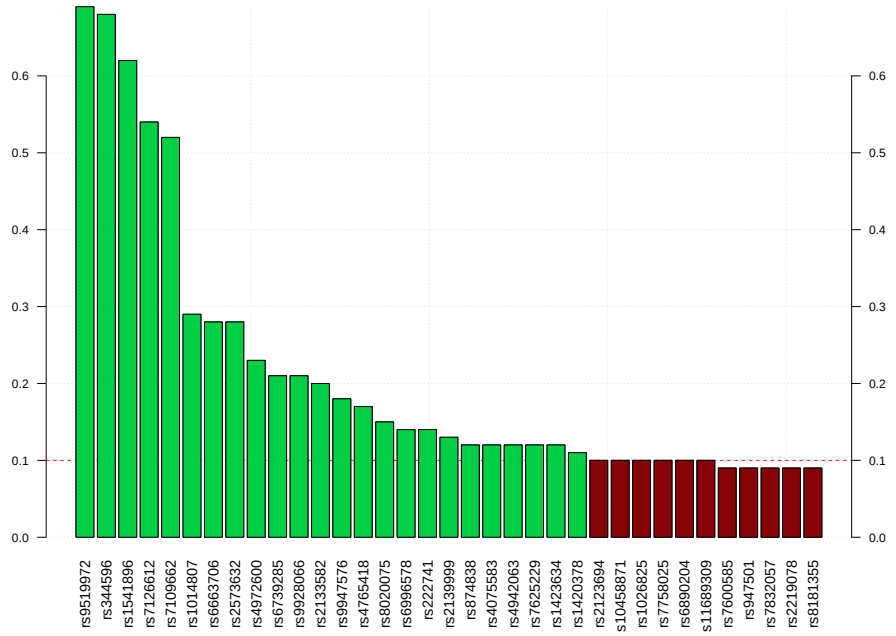

Figure 7.3: Inclusion Probability for SNPs.

7.2.2 Gini VIM Correction Procedure

In the following section, we used the feature selection algorithm described in section 5.3 to dataset X_1 as explained in Table 6.1. Figure 7.4 shows the Gini VIM Correction with 100 iterations (only the first 100 variables are plotted), where the each boxplot represents the distribution of the difference between the GINI VIM of variable X_i and its correspondent randomized replicate Z_i.

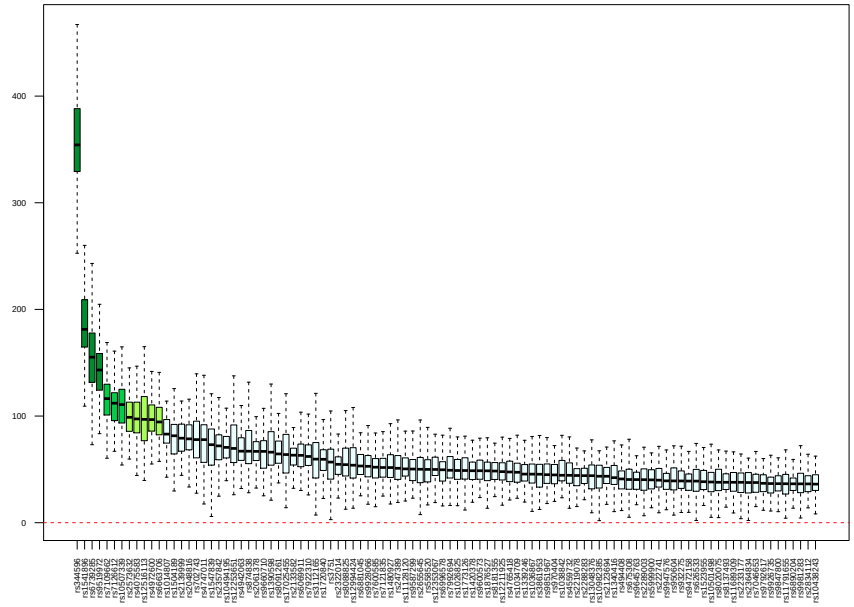

Figure 7.4: Gini Correction for SNPs.

The correction algorithm for the Gini VIM was carried out using $S = 100$ iterations. The Gini VIM of each SNP was subtracted to the Gini VIM of the corresponding pseudo-SNP (i.e. the permutation of the SNP values). The distribution of such differences results in a boxplot for every SNP.

Such results were plotted, and the selection itself was conducted relying on the resulting screeplot.

It is quite immediate to notice that a group of variables come out from the multitude: particularly we have identified a group of four highly important (dark green), a group of three middle important (green), and a group of five slightly important SNPs (light green). The results of such selection

is a subset of 12 SNPs being chosen.

7.2.3 Selection through Boruta Algorithm

In the following section, we used the feature selection algorithm described in section 5.2 to dataset X_1 as explained in Table 6.1. Initial applications of the algorithm showed that the maximal number of random forest runs (`maxRuns`) had to be increased from 100 (default) to 800 in order for the algorithm to provide a satisfactory solution. Such procedure performed 830 random forest runs (some additional iterations are needed for a warm-up phase), and took ~13 hours on a single CPU. 37 SNPs were confirmed important, while five were left tentative, meaning that the algorithm could not decide whether they were important or not, but they did not seem to be resolved further increasing the `maxRuns` parameter. Therefore, we would discard these five SNPs from further analysis. It is of interest to note that `Boruta` was easily applicable to a relatively large dataset in relative small times without the use of multithreading. Computational times may change dramatically if shadow attributes are never dropped from the covariates set (parameter `light` =FALSE).

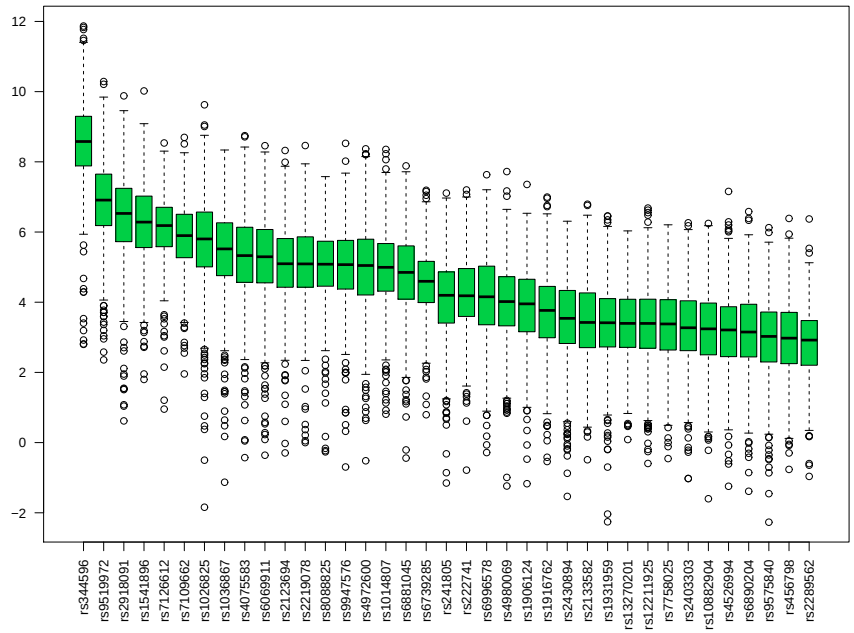

Figure 7.5: z-score distribution of attributes marked as confirmed in the Algorithm

Figure 7.5 shows the distribution of the z-scores of the SNPs declared important across the random forest iterations.

7.3 Results

Each feature selection algorithm selects a different number of SNPs, while some SNPs were selected by all three methods, some others were selected by two of them, while others are unique to each method, providing an overall subset of 50 SNPs.

It is interesting to note, that - while most SNPs are not correlated - some other are; e.g. SNPs rs7109662 and rs7126612 have a Cramer's V coefficient equal to 0.978. Consequently, we decided to retain rs7126612 for further study, because it is the one of the two having the lesser average correlation with other selected SNPs.

Similarly, SNPs rs2219078 and rs2123694 (retained) have a Cramer's V coefficient equal to 1, which means that each of the two works as a replicate for the other, thus, any of the two can be retained for further analysis.

In order to state the relevance of such selection procedures, we have compared the performance of Random Forest (iterated 1000 times) with the SNPs selected by each feature selection algorithm to a Random Forest (iterated 1000 times) with 50 SNPs sampled at random from the initial 290319 SNPs, as shown in Table 7.2. All SNPs is a subset of 50 SNPs resulting from the union of the subsets obtained from the three feature selection algorithms.

Table 7.2: Number of SNPs used for each RF model.

	All SNPs	Boruta	RF-RFE	Gini	Random
number of SNPs	50	37	24	12	50

For each of the five subsets, we have performed 1000 random forest iterations and computed the distribution of the RMSE. The RMSE is often suggested as the correct measure to determine the best model. It is usually to prefer over MSE because RMSE is measured in the same units as the data, rather than in squared units (Ratner, 2011).

The plot 7.6 shows that

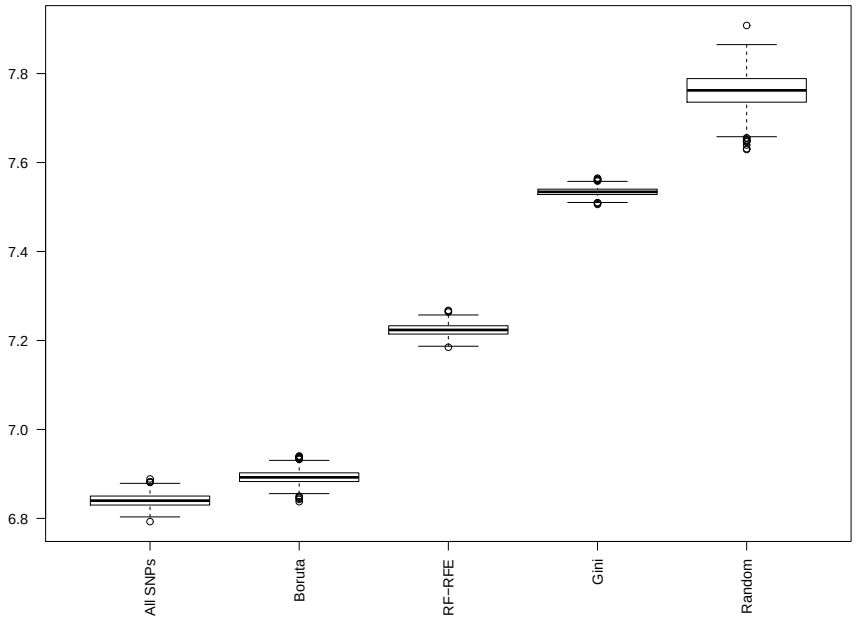

Figure 7.6: RMSE for different SNPs selections

1. 50 randomly selected SNPs provide the worst predictive performance (mean RMSE=7.761),
2. the SNPs selected by the Gini Correction algorithm provide a slightly more accurate predictor (mean RMSE=7.534),
3. the SNPs selected by the RF-RFE algorithm increase the prediction to a greater extent (mean RMSE=7.224),
4. the SNPs selected by the Boruta algorithm provide the most accurate predictor (mean RMSE=6.893),
5. the SNPs selected by aggregating all results provide slightly better results (mean RMSE=6.840, Welch test p-value 2.2e-16).

These results are summarized in table 7.3.

Table 7.3: Average RMSE comparison for different SNPs selections

All SNPs	Boruta	RF-RFE	Gini	Random
6.840	6.893	7.224	7.534	7.761

Such results show that the Boruta algorithm performed better in both computational costs and in terms of selecting the SNPs that mostly contribute to explaining the outcome. The computing times of all methods are summarized in Table 7.4.

Table 7.4: Computational Times of feature selection algorithms.

	Boruta	RF-RFE	Gini Correction
number of iterations	-	100	100
single iteration time (minutes)	-	~50	~150
total time (minutes)	-	~5000	~150000
total time (hours)	~13	~83	~250

It is however important to state that such results should be checked on different datasets, in order to have honest estimates of the RMSE. This is a key point in our future research, which will be devoted to assess (i) the prediction performance of these SNPs in an independent population sample and (ii) the biological relevance of the newly identified SNPs. It is of interest to note that there is some coherence with the results provided from the

previous study in Pattaro *et al.* (2007). Although none of the SNPs could be claimed to be statistically significant because they could not exceed genome-wide significance, it should be noted that some of the SNPs ranking high in terms of p-value were detected in the RF approach. For instance SNP rs663706 is the one ranking highest with p-value = 0.000007, rs1906124 ranking ninth with p-value = 0.000042, and rs6881045 ranking 23rd with p-value = 0.000094. This information is shown in Tables 7.6 and 7.7: The table columns show the SNP names, the name of each feature selection algorithm used (i.e. Boruta, RF-RFE, Gini Correction) indicating whether a given SNPs was selected by the corresponding method (1=selected, 0=not selected), # indicates the number of feature selection algorithms that selected a given variable (ranging from 1 to 3), while p-value and rank columns show the p-value and the rank resulting from the GWAS in Pattaro *et al.* (2007). This shows some coherence with regard to some SNPs, and may suggest that interaction effects are relevant for prediction purposes, while marginal main effects are in general too weak.

We have later compared the predictive ability of previously selected 28 clinical variables (i.e. using dataset X_4 as explained in Table 6.1), mostly containing parameters measured from blood and urine samples, in order to investigate if the addition of SNPs (i.e. using dataset X_5 as explained in Table 6.1) could provide an increase in prediction compared with the use of clinical variables only. We then performed 1000 random forest iterations and computed the distribution of RMSE and compared this statistic to 1000 random forest iterations containing both clinical variables and SNPs.

Figure 7.7 shows the distribution of the RMSE's of the prediction based on the solely clinical variables and with the addition of the fifty SNPs selected. The results are summarized in table 7.5 and show that there is a gain in terms of RMSE, while figure 7.8 shows the distribution of the RMSE's compared through kernel estimation, plus graphical reference band for equality of a bootstrap based test for the two distributions, which led to the rejection of the equality of the two distributions. This was performed with the `sm` package in R as described in (Bowman and Azzalini, 1997).

Moreover, it is interesting to check the ranking given to each SNP according to the three different VIMs and the inclusion frequency in the trees created in the first iteration of the RF algorithm with the Random Jungle

Table 7.5: Average RMSE comparing the two subsets.

Clinical Variables and SNPs	Clinical Variables
5.691	5.731

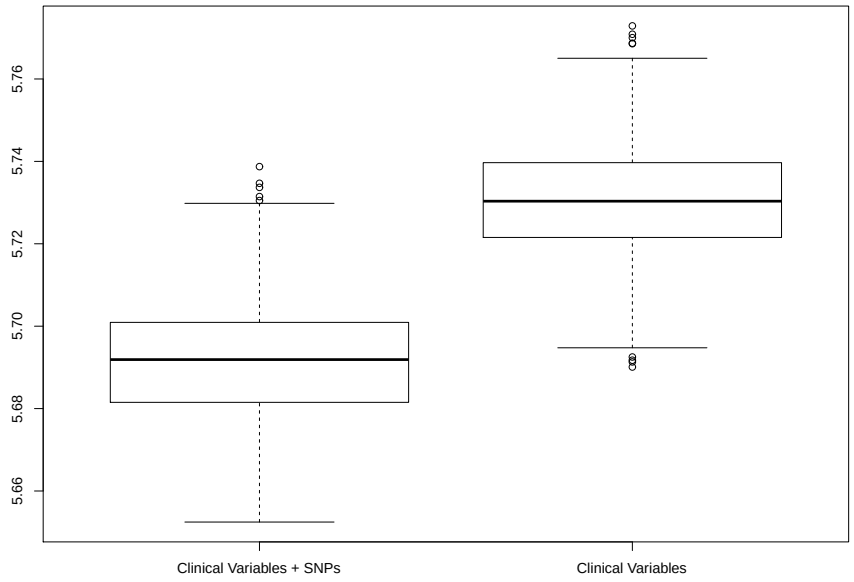

Figure 7.7: RMSE for clinical variables and clinical variables with the addition of the 50 selected SNPs

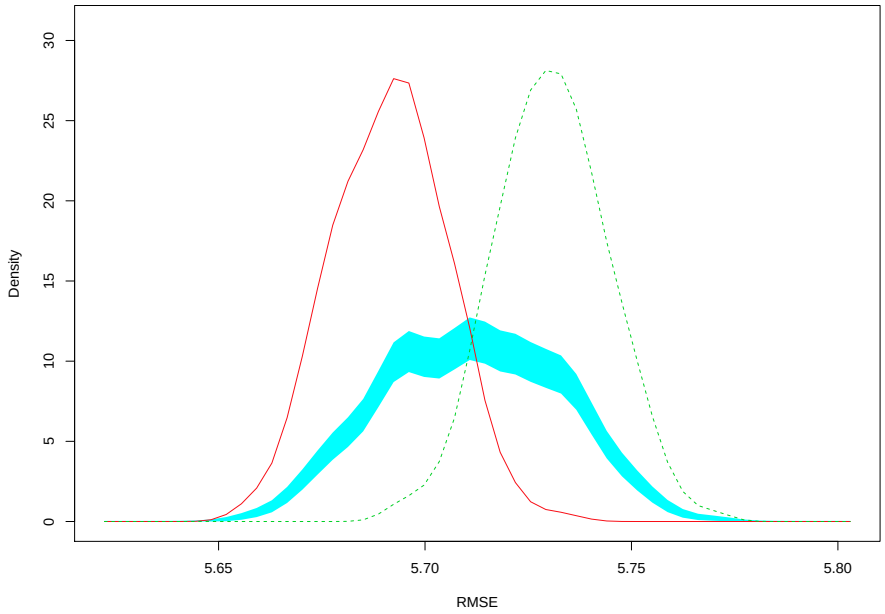

Figure 7.8: RMSE distribution for clinical variables (dotted cline) and for clinical variables with the addition of SNPs (solid line)

Table 7.6: SNPs selected 3/2 times plus p-values and GWAS(MICROS) rank in regression

SNP ID	#	Boruta	RF-RFE	Gini-Correction	p-value	rank
rs1541896	3	1	1	1	0.531508	154952
rs344596	3	1	1	1	0.401354	117026
rs4075583	3	1	1	1	0.061444	18150
rs4972600	3	1	1	1	0.02872	8552
rs6739285	3	1	1	1	0.12274	36043
rs7109662	3	1	1	1	0.372704	108662
rs7126612	3	1	1	1	0.987029	286556
rs9519972	3	1	1	1	0.081051	23846
rs1014807	2	1	1	0	0.162126	47579
rs2133582	2	1	1	0	0.040343	11997
rs222741	2	1	1	0	0.155381	45663
rs2573632	2	0	1	1	0.038378	11449
rs6663706	2	0	1	1	7.09E-06	1
rs6996578	2	1	1	0	0.312977	91222
rs9947576	2	1	1	0	0.30967	90235

Table 7.7: SNPs selected once plus p-values and GWAS(MICROS) rank in regression

SNP ID	#	Boruta	RF-RFE	Gini-Correction	p-value	rank
rs1026825	1	1	0	0	0.415659	121134
rs1036867	1	1	0	0	0.008986576	2706
rs10507339	1	0	0	1	0.3928927	114557
rs10882904	1	1	0	0	0.3844381	112125
rs12211925	1	1	0	0	0.2076489	60777
rs12516113	1	0	0	1	0.197702	57882
rs13270201	1	1	0	0	0.003679912	1074
rs1420378	1	0	1	0	0.003684297	1077
rs1423634	1	0	1	0	0.1197732	35148
rs1906124	1	1	0	0	0.0000416	9
rs1916762	1	1	0	0	0.002373611	686
rs1931959	1	1	0	0	0.09719026	28554
rs2123694	1	1	0	0	0.2333583	68132
rs2139999	1	0	1	0	0.2248291	65699
rs2219078	1	1	0	0	0.2267321	66241
rs2289562	1	1	0	0	0.7109454	207058
rs2403303	1	1	0	0	0.3844272	112123
rs241805	1	1	0	0	0.03046019	9098
rs2430894	1	1	0	0	0.2806196	81911
rs2918091	1	1	0	0	0.01081076	3259
rs4526994	1	1	0	0	0.005391562	1604
rs456798	1	1	0	0	0.3560212	103760
rs4765418	1	0	1	0	0.5898113	171844
rs4942063	1	0	1	0	0.3815121	111225
rs4980069	1	1	0	0	0.02746785	8222
rs6069911	1	1	0	0	0.000419317	135
rs6881045	1	1	0	0	0.0000937	23
rs6890204	1	1	0	0	0.00440025	1273
rs7625229	1	0	1	0	0.0634134	18720
rs7758025	1	1	0	0	0.08595233	25331
rs8020075	1	0	1	0	0.07778159	22865
rs8088825	1	1	0	0	0.08477194	24967
rs874838	1	0	1	0	0.5639512	164308
rs9575840	1	1	0	0	0.5185735	151092
rs9928066	1	0	1	0	0.1283263	37663

Package. Tables 7.8 and 7.9 show the ID of each of the 50 SNPs selected, along with the rank for each VIM (Raw, Gini, Meng). Inclusion frequency indicates the frequency with which a given SNP was included in the Random Forest used in the screening phase using the Random Jungle package.

Table 7.8: Inclusion frequency greater than or equal to 0.1 in trees in RJ and rankings

SNP ID	Raw rank	GINI rank	MENG rank	inclusion frequency
rs344596	2	1	134	0.1350
rs6663706	1	2	3	0.1309
rs874838	11	3	954	0.0424
rs10507339	27	7	3340	0.0386
rs1541896	16	9	1214	0.0369
rs6069911	4	13	15	0.0326
rs1420378	8	17	155	0.0315
rs6739285	54	10	4436	0.0267
rs7126612	73	30	7096	0.0257
rs2573632	18	15	647	0.0253
rs1036867	9	28	80	0.0244
rs7109662	92	33	7853	0.0236
rs13270201	41	38	1358	0.0200
rs6996578	122	40	7938	0.0193
rs2918091	32	39	594	0.0186
rs12516113	144400	36	158977	0.0183
rs8088825	49	31	1428	0.0177
rs9519972	64	64	1753	0.0151
rs6881045	19	66	93	0.014120
rs4075583	107	115	1683	0.010940
rs4972600	217	110	5876	0.010160

Inspecting the rankings in each variable importance measure, it is clear that the contribution of the Meng VIM was not important in the screening phase in the light of the feature selection that led to these fifty SNPs. The ranks given by this VIM are usually large, oftentimes larger than 1000, which was the threshold selected for each VIM: a rank larger than 1000 means that that particular VIM did not contribute to the screening phase, i.e. that specific SNP was not selected according to that VIM but only according to one or two other VIMs. It is also important to note that, in those case when Meng VIM is less than 1000, it is always larger than at least one of the other VIMs, this means that the screening of SNPs based

Table 7.9: Inclusion frequency less than 0.1 in trees in RJ and rankings

SNP ID	Raw rank	GINI rank	MENG rank	inclusion frequecy
rs9947576	98	97	1180	0.009960
rs4942063	206	111	3351	0.008060
rs4980069	188	178	2064	0.007140
rs2403303	115	171	590	0.006900
rs222741	276	179	3448	0.006560
rs2289562	445	131	7105	0.005740
rs9928066	287	199	2921	0.005720
rs2139999	286	231	2675	0.005520
rs2123694	279	267	2270	0.005460
rs1026825	572	355	10016	0.005400
rs2133582	488	270	7194	0.005320
rs1906124	364	294	3804	0.005260
rs2219078	178	279	771	0.005200
rs4526994	209	318	1079	0.005080
rs456798	342	491	2146	0.004340
rs1014807	233	381	824	0.004240
rs6890204	2424	837	35130	0.003740
rs8020075	336	479	1127	0.003440
rs10882904	449	667	1999	0.003220
rs9575840	245016	407	187217	0.003180
rs2430894	4635	684	44619	0.002940
rs1423634	945	1088	6339	0.002860
rs7625229	506	1056	1573	0.002680
rs7758025	1393	827	10374	0.002660
rs1931959	1208	752	8049	0.002620
rs4765418	823	1169	4150	0.002600
rs12211925	4497	913	30348	0.002260
rs1916762	632	2675	421	0.001400
rs241805	2391	17320	109	0.000340

on the Meng VIM led to a redundancy in terms of relevant SNPs. These SNPs have always smaller ranks on the two other VIMs, which means that they would have been selected anyway without the screening phase based on this VIM. This could depend on the fact that in this specific dataset there was no substantial linkage disequilibrium. Generally speaking, we suggest to include the screening based on all three VIMs, or to make sure that linkage disequilibrium is not an issue in the particular dataset at hand.

At last, for the purpose of completeness we report a summary of the 50 SNPs selected. Tables 7.10 and 7.11 show the summary provided by the `summary` function of the `GeneABEL` package in R. They show the ID of each SNP (column SNPID), the chromosome position of each of the selected SNPs (columns Chromosome and Position), the type of the two alleles measured (i.e. A, T, C or G, columns A1 and A2), Q.2, which provides the minor allele frequency (MAF) of a given SNP, i.e. the frequency of the less frequent B allele, Pexact, which refers to exact p-value for the test of Hardy-Weinberg equilibrium (HWE), and Fmax, which is a parameter describing what is the direction of deviation from HWE.

Table 7.10: SNPs summary (1:25)

SNPID	Chromosome	Position	A1	A2	Q.2	Pexact	Fmax
rs1014807	9	70492016	T	C	0.169	0.0027	0.0868
rs1026825	18	58971255	A	G	0.440	0.9548	0.0022
rs1036867	13	75362516	T	C	0.402	0.5615	0.0161
rs10507339	13	24184564	G	A	0.050	0.5485	0.0141
rs10882904	10	99105673	A	G	0.345	0.3873	-0.0248
rs12211925	6	87166330	C	T	0.145	0.3660	0.0261
rs12516113	5	139862830	A	G	0.044	0.5103	-0.0270
rs13270201	8	71731645	T	G	0.052	0.0017	0.1023
rs1420378	2	171338877	C	A	0.324	0.1109	0.0447
rs1423634	5	87131861	A	G	0.299	0.0831	-0.0492
rs1541896	18	37638408	C	T	0.155	0.1985	-0.0395
rs1906124	9	11807216	C	T	0.215	0.0036	0.0832
rs1916762	2	224999036	G	A	0.187	0.3576	0.0268
rs1931959	13	73916739	C	T	0.190	0.5260	0.0178
rs2123694	2	108223937	A	G	0.214	0.7396	-0.0121
rs2133582	12	24912378	A	G	0.162	0.2595	-0.0331
rs2139999	18	30272073	A	G	0.162	0.2564	0.0320
rs2219078	2	108241630	G	A	0.214	0.7396	-0.0121
rs222741	17	3455629	A	G	0.193	0.4724	-0.0232
rs2289562	11	20075983	G	T	0.109	0.1509	-0.0425
rs2403303	11	18652983	T	C	0.396	0.6407	0.0129
rs241805	20	50812838	G	A	0.062	0.8129	-0.0125
rs2430894	18	52568589	A	G	0.256	0.7699	-0.0092
rs2573632	15	98217286	G	A	0.078	0.8485	0.0012
rs2918091	10	132946118	A	G	0.100	0.0594	0.0550

Table 7.11: SNPs summary (26:50)

SNPID	Chromosome	Position	A1	A2	Q.2	Pexact	Fmax
rs344596	19	6545274	A	G	0.083	0.3611	0.0229
rs4075583	15	61127280	A	G	0.368	0.7185	-0.0119
rs4526994	15	21307741	G	A	0.242	0.1952	-0.0384
rs456798	20	30669432	G	T	0.405	1.0000	0.0004
rs4765418	12	125658925	T	G	0.238	0.3551	-0.0275
rs4942063	13	40989334	C	T	0.096	0.8734	-0.0081
rs4972600	2	172689891	A	G	0.301	0.1445	-0.0419
rs4980069	10	80815272	A	G	0.163	0.1009	0.0462
rs6069911	20	54870125	T	C	0.103	0.5447	-0.0220
rs6663706	1	117633695	T	G	0.194	0.3711	0.0249
rs6739285	2	104723749	C	A	0.097	0.0242	-0.0625
rs6881045	5	124208320	A	G	0.324	0.0256	0.0626
rs6890204	5	137846077	C	T	0.456	0.2157	-0.0354
rs6996578	8	138249241	G	A	0.078	0.8485	0.0012
rs7109662	11	6569808	T	C	0.077	0.6942	-0.0181
rs7126612	11	6567139	A	G	0.071	1.0000	-0.0063
rs7625229	3	134679961	G	A	0.308	0.1495	-0.0410
rs7758025	6	129959932	C	T	0.168	0.6168	-0.0172
rs8020075	14	49205467	A	G	0.174	0.6980	0.0109
rs8088825	18	11502551	T	G	0.078	1.0000	-0.0096
rs874838	2	85275202	A	G	0.079	0.2454	0.0326
rs9519972	13	105628162	C	T	0.140	0.9076	-0.0075
rs9575840	13	84701898	C	T	0.082	0.8533	0.0032
rs9928066	16	72754692	T	C	0.172	0.6941	0.0122
rs9947576	18	23813740	A	G	0.146	0.9107	-0.0081

Chapter 8

Rule Extraction

In the following pages, we will focus on extracting information for the purposes of interpreting the relation between the outcome variable and the set of covariates. This is carried out using methods naturally embedded in random forests, i.e. variable importance measures and partial dependence plots, as well as two novel approaches, namely the Fitting Estimation method, and the Node Harvest estimator.

8.1 Partial Dependence Plots

It can be useful to asses how each predictor is related to the response of interest. This can be achieved through partial dependence plots, as proposed by Friedman (2001). Such plots provide graphical interpretation of the mapping function $F(\mathbf{x})$ (as illustrated in Chapter 2) as a function of the predictors, and provide a strong interpretation tool, although such visualization is limited to low dimensions. As the name suggests, one such visualization provides a graphical tool showing how the value of one (or two) predictor(s) influences the values taken on by the response, leaving all other predictor values unchanged. Functions of a single real valued variable can be plotted as a graph of the averaged values of $F(\mathbf{x})$ against each corresponding value of a predictor X_i. Functions of single categorical predictors are represented through a a barplot, each for every level of the predictor, where the height of the barplot corresponds to the averaged value of the function. For quantitative response variables (and also for ordered categorical variables), the units represented on the y-axis are the natural units of

the response, while for categorical variables they are not (see Berk (2008) for insights). Single partial dependence plots can be carried out through the `partialPlot` function in the `randomForest` package in R. The resulting graphs show on the x-axis the deciles of the predictor distribution. For tree-based approaches like random forest, one proceeds as described in Algotithm 8:

Algorithm 8: Partial Dependence Plot Algorithm

for *each predictor* x_i *of interest, which has* υ *different values on the training data* **do**
  for *each* υ *value taken by* x_i*, create a new data set where* x_i *only takes on that value, leaving all other predictor unchanged* **do**
    Predict the response using random forests, for each of the υ datasets. There will be a single value averaged over all observations;
    Average each of these predictions over the trees;
  Plot the average prediction for each value of the υ datasets against the υ values of x_i;

8.2 Variable Importances

Although graphically representing the relationship between a particular predictor and the outcome variable is of great importance, it is also advisable to investigate such relationships in the light of how much a given predictor impacts on the response. Although in the previous section we have largely used variable importance measures, it may not be clear how to compare the importance measures of a given variable selection procedure. Thus, it is advisable to recompute variable importance measures on the final subset of variables. Due to the underlying variability of results, it is always advisable to compute VIMs inside an iterative scheme, in order to obtain some kind of distribution, and not only a single value, that could either over- or under-estimate the importance of a predictor.

Because checking each VIM individually could be quite chaotic, we sug-

gest the use of a simple procedure that allows us to create a single ranking of all the variables at hand, e.g. a principal component analysis (PCA). PCA is a well known dimensionality reduction technique which is usually carried out for higher dimensional problems, but the following procedure could be carried out in cases where a larger number of VIM's are computed. This was suggested by the use of PCA inside the framework of variable selection in Sandri and Zuccolotto (2006). Given a large enough number of iterations to compute VIM's, we take the median value in order to take outliers into account, and compute a PCA using the median VIM values of the three measures seen above.

8.3 Fitting Estimation

As previously stated, ensemble methods provide accurate predictions but leave interpretation needs unsatisfied. Ensemble methods are often considered as *black-boxes*, i.e. one cannot explore the overall mechanism with which the covariates influence the outcome variable (considering both interaction and main effects). The idea behind the Fitting Estimation procedure is the following: the presence of noise compromises the approximation of the function f during the building of binary recurse partitioning algorithm like CART, but such noise can be eliminated through a complex algorithm. This leads to a two step procedure (Manisera *et al.*, 2012), namely, first fitting the data with an algorithmic procedure - e.g. tree-based ensembles - providing a good estimation of f, secondly a CART is constructed using the fitted values $\hat{y}_i = \hat{f}(\mathbf{x}_i)$ instead of the original y_i's. This idea is not new, but until the contribution of Manisera *et al.* (2012), this procedure had always been treated as an heuristic method: the authors provide a theoretical background, showing the reasons why this approach can be successful. The resulting CART conveys two outcomes: a tree-based interpretation of the function $\hat{f}$ generated by so-called black-box algorithms, and unbiased variable importance measures, because the noise - and the resulting bias in VIM's resulting from unimportant splits - have been eliminated after the fit of a complex algorithm. This concept is summarized in Algorithm 9.

Formally, observed values are defined as $y_i = f(\mathbf{x}_i) + \varepsilon_i$, where ε_i is the irreducible error component present in the training data, and $f(\mathbf{x})$ is the data generating process. The predicted values $\hat{y}_i = \hat{f}(\mathbf{x}_i)$ are evaluations

Algorithm 9: Fitting-Estimation Procedure

1 Fit the training data $\{y, \mathbf{x}\}_1^N$ through a tree-based ensemble algorithm;
2 Take the fitted values $\hat{y}_i$ and replace the original y_i's, as to build a new dataset $\{\hat{y}, \mathbf{x}\}_1^N$;
3 Grow a CART using the training set $\{\hat{y}, \mathbf{x}\}_1^N$;
4 Provide a graphical representation or an explicit definition of the resulting tree;
5 Compute variable importance measures [Optional].

of f at given points of the input space. The noise component ε_i induces overfitting in the predictions in many known algorithms, as it is the case in CARTs when uninformative splits take place. We have $y_i = \hat{f}(\mathbf{x}_i) + e_i$ where e_i is the prediction error. If we let $\delta_i = \hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i)$ be the error committed by the algorithm in approximating f, then we can define the y_i's as $y_i = f(\mathbf{x}_i) + \delta_i + e_i$, so that $e_i = \varepsilon_i - \delta_i$, which means that the prediction error depends both on the noise component and the error committed by algorithm in approximating f. Let $\hat{\sigma}_e^2$, $\hat{\sigma}_\delta^2$, and $\hat{\sigma}_\varepsilon^2$ be the variances of the e_i's, δ_i's, and ε_i's. Hopefully the algorithm we are using in step 1 in algorithm 9 provides a good fit of training data, such goodness of fit is ensured by a low value of $\hat{\sigma}_\delta^2 \left(\ll \hat{\sigma}_\varepsilon^2\right)$. For in step 2 of the algorithm the fitted values $\hat{y}_i$'s are used to build a CART and $\hat{y}_i = f(\mathbf{x}_i) + \delta_i$, there is still some noise in the new training data, yet the variance of the noise component $\hat{\sigma}_\delta^2$ is less than that in the original data, i.e. $\hat{\sigma}_\delta^2 \ll \hat{\sigma}_\varepsilon^2$.

8.4 Node Harvest

Node harvest (Meinshausen, 2010) is a method with the aim of reconciling two features of trees and tree ensembles, i.e. interpretability versus and high predictive accuracy. The idea behind it is to generate a large number of nodes in a random fashion, and to pick the right number of nodes by choosing node weights, which result from a quadratic programming problem with linear inequality contraints. The algorithm is presented in the following lines. Let $\mathbf{y} = \{y_1, \ldots, y_n\}$ be a vector of size $n \times 1$ including observations of a real-valued response variable, and $\mathbf{X}$ be the $n \times p$-dimensional covariate

matrix from a domain $\mathcal{X}$, where $\mathbf{x}_i$ is i-th observation for $i = 1, \ldots, n$. Let $\mathcal{Q}$ be a collection of q nodes, where a node $Q_g \in \mathcal{Q}$, $g = 1, \ldots, q$, is defined by a rectangular subspace in $\mathcal{X}$,

$$Q_g = \left\{ \mathbf{x} \in \mathcal{X} : \mathrm{x}_k \in I_k^{(g)} \text{ for } k = 1, \ldots, p \right\}, \tag{8.1}$$

at each interval $I_k^{(g)}$ is a subset of the support of the k-th covariate. If each leaf node in a tree is an element of $\mathcal{Q}$, the partition provided by a tree can be expressed by a weight vector $\mathbf{w} = \{0, 1\}^q$, where $w_g = 1$ ($w_g = 0$) if that node g is (not) used in the partition. The prediction $\hat{Y}(\mathbf{x})$ at a point $\mathbf{x} \in \mathcal{X}$ is defined as:

$$\hat{Y}(\mathbf{x}) = \sum_{g=1}^{q} \mu_g \mathbf{1}\{\mathbf{x} \in Q_g\} \mathbf{w}_g, \tag{8.2}$$

where $\mathbf{1}$ is the indicator function which equals 1 if $\mathbf{x} \in Q_g$, and 0 otherwise, while μ_g is the mean over all observations into node Q_q,

$$\mu_g = \frac{\sum_{i=1}^{n} \mathbf{1}\{\mathbf{x}_i \in Q_g\} y_i}{\sum_{i=1}^{n} \mathbf{1}\{\mathbf{x}_i \in Q_g\}}. \tag{8.3}$$

The predictions on the n observed samples can be rewritten as $\mathbf{M}$, where $\mathbf{M}$ is the $n \times q$-dimensional matrix, with rows entries for $i = 1, \ldots, n$ given by

$$\mathbf{M}_{ig} = \begin{cases} \mu_g, & \text{if } \mathbf{x}_i \in Q_g \\ 0, & \text{if } \mathbf{x}_i \notin Q_g \end{cases} \text{ for } g = 1, \ldots, q = |\mathcal{Q}|. \tag{8.4}$$

The empirical squared loss on the training samples is then

$$\|\mathbf{y} - \mathbf{M}\mathbf{w}\|^2 \tag{8.5}$$

Trees try to pick a partitioning that minimizes this empirical squared loss under certain complexity contraints (e.g. the number of observations in each node). In tree ensembles, predictions are weighted averages over the

output of all trees in the ensemble, and each observation is allowed to be part of more than one node. For instance, in random forests each of the m trees receives a weight equal to $1/m$. If all leaf nodes of the random forest are part of the set $\mathcal{Q}$, the vector of weights $\mathbf{w}_g \in \left\{0, \frac{1}{m}, \frac{2}{m}, \ldots, 1\right\}$ instead of binary weights $\mathbf{w}_g = \{0, 1\}$ for single trees. If a node appears once, its corresponding weight is $1/m$, while if it appears more than once its weight is the corresponding multiple of $1/m$ (up to a maximum of 1 if it appears in every tree). The node harvest estimator is obtained through:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{y}\text{-}\mathbf{M}\mathbf{w}\|^2 \text{ such that } \mathbf{I}\mathbf{w} = \mathbf{1} \text{ and } \mathbf{w} \geq \mathbf{0} \tag{8.6}$$

where

$$\mathbf{I}_{ig} = \begin{cases} 1, & \text{if } \mathbf{x}_i \in Q_g \\ 0, & \text{if } \mathbf{x}_i \notin Q_g \end{cases} \text{ for } g = 1, \ldots, q. \tag{8.7}$$

is the $n \times q$ matrix indicating whether or not an observation i, with $i = 1, \ldots, n$, falls into a given leaf. The constraint $\mathbf{I}\mathbf{w} = \mathbf{1}$ requires each observation to be part of exactly one selected node. Similarly to tree ensembles, the weights across all nodes for a single observation have to sum up to 1. The second constraint $\mathbf{w} \geq \mathbf{0}$ forces weights to take on nonnegative values only. In tree ensembles, this is achieved implicitly through the averaging process. The outcome of such process is a small subset of nodes being picked by a large initial ensemble of nodes, since most of the nodes receive a null weight, through the constraint imposed by $\mathbf{I}\mathbf{w} = \mathbf{1}$. The prediction of a new data point $\mathbf{x} \in \mathcal{X}$ is carried out through a weighted average over all nodes it falls into:

$$\hat{Y}(\mathbf{x}) = \frac{\sum_{g \in G_{\mathbf{x}}} \hat{\mathbf{w}}_g \mu_g}{\sum_{g \in G_{\mathbf{x}}} \hat{\mathbf{w}}_g} \tag{8.8}$$

where $G_{\mathbf{x}} := \{g : \mathbf{x} \in Q_g\}$ is the collection of nodes that $\mathbf{x}$ falls into. The root node is forced to be a member of the set Q through a small weight equal to 0.001 so that the set $G_{\mathbf{x}}$ is always nonempty. This covers the case in which a new observation does not fall in any of the selected nodes. Should that be the case, the prediction of a new data point would be achieved as

the mean of the root node, i.e. the mean over all training data.

The node harvest algorithm is implemented in the `nodeHarvest` package in R. There are three parameters that affect the characteristics of the set $\mathcal{Q}$.

1. Number of nodes (`nodes`). The number of randomly generated nodes $q = |\mathcal{Q}|$ usually ranges from hundreds to thousands. Although for some datasets the performance may increase as more nodes are added to the set, default number of 1000 is recommended. Since computational times depend greatly on this parameter, one should try to use the maximum number that is computationally feasible.

2. Maximal interaction order (`maxinter`). The maximal interaction order of node Q_g is the number of variables necessary to establish whether a given observation falls into a node. Main effects are obtainable with a parameter `maxinter`=1, while two-variable interactions with a parameter `maxinter`=2 (default), while three-variable interactions with a parameter `maxinter`=3. Meinshausen (2010) suggest to keep the default value in order to keep the results as interpretable as possible, stating that it is enough to provide competitive results with many ensemble methods like random forests.

3. Minimal node size (`nodesize`). The minimal node size $\min_g |\{i : \mathbf{X}_i \in Q_g\}|$ influences the degree of smoothing. Although some datasets could be sensitive to this parameter, a `nodesize`=5 has been shown to be competitive on a wide range of different datasets.

The initial set of nodes Q could be either generated at random (i.e. without the use of a response) or through the use of a tree ensemble. Meinshausen (2010) states that results are quite insensitive to this choice, with the latter method requiring a smaller initial set of nodes. The algorithm then starts with an empty set Q, nodes are generated by trees inside the random forest algorithm, but trees are grown from subset of the training data of size $n/10$ instead of bootstrap samples to speed up computational times. All the nodes that satisfy the maximal interaction depth and minimal node size contraints are added to the set Q.

8.5 Rule Extraction for Clinical Variables

8.5.1 Partial Dependence Plots for Clinical Variables

The partial dependence plots shown in figures 8.1 and 8.2 display the relationship between each clinical variable and the response variable inside the random forest ensemble run using dataset X_4 as explained in Table 6.1. Each plot shows the value of the response value (y-axis) changing depending on the different values taken for every predictor (x-axis). Note that only sex is a categorical variable, so its corresponding partial dependence plot is rendered as a barplot for each category (0=female).

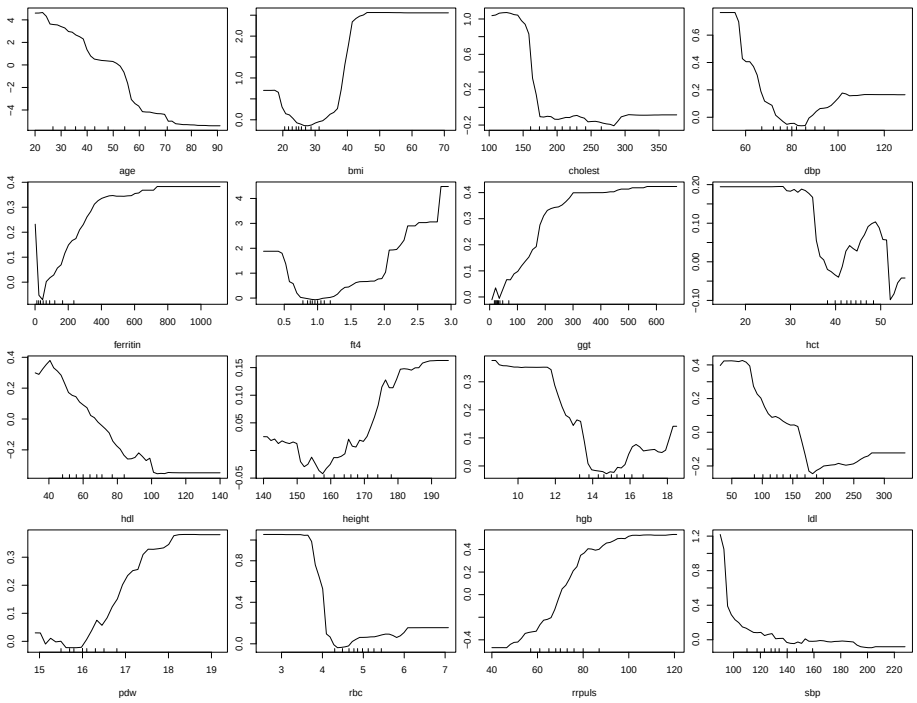

Figure 8.1: Partial Dependence Plots for clinical variables 1:16

No smoothness is imposed upon partial dependence plots, so the resulting graph is sometimes irregular. As suggested in Berk (2008), it is advisable to apply an *eyeball* smoother to interpret the results, although it is also possible to overlay any smoothing function provided by regular packages.

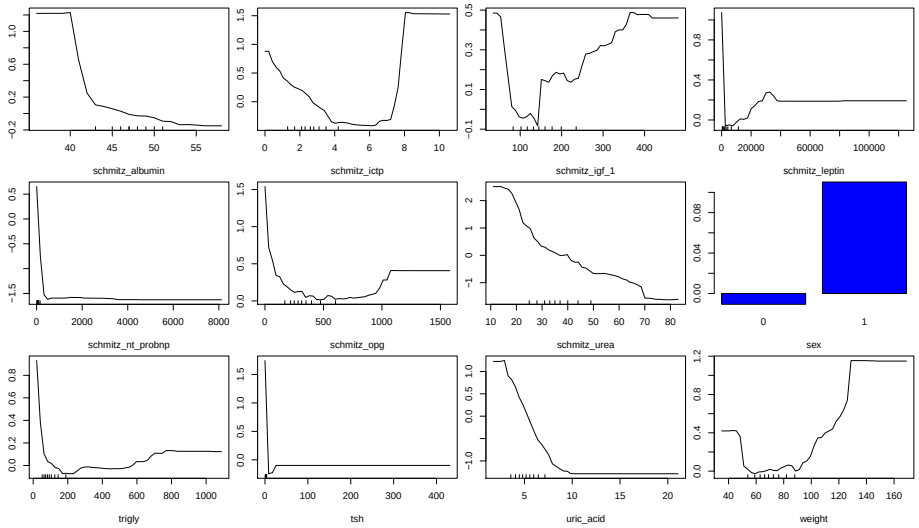

Figure 8.2: Partial Dependence Plots for clinical variables 17:28

8.5.2 Variable Importances for Clinical Variables

We recomputed three VIM's (Gini VIM, unscaled and scaled VIM) inside a hundred RF iterations, and then computed the median value for each VIM.

Table 8.1: Median VIM values for clinical variable age

	Gini	unscaled	scaled
age	17615.27	29.25	53.72

The results show - as it was clear from the previous sections - that age is the most important clinical predictor. Table 8.1 shows the median VIM values for age. Because of the large VIM values taken on by age, the resulting VIM boxplot may be of little use as show in Figure 8.3. The results provided in Figure 8.4 are a little easier to interpret.

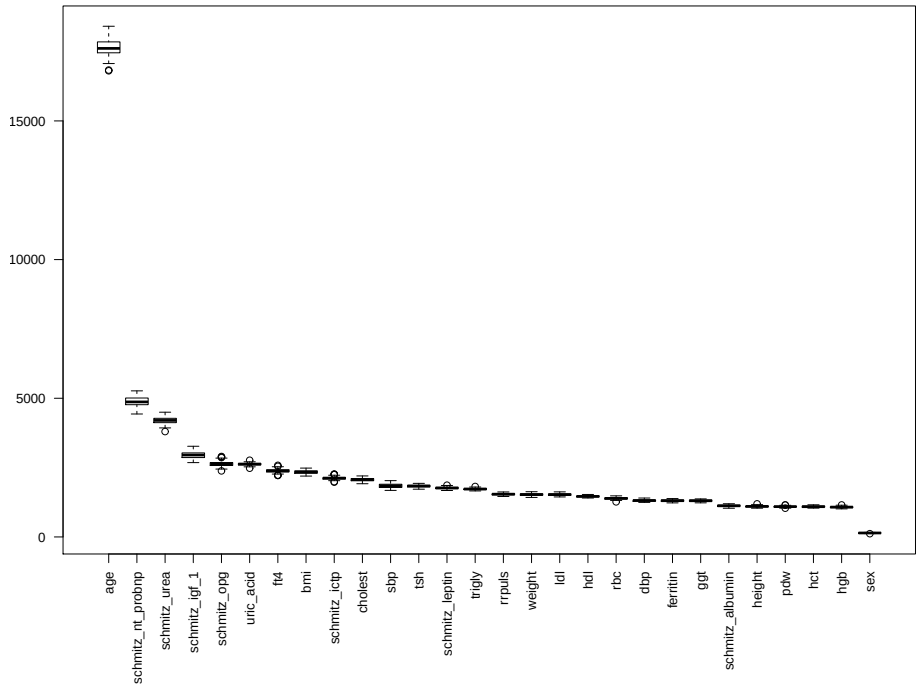

Figure 8.3: Gini VIM for clinical variables

Figure 8.5 shows the value of each clinical variable according to the first two principal components, which explain 98.27% and 1.16% of variance respectively. The great importance of variable age makes all the other clinical

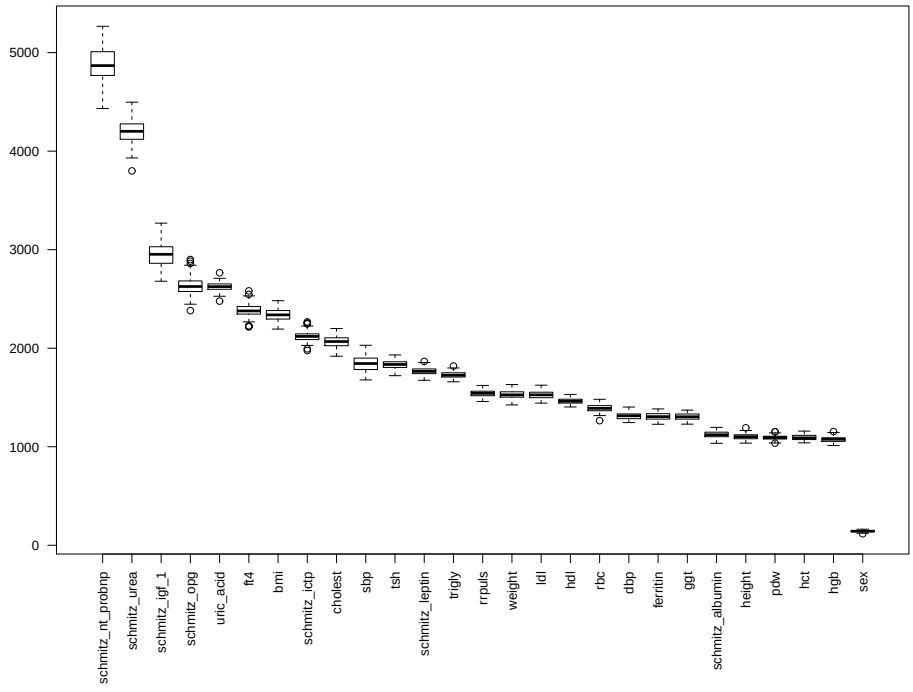

Figure 8.4: Gini VIM for clinical variables - except age

variables cluster together, i.e. age can be considered as an outlier, therefore PCA can be carried out excluding variable age, which we declare the most important among clinical variables. Figure 8.6 is clearly more readable: the first two principal components explain 93.31% and 6.31% of variance respectiverly. Although a group of the least important variables are still clustered together, other very important variables like schmitz_nt_probn and schmitz_urea stand out of the cloud. Due to the large amount of variability explained by the first principal component, it is advisable to create a single and final ranking based upon it (see Table 8.2).

Table 8.2: Ranking and PC value of clinical variables

rank	variable	PC
1	schmitz_nt_probnp	4.972
2	schmitz_urea	4.805
3	uric_acid	2.819
4	schmitz_igf_1	1.440
5	schmitz_opg	0.925
6	schmitz_ictp	0.660
7	bmi	0.566
8	cholest	0.211
9	weight	-0.377
10	trigly	-0.418
11	ldl	-0.502
12	rbc	-0.594
13	sbp	-0.633
14	ft4	-0.656
15	tsh	-0.722
16	ferritin	-0.752
17	schmitz_leptin	-0.762
18	dbp	-0.829
19	hdl	-0.878
20	hgb	-0.893
21	rrpuls	-0.921
22	hct	-0.994
23	height	-1.087
24	ggt	-1.108
25	schmitz_albumin	-1.260
26	pdw	-1.348
27	sex	-1.665

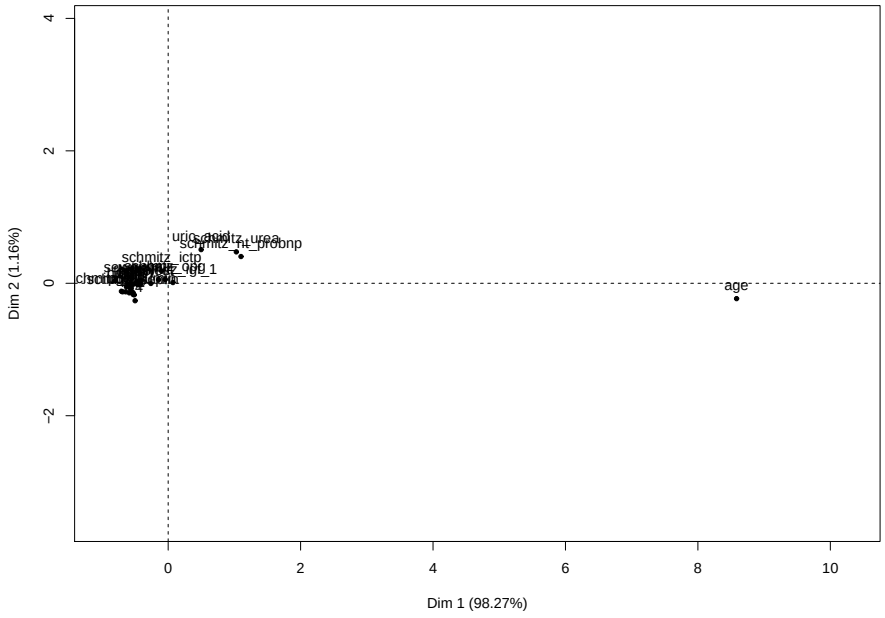

Figure 8.5: Principal Components of VIM's for clinical variables

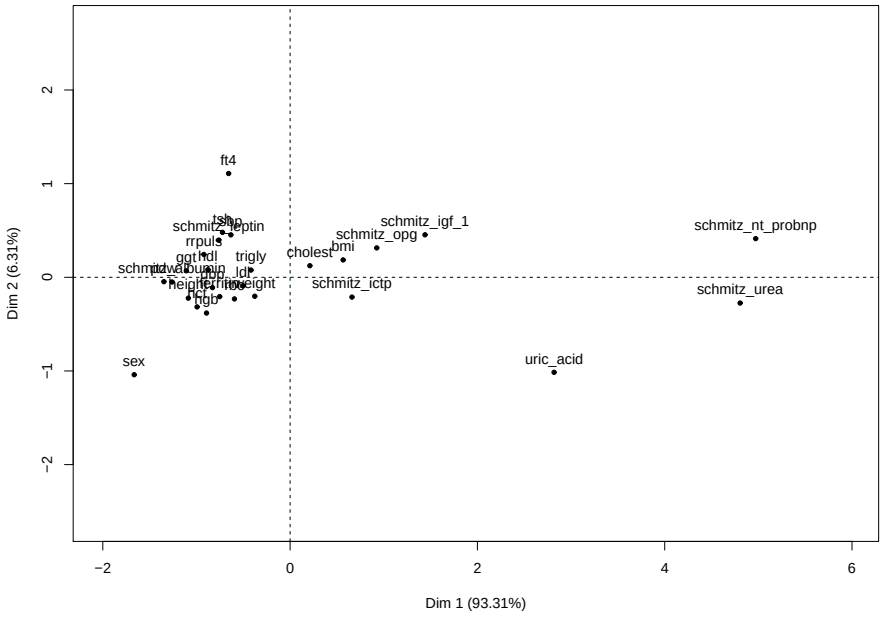

Figure 8.6: Principal Components of VIM's for clinical variables - except age

After a ranking is provided, it could be of interest to inspect the partial dependence plot in detail for some variable identified among the most important. For instance, Figure 8.7 shows the partial dependence plot for variable age. It can be seen that eGFR levels decrease while age increases, but it should also be noted that eGFR decreases quite steeply between age values 50-60. This is coherent with the result provided by many studies which have shown that the GFR declines steadily with aging (Glassock and Winearls, 2009; Macías-Núñez and López-Novoa, 2008). This decline seems to be related to the physiologic process of cellular and organ senescence and with structural changes in the kidneys.

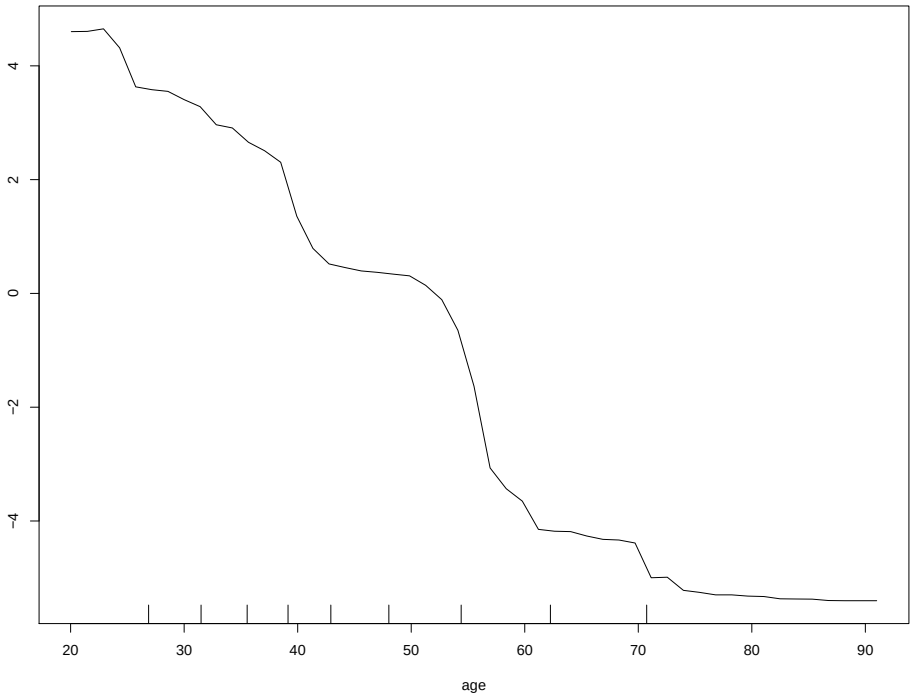

Figure 8.7: Partial dependence plot for age

The same can be done for instance for variable schmitz_urea: Figure 8.7 shows its corresponding partial dependence plot. It can be seen that eGFR levels decrease when schmitz_urea increases. eGFR rapidly decreases for small values of schmitz_urea, while then it decreases quite proportionally.

As previously mentioned, these plots can be quite irregular, therefore a proper smoothing function can be used. For instance, Figure 8.9 shows

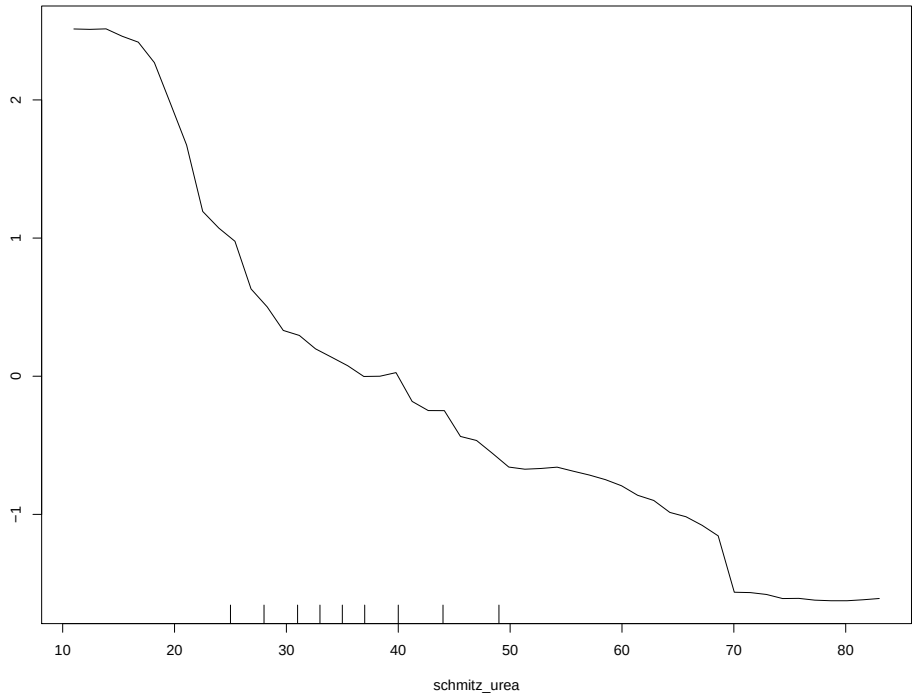

Figure 8.8: Partial dependence plot for schmitz_urea

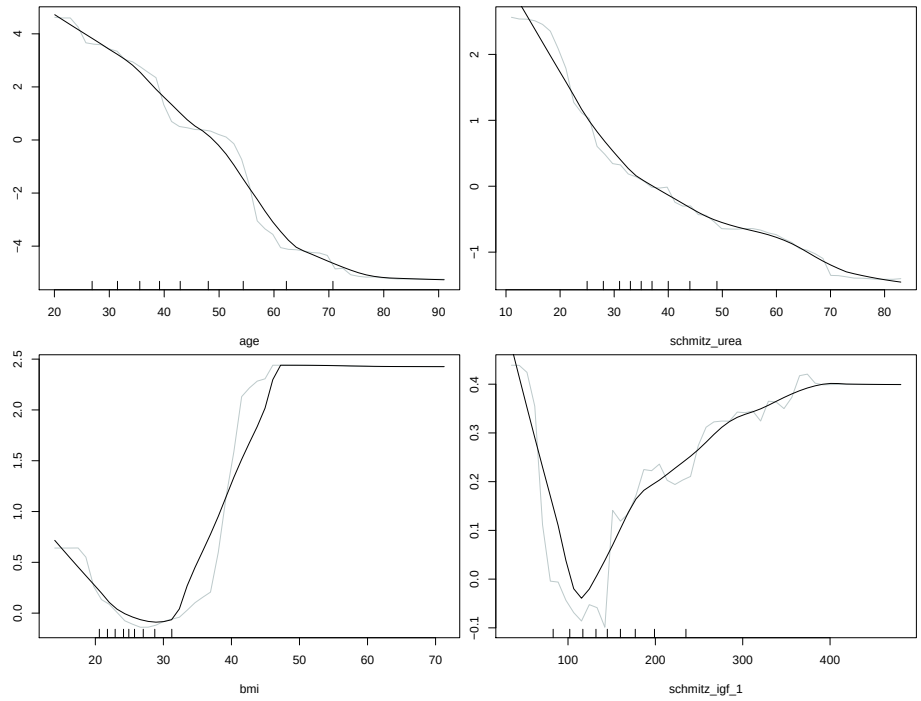

Figure 8.9: Smoothed partial dependence plot for age, schmitz_urea, bmi, and schmitz_igf_1.

some smoothed partial dependence plots for four selected clinical variables, namely age, schmitz_urea, bmi, and schmitz_igf_1, created using the `lowess` function in R with smoothing parameter `f=.3`.

8.6 Rule Extraction for SNPs

8.6.1 Partial Dependence Plots for SNPs

The partial dependence plots shown in figures 8.10 and 8.11 display the relationship between each SNP and the response variable inside the random forest ensemble. Each plot shows the value of the response value (y-axis) changing depending on the three different values taken for every SNPs (x-axis), due to the fact that each SNP is an ordered categorical variable taking on values 0-1-2. Namely, the homozygous dominant genotype (i.e. AA) is coded by 0, the heterozygous genotype (i.e. AB/BA) by 1, and the homozygous recessive genotype (i.e. BB) by 2.

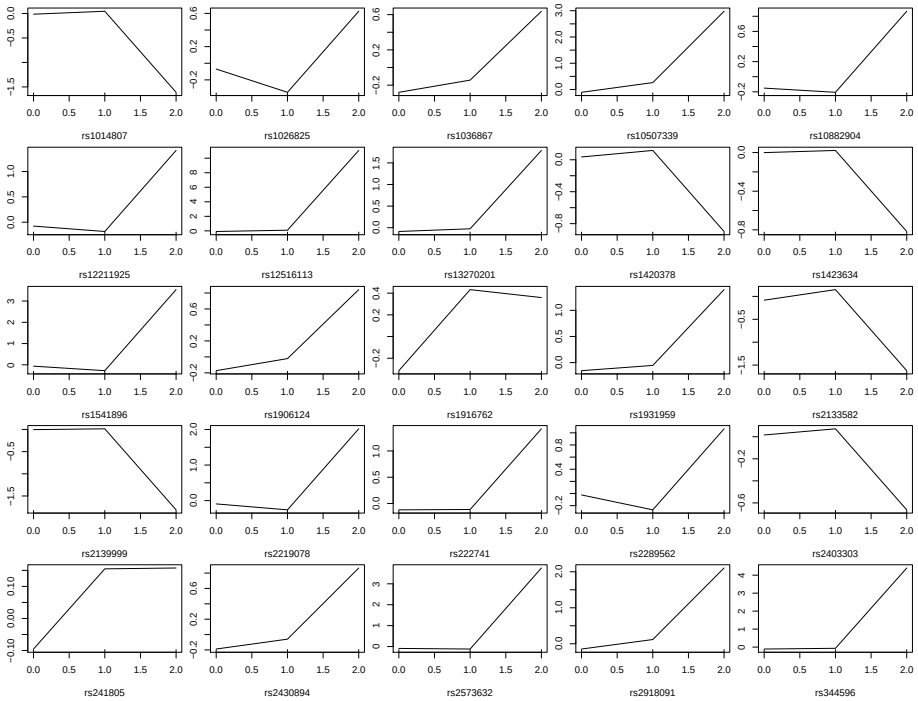

Figure 8.10: Partial Dependence Plots for SNPs 1:25

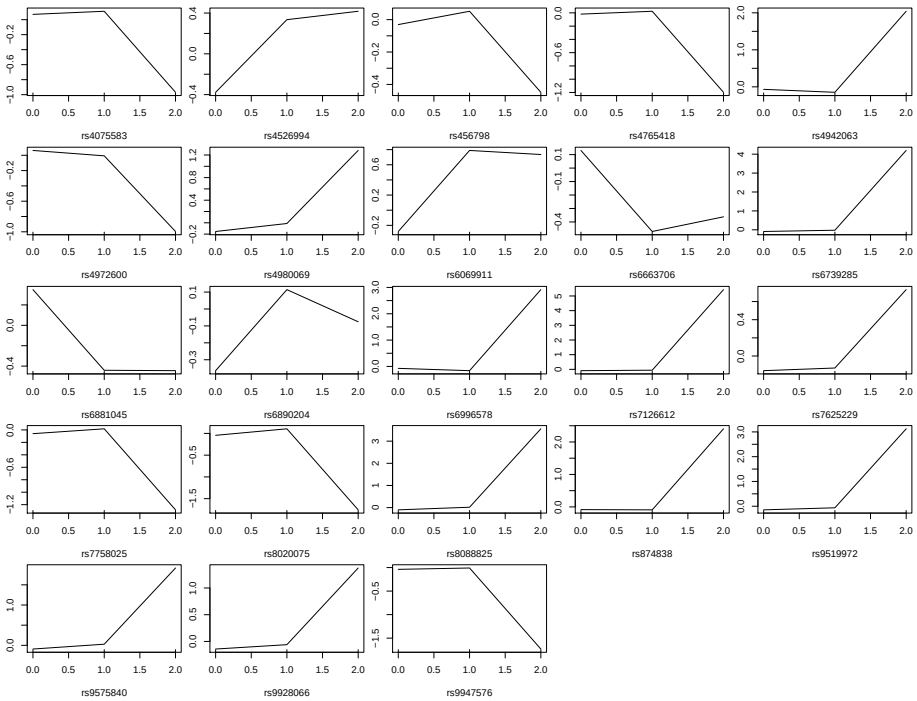

Figure 8.11: Partial Dependence Plots for SNPs 26:48

8.6.2 Variable Importances for SNPs

We recomputed three VIM's (Gini VIM, unscaled, and scaled VIM) inside a hundred RF iterations, and then computed the median value for each VIM.

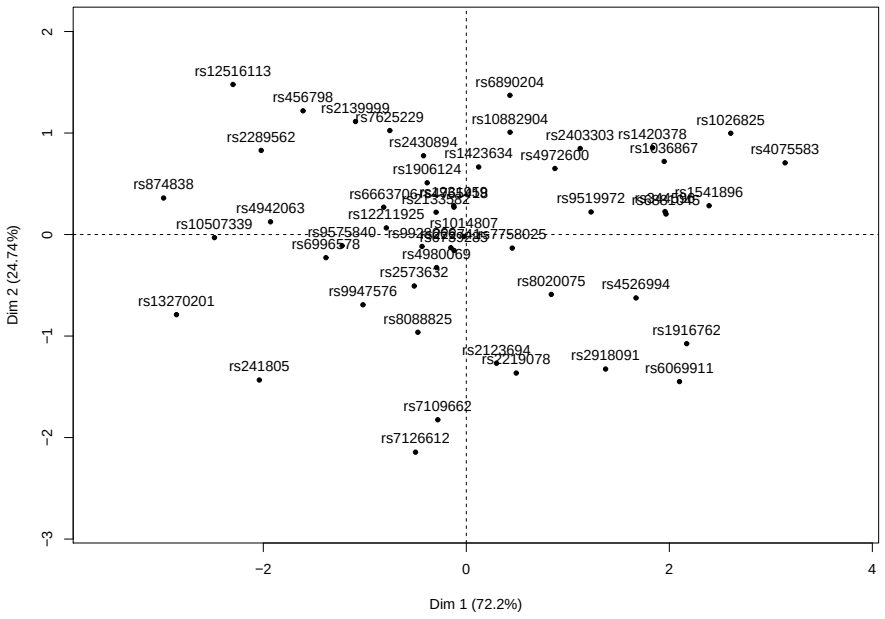

Figure 8.12: Ranking and PC value of SNPs

We take the median value in order to take into account outliers, and perform a PCA using the median VIM values of the three measures seen above. Figure 8.12 shows the value of each SNP according to the first two principal components, which explain 72.2% and 24.74% of variance respectively. Due to the large amount of variability explained by the first principal component, it is advisable to create a single and final ranking based upon it (see Table 8.3).

8.6.3 Fitting Estimation for SNPs

The pair of CARTs in Figure 8.13 show the Fitting Estimation procedure for the SNPs selected (i.e. using dataset X_2 as explained in Table 6.1), in detail the figure provides the CART build to its maximum size (left) and the CART build with default value `mindev=1e-6`, both using the predicted

Table 8.3: Ranking and first PC value of SNPs

rank	variable	PC	rank	variable	PC
1	rs4075583	3.140	26	rs222741	-0.152
2	rs1026825	2.605	27	rs7109662	-0.281
3	rs1541896	2.391	28	rs4980069	-0.293
4	rs1916762	2.171	29	rs2133582	-0.298
5	rs6069911	2.100	30	rs1906124	-0.386
6	rs6881045	1.967	31	rs2430894	-0.422
7	rs344596	1.957	32	rs9928066	-0.437
8	rs1036867	1.949	33	rs8088825	-0.477
9	rs1420378	1.835	34	rs7126612	-0.501
10	rs4526994	1.672	35	rs2573632	-0.513
11	rs2918091	1.372	36	rs7625229	-0.754
12	rs9519972	1.229	37	rs12211925	-0.788
13	rs2403303	1.121	38	rs6663706	-0.814
14	rs4972600	0.873	39	rs9947576	-1.019
15	rs8020075	0.837	40	rs2139999	-1.092
16	rs2219078	0.492	41	rs9575840	-1.224
17	rs7758025	0.452	42	rs6996578	-1.383
18	rs10882904	0.431	43	rs456798	-1.609
19	rs6890204	0.429	44	rs4942063	-1.929
20	rs2123694	0.297	45	rs2289562	-2.021
21	rs1423634	0.121	46	rs241805	-2.039
22	rs1014807	-0.023	47	rs12516113	-2.299
23	rs6739285	-0.121	48	rs10507339	-2.482
24	rs4765418	-0.122	49	rs13270201	-2.856
25	rs1931959	-0.125	50	rs874838	-2.983

values from a Random Forest with `ntree`=5000. It is clear that noise has been eliminated from the data, and extremely uninformative splits are evident in the maximally grown tree, and are easily removed in the CART with default values in the function `tree` in R.

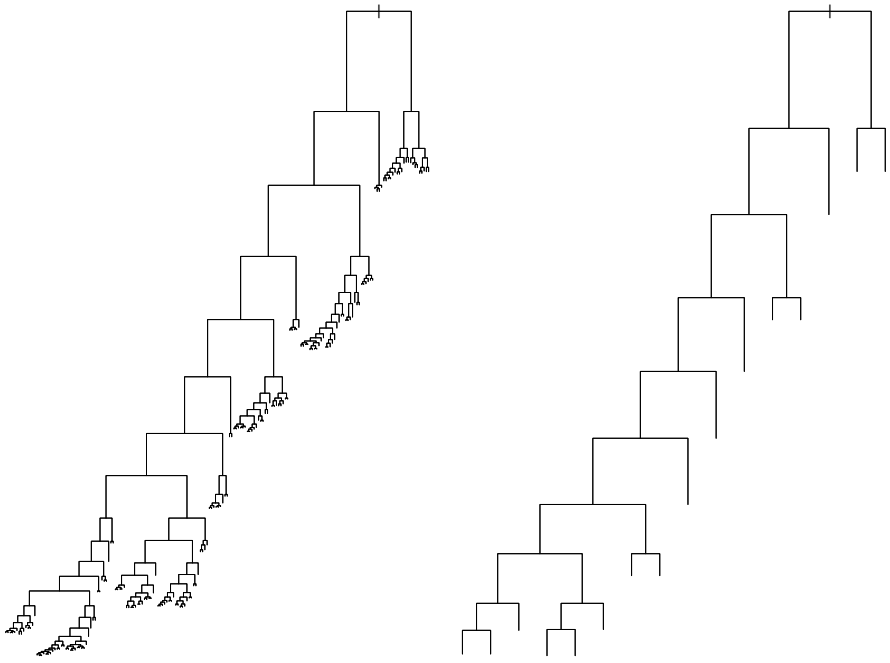

Figure 8.13: Fitting Estimation showing overfitting (left), and Fitting Estimation (right)

Figure 8.14 shows a regression tree providing a stable representation of the underlying structure of the ensemble used. Let us recall that one important drawback of CART is its instability, i.e. small changes in the training data can cause big changes in the resulting model. In this case, the changes depend on the seed used to run the ensemble method that produces the predicted values: multiple runs, with different seed values, showed that the CARTs resulting from the Fitting Estimation procedure did not differ much from one another. As a result, the Fitting Estimation procedure produced a stable tree. It is important to note that all splits are carried out according to the homozygous dominant genotype (i.e. AA, coded by 0) or the heterozygous genotype (i.e. AB/BA, coded by 1), while the the homozygous recessive genotype (i.e. BB, coded by 2) is never included in

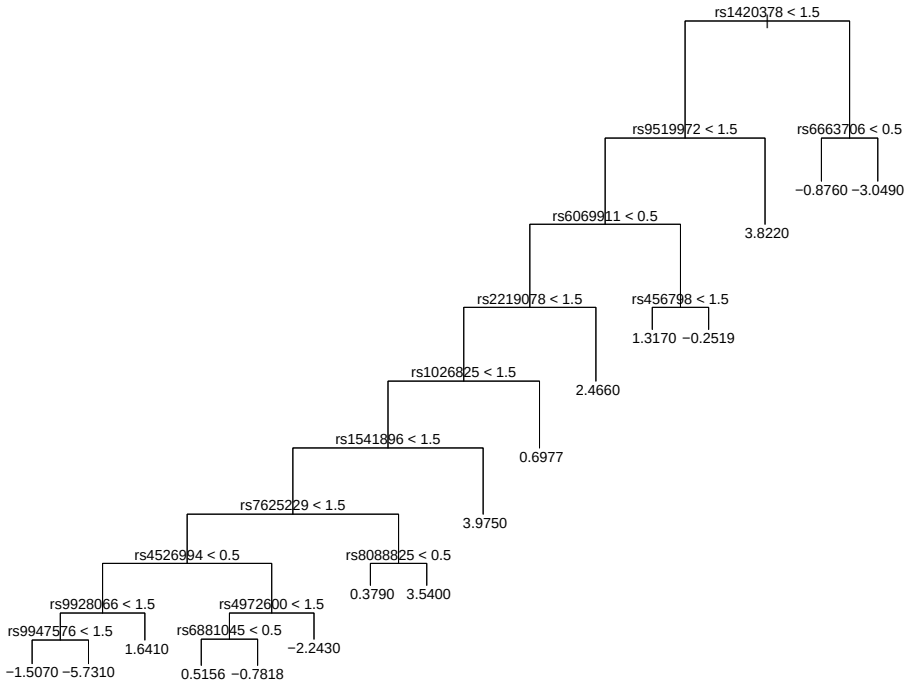

Figure 8.14: Fitting Estimation procedure showing cutting points and SNP names.

the splits. Similarly, Table 8.4 shows the result of the Fitting Estimation procedure, showing the number of the node (column node), the splitting variable with its corresponding cutting point (column variable), the node size (column size), the deviance value of the node (column deviance), and the predicted value (column outcome).

8.6.4 Node Harvest for SNPs

While the author claims that satisfactory results can be achieved using two-order interaction (`maxinter`=2), the parameters in Node Harvest should be tuned like in any other method: Figure 8.15 shows the RMSE values obtained for different interaction orders achieved inside a ten-fold cross-validation using the subset of SNPs previously achieved (i.e. using dataset X_2 as explained in Table 6.1).

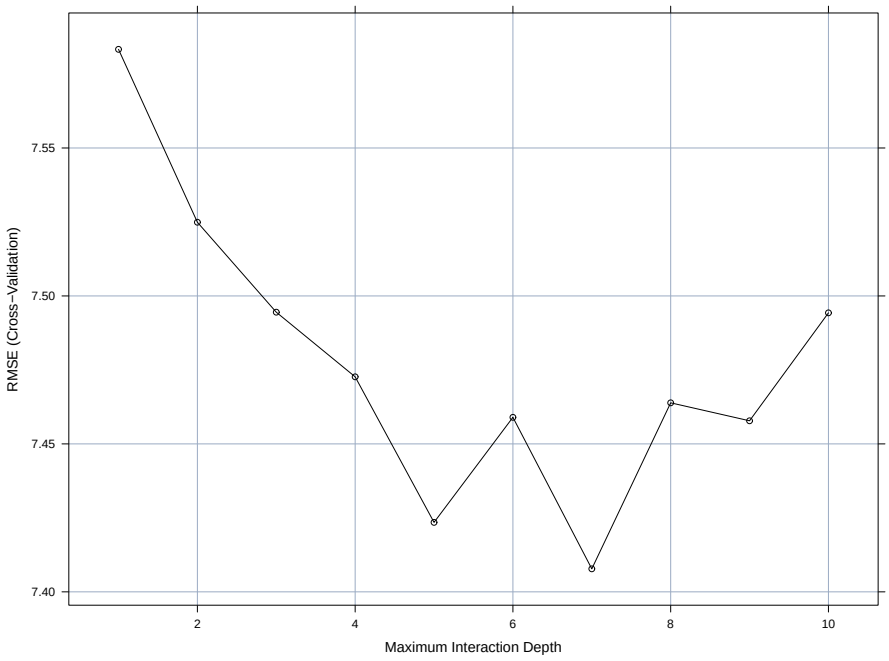

Figure 8.15: Tuning of node harvest interaction depth.

The smallest RMSE is obtained for `maxinter`=7, with `maxinter`=5 providing a similar result. In this case, one has to bear in mind that at this level one is particularly interested in obtaining a simple interpretable model, as a

Table 8.4: Fitting Estimation description, * indicates terminal node

node	variable			size	deviance	outcome	
1)	root			1201	7713	-0.1676	
2)	rs1420378	<	1.5	1064	6506	0.04822	
4)	rs9519972	<	1.5	1042	6121	-0.03145	
8)	rs6069911	<	0.5	823	4701	-0.3121	
16)	rs2219078	<	1.5	789	4309	-0.4318	
32)	rs1026825	<	1.5	633	3410	-0.7101	
64)	rs1541896	<	1.5	622	3121	-0.793	
128)	rs7625229	<	1.5	568	2613	-0.9601	
256)	rs4526994	<	0.5	313	1329	-1.474	
512)	rs9928066	<	1.5	303	1178	-1.577	
1024)	rs9947576	<	1.5	298	1086	-1.507	*
1025)	rs9947576	>	1.5	5	4.474	-5.731	*
513)	rs9928066	>	1.5	10	50.29	1.641	*
257)	rs4526994	>	0.5	255	1100	-0.3291	
514)	rs4972600	<	1.5	231	922.7	-0.1303	
1028)	rs6881045	<	0.5	116	399.8	0.5156	*
1029)	rs6881045	>	0.5	115	425.7	-0.7818	*
515)	rs4972600	>	1.5	24	80.79	-2.243	*
129)	rs7625229	>	1.5	54	325.1	0.9645	
258)	rs8088825	<	0.5	44	158.7	0.379	*
259)	rs8088825	>	0.5	10	84.89	3.54	*
65)	rs1541896	>	1.5	11	43.06	3.975	*
33)	rs1026825	>	1.5	156	651	0.6977	*
17)	rs2219078	>	1.5	34	118.4	2.466	*
9)	rs6069911	>	0.5	219	1111	1.023	
18)	rs456798	<	1.5	178	925.2	1.317	*
19)	rs456798	>	1.5	41	104.2	-0.2519	*
5)	rs9519972	>	1.5	22	65.07	3.822	*
3)	rs1420378	>	1.5	137	773.4	-1.844	
6)	rs6663706	<	0.5	76	325	-0.876	*
7)	rs6663706	>	0.5	61	288.6	-3.049	*

result imposing an interaction order equal to five (or even seven) could jeopardize such purpose, i.e. providing many nodes with many interaction rules, which could be too lengthy to visually inspect. Bearing this in mind, as a compromise resulting between interpretation and performance, we select an interaction order equal to three, while keeping `nodesize`=5 as suggested by Meinshausen (2010). Figure 8.16 shows the result plotted for such Node Harvest solution: it is evident that even an interaction depth equal to three is already too much for such solution to be plotted, and except for some nodes, the result is quite chaotic and of little use. The resulting Node Harvest estimator features 82 nodes, and tables 8.6, 8.7, and 8.8 provide the values for each node, namely, the weight, the mean of the node, and the number of samples included in such node, along with the SNPs used to create each node.

The resulting estimator is a small number of nodes that predict the training observations as a weighted average of node means. Because nodes are shown with decreasing weights, progressively approaching zero, for interpretation purposes, one can focus on the first ranking nodes.

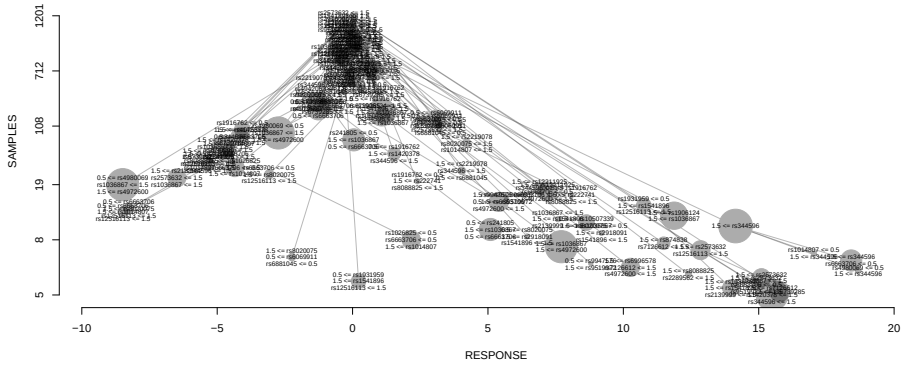

Figure 8.16: Node Harvest for SNPs selected.

It is of interest to note that the first splitting variable in the Fitting Estimation procedure (i.e. SNP rs1420378) shows up in three nodes (among the first teen) in the Node Harvest solution. Similarly, SNP rs6663706 shows up in many nodes in the Node Harvest solution, which rank high. In order to summarize the overlap between the two methods, Table 8.5 shows the inclusion frequencies of the SNPs selected in the Fitting Estimation proce-

dure that also compare in the Node Harvest solution. For instance, SNP rs6663706 appears most frequently (30.9%), while SNP rs1420378 appears 12.3% of times.

Table 8.5: Inclusion frequencies of SNPs in the Fitting Estimation that also appear in the Node Harvest solution.

SNP ID	frequencies	
rs6663706	25	0.309
rs1541896	13	0.160
rs4972600	10	0.123
rs1420378	7	0.086
rs6069911	7	0.086
rs8088825	6	0.074
rs6881045	5	0.062
rs9519972	4	0.049
rs2219078	4	0.049
rs9928066	2	0.025
rs1026825	2	0.025
rs9947576	2	0.025
rs456798	0	0.000
rs7625229	0	0.000
rs4526994	0	0.000

Table 8.6: Node Harvest Estimator for SNPs: nodes 1-27

	node #	samples	mean	weight	rule		
1	21	10	14.1	0.321	1.5 <= rs344596		
2	33	19	-8.49	0.293	1.5 <= rs4972600	rs1036867 <= 1.5	0.5 <= rs4980069
3	48	69	-2.74	0.293	1.5 <= rs4972600	rs1036867 <= 1.5	rs4980069 <= 0.5
4	17	8	7.71	0.293	1.5 <= rs4972600	1.5 <= rs1036867	
5	4	5	15.5	0.285	rs344596 <= 1.5	rs1420378 <= 1.5	1.5 <= rs7126612
6	54	184	1.07	0.212	rs344596 <= 1.5	1.5 <= rs1036867	rs1906124 <= 1.5
7	24	13	11.9	0.212	1.5 <= rs1036867	1.5 <= rs1906124	
8	71	990	-0.567	0.140	rs344596 <= 1.5	rs7126612 <= 1.5	rs1036867 <= 1.5
9	45	50	-4.78	0.139	0.5 <= rs6663706	rs1036867 <= 1.5	1.5 <= rs1420378
10	55	292	-1.26	0.139	0.5 <= rs6663706	rs1036867 <= 1.5	rs1420378 <= 1.5
11	20	9	5.09	0.139	0.5 <= rs6663706	1.5 <= rs1036867	0.5 <= rs241805
12	46	63	0.0449	0.139	0.5 <= rs6663706	1.5 <= rs1036867	rs241805 <= 0.5
13	8	7	18.4	0.128	rs6663706 <= 0.5	1.5 <= rs344596	
14	3	5	15.8	0.117	1.5 <= rs6739285		
15	66	712	0.811	0.104	rs4972600 <= 1.5	rs6663706 <= 0.5	rs12211925 <= 1.5
16	31	17	7.2	0.104	rs4972600 <= 1.5	rs6663706 <= 0.5	1.5 <= rs12211925
17	26	14	5.4	0.104	rs4972600 <= 1.5	0.5 <= rs6663706	1.5 <= rs9928066
18	57	362	-1.34	0.104	rs4972600 <= 1.5	0.5 <= rs6663706	rs9928066 <= 1.5
19	18	8	12.9	0.102	rs12516113 <= 1.5	1.5 <= rs2573632	
20	15	8	2.17	0.097	1.5 <= rs1014807	rs6663706 <= 0.5	rs1026825 <= 0.5
21	38	28	-4.35	0.097	1.5 <= rs1014807	rs6663706 <= 0.5	0.5 <= rs1026825
22	75	1092	0.0995	0.095	rs4972600 <= 1.5	rs2573632 <= 1.5	rs7126612 <= 1.5
23	74	1087	0.0612	0.095	rs6996578 <= 1.5	rs4972600 <= 1.5	rs344596 <= 1.5
24	10	7	10.3	0.095	rs4972600 <= 1.5	rs7126612 <= 1.5	1.5 <= rs6996578
25	5	6	15.1	0.094	rs1036867 <= 1.5	1.5 <= rs2573632	
26	35	23	-6.49	0.094	rs1036867 <= 1.5	rs2573632 <= 1.5	1.5 <= rs2133582
27	70	974	-0.328	0.094	rs1036867 <= 1.5	rs2573632 <= 1.5	rs2133582 <= 1.5

Table 8.7: Node Harvest Estimator for SNPs: nodes 28-54

	node #	samples	mean	weight	rule		
28	52	174	3.09	0.093	rs6739285 <= 1.5	rs6663706 <= 0.5	0.5 <= rs6069911
29	64	610	-0.0471	0.093	rs6739285 <= 1.5	rs6663706 <= 0.5	rs6069911 <= 0.5
30	47	65	-4.17	0.085	rs6739285 <= 1.5	0.5 <= rs6663706	1.5 <= rs4075583
31	56	347	-0.913	0.085	rs6739285 <= 1.5	0.5 <= rs6663706	rs4075583 <= 1.5
32	2	5	14	0.078	rs2139999 <= 1.5	1.5 <= rs1541896	1.5 <= rs1036867
33	49	97	-4.07	0.070	rs344596 <= 1.5	1.5 <= rs1420378	rs1916762 <= 0.5
34	53	181	1.13	0.070	1.5 <= rs1036867	rs10507339 <= 0.5	rs1541896 <= 1.5
35	22	12	8.64	0.070	1.5 <= rs1036867	0.5 <= rs10507339	
36	67	744	0.72	0.069	rs1014807 <= 1.5	rs6663706 <= 0.5	rs344596 <= 1.5
37	6	6	0.221	0.061	rs12516113 <= 1.5	1.5 <= rs1541896	0.5 <= rs1931959
38	27	14	10.8	0.061	rs12516113 <= 1.5	1.5 <= rs1541896	rs1931959 <= 0.5
39	39	34	-5.33	0.049	rs344596 <= 1.5	rs1036867 <= 1.5	1.5 <= rs2139999
40	69	960	-0.325	0.049	rs344596 <= 1.5	rs1036867 <= 1.5	rs2139999 <= 1.5
41	42	39	1.54	0.046	rs344596 <= 1.5	1.5 <= rs1420378	0.5 <= rs1916762
42	72	1050	0.11	0.046	rs344596 <= 1.5	rs1420378 <= 1.5	rs7126612 <= 1.5
43	16	8	11.5	0.045	rs7126612 <= 1.5	1.5 <= rs874838	
44	78	1124	-0.0302	0.045	rs1541896 <= 1.5	rs874838 <= 1.5	rs1014807 <= 1.5
45	43	48	-4.56	0.045	rs1541896 <= 1.5	rs874838 <= 1.5	1.5 <= rs1014807
46	68	762	0.402	0.043	rs6663706 <= 0.5	rs344596 <= 1.5	rs12211925 <= 1.5
47	32	18	6.91	0.043	rs6663706 <= 0.5	rs344596 <= 1.5	1.5 <= rs12211925
48	28	14	-8.26	0.032	rs12516113 <= 1.5	1.5 <= rs8020075	0.5 <= rs6663706
49	23	13	-8.47	0.028	rs12516113 <= 1.5	1.5 <= rs1014807	0.5 <= rs6663706
50	63	609	-0.089	0.027	rs2573632 <= 1.5	rs6663706 <= 0.5	rs6069911 <= 0.5
51	1	5	14.9	0.027	rs9519972 <= 1.5	rs6663706 <= 0.5	1.5 <= rs2573632
52	51	173	3.12	0.027	rs2573632 <= 1.5	rs6663706 <= 0.5	0.5 <= rs6069911
53	65	698	-0.111	0.024	rs6739285 <= 1.5	rs1916762 <= 0.5	rs1420378 <= 1.5
54	36	25	-3.08	0.024	rs12516113 <= 1.5	1.5 <= rs8020075	rs6663706 <= 0.5

Table 8.8: Node Harvest Estimator for SNPs: nodes 55-82

	node #	samples	mean	weight	rule		
55	73	1063	0.185	0.020	rs1014807 <= 1.5	rs8020075 <= 1.5	rs2219078 <= 1.5
56	44	50	4.22	0.020	rs1014807 <= 1.5	rs8020075 <= 1.5	1.5 <= rs2219078
57	37	27	4.03	0.016	0.5 <= rs6881045	rs344596 <= 1.5	1.5 <= rs2219078
58	62	603	-1.14	0.016	0.5 <= rs6881045	rs344596 <= 1.5	rs2219078 <= 1.5
59	7	7	-2.24	0.016	rs6881045 <= 0.5	0.5 <= rs6069911	1.5 <= rs8020075
60	50	108	3.25	0.016	rs6881045 <= 0.5	0.5 <= rs6069911	rs8020075 <= 1.5
61	61	452	0.317	0.016	rs6881045 <= 0.5	rs6069911 <= 0.5	
62	60	400	0.972	0.016	rs6739285 <= 1.5	0.5 <= rs1916762	
63	12	7	12.4	0.011	rs2289562 <= 1.5	1.5 <= rs8088825	
64	82	1201	-1.00E-16	0.010	ROOT NODE		
65	58	396	0.836	0.008	rs6739285 <= 1.5	0.5 <= rs1916762	rs344596 <= 1.5
66	59	399	-1.17	0.008	0.5 <= rs6663706	rs8020075 <= 1.5	rs4942063 <= 1.5
67	25	13	7.59	0.008	rs2139999 <= 1.5	1.5 <= rs1541896	rs1036867 <= 1.5
68	19	8	17	0.008	1.5 <= rs344596	rs1014807 <= 0.5	
69	41	36	-5.32	0.008	rs12516113 <= 1.5	rs6739285 <= 1.5	1.5 <= rs2139999
70	11	7	18.7	0.008	1.5 <= rs344596	rs4980069 <= 0.5	
71	77	1120	-0.0682	0.008	rs8088825 <= 1.5	rs222741 <= 1.5	rs2139999 <= 1.5
72	76	1097	0.172	0.008	rs2139999 <= 1.5	rs1541896 <= 1.5	rs1014807 <= 1.5
73	40	35	-5.27	0.008	rs1014807 <= 1.5	rs8088825 <= 1.5	1.5 <= rs2139999
74	34	22	2.39	0.008	rs8088825 <= 1.5	1.5 <= rs222741	rs1916762 <= 0.5
75	30	16	8.08	0.008	rs8088825 <= 1.5	1.5 <= rs222741	0.5 <= rs1916762
76	81	1157	-0.328	0.007	rs1541896 <= 1.5	rs2918091 <= 1.5	rs2573632 <= 1.5
77	13	8	9.2	0.007	rs1541896 <= 1.5	1.5 <= rs2918091	rs8020075 <= 0.5
78	14	8	6.45	0.007	rs1541896 <= 1.5	1.5 <= rs2918091	0.5 <= rs8020075
79	80	1154	-0.326	0.005	rs9519972 <= 1.5	rs7126612 <= 1.5	rs1541896 <= 1.5
80	9	7	8.8	0.005	1.5 <= rs9519972	0.5 <= rs9947576	
81	29	15	5.7	0.005	1.5 <= rs9519972	rs9947576 <= 0.5	
82	79	1135	-0.0406	0.003	rs8088825 <= 1.5	rs1541896 <= 1.5	rs8020075 <= 1.5

Chapter 9

Concluding Remarks

This work has offered an overview of the performance of some feature selection algorithms and rule extraction methods in a combined view to GWAS data. Although the use of random forest is not new in genetics, its use as a screening procedure combined with the use of the feature selection algorithms presented is wrapped in a novel perspective.

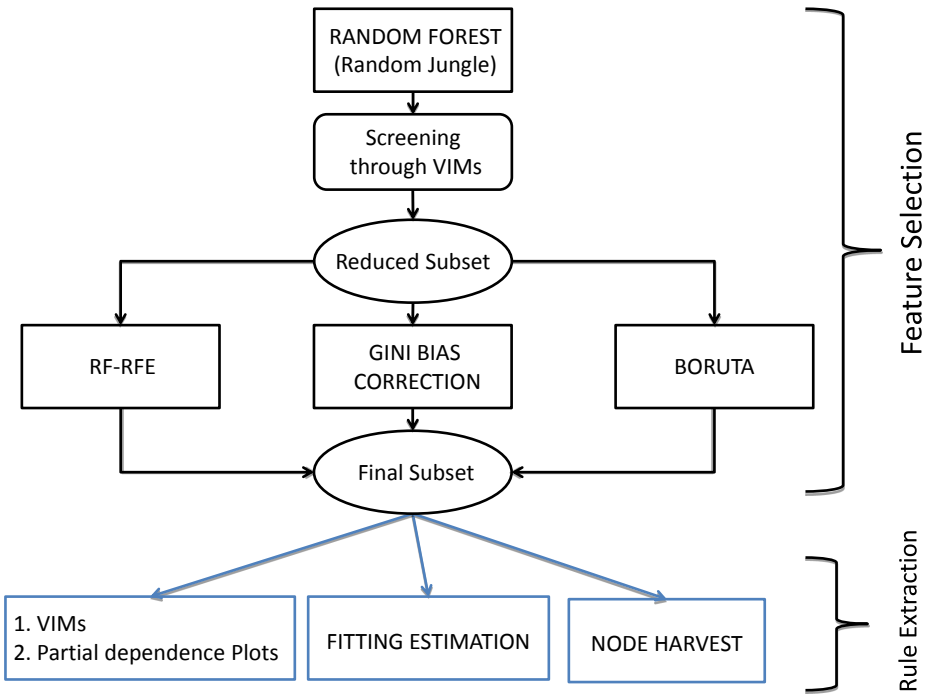

Figure 9.1: Summary Schema

Figure 9.1 shows the schema of such approach (focusing on the selection of genetic variables, i.e. SNPs). In many fields - specially in genetics where one wants to investigate the association of a disease or trait with a given set of variables - one has to deal with a huge amount of covariates, and the removal of irrelevant features is of prime importance. This can be achieved as follows:

1. Feature selection phase:

    (a) A single iteration of random forest is performed through the use of the Random Jungle package;

    (b) Variable importance measures are used as a guidance towards a computationally handable subset of covariates (e.g. ~ 2000, but that directly depends on the computational power at hand);

    (c) The application of finer feature selection algorithms (e.g. the ones outlined in this work) in order to strongly eliminate irrelevant features;

    (d) Results can be aggregated from all the feature selection algorithms used as to obtain a final subset with a small number of features.

If one is interested not only in a small number of SNPs associated with the outcome, but also in interpreting the relationship between such SNPs and the response, then a rule extraction phase should be performed.

2. Rule extraction phase:

    (a) Ensembles provide variable importance measures in order to assess which features are the most useful in the prediction of the outcome, as well as partial dependence plots, which graphically show how the outcome varies upon changing the values of a given set (one or two) of covariates.

    (b) The Fitting Estimation procedure produces a clear *white-box* model, that is as interpretable as any CART, yet providing stable results.

    (c) The Node Harvest estimator is a new ensemble method that restores interpretability providing an ensemble of just a few nodes, predicting the training observations as weighted means across nodes.

The application of CART-based methods requires no assumption about the underlying generating process, allowing interaction effects and non-linear relations among the outcome and the predictor variables.

While the application of the screening phase still requires a powerful machine, the wrappers presented show to be feasible for relatively large datasets even on less capable computers.

Our findings support some of the results from the previous study based on the univariate approach used in Pattaro *et al.* (2007), while new loci were identified, although that might need further biological investigation.

This procedure was effective in reducing the number of variables from an extremely large dataset. The initial number of SNPs ($p \simeq 300000$) was reduced to $p_1 \simeq 2000$, and then further reduced to $p_2 = 50$.
These 50 SNPs showed to be helpful in the prediction of the estimated glomerular filtration rate (eGFR) in addition to clinical variables, providing a Root Mean Squared Error that was statistically lower than that provided by clinical variables alone.
The rule extraction phase allowed to create a clear model, which could be helpful to a physician or biologist as a starting point for the inspection of the interactions between genes relevant to the estimated glomerular filtration rate (eGFR).

Future research is now devoted to assess (i) the prediction performance of these 50 SNPs in an independent population sample and (ii) the biological relevance of the newly identified SNPs.

Appendix

Linear Mixed Models

This section has the aim of providing a brief overview of linear mixed models, that can be useful in cases - like the one provided in the following work - when one has to deal with particular types of data, e.g. clustered data, that cannot be implicitly modeled within machine learning algorithms. Thus, linear mixed models can be seen as a preprocessing step. The name mixed effects model come from the fact that these models incorporate both *fixed effects*, which describe the relationships of the covariates to the dependent variable for an entire population, and *random effects*, which are associated with individual experimental units drawn at random from a population. These models are used to describe relationships between a response variable and some covariates that are grouped according to some classification factors. Typical data involved in these analysis are clustered data, repeated measures data, longitudinal data, and block designs. Our main concern is about clustered data, which can be defined as datasets in which the dependent variable is measured only once for each observation, but these observations are grouped into, or nested within, clusters of observations, e.g. data might be collected from students withing the same classroom or patients in the same clinic. In our dataset, observations are sampled inside families, therefore we need to model random genetic effects, namely using a kinship matrix. Roughly speaking a kinship matrix can be seen as a similarity matrix, showing to what extent observations are close to each other in terms of a predefined set of single nucleotide polymorphisms (SNPs). This can be achieved with the `GenABEL` package in R through the `ibs` function, which computes the identity-by-state coefficient, ranging from 0 to 1: e.g people with extremely high ibs values (close to 1) may indicate duplicated samples

(or twins), while simply high values of IBS may indicate relatives.
The general matrix specification of a linear mixed model takes the form

$$\mathbf{Y}=\mathbf{X}\boldsymbol{\beta}+\mathbf{Z}\mathbf{u}+\boldsymbol{\varepsilon} \tag{9.1}$$

where $\mathbf{Y}$ represents a vector of a continuous response, $\mathbf{X}$ is an $n \times p$ design matrix. In a model including an intercept term, the first column would be simply equal to 1 for all observation. $\boldsymbol{\beta}$ is a vector of p unknown regression coefficients (or fixed effect parameters). $\mathbf{Z}$ is an $n \times q$ is a covariate matrix associated with random effects, while $\mathbf{u}$ is a vector of q random effects associated with the q covariates in the $\mathbf{Z}$ matrix. The q random effects in the $\mathbf{u}$ vector follow a multivariate normal distribution with mean vector $\mathbf{0}$ and a variance-covariance matrix denoted by $\mathbf{D}$

$$\mathbf{u} \sim N(\mathbf{0}, \mathbf{D}) \tag{9.2}$$

Finally, $\boldsymbol{\varepsilon}$ is a vector of random error terms of size n. The components of this vector are random variables that follow a multivariate normal distribution with mean vector $\mathbf{0}$ and a variance-covariance matrix denoted by $\mathbf{R}$

$$\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \mathbf{R}) \tag{9.3}$$

For an exhaustive explanation of linear mixed models, one can refer to Pinheiro and Bates (2000) or West *et al.* (2006). Therefore, in case of individuals related through a multigenerational pedigree, we propose to fit a linear mixed model using a kinship matrix, where familiality can be treated as a random effect in order to remove the effect of relatedness, and then to retain the residuals from this fit and to use them as the new response variable.
In this case matrix $\mathbf{D} = \sigma_u^2\mathbf{K}$, where $\mathbf{K}$ is the kinship matrix, and σ_u^2 is the unknown genetic variance, while $\mathbf{R} = \mathbf{I}\sigma_\varepsilon^2$, where σ_ε^2 is the unknown residual variance; while matrices $\mathbf{X}$ and $\mathbf{Z}$ are reduced to intercept terms.

Bibliography

Almuallim, H. and Dietterich, T. (1991). Learning with many irrelevant features. In *Proceedings of the ninth National conference on Artificial intelligence*, volume 2, pages 547–552.

Amaratunga, D., Cabrera, J., and Lee, Y. (2008). Enriched random forests. *Bioinformatics*, **24**(18), 2010–2014.

Ambroise, C. and McLachlan, G. (2002). Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the National Academy of Sciences*, **99**(10), 6562–6566.

Barakat, N. and Diederich, J. (2004). Learning-based rule-extraction from support vector machines. In *The 14th International Conference on Computer Theory and applications ICCTA'2004*. not found.

Bell, D. and Wang, H. (2000). A formalism for relevance and its application in feature subset selection. *Machine learning*, **41**(2), 175–195.

Berk, R. (2008). *Statistical learning from a regression perspective*. Springer.

Berrar, D., Bradbury, I., and Dubitzky, W. (2006). Avoiding model selection bias in small-sample genomic datasets. *Bioinformatics*, **22**(10), 1245–1250.

Biau, G., Devroye, L., and Lugosi, G. (2008). Consistency of random forests and other averaging classifiers. *The Journal of Machine Learning Research*, **9**, 2015–2033.

Blum, A. and Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial intelligence*, **97**(1), 245–271.

Böger, C., Gorski, M., Li, M., Hoffmann, M., Huang, C., Yang, Q., Teumer, A., Krane, V., O'Seaghdha, C., Kutalik, Z., *et al.* (2011). Association of egfr-related loci identified by gwas with incident ckd and esrd. *PLoS genetics*, **7**(9), e1002292.

Bowman, A. and Azzalini, A. (1997). Applied smoothing techniques for data analysis.

Braga-Neto, U., Hashimoto, R., Dougherty, E., Nguyen, D., and Carroll, R. (2004). Is cross-validation better than resubstitution for ranking genes? *Bioinformatics*, **20**(2), 253–258.

Breiman, L. (1996). Bagging predictors. *Machine learning*, **24**(2), 123–140.

Breiman, L. (2001). Random forests. *Machine learning*, **45**(1), 5–32.

Breiman, L. (2002). Manual on setting up, using, and understanding random forests v3. 1. *Statistics Department University of California Berkeley, CA, USA*.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and decision trees. *Wadsworth, Belmont*.

Bureau, A., Dupuis, J., Falls, K., Lunetta, K., Hayward, B., Keith, T., and Van Eerdewegh, P. (2005). Identifying snps predictive of phenotype using random forests. *Genetic epidemiology*, **28**(2), 171–182.

Chang, J., Yeh, R., Wiencke, J., Wiemels, J., Smirnov, I., Pico, A., Tihan, T., Patoka, J., Miike, R., Sison, J., *et al.* (2008). Pathway analysis of single-nucleotide polymorphisms potentially associated with glioblastoma multiforme susceptibility using random forests. *Cancer Epidemiology Biomarkers & Prevention*, **17**(6), 1368–1373.

Cockcroft, D. and Gault, M. (1976). Prediction of creatinine clearance from serum creatinine. *Nephron*, **16**(1), 31–41.

Cover, T. and Thomas, J. (2006). *Elements of information theory*. Wiley-interscience.

Culverhouse, R., Klein, T., and Shannon, W. (2004). Detecting epistatic interactions contributing to quantitative traits. *Genetic epidemiology*, **27**(2), 141–152.

Díaz-Uriarte, R. and De Andres, S. (2006). Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, **7**(1), 3.

Dietterich, T. (1997). Machine-learning research. *AI magazine*, **18**(4), 97.

Dietterich, T. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, **40**(2), 139–157.

Dobra, A. and Gehrke, J. (2001). Bias correction in classification tree construction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 90–97. Morgan Kaufmann Publishers Inc.

Efron, B. and Tibshirani, R. (1997). Improvements on cross-validation: the 632+ bootstrap method. *Journal of the American Statistical Association*, **92**(438), 548–560.

Fan, J. and Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **70**(5), 849–911.

Freund, Y. and Schapire, R. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer.

Friedman, J. (1991). Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67.

Friedman, J. (2001). Greedy function approximation: a gradient boosting machine. *Ann. Statist*, **29**(5), 1189–1232.

Friedman, J. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, **38**(4), 367–378.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer Series in Statistics.

Friedman, J., Popescu, B., *et al.* (2003). Importance sampled learning ensembles. *Journal of Machine Learning Research*, **94305**.

Gärdenfors, P. (1978). On the logic of relevance. *Synthese*, **37**(3), 351–367.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, **63**(1), 3–42.

Gibbs, R., Belmont, J., Hardenbol, P., Willis, T., Yu, F., Yang, H., Ch'ang, L., Huang, W., Liu, B., Shen, Y., *et al.* (2003). The international hapmap project. *Nature*, **426**(6968), 789–796.

Glassock, R. and Winearls, C. (2009). Ageing and the glomerular filtration rate: truths and consequences. *Transactions of the American Clinical and Climatological Association*, **120**, 419.

Goldstein, B., Hubbard, A., Cutler, A., and Barcellos, L. (2010). An application of random forests to a genome-wide association dataset: Methodological considerations & new findings. *BMC genetics*, **11**(1), 49.

Granitto, P., Furlanello, C., Biasioli, F., and Gasperi, F. (2006). Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products. *Chemometrics and intelligent laboratory systems*, **83**(2), 83–90.

Guyon, I., Saffari, A., Dror, G., and Cawley, G. (2010). Model selection: Beyond the bayesian/frequentist divide. *The Journal of Machine Learning Research*, **11**, 61–87.

Hahn, L., Ritchie, M., and Moore, J. (2003). Multifactor dimensionality reduction software for detecting gene–gene and gene–environment interactions. *Bioinformatics*, **19**(3), 376–382.

Hapfelmeier, A., Hothorn, T., Ulm, K., and Strobl, C. (2011). A new variable importance measure for random forests with missing data. *Statistics and Computing*, pages 1–14.

Hindorff, L., Sethupathy, P., Junkins, H., Ramos, E., Mehta, J., Collins, F., and Manolio, T. (2009). Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. *Proceedings of the National Academy of Sciences*, **106**(23), 9362–9367.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **20**(8), 832–844.

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

Hyafil, L. and Rivest, R. L. (1976). Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, **5**(1), 15–17.

Ishwaran, H. and Kogalur, U. (2010). Consistency of random survival forests. *Statistics & probability letters*, **80**(13), 1056–1064.

Jiang, R., Tang, W., Wu, X., and Fu, W. (2009). A random forest approach to the detection of epistatic interactions in case-control studies. *BMC bioinformatics*, **10**(Suppl 1), S65.

Johansson, U., König, R., and Niklasson, L. (2010). Genetic rule extraction optimizing brier score. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1007–1014. ACM.

John, G., Kohavi, R., Pfleger, K., *et al.* (1994). Irrelevant features and the subset selection problem. In *Proceedings of the eleventh international conference on machine learning*, volume 129, pages 121–129. San Francisco.

Johnson, A., Handsaker, R., Pulit, S., Nizzari, M., O'Donnell, C., and de Bakker, P. (2008). Snap: a web-based tool for identification and annotation of proxy snps using hapmap. *Bioinformatics*, **24**(24), 2938–2939.

Kass, G. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied statistics*, pages 119–127.

Kim, J. (2009). Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics & Data Analysis*, **53**(11), 3735–3745.

King, R., Stansfield, W., and Mulligan, P. (2006). *A dictionary of genetics*. Oxford University Press, USA.

Kira, K. and Rendell, L. (1992). A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256. Morgan Kaufmann Publishers Inc.

Kohavi, R. (1994). Feature subset selection as search with probabilistic estimates. In *AAAI fall symposium on relevance*, volume 224.

Kohavi, R. and John, G. (1997). Wrappers for feature subset selection. *Artificial intelligence*, **97**(1), 273–324.

Kursa, M. and Rudnicki, W. (2011). A deceiving charm of feature selection: The microarray case study. *Man-Machine Interactions 2*, pages 145–152.

Kursa, M., Jankowski, A., and Rudnicki, W. (2010). Boruta–a system for feature selection. *Fundamenta Informaticae*, **101**(4), 271–285.

Levey, A., Bosch, J., Lewis, J., and Greene, T. (1999). A more accurate method to estimate glomerular filtration rate from serum creatinine: a new prediction equation.

Levey, A., Stevens, L., Schmid, C., Zhang, Y., Castro, A., Feldman, H., Kusek, J., Eggers, P., Van Lente, F., Greene, T., *et al.* (2009). A new equation to estimate glomerular filtration rate. *Annals of internal medicine*, **150**(9), 604.

Li, J., Horstman, B., and Chen, Y. (2011). Detecting epistatic effects in association studies at a genomic level based on an ensemble approach. *Bioinformatics*, **27**(13), i222–i229.

Liakhovitski, D., Bryukhov, Y., and Conklin, M. (2010). Relative importance of predictors: Comparison of random forests with johnson's relative weights. *Model Assisted Statistics and Applications*, **5**(4), 235–249.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R news*, **2**(3), 18–22.

Lin, Y. and Jeon, Y. (2006). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, **101**(474), 578–590.

Lunetta, K., Hayward, L., Segal, J., and Van Eerdewegh, P. (2004). Screening large-scale association study data: exploiting interactions using random forests. *BMC genetics*, **5**(1), 32.

Macías-Núñez, J. and López-Novoa, J. (2008). Physiology of the healthy aging kidney. *The Aging Kidney in Health and Disease*, pages 93–112.

Manisera, M., Sandri, M., and Zuccolotto, P. (2012). Fitting estimation. *Unpublished Manuscript*.

Manolio, T., Collins, F., Cox, N., Goldstein, D., Hindorff, L., Hunter, D., McCarthy, M., Ramos, E., Cardon, L., Chakravarti, A., *et al.* (2009). Finding the missing heritability of complex diseases. *Nature*, **461**(7265), 747–753.

Meinshausen, N. (2006). Quantile regression forests. *The Journal of Machine Learning Research*, **7**, 983–999.

Meinshausen, N. (2010). Node harvest. *The Annals of Applied Statistics*, **4**(4), 2049–2072.

Meng, Y., Yu, Y., Cupples, L., Farrer, L., and Lunetta, K. (2009). Performance of random forest when snps are in linkage disequilibrium. *BMC bioinformatics*, **10**(1), 78.

Moore, J. and Dunlap, J. (2010). *Computational Methods for Genetics of Complex Traits*. Academic Press.

Nelson, M., Kardia, S., Ferrell, R., and Sing, C. (2001). A combinatorial partitioning method to identify multilocus genotypic partitions that predict quantitative trait variation. *Genome Research*, **11**(3), 458–470.

Nicodemus, K. and Malley, J. (2009). Predictor correlation impacts machine learning algorithms: implications for genomic studies. *Bioinformatics*, **25**(15), 1884–1890.

Nicodemus, K., Malley, J., Strobl, C., and Ziegler, A. (2010). The behaviour of random forest permutation-based variable importance measures under predictor correlation. *BMC bioinformatics*, **11**(1), 110.

Pang, H., George, S., Hui, K., and Tong, T. (2012). Gene selection using iterative feature elimination random forests for survival outcomes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, **9**(5), 1422–1431.

Pattaro, C., Marroni, F., Riegler, A., Mascalzoni, D., Pichler, I., Volpato, C., Dal Cero, U., De Grandi, A., Egger, C., Eisendle, A., *et al.* (2007). The genetic study of three population microisolates in south tyrol (micros): study design and epidemiological perspectives. *BMC medical genetics*, **8**(1), 29.

Pattaro, C., De Grandi, A., Vitart, V., Hayward, C., Franke, A., Aulchenko, Y., Johansson, A., Wild, S., Melville, S., Isaacs, A., *et al.* (2010). A meta-analysis of genome-wide data from five european isolates reveals an association of col22a1, syt1, and gabrr2 with serum creatinine level. *BMC medical genetics*, **11**(1), 41.

Pattaro, C., Köttgen, A., Teumer, A., Garnaas, M., Böger, C., Fuchsberger, C., Olden, M., Chen, M., Tin, A., Taliun, D., *et al.* (2012). Genome-wide association and functional follow-up reveals new loci for kidney function. *PLoS Genetics*, **8**(3), e1002584.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Pepe, M., Longton, G., Anderson, G., and Schummer, M. (2003). Selecting differentially expressed genes from microarray experiments. *Biometrics*, **59**(1), 133–142.

Pinheiro, J. and Bates, D. (2000). *Mixed-effects models in S and S-PLUS*. Springer Verlag.

Quinlan, J. (1993). *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann.

Ratner, B. (2011). *Statistical and machine-learning data mining: techniques for better predictive modeling and analysis of big data*. CRC Press.

Rudnicki, W., Kierczak, M., Koronacki, J., and Komorowski, J. (2006). A statistical method for determining importance of variables in an information system. In *Rough Sets and Current Trends in Computing*, pages 557–566. Springer.

Sandri, M. and Zuccolotto, P. (2006). Variable selection using random forests. *Data Analysis, Classification and the Forward Search*, pages 263–270.

Sandri, M. and Zuccolotto, P. (2008). A bias correction algorithm for the gini variable importance measure in classification trees. *Journal of Computational and Graphical Statistics*, **17**(3), 611–628.

Schwarz, D., König, I., and Ziegler, A. (2010). On safari to random jungle: a fast implementation of random forests for high-dimensional data. *Bioinformatics*, **26**(14), 1752–1758.

Schweitzer, H. (1995). Occam algorithms for computing visual motion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **17**(11), 1033–1042.

Seni, G. and Elder, J. F. (2010). Ensemble methods in data mining: improving accuracy through combining predictions. *Synthesis Lectures on Data Mining and Knowledge Discovery*, **2**(1), 1–126.

Somorjai, R., Dolenko, B., and Baumgartner, R. (2003). Class prediction and discovery using gene microarray and proteomics mass spectroscopy data: curses, caveats, cautions. *Bioinformatics*, **19**(12), 1484–1491.

Stevens, L., Coresh, J., Greene, T., and Levey, A. (2006). Assessing kidney functionmeasured and estimated glomerular filtration rate. *New England Journal of Medicine*, **354**(23), 2473–2483.

Strobl, C. and Zeileis, A. (2008). Danger: High power!–exploring the statistical properties of a test for random forest variable importance.

Strobl, C., Boulesteix, A., Zeileis, A., and Hothorn, T. (2007a). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics*, **8**(1), 25.

Strobl, C., Boulesteix, A., and Augustin, T. (2007b). Unbiased split selection for classification trees based on the gini index. *Computational Statistics & Data Analysis*, **52**(1), 483–501.

Strobl, C., Boulesteix, A., Kneib, T., Augustin, T., and Zeileis, A. (2008). Conditional variable importance for random forests. *BMC bioinformatics*, **9**(1), 307.

Sun, Y., Cai, Z., Desai, K., Lawrance, R., Leff, R., Jawaid, A., Kardia, S., and Yang, H. (2007). Classification of rheumatoid arthritis status with candidate gene and genome-wide single-nucleotide polymorphisms using random forests. In *BMC proceedings*, volume 1, page S62. BioMed Central Ltd.

Svetnik, V., Liaw, A., Tong, C., and Wang, T. (2004). Application of breimans random forest to modeling structure-activity relationships of pharmaceutical molecules. *Multiple Classifier Systems*, pages 334–343.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Traskin, M. (2007). *On the consistency of ensemble classification algorithms*. ProQuest.

West, B., Welch, K., and Galecki, A. (2006). *Linear mixed models: a practical guide using statistical software*. Chapman & Hall/CRC.

Wille, A., Hoh, J., and Ott, J. (2003). Sum statistics for the joint detection of multiple disease loci in case-control association studies with snp markers. *Genetic epidemiology*, **25**(4), 350–359.

Wolpert, D. (1990). The relationship between occams razor and convergent guessing. *Complex Systems*, **4**, 319–368.

Xin, L. and Zhu, M. (2012). Stochastic stepwise ensembles for variable selection. *Journal of Computational and Graphical Statistics*, **21**(2), 275–294.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **67**(2), 301–320.