NTNU
Norwegian University of
Science and Technology

Using Ensemble Methods to Improve the Performance of Prediction

Statistical Analysis of a Myocardial Infarction
Data Set from the HUNT Study

Julia Barbara Debik

Master of Science in Physics and Mathematics
Submission date: June 2017
Supervisor: Mette Langaas, IMF

Norwegian University of Science and Technology
Department of Mathematical Sciences

Preface

This master thesis concludes my Maser of Science in Applied Physics and Mathematics with specialization in Industrial Mathematics at the Norwegian University of Science and Technology (NTNU). The work was done during spring 2017 at the Department of Mathematical Sciences.

I would like to thank my supervisor Mette Langaas for outstanding guidance and support in the process of writing this thesis. Further, I would like to thank Anja Bye at the Department of Circulation and Medical Imaging for providing the HUNT data set, and to Torbjørn Velle-Forbord for his support in understanding the role of various predictors in connection with myocardial infarction.

I would also like to thank Ingelin Steinsland for encouragement and excellent supervision during my TMA4500 - Specialization Project in Industrial Mathematics.

Trondheim, Norway
Julia Debik
June, 2017

Abstract

The main focus of this thesis is to evaluate the use of ensemble methods to improve the performance of prediction, when applied on a myocardial infarction (MI) data set from the HUNT study. The data set comes from a prospective case-control study with a 10-years follow-up (fatal and non-fatal) where MI was used as the primary endpoint. The subjects of this study were 200 healthy individuals with age 60-79 from the HUNT2 study. The cases ($n_1 = 100$) experienced an MI within the 10-year follow-up, whereas the controls ($n_2 = 100$) remained health during this period. Several risk factors for experiencing a MI have been identified over the last years and are used in risk prediction models. The most popular prediction model is the Framingham score. However, about 15-20% of patients experiencing MI did not score high at any of the traditional risk factors.

Recent studies have shown that microRNAs, which are small non-coding RNA molecules, have a large potential as diagnostic biomarkers for cardiovascular disease. It is thus interesting to investigate if microRNAs also have a potential as predictive biomarkers for predicting future instances of MI.

Logistic regression and tree-based methods are commonly used to predict a binary outcome, when predictor variables are observed. In recent years we have seen increased popularity of ensemble methods. One such method is bagging (bootstrap aggregation). Bagging is performed by resampling a data set many times, customizing a prediction model for each resampled data set and then combining prediction models for these data sets into a new prediction. In this thesis we examine how bagging can be applied to classification trees and to logistic regression. We also investigate the closely related ensemble methods random forests and random GLMs, which include only a subset of predictors in each step of the model fitting. The predictive performance of 6 different statistical models (a *pruned tree, logistic GLM, bagged tree, bagged GLM, random forest* and *random GLM*) is evaluated through a simulation study, where we use the area under the ROC curve and the Brier score for assessing prediction accuracy.

We then fit our 6 models to the HUNT data set in order to obtain conclusions about which predictors are relevant for predicting a future MI event. The conclusion is that ensemble methods increase the predictive performance, in particular when applied to classification trees. The best predictive power was obtained by fitting a random GLM. Further, we have seen that microRNAs are highly relevant for predicting MI, and that the predictors BMI, serum triglycerides and serum glucose non fasting, which are not included in the Framingham risk score, are of high importance.

Sammendrag

Hovedfokuset i denne oppgaven er å evaluere bruken av ensemble metoder for å forbedre nøyaktigheten av prediksjoner, når disse anvendes på et hjerteinfarkt datasett fra HUNT studien. Datasettet kommer fra en kasus-kontroll studie med 10 års oppfølging (fatal og ikke-fatal) hvor hjerteinfarkt ble brukt som primært endepunkt. Deltakerne av denne studien var 200 friske individer i alderen 60-79 år fra HUNT2 studien. Kasusene ($n_1 = 100$) opplevde hjerteinfarkt i løpet av den 10-år lange observasjonsperioden, mens kontrollene ($n_2 = 100$) forble friske i løpet av oppfølgingsperioden. Flere risiko-faktorer for å utvikle hjerteinfarkt har blitt identifisert i løpet av de siste årene og blir brukt i risikovurderingsmodeller. Den mest populære prediksjonsmodellen er Framingham skåre. Likevel har omtrent 15-20% av pasientene som har opplevd hjerteinfarkt ikke skåret høyt på noen av de tradisjonelle risiko-faktorene.

Nylig forskning har vist at microRNAer, som er små ikke-kodende RNA molekyler, har et stort potensial som diagnostiske biomarkører for hjerte- og karsykdommer. Det er derfor interessant å utforske om microRNAer også kan ha et potensial som prediktorer for fremtidig utvikling av hjerteinfarkt.

Logistisk regresjon og klassifikasjonstrær er mye brukt for å predikere et binært utfall når forklaringsvariabler er blitt observerte. I løpet av de siste årene har vi sett en økning i populæriteten av ensemble metoder. En slik metode er "bagging" ("bootstrap aggregation"). "Bagging" utføres ved å resample et datasett mange ganger, å tilpasse en prediksjonsmodell for hvert resamplet datasett for deretter å kombinere prediksjonsmodellene for disse datasettene til en ny prediksjon. I denne oppgaven utforsker vi hvordan "bagging" kan anvendes på klassifikasjonstrær og på logistisk regresjon. Vi utforsker også de nært beskletede ensemble metodene "random forest" og "random GLM", som inkluderer kun en delmengde av prediktorer i hvert steg av modelltilpasningen. Prediksjonsnøyaktigheten av 6 forskjellige statistiske metoder (*"pruned tree", logistisk GLM, "bagged tree", "bagged GLM", "random forest"* og *"random GLM"*) blir evaluert ved hjelp av en simuleringsstudie, hvor vi bruker arealet under en ROC kurve og Brier skåre for å evaluere prediksjonsnøyaktigheten.

Vi tilpasser deretter våre 6 modeller til datasettet fra HUNT for å trekke konklusjoner om hvilke prediktorer som er relevante for å predikere et fremtidig tilfelle av hjerteinfarkt. Modellen med best prediksjonsnøyaktighet viste seg å være "random GLM". Videre har vi sett at microRNAer er veldig relevante for å predikere hjerteinfarkt, og at prediktorene kroppsmasseindeks, serum triglycerider og serum ikke-fastende glukose, som ikke er inkludert i Framingham skåre, er av høy viktighet.

List of Abbreviations

AIC	Akaike Information Criterion
AUC	Area Under Curve
BMI	Body Mass Index
BP	Blood Pressure
BS	Brier Score
CAD	Coronary Artery Disease
CART	Classification and Regression Tree
CHD	Coronary Heart Disease
CV	Cross-Validation
CVD	Cardiovascular Disease
GLM	Generalized Linear Model
HCHD	Hard Coronary Heart Disease
HDL	High-Density Lipoproteins
HUNT	Nord-Trøndelag Health Study
LOOCV	Leave-One-Out Cross-Validation
miRNA	microRNA, small non-coding RNA molecule
MI	Myocardial Infarction
OOB	Out-Of-Bag
PCR	Polymerase Chain Reaction
RGLM	Random Generalized Linear Model
ROC	Receiver Operating Characteristics
SBP	Systolic Blood Pressure
WHR	Waist-to-Hip Ratio

Contents

Chapter 1

Introduction

1.1 Motivation from a Medical Perspective

Over the last years coronary artery disease (CAD) has emerged as one of the most common causes of death worldwide. In 2015 cardiovascular disease (CVD) was the most common cause of death within Europe, causing 45% of deaths (Townsend et al., 2015). The number of people suffering from CAD in Norway is expected to increase further as the number of persons at risk is increasing. This increase is mainly due to the prevalence of obesity and diabetes. It is crucial to identify risk factors such that accurate prediction models can be constructed. With accurate prediction models individuals with a high risk of developing CAD can be identified and preventive actions can be taken.

To accomplish this, several CVD risk prediction models have been developed. These models include known risk factors and return the likelihood of a CVD event to occur within a defined period of time (Velle-Forbord, 2017). The most widely used risk prediction model is the Framingham risk score. However, about $15-20\%$ of those developing a MI would be categorized as low risk by these risk prediction models and hence not captured by medical specialists (UN et al., 2003).

Several studies have in the last decade shown that circulating miRNAs have potential as biomarkers of cardiovascular diseases. If they have potential as predictive biomarkers is still unsure. A study performed by Bye et al. (2016) has identified and verified significant associations between the level of 10 different circulating miRNAs and subsequent fatal MI. This thesis is inspired by the that study and the works of two undergraduate medical researchers Velle-Forbord (2017) and Eidlaug (2017). They have concluded that microRNAs

are promising as predictive biomarkers for MI by making a statistical analysis using logistic regression and ROC curves. In this thesis we will continue their work by introducing statistical ensemble methods.

1.2 Motivation from a Statistical Perspective

Logistic regression and tree-based methods are commonly used to predict a binary outcome, when predictor variables are observed. In the fields of medicine and biology logistic regression plays a prominent role and is one of the most commonly used statistical tools. During recent years ensemble methods have become increasingly popular. Ensemble methods use multiple learning algorithms to obtain better predictive performance than can be obtained by using one learning algorithm alone.

Examples of ensemble methods are bagging, boosting and random forests. In this thesis we will concentrate on *bagging*, also known as bootstrap aggregation, and on *random forests*. Bagging is a procedure for reducing the variance of a statistical learning method. It is performed by re-sampling a data set many times, customizing a prediction for each re-sampled data set and then combining prediction models for these data sets into a new prediction. By averaging a set of predictions the variance of the combined predictions is reduced, and the accuracy of the prediction is increased. Random forests are closely related to bagging. The difference is that when customizing a prediction to each re-sampled data set, only a random set of predictors is used. The result is that the correlation between the prediction models is weakened and the variance of the combined prediction is further reduced.

In this thesis we investigate how these ensemble methods perform in combination with logistic regression and in combination with classification trees. We will see how the idea of random forests can be extended to logistic regression, where logistic regression in performed on bootstrapped samples of the data, and where only a subset of the covariates is used to fit the model. The latter gives us a random GLM (RGLM).

The main motivation for this thesis, from a statistical point of view, has been to understand ensemble methods and to investigate when they might be useful. In order to do so we use a simulation study. We test the methods on a simulated data set which is constructed such that it resembles our data set from the HUNT study. We then apply these methods to a myocardial data set from the HUNT study. We conclude by giving guidelines for when these ensemble methods might be useful for data sets of the same type as our HUNT data

set.

1.3 The HUNT Data Set

The data set analyzed in this thesis originates from the second Nord-Trøndelag Health Study (the HUNT study). The HUNT databank is Norway's largest collection of health data about a population. Data has been collected through three population studies, the HUNT1 (1984-1986), HUNT2 (1995-1997) and HUNT3 (2006-2008) studies. A total of 120 000 individuals from the Nord-Trøndelag region have participated in the studies, of which 80 000 participants have submitted blood samples. The HUNT study provided a solid basis for research into population health on a wide variety of conditions and life style factors, ranging over research fields from social epidemiology to genetic research. The HUNT Research Center collaborates with national and international research groups on some of the important health topics facing our world today using the most modern techniques and their state of the art biobank (NTNU, 2017).

Subjects

The data analysed in this thesis comes from a prospective case-control study with a 10-years observation period (fatal and non-fatal) in which myocardial infarction (MI) was used as the primary endpoint. By linking the Norwegian Myocardial Infarction Registry and the Norwegian Cause of Death Registry to the HUNT2 data base, 200 healthy participants aged 60-79 years from the HUNT2 study wer included. Three subjects have been excluded because of a missing smoking status or waist circumference. The subjects of this study have thus been reduced to 197 individuals, of which 126 are men and 71 are women. The cases ($n_1 = 97$) had suffered from a fatal or non-fatal MI within the 10 year follow-up, whereas the controls ($n_2 = 100$) remained at health during this period. The controls are age and gender matched based on information from HUNT2. This is partial matching. Exclusion criteria were self-reported CVD (previous/current), known hypertension (BP>140/90), chronic kidney disease or diabetes mellitus. Participants who died from other causes than MI were excluded.

1.4 Thesis Outline

In Chapter 2 we give background on MI. We identify known risk factors and introduce existing risk prediction models. We continue by explaining what microRNAs are, what their function in the human body is, and why they might have a potential use as biomarkers for

predicting MI.

We start Chapter 3 by giving an introduction notation and terminology. Further, we look at statistical properties of logistic regression and classification trees. We proceed by discussing the dilemma in choosing a model from a number of potential models in light of the bias-variance tradeoff. Furthermore, we describe different methods for model evaluation, among them ROC curves, AUC and Brier scores and cross-validation.

Chapter 4 motivates ensemble methods. We start by introducing bagging and explain how it can be applied to classification trees and to logistic regression. We proceed by presenting random forests in the context of classification trees. We finally explain the concept of a random generalized linear model.

In Chapter 5 we perform a simulation study using the methods discussed in Chapter 3. We simulate two data sets of equal size which we will refer to as our training and test sets. The simulated data sets are constructed in such a way that they capture some of the most important features of our original myocardial infarction data set, but do not reproduce it exactly. This allows us to compare our methods and to arrive at guidelines for when bagging and random forests might be useful for data set of the same type as our HUNT data set.

Finally, we apply the ensemble methods to our original HUNT data set in Chapter 6.

Chapter 7 summarizes the results. Conclusions about the statistical methods are obtained as well as conclusions about the medical aspects concerning this thesis.

An overview of the `R`-codes is included in the Appendix.

Chapter 2

Background

Coronary heart disease (CHD) is a group of diseases that includes stable angina, unstable angina, myocardial infarction, and sudden cardiac death. Over the last years CHD has emerged as one of the most common causes of death worldwide, with over 8 million deaths (16.8 %) in 2013 (Bye et al., 2016). The number of people suffering from CHD is expected to increase further as the obesity and diabetes are becoming more and more prevalent (Velle-Forbord, 2017). In this thesis the focus is on myocardial infarction (MI). Several risk factors for experiencing an MI have been identified and are used in risk prediction models. Most of these models determine the 10-year risk for an individual to develop any type of cardiovascular disease (CVD). The most popular prediction model is the Framingham score. However, these types of predictions fail to explain a proportion of the incidences. In general, about 15-20% of patients experiencing MI will not score high at any of the traditional risk factors and be categorized as in low risk of developing the disease (UN et al., 2003).

2.1 Framingham Score

The Framingham score is a sex specific algorithm used to estimate the risk of developing a hard coronary heart disease (myocardial infarction or coronary death) for an individual during the next ten years. The algorithm is based on data from the Framingham Heart Study (The National Heart, Institute, and University, The National Heart et al.), which is an ongoing cardiovascular cohort study. The study began in 1948 with 5209 adult participants. The participants were all from the town Framingham, in Massachusetts, with age between 30 and 62 years. None of the participants had developed symptoms of cardiovascular disease or suffered from heart attack or stroke when entering the study.

Table 2.1: Points corresponding to age used in estimating the 10-year risk of developing a hard coronary heart disease. Right: men. Left: women.

Age interval	Points
20 - 34	-9
35 - 39	-4
40 - 44	0
45 - 49	3
50 - 54	6
55 - 59	8
60 - 64	10
65 - 69	11
70 - 74	12
75 - 79	13

Age interval	Points
20 - 34	- 7
35 - 39	- 3
40 - 44	0
45 - 49	3
50 - 54	6
55 - 59	8
60 - 64	10
65 - 69	12
70 - 74	14
75 - 79	16

The study enrolled a second generation in 1971, consisting of 5124 adult children of the original participants. A third generation was enrolled in 2002 consisting of grandchildren of the original cohort. The Framingham Heart Study followed the participants over three generations, which allowed the scientists to identify CVD risk factors.

The Framingham score uses the following predictors:

- Age in the interval 20-79
- Total cholesterol in mg/dl
- HDL cholesterol in mg/dl
- Systolic blood pressure in mmHg
- Treatment for hypertension: yes/no
- Current smoking status: smoker/nonsmoker

Tables 2.1, 2.2, 2.3 and 2.4 show how the predictors age, total cholesterol and smoking status, HDL cholesterol and systolic blood pressure, are divided into intervals, and how each interval is assigned a number of points.

Table 2.2: Points corresponding to total cholesterol and smoking status used in estimating the 10-year risk of developing a hard coronary heart disease. Top: men. Bottom: women.

Points by age interval					
Total cholesterol	20-39	40-49	50-59	60-69	70-79
< 160	0	0	0	0	0
160 - 199	4	3	2	1	0
200 - 239	7	5	3	1	0
240 - 279	9	6	4	2	1
≥ 280	11	8	5	3	1
Nonsmoker	0	0	0	0	0
Smoker	8	5	3	1	1

Points by age interval					
Total cholesterol	20-39	40-49	50-59	60-69	70-79
< 160	0	0	0	0	0
160 - 199	4	3	2	1	1
200 - 239	8	6	4	2	1
240 - 279	11	8	5	3	2
≥ 280	13	10	7	4	2
Nonsmoker	0	0	0	0	0
Smoker	9	7	4	2	1

Table 2.3: Points corresponding to HDL used in estimating the 10-year risk of developing a hard coronary heart disease. Right: men. Left: women.

HDL (mg/dl)	Points
≥ 60	-1
50 - 59	0
40 - 49	1
< 40	2

HDL (mg/dl)	Points
≥ 60	-1
50 - 59	0
40 - 49	1
< 40	2

Table 2.4: Points corresponding to systolic blood pressure used in estimating the 10-year risk of developing a hard coronary heart disease. Top: men. Bottom: women.

Systolic BP (mmHg)	if untreated	if treated
< 120	0	0
120 - 129	0	1
130 - 139	1	2
140 - 159	1	2
≥ 160	2	3

Systolic BP (mmHg)	if untreated	if treated
< 120	0	0
120 - 129	1	3
130 - 139	2	4
140 - 159	3	5
≥ 160	4	6

Table 2.5: The sum of points obtained in Tables 2.1, 2.2, 2.3 and 2.4 are translated into the Framingham risk score. Left: men. Right: women.

10-year risk for *men*	
Point total	10-year risk
< 0	< 1 %
0	1 %
1	1 %
2	1 %
3	1 %
4	1 %
5	2 %
6	2 %
7	3 %
8	4 %
9	5 %
10	6 %
11	8 %
12	10 %
13	12 %
14	16 %
15	20 %
16	25 %
≥ 17	≥ 30 %

10-year risk for *women*	
Point total	10-year risk
< 9	< 1 %
9	1 %
10	1 %
11	1 %
12	1 %
13	2 %
14	2 %
15	3 %
16	4 %
17	5 %
18	6 %
19	8 %
20	11 %
21	14 %
22	17 %
23	22 %
24	27 %
≥ 25	≥ 30 %

Table 2.6: Example: Calculation of the Framingham risk score for four hypothetical patients.

	Patient 1	Points	Patient 2	Points	Patient 3	Points	Patient 4	Points
Sex	M		F		F		M	
Age	58	8	65	12	34	-7	76	13
Total chol.	167	2	202	2	193	4	243	1
HDL chol.	55	0	50	0	< 40	2	45	1
Smoker	no	0	yes	2	yes	9	yes	1
SBP	141	2	135	2	128	1	≥ 160	3
Hyp.treat.	yes		no		no		yes	
Point total		12		18		9		19
Risk		10 %		6 %		1 %		≥ 30 %

The sum of points is further translated into the Framingham risk score. This translation is also sex specific and can be found in Table 2.5. These calculations are best illustrated by an example. Table 2.6 shows how the points and the corresponding risks are calculated for four hypothetical patients. This risk score represents the risk of developing a hard CHD in the scope of the next ten years.

2.2 SCORE and NORRISK

The Framingham risk score is developed using a data set from participants in the United States. Recent studies have shown that it tends to overestimate the risk when tested on individuals from Europe. We will therefore shortly introduce two alternative scoring rules: the SCORE and the NORRISK.

The SCORE is an risk score developed in 2003 and is based on data from cohort studies in 12 European countries (Helsedirektoratet, 2009). The data was taken from a total of 205 178 healthy individuals with a total observation-period of 2.7 million years. 48 425 Norwegians have participated in this study. SCORE calculates the risk of developing cardiovascular death but can per today not estimate the risk for developing a non-fatal cardiovascular disease. High risk is defined in SCORE as at least 5% risk for cardiovascular death during the next 10 years. This is equivalent to a 20% risk in the Framingham score. The SCORE algorithm uses the predictors age, sex, systolic blood pressure and smoking

status. Additionally, the total cholesterol or total cholesterol/HDL cholesterol ratio is used as a predictor.

The SCORE Norway (NORRISK) is based on SCORE, but is adapted to the current mortality level and to levels of risk factors in Norway. The data set NORRISK is based on, consists of age-and sex specific mortality rates from the period 1999-2003, age- and sex-specific mean values of risk factors in Norwegian population studies from 2000-2003 and from relative risk assessment estimates in connection to specific risk factors as they appear in Norwegian cohort studies. It uses the same predictors as SCORE.

2.3 Micro RNAs

A microRNA (miRNA) is a small non-coding molecule containing 21-25 nucleotides. Their main location is inside the cell, but can also be found in the extracellular environment, for example in biological fluids and cell culture media. They play an important regularity role in plants and animals by targeting specific mRNAs for degradation or translation repression (Wahid et al., 2010). In the recent years scientists have discovered that there often are connections between the presence of miRNAs and diseases: certain miRNA expression patterns could be disease-specific. A comprehensive miRNA profiling study demonstrated that distinct miRNA expression patterns were specific to various types of cancers and were able to reflect the developmental lineage and differentiation state of tumors (Li and Kowdley, 2012). Similar connections have been found for other diseases, among them cardiovascular, inflammatory, neurodevelopmental, autoimmune, liver, skeletal muscle and skin diseases (Ardekani and Naeini, 2010). One can thus gain predictive value by identifying a disease-specific miRNA expression. Once this disease-specific miRNA expression is known, one can use it to evaluate the risk for an individual to develop the corresponding disease.

2.3.1 miRNA Linked to CVD

miRNAs have shown a potential as *diagnostic* biomarkers for MI, but whether they have potential as *predictive* biomarkers is still uncertain. A study performed by Bye et al. (2016) has identified and verified significant associations between the level of 10 different circulating miRNAs and subsequent fatal MI. These micRNAs are: miR-106a-5p, let-7g-5p, miR-26a5p, miR-424-5p, miR-151a-5p, let-7d-5p, miR-103a-3p, miR-148b-3p, miR-660-5p, miR-29c-3p. It important to test the clinical relevance of the candidate miRs as risk markers, by determining if they can add value on top of the traditional risk factors in the currently used risk prediction algorithms.

2.3.2 Quantitative Polymerase Chain Reaction (qPCR)

Quantitative polymerase chain reaction (qPCR) is a method for measuring the gene expression in a biological sample. We are interested in the difference of some specific micro RNA levels between healthy and diseased individuals. The measurement of a gene expression directly is challenging because the number of copies of a gene is very small and difficult to detect.

The qPCR method amplifies and simultaneously quantifies the expression of a gene (or miRNA) by generating thousands to millions of copies of a particular DNA sequence (Lien, 2011). The method works as follows: We start up with a sample, in which we have n_0 copies of different DNA molecules. In one PCR cycle these DNA molecules are replicated once, giving $n_1 = 2 \cdot n_0$ copies. This is called a perfect amplification. A perfect amplification is not always possible. In such a case one needs to estimate the amplification factor. After j runs of the PCR cycle one has n_j number of DNA strings:

$$n_j = 2 \cdot n_{j-1} = 2^j \cdot n_0.$$

In practice instead of measuring the actual number of DNA copies, they are measured using a fluorescence dying. We assume that the fluorescence level is proportional to the number of DNA copies: $f_j = \gamma \cdot n_j$, where γ is a scalar independent of cycle number. We can thus create a fluorescence curve

$$f_0 = E^{-j} \cdot f_j,$$

where E is the efficiency. $E = 2$ corresponds to a perfect amplification. The *Quantification Cycle (Cq)* is the cycle number at which the fluorescence signal crosses a threshold value. This is illustrated in the top plot of Figure 2.1[1].

The gene expression for each individual i and microRA j is different. The microRNA data used in this thesis have been normalized to the global mean (over all samples and all miRNAs). A variant of the threshold method, referred to as the second derivative method, is used for this data set, see Velle-Forbord (2017), Section 2.8 and 2.9. The difference in the Cq values of microRNA j for person i and the global mean is denoted by ΔCq_{ij},

$$\Delta Cq_{ij} = \bar{Cq} - Cq_{ij}, \quad \text{where} \quad \bar{Cq} = \frac{1}{n} \cdot \frac{1}{11} \sum_{i=1}^{n} \sum_{j=1}^{11} Cq_{ij},$$

[1]This figure was created using the `chipPCR` package (Roediger and Burdukiewicz, 2014).

as we have 11 different miRNAs in this data set. A higher value thus indicates that the miRNA is more abundant in this particular sample (Velle-Forbord, 2017) than the global mean. We thus have that

$$\Delta Cq_i \begin{cases} > 0 \implies \text{Individual } i \text{ has a high amount of miRNA compared to the global mean} \\ = 0 \implies \text{Individual } i \text{ has an amount of miRNA equal to the global mean} \\ < 0 \implies \text{Individual } i \text{ has a small amount of miRNA compared to the global mean.} \end{cases}$$

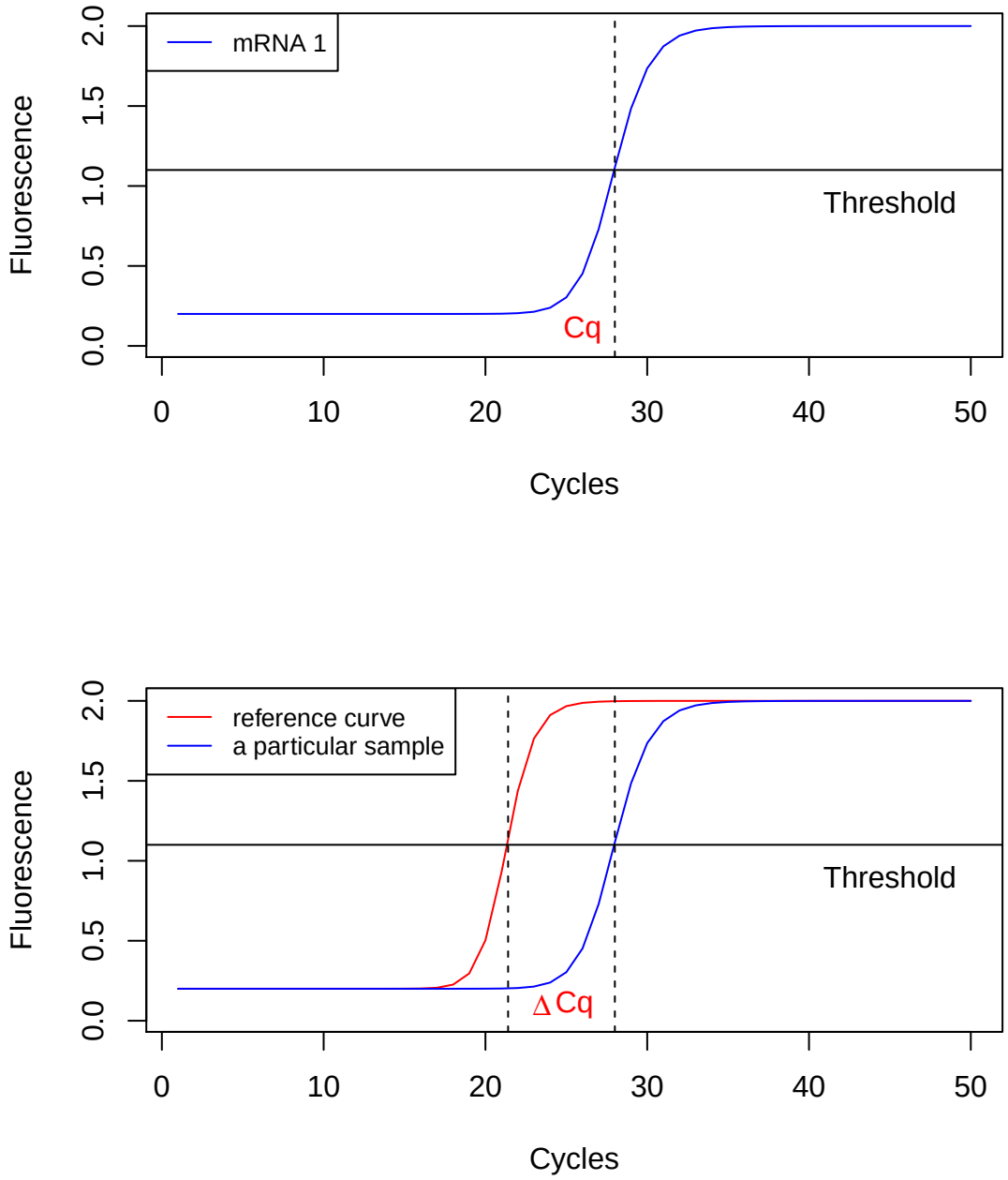


Figure 2.1: Simulated amplification curves. Top: One amplification curve showing the threshold and Cq. Bottom: ΔCq explained using two amplification curves for the threshold method.

Chapter 3

Statistical Methods

3.1 Terminology and Notation

A set of variables which can be measured or selected are called the *input* variables. These input variables may be *quantitative* or *qualitative* and can have an influence on one or more *output* variables. When *training* a statistical model one aims at fitting it in such a way that it describes the relationship between the input and output variables in a best possible way.

A quantitative variable is a numerical variable, i.e. one that can be measured, and will be denoted by Y. Often $Y \in \mathbb{R}$ or $Y \in \mathbb{N}$.

Qualitative variables are variables that are assumed to come from a finite set $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ of distinct *categories* or groups, where C_j denotes category j. Quantitative variables are often referred to as categorical or discrete variables. If the qualitative variable consists of only two categories, for example "success" or "failure", these are often coded by the numbers 0 and 1. When there are more than two possible outcomes numerical coding is often performed by introducing *dummy variables*, which are indicator variables that take the value 0 or 1 to indicate the absence or presence of the categorical effect.

When a statistical method has been trained the result is a model which can be used to *predict* the output given a new set of input variables. A set of predicted output variables is referred to as the *predictions*.

The naming of the input variables varies in different books. Possible names for the input

variables include *predictors, predictors, features, explanatory variables* or just *variables*. We will in this thesis mainly use the term *predictors* for the input variables and the *response* or *outcome* for the output variables.

In this thesis we will follow usual notation and will use uppercase letters when referring to a random variables. Observed values will be written in lowercase. An observation of a variable X is thus denoted by x. Further, matrices will be represented by bold uppercase letters. Fitted models will be represented by a hat, for example $\hat{\mathbf{y}} = \hat{f}(\mathbf{X})$ is a model fitted to the set of input variables $\mathbf{X}$, where $\hat{y}_i = \hat{f}(\mathbf{x}_i)$ is the prediction for individual i.

3.2 Logistic Regression

A logistic regression model is a variant of a generalized linear model. This section thus starts with an introduction to generalized linear models in Section 3.2.1 from which we proceed to logistic regression in Section 3.2.2. Logistic regression is one of the most important methods for analyzing categorical data and plays a prominent role in medical and biological statistics.

3.2.1 Generalized Linear Models

A generalized linear model (GLM) is a generalization of the ordinary linear regression. This generalization contains two components. The first component considers the distribution function of the response variable. The second component consists of the introduction of a link function. This presentation is based on Dobson and Garnett (2008) and Rodríguez (2016).

The Exponential Family

Assume we have n independent pairs of observations $\{(y_i, \mathbf{x}_i), i = 1, ..., n\}$. We further assume that y_i is a realization of the random variables Y_i. In a generalized linear model we allow for Y_i to follow a distribution from the exponential family. The exponential family consists of all distributions which can be written in the following way

$$f_{Y_i}(y_i|\theta) = h(y_i)c(\theta)\exp\Big(\sum_{i=1}^{k} w_i(\theta)t_i(y_i)\Big). \tag{3.1}$$

where θ is a vector of parameters, $h(y_i) \geq 0, c(\theta) \geq 0$ and $t_i(y_i)$ and $w_i(\theta)$ are real-valued functions (Casella and Berger, 2008, chap. 3).

$$Y_i \sim f_{Y_i}(y_i|\theta)$$

The exponential family includes several of the most popular distributions such as the Poisson, binomial, gamma and normal distributions.

The Link Function

Let $\theta = \pi$. Further, let μ_i denote the expected value of Y_i, $\mathrm{E}(Y_i) = \mu_i$. The *link function* is a one-to-one continuous differentiable transformation of μ_i

$$\eta_i = g(\mu_i). \tag{3.2}$$

There are many possible choices of the link function. These include the identity, log, reciprocal, logit and probit. We assume that the transformed mean follows a linear model

$$\eta_i = \boldsymbol{\beta}^T \mathbf{x}_i, \tag{3.3}$$

where

$$\mathbf{x}_i = \{1, x_{i1}, x_{i2}, ..., x_{ip}\}$$

is a row vector of p predictors for observation i and

$$\boldsymbol{\beta}^T = \{\beta_0, \beta_1, ..., \beta_p\}^T$$

is a column vector of $p + 1$ unknown coefficients. The parameter η_i is called the *linear predictor*. As the link function is one-to-one, we can obtain the mean value by finding the inverse function of the link function, g

$$\mu_i = g^{-1}(\boldsymbol{\beta}^T \mathbf{x}_i).$$

3.2.2 Logistic Regression

A logistic regression model can be used to fit a binary response variable, i.e. that the response variable can obtain one of two possible outcomes. This binary variable can be coded as a 0/1 response Y_i, where $Y_i = 1$ corresponds to class 1 (”success”), and $Y_i = 0$ corresponds to class 2 (”failure”). Now the probability for an observation to come from class 1 (probability of success) is $\pi_i(\mathbf{x}_i)$ and the probability for an observation to come from class 2 (probability of failure) is $1 - \pi_i(\mathbf{x}_i)$. A logistic regression model can thus be expressed as

$$Y_i = \begin{cases} 1, & \text{with probability } P(Y_i = 1 | X_i = \mathbf{x}_i) = \pi_i(\mathbf{x}_i) \\ 0, & \text{with probability } P(Y_i = 0 | X_i = \mathbf{x}_i) = 1 - \pi_i(\mathbf{x}_i) \end{cases},$$

Assume we have n independent observations of $Y, \{Y_1, Y_2, .., Y_n\}$. The probability mass function for observation $Y_i, i = 1, .., n$ is given by

$$f_{Y_i}(y_i|\pi_i) = \binom{1}{y_i}\pi_i^{y_i}(1-\pi_i)^{1-y_i}.$$

This is the probability mass function of the binomial distribution with one trial and can equivalently be written as

$$f_{Y_i}(y_i|\pi_i) = \binom{1}{y_i}(1-\pi_i)\exp\Big(y_i \log\Big(\frac{\pi_i}{1-\pi_i}\Big)\Big). \tag{3.4}$$

We can recognize that the binomial distribution is a member of the exponential family of distributions defined previously, by comparing Equation (3.4) with Equation (3.1) and identifying each of the functions that make up the equation

$$\begin{aligned} h(y_i) &= \binom{1}{y_i} \\ c(\pi_i) &= (1-\pi_i), \quad 0 < \pi_i < 1 \\ w_1(\pi_i) &= \log\Big(\frac{\pi_i}{1-\pi_i}\Big) \\ t_1(y_i) &= y_i. \end{aligned}$$

When fitting a logistic regression model, we choose the logit function to be our link function

$$\eta_i = \text{logit}(\pi_i) = \log\frac{\pi_i}{1-\pi_i} = \boldsymbol{\beta}^T\mathbf{x}_i,$$

as this assures that the probability lies in the interval between zero and one, $\pi_i \in [0, 1]$.

3.2.3 Interpretation of the Logistic Regression Model

The fraction $\frac{\pi_i}{1-\pi_i}$ is called the *odds ratio* and is the probability of success divided by the probability of failure. We further have that

$$\pi_i = \text{logit}^{-1}(\boldsymbol{\beta}^T\mathbf{x}_i) = \frac{e^{\boldsymbol{\beta}^T\mathbf{x}_i}}{1+e^{\boldsymbol{\beta}^T\mathbf{x}_i}}.$$

This formulation of the logistic regression thus refers directly to the probability of success.

Interpretation of β_l

The logistic regression curve is S-shaped. The parameter β_l, where $l \in \{0, 1, ..., p\}$, determines the rate of increase or decrease of the S-shaped curve. The sign of β_l indicates whether the curve ascends or descends, and the rate of change increases as $|\beta_l|$ increases [chap. 4](Agresti, 1996).

Odds Ratio Interpretation

The odds of $Y = 1$ (the odds of a success) for an individual i is

$$\begin{aligned}\frac{\pi_i(\mathbf{x}_i)}{1-\pi_i(\mathbf{x}_i)} &= e^{\boldsymbol{\beta}^T \mathbf{x}_i} \\ &= e^{\beta_0}(e^{\beta_1})^{x_{i1}}(e^{\beta_2})^{x_{i2}} \dots (e^{\beta_p})^{x_{ip}}\end{aligned}$$

The interpretation is as follows: the odds increase multiplicatively by e^{β_l} for every one-unit increase in x_{il}. Thus the odds at level $x_{il} + 1$ is equal the odds at x_{il} multiplied by e^{β_l}. Thus when $\beta_l = 0$, $e^{\beta_l} = 1$ and the odds does not change as x_{il} changes.

3.2.4 Case-Control Studies

Case-control studies are popular in the biomedical sciences. It is a retrospective sampling design as it looks into the past. For example: A scientist can instead of waiting many years to observe which subjects will suffer from a MI, take out a sample of subjects which actually have suffered from a MI. These subjects are coded by $Y = 1$ and are referred to as *cases*. One in addition takes out subjects which have not suffered from a MI. These are coded by $Y = 0$ and are referred to as *controls*.

For all subjects the value of X is observed. Evidence exists of an association between X and Y if the distribution of X values differs between cases and controls (Agresti, 1996, chap. 4). In such a case the predictors X rather than the response variable Y are random. As logistic parameters refer to the odds and odds ratios, one can fit such a model to retrospective data and estimate the effects in case-control studies. This is a property of the logistic regression model which is not shared by other models for binary responses. Specifically, logit link for this reason provides an important advantage over links such as the probit.

3.2.5 Fitting Logistic Regression Models

This section in largely based on Hastie et al. (2009, chap. 4). The maximum likelihood estimator for the parameters $\boldsymbol{\beta}$ cannot be obtained in a closed form and therefore need to be estimated through the use of nonlinear optimization methods. The log-likelihood function is

$$\begin{aligned}
l(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \Big(y_i \log \pi_i + (1 - y_i) \log(1 - \pi_i) \Big) \\
&= \sum_{i=1}^{n} \Big(y_i \log \pi_i + \log(1 - \pi_i) - y_i \log(1 - \pi_i) \Big) \\
&= \sum_{i=1}^{n} \Big(y_i \log \big(\frac{\pi_i}{1 - \pi_i} \big) + \log(1 - \pi_i) \Big) \\
&= \sum_{i=1}^{n} \Big(y_i \boldsymbol{\beta}^T \mathbf{x}_i + \log \big(1 - \frac{e^{\boldsymbol{\beta}^T \mathbf{x}_i}}{1 + e^{\boldsymbol{\beta}^T \mathbf{x}_i}} \big) \Big) \\
&= \sum_{i=1}^{n} \Big(y_i \boldsymbol{\beta}^T \mathbf{x}_i + \log \big(\frac{1}{e^{\boldsymbol{\beta}^T \mathbf{x}_i} - 1} \big) \Big) \\
&= \sum_{i=1}^{n} \Big(y_i \boldsymbol{\beta}^T \mathbf{x}_i - \log(1 + e^{\boldsymbol{\beta}^T \mathbf{x}_i}) \Big),
\end{aligned}$$

where $\boldsymbol{\beta}$ is a vector of $p+1$ coefficients for the parameters. To maximize the log-likelihood function we set the derivatives equal to zero

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \beta_j} = \sum_{i=1}^{n} \mathbf{x}_i \Big(y_i - \frac{e^{\boldsymbol{\beta}^T \mathbf{x}_i}}{1 + e^{\boldsymbol{\beta}^T \mathbf{x}_i}} \Big) = 0, \quad j = 0, ..., p.$$

We have $p+1$ nonlinear equations in $\boldsymbol{\beta}$. To solve these equations one can use the Newton-Raphson algorithm, which requires the second-derivative of the log-likelihood function, called the Hessian

$$H = \frac{\partial^2 l(\boldsymbol{\beta})}{\partial \beta_j \partial \beta_k} = -\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T \frac{e^{\boldsymbol{\beta}^T \mathbf{x}_i}}{1 + e^{\boldsymbol{\beta}^T \mathbf{x}_i}} \Big(1 - \frac{e^{\boldsymbol{\beta}^T \mathbf{x}_i}}{1 + e^{\boldsymbol{\beta}^T \mathbf{x}_i}} \Big), \quad j = 0, ..., p, \quad k = 0, ..., p.$$

The Newton-Raphson algorithm starts with an initial guess, often $\boldsymbol{\beta} = \{0, \ldots, 0\}$, and solves the equation iteratively by updating $\boldsymbol{\beta}$ in each step. One single update of the

algorithm is

$$\boldsymbol{\beta}^{k+1} = \boldsymbol{\beta}^k - H^{-1}\frac{\partial l(\boldsymbol{\beta}^k)}{\partial \boldsymbol{\beta}^k},$$

where k is the step counter. This trial value is updated by each iteration of the algorithm.

Iteratively Reweighted Least Squares

Introducing a n-dimensional column vector of observations $\mathbf{y}$, a n-dimensional column vector $\boldsymbol{\pi}$ of fitted probabilities π_i and a diagonal $n \times n$ matrix $\mathbf{W}$ of weights $\frac{e^{\boldsymbol{\beta}^T \mathbf{x}_i}}{1+e^{\boldsymbol{\beta}^T \mathbf{x}_i}}\left(1 - \frac{e^{\boldsymbol{\beta}^T \mathbf{x}_i}}{1+e^{\boldsymbol{\beta}^T \mathbf{x}_i}}\right)$, the Newton step can be written in matrix form

$$\begin{aligned}\boldsymbol{\beta}^k &= \boldsymbol{\beta}^{k-1} + (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - \mathbf{p}) \\ &= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{z},\end{aligned}$$

where we have

$$\mathbf{z} = \mathbf{X}\boldsymbol{\beta}^{k-1} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})$$

and

$$\begin{aligned}\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \mathbf{X}^T(\mathbf{y} - \mathbf{p}) \\ \frac{\partial^2 l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} &= -\mathbf{X}^T\mathbf{W}\mathbf{X}.\end{aligned}$$

The Newton step can thus be written as a weighted least squares step, where $\mathbf{z}$ is the adjusted response. Since at each step of the algorithm we solve the weighted least squares problem, this is an *iteratively reweighted least squares algorithm*.

3.2.6 The Wald Test

The Wald test can be used to test the significance of one particular coefficient. Let

$$H_0 : \beta_l = 0, \quad H_1 : \beta_l \neq 0,$$

where β_l is the coefficient of predictor l. Under certain regularity conditions, the maximum likelihood estimator $\hat{\beta}_l$ has approximately in large samples a normal distribution

(Rodríguez, Rodríguez). Under H_0 we have mean 0 and variance $\text{Var}(\hat{\beta}_j)$. The z-statistic is thus given by

$$z = \frac{\hat{\beta}_l}{\sqrt{\hat{\text{Var}}(\hat{\beta}_l)}}.$$

3.2.7 Akaike Information Criterion (AIC)

The Akaike information criterion can be used to choose between two candidate GLM models. The formula for calculating the AIC for a logistic regression model is

$$AIC = -\frac{2}{n} \cdot \text{loglik} + 2 \cdot \frac{p}{n}$$

where n is the number of observations and p is the number of model parameters. Further, loglik is the maximized log-likelihood

$$\text{loglik} = \sum_{i=1}^{n} \log \Pr_{\hat{\boldsymbol{\beta}}}(y_i)$$

where $\hat{\boldsymbol{\beta}}$ is the maximum-likelihood estimate of the parameter $\boldsymbol{\beta}$. The role of p is to penalize models with parameters, since many parameters may lead to poor predictive power due to overfitting. For logistic regression we have the following expression for the loglik function

$$\text{loglik} = \sum_{i=1}^{n} \Big(y_i \hat{\boldsymbol{\beta}}^T \mathbf{x}_i - \log(1 + e^{\hat{\boldsymbol{\beta}}^T \mathbf{x}_i}) \Big).$$

The value of the AIC does not say much alone, but when comparing two candidate models, the one with the lower AIC value is the one expected to give the better fit of the two.

3.2.8 Forward Stepwise Selection

If there is a large number of predictors in our logistic regression model, the model becomes less interpretable and can overfit the data. The most typical case is, that not all predictors are significant. *Forward stepwise regression* is an algorithm, which adds one predictor to the model at each step. We here assume we have observations from an a logistic model defined by $\eta = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$. The procedure is, given a data set of p predictors:

- Fit a logistic model to the data set, including the intercept only. This gives the model

$$\eta = \beta_0.$$

- Create p new models, where in each model, only the l-th predictor is included, where $l = 1, 2, ..., p$

$$\begin{gathered}\beta_0 + \beta_1 x_1 \\ \beta_0 + \beta_1 x_2 \\ \vdots \\ \beta_0 + \beta_1 x_l \\ \vdots \\ \beta_0 + \beta_1 x_p\end{gathered}$$

  Given, one of the models has a lower AIC score than the model with only β_0, choose the model with the predictor x, denoted by x_*, corresponding to the lowest AIC score.

- Create p new models, where in each model, only the l-th predictor is included

$$\begin{gathered}\beta_0 + \beta_1 x_* + \beta_2 x_1 \\ \beta_0 + \beta_1 x_* + \beta_2 x_2 \\ \vdots \\ \beta_0 + \beta_1 x_* + \beta_2 x_l \\ \vdots \\ \beta_0 + \beta_1 x_* + \beta_2 x_p\end{gathered}$$

  Again, choose the model corresponding to the smallest AIC score, given there is an AIC score lower than the model with β_0 and β_1 only.

- Follow the procedure until a lower AIC score cannot be obtained by the addition of a predictor.

3.3 Classification Trees

Classification and regression trees (CART) make partitions of the space of the predictor variables into a set of non-overlapping rectangles. Each rectangle has a corresponding simple model: a constant for a regression problem and a category for a classification problem. The rectangles are translated into a schematic figure, which has the form of a tree.

Algorithm 1: Growing a classification tree.

Input: Data set **Z**
Minimum node size n_{min}

Make a partition the data set according to the Gini splitting rule to obtain two regions R_1 and R_2;
$M = 2$, the number of terminal nodes/regions;
n_m = Number of observations in R_m, $m = 1, 2$;
while $\exists\ n_m \geq n_{min}$ **do**
    for $m = 1$ *to* M*:* **do**
        n_m = Number of observations in region R_m;
        if $n_m \leq n_{min}$ **then**
            break;
        end
        else
            Make a partition of R_m using the Gini splitting rule;
        end
    end
    M = The number of terminal nodes/regions;
end

Output: Tree T.

The CART method follows a top-down approach, that is known as *binary splitting*. When growing a tree, we begin at the top of the tree. The top node is denoted by the *root node* and represents the first partition of the predictor space. That is, it corresponds to a partition of the predictor space into two distinct, rectangular regions. A root node has no *internal nodes* but two *terminal nodes*. We follow by successive binary splits on the terminal nodes, creating new branches. The terminal nodes become internal nodes if a split is created. At the same time two new terminal nodes are created. The algorithm is a *greedy* algorithm, because the best split is made at the particular step. It searches for the local optimum and not the global optimum. The algorithm never looks ahead and picks a split that will lead to a better tree in the future. We proceed the procedure of binary splitting, until each region R_i corresponding to the terminal node i has fewer than a minimum number of observations, $n_{\min}$. In this final tree, the terminal nodes contain the predicted category. This predicted category (class) is the most commonly occurring class of training observations in the region (node) to which it belong. The This procedure is summarized in Algorithm 1. See Figures 3.1 and 3.2 [1] for an illustration of the tree-growing process.

3.3.1 Gini Splitting Criterion

Suppose we have a data set $\mathbf{Z} = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)$ consisting of n observations. Further, suppose we have p variables such that $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{ip}), i = 1, ..., n$. Now, starting with the full data set, consider a splitting variable j and a splitting point s. Define the a pair of half-planes

$$R_1(j, s) = \{X | X_j \leq s\} \text{ and } R_2(j, s) = \{X | X_j > s\}.$$

The splitting is done by choosing the values of s and j such that the impurity of the two regions R_1 and R_2 is minimized. In other words, we want the homogeneity of each region to be as large as possible.

To evaluate the homogeneity of a region an *impurity* measure is needed. There are many possible choices for the impurity measure. These include the Gini index, misclassification error and the cross-entropy or deviance. In this thesis we will use the Gini impurity measure, which is defined, for node m, in the following way

$$i(m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

[1] Tree figures have been created using the `tree` package (Ripley, 2016).

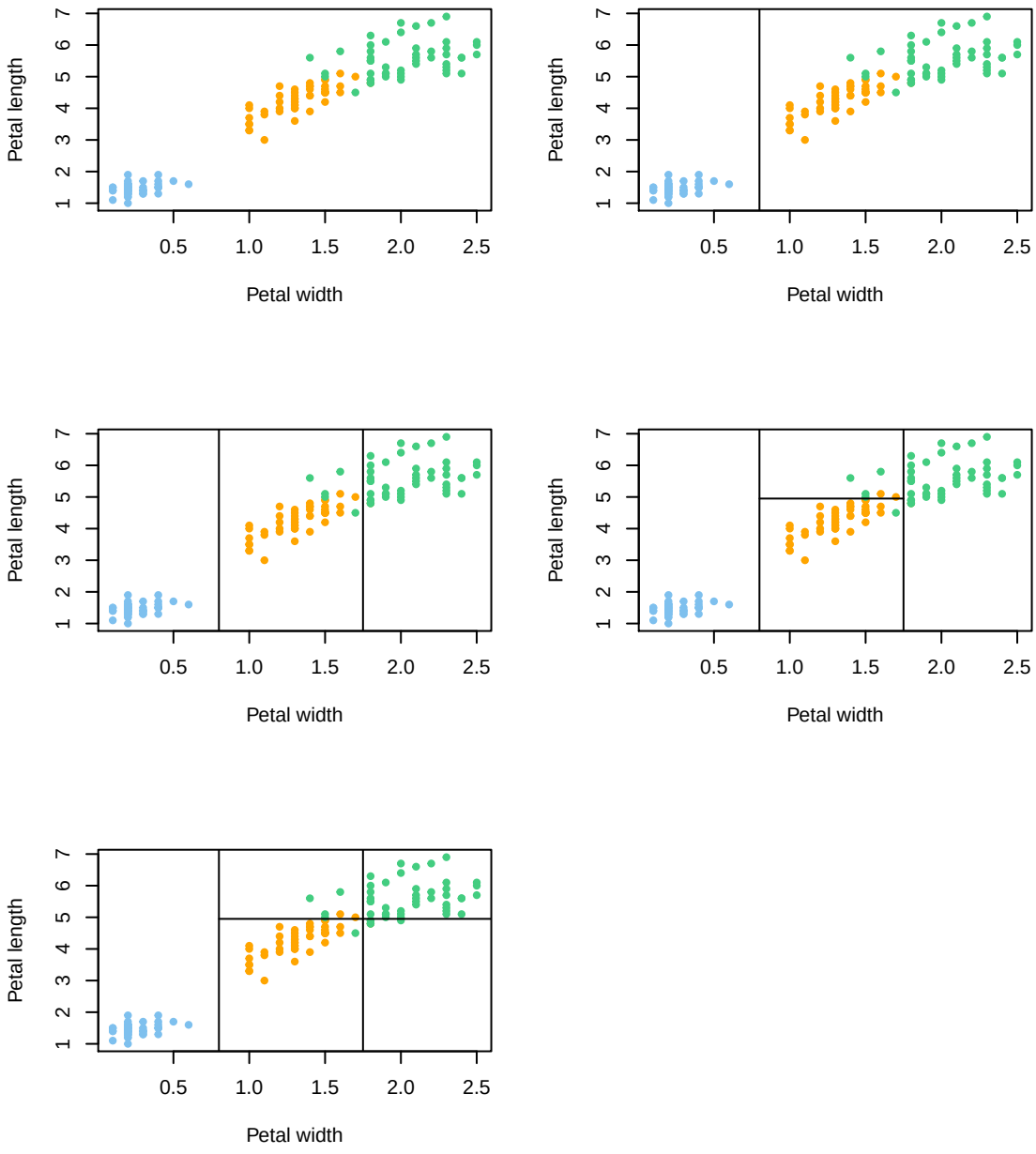


Figure 3.1: Classification of iris plants. The top left figure shows a scatter plot of three different iris plants as a function of Petal width and Petal length. There exist three different species of iris plants. These are called setosa, versicolor and virginica. The color coding is as follows: blue = setosa, orange = versicolor, green = virginica. The following figures (going from left to right, top to bottom) show the steps when growing a classification tree to the data. At each step a binary partition is performed, which is represented by a black line. Each partition corresponds to a node in the classification tree, see Figure 3.2.

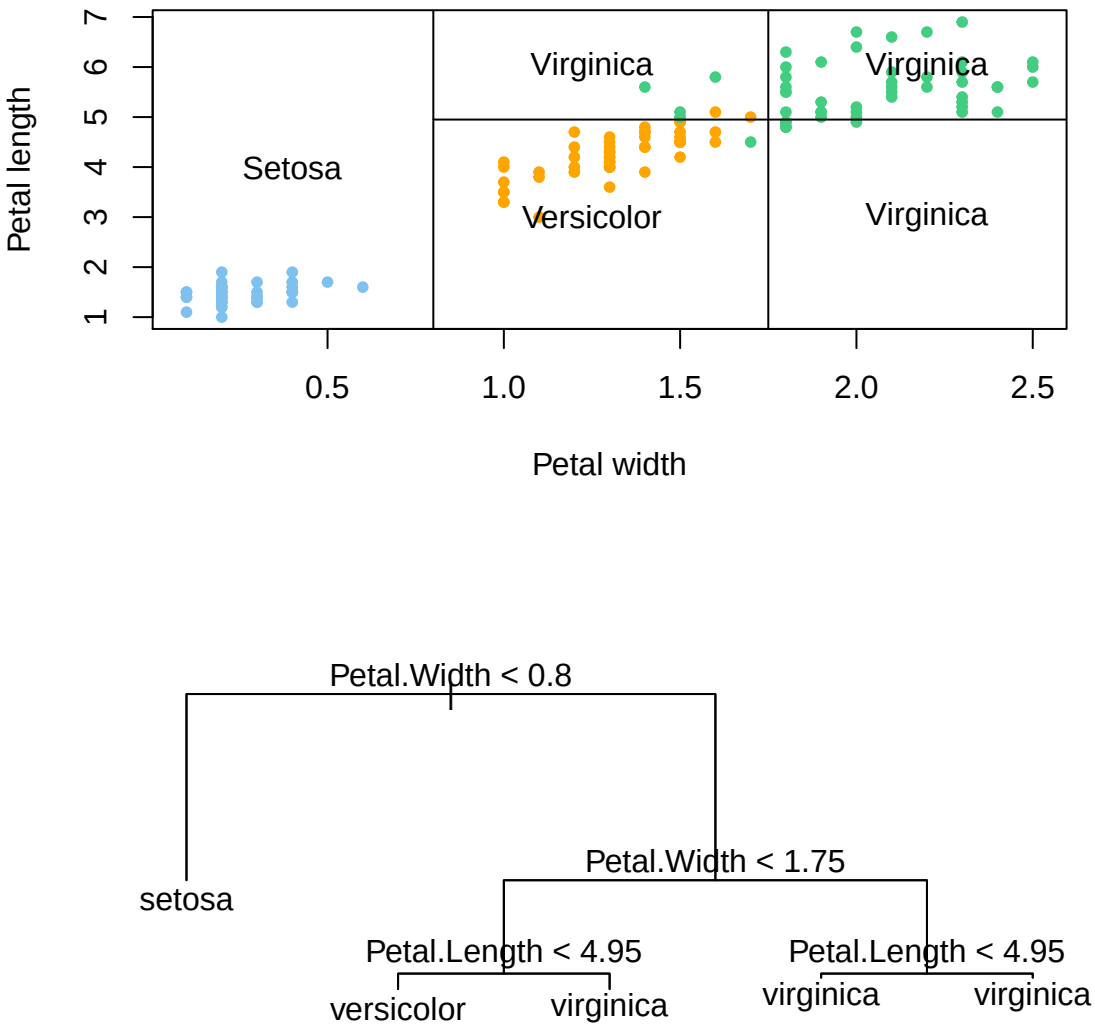


Figure 3.2: Classification of iris plants continued. The top figure shows the final step of the binary partitions, with corresponding specie labels. The bottom figure shows the corresponding classification tree. The text at each node show the condition for choosing the left branch. If the condition is not satisfied, the right branch is to be chosen. The terminal nodes show the classification of the species.

where

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

represents the proportion of class k observations in region R_m with N_m observations.

When growing a classification tree the algorithm evaluates the reduction in impurity, to decide on where to make a split, using this formula

$$\Delta i(m) = i(m) - \frac{N_1}{N_m} i(m_1) - \frac{N_2}{N_m} i(m_2)$$

Here $i(m)$ is the impurity of region m before a split $i(m_1)$ and $i(m_2)$ are the impurities of the regions R_1 and R_2 after a split has been made. Furthermore, N_m, N_1 and N_2 are the number of observations in regions R_m, R_1 and R_2. This reduction in impurity is calculated for all p variables and for all possible split points s. The split chosen is the one for which the reduction in impurity is highest, which can symbolically be written as

$$\underset{x_j \leq x_j^s j=1,...,N}{\operatorname{argmax}} [\Delta i(m)]$$

When looking at the tree-growing process and the final classification tree fitted to an iris data set, respectively, one can notice that there is an excessive branch. In Figure 3.1 one can see that the last step of the algorithm divides the red region into two new regions, creating a new node. It seems that this does not make much sense, as each of the new regions receive the same class: virginica. This is because the Gini index has found one of the two regions to be more pure than the other.

3.3.2 Making Predictions

We now assume we have grown a classification tree T and are given a new observation $\mathbf{x}_i^* = \{x_{1i}^*, x_{2i}^*, ..., x_{pi}^*\}$. To make a prediction for y_i, we start at the top of the tree (at the root node). By following the splitting criterion at this node we enter either the left or right child node. We successively proceed by following the splitting criterion at each internal node until we have reached a terminal node. The class label at this terminal node is the class prediction for $\mathbf{x}_i^*$.

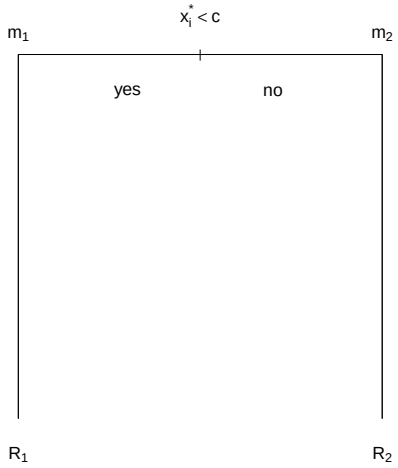


Figure 3.3: A tree stump.

Figure 3.3 illustrates this procedure, where there is only one predictor x_i^* and two possible classes m_i, $i = 1, 2$. The predictions are in this example given by

$$y_i = \begin{cases} m_1, & x_i^* < c \\ m_2, & x_i^* \geq c. \end{cases}$$

Sometimes instead of obtaining a classification for a new prediction, the class probabilities are of more interest. These can be estimated by the class proportions in the terminal node. Written symbolically, the estimated probability that a new observations x_i^* will be classified as 1 is

$$\hat{\pi}_{m_1}(x_i^*) = \frac{\#\text{observations in node classified as } m_1}{\#\text{observations in node}}.$$

3.3.3 Pruning

Tree size is a tuning parameter governing the models complexity (Hastie et al., 2009, chap. 9). If one grows a tree with many predictors, the interpretability of the tree can be reduced. Additionally, the tree might overfit the data and thus have a reduced predicting performance. For trees the best strategy (James et al., 2013a, chap. 8) is to fit a very large

tree, T_0, and then to prune it back in order to obtain a sub-tree. Instead of trying each possible sub-tree [2] and using cross-validation to evaluate the best sub-tree, cost-complexity pruning can be used. Cost-complexity pruning, or weakest-link pruning was proposed by Breiman et al. (1984) and returns a small set of sub-trees for consideration. One considers a sequence of trees indexed by a non-negative tuning parameter α. For each value of α there corresponds a sub-tree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|$$

is minimized. Here $|T|$ is the number of terminal nodes in T, R_m is the region corresponding to terminal node m and N_m is the number of observations in R_m. $Q_m(T)$ is the impurity measure, which we in this thesis have chosen to be the Gini index.

We show how this works in detail: Let T_0 denote the full tree and introduce for simplicity:

$$Q_\alpha = Q(T) + \alpha|T|$$

where

$$Q(T) = \sum_{m=1}^{|T|} N_m Q_m(T).$$

Thus, $Q(T)$ is the total impurity of the tree T while Q_α is the Ts impurity penalized by the term $\alpha|T|$ for the number of terminal nodes. We take the full tree as our starting point. Then for each subtree $T_k, k = 1, 2, ..., m$ the increase in $Q(\cdot)$ per decrease in the number of terminal nodes can be calculated

$$\begin{aligned}
T_1: &\quad \frac{Q(T_1) - Q(T_0)}{|T_0| - |T_1|} = \alpha_1 \\
T_2: &\quad \frac{Q(T_2) - Q(T_0)}{|T_0| - |T_2|} = \alpha_2 \\
&\quad \vdots \\
T_m: &\quad \frac{Q(T_m) - Q(T_0)}{|T_0| - |T_m|} = \alpha_m
\end{aligned}$$

[2] There are $\approx 1.5^l$ rooted subtrees for a binary tree with l leaves (terminal nodes) (Breiman et al., 1984)

These are nested subtrees T_k of T_0 such that each is optimal for a range of α, and there hence are values

$$-\infty = \alpha_0 < \alpha_1 < \alpha_2 < \cdots < \infty$$

such that T_i is an optimal tree for $\alpha \in [\alpha_i, \alpha_{i+1})$ (Ripley, 1995). These steps have been summarized in Algorithm 2.

Figure 3.4 shows a complex classification tree fit to the `Carseats` dataset from the `ISLR` library (James et al., 2013b). The top figure shows the full tree, while the bottom figure shows the pruned tree, where the optimal number of terminal nodes have been found by cross-validation. These plots show that pruning can be essential for the interpretability of the tree.

Algorithm 2: Tree pruning

Input: The original tree T_0
Data set $\mathbf{Z}$

1 Perform cost-complexity pruning on T_0 to obtain a sequence of best sub-trees as a function of α: $T_0, T_1, ..., T_m$
2 Use K-fold cross-validation to choose α. Divide the training set $\mathbf{Z}$ into K folds
3 **for** $k = 1$ *to* K : **do**
4 | Grow a tree T_k on the data $\mathbf{Z}^{-k}$
5 | Perform cost-complexity pruning on T_k as a function of α: $T_k^0, T_k^1, ..., T_k^m$
6 | Evaluate the prediction error on $\mathbf{Z}^k$ as a function of α
7 **end**
8 Average the results for each value of α. Let $\hat{\alpha}$ denote the α that minimizes the average prediction error

Output: The sub-tree $T_i, i \in \{0, ..., m\}$ that corresponds to $\hat{\alpha}$. This is the pruned tree.

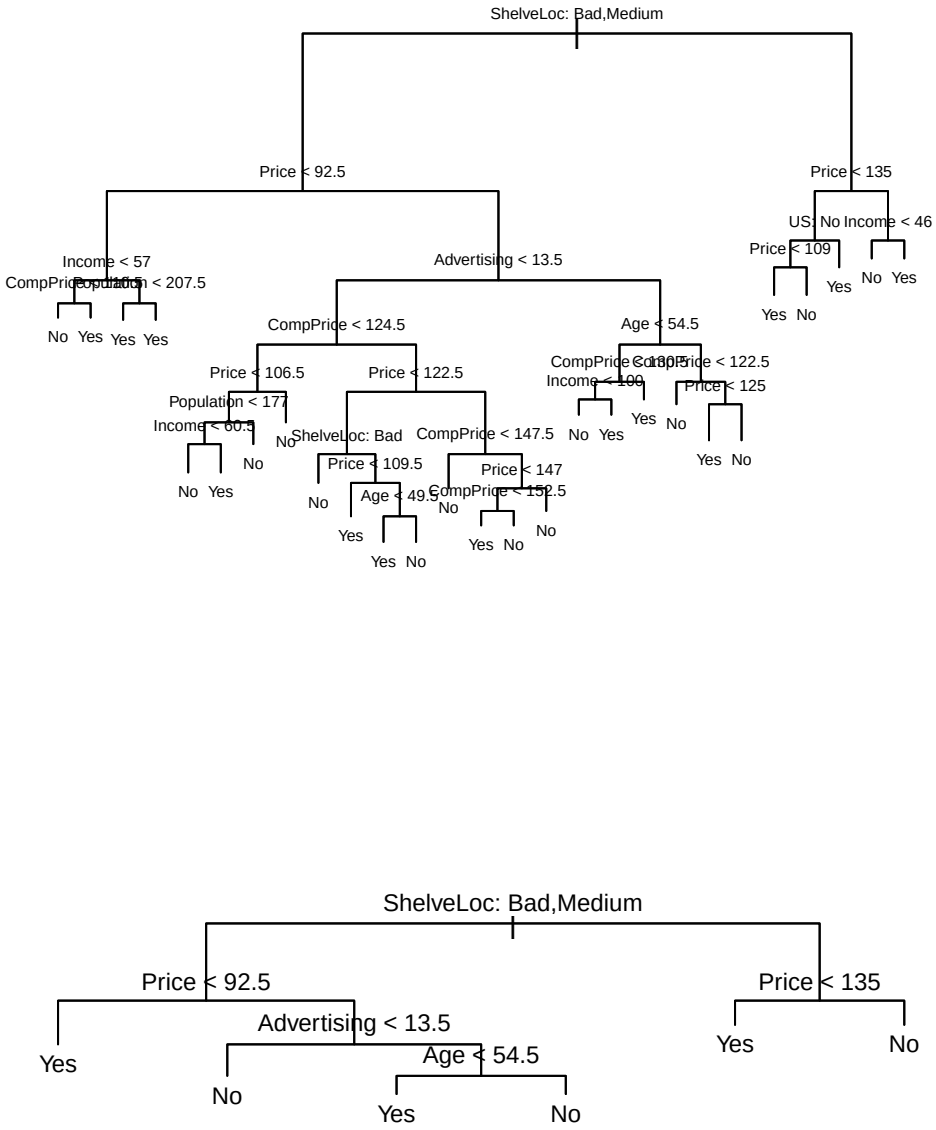


Figure 3.4: Pruning a classification tree. Top: this figure shows the full classification tree fit to the `Carseats` dataset from the `ISLR` library. The tree predicts if unit sales (in thousands) at a particular location will be high (> 8) or low (otherwise). Bottom: the pruned classification tree, where the optimal number of terminal nodes has been found by cross-validation. At each step of the pruning algorithm an internal node is collapsed, giving a reduction in the terminal nodes (and regions R_i).

3.3.4 Benefits and Limitations

Classification trees have several benefits (James et al., 2013a, chap. 8):

- Trees are easy to interpret.
- Trees can easily be displayed graphically.
- Trees can easily handle qualitative predictors without the need to create dummy variables.

There are however some limitations of classification trees that one needs to be aware of:

- Trees do not have the same level of predictive accuracy as many other statistical models.
- Trees can be very unstable. A small change in the data can cause a large change in the final tree.
- The classes assigned to terminal nodes can be unstable too.

There are methods which can improve the predictive performance of trees, by combining several trees into one model. However this comes at the cost of interpretability. Examples of such methods are bagging, boosting and random forests. We will discuss bagging and random forests in detail in Chapter 4.

3.4 Model Evaluation

Different models require different tools for model assessment and evaluation. This section aims at giving an introduction to the tools used in this thesis.

3.4.1 Model Assessment and Selection

When training a statistical method, it is important to be aware of the goal one has in mind. Hastie et al. (2009, chap. 7) have identified two different goals:

Figure 3.5: Model assessment and selection.

- *Model selection:* estimating the performance of different models in order to choose the best one.

- *Model assessment:* having chosen a final model, estimating its prediction error on new data.

The *best model* depends on whether the aim is to understand the underlying interactions of the dataset or to best predict the outcome for new observations. Assume our aim is to obtain a model with the best possible predictive power. The procedure for making a prediction should be as follows, given the data set is large enough: One divides the data into three separate parts: a *training* set, consisting of about 50% of the data, a *validation* set, with about 25 % of the data and a *test* set, with about 25 % of the data. Now, one uses the training set to fit models. There are often many candidate models, and one needs to choose the best one among these candidates. This is called model selection. For this purpose one uses the validation set to estimate the prediction errors of the candidate models. The model with the lowest prediction error and the best tradeoff between the variance and bias is chosen. The final step is to use the test set for model assessment. The test set contains observations which have been left out during model selection. By following this procedure, one gets the most correct estimate of the prediction error.

3.4.2 The Variance-Bias Tradeoff

When training a statistical model it is important to be aware of the interplay between the bias, variance and model complexity. One also needs to be aware of the difference between the test error and the training error. This section will discuss these issues and will start by introducing the bias-variance decomposition as it illustrates the problem of model selection well.

The Variance-Bias Decomposition

The variance-bias decomposition can be done for many different models. We here base the analysis on a simple linear regression model. Assume we have fitted a regression line, $Y = f(X) + \varepsilon$ to a set of independent observation pairs $\{(x_i, y_i)\}_{i=1}^{n}$. Here $\varepsilon \sim N(0, \sigma_\varepsilon^2)$ is an unobserved random variable that adds noise to the relationship between the input and output variables, and is called the random error. The fitted regression line is denoted by $\hat{f}$. Using squared-error loss, we can derive an expression for the expected prediction error

of $\hat{f}$ at a new input point x_0 (Hastie et al., 2009, chap. 7)

$$\begin{aligned}
\text{Error}(x_0) &= \text{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \text{E}[Y^2 + \hat{f}(x_0)^2 - 2Y\hat{f}(x_0)] \\
&= \text{E}[Y^2] + \text{E}[\hat{f}^2(x_0)] - E[2Y\hat{f}(x_0)] \\
&= \text{Var}[Y] + \text{E}[Y]^2 + \text{Var}[\hat{f}(x_0)] + \text{E}[\hat{f}(x_0)]^2 - 2\text{E}[Y]\text{E}[\hat{f}(x_0)] \\
&\overset{\text{E}[Y]=f(x_0)}{=} \text{Var}[Y] + f(x_0)^2 + \text{Var}[\hat{f}(x_0)] + \text{E}[\hat{f}(x_0)]^2 - 2f(x_0)\text{E}[\hat{f}(x_0)] \\
&= \text{Var}[Y] + \text{Var}[\hat{f}(x_0)] + (f(x_0) - \text{E}[\hat{f}(x_0)])^2 \\
&= \sigma_\varepsilon^2 + \text{Var}[\hat{f}(x_0)] + \text{Bias}^2[\hat{f}(x_0)].
\end{aligned}$$

The first term in the last line of the expression is the variance of observations. It is called the irreducible error as it is always present unless we have measurements without error, i.e. $\sigma_\varepsilon^2 = 0$. The second term is the variance of the prediction at x_0 or the expected deviation around the mean at x_0. If the variance is high, there is a large uncertainty associated with the prediction. The third term is the the squared bias. The bias gives an estimate of how much the prediction differs from the true mean. Hence if the bias is low the model gives a prediction which is close to the true value.

The Variance-Bias Tradeoff

When training a statistical model the aim is often to obtain the most predictive model, i.e. the model that gives the most accurate predictions for new observations. There are often many candidate models, and the task of the statistician is to decide which model to chose by making model assessment.

The observations used to train the statistical method make up the training set. The training error is the average loss over the training sample. As the complexity of a model increases the model becomes more adaptive to underlying structures and the training error falls. However, the variance increases. This means that new observations will be predicted with a higher uncertainty. Hence as the training error falls, the test error first falls but often starts to increase as the complexity is too high. This phenomena is illustrated in Figure 3.6. The test error is the prediction error over an test sample. The test sample will have new observations which were not used when fitting the model. One wants the model to capture important relationships between the input and output variables, else we will *underfit*. However, if the model fits the training set too well, one *overfits*, and the model has a reduced predictive power for a test set. This tradeoff in selecting a procedure with the right amount

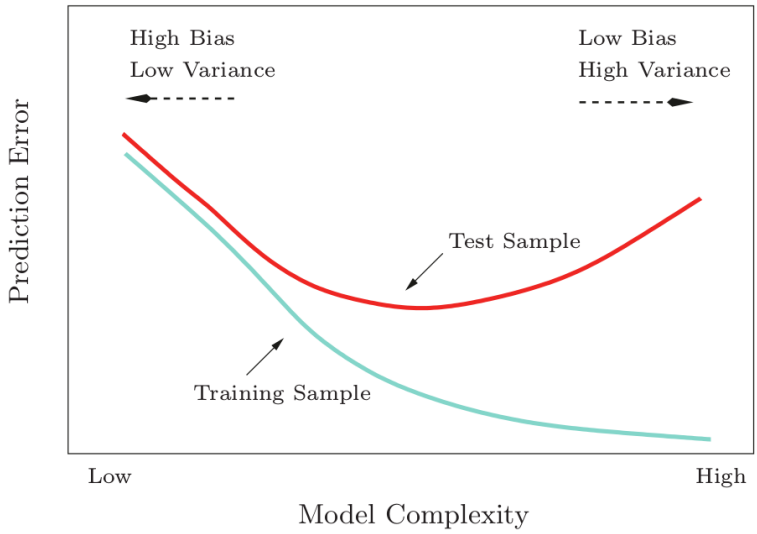


Figure 3.6: Test and training error as a function of model complexity. This figure has been taken from (Hastie et al., 2009, chap. 2), with permission from Springer.

of complexity is the *variance-bias tradeoff*.

Many statistical methods have a *complexity parameter* which can be used to balance between the variance and the bias of the model. In the case of logistic regression, the complexity parameter is the number of variables. A model with more variables is more complex, has a lower bias and higher variance. In the case of CART the complexity parameter is the number of nodes. A smaller tree (with fewer nodes) has a larger bias but smaller variance than a larger tree fit to the same data.

We will in Chapter 4 look at techniques called *bagging* and *random forests*, which reduce the variance while keeping the bias fixed, and hence reduce the expected prediction error.

3.4.3 ROC Curves

A receiver operating characteristics (ROC) curve is a tool widely used to evaluate classifiers in biomedical and bioinformatics applications.

		Predicted Condition	
		Positive	Negative
True Condition	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

Table 3.1: Confusion matrix.

A *threshold* parameter in a binary classifier, T, is a value which defines a classification rule. For instance: Classify all observations $Y > T$ as "ill".

Assume we want to classify a patient as "ill" or "not ill" based on the result of a clinical test. The *Sensitivity* is the probability that a test result will be positive when the disease is present. It is estimated by the proportion of correctly classified positive (ill) observations (patients).

$$\begin{aligned} &\text{Sensitivity} = P(\text{positive test} \mid \text{ill}) \\ &\textit{Estimated} \text{ sensitivity} = \frac{\#\text{True Positive}}{\#\text{Condition Positive}}. \end{aligned}$$

Specificity is the probability that a test result will be negative when the disease is absent. It is estimated by the proportion of correctly classified negative (not ill) observations.

$$\begin{aligned} &\text{Specificity} = P(\text{negative test} \mid \text{not ill}) \\ &\textit{Estimated} \text{ specificity} = \frac{\#\text{True Negative}}{\#\text{Condition Negative}}. \end{aligned}$$

A ROC curve plots the sensitivity vs 1-specificity as the threshold is moved over the range of all possible values (Gerds et al., 2008). Table 3.1 shows the confusion matrix which gives a schematic explanation of the terms. An ideal ROC curve will hug the top left corner, while a straight line represents a classifier with a random guess of the outcome. Figure 3.7[3] shows a ROC curve in which the classifier is somewhat better than random guessing. ROC curves are useful for comparing different classifiers, since they take into account all possible thresholds (James et al., 2013a, chap. 9).

[3]Figure created using the `rpart` (Therneau et al., 2017) and `pROC` (Robin et al., 2011) packages.

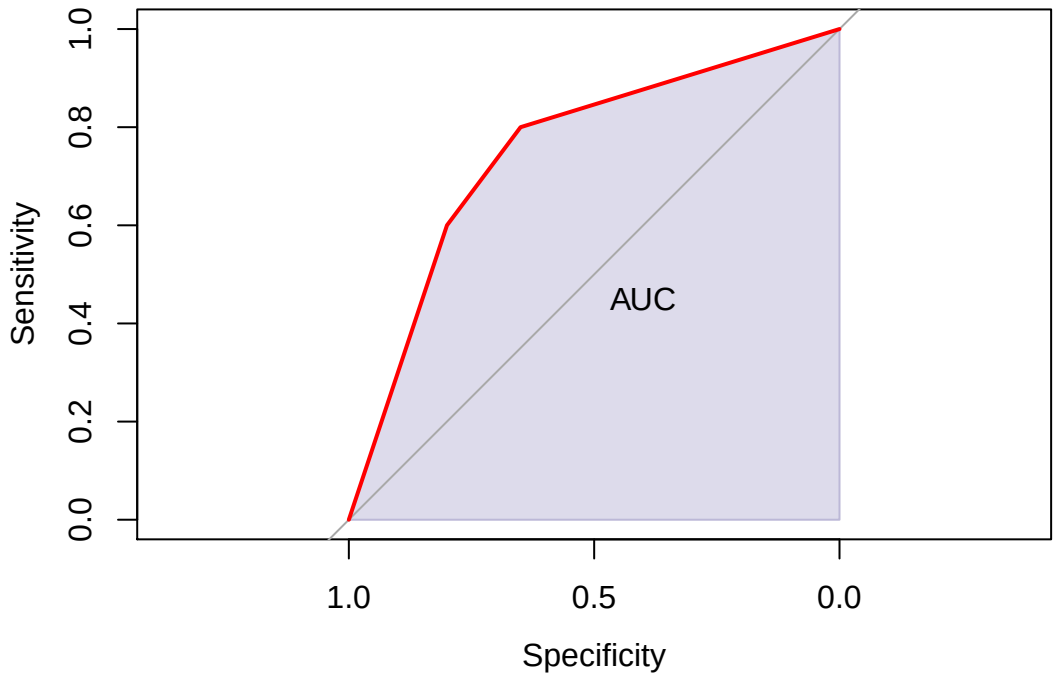


Figure 3.7: The red line shows the ROC curve. The area under the ROC curve is the AUC score.

Comparison of ROC Curves Through the AUC Score

There is a number of ways in which competing ROC curves can be compared, for example by the use of optimal points (Gerds et al., 2008) or the AUC score. In this thesis, the AUC score has been chosen as a summary statistic. The AUC score is the area under the ROC curve, as shown in Figure 3.7. It ranges between the values 0 and 1, where a higher value indicates a better classifier. An AUC score equal to 1 would imply that all observations are correctly classified. The advantage of the AUC score is that the whole risk spectrum is taken into consideration when comparing different classifiers.

3.4.4 K-Fold Cross-Validation

When training a statistical model, the ideal scenario would be to have enough data such that one can set aside a part of the data when fitting the statistical model. The observations used to fit the model are referred to as the *training* set, while the observations left out are referred to as the *test* set. The statistical model could thus be fitted by using the training set. The predictive power of this model could thereafter be evaluated by using the test set. However, typically data is scarce. K-fold cross validation is a method which estimates the expected prediction method.

In cross-validation one uses the same data set to train and evaluate a model. In K-fold cross-validation the procedure is repeated K times. Assume we have n observations. These are split into K random parts. These parts are approximately of equal size and are called folds. The number K represents the number of folds. One fits the model to $K-1$ of the parts. These parts together make up the training set. The remaining part, part k, is left out when fitting the model and is referred to as the test set. As a statistical model has been fitted to the training set, one uses this model to make predictions on the test set. The prediction error is then calculated. This procedure is repeated for $k = 1, 2, ..., K$ and the total prediction error can be estimated. This procedure is illustrated in Figure 3.8

Let $\kappa : \{1, ..., n\} \to \{1, ..., K\}$ be function indicating the partition to which observation i is allocated by the randomization (Hastie et al., 2009, chap. 7). Further let $\hat{f}^{-k}(x)$ represent the fitted model with k-th part of the data left out. The cross-validation estimate of the prediction error is then given by:

$$CV(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}^{-\kappa(i)}(x_i)),$$

where L is a loss function and (x_i, y_i) is the i-th observation.

In this thesis we will use the $0-1$ loss function for qualitative responses

$$L(C, \hat{C}(X)) = I(C \neq \hat{C}(X)),$$

where C denotes the true category (or group), $\hat{C}$ denotes the predicted category and I denotes the indicator function, returning 1 (true) or 0 (false).

When doing cross-validation the choice of K must be given some thought, as there is a bias-variance tradeoff associated to the choice of K. A special case of K-fold cross-validation is $K = n$ and is known as leave-one-out cross-validation (LOOCV). In this case

1	2	3	...	k	...	$K-1$	K
Train	Train	Train	...	Test	...	Train	Train

Figure 3.8: A graphical representation of K-fold cross-validation. One partitions the data set into K subsets. One fits a model using $K-1$ of the subsets, and then tests the prediction error by making predictions for left-out subset k. This is repeated in turn for all of the k subsets. The total error is averaged over the K estimates.

a single observation is left out as the test set in each step of the modelling procedure. LOOCV is approximately unbiased for the prediction error, but has high variance as each of the training sets differ by only one observation (Hastie et al., 2009, chap. 7). LOOCV could however be good for small data sets, as each additional left-out observation gives a worse fitted model. A higher choice for K is computationally faster and cheaper and could be a good choice for big data sets. Typical choices for K are 5 and 10 as they lead to an intermediate level of bias and variance.

3.4.5 Proper Scoring Rules

Scoring rules provide summary measures in the evaluation of probabilistic forecasts, by assigning a numerical score based on the predictive distribution and the observation (Czado et al., 2009). A scoring rule $S(P, \omega)$ rewards an expert when his prediction is P and the realized outcome is ω (Chen, 2012). Let $\mathcal{P}$ be a convex class of probability measures. We write

$$S(P, Q) = \int S(P, \omega) dQ(\omega)$$

for the expected score under Q when the probabilistic forecast is P (Gneiting and Raftery, 2007). The scoring rule S is *proper* relative to $\mathcal{P}$ if

$$S(Q, Q) \geq S(P, Q) \quad \forall P, Q \in \mathcal{P} \tag{3.5}$$

and is *strictly proper* if (3.5) holds with equality if and only if $P = Q$. A proper scoring rule thus encourages honest predictions. If the scoring rule is strictly proper both calibration and sharpness are being addressed (Winkler, 1996). Calibration refers to the statistical consistency between the probabilistic forecasts and the observations, and is a joint property of the predictive distributions and the observations. Sharpness refers to the concentration of the predictive distributions, and is a property of the forecasts only (Czado et al., 2009).

Brier Score

The Brier score was introduced by Brier in 1950. It can be used to evaluate a forecaster by comparing a number of past forecast probabilities $\hat{\pi}_1, ..., \hat{\pi}_n$ to their verifying observations $y_1, ..., y_n$ (Siegert, 2017). It is given by the squared distance between the actual outcome $y_i \in \{0, 1\}$ and the forecast $\hat{\pi}_i \in [0, 1]$. The Brier score is given by

$$BS = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{\pi}_i)^2$$

where n is the number of forecasts. The decision space for the Brier score is the interval [0,1] and generally the lower the better (Gerds et al., 2008). The Brier score is a proper scoring rule.

Chapter 4

Ensemble Methods

What characterizes an ensemble method, is that a statistical model is fitted on many sets of outputs aggregated to produce one prediction. This is done by making a number of passes over the data, where linkages between the response and explanatory variables are obtained on each pass (Berk, 2016, chap. 4). The topic of interest is the collection of linkages made on each pass. Multiple learning algorithms are used to obtain better predictive performance than can be obtained by using one learning algorithm alone.

Examples of ensemble methods are bootstrap aggregation (bagging), boosting and random forests. In this thesis our focus is on bagging, Section 4.2 and random forests, Section 4.3. Bagging was proposed by Leo Breiman in 1994, and is a technique for reducing the variance of an estimated prediction function (Hastie et al., 2009, chap. 8), by taking the average of many noisy but unbiased models. The random forests algorithm was introduced by Breiman in 2001 and is a modification of bagging. As in bagging, an average of many models is taken. However, in random forests, each model is built on a random subset of the predictors. In that way, the correlation between the models is reduced, and the predictive performance is further improved. In the case of CART this means that the prediction is taken as an average of many small trees, each fit to a subset of the predictors.

4.1 The Bootstrap

As mentioned earlier, the motivation for bagging is that one can reduce the variance of a prediction by taking the average prediction from many predictive models. Because the most common case is that one has a single sample to work on, the bootstrap is a helpful tool. Bagging averages the prediction over a collection of models fit to bootstrap samples.

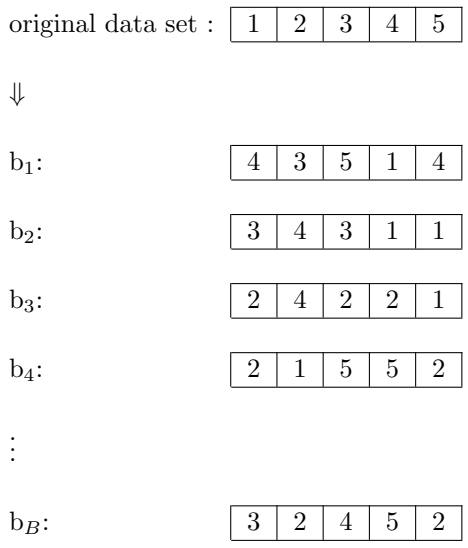


Figure 4.1: The bootstrap. The original sample is shown at the top. To create a bootstrap sample one draws from the original sample with replacement.

The bootstrap is a statistical tool introduced by Efron (1979). The most common use is to quantify the uncertainty associated with a given estimator or a statistical model. By drawing a large number of samples with replacement from a single realized data set, we simulate data sets which differ somewhat from one another. This is called re-sampling with replacement and is illustrated in Figure 4.1 The probability samples are denoted by $b_1, b_2, ..., b_B$, and are called the bootstrap data sets. B is the total number of resampled data set. Each data set contains n elements. When sampling with replacement, about 63% of the data points are in fact the same as in the original sample (Izenman, 2008, chap. 14). Consider a sample of size n, denoted by S. The probability of drawing an observation x_i from the sample is $\Pr(X = x_i) = \frac{1}{n}, i = 1, ..., n$. Hence the probability that observation x_i has not been drawn is $1 - \frac{1}{n}$.

When we apply the bootstrap algorithm, we select n observations from the sample *with replacement*. The probability for choosing a specific observation is always the same for all observations. This is the same as sampling from independent and identically distributed random variables

$$\Pr(x_i \in b_j) = 1 - \left(1 - \frac{1}{n}\right)^n = 1 - \frac{1}{e} = 0.632$$

where b_j denotes the j-th bootstrap sample.

4.2 Bagging

Assume we have n observations of a random variable X each with variance σ^2. These random variables are thus independently and identically distributed. We calculate the mean $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. The variance of the mean is

$$\mathrm{Var}(\bar{X}) = \mathrm{Var}\Big(\frac{1}{n}\sum_{i=1}^{n} X_i\Big) = \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}(X_i) = \frac{\sigma^2}{n}.$$

Thus by averaging one can reduce the variance.

When bagging we take several training sets from a population and build a separate prediction model using each training set. If we have a numerical outcome we obtain a final prediction by averaging all of the separate predictions. In the case of a classification problem, where the output is the class membership, we take a majority vote over the separate predictions. The multiple training sets are formed by making bootstrap replicates of the the original training set and to use these as the new training sets (Breiman, 1996). The idea is that one can obtain a model with a higher prediction accuracy since the resulting average prediction has a smaller variance than one prediction alone. The vital element is the instability of the prediction method. If perturbing the training set can cause significant changes in the predictor constructed, then bagging can improve accuracy (Breiman, 1996).

4.2.1 Classification and the Majority Vote

Suppose we have a training set, make B predictions on bootstrap samples of this training set, and make a class prediction for each $\mathbf{x}_i$

$$\{\hat{C}_b(\mathbf{x}_i\}_{b=1}^{B} = \{\hat{C}_1(\mathbf{x}_i), \hat{C}_2(\mathbf{x}_i), ..., \hat{C}_B(\mathbf{x}_i)\}.$$

Further, let $\hat{p}_k(\mathbf{x}_i)$ be the proportion of models that predicts $\mathbf{X}_i = \mathbf{x}_i$ to be a member of the kth class. The final classification is then obtained by the *majority vote*:

$$\hat{C}_{\mathrm{bag}}(\mathbf{x}_i) = \underset{k}{\mathrm{argmax}}\{\hat{p}_k(\mathbf{x}_i)\} = \text{majority vote } \{\hat{C}_b(\mathbf{x}_i)\}_{b=1}^{B}.$$

In the case of a tie, the normal conversion is that these are split at random. If we have two classes we can avoid a tie by selecting an odd number of bootstrap samples B.

4.2.2 Out-of-Bag Observations

One of the advantages of bagging is that cross-validation no longer is needed to provide the estimate of the test error. When bagging we fit models to bootstrapped subsets of the observations. On average, each bagged model makes use of around two-thirds of the observations. The last one-third of the observations are not used to make the fit and are called the out-of-bag observations (OOB). These OOB's can be used to estimate the prediction accuracy. We can predict the response for the i-th observation by averaging the predictions by the fitted models in which the i-th observations was an OOB. This leads to a single OOB prediction. By repeating this for all OOB observations, the overall OOB mean squared error or classification error can be computed. The resulting OOB error is a valid estimate of the test error for the bagged model, since the response for each observation is predicted using only the models that were not fit using that observation (James et al., 2013a, chap. 8). This is essentially the LOOCV error, if B is large.

4.2.3 The Bagging Algorithm Applied to Classification Trees

The procedure when bagging a classification tree is as follows: An ensemble of B large trees $\{T_b\}_1^B$ is fitted to bootstrap samples $\mathbf{Z}^{*b}$ of the original data set. We leave the trees unpruned to minimize the bias (Izenman, 2008, chap. 14). B class predictions $\hat{C}_b(\mathbf{x}_i)$ for $\mathbf{x}_i$ are made using each of the trees. Finally, a majority vote is taken among these B predictions to give the final prediction

$$\hat{C}^B_{\text{bag}}(\mathbf{x}_i) = \text{majority vote}\{\hat{C}_b(\mathbf{x}_i)\}_1^B.$$

The details of the bagging algorithm applied to classification trees are presented in Algorithm 3.

Suppose there are k_i ($\leq B$) trees for which $\mathbf{x}_i$ is a member of the corresponding OOB sample. The OOB misclassification rate is then

$$err_{OBB} = \frac{1}{k_i}\sum_{i=1}^{k_i} I[\hat{C}_b(\mathbf{x}_i) \neq y_i]. \tag{4.1}$$

The only tuning parameter when bagging a CART is the number of bootstrap samples B. It is important to have a sufficiently large B. With a low value for B the OOB error will have high variance. When B increases, the variance of the error will decrease, and will reach a stable minimum value. However, as the misclassification error stabilizes, additional trees will typically not give a further improvement.

Algorithm 3: Bagging classification trees

Input: Training data $\mathbf{Z} = (\mathbf{x}, y)$
Number of trees to be grown: B

for $b = 1$ *to* B : **do**
  Draw a bootstrap sample $\mathbf{Z}^{*b}$ from the training data;
  Grow a tree T_b to the bootstrapped data;
  Make class predictions for the OOB $\mathbf{x}$'s using T_b, $\hat{C}_b(\mathbf{x})$;
end

Output: Final prediction: $\hat{C}^B_{\text{bag}}(\mathbf{x}) =$ majority vote $\{\hat{C}_b(\mathbf{x})\}_1^B$

4.2.4 The Bagging Algorithm Applied to Logistic Regression

When bagging a logistic regression model one separately fits B logistic GLMs on bootstrapped data sets. We will base this theory on the method purposed by Song et al. (2013). The procedure of fitting GLMs includes a subset selection. Each of the logistic GLMs will be created using forward stepwise regression, where at each step the model with the lowest AIC score will be chosen as the best one. The result is an ensemble of B logistic GLMs, which may contain different predictors.

To make a prediction of the outcome for a new observation, there are two possibilities:

1. By the *majority vote:* Let each of the GLMs make a prediction $\hat{f}_i$ for $y_i = \{0, 1\}$, given the input values $\mathbf{x}_i$. This gives a total of B predictions. The predictions for new observations $\mathbf{x}$ are obtained by taking a majority vote over all of these B predictions:
$$\hat{f}^B_{\text{bag}}(\mathbf{x}) = \text{ majority vote } \{\hat{f}^b(\mathbf{x})\}_1^B.$$

2. By *averaging the probabilities:* (or the adjusted majority vote). Let $\hat{\pi}_i$ denote the estimated probability for $y_i = 1$, given the input values $\mathbf{x}_i$. This gives a total of B probabilities. The prediction for new observations $\mathbf{x}$ are obtained by taking an average over all of these B probabilities, and then classify according to a threshold value c
$$\bar{\hat{\pi}}^B_{\text{bag}}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} \hat{\pi}^b(\mathbf{x})$$
$$\hat{f}^B_{\text{bag}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \bar{\hat{\pi}}^B_{\text{bag}}(\mathbf{x}) \geq c \\ 0 & \text{if } \bar{\hat{\pi}}^B_{\text{bag}}(\mathbf{x}) < c \end{cases}.$$

As the bagging algorithm applied to logistic regression builds on bootstrap data sets, we have OOB observations, which can be used to estimate the prediction error using Equation 4.1.

Bagging applied to logistic regression is not expected to give an equally considerable increase in the predictive performance, as for classification trees. Logistic regression is often a stable method, while bagging gives best improvements to unstable methods (Breiman, 1996).

Algorithm 4: Bagging logistic GLM, based on the adjusted majority vote

Input: Training data $\mathbf{Z}$
Number of bootstrap samples: B

1 **for** $b = 1$ *to* B : **do**
2 Draw a bootstrap sample $\mathbf{Z}^{*b}$ from the training data;
3 Fit a logistic GLM to the bootstrapped data $\hat{f}^b$;
4 Make a prediction for the probability of success $\hat{\pi}^b(\mathbf{x})$ for the observations in the test set;
5 **end**

Output: Final prediction: Average the predicted probabilities:
$\bar{\hat{\pi}}^B_{\text{bag}}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B}\hat{\pi}^b(\mathbf{x})$. Classify $\hat{f}^B_{\text{bag}}$ according to a threshold value.

4.2.5 Benefits and Limitations

- Bagging works especially well for unstable procedures, procedures which have high variance and low bias (Breiman, 1996). Examples of unstable procedures are neural networks, CART and subset selection in linear regression. k-nearest neighbor classifier is an example of a stable procedure, in which bagging will not help.

- Trees are ideal candidates for bagging, since they can capture complex interaction structures in the data, and have relatively low bias (Hastie et al., 2009, chap. 8).

- One of the benefits of bagging is that pruning in CART is no longer needed.

- Bagging cannot reduce the bias. Random forests is a modification of bagging which in addition to reducing the variance can reduce the bias, and will be discussed in the next chapter.

4.3 Random Forests

The random forest algorithm is similar to the bagging algorithm. In fact, bagging is a special case of a random forest. We saw in Section 4.2 that one can reduce the variance by averaging many noisy but approximately unbiased models. Further, we saw that tree structures gain much by bagging as they have high variance and low bias, but are capable of capturing complex interaction structures in the data.

In Section 4.2 we saw that the variance of the average of n observations of independent, identically distributed random variables X, each with variance σ^2 is $\frac{\sigma^2}{n}$. Now, suppose we have n observations of a random variable X which are identically distributed, each with mean μ and variance σ^2, but not independent. That is, suppose the variables have a positive pairwise correlation ρ

$$\mathrm{Cov}(X_i, X_j) = \rho\sigma^2, \quad i \neq j.$$

These observations have a compound symmetry correlation structure. We calculate the variance of the average

$$\begin{aligned}
\mathrm{Var}(\bar{X}) &= \mathrm{Var}\Big(\frac{1}{n}\sum_{i=1}^{n} X_i\Big) \\
&= \sum_{i=1}^{n} \frac{1}{n^2}\mathrm{Var}(X_i) + 2\sum_{i=2}^{n}\sum_{j=1}^{i-1} \frac{1}{n}\frac{1}{n}\mathrm{Cov}(X_i, X_j) \\
&= \frac{1}{n}\sigma^2 + \frac{n-1}{n}\rho\sigma^2 \\
&= \frac{1}{n}\sigma^2 + \rho\sigma^2 - \frac{1}{n}\rho\sigma^2 \\
&= \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2.
\end{aligned}$$

The idea behind random forests is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much (Hastie et al., 2009, chap. 15). This is achieved by a growing trees on B bootstrapped data set, but with a random selection of the input variables at each split, see Algorithm 5. Before each split, only m of the p input variables are selected at random as candidates for splitting, where $m \leq p$. Typically $m \approx \sqrt{p}$. A value of $m = p$ amounts to bagging.

As in bagging, the predictions for new observations $\mathbf{x}$ are made by averaging the predictions

of the B unpruned trees

$$\hat{C}_{rf}^{B}(\mathbf{x}) = \text{ majority vote } \{\hat{C}_b(\mathbf{x})\}_1^B.$$

Tuning Parameters

There are only two tuning parameters in random forests: the number of m variables randomly chosen as a subset at node and the number of bootstrap samples B (Izenman, 2008, chap. 14).

The prediction error will typically have a lower variance by increasing the number of bootstrap samples B. When B is sufficiently large, the error will reach a minimum value (a flattening on the curve). Hence a B big enough is crucial, while increasing B will not help when the minimum prediction error is reached.

Using a small value of m in building a random forest will typically be helpful when we have a large number of correlated predictors (James et al., 2013a, chap. 8). The optimal number of variables m to be chosen as a subset can be found by cross validation.

Algorithm 5: Growing a random forest

Input: Training set $\mathbf{Z}$
Number of candidate variables to be tried at each iteration: m
Number of trees to be grown: B
Minimum node size $n_{\min}$

for $b = 1$ *to* B : **do**
  Draw a bootstrap sample $\mathbf{Z}^*$ from the training data;
  Grow a random forest tree T_b on the bootstrapped data by following these steps in each terminal node:
  repeat
    Select m variables at random from the p variables;
    Pick the best variable/split-point among the m;
    Split the node into two daughter nodes;
  until *the minimum node size* n_{min} *is reached*;
  Make class predictions for $\mathbf{x}$ using T_b, $\hat{C}_b(\mathbf{x})$;
end

Output: Final prediction: $\hat{C}_{\text{rf}}^{B}(\mathbf{x}) = \text{ majority vote } \{\hat{C}_b(\mathbf{x})\}_1^B$

4.4 Random GLM

A random generalized linear model (RGLM) is an ensemble predictor based on bootstrap aggregation (bagging) of generalized linear models, where a subset of predictors is chosen at random from the predictor space, for each bootstrap sample. Using these predictors, a generalized linear model is build using forward regression according to the AIC criterion (Song et al., 2013). We will in this thesis concentrate on logistic regression, as our outcome is binary.

The construction of a RGLM consists of many steps. As before, assume we have a data set consisting of n observations: $\mathbf{Z} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i = \{(x_{i1}, x_{i2}, ..., x_{ip})\}$. The following procedure takes place at each of the B iterations: A bootstrap sample is drawn from the original data set. A number of m variables is chosen at random from the p predictors. These are rank-ordered according to their individual association with the outcome variable y, by fitting a univariate GLM (a GLM with one predictor only). The l top ranking variables are identified by using the Wald test or a likelihood ratio test and make up the candidate predictors. These candidate predictors are used to construct a logistic GLM using forward stepwise regression based on the AIC score. We now have B different logistic GLMs. We use these models to make B separate predictions. The final prediction is a result of an *adjusted majority vote* (aMV). The adjusted majority vote has been defined in Section 4.2.4, and is simply the average of the predicted probabilities for each of the B models. This procedure is summarized in Algorithm 6.

User-Defined Parameters

A RGLM has several parameters which need to be chosen by the user. The first parameter is the number of bags B. This is a tuning parameter, and one needs to make sure that B is large enough, so that the lowest possible misclassification error is reached. To choose the value of B one can plot the misclassification error against the number of bootstrap samples. The value of B should be chosen such that the misclassification error has reached a stable value (a flattening on the curve).

Another tuning parameter is the number of random predictors to be chosen for each GLM, denoted by m. This tuning parameter is similar to the m in random forests. If $m = p$ all predictors are chosen in each model. Song et al. (2013) present recommendations for the choice of m, based on the dimensions of the data set and number of interactions between the predictors. We have not considered modelling interactions in this thesis.

Algorithm 6: Random GLM for logistic models

Input: Training set $\mathbf{Z}$
Number of bootstrapped samples to be made B
Number of candidate predictors m
Top number of candidate predictors l

1 **for** $b = 1$ *to* B : **do**
2 Draw a bootstrap sample $\mathbf{Z}^*$ from the training data;
3 Select m variables at random from the p variables;
4 Fit a univariate GLM model to each of the m variables to arrive at a association measure;
5 Pick out the l top ranking (most significant) variables as candidate predictors;
6 Fit a GLM model by forward selection using the l chosen candidate predictors using the AIC criterion;
7 Make a prediction for the probability of success $\hat{\pi}^b(\mathbf{x})$, for the observations in the test set;
8 **end**

Output: Final prediction: Average the predicted probabilities:
$\bar{\hat{\pi}}^B_{\text{RGLM}}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B}\hat{\pi}^b$. Classify $\hat{f}^B_{\text{RGLM}}$ according to a threshold value.

The last tuning parameter is the maximum number of predictors l. The motivation for this parameter is to reduce the computational time of the algorithm in data set with a large number of predictors, of which not all of equal importance. The default value is $l = 50$, in the software. As our number of predictors, in Chapter 5 and 6, is lower than 50, we can skip this step.

Figure 4.2 shows an overview over the random GLM algorithm. It is based on a similar overview in the article by Song et al. (2013), but is adjusted and therefore more relevant for this thesis.

OOB Observations

The bootstrap aggregation step makes it possible to use OOB observations to estimate the predictive accuracy. By recording which observations have been left out in which bootstrap sample, an overall OOB error estimate of the predictive accuracy can be estimated, as in the bagging and random forest algorithms. It is recommended to use the OOB estimate of prediction accuracy to advise the choice of the parameter values (Song et al., 2013).

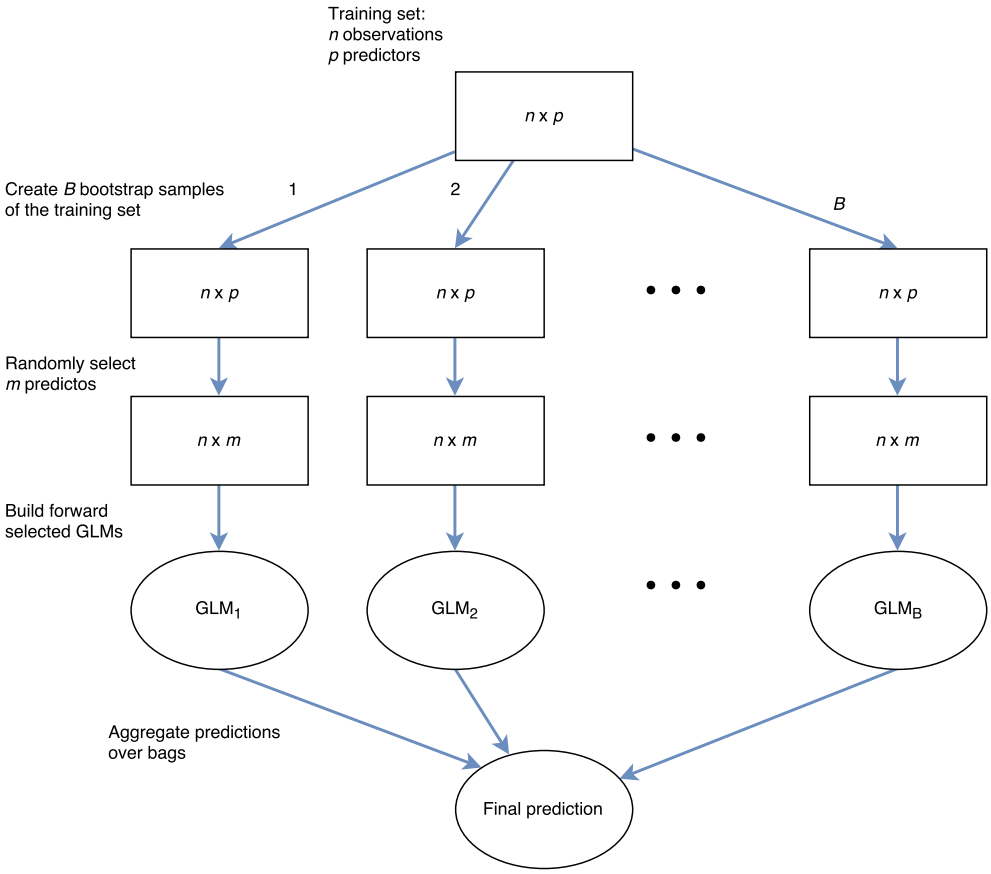


Figure 4.2: Overview of the RGLM construction. This is an simplified overview, based on the overview in the article by Song et al. (2013). Our number of predictors is such that we can skip the step of selecting the l top-ranking predictors in the algorithm. The rectangles represent data matrices at each step.

4.5 Variable Importance

When training a statistical model the case is often that several input variables are used. The input variables are however seldom of equal importance. Some of the input predictors turn out to be completely irrelevant. If the objective is to classify new observations, it is useful to know which of variables control the classification process (Izenman, 2008, chap. 14). This also helps us in understanding what influences the outcome.

4.5.1 Variable Importance for Tree Ensembles

The relative importance of predictor variables can be illustrated by a *variance importance plot*. In such a plot, the variables are sorted according to their importance, such that the top variables have a higher importance than the bottom variables. We will here present two types of variable importance plots which can be used for tree ensembles.

Variable Importance Based on Randomization

The variable importance based on randomization measures the prediction strength of each variable. Computations are carried out for one bootstrap sample at a time. Let T_b be the tree grown to the $\mathbf{Z}^b$th bootstrapped data. Let $\mathbf{Z}^{-b}$ be the OOB observations corresponding to this tree. Pass these OOB observations down the tree T_b, note the classifications and compute the OOB error err_{oob}. Next, the OOB observations for the jth variable X_j are randomly permuted. Pass these permuted OOB samples down tree T_b and compute the new OOB error err^*_{oob}. If X_j is important, permuting its observed values will reduce our ability to classify successfully each of the OOB observations (Izenman, 2008, chap. 14). The decrease in accuracy as a result of this permuting is averaged over all trees, and is used as a measure of the importance of variable j in the tree ensemble (Hastie et al., 2009, chap. 15). The plot to the left in Figure 4.3 shows a variable importance plot of this type for the `iris` data set.

Variable Importance Based on the Mean Decrease in Gini Index

From Section 3.3.1 we know that the Gini index of a parent node always is larger than the Gini index of each of its daughter nodes. When bagging a classification tree or growing a random forest one creates an ensemble of trees. Gini importance index is obtained by averaging the decrease in node impurities over all trees (Izenman, 2008, chap. 14) for each variable separately. The plot to the right in Figure 4.3 shows a variable importance plot of this type for the `Carseats` data set in the `ISLR` package.

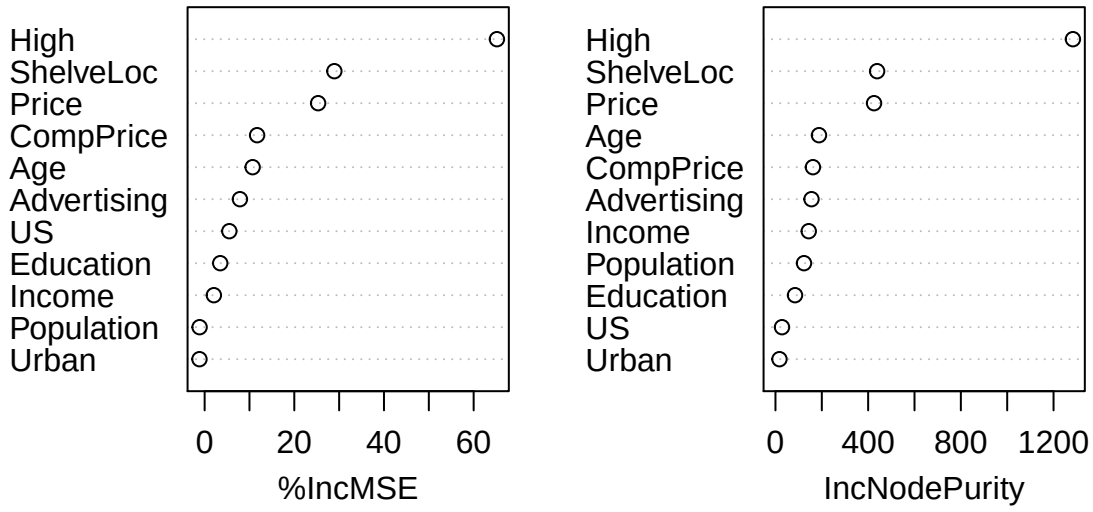


Figure 4.3: Variance importance plots. Right : Variance importance based on randomization (mean decrease in accuracy). Left: Mean decrease in node impurity (Gini index).

4.5.2 Variable Importance for GLMs and Random GLMs

Variable Importance for a Single GLM

To compare the variable importance of the predictors in a logistic model, one can use the absolute values of the z-statistics for each model parameter. The z-statistic emerges from the Wald test for regression coefficients. A higher $|z|$-value implies a more significant predictor.

Variable Importance for RGLM

There is a number of ways one can evaluate the importance of variables in a RGLM. These are shortly explained in the article by Song et al. (2013). In this thesis we will use variable importance plots based on the number of times a variable has been selected by forward

selection in building the RGLM.

Chapter 5

Simulation Study

In order to gain a better understanding of how the statistical methods we want to use can be applied to our medical data set, we generate a model for an artificial data set. The aim for the simulations is to reproduce a similar, but not identical, data set as the original one. We want some features of the original data set to be captured by the simulations, but we are not interested in reproducing all features exactly.

We start this chapter by an exploratory analysis of our original HUNT data set in order to gain an understanding of it. This will enable us to choose how to simulate each predictor. Thereafter in Section 5.3 we present our model for simulating an artificial data set. This model will be used to simulate values for the predictors. In Section 5.4 we describe how we will generate response data, using our artificial data set. We divide our simulated data in two: into a training and a test set. We proceed by fitting 6 different statistical models to our training set. By using the test set we can evaluate the predictive performance of the models. Furthermore, we repeat the procedure of simulating and model fitting a large number of times. The results of our simulation study are presented in Section 5.7. Our focus will be on comparing the predictive performance of the various statistical models.

5.1 Predictors in the HUNT Data Set

The set of predictors which we want to include in our model can be divided into three parts. The first part consists of predictors which are included in the Framingham risk score, discussed in Section 2.1. These are the following:

- $\mathbf{x}_{age}$: Participation age [yr] (PartAg.NT2BLQ1)

- $\mathbf{x}_{sechol}$: Serum cholesterol [mmol/L] (SeChol.NT2BLM)
- $\mathbf{x}_{hdl}$: Serum High-density lipoprotein cholesterol [mmol/L] (SeHDLChol.NT2BLM)
- $\mathbf{x}_{bp}$: Systolic blood pressure [mmHg](BPSystMn23.NT2BLM)
- $\mathbf{x}_{smoke}$: Smoking status

Each $\mathbf{x}_k$ represents observations of the predictor coded by the subscript. The unit in which the predictor has been measured is in squared brackets. The corresponding coding in the HUNT data base is in the round brackets. According to Velle-Forbord (2017), additional predictors may include:

- $\mathbf{x}_w$: Weight [kg] (Wei.N21BLM)
- $\mathbf{x}_h$: Height [cm] (Hei.NT2BLM)
- $\mathbf{x}_{wc}$: Waist circumference [cm] (WaistCirc.NT2BLM)
- $\mathbf{x}_{hc}$: Hip circumference [cm] (HipCirc.NT2BLM)
- $\mathbf{x}_{whr}$: Waist-to-hip ratio (WHR)
- $\mathbf{x}_{bmi}$: Body mass index [kg/m^2] (BMI.NT2BLM)
- $\mathbf{x}_{setrig}$: Serum triglycerides [mmol/L] (SeTrig.NT2BLM)
- $\mathbf{x}_{secrea}$: Serum creatinine [μmol/L] (SeCrea.NT2BLM)
- $\mathbf{x}_{seglu}$: Serum glucose non fasting [mmol/L] (SeGluNonFast.NT2BLM)

Finally, we want our models to contain miRNAs to investigate if they can have predictive power to predict myocardial infarction. The miRNAs available are the following:

- $\mathbf{x}_{hsa\_let\_7g\_5p}$: let-7g-5p (hsa_let_7g_5p)
- $\mathbf{x}_{hsa\_miR\_106a\_5p}$: miR-106a-5p (hsa_miR_106a_5p)
- $\mathbf{x}_{hsa\_miR\_144\_3p}$: miR-144-3p (hsa_miR_144_3p)
- $\mathbf{x}_{hsa\_miR\_151a\_5p}$: miR-151a-5p (hsa_miR_151a_5p)
- $\mathbf{x}_{hsa\_miR\_191\_5p}$: miR-191-5p (hsa_miR_191_5p)

- $\mathbf{x}_{hsa\_miR\_21\_5p}$: miR-21-5p (hsa_miR_21_5p)
- $\mathbf{x}_{hsa\_miR\_26a\_5p}$: miR-26a-5p (hsa_miR_26a_5p)
- $\mathbf{x}_{hsa\_miR\_29c\_3p}$: miR-29c-3p (hsa_miR_29c_3p)
- $\mathbf{x}_{hsa\_miR\_424\_5p}$: miR-424-5p (hsa_miR_424_5p)
- $\mathbf{x}_{hsa\_miR\_425\_5p}$: miR-425-5p (hsa_miR_425_5p)
- $\mathbf{x}_{hsa\_miR\_451a}$: miR-451a (hsa_miR_451a)

We start the exploratory analysis of our data set by creating a heatmap of the predictors, which can be found in Figure 5.1 [1]. Each column represents one individual, and each row represents one predictor. The row named *MI* is the corresponding outcome of the response variable for each individual. The values have been centered and scaled in the row direction, and have thereafter been translated into colors, where the color-coding is found in the top-left corner of the figure. This because some of the predictors are much higher in magnitude than others, which would mask the relations between the variables. The dendograms show the arrangement of clustering between the predictors and between the subjects. We see that it is difficult to observe obvious groupings of the predictors. The levels of serum creatinine is clearly related to the waist and hip circumference of a subject, and with the blood pressure. Also most of the miRNAs have been clearly grouped together.

The top plot in Figure 5.2 shows box plots of centered predictors of the original data set. These predictors will be used in our statistical models. The bottom plot in Figure 5.2 shows box plots of the miRNA's in the original data set. We see that most of the predictors have a symmetric distribution, with an even spread around the mean. We therefore conclude that it is justified to simulate these predictors from the normal distribution.

Figure 5.3 shows the correlation matrix[2] of some of the predictors. These are the predictors we want to use in our statistical models. We see that there are many predictors which are uncorrelated. However, body measurements are highly correlated. We want to account for this correlation structure in our simulated data set, hence we will use this correlation matrix when making simulations.

[1]The heatmap has been created using the `heatmap.2` function from the `gplots` package, created by Warnes et al. (2016). The function used to compute the distance between the rows and columns has been Euclidean, $L_2 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$.

[2]These plots have been created using the `corrplot` package (Wei and Simko, 2016).

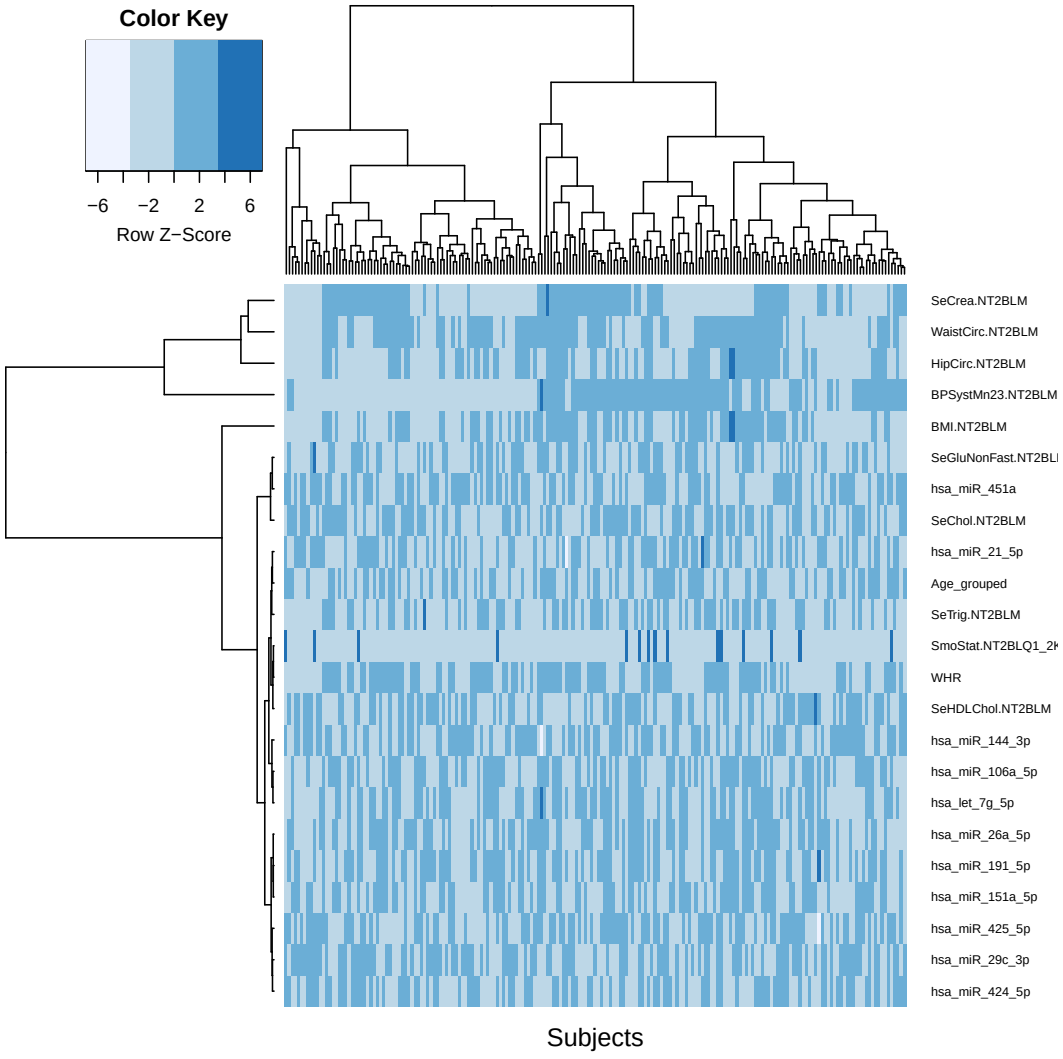


Figure 5.1: A heatmap comparing the predictors in our data set. These predictors have been centered and scaled in the row direction. Each column corresponds to one subject and each row corresponds to one predictor. Each colored rectangle represents the numerical value of predictor l for for subject i, where the color key shows the interval of each color. The dendograms show the hierarchical clustering of subjects and predictors using Euclidean distance function.

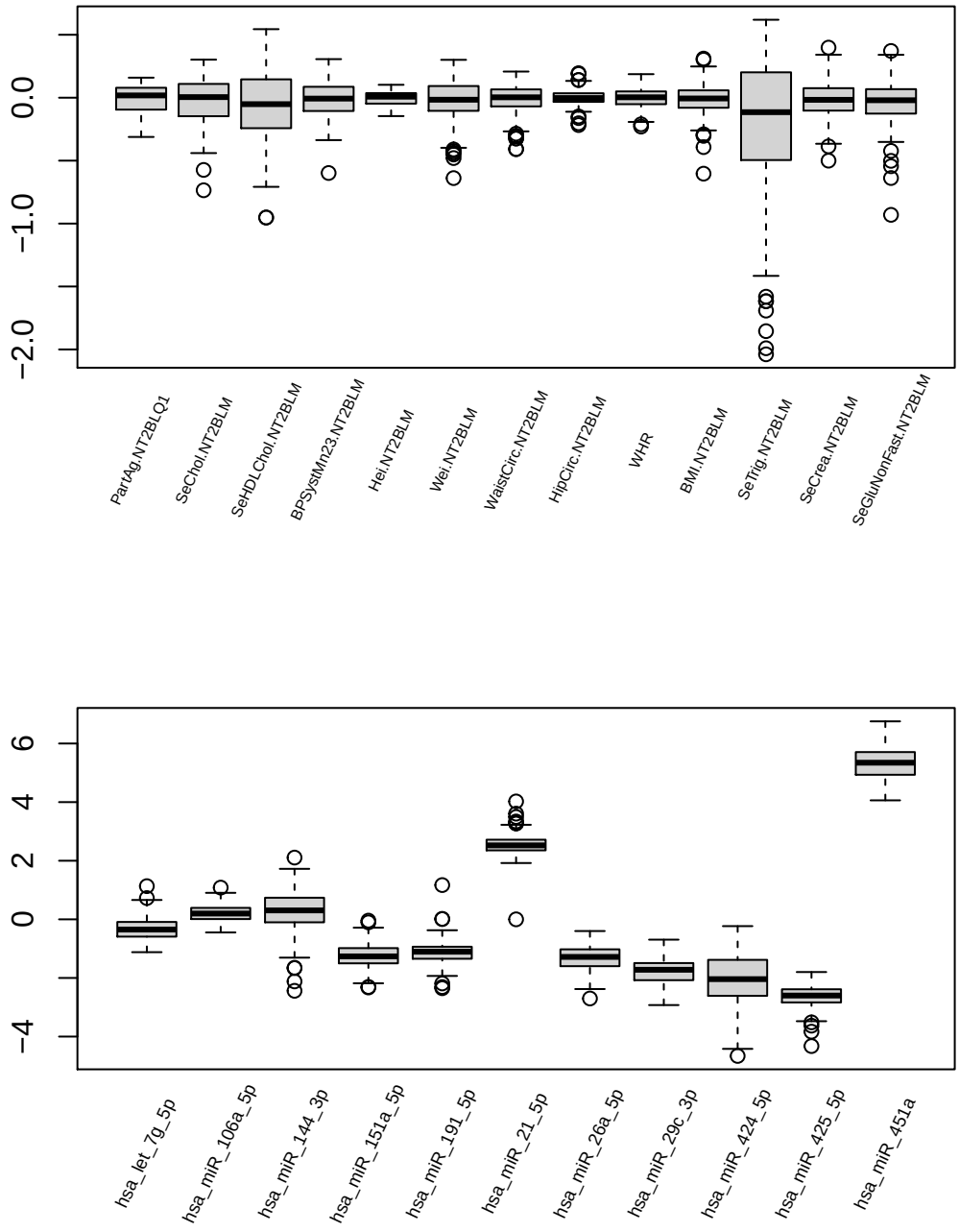


Figure 5.2: Top: Box plots of the centered predictors Bottom: Box plots of the miRNA's.

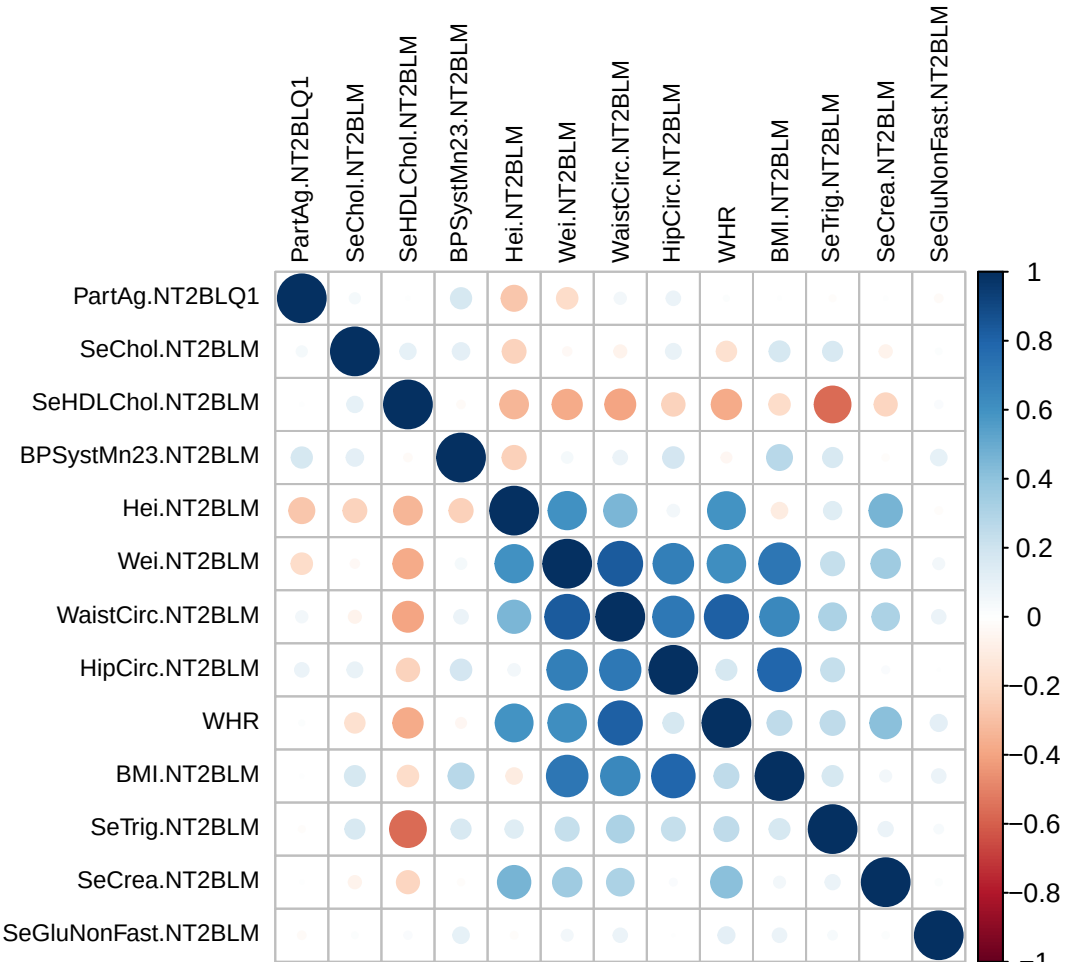


Figure 5.3: The estimated correlation matrix of some of the predictors from the original data set. The diameter of the circle shows the absolute value of the correlation (i.e. a larger circle indicates a higher correlation), while the color indicates if the correlation is positive or negative (blue=positive, red=negative).

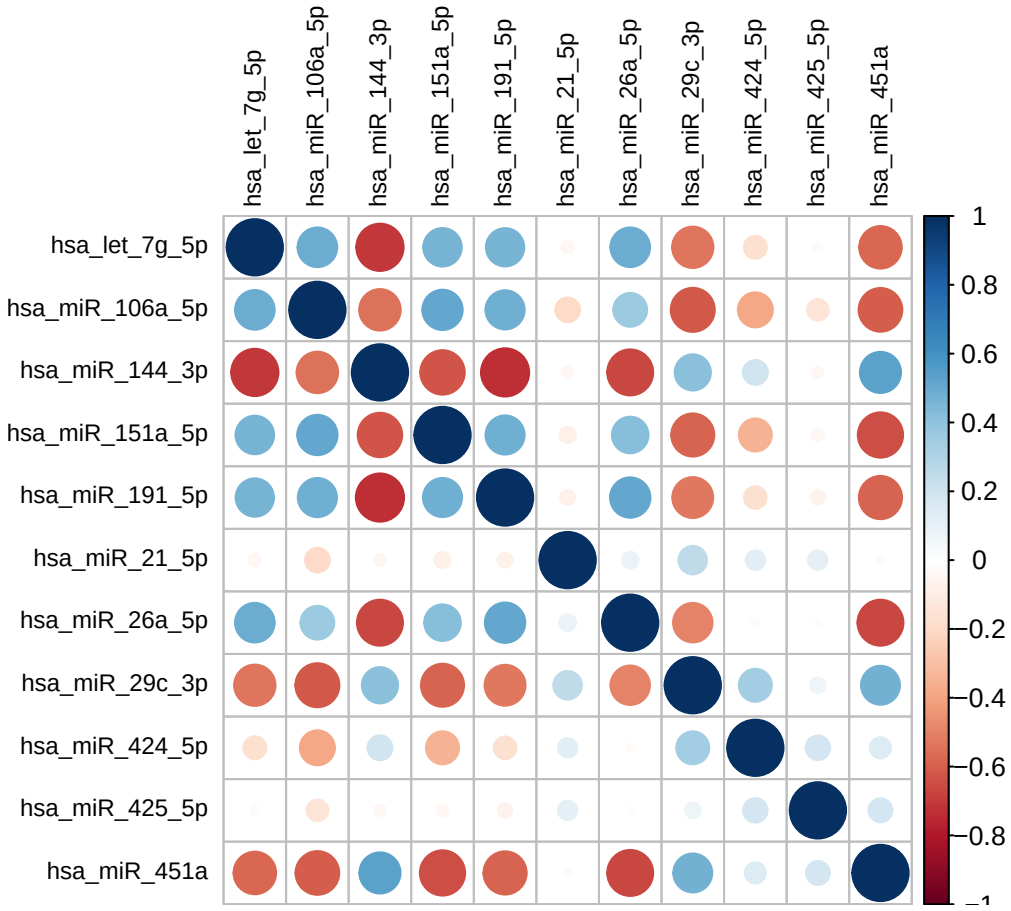


Figure 5.4: The estimated correlation matrix of the miRNAs of the original data set.

We further see that the serum triglyceride and the HDL cholesterol have a high negative correlation. This relation will also be included in the simulations. Figure 5.4 shows the covariance matrix of the miRNA's we might want to use in our models. Most of these are clearly highly correlated. This needs to be accounted for when doing simulations.

5.2 Generating Artificial Predictor Data

Let $x_{i,l}, i = 1, ..., 197$ denote the i-th observation of the l-th predictor of the original data set. Further, let $z_{j,l}, j = 1, ..., 200$ denote the j-th simulated predictor l.

Age has been simulated as a factor, with levels corresponding to intervals of ten years: Age = {(49, 59], (59, 69], (69, 79], (79,89]}, ranging over the same values as the original data set. The simulation of age is done by a random draw (without replacement) of one age-interval. This makes each level equally likely. This is slightly different than the ages in the original data set, in which the ages follow a skew distribution, with a peak around the age 71.

Body measurements have been simulated from a multivariate normal distribution, because of the high correlations between the predictors. The mean vector consists of the sample means of the original data set. The covariance matrix is equal to the sample covariance matrix of the original data set. The following abbreviations have been used: wc = waist circumference, hc = hip circumference

$$\begin{bmatrix} z_{j,wc} \\ z_{j,hc} \\ z_{j,height} \\ z_{j,weight} \end{bmatrix} \sim \mathcal{N}_4 \left(\begin{bmatrix} \bar{x}_{wc} \\ \bar{x}_{hc} \\ \bar{x}_{height} \\ \bar{x}_{weight} \end{bmatrix}, \hat{\mathbf{\Sigma}}_{wc,hc,height,weight} \right).$$

Furthermore, as we want to have the body mass index (BMI) and the waist-to-hip ratio as predictors, these have been calculated from the simulated body measurements

$$z_{j,bmi} = \frac{z_{j,weight}}{z_{j,height}^2}$$
$$z_{j,whr} = \frac{z_{j,wc}}{z_{j,hc}}.$$

Smoking status has been simulated from the binomial distribution, where the number 1 indicates a smoker, and the number 0 indicates a non-smoker. The probability for an

individual j to be a smoker has been set to 0.4 in order to make the simulated values more similar to the original data set

$$z_{j,smoke} = Binom(n = 1, \pi_j = 0.4).$$

The predictors serum cholesterol, blood pressure, serum creatinine and serum glucose did not show high correlations with any of the other predictors. These predictors have thus been simulated from univariate normal distributions. The mean has for each predictor been set equal to the corresponding sample mean of the predictor in the original data set, and variance equal to the sample variance of the predictor in the original data set

$$\begin{aligned} z_{j,sechol} &= \mathcal{N}(\bar{x}_{sechol}, \hat{\sigma}^2_{sechol}) \\ z_{j,bp} &= \mathcal{N}(\bar{x}_{bp}, \hat{\sigma}^2_{bp}) \\ z_{j,secrea} &= \mathcal{N}(\bar{x}_{secrea}, \hat{\sigma}^2_{secrea}) \\ z_{j,seglu} &= \mathcal{N}(\bar{x}_{seglu}, \hat{\sigma}^2_{seglu}). \end{aligned}$$

The predictors serum triglyceride and HDL cholesterol are highly negatively correlated. These predictors have thus been simulated from the bivariate normal distribution, where the mean vector consists of the sample means from the original data set, and the covariance matrix consists of the corresponding sample covariance matrix

$$\begin{bmatrix} z_{j,setrig} \\ z_{j,hdl} \end{bmatrix} \sim \mathcal{N}_2 \left(\begin{bmatrix} \bar{x}_{setrig} \\ \bar{x}_{hdl} \end{bmatrix}, \begin{bmatrix} \hat{\sigma}^2_{setrig} & \hat{\rho}\hat{\sigma}_{setrig}\hat{\sigma}_{hdl} \\ \hat{\rho}\hat{\sigma}_{hdl}\hat{\sigma}_{setrig} & \hat{\sigma}^2_{hdl} \end{bmatrix} \right).$$

Finally, the miRNA's have been simulated from a multivariate normal distribution. We saw in Section 5.1 that most of the miRNA's are highly correlated. The vector of mean values will consist of the original sample means, while the covariance matrix will consist of the corresponding original sample covariances

$$\mathbf{z}_{j,mRNA} \sim \mathcal{N}_{11}(\bar{\mathbf{x}}_{mRNA}, \hat{\mathbf{\Sigma}}_{mRNA}).$$

5.3 Statistical Model for MI

The main motivation for creating a statistical model for MI is to create a response on which we can evaluate and compare methods. We can treat the simulated data set as the truth and can test whether or not our statistical methods are able to reveal the truth, that is, capture the underlying structures in the data.

As our statistical model for myocardial infarction we have chosen a logistic model

$$Y \sim Binom\Big(n = 1, \pi_j = \frac{\exp(\eta_j)}{1 + \exp(\eta_j)}\Big)$$

where

$$\eta_j = \mathbf{z}_j'\boldsymbol{\beta}, \quad \mathbf{z}_j = \{1, z_{j1}, z_{j2}, ..., z_{jp}\}, \quad \boldsymbol{\beta} = \{\beta_0, \beta_1, \beta_2, ..., \beta_p\}.$$

Because of the large number of predictors, of which not all are significant, we have chosen to select eleven predictors. These predictors will have β's different from zero in our model. All of the other predictors will have a β exactly equal to zero. The choice of these eleven predictors has been done through a logistic regression and bagging:

1. A logistic regression model was fitted to the original data set using all predictors. The predictors were ranked after how significant they were in the logistic model according to the Wald test.

2. The bagging algorithm was run on the original data set using all predictors. A variance importance plot was used to see a ranking of the prediction strength of each variable.

3. Based on steps 1. and 2., top eleven predictors were chosen intuitively.

By fitting a logistic model using these eleven predictors we thereafter obtained the estimates of the regression coefficients. The resulting β's are summarized in Table 5.1, and will serve as the true values.

5.4 Generating Response Data

Our original medical data set is a case-control data set with 97 cases and 100 controls. We want our artificial data set to inherit this property. We therefore create 200 artificial individuals of which a 100 have experienced an myocardial infarction (cases) and a 100 have not (controls)

$$Y_i = \begin{cases} 1 = \text{Case} & (n_1 = 100) \\ 0 = \text{Control} & (n_2 = 100). \end{cases}$$

Table 5.1: The true β values used in our statistical model for MI.

β_0	-8.1705
β_{bmi}	0.1495
β_{sechol}	0.2933
β_{seglu}	-0.2636
β_{bp}	0.0279
$\beta_{age,1}$	-0.1057
$\beta_{age,2}$	0.0132
$\beta_{age,3}$	-0.2035
$\beta_{hsa\_let\_7g\_5p}$	0.4887
$\beta_{hsa\_miR\_144\_3p}$	-1.4302
$\beta_{hsa\_miR\_26a\_5p}$	1.2514
$\beta_{hsa\_miR\_21\_5p}$	0.6019
$\beta_{hsa\_miR\_424\_5p}$	0.5198
$\beta_{hsa\_miR\_191\_5}$	-1.2141

This will make up our *training set* and will be used to fit the best model for each model type.

The response data has will be simulated using the logistic model of Section 5.3. Using the β-values from Table 5.1 and the artificial data set for the predictors, an individual π_j will be calculated for each individual j, where $j = 1, 2, ..., 200$. This π_j is the probability of $Y_j = 1$. Thereafter calls will been made to a function which returns random generations from the binomial distribution using this probability. Furthermore the process of simulating the data for predictors and a corresponding response value will be repeated until 100 cases are and 100 controls are created. Algorithm 7 summarizes the simulation procedure, including simulations of the predictors.

We will in addition simulate a *test set* of equal size using the same statistical model in order to evaluate the predictive performance of the models.

Algorithm 7: Simulating procedure

Input: $\boldsymbol{\beta}_{p\times 1}$

$j = n_1 = n_2 = 0$;
repeat
  for l = *1 to* p **do**
    Simulate predictor l for individual j: $z_{j,l}$;
    Save to $\mathbf{Z}[j, l]$;
  end
  $\mathbf{z_j} = \mathbf{Z}[j,]$;
  $\eta_j = \mathbf{z_j}'\boldsymbol{\beta}$;
  $\pi_j = 1/(1 + e^{-\eta_j})$;
  Simulate response value for individual j: $y_j = Binom(1, \pi_j)$;
  Save to $\mathbf{Y}[j]$;
  if $y_j = 1$ **then**
    $n_1 = n_1 + 1$;
  else
    $n_2 = n_2 + 1$;
  end
  $j = j + 1$;
until $n_1 \geq 100$ *or* $n_2 \geq 100$;
Sort data set according to the response values;
Remove excessive cases and/or controls such that $n_1 = 100$ and $n_2 = 100$;

Output: Data set $\mathcal{D} = [\mathbf{Y}_{n\times 1}, \mathbf{Z}_{n\times p}], n = n_1 + n_2 = 200$.

5.5 Choice of Tuning Parameters and R Packages

Our motivation for the simulation study is to compare the prediction accuracy of different statistical models to a similar data set as our medical data set. We are going to fit the following models to each of our simulated data sets: *pruned tree, logistic GLM, bagged tree, bagged logistic GLM, random forest* and *random GLM*. This section aims at argumenting for the choice of tuning parameters in the various models, in relation to the software used. We have used the statistical language `R`, created by R Core Team (2013), throughout this thesis and will in this Section introduce relevant `R` packages.

Model 1: Pruned Tree

A classification tree usually has low bias and high variance. The size of the tree governs the trees complexity and therefore has the role of a tuning parameter. A large tree will tend to overfit the data, and will have a low predictive power. A pruned tree will have higher bias but lower variance.

The `tree` Package

The package `tree`, created by Ripley (2016), will be used to fit and prune classification trees. We use the function `tree`, from the `tree` package, to grow a full classification tree. Thereafter, we use the function `cv.tree` to find the optimal tree size. The cost-complexity tuning parameter α is denoted by `k` in this function and the tree size is called `size`. We choose the tree size based on the minimal total deviance in the cost-complexity pruning sequence. Finally, we use the function `predict.tree` to obtain predicted probabilities for the test set.

Model 2: Logistic GLM

A logistic GLM which contains all of the predictors will usually have a low bias and high variance, and might overfit the data. By carrying out a subset selection the less relevant predictors can be removed from the model. The result is a more interpretable model with a better predictive performance.

The `glm` and `stepAIC` Functions

Our logistic GLMs will be fit using the `glm` function of the `stats` package, which is a part of `R`. To specify a logistic regression model we set `family = "binomial"`. We use the default link function, which is the logit function. Further, we use the `stepAIC` function,

from the `MASS` package (created by Venables and Ripley (2002)), to perform a subset selection. Since we are interested in a model obtained by forward stepwise regression, we set `direction="forward"` in the `stepAIC` function. We specify that the function should look through models with the scope ranging from the null model to the full model. The model with the lowest AIC value is chosen as the best model. The predicted probabilities for the test set were obtained by using the function `predict.glm` with `type = "response"`.

Model 3: Bagged Tree

The only tuning parameter in a bagged tree is the number of trees in the tree ensemble, B. However, the number of trees is not a critical parameter and using a very large value for B will not lead to overfitting. Choosing a value for B which is too small, will not give the best results, as the error will not have settled down. One therefore needs to select the value of B so that the error has stabilized. Selecting a value for B beyond that will not increase the predictive performance, and should be avoided due to computational time. In order to decide on a value for B we have simulated one training set, to which we have fitted a bagged tree, and one test set on which we have estimated the prediction error. Figure 5.5 shows the OOB error rate and the test error rate for the bagged tree as a function of the ensemble size B, from $B = 1$ (a single tree) to $B = 1000$. Based on this Figure, we assume an ensemble size of $B = 500$ should be sufficient.

The `randomForest` Package

To create a bagged tree we use the `randomForest` function from the `randomForest` package created by Liaw and Wiener (2002). To use this function one needs to specify the `mtry` variable. To create a bagged tree one needs to set this variable equal to the total number of predictors. This means that all predictors will be considered in each of the split in each tree, which is not the case for a random forest. To make predictions for the test set, the function `predict.randomForest` will be used with `type = "prob"`.

Model 4: Bagged Logistic GLM

In order to create a bagged logistic GLM, we will fit logistic GLMs to bootstrap samples of the simulated training data set. These logistic models will be created using forward stepwise regression, and the best models will be chosen based on the AIC score. As is the case of a bagged tree, the ensemble size should be big enough for the error to stabilize.

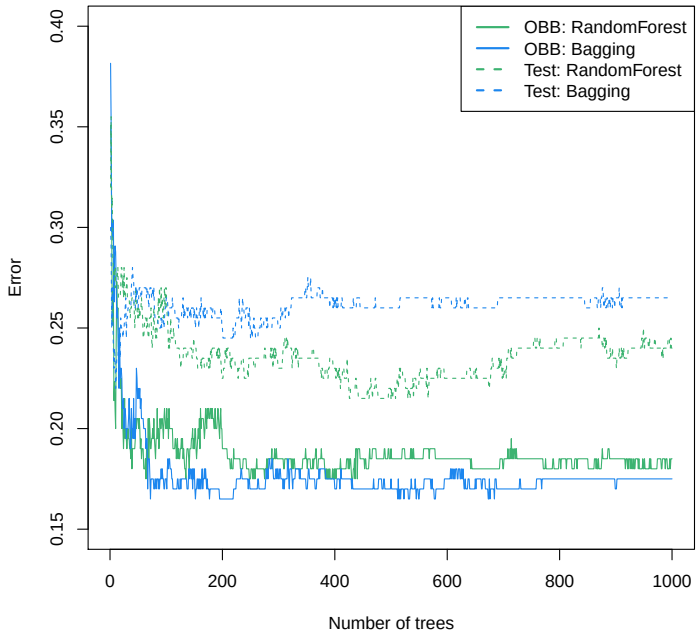


Figure 5.5: The training and test errors of a random forest as a function of `ntree`: the number of trees in the ensemble.

The `randomGLM` Package

The `randomGLM` function from the `randomGLM` package, created by (Song and Langfelder, 2013), will be used to fit a bagged logistic GLM. The parameter `nFeaturesInBag` must be set equal to the total number of predictors. The variable `nBags` relates to the ensemble size. We will restrict this variable to 100, as this is the default value and should be sufficient. By calling the function `predict.randomGLM` we get the predicted probabilities for our test set.

Model 5: Random Forest

A random forest is closely related to a bagged tree. However it contains an additional tuning parameter. In order to create a random forest only a random sample of m predictors is considered when making each split growing the tree. This parameter has to be set when

growing a random forest. Because the `randomForest` function which we will use to grow a random forest, is computationally efficient, we select a separate value for m for each random forest. The OOB error will be used as a criterion for selecting the value of m. Figure 5.5 shows the OOB and test errors as a function of the number of trees in the random forest. Based on this Figure we conclude that an ensemble size of $B = 500$ trees is sufficient.

`randomForest`

We will use the `randomForest` function to grow random forests. The variable `mtry` corresponds to m, which is the number of variables randomly sampled as candidates at each split. The value of `mtry` corresponding to the lowest OOB error will be chosen for each random forest separately. We will use `ntree` = 500 for each random forest. The function `predict.randomForest`, with `type = "prob"`, will be used to predict probabilities for the test set.

Model 6: Random GLM

A random GLM will be fit in a similar fashion as the bagged logistic regression model. The important difference is that as the model is built, using forward stepwise regression, only a random subset of m predictors will be chosen as candidates.

`randomGLM`

We will here also set `nBags` = 100. The tuning parameter m is called `nFeaturesInBag` in the `randomGLM` package. Song et al. (2013) recommend to use set this parameter based on the formula: $N \cdot (1.0276 - 0.00276 \cdot N)$, where N is the total number of predictors. For our simulated datasets, this is equal to $23 \cdot (1.0276 - 0.00276 \cdot 23) \approx 22$. The results would be almost identical as for the bagged logistic regression model. We for this reason choose not to follow this recommendation. Because the `randomGLM` function uses some computational time when proceeding with the stepwise regression, we will not go through each possible value, as for the case of `mtry` in the `randomForest`. Figure 5.6 shows the Brier score for different values of `nFeaturesInBag`. This Figure is created by simulating 100 train and test sets. A separate random GLM has been fitted to each of the training set and the Brier score has been calculated for using these models for the 100 test sets. The grey lines correspond to one train and test set pair. The black line shows the mean Brier score for the different choices of `nFeaturesInBag`. Based on this figure, we have chosen to let each RGLM test two values for the variable `nFeatureInBag`: 15 and 20. The model returning the lowest OOB error will be chosen as the final model. By calling the function `predict.randomGLM`

we get the predicted probabilities for the test set.

5.6 Procedure for Simulation and Model Fitting

The procedure of simulation and model fitting is illustrated in Figure 5.7 as well as described in this section. Because there is randomness associated with the simulated artificial data sets as well as in the fitted models, we have simulated 1000 train and test sets to evaluate the predictive performance of our statistical models. This has been done by following the following steps, 1000 times.

1. Simulate predictor data $\mathbf{z}_j = (z_{j1}, z_{j2}, z_{j,3}, ..., z_{jp})$ for individual j.

2. Calculate the probability of a case for individual j using the simulated predictor values and the coefficients in Table 5.1:

$$\pi_j = \frac{e^{\boldsymbol{\beta}^T \mathbf{z}_j}}{1 + e^{\boldsymbol{\beta}^T \mathbf{z}_j}} = \frac{1}{1 + e^{-\boldsymbol{\beta}^T \mathbf{z}_j}}.$$

3. Use the simulated predictor data to simulate response data $y_j|\mathbf{z}_j \sim \text{Binom}(1, 1, \pi_j)$.

4. Repeat steps 1. and 2. until 200 controls and 200 cases have been generated.

5. Divide the simulated data into a training set and a test set of equal size ($n = 200$), each with $n_1 = 100$ cases and $n_2 = 100$ controls.

6. Fit six different models to the training set.

   - Model 1: *Pruned tree*: A full classification tree has been fitted to the training set. The tree has been pruned, where the optimal tree size has been chosen using cross-validation.
   - Model 2: *Logistic GLM*: A logistic GLM has been fitted to the training set using the AIC score for choosing the best model.
   - Model 3: *Bagged tree*: An ensemble of trees has been fitted to separate bootstrap samples of the training set.
   - Model 4: *Bagged logistic GLM*: An ensemble of logistic GLMs has been fitted to separate bootstrap samples of the training set. Each of the logistic GLMs has been fitted using forward stepwise regression and the AIC score for choosing the best model.

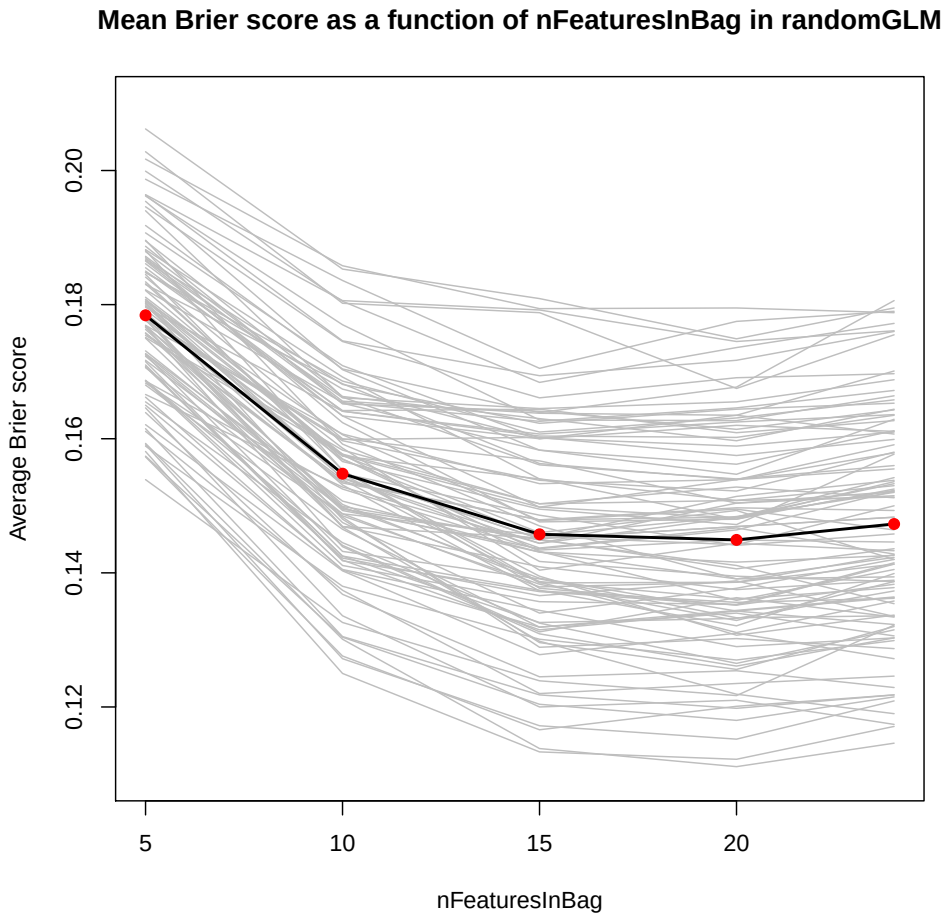


Figure 5.6: The Brier score for different choices of `nFeaturesInBag` based on 100 simulated training and test sets, each of size $n = 200$. The grey lines show the Brier scores for each individual simulated training and test set. The black line shows the mean value. The red dots indicate the values of `nFeaturesInBag` for which the Brier score has been calculated.

- Model 5: *Random forest*: An ensemble of trees has been fit to bootstrap samples of the training set for different choices of the tuning parameter `mtry`. The random forest corresponding to the `mtry` with the lowest OOB error was chosen.
- Model 6: *Random GLM*: Two randomGLMs have been fitted to the training set, with `nFeaturesInBag` = 15 and 20. The model with the lowest OOB error was chosen. This model thus consists of an ensemble of logistic GLMs, each fitted to a bootstrap sample of the training set. Each model has been fit by stepwise regression using the AIC score to choose the best model. At each step in the forward regression only a random sample of `nFeaturesInBag` predictors have been used.

7. Estimate the training error of each model by calculating the misclassification rate.
8. Make separate predictions for the test set, using each of the fitted models.
9. Estimate the prediction error using the Brier and AUC scores.

5.7 Results

The results based on 1000 repeats of simulation and model fitting are presented in Table 5.2. These values are the sample means for the statistics: AUC score, Brier score and training error. The training error has been estimated is the misclassification rate. These results are also presented graphically in Figures 5.8, 5.9 and 5.10, where we see the mean values of the AUC score, the Brier score and the training error respectively. The dots indicate the mean values, while the bars indicate the interquantile range of the score statistics. The red lines indicate the maximum or minimum mean.

In Section 3.4.3 we saw that the AUC score is the area under an ROC curve. An ROC curve is created by plotting the sensitivity against the specificity over all possible threshold values. The AUC ranges from 0 to 1, where a higher value indicates a better classifier. In Figure 5.8 and Table 5.2 we see that the random GLM has obtained the highest AUC score, which is equal to 0.791. This is a high AUC score, and implies that the model has high predictive ability. The bagged logistic GLM and the logistic GLM have very similar AUC scores. The pruned tree shows bad predictive power, with an AUC score equal to 0.640. The bagged tree and random forest show quite similar performance, and are comparable to the performance of the logistic models. We see that their interquantile ranges overlap with the interquantile ranges of the logistic models.

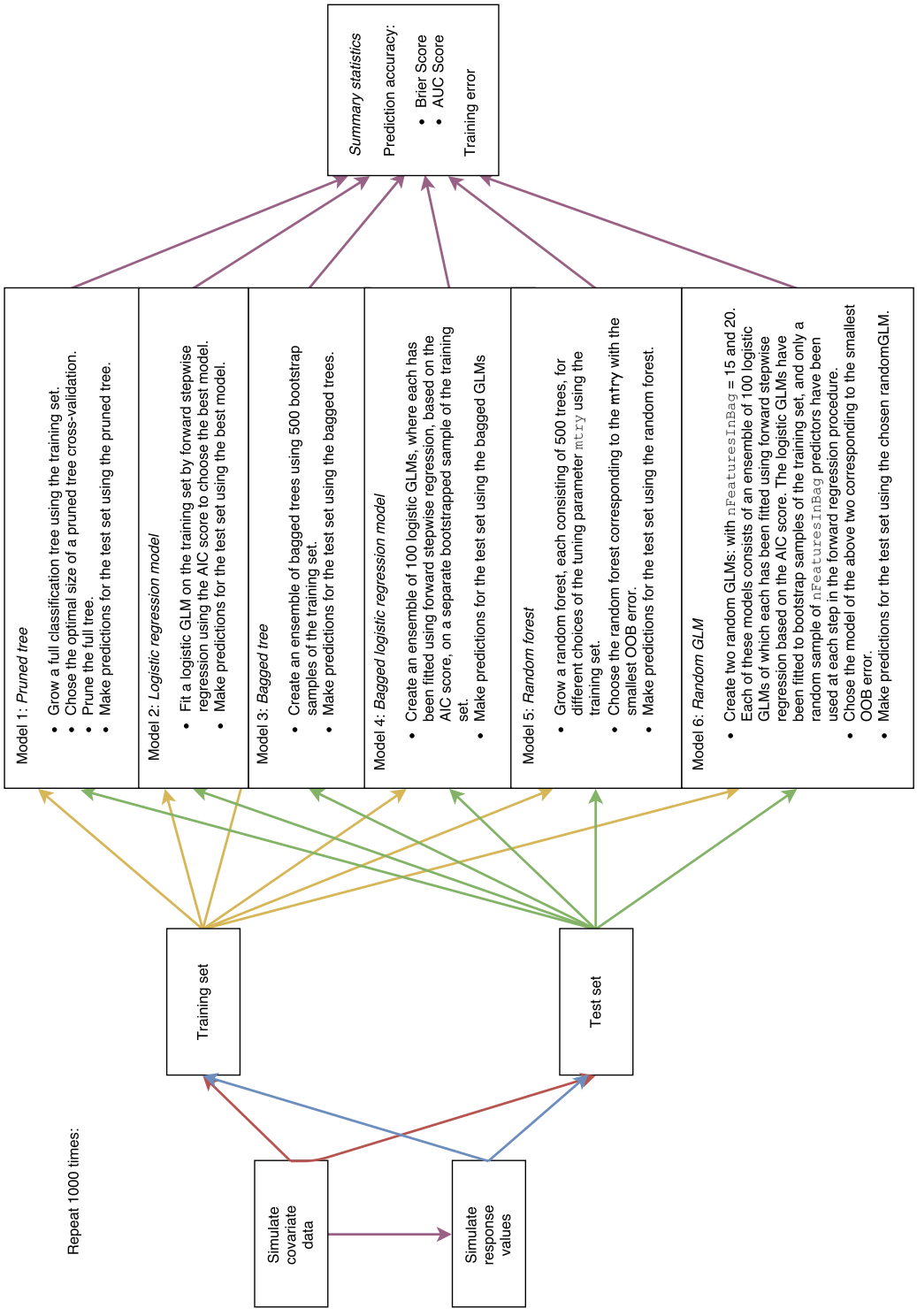


Figure 5.7: A graphical representation of the procedure of model fitting to the simulated data sets. This procedure (of simulating and model fitting) has been repeated 1000 times.

Table 5.2: The mean values (on the test sets) based on 1000 simulated test and training set of score statistics for the 6 different models.

	Pruned tree	Logistic GLM	Bagged tree	Bagged logistic	Random forest	Random GLM
AUC score	0.640	0.781	0.744	0.787	0.751	0.791
Brier score	0.235	0.197	0.206	0.192	0.204	0.188
Training error	0.305	0.229	0.321	0.292	0.313	0.283

From Section 3.4.5 we know that the Brier score is a proper score function which measures the accuracy of probabilistic predictions. It is defined as the mean squared distance between the predicted probability and the true outcome and ranges from 0 to 1, where a lower value indicates a better predictive power. In Figure 5.9 and Table 5.2 we see that the random GLM has the lowest Brier score, which is equal to 0.188. The bagged GLM and the logistic GLM show comparable values. The pruned tree shows bad predictive performance. The bagged tree and random forest give similar results, and are comparable to the logistic models. Again we see that the interquantile ranges overlap for 5 of the models.

The training error has been estimated by calculating the misclassification rate of the fitted model on the training set. We saw in Section 3.4.2 that one cannot rely on the training error to evaluate the predictive performance of a statistical model. If a model has a very low training error, it might be because it overfits the data, and will give a bad predictive performance. We see from Figure 5.10 and Table 5.2 that the logistic GLM has the lowest training error. However, based on a quite low the test error, it does not seem to overfit the data.

5.8 Discussion and Conclusion

In our simulation study we have simulated 1000 artificial training sets and 1000 artificial test sets. The predictor values have been simulated using the sample means, standard deviations and correlation matrices of the original data set. The response values have been simulated, one at a time, using a logistic model and the simulated predictor values. We have fitted 6 different statistical models to each of the training sets. We have used these models to make predictions for each of the test sets. The predictive powers (test errors) of the models have been estimated by calculating the AUC and Brier scores. The training errors have been estimated by calculating the misclassification rates when the models have

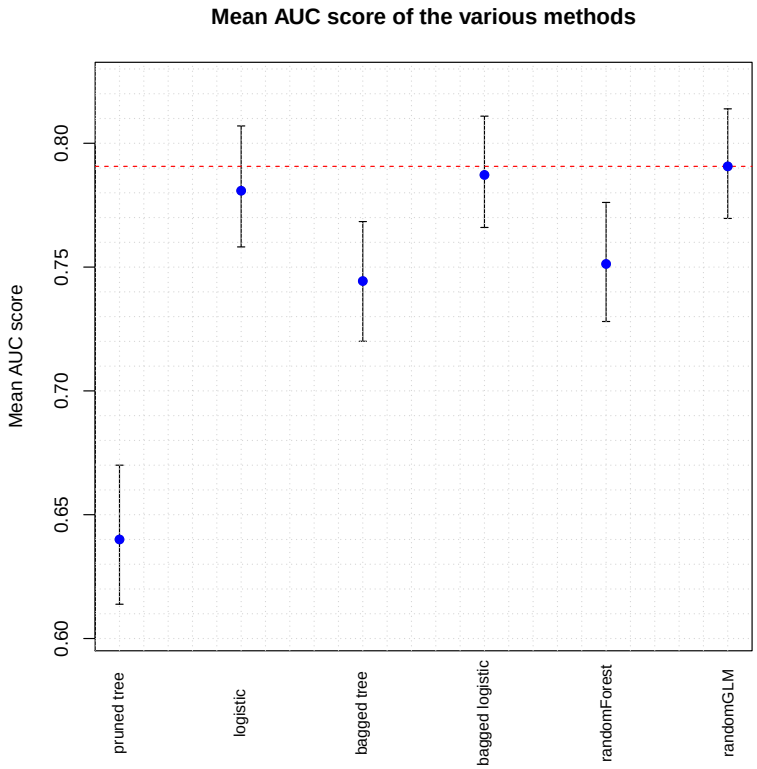


Figure 5.8: The AUC scores based on 1000 repetitions of simulation and model fitting. The blue dots show the mean AUC score on the test set for each model. The black bars show the interquantile ranges.

made predictions on the same training sets as they were fitted to.

Section 5.7 showed that the random GLM gave the overall best predictive performance. However, the bagged GLM and the logistic GLM showed very comparative results. The conclusion is that one may gain a slight improve in the prediction accuracy by creating an ensemble of logistic regression models. However, this comes at the cost of the model simplicity and interpretability and there is not very much to gain. A drawback of the random GLM is that is has a slightly longer computational time compared to the tree-based methods.

The pruned tree showed an overall bad performance and should not be recommended as a

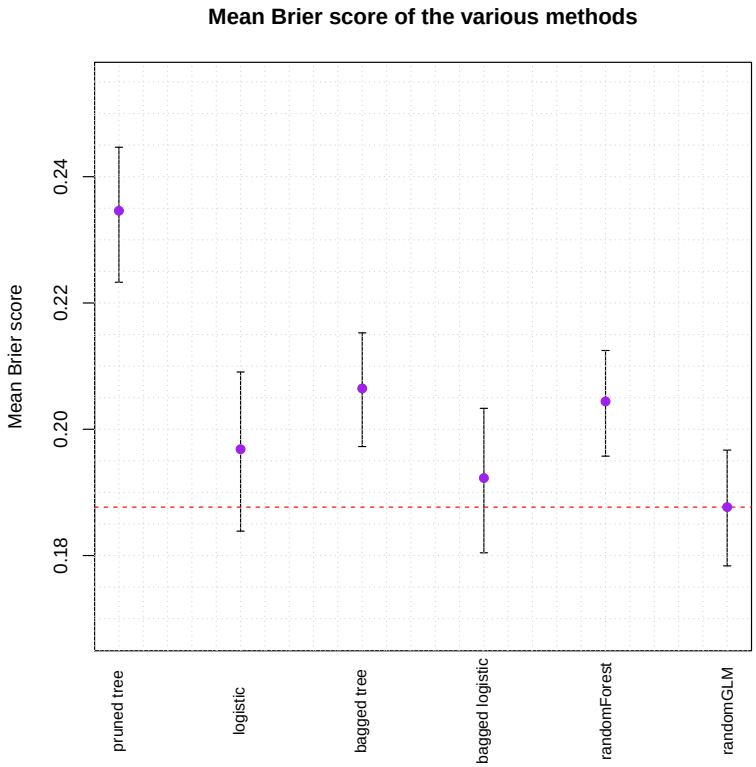


Figure 5.9: The Brier scores based on 1000 repetitions of simulation and model fitting. The purple dots show the mean BS score on the test sets for each model. The black bars show the interquantile ranges.

predictive tool to a similar data set as our simulated one. However, by creating an ensemble of trees, the predictive performance can be greatly improved. We have seen that the bagged tree and random forest show good performance, and comparable results as the logistic models. The reason for why the logistic models outperformed the tree-based methods might be that our simulated response values have been created using a logistic model. We will test this hypothesis when fitting the same models to our original data set in Chapter 6.

In this simulation study we have not looked at how accurately the methods have been able to estimate the original model parameters, shown in Table 5.1. This is because it would be very difficult to obtain such conclusions using our models.

Figure 5.10: The training errors based on 1000 repetitions of simulation and model fitting. The orange dots show the mean training error for each model. The black bars show the interquantile ranges.

Chapter 6

Data Analysis

Motivated by the simulation study in Chapter 5, we have chosen to fit 6 different statistical models also to our HUNT data set: a *pruned tree, logistic GLM, bagged tree, bagged logistic GLM, random forest* and *random GLM*. Since we in this thesis are not only interested in identifying the most relevant predictors for predicting an event of MI, but also in evaluating the *predictive* performance of different statistical models, we have divided the data analysis into two parts. Section 6.1 aims at evaluating the predictive power of our chosen statistical models, when applied to our HUNT data set. This has been done by cross-validation. Section 6.2 aims at identifying the most relevant predictors for an event of MI, and the original data has been used for this purpose.

6.1 The Predictive Power of Our Statistical Models

To estimate the predictive power of our statistical models we have, inspired by Song et al. (2013), used cross-validation. The following procedure was repeated 100 times: We randomly partitioned the HUNT data set into two parts: a training set and a test set. The training set contained data of 131 randomly chosen subjects. The test set contained data of the remaining 66 subjects. We fitted 6 different models to the training set. The fitting procedure followed the procedure described in Section 5.5. The training errors were calculated by counting the misclassification rates when using the fitted models to make predictions on the observations in the training set. Thereafter, the 6 different models were used to make separate predictions on the test set. The test errors were calculated by using the Brier and AUC score. Figure 6.1 shows a graphical representation of this procedure.

The results are presented in Figures 6.3, 6.2 and 6.4, where the AUC score, Brier score

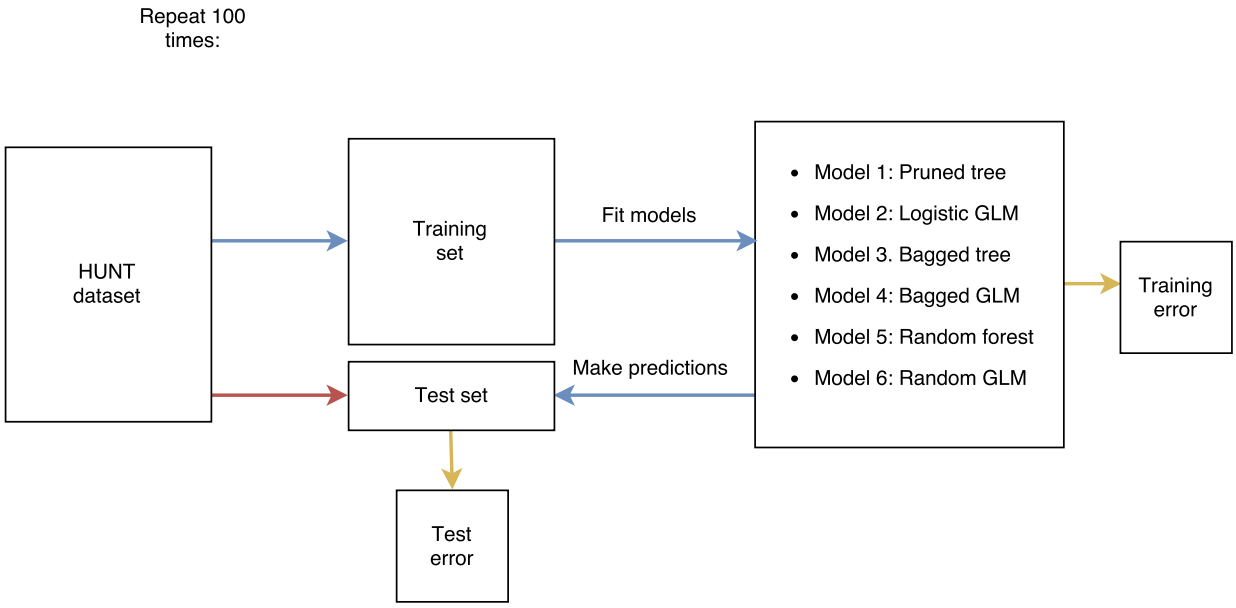


Figure 6.1: A graphical representation of the procedure of evaluating the predictive power of our 6 different statistical methods, when applied to the HUNT data set.

and the training error is plotted for the various statistical models. The dots show the mean values, while the bars indicate the interquantile ranges. The red lines show the maximum or minimum mean value. The corresponding numerical results are found in Table 6.1.

We see that the statistical models applied to the HUNT data set give similar results as when applied to our artificial data sets. In total 5 of the models perform well and have interquantile ranges which overlap each other. The model giving the lowest test error, and hence has the best predictive performance, is the random GLM, followed by a bagged GLM and a logistic GLM. Furthermore, we see that a single tree makes a bad performance, but that by applying ensemble methods on a tree, the predictive performance increases substantially. The logistic regression model has the lowest training error, which could imply that the model has been overfit to the data. However, the AUC and Brier scores do not support this assumption.

Table 6.1: The mean values of score statistics for the 6 different models when applied to the HUNT data set, based on 100 repetitions of training and testing.

	Pruned tree	Logistic GLM	Bagged tree	Bagged logistic	Random forest	Random GLM
AUC score	0.622	0.795	0.777	0.809	0.790	0.816
Brier score	0.243	0.199	0.196	0.185	0.193	0.180
Training error	0.299	0.183	0.288	0.269	0.270	0.254

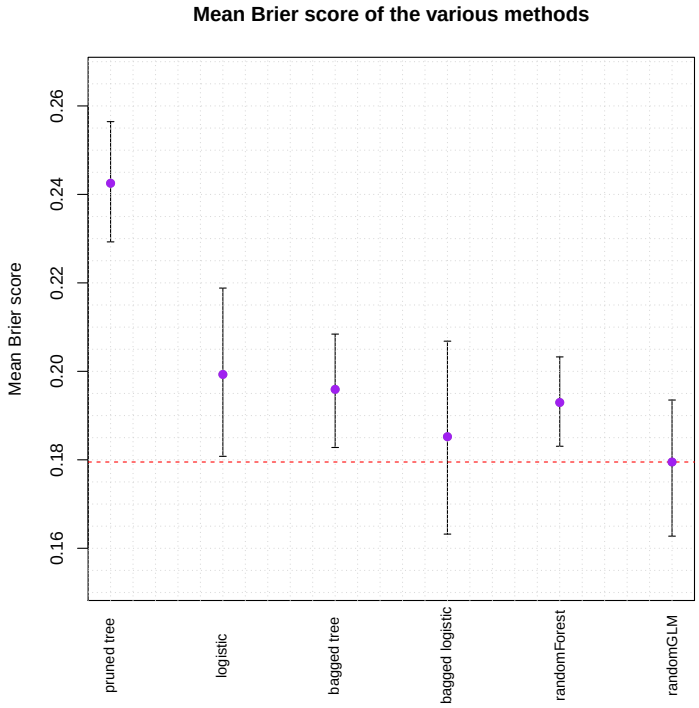


Figure 6.2: The mean Brier score for different models, based on 100 repetitions of training and testing. The purple dots show the mean values for each model, the black bars show the interquantile ranges, while the red line shows the lowest mean value.

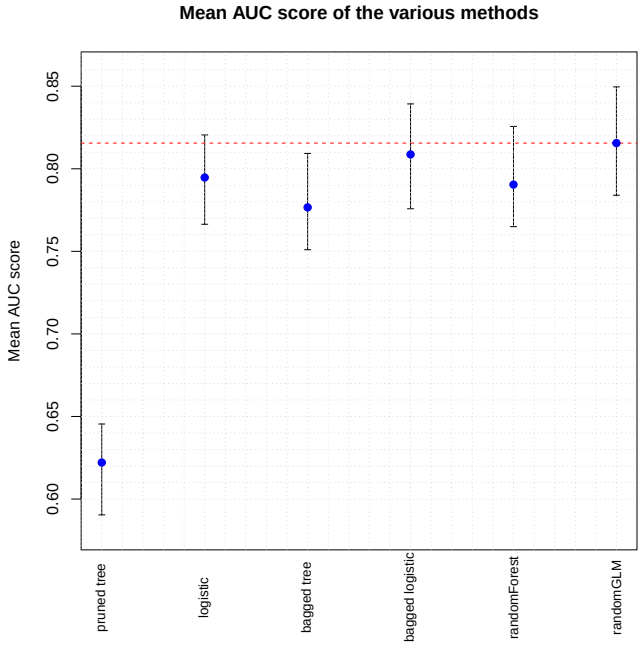


Figure 6.3: The mean AUC score for different models, based on 100 repetitions of training and testing. The blue dots show the mean values for each model, the black bars show the interquantile ranges, while the red line shows the highest mean value.

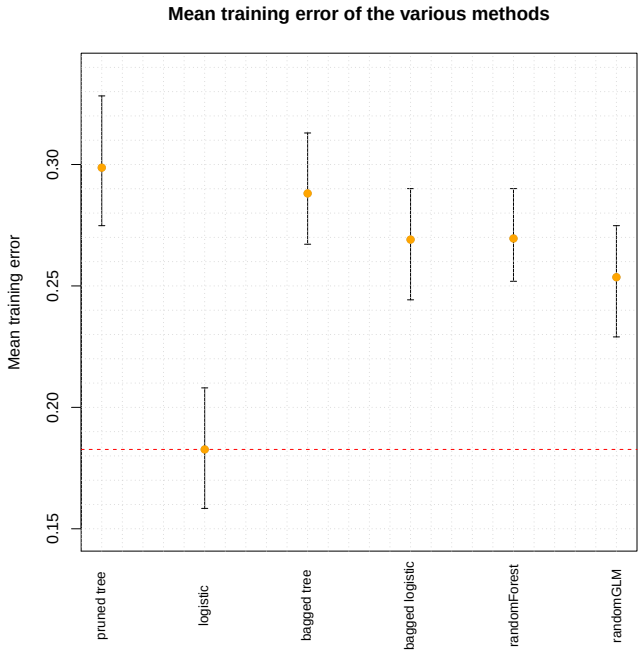


Figure 6.4: The mean training error for different models, based on 100 repetitions of training and testing. The orange dots show the mean values for each model, the black bars show the interquantile ranges, while the red line shows the lowest mean value.

6.2 Final Results

This section presents the medical results, from fitting our statistical models to the HUNT data set. The procedure has been to fit the 6 models to the whole data set once. The presentation of the results is divided into two parts. Firstly, in Section 6.2.1, we will present the results for one model at a time. Secondly, in Section 6.2.2, we compare results across the models. This chapter contains a mix of the HUNT variable coding, in the `R` outprints, and of the coding used in this thesis. Please see Section 5.1 for an overview of the coding of the predictors.

6.2.1 Model-Wise Results

Model 1: Pruned Tree

We present the result of our pruned tree by plotting the actual tree, in Figure 6.5. This is a small tree, with only 4 terminal nodes, which probably explains the low predictive power. The miRNA coded by hsa_let_7g_5p gives the first splitting criterion. Thereafter comes the predictor smoking status (smoke), followed by serum triglycerides (setrig) and the miRNA coded by hsa_miR_26a_5p. The predictor serum triglycerides is not included in the Framingham risk score.

Model 2: Logistic GLM

We present the results of the logistic GLM in the usual manner, by an outprint created by the `summary` function. This outprint shows which predictors have been included by the forward stepwise procedure in creating the best model based on the AIC score. The values under the `Estimate` column show the estimated coefficients, where $\beta_0 = -2.006344$, $\beta_{hsa\_miR\_144\_3p} = -1.592524$, $\beta_{smoke} = -9.88172$ etc. The columns `Std.Error` and `z value` show the corresponding standard errors and z values. The last column shows the p-values, where the significant predictors for testing, $H_0 : \beta = 0$ vs. $H_1 : \beta \neq 0$, are the ones with the lowest p-value. We see that 4 of the miRNAs have been included in the final GLM and are significant. Furthermore, we see that the predictors waist-to-hip ratio (whr), BMI and serum glucoe non-fast (seglu) have been included in the model, which are not present in the Framingham risk score.

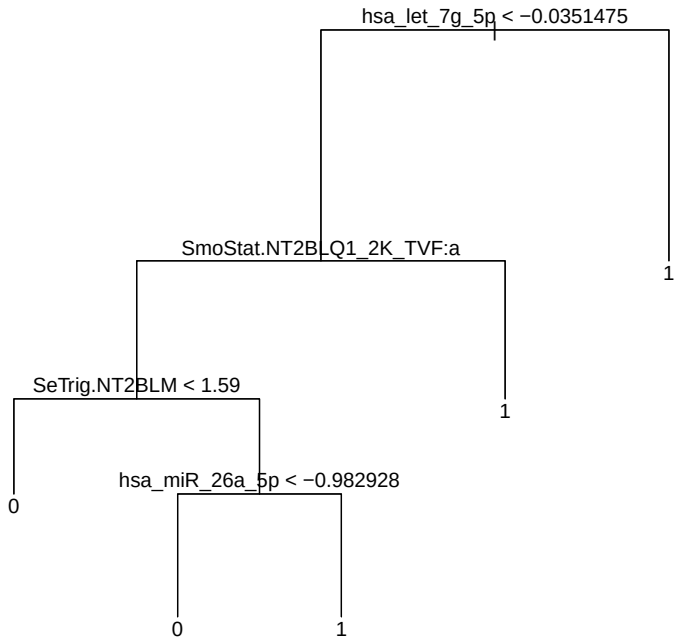


Figure 6.5: The pruned tree fitted to the HUNT data set, where the size of the tree has been determined by cost-complexity pruning. The variable names follow the HUNT coding.

Model 3: Bagged Tree

Figure 6.6 shows the two types of variable importance plots for bagged trees. We see that the top predictors differ somewhat between the plots, but that the most of the top predictors are the same. Importantly, we see that many of the miRNAs have been identified as important predictors. Furthermore, the predictors serum triglycerides (setrig) and body mass index are among the top-ten predictors in both plots. These predictors are not included in the Framingham risk score.

```
Call:
glm(formula = MI ~ hsa_miR_144_3p + SmoStat.NT2BLQ1_2K_TVF +
    BMI.NT2BLM + SeHDLChol.NT2BLM + hsa_miR_26a_5p + SeChol.NT2BLM +
    BPSystMn23.NT2BLM + hsa_miR_424_5p + SeGluNonFast.NT2BLM +
    hsa_miR_191_5p + WHR, family = "binomial", data = ds)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1298  -0.6939  -0.1033   0.6329   2.4261

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)              -2.00634  627.93381  -0.003  0.99745
hsa_miR_144_3p           -1.59252    0.51484  -3.093  0.00198 **
SmoStat.NT2BLQ1_2K_TVF1  -9.88172  627.91828  -0.016  0.98744
BMI.NT2BLM                0.17445    0.07664   2.276  0.02284 *
SeHDLChol.NT2BLM         -1.33170    0.60703  -2.194  0.02825 *
hsa_miR_26a_5p            1.59039    0.68699   2.315  0.02061 *
SeChol.NT2BLM             0.54031    0.20022   2.699  0.00696 **
BPSystMn23.NT2BLM         0.03498    0.01241   2.818  0.00483 **
hsa_miR_424_5p            0.41011    0.23906   1.715  0.08626 .
SeGluNonFast.NT2BLM      -0.60204    0.29078  -2.070  0.03842 *
hsa_miR_191_5p           -1.34736    0.77032  -1.749  0.08028 .
WHR                       5.91296    3.71684   1.591  0.11164
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 273.05  on 196  degrees of freedom
Residual deviance: 164.36  on 185  degrees of freedom
AIC: 188.36

Number of Fisher Scoring iterations: 17
```

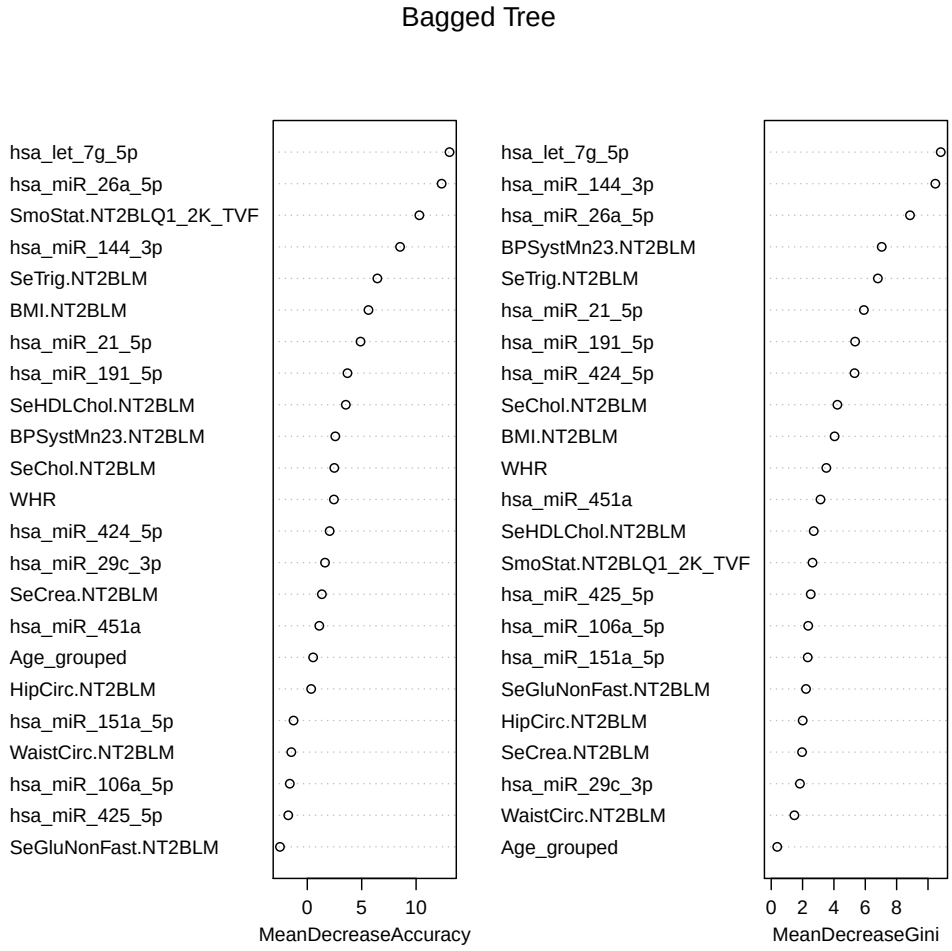


Figure 6.6: Variable importance plots when growing a bagged tree on our HUNT data set.

Model 4: Bagged Logistic GLM

The results for the bagged logistic GLM are presented in Figure 6.7, by a variable importance. Recall that this variable importance plot is created by counting the number of times a given predictor has been chosen by the stepwise forward regression procedure and cannot be directly compared with the variable importance plots of tree ensembles. Again we see that many of the miRNAs have shown to bee important predictors. In addition, from the predictors not included in the Framingham score, serum glucose non fasting (seglu) and

BMI, are highly relevant.

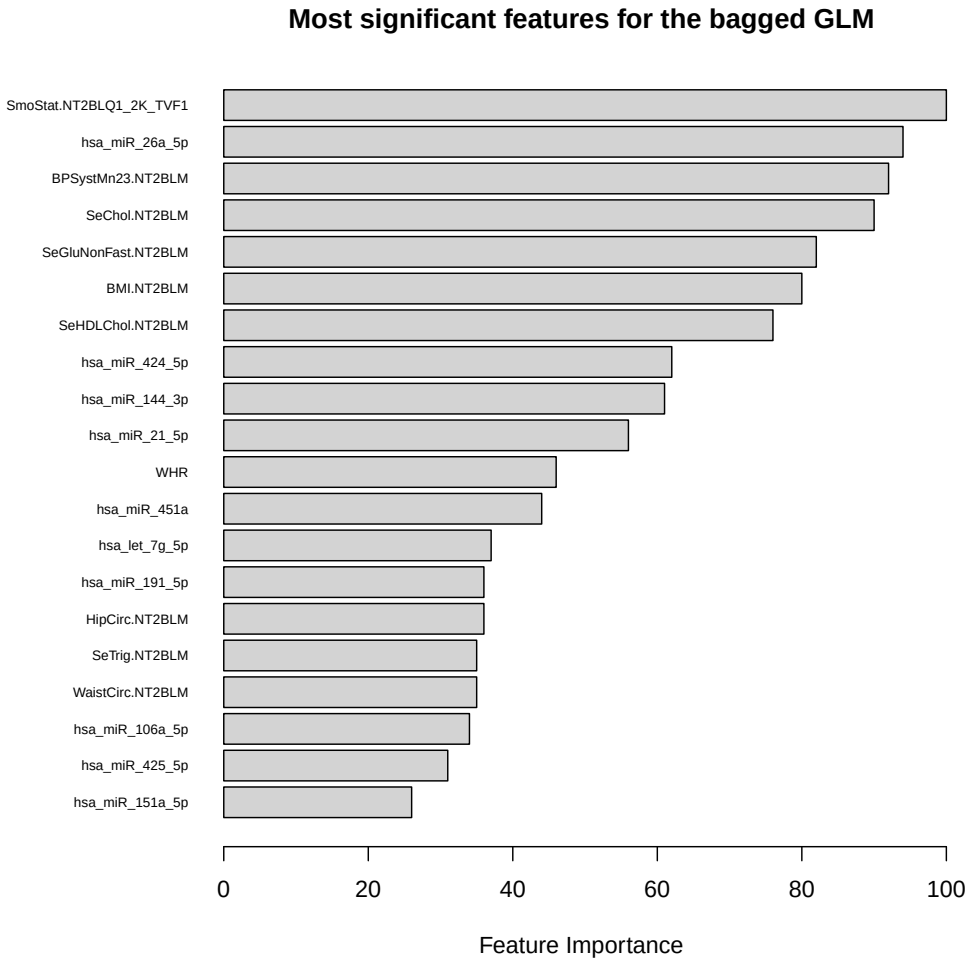


Figure 6.7: Variable importance plots when bagging a logistic GLM on our HUNT data set.

Model 5: Random Forest

The results from the random forest are represented through the variable importance plots in Figure 6.8. We again see that the two types of variable importance plots differ somewhat in the top predictors. When comparing these plots with the corresponding variable importance

plots for the bagged tree, we see that they are very similar. Again we see that several of the miRNAs play an important role. Furthermore, we see that the predictors BMI and serum trilglycerides (setrig), which are not present in the Framingham risk score, are among the top ten predictors.

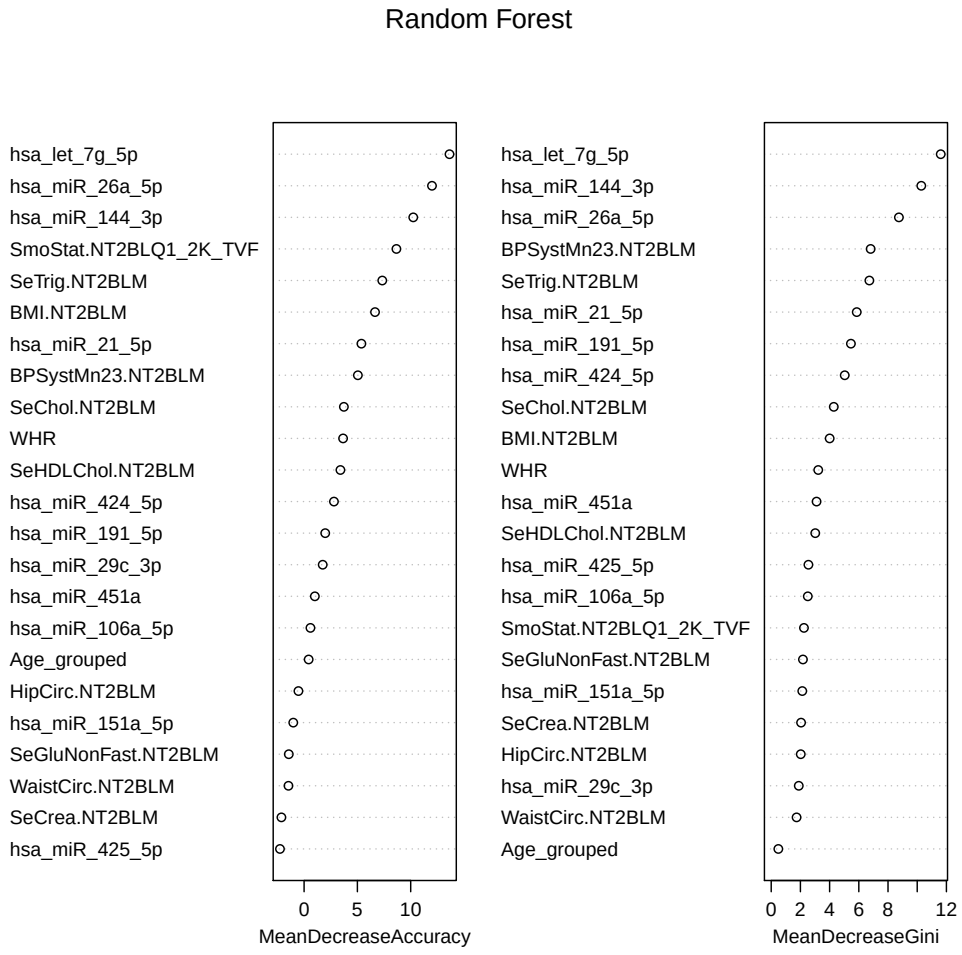


Figure 6.8: Variable importance plots when growing a random forest on our HUNT data set.

Model 6: Random GLM

Variable importance plot for the random GLM is presented in Figure 6.9. This variable importance plot is quite similar to the variable importance plot of the bagged GLM, but not similar to the variable importance plot of the random forest. We see that 4 of the miRNAs are among the top ten significant predictors. The predictor BMI is significant.

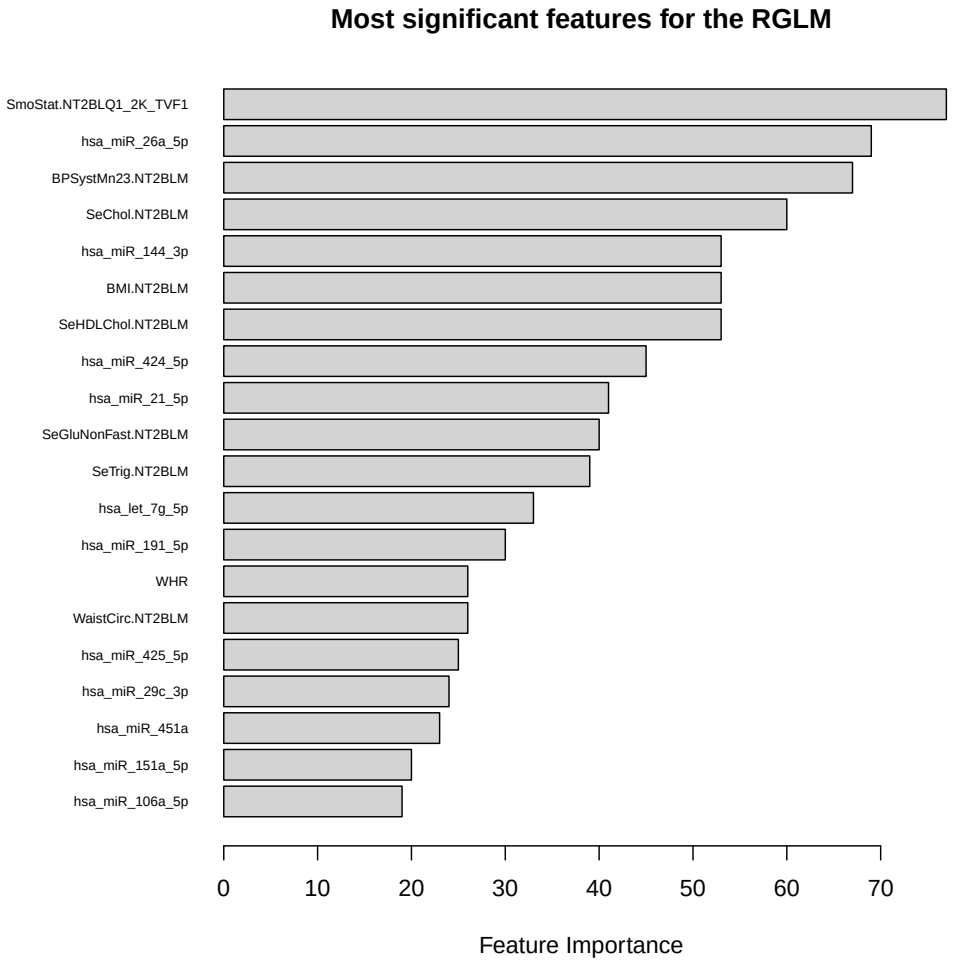


Figure 6.9: Variable importance plots when fitting a random GLM on our HUNT data set.

6.2.2 Overall Results

The previous section showed the results for one model at a time. In this section we aim at comparing the results across the models.

We have fitted 6 different models. In 3 of the models the underlying structure was a classification tree while in the remaining 3 models the underlying structure was a logistic GLM. We have seen that the tree based methods tend to identify the same predictors as the most relevant. The same is true for the methods based on logistic GLMs. The most important predictors however tend to differ across these two model groups. This can also be seen by drawing a heatmap of the estimated probabilities. Such a heatmap is shown in Figure 6.10, where each small rectangle represents one predicted probability for the subjects in the HUNT data set, when using the fitted models. The predicted probabilities have been translated into colors, based on the magnitude of the numerical values. The color-coding is found in the top-left corner in the figure. Further, each row corresponds to one of our 6 methods. The row named *MI event* corresponds to the true outcome (case/control) of each subject. Each column corresponds to one subject. The dendograms show the arrangement of clustering. The dendogram corresponding to the model types has clearly identified two groups of methods: the tree-based methods and the methods based on logistic GLMs. This can also be seen by comparing the coloring across one subject at a time: the tree-based methods tend to give similar classifications and the methods based on the logistic GLM tend to give similar classifications.

The heatmap in Figure 6.10 has identified a group of subjects which none of our models have been able to classify correctly, the leftmost grouping. It would be very interesting to investigate what properties these subjects share. Why were we not able to capture this group? Maybe the inclusion of additional predictors in our models could improve the predictions.

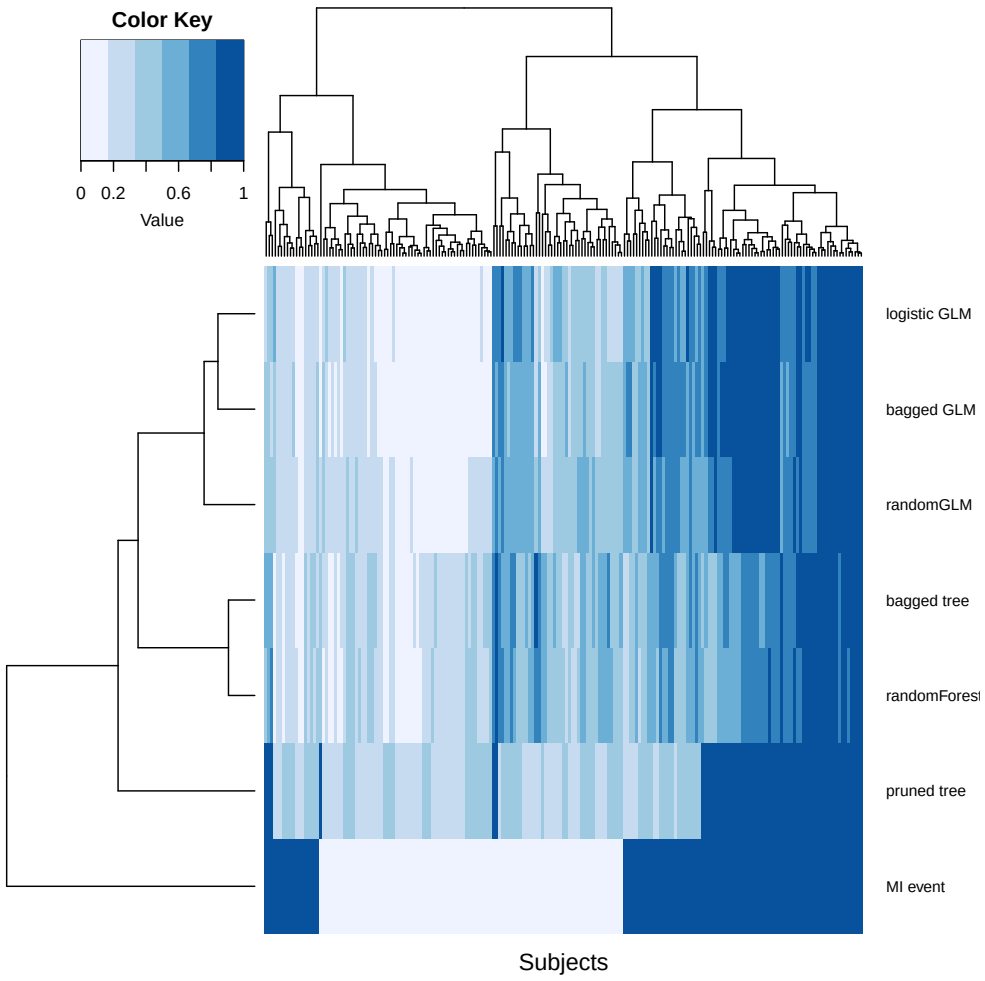


Figure 6.10: A heatmap comparing predictions obtained by using different statistical methods. Each row corresponds to one of our 6 methods. The row named *MI event* corresponds to the true outcome (case/control) of each subject. Each column corresponds to one subject. One colored rectangle represents one prediction for subject i, where the color key shows the interval of each color. The dendograms show the hierarchical clustering of subjects and methods using Euclidean distance function.

Chapter 7

Discussion and Conclusion

7.1 Statistical Issues

The main motivation for this thesis, from a statistical point of view, was to see if one can increase the prediction accuracy when predicting a MI event by applying ensemble methods on the data set. Based on literature, we expected to see a greater increase in prediction accuracy for the tree-based methods than the models based on logistic regression, as trees have low bias but high variance. Our data analysis has confirmed this. The prediction accuracy increased when creating a tree ensemble. We saw a great improvement when creating a bagged tree, and a further improvement when growing a random forest, compared to the single pruned tree. We have also seen that the predictive power of a logistic GLM can be improved when combining the prediction from an ensemble of GLMs. Again we saw an increase in the prediction accuracy when creating a bagged GLM, and a further increase when creating a random GLM. Thus, by including only a subset of predictors in each model, and then combining the models to make one prediction, is an effective method.

There is an important difference between how the ensemble of GLMs and the tree ensembles made predictions (final classifications), which needs to be addressed. The ensembles of GLMs made the final prediction by averaging the *predictions*, $\hat{\pi}^b(\mathbf{x}_i)$ from each model, and then classified according to a threshold value of 0.50. However, the tree based methods made the final prediction by the *majority vote* over all classifications $\hat{C}^b(\mathbf{x}_i)$. The classification, according to a threshold value of 0.50, was made by each tree separately. The difference between these two procedures can be compared to the difference between the mean value and median of a set of numbers. The methods based on logistic regression obviously use the mean value. In the case of two classes, if one would sort all of the

classifications, so that all classifications corresponding to one class would be lined up after each other, followed by all classifications corresponding to the other class, the majority vote would be the same as the median value. Let's examine this further by an example: Suppose we fit 5 models and obtain the following probabilities: 0.4, 0.4, 0.4, 0.4 and 0.9. The classification made by the methods which average the probabilities would be 1, while it would be 0 if classified by taking the majority vote (using 0.5 as the cut-off value). The two procedures thus have their pros and cons: By averaging the predictions no information is lost. We do not only get the final classification, but the probability for belonging to the class 0 or 1. However, this method is not robust to outliers. By taking a majority vote, outliers have a smaller influence on the result. This difference in how the classifications have been obtained should not have a large influence when using the Brier and AUC scores as a measure of the prediction accuracy. We used the class proportions for the tree based methods when calculating the Brier and AUC scores.

There is one aspect of the random GLM which we found somewhat strange. Based on the recommendation by Song and Langfelder (2013), we should have included 22 out of 23 in each step in the forward selection procedure of our random GLM. The models would not be independent when almost all of the predictors are present in each model. We have used a loop which searches for the parameter value which gives the smallest OOB error for our HUNT dataset. This resulted in using `nFeaturesInBag` = 17. In the simulation study we used 15 or 20, whichever gave the lowest OOB error of the two.

7.2 Discussion of the Medical Results

The main motivation for this thesis, from a medical point of view, was to improve the prediction accuracy when predicting a MI event for a new subject. We were also interested in finding out if microRNAs have a predictive potential for developing MI. Our results have shown that the best predictive power was obtained by the random GLM. That is, an ensemble of forward stepwise obtained logistic GLMs, where only a subset of predictors were considered as candidates in each step of the forward selection procedure. We have evaluated the predictive power of our methods by dividing our data set into a train set of $\frac{2}{3}$ of the observations, and a test set of $\frac{1}{3}$ of the observations, 100 times. The Brier score for the random GLM was 0.180. The AUC score for the random GLM was 0.816. The AUC score of a single logistic GLM was 0.795. We can compare this result to the results obtained by Velle-Forbord (2017). Velle-Forbord (2017) used the same data set and fitted logistic regression models using best subset selection based on the AIC score. To evaluate the prediction accuracy, Velle-Forbord (2017) used leave-one-out cross-validation. The AUC

score obtained by Velle-Forbord (2017) was 0.80. We have thus increased the AUC score. However, in addition to applying ensemble methods, we have added additional predictors.

Further, we have seen that many predictors, which are not included in the Framingham risk score, are relevant in our models. In particular the predictors BMI, Serum triglycerides (setrig) and Serum glucose non fasting (seglu). We have also seen that microRNAs are very relevant. The microRNAs have shown high importance in all of our models. However, the specific microRNAs have varied. This is can be due to the high correlation between the microRNAs, which we saw in Figure 5.4. The three, most relevant microRNAs have shown to be hsa_let_7g_5p, hsa_miR_26a_5p and hsa_miR_144_3p.

None of our models have captured an effect of the subjects age, which is a very important predictor in the Framingham risk score. The reason for this is probably that the age of the subjects in the HUNT study has been relatively homogeneous. The minimum age was 50 and the highest 80, with median value of 68 years. The Framingham risk score uses age intervals from 20 to 79 years.

To improve the predictive power further, one could start with the heatmap in Figure 6.10. The dendogram has identified a group of subjects which none of our models have been able to classify correctly, the outermost left grouping. What is specific with these subjects? What do they have in common? Maybe an additional predictor, which we have not included in our models, would have captured this group.

This thesis has shown that by applying ensemble methods one can improve the prediction accuracy. However, this increase is not very high. There are two drawbacks which should be taken into consideration: computational cost and model interpretability. However, for a data set of the size as our HUNT data set, the computational cost is low enough so that an average computer can create the model in less than a minute. The problem of model interpretability is somewhat solved by the variable importance plot.

Bibliography

Agresti, A. (1996). *An Introduction to Categorical Data Analysis*. John Wiley and Sons, Inc.

Ardekani, A. M. and M. M. Naeini (2010). The role of microRNAs in human diseases. *Avicenna Journal of Medical Biotechnology*. doi:0.1007/978-1-62703-748-8_3.

Berk, R. A. (2016). *Statistical Learning from a Regression Perspective*. Springer. ISBN: 978-3-319-44047-7.

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

Breiman, L., J. Friedman, C. J. Stone, and R. Ohlsen (1984). *Classification and Regression Trees*. Brooks/Cole Publishing.

Bye, A., H. Røsjø, J. Nauman, G. J. Silva, T. Follestad, T. Omland, and U. Wisløff (2016). Circulating microRNAs predict future fatal myocardial infarction in healthy individuals. The HUNT study. *Journal of Molecular and Cellular Cardiology*. doi:10.1016/j.yjmcc.2016.05.009.

Casella, G. and R. L. Berger (2008). *Statistical Inference* (Second ed.). Duxbury Press.

Chen, Y. (2012). Lecture notes: Information, prediction and collective intelligence. `http://www.eecs.harvard.edu/cs286r/courses/fall12/`. (Date last accessed 10-June-2017).

Czado, C., T. Gneiting, and L. Held (2009). Predictive model assesment for count data. *Biometrics 65*, 1254–1261. doi:10.1111/j.1541-0420.2009.01191.x.

Dobson, A. J. and A. G. Garnett (2008). *An Introduction to Generalized Linear Models*. Chapman and Hall CRC.

Efron, B. (1979). Bootstrap methods: Another look at the jacknife. *The Annals of Statistics 7*(1), 1–26.

Eidlaug, M. (2017). Circulating microRNAs as predictive biomarkers of fatal myocardial infarction. The HUNT study. Hovedoppgave, Norwegian University of Science and Technology.

Gneiting, T. and A. E. Raftery (2007). Strictly proper scoring rules, prediction and estimation. *Journal of the Americal Statistical Association 102*, 359–378. doi:10.1198/016214506000001437.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning. Data Mining, Inference and Prediction*. Springer. ISBN: 978-0-387-84857-0.

Helsedirektoratet (2009). *Nasjonale retningslinjer for individuell primærforebygging av hjerte- og karsykdommer*. `http://www.helsedirektoratet.no/publikasjoner`.

Izenman, A. J. (2008). *Modern Multivariate Statistical Techniques. Regression, Classification and Manifold Learning*. Springer. ISBN: 978-0-387-78188-4.

James, G., D. Witten, T. Hastie, and R. Tibshirani (2013a). *An Introduction to Statistical Learning with Applications in R*. Springer. ISBN: 978-1-4614-7137-0.

James, G., D. Witten, T. Hastie, and R. Tibshirani (2013b). *ISLR: Data for An Introduction to Statistical Learning with Applications in R*. R package version 1.0.

Li, Y. and K. V. Kowdley (2012). MicroRNAs in common human diseases. *Genomics, Proteomics and Bioinformatics 10*(5), 246–253. doi:10.1016/j.gpb.2012.07.005.

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

Lien, T. G. (2011). Statistical analysis of quantitative PCR data. Master thesis, Norwegian University of Science and Technology.

NTNU (2017). About HUNT. `http://www.ntnu.edu/web/hunt/about-hunt/`. (Date last accessed 10-June-2017).

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Ripley, B. (2016). *tree: Classification and Regression Trees*. R package version 1.0-37.

Ripley, B. D. (1995). *Pattern Recognition and Neural Networks*. Cambridge University Press. ISBN: 978-0521717700.

Robin, X., N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez, and M. Müller (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics 12*(11).

Rodríguez, G. Lecture notes: Generalized linear models. `http://data.princeton.edu/wws509/notes`. (Date last accessed 10-June-2017).

Roediger, S. and M. Burdukiewicz (2014). *chipPCR: Toolkit of helper functions to preprocess amplification data*. R package version 0.0.8-4.

Siegert, S. (2017, January). Notes and correspondence. simplifying and generalising Murphys Brier score decompositions. *Quarterly Journal of the Royal Meteorological Society 143*, 1178–1183. doi:10.1002/qj.2985.

Song, L. and P. Langfelder (2013). *randomGLM: Random General Linear Model Prediction*. R package version 1.02-1.

The National Heart, L., B. Institute, and B. University. Framingham heart study. `https://www.framinghamheartstudy.org/about-fhs/history.php`. (Date last accessed 10-June-2017).

Therneau, T., B. Atkinson, and B. Ripley (2017). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-11.

Townsend, N., M. Nichols, P. Scarborough, and M. Rayner (2015). Cardiovascular disease in europe–epidemiological update 2015. *36*(40), 2696–705. doi:10.1093/eurheartj/ehv428.

UN, K., K. MB, B. CT, and et al (2003). Prevalence of conventional risk factors in patients with coronary heart disease. *JAMA 290*(7), 898–904.

Velle-Forbord, T. (2017). Circulating microRNAs as predictive biomarkers of myocardial infarction. the HUNT study. Hovedoppgave, Norwegian University of Science and Technology.

Venables, W. N. and B. D. Ripley (2002). *Modern Applied Statistics with S* (Fourth ed.). New York: Springer. ISBN 0-387-95457-0.

Wahid, F., A. Shehzad, T. Khan, and Y. Y. Kim (2010). MicroRNAs: Synthesis, mechanism, function, and recent clinical trials. doi:10.1016/j.bbamcr.2010.06.013.

Warnes, G. R., B. Bolker, L. Bonebakker, R. Gentleman, W. H. A. Liaw, T. Lumley,

M. Maechler, A. Magnusson, S. Moeller, M. Schwartz, and B. Venables (2016). *gplots: Various R Programming Tools for Plotting Data*. R package version 3.0.1.

Wei, T. and V. Simko (2016). *corrplot: Visualization of a Correlation Matrix*. R package version 0.77.

Winkler, R. L. (1996). Scoring rules and the evaluation of probabilities. *Test 5*, 1–60. doi:10.2307/2283486.

Appendix A

R Codes for Simulating an Artificial Data Set

This appendix shows the R codes used to simulate artificial predictor values and artificial response values, by following the procedure descried in Chapter 5. Since we want to compare the predictive power of statistical models, we simulate one big artificial data set, which we later divide into a training and test set of equal size. We start by loading all the necessary packages, and we set a sum-to-zero constraint on the categorical variables.

```
> library(foreach)
> library(MASS)
> library(caret)
> library(ModelMetrics)
> library(matrixStats)
> options(contrasts=c("contr.sum", "contr.sum"))
```

Our HUNT data set is stored in the data frame `ds`. The original coding of the predictors, and corresponding own coding of the predictors, can be found in the overview of Section 5.1. The first step is to create the logistic model, which will serve as the statistical model for MI.

```
> logistic = glm(StatCaseControl ~ BMI.NT2BLM + SeChol.NT2BLM + SeGluNonFast.NT2BLM +
+                  BPSystMn23.NT2BLM + Age_groupedby10 + hsa_let_7g_5p +
+                  hsa_miR_144_3p + hsa_miR_26a_5p + hsa_miR_21_5p + hsa_miR_424_5p +
+                  hsa_miR_191_5p, family="binomial", data=ds)
```

```
> print(signif(logistic$coefficients), digits=3)
       (Intercept)          BMI.NT2BLM       SeChol.NT2BLM SeGluNonFast.NT2BLM
           -8.1705              0.1495              0.2933             -0.2636
  BPSystMn23.NT2BLM    Age_groupedby101    Age_groupedby102    Age_groupedby103
            0.0279             -0.1057              0.0132             -0.2035
      hsa_let_7g_5p      hsa_miR_144_3p      hsa_miR_26a_5p       hsa_miR_21_5p
            0.4887             -1.4302              1.2514              0.6019
     hsa_miR_424_5p      hsa_miR_191_5p
            0.5198             -1.2141
```

Since we want our artificial data set to resemble the original data set, we calculate the sample means, standard deviations and sample covariance matrices, and use these to simulate predictor values.

```
> ages = factor(c("(49, 59]", "(59, 69]", "(69, 79]", "(79, 89]"))
> age.effects = matrix(c(1, 0, 0, -1, 0, 1, 0, -1, 0, 0, 1, -1), ncol=3)
>
> smoke = as.factor(c(0,1))
>
> measurements.mu = c(mean(ds$WaistCirc.NT2BLM), mean(ds$HipCirc.NT2BLM),
+                     mean(ds$Hei.NT2BLM), mean(ds$Wei.NT2BLM))
> measurements.Sigma = cov(ds[,  c("WaistCirc.NT2BLM", "HipCirc.NT2BLM",
+                                  "Hei.NT2BLM", "Wei.NT2BLM")])
>
> hdl_setrig.mu = c(mean(ds$SeHDLChol.NT2BLM), mean(ds$SeTrig.NT2BLM))
> hdl_setrig.Sigma = cov(ds[, c("SeHDLChol.NT2BLM", "SeTrig.NT2BLM")])
>
> mRNA.mu = sapply(ds[,279:289], FUN=mean)
> mRNA.Sigma = cov(ds[,279:289])
```

We initialize an empty data frame `df.sim` in which we will store the simulated values.

```
> df.sim = data.frame(statcasecontrol = integer(), age = factor(),
+                     sechol = double(), hdlchol = double(), bp = double(),
+                     setrig = double(), secrea = double(), seglu = double(),
+                     smostat = factor(), wc = double(), hc = double(),
+                     bmi = double(), whr = double(), hsa_let_7g_5p = double(),
+                     hsa_miR_106a_5p = double(), hsa_miR_144_3p = double(),
+                     hsa_miR_151a_5p = double(), hsa_miR_191_5p = double(),
+                     hsa_miR_21_5p = double(), hsa_miR_26a_5p = double(),
+                     hsa_miR_29c_3p = double(), hsa_miR_424_5p = double(),
+                     hsa_miR_425_5p = double(), hsa_miR_451a = double())
```

Now we can simulate the artificial data set. By using the `repeat` function, at each iteration, the predictor values for one subject were simulated at a time. This by using the sample mean, standard deviations and covariance matrices of the original data set and making calls to the functions `rnorm` and `mvrnorm`, which give random realizations of the univariate and multivariate normal distribution functions, respectively. By setting the parameter `n` equal to 1, we tell the function that we only want one realization for each iteration of the `repeat` function. Using these predictor values, and our logistic model for MI, a probability `pi` for $Y = 1$ for the subject has been calculated. Using this probability, a call has been made to the `rbinom` function, which gives random realizations from the binomial distribution. According to the value returned by `rbinom`, the subject has been classified as an case ($Y = 1$) or an control ($Y = 0$). Corresponding counters `ncase` or `ncontrol` are incremented by one. The `repeat` function terminates when both the number of cases and controls is greater or equal to 200.

```
> ncase = 0; ncontrol = 0
> {
+   repeat{
+     mRNA = mvrnorm(n = 1, mu = mRNA.mu, Sigma = mRNA.Sigma)
+
+     age.indx =  sample(1:4, 1)
+     age = ages[age.indx]
+
+     sechol = rnorm(n = 1, mean = sechol.mu, sd = sechol.sd)
+     bp = rnorm(n = 1, mean = bp.mu, sd = bp.sd)
+     secrea = rnorm(n = 1, mean = secrea.mu, sd = secrea.sd)
+     seglu = rnorm(n = 1, mean = seglu.mu, sd = seglu.sd)
+
+     hdl_setrig = mvrnorm(n = 1, mu = hdl_setrig.mu, Sigma = hdl_setrig.Sigma)
+
+     smostat = sample(smoke, 1, prob=c(0.4, 0.6))
+
+     measurements = mvrnorm(n = 1, mu = measurements.mu,
+                            Sigma = measurements.Sigma)
+
+     wc = measurements[1]; hc = measurements[2]
+     bmi = measurements[4]/(measurements[3]/100)^2
+     whr = measurements[1]/measurements[2]
+
+     eta = sum(c(1, bmi, sechol, seglu, bp, age.effects[age.indx,], mRNA[1], mRNA[3],
+                 mRNA[7], mRNA[6], mRNA[9], mRNA[5]) %*% logistic$coefficients)
+
+     pi = 1/(1+exp(-eta))
+     y = rbinom(n = 1, size = 1, prob = pi)
```

```
+
+     if(y == 1){
+       ncase = ncase + 1} else{
+         ncontrol = ncontrol + 1}
+
+     addline = data.frame(y, age, sechol, hdl_setrig[1], bp, hdl_setrig[2], secrea,
+                          seglu, smostat, wc, hc, bmi, whr, mRNA[1], mRNA[2],
+                          mRNA[3], mRNA[4], mRNA[5], mRNA[6], mRNA[7], mRNA[8],
+                          mRNA[9], mRNA[10], mRNA[11], fix.empty.names = FALSE,
+                          row.names=NULL)
+
+     df.sim = rbind(df.sim, addline)
+     if(ncase >= 200 && ncontrol >= 200) break
+   }
+ }
```

This last code chunk simply divides the simulated data set into a training and test set, of equal size, which contains 100 cases and 100 control each.

```
> colnames(df.sim) = c("statcasecontrol", "age", "sechol", "hdlchol", "bp", "setrig",
+                      "secrea", "seglu", "smostat", "wc", "hc", "bmi", "whr",
+                      "hsa_let_7g_5p", "hsa_miR_106a_5p", "hsa_miR_144_3p",
+                      "hsa_miR_151a_5p", "hsa_miR_191_5p", "hsa_miR_21_5p",
+                      "hsa_miR_26a_5p", "hsa_miR_29c_3p", "hsa_miR_424_5p",
+                      "hsa_miR_425_5p", "hsa_miR_451a")
>
> df.train = df.sim[which(df.sim$statcasecontrol==1)[1:100],]
> df.train = rbind(df.train, df.sim[which(df.sim$statcasecontrol==0)[1:100],])
> rownames(df.train) = NULL
>
> df.test = df.sim[which(df.sim$statcasecontrol==1)[101:200],]
> df.test = rbind(df.test, df.sim[which(df.sim$statcasecontrol==0)[101:200],])
> rownames(df.test) = NULL
```

Appendix B

R Codes for Analysis of the HUNT Dataset

These codes show the analysis of the HUNT dataset by the 6 different statistical methods: *pruned tree, logistic GLM, bagged tree, bagged logistic GLM, random forest* and *random GLM*. We start by loading all packages we are going to use.

```
> library(foreign)
> library(tree)
> library(randomForest)
> library(randomGLM)
> library(foreach)
> library(MASS)
> library(ModelMetrics)
> library(pROC)
```

Set an constraint on the factorial (categorical) variables, so that they sum to zero and set a seed so that the results are reproducible.

```
> options(contrasts=c("contr.sum", "contr.sum"))
> set.seed(10)
```

Prepare our data set, which is saved as *Backup global mean og FRM.sav*. `WHR` is our calculated waist-to-hip ratio, `Age_grouped` is a categorical variable which has groupings of ages and `MI` stores the response variables.

```
> ds = read.spss("Backup global mean og FRM.sav", to.data.frame=TRUE)
> WHR = ds$WaistCirc.NT2BLM/ds$HipCirc.NT2BLM
> Age_grouped = cut(ds$PartAg.NT2BLQ1, breaks = c(49, 59, 69, 79, 89))
> MI = (ds$MICases=="Fatale MI" & ds$MICases=="Non-fatale MI")
> cases = which(MI==0)
> MI[cases] = 1; MI[-cases] = 0
> MI = as.factor(MI)
```

Further, we make the smoking status, `SmoStat.NT2BLQ1_2K_TVF` a categorical variable, and reduce the data frame so that it only contains predictors which we will use. Next we remove all subjects with missing smoking status and WHR.

```
> ds$SmoStat.NT2BLQ1_2K_TVF = as.factor(ds$SmoStat.NT2BLQ1_2K_TVF)
> predictors = c("SeChol.NT2BLM", "SmoStat.NT2BLQ1_2K_TVF", "SeHDLChol.NT2BLM",
+                "BPSystMn23.NT2BLM", "WaistCirc.NT2BLM", "HipCirc.NT2BLM",
+                "BMI.NT2BLM", "SeTrig.NT2BLM", "SeCrea.NT2BLM",
+                "SeGluNonFast.NT2BLM")
> ds = data.frame(MI, ds[, predictors], WHR, Age_grouped, ds[,279:289])
> ds = ds[is.na(ds$SmoStat.NT2BLQ1_2K_TVF)==FALSE,]
> ds = ds[is.na(ds$WHR)==FALSE,]
> n = dim(ds)[1]
```

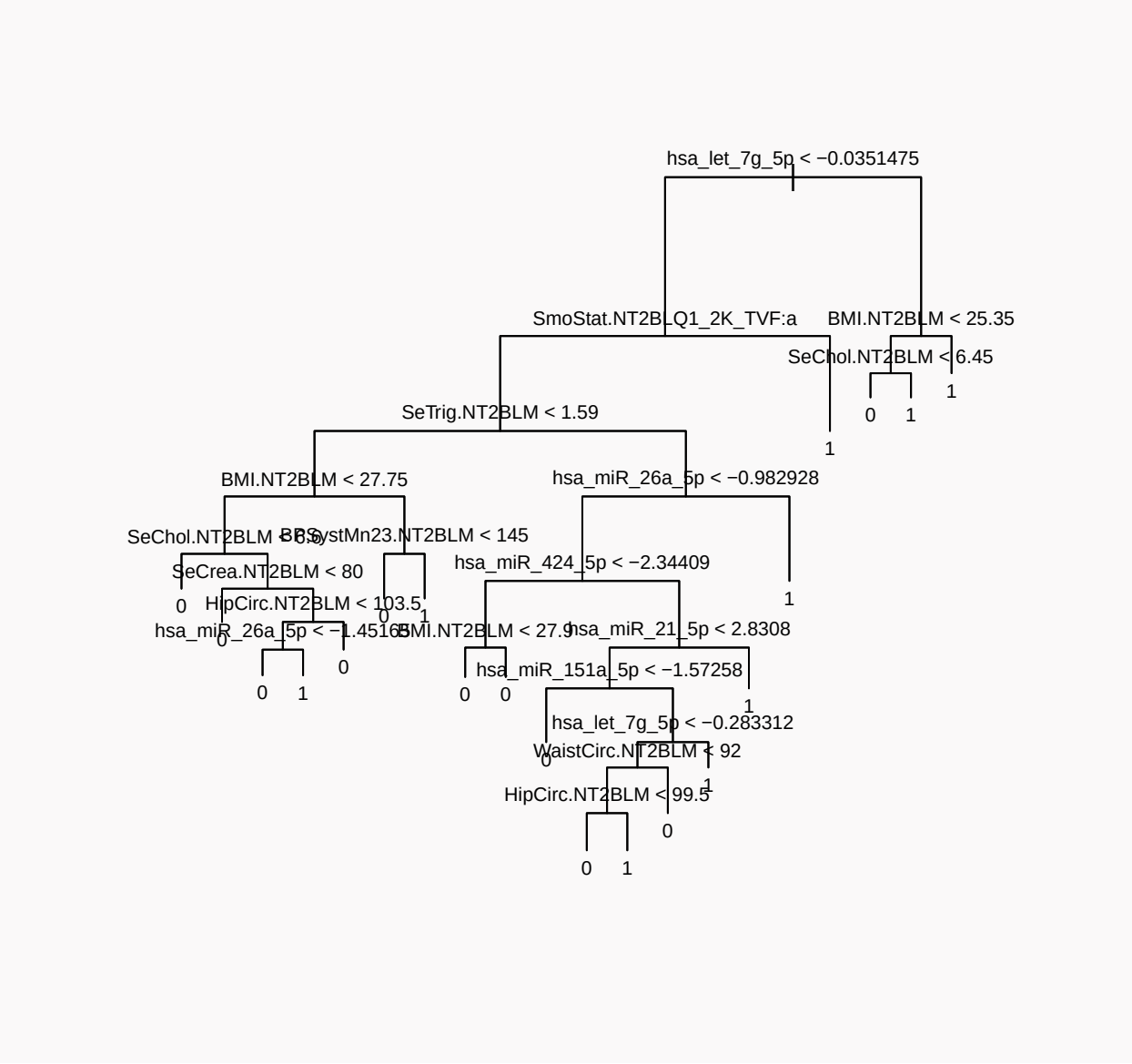
Our HUNT data set is stored in the data frame `ds`. We proceed by fitting our 6 different statistical models. We suppress the outprints here as the results of the data analysis have been presented in Chapter 6.

B.1 Data Analysis

Model 1: Pruned tree

We start by growing a full tree on our data set, and set this equal to `tree.full`. `MI` $\sim$. tells the `tree` function that all predictors are to be included. We can plot the full tree by making a call to the `plot` function, and add text to the branches by calling `text`.

```
> tree.full = tree(MI~., data = ds)
> plot(tree.full)
> text(tree.full, cex = 0.7)
```



To find the optimal tree size, we use the `cv.tree` function, which runs k-fold cross validation to find the deviance of the cost-complexity parameter `k`. We choose the tree size, denoted by `size`, which corresponds to the lowest deviance. Now we prune the tree to obtain the optimal tree size, and make a plot of it.

```
> size = cv.tree(tree.full)$size[which.min(cv.tree(tree.full)$dev)]
> tree.pruned = prune.tree(tree.full, best = size)
> plot(tree.pruned)
> text(tree.pruned, cex=0.7)
```

Model 2: Logistic GLM

To fit an logistic regression model by stepwise forward regression, we start by defining the null and full models. The null model, `logistic.null`, contains only the intercept β_0. The full model, `logistic.full` contains all of the predictors. We then make a call to the `stepAIC` function, and tell it start from the null model, and add predictors, one at a time, by stepwise forward regression, until all predictors from the full model have been added. The model returning the lowest AIC value is our chosen model and is saved as `logistic.best`.

```
> logistic.null = glm(MI ~ 1, data = ds, family = "binomial")
> logistic.full = glm(MI ~ ., data = ds, family = "binomial")
> logistic.best = stepAIC(logistic.null, scope =
+                           list(upper = logistic.full, lower = logistic.null),
+                         direction = "forward", data = ds, trace = FALSE)
```

We can call the `summary` function to get an outprint of our logistic GLM.

```
> summary(logistic.best)
```

Model 3: Bagged Tree

To fit a bagged tree we use the `randomForest` function. We set the parameter `ntree` which is the ensemble size equal to 500. What is important, is that `mtry` must be equal to the number of predictors. Otherwise the model fitted will be a random forest. The parameter `importance=TRUE` needs to be set to create a variable importance plot. One can plot the variable importance plots by calling the function `varImpPlot`.

```
> bag = randomForest(MI~., data = ds, ntree = 500, mtry = 23, importance = TRUE)
> varImpPlot(bag, cex = 0.8, main = "Bagged Tree")
```

Model 4: Bagged GLM

The creation of a bagged GLM has been done using the `randomGLM` function. We specify the predictors by setting `x = x`, where `x` is the model matrix used by the logistic model.

To specify the response variable we set `y = ds[,1]`. We set `nFeaturesInBag` equal to the number of predictors modeled by the model matrix `x`. The parameter `nBags` sets the size of the ensemble. There is no function one can use the make a variable importance plot directly, thus we create one manually.

```
> x =  model.matrix(logistic.full)[,-1]
> bag.logistic = randomGLM(x = x , y = ds[,1], nBags = 100, nFeaturesInBag = 24)
> datvarImp.bag=data.frame(Feature = as.character(dimnames(
+   bag.logistic$timesSelectedByForwardRegression)[[2]]),
+                        timesSelectedByForwardRegression = as.numeric(
+                          bag.logistic$timesSelectedByForwardRegression))
> datVarImpSelected.bag = datvarImp.bag[rank(-datvarImp.bag[,2],
+                                           ties.method="first")<=20,]
> datVarImpSelected.bag = datVarImpSelected.bag[order(datVarImpSelected.bag[,2]),]
> par(mfrow = c(1,1), mar = c(4,8,3,1))
> barplot(datVarImpSelected.bag[,2], horiz = TRUE, names.arg =
+           datVarImpSelected.bag[,1], xlab = "Feature Importance", las = 1,
+         cex = 0.6, main = "Most significant features for the bagged GLM",
+         cex.axis = 1, cex.main = 1.2, cex.lab = 1, col="lightgrey")
```

Model 5: Random Forest

To grow a random forest we use the `randomForest` function. We include a for-loop, to decide on the tuning parameter `mtry`. This tuning parameter tells how many predictors are to be considered in each split when growing one tree, and we choose is based on the minimum OOB error. The parameter `ntree` sets the size of the ensemble. We can plot the variable importance plots by calling `varImpPlot`.

```
> rf.OOBerror = numeric(22)
> for(rf.mtry in 1:22){
+   rf = randomForest(MI~., data = ds, mtry = rf.mtry)
+   rf.OOBerror[rf.mtry] = rf[["err.rate"]][nrow(rf[["err.rate"]]),"OOB"]
+ }
> rf = randomForest(MI~., data = ds, ntree = 500, mtry = which.min(rf.OOBerror),
+                   importance=TRUE)
> varImpPlot(rf, cex = 0.8, main="Random Forest")
```

Model 6: Random GLM

The creation of a random GLM has been done in a similar fashion as the bagged GLM, with the important difference, that the parameter `nFeaturesInBag` must be decided on.

We do this by creating a for-loop. We thereafter choose the value for `nFeaturesInBag` which corresponds to the lowest OOB-error.

```
> RGLM.OOBerror = numeric(22)
> for(RGLM.mtry in 1:22){
+   RGLM = randomGLM(x = x, y = ds[,1], nBags = 100, nFeaturesInBag = RGLM.mtry)
+   RGLM.OOBerror[RGLM.mtry] = sum(RGLM$predictedOOB != ds[,1])
+ }
> randomGLM = randomGLM(x = x, y = ds[,1], nBags = 100,
+                       nFeaturesInBag = which.min(RGLM.OOBerror))
> datvarImp = data.frame(Feature = as.character(dimnames(
+   randomGLM$timesSelectedByForwardRegression)[[2]]),
+                       timesSelectedByForwardRegression = as.numeric(
+                         randomGLM$timesSelectedByForwardRegression))
> datVarImpSelected = datvarImp[rank(-datvarImp[,2], ties.method="first") <= 20, ]
> datVarImpSelected = datVarImpSelected[order(datVarImpSelected[,2]),]
> par(mfrow = c(1,1), mar = c(4,8,3,1))
> barplot(datVarImpSelected[,2], horiz = TRUE, names.arg = datVarImpSelected[,1],
+         xlab = "Feature Importance", las = 1, cex = 0.6,
+         main = "Most significant features for the RGLM",
+         cex.axis = 1, cex.main = 1.2, cex.lab = 1, col = "lightgrey")
```

B.2 Testing the Predictive Power of Our Statistical Models

The following codes show how have tested the predictive power of our statistical models, when applied to the HUNT data set. We have randomly partitioned the data set into a training and test set, where the training set consisted of about $\frac{2}{3}$ of the observations and the test set the remaining $\frac{1}{3}$. Models have been fitted to the training sets. The fitted models have been thereafter used to make predictions for the test sets. The accuracy of the predictions has been evaluated by calculating the Brier and AUC scores. The training error was also recorded. This has been repeated 100 times.

```
> K = 100
> BS = matrix(NA, ncol=6, nrow=K)
> AUC = matrix(NA, ncol=6, nrow=K)
> TRAIN.error = matrix(NA, ncol=6, nrow=K)
> for(k in 1:K){
+   set.seed(k+6)
+
```

```
+   test = sample(seq(1,197,1), 66, replace=FALSE)
+
+   ds.test = ds[test, ]
+   ds.train = ds[-test,]
+   y.test = as.numeric(as.character(ds.test$MI))
+   y.train = as.numeric(as.character(ds.train$MI))
+
+   #### PRUNED TREE ####
+   tree.train = tree(MI~., data=ds.train)
+   size.train = cv.tree(tree.train)$size[which.min(cv.tree(tree.train)$dev)]
+   while(size.train == 1){
+     size.train = cv.tree(tree.train)$size[which.min(cv.tree(tree.train)$dev)]
+   }
+
+   tree.train.pruned = prune.tree(tree.train, best = size.train)
+
+   tree.pred = predict(tree.train.pruned, newdata=ds.test)[,2]
+
+   tree.bs = brier(y.test, tree.pred)
+   tree.auc = auc(y.test, tree.pred)
+
+   tree.train.error = mean(ds.train[,1] != predict(tree.train.pruned, type="class"))
+
+   #### LOGISTIC ####
+   logistic.train.null = glm(MI~1, data=ds.train, family="binomial")
+   logistic.train.full = glm(MI~., data=ds.train, family="binomial")
+
+   logistic.train.best = stepAIC(logistic.train.null, scope =
+                           list(upper = logistic.train.full,
+                                lower = logistic.train.null),
+                           direction="forward", data=ds.train, trace=FALSE)
+
+   logistic.pred = predict(logistic.train.best, newdata=ds.test, type="response")
+
+   logistic.bs = brier(y.test, logistic.pred)
+   logistic.auc = auc(y.test, logistic.pred)
+
+   logistic.train.error = mean(ds.train[,1] != round(
+     logistic.train.best$fitted.values))
+
+   #### BAGGED LOGISTIC ####
+   x.train =  model.matrix(logistic.train.full)[,-1]
+
+   logistic.test.full = glm(MI~., data=ds.test, family="binomial")
+   x.test =  model.matrix(logistic.test.full)[,-1]
```

```
+
+   bag.logistic.train = randomGLM(x = x.train , y = ds.train[,1], nBags = 100,
+                                  nFeaturesInBag = 24)
+
+
+   bag.logistic.pred = predict(bag.logistic.train, newdata=x.test)[,2]
+
+   bag.logistic.bs = brier(y.test, bag.logistic.pred)
+   bag.logistic.auc = auc(y.test, bag.logistic.pred)
+
+   bag.logistic.train.error = mean(ds.train[,1] != bag.logistic.train$predictedOOB)
+
+   #### BAGGED TREE ####
+   bag.train = randomForest(MI~., data=ds.train, ntree = 500, mtry=23)
+   bag.pred = predict(bag.train, newdata=ds.test, type = "prob")[,2]
+
+   bag.bs = brier(y.test, bag.pred)
+   bag.auc = auc(y.test, bag.pred)
+
+   bag.train.error = mean(ds.train[,1] != predict(bag.train, type="class"))
+
+   #### RANDOM FOREST ####
+   rf.OOBerror = numeric(22)
+   for(rf.mtry in 1:22){
+     rf = randomForest(MI~., data=ds.train, mtry=rf.mtry)
+     rf.OOBerror[rf.mtry] = rf[["err.rate"]][nrow(rf[["err.rate"]]),"OOB"]
+   }
+   rf.train = randomForest(MI~., data=ds.train, ntree =500,
+                           mtry = which.min(rf.OOBerror))
+   rf.pred = predict(rf.train, newdata=ds.test, type = "prob")[,2]
+
+   rf.bs = brier(y.test, rf.pred)
+   rf.auc = auc(y.test, rf.pred)
+
+   rf.train.error = mean(ds.train[,1] != predict(rf.train, type="class"))
+
+   #### RANDOM GLM ####
+   x.train =  model.matrix(logistic.train.full)[,-1]
+
+   logistic.test.full = glm(MI~., data=ds.test, family="binomial")
+   x.test =  model.matrix(logistic.test.full)[,-1]
+
+   rGLM15 = randomGLM(x = x.train, y = ds.train[,1], nBags = 100,
+                      nFeaturesInBag = 15)
+   rGLM15.misclas = sum(rGLM15$predictedOOB != ds.train[,1])
```

```
+
+   rGLM20 = randomGLM(x = x.train, y = ds.train[,1], nBags = 100,
+                      nFeaturesInBag = 20)
+   rGLM20.misclas = sum(rGLM20$predictedOOB != ds.train[,1])
+
+   rGLM.OOBerror = c(rGLM15.misclas, rGLM20.misclas)
+   rGLM.misclas.min = which.min(rGLM.OOBerror)
+
+   if(rGLM.misclas.min == 1){
+     randomGLM.train = rGLM15
+   }else
+   {
+     randomGLM.train = rGLM20
+   }
+
+   randomGLM.pred = predict(randomGLM.train, newdata=x.test)[,2]
+
+   randomGLM.bs = brier(y.test, randomGLM.pred)
+   randomGLM.auc = auc(y.test, randomGLM.pred)
+
+   randomGLM.train.error = mean(ds.train[,1] != randomGLM.train$predictedOOB)
+
+
+   BS[k,] = c(tree.bs, logistic.bs, bag.bs, bag.logistic.bs, rf.bs, randomGLM.bs)
+   AUC[k,] = c(tree.auc, logistic.auc, bag.auc, bag.logistic.auc,
+               rf.auc, randomGLM.auc)
+   TRAIN.error[k, ] = c(tree.train.error, logistic.train.error, bag.train.error,
+                        bag.logistic.train.error, rf.train.error,
+                        randomGLM.train.error)
+ }
```