Classification Trees for High-Dimensional Highly-Correlated Data

Dodi Vionanda

Department of Statistics

University of Leeds

A thesis submitted for the degree of

Doctor of Philosophy

January 2020

Acknowledgements

I firstly would like to thank my wife, my children, my parents, my parents-in- law, my brother and my sister for their love and support during this PhD journey.

I thank my supervisors Dr. Arief Gusnanto, Dr. Jochen Voss and Professor Charles Taylor for all their help and patience while supervising me.

Throughout my study, I was funded by Ministry of Research and Higher Education of Republic of Indonesia.

Abstract

This thesis focuses on building classification model to classify tumour subtypes of lung cancer using CNA estimates datasets. As a genomic datasets, these CNA estimates datasets have much more variables than observations and blocks of correlated variables. There are also several variables of high variance. In terms of data classification, these datasets have many variables as strong relevant predictors and larger number of variables as weak relevant predictors.

Applying Classification Trees on such this dataset we have to cope with three issues. Firstly, fitting Classification Trees using datasets of much more variables than observations, we only utilize the discriminatory information contained in small number of variables which can be seen as strong relevant predictors and leave large number of variables including the informative ones. Whereas, some of these variables are important for prediction.

Secondly, having variables which contain the discriminative information with high data variation yields less accurate Classification Trees. Finally, in the light of cross validated estimation for the prediction error, we find that having datasets of small sample size causes the unstable error rate estimates.

To overcome these issues we make use of PCA as a data dimensionality reduction method. We apply PCA prior to Classification Trees construction to reduce data dimensionality while retain the variation of the data. However, applying PCA does not improve Classification Trees performance in terms of the prediction error rate. PCA produces new variables by finding linear combinations of original datasets with maximum variance. Hence, the resulting principal component scores

might be more affected by variables of extremely high variances. In addition, PCA is also strongly affected by the presence of many blocks of correlated variables.

Furthermore, we apply ICA as a feature extraction method prior to Classification Trees construction. However, applying this feature extraction method does not improve the resulting Classification Trees as well. There are two reasons for this. Firstly, the whitening step applied prior to maximising non-Gaussianity during the estimation of independent components greatly reduces data dimension as PCA does. Therefore, we end up with nearly similar results as applying PCA. Secondly, ICA estimates involve maximising non-Gaussianity on the whitened variables. This maximisation only deals with non-Gaussianity and might not exploit the discriminative information contained in dataset.

Finally, we apply Random Forests to overcome the issues mentioned above. Random Forests involve subsetting of variables for splitting each node in an individual tree. This in turn enables weak relevant predictors to be selected as the classifying ones. Apart from the number of variables to be picked for splitting each node in an individual tree, there are other two tuning parameters: number of trees to be generated and number of observations in each terminal node of an individual trees.

We should search for the optimal setting of these hyperparameters for the forest to produce Random Forest with low error rate. For the number of observations in a terminal node we recommend the use of one observation. For the number of trees, based on our simulation studies, we find that one should generate at least 500 trees in order to obtain a stable estimate. However, we cannot prescribe one suggestion for the number of variables to be selected. For our datasets, we find that for Smooth CNA dataset we have to set this hyperparameter small. On contrast, for DNACopy CNA dataset, this tuning

parameter should be set to become large for getting Random Forests with small error rate.

With regard to the comparison across the models explained above, we recommend the use of Random Forests rather than Classification Trees. We also do not suggest the application of both PCA and ICA as the data dimension reduction methods. In terms of prediction error, Random Forests give the lowest error rate. In the light of the insight underlying the resulting classification model, Random Forests are able to precisely produce the accurate classifiers. For DNACopy dataset, Random Forests successfully identify the contribution of variables within both Chromosomes 3 and 10. Whereas Classification Trees only recognise the contribution of those of Chromosome 10. However, in terms of computational time, Random Forests are expensive. Compared to the other models, Random Forests spend the longest time.

Furthermore, with respect to the genetic point of view for the resulting Classification Trees and Random Forests for both Smooth CNA and DNACopy CNA datasets, we end up with different results. For Classification Trees of Smooth CNA dataset, we obtain genes SOX2 and PIK3CA as the genetic markers. Meanwhile, for Classification Trees of DNACopy CNA dataset, we attains gene KIF5B and RET as the genetic markers. Moreover, for Random Forests of Smooth CNA datasets, we get the same result as of Classification Trees. Nevertheless, for Random Forests of DNACopy CNA dataset, we end up with genes KIF5B, RET, SOX2 and PIK3CA as the genetic markers. The amplification of genes SOX2 and PIK3CA within loci 3q24 up to 3q27.3 is common in squamous carcinoma lung cancer. The fusion between gene KIF5B in locus 10p11.22 and gene RET in locus 10q11.21 is common in adeno carcinoma lung cancer.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Background

Copy Number Alterations (CNA) are structural variations in the human genome, in which some regions exhibit more or less copies than the normal two copies. These structural variations are common in cancer patients. CNA profile of a cancer patient can be generated using sequencing dataset. The recent advance of high-throughput sequencing has been providing us with the whole genome sequence data for low cost. This whole genome sequence data might be used to produce CNA profile of a cancer patient since it provides base-pair resolution of an entire cancer genome.

Certain type of cancer has multiple subtypes. Using CNA profiles generated by next-generation sequncing (NGS) dataset, one might find a determinant for tumour subtyping of this cancer. This approach leads to better-adjusted treatments for each patient. However, the stratification to identify these subtypes is still a challenging work.

We have two datasets contain CNA profiles of 76 patients who suffered from lung cancer with two subtypes: adeno carcinoma and squamos carcinoma. Using these datasets to build a classification model will help us to identify the subtype of cancer for a patient with an unknown subtype. The first dataset is the one that obtained by involving Robust Smooth Segmentation (RSS) method of (Huang *et al.*, 2007) as the segmentation method during the estimation of CNA profiles from NGS dataset. Meanwhile, the second dataset is the one attained by involving

Circular Binary Segmentation (CBS) algorithm of (Olshen *et al.*, 2004) as the segmentation method.

These CNA datasets had been discussed in Gusnanto *et al.* (2015). Gusnanto *et al.* (2015) applied Logistic Regression on these two datasets to stratify tumour subtypes. In their work, Gusnanto *et al.* (2015) focused more on the classification error rate rather than the insight obtained. Meanwhile, in this thesis, we carry out data classification on these two datasets utilizing Classification Trees and Random Forests.

With regard to the comparison of Logistic Regression and Classification Trees as the classification models for genomic datasets, several works have been carried out. To name a few, Schwarzer *et al.* (2003), Muller & Möckel (2008) and Yoo *et al.* (2012) found that both Logistic Regression and Classification Trees gave similar relevant predictors and cut-offs.

Moreover, Breiman (2001b) proposed two cultures in statistical modelling: the data modelling culture and the algorithmic modelling culture. Breiman (2001b) classified Logistic Regression as part of data modelling culture and Classsification Trees and Random Forests as parts of the algorithmic modelling culture. In this point of view, the application of Logistic Regression requires several assumptions to be fulfilled. Nevertheless, the utilization of either Classification Trees or Random Forests does not requires any assumption. Hence, Breiman (2001b) argued that applying of the data modelling culture prevents statisticians from working on new exciting problem.

Besides, in the light of interpretability, for datasets which have large number of variables, one might find that Classification Trees are easier to interpret than Logistic Regression model. In addition, using Classification Trees, one might also find that it is more easier for the trees to identify the interaction between variables in datasets than Logistic Regression.

Furthermore, there are several works have been carried out on the application of Classification Trees and Random Forests on genomic datasets. To name a few, both Zhang *et al.* (2001) and Allory *et al.* (2008) propose the use of Classification Trees for tumour stratification using genomic data. Meanwhile, Zhang *et al.* (2003), Díaz-Uriarte & De Andres (2006), Statnikov *et al.* (2008), Goldstein *et al.*

(2010), Chen & Ishwaran (2012) and Botta *et al.* (2014) put forward the application of Random Forests on genomic datasets. However, among these works, no one has applied either Classification Trees or Random Forests for CNA estimate datasets.

1.2 Objectives

There are many problems arise from constructing a classification model on a genomic dataset such as our CNA estimates one. We address three main problems in this thesis.

1.2.1 Data Characteristic

Working with a genomic datasets, we have to cope with the issues arise from the characteristics of such dataset. Firstly, the genomic dataset consists of much more variables than observations. Secondly, this dataset might have many variables whose extremely high variance. Thirdly, this dataset has many blocks of correlated variables since those of variables share the common genomic location are highly correlated. Finally, in terms of classification task, such dataset does not only have many variables as strong relevant predictors, but also larger number of variables as weak relevant predictors. These four characteristics make data classification for a genomic dataset becomes a challenging work.

1.2.2 Constructing of Classification Rules using Classification Trees

Applying Classification Trees on a genomic dataset, one needs to address several issues emerge from the characteristics of such dataset. The first issue arises from the dimension of the dataset. Fitting a Classification Tree on a dataset which has much more variables than observations, one needs to manage the availability of huge amount of information, whilst one is only able to pick very little from such this huge amount of information while fitting a Classification Tree. Hence,

we need to find either the appropriate variable selection or the appropriate feature extraction methods prior to Classification Trees construction as dimension reduction approaches.

Furthermore, we need to investigate how the presence of blocks of correlated variables in the dataset affects the performance of the resulting Classification Trees. In addition, we also have to cope with the issue comes up from having many relevant predictors whose extremely high variance.

1.2.3 Constructing of Classification Rules using Random Forests

Due to fact that it is small number of variables which are involved in building Classification Trees, we apply Random Forests as a classification rule. We also apply Random Forests to avoid instability issue in estimating prediction error rate of Classification Trees.

However, applying Random Forests, we need to find optimal setting of hyperparameters to obtain the forests with low error rate. There are three tuning parameters of Random Forests: number of trees to be generated to build a forest ($ntree$), number of observations in each terminal node of an individual tree ($nodesize$) and number of variables to be subsetted for splitting each node in an individual tree ($mtry$). Applying Random Forests on our datasets, we have to perform exploration study to find the setting for these tuning parameters. We can not use the default setting for these tuning parameters as we work with datasets whose tens of thousands of variables.

1.3 Thesis Overview

Throughout this thesis we attempt to find a classification rule whose high accuracy and to identify the genomic regions that are responsible for the resulting classification rule. This work was begun by presenting the description of real datasets and the simulation setting for data generating throughout our study in Chapter 2.

As previously mentioned, there are four main characteristics of the datasets that we described in this chapter. First, our datasets have much more variables than observations. Second, they contain correlated blocks of variables due to the fact that variables within the same genomic location share many features in common. Third, there are many variables in our datasets that have extremely high variance. Fourth, in terms of classification task, our datasets have many variables as strong relevant predictors along with large number of variables as weak relevant predictors. These two kinds of variables are crucial for classification task. The simulated datasets are also generated using the setting that takes those description above into consideration.

In Chapter 3, we present theoretical explanation for Classification Trees. Our work on Classification Trees is mostly based on Classification Trees of Breiman *et al.* (1984). In this chapter, we also explain various estimation methods of prediction error rate to assess the performance of the resulting Classification Trees. Among all methods presented, the test set error method is used to estimate the prediction error rate for simulated datasets, while the repeated 10-fold stratified cross validation estimation is utilized for real datasets.

In Chapter 4, we apply Classification Trees on real and simulated datasets. The application of Classification Trees on real datasets intends to stratifying tumour subtypes using CNA estimates datasets. There are two aims for this task: obtaining an accurate prediction model and getting better insight on the underlying structure of the resulting Classification Trees.

For the first purpose, initially, the trees are build using the raw dataset. However, the resulting Classification Trees involve too few variables such that large number of variables including the informative ones are left unused. To improve this result, variable selection step has been applied prior to the trees construction. This variable selection step was performed by taking the information about genomic regions whose discriminative information into account. In addition, simple data projection methods have been performed as well. There are two methods have been used: mean-projection and median-projection. Yet, either variable selection or simple data projection methods do not improve the performance of the resulting Classification Trees in the light of cross-validated error rates.

Moreover, for the second purpose, we interpret the resulting Classification Trees by looking at the classifying variables along with their cut-offs. For both Smooth CNA and DNACopy CNA datasets, we end up with variables within Chromosome 3 as the ones which are responsible for the tumour stratifying. In addition, for DNACopy CNA, we find that there are also several variables within Chromosome 10 whose discriminative information.

Furthermore, we perform simulation study using the hypothetical datasets to get deeper insight on the results obtained using real datasets. This simulation study takes various settings of mean difference between group of observations, standard deviation of individual variables and correlation into consideration.

In Chapter 5, we apply PCA as a feature extraction method prior to Classification Trees construction for both real and simulated datasets. PCA has been used as a data dimension reduction method since it produces smaller number of new variables while maintains data variation. Applying PCA for this purpose, one has to find the optimal number of components to be used for fitting the trees.

For real datasets, we find that the presence of variables of high variance affects the performance of PCA as a projection method prior to Classification Trees construction. Hence, we apply data standardising and data transformation to improve the performance of PCA for this classification task. Yet, applying PCA on either the raw, the standardised or the transformed datasets does not improve the performance of the resulting components on the data classification task.

Moreover, for simulated datasets, we perform simulation study by fitting the trees using the training sets and assessing the trees performance using the test sets. This simulation study shows that apart from variance of each variable, the presence of correlation also diminishes the performance of PCA in the light of classification task.

In Chapter 6, we apply ICA as a feature extraction method prior to Classification Trees construction too. We apply this method on both real and simulated datasets. We implement FastICA of Hyvärinen & Oja (1997) as an algorithm and fastICA package of Marchini *et al.* (2019) as a computational tool. For both real and simulated datasets, we find that the results obtained from applying ICA share many things in common with those of PCA. We obtain this result as we have to

whiten variables prior to maximising the non-Gausianity for estimating the independent components. This whitening step involves Singularvalue decomposition which relates to Eigenvalue decomposition used in PCA.

In Chapter 7, we discuss the results attained from applying of Random Forest on real datasets. We make use of Random Forests of Breiman (2001a) and implement **randomForest** of Liaw *et al.* (2002) for computational purpose . For the application of Random Forests, we have to perform the exploration study to find the hyperparameters that produce the optimal Random Forests in the light of prediction error. These hyperparameter settings then are used to fit an optimal Random Forest. By looking in more detail at the resulting Random Forests, one might study the underlying structures of the resulting classifier. The summary of this thesis is given in Chapter 8.

Chapter 2

Data

2.1 Introduction

In this chapter, we present the description of datasets that we are going to discuss in chapters that follow. In general, we have two kinds of datasets: real and simulated datasets. For the real datasets, we have two datasets on Copy Number Alteration (CNA) profiles of 76 patients suffered from lung cancer with two subtypes: adeno carcinoma and squamos carcino. Among these 76 patients, there are 38 patiets who suffered from adeno carcinoma lung cancer, while the rest underwent squamos carcino lung cancer.

The first CNA profile dataset consists of the CNA estimates obtained by by involving Robust Smooth Segmentation (RSS) method of (Huang *et al.*, 2007) as the segmentation method during the estimation of CNA profiles from NGS dataset. Meanwhile, the second CNA profile dataset consists of CNA estimates yielded by involving Circular Binary Segmentation (CBS) algorithm of (Olshen *et al.*, 2004) as the segmentation method. For simplicity, we name the first one as Smooth CNA dataset, while the second one as DNACopy CNA dataset. Using these two datasets, we are going to build classification models to stratify the two lung cancer subtypes mentioned earlier.

Like many other genomic datasets, these datasets have much more variables than observations. They also have blocks of correlated structure due to the fact that variables within the same genomic locations are correlated.

Moreover, for the simulated datasets, we generate datasets from a population which has a distribution as a mixture of two multivariate normal distributions since we are dealing with real datasets whose two groups of observations. We classify the samples drawn from the subpopulation with the first distribution as the samples of Class 1 and the ones picked from the subpopulation with the second distribution as the samples of Class 2. Our simulated datasets also have more variables than observations. However, due to the computational complexity, we only generate the ones whose slightly more variables than observations. In addition, the generated datasets have correlated structure as the real ones do.

2.2 Copy Number Alterations of Next-Generation Sequence Data

The following is a brief description on how our two CNA profile datasets are obtained. CNA profile of lung cancer patient datasets analysed in this research are generated from NGS dataset. The details on biological sample preparation to get this NGS dataset can be found in Wood *et al.* (2010). The resulting DNA samples are sequenced using Illumina GAII to produce short DNA fragments which are called reads. These small fragments are mapped to a normal human genome and split into windows.

However, since we are dealing with high-throughput sequencing data, we have to find an optimal size for these windows. This optimal size is required to assure that each window does not contain too small or too large number of reads. To deal with this problem, we use NGOptwin package of Gusnanto *et al.* (2014) and end up with the window of size 150 kbp. We are going to represent each window into a single explanatory variable.

The reads in each single window of cancer cell are then compared to the reads of the corresponding normal cell to get the initial estimates of the copy number alteration (CNA) profiles. Yet, these initial estimates contain random error. To address this issue, we apply two segmentation methods: Smooth Segmentation of Huang *et al.* (2007) and Circular Binary Segmentation of Olshen *et al.* (2004).

Later on, we call CNA profile estimate dataset attained from the first segmentation method mentioned as Smooth CNA dataset, while the dataset obtained from the second method as DNAcopy CNA dataset.

Furthermore, we apply normalisation using CNAnorm package of Gusnanto *et al.* (2012) on the smoothed ratios obtained from either one of the two segmentation methods. This normalisation is aimed to cope with difference in genome size and number of reads. We also employ this CNAnorm package to do scaling which is required to manage the contamination of the tumour samples by the normal cell.

The estimates of CNA profiles as in DNACopy CNA dataset have the main characteristic that the segmented line tends to form relatively long constant segments. The top two panels of Figure 2.1 show these estimates. Meanwhile, those in Smooth CNA dataset have the main characteristic that the segmented line is smooth and follows the sudden "jumps" and "drops" in the CNA profile. The bottom panel of Figure 2.1 shows these estimates.

2.3 Genomic Data

In this section we discuss the properties of our real datasets in more detail. We begin with the explanation on the preprocessing steps applied to cope with variables whose missing values and we continue by the discussion on the description about these datasets in the light of the discriminative information they contain.

2.3.1 Data Description

As previously mentioned, we have two real datasets of CNA estimates for 76 patients who suffered from lung cancer: Smooth CNA and DNACopy CNA datasets. These datasets had been discussed in Gusnanto *et al.* (2015). Gusnanto *et al.* (2015) applied logistic regression on these two datasets to stratify tumour subtypes and to find the genomic contribution on this stratification. Meanwhile, in this thesis, we carry out data classification on these two datasets utilizing Classification Trees and Random Forests.

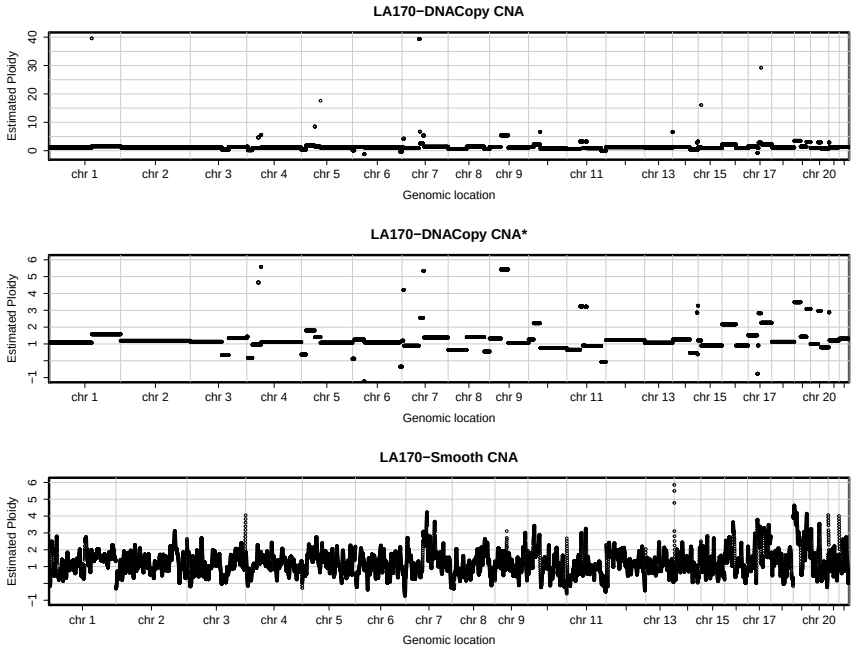


Figure 2.1: Example of CNA estimates from one patient, LA170, who suffered from adeno carcinoma lung cancer. The horizontal axis corresponds to genome regions and the points are ordered according to the human reference genome. Each point corresponds to one genomic window of size 150 kb. The top panel corresponds to DNACopy CNA estimates, the middle panel corresponds to DNACopy CNA estimates after zooming in to see the variability of CNA across genome and the bottom panel corresponds to the Smooth CNA estimate.

Both Smooth CNA and DNACopy CNA datasets we are working with in this chapter consist of 76 observations and 19,220 variables. Among these variables, 19219 of them are explanatory variables. Each of these explanatory variables represents a window of the size 150 kbp. Meanwhile, the other variable is a categorical response variable which has two categories and the rest variables are the explanatory ones. This categorical response variable contains information which explains that, among 76 observations in our datasets, there are 38 patients who suffered from adeno carcinoma lung cancer and the other 38 suffered from squamos carcinoma lung cancer. We denote a patient with adeno carcinoma as an observation from Class 1 and one with squamos carcinoma as an observation from Class 2. As we are not dealing with sex chromosomes, so we omit the variables that come from X and Y chomosomes from the datasets. Hence, we are only working with 22 chromosomes.

Before further analysis of these two datasets, we remove the variables that have more than one missing values and impute the missing value for the variables that have only one missing value. Thus, we end up with the datasets which have the number of variables as in Table 2.1. In addition, from Table 2.1, we can see that both Smooth CNA and DNACopy CNA datasets that we are working with have much more variables than observations.

Table 2.1: Number of variables of Smooth CNA and DNACopy CNA datasets after preprocessing step

Dataset	Number of observations	Number of variables
Smooth CNA	76	17,571
DNACopy CNA	76	18,388

2.3.2 Smooth CNA Dataset

The following subsection of this chapter moves on to describe Smooth CNA dataset. Initially, we present the line plot for the mean of each variable within this dataset along with the line plot for each observation in Figure 2.2. Next, we

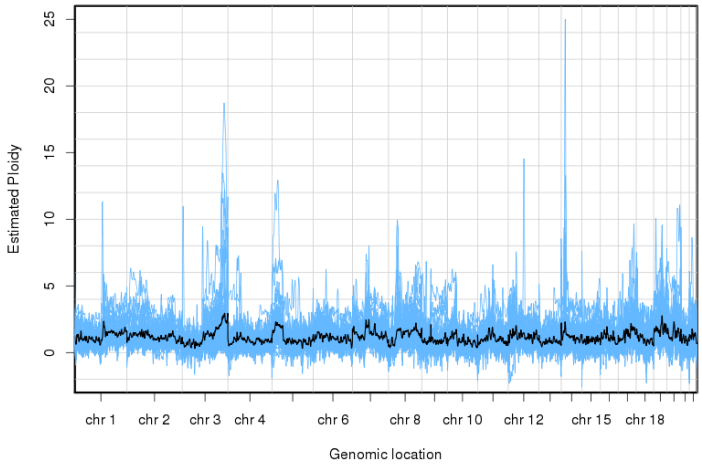


Figure 2.2: Plot for mean of each variable (gene segment) along with line plot for each observation of Smooth CNA dataset. The black solid line presents the plot of mean. Each point in this black line corresponds to the mean of each variable. The blue region nearby this black solid line is the overlapping line plot for each observation. This blue region consists of 76 overlapping line plots.

display line plots for the variance and skewness of each variable in Figures 2.3 and 2.4, respectively.

Looking at Figure 2.2, it is apparent that this dataset contains many variables which have extreme observations. Some of these variables can be found in Chromosomes 3 and 14. The presence of these extreme observations then have contributed to the high variability within these variables as can be seen in Figure 2.3. These extreme observations are also responsible for many variables being skewed to the right as shown in Figure 2.4.

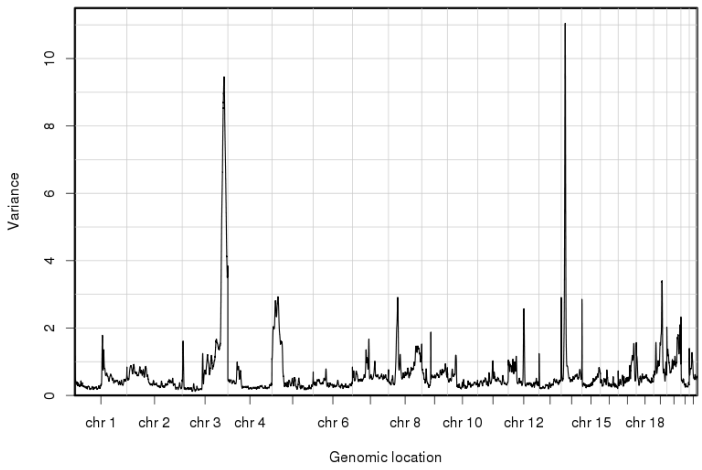


Figure 2.3: Line plot for variance of each variable for Smooth CNA dataset. Each point in this line plot corresponds to the variance of a single variable.

Moreover, Figure 2.5 presents the p-values obtained from performing normality test on each individual variable of Smooth CNA dataset. We use **shapiro.test** function for this purpose. Panel (a) of this figure indicates the presence of several variables which are normally distributed. However, vast majority of variables do not follow normal distribution. The histogram in panel (b) of this figure shows that more than 80% of variables yield p-values of normality test less than 0.05. This indicates that those variables are not normally distributed. With regard to the relation between normality and skewness of individual variable for this dataset, we find that all right-skewed variables are not normally distributed. In

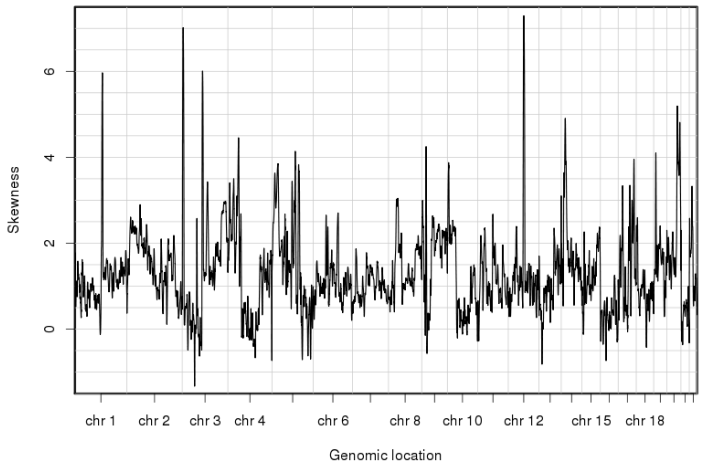


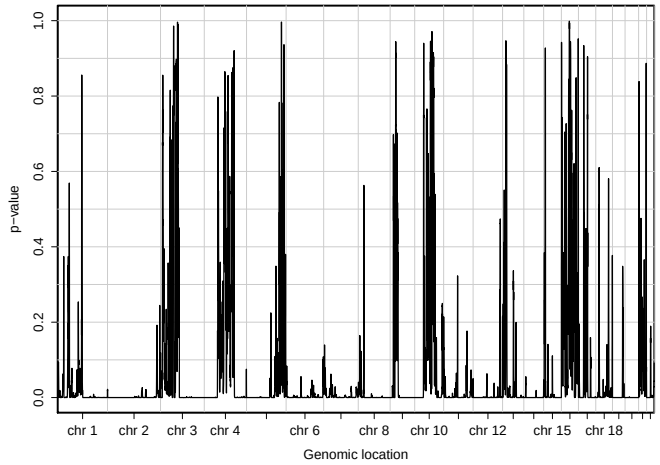
Figure 2.4: Line plot for the skewness of each variable for Smooth CNA dataset. Each point in this line plot corresponds to the skewness of a single variable.

addition, we also get large number of variables whose nearly zero skewness which are not normally distributed.

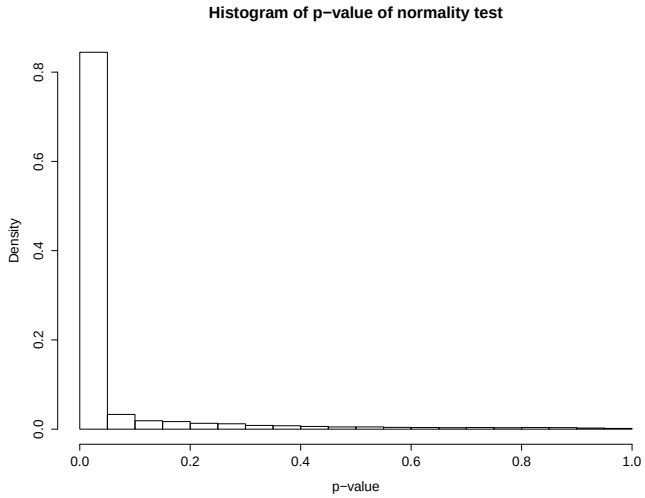
Furthermore, plots in Figures 2.2, 2.3, 2.4 and 2.5 show that variables which share the same genomic locations also share the same characteristics. We obtain this result because in a genomic dataset, features which have similar genomic location are highly correlated. In contrast, there is only low correlation between variables across distinct genomic locations as they have different characteristics. This in turn leads to the presence of several blocks of correlated variables in our dataset. Figure 2.6 indicates the presence of such these blocks of variables using the bright shaded areas in the image of correlation matrix produced.

In addition, we display the image of correlation matrix for variables of genomic region of Chromosome 3 as in Figure 2.7. We present the result for variables within this genomic region since most of them are relevant predictors as we will discuss later on. This image of correlation matrix indicates the presence of two blocks of correlated variables in the genomic region of Chromosome 3.

Having presented the general description of Smooth CNA dataset, let us move



(a)



(b)

Figure 2.5: Plot and histogram of p-values obtained from Normality Test for Smooth CNA dataset. Panel (a) presents the line plot for p-values of Normality Test performed on each variable of Smooth CNA dataset. Each dot in this line plot corresponds to the p-value of Normality Test upon single variable. Panel (b) shows the histogram of these resulting p-values.

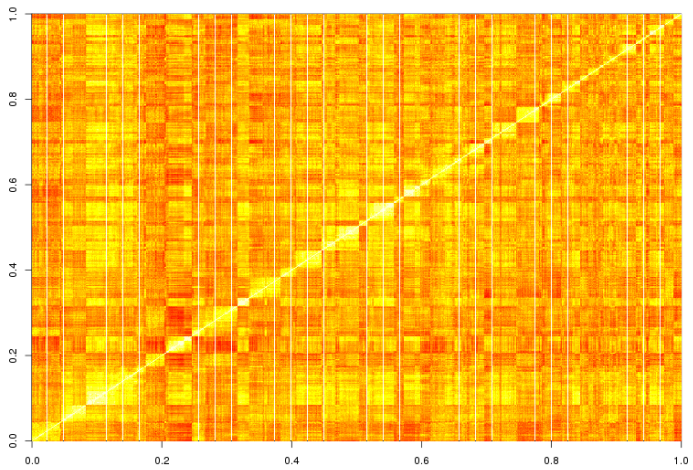


Figure 2.6: Image of correlation matrix for the whole genome of Smooth CNA dataset. Both the vertical and horizontal axes correspond to the whole genomic region.

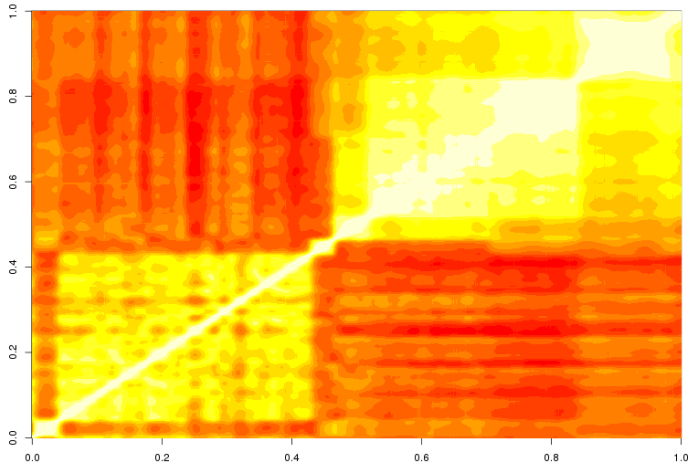
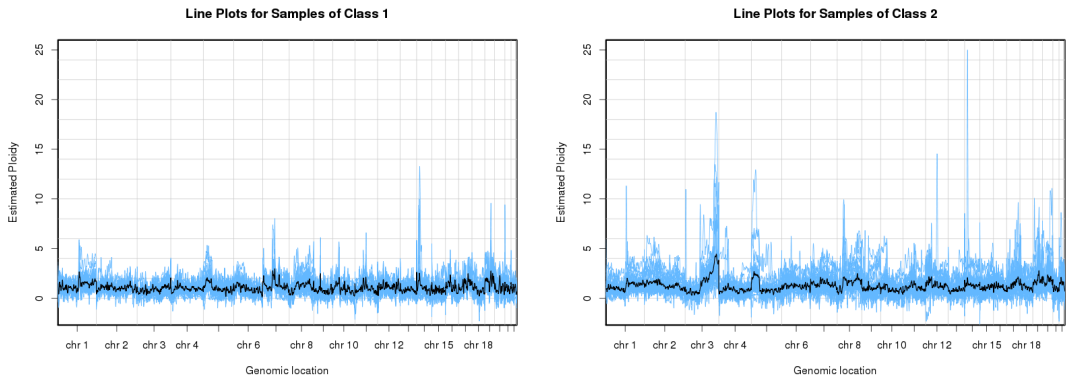


Figure 2.7: Image of correlation matrix for variables within genomic region of Chromosome 3 for Smooth CNA dataset. Both the vertical and horizontal axes correspond to variables within Chromosome 3.

now to describe Smooth CNA dataset in the light of data classification task that we are going to perform in chapters that follow. We begin to view Smooth CNA dataset as a dataset that consists of two groups of observations: Class 1 and Class 2. The observations of Class 1 are the patients who suffered from adeno carcinoma lung cancer, while those of Class 2 are the patients who underwent squamos carcinoma lung cancer.

To begin with, we present line plots for all samples from each group of observations along with their group mean in Figure 2.8 to show the main difference in characteristics between the observations of Class 1 and Class 2. The black solid lines within two panels of this figure display the plots for means of each group of observations, whereas the blue lines show the line plots for an individual observation of each group.

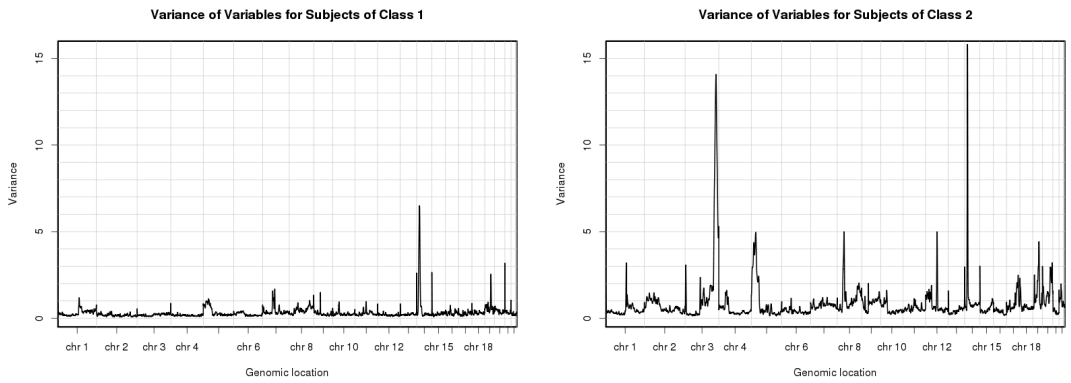


(a) (b)

Figure 2.8: Line plot for the group mean of each variable along with line plot for each observation of the corresponding group for smooth CNA dataset. Panel (a) shows the plots for the observations of Class 1, whereas Panel (b) presents the plots for the observations of Class 2. The black solid line presents the plot of group mean and the blue region nearby this black solid line is the overlapping line plot for each observation within the corresponding group.

It is apparent from Figure 2.8 that observations of Class 2 have higher data variation than the ones of Class 1. These observations also have several extreme

measurements. One can easily see such these measurements within genomic locations of Chromosome 3 and Chromosome 14. As a result, the samples of Class 2 give higher variance than those of Class 1 as can be seen in Figure 2.9.



(a) (b)

Figure 2.9: Plots of within-group variance of samples from Group 1 and Group 2 of Smooth CNA dataset. Panel (a) presents line plot of within-group variance of samples from Group 1. Each point in this plot corresponds with the variance of an individual variable for the observations of Class 1. Panel (b) shows plot of within-group variance of samples from Group 2. Each point in this plot corresponds with the variance of an individual variable for the observations of Class 2.

Panel (a) of Figure 2.9 shows that the presence of one block of variables within genomic location Chromosome 14 whose much higher variance than variables the remaining genomic locations for samples of Class 1. Panel (b) of this figure indicates the presence of two blocks of variables within genomic locations of Chromosome 3 and Chromosome 14 whose much higher variance than variables the remaining genomic locations for observations of Class 2.

Although both panels of Figure 2.9 show that the block of variables within genomic location of Chromosome 14 have high data variation for either the observations of Class 1 or Class 2, but the variability of those of Class 1 is not as high as the ones of Class 2. Thus, as shown in Figure 2.3, within-group variance of observations of Class 2 contributes upon the overall variance more that

within-group variance of observation of Class 1 does. This figure shows that among 22 genomic locations, variables within genomic locations of Chromosome 3 and Chromosome 10 have much higher variance than the remaining genomic locations. Similar pattern can be seen for within-group variance of observations of Class 2.

Returning to both panels of Figure 2.8, it is also obvious from this figure that, in terms of group mean, observations of Class 2 have higher mean than samples of Class 1 for genomic location of Chromosome 3. This indicates the presence of the difference between groups of observations of this dataset which can be exploited for classification purpose. This result is confirmed by the plot presented in Figure 2.10. This figure displays the plot of the difference between the group-mean of observations of Class 1 and the group-mean of the ones of Class 2.

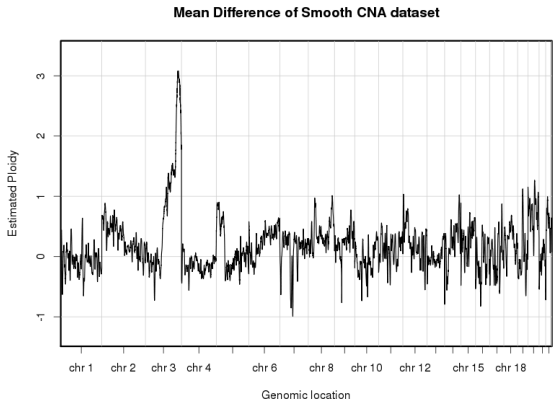


Figure 2.10: Line plot of the difference between mean of samples from distinct groups of observations. The mean difference is obtained by subtracting the mean of each variable for the observations of Class 1 from the mean of each variable for those of Class 2.

Plot of mean difference in Figure 2.10 shows that there is one block of variables within genomic location of Chromosome 3 whose higher mean difference than the remaining variables. Several variables within this block have mean difference that is greater than three. In addition, there are also several blocks of variables whose unit mean difference. These variables spread across various genomic locations.

This result indicates that those variables also have ability to perform data classification. However, their ability for carrying out data classification is not as strong as variables within Chromosome 3.

2.3.3 DNACopy CNA Dataset

Let us now consider DNACopy CNA dataset. This dataset has more complex structure than Smooth CNA dataset does. Firstly, DNACopy CNA dataset contains much more variables which have extreme observations as shown in Figure 2.11. These variables are not only found in Chromosomes 3 and 14, but also in some other genomic locations.

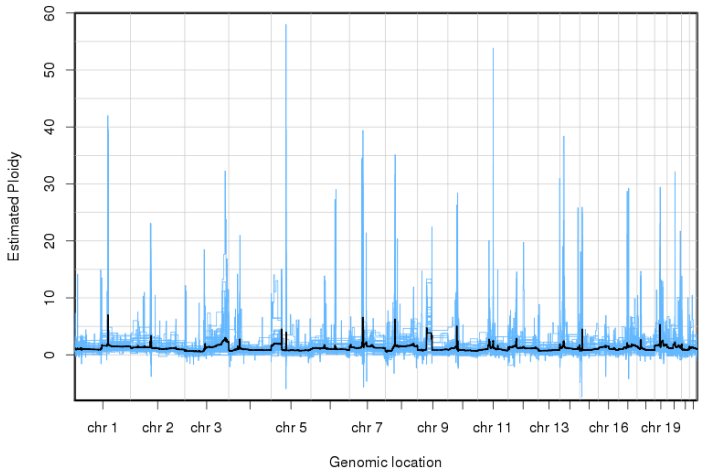


Figure 2.11: Plot for mean of each variable (gene segment) along with line plot for each observation of DNACopy CNA dataset. The black solid line presents the plot of mean and the blue region nearby this black solid line is the overlapping line plot for each observation.

In addition, compared to Smooth CNA dataset, DNACopy CNA dataset has much more extreme observations. As the result of this, DNACopy CNA dataset has more variables with much more variability than those of Smooth CNA dataset. We visualise this information in the plot of variance of individual variable of DNACopy CNA dataset as in Figure 2.12.

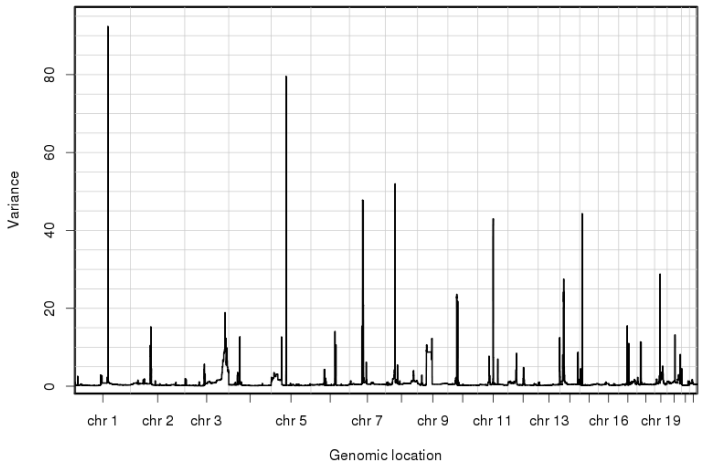


Figure 2.12: Line plot of variance for each variable of DNACopy CNA dataset. Each point in this line plot corresponds to the variance of a single variable.

These extreme observations are also responsible for many variables being skewed to the right as shown in Figure 2.13. This figure indicates the presence of more skewed variables in DNACopy CNA dataset than in Smooth CNA dataset. For the latter we only find positive skewed variables, while for the former we even get negative skewed variables.

Moreover, Figure 2.14 shows the p-values attained from performing normality test on each individual variable of DNACopy CNA dataset. Both panels of this figure indicates the presence of less normally distributed variables for this dataset. Histogram in Panel (b) displays that there are 90% of variables yield p-values of normality test less than 0.05. With regard to the relation between normality and skewness of individual variable for this dataset, we also find that all right-skewed variables are not normally distributed. Likewise, we get large number of variables whose nearly zero skewness which are not normally distributed.

Furthermore, this dataset also contains several blocks of correlated variables. These variables have high within block correlation. Figure 2.15 shows the presence of such these blocks of variables using the bright shaded area in the image of

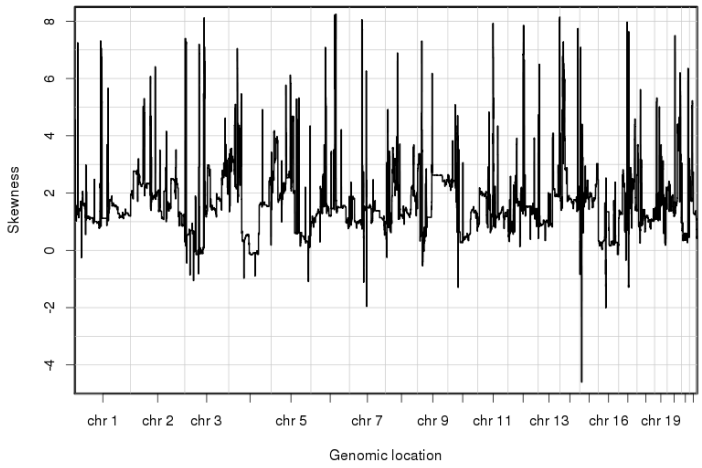


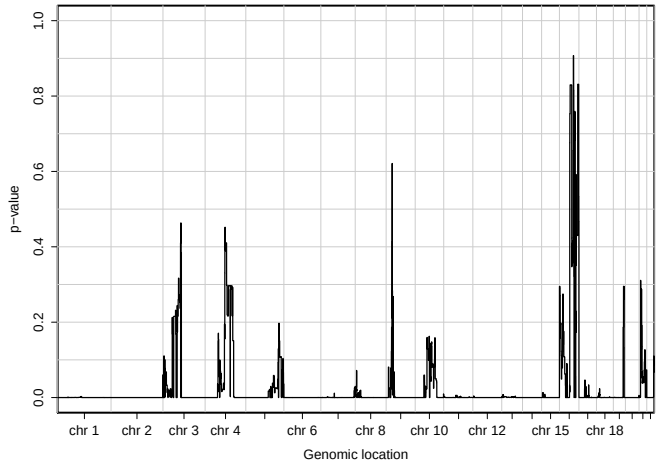
Figure 2.13: Line plot for the skewness of each variable for DNACopy CNA dataset. Each point in this line plot corresponds to the skewness of a single variable.

correlation matrix displayed. In addition, we present the image of correlation matrix for variables of genomic regions of Chromosomes 3 and 10 as in Figure 2.16.

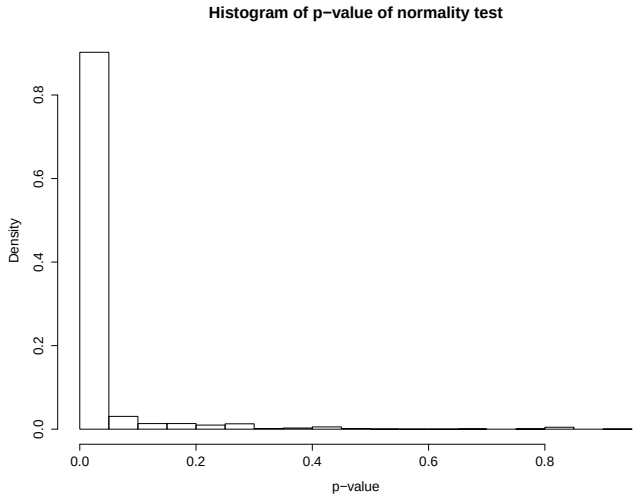
Having discussed the general description of DNACopy CNA dataset, let us now consider to describe DNACopy CNA dataset in the light of data classification task. We begin by presenting line plots for all samples from each group of observations along with their group mean in Figure 2.17 to show the main difference of characteristics between the observations of Class 1 and Class 2. The black solid lines within two panels of this figure display the plots for means of each group of observations, whereas the blue lines show the line plots for an individual observation of each group.

Looking at Figure 2.17, it is obvious that observations from either Class 1 or Class 2 have high variability. One can easily notice the presence of many extreme measurements across all genomic locations. As a result, samples of both classes yield high within-group variance as shown in Figure 2.18.

Compared to samples from Smooth CNA dataset, observations of DNACopy



(a)



(b)

Figure 2.14: Plot and histogram of p-values of Normality Test for DNACopy CNA dataset. Panel (a) presents the line plot for p-values of Normality Test performed on each variable of DNACopy CNA dataset. Each dot in this line plot corresponds to the p-value of Normality Test upon a single variable. Panel (b) shows the histogram of these resulting p-values.

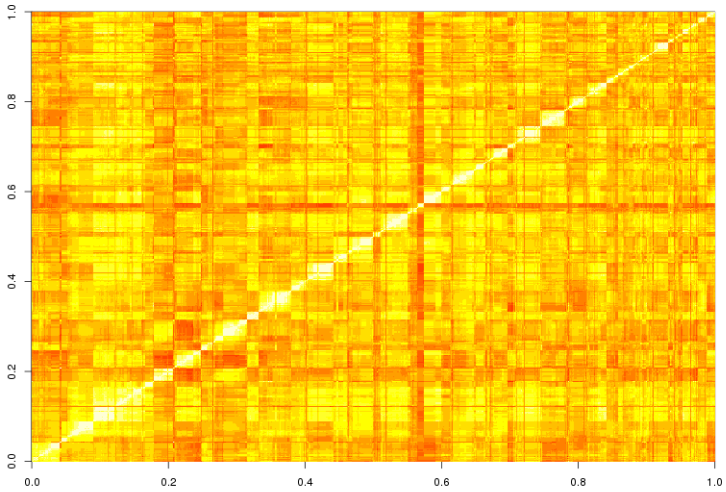
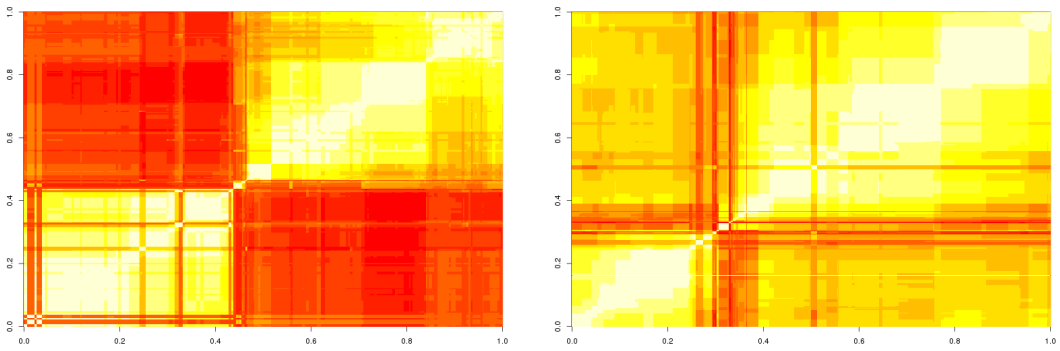


Figure 2.15: Image of correlation matrix for the whole genome of DNACopy CNA dataset. Both the vertical and horizontal axes correspond to the whole genomic region of DNACopy CNA dataset.

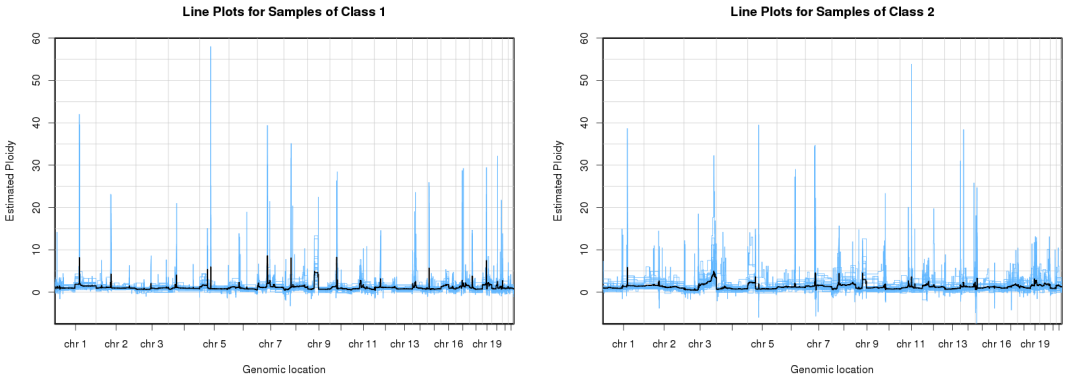
CNA dataset from both classes have much higher within-group variance. Both Panel (a) and Panel (b) of Figure 2.18 show that there are blocks of variables in many genomic locations such as Chromosomes 1, 5, 7, and 14 that have high within-group variance. As a result of this, variables from these genomic locations produce high overall variance as shown in Figure 2.12. This figure shows the presence of extremely high variance variables in genomic locations of Chromosomes 1, 5, 7, 8, 11 and 14.

Returning to Figure 2.17, it is also apparent from this figure that, in terms of group mean, samples of Class 2 have higher mean than samples of Class 1 for genomic location of Chromosome 3. However, there are also some other blocks of variables which spread across various genomic locations where observations of Class 1 have higher mean than samples of Class 2. One can easily remark these variables at genomic locations of Chromosomes 1 and 10. This indicates that, for DNACopy datasets, there are several blocks of variables can be utilized for classification purpose. This result is confirmed by the plot of mean difference



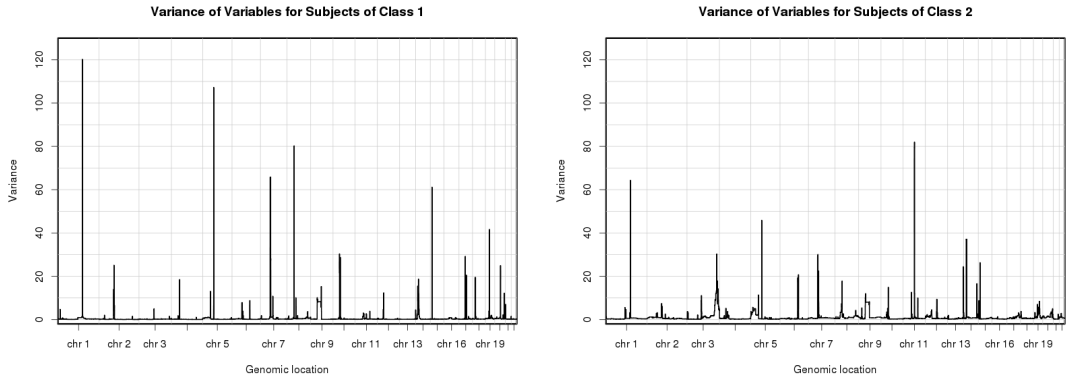
(a) (b)

Figure 2.16: Image of correlation matrix for the variables within Chromosomes 3 and 10 of Smooth CNA dataset. Panel (a) presents the image of correlation matrix produced by variables within genomic region of Chromosome 3. Both the vertical and horizontal axes of the image in this panel correspond to the variables within genomic region of Chromosome 3. Panel (b) displays the image of correlation matrix produced by variables within genomic region of Chromosome 10. Both the vertical and horizontal axes of the image in this panel correspond to the variables within genomic region of Chromosome 10.



(a) (b)

Figure 2.17: Line plot for the group mean of each variable along with line plot for each observation of the corresponding group for DNACopy CNA dataset. Panel (a) shows the plots for the observations of Class 1, whereas Panel (b) presents the plots for the observations of Class 2. The black solid line presents the plot of group mean and the blue region nearby this black solid line is the overlapping line plot for each observation within the corresponding group.



(a) (b)

Figure 2.18: Plots for within-group variance of samples from Group 1 and Group 2 of DNACopy CNA dataset. Panel (a) presents line plot of within-group variance of samples from Group 1. Each point in this plot corresponds with the variance of an individual variable for the observations of Class 1. Panel (b) shows line plot of within-group variance of samples from Group 2. Each point in this plot corresponds with the variance of an individual variable for the observations of Class 2.

between observations of Class 1 and Class 2 as presented in Figure 2.19. Figure 2.19 shows that there are blocks of variables whose high mean differenece within genomic locations of Chromosomes 3, 5, 7, 8, 10 and 19. These variables have mean difference that is greater than 3 or less than -3. In addition, there are also several blocks of variables whose mean difference ranges between 1 and 2.

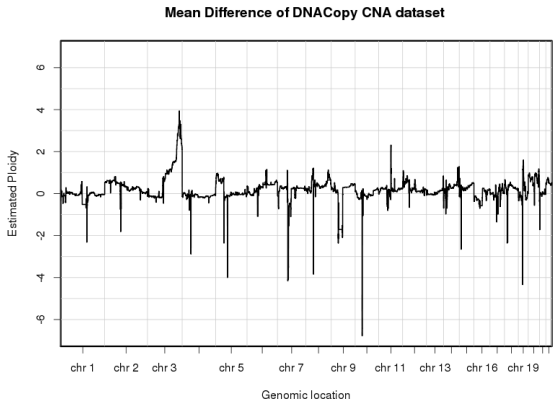


Figure 2.19: Line plot of the difference between mean of samples from distinct groups of observations. The mean difference is obtained by subtracting the mean of each variable for the observations of Class 1 from the mean of each variable for those of Class 2.

2.4 Simulated Data

Having described two real datasets used in this research, let us now turn to describe the simulated datasets that will be generated for our simulation study. On general, we wish to generate simulated datasets which have more variables than observations and contain block correlated structure. To begin with, we describe the population from which we generate these datasets and we continue with the explanation on how this data generating process carried out.

With regard to the population from where the simulated datasets are generated, dealing with datasets that consist of two distinct groups of observations, we should define two sub-populations where we draw the samples from. For the sake of simplicity, we only consider the case of populations that have distribution as mixtures of two multivariate normal distributions.

2.4.1 Mean, Within-group Covariance and Within-group Correlation of Population

Let the first sub-population be a sub-population which has multivariate normal distribution $N_p(\boldsymbol{\mu}^{(1)}, \Sigma^{(1)})$ and the second sub-population be a sub-population which has multivariate normal distribution $N_p(\boldsymbol{\mu}^{(2)}, \Sigma^{(2)})$ where $\boldsymbol{\mu}^{(k)}$ is the mean vector of the k-th sub-population and $\Sigma^{(k)}$ is the covariance matrix of the k-th sub-population for $k = 1, 2$. For simplicity, we set $\boldsymbol{\mu}^{(1)}$ as a zero vector

$$\boldsymbol{\mu}^{(1)} = \mathbf{0}$$

and $\boldsymbol{\mu}^{(2)}$ as a vector of not-all-zero entries

$$\boldsymbol{\mu}^{(2)} = \mathbf{d}.$$

We name this vector $\mathbf{d}$ as the vector of mean difference. We introduce this vector as the vector which is used to segregate the second sub-population from the first one. In terms of classification task that we are going to discuss, this quantity can also be seen as between-group variation. Whereas, the covariance matrix $\Sigma^{(k)}$ can be viewed as within-group variation. Hence, this covariance matrix will also be named as within-group covariance matrix of the k-th sub-population.

Furthermore, within-group covariance matrices are obtained from variance of each variable along with the correlation matrices $P^{(k)}$, for $k = 1, 2$, as we are going to study the effect of the presence of correlation between variables. We define

$$\Sigma^{(k)} = D^{(k)} P^{(k)} D^{(k)}$$

where $D^{(k)}$ is diagonal matrix of the k-th sub-population standard deviation.

2.4.2 Settings for Population Study

Based on the explanation from the subsection above on some quantities dealing with population study, we take mean difference, within-group covariance and correlation structure into consideration. To begin with, we set $p = 150$. Henceforth, for simulated datasets, we always deal with 150-variate normal population.

Moreover, as we are going to do simulation study on some populations with blocks of correlated-variates, we then define the correlation matrix P as a three-block-diagonal matrix of the form

$$P = \begin{pmatrix} P_1 & 0 & 0 \\ 0 & P_2 & 0 \\ 0 & 0 & P_3 \end{pmatrix}$$

where

$$P_l = (1 - \rho) I_{50} + \rho J_{50}$$

where I_{50} is identity matrix with size 50×50, J_{50} is and all-ones matrix with size 50×50, ρ is correlation value and $l = 1, 2, 3$. We set the correlation values within entire blocks to be same and we set them to range from $\rho = 0$ to 0.9.

Furthermore, we consider two cases for within-group covariance matrix. Firstly, we consider the case of unit standard deviation, or $D^{(k)} = I_{50}$, for $k = 1, 2$. Secondly, we consider the case of three standard deviation, or $D^{(k)} = 3I_{50}$, for $k = 1, 2$. Hence, for the first setting, we obtain

$$\Sigma^{(k)} = I_{150} P^{(k)} I_{150} = P^{(k)}$$

for $k = 1, 2$, while for the second one, we have

$$\Sigma^{(k)} = 3I_{150} P^{(k)} 3I_{150} = 9P^{(k)}$$

for $k = 1, 2$.

Turning now to mean difference vector $\mathbf{d}$. We define the structure of mean difference vector $\mathbf{d}$ by considering the presence of three diagonal block structure in correlation matrix. Therefore we set the mean difference vector $\mathbf{d}$ as

$$\mathbf{d} = (\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3).$$

where $\mathbf{d_1}$ is a mean difference vector of the first 50 variates, $\mathbf{d_2}$ is a mean difference vector of the second 50 variates and $\mathbf{d_3}$ is a mean difference vector of the last 50 variates.

In terms of mean difference structure, we consider three distinct cases: small difference, medium difference and large difference. For small difference of mean, we set

$$\mathbf{d}_1^T = (0, 0, \ldots, 0),$$

$$\mathbf{d}_2^T = (-\frac{0}{50}, -\frac{1}{50}, \ldots, -\frac{49}{50}),$$

and

$$\mathbf{d}_3^T = (\frac{0}{50}, \frac{1}{50}, \ldots, \frac{49}{50}).$$

While for medium difference of mean, we set

$$\mathbf{d}_1^T = (0, 0, \ldots, 0),$$

$$\mathbf{d}_2^T = (-\frac{0}{25}, -\frac{1}{25}, \ldots, -\frac{49}{25}),$$

and

$$\mathbf{d}_3^T = (\frac{0}{25}, \frac{1}{25}, \ldots, \frac{49}{25}).$$

For large difference of mean, we set

$$\mathbf{d}_1^T = (0, 0, \ldots, 0),$$

$$\mathbf{d}_2^T = (-\frac{0}{15}, -\frac{1}{15}, \ldots, -\frac{49}{16}),$$

and

$$\mathbf{d}_3^T = (\frac{0}{15}, \frac{1}{15}, \ldots, \frac{49}{16}).$$

2.4.3 Data Generating

Having defined some study populations, we will now move on to define the data generating procedure for each population defined above. To begin with, we define data matrix X as an 100×150 matrix that consists of 100 samples and 150 variables. We set the number of variables to be larger than the number of observations since we will perform simulation study on the case of more variables than observations. Hence, we define X as

$$\underset{100\times150}{\mathrm{X}} = \begin{bmatrix} \underset{n_1\times150}{\mathrm{X}^{(1)}} \\ \hline \underset{n_2\times150}{\mathrm{X}^{(2)}} \end{bmatrix}$$

where $\underset{n_1\times150}{\mathrm{X}^{(1)}}$ is data matrix of n_1 samples drawn from sub-population 1 which follows $N(\mu^{(\mathbf{1})}, \Sigma^{(1)})$ distribution and $\underset{n_2\times150}{\mathrm{X}^{(2)}}$ is data matrix of n_2 samples drawn from sub-population 2 which follows $N(\mu^{(\mathbf{2})}, \Sigma^{(2)})$ distribution. For simplicity, we name the samples drawn from sub-population 1 as samples of Class 1 and the ones drawn from sub-population 2 as samples of Class 2.

2.4.4 Conclusion

Thus far, we have discussed on the description of both real and simulated datasets. The chapter that follows moves on to explanation of Classification Trees as an algorithm used for data classification purpose along with their error rate prediction method.

Chapter 3

Classification Trees

3.1 Introduction

This chapter lays out the theoretical explanation on how to built Classification Trees and how to assess their performance. This explanation was begun with the presentation of a realisation of Classification Trees fitted using a toy example to explain the Classification Trees elements. This chapter then goes on explaining some theoretical derivation of formula used in constructing Classification Trees.

Having discussed the Classification Trees construction, this chapter continues on discussing on how to assess the Classification Trees performance. As a prediction rule, one could evaluate Classification Trees performance by measuring their prediction accuracy. In this chapter we propose the prediction error rate estimation methods that based on cross-validation approach.

The theoretical explanation presented in this chapter will be used as the computational algorithm implemented for constructing Classification Trees of unprojected datasets in Chapter 4, building Classification Trees of principal component scores in Chapter 5 and fitting Classification Trees of independent components in Chapter 6.

3.2 Classification Trees

Classification Tree is a prediction rule that is used to predict the future classification of an observation. As a rule, Classification Tree is fitted using a dataset which is called learning set $\mathcal{L}$. The learning set $\mathcal{L}$ consists of measurement data on n cases or objects together with their actual classifications.

Using the learning set $\mathcal{L}$, a Classification Tree is constructed by finding a rule that partitions the observations within the learning set $\mathcal{L}$ into two disjoint subsets such that each subset contains as many as possible observations of the same class. This rule is referred as a splitting rule. This partitioning task goes on each subset of the learning set $\mathcal{L}$ produced. For this reason, Classification Trees can be seen as a recursive partitioning algorithm. Zhang & Singer (2010) uses this term instead of Classification Trees. The same name is also used for an R (R Core Team, 2015) for Classification Trees which is developed by Therneau *et al.* (2017). They use the term rpart. Breiman *et al.* (1984) proposed CART as one of the algorithms for this partitioning purpose. We base our work here mostly on the method proposed by Breiman *et al.* (1984).

There are two aims of the Classification Trees application. Firstly, the Classification Trees are used to produce an accurate prediction rule. Secondly, the Classification Trees are applied to study the predictive structure of classification task carried out. For the former, one focuses on finding an accurate prediction model. Given an object without an actual classification, the fitted trees is used to predict its classification. While for the latter, one applies the Classification Trees to get better insight about what variables or interaction of variables that manage the classification task. Hence, one can perceive the reason behind the condition that prompts which class the samples or cases belong to.

In this thesis, we apply the Classification Trees as a tool to identify a gene or combination of genes which are responsible for the characterisation of adeno carcinoma and squamous carcinoma lung cancer on both Smooth CNA DNA and DNACopy CNA datasets as previously explained. For this reason, we study the Classification Trees for binary classification problem. The fitted trees are used to predict the stratification of the future lung cancer cases afterwards. Therefore, we intend the application of the Classification Trees for both purposes.

3.2.1 Mathematical Expression of Classification Trees

Prior to the discussion on technical aspects of the Classification Trees, in this subsection, we present a Classification Tree as a mathematical entity. We begin with expressing the probability space for the learning set $\mathcal{L}$.

Let $\mathcal{X}$ be a p-dimensional space and $\mathcal{Y}$ be a one-dimensional space. We define the space $\mathcal{X} \times \mathcal{Y}$ as a set of all couples $(\mathbf{x}^T, y)$, where $\mathbf{x} \in \mathcal{X}$ and y is a class label, $y \in \mathcal{Y}$. Let $P(A, y)$ be a probability defined on space $\mathcal{X} \times \mathcal{Y}$, where $A \subset \mathcal{X}$ and $y \in \mathcal{Y}$. We define $P(A, y)$ as a probability of a case drawn randomly from a population having probability distribution $P(A, y)$ and it has a measurement $\mathbf{x}$ along with a classification y. For the shorter notation, we denote $P(A, y) = F$

Moreover, we define the learning set $\mathcal{L}$ as a random sample of size n which is drawn from the population with probability distribution F. Hence, we denote the learning set $\mathcal{L}$ as

$$\mathcal{L} = [(\mathbf{x}_1^T, y_1), (\mathbf{x}_2^T, y_2), \ldots, (\mathbf{x}_n^T, y_n)]^T \tag{3.2.1}$$

where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, $i = 1, 2, \ldots, n$.

Using this learning set, we construct the Classification Trees $\mathcal{R}$. It should be highlighted here that we do not express the Classification Trees $\mathcal{R}$ as a function of a measurement vector $\mathbf{x}$ of the learning set $\mathcal{L}$. Instead, we view the Classification Trees as a rule defined by both the measurement vectors of the learning set $\mathcal{L}$ along with their actual classification. Hence, we denote the Classification Trees fitted using the learning set $\mathcal{L}$ as $\mathcal{R}_\mathcal{L}$

However, we can express the Classification Trees $\mathcal{R}_\mathcal{L}$ as a function of a measurement vector in terms of prediction. Having fitted the Classification Tress $r_\mathcal{L}$ for a given learning set $\mathcal{L}$, we perform the prediction for a new observation drawn from the same population as the learning set $\mathcal{L}$. Let $\mathbf{x}_0$ be the measurement upon this new observation. We define the prediction for this measurement as $\mathcal{R}_\mathcal{L}(\mathbf{x}_0)$.

3.2.2 Elements of Classification Trees

In the following subsection, we explain the elements of the Classification Trees through the tree constructed from a toy example dataset. Given a dataset of two explanatory variables, $\mathbf{x}_1$ and $\mathbf{x}_2$, and one categorical response variable with two

categories for 76 samples, we construct the Classification Trees as presented in Figure 3.1. We name the subset of the dataset with response variable is equal to 1 as samples from Class 1 and the one with response variable is equal to 2 as samples from Class 2. Henceforward we restrict our study on the binary Classification Trees as we are only dealing with the classification task on the datasets with two classes of samples.

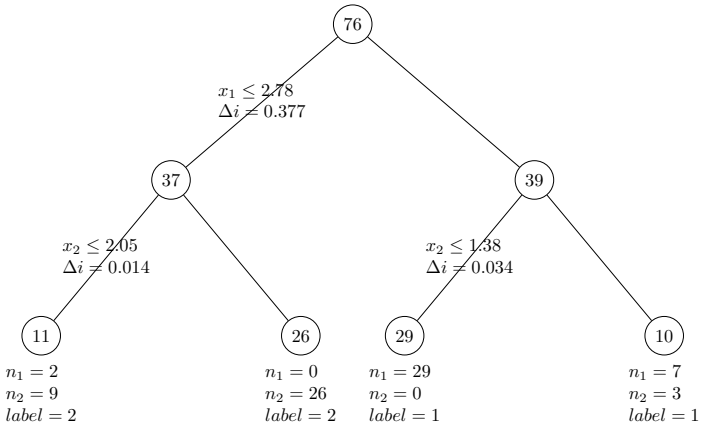


Figure 3.1: Classification Tree of toy example dataset. This Classification Tree consists of three splittings. The first splitting is the one which has variable $\mathbf{x}_1$ as classifying variable. This splitting has maximum goodness of split $\Delta i = 0.377$. The second and the third splittings are the ones which have variables $\mathbf{x}_2$ as the classifying variables. This Classification Tree have four terminal nodes. Two of them are labelled as Class 2 and the rest as Class 1. The notation n_1 and n_2 in each terminal node correspond with the number of observations from Class 1 and Class 2 fall into that node, respectively.

We present the scatter plot for this dataset as in Figure 3.2. The black dots in the scatter plot present samples of Class 1, while the red dots present samples of Class 2. We also add the splittings obtained by the Classification Trees on the scatter plot to visualise the classification task performed.

Figure 3.1 indicates that there are three main elements of the Classification Trees: nodes, splittings and class labels. The tree diagram in Figure 3.1 has three layers of nodes. The first layer node which contains 76 samples is named as the

root node or parent node. This node then splits into two nodes which contain 37 and 39 samples in the second layer. Both nodes within the second layer then split to produce four nodes in the third layer. All nodes within the second and the third layers refer to daughter nodes.

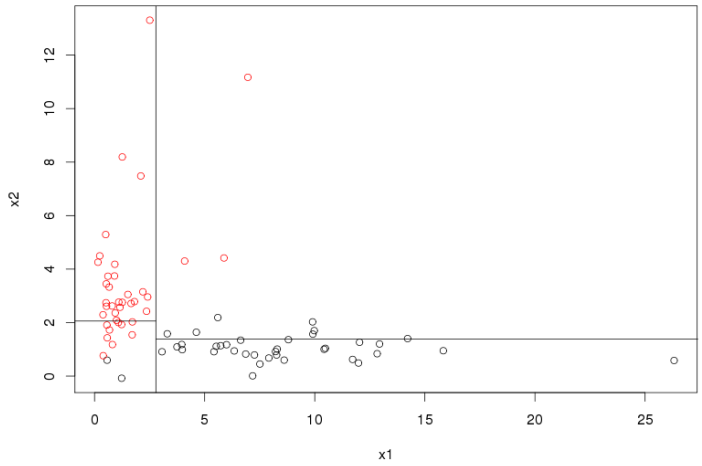


Figure 3.2: Scatterplot for toy example dataset along with the splittings obtained from Classification Tree. The horizontal axis of this plot corresponds with variable $\mathbf{x}_1$ while the vertical axis corresponds with variable $\mathbf{x}_2$. Three splitting lines within this panel corresponds with the splittings obtained from Classification Tree of Panel (b)

To perform the splitting, one needs to find a rule for splitting and partitions cases within a certain node based the rule produced. For the dataset with one explanatory variable, the splitting rule is defined as a cut-off point. This cut-off is a mid value of two neighbouring values of the explanatory variable that corresponds to a quantity called maximum goodness of split. We will discuss this quantity in the next following subsection.

Furthermore, for the dataset with two or more explanatory variables, the splitting involves the search for a variable which produces the highest maxima goodness of split. The splitting rule is then defined as the cut-off point that corresponds to the maxima goodness of split within this selected variable.

Figure 3.1 shows that variable $\mathbf{x}_1$ is picked as the classifying variable. It is used to partition samples within the root node into the daughter nodes in the second layer. Samples with $x_{i1} \leq 2.78$, for $i = 1, 2, \ldots, n$, fall into the left node whilst samples with $x_{i1} > 2.78$, for $i = 1, 2, \ldots, n$, fall into the right node. The splitting process goes on these two daughter nodes to produce four nodes in the third layer.

These four nodes then stop splitting as they achieve the determined stopping criteria. The tree stops splitting either the resulting daughter node is a pure node or it contains less than or equal 10 samples. We name these non-splitting nodes as terminal nodes. These terminal nodes then being provided with the class labels. We will also discuss on the split stopping criteria and class labelling in the next two sections.

3.2.3 Node Impurity

Classification Trees as the recursive partitioning algorithm partition cases within a single node into two daughter nodes. For binary classification, this partitioning task is performed to seek two homogeneous daughter nodes in the sense that each daughter nodes contains observations of the same class. A homogeneous node that contains observations of one class is also referred as a pure node. One might measure homogeneity of a node using a quantity called node impurity. To explain the mathematical formulation for this partitioning task, we begin with deriving the impurity of a single node in a simple learning set of one explanatory variable and one categorical response variable.

Let t be a single node contains n_t observations that come from two distinct classes. Let $n_t^{(1)}$ and $n_t^{(2)}$ be the number of cases from Class 1 and Class 2 within node t, respectively. Let $\mathcal{L}_t$ be a learning set represents n_t univariate measurements over all observations within a single node t. We define the proportion of objects from Class l within node t as

$$p_t^{(l)} = \frac{n_t^{(l)}}{n_t} \tag{3.2.2}$$

where $l = 1, 2$.

Moreover, we define impurity function ϕ on the set of proportion $\{p_t^{(1)}, p_t^{(2)}\}$ with properties

1. ϕ is maximum only at the point $\left(\frac{1}{2}, \frac{1}{2}\right)$,
2. ϕ achieves its minimum only at the points $(1, 0)$ and $(0, 1)$,
3. ϕ is a symmetric function of $p_t^{(1)}, p_t^{(2)}$.

Using impurity function ϕ, we have impurity measure for node t, $imp(t)$ as

$$imp(t) = \phi(p_t^{(1)}, p_t^{(2)}). \tag{3.2.3}$$

There are three common choices of impurity: Gini index, entropy and minimum error. Following Breiman *et al.* (1984), we choose Gini index as impurity. Hence, for node t we have

$$imp(t) = \phi(p_t^{(1)}, p_t^{(2)}) = \sum_{k=1}^{2} p_t^{(k)}(1 - p_t^{(k)}) = 2p_t^{(1)}p_t^{(2)}. \tag{3.2.4}$$

Furthermore, let s be a split that partitions cases within node t into two daughter nodes, left daughter node and right daughter node, as shown in Figure 3.3. Let t_L and t_R be left node and right node, respectively. As defined for node t, both left node t_L and right node t_R have measure of impurity as follows. For the left node t_L

$$imp(t_L) = \phi(p_{t_L}^{(1)}, p_{t_L}^{(2)}) = \sum_{l=1}^{2} p_{t_L}^{(l)}(1 - p_{t_L}^{(l)}) = 2p_{t_L}^{(1)}p_{t_L}^{(2)} \tag{3.2.5}$$

where $p_{t_L}^{(l)}$, for $l = 1, 2$, is the proportion of cases from Class l fall into node t_L.

Likewise, for the right node t_R

$$imp(t_R) = \phi(p_{t_R}^{(1)}, p_{t_R}^{(2)}) = \sum_{l=1}^{2} p_{t_R}^{(l)}(1 - p_{t_R}^{(l)}) = 2p_{t_R}^{(1)}p_{t_R}^{(2)} \tag{3.2.6}$$

where $p_{t_R}^{(l)}$, for $l = 1, 2$, is the proportion of cases from Class l fall into node t_R.

Carrying out partitioning over node t leads to the decrease in impurity of both left node t_L and right node t_R. One might measure this decrease using a quantity

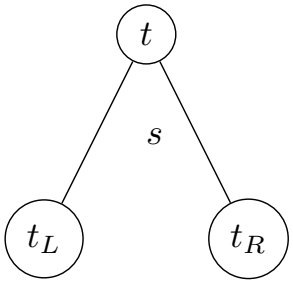


Figure 3.3: Diagram for simple Classification Trees with single splitting. In this diagram, node t is the root node and nodes t_L and t_R can be seen as either the daughter nodes of node t or the terminal nodes for the splitting s.

called the goodness of split. Let $\Delta imp(s,t)$ be the goodness of split for split s over cases in node t, we define $\Delta imp(s,t)$ as

$$\Delta imp(s,t) = imp(t) - p_{t_L} imp(t_L) - p_{t_R} imp(t_R). \tag{3.2.7}$$

where p_{t_L} is the proportion of objects in node t fall into node t_L and defined as

$$p_{t_L} = \frac{n_{t_L}}{n_t}, \tag{3.2.8}$$

while p_{t_R} is the proportion of objects in node t fall into node t_{t_R} and defined as

$$p_{t_R} = \frac{n_{t_R}}{n_t} \tag{3.2.9}$$

3.2.4 Splitting Rules

Having defined the node impurity along with the goodness of split for a single split carried out over a node, we move on to the discussion on how to perform the splitting itself. As previously mentioned, split s partitions cases within node t so that both left node t_L and right node t_R are as homogeneous as possible. This can be achieved by splitting the observations within node t into two daughter node such that the corresponding goodness of split achieves its maxima. Hence, finding the splitting rules for a single node can be seen as seeking for the mid point of two neighbouring observations within this node which corresponds to

the highest goodness of split. To be precise, let us express this procedure into mathematical notation. We begin with binary classification task over univariate case.

Let t be a root node which contains n observations. Let $\mathcal{L} = (\mathbf{x}, \mathbf{y})$ be a learning set consists of univariate measurements on n cases within node t along with their actual classification. Let $n^{(l)}$ be the number of observations from Class l, for $l = 1, 2$ within node t.

Prior to partitioning cases within node t, one has to sort the learning set $\mathcal{L}$ with respect to measurement vector $\mathbf{x}$. Let $\tilde{\mathcal{L}}$ be the ordered learning set and $\tilde{\mathbf{x}}^T = (x_{(1)}, x_{(2)}, \ldots, x_{(n)})$ be the ordered vector of $\mathbf{x}$. Moreover, we define a cut-off c_i over the ordered learning set $\tilde{\mathcal{L}}$

$$c_i = \frac{x_{(i)} + x_{(i+1)}}{2} \tag{3.2.10}$$

for $i = 1, 2, \ldots, n-1$. Let C be the set for all possible cut-off c_i, for $i = 1, 2, \ldots, n-1$. Thus $C = \{c_1, c_2, \ldots, c_{n-1}\}$.

Using this cut-off, we partition cases within node t into the left node and the right node. All cases with $x_{(i)} < c_i$, for $i = 1, 2, \ldots, n-1$, will fall into the left node t_L, while the rest will fall into the right node t_R. Hence, we end up with $n-1$ possible splits which corresponds to $n-1$ possible values of the goodness of split for node t which contains n cases.

Let s_i be these $n-1$ possible splits and $\Delta imp(s_i, t)$ as their corresponding goodness of split values, for $i = 1, 2, \ldots, n-1$. Among these $n-1$ candidates for the cut-off, we pick the cut-off which corresponds to the highest goodness of split as the real cut-off for split s within node t. Let c_{max} be this cut-off and defined as

$$c_{max} = \underset{c_i \in C}{\operatorname{argmax}}\, \Delta imp(s_i, t), \tag{3.2.11}$$

and $\Delta imp(s, t)$ be the maximum goodness of split and defined as

$$\Delta imp(s, t) = \max_{i \in \{1,2,\ldots,n-1\}} \Delta imp(s_i, t). \tag{3.2.12}$$

We define the rule for this split as all cases with $x_{(i)} < c_{max}$, fall into the left node t_L, while the rest fall into the right node t_R, for $i = 1, 2, \ldots, n-1$. Algorithm 1 presents this procedure in algorithmic form.

Having discussed how to perform partitioning over single node using learning set which has univariate explanatory variable, the following is the derivation for multivariate case. We begin with deriving the splitting rule for the root node t which contains n observations as described for univariate case above. Yet, now we are dealing with the multivariate case.

Let $\mathcal{L} = (\mathrm{X}, \mathbf{y})$ be a learning set consists of multivariate measurements on n cases within node t along with their actual classification. Let X be any $n \times p$ matrix that represents these n measurements on p dimensional space $\mathcal{X}$. Hence matrix X contains p variables.

In contrast to univariate case in which we pick a cut-off point as the one that corresponds with the highest goodness of split, for this multivariate case we have to seek for a variable which has highest maximum goodness of split among p variables. Having obtained the variable with the highest maximum goodness of split, we can pick the cut-off point which corresponds to this variable as the splitting rule. Thus, we define the goodness of split that corresponds with the cut-off as

$$\Delta imp(s,t) = \max_{j \in \{1,2,\ldots,p\}} \Delta imp(s,t)_j \tag{3.2.13}$$

where $\Delta imp(s,t)_j$ is the maximum goodness of split for the j-th variable. The cut-off point c_{max} for this case is then defined as

$$c_{max} = \operatorname*{argmax}_{j \in \{1,2,\ldots,p\}} \Delta imp(s,t)_j, \tag{3.2.14}$$

Having obtained the variable which has the highest goodness of split along with its cut-off point, the splitting s upon node t with multivariate variables can be performed in the same way as it is for univariate case as described above. Algorithm 2 presents this procedure in algorithmic form.

3.2.5 Classification Trees Construction

The previous subsection has focused on performing partitioning either for univariate or multivariate measurements. This subsection will discuss on constructing the whole Classification Trees by recursively partitioning the observations the learning set $\mathcal{L}$.

To begin the Classification Trees construction, one applies the splitting procedure presented above to partition the root node into two daughter nodes. Once all cases within the root node partitioned into these two daughter nodes, one uses the same splitting procedure upon each of these two daughter nodes to partition all observations into the new four daughter nodes. This splitting process then goes on each daughter node produced until a criterion is achieved. This criterion is called stopping criterion.

As previously mentioned, during the Classification Trees construction, the splitting is performed upon a parent node to produce two purer daughter nodes. Hence, once a pure daughter node is obtained, the need to proceed partitioning is eliminated. However, it is not always be the case that one ends up with a pure daughter node.

In the absence of pure daughter node, one can set a tuning parameter $n.min$ which is the number of minimum objects in a daughter node as a stopping criterion. In this thesis, we use these two criteria for stopping the splitting. In addition, one might name a node that does not split any more as a terminal node or a leaf. Meanwhile, one might call a node that splits as an internal node. The node that contains the whole observations within the learning set is names the root node.

Once the splitting process stopped, one labels each terminal node obtained using majority vote. As we are dealing with the binary classification task, one labels a terminal node by the class in which majority of observations belong to. However, in the presence of the balance number of observations from both classes, one labels this terminal node by picking the class label randomly.

In summary, the Classification Trees construction procedure consists of the following three steps:

1. the selection of the splits,
2. the decisions when to declare a node terminal or to continue splitting it and
3. the assignment of each terminal node to a class

Algorithm 2 presents this procedure in algorithmic form.

We can also express the above procedure into the Algorithm 1,2 and 3. The first algorithm can be used for getting the goodness of split and the threshold of single variable. The second algorithm can be applied for choosing a single variable as the classifying variable. The third algorithm can be utilized for constructing the Classification Trees.

Algorithm 1 Getting threshold of single variable

input: $\mathbf{x}^T = (x_1, x_2, \ldots, x_n)^T$ and $\mathbf{y}^T = (x_1, x_2, \ldots, x_n)^T$

output: goodness of split and threshold

1: Build dataset using explanatory variable $\mathbf{x}$ along with response variable $\mathbf{y}$
2: Sort dataset based on the value of $\mathbf{x}$ variable
3: **for** $i = 1$ to $length(\mathbf{y} - 1)$ **do**
4: Find the mid values using $mid.x_i = \frac{x_i + x_{i+1}}{2}$ as threshold candidates
5: Count the number of objects from both classes that fall into the left node or the right node using . An object falls into left node if it is less than or equal to $mid.x_i$. Otherwise, it falls into the right node
6: Count the corresponding impurity for the ith splitting using equation
7: count the corresponding goodness of split for the ith splitting using equation
8: **end for**
9: Find the maximum value of goodness of split

Algorithm 2 Choosing variable that will be used as classifying variable

input: data matrix $\mathrm{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p)$ and the categorical variable $\mathbf{y}$

output: classifying variable, goodness of split, splitting threshold

1: $no.var \leftarrow no.column$ of X
2: **for** $j = 1$ to $no.var$ **do**
3: Find the the goodness of split and the threshold of each variable $\mathbf{x}_j$ using Algorithm 1.
4: Find the maximum value of goodness of split
5: **end for**
6: Choose the variable that has maximum goodness of split as the classifying variable
7: Choose the threshold that corresponds with the classifying variable as the splitting threshold

Algorithm 3 Getting tree algorithm

input: learning set $\mathcal{L}$, tuning parameter $n.min$

output: classification trees

1: **if** Number of observations $< n.min$ or all observations are from the same class **then**
2: Define a terminal node and assign the class label
3: **else**
4: Find the classifying variable along with the splitting cut-off by applying Algorithm 2.
5: Classify objects which have measurements less than or equal to the splitting threshold upon the splitting variable to the left node.
6: Classify objects which have measurements greater than the splitting threshold upon the splitting variable to the right node.
7: Apply Algorithm 3 on the left node.
8: Apply Algorithm 3 on the right node.
9: **end if**

3.2.6 Classification Trees for Inference and Prediction Purposes

Applying the procedure explained above upon a learning set $\mathcal{L}$, we end up with a Classification Trees. This fitted Classification Trees can be used to get the better insight about variables or interaction of variables that manage the classification task. This can be performed by identifying the variables picked as the classifying variables upon the splittings within the trees. It also can be accomplished by observing the maximum goodness of split for a single split. A variable which has high maximum goodness of split can be identified as the variable that is responsible for the classification carried out. In this case, the Classification Trees is utilized for inferential purpose where we focus more on the interpretability of the Classification Trees.

On the other hand, the Classification Trees can also be implemented for the prediction purpose. Having fitted the Classification Trees using the learning set $\mathcal{L}$, we make use of this Classification Trees to predict the classification for a new observation drawn from the same population as all the observations within the learning set $\mathcal{L}$ are. In this case, it is the prediction accuracy matters. The next section describes this the theoretical aspect and some practical issues of estimating the accuracy of the Classification Trees as a prediction rule.

3.3 Stratified Cross Validation for Classification Trees Misclassification Rate

Implementing the Classification Trees as a prediction rule one should assess the Classification Trees generalisation performance. This sort of performance relates to the Classification Trees prediction ability on independent test set. Given a Classification Trees that is fitted using the learning set, one should assess the trees accuracy upon the new dataset which is called test set or validation set.

In this section, we present the true error rate as a quantity to assess the accuracy of the Classification Trees fitted using a learning set in predicting a classification of a new test point. Once we discussed the true error rate, we propose several approach for estimating this quantity.

3.3.1 True Error Rate

To begin with, we refer back to the mathematical expression of the Classification Trees as presented in Section 3.2.1. In that section, we define the learning set $\mathcal{L}$ as a sample of size n where $\mathcal{L} = \left[(\mathbf{x}_1^T, y_1), (\mathbf{x}_2^T, y_2), \ldots, (\mathbf{x}_n^T, y_n)\right]^T$. We also assume that $(\mathbf{x}_1^T, y_1), (\mathbf{x}_2^T, y_2), \ldots, (\mathbf{x}_n^T, y_n) \overset{\text{i.i.d}}{\sim} F$. Using the learning set $\mathcal{L}$, we fit the Classification Trees and denote this Classification Trees as $\mathcal{R}_\mathcal{L}$.

Let $(\mathbf{x}_0^T, y_0)$ be a new test point which is drawn from the same population of the learning set $\mathcal{L}$. Given the classification rule $\mathcal{R}_\mathcal{L}$, we define the prediction over $\mathbf{x}_0$ as $\mathcal{R}_\mathcal{L}(\mathbf{x}_0) = \widehat{y_0}$. Let $Q[y_0, \mathcal{R}_\mathcal{L}(\mathbf{x}_0)]$ be the discrepancy between the predicted classification $\widehat{y_0}$ and the actual classification y_0. $Q[y_0, \mathcal{R}_\mathcal{L}(\mathbf{x}_0)]$ is an indicator function defined as

$$Q[y_0, \mathcal{R}_\mathcal{L}(\mathbf{x}_0)] = \begin{cases} 0 & \text{if } \mathcal{R}_\mathcal{L}(\mathbf{x}_0) = y_0 \\ 1 & \text{if } \mathcal{R}_\mathcal{L}(\mathbf{x}_0) \neq y_0 \end{cases} \tag{3.3.1}$$

Using the notation presented above, we define the true error rate for the Classification Trees $\mathcal{R}_\mathcal{L}$ fitted using the learning set $\mathcal{L}$, Err, as the prediction error over an independent sample test $(\mathbf{x}_0^T, y_0)$ as

$$Err = Err(\mathcal{L}, F) = E_{0F} Q[y_0, \mathcal{R}_\mathcal{L}(\mathbf{x}_0)]. \tag{3.3.2}$$

The term E_{0F} indicates that Err is only random on $(\mathbf{x}_0^T, y_0)$. Meanwhile, both the learning set $\mathcal{L}$ and the prediction rule $\mathcal{R}_\mathcal{L}$ being fixed. The true error rate Err is also refered as the generalisation error as this quantity assess the generalisation performance of the prediction rule $\mathcal{R}_\mathcal{L}$ fitted using learning set $\mathcal{L}$ over another new independent test point. In addition, as Err is an expectation of an indicator function, so it can also be viewed as a probability of an independent test point $(\mathbf{x_0}^T, y_0)$ to be misclassified by the prediction rule $\mathcal{R}_\mathcal{L}$.

Furthermore, we define the expected value of the true error rate Err as

$$E_F\{Err\} = E_F E_{0F} Q[y_0, \mathcal{R}_\mathcal{L}(\mathbf{x}_0)]. \tag{3.3.3}$$

Unlike for the true error rate Err, for its expected value $E_F\{Err\}$, it is not only the test point $(\mathbf{x}_0^T, y_0)$ that is random, but also the learning set $\mathcal{L}$ along with the corresponding fitted Classification Trees $\mathcal{R}_\mathcal{L}$.

Therefore, the true error rate Err can be viewed as the expected error of a Classification Trees fitted using a particular learning set drawn from the population which has probability distribution F. Meanwhile, the expected true error rate $E_F\{Err\}$ can be seen as an average performance of the Classification Trees algorithm across learning sets drawn from the population which has probability distribution F. Braga-Neto & Dougherty (2004) introduced the term local performance for the the true error rate Err and global performance for the expected true error rate $E_F\{Err\}$.

The estimation methods presented in the following subsection will cover the estimation of the true error rate Err. Therefore, all of these methods only deal with assessing the performance of the Classification Trees fitted by a particular learning set rather than the average performance of the Classification Trees as a prediction rule upon a population.

3.3.2 Estimating True Error Rate

Having defined the true error rate as an assessment tool for the Classification Trees accuracy, we now move on to discuss its estimation methods.

Training Error Rate

In this part we discuss the training error for the Classification Trees $\mathcal{R}_{\mathcal{L}}$ which fitted using the learning set $\mathcal{L}$. We denote this error rate by $\overline{err}$ and employ it as an estimate for the true error rate Err. We define the training error $\overline{err}$ as

$$\overline{err} = Err(\mathcal{L}, \widehat{F}) = E_{0\widehat{F}} Q[y_0, \mathcal{R}_{\mathcal{L}}(\mathbf{x}_0)] = \frac{1}{n} \sum_{i=1}^{n} Q[y_i, \mathcal{R}_{\mathcal{L}}(\mathbf{x}_i)]. \tag{3.3.4}$$

with $\widehat{F}$ indicates the empirical distribution that gives probability $\frac{1}{n}$ on each observation $(\mathbf{x}_1^T, y_1), (\mathbf{x}_2^T, y_2), \ldots, (\mathbf{x}_n^T, y_n)$. Therefore, the test point $(\mathbf{x}_0^T, y_0)$ is an observation drawn from the set contains all observations in the learning set $\mathcal{L}$.

However, the training error rate $\overline{err}$ results biased downward estimates for the true error rate Err due to the use of the cases within the learning set for both model fitting and prediction. In addition, the training error rate is also referred as the resubstitution error rate or the apparent error rate.

Test Set Error Rate

To avoid the use of the learning set $\mathcal{L}$ for both model fitting and prediction, we can use another large data set for estimating the true error rate. Let $\mathcal{V}$ be a large sample which is drawn from the same population as the learning set $\mathcal{L}$ is. Let n^* be the number of cases within the sample set $\mathcal{V}$. We express $\mathcal{V} = [(\mathbf{x}_1^{*T}, y_1^*), (\mathbf{x}_2^{*T}, y_2^*), \ldots, (\mathbf{x}_{n^*}^{*T}, y_{n^*}^*)]^T$, where $(\mathbf{x}_1^{*T}, y_1^*), (\mathbf{x}_2^{*T}, y_2^*), \ldots, (\mathbf{x}_{n^*}^{*T}, y_{n^*}^*) \overset{\text{i.i.d}}{\sim} F$. We call this set as a test set.

We can use the test set $\mathcal{V}$ to estimate the true error rate Err for the Classification Trees $\mathcal{R}_\mathcal{L}$ using the test set error rate $\widehat{Err}^{TS}$. We define the test set error rate $\widehat{Err}^{TS}$ as

$$\widehat{Err}^{TS} = \frac{1}{n^*} \sum_{i=1}^{n^*} Q[y_i^*, \mathcal{R}_\mathcal{L}(\mathbf{x}_i^*)]. \tag{3.3.5}$$

Equation 3.3.5 indicates that $\widehat{Err}^{TS}$ can be seen as the proportion of the misclassified cases within the test set that are incorrectly misclassified by the prediction rule $\mathcal{R}_\mathcal{L}$. Compared to the training error estimation, the test set error rate presented in Equation 3.3.5 gives more accurate result. Breiman *et al.* (1984) named this estimate as true misclassification rate estimate.

The test set error rate can be calculated easily if there is a designated test set of large sample size. However, this is not always be the case. It is common that one only has a learning set. As a result of this, to avoid using the learning set for both model fitting and error rate prediction, one might split the available dataset into two disjoint subsets and employ the first subset as the learning set while the second one as the test set. We will discuss this approach in the subsection that follows.

Hold-out Error Rate

We begin this subsubsection by introducing the splitting of the learning set $\mathcal{L}$ into two disjoint subsets that will be used for fitting the model and predicting the true error rate. Let $\mathcal{L}^*$ be the subset of $\mathcal{L}$ used for model fitting and let $\mathcal{V}^*$ be the subset of $\mathcal{L}$ used for error rate prediction. Hence, we can see $\mathcal{L}^*$ as the learning set while $\mathcal{V}^*$ as the test set. Even though there is no any theoretical justification, it is common to allocate $\frac{2}{3}$ of the number of cases within the learning

set $\mathcal{L}$ into the new learning set $\mathcal{L}^*$, while another $\frac{1}{3}$ of the number of cases into the validation set $\mathcal{V}^*$

Unlike, the previous two methods proposed, for Hold-out Error Rate approach, we have to redefine the prediction rule. Instead of having $\mathcal{R}_{\mathcal{L}}$ as the prediction rule fitted by the learning set $\mathcal{L}$, we use $\mathcal{R}_{\mathcal{L}^*}$ to denote the prediction rule fitted by the learning set $\mathcal{L}^*$. Let n_{V^*} be the number of cases within the validation set $\mathcal{V}^*$. We define the Hold-out error rate $\widehat{Err}^{HO}$ as

$$\widehat{Err}^{HO} = \frac{1}{n_{V^*}} \sum_{(\mathbf{x}_i^{*T}, y_i^*) \in \mathcal{V}^*} Q[y_i^*, \mathcal{R}_{\mathcal{L}^*}(\mathbf{x}_i^*)]. \tag{3.3.6}$$

The prediction rule $\mathcal{R}_{\mathcal{L}^*}$ used for calculating $\widehat{Err}^{HO}$ in Equation 3.3.6 is obtained from the new learning set $\mathcal{L}^*$. This new learning set contains only $\frac{2}{3}$ of the number of cases from the learning set $\mathcal{L}$. Hence, the number of cases within the new learning set $\mathcal{L}^*$ is reduced effectively from the number of cases within the learning set $\mathcal{L}$. Because of this, the Hold-out Error Rate $\widehat{Err}^{HO}$ may tend to be biased upward for the true error rate Err in the small learning set case. This estimate also has high variance since it depends upon the observations allocated in either the learning set or the test set. To mitigate the latter issue, Burman (1990) and Witten *et al.* (2016) proposed the Repeated Hold-out estimation.

Using Repeated Hold-out, one has to repeat the entire process including allocating the observations into either the learning or test sets randomly, fitting the Classification Trees and calculating the error rate estimation. Let $\widehat{Err}_m^{HO}$ be the Hold-out estimate for the m-th repetition, for $m = 1, 2, \ldots, M$. We define M-Repeated Hold-out estimation $\widehat{Err}^{rHO}$ as

$$\widehat{Err}^{rHO} = \frac{1}{M} \sum_{m=1}^{M} \widehat{Err}_m^{HO}. \tag{3.3.7}$$

Leave-One-Out Cross-Validation

For Leave-One-Out Cross-Validation (LOOCV), we also split the learning set $\mathcal{L}$ into two subsets, one subset will be used for the training set while the other one will be employed for the test set. Yet, for LOOCV, the test set always contains single case. Let $\mathcal{L}^{(1)}$ be the training set for LOOCV approach and $\mathcal{R}_{\mathcal{L}^{(1)}}$ be the

prediction rule fitted using this learning set. We define LOOCV error rate $\widehat{Err}^{(1)}$ as

$$\widehat{Err}^{(1)} = \frac{1}{n}\sum_{i=1}^{n} Q[y_i, \mathcal{R}_{\mathcal{L}^{(1)}}(\mathbf{x}_i)] \tag{3.3.8}$$

LOOCV is a nearly unbiased estimate for the true error rate. Holding only one instance out, the training set of LOOCV has a chance to produce a prediction rule as accurate as the one obtained using the full dataset. However, LOOCV estimates has high variance if the prediction rule is unstable and it is also requires high computational cost. In addition, Witten *et al.* (2016) did not recommend the use of LOOCV for estimating the error rate of prediction rule involving classification task. LOOCV cannot take the stratification within the classification into account as it always pick one instance as a test point.

v-fold Cross-Validation

We apply the v-fold Cross Validation for estimating the true error rate instead of LOOCV to avoid the high variability of LOOCV estimates. To perform the v-fold Cross Validation, we have to split the learning set $\mathcal{L}$ into v disjoint subsets of equal size to construct v test sets. Let $\mathcal{V}^{(v)}$ be the test set for the v-th splitting and let n_v be the number of cases allocated into the test set $\mathcal{V}^{(v)}$, for $v = 1, 2, \ldots, V$. Let $\mathcal{L}^{(v)} = \mathcal{L} \setminus \mathcal{V}^{(v)}$ be the corresponding learning set, for $v = 1, 2, \ldots, V$. We denote the prediction rule fitted by the learning set $\mathcal{L}^{(v)}$ as $\mathcal{R}_{\mathcal{L}^{(v)}}$

We define the k-fold Cross Validation estimate $\widehat{Err}^{(cv)}$ for the true error rate as

$$\widehat{Err}^{(cv)} = \frac{1}{V}\sum_{v=1}^{V}\frac{1}{n_v}\sum_{(\mathbf{x}_i^{*T}, y_i^*)\in\mathcal{V}^v} Q[y_i, \mathcal{R}_{\mathcal{L}^{(v)}}(\mathbf{x}_i)] \tag{3.3.9}$$

Among others, Breiman & Spector (1992) and Kohavi *et al.* (1995) suggested the use of $v = 5$ or $v = 10$. For v-fold Cross-Validation, smaller v means smaller training set. This in turn yields less accurate prediction rule.

v-fold Cross-Validation gives nearly unbiased estimate for the true error rate. Yet, this estimator corresponds with high variability. Apart from the variation due to the randomness in the training set, the application of v-fold Cross-Validation introduces another kind of variation which is called internal variance.

Efron & Tibshirani (1997) and Braga-Neto & Dougherty (2004) proposed that internal variance is a variation due to random factor other than data sample. For v-fold Cross-Validation, this variation is due to the randomness in partitioning the samples into v fold.

v-fold Stratified Cross Validation

v-fold Cross Validation estimate as in Equation 3.3.9 does not take into account the class information within the learning set $\mathcal{L}$. Kohavi *et al.* (1995) and Witten *et al.* (2016) suggested to take this sort of information into consideration. Ignoring this information might cause the uneven representation both in training and test sets. This produces less accurate prediction rule. Hence, in this section we will discuss v-fold Stratified Cross Validation estimate as an improvement of v-fold Cross Validation estimate which considers classification information.

To take the class information in the learning set into account, we perform the random splitting to produce v disjoint subsets of learning set $\mathcal{L}$ within each class of observation. Let $\tilde{\mathcal{L}}_{(1)}$ and $\tilde{\mathcal{L}}_{(2)}$ be the learning sets contain cases of Class 1 and Class 2 only, respectively. The random allocation to construct k disjoint subsets then is performed upon $\tilde{\mathcal{L}}_{(1)}$ and $\tilde{\mathcal{L}}_{(2)}$.

Let $\tilde{\mathcal{L}}_{(l)}^{(v)}$ and $\tilde{\mathcal{V}}_{(l)}^{(v)}$ be the learning set and the test set for the v-th split upon the learning set of Class l, for $v = 1, 2, \ldots, V$ and $l = 1, 2$. The learning set $\tilde{\mathcal{L}}^{(v)}$ for the v-th split is defined as $\tilde{\mathcal{L}}^{(v)} = \tilde{\mathcal{L}}_{(1)}^{(v)} \cup \tilde{\mathcal{L}}_{(2)}^{(v)}$, while its test set $\tilde{\mathcal{V}}^{(v)}$ is defined as $\tilde{\mathcal{V}}^{(v)} = \tilde{\mathcal{V}}_{(1)}^{(v)} \cup \tilde{\mathcal{V}}_{(2)}^{(v)}$. From this point onwards, we apply the same procedure as the one used for v-fold Cross Validation estimate as in Equation 3.3.9

Let $\mathcal{R}_{\tilde{\mathcal{L}}^{(v)}}$ be the prediction rule fitted by the learning set $\tilde{\mathcal{L}}^{(v)}$ for the v-th fold. We define the v-fold Stratified Cross Validation estimate $\widehat{Err}^{(scv)}$ for the true error rate as

$$\widehat{Err}^{(scv)} = \frac{1}{V} \sum_{v=1}^{V} \frac{1}{n_v} \sum_{(\mathbf{x}_i^{*T}, y_i^*) \in \tilde{\mathcal{V}}^v} Q[y_i, \mathcal{R}_{\tilde{\mathcal{L}}^{(v)}}(\mathbf{x}_i)] \tag{3.3.10}$$

As an improvement of v-fold Cross Validation, v-fold Stratified Cross Validation estimate is less variable than the former. However, this estimate still gives the internal variance as v-fold Cross Validation estimate does.

Repeated v-fold Stratified Cross Validation

In this subsubsection we discuss Repeated v-fold Stratified Cross Validation to reduce the variability in the estimation of the true error rate due to the internal variance. For this approach, we repeat the whole process of partitioning, model fitting and calculating the error rate as we applied for Repeated Hold-out. We calculate the error rate for each repetition using v-fold Stratified Cross Validation as in Equation 3.3.10.

Let $\widehat{Err}_m^{scv}$ be the v-fold Cross Validation estimate of the m-th iteration, for $m = 1, 2, \ldots, M$. We define M-Repeated v-fold Stratified Cross Validation estimation $\widehat{Err}^{rscv}$ as

$$\widehat{Err}^{(rscv)} = \frac{1}{M} \sum_{m=1}^{M} \widehat{Err}_m^{(scv)}. \tag{3.3.11}$$

3.4 Conclusion

To conclude this chapter, among all true error rate estimation methods explained above, we make use only the test set error rate and repeated v-fold stratified cross validation estimations. We use the test set error rate estimates to find the prediction error rate for the fitted Classification Trees of simulated datasets. In this case, we might apply this estimation method since we have both the training sets and the test sets. We use the training set for model fitting and the test sets for model assessment.

Moreover, we use repeated v-fold stratified cross validation estimation to estimate the prediction error rate of Classification Trees of real datasets. We apply this cross-validated estimation method since we only have one learning set such that we have to split it into the training set and the test set.

With regard to the algorithm to built Classification Trees we implement the procedure proposed in Algorithm 3. We name R code that we develop based on this algorithm as **cltrees**. Instead of using **rpart** for fitting the trees, we prefer to use **cltrees** since while splitting a single node **rpart** always picks variable with the lowest index as the classifying variable among several variables of the same maximum goodness of split.

Chapter 4

Classification Trees on High-Dimensional Data

4.1 Introduction

In this chapter we present the application of Classification Tress on real datasets: Smooth CNA and DNACopy CNA datasets. These two datasets consist of CNA profiles of 76 patients whose suffered from lung cancer. Among these patients, 38 of them underwent adeno carcinoma lung cancer, while the other 38 suffered from squamos carcinoma lung cancer. To the best of our knowledge, no one has applied Classification Trees to stratifying cancer subtypes using datasets of CNA estimates.

We apply Classification Trees on these two datasets for two purposes: to build an accurate prediction model and to understand the structure upon these two datasets underlying the resulting classification model. For the former, we construct Classification Trees that could be used to classify a new patient without actual classification into the appropriate cancer subtype stratification. Meanwhile, for the latter, we make use of the fitted Classification Trees to identify set of variables that are responsible for this classification task.

With regard to the data analysis procedure, we begin with discussing the description our datasets in the light of their discriminative information. We then explain the resulting Classifications Trees along with their predictive performance. For computational purpose, we implement rpart package and cltrees.

Moreover, we present the results obtained from constructing Classification Trees using subset of variables. Instead of building Classification Trees using the whole variables, we construct Classification Trees using variables within either one or two genomic regions whose high discriminative information. We implement this approach as an attempt to study whether subsetting variables improves Classification Trees performance in a sense that it increases the trees accuracy level.

Furthermore, we also study the performance of Classification Trees of a single vector which is obtained from scalar projections of each row vector of our real datasets on either their mean or median difference vector. We perform this study as an attempt to apply simple data projection prior to Classification Trees construction.

Finally, in this chapter we present the results attained from the simulation study in which we fit Classification Trees using the generated datasets. For data generating purpose, we implement the simulation settings as described in Section of Chapter 2.

4.2 Application on Real Datasets

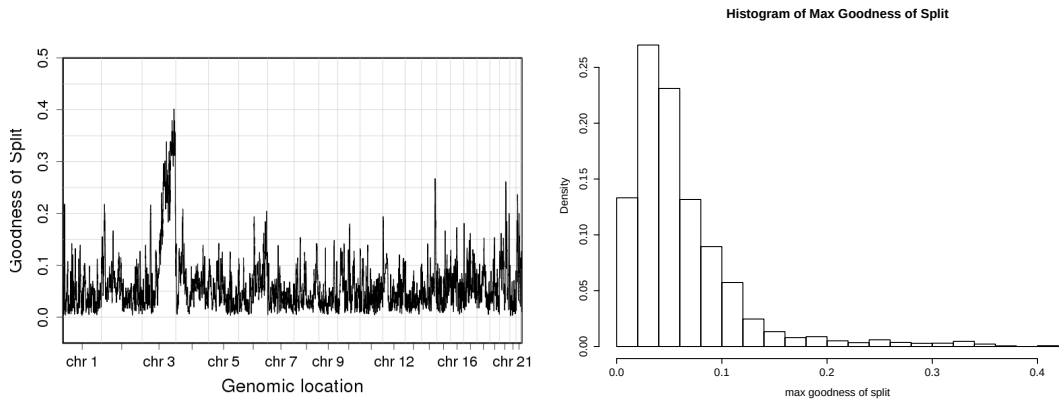
In this section we present the results obtained from applying Classification Trees algorithm on our real datasets: Smooth CNA and DNACopy CNA datasets. We begin by presenting the description of these two datasets in terms of their discriminative information and we continue by constructing Classification Trees using these two datasets along with assessing their predictive performance.

4.2.1 Smooth CNA Dataset

Class Information within Smooth CNA dataset

The following subsubsection moves to describe Smooth CNA dataset in more detail in the light of data classification. We present a measure of discriminative performance of this dataset using a quantity called maximum goodness of split as defined in Equation 3.2.12. This quantity presents the performance of each variable in our dataset to perform data classification task. Figure 4.1 shows both

the line plot of this quantity for the whole variables of our dataset along with the corresponding histogram.



(a) (b)

Figure 4.1: Line plot for maximum goodness of split for individual variables of Smooth CNA dataset along with their histogram. Panel (a) presents line plot for maximum goodness of split for individual variables of Smooth CNA dataset and Panel (b) displays their histogram. We present these maximum goodness of split values to show the performance of each individual variable in carrying classification task. Higher maximum goodness of split indicates better performance of a variable to carry out data classification.

Panel (a) of Figure 4.1 indicates that there is one block of variables within genomic location of Chromosome 3 whose higher maximum goodness of split than the remaining variables. This confirms the result obtained for mean difference as shown in Figure 2.10. There are also several blocks of variables whose maximum goodness of split around 2. This result indicates that these variables also have ability to perform data classification task. However, their performance level is lower than the ones of Chromosome 3.

Panel (b) of Figure 4.1 shows the distribution of maximum goodnes of split across 17571 variables of Smooth CNA dataset. Based on the histogram shown, out of 17571 variables, 63.42 % of them have maximum goodness of split that

less than or equal 0.06. These variables are the ones with few class information. On the hand, there are 3.25% which might be seen as variables with much class information. In addition, we display the relation between mean difference and maximum goodness of split of individual variable using the scatter plot shown in Figure 4.2.

This figure indicates that variables whose high mean difference, around three, also have high maximum goodness of split. Thus, these variables are the ones that have large contribution on performing data classification task. However, this plot also displays the presence of several variables whose mean difference ranges from one up to two which have high maximum goodness of split. These variables spread across several genomic locations. Some of them can be found in Chromosome 3. As mentioned early, these variables contain discriminative information such that they might have contribution on data classification task. Nevertheless, one might not recognise their contribution on fitting Classification Trees. We will discuss the presence of such variables later in this subsection. Yet, one could notice their presence whilst fitting Random Forests on Smooth CNA dataset as in Subsection 7.3.1 of Chapter 7.

Constructing Classification Trees of Smooth CNA dataset

Having discussed the descriptions of Smooth CNA dataset characteristics that are dealing with data classification, we now turn to discuss the application of Classification Tree as a data classification algorithm on Smooth CNA dataset. For computational purpose, we have developed an R code function which is named cltrees.

We implement cltrees in place of rpart package of R since the latter produce bias in selecting variable for splitting an individual node. While splitting an individual node of a tree, rpart always picks variable with the lowest index as the splitting variable in the presence of more than one variable with the same maximum goodness of split.

Function cltrees has one tuning parameter: minimum number of samples in a terminal node ($n.min$). From this point onward, we always construct our Classification Trees by setting $n.min = 10$. In addition, cltrees function makes use

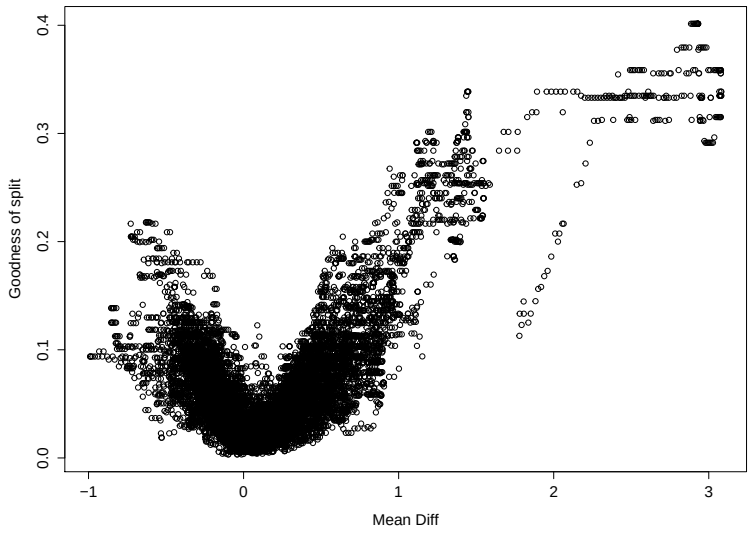


Figure 4.2: Scatterplot of mean difference and maximum goodness of split for each variable of Smooth CNA dataset. The horizontal axis of this plot corresponds with the difference between the mean observation of Class 1 and Class 2. The vertical axis of this plot corresponds with the maximum goodness of split values. Each dot in this plot corresponds with a single variable.

of Gini impurity as the impurity function. The detail algorithm of this function can be found in Section 3.2 of Chapter 3.

Turning now to the fitted Classification Trees of Smooth CNA dataset. Figure 4.3 presents the numerical diagram of the fitted tree. This tree consists of three splittings, three parent nodes and four terminal nodes. Each splitting can be characterised by a classifying variable along with its maximum goodness of split and a threshold or a cut-off that is used for partitioning the samples within a node into two daughter nodes. From Figure 4.3, it can be seen that, the first splitting partitions 76 objects in the root node into two daughter nodes using variable X_{4246} as the classifying variable. This variable is selected randomly from a set of variables which share the same highest maximum goodness of split. It has maximum goodness of split $\Delta imp = 0.4$.

Using this variable, each observation within root node will be classified as object of the left node if $x_{i,4246} < 1.4$ and it will be classified as object of the right node if $x_{i,4246} > 1.4$, for $i = 1, 2, \ldots, 76$. This splitting then produce two daughter nodes which contain 36 and 40 samples, respectively.

Furthermore, the second splitting partitions samples within the parent node of 36 samples using the classifying variable X_{16432}. This splitting variable is also chosen randomly from the set of variables with the highest maximum goodness of split. Unlike the maximum goodness of split for the first splitting, the maximum goodness of split for this second splitting is estimated using the samples fall into the left node which contains 36 observations.

In addition, this splitting also has much lower maximum goodness of split than the first one does. The former has maximum goodness of split $\Delta imp = 0.004$. This second splitting segregates 36 observations within the parent node into two daughter nodes which contains 26 and 10 samples, respectively. Using variable X_{16432}, this second splitting classifies each observation within the parent node into the left node if $x_{i,16432} < 2.18$ and into the right node if $x_{i,16432} > 2.18$.

Moreover, the third splitting partitions samples within the parent node of 40 samples using the classifying variable X_{15241}. This splitting partitions 40 observations within the root node into two daughter nodes which contains 30 and 10 samples, respectively. Using variable X_{115241}, this third splitting classifies each observation within the parent node into the left node if $x_{i,15241} < 1.36$ and into

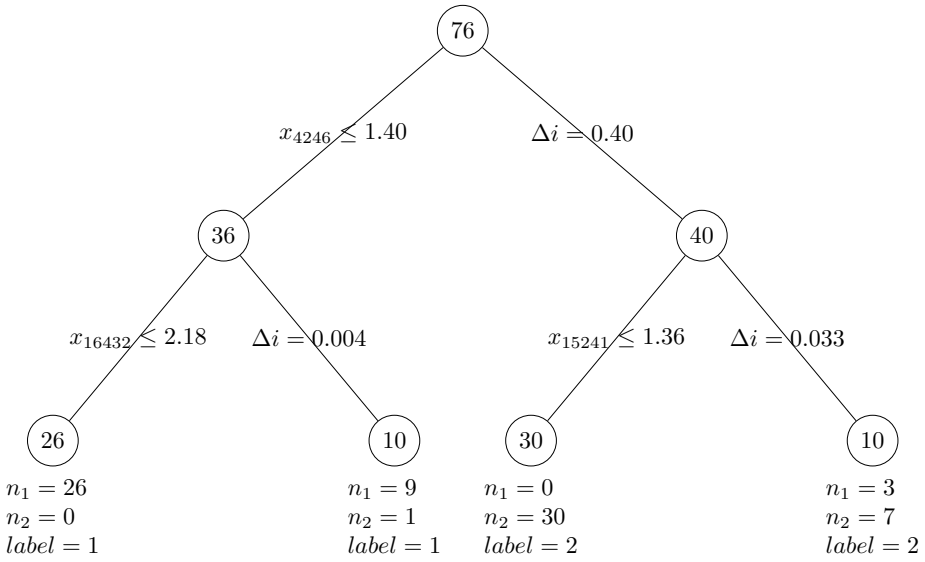


Figure 4.3: Numerical diagram for Classification Trees of Smooth CNA dataset. This diagram shows that Classification Trees of Smooth CNA dataset has three splits and four terminal nodes. Among the three splits, it is only the first split that produces daughter nodes of different class labels. This splitting has variable X_{4246} of Chromosome 3 as the classifying variable.

the right node if $x_{i,15241} > 1.36$. This splitting has maximum goodness of split $\Delta imp = 0.033$.

Contrary to the first splitting, both the second and third splittings produce terminal nodes. The four daughter nodes attained by these two splittings do not split any more as each resulting node either contains observations of the same class or contains less then or equal ten observations. These four terminal nodes then are labelled using the vast majority criterion. Two terminal nodes which contain 26 and 9 observations are labelled as terminal nodes of Class 1. Whereas, the other two terminal nodes which contain 30 and 10 observations are labelled as terminal nodes of Class 2.

This class labelling shows that two terminal nodes that are yielded from segregating the left node of the first splitting are assigned the same class label. Likewise, the two terminal nodes that are obtained from partitioning the right node of the first splitting are also assigned the same class label. This indicates that among the three splittings within the fitted tree of Smooth CNA dataset, it is only the first splitting that is responsible for the classification task has been carried out. This result is related to the performance of each splitting in partitioning the observations within a parent node. The first splitting which has maximum goodness of split $\Delta imp = 0.4$ contributes mostly in classifying the observations into two distinct classes. Meanwhile, both the second and third splittings only partition the observations in order to yield purer daughter nodes.

Having fitted Classification Trees as in Figure 4.3, one can make use the resulting tree for model prediction purpose. Given a measurement on 17571 variables from a new observation without actual class label, one might obtain the predicted label by employing the fitted model. In case one uses Smooth CNA dataset for both model fitting and model prediction, one will end up with four misclassified observations. We refer this number of misclassification as training error as explained in Subsection 3.3.1.

Moreover, Figure 4.3 also shows that the fitted Classification Trees only involves three out of 17571 variables of Smooth CNA dataset. This indicates that for dataset which has much more variables than observations, applying Classification Trees will lead to exploiting only very small number of variables for

classification task. There is no direct contribution of the remaining unused variables for the classification task performed. On the other hand, it is possible that among these unused variables there are some variables that are important for prediction purpose.

With regard to the interpretation of the Classification Trees presented in Figure 4.3 in the genetic point of view, the classifying variable X_{4246} represents a window that consists of 150 kbp which range from 186,750,001 up to 186,900,000 within Chromosome 3. This window is located within locus 3q27.3. As shown by the Classification Trees in Figure 4.3, higher value of this window leads to allocating a patient into squamous carcinoma lung cancer. This result confirms the findings proposed by Brunelli *et al.* (2012), Maier *et al.* (2011) and McGowan *et al.* (2017). They put forward that amplification in genes SOX2 and PIK3CA is common in squamous carcioma lung cancer. These two genes are allocated in loci 3q24 up to 3q27.3. In the light of CNA profile, this amplification leads to the increase in estimated ploidy.

Prediction Error Rate for Classification Trees of Smooth CNA dataset

In this subsubsection we present the results obtained from assessing the predictive ability of Classification Trees fitted using Smooth CNA dataset. We make use of test error rate as a quantity for this assessment. For computational purpose, we implement 100-repeated 10-fold stratified cross validation as explained in Subsection 3.3.1 of Chapter 3.

We display the estimated test error rate through boxplots in Panel (a) of Figure 4.4. This panel presents the error rate of Classification Trees fitted by using both rpart package and cltrees function. This figure also shows boxplot of paired-difference data for the error rate of Classification Trees fitted using these two methods as presented in Panel (b). The difference obtained by subtracting each individual error rate of Classification Trees fitted using rpart and the corresponding error rate of Classification Trees built using cltrees.

Panel (a) of Figure 4.4. indicates that the Classification Trees built using rpart produces higher error rate than the one built using cltrees. Panel (b) of this plot exhibits the same information. Boxplot within this panel indicates that the

are more paired-difference values greater than zero. We obtain this result as rpart always picks variable with the lowest index among several variables which have the same goodness of split. Hence, for Smooth CNA dataset, the application of rpart causes bias.

For inferential purpose, we perform paired sample test to infer whether the Classification Trees fitted using rpart produce higher error rate than those fitted using cltrees. We apply the paired sample test since the error rate of the trees of either cltrees or rpart for each iteration of the repeated stratified cross validation carried out is yielded using the same training and test sets.

Having obtained the paired-difference data of the error rate estimates from both Classification Trees fitted using rpart and cltrees, the normality test of Shapiro-Wilk was performed using shapiro.test() function in R. For this comparison, this test gives p-value which is much less than 0.05. Hence, for significance level $\alpha = 5\%$, it can be concluded that the paired-difference data are not normally distributed. As a result, paired sample Wilcoxon Test was applied as an alternative to paired t-test to compare the error rates obtained from Classification Trees built using both methods mentioned above.

Paired sample Wilcoxon Test was performed using wilcox.test() function. The null hypothesis for this test is H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero. This test produces p-value equals 0.00114. Therefore, for significance level $\alpha = 5\%$, it could be inferred that the null hypothesis H_0 that the median of the difference between the pairs equals zero is being rejected. To put it in another way, it can be concluded that the trees fitted by rpart produce higher error rates than the ones developed by cltrees.

This confirms the results shown in Figure 4.4. that by applying rpart one ends up with trees of higher error rate for Smooth CNA dataset. However, applying cltrees to fit a single Classification Tree takes longer time than utilizing rpart. rpart needs 0.57247 second to fit a tree, whereas cltrees requires 16.69539 second.

With regards to the results obtained from performing inferential task to compare all pairs of algorithms and computational procedures within this thesis, the complete list of significances of Shapiro Wilk test for normality of the resulting pair difference data is presented in Appendix A. Meanwhile, the complete list of

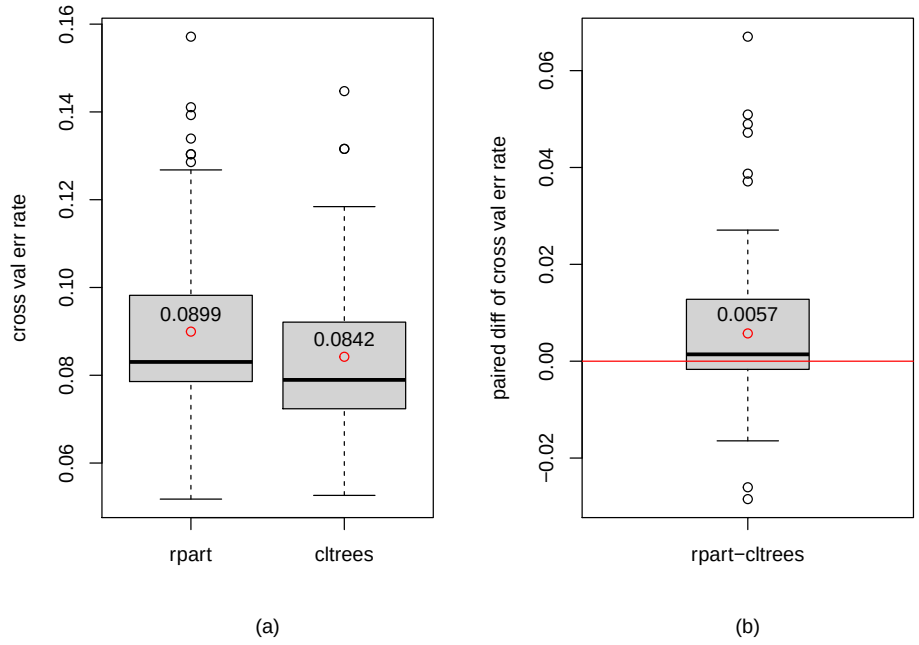


Figure 4.4: Boxplots for misclassification rate of Classification Trees obtained using cltrees and rpart for Smooth CNA dataset along with their paired difference data. Panel (a) shows boxplots of the error rate estimates of Classification Trees fitted using rpart (the left boxplot) and the error rate estimates of Classification Trees fitted using cltrees (the right boxplot). Both boxplots represent 100 estimates of Classification Trees error rate obtained using 100-repeated 10-fold cross validation estimation. Panel (b) displays paired-difference data between these two sets of the error rate estimates. The red horizontal line in this panel corresponds with paired-difference equals 0. Using paired sample Wilcoxon Test with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero, we end up with p-value 0.00114. Hence, it can be concluded that rpart produces Classification Trees with higher error rates than the ones developed by cltrees.

significances of two-sided paired sample Wilcoxon Test is shown in Appendix B and the one of one-sided paired sample Wilcoxon Test is presented in Appendix C.

Apart from the prediction error rate presented in Figure 4.4, it is worth noting that the resulting Classification Tree of Smooth CNA dataset shown in Figure 4.3 has variable X_{4246} as the classifying variable of the root node. This variable is picked randomly from the set of variables whose the highest maximum goodness of split for the root node. It is also the only variable within the resulting tree that partitions the observations of a parent node into two daughter nodes with different class labels. However, variable X_{4246}, along with the other variables that belong to the set of those which have highest maximum goodness of split, does not only have high maximum goodness of split but also has high variance. This in turn affect their performance in carrying out data classification task. The simulation study explained in Section 4.5 confirms this result.

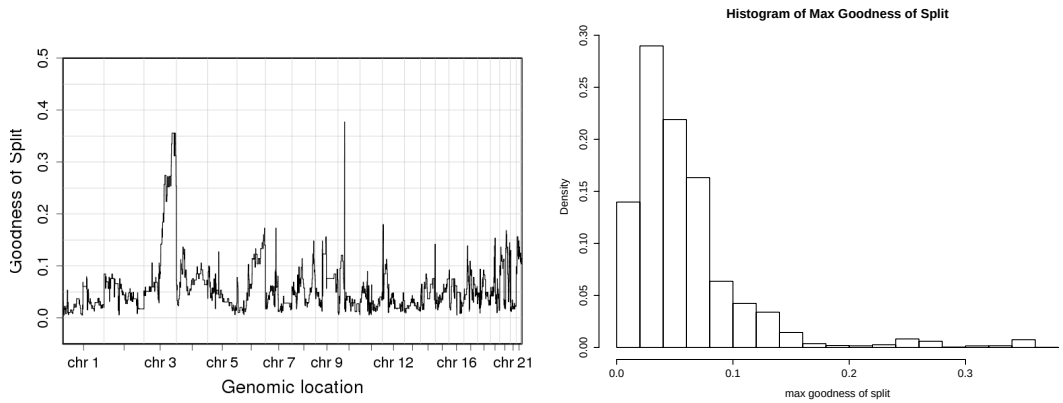
In addition, having smaller number of observations, less than 100 observations, also affects the performance of 100-repeated stratified cross validation estimation that we use for estimating the true error rate. Small sample size corresponds with the unstable estimates of cross-validated error rate. This in turn leads to upward-biased estimates.

4.2.2 DNACopy CNA Dataset

Class Information within DNACopy CNA dataset

In subsection that follows we describe DNACopy CNA dataset properties regarding to data classification in more detail. We begin with presenting the plot of maximum goodness of split for each individual variable of DNACopy CNA dataset along with its histogram in Figure 4.5. We make use of this quantity to measure the performance of a variable in carrying data classification.

Panel (a) of Figure 4.5 indicates that there are two blocks of variables within genomic locations of Chromosomes 3 and 10, respectively, whose higher maximum goodness of split than the remaining variables. They also belong to the set of variables whose high absolute mean difference. However, both blocks of variables classify the observations from distinct groups in opposite directions.



(a) (b)

Figure 4.5: Line plot for maximum goodness of split for individual variables of DNACopy CNA dataset along with their histogram. Panel (a) presents line plot for maximum goodness of split for individual variables of DNACopy CNA dataset and Panel (b) displays their histogram. We present these maximum goodness of split values to show the performance of each individual variable in carrying classification task.

Those variables of Chromosome 3 classify the observations into Class 1 for small measurements. Meanwhile those of Chromosome 10 classify the observations into Class 1 for large measurements. This can be noticed from the direction of mean difference as shown in Figure 2.19.

There are also several blocks of variables whose maximum goodness of split ranges from 1 up to 2. This indicates that these variables also have ability to perform data classification task. However, their performance level is lower than the ones of Chromosomes 3 and 10. Surprisingly, among these variables there are some variables whose large absolute mean difference as presented in Figure 4.6.

Furthermore, Panel (b) of Figure 4.5 shows the distribution of maximum goodnes of split across 18388 variables of DNACopy CNA dataset. Based on the histogram shown, out of 18388 variables, 64.84 % of them have maximum goodness of split that less than or equal 0.06. These variables are the ones with few class information. On contrast, there are 3.43% which might be seen as variables with much class information.

Turning now to discuss the relation between mean difference and maximum goodness of split as shown in Figure 4.6, one might easily find that there are variables whose large absolute mean difference but they have low maximum goodness of split. This indicates that those variables produce large absolute mean difference due to the presence of few extreme observations. Hence, for the maximum goodness split which is based on the order of observations instead of their values, one ends up with low maximum goodness of split. Therefore, these variables do not affect Classification Trees construction as much as the other variables whose high maximum goodness of split. However, those of variables whose large mean difference along with small maximum goodness of split do influence the performance of principal component scores of DNACopy CNA dataset while carrying out data classification task. We are going to discuss this in more detail at Subsection 5.4.2 of Chapter 5.

Constructing Classification Trees of DNACopy CNA dataset

Having discussed the descriptions of DNACopy CNA dataset that are dealing with its performance in carrying out data classification, it is now we turn to

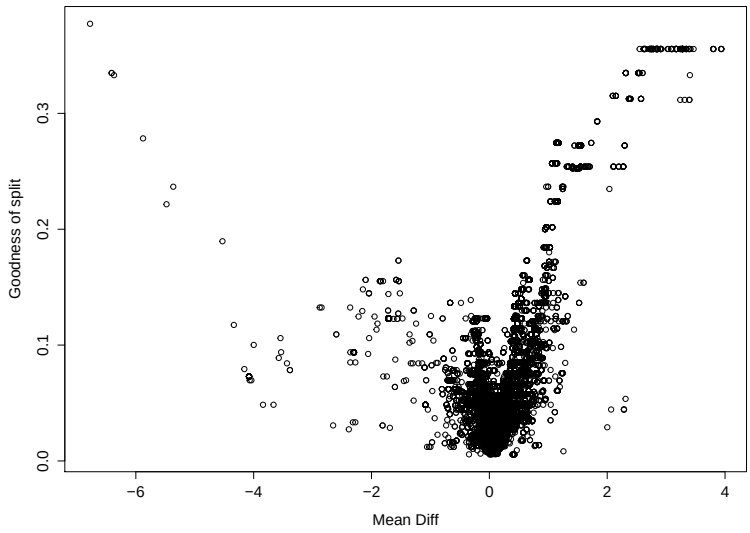


Figure 4.6: Scatterplot of mean difference and maximum goodness of split for each variable of DNACopy CNA dataset. The horizontal axis of this plot corresponds with the difference between the mean observation of Class 1 and Class 2.The vertical axis of this plot corresponds with the maximum goodness of split values. Each dot in this plot corresponds with a single variable.

present the application of Classification Tree as a data classification algorithm on DNACopy CNA dataset. For computational purpose, we implement cltrees function along with rpart.

Figure 4.7 displays the numerical diagram of Classification Trees fitted by DNACopy dataset using cltrees function. This tree consists of three splittings, three parent nodes and four terminal nodes. The first splitting partitions 76 objects in the root node into two daughter nodes using variable X_{11284} as the classifying variable with maximum goodness of split $\Delta imp = 0.38$. This variable is the one from genomic location of Chromosome 10.

Furthermore, the second and the third splittings partition the nodes obtained by the first splitting into four terminal nodes. These terminal nodes are assigned the class labels in the same vein as the terminal nodes of Classification Trees of Smooth CNA dataset. Two terminal nodes obtained from partitioning the left node of the first splitting are assigned as Class 2. Meanwhile, the two terminal nodes that are attained from segregating the right node of the first splitting are also assigned as Class 1.

With regard to the classifying variable of the first split, as previously mentioned, variable X_{11284} of Chromosome 10 is chosen. In contrast, there is no variable of Chromosome 3 that is picked as the classifying variable although they have high maximum goodness of split. This result exhibits how constructing Classification Trees using dataset whose much more variables than observations might lead to leaving some variables whose high maximum goodness of split unused. For this reason, we are going to apply data dimension reduction approach by either variable selection or feature extraction prior to Classification Trees construction.

Furthermore, with respect to the interpretation of the Classification Trees presented in Figure 4.7 in the genetic point of view, the classifying variable $X11284$ represents a window that consists of 150 kbp which range from 39,150,001 up to 39,300,000 within Chromosome 10. This window is located within loci 10p11.22 and 10q11.21. As shown by the Classification Trees in Figure 4.7, higher value of this window leads to allocating a patient into the adeno carcinoma lung cancer. This result confirms the findings proposed by Kohno *et al.* (2012), Ju *et al.* (2012), Yokota *et al.* (2012) and Popper *et al.* (2014). They put forward that fusion between the neighbouring gene KIF5B in locus 10p11.22 and gene RET in

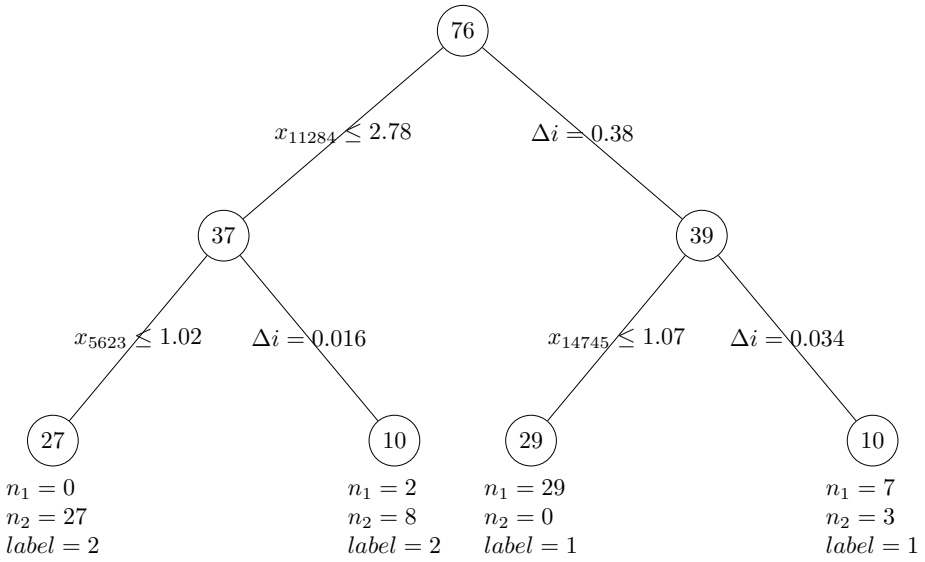


Figure 4.7: Numerical diagram for Classification Trees of DNACopy CNA dataset. This diagram shows that Classification Trees of DNACopy CNA dataset has three splits and four terminal nodes. Among the three splits, it is only the first split that produces daughter nodes of different class labels. This splitting has variable X_{11284} of Chromosome 10 as the classifying variable.

locus 10q11.21 is common in adeno carcinoma lung cancer. In the light of CNA profile, this gene fusion leads to the increase in estimated ploidy.

Moreover, for DNACopy dataset, there are also other windows in Chromosome 3 which can be used to stratify the patients into two subtypes of lung cancer, adeno carcinoma and squamous carcinoma, as for Smooth CAN dataset. These windows are marked by the amplification of genes SOX2 and PIK3CA which are located in loci 3q24 up to 3q27.3. This amplification is responsible for the occurrence of squamous carcinoma lung cancer. However, for DNACopy CNA dataset, the discriminative power of these windows while fitting the Classification Trees is overcome by windows within Chromosome 10. Yet, we are going to recognize their contribution in performing data classification while applying Random Forests in Chapter 8.

In addition, for Smooth CNA dataset, the resulting Classification Trees indicate the absence for the contribution of windows within loci 10p11.22 and 10q11.21 upon the trees construction. We exclude these windows from Smooth CNA dataset during data cleaning since there are more than ones missing values for these windows.

Prediction Error Rate for Classification Trees of DNACopy CNA dataset

In this subsubsection we present the results obtained from assessing the predictive ability of Classification Trees fitted using DNACopy CNA dataset. We make use of test error rate as a quantity for this assessment. For computational purpose, we implement 100-repeated 10-fold stratified cross validation.

We display the estimated test error rate in boxplots of Panel (a) within Figure 4.8. These results are obtained from using using both rpart package and cltrees function. This figure also shows boxplot of paired-difference data for error rate of Classification Trees fitted using these two methods as presented in Panel (b). The difference obtained by subtracting each individual error rate of Classification Trees fitted using rpart and the corresponding error rate of Classification Trees built using cltrees.

Panel (a) of Figure 4.8. indicates that the Classification Trees built using rpart produces lower error rate estimates than the one built using cltrees. Panel

(b) of this plot confirms this result. However, the latter yields the estimate with larger variance. This indicates that using rpart to construct Classification Trees for DNACopy CNA dataset leads to larger estimation instability.

To make an inference on the paired comparison between the resulting error rates of Classification Trees fitted using either rpart or cltrees for DNACopy CNA dataset, we apply the same procedure as we did for Smooth CNA dataset. We begin by performing Shapiro Wilk test of normality for the paired difference data and keep up by carrying out paired sample Wilcoxon Test.

We apply wilcox.test() function with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero. We use this alternative hypothesis since we wish to check whether the trees fitted by rpart yield greater error rates than the ones fitted by cltrees as previously mentioned. We obtain p-value equals 1 from this hypothesis testing. Hence, it can be inferred that, for DNACopy CNA dataset, the trees fitted by rpart does not produce higher error rate than the ones fitted by cltrees. This confirms the results shown in Figure 4.8. In addition, rpart needs 0.41888 second to fit a tree, whereas cltrees requires 16.83186 second.

Furthermore, for Smooth CNA dataset we obtain Classification Tree which has the classifying variable is the one with high variance. Likewise, for DNACopy CNA dataset we also attain the tree which has the classifying variable is the one with high variance as well. However, in the light of maximum goodness of split, the latter dataset has two blocks of variables whose high maximum goodness of split. These variables are within genomic locations of Chromosomes 3 and 10. The one which is picked as the classifying variable X_{11284} belong to Chromosome 10. Returning to high data variation, variables within these two genomic locations both have high variance. This in turn influences their performance in carrying out data classification task.

Moreover, having two blocks of variables whose high maximum goodness of split does not give advantage in terms of lowering the resulting Classification Trees error rate. Instead, in the presence of variables from both genomic locations, we end up with higher error rate. We discuss this in more detail in the following section.

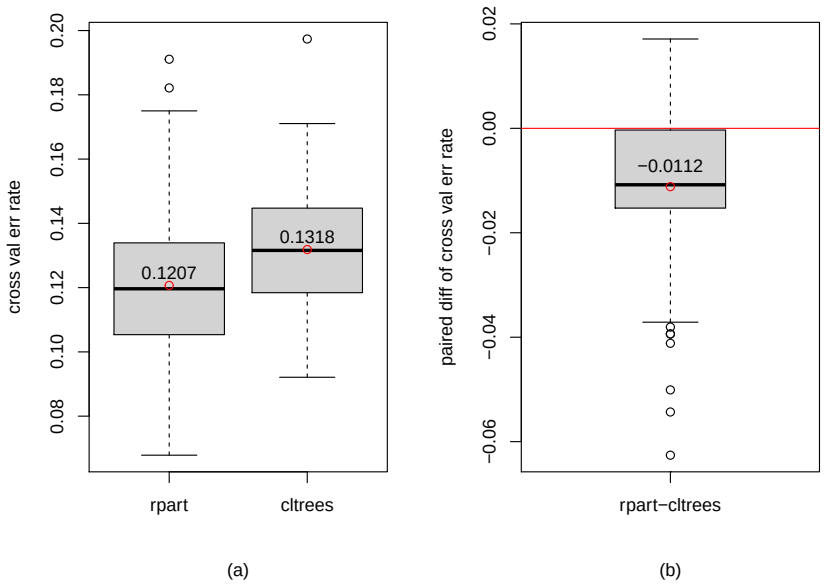


Figure 4.8: Boxplots for misclassification rate of Classification Trees obtained using cltrees and rpart for DNACopy CNA dataset along with their paired difference data. Panel (a) shows boxplots of the error rate estimates of Classification Trees fitted using rpart (the left boxplot) and the error rate estimates of Classification Trees fitted using cltrees (the right boxplot). Both boxplots represent 100 estimates of Classification Trees error rate obtained using 100-repeated 10-fold cross validation estimation. Panel (b) displays paired-difference data between these two sets of the error rate estimates. The red horizontal line in this panel corresponds with paired-difference equals 0. Using paired sample Wilcoxon Test with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero, we end up with p-value equals 1. Hence, it can be concluded that rpart does not produce Classification Trees with higher error rates than the ones developed by cltrees.

4.3 Classification Trees and Variable Selection

Turning now consider applying variable selection step prior to fitting Classification Trees on both Smooth CNA and DNACopy CNA datasets. These two datasets have tens of thousands of variables.

We apply variable selection preceding Classification Trees construction as an effort to better the performance of the resulting trees. By utilizing variable selection step, we pick up only the informative variables to fit these trees and leave the irrelevant variables aside. In this case, we make use of both mean difference and maximum goodness of split for each individual variable as the measures used while doing variable selection.

By considering these two quantities, for Smooth CNA dataset, we select variables of Chromosome 3 and fit the tree using this subset of variables. Meanwhile, for DNACopy CNA dataset, we choose variables of Chromosomes 3 and 10. We build the trees using either variables of Chromosomes 3 or 10 or both chromosomes.

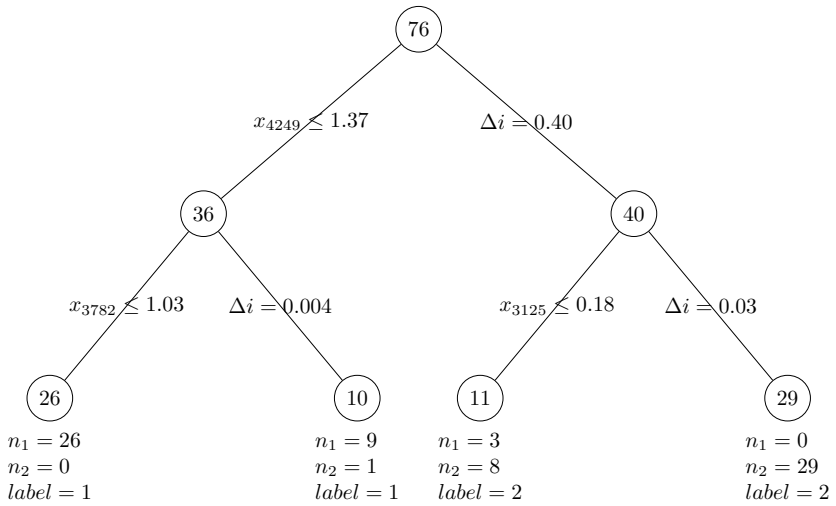


Figure 4.9: Numerical diagram for Classification Trees fitted using variables of Chromosome 3 for Smooth CNA dataset.

Let us now consider the Classification Trees fitted using variables of Chromosome 3 of Smooth CNA dataset. Figure 4.9 shows the numerical diagram of this

Classification Tree. It is apparent from this figure that the Classification Trees obtained using variables of Chromosome 3 and the trees attained using the whole variables as shown in Figure 4.3 share many features in common. Classification Trees fitted using the whole genomes also have a single variable within genomic location of Chromosome 3 as the classifying variable. In addition, the Classification Trees fitted using either the whole genomes or genomes of Chromosome 3 produce four terminal nodes whose the same class labels.

Furthermore, Figure 4.10 presents two panels, Panel (a) and Panel (b). Panel (a) shows two boxplots of prediction error rate estimates for these two trees. Panels (b) displays boxplot of paired difference data between the error rates of Classification Trees fitted using the whole variables and those of tres built using the variables of Chromosome 3. Both panels indicates the similarity between their error rates.

To make an inference on this result, we perform paired sample Wilcoxon Test. For this hypothesis testing, we have null hypothesis H_0: median of difference between the pairs equals zero and two-sided alternative hypothesis H_1: median of difference between the pairs is not zero. We obtain p-value 0.3222 from this hypothesis testing. Hence, it can be inferred that, fitting Classification Trees using variables within genomic region of Chromosome 3 does not produce trees with lower error rate than fitting trees using the whole variables. Instead, we obtain the trees with the same level of accuracy. This confirms the results shown in Figure 4.10.

Moreover, the above result may be explained by the fact that while fitting Classification Trees using the whole variables, we obtain block of variables of Chromosome 3 as the only block of variables with highest maximum goodness of split. As a result of this, constructing Classification Trees using variables of Chromosome 3 leads to the same trees as using the whole variables.

Moving on now to consider the result for DNACopy CNA dataset, we present numerical diagrams for Classification Trees fitted using variables of Chromosomes 3, 10 and both Chromosomes 3 and 10 in Figures 4.11, 4.12 and 4.13, respectively. In addition, we display Figure 4.14 to compare the prediction error rate for these three Classification Trees along with the tree fitted using the whole variables. We present four boxplots of 100-repeated 10-fold stratified cross validation estimates

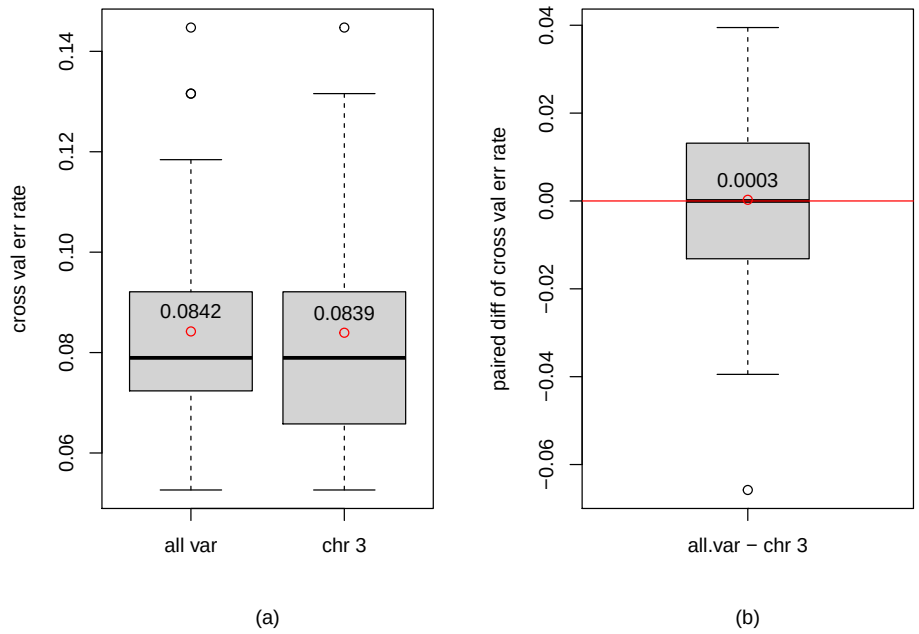


Figure 4.10: Boxplots for comparing the accuracy of Classification Trees fitted using all variables and variables of Chromosome 3 along with their paired difference data for Smooth CNA dataset. Panel (a) shows boxplots of the error rate estimates of Classification Trees fitted using all variables (the left boxplot) and the error rate estimates of Classification Trees fitted using variables within Chromosome 3 (the right boxplot). Both boxplots represent 100 estimates of Classification Trees error rate obtained using 100-repeated 10-fold cross validation estimation Panel (b) displays paired-difference data between these two sets of error rate estimates. The red horizontal line in this panel corresponds with paired-difference equals 0. Using paired sample Wilcoxon Test with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero, we end up with p-value equals 0.322. Hence, it can be concluded that fitting Classification Trees using variables of Chromosome 3 does not produce the trees with lower error rates.

for this comparison purpose. These four boxplots can be found in Panel (a) of this figure. We also present three boxplots of paired-difference data in Panel (b) of this figure. Each boxplot presents paired-difference data obtained from subtracting error rates of Classification Trees fitted using the whole variables and error rates of trees built using either the variables of Chromosome 3, Chromosome 10 or both Chromosomes 3 and 10.

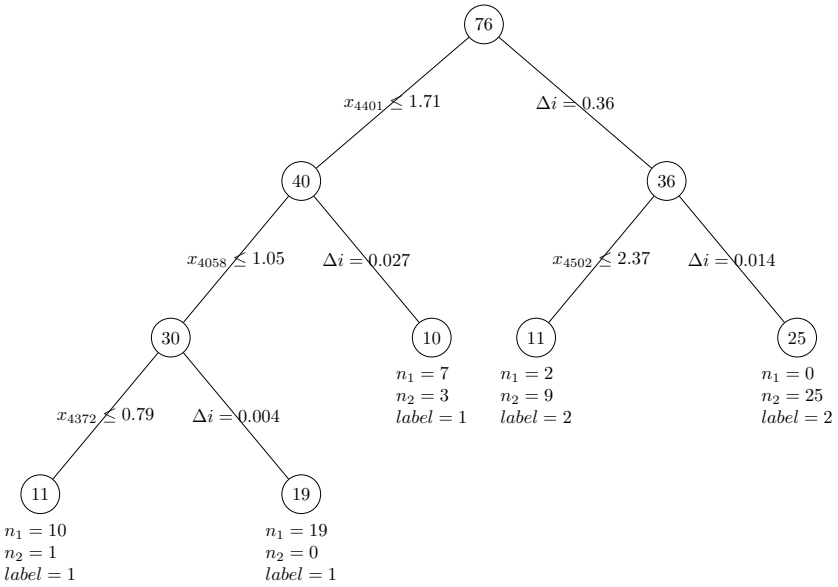


Figure 4.11: Numerical diagram for Classification Trees fitted using variables of Chromosome 3 for DNACopy CNA dataset.

With regard to the resulting Classification Trees, it is obvious from Figures 4.11, 4.12 and 4.13 that fitting Classification Trees using variables of Chromosome 10 produce the same trees as fitting the one using variables of both Chromosomes 3 and 10. The resulting Classification Trees for these two subsets of variables even share many features in common with the tree built using the whole variables as shown in Figure 4.7. On contrast, constructing Classification Trees using variables of Chromosome 3, one ends up with different result.

However, in terms of the prediction error rate, it is only Classification Trees fitted using variables of both Chromosomes 3 and 10 that produce the same accuracy level as the trees fitted the whole variables. Surprisingly, fitting Classification

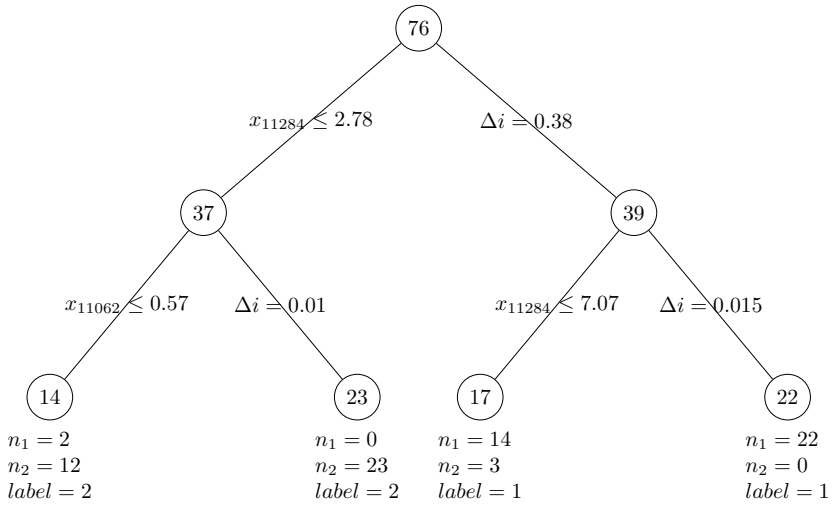


Figure 4.12: Numerical diagram for Classification Trees fitted using variables of Chromosome 10 for DNACopy CNA dataset.

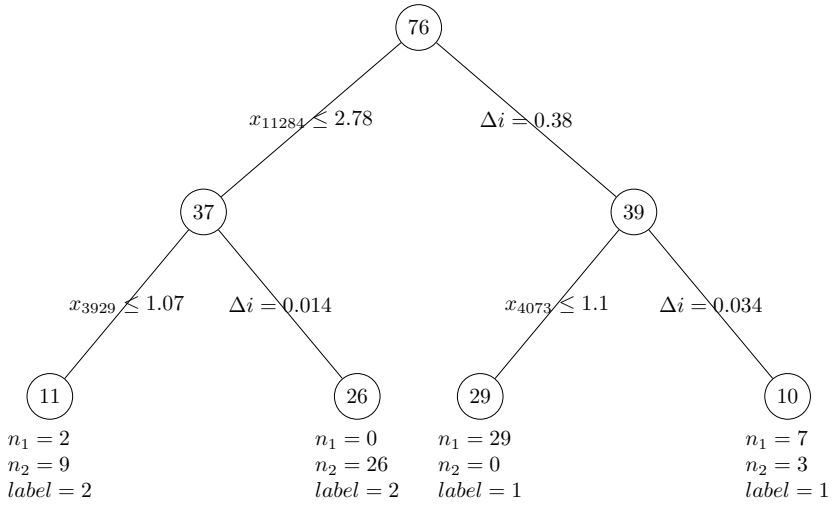


Figure 4.13: Numerical diagram for Classification Trees fitted using variables of Chromosomes 3 and 10 for DNACopy CNA dataset.

Trees using variables of Chromosome 10, one obtains Classification Trees whose much lower error rate estimates than the error rate estimates for Classification Trees fitted using the whole variables. Interestingly, the cross validated estimates for Classification Trees of variables from this Chromosome 10 approximate OOB error rate of Random Forest built using the whole variables of this dataset. We discuss this result in Subsection 7.3.2 of Chapter 7. In addition, constructing Classification Trees using variables of Chromosome 3, one ends up with the more accurate trees than using the whole variables.

Furthermore,Table 4.1 presents the results obtained from performing Shapiro-Wilk test of normality and paired sample Wilcoxon test. These inferential tasks are carried out to make inferences on comparing the resulting error rates for Classification Trees fitted using all variables of DNACopy CNA dataset with those obtained from the trees fitted using variables of either Chromosome 3, Chromosome 10 or both Chromosomes 3 and 10.

Table 4.1: P-values for Shapiro-Wilk test and Wilcoxon test for the comparison of error rates obtained from Classification Trees fitted using the whole genomes and variables of Chromosomes 3 and 10 of DNACopy CNA dataset

Comparison	Shapiro-Wilk test	Wilcoxon test
All chr - Chr 3	0.00024	3.09764×10^{-15}
All chr - Chr 10	0.00030	1.43999×10^{-18}
All chr - Chr 3 & 10	1.17815×10^{-15}	0.14527

For the three paired sample Wilcoxon tests carried out, we define the null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero. Table 4.1 shows that all the paired difference data are not normally distributed. Moreover, this table also indicates that for the paired difference of error rates of Classification Trees fitted using all chromosomes and those built using Chromosome 3, we reject the null hypothesis that the median of the difference between the pairs equals zero. To put it in another way, it can be concluded that the trees fitted by Chromosome 3 produce lower error rates than the ones

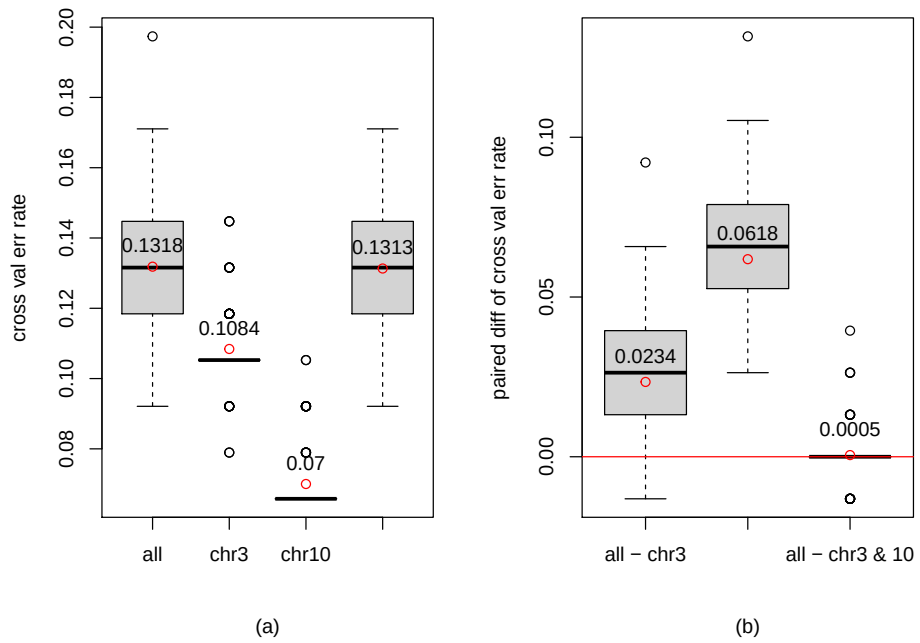


Figure 4.14: Boxplots for comparing the accuracy of Classification Trees fitted using all variables, variables of Chromosome 3, variables of Chromosome 10 and variables of both Chromosomes 3 and 10 with their paired difference data for DNACopy CNA dataset. Panel (a) shows boxplots of the error rate estimates of Classification Trees fitted using all variables, the variables of Chromosome 3, the variables of Chromosome 10 and the variables of both Chromosomes 3 and 10. Panel (b) displays three boxplots for paired-difference data between the error rates of Classification Trees fitted using the whole variables and those of trees built using the variables of Chromosome 3, the variables of Chromosome 10 and the variables of both Chromosomes 3 and 10. The red horizontal line in this panel corresponds with paired-difference equals 0. Using paired sample Wilcoxon Test with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero, we end up with p-value equals 3.1e-15 for the comparison of the trees fitted using the whole variables and Chromosome 3, 1.44e-18 for the trees fitted using the whole variables and Chromosome 10 and 1.45e-01 for the trees fitted using the whole variables and both Chromosomes 3 and 10.

developed by the entire chromosome. The same result is also attained for the pair of all chromosome and Chromosome 10.

On contrast, for the pair of all chromosome and Chromosomes 3 and 10, from Table 4.1 it can be inferred that we fail to reject H_0 that the median of the difference between the pairs equals zero. Therefore, it can be concluded that the trees fitted using the entire chromosome share the same level of accuracy with those fitted using Chromosomes 3 and 10.

Returning to Figure 4.14, it is apparent that fitting Classification Trees using variables of both Chromosomes 3 and 10 gives higher prediction error rate than fitting the trees using variables of either one of these two chromosomes. It is also obvious from this figure that fitting Classification Trees using variables of either Chromosomes 3 or 10, one ends up with Classification Trees whose the estimates of cross validated error rate with much lower variance. Constructing Classification Trees using variables of either one of these two chromosomes, one obtains much more stable cross validated estimates for the resulting Classification Trees.

To figure out this result, we select one variable with the highest maximum goodness of split from either Chromosome 3 or 10. For Chromosome 3, we pick variable X_{4369}. This variable has 0.3556 maximum goodness of split. For Chromosome 10, we choose variable X_{11284}. This variable has 0.3773 maximum goodness of split. Using these two variables we fit Classification Trees and perform model assessment by carry out 100-repeated 10-fold stratified cross validation estimation. In addition, we also build Classification Trees using either one of these two variables and calculate the prediction error. Figure 4.15 shows the boxplots for the resulting error rates for the three Classification Trees constructed. This figure confirms the result obtained by using the whole variables of either Chromosome 3 or 10 as presented in Figure 4.14.

For further investigation, we perform a simulation study by taking the mean and covariance structures of both variables X_{4369} and X_{11284} into consideration. In this case, we wish to investigate whether the number of samples affects the application of cross validated estimation to obtain the prediction error since we have to split the learning set into the training and test sets. There are three settings of sample size that we consider here: small ($n = 100$), medium ($n =$

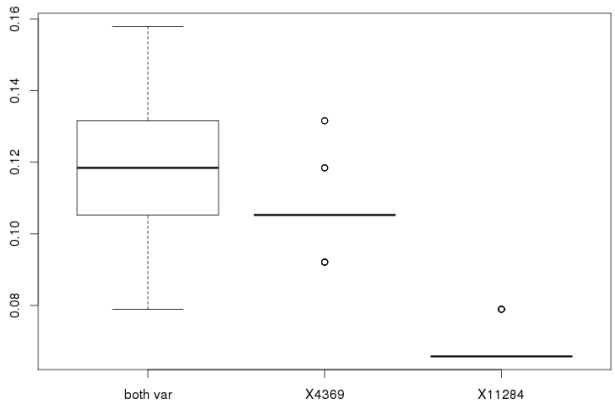


Figure 4.15: Boxplots for comparing the accuracy of Classification Trees fitted using two variables, X_{4369} and X_{11284} and one out of these two variables. Variable X_{4369} is picked from Chromosome 3 of DNACopy CNA dataset as it has high maximum goodness of split. Variable X_{11284} is choosen from Chromosome 10 of DNACopy CNA dataset for the same reason. We pick these two variables to check whether fitting Classification Trees using two variable from either Chromosomes 3 or 10 produces Classification Trees of the same cross validated error rate estimatimates as those of fitted using the whole variables of either one of these two chromosomes. Each boxplot displays the estimate for 100-repeated stratified cross-validation error rate.

1000) and large ($n = 10000$). Since we are dealing with the case of balanced classification, so we set the number of samples from each class of observations as half of the total number of samples.

Moreover, for population setting, we generate bivariate random samples from two bivariate normal subpopulations. The first subpopulation is the one which has

$$\boldsymbol{\mu}^{(1)} = (1.0218, 8.2422)^T$$

and

$$\Sigma^{(1)} = \begin{pmatrix} 0.2234 & -0.0294 \\ -0.0294 & 21.5202 \end{pmatrix}.$$

Meanwhile, the second subpopulation is the one which has

$$\boldsymbol{\mu}^{(2)} = (3.5721, 1.4655)^T$$

and

$$\Sigma^{(2)} = \begin{pmatrix} 6.6163 & 1.8407 \\ 1.8407 & 2.0476 \end{pmatrix}.$$

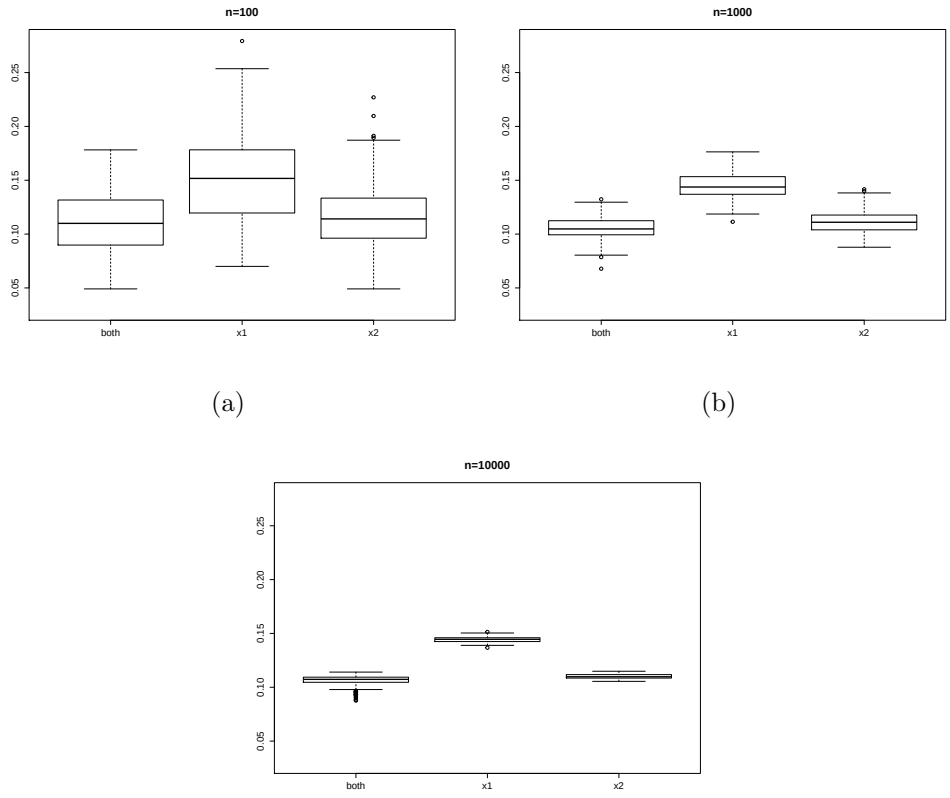
Next, for the data generating procedure, we define data matrix X as an $n \times 2$ matrix that consists of 100 samples and 2 variables. Hence, we define X as

$$\underset{n\times 2}{\mathrm{X}} = \begin{bmatrix} \underset{n_1\times 2}{\mathrm{X}^{(1)}} \\ \hline \underset{n_2\times 2}{\mathrm{X}^{(2)}} \end{bmatrix}$$

where $\underset{n_1\times 2}{\mathrm{X}^{(1)}}$ is data matrix of n_1 samples drawn from sub-population 1 which follows $N(\mu^{(\mathbf{1})}, \Sigma^{(1)})$ distribution and $\underset{n_2\times 2}{\mathrm{X}^{(2)}}$ is data matrix of n_2 samples drawn from sub-population 2 which follows $N(\mu^{(\mathbf{2})}, \Sigma^{(2)})$ distribution. In addition, for the categorical variable $\mathbf{y}$, we set its first n_1 values as 1 while another n_2 as 2.

We present the result of the simulation for this subsection as in Figure 4.16. We obtain this figure from applying 100-repeated 10-fold stratified cross validation estimation for the three simulation settings mentioned above. It should be emphasized that, for this simulation study, instead of applying the test set error rate estimation, we make use of the cross validated approach to estimate the true error rate estimate. We make use of the latter as we wish to investigate the performance of cross validated estimation rather than to perform model assessment.

Furthermore, it is apparent from this Figure 4.16 that performing cross-validated estimation for the learning set with small sample size leads to the unstable prediction error rate estimation. This figure also shows that fitting Classification Trees using two variables of the same settings as variables X_{4369} and X_{11284}, we end up with the result that is the error rate of Classification Trees fitted using two variables is equal to the error rate of Classification Trees fitted using the more informative between these two variables. The findings clearly indicate that the effect shown in Figures 4.15 and 4.15 is a result of the random fluctuation in the dataset.



(a) (b)

(c)

Figure 4.16: Boxplots for the error rates of Classification Trees fitted using simulated datasets for checking the effect of sample size on cross-validated estimation. The top left panel corresponds to the resulting cross validated error rates obtained from Classification Trees fitted using simulated datasets with 100 samples. The top right panel corresponds to the resulting cross validated error rates obtained from Classification Trees fitted using simulated datasets with 1000 samples. The bottom panel corresponds to the resulting cross validated error rates obtained from Classification Trees fitted using simulated datasets with 10000 samples. Each panel consists of three boxplots. The left boxplot is the one for the case of fitting trees using two variables, meanwhile the other two boxplots is the one for the case of fitting trees using either one of these two variables.

4.4 Classification Trees of Mean Difference and Median Difference Projected Datasets

In this section we discuss the results obtained from constructing Classification Trees using the scalar projection of data matrix on either its mean difference or its median difference. We implement this approach as an effort to improve Classification Trees performance in the light of prediction error by reducing the data dimension. As previously explained, one might make use of mean difference to state the class information contained in dataset as it represents the difference between central tendency of each group of observations. Likewise, one also might apply median difference for the same reason.

Let X be an $n \times p$ data matrix that consists of n measurements of p-variate random vector which has a distribution as a mixture of two multivariate distributions. We call the samples drawn the sub-population with the first distribution as observations of Class 1 and the ones drawn from the sub-population with the second distribution as observations of Class 2. Let n_1 and n_2 be the number of samples drawn from the sub-populations with the first distribution and the second distribution, respectively.

Let $\overline{\mathbf{X}}^{(1)}$ and $\overline{\mathbf{X}}^{(2)}$ be vectors of mean of observations from Class 1 and Class 2, respectively. We define $\overline{\mathbf{X}}^{(l)}$, for $l = 1, 2$ as

$$\overline{\mathbf{X}}^{(l)} = \frac{1}{n_l} \sum_{i=1}^{n_l} \mathbf{X}_{(i)}^{(l)} \tag{4.4.1}$$

where $\mathbf{X}_{(i)}^{(l)}$ is the i-th row vector for data matrix X for $i = 1, 2, \ldots, n_l$.

Let $\mathbf{a}$ be vector of the scalar projection for all observations of data matrix X on vector of mean difference. We define the scalar projection of the i-th observations, for $i = 1, 2, \ldots, n$, on vector of mean difference as

$$a_i = \mathbf{X}_{(i)}^T \frac{\overline{\mathbf{X}}^{(2)} - \overline{\mathbf{X}}^{(1)}}{\|\overline{\mathbf{X}}^{(2)} - \overline{\mathbf{X}}^{(1)}\|} \tag{4.4.2}$$

where a_i is the i-th entry of vector $\mathbf{a}$ and $\mathbf{X}_{(i)}$ is the vector of the i-th row of data matrix X.

Furthermore, let $\widetilde{\mathbf{X}}^{(1)}$ and $\widetilde{\mathbf{X}}^{(2)}$ be vectors of median of observations from Class 1 and Class 2, respectively. We define the j-th entry $\widetilde{X}_j^{(l)}$, for $j = 1, 2, \ldots, p$, of median vector $\widetilde{\mathbf{X}}^{(l)}$, for $l = 1, 2$, as

$$\widetilde{X}_j^{(l)} = \frac{X_{\lfloor (n_l+1):2 \rfloor} + X_{\lceil (n_l+1):2 \rceil}}{2} \tag{4.4.3}$$

where $X_{\lfloor (n_l+1):2 \rfloor}$ is the $\lfloor (n_l + 1) : 2 \rfloor$-th entry and $X_{\lceil (n_l+1):2 \rceil}$ is the $\lceil (n_l + 1) : 2 \rceil$-th entry of ordered vector $\mathbf{X}_j$ for $j = 1, 2, \ldots, p$.

Let $\mathbf{b}$ be vector of the scalar projection for all observations of data matrix X on vector of median difference. We define the scalar projection of the i-th observations, for $i = 1, 2, \ldots, n$, on vector of median difference as

$$b_i = \mathbf{X}_{(i)}^T \frac{\widetilde{\mathbf{X}}^{(2)} - \widetilde{\mathbf{X}}^{(1)}}{\|\widetilde{\mathbf{X}}^{(2)} - \widetilde{\mathbf{X}}^{(1)}\|} \tag{4.4.4}$$

where b_i is the i-th entry of vector $\mathbf{b}$ and $\mathbf{X}_{(i)}$ is the vector of the i-th row of data matrix X.

The scalar projection for the vector of row of data matrix X on vector of mean difference produces the scalar that represents the length of vector projection of the vector of row of data matrix X on vector of mean difference. In addition, this projection also contains the information about the angle between the vector of row of data matrix X on vector of mean difference. Likewise the scalar projection for the vector of row of data matrix X on vector of median difference.

Having defined vectors $\mathbf{a}$ and $\mathbf{b}$ as vectors whose entries as scalar projections between the vector of row of data matrix X on vectors of mean difference and on vector of median difference, respectively, we fit Classification Trees using either vectors $\mathbf{a}$ or $\mathbf{b}$. Thus, we construct Classification Trees using one explanatory variables instead of making use of 17571 and 18388 variables for Smooth CNA and DNACopy CNA datasets, respectively.

Furthermore, we display the comparison between either the vectors of mean difference or median difference and the maximum goodness of split of individual variables in both Smooth CNA and DNACopy CNA datasets. We begin with Figures 4.17 and 4.18 that show the line plot maximum goodness of split, mean difference and median difference for the whole variables of Smooth CNA and DNACopy CNA dataset.

These two figures indicate that both mean difference and median difference present the same information on an individual variable performance in carrying out data classification as in the maximum goodness of split. Variables whose large absolute either mean or median differences tend to produce high maximum goodness of split. As a result, these variables have large contribution on the ability of the resulting scalar projection vectors **a** and **b** to carry out data classification task. Table 4.2 presents maximum goodness of split values for these scalar projection vectors.

Table 4.2: Maximum goodness of split for vectors **a** and **b** of Smooth CNA and DNACopy CNA datasets

Dataset	Vector **a**	Vector **b**
Smooth CNA	0.4003	0.4500
DNACopy CNA	0.3837	0.3837

Moreover, Figures 4.17 and 4.18 show that DNACopy CNA dataset has more complex structure than Smooth CNA dataset in terms of the relation between either mean difference or median difference and maximum goodness of split value. DNACopy CNA dataset has several variables whose either large absolute mean difference or large absolute median difference which yield low maximum goodness of split. On the other hand, Smooth CNA dataset has variables with either high mean difference or median difference values which correspond with high maximum goodness of split values, and vice versa. This in turn affects the difference in the performance of the vectors **a** and **b** in constructing Classification Trees for Smooth CNA and DNACopy CNA datasets Table 4.2 confirms this result. This table indicates that maximum goodness of split values of both vector **a** and vector **b** of DNACopy CNA dataset are lower than those of Smooth CNA dataset.

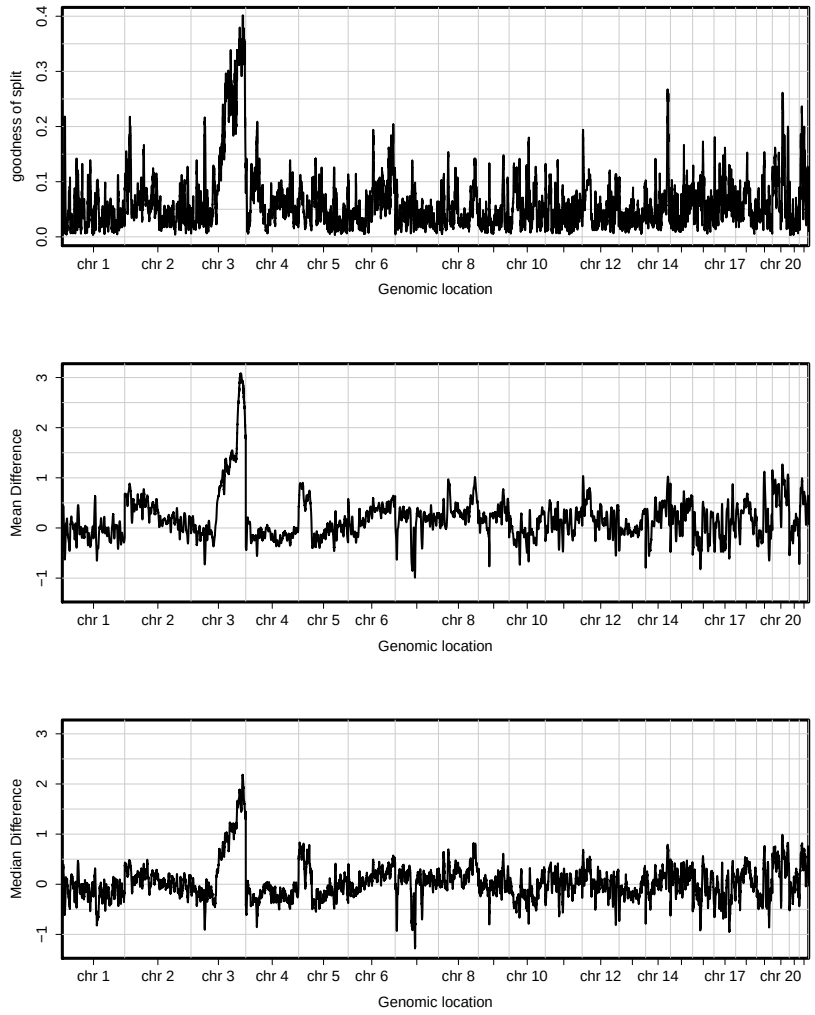


Figure 4.17: Plots of maximum goodness of split, mean difference and median difference for Smooth CNA dataset. The top panel presents the plot of maximum goodness of split for Smooth CNA dataset while the middle and the bottom panels show plots for its mean difference and median difference, respectively.

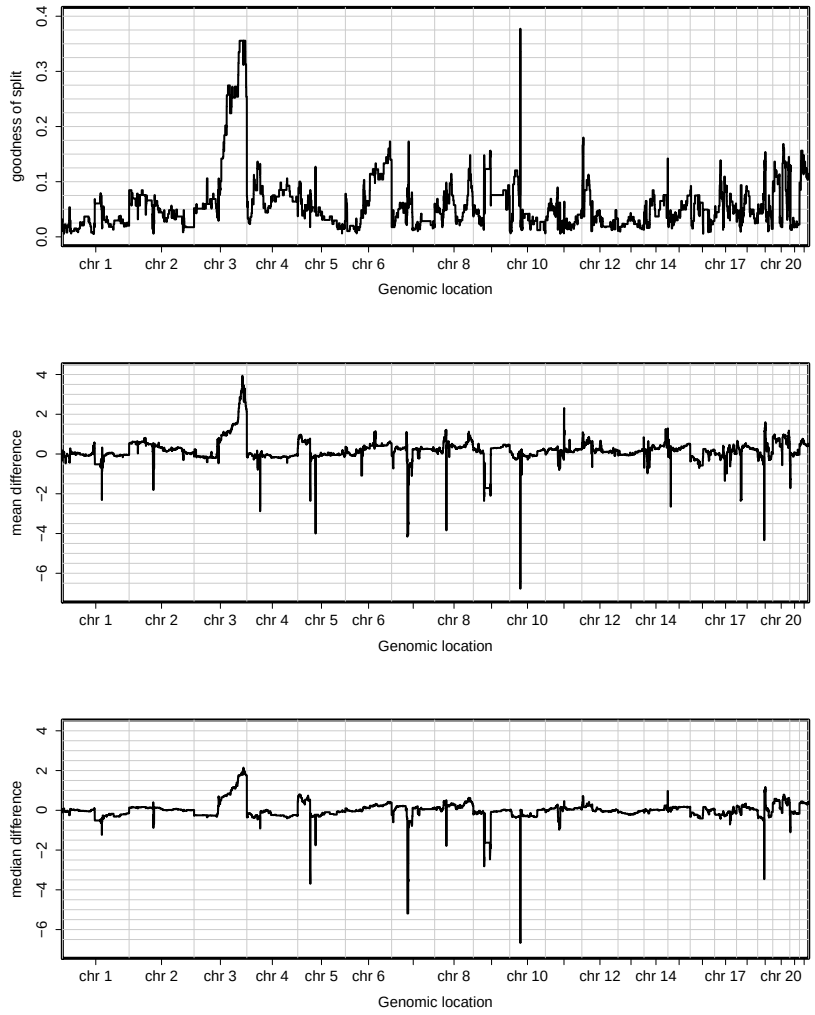


Figure 4.18: Plots of maximum goodness of split, mean difference and median difference for DNACopy CNA dataset. The top panel presents the plot of maximum goodness of split for DNACopy CNA dataset while the middle and the bottom panels show plots for its mean difference and median difference, respectively.

Furthermore, we show Figure 4.19 to present the true error rate estimates for Classification Trees fitted using the raw data, vector **a** and vector **b** along with their paired difference data. The true error rate estimates for the trees fitted using these three datasets are presented through boxplots in Panel (a) of this figure. Meanwhile, their paired difference are shown in Panel (b) of this figure. Both panels in this figure indicate how the error rates obtained using either vector **a** or vector **b** are lower than the one obtained using raw data. The same result also can be seen for DNACopy CNA dataset. We display these results in Figure 4.20.

To make an inference, we perform paired sample test to make an inference whether the Classification Trees fitted using mean difference projected datasets produce lower error rate than those of fitted using the entire variables for either Smooth CNA or DNACopy CNA datasets, likewise for median difference projected datasets. We carry out paired sample Wilcoxon Test for this task as an alternative to paired sample t-test.

For all paired sample Wilcoxon tests carried out, we define the null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero. We use this alternative hypothesis since we wish to obtain the lower error rates by involving data projection prior to Classification Trees construction. Table 4.3 shows the results for Smooth CNA dataset.

Table 4.3: P-values for Shapiro-Wilk test and Wilcoxon test for comparison of the error rates obtained from Classification Trees fitted using the whole variables and the scalar projection of Smooth CNA dataset

Comparison	Shapiro-Wilk test	Wilcoxon test
All chr - mean diff	0.00050	0.99952
All chr - median diff	0.00992	4.81683×10^{-10}

This table shows that all the paired difference data are not normally distributed. Moreover, this table also indicates that, for the paired difference of error rates of Classification Trees fitted using the raw dataset and those built

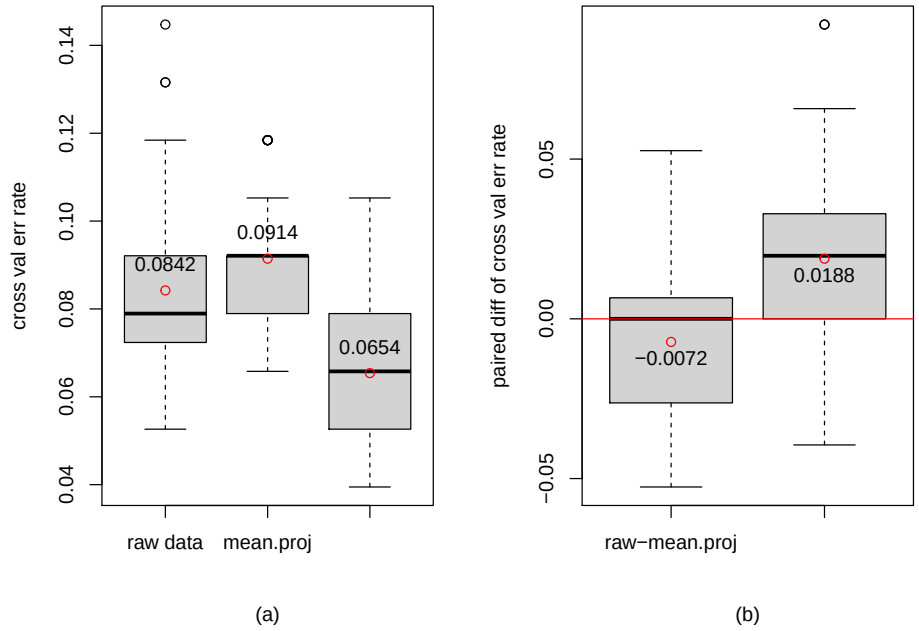


Figure 4.19: Boxplots of prediction error rate for Classification Trees or raw, mean difference projected and median difference projected along with their paired difference data for Smooth CNA dataset. Panel (a) shows boxplots of the error rate estimates of Classification Trees fitted using raw, mean difference projected and median difference projected datasets. Panel (b) displays two boxplots for paired-difference data between the error rates of Classification Trees fitted using raw and mean difference projected datasets and those of Classification Trees fitted using raw and median difference projected datasets. The red horizontal line in this panel corresponds with paired-difference equals 0. Using paired sample Wilcoxon Test with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero, we end up with p-value equals 0.999 for the paired comparison of the trees fitted using raw and mean difference projected datasets and 4.82×10^{-10} for the paired comparison of the trees fitted using raw and median difference projected datasets.

using mean-projected dataset, we fail to reject the null hypothesis that the median of the difference between the pairs equals zero. Hence, it can be concluded that, for Smooth CNA dataset, projecting dataset using mean-projection prior to Classification Trees construction does not improve the trees performance.

On contrast, for the paired difference of error rates for Classification Trees fitted using the raw dataset and those built using median-projected dataset, we reject the null hypothesis that the median of the difference between the pairs equals zero. To put it in another way, it can be concluded that, for Smooth CNA dataset, the trees fitted by the median-projected dataset produce lower error rates than those developed by the raw dataset.

Furthermore, Table 4.4 presents the results obtained from performing inferential task for DNACopy CNA dataset to make inference whether fitting Classification Trees using either mean-projected or median projected datasets yields lower error rates than fitting the trees using the raw dataset.

Table 4.4: P-values for Shapiro-Wilk test and Wilcoxon test for comparison of the error rates obtained from Classification Trees fitted using the whole variables and the scalar projection of DNACopy CNA dataset

Comparison	Shapiro-Wilk test	Wilcoxon test
All chr - mean diff	0.01206	9.22189×10^{-12}
All chr - median diff	0.00766	9.89792×10^{-10}

Table 4.4 indicates that all the paired difference data are not normally distributed. Furthermore, for the paired difference of error rates for Classification Trees fitted using the raw dataset and those built using either mean-projected or median-projected datasets, we reject the null hypothesis that the median of the difference between the pairs equals zero. Hence, it can be inferred that, for DNACopy CNA dataset, the trees fitted by either mean-projected or median-projected datasets produce lower error rates than those developed by the raw dataset.

Moreover, although fitting Classification Trees using both vectors **a** and **b** for both Smooth CNA and DNACopy CNA datasets gives advantage in terms of lower true error rate estimates, it also gives disadvantage. The resulting Classification Trees do not yield information on the contribution of variable of raw data on

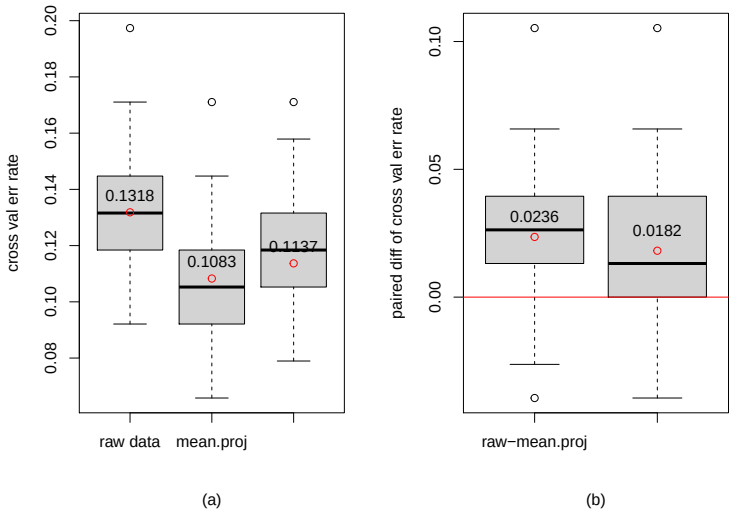


Figure 4.20: Boxplots of prediction error rate for Classification Trees or raw, mean difference projected and median difference projected along with their paired difference data for DNACopy CNA dataset. Panel (a) shows boxplots of the error rate estimates of Classification Trees fitted using raw, mean difference projected and median difference projected datasets. Panel (b) displays two boxplots for paired-difference data between the error rates of Classification Trees fitted using raw and mean difference projected datasets and those of Classification Trees fitted using raw and median difference projected datasets. The red horizontal line in this panel corresponds with paired-difference equals 0. Using paired sample Wilcoxon Test with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero, we end up with p-value equals 9.22×10^{-12} for the paired comparison of the trees fitted using raw and mean difference projected datasets and 9.90×10^{-10} for the paired comparison of the trees fitted using raw and median difference projected datasets.

Classification Trees construction. Instead, using this approach, the only available information on variable contribution in constructing Classification Trees is only either mean difference or median difference. This causes model interpretation to be more difficult.

Thus far, for our two real datasets, it has been explained that applying Classification Trees on datasets that have much more variables than observations, one ends up with Classification Trees that involve very few variables. As the consequence of this, the fitted Classification Trees leave the information contained in the vast majority of variables out. This findings suggest one to perform data dimension reduction prior to Classification Trees construction. This can be done by by applying either Principal Component Analysis (PCA) or Independent Component Analysis (ICA) on datasets prior to building Classification Trees. We are going to discuss this in the next two chapter.

4.5 Classification Trees of Simulated Datasets

In this section we discuss the results obtained from applying Classification Trees on simulated datasets. In this case, we generate several datasets using the settings presented in Section 2.4 of Chapter 2. There are three quantities are considered: correlation within-blocks of variables, mean difference between sub-populations and standard deviation of each single variable.

Among these three quantities, it is the correlation within-blocks of variables is our main concern. We pay more attention on this aspect as we wish to study how the presence of such this correlational structure affects Classification Trees performance. Our two real datasets have such this structure too. In this case, we consider ten correlation settings: $\rho = 0, 0.1, \ldots, 0.9$. For $\rho = 0$, we generate datasets without correlation block structure. On contrast, for $\rho = 0.9$, we generate datasets with correlation block structure. Due to the computational cost, we only study the case of three blocks of correlated variables where each correlational block consists of 50 variables.

Furthermore, in the light of mean difference, having three blocks of correlated variables, we generate each block of variables using distinct mean difference settings. We introduce this quantity since we might use it to generate datasets

with class information. We wish to have simulated datasets whose two groups of observations.

Among the three correlated blocks of variables, we let one of them as blocks of correlated variables without discriminative information. Meanwhile, the other two blocks have such information, yet they contain class information in the opposite direction. One of them has mean difference that moves from zero up to a negative integer value, whereas the other has mean difference that moves from zero up to a positive integer value. For the small, medium, and large mean difference settings, we let these integer values to be ± 1, ± 2 and ± 3, respectively. We consider this setting following the mean difference between group of observations in our real datasets. Our real datasets have several blocks of correlated variables with or without class information.

Finally, in terms of standard deviation, we generate our simulated datasets using two choices of standard deviation: unit standard deviation and three standard deviation. However, for the sake of simplicity, we restrict our case by setting the same standard deviation across variables. We study this quantity to compare the case of small variance and large variance. Our real datasets have many variables of large variance.

Having discussed the simulation study setting, we now move on explaining the simulation procedure applied. Firstly, we generate a dataset with a certain seed setting and fit a Classification Tree using this generated dataset. This dataset can be viewed as the training set. Secondly, we generate another dataset from the same population as of the training set. Yet, we use different seed setting. This second dataset can be seen as the test set and is applied for prediction purpose as well as for estimation of the error rate of the resulting Classification Tree. We carry out two these data generating tasks along with model fitting and model assessment for 100 times. We apply the test set error rate estimation for model assessment of the simulation study.

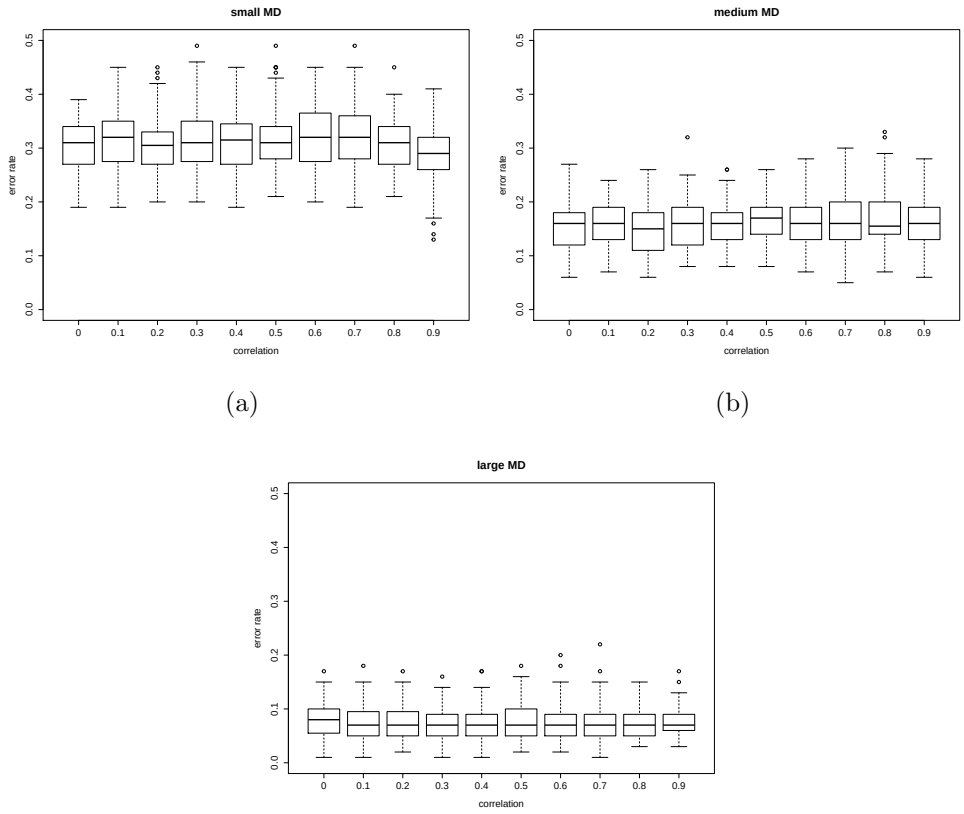
Turning now to the result obtained from this simulation study, Figures 4.21 and 4.22 show the boxplots for the prediction error rates of Classification Trees fitted using simulated datasets with the above settings. These prediction error rates are estimated using test set error rate estimation where we fit the model using one dataset and do prediction using another dataset. Therefore, for our

simulation studies, we generate two datasets with two different seed settings to obtain one prediction error estimate. We perform this data generating for 100 hundred times and average the resulting test set error estimates.

Moreover, all boxplots in the three panels of Figure 4.21 present the results of simulation studies with small standard deviations. In this case, we make use of one standard deviation. Meanwhile, those of Figure 4.22 show the results of simulation studies with three standard deviation. Within both figures, each panel presents the results obtained from distinct mean difference settings. The top left, top right and bottom panels show the results for small, medium and large mean differences, respectively. Whereas, within each panel, there are ten boxplots that display the results attained from ten correlation settings: $\rho = 0, 0.1, \ldots, 0.9$.

Looking at both Figures 4.21 and 4.22 in more detail, it is apparent that the correlation within blocks of variables does not affect the Classification Trees performance in a sense the different correlations lead to Classification Trees of the same accuracy level. That is fitting Classification Trees using either simulated datasets with or without blocks of correlated variables produces the trees of the same test set error rates. We end up with such this results regardless mean difference and standard deviation settings. This result may be explained by the fact that the correlation does not affect the discriminative information contained in datasets. Regardless the correlation setting, we always end up with the same either mean difference between groups of observations of maximum goodness of split for each individual variable.

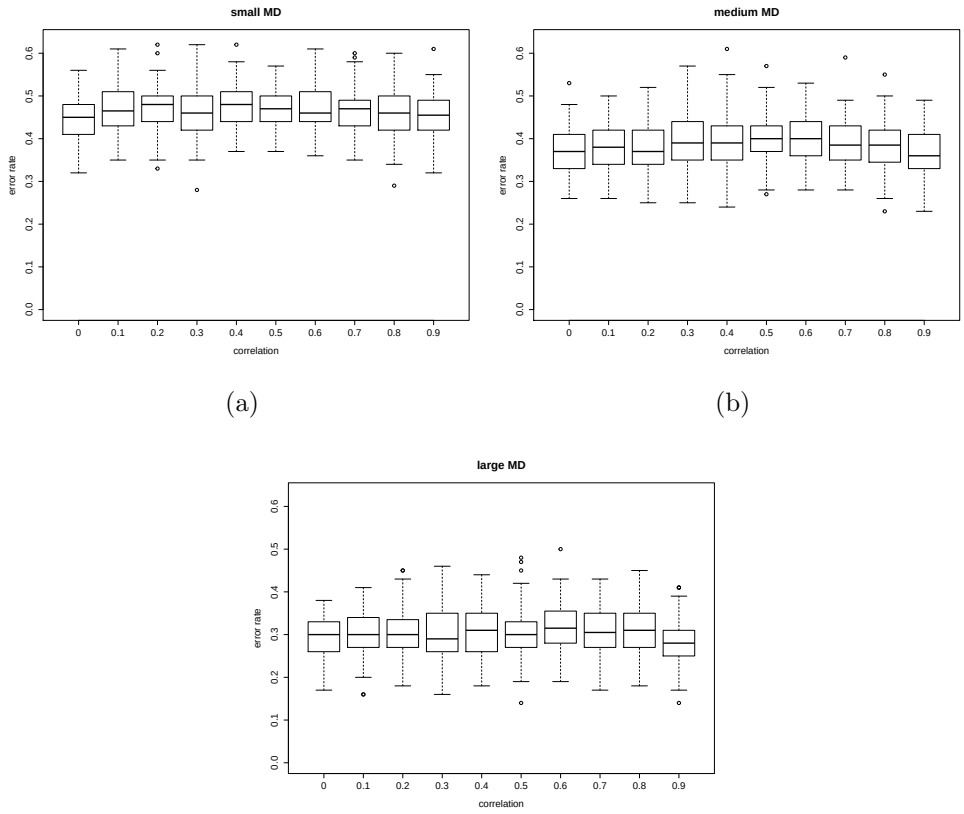
On contrast, both mean difference and standard deviation settings used for generating the simulated datasets affect Classification Trees performance. Fitting Classification Trees using datasets whose larger mean difference leads to more accurate trees, and vice versa. Whereas, fitting Classification Trees using datasets whose large standard deviation leads to less accurate trees, and visa versa.



(a) (b)

(c)

Figure 4.21: Boxplots of test set error rates of Classification Trees fitted using simulated datasets with one standard deviation. We obtain the results presented in the top-left, top-right, and bottom panels from simulation studies with small, medium and large mean difference, respectively. Meanwhile, within each panel, there are ten boxplots attained from correlation $\rho = 0, 0.1, \ldots, 0.9$.



(a) (b)

(c)

Figure 4.22: Boxplots of test set error rates of Classification Trees fitted using simulated datasets with three standard deviation. We obtain the results presented in the top-left, top-right, and bottom panels from simulation studies with small, medium and large mean difference, respectively. Meanwhile, within each panel, there are ten boxplots attained from correlation $\rho = 0, 0.1, \ldots, 0.9$.

4.6 Conclusion

Classification Trees give us simple numerical diagrams that can be easily interpreted. These diagrams provide us the insight on the classification task carried out by the trees. From a fitted tree, one can easily see how the splittings take place, how the interaction between variables involves during these splitting, and how an individual observation is allocated into a terminal node. However, for the datasets which have much more variables than observations, Classification Trees only pick very small number of strong relevant predictors as the classifying variables and leave a huge number of the variables aside. This leads to instability in both modelling and prediction. In the light of prediction error estimation, applying cross validation on such this tree also arises an issue. The cross validated error rate is unstable.

To overcome this dimensionality problem, variable selection step is applied before Classification Trees construction. In this chapter, we by picking the set of variables from genomic region of Chromosome 3 for Smooth CNA dataset, and the set of variables from genomic regions of Chromosomes 3 and 10 from DNACopy CNA dataset. These sets of variables are selected since they have high maximum goodness of split. This quantity represents the capability of variable in performing classification. For Smooth CNA dataset, the resulting trees share many things in common with the trees of the whole genomes. However, for more complex dataset such as DNACopy CNA dataset, better variable selection criterion is required.

Apart from applying variable selection step, we also utilize projection method as an attempt to reduce the data dimension. We apply the scalar projection of either mean difference of median difference. For Smooth CNA dataset, the scalar projection of median difference yields more accurate Classification Trees. Otherwise, the scalar projection of mean difference gives less accurate trees. Meanwhile, for DNACopy CNA dataset, both scalar projection methods yield more accurate trees. However, applying such this data projection method, we are no longer able to get deep insight upon the data classification task carried out.

Furthermore, the presence of blocks of correlated variables does not affect the performance of Classification Trees in terms of prediction error. On contrast,

the high data variation contained in the relevant predictors diminishes the performance of Classification Trees. These results can be seen from the simulation study carried out on the hypothetical datasets.

With respect to genetic point of view, Classification Trees of Smooth CNA dataset indicate that both genes SOX2 and PIK3CA can be used to stratify a patient into either one the two subtypes of tumour above. These two genes are allocated in loci 3q24 up to 3q27.3. The amplification in genes SOX2 and PIK3CA is common in squamous carcinoma lung cancer. Meanwhile, Classification Trees of DNACopy dataset show that both gene KIF5B in locus 10p11.22 and gene RET in locus 10q11.21 can be utilized as the markers. The fusion of these two neighbouring genes is common in adeno carcinoma lung cancer.

Chapter 5

Classification Trees of Principal Component Scores

5.1 Introduction

Fitting Classification Trees using datasets whose much more variables than observations, we leave large number of variables aside including some variables which have discriminative information. Hence, it is crucial to apply a data dimensionality reduction prior to Classification Trees construction. For this dimension reduction purpose, we can utilize feature extraction method. In this case, we apply Principal Component Analysis (PCA). By applying PCA, we reduce the data dimensionality while maintain its variability.

In the light of computational steps, we estimate the principal component scores of the datasets and use these estimated scores for constructing Classification Trees. In this case, there are two major concerns should be considered: how to explain the way PCA extracts the discriminative information that spread across variables in datasets and how to choose the relevant principal components to perform the classification task.

We begin this chapter by presenting some theoretical derivation of population and sample principal components. For the population principal components, we consider the case of random variables that have distribution as a mixture of two multivariate distributions. Meanwhile, for the sample principal components, we take the presence of two groups of observations in the datasets into account. We

base our derivation most on Koch (2013), Johnson & Wichern (2004), Mardia *et al.* (1979) and Jolliffe (2002).

Furthermore, we discuss the results obtained from the simulation study performed to figure out the way PCA extracts the informative information contained in datasets and the prediction accuracy of Classification Trees fitted using principal component scores. We carried out this study on the real and the hypothetical datasets.

5.2 Population Principal Components

In this section we present some theoretical aspects of population PCA by taking into account the classification task that will be carried out later on. Hence, we derive some concept on PCA by considering a random variable which has a mixture distribution.

5.2.1 The Expected Value and Covariance Matrix of Random Vector

Let $\mathbf{X} = (X_1, \ldots, X_p)^T$ be a p-dimensional random vector which has a distribution ψ as a mixture of two multivariate distributions, ψ_1 and ψ_2. Suppose that $\mathbf{X} \sim \psi_1$ with probability π and $\mathbf{X} \sim \psi_2$ with probability $1 - \pi$. Given that $\mathbf{X} \sim \psi_l$, for $l = 1, 2$, we define the expected value of random variable $\mathbf{X}|\psi_l$ as

$$E(\mathbf{X}|\psi_l) = \boldsymbol{\mu}^{(l)} \tag{5.2.1}$$

and its covariance matrix as

$$E\big[(\mathbf{X} - \boldsymbol{\mu}^{(l)})(\mathbf{X} - \boldsymbol{\mu}^{(l)})^T|\psi_l\big] = \Phi^{(l)}. \tag{5.2.2}$$

Hence, we obtain the expected value $\boldsymbol{\mu}$ of random variable $\mathbf{X}$ as

$$\begin{aligned} \boldsymbol{\mu} &= E(\mathbf{X}) \\ &= \pi E(\mathbf{X}|\psi_1) + (1 - \pi)E(\mathbf{X}|\psi_2) \\ &= \pi\boldsymbol{\mu}^{(1)} + (1 - \pi)\boldsymbol{\mu}^{(2)}. \end{aligned} \tag{5.2.3}$$

Moreover, for the random variable $\mathbf{X}$ given that $\mathbf{X} \sim \psi_l(\boldsymbol{\mu}^{(l)}, \Phi^{(l)})$, for $l = 1, 2$, we obtain the expected value $E(\mathbf{X}\mathbf{X}^T|\psi_l)$ as

$$\begin{aligned} E(\mathbf{X}\mathbf{X}^T|\psi_l) &= \text{Cov}(\mathbf{X}|\psi_\mathbf{l}) + E(\mathbf{X}|\psi_l)[E(\mathbf{X}|\psi_l)]^T \\ &= \Phi^{(l)} + \boldsymbol{\mu}^{(l)}[\boldsymbol{\mu}^{(l)}]^T. \end{aligned} \tag{5.2.4}$$

Let $\mathbf{d} = \boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)}$. Using the results obtained in Equation (5.2.4), we end up with the covariance of $\mathbf{X}$, Σ, as

$$\begin{aligned} \Sigma &= \text{Cov}(\mathbf{X}) \\ &= E(\mathbf{X}\mathbf{X}^T) - E(\mathbf{X})(E(\mathbf{X}))^T \\ &= \pi E(\mathbf{X}\mathbf{X}^T|\psi_1) + (1-\pi)E(\mathbf{X}\mathbf{X}^T|\psi_2) - E(\mathbf{X})(E(\mathbf{X}))^T \\ &= \pi\Phi^{(1)} + (1-\pi)\Phi^{(2)} + (\pi\boldsymbol{\mu}^{(1)} + (1-\pi)\boldsymbol{\mu}^{(2)})(\pi\boldsymbol{\mu}^{(1)} + (1-\pi)\boldsymbol{\mu}^{(2)})^T \\ &= \pi\Phi^{(1)} + (1-\pi)\Phi^{(2)} + \pi(1-\pi)\mathbf{d}\mathbf{d}^T. \end{aligned} \tag{5.2.5}$$

Based on the expression in Equation (5.2.5), it can be seen that for random vector $\mathbf{X}$ which has a distribution ψ as a mixture of two multivariate distributions, ψ_1 and ψ_2, its covariance matrix is obtained from the summation of:

1. the covariance matrix of $\mathbf{X}$ given that $\mathbf{X} \sim \psi_1$,
2. the covariance matrix of $\mathbf{X}$ given that $\mathbf{X} \sim \psi_2$ and
3. "the covariance matrix" of $\mathbf{X}$ which is determined by the difference between the expected value of $\mathbf{X}$ given that $\mathbf{X} \sim \psi_1$ and the expected value of $\mathbf{X}$ given that $\mathbf{X} \sim \psi_2$.

Furthermore, we can decompose the covariance matrix Σ obtained in Equation (5.2.5) using Eigenvalue decomposition as

$$\underset{p\times p}{\Sigma} = \underset{p\times p}{\Gamma} \times \underset{p\times p}{\Lambda} \times \underset{p\times p}{\Gamma^T} \tag{5.2.6}$$

where $\Lambda = diag\{\lambda_1, \lambda_2, \ldots, \lambda_p\}$ is a diagonal matrix of eigenvalues of Σ with $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_p \geq 0$ and $\Gamma = (\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \ldots, \boldsymbol{\gamma}_p)$ is an orthogonal matrix whose columns are eigenvectors of Σ. Each eigenvector $\boldsymbol{\gamma}_j$ corresponds to the eigenvalue λ_j for $j = 1, 2, \ldots, p$.

5.2.2 Population Principal Component Loadings and Scores

Let us now define the population principal component loadings and scores for a random vector $\mathbf{X}$ by using the result obtained by Eigenvalue decomposition in Equation (5.2.6).

Definition 5.1. *Let* $\mathbf{X} = (X_1, X_2 \ldots, X_p)^T$ *be a random vector with the covariance matrix* Σ *which has the matrix of eigenvectors* Γ *and the diagonal matrix of eigenvalues* Λ*. We define the population principal component scores* $\mathbf{L} = (L_1, L_2 \ldots, L_p)^T$ *for the random vector* $\mathbf{X}$ *as*

$$\underset{p\times 1}{\mathbf{L}} = \underset{p\times p}{\Gamma}^{T} \underset{p\times 1}{\mathbf{X}}. \tag{5.2.7}$$

We name matrix Γ as matrix of principal component loadings for random vector $\mathbf{X}$ and the j-th column of matrix Γ, $\boldsymbol{\gamma}_j$, as its j-th principal component loading. Meanwhile, we call the-j-th entry of vector $\mathbf{L}$, L_j, as its j-th principal component score. We obtain this principal component score by transforming the random vector $\mathbf{X}$ using the the j-th principal component loading $\boldsymbol{\gamma}_j$, or

$$L_j = \boldsymbol{\gamma}_j^T \mathbf{X} = \gamma_{j1} X_1 + \gamma_{j2} X_2 + \ldots + \gamma_{jp} X_p. \tag{5.2.8}$$

For the random vector of population principal components $\mathbf{L}$, we attain the covariance matrix as

$$\begin{aligned} \text{Cov}(\mathbf{L}) &= \text{Cov}(\Gamma^T \mathbf{X}) \\ &= \Gamma^T \text{Cov}(\mathbf{X}) \Gamma \\ &= \Gamma^T \Sigma \Gamma \\ &= \Gamma^T (\Gamma \Lambda \Gamma^T) \Gamma \\ &= \Lambda \end{aligned} \tag{5.2.9}$$

where Λ is the diagonal matrix of eigenvalues of covariance matrix Σ. Using this result, we obtain

$$\text{Cov}(L_j, L_{j^*}) = 0 \tag{5.2.10}$$

and

$$\text{Var}(L_j) = \lambda_j \tag{5.2.11}$$

for $j, j^* = 1, 2, \ldots, p$ and $j \neq j^*$.

This result indicates that the principal component scores are uncorrelated and have variances equal to the eigenvalues of Σ. This result also shows that the

principal component scores are ordered in terms of variance and the first principal component scores L_1 is the one that has higher variance. In the next theorem, which is adapted from Mardia *et al.* (1979), we will show that this first population principal component score L_1 is the standardized linear combination of random vector X which has highest variance.

Theorem 5.1.1. *No standardised linear combination of the random vector* $\mathbf{X}$ *has a variance larger than* λ_1.

Proof. Let $\boldsymbol{\alpha}$ be a vector of p constants $\alpha_1, \alpha_2, \ldots, \alpha_p$ and $\boldsymbol{\alpha}^T\boldsymbol{\alpha} = 1$. We shall prove that given $\Gamma = (\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \ldots, \boldsymbol{\gamma}_p)$ as matrix of eigenvectors of $\mathbf{X}$, we can express $\boldsymbol{\alpha}$ as linear combination of these eigenvectors since the eigenvectors constitute a basis for $\mathbb{R}^p$.

$$\boldsymbol{\alpha} = c_1\boldsymbol{\gamma}_1 + c_2\boldsymbol{\gamma}_2 + \ldots + c_p\boldsymbol{\gamma}_p = \Gamma\mathbf{c} \tag{5.2.12}$$

where $\mathbf{c} = (c_1, c_2, \ldots, c_p)^T$ is a vector of p constants.

Moreover, for the linear combination $\boldsymbol{\alpha}^T\mathbf{X}$, we obtain

$$\begin{aligned} \text{Cov}(\boldsymbol{\alpha}^T\mathbf{X}) &= \boldsymbol{\alpha}^T\text{Cov}(\mathbf{X})\boldsymbol{\alpha} \\ &= \boldsymbol{\alpha}^T\Sigma\boldsymbol{\alpha} \\ &= (\Gamma\mathbf{c})^T\Gamma\Lambda\Gamma^T(\Gamma\mathbf{c}) \\ &= \mathbf{c}^T\Gamma^T\Gamma\Lambda\Gamma^T\Gamma\mathbf{c} \\ &= \mathbf{c}^T\Lambda\mathbf{c} \\ &= \textstyle\sum_{j=1}^{p} \lambda_j c_j^2. \end{aligned} \tag{5.2.13}$$

Since we have $\boldsymbol{\alpha}^T\boldsymbol{\alpha} = 1$, then

$$\mathbf{c}^T\mathbf{c} = \sum_{j=1}^{p} c_j^2 = 1. \tag{5.2.14}$$

Therefore, since λ_1 is the largest eigenvalue, the maximum of $\text{Cov}(\boldsymbol{\alpha}^T\mathbf{X})$ with constraint $\mathbf{c}^T\mathbf{c} = 1$ is λ_1. This maximum is attained when $c_1 = 1$ and all the other c_j are zero. Therefore, $\text{Cov}(\boldsymbol{\alpha}^T\mathbf{X})$ is maximised when $\boldsymbol{\alpha} = \boldsymbol{\gamma}_1$. □

Result in Theorem 5.1.1 shows that the first principal component score is a linear combination of the random vector $\mathbf{X}$ which gives highest variance. The loading for this principal component score can be seen as the rotation of the

random vector $\mathbf{X}$ that rotates this random vector into the direction with highest variance. However, there is no theoretical justification that the second principal component score will yield a linear combination with the second highest variance. Instead, the second principal component score is only a linear combination of the random vector $\mathbf{X}$ which has highest variance among the set of linear combinations that are uncorrelated with the first principal component score.

Moreover, we shall show how PCA maintains the total variance of the random vector $\mathbf{X}$. We begin by obtaining the trace of the covariance matrix Σ.

$$\text{tr}\,(\Sigma) = \text{Var}(X_1) + \text{Var}(X_1) + \cdots + \text{Var}(X_p) = \sum_{j=1}^{p} \text{Var}(X_j). \tag{5.2.15}$$

On the other hand, we can also get the trace for covariance matrix Σ by expressing Σ in terms of its Eigenvalue decomposition as presented in Equation (5.2.6).

$$\text{tr}\,(\Sigma) = \text{tr}\,(\Gamma\Lambda\Gamma^T) = \text{tr}\,(\Lambda\Gamma^T\Gamma) = \text{tr}\,(\Lambda) = \lambda_1 + \lambda_2 + \ldots + \lambda_p = \sum_{j=1}^{p} \lambda_j. \tag{5.2.16}$$

Results in Equations (5.2.15) and (5.2.16) give

$$\sum_{j=1}^{p} \text{Var}(X_j) = \sum_{j=1}^{p} \lambda_j. \tag{5.2.17}$$

Equation (5.2.17) indicates that the overall variance of principal component scores includes the overall variance of random vector $\mathbf{X}$. However, due to the ordering on principal component scores in terms of their individual variance, the first few principal component scores have higher variances than the rest of principal component scores.

By utilising the result in Equation (5.2.17) we can also attain the proportion of variance due to an individual principal component score as well as the accumulation of proportion of variance due to the set of the first few principal component scores. For the j-th principal component score, we define a quantity to present the proportion of the total variance due to the j-principal component L_j as

$$\text{proportion of variance due to } L_j = \frac{\lambda_j}{\lambda_1 + \lambda_2 + \ldots + \lambda_p}. \tag{5.2.18}$$

Using result in Equation (5.2.18), we end up with the accumulation for the proportion of the overall variance explained by the first j-th principal component scores as

$$\begin{bmatrix} \text{proportion of variance} \\ \text{due to the first } j\text{-th PCs} \end{bmatrix} = \frac{\lambda_1 + \lambda_2 + \ldots \lambda_j}{\lambda_1 + \lambda_2 + \ldots + \lambda_p}. \tag{5.2.19}$$

5.2.3 Population Principal Component of Random Vector with Mixture Distribution

In this subsection, we utilise the definition of population principal components in Definition 5.1 to derive the mean and variance of these population principal components. We consider the case of a random vector which has a distribution as a mixture of two multivariate distribution.

As previously mentioned, we have a random vector $\mathbf{X}$ where $\mathbf{X} \sim \psi_1$ with probability π and $\mathbf{X} \sim \psi_2$ with probability $1 - \pi$. Using this assumption, we obtain a random vector of principal components $\mathbf{L}$ where $\mathbf{L} \sim \tilde{\psi}_1$ with probability π and $\mathbf{L} \sim \tilde{\psi}_2$ with probability $1 - \pi$. For $\mathbf{L} \sim \tilde{\psi}_l$, for $l = 1, 2$, we define the expected value of $\mathbf{L}|\tilde{\psi}_l$ as

$$E(\mathbf{L}|\tilde{\psi}_l) = E(\Gamma^T \mathbf{X}|\tilde{\psi}_l)\boldsymbol{\mu}^{(l)} = \Gamma^T E(\mathbf{X}|\psi_l) = \Gamma^T \boldsymbol{\mu}^{(l)} = \boldsymbol{\mu}_L^{(l)}. \tag{5.2.20}$$

Using the result obtained in Equation (5.2.20), we can attain the difference of the conditional expected values for the principal component scores of random vector $\mathbf{X}$.

$$\mathbf{d}_L = \boldsymbol{\mu}_L^{(1)} - \boldsymbol{\mu}_L^{(2)} = \Gamma^T \boldsymbol{\mu}^{(1)} - \Gamma^T \boldsymbol{\mu}^{(2)} = \Gamma^T \mathbf{d} \tag{5.2.21}$$

where $\mathbf{d} = \boldsymbol{\mu}^{(1)} - \boldsymbol{\mu}^{(2)}$. This result shows that the population principal component scores inherit the difference of the conditional expected values from the random vector $\mathbf{X}$. We rely the application of PCA prior to Classification Trees construction on this thought. In addition, this result also indicates that the difference of the expected values obtained from the random vector $\mathbf{X}$ might be masked by its variance when we deal with the difference of the expected values of principal component scores.

5.3 Sample Principal Components

In this section we present the derivation of sample PCA by taking into account the presence of samples drawn for two sub-populations.

5.3.1 Mean Vector and Covariance Matrix of Random Samples

Suppose that we have n independent observations on the p-dimensional random vector $\mathbf{X}$ where $n \lll p$. Among these n observations, n_1 of them are drawn from a sub-population that follows $\psi_p^{(1)}(\boldsymbol{\mu}^{(1)}, \Psi^{(1)})$ and n_2 are drawn from a sub-population that follows $\psi_p^{(2)}(\boldsymbol{\mu}^{(2)}, \Psi^{(2)})$, where $n_1 + n_2 = n$.

Let $\mathbf{X}_{(1)}^{(l)}, \mathbf{X}_{(2)}^{(l)}, \ldots, \mathbf{X}_{(n_l)}^{(l)}$, for $l = 1, 2$, denote a random sample of size n_l drawn from the l-th sub-population with $\psi_p^{(l)}(\boldsymbol{\mu}^{(l)}, \Psi^{(l)})$. It should be noted here that we use the notation of a column vector $\mathbf{X}_{(i)}^{(l)}$ to represent a row vector of data matrix $\mathbf{X}$. The notation $\mathbf{X}_{(i)}^{(l)}$ represents the i-th observation drawn from the l-th sub-population, for $i = 1, 2, \ldots, n_l$.

Meanwhile, we use the notation of a column vector $\mathbf{X}_j$ to represent a column vector of data matrix $\mathbf{X}$. The notation $\mathbf{X}_j$ represents the j-th variable. Therefore,we can express these two random samples into two matrices of random samples as

$$\underset{n_l \times p}{\mathrm{X}^{(l)}} = \begin{bmatrix} \left(\mathbf{X}_{(1)}^{(l)}\right)^T \\ \left(\mathbf{X}_{(2)}^{(l)}\right)^T \\ \vdots \\ \left(\mathbf{X}_{(n_l)}^{(l)}\right)^T \end{bmatrix} \tag{5.3.1}$$

for $l = 1, 2$. Moreover, these two random samples constitute a random sample of

size n for $n = n_1 + n_2$. We can express this random sample of size n as

$$\underset{n\times p}{\mathrm{X}} = \begin{bmatrix} \underset{n_1\times p}{\mathrm{X}^{(1)}} \\ \hline \underset{n_2\times p}{\mathrm{X}^{(2)}} \end{bmatrix} = \left[\begin{array}{c} \left(\mathbf{X}_{(1)}^{(1)}\right)^T \\ \left(\mathbf{X}_{(2)}^{(1)}\right)^T \\ \vdots \\ \left(\mathbf{X}_{(n_1)}^{(1)}\right)^T \\ \hline \left(\mathbf{X}_{(1)}^{(2)}\right)^T \\ \left(\mathbf{X}_{(2)}^{(2)}\right)^T \\ \vdots \\ \left(\mathbf{X}_{(n_2)}^{(2)}\right)^T \end{array}\right] = \begin{bmatrix} \left(\mathbf{X}_{(1)}\right)^T \\ \left(\mathbf{X}_{(2)}\right)^T \\ \vdots \\ \left(\mathbf{X}_{(n)}\right)^T \end{bmatrix}. \tag{5.3.2}$$

Furthermore, we define the sample mean for the l-th random sample as

$$\overline{\mathbf{X}}^{(l)} = \frac{1}{n_l}\sum_{i=1}^{n_l}\mathbf{X}_{(i)}^{(l)} \tag{5.3.3}$$

and the sample covariance the l-th random sample as

$$\mathrm{S}^{(l)} = \frac{1}{n_l - 1}\sum_{i=1}^{n_l}(\mathbf{X}_{(i)}^{(l)} - \overline{\mathbf{X}}^{(l)})(\mathbf{X}_{(i)}^{(l)} - \overline{\mathbf{X}}^{(l)})^T \tag{5.3.4}$$

for $l = 1, 2$.

Moreover, for the random sample of size n, we obtain the sample mean as

$$\overline{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_{(i)} = \frac{1}{n}\sum_{l=1}^{2}\sum_{i=1}^{n_l}\mathbf{X}_{(i)}^{(l)} = \frac{n_1\overline{\mathbf{X}}^{(1)} + n_2\overline{\mathbf{X}}^{(2)}}{n_1 + n_2} \tag{5.3.5}$$

and the sample covariance as

$$\mathrm{S} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{X}_{(i)} - \overline{\mathbf{X}})(\mathbf{X}_{(i)} - \overline{\mathbf{X}})^T \tag{5.3.6}$$

By doing some algebraic works on the sample covariance as in Equation (5.3.6), we end up with

$$\begin{aligned} \mathrm{S} &= \tfrac{1}{n-1}\left\{(n_1-1)\mathrm{S}^{(1)} + (n_2-1)\mathrm{S}^{(2)} + \tfrac{n_1 n_2}{n}\left(\overline{\mathbf{X}}^{(1)} - \overline{\mathbf{X}}^{(2)}\right)\left(\overline{\mathbf{X}}^{(1)} - \overline{\mathbf{X}}^{(2)}\right)^T\right\} \\ &= \tfrac{1}{n-1}\left\{(n_1-1)\mathrm{S}^{(1)} + (n_2-1)\mathrm{S}^{(2)} + \tfrac{n_1 n_2}{n}\widehat{\mathbf{d}}\widehat{\mathbf{d}}^T\right\} \end{aligned} \tag{5.3.7}$$

where $\widehat{\mathbf{d}} = \overline{\mathbf{X}}^{(1)} - \overline{\mathbf{X}}^{(2)}$.

We can also express the result obtained in Equation (5.3.7) as

$$\mathrm{S} = \mathrm{S}_W^{(1)} + \mathrm{S}_W^{(2)} + \mathrm{S}_B \tag{5.3.8}$$

where

$$\mathrm{S}_W^{(l)} = \frac{n_l - 1}{n - 1}\mathrm{S}^{(l)}$$

for $l = 1, 2$ and

$$\mathrm{S}_B = \frac{n_1 n_2}{n(n-1)}\widehat{\mathbf{d}}\widehat{\mathbf{d}}^T.$$

By viewing the samples drawn from a sub-population as the observations belong to a certain group, we might call the covariance matrix $\mathrm{S}_W^{(l)}$ as within-group covariance of samples belong to the l-th group of observations and the covariance matrix S_B as between-group covariance of samples across groups.

Within-group covariance matrix $\mathrm{S}_W^{(l)}$, for $l = 1, 2$, presents the variation across samples of the same group. This quantity shows how the observations of one group vary. On the other hand, between-group covariance S_B introduces the variation in dataset due to the presence of two distinct groups of observations. This quantity indicates how the observations across groups differ. As a result of this, we might make use of this quantity for data classification purpose. We even might utilise the quantity $\widehat{\mathbf{d}}$ for this task. We call $\widehat{\mathbf{d}}$ as mean difference between observations of Class 1 and Class 2.

Moreover, as we are dealing with data matrix $\underset{n\times p}{\mathrm{X}}$ with $p \ggg n$, thus we obtain the rank of data matrix X

$$rank(\mathrm{X}) \leq min\{n, p\} = n,$$

and the rank of its sample covariance S

$$rank(\mathrm{S}) = n - 1$$

since we obtain the sample covariance from centred data matrix. In addition, for the samples drawn from the l-th, for $l = 1, 2$, sub-population we obtain the rank for sample within-group covariance as

$$rank(\mathrm{S}_W^{(l)}) = n_l - 1$$

and the rank of between-group covariance matrix S_B

$$rank(\mathrm{S}_B) = 1.$$

Therefore, for the covariance matrix S, there are $n-1$ non-zero eigenvalues, while for between-group covariance matrix S_B there is only one non-zero eigenvalue. We express the Eigenvalue decomposition for the covariance matrix S as

$$\underset{p\times p}{\mathrm{S}} = \underset{p\times(n-1)}{\widehat{\Gamma}} \times \underset{(n-1)\times(n-1)}{\widehat{\Lambda}} \times \underset{(n-1)\times p}{\widehat{\Gamma}^T} \tag{5.3.9}$$

where $\widehat{\Lambda} = diag\{\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_{n-1}\}$ is a diagonal matrix of eigenvalues of S with $\widehat{\lambda}_1 \geq \widehat{\lambda}_2 \geq \cdots \geq \widehat{\lambda}_{n-1} > 0$ and $\widehat{\Gamma} = (\widehat{\boldsymbol{\gamma}}_1, \widehat{\boldsymbol{\gamma}}_2, \ldots, \widehat{\boldsymbol{\gamma}}_{n-1})$ is an orthogonal matrix whose columns are its eigenvectors. Each eigenvector $\widehat{\boldsymbol{\gamma}}_j$ corresponds to eigenvalue $\widehat{\lambda}_j$ for $j = 1, 2, \ldots, n-1$. These eigenvectors also determine the direction of variation within data matrix X.

5.3.2 Sample Principal Component Loadings and Scores

Having discussed the population principal component loadings and scores along with the sample mean vector and covariance matrix, let us now consider the sample principal component loadings and scores. We begin by applying the result obtained from Eigenvalue decomposition in Equation (5.3.9). Using this result, we define matrices for the sample principal component loadings and scores for data matrix X as in Definition 5.2.

Definition 5.2. *Let* X *be an* $n \times p$ *data matrix with sample covariance matrix* S. *Let* S *have matrix of eigenvectors* $\widehat{\Gamma}$ *and diagonal matrix of eigenvalues* $\widehat{\Lambda}$. *We define the matrix of sample principal components* $\widehat{\mathrm{L}}$ *for data matrix* X *as*

$$\underset{n\times(n-1)}{\widehat{\mathrm{L}}} = \underset{n\times p}{\mathrm{X}} \underset{p\times(n-1)}{\widehat{\Gamma}}. \tag{5.3.10}$$

By using result $\widehat{\Gamma} = (\widehat{\boldsymbol{\gamma}}_1, \widehat{\boldsymbol{\gamma}}_2, \ldots, \widehat{\boldsymbol{\gamma}}_{n-1})$, we can express Equation (5.3.10) as

$$\begin{aligned} \widehat{\mathrm{L}} &= \mathrm{X}\widehat{\Gamma} \\ &= \mathbf{X}(\widehat{\boldsymbol{\gamma}}_1, \widehat{\boldsymbol{\gamma}}_2, \ldots, \widehat{\boldsymbol{\gamma}}_{n-1}) \\ &= (\mathbf{X}\widehat{\boldsymbol{\gamma}}_1, \mathbf{X}\widehat{\boldsymbol{\gamma}}_2, \ldots \mathbf{X}\widehat{\boldsymbol{\gamma}}_{n-1}) \\ &= (\widehat{\mathbf{L}}_1, \widehat{\mathbf{L}}_2, \ldots, \widehat{\mathbf{L}}_{n-1}) \end{aligned} \tag{5.3.11}$$

or,

$$\widehat{\mathbf{L}}_j = \mathrm{X}\widehat{\boldsymbol{\gamma}}_j \tag{5.3.12}$$

for $j = 1, 2, \ldots, n-1$. We call $\widehat{\mathbf{L}}_j$ as the j-th sample principal component score. The j-th sample principal component score can be seen as the linear transformation of data matrix X using the j-th sample principal component loading $\widehat{\boldsymbol{\gamma}}_j$.

Moreover, by expressing data matrix X as a matrix for row vectors of observations as expressed in Equation (5.3.2), we end up with

$$\widehat{\mathrm{L}} = \mathrm{X}\widehat{\Gamma} = \begin{bmatrix} \left(\mathbf{X}_{(1)}\right)^T \\ \left(\mathbf{X}_{(2)}\right)^T \\ \vdots \\ \left(\mathbf{X}_{(n)}\right)^T \end{bmatrix} \widehat{\Gamma} = \begin{bmatrix} \left(\mathbf{X}_{(1)}\right)^T \widehat{\Gamma} \\ \left(\mathbf{X}_{(2)}\right)^T \widehat{\Gamma} \\ \vdots \\ \left(\mathbf{X}_{(n)}\right)^T \widehat{\Gamma} \end{bmatrix} = \begin{bmatrix} \left(\widehat{\mathbf{L}}_{(1)}\right)^T \\ \left(\widehat{\mathbf{L}}_{(2)}\right)^T \\ \vdots \\ \left(\widehat{\mathbf{L}}_{(n)}\right)^T \end{bmatrix} \tag{5.3.13}$$

or

$$\left(\widehat{\mathbf{L}}_{(i)}\right)^T = \left(\mathbf{X}_{(i)}\right)^T \widehat{\Gamma} \tag{5.3.14}$$

for $i = 1, 2, \ldots, n$. We call $\left(\widehat{\mathbf{L}}_{(i)}\right)^T$ as the sample principal component scores for the i-th observation. It should be noted here that $\left(\widehat{\mathbf{L}}_{(i)}\right)^T$ is a row vector of matrix principal component scores $\widehat{\mathrm{L}}$. Meanwhile $\widehat{\mathbf{L}}_j$ is its a column vector of matrix principal component scores $\widehat{\mathrm{L}}$.

Furthermore, for the matrix of sample principal component scores $\widehat{\mathrm{L}}$, we obtain the sample mean $\widetilde{\mathbf{L}}$ as

$$\widetilde{\mathbf{L}} = \frac{1}{n}\sum_{i=1}^{n} \widehat{\mathbf{L}}_{(i)} = \frac{1}{n}\sum_{i=1}^{n} \widehat{\Gamma}^T \mathbf{X}_{(i)} = \widehat{\Gamma}^T \overline{\mathbf{X}} \tag{5.3.15}$$

and the sample covariance $\mathrm{S}_{\widehat{L}}$ as

$$\begin{aligned} \mathrm{S}_{\widehat{L}} &= \tfrac{1}{n-1}\textstyle\sum_{i=1}^{n}(\widehat{\mathbf{L}}_{(i)} - \widetilde{\mathbf{L}})(\widehat{\mathbf{L}}_{(i)} - \widetilde{\mathbf{L}})^T \\ &= \tfrac{1}{n-1}\textstyle\sum_{i=1}^{n}(\widehat{\Gamma}^T\mathbf{X}_{(i)} - \widehat{\Gamma}^T\overline{\mathbf{X}})(\widehat{\Gamma}^T\mathbf{X}_{(i)} - \widehat{\Gamma}^T\overline{\mathbf{X}})^T \\ &= \widehat{\Gamma}^T\big\{\tfrac{1}{n-1}\textstyle\sum_{i=1}^{n}(\mathbf{X}_{(i)} - \overline{\mathbf{X}})(\mathbf{X}_{(i)} - \overline{\mathbf{X}})^T\big\}\widehat{\Gamma} \\ &= \widehat{\Gamma}^T \mathrm{S}\widehat{\Gamma}. \end{aligned} \tag{5.3.16}$$

Using the result of Eigenvalue decomposition of the sample covariance matrix S as in Equation (5.3.9), we can express the resulting sample covariance for the matrix of principal component scores $\widehat{\mathrm{L}}$ in Equation (5.3.16) as

$$\mathrm{S}_{\widehat{L}} = \widehat{\Gamma}^T \mathrm{S} \widehat{\Gamma} = \widehat{\Gamma}^T (\widehat{\Gamma} \widehat{\Lambda} \widehat{\Gamma}^T) \widehat{\Gamma} = \widehat{\Lambda} \tag{5.3.17}$$

where $\widehat{\Lambda}$ is diagonal matrix for the eigenvalues of data matrix X. Likewise for the population principal component scores, this result displays that the sample principal component scores are uncorrelated too and have sample variances equal to the eigenvalues of S, or

$$\widehat{\mathrm{Cov}}(\widehat{\mathbf{L}}_j, \widehat{\mathbf{L}}_{j^*}) = 0 \tag{5.3.18}$$

and

$$\widehat{\mathrm{Var}}(\widehat{\mathbf{L}}_j) = \widehat{\lambda}_j \tag{5.3.19}$$

for $j, j^* = 1, 2, \ldots, p$ and $j \neq j^*$. In addition, this result also implies that the sample principal component scores are ordered in terms of their variances.

Furthermore, using the same way as used for the population principal component, we obtain that PCA maintains the total variance of the data matrix X or

$$\sum_{j=1}^{p} \widehat{\mathrm{Var}}(\mathbf{X}_j) = \sum_{j=1}^{n-1} \widehat{\mathrm{Var}}(\widehat{\mathbf{L}}_j) = \sum_{j=1}^{n-1} \widehat{\lambda}_j \tag{5.3.20}$$

where $\mathbf{X}_j$ in the j-th column of the data matrix X and $\widehat{\mathbf{L}}_j$ is the j-th principal component score of data matrix X or the j-th column of the matrix of sample principal component scores. However, unlike for the population case, for the sample principal component we only have $n-1$ principal component scores since we are dealing with the case of $p \ggg n$.

On the other hand, as shown in Equation (5.3.20), these much smaller number of principal component scores were imposed to contain as much variation as the much larger number of variables within data matrix X. As the result of this, we end up with some principal component scores which have much higher variances than each single variable of data matrix X has. These principal component scores hold large portion of data variation.

Moreover, we can also obtain the proportion of data variation explained by a single sample principal component score. For the j-th sample principal component score, we define a quantity to present the proportion of the total variance due to the j-principal component $\widehat{\mathbf{L}}_j$ as

$$\begin{bmatrix}\text{proportion of variance} \\ \text{due to } \widehat{\mathbf{L}}_j\end{bmatrix} = \frac{\widehat{\lambda}_j}{\widehat{\lambda}_1 + \widehat{\lambda}_2 + \ldots + \widehat{\lambda}_{n-1}}. \tag{5.3.21}$$

Meanwhile the accumulation for the proportion of the data variation explained by the first j-th principal component scores as

$$\begin{bmatrix}\text{proportion of variance} \\ \text{due to the first } j\text{-th PCs}\end{bmatrix} = \frac{\widehat{\lambda}_1 + \widehat{\lambda}_2 + \ldots \widehat{\lambda}_j}{\widehat{\lambda}_1 + \widehat{\lambda}_2 + \ldots + \widehat{\lambda}_p}. \tag{5.3.22}$$

5.3.3 Sample Principal Component of Data Matrix with Class Information

In this subsection, we discuss the sample principal components of a data matrix which contains class information. We consider the case of two classes. We begin by attaining the sample principal component scores for each class of observations.

By viewing data matrix $\mathbf{X}$ as a matrix of a random sample of size n drawn from a population contains two sub-populations as also presented in Equation (5.3.2) we get

$$\widehat{\mathbf{L}} = \mathbf{X}\widehat{\Gamma} = \begin{bmatrix}\mathbf{X}^{(1)} \\ \hline \mathbf{X}^{(2)}\end{bmatrix}\widehat{\Gamma} = \begin{bmatrix}\mathbf{X}^{(1)}\widehat{\Gamma} \\ \hline \mathbf{X}^{(2)}\widehat{\Gamma}\end{bmatrix} = \begin{bmatrix}\widehat{\mathbf{L}}^{(1)} \\ \hline \widehat{\mathbf{L}}^{(2)}\end{bmatrix} \tag{5.3.23}$$

where $\widehat{\mathbf{L}}^{(k)}$, for $k = 1, 2$, is the matrix of principal component scores for observations of the k-th class. This result shows that we can obtain the sample principal component scores for each observation by considering class information contained in data matrix $\mathbf{X}$. Hence, we might view matrix $\widehat{\mathbf{L}}^{(k)}$, for $k = 1, 2$, as a data matrix or random sample which contains n_k observations.

Moreover, for the k-th group of observations, we obtain the sample mean $\widetilde{\mathbf{L}}^{(k)}$ of matrix $\widehat{\mathbf{L}}^{(k)}$, as

$$\begin{aligned}\widetilde{\mathbf{L}}^{(k)} &= \tfrac{1}{n_k}\textstyle\sum_{i=1}^{n_k}\widehat{\mathbf{L}}_{(i)}^{(k)} \\ &= \tfrac{1}{n_k}\textstyle\sum_{i=1}^{n_k}\widehat{\Gamma}^T\mathbf{X}_{(i)}^{(k)} \\ &= \widehat{\Gamma}^T\overline{\mathbf{X}}^{(k)}\end{aligned} \tag{5.3.24}$$

for $k = 1, 2$. Using this result, we end up with the mean difference of principal component scores of observations of Class 1 and Class 2 as

$$\widehat{\mathbf{d}}_{\widehat{L}} = \widetilde{\mathbf{L}}^{(1)} - \widetilde{\mathbf{L}}^{(2)} = \widehat{\Gamma}^T\overline{\mathbf{X}}^{(1)} - \widehat{\Gamma}^T\overline{\mathbf{X}}^{(2)} = \widehat{\Gamma}^T(\overline{\mathbf{X}}^{(1)} - \overline{\mathbf{X}}^{(2)}) = \widehat{\Gamma}^T\widehat{\mathbf{d}} \qquad (5.3.25)$$

where $\widehat{\mathbf{d}} = \overline{\mathbf{X}}^{(1)} - \overline{\mathbf{X}}^{(2)}$. As previously mentioned for the population principal component scores, this result also shows that for the sample principal component scores, the information on mean difference might be masked by the variance. In addition, it should be noted that both $\widehat{\mathbf{d}}$ and $\widehat{\mathbf{d}}_{\widehat{L}}$ are the vectors which have very different length. The former is the vector of length p, while the latter is the one of length $n - 1$ which is much smaller than p.

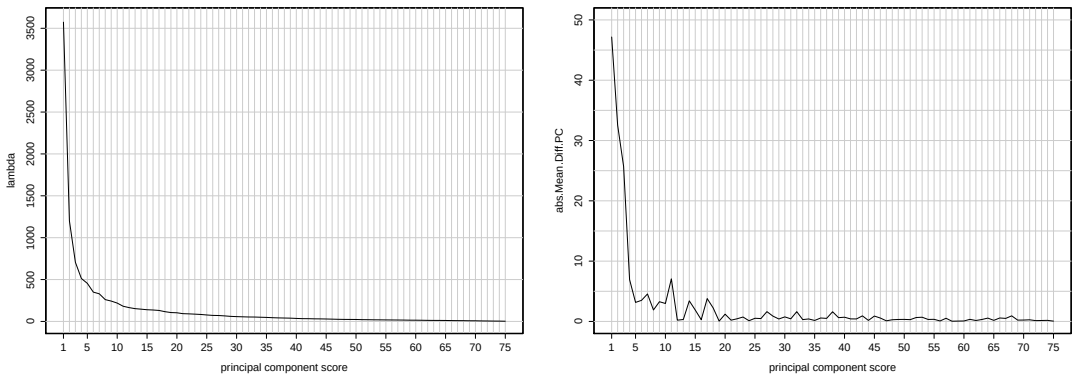
5.4 Classification Trees of Principal Component Scores for Real Datasets

In this section we discuss the result of applying Principal Component Analysis (PCA) prior to Classification Tree construction on our real datasets, Smooth CNA and DNACopy CNA datasets. To begin with, we present several plots that describe the of properties of both principal component loadings and scores of these datasets. As previously mentioned, we are dealing with datasets which have much larger number of variables than observations. This gives principal component scores that have much higher variance than the original variables. Moreover, as we apply PCA as a projection method prior to a classification task, we should consider how PCA deals with discriminative information contained in raw datasets.

For computational purpose, we use **prcomp** function to obtain principal component loadings and implement **predict** function on the resulting principal component loadings to get matrix of principal component scores. Using this principal component scores, we fit Classification Trees and do prediction. Finally, we obtain the true error rate estimate for the Classification Trees fitted using these principal component scores by making use of 100-repeated 10-fold stratified cross validation.

5.4.1 Classification Trees of Principal Component Scores for Smooth CNA Dataset

In this subsection we discuss the result of applying PCA prior to Classification Trees construction for Smooth CNA dataset. We begin by presenting the screeplot and the plot of mean difference of the principal component scores for observations of Class 1 and Class 2 of this dataset. We display the screeplot not only for showing the proportion of data variation due the the first j-th principal component scores but also for presenting how each individual of the first few principal component scores have much higher variance than each individual variable of data matrix. We present these plots in Figure 5.1.



(a) (b)

Figure 5.1: Plots for variance (the screeplot) and mean difference for principal component scores of Smooth CNA dataset. Panel (a) shows the plot for the variance for each individual principal component score or the screeplot for Smooth CNA dataset. Panel (b) presents the plot for the absolute value for the mean difference between the observations of Class 1 and Class 2 in terms of principal component scores for each individual principal component scores of Smooth CNA dataset. In both panels, the horizontal axis corresponds with each individual principal component score. Label 5 in this axis corresponds with the fifth principal component score.

Panel (a) of Figure 5.1 indicates that the first principal component score has variance that greater than 3500. This value is much greater than the variance of individual variable for the raw data matrix. Among all variables of Smooth CNA dataset, the highest variance for single variable is 11.0456. Likewise, the other ten first principal component scores also have much higher variance than individual variable of raw data does. This indicates that applying PCA on dataset that has much higher variables than observations leads to imposing the first few principal component scores to contain large proportion of data variation.

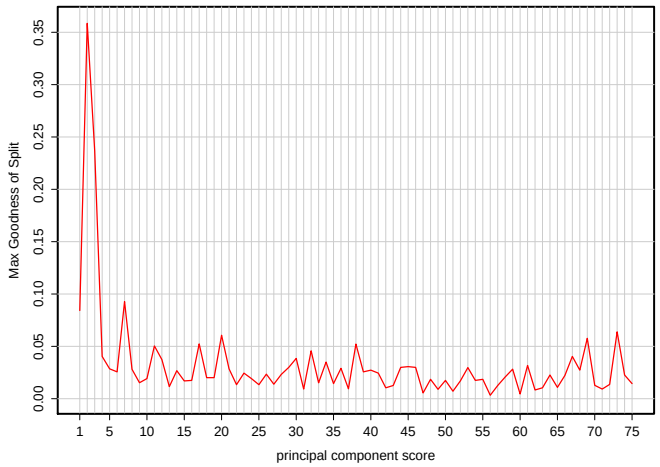


Figure 5.2: Plot of maximum goodness of split for principal component scores of Smooth CNA dataset. In this plot, the horizontal axis corresponds with each individual principal component score. Label 5 in this axis corresponds with the fifth principal component score.

These first few principal component scores which have high variance also have high absolute values of the mean difference between observations of Class 1 and Class 2. We present the plot for the latter quantity in Panel (b) of Figure 5.1. This plot displays that the first principal component score gives the highest mean difference among the entire principal component scores. However, the ratio between this mean difference and the variance of this principal component score is small.

On the other hand, the second principal component score has much lower variance than the first one, likewise for its mean difference. Yet, the ratio between its mean difference and variance is larger than that of the first principal component score. As a result, the second principal component score has better performance on classifying observations into two distinct groups. We visualise this using the plot of maximum goodness of split for principal component scores in Figure 5.2.

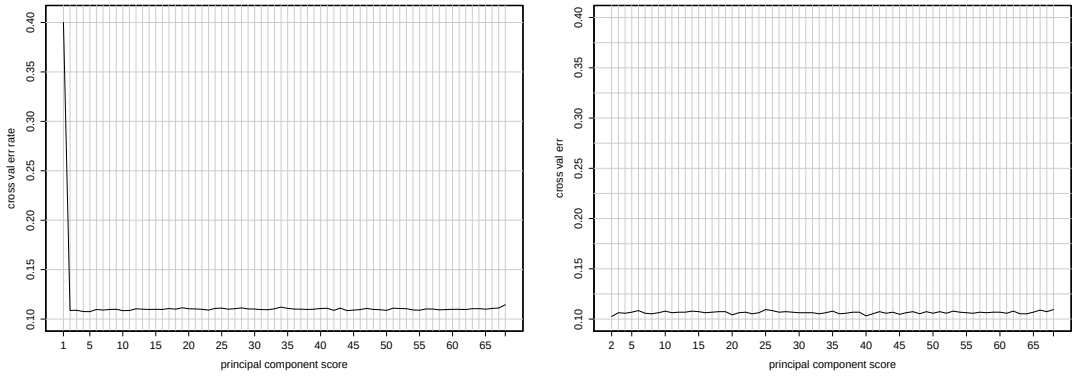
The plot in this figure shows that the second principal component scores of Smooth CNA dataset is a strong relevant predictor. This principal component score has much higher maximum goodness of split than the rest principal component scores. Therefore, during Classification Trees construction, once this principal component score is involved, adding more principal component scores do not affect the Classification Trees prediction accuracy as shown in Panel (a) of Figure 5.3.

Returning to the plot of maximum goodness of split values for principal component scores in Figure 5.2, apart from the fact that the second principal component is the strong relevant predictor, we have the first principal component as a weak relevant predictor. This component has few discriminative information. In the presence of a strong predictor principal component, excluding this component will not affect the Classification Trees performance as shown in Panel (b) of Figure 5.3.

Furthermore, both panels in Figure 5.3 present the prediction error rate for Classification Trees of principal component scores for Smooth CNA dataset. This true error estimate is obtained using repeated 10-fold stratified cross validation. We estimate the principal component scores using the training set, fit the trees using the resulting principal component scores of training set and do prediction using the estimated principal component scores of the test set.

We obtain the results presented in Panel (a) of Figure 5.3 by performing cross validation estimation involving the set of the first j principal component scores, for $j = 1, 2, \ldots, 68$. Meanwhile, we get the results presented in Panel (b) of this figure by carrying out cross validation estimation involving the set of the first j principal component scores excluding the first component.

The plot in Panel (a) shows how adding more principal component scores after involving the second principal component score does not affect the Classification



(a) (b)

Figure 5.3: Plot of prediction error for Classification Trees of the set of the first j principal component scores for Smooth CNA dataset. Panel (a) presents plot of prediction error for Classification Trees of the complete set of the first j principal component scores and Panel (b) shows plot of prediction error for Classification Trees of the set of the first j principal component scores by excluding the first component for Smooth CNA dataset. For Panel (a), the horizontal axis corresponds to the set of the first j-th principal components. For example, the error rate estimate which corresponds with label 5 at horizontal axis is the prediction error rate obtained from Classification Trees fitted using the set of all the first five principal components. On the other hand, for Panel (b), the horizontal axis corresponds with the set of the first j-th principal components excluding the first principal component score. For example, the error rate estimate which corresponds with label 5 at horizontal axis is the prediction error rate obtained from Classification Trees built using the set consists of the second, third, fourth and fifth principal component scores. The first principal component score has been excluded.

Trees prediction accuracy. Whereas, the plot in Panel (b) shows how excluding the first principal component and using the rest of components does not affect the trees accuracy either. This indicates how adding one strong relevant predictor in constructing the tree greatly influences its accuracy. On contrast, once the strong predictor is used to fit the tree, discarding either weak relevant or irrelevant predictor does not affect its performance.

Moreover, we display the comparison of the prediction error rate estimates obtained from Classification Trees fitted using raw data and the trees built using and set of the first two principal components along with their paired-difference data in Figure 5.4. This figure consists of two panels. Panel (a) presents two boxplots of the prediction error rate estimates mentioned above, meanwhile Panel (b) shows boxplot of their paired-difference data. Both panel in this figure indicate that applying PCA prior to Classification Trees construction for Smooth CNA dataset does not give any advantage. Instead, applying PCA will lead to the less accurate trees.

To make an inference on these results, we perform paired sample test to investigate whether the Classification Trees fitted using the raw dataset produce higher error rate than those of fitted using the set of the first two principal component scores. For this inferential task, we carry out paired sample Wilcoxon Test with the null hypothesis H_0: median of difference between the pairs equals zero and one-sided alternative hypothesis H_1: median of difference between the pairs is greater than zero.

From the paired sample Wilcoxon Test, we obtain p-value equals 1. Hence, it can be inferred that we fail to reject the null hypothesis that the median of the difference between the pairs equals zero. Therefore, for Smooth CNA dataset, it can be concluded that applying PCA prior to Classification Trees construction does not improve the accuracy of the resulting trees.

To figure out this result, let us consider the plot in Figure 5.5. This plot shows the variance along with between group variance for each individual variable for Smooth CNA dataset. The latter is obtained using the decomposition of sample variance as presented in Subsection 5.3.1. From the plot in Figure 5.5, it can be seen that the data variation due to between group variance only has small contribution to the overall variance. Therefore, its effect on the resulting principal

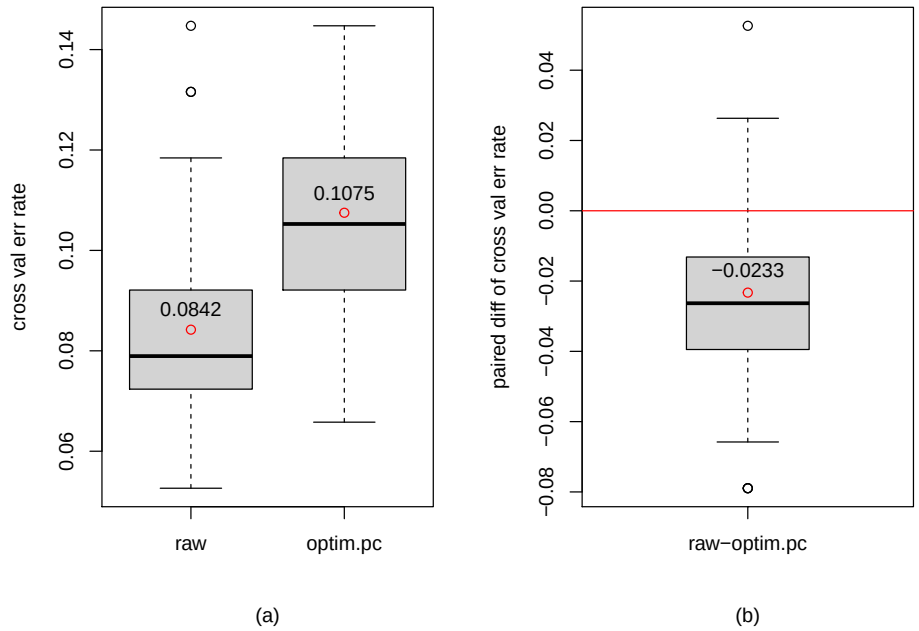


Figure 5.4: Boxplots for misclassification rate of Classification Trees obtained using raw data and set of the first two principal components for Smooth CNA dataset along with their paired difference data. Panel (a) shows boxplots for error rate estimates of Classification Trees fitted using raw dataset (the left boxplot) and error rate estimates of trees fitted using set of the first two principal components (the right boxplot). Both boxplots represent 100 estimates of Classification Trees error rate obtained using 100-repeated 10-fold cross validation estimation. Panel (b) displays paired-difference data between these two sets of the error rate estimates. The red horizontal line in this panel corresponds with paired-difference equals 0. Using paired sample Wilcoxon Test with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero, we end up with p-value equals 1.

component scores is masked by the effect of within group variance. This in turn declines the performance of PCA as a projection method prior to Classification Trees construction.

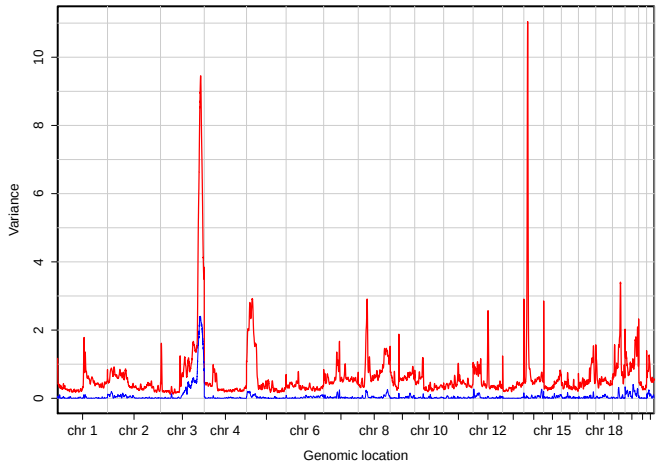


Figure 5.5: Plots for overall variance (red line) and between group variance (blue line) of each individual variable of Smooth CNA dataset. For this figure, the horizontal axis corresponds with genomic regions.

Moreover, with regard to the computational time, we present the plot in Figure 5.6. Each dot in this plot represents the computational time required to fit Classification Trees using the set of the first j-th principal components of Smooth CNA dataset. This figure indicates that we need shorter time for fitting Classification Trees using the set of the first j-th principal components than using the raw dataset. It takes 16.69539 seconds to build Classification Trees using the raw data.

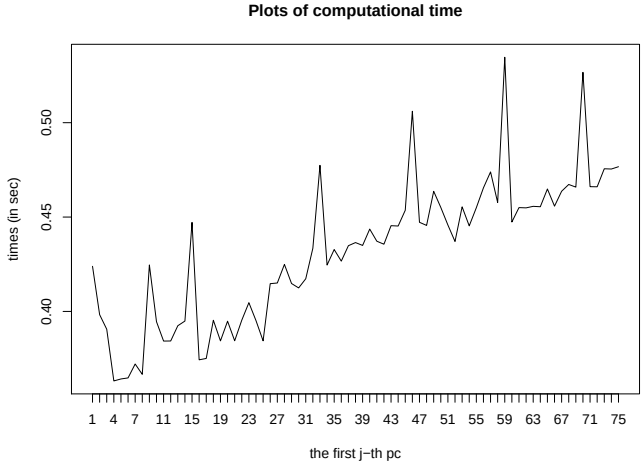
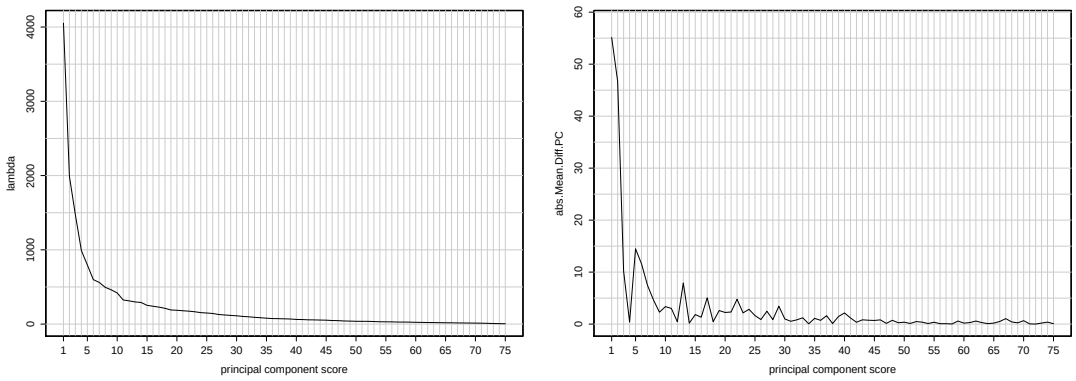


Figure 5.6: Plot of computational time for Classification Trees of principal components for Smooth CNA dataset. In this plot, the horizontal axis corresponds to the set of the first j-th principal components.

5.4.2 Classification Trees of Principal Component Scores for DNACopy CNA Dataset

In this subsection we discuss the results obtained from constructing Classification Trees using principal component scores of DNACopy CNA dataset. We present the results attained for this dataset in the same way we present the results for Smooth CNA dataset. We begin with presenting the screeplot and the plot of mean difference of the principal component scores for observations of Class 1 and Class 2 as in Figure 5.7.



(a) (b)

Figure 5.7: Plots for variance (the screeplot) and mean difference for principal component scores of DNACopy CNA dataset. Panel (a) shows the plot for the variance of principal component scores or the screeplot for DNACopy CNA dataset. Panel (b) presents the plot for the absolute value for the mean difference between the observations of Class 1 and Class 2 in terms of principal component scores of DNACopy CNA dataset. In both panels, the horizontal axis corresponds with each individual principal component score. Label 5 in this axis corresponds with the fifth principal component score.

Panel (a) of Figure 5.7 shows the similarity between principal component scores of DNACopy CNA and Smooth CNA datasets in terms of their variance. Applying PCA on data matrix which has much higher number of variables than

observations leads to the first few principal component scores which explain for the large proportion of the data variation than the rest of principal component scores.

Panel (b) of Figure 5.7 presents the plot for the absolute values of the mean difference between observations of Class 1 and Class 2 for DNACopy CNA dataset. The resulting plot indicates the same pattern as shown for Smooth CNA dataset. The first principal component scores give much higher absolute value of mean difference than the rest of principal component scores.

The second principal component score of this dataset also has the same property as the second principal component score for Smooth CNA dataset. The ratio between the absolute value of mean difference and variance of this principal component score is larger the ratio for the first principal component score. For this reason, this second principal component score has better performance of classifying observations into two distinct groups. We show this using the plot of maximum goodness of split for principal component scores in Figure 5.8.

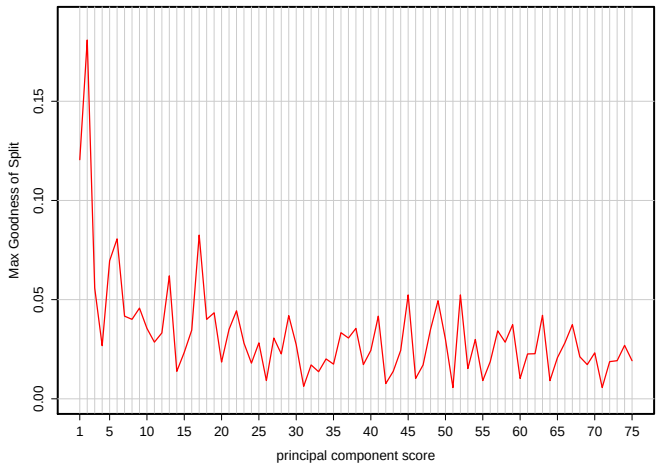


Figure 5.8: Plot of maximum goodness of split for principal component scores of DNACopy CNA dataset. In this plot, the horizontal axis corresponds with each individual principal component score. Label 5 in this axis corresponds with the fifth principal component score.

However, this principal component score does not have as high maximum

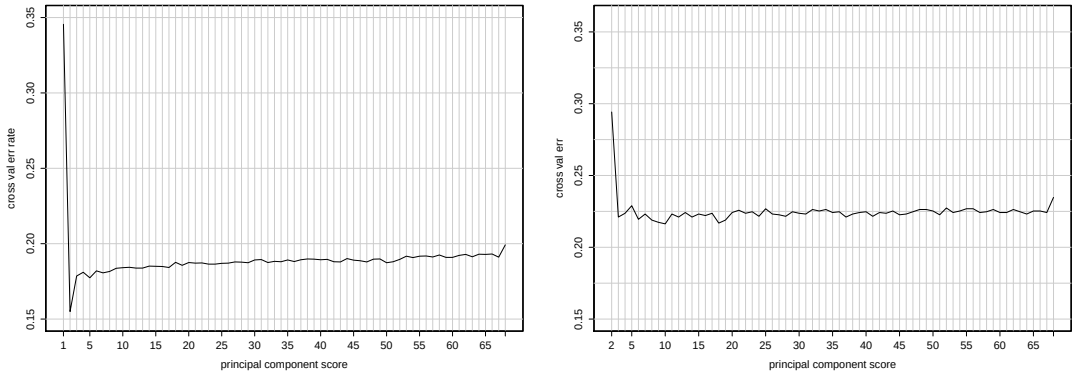
goodness of split as the second principal component score of Smooth CNA dataset does. Hence, the second principal component score for DNACopy CNA dataset does not have as strong predictive ability as the one for Smooth CNA dataset does. We present the plot in Panel (a) of Figure 5.9 to show how this fact affects the accuracy of Classification Trees of principal component scores for DNACopy CNA dataset.

Panel (a) of Figure 5.9 presents the prediction error rate for Classification Trees of the first j-th principal component scores for DNACopy CNA dataset, for $j = 1, 2, \ldots, 68$. These estimates are also attained using repeated 10-fold stratified cross validation. The plot in this panel shows that involving the second principal component score along with the first one in fitting the tree reduces the prediction error rate.

However, adding the third principal component scores increases the prediction error estimation. We get this result due to the fact that the second principal component score of DNACopy dataset does not have strong predictive ability such that it can be easily affected by including either one weak relevant or one irrelevant predictor into the model. In the absence of any strong relevant predictor, an irrelevant predictor might be picked as the classifying variable during the Classification Trees construction. This in turn diminishes the performance of the resulting trees.

Interestingly, the first principal component of DNACopy CNA dataset has different effect then the first component of Smooth CNA dataset. For DNACopy CNA dataset, excluding the first component and constructing the Classification Trees using the rest of principal components lead the less accurate trees as shown in Panel (b) of Figure 5.9. This indicates that, for this dataset, both the first and the second components should be picked to fit the model simultaneously.

Moreover, we present the comparison between the performance of Classification Trees of raw and PCA projected for DNACopy CNA dataset along with their paired difference data using boxplots in Figure 5.10. For PCA projected dataset, we present the result obtained from using the first two principal component scores to fit the trees. Using this set of principal component scores, we end up with the lowest error rate.



(a) (b)

Figure 5.9: Plot of prediction error for Classification Trees of the set of the first j principal component scores for DNACopy CNA dataset. Panel (a) presents plot of prediction error for Classification Trees of the complete set of the first j principal component scores and Panel (b) shows plot of prediction error for Classification Trees of the set of the first j principal component scores excluding the first component score for DNACopy CNA dataset. For Panel (a), the horizontal axis corresponds to the set of the first j-th principal components. For example, the error rate estimate which corresponds with label 5 at horizontal axis is the prediction error rate obtained from Classification Trees fitted using the set of all the first five principal components. On the other hand, for Panel (b), the horizontal axis corresponds with the set of the first j-th principal components excluding the first principal component score. For example, the error rate estimate which corresponds with label 5 at horizontal axis is the prediction error rate obtained from Classification Trees built using the set consists of the second, third, fourth and fifth principal component scores. The first principal component score has been excluded.

Boxplots in Panel (a) of Figure 5.10 show that applying PCA prior to Classification Trees construction for DNACopy CNA dataset does not give advantage. Boxplot of paired difference data in Panel (b) of this figure confirms this result. Compared to the results obtained previously for Smooth CNA dataset, the result for DNACopy dataset indicates the worse performance of PCA as projection method prior to classification task. In this case, adding more principal component scores after achieving an optimal result increases the prediction error rate significantly.

To make an inference on these results, we perform paired sample test to investigate whether the Classification Trees fitted using the raw dataset produce lower error rate than those of fitted using the set of the first two principal component scores. For this inferential task, we carry out paired sample Wilcoxon Test with the null hypothesis H_0: median of difference between the pairs equals zero and one-sided alternative hypothesis H_1: median of difference between the pairs is greater than zero. The use of this non-parametric test is based on the fact that the data of the difference between the pairs of the error rates are not normally distributed. We end up with p-value 0.00896 from Shapiro-Wilk test of normality.

From the paired sample Wilcoxon Test, we obtain p-value equals 1. Hence, it can be inferred that we fail to reject the null hypothesis that the median of the difference between the pairs equals zero. Therefore, for DNACopy CNA dataset, it can be concluded that applying PCA before Classification Trees construction does not improve the accuracy of the resulting trees. We attain the same result for Smooth CNA dataset.

We end up with this result due to the more complex structure of DNACopy CNA dataset. This dataset has many blocks of variables which do not only have absolute high mean differences but also high variance. There are also several variables spread across many genomic locations which have extremely high variances with low mean differences. We display this result in Figure 5.11. The variances of these variables dominate the eigenvalues of the covariance matrix.

Moreover, with regard to the computational time, we present the plot in Figure 5.12. Each dot in this plot represents the computational time required to fit Classification Trees using the set of the first j-th principal components of DNACopy CNA dataset. This figure indicates that we need shorter time for

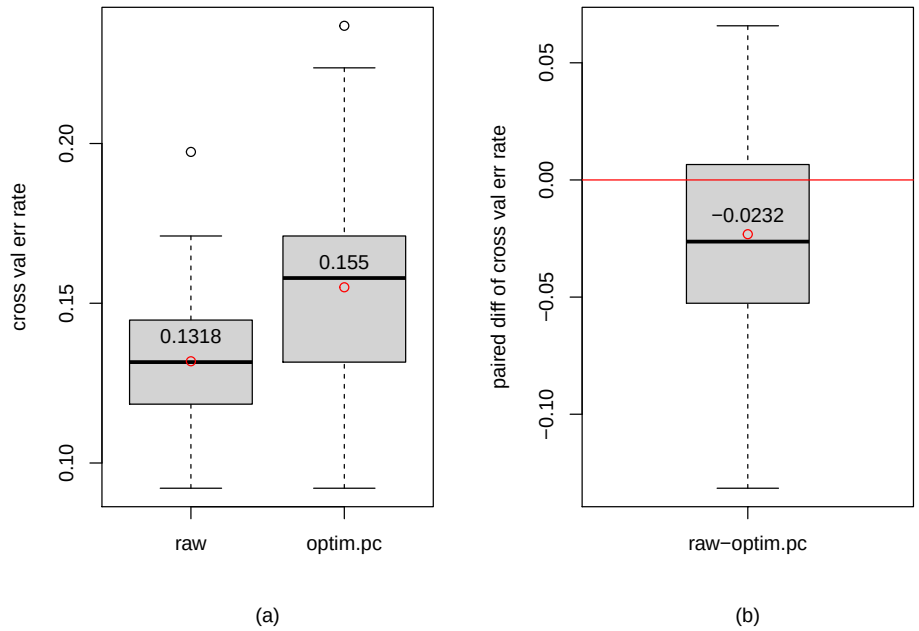


Figure 5.10: Boxplots for misclassification rate of Classification Trees obtained using raw data and set of the first two principal components for DNACopy CNA dataset along with their paired difference data. Panel (a) shows boxplots for error rate estimates of Classification Trees fitted using raw dataset (the left boxplot) and error rate estimates of trees fitted using set of the first two principal components (the right boxplot). Both boxplots represent 100 estimates of Classification Trees error rate obtained using 100-repeated 10-fold cross validation estimation. Panel (b) displays paired-difference data between these two sets of the error rate estimates. The red horizontal line in this panel corresponds with paired-difference equals 0. Using paired sample Wilcoxon Test with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero, we end up with p-value equals 1.

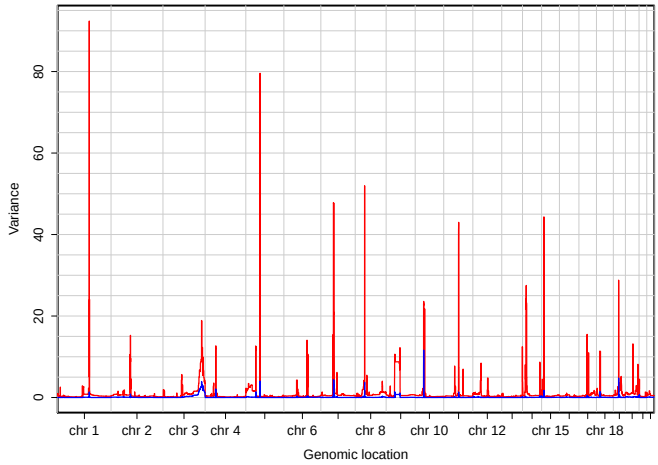


Figure 5.11: Plots for overall variance (red line) and between group variance (blue line) of each individual variable of DNACopy CNA dataset. For this figure, the horizontal axis corresponds with genomic regions.

fitting Classification Trees using the set of the first j-th principal components than using the raw dataset. It takes 16.83186 seconds to build Classification Trees using the raw data.

Thus far, for Classification Trees of principal components of either Smooth CNA or DNACopy CNA datasets, we can see how between-group variability in these two datasets are masked by their within group variation while we apply PCA. These result confirms findings proposed by Chang (1983) who applied principal component projection for reducing data dimension before clustering. Chang (1983) put forward that the set of the first few principal components do not necessarily contain discriminatory information. Hence, it is not recommended to use these components without additional consideration. In addition, Jolliffe (2002) mentioned that applying PCA as data dimension reduction before discriminant analysis only will work well if between group variation dominates within group variation. For this reason, it is suggested that one standardises the datasets before applying PCA. We will discuss this in the next section.

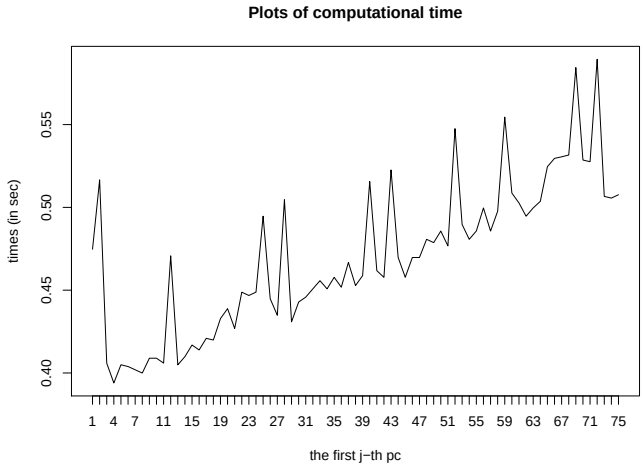


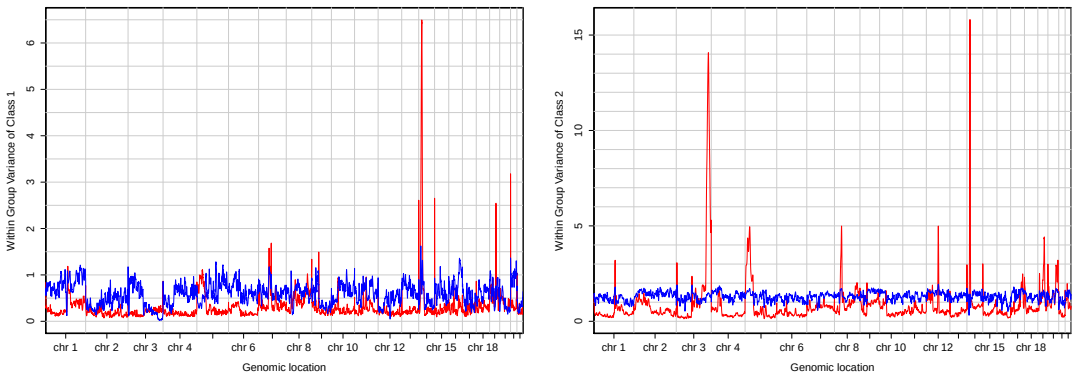
Figure 5.12: Plot of computational time for Classification Trees of principal components for DNACopy CNA dataset. In this plot, the horizontal axis corresponds to the set of the first j-th principal components.

5.5 Classification Trees Built Using Principal Component Scores of Standardised Dataset

In this section, we discuss the result obtained from building Classification Trees using principal component scores of standardised datasets. Instead of applying PCA directly on the data matrix X, we standardise this dataset and apply PCA on this standardised data. For computational purpose, we use **scale** function. We construct Classification Trees using the resulting principal component scores.

We perform data standardisation as an effort to reduce the effect of several variables which have extremely high variance. Such variables affect PCA strongly that they decline the performance of principal component scores to carry out data classification task. In addition, we employ standardisation before applying PCA since data standardisation does not change the order of observations in datasets.

We present Figures 5.13, 5.14 and 5.15 to describe how standardising affects the construction of Classification Trees using principal component scores of standardised datasets on Smooth CNA dataset. Figure 5.13 shows how standardising the data reduces the effect of several extremely high within group variances within

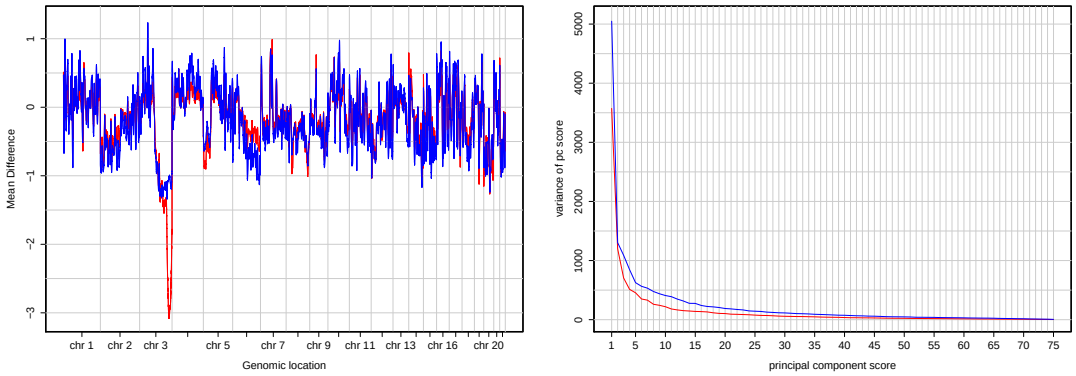


(a) (b)

Figure 5.13: Plots of within group variance for original variables and standardised variables of observations of Class 1 and Class 2 of Smooth CNA dataset. Panel (a) presents plots of within group variance for original variables (red line) and standardised variables (blue line) of observations of Class 1 and panel (b) presents plots of within group variance for original variables (red line) and standardised variables (blue line) of observations of Class 2 for Smooth CNA dataset. For both panels, the horizontal axis corresponds with genomic regions.

several genomic location for both classes of observations.

However, standardising also increases the variances of large number of variables with low variances. Vast majority of variables in Smooth CNA dataset have within group variance which is less than 1. This in turn leads to the increase in data variation. We can see how the variance of principal component scores of standardised variables increases as in Panel (b) of Figure 5.14

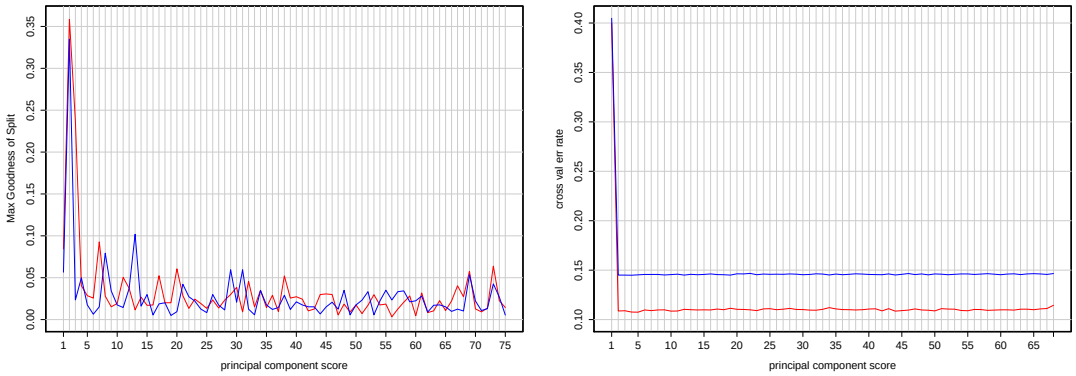


(a) (b)

Figure 5.14: Plots for mean difference of observations from Class 1 and Class 2 using raw and standardised variables along with plots of goodness of split principal component scores of raw and standardised variables for Smooth CNA dataset. Panel (a) shows plots for mean difference of observations from Class 1 and Class 2 using raw (red line) and standardised variables (blue line) for Smooth CNA dataset. For this panel, the horizontal axis corresponds with genomic locations. Panel (b) presents plots of variance for principal component scores of raw (red line) and standardised variables (blue line) for Smooth CNA dataset. For this panel, the horizontal axis corresponds with each individual principal component score.

Moreover, standardising reduces the effect of high mean difference variables too. We show this in Panel (a) of Figure 5.14. Many variables which have high absolute values of mean difference also have high variances. As standardising

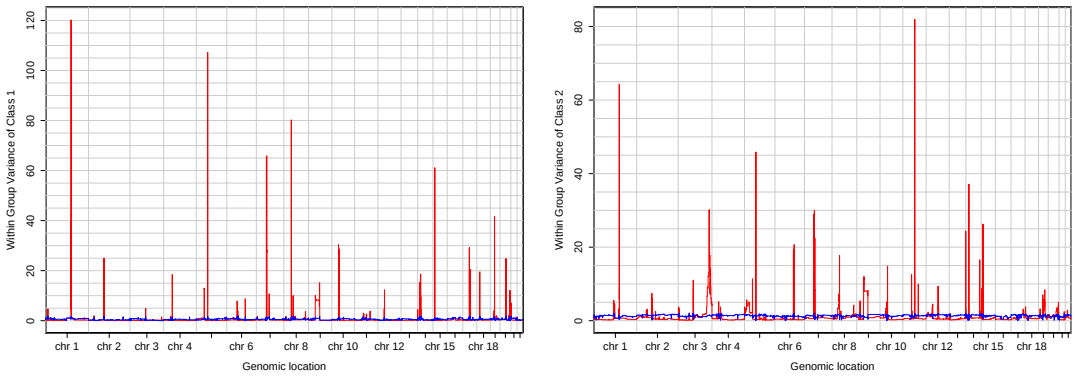
increases within group variance for vast majority of variables and lowers high mean difference values, it decline the performance of PCA as a projection method prior to classification task. Panel (a) of Figure 5.15 shows how standardising reduce the maximum goodness of split values of principal component scores of standardised variables. Meanwhile, Panel (b) of this figure indicates the increase in the true error rate estimates of the Classification Trees built using the set of the first j principal component scores of standardised Smooth CNA dataset, for $j = 1, 2, \ldots, 68$.



(a) (b)

Figure 5.15: Plots of goodness of split principal component scores of raw and standardised variables along with the plots of their error rates for Smooth CNA dataset. Panel (a) presents plots of goodness of split principal component scores of raw (red line) and standardised variables (blue line) for Smooth CNA dataset. Panel (b) presents plots of prediction error for Classification Trees of the set of the first j principal component scores of raw (red line) and standardised variables (blue line) for Smooth CNA dataset. It should be emphasized here that the horizontal axis for these two panels are different. For Panel (a), the horizontal axis corresponds with each individual principal component score. For Panel (b), the horizontal axis corresponds with the set of the first j-th principal component scores.

Turning now to the discussion on how standardising affects the construction of Classification Trees using principal component scores of standardised datasets for DNACopy CNA dataset. We begin by presenting Figure 5.16 followed by Figures 5.17 and 5.18. These three figures present how standardising DNACopy CNA dataset before applying PCA yields the same results as for Smooth CNA dataset.

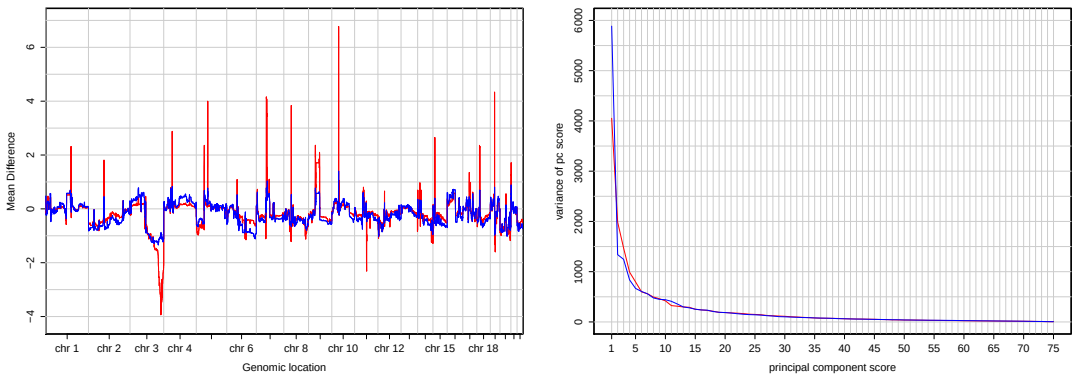


(a) (b)

Figure 5.16: Plots of within group variance for original variables and standardised variables of observations of Class 1 and Class 2 of DNACopy CNA dataset. Panel (a) presents plots of within group variance for original variables (red line) and standardised variables (blue line) of observations of Class 1 and Panel (b) presents plots of within group variance for original variables (red line) and standardised variables (blue line) of observations of Class 2 of DNACopy CNA dataset. For both panels, the horizontal axis corresponds with genomic regions.

As previously explained for Smooth CNA dataset, for DNACopy CNA dataset, we also find that standardising dataset prior to applying PCA declines the performance the resulting principal component scores while carrying out the classification task. We end up with this result for DNACopy CNA dataset for the same reasons as for Smooth CNA dataset.

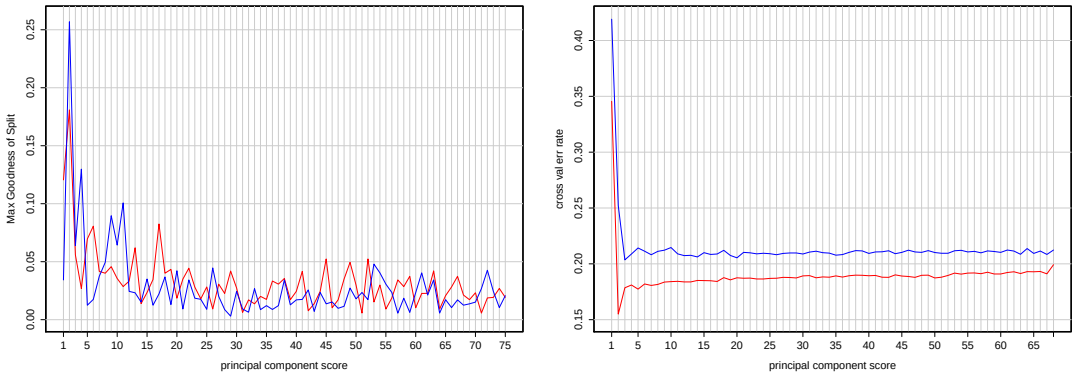
Firstly, standardising increases within group variances of variables which have



(a) (b)

Figure 5.17: Plots for mean difference of observations from Class 1 and Class 2 using raw and standardised variables along with plots of goodness of split principal component scores of raw and standardised variables for DNACopy CNA dataset. Panel (a) shows plots for mean difference of observations from Class 1 and Class 2 using raw (red line) and standardised variables (blue line) for DNACopy CNA dataset. For this panel, the horizontal axis corresponds with genomic regions. Panel (b) presents plots of variance for principal component scores of raw (red line) and standardised variables (blue line) for DNACopy CNA dataset. For this panel, the horizontal axis corresponds with each individual principal component score.

variances less than 1 as shown in Figure 5.16. Secondly, standardising reduces the values of mean difference of variables which have high variances. This can be seen in Panel (a) of Figure 5.17. As the consequence of these two reasons, principal component scores of standardised DNACopy dataset have higher variances than principal component scores of unstandardised one. One might find this result in Panel (b) of Figure 5.17.



(a) (b)

Figure 5.18: Plots of goodness of split principal component scores of raw and standardised variables along with the plots of their error rates for DNACopy CNA dataset. Panel (a) presents plots of goodness of split principal component scores of raw (red line) and standardised variables (blue line) for DNACopy CNA dataset. Panel (b) presents plots of prediction error for Classification Trees of the set of the first j principal component scores of raw (red line) and standardised variables (blue line) for DNACopy CNA dataset. It should be emphasized here that the horizontal axis for these two panels are different. For Panel (a), the horizontal axis corresponds with each individual principal component score. For Panel (b), the horizontal axis corresponds with the set of the first j-th principal component scores.

However, standardising DNACopy CNA dataset prior to applying PCA leads

to the increase of the maximum goodness of split of the second principal component score which has the highest maximum goodness of split among all principal component scores as can be seen in Panel (a) of Figure 5.18 . Yet, this increase does not improve the performance of the resulting Classification Trees. Panel (b) of Figure 5.18 shows this result.

Having obtained the fitted Classification Trees using principal components of standardised datasets which give higher error rate for both Smooth CNA and DNACopy CNA datasets, it is recommended for us to apply data transformation as another data preprocessing approach.

5.6 Classification Trees of Logarithmic Transformation of Principal Component Scores

In this section we discuss the application of logarithmic transformation of data matrix X prior to applying PCA to obtain principal component scores for constructing Classification Trees. We use this transformation to reduce the effect of several extremely high variance variables in both Smooth CNA and DNACopy CNA datasets.

Logarithmic transformation requires the data to be positive, hence we have to apply a data preprocessing step prior to data transformation. For $x_{i,j} \in X$ as the measurement for the j-th variable on the i-th observation, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, p$, we define three preprocessing steps as follows:

1. $x^*_{i,j} = x_{i,j} + c_1$, where $c_1 > \min\{x_{i,j}\}$,
2. $x^*_{i,j} = \max\{x_{i,j}, d_2\} + c_2$, where $c_2 > \min\{x_{i,j}, d_2\}$ and $\min\{x_{i,j}\} \leq d_2 \leq 0$,
3. $x^*_{i,j} = \min\{x_{i,j}, d_3\} + c_3$, where $c_3 > \min\{x_{i,j}, d_3\}$ and $0 \leq d_3 \leq \min\{x_{i,j}\}$.

We define the logarithmic transformation for the preprocessed data as

$$\widetilde{x}_{i,j} = \log x^*_{i,j} \tag{5.6.1}$$

for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, p$.

We propose the first preprocessing step as an attempt to shift all measurement values into positive values. For this reason, we name the constant c_1 as shifting constant. Moreover, for both the second and the third pre-processing steps we involve replacement constants, d_2 and d_3, in addition to the shifting constants c_2 and c_3. In pre-processing step 2, we replace the large negative values of $x_{i,j}$ with a constant d_2. Meanwhile, in pre-processing step 3, we replace the large positive values of $x_{i,j}$ with a constant d_3. We apply these two preprocessing step as an attempt to reduce the effect of either large negative of large positive values of $x_{i,j}$.

We obtain all of these shifting and replacement constants by maximising the sum of p-values from normality test carried on the logarithmic transformed data variable-wise. For computational purpose, we implement **optim** function of R. We present the resulting constants for Smooth CNA and DNACopy CNA datasets in Table 5.1 and 5.2, respectively.

Table 5.1 indicates that both preprocessing 1 and 2 for Smooth CNA dataset produce the same result. Using the replacement constant $d_2 = -2.58$, which is the minima of $x_{i,j}$, applying preprocessing 2 yields the same result as preprocessing 1. Meanwhile applying preprocessing 3, large number of $x_{i,j}$ values are replaced by 3 before added by the shifting constant.

Table 5.1: Shifting and replacement constants for preprocessing steps of Smooth CNA dataset

Pre-processing	Shifting constant	Replacement constant
1	2.6	-
2	2.6	-2.58
3	2.6	3

Moreover, Table 5.2 indicates that all preprocessing steps for logarithmic transformation of DNACopy CNA dataset give the same result. Using the replacement constants $d_2 = -7.4$, which is the minima of $x_{i,j}$, and $d_3 = 58$, which is the maxima of $x_{i,j}$, for both preprocessing steps 2 and 3, respectively, leads to the same result as preprocessing 1. Hence, for DNACopy datasets, we only consider one out of three preprocessing steps used.

Table 5.2: Shifting and replacement constants for preprocessing steps of DNA-Copy CNA dataset

Pre-processing	Shifting constant	Replacement constant
1	7.5	-
2	7.5	-7.4
3	7.5	58

Turning now on discussing the results obtained by transforming Smooth CNA dataset. We present the results for this dataset in Figures 5.19, 5.20, 5.21, 5.22 and 5.23. Figure 5.19 shows the plots of variance and between group variance for untransformed and transformed Smooth CNA dataset.

On general, these plots indicates that transforming this dataset leads to the decrease of its variance. Applying the logarithmic transformation on preprocessed data using the three preprocessing steps proposed gives logarithmic transformed data with lower variance. This in turn leads to the decrease of variance of the estimated principal components of the transformed dataset as shown in Figure 5.20.

However, this decrease of variance of these transformed datasets does not improve the performance of their estimated principal components to carry out data classification task. For preprocessing 1 and 2, applying these two preprocessing steps and transforming the preprocessed data yield the estimated principal components whose less maximum goodness of split than the components of untransformed data. This can be seen in Figure 5.21.

Furthermore, this figure shows how applying preprocessing 3 and logarithmic transformation gives principal components whose almost the same discriminative ability as the components of untransformed data. The second principal component of the transformed data with preprocessing 3 get the slightly lower goodness of split than the second component of the untransformed data does. The former obtains 0.3545, while the latter attains 0.3585.

As a result, constructing Classification Trees using principal components of logarithmic transformed dataset with either preprocessing step 1 or preprocessing step 2 leads to significant increase of error rate. Meanwhile, fitting the trees using

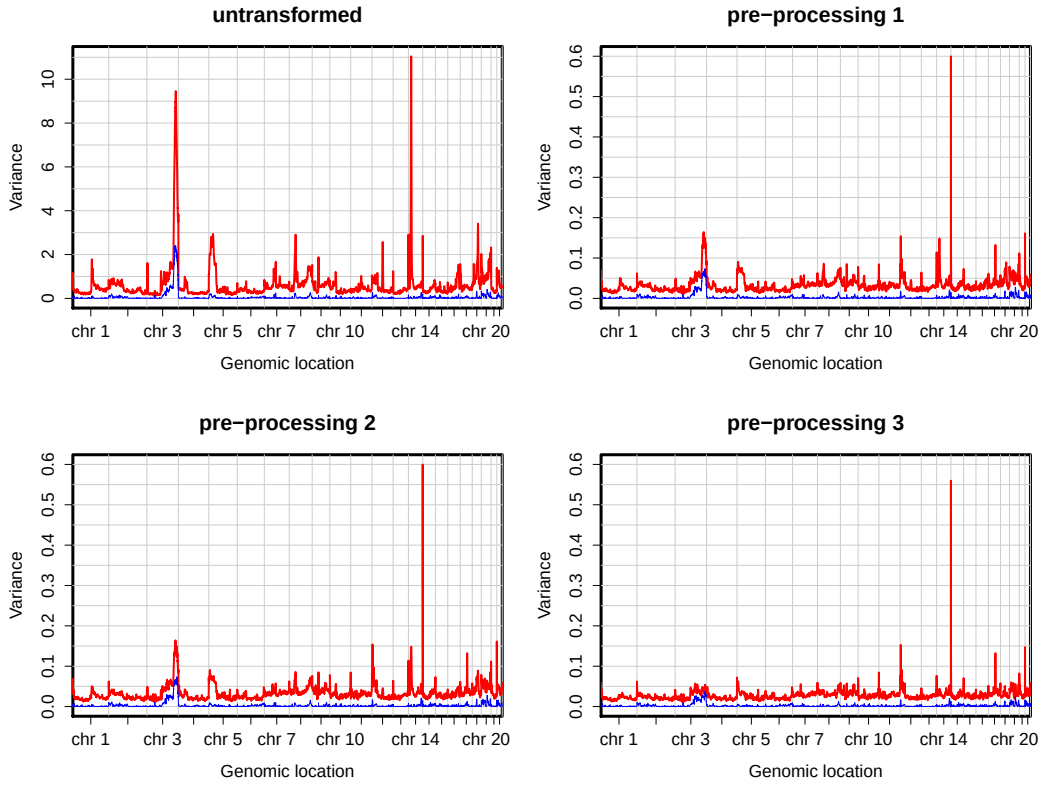


Figure 5.19: Plots for comparing the overall variance (red line) and between group variance (blue line) for untransformed and transformed Smooth CNA dataset with preprocessing 1, 2 and 3. For these four panels, the horizontal axis corresponds with genomic regions.

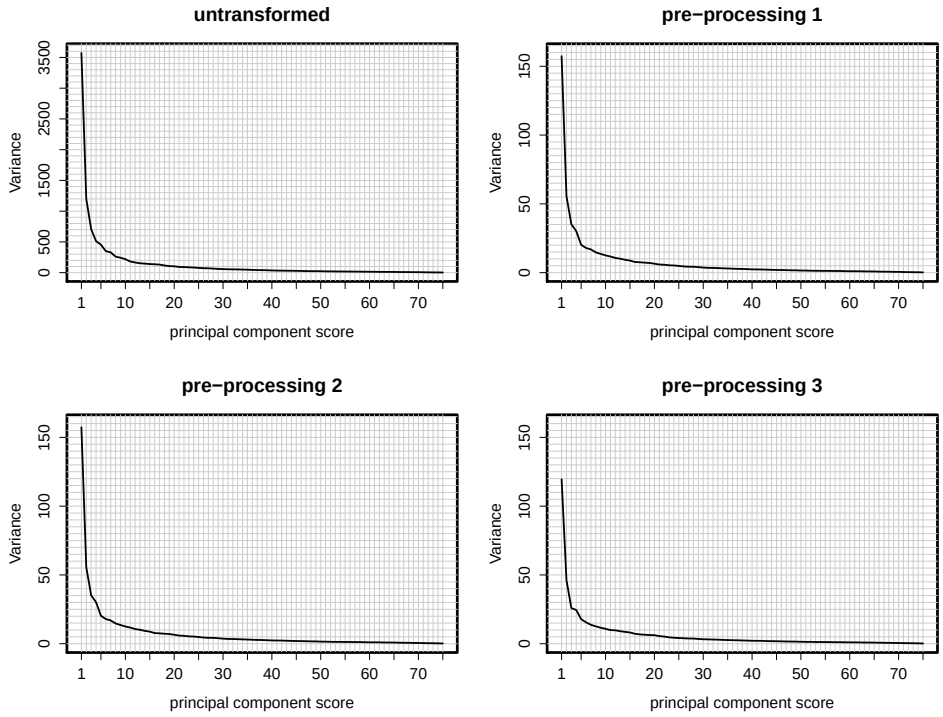


Figure 5.20: Plots for variances of principal component scores for untransformed and transformed Smooth CNA dataset with pre-processing 1, 2 and 3. For these four panels, the horizontal axis corresponds with each individual principal component score.

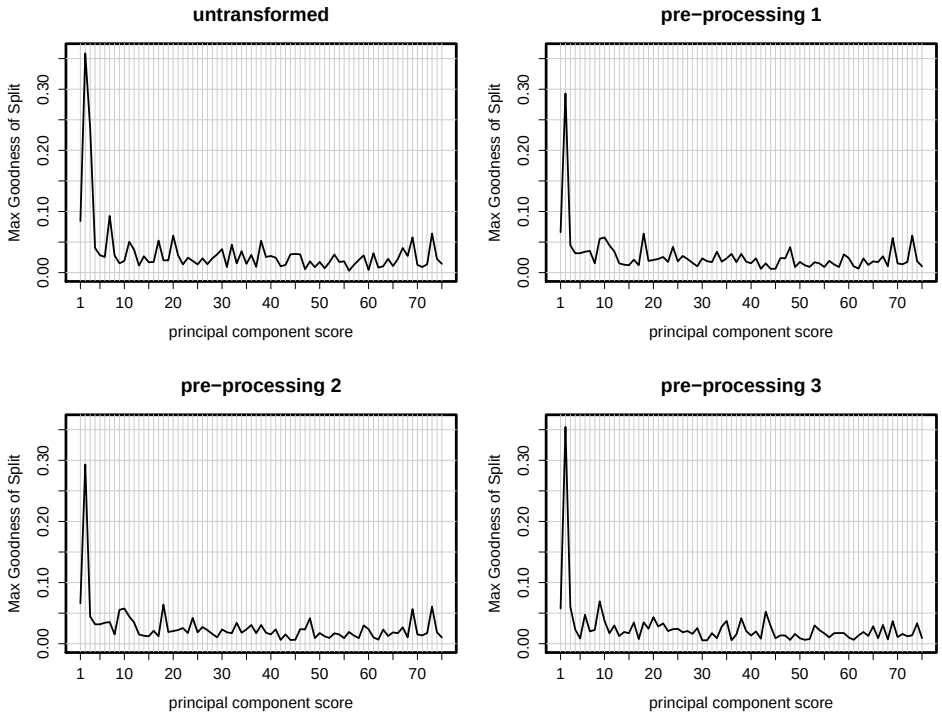


Figure 5.21: Plots for the maximum goodness of split values of principal component scores for untransformed and transformed Smooth CNA dataset with pre-processing 1, 2 and 3. For these four panels, the horizontal axis corresponds with each individual principal component score.

components of transformed data with preprocessing step 3 gives slightly different error rate than building the trees with components of the untransformed dataset. We present these results using the plots in Figures 5.22 and 5.23. Figure 5.22 presents the plots of the estimated error rate of Classification Trees for the set of the first j principal components of untransformed and transformed datasets. These error rate estimates are obtained using 100-repeated 10-fold stratified cross validation.

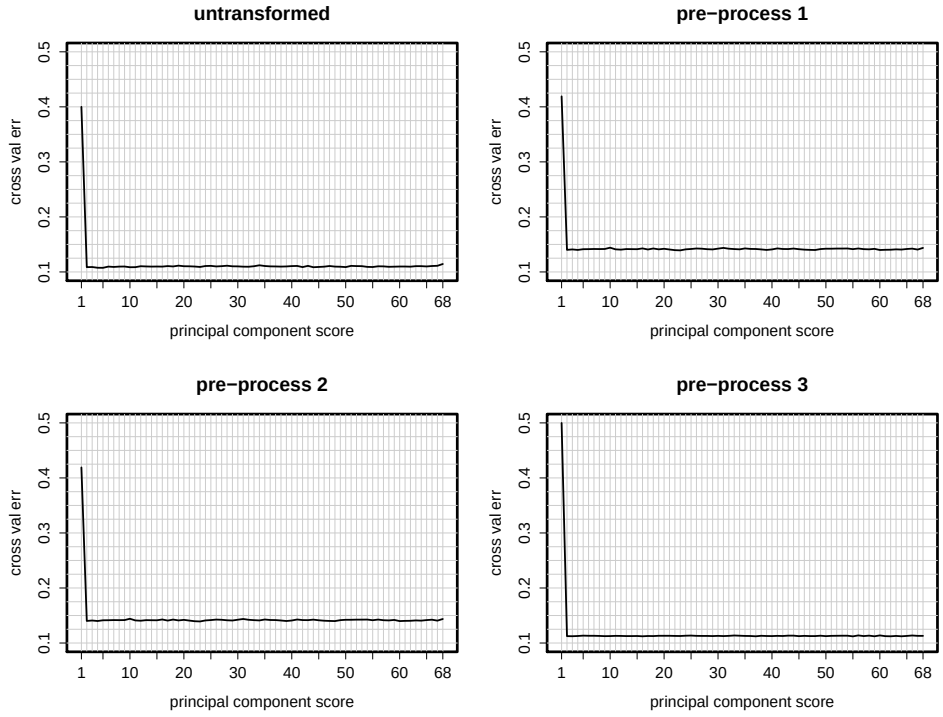


Figure 5.22: Plots of prediction error for Classification Trees of the set of the first j principal component scores of untransformed and transformed Smooth CNA dataset with pre-processing 1, 2 and 3. For these four panels, the horizontal axis corresponds with the set of the first j-th principal component scores, for $j = 1, 2, \ldots, 68$.

Figure 5.23 presents the comparison between the accuracy for Classification Trees of raw dataset, Classification Trees of principal components of untransformed dataset and Classification Trees of principal components of transformed

dataset for Smooth CNA dataset. Apart from boxplots for error rate of Classification Trees fitted using principal components of untransformed data and transformed data separately, this figure also presents boxplots for paired-difference data between the error rates of Classification Trees fitted using principal components of untransformed data and transformed data. This figure indicates that transforming the dataset to improve its individual variable normality prior to applying PCA does not give advantage. The resulting Classification Trees are less accurate than the ones obtained from the raw datasets.

To make inference on these results, we perform paired sample test for the comparison between the resulting error rates of Classification Trees fitted by the principal components of the raw dataset and those of Classification Trees built by the principal components of the transformed dataset. We present Table 5.3 to display the result obtained from performing Shapiro-Wilk test of normality and paired sample Wilcoxon test.

For the three paired sample Wilcoxon tests carried out, we define the null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero. We use this alternative hypothesis since we wish to obtain the lower error rates for the resulting Classification Trees of principal components by transforming variable prior to applying PCA.

Table 5.3: P-values for Shapiro-Wilk test and Wilcoxon test

Comparison	Shapiro-Wilk test	Wilcoxon test
PCA - Tr1 & PCA	0.00141	1
PCA - Tr2 & PCA	0.00141	1
PCA - Tr3 & PCA	0.02713	0.95814

Table 5.3 shows that all the paired difference data are not normally distributed. Moreover, from Table 5.3, it can be inferred that we fail to reject H_0 that the median of the difference between the pairs equals zero. Therefore, for Smooth CNA dataset, it can be concluded that transforming the data prior to applying PCA does not improve the performance of the Classification Trees of the resulting principal components.

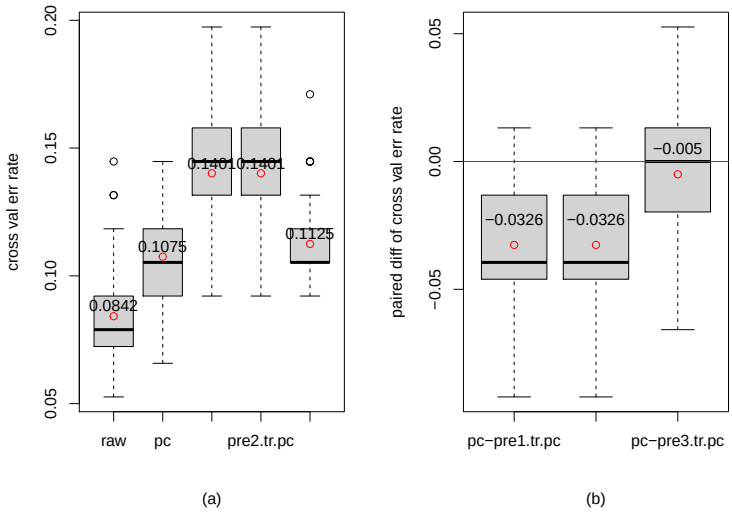


Figure 5.23: Boxplots for cross validated error rate of Classification Trees of raw dataset, principal component scores of raw dataset and principal component scores of transformed dataset for Smooth CNA dataset along with their paired difference data. Panel (a) shows boxplots for error rate estimates of Classification Trees fitted using raw dataset (the first boxplot), set of the first two principal components of untransformed data (the second boxplot), set of the first two principal components of transformed data with pre-processing 1 (the third boxplot), set of the first two principal components of transformed data with pre-processing 2 (the fourth boxplot) and set of the first two principal components of transformed data with pre-processing 1 (the fifth boxplot). All boxplots represent 100 estimates of Classification Trees error rate obtained using 100-repeated 10-fold cross validation estimation. Panel (b) displays paired-difference data between these sets of the error rate estimates. The red horizontal line in this panel corresponds with paired-difference equals 0. Using paired sample Wilcoxon Test with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero, we obtain p-value equals 1 for the first two comparisons and p-value equals 0.95814 for the third one.

Let us now discuss the results obtained by transforming DNACopy CNA dataset. We present the results for this dataset in Figures 5.24, 5.25, 5.26, 5.27 and 5.28. Figure 5.24 presents the plots of variance and between group variance for untransformed and transformed DNACopy CNA dataset. Overall, these plots show that transforming this dataset leads to the decrease of its variance. This decrease then gives the decline of variance for the estimated principal components of the transformed dataset as shown in Figure 5.25.

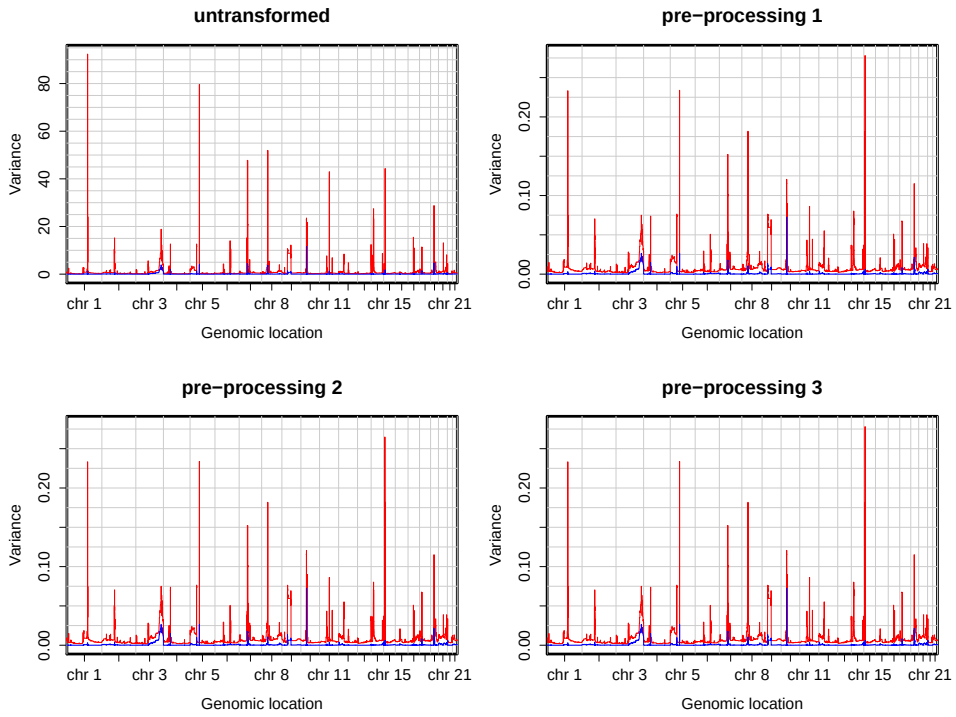


Figure 5.24: Plots for comparing the overall variance (red line) and between group variance (blue line) for untransformed and transformed DNACopy CNA dataset with preprocessing 1, 2 and 3. For these four panels, the horizontal axis corresponds with genomic regions.

Moreover, Figure 5.26 presents the plots of the maximum goodness of split for the principal components of either untransformed or transformed datasets. This figure indicates the increase of maximum goodness of split of the second principal

component of the transformed datasets. The second component of transformed dataset gets 0.245, while the one of the untransformed dataset attains 0.18.

However, for the transformed dataset, we obtain the first principal component whose lower maximum goodness of split than we get for the component of the untransformed dataset. We obtain 0.066 for the former and 0.12 for the latter. On the other hand, as previously explained in Subsection 5.4.2, for DNACopy CNA dataset, along with the second principal component, the first component has contribution in producing the accurate Classification Trees. Removing this component and constructing Classification Trees using the set of the rest components yield less accurate trees as shown in Panel (b) of Figure 5.9.

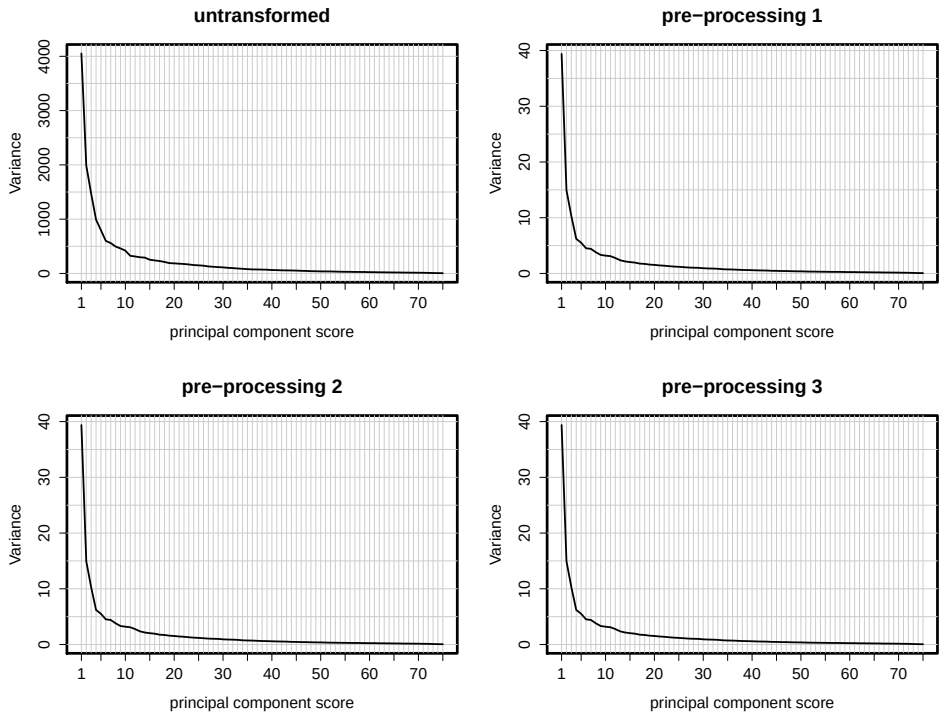


Figure 5.25: Plots for variances of principal component scores for untransformed and transformed DNACopy CNA dataset with pre-processing 1, 2 and 3. For these four panels, the horizontal axis corresponds with each individual principal component score.

As previously mentioned, we get this result due to the fact that, for the un-

transformed DNACopy CNA dataset, the second principal component is not a strong relevant predictor. Meanwhile, for the transformed DNACopy dataset, having obtained the second principal component whose maximum goodness of split 0.245 indicates that the resulting component is not the strong relevant predictor either. Therefore, in the absence of another relevant predictor, the second principal component of the transformed dataset does not have good performance in carrying out data classification. Figures 5.27 and 5.28 show how transforming the data does not give advantage. Figure 5.27 presents the plots of the estimated error rate of Classification Trees for the set of the first j principal components of untransformed and transformed DNACopy CNA dataset. These error rate estimates are obtained using 100-repeated 10-fold stratified cross validation.

Figure 5.28 shows the comparison between the accuracy of Classification Trees of raw dataset, Classification Trees of principal components of untransformed dataset and Classification Trees of principal components of transformed dataset for DNACopy CNA dataset. Aside from boxplots for error rate of Classification Trees fitted using principal components of untransformed data and transformed data separately, this figure also presents boxplots for paired-difference data between the error rates of Classification Trees fitted using principal components of untransformed data and transformed data. This figure indicates that transforming the dataset to improve its individual variable normality prior to applying PCA does not give advantage. The resulting Classification Trees are less accurate than the ones obtained from the principal components of raw datasets.

To make inference on these results, we perform paired sample test for the comparison between the resulting error rates of the Classification Trees fitted by the principal components of the raw dataset and the error rates of the Classification Trees built by the principal components of the transformed dataset. We present Table 5.4 to show the result obtained from performing Shapiro-Wilk test of normality and paired sample Wilcoxon test.

For the three paired sample Wilcoxon tests carried out, we define the null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero. We use this alternative hypothesis as we wish to obtain the lower

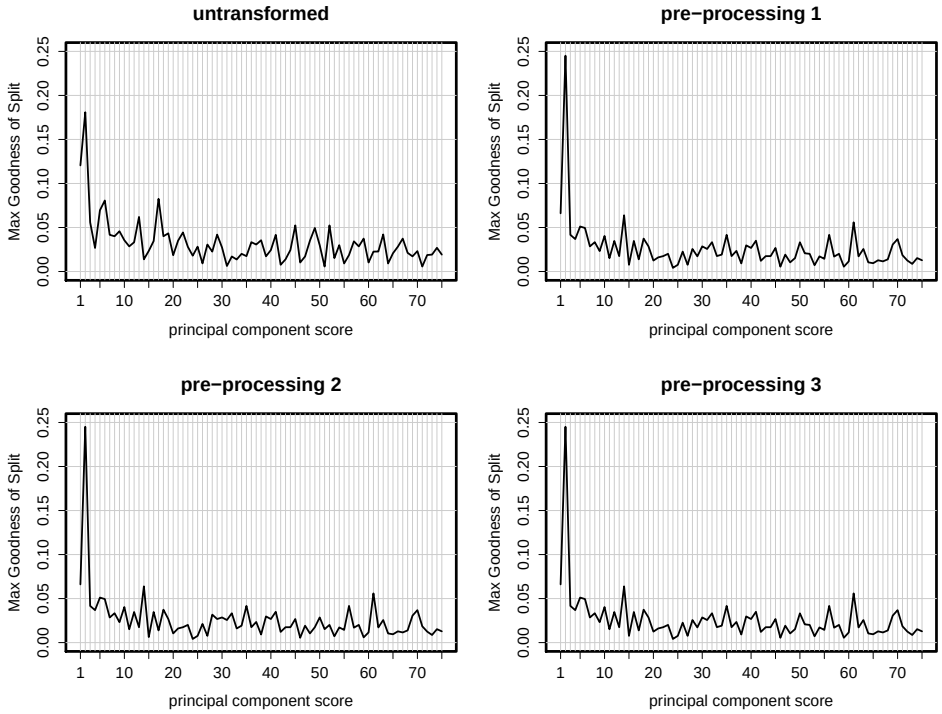


Figure 5.26: Plots for the maximum goodness of split values of principal component scores for untransformed and transformed DNACopy CNA dataset with pre-processing 1, 2 and 3. For these four panels, the horizontal axis corresponds with each individual principal component score.

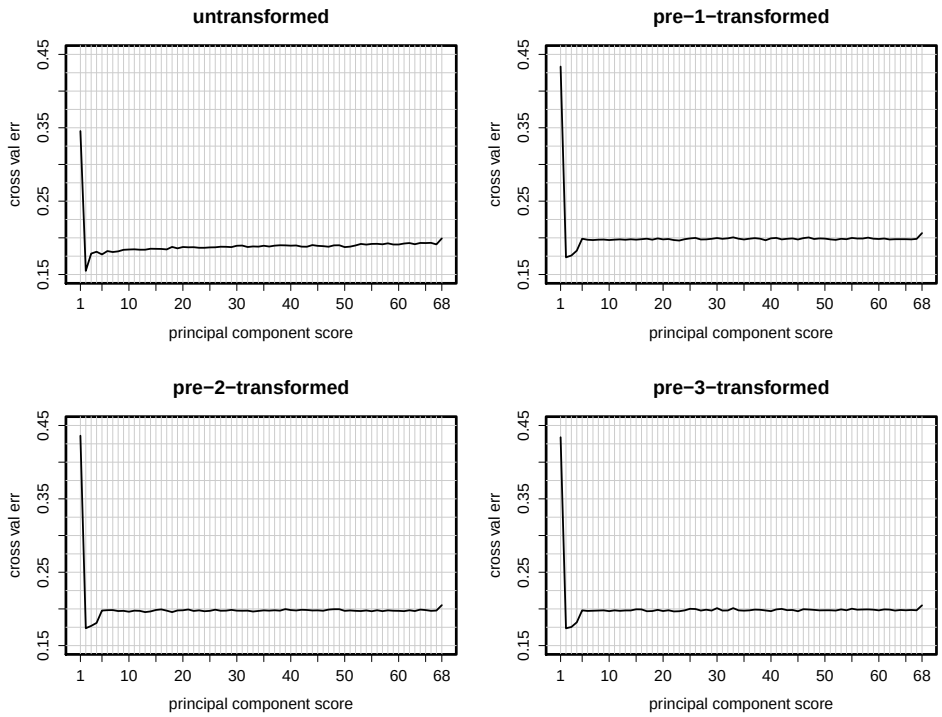


Figure 5.27: Plots of prediction error for Classification Trees of the set of the first j principal component scores of untransformed and transformed DNACopy CNA dataset with pre-processing 1, 2 and 3. For these four panels, the horizontal axis corresponds with the set of the first j-th principal component scores, for $j = 1, 2, \ldots, 68$.

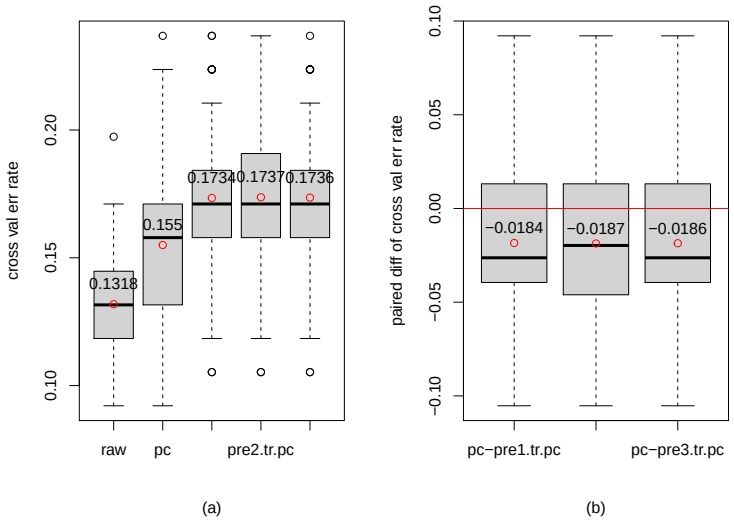


Figure 5.28: Boxplots for cross validated error rate of Classification Trees of raw dataset, principal component scores of raw dataset and principal component scores of transformed dataset for DNACopy CNA dataset along with their paired difference data. Panel (a) shows boxplots for error rate estimates of Classification Trees fitted using raw dataset (the first boxplot), set of the first two principal components of untransformed data (the second boxplot), set of the first two principal components of transformed data with pre-processing 1 (the third boxplot), set of the first two principal components of transformed data with pre-processing 2 (the fourth boxplot) and set of the first two principal components of transformed data with pre-processing 1 (the fifth boxplot). All boxplots represent 100 estimates of Classification Trees error rate obtained using 100-repeated 10-fold cross validation estimation. Panel (b) displays paired-difference data between these sets of the error rate estimates. The red horizontal line in this panel corresponds with paired-difference equals 0. Using paired sample Wilcoxon Test with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero, we obtain p-value equals 0.999 for all paired comparisons.

error rates for the Classification Trees of principal components by transforming variable prior to applying PCA.

Table 5.4: P-values for Shapiro-Wilk test, Wilcoxon test, and paired-t test for comparing the error rates of the Classification Trees of principal components of untransformed and transformed datasets for DNACopy CNA dataset

Comparison	Shapiro-Wilk test	Wilcoxon test	paired-t test
PCA - Tr1 & PCA	0.21855	0.99999	0.99999
PCA - Tr2 & PCA	0.24193	0.99999	0.99999
PCA - Tr3 & PCA	0.20829	0.99999	0.99999

Table 5.4 shows that all the paired difference data are normally distributed. Moreover, from Table 5.4 it can be inferred that we fail to reject H_0 that the median of the difference between the pairs equals zero. Therefore, for DNACopy CNA dataset, it can be concluded that transforming the data prior to applying PCA does not improve the performance of the Classification Trees of the resulting principal components. In addition, we end up with the same conclusion from performing paired-sample t test.

5.7 Classification Trees of Principal Components of Simulated Datasets

In this section we discuss the results obtained from applying Classification Trees of principal component scores on simulated datasets. More detail simulation setting used in this section can be found in Section 2.4 of Chapter 2 and partly in Section 4.5 of Chapter 4. Generally, there are three quantities are considered: correlation within-blocks of variables, mean difference between sub-population and standard deviation of each single variable.

Having generated datasets in the forms of data matrix of the size 100×150, we have the matrix for the principal component scores of the size 100×99 as the simulated data matrix has 99 degree of freedom. However, instead of presenting the prediction error rates for Classification Trees fitted using the first jth principal components, for $j = 1, 2, \ldots, 99$, we only show 15 of them. We find similar results from using the set of the first j-th principal components, for $j = 16, 17, \ldots, 99$.

Let us now consider the results obtained from this simulation study. We present these results in the following 12 figures, from Figure 5.29 up to Figure 5.40. The first six figures show the results for one standard deviation setting, while the remaining six figures display the results for three standard deviation. All of these figures consist of either six or four panels. Each panel contains 15 boxplots for 100 estimates of test set error rate of Classification Trees fitted using the first j-th principal components, for $j = 1, 2, \ldots, 15$.

In general, assuming that both standard deviation and correlation remain constant, it is obvious that the increase in mean difference of generated datasets causes the increase in the accuracy level of Classification Trees fitted using the resulting principal component scores. On contrast, assuming that both standard deviation and mean difference remain constant, the increase in standard deviation leads to the decrease in the accuracy level of Classification Trees of the resulting principal components.

However, we can not obtain one general pattern of our simulation results by considering the change in the correlation setting. Instead, there are two distinct patterns of the results obtained by considering this quantity. Firstly, regardless the number of principal components used to fit Classification Trees, one always

ends up with the trees of the same accuracy level. Secondly, using the first three principal components for fitting the trees leads to those of high test set error rate, while using the first fourth and so on leads to those of lower prediction error rate.

The two top panels of Figure 5.29 show the results of the first pattern. Meanwhile, the two bottom panels of this figure display the results of the second pattern. Setting the correlation to become small, one is more likely to obtain the results of the first pattern. Whereas, setting the correlation to become large, one is more likely to attain the results of the second pattern. However, this is not always be the case.

Let us now consider the cases presented in both Figures 5.29 and 5.30. These figures show boxplots for prediction error rate for Classification Trees of principal components of simulated datasets generated with small mean difference and unit standard deviation. The first figure displays the results for the correlation setting of $\rho = 0, 0.1, \ldots, 0.5$ and the second presents the results for the correlation setting of $\rho = 0.6, 0.7, \ldots, 0.9$.

These two figure show that, for the correlation setting of $\rho = 0$ and 0.1, regardless the number of components, fitting Classification Trees using principal components gives the trees with the same accuracy level. Meanwhile, for the correlation setting of $\rho = 0.3, 0.7, \ldots, 0.9$, one has to fit the Classification Trees using the first four principal components to obtain the more accurate trees. Constructing Classification Tree using either one, two or three principal components, one ends up with an inaccurate trees.

Furthermore, both Figure 5.35 and 5.36 indicate that, except for the correlation setting of 0.9, one always obtains the results where regardless the number of components, fitting Classification Trees using principal components gives the trees with the same accuracy level. These two figures present the results for small mean difference and three standard deviation.

Moreover, both Figure 5.33 and 5.34 show that for all correlation settings, one always obtains the results where regardless the number of components, fitting Classification Trees using principal components gives the trees with the same accuracy level. These two figure present the results for large mean difference and unit standard deviation.

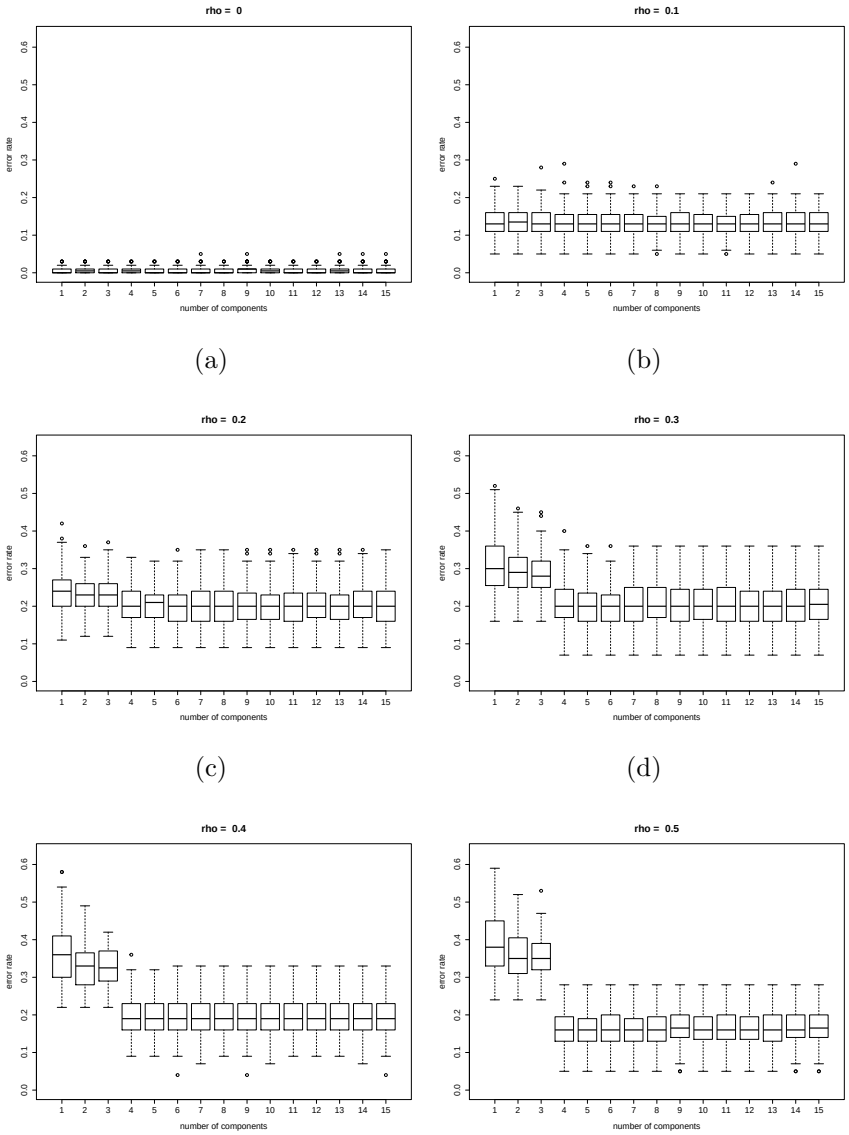
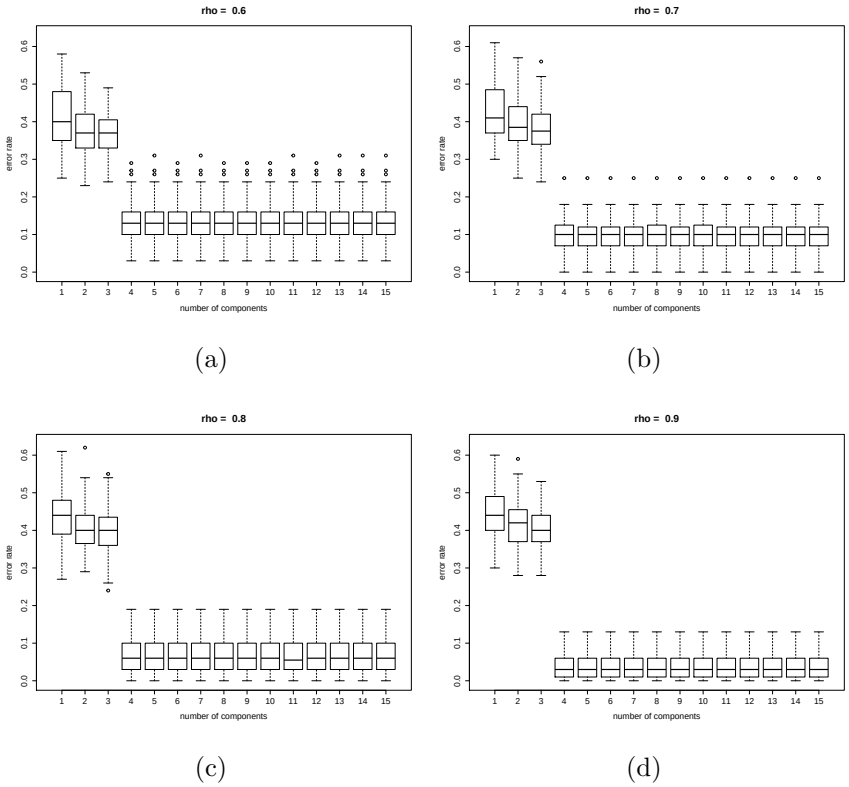


Figure 5.29: Boxplots of test set error rates of Classification Trees fitted using principal components of simulated datasets with small mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0, 0.1, \ldots, 0.5$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the set of the first j-th principal components for j= 1,2,...,15.



(a) (b)

(c) (d)

Figure 5.30: Boxplots of test set error rates of Classification Trees fitted using principal components of simulated datasets with small mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0.6, 0.7, \ldots, 0.9$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the set of the first j-th principal components for j= 1,2,. . .,15.

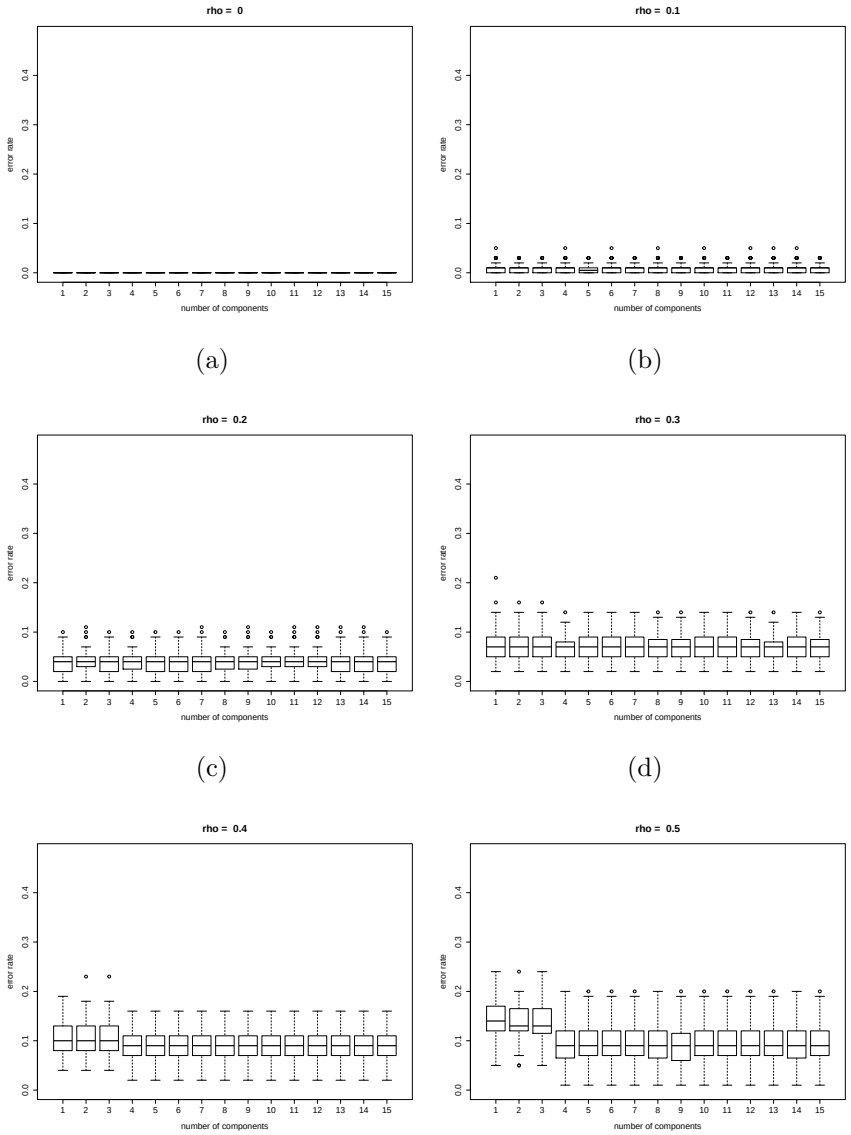


Figure 5.31: Boxplots of test set error rates of Classification Trees fitted using principal components of simulated datasets with medium mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0, 0.1, \ldots, 0.5$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the set of the first j-th principal components for j= 1,2,. . .,15.

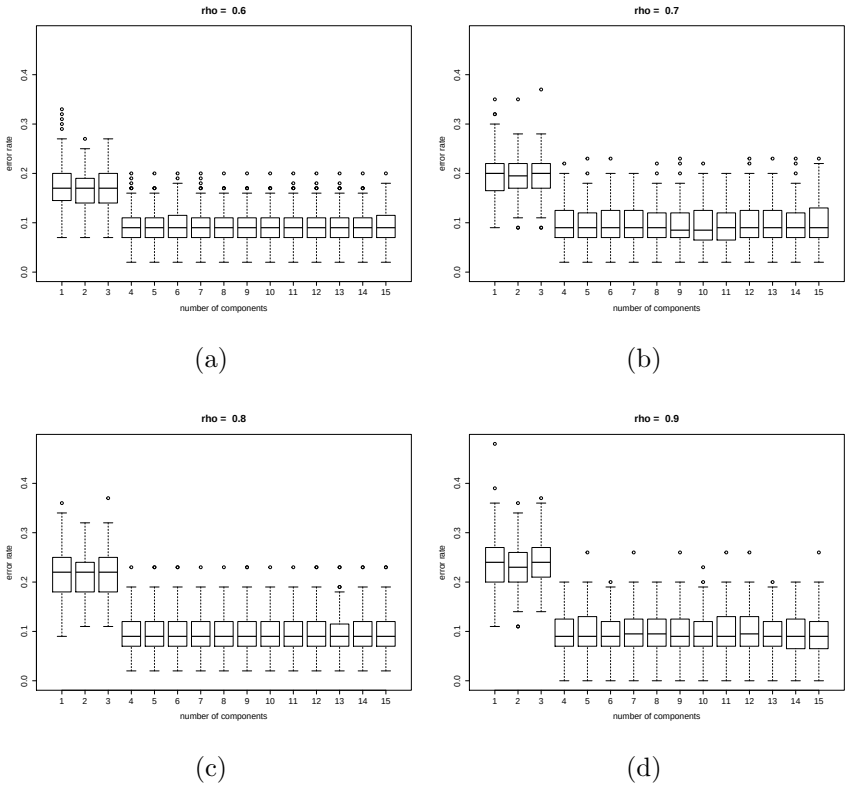
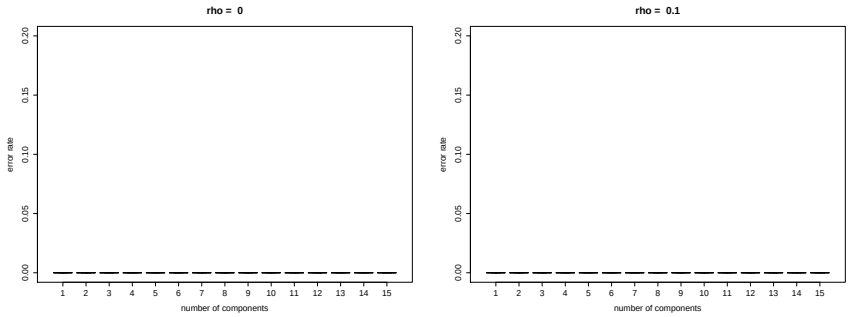
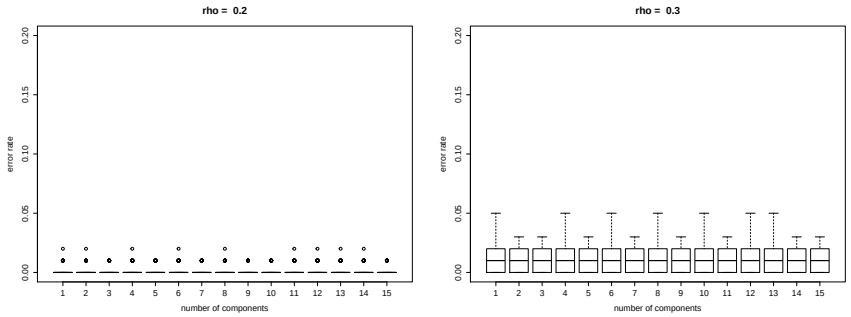


Figure 5.32: Boxplots of test set error rates of Classification Trees fitted using principal components of simulated datasets with medium mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0.6, 0.7, \ldots, 0.9$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the set of the first j-th principal components for j= 1,2,. . .,15.



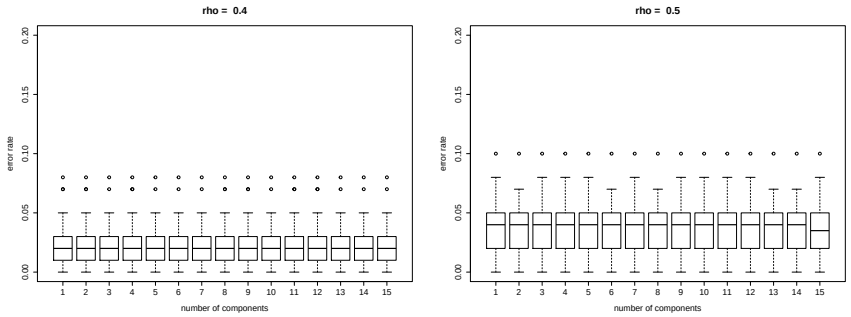
(a)

(b)



(c)

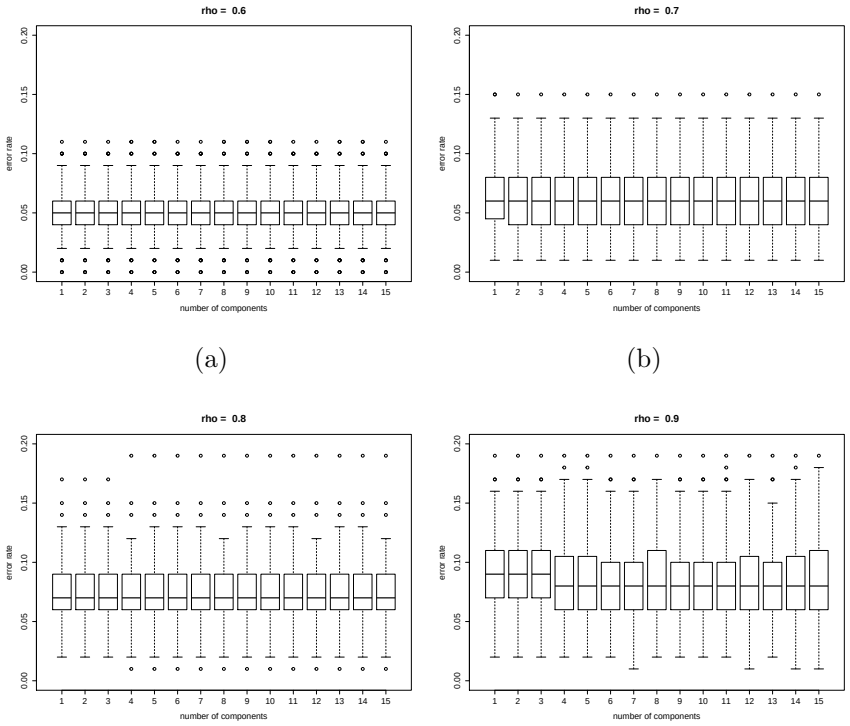
(d)



(e)

(f)

Figure 5.33: Boxplots of test set error rates of Classification Trees fitted using principal components of simulated datasets with large mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0, 0.1, \ldots, 0.5$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the set of the first j-th principal components for j= 1,2,...,15.



(a) (b)

(c) (d)

Figure 5.34: Boxplots of test set error rates of Classification Trees fitted using principal components of simulated datasets with large mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0.6, 0.7, \ldots, 0.9$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the set of the first j-th principal components for j= 1,2,. . .,15.

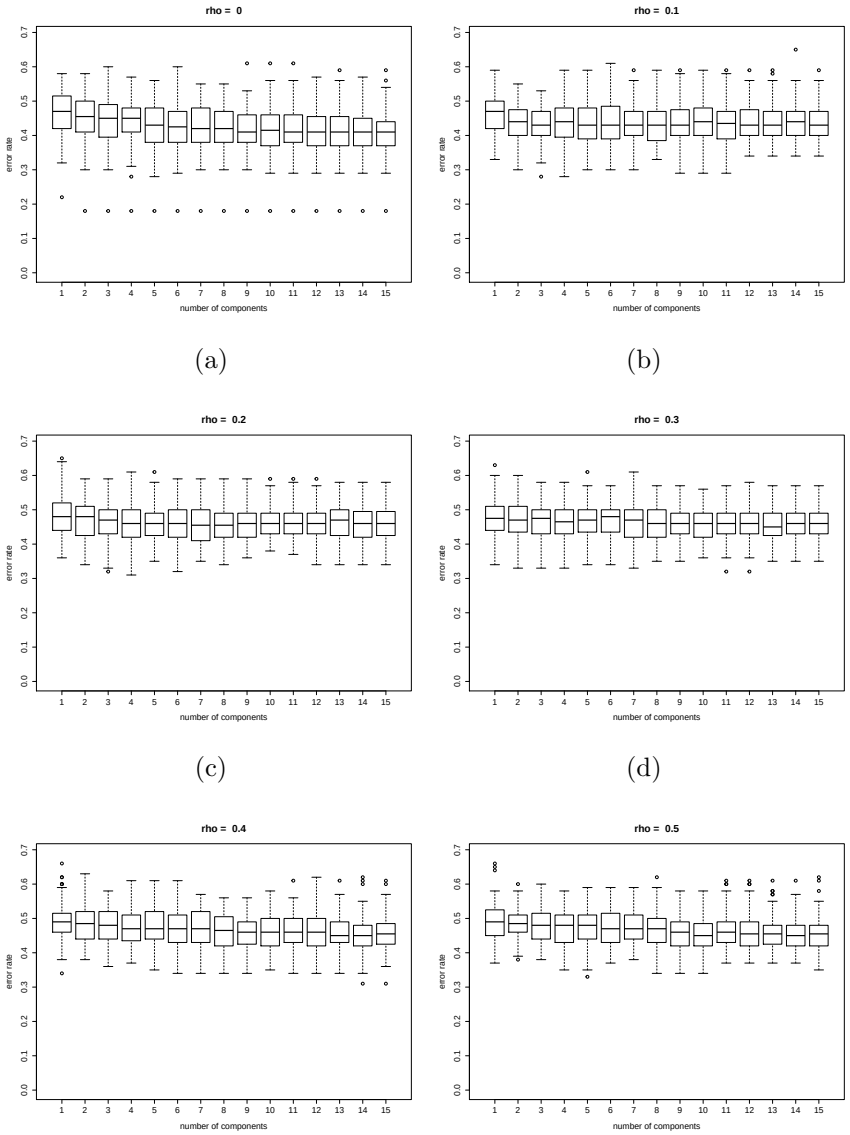
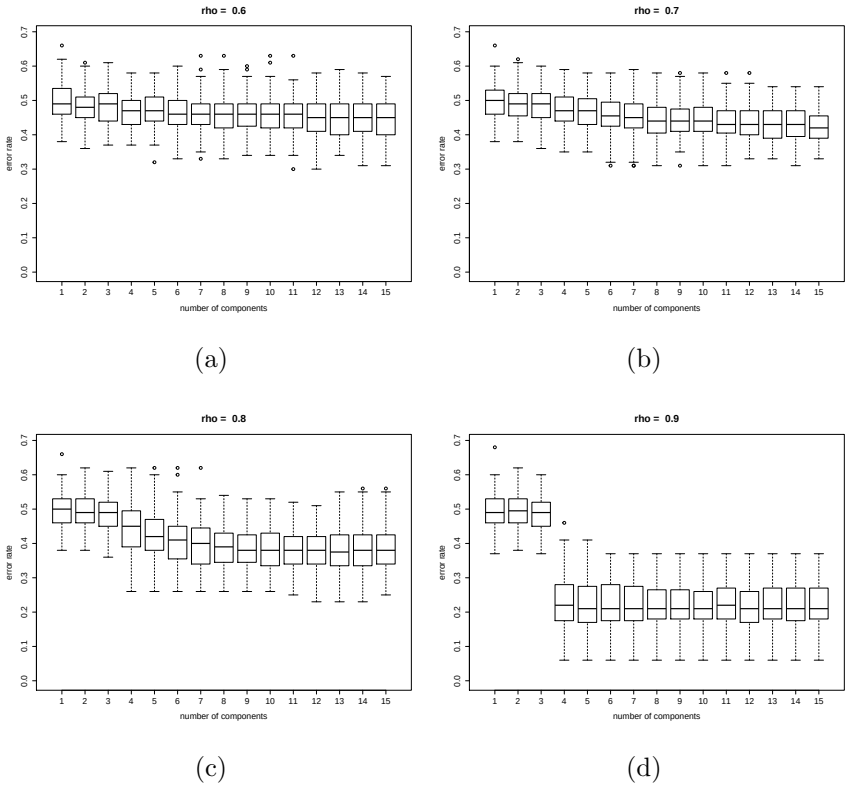


Figure 5.35: Boxplots of test set error rates of Classification Trees fitted using principal components of simulated datasets with small mean difference and three standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0, 0.1, \ldots, 0.5$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the set of the first j-th principal components for j= 1,2,. . .,15.



(a) (b)

(c) (d)

Figure 5.36: Boxplots of test set error rates of Classification Trees fitted using principal components of simulated datasets with small mean difference and three standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0.6, 0.7, \ldots, 0.9$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the set of the first j-th principal components for j= 1,2,. . .,15.

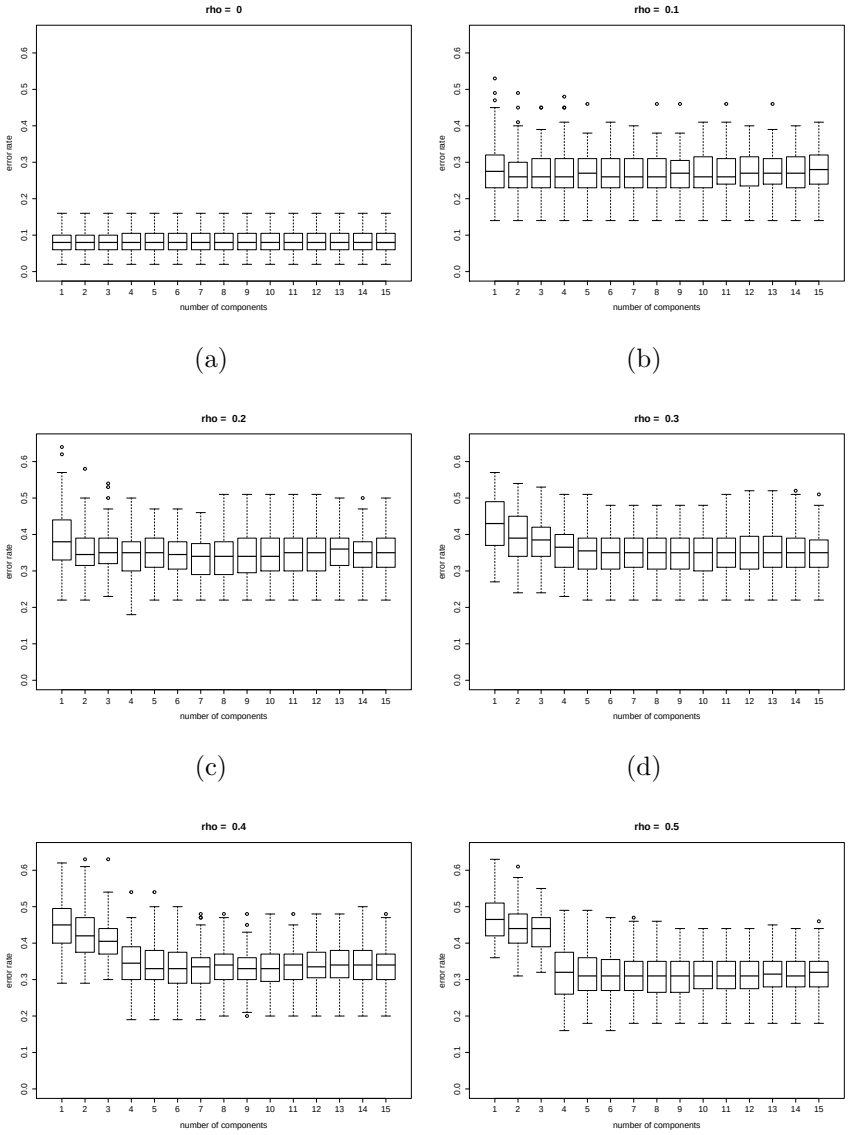
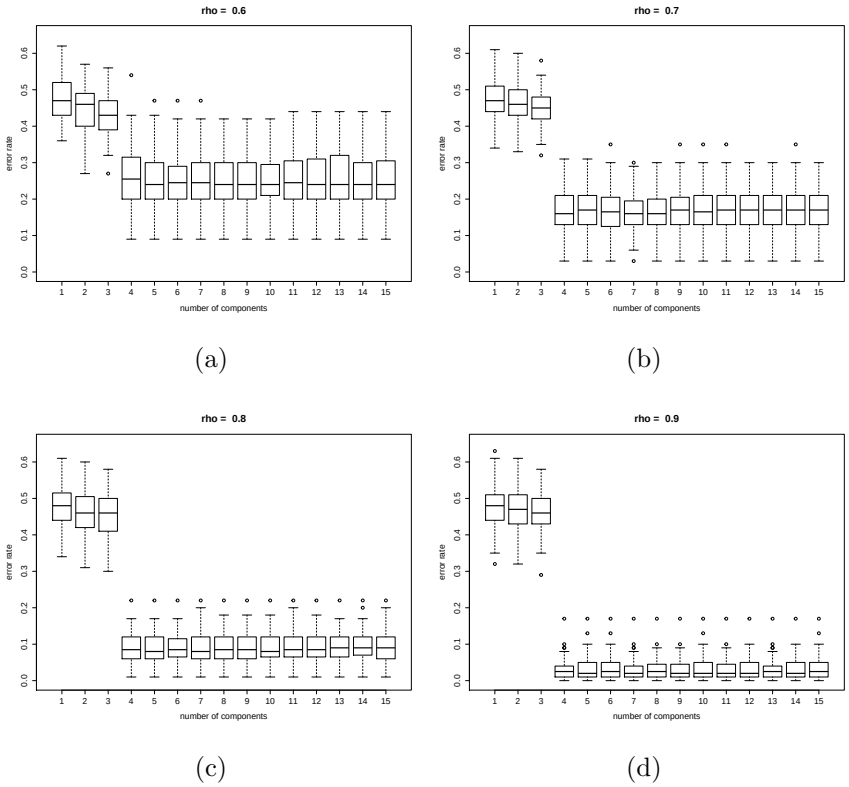


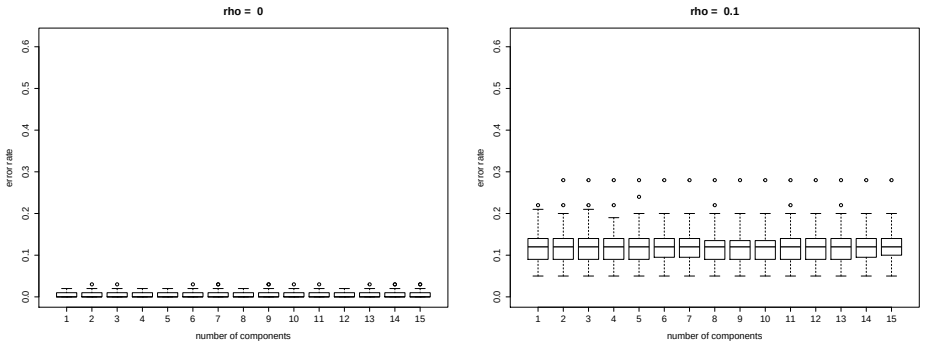
Figure 5.37: Boxplots of test set error rates of Classification Trees fitted using principal components of simulated datasets with medium mean difference and three standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0, 0.1, \ldots, 0.5$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the set of the first j-th principal components for j= 1,2,. . .,15.



(a) (b)

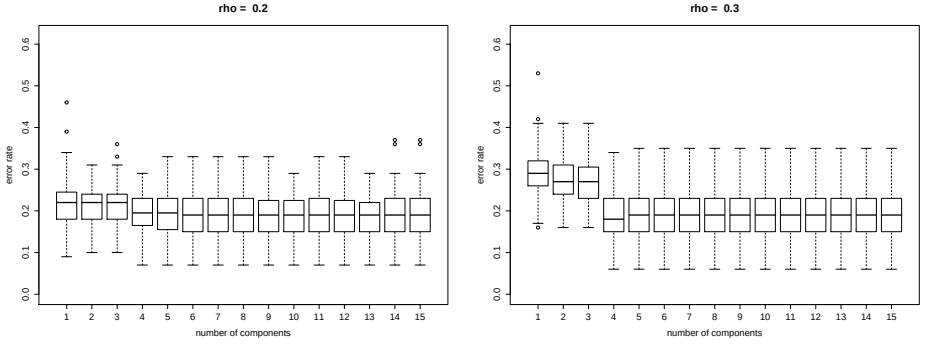
(c) (d)

Figure 5.38: Boxplots of test set error rates of Classification Trees fitted using principal components of simulated datasets with medium mean difference and three standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0.6, 0.7, \ldots, 0.9$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using tthe set of the first j-th principal components for j= 1,2,. . .,15.



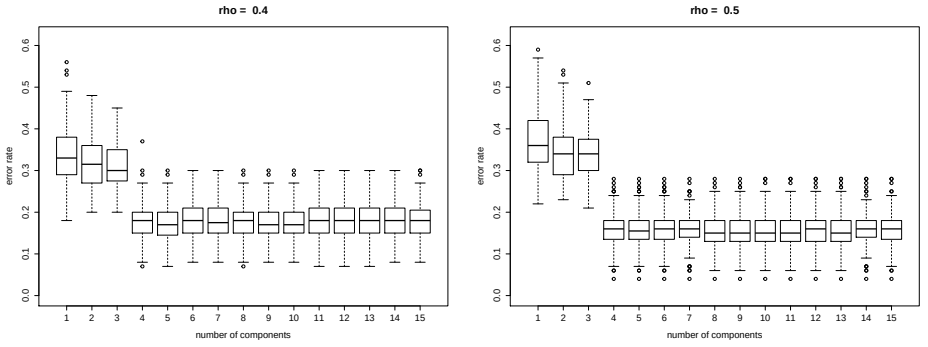
(a)

(b)



(c)

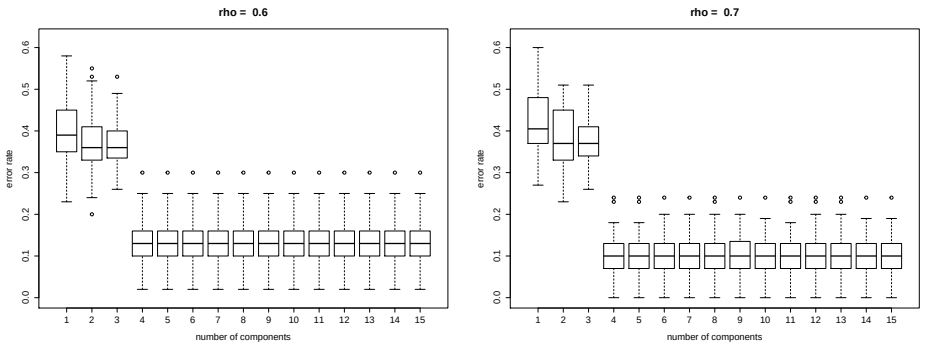
(d)



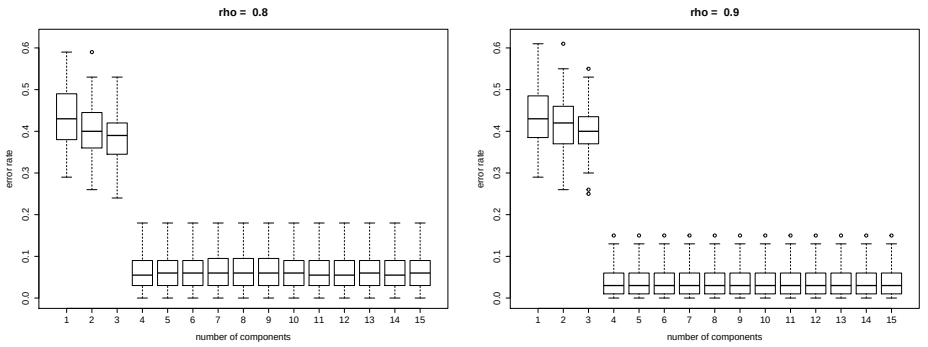
(e)

(f)

Figure 5.39: Boxplots of test set error rates of Classification Trees fitted using principal components of simulated datasets with large mean difference and three standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0, 0.1, \ldots, 0.5$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the set of the first j-th principal components for j= 1,2,. . .,15.



(a) (b)



(c) (d)

Figure 5.40: Boxplots of test set error rates of Classification Trees fitted using principal components of simulated datasets with large mean difference and three standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0.6, 0.7, \ldots, 0.9$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the set of the first j-th principal components for j= 1,2,. . .,15.

All of the results mentioned above might be explained by looking at how PCA works. As discussed earlier, applying PCA on a data matrix, one tries to find the projections that maximises the variance. As a consequence of this, in the presence of variables of large mean difference, one will obtain the covariance matrix that contained large proportion of information on this between group variation. This in turns leads to principal components that contain large proportion of discriminative information which is useful for Classification Trees construction.

Likewise, in the presence of variables with large standard deviation, this data variation information will dominate the covariance matrix and the resulting principal component scores. As a result, the influence of between group variability on the resulting component will be masked by this high standard deviation. This worsens the performing Classification Trees of principal components.

Furthermore, to figure out how the correlation structure affects the performance of Classification Trees fitted using principal component scores, let us now consider the result shown in Figures 5.41 up to 5.44. Figure 5.41 presents the plots for the loadings of principal component obtained using simulated datasets generated using the simulation settings of small mean difference, unit standard deviation and $\rho = 0$. This figure consists six panels. Each panel presents the plot for an individual loading of principal component. In this case we only display the fist six loadings since we are interesting only in investigating the of properties of these six loadings. Each dot in all panels of this plot represents the average from 100 estimates obtained from 100 datasets generated using different seed setting.

This figure indicates that among six loadings shown, it is the first loading that contain a systematic pattern. Whereas the rest loadings only contain random information. This systematic pattern corresponds with the mean difference vector of this dataset. This result shows that for the simulation setting of small mean difference, unit standard deviation and $\rho = 0$, one obtains the first principal component scores as the principal component score whose discriminative information. Therefore, for this simulation setting, adding more principal components do not affect anything on the resulting Classification Trees.

On the other hand, for the simulation setting of small mean difference, unit standard deviation and $\rho = 0.9$, Figure 5.42 indicates that it is the fourth principal component is the one which is responsible for classification. The first three

components correspond with the correlation structures introduced into these simulated datasets.

Panel (a) of this figure shows that the loading for the first component relates to the second and the third blocks of correlated variables. Setting the same correlation value across variables leads to the same contribution of variables of these two blocks on data variation. These two blocks also have higher contribution than the first block of variables as the latter only consists of noise variables without mean difference information. Panel (b) and Panel (c) display the effect of correlation structure as well.

Moreover, one could easily see the effect of mean difference on data variation in the loading of the fourth components. This loading has similar pattern with mean difference structure. As a result of this, whilst fitting Classification Trees one should include the fourth principal component score in order to obtain the trees of low error rate. This explains the pattern shown in the plot of prediction error rate for Classification Trees of simulated datasets with the simulation setting of small mean difference, unit standard deviation and $\rho = 0.9$ presented in bottom right panel of Figure 5.42.

According to the above explanation for the results shown in both Figures 5.41 and 5.42, we can infer that, in the absence of correlation blocks between variables, it is only mean difference as the source of data variation that influences the resulting principal component scores. On the other hand, in the presence of high correlated blocks of variables, the effect of data variation due to these correlated blocks masks the effect of both small mean difference and unit standard deviation on principal component scores.

Furthermore, Figure 5.43 presents the plots for loadings of principal components obtained using simulated datasets generated using the simulation settings of large mean difference, unit standard deviation and $\rho = 0$. This figure shows the same pattern as the plot for loadings of principal component for simulated datasets simulated datasets generated using the simulation settings of small mean difference, unit standard deviation and $\rho = 0$ as explained earlier. Therefore, for this simulation setting, it is only the first component which contains systematic pattern that corresponds with mean difference. In addition, adding more principal components does not affect on the resulting Classification Trees.

Meanwhile, Figure 5.44 presents the plots for loading of principal components obtained using simulated datasets generated using the simulation settings of large mean difference, unit standard deviation and $\rho = 0.9$. We present this plot to explain the results that, for this simulation setting, using the first three principal components one ends with Classification Trees which have slightly higher error rate than using the first fourth components and so on.

It is obvious from Figure 5.44 that the loading for the first component has nearly the same pattern as mean difference vector for this simulation setting. However, the resulting principal component score does not contain enough information to perform classification task. The additional information on data classification can be seen on the fourth component. Whereas the second and the third components explain data variation due to the introduction of correlation structure. The second loading relates to the correlation between variables of the first block, while the third loading corresponds with the correlation between variables of the second and the third block. As a result of this, one will obtain Classification Trees with lowest error rate by fitting the trees using the first fourth principal component scores and so forth.

Based on the above explanation for the resulting principal components of generated datasets with the simulation settings of large mean difference and unit standard deviation with $\rho = 0$ and 0.9, it can be inferred that, in the absence of correlation between variables, it is only mean difference as the source of data variation that influences the resulting principal component scores as previously mentioned. Meanwhile, in the presence of high correlated blocks of variables, one can see how the effect of data variation due to these correlated blocks weakens the effect of data variation due to large mean difference between.

Furthermore, Figure 5.45 shows the plots for loadings of principal components obtained using simulated datasets generated using the simulation settings of small mean difference, three standard deviation and $\rho = 0.0$. We present this plot to explain the results that, for this simulation setting, regardless the number of components used to fit Classification Trees, we always attain the trees with the same accuracy level. In this case, we get the ones with high error rate.

It is apparent from Figure 5.45 that among the six loadings presented, it is only loading of the first component that contained data variation due to between

class information. However, the effect of such this data variation on the first component is defeated by the effect of the overall variance. In addition, the other five loadings only contain random information.

Moreover, Figure 5.46 shows the plots for loadings of principal components obtained using simulated datasets generated using the simulation settings of small mean difference, three standard deviation and $\rho = 0.9$. We display this plot to explain the results that, for this simulation setting, it is the fourth principal component is the one which is responsible for classification. The first three components correspond with the correlation structures introduced into these simulated datasets as obtained for the simulation settings of small mean difference, unit standard deviation and $\rho = 0$. However, for the case of three standard deviation, involving the fourth component for fitting Classification Trees does not gives the trees whose low error rate as those for the case of unit standard deviation.

Therefore, it can be inferred that, for large standard deviation, in the absence of correlation between variables, the effect of variance on the resulting principal components overcomes the effect of mean difference. On the other hand, in the presence of blocks correlated variables, large standard deviation strengthens the effect of correlation and weakens the effect of mean difference on the resulting principal components.

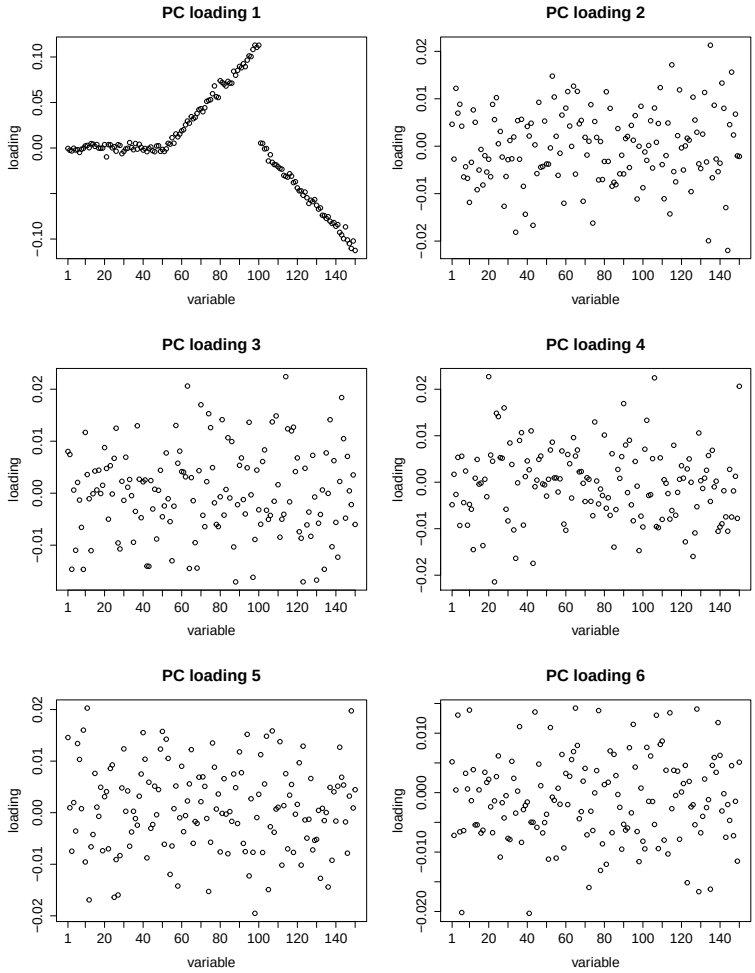


Figure 5.41: Plots of principal component loadings of simulated dataset with unit standard deviation, small mean difference and $\rho = 0.0$. All values presented in each panel of this figure are the average from 100 estimations obtained from 100 generated datasets generated using different seed settings.

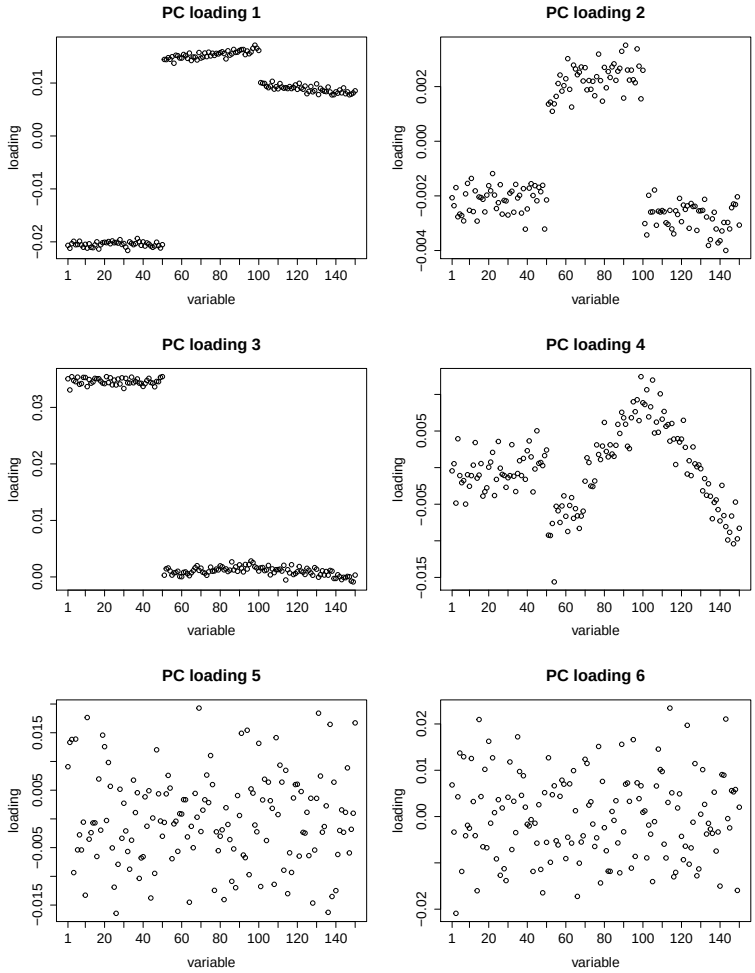


Figure 5.42: Plots of principal component loadings of simulated dataset with unit standard deviation, small mean difference and $\rho = 0.9$. All values presented in each panel of this figure are the average from 100 estimations obtained from 100 generated datasets generated using different seed settings.

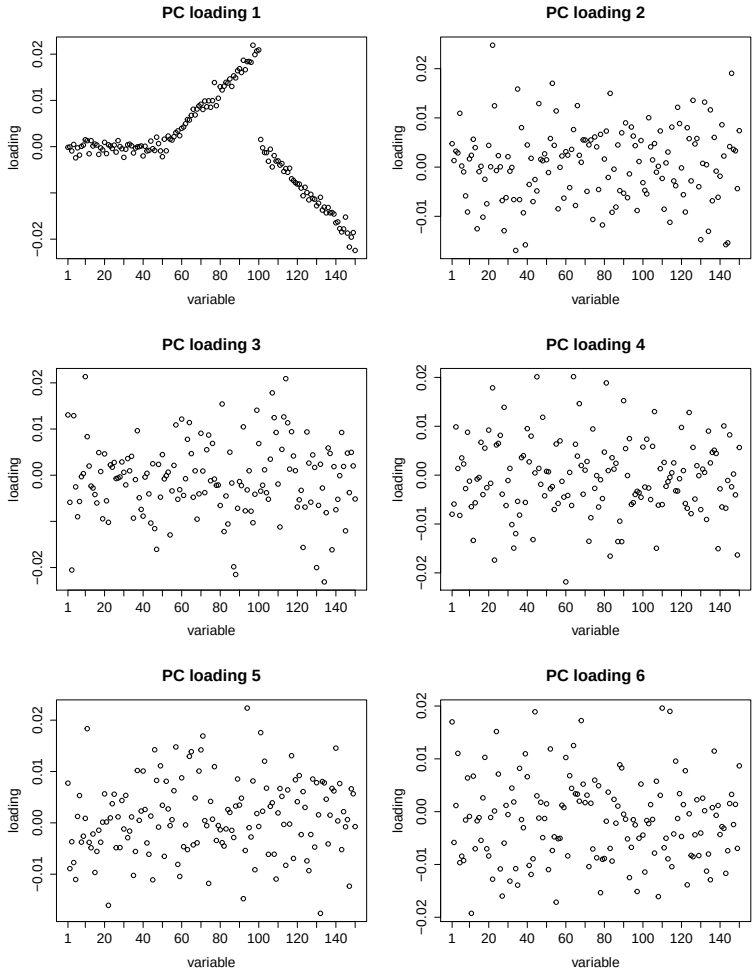


Figure 5.43: Plots of principal component loadings of simulated dataset with unit standard deviation, large mean difference and $\rho = 0.0$. All values presented in each panel of this figure are the average from 100 estimations obtained from 100 generated datasets generated using different seed settings.

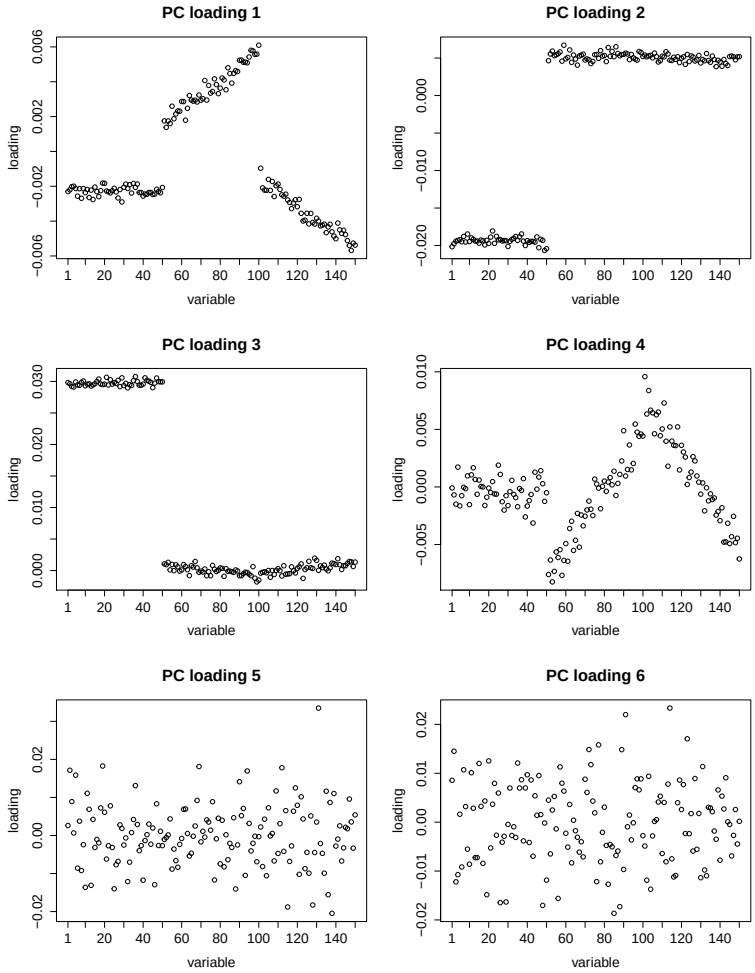


Figure 5.44: Plots of principal component loadings of simulated dataset with unit standard deviation, large mean difference and $\rho = 0.9$. All values presented in each panel of this figure are the average from 100 estimations obtained from 100 generated datasets generated using different seed settings.

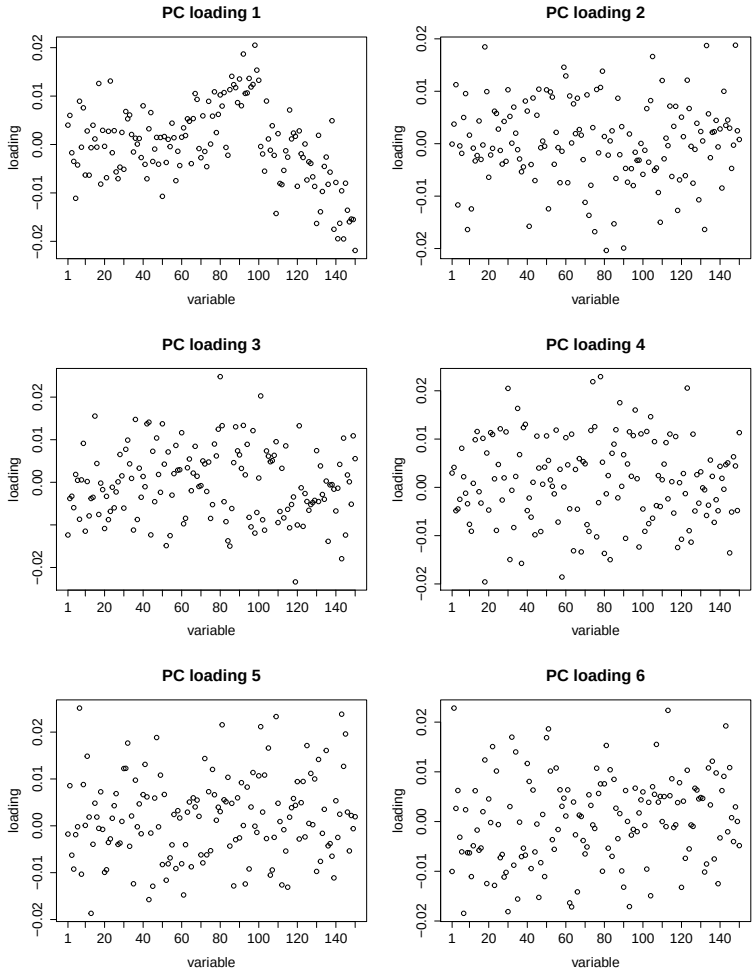


Figure 5.45: Plots of principal component loadings of simulated dataset with three standard deviation, small mean difference and $\rho = 0.0$. All values presented in each panel of this figure are the average from 100 estimations obtained from 100 generated datasets generated using different seed settings.

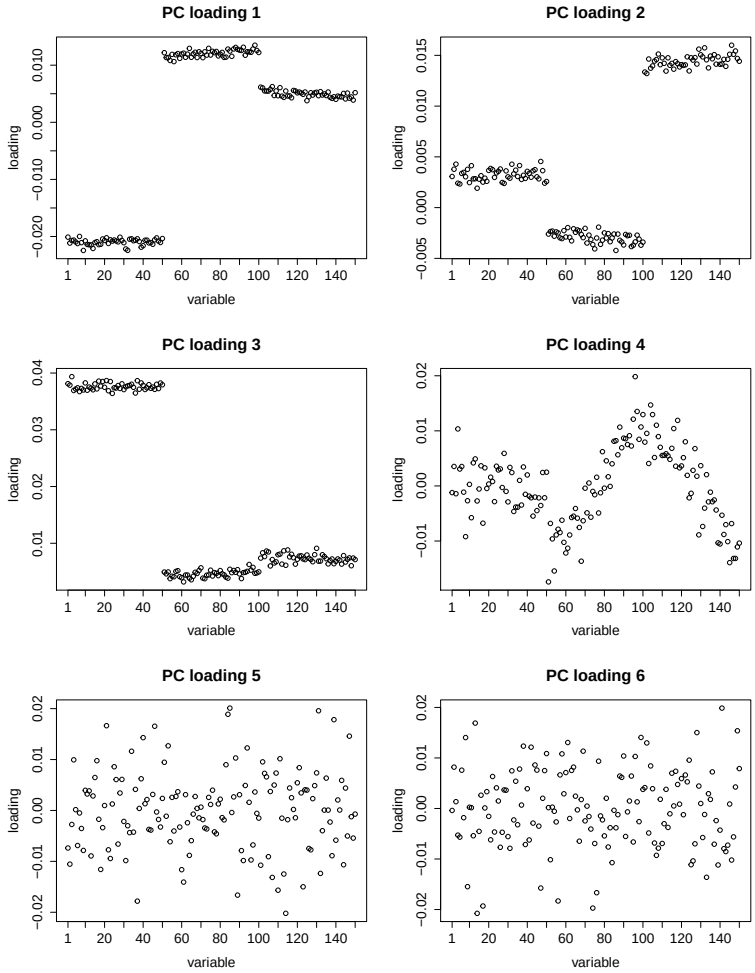


Figure 5.46: Plots of principal component loadings of simulated dataset with three standard deviation, small mean difference and $\rho = 0.9$. All values presented in each panel of this figure are the average from 100 estimations obtained from 100 generated datasets generated using different seed settings.

5.8 Conclusion

Applying PCA as a dimension reduction method before fitting Classification Trees is not an appropriate practice. For datasets which have much more variables than observations, PCA tends to impose few number of principal component scores to contain huge proportion of data variation. This can be seen from the spiky eigenvalues. As a result, it is more challenging to seek for the first few principal component scores which are the relevant predictors.

Moreover, there is also no guarantee that the discriminative information will be contained in the first few principal component scores. The first principal component score is the component that explains the largest proportion of data variation, but it is not necessarily being the component that describes the largest proportion of data variation due to between-group data variation. In the presence of high within-group variability, the effect of the between-group data variation on the principal component scores will be masked. Applying PCA as data dimension reduction before fitting Classification Trees only will work well if between-group variation dominates within- group variation. Although, applying a simple projection method such as the scalar projection of mean difference or median difference gives better results than applying PCA.

We end up with two different results while applying PCA prior to fitting Classification Trees on real and hypothetical datasets. For the former, we find that PCA cannot improve the performance of the resulting Classification Trees. PCA projected datasets yield less accurate trees. Meanwhile, for hypothetical datasets, we find that PCA is able to improve the performance of the trees. PCA projected datasets give more accurate trees. However, our simulation study shows that the first principal component score is not the one that contains the required information to carry out data classification task. The above results can be explained as follows.

We end up with more accurate trees while applying PCA on hypothetical datasets since the generated datasets do not have as complex structure as the real datasets. We take the presence of blocks of correlated variables, mean difference and variance while generating the hypothetical datasets into account. However,

the settings applied are not extreme enough to produce datasets as complex structure as the real datasets.

Reducing the data complexity by standardising and applying logarithmic does not give advantage as well. Both standardising and transforming the datasets before applying PCA do not improve the performance of the resulting principal component scores while fitting the Classification Trees.

Chapter 6

Classification Trees of Independent Component Scores

6.1 Introduction

In this chapter we discuss the application of Independent Component Analysis (ICA) prior to Classification Tree construction. We apply ICA as a method for finding the quantities hidden in our datasets and make use these latent structures to construct Classification Trees.

For computational purpose, we apply FastICA of Hyvärinen & Oja (1997) as an algorithm to estimate the independent components. This algorithm is also implemented in fastICA package of Marchini *et al.* (2019). However, we slightly modify the existing package as we deal with the datasets with $p \gg n$.

6.2 Population Independent Components

In this section we discuss the Independent Component model at the population level. We present the definition of the Independent Component model for a random vector $\mathbf{X}$. We base our discussion in this section on Ollila (2009), Nordhausen *et al.* (2011), Miettinen *et al.* (2017) and Nordhausen & Oja (2018).

6.2.1 Population Independent Component Model

Definition 6.1. *Let* $\mathbf{X} = (X_1, \ldots, X_p)^T$ *be a* p*-dimensional random vector with* $E(\mathbf{X}) = \boldsymbol{\mu}$ *and* $\text{Cov}(\mathbf{X}) = \Sigma$ *follows the independent component model where the components of* $\mathbf{X}$ *are linear mixtures of of* p *mutually independent unobserved random vectors* $\mathbf{Z} = (Z_1, \ldots, Z_p)^T$*. We define the independent component model for observed random vector* $\mathbf{X}$ *as*

$$\mathbf{X} = \Omega\mathbf{Z} \tag{6.2.1}$$

where Ω *is a full-rank* $p \times p$ *mixing matrix.*

Using the data matrix that consists of n measurements on p-dimensional random vector $\mathbf{X}$, we estimate the unmixing matrix Θ such that $\Theta\mathbf{X}$ produces Independent Components.

The p-dimensional random vector of independent components $\mathbf{Z}$ in Equation 6.2.1 should satisfy the following assumptions:

1. the component $Z_1, \ldots, Z_p$ of $\mathbf{Z}$ are statistically independent,
2. at most one of the components $Z_1, \ldots, Z_p$ of $\mathbf{Z}$ is a Gaussian component and
3. $E(Z_i) = 0$ and $\text{Var}(Z_i) = 1$, for $i = 1, 2. \ldots, p$.

The independence and Gaussianity of components of $\mathbf{Z}$ are sufficient for the consistency of the estimates of these components. Hyvärinen *et al.* (2001) propose that in the presence of more than one Gaussian component, these Gaussian components cannot be separated from each other. Meanwhile, the assumption of $E(\text{Z}_\text{i}) = 0$ and $\text{Var}(\text{Z}_\text{i}) = 1$, for $i = 1, 2. \ldots, p$, are required to fix the scale of their estimates.

There are several procedures can be used to solve the Independent Component model in Equation 6.2.1. One of the most popular ones is FastICA algorithm of Hyvärinen & Oja (1997). This algorithm consists of two computational steps:

1. whiten the random vector

2. find an orthogonal matrix that rotates the whitened random vector onto independent components.

We whiten the random vector $\mathbf{X}$ in order to obtain a linear transformed random vector which has identity matrix as its population covariance matrix. For the random vector $\mathbf{X}$, we defined its whitened random vector $\mathbf{X}^{wt}$ as

$$\mathbf{X}^{wt} = (\mathbf{X} - E(\mathbf{X}))\text{Cov}(\mathbf{X})^{-\frac{1}{2}} = (\mathbf{X} - \boldsymbol{\mu})\Sigma^{-\frac{1}{2}}. \tag{6.2.2}$$

Having obtained the whitened random vector $\mathbf{X}^{wt}$, we solve the Independent Component model by estimating an orthogonal $p \times p$ matrix $\text{U} = (\mathbf{U}_1, \ldots, \mathbf{U}_p)^T$ such that $\text{U}\mathbf{X}^{wt}$ produces independent components. Using FastICA algorithm, the orthogonal matrix U can be obtained by maximising the non-Gaussianity of the components of $\text{U}\mathbf{X}^{wt}$ which can be expressed as $\mathbf{U}_j^T\mathbf{X}^{wt}$, for $j = 1, \ldots, p$. Hyvärinen *et al.* (2001) has proposed entropy as a measure of non-Gaussianity. However, instead of using entropy for computational purpose, they apply neg-entropy which is the normalised version of entropy. We discuss this quantity in the following subsection.

6.2.2 Neg-Entropy

Let X be a random variable with density $p_X(x)$. The entropy of X is defined as

$$H(X) = -\int p_X(x) \log p_X(x) dx. \tag{6.2.3}$$

However, we can also use the normalized version of entropy called neg-Entropy. For a random variable X, its neg-Entropy is defined as

$$J(X) = H(X_{gauss}) - H(X) \tag{6.2.4}$$

where H denotes entropy and X_{gauss} represents a Gaussian random variable of the same covariance as X. To calculate J, we could use the following approximation.

$$J(X) = [E(G(X) - E(G(X^*_{gauss}))]^2 \tag{6.2.5}$$

where X^*_{gauss} denotes a Gaussian variable of zero mean and unit variance and G is either one of the two functions which are defined as follows.

$$G_1(X) = \frac{1}{a_1} \log\{\cosh(a_1 X)\} \tag{6.2.6}$$

$$G_2(X) = \exp\left(\frac{-X^2}{2}\right) \tag{6.2.7}$$

where a_1 is a constant and $1 \leq a_1 \leq 2$, but often taken to be 1.

6.2.3 Algorithm for finding one independent component

Returning now to solving the Independent Component model. We begin our discussion by deriving the algorithm for finding one independent component. This can be seen as a problem of finding a vector $\mathbf{U}$ such that $\mathbf{U}^T\mathbf{X}^{wt}$ achieves its maximum non-Gaussianity. This can be expressed mathematically as optimisation problem with objective function as follows

$$J(\mathbf{U}) = E\{G(\mathbf{U}^T\mathbf{X}^{wt})\}\} \tag{6.2.8}$$

subject to the constraint

$$E\{(\mathbf{U}^T\mathbf{X}^{wt})^2\} = \|\mathbf{U}\|^2 - 1 = 0 \tag{6.2.9}$$

where G is a function as defined in Equation (6.2.6) and $\|.\|$ represents vector norm. It should be noted here that we use the vector notation $\mathbf{U}$ to present the vector that will be used to produce an independent component. Meanwhile, we use the matrix notation U to present the matrix that will be used to attain p independent components.

According to Lagrange condition, the maxima of $J(\mathbf{U}) = E\{G(\mathbf{U}^T\mathbf{X}^{wt})\}\}$ under constraint $\|\mathbf{U}\|^2 = 1$ are obtained at points $\mathbf{U}$ satisfying

$$\nabla E\{G(\mathbf{U}^T\mathbf{X}^{wt})^2\} - \lambda\nabla\|\mathbf{U}\|^2 - 1 = 0 \tag{6.2.10}$$

where λ is Lagrange multiplier. After doing some algebraical works, we obtain

$$F(\mathbf{U}) = E\{\mathbf{X}^{wt} g(\mathbf{U}^T\mathbf{X}^{wt})\} - \lambda\mathbf{U} \tag{6.2.11}$$

where g is the derivative of G. Next, we get the Jacobi of $\nabla E\{G(\mathbf{U}^T\mathbf{X}^{wt})^2\}$ as

$$E\{\mathbf{X}^{wt}[\mathbf{X}^{wt}]^T g'(\mathbf{U}^T\mathbf{X}^{wt})\} - \lambda I \tag{6.2.12}$$

where g' is the derivative of g. It should be noted that

$$E\{\mathbf{X}^{wt}[\mathbf{X}^{wt}]^T g'(\mathbf{U}^T\mathbf{X}^{wt})\} = E\{\mathbf{X}^{wt}[\mathbf{X}^{wt}]^T\}E\{g'(\mathbf{U}^T\mathbf{X}^{wt})\} = E\{g(\mathbf{U}^T\mathbf{X}^{wt})\} \tag{6.2.13}$$

Thus, we obtain the approximation for Newton iteration as follows.

$$\mathbf{U}^{updated} \leftarrow \mathbf{U} - \frac{E\{\mathbf{X}^{wt}g(\mathbf{U}^T\mathbf{X}^{wt})\} - \lambda\mathbf{U}}{E\{g'(\mathbf{U}^T\mathbf{X}^{wt})\} - \lambda} \tag{6.2.14}$$

Multiplying both sides with $\lambda - E\{g'(\mathbf{U}^T\mathbf{X}^{wt})\}$ and doing some simplification, we get

$$\mathbf{U}^{updated} \leftarrow E\{\mathbf{X}^{wt}g(\mathbf{U}^T\mathbf{X}^{wt})\} - E\{g'(\mathbf{U}^T\mathbf{X}^{wt})\mathbf{U}\} \tag{6.2.15}$$

The result in Equation (6.2.15) is the basic fast fixed-point iteration which is also called as fastICA algorithm. We present the algorithm for this formula as in Algorithm (4).

Algorithm 4 FastICA algorithm for finding one independent component

input: a random vector $\mathbf{X}$, a constant ε

output: unmixing random vector $\mathbf{U}$

1: Do centring on random vector $\mathbf{X}$ to obtain $\mathbf{X}^{ct}$.
2: Whiten the centred random vector $\mathbf{X}^{ct}$ to get $\mathbf{X}^{wt}$.
3: Choose an initial random vector $\mathbf{U}$ of unit norm.
4: Let $\mathbf{U}^{updated} \leftarrow E\{\mathbf{X}^{wt}g(\mathbf{U}^T\mathbf{X}^{wt})\} - E\{g'(\mathbf{U}^T\mathbf{X}^{wt})\mathbf{U}\}$.
5: Let $\mathbf{U}^{updated} \leftarrow \frac{\mathbf{U}^{updated}}{\|\mathbf{U}^{updated}\|}$.
6: If $\|\mathbf{U}^{updated} - \mathbf{U}\| \geq \varepsilon$, go back to step 4.

6.2.4 Algorithm for finding multiple independent components

Moving on now to derive the algorithm to estimate several independent components. To estimate several independent components, we need to run the one-unit

fastICA algorithm several times with vectors $\mathbf{U}_1, \ldots, \mathbf{U}_p$. To prevent different vectors from converging to the same maximal point, we must orthogonalize the vectors $\mathbf{U}_1, \ldots, \mathbf{U}_p$ after each iteration.

There are two different ways of orthogonalisation of the vectors $\mathbf{U}_1, \ldots, \mathbf{U}_p$, deflationary and symmetric orthogonalisation. Using deflationary orthogonalisation, we estimate the independent component one by one. Meanwhile, using symmetric orthogonalisation, we estimate the independent components in parallel. In this case, we prefer to apply the first one rather than the second one as we need to recognise the independent components that are responsible to classification task.

Deflation orthogonalisation is based on a Gram-Schmidt orthogonalization. Having obtained q independent components for $q \leq p$, we run any one-unit algorithm for the $(q+1)-th$ independent component and after each iteration step we subtract from $\mathbf{U}_{q+1}$ the projection $(\mathbf{U}_{q+1}^T \mathbf{U}_j)\mathbf{U}_j$, for $j = 1, \ldots, q$ of the previous independent components and renormalise $\mathbf{U}_{q+1}$. Furthermore, having attained $\mathrm{U} = (\mathbf{U}_1, \ldots, \mathbf{U}_p)^T$, the unmixing matrix W is then $\mathrm{W} = \mathrm{U}\Sigma^{-\frac{1}{2}}$. We present the algorithm for this formula as in Algorithm (5). Both Algorithm (4) and (5) are adapted from Hyvärinen *et al.* (2001).

Algorithm 5 FastICA algorithm for finding several ICs with deflationary orthogonalization

input: a random vector $\mathbf{X}$, a constant ε

output: unmixing random vector $\mathbf{U}_j$, $j = 1, \ldots, q$

1: Choose m, number of Independent Components to estimate.
2: Set $q \leftarrow 1$
3: Initialise $\mathbf{U}_q$
4: Do an iteration of a one-unit algorithm on $\mathbf{U}_q$.
5: Do orthogonalisation $\mathbf{U}_q$=$\mathbf{U}_q$-$\sum_{j=1}^{q-1}(\mathbf{U}_q^T \mathbf{U}_j)\mathbf{U}_j$
6: Let $\mathbf{U}_q^{updated} \leftarrow \frac{\mathbf{U}_q^{updated}}{\|\mathbf{U}_q^{updated}\|}$.
7: If $\|\mathbf{U}_q^{updated} - \mathbf{U}_q\| \geq \varepsilon$, go back to step 4.
8: Set $q \leftarrow q + 1$. If q is not greater than m, go back to step 3.

6.3 Sample Independent Components

6.3.1 Sample Independent Component Model

In this section we discuss the Independent Component model at the sample level. We present the definition of the Independent Component model for the data matrix X of the size $n \times p$. We begin our discussion with the case of $n > p$.

Definition 6.2. *Let* X *be an* $n \times p$ *data matrix of mixtures with sample mean* $\overline{\mathbf{X}}$ *and sample variance* S *and* Z *be an* $n \times p$ *matrix of* p *unobserved sources, for* $p \leq p$. *We define the Independent Component model for the data matrix* X *as*

$$\underset{n\times p}{\mathrm{X}} = \underset{n\times p}{\mathrm{Z}} \ \underset{p\times p}{\mathrm{A}} \tag{6.3.1}$$

where A *is a full-rank* $p \times p$ *mixing matrix.*

Having defined the Independent Component model for data matrix X, we now turn to discuss the computational procedure to estimate matrix Z of independent components. We apply fastICA algorithm too for this purpose. Therefore, we also have two computational steps: whitening the data and finding an orthogonal matrix that rotates the whitened data matrix into sample independent components.

We begin with centring data matrix X. Let X^{ct} be the column-centred of data matrix X with sample mean $\overline{\mathbf{X}}$. We express X^{ct} as

$$\mathrm{X}^{ct} = \mathrm{X} - \mathbf{1}\overline{\mathbf{X}}^{T} \tag{6.3.2}$$

where $\mathbf{1}$ is a column vector with unit entries. Furthermore, we whiten this column-centred data matrix. There are three choices of whitening matrices.

1. $\mathrm{S}^{-\frac{1}{2}}$
2. $\widehat{\Gamma}\,\widehat{\Lambda}^{-\frac{1}{2}}$ and
3. $\sqrt{n-1}\,\widehat{\Upsilon}\,\widehat{D}^{-\frac{1}{2}}$

where S is sample covariance of data matrix X, $\widehat{\Gamma}$ matrix of eigenvectors of sample covariance matrix S, $\widehat{\Lambda}$ diagonal matrix of its eigenvalues, n is number of observations, $\widehat{\Upsilon}$ matrix of left singular vectors of centred data matrix X^{ct} and $\widehat{D}$ is diagonal matrix of its singular values.

Among these three whitening matrices, we prefer to use the third one as it does not involve sample covariance matrix. Instead, we only need to find the right singular matrix of the centred data matrix X^{ct} along with their singular values. Hence, we define the whitened data matrix as X^{wt} as

$$\underset{n\times p}{\mathrm{X}^{wt}} = \underset{n\times p}{\mathrm{X}^{ct}}\ \underset{p\times p}{\mathrm{K}} \tag{6.3.3}$$

where

$$\underset{p\times p}{\mathrm{K}} = \sqrt{n-1}\ \underset{p\times p}{\widehat{\Upsilon}}\ \underset{p\times p}{\widehat{D}^{-\frac{1}{2}}}. \tag{6.3.4}$$

We get both matrices $\widehat{\Upsilon}$ and $\widehat{D}^{-\frac{1}{2}}$ as $p \times p$ matrices in Equation (6.3.4) as we assume that $n > p$. However, in the computational point of view, Marchini *et al.* (2019) does not only use this whitening step for whitening the centred data matrix only, but also for setting the number of components to estimate in the following computational step.

Let q be a number of independent components to be estimated, for $q = 1, \ldots, p$. In his fastICA package, Marchini *et al.* (2019) recommend one to use the first q columns of the whitening matrix K in Equation (6.3.4) to build an $p \times q$ whitening matrix K. This $p \times q$ whitening matrix is then used to construct an $n \times q$ whitened matrix X^{wt}.

Furthermore, we estimate unmixing matrix W using the procedure presented in Algorithm (5) such that

$$\underset{n\times q}{\mathrm{X}^{wt}}\ \underset{q\times q}{\mathrm{W}} = \underset{n\times q}{\mathrm{Z}} \tag{6.3.5}$$

for $q = 1, \ldots, p$.

Equation (6.3.5) indicates that the resulting unmixing matrix is attained from the maximisation on non-Gaussianity which involves an $n \times q$ whitened data matrix. Hence, setting different values of the number of independent components q leads to different unmixing matrix W. This in turn produces different set of the estimated independent components Z as well.

Turning now to the case where we have much more variables than observations, $p \ggg n$. In this case, we no longer have data matrix X with rank $(X) = p$. Instead, we get rank $(X) = n - 1 \lll p$. Hence, for this case, we attain the whitened data matrix X^{wt} as

$$\underset{n\times p}{\mathrm{X}^{wt}} = \underset{n\times p}{\mathrm{X}^{ct}} \underset{p\times q}{\mathrm{K}} \tag{6.3.6}$$

where

$$\underset{p\times q}{\mathrm{K}} = \sqrt{n-1}\, \underset{p\times q}{\widehat{\Upsilon}}\, \underset{q\times q}{\widehat{D}^{-\frac{1}{2}}} \tag{6.3.7}$$

for $q = 1, \ldots, n-1$. This result shows how whitening step in FastICA algorithm significantly reduces the data dimension in the case of datasets with much more variables than observations. Moreover, we carry out the estimation of the unmixing matrix W as we perform for the case of $p < n$.

6.3.2 Computation of Sample Independent Components

In this subsection we discuss the application of the Independent Component model in the computational point of view. For computational purpose, we implement fastICA package of Marchini *et al.* (2019) to estimate independent components.

Applying fastICA package, one should specify the number of components to be estimated. To the best of our knowledge, there is no criterion has been proposed to determine the appropriate number of components to be extracted. In our case, we apply the Independent Component model as an attempt to find the underlying structure within our datasets and make use of this structure to improve the performance of the constructed Classification Trees. Hence, we determine the number of components by taking performance of extracted independent components in data classification into account.

Moreover, there is no order in the estimated independent components. Hence, picking $q+1$ independent components does not guarantee that the resulting first q independent components are equal to the set of q independent components obtained from the estimation with the setting q number of components. However, the whitened data matrix has order. As applied in fastICA package of Marchini *et al.* (2019), setting number of components to be q, one should pick the first

q columns of data matrix and performs non-Gaussianity optimisation involving this set of vectors of whitened data matrix.

6.4 Classification Trees of Independent Components for Real Datasets

In this section we discuss the application of Independent Component model prior to Classification Tree construction. We estimate independent components of our datasets and use the resulting independent components to construct Classification Trees. With regard to this task, there are two major concerns should be considered:

1. how to determine the number of independent components to be extracted
2. how to explain the way the independent components extract the discriminative information contained in datasets.

To begin with, we present plots of maximum goodness of splits for the resulting independent components. We present the plots of this quantity to describe the ability of the independent components in performing data classification task. Moreover, we also show the relation between the ability of independent components to perform the classification task with their non-Gaussianity. We present the plots of p-values obtained from normality test on those Independent Components.

Finally, we present the prediction error rate for the Classification Trees constructed using independent components. We obtain this true error rate estimate by making use of 10-fold stratified cross validation.

6.4.1 Classification Trees of Independent Components for Smooth CNA Dataset

In this subsection we present the results obtained from applying the Independent Component model prior to Classification Trees construction for Smooth CNA dataset. We begin with presenting the results that show how picking different

number of independent components q leads to different set of extracted independent components. In this case, we display the plots of the average of maximum goodness of split for each independent component attained for different setting of q as we deal with data classification task. As fastICA involves randomisation step, we perform the estimation of independent components 100 times using different seed setting and calculate the maximum goodness of split for each independent component. We present these results in Figure 6.1.

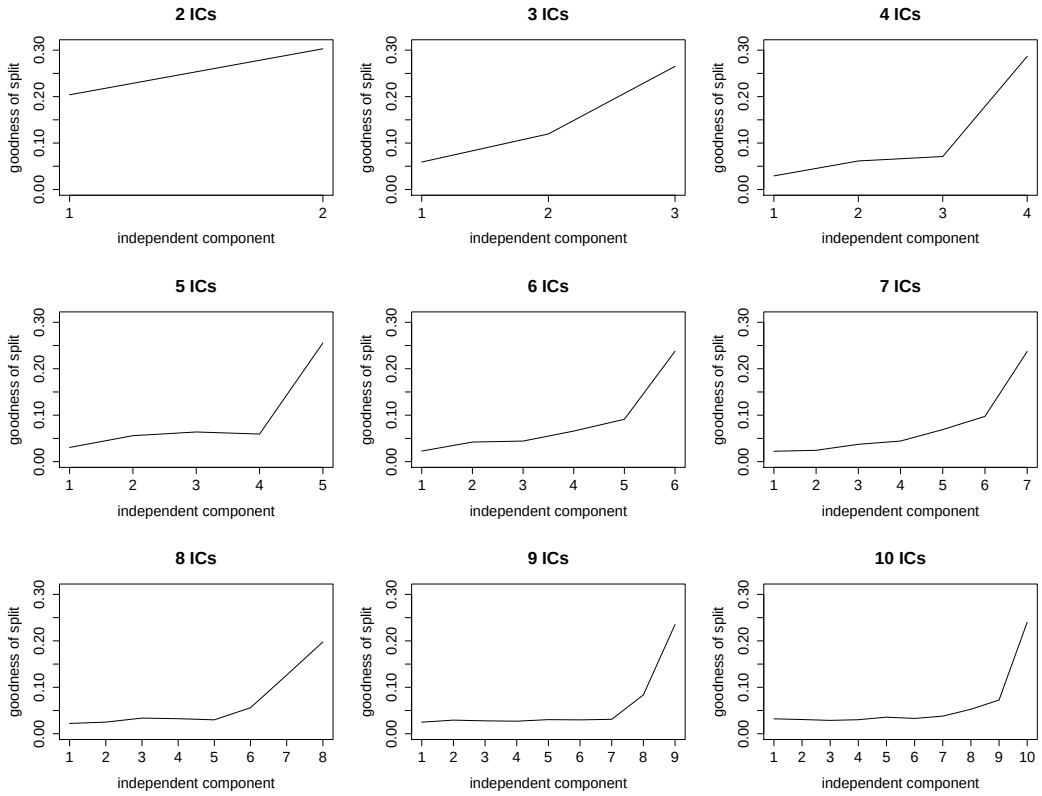


Figure 6.1: Plots of maximum goodness of split for independent components of Smooth CNA dataset obtained for number of components $q = 2, 3, \ldots, 10$. For these nine panels, the horizontal axis corresponds with each individual independent component.

Each panel in Figure 6.1 shows the maximum goodness of split of each independent components estimated by setting $q = 2, 3, \ldots, 10$. For example, the

top left panel displays the maximum goodness of split of both the first and the second independent components obtained by setting $q = 2$. We do not present the result for $q = 1$ since the independent component obtained by setting $q = 1$ is only the first column of whitened data matrix.

Interestingly, all panels indicate that the q-th independent component, for $q = 2, 3, \ldots, 10$, tends to be the one with the highest maximum goodness of split. The same pattern can also be seen for the other setting of number of components q as in in Figure 6.2. In this figure, it can be seen that for the setting of $q = 10$ and 20, the q-th independent component is the one with the highest goodness of split.

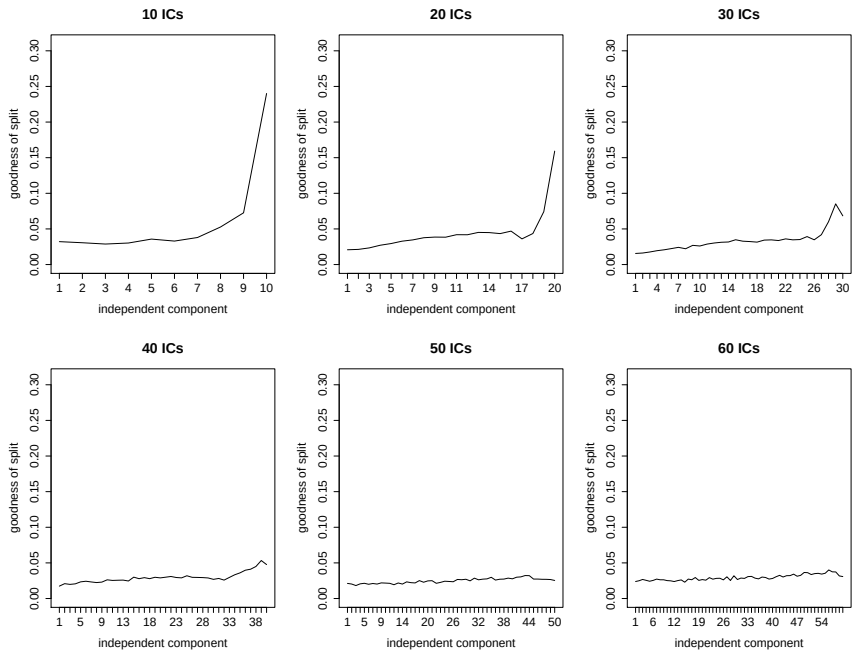


Figure 6.2: Plots of maximum goodness of split for independent components of Smooth CNA dataset obtained for number of components $q = 10, 20, \ldots, 60$. For these six panels, the horizontal axis corresponds with each individual independent component.

However, the highest goodness of split values of these two independent components are not as high as the goodness of split values of the q-th component

for the number of component setting $q = 2, 3, \ldots, 9$. This indicates the decrease in the performance of independent components in carrying classification task for large q settings.

In addition, Figure 6.2, also shows that for the number of component setting $q = 20$, the resulting maximum goodness of split for the tenth independent component is different from maximum goodness of split for the tenth independent component for $q = 10$. We obtain this result since there is no order in the extracted components. Also, setting the number of component $q = q_1$ and $q = q_2$ for $q_1 < q_2$, the set of q_1 independent components obtained from the estimation with $q = q_1$ is not the subset for the set of q_2 independent components obtained from the estimation with $q = q_2$.

Furthermore, Figure 6.2 shows that, for the setting of $q = 30$, 40, 50 and 60, the q-th independent component is not the one with the highest maximum goodness of split. Instead there are several components which have higher goodness of split values than these components do. This figure also presents that the highest maximum goodness of split values attained are also much lower than the maximum values for the independent components presented in Figure 6.1. This indicates that, for Smooth CNA dataset, the estimated independent components contain discriminative information only for small q setting. We are going to discuss this further in the end of this subsection.

Returning to the result that show the q-th independent component as the only component which has high maximum goodness of split, we might infer that among the set of estimated independent components, the last component is the one which will contribute to the data classification task. Hence, this component can be seen as the only relevant predictor. Whereas, the first $q - 1$ independent components can be seen as the irrelevant predictors.

As comparison for the resulting independent components, we present the plots of maximum goodness of split for each individual variable of whitened data matrix as in Figure 6.3. This figure shows that the second whitened variable is the one which contains discriminative information.

Moreover, we present the plots of the average of p-values attained from carrying out normality test on each individual estimated independent component for various setting of number of components q. For each setting of q, we perform the

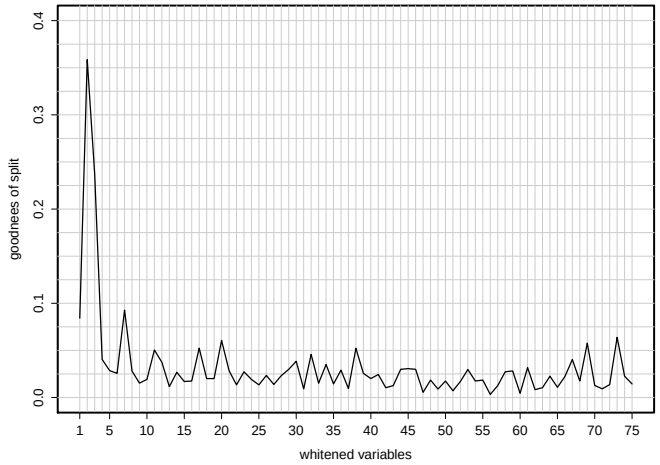


Figure 6.3: Plot of maximum goodness of split for each individual variable of whitened data matrix for Smooth CNA dataset. The horizontal axis corresponds with each individual independent component.

normality test using the function shapiro.test on each individual estimated component. In this case, we also estimate independent components 100 times using different seed settings and perform normality test on each resulting component. We perform this normality test since for the Independent Component model since it is assumed that at most one independent component follows Gaussian distribution.

We present the plots for these results in Figures 6.4 and 6.5. Panels of the first figure shows the result for $q = 2, 3, \ldots, 10$, while panels of the second one display the result for $q = 10, 20, \ldots, 60$. It is apparent from panels of Figures 6.4 and 6.5 that the q-th independent component is the component that follows Gaussian distribution from each set of estimated independent components.

Interestingly, this q-th independent component is also the one which has high maximum goodness of split. This indicates the presence of the relation between the contribution level of an estimated independent component onto data classification task and its normality. The independent component that contributes most on data classification is the only Gaussian component. Meanwhile, the other non-Gaussian independent components do not have such contribution for

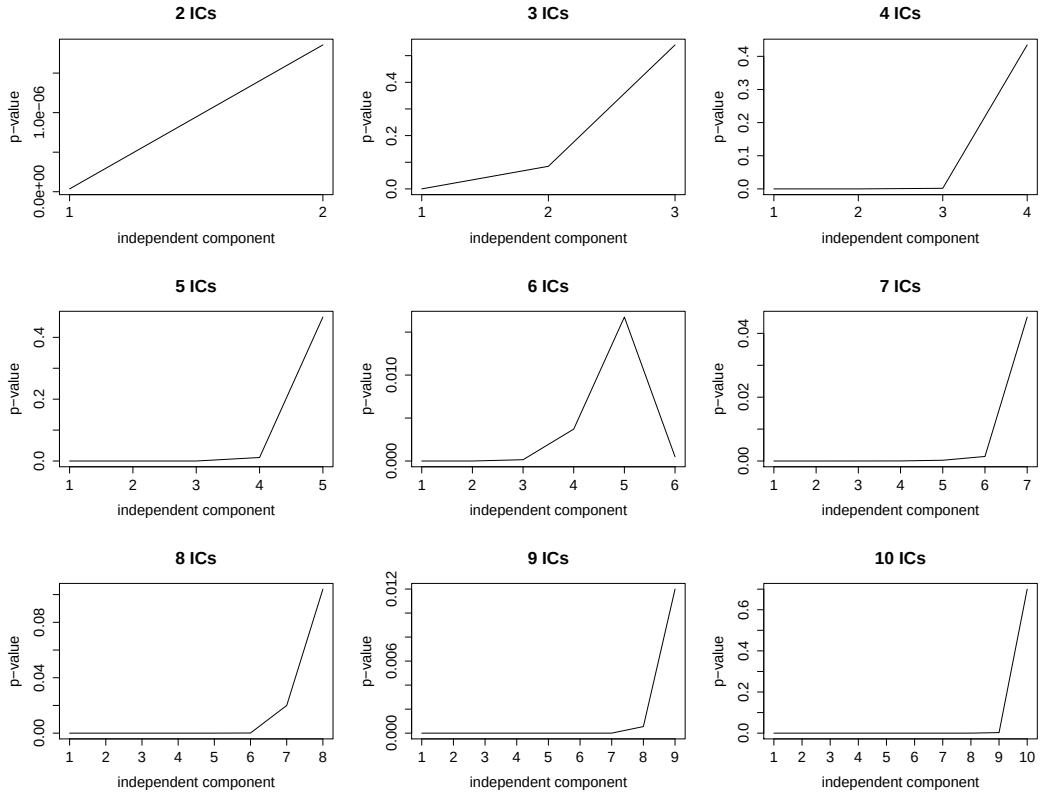


Figure 6.4: Plots for p-values obtained from normality test for independent components of Smooth CNA dataset obtained for number of components $q = 2, 3, \ldots, 10$. For these nine panels, the horizontal axis corresponds with each individual independent component.

the classification task.

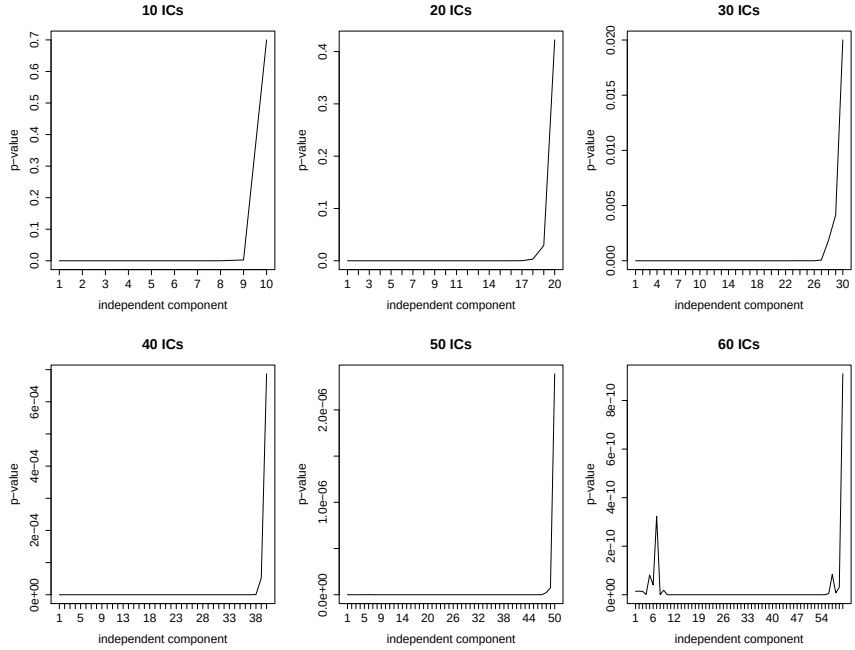
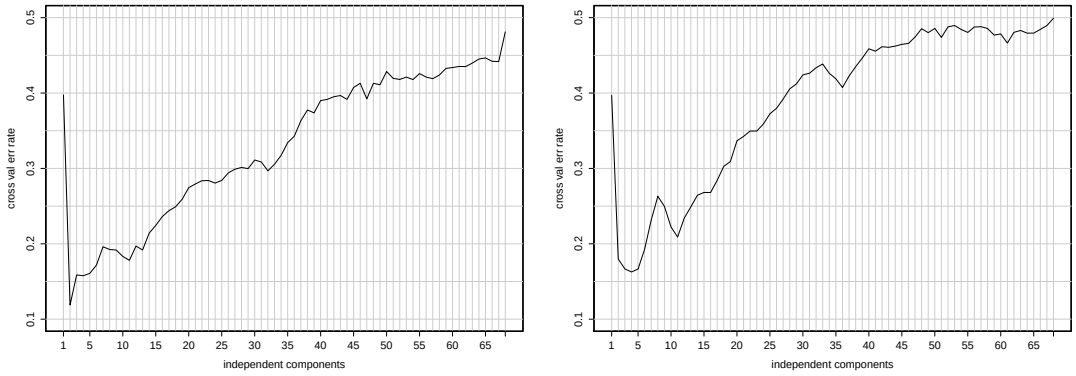


Figure 6.5: Plot for p-values obtained from normality test for independent components of Smooth CNA dataset obtained for number of components $q = 10, 20, \ldots, 60$. For these six panels, the horizontal axis corresponds with each individual independent component.

Moreover, we present the plots to show the error rates obtained from Classification Trees constructed using the set of first q independent components and the trees built using only the q-th independent component as in Figure 6.6. We get these true error rate estimates by making use of 100-repeated 10-fold stratified cross validation method. We compare the error rates obtained from the trees of either the set of first q independent components or only the q-th independent component as we found that the q-th independent component is the one which has largest contribution onto classification task.

Figure 6.6 shows that instead of building Classification Trees using the q-th independent component only, one should pick all the resulting independent components. Otherwise, one will end up with less accurate trees. This figure also



(a) (b)

Figure 6.6: Plots of prediction error for Classification Trees of independent components of Smooth CNA dataset. Panel (a) presents plots of prediction error for Classification Trees of the set of the first q independent components. Hence, the horizontal axis of the plot in this panel corresponds with the set of the first q-th independent components. Panel (b) shows plot of prediction error for Classification Trees of the q-th independent component for Smooth CNA dataset for $q = 1, 2, \ldots, 68$. Therefore, the horizontal axis corresponds with the q-th independent components.

indicates that for Smooth CNA dataset, we have to set the number of component $q = 2$ in order to obtain the optimal Classification Trees of independent components. However, compared to Classification Trees of either raw dataset or PCA projected dataset, Classification Trees of independent components have higher error rate. This can be seen in boxplots of Figure 6.7.

Figure 6.7 consists of two panels. Panel (a) shows boxplots of prediction error for Classification Trees of raw data, Classification Trees of the first two principal component scores and Classification Trees of the first two independent components for Smooth CNA dataset along with their paired difference data. Panel (b) presents boxplots for two paired difference data. The first paired difference data are obtained by subtracting the corresponding error rate estimates of Classification Trees fitted using raw dataset and those of trees built using independent components. Whereas the second paired difference data are attained by subtracting the corresponding error rate estimates of trees fitted using principal components and those of trees built using independent components.

To make inference on these results, we perform paired sample test for the comparison between the resulting error rates of Classification Trees fitted by the principal components and those built by the independent components. We present Table 6.1 to show the result obtained from performing Shapiro-Wilk test of normality and paired sample Wilcoxon test.

For the two paired sample Wilcoxon tests carried out, we define the null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero. We use this alternative hypothesis as we wish to obtain the lower error rates for the resulting Classification Trees of independent components than those for the Classification Trees of raw dataset and the trees of principal components.

Table 6.1 shows that the paired difference data for the error rates of the trees of raw data and the trees of independent components are not normally distributed. Meanwhile, the paired difference data for the error rates of the trees of principal components and the trees of independent components are normally distributed. Therefore, for the first paired difference data, we only perform the paired sample Wilcoxon test. On the other hand, for the second paired difference data, we carry out both the paired sample Wilcoxon test and the paired sample t test.

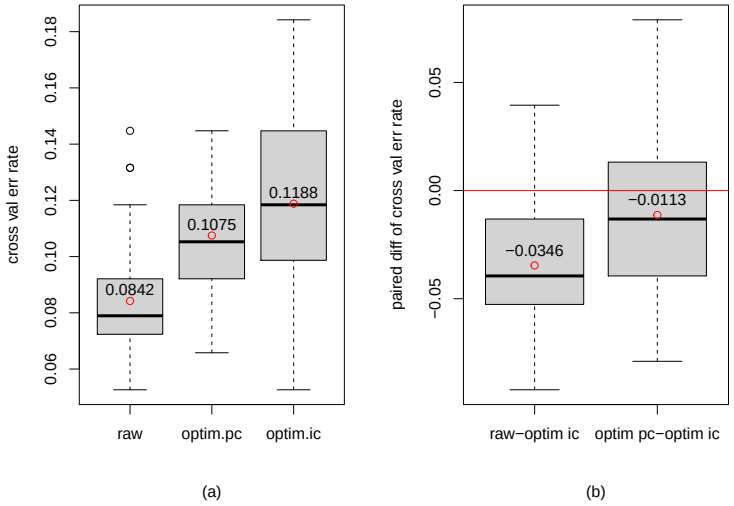


Figure 6.7: Boxplots of prediction error for Classification Trees of raw data, Classification Trees of the first two principal component scores and Classification Trees of the first two independent components for Smooth CNA dataset along with their paired difference data. Panel (a) shows boxplots for error rate estimates of Classification Trees fitted using raw dataset (the left boxplot), set of the first two principal components (the middle boxplot) and set of the first two independent components of transformed data with pre-processing 1 (the right boxplot). All boxplots represent 100 estimates of Classification Trees error rate obtained using 100-repeated 10-fold cross validation estimation. Panel (b) displays paired-difference data between these sets of the error rate estimates. The red horizontal line in this panel corresponds with paired-difference equals 0. Using paired sample Wilcoxon Test with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero, we obtain p-value equals 1 for paired comparison between error rate of Classification Trees fitted using raw data and independent components. Meanwhile, we attain p-value equals 0.99 for paired comparison between the trees built using principal components and independent components.

Table 6.1: P-values for Shapiro-Wilk test, Wilcoxon test, and paired t-test for comparing the error rates of Classification Trees of raw dataset, PCA projected, and ICA projected dataset

Comparison	Shapiro-Wilk test	Wilcoxon test	paired t-test
Raw data - ICA	0.00511	1	-
PCA - ICA	0.10667	0.99970	0.99978

Moreover, from Table 6.1, it can be inferred that we fail to reject H_0 that the median of the difference between the pairs equals zero. Therefore, for Smooth CNA dataset, it can be concluded that applying ICA prior to the Classification Trees construction does not improve the performance of the resulting trees. We attain the same conclusion from performing paired-sample t test for the second paired difference data.

Moreover, with regard to the computational time, we present the plot in Figure 6.8. Each dot in this plot represents the computational time required to fit Classification Trees using the set of the first j-th independent components of Smooth CNA dataset. This figure indicates that we need shorter time for fitting Classification Trees using the set of the first j-th independent components than using the raw dataset. It takes 16.69539 seconds to build Classification Trees using the raw data. However, it takes almost the same time duration to fit the Classification Trees using the set of the first j-th principal components.

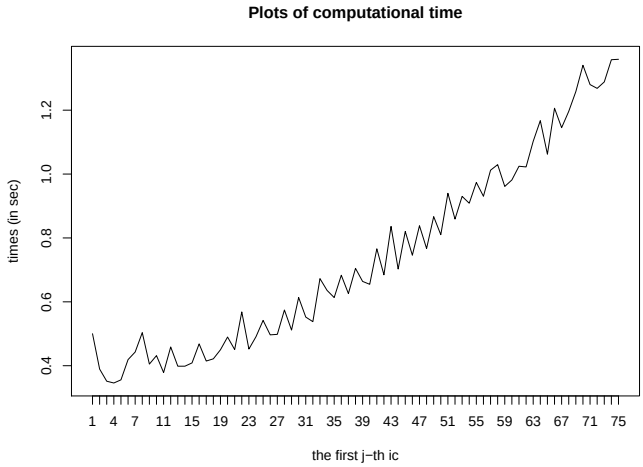


Figure 6.8: Plot of computational time for Classification Trees of independent components for Smooth CNA dataset. In this plot, the horizontal axis corresponds to the set of the first j-th independent components.

6.4.2 Classification Trees of Independent Components for DNACopy CNA Dataset

In this subsection we present the results obtained from applying the Independent Component model prior to Classification Trees construction for DNACopy CNA dataset. We organise our explanation on these results in the same way we did for Smooth CNA dataset. We begin our discussion for this subsection by showing the maximum goodness of split and the normality of the estimated independent components for DNACopy CNA dataset. We display the results for the maximum goodness of split of these independent components in Figures 6.9 and 6.10, whereas the results for their normality in Figures 6.12 and 6.13.

With regard to maximum goodness of split, Figure 6.9 and 6.10 show that the q-th independent component is always the one with the highest maximum goodness of split except for the setting for the number of component $q = 9$. The same results can also be found for the setting of number of components $q = 10$, 20 and 30.

However, as the value of the number of components increases, the maximum

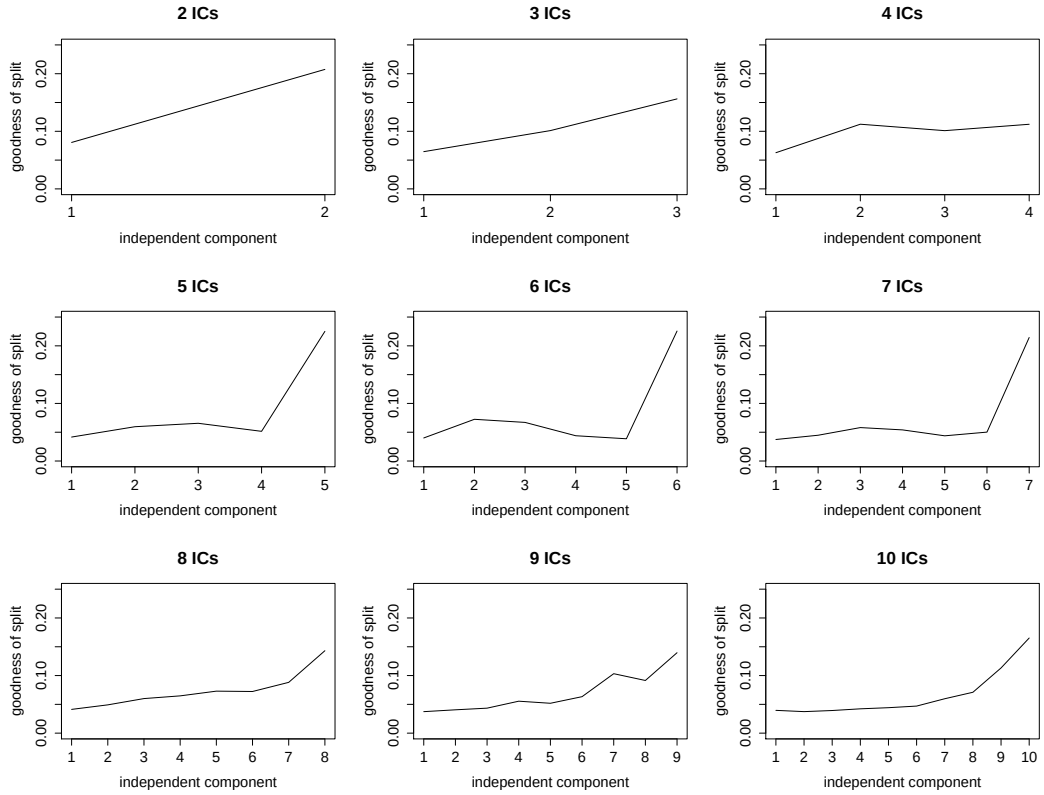


Figure 6.9: Plots of maximum goodness of split for independent components of DNACopy CNA dataset obtained for number of components $q = 2, 3, \ldots, 10$. For these nine panels, the horizontal axis corresponds with each individual independent component.

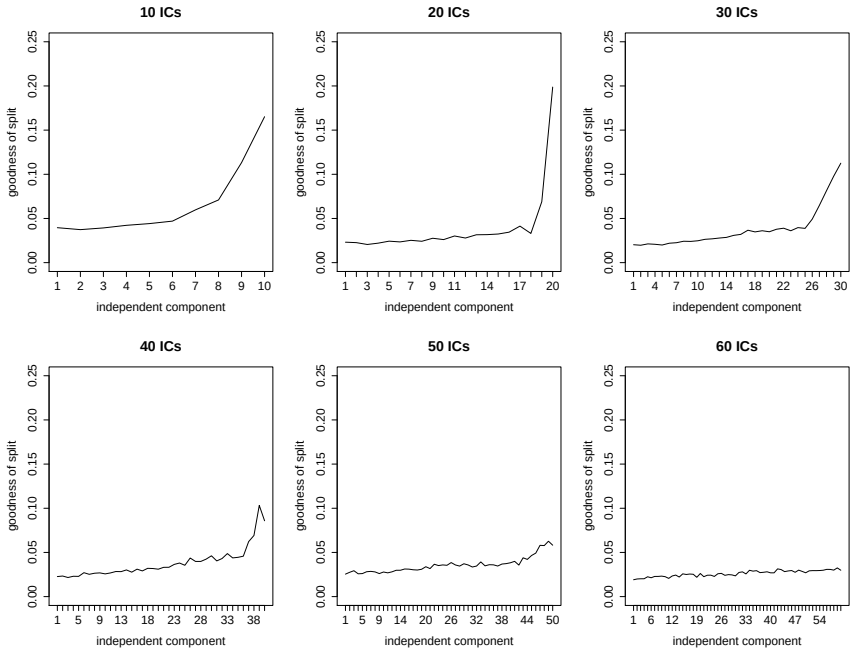


Figure 6.10: Plots of maximum goodness of split for independent components of DNACopy CNA dataset obtained for number of components $qp = 10, 20, \ldots, 60$. For these six panels, the horizontal axis corresponds with each individual independent component.

goodness of split values achieved decreases. This indicates the decrease in the performance of independent components in carrying classification task for large q settings as we found for Smooth CNA dataset. In addition, for DNACopy CNA dataset, we also might infer that among the set of estimated independent components, the last component is the one which will contribute to the data classification task. Hence, this component can be seen as the only relevant predictor. Whereas, the first $q-1$ independent components can be seen as the irrelevant predictors.

As comparison for the resulting independent components, we present the plots of maximum goodness of split for each individual variable of whitened data matrix as in Figure 6.11. This figure shows that the second whitened variable for DNACopy dataset is the one which contains discriminative information.

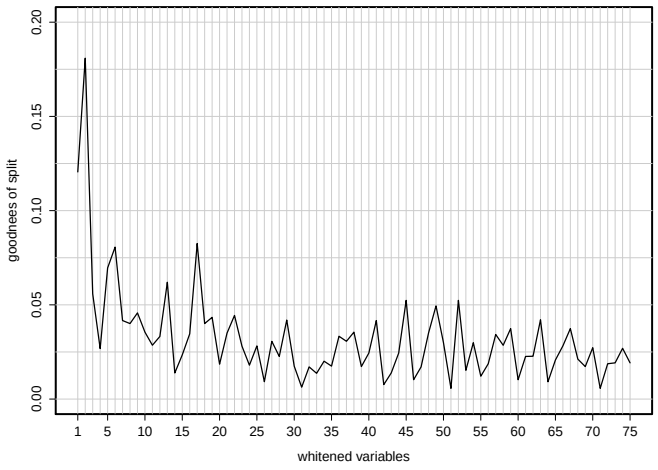


Figure 6.11: Plots of maximum goodness of split for each individual variable of whitened data matrix for DNACopy CNA dataset. The horizontal axis corresponds with each individual independent component.

Turning now to the normality of the estimated independent components for DNACopy CNA dataset. Figures 6.12 and 6.13 present the plots of the average of p-values attained from carrying out normality test on each individual estimated independent component for various setting of number of components q.

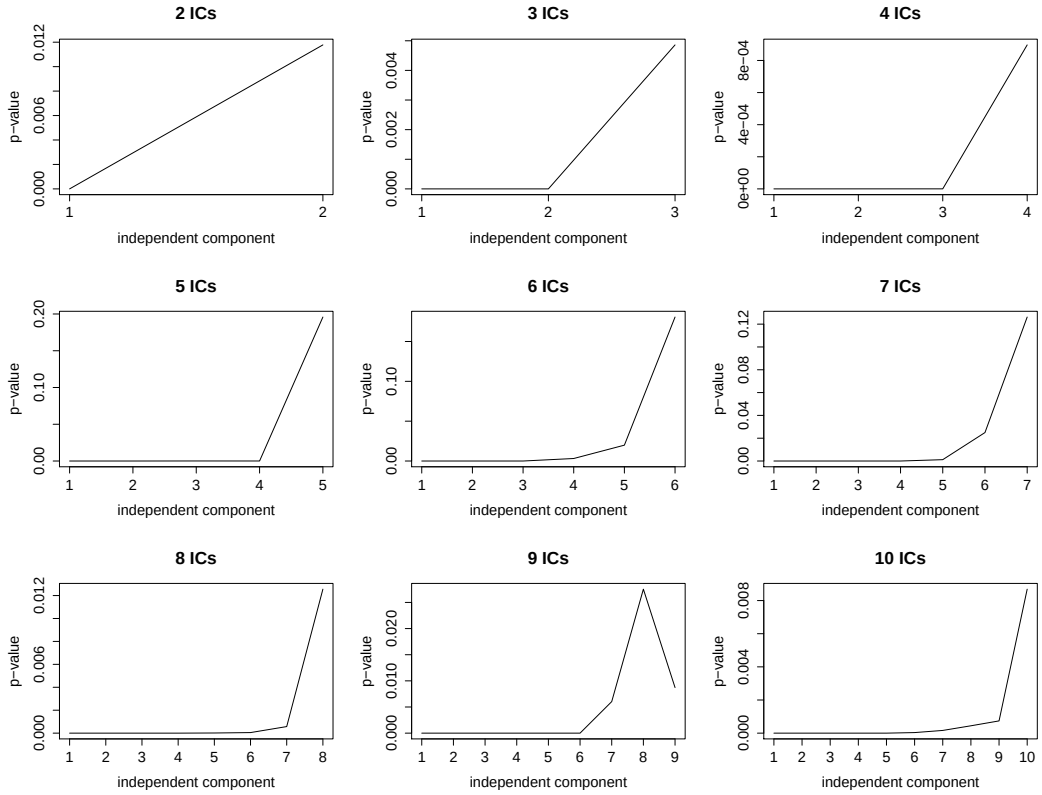


Figure 6.12: Plot for p-values obtained from normality test for independent components of DNACopy CNA dataset obtained for number of components $q = 2, 3, \ldots, 10$. For these nine panels, the horizontal axis corresponds with each individual independent component.

What stands out in Figures 6.12 and 6.13 is that, for the estimated independent components of DNACopy CNA dataset, the q-th independent component is the component that follows Gaussian distribution from each set of estimated independent components. Likewise for Smooth CNA dataset. Hence, for this dataset, we also find the interesting result that indicates the relation between the contribution level of an estimated independent component onto data classification task and its normality. We find that the independent component which contributes most on data classification is the only Gaussian component. Whereas the other non-Gaussian independent components do not have such contribution for the classification task.

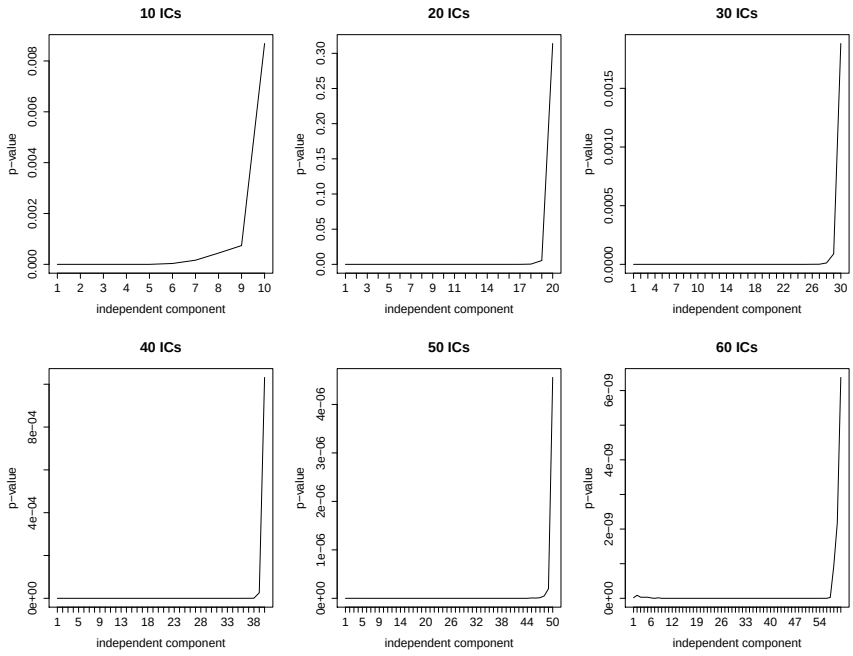
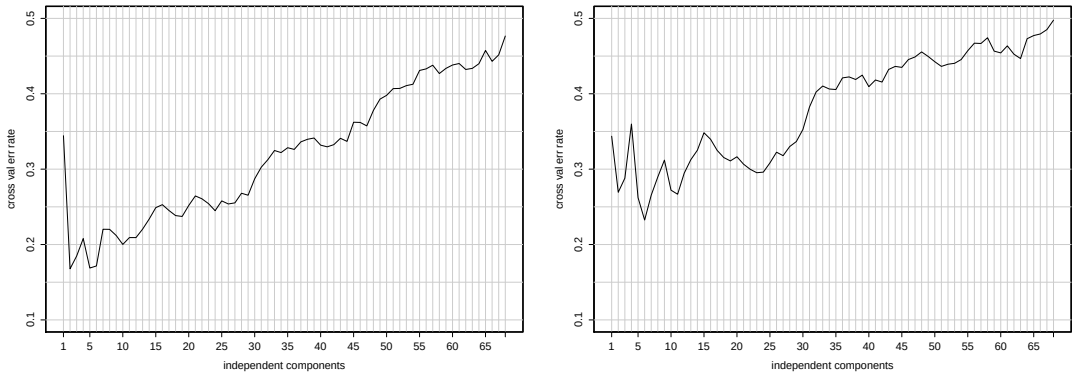


Figure 6.13: Plots for p-values obtained from normality test for independent components of DNACopy CNA dataset obtained for number of components $q = 10, 20, \ldots, 60$. For these six panels, the horizontal axis corresponds with each individual independent component.

Furthermore, we present the plots to show the error rates attained from Classification Trees built using the set of first q independent components and the trees

built using only the q-th independent component as in Figure 6.14 for DNACopy CNA dataset. We obtain these true error rate estimates by applying 100-repeated 10-fold stratified cross validation method. We compare the error rate obtained from the trees of either the set of first q independent components or only the q-th independent component as we found that the q-th independent component is the one which has largest contribution onto classification task.



(a) (b)

Figure 6.14: Plots of prediction error for Classification Trees of independent components of DNACopy CNA dataset. Panel (a) presents plots of prediction error for Classification Trees of the set of the first q independent components. Hence, the horizontal axis of the plot in this panel corresponds with the set of the first q-th independent components. Panel (b) shows plot of prediction error for Classification Trees of the q-th independent component for DNACopy CNA dataset for $q = 1, 2, \ldots, 68$. Therefore, the horizontal axis corresponds with the q-th independent components.

Figure 6.14 shows that instead of building Classification Trees using the q-th independent component only, one should pick all the resulting independent components. Otherwise, one will end up with less accurate trees. This figure also indicates that for DNACopy CNA dataset, we have to set the number of component $q = 2$ in order to obtain the optimal Classification Trees of independent

components. However, compared to Classification Trees of either raw dataset or PCA projected dataset, Classification Trees of independent components have higher error rate. This can be viewed in boxplots of Figure 6.15.

Figure 6.15 consists of two panels. Panel (a) shows boxplots of prediction error for Classification Trees of raw data, Classification Trees of the first two principal component scores and Classification Trees of the first two independent components for Smooth CNA dataset along with their paired difference data. Panel (b) presents boxplots for two paired difference data. The first paired difference data are obtained by subtracting the corresponding error rate estimates of Classification Trees fitted using raw dataset and those of trees built using independent components. Whereas the second paired difference data are attained by subtracting the corresponding error rate estimates of trees fitted using principal components and those of trees built using independent components.

To make inference on these results, we perform paired sample test for the comparison between the resulting error rates of Classification Trees fitted by the principal components and those built by the independent components. We present Table 6.2 to show the result obtained from performing Shapiro-Wilk test of normality and paired sample Wilcoxon test.

For the two paired sample Wilcoxon tests carried out, we define the null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero. We use this alternative hypothesis as we wish to obtain the lower error rates for the resulting Classification Trees of independent components than the error rates for the Classification Trees of raw dataset and the trees of principal components.

Table 6.2: P-values for Shapiro-Wilk test, Wilcoxon test, and paired t-test for comparing the error rates of Classification Trees of raw dataset, PCA projected, and ICA projected dataset of DNACopy CNA dataset

Comparison	Shapiro-Wilk test	Wilcoxon test	paired t-test
Raw data - ICA	0.05514	1	1
PCA - ICA	0.00005	0.99990	-

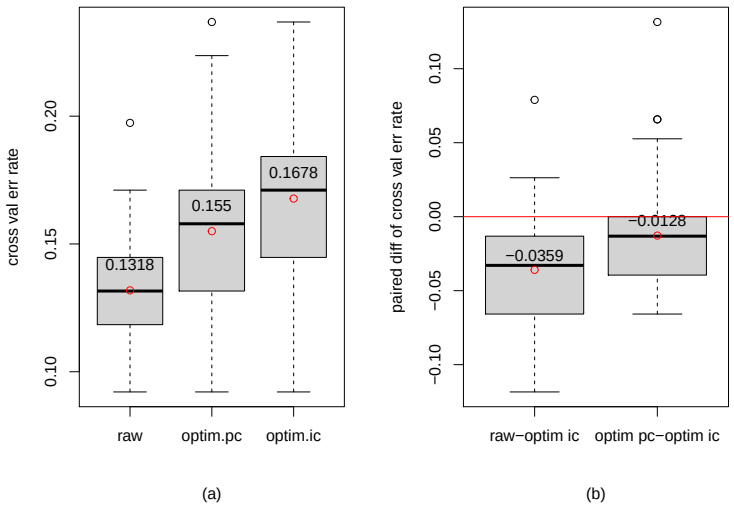


Figure 6.15: Boxplots of prediction error for Classification Trees of raw data, Classification Trees of the first two principal component scores and Classification Trees of the first two independent components for DNACopy CNA dataset along with their paired difference data. Panel (a) shows boxplots for error rate estimates of Classification Trees fitted using raw dataset (the left boxplot), set of the first two principal components (the middle boxplot) and set of the first two independent components of transformed data with pre-processing 1 (the right boxplot). All boxplots represent 100 estimates of Classification Trees error rate obtained using 100-repeated 10-fold cross validation estimation. Panel (b) displays paired-difference data between these sets of the error rate estimates. The red horizontal line in this panel corresponds with paired-difference equals 0. Using paired sample Wilcoxon Test with null hypothesis H_0: median of difference between the pairs equals zero along with one-sided alternative hypothesis H_1: median of difference between the pairs greater than zero, we obtain p-value equals 1 for paired comparison between error rate of Classification Trees fitted using raw data and independent components. Meanwhile, we attain p-value equals 0.99 for paired comparison between the trees built using principal components and independent components.

Table 6.2 shows that the paired difference data for the error rates of the trees of raw data and the trees of independent components are normally distributed. Meanwhile, the paired difference data for the error rates of the trees of principal components and the trees of independent components are not normally distributed. Therefore, for the first paired difference data, we carry out both the paired sample Wilcoxon test and the paired sample t test. On the other hand, for the second paired difference data, we only perform the paired sample Wilcoxon test

Moreover, from Table 6.2, it can be inferred that we fail to reject H_0 that the median of the difference between the pairs equals zero. Therefore, for DNACopy CNA dataset, it can be concluded that applying ICA prior to the Classification Trees construction does not improve the performance of the resulting trees. We attain the same conclusion from performing paired-sample t test for the first paired difference data.

Moreover, with regard to the computational time, we present the plot in Figure 6.16. Each dot in this plot represents the computational time required to fit Classification Trees using the set of the first j-th independent components of DNACopy CNA dataset. This figure indicates that we need shorter time for fitting Classification Trees using the set of the first j-th independent components than using the raw dataset. It takes 16.83186 seconds to build Classification Trees using the raw data. However, it takes almost the same time duration to fit the Classification Trees using the set of the first j-th principal components.

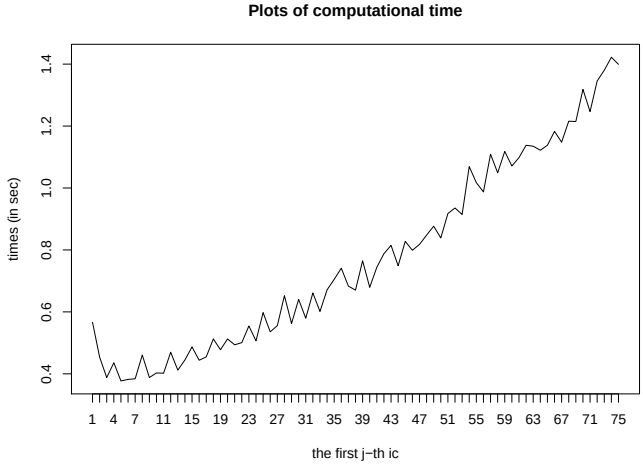


Figure 6.16: Plot of computational time for Classification Trees of independent components for DNACopy CNA dataset. In this plot, the horizontal axis corresponds to the set of the first j-th independent components.

6.5 Classification Trees of Independent Components for Simulated Dataset

In this section we discuss the results obtained from applying Classification Trees on independent components of simulated datasets. More detail simulation setting used in this section can be found in Section 2.4 of Chapter 2, partly in Section 4.5 of Chapter 4 and Section 5.7 of Chapter 5. As applied for the simulation study for Classification Trees of principal components, in this section we also present the results of the simulation by considering three quantities: correlation within-blocks of variables, mean difference between sub-population and standard deviation of each single variable.

Having generated datasets in the forms of data matrix of the size 100×150, we have independent component scores matrix of the size 100×99 as the simulated data matrix has 99 degree of freedom. However, instead of presenting the prediction error rates for Classification Trees fitted using the first jth independent components, for $j = 1, 2, \ldots, 99$, we only show 15 of them. We find that using the jth independent components, for $j = 16, 17, \ldots, 99$, we end up with Classification Trees with increasing error rates.

With regard to simulation procedure, for this simulation study, we generate the simulated datasets in the forms of data matrix of the size 100×150. Unlike the simulation involving principal components as in Section 5.7 of Chapter 5, for the simulation study involving independent components we are no longer able to obtain the estimates of independent components once for the whole independent components and then subset the set of the resulting independent components by picking only the j-th components to be studied. Instead, we have to perform this estimation for each number of components to be extracted.

For example, for fitting Classification Trees using five independent components, we should carry out the estimation by setting the number of components to become five. Meanwhile, for the trees of the four components, we should perform another calculation by setting the number of component to become four. We can not only select the first four components from the previous simulation.

Turning now to the results obtained from this simulation study. We present these results in the following 12 figures, from Figure 6.17 up to Figure 6.28. The

first six figures show the results for one standard deviation setting, while the remaining six figures display the results for three standard deviation. All of these figures consist of either six or four panels. Each panel contains 15 boxplots for 100 estimates of test set error rate of Classification Trees fitted using the first j independent components, for $j = 1, 2, \ldots, 15$.

For this simulation study, we also display the results obtained from the first j independent components, for $j = 1, 2, \ldots, 15$, and put the results from the remaining independent components aside. However, we present this result as we find that fitting Classification Trees involving the remaining independent components only increase the trees error rate.

In the light of prediction error rates, we obtain the same results from the simulation study for Classification Trees of either principal components or independent components. The increase in mean difference of generated datasets causes the increase in accuracy level of Classification Trees of independent components. On contrast, the increase in standard deviation leads to the decrease in accuracy level of Classification Trees of independent components.

We also find two general pattern for the results of this simulation by considering the change in the correlation setting. First, fitting Classification Trees using the first few set of independent components yields trees of same accuracy level, while involving the remaining components produces less accurate trees. Secondly, constructing Classification Trees using the the first three independent components leads to the trees of high test set error rate, while using the first fourth and some other components leads to those of lower prediction error rate. Whereas, involving the remaining components produces less accurate trees.

The two top panels of Figure 6.17 show the results of the first pattern. Meanwhile, the bottom two panels of this figure display the results of the second pattern. We obtain such this result for the same reason as what we obtain from simulation study for Classification Trees of principal component scores.

Let us consider the cases presented in both Figures 6.17 and 6.18. These figures show the boxplots for prediction error rate for Classification Trees of independent components of simulated datasets generated with small mean difference and unit standard deviation. The first figure displays the results for the corre-

lation setting of $\rho = 0, 0.1, \ldots, 0.5$ and the second presents the results for the correlation setting of $\rho = 0.6, 0.7, \ldots, 0.9$.

These two figure show that, for the correlation setting of $\rho = 0$ and 0.1, fitting Classification Trees using the fist few independent components gives the trees with the same accuracy level. Meanwhile, for the correlation setting of $\rho = 0.3, 0.7, \ldots, 0.9$, one should fit Classification Trees using the first four independent components to obtain the more accurate trees. Constructing Classification Tree using either one, two or three independent components, one ends up with less accurate tree.

Moreover, both Figure 6.23 and 6.24 indicate that, except for the correlation settings of 0.8 and 0.9, one finds that fitting Classification Trees using one up to 15 independent components gives the trees with the same accuracy level. These two figure present the results for small mean difference and three standard deviation.

Furthermore, both Figures 6.21 and 6.22 show that for all correlation settings except for $\rho = 0.8$ and 0.9, one always obtains the results that fitting Classification Trees using the first few independent components gives the trees with the same accuracy level. These two figures present the results for large mean difference and unit standard deviation.

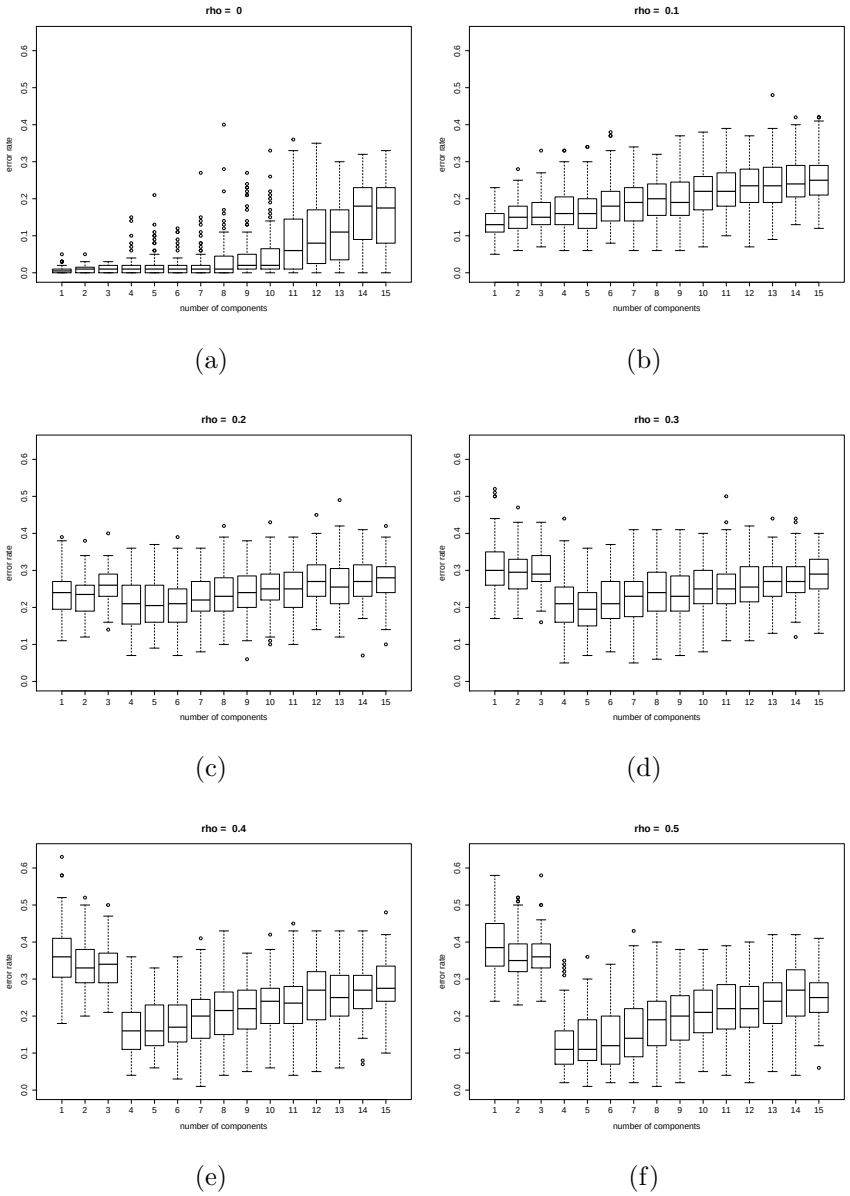


Figure 6.17: Boxplots of test set error rates of Classification Trees fitted using independent components of simulated datasets with small mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0, 0.1, \ldots, 0.5$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the first 1,2,...,15 independent components.

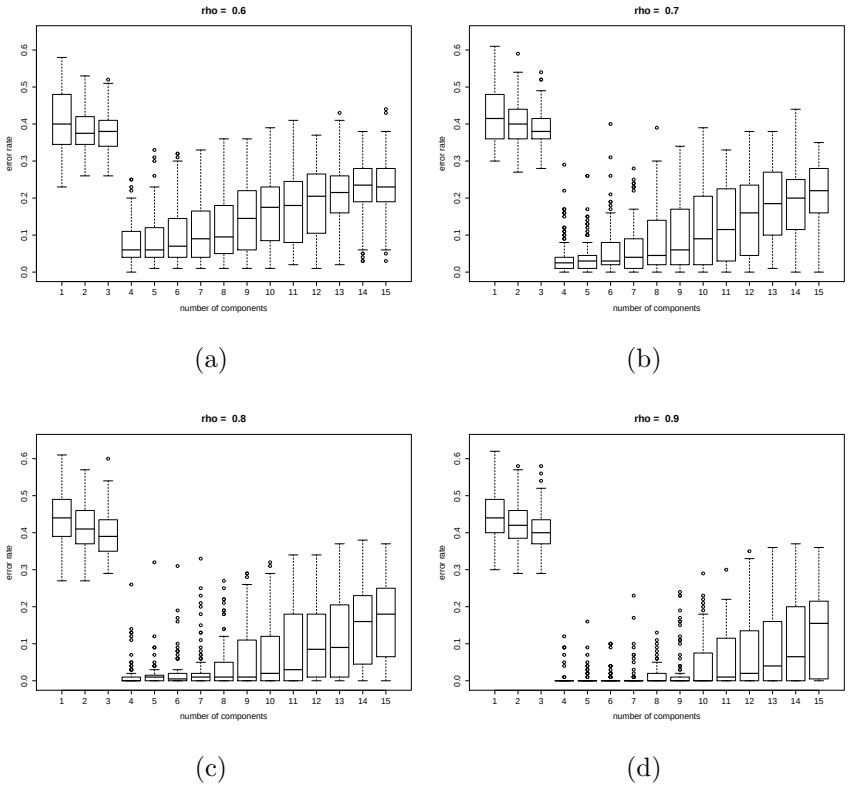
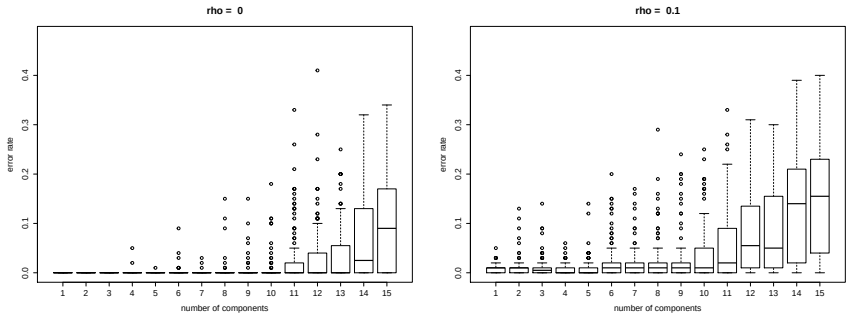
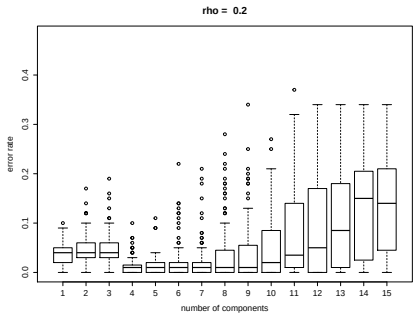


Figure 6.18: Boxplots of test set error rates of Classification Trees fitted using independent components of simulated datasets with small mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0.6, 0.7, \ldots, 0.9$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the first 1,2,...,15 independent components.

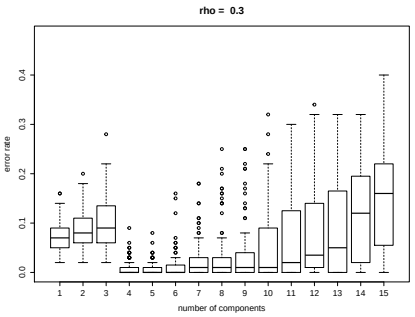


(a)

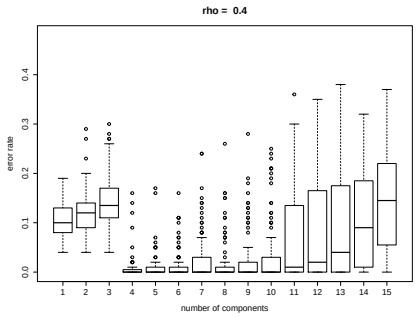
(b)



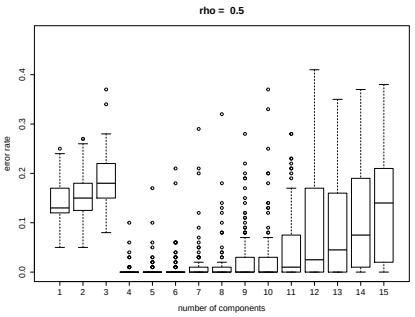
(c)



(d)



(e)



(f)

Figure 6.19: Boxplots of test set error rates of Classification Trees fitted using independent components of simulated datasets with medium mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0, 0.1, \ldots, 0.5$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the first 1,2,...,15 independent components.

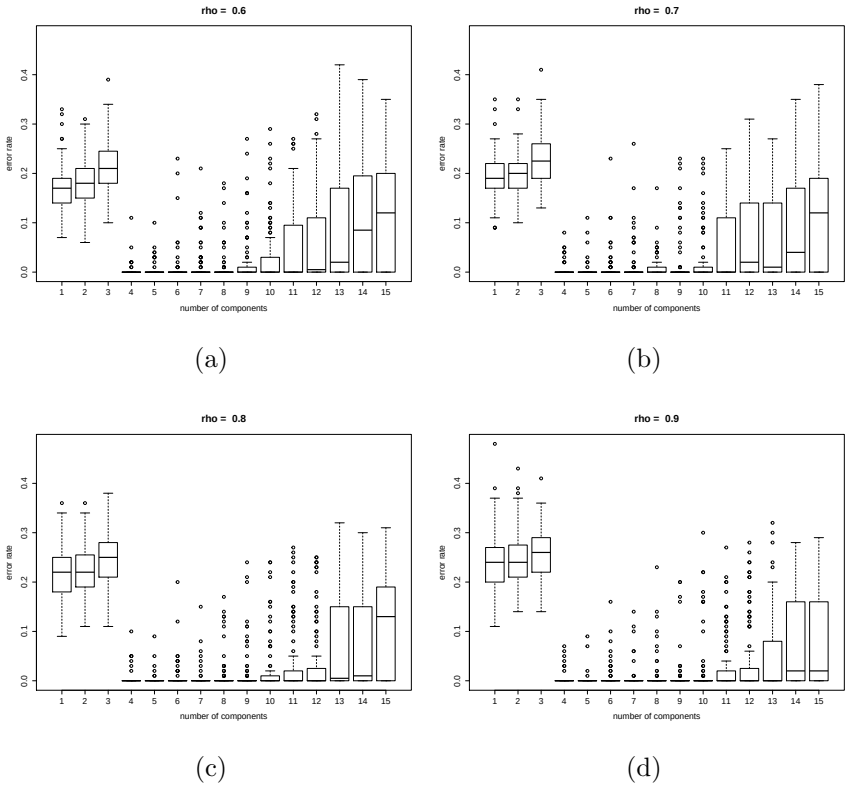
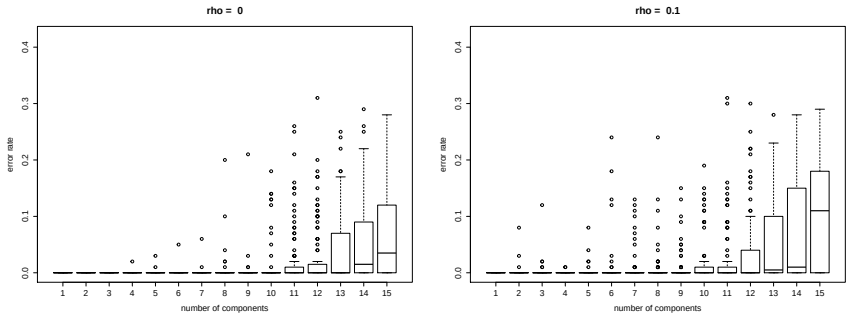
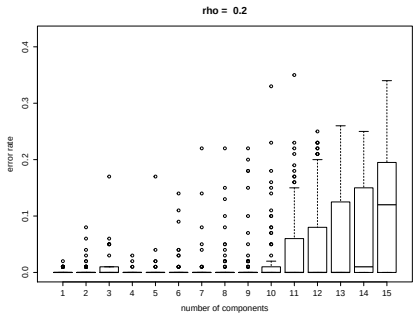


Figure 6.20: Boxplots of test set error rates of Classification Trees fitted using independent components of simulated datasets with medium mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0.6, 0.7, \ldots, 0.9$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the first 1,2,...,15 independent components.

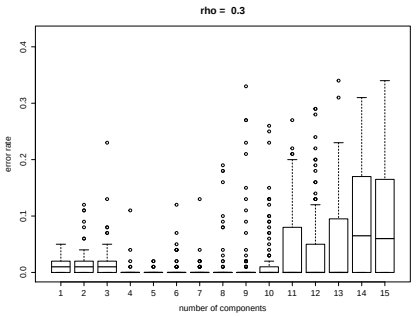


(a)

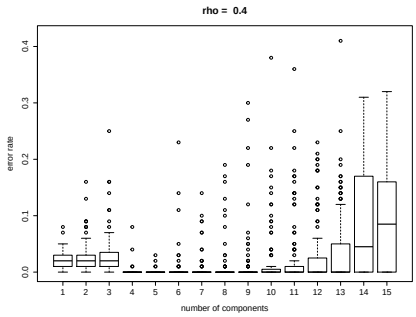
(b)



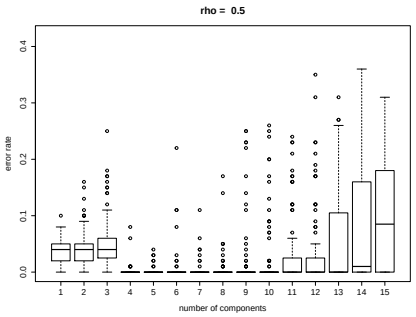
(c)



(d)

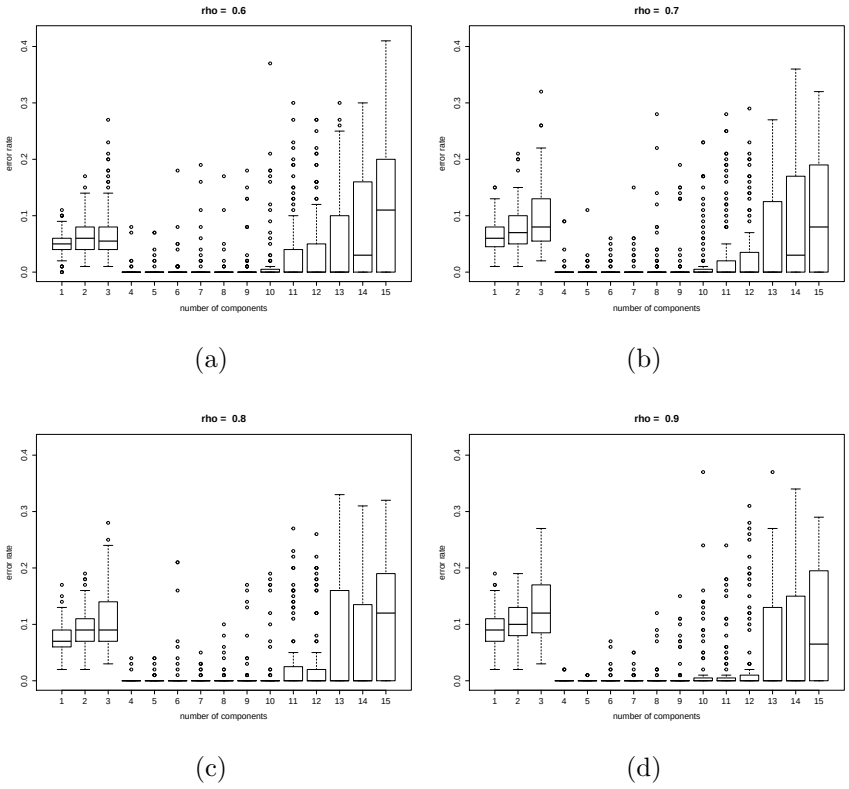


(e)



(f)

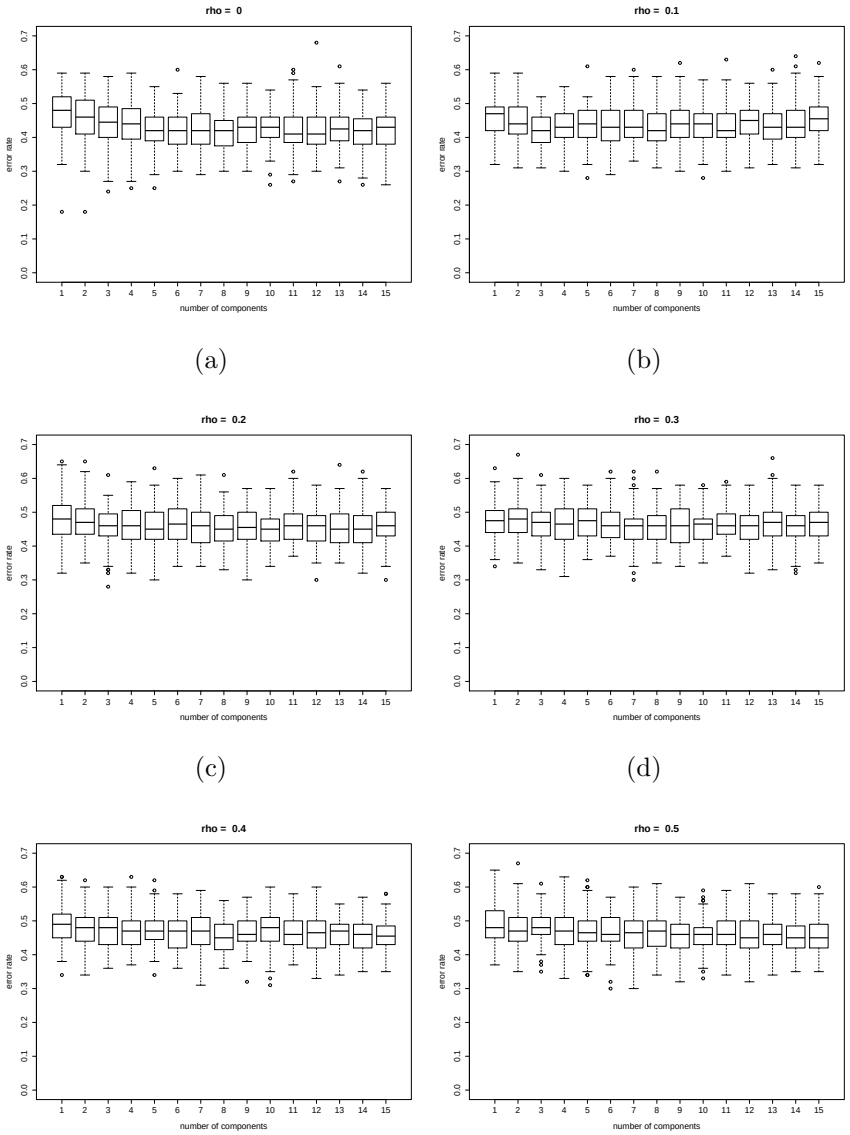
Figure 6.21: Boxplots of test set error rates of Classification Trees fitted using independent components of simulated datasets with large mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0, 0.1, \ldots, 0.5$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the first 1,2,...,15 independent components.



(a) (b)

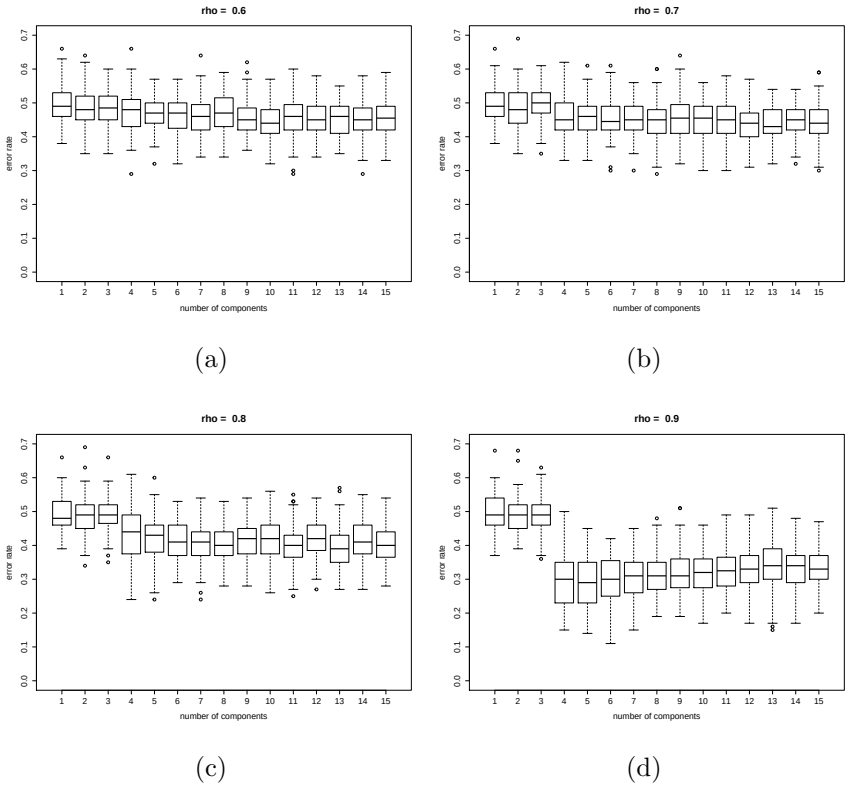
(c) (d)

Figure 6.22: Boxplots of test set error rates of Classification Trees fitted using independent components of simulated datasets with large mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0.6, 0.7, \ldots, 0.9$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the first 1,2,...,15 independent components.



(a) (b) (c) (d) (e) (f)

Figure 6.23: Boxplots of test set error rates of Classification Trees fitted using independent components of simulated datasets with small mean difference and three standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0, 0.1, \ldots, 0.5$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the first 1,2,...,15 independent components.



(a) (b)

(c) (d)

Figure 6.24: Boxplots of test set error rates of Classification Trees fitted using independent components of simulated datasets with three mean difference and one standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0.6, 0.7, \ldots, 0.9$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the first 1,2,...,15 independent components.

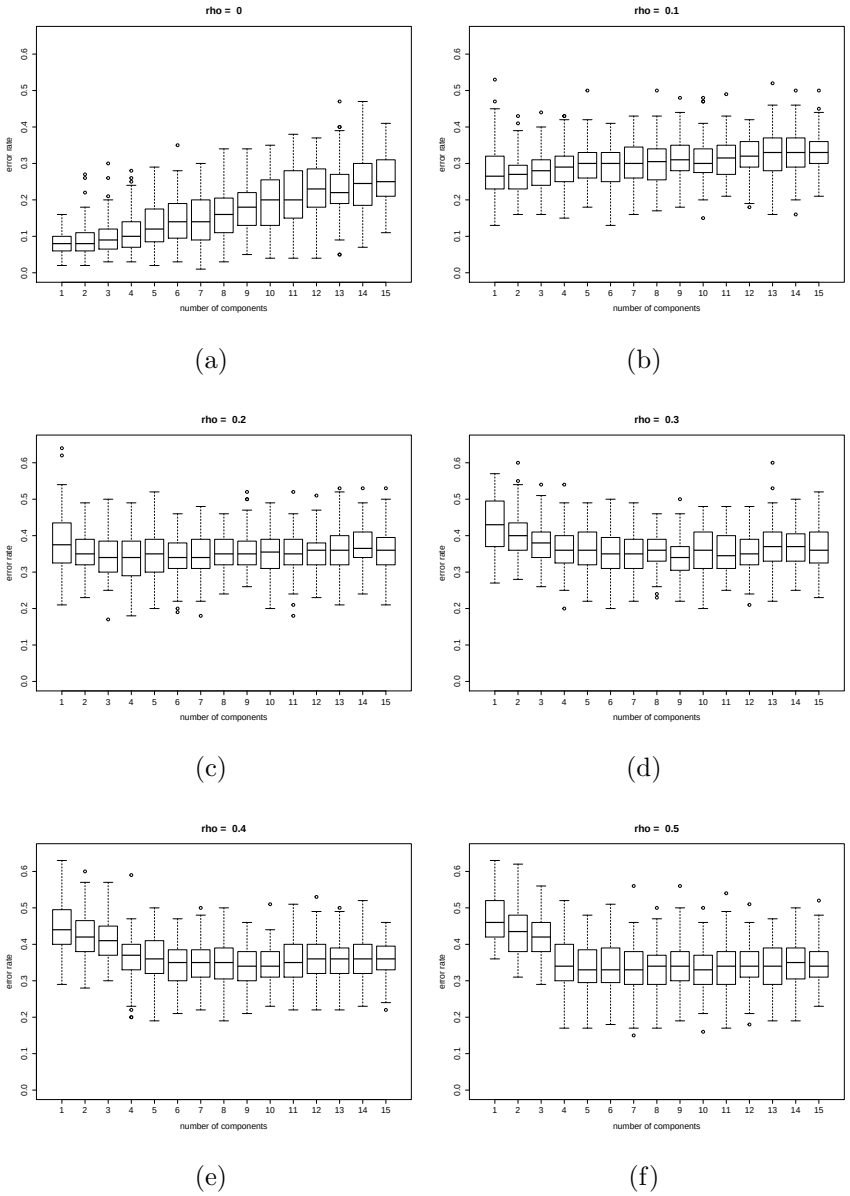
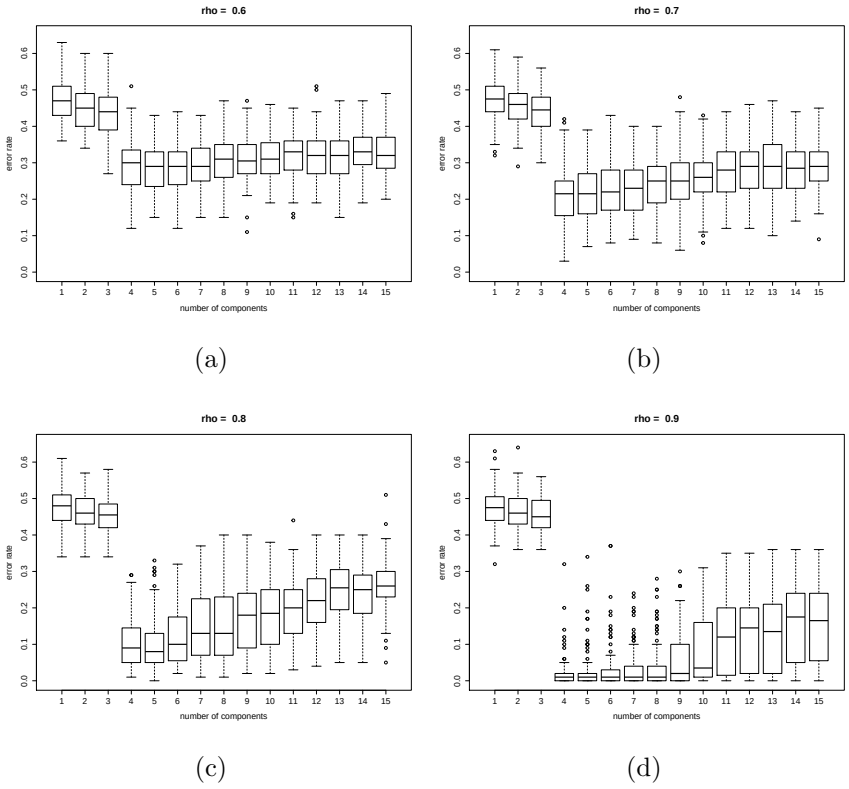


Figure 6.25: Boxplots of test set error rates of Classification Trees fitted using independent components of simulated datasets with medium mean difference and three standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0, 0.1, \ldots, 0.5$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the first 1,2,...,15 independent components.



(a) (b)

(c) (d)

Figure 6.26: Boxplots of test set error rates of Classification Trees fitted using independent components of simulated datasets with medium mean difference and three standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0.6, 0.7, \ldots, 0.9$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the first 1,2,...,15 independent components.

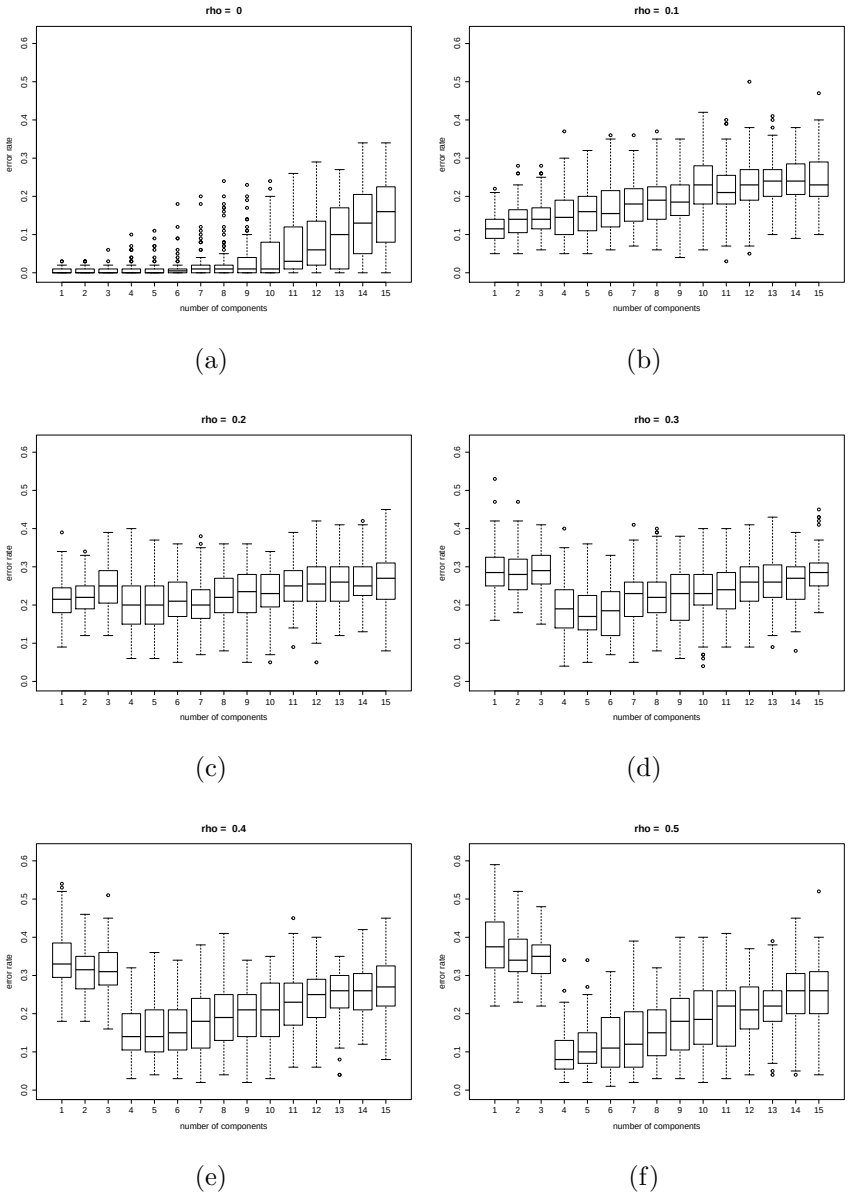
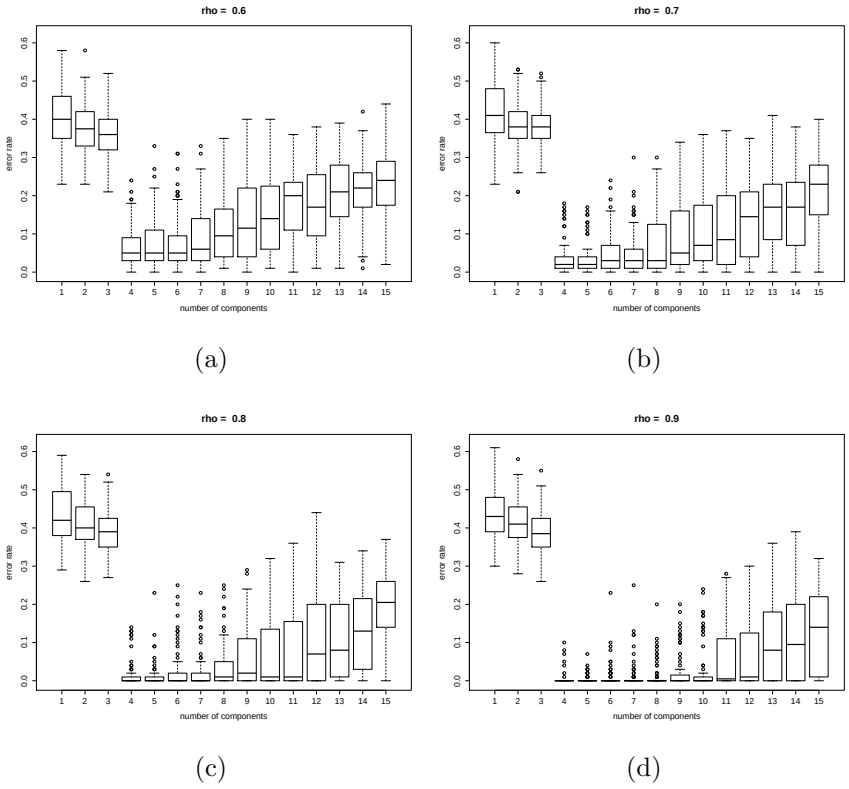


Figure 6.27: Boxplots of test set error rates of Classification Trees fitted using independent components of simulated datasets with large mean difference and three standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0, 0.1, \ldots, 0.5$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the first 1,2,...,15 independent components.



(a) (b)

(c) (d)

Figure 6.28: Boxplots of test set error rates of Classification Trees fitted using independent components of simulated datasets with large mean difference and three standard deviation. We obtain the results presented in each panel from different correlation settings. This figure shows the results for correlation $\rho = 0.6, 0.7, \ldots, 0.9$. Meanwhile, within each panel, there are 15 boxplots attained from fitting the trees using the first 1,2,...,15 independent components.

To figure out the results explained above, we begin by considering the whitened variables obtained during the estimation of independent component. Making use of Fast ICA algorithm of Hyvärinen & Oja (1997) to estimate independent components, one should applying whitening step prior to maximisation of non-Gaussianity step.

These whitened variables are obtained by involving singularvalue decomposition of data matrix X. On the other hand, applying PCA, one performs eigenvalue decomposition of sample covariance of data matrix X. As a result, both whitened variables and principal component scores share many structure in common.

Having obtained whitened variables of the same structure of principal component scores leads to the consequence that the estimated independent components obtained by maximising non-Gaussianity of these whitened variables also share many things in common with principal component scores. For these reason one might find the two prominent patterns of the resulting prediction error rate for Classification Trees of independent components as those of found for the trees of principal component scores.

However, unlike principal component scores, involving more independent components in fitting Classification Trees produces the trees of higher error rate. This might be explained that, carrying out maximisation of non-Gaussianity on the space spanned by many whitened variables which do not contain discriminative information just leads to the estimated independent components which have less discriminative information. One might find the explanation for such this phenomenon by looking at Figures 6.29 up to 6.34.

Each of these six figures consist of two panels. The top panel presents the plot for maximum goodness of split for each individual variable, while the bottom panel shows the plot for its p-value obtained from normality test. We display these six figures to explain how independent components inherit their discriminative information from whitened variables and how maximisation of non-Gaussianity on the space spanned by the whitened variables corresponds with the discriminative information contained in the resulting independent components.

To begin with, let us now consider the plots in Figure 6.29. This figure presents the plots for whitened variables obtained from the simulated datasets that are generated with the simulation settings of small mean difference, unit standard

deviation and $\rho = 0$. Each dot presented in two plots in this figure is the average for 100 estimation yielded from 100 simulated datasets generated using different seed setting.

Figure 6.29 shows that it is the first whitened variable is the only one with high goodnees of split. Therefore, once this whitened variable used in maximisation of non-Gaussianity, one will obtain the estimated indepenent component with discriminative information. Interestingly, this whitened variable is also the only non-Gaussian variable among 99 whitened variables obtained. This indicates that for our simulation study, the discriminative information that is presented as mean difference is the only data variation that leads to non-Gaussianity. It is should be noted here that, during the simulation study we only generate Gaussian variables. Thus, having the the first whitened variable as both the only non-Gaussian one as well as the only one that contains discriminative information, maximising the non-Gaussianity of this whitened variables improve their predictive performance whilst carrying out classification task.

With regard to the relation between the discriminative information contained in a whitened variable and its non-Gaussinity, we have the same explanation for Figures 6.30, 6.31, 6.32, 6.33 and 6.34. The differences across these figures are only regarding the position of the discriminative and non-Gaussian whitened variables and their maximum goodness of split values.

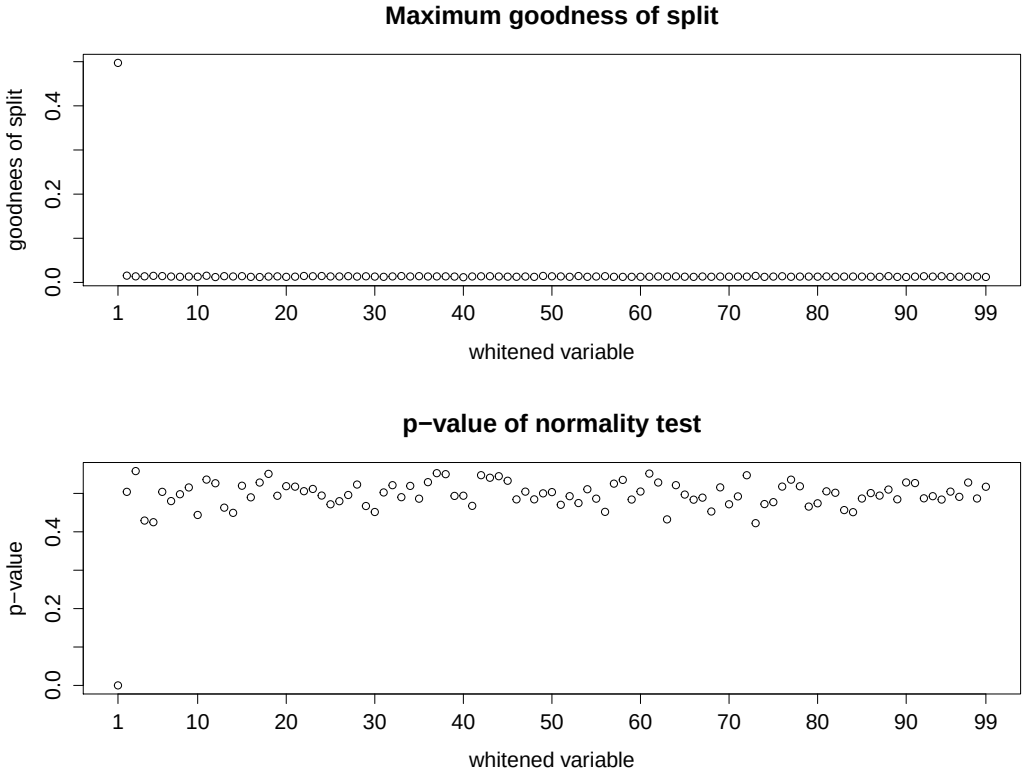


Figure 6.29: Line plots for maximum goodness of split and p-value of normality test for each individual whitened variable obtained from simulated datasets with small mean difference, unit standard deviation and $\rho = 0$. Panel (a) presents line plot for maximum goodness of split and Panel (b) shows line plot for p-value of normality test for each individual whitened variables. For both panels, we generate simulated datasets with small mean difference, unit standard deviation and $\rho = 0$. The horizontal axis in both panels corresponds with each individual independent component.

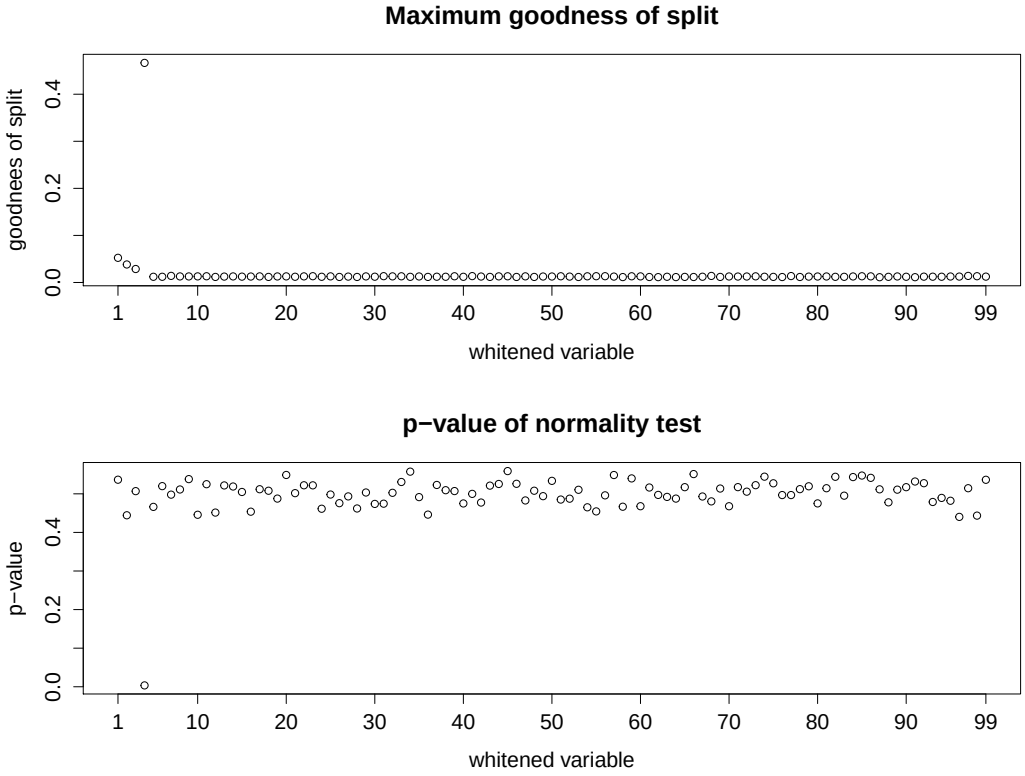


Figure 6.30: Line plots for maximum goodness of split and p-value of normality test for each individual whitened variable obtained from simulated datasets with small mean difference, unit standard deviation and $\rho = 0.9$. Panel (a) presents line plot for maximum goodness of split and Panel (b) shows line plot for p-value of normality test for each individual whitened variables. For both panels, we generate simulated datasets with small mean difference, unit standard deviation and $\rho = 0.9$. The horizontal axis in both panels corresponds with each individual independent component.

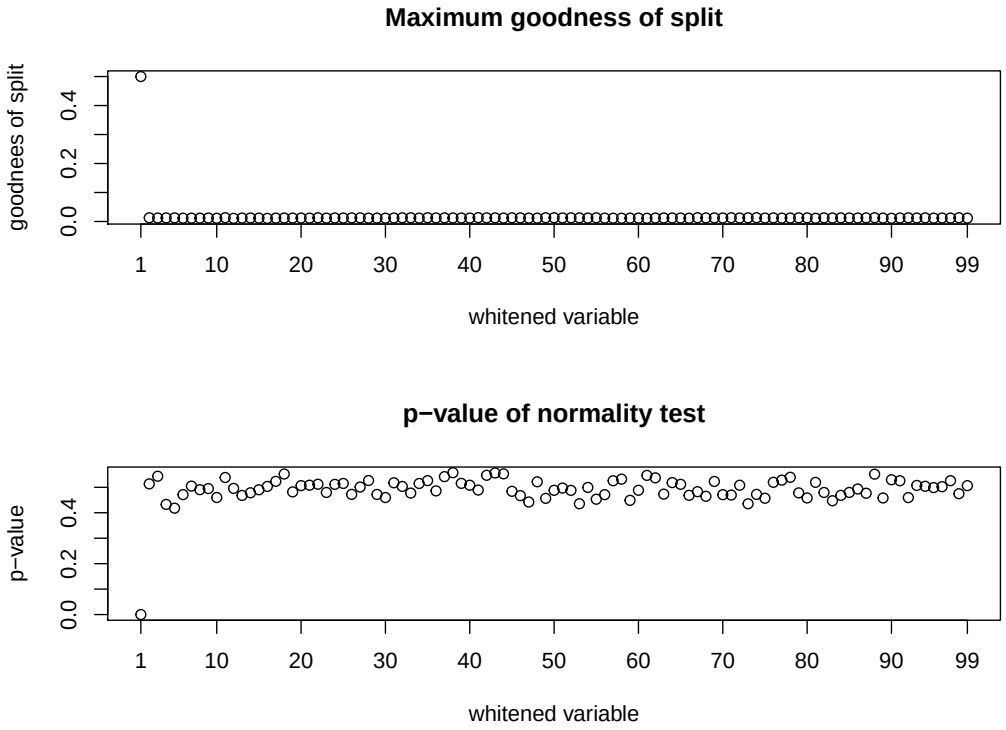


Figure 6.31: Line plots for maximum goodness of split and p-value of normality test for each individual whitened variable obtained from simulated datasets with large mean difference, unit standard deviation and $\rho = 0$. Panel (a) presents line plot for maximum goodness of split and Panel (b) shows line plot for p-value of normality test for each individual whitened variables. For both panels, we generate simulated datasets with large mean difference, unit standard deviation and $\rho = 0$. The horizontal axis in both panels corresponds with each individual independent component.

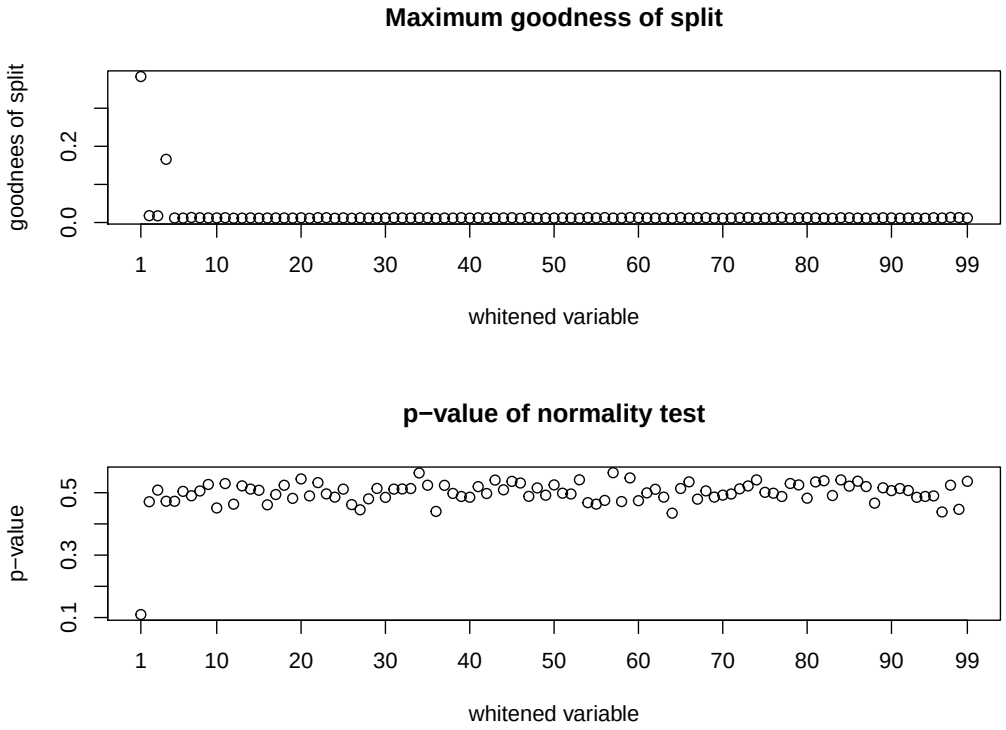


Figure 6.32: Line plots for maximum goodness of split and p-value of normality test for each individual whitened variable obtained from simulated datasets with large mean difference, unit standard deviation and $\rho = 0.9$ Panel (a) presents line plot for maximum goodness of split and Panel (b) shows line plot for p-value of normality test for each individual whitened variables. For both panels, we generate simulated datasets with large mean difference, unit standard deviation and $\rho = 0.9$. The horizontal axis in both panels corresponds with each individual independent component.

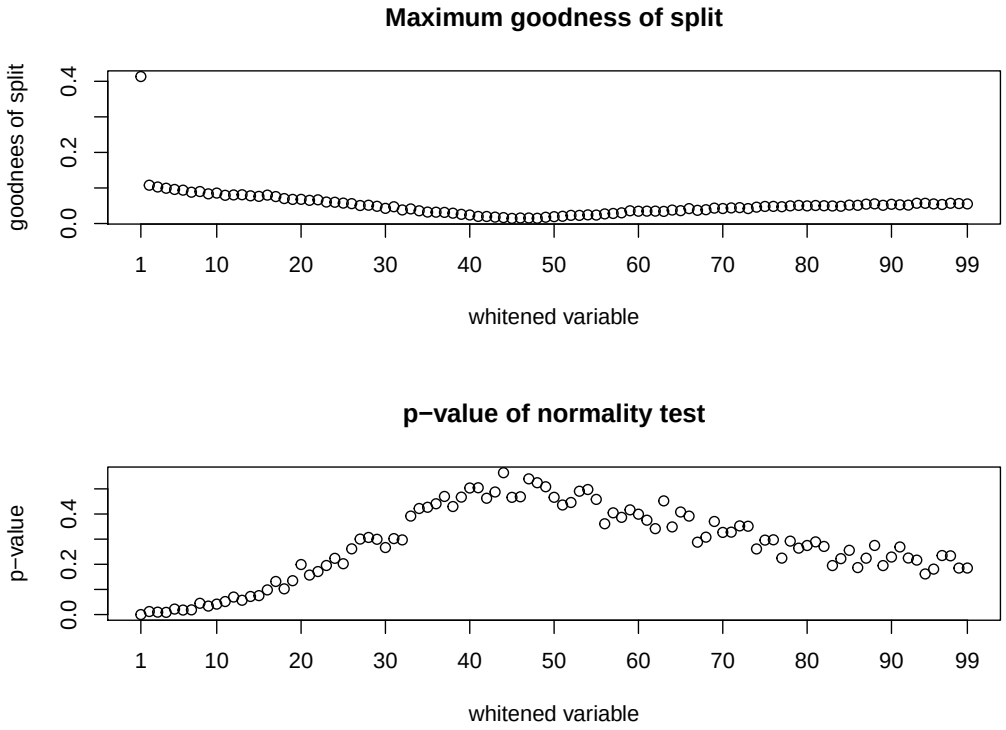


Figure 6.33: Line plots for maximum goodness of split and p-value of normality test for each individual whitened variable obtained from simulated datasets with small mean difference, three standard deviation and $\rho = 0$. Panel (a) presents line plot for maximum goodness of split and Panel (b) shows line plot for p-value of normality test for each individual whitened variables. For both panels, we generate simulated datasets with small mean difference, three standard deviation and $\rho = 0$. The horizontal axis in both panels corresponds with each individual independent component.

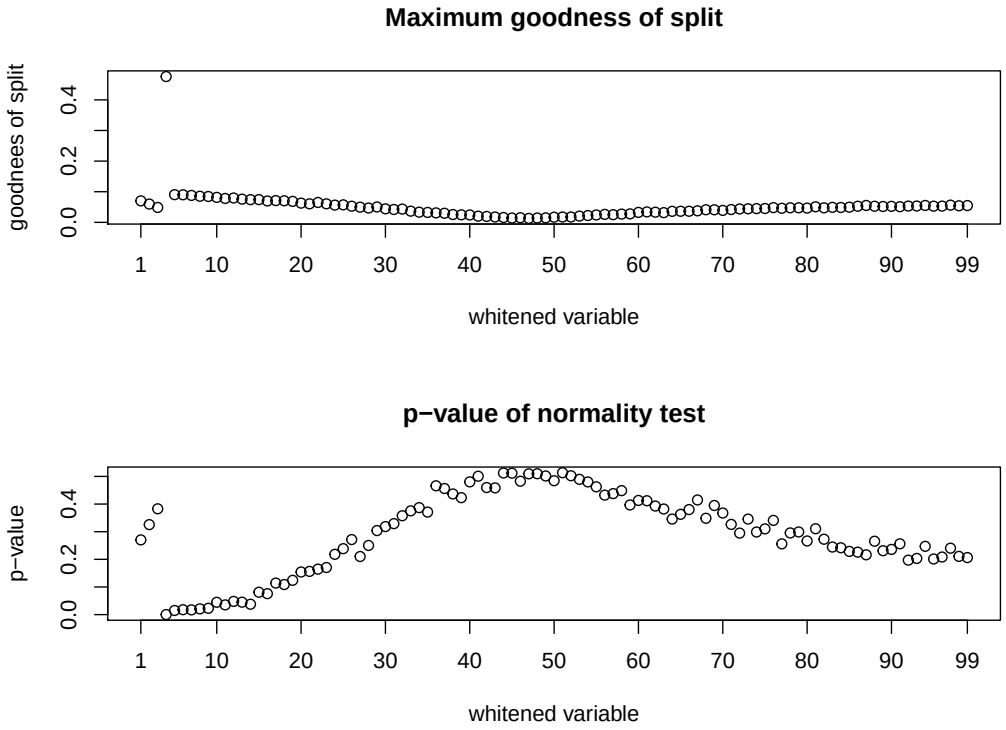


Figure 6.34: Line plots for maximum goodness of split and p-value of normality test for each individual whitened variable obtained from simulated datasets with small mean difference, three standard deviation and $\rho = 0.9$. Panel (a) presents line plot for maximum goodness of split and Panel (b) shows line plot for p-value of normality test for each individual whitened variables. For both panels, we generate simulated datasets with small mean difference, three standard deviation and $\rho = 0.9$. The horizontal axis in both panels corresponds with each individual independent component.

6.6 Conclusion

Applying ICA as a dimension reduction method prior to Classification Trees construction is not recommended as well. ICA algorithm requires whitening step which involves singularvalue decomposition. As a result, likewise for PCA, this step also forces a few number of whitened variables to congregate large portion of data variation which spreads across huge number of the original variables. Moreover, ICA also incorporates maximisation of non-Gaussianity. There is no underlying theorem that relates non-Gaussianity and the presence of discriminatory in dataset.

Compared to applying PCA prior to Classification Trees construction on real datasets, applying ICA before fitting Classification Trees gives less accurate trees. We end up with this result for the same reason as we have for applying PCA. On contrast, for hypothetical datasets, Classification Trees of independent components produce less prediction error than the trees of principal component scores.

Chapter 7

Random Forests

7.1 Introduction

This chapter consists of four sections: the construction of Random Forests, the review about existing works on the application of Random Forests on genomic data, the simulation study for applying Random Forests on large datasets and the application of Random Forests on our two real datasets. For constructing Random Forests, we describe the construction procedure for Random Forests of Breiman (2001a). We also explain hyperparameters required for fitting a Random Forest along with its generalisation error estimation. In addition, we review existing works on the application of Random Forests upon genomic data to identify the gap to be filled by the work we carry out.

For simulation study, we focus on two issues rise on the application of Random Forests for genomic data: having much more variables than observations and the presence of several blocks of correlated variables. It is not an appropriate practice to use the default hyperparameter settings for fitting Random Forests on such this dataset. Therefore, we perform simulation studies for finding optimal hyperparameters of a Random Forest in terms of prediction error and for figuring out how the correlation between predictors affect the performance of Random Forests. Having obtained optimal hyperparameters of Random Forests for large datasets, we build prediction model for our two real datasets using Random Forests.

7.2 Random Forests

A Random Forest is a prediction rule obtained by assembling a set of Classification Trees. Breiman (2001a) proposed Random Forests to significantly improve classification accuracy gained by a single Classification Trees of Breiman *et al.* (1984). Classification Trees suffer from high variability as previously discussed in Chapter 3. This variability becomes worse for small sample case.

As an ensemble method, a Random Forest is obtained by assembling a set of Classification Trees which are fitted using several bootstrapped samples drawn from the learning set. The information gained by the individual Classification Tree is then aggregated into a Random Forest in the form of the classification of observations and variables that are responsible for this classification task. For this reason, Random Forests can be seen as an ensemble method that involves bagging (bootstrap aggregation).

7.2.1 Constructing Random Forests

In this section we explain the procedure for constructing a Random Forest. We base our explanation here on Random Forest of Breiman (2001a) which is also implemented in an R package called **randomForest** (Liaw *et al.*, 2002). To begin with, let us introduce tuning parameters that will be used for constructing Random Forests. In this work we considered three tuning parameters: number of trees to be generated (*ntree*), minimum number of observations in each terminal node of individual tree (*nodesize*) and number of variables to be sampled for splitting each node of an individual tree (*mtry*). Probst *et al.* (2019) viewed the tuning parameter *ntree* as the one can be used to control the structure and size of the forest. Meanwhile, tuning parameters *nodesize* and *mtry* are the ones can be utilised to manage the size of each individual tree and to control the randomness of a forest, respectively.

Moreover, let us recall the learning set $\mathcal{L}$ which is a random sample of size n and consists of p-variate explanatory variables and univariate response variable as defined in Equation 3.2.1. Using this learning set, we fit a Classification Tree $r_{\mathcal{L}}$.

We begin the Random Forest construction by drawing *ntree* bootstrapped samples of the size n from learning set $\mathcal{L}$. We split observations of each bootstrapped sample into two disjoint subsets. The first subset contains two-thirds of the observations. We use this subset to fit a single Classification Tree of Breiman *et al.* (1984). This subset is also called as bagged sample. Let $\mathcal{L}_b^B$ be the b-th bagged sample, for $b = 1, 2, \ldots, ntree$.

The second subset consists another one third of observations. This subset is kept for prediction purpose and is named out-of bag (OOB) sample. We also use this subset for estimating the error rate of the forest. The Random Forest algorithm enables us to perform on-going prediction for the true error rate. We refer this estimate of the true error rate as Out-Of-Bag (OOB) error rate. Let $\mathcal{L}_b^{OOB}$ be the b-th OOB sample, for $b = 1, 2, \ldots, ntree$.

Returning to fitting of a Classification Tree using a bagged sample $\mathcal{L}_b^B$, the splitting of each node within individual tree of a Random Forest does not involve all p variables of the learning set $\mathcal{L}_b^B$. Instead, this splitting takes m out of p variables in, for $m < p$. We refer m as the tuning parameter *mtry* as mentioned before. Furthermore, each node of individual tree continues splitting until the resulting terminal nodes contain a certain number of observations. We call this quantity as tuning parameter *nodesize*. In addition, we should not do pruning over the trees generated for a Random Forest. Let $\mathcal{R}_{\mathcal{L}_b^B}$ be the Classification Tree fitted using the b-th bagged sample $\mathcal{L}_b^B$.

As previously mentioned, we can perform the on-going prediction for the response of each observation using OOB sample. To define the OOB prediction for the i-th observation $(\mathbf{x}_i^T, y_i)$, we need to define the set for the index of the OOB samples that contain this observation. Let $\mathcal{C}_i$ be such this set. We define $\mathcal{C}_i$ as

$$\mathcal{C}_i = \{b \mid (\mathbf{x}_i^T, y_i) \in \mathcal{L}_b^{OOB}\} \tag{7.2.1}$$

Using this notation, we define the predicted response for the i-th observation $\hat{y}_i^{OOB}$ as

$$\hat{y}_i^{OOB} = \underset{y_i \in \mathcal{Y}}{\operatorname{argmax}} \sum_{b \in \mathcal{C}_i} I(y_i = \mathcal{R}_{\mathcal{L}_b^B}(\mathbf{x}_i)) \tag{7.2.2}$$

where I is an indicator function and $i = 1, 2, \ldots, n$. This definition indicates that the OOB prediction of an observation is obtained by applying majority vote

over the set of predicted response upon this observation from individual tree. In addition, **randomForest** package provides us with **votes** values. These numbers present the proportions of an observation is classified with a class label.

Moreover, we can interpret the fitted Random Forest by making use of Variable Importance Measure (VIM). We utilise VIM to present the relative importance of contribution of each predictor variables in predicting the response. We use this quantity since we are no longer able to present the Random Forest into a simple numerical diagram.

7.2.2 Out-of-Bag Error Rate

We measure the prediction accuracy of Random Forests by calculating OOB error rate. This error rate is estimated in the place of the true error rate estimate. Bylander (2002) proposed that both the OOB error rate estimation and 10-fold cross validation have the same performance in predicting the test error rate. Liaw *et al.* (2002) recommended of making use of large enough *ntree* to obtain an accurate estimation.

OOB error rate estimate is an on-going estimate too. We perform the calculation of this quantity along with model fitting. This estimate is obtained by using OOB samples. By employing the OOB prediction of Equation 7.2.2, we define the OOB error rate as

$$Err^{OOB} = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i^{OOB}) \tag{7.2.3}$$

Cutler *et al.* (2012) reminded a common misconception in making use of this definition. It is inappropriate to obtain the OOB error rate of Random Forest by computing the OOB error rate for each tree and averaging these error rates to yield the OOB error rate of Random Forest. Instead, it should be obtained by using the OOB prediction of the forest.

7.2.3 Variable Importance Measures

As previously stated, we use Variable Importance Measure (VIM) to interpret the resulting Random Forests. Breiman (2002) proposes Mean Decrease Accuracy

(M1) and Mean Decrease Impurity (M4) to obtain Random Forests VIM.

Mean Decrease Accuracy (MDA) of the j-th variable, for $j = 1, 2, \ldots, p$ is defined as the decrease in Random Forest accuracy due to permutation of its values. Let $\{\mathcal{L}_b^{OOB}, b = 1, 2, \ldots, ntree\}$ be a collection of $ntree$ OOB samples drawn from the learning set $\mathcal{L}$. Let $\{\mathcal{L}_{b,j}^{OOB}, b = 1, 2, \ldots, ntree\}$ be a collection of $ntree$ OOB samples which have the values of the j-th variable has been randomly permuted. We define MDA of the j-th variable as

$$MDA_{X_j} = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{1}{|\mathcal{C}_i|}\sum_{b\in\mathcal{C}_i} I(y_i \neq \mathcal{R}_{\mathcal{L}_b^B}(\mathbf{x}_i)) - \frac{1}{|\mathcal{C}_i|}\sum_{b\in\mathcal{C}_i} I(y_i \neq \mathcal{R}_{\mathcal{L}_b^B}(\tilde{\mathbf{x}}_i))\right\} \quad (7.2.4)$$

where $\mathcal{C}_i$ is the set of the index of the OOB samples that contain the i-th observation as defined in Equation 7.2.1, $\mathbf{x}_i$ is the measurement over the i-th observation and $\tilde{\mathbf{x}}_i$ is the measurement over the i-th observation with the values of the j-th variable has been randomly permuted. The use of this quantity is based on the fact that if a variable is not important then rearranging its values should not diminish Random Forest prediction accuracy. This importance score also referred as Permutation Importance.

Furthermore, Mean Decrease Impurity (MDI) of the j-th variable, for $j = 1, 2, \ldots, p$ is viewed as the weighted decrease of impurity corresponding to the splits along the j-th variable and is averaged over all trees in the forest. To define MDI of the j-th variable for a forest, we begin with formulating MDI of the j-th variable for a single of an individual tree. Let t be a node in a tree $\mathcal{R}_{\mathcal{L}_b^B}$. We define MDI of the j-th variable for node t as

$$MDI_{X_j}^{(node.t)} = p_t\ \Delta Imp(s_t, t)\ I(split.var = j) \quad (7.2.5)$$

where p_t is proportion of observations fall into node t, $\Delta Imp(s_t, t)$ is the decrease in impurity due to the splitting of node t using the split s_t and $I(split.var = j)$ is the indicator function whose value 1 when the j-th variable roles as the splitting one. Moreover, we define MDI of the j-th variable for tree $\mathcal{R}_{\mathcal{L}_b^B}$ which is generated by the b-th bagged sample as

$$MDI_{X_j}^{(\mathcal{R}_{\mathcal{L}_b^B})} = \sum_{t=1}^{T_b} p_t\ \Delta Imp(s_t, t)\ I(split.var = j) \quad (7.2.6)$$

where T_b is the number of nodes within the b-th tree. Finally, for a Random Forest, we define MDI for the j-th variable as

$$MDI_{X_j} = \frac{1}{ntree} \sum_{T_b=1}^{ntree} \sum_{t=1}^{T_b} p_t \; \Delta Imp(s_t, t) \; I(split.var = j). \tag{7.2.7}$$

In addition, we can obtain both MDA and MDI for all variables in our datasets while applying **randomForest** package.

7.2.4 Hyperparameters of Random Forests

Breiman (2001a) proposed two important quantities that affect the performance of Random Forests, the strength of each individual tree within the forest and the correlation between the trees. The first quantity corresponds with the accuracy level of each individual tree, while the second one relates the dependence between the trees fitted by different in-bag bootstrapped sample.

Breiman (2001a) showed that these two quantities determined the upper bound of the generalisation error of a Random Forest. Hence, he proposed that adding more trees into a Random Forest does not lead to over-fit. Moreover, Breiman (2001a) also proved that one can improve Random Forest prediction accuracy by maximising the strength of each individual tree and minimising the correlation between them. Bernard *et al.* (2010) confirmed this result through empirical studies.

While constructing a Random Forest, we can control these two quantities by setting hyperparameters or tuning parameters such that they produce Random Forests with the optimal performance. One might maximise the strength of individual tree of Random Forest by controlling both tuning parameters *nodesize* and *ntrees*. Meanwhile, we might minimise the correlation between trees by managing the hyperparameter *mtry*. We can find the standard setting for these hyperparameters in Breiman (2002). This setting is also used in R package **randomForest** of Liaw *et al.* (2002).

For the hyperparameter number of trees, *ntree*, Breiman (2002) recommended one to use $ntree = 1000$ or even more. He suggested higher number of *ntree* in order to obtain a Random Forest with more stable estimation of Variable

Importance Measure. We are going to discuss this quantity in the following section. For *mtry*, Breiman (2002) proposed one to begin with $mtry = \sqrt{p}$ and then to try a value twice as high, $mtry = 2\sqrt{p}$, and half as low, $mtry = \frac{\sqrt{p}}{2}$. He also added that, in the presence of many noise variables, one should set *mtry* to be higher. We are going to perform some simulation studies to clarify this recommendation. Meanwhile, for the number of observations in each terminal node of individual tree, he recommended one to set $nodesize = 1$.

Moreover, the default settings used in **randomForest** of Liaw *et al.* (2002) are $ntree = 500$, $mtry = \sqrt{p}$ and $nodesize = 1$. However, this R package is also equipped with a function **tuneRF** for searching the optimal value of *mtry*. This function applies the setting for *mtry* as suggested by Breiman (2002).

These settings for hyperparameters of a Random Forest are related to the two quantities mentioned above: the strength and correlation of trees. By setting $nodesize = 1$, we let the individual Classification Tree to grow deeper to attain an unbiased tree. However, having such this kind of Classification Trees we end up with high variability. We could reduce the effect of this variability by building a large number of trees from bootstrapped samples. This can be carried out by setting the tuning parameter $ntree = 500$ or even more as can be found in Liaw *et al.* (2002). Larger number of *ntree* leads to more stable estimations for generalisation error and variable importance measure.

Furthermore, by setting $mtry = m = \sqrt{p}$ which gives $m < p$, we build different trees from each bootstrapped sample. Different Classification Trees then result different predictions which are low correlated. This in turn improves the performance of Random Forests. However, these hyperparamaters do not always yield an optimal Random Forest.

Díaz-Uriarte & De Andres (2006), who applied Random Forest as a gene selection and classification method suggested that by setting $ntree = 2000$ or $ntree = 5000$ is adequate. They also prescribed the use of $nodesize = 1$. In terms of the dependence between these three hyperparameters, they said that the setting of *mtry* is largely independent of *ntree* and *nodesize*.

Moreover, Díaz-Uriarte & De Andres (2006) mentioned that *mtry* is the most important parameter to be chosen and they proposed the default setting is often a good choice in terms of OOB error rate. However, based on their simulation

study, in the presence of very few relevant variables, Díaz-Uriarte & De Andres (2006) proposed that increasing $mtry$ might result to small decreases in error rate and decreasing $mtry$ might yield increases in the error rate.

Furthermore, Genuer *et al.* (2008) proposed the settings for hyperparameters of Random Forests for two different cases. Firstly, they considered the case where the number of variables is much lower than the number of observations, $p \lll n$. For this case, they recommended one to use the default setting of $mtry = \sqrt{p}$. Among the three settings of number of trees they examined, $ntree = 100, 500$ and 1000, they showed that $ntree = 500$ is enough. Meanwhile, using the setting $ntree = 100$ leads to significantly larger OOB error rate.

For the second case, where the number of variables is much greater than the number of observations $p \ggg n$, Genuer *et al.* (2008) recommended one to set higher value for $mtry$ to ensure that the selected subset of variables used for splitting captures the important variables. However, they did not consider the proportion of relevant variables in the datasets. For the number of trees, they considered $ntree$ = 100,500, 1000 and 5000 and ended up with suggestion that using $ntree = 500$ is adequate. For both cases considered above, They suggested one to fit Random Forests with $nodesize = 1$.

Next, Bernard *et al.* (2009) proposed that the default setting of $mtry = \sqrt{p}$ is a sensible choice, but can sometimes be improved. They also indicated the relation between the number of variables selected for splitting, $mtry$, and the number of informative variables in datasets. They proposed that if there are too few relevant variables in datasets, then using small $mtry$ will diminish each individual tree performance. On contrast, in the presence of large amount of relevant variables, choosing small $mtry$ will reduce the randomisation effects. Therefore, the choice of $mtry$ actually relies on a compromise between the attempt to attain uncorrelated trees and to ensure that the informative variables have the chance to be picked within this subset of $mtry$ variables. One can also view this as bias-variance trade-off. In addition, Bernard *et al.* (2009) only performed study for finding the optimal setting of $mtry$ and they set $ntree = 100$ for their experiments.

Goldstein *et al.* (2010) applied Random Forests on genome-wide association datasets to identify genetic regions which are related to a complex disease. They

also proposed the bias-variance trade-off in searching the optimal setting for $mtry$ as Bernard *et al.* (2009) did. Goldstein *et al.* (2010) performed study using mtry values of 1,2$\sqrt{p}$, $0.1p$, $0.5p$ and p and found that $mtry = 0.1p$ is the optimal setting.

In terms of the setting for the number of trees generated for a Random Forest, $ntree$, Goldstein *et al.* (2010) took the strength of relevant variables in datasets for performing classification task into account. The stronger the relevant variables in datasets, the quicker the Random Forest converges. Hence, in the presence of strong predictors, one needs less number of trees to obtain an optimal Random Forest. Based on their findings, Goldstein *et al.* (2010) suggested one to set $ntree = 1000$ or even less if there is strong predictor in dataset. Meanwhile, with weak predictors they recommended one to take more than 4000 trees.

Goldstein *et al.* (2010) also proposed the presence of the relation between the choice of the tuning parameter $mtry$ and the model complexity. Smaller value of $mtry$ leads to a more complex models which are less sparse. They also associated the choice of $ntree$ and the model complexity. Smaller value of $ntree$ yields to a less complex models which are more sparse. This indicates that the setting of $mtry$ is not independent $ntree$ as proposed by Díaz-Uriarte & De Andres (2006).

In genetic data, majority of variables can be seen as irrelevant variables. Therefore, to deal with such this data, we need to separate these irrelevant variables from the relevant ones by providing a sparse solution. Sparsity is a function of both $mtry$ and $ntree$, with a higher $mtry$ leading to greater sparsity and a higher $ntree$ leading to less sparsity.

In addition, Goldstein *et al.* (2010) also noticed the effect of the presence of blocks of correlated variables in genetic data. However, instead of recognising the impact of these blocks of variables on of Random Forest prediction accuracy, Goldstein *et al.* (2010) only noticed their impact on Variable Importance Measure.

Boulesteix *et al.* (2012) proposed the relation between dataset dimension and the choice of the tuning parameter $ntree$. For dataset with large number of variables, they suggested higher number trees. They recommended $ntree = 500$ or even more. However, they mentioned that this hyperparameter is not a real parameter as a larger value always gives better results than a smaller one. In contrast, they viewed the tuning parameter $mtry$ as a real one.

In the presence of large number of irrelevant variables, Boulesteix *et al.* (2012) recommended one to choose higher value for *mtry*. Small *mtry* might lead to the selection of suboptimal predictors. On the other hand, in the presence of many strong predictors, they suggested one to use lower value for *mtry*. In this case, small *mtry* might give a chance for predictors with moderate effects. Boulesteix *et al.* (2012) also added that the hyperparameter *nodesize* should also be seen as tuning parameters, but its influence on the results is expected to be lower.

Probst *et al.* (2019) confirmed the bias-variance trade-off in choosing hyperparameters for an optimal Random Forest. They also suggested higher value for the tuning parameter *ntree*. Among hyperparameters of Random Forests, Probst *et al.* (2019) mentioned that *mtry* is the most influential one. The best value of *mtry* depends on the number of variables that are related to the outcome.

7.3 Application on Real Datasets

In this section we discuss the results obtained from applying Random Forests on Smooth CNA and DNACopy CNA datasets. We apply Random Forests on these two datasets for fitting an accurate prediction model and getting insight on some features that contribute on the fitted model. For the first purpose, one should find the optimal hyperparameter setting that leads to a prediction model whose low error rate. As previously explained, there are three hyperparameters that should be set while fitting Random Forests: number of trees to be generated (*ntree*), number of variables to be selected as classifying variables for splitting a single node in an individual tree (*mtry*) and minimal number of observations in each terminal node of individual tree (*nodesize*).

Having obtained Random Forests whose lowest error rate along with their optimal hyperparameter setting, one might interpret the resulting Random Forests by comprehending the features that are responsible for data classification task carried out. This can be done by considering variable importance measures.

7.3.1 Random Forests of Smooth CNA dataset

In this subsection we present results obtained from applying Random Forests on Smooth CNA dataset. We begin by displaying the plots of OOB error rate of Random Forests with various hyperparameter settings. For all the cases considered, we set $nodesize = 1$. Meanwhile, for tuning parameter $ntree$, we study four settings: $ntree = 100$, 500, 1000 and 2000. We take these four $ntree$ settings into consideration to find the stable estimates of generalisation error and variable importance. In addition, we study the setting of $mtry$ as shown in Table 7.1.

Table 7.1: $mtry$ setting for Smooth CNA and DNACopy CNA datasets

Grouping	$mtry$
I	$1, 2, \ldots, 10$
II	$20, 30, \ldots, 100$
III	$200, 300, \ldots, 1000$
IV	$\frac{1}{2}\sqrt{p}, \sqrt{p}, 2\sqrt{p}$
V	$0.1p, 0.2p, \ldots, p$

We keep the first two groups of $mtry$ settings in mind as they are dealing with the cases of small $mtry$ options. In the presence of many weak relevant predictors, setting $mtry$ to become small ensures these weak relevant predictors to be selected as the classifying variable for splitting an individual node. On contrast, in the presence of few strong relevant predictors, setting $mtry$ to become large, as in the last two groups, ensures these strong relevant predictors to be picked as candidates for the classifying variable. Whereas, the third group of $mtry$ settings is considered following the recommended setting put forward by Breiman (2002) who suggested the making use of $mtry = \frac{\sqrt{p}}{2}, \sqrt{p}$ and $2\sqrt{p}$. In addition, all of these hyperparameter settings used for Smooth CNA dataset are also applied for DNACopy CNA dataset.

Furthermore, we discuss the variable importance measures to get the better insight of variables that are used for data classification. For computational purpose, we utilise both Mean Decrease Impurity (MDI) and Mean Decrese Accuracy (MDA) from **importance** function of **randomForest** package to perform this task.

Turning now to the plots of OOB error rates for Random Forests of Smooth CNA datasets as shown in Figure 7.1. These values are the average values of OOB error rates obtained from fitting Random Forests 100 times using different seeds. Figure 7.1 shows that among the four settings, $ntree = 500$, 1000 and 2000 produce almost the same pattern of OOB error rate estimates across various $mtry$ settings. For this reason, we decide to use results obtained from setting $ntree = 1000$.

With regard to the settings of $mtry$ which give Random Forests whose the lowest OOB error rate estimates, Figure 7.1 indicates that small $mtry$ settings which are much lower than $133 \simeq \sqrt{p}$ yield the lowest OOB error rate estimates. In order to display these results more clearly, we present the plots of OOB error rate estimates obtained from fitting Random Forests using the first 22 settings of $mtry$ as can be seen in Figure 7.2.

Figure 7.2 indicates that, for $ntree = 1000$, $mtry = 20$ yields Random Forests whose lowest OOB error rate estimate. Using this setting, we end up with $Err^{OOB} = 0.0663$. This prediction error is lower than cross-validated error rate of Classification Trees fitted using this dataset. We get $\widehat{Err}^{(rscv)} = 0.0843$ for the latter. Figure 7.3 presents boxplots for comparing the prediction error rate of Classification Trees and Random Forest with optimal tuning parameter settings for Smooth CNA dataset. Apart from the lower error rate, this figure also shows that Random Forests give more stable estimate than Classification Trees do. The variance of prediction error rate for Random Forests is lower than the variance of prediction error rate for Classification Trees.

Turning back to OOB error rate estimates for the four $ntree$ settings, we present four lowest OOB estimates for all $ntree$ settings in Table 7.2. As previously explained, in the presence of many weak relevant predictors in a dataset, setting $mtry$ become small enables these predictors to be chosen as a classifying variable in splitting a single node within an individual tree. Having obtained $mtry = 20$ as the optimal setting which produces Random Forests whose lowest error rate, one might see the presence of many weak relevant predictors in Smooth CNA dataset. We present Figure 7.4 to show the plots of variable importance for small settings of $mtry$. In addition, we also display Figure 7.5 to present the plots of variable importance for large settings of $mtry$, yet these settings also

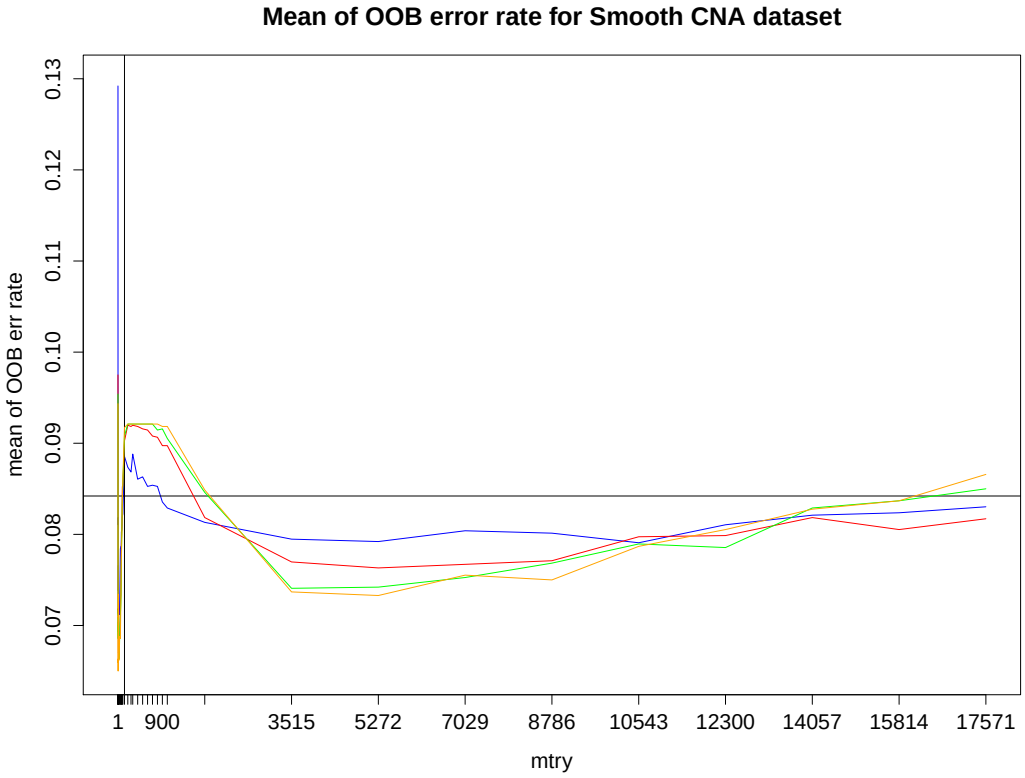


Figure 7.1: Plots for OOB error rates for Random Forests of Smooth CNA dataset with 42 settings of the number of variables selected for splitting a node in an individual tree $mtry$ as presented in Table 7.1 and four settings of number of trees $ntree$. The blue line presents results obtained for $ntree = 100$, the red one shows results gained for $ntree = 500$, the green one displays results attained for $ntree = 1000$ and the orange one shows results yielded for $ntree = 2000$. The black vertical line represents the recommended setting $mtry = \sqrt{p} \simeq 133$, whereas the black horizontal line shows the cross-validated error rate of Classification Trees. We obtained the true error rate estimate $\widehat{Err}^{(rscv)} = 0.0843$ for Classification Trees.

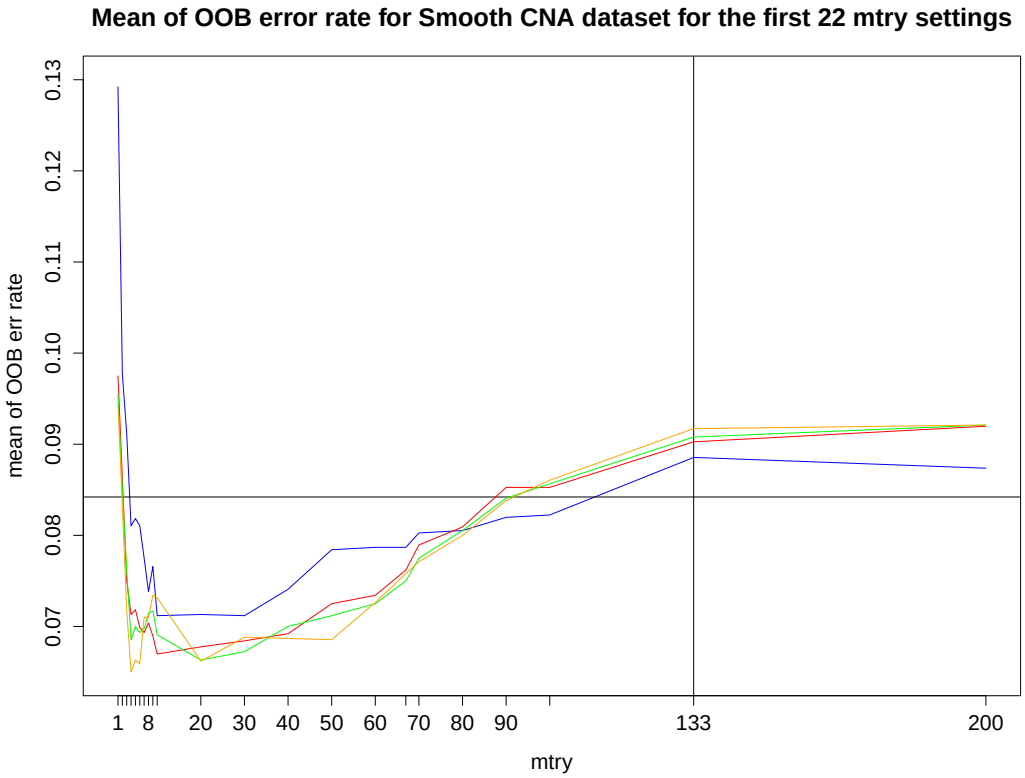


Figure 7.2: Plots for OOB error rate for Random Forests of Smooth CNA dataset for the first 22 settings of *mtry* shown in Table 7.1 and four settings of number of trees *ntree*: 100, 500, 1000 and 2000.

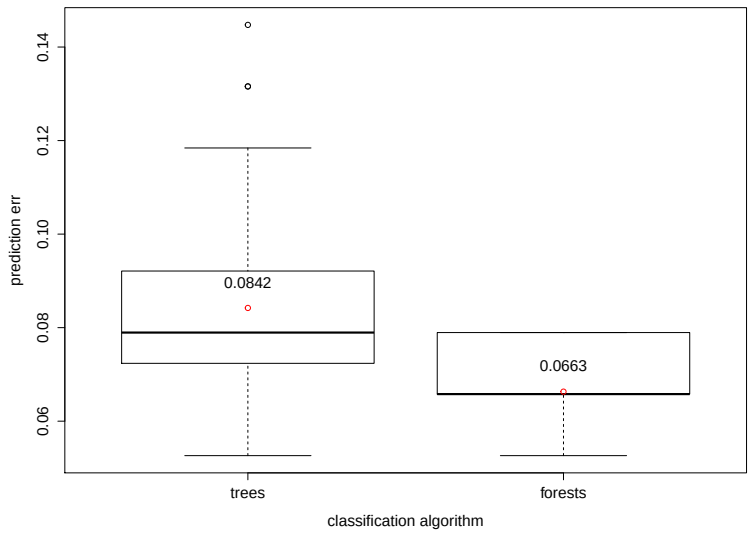


Figure 7.3: Boxplot of prediction error rates for Classification Trees and Random Forests of Smooth CNA dataset. For Classification Trees, the error rate presented is cross-validated error rate. Meanwhile, for Random Forests, the result shown is OOB error rate obtained for the forest with $ntree = 1000$, $nodesize = 1$ and $mtry = 20$.

correspond with Random Forests whose low OOB error rate estimates. For all plots of variable importance shown in Figures 7.4 and 7.5, we use MDI as variable importance measures.

Table 7.2: Four lowest OOB error rate estimates for Random Forests of Smooth CNA datasets

ntree	*mtry*	OOB error rate
100	10	0.0711
	30	0.0711
	20	0.0713
	8	0.0738
500	10	0.0670
	20	0.0678
	30	0.0684
	9	0.0689
1000	20	0.0663
	30	0.0672
	4	0.0686
	10	0.0691
2000	4	0.0650
	6	0.0659
	20	0.0662
	5	0.0663

Combining information presented in Figures 7.2 and 7.4, one might see the relation between the resulting OOB error rate estimates shown in the first figure with the pattern of MDI for all variables with small *mtry* settings. As *mtry* increases from 1 up to 20, one might find OOB error rate decreases. Using the information visualised in Figure 7.4, one might see that this decline of OOB error rate for $mtry = 1$ up to 20 relates to the increase of MDI of variables within genomic location of Chromosome 3 followed by the decrease of MDI of variables outside this genomic location. This indicates that Chromosome 3 consists of number of strong relevant predictors. Setting too small *mtry* prevent these variables

to be selected as the candidates of classifying variables.

Apart from variables within Chromosome 3, there are also many variables spread across several genomic locations which can be categorised as relevant predictors. These variables also have contribution in constructing the Random Forests, but their contribution might be not as high as the ones of Chromosome 3. However, leaving these variables while constructing a forest one might end up with less accurate forest. This can be seen for the case of fitting Random Forests with large $mtry$.

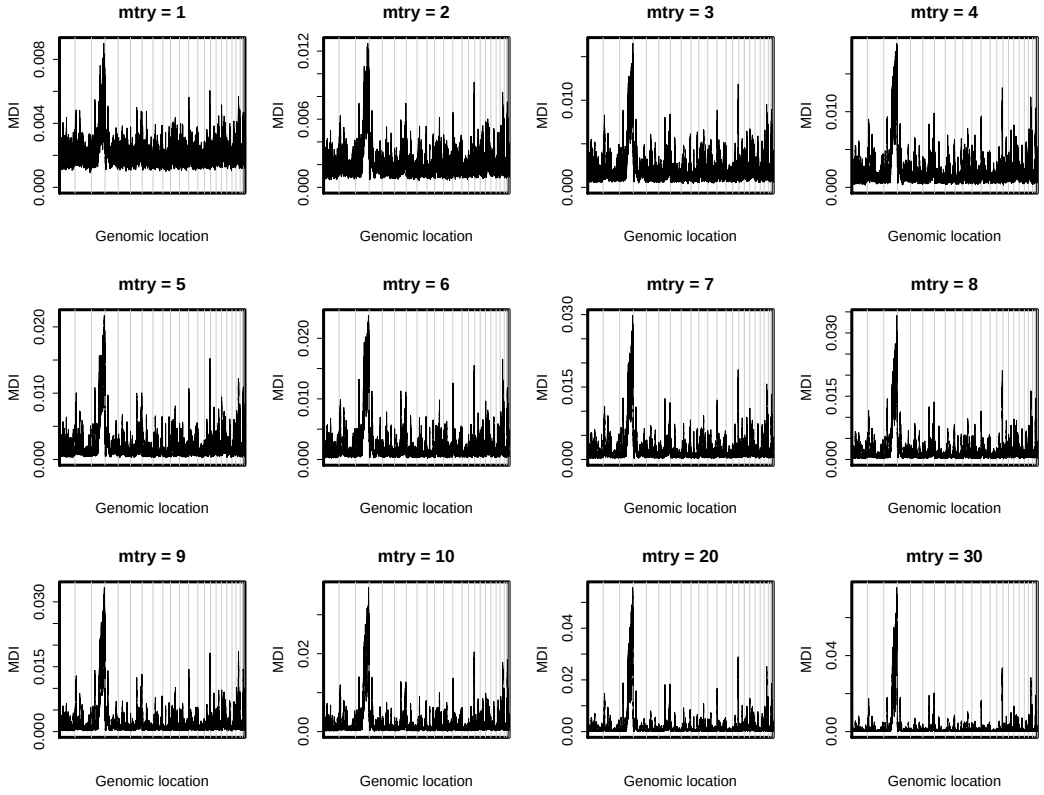


Figure 7.4: Plots for the average values of MDI for all variables of Random Forests of Smooth CNA dataset for $mtry = 1, 2, \ldots, 30$, $ntree = 1000$ and $nodesize = 1$. These MDI values are obtained from fitting Random Forests on Smooth CNA dataset 100 times with different seeds.

Returning to the plots for OOB error rate estimates in Figure 7.1, it apparent

that the resulting forests with large $mtry$ are less accurate than the forests with small $mtry$. In order to figure out this result, let us now look at MDI plots in Figure 7.5. Each panel of this figure presents the plot of MDI for large $mtry$ settings. It is obvious from this figure that, for these $mtry$ settings, it is small number of variables within genomic location of Chromosome 3 that are mostly used as classifying variable.

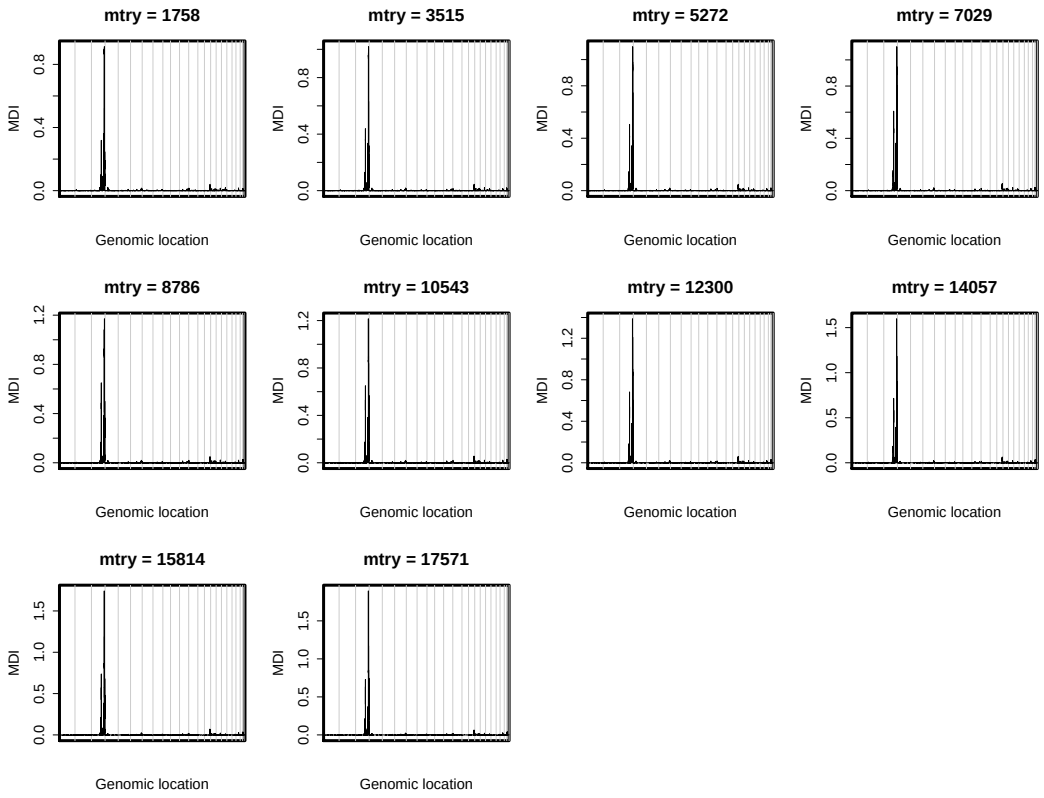


Figure 7.5: Plots for the average values of MDI for all variables of Random Forests of Smooth CNA dataset for $mtry = 0.1p, 0.2p, \ldots, p$, $ntree = 1000$ and $nodesize = 1$. These MDI values are obtained from fitting Random Forests on Smooth CNA dataset 100 times with different seeds.

One obtains such these results since by setting $mtry$ large, one gives higher chance to strong predictor variables to be selected as a member of subset of variables which will be used for splitting a single node within an individual tree.

As the consequence, these strong predictor variables will be chosen much more frequent than the rest of variables to carry out the splitting task. Thus, setting $mtry$ large lowers the contribution of weak relevant predictors in fitting Random Forests. Among large $mtry$ setting, we obtain $mtry = 3515$ which gives Random Forests whose the lowest OOB error rate. It yields $Err^{OOB} = 0.0740$.

These findings confirm those of Classification Trees as in Section 4.2.1. As previously explained, the resulting Classification Trees only involve one out of tens of thausands variables to carry out the classification task. As a result, we end up with $\widehat{Err}^{(rscv)} = 0.0843$

Furthermore, we discuss a realisation of Random Forest fitted by setting $nodesize = 1$, $ntree = 1000$ and $mtry = 20$ and 3515 with $seed = 1$. We begin with Random Forest with $mtry = 20$. Using these setting, we obtain Random Forest whose 7.89% OOB error rate. Moreover, Figure 7.6 presents the plots for variable importance measures using both MDA and MDG for this forest. Figure 7.6 shows that the top 30 variables whose highest either MDI or MDA are variables within genomic location of Chromosome 3. However, in terms of MDI, there is one variable of Chromosome 6 which has high MDI.

On the other hand, fitting Random Forest with large $mtry$, one ends up with a forest which has only variables of Chromosome 3 as variables with high importance as can be found for Random Forest with $mtry = 3515$. In this case, we obtain a forest whose 9.21% OOB error rate. We present Figure 7.7 to show MDI and MDA for the top 30 variables. This figure indicates that setting $mtry = 3515$, one obtains Random Forests which has only variables of Chromosome 3 as the classifying variables.

This result certifies the result obtained from using the Classification Trees in the genetic point of view. As previously mentioned in Section 4.2.1 of Chapter 4, these variables which correspond to the windows in loci 3q24 up to 3q27.3 are the ones that can be used to prognosticate squamous carcinoma lung cancer. The amplification in genes SOX2 and PIK3CA within these loci is common in squamous carcinoma lung cancer.

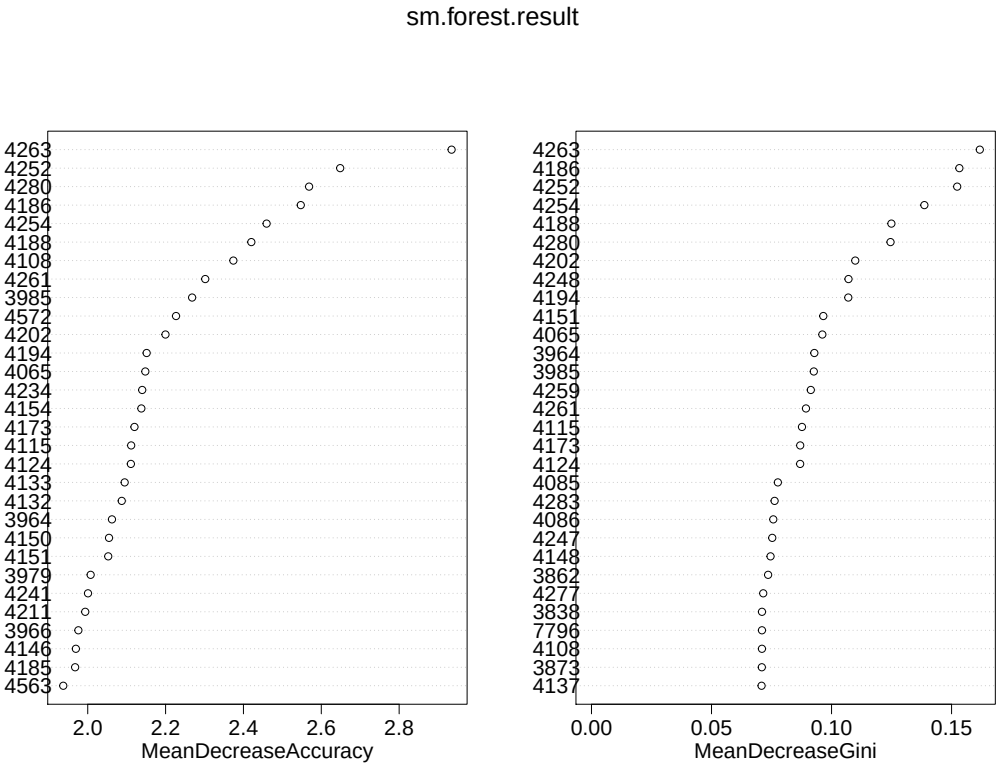


Figure 7.6: Plots of the top 30 variable importances for Random Forests of Smooth CNA dataset for $nodesize = 1$, $ntree = 1000$ and $mtry = 20$ with $seed = 1$.

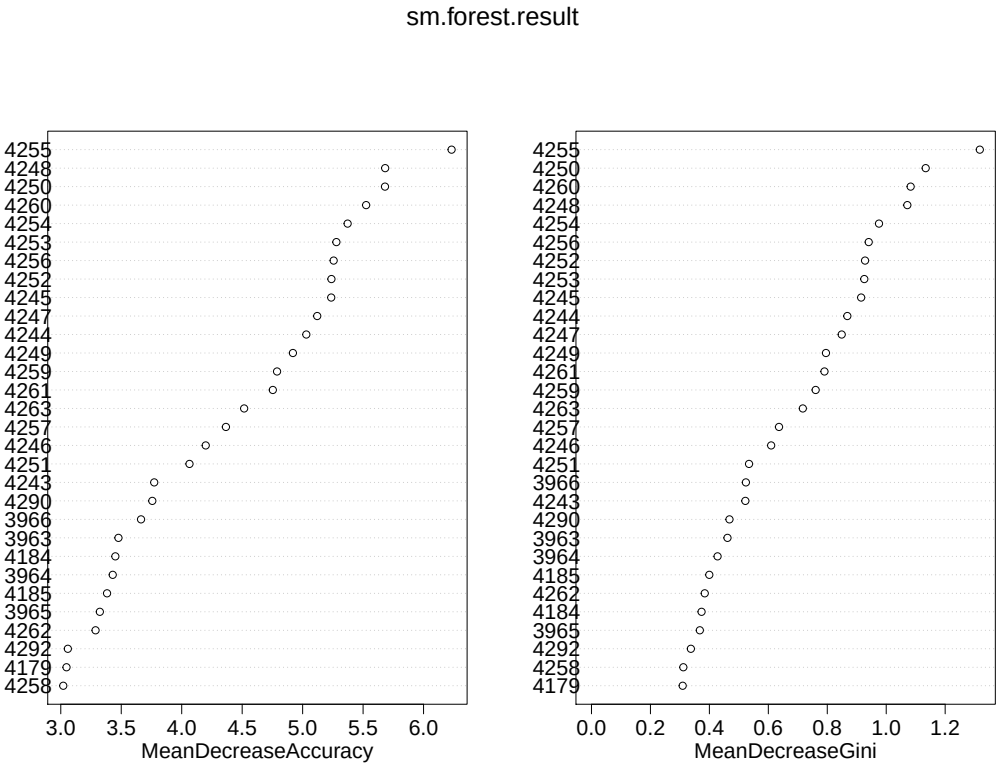


Figure 7.7: Plots of the top 30 variable importance for Random Forests of Smooth CNA dataset for $nodesize = 1$, $ntree = 1000$ and $mtry = 3515$ with $seed = 1$.

Moreover, we present the plots of computational time required for fitting Random Forests of Smooth CNA dataset. We display these plots in Figure 7.8. This figure shows the computational time required to fit Random Forests with seed = 1 and $nodesize = 1$ for various $ntree$ and $mtry$ settings. This figure indi-

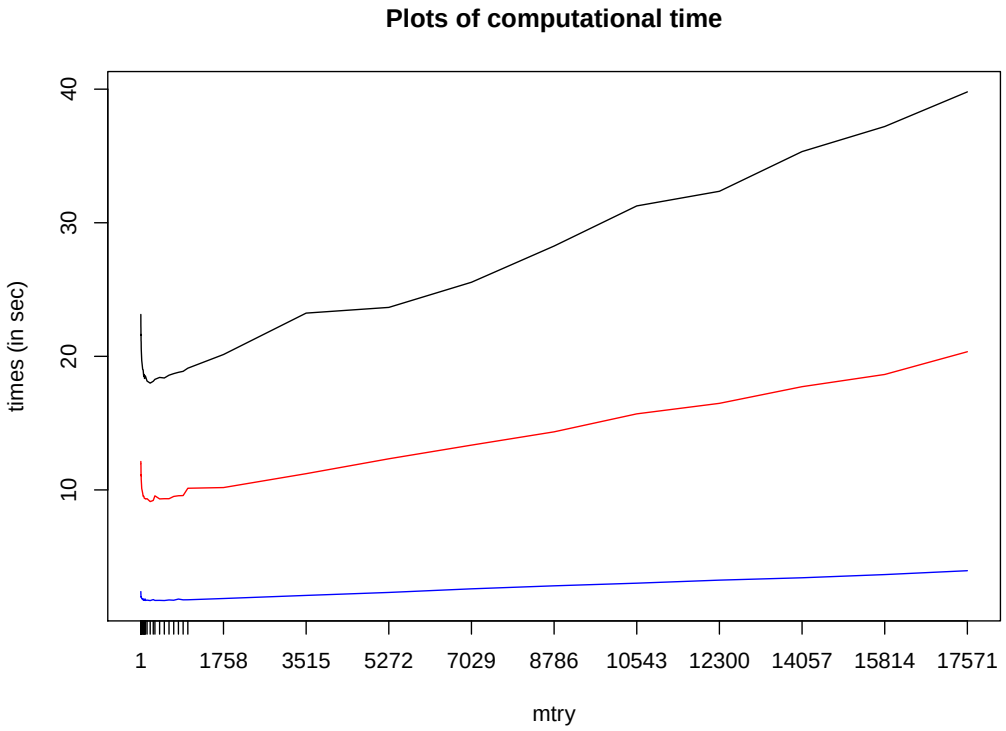


Figure 7.8: Plots of computational time for fitting Random Forests of Smooth CNA dataset. We fit the Random Forests with seed = 1 for $nodesize = 1$ and various $mtry$ settings as shown within Table 7.1. The blue line corresponds with the computational time required for fitting Random Forests with $ntree = 100$. The red line corresponds with the computational time required for fitting Random Forests with $ntree = 500$. The black line corresponds with the computational time required for fitting Random Forests with $ntree = 1000$.

cates that fitting Random Forests for larger $ntree$ leads to longer computational time. Likewise, fitting the forests for larger $mtry$ also leads to longer compu-

tational time. We end up with 19.6933 seconds to fit a Random Forest with $nodesize = 1$, $ntree = 1000$ and $mtry = 20$. These hyperparameter setting gives Random Forests with the lowest OOB error rate.

7.3.2 Random Forests of DNACopy CNA dataset

This subsection discusses the results obtained from utilising Random Forests on DNACopy CNA dataset. As previously applied for Smooth CNA dataset, we now present the plots of OOB error rate of Random Forests with various hyperparameter settings. We consider the cases of tuning $nodesize = 1$; $ntree =$ 100, 500, 1000 and 2000 and $mtry$ as presented in Table 7.1.

Let us now turn to discuss OOB estimates shown in Figure 7.9. Looking at this figure, it is apparent that setting the hyperparameter $ntree$ to be either 500, 1000 or 2000 produces Random Forests whose the nearly same OOB error rates across various $mtry$ settings. As the consequence of this, we decide to use results obtained from setting $ntree = 1000$ as we also applied from Smooth CNA dataset.

Moreover, it is also obvious from Figure 7.9 that, for Random Forests of DNACopy dataset with four settings of $ntree$, one ends up with Random Forests whose lowest OOB error rates for $mtry = 18388 = p$. Table 7.3 presents those estimates. This table shows that those estimates are much lower than the cross-validated error rate of Classification Trees fitted using this dataset. We get $\widehat{Err}^{(rscv)} = 0.1318$ for the Classification Trees as shown in Figure 7.10.

Table 7.3: Lowest OOB error rate estimates for Random Forests of DNACopy CNA datasets

$ntree$	OOB error rate
100	0.0820
500	0.0775
1000	0.0751
2000	0.0720

Figure 7.10 presents boxplots for comparing the prediction error rate of Classification Trees and Random Forests with optimal tuning parameter settings for DNACopy CNA dataset. Figure 7.10 shows that Random Forests do no only give lower prediction error rate than Classification Trees do, but also produce error rate estimates with lower variance. This indicates that Random Forests give more stable estimates of generalisation error than Classification Trees do.

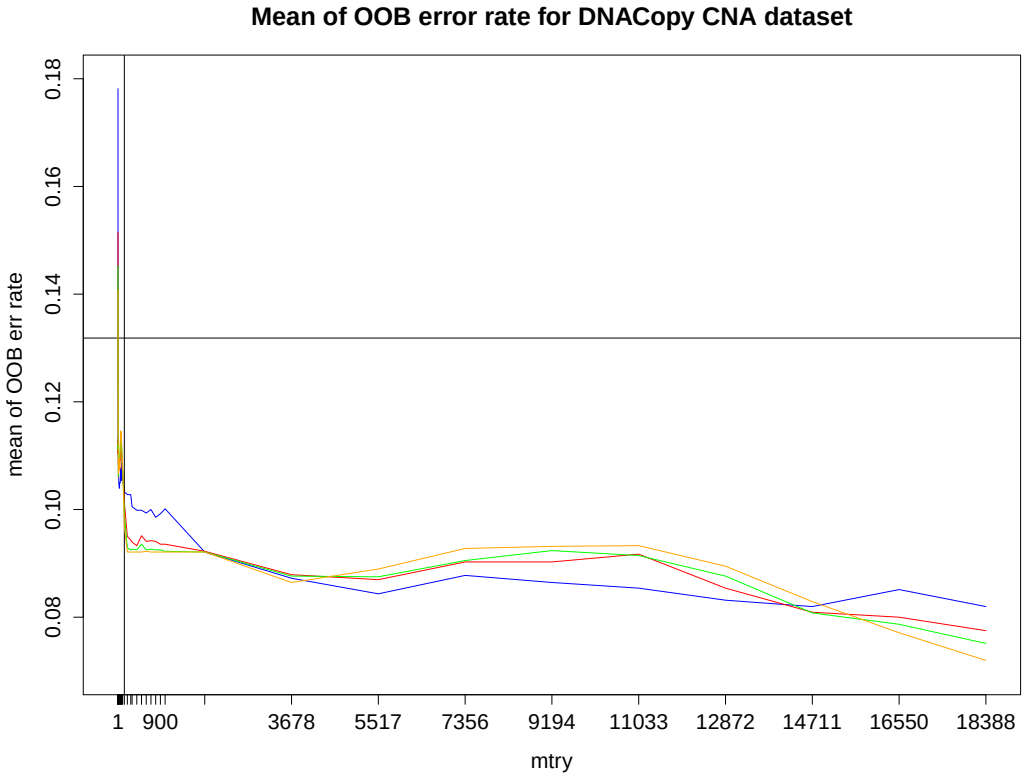


Figure 7.9: Plots for OOB error rates for Random Forests of DNACopy CNA dataset with 42 settings number of variables selected for splitting a node in an individual tree $mtry$ as presented in Table 7.1 and four settings of number of trees $ntree$. The blue line presents results obtained for $ntree = 100$, the red one shows results gained for $ntree = 500$, the green one displays results attained for $ntree = 1000$ and the orange one shows results yielded for $ntree = 2000$. The black vertical line represents the recommended setting $mtry = \sqrt{p} \simeq 136$, whereas the black horizontal line shows the cross-validated error rate of Classification Trees. We obtained the true error rate estimate $\widehat{Err}^{(rscv)} = 0.1318$ for the Classification Trees.

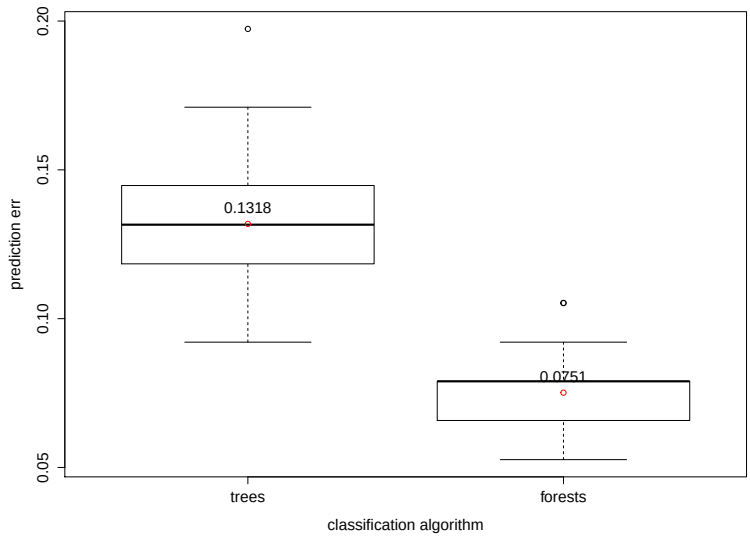


Figure 7.10: Boxplots of prediction error rates for Classification Trees and Random Forests of DNACopy CNA dataset. For Classification Trees, the error rate presented is cross-validated error rate. Meanwhile, for Random Forests, the result shown is OOB error rate obtained for the forest with $ntree = 1000$, $nodesize = 1$ and $mtry = 18388 = p$.

Returning to OOB error rate estimates for Random Forests of DNACopy CNA dataset, one might see from Figure 7.9 that the minimum values of OOB error rate are achieved for $mtry = 18388 = p$ after gradually decrease from $mtry = 8$. Meanwhile, from $mtry = 1$ up to $mtry = 7$, the estimates of OOB error rate drop significantly as can be viewed in Figure 7.11. This figure only shows OOB error rate estimates for the first 22 $mtry$ settings.

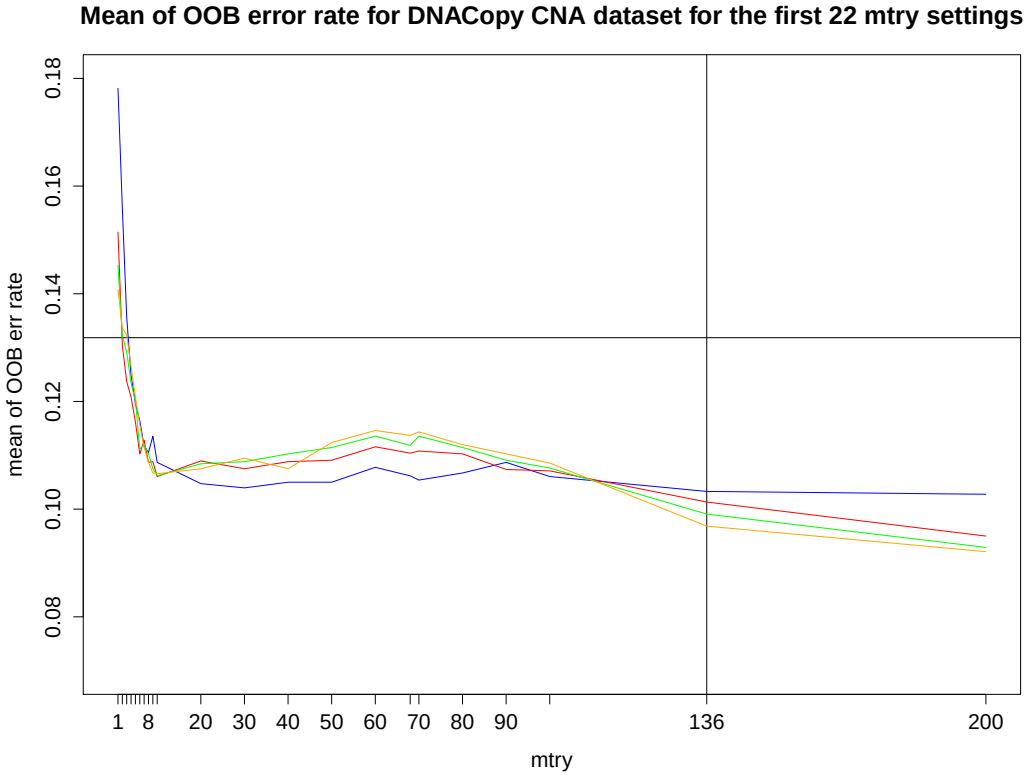


Figure 7.11: Plots for OOB error rate for Random Forests of DNACopy CNA dataset for the first 22 settings $mtry$ shown in Table 7.1 and four settings of number of trees $ntree$: 100, 500, 1000 and 2000.

To figure out what causes this significant drop, let us now look at the plots of variable importance measures obtained for these $mtry$ setting as presented in Figure 7.12. Connecting the results shown in Figures 7.11 and 7.12, one might see that the significant drop of OOB error rate estimates for $mtry = 1$ up to 7

relates to the increase of MDI for variables within Chromosomes 3 and 10 along with the decrease of MDI for variables of other genomic locations. This indicates that variables within Chromosomes 3 and 10 have large contribution in fitting Random Forests for DNACopy dataset.

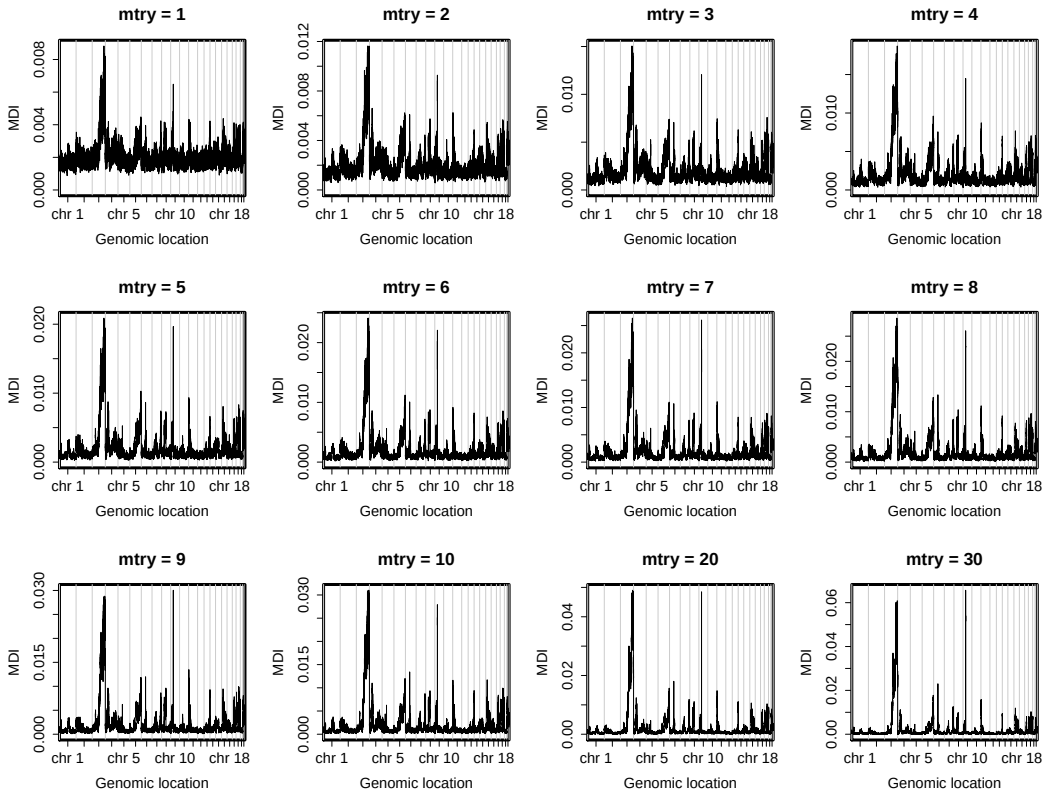


Figure 7.12: Plots for the average values of MDI for all variables of Random Forests of DNACopy CNA dataset for $mtry = 1, 2, \ldots, 30$. These MDI values are obtained from fitting Random Forests on DNACopy CNA dataset 100 times with different seeds.

Furthermore, for large *mtry* settings, one might see the decrease of OOB error rate estimates that is also followed by the increase of MDI of variables of Chromosome 10 as shown in Figure 7.13. This indicates variables of Chromosome 10 have higher contribution in fitting the forests than variables of the other genomic locations, including those of Chromosome 3. However, variables of this

genomic location still have contribution in constructing Random Forests as their have non-zero variable importance measures.

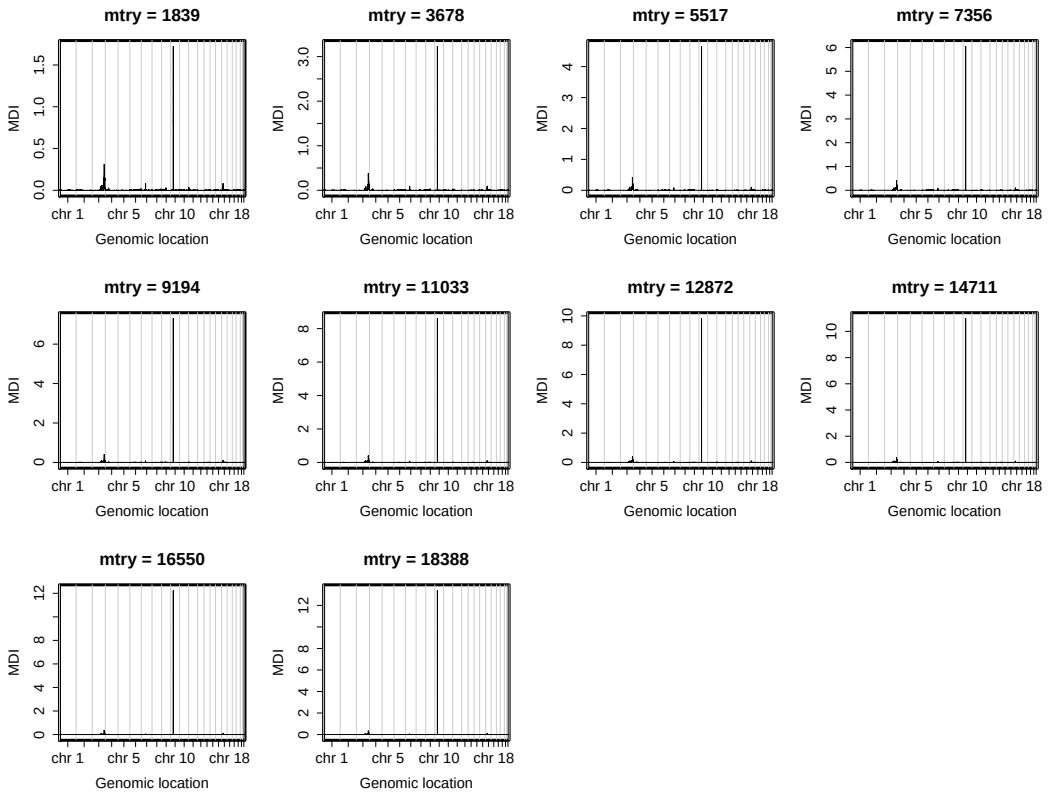


Figure 7.13: Plots for the average values of MDI for all variables of Random Forests of DNACopy CNA dataset for $mtry = 0.1p, 0.2p, \ldots, p$. These MDI values are obtained from fitting Random Forests on DNACopy CNA dataset 100 times with different seeds.

Moreover, we discuss a realisation of Random Forest fitted using DNACopy CNA dataset by setting $nodesize = 1$, $ntree = 1000$ and $mtry = 18388$ with $seed = 1$. Using these setting, we obtain Random Forest whose 7.89% OOB error rate. Figure 7.14 presents the plots for variable importance measures using both MDA and MDG for this forest.This figure shows the top 30 variables whose highest either MDI or MDA. Among these 30 variables, four of them are variables of Chromosome 10. Whereas, the other 26 variables are those of Chromosome

3. However, one out of the four variables of Chromosome 10, X_{11284}, has much higher variable importance than the other variables. Hence, variable X_{11284} is the one which has highest contribution in fitting Random Forests of DNACopy CNA dataset.

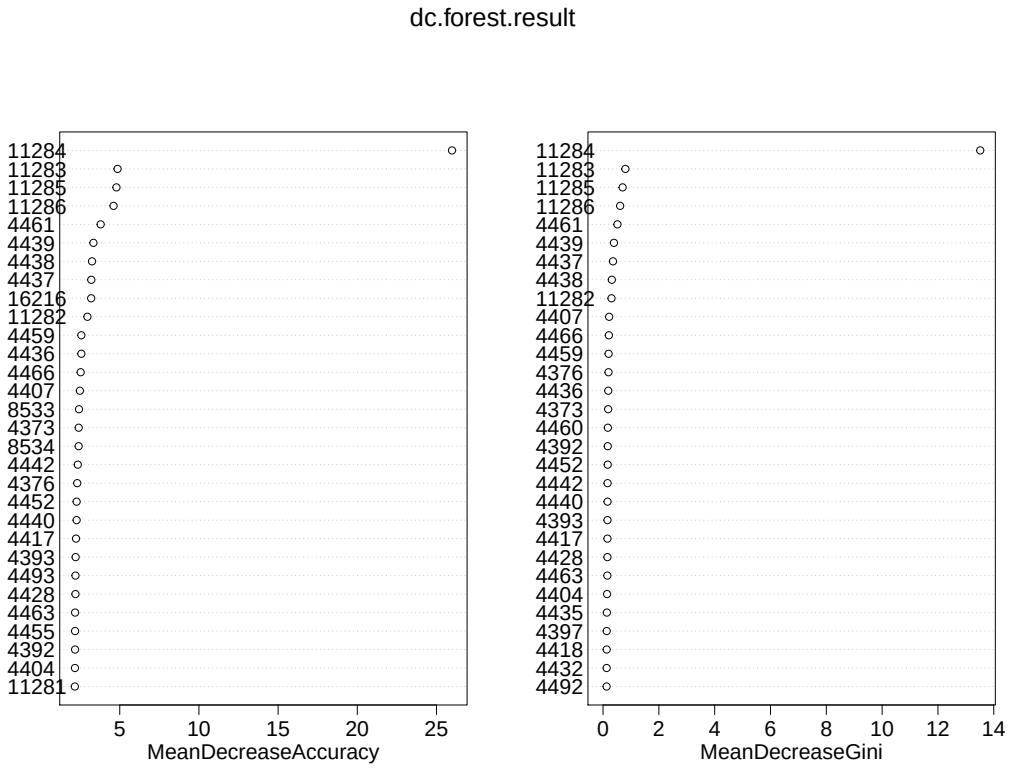


Figure 7.14: Plots of the top 30 variable importance for Random Forests of DNACopy CNA dataset for $nodesize = 1$, $ntree = 1000$ and $mtry = 18388$ with $seed = 1$.

This result indicates that Random Forests outperform Classification Trees. As previously explained for DNACopy CAN dataset, Classification Trees only indicate the variables which correspond with windows within loci 10p11.22 and 10q11.21 as the ones that re responsible to perform the classification task. The fusion between the neighboring gene KIF5B in locus 10p11.22 and gene RET in locus 10q11.21 is common in adeno carcinoma lung cancer. Meanwhile, the

other variables which correspond to windows within loci 3q24 up to 3q27.3 are put aside during the Classification Trees construction. The amplification of genes SOX2 and PIK3CA within these loci is common for squamous carcinoma lung cancer. On contrast, Random Forests are able to identify the relative contribution of these windows while fitting the classification model. The resulting Random Forests show that the windows within loci 3q24 up to 3q27.3 and those within loci 10p11.22 and 10q11.21 are responsible for stratifying a patient into either one of two tumour subtypes, adeno carcionoma and squamous carcinoma.

Moreover, we present the plots of computational time required for fitting Random Forests using DNACopy CNA dataset. We display these plots in Figure 7.15. This figure shows the computational time required to fit Random Forests with seed $= 1$ and $nodesize = 1$ for various $ntree$ and $mtry$ settings. This figure indicates that fitting Random Forests for larger $ntree$ leads to longer computational time. Likewise, fitting the forests for larger $mtry$ also leads to longer computational time. We end up with 36.36575 seconds to fit a Random Forest with $nodesize = 1$, $ntree = 1000$ and $mtry = 18388$. These hyperparameter setting gives Random Forests with the lowest OOB error rate.

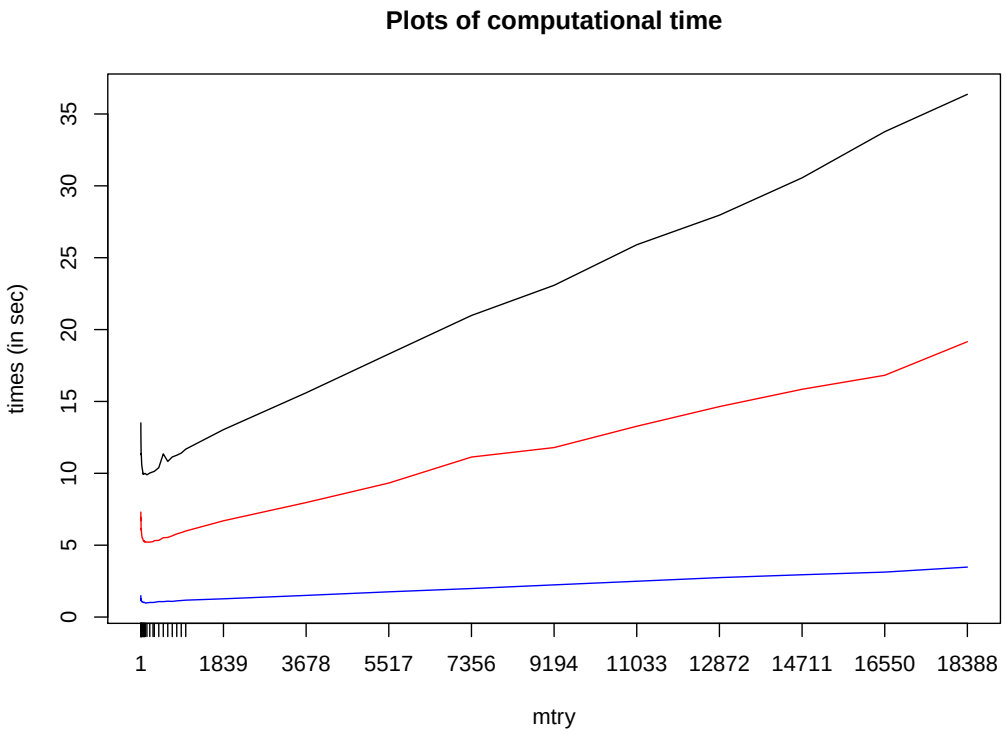


Figure 7.15: Plots of computational time for fitting Random Forests of DNA-Copy CNA dataset. We fit the Random Forests with seed = 1 for $nodesize = 1$ and various $mtry$ settings as shown within Table 7.1. The blue line corresponds with the computational time required for fitting Random Forests with $ntree = 100$. The red line corresponds with the computational time required for fitting Random Forests with $ntree = 500$. The black line corresponds with the computational time required for fitting Random Forests with $ntree = 1000$.

7.4 Conclusion

Random Forests provides us with flexibility in the light of treating different datasets in distinct ways by bootstrapping and subsetting variables while splitting a single node within an individual tree. In this chapter, it can be seen how Random Forests are able to identify the difference in the characteristics of Smooth CNA and DNACopy CNA datasets.

For Smooth CNA dataset, Random Forests indicate that this dataset contains a number of strong relevant predictors along with large number of weak relevant predictors. These relevant predictors are variables located within genomic region of Chromosome 3. Both the strong and the weak relevant predictors have contribution on fitting Random Forests. These relevant predictors represents several windows that consist of loci 3q24 up to 3q27.3. The amplification in genes SOX2 and PIK3CA which are located within these loci is common in Squamous lung cancer. Therefore, we can use genes SOX2 and PIK3CA as the genetic markers.

Meanwhile, for DNACopy dataset, Random Forests indicate the presence of small number of very strong relevant predictors along with the set of strong and weak relevant predictors. This set of small number of very strong relevant predictors consists of few variables from genomic region of Chromosome 10, while the set of either the strong or the weak relevant predictors contains large number of variables from genomic region of Chromosome 3. Owing to their strength, the set of the small number of very strong relevant predictors dominates over either the strong or the weak relevant predictors while fitting Random Forests.

Likewise for Smooth CNA dataset, for DNACopy CNA dataset, the relevant predictors within genomic region of Chromosome 3 represent the windows that consist of loci 3q24 up to 3q27.3. Hence, for this set of windows, we end up with genes SOX2 and PIK3CA as the genetic markers. Moreover, the set of small number of very strong relevant predictors from genomic region of Chromosome 10 are the ones that represent the windows that consist of loci 10p11.22 and 10q11.21. The fusion of gene KIF5B in locus 10p11.22 and gene RET in locus 10q11.21 is common for Adenocarcinoma lung cancer. Hence, genes KIF5B and RET can also be used as the genetic markers.

Therefore, the flexibility that Random Forests provide us enables us to correctly identify the presence of the genetic markers from different genomic regions for DNACopy dataset. Meanwhile, Classification Trees only supply us the genetic markers from genomic region of Chromosome 3. Hence, with this flexibility, using Random Forests, we obtain the more appropriate classification model for each dataset. This in turn leads to the more accurate prediction. Compared to the cross-validated prediction error rates of Classification Trees, we obtain lower OOB error rates by applying Random Forests.

We obtain the results mentioned above by fitting Random Forests using the optimal setting of hyperparameters. These optimal setting is determined prior to the Random Forests construction. For datasets which have much more variables than observations as in genomic datasets, it is no longer correct to use the default setting of tuning parameters. Hence, one has to perform simulation study to seek for this optimal setting of tuning parameters. This is a challenging task since it is time consuming and computationally expensive as well. In addition, unlike Classification Trees, Random Forests does not provide us with the interaction between variables. We did not do simulation study to investigate how the presence of many blocks of highly correlated variables in dataset affect the performance of Random Forests.

Chapter 8

Conclusion and future works

8.1 Summary

In this thesis we have applied Classification Trees and Random Forests as two data classification algorithms for stratifying lung cancer patients who suffered from either adeno carcinoma or squamous carcinoma using their CNA profiles which are generated from NGS datasets.

As genomic datasets, the two CNA datasets had been studied, Smooth CNA and DNACopy CNA datasets, have the following characteristics:

1. having much more variables than observation ($p \gg n$),
2. having blocks of correlated variables that share the common genomic location,
3. having many strong relevant predictor variables along with larger number of weak relevant predictor variables and
4. having blocks of variables of large variances that spread across various genomic locations.

Performing data classification on these two datasets using Classification Trees algorithm as can be found in Chapter 4, we come across three issues. Firstly, fitting Classification Trees using datasets of much more variables than observations, we only make use of the discriminative information contained in small number of

variables which can be seen as strong relevant predictors and left large number of variables including the informative ones. Whereas, some of these variables are important for prediction.

Secondly, having variables which contain the discriminative information with high data variation yield less accurate Classification Trees. Finally, in the light of cross validated estimation for the prediction error, we find that having datasets of small sample size causes the unstable error rate estimates.

To overcome the first issue mentioned above, we make use of PCA as a data dimensionality reduction method. We apply PCA prior to Classification Trees construction to reduce data dimensionality while retain the variation of the data. However, applying PCA does not always improve Classification Trees performance in terms of the prediction error rate.

PCA produces new variables by finding linear combinations of original datasets with maximum variance. Hence, the resulting principal component scores might be more affected by variables of extremely high variances. In addition, PCA is also strongly affected by the presence of many blocks of correlated variables.

We solve the issue due to the presence of several variables with such this extremely variances by standardising the data. However, standardising the data leads to another issue. Vast majority of variables in our datasets have very low variances. Standardising these variables causes the increase of their variances. Therefore using principal component scores of standardised variables also does not improve Classification Trees performance. Transforming the data using logarithmic transformation does not improve the performance of Classification Trees of principal component scores either. In this case, we apply data transformation to reduce the effect of several variables with high variance by making the data more normal. Therefore, it seems that for datasets which have much more variables than observations along with blocks of correlated variables, one should not apply PCA as a data dimensionality reduction method.

Furthermore we apply ICA as a feature extraction method prior to Classification Trees construction. However, applying this feature extraction method does not improve the resulting Classification Trees as well. There are two reasons for this. Firstly, the whitening step applied while estimating the independent components greatly reduce data dimension. Therefore, we end up with nearly

similar results as applying PCA. Secondly, ICA estimates involves maximising non-Gaussianity on whitened variables. This maximisation only deals with non-Gaussianity and might not exploit the discriminative information contained in dataset. These findings also suggest one to not make use of ICA as a data dimensionality reduction method.

Moreover, we apply Random Forests to overcome the issues mentioned above. Random Forests involve subsetting of variables for splitting each node in an individual tree. This in turn enables weak relevant predictors to be selected as the classifying one. Apart from the number of variables to be subset for splitting each node in an individual tree, there are other two tuning parameters: number of trees to be generated and number of observations in each terminal node of an individual trees.

We should search for the optimal setting of these hyperparameters for the forest to produce Random Forest with low error rate. For the number of observations in a terminal node we recommend the use of one observation. For the number of trees, based on our simulation studies, we find that ones should generate at least 500 trees in order to obtain a stable estimate. However, we can not prescribe one suggestion for the number of variables to be subsetted. For our datasets, we find that for Smooth CNA dataset we have to set this hyperparameter small. On contrast, for DNACopy CNA dataset, this tuning parameter should be set to become large for getting Random Forests with small error rate.

With regard to the comparison across the models explained above, we recommend the use of Random Forests rather than Classification Trees. We also do not suggest the application of both PCA and ICA as the data dimension reduction methods. In terms of prediction error, Random Forests give the lowest error rate. In the light of the insight underlying the resulting classification model, Random Forests are able to precisely produce the accurate classifiers. For DNACopy dataset, Random Forests successfully identify the contribution of variables within both Chromosomes 3 and 10. Whereas Classification Trees only recognise the contribution of those of Chromosome 10.

However, in terms of computational time, Random Forests are expensive. Compared to the other models, Random Forests spend the longest time. Table 8.1 presents the time needed to fit a classification model upon Smooth CNA

dataset. Table 8.2 presents the time needed to fit a classification model upon DNACopy CNA dataset.

Table 8.1: Comparison for the computational time of the classification models fitted using Smooth CNA

Model	Time (in sec)
Classification Trees	16.69539
Classification Trees of PCs (the first two PCs)	0.39830
Classification Trees of ICs (the first two ICs)	0.38949
Random Forests (ntree = 1000,seed = 1)	19.69330

Table 8.2: Comparison for the computational time of the classification models fitted using DNACopy CNA

Model	Time (in sec)
Classification Trees	16.83186
Classification Trees of PCs (the first two PCs)	0.51661
Classification Trees of ICs (the first two ICs)	0.45378
Random Forests (ntree = 1000,seed = 1)	36.36575

Furthermore, with regard to the genetic point of view for the resulting Classification Trees and Random Forests for both Smooth CNA and DNACopy CNA datasets, we end up with different results. For Classification Trees of Smooth CNA dataset, we obtain genes SOX2 and PIK3CA as the genetic markers. Meanwhile, for Classification Trees of DNACopy CNA dataset, we attains gene KIF5B and RET as the genetic markers. Moreover, for Random Forests of Smooth CNA datasets, we get the same result as of Classification Trees. Nevertheless, for Random Forests of DNACopy CNA dataset, we end up with genes KIF5B, RET, SOX2 and PIK3CA as the genetic markers.

8.2 Future works

Currently, we have identified several issues from applying Classification Trees on datasets whose much more variables than observations. It is recommended that for the future research, to perform the study on finding an appropriate data dimension reduction prior to fitting Classification Trees.

Moreover, with respect to the application of Random Forests, it is advised to carry out an investigation on how to determine the optimal tuning parameters. It is also suggested to perform the studies on the effect of the presence of blocks of correlated variables in dataset upon the optimal setting of tuning parameters, the prediction error and the variable importance of Random Forests. In addition, it is also interesting to study the interaction between variables within Random Forests.

Appendix A

Significance of Normality Test for Pair Difference of Prediction Error Rate Datasets

A.1 Smooth CNA dataset

Table A.1: P-values from Shapiro-Wilk test of normality test for all pair difference of prediction error rate datasets for Smooth CNA dataset

Method 1	Method 2	p-value	
raw-rpart	raw-cltrees	1.71e-05	*
raw-cltrees	chr3-cltrees	7.00e-07	*
raw-cltrees	mean.proj-cltrees	4.98e-04	*
raw-cltrees	median.proj-cltrees	9.92e-03	*
raw-cltrees	raw-pca-cltrees	2.52e-02	*
raw-cltrees	tr1-pca-cltrees	8.35e-02	
raw-cltrees	tr2-pca-cltrees	8.35e-02	
raw-cltrees	tr3-pca-cltrees	5.99e-03	*
raw-cltrees	raw-ica-cltrees	5.11e-03	*
chr3-cltrees	mean.proj-cltrees	5.34e-04	*
chr3-cltrees	median.proj-cltrees	6.35e-03	*
chr3-cltrees	raw-pca-cltrees	6.42e-02	
chr3-cltrees	tr1-pca-cltrees	2.74e-02	*
chr3-cltrees	tr2-pca-cltrees	2.74e-02	*

Continued on next page

Table A.1 – continued from previous page

Method 1	Method 2	p-value	
chr3-cltrees	tr3-pca-cltrees	9.25e-03	*
chr3-cltrees	raw-ica-cltrees	1.12e-02	*
mean.proj-cltrees	median.proj-cltrees	2.95e-04	*
mean.proj-cltrees	raw-pca-cltrees	1.23e-02	*
mean.proj-cltrees	tr1-pca-cltrees	1.16e-02	*
mean.proj-cltrees	tr2-pca-cltrees	1.16e-02	*
mean.proj-cltrees	tr3-pca-cltrees	1.46e-04	*
mean.proj-cltrees	raw-ica-cltrees	4.63e-02	*
median.proj-cltrees	raw-pca-cltrees	8.59e-03	*
median.proj-cltrees	tr1-pca-cltrees	1.22e-03	*
median.proj-cltrees	tr2-pca-cltrees	1.22e-03	*
median.proj-cltrees	tr3-pca-cltrees	1.02e-03	*
median.proj-cltrees	raw-ica-cltrees	1.49e-01	
raw-pca-cltrees	tr1-pca-cltrees	1.41e-03	*
raw-pca-cltrees	tr2-pca-cltrees	1.41e-03	*
raw-pca-cltrees	tr3-pca-cltrees	2.71e-02	*
raw-pca-cltrees	raw-ica-cltrees	1.07e-01	
tr1-pca-cltrees	tr2-pca-cltrees	NaN	
tr1-pca-cltrees	tr3-pca-cltrees	7.66e-03	*
tr1-pca-cltrees	raw-ica-cltrees	1.72e-02	*
tr2-pca-cltrees	tr3-pca-cltrees	7.66e-03	*
tr2-pca-cltrees	raw-ica-cltrees	1.72e-02	*
tr3-pca-cltrees	raw-ica-cltrees	5.69e-02	

A.2 DNACopy CNA dataset

Table A.2: P-values from Shapiro-Wilk test of normality test for all pair difference of prediction error rate datasets for DNACopy CNA dataset

Method 1	Method 2	p-value	
raw-rpart	raw-cltrees	1.78e-04	*
raw-cltrees	chr3-cltrees	2.38e-04	*
raw-cltrees	chr10-cltrees	2.99e-04	*
raw-cltrees	chr3-chr10-cltrees	1.18e-15	*
raw-cltrees	mean.proj-cltrees	1.21e-02	*
raw-cltrees	median.proj-cltrees	7.67e-03	*
raw-cltrees	raw-pca-cltrees	8.96e-03	*
raw-cltrees	tr1-pca-cltrees	2.79e-02	*
raw-cltrees	tr2-pca-cltrees	3.25e-02	*
raw-cltrees	tr3-pca-cltrees	1.75e-02	*
raw-cltrees	raw-ica-cltrees	5.51e-02	
chr3-cltrees	chr10-cltrees	6.79e-09	*
chr3-cltrees	chr3-chr10-cltrees	3.45e-04	*
chr3-cltrees	mean.proj-cltrees	1.14e-03	*
chr3-cltrees	median.proj-cltrees	1.67e-03	*
chr3-cltrees	raw-pca-cltrees	2.41e-02	*
chr3-cltrees	tr1-pca-cltrees	3.65e-02	*
chr3-cltrees	tr2-pca-cltrees	9.32e-01	
chr3-cltrees	tr3-pca-cltrees	4.34e-02	*
chr3-cltrees	raw-ica-cltrees	3.63e-02	*
chr10-cltrees	chr3-chr10-cltrees	2.66e-04	*
chr10-cltrees	mean.proj-cltrees	3.02e-03	*
chr10-cltrees	median.proj-cltrees	1.65e-03	*
chr10-cltrees	raw-pca-cltrees	2.41e-03	*
chr10-cltrees	tr1-pca-cltrees	1.39e-02	*
chr10-cltrees	tr2-pca-cltrees	5.63e-02	
chr10-cltrees	tr3-pca-cltrees	1.37e-02	*
chr10-cltrees	raw-ica-cltrees	5.65e-02	
chr3-chr10-cltrees	mean.proj-cltrees	4.33e-03	*
chr3-chr10-cltrees	median.proj-cltrees	1.7e-02	*
chr3-chr10-cltrees	raw-pca-cltrees	4.02e-03	*
chr3-chr10-cltrees	tr1-pca-cltrees	1.01e-01	
chr3-chr10-cltrees	tr2-pca-cltrees	9.18e-02	
chr3-chr10-cltrees	tr3-pca-cltrees	6.65e-02	

Continued on next page

Table A.2 – continued from previous page

Method 1	Method 2	p-value	
chr3-chr10-cltrees	raw-ica-cltrees	4.79e-02	*
mean.proj-cltrees	median.proj-cltrees	2.84e-02	*
mean.proj-cltrees	raw-pca-cltrees	6.7e-03	*
mean.proj-cltrees	tr1-pca-cltrees	1.43e-01	
mean.proj-cltrees	tr2-pca-cltrees	9.36e-02	
mean.proj-cltrees	tr3-pca-cltrees	9.48e-02	
mean.proj-cltrees	raw-ica-cltrees	1.39e-01	
median.proj-cltrees	raw-pca-cltrees	5.07e-02	
median.proj-cltrees	tr1-pca-cltrees	2.42e-01	
median.proj-cltrees	tr2-pca-cltrees	2.63e-01	
median.proj-cltrees	tr3-pca-cltrees	2.27e-01	
median.proj-cltrees	raw-ica-cltrees	6.65e-03	*
raw-pca-cltrees	tr1-pca-cltrees	2.19e-01	
raw-pca-cltrees	tr2-pca-cltrees	2.42e-01	
raw-pca-cltrees	tr3-pca-cltrees	2.08e-01	
raw-pca-cltrees	raw-ica-cltrees	5.06e-05	*
tr1-pca-cltrees	tr2-pca-cltrees	8.29e-17	*
tr1-pca-cltrees	tr3-pca-cltrees	1.14e-19	*
tr1-pca-cltrees	raw-ica-cltrees	2.02e-01	
tr2-pca-cltrees	tr3-pca-cltrees	8.72e-16	*
tr2-pca-cltrees	raw-ica-cltrees	1.91e-01	
tr3-pca-cltrees	raw-ica-cltrees	1.94e-01	

Appendix B

Significance of Two-sided Wilcoxon Test for Pair Difference of Prediction Error Rate Datasets

B.1 Smooth CNA dataset

Table B.1: P-values from Two-sided Wilcoxon Test for all pair difference of prediction error rate datasets for Smooth CNA dataset

Method 1	Method 2	p-value	
raw-rpart	raw-cltrees	2.27e-03	*
raw-cltrees	chr3-cltrees	6.44e-01	
raw-cltrees	mean.proj-cltrees	9.79e-04	*
raw-cltrees	median.proj-cltrees	9.63e-10	*
raw-cltrees	raw-pca-cltrees	2.64e-11	*
raw-cltrees	tr1-pca-cltrees	1.01e-17	*
raw-cltrees	tr2-pca-cltrees	1.01e-17	*
raw-cltrees	tr3-pca-cltrees	1.55e-15	*
raw-cltrees	raw-ica-cltrees	5.14e-14	*
chr3-cltrees	mean.proj-cltrees	3.46e-03	*
chr3-cltrees	median.proj-cltrees	3.07e-09	*
chr3-cltrees	raw-pca-cltrees	9.05e-11	*

Continued on next page

Table B.1 – continued from previous page

Method 1	Method 2	p-value	
chr3-cltrees	tr1-pca-cltrees	1.84e-17	*
chr3-cltrees	tr2-pca-cltrees	1.84e-17	*
chr3-cltrees	tr3-pca-cltrees	1.01e-13	*
chr3-cltrees	raw-ica-cltrees	3.91e-13	*
mean.proj-cltrees	median.proj-cltrees	2.11e-16	*
mean.proj-cltrees	raw-pca-cltrees	1.2e-08	*
mean.proj-cltrees	tr1-pca-cltrees	6.06e-18	*
mean.proj-cltrees	tr2-pca-cltrees	6.06e-18	*
mean.proj-cltrees	tr3-pca-cltrees	2e-13	*
mean.proj-cltrees	raw-ica-cltrees	1.03e-12	*
median.proj-cltrees	raw-pca-cltrees	7.63e-17	*
median.proj-cltrees	tr1-pca-cltrees	3.52e-18	*
median.proj-cltrees	tr2-pca-cltrees	3.52e-18	*
median.proj-cltrees	tr3-pca-cltrees	7.56e-18	*
median.proj-cltrees	raw-ica-cltrees	2.99e-17	*
raw-pca-cltrees	tr1-pca-cltrees	1.4e-15	*
raw-pca-cltrees	tr2-pca-cltrees	1.4e-15	*
raw-pca-cltrees	tr3-pca-cltrees	8.46e-02	
raw-pca-cltrees	raw-ica-cltrees	6.07e-04	*
tr1-pca-cltrees	tr2-pca-cltrees	NaN	
tr1-pca-cltrees	tr3-pca-cltrees	7.73e-14	*
tr1-pca-cltrees	raw-ica-cltrees	1.99e-07	*
tr2-pca-cltrees	tr3-pca-cltrees	7.73e-14	*
tr2-pca-cltrees	raw-ica-cltrees	1.99e-07	*
tr3-pca-cltrees	raw-ica-cltrees	3.18e-02	*

B.2 DNACopy CNA dataset

Table B.2: P-values from Two-sided Wilcoxon Test for all pair difference of prediction error rate datasets for DNACopy CNA dataset

Method 1	Method 2	p-value	
raw-rpart	raw-cltrees	4.53e-10	*
raw-cltrees	chr3-cltrees	6.2e-15	*
raw-cltrees	chr10-cltrees	2.88e-18	*
raw-cltrees	chr3-chr10-cltrees	2.91e-01	
raw-cltrees	mean.proj-cltrees	1.84e-11	*
raw-cltrees	median.proj-cltrees	1.98e-09	*
raw-cltrees	raw-pca-cltrees	2.2e-08	*
raw-cltrees	tr1-pca-cltrees	6.82e-15	*
raw-cltrees	tr2-pca-cltrees	1.53e-14	*
raw-cltrees	tr3-pca-cltrees	6.67e-15	*
raw-cltrees	raw-ica-cltrees	4.14e-13	*
chr3-cltrees	chr10-cltrees	8.64e-19	*
chr3-cltrees	chr3-chr10-cltrees	1.1e-14	*
chr3-cltrees	mean.proj-cltrees	7.638e-01	
chr3-cltrees	median.proj-cltrees	2.71e-01	*
chr3-cltrees	raw-pca-cltrees	5.77e-17	*
chr3-cltrees	tr1-pca-cltrees	6.22e-18	*
chr3-cltrees	tr2-pca-cltrees	8.46e-18	*
chr3-cltrees	tr3-pca-cltrees	6.52e-18	*
chr3-cltrees	raw-ica-cltrees	8.45e-18	*
chr10-cltrees	chr3-chr10-cltrees	2.97e-18	*
chr10-cltrees	mean.proj-cltrees	2.4e-17	*
chr10-cltrees	median.proj-cltrees	5.86e-18	*
chr10-cltrees	raw-pca-cltrees	3.4e-18	*
chr10-cltrees	tr1-pca-cltrees	3.33e-18	*
chr10-cltrees	tr2-pca-cltrees	3.4e-18	*
chr10-cltrees	tr3-pca-cltrees	3.35e-18	*
chr10-cltrees	raw-ica-cltrees	3.49e-18	*
chr3-chr10-cltrees	mean.proj-cltrees	1.04e-11	*
chr3-chr10-cltrees	median.proj-cltrees	5.29e-09	*
chr3-chr10-cltrees	raw-pca-cltrees	3.42e-09	*
chr3-chr10-cltrees	tr1-pca-cltrees	4.31e-15	*
chr3-chr10-cltrees	tr2-pca-cltrees	8.62e-15	*
chr3-chr10-cltrees	tr3-pca-cltrees	3.98e-15	*

Continued on next page

Table B.2 – continued from previous page

Method 1	Method 2	p-value	
chr3-chr10-cltrees	raw-ica-cltrees	1.36e-13	*
mean.proj-cltrees	median.proj-cltrees	4.48e-2	*
mean.proj-cltrees	raw-pca-cltrees	3.75e-16	*
mean.proj-cltrees	tr1-pca-cltrees	2.09e-17	*
mean.proj-cltrees	tr2-pca-cltrees	4.24e-17	*
mean.proj-cltrees	tr3-pca-cltrees	2.98e-17	*
mean.proj-cltrees	raw-ica-cltrees	2.42e-16	*
median.proj-cltrees	raw-pca-cltrees	2.75e-15	*
median.proj-cltrees	tr1-pca-cltrees	1.86e-17	*
median.proj-cltrees	tr2-pca-cltrees	1.99e-17	*
median.proj-cltrees	tr3-pca-cltrees	1.83e-17	*
median.proj-cltrees	raw-ica-cltrees	5.73e-17	*
raw-pca-cltrees	tr1-pca-cltrees	1.52e-05	*
raw-pca-cltrees	tr2-pca-cltrees	2.71e-05	
raw-pca-cltrees	tr3-pca-cltrees	1.38e-05	
raw-pca-cltrees	raw-ica-cltrees	2.02e-04	
tr1-pca-cltrees	tr2-pca-cltrees	8.25e-01	
tr1-pca-cltrees	tr3-pca-cltrees	8.33e-01	
tr1-pca-cltrees	raw-ica-cltrees	1.65e-01	
tr2-pca-cltrees	tr3-pca-cltrees	8.48e-01	
tr2-pca-cltrees	raw-ica-cltrees	1.68e-01	
tr3-pca-cltrees	raw-ica-cltrees	1.41e-01	

Appendix C

Significance of One-sided Wilcoxon Test for Pair Difference of Prediction Error Rate Datasets

C.1 Smooth CNA dataset

Table C.1: P-values from One-sided Wilcoxon Test for all pair difference of prediction error rate datasets for Smooth CNA dataset

Method 1	Method 2	p-value	
raw-rpart	raw-cltrees	1.14e-03	*
raw-cltrees	chr3-cltrees	3.22e-01	
raw-cltrees	mean.proj-cltrees	1.00e00	
raw-cltrees	median.proj-cltrees	4.82e-10	*
raw-cltrees	raw-pca-cltrees	1.00e00	
raw-cltrees	tr1-pca-cltrees	1.00e00	
raw-cltrees	tr2-pca-cltrees	1.00e00	
raw-cltrees	tr3-pca-cltrees	1.00e00	
raw-cltrees	raw-ica-cltrees	1.00e00	
chr3-cltrees	mean.proj-cltrees	9.98e-01	
chr3-cltrees	median.proj-cltrees	1.53e-09	*
chr3-cltrees	raw-pca-cltrees	1.00e00	

Continued on next page

Table C.1 – continued from previous page

Method 1	Method 2	p-value	
chr3-cltrees	tr1-pca-cltrees	1.00e00	
chr3-cltrees	tr2-pca-cltrees	1.00e00	
chr3-cltrees	tr3-pca-cltrees	1.00e00	
chr3-cltrees	raw-ica-cltrees	1.00e00	
mean.proj-cltrees	median.proj-cltrees	1.06e-16	*
mean.proj-cltrees	raw-pca-cltrees	1.00e00	
mean.proj-cltrees	tr1-pca-cltrees	1.00e00	
mean.proj-cltrees	tr2-pca-cltrees	1.00e00	
mean.proj-cltrees	tr3-pca-cltrees	1.00e00	
mean.proj-cltrees	raw-ica-cltrees	1.00e00	
median.proj-cltrees	raw-pca-cltrees	1.00e00	
median.proj-cltrees	tr1-pca-cltrees	1.00e00	
median.proj-cltrees	tr2-pca-cltrees	1.00e00	
median.proj-cltrees	tr3-pca-cltrees	1.00e00	
median.proj-cltrees	raw-ica-cltrees	1.00e00	
raw-pca-cltrees	tr1-pca-cltrees	1.00e00	
raw-pca-cltrees	tr2-pca-cltrees	1.00e00	
raw-pca-cltrees	tr3-pca-cltrees	9.58e-01	
raw-pca-cltrees	raw-ica-cltrees	1.00e00	
tr1-pca-cltrees	tr2-pca-cltrees	1.00e00	
tr1-pca-cltrees	tr3-pca-cltrees	3.86e-14	*
tr1-pca-cltrees	raw-ica-cltrees	9.97e-08	*
tr2-pca-cltrees	tr3-pca-cltrees	3.86e-14	*
tr2-pca-cltrees	raw-ica-cltrees	9.97e-08	*
tr3-pca-cltrees	raw-ica-cltrees	9.84e-01	

C.2 DNACopy CNA dataset

Table C.2: P-values from One-sided Wilcoxon Test for all pair difference of prediction error rate datasets for DNACopy CNA dataset

Method 1	Method 2	p-value	
raw-rpart	raw-cltrees	1.00e00	
raw-cltrees	chr3-cltrees	3.1e-15	*
raw-cltrees	chr10-cltrees	1.44e-18	*
raw-cltrees	chr3-chr10-cltrees	1.45e-01	
raw-cltrees	mean.proj-cltrees	9.22e-12	*
raw-cltrees	median.proj-cltrees	9.9e-10	*
raw-cltrees	raw-pca-cltrees	1.00e00	
raw-cltrees	tr1-pca-cltrees	1.00e00	
raw-cltrees	tr2-pca-cltrees	1.00e00	
raw-cltrees	tr3-pca-cltrees	1.00e00	
raw-cltrees	raw-ica-cltrees	1.00e00	
chr3-cltrees	chr10-cltrees	4.32e-19	*
chr3-cltrees	chr3-chr10-cltrees	1.00e00	
chr3-cltrees	mean.proj-cltrees	3.81e-01	
chr3-cltrees	median.proj-cltrees	9.87e-01	
chr3-cltrees	raw-pca-cltrees	1.00e00	
chr3-cltrees	tr1-pca-cltrees	1.00e00	
chr3-cltrees	tr2-pca-cltrees	1.00e00	
chr3-cltrees	tr3-pca-cltrees	1.00e00	
chr3-cltrees	raw-ica-cltrees	1.00e00	
chr10-cltrees	chr3-chr10-cltrees	1.00e00	
chr10-cltrees	mean.proj-cltrees	1.00e00	
chr10-cltrees	median.proj-cltrees	1.00e00	
chr10-cltrees	raw-pca-cltrees	1.00e00	
chr10-cltrees	tr1-pca-cltrees	1.00e00	
chr10-cltrees	tr2-pca-cltrees	1.00e00	
chr10-cltrees	tr3-pca-cltrees	1.00e00	
chr10-cltrees	raw-ica-cltrees	1.00e00	
chr3-chr10-cltrees	mean.proj-cltrees	5.2e-12	*
chr3-chr10-cltrees	median.proj-cltrees	2.64e-09	*
chr3-chr10-cltrees	raw-pca-cltrees	1.00e00	
chr3-chr10-cltrees	tr1-pca-cltrees	1.00e00	
chr3-chr10-cltrees	tr2-pca-cltrees	1.00e00	
chr3-chr10-cltrees	tr3-pca-cltrees	1.00e00	

Continued on next page

Table C.2 – continued from previous page

Method 1	Method 2	p-value	
chr3-chr10-cltrees	raw-ica-cltrees	1.00e00	
mean.proj-cltrees	median.proj-cltrees	9.78e-01	
mean.proj-cltrees	raw-pca-cltrees	1.00e00	
mean.proj-cltrees	tr1-pca-cltrees	1.00e00	
mean.proj-cltrees	tr2-pca-cltrees	1.00e00	
mean.proj-cltrees	tr3-pca-cltrees	1.00e00	
mean.proj-cltrees	raw-ica-cltrees	1.00e00	
median.proj-cltrees	raw-pca-cltrees	1.00e00	
median.proj-cltrees	tr1-pca-cltrees	1.00e00	
median.proj-cltrees	tr2-pca-cltrees	1.00e00	
median.proj-cltrees	tr3-pca-cltrees	1.00e00	
median.proj-cltrees	raw-ica-cltrees	1.00e00	
raw-pca-cltrees	tr1-pca-cltrees	1.00e00	
raw-pca-cltrees	tr2-pca-cltrees	1.00e00	
raw-pca-cltrees	tr3-pca-cltrees	1.00e00	
raw-pca-cltrees	raw-ica-cltrees	1.00e00	
tr1-pca-cltrees	tr2-pca-cltrees	6.12e-01	
tr1-pca-cltrees	tr3-pca-cltrees	6.63e-01	
tr1-pca-cltrees	raw-ica-cltrees	8.24-02	
tr2-pca-cltrees	tr3-pca-cltrees	4.24e-01	
tr2-pca-cltrees	raw-ica-cltrees	8.42e-02	*
tr3-pca-cltrees	raw-ica-cltrees	7.04e-02	

References

Allory, Y., Bazille, C., Vieillefond, A., Molinié, V., Cochand-Priollet, B., Cussenot, O., Callard, P. & Sibony, M. (2008). Profiling and classification tree applied to renal epithelial tumours. *Histopathology*, **52**, 158–166. 2

Bernard, S., Heutte, L. & Adam, S. (2009). Influence of hyperparameters on random forest accuracy. In *International Workshop on Multiple Classifier Systems*, 171–180, Springer. 243, 244

Bernard, S., Heutte, L. & Adam, S. (2010). A study of strength and correlation in random forests. In *International Conference on Intelligent Computing*, 186–191, Springer. 241

Botta, V., Louppe, G., Geurts, P. & Wehenkel, L. (2014). Exploiting snp correlations within random forest for genome-wide association studies. *PloS one*, **9**. 3

Boulesteix, A.L., Janitza, S., Kruppa, J. & König, I.R. (2012). Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **2**, 493–507. 244, 245

Braga-Neto, U.M. & Dougherty, E.R. (2004). Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, **20**, 374–380. 49, 53

Breiman, L. (2001a). Random forests. *Machine learning*, **45**, 5–32. 7, 236, 237, 241

Breiman, L. (2001b). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, **16**, 199–231. 2

Breiman, L. (2002). Manual-setting up, using, and understanding random forests v3. 1, 29 p. 239, 241, 242, 246

Breiman, L. & Spector, P. (1992). Submodel selection and evaluation in regression. the x-random case. *International statistical review/revue internationale de Statistique*, 291–319. 52

Breiman, L. *et al.* (1984). *Classification and Regression Trees*. Chapman & Hall. 5, 35, 40, 50, 237, 238

Brunelli, M., Bria, E., Nottegar, A., Cingarlini, S., Simionato, F., Caliò, A., Eccher, A., Parolini, C., Iannucci, A., Gilioli, E. *et al.* (2012). True 3q chromosomal amplification in squamous cell lung carcinoma by fish and acgh molecular analysis: impact on targeted drugs. *PLoS One*, **7**, e49689. 63

Burman, P. (1990). Estimation of optimal transformations using v-fold cross validation and repeated learning-testing methods. *Sankhyā: The Indian Journal of Statistics, Series A*, 314–345. 51

Bylander, T. (2002). Estimating generalization error on two-class datasets using out-of-bag estimates. *Machine learning*, **48**, 287–297. 239

Chang, W.C. (1983). On using principal components before separating a mixture of two multivariate normal distributions. *Applied Statistics*, 267–275. 132

Chen, X. & Ishwaran, H. (2012). Random forests for genomic data analysis. *Genomics*, **99**, 323–329. 3

Cutler, A., Cutler, D.R. & Stevens, J.R. (2012). Random forests. In *Ensemble machine learning*, 157–175, Springer. 239

Díaz-Uriarte, R. & De Andres, S.A. (2006). Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, **7**, 3. 2, 242, 243, 244

Efron, B. & Tibshirani, R. (1997). Improvements on cross-validation: the 632+ bootstrap method. *Journal of the American Statistical Association*, **92**, 548–560. 53

Genuer, R., Poggi, J.M. & Tuleau, C. (2008). Random forests: some methodological insights. *arXiv preprint arXiv:0811.3619*. 243

Goldstein, B.A., Hubbard, A.E., Cutler, A. & Barcellos, L.F. (2010). An application of random forests to a genome-wide association dataset: methodological considerations & new findings. *BMC genetics*, **11**, 49. 2, 243, 244

Gusnanto, A., Wood, H.M., Pawitan, Y., Rabbitts, P. & Berri, S. (2012). Correcting for cancer genome size and tumour cell content enables better estimation of copy number alterations from next-generation sequence data. *Bioinformatics*, **28**, 40–47. 10

Gusnanto, A., Taylor, C.C., Nafisah, I., Wood, H.M., Rabbitts, P. & Berri, S. (2014). Estimating optimal window size for analysis of low-coverage next-generation sequence data. *Bioinformatics*, **30**, 1823–1829. 9

Gusnanto, A., Tcherveniakov, P., Shuweihdi, F., Samman, M., Rabbitts, P. & Wood, H.M. (2015). Stratifying tumour subtypes based on copy number alteration profiles using next-generation sequence data. *Bioinformatics*, **31**, 2713–2720. 2, 10

Huang, J., Gusnanto, A., O'Sullivan, K., Staaf, J., Borg, Å. & Pawitan, Y. (2007). Robust smooth segmentation approach for array cgh data analysis. *Bioinformatics*, **23**, 2463–2469. 1, 8, 9

Hyvärinen, A. & Oja, E. (1997). A fast fixed-point algorithm for independent component analysis. *Neural computation*, **9**, 1483–1492. 6, 182, 183, 227

Hyvärinen, A., Karhunen, J. & Oja, E. (2001). *Independent Component Analysis*. Wiley. 183, 184, 187

Johnson, R.A. & Wichern, D.W. (2004). Multivariate analysis. *Encyclopedia of Statistical Sciences*, **8**. 104

Jolliffe, I. (2002). *Principal Component Analysis*. Springer, 2nd edn. 104, 132

Ju, Y.S., Lee, W.C., Shin, J.Y., Lee, S., Bleazard, T., Won, J.K., Kim, Y.T., Kim, J.I., Kang, J.H. & Seo, J.S. (2012). A transforming kif5b and ret gene fusion in lung adenocarcinoma revealed from whole-genome and transcriptome sequencing. *Genome research*, **22**, 436–445. 70

Koch, I. (2013). *Analysis of multivariate and high-dimensional data*, vol. 32. Cambridge University Press. 104

Kohavi, R. *et al.* (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, vol. 14, 1137–1145, Montreal, Canada. 52, 53

Kohno, T., Ichikawa, H., Totoki, Y., Yasuda, K., Hiramoto, M., Nammo, T., Sakamoto, H., Tsuta, K., Furuta, K., Shimada, Y. *et al.* (2012). Kif5b-ret fusions in lung adenocarcinoma. *Nature medicine*, **18**, 375–377. 70

Liaw, A., Wiener, M. *et al.* (2002). Classification and regression by randomforest. *R news*, **2**, 18–22. 7, 237, 239, 241, 242

Maier, S., Wilbertz, T., Braun, M., Scheble, V., Reischl, M., Mikut, R., Menon, R., Nikolov, P., Petersen, K., Beschorner, C. *et al.* (2011). Sox2 amplification is a common event in squamous cell carcinomas of different organ sites. *Human pathology*, **42**, 1078–1088. 63

Marchini, J., Heaton, C., Ripley, M.B. & Suggests, M. (2019). Package fastica. 6, 182, 189, 190

Mardia, K., Kent, J. & Bibby, J. (1979). *Multivariate analysis*. Acad. Press. 104, 107

McGowan, M., Hoven, A.S., Lund-Iversen, M., Solberg, S., Helland, Å., Hirsch, F.R. & Brustugun, O.T. (2017). Pik3ca mutations as prognostic factor in squamous cell lung carcinoma. *Lung Cancer*, **103**, 52–57. 63

Miettinen, J., Nordhausen, K., Oja, H., Taskinen, S. & Virta, J. (2017). The squared symmetric fastica estimator. *Signal Processing*, **131**, 402–411. 182

Muller, R. & Möckel, M. (2008). Logistic regression and cart in the analysis of multimarker studies. *Clinicachimicaacta*, **394**, 1–6. 2

Nordhausen, K. & Oja, H. (2018). Independent component analysis: A statistical perspective. *Wiley Interdisciplinary Reviews: Computational Statistics*, **10**, e1440. 182

Nordhausen, K., Ilmonen, P., Mandal, A., Oja, H. & Ollila, E. (2011). Deflation-based fastica reloaded. In *2011 19th European Signal Processing Conference*, 1854–1858, IEEE. 182

Ollila, E. (2009). The deflation-based fastica estimator: Statistical analysis revisited. *IEEE transactions on Signal Processing*, **58**, 1527–1541. 182

Olshen, A.B., Venkatraman, E., Lucito, R. & Wigler, M. (2004). Circular binary segmentation for the analysis of array-based dna copy number data. *Biostatistics*, **5**, 557–572. 2, 8, 9

Popper, H.H., Ryska, A., Tímár, J. & Olszewski, W. (2014). Molecular testing in lung cancer in the era of precision medicine. *Translational lung cancer research*, **3**, 291. 70

Probst, P., Wright, M.N. & Boulesteix, A.L. (2019). Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **9**, e1301. 237, 245

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. 35

Schwarzer, G., Nagata, T., Mattern, D., Schmelzeisen, R. & Schumacher, M. (2003). Comparison of fuzzy inference, logistic regression, and classification trees (cart). *Methods of information in medicine*, **42**, 572–577. 2

Statnikov, A., Wang, L. & Aliferis, C.F. (2008). A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC bioinformatics*, **9**, 319. 2

Therneau, T., Atkinson, B. & Ripley, B. (2017). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-11. 35

Witten, I.H., Frank, E., Hall, M.A. & Pal, C.J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann. 51, 52, 53

Wood, H.M., Belvedere, O., Conway, C., Daly, C., Chalkley, R., Bickerdike, M., McKinley, C., Egan, P., Ross, L., Hayward, B. *et al.* (2010). Using next-generation sequencing for high resolution multiplex analysis of copy number variation from nanogram quantities of dna from formalin-fixed paraffin-embedded specimens. *Nucleic acids research*, **38**, e151–e151. 9

Yokota, K., Sasaki, H., Okuda, K., Shimizu, S., Shitara, M., Hikosaka, Y., Moriyama, S., Yano, M. & Fujii, Y. (2012). Kif5b/ret fusion gene in surgically-treated adenocarcinoma of the lung. *Oncology reports*, **28**, 1187–1192. 70

Yoo, W., Ference, B.A., Cote, M.L. & Schwartz, A. (2012). A comparison of logistic regression, logic regression, classification tree, and random forests to identify effective gene-gene and gene-environmental interactions. *International journal of applied science and technology*, **2**, 268. 2

Zhang, H. & Singer, B. (2010). *Recursive Partitioning and Applications*. Springer. 35

Zhang, H., Yu, C.Y., Singer, B. & Xiong, M. (2001). Recursive partitioning for tumor classification with gene expression microarray data. *Proceedings of the National Academy of Sciences*, **98**, 6730–6735. 2

Zhang, H., Yu, C.Y. & Singer, B. (2003). Cell and tumor classification using gene expression data: construction of forests. *Proceedings of the National Academy of Sciences*, **100**, 4168–4172. 2