Prediction of Panel and Streaming Data using Wavelet Transform-based Decision Trees

Xin Zhao

Department of Statistics

University of Leeds

A thesis submitted for the degree of

Doctor of Philosophy

12th September 2018

Declaration

The candidate confirms that the work submitted is her own, except for work which has formed part of jointly authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Chapter 3 was published under Xin Zhao, Stuart Barber, Charles C Taylor, Zoka Milan, "Classification tree methods for panel data using wavelet-transformed time series", Computational Statistics and Data Analysis, 2018, 127(11), 204-216. The motivation for using the wavelet transform was given by Stuart Barber. Classification Method 2 was inspired by Charles C Taylor and developed by Xin Zhao. The liver transplant data set in the application was provided by Zoka Milan (Milan *et al.*, 2016). Xin Zhao developed: (1) the data cleaning process including initial filter and secondary filter; (2) classification methods including Methods 1, and (3) provided the idea of using decision trees.

Chapter 5 was written as a paper: "Interval Forecasts based on Regression Trees for Streaming Data" and has been submitted to a journal for possible publication. The motivation of using wavelet transforms and decision trees for the liver transplant data set is the same as in Chapter 3. Interval forecasts were inspired by Charles C Taylor and Stuart Barber. Xin Zhao contributed to the following aspects: (1) constructing the credible intervals using quantiles; (2) developing the model updating and retraining methods, (3) developing the single and ensemble models.

Acknowledgements

I would like to express my deep gratitude to my supervisors, Dr. Stuart Barber, Prof. Charles C Taylor, who have been excellent mentors. It would have been impossible to finish this project without their constant advice, patience and encouragement. Their valuable advice inspires me a lot for now and the future. I also thank Zoka Milan for providing the motivating data. Without this data, I would not have been able to do the real data application. I am also thankful to the China Scholarship Council (CSC)-University of Leeds joint scholarship for sponsoring my living costs and tuition fees. And it's my honour to have support from my colleagues at the Department of Statistics, and the Leeds Institute for Data Analytics, University of Leeds.

Finally, I am more than grateful to my parents who encourage me in my daily life as well as all my lovely friends who help me and support me these years.

Abstract

Decision trees are a popular model for classification and regression since they have an easy interpretation and no parameter assumptions. In the tree building process, we choose the Gini index as the splitting criterion which has good performance for data with many missing values and many categories (values). Other splitting criteria in use include averaged squared error and statistical significant testing. In the tree pruning process, we use cross validation to choose the best tree which has the minimum possible prediction error.

When the explanatory variables are time series, however, trees can not detect the potential correlation in them and may be influenced by the noise involved. So we use wavelet analysis to transform the original time series into wavelet transformed variables, by decomposing the original time series into scaling and wavelet coefficients, representing the smooth and detail information at different resolution levels. The basis we choose is the Haar wavelet, as it is simple for interpretation. Other bases are also considered, but they do not have obviously better performance than the Haar wavelet. Although the approach of using the wavelet transform is suitable for data without too many variables to control the computational time, the computational time increases due to using high dimensional wavelet transformed variables is roughly only linear in the increase in the number of variables. So the computational time will not increase rapidly when the data are transformed into suitable resolution levels or when the number of original variables is not a lot.

The first application of decision trees with wavelet transformed variables is panel data classification. Trees can classify each observation, but are not able to classify each individual which contains many observations. So we design three methods for panel data classification.

After classify each observation using trees, Method 1 classifies each individual by summarizing the major class of its observations. Method 3 transfers the panel data into cross sectional data by summarizing the information for each individual and then uses trees to classify this cross sectional data. Method 2 is based on Method 1 and is similar to but more complicated than Method 3. The difference between Method 2 and Method 3 is that the transformed cross-sectional data are no longer heart rate values or wavelet transformed heart rate values but the probabilities for each observation to be classified as group 1. The probability is calculated from Method 1. So we number this method as the second one as it is based on Method 1. Results show Method 3 is generally the best on both simulated and real data as it works directly on individuals while Methods 1 and 2 are based on classification results of observations, which is not our primary target.

The second and the third applications are time series prediction. In the second one, we explore, for static regression, whether or not wavelet transformed variables are better than original variables in regression problems under different circumstances. This includes different seasonal effects at a possible time lag of explanatory variables. The models are then applied to real liver transplantation (LT) surgery data and China air pollution data, both of which show that the wavelet transformed variables are better. Wavelet transformed variables are directly used in the third application: interval forecasting for streaming data. In the forecasting process, if both the predicted value and its prediction interval are known, we will know more about the uncertainty in the prediction. There are two choices for interval construction. Gaussian prediction intervals work well if the time series clearly follows a Gaussian distribution. The quantile interval is not restricted by the Gaussian distribution assumptions, which is suitable in this context as we do not know the distribution of the future data. The performance is measured by coverage and interval width. Instead of using only one model, ensemble models are also considered. By comparing trees pro-

duced using typical models like ARIMA and GARCH in both simulation and real data applications, we find trees are more computationally efficient than both alternative models. Compared with trees, ARIMA may have a much wider prediction interval when trend is falsely detected and is slow to react when the distribution changes. GARCH has similar performance to trees in coverage and interval width. So tree methods are suggested for time series prediction.

When comparing the performance of wavelet transformed variables and original variables in both classification and regression simulation and real data applications, results show that wavelet transformed variables are better than or equal to the performance of original variables in accuracy. Models using wavelet transformed variables also provide more detailed information, which give better understanding of the classification or regression process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The research focus is prediction: newly designed methods are applied to panel data (longitudinal data) and time series data. Prediction can be framed as a supervised learning problem, in which training data is used for model construction, and test data is used for measuring model performance. The new methods are based on existing techniques but give ways to apply them to new data structures that have not been analysed by these models before. The techniques in use are decision trees (Breiman *et al.*, 1984b; Hothorn *et al.*, 2006a,b) and the wavelet transform (Donoho & Johnstone, 1994; Percival & Walden, 2000).

1.1 Introduction to decision trees

Decision trees are a decision support tool that use a tree-like graph or model of decisions either for classification or regression. Both classification trees and regression trees can be seen as supervised learning models, the former one maps the input space into predefined classes while the latter one maps the input space into a real-valued domain. As an important part of data mining, decision trees are a discovery and prediction-oriented, supervised inductive learning method, in which the trained model is assumed to be applicable to future, unseen, examples. The meaning of classification not only includes identifying which group a new observation belongs to, on the basis of training dataset, but also includes learning how this new observation is identified by detecting the variables' difference between groups. In most cases, both identifying and learning are important, but sometimes,

learning is more important when the class has already been provided. Similarly, for regression trees, the aim is to predict the new observation's response variable value and understand how it is determined.

To some extent, regression trees and classification trees are similar. For regression problems, the numerical response variable with higher values can be regarded as group one and those with lower values can be seen as group two in classification problems. In this context, the later discussion will mainly concentrate on classification trees, which can be easily applied to regression problems.

So why do we use decision trees instead of other methods? For learning different variables' behaviour between different groups, many traditional methods test variables' values to determine whether they differ significantly or not across different groups, typically using means and variances. Subtle trends, however, may not be detected. So more complex statistical models, like logistic regression (Cox, 1958; Walker & Duncan, 1967), can be built to explore the information involved in the data, but usually require many assumptions to make parameter estimation possible. For example, logistic regression requires the observations to be independent of each other and for there to be little or no multicollinearity among the independent variables. If the assumptions are not valid, solutions obtained from these methods are not reliable. In practice, some variables are correlated with each other. These are typically against the assumptions required and will inevitably lead to unreliable results.

Since parametric methods like logistic regression have such disadvantages, nonparametric methods are suggested which can help explore nonlinear relationships between variables without needing such assumptions. There are many popular nonparametric methods like neural networks (Funahashi, 1989; Specht, 1990), support vector machine (Cauwenberghs & Poggio, 2001; Suykens & Vandewalle, 1999), and decision trees (Breiman *et al.*, 1984b; Hothorn *et al.*, 2006a,b). In terms of comprehensibility, decision trees tend to be better than "black-box" models in interpreting data structure and helping researchers understand the information involved. These advantages undoubtedly bring convenience to decision making in medicine (Abdar *et al.*, 2015; Goodman *et al.*, 2016), commerce (Sun *et al.*, 2016; Zhang *et al.*, 2014), and elsewhere. Classification and regression trees (CART) proposed by Loh (2011) is one type of decision trees. This model splits the original

dataset recursively using the Gini index (Gini, 1912), twoing criteria (Loh, 2011) or ANOVA (Cohen, 1988; Iversen & Norpoth, 1987) to decide which variable is most important and continues growing the tree until some criteria are achieved. It can output a variable importance list and the corresponding accuracy. However, CART have some undesirable properties like tending to select variables that have many classes (values) or many missing values, which will be investigated in Chapter 2. So an unbiased tree model called the Conditional Inference Trees (ctree)(Hothorn *et al.*, 2006b) will also be considered, which uses a significance test procedure to select variables instead of selecting the variable that maximizes an information measure. In fact, since the independent variables in the dataset in Chapter 3 are all long time series data, and with missing observations deleted, it actually does not make much difference whether we use CART or ctree. So in the classification analysis in Chapter 3, the traditional CART (rpart in R) is used, and in the regression analysis in Chapter 5, ctree is in use as that model can output some regression information (quantiles) to use directly in the next step. But rpart can't output quantiles directly, so the algorithm needs to be re-designed. For the others, there is no big difference due to which one to use in this thesis.

In Chapter 6, the performance of regression trees is analysed when fitted to data which simply following a mix of Gaussian and uniform distributions. When we predict this time series using simplified trees, the prediction error is calculated and decomposed into variance and other errors. When Gaussian or uniform effect is strong, those errors have different kinds of behaviour. Other exploration includes the best tree depth with minimum prediction error and the performance of Gaussian and quantile prediction intervals under different conditions.

1.2 Introduction to wavelet transforms

Generally, before attempting classification or regression, data preprocessing is essential. Input data contain redundant information as well as useful information. Redundant information may influence or even dominate the data information exploration. When the number of input variables is too large to conveniently handle, typically dimension reduction methods will be applied like PCA (principal component analysis) (Schölkopf *et al.*, 1997; Tipping & Bishop, 1999) or LASSO (least

absolute shrinkage and selection operator) (Meinshausen & Bühlmann, 2006; Tibshirani, 1996). PCA works by giving different weights to different variables, so as to strengthen the role of important variables, which have greater contributions to the data variance, while weakening that of redundant ones. LASSO works by selecting important variables when doing regression. In most cases, these methods are suitable for models with a lot of variables and are used as methods of dimension reduction. But for datasets having fewer variables, dimension reduction is not required. However, it is still important to separate useful information from the original variables. One method to do this is the wavelet transform.

To discover signal information on different resolution levels is like using a camera to enjoy landscape pictures far or near. A camera lens can take broad landscape pictures as well as zoom in to capture microscopic detail that is not easily seen by the human eye. The original idea of signal decomposition comes from the Fourier transform, which represents a signal using a sum of sine and cosine functions. But the limitation of the Fourier transform is also obvious. It has only frequency resolution and no time resolution. In other words, in the context of the Fourier transform, a given signal is composed into a collection of the individual frequencies of periodic signals, which is in the frequency zone. However, the wavelet transform gives resolution in both time and frequency.

The wavelet transform decomposes the original time series into different frequency bands including scaling (smooth) and wavelet (detail) coefficients on different levels at different time points. Noise has high frequency so it has little information on coefficients on higher resolution levels. In this way, the noise part can be separated from the signal and more information can be discovered. Originally, the wavelet transform and wavelet shrinkage (Donoho & Johnstone, 1994) were commonly used to smooth out noise variation in signals, a process called denoising. However, even without a formal denoising step, wavelets are able to separate out "signal" from "noise". This has a side effect of increasing the original data dimension as wavelet transformed data are on many resolution levels. So the method is basically suitable for datasets without high dimension.

When the data contain time series, like streaming data, the current data are usually dependent on old data. Building models using only the most recent data

seems unwise, so storing old information and utilizing long term variable information in an efficient way is important. The wavelet transform can pick out long term averages and short term fluctuations in data which can be exploited for classification when consecutive observations lack independence.

So, in this research, the wavelet transform is used before decision trees are applied to the data. Models with and without wavelet-transformed data are compared in terms of accuracy, explanatory ability and stability.

1.3 Motivating datasets

The data mainly used in this research are a medical dataset of monitoring variables recorded during the liver transplant surgery. It will be used for panel data analysis and time series analysis, including static time series and streaming data. In general, $A_{n,k,t}$ is denoted as the data value for individual $n = 1, 2, \ldots, N$, variable $k = 1, 2, \ldots, K$, and time $t = 1, 2, \ldots, T_n$, allowing the length of the time series to be different for each individual but the same for each variable in each individual. Thus, for the n^{th} individual, the data can be expressed as a $T_n \times K$ matrix

$$A_{n,\cdot,\cdot} = \begin{bmatrix} A_{n,1,1} & \cdots & A_{n,K,1} \\ A_{n,1,2} & \cdots & A_{n,K,2} \\ \vdots & \ddots & \vdots \\ A_{n,1,T_n} & \cdots & A_{n,K,T_n} \end{bmatrix}.$$

The full data can be written as a $\sum_{n=1}^{N} T_n \times K$ matrix A where

$$A^T = \begin{bmatrix} A^T_{1,\cdot,\cdot} & A^T_{2,\cdot,\cdot} & \cdots & A^T_{N,\cdot,\cdot} \end{bmatrix}.$$

Such explanatory data are referred to as panel (or longitudinal) data. The response variable y gives the group that each individual belongs to, which are shown as a vector

$$y^T = \begin{bmatrix} y^T_{1,\cdot} & y^T_{2,\cdot} & \cdots & y^T_{N,\cdot} \end{bmatrix},$$

where $y_{n,\cdot}$ has T_n identical values, defined by $(y_{n,1}, y_{n,2}, \ldots, y_{n,T_n})$. Since each individual belongs to only one group, the group labels for observations in that individual are the same.

Table 1.1: Monitoring variables recorded in the liver transplant (LT) surgery.

Abbreviation	Full name	Unit
CO	cardiac output	L/min
CI	cardiac index	L/min/m^2
SVR	systemic vascular resistance	dyne-s/cm^5
SVRI	systemic vascular resistance index	dyne-s/cm^5/m^2
Sys	systolic pressure	mm Hg
MAP	mean arterial pressure	mm Hg
Dia	diastolic pressure	mm Hg
SV	stroke volume	mL/beat
SVI	stroke volume index	mL/m^2/beat
HR	heart rate	beats/min

In medical control experiments, different treatments lead to different effects on individuals in different groups. Interpretation of how different treatments influence monitoring variables is important. That is studied as the panel data analysis in Chapter 3. During the surgery, if surgeons can monitor the real time condition of patients, that will save lives in an emergency. Forecasting monitoring variables like heart rate even one minute ahead will help surgeons monitor the condition better. That is the goal of the time series analysis in Chapter 4 for static prediction and in Chapter 5 for dynamic forecasting. Whereas static time series are used for a basis model-building analysis in Chapter 4, including a China air pollution data as the second static time series prediction application.

The data are from patients undergoing liver transplantation (LT) surgery between September 2004 and December 2011 at St James' University Hospital, Leeds, UK, which were recorded using LIDCO monitoring equipment (LIDCO, Cambridge, UK). The intraoperative monitoring variables recorded are shown in Table 1.1; for more details, see Milan *et al.* (2016).

In the panel data context of Chapter 3, the task is to classify each individual which contains many time points. Trees can only do point prediction, so three methods are designed to do classification on individual levels by combining time point information. We are going to check which method is better under situations like different noise levels and group imbalance. In addition, investigations also

include whether or not wavelet transformed data have a better performance than the original data.

In the time series context, in Chapters 4 and 5, only one variable, heart rate (HR), is in use instead of all K variables (other variables can be used as well in future work). We analyse each patient separately by regarding each time series $A_{n,k,.}$ as one dataset for analysis. The aim is point prediction for the static time series analysis in Chapters 4 as the basic exploration and interval prediction for the streaming data in Chapter 5. Static time series analysis is a basic analysis of applying wavelet transformed data to point prediction using decision trees. Although streaming data analysis is already quite popular, work based on interval prediction is rare. So new methods are designed for interval prediction with a suitable width. In Chapter 4, a weight matrix is also used to measure the influence of neighbouring provinces in the air pollution application.

Chapter 2

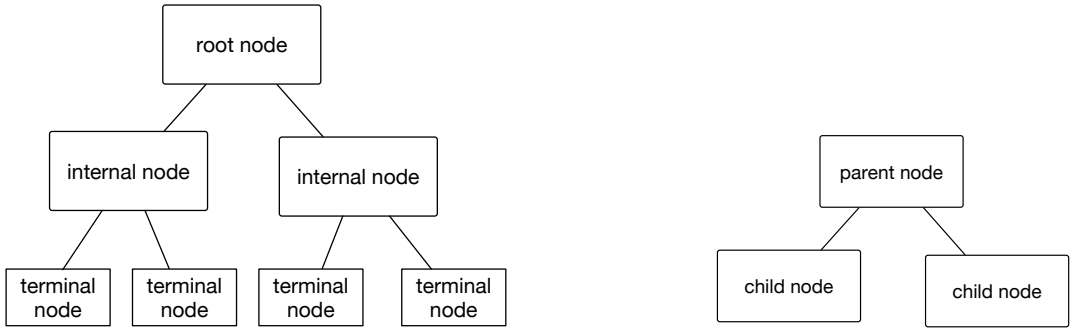
Theory review of trees and the wavelet transform

2.1 Introduction

This chapter gives a theoretical review of classification trees, regression trees and wavelet analysis. For wavelet analysis, a brief review is shown in Section 2.6 but primarily the focus is on trees. For trees, the construction process includes tree building (Section 2.2) and tree pruning (Section 2.3). After that some properties of trees are explored, including the bias property of different splitting criteria (Section 2.4), which help choose the best criterion between entropy and Gini for classification trees. Exploration also includes the computational complexity increase caused by noise variables (Section 2.5), which tells whether the increase of dimension caused by wavelet transform is worthwhile or not. Since regression trees are similar to classification trees except for the splitting criteria, we concentrate on classification trees. There are many variations of trees including the classification and regression trees (CART) (Breiman *et al.*, 1984a; Loh, 2011), C4.5 (Quinlan, 2014), CHAID (Kass, 1980) and ctree (Hothorn *et al.*, 2006b). Their tree construction processes are similar but with different splitting criteria and stopping rules or pruning criteria. In R, two tree packages are suitable for analyses: 'rpart' (Therneau *et al.*, 2014) and 'ctree' (Hothorn *et al.*, 2015), which are based on CART and ctree. So this chapter concentrates on CART and gives a brief introduction to ctree.

2.2 Tree building process

A decision tree can be seen as a set of branching decision rules. Figure 2.1 shows



(a) Example 1 (b) Example 2

Figure 2.1: Classification tree examples. The trees are not constrained to two or three levels. These are examples to illustrate the mutual relationship between successive levels.

the diagrams of typical trees. A root node is followed by internal nodes and the final nodes are called terminal nodes or leaves. Alternatively, we can refer to the sub node of a parent node as a child node. The number of child nodes per parent node is usually two but can be three or more. From the parent node to child nodes, the tree grows according to some decision rules based on an input attribute X. For example, if $X < c$, then it goes to the left child node, or else it goes to the right child node. Each node contains a subset of data. The task of inducing a decision tree is typically handled by a recursive partitioning algorithm which, at each non-terminal node in the tree, branches on that attribute which discriminates best between the cases filtered down to that node. The aim of growing the tree is to make each sub dataset after splitting as "pure" as possible, which means if there is only a few categories or values for the target attribute in this child node, the tree built has a good performance.

2.2.1 Classification tree building process

How to build such a tree? The difficulty is obvious. Global pattern optimisation of the tree's decision rules is an NP (nondeterministic polynomial time) problem,

which means finding the best pattern is high in computation complexity. Note that an ordered variable with K distinct values has $(K-1)$ splits of the form $X \leqslant c$, and an unordered variable with K distinct unordered values has $(2^{K-1}-1)$ splits of the form $X \in S$ (Loh, 2011). Alternatively, local optimisation can replace it, which means choosing the best split at every growing step. The subsequent split chosen will not influence the split chosen beforehand. At each step, the "best" split is chosen and it will not be changed in the successive steps. The "best" here can be measured in many ways, one of which is impurity. The best split chosen has the biggest decrease in impurity from parent node to child nodes.

Impurity here is highly related to information. The more information one attribute has, the more impure it is. For example, if there is only one class or one possible value in an attribute, then it is definitely pure and it has no information in it as it has no other possibles. If there are many classes or many possible values in an attribute and it is difficult to distinguish which class or value is the main one, then it becomes impure. The more equal the number of observations among classes or the more possible values the target attribute has, the more impure the attribute will be, as it is hard to tell which is the main class. Mathematically, impurity based criteria can be defined as the following (Rokach & Maimon, 2005, 2008).

For a classification tree, given a random variable Y with K discrete values, calculated from a vector of observations' proportions $P = (p_1, p_2, ..., p_K)$, where the proportions have values in $[0, 1]$, an impurity measure can be described as the projection

$$\Phi : [0,1]^K \longrightarrow R$$

that satisfies the following conditions:

(1) $\Phi(P) \geqslant 0$ (impurity value is not negative);

(2) $\Phi(P)$ is minimum if $\exists\, k$, such that component $p_k = 1$ (Y has only one value);

(3) $\Phi(P)$ is maximum if $\forall\, k,\ p_k = 1/K$, where $1 \leqslant k \leqslant K$ (Y has many values which share the same proportion);

(4) $\Phi(P)$ is exchangeable with respect to the components of P;

(5) $\Phi(P)$ is smooth (differentiable everywhere) in its range.

The impurity reduction from the parent node to child nodes is described using $\Phi(P)$ as the following. For example, a root node, containing data set S, has impurity as $\Phi(P)$. After the first splitting using variable X, the root node has M child nodes each containing sub dataset S_m, $m = 1, 2, \ldots, M$, satisfying

$$\bigcup_{m=1}^{M} S_m = S \text{ and } S_m \bigcap S_{m'} = \emptyset.$$

S_m is a sub dataset in S, in which observations have the same classes or value range. For numerical X, S_m can be a sample set satisfying $S_m = S\{X < x\}$. The value of x is chosen in the splitting step. For numerical X, the measure finds the best threshold and explore in that threshold for the best X value instead of trying every value in X. Then the impurity reduction is

$$\triangle\Phi_X(P\{Y|S\}) = \Phi(P\{Y|S\}) - \sum_{m=1}^{M} p\{S_m\} \cdot \Phi_X(P\{Y|S_m\}),$$

and

$$p\{S_m\} = \frac{\text{number of observations in } S_m}{\text{number of observations in } S}.$$

Two commonly used splitting criteria are shown as the following with the reason why they satisfy the impurity based criteria. One of the impurity measures is information (Jones, 1979), which is used to evaluate the information quantity involved in an event.

Let S be a set of events E_1, E_2, ... , E_K in which $P(E_k) = p_k$ with $0 \leqslant p_k \leqslant 1$ for $k = 1, 2, \ldots, K$ and

$$p_1 + p_2 + \cdots + p_K = 1.$$

Then comes the following definition.

Definition 1. *The self-information of event E_k is written as $I(E_k)$ $(0, \infty)$ and defined by*

$$I(E_k) = -\log_2 p_k.$$

There is no specific base of the logarithm since a base change will only influence the scale of the units. In most cases, the most common bases encountered are 2 and e. With base 2, I is measured in *bits* whereas, in base e, the units of I are *nats*, which is 0.693 times the number of bits. Without notation, base 2 is used. It can be seen that the smaller p_k is, the larger $I(E_k)$ is. This is consistent with the fact that the rarer an event is, the more information is conveyed by its occurrence.

Further, if two events E_j and E_k are independent $(j, k \in \{1, 2, \ldots, K\})$, then information has the property of

$$I(E_j \cup E_k) = I(E_j) + I(E_k).$$

Definition 2. *The entropy is the expectation, or average of the self-information*

$$E(I) = -\sum_{k=1}^{K} p_k \log_2 p_k.$$

Since p_k may be zero, $p_k \log_2 p_k$ could be indeterminate in this definition, so when $p_k = 0$, the value zero is assigned to $p_k \log_2 p_k$ (Jones, 1979). The proof that entropy information meets the conditions to be an impurity measurement is as follows.

(1) $E(I) \geqslant 0$.

Since $0 \leqslant p_k \leqslant 1$, so $-\log_2 p_k \geqslant 0$. Then it is easy to see that the sum of $p_k \log_2 p_k$ over k is also greater than or equal to zero.

(2) $E(I)$ is minimum if $\exists\, k$, such that $p_k = 1$.

If there is any k_1 such that $p_{k_1} = 1$, then the other p_k will be zero, for all $k \neq k_1$.

Then $-p_{k_1} \log_2 p_{k_1} = 0$ and the information for other k is also zero by definition. So $E(I) = 0$. According to (1), 0 is the minimum value of $E(I)$.

(3) $E(I)$ is maximised if $\forall\, k$, $p_k = 1/K$, where $1 \leqslant k \leqslant K$ Jones (1979).

It can be easily proven that for $x > 0$, $\ln x \leqslant x - 1$. Then using this result,

$$\sum_{k=1}^{K} p_k \ln \frac{1}{K\, p_k} \leqslant \sum_{k=1}^{K} p_k \left(\frac{1}{K p_k} - 1\right) \leqslant \sum_{k=1}^{K} \left(\frac{1}{K} - p_k\right) \leqslant 1 - 1 = 0. \tag{2.1}$$

So

$$\sum_{k=1}^{K} p_k \ln \frac{1}{K\, p_k} \leqslant 0.$$

Using Equation 2.1, it is easy to obtain

$$-\sum_{k=1}^{K} p_k \ln p_k \leqslant \sum_{k=1}^{K} p_k \ln K \leqslant \ln K.$$

By dividing the above inequality by ln 2, the same result is obtained for logarithm based on 2. That is

$$-\sum_{k=1}^{K} p_k \log_2 p_k \leqslant \sum_{k=1}^{K} p_k \log_2 K \leqslant \log_2 K.$$

Now it has been proven that $E(I)$ gets its maximum value of $\log_2 K$ only when $p_1 = p_2 = \cdots = p_K = 1/K$.

(4) $E(I)$ is exchangeable with respect to p_k.

Since addition is commutative, the position of $p_k \log_2 p_k$ can be changed with others, so it is obvious $E(I)$ is symmetric with respect to components of P.

(5) $E(I)$ is smooth (first differentiable everywhere) in its range.

Since I has its special value of 0 (when $p_k = 0$), so it is assumed that $p_k \neq 0$ when proving (5). The first differential result of $E(I)$ for p_k on $(0, 1]$ is

$$\frac{\partial E(I)}{\partial p_k} = -K - \sum_{k=1}^{K} \log_2 p_k.$$

So (5) is proved.

Now the entropy information has be proven as an impurity measure. The impurity reduction of the target attribute Y can be expressed as

$$\text{Entropy Gain}_X(Y, S) = \text{Entropy}(Y, S) - \sum_{m=1}^{M} p\{S_m\} \cdot \text{Entropy}(Y, S_m),$$

and the entropy gain rate is

$$\text{Entropy Gain rate}_X(Y, S) = \text{Entropy Gain}_X(Y, S)/\text{Entropy}(Y, S),$$

where

$$\text{Entropy}(Y, S) = -\sum_{Y \in S} p\{Y|S\} \cdot \log_2 p\{Y|S\}$$

and

$$\text{Entropy}(Y, S_m) = -\sum_{Y \in S_m} p\{Y|S_m\} \cdot \log_2 p\{Y|S_m\}.$$

Until now, we have applied the entropy information concept to the impurity measurement, and proven entropy information satisfying the conditions of impurity measure, so can be used as an impurity measure.

Another impurity measure is the Gini index, which is an impurity-based criterion that measures the differences between the probability distributions of the target attributes values. Specifically, the Gini index is a measure of how often a randomly chosen element from the attribute set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset.

Mathematically, it can be computed by summing the probability p_k of each class being chosen times the probability $1 - p_k$ of a mistake in categorizing that class.

To compute the Gini index for a set of classes, suppose $k \in \{1, 2, ..., K\}$, and let p_k be the fraction of classes labeled with value k in the set. Then, the Gini index can be defined as

$$\text{Gini}(p) = \sum_{k=1}^{K} p_k (1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2.$$

It can be proved that it meets all the conditions to be an impurity measurement. It reaches minimum (zero) when all classes in the attribute fall into a single target category and reaches its maximum of$(1 - 1/K)$ when all classes in the attribute have equal proportions. The decrease in Gini index from parent node to child nodes is called Gini Gain, which is defined as

$$\text{Gini Gain}_X(Y,\, S) = \text{Gini}\,(Y,\, S) - \sum_{m=1}^{M} p\{S_m\} \cdot \text{Gini}\,(Y,\, S_m),$$

where

$$\text{Gini}\,(Y,\, S) = 1 - \sum_{Y=y} p^2\{Y|S\}$$

and

$$\text{Gini}\,(Y,\, S_m) = 1 - \sum_{Y=y} p^2\{Y|S_m\}.$$

The above equations mean that the Gini Gain is just the Gini index decrease from parent node to child nodes after split by classes in an attribute.

2.2.2 Regression tree building process

Regression trees are similar to classification trees, except the response variable is numerical for regression trees while that for classification trees are classical. In a typical binary split regression tree, the input attributes space is partitioned

by a sequence of binary splits into terminal nodes. At each terminal node, the predicted response value is the mean of the data in each terminal node. Compared to a classification tree, the main difference is the impurity criterion. For regression trees, it is the averaged squared error that is used, which measures the difference between the real data and the predicted data. For one specific node with N observations, the impurity is

$$R(Y,S) = \frac{1}{N}\sum_{y_n \in S}\left(y_n - \bar{y}^S\right)^2,$$

where $\bar{y}^S$ is the mean value of the observations in dataset S. The reason why use $\bar{y}^S$ as the node predicted value is due to that only using $\bar{y}^S$, $R(Y,S)$ can reach its minimum value with given y_n. So the predicted response value is chosen as $\bar{y}$ in this sample set S. $R(Y,S)$ has a simple interpretation. For sample set S, $R(Y,S)$ is the within node sum of squares. When dataset S is split into M subsets S_m, $m = 1, 2, \ldots, M$, the proportion of observations in subset m is $p\{S_m\}$. Then the overall sum of squared error R after splitting has the decrease

$$\triangle R(Y,S) = R(Y,S) - \sum_{m=1}^{M} p\{S_m\} \cdot R\left(Y,\, S_m\right).$$

Thus a regression tree is formed by iteratively splitting nodes so as to maximise the decrease in $\triangle R$ at each step.

2.2.3 Conditional inference trees

Conditional Inference Trees (ctree) (Hothorn *et al.*, 2006a,b) estimate a regression relationship by binary recursive partitioning in a conditional inference framework. Roughly, the algorithm works as follows:

1. Test the global null hypothesis of independence between each of the input variables and the response variable (which may be multivariate as well). Stop if this hypothesis cannot be rejected. (For example, the corresponding p-value is bigger than a specific threshold. The threshold in use is 0.05.) Otherwise select the input variable with strongest association to the response. This association is measured by a p-value corresponding to a test

for the partial null hypothesis of independence between single input variable and the response. In the conditional inference (permutation tests), either multiplicity-adjusted p-values (Bonferroni adjusted, default) or univariate p-values are applied. A node is able be split when the p-value is smaller than 0.05.

2. Implement a binary split in the selected input variable. The split itself can be established by any split criterion, including those in CART, CHAID and so on.

3. Recursively repeat steps 1) and 2).

For univariate regression, variable selection methods in use are Spearman correlation test, the Wilcoxon-Mann-Whitney test or the Kruskal-Wallis test and permutation tests based on ANOVA statistics or correlation coefficients (Hothorn *et al.*, 2006b). Since ctree is based on statistical parameters, there is no bias due to missing values.

2.3 Tree pruning

2.3.1 Complexity parameter

Trees can sometimes grow very big with only one observation in each terminal node. Such over fitting problems will result in trees having poor generalisability (having high training accuracy but low testing accuracy). So building a tree with both suitable complexity and accuracy is necessary. Typically, we 'prune back' trees to avoid overfitting.

Before the tree pruning process, a sufficiently large tree T_{max} is built. For the tree that does not grow to be that large, it is defined as T, which is a subtree in T_{max} and sharing the same root node as T_{max}. The node in either T_{max} or T is defined as t, which can be any possible node in the tree. T_t refers to the subtree of T starting from node t to its terminal nodes. The tree and node misclassification costs (misclassification rate) are referred to as $R(T)$ and $R(t)$, and they have the relationship

$$R(T) = \sum_{t \in \tilde{T}} R(t),$$

where $\tilde{T}$ refers to the set of terminal nodes of tree T. In addition to the misclassification cost, The complexity of the tree is measured by the number of terminal nodes of the the tree. For example, for tree T, the complexity is defined as $|\tilde{T}|$.

In the tree pruning process, the weakest link node(s) is cut in each step. That means, with the same complexity increase, the node (branch) with the smallest misclassification rate decrease will be cut. The complexity parameter α is used to record the tree in each step, which determines the weight of complexity in the tree pruning process. The higher the value of α, the smaller the tree will be.

In the first step, T_1 is pruned from T_{max} by cutting off all the terminal nodes that do not decrease misclassification cost at all. This means T_1 is the smallest subtree of T_{max} satisfying

$$R(T_1) = R(T_{max}) \tag{2.2}$$

The α after this first step is defined as $\alpha_1 = 0$.

In the second step, the first weakest link is cut. For any node $t \in T_1$, set

$$R_\alpha(t) = R(t) + \alpha|\tilde{t}|, \tag{2.3}$$

where $R_\alpha(t)$ combines the misclassification cost and tree complexity. As node t is itself the terminal node, so it has the complexity as 1. For any branch T_t, define

$$R_\alpha(T_t) = R(T_t) + \alpha|\tilde{T_t}|. \tag{2.4}$$

In that sense, define a function $g_1(t)$, $t \in T_1$, by

$$g_1(t) = \frac{R(t) - R(T_t)}{|\tilde{T_t}| - 1}, \quad t \notin \tilde{T_1}. \tag{2.5}$$

For $t \in \tilde{T_1}$, define $g_1(t)$ as $+\infty$. $g_1(t)$ measures the misclassification cost decrease from node t to the branch T_t starting from node t. The node or branch with the minimum $g_1(t)$ value will be cut. Define the weakest link $\bar{t_1}$ in T_1 as the node such that

$$g_1(\bar{t_1}) = \min_{t \in T_1} g_1(t) \tag{2.6}$$

and put

$$\alpha_2 = g_1(\bar{t_1}). \tag{2.7}$$

Define a new tree T_2 by pruning away T_{t_1} from T_1. Now continue to prune the tree from T_2. A decreasing sequence of subtrees is recorded

$$T_1 > T_2 > T_3 > \cdots > \{t_1\}.$$

The corresponding $\{\alpha\}$ are an increasing sequence, that is

$$\alpha_1 < \alpha_2 < \alpha_3 < \cdots < \alpha_n.$$

Now we need to select one of these trees (best α) as the optimum-sized tree. Generally, the tree that has the least misclassification cost for the test data is chosen. This is done using cross-validation.

2.3.2 Best complexity parameter chosen from cross-validation

In practice, rpart uses cross-validation to select the best complexity parameter α, so as to decide the depth of the tree. Cross-validation is a technique to evaluate predictive models by partitioning the original samples into a training set to fit the model, and a test set to evaluate it. For example, if some of the observations are selected for training and the rest for testing, the results may not be robust due to sampling sensitivity especially when the data set size is not big. So k-fold cross-validation is suggested to make the results more reliable.

In k-fold cross-validation, the original dataset is randomly partitioned into k equal-sized subsets (such as $k = 10$). Of the k subsets, a single subset is retained as validation data for testing the model, and the remaining $k - 1$ subsets are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsets used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimate of tree performance. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.

The process for k-fold cross-validation in rpart is shown in Figure 2.2:

Step 1 Randomly sample a training dataset A_1 and treat the remaining data A_2 as a test dataset for which the explanatory variable is known but the response information is unknown;

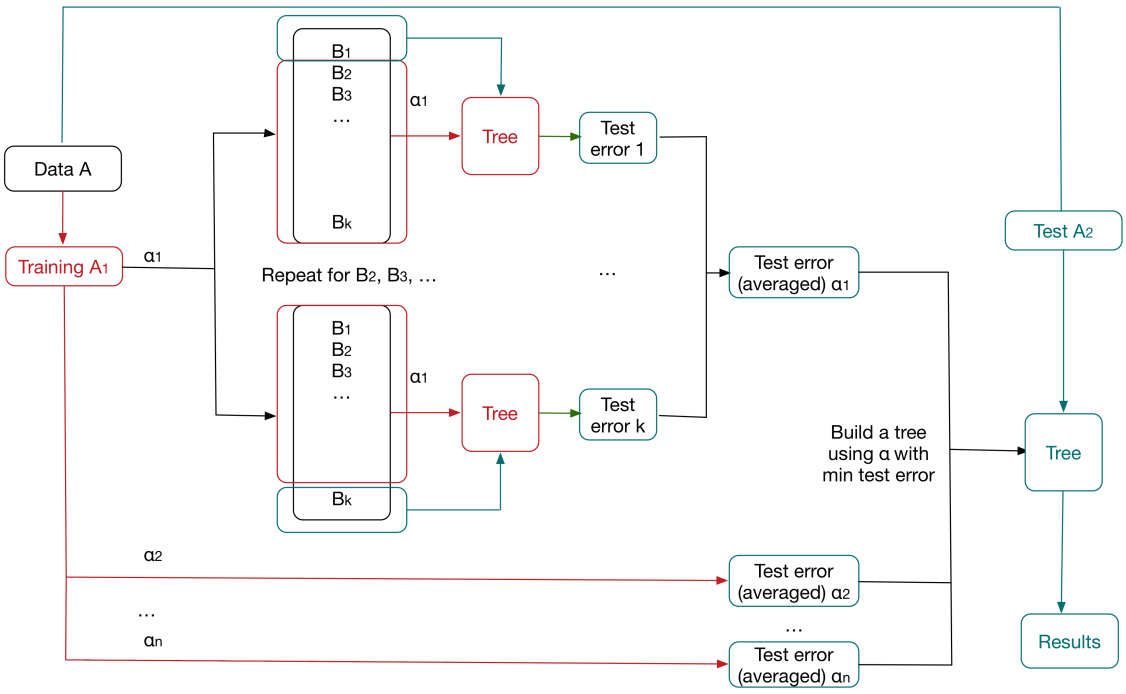


Figure 2.2: k-fold cross-validation flowchart.

Step 2 Use all the data in A_1 to build a tree, and let the corresponding α value list be $(\alpha_1, \alpha_2, ..., \alpha_n)$. So each α value leads to a tree with specific depth.

Step 3 Divide A_1 into k folds as B_1, B_2,... B_k;

Step 4 Choose α_1 as the complexity parameter in the tree pruning process;

Step 5 Choose B_2 to B_k as training data and B_1 as test data. Use this training data and $\alpha = \alpha_1$ to build a tree. The error of classify the test data using this tree is recorded as e_1^x.

Step 6 Repeat Step 5 using B_2, B_3,... B_k as test data separately and get k error values. The mean and the standard deviation of these k errors are noted as e^x and $e^{x,sd}$ (short for standard error) for α_1.

Step 7 Repeat Steps 4–6 and get the e^x and $e^{x,sd}$ for every α value.

Step 8 Choose the tree depth with the lowest e^x.

The tree being chosen will be used for testing using data A_2.

2.4 Splitting bias

In this section, the properties of different splitting criteria (entropy, Gini, etc.) are explored under different conditions. The splitting bias is defined as the difference between the observed and the theoretical information gain. For classification trees, one of the most popular criteria is information gain, namely the Shannon entropy information gain from parent node to child nodes. However this criterion is liable to unfairly favour attributes with large numbers of values or categories compared to those with few. This will be proven later in this section. In this sense, noise variables with large numbers of values could be selected in preference to genuinely informative attributes with fewer values. In general, this would lead to poorer predictive performance from the resulting tree. The probability to choose predictor variables with more information decreases.

In addition, splitting rules favour those noisy predictor variables with more missing

values since their sample size is smaller than others. In this case, as the sample size decreases, the probability for choosing noisy predictor variables with more information decreases.

The gain ratio calculated from information gain also suffers the same kind of problem. It is acknowledged that attributes with very low information values (low attribute information) appear to gain an unfair advantage (Strobl *et al.*, 2007, sections:1-2.2).

Another splitting criterion is χ^2. In fact, this criterion is not biased since for different degrees of freedom, χ^2 follows different probability distribution functions. Using degrees of freedom, χ^2 eliminates the problem of bias. Although there are splitting criteria like χ^2 that have no bias, CHAID (Kass, 1980) in R, which uses χ^2 as the splitting criterion, however requires dependent and explanatory variables both to be categorical variables, which is not suitable for the datasets. For regression problems, ctree will be used, which is an unbiased method, having no splitting bias in these cases.

2.4.1 Bias due to missing values.

In this section, it will be shown that both Gini and entropy information have bias in favour of choosing variables with more missing values. So no matter which splitting criterion is chosen, we have to face the bias due to missing values. That is why pre-processing is applied to missing values in the real data application in Chapter 3.

When information gain is calculated, there is a bias between the theoretical gain and observed gain values due to the difference between the sample and population distributions. This bias can be different when there are missing values. For missing values in independent variables, most procedures deal with them by leaving out incomplete observations. The models in this thesis actually are more ambitious. Any observation with values for the dependent variable and at least one independent variable will participate in the modelling (Therneau *et al.*, 1997). For the Gini index, how bias is influenced by missing values has been investigated

by Strobl *et al.* (2007)(Sections 1-2.2). So an equivalent analysis for entropy is conducted as the following.

Assume there are an independent variable X and a dependent variable Y with two categories. The number of observations in the first category for Y is N_1, and the other is N_2. Then the entropy information for the root node is

$$ent_N = -\frac{N_1}{N}\log_2\left(\frac{N_1}{N}\right) - \frac{N_2}{N}\log_2\left(\frac{N_2}{N}\right).$$

In order to calculate the expectation of ent_N, for simplicity, we first calculate the bias for $E\left(\frac{N_2}{N}\log_2\left(\frac{N_2}{N}\right)\right)$, where $N_2 \sim B(N,p)$ and N is fixed. The result is

$$\begin{aligned} E\left(-\frac{N_2}{N}\log_2\left(\frac{N_2}{N}\right)\right) =& E\left(-\frac{N_2}{N}\left(\log_2(N_2) - \log_2(N)\right)\right) \\ =& E\left(-\frac{N_2}{N}\left(\log_2(N_2)\right)\right) + p\log_2(N). \end{aligned}$$

If bias has value 0, that is the observed information gain is equal to the theoretical information gain, then

$$E\left(-\frac{N_2}{N}\log_2\left(\frac{N_2}{N}\right)\right) = -p\log_2(p),$$

so that

$$E\left(-\frac{N_2}{N}\left(\log_2(N_2)\right)\right) = -p\log_2(Np).$$

Then bias is given by $E\left(-\frac{N_2}{N}\left(\log_2(N_2)\right)\right) - (-p\log_2(Np))$. Similarly, we can get the bias for N_1, which follows $B(N, 1-p)$. Then the total bias for the root node is

$$\begin{aligned} E(bias_N) =& E\left(-\frac{N_1}{N}\left(\log_2(N_1)\right)\right) - (-(1-p)\log_2(N(1-p))) \\ &+ E\left(-\frac{N_2}{N}\left(\log_2(N_2)\right)\right) - (-p\log_2(Np)). \end{aligned}$$

It is not easy to get $E\left(-\frac{N_1}{N}\left(\log_2(N_1)\right)\right)$ and $E\left(-\frac{N_2}{N}\left(\log_2(N_2)\right)\right) - (-p\log_2(Np))$ analytically as they contain the terms of the form $E(N_1\log_2(N_1))$, so a polynomial expression is used to approximate the log function. Given that

$$log_2(1+a) = a - \frac{a^2}{2} + \frac{a^3}{3}\cdots, \tag{2.8}$$

for $|a| < 1$, we substitute $a = p - 1$ in Equation (2.8), and require p not be small. If $X \sim B(n, p)$, then its moments are given by

$$\begin{aligned} E(X) &= np, \\ E\left(X^2\right) &= np + n(n-1)p^2, \\ E\left(X^3\right) &= np + p^2\left(3n^2 - 3n\right) + p^3\left(n^3 - 3n^2 + 2n\right), \text{and} \\ E\left(X^{k+1}\right) &= pq \cdot \frac{d\left(E\left(X^k\right)\right)}{dp} + npE\left(X^k\right) \text{ for } k = 3, 4, \ldots. \end{aligned}$$

Given that N_1 and N_2 are binomially distributed, we obtain, using the first two terms in the expansion of the log function,

$$\begin{aligned} E\left(\widehat{ent_{N_2}}\right) =& E\left(-\frac{X}{N}\log_2\frac{X}{N}\right) \\ =& E\left[-\frac{X}{N}\left(\frac{X-N}{N} - \frac{1}{2}\left(\frac{X-N}{N}\right)^2\right)\right] \\ =& E\left[-\frac{2X^2}{N^2} + \frac{3X}{2N} + \frac{X^3}{2N^3}\right]. \end{aligned}$$

Now, using the formulae for $E(X^k)$, it is easy to get

$$E\left(\widehat{ent_{N_2}}\right) = \left(\frac{1}{2N^2} - \frac{2}{N} + \frac{3}{2}\right)p + \left(-\frac{3}{2N^2} + \frac{7}{2N} - 2\right)p^2 + \left(\frac{1}{N^2} - \frac{3}{2N} + \frac{1}{2}\right)p^3.$$

Then, the bias of entropy for N_2 can be calculated as

$$\begin{aligned} bias_{N_2} =& E\left(\widehat{ent_{N_2}}\right) - E\left(ent_{N_2}\right) \\ =& E\left(\widehat{ent_{N_2}}\right) - \left(-2p^2 + \frac{3}{2}p + \frac{1}{2}p^3\right) \\ =& \left(\frac{1}{2N^2} - \frac{2}{N}\right)p + \left(-\frac{3}{2N^2} + \frac{7}{2N}\right)p^2 + \left(\frac{1}{N^2} - \frac{3}{2N}\right)p^3. \end{aligned}$$

Similarly, the bias for N_1 is

$$bias_{N_1} = \left(\frac{1}{2N^2} - \frac{2}{N}\right)(1-p) + \left(-\frac{3}{2N^2} + \frac{7}{2N}\right)(1-p)^2 + \left(\frac{1}{N^2} - \frac{3}{2N}\right)p^3,$$

so the bias for the root node is

$$\begin{aligned} bias_N =& bias_{N_1} + bias_{N_2} \\ =& \left(\frac{1}{2N^2} - \frac{2}{N}\right) + \left(-\frac{3}{2N^2} + \frac{7}{2N}\right)(1 - 2p + 2p^2) + \left(\frac{2}{N^2} - \frac{3}{N}\right)p^3. \end{aligned} \tag{2.9}$$

For the root node, the expected entropy information is $E(\widehat{ent})$ for N observations. After splitting the root node, it is easy to get the left child node and the right child node with N_L observations and N_R observations respectively. Two cases where X and Y are independent and when they are associated are considered as the following.

Case 1: explanatory variable X is independent of response variable Y.

In this case,

$$\begin{aligned} E\left(\widehat{\triangle ent}\right) =& E\left(\widehat{ent}\right) - \frac{N_R}{N} E\left(\widehat{ent_R}\right) - \frac{N_L}{N} E\left(\widehat{ent_L}\right) \\ =& bias_N + E\left(ent\right) \\ & - \frac{N_R}{N}\left(bias_R + E\left(ent_R\right)\right) - \frac{N_L}{N}\left(bias_L + E\left(ent_L\right)\right). \end{aligned}$$

Since X is independent of Y, so $E\left(ent\right) = E\left(ent_R\right) = E\left(ent_L\right)$, and

$$\begin{aligned} E\left(\widehat{\triangle ent}\right) =& bias_N - \frac{N_R}{N} bias_{N_R} - \frac{N_L}{N} bias_{N_L} \\ =& \left(\frac{2}{N} + \frac{1}{2N^2} - \frac{1}{2N_L N_R}\right) p + \left(-\frac{3}{2N^2} + \frac{3}{2N_L N_R} - \frac{7}{2N}\right) p^2 \\ & + \left(\frac{3}{2N} + \frac{1}{N^2} - \frac{1}{N_L N_R}\right) p^3 + \left(\frac{2}{N} + \frac{1}{2N^2} - \frac{1}{2N_L N_R}\right) (1-p) \\ & + \left(-\frac{3}{2N^2} + \frac{3}{2N_L N_R} - \frac{7}{2N}\right) (1-p)^2 \\ & + \left(\frac{3}{2N} + \frac{1}{N^2} - \frac{1}{N_L N_R}\right) (1-p)^3. \end{aligned}$$

As X, Y are independent, the split in X can be anywhere. It is assumed to be in the middle of X, so $N_L = N_R = \frac{N}{2}$. The other circumstances can be explored in future work. Then we have

$$\begin{aligned} E\left(\widehat{\triangle ent}\right) = & \left(\frac{2}{N} - \frac{3}{2N^2}\right) p + \left(\frac{9}{2N^2} - \frac{7}{2N}\right) p^2 + \left(\frac{3}{2N} - \frac{3}{N^2}\right) p^3 + \\ & \left(\frac{2}{N} - \frac{3}{2N^2}\right) (1-p) + \left(\frac{9}{2N^2} - \frac{7}{2N}\right) (1-p)^2 + \\ & \left(\frac{3}{2N} - \frac{3}{N^2}\right) (1-p)^3. \end{aligned}$$

Since $E(\triangle ent) = 0$, then

$$bias = E\left(\widehat{\triangle ent}\right).$$

If $p = 0.5$, then $bias = \frac{5}{8N}$. This shows that, when X and Y are independent, as sample size N decreases, entropy gain increases, which means noise (redundant) variables with more missing values have a better chance to be chosen.

Case 2: explanatory variable X is associated with response variable Y.

In practice, if X is not a noise variable, then X and Y are associated. For example, X and Y are related as $Y = a + bX$, where a and b are constants. Since X is dependent on Y, the split should be at the same place as that in Y. In that case, the sample will become pure after splitting, which means $E\left(\widehat{ent_R}\right) = E\left(\widehat{ent_L}\right) = 0$. Then, the expectation of entropy gain is:

$$\begin{aligned} E\left(\widehat{\triangle ent}\right) =& E\left(\widehat{ent}\right) - \frac{N_R}{N} E\left(\widehat{ent_R}\right) - \frac{N_L}{N} E\left(\widehat{ent_L}\right) \\ =& bias_N + E(ent). \end{aligned}$$

Then, the bias of the entropy gain is

$$\begin{aligned} bias =& bias_N + E(ent) - E(ent) \\ =& bias_N. \end{aligned}$$

Similarly, when $p = 0.5$, from Equation (2.9), $bias_N = -5/(8N) < 0$. So, there are circumstances, when X is not a noise variable, and X, Y are dependent, that we have a negative bias. It is opposite to the situation for independent variables.

The approximation is verified by simulation, choosing $p = 0.5, 0.6, ..., 0.9$ as p and $1 - p$ are symmetric. For a specific N (the total number of observations), $N_2 \sim B(N, p)$ and $N_1 = N - N_2$ are chosen. Then, the entropy bias in the simulation can be calculated using N, N_1, N_2 and assumptions from the above two situations. The results in Figure 2.3 show that the theoretical values are roughly the same as the simulated ones, which confirms our approximation. One difference is that when N is small and p or $1 - p$ is small, the log approximation

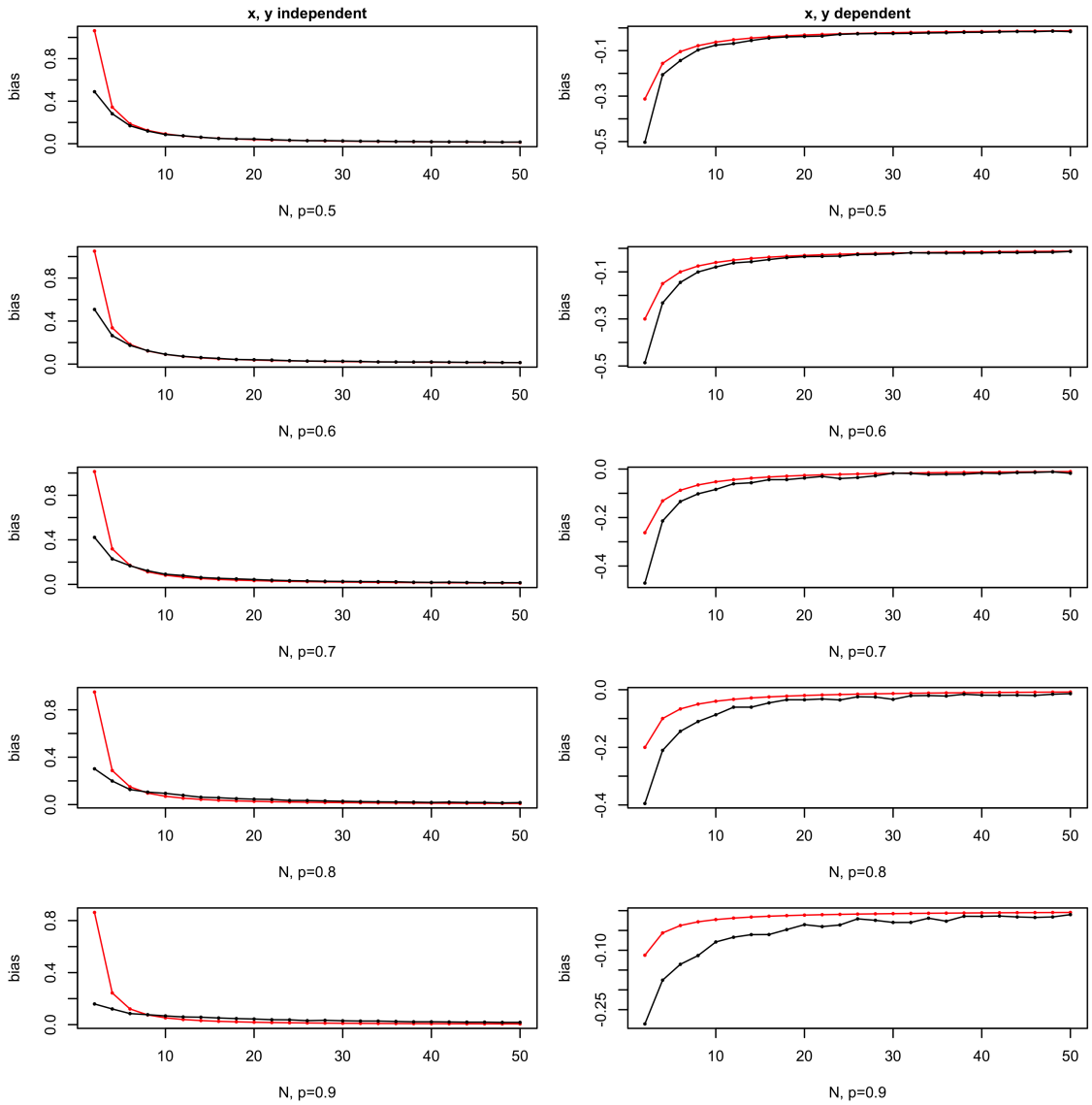


Figure 2.3: Entropy gain bias in theory and practice. Red dots are theoretical values and black dots are based on simulation. The x axis shows the total number of observations, N.

used in Equation (2.8) is not so suitable, so there is gap between the simulated and theoretical results.

For noise variables, the more missing values there are, the bigger the chance they are to be chosen as a splitting variable. For informative variables, the more missing values there are, the smaller chance they have to be chosen. Both situations will lead to bad results. That is why we deal with missing values and other outliers in the data cleaning process in Chapter 3.

2.4.2 Bias related to more values or categories

In this section, it is explored how the entropy and Gini criteria have bias related to the number of categories or number of possible values in X. A χ^2 statistic is also involved as a criterion for comparison, which does not have this bias due to more values or categories as its degree of freedom changes accordingly.

The ground truth is assumed as that X and Y are independent. When the ground truth is unknown, for any split in X, the event that X is dependent on Y in each child node is accepted with probability p. The hypotheses are

$$H_0\text{: } X \text{ is independent of } Y; \qquad H_1\text{: } X \text{ is dependent on } Y$$

When H_0 is true, then X is independent of Y for any possible split in X. The corresponding probability to accept H_0 is

$$(1-p)^r$$

where

$$r = \begin{cases} m-1, & \text{ordered variable } X \\ 2^{m-1}-1, & \text{categorical variable } X, \end{cases}$$

and m is the number of unique values for an ordered variable or categories for a categorical variable. When H_1 is accepted, we have

$$P(H_1 \text{ is accepted} | H_0 \text{ is true}) = 1-(1-p)^r\,,$$

which means that there is at least one split in X that makes X depend on Y. It is easy to see that explanatory variables with more values or categories have a better chance to be chosen even though X is independent of Y. For the Gini index or

entropy gain, they have not eliminated this multiple comparison effect, so they still have that kind of bias. But for a Chi-squared test (Kass, 1980), it uses the corresponding p value instead, and it has different distribution for different degrees of freedom calculated from the possible values or categories in X, so it eliminates this effect.

Now, a simulation is conducted to explore the bias effect for the Gini gain and entropy gain while compared with $p\left(\chi^2\right)$. The process is shown in Algorithm 1 and the corresponding results are shown in Figure 2.4. It is obvious, for entropy

Algorithm 1 Bias simulation

Require: number of values or categories in explanatory variable X : m
number of values or categories in response variable Y : k
simulation times t number of samples N

Ensure: $entropy\ gain$, $entropy\ gain\ rate$, $Gini\ gain$, $p\left(\chi^2\right)_{m,k}$

```
for t=1:1:100 do
  for m=2:1:10 do
    for k=2:1:10 do
      X = sample(1 : m, N, replace = TRUE)
      Y = sample(1 : k, N, replace = TRUE)
      entropy gain = entropy gain(X, Y)
      entropy gain rate = entropy gain rate(X, Y)
      Gini gain = Gini gain(X, Y)
      p(χ²) = p(χ²)_(X,Y)
    end for
  end for
end for
return entropy gain_{m,k}, entropy gain rate_{m,k}, Gini gain_{m,k}, p(χ²)_{m,k}
```

gain and entropy gain rate, that the bias increases when k or m increases. For the Gini index, it also increases, but the bias value changes little, always being around 0.009. For χ^2, as expected, there is no sign of bias due to more values or categories in X and Y.

For classification purposes, the Gini index is chosen as the splitting criterion as its bias due to more values or categories is not that large compared to entropy. The rpart package in R includes the choice of Gini index as the default splitting criterion. For χ^2, although it is good, the CHAID package in R can only be applied to

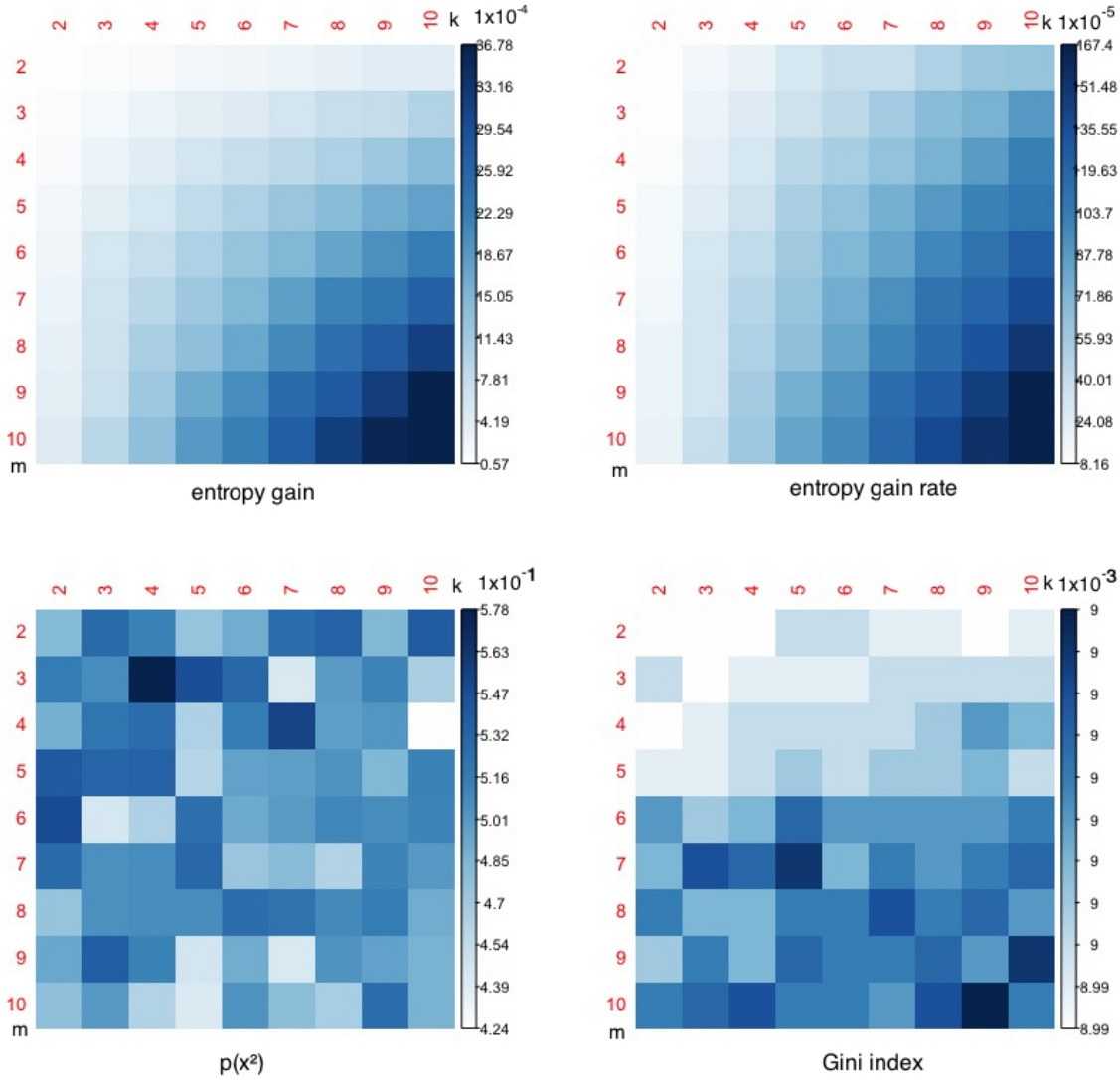


Figure 2.4: Bias when number of categories or values in X and Y changes. Here, the x axis label k is the number of values or categories in response variable Y and the y axis label m is the number of values or categories in explanatory variable X. The darker the shade, the higher the bias. The important point is how the intensity changes across k and m in each sub figure. The comparative intensity of the same k and m among different figures is also important but it is not included in our analysis context. So the values are not scaled.

categorical variables while our later analyse include continuous response variables. There are many algorithms to build classification trees, including ID3 (Quinlan, 1986), C4.5 (Quinlan, 2014) and CART (Loh, 2011), etc. ID3 is one of the original algorithms, which uses the entropy information criterion, but it does not apply any pruning nor does it deal with numeric attributes or missing values. As an evolution of ID3, C4.5 uses the entropy information gain ratio as the splitting criterion. The splitting ceases when the number of instances to be split is below a certain threshold, and error-based pruning is performed after the growing phase. Further, C4.5 can handle numeric attributes. In terms of CART, such binary trees are constructed based on the Gini index or twoing criterion and the tree is pruned by complexity criterion. It can also involve misclassification costs and prior probability distributions in the tree building process (Rokach & Maimon, 2008). As software R is used for coding, and the decision tree package rpart is generally based on CART, so CART is chosen as the classification tree using Gini index as the splitting criterion.

2.5 Influence of noise variables on CART computational complexity

2.5.1 Introduction

The contribution in this section is to explore how the number of noise variables influences the computational time under simplified conditions using the existed Bonferroni multiplier (Bonferroni, 1936).

This section explores how the number of noise variables influences the computational complexity compared to merely using informative variables. The term computational complexity here refers to the time complexity of an algorithm. In computer science, the time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the string representing the input. Time complexity is commonly estimated by counting the number of elementary operations (such as addition, subtraction, multiplication, division, comparison operations) performed by the algorithm, where an elementary operation takes a fixed amount of time to perform. Thus the amount of time

taken and the number of elementary operations performed by the algorithm differ by at most a constant factor. In that way, the number elementary operation is counted to represent the computational complexity.

For CART, the following ideal conditions are assumed:

1. All the independent variables can be divided into effective variables and noise variables. The criterion is whether they are used in the tree growing process or not. As the most effective variables will be chosen for splitting firstly. Those variables not chosen have less effect than those chosen. A tree building process includes both a growing process and pruning process (or stopping criteria). This time, the tree is assumed to choose the stopping criteria, so that we only need to concentrate on the growing process. Noise variables refer to variables that are not used in the tree growing process.

2. All variables are categorial variables for convenience of calculation.

3. For every split, no matter how many categories the independent variable has, there are always two child nodes after the parent node since CART is a binary tree. All nodes are assumed to stop splitting at the same time which means the depth is the same for every branch on the same level.

4. When one independent variable is chosen as a split, it will not be chosen again.

Such simplifying assumptions are made for easy of calculation. In reality, the process is more complex than that. Define N as the number of explanatory variables including both effective variables and noise variables, M as the number of effective variables, and c_j as the number of categories in the j^{th} independent variable. In the splitting process, the explanatory variable will be split into two intervals (numerical) or groups (categorical). The number of all possible ways of separating the c_j categories into two groups is the Bonferroni multiplier (Bonferroni, 1936). Here since all categories are split into two groups, it is

$$S\left(c_j, 2\right) = \sum_{r=1}^{2} \left(-1\right)^{2-r} \frac{r^{c_j}}{r!\left(2-r\right)!}.$$

2.5.2 Computational complexity without noise variables

For the initial split, assume variable a_1 is chosen, and the computational complexity is

$$2^0 \sum_{j=1}^{M} S\left(c_j, 2\right) b + m,$$

where b is the computational complexity involved in calculating the entropy information for one possible split in one variable and m is the computational complexity for calculating the entropy information in y.

After that, variable a_1 will not be used again because of Assumption 4. Assume variable a_2 is chosen as the split for both child nodes after a_1, and the computational complexity for both child nodes are similar, so the total computational complexity at step 2 is

$$2^1 \sum_{j=2}^{M} S\left(c_j, 2\right) b.$$

Even though it is essential to calculate the entropy gain from the parent node to child nodes, just calculating the entropy information in child nodes is sufficient since the parent node entropy information has already been calculated from the previous step. So here we just count the computational complexity for the child nodes.

Under Assumption 3, the number of terminal nodes increases in a power of 2. After summing all the computational complexity for all the nodes, the computational complexity for the whole tree is:

$$CC_{effect} = \sum_{s=0}^{M-1} 2^s \sum_{j=s+1}^{M} S\left(c_j, 2\right) b + m.$$

2.5.3 Computational complexity with noise variables

It is easy to calculate the computational complexity with noise variables in a similar way to the case without noise variables. The difference is the total number of explanatory variables in use is not M but N, which includes the noise variables.

The difference comparison will be shown in the subsection 2.5.4. For the initial split, assume variable a_1 is chosen, so the computational complexity is

$$2^0 \sum_{j=1}^{N} S\left(c_j, 2\right) b + m.$$

For the second split, it is

$$2^1 \sum_{j=2}^{N} S\left(c_j, 2\right) b.$$

There are many reasons for the tree to stop growing, such as the node becomes pure or all the variables have the same proportion in all the y categories. At level $M+1$, all the M effective variables are used, so the tree will test whether the first noise variable is effective or not. Since noise variables are assumed to be those not selected by the tree. So after the testing, the tree will stop growing. The computational complexity for the testing is

$$2^M \sum_{j=M+1}^{N} S\left(c_j, 2\right) b.$$

For the whole tree, the computational complexity is

$$CC_{effect+noise} = \sum_{s=0}^{M} 2^s \sum_{j=s+1}^{N} S\left(c_j, 2\right) b + m.$$

2.5.4 Computational complexity increase

The increase in computational complexity due to the presence of noise variables is

$$\begin{aligned} CC_{inc} =& CC_{effect+noise} - CC_{effect} \\ =& \sum_{s=0}^{M} 2^s \sum_{j=s+1}^{N} S\left(c_j, 2\right) b + m - \sum_{s=0}^{M-1} 2^s \sum_{j=s+1}^{M} S\left(c_j, 2\right) b - m \\ =& \sum_{s=0}^{M} 2^s \sum_{j=M+1}^{N} S\left(c_j, 2\right) b. \end{aligned}$$

Assuming that the c_j has the same value across different j, then we can rewrite $S\left(c_j, 2\right) b$ as one value u. Then CC_{inc} becomes

$$CC_{inc} = \left(2^{M+1} - 1\right) \cdot (N - M)\, u.$$

which is a linear function of the number of noise variables, $N - M$. So, even when methods which increase the dimension of explanatory variables are used before the application of decision trees, the computational complexity will not increase dramatically.

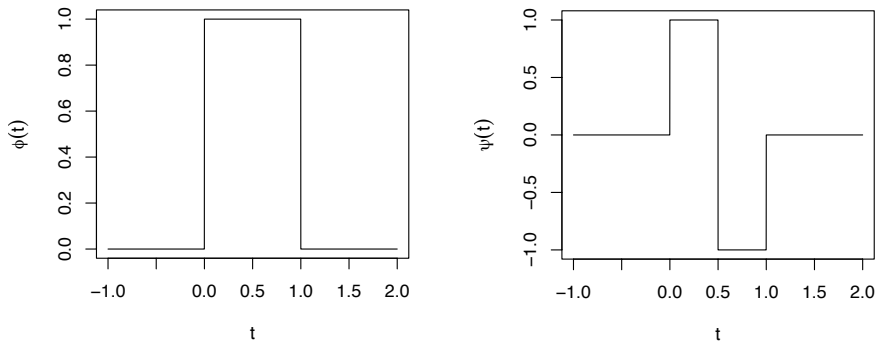
2.6 Wavelet analysis

Sometimes data have nearly the same range and mean but different frequencies like sine functions with different periods. Decision trees unfortunately can not easily classify or do regression on such data as there is no obvious split point to separate them. For example, the time series in Figure 2.6 has the same mean and range, so decision trees can not find the split to classify the groups. For time series, there might be dependence between successive observations; growing trees which treat each time point as an independent observation may ignore this information. Wavelet analysis can deal with this by picking out patterns in short term fluctuations in data which can be exploited for classification when consecutive observations lack independence. To discover signal information at different resolution levels is like using a camera to enjoy landscape pictures near or far. A camera lens can take broad landscape pictures as well as zoom in to capture microscopic detail that is not easily seen by the human eye. Decomposing data into different resolution levels, including the smooth and detail information, may help decision trees be useful.

The original idea of signal decomposition into different frequencies comes from the Fourier transform, which represents a signal using a sum of sine and cosine functions. But the limitation of the Fourier transform is also obvious: it has only frequency resolution and no time resolution. In other words, in the context of the Fourier transform, signals are just a collection of periodic signals of the individual frequencies. However, the wavelet transform decompose a signal in both time and scale (frequency).

MODWT is short for maximal overlap discrete wavelet transform, which is a non-decimated wavelet transform. For data, where values change over time and different individuals have different frequency characteristics, simple Fourier analysis is not suitable. So wavelet analysis is used here to simultaneously represent

a signal in the time and frequency domains (Aykroyd *et al.*, 2016). MODWT is actually based on the discrete wavelet transform (DWT), but unlike the DWT, the MODWT is not constrained to data whose sample size is a power of 2 (Percival & Walden, 2000).



(a) (b)

Figure 2.5: The Haar scaling function (a) and mother wavelet (b).

One of the wavelet bases is Haar wavelet. As shown in Figure 2.5(a), the Haar scaling function is defined as

$$\phi(t) = \begin{cases} 1 & t \in [0,1) \\ 0 & \text{else.} \end{cases} \tag{2.10}$$

Using dilation and translation methods, the scaling function at resolution level j and location k is:

$$\phi_{j,k}(t) = 2^{j/2}\phi\left(2^j t - k\right).$$

Note that

$$\phi\left(2^j t - k\right) = \begin{cases} 1 & t \in I_{j,k} \\ 0 & \text{else;} \end{cases}$$

and that $\phi_{j,k}(t)$ is compactly supported on $I_{j,k} = [2^{-j}k, 2^{-j}(k+1))$, where $j = 0, 1, 2, \ldots, J$ $(2^J \leqslant n)$ and $k = 0, 1, 2, \ldots n-1$ represent the resolution level and location respectively. When $j = 0$, the scaling coefficients are actually the original time series. Thus, when j is small, it means wavelets are highly localized at a fine scale resolution level, representing brief transient effects. Conversely, when j is large, it represents lower frequency activity at a coarser scale resolution level

(Aykroyd *et al.*, 2016). Here the factor of $2^{j/2}$ ensures energy preservation, defined by

$$energy = \frac{1}{2}\int_0^{2\pi} |f(x)|^2 dx \tag{2.11}$$

so that the energy in the data set will be preserved and that is why wavelets are orthogonal and have inverse transforms, for more detail, see Graps (1995). Then, the scaling coefficients $s_{j,k}$ can be calculated as

$$s_{j,k} = <x(t), \phi_{j,k}> = \int_R x(t)\,\phi_{j,k}(t)\,dt = 2^{j/2}\int_{I_{j,k}} x(t)\,dt.$$

As shown in Figure 2.5(b), the Haar mother wavelet function $\psi(t)$ is defined as

$$\psi(t) = \begin{cases} 1 & t \in [0, 0.5) \\ -1 & t \in [0.5, 1) \\ 0 & \text{else,} \end{cases}$$

and, the wavelet function at resolution level j and location k is

$$\psi_{j,k}(t) = 2^{j/2}\psi\left(2^j t - k\right).$$

The wavelet coefficients $d_{j,k}$ can be calculated as

$$d_{j,k} = s_{j-1,k} - s_{j,k}.$$

The scaling coefficients vector $s_j = (s_{j,0}, s_{j,1}, \ldots, s_{j,n-1})$ and wavelet coefficient vector $d_j = (d_{j,0}, d_{j,1}, \ldots, d_{j,n-1})$ become new variables which can be used for classification and regression.

When it comes to MODWT in R, the implementation in waveslim ignores the energy preservation factor $2^{j/2}$, which will not affect the results when using classification or regression trees. Since the MODWT is designed for time series of any length, it has its own methods for dealing with boundary conditions like periodic and reflection. In this thesis, reflection is chosen, as the time series is not periodic. After reflection, the time series of length n, becomes one of length $2n$, so the first n wavelet coefficients of the transform are used as the wavelet transformed variables. The level of the wavelet transform depends on the length

of the test data. The maximum level should not exceed the allowance of the test data ($2^J \leqslant n$).

An example of using the Haar wavelet to transfer a time series is shown in Figure 2.6. The first 300 observations are from $\sin(0.2t)$, and the second 300 observations are from $\sin(t)$. The observations all contain noise following $\mathcal{N}(0, 0.1)$. This data set can not be classified easily by trees as the two parts share the same range [-1, 1]. But the time series is different with higher frequency on the right side data. After wavelet transform, the differences for time series with different frequencies become obvious. Scaling coefficients on level 8 (s_8) are smoother than that on level 1 (s_1), which is almost the original time series. Wavelet coefficients on level 8 (d_8) show information about oscillations of length 2^8 observations. Although trees can not distinguish data set with the same mean and range, however, after wavelet transform, d_1, d_8 and s_8 have different ranges in the two parts, so that trees can classify these wavelet transformed variables.

The Haar wavelet belongs to the family of Daubechies' wavelet (Daubechies, 1988). Different wavelet functions have different vanishing moments, among which the Haar wavelet has one vanishing moment. The number of vanishing moments is the maximum degree of the polynomials that the scaling function can reproduce. The higher the vanishing moments, the smoother the basis is, as shown in Figure 2.7. For example, D4 (Daubechies' wavelet on level 4) has two vanishing moments, so that its basis is smoother than that of Haar wavelet, so the wavelet transformed data is also smoother than that of Haar. The smoother the basis is, the more complex the scaling and wavelet function are. The discrete scaling coefficients $s_{j,k}$ for Haar are

$$s_{j,k} = \frac{1}{2^j} \sum_{l=0}^{2^j-1} x_{k-l},$$

and the wavelet coefficients $d_{j,k}$ for Haar are

$$d_{j,k} = \frac{1}{2^j} \left(\sum_{l=0}^{2^{j-1}-1} x_{k-l} - \sum_{l=2^{j-1}}^{2^{j-1}-1} x_{k-l} \right).$$

The coefficients $s_{j,k}$ and $d_{j,k}$ will be used in the wavelet transformed variables in the later chapters.

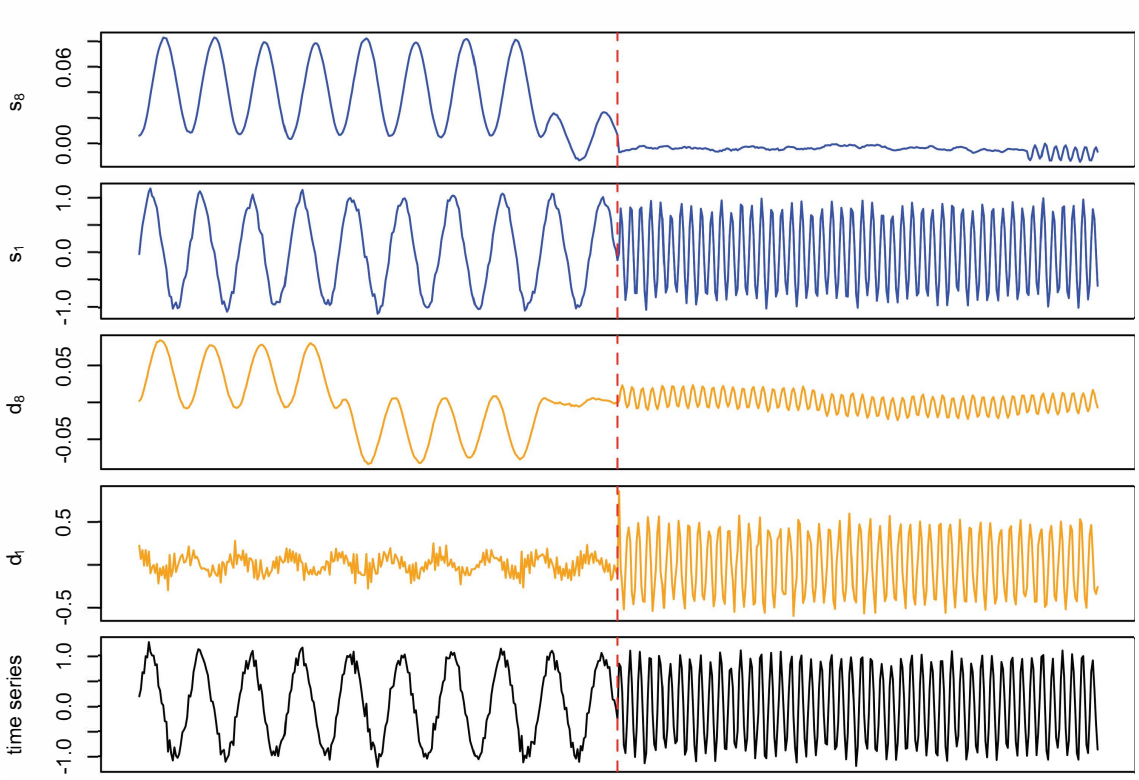


Figure 2.6: Wavelet transformed time series on level 8 using the Haar wavelet basis. The x axis is time.

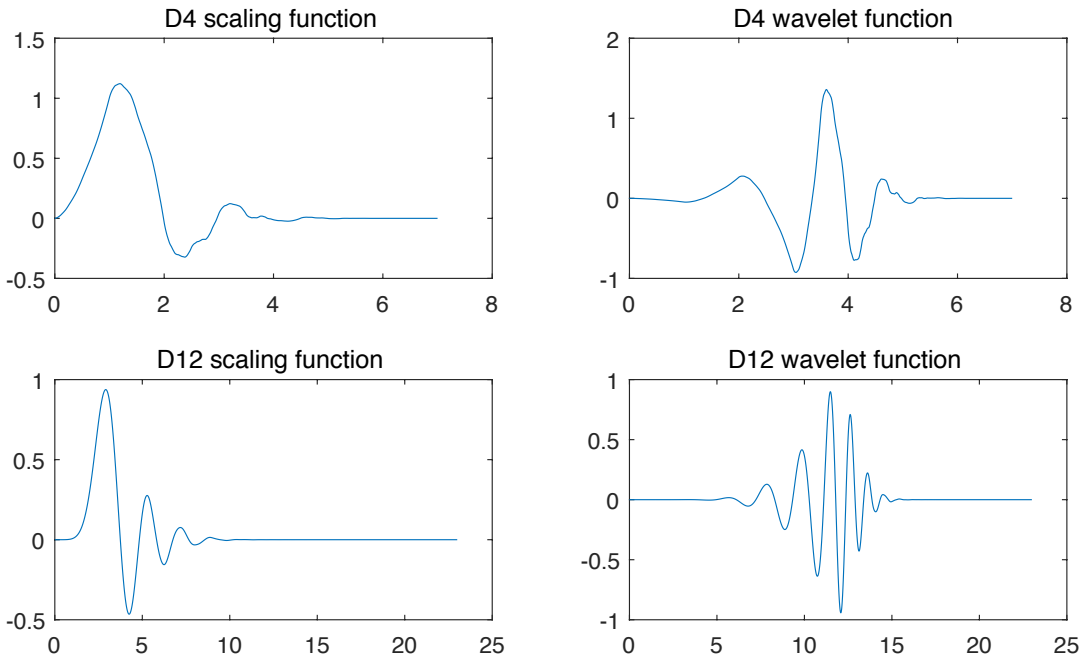


Figure 2.7: Scaling function and mother wavelet for D4 and D12.

The simple form of the Haar functions mean that the Haar wavelet is easier to interpret. Compared to other Daubechies' wavelets with higher vanishing moments, the Haar wavelet is more simple and so more interpretable. So, in the remaining chapters, the Haar wavelet is used as the wavelet transform basis. There are circumstances when other wavelet transform behaves well as shown in Section 3.3.4. But if their behaviour is similar, the Haar wavelet is chosen for its easy interpretation.

Chapter 3

Panel data prediction

3.1 Introduction

This chapter explores the circumstances under which wavelet-transformed variables have a better classification performance for panel data than using variables on their original scale. The tree models and wavelet transform methods have been described in Chapter 2. Use of wavelet-transformed data provides localized mean and difference variables which might be more effective than the original variables. Three methods are considered by aggregating panel data to classify at the individual-level, which will be described in detail in Section 3.2. This will be illustrated with simulated data and data gathered during a liver transplantation surgery. Most of the content in this chapter has been published as Zhao *et al.* (2018).

There are data which often contain multiple time series variables for organisations or individuals that need classification, especially in areas such as economics, finance, marketing, medicine and biology. It can also be important to determine which of the time series are useful in performing the classification; interpreting this information can be highly useful in investigating the relationships between the variables and the class labels.

Our interest in this problem was motivated by data collected on patients undergoing liver transplant surgery. Each patient is classified into one of two groups, according to whether they did or did not use beta-blocker medication. During the operation, monitoring took place for several variables such as heart rate and

systolic blood pressure, with data recorded once every heartbeat.

The dataset in use is the panel data mentioned in Chapter 1. The explanatory data are

$$A^T = \left[\begin{array}{cccc} A^T_{1,\cdot,\cdot} & A^T_{2,\cdot,\cdot} & \cdots & A^T_{N,\cdot,\cdot} \end{array} \right].$$

The time series length T_n varies among different individual n and the response variable is

$$y^T = \left[y^T_{1,\cdot} \quad y^T_{2,\cdot} \quad \cdots \quad y^T_{N,\cdot} \right],$$

where elements in $y_{n,\cdot}$ should all be Group 1 or all be Group 2. Difficulties in analysing such datasets include: (1) unequal values of T_n; (2) aggregating the panel data to provide classification for each individual; and (3) lack of independence between consecutive times.

Such data are generally subject to noise if they are collected or recorded by people or machines. Wavelet shrinkage (Donoho & Johnstone, 1994) is a popular denoising method, which is commonly used to smooth out random noise variation in signals. However, even without a formal denoising step, wavelets are able to separate out "signal" from "noise", and this property will be used to improve prediction performance. It will also be seen that wavelets can pick out short term fluctuations in real data which can be exploited for classification when consecutive observations lack independence. Instead of using the standard decimated discrete wavelet transform (DWT), the maximal overlap discrete wavelet transform (MODWT) will be used, see, for example, Percival & Walden (2000, ch. 5), as it is not constrained by time series length T_n and each time point is represented at all resolution levels of the MODWT. Equivalent translation-equivariant transforms are the non-decimated stationary wavelet transform (Nason & Silverman, 1995) and cycle-spinning (Coifman & Donoho, 1995).

The classification and regression trees (CART) method of Breiman *et al.* (1984b) will be used for classification purpose. Using DWT (or MODWT) with CART (or other decision trees or random forests) in time series data has already been considered (Alickovic & Subasi, 2016; Gokgoz & Subasi, 2015; Upadhyaya & Mohanty, 2016), as well as other classification methods (Maharaj & Alonso, 2007, 2014) but, to the best of our knowledge, until now the application to panel data is quite rare. Previous authors have directly converted the wavelet representation of panel data

into cross-sectional data using summaries such as energy, standard deviation, or entropy (Upadhyaya & Mohanty, 2016; Zhang *et al.*, 2015). Inspired by this idea, Method 3 is designed. But, in this chapter, other methods will be designed as well and more detailed exploration among those methods will be conducted including in simulation and real data applications.

However detecting when and how MODWT can help CART in classification accuracy and variable selection for panel data is important. Thus, this chapter uses CART with original and wavelet-transformed variables to classify panel data. This chapter makes the contributions of: (1) designing another two methods (Method 1 and Method 2) including the one (Method 3) shown in (Upadhyaya & Mohanty, 2016; Zhang *et al.*, 2015) but with mean and standard deviation as summaries; (2) applying these methods to simulated data so as to explore how different methods perform under different conditions. For chapter structure, the methodology is introduced in Section 3.2, and is applied to simulated panel data experiments in Section 3.3 before analysing the liver transplantation (LT) panel data in Section 3.4. Some concluding comments appear in Section 3.5.

3.2 Methodology

3.2.1 Methods introduction

In this section, three methods are proposed to produce individual-level classifications from panel data, which can be applied to the original data, the wavelet-transformed data, or a combination of both. A flow diagram outlining these methods is shown in Figure 3.1.

Since we wish to classify individuals, but are dealing with panel data, the predictions from CART can not be used directly as these classify each time point separately. The information needs to be combined either by combining time-point level predictions or by aggregating data first and then performing classification for individuals. Several methods are proposed to classify individuals based on panel data, by illustrating in terms of the original data A. Specifically, Method 1 is based on time-point level predictions, Method 2 is based on both time-point level and individual level predictions and Method 3 is based on individual level predictions.

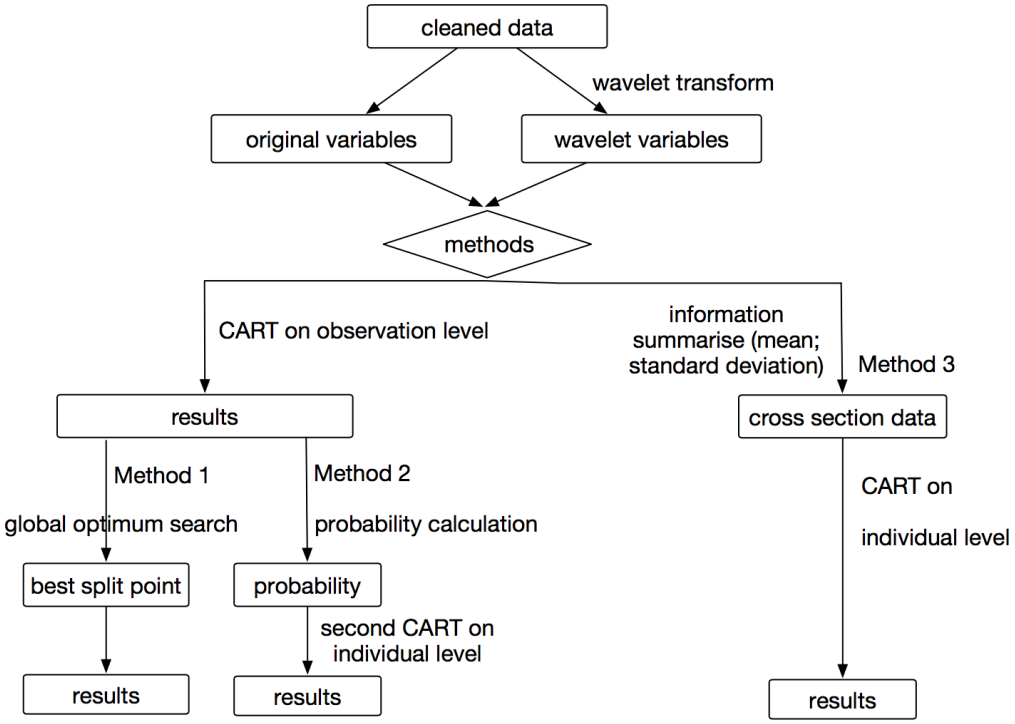


Figure 3.1: Flow diagram for panel data classification.

Equivalently, these methods can also be applied to the wavelet-transformed data W, or indeed to a combination of A and W. For simplicity, each of the methods is described in terms of a binary classification, but is easily generalized to more than two groups.

3.2.2 Method 1: prediction aggregation after classification

Using CART, it is easy to obtain the predicted class $\hat{y}_{n,t}$ of individual n and each time point $t = 1, \ldots, T_n$. Schematically we have

$$A_{n,\cdot,\cdot} = \begin{bmatrix} A_{n,1,1} & \cdots & A_{n,K,1} \\ A_{n,1,2} & \cdots & A_{n,K,2} \\ \vdots & \ddots & \vdots \\ A_{n,1,T_n} & \cdots & A_{n,K,T_n} \end{bmatrix} \rightarrow \hat{y}_{n,\cdot} = \begin{bmatrix} \hat{y}_{n,1} \\ \hat{y}_{n,2} \\ \vdots \\ \hat{y}_{n,T_n} \end{bmatrix}.$$

For individual n, we compute the proportion of time points which were classified as Group 1,

$$P_n = \frac{1}{T_n} \sum_{t=1}^{T_n} I\{\hat{y}_{n,t} = 1\},$$

where the indicator variable is defined as

$$I\{\hat{y}_{n,t} = 1\} = \begin{cases} 1 & \text{if } \hat{y}_{n,t} = 1 \\ 0 & \text{if } \hat{y}_{n,t} = 2. \end{cases}$$

The prediction at each time point is used to predict the class of individual n:

$$A_{n,\cdot,\cdot} \rightarrow \hat{y}_n = \begin{cases} 1 & \text{if } P_n \geqslant a \\ 2 & \text{otherwise,} \end{cases}$$

where the best split point a is found using a global search ($0 \leq a \leq 1$). Every P_n will be tested as a possible a value in the training dataset. The P_n that leads to the highest training accuracy will be selected as the a.

3.2.3 Method 2: predictions based on time-point level and individual level CART

Method 2 is based on Method 1 as it uses the output probability information from Method 1. The detail is shown as the following.

1. Here, a classification tree is constructed as in Method 1. The number of variables putting into the tree may not be the same as the number of variables really participating in the built tree because some noise variables will not be selected by the tree. Suppose variable k' is used in this classification tree, where $k' \in \{1, 2, \ldots, K'\}$, making a total of K' newly renumbered variables, which are used in the already built tree. So the dataset used in this tree becomes

$$A'_{n,.,.} = [A_{n,k',t}]_{T_n \times K'}.$$

For each observed value of each of these variables, the tree is used to derive the probability of classifying an observation as being from Group 1, based on the subtree descending from that observation. For example, suppose the variable HR (heart rate) is used in the tree and lower heart rate (like 60 beats/min) means higher probability (0.8) for the individual to be classified as group 1. Then the value of 60 will be replaced by 0.8. The detail is shown as below.

2. Consider variable k'. To compute the derived probabilities, we inspect the nodes in the tree where variable k' is used. In a node, with split point η, observations satisfying $A_{n,k',t} < \eta$ are directed to one sub-tree, while those satisfying $A_{n,k',t} > \eta$ are directed to a different sub-tree. In each sub-tree, $A_{n,k',t}$ is replaced by the proportion of observations classified as Group 1, which is denoted as $P_{n,k',t}$.

For example, for $A_{n,k',t}$, if $A_{n,k',t} > \eta$, then

$$P_{n,k',t} = P_{A_{n,k',.} > \eta},$$

where $P_{A_{n,k',.} > \eta}$ is the proportion classified as Group 1 for variable k' in that sub tree node satisfying $A_{n,k',.} > \eta$.

3. If variable k' is used in more than one node, we take the product of the probabilities (note that $k' \in \{1, 2, \ldots, K'\}$ implies that variable k' must be used in at least one node). For example, if variable k' appeared twice, with thresholds η_1 and η_2, then

$$P_{n,k',t} = P_{A_{n,k',.} > \eta_1} \cdot P_{A_{n,k',.} > \eta_2},$$

where $P_{A_{n,k',.}>\eta_1}$ and $P_{A_{n,k',.}>\eta_2}$ are the proportions classified as Group 1 for variable k' in the nodes satisfying $A_{n,k',.} > \eta_1$ and $A_{n,k',.} > \eta_2$ respectively.

4. This whole process is repeated for each $k' \in \{1, 2, \ldots, K'\}$ in turn, converting each time series of observed values to a vector of proportions and finally, from $A'_{n,.,.}$, we obtain
$$P_{n,.,.} = [P_{n,k',t}]_{K' \times T_n}.$$

5. Then, the mean and standard deviation of these empirical probabilities are taken as new variables to be used in a "second-stage" CART. (Of course, other summaries could be used.) Schematically, we have
$$P_{n,\cdot,\cdot} = \begin{bmatrix} P_{n,1,1} & \cdots & P_{n,K',1} \\ P_{n,1,2} & \cdots & P_{n,K',2} \\ \vdots & \ddots & \vdots \\ P_{n,1,T_n} & \cdots & P_{n,K',T_n} \end{bmatrix} \tag{3.1}$$
$$\swarrow\searrow \qquad\qquad \swarrow\searrow$$
$$P^{(m)}_{n,k,\cdot} = \begin{bmatrix} P^{(m)}_{n,1,\cdot}, P^{(sd)}_{n,1,\cdot} & \cdots & P^{(m)}_{n,K',\cdot}, P^{(sd)}_{n,1,\cdot} \end{bmatrix}.$$

6. Then the mean and standard deviation matrix for all individuals and variables are calculated as a cross sectional data matrix
$$\widetilde{P} = \left[\begin{array}{ccc|ccc} P^{(m)}_{1,1,.} & \cdots & P^{(m)}_{1,K',.} & P^{(sd)}_{1,1,.} & \cdots & P^{(sd)}_{1,K',.} \\ P^{(m)}_{2,1,.} & \cdots & P^{(m)}_{2,K',.} & P^{(sd)}_{2,1,.} & \cdots & P^{(sd)}_{2,K',.} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ P^{(m)}_{N,1,.} & \cdots & P^{(m)}_{N,K',.} & P^{(sd)}_{N,1,.} & \cdots & P^{(sd)}_{N,K',.} \end{array}\right]. \tag{3.2}$$

After that, we apply CART to $\widetilde{P}$, and get the predicted value for each individual:

$$\widetilde{P} = \left[\begin{array}{ccc|ccc} P^{(m)}_{1,1,.} & \cdots & P^{(m)}_{1,K',.} & P^{(sd)}_{1,1,.} & \cdots & P^{(sd)}_{1,K',.} \\ P^{(m)}_{2,1,.} & \cdots & P^{(m)}_{2,K',.} & P^{(sd)}_{2,1,.} & \cdots & P^{(sd)}_{2,K',.} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ P^{(m)}_{N,1,.} & \cdots & P^{(m)}_{N,K',.} & P^{(sd)}_{N,1,.} & \cdots & P^{(sd)}_{N,K',.} \end{array}\right] \rightarrow \hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix}. \tag{3.3}$$

Note that in this second stage, each individual has just one mean and one standard deviation value corresponding to each of the variables in that tree, hence there is only one predicted class for each individual.

3.2.4 Method 3: data aggregation before classification

Here, we aggregate the original data for each individual into a small number of summaries for each variable — chosen to reflect the nature of the data — to form individual level cross-sectional data, and then use CART directly. In the implementation, the mean $A^{(m)}_{n,k,\cdot}$ and standard deviation $A^{(sd)}_{n,k,\cdot}$ of variable k for each individual n $A_{n,k,t}$ are used to construct cross sectional data

$$\widetilde{A} = [A^{(m)}, A^{(sd)}]_{N\times 2K},$$

where $A^{(m)} = [\, A^{(m)}_{n,k,\cdot}\,]_{N\times K}$ and $A^{(sd)} = [\, A^{(sd)}_{n,k,\cdot}\,]_{N\times K}$. For dataset unknown, mean and variance are proposed, but when the dataset has other good statistics, they can also be included. The process is the same as in Equations (3.1) to (3.3), but applied to data A rather than probabilities P.

3.3 Simulation study

We now explore the performance of Methods 1–3 using both original and wavelet-transformed data in a simulation study, before applying the methods to our LT data in Section 3.4. 100 replicate trials are conducted for each simulation. For every trial, new data are generated on N individuals and then split the N individuals into training and test sets, with $0.8N$ individuals used for training and the remaining data used to assess performance. All three methods are used for each dataset.

This section investigates: (1) whether wavelet variables have better performance in classification than the original variables; (2) which variables are more important in the tree; (3) which of the proposed methods perform better in different circumstances. The criteria are prediction accuracy (the percent of correct classified individuals) and the ability to correctly identify informative variables.

All computations were performed in R (R Core Team, 2014), using the packages rpart (Therneau *et al.*, 2014) for constructing classification trees and waveslim (Whitcher, 2013) for wavelet decomposition.

Table 3.1: Simulation variables and parameters. Variables V_1–V_5 are contaminated at rate θ by normally distributed noise which has the same mean and variance as the variable in question.

	Variable name	Distribution	Parameters	
			Group 1	Group 2
Explanatory variables	V_1	AR(1)+$N(0,\sigma_2^2)$	$\alpha = 0.8$	$\alpha = 0.5$
			$\sigma_1^2 = 0.36$	$\sigma_1^2 = 0.75$
	V_2	sin+$N(0,\sigma_2^2)$	$2\sin(3t+5)$	$2\sin(4t+5)$
	V_3	sin+$N(0,\sigma_2^2)$	$2\sin(5t+6)$	$2\sin(5t+3)$
	V_4	Poisson	$\lambda_1 = 2$	$\lambda_1 = 2.5$
		$+N(0,\sigma_2^2)$	$\lambda_2 = 2.5$	$\lambda_2 = 2$
	V_5	exp (rate=λ)	$\lambda_1 = 1$	$\lambda_1 = 2$
		$+N(0,\sigma_2^2)$	$\lambda_2 = 2$	$\lambda_2 = 1$
Redundant variables	V_6	Poisson	$\lambda = 1$	
	V_7	Poisson	$\lambda = 2$	
	V_8	AR	$\alpha = 0.7$	
	V_9	MA	$\beta = 0.6$	
	V_{10}	exp	$\lambda = 8$	

3.3.1 Data generation

Panel data are generated with two groups, comprising a total of $N = 300$ individuals. For each individual, there are 10 "time series" variables. Five of these variables (V_1–V_5) are informative, having different distributions in the two groups, while the remainder (V_6–V_{10}) are identically distributed across both groups and referred to as redundant variables. Details of the variable models, distributions and parameters are shown in Table 3.1.

For each informative variable, the parameters of the models are chosen so that the first two moments are identical for the two groups. The five explanatory variables follow an AR process (V_1), sine models (V_2 and V_3), Poisson (V_4) and exponential (V_5) distributions, so both autocorrelated and independent variables

are included. We independently generate $T_n^s \sim U(T_n/4, 3T_n/4)$ as a "change point" for each individual, and for variables V_4 and V_5 observations before and after this point follow the same distribution but with different parameters. Noise is added to V_1–V_5 in two ways. Firstly, Gaussian white noise $\epsilon_t \sim N(0, \sigma_2^2)$ is added, and the data is contaminated by making random replacements at rate θ with $N(\mu, \sigma^2)$, where μ and σ^2 are mean and variance. These data generation methods ensure that the marginal distribution for each explanatory variable between the two groups is the same, while the joint distribution is different between the two groups.

In order to assess the influence of noise levels and group balance on classification accuracy and selection of explanatory variables, simulations are conducted under different circumstances with noise level $0 \leq \sigma_2 \leq 20$, contamination rate $0.1 \leq \theta \leq 0.8$ and the number of individuals in Group 1 ranging from 150 to 270, with the total number $N = 300$ fixed.

After generating the data, these variables are wavelet transformed using the Haar wavelet basis. Then, we use the original and wavelet-transformed data to conduct 100 replicate trials for each experiment.

3.3.2 Separate analysis for each explanatory variable

Classification accuracy

In order to tell which explanatory variable is most effective in classification, classification trees are built with only one informative explanatory variable at a time, replacing the other four informative variables with standard Gaussian white noise $N(0, \sigma_2 = 1)$. This provides a check that the CART methodology is correctly selecting informative variables while ignoring variables that contain no useful information. In each case, classification accuracy for the original variables on the test data is generally around 50%. However, Table 3.2 shows that, for wavelet variables, it is generally above 85% except for V_3, which is noticeably lower. In particular, variable V_2 is the most informative. This can be attributed to the wavelet coefficients distinguishing the different frequencies of V_2 in the two groups, which will be shown in detail later in this section. Method 3 is usually the best method due to the aggregation over time points effectively averaging out random variation, giving a cleaner picture of the differences between the groups.

Table 3.2: Classification accuracy when using wavelet-transformed version of each of the informative variables in isolation.

Method	Informative variable				
	V_1	V_2	V_3	V_4	V_5
1	86.83%	99.67%	53.67%	92.17%	97.17%
2	85.67%	100.00%	47.67%	93.00%	97.33%
3	100.00%	100.00%	94.33%	100.00%	100.00%

Interpretation of scale choice

A further advantage of using wavelet-transformed data is the added insight which can sometimes be gained by considering which scales are used in the classification. In order to illustrate clearly, parameters are simplified by fixing $T_n = 768$ (T_n can be other values as well) and $T_n^s = T_n/2 = 384$, and without white noise added to the explanatory variables, but V_2–V_5 still have 10% of generated observations randomly replaced with noise as they are originally noise free but V_1 has noise involved when the data are generated. Figures 3.2–3.4 show examples of the time series generated and plots of those wavelet-transformed variables which were most commonly selected as containing useful information by CART. We note that CART can easily detect differences in the mean level of a variable with a single split, and can also partially detect increased variance by two splits.

V_1 For V_1, the main variables chosen were s_8 (representing smoothing over a window of $2^8 = 256$ time points, which has less noise included) and d_1 (the difference between successive observations). Recall that V_1 follows an AR(1) model with autoregressive parameter $\alpha = 0.8$ in Group 1 and $\alpha = 0.5$ in Group 2, so the short-term autocorrelation is substantially higher in Group 1. Using the raw data does not access this information, but it is detected by the local averages in s_8 and local fluctuations in d_1.

V_2 For V_2, the frequency difference in the sine function is detected by both s_1 and d_1, which means successive information is more important.

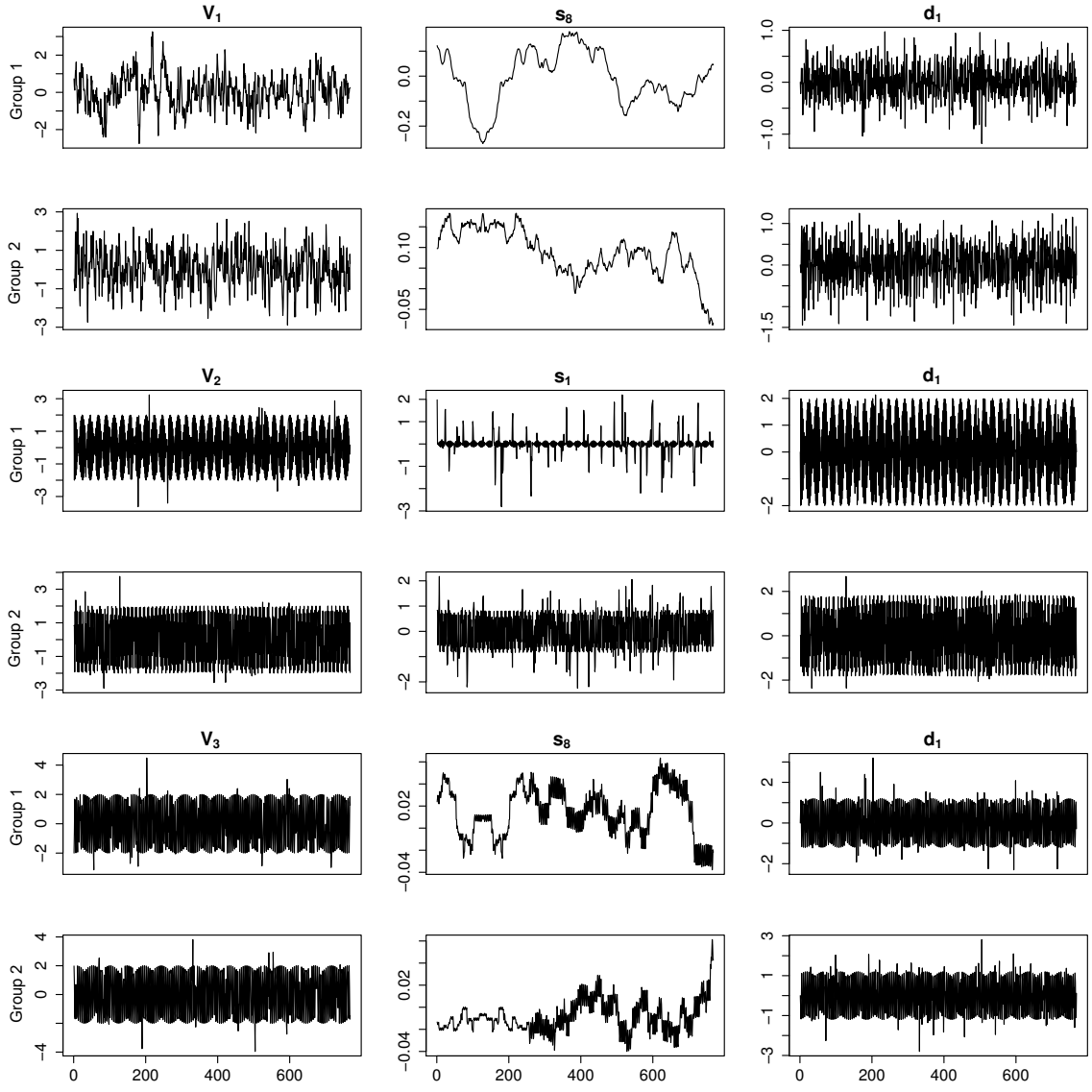


Figure 3.2: Wavelet transformed information for V_1–V_3 for individual one separately in Group 1 and Group 2 with 0.1 contamination rate and noise level 0, except V_1. Original variables have similar value range between two groups, but, after wavelet transform, the value range is no longer similar.

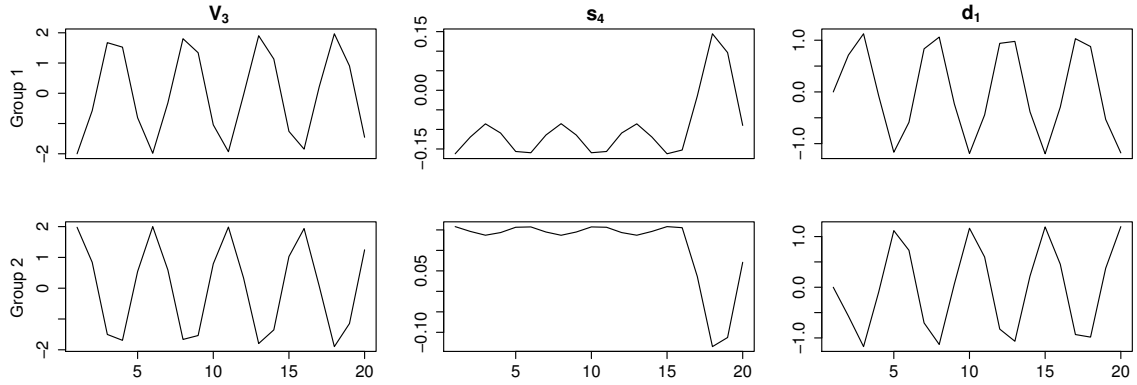


Figure 3.3: An example of V_3 for Group 1 and Group 2 with time length 20 and without noise added and contamination rate 0.

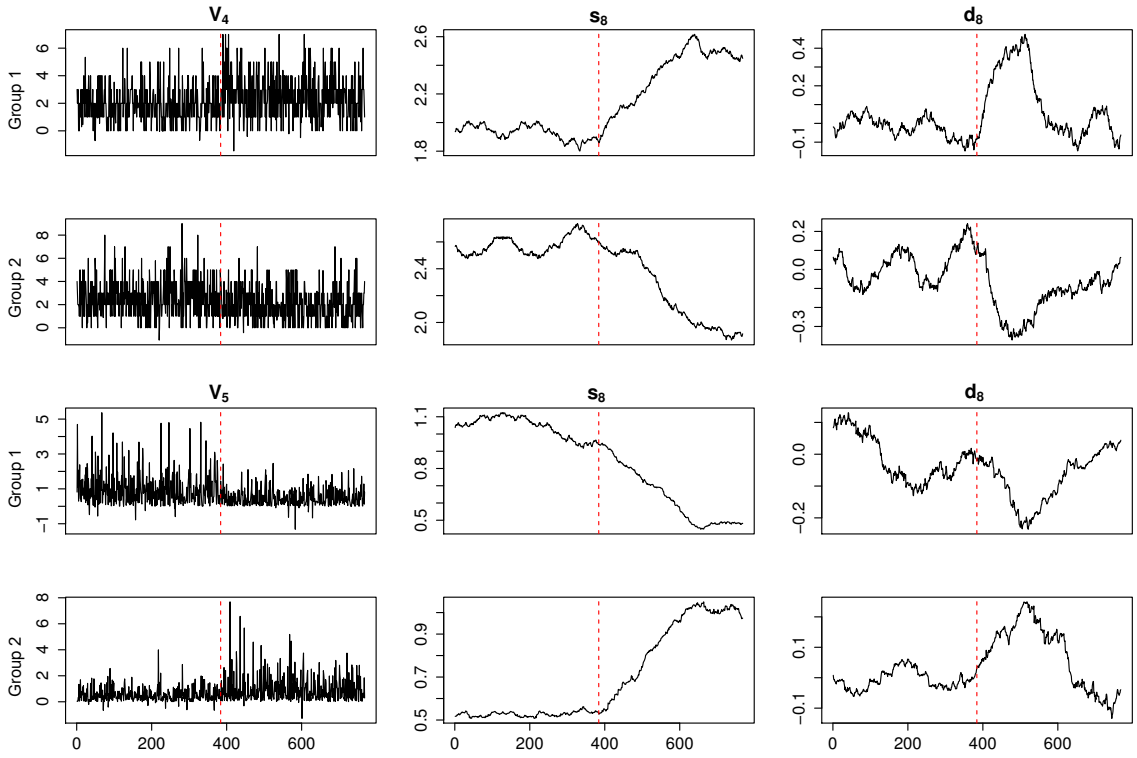


Figure 3.4: Wavelet transformed information for V_4–V_5 for first individual in Group 1 and Group 2 (alternating rows) with 0.1 contamination rate and noise level 0.

V_3 The sine function in the two groups differs only by a phase translation along the time axis. This is not easily recognized as it does not affect the auto-correlation or frequency characteristics which are encoded in the wavelet-transformed variables. Indeed, Methods 1 and 2 fail in this case. Method 3 still works here, but the reasons are subtle and the improvement is in fact an artefact caused by the data length T_n not being a multiple of the cycle length of the sine wave. This means that changing the starting point of the cycle results in one part of the cycle being slightly over-represented, illustrated in Figure 3.3. This effect, though small, is picked up when the s_8 and d_1 variables are averaged over the full time series. Since this effect will change as the relationship between the cycle length and T_n changes, we would not expect the good performance of Method 3 to be reproducible for variables like V_3 in general.

V_4, V_5 These variables have the same mixture when aggregated over time points as their distributions are symmetric between groups, but differ in which parts of the signal they are slightly higher and lower. Visually, the difference is clear in the s_8 variables which forms localized averages. This difference is lost when aggregated along the time axis, as happens with the original variables. However, the differences at level 8 are extremely helpful here as they record larger positive (negative) values when there is an increase (decrease) in the local mean. In addition, the taking of localized means effectively averages out the white noise.

These differences could be detected from the original variables, but care would be needed to compute a summary statistic that would encode the differences between groups, especially if the location of the change point T_n^s is not known. Although the examples in Figure 3.4 have $T_n^s = T_n/2$ for simplicity, the wavelet-transformed variables will detect the presence of an increase or decrease in the localized means adaptively regardless of the best scale to average over, or location of the change point.

3.3.3 Simulation with all explanatory variables included

Ideal circumstance

In ideal circumstances, noise level $\sigma_2 = 0$ is included, as well as a low contamination rate of $\theta = 0.1$ and equal group sizes. We also construct situations in which CART works for the original data, by adding an offset $\delta = 0.2$ to all observations of variables $V_1 - V_5$ in Group 1, making the expectation of these variables higher in Group 1. In that circumstance, we can check whether wavelet transformed variables can be better or similar to original variables in performance when the circumstance favours original variables.

With $\delta = 0$, Table 3.3 shows that using CART with the original data cannot distinguish the two groups using Methods 1 and 2, as it simply uses the default tree (classifying all individuals into the majority group as there is a big imbalance between the group size). It is a little better when using Method 3, with a prediction accuracy of 36.7/60 and detecting explanatory variables V_1, V_3, which are explanatory variables instead of redundant variables. However, using wavelet transformed data results in nearly 100% prediction accuracy. The explanatory variables and scales used are consistent with those in Figures 3.2–3.4. For Methods 1–2, V_2 is still the best explanatory variable, followed by V_5 (and V_1). For Method 3, V_1, V_5 and V_2 are all quite good.

When mean level is increased with $\delta = 0.2$, the original data and the wavelet-transformed data share identical accuracy. When explanatory variables have obvious mean-levels differences between two groups, wavelet-transformed variables have the same good performance as original data. In terms of variable choice, CART with original data generally chooses all the explanatory variables especially information of their means, and still selects redundant variables in some cases. However, CART with wavelet-transformed variables is more parsimonious while retaining excellent accuracy and is less likely to include redundant variables.

Practical circumstances

We now consider changes in noise level, contamination rate and group size balance. The corresponding results are shown in Figure 3.5. Each panel gives classification

Table 3.3: Accuracy results with noise level $\sigma_2 = 0$, contamination rate $\theta = 0.1$ and equal training group sizes of 150. Accuracy is the number of correctly classified individuals from a test set of 60, averaged over 100 replicate simulations. Entries – indicate that no splitting was done.

Method	Accuracy (/60, sd)		Original		Wavelet
	Original	Wavelet	Variables*	Redundant**	Variables*
Offset $\delta = 0$					
1	30.0(0.00)	59.7(0.64)	–	–	V_2 V_5 V_1
2	30.0(0.00)	60.0(0.00)	–	–	V_2 V_5
3	36.7(3.90)	59.5(0.83)	V_1 V_3 (m sd)	yes	V_1 V_5 V_2
Offset $\delta = 0.2$					
1	59.8(0.46)	59.6(0.79)	V_5–V_1	yes	V_2 V_3
2	60.0(0.00)	59.9(0.34)	V_4 V_5 (m sd)	no	V_2 V_5 V_3
3	60.0(0.10)	59.7(0.59)	V_1–V_5 (m)	no	V_5 V_2

* Main variables in the first six important variables from CART.
** Whether redundant variables are used by CART.
No redundant variables were selected by CART using wavelet variables.
The "m" and "sd" represent the mean and standard deviation.

Table 3.4: Choice of variable, resolution level and wavelet or scaling coefficient when applying CART to wavelet-transformed simulated data.

Method	Variables (information)		
Offset $\delta = 0$			
1	V_2 (s_1, s_2, d_2)	V_5 (s_8, d_8, s_7)	V_1 (s_8)
2	V_2 $(d_1, d_2, s_1, s_2$ m sd)	V_5 $(d_8, s_8$m)	
3	V_1 $(d_1, d_2$ sd)	V_5 $(d_7, d_8$ m)	V_2 $(d_1, d_2$ sd)
Offset $\delta = 0.2$			
1	V_2 $(s_6$–$s_8)$	V_3 $(s_6$–$s_8)$	
2	V_2 $(s_8$ m sd)	V_5 $(s_8$ m sd)	V_3 $(s_8$ m sd)
3	V_5 $(d_6$–d_8, s_8 m)	V_2 $(s_1, s_2$ m)	

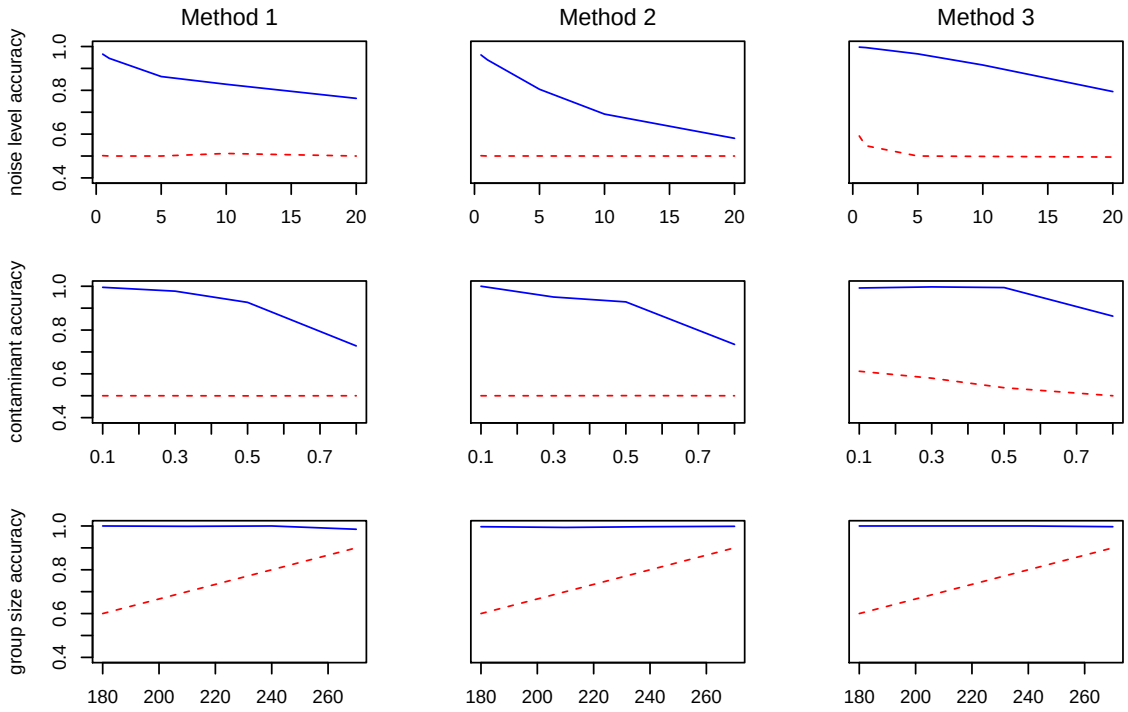


Figure 3.5: Testing accuracies results for real circumstances. Blue solid lines represent results for wavelet variables and red dashed lines represent that for the original variables. The x-axis for the first row is noise level, for the middle row is contamination rate, for the last row is Group 1 size.

accuracy as the proportion of correctly classified individuals for both wavelet-transformed and original data (solid and dashed lines respectively) under a range of circumstances.

As one would expect, increasing noise or contamination levels reduces the accuracy of the methods using wavelet-transformed data, with Method 3 being least affected since the aggregation over time points before classification effectively averages out noise in the wavelet coefficients before conducting the classification procedures. The classification accuracy obtained using the original data is broadly unchanged, since classification was already essentially arbitrary in this case by classifying the observations into the major group.

Making the groups more unbalanced in size leaves classification using wavelet-transformed data largely unchanged, while classification using the original data

appears to improve since predicting all individuals to be from the dominant group becomes an increasingly effective strategy.

3.3.4 Alternative wavelet basis functions

In the previous simulation, the Haar wavelet basis is used as defined in Equation (2.10). There are many other choices of wavelet basis available and we have repeated the analyses with minimum-bandwidth discrete-time wavelets with filter length 4 (mb4, see Morris & Peravali (1999)), Daubechies wavelets with filter length 4 (db4, see Daubechies (1992)) and Least Asymmetric wavelets with filter length 8 (la8, see Daubechies (1992)) in the R package waveslim. Full results are shown in Appendix A, and they show that Haar is generally better than the others in most circumstances. Compared with la8, Haar is almost always better. When comparing Haar with mb4 and d4, Haar is better when noise level and contaminant rate are low. When noise level and contaminant rate are high (5, 10, 20 and 0.5, 0.8 respectively), Haar is no longer the best in Methods 1 and 2 but still the best in Method 3. In most cases, Method 3 has the best performance. Therefore, due to good accuracy and the easier interpretation of the Haar wavelet, the Haar basis is recommended in practice.

3.4 Application to liver transplantation data

3.4.1 Data description and preprocessing

Liver Transplantation (LT) is a high-risk surgical treatment choice for patients suffering end-stage liver disease (Milan *et al.*, 2016). Pre-operative treatment, like beta-blockers, may help reduce the surgical risk to some extent while also influencing the chance of surgical complications. For example, systolic dysfunction and low cardiac output with beta-blockers may compromise renal perfusion (Chirapongsathorn *et al.*, 2016). So, if we can monitor variables like heart rate, systolic dysfunction and cardiac output effectively, then, adverse effects can be detected earlier. These explanatory variables will be applied to classify patients as using or not using beta-blockers. In practice, a patient's beta-blocker use is known before surgery, but here we classify patients' into beta-blocker use to investigate

Table 3.5: Monitoring variables recorded in the liver transplant (LT) dataset.

Abbreviation	Full name	Unit
CO	cardiac output	L/min
CI	cardiac index	L/min/m^2
SVR	systemic vascular resistance	dyne-s/cm^5
SVRI	systemic vascular resistance index	dyne-s/cm^5/m^2
Sys	systolic pressure	mm Hg
MAP	mean arterial pressure	mm Hg
Dia	diastolic pressure	mm Hg
SV	stroke volume	mL/beat
SVI	stroke volume index	mL/m^2/beat
HR	heart rate	beats/min

which monitoring variables are considered informative in the classification so as to explore the different behaviour of variables in different groups.

Data on patients undergoing LT between September 2004 and December 2011 at St James' University Hospital, Leeds, UK, was recorded using LIDCO monitoring equipment (LIDCO, Cambridge, UK). The ten intraoperative monitoring variables recorded are shown in Table 3.5; for more details, see Milan *et al.* (2016). After removal from the dataset of some individuals with poor-quality data, there are 90 patients who used beta-blocker (Group 1) and 236 patients who did not (Group 2). For each patient, the data consist of a multivariate time series of length one thousand to tens of thousands.

Since the number of patients in Group 2 is around 2.6 times that in Group 1, CART might be biased to predict all new patients as being from Group 2. This imbalance could be dealt with using a cost matrix, by discarding data from the larger group, or by sampling replicate data from the smaller group. The latter is chosen; after randomly sampling training and test data from the entire dataset, we triplicate the individuals from Group 1 for training and test data separately. There are then a total of 270 patients in Group 1, relatively in balance with 236 patients in Group 2. To investigate the robustness of the results to this procedure, the analysis is conducted twice, both with and without group size modification.

The data for each variable $A_{\cdot,k,\cdot}$ are shown in Figure 3.6. In order to see the

detail, for some variables, the y-axis limits are set to be smaller than the actual limits.

In Figure 3.6, There are some quite sharp increases and decreases across time, although variables like HR should not increase or decrease so suddenly as long as the patient is still alive. So the data required considerable cleaning before classification could be attempted. It is assumed that variables for each patient would not fluctuate sharply in a short time phase, and hence should remain within a limited range over a short time. Data points outside this range are regarded as outliers. The other outliers include a non-trivial number of missing or impossible values. Missing values can cause bias for the Gini index and other criteria as shown in Section 2.4.1. So they should be cleaned as well. Firstly, we deal with data values outside the range and name this as the *initial filter stage*. For the second stage, we will tackle data values which fluctuate sharply in a short time span and call this the *secondary filter stage*. The algorithm for these two stages is described below and summarised as Algorithm 2.

Initial filter Since there are outliers, robust statistics are applied for cleaning. Define

$$mad_{n,k} = med\left\{|A_{n,k,\cdot} - med(A_{n,k,\cdot})|\right\},$$

the median of the absolute deviations from the median. If $A_{n,k,t}$ is missing, infinite or zero, or satisfies

$$A_{n,k,t} \notin [med(A_{n,k,\cdot}) - 5mad_{n,k},\ med(A_{n,k,\cdot}) + 5mad_{n,k}],$$

then the last observation is carried forward and replaced by $A_{n,k,t}$ by $A_{n,k,t-1}$. 5 can be changed to other suitable values accordingly. The higher the value, the more proportion of the observations will be replaced, and the cleaner the data will be. But the main property of the data will be influenced when the value goes very high. So it depends on the requirement of the experiment. If $t - 1 = 0$, then we use the median of that variable. We use the previous value for replacement due to the presence of autocorrelation and since the previous value has already been defined as non-outlying.

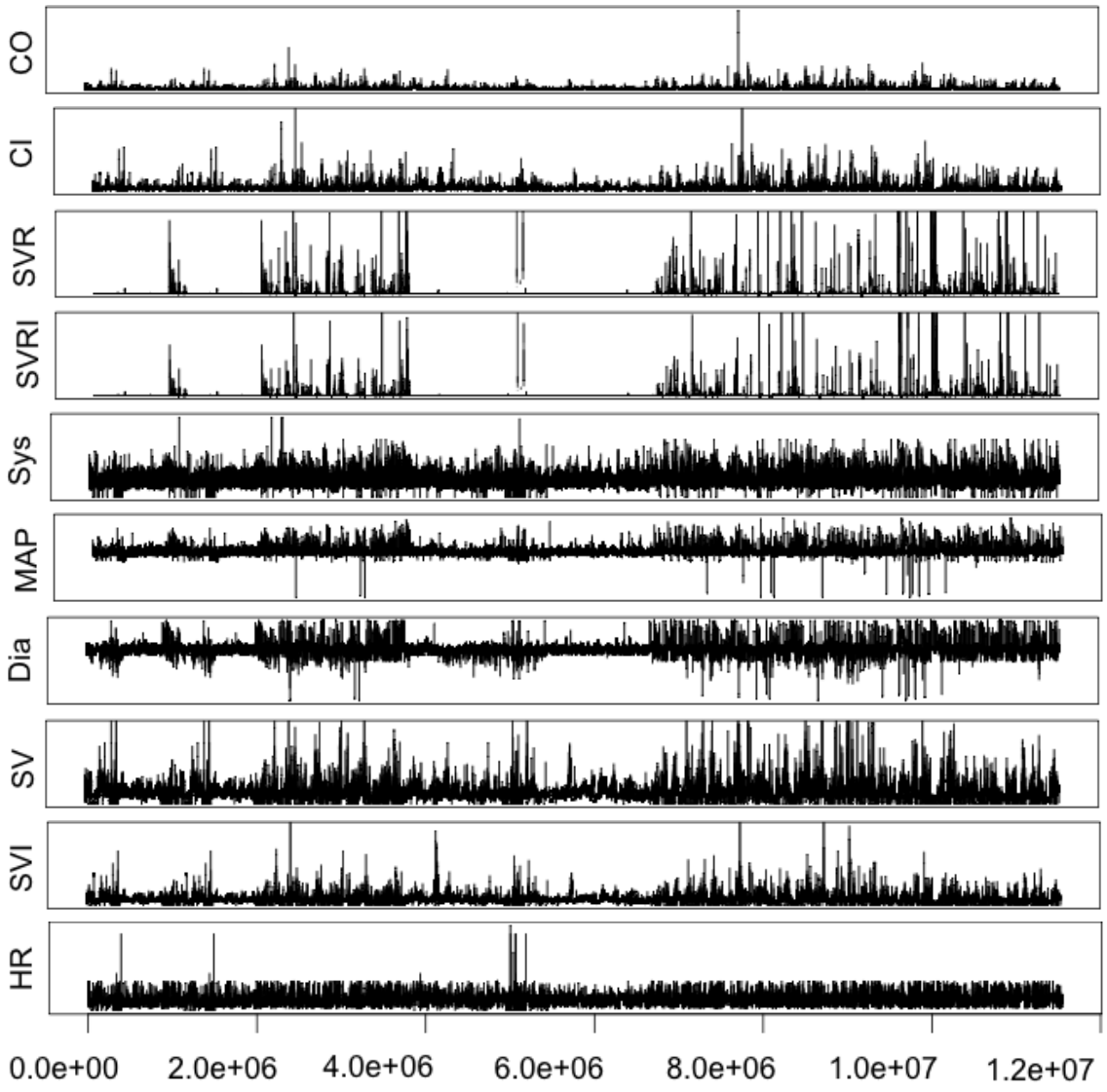


Figure 3.6: Original observations for all patients (stacked end to end). The time series have sharp increase and decrease which should not happen in reality. They will be regarded as outliers as well as missing values.

Algorithm 2 data preprocessing: smoothing

Require: *original data* : $A_{n,k,t}$

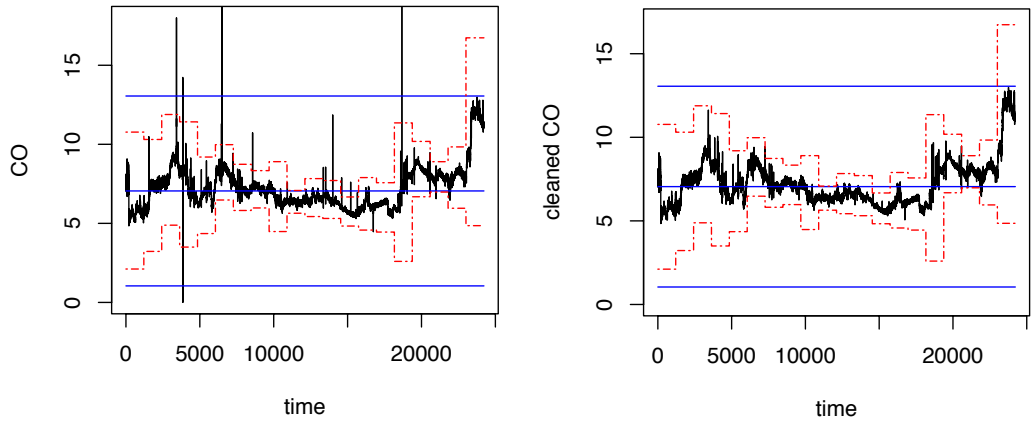
Ensure: smoothed data $A_{n,k,t}$

{ *number of patients* : $N,\ n = 1 : N$
number of observations for each n : $T_n,\ t = 1 : T_n$
number of variables : $10,\ k = 1 : 10$}
for k=1:1:10 **do**
for n=1:1:N **do**
$m_{n,k} = m(A_{n,k,.})$
$mad_{n,k} = mad(A_{n,k,.})$
for t=1:1:T **do**
if $A_{k,n,t} \notin [m_{n,k} - 5mad_{n,k},\ m_{n,k} + 5mad_{n,k}]$ or $A_{k,n,t}$ = NaN, Inf, 0, **then**
$A_{n,k,t} \leftarrow A_{n,k,t-1}$
end if
end for
$s = T_n/20$
for p=1:1:20 **do**
Q^1 =first decile of (A_{n,k,T_p})
Q^9 =ninth decile of (A_{n,k,T_p})
$d = Q^9 - Q^1$
for $t = t_{p,1} : 1 : t_{p,s}$ **do**
if $A_{k,n,t} \notin [Q^1 - 1.5d,\ Q^9 + 1.5d]$, **then**
$A_{n,k,t} \leftarrow A_{n,k,t-1}$
end if
end for
end for
end for
end for
return $A_{n,k,t}$

Secondary filter In the secondary filter stage, assumption is made that data values will not fluctuate sharply in a short time interval equal to 1/20 of the time series length. (Other time intervals were considered, but in practice for these data, intervals of $T_n/20$ worked well.) We define

$$Q^j_{n,k,p} = j^{th} \text{ decile of } \{A_{n,k,t_{i+1}}, A_{n,k,t_{i+2}}, \ldots, A_{n,k,t_{i+s}}\},$$

where $i = s(p-1)$, $s = T_n/20$ and $p = 1, 2, \ldots, 20$ refers to the p^{th} short time interval. With Q^1 and Q^9 as the first and ninth deciles and $d = Q^9 - Q^1$, we replace data values $A_{n,k,t} \notin [Q^1 - 1.5d,\ Q^9 + 1.5d]$ by $A_{n,k,t} = A_{n,k,t-1}$. If $t - 1 = 0$, then $A_{n,k,t} = Q^5$.



(a) Original CO (b) Cleaned CO

Figure 3.7: Example of data cleaning for CO data from patient 1. Solid horizontal lines represent median and initial filter range, dashed lines represent secondary filter range.

As an example, raw and cleaned CO data from patient 1 are shown in Figure 3.7. After initial filter, outliers above the 5 mad are changed to be the previous observation value. After secondary filter, outliers in detail are also changed into their previous observation value as shown in Algorithm 2.

The smoothed dataset is shown in Figure 3.8. Details of outliers are shown in Table 3.6. According to Table 3.6, nearly 0.3% to 2% of the observations are regarded as outliers, so outliers are replaced without changing the information in the observations a lot since the proportion is quite small. In most variables, the two groups have similar proportions of changes.

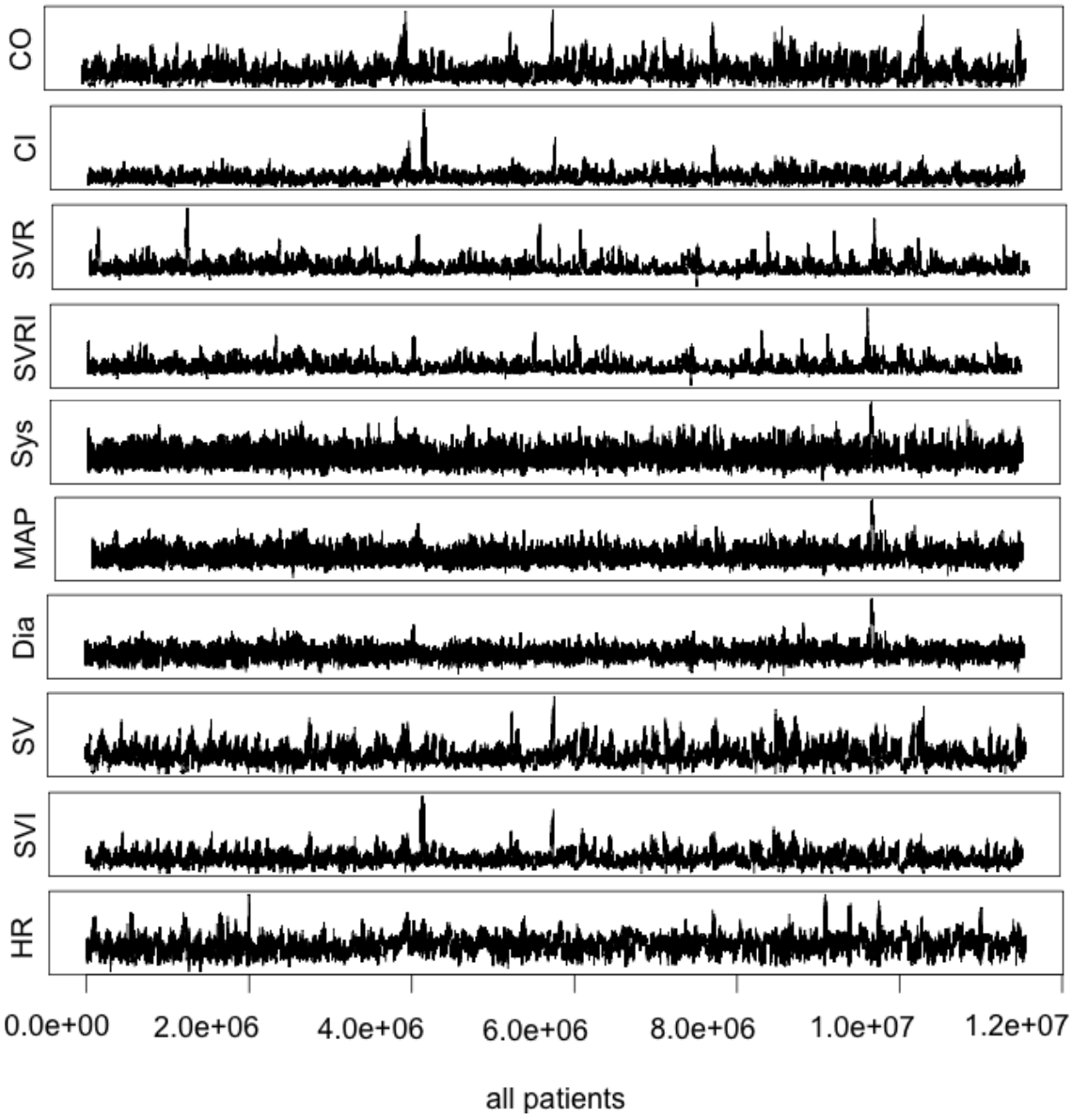


Figure 3.8: Smoothed data of all the patients. Sharp increase or decrease points have been replaced.

Table 3.6: Number of outliers for each variable.

		Variable									
Group		CO	CI	SVR	SVRI	Sys	MAP	Dia	SV	SVI	HR
1	No.	28963	21023	77144	41521	11667	17270	19985	38042	18390	57695
	%	0.77	0.56	2.04	1.10	0.31	0.46	0.53	1.01	0.49	1.53
2	No.	44227	43808	119197	119081	32335	38794	46189	49792	49509	178640
	%	0.57	0.56	1.53	1.53	0.42	0.50	0.59	0.64	0.64	2.30

3.4.2 Results

After data cleaning, the original and wavelet-transformed variables are applied to the classification methods, with 80% of the individuals randomly sampled for training and the remaining 20% for testing. To reduce sensitivity of observed classification accuracy to this sampling, we conducted 50 replicate trials for each method. The corresponding results are shown in Table 3.7.

Without group size adjustment, Methods 2 and 3 generally choose to split no variables and lead to the default tree. Method 1 does worse than this, sometimes choosing Group 1 as the main group as it builds trees on time-point level instead of directly on individual level. Method 2 also has such cases, so that is why their standard deviations are high. Without these cases, they generally choose the default tree with accuracy around 45 to 46 and standard deviation around 1 to 2. These confirm that group size adjustment is needed in this case.

After group size adjustment, individuals in Group 1 account for 53.4% of all the individuals, approximately in balance with individuals in Group 2. The student's t-test shows that there are no significant differences (with significant level 0.1) between the three methods either for wavelet-transformed or for original data. However, accuracy from the wavelet-transformed data is slightly higher in nearly all cases. This may be due to some variable means in different groups being sufficiently different for decision rules using the original data to work well. When we come across data that has no significant mean difference, the wavelet transform is highly recommended as its accuracy is then significantly higher. Size adjustment

initially seems to lower the accuracy, but it should be noted that with the balanced group sizes a default tree will now attain accuracy of 33/65. Without considering the time for wavelet transform (as both methods share the same time of wavelet transform for time points), the computation time for Method 2 (around 7 hours per trial) is even much longer than Method 3 (less than 1 second per trial). So, we find the best prediction of beta-blocker use to be achieved using wavelet-transformed data in Method 3 with a size adjustment.

The main variables chosen are HR, SV, CO, SVR, SVI, and CI. CART based on wavelet-transformed data generally chooses variables on resolution level 9 or 10 which are smoother, effectively choosing to use moving-average versions of the original data. In this case, some variables have different means between the two groups (Milan *et al.*, 2016), so CART based on original data works reasonably well, but the automatic smoothing of the variables via the wavelet transform improves the classification. It also gives us the added interpretation that the optimal smoothing is done over a time window of 2^9–2^{10} heartbeats, approximately 5–17 minutes. This finding has clinical relevance. One of the main effects of beta-blockers is a slowing heart rate. A previous study that compared heart rates among a group of patients — both treated and not treated with beta-blockers — found a difference between two large groups with more than 10,000 measurements for each patient (Milan *et al.*, 2016). Since clinical data during long surgical procedures are 'noisy', the complex statistics performed identified the need for data smoothing. Wavelet-transformed variables have shown improved interpretation via consideration of which resolution scales are the most informative. This method can be applied to other 'noisy' databases in the future.

3.5 Conclusions and discussion

Wavelets provide a basis for automatic feature extraction methods, allowing the classification technique (CART in our case) to select from localized means and differences over a range of scales. Compared to other feature extraction methods, the initial process is *not* dimension reduction. Feature extraction methods such as principal component analysis (Asavaskulkeit & Jitapunkul, 2009), locality sensitive hashing (Datar *et al.*, 2004), and manifold learning (Costa & Hero, 2004; Nie *et al.*,

Table 3.7: Testing accuracy results for the LT data using Methods 1–3 with and without group size adjustment. The main variables listed are the first six important variables list output by CART.

Method	Accuracy (/65, sd)		Main variables*	
	original	wavelet	original	wavelet
Without size adjustment				
1	39.4(11.63)	45.16(6.09)	HR CO SV SVR CI SVRI SVI	HR CO SV CI s_9 s_{10}
2	44.46(7.07)	45.56(5.76)	–	–
3	46.7(1.16)	46.7(1.20)	–	–
With size adjustment				
1	39.2(4.53)	40.4(4.78)	HR CO SV SVR CI SVI	HR SV CO s_9 s_{10}
2	40.34(3.82)	41.94(6.32)	HR SV SVI m, sd	HRs_9 s_{10} (m, sd) SVIs_{10} (m, sd) Syss_{10} (m, sd) SVs_{10} (m, sd) Dias_{10} (m, sd)
3	38.82(2.86)	39.84(3.69)	HR m, SVR m, CI m, CO m, MAP m, SVI m, SVRI m	HR s_1–s_8 m

2010), all aim to reduce the number of explanatory variables. Using wavelet-transformed variables actually *increases* the dimension by transforming original variable into detail and smoothed coefficients on different resolution levels. This can reveal hidden information which is not easy for classification trees to find using only the original data. This does mean that the wavelet transformation of the data is more suitable for experiments without an excessive number of predictors, otherwise a further variable reduction step will be required and this will increase the computational burden (Chitaliya & Trivedi, 2010; Li & Wen, 2014; Mazloom & Ayat, 2008). This also makes a solid recommendation for us to use wavelet transformed variables in Chapter 4 and Chapter 5, for future analysis of time series.

CART, as a decision tree method, can be seen as a variable reduction method

since it chooses the "best" variable to split on in each step. Compared to methods like ANN (Rowley *et al.*, 1998), SVM (You *et al.*, 2014) and LASSO (Roberts & Nowak, 2014), its main advantage is its ease of interpretation, although it might not achieve the same accuracy as other methods. When applying CART to wavelet-transformed data, the disadvantages (like dimension increase) of using the wavelet transform are mitigated since CART carries out a dimension reduction function. By learning which wavelet-transformed variables are more effective, we can also gain the added interpretation of which scales are important.

Compared to other feature extraction methods, the wavelet transform has its own advantages. It helps discover information hidden by noise that can not be achieved by other methods which do not provide information decomposition across different scales for one single variable. In our simulation, we have shown the effectiveness of wavelet-transformed data in CART classification where the key features of interest are changes in autocorrelation or frequency structures (our variables V_1 and V_2), or relatively small changes in mean level which occur at unknown times and are hidden by considerable noise (V_4 and V_5).

The scaling function we use in the wavelet decomposition is the Haar wavelet, the simplest case of the compactly-supported wavelets described by Daubechies (1992). In our experience, the Haar wavelet tends to have equal or better accuracy than other choices of wavelet and has the benefit of easier interpretation. For data whose expectation and variance have some connection, such as the Poisson and exponentially distributed V_4 and V_5, we might consider using the Haar-Fisz wavelet transform (Fryzlewicz & Nason, 2004). Since, in real situations, we will usually not know the distribution of the time series, we generally use the Haar wavelet which is a robust all-purpose selection that allows for easy interpretation compared to more complicated wavelet bases.

We set out to produce individual-level predictions from panel data, where simple application of CART produces time-point level predictions. Comparison of the different methods we used has shown that methods which perform at least some aggregation before prediction have improved performance over a naive approach of predicting at each time point and using a simple voting mechanism to aggregate these predictions. Additionally, aggregation before classification (Method 3)

is computationally fast in comparison to Methods 1 and 2. So, overall, we recommend Method 3 with wavelet transformed data.

Chapter 4

Wavelets and CART for static time series forecasting

4.1 Introduction

Chapter 3 has shown that wavelet transformed variables can be better than original variables for classification. This chapter will explore whether or not wavelet transformed variables are still better than original variables for forecasting using regression methods. Explorations include static time series in this chapter and will be extended to dynamic time series forecasting in Chapter 5. We explore the performance using simulated autoregressive (AR) time series and real data including the LT data shown in Chapter 3 and Chinese air pollution data. The simulation is done under different seasonal effect levels and forecasting length levels. The reason for using AR based time series is that the heart rate data in the real data application has autoregressive properties. It is easy to understand that the current heart rate is highly correlated with the previous ones in reality and there is some periodic property as well. That is why AR is used in the simulation with some seasonal effect added.

4.2 Simulation

In this section, we generate an AR time series following

$$y_t^{raw} = a_1 \cdot y_{t-1}^{raw} + a_2 \cdot y_{t-2}^{raw} + a_3 \cdot y_{t-3}^{raw} + \cdots + a_{12} \cdot y_{t-12}^{raw} + \epsilon_t, \tag{4.1}$$

where $\epsilon_t \sim N(0, \sigma^2)$, $\sigma = 1$, $[a_1, \ldots, a_{12}] = [0, 0, 0.1, 0.8, 0, 0, 0, 0, 0, -0.1, 0.1, -0.5]$, and $t = 1, 2, 3, \ldots, 10000$. In the model for y_t^{raw}, the parameters were chosen to ensure a stationary time series as the heart rate value will not always have decrease or increase trend and both short and long time lag effects are included. In addition, to make the simulated time series y_t^{raw} more realistic (like the heart rate), we add a periodic effect (seasonal effect) to it. Assuming that this time series is collected daily, and there is a sine shape change around the year (365 days), the seasonal effect function is

$$y_t^{season} = \alpha \sin(t' \cdot 2\pi/365), t' = t \bmod 365,\ t = 1, 2, \ldots 365.$$

The seasonal effect y_t^{season} is added to y_t^{raw} to obtain y_t with seasonal effect. An example is shown in Figure 4.1 for one simple realisation. After generating y_t, we get the wavelet transformed data $W^{T\times 16}$ using MODWT for level $j = 1, 2, \ldots, 8$, where T is the number of observations. For details of the wavelet transform, see Chapter 2. There are 8 smooth variables $s_1, s_2, \cdots s_8$ and 8 detail variables $d_1, d_2, \cdots, d_8$, making a total of 16 variables.

$$W = \left[\begin{array}{ccc|ccc} W_1^{d_1} & \cdots & W_1^{d_8} & W_1^{s_1} & \cdots & W_1^{s_8} \\ W_2^{d_1} & \cdots & W_2^{d_8} & W_2^{s_1} & \cdots & W_2^{s_8} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ W_T^{d_1} & \cdots & W_T^{d_8} & W_T^{s_1} & \cdots & W_T^{s_8} \end{array}\right].$$

After that, the time lag variables are generated from y_t and $W_t^{T\times 16}$. In that case, we get our Y_{lag} and W_{lag} (lagged MODWT of Y) variables. Since Y_{lag} is actually lagged s_0, Y_{lag} is included in the wavelet transformed variables but still use Y_{lag} as the variable name. Here t is the current time for forecasting.

$$Y_{lag} = \left[\begin{array}{cccc} y_{t-1} & y_{t-2} & \cdots & y_{t-k.lag} \end{array}\right]_{(T-k.lag)\times k.lag}$$

$$W_{lag} = \left[\begin{array}{ccccc} Y_{lag} & W_{t-1} & W_{t-2} & \cdots & W_{t-k.lag} \end{array}\right]_{(T-k.lag)\times 17\cdot k.lag}$$

So W_{lag} (including original variable Y_{lag}) is used to predict Y. To compare the performance of original variables and wavelet transformed variables, Y_{lag}, W_{lag} are applied separately into CART, and use previous y_t, up to lag $k.lag$ for forecasting the current value (forecast horizon $p = 0$ here) with and without wavelet transform:

$$\hat{y}_{(t+p)} = f(y_{t-1}, y_{t-2}, \cdots, y_{t-k.lag}),$$

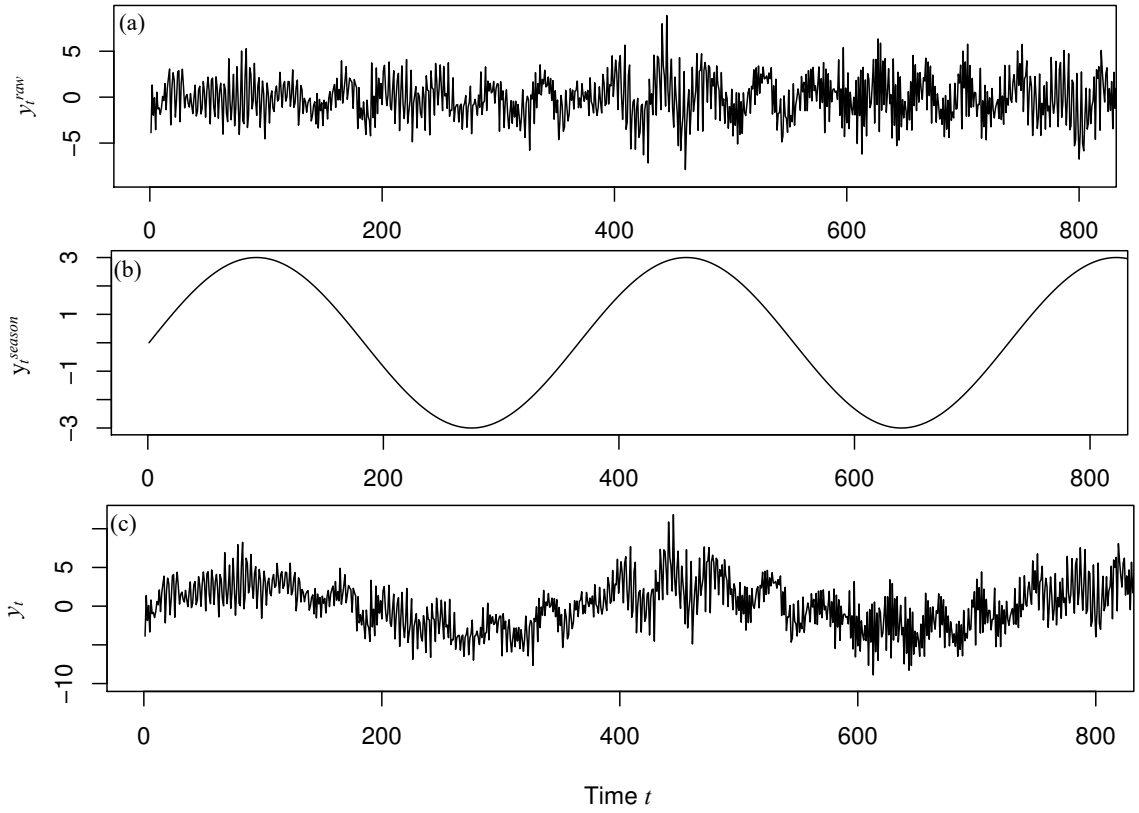


Figure 4.1: Time series data generating process. Only 800 observations are shown for illustration. (a) is the raw time series without seasonal effect. (b) is the seasonal effect with $\alpha = 3$. (c) is the time series with seasonal effect added.

where the function is estimated using training data and $\hat{y}_t$ is forecast by applying test data to this estimated function (trained model).

The accuracy of the results is measured by R-squared:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}},$$

where $SS_{res} = \sum_t (y_t - \hat{y}_t)^2$ and $SS_{tot} = \sum_t (y_t - \bar{y})^2$, in which $\hat{y}_t$ is the fitted value of y_t.

4.3 Simulation results under different seasonal effect levels

Since here y_t is generated from a known distribution with the true maximum lag 12, a long time lag. The seasonal effect in y_t means longer lags are also informative, but it might be too long to include in the models. Sometimes, when computational efficiency is important, we have to use short lag information like lag 4 or even no lag information. In that case, models under lag 12, lag 8, lag 4 and lag 1 are trained with 8000 observations and with the remaining 2000 observations used for testing.

Different seasonal effect levels α lead to different results. For each experiment, 50 trials (50 simulated data) are conducted and the averaged R^2 results are shown in Figure 4.2 and Table 4.1.

As seasonal effect level increases, R-squared increases as well. This is because, with higher seasonal effect level, the time series trend becomes more clear. In addition, the more lag variables in use, the higher the R-squared.

From Table 4.1, it is clear when only lag 1 ($k.lag = 1$) is permitted, MODWT based CART chooses to use s_2 first, as it contains information on y_{t-4}. When lag 4 and higher lags are used, CART chooses variables with lags 4, 8, 7, 11 and 12 which contain information about the true lags as the parameter values shown in Equation 4.1. When the seasonal effect level increases, CART tends to choose relatively smoother variables like s_2 and s_1. This is because, with a higher seasonal effect level, the sine pattern becomes the main trend compared to the AR pattern and relatively smoother variables can weaken the AR pattern but also can keep the sine trend.

Table 4.1: Simulation results on choice of variables for different seasonal effect levels α when comparing original variables and wavelet variables with different possible lags. The information is averaged from 50 trials.

	α	Original*	Wavelet*					
lag1	0	y.lag.1	d2.lag1	d4.lag1	s1.lag1	s2.lag1	d3.lag1	y.lag.1
	2	y.lag.1	s2.lag1	s3.lag1	s1.lag1	s4.lag1	s5.lag1	s6.lag1
	4	y.lag.1	s2.lag1	s3.lag1	s1.lag1	s4.lag1	s5.lag1	y.lag.1
	6	y.lag.1	s2.lag1	s3.lag1	s1.lag1	s4.lag1	y.lag.1	s5.lag1
	8	y.lag.1	s2.lag1	s3.lag1	s1.lag1	y.lag.1	s4.lag1	s5.lag1
	10	y.lag.1	s2.lag1	s3.lag1	s1.lag1	y.lag.1	s4.lag1	s5.lag1
lag4	0	y.lag.4,1,2,3	y.lag.4	d1.lag3	d1.lag4	s1.lag3	s1.lag4	d2.lag2
	2	y.lag.4,1,2,3	y.lag.4	s1.lag3	s1.lag4	d1.lag3	s2.lag1	s2.lag4
	4	y.lag.4,1,2,3	y.lag.4	s1.lag3	s1.lag4	s2.lag4	s2.lag1	s3.lag1
	6	y.lag.4,1,2,3	y.lag.4	s1.lag3	s1.lag4	s2.lag4	s2.lag1	s3.lag1
	8	y.lag.4,1,2,3	s1.lag3	s2.lag1	s3.lag1	y.lag.4	s1.lag4	s2.lag2
	10	y.lag.4,1,2,3	s2.lag1	s1.lag3	s3.lag1	s2.lag2	s1.lag1	s1.lag2
lag8	0	y.lag.4,8,5,3,1,7	y.lag.4	d1.lag3	y.lag.8	d1.lag4	d1.lag8	d2.lag7
	2	y.lag.4,8,7,1,2,6	y.lag.4	y.lag.8	s1.lag3	s1.lag7	s1.lag4	d1.lag4
	4	y.lag.4,8,7,1,2,6	y.lag.4	s1.lag3	y.lag.8	s1.lag4	s1.lag7	s2.lag4
	6	y.lag.4,8,7,1,2,6	y.lag.4	s1.lag3	s1.lag4	y.lag.8	s1.lag7	s2.lag4
	8	y.lag.4,8,7,1,2,6	s1.lag3	s2.lag1	s2.lag4	s2.lag3	s3.lag1	s2.lag2
	10	y.lag.4,8,7,1,2,6	s1.lag3	s2.lag1	s2.lag2	s3.lag1	s2.lag3	s2.lag4
lag12	0	y.lag.4,8,12,11,9,3	y.lag.4	y.lag.8	d1.lag3	d1.lag4	d1.lag11	d1.lag8
	2	y.lag.4,8,11,12,7,1	y.lag.4	y.lag.8	s1.lag3	s1.lag7	s1.lag4	s2.lag4
	4	y.lag.4,8,11,1,7,2	y.lag.4	s1.lag3	y.lag.8	s1.lag4	s1.lag7	s2.lag1
	6	y.lag.4,8,7,1,2,6	y.lag.4	s1.lag3	s1.lag7	s1.lag4	s2.lag4	y.lag.8
	8	y.lag.4,8,7,1,2,6	s1.lag3	s2.lag1	s2.lag3	s3.lag1	s2.lag2	s2.lag4
	10	y.lag.4,8,1,7,2,6	s1.lag3	s2.lag1	s2.lag3	s3.lag1	s2.lag2	s2.lag4

* variables given by the variable importance list in the model (same on other tables).

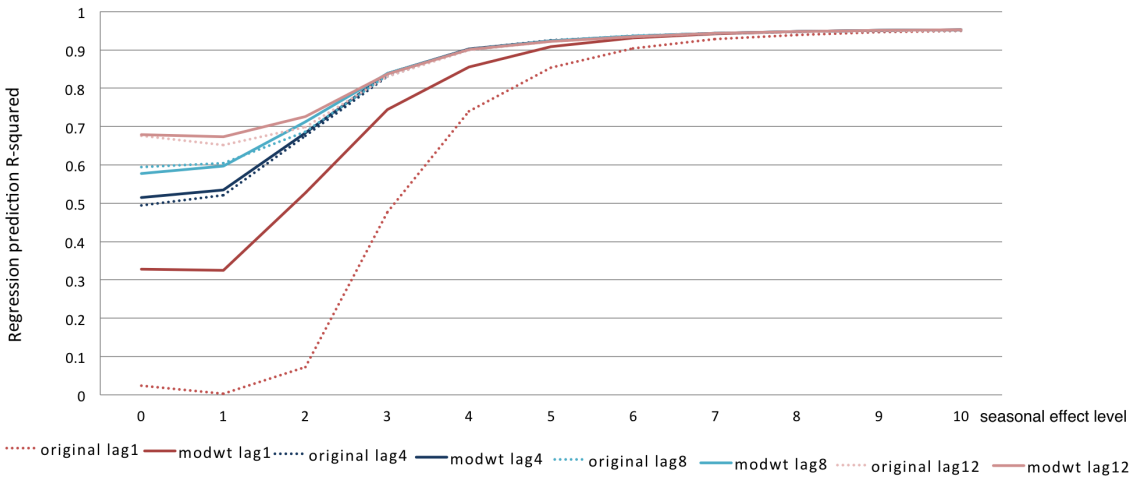


Figure 4.2: Accuracies for different time lag and seasonal effects using simulated data. Solid lines represent results of MODWT models and dashed lines represent results of original models.

4.4 Simulation results under different forecast ahead length levels.

In this section, we change the forecast horizon p from one observation ahead to 99 observations ahead in steps of 2, keeping $k.lag$ at 1, 4, 8 and 12. Here the seasonal effect level α is chosen as 3. By applying the simulated data into the model CART, 50 trials are conducted in this experiment.

$$\hat{y}_{(t+p)} = f(y_{t-1}, y_{t-2}, \cdots, y_{t-k.lag}).$$

Results in Figure 4.3 show that, as p increases, the R-squared of original data based models decreases until it is close to zero, while that of MODWT decreases much slowly to a lower limit of around 0.4. There is some big periodic fluctuation for original and wavelet results. This is because the interaction between forecasting horizon and permitted lags ($k.lag$) in explanatory variable might be around the true model lags 3, 4, 10, 11, 12. Another reason is the seasonal effect. For example, we have p as 2 and $k.lag$ as 1, so it is exact lag 3, which allows a good forecasting. Even so, MODWT based forecasts do better in terms of R-squared.

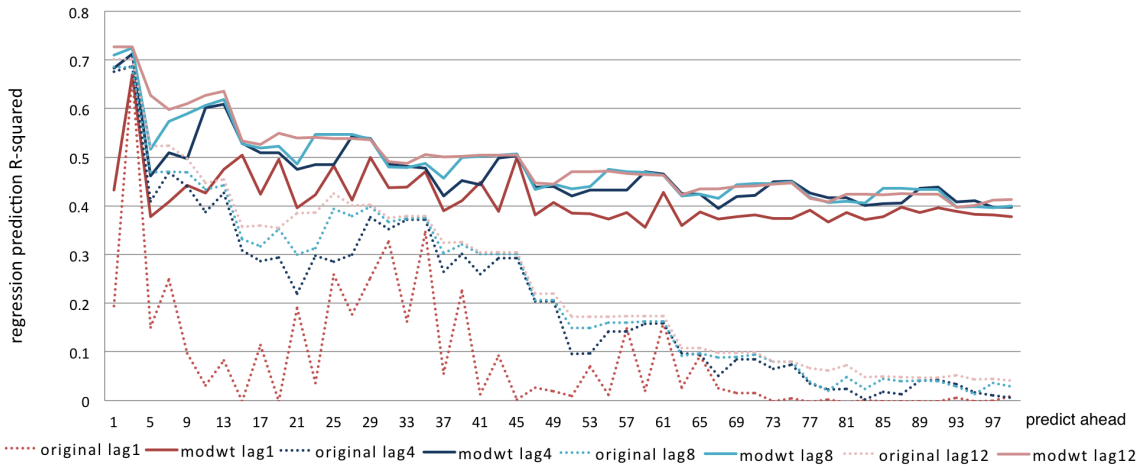


Figure 4.3: Simulation averaged accuracy results on different forecast length levels. Solid lines represent results of MODWT models. Dashed lines represent results of original models.

The variable importance lists are shown in Table 4.2. It is clear that when more long-lag variables are permitted, CART has a better chance to choose more true lag variables. When p increases, MODWT based CART uses more coarse scale variables which contain long memory information. Besides, for some p values, detail variables are more important, in which the frequency of the sine function plays a major role.

The simulation section has shown that forecasts based on wavelet transformed variables can be better than those using the original variables under different conditions. We now apply the wavelet-based forecasting to real data, including the LT data and air pollution data.

4.5 Application to LT data

As described in Chapter 1 and Chapter 3, the LT data contains 326 individuals. Each individual is regarded as one trial, making a total of 326 parallel trials in this experiment. Since it is an autoregression problem, instead of using all ten variables (other variables can also be used), the variable we use is heart rate (HR). So for each individual, the heart rate data are regarded as one time series. For

Table 4.2: Simulation results on choice of variables for different forecast horizon.

	p	original*	wavelet*					
lag1	0	y.lag.1	s2.lag1	s3.lag1	s1.lag1	s4.lag1	s5.lag1	d7.lag1
	2	y.lag.1	s1.lag1	s2.lag1	s3.lag1	y.lag.1	s4.lag1	s5.lag1
	4	y.lag.1	s2.lag1	s3.lag1	s1.lag1	s4.lag1	s5.lag1	d7.lag1
	8	y.lag.1	d7.lag1	s4.lag1	s3.lag1	d6.lag1	s2.lag1	s1.lag1
	20	y.lag.1	d7.lag1	d6.lag1	s2.lag1	s1.lag1	s3.lag1	s4.lag1
	50	y.lag.1	d6.lag1	d7.lag1	s2.lag1	s1.lag1	s3.lag1	y.lag.1
	100	y.lag.1	s7.lag1	d8.lag1	s6.lag1	s5.lag1	s4.lag1	s3.lag1
lag4	0	y.lag.4,2,1,3	y.lag.4	s1.lag3	s1.lag4	s2.lag1	s2.lag4	s3.lag4
	2	y.lag.2,4,3,1	y.lag.2	s1.lag1	s1.lag2	s2.lag2	s3.lag2	s2.lag1
	4	y.lag.3,4,1,2	s1.lag3	s2.lag3	s2.lag1	s2.lag4	s3.lag3	s3.lag1
	8	y.lag.3,1,2,4	y.lag.3	s1.lag2	s1.lag3	s2.lag3	s3.lag3	s2.lag2
	20	y.lag.2,1,4,3	d7.lag4	d7.lag3	d7.lag2	d7.lag1	s8.lag3	s8.lag2
	50	y.lag.1,3,4,2	s8.lag3	s8.lag2	s8.lag1	d6.lag4	d6.lag3	d6.lag2
	100	y.lag.4,2,1,3	s7.lag4	s7.lag3	s7.lag2	s7.lag1	d8.lag4	d8.lag3
lag8	0	y.lag.4,8,7,6,2,1	y.lag.4	y.lag.8	s1.lag3	s1.lag4	s1.lag7	s2.lag1
	2	y.lag.2,6,5,4,8,1	y.lag.2	y.lag.6	s1.lag1	s1.lag2	s1.lag5	s2.lag2
	4	y.lag.7,3,5,4,1,8	s1.lag3	s1.lag7	s2.lag1	s2.lag3	s2.lag4	s3.lag1
	8	y.lag.3,7,5,1,6,4	y.lag.3	y.lag.7	s1.lag3	s1.lag2	s1.lag6	s2.lag3
	20	y.lag.6,2,8,5,3,4	d7.lag5	d7.lag6	d7.lag4	d7.lag7	d7.lag3	d7.lag8
	50	y.lag.1,5,8,3,4,7	s8.lag7	s8.lag6	s8.lag5	s8.lag4	s8.lag3	s8.lag2
	100	y.lag.8,4,1,6,5,2	s7.lag8	s7.lag7	s7.lag6	s7.lag5	s7.lag4	s7.lag3
lag12	0	y.lag.4,8,11,7,12,1	y.lag.4	y.lag.8	s1.lag3	s1.lag4	s1.lag7	s2.lag4
	2	y.lag.2,6,9,5,10,4	y.lag.2	y.lag.6	s1.lag1	s1.lag2	s1.lag5	s2.lag2
	4	y.lag.7,11,3,4,10,5	s1.lag3	s2.lag3	s1.lag7	s2.lag1	s3.lag1	s2.lag4
	8	y.lag.3,7,10,11,6,1	y.lag.3	y.lag.7	s1.lag3	s1.lag2	s1.lag6	s2.lag3
	20	y.lag.12,8,1,6,5,4	d7.lag7	d7.lag8	d7.lag6	d7.lag9	d7.lag10	d7.lag11
	50	y.lag.1,5,12,8,9,4	s8.lag4	s8.lag5	s8.lag6	s8.lag2	s8.lag1	s8.lag3
	100	y.lag4,10,6,8,2,11	s7.lag11	s7.lag10	s7.lag9	s7.lag8	s7.lag12	s7.lag7

each individual, 80% of the time series are used for training and the remaining 20% for testing. To compare the performance of wavelet transformed variables and original variables, instead of using the ratio of R-squared, we use the ratio of SS_{res} of the two models:

$$\text{ratio} = \frac{\text{original } SS_{res}}{\text{wavelet } SS_{res}}.$$

The ratio of R-squared is actually the ratio of SS_{reg}, which is not as meaningful as the ratio of SS_{res}. SS_{reg} measures how far away the predicted values are from the averaged true values while SS_{res} measures the distance between predicted values and true values, the latter one of which is more direct. If the SS_{res} values for original data is higher than that for MODWT data, then MODWT based CART is better than using original data in that trial. This experiment is conducted under different situations and average the results over 326 individuals. The results in

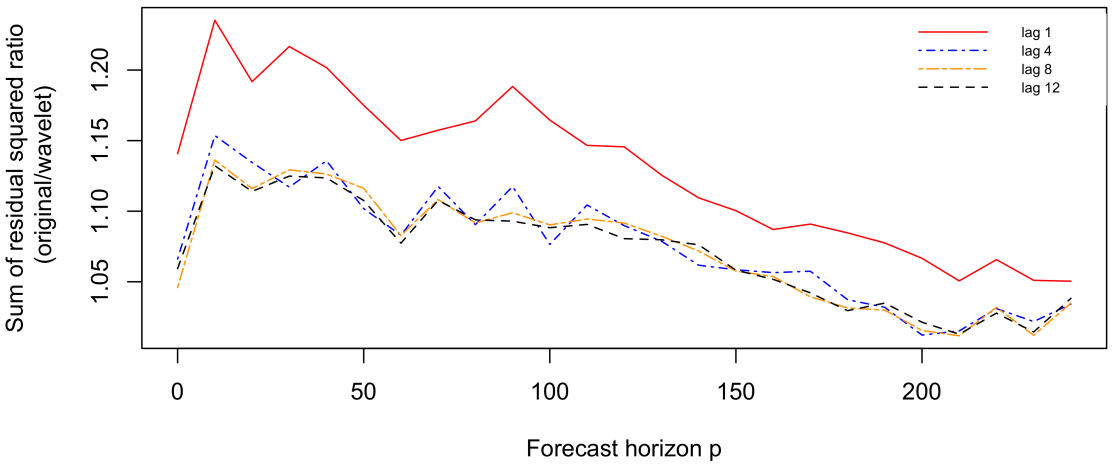


Figure 4.4: LT Prediction results using regression trees with original data and wavelet transformed data.

Figure 4.4 show that wavelet transformed data perform much better than original data when forecast length p is not high (like those before 100). As p increases, the performance of wavelet transformed data is approaching that of original data, but still better. When the permit time lag $k.lag$ is only 1, the wavelet transformed data

Table 4.3: LT data results on choice of variables for different lag levels.

	p	original*	wavelet*
lag 1	60	y.lag.1	s5, s6, s3, s2, s1
	120	y.lag.1	s5, s4, s6, s3, s2, s1
	180	y.lag.1	s5, s6, s4, s3, s2, s1
	240	y.lag.1	s4, s3, s5, s2, s6, s1
lag 4	60	y.lag.1,2,3,4	s4 lag4,3; s5 lag1-4
	120	y.lag.1,2,3,4	s4 lag4,3; s5 lag1,4,2,3
	180	y.lag.1,2,3,4	s5 lag4,3,1,2; s4 lag4,3
	240	y.lag.1,2,3,4	s4 lag1-4; s3 lag3,4
lag 8	60	y.lag.3,2,4,5,6,1	s5 lag1-6
	120	y.lag.1,3,2,4,8,7	s5 lag1-6
	180	y.lag.2,6,3,5,4,7	s5 lag1-6
	240	y.lag.1,2,3,4,5,6	s4 lag1-4; s3 lag3,4
lag 12	60	y.lag.3,2,4,5,6,1	s5 lag1-6
	120	y.lag.1,2,4,3,8,7	s5 lag1-6
	180	y.lag.6,11,10,12,8,9	s5 lag1-6
	240	y.lag.1,2,3,4,5,6	s4 lag1-4; s3 lag3,4

is obviously better than original data which has only one variable y_{t-1}, whereas MODWT based CART contains long time information.

From Table 4.3, short-term lag information is more important in forecasting which means the current HR value is more correlated with the most recent values. Also, coarser scale variables are more useful as they contain less noise and long time range information.

4.6 Application to air pollution data

Air pollution is quite a serious problem in China. People get diseases due to unhealthy air quality or even hazardous air. An Air Quality Index (AQI) is a generalised comprehensive way to describe air quality, which is based on the level of 6 atmospheric pollutants: sulfur dioxide (SO2), nitrogen dioxide (NO2), suspended particulates (PM10, PM2.5), carbon monoxide (CO), and ozone (O3) measured at monitoring stations throughout each city (Gupta *et al.*, 2006; Gurjar *et al.*, 2008). If AQI could be forecast ahead by, say one month, then government and people would have enough time to prepare at least some response measures. This is the time series forecasting problem we consider in this section.

The AQI dataset collected comes from the China Air Quality Online Monitoring and Analysis Platform (https://www.aqistudy.cn/) which summaries the information from data centre of the Ministry of Environmental Protection of the People's Republic of China (http://www.mep.gov.cn/). Since the data are shown in maps, which are not available for downloading directly, we collect the data manually. It is a monthly dataset (averaged from daily data) of 31 provinces in China (except Hongkong, Macao and Taiwan) from December 2013 to April 2017, making a total of 41 months. Monthly data are used in the analysis for prediction. But if more precise prediction is required and time permitted, daily or even hourly data can also be collected. The data can be described as

$$A^T = [A^T_{.,1}, A^T_{.,2}, \cdots, A^T_{.,T}]$$

and $A_{.,t} = [a_{1,t}, a_{2,t}, \cdots, a_{N,t}]^T$, where $t = 1, 2, \ldots, T$ and $n = 1, 2, \ldots, N$ with $T = 41$ and $N = 31$.

Since air pollution has spatial spillover effects (air pollutants are apt to diffuse and migrate across different regions), one province's AQI can also be influenced by that from neighbouring provinces. So we use another variable to describe these spillover effects. For detail of spatial regression analysis, see Ward & Gleditsch (2018). There are many ways to construct a spatial weight matrix. In this section, it is assumed that only geographically contiguous provinces share spatial influence. In this scenario, we have an adjacency matrix, B, with elements

$$B_{i,j} = \begin{cases} 1 & \text{if provinces } i \text{ and } j \text{ are neighbours } (i, j \in \{1, 2, \ldots, N\}) \\ 0 & \text{otherwise including } i = j. \end{cases}$$

Then, a standardised matrix, B is constructed by making the sum of each column equal to 1. So the elements of the standardised B matrix are

$$B^s_{i,j} = B_{i,j} / \sum_k B_{i,k}, \;\; i, j \in 1, \ldots, N$$

So, the spatial spillover effects become

$$(A^{SW}_{.,t})^T = A^T_{.,t} B^s,$$

which measures the spillover effect from all the neighbouring provinces to the current province and the overall spatial weighted variable is

$$(A^{SW})^T = [(A^{SW}_{.,1})^T, (A^{SW}_{.,2})^T, \cdots, (A^{SW}_{.,T})^T].$$

In order to detect whether wavelet transformed variables have a better performance in forecasting, we apply MODWT to both $A_{n,.}$ and $A^{SW}_{n,.}$ for each separate province, n at level j. (Since the time length T is 41, so the maximum level j is 5 ($2^5 = 32$), $j = 1, 2, \ldots, 5$.) The wavelet transformed data $MODWT(A_{n,.})$ for province n are denoted as the $T \times 2J$ matrix $W_{n,.}$ (J is the maximum level of wavelet transform in this section), where

$$W_{n,.} = \left[\begin{array}{ccc|ccc} W^{d_1}_{n,1} & \cdots & W^{d_J}_{n,1} & W^{s_1}_{n,1} & \cdots & W^{s_J}_{n,1} \\ W^{d_1}_{n,2} & \cdots & W^{d_J}_{n,2} & W^{s_1}_{n,2} & \cdots & W^{s_J}_{n,2} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ W^{d_1}_{n,T} & \cdots & W^{d_J}_{n,T} & W^{s_1}_{n,T} & \cdots & W^{s_J}_{n,T} \end{array}\right].$$

Here, s_0 is not included as it is actually the original variable, but we will combine it later. The wavelet transformed data are then

$$W = [W^T_{1,.}, W^T_{2,.}, \cdots, W^T_{N,.}]^T.$$

Similarly, we get the wavelet transformed data W^{SW} corresponding to spatial spillover effects variable A^{SW}.

For the monthly data, a maximum time lag of 12 (lag.max) is chosen as it covers one whole year and the pollution now is more likely to be similar to the pollution same time last year. Taking the original data $A_{n,.}$ as an example, with time lag i included, it becomes

$$A^{lag.i}_{n,.} = [a_{n,i}, a_{n,(i+1)}, \cdots, a_{n,(T-lag.max+i)}]^T$$

Table 4.4: AQI forecasted results using original and wavelet data. First overall 6 important variables are listed.

Accuracy (R^2)		Variables					
original	0.5451	AQI.lag12	AQI.lag11	AQI.lag1	AQIW.lag12	AQIW.lag11	AQIW.lag1
wavelet	0.5742	AQI.s2.lag10	AQI.s1.lag11	AQI.s1.lag12	AQI.s2.lag11	AQI.s3.lag8	AQI.s3.lag7

and $A_{n,\cdot}$ with time lag becomes

$$A_{n,\cdot}^{lag} = [A_{n,\cdot}^{lag.1}, A_{n,\cdot}^{lag.2}, \cdots, A_{n,\cdot}^{lag.max}].$$

After combination, we have A^{lag}. In the same way, we can also get $(A^{SW})^{lag}$, W^{lag} and $(W^{SW})^{lag}$.

Until now, we have finished constructing the explanatory variables. Specifically, the original explanatory variables X_o are

$$X_o = [A^{lag}, (A^{SW})^{lag}]$$

and wavelet transformed explanatory variables X_w are

$$X_w = [A^{lag}, (A^{SW})^{lag}, W^{lag}, (W^{SW})^{lag}].$$

Note that X_o is included in X_w, as wavelet smooth variable on level 0 is actually the original variable.

$$Y^{reg} = [A_{\cdot,lag.max+1}^T, A_{\cdot,2}^T, \cdots, A_{\cdot,T}^T]^T$$

Training data and test data are separated by time point 30. So we have 558 observations for training and the remaining 341 for testing. The regression trees are applied to conduct the experiment with 100 trials to get an robust result. That is because for regression trees, random sampling in the cross validation leads to a slightly different cross-validation error, which may cause the tree depth to differ after pruning according to the cross-validation error.

The results in Table 4.4 show that wavelet variables do better in fitting. For variables used, original data based CART chooses AQI with lags 1, 11 and 12.

For weighted AQI, CART also choose lags 1, 11 and 12. That suggests AQI has a long-lag autocorrelation effect and adjacent provinces do have obvious effects on AQI itself. The ratio is 1.07, with the wavelet transformed variables having a higher R^2 (0.5742) compared to the original variables (0.5451).

4.7 Conclusion

In this Chapter, we compared the performance of the original variables with wavelet transformed variables in time series analysis. Wavelet data generally result in better accuracy measured by R^2 and R^2 ratio. Important variable information used in the tree is also obtained.

Specifically, MODWT based CART can detect true lag information and has much better performance when only short lag information is permitted ($k.lag$ is small). When forecast length increases, R^2 from MODWT based forecasting decreases much more slowly compared to that of original forecasting. For real data analysis, MODWT based CART also performs better in both LT data and Chinese air pollution data.

Chapter 5

Interval Forecasts based on Regression Trees for Streaming Data

5.1 Introduction

Chapter 4 has shown, compared to original CART, wavelet based CART has better performance in static time series prediction. So MODWT data will be used directly for streaming data forecasting in this chapter. Streaming data are data that are continuously generated by different sources. Such data should be processed incrementally using stream processing techniques without having access to all of the data. In a data stream analysis, models are built to capture information hidden in the data, either for description or prediction. Regression trees have been widely developed to capture such information. For many applications, forecasting the target value at a given time in the future is the primary task. However, sometimes an interval forecast is also required and maybe more useful than the point forecast. The term "interval forecast" refers to an interval that will "usually" (with a specified confidence) include the true value of the streaming variable at the specified time. The interval construction method is inspired by Appice & Ceci (2006), who consider count-based (count the number of observations) and normal distribution-based procedures. In this chapter, a count-based procedure is employed using quantiles information from the trees for interval construction, preferring not to rely on any distributional assumption or approximation.

The interest in this problem is motivated by a real data example. During surgery, the patient's heart rate is monitored in case of emergency. If we can reliably predict heart rate, even just a minute ahead, surgeons have some time for preparation, which could potentially save lives. In this circumstance, point prediction for an exact heart rate value is of limited use. Surgeons pay more attention to the range of the heart rate (whether in the normal range or not), and methods designed for interval forecast are more directly targeted at range monitoring. We apply these methods to financial data as well. Other applications include industrial process monitoring, social media activity or customer behaviour data.

Forecasting can be based on statistical time series models like ARIMA or GARCH, but they require structural and distributional assumptions such as white noise should be normally distributed. In reality, unseen future data may not satisfy these assumptions. Data mining methods like decision trees do not need such assumptions. Trees are simpler to interpret and can provide simple decision rules for people to refer to when an decision mechanism is required. However, since trees are generally not based on an underlying statistical or probabilistic model, there is no distributional means of constructing confidence intervals based on regression trees. As trees allocate each observation into a terminal node, the collection of observations in each terminal node can be used to construct an interval estimate. For these reasons, we choose trees as the regression model and compare the tree based forecasts with those produced by ARIMA and GARCH models.

The coverage rate (proportion of test values falling into the interval forecast) of the interval forecast are regarded as the main measure of how well the methods are performing. If almost all the true values are in the interval forecasts, the methods are successful, though we would like a specified proportion. Interval width and computational load are also taken to be measures of performance.

The rest of this chapter is arranged as follows. Section 5.2 reviews key concepts in regression tree modelling of streaming data. The methodology is introduced in Section 5.3, and is applied to simulated data in Section 5.4 with ARIMA and GARCH for comparison. After that we apply it to real datasets followed by comparisons with ARIMA and ARMA-GARCH in Section 5.5 and Section 5.6 respectively. Some concluding comments appear in Section 5.7. All computation

was performed in R (R Core Team, 2014), using the packages partykit (Hothorn & Zeileis, 2015) for regression trees, waveslim (Whitcher, 2013) for wavelet decomposition and forecast (Hyndman, 2017; Hyndman & Khandakar, 2008) for ARIMA, rugarch (Ghalanos, 2014) and fGarch (Wuertz *et al.*, 2013) for GARCH and ARMA-GARCH.

5.2 Related work

Like other methods which use regression trees for streaming data, there are several questions we need to answer. The first one is how to utilize long term variable information. Since forecasting is conducted in the context of streaming data, most of the current data depend on old data. Building models using only the most recent data seems unwise, so storing old information in an efficient way is important. One way of doing this is via a wavelet transform, which can pick out long term averages and short term fluctuations. Instead of using the standard decimated discrete wavelet transform (DWT), we use the maximal overlap discrete wavelet transform (MODWT; Percival & Walden (2000)), as it is not constrained by time series length T_n, so MODWT can be applied to every time point. Wavelet transformed variables have already been shown to be better than or similar to original variables in the context of classification trees (Zhao *et al.*, 2018) and static time series forecasting in Chapter 4. So again, here the input variables are wavelet transformed variables.

The second question is how to deal with streaming data. Various tree-based models have been designed for streaming data, many of which are implemented in the Massive Online Analysis (MOA) open source framework for data stream mining (Bifet *et al.*, 2010). In MOA, the Hoeffding Tree family (Bifet *et al.*, 2009; Domingos & Hulten, 2000; Jin & Agrawal, 2003; Pfahringer *et al.*, 2007) provides some important models, followed by other recently developed models to include more complex situations (Duarte *et al.*, 2016; Ikonomovska *et al.*, 2011, 2015). However, performance of these models is assessed by accuracy of the predicted values as point estimates, while here we pay more attention to performance based on interval forecast accuracy. Without considering trees, there are other models which use neural networks to obtain prediction intervals such as Shrestha & Solomatine (2006) and Quan *et al.* (2014) which combines coverage and width as

one target. These models can be applied to streaming data as well, but lack the straightforward interpretation of tree-based models.

Thirdly, how to detect concept drift (statistical properties of the target variable changes)? The data generation mechanism can be referred to as "concept". This concept can remain stable or change over time, for example, the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways. This causes problems because the predictions become less accurate as time passes. If it is stable, models built now can still be used for future prediction. If it is not stable, we say concept drift happens, either gradually or abruptly. In this scenario, statistical properties of the target variable, which the model is trying to predict, change in unforeseen ways. This leads to poor prediction performance, as the model built based on the old concept is no longer suitable for prediction of the target variable. So we need to develop concept drift detections tools and update or retrain the model when drift is detected.

Concept drift detection tools are generally based on the prediction accuracy. The Drift Detection Method (DDM) (Gama *et al.*, 2004) monitors the probability of error in real time. The Early Drift Detection Methodology (EDDM)(Baena-García *et al.*, 2006) was developed as an extension of DDM, and is more suitable for slow moving gradual drifts, where DDM previously failed (Sethi & Kantardzic, 2017). The Statistical Test of Equal Proportions (STEP) (Nishida & Yamauchi, 2007) computes the accuracy of a batch of recent predictions and compares it with the overall accuracy from the beginning of the stream, using a chi-squared test to check for deviation. An incremental approach, Concept Drift Detection (ECDD), was proposed by Ross *et al.* (2012). Some window-based methods like the Adaptive Windowing (ADWIN) algorithm (Bifet & Gavalda, 2007; Yoshida *et al.*, 2011) and SeqDrift2 (Sobhani & Beigy, 2011) detect whether sub windows have significant differences in terms of predictive accuracy. These tools are generally based on the prediction accuracy.

We wish to base the response to concept drift on the accuracy of the interval forecasts rather than the prediction accuracy. An interval forecast which covers most of the data suggests that the model in use is a good one which should continue to be used as long as it remains effective. The three simple ways for a time series to change are to change mean value with variance fixed; change variance with mean

value fixed; or change both mean value and variance. When variance changes but with mean value fixed, we can enlarge or shrink our interval forecast to cover future data more efficiently. But when the mean value changes with or without variance change, we need to consider building a new model. For the terminal node in a tree, the interval forecast from this node is obtained from the sample quantiles of the observations falling into this node. So when the mean of the data changes substantially, our tree will no longer be suitable. The criterion is that when too many test data fall outside the forecast intervals, then we build a new model with the most recent batch of data to update the current ensemble model or replace the current model.

The fourth question is when concept drift has been detected, what should we do. There are many ways to improve the model when concept drift is detected. Some rebuild the model using the new batch of data after the concept drift point. Some remove the poorly performing nodes and grow new nodes with the latest batch (Bifet & Gavaldà, 2009; Domingos & Hulten, 2000). In ensemble methods, some approaches drop the worst model and replace it with the rebuilt model but with the other models unchanged. Inspired by these ideas, we develop two methods, one based on a single tree and one ensemble method.

5.3 Methodology

5.3.1 Background

In this section, two methods are proposed to create interval forecasts for streaming data; one method based on a single tree and one ensemble approach. We then describe an approach to adaptively adjust the interval forecasts to the changing characteristics of the data and discuss criteria for model retraining. Finally, some measures of model performance are proposed.

The dataset will be introduced here first. We use regression tree models to forecast time series $\{y_t\}$, denoting the predicted value h steps ahead of time t as

$$\hat{y}_{t+h} = f(y_t, y_{t-1}, \ldots, y_{t-\gamma+1}).$$

This prediction uses the previous γ observations to predict y at h time points ahead. It is assumed that the initial training dataset consists observations up to

time T. The first $T-h$ observations will be used as A and the last h observations will be used as the response variable. A schematic matrix representation is

$$A = \begin{bmatrix} y_1 & y_2 & \cdots & y_\gamma \\ y_2 & y_3 & \cdots & y_{\gamma+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{T-h-\gamma+1} & y_{T-h-\gamma+2} & \cdots & y_{T-h} \end{bmatrix} \begin{matrix} \longrightarrow \\ \longrightarrow \\ \vdots \\ \longrightarrow \end{matrix} \begin{bmatrix} y_{h+\gamma} \\ y_{h+\gamma+1} \\ \vdots \\ y_T \end{bmatrix}. \tag{5.1}$$

One thing to note is that not all of the matrix A will be used to predict each future y_t, but each line will be used to predict each time point. The matrix just represents a batch of data. Since we wish to use the wavelet transformed data of the left matrix A, a longer time range is required. For example, the level of the wavelet transform is set as 8, which requires at least 256 observations. If γ is smaller than 256, then the observations from $y_{T-h-255}$ to y_{T-h} will be used for wavelet transform instead of $y_{T-h-\gamma+1}$ to y_{T-h} and the series of wavelet coefficients will be truncated from time $T-h-r+1$ to time $T-h$ to make it the same length as $y_{T-h-\gamma+1}$ to y_{T-h}. In that case, the training can only start when the number of stored observations reaches 256. So instead of starting from y_1 to y_γ, it will start from $y_{257-\gamma}$ to y_{256}. We refer to the matrix of wavelet coefficient series as W. Then we can use each line in W to predict $y_{h+\gamma}, \ldots, y_T$. For example, y_t in A will be $[W_t^{d_1} \cdots W_t^{d_8}, W_t^{s_1} \cdots W_t^{s_8}]$ in the matrix W.

In the later sections, we will use this wavelet transformed variable W as our input variable. Specifically, the explanatory data for training and prediction are all wavelet transformed data.

5.3.2 Proposed methods

We propose two forecasting methods, an ensemble method and a single-tree method. As shown in Figure 5.1, initially, when T reaches time t_0, the observations up to time t_0 will be used to train the initial model as shown in Equation 5.1 and training stage 1 in Figure 5.1. Then, the observations $y_{h+\gamma}$, ..., y_{t_0} are changed from output prediction observations to input observation in A by changing y_{t_0-h} to y_{t_0}. So we can use data A up to t_0 to obtain predictions from $t = 2h+\gamma$ to t_0+h shown as prediction stage1. From then on, forecasting is done continuously, but after each batch of data, we review and update our models.

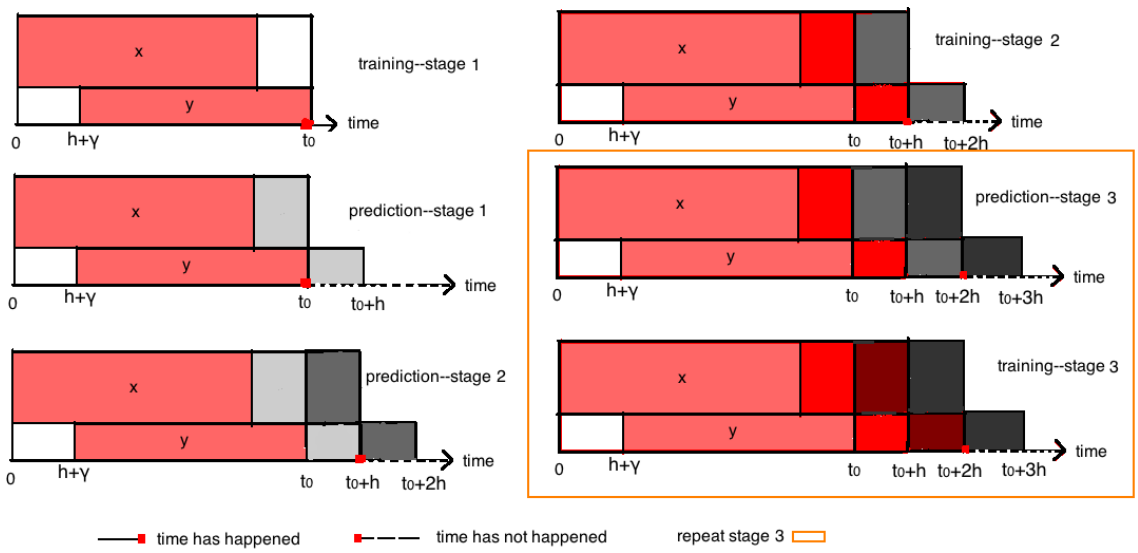


Figure 5.1: Methods process using original data for illustration.

We now describe how the methods proceed in stages. At Stage 1, we use the initial t_0 observations for initial training and forecasting. Each subsequent batch of h observations afterwards are regarded as a new stage in the process, so at Stage k, data y_t, $t = 1, 2, \ldots, t_0 + kh$ will be available. At each Stage $k = 1, 2, 3, \ldots$, we go through the following steps. (Some details of the model and forecast updating process are deferred to the Section 5.3.3.)

- Step 1: $(k+1)^{st}$ prediction
  - $(k+1)^{st}$ *prediction (single method)*

    For each new observation at Stage k, use the latest model to get a prediction P_t for $t \in S_{k+1} = \{t_0 + kh + 1, \ldots, t_0 + (k+1)h\}$.
  - $(k+1)^{st}$ *prediction (ensemble method)*

    For each new observation at Stage k, continuously use the method below to get a weighted forecast for new data at Stage $k+1$. Denote a prediction by the vector $P = (L, \hat{y}, U)$, including the predicted value $\hat{y}$ and forecast interval limits L, U. Let $P_{(1)}$ be the forecast from the first (oldest) model (not constrained to trees), $P_{(2)}$ the next oldest and so

on. Then the combined forecast from a bag of $s \leq m$ models is $\bar{P}_{(s)}$, defined recursively by

$$\begin{aligned}\bar{P}_{(2)} &= k_1 P_{(1)} + k_2 P_{(2)},\\ \bar{P}_{(3)} &= k_1 \bar{P}_{(2)} + k_2 P_{(3)},\\ &\vdots\\ \bar{P}_{(s)} &= k_1 \bar{P}_{(s-1)} + k_2 P_{(s)},\end{aligned}$$

where $k_1 + k_2 = 1$, s is the number of models trained which should be smaller than the maximum model allowance m (set beforehand). Generally $k_2 > k_1$ to make the most recent forecast the most important term.

- Step 2: k^{th} RMSE calculation

  Calculate the root mean squared error (defined later in Equation (5.2)) between the newly observed data from Stage k and the point forecasts of these data which were made in the previous Stage $k-1$ (k^{th} prediction).

- Step 3: k^{th} model updating

  If a new model was created at Stage $k-1$, retrain the model using the new data or update the model using the k^{th} RMSE as described later in §5.3.3.

- Step 4: $(k+1)^{st}$ model training

  - *$(k+1)^{st}$ model training (single method):*

    The current model may become less relevant, showing bias or excess uncertainty due to concept drift, so we choose to train a new model when necessary. However, a new model is only trained when deemed necessary and otherwise we continue with the existing model. The decision on whether to train a new model is based on the coverage of the forecasts on the most recent batch of data. For the latest h observations, let

$$p_y = \max\left\{\frac{1}{h}\sum_{t\in S_k} I[\hat{y}_t > (y_t + \delta)], \frac{1}{h}\sum_{t\in S_k} I[\hat{y}_t < (y_t - \delta)]\right\},$$

which will increase if the predictions are systematically above or below the true values by a tolerance δ. If p_y exceeds a threshold of, say, 90%, then we train a new model using these h observations to replace the old model.

– $(k+1)^{st}$ *model training (ensemble method):*

Use data at Stage k to train a new model. If the total amount of data observed until now is greater than some upper limit T_Ω, then we use the latest T_Ω observations to emphasize more recent information and make the models more responsive to concept drift.

This outlines the ongoing process of model updating which will continue as long as data are being observed. We now give more details of the construction and updating of our forecast intervals, which is largely the same for both ensemble and single-tree methods.

5.3.3 Construction and updating of interval forecasts

Forecast interval construction

For each terminal node, the 0.025 and 0.975 quantiles of the values in that node are chosen as an initial interval $[L', U']$. To make the forecast interval less sensitive to overfit the training data, it is enlarged to

$$[L, U] = [L' - \alpha(U' - L'), U' + \alpha(U' - L')], \;\; \alpha \in R.$$

Larger values of the tuning parameter α lead to wider intervals, usually with correspondingly higher coverage. So α can be used to trade off coverage and width. This tuning parameter is only applied to new models when they are first trained.

Updating forecast intervals

The width of the interval forecasts will be adaptively adjusted depending on the coverage of the most recent point forecasts. For a general Stage k, comprising

time points $t \in S_k$, the root mean squared error of the point forecasts of the h observations is

$$RMSE = \sqrt{\frac{1}{h}\sum_{t \in S_k}(\hat{y}_t - y_t)^2}. \tag{5.2}$$

Denote the forecast interval at time t by $[L_t, U_t]$. Let p_U and p_L be the proportions of intervals with $y_t < U_t$ and $y_t > L_t$ respectively:

$$p_U = \frac{1}{h}\sum_{t \in S_k} I\{y_t < U_t\}, \qquad p_L = \frac{1}{h}\sum_{t \in S_k} I\{y_t > L_t\}.$$

There are four combinations of how $[L, U]$ might be updated, which are summarized in Table 5.1. These empirical coverage rates will decide when to increase or decrease L or U by $\beta \cdot RMSE$ to adapt for the forecasting performance at Stage k, since we assume the model updated by the current data can be applied to the future prediction. The reason to use $RMSE$ is that it is a good measure of the model performance. Other metrics can also be considered. Here, β is a tuning parameter and values of $\beta \in [2, 3]$ are found to be effective. We set lower and upper target coverage rules a and b, and try to maintain coverage rates in the range (a, b). Typical values would be $a = 0.95$, $b = 0.99$.

If $p_U < a$, meaning that the proportion of observations where $U_t > y_t$ does not even reach the minimum desired coverage, then the values of U will be increased by $\beta \cdot RMSE$. Similarly, if $p_L < a$, L will be decreased by $\beta \cdot RMSE$.

If $p_U > b$, meaning that the observation is below the upper limit of the forecast interval more often than we intended, the future forecasts U will be decreased by $\beta \cdot RMSE$, so long as this does not make $U_t < \hat{y}_t$. The U_t will not be decreased, if the adjusted p_U would go below a while calibrating on the newly-arrived Stage k data. Similarly, if $p_L > b$, we increase L by $\beta \cdot RMSE$ subject to the constraint $L_t < \hat{y}_t$.

5.3.4 Performance measurement

To measure the performance of the interval forecasts, the coverage will be used, that is the proportion of observations in $[L, U]$:

$$\text{coverage} = \frac{1}{T - t_0}\sum_{t=1}^{T-t_0} I\{y_t \in [L_t, U_t]\}.$$

	$p_U < a$	$p_U > b$
$p_L < a$	$U \to U + \beta \cdot RMSE$ $L \to L - \beta \cdot RMSE$	$U \to U - \beta \cdot RMSE$ $L \to L - \beta \cdot RMSE$
$p_L > b$	$U \to U + \beta \cdot RMSE$ $L \to L + \beta \cdot RMSE$	$U \to U - \beta \cdot RMSE$ $L \to L + \beta \cdot RMSE$

Table 5.1: Situations where $[L, U]$ is adapted to recent forecasting performance, subject to the constraints, for example $L_t < \hat{y}_t < U_t$.

However if two methods have similar coverage, then we prefer the method which has the lower mean forecast-interval width:

$$\text{width} = \frac{1}{T - t_0} \sum_{t=1}^{T-t_0} (U_t - L_t).$$

5.4 Simulation study

In this section, the performance of tree based models will be compared with that from parametric models using simulated data. If the dataset is generated from a parametric-model based distribution family, then we can ensure the parametric model will be a suitable approach. If non-parametric models, like trees, can have better or similar performance than this parametric model, then the non parametric model is good. ARIMA and GARCH are chosen as the parametric models.

5.4.1 ARIMA simulation

Here ARIMA is chosen as the parametric model and trees as the non-parametric model. The general expression for an ARIMA model is

$$\left(1 - \sum_{k=1}^{p} \alpha_k B^k\right) (1 - B)^d y_t = \left(1 + \sum_{k=1}^{q} \beta_k B^k\right) \epsilon_t,$$

where B is the lag operator, the α_k are the parameters of the autoregressive part of the model, the β_k are the parameters of the moving average part and the ϵ_t are error terms assumed to be independently normally distributed with mean zero and variance σ^2. Here, in general p and q are chosen as 1, but for noise only

variables, p and q are chosen as 0. Here d can be 0 (without trend) or 1 (if trend is possible). In this simulation, we assume, for one time series generated, there is minimal distribution change (parameters share different but similar values) to allow minimal changes in one time series flow, but, for different time series, distribution changes obviously since they come from different sources. The time series are generated of length 3000 according to one of the parameter distributions j;

$$\{Y^i\} \sim ARIMA(\alpha^i, \beta^i, \sigma^i, d^i), \;\; i = 1, 2, \ldots, 10,$$

where α^i, $\beta^i \sim U(a_j, b_j), i.i.d$, $\sigma^i \sim U(c_j, d_j)$ and $d^i \sim B(1, 0.5)$ with possible value 1 to allow trend, and $d^i = 0$ without trend. After choosing parameters from distribution j, we generate parameters α_i, β_i, σ_i, and d_i 10 times from $U(a_j, b_j)$, $U(c_j, d_j)$ and $B(1, 0.5)$; so we get nine change points in each time series. For example, for the first time series ($j = 1$), we have values of $U(a_1, b_1)$ and $U(c_1, d_1)$ as $U(1, 2)$ and $U(0.1, 0.2)$ (values are only for illustration). For the second time series ($j = 2$), we have $U(a_2, b_2)$ and $U(c_2, d_2)$ as $U(5, 6)$ and $U(0.7, 0.8)$. Their values differ a lot among different j to make them different time series. For one time series (like $j = 1$), α^i, β^i, σ^i are randomly chosen from $U(1, 2)$ and $U(0.1, 0.2)$ to make slightly changes when time goes on. The time series we use for one simulation is

$$Y = \{Y^1, Y^2, \ldots, Y^{10}\}.$$

In each simulation, we generate one time series Y of length 30000 (3000 times 10) for each j, and apply all methods separately for forecasting. The number of simulations is chosen as 20 ($j = 20$). So there are 20 time series, in which each time series has 30000 observations. The parameters we find work well for tree based methods are $\beta = 2$, $\alpha = 0.2$, $b = 0.99$, $a = 0.95$, $h = 100$ (can be other values as well), $t_0 = 1311$ (including input and output variables of length 1200, prediction ahead $h = 100$, and $\gamma = 12$) and $\gamma = 12$, $\delta = 2$, $T_\Omega = 1000$, and $m = 3$.

A similar ARIMA forecasting approach will be employed to the regression tree methods for comparison. The input variable (y) will be the original variable instead of wavelet transformed variables, as ARIMA generally uses only one variable as input variable. (There is a model $ARIMAX$ that takes in multiple input variables that can also be considered in future work.) We use the function auto.arima

in the R package forecast (Hyndman, 2017; Hyndman & Khandakar, 2008) to find the best ARIMA model according to AICc (small sample size corrected AIC). This function conducts a search over possible models within the order constraints provided: $p, q \in \{0, 1, \ldots, 6\}$ and $d \in \{0, 1, 2\}$. Allowing higher orders incurs a higher computational load. In R, ARIMA can try to reach a given prediction interval level given by users. In order to compare the performance of ARIMA with trees, the prediction interval levels c chosen for Single ARIMA and Ensemble ARIMA are coverage results from Single tree and Ensemble tree methods to obtain results comparable to those obtained using regression trees. As in the regression tree methods, the first $t_0 = 1311$ (same as the tree setting for comparision) observations will be used to train the initial model. Using data from $t - \gamma + 1$ to t, we can predict the value at time $t + 1$

$$\hat{y}_{t+1} = f(y_t, y_{t-1}, \ldots, y_{t-\gamma+1}),$$

where $\gamma = 1000$. Then we recursively use this predicted $\hat{y}_{t+1}$ to predict $\hat{y}_{t+2}$

$$\hat{y}_{t+2} = f(\hat{y}_{t+1}, y_t, y_{t-1}, \cdots, y_{t-\gamma+2}),$$

and so on to get the predicted value $h = 100$ observations ahead as $\hat{y}_{t+100}$.

There are two criteria for model retraining and updating. As shown in Table 5.1, retraining occurs if the method fails to achieve coverage between $c - 0.3$ and 0.99; updating if the method fails to achieve coverage between $c - 0.15$ and 0.99, where c is the coverage result from trees. The second criteria is that the minimum gap between two successive retraining or updating is 100 new observations so as to avoid excessively frequent fitting. When retraining, instead of using h observations as in the regression model, we use $10h$ observations to make the ARIMA model robust to short-term trends. For example, if ARIMA still uses h observations as in the regression model, when trend is detected in that h observations, there might be quite sharp increase or decrease in the long-term prediction. But the reality might be the trend is only temporally appeared in that h observations, and it will not continue afterwards. The reason why trees use that h observations is that trees do not have trend effect built in the model and the predicted value at each terminal node is fixed. So there is no need to worry about the sharp increase or

decrease. But, for ARIMA, it uses $10h$ observations instead to build a relatively robust model. In the Ensemble tree method, we choose to train a new model for every $h = 100$ observations. Since we use $10h$ observations for ARIMA training, in the ensemble method for ARIMA, there is no need to train a new model every 100 observations. If not, the model will only have slightly change since only 100 out of 1000 observations are updated. The model updating and retraining criteria are the same as the Single ARIMA shown in Table 5.1. For updating, we keep p, q, and d fixed and update the parameters using the most recent $\gamma = 10h$ observations. For the ensemble ARIMA model, we choose up to $m = 3$ most recent models and weight predictions in the same way as that in the ensemble tree method. Other criteria are the same as in the tree methods. Results from 20 simulations are shown in Table 5.2, Figure 5.2 and Figure 5.3.

Table 5.2: The mean and standard deviation (sd) of coverage, width, widthsd and time for all methods across 20 simulations. Widthsd is the averaged standard deviation of width within each simulation. The sd in width(sd) is the standard deviation of mean width across 20 simulations.

α, β		U(0.1, 0.2)				U(0.8, 0.9)			
σ	white noise	U(0.2, 0.5)		U(4, 5)		U(0.2, 0.5)		U(4, 5)	
trend possible		no	yes	no	yes	no	yes	no	yes
Case	0	1	2	3	4	5	6	7	8
Single tree method									
coverage(%)	99.38(0.1)	98.90(0.5)	76.48(5.8)	99.33(0.3)	71.22(6.7)	94.89(1.3)	67.99(9.4)	94.47(1.3)	64.47(5.6)
width	5.48(0.1)	3.32(0.4)	8.40(2.4)	12.13(0.5)	33.31(12.0)	11.82(1.1)	62.41(26.7)	42.99(3.4)	232.54(84.3)
widthsd	0.01(0.0)	0.14(0.1)	8.36(2.9)	0.04(0.1)	33.97(13.3)	3.05(0.7)	82.56(32.4)	11.74(2.2)	288.72(94.1)
time	17.59(0.3)	19.04(0.4)	21.14(0.8)	18.30(0.6)	21.96(1.2)	18.35(0.6)	21.52(1.5)	17.80(0.6)	21.79(0.7)
Ensemble tree method									
coverage(%)	99.34(0.0)	99.27(0.1)	74.69(7.2)	99.31(0.1)	71.40(8.3)	94.43(0.8)	67.81(9.5)	94.16(0.8)	65.28(5.5)
width	5.46(0.0)	3.30(0.1)	8.27(1.7)	12.05(0.1)	32.83(7.2)	9.54(0.6)	59.32(19.6)	34.63(1.1)	222.70(51.9)
widthsd	0.16(0.0)	0.37(0.1)	8.25(2.5)	0.51(0.1)	29.49(5.9)	2.21(0.2)	79.42(25.0)	7.27(0.5)	270.90(60.5)
time	123.95(3.3)	122.20(2.0)	127.39(4.2)	121.08(3.4)	127.09(3.7)	125.45(3.1)	126.99(3.9)	123.95(2.6)	129.86(2.1)
Single ARIMA method									
coverage(%)	99.05(0.1)	98.38(0.8)	76.96(5.7)	99.00(0.1)	80.75(5.4)	93.08(1.2)	89.35(5.8)	93.86(1.4)	86.97(4.6)
width	6.34(0.5)	3.39(0.3)	20.06(8.1)	12.92(1.2)	65.12(23.1)	11.89(1.8)	216.87(69.8)	42.48(7.1)	765.44(196.9)
widthsd	6.26(1.4)	2.79(0.8)	66.96(48.0)	10.21(4.3)	201.71(116.8)	10.72(1.7)	396.44(151.9)	36.40(7.7)	1430.69(533.4)
time	98.50(2.8)	95.05(3.6)	83.49(2.3)	97.72(2.8)	82.89(2.3)	97.52(4.9)	90.08(2.8)	96.34(4.5)	88.62(3.9)
Ensemble ARIMA method									
coverage(%)	99.08(0.1)	98.59(0.5)	76.39(6.1)	99.02(0.1)	75.11(6.2)	95.18(0.8)	76.23(5.1)	95.10(0.8)	74.24(3.5)
width	6.33(0.5)	3.49(0.3)	18.07(6.3)	12.90(1.1)	61.03(15.1)	11.55(1.5)	197.41(63.1)	42.36(5.7)	720.77(176.5)
widthsd	5.46(1.2)	2.61(0.7)	53.63(37.8)	8.65(3.5)	163.53(98.3)	9.13(1.5)	344.90(124.7)	32.91(6.1)	1275.78(502.8)
time	414.17(7.3)	411.76(8.8)	516.41(271.1)	413.80(7.7)	466.25(227.0)	646.61(347.6)	793.38(382.8)	820.32(413.1)	630.81(342.7)

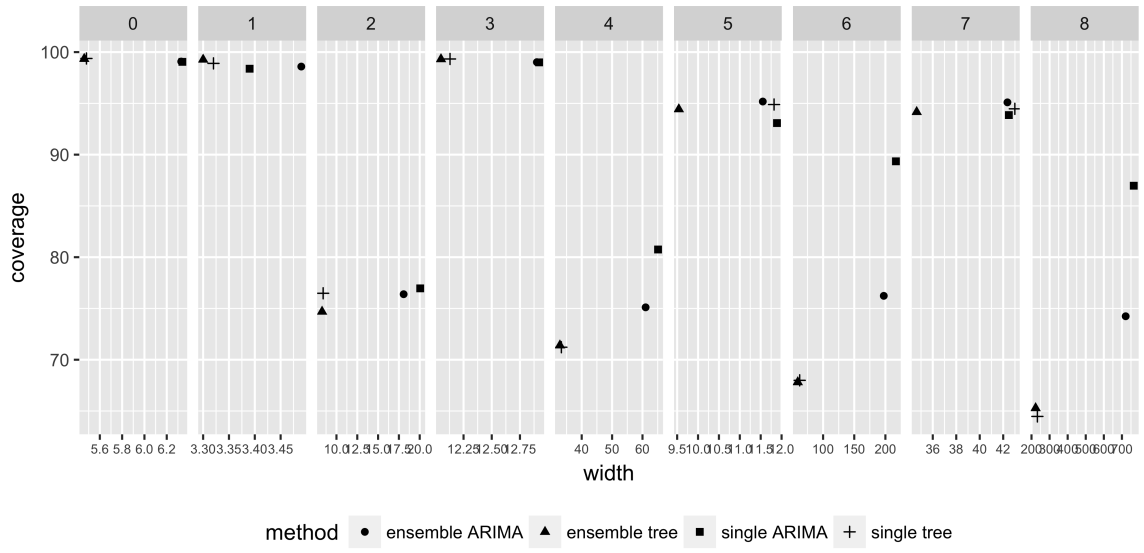


Figure 5.2: Simulation results of coverage and width. Panels 0-8 represent the 9 distributions in Table 5.2.

When comparing tree methods with ARIMA methods in terms of computation time, tree methods take less than 25% of the time of ARIMA in both single and ensemble methods.

When comparing tree methods with ARIMA in coverage and width, it is clear that the Ensemble tree method is better than Single ARIMA method and Ensemble ARIMA method in most situations except Cases 4, 6 and 8 in Figure 5.2. But for these cases, the points are roughly in a line, which means methods with higher coverage are at the cost of higher width. If we allow a wider interval for tree methods, they might achieve the same coverages as that of ARIMA. For the tree methods, the Single tree method has equal performance to that of the Ensemble tree method except Cases 5 and 7 where the Single tree method has a relatively higher width.

Comparing widthsd, tree methods are better than ARIMA. For a time series, when trend disappears, not yet updated ARIMA will still forecast with trend before retraining thus the forecasted intervals can be extremely wide. When there is no trend, but a trend is falsely detected by ARIMA, intervals can be wide as well, but tree methods, which consider trend in a different way, can avoid such situations.

When the time series distribution changes, tree methods react more quickly compared to ARIMA which uses information directly. An example is shown in the circled area of Figure 5.3.

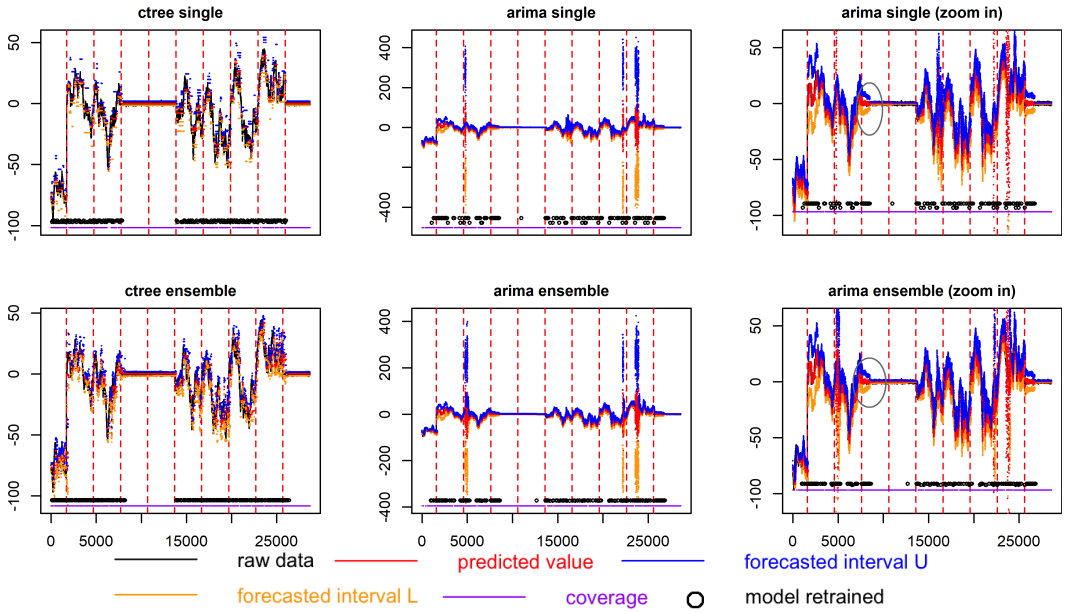


Figure 5.3: Simulation results (from one realisation of Case 3). In this plot, α, β both follow $U(0.1, 0.2)$, σ follows $U(0.2, 0.5)$ with trend possible. Red dashed vertical lines represent the time when distribution changes. ARIMA has extremely wide intervals in some cases. Take the circles area as an example, ARIMA reacts slowly when distribution changes as shown in the circles area.

In conclusion, the Ensemble tree method is somewhat better than the others when there is no time efficiency requirement. When time is critical, the Single tree method is suggested. When the ARIMA effect is strong and trend possible, the Single ARIMA method is suggested at the cost of wide intervals.

5.4.2 GARCH simulation

We now choose GARCH as our parametric model. Simulated data from the GARCH model as

$$y_t = \sigma_{t|t-1} e_t,$$

where e_t follows i.i.d. $\mathcal{N}(0,1)$ and is independent of past $y_{t'}$, with $t' < t$ and

$$\sigma_{t|t-1} = w + \sum_{k=1}^{r} \gamma_k y_{t-k}^2 + \sum_{k=1}^{s} \delta_k \sigma_{t-k|t-k-1},$$

where γ_k and δ_k measure the influence of y_{t-k}^2 and $\sigma_{t-k|t-k-1}$ to the current $\sigma_{t|t-1}$. In this simulation, both r and s are chosen to be 1. However, when fitting the GARCH model, both r and s are allowed to a maximum of 3. For the data simulation process, it is the same as the ARIMA simulation part. The functions in use include garchAuto, ugarchfit, ugarchforecast in the R packages rugarch (Ghalanos, 2014) and fGarch (Wuertz *et al.*, 2013). The criteria to find the best GARCH model is AIC. The results are shown in Figures 5.4, 5.5 and Table 5.3. Figure 5.4 shows that the forecast intervals from the Single tree method are less responsive to concept drift. For Case 4, tree methods are better than GARCH with a slightly smaller width. For the other circumstances, they share similar performance. That means, under similar parameter settings, tree methods have similar performance to the GARCH methods even the data are generated from GARCH distributions.

5.5 Forecasting heart rate during LT surgery

As introduced in Section 1.3, the data in use come from patients undergoing Liver Transplantation (LT) operation, which is a high-risk treatment choice for patients having end-stage liver disease. The data, which was recorded using a LIDCO monitor on patients undergoing LT between September 2004 and December 2011, is provided by St James's University Hospital, Leeds, UK. For details, refer to Milan *et al.* (2016). The variable in use is the heart rate (beats/min), as it is one of the most important variables to be monitored on an ongoing basis during surgery. The aim is to predict heart rate $h = 100$ heart beats ahead (in our data, each heart beat takes around 0.5 to 1 seconds), so that the operating team can have enough time for preparation if heart rate is forecasted to go out of a normal range,

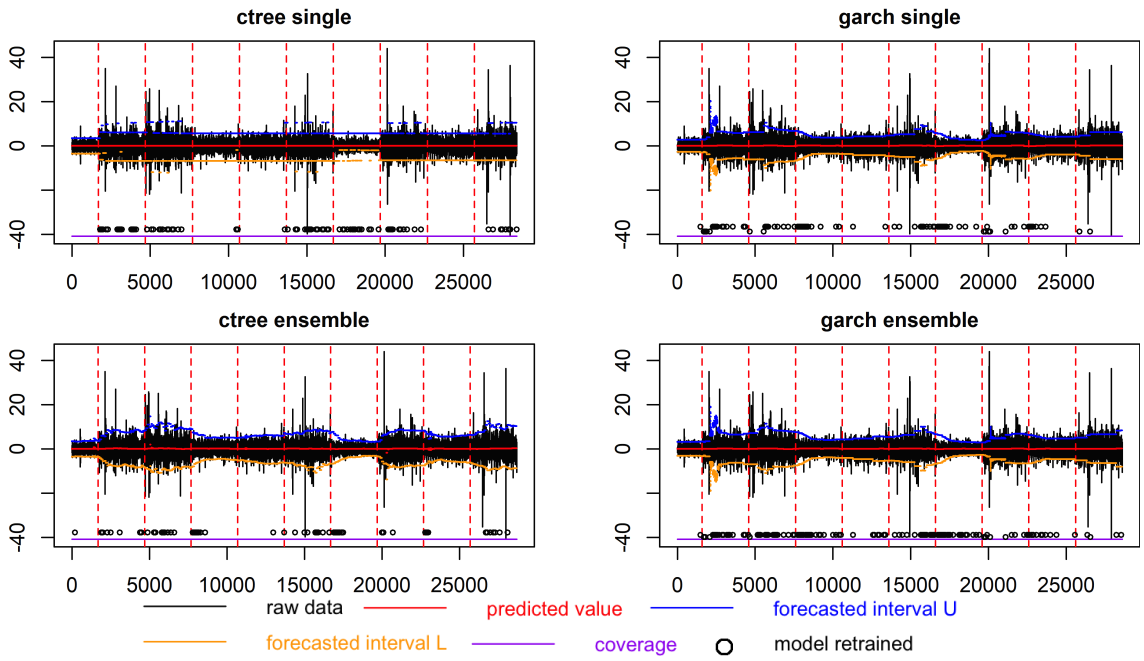


Figure 5.4: Simulation results (from one realisation of Case 4). In this plot, γ and δ both follow $U(0.3, 0.5)$. Red dashed vertical lines represent the time when distribution changes.

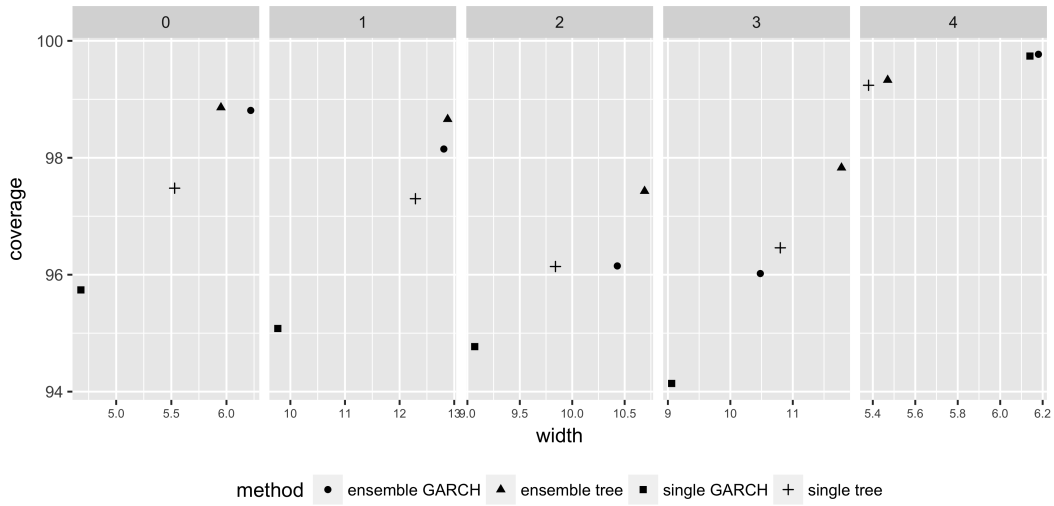


Figure 5.5: Simulation results for paired coverage and width. Panels 0-4 represent the 5 distributions in Table 5.3.

Table 5.3: The mean and standard deviation (sd) of coverage, width, widthsd and time for all methods across 20 simulations. Widthsd is the standard deviation of width within each simulation. The sd in width(sd) is the standard deviation of mean width across 20 simulations. w follows uniform distribution $U(0, 2)$.

γ	white noise	U(0, 0.2)	U(0, 0.2)	U(0.6, 0.8)	U(0.3, 0.5)
δ		U(0, 0.2)	U(0.6, 0.8)	U(0, 0.2)	U(0.3, 0.5)
Case	0	1	2	3	4
Single tree method					
coverage(%)	97.48(0.7)	97.30(0.5)	96.14(0.6)	96.46(0.7)	99.24(0.2)
width	5.53(0.6)	12.29(2.5)	9.84(1.2)	10.80(2.2)	5.38(0.2)
widthsd	1.47(0.4)	4.30(1.6)	3.24(0.8)	3.79(0.9)	0.00(0)
time	18.16(0.5)	18.28(0.4)	18.39(0.3)	18.31(0.2)	18.30(0.2)
Ensemble tree method					
coverage(%)	98.86(0.3)	98.66(0.2)	97.43(0.1)	97.83(0.3)	99.33(0.0)
width	5.95(0.5)	12.88(2.3)	10.69(1.1)	11.78(2.3)	5.47(0.0)
widthsd	1.83(0.3)	5.23(2.2)	4.03(1.1)	4.78(1.8)	0.15(0.0)
time	118.04(3.0)	119.97(1.3)	119.42(1.0)	119.68(1.0)	119.90(1.0)
Single GARCH method					
coverage(%)	95.74(1.8)	95.08(1.3)	94.77(1.0)	94.14(1.3)	99.74(0.1)
width	4.68(0.7)	9.77(2.0)	9.07(1.4)	9.06(1.8)	6.14(0.2)
widthsd	1.52(0.4)	4.19(1.8)	5.59(2.4)	4.78(1.9)	0.32(0.0)
time	659.13(337.2)	647.46(173.7)	704.80(112.0)	645.69(152.6)	1998.05(66.3)
Ensemble GARCH method					
coverage(%)	98.81(0.8)	98.15(0.8)	96.15(0.7)	96.02(1.0)	99.77(0.0)
width	6.22(0.9)	12.81(2.5)	10.43(1.8)	10.48(2.2)	6.18(0.0)
widthsd	1.99(0.3)	5.40(1.9)	6.37(2.6)	5.42(2.3)	0.31(0.0)
time	1876.97(211.8)	1715.50(206.6)	1201.78(111.1)	1185.54(122.0)	2294.39(31.2)

when the patient might be in danger. To illustrate the methods, the previously cleaned data (Zhao *et al.*, 2018) as described in Chapter 3 will be used here.

5.5.1 Tree-based forecasting

When forecasting heart rate using the Ensemble and Single tree methods, we found parameter values $\beta = 2$, $\alpha = 0.2$, $b = 0.99$, $a = 0.95$, $h = 100$, $t_0 = 1311$ and $\gamma = 12$, $\delta = 2$, $T_\Omega = 8000$, and $m = 3$ generally work well across those patients. We use one patient's data for illustration. The results are shown in Figure 5.6. Ensemble and Single tree methods have coverage $(77.7\%, 85.9\%)$, widths $(7.6, 6.8)$, and computing times $(134, 12)$ in seconds. From the results, we can see that both methods have good performance. Generally, the forecast intervals cover the true values without being excessively wide, only failing to include the true data when there are episodes of high volatility. Retraining happens with the same path of distribution change as shown in the Single method.

Summary results for all 325 patients are shown in Table 5.5 and Figure 5.7 including an enlarged version of those patients whose mean interval width was smaller than 20. Figures 5.7 to 5.8 and Table 5.5 show that the results of Single and Ensemble tree methods are generally similar, with the Ensemble tree method giving a little higher coverage but having wider intervals and being substantially more computationally demanding. The Ensemble tree method has a relatively lower density compared to Single tree when width is around 10 and the coverage density is a little higher when coverage is around 0.88.

However coverage and width are pairwise results, closely correlated for each individual. A higher coverage might happen at the cost of higher width instead of a better model. Drawing conclusions about their performance separately is not appropriate, so they should be considered together. Results of some selected patients are shown in Figure 5.9, with the results of the Ensemble and Single tree methods for a single patient joined by a dashed line for clarity. For example, for patient 34, the Ensemble tree method has better coverage but at the price of wider intervals. But for patients 45 and 46, the Single tree method has better coverage with narrower intervals. For patient 26, they share nearly the same coverage, but the Ensemble tree method has a smaller width.

We consider how often each method is better than the other in terms of coverage and width. One method is superior if it gives better coverage and width, inferior if it has worse coverage and width, and the two methods represent a trade-off if one has better coverage and the other has better width. To compare the methods, the proportions of patients in each category in shown in Table 5.4. It is clear that the single-tree method is the best as it has substantially more patients where there is both higher coverage and narrower intervals (32.62%). The Ensemble tree method is only superior in about 5% of patients. When neither method is clearly

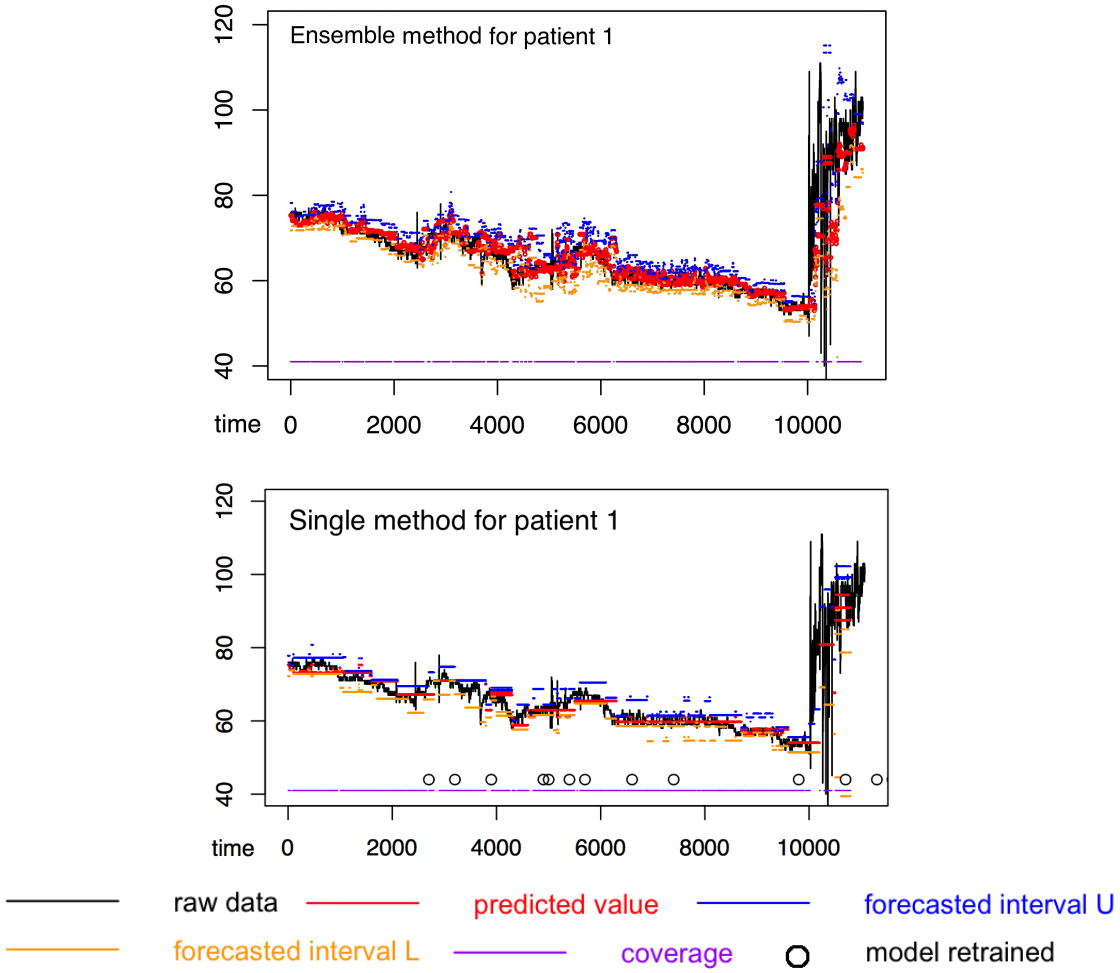


Figure 5.6: Data and monitoring forecasts using regression trees for liver transplantation on one patient.

Table 5.4: The relative performances of Ensemble and Single tree methods as percentages of the 325 patients.

		Narrower intervals	
		Ensemble method	Single method
Higher coverage	Ensemble method	5.23%	44.62%
	Single method	17.54%	32.62%

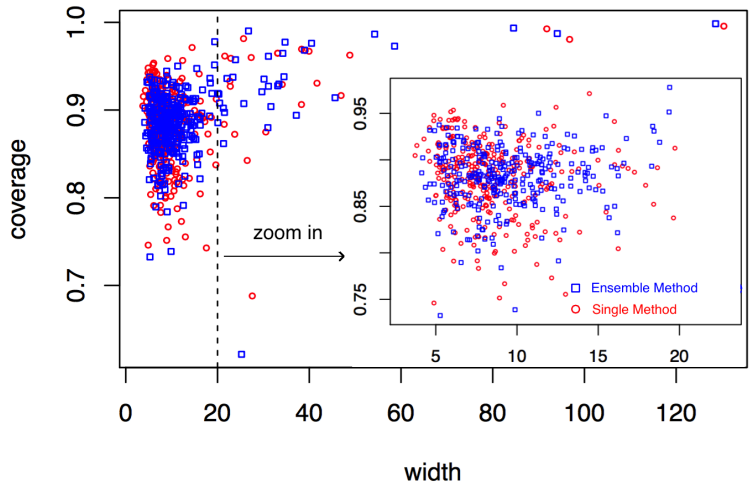


Figure 5.7: Summary results of forecasts for all 325 patients. Each point represents mean coverage and width for one patient. The inset plot is a zoom in of the larger plot.

better, the Ensemble tree method is more likely to have higher coverage, while the Single tree method is more likely to have narrower intervals. Ignoring the interval width, each method has better coverage in about 50% of cases.

In terms of the computation time required, shown in Figure 5.10, the computation time has a roughly linear relationship with time series length T_n. The fitted regression slope coefficients of 0.016 (Ensemble tree method) and 0.001 (Single-tree method) indicate that ensemble method takes around 16 times as much computation as the single-tree method. However, since new observations arrive at intervals of about 0.6–1 second (per heart beat generally takes 0.6 to 1 second), the computation can easily be done online in real time.

The variables used in the trees include both scaling coefficients and wavelet coefficients at high and low resolution levels using both short and long lag information. That means heart rate prediction needs both averages and contrasts with long and short time interval information. In our case, averages were taken over time spans between 2 and 256 heartbeats. Prediction by only using short term information may not have a good performance.

5.5.2 Comparison to ARIMA

In this section, we use the real data in an ARIMA model to compare its performance with that from trees. The parameters are the same as those in the simulation in Section 5.4.1. Results are shown in Figure 5.11 and Table 5.5. Ensemble tree and Single tree methods have coverage $(86.82\%, 90.46\%)$, widths

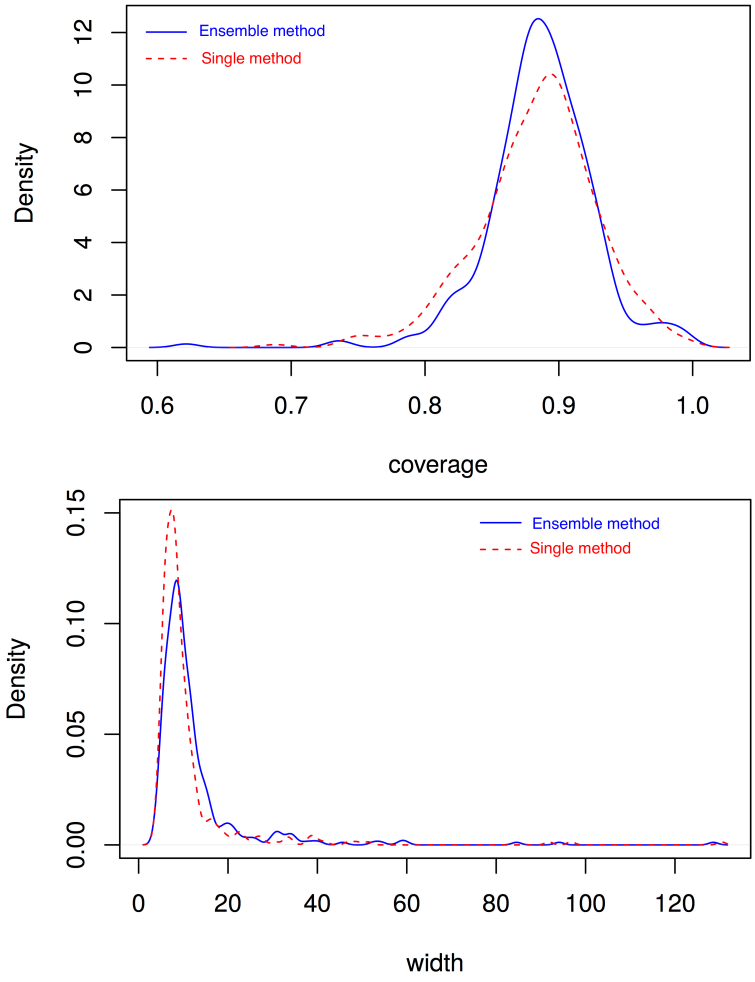


Figure 5.8: Kernel density plots of coverage (left) and mean width (right) for the full set of 325 patients.

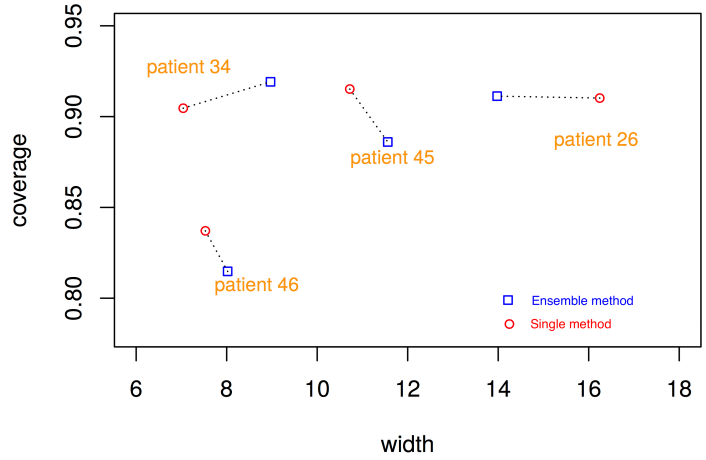


Figure 5.9: Coverage and mean interval width of Ensemble and Single tree methods for selected patients 26, 34, 45 and 46.

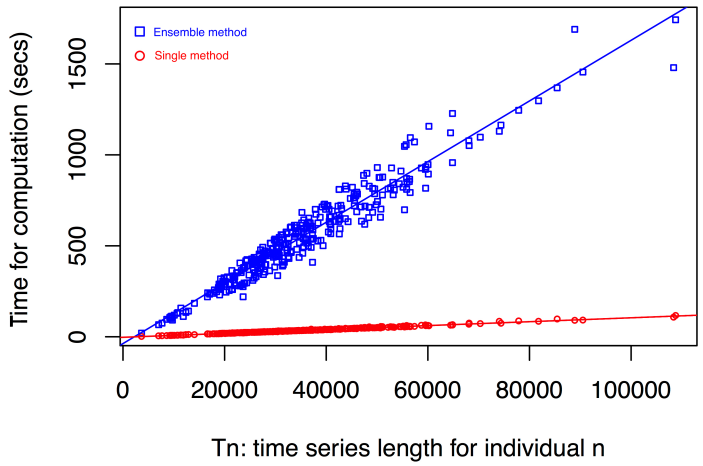


Figure 5.10: The computational time (in seconds) for Ensemble and Single tree methods for all 325 patients. The fitted linear regression slope coefficients are 0.016 and 0.001 for Ensemble and Single tree respectively.

$(9.98, 10.56)$, and computing times $(240, 65)$ for patient 1.

Table 5.5: Tree and ARIMA results: mean and standard deviation (sd) of coverage, width and time for each Method over 325 patients.

Method	Coverage		Width		Time	
	mean	sd	mean	sd	mean	sd
Tree						
Single	0.8837	0.0447	11.05	11.72	35.25	16.81
Ensemble	0.8861	0.0405	12.53	12.08	553.71	267.53
ARIMA						
Single	0.8853	0.0286	10.93	8.32	466.07	256.33
Ensemble	0.8732	0.0300	10.38	7.956	1032.20	525.58

For ARIMA, it is hard to distinguish whether the Ensemble or Single method is better in terms of coverage and width. The Single method is a little bit higher in both coverage and width, but clearly less computationally demanding. But they all face a problem: when concept drift in the trend occurs, the old model is no longer suitable for future prediction (especially when d is not 0). But for tree based models, such situations do not occur as tree based model does not consider trend. When we do a pairwise comparison, the results are shown in Table 5.6.

The single-model method (7.71%) is superior slightly more often than the ensemble method (5.03%). When neither method is clearly better, the ensemble method is more likely to have narrower intervals, while the single-model method is more likely to have higher coverage. In practice, the single-model method is proposed as the performances are comparable but is less computationally demanding.

Now we compare the single regression tree and single-model ARIMA approaches.

Table 5.6: The relative performances of ARIMA methods as percentages of the 325 patients.

		Narrower intervals	
		ensemble method	single method
Higher coverage	ensemble method	5.03%	7.38 %
	single method	79.87%	7.71%

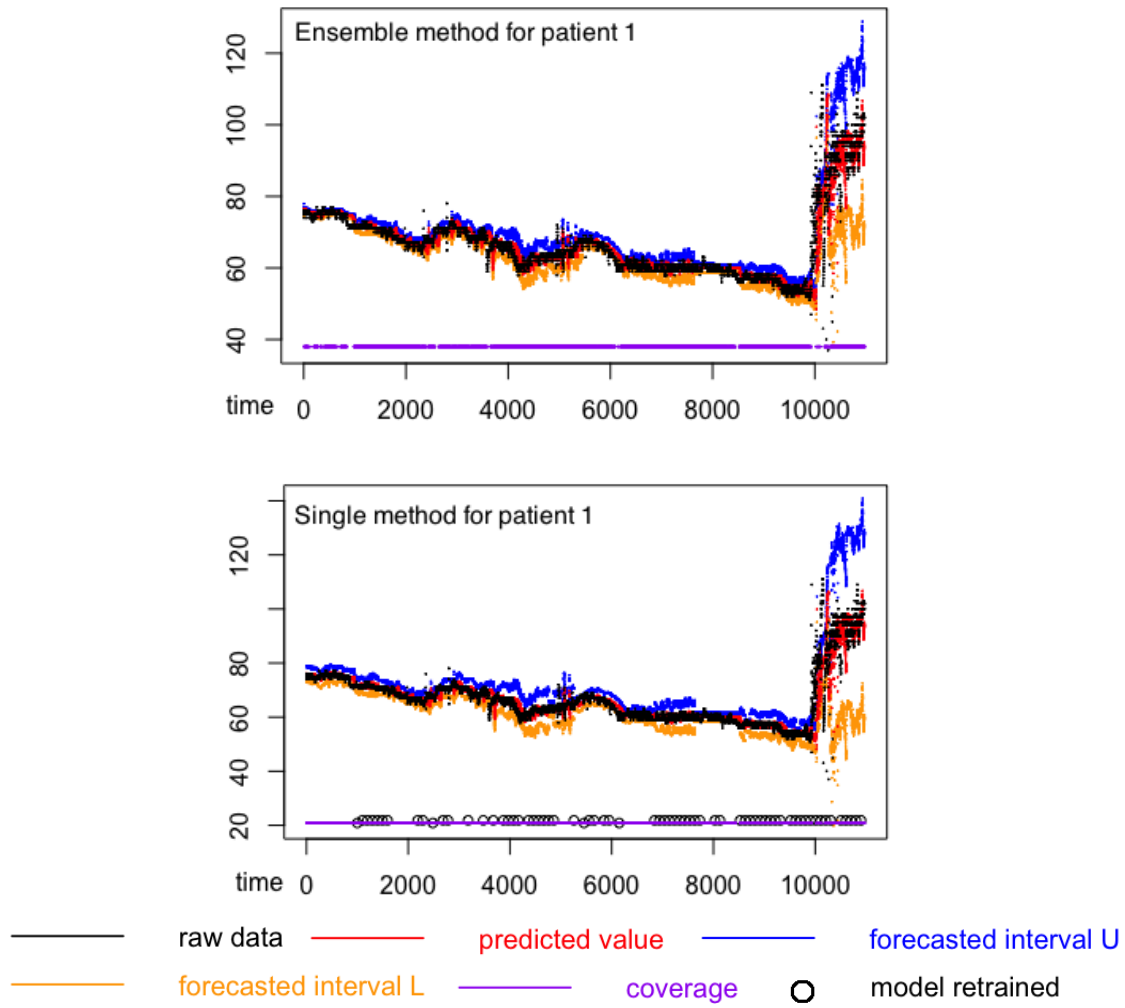


Figure 5.11: Data and monitoring forecasts using ARIMA models for liver transplantation on patient 1.

Table 5.7: The relative performances of single-forecast approaches for regression trees and ARIMA as percentages of the 325 patients.

		Narrower intervals	
		ARIMA	Ctree
Higher coverage	ARIMA	12.62%	38.77 %
	Ctree	33.85%	14.77%

The results are shown in Table 5.7 and Figure 5.12. The tree based approach is superior slightly more often than the ARIMA method (14.77% vs. 12.62%). So roughly, their performance is similar in coverage and width although the tree method is somewhat higher in standard deviation as shown in Table 5.5. However, in terms of computational time, the tree method obviously outperforms ARIMA. So in conclusion, we would prefer the single-tree based method.

5.6 Forecasting stock price

In order to compare tree based methods with more sophisticated time series models, in this section, we compare tree-based model with ARMA-GARCH in stock price forecasting. The stock we choose is Shanghai Stock Exchange Composite Stock Price Index (SSE Index), which is a stock market index of all stocks (A shares and B shares) that are traded at the Shanghai Stock Exchange. It is a relatively long time series with both stationary and non-stationary phases as well as many distribution change points. The time series dates from 19th, December, 1990 to 1st, June, 2018, making a total of 6713 closing price observations excluding market closing days like weekends and holidays. Since the time series x_t is not stationary, we take logs and do first order differencing to reduce variance fluctuation, so the variable in use is

$$y_t = \log(x_{t+1}) - \log(x_t)$$

The parameters chosen for trees are $\beta = 2$, $\alpha = 0.2$, $b = 0.99$, $a = 0.95$, $h = 100$, $t_0 = 1311$, $\gamma = 12$, $\delta = 2$, $T_\Omega = 1000$, and $m = 3$. For ARMA-GARCH, the maximum value for p, q, r and s are all 2, with $\gamma = 300$. Results are shown in Table 5.8 and Figure 5.13. Results show that, for SSE Index time series, ensemble ARMA-GARCH is definitely better than single tree method with higher coverage and lower width. For the rest comparison of the methods, ARMA-GARCH share similar performance with tree methods, and higher coverage is at the cost of higher width. But in terms of computational time, the ensemble tree method is preferred.

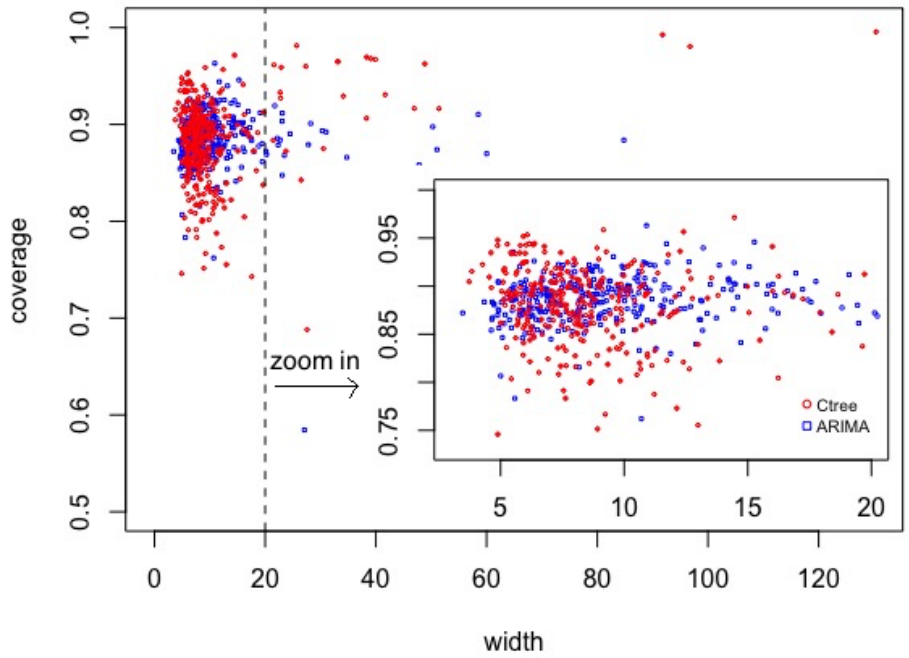


Figure 5.12: The relative performances of single-forecast approaches for regression trees and ARIMA across all 325 patients.

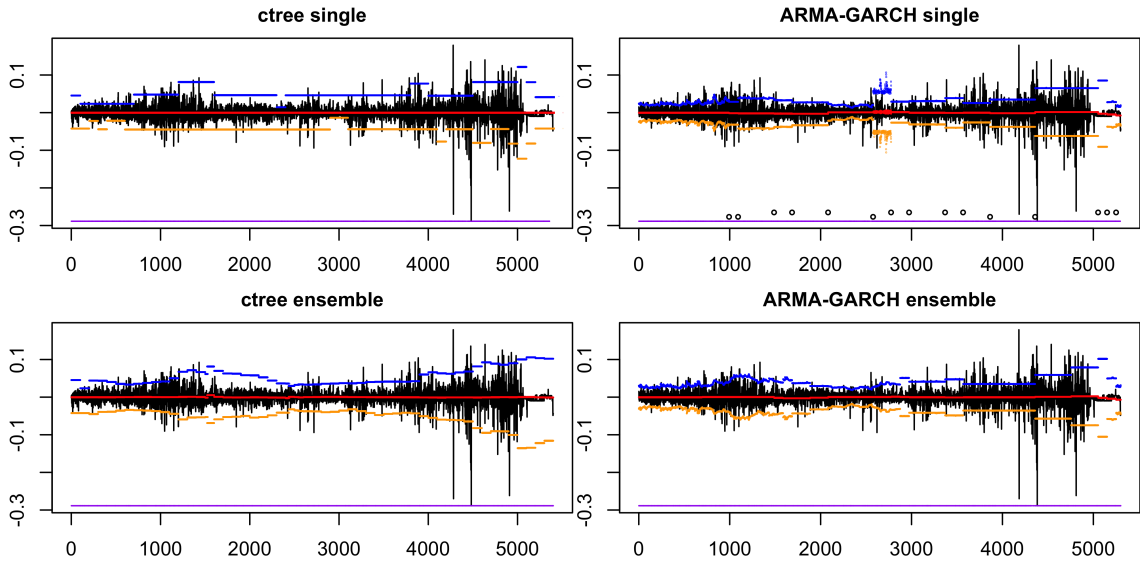


Figure 5.13: The performances of single and ensemble methods for regression trees and ARMA-GARCH using SSE Index data.

Table 5.8: The performance of tree methods and ARMA-GARCH methods using SSE Index data.

Method	coverage(%)	width(0.01)	widthsd(0.01)	time
Tree				
single	93.58%	9.84	3.4	6.06
ensemble	96.70%	11.15	4.7	38.45
ARMA-GARCH				
single	90.93%	7.25	3.1	302.36
ensemble	93.94%	8.40	3.1	598.86

5.7 Conclusion

In this chapter, two tree-based methods are proposed to deal with forecasting in a streaming data context. In contrast to many alternative methods, we pay more attention to forecast intervals than point forecasts, although the point forecasts are essential to adaptively adjust the forecast intervals for the current model's accuracy. This adaptation is accomplished by updating the forecast interval to capture the stream trend using root mean square error calculated from the most recent batch of data, so as to update the model for future prediction.

Rather than fixing a time interval of historical data to use in forecasting, we use wavelet transform to capture long term variable information. The maximal overlap wavelet transform decomposes the original time series into different resolution levels to capture averages and fluctuations over a range of time scales. This means that the tree construction algorithms can choose whichever aspects of the information in the data are most useful. Moreover, we gain the benefit of allowing long time spans of historical data to be used in the models without requiring they all be present as separate explanatory variables, so that the information in use will not be constrained to the most recent batch.

When applied to simulated data and real data, tree based methods are generally better than or similar to ARIMA, GARCH and ARMA-GARCH except when ARIMA effect is strong and trend possible. For model ARIMA, when trend disappears, ARIMA will still forecast with trend before retraining thus the forecasted intervals can be extremely wide. When there is no trend, but a trend is falsely detected by ARIMA, intervals can be wide as well. But tree methods, which consider trend in a different way, can avoid such situations. So because of this possible false trend, we suggest using tree methods even when ARIMA effect is strong.

When ARIMA effect is strong without trend, the ensemble tree method is somewhat better than the single tree method. Generally the ensemble method has a bigger width and higher coverage while single tree method has a smaller width and lower coverage. But they have roughly the same performance. When time efficiency is required, the single tree method is suggested, otherwise the ensemble tree method is preferred.

Chapter 6

Theory exploration for decision tree based linear fitting

After applying decision trees to regression problems in Chapters 4 and 5, we now explore the performance of trees when fitted to data generated from a linear model. The corresponding bias, variance, and prediction error between the fitted simplified tree and the true simple linear model will be calculated. Then how those errors vary will be explored when the linear data distribution changes. The motivation is to explore how the trees perform under different distributions. Afterwards, prediction interval is proposed using Gaussian and quantile intervals, which explains why quantile interval is chosen in Chapter 5. The simple linear model in use is

$$Y = \alpha + \beta X + \epsilon,$$

where $f(X) = \alpha + \beta X$ is the true model. It is supposed that, though out this chapter, $X \sim U(a, b)$ independently, $\epsilon \sim N(0, \sigma^2)$.

6.1 The Bias-Variance Decomposition

The expected squared prediction error (SPE) is one of the important metrics to measure how well the trained model be applied to further unseen data. As shown in Hastie *et al.* (2001), SPE of a regression fit $\hat{f}(X)$ at an input point $X = x_0$ is

$$\begin{aligned}
\text{SPE}(x_0) =& E[\left(Y - \hat{f}(x_0)\right)^2 | X = x_0] \\
=& \sigma^2 + [E\hat{f}(x_0) - f(x_0)]^2 + E[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\
=& \sigma^2 + \text{Bias}^2\left(\hat{f}(x_0)\right) + \text{Var}\left(\hat{f}(x_0)\right) \\
=& \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}.
\end{aligned}$$

The first term is the variance of the target around its true mean $f(x_0)$, and cannot be avoided no matter how well the $f(x_0)$ is estimated, unless $\sigma^2 = 0$. The second term is the squared bias, the amount by which the average of the estimate differs from the true mean; the last term is the variance; the expected squared deviation of $\hat{f}(x_0)$ around its mean. Typically the more complex the model $\hat{f}$ is, the lower the (squared) bias but the higher the variance (Hastie *et al.*, 2001) will be.

6.1.1 Decomposition background

For the i^{th} observation Y_i, the (unconditional) expectation is

$$\begin{aligned} E(Y_i) =& E(\alpha + \beta X_i + \epsilon_i) \\ =& \alpha + \beta E(X_i) + E(\epsilon_i) \\ =& \alpha + \beta \cdot \frac{a+b}{2}, \end{aligned}$$

and the variance is

$$\begin{aligned} \mathrm{Var}(Y_i) =& \mathrm{Var}(\alpha + \beta X_i + \epsilon_i) \\ =& \beta^2 \mathrm{Var}(X_i) + \mathrm{Var}(\epsilon_i) \\ =& \beta^2 \cdot \frac{(b-a)^2}{12} + \sigma^2. \end{aligned}$$

They both have no relationship to i. In that case, $E(Y) = E(Y_i)$ and $\mathrm{Var}(Y) = \mathrm{Var}(Y_i)$. So for N observations, the expectation and variance for the average $\bar{Y}$ are

$$\begin{aligned} E(\bar{Y}) =& \frac{1}{N}(E(Y_1) + E(Y_2) + \cdots + E(Y_N)) \\ =& E(Y) \\ =& \alpha + \beta \cdot \frac{a+b}{2} \end{aligned}$$

and

$$\begin{aligned} \mathrm{Var}(\bar{Y}) =& \frac{1}{N^2}(\mathrm{Var}(Y_1) + \mathrm{Var}(Y_2) + \cdots + \mathrm{Var}(Y_N)) \\ =& \frac{1}{N}\mathrm{Var}(Y) \\ =& \frac{1}{N}\{\beta^2 \cdot \frac{(b-a)^2}{12} + \sigma^2\}. \end{aligned}$$

6.1.2 Decomposition in the context of decision trees

In the context of decision trees, the fitted model is $\hat{f}(X)$ in a simplified form

$$\hat{f}(X) = \bar{Y}^i, \;\; i = 1, 2, ..., k$$

where k is the number of terminal nodes in the tree $\hat{f}(X)$, and $\bar{Y}^i$ is the mean of y in terminal node i. In a tree with only the root node, $k = 1$, and the fitted model is $\hat{f}(X) = \bar{Y}$. Then for point x_0,

$$\begin{aligned}
\text{Bias}^2\left(\hat{f}(x_0)\right) =& [E\{\hat{f}(x_0)\} - f(x_0)]^2 \\
=& [\alpha + \beta \cdot \frac{a+b}{2} - \alpha - \beta \cdot x_0]^2 \\
=& \beta^2 \left(\frac{a+b}{2} - x_0\right)^2,
\end{aligned}$$

and the variance is

$$\begin{aligned}
\text{Var}\left(\hat{f}(x_0)\right) =& E[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\
=& E[\bar{Y} - \left(\alpha + \beta \cdot \frac{a+b}{2}\right)]^2 \\
=& E\left(\bar{Y}^2\right) + \left(\alpha + \beta \cdot \frac{a+b}{2}\right)^2 - 2E\left(\bar{Y}\right)\left(\alpha + \beta \cdot \frac{a+b}{2}\right) \\
=& E^2\left(\bar{Y}\right) + \text{Var}\left(\bar{Y}\right) - \left(\alpha + \beta \cdot \frac{a+b}{2}\right)^2 \\
=& \frac{\sigma^2}{N} + \beta^2 \frac{(b-a)^2}{12N}.
\end{aligned}$$

So the SPE at point x_0 is

$$\begin{aligned}
\text{SPE}\left(\hat{f}(x_0)\right) =& \sigma^2 + \text{Bias}^2\left(\hat{f}(x_0)\right) + \text{Var}\left(\hat{f}(x_0)\right) \\
=& \sigma^2 + \beta^2 \left(\frac{a+b}{2} - x_0\right)^2 + \frac{\sigma^2}{N} + \beta^2 \frac{(b-a)^2}{12N} \\
=& \left(1 + \frac{1}{N}\right)\sigma^2 + \beta^2 \left(\frac{a+b}{2} - x_0\right)^2 + \beta^2 \frac{(b-a)^2}{12N}.
\end{aligned}$$

Then the mean squared prediction error (MSPE) is

$$\begin{aligned}
\text{MSPE} &= \int_a^b \text{SPE}\left(\hat{f}\left(x\right)\right) P\left(X = x\right) dx \\
&= \int_a^b \text{SPE}\left(\hat{f}\left(x\right)\right) \frac{1}{b-a} dx \\
&= \left(1 + \frac{1}{N}\right) \sigma^2 + \beta^2 \frac{\left(b-a\right)^2}{12N} + \beta^2 \int_a^b \left(\frac{a+b}{2} - x\right)^2 \frac{1}{b-a} dx \\
&= \left(1 + \frac{1}{N}\right) \sigma^2 + \beta^2 \frac{\left(b-a\right)^2}{12N} + \beta^2 \frac{1}{12} \left(b-a\right)^2 \\
&= \left(1 + \frac{1}{N}\right) \left(\sigma^2 + \beta^2 \frac{\left(b-a\right)^2}{12}\right),
\end{aligned}$$

comprising variance

$$E\left(\text{Var}\right) = \frac{1}{N} \left(\sigma^2 + \beta^2 \frac{\left(b-a\right)^2}{12}\right)$$

and squared bias as

$$E\left(\text{Bias}^2\right) = \beta^2 \frac{\left(b-a\right)^2}{12}.$$

Now the number of terminal nodes in the decision tree is extended from $k = 1$ to a general k, then the MSPE, Bias^2 and variance for $x \in [a, b]$ equals to that for $x \in [a, (a + (b-a)/k)]$ since the decision tree is assumed to make k equal terminal nodes with the same number of observations in each terminal node. In that case, for $x \in [a, b]$ for a general k, the MSPE is

$$\text{MSPE} = \left(1 + \frac{k}{N}\right) \left(\sigma^2 + \beta^2 \frac{\left(b-a\right)^2}{12k^2}\right)$$

with variance as

$$E\left(\text{Var}\right) = \left(\frac{k}{N}\right) \left(\sigma^2 + \beta^2 \frac{\left(b-a\right)^2}{12k^2}\right)$$

and squared bias as

$$E\left(\text{Bias}^2\right) = \beta^2 \frac{\left(b-a\right)^2}{12k^2}. \tag{6.1}$$

It is easy to see that with a lower $|\beta|$, $b-a$, σ^2 and higher N, variance, squared bias and MSPE will all decrease.

6.1.3 Optimal k to minimise MSPE

The ideal number of terminal nodes can be found by minimising the MSPE with aspect to k. Here k is a discrete integer, so the target k will be the nearest integer from the differentiate result. Calculating the first derivative of MSPE, we get

$$\frac{d\text{MSPE}\,(k)}{d\,(k)} = \frac{\sigma^2}{N} - \frac{\beta^2\,(b-a)^2}{6k^3} - \frac{\beta^2\,(b-a)^2}{12Nk^2},$$

and the second dierivative of MSPE is always positive. So we only need to solve

$$\frac{d\text{MSPE}\,(k)}{d\,(k)} = 0,\ k \in [1, N]. \tag{6.2}$$

The real root of Equation (6.2) is

$$k_{min} = \sqrt[3]{\frac{\beta^2\,(b-a)^2}{12\sigma^2}} \cdot \left(\sqrt[3]{N + \sqrt{N^2 - \frac{\beta^2\,(b-a)^2}{324\sigma^2}}} + \sqrt[3]{N - \sqrt{N^2 - \frac{\beta^2\,(b-a)^2}{324\sigma^2}}} \right). \tag{6.3}$$

Having

$$N \gg \frac{\beta\,(b-a)}{18\sigma},$$

then k_{min} can be approximated by

$$k_{min} \approx \sqrt[3]{\frac{\beta^2\,(b-a)^2\,N}{6\sigma^2}}.$$

In addition, the constraint for root k is also $k_{min} \in [1, N]$. If k_{min} is not in $[1, N]$, MSPE might always decrease.

By substituting k_{min} in Equation (6.3)back into Equation (6.1), we will get

$$E\left(\text{Bias}^2\right) = 48\left(12\sigma^2\beta\,(b-a)\,/N\right)^{2/3},$$

and it is easy to see, with the increase of σ and $\beta\,(b-a)$ when N is fixed, $E\left(\text{Bias}^2\right)$ will increase. For the others, they will be shown as figures.

So how will the ratios $E\,(\text{Var})\,/\text{MSPE}$, $E\left(\text{Bias}^2\right)/\text{MSPE}$, σ^2/MSPE vary when parameters change? Since a, b and β appear together, they are regarded as one parameter. For b and a, the thing matters is their difference, so, we use $a = 0$ and only change b. Here k is set to be k_{min} calculated using given parameters for Equation (6.3) and if k_{min} does not exist, the results will not be shown. The results in Figures 6.1 (changing $\beta^2(b-a)^2$) and 6.2 (changing σ^2) show that, under both circumstances, MSPE, $E\,(\text{Var})$ and $E\left(\text{Bias}^2\right)$ all increase.

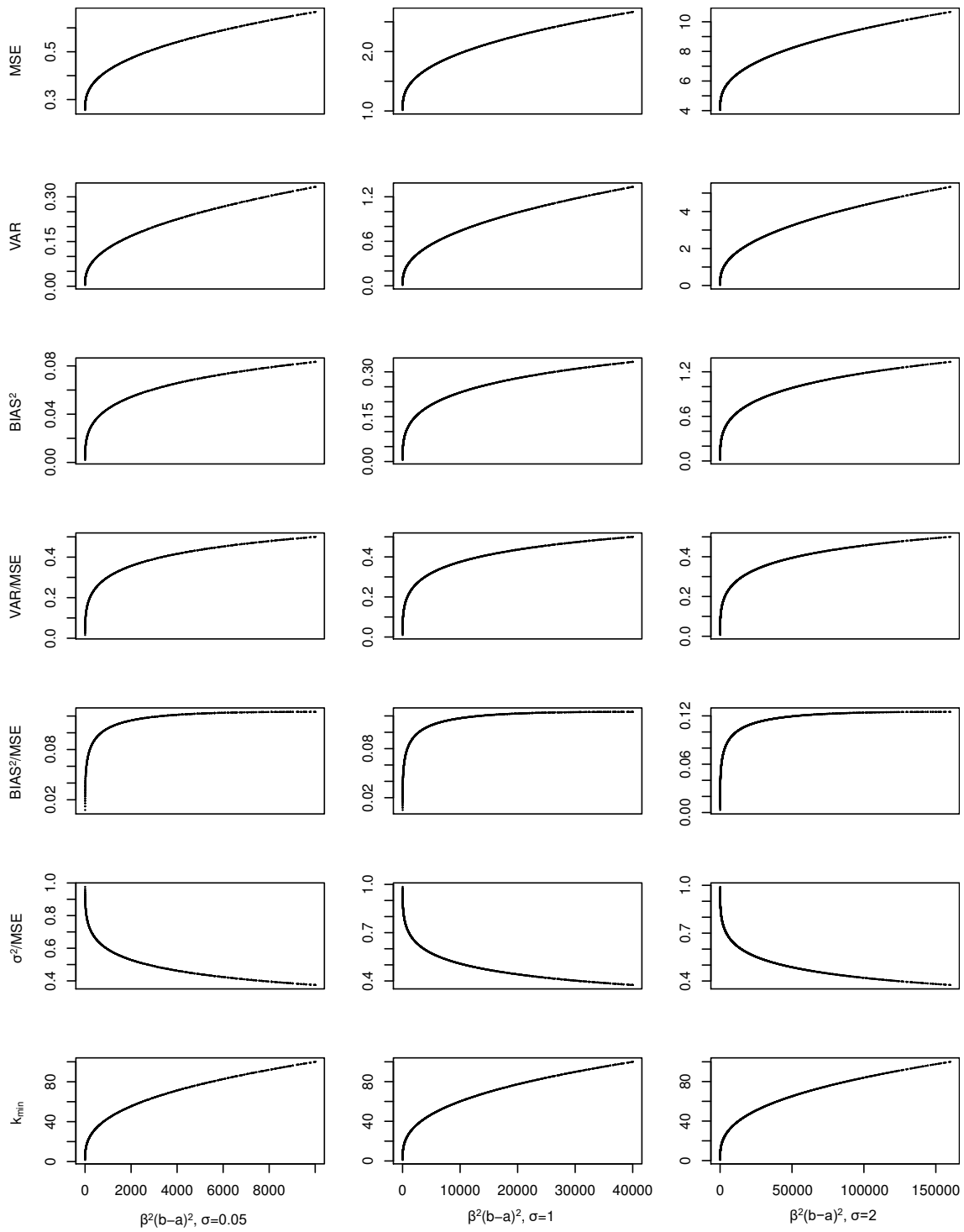


Figure 6.1: Ratios $E\left(\text{Var}\right)/\text{MSPE}$, $E\left(\text{Bias}^2\right)/\text{MSPE}$, σ^2/MSPE with different $\beta^2\left(b-a\right)^2$. $N=100$ and $\alpha=0$.

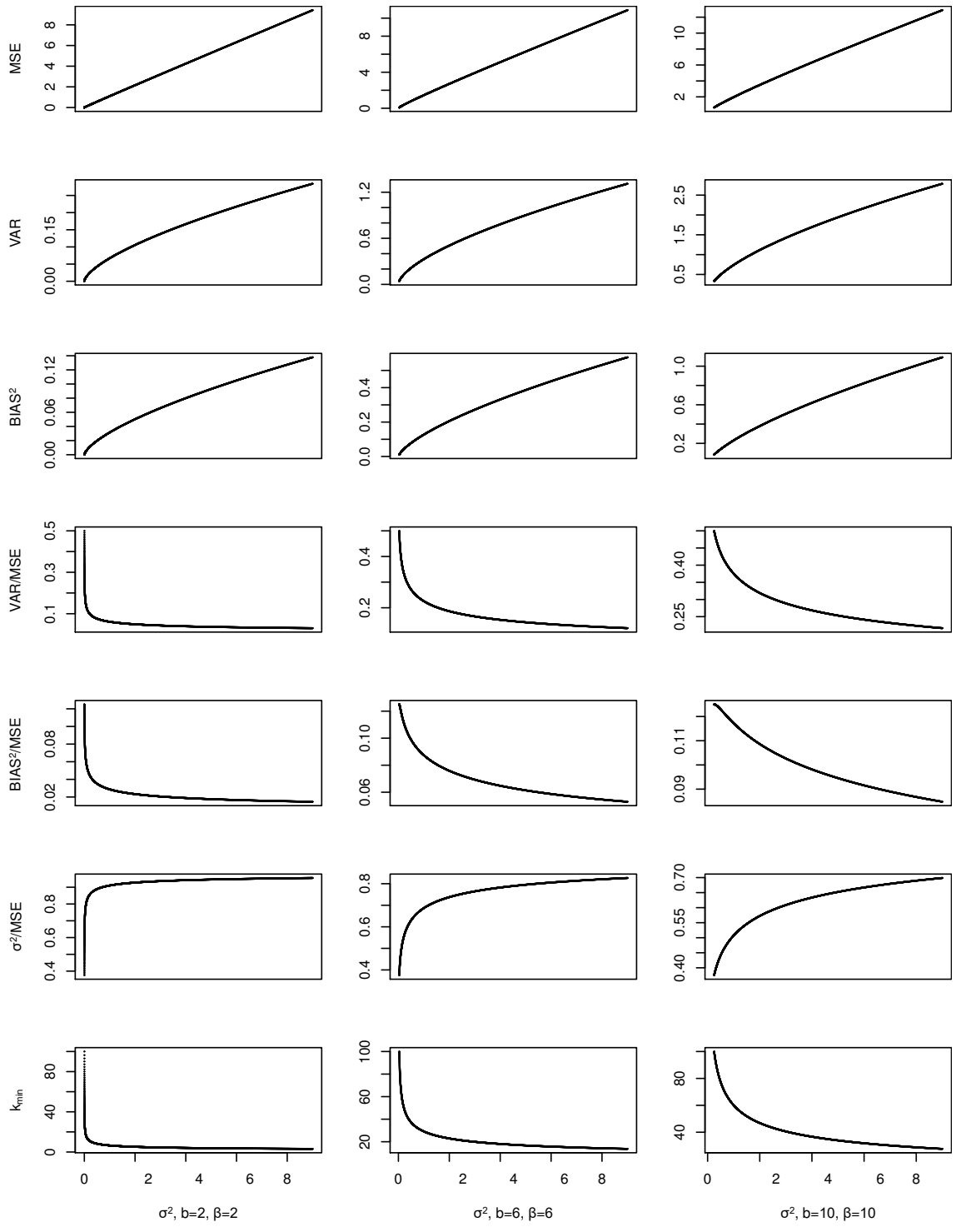


Figure 6.2: Ratios $E\left(\text{Var}\right)/\text{MSPE}$, $E\left(\text{Bias}^2\right)/\text{MSPE}$, σ^2/MSPE with different σ^2. $N = 100$ and $\alpha = 0$.

In Figure 6.1, when $\beta^2 (b-a)^2$ gets bigger, X is more likely to be uniformly distributed and k_{min} increases as y is more accurately described with a uniform distribution, and the ratio of Var, Bias^2 over MSPE gets larger while σ^2 increases. In Figure 6.2, when σ^2 gets bigger, the Gaussian distribution will play a bigger role in data generation and k_{min} decreases. That is why σ^2/MSPE increases. For $E(\text{Var})$ /MSPE, $E\left(\text{Bias}^2\right)$ /MSPE, they generally decrease. The decrease speed slows with bigger b and β as expected.

6.1.4 Simulation

In this simulation, a simplified tree model will be designed to confirm the theory results using simulated data. That is when parameters of the simulated data change, the distribution of X and y will also change. The question is how the statistics of Var, Bias^2, MSE, and k_{min} change accordingly.

In the simplified tree, X is evenly split into k intervals, $i = 1, 2, \ldots, k$. For specific k, a, b, N, α, and β, we are going to calculate the statistics of MSPE, Var and Bias^2 for the i^{th} interval in k from simulated data. So for the i^{th} interval, the x range is

$$R_i = [a + (i-1)(b-a)/K,\ a + (i)(b-a)/K].$$

The number of observations in interval i $(i = 1, 2, ...k-1)$ is n_i

$$n_i = \lfloor \left(N - \sum_{j=0}^{i-1} n_j \right) / (k-i) \rfloor,$$

defining $n_0 = 0$ and $n_k = N - \sum_{j=0}^{k-1} n_j$.

- **Step 1:** For the data $(x,\ y)$ in R_i, we train a model from them as

  $$\hat{f}_i(x) = \bar{y}$$

  for simulated y and $\bar{y}$ is the averaged value of y in R_i.

- **Step 2:** Repeat **Step 1** s times. Then we have s trained models $\{\hat{f}_j(x)\}$, $j = 1, 2, ..., s$.

- **Step 3:** Simulate one x_0 uniformly from the x range R_i. We are going to calculate the SPE(x_0), Var(x_0) and Bias$^2(x_0)$ for this specific x_0.

- **Step 4:** Simulate s values of y_j using x_0.

- **Step 5:** Calculate the statistics of SPE (x_0), Var (x_0) and Bias^2 (x_0) for this specific x_0 as

$$\text{SPE}\,(x_0) = \frac{1}{s}\sum_{j=1}^{s}\left(\hat{f}_i\,(x_0) - y_j\right)^2$$

$$\text{Bias}^2\,(x_0) = \{\frac{1}{s}\sum_{j=1}^{s}\hat{f}_i\,(x_0) - f\,(x_0)\}^2$$

$$\text{Var}\,(x_0) = \text{variance}\left(\hat{f}_i\,(x_0)\right).$$

- **Step 6:** Repeat **Step 3** to **Step 5** for $n.repeat$ times and calculate the mean of SPE (x_0), Var (x_0) and Bias^2 (x_0) as MSPE$_i$, Bias$_i^2$ and Var$_i$.

Do **Step 1** to **Step 6** for all i, $i = 1, 2, ..., k$ and calculate the mean as MSPE, Bias^2 and Var.

The results of simulations with 200 trials are shown in Figure 6.3 and Figure 6.4. For Figure 6.3, $k_{min} \in [1, N]$, we have a minimum MSPE. But when $k_{min} \notin [1, N]$ as in Figure 6.4, MSPE keeps decreasing.

6.2 Prediction interval

Instead of point prediction, a prediction interval is also desirable especially for time series with high variance. If both the point prediction as well as the prediction interval can be provided, we will be more confident for the prediction. This study also helps us decide the proper prediction interval method for Chapter 5. Gaussian based prediction interval and quantile interval are compared under different parameters distributions.

6.2.1 Probability function of Y

Since our linear model

$$Y = \alpha + \beta X + \epsilon$$

is the sum of uniform and Gaussian distributions, the probability function for Y is

$$\begin{aligned} P_Y\,(y) &= \int_a^b P\,(Y = y|X = x)\,P_X\,(x)\,dx \\ &= \int_a^b \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(y-(\alpha+\beta x))^2}{2\sigma^2}\right\}\frac{1}{b-a}dx. \end{aligned}$$

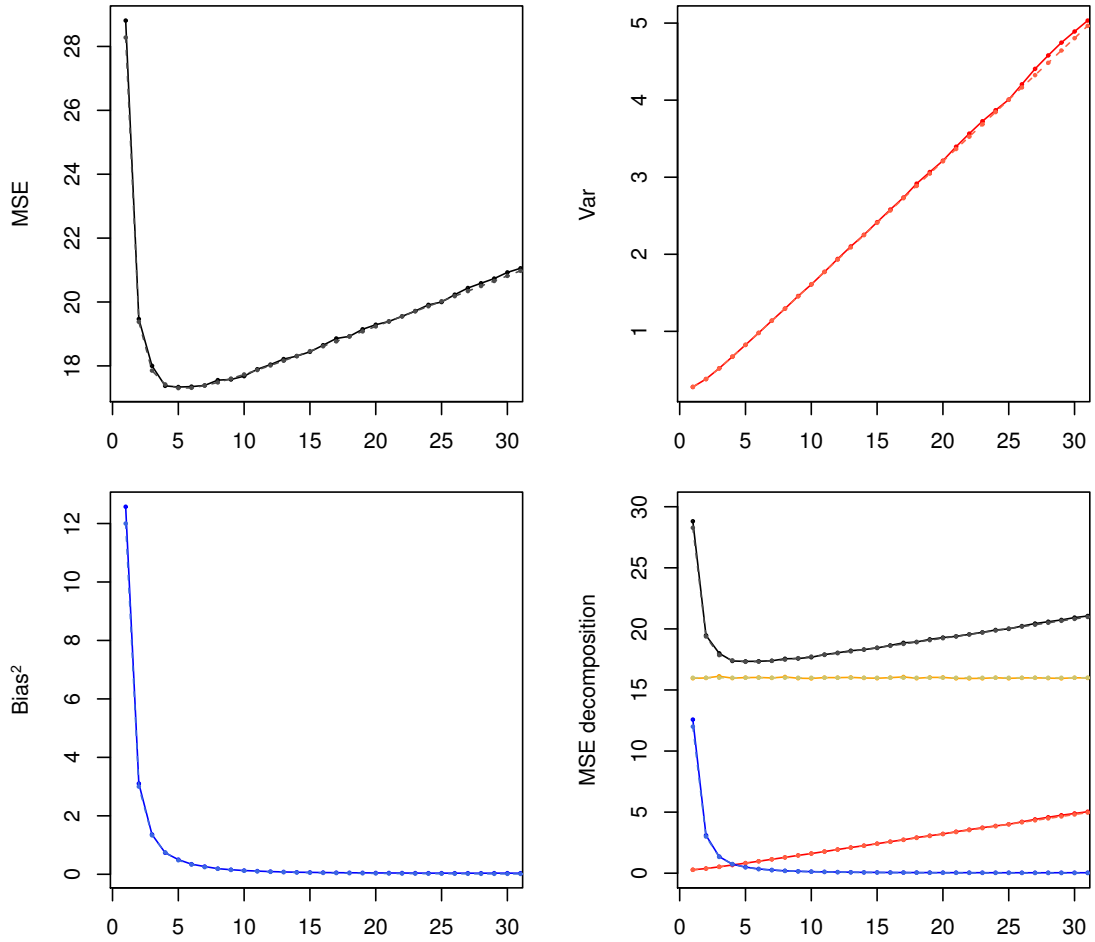


Figure 6.3: An example when k_{min} exists. The x axis label is k: the number of tree splits. MSPE, Bias^2 and Var are averaged calculations from 200 simulation trials. Black line is the MSPE, blue line is the Bias^2, orange line is σ^2 and red line is the Var. Solid lines are from simulated data and dashed lines are theoretical calculations. The parameters values are given as: $\alpha = 2$, $\beta = 4$, $N = 100$, $a = 0$, $b = 3$, $\sigma = 4$.

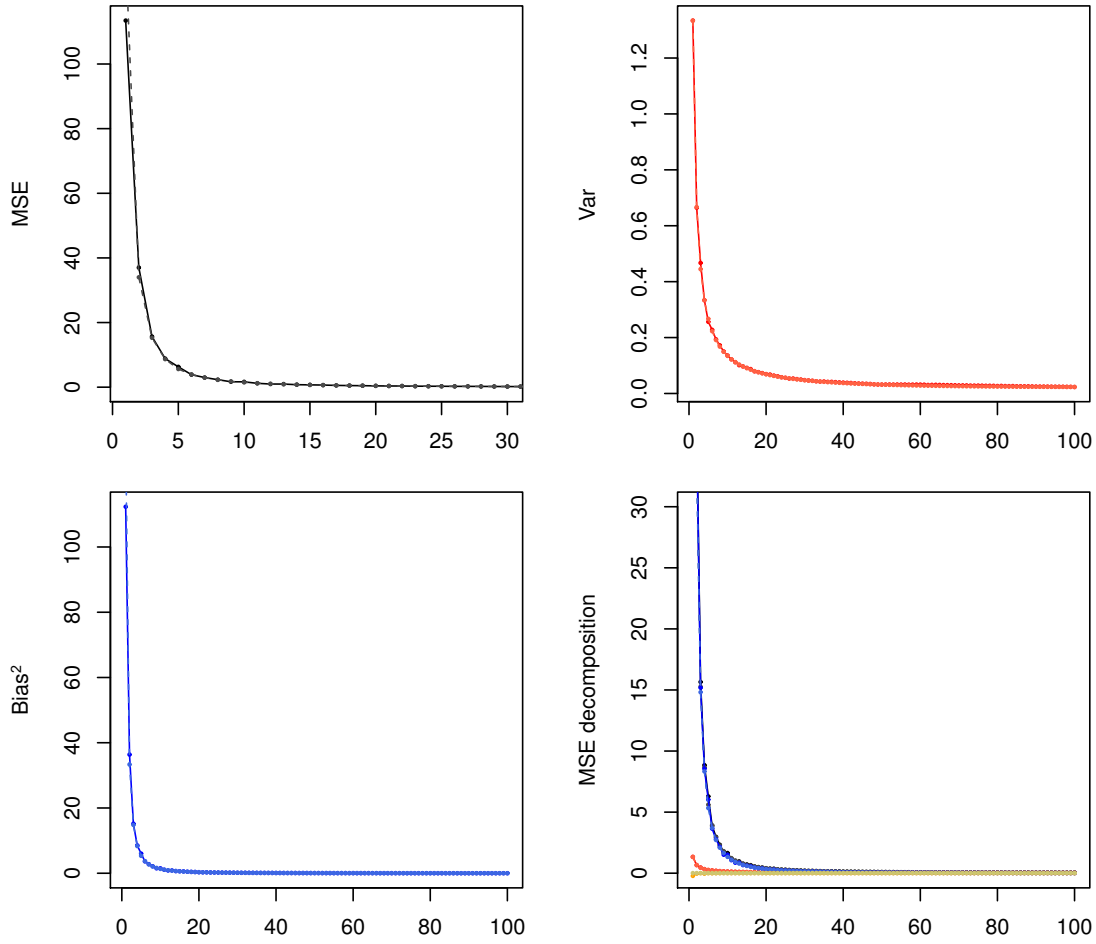


Figure 6.4: An example when k_{min} does not exist. The x axis label is k: the number of tree splits. MSPE, Bias^2 and Var are averaged calculations from 200 simulation trials. Black line is the MSPE, blue line is the Bias^2, orange line is σ^2 and red line is the Var. Solid lines are from simulated data and dashed lines are theoretical calculations. The parameters values are given as: $\alpha = 2$, $\beta = 12$, $N = 100$, $a = 0$, $b = 4$, $\sigma = 0.2$.

By letting $t = (\alpha + \beta x - y)/\sigma$, we obtain

$$\begin{aligned} P_Y(y) =& \frac{1}{\beta(b-a)} \int_{\frac{\alpha+\beta a-y}{\sigma}}^{\frac{\alpha+\beta b-y}{\sigma}} \frac{1}{\sqrt{2\pi}} \exp\{-\frac{t^2}{2}\} dt \\ =& \frac{1}{\beta(b-a)} \left[\Phi\left(\frac{\alpha+\beta b-y}{\sigma} \right) - \Phi\left(\frac{\alpha+\beta a-y}{\sigma} \right) \right]. \end{aligned}$$

Now we get the probability of Y. But $P_Y(y)$ is in a complex form meaning that the parameters are not easily solvable in theory by a given value for $P_Y(y)$.

6.2.2 Prediction interval as a Gaussian distribution

If we want to get the prediction interval, say $[y_1, y_2]$ for Y at $(1-p)$ level, the theoretical way is to obtain y_1 and y_2 from the equations

$$\int_{-\infty}^{y_1} P_Y(y)\, dy = \frac{p}{2} \text{ and } \int_{y_2}^{\infty} P_Y(y)\, dy = 1 - \frac{p}{2}.$$

However, the integral of Φ is not analytically solvable without approximating Φ with other suitable expressions. The results will also be quite complex. If we know the parameters values, then y_1 and y_2 can easily be found numerically.

From Figure 6.5, if the uniform (Gaussian) distribution plays a main role, then Y can be approximately described by a uniform (Gaussian) distribution. Under the conditions that, β is not too large and σ is not too small, and k is 1 (with only one interval), we will approximate the distribution of Y as a Gaussian distribution $N(\mu_Y, \sigma_Y^2)$:

$$Y \sim N\left(\alpha + \beta \cdot \frac{a+b}{2},\ \beta^2 \cdot \frac{(b-a)^2}{12} + \sigma^2 \right).$$

Then the prediction interval under a 95% criteria for this Gaussian distribution is around

$$\left[\bar{Y} - 1.96\sqrt{\left(1+\frac{1}{N}\right)\left(\beta^2 \cdot \frac{(b-a)^2}{12} + \sigma^2\right)},\ \bar{Y} + 1.96\sqrt{\left(1+\frac{1}{N}\right)\left(\beta^2 \cdot \frac{(b-a)^2}{12} + \sigma^2\right)} \right]$$

Then for a general k, the prediction interval becomes

$$\left[\hat{f} - 1.96 \times \sqrt{\left(1+\frac{k}{N}\right)\left(\beta^2 \cdot \frac{(b-a)^2}{12k^2} + \sigma^2\right)},\ \hat{f} + 1.96 \times \sqrt{\left(1+\frac{k}{N}\right)\left(\beta^2 \cdot \frac{(b-a)^2}{12k^2} + \sigma^2\right)} \right],$$

which is the form

$$\left[\hat{f} - 1.96 \times \text{RMSPE},\ \hat{f} + 1.96 \times \text{RMSPE} \right],$$

a typical Gaussian prediction interval.

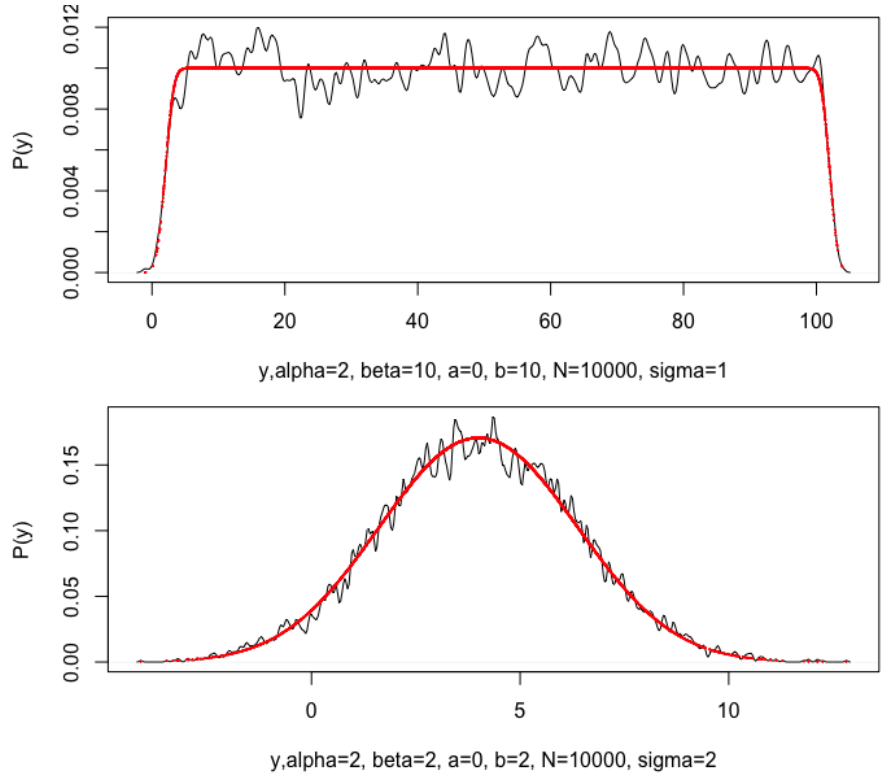


Figure 6.5: Probability density function of Y with different parameter values. Red line is the theoretical probability and black line is the simulated probability.

6.2.3 Prediction simulation using Gaussian prediction interval and quantile interval

In this simulation, we explore the performance of Gaussian prediction intervals and quantile intervals under different parameter combinations. The parameters include σ, $b-a$, β and k. When the other parameters are fixed, a higher σ means a stronger Gaussian distribution effect, in which case, Gaussian prediction interval may work well. When $\beta^2(b-a)^2$ is large, the uniform distribution plays a bigger role. Then Gaussian prediction interval may not work so well. For both Gaussian prediction interval and quantile interval, they are all influenced by the observation size of the terminal node. When the sample size is large, they can have stable performance, but when sample size is small, performance differs.

The Gaussian prediction interval in use is

$$[\hat{f} - c\,\text{RMSPE},\ \hat{f} + c\,\text{RMSPE}],$$

where c is 1.96, and RMSPE is the root mean squared error estimated from the training data in each terminal node.

The quantile interval $[L, U]$ comes from the 0.025 and 0.975 quantiles of each terminal node from the training data.

Step 1: Training data generation. Using given parameters α, β, a, b, σ, N, data are generated according to the model

$$Y = \alpha + \beta X + \epsilon.$$

So we get the true fitted values for Y.

Step 2: RMSPE and quantiles from training data. From this training data, the trained model, RMSPE and quantiles are calculated as the following steps.

Step 2.1: Model training For training data A (the rest data B is the test data), we sort the data x in an ascending order, so y will also be rearranged following x, and then $A_{training}$ is divided into k roughly successive equal folds, making a total of N observations. The number of observations in fold i $(i = 1, 2, ...k)$ is n_i

$$n_i = \lfloor \left(N - \sum_{j=0}^{i-1} n_j \right) / (k-i) \rfloor,$$

defining $n_0 = 0$.

For the i^{th} fold in A, giving x_i and y_i, the predicted value will be

$$\hat{y} = \bar{y}_i,\ x \in [\min(x_i), \max(x_i)],$$

in the trees context. The predicted value of a tree model is the averaged response values of each terminal node. Samples being split into those terminal nodes will have the corresponding averaged value as the predicted value.

Step 2.2: RMSPE and quantile calculation. When the model for each i is trained as $model_{A_i}$, the predicted values for y in A will be $\hat{y}$. Then the RMSPE for the training data is

$$\text{RMSPE} = \sqrt{\sum_{r=1}^{N} (y - \hat{y})^2 / N}.$$

The quantile interval L and U are the 0.025 and 0.975 quantiles of the i_{th} training data y.

Step 3: Test data generation and model testing. Using the same parameters α, β, a, b, σ N as in Step 1, data are generated according to

$$Y = \alpha + \beta X + \epsilon.$$

Then the test data B are put into $model_A$ and the coverage is computed as

$$\frac{1}{N} \sum_{r=1}^{N} I\left(\hat{y} - c\,\text{RMSPE} < y < \hat{y} + c\,\text{RMSPE}\right).$$

Step 4: Repeat Steps 1 to 3 Repeat Steps 1 to 3 s times to get an averaged coverage.

Using parameters $a = 0$, $\alpha = 2$ and $s = 200$, the results are shown in Figures 6.6.

The results show that quantile interval coverages are closer to the 0.95 reference line for fixed σ, b and β. Gaussian prediction interval is only closer to the 0.95 coverage when σ is large, otherwise it is larger than 0.95 at the cost of wider width. When k is chosen as the best k_{min}, the coverages get closer to the 0.95 reference line as σ increases for both quantile and Gaussian prediction intervals. But when the uniform distribution effect gets stronger, the coverages all go far away from 0.95. So when the number of observations for each terminal node is large and the data distribution is not obviously Gaussian, quantile intervals are suggested. When the data follows obvious Gaussian distribution, then Gaussian prediction intervals are recommended.

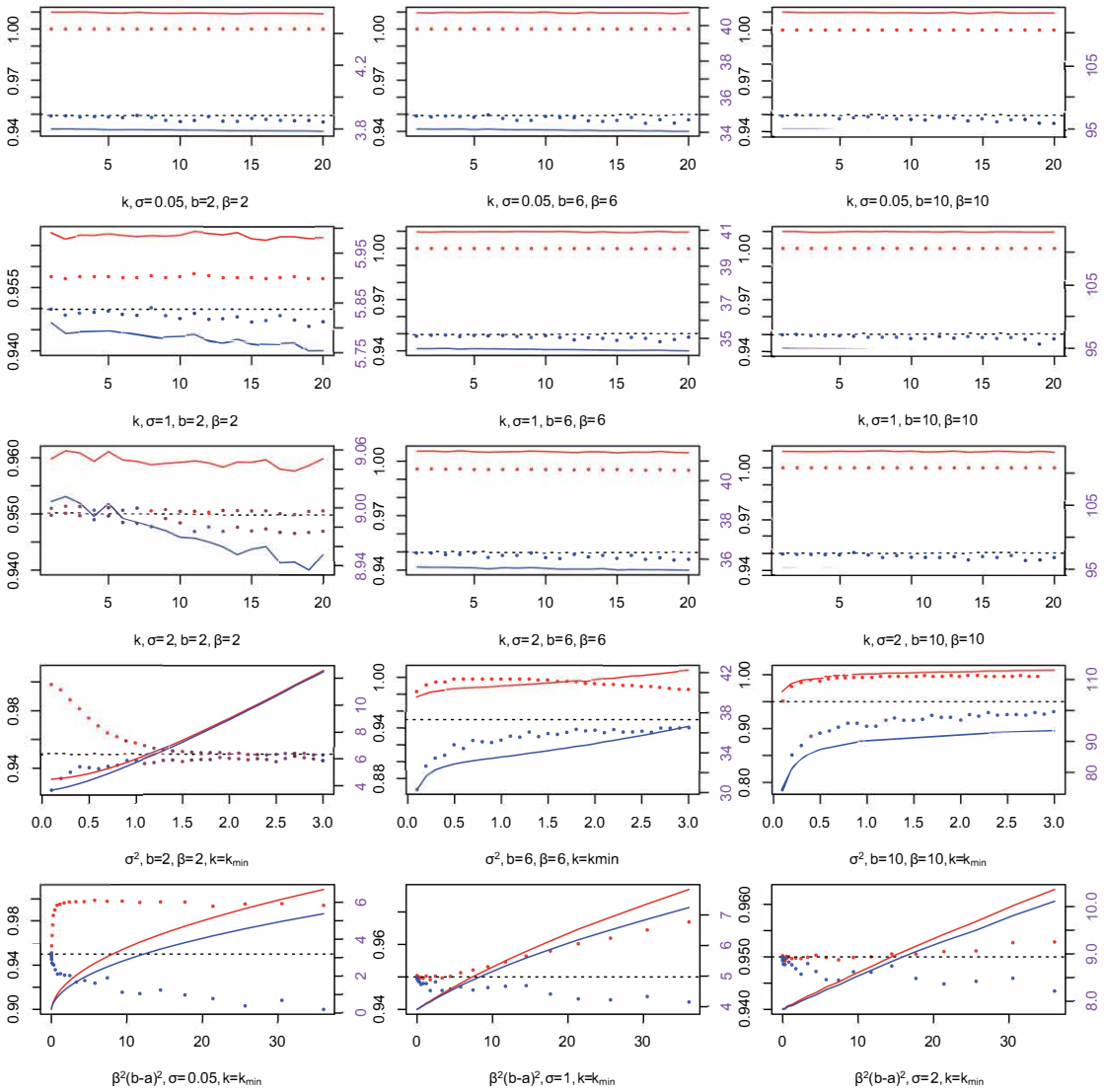


Figure 6.6: Prediction coverage using RMSPE. The black dash line is 0.95. The dash lines are coverage. The solid lines are width. The red lines represent Gaussian prediction interval and the blue lines represent quantile prediction interval. The right-hand y axis (purple) is the axis for width. $N = 10000$.

6.3 Conclusion

In this chapter, the data are constructed using a simple model that includes both Gaussian and uniform distributions. We explored the squared prediction error in the context of trees and decompose that error into bias, variance and irreducible error. The bias decreases when the tree gets bigger. But for squared prediction error and variance, the relationship is not monotonic. We also calculated the best tree depth with a minimum mean squared prediction error. When Gaussian effect dominates, the best tree depth density decreases. But when uniform effect dominates, the best tree depth increases. Under both circumstances, mean squared error, variance and bias all increase.

After that, two options are given for the prediction interval using Gaussian prediction interval or quantile interval. When Gaussian distribution is obviously dominant, Gaussian prediction intervals are suggested. Otherwise, quantile intervals are suggested, which is also why quantile intervals are chosen as the prediction intervals in Chapter 5, although they both perform poorly when the uniform distribution is quite strong. When the number of observations is small in the terminal node, both interval constructions perform poorly in terms of coverage.

Chapter 7

General conclusions and future work

7.1 Overview of work done

This chapter gives a short summary of the main ideas in this thesis, including inspiration, modelling, methods and results from simulated and real data. Then an outline of possible improvements to this study and future work are given.

In Chapter 1, we described the main contributions in this thesis: design new methods for panel data classification and time series regression for both static and dynamic time series. The tools chosen include decision trees (rpart and ctree) and MODWT. The reasons for using those models are briefly explained afterwards. For decision trees, they are easily interpreted and have no requirement of parameter distributional assumptions. MODWT can help improve the performance of trees by deeply exploring the smoothing and detailed information from the scaling and wavelet coefficients.

In Chapter 2, we gave a review of trees and the wavelet transform. The review of trees includes the tree building process, including classification trees, regression trees and conditional inference trees, and then a short description of the tree pruning process was given. For trees, there are many splitting criteria to choose. We explored their splitting bias due to missing values, variables with more values or categories. Between entropy information and the Gini index, we choose the Gini index as the splitting criterion as its bias due to more values or categories is not that large compared to entropy. Using wavelet transformed variables, the number of explanatory variables is increased. So, under some assumptions, we studied the influence of noise variables on CART computational complexity. That increase will generally only result in a linear increase in the computational complexity. For wavelet analysis, we introduced the wavelet transform process, Haar scaling

function and mother wavelet function as well as other Daubechies' wavelet. These ideas form the essential background to the proposed method.

The motivating data set comes from a medical application: monitored variables for patients undergoing liver transplantation surgery. In medical comparison experiments, different treatments make different effects on individuals in different groups. Interpretation of how different treatments influence monitoring variables is important. That is what the panel data analysis did in Chapter 3, by perfoming classification of individuals and explore how the individuals are classified. For decision trees, they can classify each observation, but the target here is to classify each individual, which contains many observations across different times. So three methods are designed based on time-point level predictions, individual level predictions or both together. For wavelet transformed variables, their performance is compared with original variables using both simulated data and the LT surgery data.

The wavelet-transformed variables are found to have better classification performance for panel data than using variables on their original scale. Examples were provided showing the types of data where using a wavelet-based representation is likely to improve classification accuracy. Results show that in most cases wavelet-transformed data have better or similar classification accuracy than the original data, and only select genuinely useful explanatory variables. Use of wavelet-transformed data provides localized mean and difference variables which can be more effective than the original variables, provide a means of separating "signal" from "noise", and bring the opportunity for improved interpretation via the consideration of which resolution scales are the most informative. Panel data with multiple observations on each individual require some form of aggregation to classify at the individual level. Three different aggregation schemes are presented and compared using simulated data and real data gathered during liver transplantation. Among the three methods, aggregating individual level data before classification outperform methods which rely solely on the combining of time-point classifications.

During the surgery, if surgeons can monitor the real time condition of patients, that may save lives in an emergency. Forecasting monitoring variables like heart rate even one minute ahead will help surgeons monitor the condition. The static time series are used in a basic model-building analysis in Chapter 4 and streaming data are used in the real time forecasting in Chapter 5. In Chapter 4, we regarded the heart rate time series during LT surgery as static time series and explored whether or not wavelet transformed variables are better than original variables in time series regression. We also explored this using an air pollution data set from China. Wavelet data generally result in better accuracy measured by metrics of R^2. For this reason, wavelet transformed variables are used in the analysis in Chapter 5.

Specifically, MODWT based CART can detect true lag information and have much better performance when only short lag information is permitted for using. When the forecast length increases, R^2 from MODWT based forecasting decreases much more slowly than that of original forecasting. For real data analysis, MODWT based CART also performs better in both LT data and China air pollution data.

In Chapter 5, wavelet transformed variables are used for streaming data interval forecasting, which was shown to be better than using the original variables in Chapter 4. The interval forecasts are required in this chapter instead of just a point forecast. In order to track streaming data effectively, we required the forecast interval to cover the true data a specified proportion of the time, and to be as narrow as possible. To achieve this, two methods are proposed: one ensemble method and one method based on a single model. For the ensemble method, we used weighted results from the most recent models, and for the single-model method, we retain one model until it becomes necessary to train a new model. For the current model in use in those two methods, we propose a method to update the interval forecast adaptively using root mean square errors calculated from the latest data batch. We choose wavelet-transformed data to capture long time variable information and Conditional Inference Tree for the underlying regression tree model. Results show that both methods perform well, having good coverage without the intervals being excessively wide. The method based on a single model performs the best in computational (CPU) time compared to the ensemble method. When compared to ARIMA and GARCH approaches, we designed simulation using ARIMA or GARCH distributed time series. Our methods still achieve better or similar coverage and width but require considerably less CPU time.

After those application, some theories are explored about trees in Chapter 6, including the behaviour of bias, variance and squared prediction error using trees as the fitted model. For the true model, it includes both Gaussian and uniform distributions. The bias decreases when the tree gets bigger. But for squared prediction error and variance, the relationship is not monotonic. When the Gaussian effect gets stronger, the best tree depth obviously decreases. But when the uniform effect gets stronger, the best tree depth increases. Under both circumstances, mean squared error, variance and bias all increase. For the prediction interval choices, when Gaussian distribution is obviously strong, Gaussian prediction interval is suggested. Otherwise, quantile interval is suggested although they both perform poorly when the uniform distribution is quite strong. When the number of observations in the terminal node is small, both interval construction methods have poor coverage.

7.2 Future work

For the LT data analysis in both classification and regression purposes, each individual are regarded independently, but, actually, their correlation could be considered in the analysis. Individuals with similar properties may have similar distributions for the recorded data. Methods for clustering can include k-means clustering (Wagstaff *et al.*, 2001), distance-based metrics, random forests (Shi *et al.*, 2005). In Chapter 5, we need to wait until there are enough data to train the first model. But if we can find previous individuals who share similar property with the current individual, we may start the forecasting using the model from those previous individuals.

In the classification in Chapter 3, each variable was assumed to be independent although trees do not have constraints of independence or dependence for explanatory variables. In the tree building process, the split comes from one variable. If we can also consider the possible combinations of many variables for one split, that may work better. Possible methods include PCA, factor analysis (Rummel, 1988), linear discriminant analysis (Izenman, 2013), or tree-based methods.

Instead of using tree models, other models can be considered for comparison. For example, after wavelet transform, logistic panel data regression (Chamberlain, 1984) can be applied to the panel data using LASSO selected variables, and summarised cross-section data can be classified using support vector machine (Cortes & Vapnik, 1995).

When group size is rebalanced, the data of minor groups were triplicated in this thesis. Other sophisticated methods, however, like Synthetic Minority Oversampling Technique (SMOTE) (Chawla *et al.*, 2002) can also be considered.

In Chapter 4, the results are shown only as numbers. It is also possible to show the predicted results using heat maps which would give a more direct way of comparing to the original data.

In Chapter 5, we updated the model when coverage is not good or retrain the model when updating does not work. There were circumstances when old data patterns are suitable for new data but those patterns have already been "forgotten" by the current model. For the ensemble method, instead of keeping the most recent models, it might be a good idea to choose which models to keep based on data patterns. It is also a good idea to save as many old models as possible, and delete those with low accuracy after a long time.

The proposed methods contain many tuning parameters. Although their values are chosen by trying many possible values. It will be better if further research can be done to evaluate the method's reliability, by exploring the influence of different parameter values on the performance of the method.

Instead of detecting concept drift by coverage and mean information, other tools can also be developed for interval prediction purpose, like ADWIN, STEP

shown in Section 5.2. The dynamic tree methods like the Hoeffding tree family, can be developed for interval prediction using quantile or other information.

If interpretation is not the major requirement of model output, then approaches like random forest, neural networks are good models as well.

Appendix A

Wavelet transform methods comparison

Table A.1: Results comparison between Haar and mb4.

			method 1		method 2		method 3	
			original	wavelet	original	wavelet	original	wavelet
sigma 0 noisep=0.1	Haar	mean	30	59.7	30	60	36.71	59.52
		sd	0	0.64	0	0	3.9	0.83
	mb4	mean	30	57.33	30	56.99	36.54	59.84
		sd	0	1.9	0	1.92	3.82	0.39
sigma 0.5 noisep=0.1	Haar	mean	30.1	57.91	30.08	57.69	35.49	59.83
		sd	0.66	1.53	1.31	1.54	3.69	0.43
	mb4	mean	30.08	56.8	30.08	56.7	35.6	59.49
		sd	0.63	1.88	1.31	1.93	3.8	0.76
sigma 1 noisep=0.1	Haar	mean	29.92	56.77	29.97	56.34	32.82	59.7
		sd	0.63	1.85	0.22	1.93	4.26	0.63
	mb4	mean	29.92	56.31	29.97	55.71	33.11	57.59
		sd	0.63	1.84	0.3	2.17	3.97	1.67
sigma 5 noisep=0.1	Haar	mean	30	51.77	30.01	48.28	30	57.97
		sd	0	2.78	0.1	3.17	3.78	1.67
	mb4	mean	30	55.35	30	52.3	30.1	52.55
		sd	0	2.19	0	2.83	3.97	2.69
sigma 10 noisep=0.1	Haar	mean	30.07	49.65	30	41.48	29.9	54.93
		sd	0.65	3.05	0.71	3.54	3.78	2.92
	mb4	mean	30.07	54.03	30.01	46.26	29.97	49.41
		sd	0.65	2.52	0.76	3.99	3.87	3.79
sigma 20 noisep=0.1	Haar	mean	29.85	45.79	29.9	34.85	29.72	47.65
		sd	0.9	5.16	0.64	4.43	3.7	7.08
	mb4	mean	29.85	52.33	29.9	39.91	29.76	40.18
		sd	0.9	3.81	0.64	4.41	3.85	7.9
sigma 0 noisep=0.3	Haar	mean	30	58.65	30	57.05	34.82	59.82
		sd	0	1.71	0	1.89	4.07	0.48
	mb4	mean	30	56.26	30	56.24	34.65	57.62
		sd	0	1.83	0	1.78	4.37	1.7
sigma 0 noisep=0.5	Haar	mean	29.97	55.58	30.03	55.71	32.2	59.63
		sd	1.03	1.8	0.98	1.86	3.8	0.72
	mb4	mean	29.97	55.11	30	55.05	32.09	56.02
		sd	1.03	2.26	0.83	2.3	3.83	1.94
sigma 0 noisep=0.8	Haar	mean	29.98	43.65	29.97	44.06	29.84	51.81
		sd	0.53	3.62	0.54	3.76	3.78	2.46
	mb4	mean	30.01	46.03	29.98	45.04	29.97	46.4
		sd	0.61	3.43	0.53	3.65	3.79	3.29
sigma 0 noisep=0.1	Haar	mean	59.81	59.59	60	59.87	59.99	59.72
delta=0.2		sd	0.46	0.79	0	0.34	0.1	0.59
	mb4	mean	59.81	59.51	60	59.57	59.99	59.95
		sd	0.46	0.79	0	0.62	0.1	0.22

Table A.2: Results comparison between Haar and d4.

			method 1		method 2		method 3	
			original	wavelet	original	wavelet	original	wavelet
sigma 0 noisep=0.1	Haar	mean	30	59.64	30	59.96	35.72	59.53
		sd	0	0.7	0	0.32	3.99	0.8
	d4	mean	30	57.7	30	57.11	35.71	59.88
		sd	0	1.78	0	1.79	3.81	0.38
sigma 0.5 noisep=0.1	Haar	mean	29.95	57.47	29.9	57.8	34.71	59.83
		sd	0.39	1.88	0.63	1.52	4.01	0.43
	d4	mean	29.98	56.76	29.91	56.62	34.67	59.4
		sd	0.49	1.97	0.55	1.98	4.27	0.78
sigma 1 noisep=0.1	Haar	mean	29.95	56.78	29.93	56.47	34.08	59.61
		sd	0.5	1.68	0.7	1.85	3.63	0.69
	d4	mean	29.95	56.41	29.93	56.09	33.95	57.35
		sd	0.5	1.85	0.7	2.14	3.55	1.56
sigma 5 noisep=0.1	Haar	mean	30.03	52	30	48.23	30.19	57.9
		sd	0.3	2.67	0.14	3.16	3.54	1.38
	d4	mean	30.03	54.84	30.01	51.71	30.16	52.57
		sd	0.3	2.53	0.1	2.65	3.71	3
sigma 10 noisep=0.1	Haar	mean	29.85	50.05	29.84	41.97	29.56	55.09
		sd	1.11	3.31	1.02	4.25	3.71	2.52
	d4	mean	29.82	54.91	29.79	46.77	29.51	50.32
		sd	1.15	2.65	1.09	3.81	3.46	2.93
sigma 20 noisep=0.1	Haar	mean	29.96	44.9	29.97	34.47	30.14	48.46
		sd	0.62	6.49	0.33	4.43	3.35	6.21
	d4	mean	29.96	51.87	29.92	38.97	30.15	39.88
		sd	0.62	5.03	0.51	4.42	3.22	7.41
sigma 0 noisep=0.3	Haar	mean	30	58.52	30	57.25	34.26	59.74
		sd	0	1.51	0	1.78	4.87	0.65
	d4	mean	30	56.22	30	56.09	34.14	57.54
		sd	0	2.13	0	2.07	4.82	1.83
sigma 0 noisep=0.5	Haar	mean	29.94	55.71	29.94	55.6	32.04	59.57
		sd	0.6	1.75	0.6	1.98	3.69	0.67
	d4	mean	29.94	55.37	29.94	55.04	31.85	55.9
		sd	0.6	1.93	0.6	2.14	3.51	2.32
sigma 0 noisep=0.8	Haar	mean	30.06	43.65	30.05	43.44	30.38	51.61
		sd	0.92	3.48	0.9	3.92	3.86	2.91
	d4	mean	30.06	46.31	30	45.07	30.39	45.93
		sd	0.92	3.2	0.14	3.44	3.71	3.14
sigma 0 noisep=0.1	Haar	mean	59.68	59.55	60	59.81	59.99	59.71
delta=0.2		sd	0.53	0.77	0	0.44	0.1	0.64
	d4	mean	59.68	59.55	60	59.6	59.99	59.96
		sd	0.53	0.74	0	0.64	0.1	0.2

Table A.3: Results comparison between Haar and la8.

			method 1		method 2		method 3	
			original	wavelet	original	wavelet	original	wavelet
sigma 0 noisep=0.1	Haar	mean	30	59.73	30	60	35.98	59.63
		sd	0	0.62	0	0	4.11	0.82
	la8	mean	30	56.33	30	56.84	36.24	59.79
		sd	0	2.8	0	4.49	4.12	0.46
sigma 0.5 noisep=0.1	Haar	mean	30.08	57.58	30.13	57.53	35.47	59.65
		sd	1.28	1.46	1	1.64	4.12	0.61
	la8	mean	30.08	51.92	30.13	49.44	35.97	59.25
		sd	1.28	2.96	1	3.36	3.82	0.89
sigma 1 noisep=0.1	Haar	mean	30.08	56.94	30.08	56.3	33.85	59.65
		sd	0.8	1.89	0.8	1.87	4.41	0.66
	la8	mean	30.08	49.94	30.08	48.24	34.05	55.72
		sd	0.8	2.91	0.8	3.38	4.69	2.35
sigma 5 noisep=0.1	Haar	mean	30	52.26	30	48.9	30.06	57.95
		sd	0	2.86	0	3.45	3.8	1.64
	la8	mean	30	43.89	30	41.49	30.73	40.75
		sd	0	3.72	0	4.02	3.94	4.51
sigma 10 noisep=0.1	Haar	mean	30.07	49.87	30.09	41.22	30.05	55.1
		sd	0.7	2.74	0.9	3.87	3.69	2.58
	la8	mean	30.07	41.56	30.09	37.45	30.44	35.68
		sd	0.7	3.89	0.9	4.3	3.79	5.44
sigma 20 noisep=0.1	Haar	mean	30.03	45.81	29.99	35.38	29.84	48.3
		sd	0.22	5.56	0.22	4.59	3.61	6.31
	la8	mean	30.03	36.46	30.02	33.12	29.88	30.98
		sd	0.22	5.08	0.14	4.79	4.03	4.04
sigma 0 noisep=0.3	Haar	mean	30	58.23	30	57.24	33.9	59.77
		sd	0	1.71	0	1.67	3.61	0.62
	la8	mean	30	50.61	30	49.04	33.82	56.37
		sd	0	2.95	0	2.87	3.75	1.8
sigma 0 noisep=0.5	Haar	mean	29.99	55.84	30	55.61	31.56	59.55
		sd	0.1	1.69	0	1.86	4.18	0.69
	la8	mean	29.99	47.72	30	46.78	31.57	50.49
		sd	0.1	3.28	0	2.93	4.37	3.01
sigma 0 noisep=0.8	Haar	mean	30	43.59	30.05	44.13	29.52	51.38
		sd	0.38	4.04	0.5	4.07	3.86	2.73
	la8	mean	30	37.87	30	37.32	29.84	38.24
		sd	0.38	3.26	0	4.23	3.64	3.44
sigma 0 noisep=0.1	Haar	mean	59.76	59.7	60	59.82	59.99	59.7
delta=0.2		sd	0.51	0.59	0	0.41	0.1	0.59
	la8	mean	59.76	59.68	60	59.87	59.99	59.82
		sd	0.51	0.69	0	0.37	0.1	0.41

References

Abdar, M., Kalhori, S.R.N., Sutikno, T., Subroto, I.M.I. & Arji, G. (2015). Comparing performance of data mining algorithms in prediction heart diseases. *International Journal of Electrical and Computer Engineering (IJECE)*, **5**, 1569–1576. 2

Alickovic, E. & Subasi, A. (2016). Medical decision support system for diagnosis of heart arrhythmia using DWT and random forests classifier. *Journal of Medical Systems*, **40**, 1–12. 42

Appice, A. & Ceci, M. (2006). Mining tolerance regions with model trees. In *International Symposium on Methodologies for Intelligent Systems*, 560–569, Springer. 84

Asavaskulkeit, K. & Jitapunkul, S. (2009). The color face hallucination with the linear regression model and MPCA in HSV space. In *2009 16th International Conference on Systems, Signals and Image Processing*, 1–4, IEEE. 66

Aykroyd, R.G., Barber, S. & Miller, L.R. (2016). Classification of multiple time signals using localized frequency characteristics applied to industrial process monitoring. *Computational Statistics & Data Analysis*, **94**, 351–362. 36, 37

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R. & Morales-Bueno, R. (2006). Early drift detection method. In *Fourth International Workshop on Knowledge Discovery from Data Streams*. 87

Bifet, A. & Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, 443–448, SIAM. 87

Bifet, A. & Gavaldà, R. (2009). Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis*, 249–260, Springer. 88

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R. & Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Aata Mining*, 139–148, ACM. 86

Bifet, A., Holmes, G., Kirkby, R. & Pfahringer, B. (2010). MOA: Massive online analysis. *Journal of Machine Learning Research*, **11**, 1601–1604. 86

Bonferroni, C. (1936). Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, **8**, 3–62. 31, 32

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984a). *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA. 8

Breiman, L., Friedman, J., Stone, C.J. & Olshen, R.A. (1984b). *Classification and Regression Trees*. Wadsworth, Belmont. 1, 2, 42

Cauwenberghs, G. & Poggio, T. (2001). Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, 409–415. 2

Chamberlain, G. (1984). Panel data. *Handbook of Econometrics*, **2**, 1247–1318. 135

Chawla, N.V., Bowyer, K.W., Hall, L.O. & Kegelmeyer, W.P. (2002). Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, **16**, 321–357. 135

Chirapongsathorn, S., Valentin, N., Alahdab, F., Krittanawong, C., Erwin, P.J., Murad, M.H. & Kamath, P.S. (2016). Nonselective β-blockers and survival in patients with cirrhosis and ascites: A systematic review and meta-analysis. *Clinical Gastroenterology and Hepatology*, **14**, 1096–1104. 58

Chitaliya, N.G. & Trivedi, A. (2010). Feature extraction using wavelet-pca and neural network for application of object classification & face recognition. In *Computer Engineering and Applications (ICCEA), 2010 Second International Conference*, vol. 1, 510–514, IEEE. 67

Cohen, J. (1988). The analysis of variance. *Statistical power analysis for the behavioral sciences*, **2**, 273–403. 3

Coifman, R.R. & Donoho, D.L. (1995). Translation-invariant de-noising. In A. Antoniadis & G. Oppenheim, eds., *Wavelets and Statistics*, vol. 103 of *Lecture Notes in Statistics*, 125–150, Springer-Verlag, New York. 42

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, **20**, 273–297. 135

Costa, J.A. & Hero, A.O. (2004). Geodesic entropic graphs for dimension and entropy estimation in manifold learning. *IEEE Transactions on Signal Processing*, **52**, 2210–2221. 66

Cox, D.R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, **20**, 215–242. 2

Datar, M., Immorlica, N., Indyk, P. & Mirrokni, V.S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, 253–262, ACM. 66

Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, **41**, 909–996. 38

Daubechies, I. (1992). *Ten Lectures on Wavelets*. SIAM, Philadelphia. 58, 68

Domingos, P. & Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 71–80, ACM. 86, 88

Donoho, D.L. & Johnstone, J.M. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, **81**, 425–455. 1, 4, 42

Duarte, J., Gama, J. & Bifet, A. (2016). Adaptive model rules from high-speed data streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, **10**, 30. 86

Fryzlewicz, P. & Nason, G.P. (2004). A Haar-Fisz algorithm for Poisson intensity estimation. *Journal of Computational and Graphical Statistics*, **13**, 621–638. 68

Funahashi, K.I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, **2**, 183–192. 2

Gama, J., Medas, P., Castillo, G. & Rodrigues, P. (2004). Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*, 286–295, Springer. 87

Ghalanos, A. (2014). *rugarch: Univariate GARCH models*. R package version 1.3-5. 86, 101

Gini, C. (1912). *Variabilità e mutabilità*. 3

Gokgoz, E. & Subasi, A. (2015). Comparison of decision tree algorithms for EMG signal classification using DWT. *Biomedical Signal Processing and Control*, **18**, 138–144. 42

Goodman, K.E., Lessler, J., Cosgrove, S.E., Harris, A.D., Lautenbach, E., Han, J.H., Milstone, A.M., Massey, C.J. & Tamma, P.D. (2016). A clinical decision tree to predict whether a bacteremic patient is infected with an extended-spectrum β-lactamase–producing organism. *Clinical Infectious Diseases*, **63**, 896–903. 2

Graps, A. (1995). An introduction to wavelets. *IEEE Computational Science and Engineering*, **2**, 50–61. 37

Gupta, P., Christopher, S.A., Wang, J., Gehrig, R., Lee, Y. & Kumar, N. (2006). Satellite remote sensing of particulate matter and air quality assessment over global cities. *Atmospheric Environment*, **40**, 5880–5892. 80

Gurjar, B., Butler, T., Lawrence, M. & Lelieveld, J. (2008). Evaluation of emissions and air quality in megacities. *Atmospheric Environment*, **42**, 1593–1606. 80

Hastie, T., Tibshirani, R. & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer. 115, 116

Hothorn, T. & Zeileis, A. (2015). Partykit: a modular toolkit for recursive partitioning in R. *Journal of Machine Learning Research*, **16**, 3905–3909. 86

Hothorn, T., Hornik, K., Van De Wiel, M.A. & Zeileis, A. (2006a). A lego system for conditional inference. *The American Statistician*, **60**, 257–263. 1, 2, 16

Hothorn, T., Hornik, K. & Zeileis, A. (2006b). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, **15**, 651–674. 1, 2, 3, 8, 16, 17

Hothorn, T., Hornik, K. & Zeileis, A. (2015). ctree: Conditional inference trees. *The Comprehensive R Archive Network*. 8

Hyndman, R.J. (2017). *forecast: Forecasting functions for time series and linear models*. R package version 8.2. 86, 96

Hyndman, R.J. & Khandakar, Y. (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, **26**, 1–22. 86, 96

Ikonomovska, E., Gama, J. & Džeroski, S. (2011). Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery*, **23**, 128–168. 86

Ikonomovska, E., Gama, J. & Džeroski, S. (2015). Online tree-based ensembles and option trees for regression on evolving data streams. *Neurocomputing*, **150**, 458–470. 86

Iversen, G.R. & Norpoth, H. (1987). *Analysis of Variance*. 1, Sage. 3

Izenman, A.J. (2013). Linear discriminant analysis. In *Modern Multivariate Statistical Techniques*, 237–280, Springer. 135

Jin, R. & Agrawal, G. (2003). Efficient decision tree construction on streaming data. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, 571–576, ACM, New York, NY, USA. 86

Jones, D.S. (1979). *Elementary Information Theory*. Clarendon Press, Oxford. 11, 12, 13

Kass, G.V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, **29**, 119–127. 8, 22, 29

Li, S. & Wen, J. (2014). A model-based fault detection and diagnostic methodology based on PCA method and wavelet transform. *Energy and Buildings*, **68**, 63–71. 67

Loh, W.Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **1**, 14–23. 2, 3, 8, 10, 31

Maharaj, E.A. & Alonso, A.M. (2007). Discrimination of locally stationary time series using wavelets. *Computational Statistics & Data Analysis*, **52**, 879–895. 42

Maharaj, E.A. & Alonso, A.M. (2014). Discriminant analysis of multivariate time series: Application to diagnosis based on ECG signal. *Computational Statistics & Data Analysis*, **70**, 67–87. 42

Mazloom, M. & Ayat, S. (2008). Combinational method for face recognition: wavelet, PCA and ANN. In *Digital Image Computing: Techniques and Applications (DICTA), 2008*, 90–95, IEEE. 67

Meinshausen, N. & Bühlmann, P. (2006). High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, **34**, 1436–1462. 4

Milan, Z., Taylor, C., Armstrong, D., Davies, P., Roberts, S., Rupnik, B. & Suddle, A. (2016). Does preoperative beta-blocker use influence intraoperative hemodynamic profile and post-operative course of liver transplantation? *Transplantation Proceedings*, **48**, 111–115. i, 6, 58, 59, 66, 101

Morris, J.M. & Peravali, R. (1999). Minimum-bandwidth discrete-time wavelets. *Signal Processing*, **76**, 181–193. 58

Nason, G.P. & Silverman, B.W. (1995). The stationary wavelet transform and some statistical applications. In A. Antoniadis & G. Oppenheim, eds., *Wavelets and Statistics*, vol. 103 of *Lecture Notes in Statistics*, 281–300, Springer-Verlag, New York. 42

Nie, F., Xu, D., Tsang, I.W.H. & Zhang, C. (2010). Flexible manifold embedding: A framework for semi-supervised and unsupervised dimension reduction. *IEEE Transactions on Image Processing*, **19**, 1921–1932. 66

Nishida, K. & Yamauchi, K. (2007). Detecting concept drift using statistical testing. In *International Conference on Discovery Science*, 264–269, Springer. 87

Percival, D.B. & Walden, A.T. (2000). *Wavelet Methods for Time Series Analysis*. Cambridge University Press, Cambridge. 1, 36, 42, 86

Pfahringer, B., Holmes, G. & Kirkby, R. (2007). New options for Hoeffding trees. In *Australasian Joint Conference on Artificial Intelligence*, 90–99, Springer. 86

Quan, H., Srinivasan, D. & Khosravi, A. (2014). Short-term load and wind power forecasting using neural network-based prediction intervals. *IEEE Transactions on Neural Networks and Learning Systems*, **25**, 303–315. 86

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, **1**, 81–106. 31

Quinlan, J.R. (2014). *C4.5: programs for machine learning*. Elsevier. 8, 31

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. 48, 86

Roberts, S. & Nowak, G. (2014). Stabilizing the lasso against cross-validation variability. *Computational Statistics & Data Analysis*, **70**, 198–211. 68

Rokach, L. & Maimon, O. (2005). Decision trees. In *Data Mining and Knowledge Discovery Handbook*, 165–192, Springer. 10

Rokach, L. & Maimon, O. (2008). *Data Mining with Decision Trees: Theory and Applications*. World Scientific, Singapore. 10, 31

Ross, G.J., Adams, N.M., Tasoulis, D.K. & Hand, D.J. (2012). Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, **33**, 191–198. 87

Rowley, H.A., Baluja, S. & Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**, 23–38. 68

Rummel, R.J. (1988). *Applied factor analysis*. Northwestern University Press. 135

Schölkopf, B., Smola, A. & Müller, K.R. (1997). Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, 583–588, Springer. 3

Sethi, T.S. & Kantardzic, M. (2017). On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, **82**, 77–99. 87

Shi, T., Seligson, D., Belldegrun, A.S., Palotie, A. & Horvath, S. (2005). Tumor classification by tissue microarray profiling: random forest clustering applied to renal cell carcinoma. *Modern Pathology*, **18**, 547. 135

Shrestha, D.L. & Solomatine, D.P. (2006). Machine learning approaches for estimation of prediction interval for the model output. *Neural Networks*, **19**, 225–235. 86

Sobhani, P. & Beigy, H. (2011). New drift detection method for data streams. *Adaptive and Intelligent Systems*, **6943**, 88–97. 87

Specht, D.F. (1990). Probabilistic neural networks. *Neural Networks*, **3**, 109–118. 2

Strobl, C., Boulesteix, A.L. & Augustin, T. (2007). Unbiased split selection for classification trees based on the Gini index. *Computational Statistics & Data Analysis*, **52**, 483–501. 22, 23

Sun, P., Cárdenas, D.A. & Harrill, R. (2016). Chinese customers evaluation of travel website quality: A decision-tree analysis. *Journal of Hospitality Marketing & Management*, **25**, 476–497. 2

Suykens, J.A. & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, **9**, 293–300. 2

Therneau, T., Atkinson, B. & Ripley, B. (2014). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-8. 8, 48

Therneau, T.M., Atkinson, E.J. *et al.* (1997). An introduction to recursive partitioning using the rpart routines. 22

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, **58**, 267–288. 4

Tipping, M.E. & Bishop, C.M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **61**, 611–622. 3

Upadhyaya, S. & Mohanty, S. (2016). Localization and classification of power quality disturbances using maximal overlap discrete wavelet transform and data mining based classifiers. *IFAC-PapersOnLine*, **49**, 437–442. 42, 43

Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S. *et al.* (2001). Constrained k-means clustering with background knowledge. In *ICML*, vol. 1, 577–584. 135

Walker, S.H. & Duncan, D.B. (1967). Estimation of the probability of an event as a function of several independent variables. *Biometrika*, **54**, 167–179. 2

Ward, M.D. & Gleditsch, K.S. (2018). *Spatial regression models*, vol. 155. Sage Publications. 80

Whitcher, B. (2013). *waveslim: Basic wavelet routines for one-, two- and three-dimensional signal processing*. R package version 1.7.3. 48, 86

Wuertz, D., Setz, T., Chalabi, Y., Boudt, C., Chausse, P., Miklovac, M., Setz, M.T. & Runit, S. (2013). Package "fgarch". Tech. rep., Technical report, working paper/manual, 09.11. 2009. URL http://cran. r-project. org/web/packages/fGarch/fGarch. pdf. 86, 101

Yoshida, S.i., Hatano, K., Takimoto, E. & Takeda, M. (2011). Adaptive online prediction using weighted windows. *IEICE Transactions on Information and Systems*, **94**, 1917–1923. 87

You, W., Yang, Z. & Ji, G. (2014). Feature selection for high-dimensional multi-category data using PLS-based local recursive feature elimination. *Expert Systems with Applications*, **41**, 1463–1475. 68

Zhang, Y., Wang, S., Phillips, P. & Ji, G. (2014). Binary PSO with mutation operator for feature selection using decision tree applied to spam detection. *Knowledge-Based Systems*, **64**, 22–31. 2

Zhang, Y., Wang, S., Phillips, P., Dong, Z., Ji, G. & Yang, J. (2015). Detection of Alzheimer's disease and mild cognitive impairment based on structural volumetric MR images using 3D-DWT and WTA-KSVM trained by PSOTVAC. *Biomedical Signal Processing and Control*, **21**, 58–73. 43

Zhao, X., Barber, S., Taylor, C.C. & Milan, Z. (2018). Classification tree methods for panel data using wavelet-transformed time series. *Computational Statistics & Data Analysis*, **127**, 204–216. 41, 86, 104