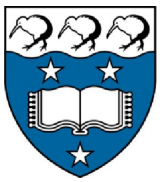
Predictive Models under Complex Sampling



Huanchao Qin

Department of Statistics
The University of Auckland

Supervisor: Prof. Thomas Lumley

A dissertation submitted in partial fulfillment of the requirements for the degree of Bachelor of Science (Honours) in Statistics, The University of Auckland, 2022

Abstract

K-fold cross-validation is frequently used in predictive models for best model selections. Complex survey data are obtained mainly through unequal probability sampling, and they often have clusters. As a result, the standard cross-validation is not suitable for the complex sampled data as those who have a higher chance of being drawn and those who have a lower chance of being drawn are treated similarly in standard cross-validation. Furthermore, a random split of the training and test sets can disrupt the clusters. We suggest alternative cross-validation approaches that might be better suited for complex sampling designs, which is the use of replicate weights in k-fold cross-validation and jackknife cross-validation. Simulations are used for experimentation and comparison of the normal k-fold cross-validation, k-fold cross-validation with replicate weights and jackknife cross-validation with replicate weights. We then validate the three approaches with public survey data from the National Health and Nutrition Examination Survey (NHANES) and Student performance survey in California schools (API). We found that using weights can improve the performance of regression trees significantly, but there is no significant difference between k-fold cross-validation and jackknife cross-validation with replicate weights.

Acknowledgements

I would like to express my deep gratitude to my research supervisor Professor Thomas Lumley, for his patience and guidance throughout the year, even though I was initially slow. You lead me where to go and when it's time to take a step forward. I had a great time working with you this year.

I also wish to thank my parents for their continued financial support and encouragement throughout my study.

Contents

Chapter 1

Introduction

Predictive modeling is a statistical technique to forecast likely future outcomes using historical and existing data. It works by analyzing current and historical data and projecting what it learns to a model generated to predict possible outcomes. Predictive models such as decision trees depend heavily on cross-validation, which is difficult for complex survey data like weights or clusters. Survey data are typically obtained through unequal probability sampling or cluster sampling to save money. For normal cross-validation, it is hard to use weights, and cross-validation can disrupt clusters when training models. In that case, the accuracy of the best-trained model will be reduced.

This project is about using predictive models in a complex survey setting, in which we modify the ordinary cross-validation for modeling data from multi-stage surveys. The experiments are done through simulation and validation with publicly available data. We compare the distribution of prediction errors under different survey designs. Our purpose is to find an optimal alternative to cross-validation under particular survey designs for survey data.

Chapter 2 explains the background knowledge and is divided into three sections. The first section introduces complex sampling, which includes sampling weights and three

core sampling methods. The second section provides a detailed introduction to cross-validation and explains why it is difficult to use on survey data. The prediction model regression tree used in this project is introduced in the last section.

Chapter 3 discusses the procedure of data simulation and how to use survey sampling weights in predictive models. For the simulated complex sampling data, we build the regression tree model and modify the original cross-validation. Then compare the performance of different methods by estimating the prediction error. The simulation results are presented through graphs and tables.

Chapter 4 experiment with real survey data and demonstrates the results from the previous chapter in a real-life setting. The first data set is from the National Health and Nutrition Examination Survey, a large multi-stage survey study conducted by Nation Center for Health Statistics in the United States. The second one is from the Academic Performance Index (API) survey data, which is about the Academic Performance of all California schools. We use these survey data to confirm our findings in the last chapter.

Chapter 2

Background

This chapter covers all the background knowledge of the dissertation. Section 2.1 deals with complex sampling, first explaining the sampling weights, then describing the three main sampling methods: simple random sampling, cluster sampling, and stratified sampling and their related concepts. After understanding these sampling methods, we talk more about the weights. The contents of this section are the cornerstones of this project. Then, section 2.2 discusses cross-validation, which starts from the validation set approach to the leave-one-out cross-validation (LOOCV) approach and k-fold cross-validation. This is followed by cross-validation for classification problems. Finally, and most importantly, it explains why cross-validation is limited for complex survey data. The last section provides information on the algorithm for the predictive model used in the project.

2.1 Complex Sampling

Taking a census is costly and time-consuming, especially for a large population with millions or even billions of people. When analyzing the characteristics of a large target group, survey sampling is useful for reducing consumption of cost and time [1]. A sample is selected from a representative population subset and used to make inferences

about the population. No other research techniques exist that can create a sample with higher accuracy for obtaining focused data to draw conclusions and make critical judgments. Usually, analyzing complex survey samples is design-based. We focus on a population and treat it as fixed, and randomly choose the sample from this fixed population. The researcher controls the sample design, which is the random selection procedure of individuals.

2.1.1 Sampling weights

New Zealand has about 5 million people, around 80% of whom are adults[1]. Assuming a random sample of 30,000 adults and 30,000 minors were drawn separately to study heart disease, so there were 60,000 Kiwis in the sample. The probability that each adult New Zealander will be sampled is p_i = 30,000/(5,000,000$\times$0.8) = 3/400 = 0.0075, and the probability of each minor being sampled is p_j = 30,000/(5,000,000$\times$0.2) = 3/100 = 0.03. The above sampling procedure is called unequal probability sampling. It's not like equal probability sampling, in which each individual in the population has the same chance of being selected. If the sample says 2,000 people have been diagnosed with heart disease, we couldn't say the whole country would be expected to have 2,000$\times$5,000,000/60,000 $\approx$ 166,667 heart diseases. This result is not representative. Because minors are four times more likely to be sampled than adults while being less likely to have heart disease. The sampling results will be biased toward the prevalence of minors, so the actual prevalence of heart disease in New Zealanders will be higher than this sampling result.

When the sample is chosen with unequal selection probabilities, ignoring the sample selection scheme in the inference process can lead to misleading results, even after conditioning on all the available design information [2]. Hence, when fitting models to complex survey data, the use of sampling weights is taken into consideration. In a restricted sense, probability weighting for the sample observations produces reli-

[1] https://www.stats.govt.nz/topics/population

able estimates of the model parameters and guards against model misspecification. In the previous example, p_i is the sampling probability for unit i, $\frac{1}{p_i}$ is called the sampling weight and each unit i represents $\frac{1}{p_i}$ individuals in the population. So how do we approach the population mean or total with the sampling weights? Horvitz and Thompson (1952) proposed the estimator [3] of a population total:

$$\widehat{total}_y = \sum_{i=1}^{n} w_i y_i \tag{2.1}$$

- n is the sample size.
- w_i is the ith weight for each individual.

The variance of the Horvitz-Thompson estimator for the total is:

$$\widehat{Var}(\widehat{total}_y) = \sum_{i \in U} \sum_{j \in U} I_i I_j (1 - \frac{Cov(I_i, I_j)}{p_{ij}}) \frac{y_i}{p_i} \frac{y_j}{p_j} \tag{2.2}$$

- U is a population indexed by $1, \ldots, i, j, \ldots, N$.
- p_i, p_j is the sampling probabilities for individual i, j which is equal to $\frac{1}{w_i}$ and $\frac{1}{w_j}$, respectively.
- p_{ij} is the joint inclusion probability in the sample for units i and j.
- $I_i \in \{0, 1\}$ is unit i's inclusion indicator, $I_j \in \{0, 1\}$ is unit j's inclusion indicator.

We can apply the formula for the variance estimator to any designed survey, and it's unbiased no matter how complex the problem is. Moreover, the formula is not only related to sampling weights but also the pairwise sampling probabilities p_{ij} [1]. Complex designs usually involve one or more methods, such as stratification, clusters, weights, etc. In this chapter, we will illustrate three main sampling methods: simple random sampling, stratified sampling, and cluster sampling.

2.1.2 Randomization - Simple Random Sampling

In simple random sampling (SRS), a subset of n people (a sample) is picked at random from a larger group of N people (a population), all with the same probability. It is a method of choosing a sample at random. Each subset of n people in SRS has the same chance of getting selected for the sample as any other subset of n people, and each individual in the SRS sample has the same sampling weight N/n [1]. We care about unbiasedness which is known as validity. It means, on average, the sampling procedure obtains the correct result. The law of large numbers implies that simple random sampling is unbiased and refers to the process, not the outcome of a sample, and what happens on average across all process repetitions. Now, let us focus on the properties of the sampling distribution:

The first is unbiasedness. We know that simple random sampling is unbiased. Hence, the expected value or average value is

$$E(\overline{y}) = \frac{1}{{}_NC_n} \sum_{s=1}^{{}_NC_n} \overline{y_s} = \overline{Y} \tag{2.3}$$

- $\overline{Y}$ is the population mean. On average, we get the right answer.
- ${}_NC_n$ is the number of sets of sample size n distinct elements from population N.
- $\overline{y_s}$ is the mean from one sample. From Horvitz-Thompson estimator, $\overline{y_s} = \sum_{i=1}^{n} w_i y_i$.

The second is sampling variance, the variability from one sample to another. The variance of the estimator $Var(\overline{y})$:

$$Var(\overline{y}) = SE(\overline{y})^2 = \frac{1}{{}_NC_n} \sum_{s=1}^{{}_NC_n} (\overline{y_s} - E(\overline{y}))^2 = (1 - \frac{n}{N})\frac{S^2}{n} \tag{2.4}$$

- S^2 is the variability of the elements of the population, which is an approximation from the sample.

- $\frac{n}{N} = f$ is the sampling fraction.
- $1 - \frac{n}{N} = 1 - f$ is called the finite population correction and it's a multiplier of the S^2 which reduces the impact of S^2.
- $\frac{1}{n}$ means that, as sample size increases, variance decreases.

If we aim to estimate the smoking rate of the youth in New Zealand, the first thing is to create a list of all young people in New Zealand. Creating the list is not difficult as the census is available. However, the travel costs and interview costs can be very expensive. We assume that a simple random sample of young people is drawn not only from the south island but also from the north island. Also, it is scattered across the cities. Hence we consider an alternative method, cluster sampling, to save money. We will talk about it in the following subsection.

2.1.3 Saving Money - Cluster Sampling

In cluster sampling, we divide a target population into several smaller groups according to their similarity, known as clusters. Typically groups are divided based on geographic location, such as cities. We only randomly choose some of the clusters. Continuing with the study of youth smoking rates, assuming Auckland and Christchurch are randomly selected. Instead of interviewing all young New Zealanders, only some youth in two cities are interviewed. All New Zealand cities are called primary sampling units (PSUs). Therefore, cluster selections lower interview costs, as we only need to interview part of the objectives for selected clusters. Besides, young people from the clusters are often already listed in the census, which reduces travel costs. So far, we have been talking about one-stage cluster sampling. Like the study of youth smoking rates, in all 44 cities of New Zealand, only two cities are randomly selected. Nevertheless, there is still a problem. Interviewing all young people in Auckland and Christchurch is a big project. In addition to this, we also need to consider various expenses. Hence, it comes up with the two-stage cluster sampling. Suppose we randomly sample 10,000 young people in total. Ideally, each cluster has the same size. Just randomly select

5,000 people from each. However, in real life, most of the clusters are unequal in size. How do we decide the number of young people to choose from each of the two cities?

One approach involves selecting a specified percentage of units from each of the chosen clusters in stage two. The number of units to be sampled varies depending on cost concerns. The chosen sample will provide an unbiased estimator. However, because the cost estimates frequently refer to a specific sample size, the sample size is no longer predetermined, which causes issues with the research plan's optimal management of expenses and results in a more complex formula for the estimator's standard error. Another approach is utilizing probability proportionate to size sampling. A large cluster has a higher likelihood of selection than a small cluster under this sampling since the probability of selecting a cluster is related to its size. The benefit in this situation is that each sampled cluster should conduct the same number of interviews in order to ensure that each unit sampled has the same probability of selection when clusters are picked with a probability proportionate to the size [4].

2.1.4 Using auxiliary data - Stratified Sampling

Before drawing the sample, we already know some things about the individuals from the population. Such as, young people can be separated into males and females by biological sex. Also, they can form three groups: students, employed and unemployed people. Those are the auxiliary variables. In stratified sampling, individuals in a population can be divided into different categories. The elements within the strata should be as similar as possible, considering the available auxiliary variables. The homogeneity of elements within a stratum leads to greater variance between strata, which may reduce sampling variance. Hence from an estimation point of view, stratified sampling can increase the precision of our estimate.

2.1.5 More on weighting

The sampling rate is the probability of a population being sampled, which is the inverse of the sampling weight. For the equal probability selection method, each individual in the sample has the same sampling rate. Hence, each individual has a constant weight. For example, we want to sample 1,200 graduates in the age range. Table 2.1 assumes a population size of 400,000 graduates per year and demonstrates the proportionate allocation, which means applying the same sampling rate (1200/400,000 = 1/333.33) to all groups. Hence, each individual in the stratum has equal probability. Therefore, it can represent the population.

Stratum	N	n	Sampling rate	Weight A	Weight B
28+	80,000	240	1/333.33	333.33	1
22-28	320,000	960	1/333.33	333.33	1
Total	400,000	1,200	1/333.33	333.33	1

Table 2.1: Proportionate allocation

When we want to compare the two strata, equal allocation, also called disproportionate allocation, would be used instead of proportionate allocation. That means we sample the same amount of graduates from each stratum (1200/2 = 600). Table 2.2 demonstrates this disproportionate allocation.

Stratum	N	n	Sampling rate	Weight A	Weight B
28+	80,000	600	1/133.33	133.33	1
22-28	320,000	600	1/533.33	533.33	4
Total	400,000	1,200	1/333.33		

Table 2.2: Disproportionate allocation

Now, let us estimate the mean score obtained from averaging the two groups of samples. It is known that the average score for graduates over the age of 28 is 82, and the average score for those between the ages of 22 and 28 is 72. Hence, for proportionate allocation, the estimated mean score is (shown in Table 2.3):

$$\frac{82 \times 240 + 72 \times 960}{1200} = 74$$

Stratum	Mean score	n	Mean score	Weight A	Weight B
28+	82	240	82	333.33	1
22-28	72	960	72	333.33	1
Total	74	1,200	74	333.33	1

Table 2.3: Proportionate allocation

For disproportionate allocation, the estimated mean score is:

$$\frac{82 \times 600 + 72 \times 600}{1200} = 77$$

Stratum	Mean score	n	Mean score	Weight A	Weight B
28+	82	600	72	133.33	1
22-28	72	600	92	533.33	4
Total	74	1,200	77		

Table 2.4: Disproportionate allocation

The result is also displayed in Table 2.4. It turns out that it is larger than the population mean score. Weights will fix this problem.

Weight A:

$$\frac{82 \times 600 \times 133.33 + 72 \times 600 \times 533.33}{600 \times 133.33 + 600 \times 533.33} = 74$$

Weight B:

$$\frac{82 \times 600 \times 1 + 72 \times 600 \times 4}{600 \times 1 + 600 \times 4} = 74$$

Therefore, when fitting models to complex survey data, it is important to use the sampling weights.

2.2 Cross-Validation

Cross-validation is one of the most common re-sampling methods, widely used for model assessment and selection. Model assessment is a process of evaluating a model's performance [5]. Under a given predictive model, we often use cross-validation to estimate the test error, which considers the model's performance. The test error is also used to choose the proper level of flexibility for a model and can be used to perform a model selection for several models. For the evaluation of the model's performance, we split the data into a training set to fit the model and a test set to see how well the method works. To evaluate the performance of the model, we predict previously unseen data in the test set and estimate the accuracy of the predictions. suppose that we fit our model on the training data set $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_n, y_n)$, we obtain the model $\widehat{f}$. Then we can compute $\widehat{f}(x_1), \widehat{f}(x_2), \widehat{f}(x_3), \ldots, \widehat{f}(x_n)$. So the training mean squared error (MSE) is

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}(x_i))^2 \tag{2.5}$$

where $\widehat{f}(x_i)$ is the prediction for the ith observation. Many statistical techniques estimate coefficients to reduce the training MSE as much as possible. The training MSE for these techniques can be relatively small, but the test MSE is usually substantially larger. It results from a lack of generalisability and the model being too biased toward the distribution of variables in the training data. This is called over-fitting, and models can be prone to over-fit to noise in training data and perform poorly with unseen data. For the purpose of prediction, we usually only care about the prediction $\widehat{f}(x_0)$ for the previously unseen test data (x_0, y_0) which is not used to train the model. We select the model with the lowest test MSE rather than the lowest training MSE. For a large number of test data set, the test MSE is Ave$(y_0 - \widehat{f}(x_0))^2$ which represents the average squared prediction error for these test observations (x_0, y_0).

2.2.1 The validation set approach

The validation set approach is the basis for cross-validation, randomly dividing the existing observations into two parts, a training set, and a validation set. We apply the method to the training set and use the trained model to predict the observations in the validation set. Then calculate the MSE from the validation set error, which is the test error. Figure 2.1 demonstrates the validation set approach.

Figure 2.1: A conceptual representation of the validation set approach. An observation set of n is randomly divided into a training set (shown in blue and including, among others, observations 7, 22, and 13) and a validation set (shown in beige and containing observation 91, among others). The training set is used to set up the predictive model, while the validation set is used to gauge its effectiveness [5].

Although it works easily, it has two disadvantages. One is that different training and validation set splits can cause a wide range of test errors. Furthermore, the other is that we only fit the model using some observations. We know that the model needs to fit the entire data set. The validation set error will likely overestimate the test error rate. Therefore, a better validation set approach is needed to solve these drawbacks. We will talk about cross-validation.

2.2.2 Leave-one-out cross-validation

Like the validation set approach, Leave-One-Out Cross-Validation (LOOCV) randomly divides the observations into two parts, but for the validation set part, it only contains one observation (x_1, y_1), and all the remainders are the training set part $\{(x_2, y_2), \ldots, (x_n, y_n)\}$ including $n-1$ observations to fit the model. We perform a prediction $\widehat{y_1}$ on x_1. The unbiased estimate for the test error is $MSE_1 = (y_1 - \widehat{y_1})^2$. It depends on each

single observation (x_1, y_1), so its variance is relatively large which makes it a poor estimate. To overcome this drawback, we pick (x_2, y_2) as the validation set and the other $n-1$ observations $\{(x_1, y_1), (x_3, y_3), \ldots, (x_n, y_n)\}$ to train the model. Calculate $MSE_2 = (y_2 - \widehat{y_2})^2$ and repeat this method n times $MSE_1, \ldots, MSE_n$. The average of these n squared errors is the LOOCV estimate for the test MSE:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} MSE_i \tag{2.6}$$

Figure 2.2 illustrates the LOOCV approach.

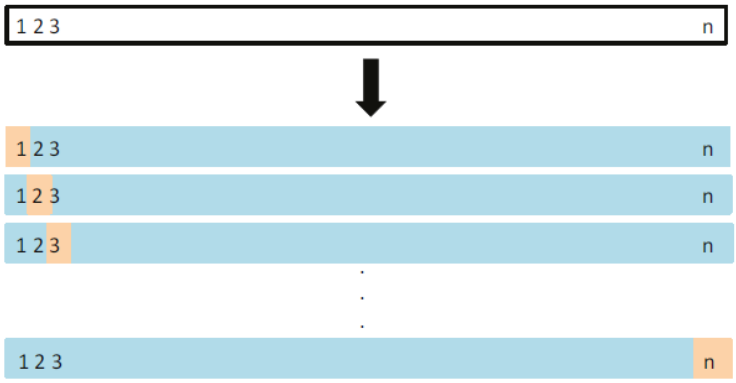


Figure 2.2: LOOCV's schematic representation. Iteratively dividing a set of n data points into a training set (shown in blue) that contains all but one observation and a validation set that contains only that observation (shown in beige). After that, the test error is estimated by averaging the n obtained MSEs. All but observation 1 are present in the first training set, all but observation 2 are present in the second training set, and so on [5].

Compared to the validation set approach, LOOCV has two main advantages. Firstly, there is much less bias. In the training and validation split, the training set is generally about half of the data set in the validation set method. For the LOOCV, we use the training set that includes n - 1 observations to fit the model, which is nearly the whole data set. As a result, the LOOCV method does not overestimate the test error like

the validation set approach. Secondly, due to randomly splitting the training/validation set, the validation set approach will produce different MSEs when repeating this procedure. However, LOOCV will always yield the same results after applying many times as there is no randomness in the training/validation set splitting.

We can apply LOOCV to any predictive model. Nevertheless, if n is large, LOOCV is time-consuming since the model needs to be fit n times. Even worse, when each model is slow to work, it is more expensive to use.

2.2.3 K-fold cross-validation

A better alternative is k-fold cross-validation. It randomly divides the observations into k folds of approximately equal size. The first fold is the validation set, and the remaining $k-1$ folds are used to train the model. Then compute the mean squared error, MSE_1, on the observations in the held-out fold. Repeating this procedure another $k-1$ times, a different fold of observations is used as the validation set each time. This process yields k test error estimates $MSE_1, MSE_2, \ldots, MSE_k$. Averaging these values results in the k-fold cross-validation estimate:

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i \tag{2.7}$$

When k is equal to n, k-fold cross-validation is LOOCV. For computational purpose, we often use k = 5 or 10 to perform k-fold cross-validation. Figure 2.3 illustrates the k-fold cross-validation approach.

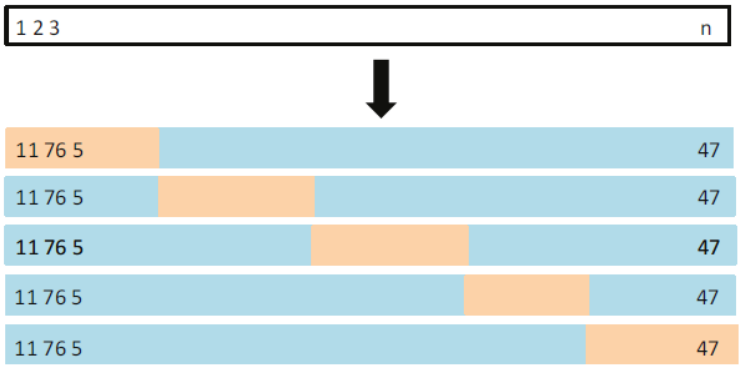


Figure 2.3: A five-fold CV is schematically displayed. Five distinct, non-overlapping groups are randomly created from n observations. Each fifth (shown in beige) serves as a validation set, with the remaining four-fifths serving as a training set (shown in blue). The five resulting MSE estimates are averaged to estimate the test error [5].

2.2.4 Cross-validation on classification problems

We have discussed cross-validation for the quantitative outcome and used MSE to represent the test error. For the classification setting, when the outcome is qualitative, the number of wrongly classified observations is used to quantify test error instead of MSE. For example, for an error rate of binary classification problems, metrics such as cross-entropy can be used:

$$CV = y \log(\widehat{y}) + (1 - y) \log(1 - \widehat{y}) \tag{2.8}$$

2.2.5 Cross-validation under complex sampling

Cross-validation is often used to select a hyper-parameter for predictive models. For example, k-fold cross-validation is applied to select the best cost-complexity parameter α in regression trees. α is used to find the subtree, which is less complex than the full-grown tree. So far, we have introduced the usual CV, which obtains the training set by random splitting. However, for complex surveys, cross-validation has limited use. We

cannot randomly divide the data set, as clusters would be split between the training and test sets and the test data is not independent of the training data. It will result in more bias in the test error. Hence, the model assessment would be unreliable, which leads to poor model selection [6]. Therefore, for non-iid data, like survey data, there are limitations to the usual cross-validation approach. We will illustrate several alternative methods which match the sampling data from complex survey sampling designs in chapter 3.

2.3 Regression Tree

Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone's monograph "CART: Classification and Regression Trees," first published in 1984 [7], is credited with setting a significant milestone in the development of artificial intelligence, machine learning, non-parametric statistics, and data mining. This work is vital for its authoritative presentation of large sample theory for trees, its complex explanation of tree-structured data analysis, its technical improvements, and its thorough examination of decision trees. Decision trees are among the most widely used machine learning methods as they are straightforward and understandable [8]. Its objective is to build a model that uses several input variables to forecast the value of a target variable. When the data needs to be divided into classes that are a part of the target variable, we typically use classification trees. On the other hand, when the response variable is continuous, regression trees are utilized. In this project, we only focus on the regression tree.

2.3.1 Terminology

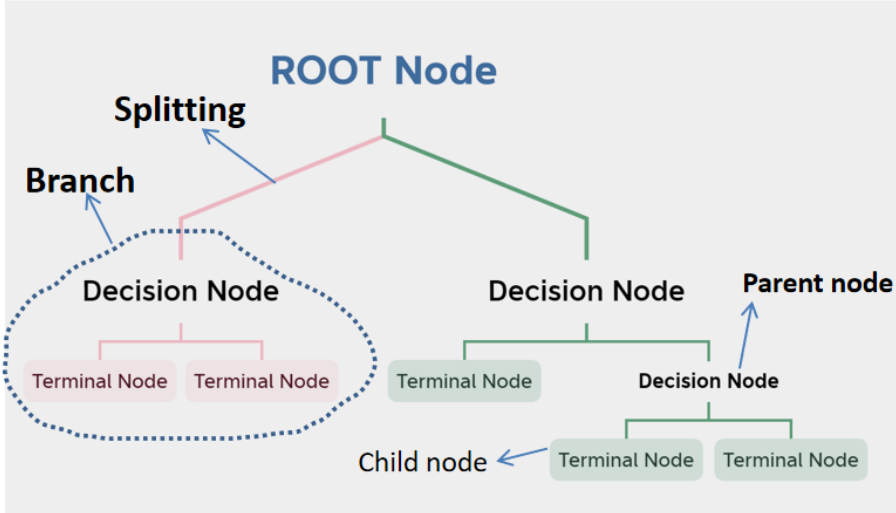


Figure 2.4: A graphical display of a regression tree with the basic concepts

Let us start by understanding the terminology associated with the regression tree. Figure 2.4 visualizes the basic concepts of regression trees. Next, we will learn about them one by one.

Root node

A regression tree can also be considered a set of nodes or a directional graph with a single node as its starting point. The root node is the initial node, and it represents the whole sample space.

Splitting

Explains the procedure of dividing a node into two or more sub-nodes.

Decision node

It is the outcome of a split. A sub-node is a decision node when it divides into more sub-nodes.

Terminal node

A sub-node is referred to as a terminal node or leaf node if it cannot be further subdivided. The terminal node represents the response value used for the prediction.

Pruning

Pruning is the process of removing sub-nodes from a decision node. It might be described as splitting in reverse.

Branch

Branch or subtree refers to a subset of the whole tree.

Parent and child node

A child node is any node that is nested beneath another node. Parent nodes are any nodes that come before those child nodes.

Figure 2.5 shows an example of a regression tree where the response variable is the monthly salary of a football player in $10,000. For the entire sample, the average monthly wage is $19,000. In this sample, if the independent variable b for a player is less than 0.97, then his/her monthly salary is predicted to be $11,000. Otherwise, it is $58,000. Moreover, 83% of the athletes in the entire sample satisfy the condition that the predictor variable b is less than 97%, and the remaining 17% have b greater than or equal to 0.97.

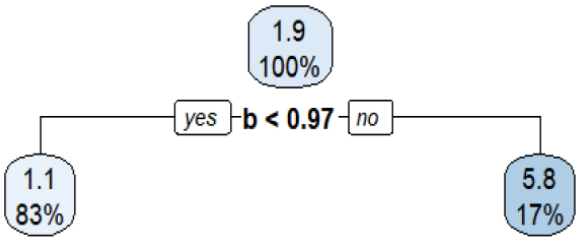


Figure 2.5: Prediction of football player's salary

2.3.2 Algorithm

Now we explore how to build a regression tree, with four steps in total [5]. The first step is to construct a huge tree using recursive binary splitting [7] with the training data, halting only when there are fewer than a certain minimal number of observations at each terminal node. At the beginning, we need to create J unique and non-overlapping regions $R_1, R_2, \ldots, R_j$ from the predictor space, or the set of possible values for $X_1, X_2, \ldots, X_p$. How do we create the regions $R_1, R_2, \ldots, R_j$? Finding $R_1, R_2, \ldots, R_j$ that minimise the sum of squared errors RSS is the objective, which is stated by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \widehat{y}_{R_j})^2$$

where $\widehat{y}_{R_j}$ is the average response for the training observations given in the jth region. However, it is computationally impossible to consider every possible J region partition of the feature space. As a result, we apply a top-down, greedy strategy known as recursive binary splitting. The method is top-down because it starts at the root node, then splits the predictor space into two new branches at successive intervals down the tree. It is greedy because the optimal split is selected at each stage of the tree-building process rather than looking forwards and selecting a split that will lead to a better tree at a later stage. Furthermore, the top-down induction of decision trees (TDIDT) [9] method is the most popular method for learning decision trees from data and illustrates how a greedy algorithm works [10]. For recursive binary splitting, we first choose the predictor X_j and the cutpoint s to divide the predictor space into the regions R_1, R_2 resulting in the most significant RSS decrease possible.

$$R_1(j, s) = X/X_j < s, R_2(j, s) = X/X_j \geq s$$

Therefore, we look for the j and s values that minimize the equation

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \widehat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \widehat{y}_{R_2})^2$$

where $\widehat{y}_{R_1}$ is the average response for the training observations given in $R_1(j, s)$ and $\widehat{y}_{R_2}$ is the average response for the training observations given in $R_2(j, s)$. Then we split one of the two regions that were previously discovered. There are now three regions. Once more, in order to minimize the RSS, we try to further partition one of these three regions. Up until a stopping requirement is achieved, the procedure goes on. If it's the regression tree with weights, we seek the value of j and s that minimize the equation:

$$\sum_{i:x_i \in R_1(j,s)} w_i(y_i - \widehat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} w_i(y_i - \widehat{y}_{R_2})^2$$

w_i is the respective weight for each observation. By default, all input records and classes are thought to be of equal relative significance [2]. We can alter this by giving each member of one or all of these things a unique weight. Doing so could be beneficial if the distribution of the data points among the training data categories is unrealistic. Weights allow us to mitigate the model's bias towards the majority group and make up for underrepresented groups in the data. The percentage of accurate predictions for a group should increase when the weight for a target value is increased. Figure 2.6 displays the first step in a general way.

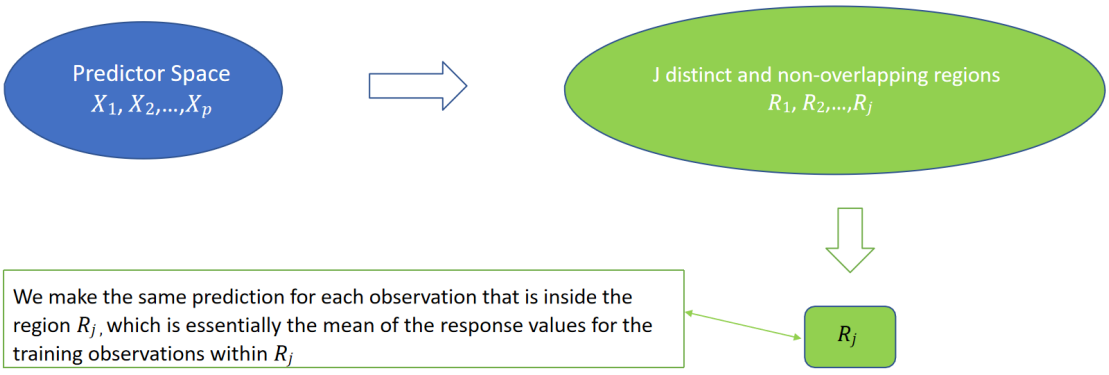


Figure 2.6: Overview of the first step

A decision tree fitted with the training data will normally try to assign every data points to their correct groups, which results in a fully-grown tree that tries to correctly assign as many data points as possible in the training data. This might cause problems with over-fitting and perform poorly on the test set. Therefore, the second step is tree pruning which applies cost-complexity pruning [7] to the fully-grown tree to generate a list of the best subtrees, as a function of α (cost-complexity parameter). We prune the fully-grown tree T_0 to create a subtree that can better generalise on test data. We take into account a sequence of trees indexed by a non-negative tuning parameter α instead of every potential subtree. There is a subtree $T \subset T_0$ that matches to each value of α:

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \widehat{y}_{R_m})^2 + \alpha |T|$$

We will choose an α with the smallest resulting value. $|T|$ shows how many terminal nodes there are in the tree T. The subtree's complexity and the accuracy of the training data are traded off according to the tuning parameter α. When $\alpha = 0$, the subtree T will be T_0 as at that point, it only counts the training error. Nevertheless, there is a cost associated with the tree having numerous terminal nodes as α rises. It will typically be minimized for a smaller subtree.

The next step is choosing α via k-fold cross-validation. Firstly randomly divide the training observations into k roughly equal groups/folds, in which for each iteration, one fold is used as the test set while the other folds as the training set. Redo steps 1 and 2 on the training set and calculate the MSE using the test set. Then repeat this process $k-1$ more times and use a new fold as the test set each time. We need to average those estimates to obtain a k-fold CV estimate as a function of α. Finally, repeat for various random splits and compare the CV estimates to determine which α has the minimum CV error.

The last step is that the subtree from step 2 that corresponds to the selected value of α should be returned. Figure 2.7 demonstrates the whole process of building a regression tree.

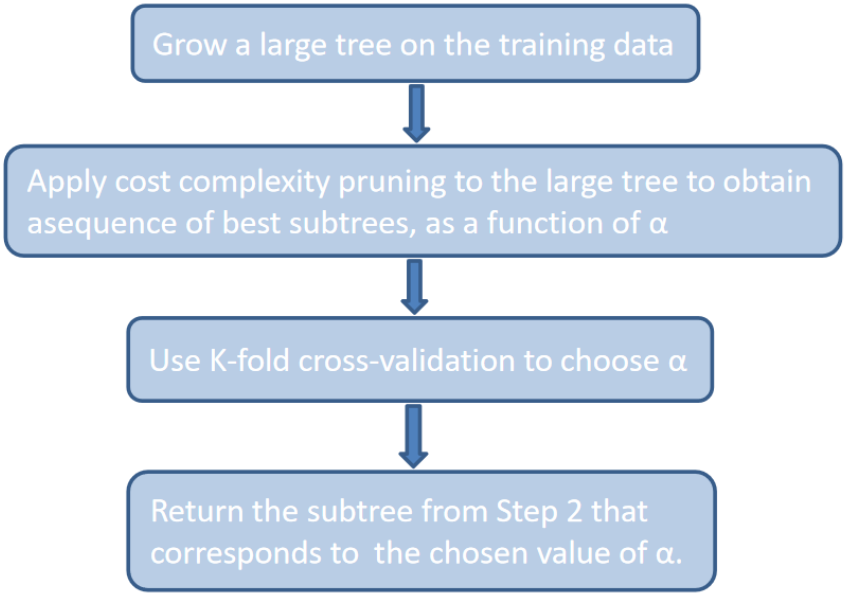


Figure 2.7: The display of the algorithm of a regression tree

Chapter 3

Methodology and Simulation

As illustrated in chapter 2, the decision tree model strongly relies on cross-validation. Nevertheless, cross-validation in the traditional sense is less applicable to survey data. In this chapter, we will present two alternative approaches to cross-validation. One uses the lowest MSE to choose a tuning parameter while building a regression tree instead of k-fold cross-validation. The other uses the replicate weights to estimate the mean squared prediction error. We will cover them in detail in the following sections.

3.1 Simulation

In the real world, data is valuable, and the exact survey data we need is even harder to obtain. It takes much expense to sample the required data. Meanwhile, we can never be sure about the actual distribution and relationships within data and the true values of these population parameters. However, with a simulation study, we know what the real relationship is as we know how the data is generated and what the true parameters are. Furthermore, simulation is a crucial tool for statistical research, especially for assessing novel approaches and contrasting alternative methods [11]. Moreover, it is utilized in place of theory testing. In this project, several populations were simulated, and we will go into more detail in the following subsections.

3.1.1 Population I

The first half of the project is about weighting. In the model I simulated, all the predictors are independent of each other, among which numeric predictors have the same mean of zero and variance of one. In total, there are 20 predictors and 100,000 observations. In addition to numerical variables, binary variables and categorical variables are covered. We make the model as complex as possible. The model includes squared and cubed relationships, also interactions between variables. And the expression is:

$$y_i = 1+x_{1_i}+x_{2_i}+x_{3_i}+x_{4_i}+x_{5_i}+x_{6_i}+x_{7_i}+x_{8_i}+x_{9_i}+x_{10_i}-x_{11_i}-0.25x_{12_i}+0.5x_{13_i}+1.25x_{14_i}+2x_{15_i}-2c_{1_i}+1.5c_{2_i}-0.6c_{3A_i}-0.05c_{3B_i}+0.5c_{3C_i}+0.05c_{4a_i}+0.07c_{4b_i}+0.09c_{4c_i}+0.11c_{4d_i}+0.001c_{51_i}+0.0015c_{52_i}+0.002c_{53_i}+0.0025c_{54_i}+0.003c_{55_i}+3x_{11_i}^2+2.515x_{12_i}^3+2.03c_{1_i}x_{13_i}+1.545c_{2_i}c_{3A_i}+1.06c_{2_i}c_{3B_i}+0.575c_{2_i}c_{3C_i}+0.09x_{14_i}c_{1_i}c_{2_i}+e_i,$$

$e_i \sim$N(0,1).

Figure 3.1 details the independent variables of the model:

Numeric variables	$x_1, x_2, \ldots, x_{15}$
Binary variables	c_1, $P_{c_{1=0}} = P_{c_{1=1}} = 0.5$ c_2, $P_{c_{1=0}} = 0.1$ and $P_{c_{1=1}} = 0.9$
Categorical variables	c_3 with levels ***A***(0.6), ***B***(0.3) and ***C***(0.1) c_4 with levels ***a***(0.2), ***b***(0.3), ***c***(0.4) and ***d***(0.1) c_5 with levels ***1***(0.03), ***2***(0.1), 3(0.6), ***4***(0.1) and ***5***(0.17)

Figure 3.1: Description of the predictors

Table 3.1 lists the summarized statistics of the response variable y of the model, and Figure 3.2 visualizes the distribution of y. There are a few extreme points. We do not remove them here as we know they are not outliers.

Summary Statistics	**y**
Minimum	-188
Lower quartile	0.26
Median	5
Upper quartile	10
Maximum	194

Table 3.1: Summary statistics of y for population I

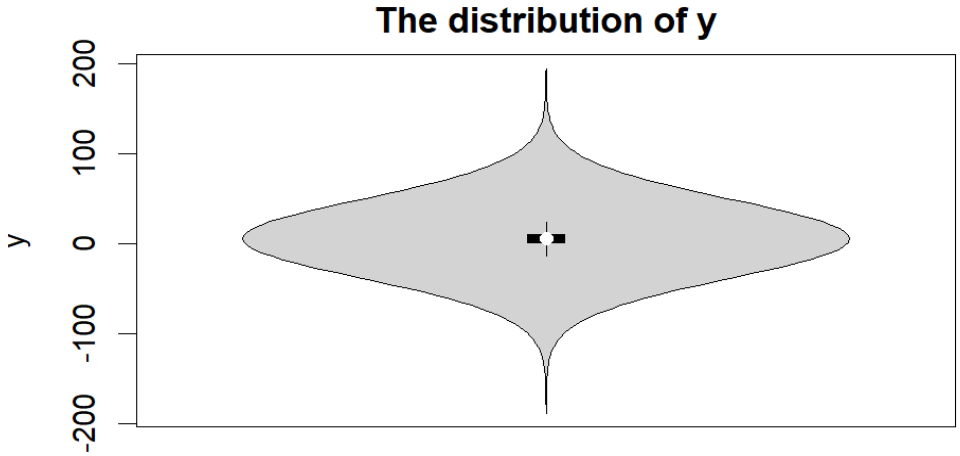


Figure 3.2: The violin plot of y for population I

3.1.2 Population II

The second half of the project is about clusters. We are using the same model expression as in the first half of the project. For example, the relationship between the response and predictor variables is the same. There are still 100,000 observations in total. We simulated two scenarios, each of which corresponds to a population. These scenarios include one-stage cluster sampling and two-stage cluster sampling. Next, we describe each of the populations one by one.

3.1.2.1 One-stage cluster sampling

We add the random effects clustering in population I for one-stage cluster sampling. Overall, there are 200 clusters, and each cluster has 500 data. Observations within each cluster are similar, but those from different clusters differ. Figure 3.3 shows the shape of the distribution of the response variable y within each cluster.

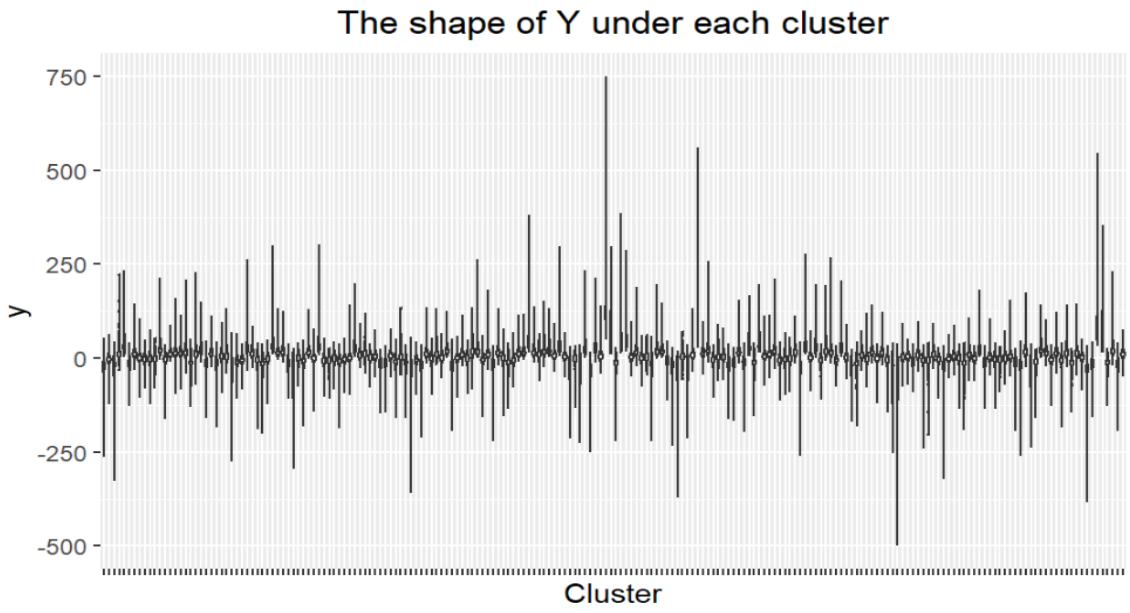


Figure 3.3: The violin plot of y under each cluster for one-stage cluster sampling

Now, let us not worry about the cluster and focus on the dependent variable y. Table 3.2 gives the details of the summary statistics of y.

Summary Statistics	**y**
Minimum	-498
Lower quartile	-8
Median	5
Upper quartile	18
Maximum	750

Table 3.2: Summary statistics of y for one-stage cluster sampling

Figure 3.4 displays the overall distribution of y. It is a wide range of data fluctuation.

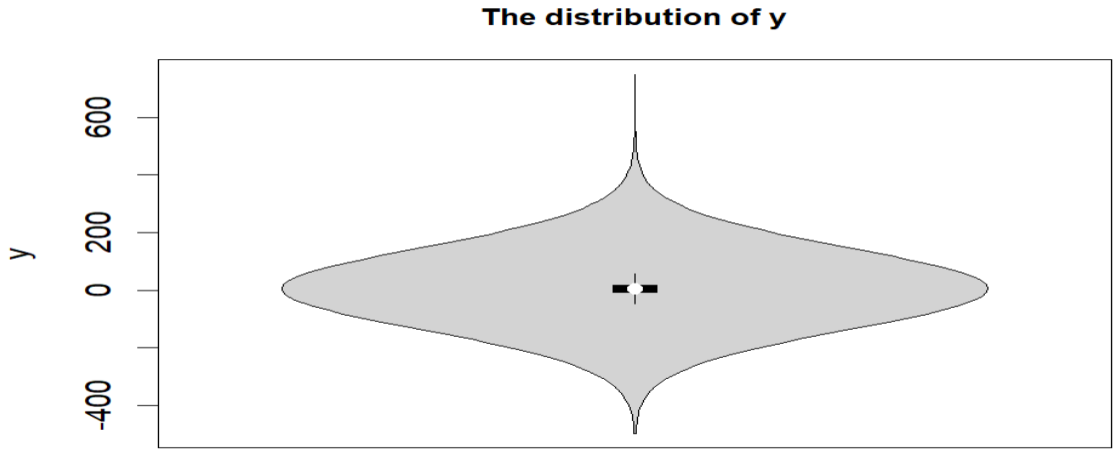


Figure 3.4: The violin plot of y for one-stage cluster sampling

3.1.2.2 Two-stage cluster sampling

We already have the stage 1 clusters. Now we add random effects clustering in stage 2. Under each cluster in stage one, there are 10 stage two clusters. As introduced earlier, there are 500 data for each first-stage cluster. Therefore, there are 50 data for each second stage cluster. Table 3.3 details the summary of y.

Summary Statistics	**y**
Minimum	-500
Lower quartile	0.39
Median	15
Upper quartile	29
Maximum	755

Table 3.3: Summary statistics of y for two-stage cluster sampling

The overall distribution of y is a bit close to that for one-stage cluster sampling, as shown in Figure 3.5.

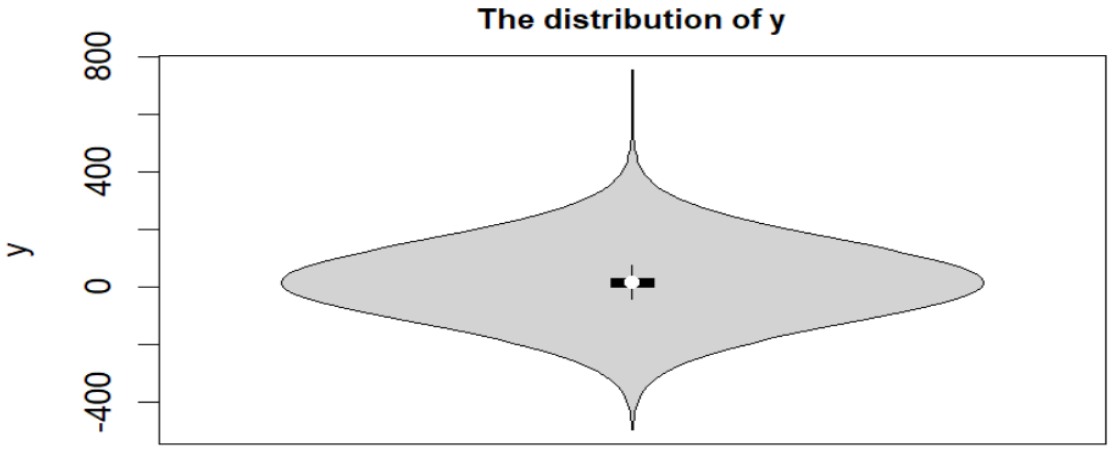


Figure 3.5: The violin plot of y for two-stage cluster sampling

Those are the simulated data that will be used in our experimentation. Next section, we will talk about the adjustment of weighting.

3.2 Weighting

Ideally, we should have an equal probability sample so that each individual has the same chance of being drawn. The sample is used directly to train the regression tree model so that the results are non-biased. However, for complex survey data, due to budget constraints and other reasons, different sampling methods are adopted, and often, the sample is not obtained by equal probability sampling. As illustrated in chapter 2, even after conditioning on all the available design information, neglecting the sample selection scheme in the inference process can result in misleading conclusions when the sample is picked with unequal selection probability. Therefore, sampling weights are considered when fitting models to complex survey data.

The package used to build a regression tree is rpart [1], a robust machine-learning library for R. It implements recursive partitioning for fitting trees and is user-friendly. Our method uses the lowest MSE to select the best tuning parameter α to get the corresponding subtree rather than k-fold cross-validation. Figure 3.6 demonstrates the method. After growing a large tree T_0 using rpart on the training data, firstly find the cost-complexity parameters in the cptable [2], which gives a brief overview of the model's overall fit. Then prune T_0 with all the cost-complexity parameters and use the training set to evaluate the performance of those pruned trees. Furthermore, we pick the tuning parameter α with the minimum MSE. Compared to the k-fold cross-validation, we do not fold the training observations into k approximately equal groups. In particular, in the case of a weighted regression tree, the lowest MSE and the test error also need to be weighted. The following simulation has proved that our method is feasible.

[1] https://CRAN.R-project.org/package=rpart

[2] https://www.rdocumentation.org/packages/gRain/versions/1.3-0/topics/cptable

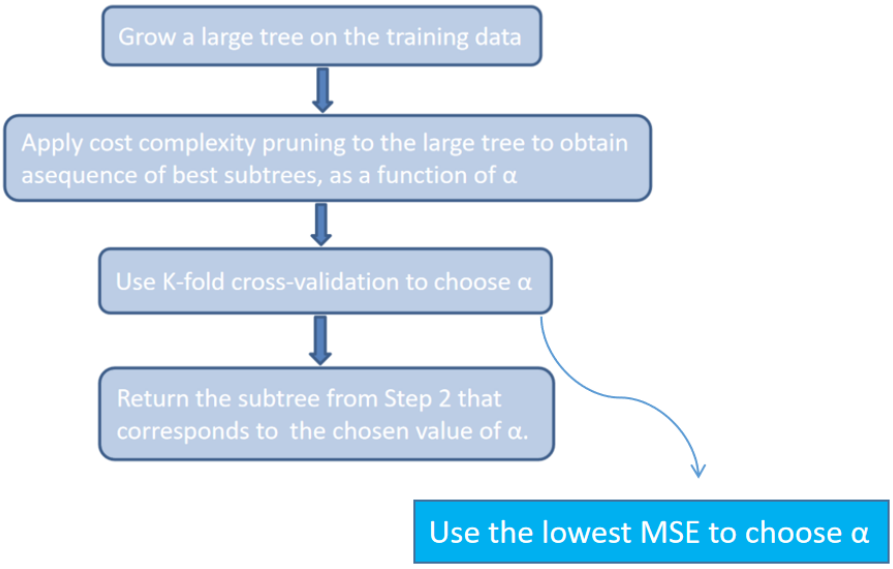


Figure 3.6: The display of the modified algorithm of a regression tree

3.2.1 Unbiased sampling

We first simulate unbiased sampling, also known as equal probability sampling. Split the population data randomly into the training set, which is 70% of the original data; the rest is the test set. So each individual in the population has the same sampling probability and weight. The training set is used to train the model. Then apply the lowest MSE and k-fold cross-validation, respectively, to tune the cost-complexity parameters α. Furthermore, the prediction errors obtained from the test set are used to compare the effectiveness of the two approaches. After simulating 300 times, the results obtained are shown in Figure 3.7.

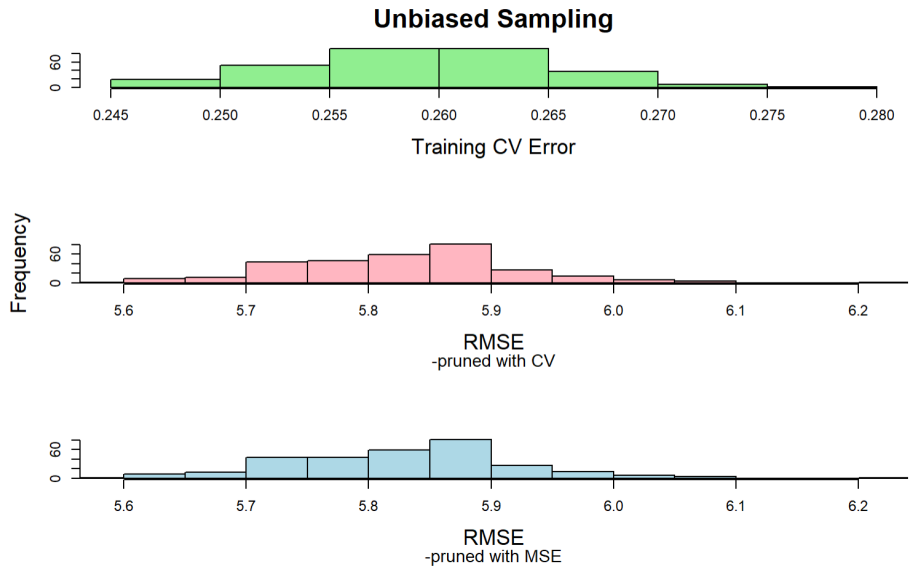


Figure 3.7: Simulation results for unbiased sampling

The training cross-validation errors range from 0.245 to 0.28 and are almost normally distributed. Also, it is much less than the others as the training error is often smaller than the test error in the regression tree model. RMSE here represents the root of the mean squared error, which is the prediction error on the test set. Both distributions of RMSE are slightly left skewed. Either by k-fold cross-validation or the lowest MSE to obtain the best cost-complexity parameters α, both methods perform almost the same. The p-value from the t-test is close to 1. Therefore, at the 5% significance level, we do not reject the null hypothesis that the RMSE is similar for the lowest MSE and k-fold cross-validation. It confirms that using the lowest MSE to tune the parameter is able to achieve similar results to the k-fold cross-validation.

3.2.2 Biased sampling

Now we do the biased sampling and discover the effect of weighting on unequal probability sampling, as survey data is usually sampled with unequal probability. Start

with sampling high values of x_{11}, x_{12}, y, and more c_1 and c_2 with 1s, where three numeric variables with a size that exceeds their upper quartiles are given a higher sampling probability. Hence, each individual does not necessarily have the same sampling probability. Trees are grown in three ways: with no weights, unscaled, and scaled weights. It is known that weights are the inverse of sampling probabilities. The scale() function is used to scale the weights. By default, the scale() function finds the mean and standard deviation of the vector and scales each element by subtracting the mean and dividing by the standard deviation. As weights must be positive, we only divide by the standard deviation when scaling. As discussed in chapter 2, we need to do the weighting for disproportionate allocation to perform better. Again simulated 300 times, Figure 3.8 shows the results.

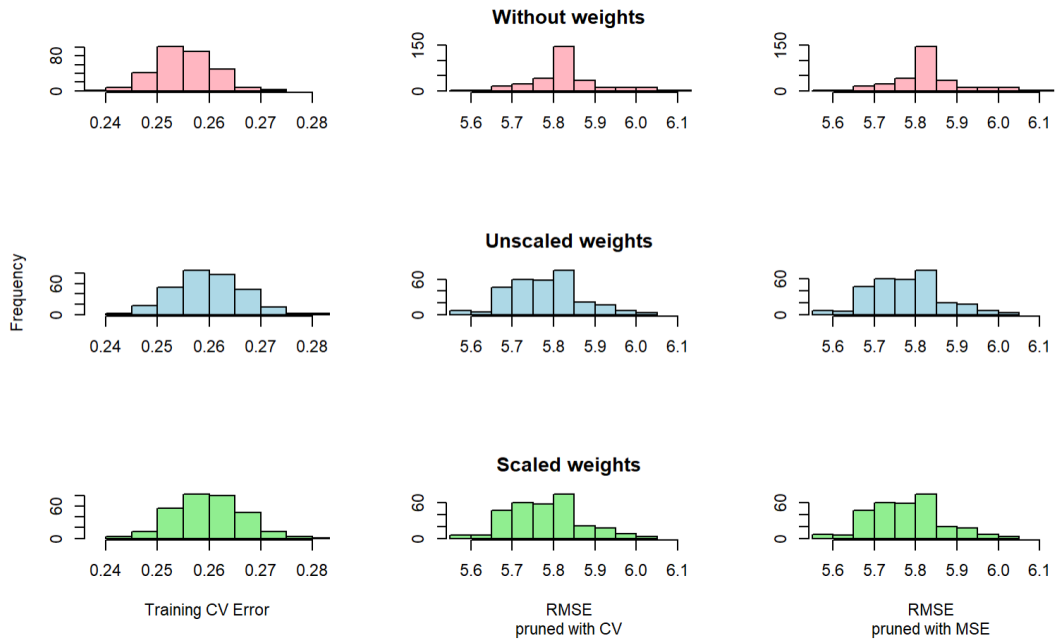


Figure 3.8: Simulation results for biased sampling

The training and testing error ranges are similar to those obtained from unbiased sampling. Without weighting, a large proportion of the prediction errors on the test

set are around 5.8. It is because the training set obtained by biased sampling has similar primary characteristics, and thus the prediction error of the test set obtained is more concentrated in one value. After weighting, the prediction error of the test set goes down. And the distributions of RMSE are slightly left skewed. Also they behaved more or less the same, whether with scaled or unscaled weights. Meanwhile, pruning the tree with the best cost-complexity parameter α obtained by the k-fold cross-validation method or with the lowest MSE gave almost the same results (no significant difference from the t-test), which is what we expected. Besides, according to the pairwise t-test, both RSME of pruning with CV or MSE with weights are lower than without weights regardless of whether the weights are scaled with significance (p-value < 0.1).

In conclusion, the simulation of biased sampling shows that weighting improves the prediction accuracy for unequal probability sampling. Therefore, it is important to use sampling weights when predictive modeling for complex survey data.

3.3 Clusters

For complex surveys, especially those with cluster samples, the common re-sampling method is replicate weights, representing the resamples by weights. For instance, the jackknife approach removes one cluster at a time. It is illustrated by various sets of weights, where the weights for one cluster are zero, and the weights for the other clusters are slightly increased. The bootstrap approach involves re-sampling entire clusters. Weights of zero are used for clusters that do not exist in the sample; the original weight is used for clusters that appear once, twice the initial weight is used for clusters that occur twice, and so on. For k-fold cross-validation, the data is randomly divided into training and test sets. If we also resample in that way for cluster sample, the cluster is disrupted and scattered between the training and test sets. A model trained in this way would yield less accurate results, notably if the target predictors varied widely across clusters. Our method for cluster samples uses the jackknife to split the sample into training and test sets. For each replicating, a cluster is used as a test set and the remainder as the training set to train the regression tree model, so the cluster will not be broken. We then weight the training set and choose the best tuning parameter α, based on the lowest MSE, as demonstrated in the first part of the project.

The R package we will use is *survey* [12], which analyses the complex survey samples. The as.svrepdesign() function generates replicate weights for a survey design. When we provide it with a survey design object and tell it the type of replicate weights we desire, it will construct them and return a survey design object with replicate weights. The weights() function takes the weights out of this object. For every replicate, there are certain observations with zero weight and others with non-zero weight. In our simulation, for each replicate, leave one cluster out with zero weight and others with non-zero weights. We train the regression tree to the ones with non-zero weight, using the lowest MSE to prune the tree. Furthermore, find the subtree related to the best cost-complexity parameters α. Then test the performance on the one with zero weight, and calculate the squared prediction error. Once we have the squared

prediction error for every observation for all the replicates, we can compute the mean square root of the prediction error. This is our idea for cross-validation in cluster samples. Figure 3.9 shows the method. In the end, we compare the three methods, original CV, CV with replicate weights, and our lowest MSE approach with replicate weights.

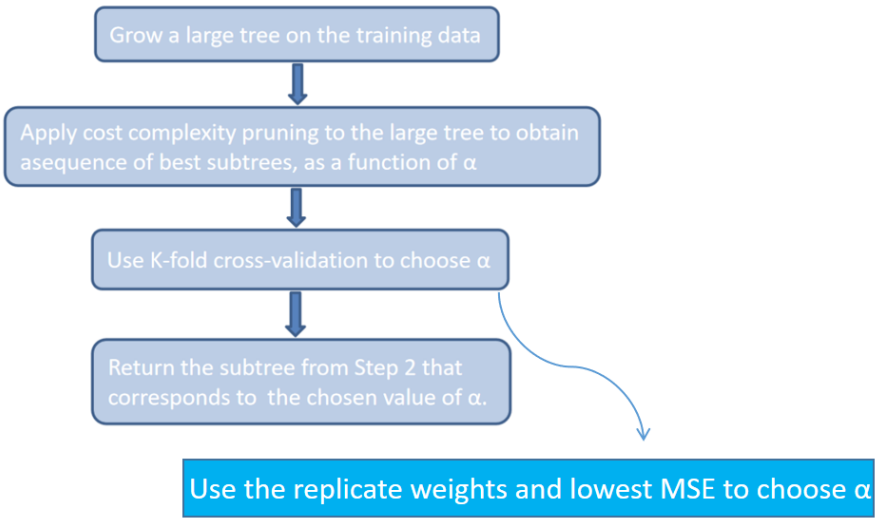


Figure 3.9: The display of the modified algorithm of a regression tree

3.3.1 One-stage cluster sampling

Let us start with the one-stage cluster sampling. With a total of 200 clusters, we randomly sampled 10, 50, and 100 clusters to compare whether the number of clusters correlated with the performance of the regression tree. First, we use the lowest MSE approach with replicate weights. For each replicate, leave one cluster out as the test set, and the remainders are the training set. Use the training set to train the regression tree model with weights and select the best cost-complexity parameters α, based on the lowest MSE. Then prune the tree with the chosen α and obtain the subtree. Finally, predict the test set and calculate the prediction error for each individual and the mean square root of the prediction errors for each replicate. Ten clusters have ten different training sets. Thus, the model was trained ten times. By extrapolation, 50 clusters are

50 times, and 100 are 100 times.

Theoretically, the larger the number of clusters and the larger the resample, the more accurate the prediction would be. Our simulation results also confirm this. Figure 3.10 demonstrates the distribution of RMSE obtained after each replicate weight for those three cluster samples. With 100 clusters, the prediction errors are most concentrated, mainly with a density of 8%. In contrast, the prediction errors for the 10 clusters are much more widely distributed. That is reasonable. For 10 clusters, data from only nine clusters were used to train the model, and the resulting model was used to predict the remaining one. Nevertheless, for 100 clusters, the model was trained with data from 99 clusters which means 11 times as much as the former.

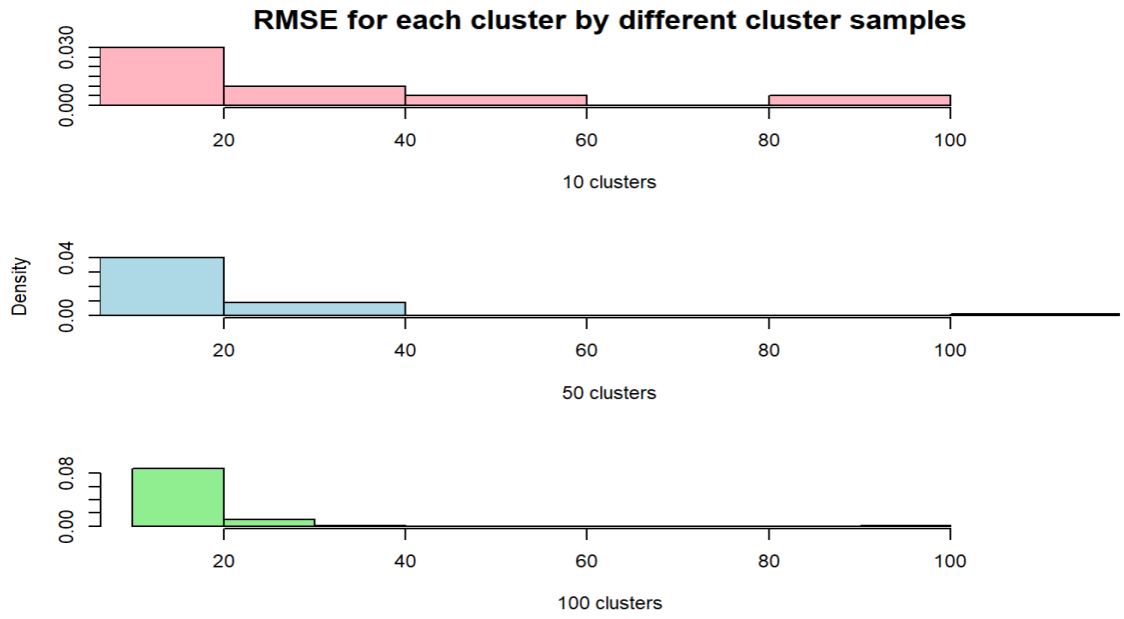


Figure 3.10: Simulation results I for one-stage cluster sampling

Figure 3.11 presents the mean square root of prediction errors for all individuals for each type of cluster sampling. As expected, 100 cluster sampling has the smallest variability for RSME, meaning better and consistent performance. 10 cluster sampling

with the largest variability of RMSE, which indicate inconsistent performance, and 50 cluster sampling has the medium performance.

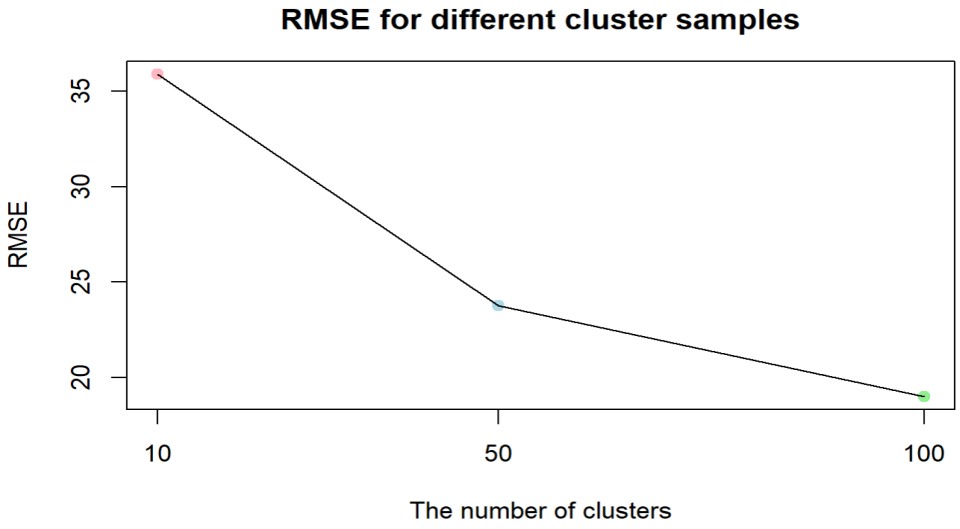


Figure 3.11: Simulation results II for one-stage cluster sampling

Next, we experiment with pruning the tree via k-fold cross-validation. The best cost-complexity parameters α is chosen based on the lowest cross-validation error. The above 100 cluster sample is used, and we simulate 100 times. For each simulation, the test and training set data is the same as the data from the previous replication weights. In this case, after 100 simulations, the prediction error is obtained for each data in the sample. Also, obtain the regression tree with the same weights as the previous replicate weights. In this way, it is easy to compare the results of the two methods. Lastly, use the original cross-validation to compute the RMSE and compare it to those from the replication weights. Table 3.4 shows the average square root of the prediction errors obtained by the standard cross-validation and replication weights. As the results obtained in the first half of our project, they behave similarly in performance after replicating weights, whether using the lowest MSE or k-fold cross-validation. In addition, replication weights perform significantly better than ordinary cross-validation. Figure 3.12 summarizes the prediction errors from two methods under replicating weights.

	RMSE
Original cross-validation	19.5
Replicate weights and cross-validation	19.2
Replicate weights and Lowest MSE	19.2

Table 3.4: Simulation results III for one-stage cluster sampling

For better visual contrast, we log the prediction errors. It confirms that the lowest MSE and k-fold cross-validation have the same performance. Figure 3.13 shows the difference between the absolute values of the prediction errors computed by the two methods for each individual in the sample. It is obtained by subtracting the absolute prediction error via k-fold cross-validation from that by the lowest MSE. The smaller the value is, the better the lowest MSE performs. The larger the value, the better the prediction performance of the k-fold cross-validation. The scatter plot shows that all of them are 0, which also says the lowest MSE performs the prediction as well as the k-fold cross-validation does. Therefore, for one-stage cluster sampling, choosing the pruning parameter via the lowest MSE is good as that from the k-fold cross-validation.

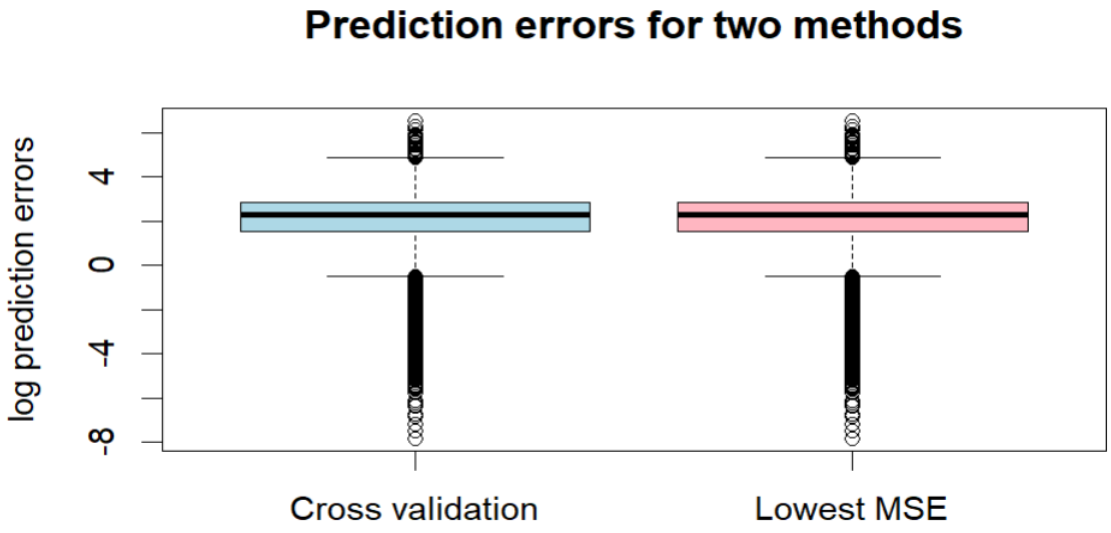


Figure 3.12: Simulation results III for one-stage cluster sampling with replicating weights

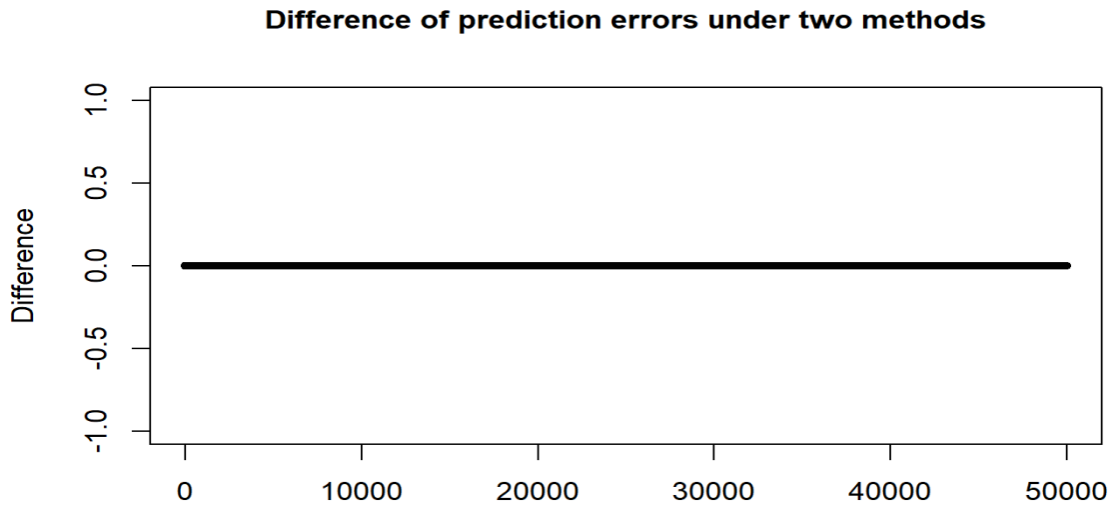


Figure 3.13: Simulation results IV for the lowest MSE and cross-validation with replicating weights

3.3.2 Two-stage cluster sampling

Now, it comes to the two-stage cluster sampling. The primary sampling unit is the stage one cluster, and the total number of clusters in the population is 200. The secondary sampling unit is the stage two clusters, and there are 10 stage two clusters under each stage one cluster. We randomly choose 100 levels from the 200 stage one clusters in the first stage. This time, we only select a subset instead of sampling all individuals in the selected cluster. This is the most common practice in real life to save money. Besides, stage one clusters are nested clusters as they are made up of multiple smaller clusters. Real data sets frequently contain nested clusters. So at the second stage, we randomly choose 7 out of the 10 stage two clusters for each selected cluster. Next, replicate the weights as before. Prune the tree with the lowest MSE and k-fold cross-validation to get the best cost-complexity parameter α, separately. In particular, the overall probability of each individual in this two-stage cluster sample being drawn is the product of the probabilities of being drawn at each stage. So a stage one cluster

is sampled with a probability of 100/200 = 0.5, and a stage two cluster is sampled from that cluster with a probability of 7/10 = 0.7. In that case, the overall sampling probability for each individual is $0.5 \times 0.7 = 0.35$. Therefore, the sampling weight is 1/0.35 = 20/7. Finally, do the original cross-validation and compare those results.

Table 3.5 shows the mean square root of the prediction error for each individual in the two-stage cluster sample under three methods. With replicating weights, the RMSE obtained from the lowest MSE is around two-thirds of that from the k-fold cross-validation, which is a significant improvement. However, there is some increase over the previous one-stage cluster sampling. This makes sense because the sample size is smaller than before. Besides, the predictive performance in the ordinary cross-validation is worse than that in the replication weights.

	RMSE
Original cross-validation	40
Replicate weights and cross-validation	31
Replicate weights and Lowest MSE	21

Table 3.5: Simulation results I for two-stage cluster sampling

Figure 3.14 demonstrates the summary of the prediction errors obtained from the lowest MSE and k-fold cross-validation with replicating weights. The pairwise t-test showed that the prediction errors obtained from the lowest MSE are significantly lower than that from k-fold cross-validation (p-value < 0.01).

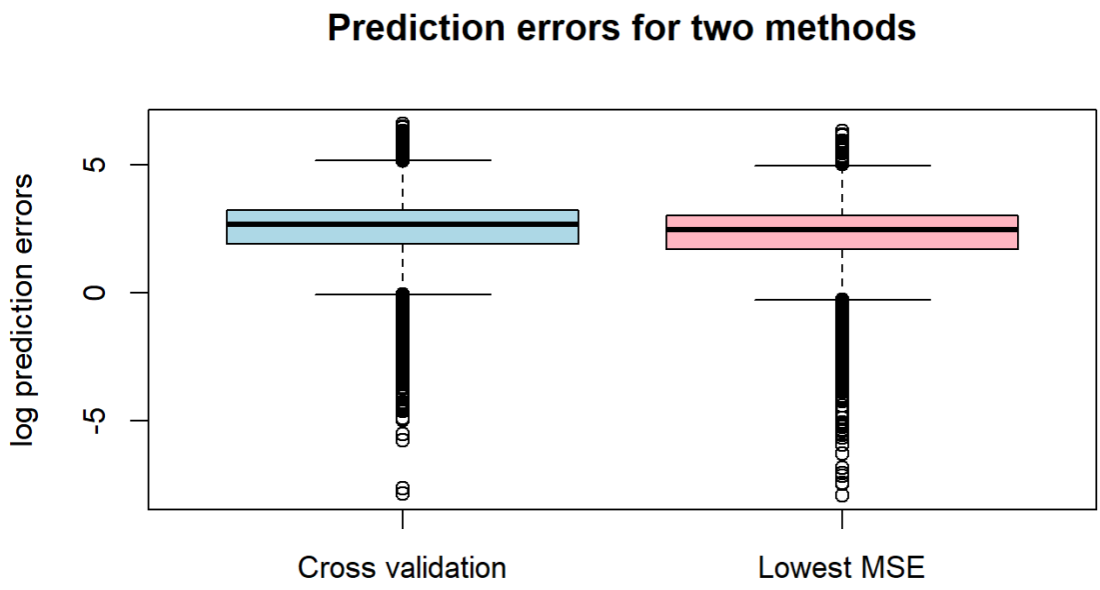


Figure 3.14: Simulation results II for two-stage cluster sampling with replicating weights

Our simulation shows that, for two-stage cluster sampling, using the lowest MSE to prune the tree is generally better than that via k-fold cross-validation when replicating weights. The sample characteristics vary by cluster, so it is better to tune the cost-complexity parameter directly from the lowest MSE of all the clusters in the training set than to break up the cluster data into k folds. Replicating weights is also better than ordinary cross-validation.

Chapter 4

Application

This chapter studies actual survey data to illustrate the points expressed in chapter 3's conclusion. The National Health and Nutrition Examination Survey (NHANES) and Student performance survey in California schools (API) are used to verify the findings from the previous chapter. As we have shown sufficient evidence on the importance of weights for unequal probability sampling in prediction models, the feasibility of replacing k-fold cross-validation with the lowest MSE, and cluster sampling with replicate weights, this chapter will not include any additional verification under each result.

4.1 National Health And Nutrition Examination Survey

The National Health and Nutrition Examination Survey (NHANES), one of many health-related programs run by the National Center for Health Statistics (NCHS), is used to gather information on the physical and dietary health of the non-institutionalized civilian resident population of the United States. It is distinctive in that it uses in-person interviews and a standardized physical examination in a mobile examination center to gather data on demographics, health, and nutrition at the individual level. Height,

weight, blood pressure, and other objective health status indicators are measured as part of the examination, along with collecting blood and urine samples for laboratory analysis. The sample design consists of four-stage, multi-year, stratified, clustered samples released at 2-year intervals, of which primary sampling units are chosen with probabilities proportionate to size. So PSUs with higher proportions of people in the subgroups selected for oversampling are given relatively higher selection probabilities.

We will use data from the 2017-2018 survey cycle. The goal is to use a regression tree to predict the total amount of cholesterol in the blood based on five predictors, including gender, age, ethnicity, weight, and height. Figure 4.1 demonstrates the distribution of total cholesterol in our sample. It ranges from 76 to 446, and the median is 176 mg/dL.

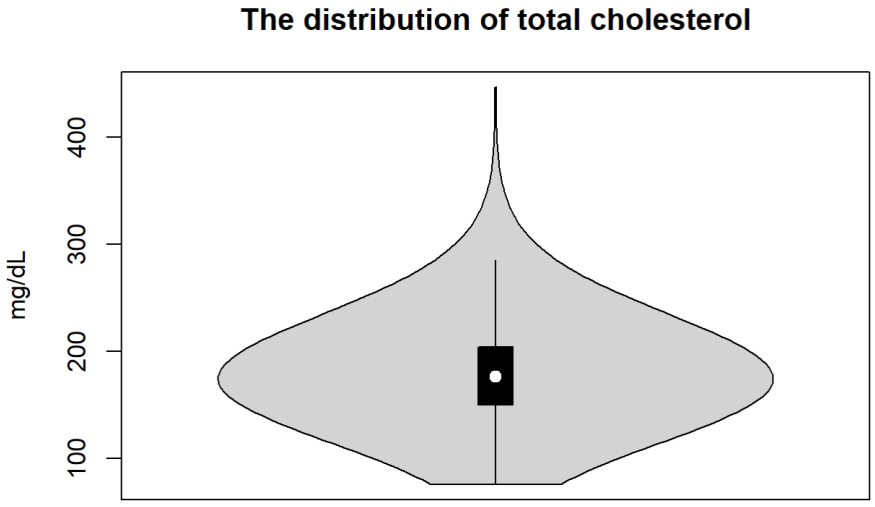


Figure 4.1: The violin plot of total cholesterol

As described earlier, it is an unequal probability sample. We perform weighting and tuning the cost-complexity parameter α via the lowest MSE. Figure 4.2 shows the distributions of RMSE in different scenarios and the training cross-validation errors after running the model 300 times, as we did before with 300 simulations.

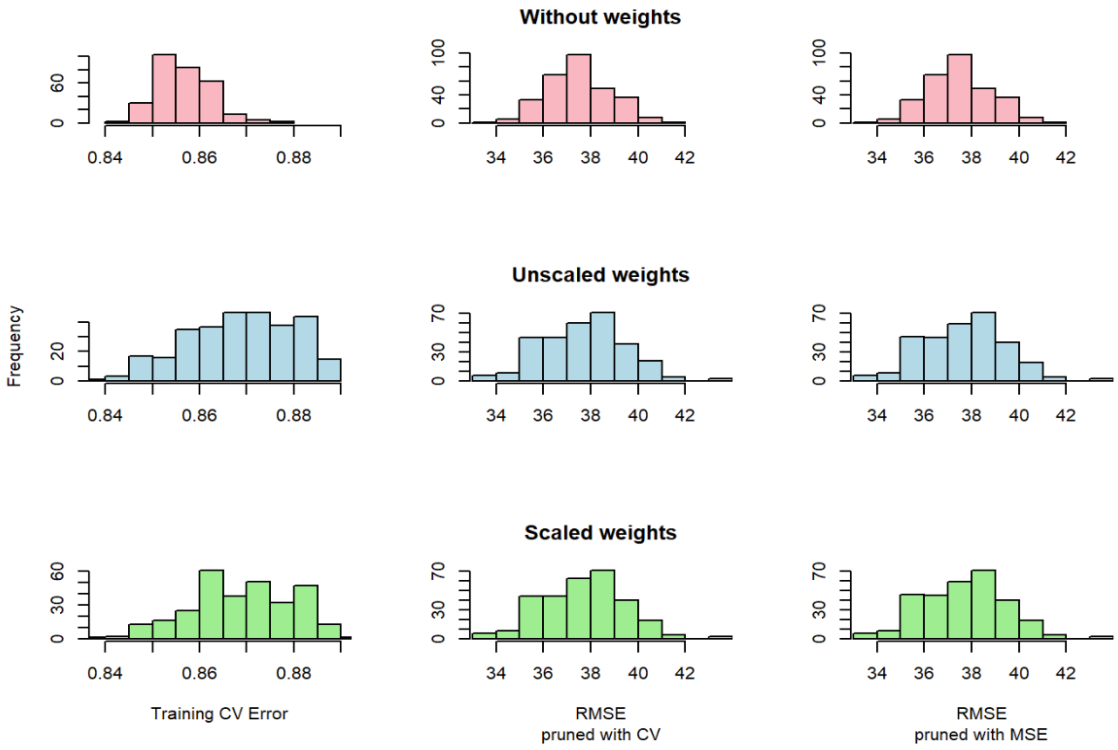


Figure 4.2: Results for NHANES 2017-2018

The results confirm our previous simulation study in chapter 3.2. The average prediction error is generally smaller after adding weights to the unequal probability sample. The performance after adjusting the weights is almost the same as the performance of the initial weights. The model obtained by choosing the best tuning parameter α with the lowest MSE performs as well as the one obtained by the standard k-fold cross-validation method. Besides, the training cross-validation error after weighting goes up a bit.

The stratified cluster sample NHANES III also studied belongs to the single-stage cluster sampling. It includes 81 primary sampling units, mainly individual counties, which are stratified. We apply replication of weights to compute the mean squared

root of the prediction errors. This time we aim to predict the amount of the 25-Hydroxyvitamin D(25OHD) by age, sex, height, and weight. Figure 4.3 displays the distribution of the 25OHD concentrations. The average concentration is about 57 nmol/L.

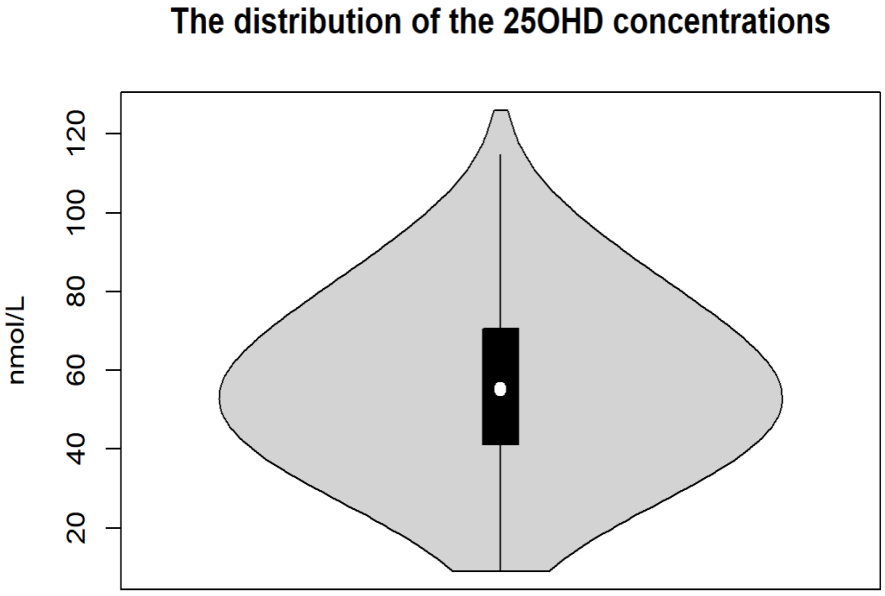


Figure 4.3: The violin plot of the 25-Hydroxyvitamin D concentrations

Table 4.1 lists the root mean square errors from replicating weights and the ordinary cross-validation. It says that the results obtained are almost identical when replicating weights, either with the lowest MSE or k-fold cross-validation. Figure 4.4 displays the summary statistics of the absolute value of the prediction error for each individual under those two methods with replication weights. The boxplots show no observable difference between them, which is confirmed by the p-value of 0.26 from the t-test. Besides, replicating weights performs slightly better than ordinary cross-validation. Therefore, it confirms the simulation results in chapter 3.3.1.

	RMSE
Original cross-validation	19.2
Replicate weights and cross-validation	18.8
Replicate weights and Lowest MSE	18.8

Table 4.1: Results I for NHANES III

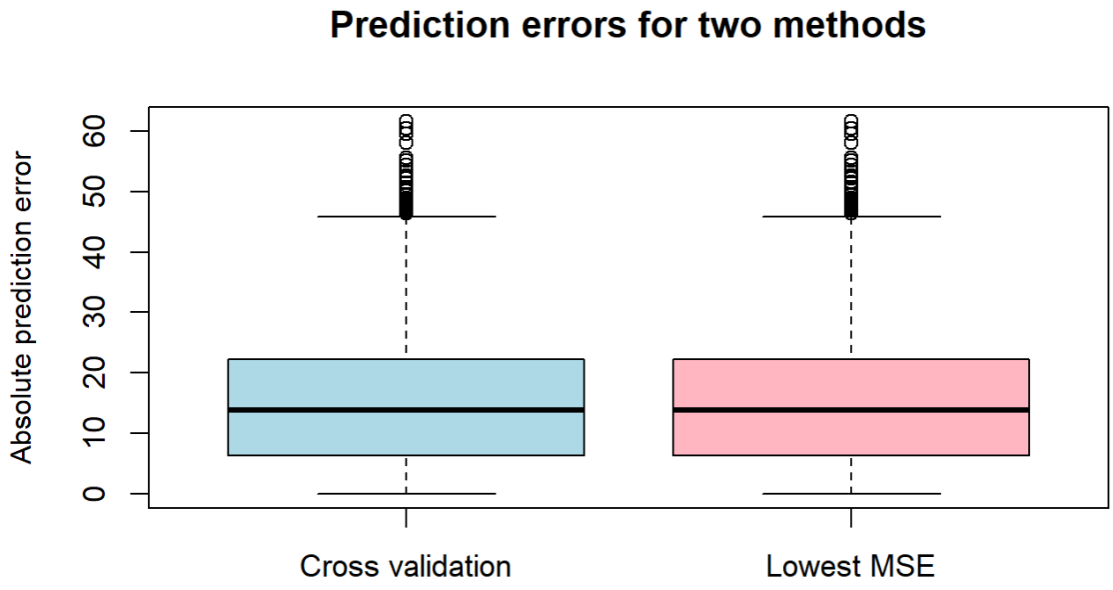


Figure 4.4: Results II for NHANES III with replicating weights

4.2 Academic Performance Index in California schools

For API survey data, the Academic Performance Index is calculated for all California schools based on Students' performance on standardized tests. Each school with at least 100 Students is represented in the data sets, along with different probability samples. We will study the data set apiclus2, a two-stage cluster sample. The primary sampling unit is the school district, and the second sampling unit is the schools within the selected district. All schools in the district were chosen if there were fewer than five; otherwise a random sampling of five schools was used. Figure 4.5 displays the distribution of the Academic Performance Index, and the mean value is about 704.

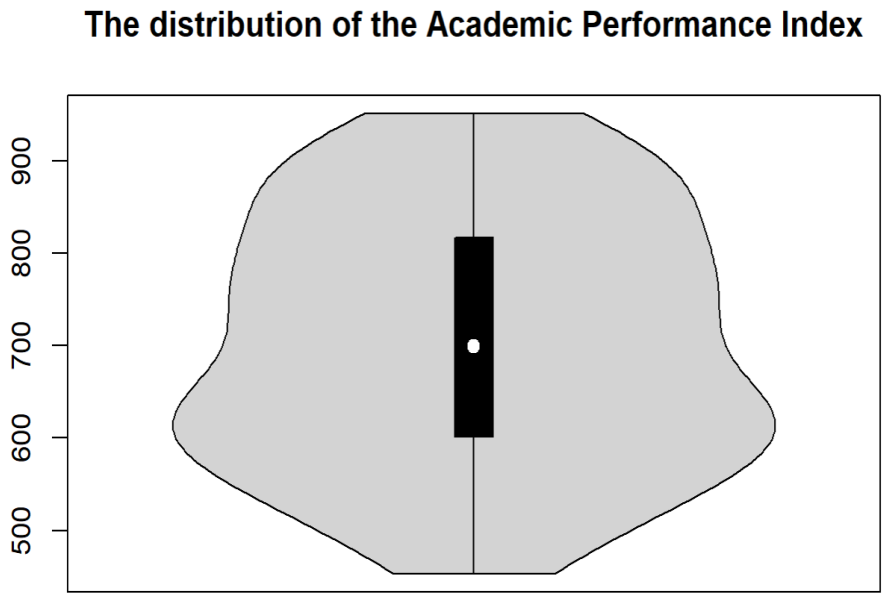


Figure 4.5: The violin plot of the Academic Performance Index

Table 4.2 shows the root mean square errors from replicating weights and the ordinary cross-validation. Using the lowest MSE to tune the cost-complexity parameter α is better than the k-fold cross-validation in the two-stage cluster sample and that from the original cross-validation. Also, replicating weights performs better than normal cross-validation. Figure 4.6 demonstrates the summarized distribution of the absolute

value of the prediction error for each individual under the two methods with replication weights. There is no significant difference between the two groups of prediction errors, as the p-value of the t-test is 0.11. The result is a bit different from our simulation in chapter 3.3.2. Replicate weights still performs better than the standard cross-validation approach, but the two methods with replicate weights have similar performance.

	RMSE
Original cross-validation	92
Replicate weights and cross-validation	83
Replicate weights and Lowest MSE	73

Table 4.2: Results I for API

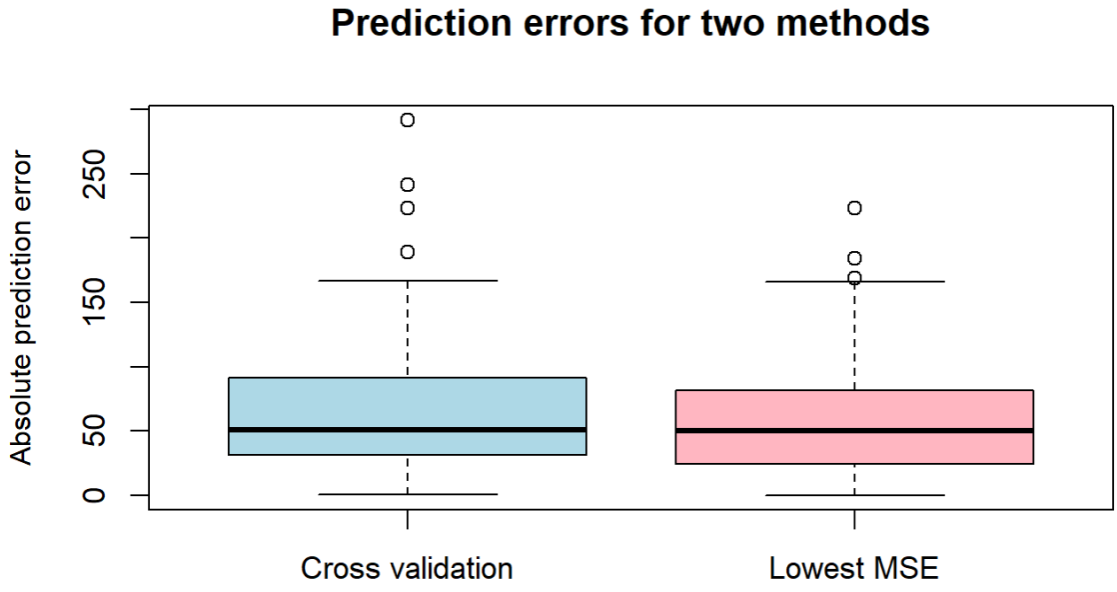


Figure 4.6: Results II for API with replicating weights

Chapter 5

Conclusion

5.1 Main Findings

This project is focused on using predictive models in a complex survey setting, in which we utilized the sampling weights and modified cross-validation techniques for complex survey data.

We have verified through simulations that, for unequal probability sampling, the prediction accuracy is improved after the model is weighted. The performance is the same regardless of the original or scaled weights. Also, tuning the parameter with the lowest MSE gives as good results as the original cross-validation method.

For cluster sampling, replicated weights perform better than ordinary cross-validation, especially for multi-stage survey sampling. This is because, after multi-stage sampling, the sample characteristics vary more significantly for each cluster. Replicating weights does not separate the data of the same cluster and disrupt the distribution in the training and test sets.

They are further confirmed in chapter 4 by the real examples. For predictive modeling of complex survey data, it is better to do the weighting and use the replicated weights with cross-validation.

Hence, the general conclusion we desired in this dissertation can be drawn. Using sampling weights for complex survey data in predictive models is necessary.

5.2 Future Directions

In this project, we used only one predictive model, the decision tree. In addition, various prediction models can be used for prediction with survey data, such as random forest. Besides that, neural networks are another popular prediction model. Neural networks, commonly referred to as artificial neural networks, are a class of deep learning technologies modeled after how human brain neurons function. They are tremendously helpful for analyzing huge data sets and are frequently used to solve challenging pattern recognition challenges. They function effectively when some variables are unknown and are excellent at managing nonlinear relationships in data [5]. However, other methods might lack the ability to apply weights to the prediction process. We could explore how to use survey sampling weights in more complex prediction models such as random forest and neural networks.

Apart from focusing on prediction models, we could investigate how survey cross-validation can take post-stratification and non-response adjustments into account when adjusting the sample weights.

References

[1] T. Lumley, *Complex surveys: a guide to analysis using R*. John Wiley & Sons, 2011.

[2] D. Pfeffermann, "The use of sampling weights for survey data analysis," *Statistical methods in medical research*, vol. 5, no. 3, pp. 239–261, 1996.

[3] D. G. Horvitz and D. J. Thompson, "A generalization of sampling without replacement from a finite universe," *Journal of the American statistical Association*, vol. 47, no. 260, pp. 663–685, 1952.

[4] S. L. Lohr, *Sampling: design and analysis*. Chapman and Hall/CRC, 2021.

[5] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.

[6] J. Wieczorek, C. Guerin, and T. McMahon, "K-fold cross-validation for complex sample surveys," *Stat*, p. e454, 2022.

[7] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Routledge, 2017.

[8] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, *et al.*, "Top 10 algorithms in data mining," *Knowledge and information systems*, vol. 14, no. 1, pp. 1–37, 2008.

[9] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[10] L. Rokach and O. Maimon, "Top-down induction of decision trees classifiers-a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 4, pp. 476–487, 2005.

[11] T. P. Morris, I. R. White, and M. J. Crowther, "Using simulation studies to evaluate statistical methods," *Statistics in medicine*, vol. 38, no. 11, pp. 2074–2102, 2019.

[12] T. Lumley, "Package survey," *Available at the following link: https://cran. r-project. org*, 2020.