Quantifying Structure in Random Forests

by

Hannah Sutton

B.Sc., Simon Fraser University, 2018

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

Under Individualized Interdisciplinary Studies with
Graduate and Postdoctoral Studies and
Department of Mathematics
Faculty of Science

Declaration of Committee

Name:	**Hannah Sutton**
Degree:	**Master of Science**
Thesis title:	**Quantifying Structure in Random Forests**
Committee:	**Chair:** Ailene MacPherson Assistant Professor, Mathematics
	Caroline Colijn Supervisor Professor, Mathematics
	Lloyd Elliott Committee Member Assistant Professor, Statistics and Actuarial Science
	Ben Adcock Committee Member Professor, Mathematics
	Derek Bingham Examiner Professor, Statistics and Actuarial Science

Abstract

Random forests are often regarded as black-box machine learning models. They are sufficiently complex that they are not easily interpretable. This fact has inspired a variety of research into improving the interpretability of random forests, which is the focus of this thesis; specifically, we wish to capture dissimilarities between random forest trees using several comparison functions on the decision trees that comprise the random forest, allowing the structure of the random forest to be quantified. These include a phylogenetic metric designed for transmission trees, as well as others we developed that involve the count and location of variables in each tree, as well as the depths of the trees. This allows us to visualise an underlying grouping of the trees using a heatmap and hierarchical clustering, and analyze the predictive accuracy of the decision tree clusters. Finally we propose a method for generating random decision trees, which we then use to generate synthetic data using a small set of trees. We use the random forest trained on this data to determine which comparison functions are statistically significant and contribute to the overall clustering. Additionally, we investigate whether or not the random forest is capable of recovering the original trees that the data was created from.

Keywords: Random forests, comparison function, heatmap, hierarchical clustering, synthetic data

Acknowledgements

I would like to express my gratitude to my two supervisors, Lloyd Elliott and Caroline Colijn, who guided me throughout this thesis. I would like to extend my gratitude to my undergraduate supervisor, Nilima Nigam, with whom I completed my first research project and who encouraged me to pursue graduate work. I would also like to thank my family and friends for supporting me endlessly. Lastly, I wish to especially thank my wife, Dana, and our two cats, Ruby and Tuna.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Random forests are a state-of-the-art statistical learning method used for classification and regression and have been commonly used since they were first introduced by Breiman (2001). Random forests are often favoured due to their versatility, parallelizability, and the fact that they work well with high dimensional and non-linear data, and are moderately robust to outliers and noise in the data. Developments of packages in programming languages such as R and Python also make them accessible and simple to implement, making them a top choice for complicated modelling problems. However, random forests are often criticised for their complexity and low interpretability. If a random forest has a low accuracy or misclassifies data objects it can be difficult to explain why that misclassification occurs.

In this thesis, we provide a method for clustering decision trees. Our primary goal is to quantify the differences in the decision trees of which a random forest is composed. The method we develop to achieve this goal works as follows: first we define five different comparison functions that capture differences between decision trees using either the topology of the trees or the internal workings of the splitting variables. (These comparison functions are non-negative symmetric functions $d(T_i, T_j)$, where T_i and T_j are decision trees extracted from a random forest. Comparison functions are similar to metrics, but unlike metrics $d(T_i, T_j) = 0$ does not necessarily imply that $T_i = T_j$.) The comparison functions are then applied to the decision trees of a random forest in a pairwise manner, resulting in a symmetric $N \times N$ difference matrix, $D^{(k)}$, where N is the number of trees in the random forest. Here, $D^{(k)}_{i,j} = d_k(T_i, T_i)$, according to the k-th comparison function d_k. We normalise (the details of the normalisation are provided in Chapter 2) the difference matrices, and sum them together into a single $N \times N$ symmetric matrix D. We use methods to visualise D and cluster the trees into groups based on D. We use this clustering to find representative trees for the random forest. We then conduct an analysis to determine if these representative trees can be used to summarize the random forest, and hence reduce the problem of interpreting the entire random forest to simply interpreting the handful of trees.

We visualise the difference matrices using heatmaps, by finding dendrograms, and by performing mutlidimensional scaling (MDS) on the clusters. The specific goal here is to determine if these clusters can be used to understand how the random forest makes decisions. Ultimately, we wish to find a representative tree for each cluster (what we will define to be a centre tree) and reduce the task of analysing the entire random forest to simply analysing this subset of trees. In our experiments we found that when a random forest is restricted to K centre trees, the accuracy of these trees is the same as that of a random forest restricted to K trees randomly selected from the random forest. This suggests that random forests cannot easily be represented by a small set of representative trees from clusters. We initially conducted our experiments on four datasets (one for regression and 3 for classifications). Furthermore, in a simulation study we designed and implemented a novel way for generating decision trees and created a synthetic dataset from these trees. In this simulation study we found that a random forest trained on data created from three decision trees are not capable of recovering the 'ground truth' trees.

Random forests are applicable to modelling problems in numerous fields of research. In this work, we apply our method to five unique datasets. The first dataset is the classic Iris dataset (Dua & Graff, 2017), which classifies species of irises by characteristics of the flowers. The second is a drug use dataset (Fehrman et al., 2015) which classifies a person's overall use level of illegal substances with personality traits and indication of types of drug usage in the past. The third dataset combined genome sequencing of the 1995 outbreak of tuberculosis in London with medical data from patients infected with tuberculosis to predict whether or not the patients are infectious. The last dataset (Harrison Jr & Rubinfeld, 1978) is constructed from the housing market in Ames, Iowa, which associates characteristics about properties with their sale prices. These datasets are all explained in more detail in Chapter 3. Note that the broad range of these datasets demonstrates that random forests are applicable to various non-related fields of study.

In the remainder of Chapter 1 we discuss related work to this thesis, as well as give detailed explanations of the important tools we use such as hierarchical clustering, and classification and regression trees. In Chapter 2 we give an overview of the methods we employ to visually represent the differences between the decision trees of the random forests, and how we use dendrograms to group the trees into clusters for further analysis. We also give descriptions of the comparisons we developed to quantify the differences between the trees. In Chapter 3 we discuss our experiments and results. This includes more detailed descriptions of the datasets we use to train the random forests, as well as a synthetic dataset we created. Lastly, in Chapter 4 we conclude the thesis by summarizing the results and present ideas for future work.

1.1 Related work

In 2019, Cynthia Rudin published a paper in which she advocates that uninterpretable, or *black box*, machine learning methods should not be used in high stakes contexts such as healthcare and criminal justice systems and that only interpretable models should be used (Rudin, 2019). When a statistical learning model is referred to as black box, this means that the inner workings of the model are complex and not easily explainable (Belevitch, 1962). Rudin reasons that our inability to understand the way in which predictions are made by black box methods results in predictions that we cannot prove are faithful to what is intended to be calculated. This can thus lead to consequences with no accountability. Rudin's solution is that researchers should focus more efforts on creating machine learning methods that are interpretable, and to abandon work that attempts to explain the internal intricacies of black box models and understand the reason as to why they are often incredibly accurate but other times are not. Black box models have existed for decades now, and because of their improved accuracy over more explainable models, are already an integral part of many fields and industries ranging from insurance to the most successful tech companies as a result of them being accurate. Interesting examples of this include the use of XGBoost to predict motor insurance claims (Pesantez-Narvaez et al., 2019), and the computer vision program DeepDream created by Google engineer Alexander Mordvinstev that uses a convolutional neural network to find and enhance patterns in images in order to give them a dream-like appearance (Mordvintsev et al., 2015). Rudin raises an important consideration that in fields such as healthcare and criminal justice, an incorrect prediction can have terrible consequences for a human life; however, since black box methods are a strongly established part of machine learning and computer science, these consequences motivate the need for explanations of the workings of such models, rather than the cessation of their use.

Previous work focused on interpreting random forests include *Forest Floor* (Welling et al., 2016), a method that aids in visualising tree based ensemble models by analysing the mapping between the feature space (the input vector to a classification problem) and the prediction space (the output to a classification problem: the class). Specifically, they define local increment vectors as the progression of an observation through the prediction space, and the final prediction is simply a sum of all the local increment vectors. Since it is a tree based method, the local increments are associated with progression between the nodes of the trees. Therefore, they can take a subset of local increment vectors for which the splitting parent node was a specific feature, and form the sum of this subset of local increment features. They call this sum the feature contribution. By plotting the feature contributions, we can determine which features are contributing to the overall prediction.

Another interesting line of work for visualising random forests is a tool called ReFINE (Kuznetsova et al., 2014). This tool consists of several visualisation methods for different aspects of the random forest, such as variable importance and proximity (variable importance is defined in Section 1.2.1). The trees themselves are represented by icicle plots, so named because the plots themselves resemble the shape of icicles, where one can see the splitting attribute of each node, and the leaf nodes are colour coded by classes. Proximity refers to how strongly instances of data are connected to each other, or how close they are, and is measured by the percentage of trees that classify the pair of data points in the same way (proximity is defined in Kuznetsova et al. (2014)). ReFINE visualises proximity by creating a heatmap of the proximity matrix, as well as highlighting the classes assigned to each data point on the edge of the heatmap. The next visualisation method is interactions, which represents the relationship between features. For this, the visualisation is simply a visualisation of the feature itself, which can either be numerical or categorical. For numerical features, the diagram shows the range of the feature, with the split points highlighted along the axis, and the assigned classes along the top. For categorical features, the total width represents the number of instances in the dataset, and the width of each category represents the number of instances; the classes are also overlaid on the top of the visualisation. Lastly, prototypes, which refer to how variables relate to the classification, are visualised together with a proximity matrix as a scatter plot. This allows the user to estimate how easily classes of a dataset are separated.

Tree metrics have preciously been defined in fields such as phylogenetics (the study of ancestral relationships between individuals and groups of organisms), and therefore could be modified as necessary to quantify differences in random forest trees. An example of related work on metrics for phylogenetic trees includes Billera et al. (2001). In this work, the authors consider a continuous space that models the set of all phylogenetic trees that have a fixed set of leaf nodes. This space has a natural metric of non-positive curvature, which allows a way of measuring distances between phylogenetic trees, as well as combining or averaging trees that have identical sets of leaf nodes.
An earlier work on metrics for phylogenetic trees is presented by Robinson and Foulds (1979). This paper extends previous work on the comparison between unweighted, labelled trees to comparisons between weighted, labelled trees. Such a metric sets a precedent for comparing different types of trees, such as transmission trees (Kendall et al., 2018), ranked evolutionary trees (Kim et al., 2020), and unlabelled, rooted trees (Liu, 2021). Naturally this leads one to consider metrics for decision trees. In this work, we will consider metrics for the decision trees that make up a random forest.

1.2 Classification and Regression Trees

A decision tree is a statistical learning method that works by recursively splitting the feature space. A tree is a rooted binary acyclic graph. Decision trees are capable of both regression and classification, and are also known as *classification and regression trees* (CART; Breiman et al. 1984). However, the adoption of CART analysis was initially slow due to the complexity of the method (Lewis, 2000), and because of the dissimilarity between CART and traditional models. Formally, a decision tree is a tree, along with splitting variables and splitting points associated with each node of the tree (i.e., the nodes are labelled by a pair indicating the splitting variable and the splitting point for that variable). Decision trees are created using a recursive partitioning algorithm. Here binary means that at each node (the parent node) in the tree can be split into two child nodes. The algorithm is recursive because each node has the potential to split into two child nodes. The two child nodes can in turn can also be split, and so on until a stopping criteria is met. Nodes that remain children nodes and do not split further are referred to as leaf nodes. CART is a partitioning algorithm because at each splitting node, the data is partitioned into two groups based on a splitting variable, as explained below.

To build a decision tree, T, the CART algorithm starts at the root node, which includes all observations in the training data. In order to find the best splitting variable and value, CART uses an exhaustive search of all possible splitting variables (Lewis, 2000). To identify the best split variable and value, the algorithm seeks to maximize the purity of the two child nodes, making the data in each of the subsets purer than the data in the parent subset. Many measures of purity exist, the most common for classification is the Gini impurity (Breiman et al., 1984); Gini impurity is a measure of non-homogeneity and is defined by:

$$\text{Gini} = 1 - \sum_{i=1}^{n} (p_i)^2. \tag{1.1}$$

Here n is the total number of classes, and p_i is the probability of class i within a node. When a node is reached such that no significant decrease in impurity is possible, this node becomes a leaf node (a terminal node that is not split further). The overall class assigned to a leaf node is determined by:

$$p(i_0|t) = \max_i p(i|t) \tag{1.2}$$

Here $t \in \tilde{T}$ such that $\tilde{T}$ is the set of all leaf nodes, and i_0 is the class assigned at this node. Then the tree can be used as a classifier for a new observation by assigning the new observation to the class of the leaf node it arrives at after progressing through the tree.

Often with this stopping criterion, the tree is grown to be too large and will have a higher misclassification rate than optimally sized trees. On the other hand, if the tree is too small it will not use some of the information in the training set that may be needed to make accurate predictions, again resulting in a higher misclassification rate. Therefore, previous work on decision trees was centred on finding an appropriate stopping rule. Eventually it was concluded that looking for a stopping rule was the wrong approach, and instead pruning larger trees was a better method. Therefore, the optimal method is to grow a tree, T_{max}, that may be too large, and then prune it sequentially until the root node is all that remains, and then choose the right sized tree from the subsequent set of subtrees Breiman et al. (1984) (note that the random forest package in R does not prune the trees). The maximal tree is reached by letting the procedure of growing the tree continue until all terminal nodes are too small to split any further, or only contain observations that belong to the same class. The pruning method developed by Breiman et al. (1984) is called *minimal cost-complexity pruning.* Before introducing this pruning method, we will first define the resubstitution estimate.

The resubstitution estimate, defined mathematically as $r(t) = 1 - \max_i p(i|t)$, tests the accuracy of a classifier by applying it to observations for which the classes are known, otherwise known as training data. However, the main weakness of the error rate is that it is derived from the same dataset from which the tree is built; hence, it underestimates the true misclassification rate. Denote $R(t) = r(t)p(t)$. Then, the resubstitution estimate for the overall misclassification rate of the tree classifier T is given by $R^*(t) = \sum_{t \in \tilde{T}} R(t)$. The idea behind behind *minimal cost-complexity pruning* is as follows. For any subtree $T \preceq T_{max}$, define its complexity $|\tilde{T}|$ as the number of terminal nodes in T. Then let $\alpha \geq 0$, $\alpha \in \mathbb{R}$ be the the complexity parameter, and define the cost-complexity measure as $R(\alpha) = R(T) + \alpha|\tilde{T}|$. The complexity is thus the misclassfication cost of the tree added to a penalization term representing the complexity of the tree. For each value of α find the subtree $T(\alpha)$, $T(\alpha) \preceq T$ that minimizes $R_\alpha(T)$, $R_\alpha(T(\alpha)) = \min_{T \preceq T_{max}} R_\alpha(T)$. Intuitively, if α is small then the penalty for having a large number of terminal nodes is small and hence $T(\alpha)$ will be large. Conversely, if α is sufficiently large then $T(\alpha)$ will become just the root node and the tree will be completely pruned.

By sequentially pruning the tree, the result is a set of subtrees $\{T_k\}, k \geq 1$, where $T_1 \succ T_2 \succ \ldots \succ \{t\}$ where $\{t\}$ the root node. Corresponding to each subtree are the penalization values $\{\alpha_k\}$ where $\alpha_k \leq \alpha \leq \alpha_{k+1}$. Therefore, the next step is to find the value of α_k to prune to the next subtree in the sequence. For example, if we want to to prune T_1 to get T_2, the process is as follows. Consider all internal nodes $t \in T_1, t \notin \tilde{T}_1$, we wish to find the node $\bar{t}$ such that if the subtree with $\bar{t}$ as the node is pruned, the cost complexity of the pruned tree is equal to the unpruned tree. To do this, set $R_\alpha(t) = R(t) + \alpha$, and for the subtree T_t let $R_\alpha(T_t) = R(T_t) + \alpha|\tilde{T}_t|$. Then as long as $R_\alpha(T_t) < R_\alpha(t)$, the subtree T_t has a smaller cost complexity than the single node t. We can use this find the inequality for α, getting

$$\alpha < \frac{R(t) - R(T_t)}{|\tilde{T}_t| - 1}.$$

Therefore, the function

$$g_1(t) = \begin{cases} \frac{R(t)-R(T_t)}{|\tilde{T}_t|-1}, & t \notin \tilde{T}_1 \\ +\infty, & t \in \tilde{T}_1 \end{cases} \tag{1.3}$$

can be used to find $\bar{t}$ such that $g_1(\bar{t}) = \min_{t \in T_1} g_1(t)$ and then define $\alpha_2 = g_1(\bar{t}_1)$. Lastly, it remains to choose a final tree from the list of subtrees. The most preferable method is the use of a test set from cross validation. This method is computationally efficient, although expensive, but overall is the most effective use of the data and gives useful information about the tree structure (Breiman et al., 1984).

1.2.1 Random Forests

Random forests are a statistical learning method developed by Breiman (2001), which extend decision trees by grouping together several base learners, such as decision trees, and then averaging together the resulting models. Random forests are so named for two reasons. First, they implement a bootstrap aggregation (bagging) method during learning to select a *random* subset of the data with replacement to train each tree in the forest. Second, at each node a *random* subsample of the variables are eligible candidates to be used for splitting (instead of allowing splits to be formed using any of the variables). In applications, the number of variables randomly selected at each node for splitting is is $p/3$ for regression, and $\sqrt{p}$ for classification, where p is the total number of variables (James et al., 2013). The reason we choose a subset of the variables is that this helps to reduce the correlation between the trees, which reduces the variance of the predictions. Consider if there was a particularly strong predictor, without a subsetting method it is likely that every tree in the forest will choose this variable as the first splitting variable, meaning the trees will be similar. By creating a random subset of the variables, the strongest variable may not be selected. For classification, $\sqrt{p}$ is the size of the subset because higher values did not improve the prediction accuracy but the correlation of the trees increased. For regression, it was found that the prediction accuracy increased but the correlation increased slowly, hence a higher number of variables is considered for the subset (Breiman, 2001).

Bagging is an important aspect of random forests. While empirically, we find that bagging increases the overall accuracy of random forest predictions, bagging also allows us to calculate accuracy estimates using 'out-of-bag' data samples, which can be used for error estimation during the training process. For the out-of-bag error estimate, consider a training set of data X. From this we can generate k bootstrap subsamples of data, X_k. If we use approximately 2/3 of the data for training a single decision tree (the in-bag data), then this leaves approximately 1/3 of the for testing (the out-of-bag sample). Using the set of training data subsamples, construct classifiers $h(\mathbf{x}, D_k)$, and let them vote on their individual in bag sample to form the bagged predictor. Each classifier is then grown using the CART method described in Section 1.2, however these trees are not pruned. Lastly, aggregate the votes of the classifiers over their individual out of bag samples. Then the out of bag estimate for the generalization error is the error rate of the out of bag classifier on the training set.

Variable importance is a useful way of examining of random forests because it ranks the relevance (importance) of the predictor variables to the response variable from most important to least important (i.e. how well the response variable can be explained using the predictor variables). Variable importance can be calculated by the random forest when the classifiers are trained. This is a useful tool for users who wish to potentially eliminate variables from their analysis, or to get a better understanding of which variables are playing a higher role in the random forest's performance. The measure of variable importance is determined by the decrease of the Gini impurity (where Gini impurity refers to the likelihood or probability that a new data point is incorrectly classified) when a variable is selected to split a node in the tree classifiers. For each variable, the total sum of the Gini impurity decreases across every tree in the random forest, which is then divided by the number of trees in the forest in order to give an average. The variables are then ranked based on these averages, where the higher the overall Gini decrease, the more important the variable. Another method for calculating variable importance is by randomly permuting the values (rows) of one particular variable whilst maintaining the row order of all other variables and observing the increase or decrease in accuracy. If the permutation does not affect the accuracy of the random forest, then this variable is determined to have low importance. Conversely, if the accuracy decreases with the shuffling of a variable, this means that the variable is important for the random forest.

1.3 Transmission Trees

A transmission tree is a directed graph that conveys the progression of a pathogen through a population under the assumption that an individual has a unique infector. It describes the transmission events between infected hosts (Ypma et al., 2013). Formally, a transmission tree is defined by a directed acyclic graph $T = (N, E)$, where each node $n_i \in N$ corresponds to an infected individual and each directed edge $e_i \in E$ corresponds to a transmission event. If there is a directed edge from node n_i to n_j, this means that n_j was infected by n_i. Since each infectee is limited to one infector, there must be a unique source node. Hence the graph has no cycles, and is a tree (Kendall et al., 2018).

As mentioned in Section 1.1, there is a small amount of literature on comparisons for transmission trees. In this work we use a metric (Kendall et al., 2018) designed for transmission trees by extending it to be applied to decision trees, which is made possible due to the similarities between transmission trees and decision trees. The most important characteristic in common is that both types of trees have node labels; in a transmission tree the nodes represent infectors and infectees in a population, but in decision trees the labels are the splitting variables and splitting values, or the final prediction of the tree.

1.4 Hierarchical Clustering and Dendrograms

A dendrogram is a visual representation of agglomerative (bottom-up) hierarchical clustering, known more formally as a binary merge tree (Nielsen, 2016). The process starts with n data points that are each in their own individual clusters, then uses a similarity measure (such as a Euclidean distance measure, as used in this work) to iteratively merge clusters until all of the points belong to a single cluster. Once a cluster contains more than one data point, in order to calculate the distance from a point outside of the cluster to the cluster itself, there are different types of measures used called linkages. In this work we use 'complete' linkage, which is the maximum distance between the point outside of the cluster and the points within the cluster. A dendrogram serves as a graph that illustrates the order in which the points are clustered. Grouping of points occur at different heights in the dendrogram, with lower heights indicating an earlier or previous grouping.

For example, suppose we have the following symmetric dissimilarity matrix for points A, B, C, D:

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline \text{A} & 0 & & & \\ \text{B} & 0.23 & 0 & & \\ \text{C} & 0.22 & 0.11 & 0 & \\ \text{D} & 0.37 & 0.20 & 0.15 & 0 \end{array}$$

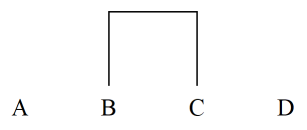


Figure 1.1: Left: the distance matrix between points A, B, C, D. Right: the first step in the hierarchical clustering process that groups together points B and C: these points are the closest together.

Notice the smallest value in the matrix is highlighted, and represents the distance between points B and C. The first merge of clusters in the dendrogram is points B and C and is shown in the first step in the dendrogram.

The next step is to adjust the dissimilarity matrix and consider B and C to be in the same cluster (and thus, occupying a single row of the new dissimilarity matrix). For example, for the entry that represents the distance between point A and the cluster $\{B, C\}$ can be calculated using complete linkage by

$$\max(dist(A, B), dist(A, C)) = \max(0.23, 0.22) = 0.23.$$

Therefore, the matrix and corresponding dendrogram is:

$$\begin{array}{cc} & \begin{array}{ccc} A & B, C & D \end{array} \\ \begin{array}{c} \text{A} \\ \text{B, C} \\ \text{D} \end{array} & \begin{bmatrix} 0 & & \\ 0.23 & 0 & \\ 0.37 & 0.20 & 0 \end{bmatrix} \end{array}$$

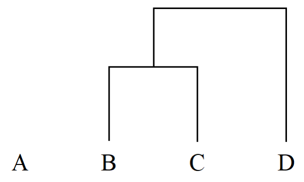


Figure 1.2: Left: the adjusted distance matrix after B, C are clustered together. Right: the second step in forming the dendrogram, grouping D with B, C

The smallest distance in the matrix is between point D and the cluster $\{B, C\}$, and so D joins the cluster with $\{B, C\}$ as shown in the dendrogram next to the matrix. Lastly, the only point not accounted for is A, so by default it is joined to the only other cluster in the dendrogram, $\{B, C, D\}$. Thus, the final dendrogram that represents the clustering of points A, B, C, D is given in Figure 1.3.

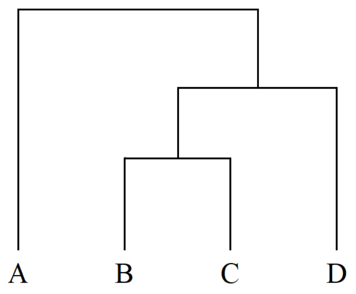


Figure 1.3: The final dendrogram that shows the clustering between points A, B, C, D.

1.5 Multi-Dimensional Scaling for Trees

The comparison functions that we develop in this work produce difference matrices which we intend to use to visualise the trees as points in a Euclidean 2-dimensional space. It is not inherently clear from a matrix of values what the proximities of the trees are to each other, and multi-dimensional scaling (MDS) will allow us to create a low dimensional representation of the distances between the trees. In other words, we can take a complex, multi-dimensional object (a tree) and "compress" it into fewer dimensions to be analysed visually. In this section, we explain how MDS works.

Given a map of points in Cartesian space, the Euclidean distance between points (x_i, y_i) is given by $d_{i,j} = \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2}$. Suppose we wish to obtain the coordinates (or a spatial representation) of points given an arbitrary distance matrix. This problem can be solved using multidimensional scaling (MDS) (Borg & Groenen, 2005). Here, by distance matrix on n items we refer to a $n \times n$ non-negative symmetric matrix in which each entry i, j specifies a non-negative distance between the items i and j. There are several types of MDS, in this work we use classical (metric) MDS which assumes that the distances between datapoints are Euclidean. This method of MDS was developed by Torgerson (1958), and it uses eigenvalue decomposition to find a coordinate matrix X. Specifically, classical MDS states that the coordinate matrix X can be derived from $B = XX'$, where the matrix B is calculated from the distance matrix D using the process of double centring. Here we will outline the steps of the algorithm in more detail:

1. Create the squared proximity matrix $D^{(2)} = [D^2_{i,j}]$
2. Apply double centring: $B = -\frac{1}{2}CD^{(2)}C$, using the centring matrix $C = I - \frac{1}{2}J_n$, where I is an $n \times n$ identity matrix, and J_n is an $n \times n$ matrix of all ones.
3. Choose a value for m such that m is the desired number of dimensions for the spatial representation.
4. Find the m largest eigenvalues $(\lambda_1, \lambda_2, ..., \lambda_m)$ and the corresponding eigenvectors $(e_1, e_2, ..., e_m)$ of B.
5. Lastly, the coordinate vector X is given by $X = E_m \Lambda_m^{\frac{1}{2}}$, where E_m is the matrix of the m eigenvectors and Λ_m is the diagonal matrix of the m corresponding eigenvalues from B mentioned in step 4.

The following example shows the final spatial representation for $m = 2$ for a given distance matrix between points A, B, C, D, E. The data for this matrix was generated using a uniform distribution, $x \sim U(0, 1)$.

$$\begin{array}{cc} \begin{array}{ccccc} A & B & C & D & E \end{array} & \\ \begin{bmatrix} 0.251 & 0.928 & 0.060 & 0.083 & 0.132 \\ 0.087 & 0.408 & 0.971 & 0.184 & 0.357 \\ 0.248 & 0.952 & 0.334 & 0.424 & 0.449 \\ 0.985 & 0.204 & 0.456 & 0.381 & 0.915 \\ 0.842 & 0.576 & 0.964 & 0.451 & 0.470 \end{bmatrix} & \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} \end{array}$$

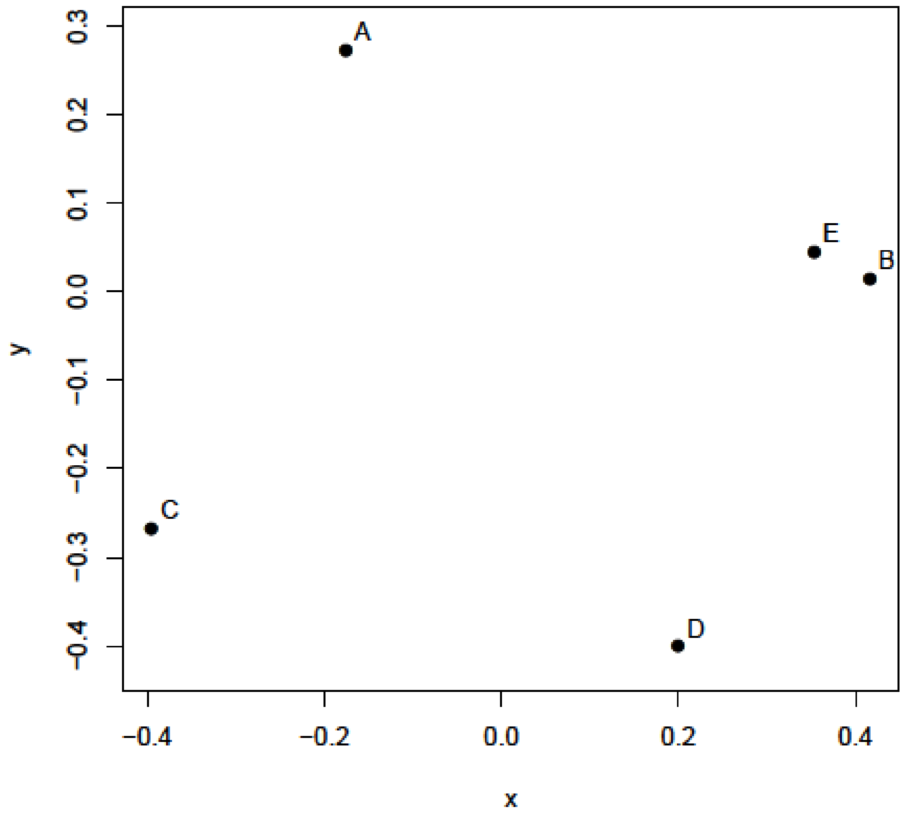


Figure 1.4: Spatial representation of points A, B, C, D, E using classic (metric) multidimensional scaling given their distance matrix (to three decimal places of precision).

Chapter 2

Methods

In order to evaluate our method of clustering decision trees, we conducted experiments on random forest trained on 5 datasets. These datasets are described in more detail in Chapter 3. In our experiments, first we attempted to quantify differences in the random forest trees using the various metrics that we developed. These comparison functions provide distance matrices for the decision trees. We then normalised all of the distance matrices for each dataset and summed them together, and used the combined distance matrix to cluster the trees into groups using dendrograms. Note that here we normalised the data to scale the values to be between $[0, 1]$ using the formula

$$\text{normalise}(x) = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}.$$

After establishing the distinct clusters, we developed a method for identifying the centre tree of each cluster using classic multidimensional scaling. With the centre trees identified, we conducted experiments to determine whether the accuracy of the centre trees increased with the number of clusters, how the accuracy of the centre trees correlates to the overall accuracy of the clusters, and how the centre tree accuracies compared with other statistical learning methods.

The overall goals of this approach is firstly, to successfully cluster the decision trees (we consider a clustering successful if it captures a signal in the dataset). Secondly, to find representative trees of each cluster and link them to the properties of the dataset. This approach presents two challenges, first to devise a method to find these representative trees, and second to demonstrate that they did indeed embody the random forest. To find the representative trees, we employed a hierarchical clustering method that groups the trees by how different they are from each other. To determine the differences between the trees in the random forest, we used an existing metric created by Colijn and Kendall (Kendall et al., 2018) as well as four other comparison functions we designed, all of which are described in more detail in this Chapter.

The result of each metric is an $n \times n$ matrix where n is the number of trees in the random forest. Each entry of the matrix i, j is then the difference calculated by the metric between trees i and j. Five metrics result in five matrices of differences, which we then add together to produce a matrix that represents the total differences between the trees. We then determine the optimal number of K clusters to group the trees (initially attempted by plotting the accuracy of representative trees as K increases, but overall is determined visually), where clusters are calculated in R using the complete linkage method. The next step is to determine a tree that can represent each cluster. Each cluster contains a group of trees, and their differences are extracted as a square matrix from the total differences matrix. We then perform classical multi-dimensional scaling on this matrix to determine the coordinates of each tree in 2-dimensional space. From this we calculate the mean of the x and y coordinates, and then determine the tree in the cluster that is closest to this mean. We call this tree the centre tree of the cluster.

2.1 Metrics for distances between trees

In this section, we describe the metrics that we use in our methodology.

2.2 The Kendall-Colijn Metric

The metric designed by Colijn and Kendall (Kendall et al., 2018) serves the purpose of quantifying the difference between two transmission trees. Given a node $n_i \in N$ of a transmission tree $T = (N, E)$, there exists a unique path, p_i, from the root node to n_i. The depth of this node is defined as the number of edges in the path p_i. The most recent common infector (MRCI) of two nodes n_i and n_j is the maximum depth of the node that lies on both paths p_i and p_j. The MRCI of the root node is assumed to have a depth of zero, and the depth of the MRCI of n_i and n_i is defined to be the depth of n_i. The concept of MRCI for the matrix $v(T)$ where the components $v_{i,j}$ are simply the depth of the MRCI of the nodes n_i and n_j, for all nodes in T. Suppose that there are two transmission trees $v(T_1)$ and $v(T_2)$. In order to compare the trees they are written as a vector for convenience and the difference is simply the Euclidean distance between the vectors.

An issue faced when comparing transmission trees is that they are dependent on sampled cases. Due to the fact that not everyone seeks medical assistance when infected with a pathogen, there can be many cases not known to the health care system. This loss of information could be detrimental to a transmission tree's ability to model the progression of a pathogen in a population. The Colijn-Kendall metric accounts for this by including a set of unsampled cases U (possibly of size zero) such that $N = S \cup U$, where S is then the set of sampled cases. However, now that a set of unsampled cases has been introduced, it is possible that the number of nodes differs between transmission trees which presents a problem with using the Euclidean distance since it relies on vectors having the same dimensionality.

The task now is to include both sampled and unsampled cases in the transmission tree so that cases that are not observed are still accounted for in the model, but guarantee that the set of sampled labels are equal and have the same magnitude across all of the transmission trees. This can be done with pruning, where attention is restricted to sampled cases,

$$v|_S(T) = (v_{s_1,s_1}, v_{s_1,s_2}, ..., v_{s_{|S|},s_{|S|}}).$$

Let $T = (N, E)$ be a transmission tree with both sampled and unsampled cases, and let $T^* = (N^*, E^*)$ be a copy of T where any unsampled cases in T without further infectees have been pruned. Specifically, if an unsampled node does not have an sampled cases in its citet path, then this node as well as its descendants are pruned. Then $v|_S(T) = v|_S(T^*)$.

The motivation for using this comparison tool is that it accounts for the entire structure of the trees. The other measures we use in this work, such as maximum depth, which are described later in this chapter do not capture the topology of the tree or the ordering of variables. Therefore, although this comparison tool is designed for phylogenetic trees and not decision trees it serves as an attempt to capture the full structure and topology. However, in order to apply the Kendall-Colijn comparison to decision trees, differences in these types of trees need to be taken into account. For example, the internal nodes in a decision tree are labelled with the splitting variable chosen at that node as well as the splitting value. The leaf nodes are assigned a prediction value. The internal nodes of phylogenetic trees represent the common ancestors of the descendants and are unlabelled. These differences imply that while it is possible to represent a decision tree in a phylogenetic-like tree format, it is not possible to reverse this process (i.e. a phylogenetic tree cannot be converted to a decision tree representation).

2.3 Node, Parent, Grandparent

In a transmission tree the labels of the nodes of the tree are unique (except for the unsampled cases) due to the condition that an individual can only be infected once and by only one person (meaning no labels are repeated within a tree). Additionally, multiple transmission trees that model the same pathogen through a given population will have the same set of sampled nodes across every tree. However, in a decision tree the node labels are the splitting variable chosen at that instance, accompanied by the splitting value associated with that variable. This raises the point that the nodes labels within a single decision tree will likely be unique, because while it is possible for a variable to be chosen multiple times within a tree, the chances of the split value being repeated is very low. Therefore, if one were to compare two of the decision trees, the set of the node labels from each tree would not be equal. In order to apply the Kendall-Colijn metric to a random forest tree we needed to devise a labeling system that made node labels less likely of being repeated within a tree but also have a large overlap of node labels when compared to another tree. The labelling system we propose is to drop the splitting value from the node label (retain the splitting variable), and concatenate the splitting variable at a particular node with the splitting variable of that of its parent and grandparent nodes (*node, parent, grandparent* format). This can be formally stated as:

$$\ell^{\text{new}}(\ell) = (\ell, p(\ell), p(p(\ell))) \tag{2.1}$$

Here ℓ^{new} is the new label that is applied to a node ℓ in a decision tree whose label is the splitting variable used at that node, $p(\ell)$ denotes parent node of ℓ, and lastly $p(p(\ell))$ is the label of the grandparent node. We will define $p(\ell) = 0$ when the parent of the node does not exist (i.e. the current node ℓ is either the root node or has a depth of 1). If $\ell = 0$ then this implies that ℓ is a leaf node and therefore is not assigned a splitting variable but rather a prediction value.

This concatenated label retains the splitting variable of the current node, as well as broadly representing the order of the decisions that went into the current location. However, it should be noted that this does not solve the uniqueness problem. It is still possible with this labelling system for sequences of splitting nodes to be repeated within a tree, and it does not guarantee that the set of node labels will be equal across two trees being compared. In order to restore the uniqueness characteristic of transmission trees, we will unlabel nodes that are repeated within a tree or do not occur in both trees that are being compared. We acknowledge that this is a significant limitation of using this comparison tool for decision trees as it will lead to removing relevant information. As mentioned later in section 2.4 we attempt to recover the lost information by developing a comparison of the symmetric set difference of labels within trees. Despite the limitations, overall this approach still serves as an excellent target for comparison between trees. If the parent and grandparent node match across two different trees, then the trees are operating and structured approximately the same way because the same set of variables is being cut in the same order (if the trees are similar, there will be many *node, parent, grandparent* matches).

The Kendall-Colijn comparison can be implemented using using the function *wiwTreeDist* which is part of the treespace package in R. This function finds the distance between transmission trees by comparing their MRCI depth matrices. These matrices can be obtained from another treespace function *findMRCIs* which takes as input a two column matrix where the first column lists the infector and the second column lists the infectees. This information can be obtained from a tree of type phylogenetic from the function *as_edgelist* from the igraph package. However, note that the tree must be of type phylogenetic, hence one cannot simply use this function on a decision tree extracted from a random forest. Hence, first we must use the output given in R that describes the decision tree to write the tree in Extended Newick format. The reason for this format is that it allows a tree written in this format to be read into R, and then we can use the function function *as.phylo* from the ape package which converts an object into a tree of class phylo (phylogenetic tree). An example of two decision trees and their corresponding extended Newick strings are in the following figure:

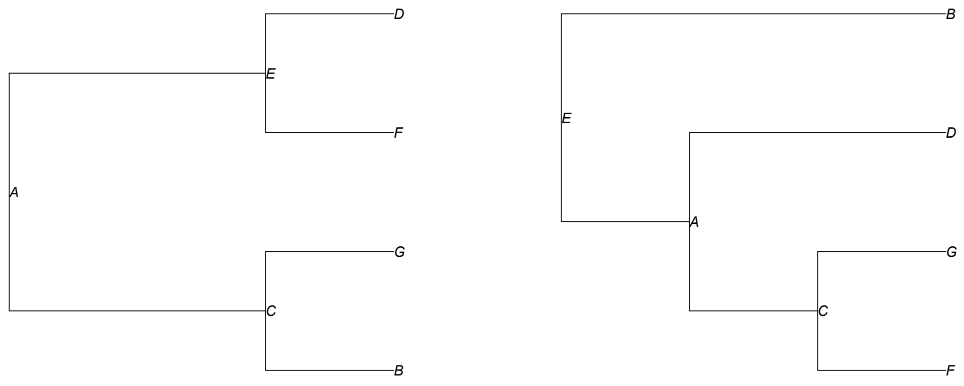


Figure 2.1: Extended Newick format strings for two simple labelled trees. Left: “((B, G)C, (F, D)E)A;”. Right: “(((F, G)C, D)A, B)E;”.

In the following two tables we display the output given by R that describes two decisions trees extracted from the same random forest. This particular forest was trained on the well-known Iris dataset.

Tree 97 from the Iris random forest						
Node	Left Daughter	Right Daughter	Split Variable	Split Point	Status	Prediction
1	2	3	3	2.35	1	0
2	0	0	0	0.00	-1	1
3	4	5	3	4.85	1	0
4	6	7	4	1.65	1	0
5	8	9	3	5.20	1	0
6	0	0	0	0.00	-1	2
7	0	0	0	0.00	-1	3
8	10	11	3	5.05	1	0
9	0	0	0	0.00	-1	3
10	0	0	0	0.00	-1	3
11	12	13	1	5.90	1	0
12	0	0	0	0.00	-1	3
13	14	15	4	1.55	1	0
14	0	0	0	0.00	-1	3
15	0	0	0	0.00	-1	2

Tree 83 from the Iris random forest						
Node	Left Daughter	Right Daughter	Split Variable	Split Point	Status	Prediction
1	2	3	3	2.70	1	0
2	0	0	0	0.00	-1	1
3	4	5	3	4.75	1	0
4	6	7	4	1.65	1	0
5	8	9	1	6.10	1	0
6	0	0	0	0.00	-1	2
7	0	0	0	0.00	-1	3
8	10	11	2	2.75	1	0
9	12	13	4	1.45	1	0
10	0	0	0	0.00	-1	2
11	0	0	0	0.00	-1	3
12	0	0	0	0.00	-1	2
13	0	0	0	0.00	-1	3

Table 2.1: Tables that show how R represents decision trees that have been extracted from a random forest. The two tables are trees 97 and 83, respectively, extracted from a random forest trained on the Iris dataset.

The important information we take from these tables are the columns *Node*, *Left Daughter*, *Right Daughter*, and *Split Variable*, as this information will allow us to construct the splitting variables of the current *node, parent and grandparent* labeling system previously described. Once we have converted the decision trees to phylogenetic-like tree formats in R, we get the two following phylogenetic trees.

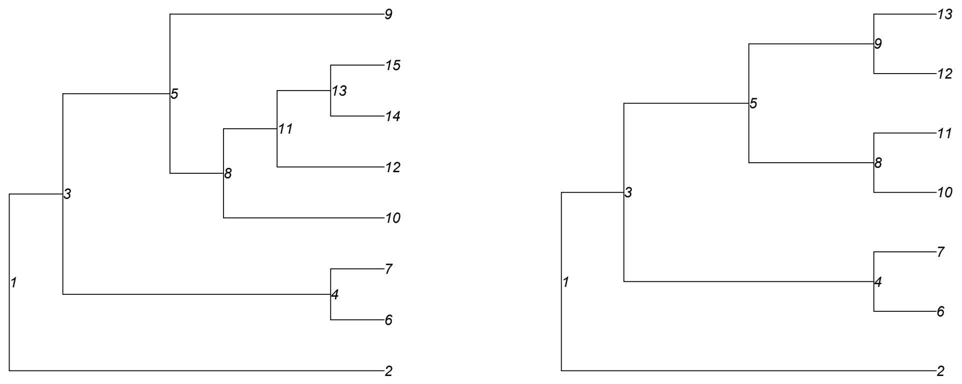


Figure 2.2: The figure on the left is a phylogenetic tree constructed from the 97th decision tree from the random forest. The figure on the right is the 83rd decision tree. The integers at the nodes refer to the node number assigned by the column "Node" in Table 2.1.

Next we must change the labels of each node so that they are in the *node, parent, grandparent* format by applying equation 2.1. As stated earlier, if a node does not have a parent or a grandparent node, we simply substitute 0. Therefore, the origin node would intuitively have the label '1, 0, 0' since it has neither a parent or a grandparent. In the figure above, we can see that the far right tip node is 9, and its parent is 5, and grandparent is 3; therefore, this tip label 9 will be replaced with the label '9, 5, 3'. If we extend this to all nodes in the trees shown in Figure 2.2, the result is the two trees shown in Figure 2.3.

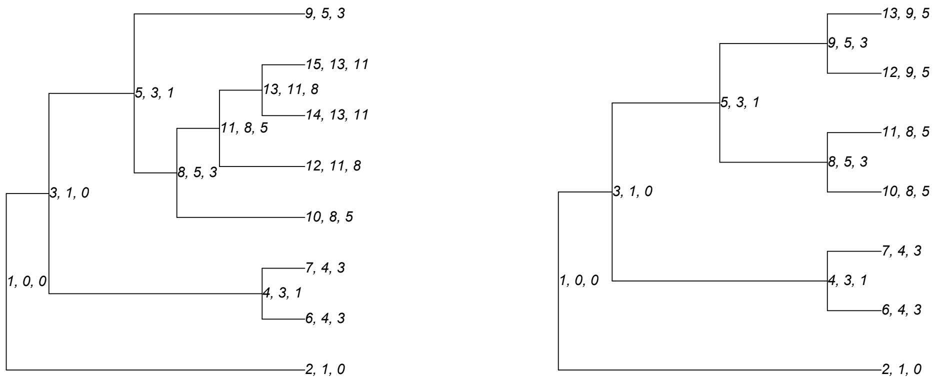


Figure 2.3: Decision trees 97 and 83 with nodes in the *node, parent, grandparent* format.

Next we will replace each node number with the splitting variable chosen at that node. We can see from the tables that the first node in both trees chose to split on variable 3. Therefore, every instance of node 1 in both trees in Figure 2.3 must be replaced with a 3. For node 2, in both trees this was a tip node, so no splitting variable was chosen, meaning every instance of 2 must be replaced with a 0. If we continue this idea for all of the splitting nodes and tip nodes, the result is the two trees shown in Figure 2.4.

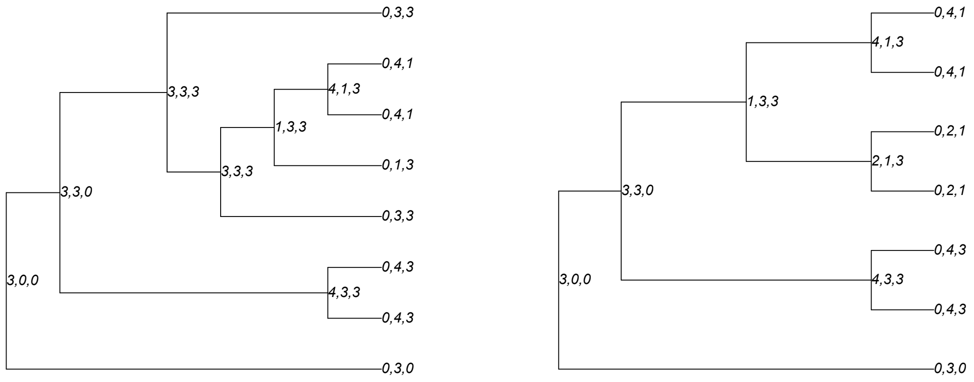


Figure 2.4: Decision trees 97 and 83, where all node numbers have been replaced with the splitting variable chosen at that node, or 0 if that node was a tip.

Lastly, as discussed previously, by definition all nodes in a transmission tree must be unique because each case has a unique infector, and the set of sampled cases does not change.

Therefore, we must remove labels that occur more than once from both trees, as well as remove nodes that are not in the intersection. Removed nodes will be replaced with ‘u’ and will serve as the random set of unsampled cases. If we examine the tree on the left in Figure 2.4, we notice that the label ‘4, 3, 3’ occurs more than once, resulting in all instances of the label ‘4, 3, 3’ being replaced with a ‘u’ in both trees. Notice that in the tree on the right there is the label ‘0, 2, 1’, and although it does occur more than once in this tree, this label is not present in the tree on the left so it must be removed for multiplicity and not being a member of the intersection of node labels. Once all of the appropriate nodes have been replaced with ‘u’ in both trees, we may apply the Colijn-Kendall comparison for these two transmission trees.

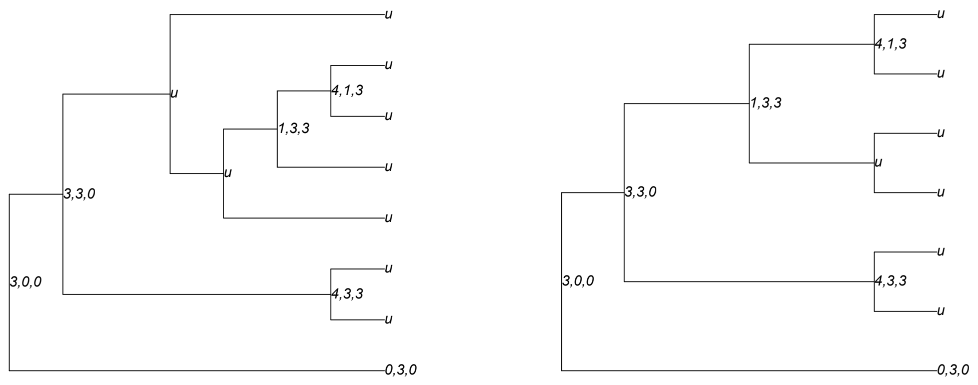


Figure 2.5: Decisions trees 97 and 83 where nodes have been replaced with the label ‘u’ if they have multiplicity or do not exist in the intersection.

2.4 Symmetric Set Difference

By reformatting the node labels of the random forest decision trees in order to apply the Kendall-Colijn comparison, if one decision tree is significantly different from another then the majority (if not all) of the labels will be replaced as unsampled cases. In this situation, a large portion of information about the trees is lost and the Kendall-Colijn metric is less useful. This motivates the use of a comparison that incorporates the symmetric set difference of two decision trees that will recapture the lost information due to node labels being replaced as unsampled cases. Suppose there are two sets A and B, then the symmetric set difference of A and B are all of the elements from each set that do not appear in the intersection $A \cap B$. In this case, the sets A and B are the labels of all of the nodes in T_i and T_j respectively, in the *node, parent, grandparent* format. We can formally state this as:

$$d(T_i, T_j) = |\ell^{\text{new}}(N_i) \triangle \ell^{\text{new}}(N_j)| \tag{2.2}$$

where N_i is the set of all nodes in the tree with their respective labels ℓ, T_i, and similarly N_j is the set of all nodes in the tree, T_j, with their own labels. Lastly, $\triangle$ denotes the symmetric set difference.

Therefore, the symmetric set difference of T_i and T_j is the magnitude of the set of nodes that are not common to both trees (which which consequently are the nodes that become unlabelled due to the Kendall-Colijn comparison). Note that if the value of the symmetric set difference is zero, this does not imply that T_i and T_j are the same decision tree. There could still be differences in the topology of the trees such as depth, balance, or number of nodes. A value of zero for this comparison function means that the set of unique labels for T_i and T_j are equal.

2.5 Maximum Depth

The depth of a node n_i within a tree is the number of edges in the unique path p_i from the root to n_i. Therefore, the maximum depth of a tree is the number of edges in a longest path that exists within the tree. Hence a comparison that quantifies the difference between two decision trees using maximum depths is simply the absolute value of the difference between the maximum depth of each tree. This can be formally written as:

$$d(T_i, T_j) = |\max(\phi(T_i)) - \max(\phi(T_j))| \tag{2.3}$$

where $\phi(T)$ is the vector of the depths of the nodes $n_l \in T, l = 1, \ldots, N$. Here, N is the total number of nodes within the tree.

2.6 Minimum Variable Depth

Previously noted in Section 1.2.3, at each node of a decision tree a random subsample of the variables from the training dataset are eligible candidates to be chosen as the splitting variable. Therefore, it is possible for a variable to be chosen at several nodes throughout the tree, or only once, or not at all. Therefore, for each decision tree we can define a vector with entries given by the minimum depth of each variable within that tree. This is written formally as:

$$d(T_i, T_i) = \|\phi(T_i) - \phi(T_j)\| \tag{2.4}$$

where $\phi(T_i)$ is the vector, $\langle \min(\text{depths}(v_1)), \min(\text{depths}(v_2)), \ldots, \min(\text{depths}(v_p)) \rangle$, which contains minimum depth of each variable in tree T_i. In this equation, $\text{depths}(v_m)$ returns a list of all of the depths of the occurrences of node v_m, $m \in 1, ..., p$, within tree T_i. Here, p

denotes the number of variables in the dataset used to train the random forest.

If a variable, v_m, is not chosen as a splitting variable for any of the nodes in the tree, T, then depths$(v_m) = \emptyset$. In this case we can define:

$$\min(\text{depths}(v_m)) = max(\phi(T)) + 1 \tag{2.5}$$

where, recall from equation 2.3, max$(\phi(T)$ is the maximum depth of the tree T.

2.7 Occurrence Count of Variables per Tree

Another comparison function that utilises the chosen splitting variables within a decision tree is one that counts the number of times a variable is chosen as a splitting variable. For each tree in the random forest this will result in a vector whose length is the number of variables in the training dataset, where each entry is the count. This comparison function can be defined as:

$$d(T_i, T_i) = \|\gamma(T_i) - \gamma(T_j)\| \tag{2.6}$$

where $\gamma(T_i)$ returns a vector, $\langle \text{count}(v_1), \text{count}(v_2), \ldots, \text{count}(v_p)\rangle$, which contains the number of times each variable occurs in tree T_i. Here, p denotes the number of variables in the dataset used to train the random forest. If a variable, $v_m, m \in 1, \ldots, p$ is not chosen as a splitting variable for any of the nodes in the tree, T, then intuitively we can assign this case the value 0.

2.8 Visualizing Tree Comparisons with Heatmaps

Each random forest trained on the datasets consists of N trees and we wish to quantify the similarities or differences in structure of these trees. Earlier in this chapter we described the comparison tools that we designed and implemented. Each comparison results in an $N \times N$ matrix where each entry i, j is the difference between trees i and j according to each criteria. With the five tools that we applied, the result is five difference matrices. Since each entry i, j corresponds to trees T_i and T_j in every matrix, we can simply add all of the matrices together to form a combined dissimilarity matrix. To account for the differences in scale of each individual matrix, before summing them together we normalized the matrices so that each row of the matrix has a mean of zero and a standard deviation of 1, with the range of values being between $(0, 1)$. We can then visualise this combined matrix using a heatmap.

Chapter 3

Experiments and Results

The datasets we used for experiments were the Iris dataset (Anderson, 1936), a dataset on drug use (Fehrman et al., 2017), tuberculosis (Xu et al., 2020), and lastly a house pricing dataset (De Cock, 2011); each dataset is described below in more detail. For each dataset, we applied a 75% split so that 75% of the data was used for training the random forest, and the remaining 25% was used for testing. The random forests that we trained each had $N = 100$ trees, and the number of variables sampled at each node was $\sqrt{p}$ for classification and $p/3$ for regression, where p is the number of variables in the dataset. For the iris, tuberculosis, drug use and synthetic datasets the random forests were used for classification, and the housing random forest was trained for regression. In this chapter, first we describe the datasets in more detail followed by the experiments we conducted to achieve the goals outlined in Chapter 2.

3.1 Datasets

3.1.1 Iris dataset

The Iris dataset was first created by Edgar Anderson (Anderson, 1936) when he collected data on three different species of iris (setosa, versicolar, and virginica) with the intention of quantifying the morphologic variation of iris flowers. For each species, he collected fifty samples where for each flower he measured sepal length, sepal width, petal length, and petal width. The dataset became more widely known when British biologist and statistician Ronald Fisher used the Iris dataset as an example for his linear discriminant analysis (Fisher, 1936). Since then it has become a well known simple test dataset for statistical classification methods in machine learning where the task is the classify the species of Iris given the recorded attributes of the flower.

3.1.2 Drug Use Dataset

The creation of this dataset was motivated by the problem of evaluating an individual's risk of drug consumption and misuse (Fehrman et al., 2017), since it stands as a serious issue globally. Over an eighteen month period in 2011 and 2012, Elaine Fehrman conducted an anonymous web survey for people over the age of 18, with the goal of using the data to predict the risk of drug consumption for each individual. The first component of the data is demographic information, including country of location, age range, ethnicity, and level of education. The second section of the questionnaire gathered personality traits using the Revised NEO-Five Factor Inventory (McCrae & Costa Jr, 2004) and Barratt Impulsiveness Scale (Stanford et al., 2009) which use likert style questions to gauge the individual's personality and impulsivity, and the Impulsiveness Seeking Scale (Zuckerman, 1994) which consists of true or false questions to determine the level of sensation seeking. Lastly, drug use was assessed by asking questions regarding the subject's use of 18 legal and illegal drugs, as well as a fictitious drug to distinguish over-claimers. The web survey gathered 2051 responses, and the final dataset consisted of 1885 observations after cleaning.

For this work we use a version of this dataset that was taken from the UCI machine learning repository (Fehrman et al., 2015) and further processed by Dr. Caroline Colijn for teaching purposes. First, the substance use numbers were adjusted from factors to integers because of how they are incremental. These numbers are the observation value for any drug and range from 0 to 6 with assigned meaning (0: never used, 1: used over a decade ago, 2: used in the last decade, 3,4,5,6 used in the last year, month, week, day, respectively). The total substance use is summed and stored as the variable "severity". Next a column is added with the name "any" which holds a true or false value and indicates whether the individual reported the use of any of the illegal substances. Since they are not illegal, caffeine, nicotine, and alcohol are excluded from this indicator. Lastly, the response column "UseLevel" is created with the two possible values "high" or "low". An individual is assigned the classification "high" if their total substance use level (severity) was above a certain threshold (in this case 14), and "low" if severity is below it.

3.1.3 Tuberculosis Dataset

In 1995 in London, UK, there was an outbreak of the large isoniazid-resistant tuberculosis (Ruddy et al., 2004). Outbreaks such as this provide researchers with key opportunities to study the transmission of this prominent and deadly disease. In 2020, Xu et al. aimed to combine whole genome sequencing (WGS) data with mathematical modelling techniques to reconstruct transmission histories of this outbreak, as well as use patient-level data to identify important variables that contribute towards transmission (Xu et al., 2020). The patient-level data was collected from patient surveys, questionnaires, interviews, and medical records. The data itself consists of predictors such as sex, age, indicators if the patient was born in UK, and history of homelessness and drug use, among others. These predictors are then associated with the categorical response variable of whether or not the patient is infecting tuberculosis. Once the data was constructed, Xu et al. additionally cleaned the data in various ways: for categorical variables with two levels (i.e., ‘yes’ and ‘no’), if the variable was missing at least 40% of the data then missing entries were replaced with ‘unknown’. For other variables the R package Mice was used for multivariate imputation. The decision to impute large amounts of missing data was motivated by the preference for using all data available, accepting that the result may be far from the truth. This patient-level data was of interest to us for this work because it is a well constructed dataset that is both relevant to phylogenetic transmission analysis and machine learning classification tasks. The heatmap and associated dendrogram presented later in this work are the result of applying our combined comparison functions to this dataset.

3.1.4 Housing Dataset

This dataset is designed to use regression to predict the price of houses in Ames, Iowa, given attributes about the properties. Dean De Cock obtained the dataset from the Ames City Assessor’s office which contained 113 variables that described 3970 property sales that had occurred in Ames, Iowa from 2006 and 2010. For teaching purposes, De Cock modified the dataset by removing all variables that required specific domain knowledge or background calculations. The final dataset contained 80 variables that were directly related to the property sales, most of them relating to questions a typical home buyer would ask.

Of the 80 variables, 20 of them are continuous and relate to various area dimensions, such as lot size. There are 14 discrete variables that count features inside the house, such as bedrooms or bathrooms. For the 46 categorical variables, 23 are nominal (variable that is used to name, label or categorize particular attributes that are being measured) and 23 are ordinal (variable that takes values with an order or rank). The values of the categorical variables range from 2 to 28 classes. Lastly, the dependent variable for this dataset is ‘SalePrice’.

3.1.5 Synthetic Dataset

Decision Tree as a Nested List

One of the questions in this research is whether or not a random forest can recover decision trees if they are used to create the data, as well as will there be quantifiable similarities between the grown trees and the original synthetic ones. In order to generate data from a random decision tree, first we must generate a random decision tree. To do this we create a representation of a decision tree in the form of a nested list. The list is initiated with four entries which represent the root node of the tree; the first entry randomly selects an integer between 1 and the number of variables there will be in the dataset. The second entry in the list is a number sampled from a uniform distribution and represents the splitting value of the node. The last two entries represent the left daughter node and the right daughter node, respectively. The third and fourth entry could either be a terminal (leaf) node, or it could be a new node that will itself have two daughter nodes and continue the tree. It is randomly decided (similarly to a coin flip) whether the daughter node is going to be a new node or a leaf node, unless the tree has already reached the set maximum depth. If the daughter node is a leaf node, it is then randomly assigned a value of either true or false, since this dataset will be used for classification. If the daughter node is not a leaf node and will continue the tree, then this entry in the list becomes a new list with the same format stated. Therefore, by recursively generating this tree, the representation becomes a nested list. Here we provide an example of a decision tree shows the splitting variable chosen at each node, as well as the splitting value that determines whether an observation will progress to the left daughter node or the right. This is a decision tree trained for classification where the leaf nodes will determine whether the final decision is true or false.

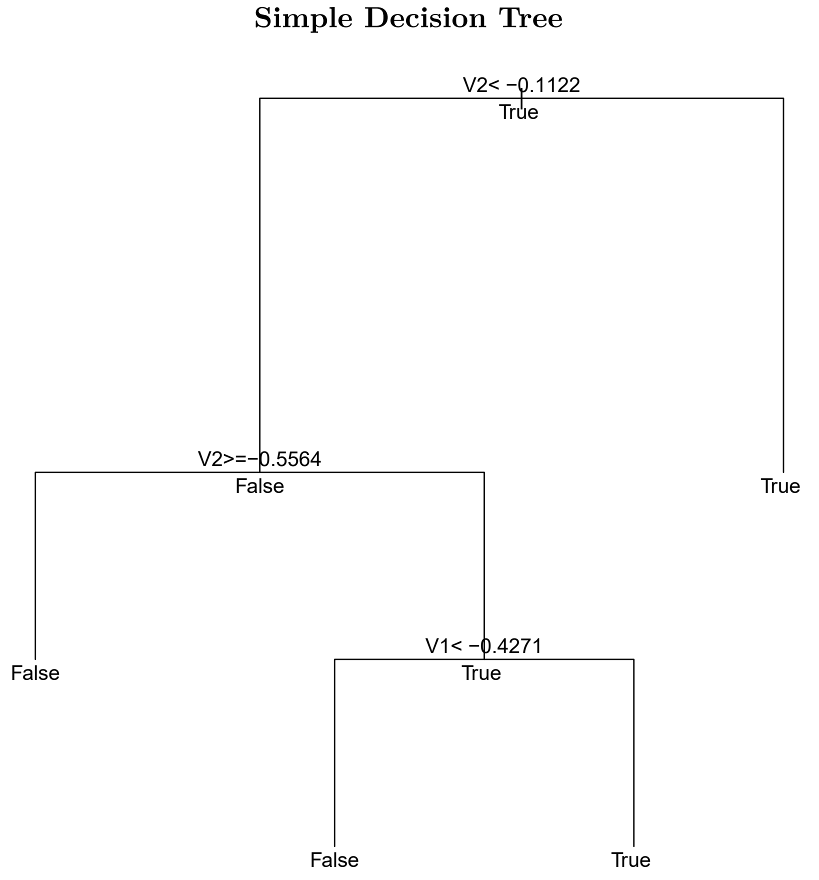


Figure 3.1: Example decision tree that is generated from a nested list.

For this particular decision tree, its nested list representation is as follows

$$[2, -0.1122, [2, -0.5564, \text{False}, [1, -0.4271, \text{False}, \text{True}]], \text{True}]$$

Generating Data from a Decision Tree

With the above method to create random decision trees, we can now use these trees to generate synthetic data. We will create a dataset consisting of 1500 observations, in which 500 observations are generated from each of three separate random decision trees, generated by the above algorithm. The data will have 10 variables, and the random decision trees used to generate the data will each have a set maximum depth of 10. In order to create an observation, we must first start with an empty vector that is the length of the number of variables plus an extra entry for the target value. Starting at the root, and continuing for all other nodes that are not leaf nodes, we must decide if the observation is going to travel to the left daughter node or the right daughter node. To do this we take a random sample from a uniform distribution $x \sim U(-1, 1)$. If $x < 0.5$ then the observation travels to the left, whereas if $x \geq 0.5$ then the observation travels to the right. This also allows us to assign a value to the variable that the parent node is split on. If the observation descends to the left, then the variable value assigned will be less than the value of the splitting node, calculated by $split\,value - |a|$ where $a \sim U(-1, 1)$. If the observation descends to the right then the variable value will be greater than the split value such that $split\,value + |b|$ where $b \sim U(-1, 1)$. The observation then continues through the tree in this manner, assigning the relevant variable values at each parent node. If a variable is used more than once within the same tree in succession and the observation encounters it multiple times, then the value assigned will be replaced at each encounter, with the final value being from the last time it passes through that variable. Note that this means we are not exactly simulating from the decision tree. Lastly, the target value is assigned when the observation reaches a leaf node, and the value is simply the decision randomly assigned to that node. In our case it will either be True or False.

Note that not all variable values will be assigned by progressing through the decision tree. For example, a variable may not be chosen as a splitting variable when the decision tree is created. Therefore, in this case this variable value will be empty for all observations created by this decision tree. Another case is if the random path that an observation takes through the decision tree may not encounter a variable. In these examples, there will be missing values of variables in observations. Therefore, for the missing values we assign a random value $x \sim U(-1, 1)$. This becomes a useful feature since it introduces noise into the data, making it more comparable to real data. However, this prompts an interesting problem for future work: if one were to create neater data without these challenges, would we then be able to recover the underlying decision trees?

3.2 Experiments and Results

3.2.1 Heatmaps on Combined Comparison Functions

In the beginning of Chapter 2, we mention that one of our goals was to identify clusters of trees using the combined difference matrix. Visually we can inspect the clusters by applying a heatmap to the differences matrix, however the entries of the matrix will have been rearranged according to the dendrogram shown on the left and top borders of the heatmap. The dendrogram clusters the trees into relevant groupings using the method described in section 1.4. Note that the dendrograms on the borders are the same since the matrix is symmetric. We applied the heatmap to each of the combined matrices for each dataset and we show them below. Additionally, notice that there are integers on the opposite axes to the dendrograms; these integers denote the tree numbers and the positions show how the trees have been rearranged by the dendrogram.

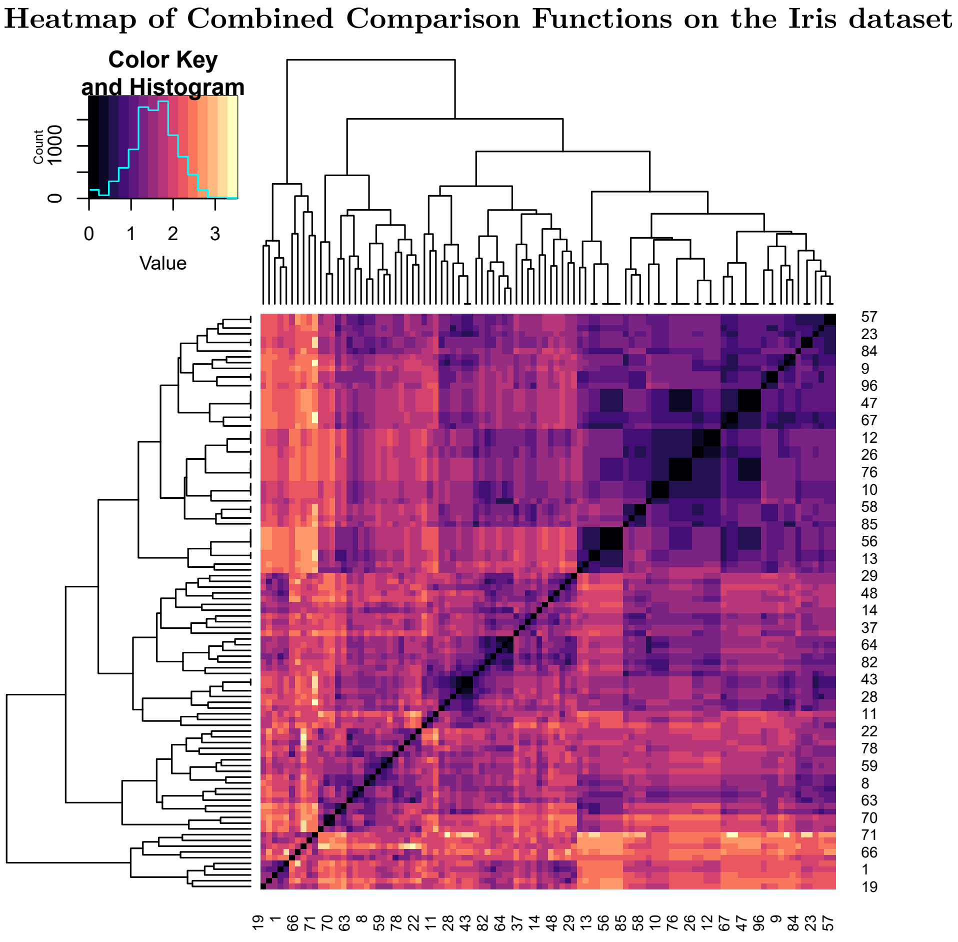


Figure 3.2: Heatmap of all comparison functions added together from the Iris dataset, with the dendrogram clustering tree differences together. For every heatmap, the integers along the right and bottom axes correspond to the tree numbers, meaning there is one tree per row and column of the heatmap. Notice there is a distinct block pattern within the heatmap, which implies clustered structures that of the trees.

For the Iris heatmap, it appears that there are numerous small clusters. Compared to the other heatmaps from the other datasets, the colour variation here tends to be on the darker side. This is supported by the colour key and histogram, that shows that the highest count of values tends to be lower. These low values indicate that the quantifiable differences between the Iris random forest decision trees is not substantial. This could be due to the simplicity of the dataset, meaning there might not be variation needed in the trees to to model the dataset. The next heatmap is for the drug use dataset.

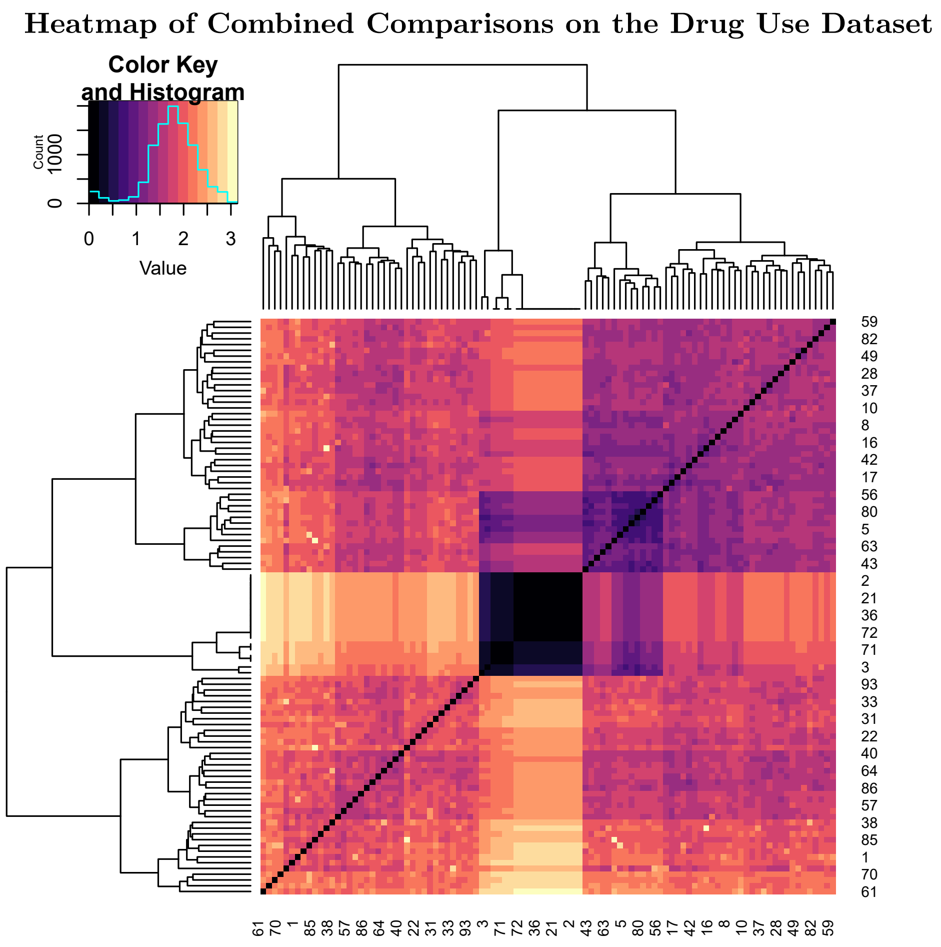


Figure 3.3: Heatmap generated by combining the heatmaps from each comparison applied to the random forest trees trained on the drug use data.

The contrast in this heatmap compared to the Iris heatmap is quite noticeable. This heatmap and associated dendrogram has clearly defined clusters, and they are bigger and fewer. The colour is mostly lighter, indicating that the difference in trees is more significant according to the comparisons. The more distinct clusters are also indicated in the dendrogram, since the majority of the clustering occurs at a low height, whereas in the Iris dendrogram clustering occurs at several heights.

A black pixel in the heatmap corresponds to a zero value in the combined matrix, implying that the two trees, T_i and T_j, are equal according to the comparisons we implemented. This is trivial across the diagonal where $i = j$, however when $i \neq j$ a value of zero is an interesting occurrence because it means two different trees within the random forest are the same. For example, note the black square in the centre of the heatmap in Figure 3.3. An off-diagonal value of zero occurs on row 44 and column 43, which corresponds to trees 99 and 71, respectively. If we plot these trees we can see that they are separate instances of the same tree:

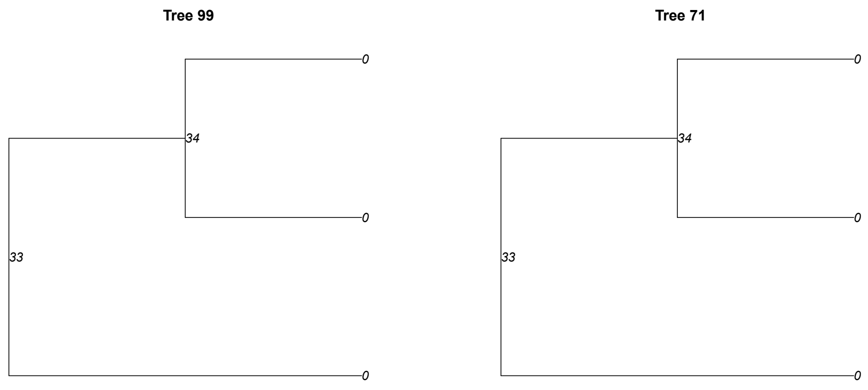


Figure 3.4: Trees 99 and 71 extracted from the random forest trained on the drug use dataset.

This is due to the possibility that a random forest can grow multiple copies of the same tree. However, this does not mean that all of the trees that contribute towards the black square in the centre are the same tree.

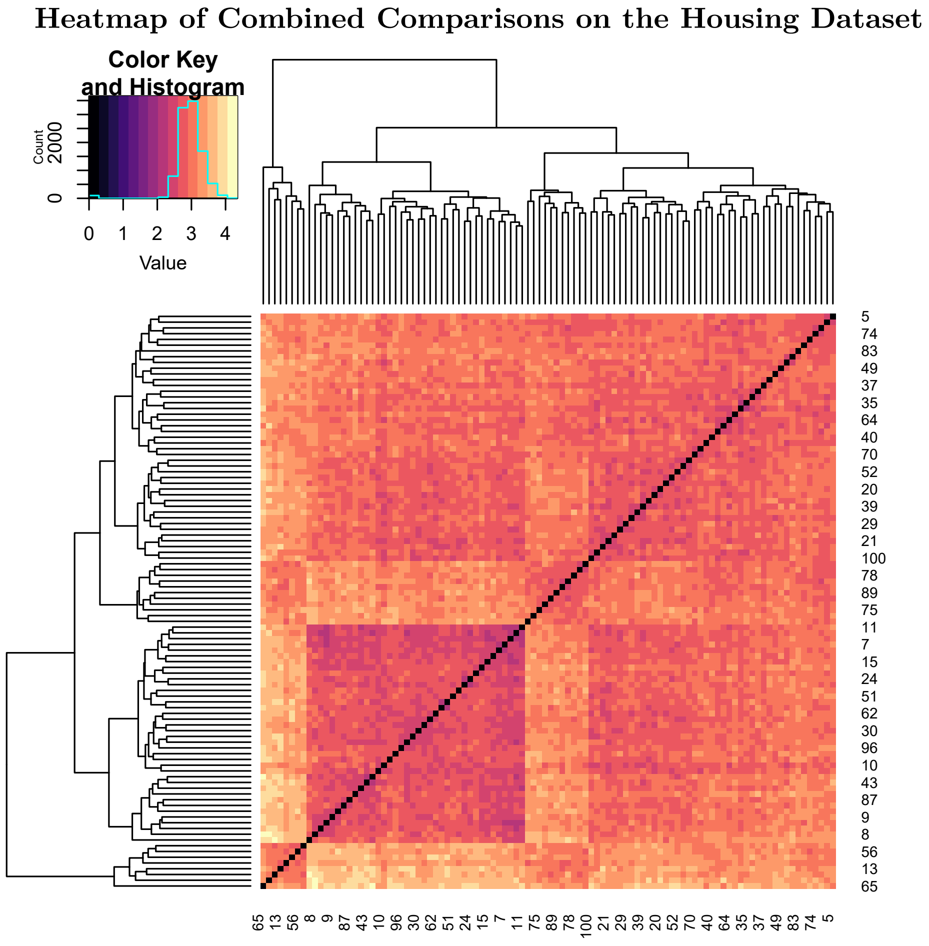


Figure 3.5: Heatmap generated by combining the heatmaps from each comparison applied to the random forest trees trained on the housing data.

For the previous heatmaps, we have noticed that there are black pixels off of the diagonal indicating that there are repeated trees within the random forest. This heatmap however only has zero values on the diagonal, meaning all of the trees are unique. The housing dataset described in section 3.1.4 is quite complicated in that there are 80 variables which are a mixture of numerical and categorical, so it stands to reason that the decision trees that make up the random forest are quantifiably different to model the complexities in the data.

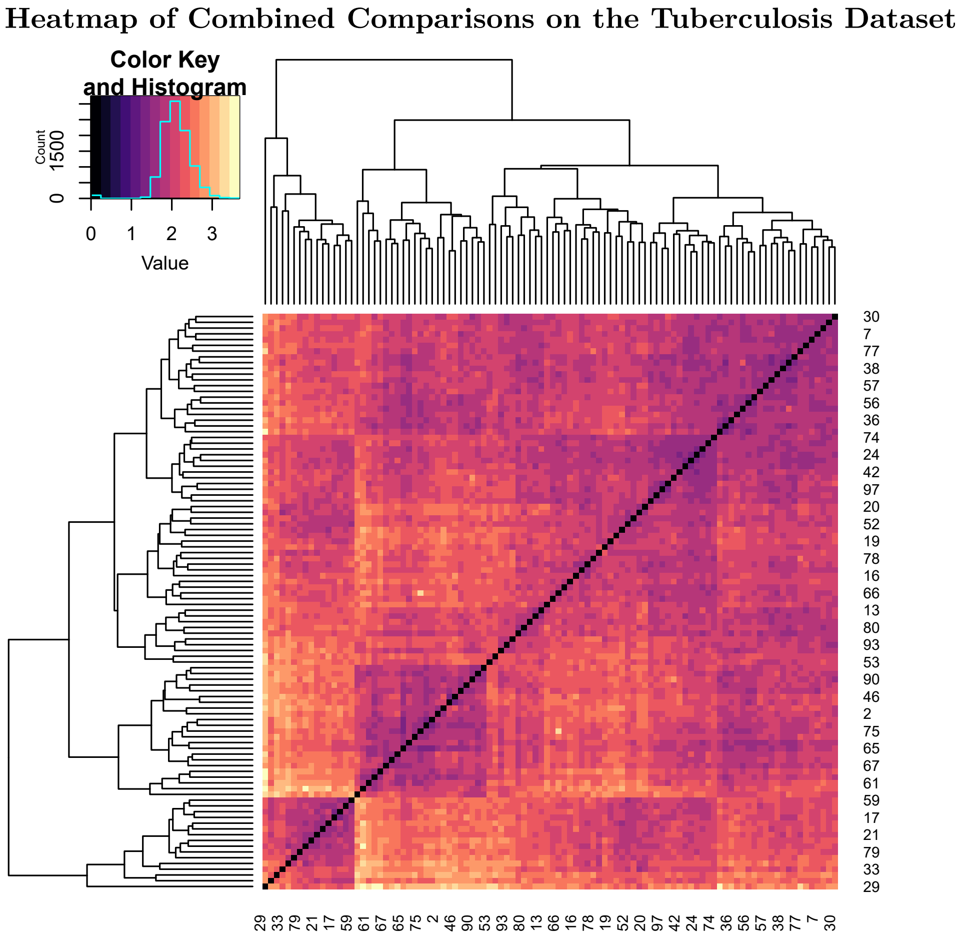


Figure 3.6: Heatmap generated by combining the heatmaps from each comparison applied to the random forest trees trained on the tuberculosis data.

Similarly to the housing heatmap, the tuberculosis heatmap does not have any off diagonal zero values. The grouping of the trees also occurs mostly at a lower height, similar to drug use and housing.

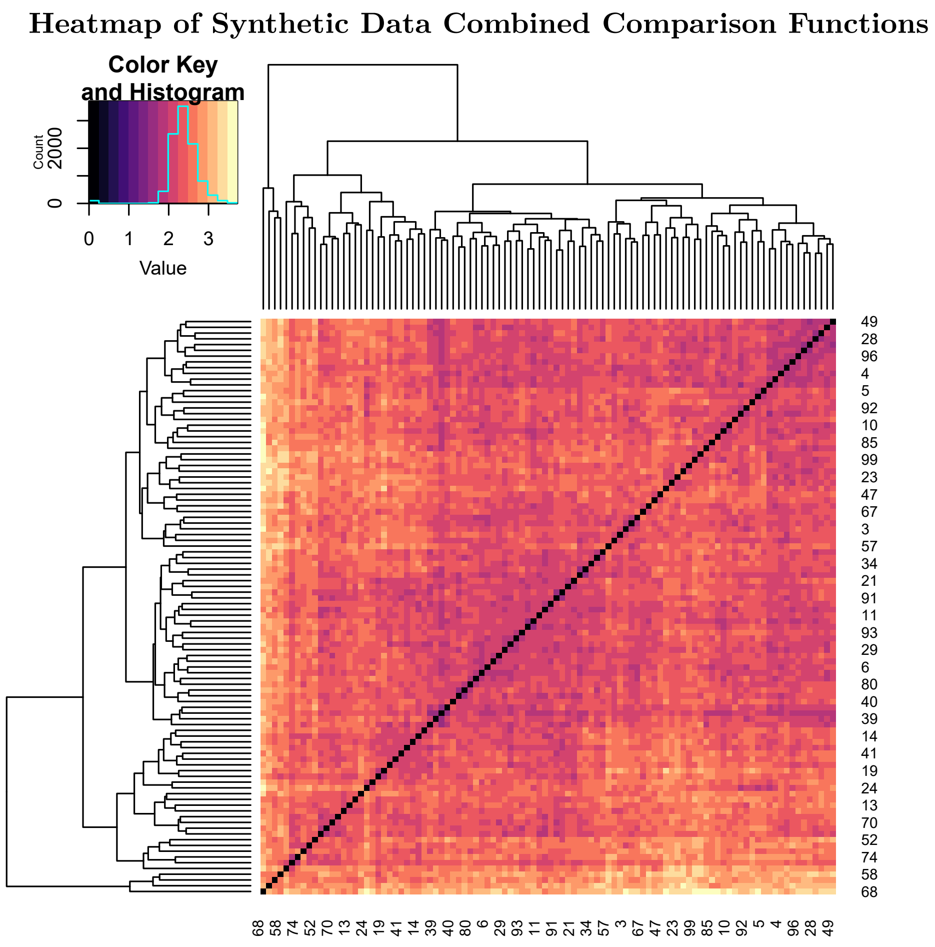


Figure 3.7: Heatmap generated by combining the heatmaps from each comparison function applied to the random forest trees trained on the synthetic data.

The heatmap for the synthetic data is the most intriguing since there is the least amount of structure out of all of the heatmaps. Since the data was generated from three unique decision trees, we expect that the trees grown in the random forest trained on this data would share characteristics of the original trees, if not fully recover them. Since we used three original decision trees, intuitively there would be at least three distinct clusters visible in the heatmap. However, there do not appear to be any discernable clusters: there is no visually apparent clustering. Due to this observation, we examine if there is any statistically significant difference between the clusters. If so, which comparison functions are contributing towards these differences (this is outlined in section 3.2.4.)

3.2.2 Accuracies of Clusters

Recall from section 3.2.1 that there are dendrograms on the side and top axes of the heatmaps. Initially these dendrograms were built to iteratively group the trees based on a dissimilarity matrix, but they can also be used to determine which trees belong to each cluster, given the desired number of clusters K. For example, if we take a closer look at the dendrogram from the tuberculosis combined heatmap in Figure 3.6, and wish to assign the trees to $K = 3$ clusters, the following dendrogram highlights how this can be achieved.

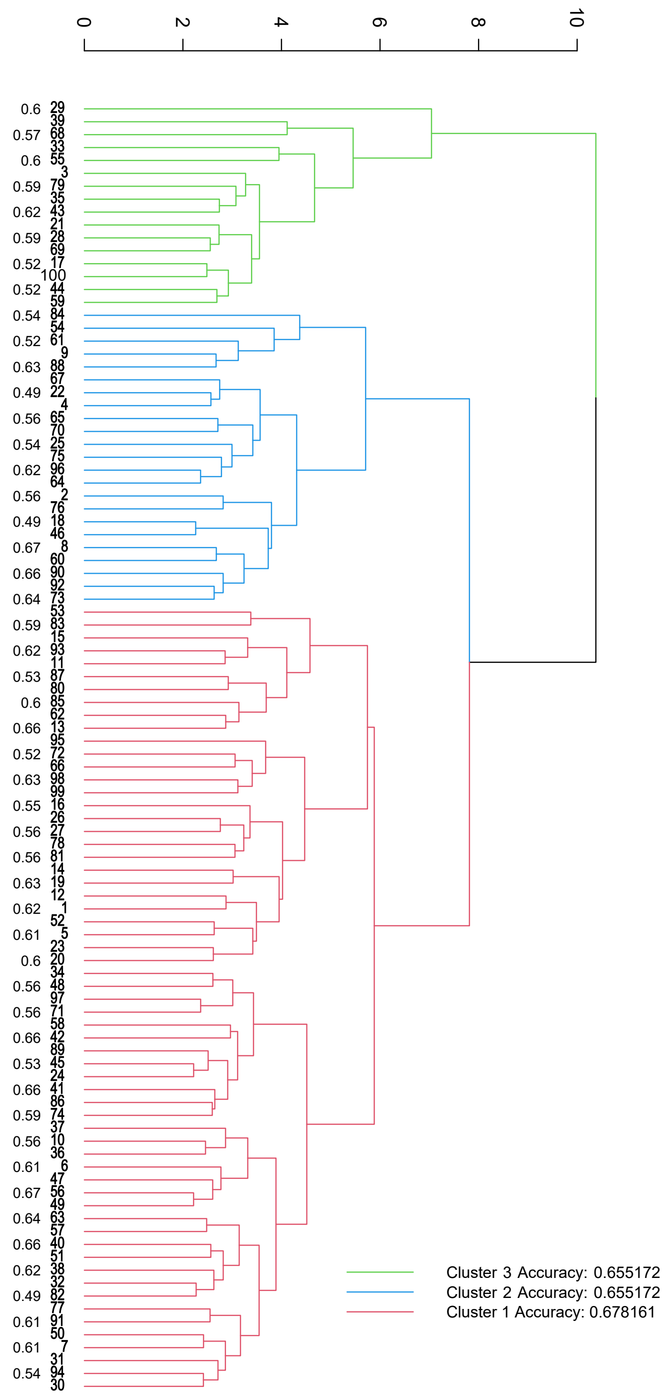
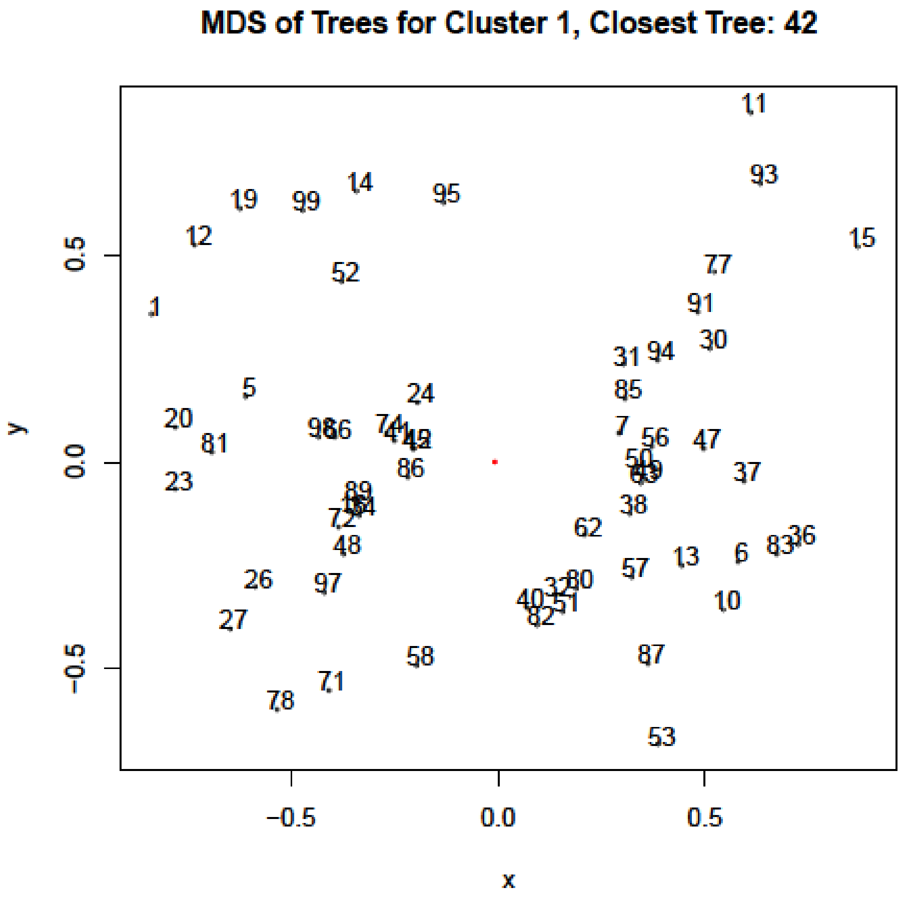
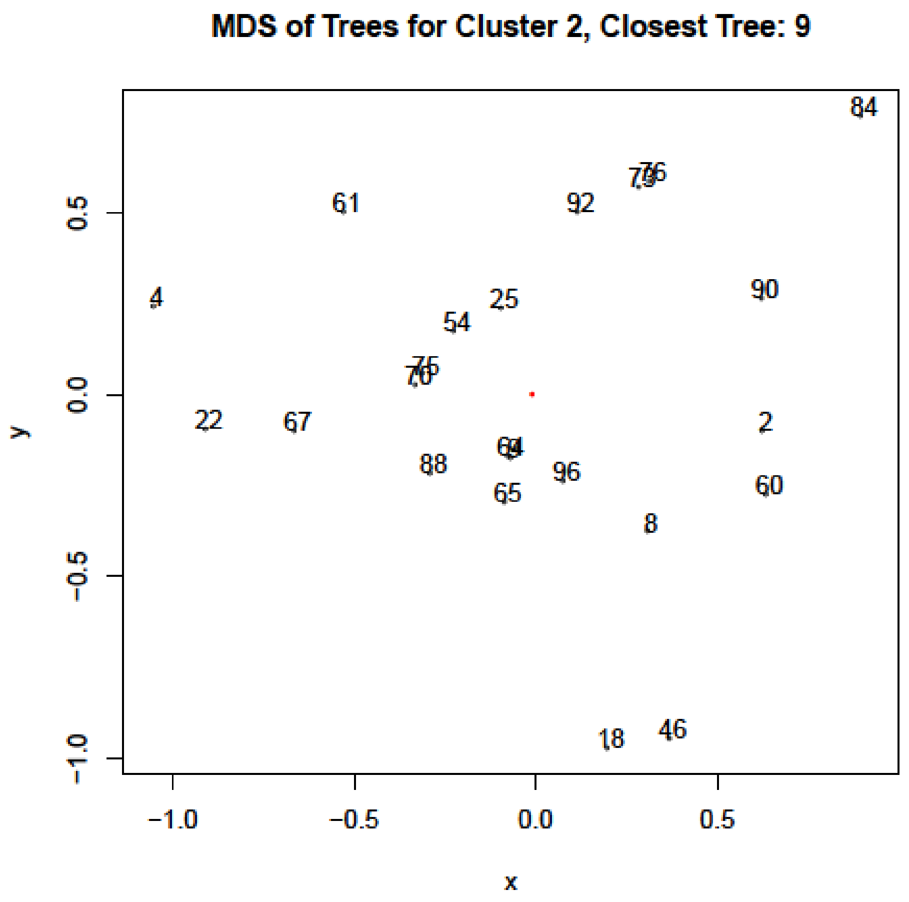


Figure 3.8: Dendrogram extracted from the combined heatmap from the tuberculosis data. The dendrogram has been colour coded to indicate which trees belong to each cluster as assigned by the dendrogram.

Equipped with a method for clustering trees by their dissimilarities, we theorised that perhaps these clusters could be summarised or represented by a singular tree closest to the centre of a cluster. Furthermore, if we can define a centre tree for every cluster, then the entire random forest could be explained by K centre trees. We propose using MDS as a method for finding the centre trees for each cluster. As mentioned in section 1.5, MDS gives a geometrical visualisation of points in a space using a matrix of dissimilarities between the points. We have identified which trees belong to which cluster, and we can extract the values of dissimilarities that correspond to these trees from the combined comparison matrix. We can then use MDS to visualise these trees and their proximities to each other in a 2-dimensional plot. This will result in 3 spatial plots, one for each cluster, shown below.





MDS of Trees for Cluster 2, Closest Tree: 9
84
76
79
61
92
4
90
25
54
75
70
22
67
2
64
96
88
60
65
8
18
46
y
x
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5

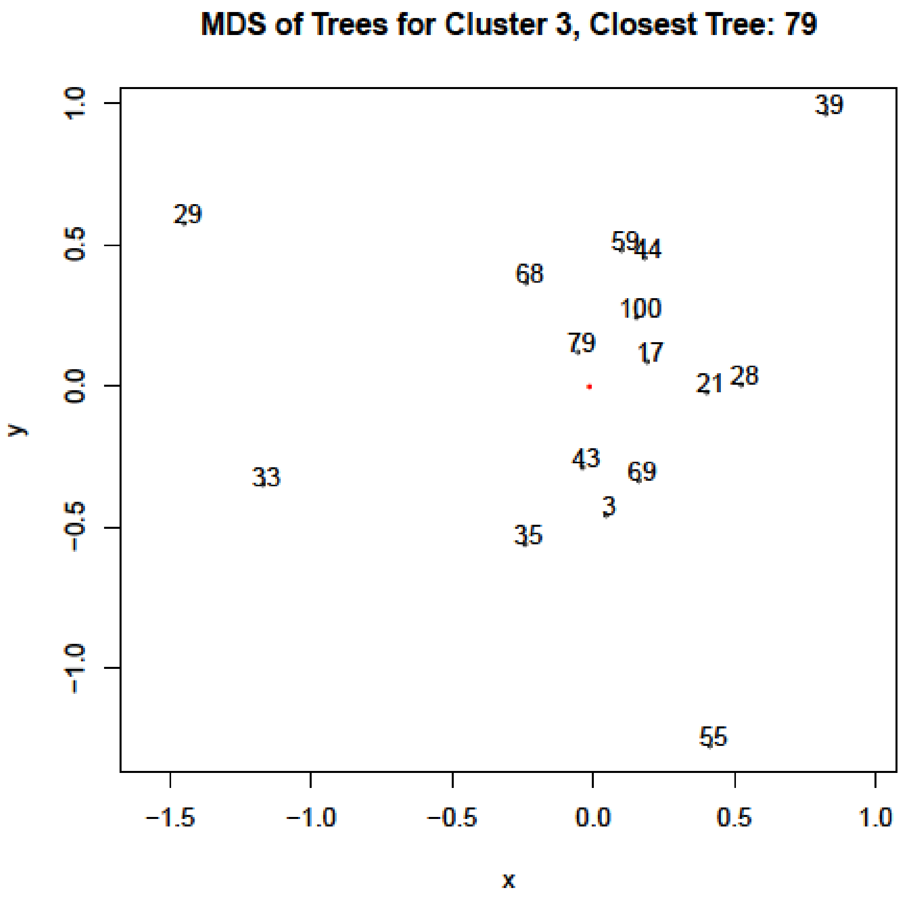


Figure 3.9: Each plot shows the geometric visualisation of the trees in their respective clusters, used to identify the centre tree and the one closest to the mean point. The centre trees for clusters 1, 2, and 3 are 42, 9, and 79, respectively.

Within the spatial plots of the trees, the red point represents the mean of the of the points, and the centre tree is calculated by determining which tree is closest to the mean. For each cluster 1, 2, and 3, the centre trees are 43, 9, and 79, respectively.

The process of assigning the trees to K clusters and finding a centre tree for each cluster motivates several questions regarding the accuracy of the random forest: Do these centre trees indeed represent the entire random forest? How do the accuracies of these centre trees compare to an equal number of random trees extracted from the forest? Lastly, do the trees in each cluster have comparative accuracies, and is there a correlation with the overall accuracy of the cluster? In turn can we discern distinction between the accuracies of the trees within a cluster against the remaining clusters?

Boxplots of Accuracies of Methods

As a first step in the analysis of representative centre trees, we compared the performance of $K = 3$ centre trees against an equal number of random trees extracted from the random forest, as well as the random forest itself, a single decision tree separately trained on the data, and logistic regression. For training and testing purposes, we split the data into 75% for training and 25% for testing, and in order to get a robust result we conducted the experiment over 100 different splits of the data for every dataset. However we kept the number of clusters fixed a $K = 3$ across all splits (further analysis into determining if there is an optimal K is described later in this chapter).

Previously we have outlined the process so far of training a random forest on a dataset, implementing the comparisons for the individual trees to form a dissimilarity matrix, and then finally clustering the trees to determine the centre trees and comparing them against other randomly selected trees and a few statistical methods. Repeating this process with 100 splits of data for every dataset (in order to reduce test set variance) is computationally expensive. Therefore, we used the super computer Compute Canada to perform these tasks; specifically, the general cluster Narval. The experiments on Compute Canada were conducted on Intel i7-8565U 1.8GHz CPU and 16GB RAM with R version 4.1.2 and 2020 standard environment. After completing all of the experiments we noted that the total CPU usage was 0.82 CPU years, meaning that if we ran this experiment on a single CPU of the same power of the clusters, it would have taken 10 months to complete. (As we could submit the restarts in parallel, the walltime, or total amount of time taken was lower: less than one month.)

The following are the box plots of the accuracies for each method across all of the splits for the Iris dataset.

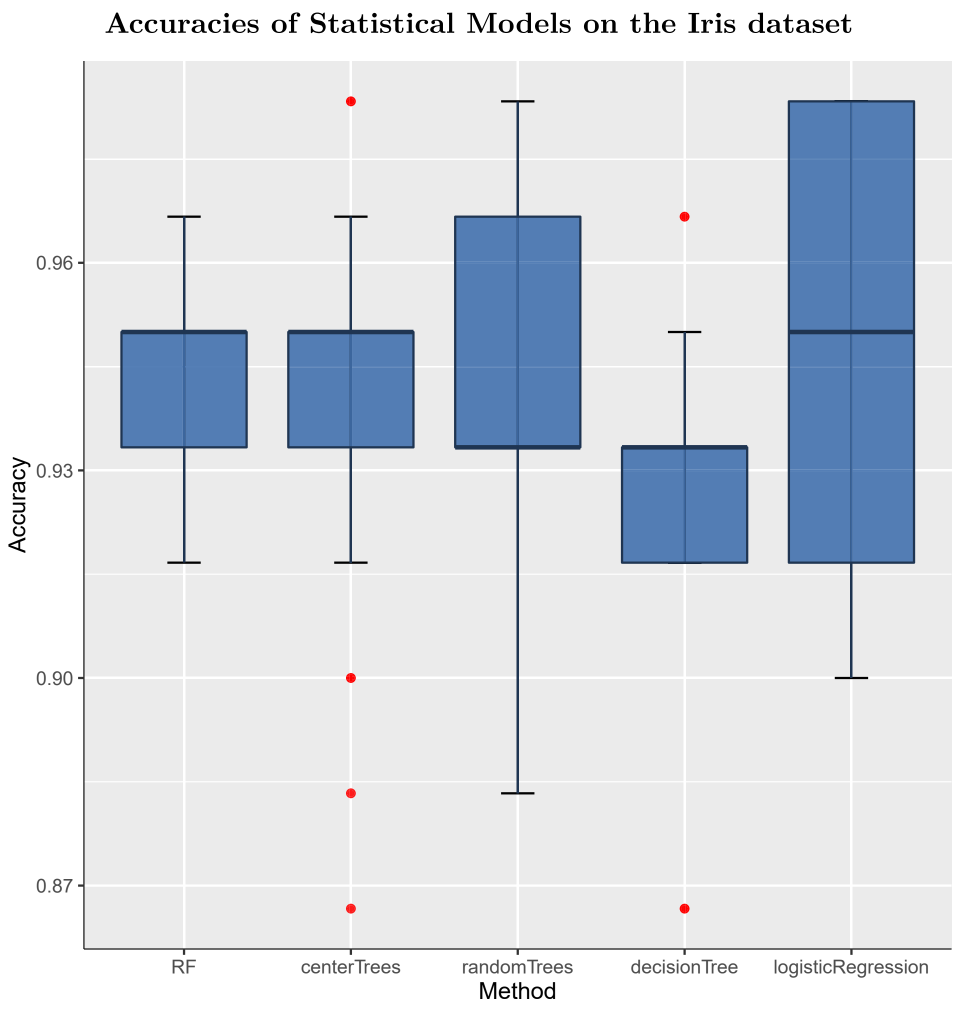


Figure 3.10: Each boxplot is the range of accuracies of each method on the test set across 100 splits of the Iris data with $K = 3$.

For the Iris dataset we see that the performance of the RF and centre trees are comparable. The random trees have more variable accuracy, with better results than the RF in some splits. The poorest accuracy overall comes from the single decision tree, and the best is logistic regression but it still highly variable.

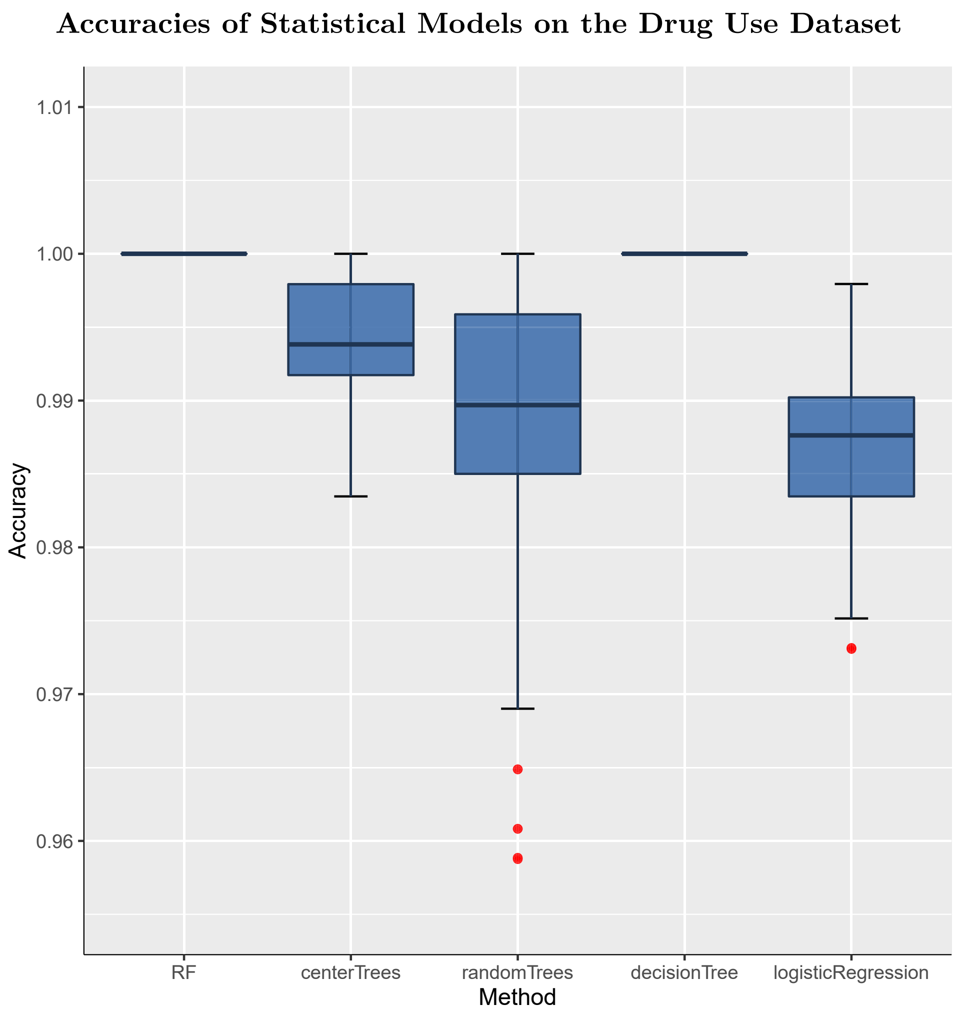


Figure 3.11: Boxplots showing the range of accuracies of each method on the test set across 100 splits of the Drug Use data with $K = 3$.

For the drug use dataset we can see that the best performing methods were the RF and the single decision tree. Interestingly, in this dataset the performance of the centre trees is noticeably better than the randomly selected trees. However, the accuracy of the methods on the test set for drug use are high. This is due to the strong correlation between the variables and the response (as mentioned in section 3.1.2).

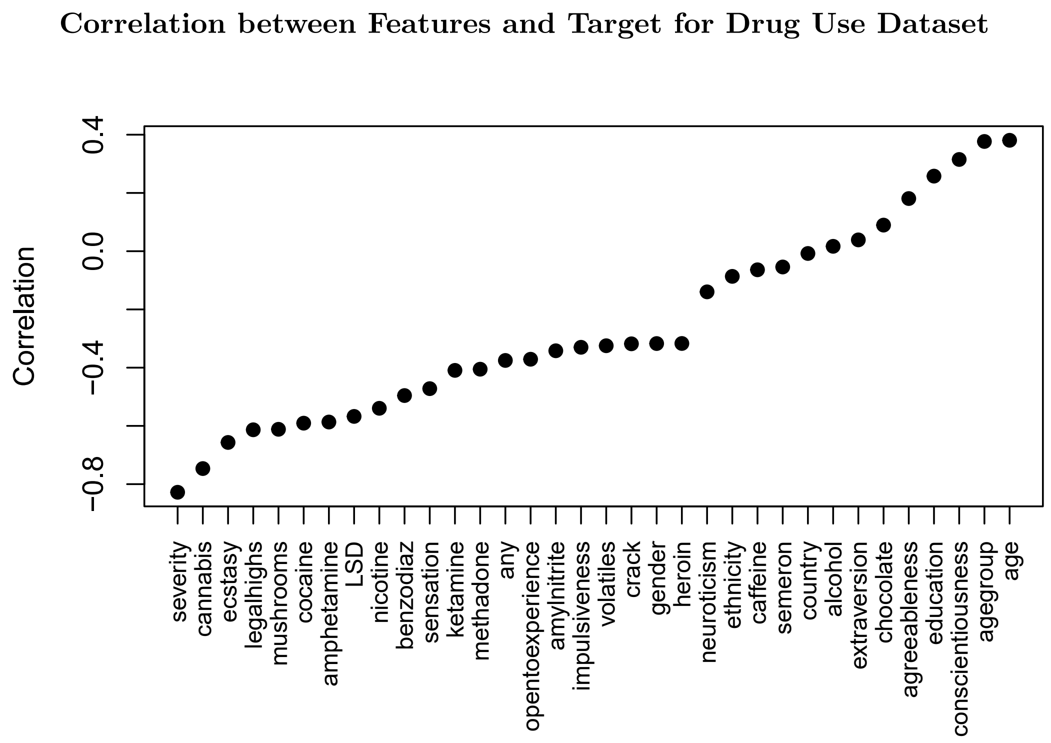


Figure 3.12: Correlations between each input variable and the target value ‘UseLevel’. Note that the highest correlation is with severity.

We can see from Figure 3.12 that none of the variables have a direct correlation with the target ‘UseLevel’. However, the feature severity does have a notably high correlation.

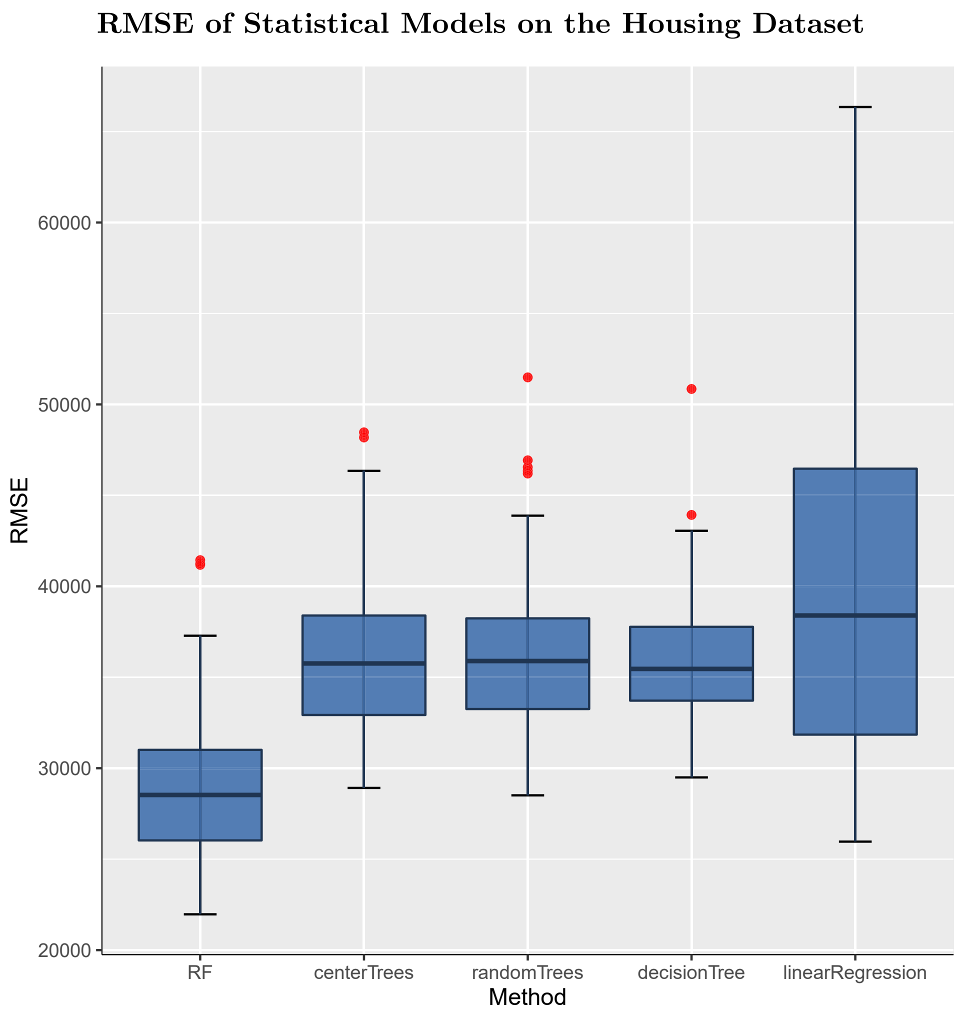


Figure 3.13: Boxplots showing the range of accuracies of each method on the test set across 100 splits of the Housing data with $K = 3$.

Note that for the housing dataset the methods applied were for regression, whereas all other datasets required classification. Hence instead of using logistic regression as one of the standard statistical models we used linear regression. Overall the random forest has the best performance, with the random and centre trees having a comparable performance to each other as well as the single decision tree. The poorest performance for this data was linear regression.

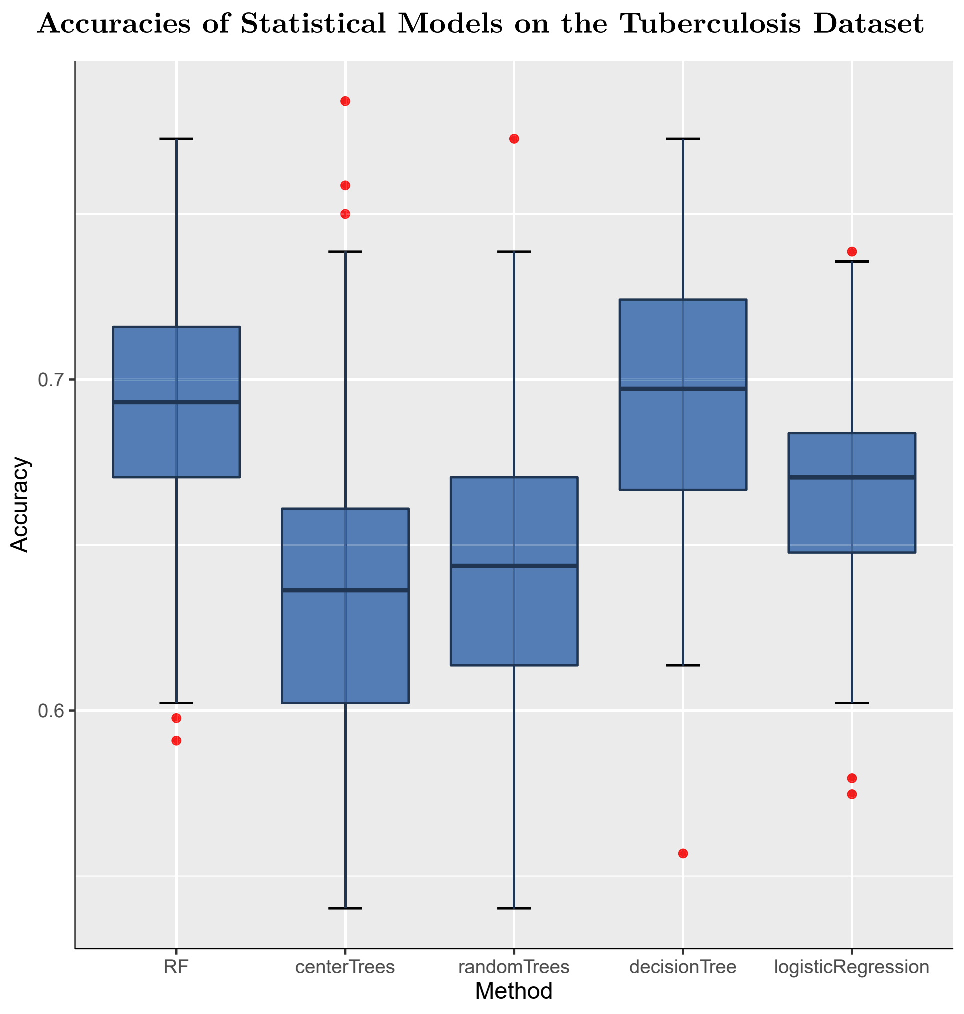


Figure 3.14: Boxplots showing the range of accuracies of each method on the test set across 100 splits of the Tuberculosis data with $K = 3$.

Notice that the centre trees and the random trees have comparable performance but also have a noticeably worse performance than all of the other standard methods. Perhaps this is because with $K = 3$ there are not enough trees to model the data sufficiently. We theorised that if we were to increase K we would observe the accuracy of the centre trees increasing, approaching the accuracy of the random forest, while the accuracy of the random trees would increase at a slower rate.

3.2.3 Further Experiments on TB Data

Tubeculosis Random Forest with $N = 20$ Trees

Initially, we wanted to more closely examine the structures of the trees that had been separated into individual clusters that can be observed in Figure 3.6 with $N = 100$. To simplify this task we reduce the number of trees by training a new random forest on the tuberculosis dataset with $N = 20$. The resulting heatmap is shown in Figure 3.6:

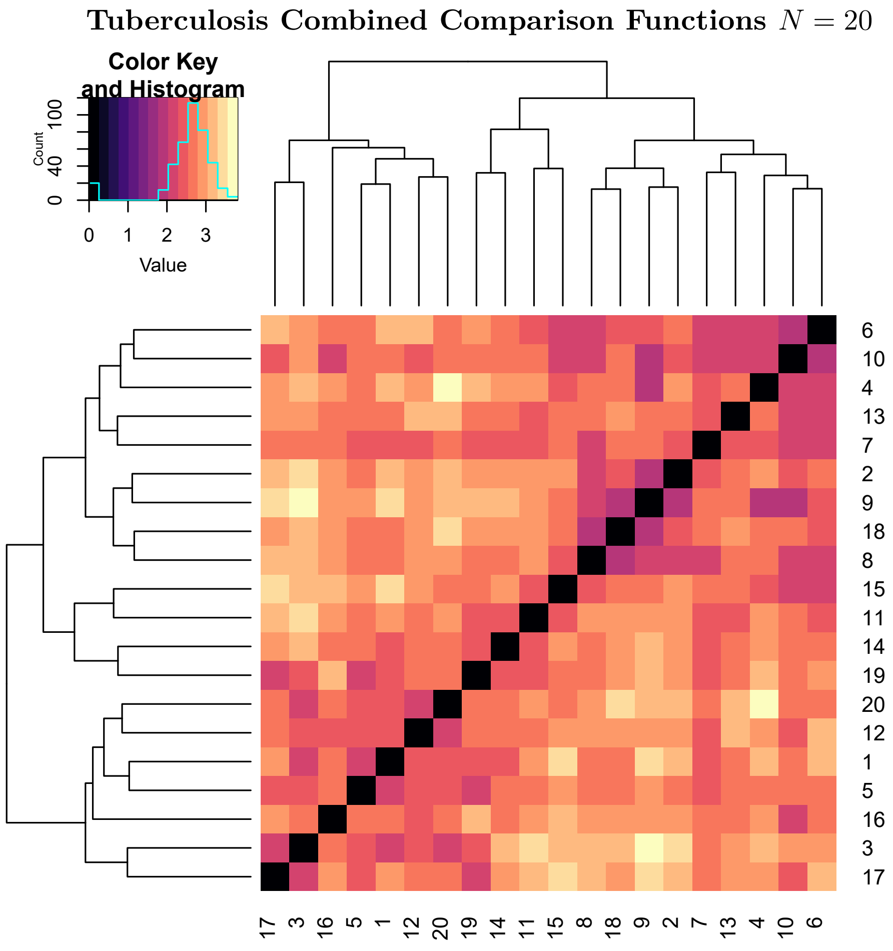


Figure 3.15: Heatmap of combined comparison functions on a random forest trained on the tuberculosis dataset with $N = 20$ trees. The seed for the split in the data is 1.

From this heatmap there is evidence of 3 or 4 clustered structures. The trees that comprise this random forest and the clusters they are assigned to with $K = 3$ can be found in Section A1. The structure of the trees in their individual clusters varies in several visual aspects. For example, the number of trees assigned to each cluster sets clusters 1 and 2 apart from 3 since they have seven and nine trees respectively, whereas cluster 3 only has four. The depths of the trees also varies across clusters. The trees in cluster 2 have a smaller maximum depth than the trees in clusters 1 and 3. The average depth of trees in cluster 2 is approximately twelve, whereas the average depths in cluster 1 and 3 are approximately fifteen and fourteen. Taking balance into consideration, in cluster 3 trees 11, 14, and 15 appear to be almost equally balanced, with only tree 19 slightly off balance. However, the trees in clusters 1 and 2 are all noticeably unbalanced except for tree 12 in cluster 1. Lastly, if we look at the first variable the trees split on we can see that the most common first split variable in cluster 2 is variable 8, where this occurs in 4 out of the 9 trees. Clusters 1 and 3 do not have a high count for any first split variable. For cluster 1, two out of 9 trees first split on variable 2, but all others are unique. One thing to note about the first split variables in cluster 3 is that three of the trees first split on 13, 1, ad 6, and these variables are not used as first split variables in any of the trees in the other two clusters. Overall, while the differences in the trees across the three clusters are not immense, there is clearly evidence of subtle distinctions between them that are captured by these clusters.

Accuracy of Centre Trees with Increasing K

Returning to the random forest trained on the tuberculosis dataset with 100 trees, we proposed using MDS to find the centre tree for each cluster and plot the predictive accuracies of the centre trees as K increases. This was done with the intent of finding an optimal number of clusters, i.e. when increasing K does not yield a higher accuracy within some tolerance. In order to ensure that the centre trees serve as the best representatives of the random forest we compared their accuracies against the performance of K random trees extracted from the forest.

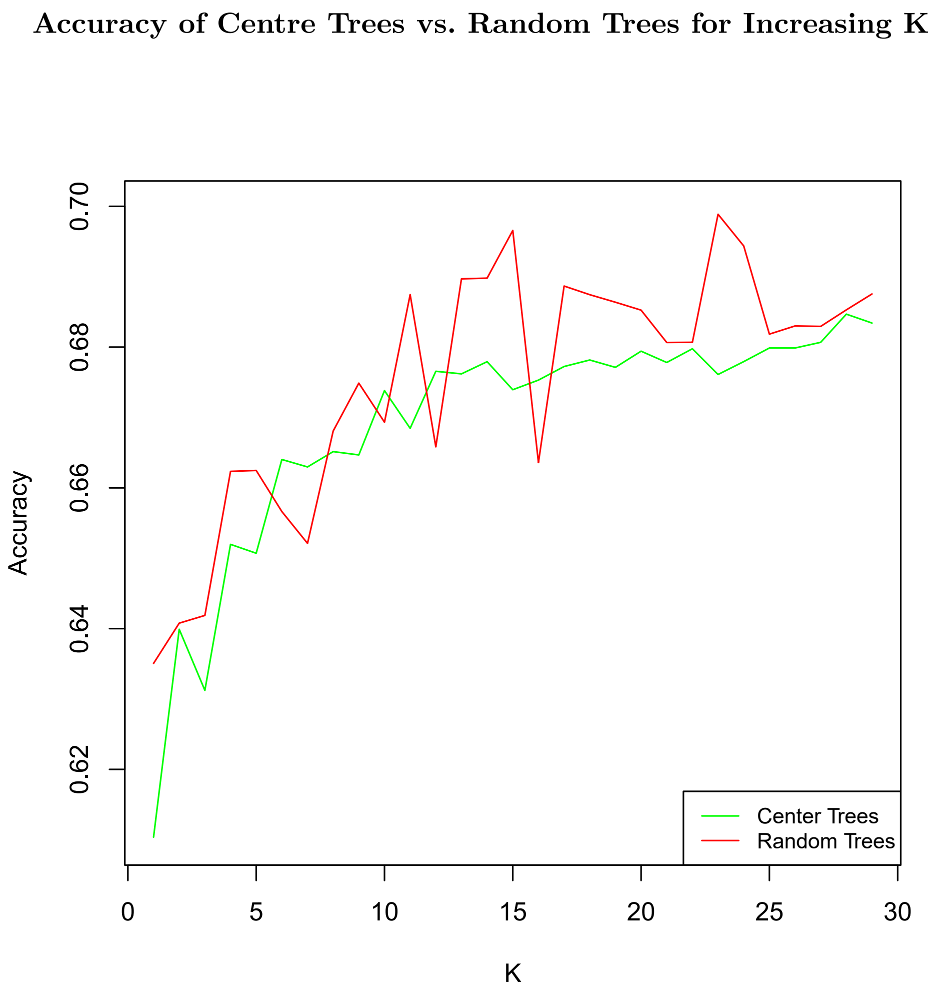


Figure 3.16: For increasing K the method of finding centre trees is used to calculate the accuracy of the centre trees versus K random trees, using a random forest trained on the tuberculosis dataset. From this plot we can determine that the centre trees do not outperform the random trees.

The plot of the accuracies of centre trees versus random trees for increasing K indicates that the centre trees do not represent the random forest. Their accuracies do not at any point significantly out-perform the randomly selected trees. As noted earlier in this chapter, we also compared the accuracy of the centre trees and random trees against the random forest, as well as the a single decision tree and logistic regression. Additionally, as we can see from Figure 3.14, overall the centre trees and random trees have comparable accuracy, with the random trees actually marginally outperforming the centre trees. Compared to other methods they both fail to represent the random forest, with the accuracy of the random forest as well as the decision tree and logistic regression being much higher. This result led us to speculate that instead there could be a correlation between the individual accuracies of the the trees and the overall accuracy of the cluster they belong to.

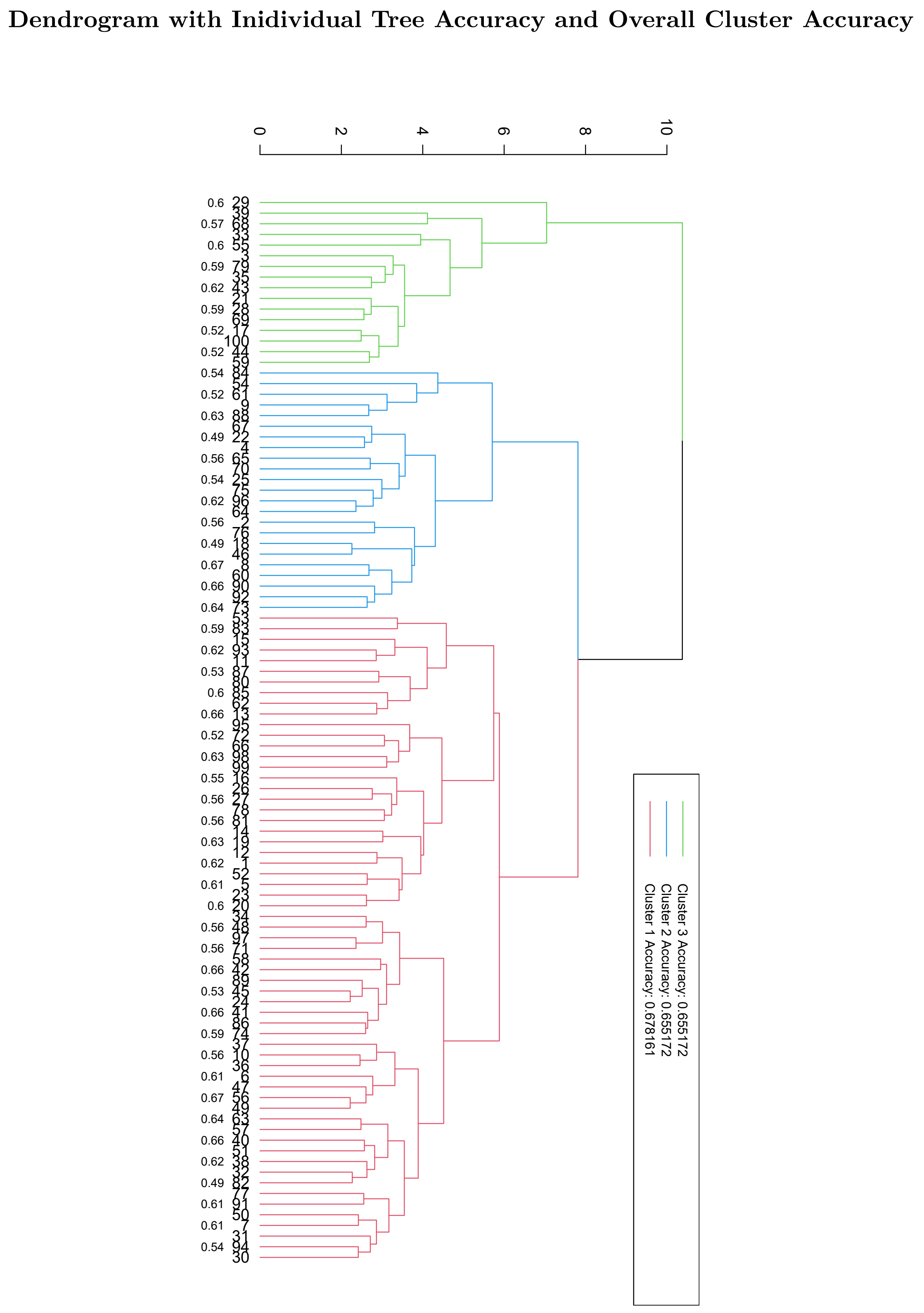


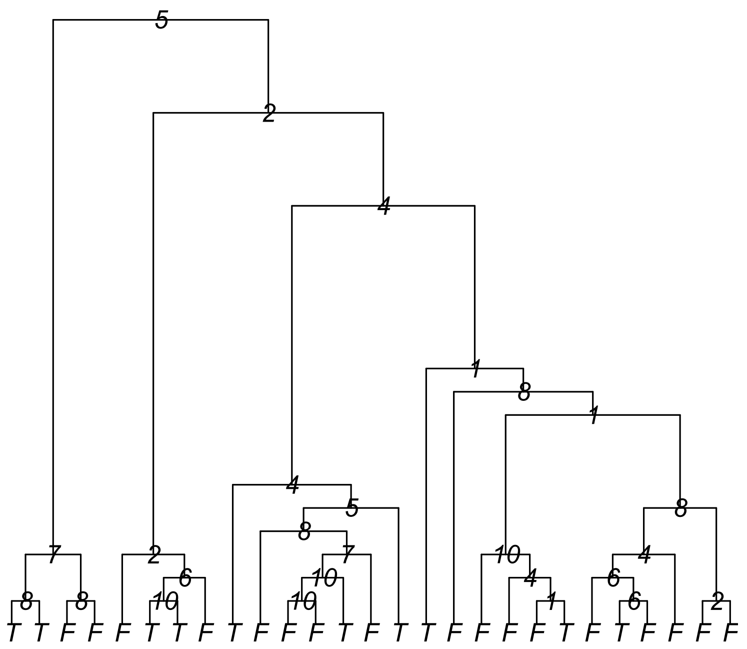
Figure 3.17: Dendrogram taken from the combined tuberculosis heatmap, clustered into three groups with the tree numbers labeled and the corresponding accuracy of each tree. The accuracy of the entire cluster is noted for each in the legend.

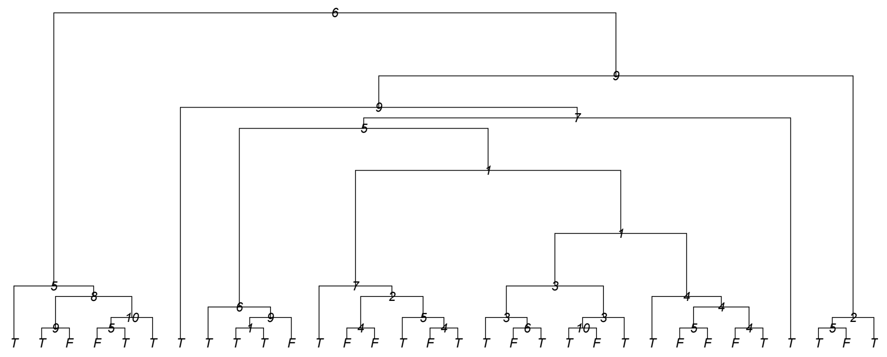
From the accuracies for the dendrogram on the individual trees we can see that there is a range of accuracies throughout each cluster, meaning the high performing trees and low performing trees did not get separated into clusters. The overall cluster accuracies are also notably close to each other, not indicating that any of them performed better than the other two as we had originally expected.

3.2.4 Synthetic Data

Following the method described in section 3.1.5 we created three synthetic decision trees, and then used those trees to generate 1500 data points, with 500 coming from each tree. Figure 3.18 below illustrates the three decision trees that were created.

Three Synthetic Decision Trees





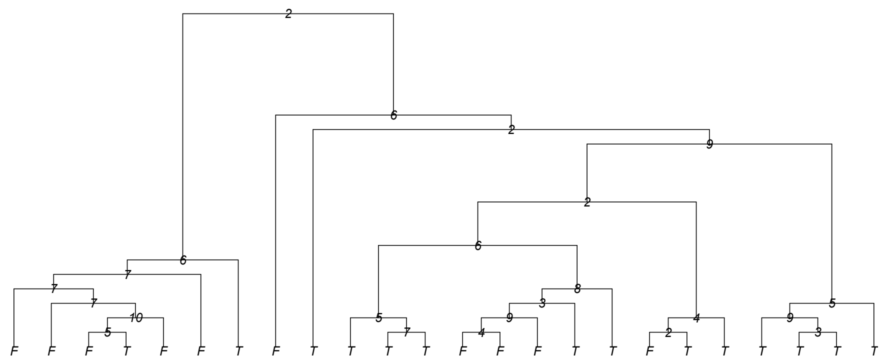


Figure 3.18: The first, second, and third randomly generated decision trees that were used to create synthetic data; 1500 observations in total with 500 from each decision tree.

Notice that the decision trees are not only different from each other but are quite different structurally. Ideally this will serve to create data points that vary greatly across the three groups, which should in turn lead to distinct clusters of trees in the random forest.
To investigate the similarity between the random forest trees and the three original trees, we selected a tree at random from the random forest so that we could examine it visually.

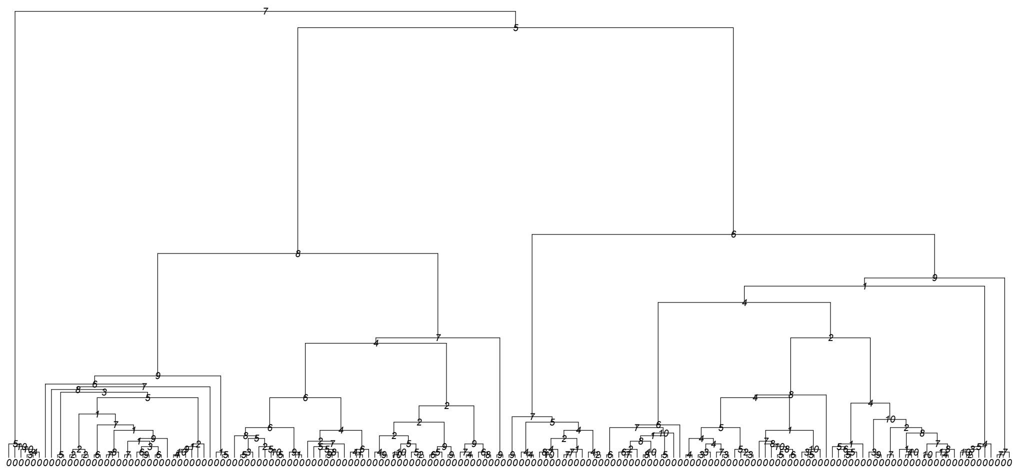


Figure 3.19: Tree 32 extracted from the random forest that was trained on the synthetic data.

By using our clustering method with the dendrogram attached to the heatmap, we determined that with $K = 3$, tree 32 belongs to cluster 2. By analysing tree 32 we can see that the two most common split variables are 4 and 5, which is similar to synthetic tree 2 whose most common variable is 5, and second most common is 4. The highest split variable for synthetic tree 1 is variable 8, but 4 is the second highest; however, it only splits on 5 twice. Synthetic tree 3 does not split on either 4 or 5 much. Another minor comparison is that for synthetic tree 2, one of the first pairs of sequential splits is variable 6 and variable 9, with variable 6 being the root node. In tree 32 this pairing occurs five times, and the pair variable 6 and variable 5 occurs three times which is the other root-daughter split. For synthetic trees 1 and 3, the root-daughter split pairs do not occur as often in tree 32 as that of synthetic tree 2. The balance of tree 32 also resembles synthetic tree 2 the most out of the three trees. However, despite considering these comparisons, there does not seem to be enough significant conclusive evidence that tree 32 strongly resembles any of the original ground truth trees.

Accuracies of Random Forest Trees

As mentioned in section 3.2.4 each of the three synthetic trees were used to generate 500 observations of data. The underlying intuition is that the trees grown in the random forest that was trained on this data would share a higher amount of characteristics with one of the original generating synthetic trees. Therefore, it follows that the trees in the random forest would have a higher performance on one of the three groups of data. For every decision tree in the random forest, we calculated the percentage of correctly classified data for each group of observations generated by the original synthetic decision trees, $A1 : 1-500$, $A2 : 501-1000$, and $A3 : 1001-1500$.

We visualised the accuracies of the decision trees on the three groups of data as a scatter plot, as seen in Figure 3.20

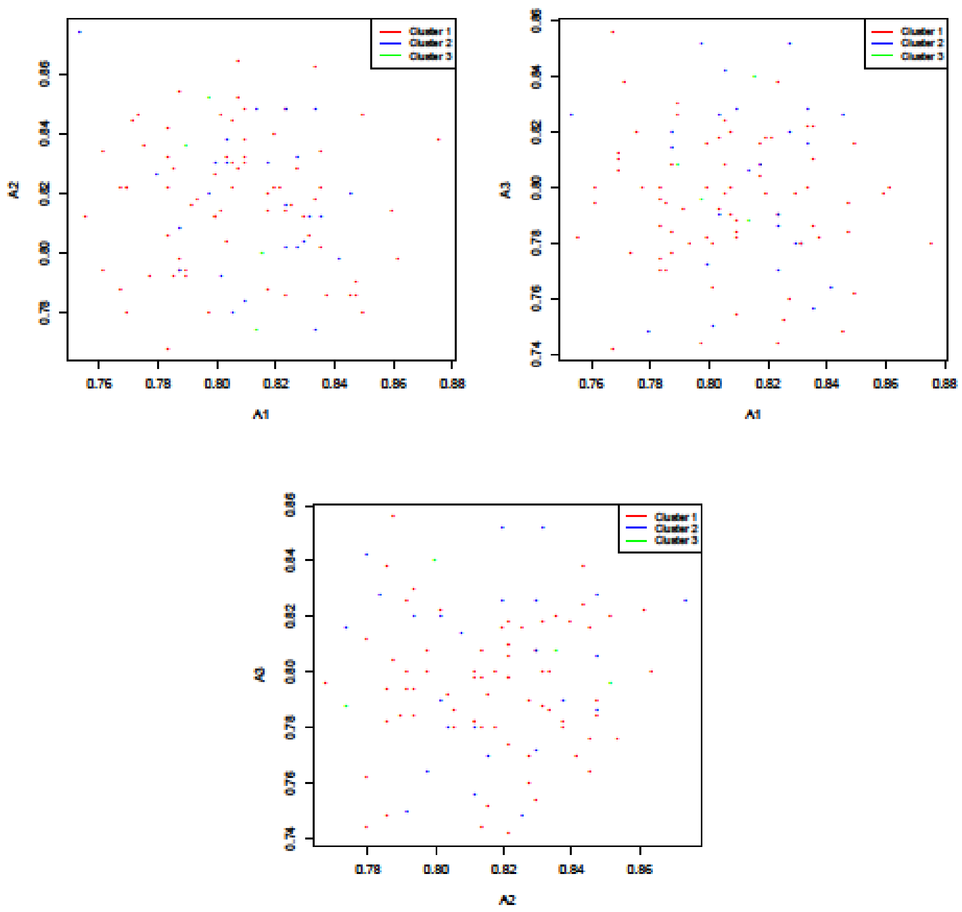


Figure 3.20: The points in each scatter plot represent the accuracy (in terms of proportion correct) of the prediction of tree T_i on sections A_j versus A_k of the synthetic data where $j \neq k$. (For example, for the top left scatter plot the x-axis is proportion correct for the A1 group of data, and the y-axis is the proportion correct for the A2 group of data.) The colour of the points represent the cluster number that the tree was assigned to by the dendrogram with $K = 3$.

Each point in the scatter plot above is the percentage of correctly classified data by the decision tree T_i, $i = 1, ..., 100$ on section A_j of the data versus section A_k, where $j \neq k$. The colour of each point references which cluster the tree belongs to which was assigned by the dendrogram on Figure 3.7 with $K = 3$. Similarly to the boxplot, we expect to see a distinction between the colours of the points in the scatter plot, meaning we would observe the colours in separate clusters. For example, in the top left plot, groups of trees that occur on the bottom right of the scatter plot indicate that trees in this region had a higher performance on classifying the data from section $A1$ and a lower performance on classifying the data from section $A2$. If the trees were clustered by accuracy (implying they have a higher resemblance to the synthetic tree that generated section $A1$ of the data) then the trees in that region of the plot would be from the same cluster and share the same colour. However, the scatter plot clearly displays a random distribution of the colours throughout the plot. This implies that trained decision trees do not have a higher classification accuracy on specific sections of data, which implies that the random forest did not recover the original synthetic decision trees in clusters.

This result is supported by the heatmap of dissimilarities shown in Figure 3.7. The lack of distinct structure and clusters is reflected in the lack of distinction in accuracies for different groups of data.

T-Tests for Synthetic Data

For the synthetic data, one of the aspects of the experiments we were interested in exploring further was which measures of the trees in the random forest, which make up the combined heatmap, are actually contributing towards the clusters. Therefore, for each measure or characteristic (such as maximum tree depth), we use a paired t-test to find a p-value to indicate whether the differences between the clusters are significant. However, there are two types of t-test that we employ depending on the comparison function. The comparison functions: maximum tree depth, minimum variable depth, and variable count, are based on the comparison of a single measurement from a tree (and then the comparison is computed by taking the pairwise absolute difference between these single measurements). Whereas, the comparison functions: transmission tree difference and symmetric set difference require a pair of trees to quantify the difference between them. Therefore, we require two types of hypothesis tests. If we consider the comparison function for maximum depth, then the hypothesis test for this can be written as:

$$H_0 : \mu_1 = \mu_2$$
$$H_1 : \mu_1 \neq \mu_2$$

where

$$\mu_1 = \frac{1}{n_1} \sum_{i=1}^{n_1} \phi(T_{1,i}),$$

n_1 is the number of trees in cluster 1 and recall from section 2.5 that $\phi(T_1)$ is a vector of the maximum tree depths for all trees in cluster 1. Therefore μ_1 is the mean of the maximum depths of the trees in cluster 1. Similarly, μ_2 is the mean of the maximum depths of all of the trees in cluster 2. In this case we reject H_0 if the t-test gives a p-value that indicates there is a significant difference between clusters 1 and 2 according to the comparison function of maximum tree depth. For the other comparison functions that only require one tree: minimum variable depth and variable count, the hypothesis test is similar.

The first structural aspect we tested was the tree depth. We calculated the maximum depths for all of the trees in cluster i and again for cluster j, where $i, j \in [1, ..., \text{number of clusters}]$, and then calculated the t-test for the two vectors of maximum depths from cluster i and j. The p-values for each combination of clusters are found in the following table.

Tree Depth	
Cluster Combination	p-value
1, 2	**4.524e-15**
1, 3	0.002
2, 3	0.008

Table 3.1: T-test of the contribution of the maximum tree depth comparison towards the overall clustering.

The next metric is the phylogenetic metric for transmission trees. Since this metric relies on a pair of trees to compute the difference between them, the t-test is more complicated than the previous one. For comparing cluster i with cluster j, we want to calculate the differences between trees within cluster i, and then compare these differences with the differences between the trees in cluster i with those in cluster j. This can be stated mathematically by

$$S^{\text{intra}(k)} = \{d(T_{ki}, T_{kj}) : i \neq j\}, \quad S^{\text{inter}(k)} = \{d(T_{ki}, T_{k'j}) : i \neq j, k \neq k'\}$$

where intra refers to the trees within the same cluster and inter the trees in different clusters. Therefore, the hypothesis test for the t-test for this comparison function is

$$H_0 : \mu_1 = \mu_2$$
$$H_1 : \mu_1 \neq \mu_2$$

where in this case $\mu_1 = \text{mean}(S^{\text{intra}(k)})$ where $k = 1$ and $\mu_2 = \text{mean}(S^{\text{inter}(k)})$ where $k = 1$ and $k' = 2$.

Transmission Tree Difference	
Cluster Combination	p-value
1, 2	**9.723-35**
1, 3	**2.699e-59**
2, 3	**4.704e-10**

Table 3.2: T-test of the contribution of the transmission tree difference comparison towards the overall clustering.

Similarly to the phylogenetic metric, the symmetric set difference comparison function also relies on a pair of trees to calculate the difference between them. Therefore, the t-test follows the same procedure as the phylogenetic t-test and H_0 and H_1 are the same as the hypothesis test for the transmission tree difference but with a different comparison function.

Symmetric Set Difference	
Cluster Combination	p-value
1, 2	6.398e-03
1, 3	**2.615e-06**
2, 3	5.236e-02

Table 3.3: T-test of the contribution of the symmetric set difference comparison towards the overall clustering.

Lastly, the final two tree components, variable count and minimum variable depth, are dependent on which variable we are analysing. For example, for variable count, we count how many times a specific variable V_p occurred in each tree in cluster i, and again how many times it occurred in each tree cluster j, then compute the t-test for these two vectors for each combination of clusters. By extending this to all variables, the t-test tells us how important each variable was in differentiating between trees and grouping them into clusters, in terms of the variable count. The same procedure is followed for the minimum variable depth.

Variable Count				
Variable	Importance	Cluster 1 and 2 p-value	Cluster 1 and 3 p-value	Cluster 2 and 3 p-value
V1	1.095	0.879	0.759	0.729
V2	10.611	0.090	0.290	0.480
V3	-1.397	0.502	0.916	0.884
V4	0.123	0.519	0.918	0.796
V5	12.986	0.286	0.901	0.544
V6	14.231	0.276	0.523	0.205
V7	6.202	0.107	0.231	0.600
V8	9.780	0.192	0.962	0.329
V9	4.811	0.094	0.600	0.187
V10	0.841	0.002	0.925	0.187

Table 3.4: T-test of the contribution of the variable count comparison towards the overall clustering.

Minimum Variable Depth				
Variable	Importance	Cluster 1 and 2 p-value	Cluster 1 and 3 p-value	Cluster 2 and 3 p-value
V1	1.095	0.454	0.220	0.128
V2	10.611	0.197	0.225	0.324
V3	-1.397	0.021	0.400	0.882
V4	0.123	0.950	0.366	0.386
V5	12.986	0.093	0.640	0.952
V6	14.231	0.469	0.001	0.000
V7	6.202	0.884	0.580	0.621
V8	9.781	0.419	0.467	0.868
V9	4.811	0.618	0.915	0.859
V10	0.841	0.138	0.663	0.971

Table 3.5: T-test of the contribution of the minimum variable depth comparison towards the overall clustering.

Since we conducted multiple experiments, we use the Bonferroni correction to confirm which tests are still significant under this condition. Therefore, if the *p*-value is less than

$$P(T_i \text{ passes } |H_0) \leq \frac{\alpha}{n} = \frac{0.05}{66} = 0.0007\bar{5}$$

given n tests T_i for hypotheses H_i, $(1 \leq i \leq n)$, under the assumption H_0 that all hypotheses H_i are false, and $\alpha = 0.05$ is the original passing p-value. Therefore, the tests that pass the Bonferroni correction and are still statistically significant are bolded in each table. We acknowledge that we violate the assusmption of independence for the Bonferroni correction in the tests for the transmission tree and SSD comparison functions (since when we compare trees within the same cluster, $d(T_i, T_j)$ and $d(T_i, T_l)$ both rely on T_i and are therefore not independent). We speculate that the dependence is not significant, and future work could be done to use a statistical test to provide concrete evidence for this.

Note that all of the tests for the comparison functions involving variables, namely, variable count and minimum variable depth, do not pass the Bonferroni correction. One explanation for this could be the bagging method that is employed to randomly select variables at each node in random forest trees when they are trained.

Chapter 4

Conclusion

We have provided a novel method for quantifying the differences in decision trees that have been extracted from a random forest. The method works by applying pairwise comparison functions to the trees to form an $N \times N$ matrix where N is the number of trees in the random forest. We then analyse the differences by normalising the matrices and summing them together and using a heatmap to visualise the values in the combined matrix. We examined the clusters that formed by the dendrogram grouping together trees that were similar according to the comparison functions. Using these clusters, we conducted research into the accuracy of what we called centre trees (trees that were found to be the centres of the clusters using multi-dimensional scaling) versus randomly extracted trees. We compared these accuracies to those acquired from other standard statistical methods. We also developed a method for generating synthetic data from decision trees that are represented as nested lists, and conducted similar accuracy analysis. We used the synthetic data to determine which of the comparison functions were significantly contributing to the overall clustering using pairwise t-tests.

The heatmaps for all of the datasets had distinct clusters except for the synthetic dataset, which had sparse structure. Arguably the synthetic dataset should have had the most distinct clustering with three groups due to the data being generated from three ground truth trees. The visible clusters in the other heatmaps motivated the analysis of accuracy using centre trees. We believed that the random forest could be reduced to a handful of centre trees extracted from it that would represent the entire random forest, hence reducing complexity. Among our datasets, we chose a deeper analysis of the tuberculosis dataset because in Figure 3.14 one can see that the centre trees and random trees have a comparable performance and both are less accurate than the entire random forest. We believed that the centre trees would outperform the random trees as K increases and approach the higher accuracy of the entire random forest. However, from figure 3.16 we can see that as K increases, the accuracy of the centre trees and the random trees increase at the same rate. This means that the random forest cannot be represented by a reduced number of trees (through our method, without losing accuracy), and these trees cannot be analysed to improve the explainability of the inner workings of the random forest. We also repeated these experiments for the other datasets (excluding the synthetic dataset) and obtained similar results making the conclusion more robust.

The synthetic data was created using three original decision trees, or ground truth trees. We used this data to train and test a random forest with the intuition of testing if a random forest is capable of recovering the decision trees from the data. If the random forest did indeed recover copies of the decision trees or have groups of trees that resembled the originals then the random forest would have high accuracy on the data. The individual decision trees in the random forest would also have higher accuracies on certain sections of data, specifically the section of data generated from the ground truth tree it resembles. However, as we can see from the decision tree that was randomly extracted from the random forest, depicted in figure 3.19, this decision tree only loosely resembles the second ground truth tree shown in figure 3.18. In terms of accuracy, we can conclude from figure 3.20 that the decision trees in the random forest that were grouped into clusters do not consistently have a higher accuracy on the three original groups of data generated by the ground truth decision trees. This shows that a random forest is not a mixture of decision trees and is not consistent.

The lack of structure in the synthetic heatmap is unintuitive and surprising. The visible clustering in the other heatmaps is strong evidence that the comparison functions are successfully identifying differences in the decision trees. It also shows that the dendrogram is supposedly grouping together trees that are similar. However, since the synthetic heatmap is where we expected to see three clusters, because of the data generated from the three ground truth trees, it raises the question of which comparison functions are actually contributing to the overall clustering. We can see from the t-test results that only three out of the five comparison functions are significant. Therefore, the two comparison functions that are not significant, are the ones that employ characteristics about the splitting variables in the decision trees, variable count and minimum variable depth.

4.1 Future Work

In this work we extended the Kendall et al. metric to apply it to decision trees, and developed four novel comparison functions. For future work we propose another comparison function which quantifies how balanced or unbalanced the tree is, and then for two decision trees take the absolute value of the difference between these values. This new comparison function is aimed at capturing more information about the topology of the tree, since currently the only comparison function we use for topology is maximum tree depth.

A second interesting avenue for future work is continuing the analysis with ground truth trees, but with a more simplified approach. In this experiment we suggest only using one ground truth decision tree, and initially only generate a small amount of data points from this tree. We theorise that if a random forest is trained on this data with a small amount of decision trees (e.g. 10), over time as we increase the number of data points generated from the ground truth tree, the decision trees in the random forest will converge to the original tree, up to some isomorphism. (This is to ask: Are random forests identifiable?)

Lastly, we were able to establish clusters of decision trees using our comparison functions, however we were unable to identify the insight into the inner workings of the random forest these clusters provide us. We attempted to show that centre trees of these clusters could be defined to reduce the complexity of the random forest, and that their accuracy would increase more than random trees with increasing cluster numbers. Future work could be done to further explore the clusters and their properties, and develop potential ways in which the clusters improve the interpretability of random forests and their decision trees.

References

Anderson, E. (1936). The species problem in iris. *Annals of the Missouri Botanical Garden*, *23*(3), 457–509.

Belevitch, V. (1962). Summary of the history of circuit theory. *Proceedings of the IRE*, *50*(5), 848–855.

Billera, L. J., Holmes, S. P., & Vogtmann, K. (2001). Geometry of the space of phylogenetic trees. *Advances in Applied Mathematics*, *27*(4), 733–767.

Borg, I., & Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications.* Springer Science & Business Media.

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

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). Classification and regression decision trees. *Monterey, CA: Wadsworth & Brooks.*

De Cock, D. (2011). Ames, Iowa: Alternative to the boston housing data as an end of semester regression project. *Journal of Statistics Education*, *19*(3).

Dua, D., & Graff, C. (2017). *UCI machine learning repository.* Retrieved from `http://archive.ics.uci.edu/ml`

Fehrman, E., Muhammad, A. K., Mirkes, E. M., et al. (2015). *The five factor model of personality and evaluation of drug consumption risk.* arXiv. Retrieved from `https://arxiv.org/abs/1506.06297` doi: 10.48550/ARXIV.1506.06297

Fehrman, E., Muhammad, A. K., Mirkes, E. M., et al. (2017). The five factor model of personality and evaluation of drug consumption risk. In *Data science* (pp. 231–242). Springer.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, *7*(2), 179–188.

Harrison Jr, D., & Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, *5*(1), 81–102.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). Springer.

Kendall, M., Ayabina, D., Xu, Y., et al. (2018). Estimating transmission from genetic and epidemiological data: a metric to compare transmission trees. *Statistical Science*, *33*(1), 70–85.

Kim, J., Rosenberg, N. A., & Palacios, J. A. (2020). Distance metrics for ranked evolutionary trees. *Proceedings of the National Academy of Sciences*, *117*(46), 28876–28886.

Kuznetsova, N., Westenberg, M., Buchin, K., et al. (2014). Random forest visualization. *Eindhoven University of Technology.*

Lewis, R. J. (2000). An introduction to classification and regression tree (cart) analysis. In *Annual meeting of the society for academic emergency medicine in san francisco, california* (Vol. 14).

Liu, P. (2021). A tree distinguishing polynomial. *Discrete Applied Mathematics*, *288*, 1–8.

McCrae, R. R., & Costa Jr, P. T. (2004). A contemplated revision of the neo five-factor inventory. *Personality and individual differences*, *36*(3), 587–596.

Mordvintsev, A., Olah, C., & Tyka, M. (2015). Inceptionism: Going deeper into neural networks.

Nielsen, F. (2016). Hierarchical clustering. In *Introduction to hpc with mpi for data science* (pp. 195–211). Springer.

Pesantez-Narvaez, J., Guillen, M., & Alcañiz, M. (2019). Predicting motor insurance claims using telematics data—xgboost versus logistic regression. *Risks*, *7*(2), 70.

Robinson, D. F., & Foulds, L. R. (1979). Comparison of weighted labelled trees. In *Combinatorial mathematics vi* (pp. 119–126). Springer.

Ruddy, M., Davies, A., Yates, M., et al. (2004). Outbreak of isoniazid resistant tuberculosis in north london. *Thorax*, *59*(4), 279–285.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206–215.

Stanford, M. S., Mathias, C. W., Dougherty, D. M., et al. (2009). Fifty years of the barratt impulsiveness scale: An update and review. *Personality and individual differences*, *47*(5), 385–395.

Torgerson, W. S. (1958). Theory and methods of scaling.

Welling, S. H., Refsgaard, H. H., Brockhoff, P. B., & Clemmensen, L. H. (2016). Forest floor visualizations of random forests. *arXiv preprint arXiv:1605.09196*.

Xu, Y., Stockdale, J. E., Naidu, V., et al. (2020). Transmission analysis of a large tuberculosis outbreak in london: a mathematical modelling study using genomic data. *Microbial genomics*, *6*(11).

Ypma, R. J., van Ballegooijen, W. M., & Wallinga, J. (2013). Relating phylogenetic trees to transmission trees of infectious disease outbreaks. *Genetics*, *195*(3), 1055–1062.

Zuckerman, M. (1994). *Behavioral expressions and biosocial bases of sensation seeking.* Cambridge university press.

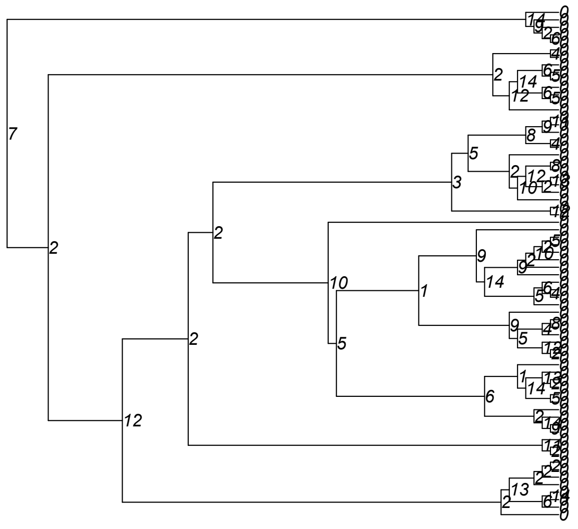
Appendix A

Appendix

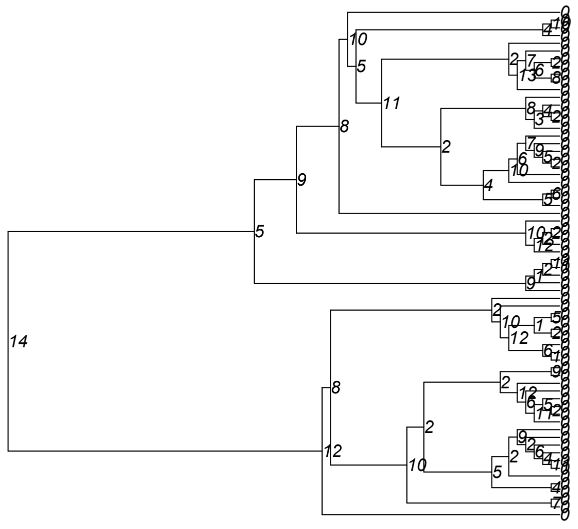
A.1 Clustering of Trees for TB

Using the method outlined in Chapter 2, we trained a random forest on the tuberculosis dataset with a reduced number of trees ($N = 20$) in order to visually analyse the characteristics of the trees within $K = 3$ clusters. In each figure below, the title first enumerates which tree in the random forest it is between 1 and 20. The second half of the title indicates which cluster the tree belongs to. The figures are organised according to the cluster number in the order of cluster 1, 2, 3.

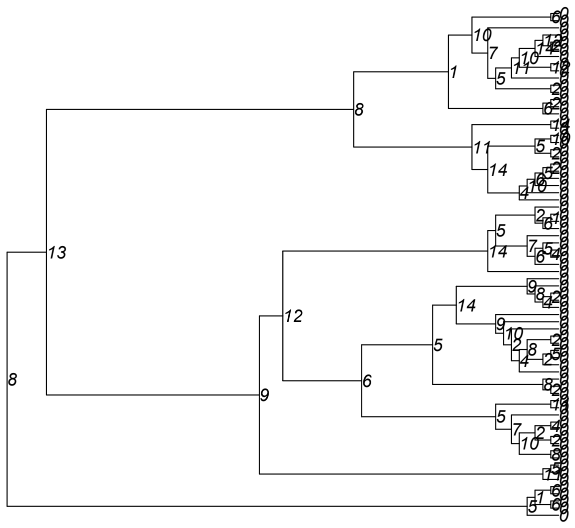
Tree: 1, Cluster: 1



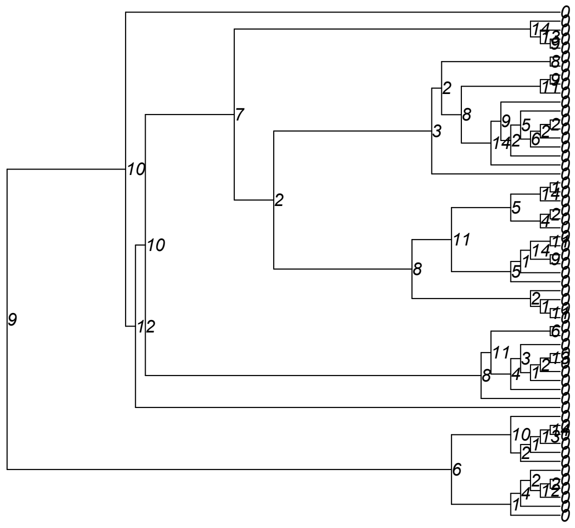
Tree: 12, Cluster: 1



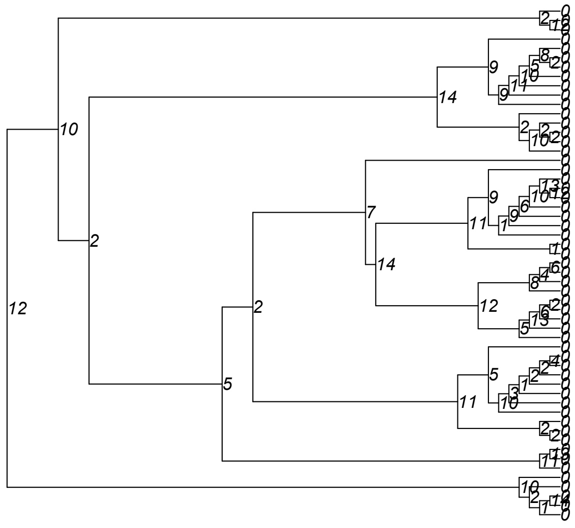
Tree: 16, Cluster: 1



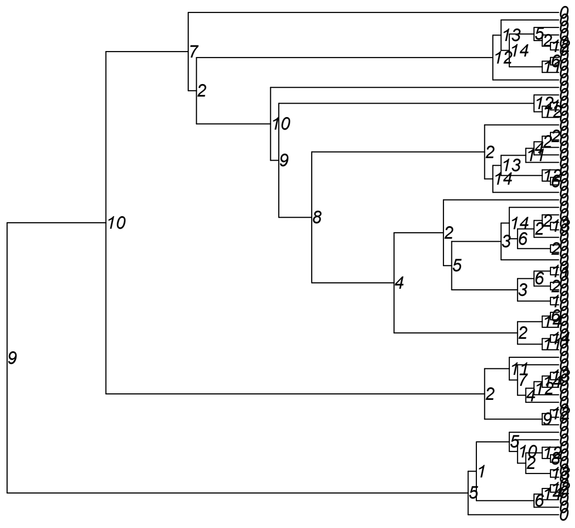
Tree: 17, Cluster: 1



Tree: 20, Cluster: 1



Tree: 3, Cluster: 1



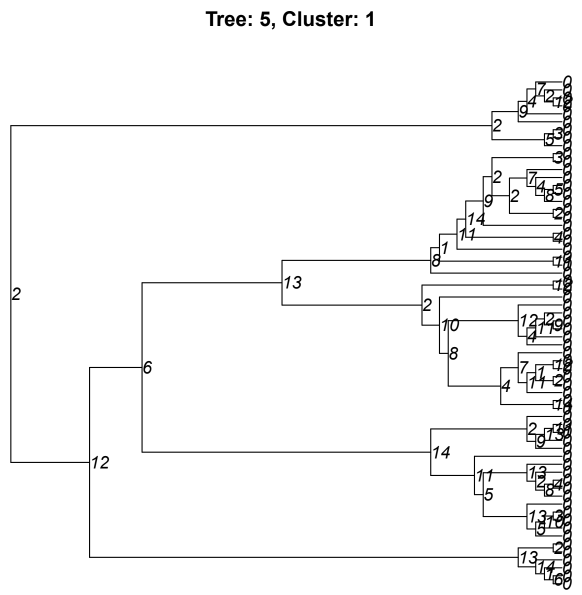
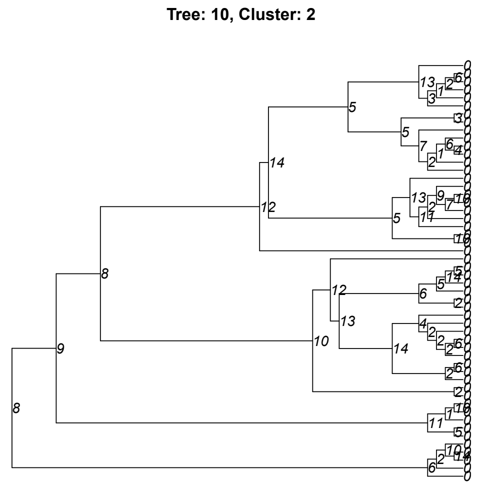
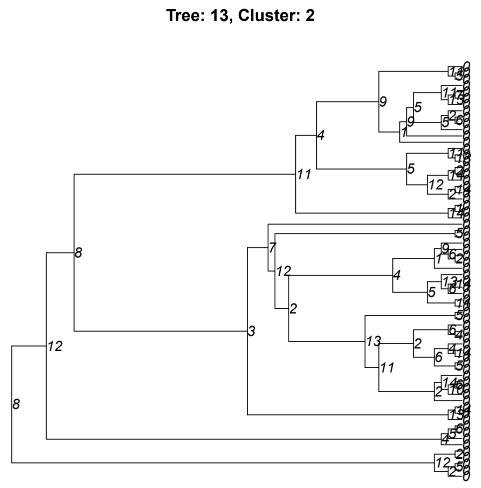
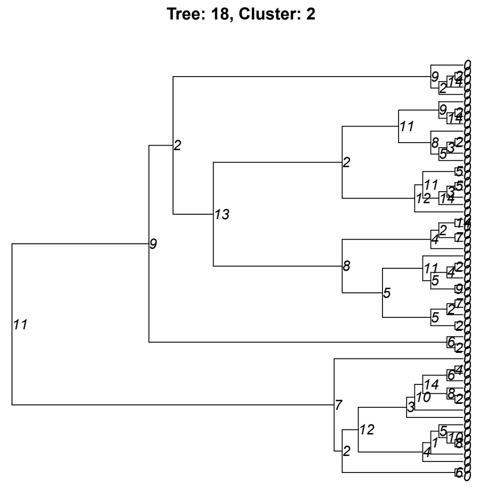
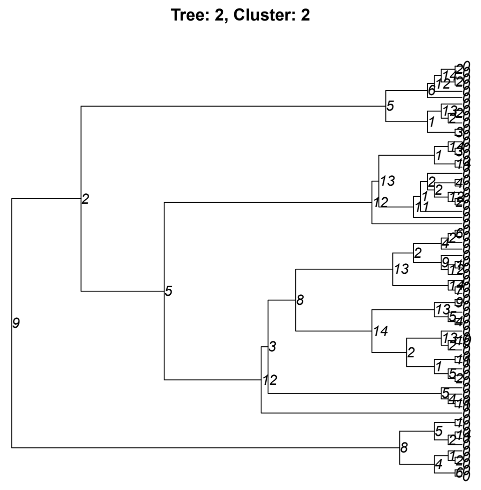

Tree: 5, Cluster: 1

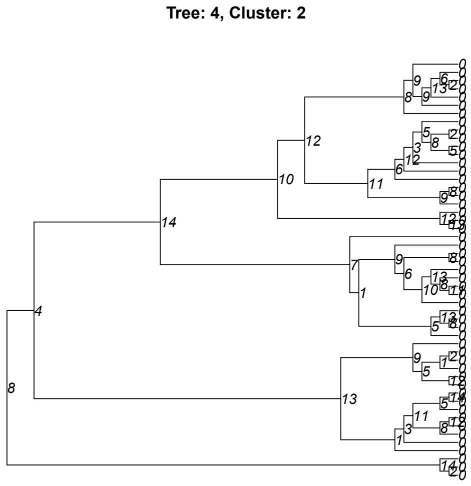
Figure A.1

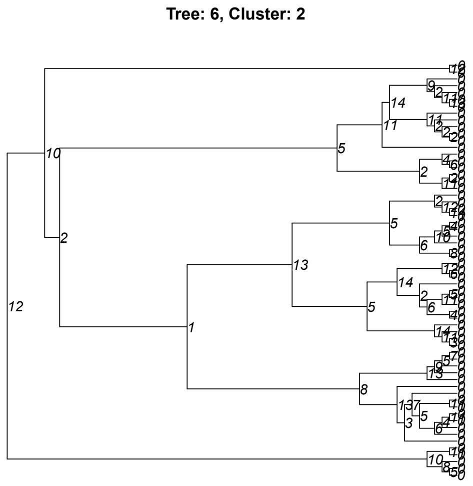

Tree: 10, Cluster: 2

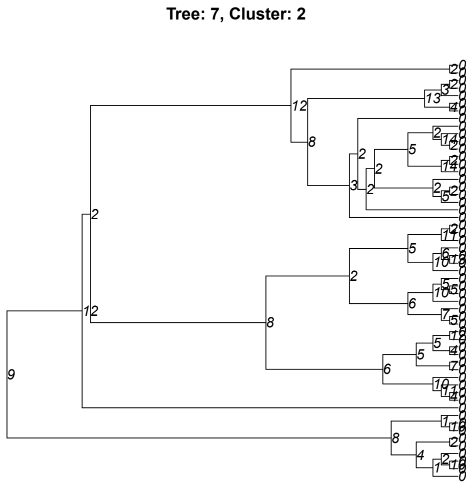

Tree: 13, Cluster: 2

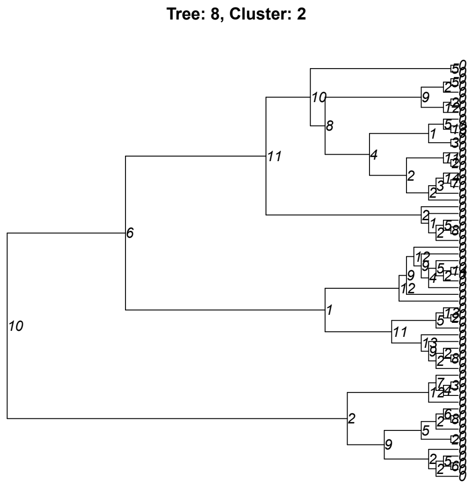

Tree: 18, Cluster: 2


Tree: 2, Cluster: 2


Tree: 4, Cluster: 2


Tree: 6, Cluster: 2


Tree: 7, Cluster: 2


Tree: 8, Cluster: 2

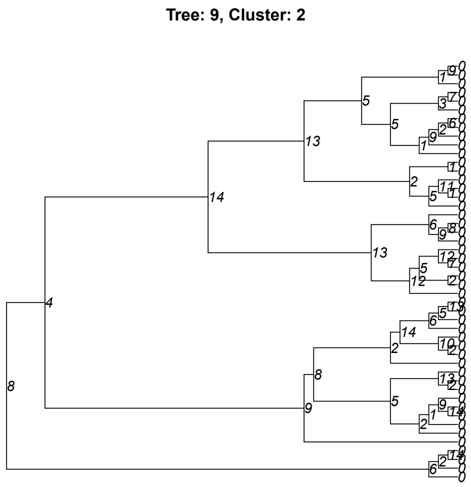

Tree: 9, Cluster: 2

Figure A.2

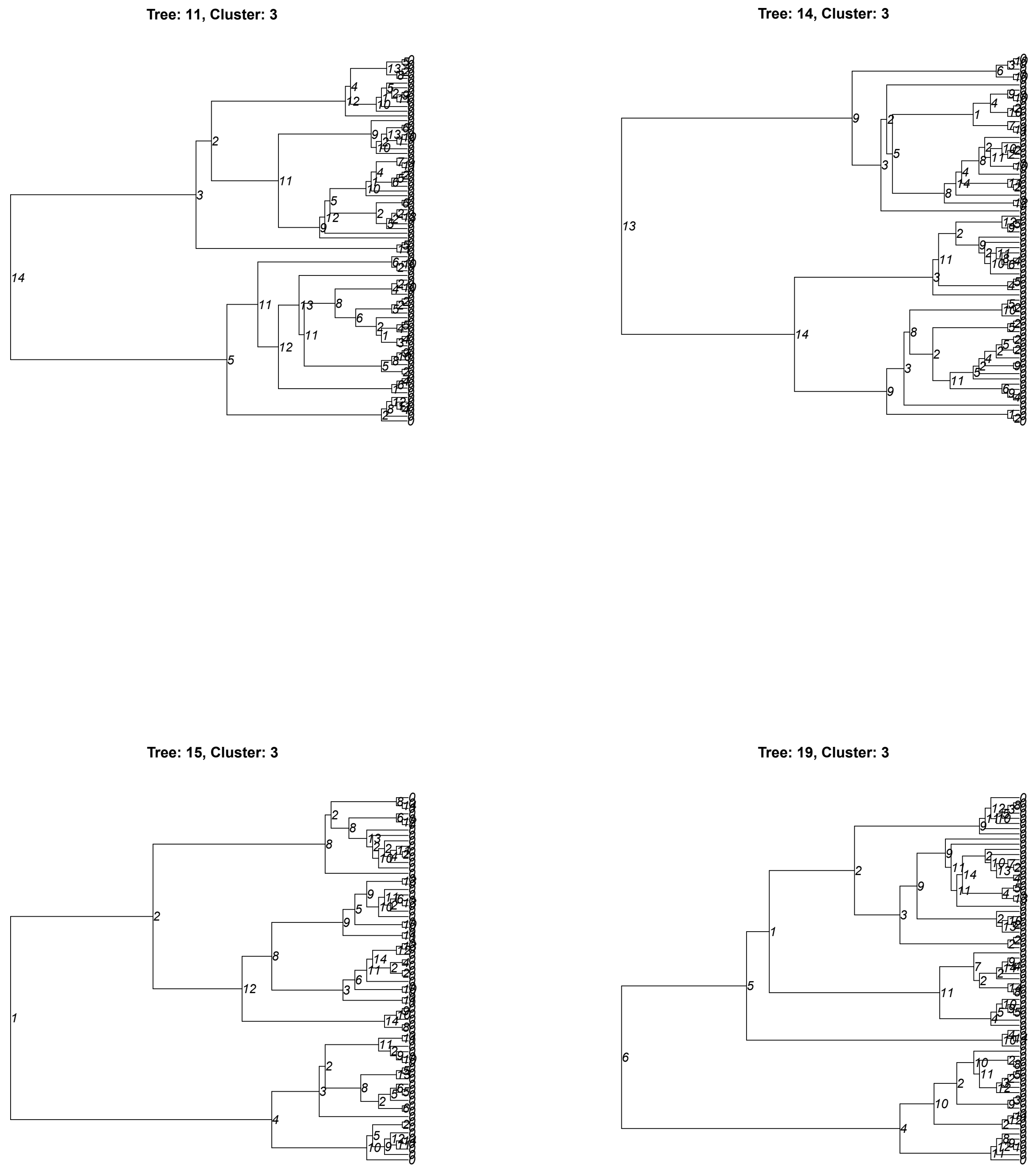

Tree: 11, Cluster: 3
Tree: 14, Cluster: 3
Tree: 15, Cluster: 3
Tree: 19, Cluster: 3

Figure A.3