Contents

Analysis of Varying Approaches to Regression Trees 1
Exploratory Data Analysis 1
Part 1: Regression Tree Using *rpart* 3
Part 2: Regression Tree Using *party* 3
Part 3: Regression Tree Using *C50* 4
Part 4: Tree Comparison 6
Conclusion 6

Analysis of Varying Approaches to Regression Trees

The purpose of this assignment was to compare a variety of possible methods for implementing regression trees in R. The *iris* data set from the *datasets* package in R was used throughout the analysis. The iris data set contains the widths and lengths of sepals and petals for three different species of iris flowers. The data set contains a total of 150 observations – 50 of each different species. For validation of results, the data was split randomly into training and testing subsets at an 80:20 ratio of training to testing.

Three different R packages were used to develop regression trees for this data. The three packages used are *rpart, party* and *C50*. Each of the three trees was developed using the training data subset and then validated for accuracy using the testing subset. The trees were used to classify the species of flower based on petal and sepal length and width. The three possible species are *setosa*, *virginica* and *versicolor*.

Exploratory Data Analysis

Before developing any tree models, the four predictor variables were analyzed to determine if any of the four had characteristics unique to a given species. The four predictor variables were:

1. Sepal length

2. Sepal width
3. Petal length
4. Petal width

For each variable, a boxplot distribution broken down by species was plotted.

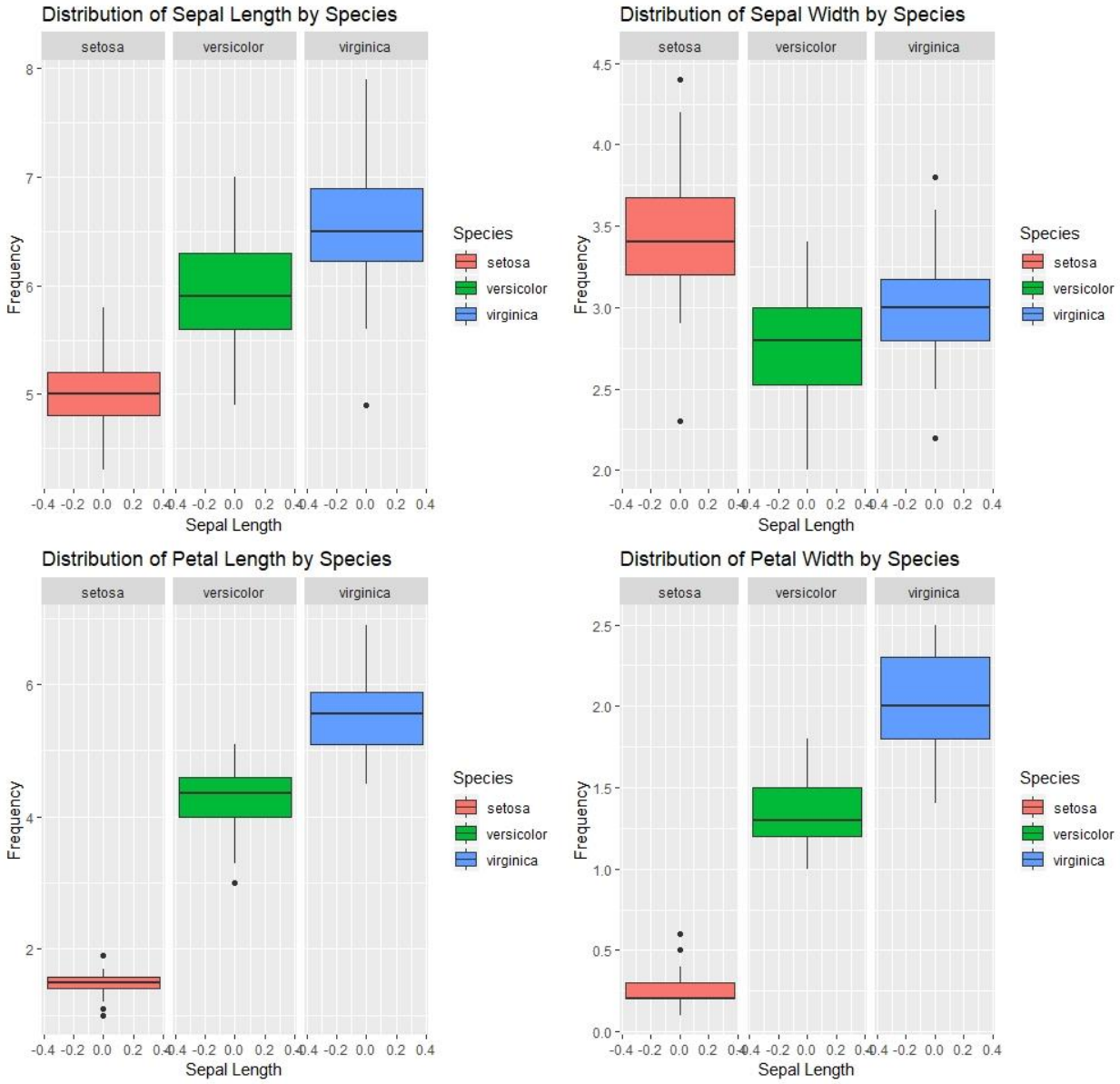

Figure 1: Boxplots of variable distribution by species

From the boxplots of each variable, we can see that the disparity among species for petal features is much greater than that of sepal features. Of all four features, petal width shows the highest disparity, indicating this variable is likely to be useful when classifying the iris flowers.

Part 1: Regression Tree Using *rpart*

The first regression tree was developed using the *rpart* library in R. Figure 2 shows the plot of the generated tree using the training data subset which contains a total of 120 observations.

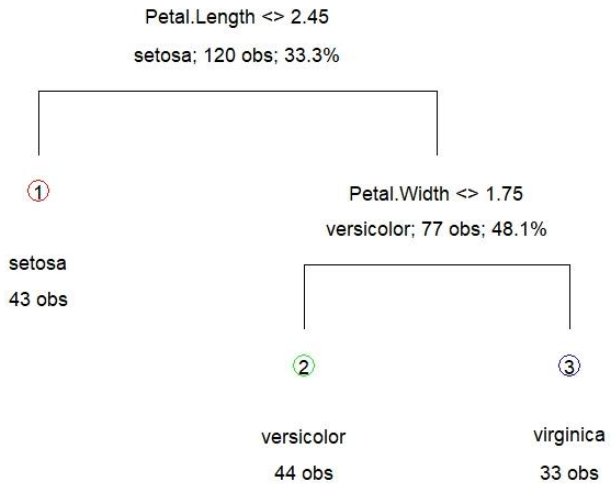

Figure 2: Tree generated using *rpart* library

The tree using the *rpart* library has a total of two internal nodes and three terminal nodes. Each terminal node corresponds to one of the three possible iris species. From the plot of the tree, we see that the first split is made based on petal length which indicates this is the most important feature when classifying species. If the petal length is less than 2.45, the observation is classified as a *setosa*. The next split uses petal width to determine between *versicolor* and *virginica*. If petal width is less than 1.75, the observation is classified as *versicolor*. Otherwise the observation is classified as *virginica*. An important note about the rpart library is how it decides to make splits within a tree. The *rpart* library uses the Gini Impurity measurement which calculates the probability of incorrectly classifying an observation based on an arbitrary split and then selecting the split with the lowest classification error.

Part 2: Regression Tree Using *party*

The second tree model was developed using the *party* package. The tree is shown in Figure 3 below.

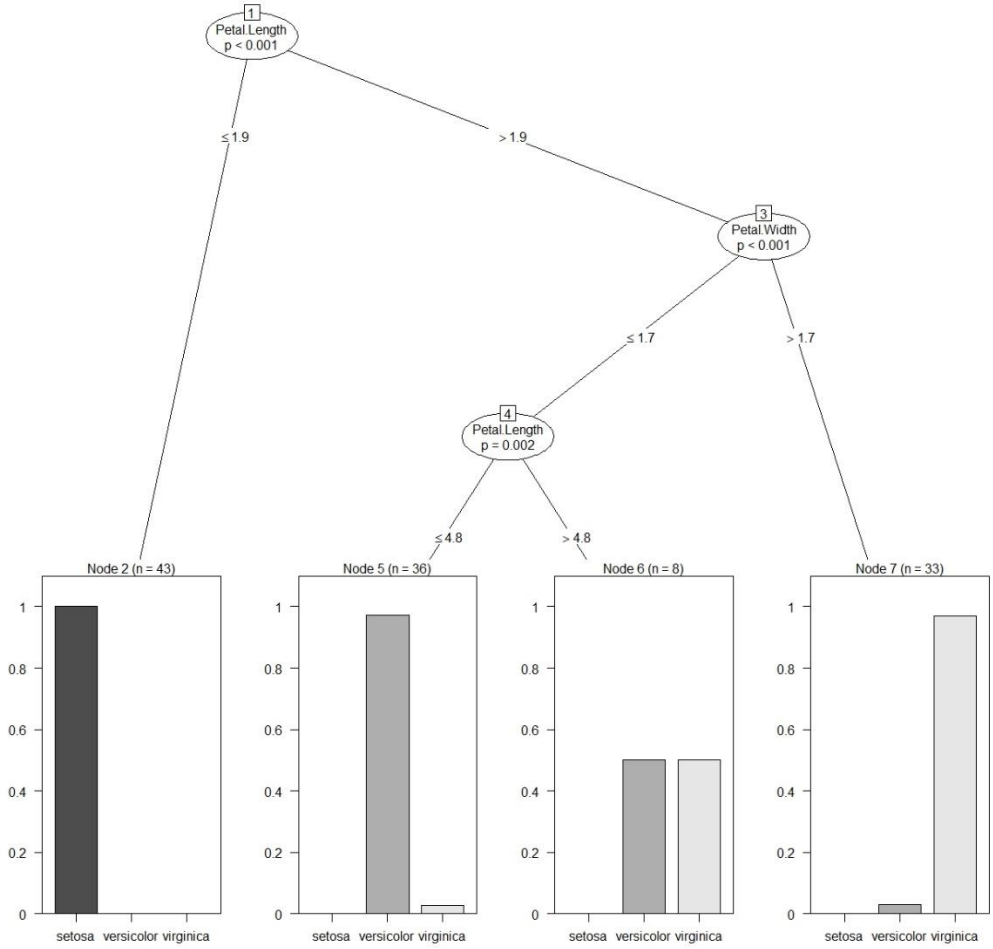

Figure 3: Tree using *party* library

Like the tree using rpart, the party tree uses petal length as the initial split variable. If the length is less than 1.9, the flower is classified as a *setosa*. Otherwise, another split is made using petal width. Any remaining flowers with width over 1.7 are considered *virginica*. Unlike the *rpart* tree, the party tree includes a third internal node, splitting between petal length if flowers are not previously classified as *setosa* or *virginica*. This split checks if petal length is less than or equal to 4.8, in which case the flowers are classified as *versicolor*.

The *party* package implements a slightly different splitting algorithm from the *rpart* package. The *party* package measures the association between the independent variables and the predicted variable with p-value used as an indicator of the strength of association. Splits are then made by which variable has the strongest association. To prevent from overfitting, the algorithm limits the number of splits by setting a threshold for the strength of association.

Part 3: Regression Tree Using *C50*

The third regression tree was implemented using the *C50* package. The tree is shown in Figure 4.

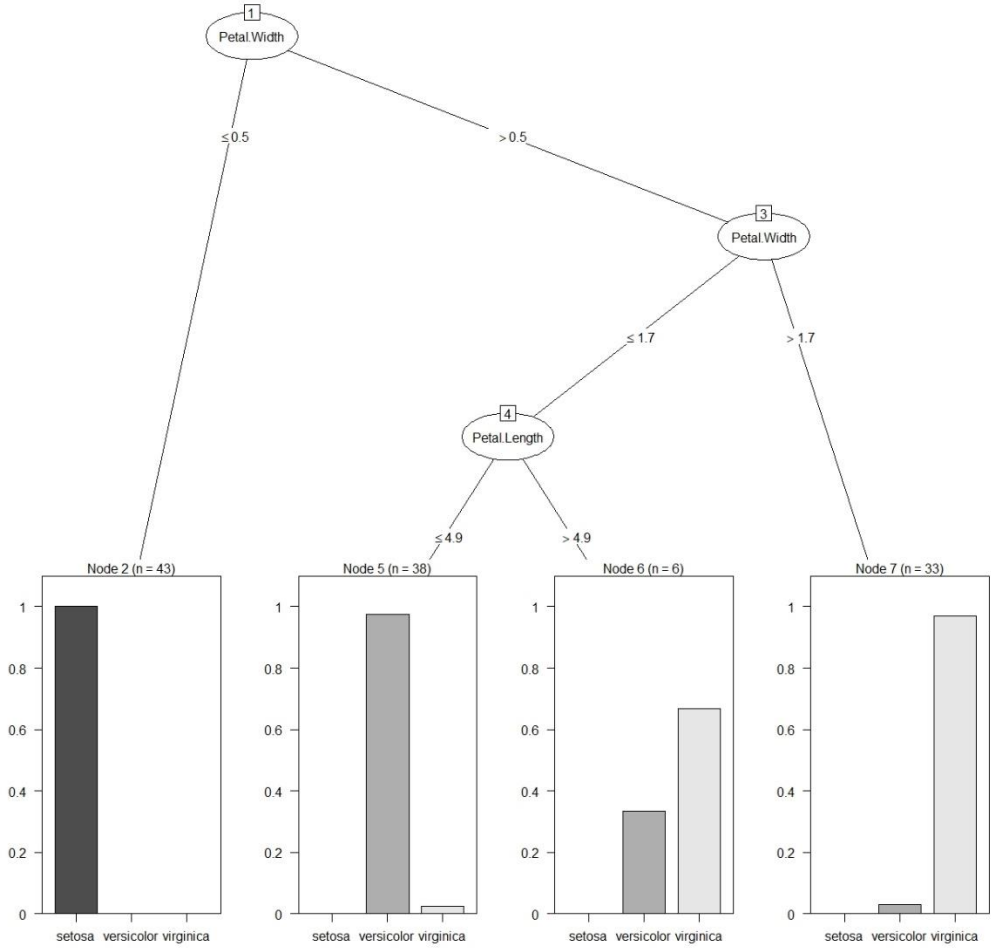

Figure 4: Tree using *C50* library.

Unlike the previous trees using *rpart* and *party* that used petal length as the initial split, the *C50* implementation uses the petal width. Although a different variable is used, the process still separates *setosa* flowers in the first split, indicating there are likely more similarities between *versicolor* and *virginica*. The next split is made using petal width, classifying those greater than 1.7 as *virginica*. The final split is made using petal length again, classifying those less than or equal to 4.9 as *versicolor*.

The *C50* tree generation follows a similar approach to that used in the *party* package. The main difference between the two is that *C50* allows for the boosting of trees. The term “boosting” refers to a process where a tree is fit using the residuals of the previously generated tree. Consequently, this can improve accuracy in large data sets but has little impact on the *iris* data given the small number of observations and variables. The model developed for this problem used five iterations to fit the data.

Part 4: Tree Comparison

To compare the performance between models, each model performed predictions for five randomly sampled testing data sets. Below are the results of the tests.

Test	Tree	Accuracy
1	rpart	90.00%
1	party	96.67%
1	C50	96.67%
2	rpart	96.67%
2	party	96.67%
2	C50	100.00%
3	rpart	96.67%
3	party	100.00%
3	C50	100.00%
4	rpart	100.00%
4	party	100.00%
4	C50	96.67%
5	rpart	90.00%
5	party	90.00%
5	C50	100.00%

Table 1: Results of prediction testing

Tree	Accuracy
rpart	94.67%
party	96.67%
C50	98.67%

Table 2: Average accuracy over all tests.

Based on the testing results of each model, the tree produced using the *C50* library produced the most accurate tree for performing predictions on the *iris* data. Both the *party* tree and *C50* tree produced higher results than the *rpart* tree which logically follows as these two packages use similar tree implementation algorithms. The *C50* model benefits from the boosting process which produces a slightly better accuracy despite using a relatively small data set.

All three models used an initial split to separate the *setosa* flowers from the other two species. While both the *rpart* and *party* libraries used the petal length as the initial split variable, the *C50* model used the petal width. All models used only variables associated with petals, indicating the petal is a better differentiator than sepal among the species.

Conclusion

After performing validation testing, all three models perform relatively well on the iris data. Given the simplicity of the data due to only four predictor variables, the performance of the

models cannot be extrapolated to more complex data. However, the *C50* method of boosting appears to be an optimal approach, as the repeated production of tree models performed better than either the recursive partitioning of *rpart* or they significance association of *party*.

The attributes of a flowers petals for the iris species are a better indicator of the subspecies than the attributes of the sepals. This distinction was shown in Figure 1, as well as the throughout the model process as the tree implementations only used petal variables. Lastly, the *virginica* and *versicolor* species share more similarities than the *setosa* species, indicated by the *setosa* separation on the initial split of all three tree models.