An Introduction to R
R Workshop 3: Decision Trees
MBA542: Baker University
January 22nd, 2020

Contents

1 Introduction

In this applied homework assignment you will learn how to use decision trees with real data. You can find the homework questions at the end of this document.

2 Workshop Procedures

The R code for decision trees that will be used in a case study of people purchasing carseats is presented below.

2.1 Decision Trees

The Carseats dataset is a dataframe with 400 observations on the following 11 variables:

- Sales: unit sales in thousands
- CompPrice: price charged by competitor at each location
- Income: community income level in 1000s of dollars
- Advertising: local ad budget at each location in 1000s of dollars
- Population: regional pop in thousands
- Price: price for car seats at each site
- ShelveLoc: Bad, Good or Medium indicates quality of shelving location
- Age: age level of the population
- Education: ed level at location
- Urban: Yes/No
- US: Yes/No

We want to be able to predict whether a Sale from a customer was likely to be high or low. To conduct this analysis, we will need the ISLR package, the tree package, the and the partykit package.

```
install.packages("ISLR")
install.packages("tree")
install.packages("partykit")
library(ISLR)
library(tree)
library(partykit)
```

Load in the dataset and check the names included. Then use a histogram to look at the continuous variable Sales and determine how you should map it into a binary outcome variable that can be predicted using a decision tree.

```
carseats<-Carseats
names(carseats)
hist(carseats$Sales)
```

We can see from the histogram that a good cutoff point might be $8. As such, we can create a variable, "high" which says "Yes" if Sales is greater than 8 and says "No" if Sales is less than 8. This variable will be a character variable and to use it in the algorithm we want it to be a factor. We can change it with the as.factor() command demonstrated below.

```
carseats$high <- ifelse(carseats$Sales<=8, "No", "Yes")
str(carseats)
carseats$high <- as.factor(carseats$high)
str(carseats)
```

We are now ready to run a decision tree algorithm which makes use of the tree command. The format here is that we assign to the object *model* the algorithm *tree.* Like we did with OLS and Logistic regression, we want to assign to an object the list information that comes from this algorithm. Inside the parentheses you can see the first part is the formula. Here, we are trying to predict the two possible outcomes in the "high" variable. The tilde is the same as in the OLS and Logistic regression. It relates the explained variable to other variables. When we put the dot on the right hand side of the tilde we are relating the explained variable to everything else in the carseats dataset. We want to be careful however, because if we leave in Sales as an explanatory variable, we will have collinearity between the explained and explanatory variables. Thus, we need to remove Sales from the right hand side of the equation, which is what $-Sales$ does.

```
model <- tree(high˜.-Sales, data=carseats)
```

We can now examine various components of this decision tree algorithm. The summary() command will give you the variables used in the construction of the tree. You can also see the number of terminal nodes, the residual mean deviance, and the misclassification error rate. Notice that the misclassification error rate is 9 percent.

```
summary(model)

Classification tree:
tree(formula = high ~ . - Sales, data = carseats)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"  "Advertising"
[7] "Age"         "US"
Number of terminal nodes:  27
Residual mean deviance:  0.4575 = 170.7 / 373
Misclassification error rate: 0.09 = 36 / 400
```

We can also visualize the decision tree. The plot() command will bring up the structure of the decision tree with the appropriate nodes and branches. The text(model) adds in the information about which variables and decision rules were used at the different nodes.

```
plot(model)
text(model, pretty=0)
model
```

2.1.1 Compare to Logistic Regression

We can compare the predictive accuracy of the decision tree to another classification approach that we undertook previously: Logistic Regression. In order to run logistic regression we should convert the factor/character variables to numeric variables.

```
carseats$high.num <- ifelse(carseats$Sales<=8, 0, 1)
carseats$urban.num <- ifelse(carseats$Sales == "Yes", 1, 0)
carseats$us.num <- ifelse(carseats$US == "Yes", 1, 0)
```

We are now ready to run to model. I have assigned the model to the object *model_logistic*; you are free to name your model whatever you choose. Recall that we need to assume the binomial distribution for our distribution from the family of exponentials.

```
model_logistic <- glm(high.num~urban.num+us.num+CompPrice+
Income+Advertising+Population+Price,
                      family = "binomial", data = carseats)

model_logistic %>% summary()
```

The logistic model assigns probabilities to a possible classification. We need to take those probabilities, which range from 0 to 1 and assign a binary classification.

```
classify <- function(probability) ifelse(probability <0.5, 0, 1)
classified_high <- classify(predict(model_logistic, carseats))
```

We can now create a table which shows the observations which were correctly classified (true positives and true negatives) and the observations which were incorrectly classified (false positives and false negatives). Here, you can see that 90 observations of 400 were incorrectly classified for a misclassification error rate of 22.5 percent.

```
table(carseats$high.num, classified_high,
      dnn=c("Data", "Predictions"))
```

2.1.2 Pruning a Decision Tree

It can be helpful to prune the decision tree if there are too many nodes and branches. Pruning can make the decision tree more readable and enhances the visual representation. Additionally, pruning can help increase the applicability of the decision tree algorithm outside of the data used to 'train' the model. This is measured in the variance of the model. The cost to pruning is that the accuracy may fall.

In order to prune the tree, we need to break the data into a training subset and a test subset. In order to do this, and get similar results each time we run the code, we need to set the seed with the set.seed() command. This guarantees that the same observations are chosen each time we choose the 250 observations for the training subset.

```
set.seed(101)
train <- sample(1:nrow(carseats), 250)
```

We need to refit the model with this subset and can re-run the plots. It should look roughly the same.

```
model1 = tree(high~.-Sales, carseats, subset=train)
plot(model1)
text(model1, pretty=0)
```

We now want to take this newly trained model and run it on the test data. This will be all the data from carseats minus the train subset.

```
tree.pred = predict(model1, carseats[-train,], type="class")
```

We now want to evaluate the model using a misclassification table. You should notice that the tree performs much worse on the test data than we saw before. This is an indication that the model was over-fitted.

```
with(carseats[-train,], table(tree.pred, high))
```

We can choose to prune the tree optimally using a process called cross-validation. This allows us to re-sample the data and choose across a range of models which optimize the out-of-sample predictive power of the decision algorithm. On the plot, the x-axis shows the best number of cross-validation samples to use. It appears that 12 is the optimal number.

```
cv.carseats <- cv.tree(model1, FUN = prune.misclass)
cv.carseats
plot(cv.carseats)
```

Here, we use a function called prune.misclass(). The inside of the parentheses specifies which decision tree is to be pruned (in our case, model1) and what the best number of cross-validations is (in our case, 12). This will create a new and purportedly improved decision tree which here will be called prune.model1.

```
prune.model1 <- prune.misclass(model1, best = 12)
plot(prune.model1)
text(prune.model1, pretty=0)
```

We now want to test this new and improved decision tree on the test dataset. Once you create the table, you should see that the pruning did not dramatically decrease the predictive power of the algorithm. As such, while we increased the efficiency of the tree.

```
tree.pred = predict(model1, carseats[-train,], type="class")
with(carseats[-train,], table(tree.pred, high))
```

3 End of Class Practice Questions

Use the Social Network Ad data from Moodle to answer each of these questions.

1. Create a numeric gender variable. Use the table command to get a count for each category (male and female in this data).

2. Create a factor variable for the purchased variable. Use the table command to get a count for each category (have purchased and have not purchased).

3. Run a decision tree algorithm predicting whether someone made the purchase and report the algorithm's misclassification error rate. Hint: in the equation, specify the three possible explanatory variables explicitly.

4. Create the decision tree plot with text.

5. Compare the accuracy of the decision tree to the predictive accuracy of a logistic regression.

6. Generate a training dataset of 250 observations. Run a decision tree algorithm on this dataset. Verify this decision tree is accurate by running the algorithm on the test dataset and reporting the confusion matrix (number of true positives and negatives as well as the number of false positives and negatives).

4 Homework Questions

Use the titanic dataset to answer each of these questions. You can obtain the titanic dataset from R:

```
install.packages("titanic")
library(titanic)
data("titanic_train")
```

1. Create a numeric gender variable. Use the table command to get a count for each category (male and female in this data).

2. Create a factor variable for the survived variable. Use the table command to get a count for each category (survived and did not survive).

3. Run a decision tree algorithm predicting whether someone survived and report the algorithm's misclassification error rate. Your explained variable will be the factor survived variable and your explanatory variables will be Pclass, numeric sex, age, SibSp, Parch, and Fare.

4. Create a decision tree plot with text. Include the plot in your homework submission.

5. Compare the accuracy of the decision tree to the predictive accuracy of a logistic regression. Use the same variables in your logistic regression.

6. Apply your algorithm from that you created from the training dataset to the test dataset. Report the confusion matrix numbers to see if predictive accuracy decreased.