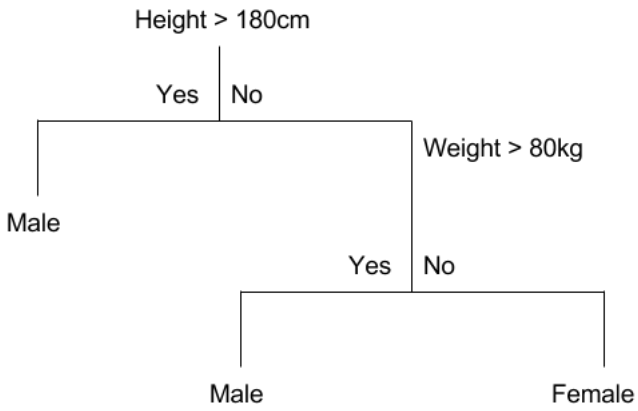
post02-marc-mansour

Predicting the Best Candies with Regression and Classification Trees

By Marc Mansour

Introduction

One aspect of data analysis that we have yet to explore this semester is the issue of prediction. Through various methods of modeling, data analysts are enabled to predict the value of a 'response' variable based on a series of given 'explanatory' variables. One of the late Cal statistician Leo Breiman's biggest research contributions revolved around two related methods of prediction modeling: regression trees and classification trees. Regression trees lead you step-by-step to a predicted response based on recursive partitioning. Classification trees are very essentially the same as regression trees; however, the response that classification trees predict is a binary 0 or 1. In other words, regression trees are a means of estimation whereas classification trees simply answer a yes or no question.



The image above is a very simple example of a classification tree. This tree uses an individual's height and weight in order to predict their gender, either male or female. In order to predict an individual's gender using this tree you would first check to see if the individual is taller than 180cm. If so, then that individual is predicted to be a male. If the individual's height is less than 180cm, then you must look to see if that individual weighs more than 80kg. If so, the tree predicts that the individual is a male. If the individual's weight is less than 80kg, then that individual is predicted to be a female.

Throughout this piece, we will use classification trees and regression trees in order to analyze the best candies on the market with data provided from fivethirtyeight's public GitHub repository. Our first tree will predict the success of a candy using regression to estimate the candy's 'winpercent', or the percentage of times this candy was picked over a competing candy. The next two tress will use classification to predict whether or not a candy contains chocolate or is fruity based on other attributes of the candy. Finally, we will implement some optimization methods to show you how to make the most effective trees for prediction.

Installing the rpart package

```
# Install the rpart package, if you don't already have it
#install.packages('rpart')

# Load the rpart package to use in your current R session
library(rpart)
```

The above lines of code will enable you to a) install the rpart package if you do not already have it and b) load the rpart package to your current session in R. The rpart package was developed by Terry Therneau, Beth Atkinson, and Brian Ripley in order to implement the functionalities described in a 1984 book Leo Brieman and a host of other statisticians wrote titled *Classification and Regression Trees*. It's main function 'rpart' is named after the recursive partitioning method the authors used as the basis for these tree models. Version 4.1-11 of the rpart package, which we will be working with today, was released earlier this year on March 12th.

The Data

```
# Load the candy data from fivethirtyeight's github repository
csv <- "https://raw.githubusercontent.com/fivethirtyeight/data/master/candy-power-ranking/candy-data.csv"
download.file(url = csv, destfile = '../data/candy-data.csv')
destination <- '../data/candy-data.csv'
candy <- read.csv(destination)
```

As previously mentioned, this data is from fivethirtyeight's public GitHub repository and was featured in a Halloween-inspired candy power rankings post earlier this year. The variables within this dataset will be explained as they are used throughout this analysis; however, I have also created a data dictionary for this dataset that can be found in the data folder.

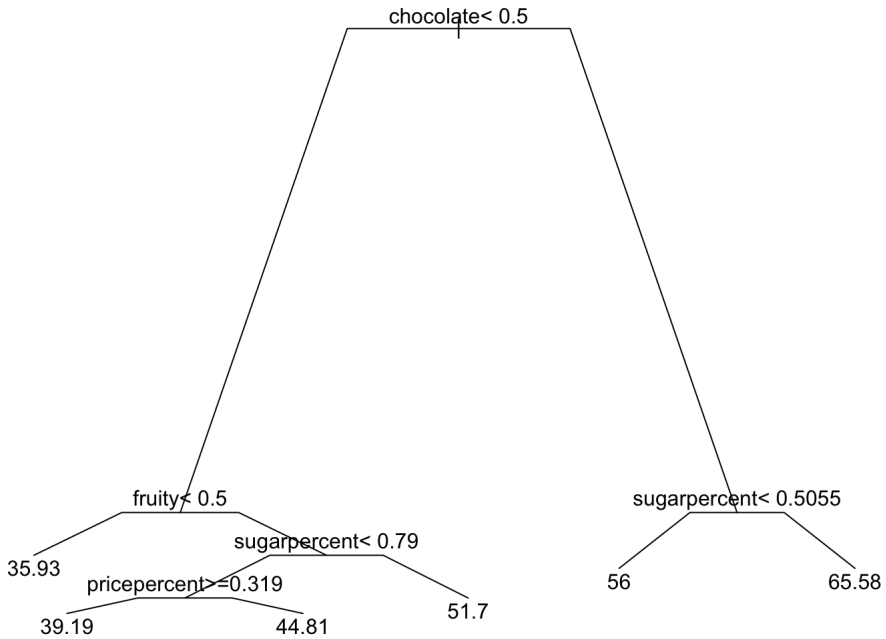
Regression Trees

```
# Create a regression tree that predicts the 'winpercent' of a given candy
reg_tree <- rpart(winpercent ~
              # the variables listed below are the explanatory variables
              chocolate + fruity + sugarpercent + pricepercent,
              # specify the data frame we are working with
              data = candy,
              # answering the method argument with 'anova' makes a regression tree
              method = 'anova')

# Export the regression tree to a png file within the images folder
png(filename = "../images/regression-tree.png")
par(mar = rep(0.2, 4))
plot(reg_tree, branch = 0.4, compress = TRUE)
text(reg_tree)
dev.off()
```

```
## quartz_off_screen
##                 2
```

```
# Display the regression tree for the 'winpercent' predictions
par(mar = rep(0.2, 4))
plot(reg_tree, branch = 0.4, compress = TRUE)
text(reg_tree)
```



The above regression tree uses whether or not the candy is chocolate, whether or not the candy is fruity, its relative price, and its relative sugar content to predict the candy's winpercent. Note that the predicted winpercent values according to the model are all listed below in increasing order from left to right. We start by reading the tree at the top and if the logical statement provided above is true for the given candy, then you continue on down the righthand branch of the tree. If the statement is false for you given candy, then you proceed down the lefthand branch of

the tree. For instance, according to this regression tree, if the given candy is not chocolate (has a 'chocolate' value of 0), then you proceed down the lefthand branch of the tree and determine whether or not it is fruity. If the candy is indeed fruity, then you go down the righthand branch of the tree and check to see if it's 'sugarpercent' is less than 0.79. If it's sugarpercent is NOT less than 0.79, then you proceed to the lefthand branch of the tree and see if the candy's 'pricepercent' is greater than or equal to 0.319. If so, the candy's predicted 'winpercent' is 44.81. If not, the candy's predicted 'winpercent' is 39.19. A png of this regression tree can be found in the images folder.

There are several takeaways that we can gather about whether or not these attributes seem to contribute positively or negatively to the percentage of times a given candy is chosen over others. For example, we can note that the chocolate candies have universally higher predicted 'winpercent' values with a minimum of 56 compared to a maximum of 51.7 for the non-chocolate candies. On top of that, non-chocolate candies that are not classified as fruity are even less valued with the lowest 'winpercent' prediction, a meager 35.93. Additionally, sugar seems to contribute positively to the predicted 'winpercent' values according to this model. Finally, this model predicts that more expensive fruity candies are predicted to be chosen over competitors more than the cheaper fruity candies.

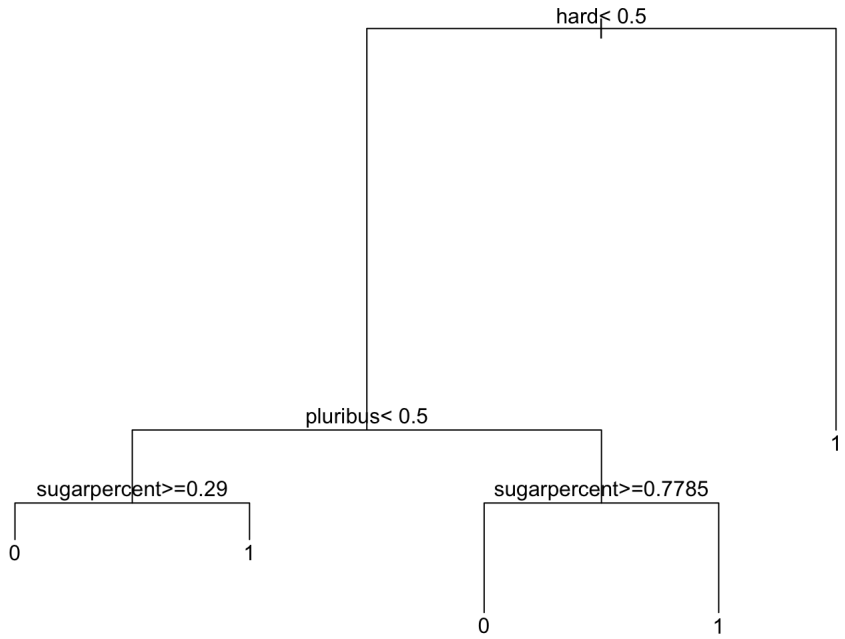
Classification Trees

```
# Create a classification tree that predicts whether or not a candy is fruity
fruity_tree <- rpart(as.factor(fruity) ~
               # the variables listed below are the explanatory variables
               sugarpercent + hard + pluribus,
               # specify the data frame we are working with
               data = candy,
               # answering the method argument with 'class' makes a classification tree
               method = 'class')

# Export the classification tree to a png file within the images folder
png(filename = "../images/fruity-classification-tree.png")
par(mar = rep(0.2, 4))
plot(fruity_tree)
text(fruity_tree)
dev.off()
```

```
## quartz_off_screen
##                 2
```

```
# Display the classification tree
par(mar = rep(0.2, 4))
plot(fruity_tree, branch = 1.0, compress = TRUE)
text(fruity_tree)
```



This classification tree uses the variables sugarpercent, hard, and pluribus to determine whether or not the candy in question is fruity. First, if the candy is hard then it is fruity right off of the bat. If the given candy is not hard, then you determine whether or not it is pluribus (if it comes with many pieces). If the candy is pluribus, then you look it's 'sugarpercent'. If the candy is as sugary or more sugary than 77.85% of candies, then it is fruity. If not, the candy is not fruity. However, if the candy comes in a single piece and is not pluribus, then you look at it's 'sugarpercent'. If the candy has a 'sugarpercent' value greater than or equal to 0.29, then it is fruity. If not, then it is not fruity. A png of this classification tree can be found in the images folder.

From this classification tree, we can intuitively affirm that hard candies are typically fruity. In addition, very sugary candies that come in many pieces and moderately sugary candies that come in one piece are also fruity according to this tree. We can use these trees as models and obtain predictions from them using the predict() function while specifying the tree we made using rpart and the data frame of candies that you want to use to predict whether or not they are fruity or their 'winpercent' value.

Complexity Parameters (CP)

Now we will discuss an argument within the rpart function called cp that allows us to optimize our model by specifying the best complexity parameter value. We will start by making a classification tree below that predicts whether or not a candy is chocolate solely based of its relative

sugar content and the percentage of times the candy is chosen over others. Making a more focused model like this can uncover more specific insights on the degree to which each explanatory contributes to the response. After creating this model with rpart, we check the errors for varying complexity parameter values of the tree.

```
# Create a classification tree that predicts whether or not a candy has chocolate
choco_tree <- rpart(as.factor(chocolate) ~
               pricepercent + winpercent,
               data = candy,
               method = 'class')

# Check the complexity parameters with printcp()
printcp(choco_tree)
```

```
##
## Classification tree:
## rpart(formula = as.factor(chocolate) ~ pricepercent + winpercent,
##     data = candy, method = "class")
##
## Variables actually used in tree construction:
## [1] pricepercent winpercent
##
## Root node error: 37/85 = 0.43529
##
## n= 85
##
##         CP nsplit rel error  xerror     xstd
## 1 0.56757      0   1.00000 1.00000 0.123541
## 2 0.10811      1   0.43243 0.59459 0.109137
## 3 0.01000      2   0.32432 0.45946 0.099671
```

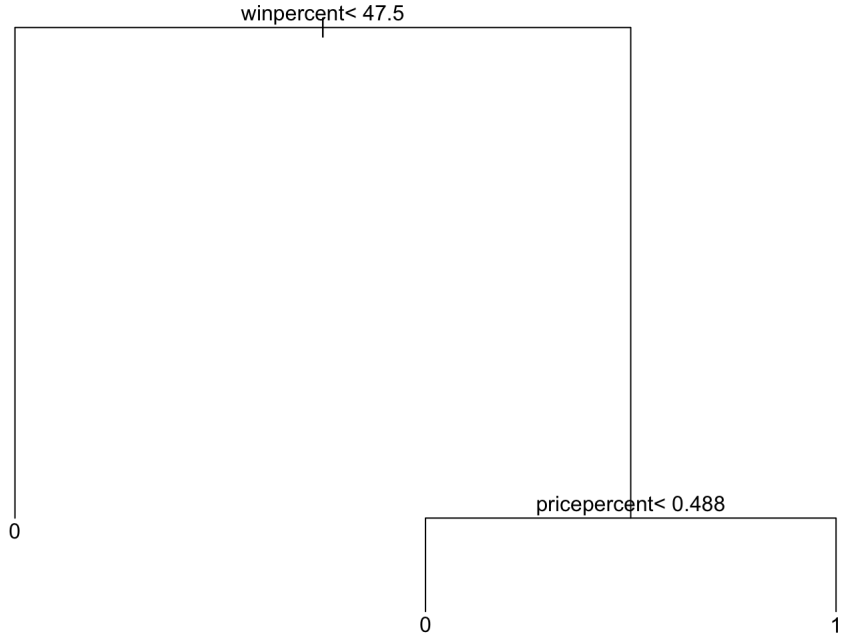
As you can see from the above output, there are 3 CP values listed: 0.56757, 0.10811, and 0.01000. The latter two CP values are tied for the lowest error at 0.56757, but the 0.01000 CP values has a lower relative error. Therefore, change our previous classification model by specifying a cp value of 0.01000.

```
# Create a classification tree that predicts whether or not a candy has chocolate
choco_tree <- rpart(as.factor(chocolate) ~
               # the variables listed below are the explanatory variables
               winpercent + sugarpercent + pricepercent,
               # specify the data frame we are working with
               data = candy,
               # answering the method argument with 'class' makes a classification tree
               method = 'class',
               # specify a cp value of 0.01000
               cp = 0.01)

# Export the classification tree to a png file within the images folder
png(filename = "../images/chocolate-classification-tree.png")
par(mar = rep(0.2, 4))
plot(choco_tree)
text(choco_tree)
dev.off()
```

```
## quartz_off_screen
##                 2
```

```
# Display the classification tree
par(mar = rep(0.2, 4))
plot(choco_tree)
text(choco_tree)
```



This optimized classification tree is very simple to read. If the candy is chosen over its competitors less than 47.5% of the time, then you look at it's relative price. If the candy is pricier than no more than 48.8% of candies, then the candy contains chocolate. If not, the candy does not contain chocolate. However, if the candy is chosen over its competitors more than 47.5% of the time, then it is chocolate. According to this model, chocolate candies seem to be favored more and priced lower. An image of this classification tree can also be found in the images folder as a png.

Looking Forward

Through our analysis of this seemingly trivial candy dataset, we have experimented with some of the main features of the rpart package. Through our implementation of the rpart package, we have explored Leo Breiman's tree-based models. We first learned how to create regression trees to estimate a given response variable based on specified explanatory variables. Then we made a classification tree-based model to predict whether or not a given candy is fruity. Finally, we made another classification tree-based model that was optimized with a more efficient complexity parameter. Each of the tree-based models are provided in png files in the images folder. The dataset used and a data dictionary are provided in the data folder. For additional information on tree-based models, the rpart package, recursive partitioning, and complexity parameters please see the references listed below. Finally, we must keep in mind that these tree-based models are only two one of the many other means of making prediction models in R. Consequently, there are plenty of other methods to be explored!

References

- Official R Project Documentation on rpart
- R Documentation of rpart by author Brian Ripley
- Trees with the rpart package by Christoph Molnar of R-bloggers
- Tree-Based Models by Quick-R
- Classification and Regression Trees by Leo Pekelis of Stanford University
- An Introduction to Recursive Partitioning Using the RPART Routines
- The Ultimate Halloween Candy Power Ranking by fivethirtyeight
- Decision Trees in R by Arpan Gupta of IIT