Final Project Presentation

Mitchell Everetts

4/18/2022

Introduction

There's hundreds of brands producing thousands of different workout supplemental products. With so many items available in the industry, the consumer naturally asks questions seen in every industry. Questions like "What is the best bang for my buck" or "Which flavors are consistently the best" pop up frequently. Some of these are more easily answered, but industry heads and product producers ask some harder hitting questions like "What variables could determine the rating of a given product" or "Can you determine a products category based solely off the nutritional information?" It's these questions this project has sought out to try and answer.

This project seeks to tackle some of those questions both simple and complex.

Initialization

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ISLR)
library(rpart)

df <- read_csv("fullsupp.csv")
```

```
## Warning: One or more parsing issues, see `problems()` for details
```

```
## Rows: 2719 Columns: 2837
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (832): Category, Name, Brand, Size, Flavor, Rating, Price, PricePerServ...
## dbl (2005): Servings, FlavorRating, Reviews, VITAMIN D 0MCG, BRANCHED-CHAIN ...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Inital EDA

With the data loaded up, before any analysis can be made, It's crucial to do some preliminary EDA.

```
head(df)
```

```
## # A tibble: 6 x 2,837
##   Category  Name   Brand Size  Flavor Servings Rating FlavorRating Reviews Price
##   <chr>     <chr>  <chr> <chr> <chr>     <dbl> <chr>         <dbl>   <dbl> <chr>
## 1 WHEY PRO~ whey ~ Opti~ 1 Lb. Delic~       14 9.3             8.1   10031 $19.~
## 2 WHEY PRO~ whey ~ Opti~ 1 Lb. Doubl~       14 9.3             8.5   10031 $19.~
## 3 WHEY PRO~ whey ~ Opti~ 1 Lb. Vanil~       14 9.3             7.8   10031 $19.~
## 4 WHEY PRO~ whey ~ Opti~ 2 Lb~ Banan~       29 9.3             8.4   10031 $29.~
## 5 WHEY PRO~ whey ~ Opti~ 2 Lb~ Birth~       27 9.3             7.4   10031 $29.~
## 6 WHEY PRO~ whey ~ Opti~ 2 Lb~ Cake ~       28 9.3             5     10031 $29.~
## # ... with 2,827 more variables: PricePerServing <chr>, CALORIES <chr>,
## #   TOTAL FAT <chr>, CHOLESTEROL <chr>, SODIUM <chr>, TOTAL CARBOHYDRATE <chr>,
## #   PROTEIN <chr>, CALCIUM <chr>, POTASSIUM <chr>, DIETARY FIBER <chr>,
## #   IRON <chr>, TOTAL SUGARS <chr>, VITAMIN D <chr>, VITAMIN D 0MCG <dbl>,
## #   PHOSPHORUS <chr>, MAGNESIUM <chr>, VITAMIN A <chr>, VITAMIN C <chr>,
## #   VITAMIN A (AS BETA-CAROTENE) <chr>, VITAMIN C (AS ASCORBIC ACID) <chr>,
## #   VITAMIN D (AS CHOLECALCIFEROL) <chr>, ...
```

Analyzing the head of the data frame shows some glaring issues. Firstly, there's nearly 3 thousand columns. The documentation shows that every time a product ingredient is mentioned in a different context, a new column is created (such as a different patented "power blend" containing the same vitamin mixes under different names). As such any information beyond column 22 could potentially be useful, but would require hours and hours of data manipulation and wrangling to shed any insight.

A note from the project authors; this was especially obnoxious, as important ingredients of pre-workout such as Creatine and Beta-Alanine were not able to be used because of the above reason

Inital Data Wranglin'

From the inital glimpse at the head, we see a few different problems. Firstly there's too much fluff extra data with little value to us (at least at the moment). The columns aren't consistently named, and the features aren't classified appropriately, almost everything is a character but almost everything should be either a factor or a numeric.

```
df_clean <- df[1:22]
#removing the nutritional value stuff, as expressed earlier far too little info to keep columns in gene

df_clean <- df_clean%>%
  mutate(Rating = as.numeric(Rating))%>%
  mutate(Category = as.factor(Category))%>%
  mutate(Brand = as.factor(Brand))%>%
  drop_na()
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

```
#the two factors and singular easy numeric change are done first before dropping NA's

names(df_clean)[names(df_clean) == 'PricePerServing'] <- 'PPS'
names(df_clean)[names(df_clean) == 'CALORIES'] <- 'Calories'
names(df_clean)[names(df_clean) == 'TOTAL FAT'] <- 'Fats'
names(df_clean)[names(df_clean) == 'CHOLESTEROL'] <- 'Cholesterol'
names(df_clean)[names(df_clean) == 'SODIUM'] <- 'Sodium'
```

```
names(df_clean)[names(df_clean) == 'TOTAL CARBOHYDRATE'] <- 'Carbs'
names(df_clean)[names(df_clean) == 'PROTEIN'] <- 'Protein'
names(df_clean)[names(df_clean) == 'CALCIUM'] <- 'Calcium'
names(df_clean)[names(df_clean) == 'POTASSIUM'] <- 'Potassium'
names(df_clean)[names(df_clean) == 'DIETARY FIBER'] <- 'Fiber'
names(df_clean)[names(df_clean) == 'IRON'] <- 'Iron'
names(df_clean)[names(df_clean) == 'TOTAL SUGARS'] <- 'Sugars'
#all wonky names are changed to be consistent throughout the data frame

df_clean <- df_clean%>%
  mutate(Price = parse_number(Price))%>%
  mutate(PPS = parse_number(PPS))%>%
  mutate(Calories = parse_number(Calories))%>%
  mutate(Fats = parse_number(Fats))%>%
  mutate(Cholesterol = parse_number(Cholesterol))%>%
  mutate(Sodium = parse_number(Sodium))%>%
  mutate(Carbs = parse_number(Carbs))%>%
  mutate(Protein = parse_number(Protein))%>%
  mutate(Calcium = parse_number(Calcium))%>%
  mutate(Potassium = parse_number(Potassium))%>%
  mutate(Fiber = parse_number(Fiber))%>%
  mutate(Iron = parse_number(Iron))%>%
  mutate(Sugars = parse_number(Sugars))
#all character columns are converted to numerics via number parsing

df_clean[4] <- NULL
#size would be a useful column were it not for the inconsistency from product to product (lb, mg, g, SE

df_clean[2] <- NULL
#name was just a character duplicate of the category column

head(df_clean)
```

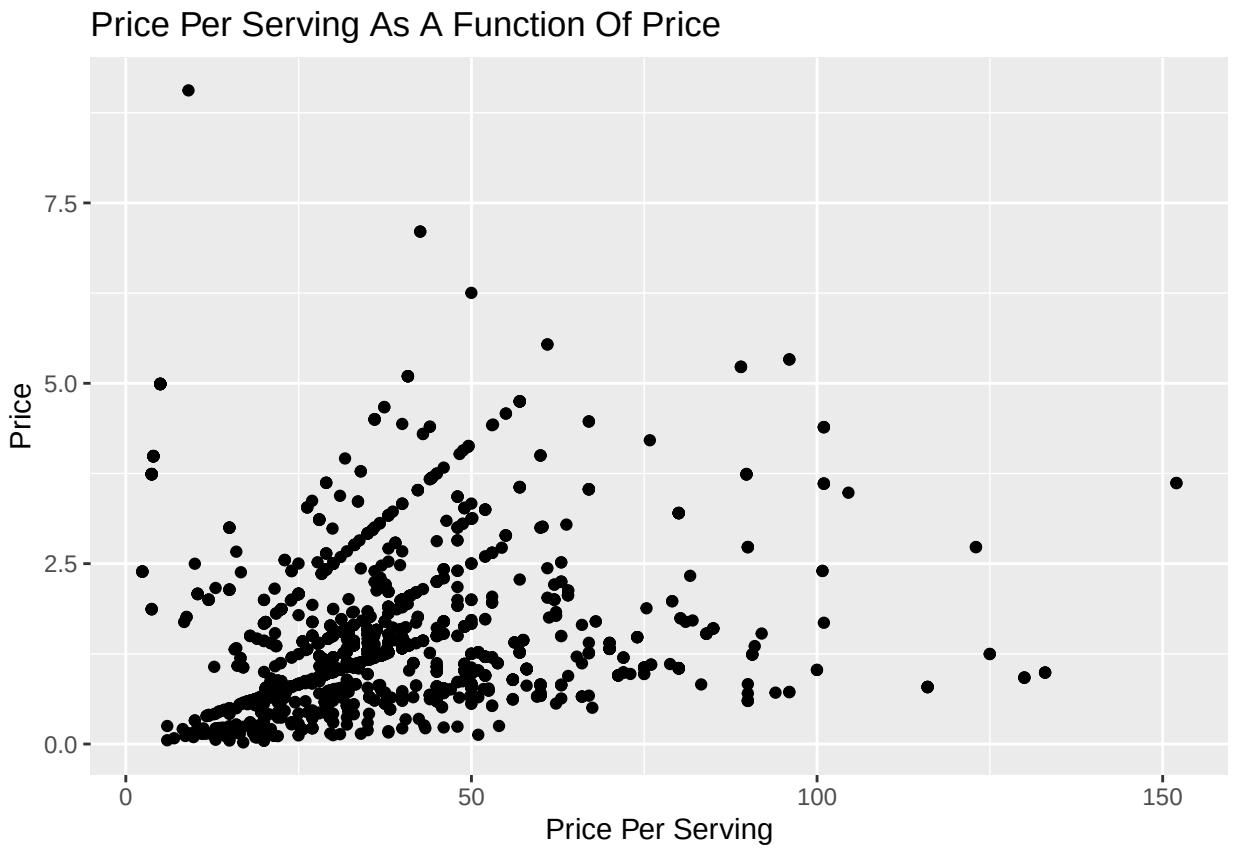
```
## # A tibble: 6 x 20
##   Category     Brand   Flavor   Servings Rating FlavorRating Reviews Price   PPS
##   <fct>        <fct>   <chr>       <dbl>  <dbl>        <dbl>   <dbl> <dbl> <dbl>
## 1 WHEY PROTEIN Optimu~ Delicio~       14    9.3          8.1   10031  20.0  1.43
## 2 WHEY PROTEIN Optimu~ Double ~       14    9.3          8.5   10031  20.0  1.43
## 3 WHEY PROTEIN Optimu~ Vanilla~       14    9.3          7.8   10031  20.0  1.43
## 4 WHEY PROTEIN Optimu~ Banana ~       29    9.3          8.4   10031  30.0  1.03
## 5 WHEY PROTEIN Optimu~ Birthda~       27    9.3          7.4   10031  30.0  1.03
## 6 WHEY PROTEIN Optimu~ Cake Do~       28    9.3          5     10031  30.0  1.03
## # ... with 11 more variables: Calories <dbl>, Fats <dbl>, Cholesterol <dbl>,
## #   Sodium <dbl>, Carbs <dbl>, Protein <dbl>, Calcium <dbl>, Potassium <dbl>,
## #   Fiber <dbl>, Iron <dbl>, Sugars <dbl>
```

EDA Visualized

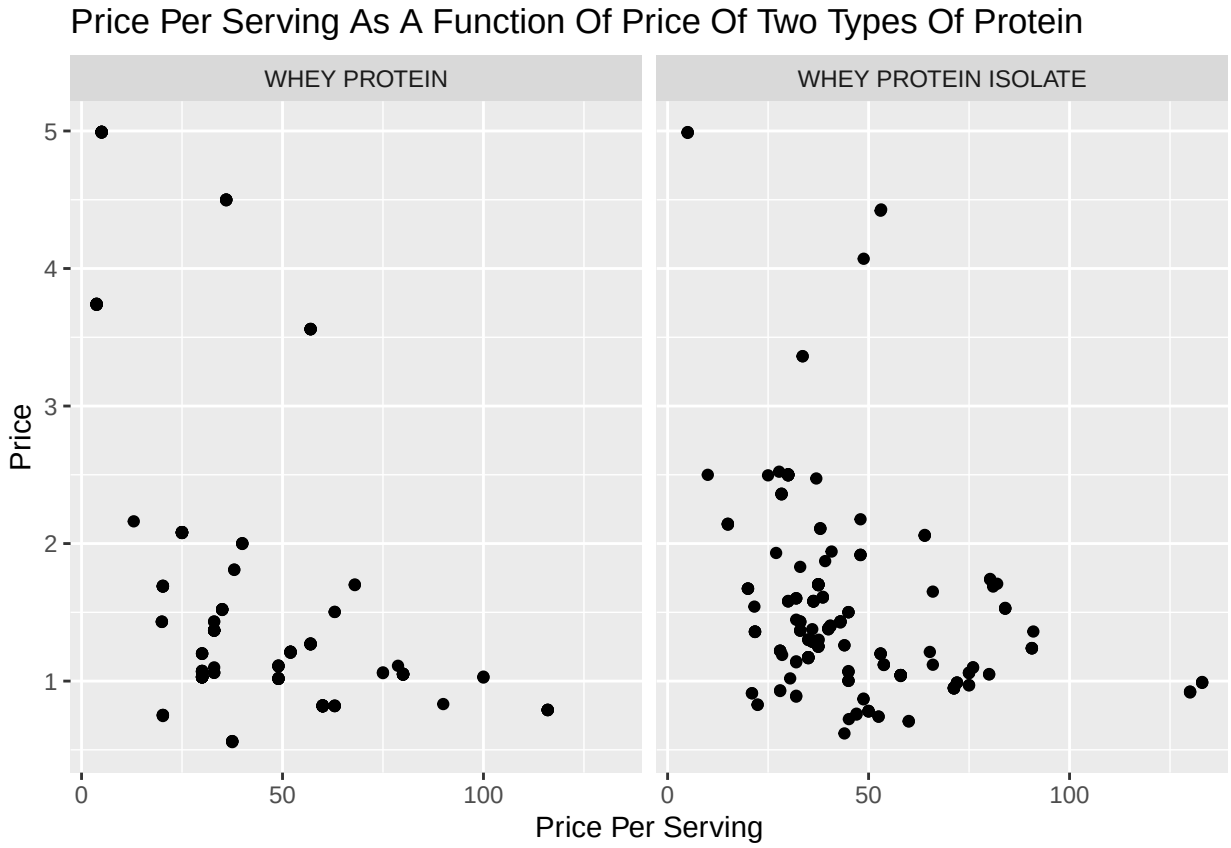
Now with a clean data frame, we can begin to do some proper EDA

```
df_brot <- df_clean%>%
  filter(Category == "WHEY PROTEIN ISOLATE" | Category == "WHEY PROTEIN")%>%
  mutate(Category = as.character(Category))%>%
  mutate(Category = as.factor(Category))
```

```
#quick dataset that only has Whey Protein and Whey Protein Isolate

ggplot(df_clean, aes(Price, PPS))+
  geom_jitter()+
  ggtitle("Price Per Serving As A Function Of Price") +
  xlab("Price Per Serving") + ylab("Price")
```

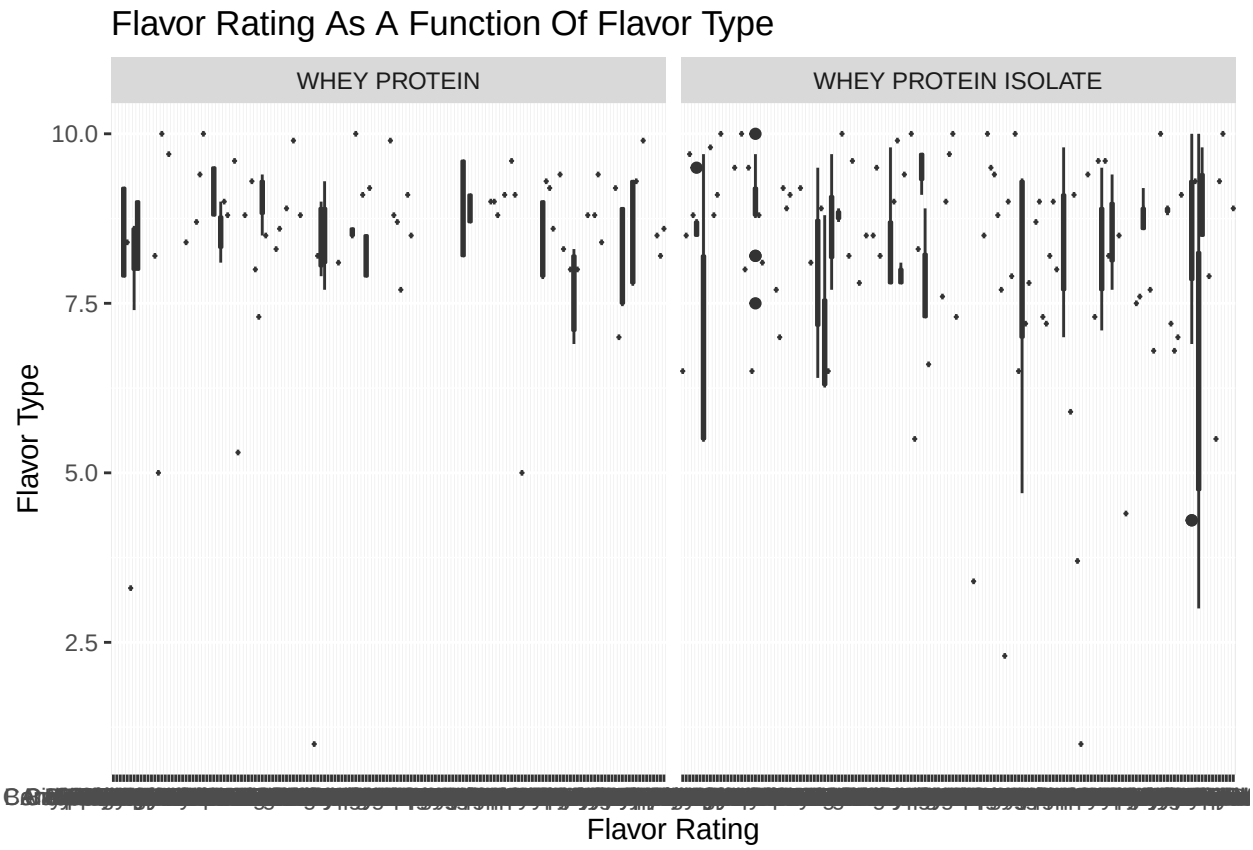



```
ggplot(df_brot, aes(Price, PPS))+
  geom_jitter()+
  facet_wrap(~Category)+
  ggtitle("Price Per Serving As A Function Of Price Of Two Types Of Protein") +
  xlab("Price Per Serving") + ylab("Price")
```



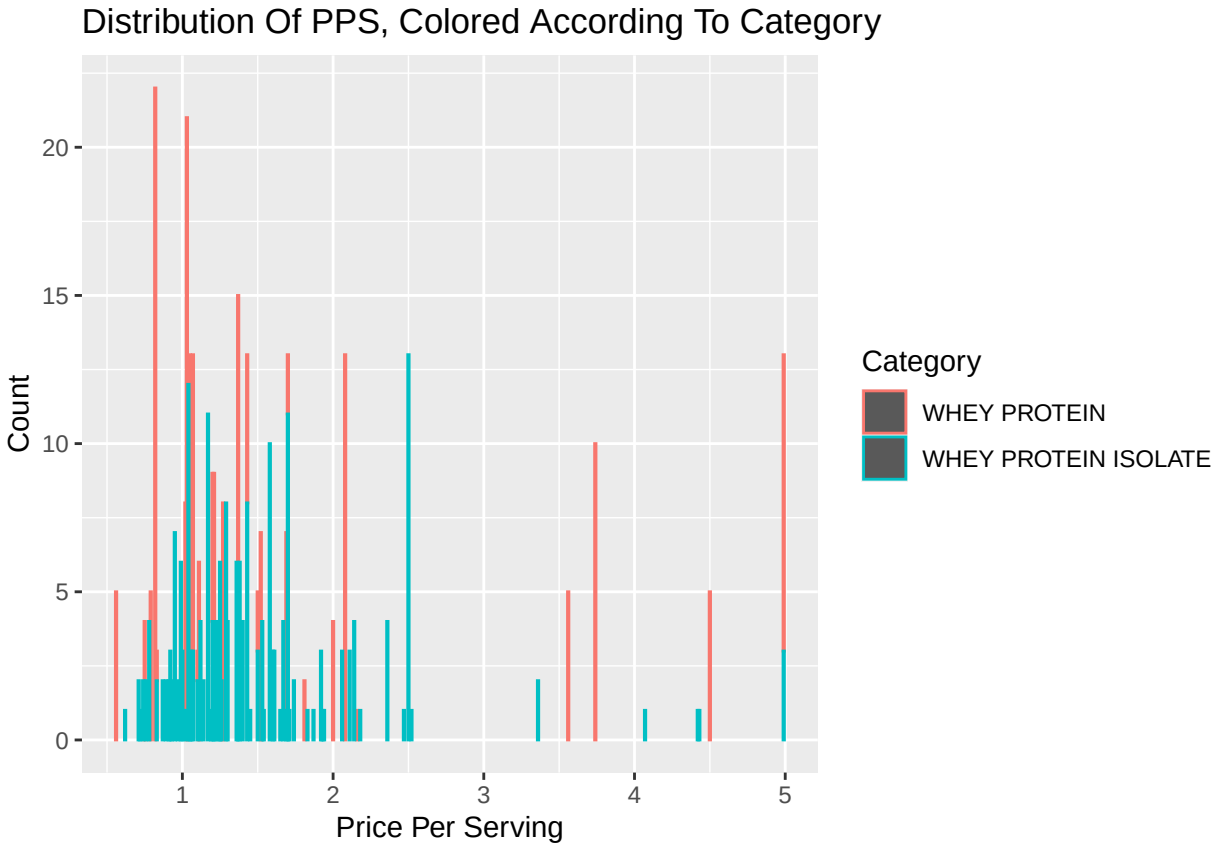
We can see through the above graphs that in general, when looking at PPS (Price Per Serving) as a function of Price, we see a positive relationship. This makes sense initially, the more something costs, the more expensive the serving cost is going to be. However when breaking apart this graph into every category, we see that Whey Protein, and Whey Protein Isolate actually seem to have the reverse of that relation, as the product gets more expensive the PPS decreases.

```
ggplot(df_brot, aes(Flavor, FlavorRating))+
  geom_boxplot()+
  facet_wrap(~Category)+
  ggtitle("Flavor Rating As A Function Of Flavor Type") +
  xlab("Flavor Rating") + ylab("Flavor Type")
```



This graph isn't extremely readable, but the purpose of it is just to see if certain flavors have a consistently high/low flavor rating, which is certainly found. With over a hundred flavors found in just the two different proteins, some data trimming and models will show which flavors are consistently good, bad, and mixed.

```
ggplot(df_brot, aes(PPS, color=Category))+
  geom_bar()+
  ggtitle("Distribution Of PPS, Colored According To Category") +
  xlab("Price Per Serving") + ylab("Count")
```



Continuing to analyze the difference in Whey Protein, and Whey Protein Isolate, Plotting the bar chart of PPS using colored category to reveal where the two categories are distributed. There seems to be a heavier prevalence of Whey Protein Isolate in the more expensive PPS range, whereas Whey Protein is evenly spread out with a very heavy concentration in the cheaper PPS range.

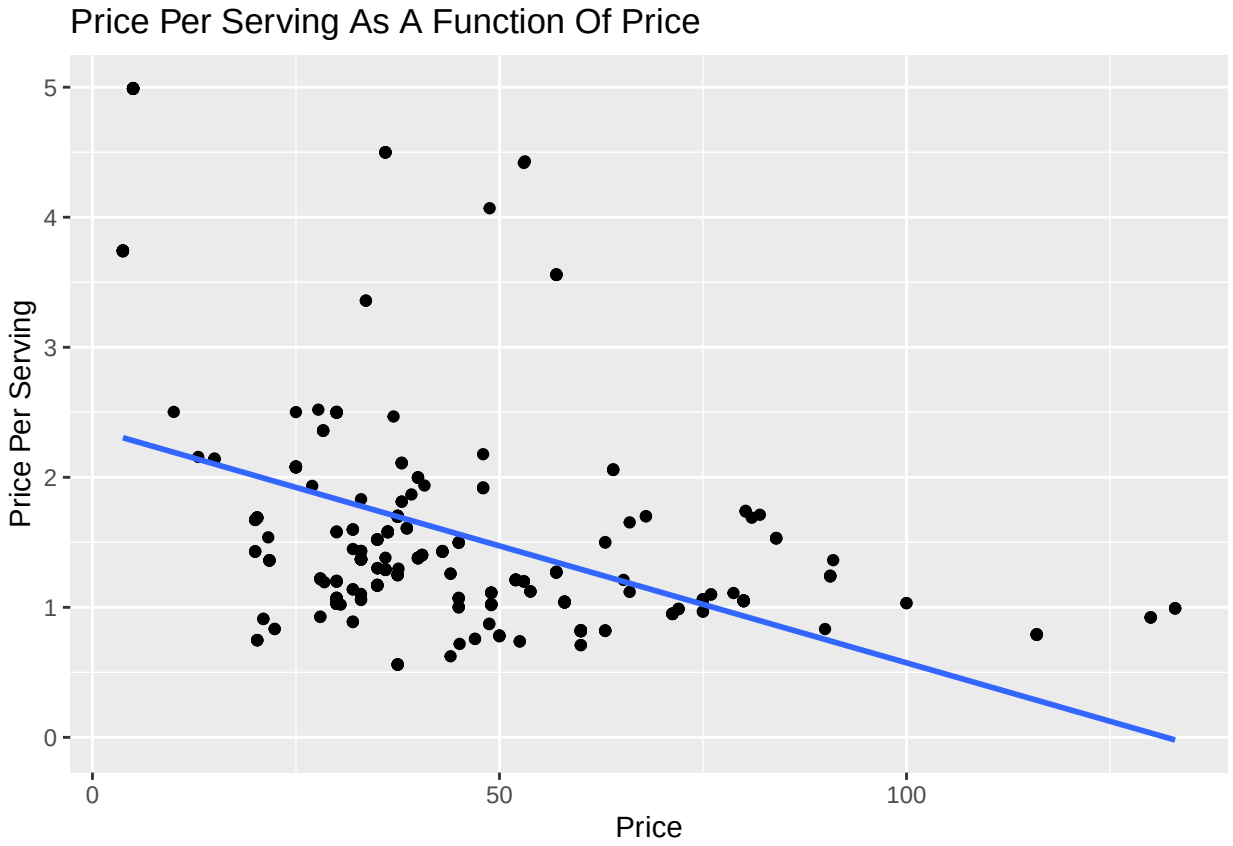
Methods

Linear Modeling

With the EDA done, there seem to be some interesting relationships worth exploring via model exploration. From the first EDA graphs, there appears to be some connection between the price and the PPS, suggesting something like with the spendier a product gets the more affordable the individual serving is.

```
mod1 = lm(PPS ~ Price, data = df_brot)

ggplot(df_brot, aes(x = Price, y = PPS))+
  geom_jitter()+
  geom_smooth(se = FALSE, method = "lm")+
  ggtitle("Price Per Serving As A Function Of Price") +
  xlab("Price") + ylab("Price Per Serving")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
summary(mod1)
```

```
##
## Call:
## lm(formula = PPS ~ Price, data = df_brot)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2576 -0.4727 -0.2266  0.1578  3.0137
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.371692   0.086942   27.28   <2e-16 ***
## Price       -0.017986   0.001747  -10.30   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.852 on 435 degrees of freedom
## Multiple R-squared:  0.196,  Adjusted R-squared:  0.1942
## F-statistic: 106.1 on 1 and 435 DF,  p-value: < 2.2e-16
```

```
anova(mod1)
```

```
## Analysis of Variance Table
##
## Response: PPS
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## Price       1  76.994  76.994  106.06 < 2.2e-16 ***
```

```
## Residuals 435 315.799   0.726
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

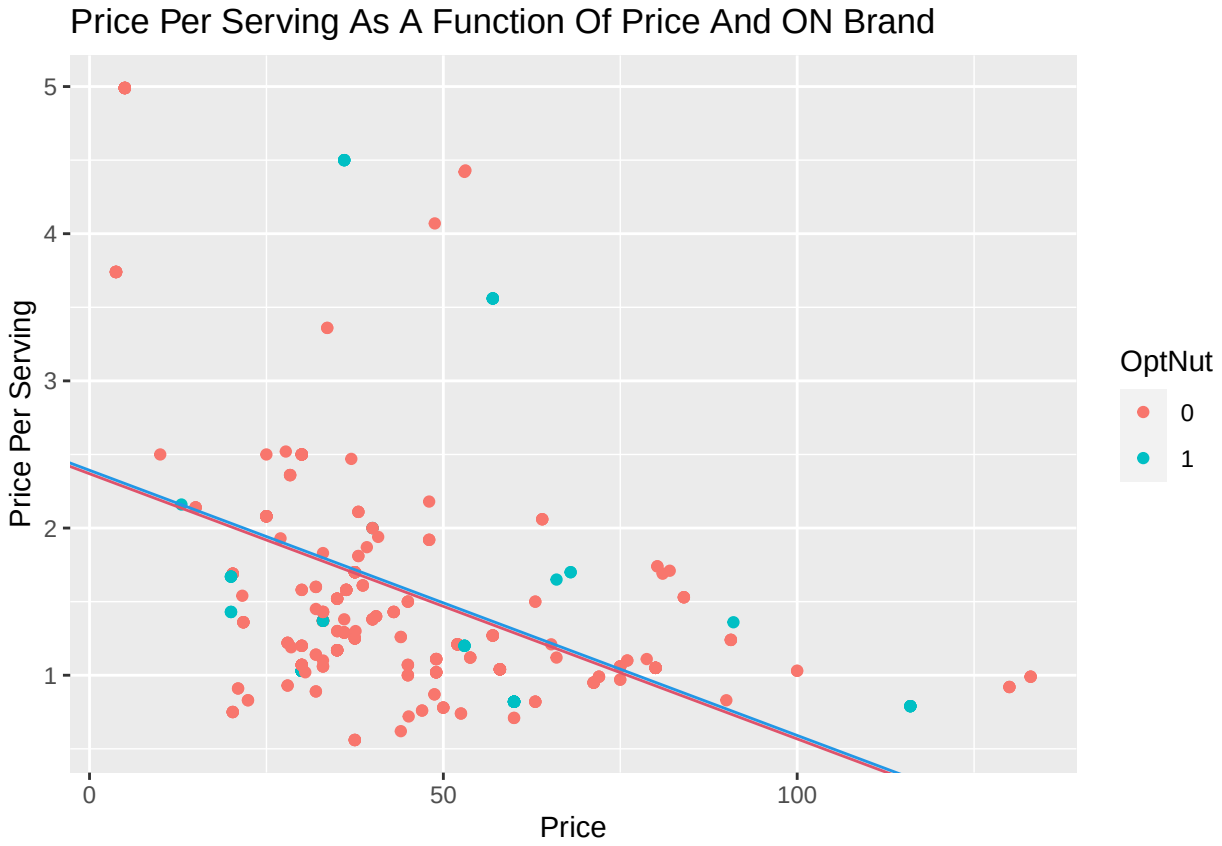
The model backs-up this theory with both the linear model and the ANOVA table showing extreme significance between PPS as a response to the explanatory variable Price. This makes sense in the context discussed earlier, as buying bulk products (especially protein powders that this specific graph is about).

To explore this question further, the model would need additional variables to get a better approximation of if your "bang" for your "buck" does increase or decrease. Brand could help give a better idea, however the Brand category contains over a 100 separate brands. As such, the market leader in protein powder was chosen to look at, Optimum Nutrition was chosen.

```
df_brot <- df_brot%>%
  mutate(OptNut = case_when(
    (Brand == "Optimum Nutrition") ~ "1",
    (Brand != "Optimum Nutrition") ~ "0"
    ))
```

Therefore by creating a column registering a 1 if it's an Optimum Nutrition product, and a 0 if not, it is possible can look at the effect of this singular brand on the above model.

```
mod2 = lm(PPS ~ Price+OptNut, data = df_brot)

ggplot(df_brot, aes(x = Price, y = PPS, color = OptNut))+
  geom_point()+
  #geom_smooth(se = FALSE, method = "lm")+
  geom_abline(slope=mod2$coefficients[2], intercept=mod2$coefficients[1], col=2)+
  geom_abline(slope=mod2$coefficients[2], intercept=mod2$coefficients[1]+mod2$coefficients[3], col=4)+
  ggtitle("Price Per Serving As A Function Of Price And ON Brand") +
  xlab("Price") + ylab("Price Per Serving")
```




```
summary(mod2)
```

```
##
## Call:
## lm(formula = PPS ~ Price + OptNut, data = df_brot)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2545 -0.4917 -0.2378  0.1611  3.0179
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.369176   0.087836  26.973   <2e-16 ***
## Price       -0.018018   0.001755 -10.268   <2e-16 ***
## OptNut1      0.023360   0.109780   0.213    0.832
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.853 on 434 degrees of freedom
## Multiple R-squared:  0.1961, Adjusted R-squared:  0.1924
## F-statistic: 52.93 on 2 and 434 DF,  p-value: < 2.2e-16
```

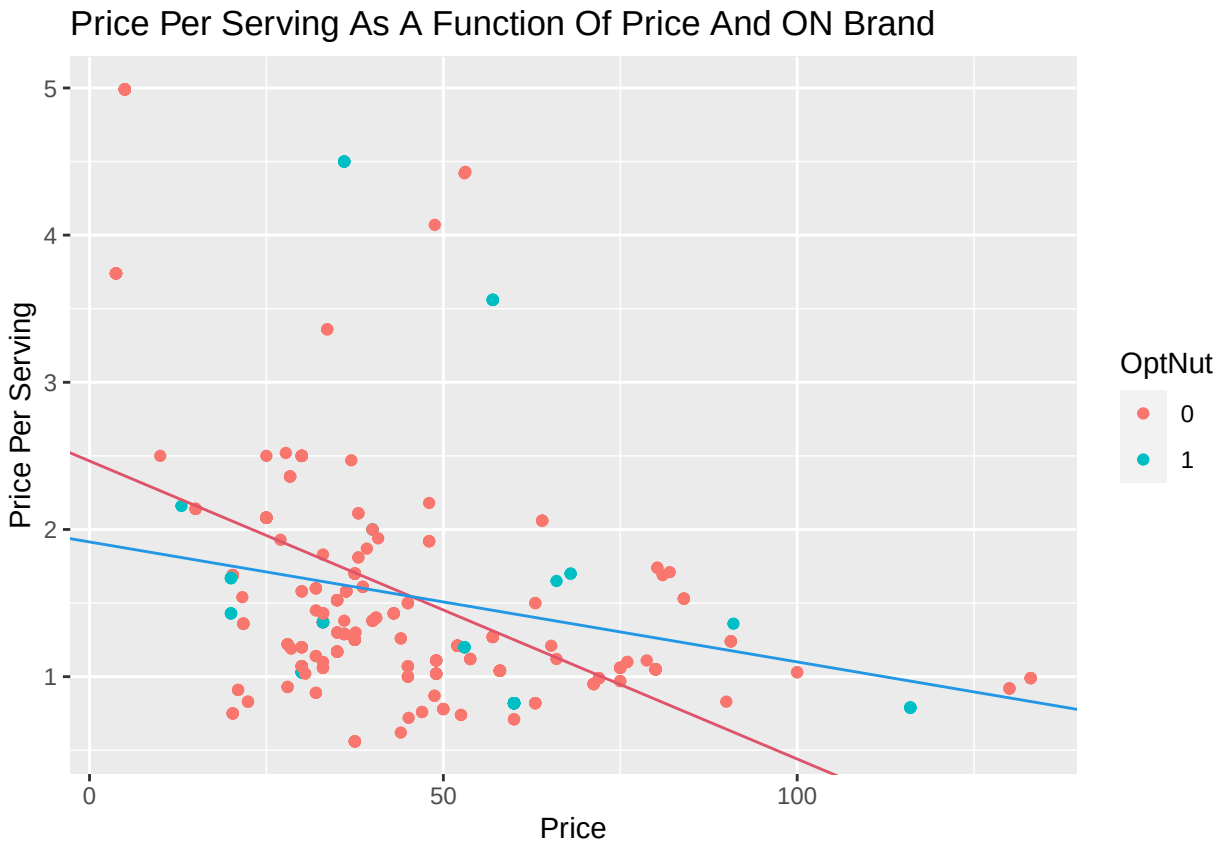
```
anova(mod2)
```

```
## Analysis of Variance Table
##
## Response: PPS
##            Df  Sum Sq Mean Sq  F value Pr(>F)
```

```
## Price       1  76.994  76.994 105.8226 <2e-16 ***
## OptNut      1   0.033   0.033   0.0453 0.8316
## Residuals 434 315.767   0.728
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Sadly the results were extremely insignificant, the resultant model of PPS as a function of Price and the ON marker was nearly identical to the original model with no added insights. With this loss, it was still worth looking to see if the interaction between the two separate variables held any insight.

```
mod3 = lm(PPS ~ Price*OptNut, data = df_brot)

modplot3 <- ggplot(df_brot, aes(x = Price, y = PPS, color = OptNut))+
  geom_point()+
  #geom_smooth(se = FALSE, method = "lm")+
  geom_abline(slope=mod3$coefficients[2], intercept=mod3$coefficients[1], col=2)+
  geom_abline(slope=mod3$coefficients[2]+mod3$coefficients[4], intercept=mod3$coefficients[1]+mod3$coefficients[3], col=4)+
  ggtitle("Price Per Serving As A Function Of Price And ON Brand") +
  xlab("Price") + ylab("Price Per Serving")
modplot3
```




```
summary(mod3)
```

```
##
## Call:
## lm(formula = PPS ~ Price * OptNut, data = df_brot)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -1.3055 -0.5543 -0.2368  0.2044  3.0403
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.465286   0.094232  26.162  < 2e-16 ***
## Price         -0.020248   0.001929 -10.496  < 2e-16 ***
## OptNut1       -0.550026   0.239216  -2.299  0.02196 *
## Price:OptNut1  0.012097   0.004493   2.693  0.00736 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8469 on 433 degrees of freedom
## Multiple R-squared:  0.2093, Adjusted R-squared:  0.2039
## F-statistic: 38.21 on 3 and 433 DF,  p-value: < 2.2e-16
```

```
anova(mod3)
```

```
## Analysis of Variance Table
##
## Response: PPS
##               Df  Sum Sq Mean Sq  F value    Pr(>F)
## Price          1  76.994  76.994 107.3467 < 2.2e-16 ***
## OptNut         1   0.033   0.033   0.0459  0.830399
## Price:OptNut   1   5.200   5.200   7.2506  0.007363 **
## Residuals    433 310.566   0.717
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Fascinatingly enough, despite the lack of significance in the previous model, this new model has significance in both variables and the variable interaction. This model will be further explained in the analysis section.

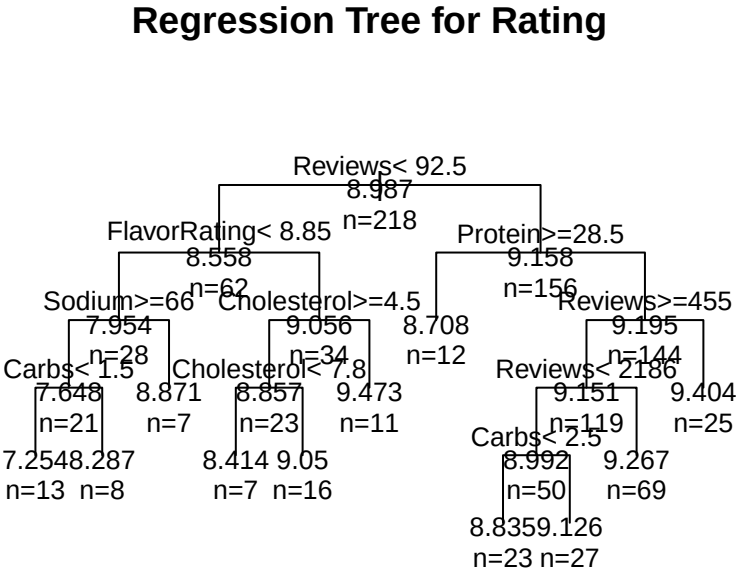
Tree Modeling

With having found a really concrete linear relationship for the PPS as a function of Price and Brand, answer the questions about the price per serving to price ratio and if brands have an influence in that, this project then turns to answer questions about the ratings. Specifically if the rating of whey protein products can be determined by a multitude of factors. For this, a tree will be used.

```
df_brotnf <- df_brot
df_brotnf[3] <- NULL
df_brotnf[2] <- NULL

df_brotnf <- df_brotnf

set.seed(1)
train<-sample(1:nrow(df_brotnf["Rating"]), nrow(df_brotnf["Rating"])/2)

broteintree <- rpart(Rating~., df_brotnf, subset=train, method = "anova")
#tree-building from the proteins data set

#summary(broteintree)
```

As there are so many different Flavors and Brands, the resultant trees are extremely messy and incredibly hard to understand. As such they are pruned prior to the tree construction as otherwise the trees are made entirely of two variables not easily decipherable.

```
par(mfrow=c(1,1))
plot(broteintree , uniform = TRUE, margin = 0.2, main = "Regression Tree for Rating")
text(broteintree, use.n=TRUE, all = TRUE, cex=.8)
```

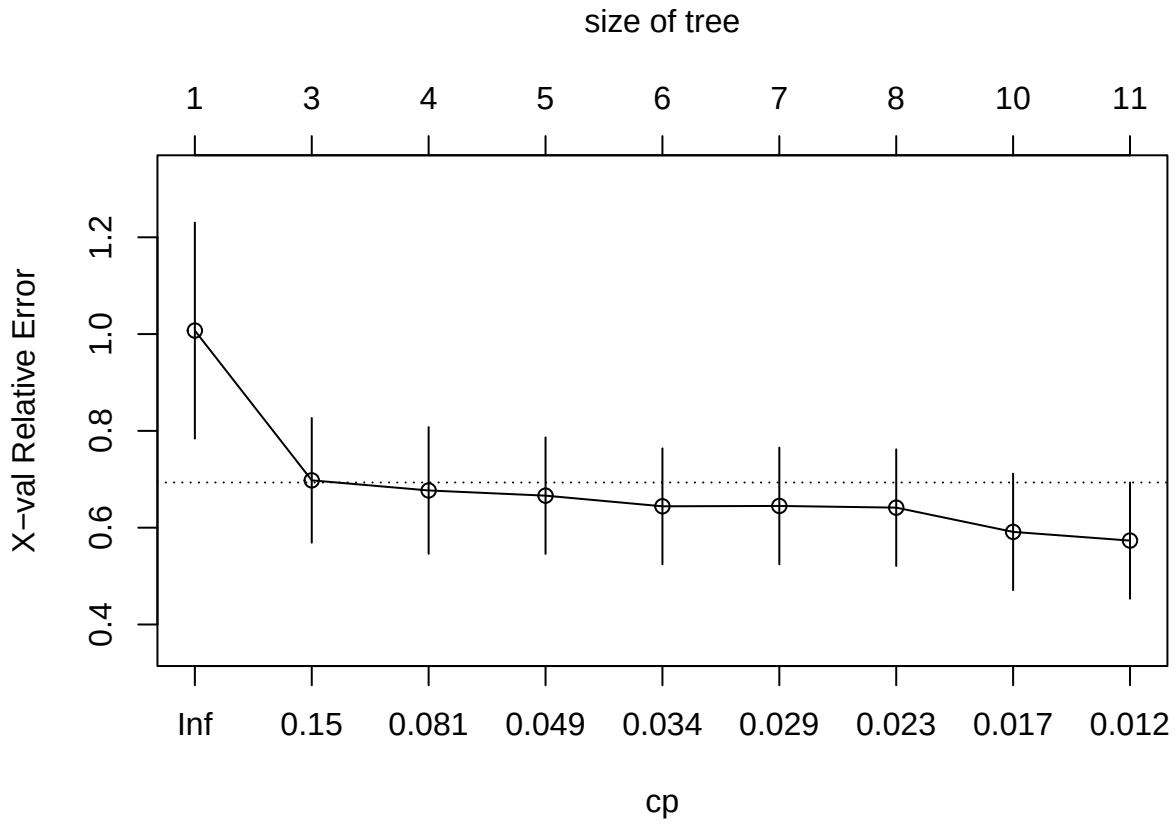


The initial tree looks promising if a bit messy, it is a priority to check the cross validation to ensure both the tree isn't drawn from non significant conclusions, and to ensure the CP that the tree is pruned from is an appropriate value.

```
printcp(broteintree )
```

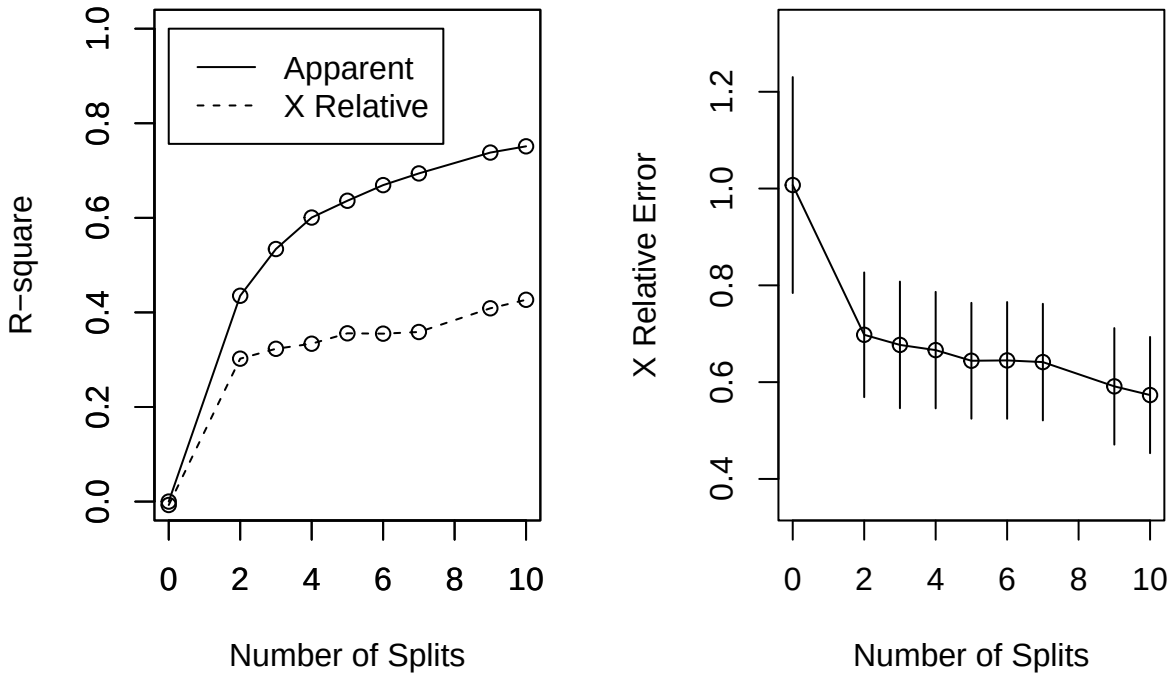
```
##
## Regression tree:
## rpart(formula = Rating ~ ., data = df_brotnf, subset = train,
##     method = "anova")
##
## Variables actually used in tree construction:
## [1] Carbs        Cholesterol  FlavorRating Protein      Reviews
## [6] Sodium
##
## Root node error: 79.524/218 = 0.36479
##
## n= 218
##
##         CP nsplit rel error  xerror    xstd
## 1 0.217606      0   1.00000 1.00722 0.22310
## 2 0.098875      2   0.56479 0.69782 0.12883
## 3 0.066537      3   0.46591 0.67694 0.13084
## 4 0.035530      4   0.39938 0.66617 0.12032
## 5 0.033009      5   0.36385 0.64417 0.11969
## 6 0.024747      6   0.33084 0.64492 0.12062
## 7 0.022049      7   0.30609 0.64143 0.12048
## 8 0.013238      9   0.26199 0.59130 0.12040
## 9 0.010000     10   0.24875 0.57331 0.12009
```

```
plotcp(broteintree )
```



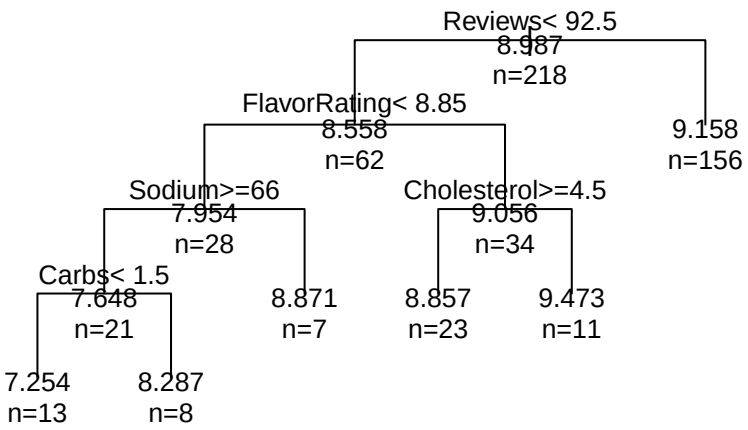

```
par(mfrow=c(1,2))
rsq.rpart(broteintree)
```

```
##
## Regression tree:
## rpart(formula = Rating ~ ., data = df_brotnf, subset = train,
##     method = "anova")
##
## Variables actually used in tree construction:
## [1] Carbs        Cholesterol  FlavorRating Protein      Reviews
## [6] Sodium
##
## Root node error: 79.524/218 = 0.36479
##
## n= 218
##
##         CP nsplit rel error  xerror    xstd
## 1 0.217606      0   1.00000 1.00722 0.22310
## 2 0.098875      2   0.56479 0.69782 0.12883
## 3 0.066537      3   0.46591 0.67694 0.13084
## 4 0.035530      4   0.39938 0.66617 0.12032
## 5 0.033009      5   0.36385 0.64417 0.11969
## 6 0.024747      6   0.33084 0.64492 0.12062
## 7 0.022049      7   0.30609 0.64143 0.12048
## 8 0.013238      9   0.26199 0.59130 0.12040
## 9 0.010000     10   0.24875 0.57331 0.12009
```



From the CP table, there seems to be slight minor gains in xerror rate after the 5th CP value, and 5 splits in a tree is still relatively readable. As such CP(0.033009) is chosen to prune the tree for the balence between readability and accuracy.

```
pfit<- prune(broteintree, cp=0.033009) # from cptable

plot(pfit, uniform=TRUE,margin=0.2,
     main="Pruned Regression Tree for Nsplit=5")
text(pfit, use.n=TRUE, all=TRUE, cex=.8)
```



The final model tree is much easier to understand then the original model summary, and relatively clean. This tree model will further be explored in the analysis section.

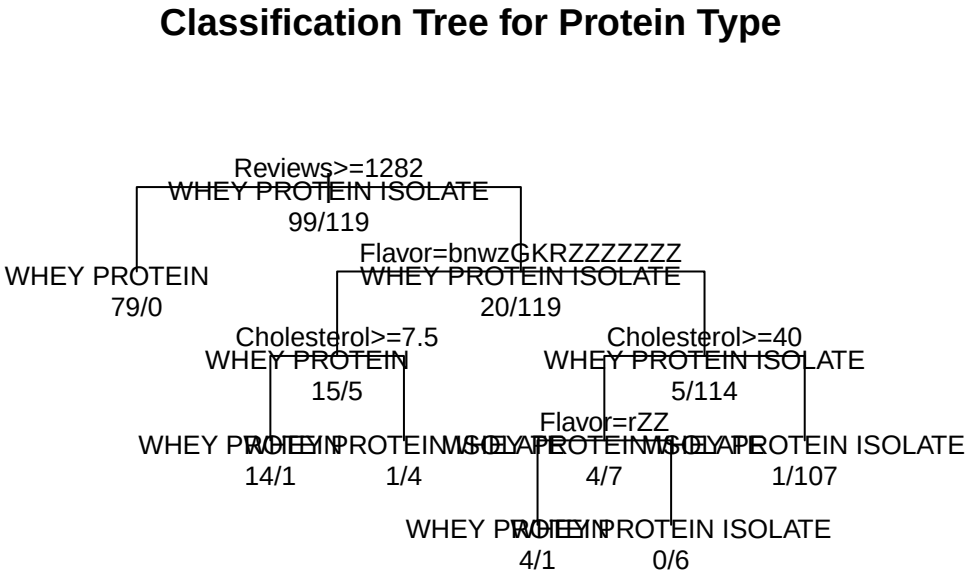
Looking now at classification trees, there's two separate types of Whey Protein contained within the data set. There's standard Whey Protein (hereon refereed to as WP) and Whey Protein Isolate (WPI), A more

expensive and "pure" form of protein powder. Using a tree, is it possible to see if just based on the variables contained withen to guess which Category of protein powder a given product is?

```
df_brotnb <- df_brot
df_brotnb[2] <- NULL

set.seed(1)

train<-sample(1:nrow(df_brotnb["Category"]), nrow(df_brotnb["Category"])/2)

wheytree<-rpart(Category~., data=df_brotnb, control=rpart.control(minsplit=10), subset=train, method="cl

#summary(wheytree)
```

```
par(mfrow=c(1,1))
plot(wheytree , uniform=TRUE,margin=0.2,
     main="Classification Tree for Protein Type")
text(wheytree , use.n=TRUE, all=TRUE, cex=.8)
```

```
## Warning in labels.rpart(x, minlength = minlength): more than 52 levels in a
## predicting factor, truncated for printout
```

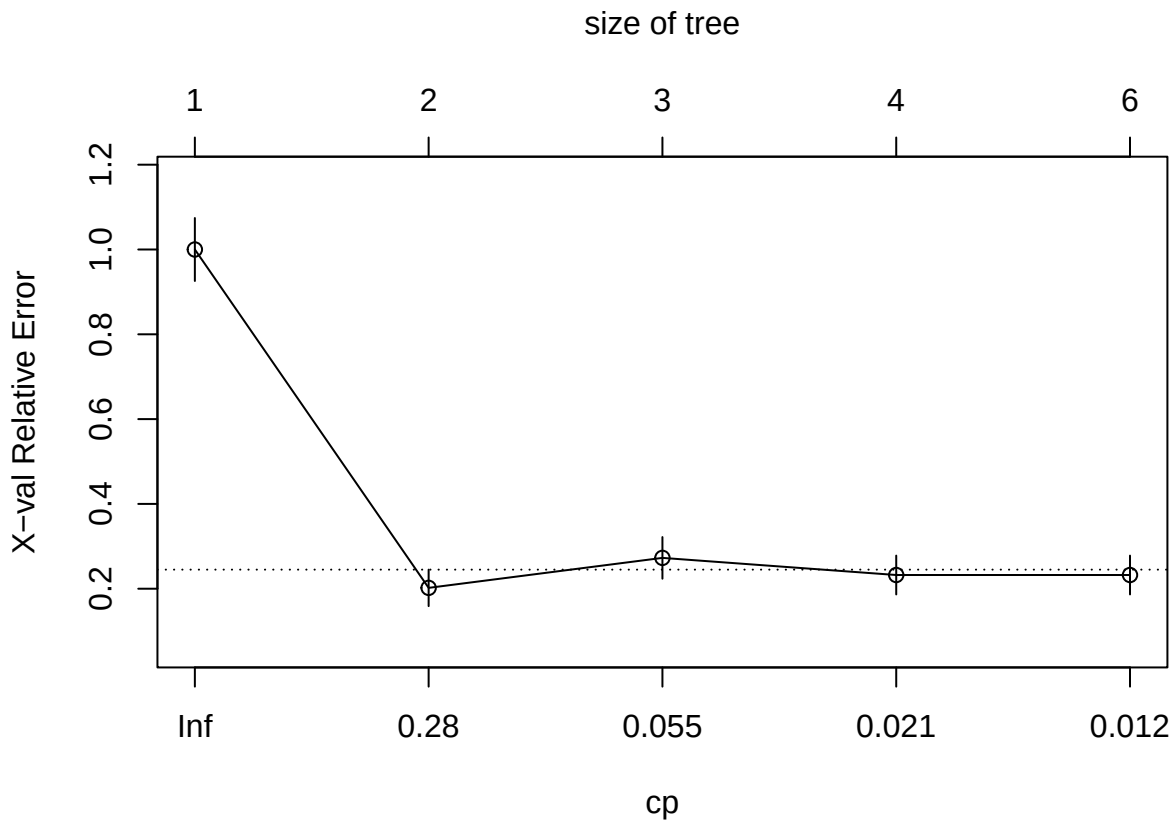



```
printcp(wheytree )
```

```
##
## Classification tree:
## rpart(formula = Category ~ ., data = df_brotnb, subset = train,
##     method = "class", control = rpart.control(minsplit = 10))
##
## Variables actually used in tree construction:
## [1] Cholesterol Flavor      Reviews
##
## Root node error: 99/218 = 0.45413
##
## n= 218
##
##         CP nsplit rel error  xerror     xstd
```

```
## 1 0.797980      0  1.000000 1.00000 0.074255
## 2 0.101010      1  0.202020 0.20202 0.043051
## 3 0.030303      2  0.101010 0.27273 0.049129
## 4 0.015152      3  0.070707 0.23232 0.045816
## 5 0.010000      5  0.040404 0.23232 0.045816
```

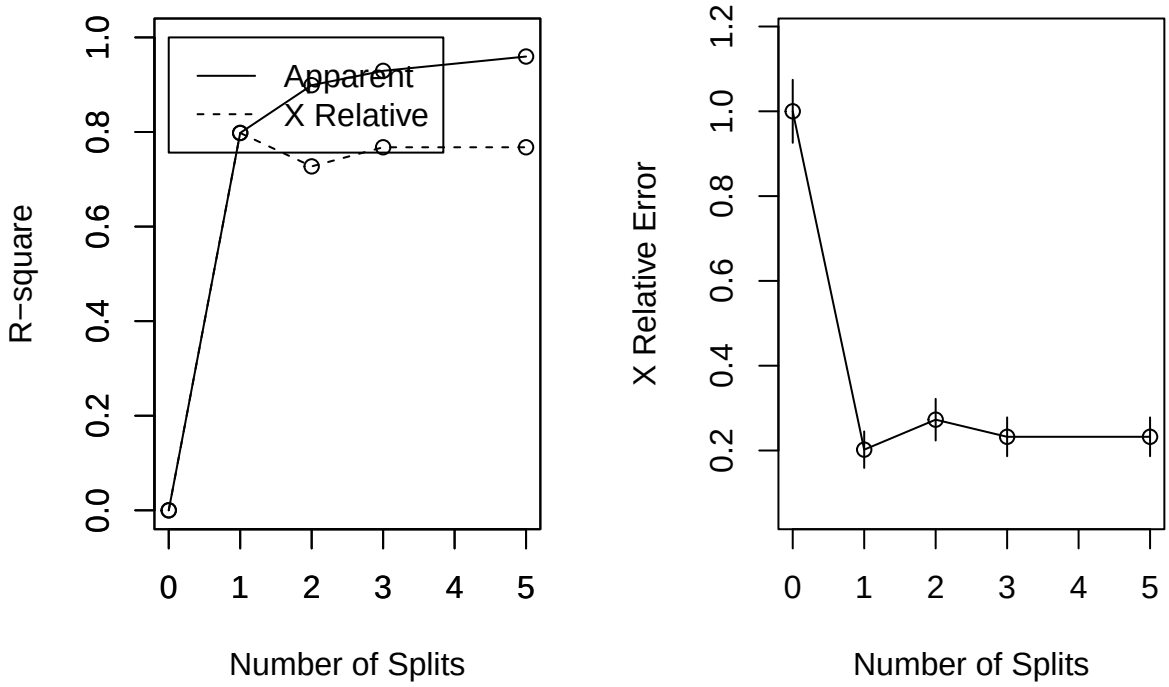
```
plotcp(wheytree )
```




```
par(mfrow=c(1,2))
rsq.rpart(wheytree)
```

```
##
## Classification tree:
## rpart(formula = Category ~ ., data = df_brotnb, subset = train,
##     method = "class", control = rpart.control(minsplit = 10))
##
## Variables actually used in tree construction:
## [1] Cholesterol Flavor      Reviews
##
## Root node error: 99/218 = 0.45413
##
## n= 218
##
##         CP nsplit rel error  xerror     xstd
## 1 0.797980      0  1.000000 1.00000 0.074255
## 2 0.101010      1  0.202020 0.20202 0.043051
## 3 0.030303      2  0.101010 0.27273 0.049129
## 4 0.015152      3  0.070707 0.23232 0.045816
## 5 0.010000      5  0.040404 0.23232 0.045816
```

```
## Warning in rsq.rpart(wheytree): may not be applicable for this method
```

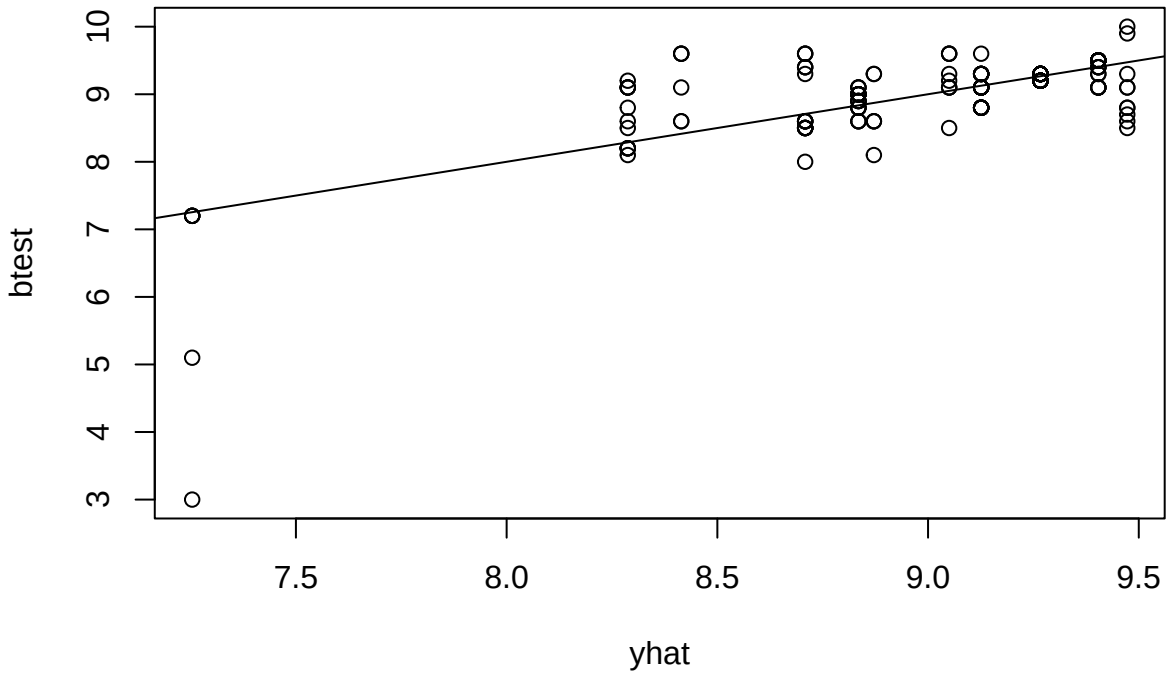


Plotting the tree and looking at it, gives an extremely good result right off the bat, with only 5 splits and relatively good leaf purity, without being extremely over restrictive. There was no need to prune it any further as once expanded to full size the tree is readable and workable with.

This Tree will be further dissected in the Analysis section

Using the initial regression tree model to see how effective it is at the testing set.

```
# making predictions
# make predictions
yhat <- predict(broteintree, newdata=df_brotnf[-train,])
brotein.test<-df_brotnf[-train, "Rating"]
btest <- pull(brotein.test, "Rating")
#for some reason the brotein test was a tibble and not a vector
plot(yhat, btest)
abline(0,1)
```




```
# mse
mean((yhat-btest)^2)
```

```
## [1] 0.2112805
```

Analysis

From our first Method section in linear modeling, we ended up with three separate models. The question of which linear model is best, can be deduced by seeing the mean of the residual sums.

```
mod1_summ <- summary(mod1)
mod2_summ <- summary(mod2)
mod3_summ <- summary(mod3)

mean(mod1_summ$residuals^2)
```

```
## [1] 0.7226533
```

```
mean(mod2_summ$residuals^2)
```

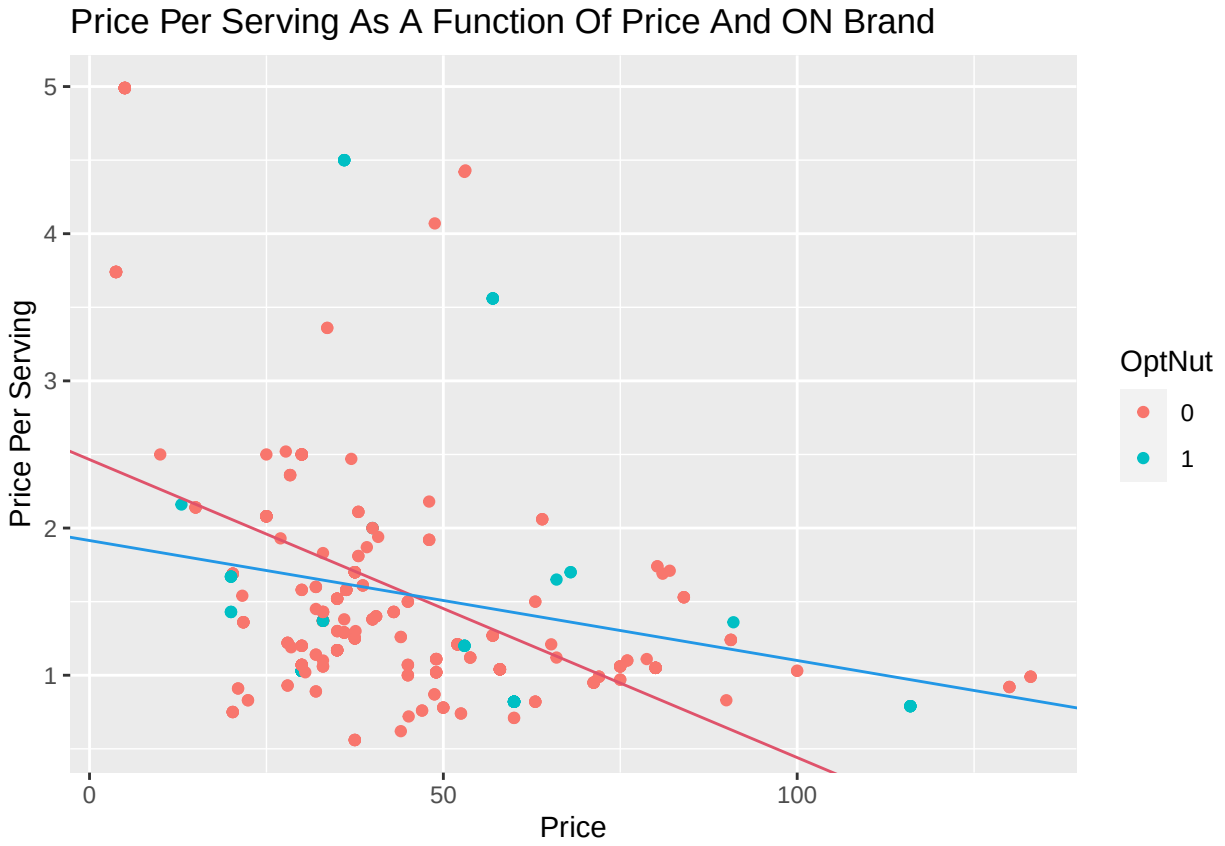
```
## [1] 0.7225779
```

```
mean(mod3_summ$residuals^2)
```

```
## [1] 0.7106776
```

The residuals are extremely small in all three models but the third model is slightly smaller then the rest while the third model still retains great significance without over complicating the model. Therefore out of the above three models it can be argued that 3 shows the best relationship

```
modplot3
```

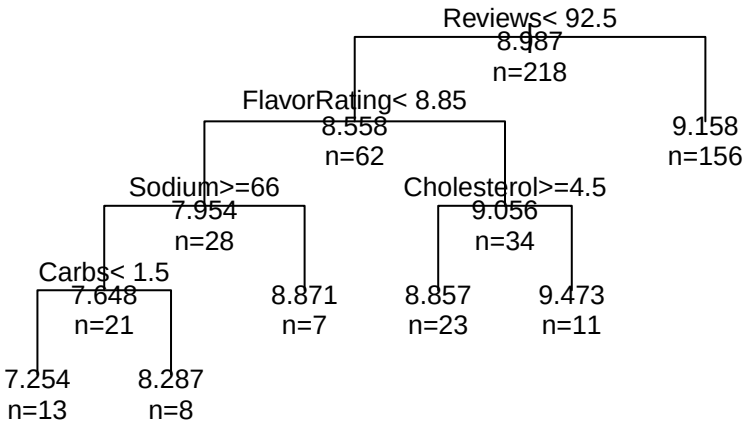


What this model is saying is that the brand Optimum Nutrition has an interaction with the Pricing of Whey Protein Powders. Meaning the Pricing of the ON brand is different and as such follows a different "Bang for Buck" slope line then the rest of the industry. A cheaper ON product will have a more expensive PPS:Price ratio then competitors but a more expensive ON product will in turn have a cheaper PPS:Price ratio then competitors. The ANOVA table does show however that the actual ON variable lacks significance, but the variable interaction still holds good significance.

Not only is this model significantly important, it's got the best residual score and makes logic sense when analyzed. Therefore out of all the linear models, this was best showcase of the data set.

Both of the trees showed promising ideas, but the one tackling the bigger question of "how to determine a products rating" was the Regression Tree pictured below.

```
plot(pfit, uniform=TRUE,margin=0.2,
     main="Pruned Regression Tree for Nsplit=5")
text(pfit, use.n=TRUE, all=TRUE, cex=.8)
```



When analyzing the Reg. Tree it is again possible to plot the mean residuals as previously done

```
mean((yhat-btest)^2)
```

```
## [1] 0.2112805
```

From this, a relatively simple tree with a relatively low residual mean concludes that the reg tree was a great model for finding rating Regression. The tree also logically makes sense, going through some of the major splits, it's reasonable to conclude that a small amount of reviews would lead to a relatively high rating (as a principal of online reviews are that people are more likely to review something they really like, then something they're more meh about). Below that split, it's completely reasonable to conclude that a higher flavor rating would lead to a high overall rating. Sodium makes sense as well, it's a filler ingrident in protein powder and tends to taste bad, thus leading to a worse rating in higher amounts. Finally Carbs makes sense, like sodium it's very much a filler ingrident and the greater the amount the more likely it's a cheap product, and therefore, a worse rating.

Together these two models accurately answer the questions set forth in the beginning, the question about the relation between price and price per serving, and the question about if rating can be roughly determined by other factors.

Conclusion

Overall the models displayed within made sense but opened ideas about their respective supplements that wouldn't have been initially obvious, or went far more in-depth then previously thought. The best example was easily in the linear model as it evolved. After using Optimum Nutrition in the graph, and seeing the lack of any significance, but then seeing the significance from the interaction between the brand and the price was quick shocking. Seeing that each brand could potentially be broken down to find their own respective Price to PPS ratio.

The Trees as well made sense, and were split based on the different ingredients mostly made sense. No one tree split really took the project off a normal idea or were extreme outliers, but some gave way to general thoughts, like filler ingredients leading to worse ratings.

The only regrets about this data set was how little we were able to do with the total data set, as we wanted to peer more into the other categories like pre-workouts and other industry niche products, but as much of the data frame was unusable, many of the cool ideas we started off with (like analyzing pre-workout price based on ingredients within) were unable to do. Still just predominately using the two Whey Protein categories, there was a lot of insight to be found.