Choose Your Own Project N.Clode

Nclode

15/06/2019

Contents

1 Executive Summary **2**

1.1 Source . . . 2

1.2 Approach . . . 2

1.3 Data Subsets . . . 2

1.4 Read Data & create test & training sets . . . 3

2 Data Exploration **3**

2.1 Density, Histogram and Box Plots by Sex . . . 5

2.2 Plots of Abalone Dimension variables against Rings . . . 6

2.3 Plots of Abalone Weights against Rings . . . 7

2.4 Full Correlation Chart . . . 9

2.5 Correlation of each Variable to Rings . . . 10

3 Base Models **12**

3.1 Linear Regression . . . 12

3.2 Regression Tree with rpart . . . 16

3.3 Modal Tree Using Cubist . . . 17

3.4 Random Forrest . . . 19

3.5 K Nearest Neighbours Regression Model (KNN) . . . 20

3.6 Generalized Additive Model (GAM) . . . 21

3.7 Multivariate Adaptive Regression Splines (MARS) Model . . . 24

3.8 Base Model Results . . . 27

4 Ensemble Models to improve RMSE **27**

4.1 Ensemble 1 - Simple Average. . . . 28

4.2 Ensemble 2 - Linear Regression . . . 28

4.3 Fit each Base Model with the new train_cv subsets and predict using the test_cv subsets. . . . 29

5 Results **34**

6 Conclusion **35**

1 Executive Summary

The purpose of this project is to build and train a machine learning model in R that can predict the age of Abalone from its physical attributes.

Abalone are marine snails whose shells are used as a popular inlay material and a source of mother of pearl for the jewellery industry. The age of abalone is typically determined by counting its Rings though a time-consuming highly manual process, which involves cutting the shell through the cone, staining it, and counting the number of rings through a microscope. It would save time and costs if the number of Rings (hence the age) could be deduced from physical attributes of Abalone which are easier to obtain.

This project will take data of Abalone shells from the Marine Research Laboratories of the Department of Primary Industry and Fisheries in Tasmania and use the physical observable attributes: height, diameter, length, weights and sex to train models which predict the number of Rings.

1.1 Source

The Abalone Data is sourced from
link Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/datasets/abalone]. Irvine, CA: University of California, School of Information and Computer Science.

Data is provided by Marine Resources Division Marine Research Laboratories - Taroona Department of Primary Industry and Fisheries, Tasmania

1.2 Approach

This project will go beyond a simple Linear regression and train 7 different models using different algorithms which predict the count of Rings based on physical measurements.
The Base models trained will be:
1. Stepwise Linear Regression
2. Regression Tree
3. Modal Tree
4. Random Forrest
5. K Nearest Neighbours
6. Generalized Additive Model using Splines
7. Multivariate Adaptive Regression Splines

In addition to the above, I will improve on the individual predictions by generating ensemble models. The ensembles will take predictions of each of the above base models as inputs and then generate a final Prediction. There will be 2 different Ensembles generated
1. Ensemble_1: Simple Average, where the Ensemble prediction for a given Abalone is the average prediction of each of the 7 base models
2. Ensemble_2: Stepped Linear Regression, where the individual predictions are used as inputs into a regression model to generate the final prediction

1.3 Data Subsets

The abalone dataset is split into subsets for the purpose of training, testing and cross validating.
1.Validation set: This subset is 10% of the total data and is used for testing generating the final RMSE scores only. This data will **not** be used for training at any stage of the process.

2.The Training set: This subset is 90% of the total data and is used to train the models model. In addition, this set is split into additional subsets which are used for cross validating and training the Ensemble models.

1.4 Read Data & create test & training sets

```
# Create the training and testing subsets
set.seed(5)
test_index <- createDataPartition(full_data$Rings, times = 1, p = 0.1, list = FALSE)
train_set <- full_data %>% slice(-test_index)
test_set <- full_data %>% slice(test_index)
```

2 Data Exploration

This section explores basic the structure and attributes of the Abalone data.

```
# Explore the data
formattable(as.table(summary(full_data)), format = "markdown")
```

```
##  Sex                    Length           Diameter
##  "F:1307          " "Min.   :0.075   " "Min.   :0.0550  "
##  "I:1342          " "1st Qu.:0.450   " "1st Qu.:0.3500  "
##  "M:1528          " "Median :0.545   " "Median :0.4250  "
##  "NA              " "Mean   :0.524   " "Mean   :0.4079  "
##  "NA              " "3rd Qu.:0.615   " "3rd Qu.:0.4800  "
##  "NA              " "Max.   :0.815   " "Max.   :0.6500  "
##      Height          Whole.weight      Shucked.weight
##  "Min.   :0.0000  " "Min.   :0.0020  " "Min.   :0.0010  "
##  "1st Qu.:0.1150  " "1st Qu.:0.4415  " "1st Qu.:0.1860  "
##  "Median :0.1400  " "Median :0.7995  " "Median :0.3360  "
##  "Mean   :0.1395  " "Mean   :0.8287  " "Mean   :0.3594  "
##  "3rd Qu.:0.1650  " "3rd Qu.:1.1530  " "3rd Qu.:0.5020  "
##  "Max.   :1.1300  " "Max.   :2.8255  " "Max.   :1.4880  "
##  Viscera.weight      Shell.weight          Rings
##  "Min.   :0.0005  " "Min.   :0.0015  " "Min.   : 1.000  "
##  "1st Qu.:0.0935  " "1st Qu.:0.1300  " "1st Qu.: 8.000  "
##  "Median :0.1710  " "Median :0.2340  " "Median : 9.000  "
##  "Mean   :0.1806  " "Mean   :0.2388  " "Mean   : 9.934  "
##  "3rd Qu.:0.2530  " "3rd Qu.:0.3290  " "3rd Qu.:11.000  "
##  "Max.   :0.7600  " "Max.   :1.0050  " "Max.   :29.000  "
```

```
# Row and Column Counts
dim(full_data)
```

```
## [1] 4177    9
```

```
# Classes of each Column
as.matrix(sapply(full_data, class))
```

```
##                [,1]
## Sex            "factor"
## Length         "numeric"
## Diameter       "numeric"
```

```
## Height          "numeric"
## Whole.weight    "numeric"
## Shucked.weight "numeric"
## Viscera.weight "numeric"
## Shell.weight    "numeric"
## Rings           "integer"
```

```
# Column Names
names(full_data)
```

```
## [1] "Sex"             "Length"         "Diameter"       "Height"
## [5] "Whole.weight"    "Shucked.weight" "Viscera.weight" "Shell.weight"
## [9] "Rings"
```

```
# Check for N/A's
as.matrix(apply(full_data, 2, function(x) any(is.na(x))))
```

```
##                  [,1]
## Sex             FALSE
## Length          FALSE
## Diameter        FALSE
## Height          FALSE
## Whole.weight    FALSE
## Shucked.weight FALSE
## Viscera.weight FALSE
## Shell.weight    FALSE
## Rings           FALSE
```

```
# Examine the first 5 rows
formattable(full_data[1:5, ], format = "markdown")
```

Sex	Length	Diameter	Height	Whole.weight	Shucked.weight	Viscera.weight	Shell.weight	Rings
M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

2.0.1 Observations

- There are 4177 rows each is of an individual Abalone.
- There are 9 columns: the dependant variable **Rings** which is an integer as well as 7 numeric variables (dimensions and weights) and one factor variable Sex which has three options, *Male*, *Female* and *Infant*
- There are no N/As in any of the variables
- The dependant variable **Rings** Ranges between 1 and 29, and has a Mean of 9.9. The Median is lower than the Mean at 9.0

2.1 Density, Histogram and Box Plots by Sex

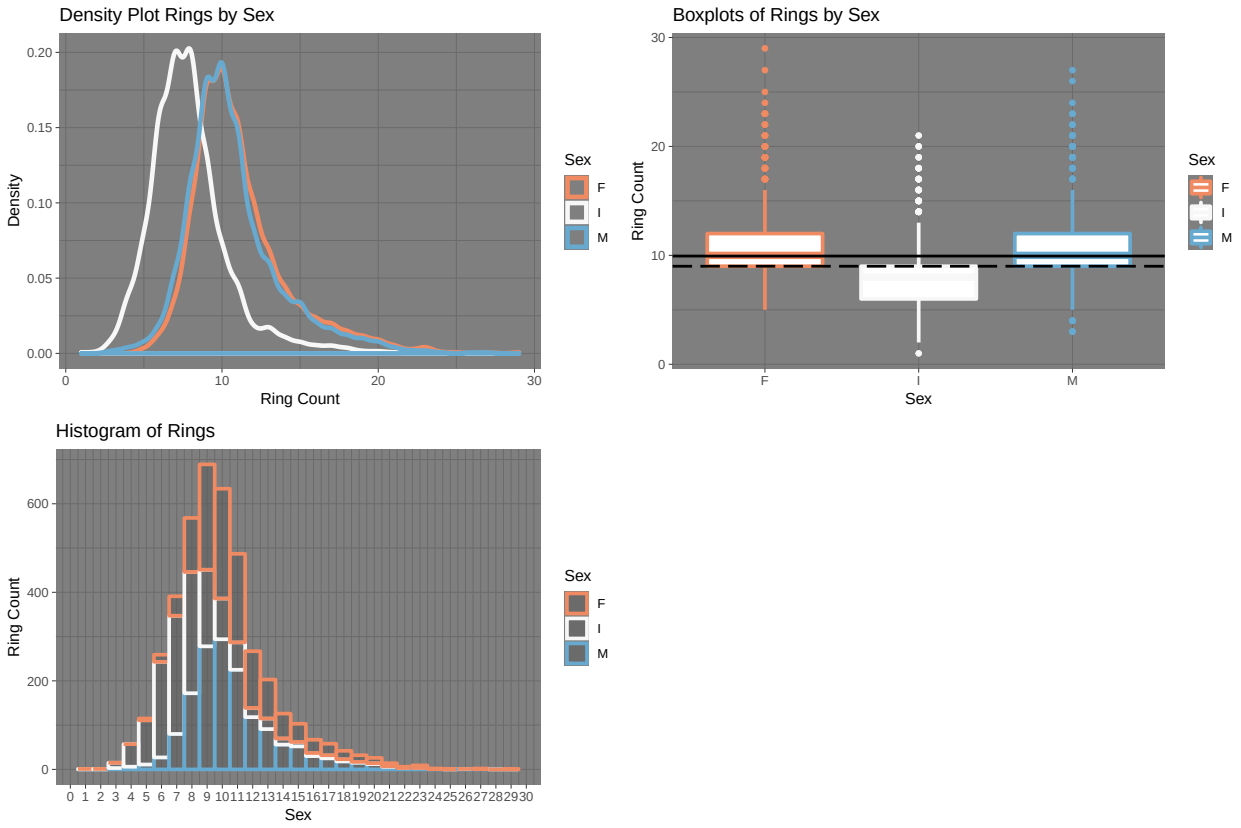
This section examines the Sex variable

```
# Density Plot of Rings
S1 <- full_data %>% ggplot(aes(Rings, color = Sex)) %>%
  + geom_density(size=1.5) %>%
+ xlab("Ring Count") %>%
+ ylab("Density") %>%
+ ggtitle("Density Plot Rings by Sex") %>%
+ scale_color_hue(labels = c("Female", "Infant","Male")) %>%
+ scale_colour_brewer(palette = "RdBu") %>%
+ theme_dark()

# Boxplot of Rings by Sex
S2 <-  full_data %>% ggplot(aes(x=Sex,y=Rings, color = Sex))  %>%
+ geom_boxplot(size = 1.2) %>%
+ xlab("Sex") %>%
+ ylab("Ring Count") %>%
+ ggtitle("Boxplots of Rings by Sex") %>%
+ geom_hline(aes(yintercept=median(Rings, na.rm = FALSE))
             , linetype = 5, color = "black",size = .9) %>%
+ geom_hline(aes(yintercept=mean(Rings, na.rm = FALSE))
             , linetype = 1, color = "black",size = .9) %>%
+ scale_colour_brewer(palette = "RdBu") %>%
+ theme_dark()

# Histogram by Rings
S3 <-  full_data %>% ggplot(aes(Rings,color = Sex))  %>%
+ geom_histogram(size = 1.2 , alpha=0.5,binwidth=1) %>%
+ xlab("Sex") %>%
+ ylab("Ring Count") %>%
+ ggtitle("Histogram of Rings") %>%
+ scale_colour_brewer(palette = "RdBu") %>%
+ scale_x_continuous(breaks = seq(0, 30, by = 1)) %>%
+ theme_dark()

# Arrange the Plots
grid.arrange(S1, S2, S3, ncol = 2)
```



2.1.1 Observations

- The boxplot shows the mean Ring count *(solid black line)* lines up with the mean Ring count for both Male and Female Abalones
- The median *(dashed black line)* is lower than the mean) Possibly due to outliers of older Abalone.
- Infant abalone has lower Mean, Median and Range than both Male and Female.
- The abalone with the lowest Ring Count is an Infant and the abalone with the highest ring count is Female
- The Male and Female Densities are very similar; however the Infant density is observably is lower.

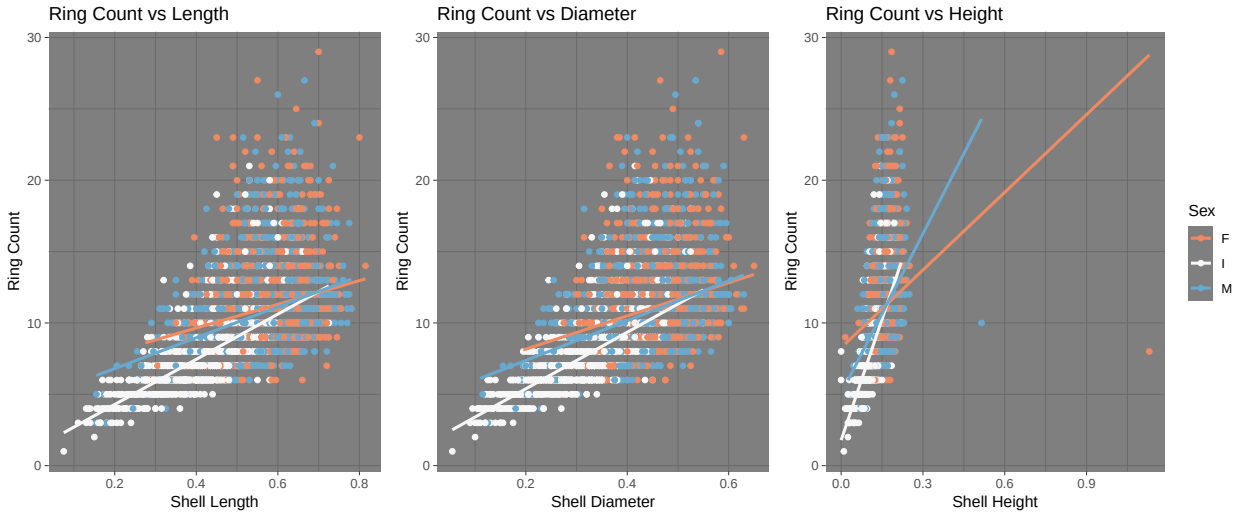
2.2 Plots of Abalone Dimension variables against Rings

This Section plots the dependant Variable Rings against each of the dimensions variables split by Sex.

```
# Length plot
L <-  full_data %>% ggplot(aes(Length, Rings, color = Sex)) %>%
+ geom_point() %>%
+ xlab("Shell Length") %>%
+ ylab("Ring Count") %>%
+ ggtitle("Ring Count vs Length") %>%
+ scale_colour_brewer(palette = "RdBu",guide=FALSE) %>%
+ theme_dark()%>%
+  stat_smooth(method = "lm", se = FALSE)
# Diameter plot
D <- full_data %>% ggplot(aes(Diameter, Rings, color = Sex)) %>%
```

```
+ geom_point() %>%
+ xlab("Shell Diameter") %>%
+ ylab("Ring Count") %>%
+ ggtitle("Ring Count vs Diameter") %>%
+ scale_colour_brewer(palette = "RdBu",guide=FALSE) %>%
+ theme_dark()%>%
+  stat_smooth(method = "lm", se = FALSE)
# Height plot
H <- full_data %>% ggplot(aes(Height, Rings, color = Sex)) %>%
+ geom_point() %>%
+ xlab("Shell Height") %>%
+ ylab("Ring Count") %>%
+ ggtitle("Ring Count vs Height") %>%
+ scale_color_hue(labels = c("Female", "Infant", "Male")) %>%
+ scale_colour_brewer(palette = "RdBu") %>%
+ theme_dark() %>%
+  stat_smooth(method = "lm", se = FALSE)

# Arrange the plots
grid.arrange(L, D,H, widths = c(1.65, 1.65,2))
```



2.2.1 Observations

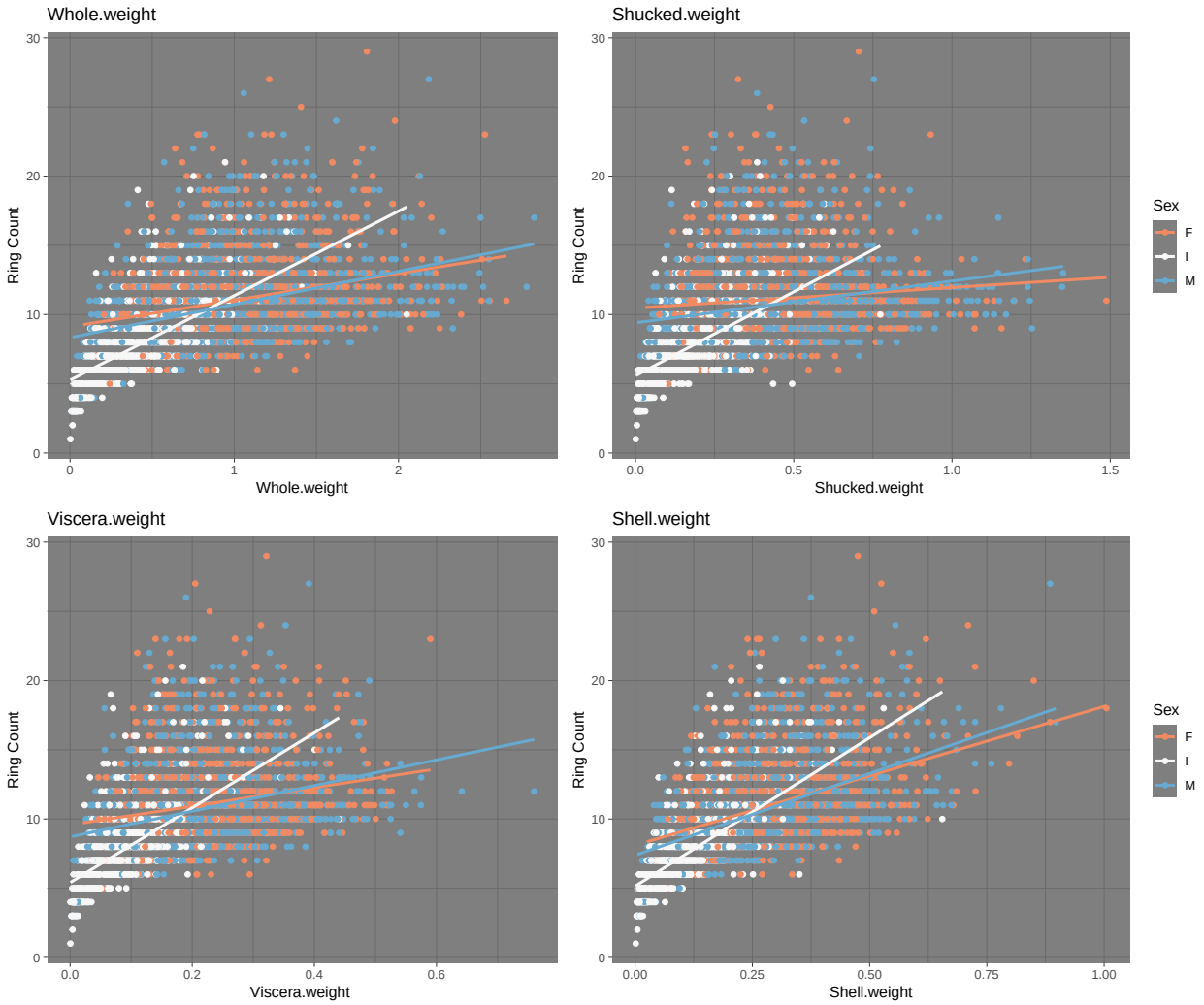
The above charts show Rings by the variables Shell Length, Shell Diameter, and Shell Height
* Infant Abalone values are concentrated in the lower left-hand corner for all three attributes * The Variation in Rings is high for Shell Lengths and Shell Diameters above ~0.4 * There are 2 outliers in Shell Height one Female and one Male

2.3 Plots of Abalone Weights against Rings

This Section produces the same style plots for the weight variables split by Sex

```
# Whole Weight plot
WW <-  full_data %>% ggplot(aes(Whole.weight, Rings, color = Sex)) %>%
+ geom_point() %>%
+ xlab("Whole.weight") %>%
+ ylab("Ring Count") %>%
+ ggtitle("Whole.weight") %>%
+ scale_colour_brewer(palette = "RdBu",guide=FALSE) %>%
+ theme_dark()%>%
+  stat_smooth(method = "lm", se = FALSE)
# Shucked Weight plot
SUW <- full_data %>% ggplot(aes(Shucked.weight, Rings, color = Sex)) %>%
+ geom_point() %>%
+ xlab("Shucked.weight ") %>%
+ ylab("Ring Count") %>%
+ ggtitle("Shucked.weight ") %>%
+ scale_color_hue(labels = c("Female", "Infant", "Male")) %>%
+ scale_colour_brewer(palette = "RdBu") %>%
+ theme_dark()%>%
+  stat_smooth(method = "lm", se = FALSE)
# Viscera Weight plot
VW <- full_data %>% ggplot(aes(Viscera.weight, Rings, color = Sex)) %>%
+ geom_point() %>%
+ xlab("Viscera.weight ") %>%
+ ylab("Ring Count") %>%
+ ggtitle("Viscera.weight ") %>%
+ scale_colour_brewer(palette = "RdBu",guide=FALSE) %>%
+ theme_dark()%>%
+  stat_smooth(method = "lm", se = FALSE)
# Shell Weight plot
SEW <- full_data %>% ggplot(aes(Shell.weight, Rings, color = Sex)) %>%
+ geom_point() %>%
+ xlab("Shell.weight") %>%
+ ylab("Ring Count") %>%
+ ggtitle("Shell.weight") %>%
+ scale_color_hue(labels = c("Female", "Infant", "Male")) %>%
+ scale_colour_brewer(palette = "RdBu") %>%
+ theme_dark()%>%
+  stat_smooth(method = "lm", se = FALSE)

# Arrange the plots
grid.arrange(WW, SUW,VW,SEW ,ncol = 2, widths = c(1.75,2),heights = c(2,2))
```

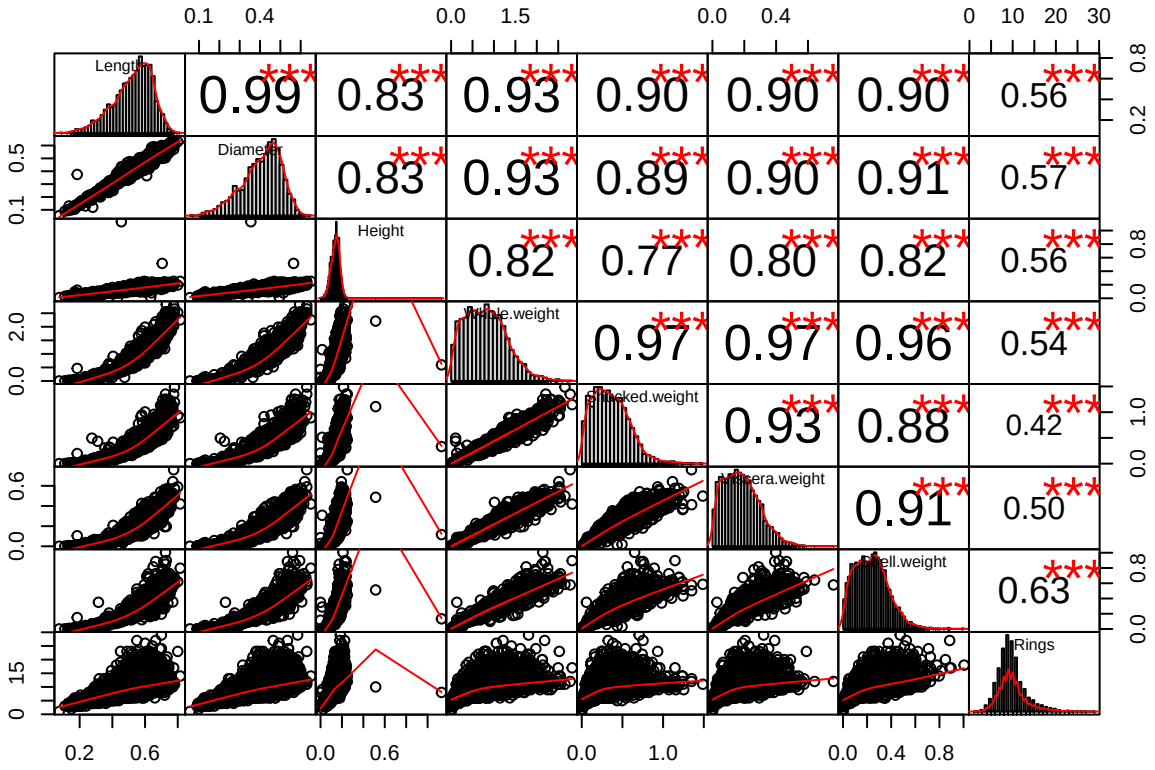


2.3.1 Observations

The above charts show Rings by the each of the weight variables split by Sex
* Infant Abalone values are concentrated in the lower left-hand corner for all attributes
* There are outliers in all 4 variables in the higher values.
* The linear trend lines do not fit models particularly well due to variation, especially with male and female abalone

2.4 Full Correlation Chart

This section will plot the correlation chart for all numeric variables.



2.4.1 Observations

The above plot shows the distribution of each numeric variable on the diagonal.
Bivariate scatter plots with a fitted line are displayed on the bottom of the diagonal The values of the correlation plus the significance level as stars are displayed on the top of the diagonal

Most of the measurements have high correlations with each other but only moderate correlations with the dependant variable Rings

2.5 Correlation of each Variable to Rings

This section examines more closely the correlation between Rings and each numeric variable individually, by splitting the data by Sex

```
# Create 3 separate subsets, one for each Sex
full_M <- full_data %>% filter(Sex == "M")
full_I <- full_data %>% filter(Sex == "I")
full_F <- full_data %>% filter(Sex == "F")

# Calculate the correlation with Rings of each variable individually split by Sex
   Correltations_with_rings <-  as.data.frame(rbind(
       c(Variable = "Length"
         ,Correlation = round(cor(full_data$Rings,full_data$Length),4)
         ,Correlation_Male = round(cor(full_M$Rings,full_M$Length),4)
         ,Correlation_Female = round(cor(full_F$Rings,full_F$Length),4)
```

```
    ,Correlation_Infant = round(cor(full_I$Rings,full_I$Length),4)),
  c(Variable = "Diameter"
    ,Correlation = round(cor(full_data$Rings,full_data$Diameter),4)
    ,Correlation_Male = round(cor(full_M$Rings,full_M$Diameter),4)
    ,Correlation_Female = round(cor(full_F$Rings,full_F$Diameter),4)
    ,Correlation_Infant = round(cor(full_I$Rings,full_I$Diameter),4)),
  c(Variable = "Height"
    ,Correlation = round(cor(full_data$Rings,full_data$Height),4)
    ,Correlation_Male = round(cor(full_M$Rings,full_M$Height),4)
    ,Correlation_Female = round(cor(full_F$Rings,full_F$Height),4)
    ,Correlation_Infant = round(cor(full_I$Rings,full_I$Height),4)),
  c(Variable = "Whole.weight"
    ,Correlation = round(cor(full_data$Rings,full_data$Whole.weight),4)
    ,Correlation_Male = round(cor(full_M$Rings,full_M$Whole.weight),4)
    ,Correlation_Female = round(cor(full_F$Rings,full_F$Whole.weight),4)
    ,Correlation_Infant = round(cor(full_I$Rings,full_I$Whole.weight),4)),
  c(Variable = "Shucked.weight"
    ,Correlation = round(cor(full_data$Rings,full_data$Shucked.weight),4)
    ,Correlation_Male = round(cor(full_M$Rings,full_M$Shucked.weight),4)
    ,Correlation_Female = round(cor(full_F$Rings,full_F$Shucked.weight),4)
    ,Correlation_Infant = round(cor(full_I$Rings,full_I$Shucked.weight),4)),
  c(Variable = "Viscera.weight"
    ,Correlation = round(cor(full_data$Rings,full_data$Viscera.weight),4)
    ,Correlation_Male = round(cor(full_M$Rings,full_M$Viscera.weight),4)
    ,Correlation_Female = round(cor(full_F$Rings,full_F$Viscera.weight),4)
    ,Correlation_Infant = round(cor(full_I$Rings,full_I$Viscera.weight),4)),
  c(Variable = "Shell.weight"
    ,Correlation = round(cor(full_data$Rings,full_data$Shell.weight),4)
    ,Correlation_Male = round(cor(full_M$Rings,full_M$Shell.weight),4)
    ,Correlation_Female = round(cor(full_F$Rings,full_F$Shell.weight),4)
    ,Correlation_Infant = round(cor(full_I$Rings,full_I$Shell.weight),4))))

# Format the table
  formattable(Correltations_with_rings,align =c("l","l","l", "l","l")
       ,format = "markdown",
          list(`Variable` = formatter("span"
            ,style = ~ style(color = "black",font.weight = "bold"))))
```

Variable	Correlation	Correlation_Male	Correlation_Female	Correlation_Infant
Length	0.5567	0.3666	0.2306	0.686
Diameter	0.5747	0.3889	0.2659	0.6951
Height	0.5575	0.4297	0.2339	0.72
Whole.weight	0.5404	0.3722	0.2668	0.6963
Shucked.weight	0.4209	0.2224	0.0948	0.6202
Viscera.weight	0.5038	0.321	0.2116	0.6733
Shell.weight	0.6276	0.511	0.4059	0.7254

2.5.1 Observations

- When not split by Sex, the correlation between Shell.weight and Rings is the highest at 0.6276
- When not split by Sex, the correlation between Whole.Weight and Rings is the lowest at 0.4209

- When each Variable is split by Sex however, the individual correlations between Rings and each numeric variable is higher for Infant, but lower for both Male and Female.

3 Base Models

This section trains each of the Base models using train_set data.
Each model is then evaluated by predicting Rings on the test_set data and generating an RMSE score.

3.1 Linear Regression

The first model trained is a simple Linear Regression

```
# Linear regression
# Train the Linear Regression model using the train_set
set.seed(1)
lm.Rings <- lm(Rings ~., data = train_set)
summary(lm.Rings)
```

```
##
## Call:
## lm(formula = Rings ~ ., data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.6066 -1.3112 -0.3564  0.8736 13.9649
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.00083    0.31070  12.877  < 2e-16 ***
## SexI            -0.83258    0.10844  -7.678 2.05e-14 ***
## SexM             0.07312    0.08793   0.832    0.406
## Length          -0.57297    1.91026  -0.300    0.764
## Diameter        11.09621    2.34312   4.736 2.26e-06 ***
## Height           9.88736    1.57549   6.276 3.88e-10 ***
## Whole.weight     8.76807    0.76725  11.428  < 2e-16 ***
## Shucked.weight -19.47851    0.85739 -22.718  < 2e-16 ***
## Viscera.weight -10.54052    1.37126  -7.687 1.92e-14 ***
## Shell.weight     9.25789    1.18448   7.816 7.03e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.202 on 3748 degrees of freedom
## Multiple R-squared:  0.5338, Adjusted R-squared:  0.5327
## F-statistic: 476.9 on 9 and 3748 DF,  p-value: < 2.2e-16
```

3.1.1 Linear Regression Details

- Looking at the model trained, all coefficients are statistically significant at the 99% level except for **Sex = M** and **Length**

- In addition the null hypothesis that there is **no** dependence of Rings on any of the explanatory variables, can be rejected at the 99% level due to the small F statistic p-value.
- The R Squared of 0.533 means approximately half the variation in Rings is explained by the model.

3.1.2 Use Backwards Stepwise Model selection to improve the regression.

This section uses AIC from the MASS package to check if the model can be improved by removing some of the variables

```
# Stepwise Regression using MASS

# Improve the model by performing a backwards stepwise regression
lm.step.Rings  <- stepAIC(lm.Rings, direction="backward")
```

```
## Start:  AIC=5943.78
## Rings ~ Sex + Length + Diameter + Height + Whole.weight + Shucked.weight +
##     Viscera.weight + Shell.weight
##
##                   Df Sum of Sq   RSS    AIC
## - Length           1      0.44 18178 5941.9
## <none>                         18178 5943.8
## - Diameter         1    108.77 18287 5964.2
## - Height           1    191.02 18369 5981.1
## - Viscera.weight   1    286.57 18464 6000.6
## - Shell.weight     1    296.29 18474 6002.5
## - Sex              2    416.83 18595 6025.0
## - Whole.weight     1    633.39 18811 6070.5
## - Shucked.weight   1   2503.21 20681 6426.6
##
## Step:  AIC=5941.87
## Rings ~ Sex + Diameter + Height + Whole.weight + Shucked.weight +
##     Viscera.weight + Shell.weight
##
##                   Df Sum of Sq   RSS    AIC
## <none>                         18178 5941.9
## - Height           1    190.61 18369 5979.1
## - Viscera.weight   1    291.77 18470 5999.7
## - Shell.weight     1    297.46 18476 6000.9
## - Sex              2    420.12 18598 6023.7
## - Diameter         1    484.77 18663 6038.8
## - Whole.weight     1    633.55 18812 6068.6
## - Shucked.weight   1   2518.01 20696 6427.4
```

```
summary(lm.step.Rings)
```

```
##
## Call:
## lm(formula = Rings ~ Sex + Diameter + Height + Whole.weight +
##     Shucked.weight + Viscera.weight + Shell.weight, data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -9.5858 -1.3093 -0.3560  0.8743 13.9832
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.96962    0.29272  13.561  < 2e-16 ***
## SexI            -0.83444    0.10825  -7.708 1.62e-14 ***
## SexM             0.07299    0.08791   0.830    0.406
## Diameter        10.46747    1.04687   9.999  < 2e-16 ***
## Height           9.87050    1.57429   6.270 4.03e-10 ***
## Whole.weight     8.76908    0.76715  11.431  < 2e-16 ***
## Shucked.weight -19.49513    0.85549 -22.788  < 2e-16 ***
## Viscera.weight -10.58188    1.36415  -7.757 1.11e-14 ***
## Shell.weight     9.27043    1.18360   7.832 6.18e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.202 on 3749 degrees of freedom
## Multiple R-squared:  0.5338, Adjusted R-squared:  0.5328
## F-statistic: 536.6 on 8 and 3749 DF,  p-value: < 2.2e-16
```

```
  #length is removed as a predictor variable

  # Compare the two models
  anova(lm.Rings, lm.step.Rings)
```

```
## Analysis of Variance Table
##
## Model 1: Rings ~ Sex + Length + Diameter + Height + Whole.weight + Shucked.weight +
##     Viscera.weight + Shell.weight
## Model 2: Rings ~ Sex + Diameter + Height + Whole.weight + Shucked.weight +
##     Viscera.weight + Shell.weight
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1   3748 18178
## 2   3749 18178 -1  -0.43634 0.09 0.7642
```

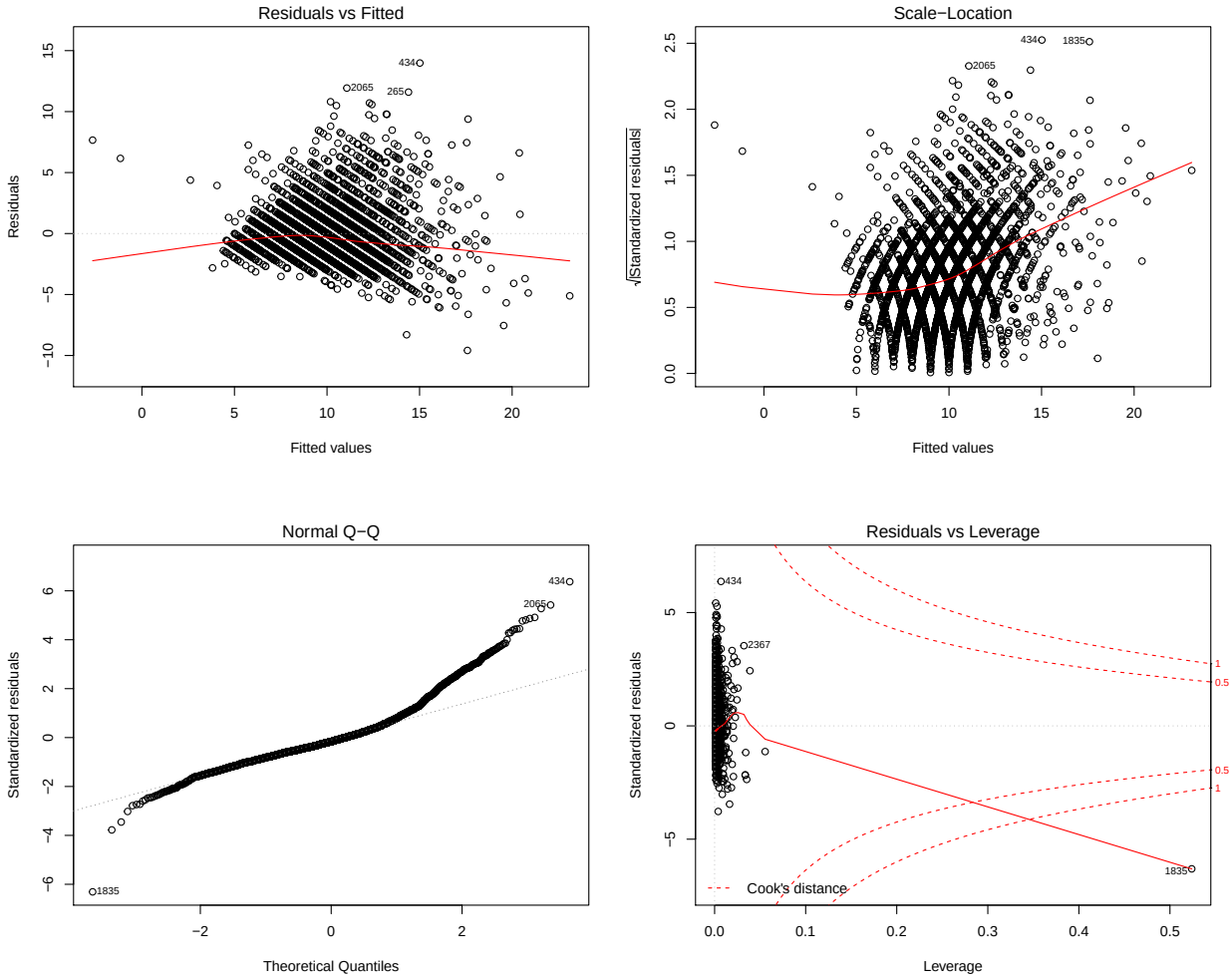
```
  #F statistic is high so we cannot reject the null hypothesis that the models are different
```

The length variable has been removed for the final model. When the 2 regression models are compared with the Anova test, the high F statistic p-value means that we cannot reject the null hypothesis that the models are different.

3.1.3 Plotting the final Model

```
  layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page
  plot(lm.step.Rings)
```

plot-1.bb



3.1.4 Observations

- The Residuals deviate from the line in the QQ plot which challenges the normality assumption
- Residuals vs Fitted shows a slight curve which some non-linearity in the data was not explained by the model
- Residuals vs Leverage shows an outlier at 1835 which is outside Cooks distance and is influencing the regression result

3.1.5 Liner Model RMSE

Predict Rings using the test set data with the linear model and generate the RMSE score

```
# 1 Linear Regression
lm.predictions.Rings <- predict(lm.step.Rings, test_set)
lm.RMSE <- RMSE(lm.predictions.Rings,test_set$Rings)
lm.RMSE
```

```
## [1] 2.121661
```

3.2 Regression Tree with rpart

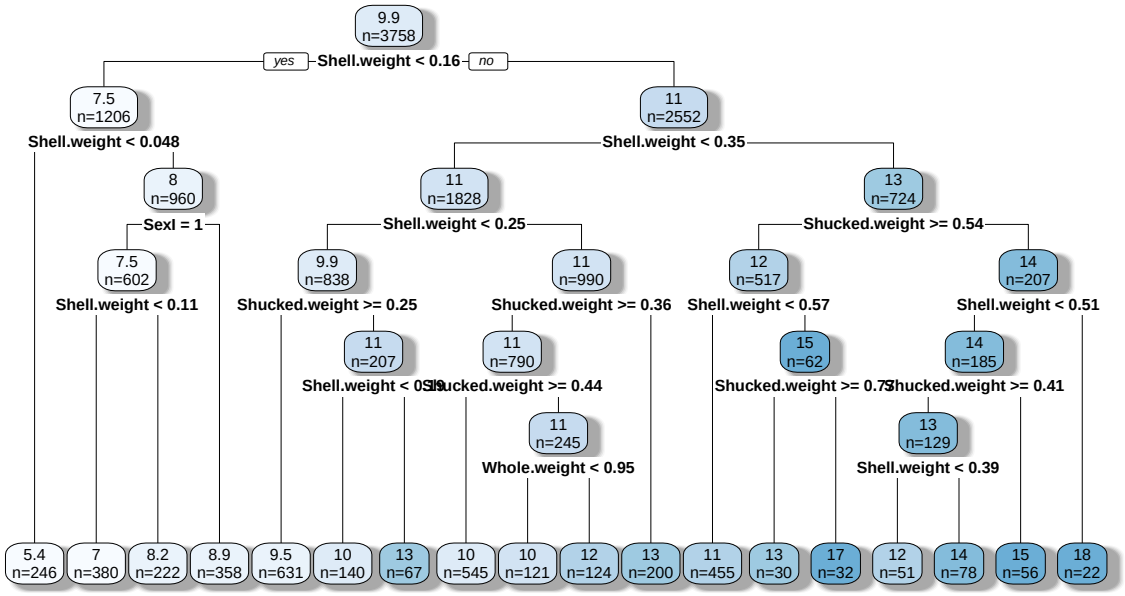
This section trains a regression tree using rpart. The models tuning parameters are selected using cross validation

```
  set.seed(1)
#Train the Model
  rt.Rings <- train(Rings ~ .,
                        method = "rpart",
                        tuneGrid = data.frame(cp = seq(0, 0.1, len = 25)),
                        data = train_set)
 rt.Rings
```

```
## CART
##
## 3758 samples
##    8 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3758, 3758, 3758, 3758, 3758, 3758, ...
## Resampling results across tuning parameters:
##
##   cp           RMSE      Rsquared   MAE
##   0.000000000  2.575410  0.4142168  1.828100
##   0.004166667  2.378968  0.4579461  1.667297
##   0.008333333  2.404039  0.4424883  1.717383
##   0.012500000  2.451803  0.4191284  1.766749
##   0.016666667  2.475604  0.4071113  1.795215
##   0.020833333  2.512535  0.3891735  1.828383
##   0.025000000  2.555001  0.3681976  1.873276
##   0.029166667  2.578112  0.3560986  1.900153
##   0.033333333  2.594560  0.3477250  1.918144
##   0.037500000  2.621801  0.3338870  1.947094
##   0.041666667  2.633216  0.3280049  1.957899
##   0.045833333  2.655284  0.3166424  1.970745
##   0.050000000  2.687375  0.3001221  1.994252
##   0.054166667  2.705126  0.2907663  2.005656
##   0.058333333  2.710384  0.2878195  2.007256
##   0.062500000  2.727703  0.2788173  2.016817
##   0.066666667  2.744363  0.2698416  2.025801
##   0.070833333  2.744363  0.2698416  2.025801
##   0.075000000  2.744363  0.2698416  2.025801
##   0.079166667  2.744363  0.2698416  2.025801
##   0.083333333  2.744363  0.2698416  2.025801
##   0.087500000  2.744363  0.2698416  2.025801
##   0.091666667  2.744363  0.2698416  2.025801
##   0.095833333  2.744363  0.2698416  2.025801
##   0.100000000  2.744363  0.2698416  2.025801
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.004166667.
```

3.2.1 Plot of the Regression Tree

```
#Plot the Regression Tree with Rpart Plot
rpart.plot(rt.Rings$finalModel, type = 2, extra =1,shadow.col = "darkgray",tweak = 1.3)
```



The above diagram shows the regression tree generated and the decisions at each node as well as the number of observations at each

3.2.2 Regression Tree RMSE

Predict Rings using the test set data with the regression tree and generate the RMSE score

```
# 2 Regression Tree using R part
  rt.predictions.Rings <- predict(rt.Rings, test_set)
  rt.RMSE <- RMSE(rt.predictions.Rings,test_set$Rings)
  rt.RMSE
```

```
## [1] 2.325368
```

In this case the Regression tree did not perform better than the linear Regression

3.3 Modal Tree Using Cubist

Model trees improve on regression trees by replacing the leaf nodes with regression models.
This may result in more accurate results than regression trees, which use only a single value for prediction

at the leaf nodes.

Cubist uses the concept of *committees* that develops a series of trees sequentially with adjusted weights. The final prediction is the simple average of predictions from all *committee* members

This section uses Cubist to train a Model Tree. The models tuning parameters *committees* and *neighbo n rs* are selected using cross validation

```
# Modal Tree Using Cubist

# Set the grid to train the model
mt.grid <- expand.grid(committees = c(1, 10, 50, 100),
                       neighbors = c(0, 1, 5, 9))
set.seed(1)
# Set the grid to train the model
Modal_Tree.Rings <- train(Rings ~ .,
                          method = "cubist",
                          data = train_set,
                          tuneGrid = mt.grid,
                          trControl = trainControl(method = "cv")
                           )

# Final Model
  Modal_Tree.Rings
```

```
## Cubist
##
## 3758 samples
##    8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 3382, 3381, 3382, 3381, 3384, 3384, ...
## Resampling results across tuning parameters:
##
##   committees  neighbors  RMSE      Rsquared   MAE
##     1         0          2.155385  0.5579780  1.497184
##     1         1          2.530448  0.4389759  1.781191
##     1         5          2.225833  0.5276994  1.578845
##     1         9          2.179883  0.5432359  1.545428
##    10         0          2.142459  0.5648729  1.487622
##    10         1          2.516072  0.4423892  1.775080
##    10         5          2.215077  0.5311552  1.574292
##    10         9          2.169263  0.5469793  1.539622
##    50         0          2.147160  0.5629388  1.488511
##    50         1          2.517188  0.4416226  1.773562
##    50         5          2.218480  0.5297056  1.573357
##    50         9          2.173127  0.5453591  1.539284
##   100         0          2.148169  0.5622979  1.488647
##   100         1          2.520101  0.4409530  1.773895
##   100         5          2.221079  0.5288464  1.574075
##   100         9          2.175433  0.5445381  1.539656
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were committees = 10 and neighbors = 0.
```

```
  Modal_Tree.Rings$finalModel
```

```
##
## Call:
## cubist.default(x = x, y = y, committees = param$committees)
##
## Number of samples: 3758
## Number of predictors: 9
##
## Number of committees: 10
## Number of rules per committee: 5, 5, 4, 4, 4, 5, 3, 5, 6, 3
```

10 committees and 0 neighbours were selected.

3.3.1 Modal Tree RMSE

Predict Rings using the test set data with the regression tree and generate the RMSE score

```
# 2 Regression Tree using R part
  Modal_tree.predictions.Rings <- predict(Modal_Tree.Rings, test_set)
  Modal_tree.RMSE <- RMSE(Modal_tree.predictions.Rings,test_set$Rings)
  Modal_tree.RMSE
```

```
## [1] 2.002387
```

In this case the Modal Tree performs better than the Regression Tree and the Linear Regression model

3.4 Random Forrest

Random Forests combines the output of multiple decision trees and then generates its own output. Random Forest works on the same principle as decision Tress; however, it does not select all the data points and variables in each of the trees. It randomly samples data points and variables in each of the tree that it creates and then combines the output at the end.

This Section trains a Random Forrest Model

```
# Random Forrest

set.seed(2)
rf.Rings <- randomForest(Rings ~., data = train_set,ntree=500)
rf.Rings
```

```
##
## Call:
##  randomForest(formula = Rings ~ ., data = train_set, ntree = 500)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 4.663218
##                     % Var explained: 55.06
```

3.4.1 Random Forrest RMSE

Use the Random Forrest model to predict Rings with the test set.

```
  # 4 Random Forrest RMSE
  rf.predictions.Rings <- predict(rf.Rings, test_set)
  rf.RMSE <- RMSE(rf.predictions.Rings,test_set$Rings)
  rf.RMSE
```

```
## [1] 2.025158
```

In this case the Random Forrest performed similarly to the Modal Tree

3.5 K Nearest Neighbours Regression Model (KNN)

KNN uses similarity to predict values of any new data points.
This means that the new point is assigned a value based on how closely it resembles the points in the training set.

This Section trains a KNN Model. The Tuning parameter k is selected using cross validation

```
# KNN

set.seed(1)
ctrl <- trainControl(method="repeatedcv",repeats = 10)

knn_grid  <- data.frame(k = seq(3, 30, 2))
knn.Rings <- train(Rings ~ .,
                   method = "knn",
                   data = train_set,
                   tuneGrid = knn_grid
                    )

# Model Training results
formattable(knn.Rings$results,format = "markdown")
```

k	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
3	2.531681	0.4189142	1.775936	0.06278261	0.01910262	0.03185438
5	2.401040	0.4561768	1.691447	0.05337530	0.01942020	0.02810102
7	2.341043	0.4756077	1.646454	0.06422047	0.01879816	0.03189672
9	2.313076	0.4852474	1.624077	0.06885998	0.01883798	0.03331014
11	2.292432	0.4931603	1.606278	0.06808503	0.01702102	0.03176040
13	2.279087	0.4987338	1.595207	0.06845226	0.01585050	0.03130180
15	2.266933	0.5042321	1.585107	0.07123572	0.01615676	0.03376935
17	2.261179	0.5070720	1.578948	0.07398793	0.01667868	0.03595810
19	2.251740	0.5117210	1.570961	0.07473846	0.01568108	0.03678686
21	2.251971	0.5120840	1.568670	0.07339148	0.01528992	0.03522752
23	2.248958	0.5140702	1.565630	0.07517244	0.01557137	0.03635001
25	2.248794	0.5147507	1.563707	0.07750775	0.01625564	0.03634430
27	2.248928	0.5153519	1.562472	0.07727302	0.01669948	0.03570521
29	2.250463	0.5151591	1.562774	0.07699885	0.01661916	0.03544512

```r
# Final Model Chosen
knn.Rings$finalModel
```

```
## 25-nearest neighbor regression model
```

3.5.1 K Nearest Neighbours RMSE

Use the 25-nearest neighbour KNN regression model to predict Rings with the test set.

```r
  # 5 K Nearest Neighbours RMSE
  knn.predictions.Rings <- predict(knn.Rings, test_set)
  knn.RMSE <- RMSE(knn.predictions.Rings,test_set$Rings)
  knn.RMSE
```

```
## [1] 2.072496
```

3.6 Generalized Additive Model (GAM)

GAMs are a class of statistical Model in which the Linear relationship between the Response and Predictors are replaced by several Non-linear smooth functions (Splines) to model and capture the Non linearities in the data.

This section will train a GAM model.

```r
# Generalized Additive Model using Splines
set.seed(1)
gam.Rings <- train(Rings ~ ., method = "gam", data = train_set)
summary(gam.Rings)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ SexI + SexM + s(Height) + s(Diameter) + s(Length) +
##     s(Viscera.weight) + s(Shell.weight) + s(Shucked.weight) +
##     s(Whole.weight)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.10236    0.06610 152.836  < 2e-16 ***
## SexI        -0.57696    0.10947  -5.270 1.44e-07 ***
## SexM         0.05438    0.08450   0.644     0.52
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                     edf Ref.df      F  p-value
## s(Height)         2.283      9  2.658 1.77e-06 ***
## s(Diameter)       4.797      9  1.378  0.00546 **
## s(Length)         3.997      9  2.253 2.40e-05 ***
```

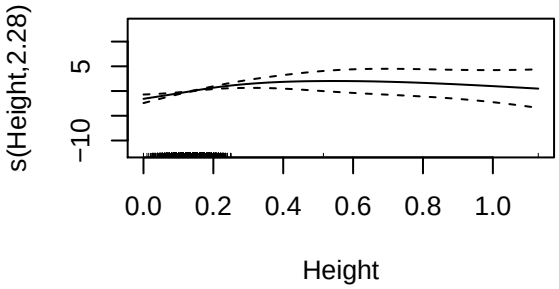
```
## s(Viscera.weight) 5.153      9  8.607  < 2e-16 ***
## s(Shell.weight)   6.762      9 11.374  < 2e-16 ***
## s(Shucked.weight) 5.966      9 68.024  < 2e-16 ***
## s(Whole.weight)   6.601      9 27.550  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.572   Deviance explained = 57.7%
## GCV = 4.4844  Scale est. = 4.4383    n = 3758
```

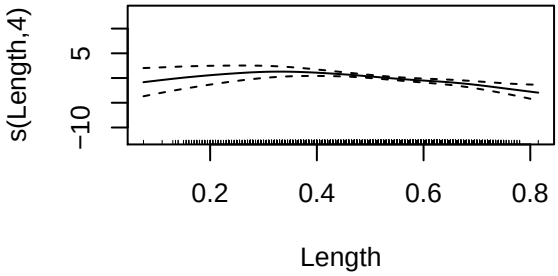
```
gam.Rings$finalModel
```

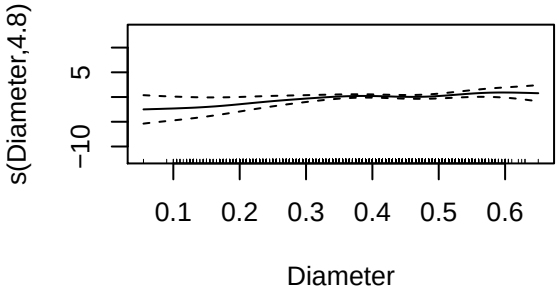
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ SexI + SexM + s(Height) + s(Diameter) + s(Length) +
##     s(Viscera.weight) + s(Shell.weight) + s(Shucked.weight) +
##     s(Whole.weight)
##
## Estimated degrees of freedom:
## 2.28 4.80 4.00 5.15 6.76 5.97 6.60
##  total = 38.56
##
## GCV score: 4.48435
```

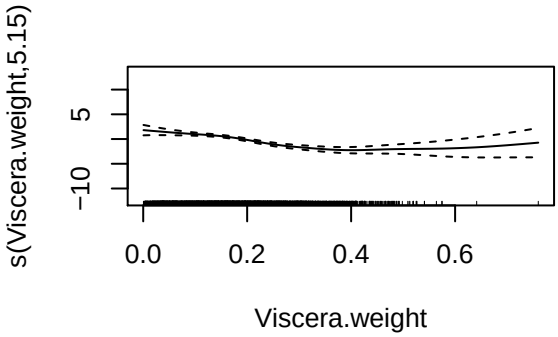
The R squared of the GAM model is higher than that of the linear regression meaning this model can explain more of the Variation in Rings than the linear regression.

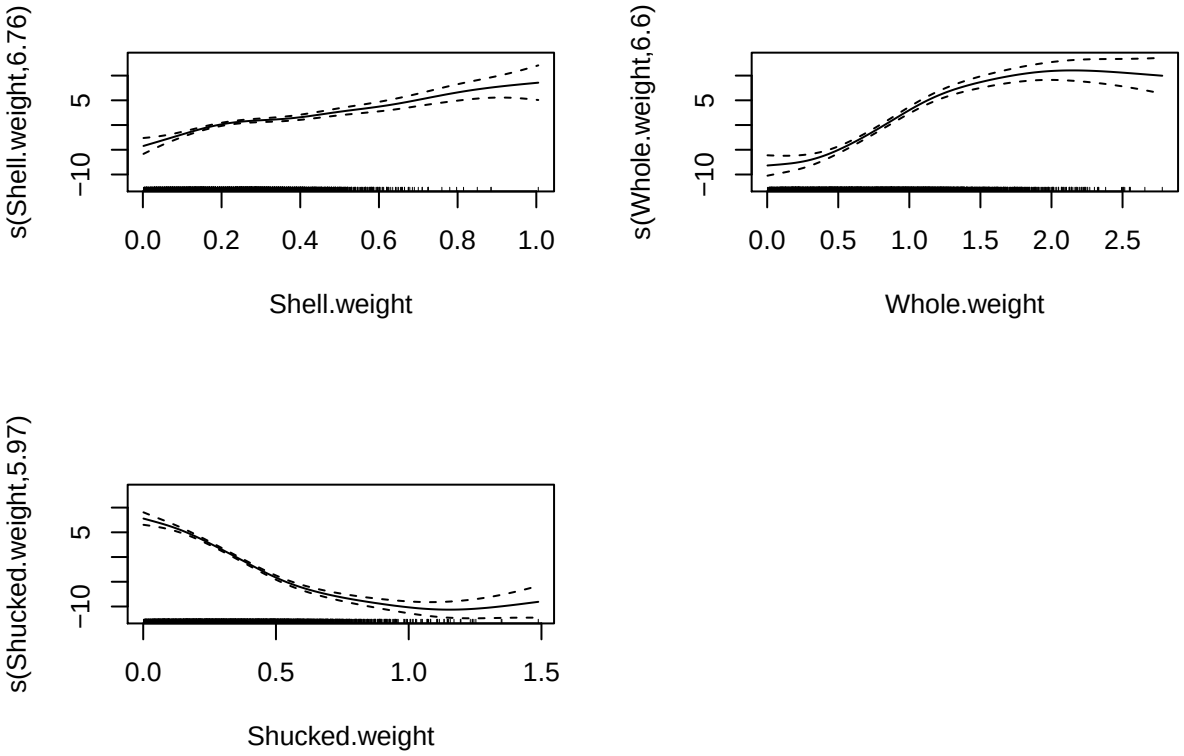
```
layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page
plot(gam.Rings$finalModel)
```


s(Height,2.28)
5
−10
0.0
0.2
0.4
0.6
0.8
1.0
Height


s(Length,4)
5
−10
0.2
0.4
0.6
0.8
Length


s(Diameter,4.8)
5
−10
0.1
0.2
0.3
0.4
0.5
0.6
Diameter


s(Viscera.weight,5.15)
5
−10
0.0
0.2
0.4
0.6
Viscera.weight



The above shows plots for each variable included in the Model.
The X-axis contains the variable values itself and the Y-axis contains the Response values Rings
The curvy shapes for each of the variables is due to the Smoothing splines which models the Non linearities in the data.
The dotted Lines around the main curve lines are the Standard Error Bands.

GAM appears to be an effective way of fitting Non-linear functions on several variables

3.6.1 Generalized Additive Model using Splines

Use the GAM model to predict Rings with the test set and generate the RMSE score.

```
# Generalized Additive Model using Splines
gam.predictions.Rings <- predict(gam.Rings, test_set)
gam.predictions.Rings.RMSE <- RMSE(gam.predictions.Rings,test_set$Rings)
gam.predictions.Rings.RMSE
```

```
## [1] 2.055675
```

3.7 Multivariate Adaptive Regression Splines (MARS) Model

This algorithm can capture any nonlinearity aspects of regression by assessing cut points called knots (similar to step functions). The procedure assesses each data point for each predictor as a knot and creates a linear regression model with the candidate feature(s).

This section will train a MARS model. Tuning parameters are selected using cross validation

```
# Create a grid needed to pick the tuning parameters via cross validation
 hyper_grid <- expand.grid(
   degree = 1:3,
   nprune = seq(2, 100, length.out = 10) %>% floor()
 )

   earth.Rings_Tuned <- train(
    Rings ~ .,
     method = "earth",
     metric = "RMSE",
     trControl = trainControl(method = "cv", number = 10),
     tuneGrid = hyper_grid,
     data = train_set
   )

 #  Show the best model best tune
   earth.Rings_Tuned$bestTune
```

```
##    nprune degree
## 12     12      2
```

```
 #  Show the final model
  earth.Rings_Tuned$finalModel
```

```
## Selected 12 of 20 terms, and 7 of 9 predictors
## Termination condition: Reached nk 21
## Importance: Whole.weight, Shucked.weight, Height, Viscera.weight, ...
## Number of terms at each degree of interaction: 1 4 7
## GCV 4.460136    RSS 16507.93    GRSq 0.5703903    RSq 0.5766565
```

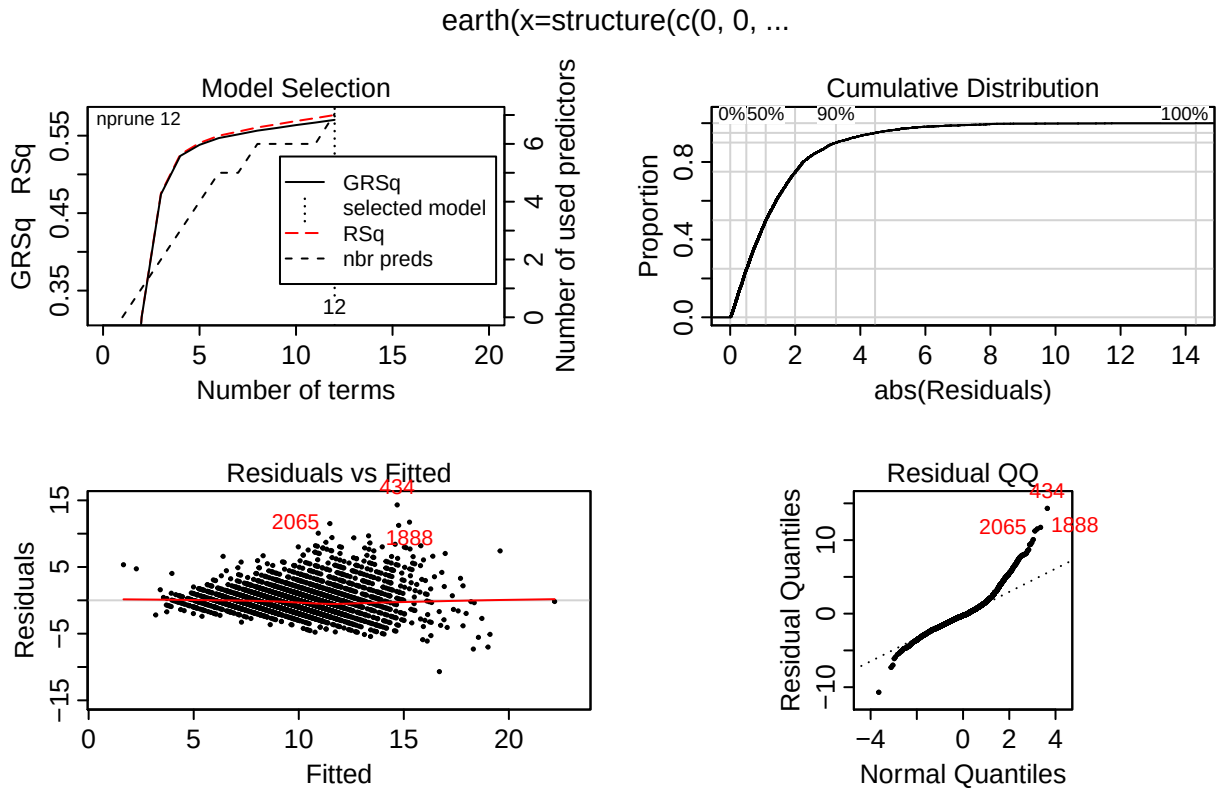
```
    summary(earth.Rings_Tuned$finalModel)
```

```
## Call: earth(x=matrix[3758,9], y=c(15,7,9,10,7,8...), keepxy=TRUE,
##             degree=2, nprune=12)
##
##                                                 coefficients
## (Intercept)                                        10.61967
## h(1.8075-Whole.weight)                            -16.89375
## h(0.813-Shucked.weight)                            33.35761
## h(0.385-Viscera.weight)                            13.87801
## h(0.16-Shell.weight)                              -20.54850
## SexI * h(0.813-Shucked.weight)                     -1.14009
## h(Length-0.395) * h(0.813-Shucked.weight)         -18.29862
## h(0.215-Height) * h(0.813-Shucked.weight)         -31.02632
## h(1.8075-Whole.weight) * h(Shucked.weight-0.6115)  43.02398
## h(1.6935-Whole.weight) * h(Shell.weight-0.16)      12.18550
## h(Whole.weight-1.6935) * h(Shell.weight-0.16)       8.33375
## h(Shucked.weight-0.2745) * h(0.16-Shell.weight)   435.96630
##
## Selected 12 of 20 terms, and 7 of 9 predictors
```

```
## Termination condition: Reached nk 21
## Importance: Whole.weight, Shucked.weight, Height, Viscera.weight, ...
## Number of terms at each degree of interaction: 1 4 7
## GCV 4.460136    RSS 16507.93    GRSq 0.5703903    RSq 0.5766565
```

3.7.1 Plot the MARS Model

```
layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page
plot( earth.Rings_Tuned$finalModel)
```



3.7.2 MARS

Use the MARS model to predict Rings with the test set.

```
  # Multivariate Adaptive Regression Splines MARS
  earth.predictions.Rings <- predict(earth.Rings_Tuned, test_set)
  earth.RMSE <- RMSE(earth.predictions.Rings,test_set$Rings)
  earth.RMSE
```

```
## [1] 2.010556
```

3.8 Base Model Results

The below charts show the RMSE scores of each of the Base models trained with the train set data and predicting Rings with the test set data

```
# Make predictions on test set and compare RMSE

   model_results <- as.data.frame(

      rbind(
         c(model = "Linear Regression"
           , RMSE =  round(lm.RMSE,6))
        ,c(model = "Regression Tree using rpart"
           , RMSE =  round(rt.RMSE,6))
        ,c(model = "Modal Tree using Cubist"
           , RMSE = round(Modal_tree.RMSE,6))
        ,c(model = "Random Forrest"
           , RMSE =  round(rf.RMSE,6))
        ,c(model = "K Nearest Neighbours"
           , RMSE =  round(knn.RMSE,6))
        ,c(model = "Generalized Additive Model using Splines GAM"
           , RMSE =  round(gam.predictions.Rings.RMSE,6))
        ,c(model = "Multivariate Adaptive Regression Splines MARS"
           , RMSE =  round(earth.RMSE,6))
        )
     )


   formattable(model_results, format = "markdown")
```

model	RMSE
Linear Regression	2.121661
Regression Tree using rpart	2.325368
Modal Tree using Cubist	2.002387
Random Forrest	2.025158
K Nearest Neighbours	2.072496
Generalized Additive Model using Splines GAM	2.055675
Multivariate Adaptive Regression Splines MARS	2.010556

3.8.1 Observations

- All models except the rpart Regression Tree have outperformed the Linear Regression
- The best performing Model is the Modal Tree using Cubist, closely followed by the MARS Model.

4 Ensemble Models to improve RMSE

In order to improve upon the Base Models, this section will train Ensemble models which take predictions of the base models as an input to generate the final prediction.

The First is a simple average of all the predictions, while the second uses a Linear regression across the predictions of each base model generate a final prediction.

4.1 Ensemble 1 - Simple Average.

This Section makes predictions on the Test Set by Averaging each Prediction from the 7 Base Models

```
# Matrix of predictions of each of the base models for the test set data
preds_v1 <- as.data.frame(cbind(lm.predictions.Rings
              ,rt.predictions.Rings
              ,Modal_tree.predictions.Rings
              ,rf.predictions.Rings
              ,knn.predictions.Rings
              ,gam.predictions.Rings
              ,earth.predictions.Rings))

names(preds_v1) <- c("lm","rpart","Cubist","rf","knn","gam","Mars")


# Ensemble 1 Predictions
ens_v1.pred = as.matrix(rowMeans(preds_v1))

#Ensemble 1 RMSE
ens_v1.RMSE <- RMSE(ens_v1.pred,test_set$Rings)
ens_v1.RMSE
```

```
## [1] 2.019524
```

In this Case the Ensemble model 1 did not outperform the Modal Tree.

4.2 Ensemble 2 - Linear Regression

This section will create a more sophisticated ensemble model by which the predictions of the base models are taken as inputs for a linear regression ensemble which outputs a final Rings prediction.

We cannot use the test set data to train the ensemble model in any circumstance
This means that in order to train the ensemble regression model, we need to take the train data and create k folds of train subsets (train_cv) and test subsets (test_cv) which will be used to generate new base model predictions as an input to train the Ensemble

4.2.1 Create Subsets of the train set data

This section creates 5 folds of train_cv and test_cv subsets of the train set data

```
# Ensemble Cross Validations

# Create 5 folds from the train set data
set.seed(5)
train_set_index_cv <- createDataPartition(train_set$Rings
                                          , times = 5, p = 0.1, list = FALSE)
train_cv1 <- train_set %>% slice(-train_set_index_cv[,1])
test_cv1 <- train_set %>% slice(train_set_index_cv[,1])

train_cv2 <- train_set %>% slice(-train_set_index_cv[,2])
test_cv2 <- train_set %>% slice(train_set_index_cv[,2])
```

```
train_cv3 <- train_set %>% slice(-train_set_index_cv[,3])
test_cv3 <- train_set %>% slice(train_set_index_cv[,3])

train_cv4 <- train_set %>% slice(-train_set_index_cv[,4])
test_cv4 <- train_set %>% slice(train_set_index_cv[,4])

train_cv5 <- train_set %>% slice(-train_set_index_cv[,5])
test_cv5 <- train_set %>% slice(train_set_index_cv[,5])
```

4.3 Fit each Base Model with the new train_cv subsets and predict using the test_cv subsets.

This section retrains each base model with each of the 5 folds of train_cv and test_cv data

```
 ################################################
# This code can take several minutes to run
################################################

# Fit each model to the train_cv folds
# Predict on each of the test_cv folds

# Backwards Stepped Linear Regression Model

lm.ens.cv1 <- stepAIC(lm(Rings ~., data = train_cv1), direction="backward")
lm.ens.cv2 <- stepAIC(lm(Rings ~., data = train_cv2), direction="backward")
lm.ens.cv3 <- stepAIC(lm(Rings ~., data = train_cv3), direction="backward")
lm.ens.cv4 <- stepAIC(lm(Rings ~., data = train_cv4), direction="backward")
lm.ens.cv5 <- stepAIC(lm(Rings ~., data = train_cv5), direction="backward")

pred_lm.ens.cv1 <- predict(lm.ens.cv1, test_cv1)
pred_lm.ens.cv2 <- predict(lm.ens.cv2, test_cv2)
pred_lm.ens.cv3 <- predict(lm.ens.cv3, test_cv3)
pred_lm.ens.cv4 <- predict(lm.ens.cv4, test_cv4)
pred_lm.ens.cv5 <- predict(lm.ens.cv5, test_cv5)

# rpart Regression Tree

rt.ens.cv1 <- train(Rings ~ ., method = "rpart"
                    , tuneGrid = data.frame(cp = seq(0, 0.1, len = 25)), data = train_cv1)
rt.ens.cv2 <- train(Rings ~ ., method = "rpart"
                    , tuneGrid = data.frame(cp = seq(0, 0.1, len = 25)), data = train_cv2)
rt.ens.cv3 <- train(Rings ~ ., method = "rpart"
                    , tuneGrid = data.frame(cp = seq(0, 0.1, len = 25)), data = train_cv3)
rt.ens.cv4 <- train(Rings ~ ., method = "rpart"
                    , tuneGrid = data.frame(cp = seq(0, 0.1, len = 25)), data = train_cv4)
rt.ens.cv5 <- train(Rings ~ ., method = "rpart"
                    , tuneGrid = data.frame(cp = seq(0, 0.1, len = 25)), data = train_cv5)

pred_rt.ens.cv1 <- predict(rt.ens.cv1, test_cv1)
pred_rt.ens.cv2 <- predict(rt.ens.cv2, test_cv2)
pred_rt.ens.cv3 <- predict(rt.ens.cv3, test_cv3)
pred_rt.ens.cv4 <- predict(rt.ens.cv4, test_cv4)
```

```
pred_rt.ens.cv5 <- predict(rt.ens.cv5, test_cv5)

# Modal Tree using Cubist

Modal_Tree.ens.cv1 <- train(Rings ~ ., method = "cubist"
                            , data = train_cv1, tuneGrid = mt.grid
                            ,trControl = trainControl(method = "cv"))
Modal_Tree.ens.cv2 <- train(Rings ~ ., method = "cubist"
                            , data = train_cv2, tuneGrid = mt.grid
                            ,trControl = trainControl(method = "cv"))
Modal_Tree.ens.cv3 <- train(Rings ~ ., method = "cubist"
                            , data = train_cv3, tuneGrid = mt.grid
                            ,trControl = trainControl(method = "cv"))
Modal_Tree.ens.cv4 <- train(Rings ~ ., method = "cubist"
                            , data = train_cv4, tuneGrid = mt.grid
                            ,trControl = trainControl(method = "cv"))
Modal_Tree.ens.cv5 <- train(Rings ~ ., method = "cubist"
                            , data = train_cv5, tuneGrid = mt.grid
                            ,trControl = trainControl(method = "cv"))

pred_Modal_Tree.ens.cv1 <- predict(Modal_Tree.ens.cv1, test_cv1)
pred_Modal_Tree.ens.cv2 <- predict(Modal_Tree.ens.cv2, test_cv2)
pred_Modal_Tree.ens.cv3 <- predict(Modal_Tree.ens.cv3, test_cv3)
pred_Modal_Tree.ens.cv4 <- predict(Modal_Tree.ens.cv4, test_cv4)
pred_Modal_Tree.ens.cv5 <- predict(Modal_Tree.ens.cv5, test_cv5)

# Random Forrest

rf.cv1 <- randomForest(Rings ~., data = train_cv1,ntree=500)
rf.cv2 <- randomForest(Rings ~., data = train_cv2,ntree=500)
rf.cv3 <- randomForest(Rings ~., data = train_cv3,ntree=500)
rf.cv4 <- randomForest(Rings ~., data = train_cv4,ntree=500)
rf.cv5 <- randomForest(Rings ~., data = train_cv5,ntree=500)

pred_rf.ens.cv1 <- predict(rf.cv1, test_cv1)
pred_rf.ens.cv2 <- predict(rf.cv2, test_cv2)
pred_rf.ens.cv3 <- predict(rf.cv3, test_cv3)
pred_rf.ens.cv4 <- predict(rf.cv4, test_cv4)
pred_rf.ens.cv5 <- predict(rf.cv5, test_cv5)

# K Nearest Neighbours KNN

knn.cv1 <- train(Rings ~ .,method = "knn", data = train_cv1,tuneGrid = knn_grid)
knn.cv2 <- train(Rings ~ .,method = "knn", data = train_cv2,tuneGrid = knn_grid)
knn.cv3 <- train(Rings ~ .,method = "knn", data = train_cv3,tuneGrid = knn_grid)
knn.cv4 <- train(Rings ~ .,method = "knn", data = train_cv4,tuneGrid = knn_grid)
knn.cv5 <- train(Rings ~ .,method = "knn", data = train_cv5,tuneGrid = knn_grid)

pred_knn.ens.cv1 <- predict(knn.cv1, test_cv1)
pred_knn.ens.cv2 <- predict(knn.cv2, test_cv2)
pred_knn.ens.cv3 <- predict(knn.cv3, test_cv3)
pred_knn.ens.cv4 <- predict(knn.cv4, test_cv4)
pred_knn.ens.cv5 <- predict(knn.cv5, test_cv5)
```

```r
# Generalized Additive Model using Splines (GAM)

gam.cv1 <- train(Rings ~ ., method = "gam", data = train_cv1)
gam.cv2 <- train(Rings ~ ., method = "gam", data = train_cv2)
gam.cv3 <- train(Rings ~ ., method = "gam", data = train_cv3)
gam.cv4 <- train(Rings ~ ., method = "gam", data = train_cv4)
gam.cv5 <- train(Rings ~ ., method = "gam", data = train_cv5)

pred_gam.ens.cv1 <- predict(gam.cv1, test_cv1)
pred_gam.ens.cv2 <- predict(gam.cv2, test_cv2)
pred_gam.ens.cv3 <- predict(gam.cv3, test_cv3)
pred_gam.ens.cv4 <- predict(gam.cv4, test_cv4)
pred_gam.ens.cv5 <- predict(gam.cv5, test_cv5)

### Multivariate Adaptive Regression Splines (MARS)

earth.cv1 <- train(Rings ~ .,method = "earth",metric = "RMSE"
                   ,trControl = trainControl(method = "cv"
                                             , number = 10),tuneGrid = hyper_grid,data = train_cv1)
earth.cv2 <- train(Rings ~ .,method = "earth",metric = "RMSE"
                   ,trControl = trainControl(method = "cv"
                                             , number = 10),tuneGrid = hyper_grid,data = train_cv2)
earth.cv3 <- train(Rings ~ .,method = "earth",metric = "RMSE"
                   ,trControl = trainControl(method = "cv"
                                             , number = 10),tuneGrid = hyper_grid,data = train_cv3)
earth.cv4 <- train(Rings ~ .,method = "earth",metric = "RMSE"
                   ,trControl = trainControl(method = "cv"
                                             , number = 10),tuneGrid = hyper_grid,data = train_cv4)
earth.cv5 <- train(Rings ~ .,method = "earth",metric = "RMSE"
                   ,trControl = trainControl(method = "cv"
                                             , number = 10),tuneGrid = hyper_grid,data = train_cv5)

pred_earth.ens.cv1 <- predict(earth.cv1, test_cv1)
pred_earth.ens.cv2 <- predict(earth.cv2, test_cv2)
pred_earth.ens.cv3 <- predict(earth.cv3, test_cv3)
pred_earth.ens.cv4 <- predict(earth.cv4, test_cv4)
pred_earth.ens.cv5 <- predict(earth.cv5, test_cv5)
```

Create a matrix of predictions for each test_cv subset

```r
# Create a matrix of predictions
cv1 <- as.matrix(cbind(pred_lm.ens.cv1
                       ,pred_rt.ens.cv1
                       ,pred_Modal_Tree.ens.cv1
                       ,pred_rf.ens.cv1
                       ,pred_knn.ens.cv1
                       ,pred_gam.ens.cv1
                       ,pred_earth.ens.cv1
                       ,test_cv1$Rings))

cv2 <- as.matrix(cbind(pred_lm.ens.cv2
                       ,pred_rt.ens.cv2
                       ,pred_Modal_Tree.ens.cv2
```

```
                        ,pred_rf.ens.cv2
                        ,pred_knn.ens.cv2
                        ,pred_gam.ens.cv2
                        ,pred_earth.ens.cv2
                        ,test_cv2$Rings))

cv1 <- as.matrix(cbind(pred_lm.ens.cv1
                        ,pred_rt.ens.cv1
                        ,pred_Modal_Tree.ens.cv1
                        ,pred_rf.ens.cv1
                        ,pred_knn.ens.cv1
                        ,pred_gam.ens.cv1
                        ,pred_earth.ens.cv1
                        ,test_cv1$Rings))

cv3 <- as.matrix(cbind(pred_lm.ens.cv3
                        ,pred_rt.ens.cv3
                        ,pred_Modal_Tree.ens.cv3
                        ,pred_rf.ens.cv3
                        ,pred_knn.ens.cv3
                        ,pred_gam.ens.cv3
                        ,pred_earth.ens.cv3
                        ,test_cv3$Rings))

cv4 <- as.matrix(cbind(pred_lm.ens.cv4
                        ,pred_rt.ens.cv4
                        ,pred_Modal_Tree.ens.cv4
                        ,pred_rf.ens.cv4
                        ,pred_knn.ens.cv4
                        ,pred_gam.ens.cv4
                        ,pred_earth.ens.cv4
                        ,test_cv4$Rings))

cv5 <- as.matrix(cbind(pred_lm.ens.cv5
                        ,pred_rt.ens.cv5
                        ,pred_Modal_Tree.ens.cv5
                        ,pred_rf.ens.cv5
                        ,pred_knn.ens.cv5
                        ,pred_gam.ens.cv5
                        ,pred_earth.ens.cv5
                        ,test_cv5$Rings))
# Combine in a single dataframe
full_set_cv <- as.data.frame(rbind(cv1,cv2,cv3,cv4,cv5))
names(full_set_cv) <- c("lm","rpart","Cubist","rf","knn","gam","Mars","Rings")
```

4.3.1 Train Ensemble 2 on the test_cv predictions

The Ensemble regression can now be trained using the test_cv subset, and the Rings predictions form all 7 base models

```
# Train the Ensemble 2 Linear Regression model
# Use stepwise model selection 'stepAIC'
ens.2.lm <- stepAIC(lm(Rings ~., data = full_set_cv), direction="both")
```

```
## Start:  AIC=2878.8
## Rings ~ lm + rpart + Cubist + rf + knn + gam + Mars
##
##          Df Sum of Sq    RSS    AIC
## - gam     1     0.054 8607.7 2876.8
## - rpart   1     5.021 8612.7 2877.9
## - Mars    1     6.792 8614.5 2878.3
## - lm      1     6.885 8614.6 2878.3
## <none>                8607.7 2878.8
## - Cubist  1    10.652 8618.3 2879.1
## - knn     1    13.211 8620.9 2879.7
## - rf      1   107.868 8715.6 2900.3
##
## Step:  AIC=2876.81
## Rings ~ lm + rpart + Cubist + rf + knn + Mars
##
##          Df Sum of Sq    RSS    AIC
## - rpart   1     4.994 8612.7 2875.9
## - lm      1     6.901 8614.6 2876.3
## <none>                8607.7 2876.8
## - Cubist  1    11.521 8619.3 2877.3
## - knn     1    13.183 8620.9 2877.7
## + gam     1     0.054 8607.7 2878.8
## - Mars    1    25.053 8632.8 2880.3
## - rf      1   111.693 8719.4 2899.1
##
## Step:  AIC=2875.91
## Rings ~ lm + Cubist + rf + knn + Mars
##
##          Df Sum of Sq    RSS    AIC
## - lm      1     6.113 8618.9 2875.2
## <none>                8612.7 2875.9
## - knn     1    10.718 8623.5 2876.2
## - Cubist  1    11.063 8623.8 2876.3
## + rpart   1     4.994 8607.7 2876.8
## + gam     1     0.026 8612.7 2877.9
## - Mars    1    25.464 8638.2 2879.5
## - rf      1   107.862 8720.6 2897.4
##
## Step:  AIC=2875.25
## Rings ~ Cubist + rf + knn + Mars
##
##          Df Sum of Sq    RSS    AIC
## <none>                8618.9 2875.2
## - knn     1     9.937 8628.8 2875.4
## + lm      1     6.113 8612.7 2875.9
## + rpart   1     4.206 8614.6 2876.3
## + gam     1     0.039 8618.8 2877.2
## - Mars    1    32.015 8650.9 2880.2
## - Cubist  1    44.759 8663.6 2883.0
## - rf      1   103.072 8721.9 2895.7
```

```
#Summary Linear Regression
summary(ens.2.lm)
```

```
##
## Call:
## lm(formula = Rings ~ Cubist + rf + knn + Mars, data = full_set_cv)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.8853 -1.3058 -0.2851  0.8977 10.8574
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.31532    0.23950  -1.317  0.18815
## Cubist       0.28006    0.08963   3.125  0.00181 **
## rf           0.42263    0.08913   4.742 2.28e-06 ***
## knn          0.13011    0.08837   1.472  0.14111
## Mars         0.21139    0.07999   2.643  0.00830 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.141 on 1880 degrees of freedom
## Multiple R-squared:  0.5589, Adjusted R-squared:  0.5579
## F-statistic: 595.4 on 4 and 1880 DF,  p-value: < 2.2e-16
```

4.3.2 Observations

The Ensemble 2 regression model uses predictions from the Cubist Modal Tree, Random Forrest, KNN and Mars Models. It does not use the linear regression, GAM or rpart model.

4.3.3 Predict Rings with Ensemble 2 Linear Regression

Now Predict Rings using the Test set data and Generate the RMSE Score

```
#Ensemble 2 Predictions
# Predict using the predictions of each base models predictions on the full test set data
ens.2.pred <- predict(ens.2.lm, preds_v1)
ens.2.RMSE <- RMSE(ens.2.pred,test_set$Rings)

#Ensemble 2 RMSE
ens.2.RMSE
```

```
## [1] 1.975053
```

5 Results

This section combines the results of each model in a table

```
# Show the RMSE scores of each model in a single table
 model_results_FINAL <- as_tibble(

   rbind(
      c(model = "1 Linear Regression"
        , RMSE =  round(lm.RMSE,4))
```

```
      ,c(model = "2 Regression Tree using rpart"
         , RMSE =  round(rt.RMSE,4))
      ,c(model = "3 Modal Tree using Cubist"
         , RMSE = round(Modal_tree.RMSE,4))
      ,c(model = "4 Random Forrest"
         , RMSE =  round(rf.RMSE,4))
      ,c(model = "5 K Nearest Neighbours"
         , RMSE =  round(knn.RMSE,4))
      ,c(model = "6 Generalized Additive Model using Splines GAM"
         , RMSE =  round(gam.predictions.Rings.RMSE,4))
      ,c(model = "7 Multivariate Adaptive Regression Splines MARS"
         , RMSE =  round(earth.RMSE,4))
      ,c(model = "Ensemble 1 - average of base model predictions "
         , RMSE =  round(ens_v1.RMSE,4))
      ,c(model = "Ensemble 2 -Stepped Linear Regression of base model predictions"
         , RMSE =  round(ens.2.RMSE,4))
    )
  )
   # Format the table
     formattable(model_results_FINAL ,align = c("l","r")
                 , format = "markdown",list(`model` = formatter(
       "span", style = ~ style(color = "grey",font.weight = "bold"))
     ))
```

model	RMSE
1 Linear Regression	2.1217
2 Regression Tree using rpart	2.3254
3 Modal Tree using Cubist	2.0024
4 Random Forrest	2.0252
5 K Nearest Neighbours	2.0725
6 Generalized Additive Model using Splines GAM	2.0557
7 Multivariate Adaptive Regression Splines MARS	2.0106
Ensemble 1 - average of base model predictions	2.0195
Ensemble 2 -Stepped Linear Regression of base model predictions	1.9751

6 Conclusion

The above Table shows the RMSE of each models predictions for Rings using the test set data.
The Ensemble Model which combines the predictions of the base models with a linear regression outperforms each individual model
Ensemble 2 is the recommended model to use to predict the age of Abalone

In conclusion. It is possible to predict the age of Abalone by using physical dimensions, weights and sex. This can be used to save time and costs provided the RMSE is within acceptable tolerances for those in the industry.

6.0.1 Additional thoughts

In addition the below table shows RMSE scores of the Fina Ensemble 2 model predictions split by Sex. It shows that the RMSE is lower when predicting the age of Infant Abalone than it is when Predicting Male

or Female Abalone.
Future improvements to this prediction model should focus features which may improve predictions of older Female and Male Abalone

Sex	RMSE
Female	2.516
Male	1.824
Infant	1.503