Abalone Age Predictive Modeling

Jose Luis Estrada, Nava Roohi, and Ashutosh Singh

Shiley-Marcos School of Engineering

University of San Diego

Define the Purpose

The purpose of this project is to be able to assist the abalone researchers in predicting the age of abalone from its physical measurement. Therefore, the hypothesis and scope of this analysis can be framed as: Can predictive modeling and data mining be used to predict the age of abalones using physical measurements. The outcomes of this study will help researchers have a more accurate prediction of age by reducing the time of the actual process.

Background

Abalones are species of marine snails that are more likely to be found in coastal waters. Like other unveiled snails, it has a single shell on top. However, its shell is flat and spiral-shaped with small holes around its edges. The current process to determine the age of the abalone is cutting the shell through the cone, staining it, and counting the number of rings through the microscope. As it sounds, it's a complicated and monotonous process. However, other measurements (e.g., height, shell weight) are much easier to obtain and predict age. These physical measurements provided in a dataset could pave the way for more accurate age prediction. This is the primary motivation for investing time, effort, and resources into developing these machine learning algorithms. The application of data mining toward predicting the age of these unique and popular species could be insightful and advantageous. It could help the researchers better understand these species and the ways to protect them to live longer.

Obtain Data

The dataset was originally provided by the University of California Irvine Repository and hosted on Kaggle. In total, the dataset has 4,177 observations and nine columns, which are

composed of a single categorical attribute, which is Sex, and eight numerical attributes. The eight numerical attributes are about abalone's physical measurement: Length, Diameters, Weight, Whole Weight, Shucked Weight, Viscera Weight, and Shell Weight. The last column, named "Rings," is the target variable of interest and our dependent variable. However, we replaced the attribute Rings with "Age" since each ring equals the number of rings plus one and a half years (Hossain). This will be helpful and efficient when the dataset is split into training and test sets at a later stage of the project.

Table 1.

Glossary of Attribute in the Dataset

Name	**Description**
Sex	M=Male F=Female I=Infant
Length	Longest shell measurement in (mm)
Diameter	Measurement perpendicular to length in (mm)
Height	With meat in shell in (mm)
Whole weight	Whole abalone (in grams)
Shucked weight	Weight of meat (in grams)
Viscera weight	Gut weight (after bleeding)
Shell weight	After being dried (in grams)
Rings	+1.5 gives the age in years

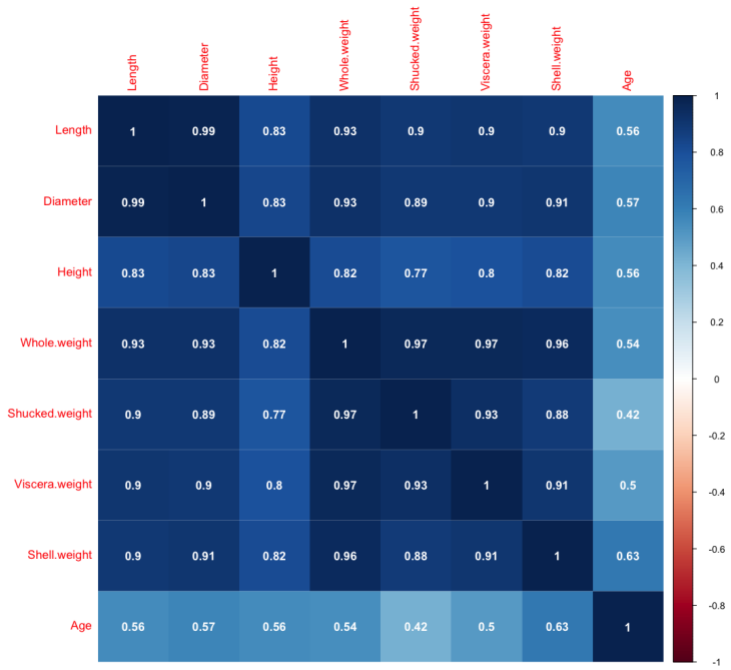
Note. These are the original attributes included in the dataset from UCI Machine Learning Repository.

Exploratory Data Analysis

A thorough exploratory analysis of the data was then conducted. To begin with the exploration, it was identified that there were no missing values in the dataset. However, since our target variable is to predict age, we replaced the variable "Rings" by adding 1.5 to the number of rings (Hossain). After this process, we verified that the data type was correct, which resulted in one categorical attribute and eight numeric variables. As for the numerical variables, we had to check the correlation between each other to see if we could potentially remove one of the variables for the modeling portion.

Figure 1.

Correlation Plot for Numerical Attributes



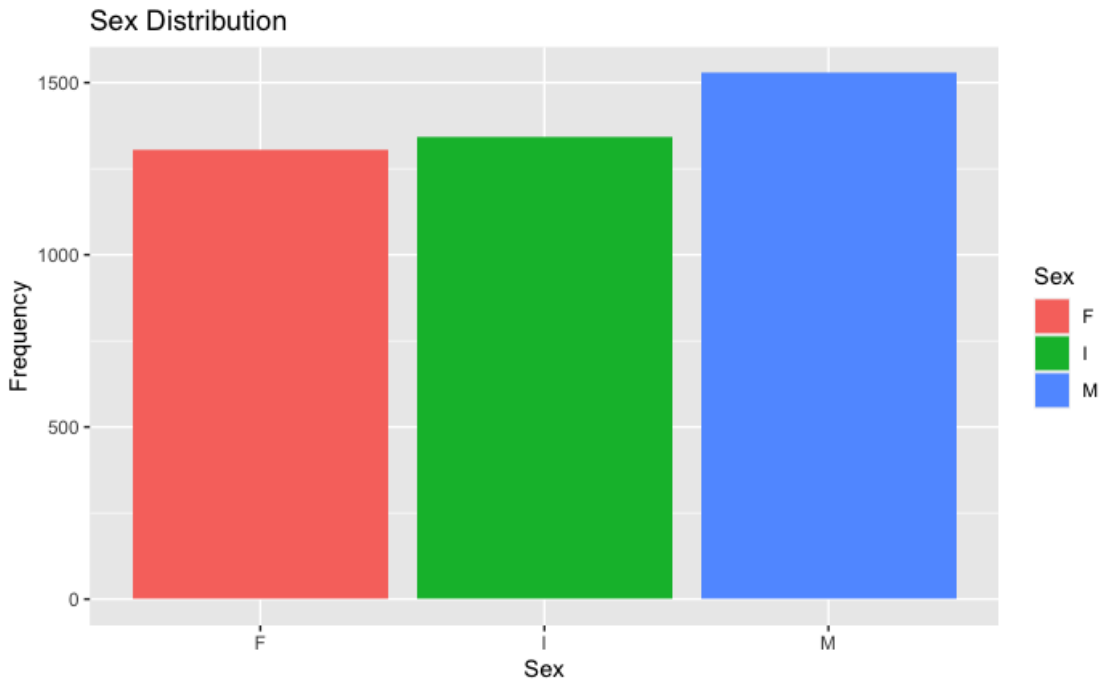
Note. The correlation plot also includes the target variable to test correlation with independent variables.

The correlation plot illustrates a high correlation within independent variables. Still, since the correlation values are remarkably close, it is complicated to choose which variable to eliminate. Therefore, we decided to keep them at this point of the analysis and use different statistical tools to verify this decision.

For the categorical variable, we decided to use a histogram to understand the distribution for "Sex." This will help us see if each sex response is well represented.

Figure 2.

Categorical Attribute "Sex" Distribution



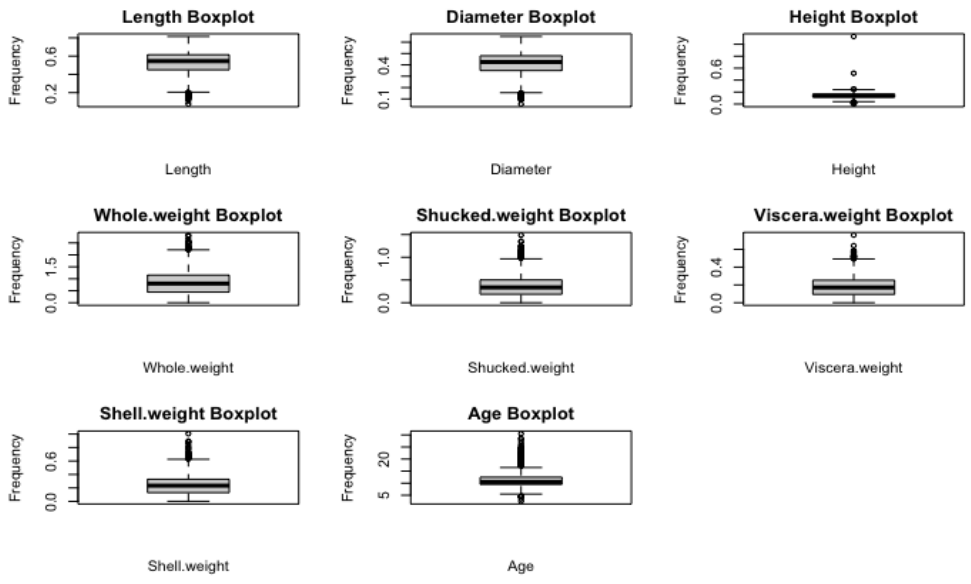
Note. The frequencies for each sex option are very similar.

The histogram helped us understand that there is a similar frequency between the types of responses. Each response is repeated between 1250 and 1500 instances, looking almost to a uniform distribution. This is an excellent representation to input into the model because it will avoid bias in terms of the sex of the abalone.

For the numerical values, we decided to start looking at different distributions as well to see if they were normal or if they had a skewed distribution. We quickly saw that all the attributes were normally distributed with different skewness by the first look. Variables like Length and Diameter were the only attributes that were skewed to the right. Based on the frequency for each bar, it does not appear to have outliers that concern us, but we can verify this at a later step of the analysis. The rest of the attributes were skewed to the right, but at first glance, Height and Age had some concerning results based on the skewness. So, we decided to create boxplots and see where the outliers are between the distributions to take a closer look.

Figure 3.

Numerical Variables Boxplots and Outliers



Note. Taking a closer look to the outliers between numerical attributes distributions

There are outliers within each boxplot, but erasing them could not be a realistic representation of our final model. However, the Height Boxplot shows two significant outliers, so we decided to remove the outliers from this attribute to help us make better predictions in our modeling portion of the project.

Data Splitting

Since our target variable is Age, and most of the attributes are numeric, we decided to use regression models. Sex is the only attribute that is categorical with almost a uniform distribution, so we decided not to add into our modeling portion of the analysis. After removing the outliers, we ended up with 4,148 observations, which are enough to do a data split of 80%/20% for training and test sets, respectively.

Machine learning algorithms

In this dataset we want to predict a continuous dependent variable from a number of independent variables, so we have used regression analysis to predict the abalone age.
Below are the models executed to analyze the age:

Penalized regression model

Penalized logistic regression imposes a penalty to the logistic model for having too many variables. This results in shrinking the coefficients of the less contributive variables toward zero. This is also known as regularization. We have used the GLMNET package, Glmnet is a package that fits generalized linear and similar models via penalized maximum likelihood. This algorithm is extremely fast and can exploit sparsity in the input matrix x. We have used glmnet with alpha from 0 to 1 to use Ridge, Lasso, and ElasticNet.

Lasso will eliminate many features and reduce overfitting in your linear model. Ridge will reduce the impact of features that are not important in predicting your y values.

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination or the coefficient of multiple determination for multiple regression.

Penalized model is giving an R-squared of 0.5170290. An **R^2** of **0.5** indicates that the model cannot explain 50% of the variability in the outcome data.

PLS model

Partial least squares (PLS) regression is a technique that reduces the predictors to a smaller set of uncorrelated components and performs least squares regression on these components instead of on the original data. PLS regression is especially useful when your predictors are highly collinear, or when you have more predictors than observations and ordinary least-squares regression either produces coefficients with high standard errors or fails completely. PLS does not assume that the predictors are fixed, unlike multiple regression. This means that the predictors can be measured with error, making PLS more robust to measurement uncertainty.

PLS model is giving an R-squared of 0.5170290. An R^2 of 0.5 indicates that 50% of the variability in the outcome data cannot be explained by the model.

ENET model

Elastic net linear regression uses the penalties from both the lasso and ridge techniques to regularize regression models. The technique combines both the lasso and ridge regression methods by learning from their shortcomings to improve on the regularization of statistical models. I have used alpha values between 0 and 1 to optimize the elastic net.

With ENET, R-squared of 0.5191465. With $\boldsymbol{R^2}$ of **0.5** indicates that 50% of the variability in the outcome data cannot be explained by the model.

Generalized Linear Model

A generalized linear model (GLM) is a flexible generalization of ordinary linear regression that allows for the response variable to have an error distribution other than the normal distribution. The GLM generalizes linear regression by allowing the linear model to be related to the response variable via a *link function* and by allowing the magnitude of the variance of each measurement to be a function of its predicted value.

With GLM, calculated R-squared is 0.5197778. With R^2 of 0.5 indicates that 50% of the variability in the outcome data cannot be explained by the model.

K-Nearest Neighbors

This model simply predicts a new sample using the K-closest samples from the training set and its construction is solely based on the individual samples from the training data. To predict a new sample for regression, KNN identifies that sample's KNNs in the predictor space. The predicted response for the new sample is then the mean of the K neighbors' responses. The basic KNN method as described above depends on how the user defines distance between samples. Euclidean distance (i.e., the straight-line distance between two samples) is the most used metric and is defined as follows:

$$\left(\sum_{j=1}^{P} \quad \left(x_{aj} - x_{bj}\right)^2\right)^{1/2} \tag{1}$$

where x_a and xb x_b are two individual samples. Minkowski distance is a generalization of Euclidean distance and is defined as

$$\left(\sum_{j=1}^{P} \left|x_{a_j} - x_{b_j}\right|^q\right)^{1/q} \quad (2)$$

Where q > 0. It is easy to see that when q = 2, then Minkowski distance is the same as Euclidean distance. When q = 1, then Minkowski distance is equivalent to Manhattan distance.

MARS (multivariate adaptive regression splines)

MARS is a type of ensemble of simple linear functions and can achieve good performance on challenging regression problems with many input variables and complex non-linear relationships. This algorithm automatically creates a piecewise linear model which provides an intuitive stepping block into nonlinearity after grasping the concept of multiple linear regression.

MARS model uses the **earth** package, by default, earth::earth() will assess all potential knots across all supplied features and then will prune to the optimal number of knots based on an expected change in R^2(for the training data) of less than 0.001. This calculation is performed by the Generalized cross-validation (GCV) procedure, which is a computational shortcut for linear models that produces an approximate leave-one-out cross-validation error metric.

SVM

Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. To estimate the model parameters, SVM uses the loss function.

SVMs for regression use a function like the Huber function, with an important difference. Given a threshold set by the user (denoted as $\in$) data points with residuals within the threshold

do not contribute to the regression fit while data points with an absolute difference greater than the threshold contribute a linear-scale amount.

The SVM regression coefficients minimize.

$$Cost \sum_{i=1}^{n} \quad L_{\epsilon}(y_i - \hat{y}_i) + \sum_{j=1}^{P} \quad \beta_j^2 \tag{3}$$

where L_{ϵ} $(\cdot)$ is the ∈-insensitive function. The Cost parameter is the cost penalty that is set by the user, which penalizes large residuals.

Conclusion

Table 2.

Model Comparison for Train and Test Sets

Model Comparison For Train

	rmse	r2
GLMNET	2.212	0.5265
PLS	2.209	0.5279
ENET	2.208	0.5282
GLM	2.207	0.5285
MARS	2.123	0.5639
KNN	2.112	0.5775
SVM	2.052	0.607

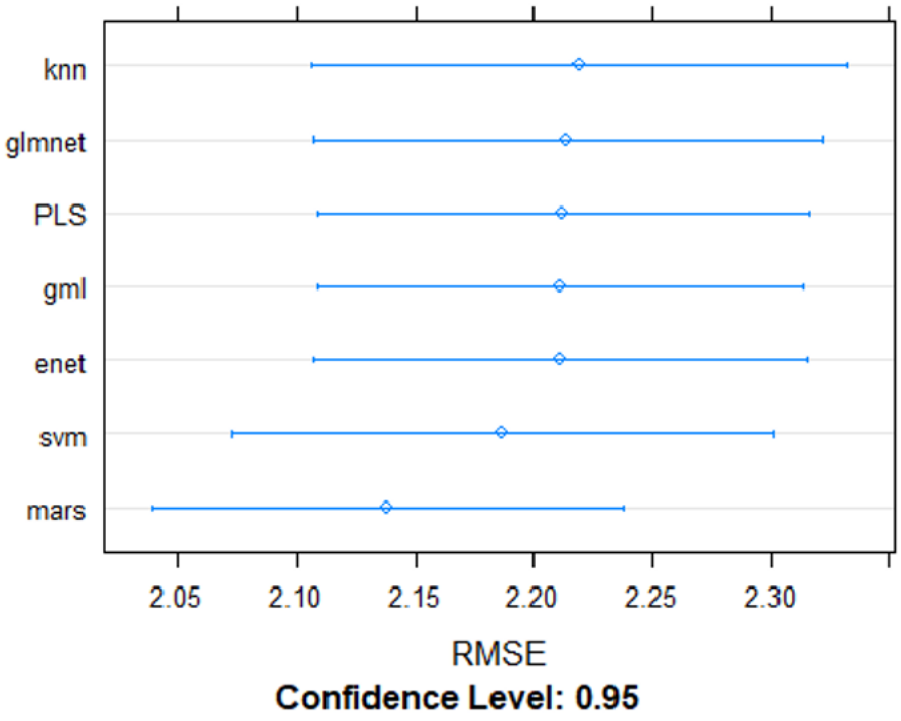
Model Comparison For Test

	rmse	r2
GLMNET	2.19	0.517
PLS	2.191	0.5177
ENET	2.187	0.5191
GLM	2.186	0.5198
MARS	2.133	0.5442
KNN	2.158	0.5336
SVM	2.121	0.5566

Note. SVM train and test sets have a higher R2 than the rest of the other models

Figure 4.

RMSE Comparison Within Models



Note. The RMSE range for all the models are between 2 and 2.3

We ran all 7 models on same train dataset, and we can see RMSE, and R squared are close for each model however SVM has highest R2 of 0.62 and least RMSE which is approx. 2, ideally, we would want this value to be closer to 0 but it's still quite low which is a good sign among all so we can propose SVM model by ignoring the categorical variable sex. Furthermore, we validated all 7 models on the Test dataset and the results were the same. We evaluated the model by quantifying the R-squared, RMSE, and the variance. The R-squared was average with a value of 0.62 but the mean squared error and variance were relatively low for SVM.

To conclude, the economic value of abalone is positively correlated with its age. Estimating the age of abalone accurately is important for both farmers and customers to determine its price. Based on this analysis, it seems the proposed SVM regression models worked well to predict the abalone age with

less error. In other words, we do not need any laboratory experiment to predict the age of abalones. We can predict the age of abalone using the quite simple physical characteristics like weight, height, diameter, and length which we did in the project.

We can further work on model Tuning and preprocessing to get better RMSE which should be close to zero and R-squared should be near 1 to get a more accurate prediction with less error.

References

Hossain, M. & Chowdhury, N. M. (2019, January 3). Econometric Ways to Estimate the Age and Price of Abalone. *Munich Personal RePEc Archive*. https://mpra.ub.uni-muenchen.de/91210/

Waugh, S. (2019, January 3): Abalone Data Set. *University of California Irvine Machine Learning Repository*. https://archive.ics.uci.edu/ml/datasets/abalone

Appendix

Traditionally, the process to predict the age of the abalone is by cutting the shell through the cone, staining it, and counting the number of rings through a microscope. The abalone dataset includes numeric attributes with a different types of measurements with the goal to predict the age of an abalone more efficiently.

```
abalone <- read.csv('abalone.csv', header = TRUE)
abalone$Age <- abalone$Rings+1.5
abalone <- subset(abalone, select = -c(Rings))
```

The table has 4,177 observations and nine columns. The variable Age replaced the attribute Rings since each ring is equivalent to the number of rings plus 1.5 (Hossain). This will be helpful at a later stage of the project when the dataset is split into training and test set. As a reminder, this project aims to calculate the Age of the abalone so that the dependent variable will be Age.

```
summary(abalone)
```

```
##      Sex                Length         Diameter          Height
##  Length:4177        Min.   :0.075   Min.   :0.0550   Min.   :0.0000
##  Class :character   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150
##  Mode  :character   Median :0.545   Median :0.4250   Median :0.1400
##                     Mean   :0.524   Mean   :0.4079   Mean   :0.1395
##                     3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650
##                     Max.   :0.815   Max.   :0.6500   Max.   :1.1300
##   Whole.weight    Shucked.weight   Viscera.weight    Shell.weight
##  Min.   :0.0020   Min.   :0.0010   Min.   :0.0005   Min.   :0.0015
##  1st Qu.:0.4415   1st Qu.:0.1860   1st Qu.:0.0935   1st Qu.:0.1300
##  Median :0.7995   Median :0.3360   Median :0.1710   Median :0.2340
##  Mean   :0.8287   Mean   :0.3594   Mean   :0.1806   Mean   :0.2388
##  3rd Qu.:1.1530   3rd Qu.:0.5020   3rd Qu.:0.2530   3rd Qu.:0.3290
##  Max.   :2.8255   Max.   :1.4880   Max.   :0.7600   Max.   :1.0050
##       Age
##  Min.   : 2.50
##  1st Qu.: 9.50
##  Median :10.50
##  Mean   :11.43
##  3rd Qu.:12.50
##  Max.   :30.50
```

Furthermore, the attributes in the abalone dataset are numeric in its majority, and Sex is the only categorical data (binary). The dataset is not missing any values, so no data cleaning is needed.

```
str(abalone)
```

```
## 'data.frame':    4177 obs. of  9 variables:
##  $ Sex           : chr  "M" "M" "F" "M" ...
```

```
##  $ Length        : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 ...
##  $ Diameter      : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44 ...
##  $ Height        : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15 ...
##  $ Whole.weight  : num  0.514 0.226 0.677 0.516 0.205 ...
##  $ Shucked.weight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
##  $ Viscera.weight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
##  $ Shell.weight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ...
##  $ Age           : num  16.5 8.5 10.5 11.5 8.5 9.5 21.5 17.5 10.5 20.5 ...
```

```
sapply(abalone, function(x) sum(is.na(abalone)))
```

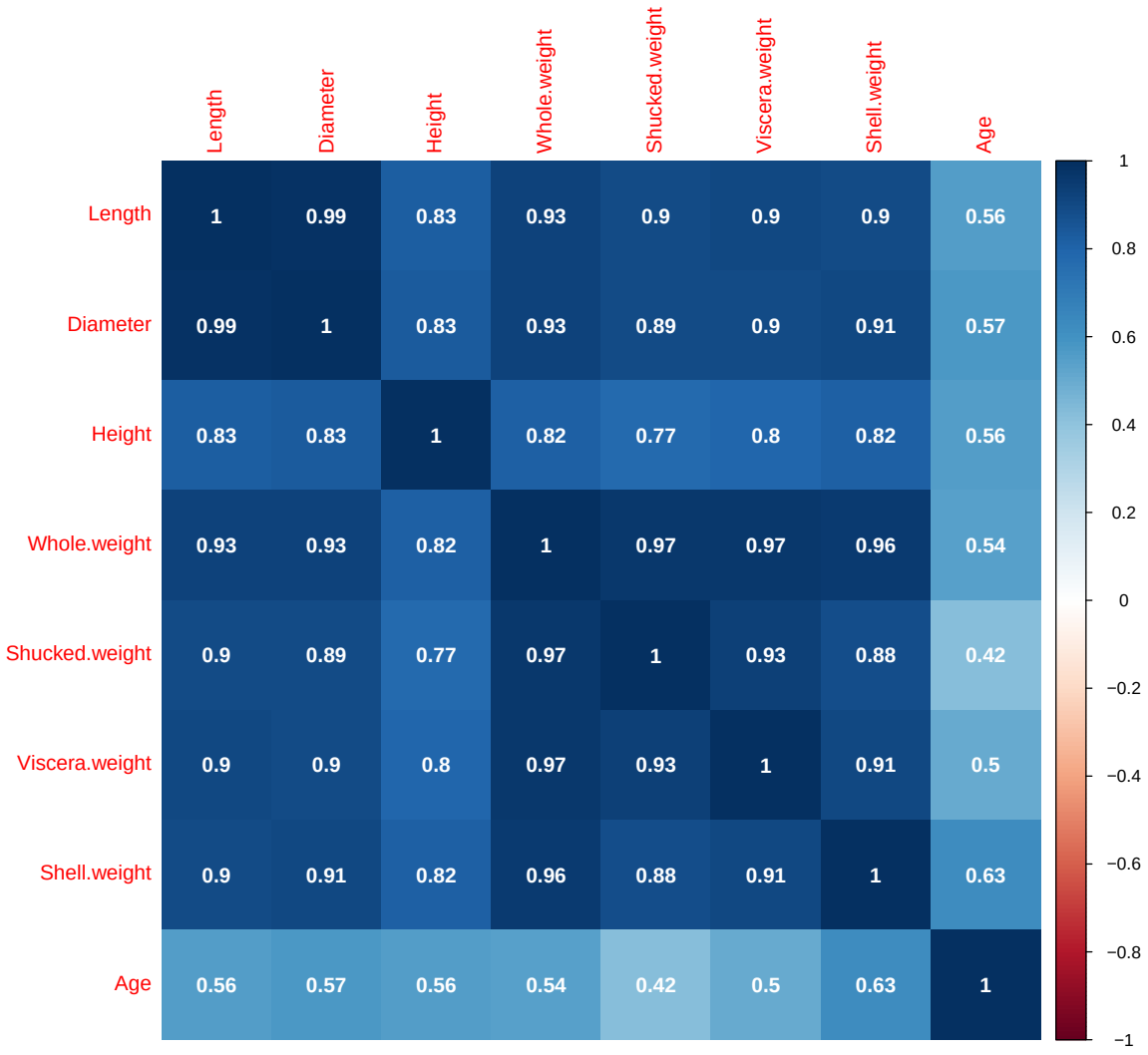
```
##            Sex         Length       Diameter         Height   Whole.weight
##              0              0              0              0              0
## Shucked.weight Viscera.weight   Shell.weight            Age
##              0              0              0              0
```

The correlation plot shows high correlations (between 0.75 and 0.99) within the independent variables, but a medium direct correlation with the target value (between 0.4 and .65)

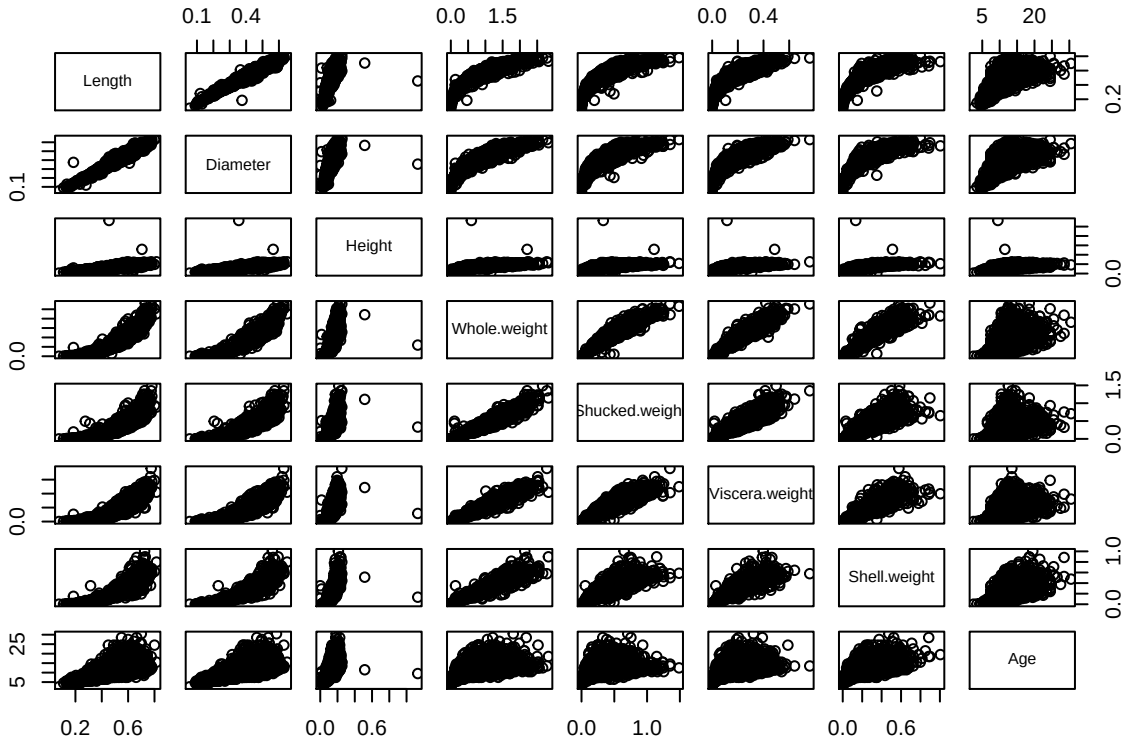
```
library(corrplot)
```

```
## corrplot 0.88 loaded
```

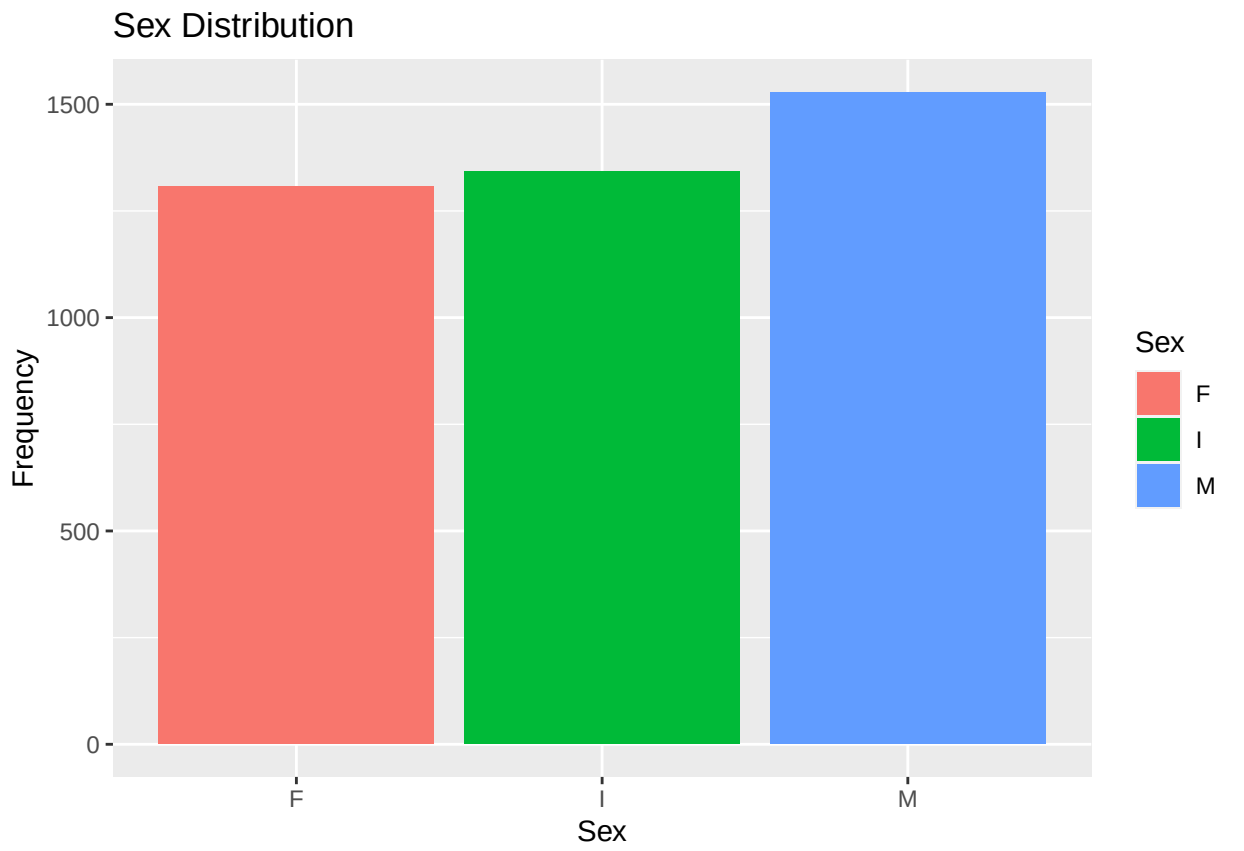
```
corrplot(cor(abalone[c(2:9)]), method = "shade", addCoef.col = "white")
```




```
pairs(abalone[2:9])
```




```
library(ggplot2)
ggplot(data=abalone,aes(x=Sex,fill=Sex))+
  geom_bar() +
  ggtitle("Sex Distribution") +
  ylab("Frequency") +
  xlab("Sex")
```



The sex variable , our only categorical variable was plotted by bar plot. the values are M for male, F for female and I for Infants.based on the analysis and visualization, we can conclude that in relation to this variable, the dataset is balanced.

```
library(Hmisc)
```

```
## Loading required package: lattice
```

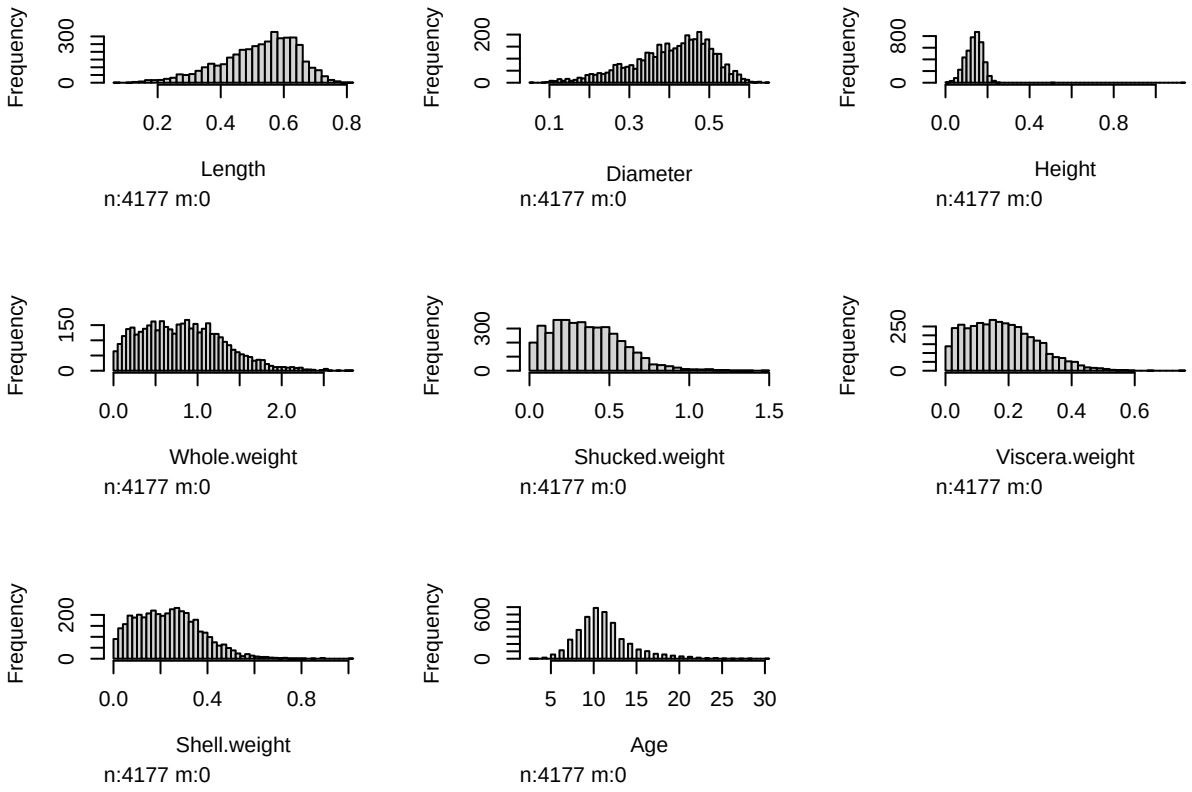
```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

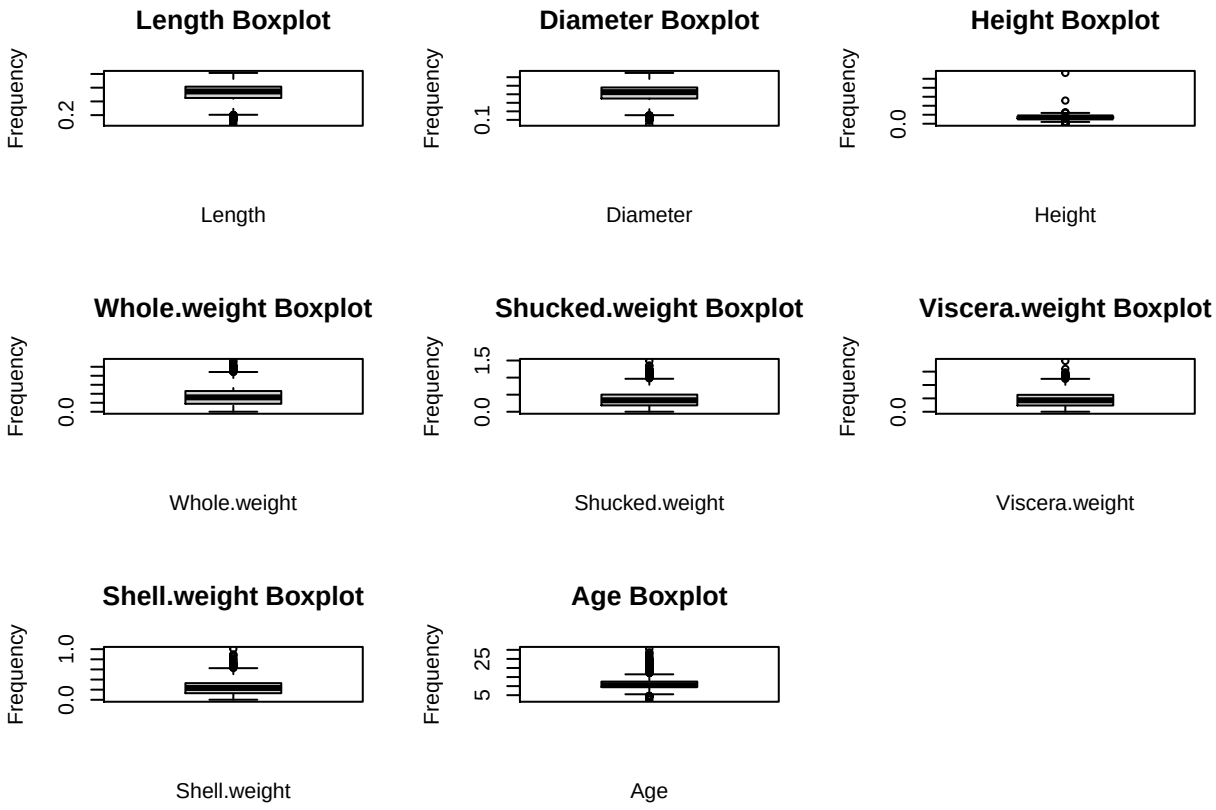
```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
hist.data.frame(abalone[2:9])
```



A histogram was plotted for showing the distributions of continuous variables. we notice an approximate normal distributions among all of variables.However, there are some high peaks in "Height" histogram. The peaks worth the more deep explorations in depth to see if there are any outlines.

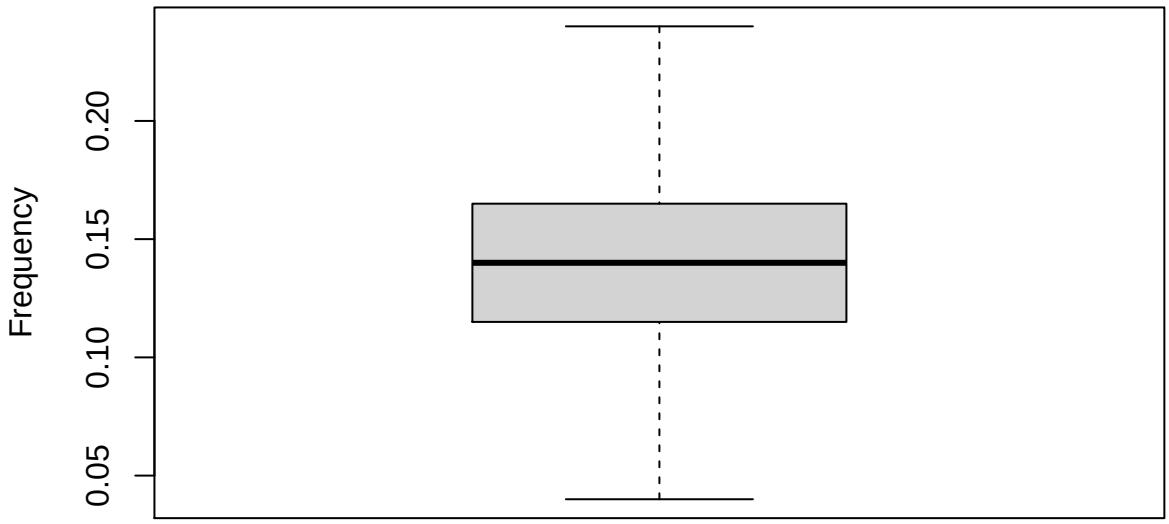
```
par(mfrow=c(3,3))
for(i in 2:9) {
    boxplot(abalone[,i],
            main= paste(names(abalone)[i], "Boxplot"),
            xlab = names(abalone)[i],
            ylab = "Frequency")
}
```



By looking at the "Height" boxplot we can see that the high peak we observed is due to the presence of few(2) outliers that are laying beyond the central position on the distributions. Therefore, we can conclude that those are extreme outlines and should be removed prior to modeling for a better Performance.

```
outliers <- boxplot(abalone$Height, plot=FALSE)$out
abalone <- abalone[-which(abalone$Height %in% outliers),]

boxplot(abalone$Height, main="Height",
        xlab = "Height",
          ylab = "Frequency")
```




```
dim(abalone)
```

```
## [1] 4148    9
```

```
library(caret)
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:survival':
##
##     cluster
```

```
set.seed(100)
abalone<- abalone[,-c(1)]
partition<-createDataPartition(y=abalone$Age,p=0.80,list = FALSE)


abalone_train<-abalone[partition, ]
abalone_test<-abalone[-partition, ]
control <- trainControl(method="repeatedcv", number=10)
```

```r
set.seed(100)
glm_fit<- train(Age~., data=abalone_train, method="glm", metric="RMSE",trControl=control)
glm_pred_tr <- predict(glm_fit, newdata=abalone_train[,c(1:7)])
glm_PR_tr <- postResample(pred=glm_pred_tr, obs=abalone_train$Age)
rmses_training = c(glm_PR_tr[1])
r2s_training = c(glm_PR_tr[2])
methods = c("GLM")

glm_pred <- predict(glm_fit, newdata=abalone_test[,c(1:7)])
glm_PR <- postResample(pred=glm_pred, obs=abalone_test$Age)
rmses_testing = c(glm_PR[1])
r2s_testing = c(glm_PR[2])
r2s_testing
```

```
##  Rsquared
## 0.5197778
```

```r
### Penalized Models
glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),
 .lambda = seq(.01, .2, length = 40))
set.seed(100)
glmnet_fit <- train(Age~., data=abalone_train, method = "glmnet", tuneGrid = glmnGrid, preProc = c("cen
glmnet_pred_tr <- predict(glmnet_fit, newdata=abalone_train[,c(1:7)])
glmnet_PR_tr <- postResample(pred=glmnet_pred_tr, obs=abalone_train$Age)
rmses_training = c(rmses_training, glmnet_PR_tr[1])
r2s_training = c(r2s_training, glmnet_PR_tr[2])
methods = c(methods,"GLMNET")

glmnet_pred <- predict(glmnet_fit, newdata=abalone_test[,c(1:7)])
glmnet_PR <- postResample(pred=glmnet_pred, obs=abalone_test$Age)
rmses_testing = c(rmses_testing, glmnet_PR[1])
r2s_testing = c(r2s_testing, glmnet_PR[2])
r2s_testing
```

```
##  Rsquared  Rsquared
## 0.5197778 0.5170290
```

In r2s_testing variable, we have stored Rsquared for GLM and GLMNET and displayed. Both model shows similar Rsquared of 50%.

```r
### Enet
enetGrid <- expand.grid(.lambda = c(0, .001, .01, .1),
.fraction = seq(0.05, 1, length = 20))
set.seed(100)
enet_fit <- train(Age~., data=abalone_train, method = "enet", preProc = c("center", "scale"), tuneGrid =
enet_pred_tr <- predict(enet_fit, newdata = abalone_train[,c(1:7)])
enet_PR_tr <- postResample(pred = enet_pred_tr, obs=abalone_train$Age)
rmses_training = c(rmses_training, enet_PR_tr[1])
r2s_training = c(r2s_training, enet_PR_tr[2])
methods = c(methods,"ENET")

enet_pred <- predict(enet_fit, newdata=abalone_test[,c(1:7)])
```

```
enet_PR <- postResample(pred=enet_pred, obs=abalone_test$Age)
rmses_testing = c(rmses_testing, enet_PR[1])
r2s_testing = c(r2s_testing, enet_PR[2])
r2s_testing
```

```
##  Rsquared  Rsquared  Rsquared
## 0.5197778 0.5170290 0.5191465
```

Similar to GLM and GLMNET, ENET also has Rsquared close to 50% (0.5014820).

```
### PLS

set.seed(100)
pls_fit <- train(Age~., data=abalone_train, method = "pls", preProc = c("center", "scale"), tuneLength =
pls_pred_tr <- predict(pls_fit, newdata=abalone_train[,c(1:7)])
pls_PR_tr <- postResample(pred=pls_pred_tr, obs=abalone_train$Age)
rmses_training = c(rmses_training, pls_PR_tr[1])
r2s_training = c(r2s_training, pls_PR_tr[2])
methods = c(methods, "PLS")


pls_pred <- predict(pls_fit, newdata=abalone_test[,c(1:7)])
pls_PR <- postResample(pred=pls_pred, obs=abalone_test$Age)
rmses_testing = c(rmses_testing, pls_PR[1])
r2s_testing = c(r2s_testing, pls_PR[2])
r2s_testing
```

```
##  Rsquared  Rsquared  Rsquared  Rsquared
## 0.5197778 0.5170290 0.5191465 0.5177052
```

In PLS (last column), we are getting rsquared of 50%. Till now, allmodels are showing rsquared of 50%.

```
library(earth)
```

```
## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos

##
## Attaching package: 'TeachingDemos'

## The following object is masked _by_ '.GlobalEnv':
##
##     outliers

## The following objects are masked from 'package:Hmisc':
##
##     cnvrt.coords, subplot
```

```r
library(e1071)
```

```
##
## Attaching package: 'e1071'

## The following object is masked from 'package:Hmisc':
##
##     impute
```

```r
library(MASS)
```

```r
### MARS
set.seed(100)
earth_fit <- train(Age~., data=abalone_train, method = "earth", tuneGrid = expand.grid(.degree = 1,.npru
earth_pred_tr <- predict(earth_fit, newdata = abalone_train[,c(1:7)])
earth_PR_tr <- postResample(pred = earth_pred_tr, obs=abalone_train$Age)
rmses_training = c(rmses_training, earth_PR_tr[1])
r2s_training = c(r2s_training, earth_PR_tr[2])
methods = c(methods,"MARS")

earth_pred <- predict(earth_fit, newdata=abalone_test[,c(1:7)])
earth_PR <- postResample(pred=earth_pred, obs=abalone_test$Age)
rmses_testing = c(rmses_testing, earth_PR[1])
r2s_testing = c(r2s_testing, earth_PR[2])
r2s_testing
```

```
##  Rsquared  Rsquared  Rsquared  Rsquared  Rsquared
## 0.5197778 0.5170290 0.5191465 0.5177052 0.5441757
```

MARS has shown some improvement in Squared which is 57% and all the models are balanced and giving Rsquared close to 50% in average.

```r
### Svm
set.seed(100)
svm_fit <- train(Age~., data=abalone_train, method = "svmRadial", preProc = c("center", "scale"), metri
svm_pred_tr <- predict(svm_fit, newdata = abalone_train[,c(1:7)])
svm_PR_tr <- postResample(pred = svm_pred_tr, obs=abalone_train$Age)
rmses_training = c(rmses_training, svm_PR_tr[1])
r2s_training = c(r2s_training, svm_PR_tr[2])
methods = c(methods,"SVM")

svm_pred <- predict(svm_fit, newdata=abalone_test[,c(1:7)])
svm_PR <- postResample(pred=svm_pred, obs=abalone_test$Age)
rmses_testing = c(rmses_testing, svm_PR[1])
r2s_testing = c(r2s_testing, svm_PR[2])
r2s_testing
```

```
##  Rsquared  Rsquared  Rsquared  Rsquared  Rsquared  Rsquared
## 0.5197778 0.5170290 0.5191465 0.5177052 0.5441757 0.5565511
```

SVM model has improved further compared to MARS with Rsquared is close to 60%. Till now SVM showed best Rsquared among all models.

```
### Knn Model
set.seed(100)
knn_fit <- train(Age~., data=abalone_train, method="knn",
 preProc=c("center","scale"), tuneLength=10, metric = "RMSE", trControl = control)
knn_pred_tr <- predict(knn_fit, newdata = abalone_train[,c(1:7)])
knn_PR_tr <- postResample(pred = knn_pred_tr, obs=abalone_train$Age)
rmses_training = c(rmses_training, knn_PR_tr[1])
r2s_training = c(r2s_training, knn_PR_tr[2])
methods = c(methods,"KNN")

knn_pred <- predict(knn_fit, newdata=abalone_test[,c(1:7)])
knn_PR <- postResample(pred=knn_pred, obs=abalone_test$Age)
rmses_testing = c(rmses_testing, knn_PR[1])
r2s_testing = c(r2s_testing, knn_PR[2])
r2s_testing
```

```
##  Rsquared  Rsquared  Rsquared  Rsquared  Rsquared  Rsquared  Rsquared
## 0.5197778 0.5170290 0.5191465 0.5177052 0.5441757 0.5565511 0.5336223
```

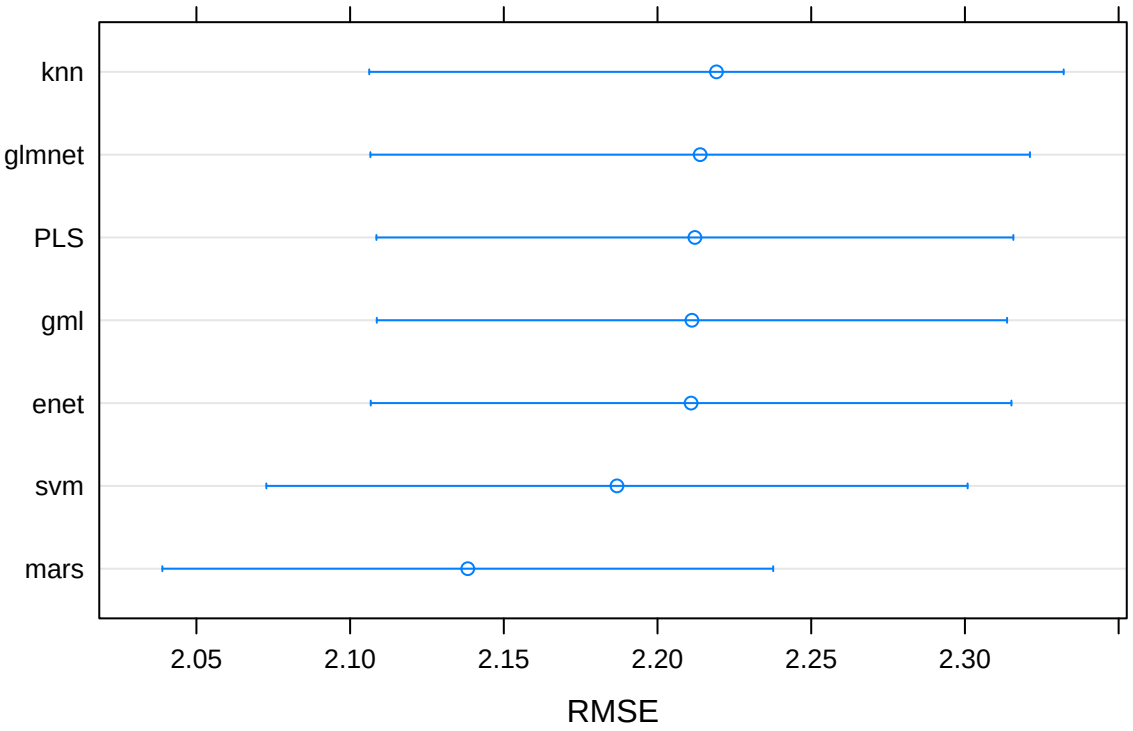
Rsquared is 56% by KNN which is lower compared to SVN. Till now SVN is highest Rsquared of 60%.

```
resamp = resamples(
list(knn=knn_fit,svm=svm_fit,mars=earth_fit, PLS=pls_fit, enet=enet_fit, glmnet=glmnet_fit, gml=glm_fit)
print(summary(resamp))
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: knn, svm, mars, PLS, enet, glmnet, gml
## Number of resamples: 10
##
## MAE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## knn    1.388554 1.515038 1.552952 1.551778 1.601738 1.700319    0
## svm    1.339740 1.496211 1.530459 1.505252 1.554745 1.589883    0
## mars   1.377142 1.531434 1.571232 1.543243 1.590525 1.630194    0
## PLS    1.430143 1.577063 1.630613 1.611680 1.667453 1.754958    0
## enet   1.433858 1.575643 1.631854 1.610898 1.662740 1.762504    0
## glmnet 1.435417 1.570390 1.634042 1.611661 1.665032 1.766528    0
## gml    1.434810 1.580490 1.630002 1.611484 1.662097 1.760934    0
##
## RMSE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## knn    1.993940 2.135946 2.196262 2.219172 2.347752 2.448545    0
## svm    1.931026 2.118870 2.201158 2.186813 2.294155 2.400492    0
## mars   1.920568 2.067113 2.160908 2.138277 2.220680 2.329220    0
## PLS    1.983237 2.105628 2.231851 2.212158 2.315208 2.436385    0
## enet   1.986061 2.094814 2.232823 2.210924 2.312758 2.441523    0
## glmnet 1.989667 2.082276 2.242984 2.213874 2.306962 2.453123    0
```

```
## gml    1.986216 2.105279 2.227975 2.211198 2.319917 2.436376    0
##
## Rsquared
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## knn    0.4635530 0.5167538 0.5290945 0.5292664 0.5481951 0.5950605    0
## svm    0.4927218 0.5181545 0.5499134 0.5488369 0.5682334 0.6238488    0
## mars   0.4962963 0.5346555 0.5611284 0.5593625 0.5788372 0.6199733    0
## PLS    0.4855215 0.5057556 0.5149187 0.5280242 0.5404604 0.5955083    0
## enet   0.4880300 0.5086188 0.5172918 0.5284836 0.5422892 0.5945304    0
## glmnet 0.4821905 0.5069806 0.5203395 0.5270615 0.5442775 0.5933901    0
## gml    0.4915703 0.5084923 0.5142211 0.5285740 0.5398433 0.5943583    0
```

```
dotplot(resamp, metric="RMSE")
```



We have plotted RMSE on dotplot with confidence level of 0.95. MARS has low RMSE compared to all models. SVM has the second lowest RMSE and it has highest Rsquared of 60%. Worst model is KNN by dotplot which has highest RMSE.

```
res_training = data.frame(rmse=rmses_training, r2=r2s_training)
rownames(res_training) = methods
training_order = order(-res_training$rmse)

res_training = res_training[ training_order, ]
res_testing = data.frame( rmse=rmses_testing, r2=r2s_testing )
rownames(res_testing) = methods
```

```
res_testing = res_testing[ training_order, ]

library(pander)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:Hmisc':
##
##     src, summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
res_training %>% pander(style = "simple", split.table = Inf, justify = "left",
caption="Model Comparison For Train ")
```

Table 1: Model Comparison For Train

	rmse	r2
GLMNET	2.212	0.5265
PLS	2.209	0.5279
ENET	2.208	0.5282
GLM	2.207	0.5285
MARS	2.123	0.5639
KNN	2.112	0.5775
SVM	2.052	0.607

```
res_testing %>% pander(style = "simple", split.table = Inf, justify = "left",
caption="Model Comparison For Test")
```

Table 2: Model Comparison For Test We have used 7 models which are GLMNET, PLS, ENET, GLM, KNN, MARS & SVM. here we are showing model comparison on Train and Test data using RMSE and R2 in a tabular format in ascending order. SVM has come out as the best model with lowest RMSE of 2 in Train and close to 2 on Test data.Similarly, R2 is highest for SVM on Train and Test dataset.

	rmse	r2
GLMNET	2.19	0.517
PLS	2.191	0.5177
ENET	2.187	0.5191
GLM	2.186	0.5198
MARS	2.133	0.5442
KNN	2.158	0.5336
SVM	2.121	0.5566