Deciphering Abalone Ages through Machine Learning Methods

Anonymous Marking Code: Z0195806

February 12, 2024

Contents

1 Introduction

1.1 Background

Traditional methods of determining the age of abalones involve cutting open the abalone shells at the cone, staining them, and then counting the rings under a microscope. This process is time-consuming and destructive, relies on manual effort, and is subject to the subjective judgement of technicians, potentially causing harm to the abalones. These factors limit the efficiency and accuracy of the method, prompting researchers to seek better solutions. Alternative, more readily obtainable measurements may be used to predict age, but might require additional information such as weather patterns, location, and food supply.

In this report, I leverage the UCI Abalone dataset [1], which contains physical measurements of abalones (such as length, height, and total weight), and use R's tidyverse to build a machine learning model. This approach offers a more sophisticated method for predicting the age of abalones.

1.2 Data Set Introduction

The table (Table 1) is some description of the data set, in which Sex, Length, Diameter, Height, Whole_weight, Shucked_weight, Viscera_weight and Shell_weight are the predictor

Variable Name	Type	Description	Units
Sex	Categorical	M, F, and I (infant)	-
Length	Continuous	Longest shell measurement	mm
Diameter	Continuous	Perpendicular to length	mm
Height	Continuous	With meat in shell	mm
Whole_weight	Continuous	Whole abalone	grams
Shucked_weight	Continuous	Weight of meat	grams
Viscera_weight	Continuous	Gut weight (after bleeding)	grams
Shell_weight	Continuous	After being dried	grams
Rings	Integer	+1.5 gives the age in years	-

Table 1: Abalone Dataset Variable Description

variables. The target variable Age does not appear in the data set, but its value is equal to Rings+1.5.

1.3 Model in Use

Figure 1: Pic of Abalone

In the report, two types of machine learning algorithms, linear regression with Lp regularization and random forest, were used to train the processed training set. The performance of the two models was then compared based on their performance on the test set.

2 Data Pre-processing and EDA

Details on the datasets used in Parts 2, 3, and 4 can be found in the Appendix.

2.1 Data Pre-processing

2.1.1 Data Cleaning

Since all columns in the dataset, except for the Sex column, contain continuous data, there were no typographical errors found upon inspection. Also, using **any(is.na(dataset))** revealed no missing values in the data.

Before executing machine learning tasks, it's crucial to inspect the dataset, as ambiguous data in the training set can lead to biases in model training. At the same time, I identified some logical inconsistencies in the data and used the filter from tidyverse to remove these rows from the dataset:

1. There were 2 instances where the Height was equal to 0, which is impossible for abalones;
2. Theoretically, the total weight of an abalone should equal the sum of its parts, but there were a significant number of rows (15 rows) where:

$$WholeWeight < ShuckedWeight + ShellWeight$$

2.1.2 Add Age Column

Although using "Rings" might be slightly simpler and keep the prediction task closer to the original data structure, using "Age" can make the prediction more interpretable from a biological perspective, as it directly represents the age of the abalones. Hence, I added a new column named Age using the formula Age = Rings + 1.5 and removed the original Rings column.

2.1.3 Dealing with Sex Columns

Machine learning algorithms typically require numerical input, and the "Sex" column in the Abalone dataset contains categorical values "F" (female), "M" (male), and "I" (infant), which need to be encoded before they can be effectively used in most machine learning models.

One-hot encoding converts each category value into a new binary column and assigns a 1 or 0 (True/False) value to those columns. For the 'Sex' column with three categories, it would create three new columns, one for each category ('F', 'M', 'I'), with binary values indicating the presence of each category. Here, I added three new columns named SexM, SexF, SexI, and deleted the Sex column.

2.2 Exploratory Data Analysis

2.2.1 Descriptive Statistics for Single Variables

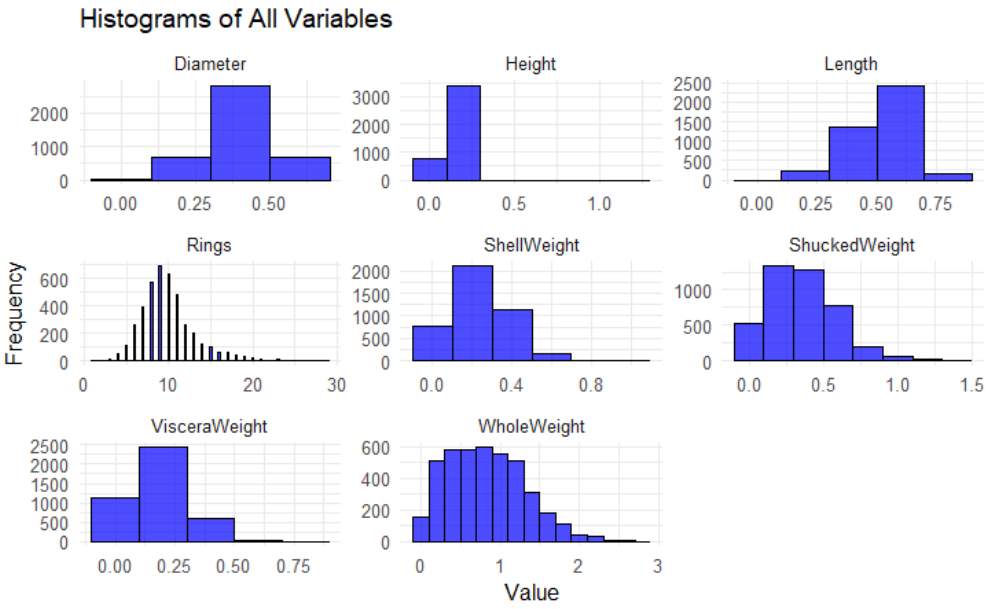


Figure 2: Histograms of Variables

```
     Sex                Length          Diameter         Height        WholeWeight
 Length:4177        Min.   :0.075   Min.   :0.0550   Min.   :0.0000   Min.   :0.0020
 Class :character   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150   1st Qu.:0.4415
 Mode  :character   Median :0.545   Median :0.4250   Median :0.1400   Median :0.7995
                    Mean   :0.524   Mean   :0.4079   Mean   :0.1395   Mean   :0.8287
                    3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650   3rd Qu.:1.1530
                    Max.   :0.815   Max.   :0.6500   Max.   :1.1300   Max.   :2.8255
 Shuckedweight    Visceraweight     Shellweight         Rings
 Min.   :0.0010   Min.   :0.0005   Min.   :0.0015   Min.   : 1.000
 1st Qu.:0.1860   1st Qu.:0.0935   1st Qu.:0.1300   1st Qu.: 8.000
 Median :0.3360   Median :0.1710   Median :0.2340   Median : 9.000
 Mean   :0.3594   Mean   :0.1806   Mean   :0.2388   Mean   : 9.934
 3rd Qu.:0.5020   3rd Qu.:0.2530   3rd Qu.:0.3290   3rd Qu.:11.000
 Max.   :1.4880   Max.   :0.7600   Max.   :1.0050   Max.   :29.000
```

Figure 3: Descriptive Statistical Data

Descriptive statistics for a single variable involve summarizing and analyzing data to describe the main characteristics of the abalone dataset without making any assumptions about the data generation process. In this section, using the original dataset (abalone_origin), measures of central tendency (such as mean and median), measures of variability (such as range, variance, and standard deviation, Figure 2), and graphical representations (such as histograms, Figure 3) were analyzed.

2.2.2 Analysis of Correlations Between Variables

Bivariate analysis can clearly show how each feature is influenced by the presence of other features. It also helps us understand and identify significant features, overcome multicollinearity effects and interdependencies, thus providing insights into hidden noise patterns in the data.

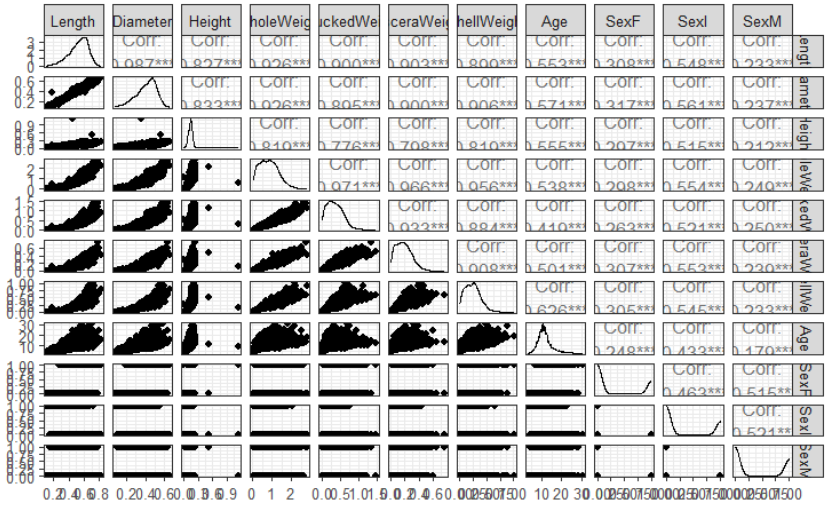


Figure 4: Correlation Plot between Variables

In this section, using the abalone_Age_DummySex dataset, we were particularly interested in the impact of gender factors on age prediction. The correlation analysis charts showed that the relationships between the three genders and other variables are two parallel lines, suggesting that gender has no correlation with other predictors or age. This can be a reference for subsequent modeling. However, correlation analysis is just a basic method and not sufficient to support the removal of the Sex variable (Figure 4).

2.2.3 Best Subset Selection

With multiple features in the abalone dataset, directly using all predictors as model inputs is not a wise choice; it's necessary to filter input variables. Since the number of variables in the abalone dataset is not large, we opted for best subset selection here. In statistics and machine learning, best subset selection is a method used to select the best subset of predictors from a set of predictors to build a model.

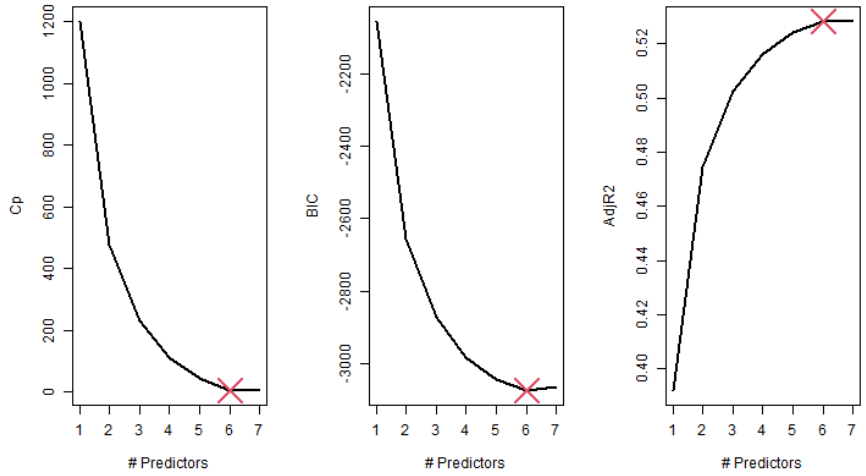


Figure 5: Best Subset Selection

I applied the regsubsets function from the leaps package to both the abalone_Age_DummySex and abalone_Age_NoSex datasets to find the best subset for each. The following conclusions were drawn: 1. On the abalone_Age_DummySex dataset with Dummy variables, the function returned "1 linear dependencies found Warning," indicating that the dataset contains linearly dependent variables, which is not favorable for regression prediction. 2. On the dataset without Sex, the best subset contained 6 variables (7 in total, Figure 5) and did not include Height, showing consistency in the Adj Rsq, BIC, Cp charts. These conclusions provide statistical support for later modeling but do not result in the deletion of any variables.

3 Modeling

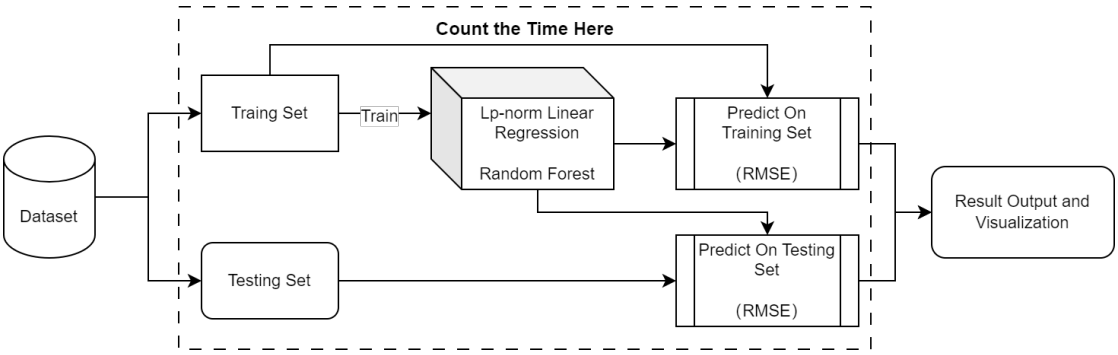


Figure 6: Machine Learning Pipeline

In the machine learning modeling section of the report, Lp Norm linear regression and random forest algorithms were employed. The core methodology of this section is divided into two steps: 1. Splitting the dataset into a training set and a testing set (approximately at an 8:2 ratio), using a fixed random seed to ensure consistent results across multiple trainings and tests; 2. Encapsulating the process of training the machine learning model on the training set, making predictions on the test set, calculating model error using RMSE, and visualization into a pipeline (by calling perform_cv_glmnet and perform_random_forest) to swiftly execute the machine learning workflow on the dataset.

Dataset configurations: abalone_Age, abalone_Age_Dummysex, abalone_age_NoSex, SexM, SexF, SexI, and their corresponding training and testing sets split for each dataset. These datasets meet a variety of analytical needs.

3.1 Lp Norm Linear Regression (Ridge, LASSO, and Elastic Net)

3.1.1 Reason for choosing this instead of Decision Tree

Linear regression models, including those with regularization, are generally more interpretable than decision tree models. In Lasso, Ridge, and Elastic Net regressions, the weight of each feature can be directly interpreted as its impact on the target variable, which is crucial for statistical analysis and result interpretation. Lasso regression (L1 regularization) is particularly suited for feature selection as it can shrink the coefficients of unimportant features to zero. If there are many unimportant features in the abalone dataset, Lasso regression can automatically perform feature selection and simplify the model. Ridge regression (L2 regularization) is particularly useful if the features in the abalone dataset are highly correlated

(multicollinearity). Ridge regression reduces the variance of parameter estimates by adding a regularization term, thereby improving the model's stability and generalization ability.

3.1.2 Introduction to Lp Norm Linear Regression

Lp linear regression is a set of methods that optimize the model by introducing regularization terms to the traditional linear regression. These regularization techniques aim to address issues like overfitting, multicollinearity, and, in some cases, perform feature selection.

3.1.3 Introduction to Lp Norm Linear Regression

Lp linear regression is a set of methods that optimize the model by introducing regularization terms to the traditional linear regression. These regularization techniques aim to address issues like overfitting, multicollinearity, and, in some cases, perform feature selection.

3.1.4 Model Assumption

1. Linearity assumption: The model assumes a linear relationship between the predictor variables (such as the abalone's length, diameter, height, weight, etc.) and the response variable (such as age or rings). This means the response variable can be expressed as a weighted sum of the predictor variables plus an error term.

2. Multicollinearity assumption: While traditional linear regression models assume no perfect multicollinearity among predictor variables, Lp linear regression can tolerate a certain degree of multicollinearity and mitigate its adverse effects through regularization. If the predictor variables in the abalone dataset are highly correlated (like whole weight and shucked weight), using Ridge or Elastic Net might be more appropriate as these methods can handle such multicollinearity.

3. Proper choice of alpha and lambda: The model assumes that the alpha and lambda values chosen through cross-validation produce a well-regularized model that is neither overfit nor underfit.

4. Proper representation of the feature space: The model assumes that the input features have been properly transformed or selected to allow the linear model to capture the relationships between the target variable and the features.

3.1.5 Design of Experiment (DoE)

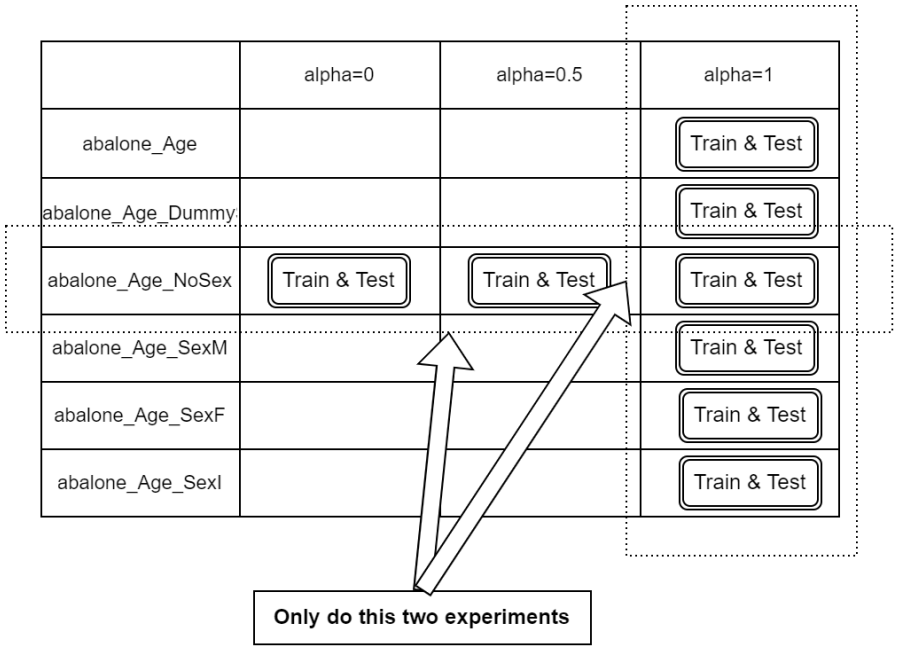


Figure 7: Design of Experiment

As mentioned at the beginning of part 3, five datasets were set up to meet various analytical needs, and I set three different alpha values (0, 0.5, 1), corresponding to Ridge, Elastic Net, and Lasso regression, theoretically requiring 15 machine learning analyses to analyze results. Due to time constraints and the energy required to analyze 15 outcomes, I designed the following experimental methods for analysis: The first part uses the same dataset with different alpha values to verify model assumption three; the second part uses different datasets with the same alpha value to verify model assumption four. This allows conclusions to be drawn from 8 experiments. The experimental design is as follows:

3.1.6 Result Analysis

Since model assumptions 1 and 2 have been proven correct in part 2, I will focus on analyzing based on assumptions 3 and 4. According to the experimental results, it was found:

1. On the same dataset `Abalone_Age_NoSex`, by changing the value of α, the model's final error RMSE on both the training and testing sets was between 2.1 and 2.2, changing as α changed from 0 to 1, but not significantly.

2. After dividing the data into three groups by sex, the RMSE of the Infant dataset was much lower than that of the `Abalone_Age_NoSex` dataset, while the RMSEs of the SexM and SexF datasets were slightly higher than that dataset, showing a significant change in model performance, which may reflect the inherent differences in samples of different genders. This means that if the prediction effect on SexI is better after splitting.

3. In the second part of the experiment, with $\alpha = 1$, as lambda increases, the best model has only 6 variables, and the variable with a coefficient of 0 is Height. This is the same as the result of the Best Subset Selection in Part 2. At the same time, the optimal $\lambda_{\min}$ values of different datasets are different, indicating that different characteristics of the data may require different degrees of regularization.

4. Model performance: Looking at the RMSE, Ridge regression ($\alpha = 0$) performed best on this specific dataset, although the improvement was very minimal. The performance of the elastic network ($\alpha = 0.5$) and LASSO regression ($\alpha = 1$) was similar but slightly inferior.

5. Variable selection: LASSO regression performs variable selection by reducing some coefficients to zero, while the elastic network and Ridge regression retain all variables. This may mean that LASSO or an elastic network (when alpha is close to 1) might be more applicable when variable selection is needed.

6. Regularization strength: The value of $\lambda_{\min}$ decreases as alpha decreases, indicating that stronger regularization may be needed for Ridge regression to achieve optimal performance.

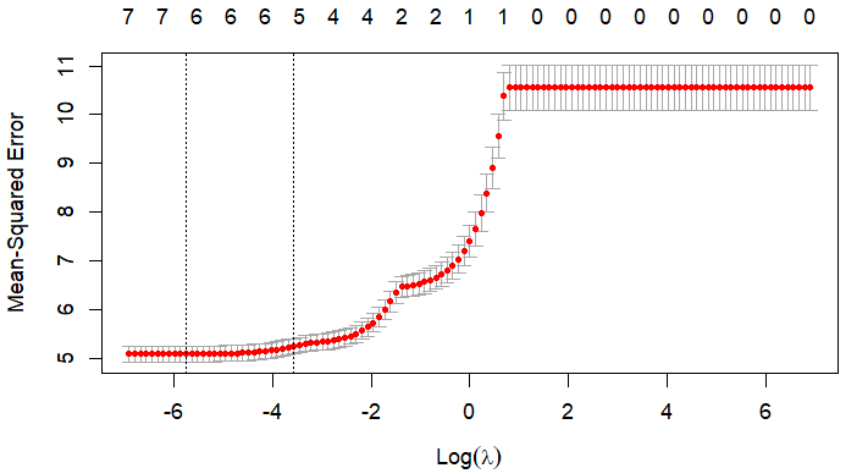


Figure 8: Lasso Regression with abalone_Age_NoSex Dataset

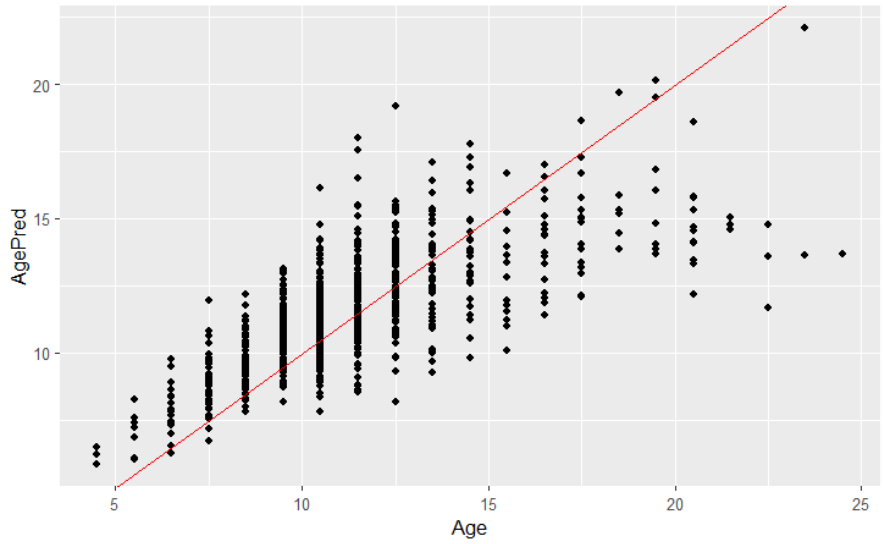


Figure 9: The Pridicted Value vs True

3.2 Random Forest

3.2.1 Reason for choosing This instead of Neural Network

Since random forest is an ensemble method based on decision trees, it provides better model interpretability. We can easily see which features have the greatest impact on the prediction

results and even interpret the decision paths of individual decision trees. In contrast, neural networks are often seen as "black box" models, difficult to interpret their internal workings and decision-making processes. Besides, unlike neural networks, it is not sensitive to the scaling of input features. Decision trees split based on the threshold of features, so there is no need for feature standardization or normalization. Due to the averaging of predictions from multiple decision trees, random forests are usually more robust to noise and outliers in the data.

3.2.2 Introduction to Random Forest

Random forest is a popular and powerful machine learning algorithm that belongs to the category of ensemble learning methods. It consists of multiple decision trees and makes predictions by aggregating the results of these trees, thereby improving the accuracy and robustness of the model.

3.2.3 Model Assumption

1. Variable correlation: Random forest reduces the impact of high correlation among variables by choosing the best feature from a random subset of features at each decision tree split. Therefore, the model assumes that even if there is a certain degree of correlation among features, it will not significantly negatively affect model performance.

2. Model complexity and overfitting: Random forest improves predictive accuracy by integrating multiple decision trees and reduces the risk of overfitting by introducing randomness. Therefore, the model assumes that it can maintain good generalization ability even on high-dimensional data or complex data structures.

3. Feature importance: Random forest can assess the contribution of each feature to the prediction results, so the model assumes that some features may be more important than others for predicting age.

4. No need for feature scaling: Random forest is not sensitive to the scale of features, so the model assumes that there is no need to standardize or normalize features.

3.2.4 Model Tuning

When optimizing model performance using the randomForest function in R, key parameter adjustments include ntree (number of trees), mtry (number of features to consider at each

split), nodesize (minimum number of samples in terminal nodes), sampsize (sample size for each tree), and maxnodes (maximum number of nodes per tree). Increasing ntree can enhance model stability, adjusting mtry affects model diversity and flexibility, and adjusting nodesize and maxnodes help prevent overfitting.

In addition to the parameter adjustment within randomForest, we can also try different feature combinations multiple times when predicting Age, such as removing Sex or Height, and viewing the variance explanatory and RMSE changes of the model.

3.2.5 Result Analysis

In this part, I trained the random forest using four data models. The first model: using all features; the second model: excluding gender features; the third model: excluding height features; the fourth model: excluding both gender and height features. In all four models, I defaulted to using 500 trees and considering 2 variables at each split, and came to the following conclusions:

1. The impact of features: Excluding specific features (such as gender or height) has a relatively limited impact on the overall performance of the model. This may indicate that these features are not decisive factors for predicting age, or their information may have been indirectly captured by the model through other features.

2. Training and testing errors: In all models, the training error is consistently significantly lower than the testing error. This is a typical sign of overfitting, indicating that the model may be too sensitive to the training data, capturing some noise in the training data that does not commonly exist in the testing data.

3. Model-explained variance: All models can explain about 55% of the variance, indicating that the random forest model has some ability to capture the intrinsic structure of the dataset, but there is still a certain proportion of variance not explained by the model, which may be caused by random noise in the data or unobserved variables.

4. Mean squared residuals: The difference in mean squared residuals between the models is not significant, further confirming that removing a single feature has a limited impact on model predictive performance.

5. Running time: The running time of the models is relatively consistent, indicating that the removal of features has little impact on computational efficiency. This may be because the random forest algorithm itself is highly efficient in handling features, and the number of trees in the model (500 trees) is the main factor determining the running time.

4 Model Comparison

Table 2: Summary of Lp Linear Regression and Random Forest Results

Metric/Model	**Lp Linear Regression**	**Random Forest**
Model Coefficients	Provided for each feature	-
Training Error (RMSE)	1.55983 to 2.541056	1.012617 to 1.026651
Testing Error (RMSE)	1.646548 to 2.502254	2.064943 to 2.08656
Running Time	0.08 to 0.14 seconds	4.35 to 4.51 seconds
Model Details	$\alpha = 0, 0.5, 1$	500 trees, 2 variables per split
Mean Squared Residuals	-	4.717192 to 4.764544
Percentage of Explained Variance	-	54.83% to 55.27%

This table (Table 2) provides a clear comparison of key metrics between Lp linear regression and random forest.

Performance: The testing errors of random forest and Ridge regression are similar, indicating that both methods have similar predictive capabilities on this specific dataset. However, random forest has a slight advantage in explaining variance.

Interpretability: Lp-norm regression provides coefficients for features, offering better model interpretability. Although random forest can assess feature importance, its overall interpretability is not as good as linear models.

Computational efficiency: The running time of Lp norm linear regression is much lower than that of random forest on my personal computer, especially important when dealing with large datasets. Although the ablone dataset is not big enough, I can still obeserve that difference in speed.

Handling non-linear relationships: Random forest is better at handling non-linear relationships, while Ridge regression may need additional feature engineering to capture these relationships.

In conclusion, which model to choose depends on the specific application scenario. If model interpretability is a key consideration and the data relationships are close to linear, Lp-norm regression might be a better choice. For datasets with complex non-linear relationships or when model performance is the sole focus, random forest might offer better results, albeit at the cost of longer training time.

5 Results and Conclusion

5.1 Conclusion

The analysis of the abalone dataset through Lp linear regression and random forest models has provided valuable insights into predicting the age of abalones based on their physical measurements. The Lp linear regression, particularly Ridge regression, offered interpretable coefficients for each feature, highlighting the importance of feature selection and regularization in addressing multicollinearity and overfitting.

Random forest, on the other hand, showcased its strength in handling non-linear relationships and providing robust predictions despite its longer computation time. The ability to assess feature importance in random forest models is beneficial for understanding the driving factors behind abalone age, even if the model itself is less interpretable than linear regression models.

5.2 Future Work

Data Collection: Expanding the dataset to include additional relevant features, such as environmental conditions (water temperature, salinity, etc.), could provide a more holistic view of the factors influencing abalone age. Collecting data over a longer period could also help in understanding temporal variations in growth patterns. We can even use the Embedded Equipments with sensors or camera to get more of this kind of data automatically.

Modelling Approaches: Exploring more complex models like ensemble methods beyond random forest deep learning approaches could uncover non-linear relationships not captured by the current models. I made the tradeoff between deep learning and random forest because I was afraid about I can't explain the principle why neural network perform better when I change some parameters. Additionally, incorporating cross-validation and hyperparameter tuning more extensively could improve model performance and reliability.

References

[1] Warwick Nash, Tracy Sellers, Simon Talbot, Andrew Cawthorn, and Wes Ford. Abalone. UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C55C7W.

A Dataset Router

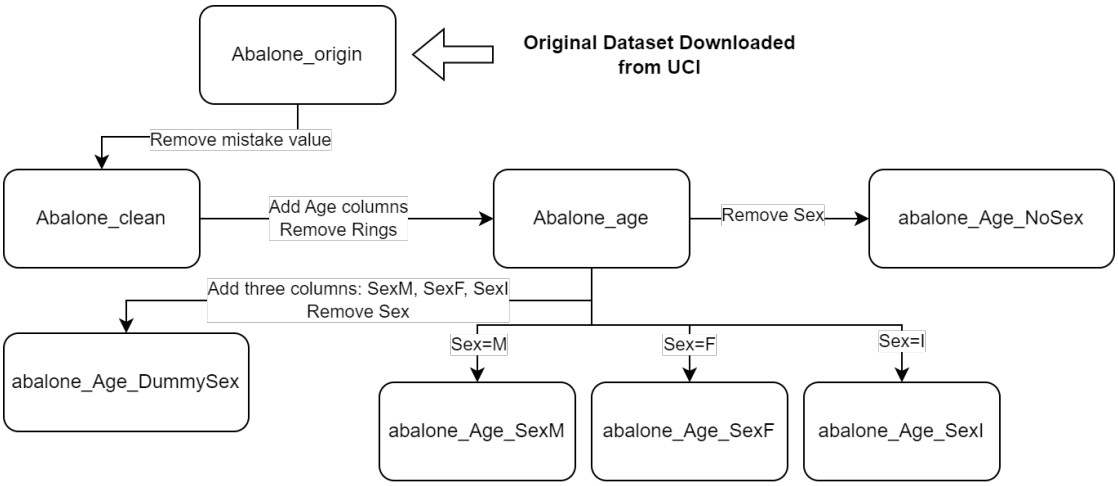


Figure 10: Dataset Family Used in This Report

B RMarkdown Source

I used rmarkwon to write and run R code analysis and output PDF version below.

Machine Learning Assignment 2

Zehao Qian

2024-02-12

Abalone Learning

Part 0: Install requirements and clean Env values

```
# install.packages('dplyr')
# install.packages('tidyr')
# install.packages('ggplot2')
# install.packages('glmnet')
# install.packages('randomForest')
# install.packages('GGally')
```

```
# -----------------------------------------------------
# clear the environment var area
rm(list = ls())
# clear all plots
graphics.off()
# clear the console area
cat("\014")
```

```
# ------------------------------------------------------
```

Part 1: Data Read and Cleaning

Step 1 Read the Data

```
library(dplyr)
```

```
##
## 载入程辑包：'dplyr'
```

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

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

```
# current directory
current_directory = getwd()
# read_csv
# joint file path
file = file.path(current_directory, "abalone/abalone.data")
abalone_origin = read.csv(file, header = FALSE)

# Manually set the column names
colnames(abalone_origin) =
  c(
    "Sex",
    "Length",
    "Diameter",
    "Height",
    "WholeWeight",
    "ShuckedWeight",
    "VisceraWeight",
    "ShellWeight",
    "Rings"
  )

# Display the first few rows of the dataframe to verify
head(abalone_origin)
```

```
##   Sex Length Diameter Height WholeWeight ShuckedWeight VisceraWeight
## 1   M  0.455    0.365  0.095      0.5140        0.2245        0.1010
## 2   M  0.350    0.265  0.090      0.2255        0.0995        0.0485
## 3   F  0.530    0.420  0.135      0.6770        0.2565        0.1415
## 4   M  0.440    0.365  0.125      0.5160        0.2155        0.1140
## 5   I  0.330    0.255  0.080      0.2050        0.0895        0.0395
## 6   I  0.425    0.300  0.095      0.3515        0.1410        0.0775
##   ShellWeight Rings
## 1       0.150    15
## 2       0.070     7
## 3       0.210     9
## 4       0.155    10
## 5       0.055     7
## 6       0.120     8
```

Step 2 Check if the Abalone Data Set Exist Empty Values

```
any(is.na(abalone_origin))
```

```
## [1] FALSE
```

Step3 Data Cleaning

```
# names(abalone_origin)
cleaned_abalone <- abalone_origin %>%
  filter(Height > 0) %>%
  filter(WholeWeight >= ShuckedWeight + ShellWeight)
```

Part 2 Exploratory Data Analysis

Step 0 Data Set

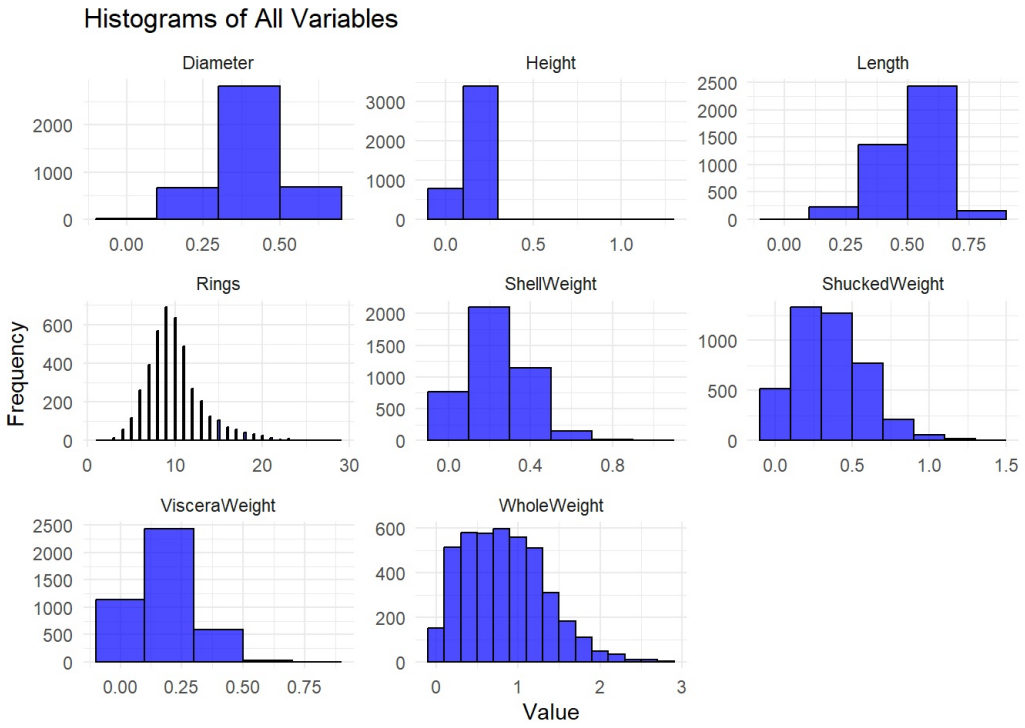
```
# Add a new column 'Age' which is 'Rings' + 1.5
# Add 'Age' column and remove 'Rings' column
abalone_Age = cleaned_abalone %>%
  mutate(Age = Rings + 1.5) %>%
  select(-Rings)
# Convert 'Sex' to dummy variables (one-hot encoding)
# dummySex = model.matrix(~ Sex - 1, data = abalone_Age)
# Bind the dummy variables back to the original dataset (excluding the original 'Sex' column)
abalone_Age_DummySex = abalone_Age %>%
  bind_cols(model.matrix( ~ Sex - 1, data = abalone_Age)) %>%
  select(-Sex)
# Dataset without sex
abalone_Age_NoSex = abalone_Age %>% select(-Sex)
# Dataset with sex=M
abalone_Age_SexM = abalone_Age %>% filter(Sex == 'M') %>% select(-Sex)
# Dataset with sex=F
abalone_Age_SexF = abalone_Age %>% filter(Sex == 'F') %>% select(-Sex)
# Dataset with sex=I
abalone_Age_SexI = abalone_Age %>% filter(Sex == 'I') %>% select(-Sex)
```

Step 1 Single Variable Analysis

```
summary(abalone_origin)
```

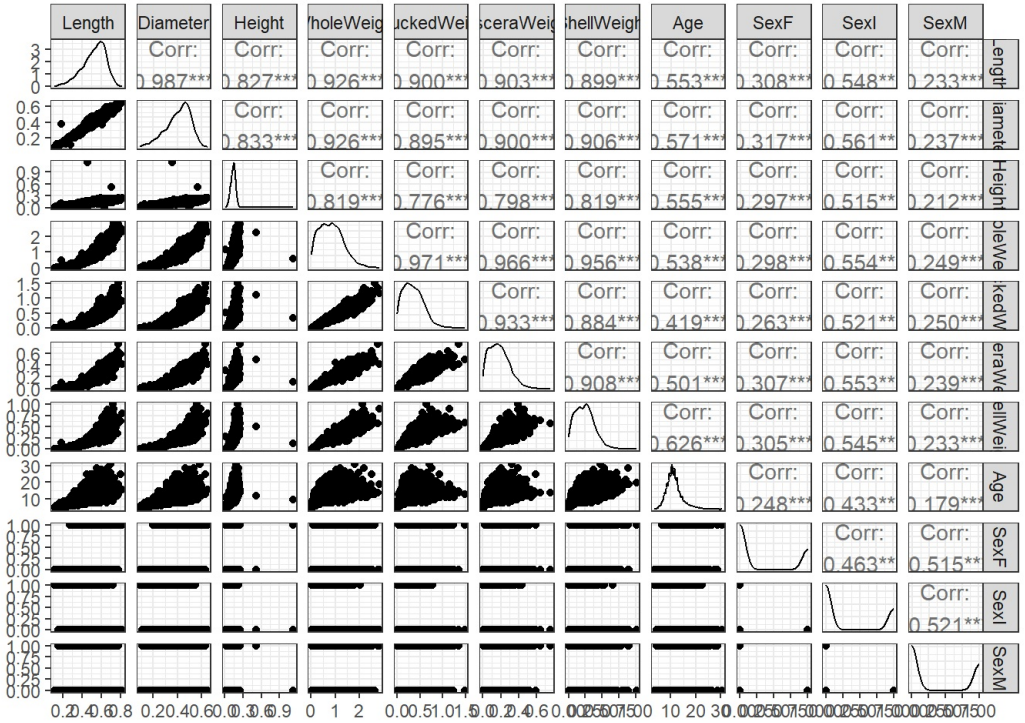
```
##      Sex                Length         Diameter          Height
##  Length:4177        Min.   :0.075   Min.   :0.0550   Min.   :0.0000
##  Class :character   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150
##  Mode  :character   Median :0.545   Median :0.4250   Median :0.1400
##                     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
##   WholeWeight     ShuckedWeight    VisceraWeight     ShellWeight
##  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
##      Rings
##  Min.   : 1.000
##  1st Qu.: 8.000
##  Median : 9.000
##  Mean   : 9.934
##  3rd Qu.:11.000
##  Max.   :29.000
```

```
# hist
library(tidyr)
library("ggplot2")
abalone_origin_long_data <- abalone_origin %>%
  pivot_longer(cols = where(is.numeric),
               names_to = "Variable",
               values_to = "Value")

ggplot(abalone_origin_long_data, aes(x = Value)) +
  geom_histogram(
    binwidth = 0.2,
    fill = "blue",
    color = "black",
    alpha = 0.7
  ) +  # Adjust binwidth as needed
  facet_wrap(~ Variable, scales = "free") +
  labs(title = "Histograms of All Variables", x = "Value", y = "Frequency") +
  theme_minimal()
```



Step 2 Bivariate Analysis

```
library("GGally")
ggpairs(abalone_Age_DummySex)+theme_bw()
```



Step 3 Find the Best Subset

```
# Best Subset Selection with one-hot encoding DataSet
library(leaps)
best_models.abalone_Age_DummySex = regsubsets(Age ~ ., data = abalone_Age_DummySex, nvmax=10)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 1 linear dependencies found
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : nvmax reduced to 9
```

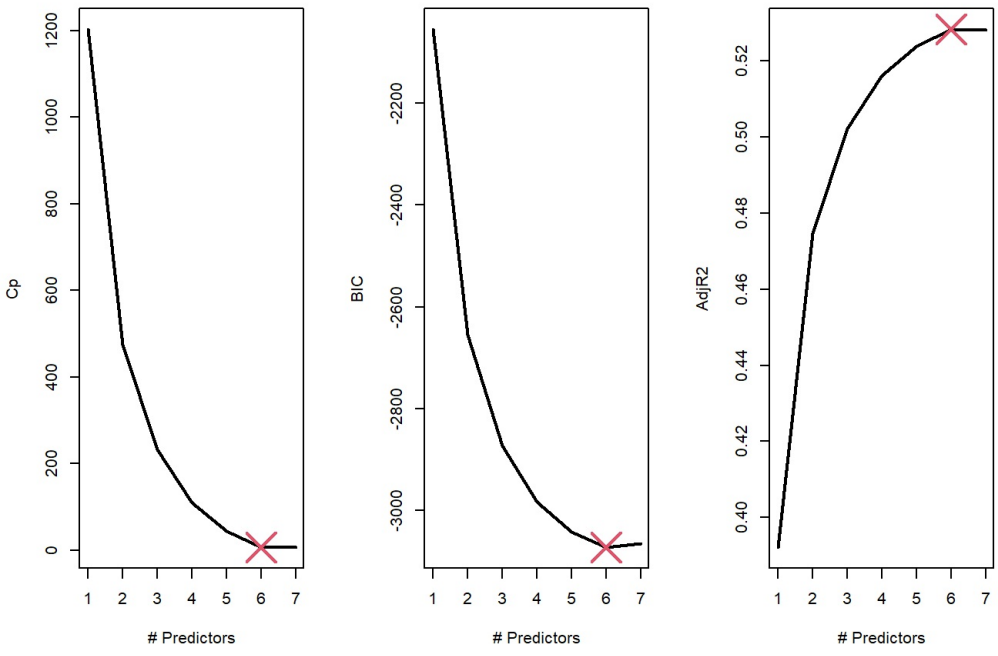
```
summary(best_models.abalone_Age_DummySex)
```

```
## Subset selection object
## Call: regsubsets.formula(Age ~ ., data = abalone_Age_DummySex, nvmax = 10)
## 10 Variables  (and intercept)
##               Forced in Forced out
## Length            FALSE      FALSE
## Diameter          FALSE      FALSE
## Height            FALSE      FALSE
## WholeWeight       FALSE      FALSE
## ShuckedWeight     FALSE      FALSE
## VisceraWeight     FALSE      FALSE
## ShellWeight       FALSE      FALSE
## SexF              FALSE      FALSE
## SexI              FALSE      FALSE
## SexM              FALSE      FALSE
## 1 subsets of each size up to 9
## Selection Algorithm: exhaustive
##          Length Diameter Height WholeWeight ShuckedWeight VisceraWeight
## 1  ( 1 ) " "    " "      " "    " "         " "           " "
## 2  ( 1 ) " "    " "      " "    " "         "*"           " "
## 3  ( 1 ) " "    "*"      " "    "*"         "*"           " "
## 4  ( 1 ) " "    "*"      " "    "*"         "*"           "*"
## 5  ( 1 ) " "    "*"      " "    "*"         "*"           "*"
## 6  ( 1 ) " "    "*"      "*"    "*"         "*"           "*"
## 7  ( 1 ) " "    "*"      "*"    "*"         "*"           "*"
## 8  ( 1 ) " "    "*"      "*"    "*"         "*"           "*"
## 9  ( 1 ) "*"    "*"      "*"    "*"         "*"           "*"
##          ShellWeight SexF SexI SexM
## 1  ( 1 ) "*"         " "  " "  " "
## 2  ( 1 ) "*"         " "  " "  " "
## 3  ( 1 ) " "         " "  " "  " "
## 4  ( 1 ) " "         " "  " "  " "
## 5  ( 1 ) " "         " "  "*"  " "
## 6  ( 1 ) " "         " "  "*"  " "
## 7  ( 1 ) "*"         " "  "*"  " "
## 8  ( 1 ) "*"         "*"  "*"  " "
## 9  ( 1 ) "*"         "*"  "*"  " "
```

```
best_models.abalone_Age_NoSex = regsubsets(Age ~ ., data = abalone_Age_NoSex)
BBSsummary = summary(best_models.abalone_Age_NoSex)
BBSsummary
```

```
## Subset selection object
## Call: regsubsets.formula(Age ~ ., data = abalone_Age_NoSex)
## 7 Variables  (and intercept)
##               Forced in Forced out
## Length            FALSE      FALSE
## Diameter          FALSE      FALSE
## Height            FALSE      FALSE
## WholeWeight       FALSE      FALSE
## ShuckedWeight     FALSE      FALSE
## VisceraWeight     FALSE      FALSE
## ShellWeight       FALSE      FALSE
## 1 subsets of each size up to 7
## Selection Algorithm: exhaustive
##          Length Diameter Height WholeWeight ShuckedWeight VisceraWeight
## 1  ( 1 ) " "    " "      " "    " "         " "           " "
## 2  ( 1 ) " "    " "      " "    " "         "*"           " "
## 3  ( 1 ) " "    "*"      " "    "*"         "*"           " "
## 4  ( 1 ) " "    "*"      " "    "*"         "*"           "*"
## 5  ( 1 ) " "    "*"      "*"    "*"         "*"           "*"
## 6  ( 1 ) " "    "*"      "*"    "*"         "*"           "*"
## 7  ( 1 ) "*"    "*"      "*"    "*"         "*"           "*"
##          ShellWeight
## 1  ( 1 ) "*"
## 2  ( 1 ) "*"
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
## 6  ( 1 ) "*"
## 7  ( 1 ) "*"
```

```
par(mfrow = c(1,3)) # allows for 3 plots to be plotted side by side
# ------------------------------------------------------
plot(BBSsummary$cp,
     xlab = "# Predictors", #x-axis label
     ylab = "Cp", #y-axis label
     type = "l", #line plot
     lwd = 2) #line thickness

cp_min = which.min(BBSsummary$cp)

#overlay the minimum of cp on the previous plot using the points function
points(cp_min,
       BBSsummary$cp[cp_min],
       pch = 4, #cross symbol used
       col = 2, #red colour
       cex = 4, #make it bigger!
       lwd = 2) #make the cross lines thicker
# ------------------------------------------------------
plot(BBSsummary$bic,
     xlab = "# Predictors", #x-axis label
     ylab = "BIC", #y-axis label
     type = "l", #line plot
     lwd = 2) #line thickness

bic_min = which.min(BBSsummary$bic)

points(bic_min,
       BBSsummary$bic[bic_min],
       pch = 4, #cross symbol used
       col = 2, #red colour
       cex = 4, #make it bigger!
       lwd = 2) #make the cross lines thicker
# ------------------------------------------------------
plot(BBSsummary$adjr2,
     xlab = "# Predictors", #x-axis label
     ylab = "AdjR2", #y-axis label
     type = "l", #line plot
     lwd = 2) #line thickness

adjr2_max = which.max(BBSsummary$adjr2)
points(adjr2_max,
       BBSsummary$adjr2[adjr2_max],
       pch = 4, #cross symbol used
       col = 2, #red colour
       cex = 4, #make it bigger!
       lwd = 2) #make the cross lines thicker
```



Part 3 Modeling

Lasso, Ridge and Elastic Net Regression

Split the Data Set with training and testing set

```
# ------------------------------------------------------------
# abalone_Age
set.seed(2024)
ind = sample(1:nrow(abalone_Age),
             size = 800,
             replace = FALSE)
training.abalone_Age = abalone_Age[-ind, ]
testing.abalone_Age = abalone_Age[ind, ]
# ------------------------------------------------------------
# abalone_Age_NoSex
set.seed(2024)
ind = sample(1:nrow(abalone_Age_NoSex),
             size = 800,
             replace = FALSE)
training.abalone_Age_NoSex = abalone_Age_NoSex[-ind, ]
testing.abalone_Age_NoSex = abalone_Age_NoSex[ind, ]
# ------------------------------------------------------------
# abalone_Age_DummySex
set.seed(2024)
ind = sample(1:nrow(abalone_Age_DummySex),
             size = 800,
             replace = FALSE)
training.abalone_Age_DummySex = abalone_Age_DummySex[-ind, ]
testing.abalone_Age_DummySex = abalone_Age_DummySex[ind, ]
# ------------------------------------------------------------
# abalone_Age_SexF
set.seed(2024)
ind = sample(1:nrow(abalone_Age_SexF),
             size = 300,
             replace = FALSE)
training.abalone_Age_SexF = abalone_Age_SexF[-ind, ]
testing.abalone_Age_SexF = abalone_Age_SexF[ind, ]
# ------------------------------------------------------------
# abalone_Age_SexM
set.seed(2024)
ind = sample(1:nrow(abalone_Age_SexM),
             size = 300,
             replace = FALSE)
training.abalone_Age_SexM = abalone_Age_SexM[-ind, ]
testing.abalone_Age_SexM = abalone_Age_SexM[ind, ]
# ------------------------------------------------------------
# abalone_Age_SexI
set.seed(2024)
ind = sample(1:nrow(abalone_Age_SexI),
             size = 300,
             replace = FALSE)
training.abalone_Age_SexI = abalone_Age_SexI[-ind, ]
testing.abalone_Age_SexI = abalone_Age_SexI[ind, ]
```

```
library(glmnet)
```

```
## 载入需要的程辑包：Matrix
```

```
##
## 载入程辑包：'Matrix'
```

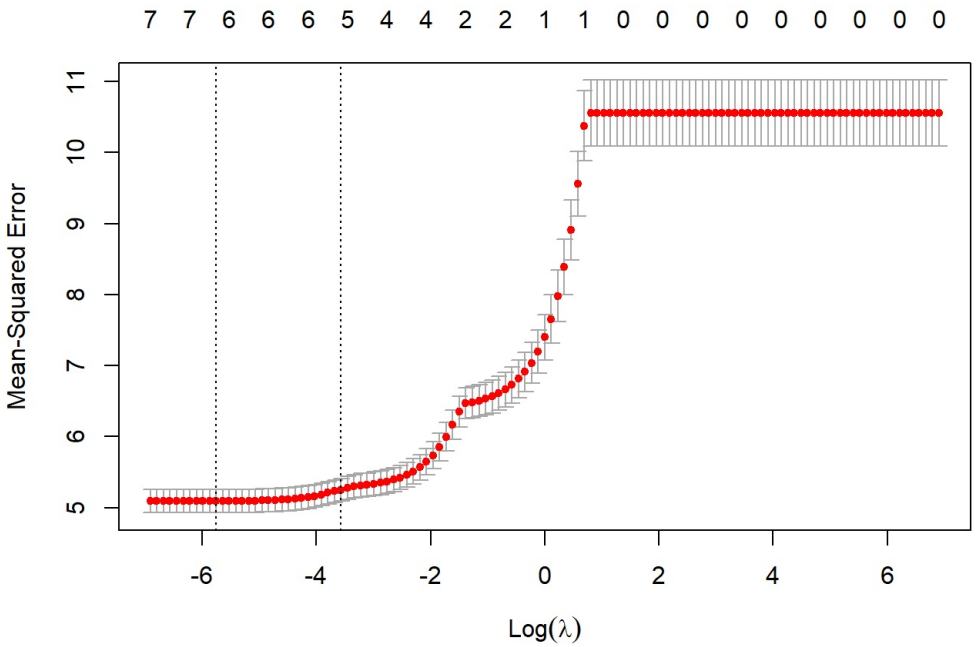
```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

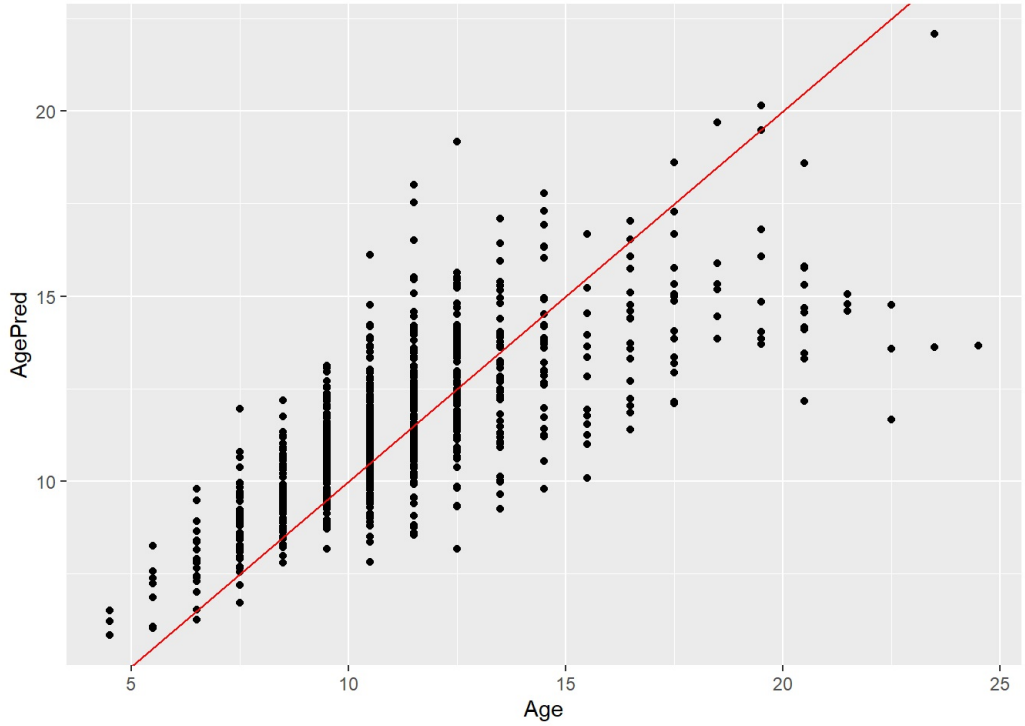
```
## Loaded glmnet 4.1-8
```

```r
# Define the function
perform_cv_glmnet <-
  function(TrainSet, TestSet, target_var, alpha) {
    print('-------------------------------------------------')
    start_time <- Sys.time()
    # Generate a sequence of lambda values
    lambdas <- 10 ^ seq(-3, 3, by = 0.05)
    # Prepare the data
    y_train <- as.matrix(TrainSet[[target_var]])
    X_train <-
      as.matrix(TrainSet[,!(names(TrainSet) %in% target_var)])
    y_test <- as.matrix(TestSet[[target_var]])
    X_test <- as.matrix(TestSet[,!(names(TestSet) %in% target_var)])
    # Fit the model using cross-validation
    cv_fit <-
      cv.glmnet(
        X_train,
        y_train,
        alpha = alpha,
        lambda = lambdas,
        nfolds = 10,
        thresh = 1e-10
      )
    # Extract the lambda that minimizes the cross-validation error
    lambda_min <- cv_fit$lambda.min
    # Extract coefficients at the best lambda
    coef_best <- coef(cv_fit, s = "lambda.min")
    # ------------------------------------------------------------
    # test on training set
    # Make predictions using the best lambda
    predictionsTrain <-
      predict(cv_fit, s = "lambda.min", newx = X_train)
    # Calculate RMSE
    RMSE_Train <- sqrt(mean((predictionsTrain - y_train) ^ 2))
    # ------------------------------------------------------------
    # test on testing set
    predictionsTest <-
      predict(cv_fit, s = "lambda.min", newx = X_test)
    # Calculate RMSE
    RMSE_Test <- sqrt(mean((predictionsTest - y_test) ^ 2))
    # ------------------------------------------------------------
    # Return the results
    result = list(
      alpha = alpha,
      lambda_min = lambda_min,
      coef_best = coef_best,
      RMSE_Train = RMSE_Train,
      RMSE_Test = RMSE_Test
    )
    print(result)
    end_time <- Sys.time()
    duration <- end_time - start_time
    print(duration)
    plot(cv_fit)
    # compare the age with the pridict age
    Test = TestSet %>% mutate(AgePred = predictionsTest)
    ggplot(Test) + geom_point(aes(x=Age, y=AgePred)) + geom_abline(intercept = 0,slope = 1, colour = "red")


  }
```

```r
# compare different alpha (1, 0.5, 0) with the same dataset: abalone_Age_NoSex
set.seed(2024)
perform_cv_glmnet(training.abalone_Age_NoSex,
                  testing.abalone_Age_NoSex,
                  'Age',
                  1)
```

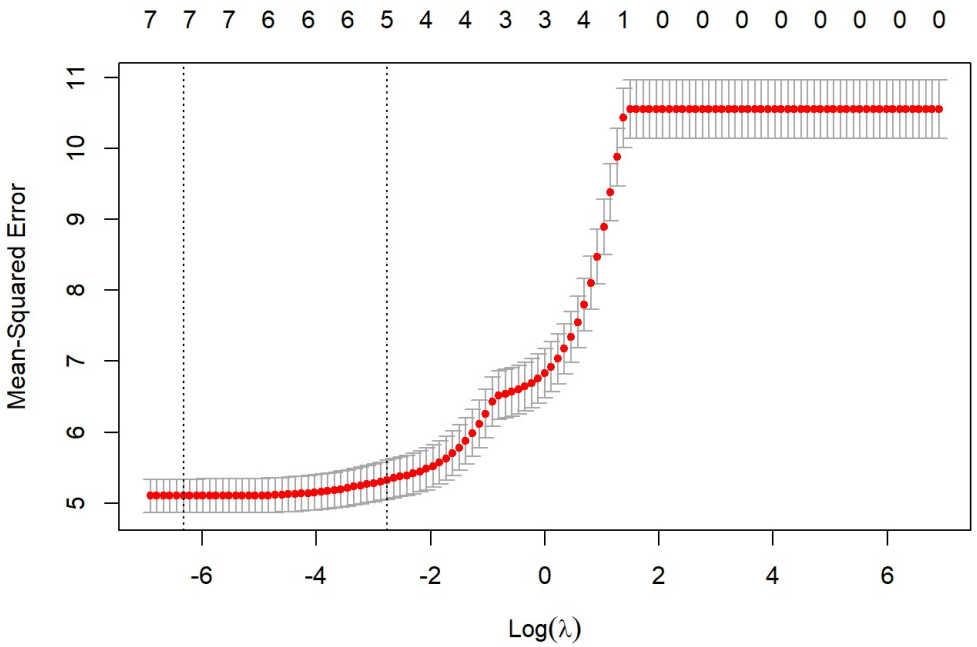
```
## [1] "------------------------------------------------"
## $alpha
## [1] 1
##
## $lambda_min
## [1] 0.003162278
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     4.402160
## Length          .
## Diameter       12.101448
## Height         10.243084
## WholeWeight     8.739695
## ShuckedWeight -19.931621
## VisceraWeight  -9.066490
## ShellWeight     9.457625
##
## $RMSE_Train
## [1] 2.228554
##
## $RMSE_Test
## [1] 2.135823
##
## Time difference of 0.122649 secs
```

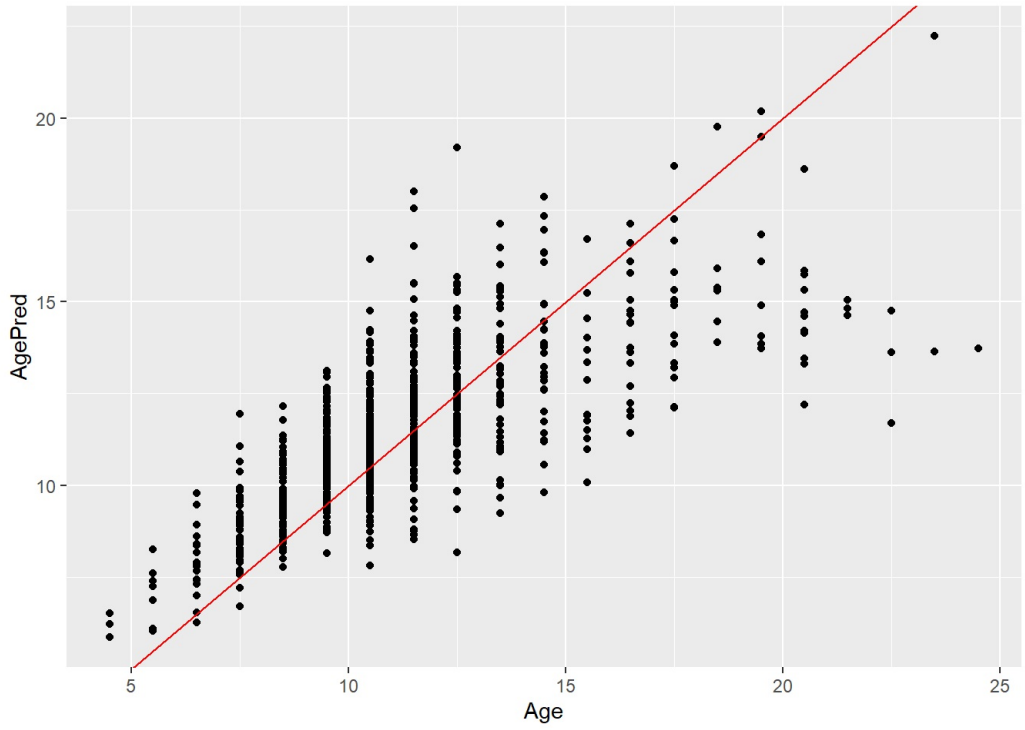





```
perform_cv_glmnet(training.abalone_Age_NoSex,
                  testing.abalone_Age_NoSex,
                  'Age',
                  0.5)
```

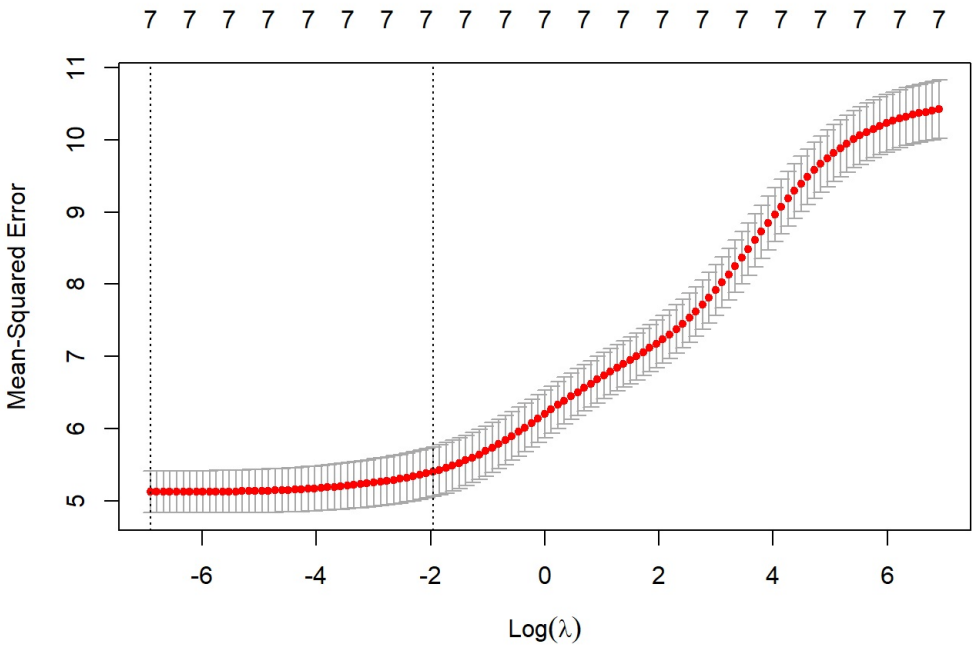
```
## [1] "--------------------------------------------------"
## $alpha
## [1] 0.5
##
## $lambda_min
## [1] 0.001778279
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)     4.4246041
## Length         -0.9339564
## Diameter       13.2359107
## Height         10.3219553
## WholeWeight     9.0449441
## ShuckedWeight -20.2052747
## VisceraWeight  -9.5694748
## ShellWeight     9.1615489
##
## $RMSE_Train
## [1] 2.228101
##
## $RMSE_Test
## [1] 2.134793
##
## Time difference of 0.09163499 secs
```

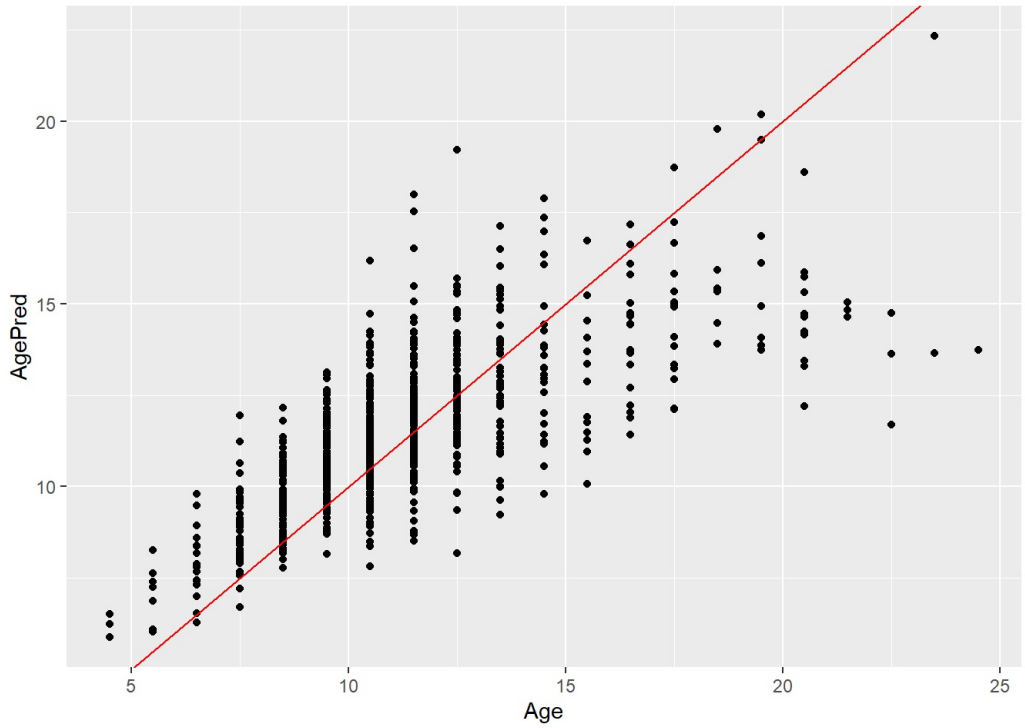





```
perform_cv_glmnet(training.abalone_Age_NoSex,
                  testing.abalone_Age_NoSex,
                  'Age',
                  0)
```

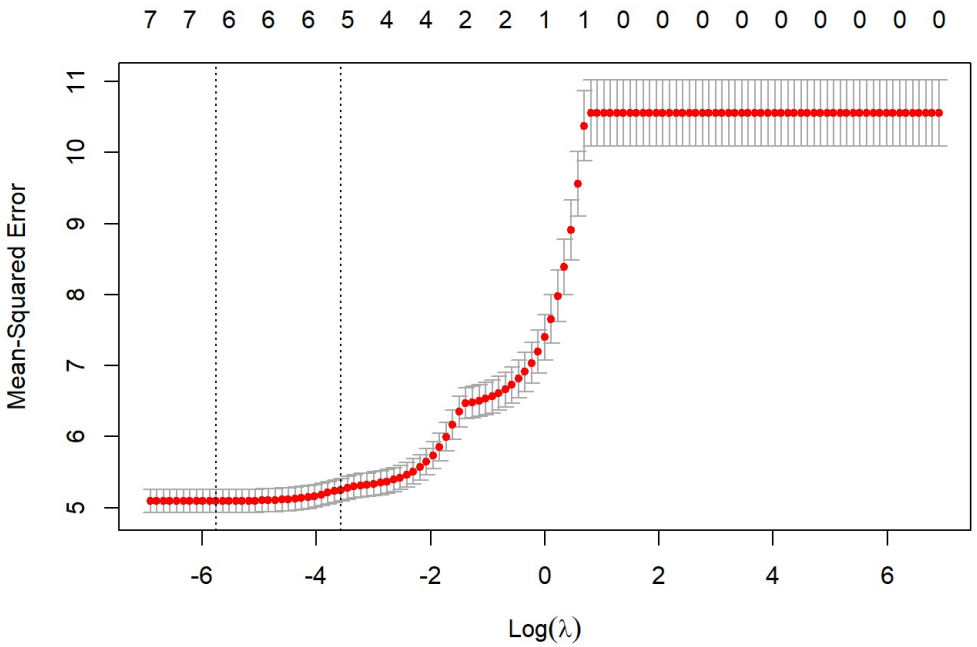
```
## [1] "------------------------------------------------"
## $alpha
## [1] 0
##
## $lambda_min
## [1] 0.001
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     4.441658
## Length         -1.465628
## Diameter       13.871648
## Height         10.350708
## WholeWeight     9.261584
## ShuckedWeight -20.412027
## VisceraWeight  -9.855173
## ShellWeight     8.929117
##
## $RMSE_Train
## [1] 2.227943
##
## $RMSE_Test
## [1] 2.134162
##
## Time difference of 0.1205761 secs
```

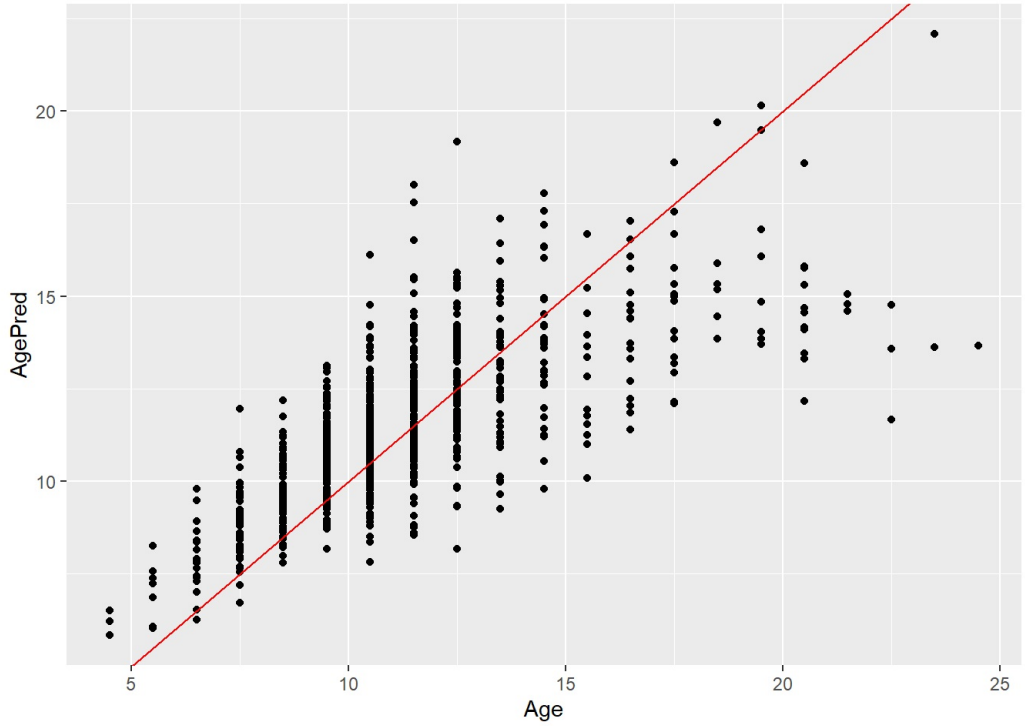





```
# compare same alpha (1) with different datase
set.seed(2024)
perform_cv_glmnet(training.abalone_Age_NoSex,
                  testing.abalone_Age_NoSex,
                  'Age',
                  1)
```

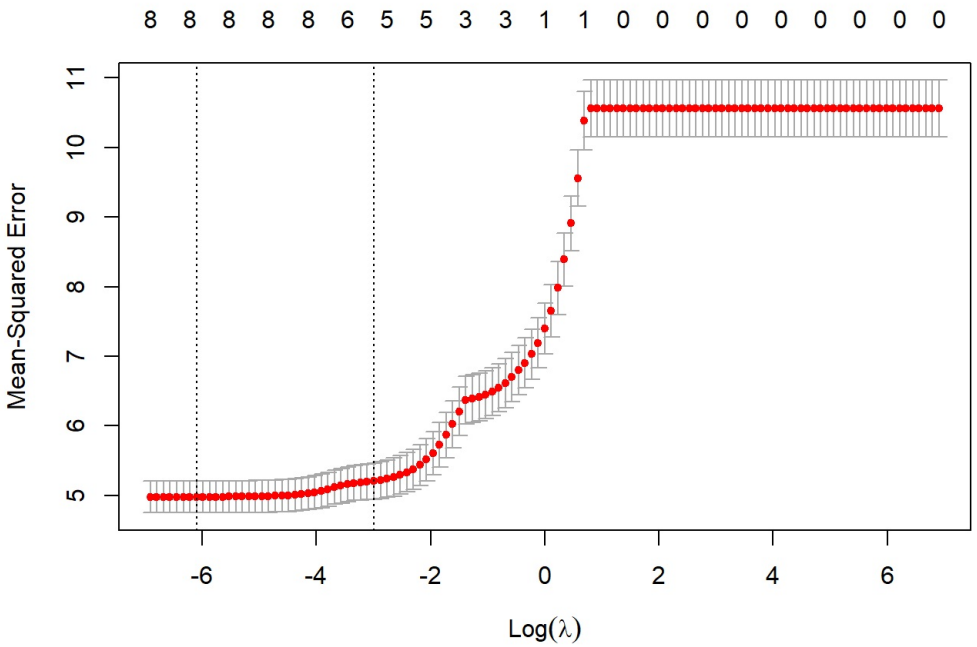
```
## [1] "------------------------------------------------"
## $alpha
## [1] 1
##
## $lambda_min
## [1] 0.003162278
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     4.402160
## Length          .
## Diameter       12.101448
## Height         10.243084
## WholeWeight     8.739695
## ShuckedWeight -19.931621
## VisceraWeight  -9.066490
## ShellWeight     9.457625
##
## $RMSE_Train
## [1] 2.228554
##
## $RMSE_Test
## [1] 2.135823
##
## Time difference of 0.1145971 secs
```

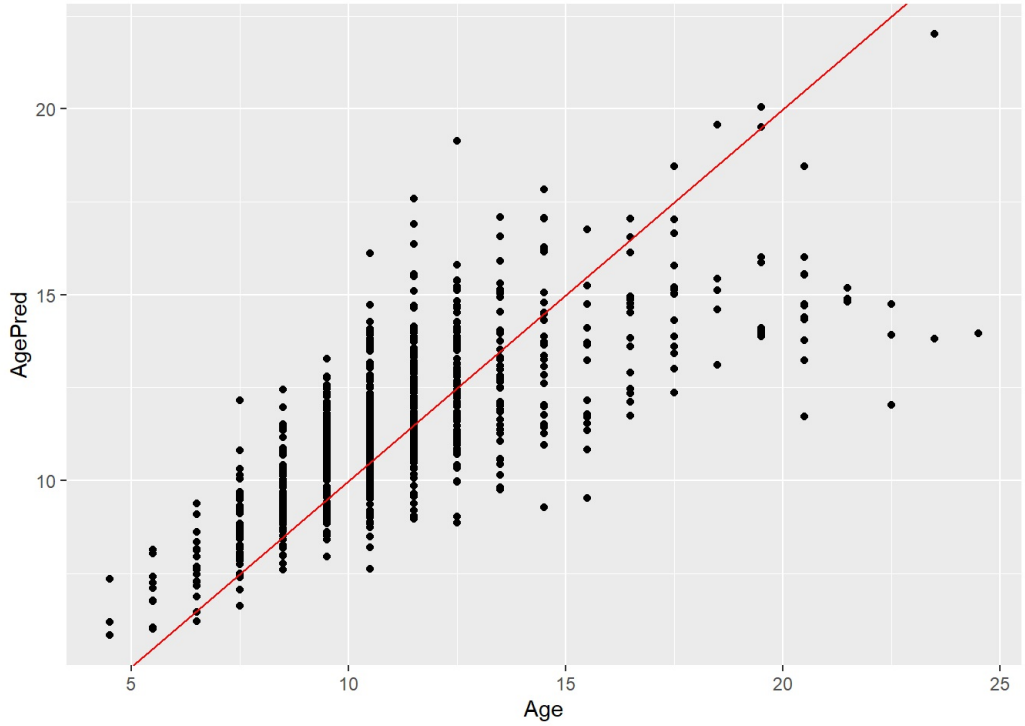





```
perform_cv_glmnet(training.abalone_Age_DummySex,
                  testing.abalone_Age_DummySex,
                  'Age',
                  1)
```

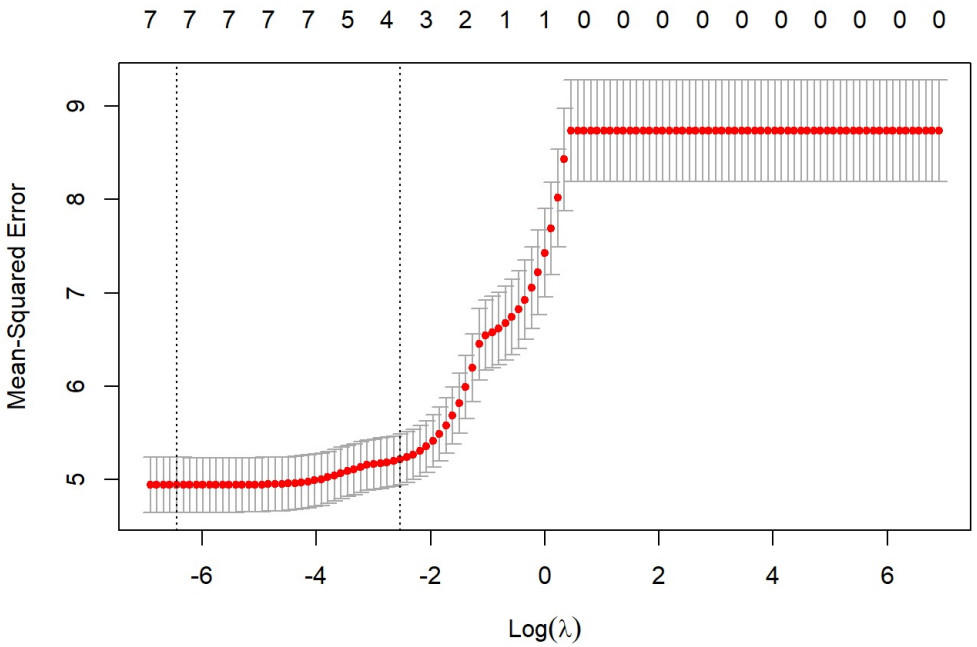
```
## [1] "-------------------------------------------------"
## $alpha
## [1] 1
##
## $lambda_min
## [1] 0.002238721
##
## $coef_best
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)     5.37775119
## Length          .
## Diameter       11.08341866
## Height          9.29015697
## WholeWeight     8.79877105
## ShuckedWeight -19.82037919
## VisceraWeight -10.19074763
## ShellWeight     9.20767735
## SexF            .
## SexI           -0.85137373
## SexM            0.05098286
##
## $RMSE_Train
## [1] 2.202805
##
## $RMSE_Test
## [1] 2.113269
##
## Time difference of 0.1174269 secs
```

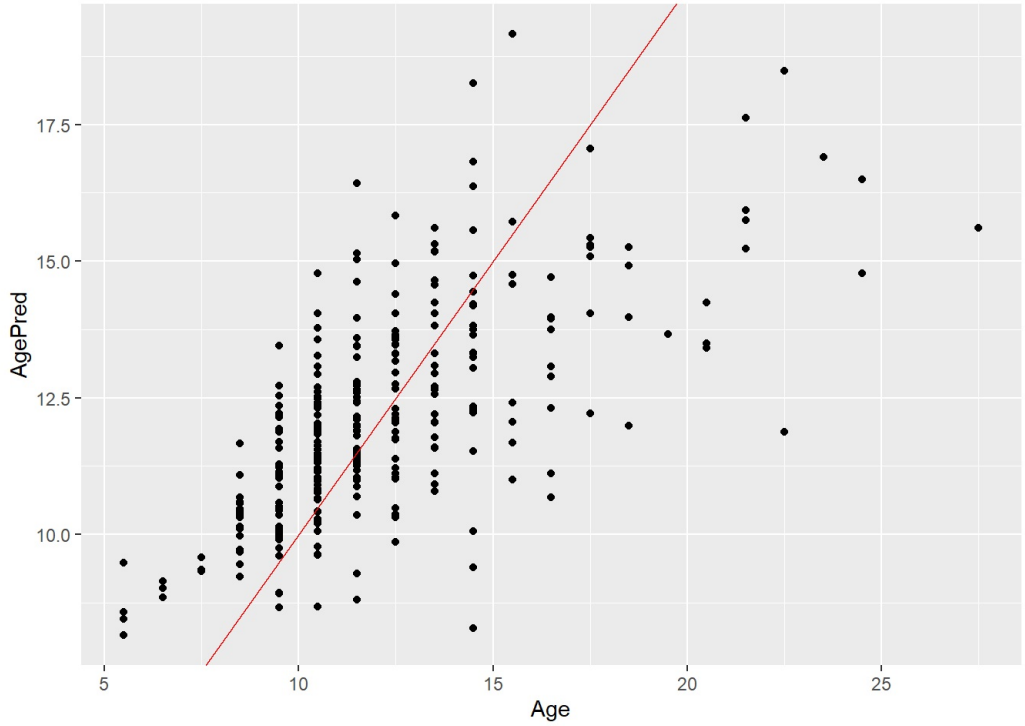





```
perform_cv_glmnet(training.abalone_Age_SexM,
                  testing.abalone_Age_SexM,
                  'Age',
                  1)
```

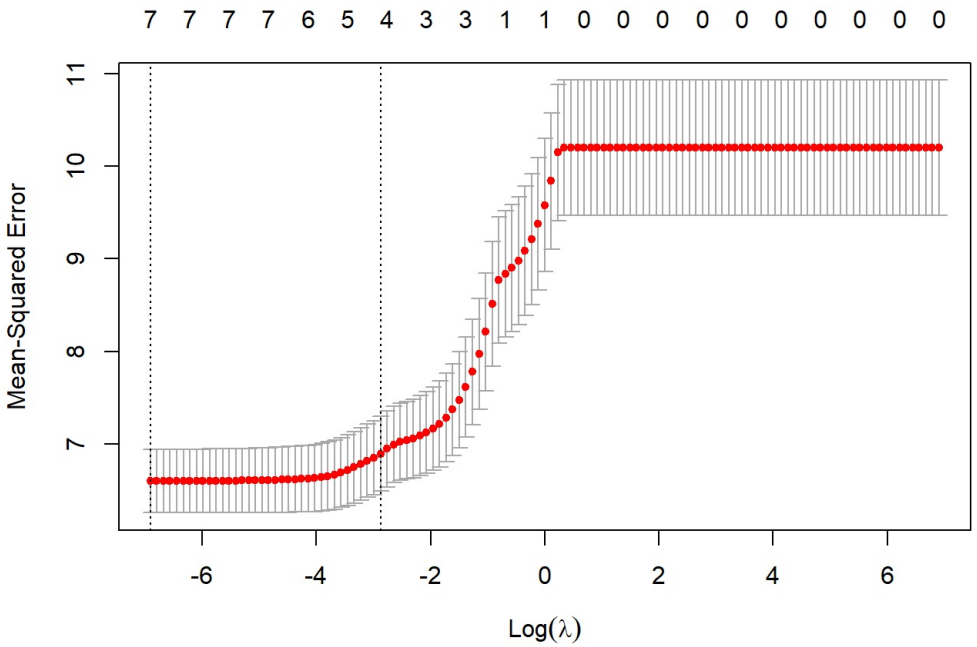
```
## [1] "------------------------------------------------"
## $alpha
## [1] 1
##
## $lambda_min
## [1] 0.001584893
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     7.116289
## Length          1.673474
## Diameter        1.989576
## Height         11.631554
## WholeWeight     8.810806
## ShuckedWeight -18.714842
## VisceraWeight  -9.909287
## ShellWeight    10.644187
##
## $RMSE_Train
## [1] 2.201122
##
## $RMSE_Test
## [1] 2.501699
##
## Time difference of 0.07298112 secs
```

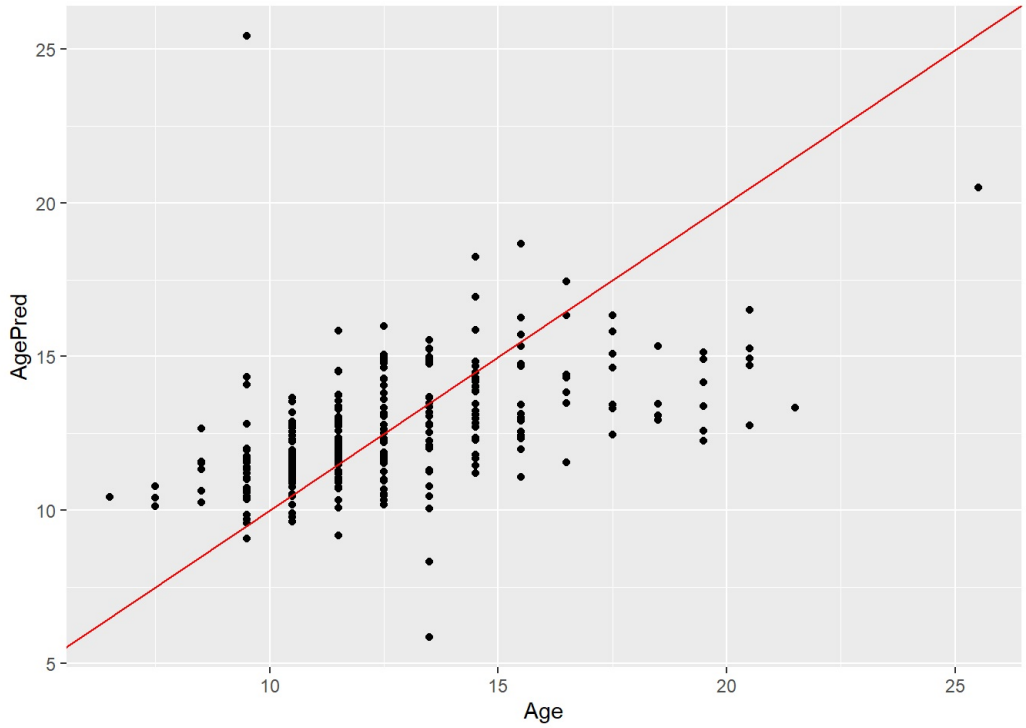





```
perform_cv_glmnet(training.abalone_Age_SexF,
                  testing.abalone_Age_SexF,
                  'Age',
                  1)
```

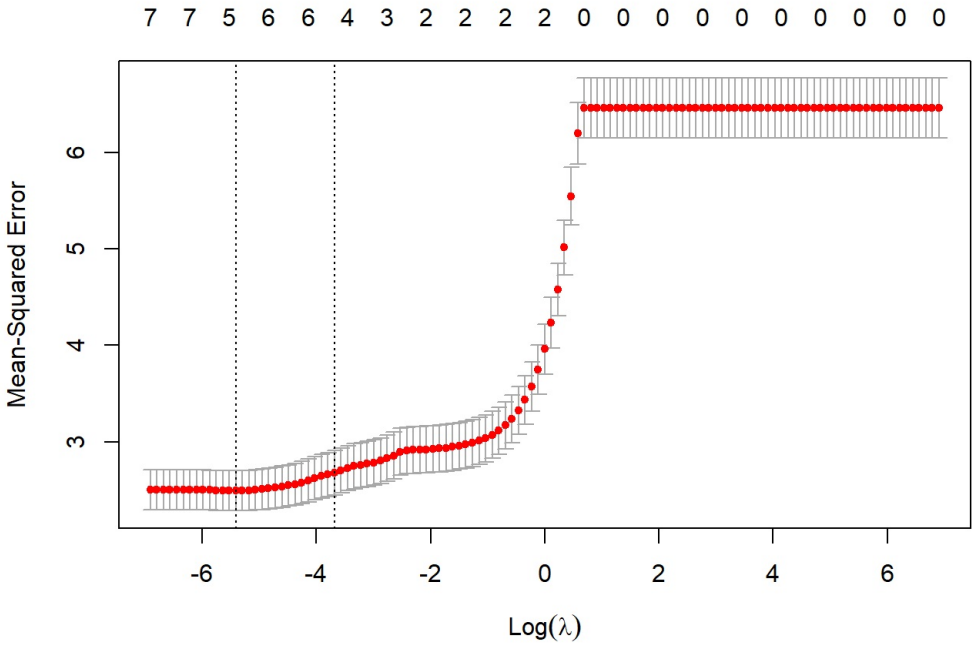
```
## [1] "------------------------------------------------"
## $alpha
## [1] 1
##
## $lambda_min
## [1] 0.001
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     9.051641
## Length         -4.763426
## Diameter        5.505577
## Height         15.632485
## WholeWeight    11.569656
## ShuckedWeight -22.634452
## VisceraWeight  -9.679399
## ShellWeight     5.212572
##
## $RMSE_Train
## [1] 2.54103
##
## $RMSE_Test
## [1] 2.450095
##
## Time difference of 0.05871606 secs
```

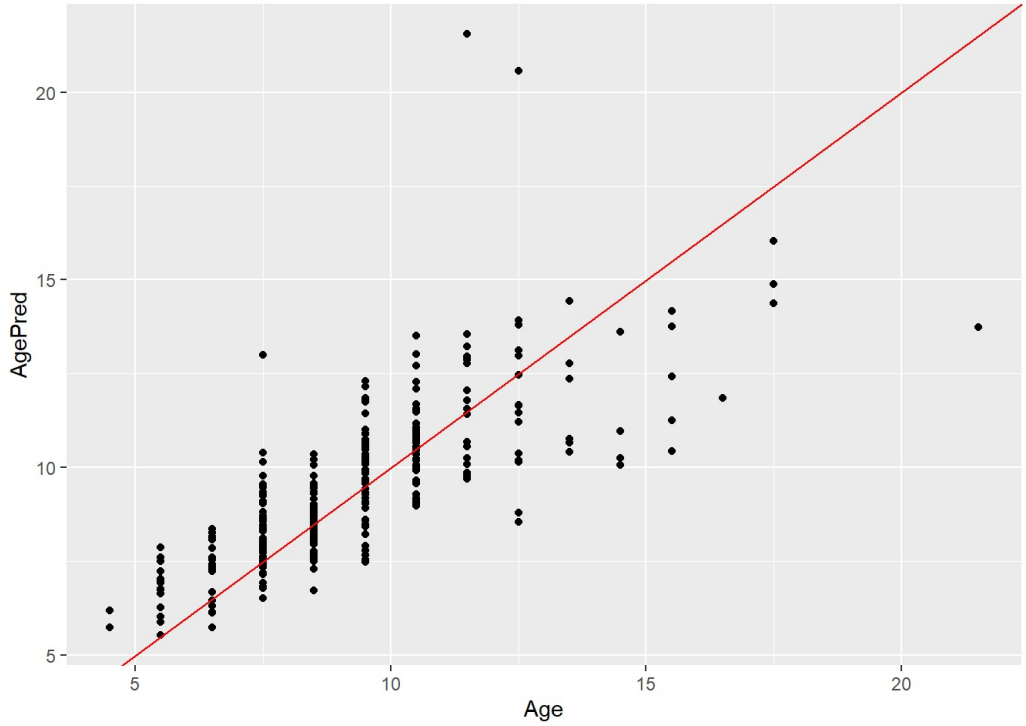





```
perform_cv_glmnet(training.abalone_Age_SexI,
                  testing.abalone_Age_SexI,
                  'Age',
                  1)
```

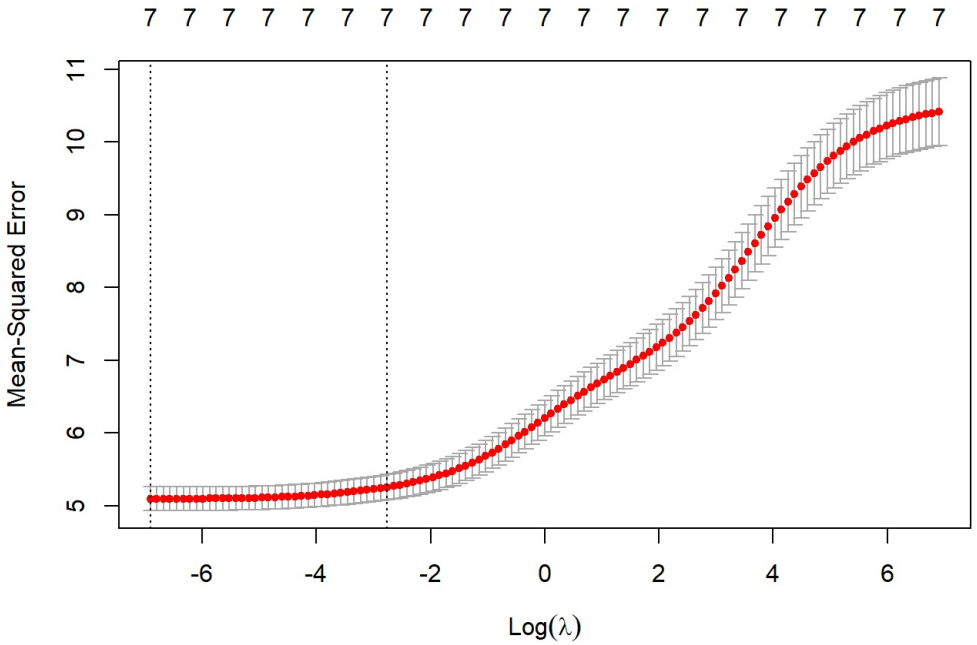
```
## [1] "-------------------------------------------------"
## $alpha
## [1] 1
##
## $lambda_min
## [1] 0.004466836
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     4.550796
## Length          .
## Diameter        1.945700
## Height         27.682372
## WholeWeight    20.382449
## ShuckedWeight -30.360268
## VisceraWeight -19.195895
## ShellWeight     .
##
## $RMSE_Train
## [1] 1.562927
##
## $RMSE_Test
## [1] 1.633043
##
## Time difference of 0.07989192 secs
```

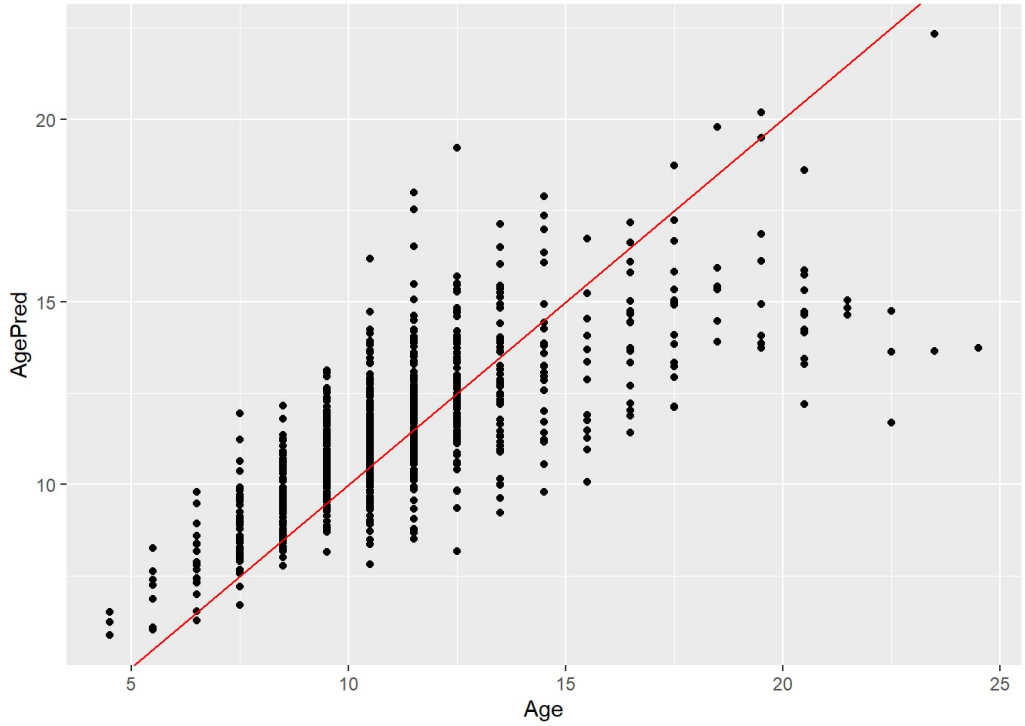





```
# compare same alpha (1) with different datase
set.seed(2024)
perform_cv_glmnet(training.abalone_Age_NoSex,
                  testing.abalone_Age_NoSex,
                  'Age',
                  0)
```

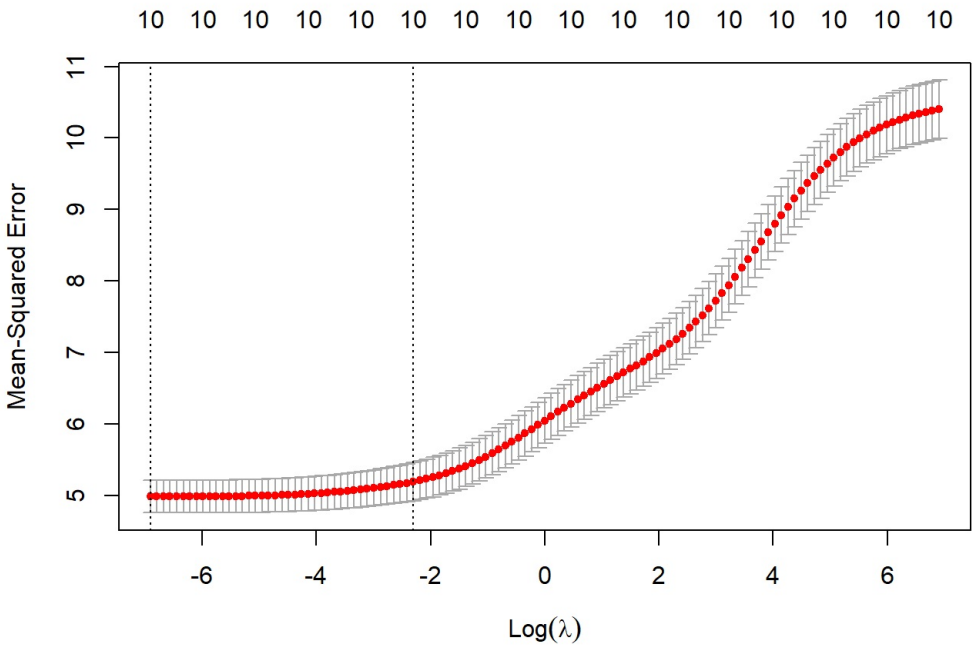
```
## [1] "------------------------------------------------"
## $alpha
## [1] 0
##
## $lambda_min
## [1] 0.001
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     4.441658
## Length         -1.465628
## Diameter       13.871648
## Height         10.350708
## WholeWeight     9.261584
## ShuckedWeight -20.412027
## VisceraWeight  -9.855173
## ShellWeight     8.929117
##
## $RMSE_Train
## [1] 2.227943
##
## $RMSE_Test
## [1] 2.134162
##
## Time difference of 0.238126 secs
```

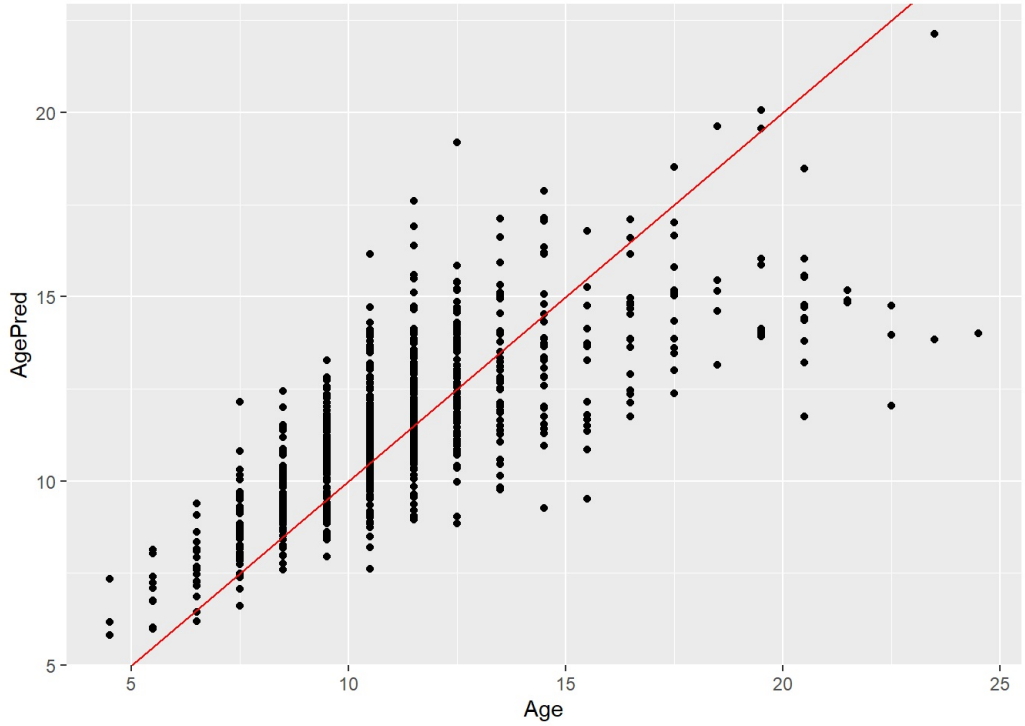





```
perform_cv_glmnet(training.abalone_Age_DummySex,
                  testing.abalone_Age_DummySex,
                  'Age',
                  0)
```

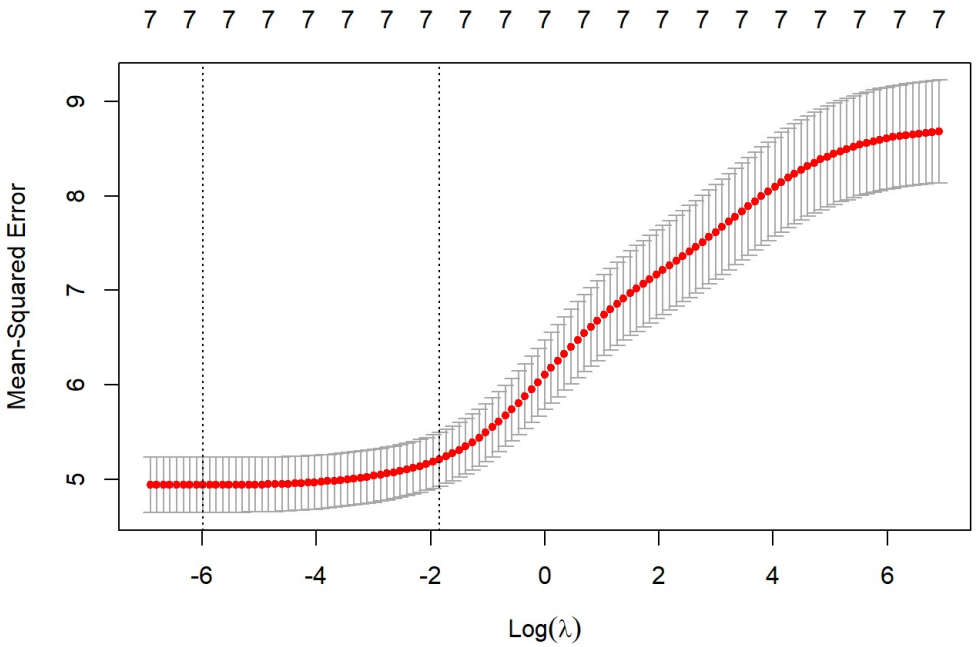
```
## [1] "------------------------------------------------"
## $alpha
## [1] 0
##
## $lambda_min
## [1] 0.001
##
## $coef_best
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)     5.08526636
## Length          0.05449177
## Diameter       11.13020749
## Height          9.35170382
## WholeWeight     9.04690073
## ShuckedWeight -20.07003952
## VisceraWeight -10.68624667
## ShellWeight     8.99813769
## SexF            0.25921217
## SexI           -0.59468808
## SexM            0.31392898
##
## $RMSE_Train
## [1] 2.202636
##
## $RMSE_Test
## [1] 2.112624
##
## Time difference of 0.1331129 secs
```

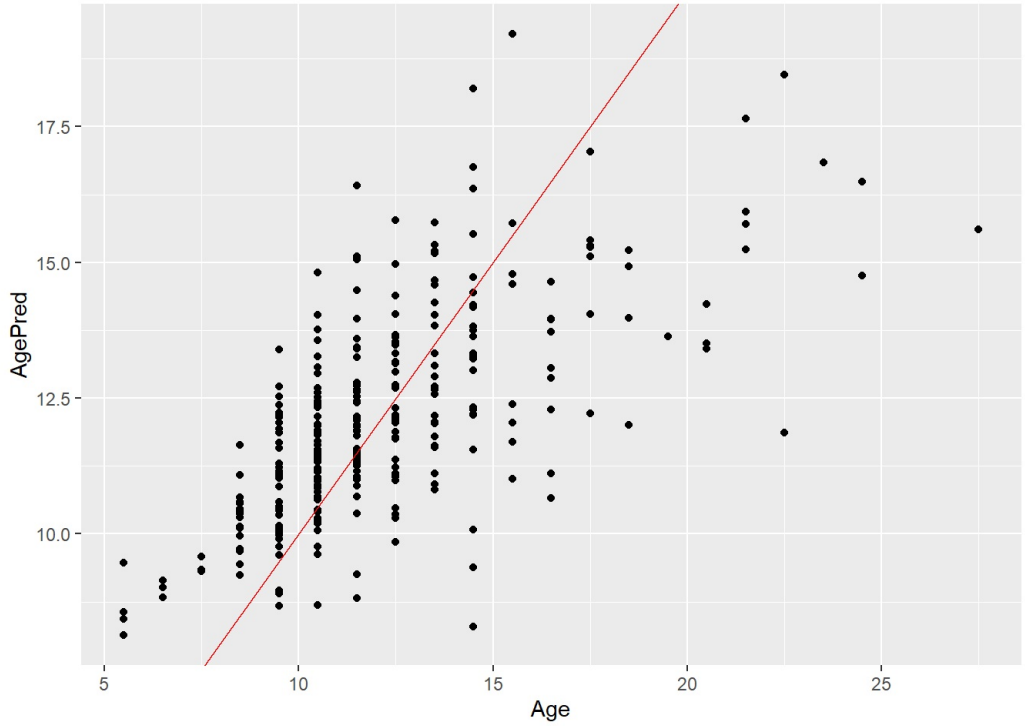





```
perform_cv_glmnet(training.abalone_Age_SexM,
                  testing.abalone_Age_SexM,
                  'Age',
                  0)
```

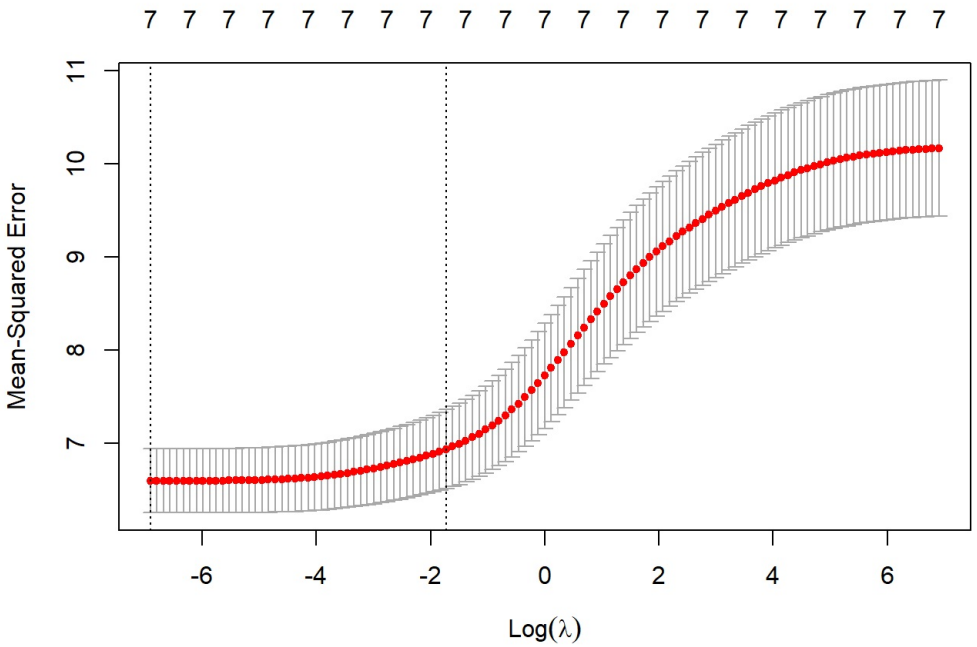
```
## [1] "------------------------------------------------"
## $alpha
## [1] 0
##
## $lambda_min
## [1] 0.002511886
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     7.082048
## Length          1.744789
## Diameter        1.993075
## Height         11.682296
## WholeWeight     8.335884
## ShuckedWeight -18.220608
## VisceraWeight  -9.558179
## ShellWeight    11.232497
##
## $RMSE_Train
## [1] 2.201466
##
## $RMSE_Test
## [1] 2.502254
##
## Time difference of 0.0692358 secs
```

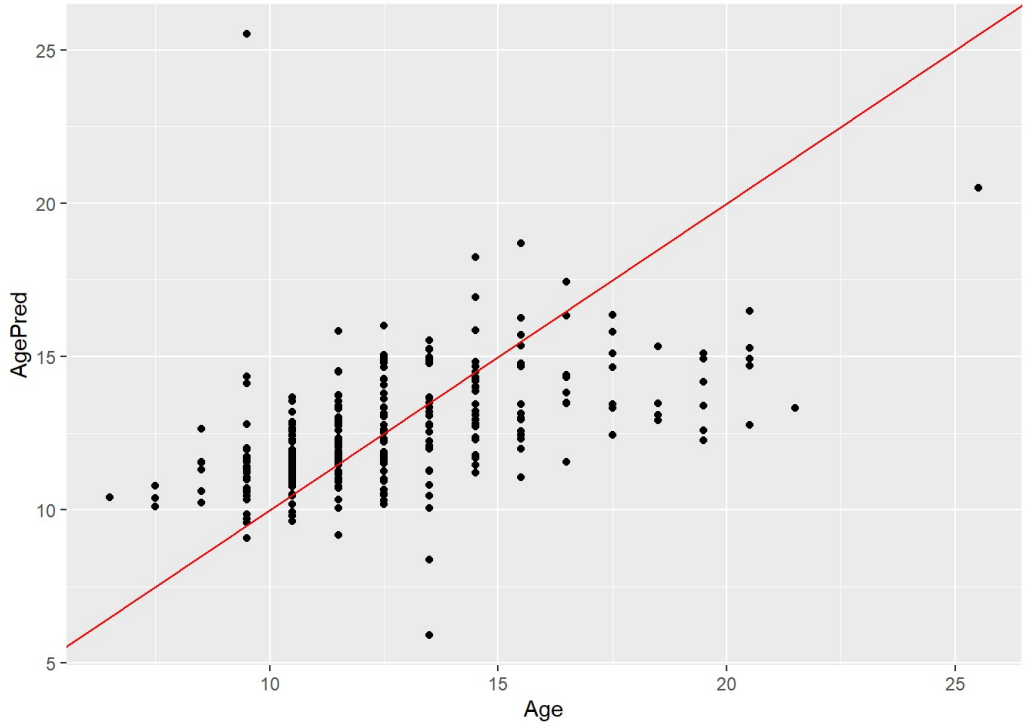





```
perform_cv_glmnet(training.abalone_Age_SexF,
                  testing.abalone_Age_SexF,
                  'Age',
                  0)
```

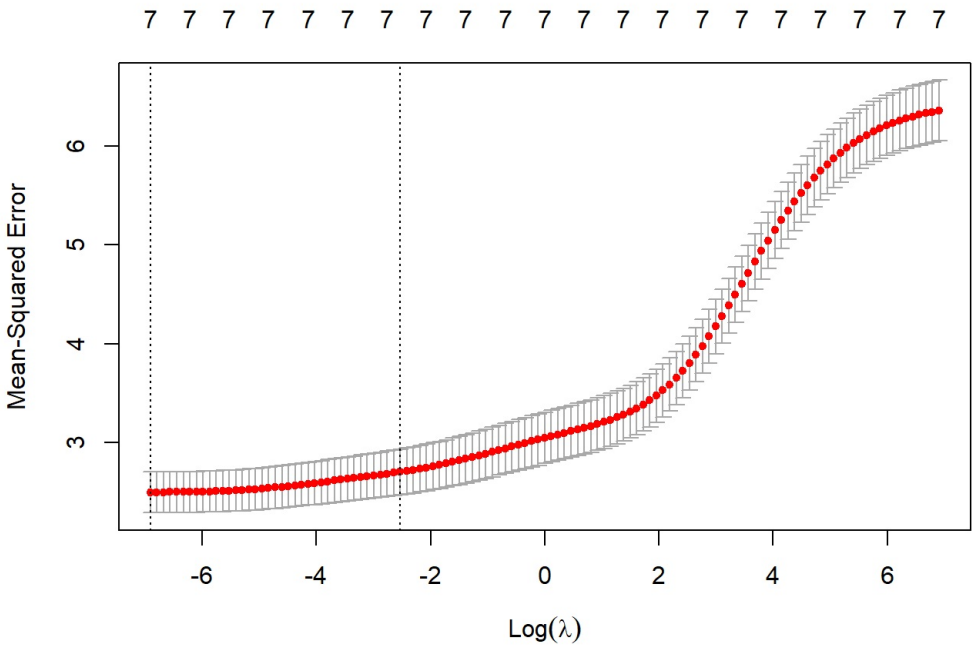
```
## [1] "------------------------------------------------"
## $alpha
## [1] 0
##
## $lambda_min
## [1] 0.001
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     9.056941
## Length         -5.103387
## Diameter        5.902661
## Height         15.706002
## WholeWeight    11.435998
## ShuckedWeight -22.482097
## VisceraWeight  -9.599481
## ShellWeight     5.388076
##
## $RMSE_Train
## [1] 2.541056
##
## $RMSE_Test
## [1] 2.449849
##
## Time difference of 0.06970596 secs
```

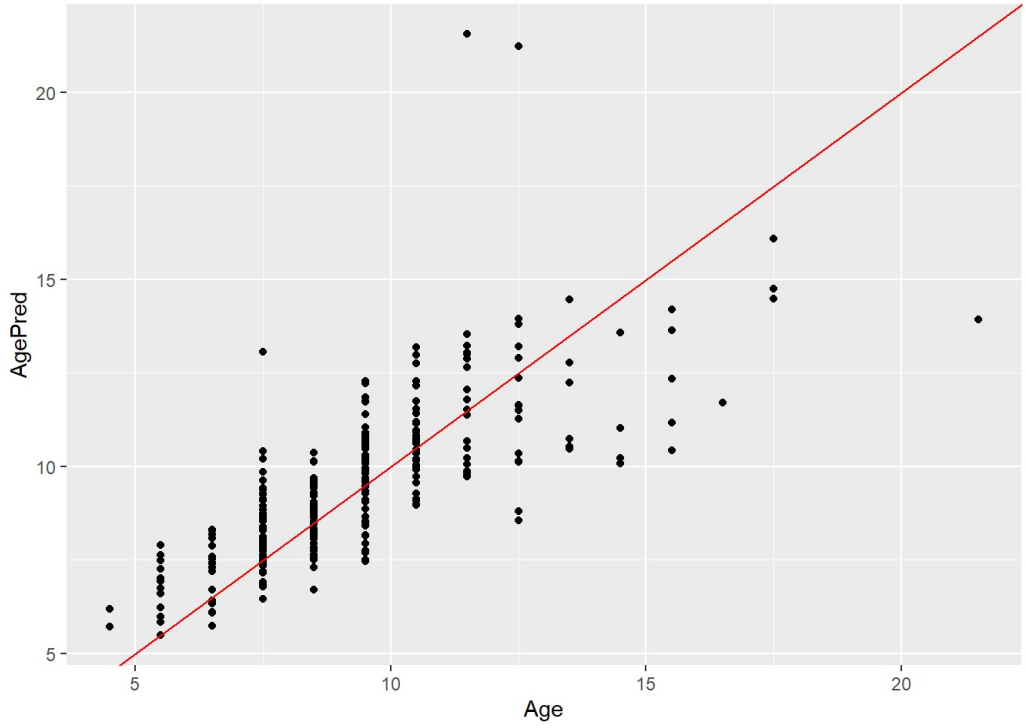





```
perform_cv_glmnet(training.abalone_Age_SexI,
                  testing.abalone_Age_SexI,
                  'Age',
                  0)
```

```
## [1] "------------------------------------------------"
## $alpha
## [1] 0
##
## $lambda_min
## [1] 0.001
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     4.486040
## Length         -1.366678
## Diameter        4.112572
## Height         27.849361
## WholeWeight    23.161552
## ShuckedWeight -33.250717
## VisceraWeight -22.791246
## ShellWeight    -3.107445
##
## $RMSE_Train
## [1] 1.55983
##
## $RMSE_Test
## [1] 1.646548
##
## Time difference of 0.09039497 secs
```





Random Forest

```
set.seed(2024)
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## 载入程辑包：'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
rf = randomForest(Age ~ ., data = training.abalone_Age)
rf
```

```
##
## Call:
##  randomForest(formula = Age ~ ., data = training.abalone_Age)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 4.746797
##                     % Var explained: 54.99
```

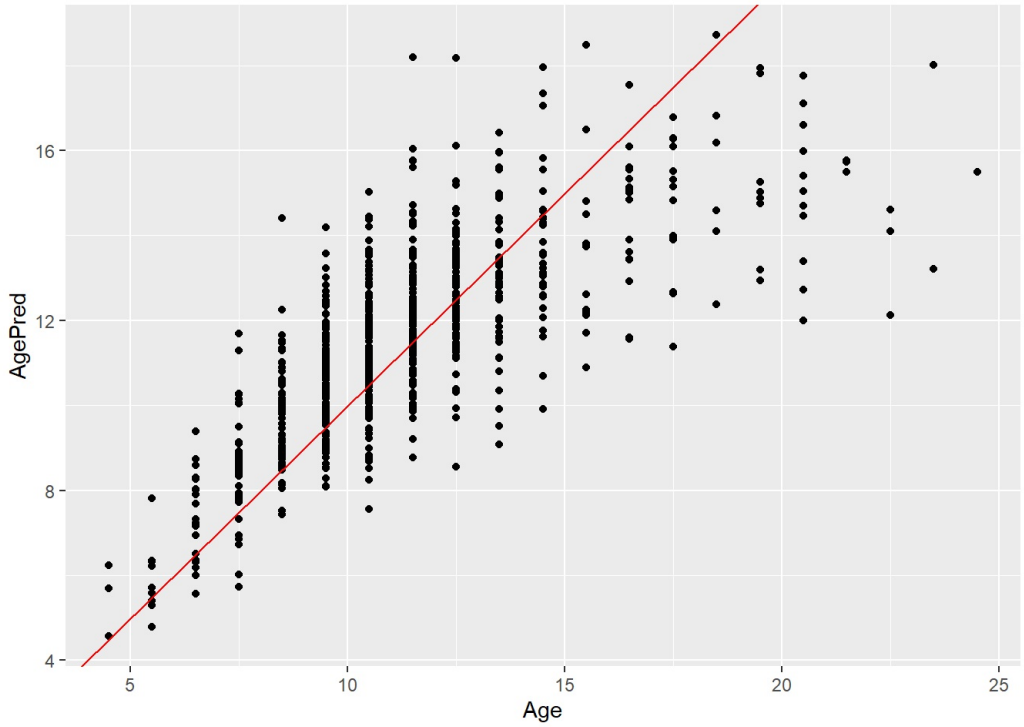
```
library(caret)
```

```
## 载入需要的程辑包：lattice
```

```
perform_random_forest <- function(Formula, TrainSet, TestSet) {
  start_time <- Sys.time()

  print('-------------------------------------------------')
  # Train Random Forest
  rf = randomForest(Formula, data = TrainSet)
  Train = training.abalone_Age %>% mutate(AgePred = predict(rf, newdata = TrainSet))
  Test = TestSet %>% mutate(AgePred = predict(rf, newdata = TestSet))
  TrainError = RMSE(Train$Age, Train$AgePred)
  TestError = RMSE(Test$Age, Test$AgePred)
  result = list(rf = rf, TrainError = TrainError, TestError = TestError)
  print(result)
  end_time <- Sys.time()
  duration <- end_time - start_time
  print(duration)
  ggplot(Test) + geom_point(aes(x=Age, y=AgePred)) + geom_abline(intercept = 0,slope = 1, colour = "red")
}
```

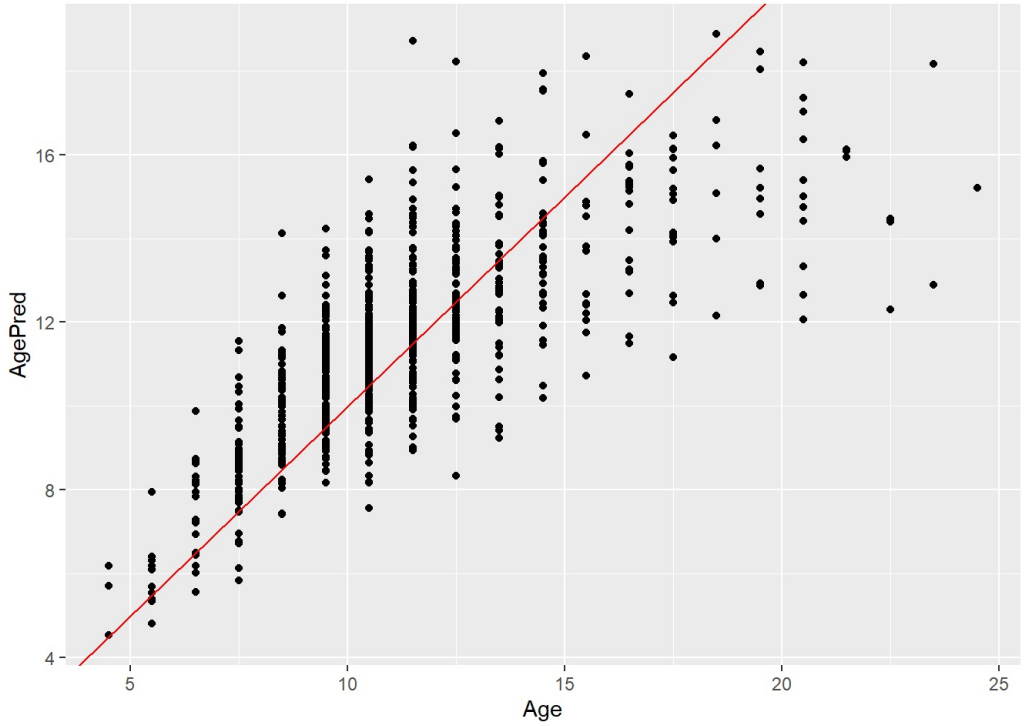
```
perform_random_forest(Age ~ ., training.abalone_Age, testing.abalone_Age)
```

```
## [1] "-------------------------------------------------"
## $rf
##
## Call:
##  randomForest(formula = Formula, data = TrainSet)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 4.70616
##                     % Var explained: 55.38
##
## $TrainError
## [1] 1.028692
##
## $TestError
## [1] 2.055101
##
## Time difference of 4.244303 secs
```



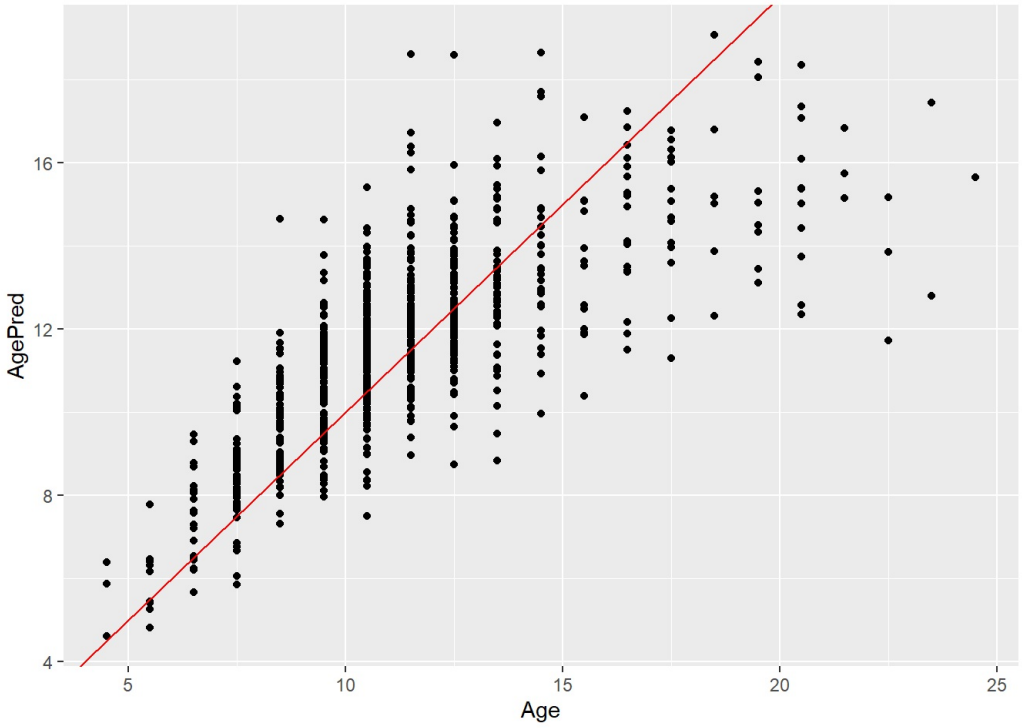

```
perform_random_forest(Age ~ .-Sex, training.abalone_Age, testing.abalone_Age)
```

```
## [1] "--------------------------------------------------"
## $rf
##
## Call:
##  randomForest(formula = Formula, data = TrainSet)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 4.73906
##                     % Var explained: 55.07
##
## $TrainError
## [1] 1.016856
##
## $TestError
## [1] 2.080416
##
## Time difference of 4.626057 secs
```



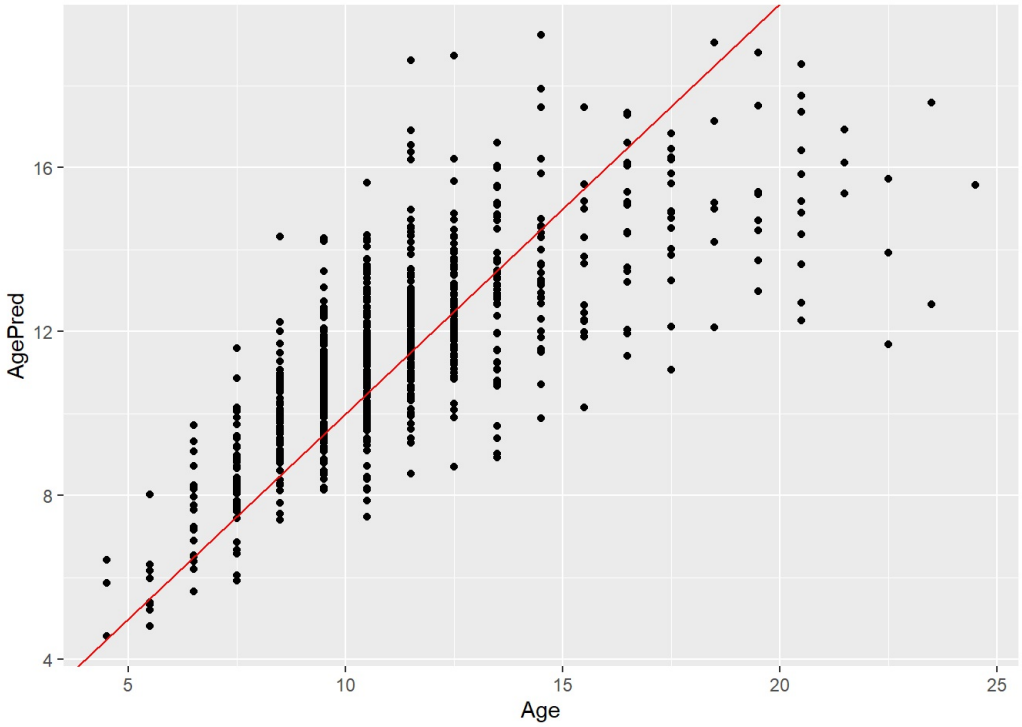
```
perform_random_forest(Age ~ .-Height, training.abalone_Age, testing.abalone_Age)
```

```
## [1] "-------------------------------------------------"
## $rf
##
## Call:
##  randomForest(formula = Formula, data = TrainSet)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 4.724584
##                     % Var explained: 55.2
##
## $TrainError
## [1] 1.02466
##
## $TestError
## [1] 2.06682
##
## Time difference of 4.333578 secs
```




```
perform_random_forest(Age ~ .-Height-Sex, training.abalone_Age, testing.abalone_Age)
```

```
## [1] "-------------------------------------------------"
## $rf
##
## Call:
##  randomForest(formula = Formula, data = TrainSet)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 4.752599
##                     % Var explained: 54.94
##
## $TrainError
## [1] 1.011415
##
## $TestError
## [1] 2.086373
##
## Time difference of 4.466075 secs
```


AgePred
16
12
8
4
5
10
15
20
25
Age