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

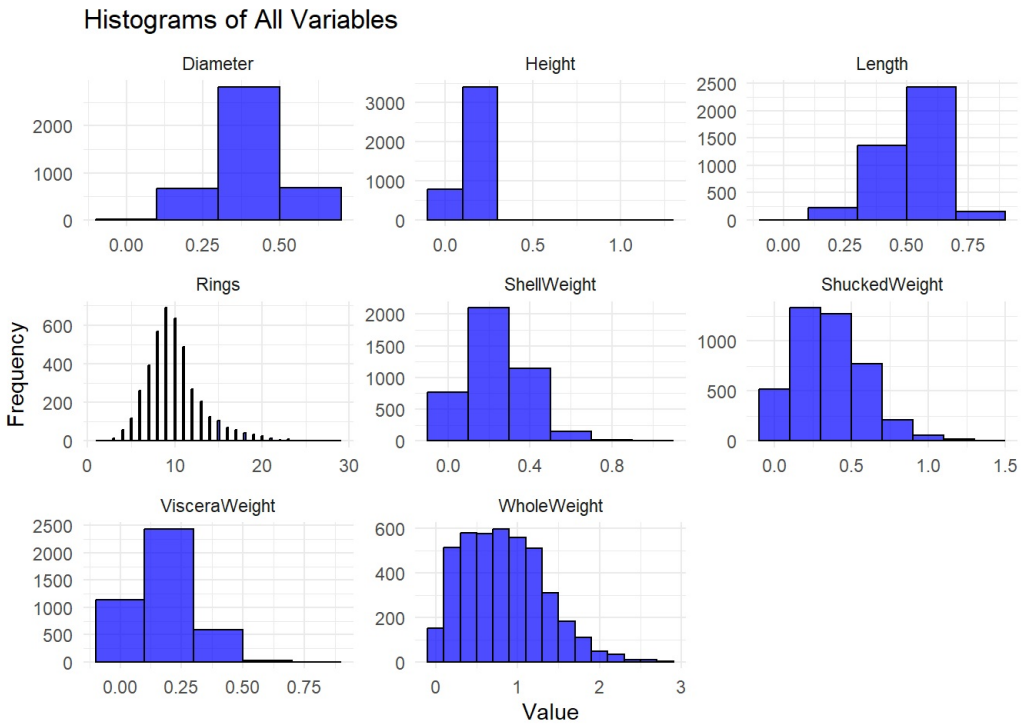
```
# Add a new column 'Age' which is 'Rings' + 1.5
# Add 'Age' column and remove 'Rings' column
abalone_Age = cleaned_abalone %>%
  mutate(Age = Rings + 1.5) %>%
  select(-Rings)
# Convert 'Sex' to dummy variables (one-hot encoding)
# dummySex = model.matrix(~ Sex - 1, data = abalone_Age)
# Bind the 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)
```

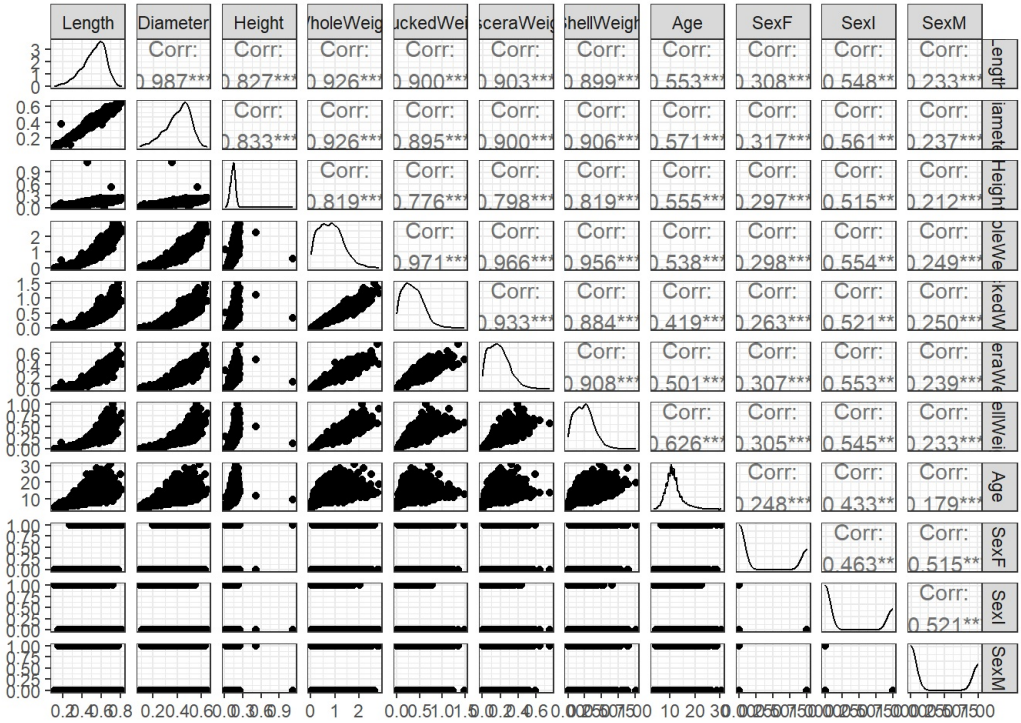
```
##      Sex                Length         Diameter          Height
##  Length:4177        Min.   :0.075   Min.   :0.0550   Min.   :0.0000
##  Class :character   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150
##  Mode  :character   Median :0.545   Median :0.4250   Median :0.1400
##                     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
```

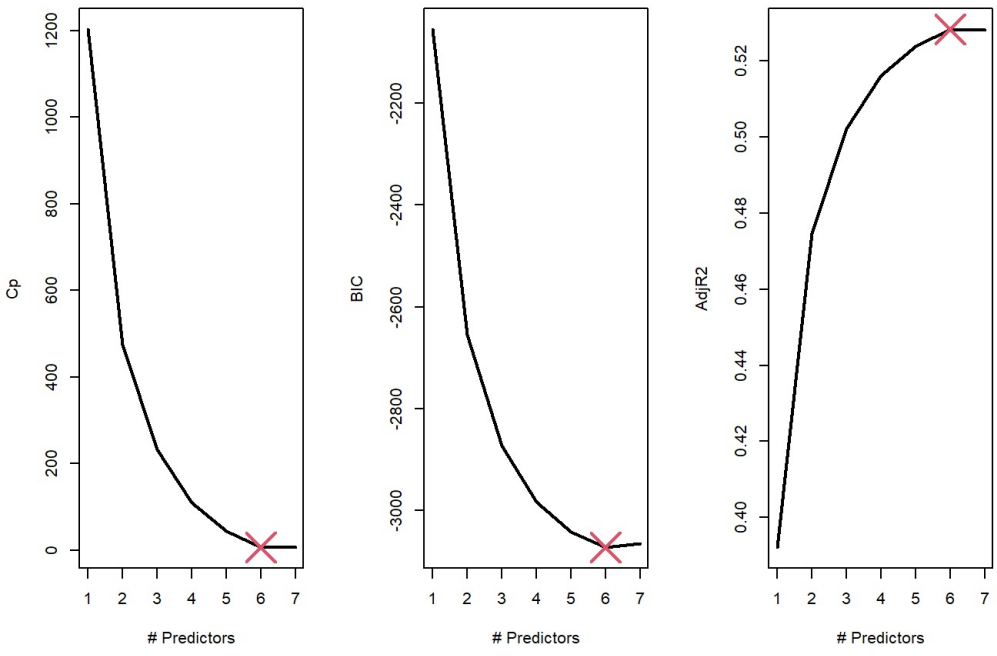
```
summary(best_models.abalone_Age_DummySex)
```

```
## 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'
```

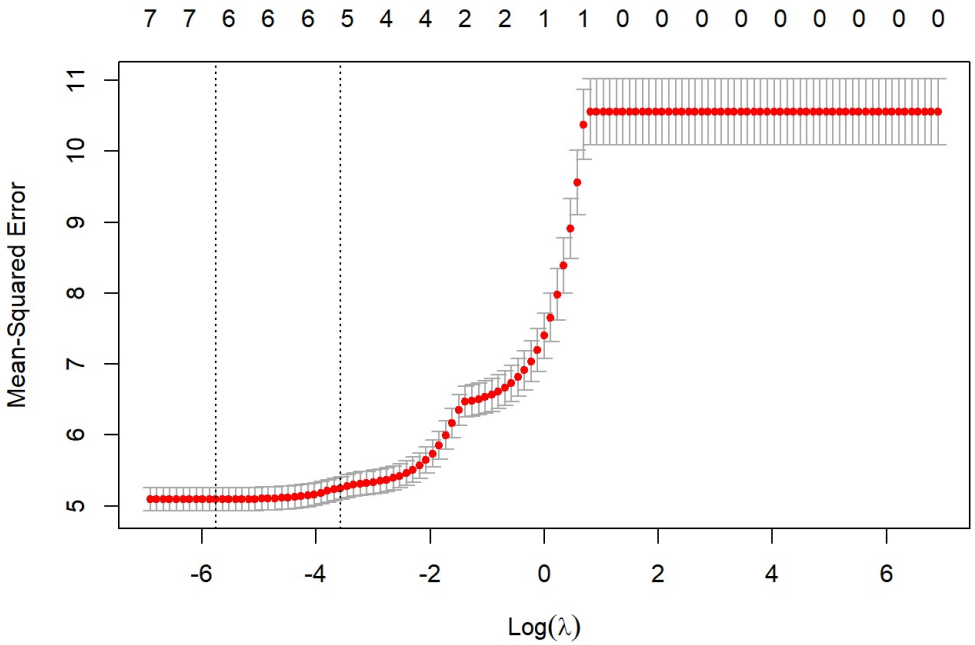
```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

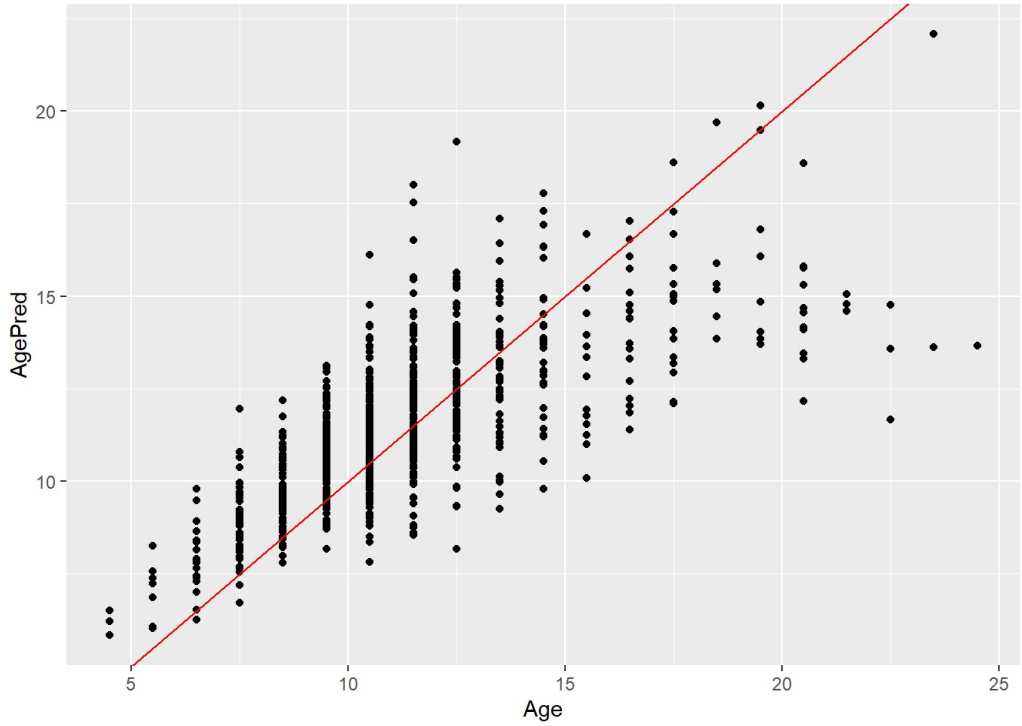
```
## Loaded glmnet 4.1-8
```

```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)
```

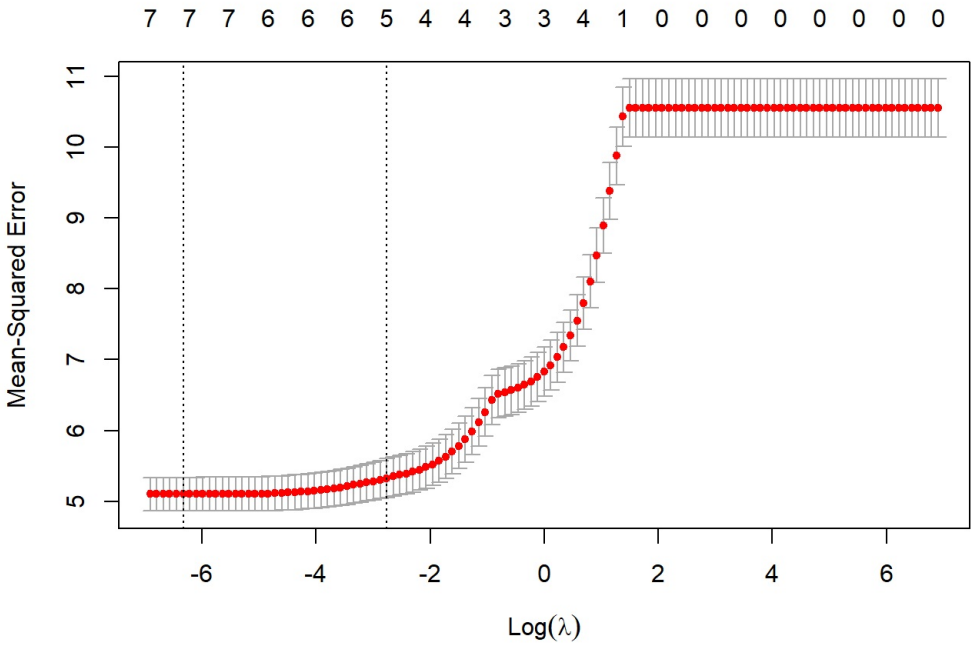
```
## [1] "--------------------------------------------------"
## $alpha
## [1] 1
##
## $lambda_min
## [1] 0.003162278
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     4.402160
## Length          .
## Diameter       12.101448
## Height         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
```

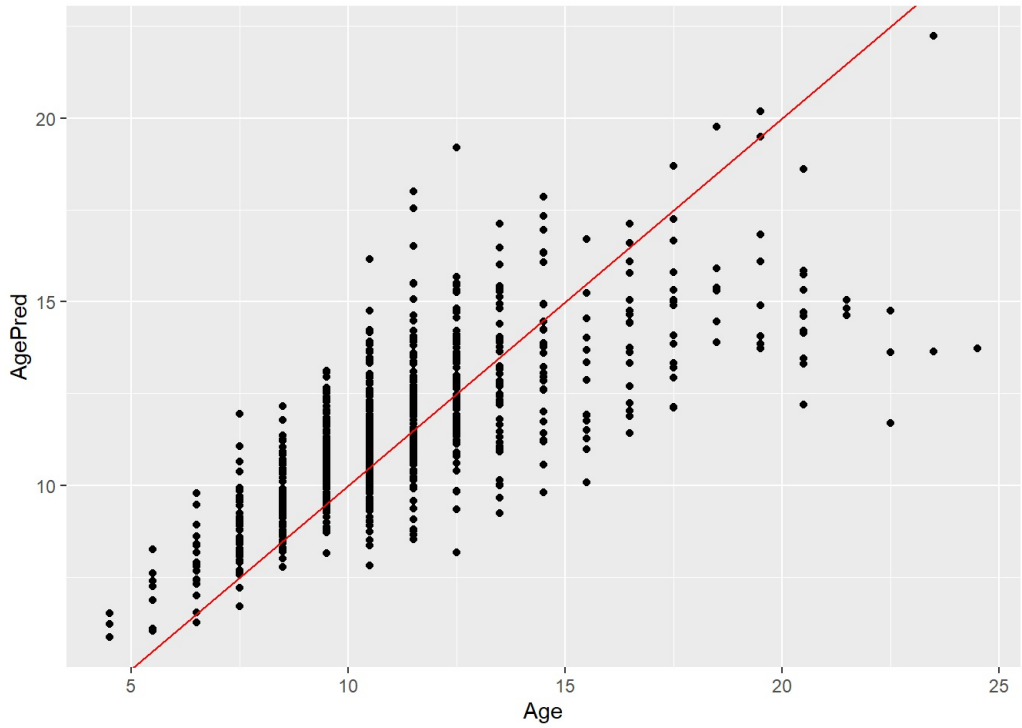





```
perform_cv_glmnet(training.abalone_Age_NoSex,
                  testing.abalone_Age_NoSex,
                  'Age',
                  0.5)
```

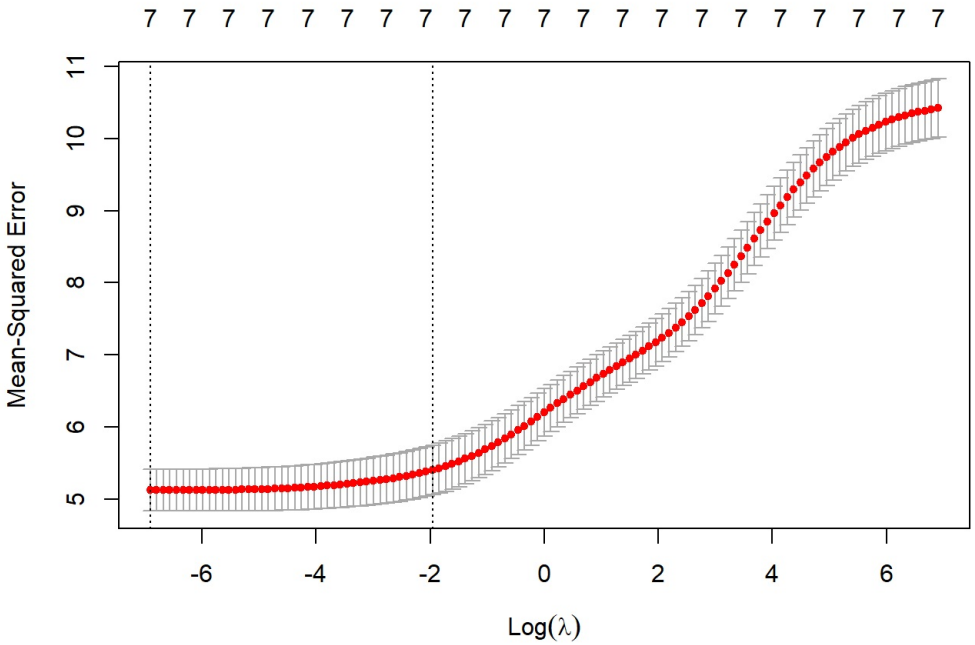
```
## [1] "------------------------------------------------"
## $alpha
## [1] 0.5
##
## $lambda_min
## [1] 0.001778279
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)     4.4246041
## Length         -0.9339564
## Diameter       13.2359107
## Height         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
```

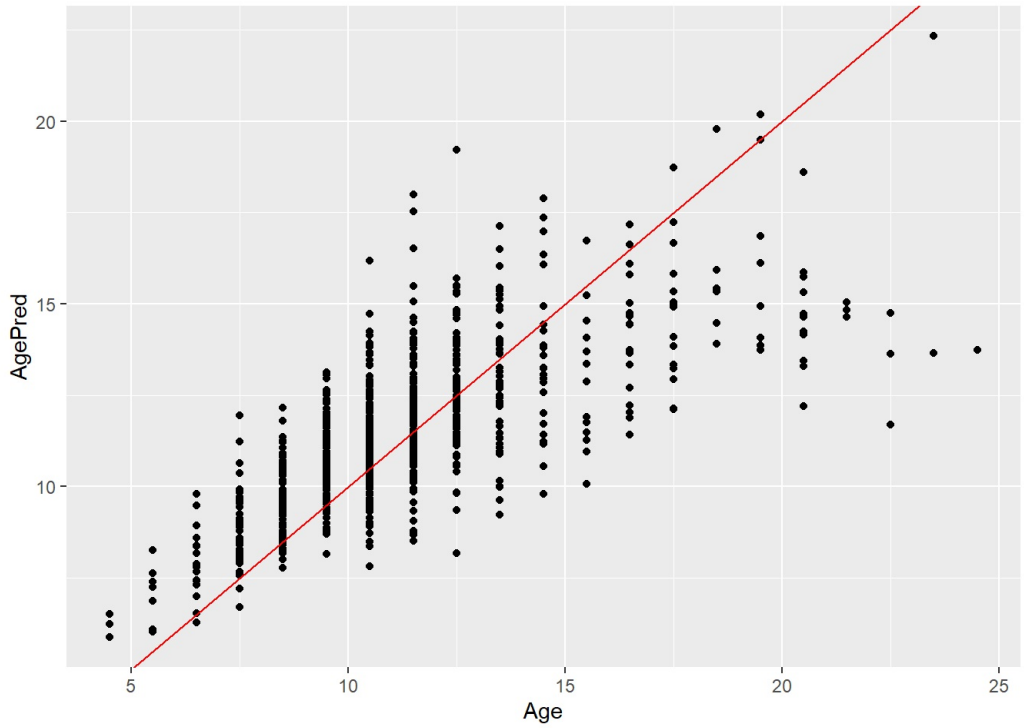





```
perform_cv_glmnet(training.abalone_Age_NoSex,
                  testing.abalone_Age_NoSex,
                  'Age',
                  0)
```

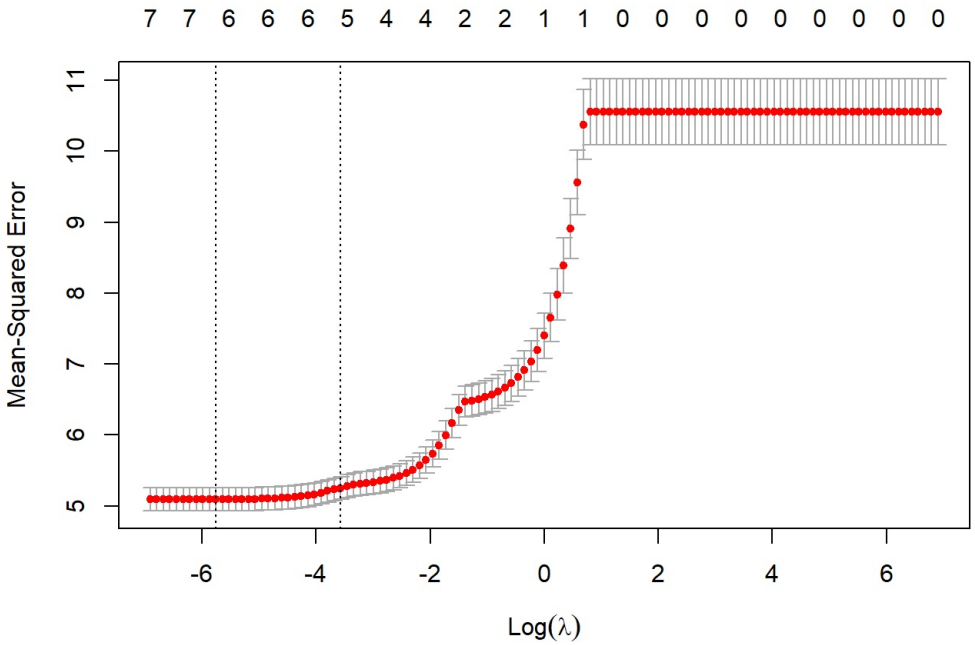
```
## [1] "--------------------------------------------------"
## $alpha
## [1] 0
##
## $lambda_min
## [1] 0.001
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     4.441658
## Length         -1.465628
## Diameter       13.871648
## Height         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
```

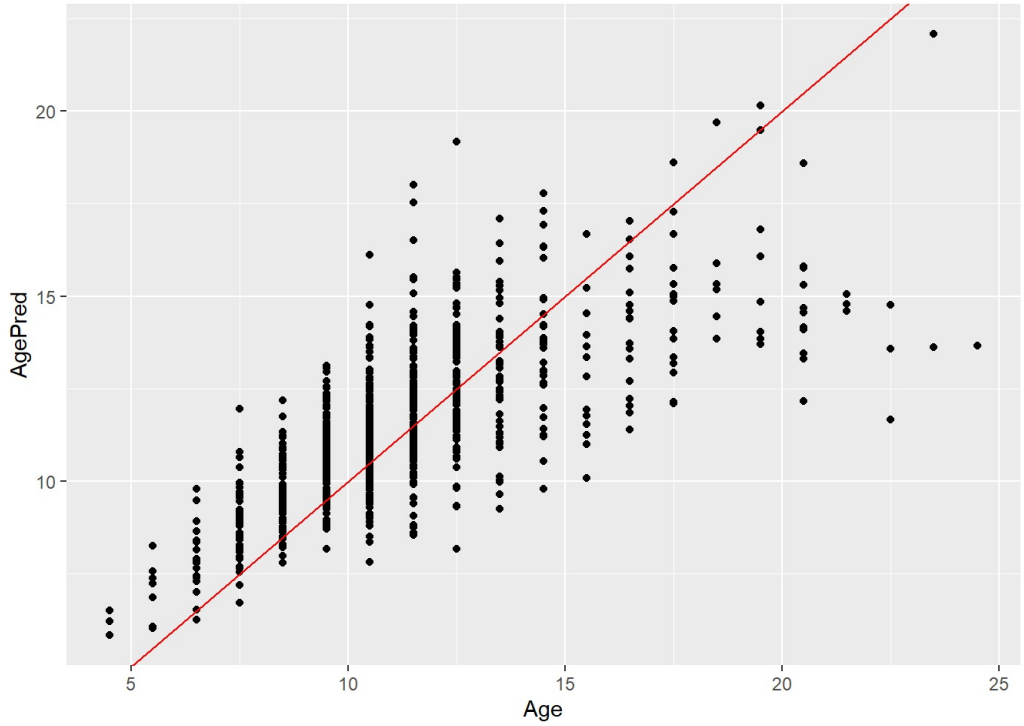





```
# compare same alpha (1) with different datase
set.seed(2024)
perform_cv_glmnet(training.abalone_Age_NoSex,
                  testing.abalone_Age_NoSex,
                  'Age',
                  1)
```

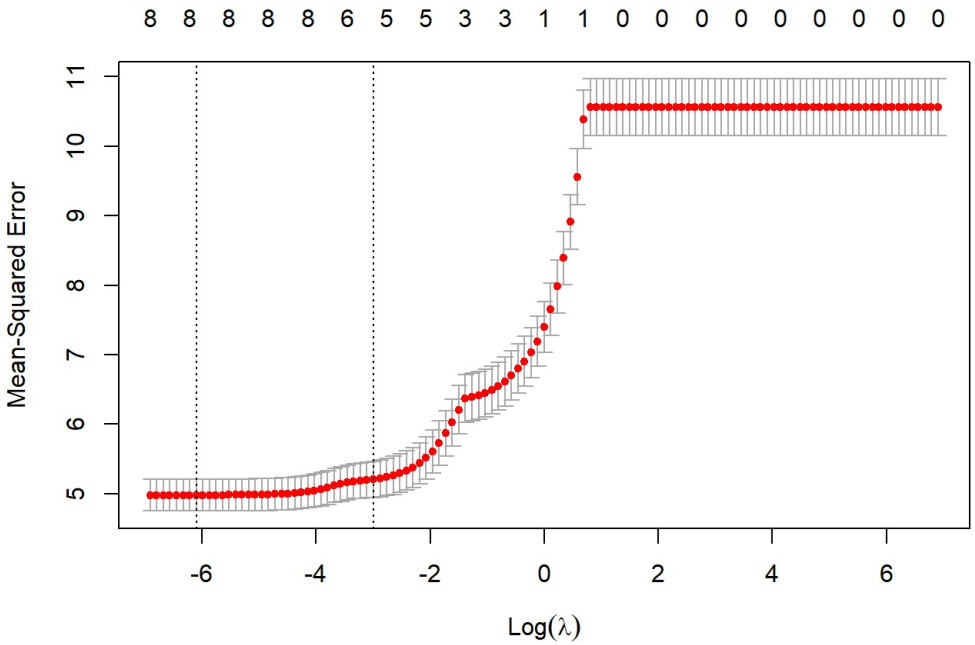
```
## [1] "--------------------------------------------------"
## $alpha
## [1] 1
##
## $lambda_min
## [1] 0.003162278
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     4.402160
## Length          .
## Diameter       12.101448
## Height         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
```

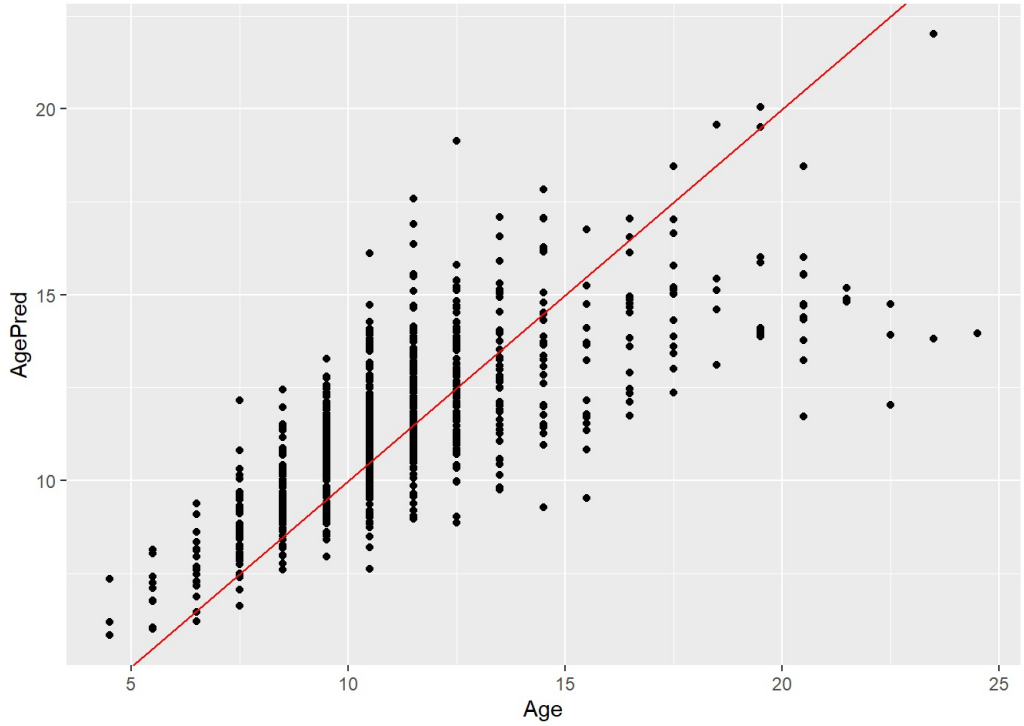





```
perform_cv_glmnet(training.abalone_Age_DummySex,
                  testing.abalone_Age_DummySex,
                  'Age',
                  1)
```

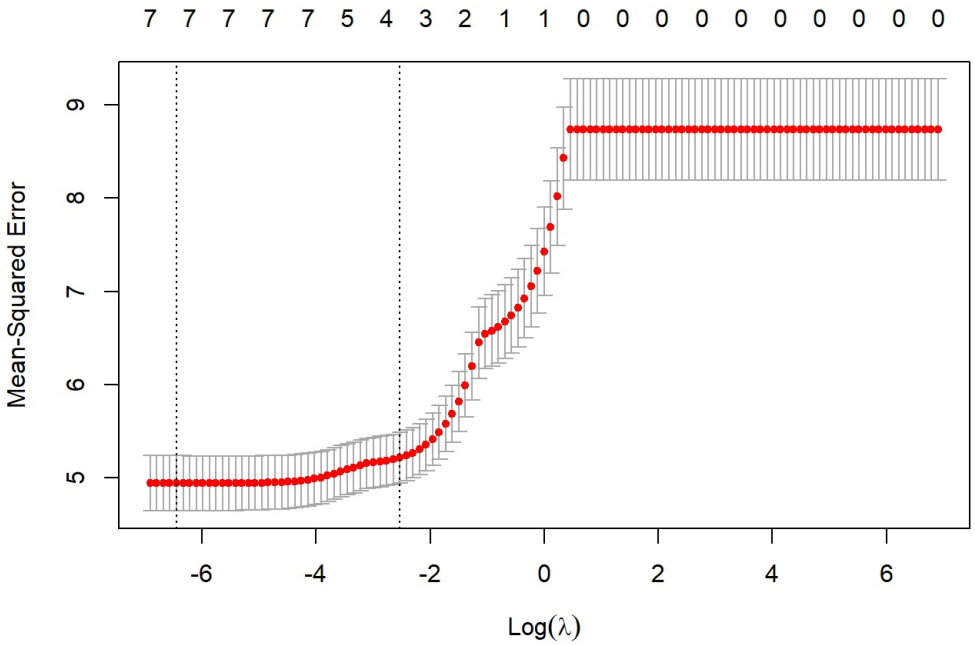
```
## [1] "-------------------------------------------------"
## $alpha
## [1] 1
##
## $lambda_min
## [1] 0.002238721
##
## $coef_best
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)     5.37775119
## Length          .
## Diameter       11.08341866
## Height          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
```

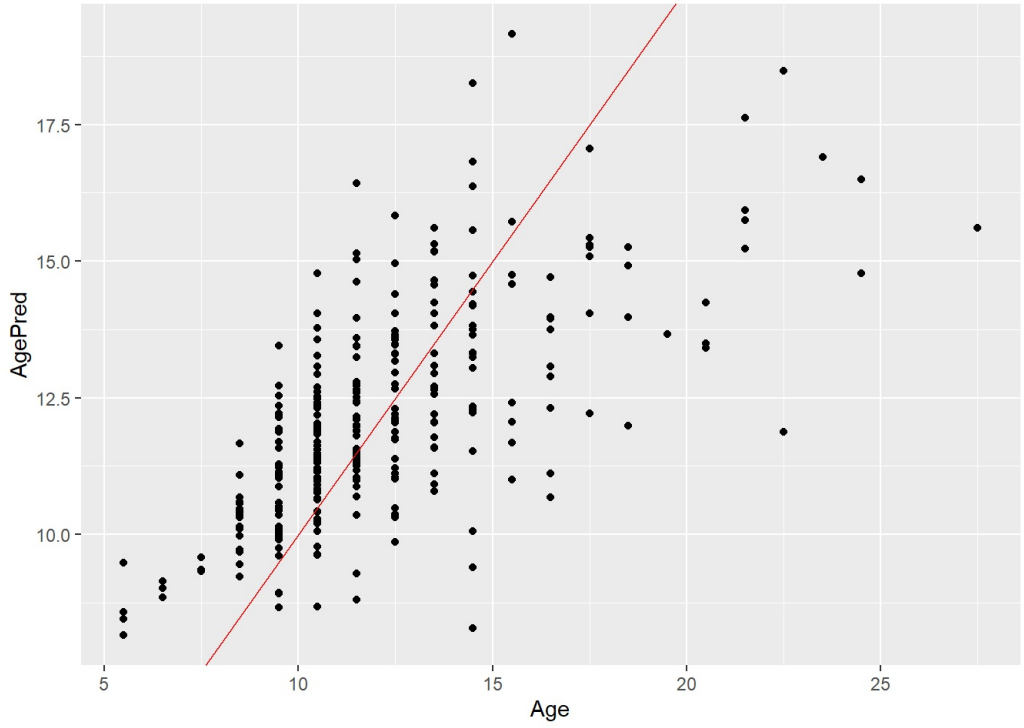





```
perform_cv_glmnet(training.abalone_Age_SexM,
                  testing.abalone_Age_SexM,
                  'Age',
                  1)
```

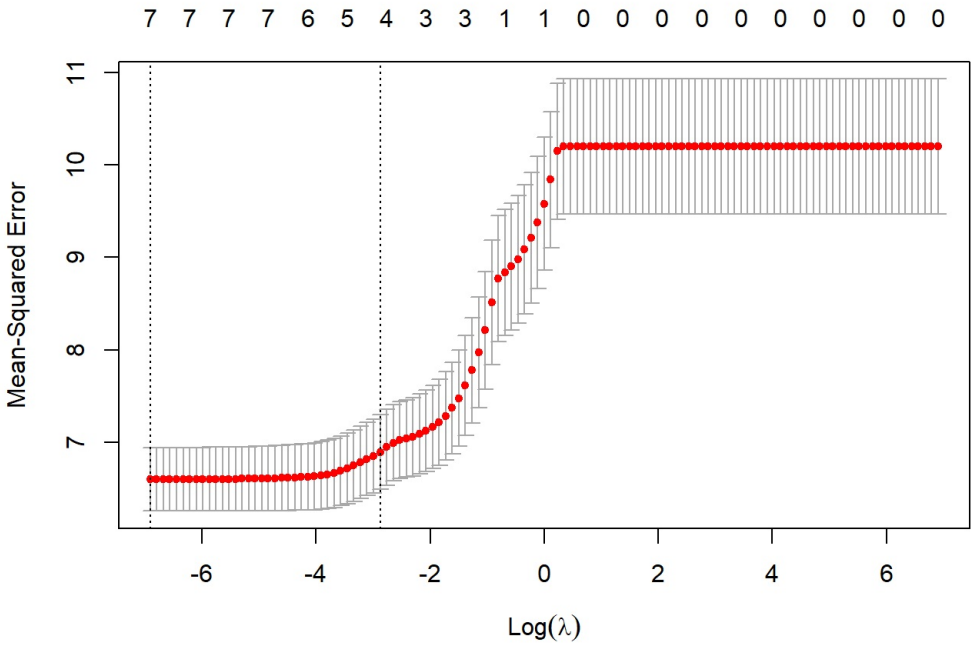
```
## [1] "--------------------------------------------------"
## $alpha
## [1] 1
##
## $lambda_min
## [1] 0.001584893
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     7.116289
## Length          1.673474
## Diameter        1.989576
## Height         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
```

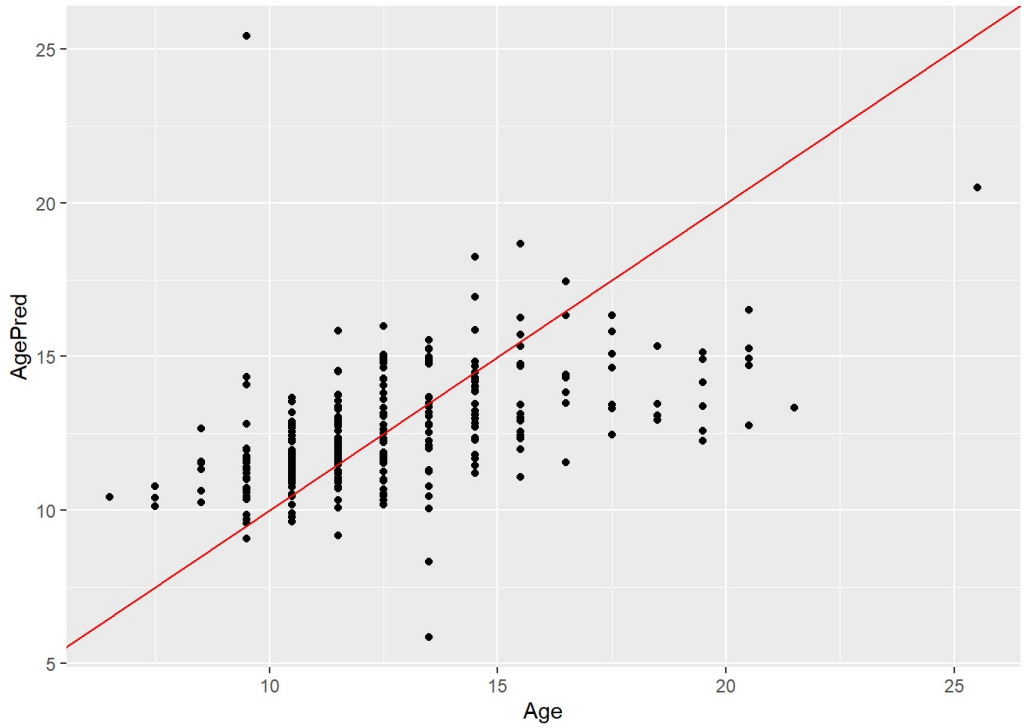





```
perform_cv_glmnet(training.abalone_Age_SexF,
                  testing.abalone_Age_SexF,
                  'Age',
                  1)
```

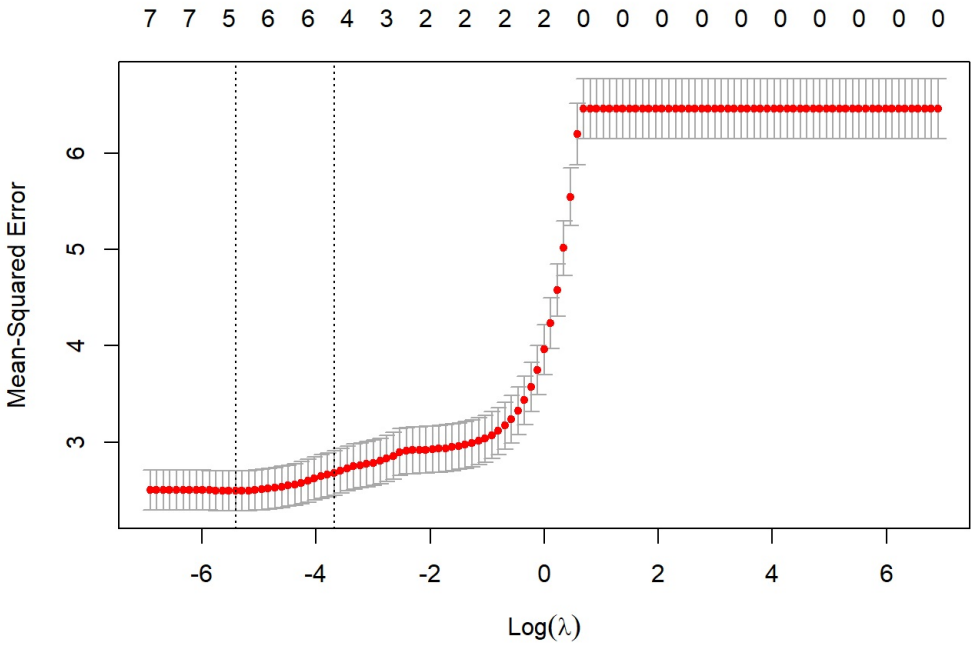
```
## [1] "--------------------------------------------------"
## $alpha
## [1] 1
##
## $lambda_min
## [1] 0.001
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     9.051641
## Length         -4.763426
## Diameter        5.505577
## Height         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
```

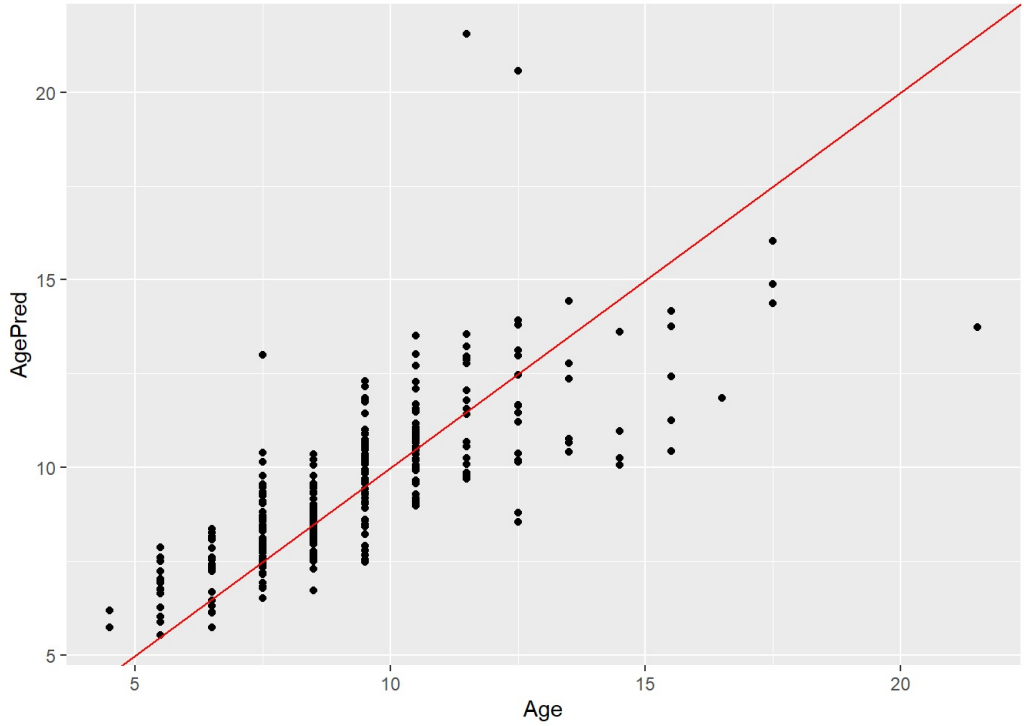





```
perform_cv_glmnet(training.abalone_Age_SexI,
                  testing.abalone_Age_SexI,
                  'Age',
                  1)
```

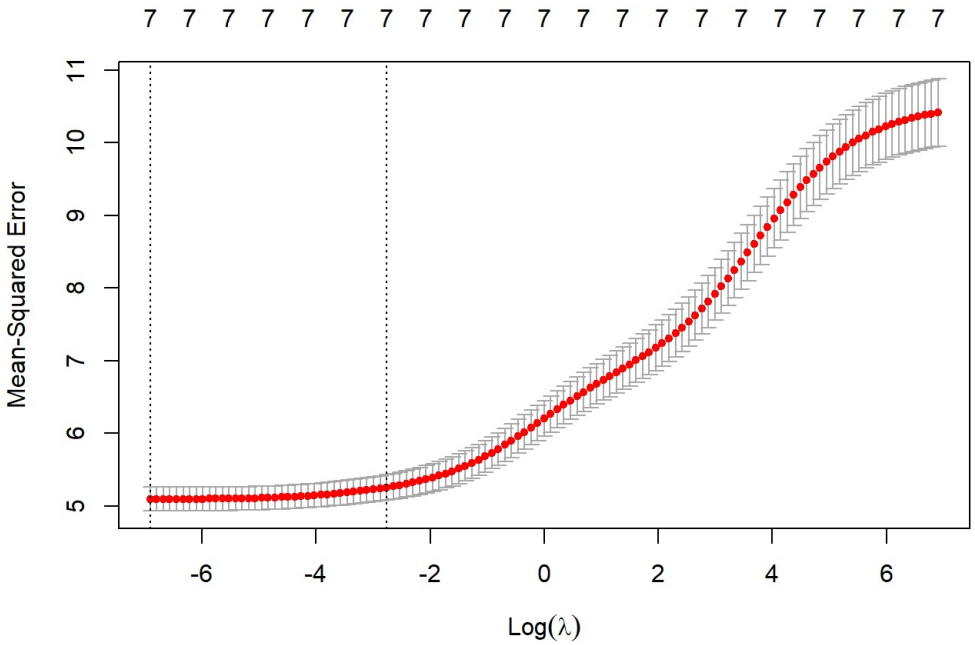
```
## [1] "--------------------------------------------------"
## $alpha
## [1] 1
##
## $lambda_min
## [1] 0.004466836
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     4.550796
## Length          .
## Diameter        1.945700
## Height         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
```

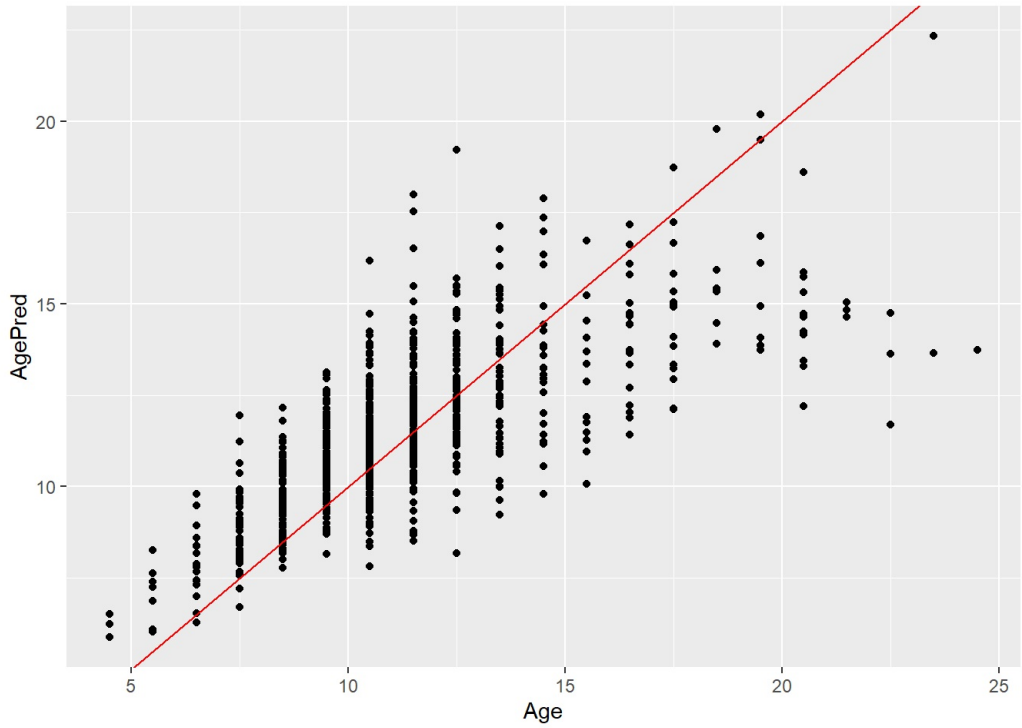





```
# compare same alpha (1) with different datase
set.seed(2024)
perform_cv_glmnet(training.abalone_Age_NoSex,
                  testing.abalone_Age_NoSex,
                  'Age',
                  0)
```

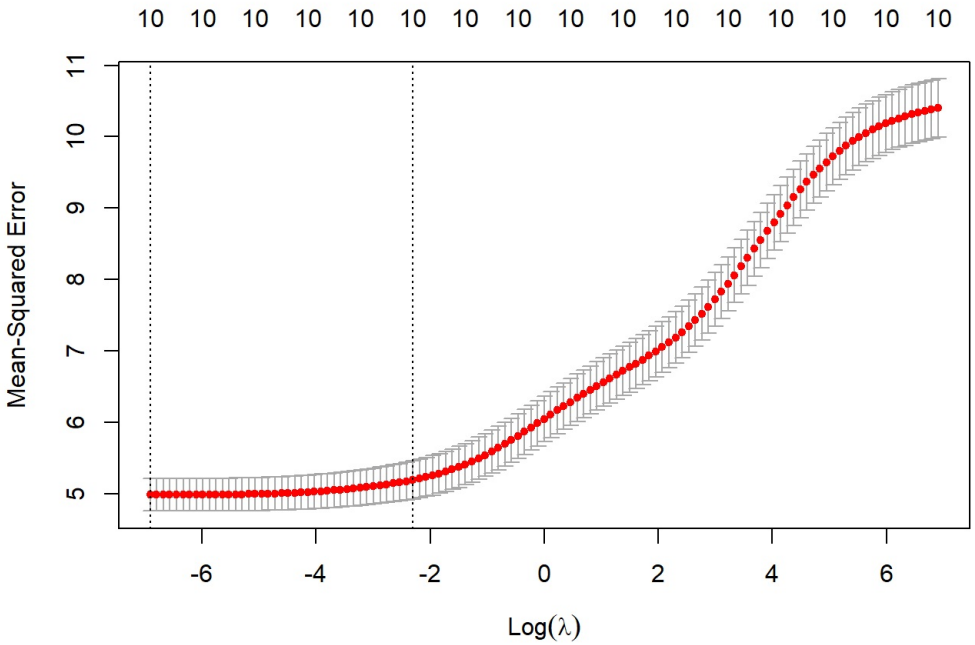
```
## [1] "--------------------------------------------------"
## $alpha
## [1] 0
##
## $lambda_min
## [1] 0.001
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     4.441658
## Length         -1.465628
## Diameter       13.871648
## Height         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
```

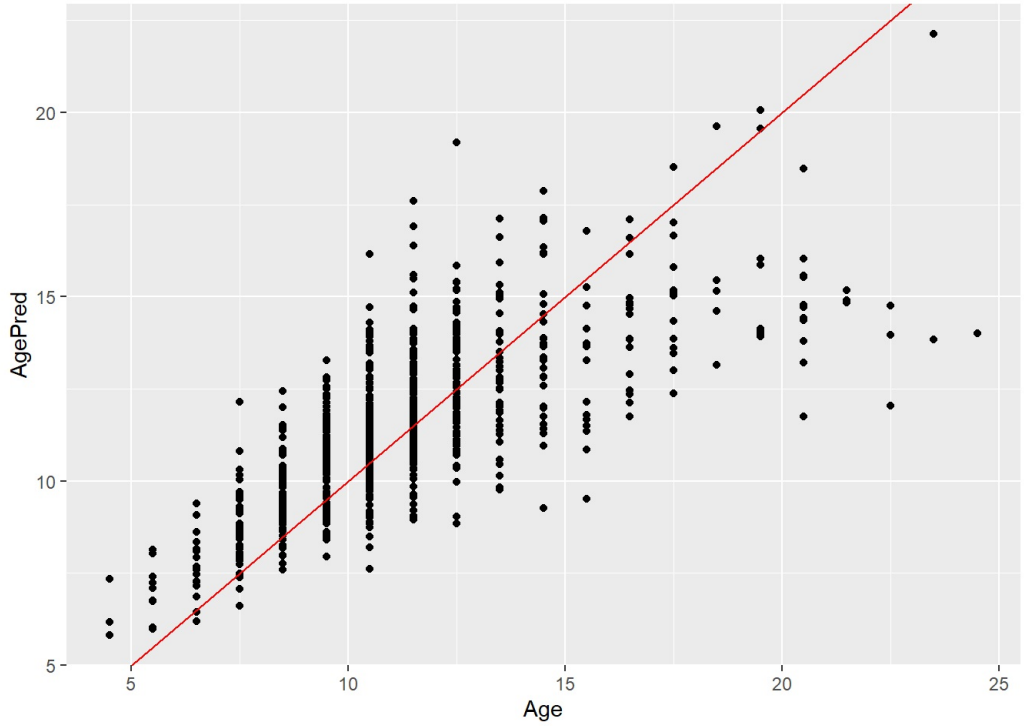





```
perform_cv_glmnet(training.abalone_Age_DummySex,
                  testing.abalone_Age_DummySex,
                  'Age',
                  0)
```

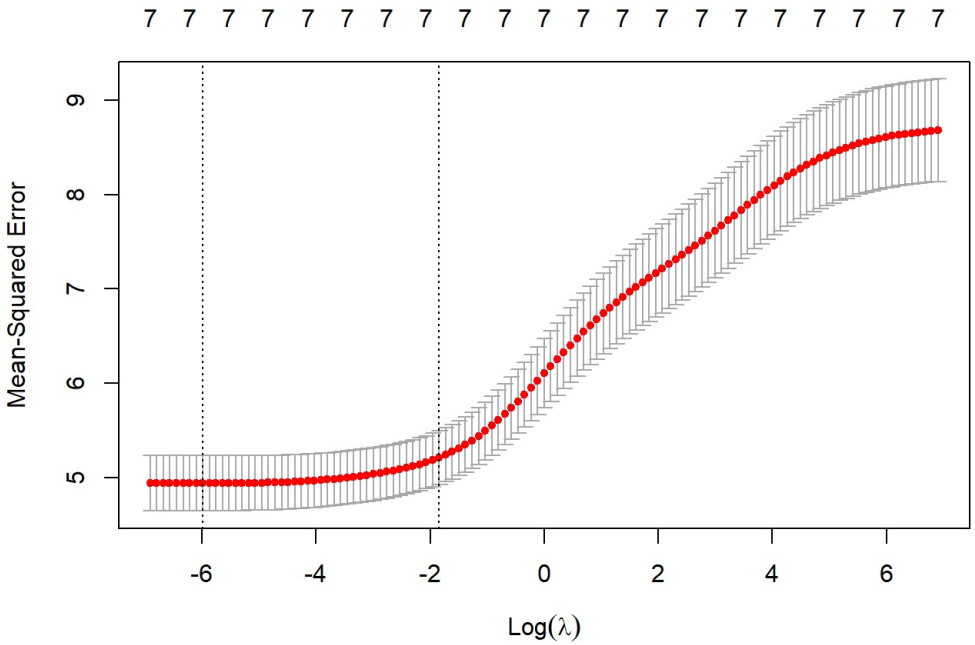
```
## [1] "-------------------------------------------------"
## $alpha
## [1] 0
##
## $lambda_min
## [1] 0.001
##
## $coef_best
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)     5.08526636
## Length          0.05449177
## Diameter       11.13020749
## Height          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
```

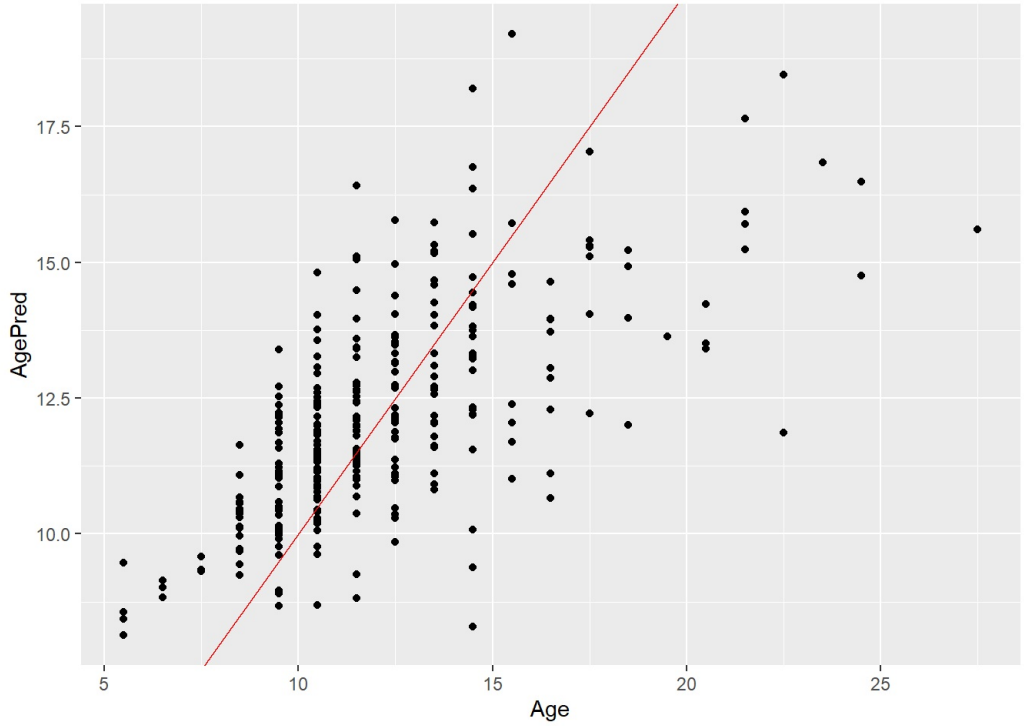





```
perform_cv_glmnet(training.abalone_Age_SexM,
                  testing.abalone_Age_SexM,
                  'Age',
                  0)
```

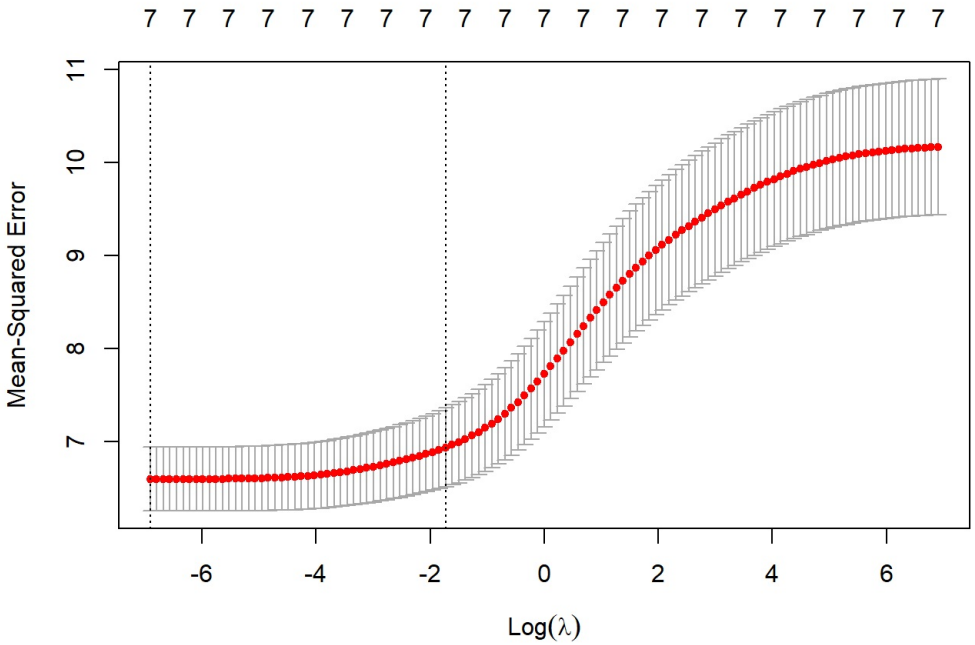
```
## [1] "--------------------------------------------------"
## $alpha
## [1] 0
##
## $lambda_min
## [1] 0.002511886
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     7.082048
## Length          1.744789
## Diameter        1.993075
## Height         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
```

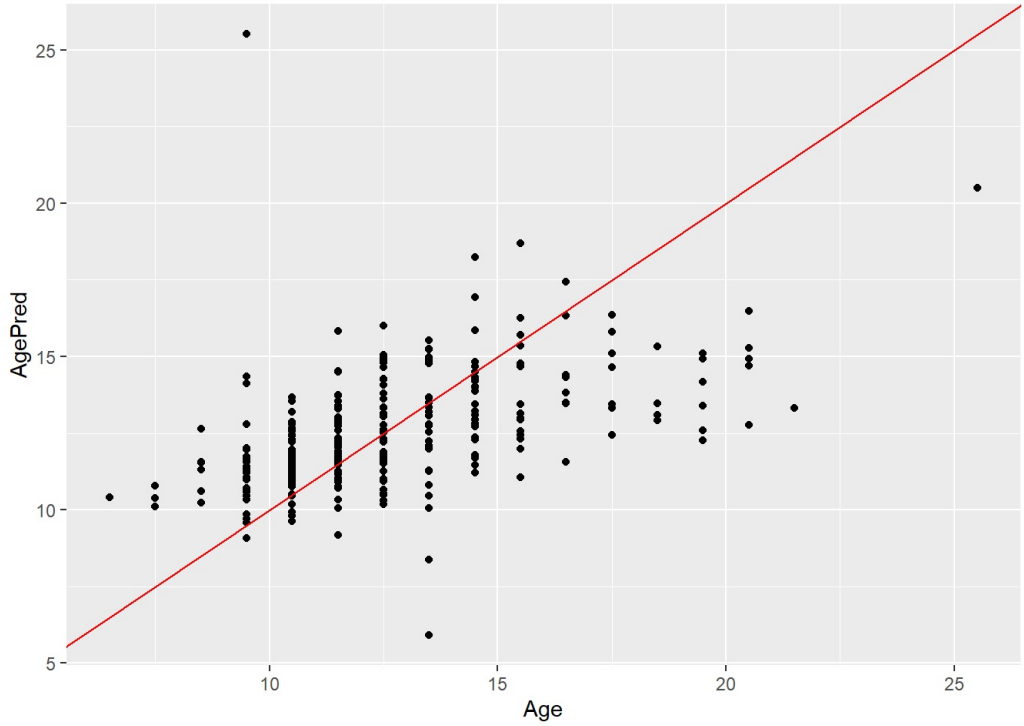





```
perform_cv_glmnet(training.abalone_Age_SexF,
                  testing.abalone_Age_SexF,
                  'Age',
                  0)
```

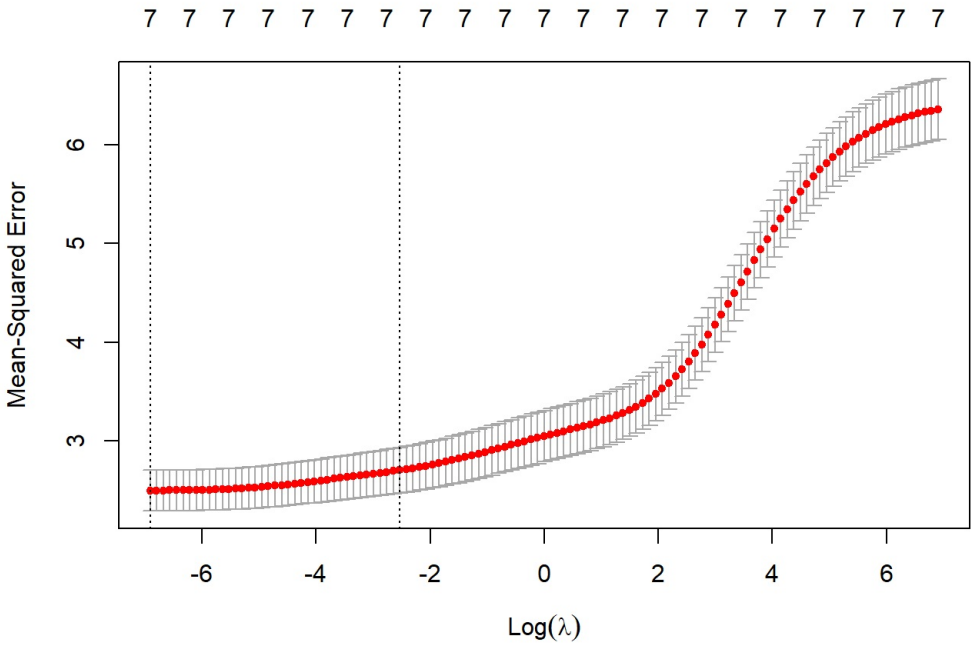
```
## [1] "--------------------------------------------------"
## $alpha
## [1] 0
##
## $lambda_min
## [1] 0.001
##
## $coef_best
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)     9.056941
## Length         -5.103387
## Diameter        5.902661
## Height         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
```

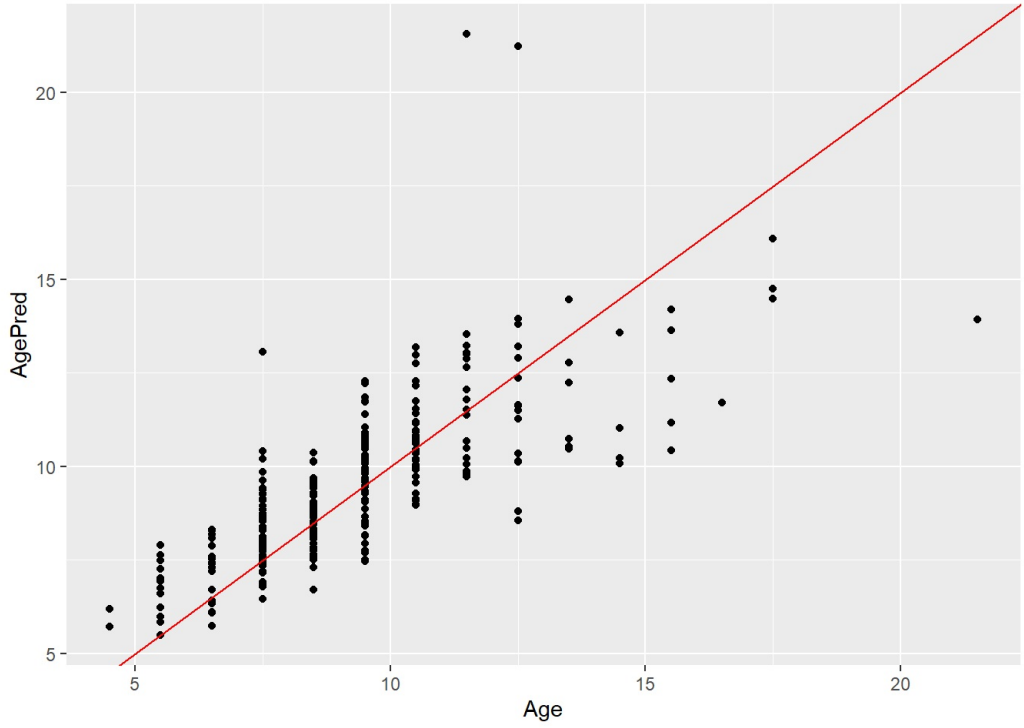





```
perform_cv_glmnet(training.abalone_Age_SexI,
                  testing.abalone_Age_SexI,
                  'Age',
                  0)
```

```
## [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
```

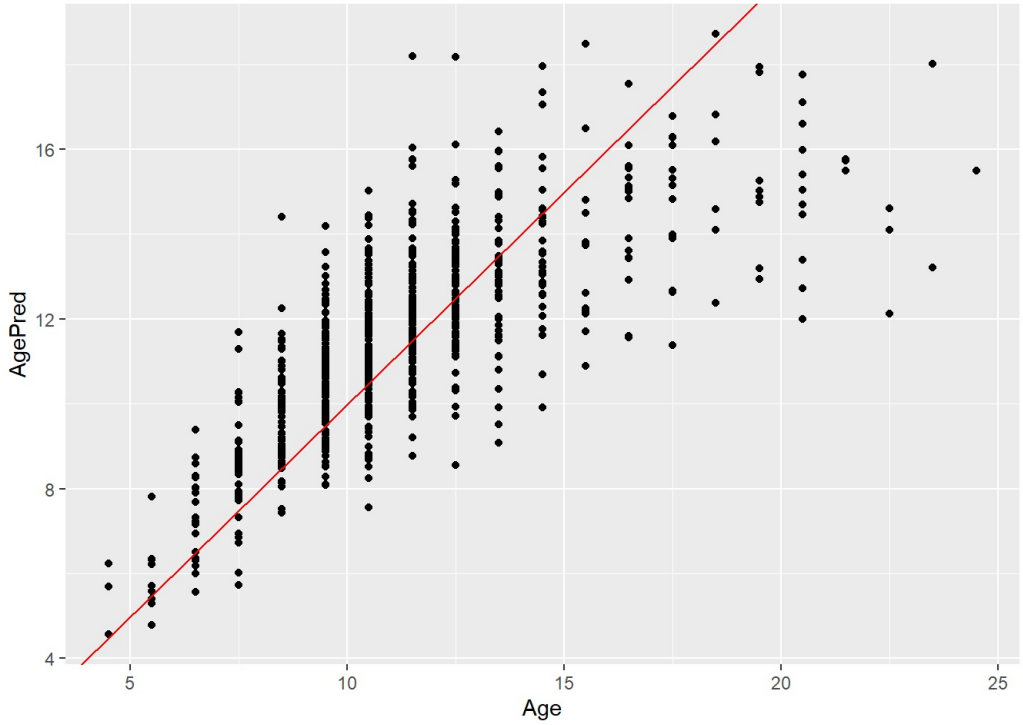
```
library(caret)
```

```
## 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")
}
```

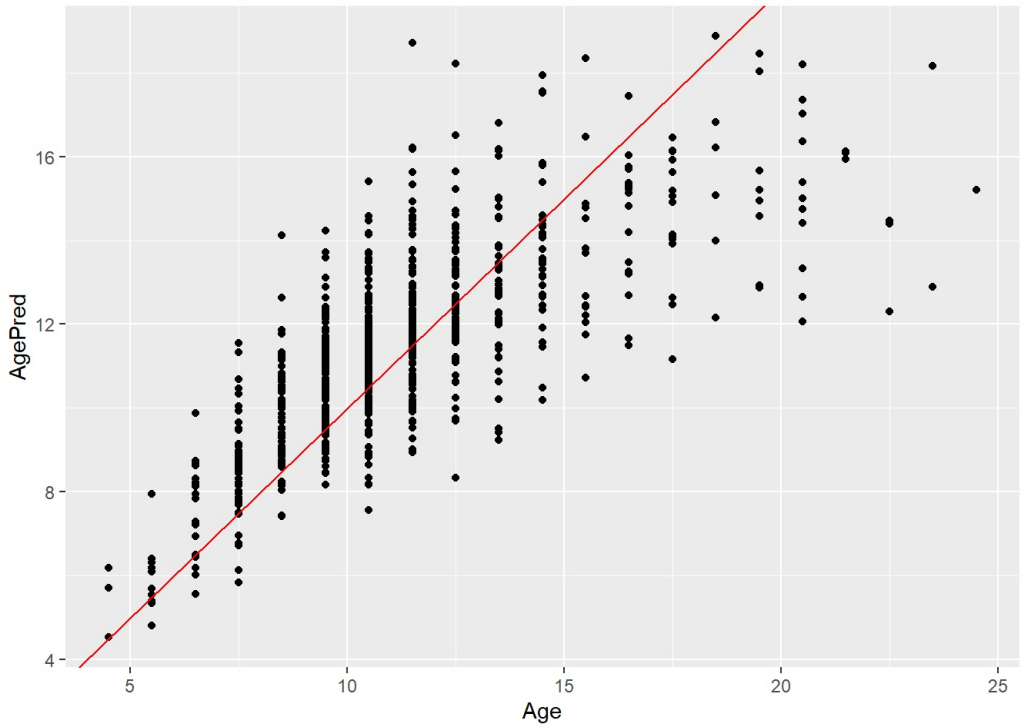
```
perform_random_forest(Age ~ ., training.abalone_Age, testing.abalone_Age)
```

```
## [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
```



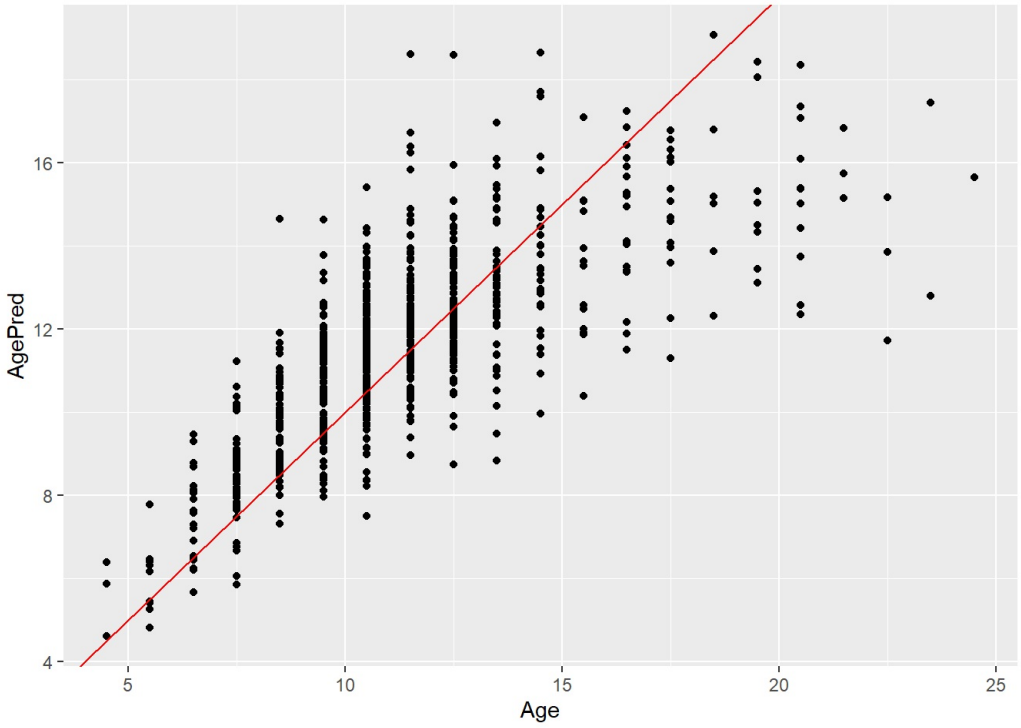

```
perform_random_forest(Age ~ .-Sex, training.abalone_Age, testing.abalone_Age)
```

```
## [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
```



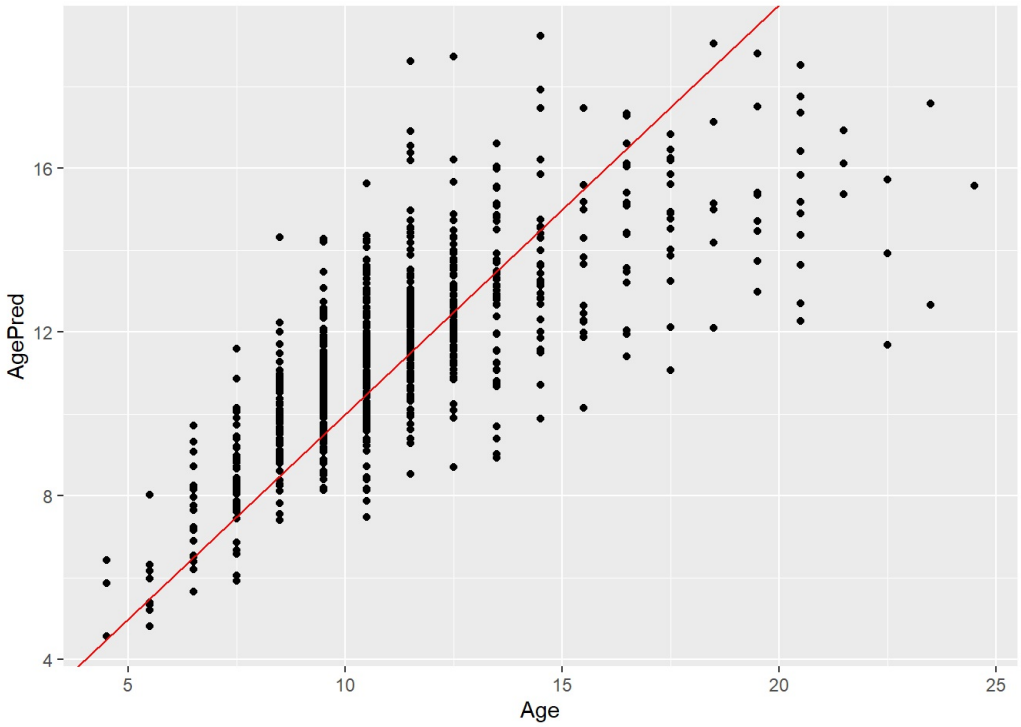
```
perform_random_forest(Age ~ .-Height, training.abalone_Age, testing.abalone_Age)
```

```
## [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