Regression

YU FAN SHIU

2024-09-28

Table of contents

Data prepare

```
library(rsample)
```

```
Warning:   'rsample'   R   4.2.3
```

```
library(dplyr)
```

```
Warning:   'dplyr'   R   4.2.3
```

```
   'dplyr'
```

```
    'package:stats':

   filter, lag

    'package:base':

   intersect, setdiff, setequal, union
```

```r
library(magrittr)
library(AmesHousing)
```

```
Warning:   'AmesHousing'   R   4.2.3
```

```r
data <- make_ames()
set.seed(123)
ames_split <- initial_split(data, prop = 0.7)
ames_train <- training(ames_split)
ames_test <- testing(ames_split)
```

Multicollinearity

```r
library(Hmisc)
```

```
Warning:   'Hmisc'   R   4.2.2

    lattice

    survival

    Formula

    ggplot2

Warning:   'ggplot2'   R   4.2.3
```

'Hmisc'

'package:dplyr':

src, summarize

'package:base':

format.pval, units

```
data <- ames_train

numeric_vars <- sapply(data, is.numeric)
data_numeric <- data[, numeric_vars]
high_cor <- data.frame(var1 = character(),
                       var2 = character(),
                       correlation = numeric(),
                       p_value = numeric())

#    p
for(i in 1:(ncol(data_numeric)-1)) {
  for(j in (i+1):ncol(data_numeric)) {
    cor_test <- cor.test(data_numeric[[i]], data_numeric[[j]])
    if(abs(cor_test$estimate) > 0.6) {
      high_cor <- rbind(high_cor, data.frame(
        var1 = colnames(data_numeric)[i],
        var2 = colnames(data_numeric)[j],
        correlation = cor_test$estimate,
        p_value = cor_test$p.value
      ))
    }
  }
}

#
high_cor <- high_cor[order(-abs(high_cor$correlation)),]

#
print(high_cor)
```

```
               var1           var2 correlation       p_value
```

```
cor10   Garage_Cars    Garage_Area   0.8884495  0.000000e+00
cor1  Total_Bsmt_SF    First_Flr_SF   0.8120419  0.000000e+00
cor7    Gr_Liv_Area  TotRms_AbvGrd   0.8093677  0.000000e+00
cor8    Gr_Liv_Area     Sale_Price   0.6985090 4.195282e-300
cor9  Bedroom_AbvGr  TotRms_AbvGrd   0.6598809 1.124915e-256
cor4  Second_Flr_SF    Gr_Liv_Area   0.6469628 1.477759e-243
cor11   Garage_Cars     Sale_Price   0.6400904 7.794268e-237
cor12   Garage_Area     Sale_Price   0.6370741 6.139591e-234
cor2  Total_Bsmt_SF     Sale_Price   0.6276502 4.283657e-225
cor6    Gr_Liv_Area      Full_Bath   0.6185420 7.835941e-217
cor5  Second_Flr_SF      Half_Bath   0.6087904 2.776851e-208
cor3   First_Flr_SF     Sale_Price   0.6085229 4.718873e-208
cor      Year_Built Year_Remod_Add   0.6045163 1.252026e-204
```

```r
lm(Sale_Price ~ Gr_Liv_Area + TotRms_AbvGrd, data = ames_train)
```

```

Call:
lm(formula = Sale_Price ~ Gr_Liv_Area + TotRms_AbvGrd, data = ames_train)

Coefficients:
  (Intercept)    Gr_Liv_Area  TotRms_AbvGrd
        42563            137         -10563
```

```r
lm(Sale_Price ~ Gr_Liv_Area, data = ames_train)
```

```

Call:
lm(formula = Sale_Price ~ Gr_Liv_Area, data = ames_train)

Coefficients:
(Intercept)  Gr_Liv_Area
    14045.9        110.7
```

```r
lm(Sale_Price ~ TotRms_AbvGrd, data = ames_train)
```

```

Call:
lm(formula = Sale_Price ~ TotRms_AbvGrd, data = ames_train)
```

```
Coefficients:
  (Intercept)  TotRms_AbvGrd
        15486          25538
```

Ridge Regression

```r
library(glmnet)
```

```
Warning:   'glmnet'   R   4.2.3

    Matrix

Warning:   'Matrix'   R   4.2.2

Loaded glmnet 4.1-8
```

```r
library(ggplot2)
library(caret)
```

```
Warning:   'caret'   R   4.2.3


   'caret'

     'package:survival':

    cluster
```

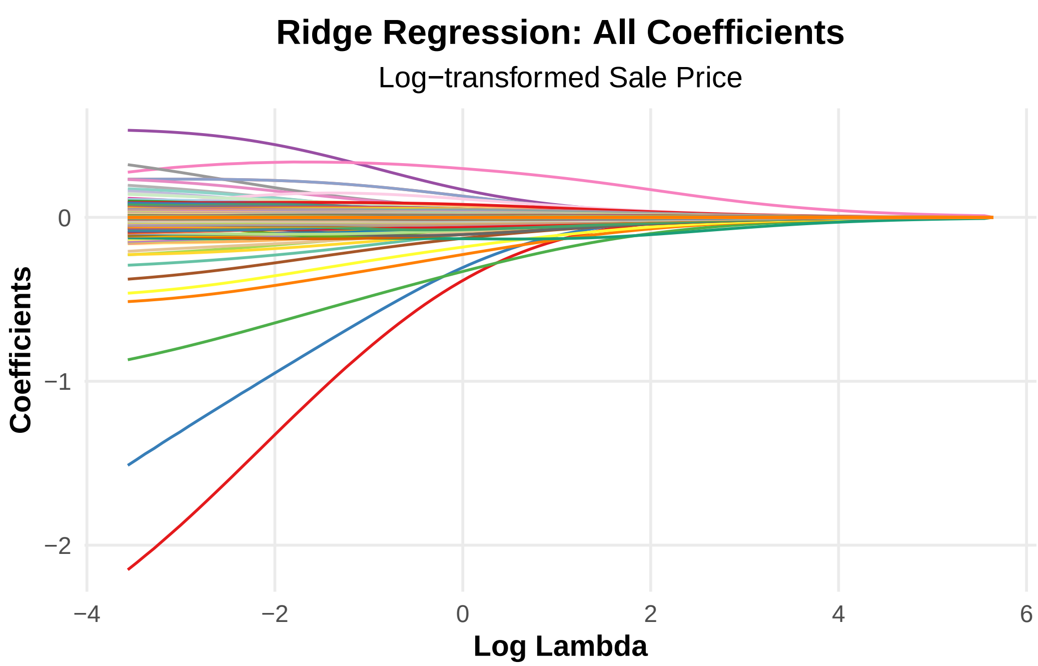
```r
library(RColorBrewer)
X <- ames_train[, -which(names(ames_train) == "Sale_Price")]
y <- log(ames_train$Sale_Price)  #   Sale_Price


numeric_vars <- names(X)[sapply(X, is.numeric)]
categorical_vars <- names(X)[sapply(X, is.factor)]
dummy <- dummyVars(" ~ .", data = X[, categorical_vars])
```

```r
X_dummy <- predict(dummy, newdata = X[, categorical_vars])
X_processed <- cbind(X[, numeric_vars], X_dummy)

ridge_model <- glmnet(as.matrix(X_processed), y, alpha = 0)

coef_matrix <- coef(ridge_model)[-1, ]
lambda_values <- ridge_model$lambda
plot_data <- data.frame(
  lambda = rep(log(lambda_values), each = nrow(coef_matrix)),
  coefficient = as.vector(coef_matrix),
  variable = rep(rownames(coef_matrix), times = ncol(coef_matrix))
)

max_coef <- plot_data %>%
  group_by(variable) %>%
  summarise(max_abs_coef = max(abs(coefficient))) %>%
  arrange(desc(max_abs_coef))

#
plot_data$variable <- factor(plot_data$variable, levels = max_coef$variable)

n_vars <- length(unique(plot_data$variable))
colors <- c(
  brewer.pal(9, "Set1"),
  brewer.pal(8, "Set2"),
  brewer.pal(12, "Set3"),
  brewer.pal(8, "Dark2"),
  brewer.pal(8, "Paired")
)
if(n_vars > length(colors)) {
  colors <- c(colors, colorRampPalette(colors)(n_vars - length(colors)))
}

p <- ggplot(plot_data, aes(x = lambda, y = coefficient, group = variable, color = variable))
  geom_line(size = 0.5) +
  theme_minimal() +
  labs(title = "Ridge Regression: All Coefficients",
       subtitle = "Log-transformed Sale Price",
       x = "Log Lambda",
       y = "Coefficients") +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
```

```r
    plot.subtitle = element_text(hjust = 0.5),
    axis.title = element_text(face = "bold"),
    legend.position = "none",
    panel.grid.minor = element_blank(),
    panel.background = element_rect(fill = "white", color = NA),
    plot.background = element_rect(fill = "white", color = NA)
  ) +
  scale_color_manual(values = colors) +
  scale_x_continuous(
    breaks = seq(floor(min(plot_data$lambda)),
                 ceiling(max(plot_data$lambda)),
                 by = 2)
  )
```

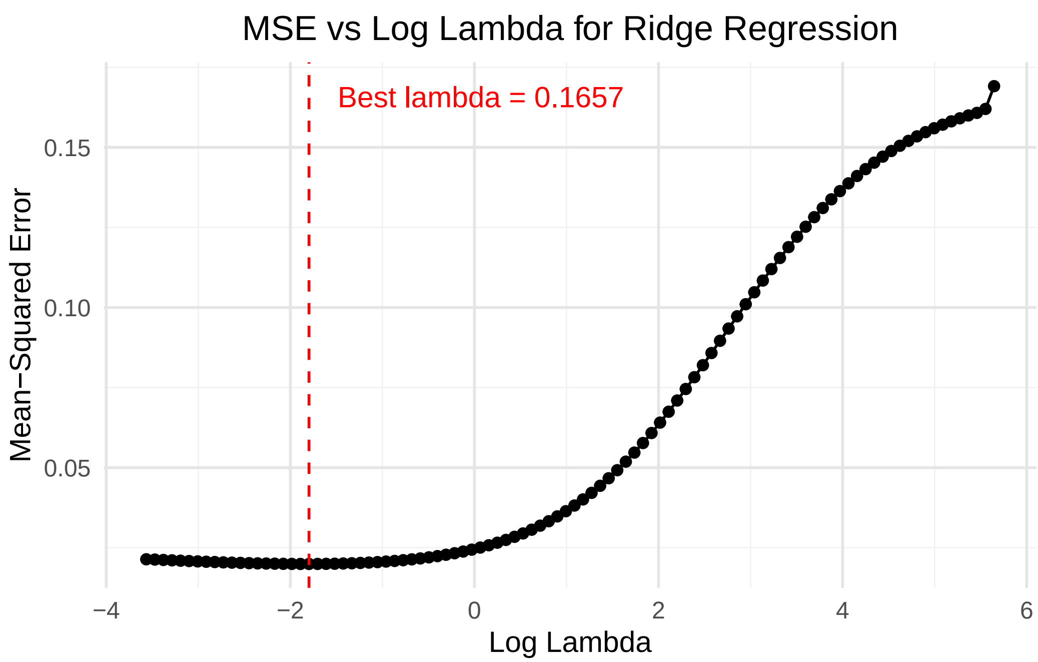
```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

```r
print(p)
```




```r
cv_ridge <- cv.glmnet(as.matrix(X_processed), y, alpha = 0, nfolds = 10)
ridge_best_lambda <- cv_ridge$lambda.min
```

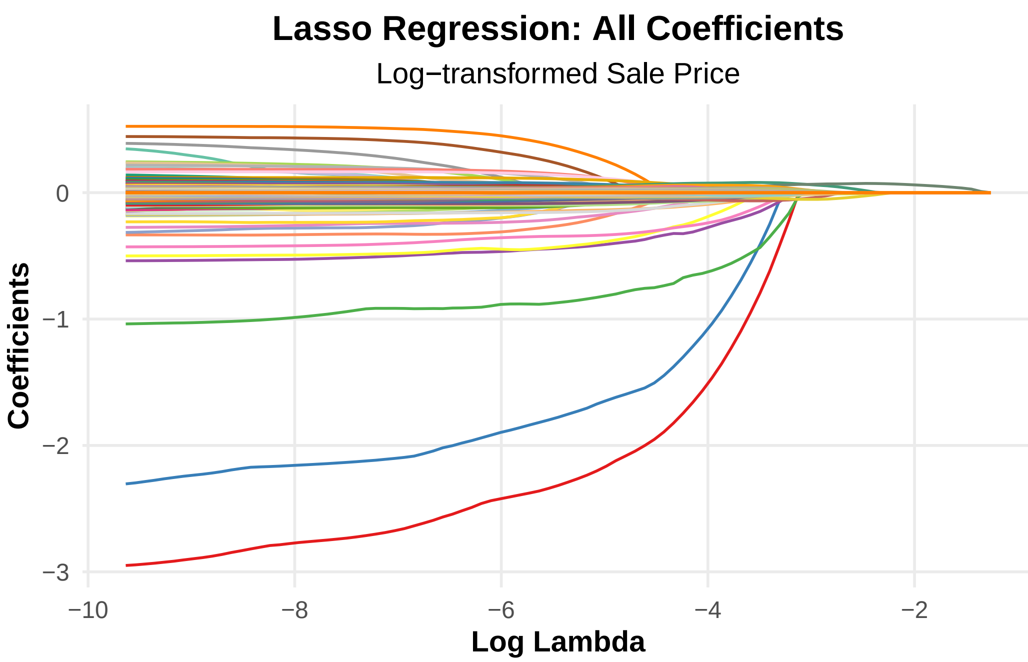
```r
plot_data <- data.frame(
  lambda = cv_ridge$lambda,
  mse = cv_ridge$cvm
)

p <- ggplot(plot_data, aes(x = log(lambda), y = mse)) +
  geom_point() +
  geom_line() +
  geom_vline(xintercept = log(ridge_best_lambda), linetype = "dashed", color = "red") +
  annotate("text", x = log(ridge_best_lambda), y = max(plot_data$mse),
           label = paste("Best lambda =", round(ridge_best_lambda, 4)),
           hjust = -0.1, vjust = 1, color = "red") +
  labs(x = "Log Lambda",
       y = "Mean-Squared Error",
       title = "MSE vs Log Lambda for Ridge Regression") +
  theme_minimal() +
  theme(
    panel.background = element_rect(fill = "white", color = NA),
    plot.background = element_rect(fill = "white", color = NA),
    panel.grid.major = element_line(color = "grey90"),
    panel.grid.minor = element_line(color = "grey95"),
    plot.title = element_text(hjust = 0.5)
  )

print(p)
```



Lasso Regression

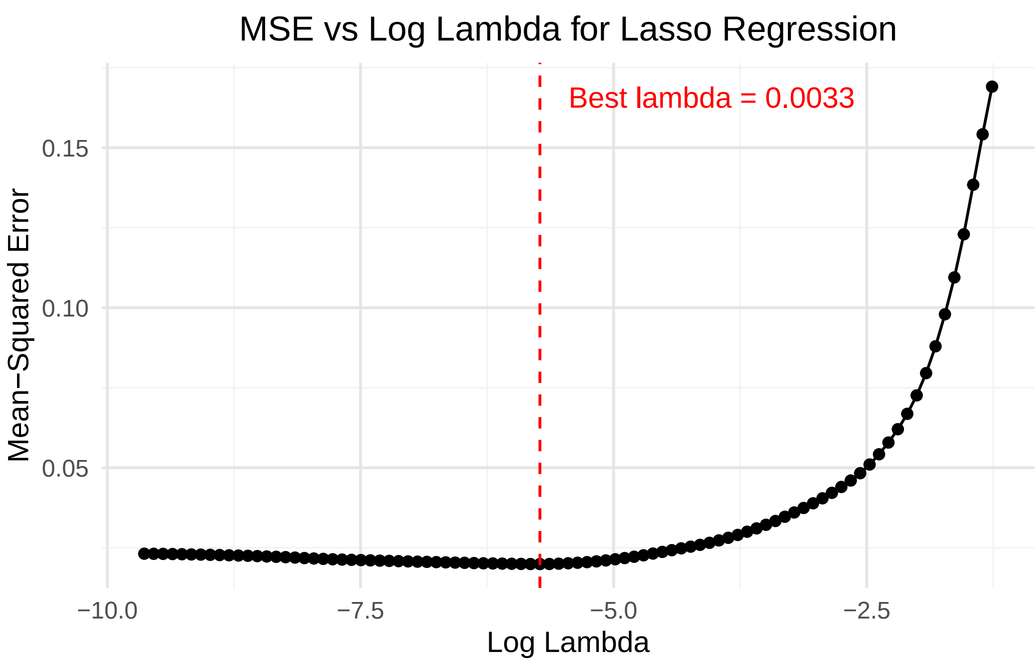
```
lasso_model <- glmnet(as.matrix(X_processed), y, alpha = 1)

coef_matrix <- coef(lasso_model)[-1, ]
lambda_values <- lasso_model$lambda


plot_data <- data.frame(
  lambda = rep(log(lambda_values), each = nrow(coef_matrix)),
  coefficient = as.vector(coef_matrix),
  variable = rep(rownames(coef_matrix), times = ncol(coef_matrix))
)

max_coef <- plot_data %>%
  group_by(variable) %>%
  summarise(max_abs_coef = max(abs(coefficient))) %>%
  arrange(desc(max_abs_coef))

#
```

```r
plot_data$variable <- factor(plot_data$variable, levels = max_coef$variable)

n_vars <- length(unique(plot_data$variable))
colors <- c(
  brewer.pal(9, "Set1"),
  brewer.pal(8, "Set2"),
  brewer.pal(12, "Set3"),
  brewer.pal(8, "Dark2"),
  brewer.pal(8, "Paired")
)
if(n_vars > length(colors)) {
  colors <- c(colors, colorRampPalette(colors)(n_vars - length(colors)))
}


p <- ggplot(plot_data, aes(x = lambda, y = coefficient, group = variable, color = variable))
  geom_line(size = 0.5) +
  theme_minimal() +
  labs(title = "Lasso Regression: All Coefficients",
       subtitle = "Log-transformed Sale Price",
       x = "Log Lambda",
       y = "Coefficients") +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5),
    axis.title = element_text(face = "bold"),
    legend.position = "none",
    panel.grid.minor = element_blank(),
    panel.background = element_rect(fill = "white", color = NA),
    plot.background = element_rect(fill = "white", color = NA)
  ) +
  scale_color_manual(values = colors) +
  scale_x_continuous(
    breaks = seq(floor(min(plot_data$lambda)),
                 ceiling(max(plot_data$lambda)),
                 by = 2)
  )
print(p)
```



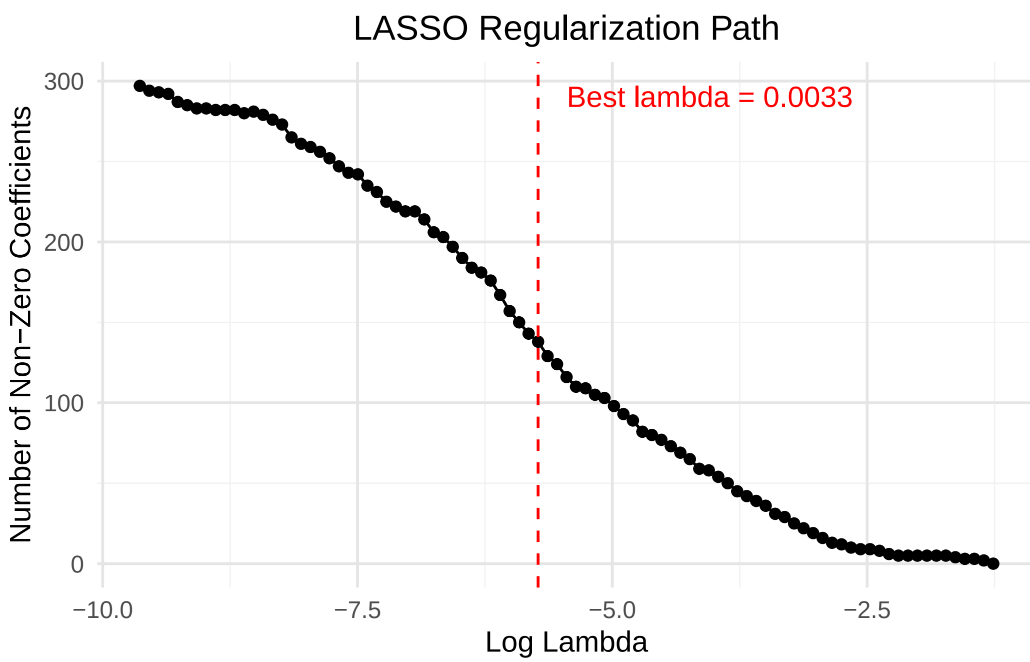

```
cv_lasso <- cv.glmnet(as.matrix(X_processed), y, alpha = 1, nfolds = 10)
lasso_best_lambda <- cv_lasso$lambda.min

plot_data <- data.frame(
  lambda = cv_lasso$lambda,
  mse = cv_lasso$cvm
)

p <- ggplot(plot_data, aes(x = log(lambda), y = mse)) +
  geom_point() +
  geom_line() +
  geom_vline(xintercept = log(lasso_best_lambda), linetype = "dashed", color = "red") +
  annotate("text", x = log(lasso_best_lambda), y = max(plot_data$mse),
           label = paste("Best lambda =", round(lasso_best_lambda, 4)),
           hjust = -0.1, vjust = 1, color = "red") +
  labs(x = "Log Lambda",
       y = "Mean-Squared Error",
       title = "MSE vs Log Lambda for Lasso Regression") +
  theme_minimal() +
  theme(
    panel.background = element_rect(fill = "white", color = NA),
    plot.background = element_rect(fill = "white", color = NA),
    panel.grid.major = element_line(color = "grey90"),
```

```
    panel.grid.minor = element_line(color = "grey95"),
    plot.title = element_text(hjust = 0.5)
  )

print(p)
```




```
#   LASSO
lasso_model <- glmnet(as.matrix(X_processed), y, alpha = 1)

#    lambda
nzero_coef <- data.frame(
  lambda = lasso_model$lambda,
  nzero = lasso_model$df
)

#
p <- ggplot(nzero_coef, aes(x = log(lambda), y = nzero)) +
  geom_line() +
  geom_point() +
  labs(x = "Log Lambda",
       y = "Number of Non-Zero Coefficients",
       title = "LASSO Regularization Path") +
```

```
  theme_minimal() +
  theme(
    panel.background = element_rect(fill = "white", color = NA),
    plot.background = element_rect(fill = "white", color = NA),
    panel.grid.major = element_line(color = "grey90"),
    panel.grid.minor = element_line(color = "grey95"),
    plot.title = element_text(hjust = 0.5)
  )
p <- p +
  geom_vline(xintercept = log(cv_lasso$lambda.min), linetype = "dashed", color = "red") +
  annotate("text", x = log(cv_lasso$lambda.min), y = max(nzero_coef$nzero),
           label = paste("Best lambda =", round(lasso_best_lambda, 4)), hjust = -0.1, vjust =

print(p)
```

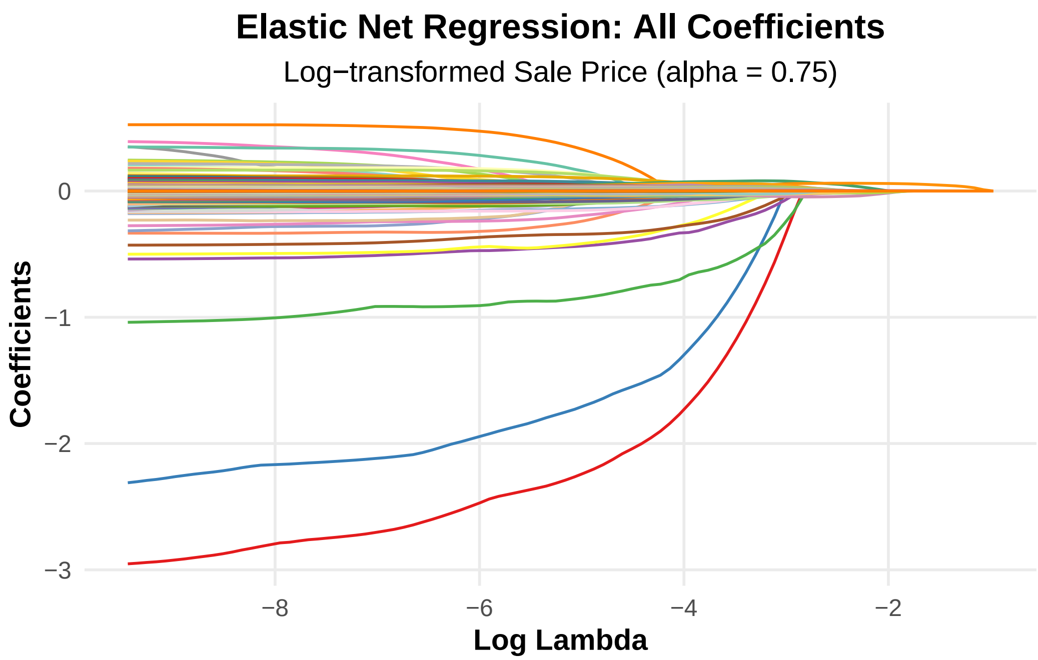


Elastic Net

```
#   Elastic Net
elastic_net_model <- glmnet(as.matrix(X_processed), y, alpha = 0.75)
```

```r
#
coef_matrix <- coef(elastic_net_model)[-1, ]

#   lambda
lambda_values <- elastic_net_model$lambda

#
plot_data <- data.frame(
  lambda = rep(log(lambda_values), each = nrow(coef_matrix)),
  coefficient = as.vector(coef_matrix),
  variable = rep(rownames(coef_matrix), times = ncol(coef_matrix))
)

#
max_coef <- plot_data %>%
  group_by(variable) %>%
  summarise(max_abs_coef = max(abs(coefficient))) %>%
  arrange(desc(max_abs_coef))

#
plot_data$variable <- factor(plot_data$variable, levels = max_coef$variable)

#
n_vars <- length(unique(plot_data$variable))
colors <- c(
  brewer.pal(9, "Set1"),
  brewer.pal(8, "Set2"),
  brewer.pal(12, "Set3"),
  brewer.pal(8, "Dark2"),
  brewer.pal(8, "Paired")
)
if(n_vars > length(colors)) {
  colors <- c(colors, colorRampPalette(colors)(n_vars - length(colors)))
}

#   ggplot
p <- ggplot(plot_data, aes(x = lambda, y = coefficient, group = variable, color = variable))
  geom_line(size = 0.5) +
  theme_minimal() +
  labs(title = "Elastic Net Regression: All Coefficients",
       subtitle = "Log-transformed Sale Price (alpha = 0.75)",
       x = "Log Lambda",
```

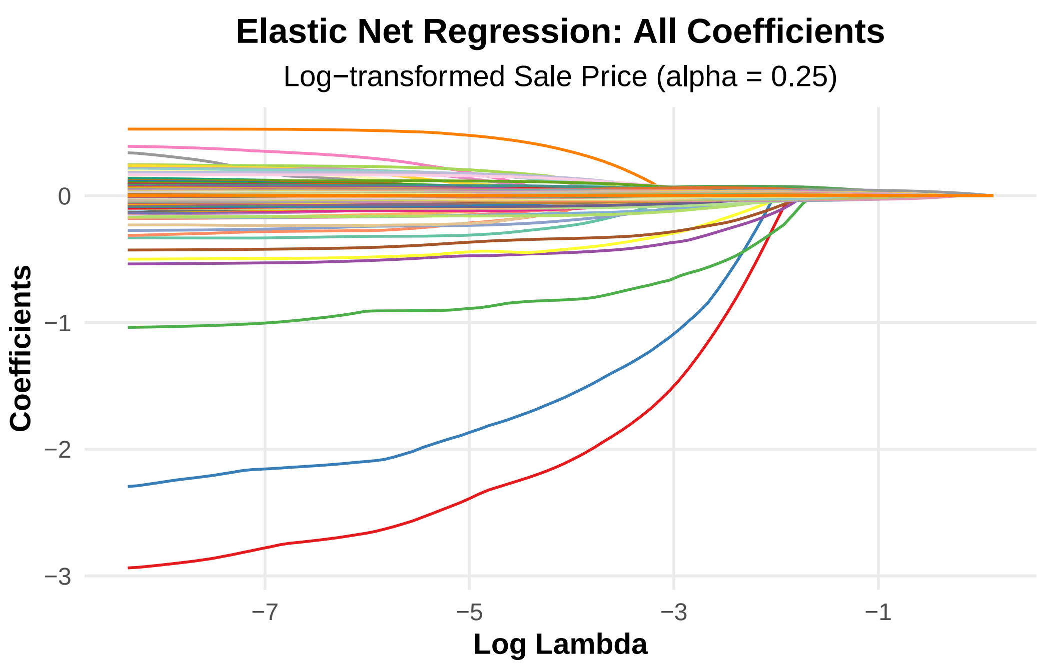
```r
        y = "Coefficients") +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5),
    axis.title = element_text(face = "bold"),
    legend.position = "none",
    panel.grid.minor = element_blank(),
    panel.background = element_rect(fill = "white", color = NA),
    plot.background = element_rect(fill = "white", color = NA)
  ) +
  scale_color_manual(values = colors) +
  scale_x_continuous(
    breaks = seq(floor(min(plot_data$lambda)),
                 ceiling(max(plot_data$lambda)),
                 by = 2)
  )

#
print(p)
```




```r
#   Elastic Net
elastic_net_model <- glmnet(as.matrix(X_processed), y, alpha = 0.25)
```

```r
#
coef_matrix <- coef(elastic_net_model)[-1, ]

#   lambda
lambda_values <- elastic_net_model$lambda

#
plot_data <- data.frame(
  lambda = rep(log(lambda_values), each = nrow(coef_matrix)),
  coefficient = as.vector(coef_matrix),
  variable = rep(rownames(coef_matrix), times = ncol(coef_matrix))
)

#
max_coef <- plot_data %>%
  group_by(variable) %>%
  summarise(max_abs_coef = max(abs(coefficient))) %>%
  arrange(desc(max_abs_coef))

#
plot_data$variable <- factor(plot_data$variable, levels = max_coef$variable)

#
n_vars <- length(unique(plot_data$variable))
colors <- c(
  brewer.pal(9, "Set1"),
  brewer.pal(8, "Set2"),
  brewer.pal(12, "Set3"),
  brewer.pal(8, "Dark2"),
  brewer.pal(8, "Paired")
)
if(n_vars > length(colors)) {
  colors <- c(colors, colorRampPalette(colors)(n_vars - length(colors)))
}

#   ggplot
p <- ggplot(plot_data, aes(x = lambda, y = coefficient, group = variable, color = variable))
  geom_line(size = 0.5) +
  theme_minimal() +
  labs(title = "Elastic Net Regression: All Coefficients",
       subtitle = "Log-transformed Sale Price (alpha = 0.25)",
       x = "Log Lambda",
```

```
      y = "Coefficients") +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5),
    axis.title = element_text(face = "bold"),
    legend.position = "none",
    panel.grid.minor = element_blank(),
    panel.background = element_rect(fill = "white", color = NA),
    plot.background = element_rect(fill = "white", color = NA)
  ) +
  scale_color_manual(values = colors) +
  scale_x_continuous(
    breaks = seq(floor(min(plot_data$lambda)),
                 ceiling(max(plot_data$lambda)),
                 by = 2)
  )

#
print(p)
```




```
library(tidyverse)
```

```
Warning:   'tidyverse'   R   4.2.3

Warning:   'tibble'   R   4.2.3

Warning:   'tidyr'   R   4.2.3

Warning:   'readr'   R   4.2.3

Warning:   'purrr'   R   4.2.2

Warning:   'stringr'   R   4.2.3

Warning:   'forcats'   R   4.2.3

Warning:   'lubridate'   R   4.2.3

-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v stringr   1.5.0
v lubridate 1.9.3     v tibble    3.2.1
v purrr     1.0.1     v tidyr     1.3.0
v readr     2.1.5
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x tidyr::expand()    masks Matrix::expand()
x tidyr::extract()   masks magrittr::extract()
x dplyr::filter()    masks stats::filter()
x dplyr::lag()       masks stats::lag()
x purrr::lift()      masks caret::lift()
x tidyr::pack()      masks Matrix::pack()
x purrr::set_names() masks magrittr::set_names()
x Hmisc::src()       masks dplyr::src()
x Hmisc::summarize() masks dplyr::summarize()
x tidyr::unpack()    masks Matrix::unpack()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
set.seed(123)

#   fold_id
nfolds <- 10
fold_id <- sample(rep(1:nfolds, length.out = nrow(X_processed)))
```

```r
#
tuning_grid <- expand.grid(
  alpha = seq(0, 1, by = .1),
  mse_min = NA,
  mse_1se = NA,
  lambda_min = NA,
  lambda_1se = NA
)

#    alpha
for(i in seq_along(tuning_grid$alpha)) {
  #
  fit <- cv.glmnet(
    x = as.matrix(X_processed),
    y = y,
    alpha = tuning_grid$alpha[i],
    foldid = fold_id,
    nfolds = nfolds
  )

  #   MSE   lambda
  tuning_grid$mse_min[i] <- min(fit$cvm)
  tuning_grid$mse_1se[i] <- fit$cvm[fit$lambda == fit$lambda.1se]
  tuning_grid$lambda_min[i] <- fit$lambda.min
  tuning_grid$lambda_1se[i] <- fit$lambda.1se
}

#
print(tuning_grid)
```

```
   alpha    mse_min    mse_1se  lambda_min  lambda_1se
1    0.0 0.02027839 0.02289314 0.181871094 0.734216744
2    0.1 0.02056912 0.02371082 0.029640457 0.109028856
3    0.2 0.02082546 0.02394396 0.014820228 0.054514428
4    0.3 0.02092240 0.02456036 0.009880152 0.039886345
5    0.4 0.02098157 0.02460525 0.006751820 0.029914759
6    0.5 0.02099941 0.02471517 0.005401456 0.023931807
7    0.6 0.02099293 0.02433927 0.004501213 0.018171476
8    0.7 0.02094646 0.02434270 0.003858183 0.015575551
9    0.8 0.02081266 0.02427863 0.003375910 0.013628607
10   0.9 0.02043994 0.02404927 0.003000809 0.012114317
11   1.0 0.01971851 0.02319865 0.002460803 0.009934303
```

```r
#   MSE ±
tuning_grid %>%
  mutate(se = mse_1se - mse_min) %>%
  ggplot(aes(alpha, mse_min)) +
  geom_line(size = 2) +
  geom_ribbon(aes(ymax = mse_min + se, ymin = mse_min - se), alpha = .25) +
  ggtitle("MSE ± one standard error") +
  xlab("Alpha") +
  ylab("Mean Squared Error") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))  #
```


```r
#     alpha   lambda
best_model <- tuning_grid %>%
  filter(mse_min == min(mse_min))

cat("Best alpha:", best_model$alpha, "\n")
```

```
Best alpha: 1
```

```
cat("Best lambda (min MSE):", best_model$lambda_min, "\n")
```

```
Best lambda (min MSE): 0.002460803
```

```
cat("Best lambda (1SE rule):", best_model$lambda_1se, "\n")
```

```
Best lambda (1SE rule): 0.009934303
```

Prediction

```
library(Metrics)
```

```
Warning:   'Metrics'   R   4.2.3


   'Metrics'

      'package:caret':

    precision, recall
```

```
X_test <- ames_test[, -which(names(ames_test) == "Sale_Price")]
y_test <- log(ames_test$Sale_Price)  #   Sale_Price

#       dummy
X_dummy_test <- predict(dummy, newdata = X_test[, categorical_vars])
X_processed_test <- cbind(X_test[, numeric_vars], X_dummy_test)

#
X_processed_test <- X_processed_test[, colnames(X_processed)]
lm_model <- lm(y ~ ., data = as.data.frame(X_processed))

#    lambda   Ridge   Lasso
ridge_model <- glmnet(as.matrix(X_processed), y, alpha = 0, lambda = ridge_best_lambda)
lasso_model <- glmnet(as.matrix(X_processed), y, alpha = 1, lambda = lasso_best_lambda)
```

```r
#
X_processed_test <- as.matrix(X_processed_test)

#
lm_predictions <- predict(lm_model, newdata = as.data.frame(X_processed_test))
```

```
Warning in predict.lm(lm_model, newdata = as.data.frame(X_processed_test)):
```

```r
# Ridge
ridge_predictions <- predict(ridge_model, newx = X_processed_test)

# Lasso
lasso_predictions <- predict(lasso_model, newx = X_processed_test)

#   MSE
lm_mse <- mean((exp(y_test) - exp(lm_predictions))^2)
ridge_mse <- mean((exp(y_test) - exp(ridge_predictions))^2)
lasso_mse <- mean((exp(y_test) - exp(lasso_predictions))^2)

#   R-squared
lm_r_squared <- cor(exp(y_test), exp(lm_predictions))^2
ridge_r_squared <- cor(exp(y_test), exp(ridge_predictions))^2
lasso_r_squared <- cor(exp(y_test), exp(lasso_predictions))^2

#
cat("Linear Regression:\n")
```

```
Linear Regression:
```

```r
cat(paste("MSE:", lm_mse, "\n"))
```

```
MSE: 1133542940.93825
```

```r
cat(paste("R-squared:", lm_r_squared, "\n\n"))
```

```
R-squared: 0.835705756583587
```

```r
cat("Ridge Regression:\n")
```

```
Ridge Regression:
```

```r
cat(paste("MSE:", ridge_mse, "\n"))
```

```
MSE: 757893241.524645
```

```r
cat(paste("R-squared:", ridge_r_squared, "\n\n"))
```

```
R-squared: 0.879322121231107
```

```r
cat("Lasso Regression:\n")
```

```
Lasso Regression:
```

```r
cat(paste("MSE:", lasso_mse, "\n"))
```

```
MSE: 1027655818.72105
```

```r
cat(paste("R-squared:", lasso_r_squared, "\n"))
```

```
R-squared: 0.843884072404135
```