Task 2 - Ridge regression

Group 3

25/02/2020

1 Setup

```
# libraries
library(ggplot2)
library(tidyverse)
library(kableExtra)
library(gridExtra)

# function to format printed numbers
formatgraph <- function(x) {
    formatC(x, format = "e", digits = 3)
}
```

2 Choosing the penalization parameter λ

2.1 Exercise 1

Write an R function implementing the ridge regression penalization parameter λ choice based on the minimization of the mean squared prediction error in a validation set ($MSPE_{val}(\lambda)$). Input: * Matrix x and vector y corresponding to the training sample * matrix eval and vector eval corresponding to the validation set * a vector lambda.v of candidate values for λ.

Output: For each element λ in lambda.v, the value of MSPEval(λ).

Additionally you can plot these values against log(1 + λ) - 1, or against df(λ).

```
# MSE function
mse <- function(obs, pred) {
    return(mean((obs - pred)^2))
}

# Validation set function
lambda_validation <- function(x, y, xval, yval, lambda.v) {
    # Ridge regression training nums <- unlist(lapply(boston.c,is.numeric))
    Y <- scale(as.matrix(y), center = TRUE, scale = FALSE)
    X <- scale(as.matrix(x), center = TRUE, scale = TRUE)
    center_X <- colMeans(as.matrix(x), na.rm = TRUE)
    center_Y <- colMeans(as.matrix(y), na.rm = TRUE)
    scale_X <- sqrt(diag(cov(x)))

    n <- dim(X)[1]
    p <- dim(X)[2]
    n.lambdas <- length(lambda.v)
```

```
    XtX <- t(X) %*% X
    beta.path <- matrix(0, nrow = n.lambdas, ncol = p)
    diag.H.lambda <- matrix(0, nrow = n.lambdas, ncol = n)
    for (l in 1:n.lambdas) {
        lambda <- lambda.v[l]
        H.lambda.aux <- t(solve(XtX + lambda * diag(1, p))) %*% t(X)
        beta.path[l, ] <- H.lambda.aux %*% Y
        H.lambda <- X %*% H.lambda.aux
        diag.H.lambda[l, ] <- diag(H.lambda)
    }

    # MSPE on validation set
    Y_val <- scale(as.matrix(yval), center = center_Y, scale = FALSE)
    X_val <- scale(as.matrix(xval), center = center_X, scale = scale_X)
    pred <- X_val %*% t(beta.path)
    mspe_val <- apply(pred, 2, mse, obs = Y_val)

    lambda_opt <- lambda.v[which.min(mspe_val)]

    # Plot MSPE in function of log(1 + lambda) - 1
    df <- data.frame(MSPE = mspe_val, f.lambda = log(1 + lambda.v) - 1)
    g <- ggplot(df, aes(x = f.lambda, y = MSPE)) + geom_point() + geom_line() + xlab("log(1+lambda) - 1
        geom_vline(xintercept = log(1 + lambda_opt) - 1, linetype = "dashed") + annotate("label",
        x = log(1 + lambda_opt) - 1, y = mean(mspe_val), label = paste("lambda =",
            round(lambda_opt, 3)), fill = "white", size = 3.5) + ggtitle("Validation set") +
        theme(plot.title = element_text(size = 10, face = "bold"), text = element_text(size = 10))
    # print(g)

    return(list(mspe_val = mspe_val, lambda.opt = lambda_opt, MSPE.opt = min(mspe_val),
        plot = g, coeff.lambda = beta.path))
}
```

2.2 Exercise 2

Write an R function implementing the ridge regression penalization parameter λ choice based on k-fold cross-validation ($MSPE_{k-CV}(\lambda)$). Input, output and graphics as before (except that xval and yval are not required now as input).

```
lambda_kcv <- function(x, y, lambda.v, k = 5) {

    # Folds
    n_all <- dim(x)[1]
    shuffled <- sample(1:n_all, n_all, replace = FALSE)
    max <- n_all%/%k
    folds <- split(shuffled, ceiling(seq(1:n_all)/max))
    folds[[k]] <- c(folds[[k]], folds[[(k + 1)]])
    folds[[(k + 1)]] <- NULL

    MSPE_folds <- data.frame(matrix(ncol = k, nrow = length(lambda.v)))

    n.lambdas <- length(lambda.v)

    # Training and MSPE per fold
```

```
    for (i in 1:k) {
        test <- folds[[i]]
        train <- shuffled[!(shuffled %in% test)]
        # Ridge regression training
        Y <- scale(as.matrix(y[train]), center = TRUE, scale = FALSE)
        X <- scale(as.matrix(x[train, ]), center = TRUE, scale = TRUE)
        center_X <- colMeans(as.matrix(x), na.rm = TRUE)
        center_Y <- colMeans(as.matrix(y), na.rm = TRUE)
        scale_X <- sqrt(diag(cov(x)))

        n <- dim(X)[1]
        p <- dim(X)[2]

        XtX <- t(X) %*% X
        beta.path <- matrix(0, nrow = n.lambdas, ncol = p)
        diag.H.lambda <- matrix(0, nrow = n.lambdas, ncol = n)
        for (l in 1:n.lambdas) {
            lambda <- lambda.v[l]
            H.lambda.aux <- t(solve(XtX + lambda * diag(1, p))) %*% t(X)
            beta.path[l, ] <- H.lambda.aux %*% Y
            H.lambda <- X %*% H.lambda.aux
            diag.H.lambda[l, ] <- diag(H.lambda)
        }

        # MSPE on validation fold
        Y_val <- scale(as.matrix(y[test]), center = center_Y, scale = FALSE)
        X_val <- scale(as.matrix(x[test, ]), center = center_X, scale = scale_X)
        pred <- X_val %*% t(beta.path)
        MSPE_folds[, i] <- apply(pred, 2, mse, obs = Y_val)
    }

    MSPE_lambda <- rowMeans(MSPE_folds, na.rm = TRUE)

    lambda_opt <- lambda.v[which.min(MSPE_lambda)]

    # Plot MSPE in function of log(1 + lambda) - 1
    df <- data.frame(MSPE = MSPE_lambda, f.lambda = log(1 + lambda.v) - 1)

    g <- ggplot(df, aes(x = f.lambda, y = MSPE)) + geom_point() + geom_line() + xlab("log(1+lambda) - 1
        geom_vline(xintercept = log(1 + lambda_opt) - 1, linetype = "dashed") + annotate("label",
        x = log(1 + lambda_opt) - 1, y = mean(MSPE_lambda), label = paste("lambda =",
            round(lambda_opt, 3)), fill = "white", size = 3.5) + ggtitle(paste0(k,
        "-fold cross-validation")) + theme(plot.title = element_text(size = 10, face = "bold"),
        text = element_text(size = 10))

    # print(g)

    return(list(MSPE.all = MSPE_lambda, lambda.opt = lambda_opt, MSPE.opt = min(MSPE_lambda),
        plot = g, coeff.lambda = beta.path))
}
```

2.3 Exercise 3

Consider the prostate date used in class. Use your routines to choose the penalization parameter λ by the following criteria: * behaviour in the validation set (the 30 observations not being in the training sample) * 5-fold and 10-fold cross-validation. Compare your results with those obtained when using leave-one-out and generalized cross-validation.

```
# data
prostate <- read.table("prostate_data.txt", header = TRUE, row.names = 1)

# Initialization of values of lambda
lambda.max <- 150
n.lambdas <- 50

lambda.v <- exp(seq(0, log(lambda.max + 1), length = n.lambdas)) - 1
```

2.3.1 Validation set

```
set.seed(615)

# Partition between training and validation set No usado

# n_all <- dim(prostate)[1] shuffled <- sample(1:n_all, n_all, replace = FALSE)
# training <- shuffled[1:(round(n_all*2/3,0)+1)] validation <-
# shuffled[!(shuffled %in% training)]

y <- prostate[prostate$train, ]$lpsa
x <- prostate[prostate$train, 1:8]

yval <- prostate[!prostate$train, ]$lpsa
xval <- prostate[!prostate$train, 1:8]

validation <- lambda_validation(x, y, xval, yval, lambda.v)
```

2.3.2 5-fold cross-validation

```
y <- prostate$lpsa
x <- prostate[, 1:8]
cv.5 <- lambda_kcv(x, y, lambda.v, k = 5)
```

2.3.3 10-fold cross-validation

```
y <- prostate$lpsa
x <- prostate[, 1:8]
cv.10 <- lambda_kcv(x, y, lambda.v, k = 10)
```

2.3.4 Leave-one-out cross-validation

```
Y <- scale(prostate$lpsa, center = TRUE, scale = FALSE)
X <- scale(as.matrix(prostate[, 1:8]), center = TRUE, scale = TRUE)
n <- dim(X)[1]
p <- dim(X)[2]
```

```
MSPE.loocv <- numeric(n.lambdas)
for (l in 1:n.lambdas) {
    lambda <- lambda.v[l]
    MSPE.loocv[l] <- 0
    for (i in 1:n) {
        m.Y.i <- 0
        X.i <- X[-i, ]
        Y.i <- Y[-i] - m.Y.i
        Xi <- X[i, ]
        Yi <- Y[i]
        beta.i <- solve(t(X.i) %*% X.i + lambda * diag(1, p)) %*% t(X.i) %*% Y.i
        hat.Yi <- Xi %*% beta.i + m.Y.i
        MSPE.loocv[l] <- MSPE.loocv[l] + (hat.Yi - Yi)^2
    }
    MSPE.loocv[l] <- MSPE.loocv[l]/n
}

lambda_opt_loocv <- lambda.v[which.min(MSPE.loocv)]
MSPE.opt.loocv <- min(MSPE.loocv)

df <- data.frame(MSPE = MSPE.loocv, f.lambda = log(1 + lambda.v) - 1)

# Plot MSPE in function of log(1 + lambda) - 1
g.loocv <- ggplot(df, aes(x = f.lambda, y = MSPE)) + geom_point() + geom_line() +
    xlab("log(1+lambda) - 1") + geom_vline(xintercept = log(1 + lambda_opt_loocv) -
    1, linetype = "dashed") + annotate("label", x = log(1 + lambda_opt_loocv) - 1,
    y = mean(MSPE.loocv), label = paste("lambda =", round(lambda_opt_loocv, 3)),
    fill = "white", size = 3.5) + ggtitle("Leave-one cross-validation") + theme(plot.title = element_te
    face = "bold"), text = element_text(size = 10))
```

2.3.5 Generalized cross-validation

```
XtX <- t(X) %*% X
beta.path <- matrix(0, nrow = n.lambdas, ncol = p)
diag.H.lambda <- matrix(0, nrow = n.lambdas, ncol = n)

for (l in 1:n.lambdas) {
    lambda <- lambda.v[l]
    H.lambda.aux <- t(solve(XtX + lambda * diag(1, p))) %*% t(X)
    beta.path[l, ] <- H.lambda.aux %*% Y
    H.lambda <- X %*% H.lambda.aux
    diag.H.lambda[l, ] <- diag(H.lambda)
}

MSPE.gcv <- numeric(n.lambdas)
for (l in 1:n.lambdas) {
    lambda <- lambda.v[l]
    hat.Y <- X %*% beta.path[l, ]
    nu <- sum(diag.H.lambda[l, ])
    MSPE.gcv[l] <- sum(((Y - hat.Y)/(1 - nu/n))^2)/n
}

lambda_opt_gcv <- lambda.v[which.min(MSPE.gcv)]
```

```
MSPE.opt.gcv <- min(MSPE.gcv)

df <- data.frame(MSPE = MSPE.gcv, f.lambda = log(1 + lambda.v) - 1)

# Plot MSPE in function of log(1 + lambda) - 1
g.gcv <- ggplot(df, aes(x = f.lambda, y = MSPE)) + geom_point() + geom_line() + xlab("log(1+lambda) - 1
    geom_vline(xintercept = log(1 + lambda_opt_gcv) - 1, linetype = "dashed") + annotate("label",
    x = log(1 + lambda_opt_gcv) - 1, y = mean(MSPE.gcv), label = paste("lambda =",
        round(lambda_opt_gcv, 3)), fill = "white", size = 3.5) + ggtitle("Generalized cross-validation"
    theme(plot.title = element_text(size = 10, face = "bold"), text = element_text(size = 10))
```

2.3.6 Comparison

```
methods <- c("Validation set", "5-fold CV", "10-fold CV", "Leave-one-out CV", "Generalized CV")

lambdas <- round(c(validation$lambda.opt, cv.5$lambda.opt, cv.10$lambda.opt, lambda_opt_loocv,
    lambda_opt_gcv), 3)

MSPE.all <- round(c(validation$MSPE.opt, cv.5$MSPE.opt, cv.10$MSPE.opt, MSPE.opt.loocv,
    MSPE.opt.gcv), 3)

tbl_lambdas <- data.frame(Method = methods, Lambda = lambdas, MSPE = MSPE.all)
row.names(tbl_lambdas) <- NULL
colnames(tbl_lambdas)[2] <- "$\\lambda$"
```

For each tuning methodology, we list the lambdas obtained in Table 1 and we plot the MPSE in function of values of λ in Figure 1.

The values of λ obtained by 10-fold cross-validation, leave-one-out cross-validation and generalized cross-validation are all very similar. The values obtained by 5-fold cross-validation and the validation set approach are further away. We note however that all the lowest MSPE values found are close to each other and that the MSPE functions of the cross-validation methods have low convexity around the minimums, leading to high variability of the best λ.

```
kable(tbl_lambdas, "latex", booktabs = T, caption = "\\label{tab:tab1} Table of lambdas and MPSE obtain
    escape = FALSE) %>% kable_styling(latex_options = c("hold_position"), position = "center")
```

Table 1: Table of lambdas and MPSE obtained with each method

Method	λ	MSPE
Validation set	11.934	0.487
5-fold CV	3.193	0.520
10-fold CV	5.316	0.543
Leave-one-out CV	5.997	0.524
Generalized CV	6.751	0.522

```
grid.arrange(validation$plot, cv.5$plot, cv.10$plot, g.loocv, g.gcv, ncol = 2)
```

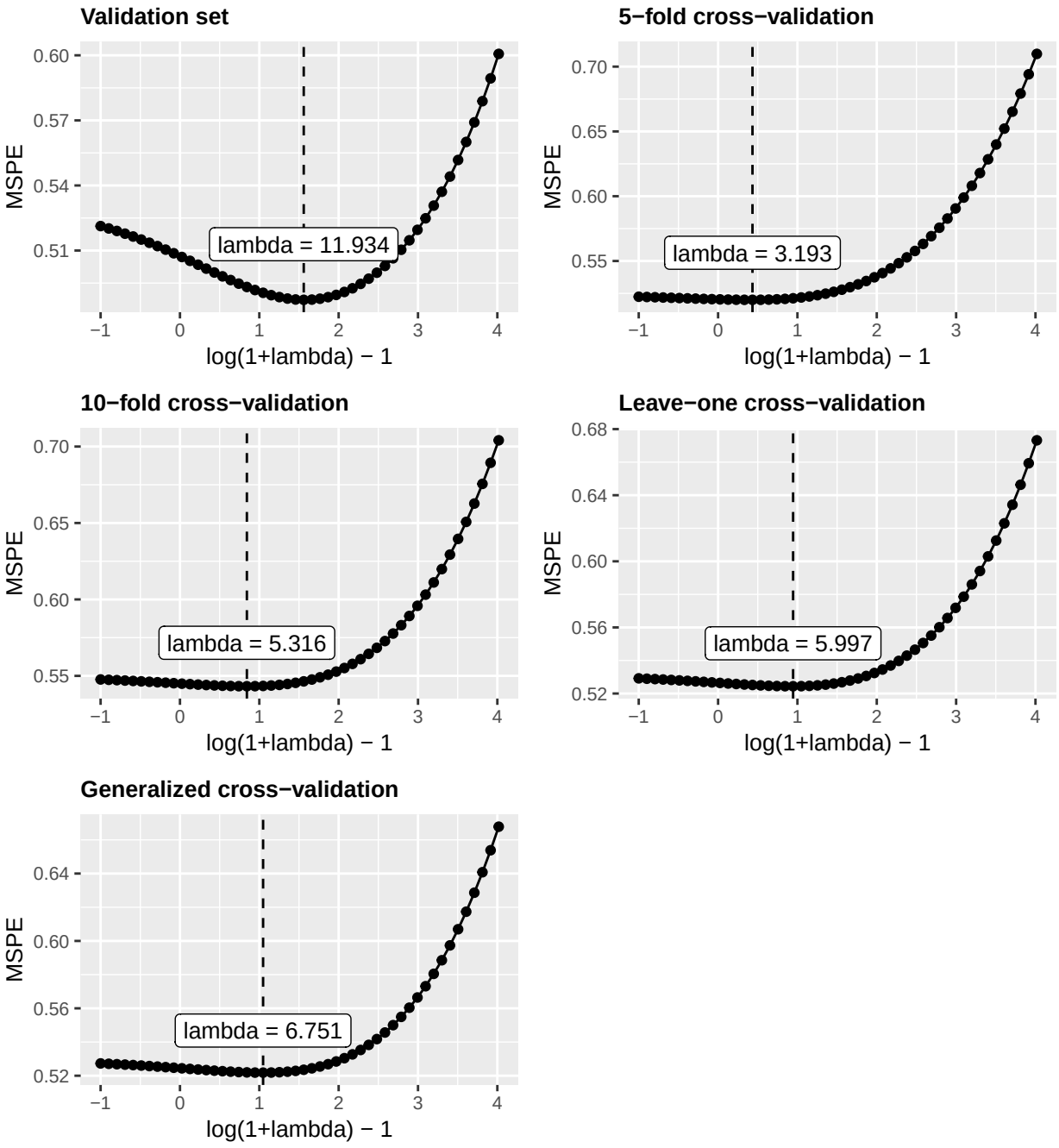

Figure 1: MSPE in function of lambda by tuning method

3 Ridge regression for the Boston Housing data

For the Boston House-price corrected dataset use ridge regression to fit the regression model where the response is MEDV and the explanatory variables are the remaining 13 variables in the previous list. Try to provide an interpretation to the estimated model.

```
# data
load("boston.Rdata")
```

```
response <- boston.c$MEDV
explanatory <- boston.c %>% mutate(CHAS = as.numeric(CHAS)) %>% select(c(CRIM, ZN,
    INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT))
```

```
# Initialization of values of lambda
lambda.max <- 10
n.lambdas <- 50

lambda.v <- exp(seq(0, log(lambda.max + 1), length = n.lambdas)) - 1

set.seed(45)
cv.5.boston <- lambda_kcv(explanatory, response, lambda.v, k = 5)
```

We implement a 5-fold cross-validation to select the tuning parameter λ. In Figure 2, we plot the MPSE in function of values of λ. We obtain $\lambda \approx 3.341$ and a MPSE of approximately 23.195 for that value of λ.

```
cv.5.boston$plot
```

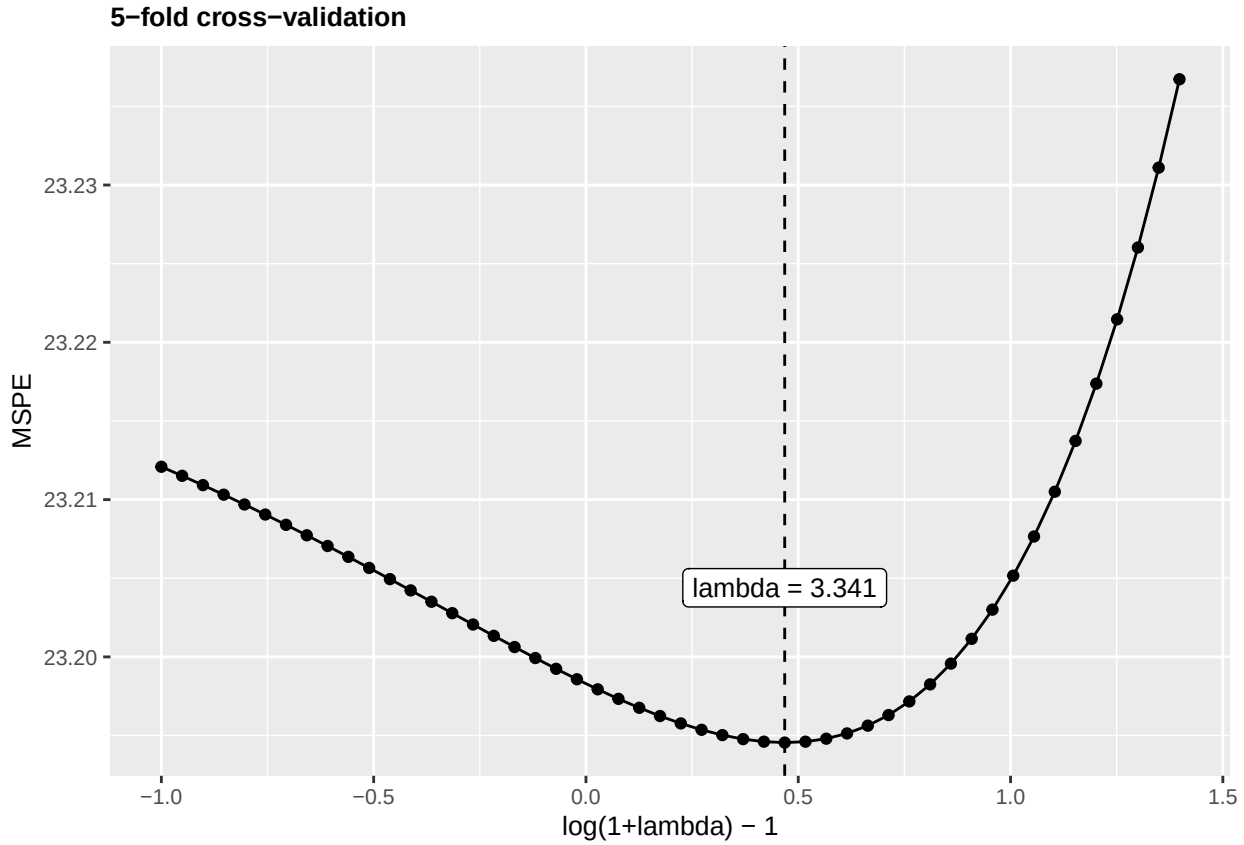

Figure 2: 5-fold CV on Boston data - MSPE in function of lambda

```
l <- which(cv.5.boston$MSPE.all == cv.5.boston$MSPE.opt)

df <- data.frame(Variable = colnames(explanatory), Coefficient = cv.5.boston$coeff.lambda[l,
    ]) %>% arrange(-abs(Coefficient))
df$Coefficient <- round(df$Coefficient, 3)
```

We obtain the coefficients in Table 2, ranked from the largest to the smallest in absolute value. We see that

the coefficients of AGE and INDUS are very small in absolute values, close to 0 while other variables have larger coefficients in absolute value. The coefficients of those two variables have been reduced the most to meet the penalization restriction. Those two variables have then the smallest predictive power for MEDV, we could probably remove them from the final model. Unlike LASSO, the Ridge regression does not make coefficients exactly zero.

The rest of the variables have larger coefficients. LSTAT, DIS, PTRATIO, NOX, TAX and CRIM all have negative coefficients which is intuitive because they correspond to characteristics for which positive values lower the attractiveness of homes and therefore are expected to have a negative impact on prices. On the contrary, RM and RAD have positive coefficients and are characteristics for which positive values increase attractiveness and prices for homes.

As all the variables are scaled, the variables with the largest coefficients have the largest impact for a change of one scaled unit in values. Then RM, DIST and LSTAT are the largest drivers of home prices.

```
# print results
kable(df, "latex", booktabs = T, caption = "\\label{tab:tab2} Coefficients from the selected Ridge regr
    escape = FALSE) %>% kable_styling(latex_options = c("hold_position"), position = "center")
```

Table 2: Coefficients from the selected Ridge regression

Variable	Coefficient
LSTAT	-3.990
DIS	-3.297
RM	2.308
RAD	2.260
PTRATIO	-2.125
NOX	-2.046
TAX	-1.718
ZN	1.174
B	0.946
CRIM	-0.735
CHAS	0.561
INDUS	-0.047
AGE	-0.033