Introduction to Machine Learning - Lab 2

Andrea Bruzzone, Araya Eamrurksiri, Oscar Pettersson, Gustav Sternelöv

Thursday, November 05, 2015

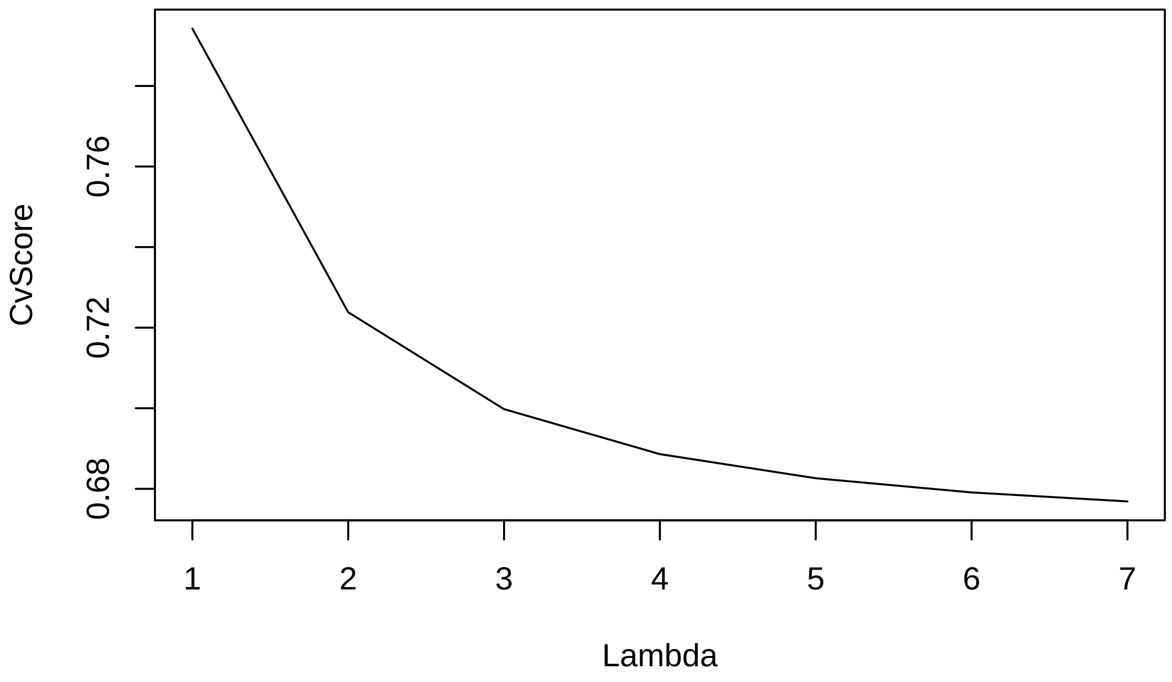
Assignment 1

1.1

The function CV() is created in R. It takes X, Y, Lambda and Nfolds as arguments and performs ridge regression by means of n-fold cross-validation. X is a matrix of predictors, Y is a vector of responses, Lambda is the shrinkage factor λ and Nfolds is n. CV() returns a score value that is the sum of squared errors.

1.2

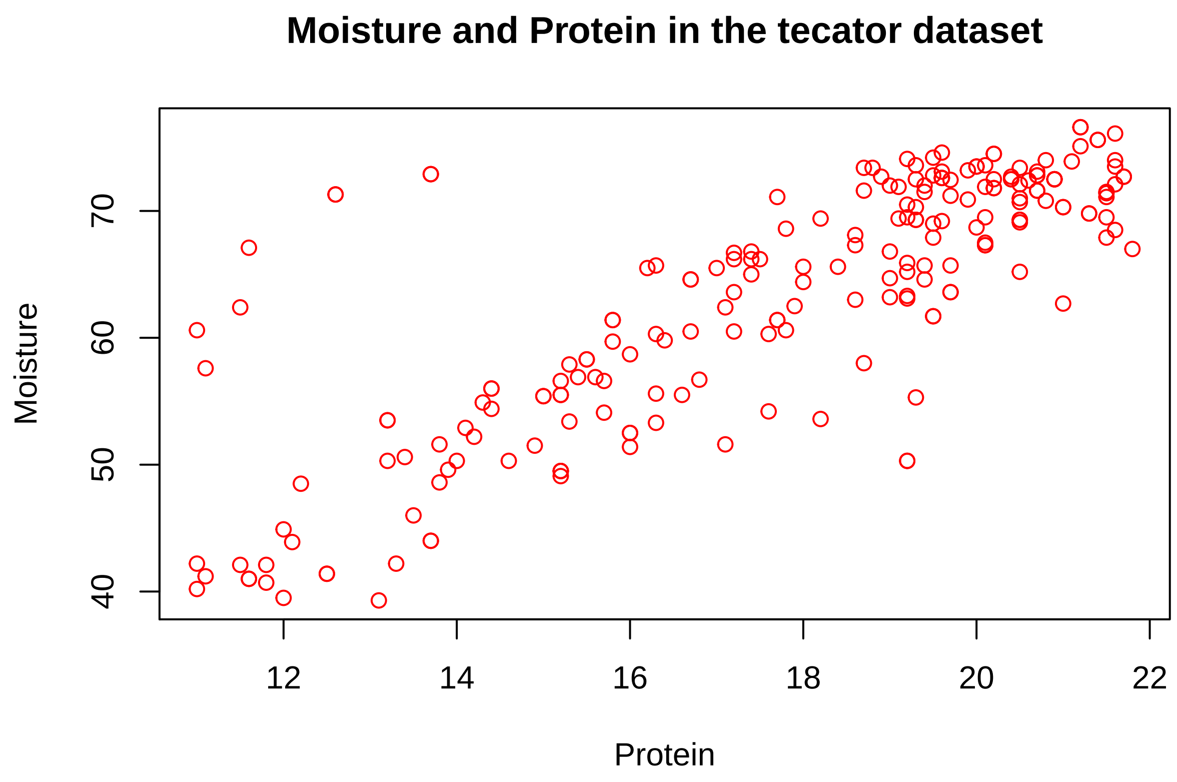
The function implemented in 1.1 is tested with the longley data set. The variables has been centered, but not scaled. Number of folds is set to 10 and the value for lambda goes from 1 to 7, by 1. The given values for the CV score is shown in the following plot below and it can be seen that the CV score decreases for higher values of lambda. The conclusion from this result is that the best model is given with lambda set to 7.



Assignment 2

2.1

Following is a plot of Moisture versus Protein:



It can be seen that even though there are some outliers in the beginning of the plot, this data might be well described by a linear regression.

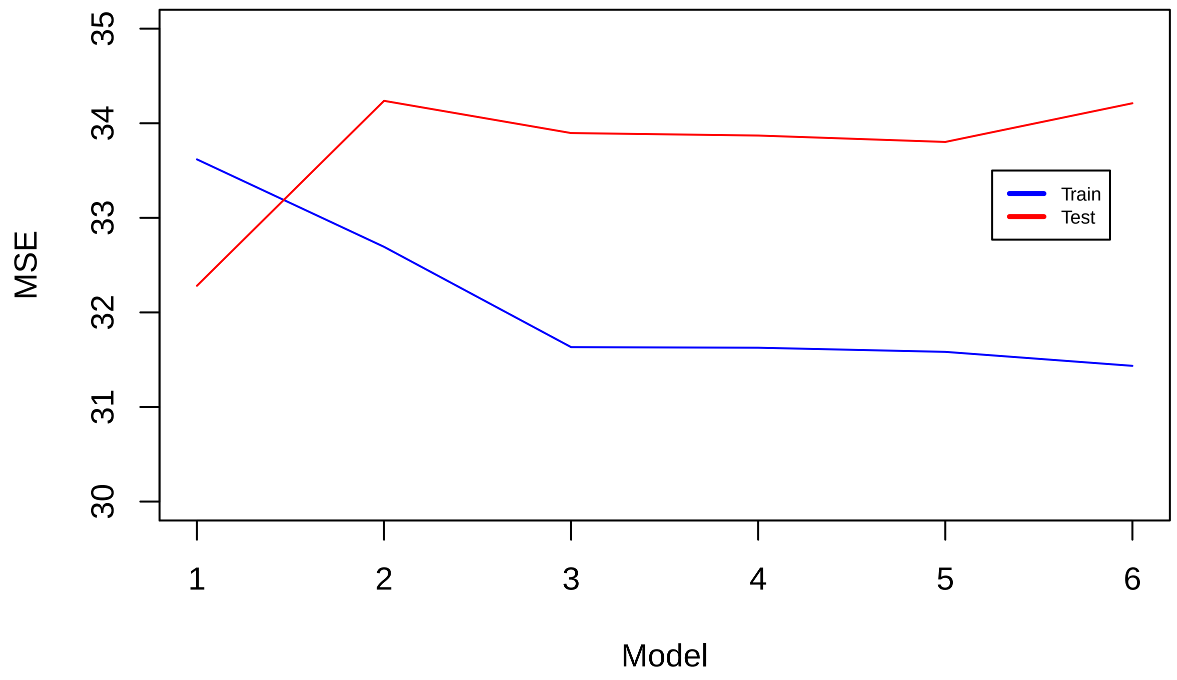
2.2

We consider a model M_i in which expected Moisture is a polynomial function of Protein, including the polynomial terms up to power i.

This model can be described by a normal model as:

$$\mathcal{N}(w_0 + \sum_i w_i x^i, \sigma^2)$$

2.3



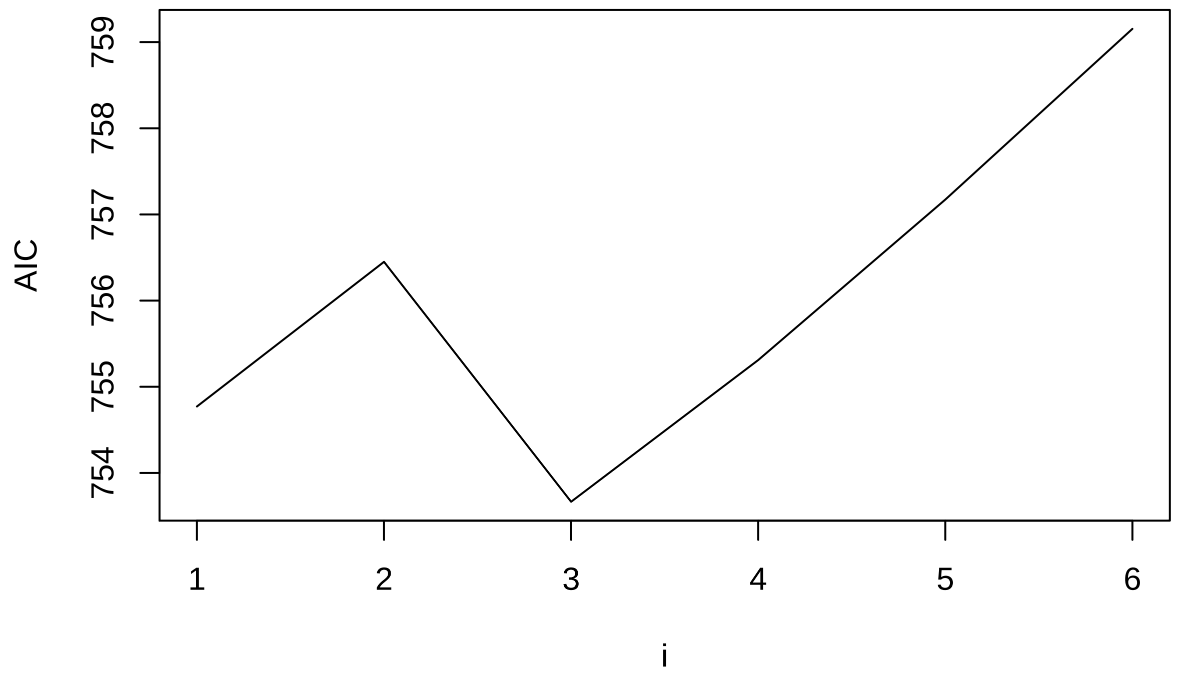
The mean square error for the train dataset appears to be high in the simple model and gradually decreases when the model gets more complex. This is due to the fact that with higher model complexity the model will fit itself closely to the train dataset. Using the model which is too overfitted to train data will result in higher mean square error for the test dataset as can be seen in the plot. Therefore, the model that gives the optimal value of mean square error for both train and test dataset needs to be found.

When consider the MSE, this plot suggests that model M_1, or linear model, is the best among all other models. This model has the lowest value of mean square error for the test dataset. Even though this model has the highest in mean square error in train dataset, it is still better than choosing other models that will overfit data.

In terms of the bias-variance tradeoff we would say that M_6 , e.g., has a lower bias (a new model made on a different sample would be very similar) but higher variance (it's a sixth order polynomial model so it "swings" very much) but M_1 does instead has a higher bias and lower variance.

2.4

Now the whole data set is used to construct the same models. AIC values is computed to evaluate the models and decide which model that is the best one. The criterion used for comparing the models is that the AIC value should be as low as possible. With that criterion in mind is it concluded that the model with polynomial terms up to a power of 3 is the preferred model.



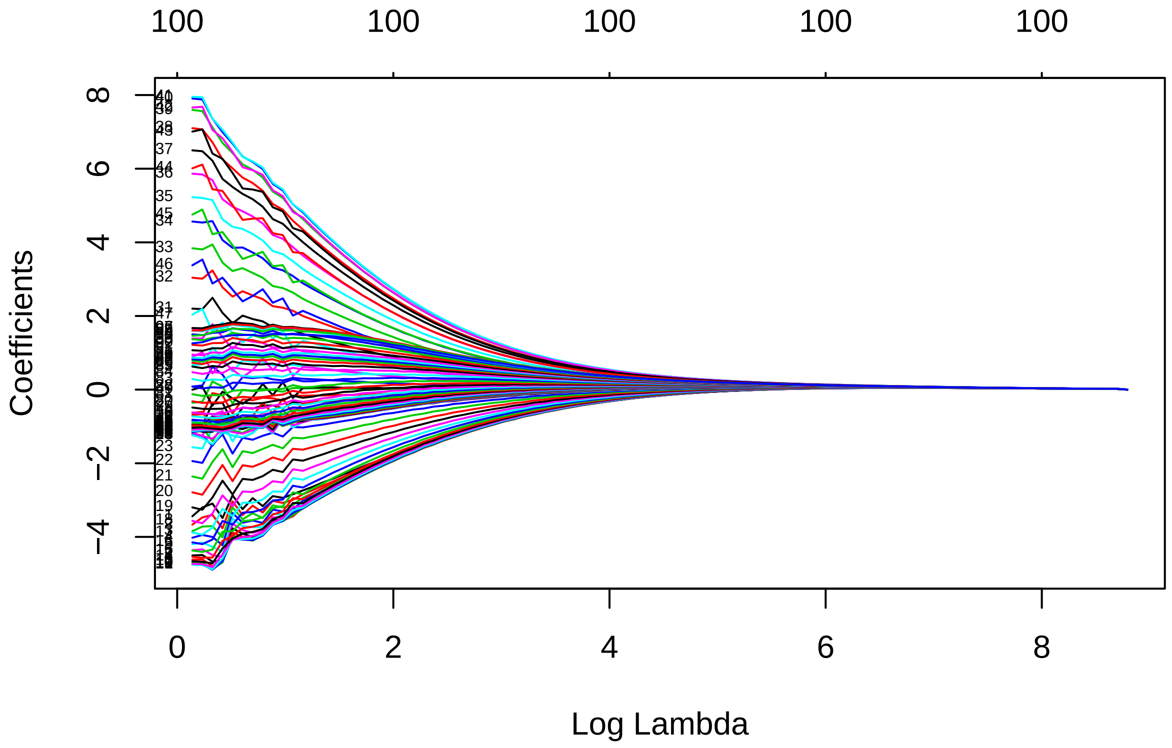
2.5

Variable selection is performed for a linear model using `stepAIC` function. `Fat` is response while `Channel1-Channel100` are predictors.

The result of `stepAIC` indicated that 63 variables from 100 variables were selected for this model.

2.6

With the same response and predictor variables as in 2.5 a ridge regression model is fitted. The difference between a ridge regression model and a linear model is the use of a penalty factor λ in the former model. The value of λ affects the coefficients in such a way that they shrinks.
How the values of the coefficients in the ridge regression model depends on the log of λ is illustrated by plotting these values.

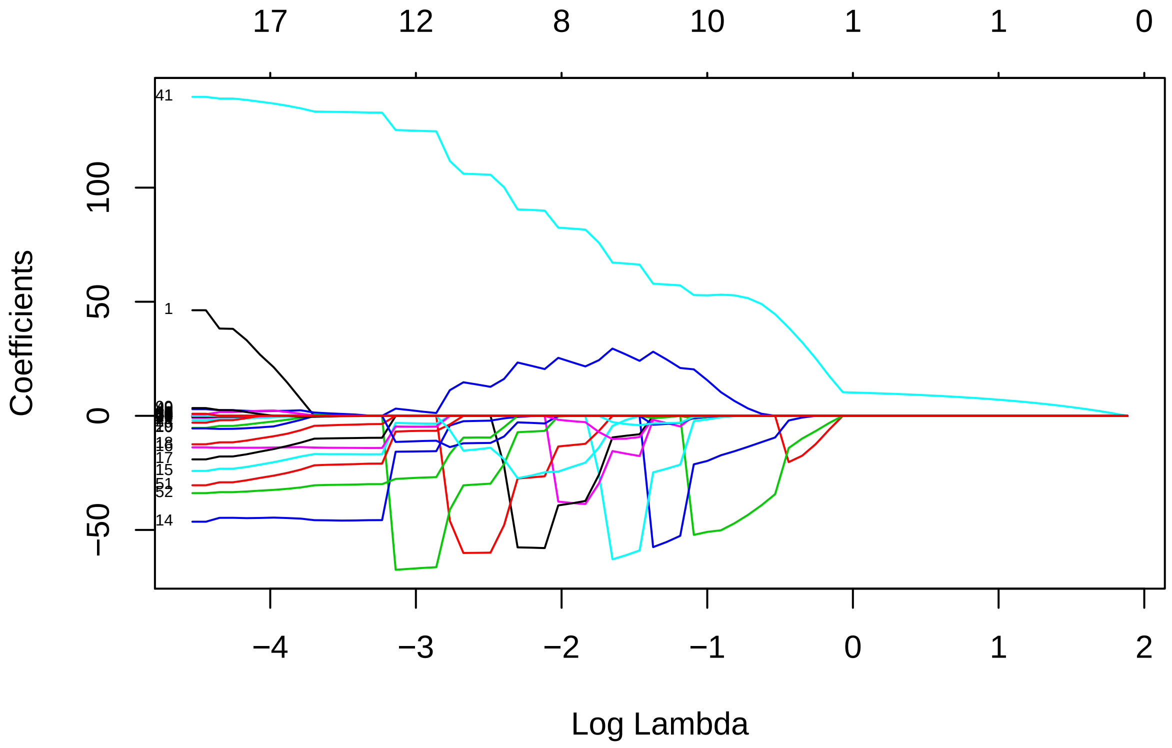


From ridge regression plot, all the coefficients are essentially zero when $log(\lambda)$ is 8. Ridge regression will always include all the variables in the model, which is 100 variables, and the value of λ selected is indicated by the vertical lines. The plot shows the whole path of variables as they shrink towards zero.

2.7

We do the same as in 2.6 but using LASSO instead of the Ridge regression. The idea of the LASSO is very similar to the Ridge regression.

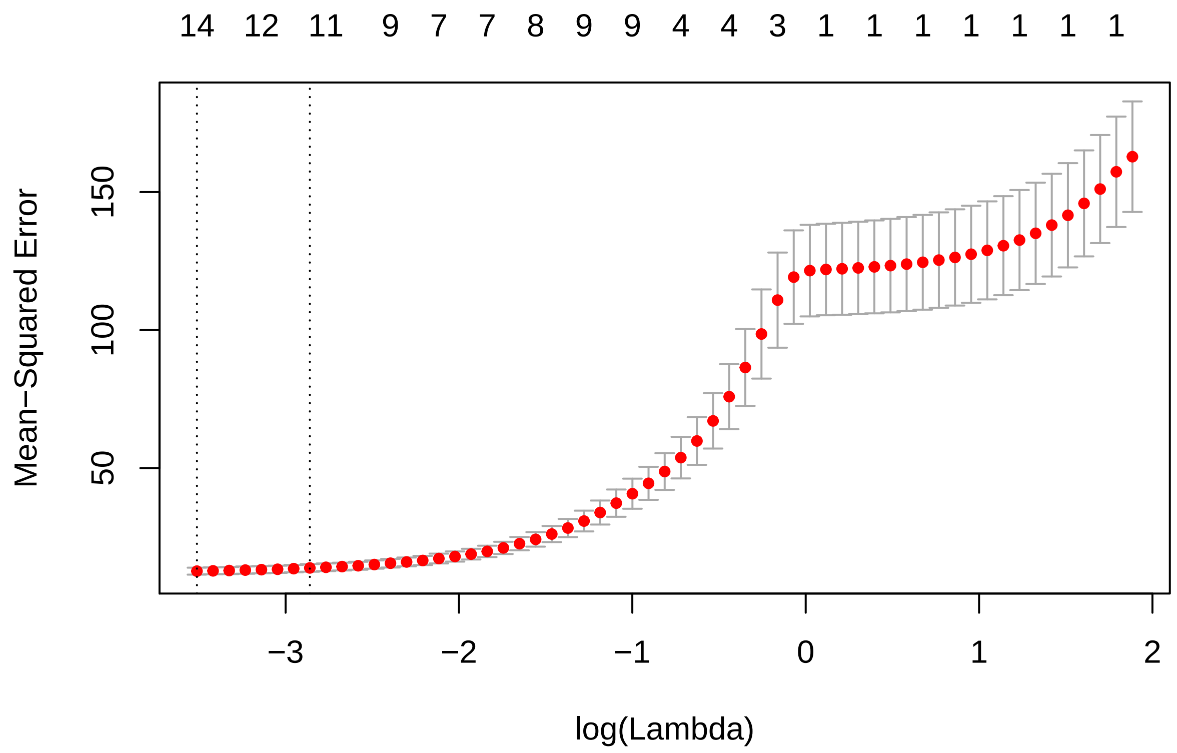
It is used the same function as before, with $\alpha = 1$, that means we use LASSO.



When comparing the plot for the ridge regression and for LASSO the characteristics for the respective methods becomes clear. For the ridge model the coefficients shrinks with higher values of lambda and slowly approaches zero. For the LASSO models coefficients it is harder to se a clear pattern in the dependence between the coefficients and log of λ. The majority of the coefficients quickly equals zero, some has a sort of wiggly curve and some of the coefficients decreases slowly for higher values of log λ.

2.8

Cross-validation is performed to find the optimal LASSO model that is to choose the appropriate lambda and coefficients.



The optimal λ which gives the minimum mean cross-validated error is eqaul to 0.02985605. Fourteen variables were chosen for a optimal LASSO model.

The plot shows the dependence of the CV score on the log of lambda. The optimal value of λ in terms of $log(\lambda)$ is -3.511368. When looking at the plot within the one standard error area , it is pretty hard to indicate the exact value of optimal $log(\lambda)$. It's pretty flat in between. By far, we see that the suggested variables for the optimal model can be varied between 11 to 14 variables with only a slighly difference in CV score. It might also be possible to use other lambda for the model and get the small CV score as the suggested lambda as well. Therefore, it is not quite reliable for the choice of lambda given above.

2.9

Forward-and-backward stepwise regression and cross-validation with LASSO give a different number of variables included in the model. Stepwise regression is a greedy algorithm so it might not find the most optimal number of variables. We also see that ridge regression and LASSO differ. The parameters shrink pretty smoothly for ridge regression while they do so in a more step-wise manner, and eventually become zero, for LASSO.

Group members contribution

1.1 is Oscar's text 1.2 is Gustav's text and plot

2.1 Oscar's plot and Lynn's text. 2.2 Andrea's. 2.3 Gustav's plot, Lynn's and Oscar´s text. 2.4 Gustav's. 2.5 Lynn's text and code. 2.6 Andrea's, Gustav's and Lynn's text and code. 2.7 Andrea's and Gustav's text and code. 2.8 Lynn's text and code. 2.9 Oscar's text.

Appendix

```r
library(MASS)

# Reads in data set Longley
data(longley)
data <- longley

longley.x <- data.matrix(longley[, 1:6])
longley.y <- longley[, "Employed"]

ridgereg_nfoldCV <- function(x, y, lambda, nfolds){
  # Create the folds and initialize CV vector
  n <- length(y)
  seques <- floor(n/nfolds)
  reps <- n%%nfolds
  groups <- rep(seq(1,nfolds, 1), seques)
  end_values <- rep(nfolds, reps)
  folds <- c(groups, end_values)
  folds <- sort(folds)

  #x <- cbind(rep(1, nrow(x)), x)
  CV <- 0
  for (i in 1:nfolds){
    testIndexes <- which(folds==i,arr.ind=TRUE)
    testData_x <- x[testIndexes, ]
    testData_y <- y[testIndexes]
    trainData_x <- x[-testIndexes, ]
    trainData_y <- y[-testIndexes]

    # find the mean value for columns in train, use that mean to scale the values
    # in test. For example, if mean in column 1 in train is 3, then use that value
    # to scale column 1 in test data.
    if (class(testData_x) == "numeric"){
      testData_x <- data.frame(t(testData_x))
    }else{
      testData_x <- data.frame(testData_x)
    }
    mean_x <- 0
    center_test <- matrix(ncol=ncol(testData_x), nrow=nrow(testData_x))
    for (j in 1:ncol(trainData_x)){
      mean_x[j] <- mean(trainData_x[,j])
      center_test[,j] <- testData_x[,j] - mean_x[j]
    }
    testData_y <- testData_y - mean(trainData_y)
    trainData_x <- scale(trainData_x, center=TRUE, scale=FALSE)
    trainData_y <- scale(trainData_y, center=TRUE, scale=FALSE)
    testData_x <- as.matrix(center_test)


    # Perform ridge regression on train data
    x_t <- t(trainData_x)
    I <- diag(ncol(trainData_x))
    BetaRidge <- solve(x_t %*% trainData_x + lambda * I) %*% x_t %*% trainData_y
    # Test regression on test data and compare with true values for test data
    y_hat_test <- testData_x %*% BetaRidge
    # Calculates CV
    CV[i] <- sum((testData_y - y_hat_test)^2)
  }
  CV_score <- (1/nfolds) * sum(CV)
  return(CV_score)
}
CvScore <- 0
for (i in 1:7){
  CvScore[i] <- ridgereg_nfoldCV(longley.x, longley.y, lambda=i, nfolds=10)
}
plot(CvScore, type="l", xlab="Lambda")
tecator <- read.csv(file = "tecator.csv", header = TRUE, sep = ";")
plot(Moisture ~ Protein, data = tecator, col="red",
     main="Moisture and Protein in the tecator dataset")
n=dim(tecator)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=tecator[id,]
test=tecator[-id,]

# Extract data sets for moisture and protein data from train and test
Protein_train <- train$Protein
Moisture_train <- train$Moisture
Protein_test <- test$Protein
Moisture_test <- test$Moisture

poly_modelTrain <- list()
MSE_train <- 0
MSE_test <- 0

for (i in 1:6){
poly_modelTrain[[i]] <-  lm(Moisture_train ~ poly(Protein_train, i, raw=TRUE))
MSE_train[i] <- 1/length(Moisture_train) * sum(poly_modelTrain[[i]]$residuals^2)
y_hat_test <- predict(lm(Moisture_train ~ poly(Protein_train, i)), data.frame(Protein_train=Protein_tes
MSE_test[i] <- 1/length(Moisture_test)  * sum((y_hat_test-Moisture_test)^2)
}


plot(1:6, MSE_train, type="l", ylim=c(30,35), col="blue", ylab="MSE", xlab="Model")
lines(1:6, MSE_test, type="l", col="red")
legend(5.25,33.5,c("Train","Test"), lty=c(1,1),
  lwd=c(2.5,2.5),col=c("blue","red"),  cex=0.6)
AIC <- 0
for (i in 1:6){
  poly_modelTrain[[i]] <-  lm(tecator$Moisture ~ poly(tecator$Protein, i, raw=TRUE))
  n <- length(tecator$Moisture)
  RSS <- sum(poly_modelTrain[[i]]$residuals^2)
  AIC[i] <- 2*(i+1) + n * log(RSS/n)
}
plot(AIC, type="l", xlab="i")
```

```r
data.fat <- tecator[,2:102]
MFull <- lm(Fat~., data=data.fat)
## step <- stepAIC(MFull, direction="both")
## step$coefficients
## num.coef <- length(step$coefficients) - 1 #minus one for interception
library(glmnet)
vars <- data.frame(tecator[,2:102])
# Create matrix with x variables
mat_vars <- as.matrix(vars[,1:100])
# y variable
mat_y <- as.matrix(vars[,101])

ridge_mod <- glmnet(mat_vars, mat_y, alpha=0, family = "gaussian")
plot(ridge_mod, xvar="lambda",label=TRUE)
model2 <- glmnet(mat_vars, mat_y, alpha = 1, family = "gaussian")
plot(model2 , xvar = "lambda", label = TRUE)

set.seed(12345)
lasso_cv <- cv.glmnet(mat_vars, mat_y, alpha=1, family = "gaussian")
plot(lasso_cv)
##
```