library(mlbench)

#load the dataset

data("BostonHousing")

head(BostonHousing)

########################### Forward Selection #######################

lm = lm(medv~., data=BostonHousing)

null.lm = lm(medv~1, data=BostonHousing)# then do FS

f<-step(null.lm, scope=formula(lm), direction="forward")

summary(f)

########################### Backward Selection #######################

lm = lm(medv~., data=BostonHousing)

b<-step(lm, scope=formula(lm), direction="backward")

summary(b)

########################### Stepwise Regression

lm = lm(medv~., data=BostonHousing)

null.lm = lm(medv~1, data=BostonHousing)# then do FS

step(null.lm, scope=formula(lm), direction="both")

########################### Ridge Regression

x <- as.matrix(BostonHousing[,c(1:3,5:13)])

y <- as.matrix(BostonHousing[,14])

library("glmnet")

fit <- glmnet(x, y, family="gaussian", alpha=0, lambda=0.01)

fit

# summarize the fit

summary(fit)

coef(fit)

# make predictions

predictions <- predict(fit, x, type="link")

# summarize accuracy

mse <- mean((y - predictions)^2)

print(mse)

## Finding optimal lambda

cv.out <- cv.glmnet(x, y, alpha = 0)

bestlam <- cv.out$lambda.min

fit <- glmnet(x, y, family="gaussian", alpha=0, lambda=bestlam)

fit

# summarize the fit

summary(fit)

coef(fit)

# make predictions

predictions <- predict(fit, x, type="link")

# summarize accuracy

mse <- mean((y - predictions)^2)

print(mse)

########################### Lasso Regression

x <- as.matrix(BostonHousing[,c(1:3,5:13)])

y <- as.matrix(BostonHousing[,14])

library("glmnet")

fit <- glmnet(x, y, family="gaussian", alpha=1, lambda=0.01)

fit

# summarize the fit

summary(fit)

coef(fit)

# make predictions

predictions <- predict(fit, x, type="link")

# summarize accuracy

mse <- mean((y - predictions)^2)

print(mse)

## Finding optimal lambda

cv.out <- cv.glmnet(x, y, alpha = 1)

bestlam <- cv.out$lambda.min

fit <- glmnet(x, y, family="gaussian", alpha=1, lambda=bestlam)

fit

# summarize the fit

summary(fit)

coef(fit)

# make predictions

predictions <- predict(fit, x, type="link")

# summarize accuracy

mse <- mean((y - predictions)^2)

print(mse)

############################## Least Angle Regression

library(lars)

x <- as.matrix(BostonHousing[,c(1:3,5:13)])

y <- as.matrix(BostonHousing[,14])

# fit model

fit <- lars(x, y, type="lasso")

# summarize the fit

summary(fit)

#result

fit

# select a step with a minimum error

best\_step <- fit$df[which.min(fit$RSS)]

best\_step

# make predictions

predictions <- predict(fit, x, s=best\_step, type="fit")$fit

# summarize accuracy

mse <- mean((y - predictions)^2)

print(mse)

# load the package

library(glmnet)

#########################Multivariate Adaptive Regression Splines (MARS) is a non-parametric

#regression method that models multiple nonlinearities in data using

#hinge functions (functions with a kink in them)

# load the package

#install.packages('earth')

library(earth)

# fit model

fit <- earth(medv~.,data=BostonHousing)

fit

# summarize the fit

summary(fit)

# summarize the importance of input variables

evimp(fit)

# make predictions

predictions <- predict(fit, BostonHousing)

# summarize accuracy

mse <- mean((BostonHousing$medv - predictions)^2)

print(mse)

#Principal Component Regression (PCR) creates a linear regression model

#using the outputs of a Principal Component Analysis (PCA) to estimate

#the coefficients of the model. PCR is useful when the data has highly

#correlated predictors.

# load the package

install.packages('pls')

library(pls)

# fit model

fit <- pcr(medv~., data=BostonHousing, validation="CV")

fit

# summarize the fit

summary(fit)

# make predictions

predictions <- predict(fit, BostonHousing, ncomp=10)

# summarize accuracy

mse <- mean((BostonHousing$medv - predictions)^2)

print(mse)

#PLS is appropriate for data with highly-correlated predictors.

#Partial Least Squares (PLS) Regression creates a linear model of the

#data in a transformed projection of problem space

# load the package

library(pls)

# fit model

fit <- plsr(medv~., data=BostonHousing, validation="CV")

fit

# summarize the fit

summary(fit)

# make predictions

predictions <- predict(fit, BostonHousing, ncomp=6)

# summarize accuracy

mse <- mean((BostonHousing$medv - predictions)^2)

print(mse)

# Robust regression is particularly resourceful when there are no

#compelling reasons to exclude outliers in your data

library(MASS)

rmod <- rlm (medv ~ ., psi = psi.huber, data = BostonHousing)

rmod

# Robust Regrression

summary(rmod) # model summary

# make predictions

predictions<-predict(rmod, BostonHousing) # apply model to predict response on test data

# summarize accuracy

mse <- mean((BostonHousing$medv - predictions)^2)

print(mse)