Week 5 Exercises

Adeline Casali

2023-11-21

# Question 9

Load packages

library(ISLR2)  
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-8

library(pls)

##   
## Attaching package: 'pls'

## The following object is masked from 'package:stats':  
##   
## loadings

library(leaps)

1. Split the dataset randomly with 50% in a training set and 50% in a test set.

sum(is.na(College$Apps))

## [1] 0

set.seed(123)  
training\_indices <- sample(1:nrow(College), 0.5 \* nrow(College))  
train <- College[training\_indices, ]  
test <- College[-training\_indices, ]

1. Linear model predicting the number of applications based on all variables in the College dataset. The MSE was calculated as 1373995.

lm\_model <- lm(Apps ~ ., data = train)  
predictions <- predict(lm\_model, newdata = test)  
cat("Test Error (MSE):", mean((test$Apps - predictions)^2), "\n")

## Test Error (MSE): 1373995

1. Here is the model using ridge regression. The MSE is 2090325.

set.seed(123)  
train\_mat <- model.matrix(Apps~., data = train)  
test\_mat <- model.matrix(Apps~., data = test)  
cv.out <- cv.glmnet(train\_mat, train$Apps, alpha = 0)  
bestlam <- cv.out$lambda.min  
cat("Chosen lambda value:", bestlam, "\n")

## Chosen lambda value: 301.9518

ridge\_mod <- glmnet(train\_mat, train$Apps, alpha = 0)  
ridge\_pred <- predict(ridge\_mod, s = bestlam, newx = test\_mat)  
cat("Test Error (MSE):", mean((ridge\_pred - test$Apps)^2), "\n")

## Test Error (MSE): 2090325

1. Here is the model using the lasso method. The MSE is 1391022.

set.seed(123)  
cv.out2 <- cv.glmnet(train\_mat, train$Apps, alpha = 1)  
bestlam2 <- cv.out2$lambda.min  
cat("Chosen lambda value:", bestlam2, "\n")

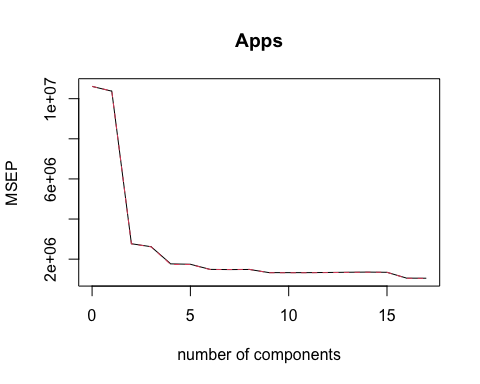
## Chosen lambda value: 15.02821

lasso\_mod <- glmnet(train\_mat, train$Apps, alpha = 1)  
lasso\_pred <- predict(lasso\_mod, s = bestlam2, newx = test\_mat)  
cat("Test Error (MSE):", mean((lasso\_pred - test$Apps)^2), "\n")

## Test Error (MSE): 1391022

1. Here is the model using PCR. The MSE is 2887472.

set.seed(123)  
pcr\_fit <- pcr(Apps~., data = train, scale = TRUE, validation = "CV")  
validationplot(pcr\_fit, val.type = "MSEP")



# 10 variables seems to have the lowest MSEP. It doesn't get much lower after that   
summary(pcr\_fit)

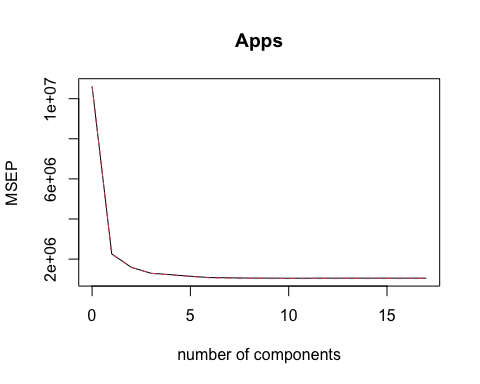
## Data: X dimension: 388 17   
## Y dimension: 388 1  
## Fit method: svdpc  
## Number of components considered: 17  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 3258 3221 1664 1619 1329 1321 1221  
## adjCV 3258 3222 1663 1617 1317 1317 1218  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 1217 1219 1153 1152 1153 1156 1161  
## adjCV 1218 1221 1150 1150 1150 1153 1158  
## 14 comps 15 comps 16 comps 17 comps  
## CV 1164 1160 1027 1024  
## adjCV 1161 1160 1022 1020  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 32.969 59.69 66.68 72.25 77.31 81.82 85.18 88.34  
## Apps 3.259 74.43 76.35 84.40 84.42 86.69 86.79 87.01  
## 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps  
## X 91.32 93.61 95.48 97.05 98.11 98.90 99.40  
## Apps 88.38 88.39 88.47 88.49 88.50 88.51 88.74  
## 16 comps 17 comps  
## X 99.81 100.00  
## Apps 91.48 91.58

pcr\_pred <- predict(pcr\_fit, test, ncomp = 10)  
cat("Test Error (MSE):", mean((pcr\_pred - test$Apps)^2), "\n")

## Test Error (MSE): 2887472

1. Here is the model using PLS. The MSE is 1389525.

set.seed(123)  
pls\_fit = plsr(Apps~., data = train, scale = TRUE, validation = "CV")  
validationplot(pls\_fit, val.type = "MSEP")



# 8 variables seems to have the lowest MSEP. It doesn't get much lower after that   
summary(pls\_fit)

## Data: X dimension: 388 17   
## Y dimension: 388 1  
## Fit method: kernelpls  
## Number of components considered: 17  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 3258 1502 1260 1137 1105 1070 1039  
## adjCV 3258 1501 1260 1135 1102 1067 1033  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 1033 1026 1025 1023 1024 1025 1025  
## adjCV 1028 1021 1021 1019 1019 1020 1020  
## 14 comps 15 comps 16 comps 17 comps  
## CV 1025 1025 1024 1024  
## adjCV 1020 1020 1020 1020  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 26.97 43.83 65.16 69.79 73.63 76.18 79.97 81.98  
## Apps 79.33 85.98 88.76 89.67 90.59 91.43 91.54 91.56  
## 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps  
## X 84.18 86.85 90.18 91.77 93.55 95.40 97.35  
## Apps 91.57 91.57 91.57 91.58 91.58 91.58 91.58  
## 16 comps 17 comps  
## X 99.06 100.00  
## Apps 91.58 91.58

pls\_pred = predict(pls\_fit, test, ncomp = 8)  
cat("Test Error (MSE):", mean((pls\_pred - test$Apps)^2), "\n")

## Test Error (MSE): 1389525

1. The method with the lowest MSE is the Linear Model, followed closely by PLS and Lasso. Ridge Regression and PCR had the highest MSE values, but were still relatively accurate. The R-squared value of the linear model is 0.9289, indicating that 92.89% of the variance in Apps can be explained using this model.

methods <- c("Linear Model", "Ridge Regression", "Lasso", "PCR", "PLS")  
mse\_values <- c(1373995, 2090325, 1391022, 2887472, 1389525)  
results\_df <- data.frame(Method = methods, `Test Error (MSE)` = mse\_values)  
results\_df

## Method Test.Error..MSE.  
## 1 Linear Model 1373995  
## 2 Ridge Regression 2090325  
## 3 Lasso 1391022  
## 4 PCR 2887472  
## 5 PLS 1389525

TSS <- sum((mean(test$Apps) - test$Apps)^2)  
TSR <- sum((predictions - test$Apps)^2)  
cat("R-squared for the best model (linear):", 1 - (TSR)/(TSS), "\n")

## R-squared for the best model (linear): 0.9289176

# Question 11

1. I created a linear model, ridge regression model, lasso model, PCR model, and PLS model.

# Creating a training and test set  
sum(is.na(Boston$crim))

## [1] 0

set.seed(123)  
training\_indices2 <- sample(1:nrow(Boston), 0.5 \* nrow(Boston))  
train2 <- Boston[training\_indices2, ]  
test2 <- Boston[-training\_indices2, ]  
  
# Creating and evaluating a linear model  
lm\_model2 <- lm(crim ~ ., data = train2)  
predictions2 <- predict(lm\_model2, newdata = test2)  
cat("Test Error (MSE):", mean((test2$crim - predictions2)^2), "\n")

## Test Error (MSE): 51.51577

TSS2 <- sum((mean(test2$crim) - test2$crim)^2)  
TSR\_lm <- sum((predictions2 - test2$crim)^2)  
cat("R-squared for the linear model:", 1 - (TSR\_lm)/(TSS2), "\n")

## R-squared for the linear model: 0.4001315

# Creating and evaluating a Ridge Regression model  
set.seed(123)  
train\_mat2 <- model.matrix(crim~., data = train2)  
test\_mat2 <- model.matrix(crim~., data = test2)  
cv.out3 <- cv.glmnet(train\_mat2, train2$crim, alpha = 0)  
bestlam3 <- cv.out3$lambda.min  
ridge\_mod2 <- glmnet(train\_mat2, train2$crim, alpha = 0)  
ridge\_pred2 <- predict(ridge\_mod2, s = bestlam3, newx = test\_mat2)  
cat("Test Error (MSE):", mean((ridge\_pred2 - test2$crim)^2), "\n")

## Test Error (MSE): 52.23721

TSR\_rr <- sum((ridge\_pred2 - test2$crim)^2)  
cat("R-squared for the Ridge Regression model:", 1 - (TSR\_rr)/(TSS2), "\n")

## R-squared for the Ridge Regression model: 0.3917309

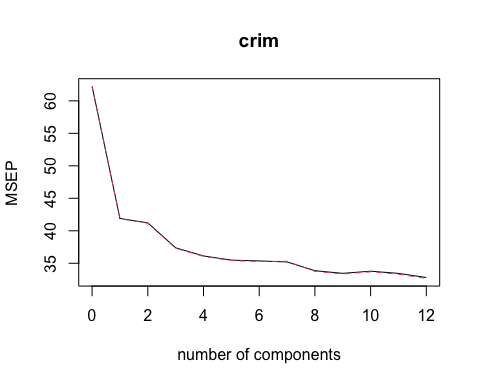
# Creating and evaluating Lasso model  
set.seed(123)  
cv.out4 <- cv.glmnet(train\_mat2, train2$crim, alpha = 1)  
bestlam4 <- cv.out4$lambda.min  
lasso\_mod2 <- glmnet(train\_mat2, train2$crim, alpha = 1)  
lasso\_pred2 <- predict(lasso\_mod2, s = bestlam4, newx = test\_mat2)  
cat("Test Error (MSE):", mean((lasso\_pred2 - test2$crim)^2), "\n")

## Test Error (MSE): 51.78056

TSR\_l <- sum((lasso\_pred2 - test2$crim)^2)  
cat("R-squared for the Lasso model:", 1 - (TSR\_l)/(TSS2), "\n")

## R-squared for the Lasso model: 0.3970483

# Creating and evaluating PCR model  
set.seed(123)  
pcr\_fit2 <- pcr(crim~., data = train2, scale = TRUE, validation = "CV")  
validationplot(pcr\_fit2, val.type = "MSEP")



summary(pcr\_fit2)

## Data: X dimension: 253 12   
## Y dimension: 253 1  
## Fit method: svdpc  
## Number of components considered: 12  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 7.888 6.473 6.420 6.114 6.011 5.957 5.946  
## adjCV 7.888 6.469 6.416 6.107 6.006 5.951 5.939  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps  
## CV 5.933 5.819 5.784 5.812 5.783 5.728  
## adjCV 5.942 5.810 5.774 5.804 5.771 5.716  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 49.83 64.02 73.28 80.29 86.65 90.33 92.91 95.23  
## crim 33.53 35.33 41.42 43.17 44.38 45.08 45.36 48.29  
## 9 comps 10 comps 11 comps 12 comps  
## X 97.02 98.40 99.5 100.00  
## crim 49.01 49.02 50.0 50.96

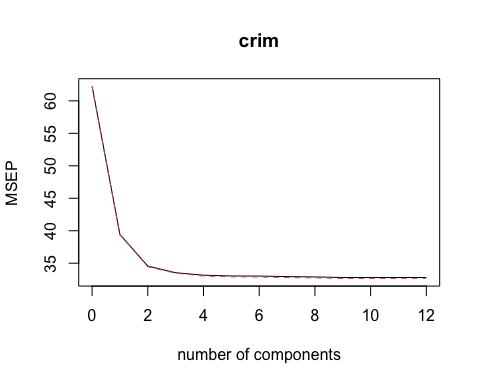
pcr\_pred2 <- predict(pcr\_fit2, test2, ncomp = 9)  
cat("Test Error (MSE):", mean((pcr\_pred2 - test2$crim)^2), "\n")

## Test Error (MSE): 53.82982

TSR\_pcr <- sum((pcr\_pred2 - test2$crim)^2)  
cat("R-squared for the PCR model:", 1 - (TSR\_pcr)/(TSS2), "\n")

## R-squared for the PCR model: 0.3731859

# Creating and evaluating PLS model  
set.seed(123)  
pls\_fit2 <- plsr(crim~., data = train2, scale = TRUE, validation = "CV")  
validationplot(pls\_fit2, val.type = "MSEP")



summary(pls\_fit2)

## Data: X dimension: 253 12   
## Y dimension: 253 1  
## Fit method: kernelpls  
## Number of components considered: 12  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 7.888 6.278 5.878 5.791 5.758 5.748 5.747  
## adjCV 7.888 6.275 5.871 5.785 5.747 5.737 5.734  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps  
## CV 5.739 5.735 5.727 5.727 5.727 5.728  
## adjCV 5.727 5.722 5.715 5.715 5.716 5.716  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 49.34 59.81 69.60 75.68 81.59 84.42 88.24 92.75  
## crim 37.82 46.73 48.74 50.13 50.50 50.79 50.90 50.93  
## 9 comps 10 comps 11 comps 12 comps  
## X 94.52 96.23 98.61 100.00  
## crim 50.96 50.96 50.96 50.96

pls\_pred2 <- predict(pls\_fit2, test2, ncomp = 6)  
cat("Test Error (MSE):", mean((pls\_pred2 - test2$crim)^2), "\n")

## Test Error (MSE): 51.90707

TSR\_pls <- sum((pls\_pred2 - test2$crim)^2)  
cat("R-squared for the PLS model:", 1 - (TSR\_pls)/(TSS2), "\n")

## R-squared for the PLS model: 0.3955752

1. Using test set error to validate the models, the model with the best fit (lowest MSE and highest R-squared value) is the linear model with an MSE of 51.5 and an r-squared value of 0.40. This is not perfect, as only 40% of the variation in the data can be explained by the model, but it was the best out of the 5 models explored.

model\_results <- data.frame(  
 Model = c("Linear Model", "Ridge Regression", "Lasso Model", "PCR Model", "PLS Model"),  
 MSE = c(mean((test2$crim - predictions2)^2),  
 mean((ridge\_pred2 - test2$crim)^2),  
 mean((lasso\_pred2 - test2$crim)^2),  
 mean((pcr\_pred2 - test2$crim)^2),  
 mean((pls\_pred2 - test2$crim)^2)),  
 R\_squared = c(1 - (TSR\_lm)/(TSS2),  
 1 - (TSR\_rr)/(TSS2),  
 1 - (TSR\_l)/(TSS2),  
 1 - (TSR\_pcr)/(TSS2),  
 1 - (TSR\_pls)/(TSS2))  
)  
model\_results

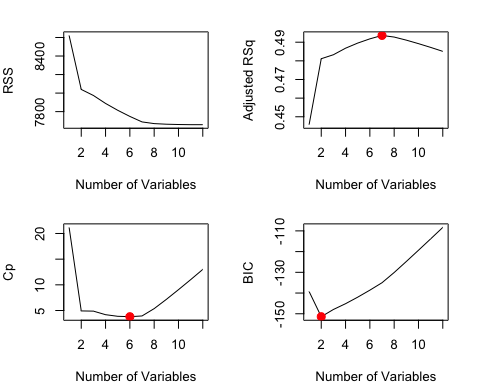
## Model MSE R\_squared  
## 1 Linear Model 51.51577 0.4001315  
## 2 Ridge Regression 52.23721 0.3917309  
## 3 Lasso Model 51.78056 0.3970483  
## 4 PCR Model 53.82982 0.3731859  
## 5 PLS Model 51.90707 0.3955752

1. This model does involve all of the predictors, but it can be created without them. For example, using best subset selection, a linear model with only 2 predictors, rad and lstat, selected using best subset selection. This results in a better MSE of 53.77. However, it does result in a worse r-squared value of 0.37, which is to be expected as adding in predictors increases the r-squared value, regardless of how useful they actually are in predicting.

regfit\_full <- regsubsets(crim ~ ., data = train2, nvmax = 13)  
reg\_summary <- summary(regfit\_full)  
  
# Display the results  
print(reg\_summary)

## Subset selection object  
## Call: regsubsets.formula(crim ~ ., data = train2, nvmax = 13)  
## 12 Variables (and intercept)  
## Forced in Forced out  
## zn FALSE FALSE  
## indus FALSE FALSE  
## chas FALSE FALSE  
## nox FALSE FALSE  
## rm FALSE FALSE  
## age FALSE FALSE  
## dis FALSE FALSE  
## rad FALSE FALSE  
## tax FALSE FALSE  
## ptratio FALSE FALSE  
## lstat FALSE FALSE  
## medv FALSE FALSE  
## 1 subsets of each size up to 12  
## Selection Algorithm: exhaustive  
## zn indus chas nox rm age dis rad tax ptratio lstat medv  
## 1 ( 1 ) " " " " " " " " " " " " " " "\*" " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " " " " "\*" " " " " "\*" " "   
## 3 ( 1 ) "\*" " " " " " " " " " " " " "\*" " " " " "\*" " "   
## 4 ( 1 ) "\*" " " " " " " " " " " "\*" "\*" " " " " " " "\*"   
## 5 ( 1 ) "\*" " " " " " " " " " " "\*" "\*" " " " " "\*" "\*"   
## 6 ( 1 ) "\*" " " " " "\*" " " " " "\*" "\*" " " " " "\*" "\*"   
## 7 ( 1 ) "\*" " " " " "\*" " " " " "\*" "\*" " " "\*" "\*" "\*"   
## 8 ( 1 ) "\*" "\*" " " "\*" " " " " "\*" "\*" " " "\*" "\*" "\*"   
## 9 ( 1 ) "\*" "\*" "\*" "\*" " " " " "\*" "\*" " " "\*" "\*" "\*"   
## 10 ( 1 ) "\*" "\*" "\*" "\*" " " " " "\*" "\*" "\*" "\*" "\*" "\*"   
## 11 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*"   
## 12 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

# Plot RSS, adjusted R2, Cp, and BIC  
par(mar = c(4, 4, 2, 2), mfrow = c(2, 2))  
plot(reg\_summary$rss, xlab = "Number of Variables", ylab = "RSS", type = "l")  
plot(reg\_summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq", type = "l")  
  
# Identify the model with the largest adjusted R2 and plot a red dot  
best\_adjr2\_index <- which.max(reg\_summary$adjr2)  
points(best\_adjr2\_index, reg\_summary$adjr2[best\_adjr2\_index], col = "red", cex = 2, pch = 20)  
  
# Plot Cp and identify the model with the smallest Cp  
plot(reg\_summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")  
best\_cp\_index <- which.min(reg\_summary$cp)  
points(best\_cp\_index, reg\_summary$cp[best\_cp\_index], col = "red", cex = 2, pch = 20)  
  
# Plot BIC and identify the model with the smallest BIC  
plot(reg\_summary$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")  
best\_bic\_index <- which.min(reg\_summary$bic)  
points(best\_bic\_index, reg\_summary$bic[best\_bic\_index], col = "red", cex = 2, pch = 20)



best\_bic\_index <- which.min(reg\_summary$bic)  
  
# Extract coefficients for the best model  
best\_model\_coeffs <- coef(regfit\_full, id = best\_bic\_index)  
  
# Exclude intercept  
best\_model\_coeffs <- best\_model\_coeffs[-1]  
  
# Display the number of variables and the variable names  
num\_variables <- sum(best\_model\_coeffs != 0)  
variable\_names <- names(best\_model\_coeffs[best\_model\_coeffs != 0])  
  
cat("Number of variables in the best model:", num\_variables, "\n")

## Number of variables in the best model: 2

cat("Variable names in the best model:", paste(variable\_names, collapse = ", "), "\n")

## Variable names in the best model: rad, lstat

# Create the linear model with only the selected variables  
lm\_model3 <- lm(crim ~ rad + lstat, data = train2)  
predictions3 <- predict(lm\_model3, newdata = test2)  
cat("Test Error (MSE):", mean((test2$crim - predictions3)^2), "\n")

## Test Error (MSE): 53.76839

TSR\_lm2 <- sum((predictions3 - test2$crim)^2)  
cat("R-squared for the new linear model:", 1 - (TSR\_lm2)/(TSS2), "\n")

## R-squared for the new linear model: 0.3739012