PSET4_ParthDesai

Parth Desai

2023-02-24

Question 1

Part 1.1

```
dat <- read.csv("vote92plus.csv")
set.seed(5000)
k <- sample(1:nrow(dat), round(nrow(dat)*2/3), replace = FALSE)
train.dat <- dat[k,]
test.dat <- dat[-k,]
```

The first line of code reads the csv file and saves is as 'dat'. The second line of code sets random number generator to the seed value of 5000. The purpose of this line is for consistency when 'random' sampling as R uses an algorithm for 'randomness' so having a set seed allows for the same 'randomness' to be replicated. The third line of code samples without replacement the 'dat' dataset. It from 1 to the number of rows in 'dat' and samples 2/3rds of the rows in 'dat' rounded to the nearest integer. The third line is for sampling the data for later purposes. The sample is saved as 'k'. The fourth line of code subsets 'dat' with the sampled 'k' and saves it as 'train.dat'. This is used for training the models. The fifth line of code subsets 'dat' with all of the non-'k' rows. This is used for testing the models.

Part 1.2

```
model_A <- lm(train.dat$clintondis ~ vote + dem + rep + female + persfinance + natlecon,
              data = train.dat)

mse_train_A <- mean(resid(model_A)^2)
prediction_A <- predict(model_A, newdata = test.dat)
mse_test_A <- mean((test.dat$clintondis - prediction_A)^2)


train_and_test_A <- c(mse_train_A, mse_test_A)
train_and_test_A
```

```
## [1] 11.34638 13.02046
```

Part 1.3

```
model_B <- lm(train.dat$clintondis ~ vote + dem + rep + female + persfinance + natlecon +
                fake1 + fake2 + fake3 + fake4 + fake5, data = train.dat)

mse_train_B <- mean(resid(model_B)^2)
prediction_B <- predict(model_B, newdata = test.dat)
mse_test_B <- mean((test.dat$clintondis - prediction_B)^2)

train_and_test_B <- c(mse_train_B, mse_test_B)
train_and_test_B
```

```
## [1] 11.31480 13.05915
```

Part 1.4

```
train.dat$fake1_square <- '^'(train.dat$fake1, 2)
train.dat$fake2_square <- '^'(train.dat$fake2, 2)
train.dat$fake3_square <- '^'(train.dat$fake3, 2)
train.dat$fake4_square <- '^'(train.dat$fake4, 2)
train.dat$fake5_square <- '^'(train.dat$fake5, 2)

test.dat$fake1_square <- '^'(test.dat$fake1, 2)
test.dat$fake2_square <- '^'(test.dat$fake2, 2)
test.dat$fake3_square <- '^'(test.dat$fake3, 2)
test.dat$fake4_square <- '^'(test.dat$fake4, 2)
test.dat$fake5_square <- '^'(test.dat$fake5, 2)

model_C <- lm(train.dat$clintondis ~ vote + dem + rep + female + persfinance + natlecon +
                fake1 + fake2 + fake3 + fake4 + fake5 +
                fake1_square + fake2_square + fake3_square + fake4_square +
                fake5_square, data = train.dat)

mse_train_C <- mean(resid(model_C)^2)
prediction_C <- predict(model_C, newdata = test.dat)
mse_test_C <- mean((test.dat$clintondis - prediction_C)^2)

train_and_test_C <- c(mse_train_C, mse_test_C)
train_and_test_C
```

```
## [1] 11.18850 13.22988
```

Part 1.5

```
train.dat$fake1_cube <- '^'(train.dat$fake1, 3)
train.dat$fake2_cube <- '^'(train.dat$fake2, 3)
train.dat$fake3_cube <- '^'(train.dat$fake3, 3)
train.dat$fake4_cube <- '^'(train.dat$fake4, 3)
```

```r
train.dat$fake5_cube <- '^'(train.dat$fake5, 3)

test.dat$fake1_cube <- '^'(test.dat$fake1, 3)
test.dat$fake2_cube <- '^'(test.dat$fake2, 3)
test.dat$fake3_cube <- '^'(test.dat$fake3, 3)
test.dat$fake4_cube <- '^'(test.dat$fake4, 3)
test.dat$fake5_cube <- '^'(test.dat$fake5, 3)

model_D <- lm(train.dat$clintondis ~ vote + dem + rep + female + persfinance + natlecon +
                fake1 + fake2 + fake3 + fake4 + fake5 +
                fake1_square + fake2_square + fake3_square + fake4_square +
                fake5_square + fake1_cube + fake2_cube + fake3_cube +
                fake4_cube + fake5_cube, data = train.dat)

mse_train_D <- mean(resid(model_D)^2)
prediction_D <- predict(model_D, newdata = test.dat)
mse_test_D <- mean((test.dat$clintondis - prediction_D)^2)

train_and_test_D <- c(mse_train_D, mse_test_D)
train_and_test_D
```

```
## [1] 11.12317 13.33001
```

Part 1.6

```r
mse_train_table <- data.frame("Model A trainmse" = mse_train_A, "Model B trainmse" = mse_train_B,
                              "Model C trainmse" = mse_train_C, "Model D train_mse" = mse_train_D)
mse_train_table
```

```
##   Model.A.trainmse Model.B.trainmse Model.C.trainmse Model.D.train_mse
## 1         11.34638          11.3148          11.1885          11.12317
```

Part 1.7

```r
mse_test_table <- data.frame("Model A testmse" = mse_test_A, "Model B testmse" = mse_test_B,
                             "Model C testmse" = mse_test_C, "Model D testmse" = mse_test_D)
mse_test_table
```

```
##   Model.A.testmse Model.B.testmse Model.C.testmse Model.D.testmse
## 1        13.02046        13.05915        13.22988        13.33001
```

Part 1.8

The training MSE is lower than the test MSE for each model. In addition, it can be that as more of the fake variables were added in the model, the training MSE decreased. The opposite happened in the test MSE, where as more fake variables were added the MSE increased.

Part 1.9

```
set.seed(202)
k_1 <- sample(1:nrow(dat), round(nrow(dat)*1/3), replace = FALSE)
train.dat_1 <- dat[k_1,]
test.dat_1 <- dat[-k_1,]

model_A_1 = lm(train.dat_1$clintondis ~ vote + dem + rep + female + persfinance + natlecon,
               data = train.dat_1)
mse_train_A_1 = mean(resid(model_A_1)^2)
prediction_A_1 = predict(model_A_1, newdata = test.dat_1)
mse_test_A_1 = mean((test.dat_1$clintondis - prediction_A_1)^2)
train_and_test_A_1 = c(mse_test_A_1, mse_train_A_1)


model_B_1 = lm(train.dat_1$clintondis ~ vote + dem + rep + female + persfinance + natlecon
               + fake1 + fake2 + fake3 + fake4 + fake5, data = train.dat_1)
mse_train_B_1 = mean(resid(model_B_1)^2)
prediction_B_1 = predict(model_B_1, newdata = test.dat_1)
mse_test_B_1 = mean((test.dat_1$clintondis - prediction_B_1)^2)
train_and_test_B_1 = c(mse_test_B_1, mse_train_B_1)

train.dat_1$fake1_square <- '^'(train.dat_1$fake1, 2)
train.dat_1$fake2_square <- '^'(train.dat_1$fake2, 2)
train.dat_1$fake3_square <- '^'(train.dat_1$fake3, 2)
train.dat_1$fake4_square <- '^'(train.dat_1$fake4, 2)
train.dat_1$fake5_square <- '^'(train.dat_1$fake5, 2)

test.dat_1$fake1_square <- '^'(test.dat_1$fake1, 2)
test.dat_1$fake2_square <- '^'(test.dat_1$fake2, 2)
test.dat_1$fake3_square <- '^'(test.dat_1$fake3, 2)
test.dat_1$fake4_square <- '^'(test.dat_1$fake4, 2)
test.dat_1$fake5_square <- '^'(test.dat_1$fake5, 2)

model_C_1 = lm(train.dat_1$clintondis ~ vote + dem + rep + female + persfinance + natlecon
               + fake1 + fake2 + fake3 + fake4 + fake5
               + fake1_square + fake2_square + fake3_square + fake4_square + fake5_square,
               data = train.dat_1)
mse_train_C_1 = mean(resid(model_C_1)^2)
prediction_C_1 = predict(model_C_1, newdata = test.dat_1)
mse_test_C_1 = mean((test.dat_1$clintondis - prediction_C_1)^2)
train_and_test_C_1 = c(mse_test_C_1, mse_train_C_1)

train.dat_1$fake1_cube <- '^'(train.dat_1$fake1, 3)
train.dat_1$fake2_cube <- '^'(train.dat_1$fake2, 3)
train.dat_1$fake3_cube <- '^'(train.dat_1$fake3, 3)
train.dat_1$fake4_cube <- '^'(train.dat_1$fake4, 3)
train.dat_1$fake5_cube <- '^'(train.dat_1$fake5, 3)

test.dat_1$fake1_cube <- '^'(test.dat_1$fake1, 3)
test.dat_1$fake2_cube <- '^'(test.dat_1$fake2, 3)
test.dat_1$fake3_cube <- '^'(test.dat_1$fake3, 3)
test.dat_1$fake4_cube <- '^'(test.dat_1$fake4, 3)
```

```
test.dat_1$fake5_cube <- '^'(test.dat_1$fake5, 3)

model_D_1 = lm(clintondis ~ vote + dem + rep + female + persfinance + natlecon
               + fake1 + fake2 + fake3 + fake4 + fake5
               + fake1_square + fake2_square + fake3_square + fake4_square + fake5_square
               + fake1_cube + fake2_cube + fake3_cube + fake4_cube + fake5_cube,
               data = train.dat_1)
mse_train_D_1 = mean(resid(model_D_1)^2)
prediction_D_1 = predict(model_D_1, newdata = test.dat_1)
mse_test_D_1 = mean((test.dat_1$clintondis - prediction_D_1)^2)
train_and_test_D_1 = c(mse_test_D_1, mse_train_D_1)

mse_train_table_1 <- data.frame("Model A_1 trainmse" = mse_train_A_1,
                                "Model B_1 trainmse" = mse_train_B_1,
                                "Model C_1 trainmse" = mse_train_C_1,
                                "Model D_1 train_mse" = mse_train_D_1)
mse_train_table_1
```

```
##   Model.A_1.trainmse Model.B_1.trainmse Model.C_1.trainmse Model.D_1.train_mse
## 1           12.41982           12.19242           11.95335            11.75048
```

```
mse_test_table_1 <- data.frame("Model A_1 testmse" = mse_test_A_1,
                               "Model B_1 testmse" = mse_test_B_1,
                               "Model C_1 testmse" = mse_test_C_1,
                               "Model D_1 testmse" = mse_test_D_1)
mse_test_table_1
```

```
##   Model.A_1.testmse Model.B_1.testmse Model.C_1.testmse Model.D_1.testmse
## 1          11.75088          12.13872           12.5985          15.83061
```

The results are different for the new split as there is a different proportion assigned to the training and test dataset are, thus the error would also vary accordingly as there is now less training values for which the model can use to learn the correlations in the data. However, the same intrinsic trends of MSE decreasing/increasing as mentioned in Part 1.8 are present here as well. #Question 2

Part 2.1

The lasso, relative to ordinary least squares regression, is (1) more flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance. The reason for this is the lasso subsets variables, and by subsetting and not including all variables in the final model, introduces bias. In addition, by not including all variables, there is more variance in the data so there is a U-shaped graph formed where bias decreases and then increases as variance responds inversely.

Part 2.2

a) As we increase Lambda from 0, the training mean squared error will (iii) steadily increase. Lambda represents the magnitude of the penalty term, meant to correct overfitting, so when Lambda = 0; it is an ordinary least squares regression, thus the MSE will be the smallest it would be and the MSE would increase proportionally with Lambda.

b) As we increase Lambda from 0, the test mean squared error will (ii) decrease initially, and then eventually start increasing in an inverted U shape. As Lambda increases, the model becomes simpler with fewer predictors. There is a point where the optimization of the predictors will allow for few to be used and high levels of accuracy. However, as the model continually increases, the error will begin to increase as the predictors are not enough to understand the data properly.

c) As we increase Lambda from 0, the variance will (iv) steadily decrease. Since variance is proportional to the squared value of the regression coefficients, as Lambda increases, the coefficients decrease as well as their squared values. Thus, the variance decreases as well.

d) As we increase Lambda from 0, the squared bias will (iii) steadily increase. As Lambda increases, there is more underfitting, and thus, bias. Following from that, the squared values for bias will also increase.

e) As we increase Lambda from 0, the irreducible error will, as its name suggests, (v) remain constant. The fact that the error is irreducible means it is not dependent on any coefficients or factors; thus, Lambda has no effect on it.