STT 810

Homework 6

Aditya Jain

2022-12-01

Contents

Question 1

1. A linear regression model prediction is in the form of y = 2.18 x1 – 4.56x2. The model is built using 18 data points. The root mean square error of the residuals is 0.856.

a What is the E(y | (x1,x2) = (2, -3))?

```
funky <-  function(x1,x2) (2.18*x1 - 4.56*x2)
funky(2,-3)
```

```
## [1] 18.04
```

b What is the probability that y > 20, given that (x1,x2) = (2, -3)?

```
p_greater_20 <- 1 - pt((20 - 18.04)/0.856, 16)
p_greater_20
```

```
## [1] 0.01798231
```

Question 2

2. Take the Boston dataset, available in D2L. This data has information about different neighborhoods in Boston, and we will use it to predict the median housing price for the neighborhoods. Here is an explanation of the variables:

CRIM: Per capita crime rate by town

ZN: Proportion of residential land zoned for lots over 25,000 sq. ft

INDUS: Proportion of non-retail business acres per town

CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

NOX: Nitric oxide concentration (parts per 10 million)

RM: Average number of rooms per dwelling

AGE: Proportion of owner-occupied units built prior to 1940

DIS: Weighted distances to five Boston employment centers

RAD: Index of accessibility to radial highways

TAX: Full-value property tax rate per $10,000

PTRATIO: Pupil-teacher ratio by town

B: $1000(Bk — 0.63)^2$, where Bk is the proportion of [people of African American descent] by town

LSTAT: Percentage of lower status of the population

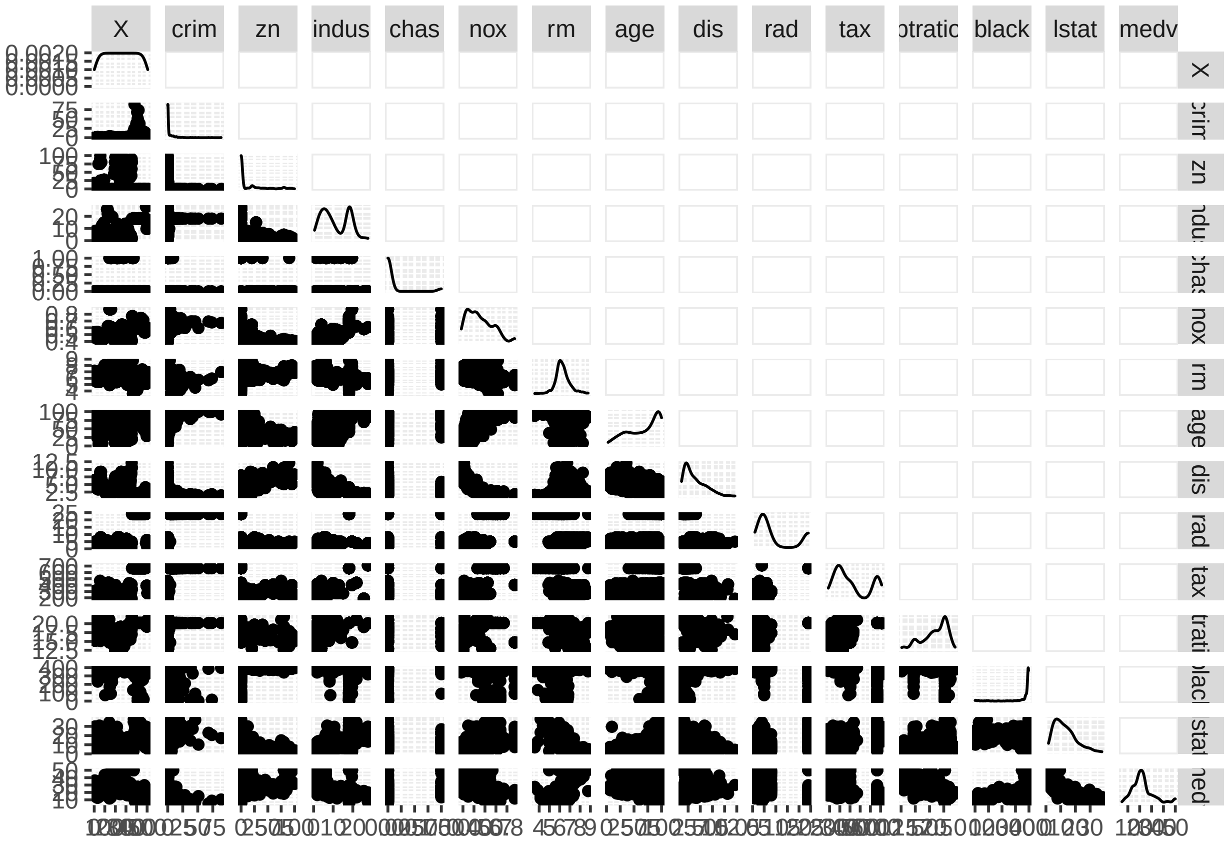
MEDV: Median value of owner-occupied homes in $1000s

a. Create a pairs plot with ggpairs for the data.

```
## Warning: package 'GGally' was built under R version 4.2.2

## Loading required package: ggplot2

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```



b. Which other variable correlates the strongest with medv? Note that strongest mean largest absolute value, whether it's positive or negative.

Ans: LSTAT: Percentage of lower status of the population has the highest correlation with the medv and equal to -0.737

c. Build a simple linear regression model with that variable as the x, with

1. Constructing the normal equations ATAx = ATb, and solving for the coefficient vector x.

```r
# setting up normal equations
A = cbind(1,data_boston$lstat)
b = data_boston$medv
x = solve(t(A) %*% A) %*% t(A) %*% b
x
```

```
##              [,1]
## [1,] 34.5538409
## [2,] -0.9500494
```

2. Using lm. Do the coefficients agree?

```r
real_mod <- (lm(data = data_boston, medv ~ lstat))
summary(real_mod)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = data_boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

The coefficients agree in both the methods.

d. Next, build a linear regression model with the 2 other variables in addition that correlate most strongly with medv, using lm (so 3 variables total).

```r
real_mod <- (lm(data = data_boston, medv ~ lstat + rm + ptratio))
summary(real_mod)
```

```
##
## Call:
## lm(formula = medv ~ lstat + rm + ptratio, data = data_boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.4871  -3.1047  -0.7976   1.8129  29.6559
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 18.56711    3.91320   4.745 2.73e-06 ***
## lstat       -0.57181    0.04223 -13.540  < 2e-16 ***
## rm           4.51542    0.42587  10.603  < 2e-16 ***
## ptratio     -0.93072    0.11765  -7.911 1.64e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.229 on 502 degrees of freedom
## Multiple R-squared:  0.6786, Adjusted R-squared:  0.6767
## F-statistic: 353.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

1. How do the adjusted R-squared values compare between the models?

Ans. The adjusted R-squared value increased when we added more features into our model. This, shows that we were able to cover more variance in our data by adding these features.

2. Comment on the significance of the coefficients

Ans. The value of the coefficient associated with column rm (average room per dwelling) is comparatively higher than the other two features. As, increasing even one room in a household drastically changes its price, hence the coefficient for column rm is higher as a change in rm will have bigger impact on median price of the house as compared to other two.

3. Are the coefficients in the correct direction? Be sure to explain.

Ans. As we can see the coefficients of lstat and ptratio are negative as for obvious reasons if % of lower status of people and pupil-teacher ratio increases the median house prices should go down. Similarly if average number of rooms per dwelling increases the expected value of the house should also increase. Thus, the direction of the coefficients seems reasonable.

e. Now, re-do the regression from (c), except this time

- Split the data into training and testing datasets (70/30 split)
- Build the model on the training dataset
- Use the model to predict values for the test dataset
- How do the results compare, in terms of RMSE?

```r
split_pct <- 0.7
n <- length(data_boston$medv)*split_pct # train size
row_samp <- sample(1:length(data_boston$medv), n, replace = FALSE)
train <- data_boston[row_samp,]
test <- data_boston[-row_samp,]
boston_train_mod <- lm(data = data_boston, medv ~ lstat)
test_pred <- predict(boston_train_mod,test)
test_error <- test$medv - test_pred
rmse_train <- sqrt(mean(boston_train_mod$residuals^2))
rmse_test <- sqrt(mean(test_error^2))
rmse_train
```

```
## [1] 6.203464
```

```r
rmse_test
```

```
## [1] 6.679663
```

f. For the linear regression in (b) re-do the linear regressions in the following ways:

- Construct a linear model with the variable with the strongest correlation, like in 4(b), but this time do the minimization of RMSE directly, with the optim function. Check to make sure the result matches what you got in the previous homework.

```
coeff <- rep(0,2)
lin_reg <- function(coeff) sum((data_boston$medv - (coeff[1] + coeff[2] * data_boston$lstat))^2)
RMSE_fit <- optim(c(100, -1), lin_reg, data_boston)
RMSE_fit$par
```

```
## [1] 34.5517953 -0.9501778
```

- Next construct a linear model of the same form, with mean absolute error as the loss function, and then use optim to find the fit. Compare the model with what you did in the previous part.

```
coeff_MAE <- rep(0,2)
MAE_reg <- function(coeff_MAE) sum(abs(data_boston$medv -
                  (coeff_MAE[1] + coeff_MAE[2] * data_boston$lstat)))
MAE_fit <- optim(c(100, -1), MAE_reg, data_boston)
MAE_fit$par
```

```
## [1] 31.4605486 -0.8253151
```

g. Use maximum likelihood to construct the linear regression model from (b). Do the coefficients agree?

```
ll_coeff <- c(0,0,0)
LLoptim <- function(ll_coeff)
      -1*sum(log(dnorm(data_boston$medv
      - (ll_coeff[1] + ll_coeff[2]*data_boston$lstat), 0, ll_coeff[3])))
LL_fit <- optim(c(100,-1, 2000), LLoptim, data_boston)
```

```
## Warning in dnorm(data_boston$medv - (ll_coeff[1] + ll_coeff[2] *
## data_boston$lstat), : NaNs produced

## Warning in dnorm(data_boston$medv - (ll_coeff[1] + ll_coeff[2] *
## data_boston$lstat), : NaNs produced

## Warning in dnorm(data_boston$medv - (ll_coeff[1] + ll_coeff[2] *
## data_boston$lstat), : NaNs produced
```

```
LL_fit
```

```
## $par
## [1] 34.5539498 -0.9499646  6.2009538
##
## $value
## [1] 1641.488
##
## $counts
## function gradient
##      210       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

h. For the model in (c), what are the 95% confidence intervals for the parameters, according to the t-values?

```r
# for intercept
CI_lstat <- 34.55384 + 0.56263*qt(c(0.025,0.975),length(data_boston$lstat)-1)
CI_lstat
```

```
## [1] 33.44846 35.65922
```

```r
# for coefficient
CI_medv <- -0.95005 + 0.03873*qt(c(0.025,0.975),length(data_boston$medv)-1)
CI_medv
```

```
## [1] -1.0261418 -0.8739582
```

i. For the model in (c), what are the 95% confidence intervals for the parameters, using

- regular bootstrapping, and

```r
# regular bootstrap
coeff1 <- rep(0, 100)
coeff2 <- rep(0, 100)

for(i in 1:100){
  n <- length(data_boston$lstat)
  row_samp <- sample(1:n, n, replace = TRUE)
  data_samp <- data_boston[row_samp,]
  temp_mod <- lm(data = data_samp, medv ~ lstat)
  coeff1[i] <- temp_mod$coefficients[1]
  coeff2[i] <- temp_mod$coefficients[2]
}
quantile(coeff1, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 32.71828 35.84993
```

```r
quantile(coeff2, c(0.025, 0.975))
```

```
##       2.5%      97.5%
## -1.0466494 -0.8348616
```

Bayesian bootstrapping?

```r
# Bayesian bootstrap
coeff1 <- rep(0, 100)
coeff2 <- rep(0, 100)
weight <- rep(0,length(data_boston$lstat))
n <- length(data_boston$lstat)
for(i in 1:100){
  weight <- rdirichlet(1, rep(1,n))
  row_samp <- sample(1:n, n, prob = weight, replace = TRUE)
  data_samp <- data_boston[row_samp,]
  temp_mod <- lm(data = data_samp, medv ~ lstat)
  coeff1[i] <- temp_mod$coefficients[1]
  coeff2[i] <- temp_mod$coefficients[2]
}
quantile(coeff1, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 32.41962 36.63809
```

```r
quantile(coeff2, c(0.025, 0.975))
```

```
##       2.5%      97.5%
## -1.0723543 -0.8155308
```