DSO 530: Multiple Linear Regression

Abbass Al Sharif

Multiple Linear Regression

We will conitue working with the Boston dataset which is part of the `MASS` package. It recordes the median value of houses for 506 neighborhoods around Boston. Now, we want to use more predictors to predict the response variables `medv`. But first, let's split the data:

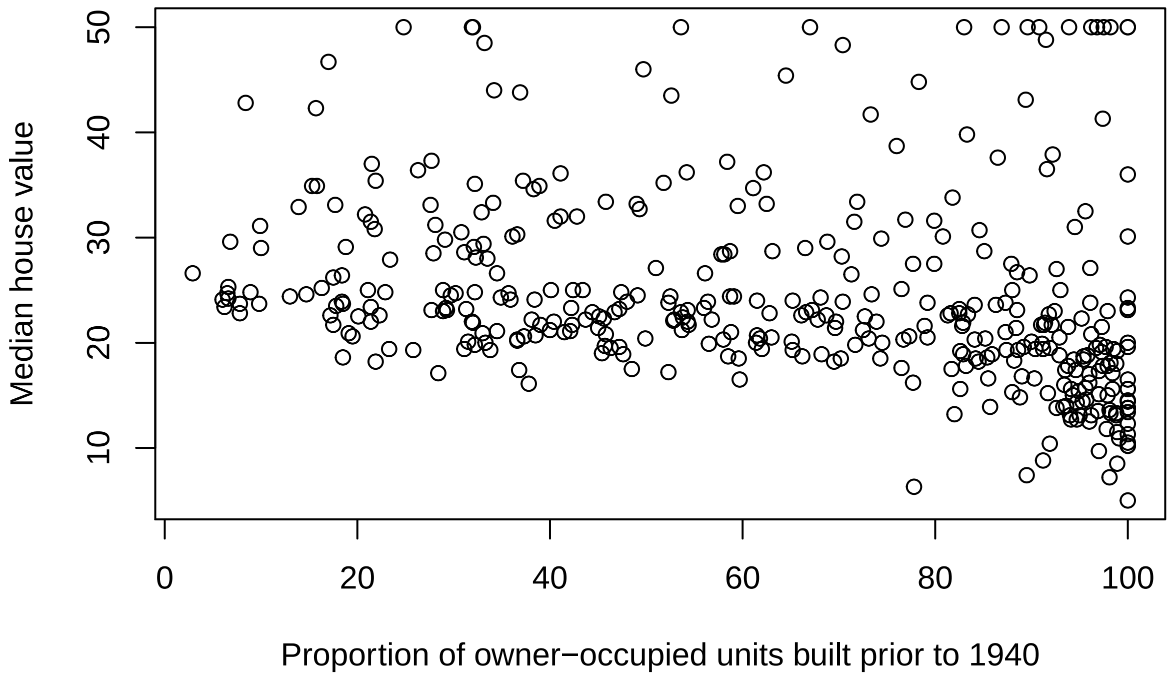
```
# load MASS package
library(MASS)

#split the data by using the first 400 observations as the training data and the remaining as the testi
train = 1:400
test = -train

# we are keeping all variables in training and testing data
training_data = Boston[train,]
testing_data = Boston[test,]
```

Create a linear regression model using both `lstat` and `age` as predictors, but before we should check if there is a linear relationship between `mdev` and `age`.

```
#use cor() function to find correlations between variables
cor(training_data$age, training_data$medv)
```

```
## [1] -0.2782
```

```
#plot both varaiables
plot(training_data$age,
     training_data$medv,
     xlab = "Proportion of owner-occupied units built prior to 1940",
     ylab = "Median house value")
```



The relationship between age and median house value is negative, and eventhough the correlation is not that high, but we can see some linear tred in the plot. So there is no need to transform it like we did for `lstat` (Refer to Simple Linear Regeression Document)

```
#use the training dataset to tarin the model. lm() allows you to specify the data you want to use.
model = lm(medv~ log(lstat) + age, data = training_data)
summary(model)
```

```
##
## Call:
## lm(formula = medv ~ log(lstat) + age, data = training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.526  -3.394  -0.839   2.750  22.945
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  51.6021     1.0364   49.79  < 2e-16 ***
## log(lstat)  -14.3390     0.5390  -26.60  < 2e-16 ***
## age           0.0778     0.0111    7.03  8.8e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.29 on 397 degrees of freedom
## Multiple R-squared:  0.668,  Adjusted R-squared:  0.667
## F-statistic:  400 on 2 and 397 DF,  p-value: <2e-16
```

Nice, the first happy thing to notice is that R^2 has increased to 66.84% compared to 62.7% for the simple linear regression. Thus, 66.84% of the model variation is being explained by the predictors `log(lstat)` and `age`. Both of the predictor variables are significant to the model (p-value <0.05), and the model as a whole is significant (look at the p-value associated with the F-statistics. It is less than the 0.05 significance level).

Now, let's use all predictor variables in our model. We can use the `.` as a short cut instead of writing down all of them:

```
model = lm(medv~., data = training_data)
summary(model)
```

```
##
## Call:
## lm(formula = medv ~ ., data = training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.564  -2.694  -0.615   1.695  25.033
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  28.67260    6.15170    4.66  4.3e-06 ***
## crim         -0.19125    0.05404   -3.54  0.00045 ***
## zn            0.04423    0.01411    3.13  0.00185 **
## indus         0.05522    0.06553    0.84  0.39994
## chas          1.71631    0.89117    1.93  0.05485 .
## nox         -14.99572    4.55759   -3.29  0.00109 **
## rm            4.88773    0.48495   10.08  < 2e-16 ***
## age           0.00261    0.01433    0.18  0.85562
## dis          -1.29481    0.21172   -6.12  2.4e-09 ***
## rad           0.48479    0.08735    5.55  5.3e-08 ***
## tax          -0.01540    0.00445   -3.46  0.00059 ***
## ptratio      -0.80880    0.14008   -5.77  1.6e-08 ***
## black        -0.00129    0.00654   -0.20  0.84338
## lstat        -0.51795    0.05951   -8.70  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.81 on 386 degrees of freedom
## Multiple R-squared:  0.734,  Adjusted R-squared:  0.725
## F-statistic: 81.9 on 13 and 386 DF,  p-value: <2e-16
```

Of course, we expect R^2 to increase. Now we see that 73.4% of the variation in the model is explained by all explanatory variables. Wait, didn't we transform `lstat`? Alright, let's substract `lstat` from the model, and then add `log(lstat)`.

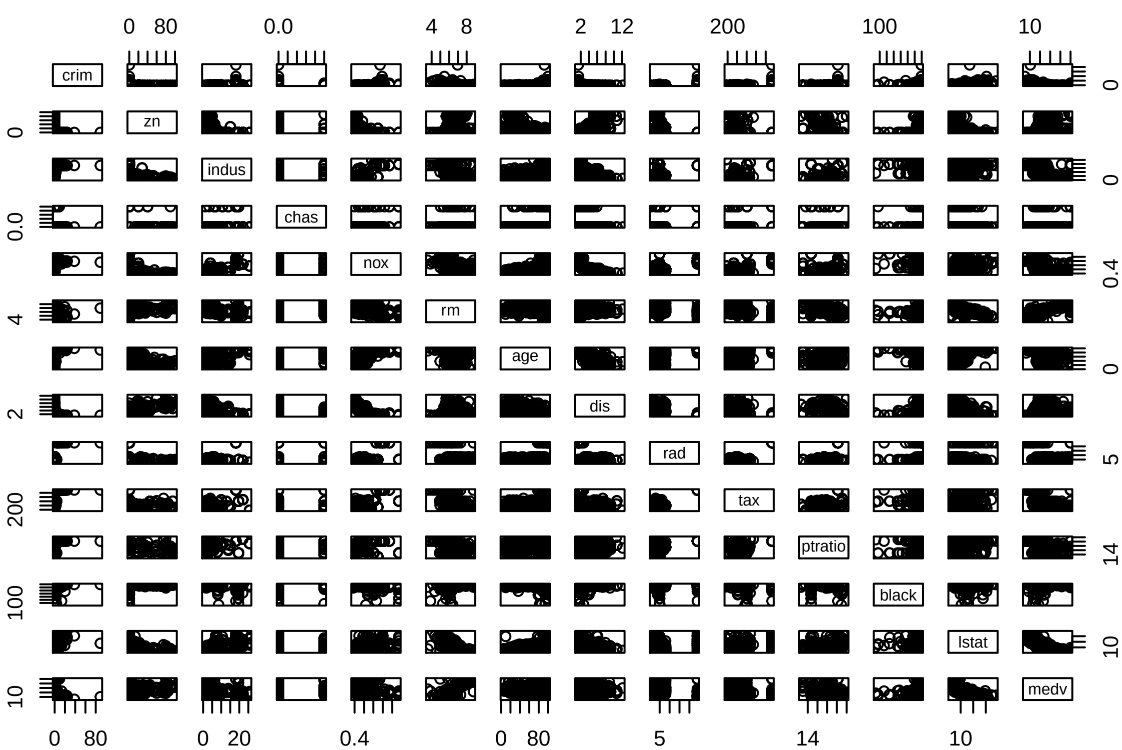
```
model = lm(medv~.-lstat+log(lstat), data = training_data)
summary(model)
```

```
##
## Call:
## lm(formula = medv ~ . - lstat + log(lstat), data = training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.266  -2.670  -0.268   1.880  23.561
##
## Coefficients:
```

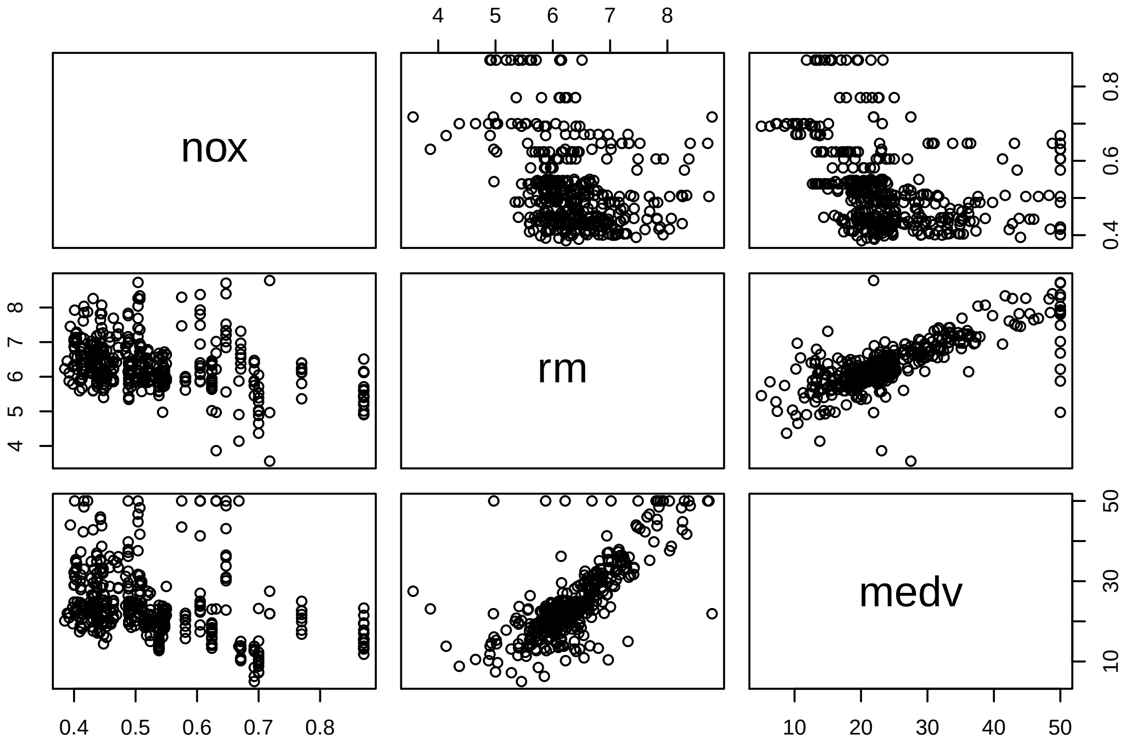
```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  48.80635    5.89369    8.28  2.0e-15 ***
## crim         -0.17804    0.04804   -3.71  0.00024 ***
## zn            0.02644    0.01269    2.08  0.03784 *
## indus         0.03365    0.05870    0.57  0.56680
## chas          1.54677    0.80120    1.93  0.05427 .
## nox         -14.00731    4.09506   -3.42  0.00069 ***
## rm            3.32190    0.46181    7.19  3.3e-12 ***
## age           0.03044    0.01314    2.32  0.02110 *
## dis          -1.10964    0.19115   -5.81  1.3e-08 ***
## rad           0.41802    0.07873    5.31  1.9e-07 ***
## tax          -0.01457    0.00399   -3.65  0.00030 ***
## ptratio      -0.73653    0.12613   -5.84  1.1e-08 ***
## black        -0.00140    0.00587   -0.24  0.81171
## log(lstat)   -8.79527    0.64576  -13.62  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.32 on 386 degrees of freedom
## Multiple R-squared:  0.785,  Adjusted R-squared:  0.778
## F-statistic:  108 on 13 and 386 DF,  p-value: <2e-16
```

Way Better!! Our R^2 has improved to 78.5%. Now try as an excersise to see if there are other variables which needs transformation, and see if you can get a higher R^2. You may use the `pairs()` function:

```r
pairs(training_data)
```




```r
#remember that you can choose what variables to include. In this way, you are zooming in!

pairs(training_data[,c(5, 6, 14)])
```



Now it's time to check for collinearity. We can use the VIF (Variance Inflation Factor) criteria to see if one (or more) of our predictors were uncorrelated with the other predictor variables in the model. We use a function called `vif()`, and it is found in an R package called `car`. The higher the VIF for a variable gets, then the variable would be highly correlated with at least one of the other predictors in the model.

```
#vif() is in package car
library(car)
#vif takes the linear model created as its argument
vif(model)
```

```
##       crim          zn       indus        chas         nox          rm
##      1.768       2.219       3.121       1.098       4.650       2.453
##        age         dis         rad         tax     ptratio       black
##      3.102       3.701       5.261       5.618       1.676       1.217
## log(lstat)
##      3.152
```

We notice that `tax` has a high VIF (5.61), which means that the variance of the estimated coefficient of `tax` is inflated by a factor of 5.61 because `tax` is highly correlated with at least one of the other predictors in the model.

Let's look at the correlations:

```
cor(training_data)
```

```
##                crim        zn   indus       chas      nox        rm     age
## crim      1.00000 -0.15819  0.3520  0.0052200  0.37742 -0.16712  0.2901
## zn       -0.15819  1.00000 -0.5037 -0.0813125 -0.48833  0.30734 -0.5548
## indus     0.35198 -0.50374  1.0000  0.1572596  0.74732 -0.40333  0.6096
## chas      0.00522 -0.08131  0.1573  1.0000000  0.17230  0.07749  0.1435
```

```
## nox      0.37742 -0.48833  0.7473  0.1722982  1.00000 -0.31484  0.7144
## rm      -0.16712  0.30734 -0.4033  0.0774869 -0.31484  1.00000 -0.2340
## age      0.29009 -0.55478  0.6096  0.1435056  0.71438 -0.23398  1.0000
## dis     -0.32231  0.64137 -0.6706 -0.1666086 -0.75053  0.16545 -0.7254
## rad      0.62855 -0.23691  0.4600  0.1326525  0.53253 -0.22196  0.3860
## tax      0.57825 -0.22155  0.5931  0.0880796  0.61172 -0.30595  0.4360
## ptratio  0.20287 -0.32857  0.2352 -0.0716383  0.02526 -0.35868  0.1558
## black   -0.05114  0.13445 -0.2755 -0.0522459 -0.35252  0.19841 -0.2104
## lstat    0.38278 -0.38874  0.5512 -0.0004058  0.53483 -0.64769  0.5682
## medv    -0.26555  0.30881 -0.3739  0.1388716 -0.30336  0.75288 -0.2782
##              dis      rad      tax  ptratio    black      lstat    medv
## crim    -0.32231  0.62855  0.57825  0.20287 -0.05114  0.3827797 -0.2655
## zn       0.64137 -0.23691 -0.22155 -0.32857  0.13445 -0.3887370  0.3088
## indus   -0.67062  0.46003  0.59312  0.23524 -0.27548  0.5511707 -0.3739
## chas    -0.16661  0.13265  0.08808 -0.07164 -0.05225 -0.0004058  0.1389
## nox     -0.75053  0.53253  0.61172  0.02526 -0.35252  0.5348280 -0.3034
## rm       0.16545 -0.22196 -0.30595 -0.35868  0.19841 -0.6476871  0.7529
## age     -0.72537  0.38599  0.43600  0.15583 -0.21036  0.5682193 -0.2782
## dis      1.00000 -0.41979 -0.44527 -0.09583  0.20604 -0.4226715  0.1102
## rad     -0.41979  1.00000  0.86813  0.33169 -0.05732  0.3628684 -0.1937
## tax     -0.44527  0.86813  1.00000  0.30004 -0.16342  0.4243226 -0.3111
## ptratio -0.09583  0.33169  0.30004  1.00000  0.07824  0.2886941 -0.4354
## black    0.20604 -0.05732 -0.16342  0.07824  1.00000 -0.1664499  0.1463
## lstat   -0.42267  0.36287  0.42432  0.28869 -0.16645  1.0000000 -0.7010
## medv     0.11021 -0.19368 -0.31114 -0.43545  0.14627 -0.7010219  1.0000
```

```r
#let's make our lives easier and round our correlations
round(cor(training_data), digits =2)
```

```
##          crim    zn indus  chas   nox    rm   age   dis   rad   tax
## crim     1.00 -0.16  0.35  0.01  0.38 -0.17  0.29 -0.32  0.63  0.58
## zn      -0.16  1.00 -0.50 -0.08 -0.49  0.31 -0.55  0.64 -0.24 -0.22
## indus    0.35 -0.50  1.00  0.16  0.75 -0.40  0.61 -0.67  0.46  0.59
## chas     0.01 -0.08  0.16  1.00  0.17  0.08  0.14 -0.17  0.13  0.09
## nox      0.38 -0.49  0.75  0.17  1.00 -0.31  0.71 -0.75  0.53  0.61
## rm      -0.17  0.31 -0.40  0.08 -0.31  1.00 -0.23  0.17 -0.22 -0.31
## age      0.29 -0.55  0.61  0.14  0.71 -0.23  1.00 -0.73  0.39  0.44
## dis     -0.32  0.64 -0.67 -0.17 -0.75  0.17 -0.73  1.00 -0.42 -0.45
## rad      0.63 -0.24  0.46  0.13  0.53 -0.22  0.39 -0.42  1.00  0.87
## tax      0.58 -0.22  0.59  0.09  0.61 -0.31  0.44 -0.45  0.87  1.00
## ptratio  0.20 -0.33  0.24 -0.07  0.03 -0.36  0.16 -0.10  0.33  0.30
## black   -0.05  0.13 -0.28 -0.05 -0.35  0.20 -0.21  0.21 -0.06 -0.16
## lstat    0.38 -0.39  0.55  0.00  0.53 -0.65  0.57 -0.42  0.36  0.42
## medv    -0.27  0.31 -0.37  0.14 -0.30  0.75 -0.28  0.11 -0.19 -0.31
##         ptratio black lstat  medv
## crim       0.20 -0.05  0.38 -0.27
## zn        -0.33  0.13 -0.39  0.31
## indus      0.24 -0.28  0.55 -0.37
## chas      -0.07 -0.05  0.00  0.14
## nox        0.03 -0.35  0.53 -0.30
## rm        -0.36  0.20 -0.65  0.75
## age        0.16 -0.21  0.57 -0.28
## dis       -0.10  0.21 -0.42  0.11
## rad        0.33 -0.06  0.36 -0.19
```

```
## tax        0.30 -0.16  0.42 -0.31
## ptratio    1.00  0.08  0.29 -0.44
## black      0.08  1.00 -0.17  0.15
## lstat      0.29 -0.17  1.00 -0.70
## medv      -0.44  0.15 -0.70  1.00
```

Aha! `tax` and `rad` are highly correlated (0.87). And if you go back to the VIF output, you can see that `rad` was the other variable causing trouble. So the remedy is to delete one of them. It is a subjective issue what to delete since both their VIF is close.

Let's run the model again without `rad`, and see what happens to the R^2.

```
model = lm(medv~.-lstat+log(lstat)-tax, data = training_data)
summary(model)
```

```
##
## Call:
## lm(formula = medv ~ . - lstat + log(lstat) - tax, data = training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.847  -2.535  -0.279   1.951  23.380
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.62e+01   5.94e+00    7.78  6.7e-14 ***
## crim        -1.84e-01   4.88e-02   -3.78  0.00018 ***
## zn           1.69e-02   1.26e-02    1.34  0.18060
## indus       -3.08e-02   5.69e-02   -0.54  0.58831
## chas         1.85e+00   8.10e-01    2.28  0.02318 *
## nox         -1.60e+01   4.12e+00   -3.88  0.00012 ***
## rm           3.43e+00   4.68e-01    7.33  1.4e-12 ***
## age          2.88e-02   1.33e-02    2.16  0.03141 *
## dis         -1.15e+00   1.94e-01   -5.91  7.5e-09 ***
## rad          2.06e-01   5.40e-02    3.82  0.00016 ***
## ptratio     -7.66e-01   1.28e-01   -5.99  4.8e-09 ***
## black        3.07e-04   5.94e-03    0.05  0.95886
## log(lstat)  -8.75e+00   6.56e-01  -13.35  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.39 on 387 degrees of freedom
## Multiple R-squared:  0.778,  Adjusted R-squared:  0.771
## F-statistic:  113 on 12 and 387 DF,  p-value: <2e-16
```

Of course R^2 should go a little lower because we deleted one of the variables. But check for the model significance (F-statistic) gets higher, which means the p-values gets lower and thus our model is more significant without `rad`. Try to delete `tax` instead of `rad`!

Interaction Terms

Suppose that we want to add an interaction term between `lstat` and `age` to the model, then we can use the syntax `lstat:age`. On the other hand, if we want to make our task easier and reduce the syntax typing, then we can use the term `lstat*age` which simultanously includes `lstat`, `age`, and `lstat:age`. The latter syntax is just a shortcut for three variables.

```
model = lm(medv ~ log(lstat)*age, data = training_data)
summary(model)
```

```
##
## Call:
## lm(formula = medv ~ log(lstat) * age, data = training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.987  -3.435  -0.889   2.585  22.839
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     46.3330     2.6135   17.73  < 2e-16 ***
## log(lstat)     -11.6475     1.3388   -8.70  < 2e-16 ***
## age              0.1530     0.0360    4.25  2.7e-05 ***
## log(lstat):age  -0.0364     0.0166   -2.19    0.029 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.27 on 396 degrees of freedom
## Multiple R-squared:  0.672,  Adjusted R-squared:  0.67
## F-statistic:  271 on 3 and 396 DF,  p-value: <2e-16
```

To add all interaction terms then you can write your formula as `lm(medv~(.)^2)`.

Asseing the Model

While assesing the model, I will use the one with no interactions. But you can try and see what would be the final MSE. If you get a lower one, then that's great. Try it!

```
# re-runing the model
model = lm(medv~.-lstat+log(lstat)-tax, data = training_data)
summary(model)
```

```
##
## Call:
## lm(formula = medv ~ . - lstat + log(lstat) - tax, data = training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.847  -2.535  -0.279   1.951  23.380
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.62e+01   5.94e+00    7.78  6.7e-14 ***
## crim        -1.84e-01   4.88e-02   -3.78  0.00018 ***
## zn           1.69e-02   1.26e-02    1.34  0.18060
## indus       -3.08e-02   5.69e-02   -0.54  0.58831
```

```
## chas         1.85e+00   8.10e-01    2.28  0.02318 *
## nox         -1.60e+01   4.12e+00   -3.88  0.00012 ***
## rm           3.43e+00   4.68e-01    7.33  1.4e-12 ***
## age          2.88e-02   1.33e-02    2.16  0.03141 *
## dis         -1.15e+00   1.94e-01   -5.91  7.5e-09 ***
## rad          2.06e-01   5.40e-02    3.82  0.00016 ***
## ptratio     -7.66e-01   1.28e-01   -5.99  4.8e-09 ***
## black        3.07e-04   5.94e-03    0.05  0.95886
## log(lstat)  -8.75e+00   6.56e-01  -13.35  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.39 on 387 degrees of freedom
## Multiple R-squared:  0.778,  Adjusted R-squared:  0.771
## F-statistic:  113 on 12 and 387 DF,  p-value: <2e-16
```

```r
#save the testing median values for houses (testing y) in y
y = testing_data$medv

#compute the predicted value for this y (y hat)
y_hat = predict(model, testing_data[,-14])

#Now we have both y and y_hat for our testing data. let's find the mean square error

error = y-y_hat
error_squared = error^2
MSE = mean(error_squared)
MSE
```

```
## [1] 23.34
```

Dealing with Categorical Variables in R

Let's look at the `Carseats` dataset in the `ISLR` package. This is a simulated data set containing sales of child car seats at 400 different stores. We would like to predict `sales` of children carseats in 400 locations based on 10 predictors. One of these predictors is `ShelveLoc`, which is factor or a categorical variable with levels Bad, Good and Medium indicating the quality of the shelving location for the car seats at each site. Given a categorical variable such as `ShelvLoc`, `R` generates dummy variables automatically.

```r
library(ISLR)
model = lm(Sales~., data = Carseats)
summary(model)
```

```
##
## Call:
## lm(formula = Sales ~ ., data = Carseats)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.869 -0.691  0.021  0.664  3.411
##
## Coefficients:
```

```
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.660623   0.603449    9.38  < 2e-16 ***
## CompPrice        0.092815   0.004148   22.38  < 2e-16 ***
## Income           0.015803   0.001845    8.56  2.6e-16 ***
## Advertising      0.123095   0.011124   11.07  < 2e-16 ***
## Population       0.000208   0.000370    0.56     0.58
## Price           -0.095358   0.002671  -35.70  < 2e-16 ***
## ShelveLocGood    4.850183   0.153110   31.68  < 2e-16 ***
## ShelveLocMedium  1.956715   0.126106   15.52  < 2e-16 ***
## Age             -0.046045   0.003182  -14.47  < 2e-16 ***
## Education       -0.021102   0.019720   -1.07     0.29
## UrbanYes         0.122886   0.112976    1.09     0.28
## USYes           -0.184093   0.149842   -1.23     0.22
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.02 on 388 degrees of freedom
## Multiple R-squared:  0.873,  Adjusted R-squared:  0.87
## F-statistic:  243 on 11 and 388 DF,  p-value: <2e-16
```

You can see that we don't have `ShelveLoc` in our output variables, but instead we have `ShelveLocGood`, and `ShelveLocMedium`. Let's look at the contrasts of this variable in order to understand what's happening:

```r
contrasts(Carseats$ShelveLoc)
```

```
##        Good Medium
## Bad       0      0
## Good      1      0
## Medium    0      1
```

So, R has created a dummy variable called `ShelveLocGood` that takes on a value of 1 if the location is good, and 0 otherwise. It has also created has created a dummy variable called `ShelveLocMeduim` that takes on a value of 1 if the location is medium, and 0 otherwise.