House Price Linear Regression model in R Studio

GAURAV SIWAL (M.TECH)

2020-05-28

```
df = read.csv("C:/Users/CEA/Desktop/ML by R Alison/Complete ML in R/1. Linear
Regression/House_Price.csv", header = TRUE)
View(df)
summary(df)
```

```
##      price         crime_rate        resid_area       air_qual
##  Min.   : 5.00   Min.   : 0.00632   Min.   :30.46   Min.   :0.3850
##  1st Qu.:17.02   1st Qu.: 0.08204   1st Qu.:35.19   1st Qu.:0.4490
##  Median :21.20   Median : 0.25651   Median :39.69   Median :0.5380
##  Mean   :22.53   Mean   : 3.61352   Mean   :41.14   Mean   :0.5547
##  3rd Qu.:25.00   3rd Qu.: 3.67708   3rd Qu.:48.10   3rd Qu.:0.6240
##  Max.   :50.00   Max.   :88.97620   Max.   :57.74   Max.   :0.8710
##
##     room_num          age             dist1            dist2
##  Min.   :3.561   Min.   :  2.90   Min.   : 1.130   Min.   : 0.920
##  1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.270   1st Qu.: 1.940
##  Median :6.208   Median : 77.50   Median : 3.385   Median : 3.010
##  Mean   :6.285   Mean   : 68.57   Mean   : 3.972   Mean   : 3.629
##  3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.367   3rd Qu.: 4.992
##  Max.   :8.780   Max.   :100.00   Max.   :12.320   Max.   :11.930
##
##      dist3            dist4           teachers       poor_prop     airport
##  Min.   : 1.150   Min.   : 0.730   Min.   :18.00   Min.   : 1.73   NO :227
##  1st Qu.: 2.232   1st Qu.: 1.940   1st Qu.:19.80   1st Qu.: 6.95   YES:279
##  Median : 3.375   Median : 3.070   Median :20.95   Median :11.36
##  Mean   : 3.961   Mean   : 3.619   Mean   :21.54   Mean   :12.65
##  3rd Qu.: 5.407   3rd Qu.: 4.985   3rd Qu.:22.60   3rd Qu.:16.95
##  Max.   :12.320   Max.   :11.940   Max.   :27.40   Max.   :37.97
##
##    n_hos_beds      n_hot_rooms              waterbody      rainfall
##  Min.   : 5.268   Min.   : 10.06   Lake           : 97   Min.   : 3.00
##  1st Qu.: 6.635   1st Qu.: 11.19   Lake and River: 71   1st Qu.:28.00
##  Median : 7.999   Median : 12.72   None          :155   Median :39.00
##  Mean   : 7.900   Mean   : 13.04   River         :183   Mean   :39.18
##  3rd Qu.: 9.088   3rd Qu.: 14.17                         3rd Qu.:50.00
##  Max.   :10.876   Max.   :101.12                         Max.   :60.00
##  NA's   :8
##  bus_ter       parks
##  YES:506   Min.   :0.03329
##            1st Qu.:0.04646
##            Median :0.05351
##            Mean   :0.05445
```

```
##              3rd Qu.:0.06140
##              Max.   :0.08671
##
```

```
# treatment for missing data
mean_val = mean(df$n_hos_beds,na.rm = TRUE)
df$n_hos_beds[is.na(df$n_hos_beds)] = mean_val

summary(df$n_hos_beds)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   5.268   6.659   7.963   7.900   9.076  10.876
```

```
# outlier treatment
uv = 3* quantile(df$crime_rate, 0.99)
df$crime_rate[df$crime_rate > uv]= uv
summary(df$crime_rate)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.00632  0.08204  0.25651  3.61352  3.67708 88.97620
```

```
lv = 0.3 * quantile(df$rainfall, 0.01)
df$rainfall[df$rainfall < lv]= lv
summary(df$rainfall)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.00   28.00   39.00   39.19   50.00   60.00
```

```
head(df)
```

```
##   price crime_rate resid_area air_qual room_num  age dist1 dist2 dist3
dist4
## 1  24.0    0.00632      32.31    0.538    6.575 65.2  4.35  3.81  4.18
4.01
## 2  21.6    0.02731      37.07    0.469    6.421 78.9  4.99  4.70  5.12
5.06
## 3  34.7    0.02729      37.07    0.469    7.185 61.1  5.03  4.86  5.01
4.97
## 4  33.4    0.03237      32.18    0.458    6.998 45.8  6.21  5.93  6.16
5.96
## 5  36.2    0.06905      32.18    0.458    7.147 54.2  6.16  5.86  6.37
5.86
## 6  28.7    0.02985      32.18    0.458    6.430 58.7  6.22  5.80  6.23
5.99
##   teachers poor_prop airport n_hos_beds n_hot_rooms waterbody rainfall
bus_ter
## 1     24.7      4.98     YES      5.480     11.1920     River       23
YES
## 2     22.2      9.14      NO      7.332     12.1728      Lake       42
YES
## 3     22.2      4.03      NO      7.394    101.1200      None       38
YES
```

```
## 4     21.3      2.94     YES      9.268     11.2672      Lake         45
YES
## 5     21.3      5.33      NO      8.824     11.2896      Lake         55
YES
## 6     21.3      5.21     YES      7.174     14.2296      None         53
YES
##        parks
## 1 0.04934731
## 2 0.04614563
## 3 0.04576397
## 4 0.04715060
## 5 0.03947400
## 6 0.04590965
```

```
# average dist
df$avg_dist = (df$dist1 + df$dist2+ df$dist3+ df$dist4)/4
head(df)
```

```
##   price crime_rate resid_area air_qual room_num  age dist1 dist2 dist3
dist4
## 1  24.0    0.00632      32.31    0.538    6.575 65.2  4.35  3.81  4.18
4.01
## 2  21.6    0.02731      37.07    0.469    6.421 78.9  4.99  4.70  5.12
5.06
## 3  34.7    0.02729      37.07    0.469    7.185 61.1  5.03  4.86  5.01
4.97
## 4  33.4    0.03237      32.18    0.458    6.998 45.8  6.21  5.93  6.16
5.96
## 5  36.2    0.06905      32.18    0.458    7.147 54.2  6.16  5.86  6.37
5.86
## 6  28.7    0.02985      32.18    0.458    6.430 58.7  6.22  5.80  6.23
5.99
##   teachers poor_prop airport n_hos_beds n_hot_rooms waterbody rainfall
bus_ter
## 1     24.7      4.98     YES      5.480     11.1920     River         23
YES
## 2     22.2      9.14      NO      7.332     12.1728      Lake         42
YES
## 3     22.2      4.03      NO      7.394    101.1200      None         38
YES
## 4     21.3      2.94     YES      9.268     11.2672      Lake         45
YES
## 5     21.3      5.33      NO      8.824     11.2896      Lake         55
YES
## 6     21.3      5.21     YES      7.174     14.2296      None         53
YES
##        parks avg_dist
## 1 0.04934731   4.0875
## 2 0.04614563   4.9675
## 3 0.04576397   4.9675
```

```
## 4 0.04715060   6.0650
## 5 0.03947400   6.0625
## 6 0.04590965   6.0600
```

```r
df= df[,c(-7,-8,-9,-10)]

# categorical value treatment
library('dummies')
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```r
df = dummy.data.frame(df)
```

```
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =
FALSE):
## non-list contrasts argument ignored

## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =
FALSE):
## non-list contrasts argument ignored
```

```r
df = df[,c(-9,-15)]

# train test split
library('caTools')
```

```
## Warning: package 'caTools' was built under R version 3.6.3
```

```r
set.seed(0)
split = sample.split(df, SplitRatio = 0.8)
train = subset(df, split == 'TRUE')
test = subset(df, split == 'FALSE')

# linear regression
lreg = lm(price~.,data = train)
summary(lreg)
```

```
##
## Call:
## lm(formula = price ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.113  -2.963  -0.800   1.964  26.580
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -4.113657   5.912565  -0.696   0.4870
## crime_rate                  -0.067481   0.038063  -1.773   0.0771 .
## resid_area                  -0.037266   0.067811  -0.550   0.5830
## air_qual                   -14.791888   6.520164  -2.269   0.0239 *
```

```
## room_num                     4.036105   0.484752   8.326 1.65e-15 ***
## age                         -0.005714   0.015766  -0.362   0.7172
## teachers                     0.888358   0.137164   6.477 3.00e-10 ***
## poor_prop                   -0.652535   0.059251 -11.013  < 2e-16 ***
## airportYES                   1.141855   0.531672   2.148   0.0324 *
## n_hos_beds                   0.370085   0.178322   2.075   0.0386 *
## n_hot_rooms                  0.029834   0.043419   0.687   0.4924
## waterbodyLake                0.258283   0.787181   0.328   0.7430
## `waterbodyLake and River`   -0.709849   0.792252  -0.896   0.3708
## waterbodyRiver              -0.629699   0.629548  -1.000   0.3179
## rainfall                     0.011073   0.021009   0.527   0.5985
## parks                       31.387873  59.037418   0.532   0.5953
## avg_dist                    -1.298917   0.216917  -5.988 5.04e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.998 on 369 degrees of freedom
## Multiple R-squared:  0.7287, Adjusted R-squared:  0.7169
## F-statistic: 61.95 on 16 and 369 DF,  p-value: < 2.2e-16
```

```
pred_train = predict(lreg, test)

mse = mean((pred_train - test$price)^2)
mse
```

```
## [1] 22.2732
```

```
# Other linear models for better prediction accuracy and model
interpretability

# 1) Subset Selection Techniques

# 1.a) Best subset selection

library("leaps")
```

```
## Warning: package 'leaps' was built under R version 3.6.3
```

```
lm_sub_reg = regsubsets(price~., data = df, nvmax = 15)
summary(lm_sub_reg)$adjr2
```

```
##  [1] 0.5479428 0.6410518 0.6792769 0.6909566 0.7088919 0.7120175 0.7144706
##  [8] 0.7170322 0.7172295 0.7172748 0.7172916 0.7171582 0.7168656 0.7166040
## [15] 0.7162438
```

```
which.max(summary(lm_sub_reg)$adjr2)
```

```
## [1] 11
```

```
coef(lm_sub_reg,11)
```

```
##                (Intercept)                crime_rate                 
air_qual 
##                -8.45195104               -0.07350473               -
21.46196860 
##                   room_num                  teachers                 
poor_prop 
##                 4.15302687                0.98444383                -
0.55460143 
##                 airportYES                n_hos_beds `waterbodyLake and 
River` 
##                 1.03063947                0.36294609                -
0.72313242 
##                   rainfall                     parks                 
avg_dist 
##                 0.01790746               60.06200089                -
1.17184364
```

```
# 1.b) Forward stepwise selection

lm_fsub_reg = regsubsets(price~., data = df, nvmax = 15, method = 'forward')
summary(lm_fsub_reg)$adjr2
```

```
##  [1] 0.5479428 0.6410518 0.6792769 0.6909566 0.7088919 0.7120175 0.7144706
##  [8] 0.7170322 0.7172295 0.7172748 0.7172916 0.7171582 0.7168656 0.7166040
## [15] 0.7162438
```

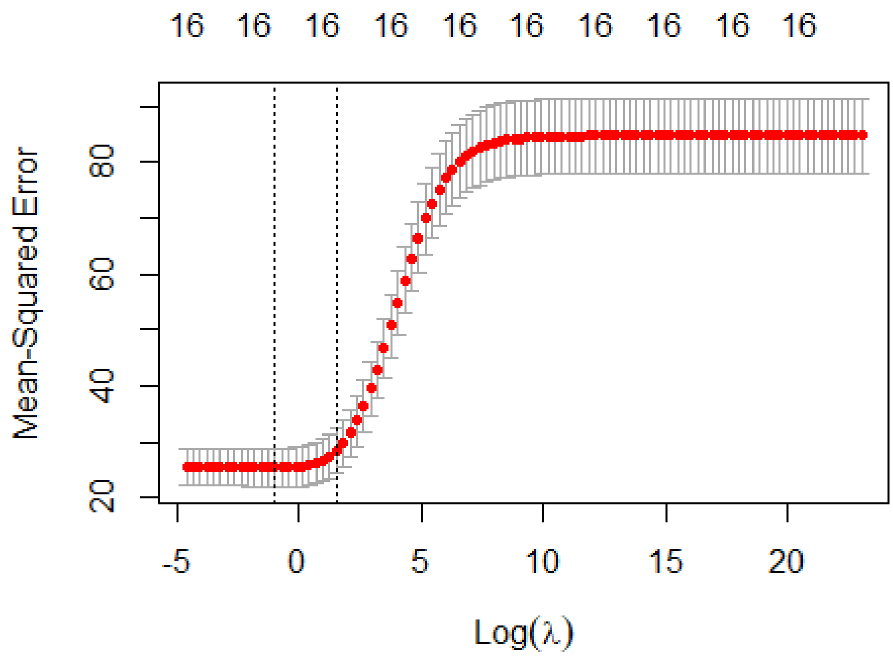
```
which.max(summary(lm_fsub_reg)$adjr2)
```

```
## [1] 11
```

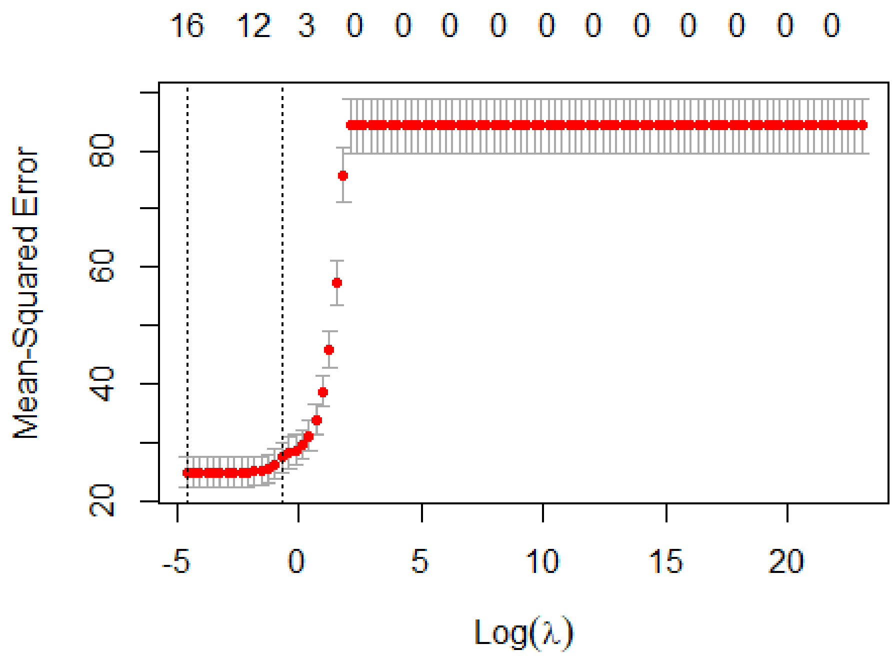
```
coef(lm_fsub_reg, 11)
```

```
##                (Intercept)                crime_rate                 
air_qual 
##                -8.45195104               -0.07350473               -
21.46196860 
##                   room_num                  teachers                 
poor_prop 
##                 4.15302687                0.98444383                -
0.55460143 
##                 airportYES                n_hos_beds `waterbodyLake and 
River` 
##                 1.03063947                0.36294609                -
0.72313242 
##                   rainfall                     parks                 
avg_dist 
##                 0.01790746               60.06200089                -
1.17184364
```

```
# 1.c) Backward stepwise selection
lm_bsub_reg = regsubsets(price~., data = df, nvmax = 15, method = 'backward')
summary(lm_bsub_reg)$adjr2

##  [1] 0.5479428 0.6410518 0.6792769 0.6909566 0.7088919 0.7120175 0.7144706
##  [8] 0.7170322 0.7172295 0.7172748 0.7172916 0.7171582 0.7168656 0.7166040
## [15] 0.7162438

which.max(summary(lm_bsub_reg)$adjr2)

## [1] 11

coef(lm_bsub_reg, 11)

##               (Intercept)                crime_rate
air_qual
##               -8.45195104               -0.07350473               -
21.46196860
##                  room_num                  teachers
poor_prop
##                4.15302687                0.98444383                -
0.55460143
##                airportYES                n_hos_beds `waterbodyLake and
River`
##                1.03063947                0.36294609                -
0.72313242
##                  rainfall                     parks
avg_dist
##                0.01790746               60.06200089                -
1.17184364

# 2) Shrinkage method ( Ridge and Lasso Technique)

# 2.1) ridge regression
library('glmnet')

## Warning: package 'glmnet' was built under R version 3.6.3

## Loading required package: Matrix

## Loaded glmnet 4.0

x= model.matrix(price~.,data = df)[,-1]
y = df$price
grid = 10^ seq(10,-2,length = 100)
lm_ridge = glmnet(x,y,alpha = 0, lambda = grid)
lm_ridgecv = cv.glmnet(x,y,alpha = 0, lambda = grid)
plot(lm_ridgecv)
```




```
optlambda = lm_ridgecv$lambda.min

pred_ridge = predict(lm_ridge, s= optlambda, newx = x)
mse_ridge = mean((pred_ridge - y)^2)
mse_ridge

## [1] 23.27537

TSS = mean((mean(y)- y)^2)
RSS = mean((pred_ridge - y)^2)
r2 = (1-(RSS/TSS))
r2

## [1] 0.7233922

# 2.2) Lasso regression
lm_lasso = glmnet(x,y, alpha = 1, lambda = grid)
lm_lassocv = cv.glmnet(x,y, alpha = 1, lambda = grid)
plot(lm_lassocv)
```




```
optlambda_lasso = lm_lassocv$lambda.min

pred_lasso = predict(lm_lasso, s= optlambda_lasso, newx = x)
mse_lasso = mean((pred_lasso - y)^2)
mse_lasso

## [1] 23.16464

TSS_lasso = mean((mean(y)- y)^2)
RSS_lasso = mean((pred_lasso - y)^2)
r2_lasso = 1-(RSS_lasso/TSS_lasso)
r2_lasso

## [1] 0.7247081
```