Stat 2025- HW 2

Michael Discenza

March 11, 2013

1 Best-subset Model Selection using Cross-validation

I first load the library that will help with the cross-validation and then I import houseprice dataset into R

```
## Loading required package:  MASS
## Loading required package:  lattice
## Loading required package:  latticeExtra
## Loading required package:  RColorBrewer
## Loading required package:  rpart
## Loading required package:  randomForest
## randomForest 4.6-7
## Type rfNews() to see new features/changes/bug fixes.
## Loading required package:  boot
##
## Attaching package:  'boot'
## The following object(s) are masked from 'package:lattice':
##
##    melanoma
## Loading required package:  survival
## Loading required package:  splines
##
## Attaching package:  'survival'
## The following object(s) are masked from 'package:boot':
##
##    aml
##
## Attaching package:  'DAAG'
## The following object(s) are masked from 'package:survival':
##
##    lung
## The following object(s) are masked from 'package:MASS':
##
##    hills
```

Next, I define two helper functions. The helper function were `helper1` and helper2 The first, which is based on the source from Hadley Wickham's Meify Package (see `https://github.com/hadley/meifly/blob/master/R/generate.r`) returns all possible simple linear models that can be made from the set of predictors:

-

```
all_possible_models <- function(x.column.names, y.col.name){
    Cols <- x.column.names
    n <- length(Cols)
    id <- unlist(
        lapply(1:n,
               function(i)combn(1:n,i,simplify=F)
        )
        ,recursive=F)
    Formulas <- sapply(id,function(i)
        paste(y.col.name, "~",paste(Cols[i],collapse="+"))
    )
}
```

The second function does five-fold cross vaidation on the data set for a given model (formula) input and returns the cross-validated means squared error:

```
apply_formula <- function(model, cv=5){ #make the default value for this function 5
    cv.model <- CVlm(houseprices,lm(model, data=houseprices),m=cv)
    cvmse <-mean((cv.model$sale.price - cv.model$cvpred)^2)
}
```

I then combine these functions to return the cross-validated mean squared error for the all of the possible simple linear models:

```
FORMULAS <- all_possible_models(
    x.column.names=colnames(houseprices[1:3]),
    y.col.name=colnames(houseprices[4])
    )
FORMULAS[8] <- "sale.price ~ 1" # add the null model
CV.MSE <- lapply(FORMULAS, apply_formula)
final.results <- cbind(FORMULAS, CV.MSE)
```

```
final.results

##      FORMULAS                           CV.MSE
## [1,] "sale.price ~ floors"              4009
## [2,] "sale.price ~ area"                3531
## [3,] "sale.price ~ bedrooms"            2103
## [4,] "sale.price ~ floors+area"         3689
## [5,] "sale.price ~ floors+bedrooms"     2139
## [6,] "sale.price ~ area+bedrooms"       1322
## [7,] "sale.price ~ floors+area+bedrooms" 1542
## [8,] "sale.price ~ 1"                   3692
```

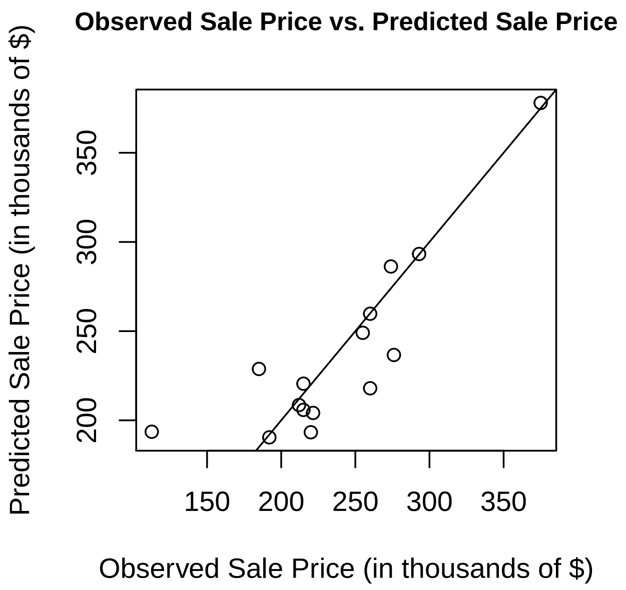
Comparing the cross-validated mean squared errors for each model, we can see that the best simple linear model is , sale.price, area + bedrooms. A summary of this model is shown below:

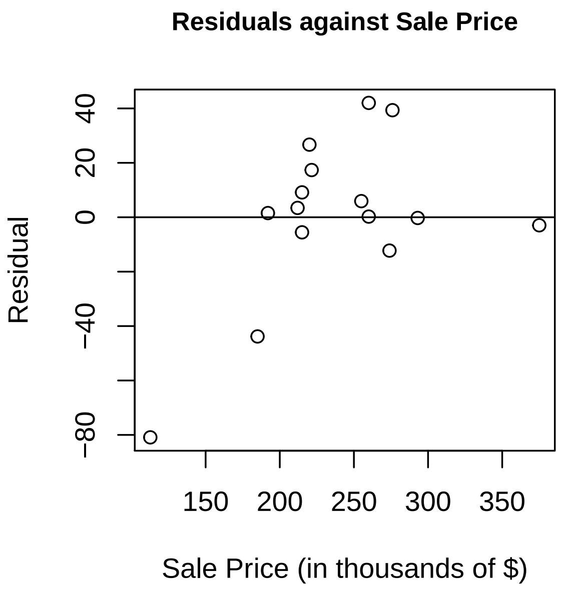
```
lm.optimatal <- lm(sale.price ~ area + bedrooms, data = houseprices)
summary(lm.optimatal)

##
## Call:
## lm(formula = sale.price ~ area + bedrooms, data = houseprices)
```

```
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -80.90  -4.25   1.54  13.25  42.03
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -141.761     67.872   -2.09   0.0587 .
## area           0.142      0.047    3.03   0.0104 *
## bedrooms      58.324     14.760    3.95   0.0019 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33.1 on 12 degrees of freedom
## Multiple R-squared: 0.731,Adjusted R-squared: 0.686
## F-statistic: 16.3 on 2 and 12 DF,  p-value: 0.000379
```

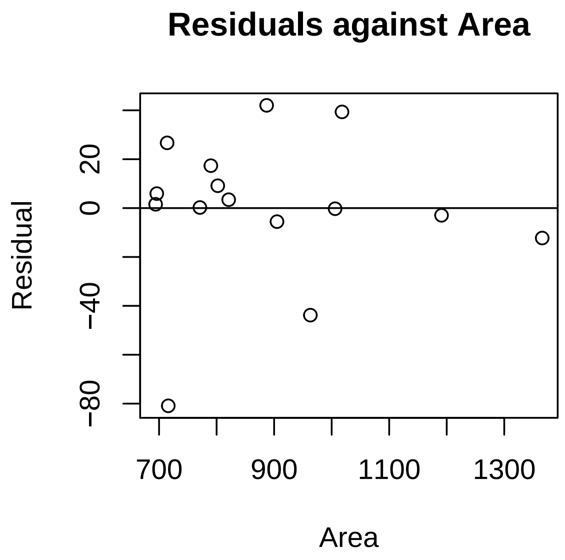
2 Checking Assumptions for Linear Regression

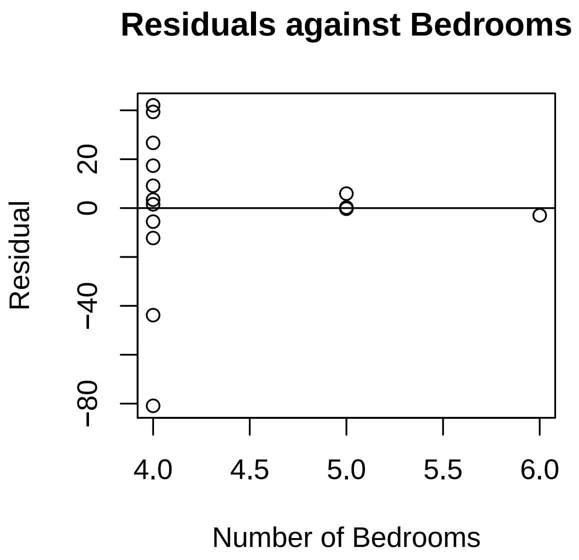
First we check for to see that we were correct in assuming that the relationship we are modeling is actually linear. One test we can conduct is plotting our our observed values for sale price against our predicted values for sale price. Here we see that the points seem fairly randomoly distributed around the line through the diagonal so we can accept the linearity assumption. Another test for linearity is plotting residuals versus our preidcted values. Below, this plot, the one to the right, is really only a rotation of the plot on the left. It seems as though there is slightly more of a pattern than we might have been able to visually identify before. This casts some doubt on our linearity assumption.



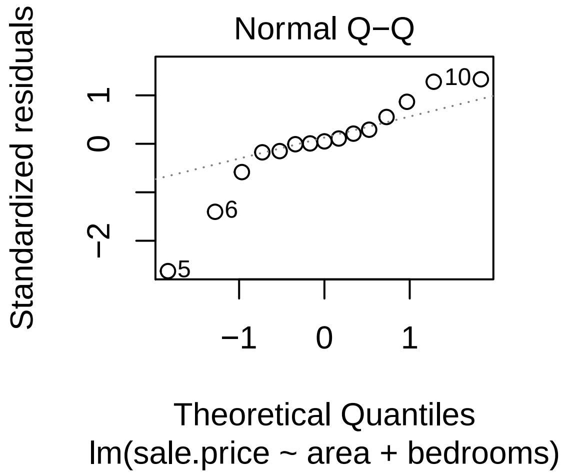


An additional assumption that we need to check is homoscedasticity, the constant variance of home sale prices across different values of home area and number of bedrooms. Below, it seems that in larger houses there is less variance around the fitted values. This, however, is due to the fact that these larger houses are influential points, which have the effect of pulling the line through (or at least closely around) themselves. Ideally, there would be more houses throughout the range of the variables so that the relationships we identified are not merely the product of a few extreme data points.





Next, we test the assumption that residuals are noramlly distributed about the fitted regression line. To analyze this relationship visually, we examine a "qqnorm" plot, which helps us compare the distribution of the scaled residuals to that of the standard normal distribution. On the tails, there seems to be significant deviation from the normal distribution.

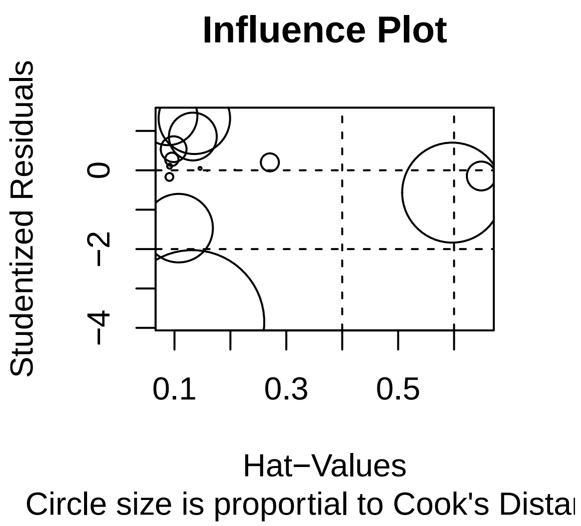


Another assumption that we would want to check in fitting this model is that the predictors in are model are not correlated. This is important becase if we have multicolinearity, the variance of the fitted values increases. To check for multicolinearity, we calculate a variance inflation factor (vif) using a function from the HH package. We get values that are less than 5 so conclude that in our model, the predictors' colinearity is acceptable.

```
library(HH)
vif(lm.optimatal)

##     area bedrooms
##     1.06     1.06
```

A final check that we might want to do is for outliers and influential observations. These are important to check for because if they do exist, it indicates that our model might not be representative of the true population relationship in that a single observation biases it significantly. Using the Bonferroni Procedure, deleted or externally studentized residuals, we see that that the 5th observation is an outlier. Conceptually speaking, we know this to be the case because our fitted values for each data point would be statistically signifianctly different if this observation were not included in the data. The graph below also shows that two other observations have large hat-values, meaning they are are highly leveraged.

```
outlierTest(lm.optimatal)

##   rstudent unadjusted p-value Bonferonni p
## 5    -3.86            0.00268       0.0401

influencePlot(lm.optimatal, main="Influence Plot",
              sub="Circle size is proportial to Cook's Distance" )
```



A combinatination of having a large residual and being highly leveraged makes a point influential, which is indicated by Cook's Distance. An observation is considered influential if it has a Cook's Distance that is greater than 4/n, where n is the number of observations in the data set. In this dataset, we see that the 5th observation is influential, which makes sense because it is the house that sold for 112,000 dollars, so is a highly leveraged point.

```
cd <- cooks.distance(lm.optimatal)
subset(cd, cd > 4/15)

##     5
## 0.348
```

Overall, seeing that we violate a few assumptions and are not assured of really satisfying some because of our small n, we should interpet this model with caution. Ideally to fit a more robust model, we would have more data points to more accurately understand the actual variablity of the data.

3 Confidence Interval for the Average Home Price

```
nd <- data.frame(floors=2,area=1000, bedrooms=4)
confidence_interval <- predict(lm.optimatal,interval="confidence",
                               newdata=nd, level=.95)
confidence_interval

##   fit lwr upr
## 1 234 209 260
```

The confidence interval for the average price of two-floor, 1000 square foot house with 4 bedrooms is (209, 260), centered around 234.

4 Prediction Interval for a Home's Price

```
nd <- data.frame(floors=2,area=1000, bedrooms=4)
prediction_interval <- predict(lm.optimatal,interval="prediction",
                               newdata=nd, level=.95)
prediction_interval

##   fit lwr upr
## 1 234 158 310
```

The prediction interval for the price of a single two-floor, 1000 square foot house with 4 bedrooms is (158, 310), centered around 234.

5 Comparison between Confidence and Prediction Intervals

The prediction internval is wider than the confidence interval for houses with the same values of the independent variables because for a confidence interval, uncertainty comes only from variability of the data to which the model is fit. In the case of a prediction interval, there is additional variability associated with picking a single observation and not the average of all observations.

6 References

- I used the following site as reference for assumptions of linear regression: `http://people.duke.edu/~rnau/testing.htm`.
- The full source for this homework without the markup can be found at `https://github.com/mdiscenza/STAT_2025_homework/blob/master/HW2/hw2_script.R`.