Saratoga House Prices

Bao Doquang, Dhwanit Agarwal, Akksay Singh and Shristi Singh

March 13, 2020

Classwork:

```r
library(tidyverse, quietly = TRUE)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.2

## -- Attaching packages ---------------------------------------------------------------------------------

## <U+2713> ggplot2 3.2.1     <U+2713> purrr   0.3.3
## <U+2713> tibble  2.1.3     <U+2713> dplyr   0.8.4
## <U+2713> tidyr   1.0.0     <U+2713> stringr 1.4.0
## <U+2713> readr   1.3.1     <U+2713> forcats 0.4.0

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

## -- Conflicts -------------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(dplyr, quietly = TRUE)
library(mosaic, quietly = TRUE)
```

```
## Warning: package 'mosaic' was built under R version 3.6.2

## Warning: package 'ggstance' was built under R version 3.6.2

##
## Attaching package: 'ggstance'

## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh

##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")

## Warning: package 'mosaicData' was built under R version 3.6.2

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2
```

```
##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.
##
## Note: If you use the Matrix package, be sure to load it BEFORE loading mosaic.

##
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':
##
##     mean

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally

## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:ggplot2':
##
##     stat

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var

## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```r
library(FNN, quietly = TRUE)
```

```
## Warning: package 'FNN' was built under R version 3.6.2
```

```r
library(foreach, quietly = TRUE)
```

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

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```r
data(SaratogaHouses)

summary(SaratogaHouses)
```

```
##      price           lotSize             age           landValue
##  Min.   :  5000   Min.   : 0.0000   Min.   :  0.00   Min.   :   200
##  1st Qu.:145000   1st Qu.: 0.1700   1st Qu.: 13.00   1st Qu.: 15100
##  Median :189900   Median : 0.3700   Median : 19.00   Median : 25000
##  Mean   :211967   Mean   : 0.5002   Mean   : 27.92   Mean   : 34557
##  3rd Qu.:259000   3rd Qu.: 0.5400   3rd Qu.: 34.00   3rd Qu.: 40200
##  Max.   :775000   Max.   :12.2000   Max.   :225.00   Max.   :412600
##    livingArea     pctCollege       bedrooms       fireplaces       bathrooms
```

```
##  Min.   : 616   Min.   :20.00   Min.   :1.000   Min.   :0.0000   Min.   :0.0
##  1st Qu.:1300   1st Qu.:52.00   1st Qu.:3.000   1st Qu.:0.0000   1st Qu.:1.5
##  Median :1634   Median :57.00   Median :3.000   Median :1.0000   Median :2.0
##  Mean   :1755   Mean   :55.57   Mean   :3.155   Mean   :0.6019   Mean   :1.9
##  3rd Qu.:2138   3rd Qu.:64.00   3rd Qu.:4.000   3rd Qu.:1.0000   3rd Qu.:2.5
##  Max.   :5228   Max.   :82.00   Max.   :7.000   Max.   :4.0000   Max.   :4.5
##      rooms                   heating           fuel
##  Min.   : 2.000   hot air        :1121   gas     :1197
##  1st Qu.: 5.000   hot water/steam: 302   electric: 315
##  Median : 7.000   electric       : 305   oil     : 216
##  Mean   : 7.042
##  3rd Qu.: 8.250
##  Max.   :12.000
##                sewer      waterfront newConstruction centralAir
##  septic           : 503   Yes:  15   Yes:  81        Yes: 635
##  public/commercial:1213   No :1713   No :1647        No :1093
##  none             :  12
##
##
##
```

```
#Defining models

# Baseline model
lm_small = lm(price ~ bedrooms + bathrooms + lotSize, data=SaratogaHouses)

# 11 main effects
lm_medium = lm(price ~ lotSize + age + livingArea + pctCollege + bedrooms +
                 fireplaces + bathrooms + rooms + heating + fuel + centralAir, data=SaratogaHouses)

# Sometimes it's easier to name the variables we want to leave out
# The command below yields exactly the same model.
# the dot (.) means "all variables not named"
# the minus (-) means "exclude this variable"
lm_medium2 = lm(price ~ . - sewer - waterfront - landValue - newConstruction, data=SaratogaHouses)

coef(lm_medium)
```

```
##            (Intercept)               lotSize                   age
##            28627.73165            9350.45188              47.54722
##             livingArea             pctCollege              bedrooms
##               91.86974             296.50809          -15630.71950
##             fireplaces             bathrooms                 rooms
##              985.06117           22006.97108            3259.11923
## heatinghot water/steam        heatingelectric          fuelelectric
##            -9429.79463           -3609.98574          -12094.12195
##                fueloil           centralAirNo
##            -8873.13971          -17112.81908
```

```
coef(lm_medium2)
```

```
##            (Intercept)               lotSize                   age
##            28627.73165            9350.45188              47.54722
##             livingArea             pctCollege              bedrooms
##               91.86974             296.50809          -15630.71950
##             fireplaces             bathrooms                 rooms
```

```
##              985.06117           22006.97108            3259.11923
## heatinghot water/steam        heatingelectric           fuelelectric
##             -9429.79463           -3609.98574           -12094.12195
##                 fueloil           centralAirNo
##             -8873.13971          -17112.81908
```

```
# All interactions
# the ()^2 says "include all pairwise interactions"
lm_big = lm(price ~ (. - sewer - waterfront - landValue - newConstruction)^2, data=SaratogaHouses)


####
# Compare out-of-sample predictive performance
####

# Split into training and testing sets
n = nrow(SaratogaHouses) # number of rows
n_train = round(0.8*n)  # round to nearest integer
n_test = n - n_train
train_cases = sample.int(n, n_train, replace=FALSE)
test_cases = setdiff(1:n, train_cases)
saratoga_train = SaratogaHouses[train_cases,]
saratoga_test = SaratogaHouses[test_cases,]

# Fit to the training data
lm1 = lm(price ~ lotSize + bedrooms + bathrooms, data=saratoga_train)
lm2 = lm(price ~ . - sewer - waterfront - landValue - newConstruction, data=saratoga_train)
lm3 = lm(price ~ (. - sewer - waterfront - landValue - newConstruction)^2, data=saratoga_train)

# Predictions out of sample
yhat_test1 = predict(lm1, saratoga_test)
yhat_test2 = predict(lm2, saratoga_test)
yhat_test3 = predict(lm3, saratoga_test)
```

```
## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading
```

```
rmse = function(y, yhat) {
  sqrt( mean( (y - yhat)^2 ) )
}

# Root mean-squared prediction error
rmse(saratoga_test$price, yhat_test1)
```

```
## [1] 82835.17
```

```
rmse(saratoga_test$price, yhat_test2)
```

```
## [1] 68531.43
```

```
rmse(saratoga_test$price, yhat_test3)
```

```
## [1] 184405.9
```

```
# easy averaging over train/test splits

n_train = round(0.8*n)  # round to nearest integer
```

```
n_test = n - n_train

rmse_vals = do(100)*{

  # re-split into train and test cases with the same sample sizes
  train_cases = sample.int(n, n_train, replace=FALSE)
  test_cases = setdiff(1:n, train_cases)
  saratoga_train = SaratogaHouses[train_cases,]
  saratoga_test = SaratogaHouses[test_cases,]

  # Fit to the training data
  lm1 = lm(price ~ lotSize + bedrooms + bathrooms, data=saratoga_train)
  lm2 = lm(price ~ . - sewer - waterfront - landValue - newConstruction, data=saratoga_train)
  lm3 = lm(price ~ (. - sewer - waterfront - landValue - newConstruction)^2, data=saratoga_train)

  lm_dominate = lm(price ~ lotSize + age + livingArea + pctCollege +
                     bedrooms + fireplaces + bathrooms + rooms + heating + fuel +
                     centralAir + lotSize:heating + livingArea:rooms + newConstruction + livingArea:new

  # Predictions out of sample
  yhat_test1 = predict(lm1, saratoga_test)
  yhat_test2 = predict(lm2, saratoga_test)
  yhat_test3 = predict(lm3, saratoga_test)
  yhat_test4 = predict(lm_dominate, saratoga_test)

  c(rmse(saratoga_test$price, yhat_test1),
    rmse(saratoga_test$price, yhat_test2),
    rmse(saratoga_test$price, yhat_test3),
    rmse(saratoga_test$price, yhat_test4))
}
```

```
## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading
```

```
## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading
```

```
## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(lm3, saratoga_test): prediction from a rank-deficient fit
## may be misleading
```

```
rmse_vals
```

```
##           V1       V2        V3       V4
## 1   73433.02 69085.45  71473.16 69013.39
## 2   74783.15 65912.30  63990.03 66043.21
## 3   75916.47 64260.88  64637.23 63618.49
## 4   70012.03 58817.51  63800.41 58521.61
## 5   71175.37 59376.87  71098.31 59575.28
## 6   79444.03 68603.30  68281.47 68479.84
## 7   74664.56 64899.32  85899.18 64381.48
## 8   75308.60 69525.27  96834.70 70169.55
## 9   75366.72 63292.70  71135.05 63424.16
## 10  83635.27 72709.34  75521.84 74500.15
## 11  77429.41 62661.33  73969.83 61807.68
## 12  74249.07 62151.13  69579.21 62356.79
## 13  70334.10 61774.12  63661.13 61651.47
## 14  75666.33 68524.00  90633.07 68180.19
## 15  80141.92 68794.58  68413.08 69016.35
## 16  76809.26 63274.86  66289.72 62280.59
## 17  76898.18 66880.74  75183.03 66757.36
## 18  70224.82 59188.99  63955.72 58731.71
## 19  75963.68 65589.80  62559.60 64570.25
## 20  81246.80 70366.91  70327.36 70438.37
## 21  84219.85 73558.54  73270.69 72883.41
## 22  77656.96 63307.52  66452.38 62105.96
## 23  80384.06 68075.49  95526.22 68750.86
## 24  80468.07 69528.04  97522.43 69110.50
## 25  80023.31 72681.97  71363.63 73567.45
## 26  80666.04 73136.84  83145.80 73277.77
## 27  75681.73 66401.80  66084.84 66778.87
## 28  76521.40 67322.35  65114.96 67888.47
## 29  79419.79 70572.11  69698.86 69513.10
## 30  73642.68 65399.41  67339.92 65360.56
```

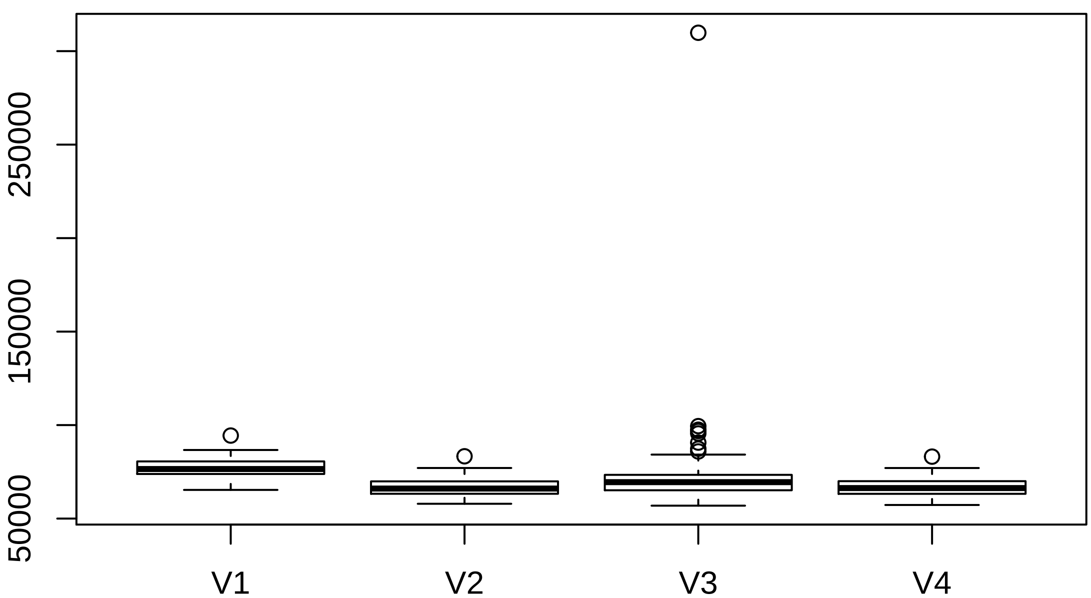
```
## 31  78639.64 70942.05  71618.99 72210.29
## 32  75552.19 63889.56  62734.60 63535.18
## 33  80910.98 69322.75  69381.79 68392.60
## 34  84186.65 74832.22  73510.37 74199.64
## 35  72795.51 62545.98  82869.11 61538.11
## 36  81822.22 73334.55  80388.10 72744.12
## 37  73759.90 60788.96  87394.52 60512.22
## 38  75149.67 63480.72  64041.32 63231.87
## 39  81774.90 71071.60  72855.50 70702.89
## 40  73804.48 64386.04  65654.35 64089.70
## 41  82119.01 69812.95  71519.95 69985.32
## 42  71974.43 61634.09  60600.61 61118.11
## 43  74738.30 66207.27  68278.96 66757.27
## 44  80835.42 68858.37  68610.56 68240.90
## 45  82451.62 71150.46  72562.89 71199.03
## 46  74174.75 65561.39  68617.75 66142.51
## 47  76121.98 69763.55  70246.35 69454.58
## 48  67049.91 58277.49  62231.60 57742.96
## 49  74779.34 65098.33  72738.43 65250.06
## 50  75592.00 67386.72  66209.66 67364.44
## 51  77403.32 67599.61  73253.44 69396.50
## 52  79712.16 69591.53 309818.14 68688.68
## 53  71477.22 60262.88  60316.16 59499.19
## 54  84286.66 69889.93  72748.45 69768.67
## 55  82133.04 69907.08  66872.16 69152.46
## 56  75315.08 64952.55  65450.85 65325.83
## 57  94407.26 83325.03  81160.25 83136.33
## 58  81161.19 72312.17  71679.92 71648.47
## 59  86642.89 77059.08  75802.52 77039.28
## 60  86686.01 71050.13  68536.42 70034.02
## 61  73764.89 64360.28  70009.62 65480.53
## 62  77199.74 66711.84  66748.04 66490.86
## 63  77929.35 68148.46  69222.39 67370.29
## 64  84875.65 73965.86  83249.17 73586.25
## 65  77003.42 70912.08  71081.07 70371.74
## 66  81175.06 69911.88  70652.57 70274.13
## 67  72021.89 62959.53  65177.05 63243.11
## 68  71741.10 61627.98  63272.95 60915.99
## 69  71729.92 64079.61  63892.84 63665.93
## 70  75485.40 64629.59  66335.41 63972.80
## 71  80854.24 69319.74  67685.46 68837.59
## 72  79079.31 65806.57  65622.62 64982.15
## 73  73626.40 62321.39  63534.90 63605.03
## 74  70187.70 64429.87  71173.43 63461.89
## 75  78864.15 64326.64  66569.37 65809.76
## 76  77538.29 62297.55  63950.58 62119.46
## 77  79416.47 67545.93  64063.33 66850.04
## 78  80488.36 73971.46  74439.54 74238.21
## 79  73898.38 63804.04  84223.72 64257.27
## 80  67395.31 58916.31  58576.66 58413.34
## 81  84649.66 74061.29  72041.97 73480.39
## 82  70569.86 60969.77  81188.42 60564.41
## 83  74091.84 65174.86  74364.72 65493.22
## 84  68521.47 57925.51  56913.32 57276.87
```

```
## 85  72494.92 60818.97  99451.08 60608.35
## 86  69753.96 60976.76  60352.38 62245.35
## 87  73183.32 61531.89  61206.17 61005.53
## 88  84662.14 71995.56  71823.42 71532.57
## 89  75280.71 63276.36  67929.84 63682.45
## 90  86539.44 74794.91  75624.62 75538.62
## 91  75938.17 65751.15  67275.08 65019.82
## 92  80491.63 70034.87  76823.49 69997.42
## 93  75229.59 60780.31  60406.74 59957.90
## 94  65343.40 59218.29  63628.34 59814.81
## 95  81856.27 71238.78  71000.66 71039.02
## 96  78345.83 64651.67  68235.97 64266.96
## 97  77424.44 67765.57  70403.65 70173.80
## 98  75662.01 63107.54  64997.00 62175.09
## 99  84652.79 75626.64  80737.44 75348.45
## 100 76486.10 64965.99  64061.88 64409.87
```

```r
colMeans(rmse_vals)
```

```
##       V1       V2       V3       V4
## 77142.97 66746.82 73284.14 66624.33
```

```r
boxplot(rmse_vals)
```



Attempt at "hand-building" a model for price that outperforms the "medium" model that we considered in class by using combinations of transformations, polynomial terms, and interactions:

```r
str(SaratogaHouses)
```

```
## 'data.frame':    1728 obs. of  16 variables:
##  $ price          : int  132500 181115 109000 155000 86060 120000 153000 170000 90000 122900 ...
##  $ lotSize        : num  0.09 0.92 0.19 0.41 0.11 0.68 0.4 1.21 0.83 1.94 ...
##  $ age            : int  42 0 133 13 0 31 33 23 36 4 ...
##  $ landValue      : int  50000 22300 7300 18700 15000 14000 23300 14600 22200 21200 ...
##  $ livingArea     : int  906 1953 1944 1944 840 1152 2752 1662 1632 1416 ...
##  $ pctCollege     : int  35 51 51 51 51 22 51 35 51 44 ...
##  $ bedrooms       : int  2 3 4 3 2 4 4 4 3 3 ...
##  $ fireplaces     : int  1 0 1 1 0 1 1 1 0 0 ...
##  $ bathrooms      : num  1 2.5 1 1.5 1 1 1.5 1.5 1.5 1.5 ...
##  $ rooms          : int  5 6 8 5 3 8 8 9 8 6 ...
##  $ heating        : Factor w/ 3 levels "hot air","hot water/steam",..: 3 2 2 1 1 1 2 1 3 1 ...
##  $ fuel           : Factor w/ 3 levels "gas","electric",..: 2 1 1 1 1 1 3 3 2 1 ...
##  $ sewer          : Factor w/ 3 levels "septic","public/commercial",..: 1 1 2 1 2 1 1 1 1 3 ...
##  $ waterfront     : Factor w/ 2 levels "Yes","No": 2 2 2 2 2 2 2 2 2 2 ...
##  $ newConstruction: Factor w/ 2 levels "Yes","No": 2 2 2 2 1 2 2 2 2 2 ...
##  $ centralAir     : Factor w/ 2 levels "Yes","No": 2 2 2 2 1 2 2 2 2 2 ...
```

```r
# New variables for "hand-built" model

SaratogaHouses$ConstructionCost <- SaratogaHouses$price - SaratogaHouses$landValue
SaratogaHouses$waterfrontDummy <- ifelse(SaratogaHouses$waterfront == "yes", 1,0)
SaratogaHouses$newConstructionDummy <- ifelse(SaratogaHouses$age == "yes", 1,0)
SaratogaHouses$centralAirDummy <- ifelse(SaratogaHouses$age == "yes", 1,0)

str(SaratogaHouses)
```

```
## 'data.frame':    1728 obs. of  20 variables:
##  $ price               : int  132500 181115 109000 155000 86060 120000 153000 170000 90000 122900 ..
##  $ lotSize             : num  0.09 0.92 0.19 0.41 0.11 0.68 0.4 1.21 0.83 1.94 ...
##  $ age                 : int  42 0 133 13 0 31 33 23 36 4 ...
##  $ landValue           : int  50000 22300 7300 18700 15000 14000 23300 14600 22200 21200 ...
##  $ livingArea          : int  906 1953 1944 1944 840 1152 2752 1662 1632 1416 ...
##  $ pctCollege          : int  35 51 51 51 51 22 51 35 51 44 ...
##  $ bedrooms            : int  2 3 4 3 2 4 4 4 3 3 ...
##  $ fireplaces          : int  1 0 1 1 0 1 1 1 0 0 ...
##  $ bathrooms           : num  1 2.5 1 1.5 1 1 1.5 1.5 1.5 1.5 ...
##  $ rooms               : int  5 6 8 5 3 8 8 9 8 6 ...
##  $ heating             : Factor w/ 3 levels "hot air","hot water/steam",..: 3 2 2 1 1 1 2 1 3 1 ...
##  $ fuel                : Factor w/ 3 levels "gas","electric",..: 2 1 1 1 1 1 3 3 2 1 ...
##  $ sewer               : Factor w/ 3 levels "septic","public/commercial",..: 1 1 2 1 2 1 1 1 1 3 ...
##  $ waterfront          : Factor w/ 2 levels "Yes","No": 2 2 2 2 2 2 2 2 2 2 ...
##  $ newConstruction     : Factor w/ 2 levels "Yes","No": 2 2 2 2 1 2 2 2 2 2 ...
##  $ centralAir          : Factor w/ 2 levels "Yes","No": 2 2 2 2 1 2 2 2 2 2 ...
##  $ ConstructionCost    : int  82500 158815 101700 136300 71060 106000 129700 155400 67800 101700 ...
##  $ waterfrontDummy     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ newConstructionDummy: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ centralAirDummy     : num  0 0 0 0 0 0 0 0 0 0 ...
```

```r
HeatingElectric <- SaratogaHouses[grep("electric", SaratogaHouses$heating), ]
#View(HeatingElectric)
#str(HeatingElectric)
```

```r
HeatingSteam <- SaratogaHouses[grep("hot water/steam", SaratogaHouses$heating), ]
#View(HeatingSteam)
#str(HeatingSteam)

HeatingHotAir <- SaratogaHouses[grep("hot air", SaratogaHouses$heating), ]
#View(HeatingHotAir)
#str(HeatingHotAir)

FuelOil <- SaratogaHouses[grep("oil", SaratogaHouses$fuel), ]
#View(FuelOil)
#str(FuelOil)

FuelGas <- SaratogaHouses[grep("gas", SaratogaHouses$fuel), ]
#View(FuelGas)
#str(FuelGas)

FuelElectric <- SaratogaHouses[grep("electric", SaratogaHouses$fuel), ]
#View(FuelElectric)
#str(FuelElectric)

SewerSeptic <- SaratogaHouses[grep("septic", SaratogaHouses$sewer), ]
#View(SewerSeptic)
#str(SewerSeptic)

SewerPublicCommercial <- SaratogaHouses[grep("public/commercial", SaratogaHouses$sewer), ]
#View(SewerPublicCommercial)
#str(SewerPublicCommercial)

SewerNone <- SaratogaHouses[grep("none", SaratogaHouses$sewer), ]
#View(SewerNone)
#str(SewerNone)

#Defining the models

#Baseline model

lm_medium = lm(price ~ lotSize + age + livingArea + pctCollege + bedrooms +
              fireplaces + bathrooms + rooms + heating + fuel + centralAir, data=SaratogaHouses)

#Hand-built Model

lm_handbuilt = lm(price ~ lotSize + age + livingArea + pctCollege + bedrooms +
                 fireplaces + bathrooms + rooms + heating + fuel + centralAir + ConstructionCost +
                 ConstructionCost*landValue + newConstructionDummy*landValue + newConstructionDummy*lot
                 pctCollege*age + bathrooms*bedrooms, data = SaratogaHouses)


#Defining only the numerics of the train-test data sets
N = nrow(SaratogaHouses)
train = round(0.8*N)
test = (N-train)

#Defining the function
```

```r
rmse = function(y, yhat) {
  sqrt( mean( (y - yhat)^2 ) )
}

#Rmse iterations
rmse1 <- NULL
rmse2 <- NULL


for (i in seq(1:00)){
  #Choosing data for training and testing
  train_cases = sample.int(N, train, replace=FALSE)
  test_cases = setdiff(1:N, train_cases)

  #Define the train-test data sets (for all X's and Y)
  saratoga_train = SaratogaHouses[train_cases,]
  saratoga_test = SaratogaHouses[test_cases,]

  #Training
  #Baseline model

  lm_medium = lm(price ~ lotSize + age + livingArea + pctCollege + bedrooms +
              fireplaces + bathrooms + rooms + heating + fuel + centralAir, data=saratoga_train)

  #Hand-built Model
  lm_handbuilt = lm(price ~ lotSize + age + livingArea + pctCollege + bedrooms +
                 fireplaces + bathrooms + rooms + heating + fuel + centralAir + ConstructionCost +
                 ConstructionCost*landValue + newConstructionDummy*landValue + newConstructionDummy*lotS
                 pctCollege*age + bathrooms*bedrooms, data = saratoga_train)

  #Testing
  yhat_test1 = predict(lm_medium, saratoga_test)
  yhat_test2 = predict(lm_handbuilt, saratoga_test)

  #Run it on the actual and the predicted values
  rmse1[i]= rmse(saratoga_test$price, yhat_test1)
  rmse2[i]= rmse(saratoga_test$price, yhat_test2)

}
```

```
## Warning in predict.lm(lm_handbuilt, saratoga_test): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(lm_handbuilt, saratoga_test): prediction from a rank-
## deficient fit may be misleading
```

```r
mean(rmse1)
```

```
## [1] 66576.58
```

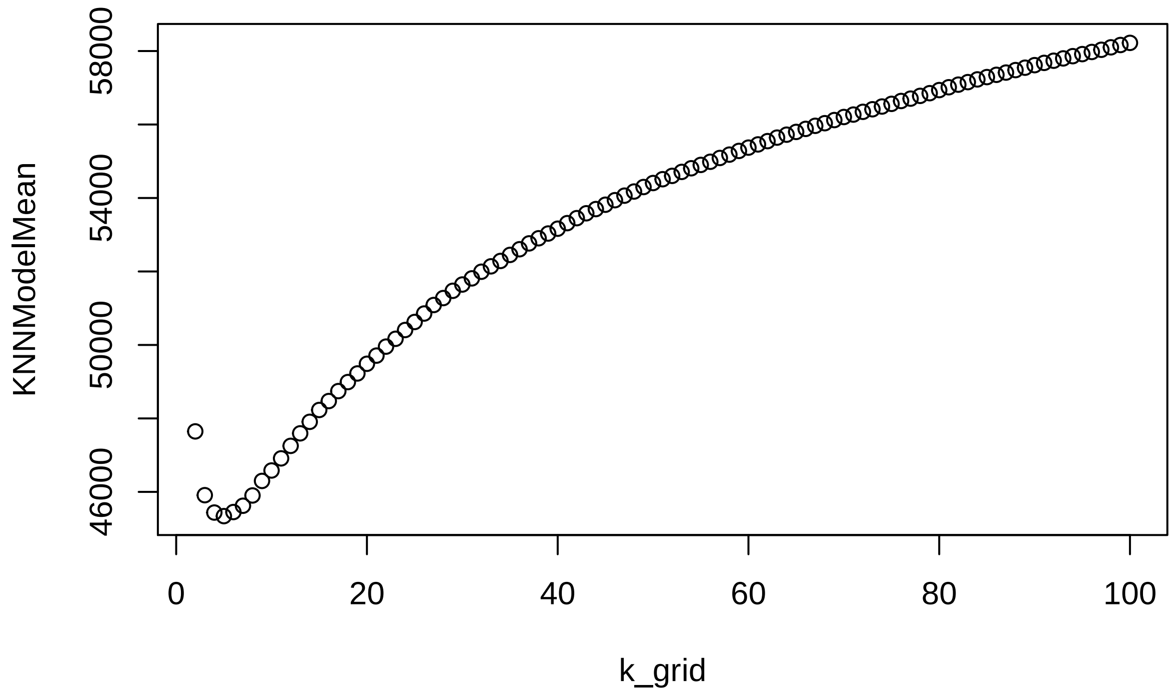
```r
mean(rmse2)
```

```
## [1] 2.33846e-11
```

The variable ConstructionCost and the interations of ConstructionCost and landValue, newConstruction-Dummy and landValue, newConstructionDummy and lotSize, pctCollege and age, and bathrooms and bedrooms, all seem to be especially strong drivers of house prices.

Attempt at turning my hand-built linear model into a better-performing KNN model:

```r
# K-Nearest Neighbors Model

#Defining train-test sets for the hand-built regression model

KNNModel = do(100)*{
  N = nrow(SaratogaHouses)
  train = round(0.8*N)
  test = (N-train)

  train_cases = sample.int(N, train, replace=FALSE)
  test_cases = setdiff(1:N, train_cases)

  saratoga_train = SaratogaHouses[train_cases,]
  saratoga_test = SaratogaHouses[test_cases,]

  Xtrain = model.matrix(~ lotSize + age + livingArea + pctCollege + bedrooms +
               fireplaces + bathrooms + rooms + heating + fuel + centralAir + ConstructionCost
               - 1, data=saratoga_train)
  Xtest = model.matrix(~ lotSize + age + livingArea + pctCollege + bedrooms +
               fireplaces + bathrooms + rooms + heating + fuel + centralAir + ConstructionCost
               - 1, data=saratoga_test)

  Ytrain = saratoga_train$price
  Ytest = saratoga_test$price

  #Scaling the features (Standardization)
  scale_train = apply(Xtrain, 2, sd)
  Xtilde_train = scale(Xtrain, scale = scale_train)
  Xtilde_test = scale(Xtest, scale = scale_train)

  #The for loop
    k_grid = seq(2,100)
    rmse_grid = foreach(K = k_grid, .combine='c') %do% {
      KNNModel = knn.reg(Xtilde_train, Xtilde_test, Ytrain, k=K)
    rmse(Ytest, KNNModel$pred)
  }
}
KNNModelMean = colMeans(KNNModel)

#Plotting
plot(k_grid, KNNModelMean)
abline(h=rmse(Ytest, yhat_test2))
```



We conclude that variables giving the same information that is completely captured by another variable do not need to be included in the model. For example, the variable 'newConstruction' is not essential because we can just look at the value of the age variable of the house and if its' value is 0 then the house is newly built. When a variable does not completely capture all the information about the house then we should not eliminate it withoug giving it any thought because then we may lose some important information. For example, we should not eliminate bathrooms and bedrooms variables because knowing how many of bathrooms and bedrooms specifically is important for buyers which is not fully captured by the rooms variable. On the other hand, we cannot eliminate rooms and only have bedrooms and bathrooms because bedrooms and bathrooms are not the only type of rooms that effects house prices. Other types of rooms such as laundry room, storeroom, sunroom etc. are also included in rooms and how many rooms besides bathrooms and bedrooms are important in determining house prices.

Additionally, we have found that newer houses are bigger and are correlated with an increase in pricing. Also, it appears as if the age of the house is correlated with the percentage of college graduates living in the neighborhood and the higher the age and/or percent of college graduates, the higher is the predicted price.