Saratoga Houses Price Modeling Strategies for the Local Taxing Authority

Bao Doquang, Dhwanit Agarwal, Akksay Singh and Shristi Singh

March 13, 2020

Classwork:

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

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

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

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

```
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        livingArea     pctCollege
##  Min.   :  5000   Min.   : 0.0000   Min.   :  0.00   Min.   :   200   Min.   : 616   Min.   :20.00
##  1st Qu.:145000   1st Qu.: 0.1700   1st Qu.: 13.00   1st Qu.: 15100   1st Qu.:1300   1st Qu.:52.00
##  Median :189900   Median : 0.3700   Median : 19.00   Median : 25000   Median :1634   Median :57.00
##  Mean   :211967   Mean   : 0.5002   Mean   : 27.92   Mean   : 34557   Mean   :1755   Mean   :55.57
```

```
##  3rd Qu.:259000   3rd Qu.: 0.5400   3rd Qu.: 34.00   3rd Qu.: 40200   3rd Qu.:2138   3rd Qu.:64.00
##  Max.   :775000   Max.   :12.2000   Max.   :225.00   Max.   :412600   Max.   :5228   Max.   :82.00
##    bathrooms       rooms                     heating           fuel                      sewer          wate
##  Min.   :0.0   Min.   : 2.000   hot air          :1121   gas     :1197   septic            : 503   Yes:
##  1st Qu.:1.5   1st Qu.: 5.000   hot water/steam: 302   electric: 315   public/commercial:1213   No :
##  Median :2.0   Median : 7.000   electric         : 305   oil      : 216   none               :  12
##  Mean   :1.9   Mean   : 7.042
##  3rd Qu.:2.5   3rd Qu.: 8.250
##  Max.   :4.5   Max.   :12.000
```

```r
#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            livingArea
##           28627.73165          9350.45188            47.54722              91.86974
##              bedrooms          fireplaces           bathrooms                 rooms heatinghe
##          -15630.71950           985.06117         22006.97108            3259.11923
##       heatingelectric        fuelelectric             fueloil          centralAirNo
##           -3609.98574        -12094.12195         -8873.13971          -17112.81908
```

```r
coef(lm_medium2)
```

```
##           (Intercept)             lotSize                 age            livingArea
##           28627.73165          9350.45188            47.54722              91.86974
##              bedrooms          fireplaces           bathrooms                 rooms heatinghe
##          -15630.71950           985.06117         22006.97108            3259.11923
##       heatingelectric        fuelelectric             fueloil          centralAirNo
##           -3609.98574        -12094.12195         -8873.13971          -17112.81908
```

```r
# 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
```

```r
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
```

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

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

```
## [1] 73912.25
```

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

```
## [1] 63735.47
```

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

```
## [1] 66443.11
```

```r
# 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
```

```r
  # 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
```

```
rmse_vals
```

```
##            V1       V2        V3       V4
## 1   70213.49 59615.36  65351.40 60441.59
## 2   74423.52 65157.37  68975.44 64186.81
## 3   70199.89 58324.93  61260.83 58198.01
## 4   74686.86 65615.51  65207.59 65078.01
## 5   80154.82 70304.02  72704.03 69465.89
## 6   73995.49 62372.72  62424.18 62547.13
## 7   76714.70 65561.27  91776.22 64988.48
## 8   79277.74 70692.33  70176.87 70193.57
## 9   81314.52 68531.56  94883.00 67730.30
## 10  77100.11 69310.86  70380.89 69875.76
## 11  78415.45 70273.75  70135.52 69833.97
## 12  78930.01 67379.68  71565.58 66644.61
## 13  62427.37 56485.49  61738.79 55896.68
## 14  73001.50 63367.28 104114.57 62895.40
## 15  77070.32 63774.23  62648.17 63214.91
## 16  74679.02 67262.06  69102.96 66674.31
## 17  73916.92 63984.83  63368.08 63652.35
## 18  82123.23 69714.90  68536.64 70573.59
## 19  79138.40 64775.42  67671.84 65049.15
## 20  70503.81 59815.92  59636.30 59713.80
## 21  88740.89 73703.58  83595.83 74904.56
## 22  75929.42 66966.92  66500.23 67554.00
## 23  75775.28 68330.95  71168.98 68065.67
## 24  80713.73 69669.42  72458.50 69075.26
## 25  79834.57 64362.17  68995.91 64313.02
## 26  77554.48 67156.04  66696.53 66879.44
## 27  74908.27 61265.53  65377.20 60570.19
## 28  77965.98 67397.88  65048.75 67097.56
```

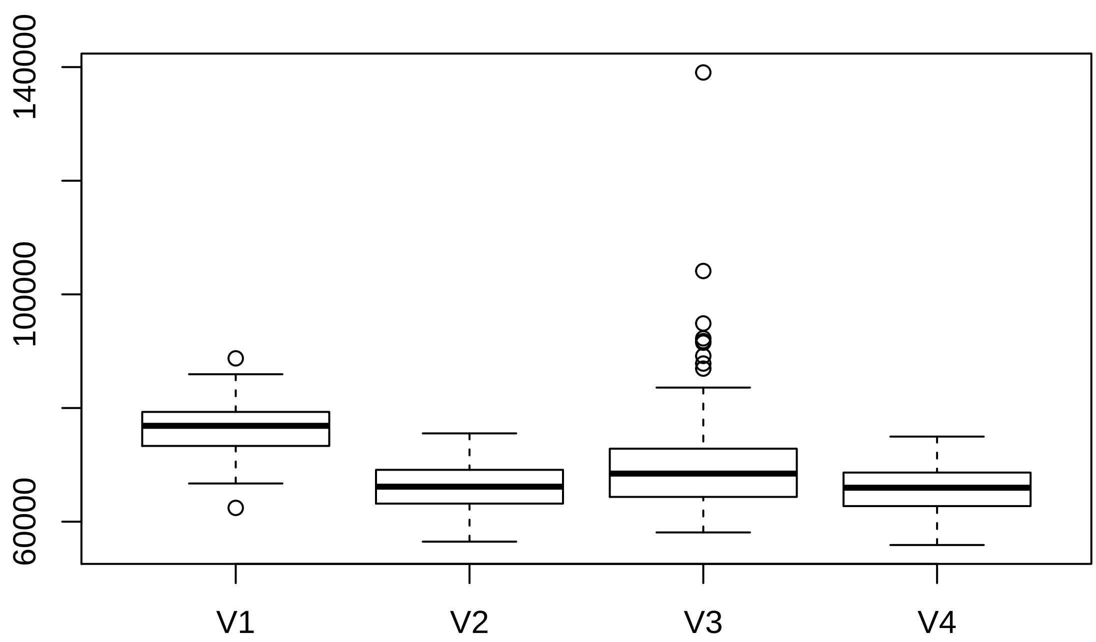
```
## 29  78906.27 67448.06  87827.38 66960.72
## 30  72603.10 64531.55  63903.17 63837.01
## 31  85944.18 71174.07  72460.07 70707.10
## 32  77073.32 67058.78  80303.53 66591.94
## 33  69881.76 57057.22  63569.95 56445.94
## 34  75818.33 66312.82  73753.59 67509.56
## 35  77314.43 66393.77  65961.25 66044.24
## 36  79352.52 68150.71  79599.16 68031.89
## 37  82195.21 71950.25  72954.78 71383.53
## 38  70795.35 60281.02  63055.80 60524.48
## 39  66723.63 57783.62  60341.32 57134.41
## 40  80655.80 69246.35  67808.23 68397.42
## 41  79077.99 69893.93  77894.81 70587.02
## 42  83301.96 71767.73  76138.96 72117.60
## 43  77299.63 67854.80  92264.77 66981.20
## 44  71908.14 66370.18  70651.16 67629.40
## 45  70740.41 63102.72  63567.35 62593.22
## 46  78926.16 68786.52  70716.91 68455.25
## 47  76814.57 63769.57  62845.79 63946.96
## 48  71813.52 62122.25  64183.40 61380.54
## 49  76572.02 62403.78  62150.81 62257.16
## 50  69432.24 63366.64  67284.22 63274.30
## 51  69326.75 60098.67  61568.54 61373.37
## 52  78040.35 68479.52  65840.36 67954.95
## 53  83156.01 70718.79  70205.94 70849.32
## 54  77479.45 70687.16  74002.56 71632.65
## 55  78735.63 66080.08  70551.34 65905.20
## 56  74913.24 69263.99  72226.12 69919.26
## 57  78552.88 71283.78  67850.77 70362.42
## 58  72257.64 60616.15  67462.50 62232.68
## 59  81320.44 66236.13  70458.89 65016.42
## 60  81527.67 68594.71  65219.15 68128.36
## 61  72425.73 63440.99  62882.18 63024.57
## 62  70462.61 64385.55 139049.26 65226.19
## 63  80204.82 67626.00  86936.87 68834.97
## 64  78089.41 71085.18  72736.01 70550.88
## 65  70987.72 60437.82  73921.54 59850.42
## 66  73863.53 62033.17  64277.65 61747.64
## 67  78784.76 66583.95  70389.94 66706.36
## 68  72276.28 60559.57  58106.55 60032.64
## 69  76878.90 64408.15  83245.58 64348.45
## 70  83835.80 72780.39  75812.14 72438.15
## 71  71832.86 63771.38  64218.79 63817.36
## 72  85644.65 70403.74  67431.29 69881.04
## 73  76857.02 62441.31  64443.30 64031.58
## 74  72614.19 61254.17  75033.02 61080.47
## 75  75981.67 67360.69  66237.48 66771.13
## 76  76218.80 64073.61  69804.56 64584.94
## 77  82318.77 67996.13  71162.48 67568.03
## 78  70510.41 59763.49  61127.90 60378.13
## 79  75582.75 64675.06  66251.27 63814.87
## 80  67628.66 57177.97  60544.24 58182.06
## 81  78265.57 71395.93  70443.45 70649.81
## 82  84199.37 72166.67  78619.41 73893.92
```

```
## 83  81222.75 71588.14  76520.17 71203.75
## 84  83882.46 72376.89  89168.87 72445.71
## 85  76350.77 65862.75  82487.58 65216.70
## 86  78830.55 66810.84  67078.08 67258.17
## 87  81231.64 68999.62  68957.96 68359.73
## 88  73651.98 60222.55  66882.55 59964.62
## 89  85375.76 75536.15  76671.80 74976.02
## 90  76030.06 61903.12  60926.31 60945.44
## 91  80071.73 66079.50  91501.99 66328.22
## 92  76992.14 64315.77  70449.61 65525.96
## 93  76515.32 66315.71  66141.78 66858.45
## 94  74495.79 63264.73  63255.65 62270.32
## 95  72215.90 63747.13  66011.45 64080.18
## 96  77531.55 63966.45  66461.81 64561.99
## 97  76440.95 65944.99  66480.08 65857.42
## 98  72121.24 59728.74  61443.35 59101.60
## 99  75087.29 61877.18  62716.70 60648.02
## 100 84390.95 71943.18  68388.55 71137.79
```

```r
colMeans(rmse_vals)
```

```
##       V1       V2       V3       V4
## 76660.74 65863.62 71003.43 65756.33
```

```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)
```

```
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*lotS
                 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 misle

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

```r
mean(rmse1)
```

```
## [1] 65999.49
```

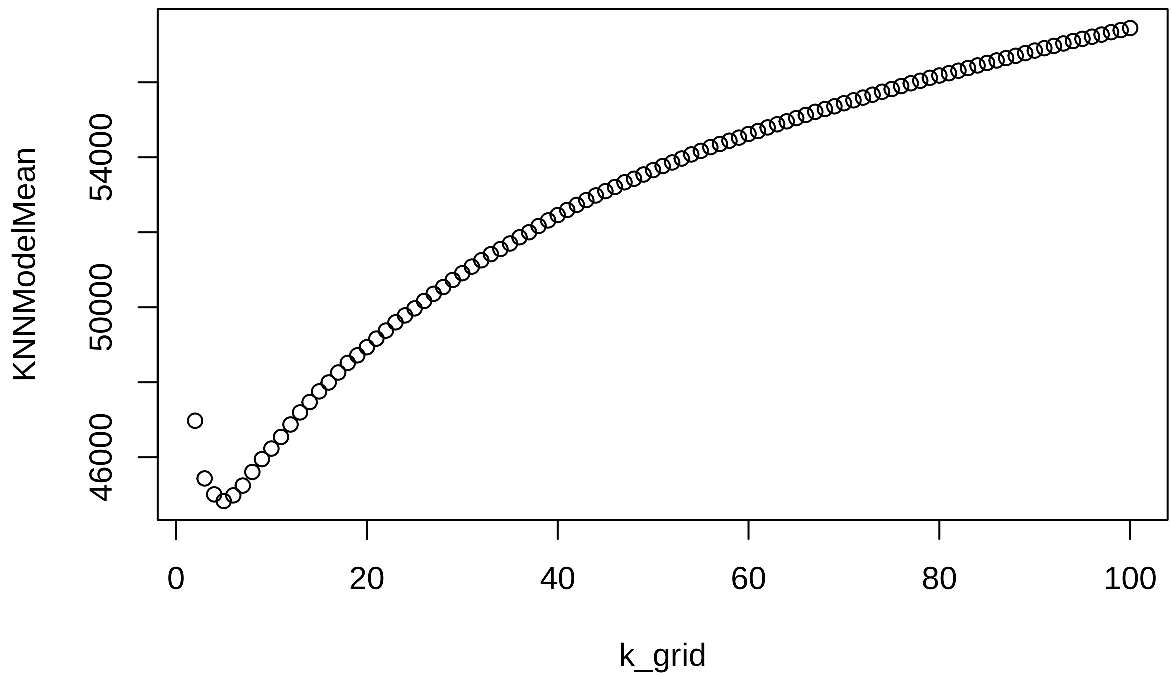
```r
mean(rmse2)
```

```
## [1] 2.965049e-11
```

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

```r
# K-Nearest Neighbors Model
```

```r
#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))
```



Conclusion:

While building our model we realized that when a variable does not completely capture all the information about the house then we should not eliminate it without 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 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 affects 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. We have added one composite variable ConstructionCost and five interaction terms to the medium model to improve the model's predictive performance.The variable ConstructionCost and the interactions of ConstructionCost and landValue, newConstructionDummy and landValue, newConstructionDummy and lotSize, pctCollege and age, and bathrooms and bedrooms, all seem to be especially strong drivers of house prices because addicting these has decreased the rmse from something in the 60,000 to almost 0 depending on random variation.

ConstructionCost is calculated by subtracting landvaule from price and used to represent the cost of building the house on any piece of land. The interaction of ConstructionCost and landvalue is capturing how luxurious houses using high-quality raw materials are built in neighborhoods where land value is high or appreciating. Similarly cheaper houses are built in neighborhoods where land value is low because poorer people live there. The interaction term newConstructionDummy and landvlue is capturing that new constructions happened in neighborhoods where land value is higher or appreciating and the interaction of newConstructionDummy and lotSize is capturing the fact that more new construction happens as lot size increases. The interaction of pctCollege and age captures the fact that the older the houses are, the higher is the percent of the neighborhood that graduated from college.The interaction of bedroom and bathroom captures that the more bedrooms a house the more bathrooms it will have and so the higher will be the price of the house.

From our analysis, it appears that your organization can tax newly built houses slightly more because after analyzing the data we have found that newer houses are bigger and are correlated with an increase in pricing. Normally these houses most probably belong to the rich class of society. We will not recommend you to charge newer built houses a really high rate though because it also appears that 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 price of the house. In most cases, the former effect of newly built houses dominates the latter effect of age and percent of college graduates in the neighborhood.