University of Waterloo

STAT 444

STAT 444 Spring 2019

Group Gengyao_Yuan:

Gengyao Yuan(20613017)

Haohan Li(20610397)

Contents

1 Executive summary

The prediction of Price in Residential Washington D.C.

SALEDATE is the most critical variable when estimating PRICE in all of the three methods. GRADE, LONGITUDE, BATHTRM, GBA,WARD,CNDTH,ZIPCODE,FIRSPLACE,HF_ BATHRM, and EYB are the most important ten variables after SALEDATE. BATHRM & HF_BETHRM, ZIPCODE&LONGITUDE, LANDAREA & LONGITUDE, and SALEDATE & CENSUS_TRCT have interpretations. The variable HF_BATHRM has a quadratic relationship with PRICE.

Overall the boosting method model has the best coefficient and accuracy basing on our implement. We achieve 15 on smoothing, 13 on random forest, 12 on boosting by the due time on kaggle. However, by the time we are finishing this report, our model has been improved a lot. Thus cause the kaggle RMSE is different than the report RMSE.

```
## [1] "RMSE of Prediction On kaggle With Rank"
```

```
##       smoothing random forest   boosting
## RMSE  0.1910393     0.1767595  0.1538500
## Rank 15.0000000    13.0000000 12.0000000
```

```
## [1] "Prediction For This report"
```

```
##      smoothing random forest  boosting
## RMSE 0.1910393     0.1767595 0.1538500
```

2 Itroduction

The goal of the STAT 444 final project is to build three prediction models, which respectively basing on smoothing, random forest and boosting method, to predict the Price variable in a subset of D.C. Residential Properties Kaggle dataset. The error metrics will be presented as RMLSE (Root-Mean-Squared-Logarithmic-Error), which is the Root-Mean-Squared-Error (RMSE) between the (natural) log of the predicted value and the log of the observe.

By comparing the final best-fitted smoothing, random forest and boosting method models. To find out what the difference of importance between different variables, is there any variables has correlation and basing on coefficient and accuracy which model is better.

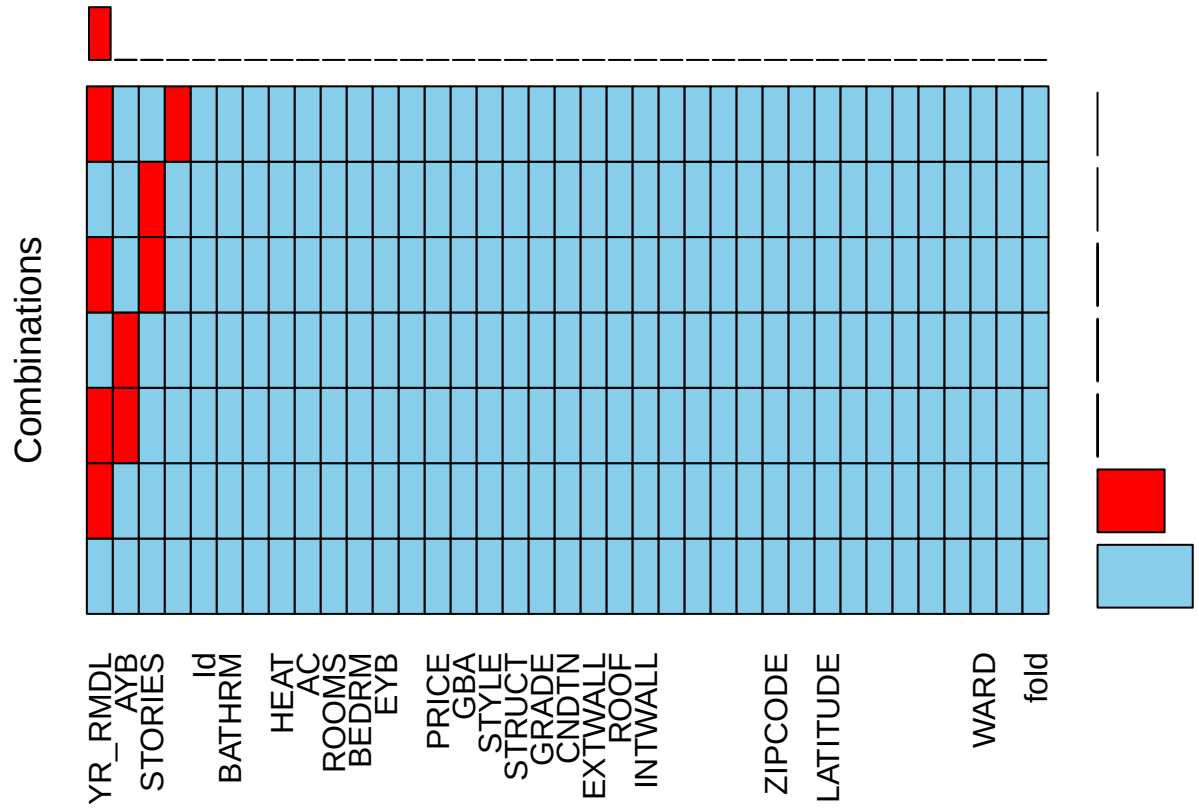
3 Data

The subset of D.C. Residential Properties used in the project contain 37 variables, which are:
Id
BATHRM: Number of Full Bathrooms
HF_BATHRM: Number of Half Bathrooms (no bathtub or shower)
HEAT: Heating
AC: Cooling
ROOMS: Number of Rooms
BEDRM: Number of Bedrooms
AYB: The earliest time the main portion of the building was built
YR_RMDL: Year structure was remodeled
EYB: The year improvement was built more recent than actual year built
STORIES: Number of stories in primary dwelling
SALEDATE: Date of most recent sale
PRICE: Price of most recent sale
GBA: Gross building area in square feet
STYLE: Style
STRUCT: Structure
GRADE: Grade
CNDTN: Condition
EXTWALL: Extrerior wall
ROOF: Roof type
INTWALL: Interior wall
KITCHENS: Number of kitchens
FIREPLACES: Number of fireplaces
USECODE: Property use code
LANDAREA: Land area of property in square feet
FULLADDRESS: Full Street Address
ZIPCODE: Zip Code
NATIONALGRID: Address location national grid coordinate spatial address
LATITUDEP: Latitude
LONGITUDE: Longitude
ASSESSMENT_NBHD: Neighborhood ID
ASSESSMENT_SUBNBHD: Subneighborhood ID
CENSUS_TRACT: Census tract
CENSUS_BLOCK: Census block
WARD: Ward (District is divided into eight wards, each with approximately 75,000 residents)
QUADRANT: City quadrant (NE,SE,SW,NW)
fold: A variable for k-fold corss validation
Since ID ued for indentify PRICE ,and fold is for make gourp like “train” and “test” for satitic learning to build model, there are at most 34 variables.

4 preprocessing

4.1 Fill Missing And NA Values

Although the prediction should close to the original data as much as possible(except overfit situation), missing and NA values are not logic and may case error when building a model.




```
##
##  Variables sorted by number of missings:
##            Variable Count
##             YR_RMDL 16351
##                 AYB    61
##             STORIES    26
##            KITCHENS     1
##                  Id     0
##              BATHRM     0
##           HF_BATHRM     0
##                HEAT     0
##                  AC     0
##               ROOMS     0
##               BEDRM     0
##                 EYB     0
##            SALEDATE     0
##               PRICE     0
##                 GBA     0
##               STYLE     0
##              STRUCT     0
```

```
##               GRADE     0
##               CNDTN     0
##             EXTWALL     0
##                ROOF     0
##             INTWALL     0
##          FIREPLACES     0
##             USECODE     0
##            LANDAREA     0
##         FULLADDRESS     0
##             ZIPCODE     0
##        NATIONALGRID     0
##            LATITUDE     0
##           LONGITUDE     0
##     ASSESSMENT_NBHD     0
##  ASSESSMENT_SUBNBHD     0
##        CENSUS_TRACT     0
##        CENSUS_BLOCK     0
##                WARD     0
##            QUADRANT     0
##                fold     0
```

With the help of Aggregations for missing/imputed values aggr() in VIM library, it is clearly showing that there are 16351 missing data in YR_RMDL, 61 in AYB, 26 in STORIES and 1 in KITCHENS.
Furthermore, by checking the data from excel filter, there are more data in YR_RMDL, ROOF, NATIONALGRID AND CENSUS_BLOOK just label as "no data" or "NA" which we should also consider as missing data.
The kitchen missing can be simple replace by one which is the mode of the variable. And another missing value Should be replaced by the no-missing value which produced by the same variable. An easy way to achieve this is to use sample(data) and set the data from the variable's no-missing values. This way will help the prediction missing data close to the variables' original distribution(if exist), but will lead random into the model may cause estimate a bit different every time. However, by simulating 20 times for smoothing method prediction model, the differences are smaller than 0.001 which is acceptable.

4.2 Outliers And Unlogic Variables

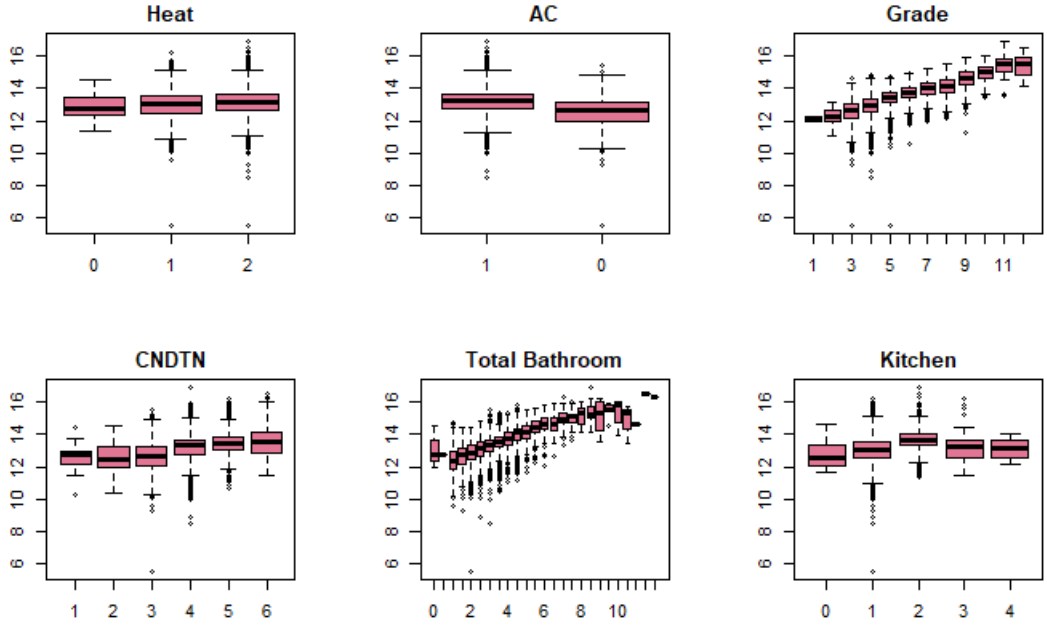
Considering there are 34 possible predictors, signal 2d plots between predictors to PRICE are not appropriate because an outlier for one predictor may be a reasonable value for another predictor.

Since there some extreme values in the data set, it is necessary to modify them before building models.

For example, AYB means The earliest time the main portion of the building was built, and YR_RMDL&EYB means Year structure was remodeled & year improvement was built more recent than actual year built. Which is logically YR_RMDL & EYB should larger than AYB. We remove the unlogic values and fill them with sample().

And the height of buildings in Washington is limited by the Height of Buildings Act. Tallest residential building in Washington, D.C. completed in the city in the 2000s has 14 floors. Thus we should also remove all STORIES " 14 and get the new values. "

4.3 New Feature And Variable types



Basically, by Observation results, the first 5 variables have the strongest relationship with PRICE,but as kitchen increase, PRCIE don't change a lot. Thus in the first part of the smoothing method later, we plug the first 5 variables into predict model first becuase they have strong relationship with PRICE.

Since the implement of smoothing, random forest and boosting method are different. Thus the concreteness data preprocessing will be mainly discussing respectively below.

5 Smoothing methods

The main purpose of using the smoothing method is applying the spline and local regression rule into high dimensional data analyst. In this part, all the parameters automatically selected by s() (low rank thin plate(smoothing) spline), te() (tensor product smoothing spline) and ti()(interaction).

5.1 data preprocessing and modification

Smoothing method is a specific kind of linear(quadratic) method thus its data has more conditions than random-forest method and boosting method. Therefore it is necessary to preprocess the data for smoothing method first.

There are two kinds of data in the data set: numeric and categorical, and some variable can treat as numeric variable since it has significant priority between the levels.

Continuous numeric variables:
BATHROOM, ROOMS, REDRM, AYB, YR_RMDL, EYB, STORIES, STYLE, GBA, SALEDATE, FIREPLACES, LANDAREA, ZIPCODE, LATITUDE, LONGTITUDE, GENUSU_TRACT

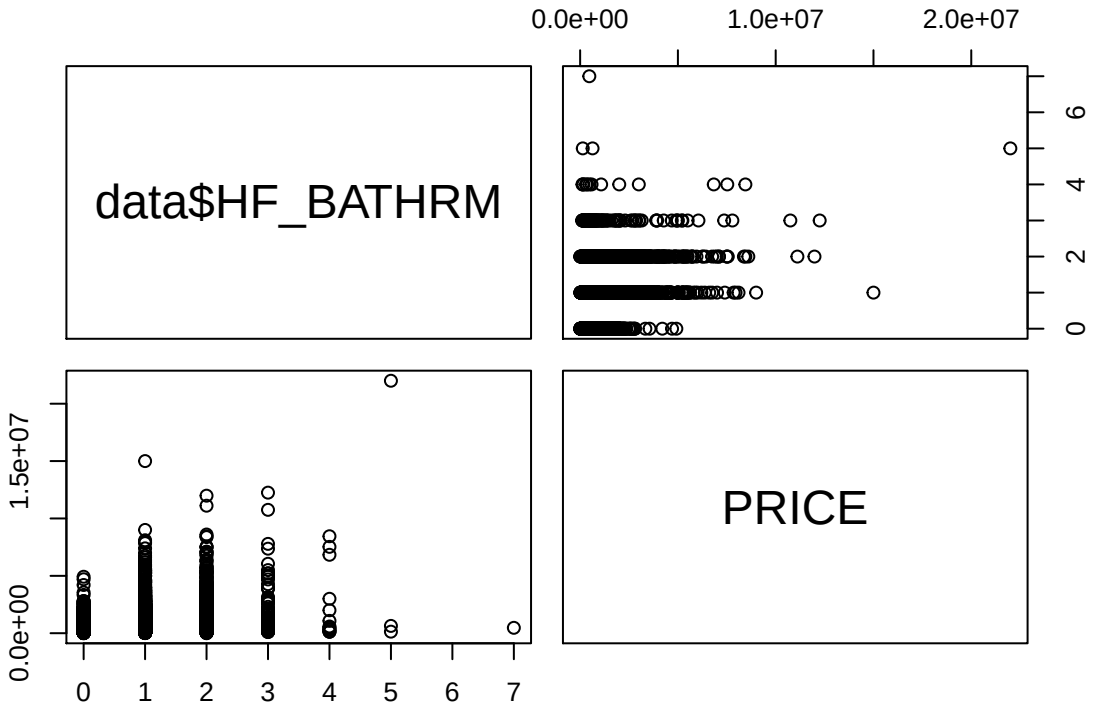
All the variables above are obvious continuous numeric variables without missing or NA data. Consider the large data size, make very variables into smoothing spline it help will increase the prediction accuracy.

numeric variables tranded from string: GRADE, CNDTN

These two variables were saved as a string in the original dataset, but they actually present the quality of the house, which showed have priority for different factors. Thus we transfer these variables to numeric.

```
data$GRADE <- as.numeric(factor(data$GRADE,level=
                                  c('Low Quality', 'Fair Quality', 'Average'
                                    , 'Above Average', 'Good Quality', 'Very Good', 'Excellent',
                                    'Superior', 'Exceptional-A', 'Exceptional-B', 'Exceptional-C',
                                    'Exceptional-D', ordered= TRUE)))
data$CNDTN <- as.numeric(factor(data$CNDTN,
                                level=c('Poor', 'Fair', 'Average',
                                        'Good', 'Very Good','Excellent', ordered= TRUE)))
```

HF_BATHROOM: Since the data levels of half bathroom is only 8, and based on the pairs plot.



Only 4 of the levels actually have most of the data, thus at first try to treat this variable as categorization. However, an error will be reported as 'Error in predict.gam(gam.object, test): 7 not in original fit', this error frequently occurs when we predict categorical variables.

NOT_IN_ORIGINAL_FIT

A frequently occur error when predicting categorical variables. The main reason occurs this error is categorical factor may not obvious in every fold. Thus case if the factor does not exist in 'train' fold but exist in test fold, the trained smooth model won't have an estimate parameter for that factor. This is the reason case the r crash. The way we deal with this kind of problem is replacing the rare showup factor' that not show up in every fold to some common factors.

However, for HF_BATHRM variable, the pair graph clearly shows that there should be a quadratic relationship between HF_BATHRM and PRICE, so treat HF_BATHRM as a numeric variable which has a quadratic relationship.

Categorical variables:
AC, STRUCT, KITCKENS, USECODE, GENSUS_TRACT,WARD,QUADRANT,ASSESSMENT_NBHD
All the Categorical variables above do not have missing data or NA, and all of their factors exist in every fold.

Heat: it is an obvious categorical variable, but NOT_IN_ORIGINAL_FIT error exists, do the following transfer to avoid it.

```
# data$HEAT <- as.character(data$HEAT)
# data$HEAT[data$HEAT=='Air-oil'|
#                   data$HEAT=='Electric Rad'|
#                    data$HEAT=='Evp Cool'|
#                    data$HEAT=='Gravity Furnac'|
#                    data$HEAT=='Ind Unit'|
#                    data$HEAT=='No Data'|
#                    data$HEAT=='Wall Furnace'
#                    ] <- sample((data$HEAT[data$HEAT!='Air-oil'& data$HEAT=='Electric R
#                                           & data$HEAT!='Evp Cool'&
#                                             data$HEAT!='Gravity Furnac'&
#                                             data$HEAT!='Ind Unit'&
#                                             data$HEAT!='No Data'&
#                                             data$HEAT!='Wall Furnace'
#                                           ]),size = 8)
# data$HEAT<-as.factor(data$HEAT)
```

Where we replace the rare obvious data as simple from other data, random drawn exist here, may case every estimate lead to slight different k-cross-validation! And use a similar idea to preprocessing EXTWALL/ROOF/INTWALL

```
#EXTWALL
data$EXTWALL[data$EXTWALL == 'Adobe'|
               data$EXTWALL == 'Default'|
               data$EXTWALL == 'Plywood' ] <-
                         sample((data$EXTWALL[data$EXTWALL !=  'Adobe'
                        & data$EXTWALL != 'Default'& data$EXTWALL != 'Plywood' ]),size =

data$EXTWALL<-as.factor(data$EXTWALL)
# QUARANT
data$QUADRANT[data$QUADRANT == ''] <- sample(data$QUADRANT[data$QUADRANT != ''],size = 6
#INITWALL
data$INTWALL[data$INTWALL == 'Vinyl Comp'] <- 'Carpet'
```

Since there are too many missing data, we avoid estimate ASSESSMENT_SUBNBHD and CENUSU_BLOOK.
FULLADDRESS & NATIONALGRID has too many observations that cannot treat as categorical, but they are also meanless as numerical, so drop them out of estimate.

5.2 estimate single variable

Firstly build a linear regression model for all of the numeric variables as a smooth spline. If the variable is essential in a linear model(variable has less p-value), it also means it will be valuable in the prediction model.

```
Family: gaussian
Link function: identity

Formula:
PRICE ~ s(BATHRM) + s(ROOMS) + s(BEDRM) + s(AYB) + s(YR_RMDL) +
    s(EYB) + s(STORIES) + s(STYLE) + s(FIREPLACES) + s(LANDAREA) +
    s(ZIPCODE) + s(LATITUDE) + s(LONGITUDE) + s(CENSUS_TRACT) +
    GRADE + CNDTN

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -83106      14916  -5.572 2.54e-08 ***
GRADE          68076       2276  29.910  < 2e-16 ***
CNDTN         106001       2800  37.860  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                edf Ref.df       F  p-value
s(BATHRM)     8.896  8.993 448.448  < 2e-16 ***
s(ROOMS)      9.000  9.000  91.706  < 2e-16 ***
s(BEDRM)      8.835  8.982  45.181  < 2e-16 ***
s(AYB)        8.965  8.999 284.705  < 2e-16 ***
s(YR_RMDL)    7.188  8.025  70.933  < 2e-16 ***
s(EYB)        8.691  8.951 354.451  < 2e-16 ***
s(STORIES)    1.168  1.314   2.829 0.077502 .
s(STYLE)      5.867  6.697   3.990 0.000332 ***
s(FIREPLACES) 8.957  8.999 153.251  < 2e-16 ***
s(LANDAREA)   8.981  9.000 525.509  < 2e-16 ***
s(ZIPCODE)    8.896  8.994  20.553  < 2e-16 ***
s(LATITUDE)   8.875  8.995  59.982  < 2e-16 ***
s(LONGITUDE)  8.714  8.977  41.996  < 2e-16 ***
s(CENSUS_TRACT) 8.705  8.971  34.638  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.757   Deviance explained = 75.8%
GCV = 8.1506e+10  Scale est. = 8.127e+10  n = 39520
```

By the p-value of summary, Stories and style have significant larger p-value than others, so probably we have to drop these two variables from the model.
Do the same for categorical variables.

```
Formula:
PRICE ~ s(BATHRM) + s(ROOMS) + s(BEDRM) + HEAT + EXTWALL + ROOF +
    INTWALL + AC + STRUCT + KITCHENS + USECODE + ASSESSMENT_NBHD +
    WARD + QUADRANT

Parametric coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                         644675.3   313032.2   2.059 0.039457 *
HEATAir-Oil                          39985.3   132937.8   0.301 0.763582
HEATElec Base Brd                   135435.9   126542.0   1.070 0.284498
HEATElectric Rad                     58026.3   117913.8   0.492 0.622646
HEATForced Air                       91424.8   113516.2   0.805 0.420599
HEATHot Water Rad                    68591.5   113498.6   0.604 0.545622
HEATHt Pump                          94571.4   114461.7   0.826 0.408680
HEATWarm Cool                        54417.6   113552.9   0.479 0.631780
HEATWater Base Brd                  143128.3   121784.2   1.175 0.239897
EXTWALLBrick Veneer                  49950.5    25824.4   1.934 0.053091 .
EXTWALLBrick/Siding                  -8170.9    20673.6  -0.395 0.692673
EXTWALLBrick/Stone                   58389.4    29053.8   2.010 0.044470 *
EXTWALLBrick/Stucco                   9105.3    27820.4   0.327 0.743451
EXTWALLCommon Brick                    959.8    19760.5   0.049 0.961261
EXTWALLConcrete                      48314.7    72191.5   0.669 0.503336
EXTWALLConcrete Block               -77345.7    78464.6  -0.986 0.324267
EXTWALLFace Brick                    17660.5    30064.6   0.587 0.556927
EXTWALLHardboard                    152138.5    52322.0   2.908 0.003643 **
EXTWALLMetal Siding                  43822.6    80276.6   0.546 0.585141
EXTWALLShingle                       -1900.4    26363.6  -0.072 0.942535
EXTWALLStone                         67462.7    28956.8   2.330 0.019824 *
EXTWALLStone Veneer                  24645.6    38357.6   0.643 0.520539
EXTWALLStone/Siding                  32273.8    32612.1   0.990 0.322364
EXTWALLStone/Stucco                 131590.9    36584.2   3.597 0.000322 ***
EXTWALLStucco                        49474.9    21829.6   2.266 0.023431 *
EXTWALLStucco Block                 -47288.2    84753.4  -0.558 0.576881
EXTWALLVinyl Siding                 -13382.9    20713.4  -0.646 0.518218
EXTWALLWood Siding                   20473.7    21206.8   0.965 0.334336
ROOFClay Tile                        22232.3    23255.4   0.956 0.339075
ROOFComp Shingle                    -24978.4     6366.0  -3.924 8.73e-05 ***
ROOFComposition Ro                   27307.5    50658.2   0.539 0.589853
ROOFConcrete                        -93823.9   339218.1  -0.277 0.782097
ROOFConcrete Tile                  -416502.2   195122.2  -2.135 0.032802 *
ROOFMetal- Cpr                        -648.6    94593.3  -0.007 0.994529
ROOFMetal- Pre                       30897.4    37875.2   0.816 0.414637
ROOFMetal- Sms                      -17507.9     5511.3  -3.177 0.001490 **
ROOFNeopren                          93688.1    14937.2   6.272 3.60e-10 ***
ROOFShake                           -15590.8    20560.9  -0.758 0.448291
ROOFShingle                         -41360.2    25530.1  -1.620 0.105227
ROOFSlate                            33334.8     8333.9   4.000 6.35e-05 ***
ROOFTypical                         -50659.0    43098.1  -1.175 0.239828
ROOFWater Proof                    -140750.2   239883.0  -0.587 0.557378
ROOFWood- FS                       -182078.0   196204.9  -0.928 0.353414
INTWALLCeramic Tile                 -45671.1    66068.3  -0.691 0.489400
INTWALLDefault                       52225.5    70138.6   0.745 0.456516
INTWALLHardwood                      42006.1    10358.3   4.055 5.02e-05 ***
INTWALLHardwood/Carp                  6089.5    11060.1   0.551 0.581920
INTWALLLt Concrete                  -36174.7    58488.3  -0.618 0.536252
INTWALLParquet                       46434.1   138819.7   0.334 0.738010
```

By

the definition of the categorical estimate, r treat every factor as an independent variable, but in our prediction model later we can not only a part of the categorical variable. Since most of the categorical variables in summary(t2) printed above, their factors' p-values are pretty different from each other. We can not decide which variables we want.

It will be helpful to go straight to build the prediction medal and calculate the k-cross-validation.

Firstly, build the smoothing method prediction model only with numeric variables.

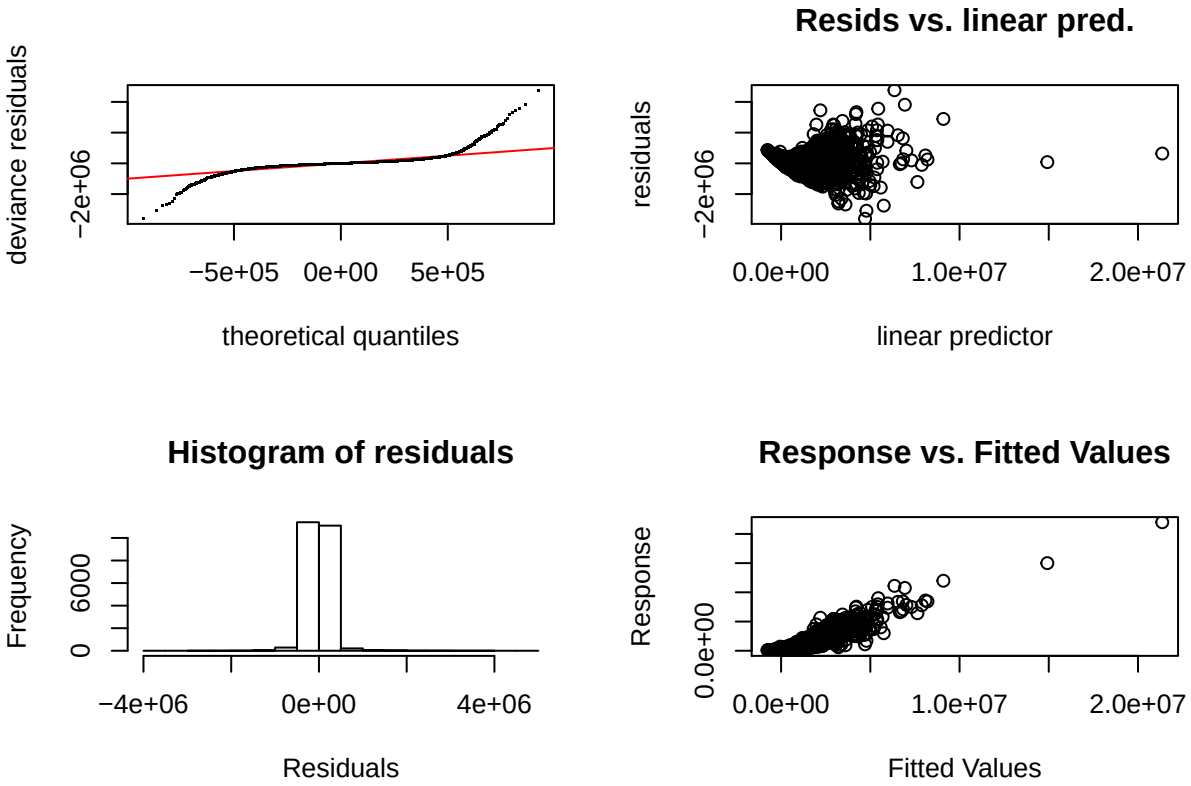
```
## [1] 0.431072
```

Then basing on the k-cross-validation got here, plug in every categoricals to the medal and calculate the k-cross-validation out. If the k-cross-validation increase, delete the categorical variable, otherwise keep it in the model.

```
## [1] 0.2140155
```

At this step the mode contains s(BATHRM)+ HF_BATHRM + I(HF_BATHRMˆ2) + AC + s(ROOMS) + s(BEDRM)+ s(AYB) + s(YR_RMDL)+ s(EYB) + s(SALEDATE) + s(GBA) + CNDTN + KITCHENS + s(LANDAREA) + s(FIREPLACES) + s(ZIPCODE) + CENSUS_TRACT + ASSESSMENT_NBHD + STRUCT + GRADE + WARD + QUADRANT.

A intersting fact is some factors of EXTWALL,INTWALL,ROOF has eˆ-16 p-value but these 3 variables still got kicked out the model becuase they reduce the estimate accuracy proved by k-cross-validation.



```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 38 iterations by steepest
## descent step failure.
## The RMS GCV score gradient at convergence was 227784.3 .
## The Hessian was positive definite.
## Model rank =  186 / 187
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                 k'  edf k-index p-value
## s(BATHRM)     9.00 8.98    1.01   0.715
## s(ROOMS)      9.00 9.00    1.04   1.000
## s(BEDRM)      9.00 8.75    0.98   0.075 .
## s(AYB)        9.00 8.98    0.99   0.325
## s(YR_RMDL)    9.00 6.83    0.99   0.215
## s(EYB)        9.00 8.81    0.98   0.125
## s(SALEDATE)   9.00 8.54    1.00   0.430
## s(GBA)        9.00 8.95    0.98   0.060 .
## s(LANDAREA)   9.00 8.99    1.00   0.595
## s(FIREPLACES) 9.00 8.79    1.12   1.000
## s(ZIPCODE)    9.00 8.31    1.00   0.370
## s(LATITUDE)   9.00 8.02    0.96   0.015 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

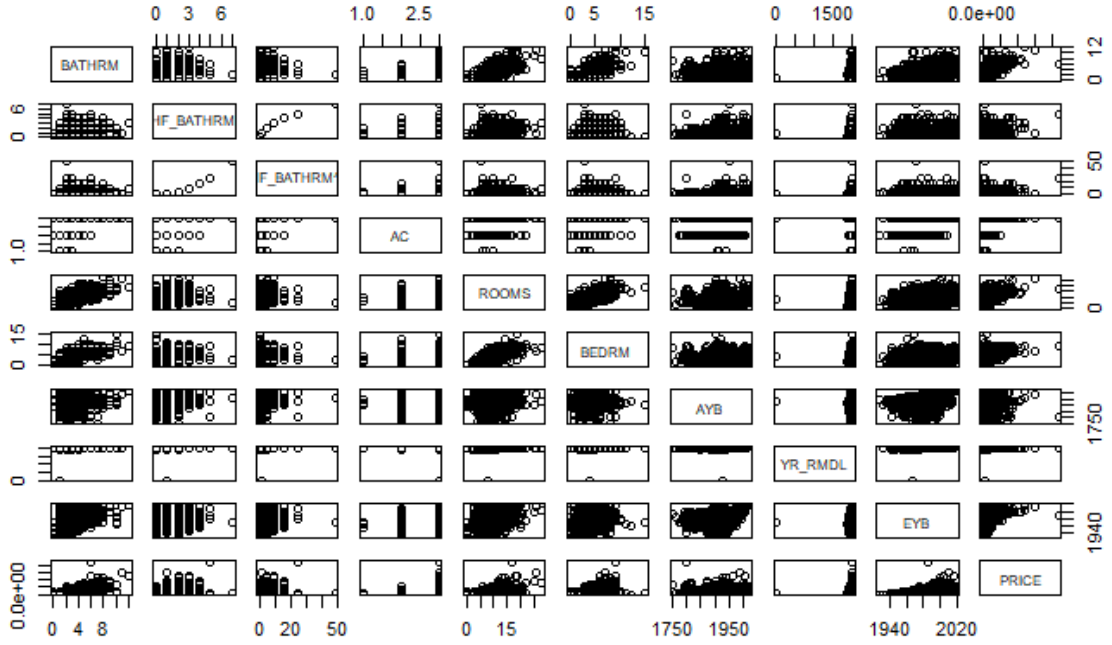
By plot out the gam.check of the current model, The residual of the current model is not perfect for now. And it is clearly showing that the residual response is crowded at the left bottom corner which means there should be quadratic relationships between predictors and PRICE.

5.3 quadratic, correlation and interaction

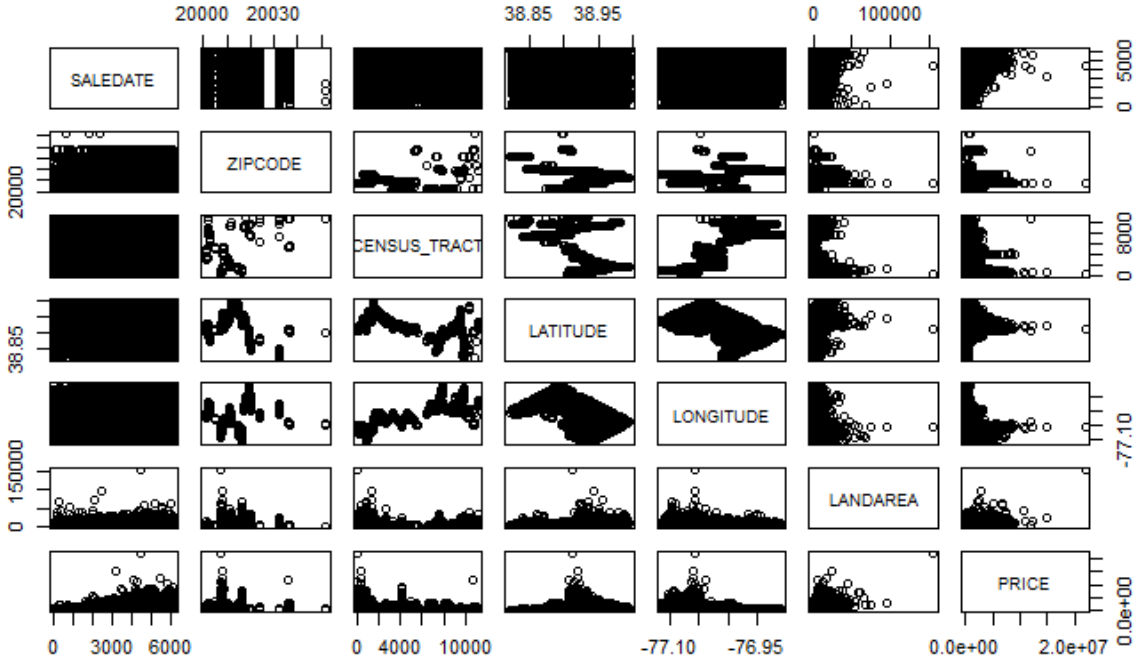
Since the quadratic will obvious between the PRICE and variable, the correlation will visible between variables themselves. A pair of the graph will be helpful to find these relationships.

Since by the actual meaning of latitude & longitude, plot them into the interaction in pairs graph.

pair of (SALEDATE + ZIPCODE + CENSUS_TRACT + LATITUDE + LONGITUDE + LANDAREA + PRICE)



pair of (BATHRM+ HF_BATHRM + I(HF_BATHRM^2) + AC + ROOMS + BEDRM+ AYB + YR_RMDL+EYB + PRICE)



From the pair graph above, it is showing that there could be a correlation between SALEDATE & ZIPCODE, SALE DATE & CENSUS_TRACT, LANDAREA & LONGITUDE, and zipcode & (latitude & longitude).

Modify them into the prediction model.

```
## [1] 0.1911396
```

After output all of these k-cross-validations, it is clearly all of the interaction estimate improve the prediction.

Furthermore, from the pairs graph, it seems like ROOMS, BEDRM and LATITUDE have a quadratic relationship with PRICE. Modify the inner product of ROOMˆ2, BEDRMˆ2, and LATITUDEˆ2 into the model. And print out the resullts.

```
## [1] 0.1911354
```

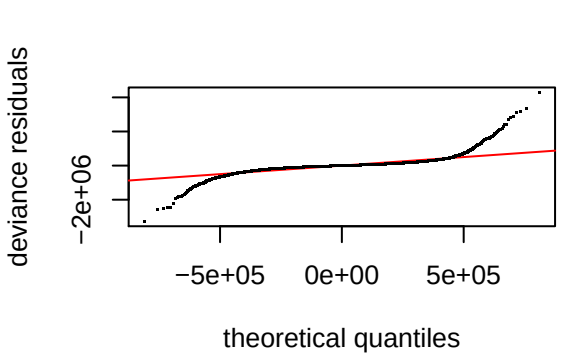
```
## [1] 0.1910852
```

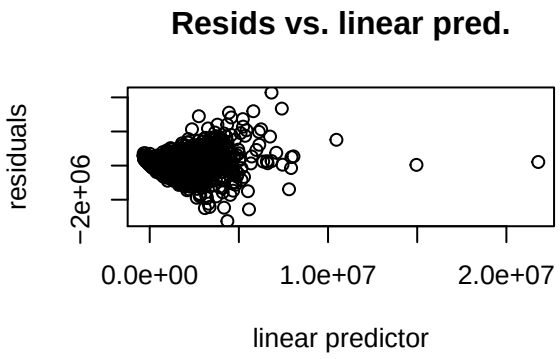
```
## [1] 0.1912356
```

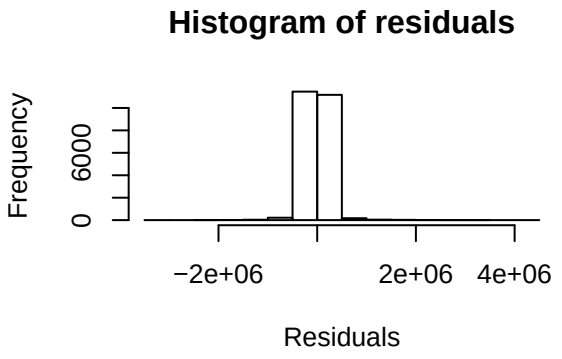
Since the second result(BEDRMˆ2's ouput) decrease the k-cross-validation, the forecast from pair graph is correct, BEDRM does have a quadratic relationship with PRICE.

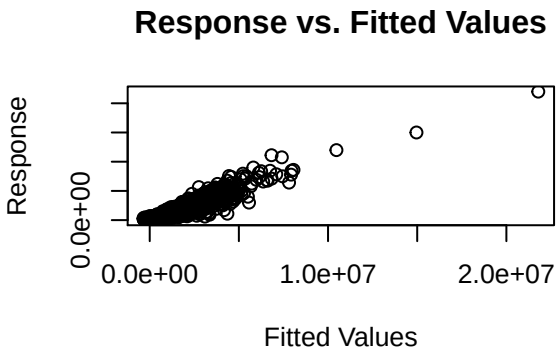
The final smoothing method prediction's k-cross-validation is:

```
## [1] 0.1910852
```










```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 37 iterations.
## The RMS GCV score gradient at convergence was 752760.3 .
```

```
## The Hessian was not positive definite.
## Model rank =  341 / 369
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                                   k'    edf k-index p-value
## s(BATHRM)                       9.00   8.95    1.03   0.960
## s(ROOMS)                        9.00   9.00    0.91  <2e-16 ***
## s(BEDRM)                        9.00   7.58    0.97   0.010 **
## s(AYB)                          9.00   8.90    1.01   0.700
## s(YR_RMDL)                      9.00   1.96    1.01   0.705
## s(EYB)                          9.00   8.98    0.98   0.050 *
## s(SALEDATE)                     9.00   8.93    0.96   0.010 **
## s(GBA)                          9.00   8.92    1.08   1.000
## s(LANDAREA)                     9.00   8.99    0.95  <2e-16 ***
## te(LANDAREA,LONGITUDE):CNDTN   25.00  23.21    1.00   0.450
## s(FIREPLACES)                   9.00   8.99    0.98   0.035 *
## s(ZIPCODE)                      9.00   1.00    1.01   0.835
## te(SALEDATE,ZIPCODE)           22.00  11.34    0.98   0.215
## te(SALEDATE,CENSUS_TRACT)      21.00  18.17    0.87  <2e-16 ***
## te(ZIPCODE,LATITUDE,LONGITUDE) 113.00  85.39    0.87  <2e-16 ***
## s(LATITUDE)                     9.00   1.33    0.98   0.085 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The final prediction of the model has a better residual plot than before, but the responses still crowded left bottom corner. However, if we add more quadratic interaction into the model, the running time will be too long, the coefficiency of the analyst will decrease too much. Thus we stop at the current model as the final smoothing model.

5.4 efficiency against accuracy

Change default bs="tp" to bs="cr" in spline fuction s() can significantly increase the efficiency, but it will decrease the accuracy at the same itme. Thus in order to get the most accurate result, all of the s() in this report use bs = "tp".

Since all of the interactions of quantities measured in differentunits units in this project, we use te(), Tensor produc fuction rather than plug 2 variables in s() function. This also help a lot on improve accuracy.

6 Random Forests

Random Forest is a supervised ensemble learning method for classification and regression. The "Forest" is an ensemble of **Decision Trees** and usually trained with the **bagging** method.

In our case, we use package **ranger** which provides shorter running time than "randomforest"

package

6.1 data preprocessing and modification

Different from the smoothing method model, both Random Forests and Boosting method do not need to worry about categorical variables. Thus all of the data can transfer by as.numeric(). Since Random Forests and Boosting method are tree method, missing values prediction can divide into two groups which are easily used by tree method and by estimate this way can increase accuracy.

6.2 Parameter Tuning

Note that our final model is shown as below.

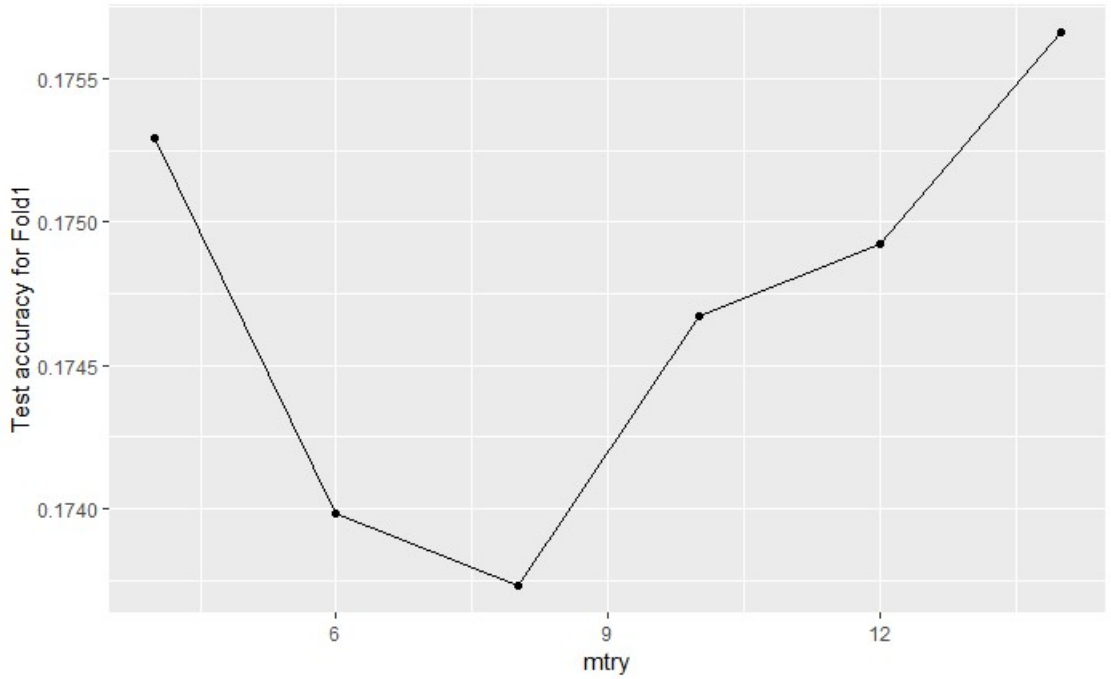
ranger(formula, data = train,importance = "permutation",mtry = 6,num.trees = 750,min.node.size=2,max.depth = 25,seed=12345)

There are three important hyperparameter we can see here: **mtry**, **num.trees**(learning rate), **max.depth** In this section, we assue all of the folds has the smae distribution. So we use fold 1 to tune the hyper-parameter to save some running time

6.2.1 mtry

mtry stands for number of variables to possibly split at in each node.

By fixing other hyperparameters, we can see the different testing error among different mtry. mtry = 8 and mtry = 6 has really similar result and they are better than other values. In our case, we choose mtry = 6

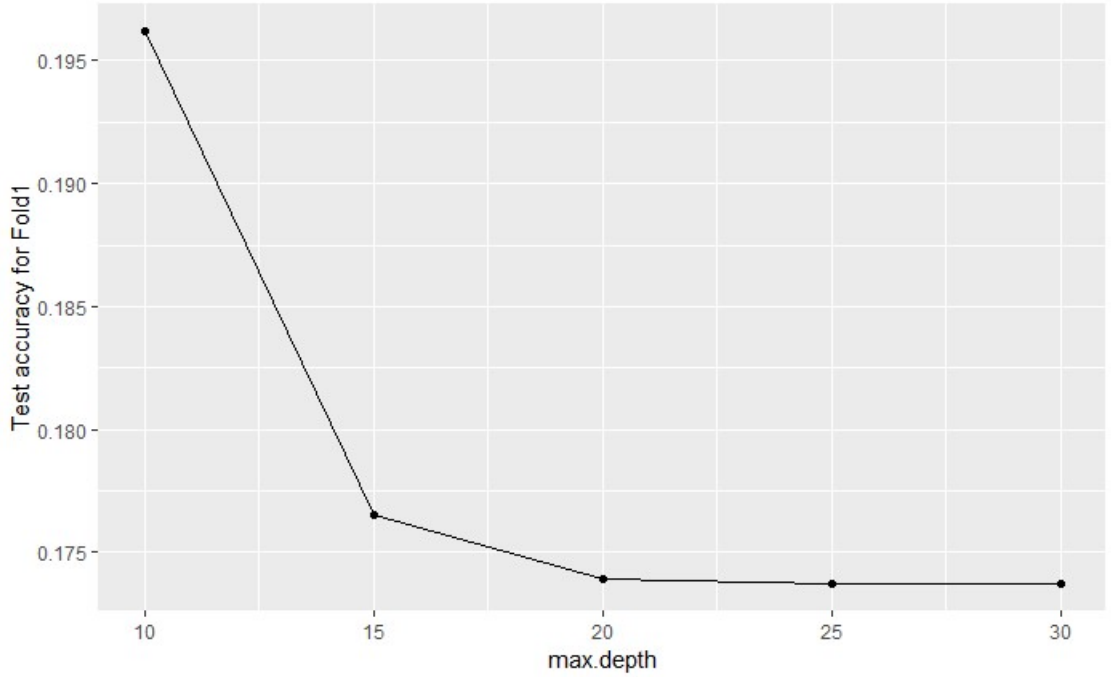


6.2.2 num.trees

num.trees stands for number of trees, usaually the more the merrier. However, the large number of trees could result in slow learning speed.

Based on our hard ware, we choose num.trees=750

6.2.3 Max Depth



From the plot above, we actually can see random forest has a really strong ability to choose the depth, even if we set a large maximum depth, it won't overfit.

Therefore, based on above and running time, we choose max.depth = 25

6.3 Variable Selection

RFCV function recommends to use more than 15 varaiables. Also, based on importance from other methods, such as boosting, we selected variables that result in relativly high importance.

YR_RMDL	SALEDATE	GBA
0.011928558	0.231533859	0.044109965
GRADE	CNDTN	FIREPLACES
0.055659510	0.011170952	0.006573469
LANDAREA	ZIPCODE	NATIONALGRID
0.010948689	0.046357677	0.060345086
LONGITUDE	ASSESSMENT_SUBNBHD	CENSUS_TRACT
0.116367378	0.012472276	0.040281566
QUADRANT	CENSUS_BLOCK_NUM	SALEYEAR
0.003818272	0.047966327	0.140848839

```
  BATHTOTAL ASSESSMENT_SUBNBHD_Modified
0.019042955                  0.005908665
```

7 Boosting

The main purpose of using the boosting method is by giving heavier weight to some data we convert weak learners to stronger ones. In this section, we used **xgboost** (a gradient boosting library) to build the model.

Note: Xgboost could dealing with missing value by itself, so we don't need to worry about that **Note:** The data preprocessing for boosting is same as random forest since they both tree method.

7.1 Parameter Tuning

Note that our final model is shown below.

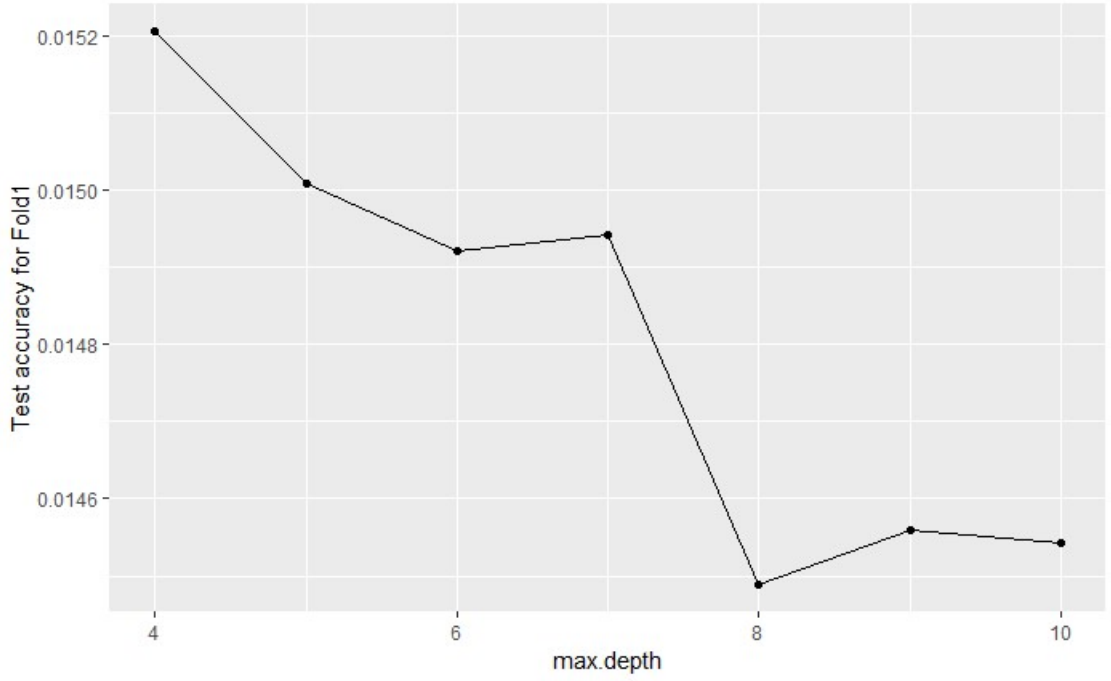
xgb.train(data=dtrain, max.depth=8, eta=0.02, nthread = 4, nrounds=1300, verbose = 0)

There are three important hyperparameters we can see here: **max.depth**, **eta**(learning rate), **nround** In this section, we assume all of the folds has the same distribution. So we use fold 1 to tune the hyper-parameter to save some running time

7.1.1 Max Depth

max_depth is a hyperparameter indicates the maximum depth of a tree.

By fixing other hyperparameters, we try a different number of max depth with fold 2-5 as the training set and fold 1 data as test set we can see the test error changes.



7.1.2 Learning Rate

Learning Rate (eta) is a tricky parameter. In **xgboost**, new trees are created to correct the errors from the predictions of existing trees. A significant learning rate could make the model fit the quicker (imagine we give huge weight to data that have prediction error) A small learning rate could result in slow training and wasting of time.

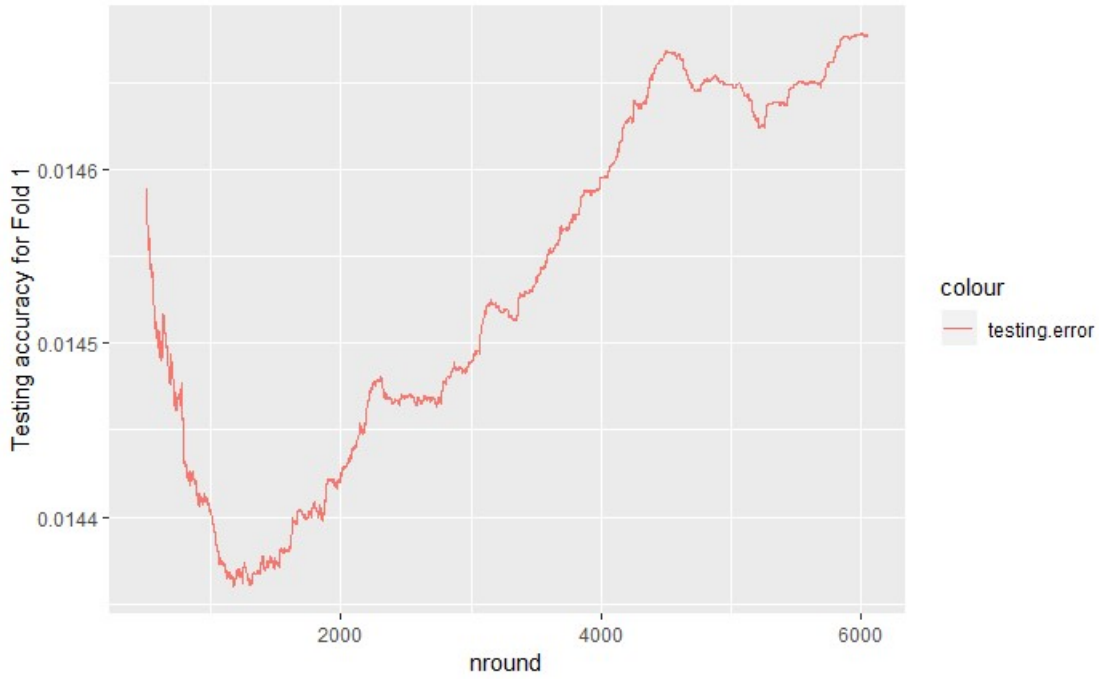
It is pretty common to have small values, usually smaller than **0.2**. Also, this is a highly hardware-dependent parameter. So, based on my computer, I chose **0.02**.

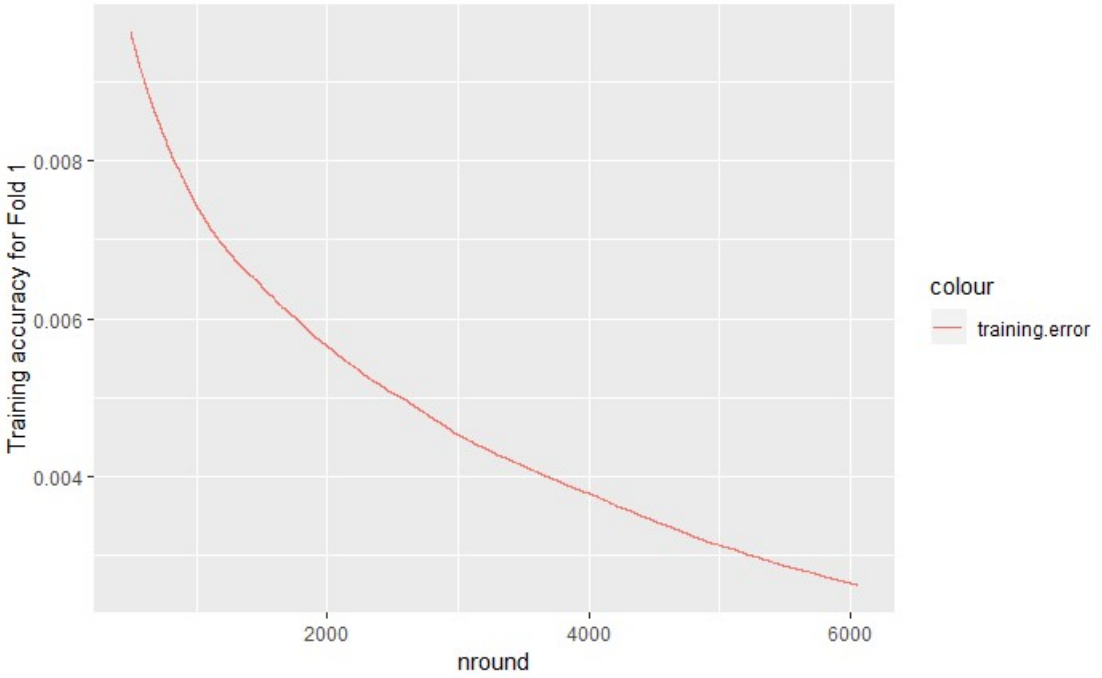
7.1.3 nround

nround stands for the max number of boosting iterations. Too many iterations will cause overfitting, while too few iterations will cause underfitting.

Similar to previous subsections, we used fold 1 data as a training set. So, we fixed all other hyperparameters and print out the training error and testing error at each iteration.

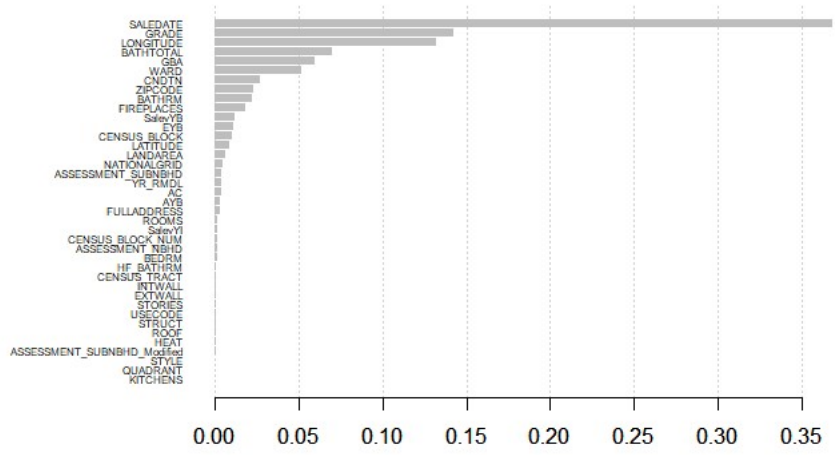
Clearly, after around 1300 round, the model tends to have some overfitting behavior





Clearly, we can see training keep decreasing all the way until the end. However, training error drops sharply till around 1300 iterations and grows up again. This is a sign of overfitting. Therefore, we better stop running it before it gets overfit. So we choose 1300 as nround parameter.

7.2 Variable Selection



Xgboost usually could select variable by itself, but it is always good to see the variable importance and base on that and other models to choose the most effective variables

8 Statistical Conclusions

SALEDATE is the most important variable when estimating PRICE in all of the three methods. GRADE, LONGITUDE, BATHTRM, GBA,WARD,CNDTH,ZIPCODE,FIRSPLACE,HF_BATHRM, and EYB are also pretty important overall. In all the three methods, boosting method has the least RMSE, which means has the best accurate. Basing on our modification smoothing worse than random forest method but it depends on how well smoothing are, the accuracy of smoothing method could go pretty close to original data when adding more and more high dimensional variables into the model. The boosting method has close running time with smoothing, where the random forest almost has double time to them.

9 Future work

We drop FULLADRESS, NATIONALGRID and CENSUS_BLOOK in smoothing method model because these variables are too complex. It is possible to divide these variables into some groups by their actual space located on the city map. Thus they are meaningful and could play an important part in estimate PRICE.
For tree methods, if we get better hardware, we can make more iterations and have a better estimate than this.

10 Contribution

Report: Gengyao Yuan, Haohan Li
Smoothing: Gengyao Yuan
Random Forest: Haohan Li
Boosting: Haohan Li

11 Appendix

11.1 code for 1

```
ta1 <- matrix(c( 0.1910393,0.1767595,0.15385,15,13,12),ncol=3,byrow=TRUE)
colnames(ta1) <- c("smoothing","random forest","boosting")
rownames(ta1) <- c("RMSE","Rank")
print("RMSE of Prediction On kaggle With Rank")
as.table(ta1)

ta2 <- matrix(c( 0.1910393,0.1767595,0.15385),ncol=3,byrow=TRUE)
colnames(ta2) <- c("smoothing","random forest","boosting")
rownames(ta2) <- c("RMSE")
print("Prediction For This report")
as.table(ta2)
```

11.2 code for 3

```
# input of data here


data = read.csv('housing_price.csv')
library('VIM') # Missing value
library(gbm)
library(mgcv)
```

11.3 code for 4.1

```
miss <- aggr(data,prop = FALSE, combined = TRUE, sortVars=TRUE)
```

```
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

data$KITCHENS[is.na(data$KITCHENS)] <- getmode(data$KITCHENS[!is.na(data$KITCHENS)])



med_diff_built_remodel <-floor(median(data$YR_RMDL[!is.na(data$YR_RMDL)&!is.na(data$AYB)
                                    data$AYB[!is.na(data$YR_RMDL)&!is.na(data$AYB)] ))


# Fill AYB and YR_RMDL
data$YR_RMDL[is.na(data$YR_RMDL)&!is.na(data$AYB)] <-data$AYB[is.na(data$YR_RMDL)&!is.na
```

```
#Missing Both
#missing_both <- is.na(data$YR_RMDL)&is.na(data$AYB)

#data$AYB[missing_both]<-floor(median(data$AYB[!is.na(data$AYB)]))

#data$YR_RMDL[missing_both]<-data$AYB[missing_both]- med_diff_built_remodel


#missing_built_have_remodel <- (!is.na(data$YR_RMDL)&is.na(data$AYB))
#data$AYB[missing_built_have_remodel] <- #data$YR_RMDL[missing_built_have_remodel]-med

#data$STORIES[is.na(data$STORIES)]<-floor(median(data$STORIES[!is.na(data$STORIES)]))}
```

```
data[data$YR_RMDL==20,]
data$YR_RMDL[data$YR_RMDL==20]=data$AYB[data$YR_RMDL==20]+med_diff_built_remodel
data$AC[data$AC ==0] <- getmode(data$AC[data$AC !=0])
data$STORIES[data$STORIES>=14] <- floor(median(data$STORIES[data$STORIES<14]))
```

11.4 code for 4.3

```
par(mfrow= c(1, 2))
boxplot(data$PRICE ~ data$HEAT, main = "Heat",col= rgb(0.8,0.1,0.3,0.6))
boxplot(data$PRICE ~ data$AC, main = "AC",col= rgb(0.8,0.1,0.3,0.6))
boxplot(data$PRICE ~ data$GRADE, main = "Grade",col= rgb(0.8,0.1,0.3,0.6))
boxplot(data$PRICE ~ data$CNDTN, main = "CNDTN",col= rgb(0.8,0.1,0.3,0.6))
boxplot(data$PRICE ~ data$BATHTOTAL, main = "Total Bathroom",col= rgb(0.8,0.1,0.3,0.6))
boxplot(data$PRICE ~ data$KITCHENS, main = "Kitchen",col= rgb(0.8,0.1,0.3,0.6))
```

11.5 code for 5.1

```
data$AC <- factor(data$AC,level=c('Y','N'), label=c(1,0))

data$GRADE <- as.numeric(factor(data$GRADE,level=c('Low Quality', 'Fair Quality', 'Avera

data$CNDTN <- as.numeric(factor(data$CNDTN,level=c('Poor', 'Fair', 'Average', 'Good', '



data$NATIONALGRID <- as.numeric(data$NATIONALGRID)

data$ASSESSMENT_NBHD <- as.factor(data$ASSESSMENT_NBHD)

data$STYLE <- as.numeric(data$STYLE)

#HEAT
```

```
#data$STRUCT <- as.numeric(data$STRUCT)

#data$EXTWALL <- as.numeric(data$EXTWALL)
#data$INTWALL <- as.numeric(data$INTWALL)
#data$ROOF <- as.numeric(data$ROOF)
#data$WARD <- as.numeric(data$WARD)
#data$QUADRANT <- as.numeric(data$QUADRANT)



data$HEAT <- as.character(data$HEAT)
data$HEAT[data$HEAT=='Air-oil'|
                data$HEAT=='Electric Rad'|
                 data$HEAT=='Evp Cool'|
                 data$HEAT=='Gravity Furnac'|
                 data$HEAT=='Ind Unit'|
                 data$HEAT=='No Data'|
                 data$HEAT=='Wall Furnace'
                 ] <- sample(data$HEAT[data$HEAT!='Air-oil'& data$HEAT=='Electric Rad' &
                 data$HEAT!='Gravity Furnac'&
                 data$HEAT!='Ind Unit'&
                 data$HEAT!='No Data'&
                 data$HEAT!='Wall Furnace'
                 ],size = 8)
data$HEAT<-as.factor(data$HEAT)


#EXTWALL
#data$EXTWALL[data$EXTWALL == 'Adobe'| data$EXTWALL == 'Default'| data$EXTWALL == 'Ply
#data$EXTWALL<-as.factor(data$EXTWALL)



data$QUADRANT[data$QUADRANT == ""] <- sample(data$QUADRANT[data$QUADRANT != ""],size = 6

#data$INTWALL[data$INTWALL == 'Vinyl Comp'] <- 'Carpet'

library(gbm)
library(mgcv)

# year rebuild
data$SALEYEAR <- as.numeric(substr(data$SALEDATE,0,4))
data$SALEMONTH <- as.numeric(substr(data$SALEDATE,6,7))
data$SALEDATE <- as.numeric(data$SALEDATE)

RMLSE_Score <- function(real,pred, take_log = TRUE){
```

```
  if (take_log){
    print(sqrt(1/length(real)* sum( (log(real) -log(pred))^2 ,na.rm=TRUE )))
  }else{
    print(sqrt(1/length(real)* sum( (real -pred)^2 ,na.rm=TRUE )))
  }
}

pairs(~data$HF_BATHRM + PRICE, data = data)
```

11.6 code for 5.2

```
t1 <- gam(PRICE~ s(BATHRM) + s(ROOMS) + s(BEDRM)+ s(AYB)  + s(YR_RMDL)+ s(EYB) + s(STORI
summary(t1)


t2 <- gam(PRICE~ s(BATHRM) + s(ROOMS) + s(BEDRM) + HEAT +
EXTWALL+ROOF+INTWALL + AC + STRUCT + KITCHENS + USECODE + ASSESSMENT_NBHD + WARD + QUADR
summary(t2, maxsum = 1)


#########################################################################
xg_at1 <- function(fold_num){
  train<-data[data$fold !=fold_num,]
  train <- subset(train, select= - c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))



  train$PRICE = log(train$PRICE)

    gam.object<- gam(PRICE~ s(BATHRM) + s(ROOMS) + s(BEDRM)+ s(AYB)  + s(YR_RMDL)+ s(EYB
         (STYLE) + s(FIREPLACES) + s(LANDAREA) + s(ZIPCODE) + s(LATITUDE) + s(LONGITUDE)


  test<-data[data$fold ==fold_num,]

  test_price <- log(test$PRICE)


  test<-subset(test, select=-c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))

  predict_price<-predict(gam.object,test)

  return (cbind(data[data$fold==fold_num,]$Id,predict_price,test_price))
}
at1_1 <-xg_at1(1)
```

```
at1_2 <-xg_at1(2)
at1_3 <-xg_at1(3)
at1_4 <-xg_at1(4)
at1_5 <-xg_at1(5)
total<-data.frame(rbind(at1_1,at1_2,at1_3,at1_4,at1_5))
RMLSE_Score(total$test_price,total$predict_price, FALSE)

#####################################################################
xg_at2 <- function(fold_num){
  train<-data[data$fold !=fold_num,]
  train <- subset(train, select= - c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))

  train$PRICE = log(train$PRICE)

    gam.object<- gam(PRICE~ s(BATHRM)+ HF_BATHRM + I(HF_BATHRM^2) + AC + s(ROOMS) + s(BE


  test<-data[data$fold ==fold_num,]

  test_price <- log(test$PRICE)


  test<-subset(test, select=-c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))

  predict_price<-predict(gam.object,test)

  return (cbind(data[data$fold==fold_num,]$Id,predict_price,test_price))
}
at1_1 <-xg_at2(1)
at1_2 <-xg_at2(2)
at1_3 <-xg_at2(3)
at1_4 <-xg_at2(4)
at1_5 <-xg_at2(5)
total<-data.frame(rbind(at1_1,at1_2,at1_3,at1_4,at1_5))
RMLSE_Score(total$test_price,total$predict_price, FALSE)

#########################################################
sm <- gam(PRICE~ s(BATHRM)+ HF_BATHRM + I(HF_BATHRM^2) + AC + s(ROOMS) + s(BEDRM)+ s(AYE

gam.check(sm)
```

11.7 code for 5.3

```
pairs(~ SALEDATE + ZIPCODE + CENSUS_TRACT + LATITUDE + LONGITUDE + LANDAREA + PRICE, dat
```

```
pairs(~ BATHRM+ HF_BATHRM + I(HF_BATHRM^2) + AC + ROOMS + BEDRM+ AYB  + YR_RMDL+EYB + PR


xg_at1 <- function(fold_num){
  train<-data[data$fold !=fold_num,]
  train <- subset(train, select= - c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))
  train$PRICE = log(train$PRICE)
    gam.object <- gam(PRICE~ s(BATHRM)+ HF_BATHRM + I(HF_BATHRM^2) + AC + s(ROOMS) + s(B
                      + STRUCT #0.1949005
                      + GRADE
                      + WARD # 0.1945048
                      + QUADRANT #0.1944545 #[1] 0.1915153
                       , data=train)
  test<-data[data$fold ==fold_num,]

  test_price <- log(test$PRICE)


  test<-subset(test, select=-c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))

  predict_price<-predict(gam.object,test)
  return (cbind(data[data$fold==fold_num,]$Id,predict_price,test_price))
}
at1_1 <-xg_at1(1)
at1_2 <-xg_at1(2)
at1_3 <-xg_at1(3)
at1_4 <-xg_at1(4)
at1_5 <-xg_at1(5)
total<-data.frame(rbind(at1_1,at1_2,at1_3,at1_4,at1_5))
RMLSE_Score(total$test_price,total$predict_price, FALSE)



#########################################################################
foul_num <- 1
xg_at1 <- function(fold_num){
  train<-data[data$fold !=fold_num,]
  train <- subset(train, select= - c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))
  train$PRICE = log(train$PRICE)
    gam.object <- gam(PRICE~ s(BATHRM)+ HF_BATHRM + I(HF_BATHRM^2) + AC + s(ROOMS) + s(B
                      + STRUCT #0.1949005
                      + GRADE
                      + WARD # 0.1945048
                      + QUADRANT #0.1944545 #[1] 0.1915153
                      + I(ROOMS^2)
                       , data=train)
```

```
  test<-data[data$fold ==fold_num,]
  test_price <- log(test$PRICE)
  test<-subset(test, select=-c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))
  predict_price<-predict(gam.object,test)
  return (cbind(data[data$fold==fold_num,]$Id,predict_price,test_price))
}
xg_at2<- function(fold_num){
  train<-data[data$fold !=fold_num,]
  train <- subset(train, select= - c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))
  train$PRICE = log(train$PRICE)
    gam.object <- gam(PRICE~ s(BATHRM)+ HF_BATHRM + I(HF_BATHRM^2) + AC + s(ROOMS) + s(
                      + STRUCT #0.1949005
                      + GRADE
                      + WARD # 0.1945048
                      + QUADRANT #0.1944545 #[1] 0.1915153
                      + I(BEDRM^2)
                       , data=train)
  test<-data[data$fold ==fold_num,]
  test_price <- log(test$PRICE)
  test<-subset(test, select=-c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))
  predict_price<-predict(gam.object,test)
  return (cbind(data[data$fold==fold_num,]$Id,predict_price,test_price))
}
xg_at3 <- function(fold_num){
  train<-data[data$fold !=fold_num,]
  train <- subset(train, select= - c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))
  train$PRICE = log(train$PRICE)
    gam.object <- gam(PRICE~ s(BATHRM)+ HF_BATHRM + I(HF_BATHRM^2) + AC + s(ROOMS) + s(
                      + STRUCT #0.1949005
                      + GRADE
                      + WARD # 0.1945048
                      + QUADRANT #0.1944545 #[1] 0.1915153
                      + I(LATITUDE^2)
                       , data=train)
  test<-data[data$fold ==fold_num,]
  test_price <- log(test$PRICE)
  test<-subset(test, select=-c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))
  predict_price<-predict(gam.object,test)
  return (cbind(data[data$fold==fold_num,]$Id,predict_price,test_price))
}
at1_1 <-xg_at1(1)
at1_2 <-xg_at1(2)
at1_3 <-xg_at1(3)
at1_4 <-xg_at1(4)
at1_5 <-xg_at1(5)
total<-data.frame(rbind(at1_1,at1_2,at1_3,at1_4,at1_5))
```

```
RMLSE_Score(total$test_price,total$predict_price, FALSE)
at1_1 <-xg_at2(1)
at1_2 <-xg_at2(2)
at1_3 <-xg_at2(3)
at1_4 <-xg_at2(4)
at1_5 <-xg_at2(5)
total<-data.frame(rbind(at1_1,at1_2,at1_3,at1_4,at1_5))
RMLSE_Score(total$test_price,total$predict_price, FALSE)
at1_1 <-xg_at3(1)
at1_2 <-xg_at3(2)
at1_3 <-xg_at3(3)
at1_4 <-xg_at3(4)
at1_5 <-xg_at3(5)
total<-data.frame(rbind(at1_1,at1_2,at1_3,at1_4,at1_5))
RMLSE_Score(total$test_price,total$predict_price, FALSE)

#####################################################################
#final k
xg_at1 <- function(fold_num){
  train<-data[data$fold !=fold_num,]
  train <- subset(train, select= - c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))
  train$PRICE = log(train$PRICE)
gam.object <- gam(PRICE~ s(BATHRM)+ HF_BATHRM + I(HF_BATHRM^2) + AC + s(ROOMS) + s(BEDRM
                + STRUCT
                + GRADE
                + WARD
                + QUADRANT
                + I(BEDRM^2)
                , data=train)

  test<-data[data$fold ==fold_num,]

  test_price <- log(test$PRICE)


  test<-subset(test, select=-c(Id,fold, ASSESSMENT_SUBNBHD,FULLADDRESS))

  predict_price<-predict(gam.object,test)
  return (cbind(data[data$fold==fold_num,]$Id,predict_price,test_price))
}
at1_1 <-xg_at1(1)
at1_2 <-xg_at1(2)
at1_3 <-xg_at1(3)
at1_4 <-xg_at1(4)
at1_5 <-xg_at1(5)
```

```
total<-data.frame(rbind(at1_1,at1_2,at1_3,at1_4,at1_5))
RMLSE_Score(total$test_price,total$predict_price, FALSE)


mk <- gam(PRICE~ s(BATHRM)+ HF_BATHRM + I(HF_BATHRM^2) + AC + s(ROOMS) + s(BEDRM)+ s(AYE
                      + STRUCT
                      + GRADE
                      + WARD
                      + QUADRANT
                      + I(BEDRM^2)
                      , data=data)

gam.check(mk)
```

11.8 Code for boosting

```
data <- read.csv('../data/housing_price.csv', header = TRUE)

# There is a 20 value is YR_RMDL, apparently it is a wrong data
# Fill it
yr_rmdl_20_index <- data$YR_RMDL==20&!is.na(data$YR_RMDL)
data[yr_rmdl_20_index,]
data$YR_RMDL[yr_rmdl_20_index]=data$AYB[yr_rmdl_20_index]+med_diff_built_remodel

# Wrong Data in AC
data$AC[data$AC ==0] <- getmode(data$AC[data$AC !=0])
# Wrong Data in STORIES

data$STORIES[data$STORIES>=14] <-14 #floor(median(data$STORIES[data$STORIES<14]))


trim <- function (x) gsub("^\\s+|\\s+$", "", x)
# AC
data$AC <- factor(data$AC,level=c('Y','N'), label=c(1,0))
# GRADE
data$GRADE <- as.numeric(factor(data$GRADE,level=c(
'Low Quality', 'Fair Quality', 'Average', 'Above Average', 'Good Quality', 'Very Good',
data$CNDTN <- as.numeric(factor(data$CNDTN,level=c('Poor', 'Fair', 'Average', 'Good', 'V

# HEAT
data$HEAT <- as.character(data$HEAT)
data$HEAT[data$HEAT=='Air Exchng'|
                data$HEAT=='Elec Base Brd'|
                 data$HEAT=='Evp Cool'|
                 data$HEAT=='Hot Water Rad'|
```

```
                  data$HEAT=='Ind Unit'|
                  data$HEAT=='Wall Furnace'|
                  data$HEAT=='Water Base Brd'|
                  data$HEAT=='Ht Pump'|
                  data$HEAT=='Gravity Furnac'
                  ] <- 1
data$HEAT[data$HEAT=='Air-Oil'|
                 data$HEAT=='Electric Rad'|
                  data$HEAT=='Forced Air'|
                 data$HEAT=='Warm Cool'
                  ] <- 2
data$HEAT[data$HEAT=='No Data']<-0
data$HEAT<-as.numeric(data$HEAT)

# CENSUS_TRACT, CENSUS_BLOCK => CENSUS_BLOCK_NUM
data$CENSUS_BLOCK_NUM <- as.numeric(trim(gsub('\\s+',
                                              '',
                                            data$CENSUS_BLOCK)))
data$CENSUS_BLOCK_NUM[is.na(data$CENSUS_BLOCK_NUM)]<-data$CENSUS_TRACT[is.na(data$CENSUS

data$SALEYEAR <- as.numeric(format(as.Date(data$SALEDATE), "%Y"))


data$SalevYB <-
                    as.numeric(format(as.Date(data$SALEDATE),'%Y'))-
                    data$AYB
data$SalevYI <-
                    as.numeric(format(as.Date(data$SALEDATE),'%Y'))-
                    data$EYB

data$BATHTOTAL <- as.numeric(data$BATHRM+data$HF_BATHRM/2)

# Take Price's log, DON'T FORGET TO TAKE EXP AFTER
data$PRICE <- log(data$PRICE)


data$ASSESSMENT_SUBNBHD_imputed <-
          as.numeric(substr(data$ASSESSMENT_SUBNBHD,0,3))
data$ASSESSMENT_SUBNBHD_imputed <-
          ifelse(is.na(data$ASSESSMENT_SUBNBHD_imputed),28,data$ASSESSMENT_SUBNBHD_imput


#data$SALEDATE <- as.numeric(data$SALEDATE)
data$NATIONALGRID <- as.numeric(data$NATIONALGRID)
data$ASSESSMENT_NBHD <- as.numeric(data$ASSESSMENT_NBHD)
data$STYLE <- as.numeric(data$STYLE)
```

```r
data$STRUCT <- as.numeric(data$STRUCT)
data$EXTWALL <- as.numeric(data$EXTWALL)
data$INTWALL <- as.numeric(data$INTWALL)
data$ROOF <- as.numeric(data$ROOF)
data$WARD <- as.numeric(data$WARD)
data$QUADRANT <- as.numeric(data$QUADRANT)


#try
RMLSE_Score <- function(real,pred, take_log = TRUE){
  if (take_log){
    print(sqrt(1/length(real)* sum( (log(real) -log(pred))^2 ,na.rm=TRUE )))
  }else{
    print(sqrt(1/length(real)* sum( (real -pred)^2 ,na.rm=TRUE )))
  }

}


library(xgboost)
fold_num <-1
temp_data<-data
train<-temp_data[temp_data$fold!=fold_num,]
train$PRICE = log(train$PRICE)
train<-train[as.Date(train$SALEDATE)>=as.Date('1990-01-01'),]
train <- subset(train, select=c(BATHRM,HF_BATHRM,HEAT,AC,ROOMS,BEDRM,
                                AYB,YR_RMDL,EYB,STORIES,SALEDATE,PRICE,GBA,STYLE,STRUC
                                EXTWALL,ROOF,INTWALL,KITCHENS,FIREPLACES,USECODE,LANDA
                                LATITUDE,LONGITUDE,ASSESSMENT_NBHD,CENSUS_TRACT,WARD,Q
dtrain <- xgb.DMatrix(data = data.matrix(subset(train, select=-c(PRICE))), label=train$

test<-temp_data[temp_data$fold==fold_num,]
test_price <- test$PRICE
test<-subset(test, select=-c(Id,fold,ASSESSMENT_SUBNBHD,FULLADDRESS,NATIONALGRID,CENSUS_
test_price <- log(test$PRICE)
dtest <- xgb.DMatrix(data = data.matrix(subset(test, select=-c(PRICE))), label=test_pri
watchlist <- list(train=dtrain, test=dtest)

bst4 <- xgb.train(data=dtrain, max.depth=4, eta=0.02, nthread = 4, nrounds=3000, watchli
bst5 <- xgb.train(data=dtrain, max.depth=5, eta=0.02, nthread = 4, nrounds=3000, watchli
bst6 <- xgb.train(data=dtrain, max.depth=6, eta=0.02, nthread = 4, nrounds=3000, watchli
bst7 <- xgb.train(data=dtrain, max.depth=7, eta=0.02, nthread = 4, nrounds=3000, watchli
bst8 <- xgb.train(data=dtrain, max.depth=8, eta=0.02, nthread = 4, nrounds=3000, watchli
bst9 <- xgb.train(data=dtrain, max.depth=9, eta=0.02, nthread = 4, nrounds=3000, watchli
bst10 <- xgb.train(data=dtrain, max.depth=10, eta=0.02, nthread = 4, nrounds=3000, watch
```

```
library(ggplot2)
test_price <- log(test$PRICE)
test_accuracy <- sapply(list(bst4,bst5,bst6,bst7,bst8,bst9,bst10), function(x){
  predict_price <- predict(x, dtest)
  RMLSE_Score(test_price,predict_price, FALSE)
})
#[1] 0.01520559 0.01500764 0.01492193 0.01494255
# [5] 0.01448932 0.01456044 0.01454242
test_df <- data.frame(max.depth=c(4,5,6,7,8,9,10),test_accuracy=test_accuracy)
ggplot(data=test_df,
       aes(x=max.depth, y=test_accuracy)) +
  geom_path()+
  ylab('Test accuracy for Fold1')+
  geom_point()


capture.output(xgb.train(data=dtrain, max.depth=8, eta=0.02, nthread = 4, nrounds=9000,


rmse <- read.table("../Visualization/xgb_nround.txt",sep = c(':'),header = FALSE, col.na
trim <- function (x) gsub("^\\s+|\\s+$", "", x)
rmse$row.name<-1:nrow(rmse)
rmse$training.error<-as.numeric(unlist(regmatches(rmse$training.error,gregexpr("[[:digi
rmse$testing.error <- as.numeric(rmse$testing.error)


ggplot(rmse[500:nrow(rmse),], aes(row.name)) +
  geom_line(aes(y = testing.error, colour = "testing.error")) +
  xlab('nround')+
  ylab('Testing accuracy for Fold 1')
  #+
  #geom_line(aes(y = training.error, colour = "training.error"))


ggplot(rmse[500:nrow(rmse),], aes(row.name)) +
  #geom_line(aes(y = testing.error, colour = "testing.error")) +
  geom_line(aes(y = training.error, colour = "training.error"))+
  xlab('nround')+
  ylab('Training accuracy for Fold 1')
```

11.9 Code for random forest

```
data <- read.csv('../data/housing_price.csv', header = TRUE)
getmode <- function(v) {
```

```
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

# Fill missing KITCHENS value with mode
data$KITCHENS[is.na(data$KITCHENS)] <- getmode(data$KITCHENS[!is.na(data$KITCHENS)])


med_diff_built_remodel <-floor(median(data$YR_RMDL[!is.na(data$YR_RMDL)&!is.na(data$AYB)
                                   data$AYB[!is.na(data$YR_RMDL)&!is.na(data$AYB)] ))

# Fill YR_RMDL
data$YR_RMDL[is.na(data$YR_RMDL)&!is.na(data$AYB)] <-data$AYB[is.na(data$YR_RMDL)&!is.na


# Fill the situation where both YR_RMDL and AYB are missing
missing_both <- is.na(data$YR_RMDL)&is.na(data$AYB)
data$AYB[missing_both]<-floor(median(data$AYB[!is.na(data$AYB)]))
data$YR_RMDL[missing_both]<-data$AYB[missing_both]- med_diff_built_remodel


missing_built_have_remodel <- (!is.na(data$YR_RMDL)&is.na(data$AYB))
data$AYB[missing_built_have_remodel] <- data$YR_RMDL[missing_built_have_remodel]-med_dif

# Fill the missing STORIES
data$STORIES[is.na(data$STORIES)]<-floor(median(data$STORIES[!is.na(data$STORIES)]))

# There is a 20 value is YR_RMDL, apparently it is a wrong data
# Fill it
yr_rmdl_20_index <- data$YR_RMDL==20&!is.na(data$YR_RMDL)
data[yr_rmdl_20_index,]
data$YR_RMDL[yr_rmdl_20_index]=data$AYB[yr_rmdl_20_index]+med_diff_built_remodel

# Wrong Data in AC
data$AC[data$AC ==0] <- getmode(data$AC[data$AC !=0])
# Wrong Data in STORIES

data$STORIES[data$STORIES>=14] <-14 #floor(median(data$STORIES[data$STORIES<14]))

trim <- function (x) gsub("^\\s+|\\s+$", "", x)
# AC
data$AC <- factor(data$AC,level=c('Y','N'), label=c(1,0))
# GRADE
data$GRADE <- as.numeric(factor(data$GRADE,level=c(
'Low Quality', 'Fair Quality', 'Average', 'Above Average', 'Good Quality', 'Very Good',
data$CNDTN <- as.numeric(factor(data$CNDTN,level=c('Poor', 'Fair', 'Average', 'Good', '
```

```
# HEAT
data$HEAT <- as.character(data$HEAT)
data$HEAT[data$HEAT=='Air Exchng'|
                data$HEAT=='Elec Base Brd'|
                 data$HEAT=='Evp Cool'|
                 data$HEAT=='Hot Water Rad'|
                 data$HEAT=='Ind Unit'|
                 data$HEAT=='Wall Furnace'|
                 data$HEAT=='Water Base Brd'|
                 data$HEAT=='Ht Pump'|
                 data$HEAT=='Gravity Furnac'
                 ] <- 1
data$HEAT[data$HEAT=='Air-Oil'|
                data$HEAT=='Electric Rad'|
                 data$HEAT=='Forced Air'|
                data$HEAT=='Warm Cool'
                 ] <- 2
data$HEAT[data$HEAT=='No Data']<-0
data$HEAT<-as.numeric(data$HEAT)

# CENSUS_TRACT, CENSUS_BLOCK => CENSUS_BLOCK_NUM
data$CENSUS_BLOCK_NUM <- as.numeric(trim(gsub('\\s+',
                                              '',
                                            data$CENSUS_BLOCK)))
data$CENSUS_BLOCK_NUM[is.na(data$CENSUS_BLOCK_NUM)]<-data$CENSUS_TRACT[is.na(data$CENSUS

data$SALEYEAR <- as.numeric(format(as.Date(data$SALEDATE), "%Y"))


data$SalevYB <-
                    as.numeric(format(as.Date(data$SALEDATE),'%Y'))-
                    data$AYB
data$SalevYI <-
                    as.numeric(format(as.Date(data$SALEDATE),'%Y'))-
                    data$EYB

data$BATHTOTAL <- as.numeric(data$BATHRM+data$HF_BATHRM/2)

# Take Price's log, DON'T FORGET TO TAKE EXP AFTER
data$PRICE <- log(data$PRICE)


data$ASSESSMENT_SUBNBHD_imputed <-
          as.numeric(substr(data$ASSESSMENT_SUBNBHD,0,3))
data$ASSESSMENT_SUBNBHD_imputed <-
```

```
          ifelse(is.na(data$ASSESSMENT_SUBNBHD_imputed),28,data$ASSESSMENT_SUBNBHD_imput


#data$SALEDATE <- as.numeric(data$SALEDATE)
data$NATIONALGRID <- as.numeric(data$NATIONALGRID)
data$ASSESSMENT_NBHD <- as.numeric(data$ASSESSMENT_NBHD)
data$STYLE <- as.numeric(data$STYLE)
data$STRUCT <- as.numeric(data$STRUCT)
data$EXTWALL <- as.numeric(data$EXTWALL)
data$INTWALL <- as.numeric(data$INTWALL)
data$ROOF <- as.numeric(data$ROOF)
data$WARD <- as.numeric(data$WARD)
data$QUADRANT <- as.numeric(data$QUADRANT)


#try
RMLSE_Score <- function(real,pred, take_log = TRUE){
  if (take_log){
    print(sqrt(1/length(real)* sum( (log(real) -log(pred))^2 ,na.rm=TRUE )))
  }else{
    print(sqrt(1/length(real)* sum( (real -pred)^2 ,na.rm=TRUE )))
  }

}

library(ranger)
fold_num <- 1
  temp_data<- data
  train <- temp_data[temp_data$fold != fold_num,]

  train<-train[as.Date(train$SALEDATE)>=as.Date('1990-01-01'),]
  train <- subset(train, select = -c(Id, fold))


  test <- temp_data[temp_data$fold == fold_num,]
  test_price <- test$PRICE
  test <- subset(test, select = -c(PRICE, Id, fold))
  form <- as.formula('PRICE~YR_RMDL+ SALEDATE+GBA+STRUCT+GRADE+CNDTN+FIREPLACES+USECODE+
                  LANDAREA+ZIPCODE+NATIONALGRID+LATITUDE+LONGITUDE+ASSESSMENT_SUBNBHD+
                  CENSUS_TRACT+WARD+QUADRANT+CENSUS_BLOCK_NUM+SALEYEAR+SalevYB+SalevYI+
                  BATHTOTAL+ASSESSMENT_SUBNBHD_imputed')

ranger_models <- lapply(seq(4,14,2), function(x){
  return (
    ranger(form, data = train,importance = "permutation",mtry = x,num.trees = 750,min.no
  )
```

```
})


#saveRDS(ranger_models,'ranger_models.RDS')
test_accuracy<- sapply(ranger_models, function(x){
  rfpredict <- predict(x, test)$predictions
  RMLSE_Score(test_price,rfpredict, FALSE)

})
#[1] 0.1752901 0.1739828 0.1737324 0.1746719
#[5] 0.1749235 0.1756615


library(ggplot2)

test_df <- data.frame(mtry=seq(4,14,2),test_accuracy=test_accuracy)
ggplot(data=test_df,
       aes(x=mtry, y=test_accuracy)) +
  geom_path()+
  ylab('Test accuracy for Fold1')+
  geom_point()

ranger_models_max_depth <- lapply(seq(10,30,5), function(x){
  return (
    ranger(form, data = train,importance = "permutation",mtry = 8,num.trees = 750,min.no
  )
})

test_accuracy_max_depth<- sapply(ranger_models_max_depth, function(x){
  rfpredict <- predict(x, test)$predictions
  RMLSE_Score(test_price,rfpredict, FALSE)

})
#[1] 0.1962009
#[1] 0.1765021
#[1] 0.1739121
#[1] 0.1737324
#[1] 0.1737321

test_df <- data.frame(max.depth =seq(10,30,5),test_accuracy=test_accuracy_max_depth)
ggplot(data=test_df,
       aes(x=max.depth , y=test_accuracy)) +
  geom_path()+
  ylab('Test accuracy for Fold1')+
  geom_point()
```