Home Prices EDA and Machine Learning

Executive Summary

As a team, we set out to develop a model to predict apartments prices in South Korea based on various features about the location and attributes of the particular apartment. Housing prices is an area of big concern all over the world given our ever growing population. South Korea, a country in which about 59.9% of all homes are apartments, is no exception. We will specifically be examining apartments in order to understand which factors contribute most to apartments prices, and to predict prices for apartments that we have not yet seen. For a family searching for a home in South Korea, this model can help them to understand if they are overpaying. For example, if they know the location of the apartment and some basic information, such as the number of rooms, they can determine if they are being offered a fair price. In addition, the landlord or seller of the property can use this model to calculate a fair price for the apartment.

Our Dataset contains the following features:

- transaction_real_price: the price that the apartment was sold at (target variable)
- key: increments by one based on the row number
- apartment_id: a unique identifier for the building in which the apartment is found
- transaction_year_month: the year and month the transaction took place
- transaction_date: the date on which the transaction took place
- year_of_completion: the year the apartment was built
- exclusive_use_area: the total floor area of the building
- floor: the floor number that the apartment building is located
- latitude: latitude of the apartment
- longitude: longitude of the apartment
- address_by_law: address represented numerically
- total_parking_capacity_in_site: the number of parking spots for the entire building complex in which the apartment is located (potentially there could be multiple separate buildings in the site)
- total_household_count_in_site: The number of separate households located in the apartment complex
- apartment_building_count_in_sites: the number of separate apartments building in the apartment complex
- tallest_building_in_sites: the number of floors of the tallest building in the apartment complex
- lowest_building_in_sites: the number of floors of the shortest building in the apartment complex
- heat_type: the type of heat available tot he apartment (individual, central, district)
- heat_fuel: the type of heating fuel used by the apartment (gas, cogeneration)
- room_id: unique identifier for the apartment
- supply_area: Total site area (area of the entire apartment complex)
- total_household_count_of_area_type: Count of households in the immediate area
- room_count: the number of rooms in the apartment
- bathroom_count: the number of bathrooms in the apartment
- front_door_structure: the structure of the entrance to the apartment (corridor, stairway, mixed)

EDA and Data Cleaning

Load in dataset

```
setwd("C:/Users/pmank/Dropbox/BU/BA810/group_project")
Price <- fread("trainPrice.csv")
```

First few rows

We can see that we have a total of 25 features in the dataset

```
head(Price, 5)
```

```
##    key apartment_id city transaction_year_month transaction_date
## 1:   0         5584    1                 200601            11~20
## 2:   1         5584    1                 200601            11~20
## 3:   2         5059    1                 200601            11~20
## 4:   3         2816    1                 200601            11~20
## 5:   4         2816    1                 200601            11~20
##    year_of_completion exclusive_use_area floor latitude longitude
## 1:               1999              47.43     6 37.58597  127.0002
## 2:               1999              44.37     8 37.58597  127.0002
## 3:               1992              54.70     8 37.58051  127.0140
## 4:               1993              64.66    11 37.58032  127.0118
## 5:               1993             106.62     7 37.58032  127.0118
##    address_by_law total_parking_capacity_in_site total_household_count_in_sites
## 1:     1111017100                            163                            136
## 2:     1111017100                            163                            136
## 3:     1111017400                            902                            585
## 4:     1111017400                            902                            919
## 5:     1111017400                            902                            919
##    apartment_building_count_in_sites tallest_building_in_sites
## 1:                                 1                         8
## 2:                                 1                         8
## 3:                                 5                        14
## 4:                                 7                        15
## 5:                                 7                        15
##    lowest_building_in_sites  heat_type heat_fuel room_id supply_area
## 1:                        4 individual       gas   91120       65.63
## 2:                        4 individual       gas   91119       61.39
## 3:                        9 individual       gas    8430       72.36
## 4:                       11 individual       gas    5839       87.30
## 5:                       11 individual       gas    5836      127.74
##    total_household_count_of_area_type room_count bathroom_count
## 1:                                 46          1              1
## 2:                                 10          2              1
## 3:                                201          2              1
## 4:                                284          2              1
## 5:                                112          4              2
##    front_door_structure transaction_real_price
## 1:             corridor              215000000
## 2:             corridor              200000000
## 3:             corridor              168000000
## 4:             corridor              165000000
## 5:             stairway              280000000
```

Structure of the dataset

There are a total of 1,601,458 observations

```
str(Price)
```

```
## Classes 'data.table' and 'data.frame':   1601458 obs. of  25 variables:
##  $ key                               : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ apartment_id                      : int  5584 5584 5059 2816 2816 2815 2815 9867 2818 2817
...
##  $ city                              : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ transaction_year_month            : int  200601 200601 200601 200601 200601 200601 200601
200601 200601 200601 ...
##  $ transaction_date                  : chr  "11~20" "11~20" "11~20" "11~20" ...
##  $ year_of_completion                : int  1999 1999 1992 1993 1993 2000 2000 2005 1999 2002
...
##  $ exclusive_use_area                : num  47.4 44.4 54.7 64.7 106.6 ...
##  $ floor                             : int  6 8 8 11 7 9 13 10 18 12 ...
##  $ latitude                          : num  37.6 37.6 37.6 37.6 37.6 ...
##  $ longitude                         : num  127 127 127 127 127 ...
##  $ address_by_law                    :integer64 1111017100 1111017100 1111017400 1111017400 1
111017400 1111018700 1111018700 1114016200 ...
##  $ total_parking_capacity_in_site    : num  163 163 902 902 902 ...
##  $ total_household_count_in_sites    : int  136 136 585 919 919 964 964 461 2282 5150 ...
##  $ apartment_building_count_in_sites : int  1 1 5 7 7 12 12 9 19 42 ...
##  $ tallest_building_in_sites         : num  8 8 14 15 15 23 23 23 20 18 ...
##  $ lowest_building_in_sites          : num  4 4 9 11 11 10 10 6 8 11 ...
##  $ heat_type                         : chr  "individual" "individual" "individual" "individua
l" ...
##  $ heat_fuel                         : chr  "gas" "gas" "gas" "gas" ...
##  $ room_id                           : int  91120 91119 8430 5839 5836 5831 5833 11862 5843 5
842 ...
##  $ supply_area                       : num  65.6 61.4 72.4 87.3 127.7 ...
##  $ total_household_count_of_area_type: int  46 10 201 284 112 454 207 82 576 864 ...
##  $ room_count                        : num  1 2 2 2 4 3 3 3 3 3 ...
##  $ bathroom_count                    : num  1 1 1 1 2 2 1 2 2 1 ...
##  $ front_door_structure              : chr  "corridor" "corridor" "corridor" "corridor" ...
##  $ transaction_real_price            :integer64 215000000 200000000 168000000 165000000 28000
0000 415000000 267000000 415000000 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
transform(Price, transaction_real_price = as.numeric(transaction_real_price))
```

```
##              key apartment_id city transaction_year_month transaction_date
##       1:       0         5584    1                 200601            11~20
##       2:       1         5584    1                 200601            11~20
##       3:       2         5059    1                 200601            11~20
##       4:       3         2816    1                 200601            11~20
##       5:       4         2816    1                 200601            11~20
##      ---
## 1601454: 1605344        11500    0                 201810            21~31
## 1601455: 1605346        16686    1                 201810            21~31
## 1601456: 1605356        22243    0                 201810            21~31
## 1601457: 1605366         3686    1                 201810            21~31
## 1601458: 1605373         2937    1                 201810            21~31
##          year_of_completion exclusive_use_area floor latitude longitude
##       1:               1999            47.4300     6 37.58597  127.0002
##       2:               1999            44.3700     8 37.58597  127.0002
##       3:               1992            54.7000     8 37.58051  127.0140
##       4:               1993            64.6600    11 37.58032  127.0118
##       5:               1993           106.6200     7 37.58032  127.0118
##      ---
## 1601454:               1999           118.4700    14 35.15557  129.0175
## 1601455:               2007            59.9900     4 37.50239  126.9420
## 1601456:               2014            84.9669    31 35.06480  128.9831
## 1601457:               1996            59.3400     4 37.55521  127.1313
## 1601458:               1999            84.8800     5 37.60433  127.0172
##          address_by_law total_parking_capacity_in_site
##       1:     1111017100                            163
##       2:     1111017100                            163
##       3:     1111017400                            902
##       4:     1111017400                            902
##       5:     1111017400                            902
##      ---
## 1601454:     2623011100                            876
## 1601455:     1159010200                           1651
## 1601456:     2638010600                           1761
## 1601457:     1174010700                            111
## 1601458:     1129013300                            802
##          total_household_count_in_sites apartment_building_count_in_sites
##       1:                            136                                 1
##       2:                            136                                 1
##       3:                            585                                 5
##       4:                            919                                 7
##       5:                            919                                 7
##      ---
## 1601454:                            819                                 8
## 1601455:                           1122                                22
## 1601456:                           1326                                 9
## 1601457:                            107                                 1
## 1601458:                            860                                 8
##          tallest_building_in_sites lowest_building_in_sites  heat_type
##       1:                         8                        4 individual
##       2:                         8                        4 individual
##       3:                        14                        9 individual
##       4:                        15                       11 individual
```

```
##       5:                       15                           11 individual
##      ---
## 1601454:                       27                           13 individual
## 1601455:                       15                            8 individual
## 1601456:                       35                           27 individual
## 1601457:                       19                           11 individual
## 1601458:                       22                            7 individual
##          heat_fuel room_id supply_area total_household_count_of_area_type
##       1:       gas   91120       65.63                                 46
##       2:       gas   91119       61.39                                 10
##       3:       gas    8430       72.36                                201
##       4:       gas    5839       87.30                                284
##       5:       gas    5836      127.74                                112
##      ---
## 1601454:       gas   44386      143.45                                108
## 1601455:       gas   13884       79.98                                254
## 1601456:       gas   56043      109.77                                209
## 1601457:       gas  165820       88.37                                  4
## 1601458:       gas    6279      108.75                                209
##          room_count bathroom_count front_door_structure transaction_real_price
##       1:          1              1             corridor               2.15e+08
##       2:          2              1             corridor               2.00e+08
##       3:          2              1             corridor               1.68e+08
##       4:          2              1             corridor               1.65e+08
##       5:          4              2             stairway               2.80e+08
##      ---
## 1601454:          4              2             stairway               4.27e+08
## 1601455:          3              2             stairway               7.71e+08
## 1601456:          3              2             stairway               3.43e+08
## 1601457:          3              1             corridor               4.85e+08
## 1601458:          3              2             stairway               4.30e+08
```

```
#summary(Price)
```

Data Cleaning

View the count of null values in each column

The feature, total_parking_capacity_in_site has the largest number of null values (91813)

```
Price[, lapply(.SD, function(x) sum(is.na(x)))]
```

```
##    key apartment_id city transaction_year_month transaction_date
## 1:   0            0    0                      0                0
##    year_of_completion exclusive_use_area floor latitude longitude
## 1:                  0                  0     0        0         0
##    address_by_law total_parking_capacity_in_site total_household_count_in_sites
## 1:              0                          91813                              0
##    apartment_building_count_in_sites tallest_building_in_sites
## 1:                                 0                         9
##    lowest_building_in_sites heat_type heat_fuel room_id supply_area
## 1:                        9         0         0       0           0
##    total_household_count_of_area_type room_count bathroom_count
## 1:                                  0        691            691
##    front_door_structure transaction_real_price
## 1:                    0                      0
```

View all unique values present in columns

```
unique(Price$heat_type)
```

```
## [1] "individual" "central"    "district"   ""
```

How many values are empty strings or dashes?

First, we will view the unique values to understand if there are any other invalid values other than NA

```
unique(Price$city)
```

```
## [1] 1 0
```

```
unique(Price$bathroom_count)
```

```
## [1]  1  2  0 NA  3  4  5
```

```
unique(Price$room_count)
```

```
##  [1]  1  2  4  3  5  0  6 NA  8  7
```

```
unique(Price$front_door_structure)
```

```
## [1] "corridor" "stairway" "mixed"    ""         "-"
```

```
unique(Price$year_of_completion)
```

```
##  [1] 1999 1992 1993 2000 2005 2002 2001 1997 1996 1990 1989 1987 1985 1995 1988
## [16] 1991 1977 1971 1998 1974 1994 2003 1984 1986 1982 1983 2004 1975 1981 1980
## [31] 1978 1976 1979 2006 1973 1962 1970 1968 1969 1972 2007 2008 2009 1966 2010
## [46] 2011 2012 2013 2014 2015 2016 2017 2018
```

A number of columns have an empty string value or one dash rather than an NA value, we will remove these from the dataset.

```
Price[heat_type == ''][, .N]
```

```
## [1] 2017
```

```
Price[heat_fuel == ''][, .N]
```

```
## [1] 9667
```

```
Price[front_door_structure == ''][, .N]
```

```
## [1] 13892
```

```
Price[heat_fuel == '-'][, .N]
```

```
## [1] 8971
```

```
Price[front_door_structure == '-'][, .N]
```

```
## [1] 21
```

We can remove these empty string and dash values

```
Price <- Price[heat_fuel != '']
Price <- Price[heat_fuel != '-']
Price <- Price[front_door_structure != '-']
Price <- Price[heat_type != '']
Price <- Price[front_door_structure != '']
```

Drop rows room_count as 8

After examining the dataset, we determined that this was an outlier since only a very small number of apartments were shown as having 8 rooms

```
Price <- Price[room_count != 8]
```

Remove all rows that have missing values

There are a total of 1601458 million observations, and we will be removing less than 100,000 of them. We can confirm that rows were removed using .N (the number of observations)

```
Price <- na.omit(Price)
Price[, .N]
```

```
## [1] 1478931
```

Create dummy variables for use in later ML steps

We will convert transaction_date, heat_type, heat_fuel, and front_door_structure

```
Price <- Price[transaction_date == "1~10", transaction_date:=1]
Price <- Price[transaction_date == "11~20", transaction_date:=2]
Price <- Price[transaction_date == "21~30", transaction_date:=3]
Price <- Price[transaction_date == "21~28", transaction_date:=3]
Price <- Price[transaction_date == "21~29", transaction_date:=3]
Price <- Price[transaction_date == "21~31", transaction_date:=3]
Price[,transaction_date := as.numeric(transaction_date)]

Price <- Price[heat_fuel == "gas", heat_fuel:=0]
Price <- Price[heat_fuel == "cogeneration", heat_fuel:=1]
Price[,heat_fuel := as.numeric(heat_fuel)]

Price[,transaction_real_price := as.numeric(transaction_real_price)]
Price[,address_by_law := as.numeric(address_by_law)]

Price <- fastDummies::dummy_cols(Price)
```

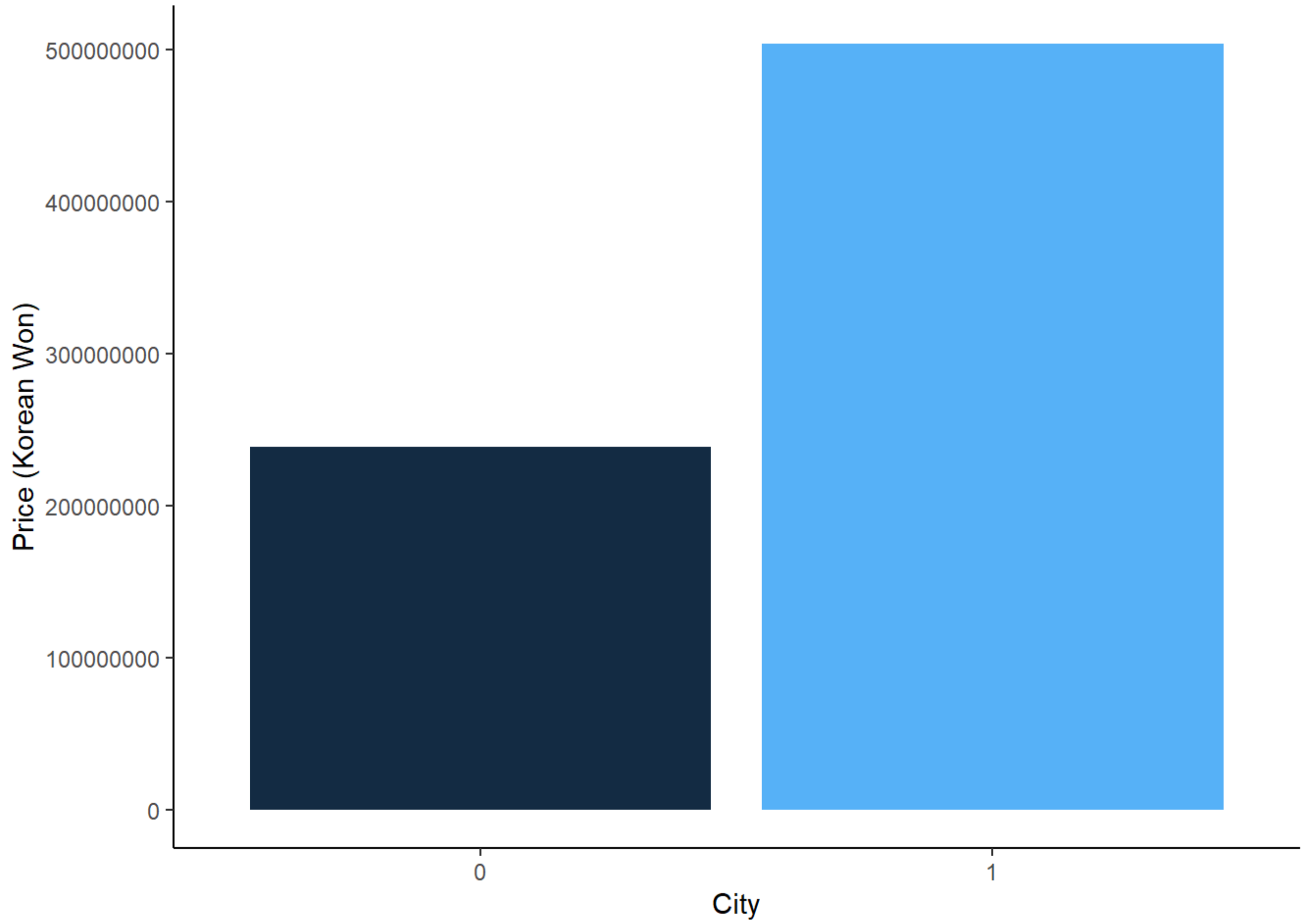
EDA

In the EDA section, we wanted to create a few graphs to help us understand the distribution of our features, and also how our features relate to the target variable, transaction_real_price (this is the price that each apartments sells at). The charts below reveal that there are a number of features that seem to relate to the price, including the city, number of rooms, front door structure, heating fuel type, and heat type. These charts were also helpful in understanding unusual values. We found that there were a few apartments that had 8 rooms, but a very low price. We decided to remove these above in the data cleaning section.

Barchart showing real price and city

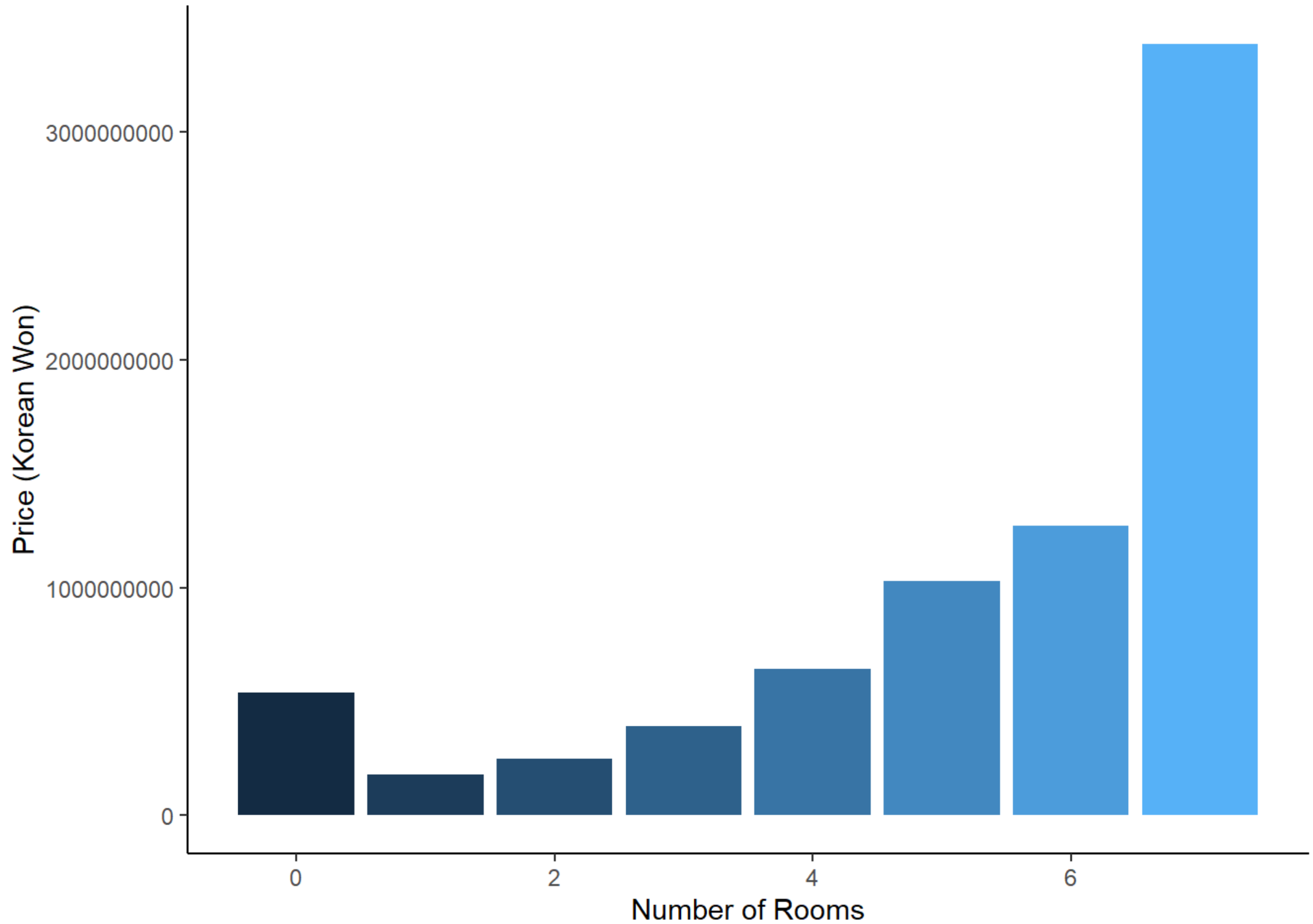
Below, 1 is Busan and 0 is Seoul. We can see that on average, prices tend to be higher in Busan.

```
ggplot(Price, aes(x=as.factor(city), y = transaction_real_price, fill = city)) +

  geom_bar(stat = "summary", fun = "mean") +
  scale_color_manual(values=c("red", "blue")) +
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE)) +

  theme(legend.position="none") +
  xlab('City') +
  ylab('Price (Korean Won)')
```



Barchart showing real price and room count

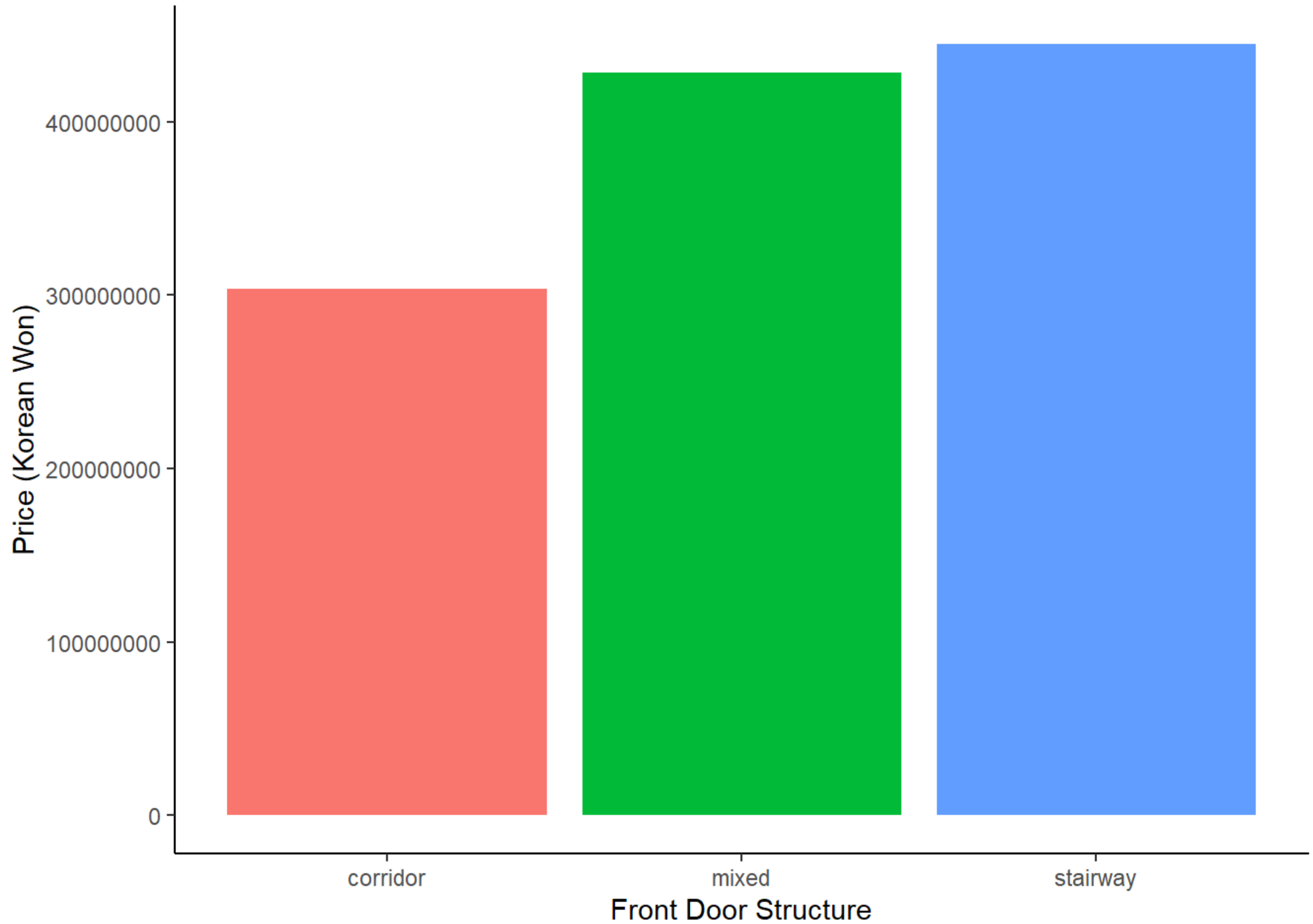
```
ggplot(Price, aes(x=room_count, y = transaction_real_price, fill = room_count)) +
  geom_bar(stat = "summary", fun = "mean")+
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE)) +
  theme(legend.position="none")+
  xlab('Number of Rooms') +
  ylab('Price (Korean Won)')
```



####

Barchart showing real price and front_door_structure

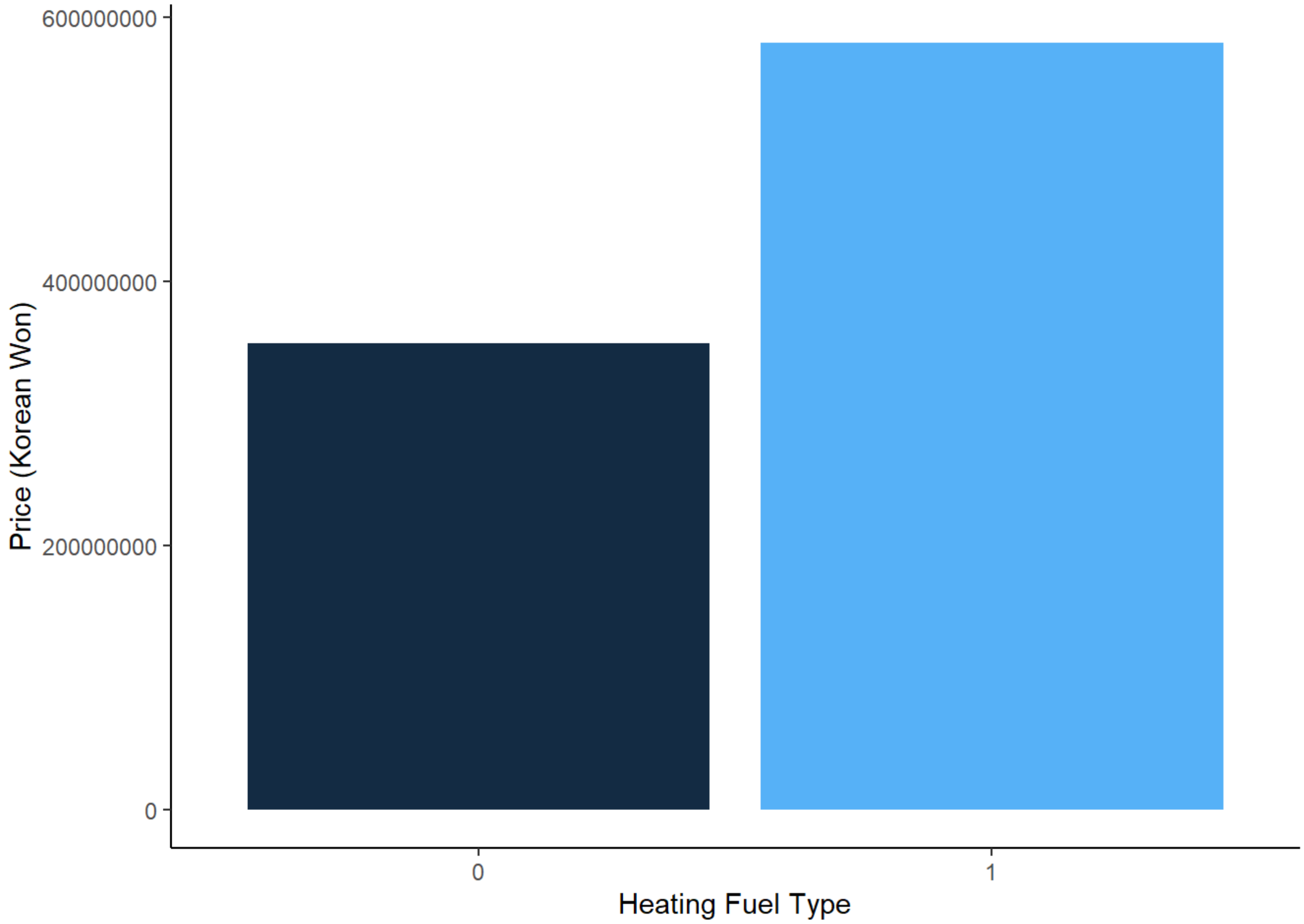
```
ggplot(Price, aes(x=front_door_structure, y = transaction_real_price, fill = front_door_structur
e)) +
  geom_bar(stat = "summary", fun = "mean")+
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE)) +
  theme(legend.position="none")+
  xlab('Front Door Structure') +
  ylab('Price (Korean Won)')
```



Barchart showing real price and heat_fuel

Below, 0 represents gas and 1 is co-generation

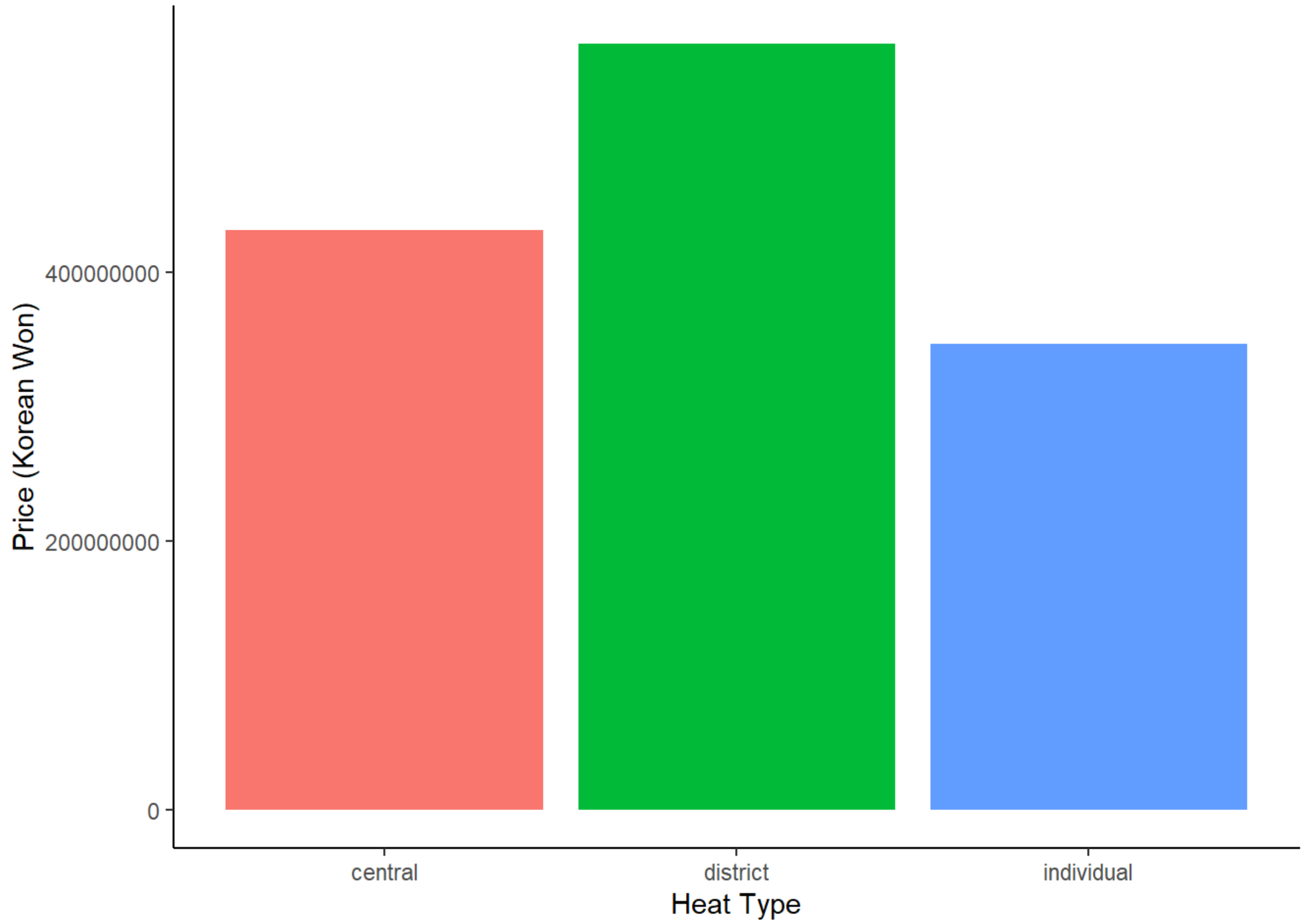
```
ggplot(Price, aes(x=as.factor(heat_fuel), y = transaction_real_price, fill = heat_fuel)) +
  geom_bar(stat = "summary", fun = "mean")+
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE)) +
  theme(legend.position="none")+
  xlab('Heating Fuel Type') +
  ylab('Price (Korean Won)')
```



####

Barchart showing real price and heat_type Below, 0 is individual, 1 is central and 2 is district

```
ggplot(Price, aes(x=heat_type, y = transaction_real_price, fill = heat_type)) +
  geom_bar(stat = "summary", fun = "mean")+
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE)) +
  theme(legend.position="none")+
  xlab('Heat Type') +
  ylab('Price (Korean Won)')
```

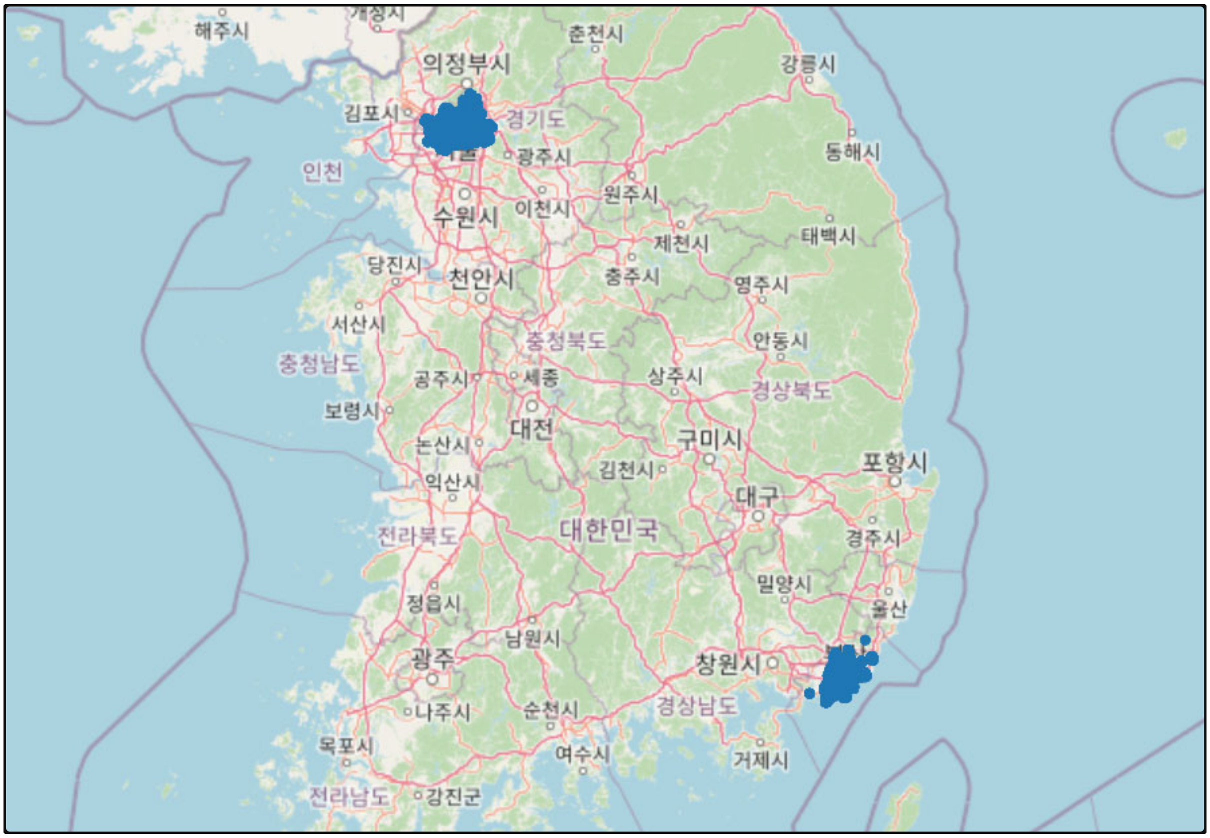


Plotly Map showing locations of apartments

We can see that all of the apartments are either in Seoul or Busan. This is coded in the data in the city feature.

```
fig <- head(Price, 100000)
fig <- fig %>%
   plot_ly(
     lat = ~latitude,
     lon = ~longitude,
     #marker = list(color = "fuchsia"),
     color = Price[,"transaction_real_price"],
     type = 'scattermapbox',
     hovertext = Price[,"transaction_real_price"])
 fig <- fig %>%
   layout(
     mapbox = list(
       style = 'open-street-map',
       zoom =2.5,
       center = list(longitude = 35.963911, latitude = 127.919770)))

fig
```

```
## No scattermapbox mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plotly.com/r/reference/#scatter-mode
```



Machine Learning

In this section, we used a number of machine learning techniques to predict the prices of apartments in South Korea. We used the following models:

- Linear Regression
- Forward Regression
- Backward Regression
- Ridge Regression
- Lasso Regression
- Decision Tree
- Bagging
- Random Forest
- Boosting

Drop useless columns

Below, we will remove apartment_id, room_id, and key since these are not useful in prediction. We also removed redundant dummy variables.

```
Price[,key := NULL]
Price[,apartment_id := NULL]
Price[,room_id := NULL]
Price[,heat_type := NULL]
Price[,front_door_structure := NULL]
Price[,heat_type_individual := NULL]
Price[,front_door_structure_stairway := NULL]
# prevent aliased coefficients
```

Setup test and train datasets

```
# Set the seed to get consistent results
set.seed(810)

rows <- sample(nrow(Price), 160000)
Price <- Price[rows,]

# Split the data
# This way we can do an 80/20 split
row_index <- sample(nrow(Price), 128000)

# we use that set of random numbers to select those random rows
dd_train <- Price[row_index,]
dd_test <- Price[-row_index,]
```

Linear Regression

We started by trying linear regression model. We used an 80/20 split between our test and train datasets, and used all of the features to predict price. For this model, we calculated a Train RMSE score of 188,332,311 Korean Won. We calculated a Test RMSE score of 188,544,540 Korean Won.

```
# our response variables to use later
set.seed(810)
y_train <- dd_train$transaction_real_price
y_test <- dd_test$transaction_real_price

# fit the full model
fit_lm1 <- lm(transaction_real_price ~ ., data=dd_train)
yhat_train_lm1 <- predict(fit_lm1)
mse_train_lm1 <- mean((y_train - yhat_train_lm1)^2)
paste("Linear Regression Train RMSE",sqrt(mse_train_lm1))
```

```
## [1] "Linear Regression Train RMSE 188332311.411548"
```

```
yhat_test_lm1 <- predict(fit_lm1, dd_test)
mse_test_lm1 <- mean((y_test - yhat_test_lm1)**2)
paste("Linear Regression Test RMSE",sqrt(mse_test_lm1))
```

```
## [1] "Linear Regression Test RMSE 188544540.361935"
```

A summary of the coefficient values in the model.

```
summary(fit_lm1)
```

```
##
## Call:
## lm(formula = transaction_real_price ~ ., data = dd_train)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.118e+09 -9.743e+07 -1.692e+07  7.074e+07  6.097e+09
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -5.436e+10  9.421e+08 -57.707  < 2e-16 ***
## city                              5.765e+09  5.642e+07 102.180  < 2e-16 ***
## transaction_year_month            1.668e+05  1.428e+03 116.823  < 2e-16 ***
## transaction_date                  6.465e+05  6.468e+05   1.000 0.317529
## year_of_completion               -2.637e+06  9.746e+04 -27.053  < 2e-16 ***
## exclusive_use_area                1.553e+06  1.346e+05  11.538  < 2e-16 ***
## floor                             1.869e+06  8.684e+04  21.525  < 2e-16 ***
## latitude                         -1.026e+09  1.106e+07 -92.799  < 2e-16 ***
## longitude                         4.498e+08  7.047e+06  63.830  < 2e-16 ***
## address_by_law                    1.417e+00  3.967e-02  35.713  < 2e-16 ***
## total_parking_capacity_in_site    2.850e+04  1.047e+03  27.226  < 2e-16 ***
## total_household_count_in_sites   -8.174e+04  1.679e+03 -48.684  < 2e-16 ***
## apartment_building_count_in_sites  6.668e+06  8.495e+04  78.494  < 2e-16 ***
## tallest_building_in_sites         2.406e+06  1.213e+05  19.832  < 2e-16 ***
## lowest_building_in_sites          3.271e+06  1.199e+05  27.291  < 2e-16 ***
## heat_fuel                         5.167e+07  3.737e+06  13.828  < 2e-16 ***
## supply_area                       3.326e+06  1.148e+05  28.977  < 2e-16 ***
## total_household_count_of_area_type -1.459e+04  1.929e+03  -7.565 3.90e-14 ***
## room_count                       -6.117e+06  1.280e+06  -4.778 1.77e-06 ***
## bathroom_count                   -6.561e+06  1.725e+06  -3.803 0.000143 ***
## heat_type_central                 7.847e+06  2.117e+06   3.707 0.000210 ***
## heat_type_district                6.760e+07  3.779e+06  17.890  < 2e-16 ***
## front_door_structure_corridor    -1.441e+07  1.760e+06  -8.184 2.77e-16 ***
## front_door_structure_mixed       -2.789e+06  4.178e+06  -0.668 0.504395
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 188300000 on 127976 degrees of freedom
## Multiple R-squared:  0.6566, Adjusted R-squared:  0.6565
## F-statistic: 1.064e+04 on 23 and 127976 DF,  p-value: < 2.2e-16
```

Forward Selection

In forward selection, we used cross validation to split our data into 10 groups. In this model, we found that as the number of predictors in the model increased, our MSE decreased. This model would suggest that all of the features would be useful; however, we know that forward and backward selection don't test every possible combination.

In the output below, we can see that there is a separate results for reach model, starting with just one explanatory variable, and ending with a model with all 24 variables.

In this model, our lowest RMSE was 188,321,979 Korean Won with 21 variables.

```
# Set up repeated k-fold cross-validation
set.seed(810)
train.control <- trainControl(method = "cv", number = 10)

# Train the model

step.model_f <- train(transaction_real_price ~., data = dd_train,
      method = "leapForward",
       tuneGrid = data.frame(nvmax = 1:23),
       trControl = train.control
       )
step.model_f$results
```

```
##    nvmax      RMSE  Rsquared       MAE  RMSESD  RsquaredSD     MAESD
## 1      1 275820794 0.2633316 183550247 5271468 0.012969232 1810020.1
## 2      2 234909141 0.4656794 153777481 5215660 0.009970077 1767563.1
## 3      3 223937947 0.5144075 145692662 4701072 0.007960067 1515730.5
## 4      4 215028464 0.5523260 137104337 4926310 0.007136277 1547467.7
## 5      5 206779570 0.5860626 129796361 4932603 0.006514049 1288489.4
## 6      6 200154326 0.6121798 126424032 5129113 0.007657021 1145261.7
## 7      7 194883374 0.6323292 123613028 5326193 0.008018565 1179427.1
## 8      8 193296484 0.6382953 122648186 5494819 0.008633204 1078313.8
## 9      9 191802101 0.6438754 121323064 5447384 0.008353271  961052.4
## 10    10 190734044 0.6478400 120350997 5522235 0.008699768 1035524.4
## 11    11 190176874 0.6498956 119592895 5569027 0.008969923 1041122.9
## 12    12 189586789 0.6520624 119844904 5670572 0.009250883 1152643.2
## 13    13 189043550 0.6540527 119405358 5720948 0.009331370 1224627.9
## 14    14 188833635 0.6548209 119216157 5741006 0.009398136 1226570.1
## 15    15 188635164 0.6555458 119113982 5813140 0.009615091 1226816.5
## 16    16 188427962 0.6563037 118836075 5841570 0.009779370 1225391.8
## 17    17 188381716 0.6564717 118850989 5837784 0.009779374 1219600.3
## 18    18 188354341 0.6565718 118734190 5845501 0.009779065 1224907.8
## 19    19 188347708 0.6565947 118769638 5841846 0.009778721 1221526.2
## 20    20 188340273 0.6566225 118745450 5832668 0.009756786 1234081.0
## 21    21 188321979 0.6566908 118720513 5817163 0.009693742 1232452.6
## 22    22 188325117 0.6566793 118720657 5815158 0.009685612 1233727.1
## 23    23 188324643 0.6566810 118717131 5814604 0.009682559 1234709.1
```

Two variable we exclude is transaction_date and front_door_structure_mixed

```
#Choose the best tuned model
summary(step.model_f$finalModel)
```

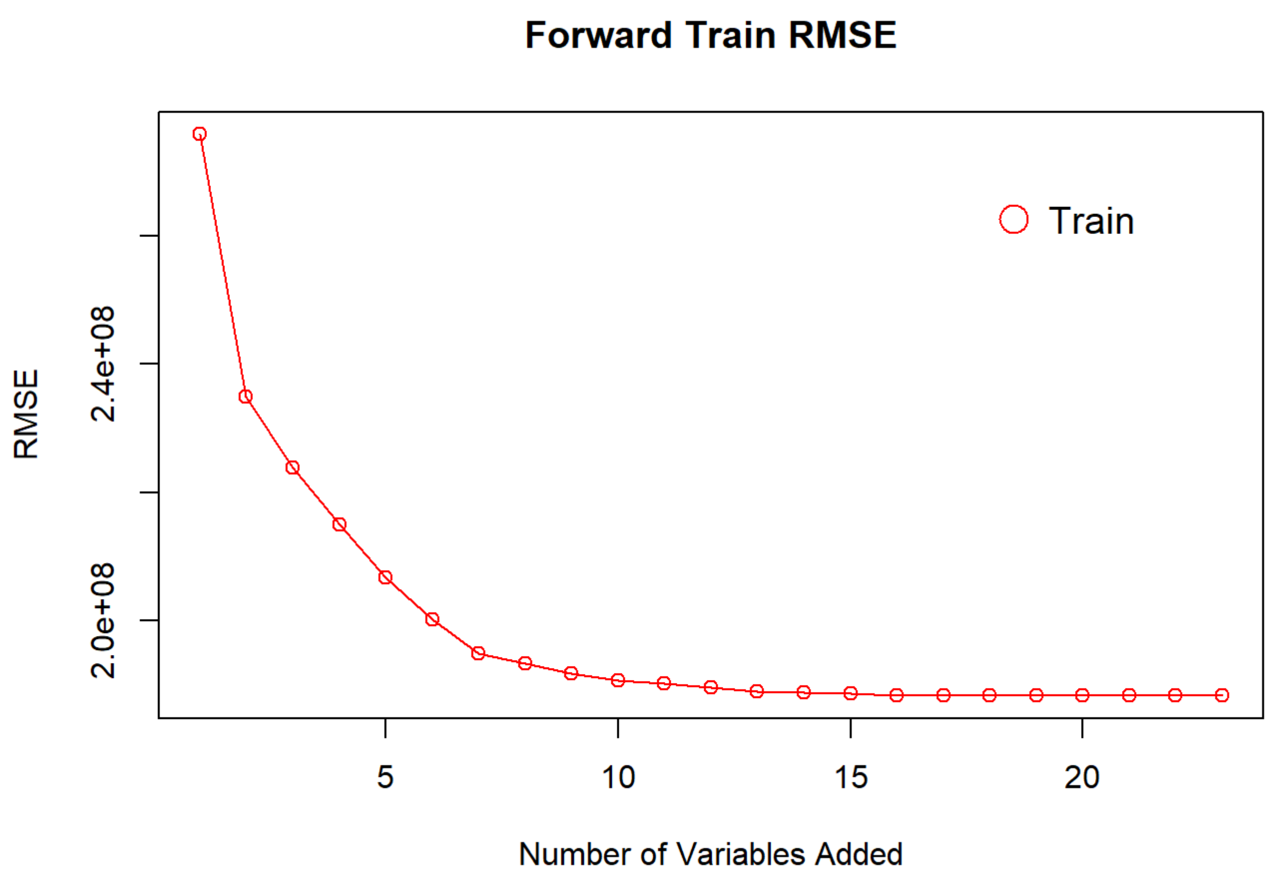
```
## Subset selection object
## 23 Variables  (and intercept)
##                                    Forced in Forced out
## city                                   FALSE      FALSE
## transaction_year_month                 FALSE      FALSE
## transaction_date                       FALSE      FALSE
## year_of_completion                     FALSE      FALSE
## exclusive_use_area                     FALSE      FALSE
## floor                                  FALSE      FALSE
## latitude                               FALSE      FALSE
## longitude                              FALSE      FALSE
## address_by_law                         FALSE      FALSE
## total_parking_capacity_in_site         FALSE      FALSE
## total_household_count_in_sites         FALSE      FALSE
## apartment_building_count_in_sites      FALSE      FALSE
## tallest_building_in_sites              FALSE      FALSE
## lowest_building_in_sites               FALSE      FALSE
## heat_fuel                              FALSE      FALSE
## supply_area                            FALSE      FALSE
## total_household_count_of_area_type     FALSE      FALSE
## room_count                             FALSE      FALSE
## bathroom_count                         FALSE      FALSE
## heat_type_central                      FALSE      FALSE
## heat_type_district                     FALSE      FALSE
## front_door_structure_corridor          FALSE      FALSE
## front_door_structure_mixed             FALSE      FALSE
## 1 subsets of each size up to 21
## Selection Algorithm: forward
##           city transaction_year_month transaction_date year_of_completion
## 1  ( 1 )  " "  " "                    " "              " "
## 2  ( 1 )  "*"  " "                    " "              " "
## 3  ( 1 )  "*"  " "                    " "              " "
## 4  ( 1 )  "*"  "*"                    " "              " "
## 5  ( 1 )  "*"  "*"                    " "              " "
## 6  ( 1 )  "*"  "*"                    " "              " "
## 7  ( 1 )  "*"  "*"                    " "              " "
## 8  ( 1 )  "*"  "*"                    " "              " "
## 9  ( 1 )  "*"  "*"                    " "              " "
## 10  ( 1 ) "*"  "*"                    " "              " "
## 11  ( 1 ) "*"  "*"                    " "              " "
## 12  ( 1 ) "*"  "*"                    " "              "*"
## 13  ( 1 ) "*"  "*"                    " "              "*"
## 14  ( 1 ) "*"  "*"                    " "              "*"
## 15  ( 1 ) "*"  "*"                    " "              "*"
## 16  ( 1 ) "*"  "*"                    " "              "*"
## 17  ( 1 ) "*"  "*"                    " "              "*"
## 18  ( 1 ) "*"  "*"                    " "              "*"
## 19  ( 1 ) "*"  "*"                    " "              "*"
## 20  ( 1 ) "*"  "*"                    " "              "*"
## 21  ( 1 ) "*"  "*"                    " "              "*"
##           exclusive_use_area floor latitude longitude address_by_law
## 1  ( 1 )  " "                " "   " "      " "       " "
## 2  ( 1 )  " "                " "   " "      " "       " "
```

```
## 3  ( 1 ) " "                " "   " "     " "       " "
## 4  ( 1 ) " "                " "   " "     " "       " "
## 5  ( 1 ) " "                " "   "*"     " "       " "
## 6  ( 1 ) " "                " "   "*"     " "       " "
## 7  ( 1 ) " "                " "   "*"     "*"       " "
## 8  ( 1 ) " "                " "   "*"     "*"       " "
## 9  ( 1 ) " "                " "   "*"     "*"       " "
## 10  ( 1 ) " "               " "   "*"     "*"       "*"
## 11  ( 1 ) " "               " "   "*"     "*"       "*"
## 12  ( 1 ) " "               " "   "*"     "*"       "*"
## 13  ( 1 ) " "               "*"   "*"     "*"       "*"
## 14  ( 1 ) " "               "*"   "*"     "*"       "*"
## 15  ( 1 ) " "               "*"   "*"     "*"       "*"
## 16  ( 1 ) "*"               "*"   "*"     "*"       "*"
## 17  ( 1 ) "*"               "*"   "*"     "*"       "*"
## 18  ( 1 ) "*"               "*"   "*"     "*"       "*"
## 19  ( 1 ) "*"               "*"   "*"     "*"       "*"
## 20  ( 1 ) "*"               "*"   "*"     "*"       "*"
## 21  ( 1 ) "*"               "*"   "*"     "*"       "*"
##           total_parking_capacity_in_site total_household_count_in_sites
## 1  ( 1 )  " "                            " "
## 2  ( 1 )  " "                            " "
## 3  ( 1 )  " "                            " "
## 4  ( 1 )  " "                            " "
## 5  ( 1 )  " "                            " "
## 6  ( 1 )  " "                            " "
## 7  ( 1 )  " "                            " "
## 8  ( 1 )  " "                            " "
## 9  ( 1 )  " "                            "*"
## 10  ( 1 ) " "                            "*"
## 11  ( 1 ) "*"                            "*"
## 12  ( 1 ) "*"                            "*"
## 13  ( 1 ) "*"                            "*"
## 14  ( 1 ) "*"                            "*"
## 15  ( 1 ) "*"                            "*"
## 16  ( 1 ) "*"                            "*"
## 17  ( 1 ) "*"                            "*"
## 18  ( 1 ) "*"                            "*"
## 19  ( 1 ) "*"                            "*"
## 20  ( 1 ) "*"                            "*"
## 21  ( 1 ) "*"                            "*"
##           apartment_building_count_in_sites tallest_building_in_sites
## 1  ( 1 )  " "                               " "
## 2  ( 1 )  " "                               " "
## 3  ( 1 )  "*"                               " "
## 4  ( 1 )  "*"                               " "
## 5  ( 1 )  "*"                               " "
## 6  ( 1 )  "*"                               " "
## 7  ( 1 )  "*"                               " "
## 8  ( 1 )  "*"                               " "
## 9  ( 1 )  "*"                               " "
## 10  ( 1 ) "*"                               " "
## 11  ( 1 ) "*"                               " "
## 12  ( 1 ) "*"                               " "
```

```
## 13  ( 1 ) "*"                              " "
## 14  ( 1 ) "*"                              "*"
## 15  ( 1 ) "*"                              "*"
## 16  ( 1 ) "*"                              "*"
## 17  ( 1 ) "*"                              "*"
## 18  ( 1 ) "*"                              "*"
## 19  ( 1 ) "*"                              "*"
## 20  ( 1 ) "*"                              "*"
## 21  ( 1 ) "*"                              "*"
##           lowest_building_in_sites heat_fuel supply_area
## 1  ( 1 )  " "                      " "       "*"
## 2  ( 1 )  " "                      " "       "*"
## 3  ( 1 )  " "                      " "       "*"
## 4  ( 1 )  " "                      " "       "*"
## 5  ( 1 )  " "                      " "       "*"
## 6  ( 1 )  " "                      " "       "*"
## 7  ( 1 )  " "                      " "       "*"
## 8  ( 1 )  "*"                      " "       "*"
## 9  ( 1 )  "*"                      " "       "*"
## 10  ( 1 ) "*"                      " "       "*"
## 11  ( 1 ) "*"                      " "       "*"
## 12  ( 1 ) "*"                      " "       "*"
## 13  ( 1 ) "*"                      " "       "*"
## 14  ( 1 ) "*"                      " "       "*"
## 15  ( 1 ) "*"                      "*"       "*"
## 16  ( 1 ) "*"                      "*"       "*"
## 17  ( 1 ) "*"                      "*"       "*"
## 18  ( 1 ) "*"                      "*"       "*"
## 19  ( 1 ) "*"                      "*"       "*"
## 20  ( 1 ) "*"                      "*"       "*"
## 21  ( 1 ) "*"                      "*"       "*"
##           total_household_count_of_area_type room_count bathroom_count
## 1  ( 1 )  " "                                " "        " "
## 2  ( 1 )  " "                                " "        " "
## 3  ( 1 )  " "                                " "        " "
## 4  ( 1 )  " "                                " "        " "
## 5  ( 1 )  " "                                " "        " "
## 6  ( 1 )  " "                                " "        " "
## 7  ( 1 )  " "                                " "        " "
## 8  ( 1 )  " "                                " "        " "
## 9  ( 1 )  " "                                " "        " "
## 10  ( 1 ) " "                                " "        " "
## 11  ( 1 ) " "                                " "        " "
## 12  ( 1 ) " "                                " "        " "
## 13  ( 1 ) " "                                " "        " "
## 14  ( 1 ) " "                                " "        " "
## 15  ( 1 ) " "                                " "        " "
## 16  ( 1 ) " "                                " "        " "
## 17  ( 1 ) "*"                                " "        " "
## 18  ( 1 ) "*"                                " "        " "
## 19  ( 1 ) "*"                                "*"        " "
## 20  ( 1 ) "*"                                "*"        "*"
## 21  ( 1 ) "*"                                "*"        "*"
##           heat_type_central heat_type_district front_door_structure_corridor
```

```
## 1  ( 1 )  " "                " "                 " "
## 2  ( 1 )  " "                " "                 " "
## 3  ( 1 )  " "                " "                 " "
## 4  ( 1 )  " "                " "                 " "
## 5  ( 1 )  " "                " "                 " "
## 6  ( 1 )  " "                "*"                 " "
## 7  ( 1 )  " "                "*"                 " "
## 8  ( 1 )  " "                "*"                 " "
## 9  ( 1 )  " "                "*"                 " "
## 10  ( 1 ) " "                "*"                 " "
## 11  ( 1 ) " "                "*"                 " "
## 12  ( 1 ) " "                "*"                 " "
## 13  ( 1 ) " "                "*"                 " "
## 14  ( 1 ) " "                "*"                 " "
## 15  ( 1 ) " "                "*"                 " "
## 16  ( 1 ) " "                "*"                 " "
## 17  ( 1 ) " "                "*"                 " "
## 18  ( 1 ) " "                "*"                 "*"
## 19  ( 1 ) " "                "*"                 "*"
## 20  ( 1 ) " "                "*"                 "*"
## 21  ( 1 ) "*"                "*"                 "*"
##           front_door_structure_mixed
## 1  ( 1 )  " "
## 2  ( 1 )  " "
## 3  ( 1 )  " "
## 4  ( 1 )  " "
## 5  ( 1 )  " "
## 6  ( 1 )  " "
## 7  ( 1 )  " "
## 8  ( 1 )  " "
## 9  ( 1 )  " "
## 10  ( 1 ) " "
## 11  ( 1 ) " "
## 12  ( 1 ) " "
## 13  ( 1 ) " "
## 14  ( 1 ) " "
## 15  ( 1 ) " "
## 16  ( 1 ) " "
## 17  ( 1 ) " "
## 18  ( 1 ) " "
## 19  ( 1 ) " "
## 20  ( 1 ) " "
## 21  ( 1 ) " "
```

```
x1f <- step.model_f$results[,2]
plot(x1f,type = "o",col = "red",xlab = "Number of Variables Added", ylab = "RMSE",
   main = "Forward Train RMSE")
legend("topright",
  legend = c("Train"),
  col = c("red"),
  pch = c(1,1),
  bty = "n",
  pt.cex = 2,
  cex = 1.2,
  text.col = "black",
  horiz = F ,
  inset = c(0.1, 0.1))
```



Backward selection

In this model, we came to the same conclusion that the most optimal model has 21 variable with the lowest RMSE of 188,321,979 Korean Won.

```
set.seed(810)
#Backward train
# Set up repeated k-fold cross-validation
train.control <- trainControl(method = "cv", number = 10)
# Train the model
step.model_b <- train(transaction_real_price ~., data = dd_train,
                    method = "leapBackward",
                    tuneGrid = data.frame(nvmax = 1:23),
                    trControl = train.control
                    )
step.model_b$results
```

```
##    nvmax      RMSE  Rsquared       MAE  RMSESD  RsquaredSD     MAESD
## 1      1 275820794 0.2633316 183550247 5271468 0.012969232 1810020.1
## 2      2 234909141 0.4656794 153777481 5215660 0.009970077 1767563.1
## 3      3 225983655 0.5055657 147442830 5086690 0.008856248 1418133.3
## 4      4 216077790 0.5479695 141673953 5201928 0.009422754 1347680.6
## 5      5 206243197 0.5882408 132030639 5346512 0.008208107 1244955.9
## 6      6 200214993 0.6119412 126968771 5216558 0.008366577 1771089.2
## 7      7 194816961 0.6325771 123768489 5318325 0.008077787 1169235.0
## 8      8 193088790 0.6390701 122727058 5497882 0.008709827 1068180.3
## 9      9 191591793 0.6446538 121386085 5458523 0.008446345  973728.1
## 10    10 190617456 0.6482669 120455377 5536638 0.008785837 1044868.6
## 11    11 190082271 0.6502400 119726562 5583167 0.009057233 1054663.3
## 12    12 189478964 0.6524544 119972543 5689114 0.009350330 1177436.5
## 13    13 188935542 0.6544445 119543844 5737249 0.009421530 1241138.6
## 14    14 188699814 0.6553071 119325857 5759809 0.009505695 1226964.7
## 15    15 188635164 0.6555458 119113982 5813140 0.009615091 1226816.5
## 16    16 188427962 0.6563037 118836075 5841570 0.009779370 1225391.8
## 17    17 188381716 0.6564717 118850989 5837784 0.009779374 1219600.3
## 18    18 188354341 0.6565718 118734190 5845501 0.009779065 1224907.8
## 19    19 188347708 0.6565947 118769638 5841846 0.009778721 1221526.2
## 20    20 188340273 0.6566225 118745450 5832668 0.009756786 1234081.0
## 21    21 188321979 0.6566908 118720513 5817163 0.009693742 1232452.6
## 22    22 188325117 0.6566793 118720657 5815158 0.009685612 1233727.1
## 23    23 188324643 0.6566810 118717131 5814604 0.009682559 1234709.1
```

Two variable we exclude is transaction_date and front_door_structure_mixed

```
#Choose the best tuned model
summary(step.model_b$finalModel)
```

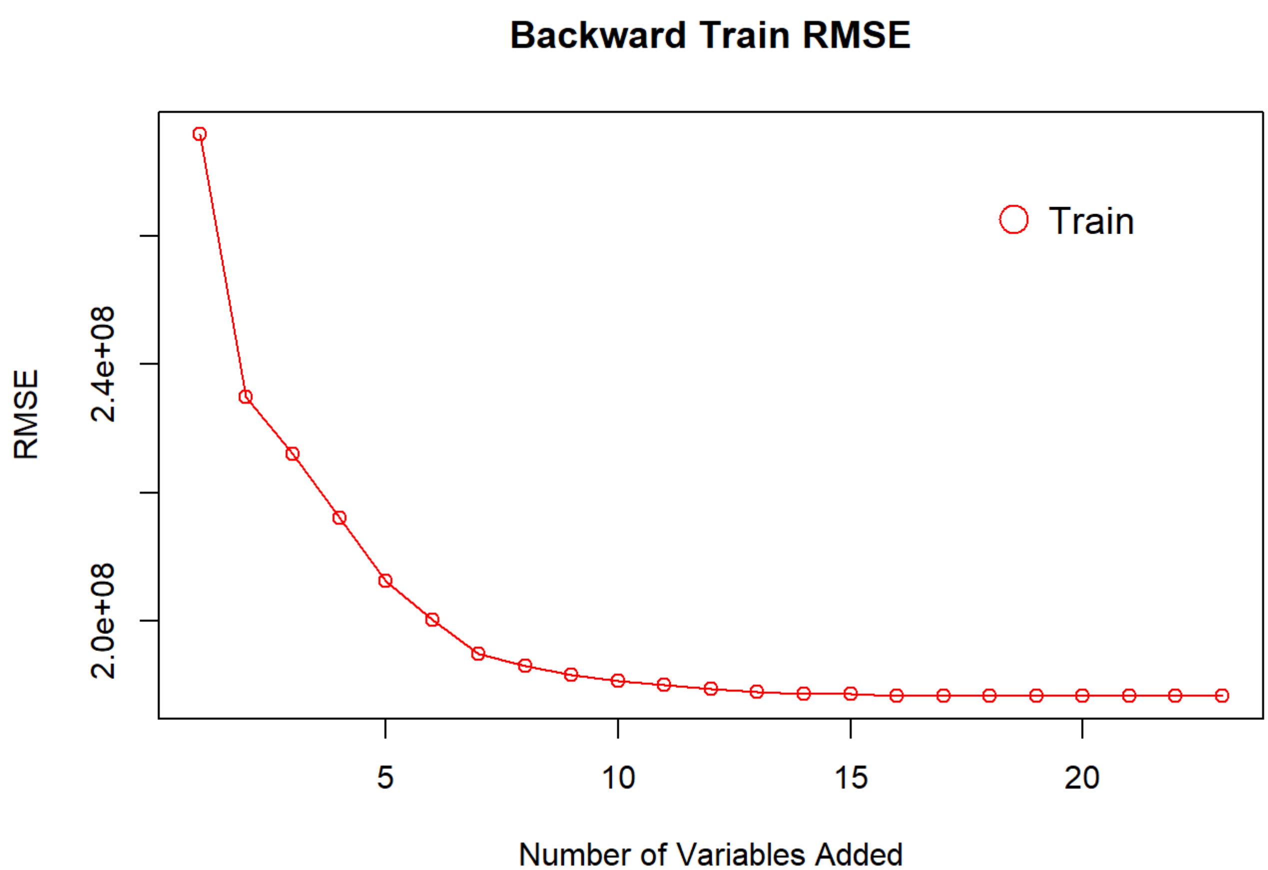
```
## Subset selection object
## 23 Variables  (and intercept)
##                                    Forced in Forced out
## city                                   FALSE      FALSE
## transaction_year_month                 FALSE      FALSE
## transaction_date                       FALSE      FALSE
## year_of_completion                     FALSE      FALSE
## exclusive_use_area                     FALSE      FALSE
## floor                                  FALSE      FALSE
## latitude                               FALSE      FALSE
## longitude                              FALSE      FALSE
## address_by_law                         FALSE      FALSE
## total_parking_capacity_in_site         FALSE      FALSE
## total_household_count_in_sites         FALSE      FALSE
## apartment_building_count_in_sites      FALSE      FALSE
## tallest_building_in_sites              FALSE      FALSE
## lowest_building_in_sites               FALSE      FALSE
## heat_fuel                              FALSE      FALSE
## supply_area                            FALSE      FALSE
## total_household_count_of_area_type     FALSE      FALSE
## room_count                             FALSE      FALSE
## bathroom_count                         FALSE      FALSE
## heat_type_central                      FALSE      FALSE
## heat_type_district                     FALSE      FALSE
## front_door_structure_corridor          FALSE      FALSE
## front_door_structure_mixed             FALSE      FALSE
## 1 subsets of each size up to 21
## Selection Algorithm: backward
##           city transaction_year_month transaction_date year_of_completion
## 1  ( 1 )  " "  " "                    " "              " "
## 2  ( 1 )  "*"  " "                    " "              " "
## 3  ( 1 )  "*"  " "                    " "              " "
## 4  ( 1 )  "*"  " "                    " "              " "
## 5  ( 1 )  "*"  "*"                    " "              " "
## 6  ( 1 )  "*"  "*"                    " "              " "
## 7  ( 1 )  "*"  "*"                    " "              " "
## 8  ( 1 )  "*"  "*"                    " "              " "
## 9  ( 1 )  "*"  "*"                    " "              " "
## 10  ( 1 ) "*"  "*"                    " "              " "
## 11  ( 1 ) "*"  "*"                    " "              " "
## 12  ( 1 ) "*"  "*"                    " "              "*"
## 13  ( 1 ) "*"  "*"                    " "              "*"
## 14  ( 1 ) "*"  "*"                    " "              "*"
## 15  ( 1 ) "*"  "*"                    " "              "*"
## 16  ( 1 ) "*"  "*"                    " "              "*"
## 17  ( 1 ) "*"  "*"                    " "              "*"
## 18  ( 1 ) "*"  "*"                    " "              "*"
## 19  ( 1 ) "*"  "*"                    " "              "*"
## 20  ( 1 ) "*"  "*"                    " "              "*"
## 21  ( 1 ) "*"  "*"                    " "              "*"
##           exclusive_use_area floor latitude longitude address_by_law
## 1  ( 1 )  " "                " "   " "      " "       " "
## 2  ( 1 )  " "                " "   " "      " "       " "
```

```
## 3  ( 1 ) " "                " "   "*"      " "         " "
## 4  ( 1 ) " "                " "   "*"      " "         " "
## 5  ( 1 ) " "                " "   "*"      " "         " "
## 6  ( 1 ) " "                " "   "*"      " "         " "
## 7  ( 1 ) " "                " "   "*"      "*"         " "
## 8  ( 1 ) " "                " "   "*"      "*"         " "
## 9  ( 1 ) " "                " "   "*"      "*"         " "
## 10  ( 1 ) " "               " "   "*"      "*"         "*"
## 11  ( 1 ) " "               " "   "*"      "*"         "*"
## 12  ( 1 ) " "               " "   "*"      "*"         "*"
## 13  ( 1 ) " "               "*"   "*"      "*"         "*"
## 14  ( 1 ) " "               "*"   "*"      "*"         "*"
## 15  ( 1 ) " "               "*"   "*"      "*"         "*"
## 16  ( 1 ) "*"               "*"   "*"      "*"         "*"
## 17  ( 1 ) "*"               "*"   "*"      "*"         "*"
## 18  ( 1 ) "*"               "*"   "*"      "*"         "*"
## 19  ( 1 ) "*"               "*"   "*"      "*"         "*"
## 20  ( 1 ) "*"               "*"   "*"      "*"         "*"
## 21  ( 1 ) "*"               "*"   "*"      "*"         "*"
##            total_parking_capacity_in_site total_household_count_in_sites
## 1  ( 1 )  " "                            " "
## 2  ( 1 )  " "                            " "
## 3  ( 1 )  " "                            " "
## 4  ( 1 )  " "                            " "
## 5  ( 1 )  " "                            " "
## 6  ( 1 )  " "                            " "
## 7  ( 1 )  " "                            " "
## 8  ( 1 )  " "                            " "
## 9  ( 1 )  " "                            "*"
## 10  ( 1 ) " "                            "*"
## 11  ( 1 ) "*"                            "*"
## 12  ( 1 ) "*"                            "*"
## 13  ( 1 ) "*"                            "*"
## 14  ( 1 ) "*"                            "*"
## 15  ( 1 ) "*"                            "*"
## 16  ( 1 ) "*"                            "*"
## 17  ( 1 ) "*"                            "*"
## 18  ( 1 ) "*"                            "*"
## 19  ( 1 ) "*"                            "*"
## 20  ( 1 ) "*"                            "*"
## 21  ( 1 ) "*"                            "*"
##            apartment_building_count_in_sites tallest_building_in_sites
## 1  ( 1 )  " "                               " "
## 2  ( 1 )  " "                               " "
## 3  ( 1 )  " "                               " "
## 4  ( 1 )  " "                               " "
## 5  ( 1 )  " "                               " "
## 6  ( 1 )  "*"                               " "
## 7  ( 1 )  "*"                               " "
## 8  ( 1 )  "*"                               " "
## 9  ( 1 )  "*"                               " "
## 10  ( 1 ) "*"                               " "
## 11  ( 1 ) "*"                               " "
## 12  ( 1 ) "*"                               " "
```

```
## 13  ( 1 ) "*"                              " "
## 14  ( 1 ) "*"                              "*"
## 15  ( 1 ) "*"                              "*"
## 16  ( 1 ) "*"                              "*"
## 17  ( 1 ) "*"                              "*"
## 18  ( 1 ) "*"                              "*"
## 19  ( 1 ) "*"                              "*"
## 20  ( 1 ) "*"                              "*"
## 21  ( 1 ) "*"                              "*"
##           lowest_building_in_sites heat_fuel supply_area
## 1  ( 1 )  " "                      " "       "*"
## 2  ( 1 )  " "                      " "       "*"
## 3  ( 1 )  " "                      " "       "*"
## 4  ( 1 )  " "                      " "       "*"
## 5  ( 1 )  " "                      " "       "*"
## 6  ( 1 )  " "                      " "       "*"
## 7  ( 1 )  " "                      " "       "*"
## 8  ( 1 )  "*"                      " "       "*"
## 9  ( 1 )  "*"                      " "       "*"
## 10  ( 1 ) "*"                      " "       "*"
## 11  ( 1 ) "*"                      " "       "*"
## 12  ( 1 ) "*"                      " "       "*"
## 13  ( 1 ) "*"                      " "       "*"
## 14  ( 1 ) "*"                      " "       "*"
## 15  ( 1 ) "*"                      "*"       "*"
## 16  ( 1 ) "*"                      "*"       "*"
## 17  ( 1 ) "*"                      "*"       "*"
## 18  ( 1 ) "*"                      "*"       "*"
## 19  ( 1 ) "*"                      "*"       "*"
## 20  ( 1 ) "*"                      "*"       "*"
## 21  ( 1 ) "*"                      "*"       "*"
##           total_household_count_of_area_type room_count bathroom_count
## 1  ( 1 )  " "                                " "        " "
## 2  ( 1 )  " "                                " "        " "
## 3  ( 1 )  " "                                " "        " "
## 4  ( 1 )  " "                                " "        " "
## 5  ( 1 )  " "                                " "        " "
## 6  ( 1 )  " "                                " "        " "
## 7  ( 1 )  " "                                " "        " "
## 8  ( 1 )  " "                                " "        " "
## 9  ( 1 )  " "                                " "        " "
## 10  ( 1 ) " "                                " "        " "
## 11  ( 1 ) " "                                " "        " "
## 12  ( 1 ) " "                                " "        " "
## 13  ( 1 ) " "                                " "        " "
## 14  ( 1 ) " "                                " "        " "
## 15  ( 1 ) " "                                " "        " "
## 16  ( 1 ) " "                                " "        " "
## 17  ( 1 ) "*"                                " "        " "
## 18  ( 1 ) "*"                                " "        " "
## 19  ( 1 ) "*"                                "*"        " "
## 20  ( 1 ) "*"                                "*"        "*"
## 21  ( 1 ) "*"                                "*"        "*"
##           heat_type_central heat_type_district front_door_structure_corridor
```

```
## 1  ( 1 )  " "                " "                 " "
## 2  ( 1 )  " "                " "                 " "
## 3  ( 1 )  " "                " "                 " "
## 4  ( 1 )  " "                "*"                 " "
## 5  ( 1 )  " "                "*"                 " "
## 6  ( 1 )  " "                "*"                 " "
## 7  ( 1 )  " "                "*"                 " "
## 8  ( 1 )  " "                "*"                 " "
## 9  ( 1 )  " "                "*"                 " "
## 10  ( 1 ) " "                "*"                 " "
## 11  ( 1 ) " "                "*"                 " "
## 12  ( 1 ) " "                "*"                 " "
## 13  ( 1 ) " "                "*"                 " "
## 14  ( 1 ) " "                "*"                 " "
## 15  ( 1 ) " "                "*"                 " "
## 16  ( 1 ) " "                "*"                 " "
## 17  ( 1 ) " "                "*"                 " "
## 18  ( 1 ) " "                "*"                 "*"
## 19  ( 1 ) " "                "*"                 "*"
## 20  ( 1 ) " "                "*"                 "*"
## 21  ( 1 ) "*"                "*"                 "*"
##           front_door_structure_mixed
## 1  ( 1 )  " "
## 2  ( 1 )  " "
## 3  ( 1 )  " "
## 4  ( 1 )  " "
## 5  ( 1 )  " "
## 6  ( 1 )  " "
## 7  ( 1 )  " "
## 8  ( 1 )  " "
## 9  ( 1 )  " "
## 10  ( 1 ) " "
## 11  ( 1 ) " "
## 12  ( 1 ) " "
## 13  ( 1 ) " "
## 14  ( 1 ) " "
## 15  ( 1 ) " "
## 16  ( 1 ) " "
## 17  ( 1 ) " "
## 18  ( 1 ) " "
## 19  ( 1 ) " "
## 20  ( 1 ) " "
## 21  ( 1 ) " "
```

```
x1 <- step.model_b$results[,2]
plot(x1,type = "o",col = "red",xlab = "Number of Variables Added", ylab = "RMSE",
   main = "Backward Train RMSE")
legend("topright",
  legend = c("Train"),
  col = c("red"),
  pch = c(1,1),
  bty = "n",
  pt.cex = 2,
  cex = 1.2,
  text.col = "black",
  horiz = F ,
  inset = c(0.1, 0.1))
```



Since the model chosen by Forward/Backward is the same containing the same variables, let's fit it to calculate the final test RMSE. So, Forward/Backward Selection Regression Test RMSE is 188,550,321 Korean Won.

```
# Fit the model chosen by forward and backward
set.seed(810)
train.control <- trainControl(method = "cv", number = 10)
# Train the model
fit_fb <- train(transaction_real_price ~ city + transaction_year_month  + year_of_completion + e
xclusive_use_area + floor + longitude + latitude + address_by_law + total_parking_capacity_in_si
te + total_household_count_in_sites + apartment_building_count_in_sites + tallest_building_in_si
tes + lowest_building_in_sites + heat_type_central +heat_type_district + heat_fuel + supply_area
+ total_household_count_of_area_type + room_count + bathroom_count +front_door_structure_corrido
r, data=dd_train, method = "lm",
               trControl = train.control)
# Summarize the results
print(fit_fb)
```

```
## Linear Regression
##
## 128000 samples
##     21 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 115201, 115201, 115199, 115201, 115199, 115199, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   188321979  0.6566908  118720513
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
yhat_test_fb <- predict(fit_fb, dd_test)
mse_test_fb <- mean((y_test - yhat_test_fb)**2)
paste("Forward/Backward Selection Regression Test RMSE",sqrt(mse_test_fb))
```

```
## [1] "Forward/Backward Selection Regression Test RMSE 188550321.251422"
```

Setup for Ridge and Lasso Regression

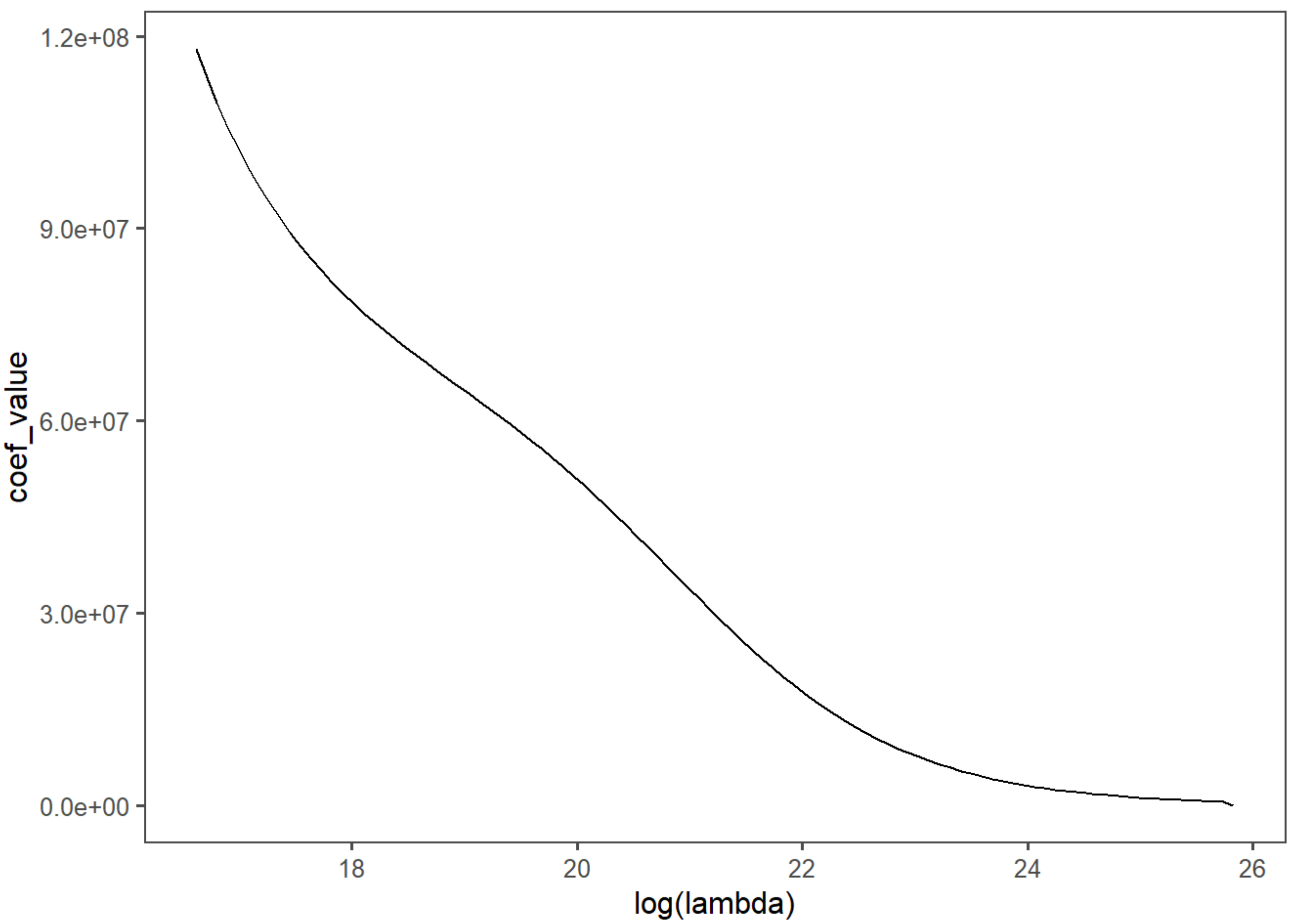
We created a new formula that includes all 23 predictors, and split out test and train datasets

```
# added all of the variables to the formula so that we can have 24 predictors
f2 <- as.formula(transaction_real_price ~  city + transaction_year_month + transaction_date + ye
ar_of_completion + exclusive_use_area + floor + longitude + latitude + address_by_law + total_pa
rking_capacity_in_site + total_household_count_in_sites + apartment_building_count_in_sites + ta
llest_building_in_sites + lowest_building_in_sites + heat_type_central +heat_type_district + hea
t_fuel + supply_area + total_household_count_of_area_type + room_count + bathroom_count + front_
door_structure_mixed +front_door_structure_corridor)

x1_train_sample <- model.matrix(f2, dd_train)[,-1]
x1_test <- model.matrix(f2, dd_test)[,-1]
```

Ridge Regression

We used cross validation to identify the best lambda value. We can see that this model actually performs slightly worse compared to linear regression. The graph reveals that as the training ran, our coefficients converged towards zero.

```
# fit the ridge regression using the cross validation data
fit.ridge <- cv.glmnet(x1_train_sample, y_train, alpha = 0)

# make predictions using fitted model
ridge.coef <- predict(fit.ridge,
        type = "coefficients",
        s = fit.ridge$lambda)
to_plot <- data.table(
  lambda = fit.ridge$lambda,
  coef_value = ridge.coef[2, ]
)

# plot the coefficient values for different values of lambda
ggplot(to_plot, aes(log(lambda), coef_value)) +
  geom_line() +
  theme_few()
```



```
# MSE for train
yhat.train.ridge <- predict(fit.ridge, x1_train_sample, s = fit.ridge$lambda.min)
mse.train.ridge <- mean((y_train - yhat.train.ridge)^2)

# MSE to test
yhat.test.ridge <- predict(fit.ridge, x1_test, s = fit.ridge$lambda.min)
mse.test.ridge <- mean((y_test - yhat.test.ridge)^2)

cat("Train RMSE: ",sqrt(mse.train.ridge))
```

```
## Train RMSE:  201423543
```

```
cat(" Test RMSE: ",sqrt(mse.test.ridge))
```

```
##  Test RMSE:  201931073
```

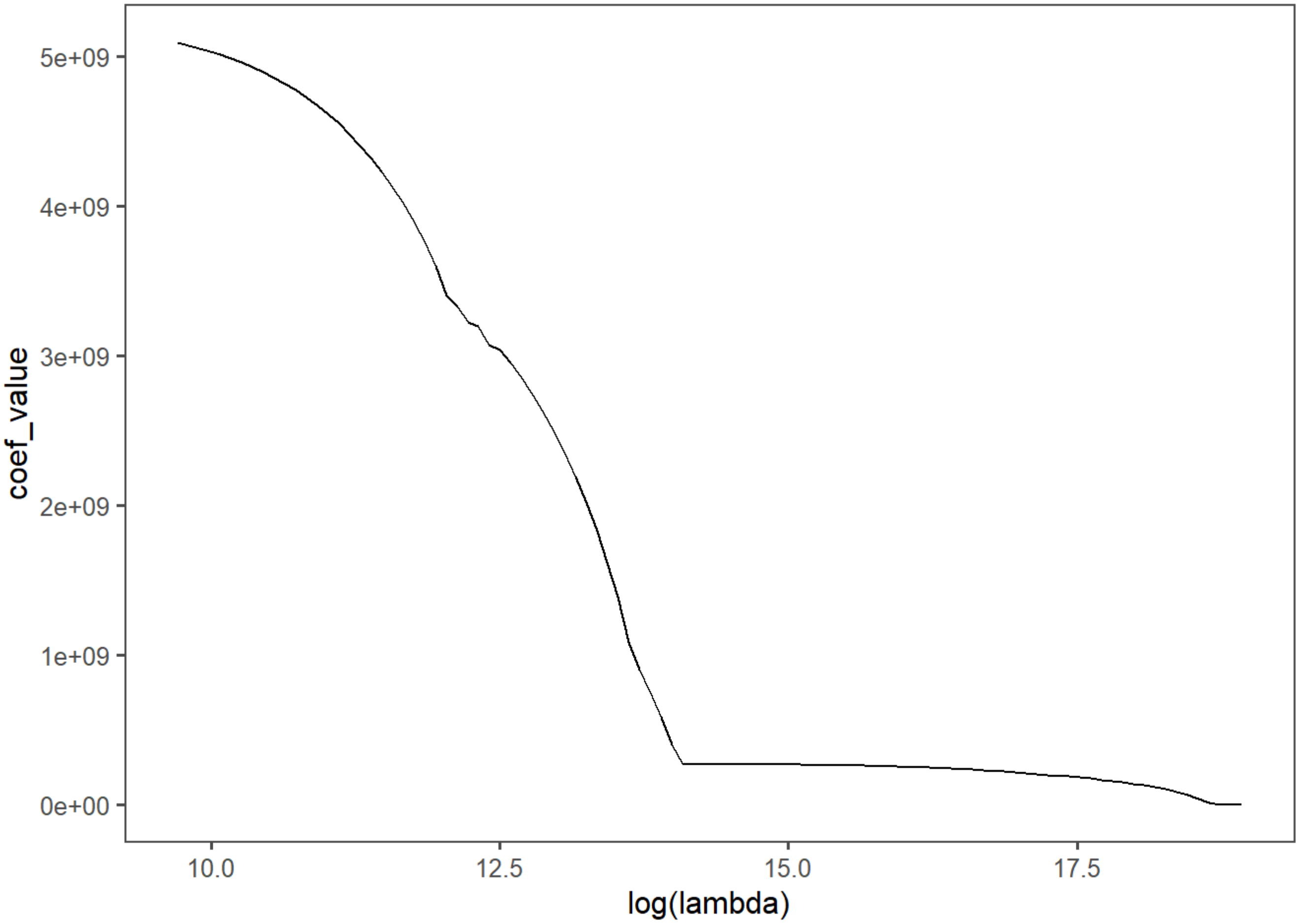
```
cat(" Best Lambda: ", fit.ridge$lambda.min)
```

```
##  Best Lambda:  16491016
```

Lasso Regression

In this model, we again used cross validation to find the optimal lambda value. In the output below, we can see the coefficient value associated with each of our predictors. None of the coefficient values are zero,increasing confidence that we are not overfiting, and that all of our features are contributing to the model.

```
fit.lasso <- cv.glmnet(x1_train_sample, y_train, alpha = 1)

# predict based on most optimal lambda found above
lasso.coef <- predict(fit.lasso,
        type = "coefficients",
        s = fit.lasso$lambda)
to_plot <- data.table(
  lambda = fit.lasso$lambda,
  coef_value = lasso.coef[2, ]
)

# plot the coefficient values for different values of lambda
ggplot(to_plot, aes(log(lambda), coef_value)) +
  geom_line() +
  theme_few()
```




```
yhat.train.lasso <- predict(fit.lasso, x1_train_sample, s = fit.lasso$lambda.min)
mse.train.lasso <- mean((y_train - yhat.train.lasso)^2)

yhat.test.lasso <- predict(fit.lasso, x1_test, s = fit.lasso$lambda.min)
mse.test.lasso <- mean((y_test - yhat.test.lasso)^2)

cat("Train RMSE: ",sqrt(mse.train.lasso))
```

```
## Train RMSE:  188454633
```

```
cat(" Test RMSE: ",sqrt(mse.test.lasso))
```

```
##  Test RMSE:  188678874
```

```
cat(" Best Lambda: ", fit.lasso$lambda.min)
```

```
##  Best Lambda:  16491.02
```

Decision Tree

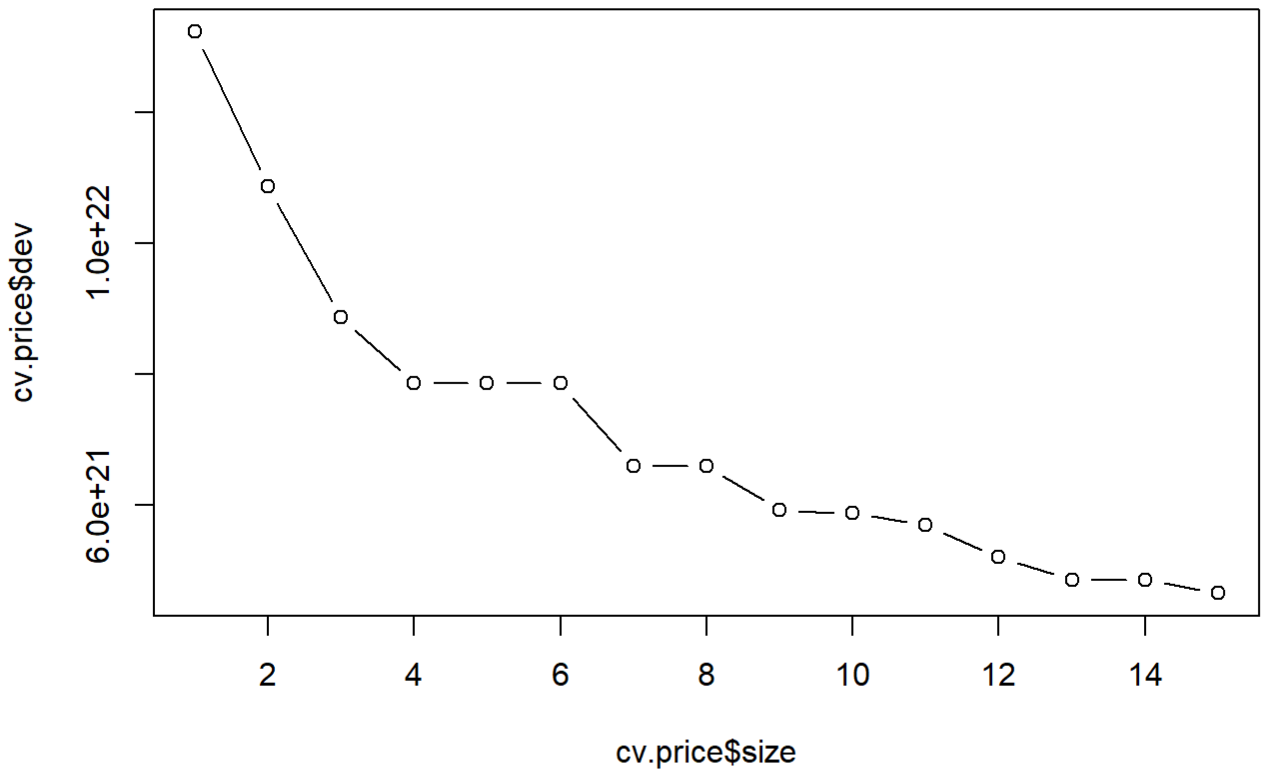
In our decision tree model, we use row_index to get the same observations used in the previous models.

```
tree.price = tree(transaction_real_price ~ . , Price, subset = row_index)

summary(tree.price)
```

```
##
## Regression tree:
## tree(formula = transaction_real_price ~ ., data = Price, subset = row_index)
## Variables actually used in tree construction:
## [1] "supply_area"                    "latitude"
## [3] "address_by_law"                 "exclusive_use_area"
## [5] "heat_fuel"                      "longitude"
## [7] "tallest_building_in_sites"      "apartment_building_count_in_sites"
## [9] "transaction_year_month"
## Number of terminal nodes:  15
## Residual mean deviance:  3.519e+16 = 4.504e+21 / 128000
## Distribution of residuals:
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -1.691e+09 -9.586e+07 -2.407e+07  0.000e+00  6.814e+07  3.883e+09
```

We can see that we don’t have to prune that tree because the largest tree (size = 14), has the lowest cross validation error.

```
# Prune Tree
cv.price = cv.tree(tree.price)
plot(cv.price$size,cv.price$dev, type = "b")
```




```
cv.price
```

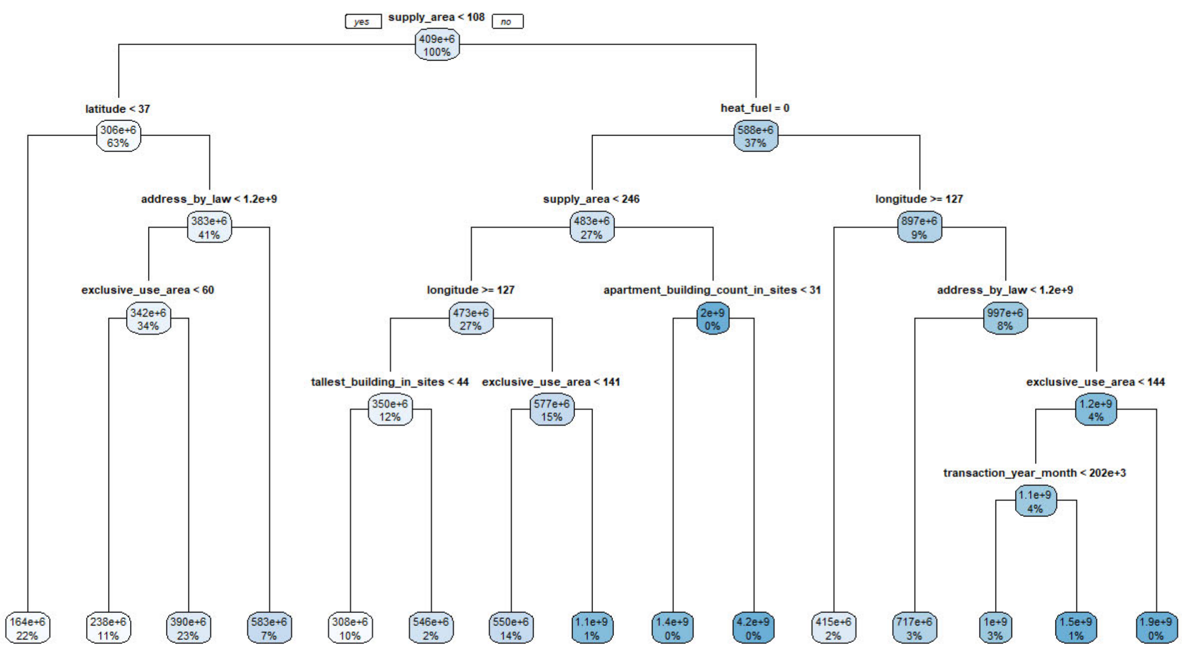
```
## $size
##  [1] 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
##
## $dev
##  [1] 4.657197e+21 4.864690e+21 4.864690e+21 5.202028e+21 5.700069e+21
##  [6] 5.873078e+21 5.925532e+21 6.598284e+21 6.598284e+21 7.858926e+21
## [11] 7.858926e+21 7.858926e+21 8.862230e+21 1.086308e+22 1.322130e+22
##
## $k
##  [1]         -Inf 1.333288e+20 1.368796e+20 2.168168e+20 2.952076e+20
##  [6] 3.052146e+20 3.356930e+20 4.303563e+20 4.439117e+20 5.320393e+20
## [11] 5.576437e+20 5.724015e+20 8.868482e+20 1.512390e+21 2.358311e+21
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

Our RMSE below is 189,891,653.318 Korean Won.

```
#Test MSE
tree.yhat = predict(tree.price,newdata=dd_test)
mean((tree.yhat - y_test)^2)
```

```
## [1] 3.692462e+16
```

```
#Lecture Method to deal with decision tree
set.seed(217)
tree.price.lec <- rpart(transaction_real_price ~ ., Price,subset = row_index,
                    control = rpart.control(cp=0.01))
rpart.plot(tree.price.lec,type = 1)
```



Bagging

```
#### Trends of Test MSE as number of data in training growing (1% to 5%)
#set.seed(217)
#test.mse = c()

#for (i in seq(1,5)) {
 # train = sample(1:nrow(Price),(nrow(Price)/100)*i)
#  tree.testy = Price[-train,transaction_real_price]
#  tree.test = Price[-train]
#  bag.price = randomForest(transaction_real_price ~ ., data = Price, subset = train, mtry = 24,
importance = TRUE)
#  yhat.bag = predict(bag.price, newdata = tree.test)
 # test.mse = c(test.mse,mean((tree.testy-yhat.bag)^2))
#}

#test.mse
```

Bagging using 5000 rows as training

We decided to use 5000 rows for compute resource reasons. We then used the remaining data to calculate the test MSE. We found that there was a significant improvement in the RMSE, which we calculate to be 101,033,509.293 Korean Won. Also, we found that when we increased the number of observations in the train dataset, our MSE went down significantly.

```
bag.price = randomForest(transaction_real_price ~ ., data = dd_train, mtry=21, importance = TRUE
)
yhat.bag.train = predict(bag.price, newdata = dd_train)
cat("RMSE train: ", sqrt(mean((y_train-yhat.bag.train)^2)))
```

```
## RMSE train:  21527264
```

```
yhat.bag = predict(bag.price, newdata = dd_test)
cat(" RMSE test: ", sqrt(mean((y_test-yhat.bag)^2)))
```

```
##  RMSE test:  48105079
```

Random Forest

This process is similar to bagging; we took a sample of 5000 observations from the dataset. We then ran the random forest model on out sample, and computed an MSE. The MSE was again an improvement over some of the less flexible models earlier in the report. We calculated a test MSE of 102,026,222.12 Korean Won. Due to the fact that we are only able to take a sample of 5000 observations, the model has higher variability since the MSE can change significantly every time we run the model.

```
rf.price = randomForest(transaction_real_price ~ ., data = Price, subset = row_index, mtry = 5,
 importance = TRUE)

yhat.rf.train = predict(rf.price, newdata = dd_train)
cat("RMSE train: ", sqrt(mean((y_train-yhat.rf.train)^2)))
```

```
## RMSE train:  34080272
```

```
yhat.rf = predict(rf.price, newdata = dd_test)
cat(" RMSE test: ", sqrt(mean((y_test-yhat.rf)^2)))
```

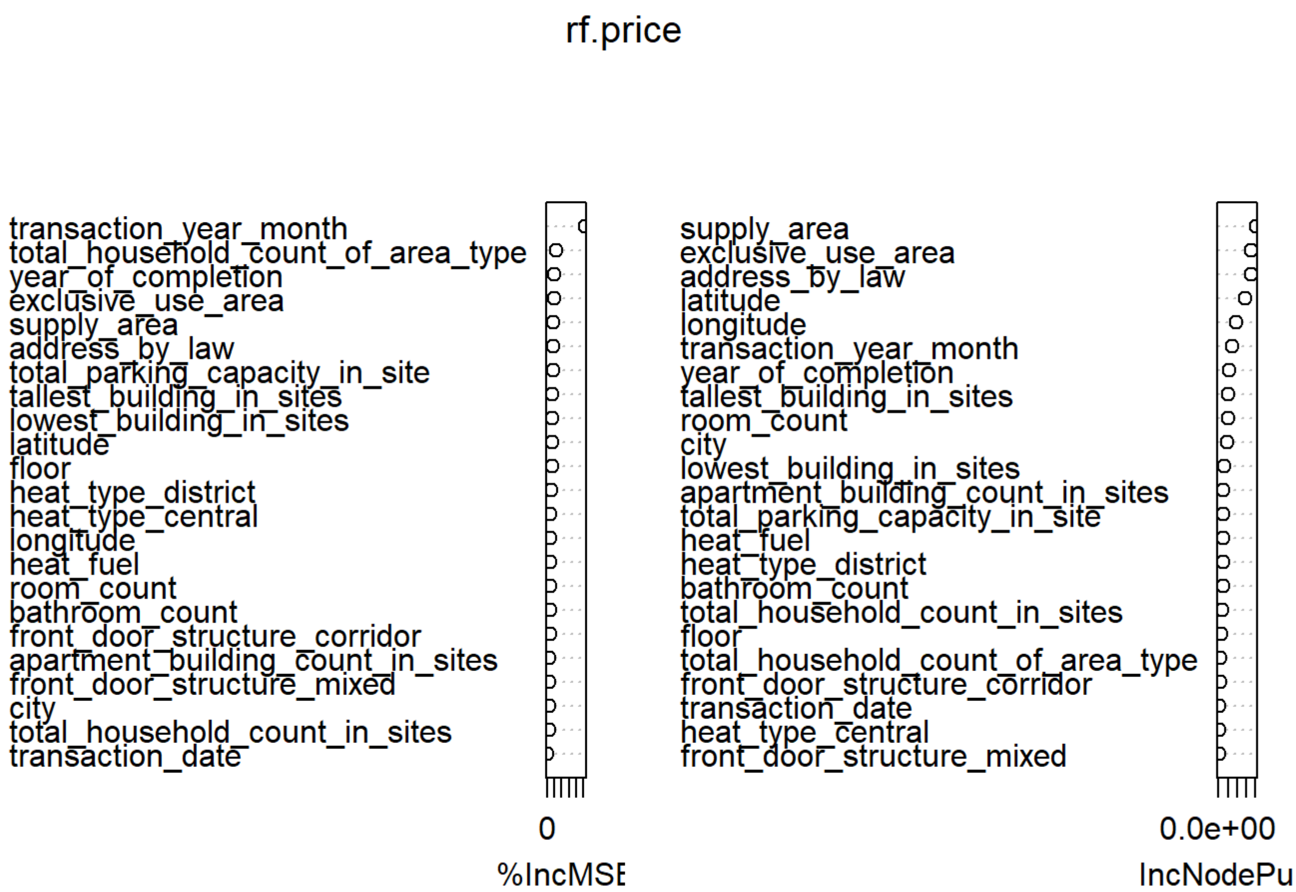
```
##  RMSE test:  54993204
```

```
importance(rf.price)
```

```
##                                       %IncMSE IncNodePurity
## city                                13.296800  4.672313e+20
## transaction_year_month             261.082473  7.331360e+20
## transaction_date                     2.130253  2.112215e+19
## year_of_completion                  51.249919  5.665932e+20
## exclusive_use_area                  47.950984  1.794052e+21
## floor                               33.677676  1.163679e+20
## latitude                            34.459616  1.452699e+21
## longitude                           21.894668  9.760793e+20
## address_by_law                      43.654317  1.771232e+21
## total_parking_capacity_in_site      40.089590  2.738914e+20
## total_household_count_in_sites      12.150153  2.007782e+20
## apartment_building_count_in_sites   16.257821  2.741756e+20
## tallest_building_in_sites           35.551012  5.369457e+20
## lowest_building_in_sites            34.851280  3.135553e+20
## heat_fuel                           21.851396  2.737434e+20
## supply_area                         43.904885  2.026513e+21
## total_household_count_of_area_type  62.743647  1.100054e+20
## room_count                          21.126094  4.995193e+20
## bathroom_count                      19.225478  2.360294e+20
## heat_type_central                   22.830232  1.338015e+19
## heat_type_district                  27.507456  2.674862e+20
## front_door_structure_corridor       18.231974  7.695837e+19
## front_door_structure_mixed          13.825165  8.157309e+18
```

The visualization below reveals that supply area is relatively more important compared to the other predictors. Exclusive use area is also an important predictor.

```
varImpPlot(rf.price)
```



Boosting

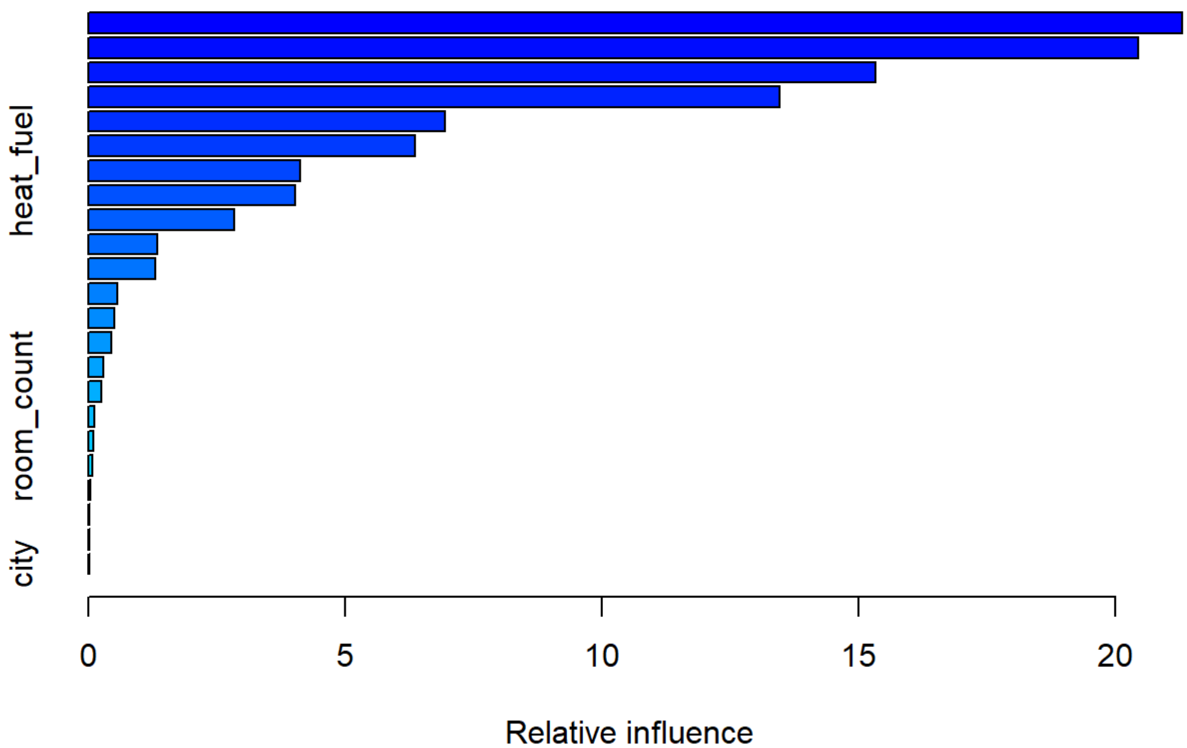
the boosting process reveals similar finding to what we saw with random forest above. Again, we can see that supply area, and exclusive use area are both relatively more important compared to the other predictors.

select a random sample of 10000 observations

```
# get a random sample of row numbers from the train dataset
train_sample_index = sample(1:nrow(dd_train),nrow(dd_train)*4/5)
# use those rows to get the actual data from the train data
train_sample = dd_train[train_sample_index,]
# get the price column from the train dataset
tree.testy_sample <- dd_test[train_sample_index, transaction_real_price]
```

In the below model, we started with just 5000 trees. We then go on to perform feature engineering, and use cross validation to find the optimal number of trees.

```
boost.price = gbm(transaction_real_price ~ ., data = train_sample, distribution = "gaussian", n.
trees = 5000, interaction.depth = 4 )
summary(boost.price)
```


heat_fuel
room_count
city
0
5
10
15
20
Relative influence

```
##                                                                      var
## address_by_law                                            address_by_law
## supply_area                                                  supply_area
## exclusive_use_area                                    exclusive_use_area
## latitude                                                        latitude
## transaction_year_month                            transaction_year_month
## longitude                                                      longitude
## heat_fuel                                                      heat_fuel
## year_of_completion                                    year_of_completion
## tallest_building_in_sites                      tallest_building_in_sites
## apartment_building_count_in_sites   apartment_building_count_in_sites
## total_parking_capacity_in_site         total_parking_capacity_in_site
## lowest_building_in_sites                        lowest_building_in_sites
## total_household_count_in_sites         total_household_count_in_sites
## floor                                                              floor
## heat_type_district                                    heat_type_district
## total_household_count_of_area_type total_household_count_of_area_type
## room_count                                                    room_count
## front_door_structure_corridor           front_door_structure_corridor
## bathroom_count                                            bathroom_count
## heat_type_central                                      heat_type_central
## transaction_date                                        transaction_date
## front_door_structure_mixed                 front_door_structure_mixed
## city                                                                city
##                                        rel.inf
## address_by_law                     21.30022955
## supply_area                        20.44244313
## exclusive_use_area                 15.33593540
## latitude                           13.47260822
## transaction_year_month              6.94010123
## longitude                           6.36395481
## heat_fuel                           4.12824315
## year_of_completion                  4.03493562
## tallest_building_in_sites           2.83937368
## apartment_building_count_in_sites   1.33846569
## total_parking_capacity_in_site      1.31318606
## lowest_building_in_sites            0.56585616
## total_household_count_in_sites      0.50259046
## floor                               0.44629422
## heat_type_district                  0.30050003
## total_household_count_of_area_type  0.25592111
## room_count                          0.12440337
## front_door_structure_corridor       0.09784408
## bathroom_count                      0.08454600
## heat_type_central                   0.04972401
## transaction_date                    0.03084237
## front_door_structure_mixed          0.01854806
## city                                0.01345361
```

The results of the boosting model reveal the following RMSEs in Korean Won. This model produced the lowest error of any of the models in the report.

```
yhat.boost.train = predict(boost.price, newdata = dd_train, n.trees = 5000)
cat("RMSE train: ", sqrt(mean((dd_train$transaction_real_price - yhat.boost.train)^2)))
```

```
## RMSE train:  44452017
```

```
#Evaluate boosted tree model
yhat.boost.test = predict(boost.price, newdata = dd_test, n.trees = 5000)
cat(" RMSE test: ", sqrt(mean((dd_test$transaction_real_price - yhat.boost.test)^2)))
```

```
##  RMSE test:  52281154
```

Feature Engineering

```
dd_train_engineering <- copy(dd_train)
dd_test_engineering <- copy(dd_test)

# feature engineering for train
# perform feature engineering to put the latitude and longitude in 3d space
dd_train_engineering[, x := cos(latitude) * cos(longitude)]
dd_train_engineering[, y := cos(latitude) * sin(longitude)]
dd_train_engineering[, z := sin(latitude)]

dd_train_engineering[, living_area := supply_area - exclusive_use_area]
dd_train_engineering[, bathroom_per_living_area := bathroom_count/living_area]
dd_train_engineering[, area_ratio := exclusive_use_area / supply_area]
dd_train_engineering[, household_ratio := total_household_count_of_area_type / total_household_c
ount_in_sites]
dd_train_engineering[, total_household_per_building_count := total_household_count_in_sites / ap
artment_building_count_in_sites]
dd_train_engineering[, age_of_apartment := 2021 - year_of_completion]

dd_train_engineering[, year := as.numeric(substr(transaction_year_month, 1,4))]
dd_train_engineering[, month := as.numeric(substr(transaction_year_month, 5,6))]

# feature engineering for test
dd_test_engineering[, x := cos(latitude) * cos(longitude)]
dd_test_engineering[, y := cos(latitude) * sin(longitude)]
dd_test_engineering[, z := sin(latitude)]

dd_test_engineering[, living_area := supply_area - exclusive_use_area]
dd_test_engineering[, bathroom_per_living_area := bathroom_count/living_area]
dd_test_engineering[, area_ratio := exclusive_use_area / supply_area]
dd_test_engineering[, household_ratio := total_household_count_of_area_type / total_household_co
unt_in_sites]
dd_test_engineering[, total_household_per_building_count := total_household_count_in_sites / apa
rtment_building_count_in_sites]
dd_test_engineering[, age_of_apartment := 2021 - year_of_completion]

dd_test_engineering[, year := as.numeric(substr(transaction_year_month, 1,4))]
dd_test_engineering[, month := as.numeric(substr(transaction_year_month, 5,6))]
```

Boosting: cross validation without feature engineering

```
boost.price.cv <- gbm(transaction_real_price ~ ., data = train_sample, distribution = "gaussian"
, n.trees = 5000, interaction.depth = 4, cv.folds=10)

# predict on train dataset
yhat.boost.cv.train = predict(boost.price.cv, newdata = dd_train, n.trees = which.min(boost.pric
e.cv$cv.error))
cat("Train RMSE: ", sqrt(mean((dd_train$transaction_real_price - yhat.boost.cv.train)^2)))
```

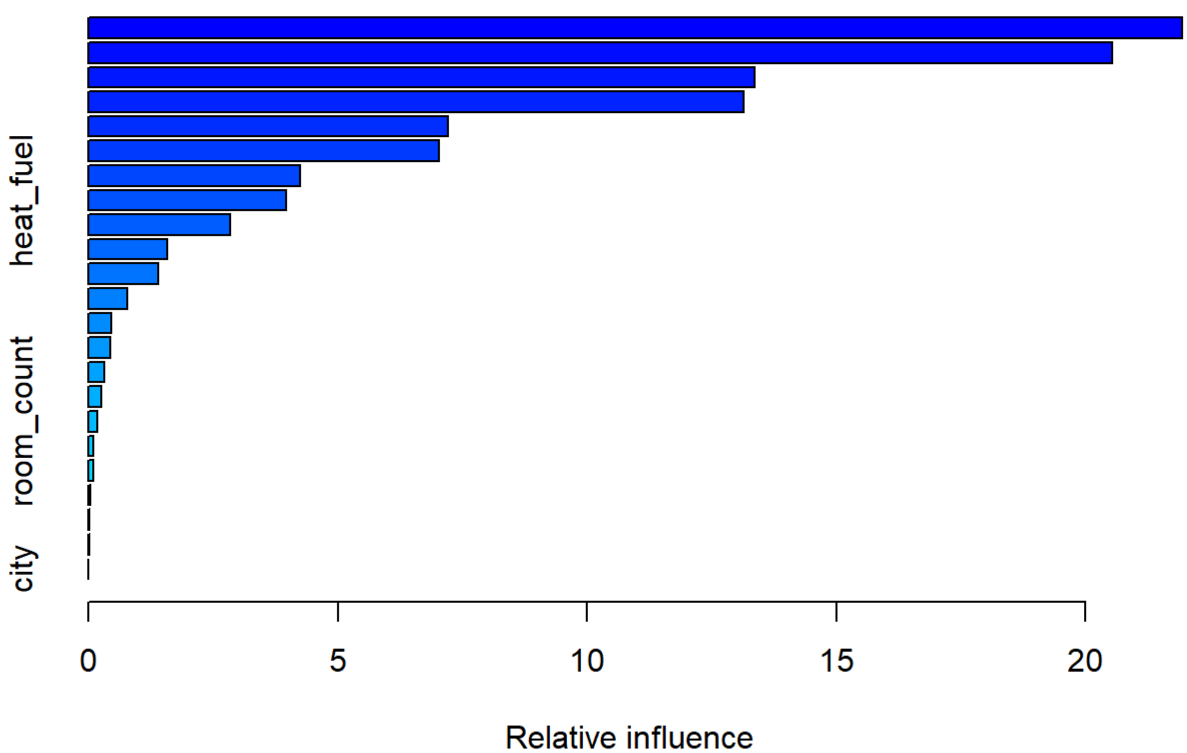
```
## Train RMSE:  44130971
```

```
# predict on the test dataset using the recommended number of trees
predictions <- predict(boost.price.cv, newdata=dd_train, n.trees=which.min(boost.price.cv$cv.err
or))
cat(" Test RMSE: ", sqrt(mean(dd_train$transaction_real_price - predictions)**2))
```

```
##  Test RMSE:  45081.09
```

The most significant features according to the model.

```
summary(boost.price.cv)
```



```
##                                                                       var
## supply_area                                                   supply_area
## address_by_law                                             address_by_law
## exclusive_use_area                                     exclusive_use_area
## latitude                                                         latitude
## longitude                                                       longitude
## transaction_year_month                             transaction_year_month
## year_of_completion                                     year_of_completion
## heat_fuel                                                       heat_fuel
## tallest_building_in_sites                       tallest_building_in_sites
## apartment_building_count_in_sites   apartment_building_count_in_sites
## total_parking_capacity_in_site         total_parking_capacity_in_site
## lowest_building_in_sites                         lowest_building_in_sites
## total_household_count_in_sites         total_household_count_in_sites
## floor                                                               floor
## heat_type_district                                     heat_type_district
## total_household_count_of_area_type total_household_count_of_area_type
## room_count                                                     room_count
## front_door_structure_corridor           front_door_structure_corridor
## bathroom_count                                             bathroom_count
## transaction_date                                         transaction_date
## front_door_structure_mixed                 front_door_structure_mixed
## heat_type_central                                       heat_type_central
## city                                                                 city
##                                         rel.inf
## supply_area                         21.932027794
## address_by_law                      20.524205923
## exclusive_use_area                  13.364367188
## latitude                            13.146690161
## longitude                            7.207243203
## transaction_year_month               7.040710045
## year_of_completion                   4.247379210
## heat_fuel                            3.966227518
## tallest_building_in_sites            2.850453637
## apartment_building_count_in_sites    1.583823587
## total_parking_capacity_in_site       1.403894078
## lowest_building_in_sites             0.786461733
## total_household_count_in_sites       0.457996352
## floor                                0.434545434
## heat_type_district                   0.313210265
## total_household_count_of_area_type   0.267675320
## room_count                           0.176855572
## front_door_structure_corridor        0.103106493
## bathroom_count                       0.102814494
## transaction_date                     0.033487882
## front_door_structure_mixed           0.027144779
## heat_type_central                    0.026619081
## city                                 0.003060251
```

select a random sample of 10000 observations

```
train_sample_index = sample(1:nrow(dd_train_engineering),nrow(dd_train_engineering)*4/5)
train_sample = dd_train_engineering[train_sample_index,]
tree.testy_sample <- dd_test_engineering[train_sample_index, transaction_real_price]
```

Boosting: cross validation with feature engineering

```
boost.price.cv <- gbm(transaction_real_price ~ ., data = train_sample, distribution = "gaussian"
, n.trees = 5000, interaction.depth = 4, cv.folds=10)

yhat.boost.train = predict(boost.price.cv, newdata = dd_train_engineering, n.trees = which.min(b
oost.price.cv$cv.error))
cat("RMSE train: ", sqrt(mean((dd_train_engineering$transaction_real_price - yhat.boost.train)^2
)))
```

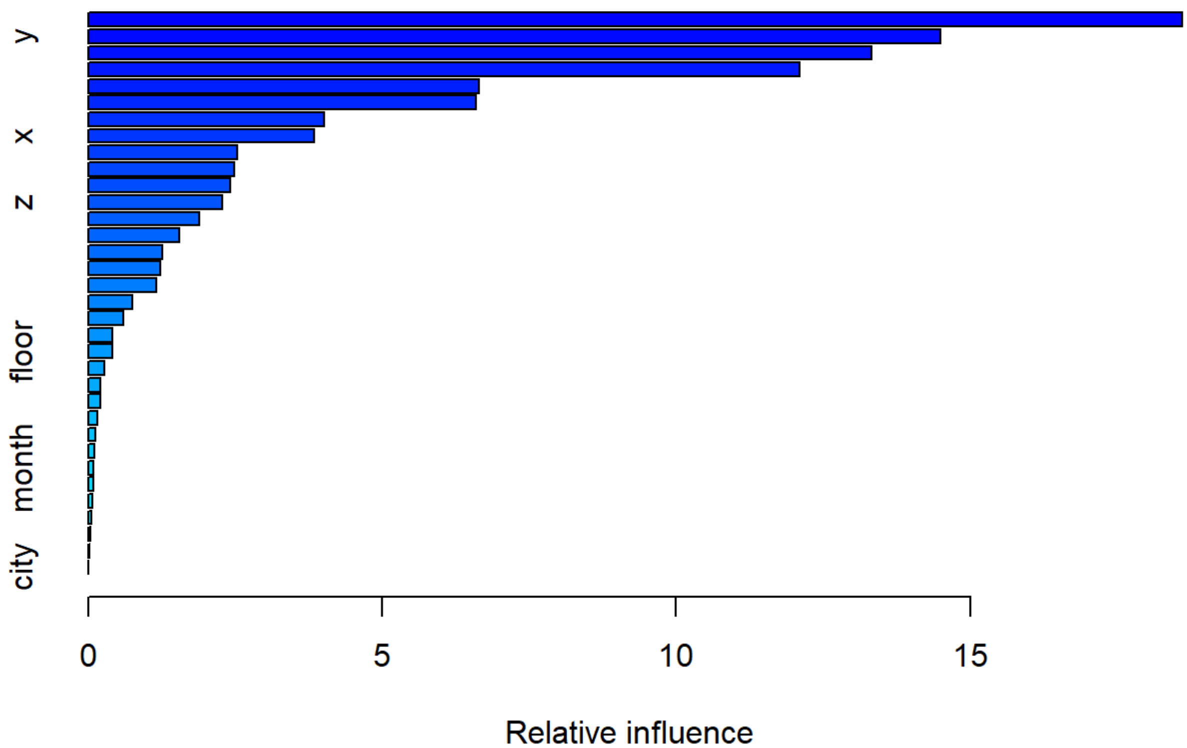
```
## RMSE train:  42532081
```

```
# predict on the test dataset using the recommended number of trees
predictions <- predict(boost.price.cv, newdata = dd_test_engineering, n.trees = which.min(boost.
price.cv$cv.error))
cat(" RMSE test: ", sqrt(mean((dd_test_engineering$transaction_real_price - predictions)**2)))
```

```
##  RMSE test:  49999316
```

The most significant features according to the model.

```
summary(boost.price.cv)
```


y
x
z
floor
month
city
0
5
10
15
Relative influence

```
##                                                                      var
## supply_area                                                  supply_area
## y                                                                      y
## exclusive_use_area                                    exclusive_use_area
## address_by_law                                            address_by_law
## latitude                                                        latitude
## transaction_year_month                            transaction_year_month
## heat_fuel                                                      heat_fuel
## x                                                                      x
## longitude                                                      longitude
## year_of_completion                                    year_of_completion
## tallest_building_in_sites                      tallest_building_in_sites
## z                                                                      z
## living_area                                                  living_area
## total_household_per_building_count total_household_per_building_count
## apartment_building_count_in_sites   apartment_building_count_in_sites
## total_parking_capacity_in_site        total_parking_capacity_in_site
## age_of_apartment                                        age_of_apartment
## heat_type_district                                    heat_type_district
## lowest_building_in_sites                        lowest_building_in_sites
## area_ratio                                                    area_ratio
## floor                                                              floor
## total_household_count_in_sites        total_household_count_in_sites
## year                                                                year
## bathroom_per_living_area                        bathroom_per_living_area
## household_ratio                                          household_ratio
## total_household_count_of_area_type total_household_count_of_area_type
## room_count                                                    room_count
## month                                                              month
## bathroom_count                                            bathroom_count
## front_door_structure_corridor          front_door_structure_corridor
## heat_type_central                                      heat_type_central
## transaction_date                                        transaction_date
## front_door_structure_mixed                front_door_structure_mixed
## city                                                                city
##                                         rel.inf
## supply_area                        18.608857749
## y                                  14.494912540
## exclusive_use_area                 13.326192767
## address_by_law                     12.105804529
## latitude                            6.648128023
## transaction_year_month              6.596668873
## heat_fuel                           4.006117385
## x                                   3.835406950
## longitude                           2.538490247
## year_of_completion                  2.491341887
## tallest_building_in_sites           2.413669663
## z                                   2.272997050
## living_area                         1.888397151
## total_household_per_building_count  1.556343773
## apartment_building_count_in_sites   1.260717839
## total_parking_capacity_in_site      1.227589358
## age_of_apartment                    1.158348716
```

```
## heat_type_district                0.758079011
## lowest_building_in_sites          0.590870920
## area_ratio                        0.406922451
## floor                             0.405400016
## total_household_count_in_sites    0.269270636
## year                              0.212812289
## bathroom_per_living_area          0.202415716
## household_ratio                   0.154866778
## total_household_count_of_area_type  0.122390273
## room_count                        0.098498882
## month                             0.085613219
## bathroom_count                    0.083490173
## front_door_structure_corridor     0.072583439
## heat_type_central                 0.049241930
## transaction_date                  0.034340784
## front_door_structure_mixed        0.021255344
## city                              0.001963639
```