HDB Price Analysis

2024-04-30

```
#Importing Modules
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.3.3
```

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.3.2
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(fitdistrplus)
```

```
## Warning: package 'fitdistrplus' was built under R version 4.3.3
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select

## Loading required package: survival
```

```
library(olsrr)
```

```
## Warning: package 'olsrr' was built under R version 4.3.3

##
## Attaching package: 'olsrr'

## The following object is masked from 'package:MASS':
##
##     cement

## The following object is masked from 'package:datasets':
##
##     rivers
```

```
library(leaps)
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.3.3

##
## Attaching package: 'boot'

## The following object is masked from 'package:survival':
##
##     aml
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.3

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

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

## Loaded glmnet 4.1-8
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.3.3
```

```
library(ISLR)
library(zoo)
```

```
## Warning: package 'zoo' was built under R version 4.3.2
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.3.3
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 4.3.3
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:boot':
##
##     logit
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.3.3
```

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```r
library(dgof)
```

```
##
## Attaching package: 'dgof'
```

```
## The following object is masked from 'package:stats':
##
##     ks.test
```

```r
library(htmltools)
```

```
## Warning: package 'htmltools' was built under R version 4.3.3
```

```r
library(tinytex)
```

```
## Warning: package 'tinytex' was built under R version 4.3.3
```

```r
setwd("C:/Users/keith/OneDrive/Desktop/Github Projects/HDB Price Analysis")

#Data Pre-Processing
data_h = read.csv("HDB_Hougang_Final.csv", stringsAsFactors = TRUE)
data_s = read.csv("HDB_Serangoon_Final.csv", stringsAsFactors = TRUE)

#Data Wrangling
final_data = rbind(data_h, data_s)
head(final_data)
```

```
##     month    town flat_type storey_range floor_area_sqm     flat_model
## 1 2017-01 HOUGANG    2 ROOM     10 TO 12             47        Model A
## 2 2017-01 HOUGANG    3 ROOM     01 TO 03             73 New Generation
## 3 2017-01 HOUGANG    3 ROOM     01 TO 03             60       Improved
## 4 2017-01 HOUGANG    3 ROOM     01 TO 03             67 New Generation
## 5 2017-01 HOUGANG    3 ROOM     01 TO 03             67 New Generation
```

```
## 6 2017-01 HOUGANG    3 ROOM     01 TO 03             60       Improved
##   lease_commence_date    remaining_lease resale_price       full_address
## 1                2012 94 years 04 months       252000 986A BUANGKOK CRES
## 2                1978 60 years 04 months       260000   22 HOUGANG AVE 3
## 3                1987           69 years       268000  644 HOUGANG AVE 8
## 4                1978 60 years 05 months       269000   24 HOUGANG AVE 3
## 5                1984 66 years 03 months       270000  108 HOUGANG AVE 1
## 6                1985 67 years 09 months       275000  619 HOUGANG AVE 8
##   X_coordinates Y_coordinates  Closest_Mrt Mrt_Distance Price.Index
## 1      1.381917      103.8798 Buangkok mrt    1.4871838       133.9
## 2      1.364285      103.8905  Hougang mrt    0.6980562       133.9
## 3      1.371437      103.8813  Hougang mrt    1.2302958       133.9
## 4      1.365512      103.8921  Hougang mrt    0.5336041       133.9
## 5      1.352774      103.8901    Kovan mrt    1.0046931       133.9
## 6      1.369848      103.8837  Hougang mrt    0.9542071       133.9
```

```
final_data = mutate(final_data, ID = row_number())
final_data = final_data %>% mutate(month = ym(as.character(month)))
final_data = final_data %>% mutate(remaining_lease = as.integer(substr(final_data[,"remaining_lease"], s
final_data["Real_Value"] = (final_data[,"resale_price"]/final_data[,"Price.Index"])*100

#Removing redundant data
final_data = final_data %>% dplyr::select(month, ID, town, flat_type, storey_range, floor_area_sqm, fla
head(final_data)
```

```
##        month ID    town flat_type storey_range floor_area_sqm     flat_model
## 1 2017-01-01  1 HOUGANG    2 ROOM     10 TO 12             47        Model A
## 2 2017-01-01  2 HOUGANG    3 ROOM     01 TO 03             73 New Generation
## 3 2017-01-01  3 HOUGANG    3 ROOM     01 TO 03             60       Improved
## 4 2017-01-01  4 HOUGANG    3 ROOM     01 TO 03             67 New Generation
## 5 2017-01-01  5 HOUGANG    3 ROOM     01 TO 03             67 New Generation
## 6 2017-01-01  6 HOUGANG    3 ROOM     01 TO 03             60       Improved
##   lease_commence_date remaining_lease Mrt_Distance Real_Value
## 1                2012              94    1.4871838   188200.1
## 2                1978              60    0.6980562   194174.8
## 3                1987              69    1.2302958   200149.4
## 4                1978              60    0.5336041   200896.2
## 5                1984              66    1.0046931   201643.0
## 6                1985              67    0.9542071   205377.1
```

```
#[Look specific into storey_range & flat_model might have lack of data for specific points]

final_data %>% group_by(flat_model, town) %>% count() #[Remove adjoined flat due to low count + Remove
```

```
## # A tibble: 21 x 3
## # Groups:   flat_model, town [21]
##    flat_model          town          n
##    <fct>               <fct>     <int>
##  1 3Gen                HOUGANG       2
##  2 Adjoined flat       HOUGANG       7
##  3 Adjoined flat       SERANGOON     7
##  4 Apartment           HOUGANG     312
##  5 Apartment           SERANGOON   176
```

```
##  6 DBSS                HOUGANG     210
##  7 Improved            HOUGANG    1908
##  8 Improved            SERANGOON   888
##  9 Improved-Maisonette HOUGANG      25
## 10 Maisonette          HOUGANG     632
## # i 11 more rows
```

```
categories = c("Apartment", "Improved", "Maisonette", "Model A", "New Generation", "Simplified")
final_data = subset(final_data, flat_model %in% categories)

final_data %>% group_by(storey_range, town) %>% count() #[Remove 19 TO 21 due to lack of data points]
```

```
## # A tibble: 13 x 3
## # Groups:   storey_range, town [13]
##    storey_range town          n
##    <fct>        <fct>     <int>
##  1 01 TO 03     HOUGANG    1679
##  2 01 TO 03     SERANGOON   755
##  3 04 TO 06     HOUGANG    2023
##  4 04 TO 06     SERANGOON   923
##  5 07 TO 09     HOUGANG    1751
##  6 07 TO 09     SERANGOON   707
##  7 10 TO 12     HOUGANG    1564
##  8 10 TO 12     SERANGOON   645
##  9 13 TO 15     HOUGANG     762
## 10 13 TO 15     SERANGOON   169
## 11 16 TO 18     HOUGANG     272
## 12 16 TO 18     SERANGOON    13
## 13 19 TO 21     HOUGANG      19
```

```
final_data = subset(final_data, !(storey_range %in% "19 TO 21"))
head(final_data)
```

```
##        month ID    town flat_type storey_range floor_area_sqm     flat_model
## 1 2017-01-01  1 HOUGANG    2 ROOM     10 TO 12             47        Model A
## 2 2017-01-01  2 HOUGANG    3 ROOM     01 TO 03             73 New Generation
## 3 2017-01-01  3 HOUGANG    3 ROOM     01 TO 03             60       Improved
## 4 2017-01-01  4 HOUGANG    3 ROOM     01 TO 03             67 New Generation
## 5 2017-01-01  5 HOUGANG    3 ROOM     01 TO 03             67 New Generation
## 6 2017-01-01  6 HOUGANG    3 ROOM     01 TO 03             60       Improved
##   lease_commence_date remaining_lease Mrt_Distance Real_Value
## 1                2012              94    1.4871838   188200.1
## 2                1978              60    0.6980562   194174.8
## 3                1987              69    1.2302958   200149.4
## 4                1978              60    0.5336041   200896.2
## 5                1984              66    1.0046931   201643.0
## 6                1985              67    0.9542071   205377.1
```

```
summary(final_data)
```

```
##      month                  ID               town          flat_type
##  Min.   :2017-01-01   Min.   :    1   HOUGANG  :8051   2 ROOM   : 109
```

```
##  1st Qu.:2018-12-01   1st Qu.: 2936   SERANGOON:3212   3 ROOM   :2467
##  Median :2020-12-01   Median : 6210                    4 ROOM   :4971
##  Mean   :2020-10-17   Mean   : 6121                    5 ROOM   :2271
##  3rd Qu.:2022-08-01   3rd Qu.: 9250                    EXECUTIVE:1445
##  Max.   :2024-03-01   Max.   :12070
##
##    storey_range  floor_area_sqm           flat_model   lease_commence_date
##  01 TO 03:2434   Min.   : 38.0   Model A       :3861   Min.   :1975
##  04 TO 06:2946   1st Qu.: 84.0   Improved      :2795   1st Qu.:1985
##  07 TO 09:2458   Median :101.0   New Generation:1943   Median :1988
##  10 TO 12:2209   Mean   :100.8   Simplified    :1219   Mean   :1992
##  13 TO 15: 931   3rd Qu.:120.0   Maisonette    : 957   3rd Qu.:1997
##  16 TO 18: 285   Max.   :177.0   Apartment     : 488   Max.   :2019
##  19 TO 21:   0                   (Other)       :   0
##  remaining_lease  Mrt_Distance       Real_Value
##  Min.   :50.00   Min.   :0.09927   Min.   :129081
##  1st Qu.:63.00   1st Qu.:0.57940   1st Qu.:258410
##  Median :67.00   Median :0.82887   Median :312145
##  Mean   :70.33   Mean   :0.95934   Mean   :333148
##  3rd Qu.:75.00   3rd Qu.:1.30449   3rd Qu.:386675
##  Max.   :95.00   Max.   :2.29621   Max.   :759878
##
```

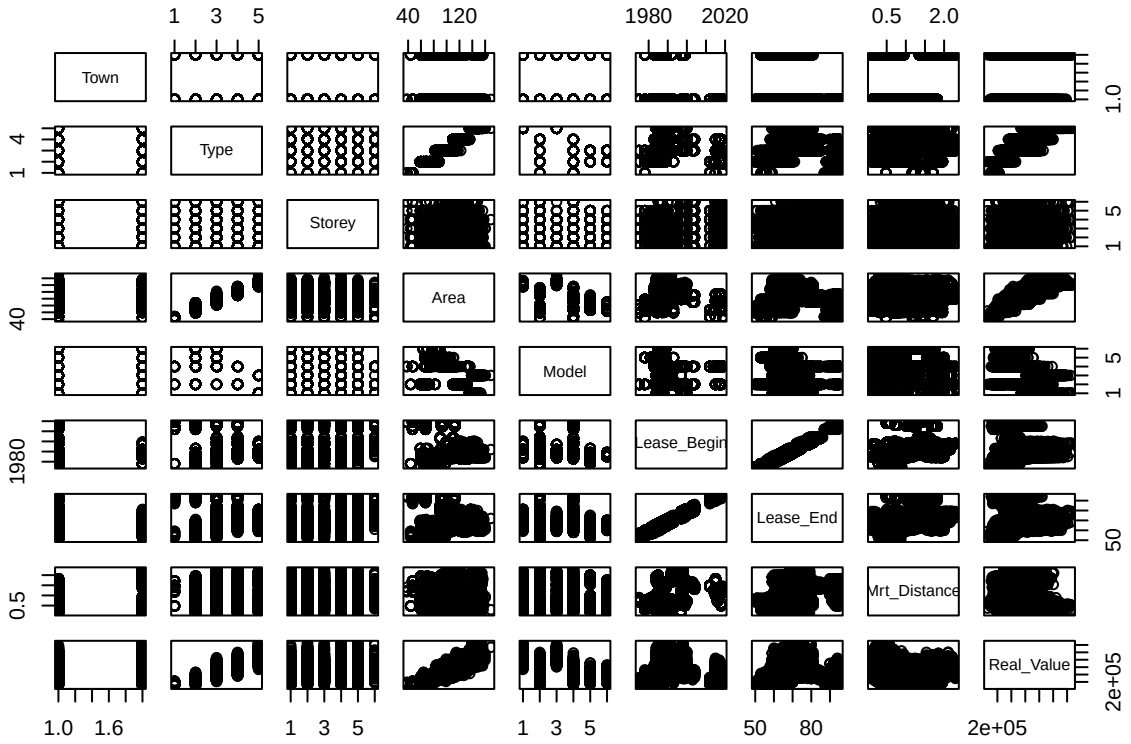
```
#Removing Redundant Factors
final_data[, "storey_range"] = factor(final_data[, "storey_range"])
final_data[, "flat_model"] = factor(final_data[, "flat_model"])

#Renaming Columns
final_data = final_data %>% dplyr::rename(Town = town, Type = flat_type, Storey = storey_range, Area = f

#Sectioning out Final Test Data
final_train = final_data %>% group_by(Town) %>% sample_frac(0.9, replace = FALSE)
final_test = anti_join(final_data, final_train)
```

```
## Joining with `by = join_by(month, ID, Town, Type, Storey, Area, Model,
## Lease_Begin, Lease_End, Mrt_Distance, Real_Value)`
```

```
#Sectioning out Validation Set
train = final_train %>% group_by(Town) %>% sample_frac(0.8, replace = FALSE)
test = anti_join(final_train, train)
```

```
## Joining with `by = join_by(month, ID, Town, Type, Storey, Area, Model,
## Lease_Begin, Lease_End, Mrt_Distance, Real_Value)`
```

```
#Remove months and ID
train = subset(train, select = -c(month, ID))
test = subset(test, select = -c(month, ID))

#General plot of all variables
pairs(train)
```




```
#Testing for Multicollinearity
#Area vs Type
summary(lm(Area~Type, data = train)) #[R^2 suggest high collinearity]
```

```
##
## Call:
## lm(formula = Area ~ Type, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.6138  -3.8616   0.1384   4.3862  25.3862
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     46.0128     0.7189   64.00   <2e-16 ***
## Type3 ROOM      21.6066     0.7343   29.43   <2e-16 ***
## Type4 ROOM      50.6009     0.7267   69.63   <2e-16 ***
## Type5 ROOM      73.8487     0.7361  100.33   <2e-16 ***
## TypeEXECUTIVE 100.3231     0.7461  134.47   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.35 on 8105 degrees of freedom
## Multiple R-squared:  0.9384, Adjusted R-squared:  0.9384
## F-statistic: 3.088e+04 on 4 and 8105 DF,  p-value: < 2.2e-16
```

```r
summary(lm(Real_Value ~ Type, data = train))
```

```
##
## Call:
## lm(formula = Real_Value ~ Type, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -151087  -31634   -8777   22309  216209
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     181185       5634  32.156  < 2e-16 ***
## Type3 ROOM       42638       5755   7.409  1.4e-13 ***
## Type4 ROOM      130510       5695  22.915  < 2e-16 ***
## Type5 ROOM      205633       5769  35.646  < 2e-16 ***
## TypeEXECUTIVE   339101       5847  57.996  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 49760 on 8105 degrees of freedom
## Multiple R-squared:  0.7648, Adjusted R-squared:  0.7647
## F-statistic:  6588 on 4 and 8105 DF,  p-value: < 2.2e-16
```

```r
summary(lm(Real_Value ~ Area, data = train))
```

```
##
## Call:
## lm(formula = Real_Value ~ Area, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -141346  -35951  -10063   25163  263722
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -14833.41    2355.33  -6.298 3.18e-10 ***
## Area          3449.68      22.71 151.922  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 52310 on 8108 degrees of freedom
## Multiple R-squared:   0.74,  Adjusted R-squared:   0.74
## F-statistic: 2.308e+04 on 1 and 8108 DF,  p-value: < 2.2e-16
```

```r
#[Both R^2 values are close, hence can test both for best model]

#Lease_Begin vs Lease_End
summary(lm(Lease_Begin ~ Lease_End, data = train)) #[R^2 suggest high collinearity]
```

```
##
```

```
## Call:
## lm(formula = Lease_Begin ~ Lease_End, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8276 -1.7544  0.2293  1.4814  4.4570
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.922e+03  1.619e-01 11875.6   <2e-16 ***
## Lease_End   9.919e-01  2.279e-03   435.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.112 on 8108 degrees of freedom
## Multiple R-squared:  0.959,  Adjusted R-squared:  0.959
## F-statistic: 1.894e+05 on 1 and 8108 DF,  p-value: < 2.2e-16
```

```
summary(lm(Real_Value ~ Lease_Begin, data = train))
```

```
##
## Call:
## lm(formula = Real_Value ~ Lease_Begin, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -212032  -69822  -26424   48021  395279
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2846493.9   214807.0  -13.25   <2e-16 ***
## Lease_Begin     1595.6      107.8   14.80   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 101200 on 8108 degrees of freedom
## Multiple R-squared:  0.02629,    Adjusted R-squared:  0.02617
## F-statistic: 218.9 on 1 and 8108 DF,  p-value: < 2.2e-16
```

```
summary(lm(Real_Value ~ Lease_End, data = train))
```

```
##
## Call:
## lm(formula = Real_Value ~ Lease_End, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -216939  -69839  -26234   48500  394492
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   208665       7739   26.96   <2e-16 ***
## Lease_End       1754        109   16.10   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 101000 on 8108 degrees of freedom
## Multiple R-squared:  0.03097,    Adjusted R-squared:  0.03085
## F-statistic: 259.2 on 1 and 8108 DF,  p-value: < 2.2e-16
```

```
#[Both R^2 values are close, hence can test both for best model]

#Area vs Model
summary(lm(Area~Model, data = train)) #[R^2 suggest ambiguous collinearity]
```

```
##
## Call:
## lm(formula = Area ~ Model, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -63.126  -4.173   4.972   9.512  39.827
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          142.0281     0.8873 160.071  < 2e-16 ***
## ModelImproved        -34.9017     0.9559 -36.513  < 2e-16 ***
## ModelMaisonette        6.2913     1.0723   5.867  4.6e-09 ***
## ModelModel A         -45.8552     0.9367 -48.952  < 2e-16 ***
## ModelNew Generation  -60.5398     0.9822 -61.634  < 2e-16 ***
## ModelSimplified      -64.5281     1.0346 -62.372  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.87 on 8104 degrees of freedom
## Multiple R-squared:  0.6152, Adjusted R-squared:  0.615
## F-statistic:  2592 on 5 and 8104 DF,  p-value: < 2.2e-16
```

```
ols_vif_tol(lm(Real_Value ~ Area + Model, data = train)) #[Slight Correlation]
```

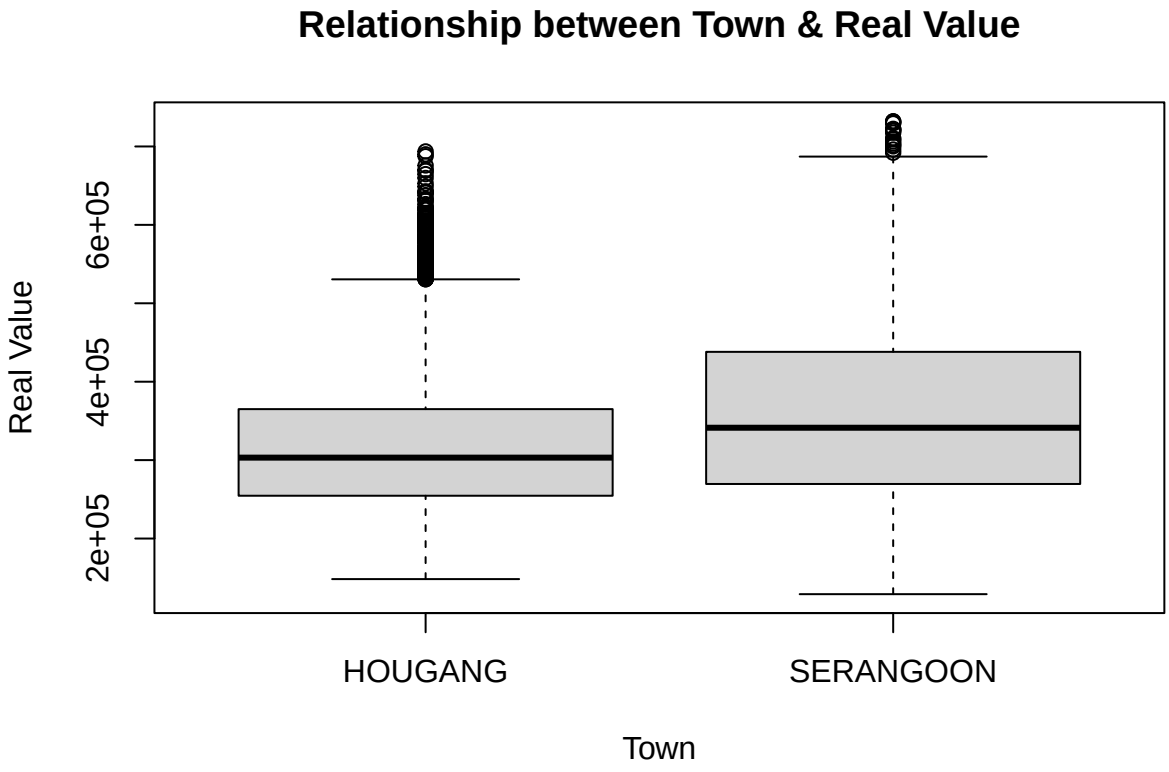
```
##             Variables Tolerance      VIF
## 1                Area 0.3847586 2.599032
## 2       ModelImproved 0.1575116 6.348738
## 3     ModelMaisonette 0.3433624 2.912375
## 4        ModelModel A 0.1210103 8.263758
## 5 ModelNew Generation 0.1518564 6.585167
## 6     ModelSimplified 0.2007127 4.982246
```

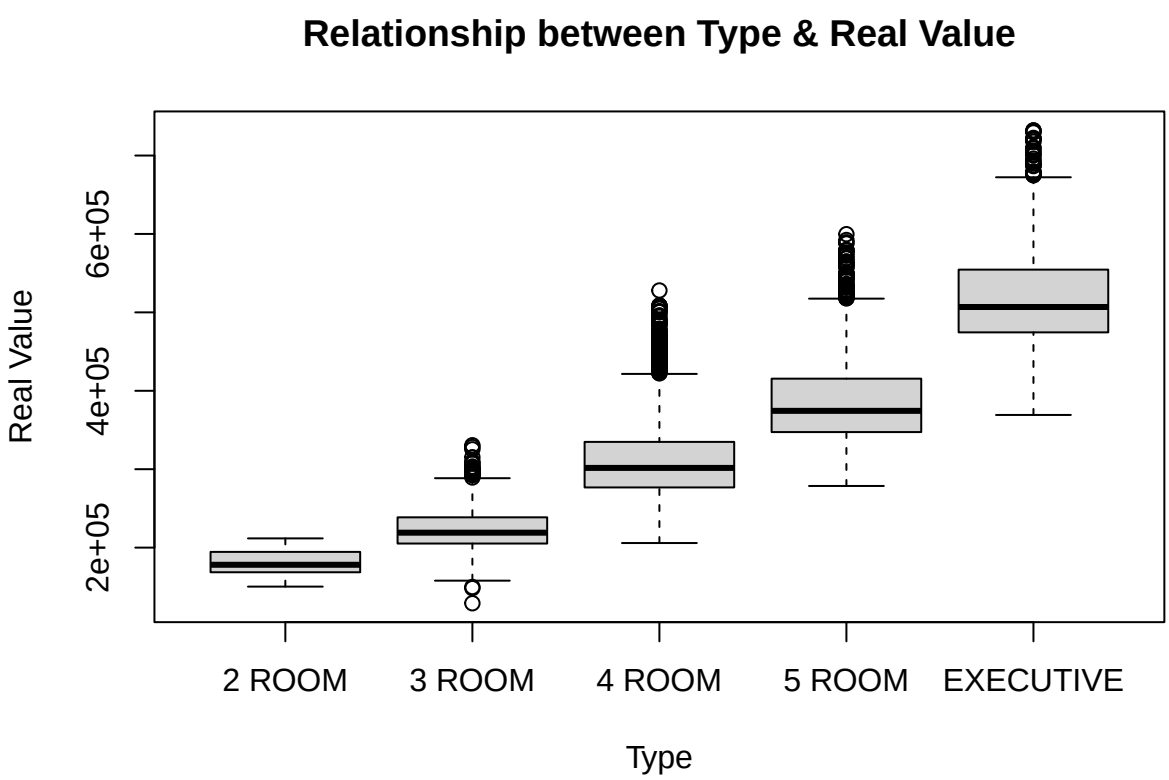
```
#Type vs Model
type_model = lm(Real_Value ~ Type + Model, data = train)
ols_vif_tol(type_model) #[High vif value indicates collinearity issue]
```

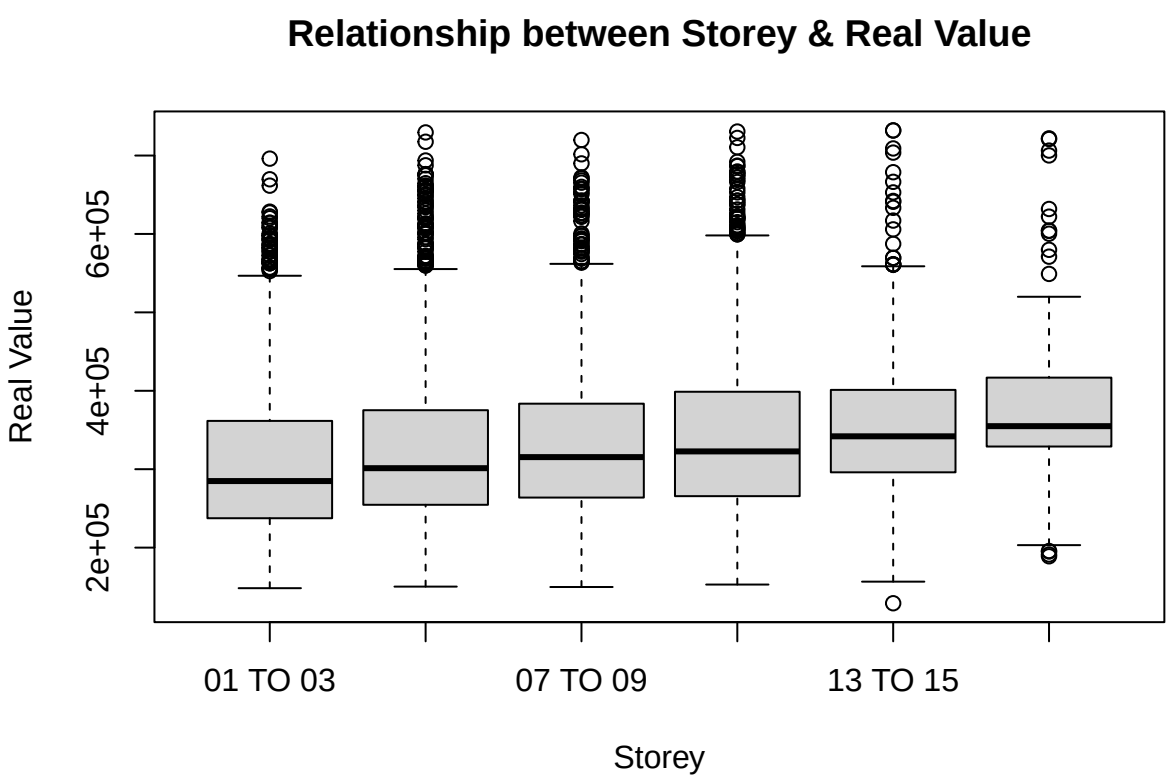
```
##             Variables  Tolerance       VIF
## 1          Type3 ROOM 0.05193766 19.253853
## 2          Type4 ROOM 0.03776448 26.479909
```

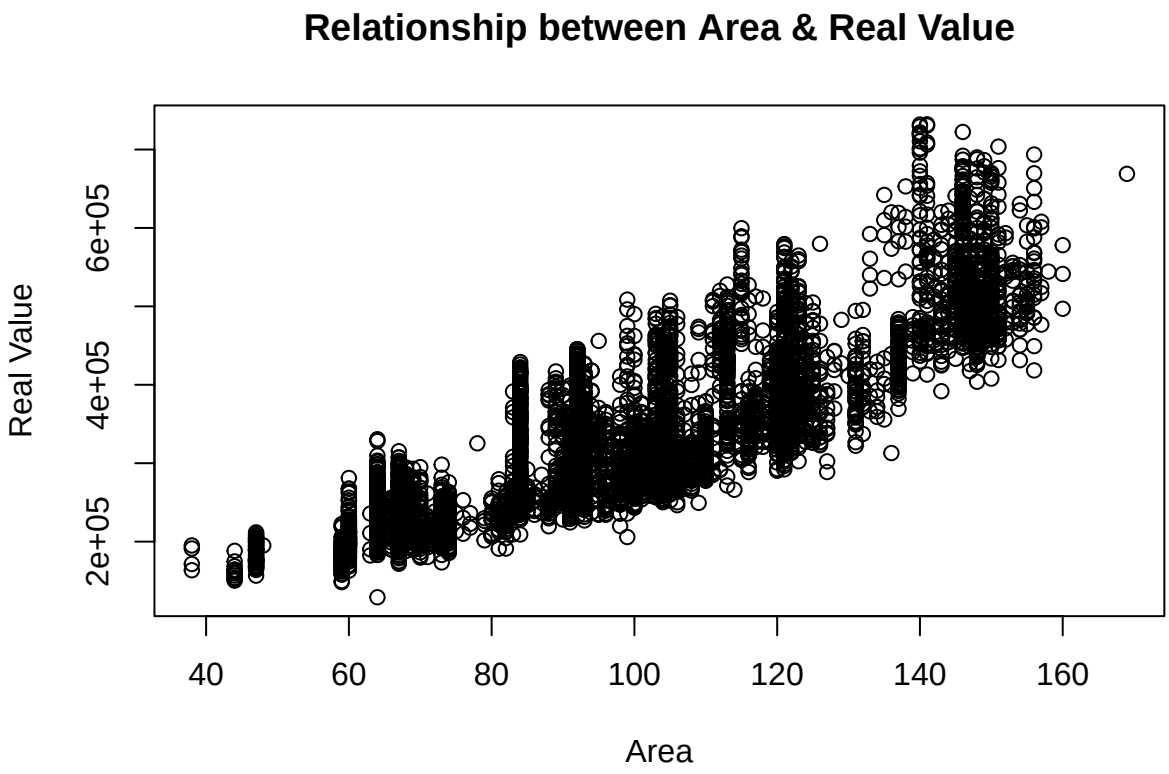
```
## 3          Type5 ROOM 0.05095137 19.626559
## 4       TypeEXECUTIVE 0.00000000       Inf
## 5       ModelImproved 0.00000000       Inf
## 6     ModelMaisonette 0.34482094  2.900056
## 7        ModelModel A 0.00000000       Inf
## 8 ModelNew Generation 0.00000000       Inf
## 9     ModelSimplified 0.00000000       Inf
```

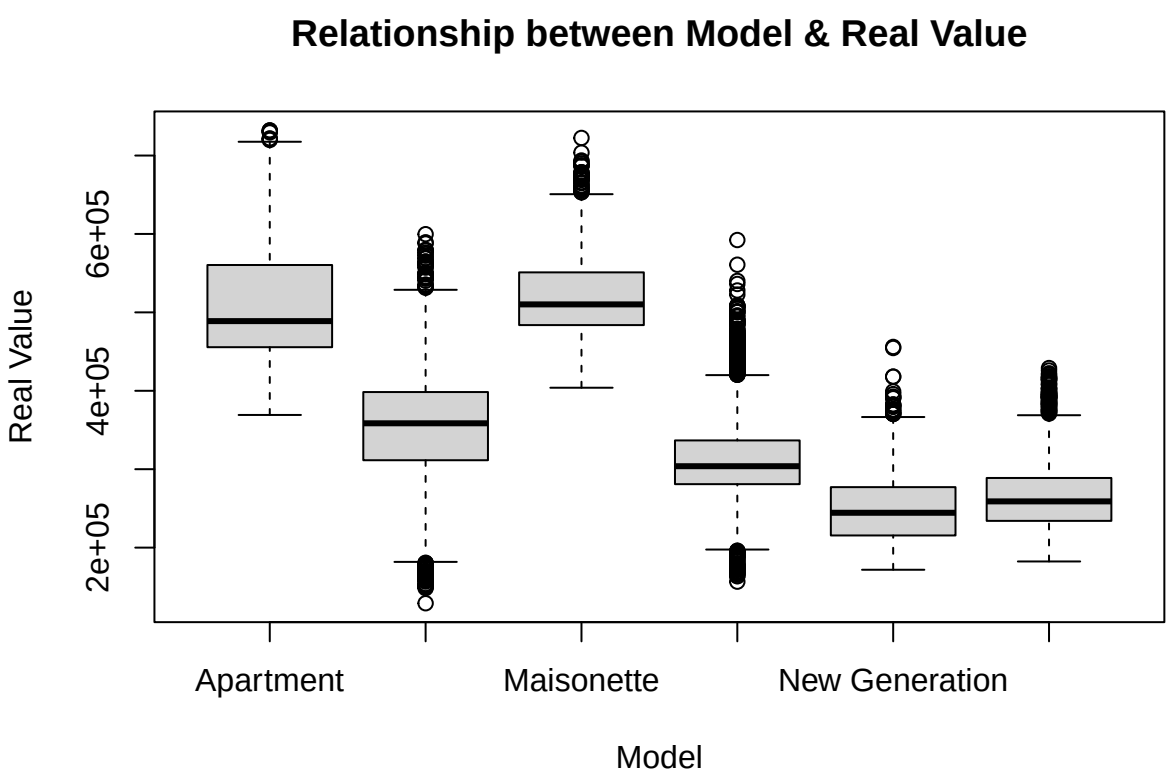
```r
#Plot of individual variables against price
for (i in colnames(train[1:(length(train[1,])-1)])) {
  plot(train[[i]], train[["Real_Value"]], xlab = i, ylab = "Real Value", main = sprintf("Relationship be
}
```

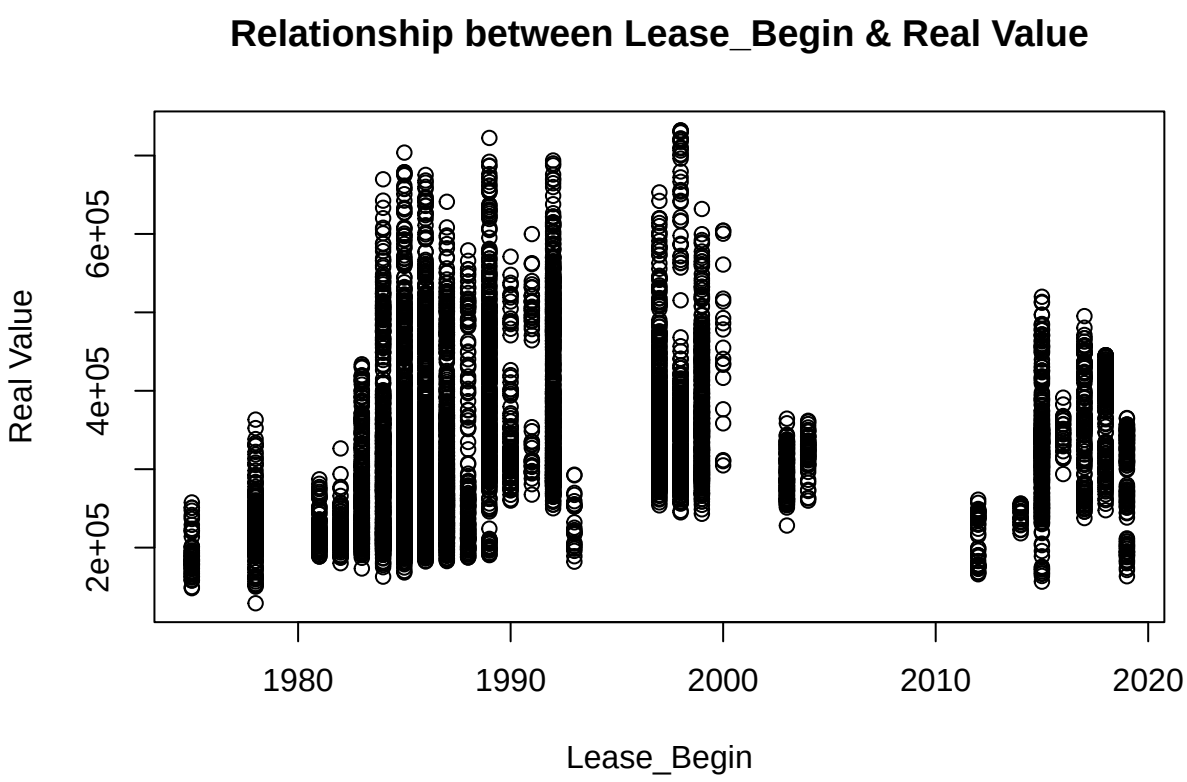


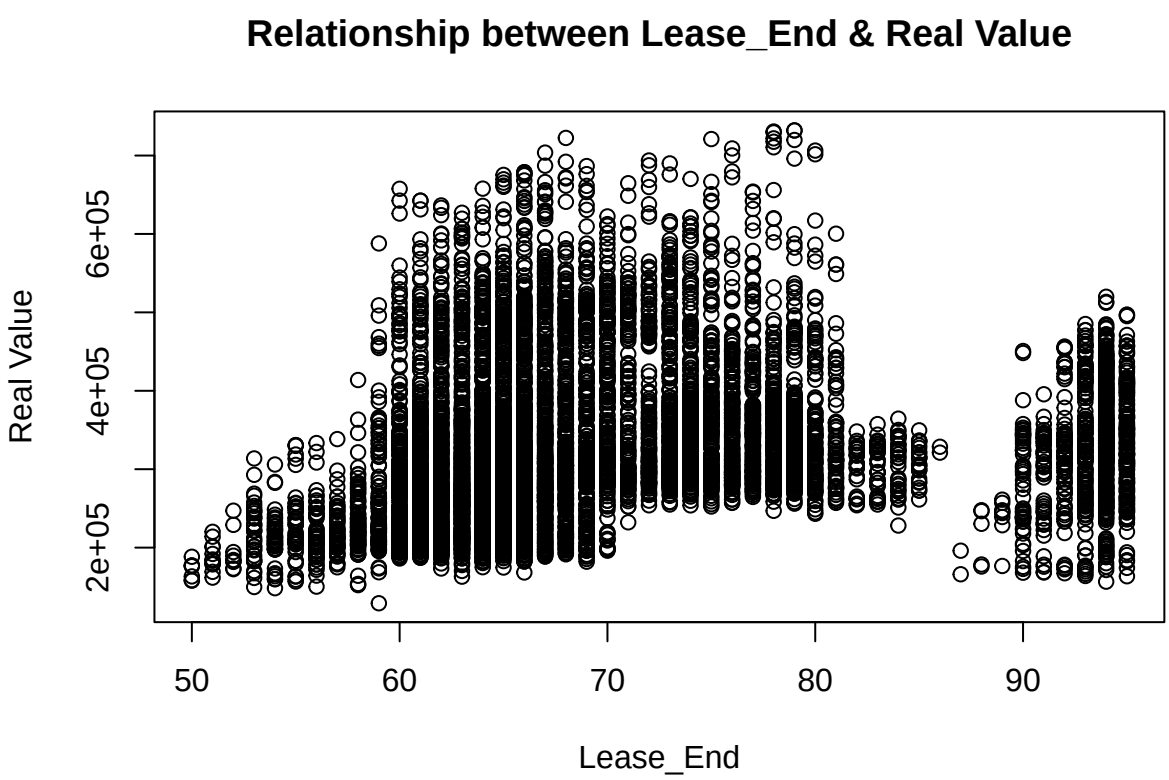

Relationship between Type & Real Value
Real Value
2e+05
4e+05
6e+05
2 ROOM
3 ROOM
4 ROOM
5 ROOM
EXECUTIVE
Type

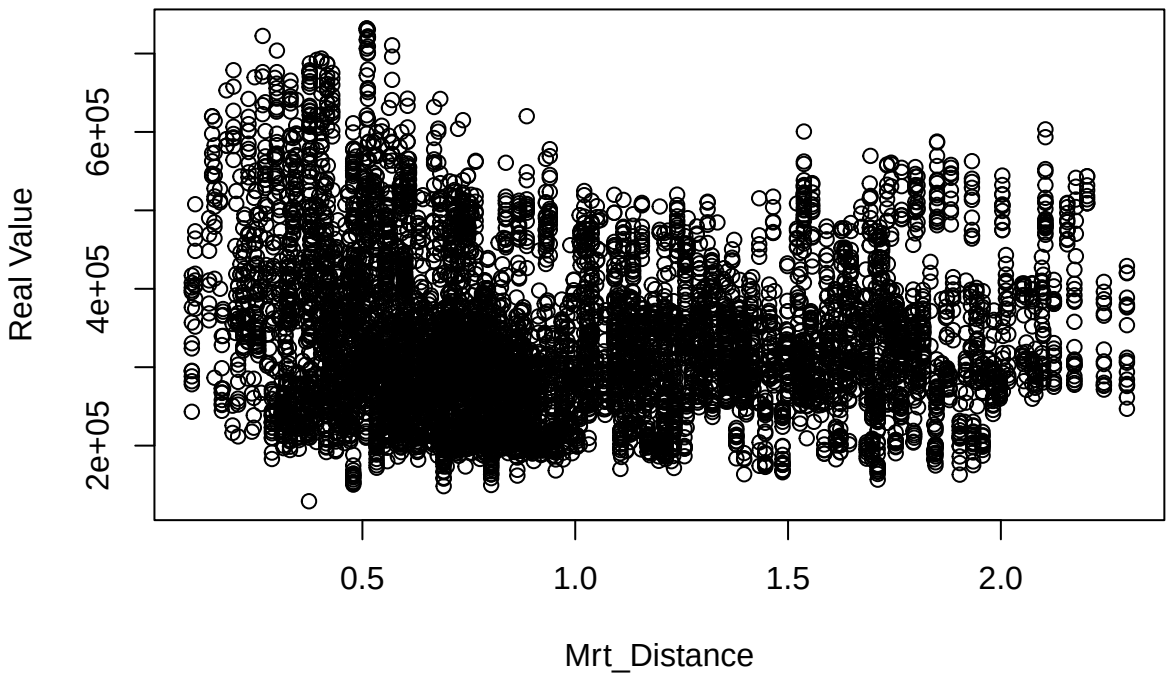

Relationship between Storey & Real Value
Real Value
2e+05
4e+05
6e+05
01 TO 03
07 TO 09
13 TO 15
Storey


Relationship between Area & Real Value
Real Value
2e+05
4e+05
6e+05
40
60
80
100
120
140
160
Area


Relationship between Model & Real Value
Real Value
2e+05
4e+05
6e+05
Apartment
Maisonette
New Generation
Model


Relationship between Lease_Begin & Real Value
Real Value
2e+05
4e+05
6e+05
1980
1990
2000
2010
2020
Lease_Begin


Relationship between Lease_End & Real Value
Real Value
2e+05
4e+05
6e+05
50
60
70
80
90
Lease_End




```
#[Serangoon seems to display higher real value as opposed to Hougang symbolizing mature estate effect]
#[Strong linear trend in type of HDB and real value]
#[Slight linear trend in storey and real value]
#[Strong linear trend in area and real value]
#[General linear trend in model and real value if ordered by mean]
#[Ambiguous trends hence might be favorable to drop this for lease end]
#[Possibly a polynomial trend]
#[Very slight linear trend or polynomial trend]

#Base Relationship
for (i in colnames(train)) {
  print(summary(lm(paste("Real_Value ~", i), data = train)))
}
```

```
##
## Call:
## lm(formula = paste("Real_Value ~", i), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -231840  -72714  -17879   51117  373278
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     320381       1326  241.66   <2e-16 ***
## TownSERANGOON    40540       2482   16.33   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 100900 on 8108 degrees of freedom
## Multiple R-squared:  0.03184,    Adjusted R-squared:  0.03173
## F-statistic: 266.7 on 1 and 8108 DF,  p-value: < 2.2e-16
##
##
## Call:
## lm(formula = paste("Real_Value ~", i), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -151087  -31634   -8777   22309  216209
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     181185       5634  32.156  < 2e-16 ***
## Type3 ROOM       42638       5755   7.409  1.4e-13 ***
## Type4 ROOM      130510       5695  22.915  < 2e-16 ***
## Type5 ROOM      205633       5769  35.646  < 2e-16 ***
## TypeEXECUTIVE   339101       5847  57.996  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 49760 on 8105 degrees of freedom
## Multiple R-squared:  0.7648, Adjusted R-squared:  0.7647
## F-statistic:  6588 on 4 and 8105 DF,  p-value: < 2.2e-16
##
##
## Call:
## lm(formula = paste("Real_Value ~", i), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -222626  -72350  -22004   50038  402188
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      311455       2430 128.187  < 2e-16 ***
## Storey04 TO 06    15841       3272   4.842 1.31e-06 ***
## Storey07 TO 09    22450       3427   6.551 6.06e-11 ***
## Storey10 TO 12    33146       3542   9.357  < 2e-16 ***
## Storey13 TO 15    40252       4601   8.749  < 2e-16 ***
## Storey16 TO 18    63628       7326   8.685  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 101600 on 8104 degrees of freedom
## Multiple R-squared:  0.01994,    Adjusted R-squared:  0.01934
## F-statistic: 32.98 on 5 and 8104 DF,  p-value: < 2.2e-16
##
##
## Call:
```

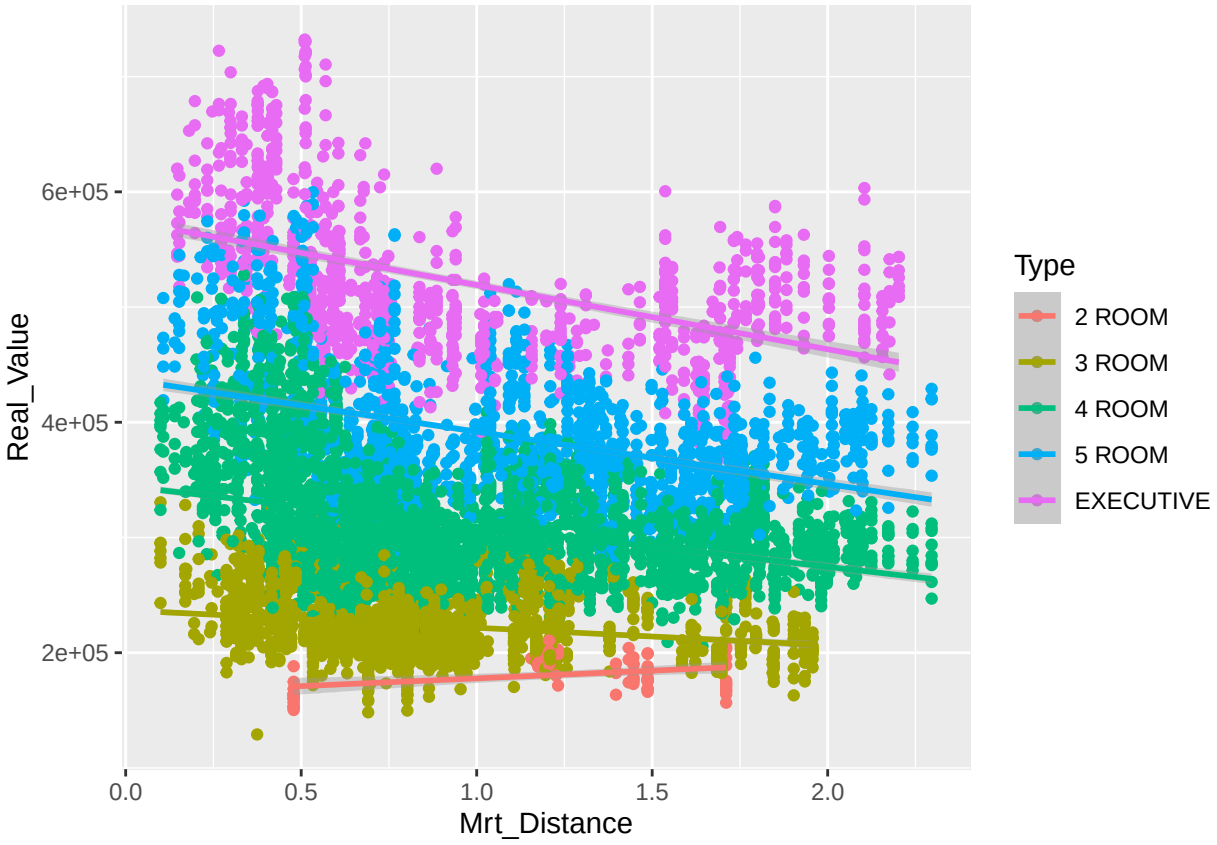
```
## lm(formula = paste("Real_Value ~", i), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -141346  -35951  -10063   25163  263722
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -14833.41    2355.33  -6.298 3.18e-10 ***
## Area          3449.68      22.71 151.922  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 52310 on 8108 degrees of freedom
## Multiple R-squared:   0.74,  Adjusted R-squared:   0.74
## F-statistic: 2.308e+04 on 1 and 8108 DF,  p-value: < 2.2e-16
##
##
## Call:
## lm(formula = paste("Real_Value ~", i), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -218928  -36007   -6086   32240  280262
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            512448       3692 138.796   <2e-16 ***
## ModelImproved         -164438       3978 -41.343   <2e-16 ***
## ModelMaisonette         11446       4462   2.565   0.0103 *
## ModelModel A          -200454       3898 -51.427   <2e-16 ***
## ModelNew Generation   -259875       4087 -63.582   <2e-16 ***
## ModelSimplified       -242125       4305 -56.243   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 66050 on 8104 degrees of freedom
## Multiple R-squared:  0.5857, Adjusted R-squared:  0.5855
## F-statistic:  2291 on 5 and 8104 DF,  p-value: < 2.2e-16
##
##
## Call:
## lm(formula = paste("Real_Value ~", i), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -212032  -69822  -26424   48021  395279
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2846493.9   214807.0  -13.25   <2e-16 ***
## Lease_Begin     1595.6      107.8   14.80   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 101200 on 8108 degrees of freedom
## Multiple R-squared:  0.02629,    Adjusted R-squared:  0.02617
## F-statistic: 218.9 on 1 and 8108 DF,  p-value: < 2.2e-16
##
##
## Call:
## lm(formula = paste("Real_Value ~", i), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -216939  -69839  -26234   48500  394492
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   208665       7739   26.96   <2e-16 ***
## Lease_End       1754        109   16.10   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 101000 on 8108 degrees of freedom
## Multiple R-squared:  0.03097,    Adjusted R-squared:  0.03085
## F-statistic: 259.2 on 1 and 8108 DF,  p-value: < 2.2e-16
##
##
## Call:
## lm(formula = paste("Real_Value ~", i), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -213957  -75434  -18705   54733  391630
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    350098       2470 141.718   <2e-16 ***
## Mrt_Distance   -18853       2279  -8.273   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 102200 on 8108 degrees of freedom
## Multiple R-squared:  0.00837,    Adjusted R-squared:  0.008248
## F-statistic: 68.44 on 1 and 8108 DF,  p-value: < 2.2e-16

## Warning in model.matrix.default(mt, mf, contrasts): the response appeared on
## the right-hand side and was dropped

## Warning in model.matrix.default(mt, mf, contrasts): problem with term 1 in
## model.matrix: no columns are assigned

##
## Call:
## lm(formula = paste("Real_Value ~", i), data = train)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -202862  -73903  -21019   53585  400166
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   331943       1139   291.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 102600 on 8109 degrees of freedom
```

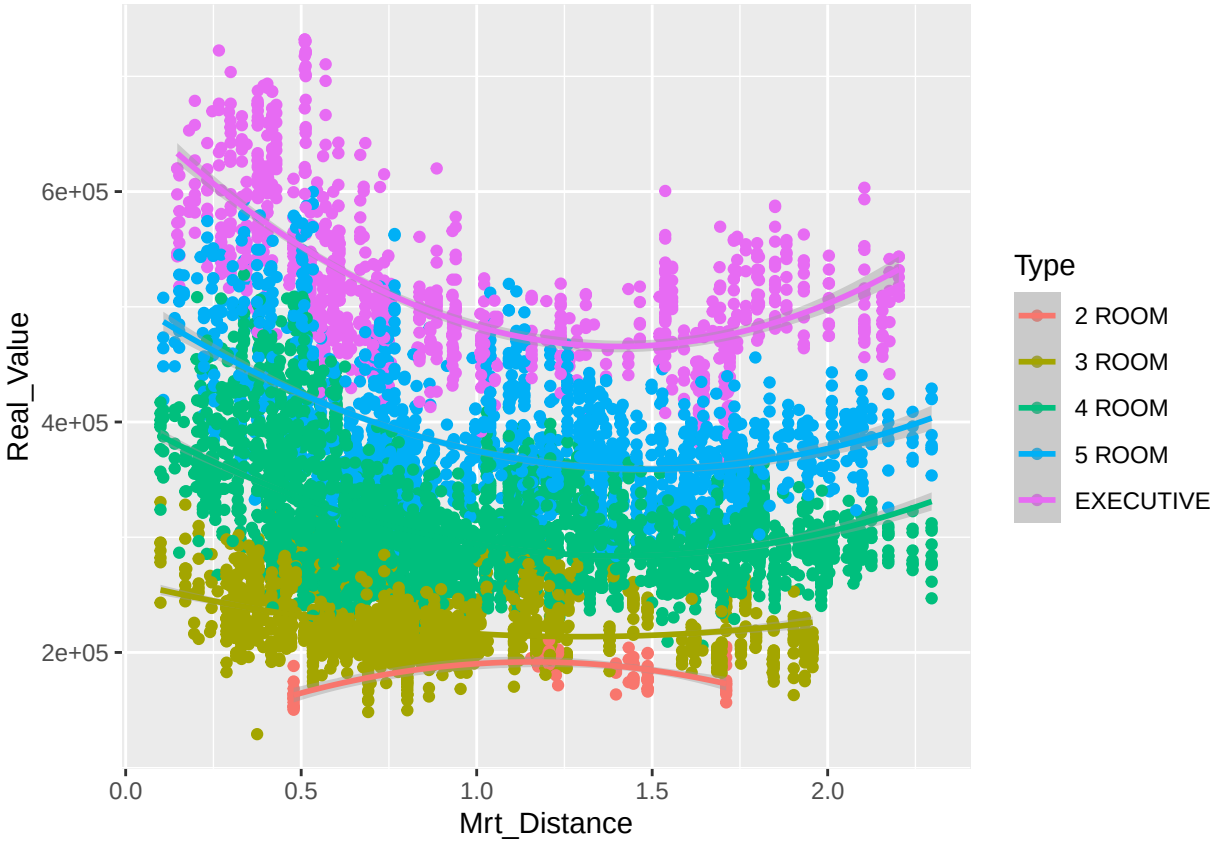
```
#[All variables shows statistical significance to 1% level however low explanatory power except Model,

#Polynomial Variations
ggplot(train, aes(x = Mrt_Distance, y = Real_Value, color = Type)) + geom_point() + geom_smooth(formula
```

```
## `geom_smooth()` using method = 'gam'
```



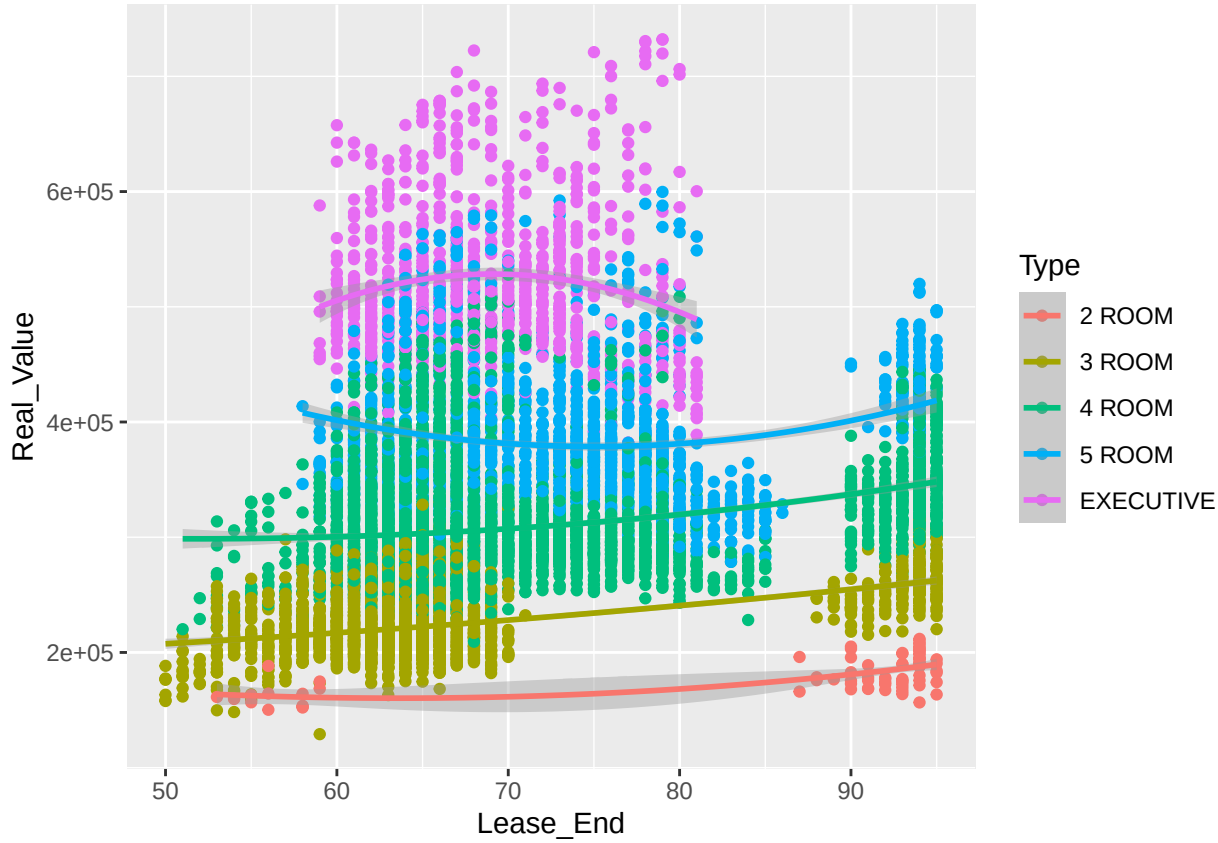

```
ggplot(train, aes(Mrt_Distance, Real_Value, color = Type)) + geom_point() + geom_smooth(formula = y ~ p
```

```
## `geom_smooth()` using method = 'gam'
```



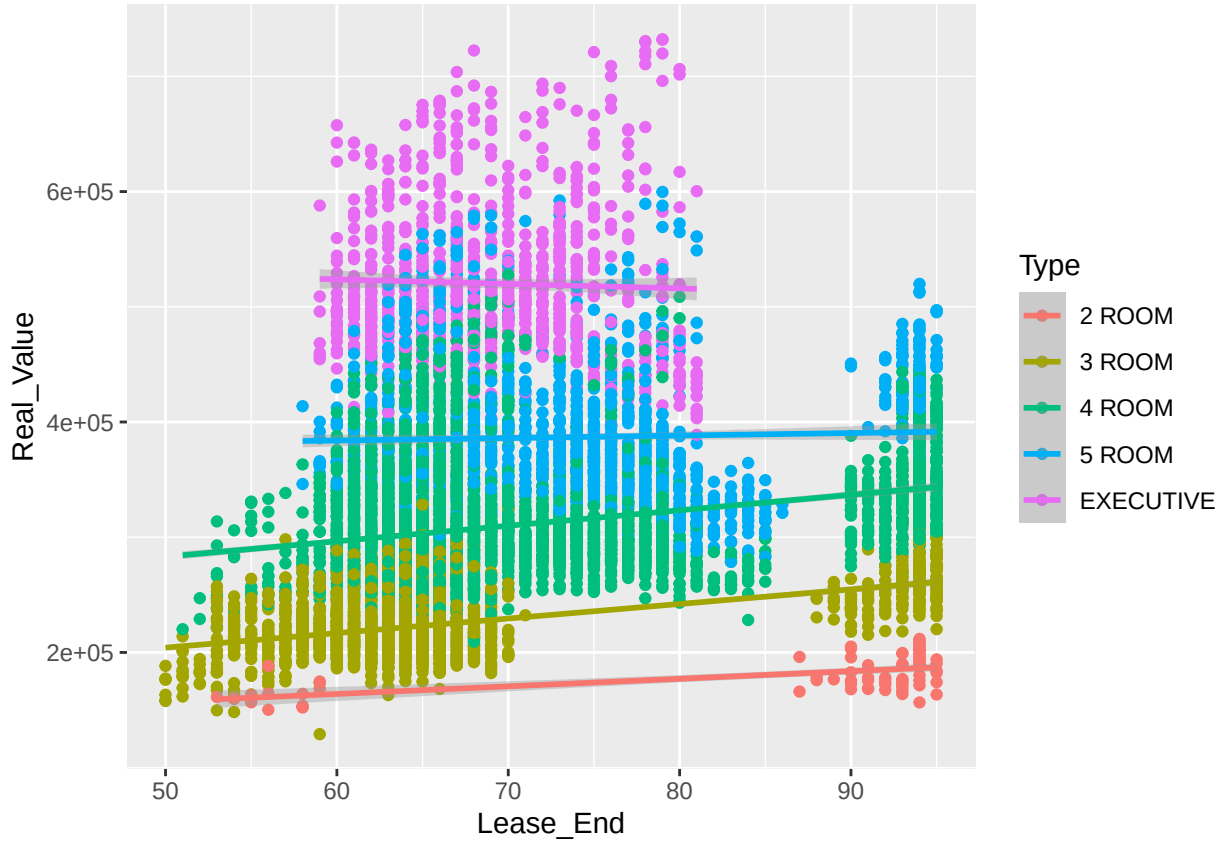

```
ggplot(train, aes(Lease_End, Real_Value, color = Type)) + geom_point() + geom_smooth(formula = y ~ poly
```

```
## `geom_smooth()` using method = 'gam'
```



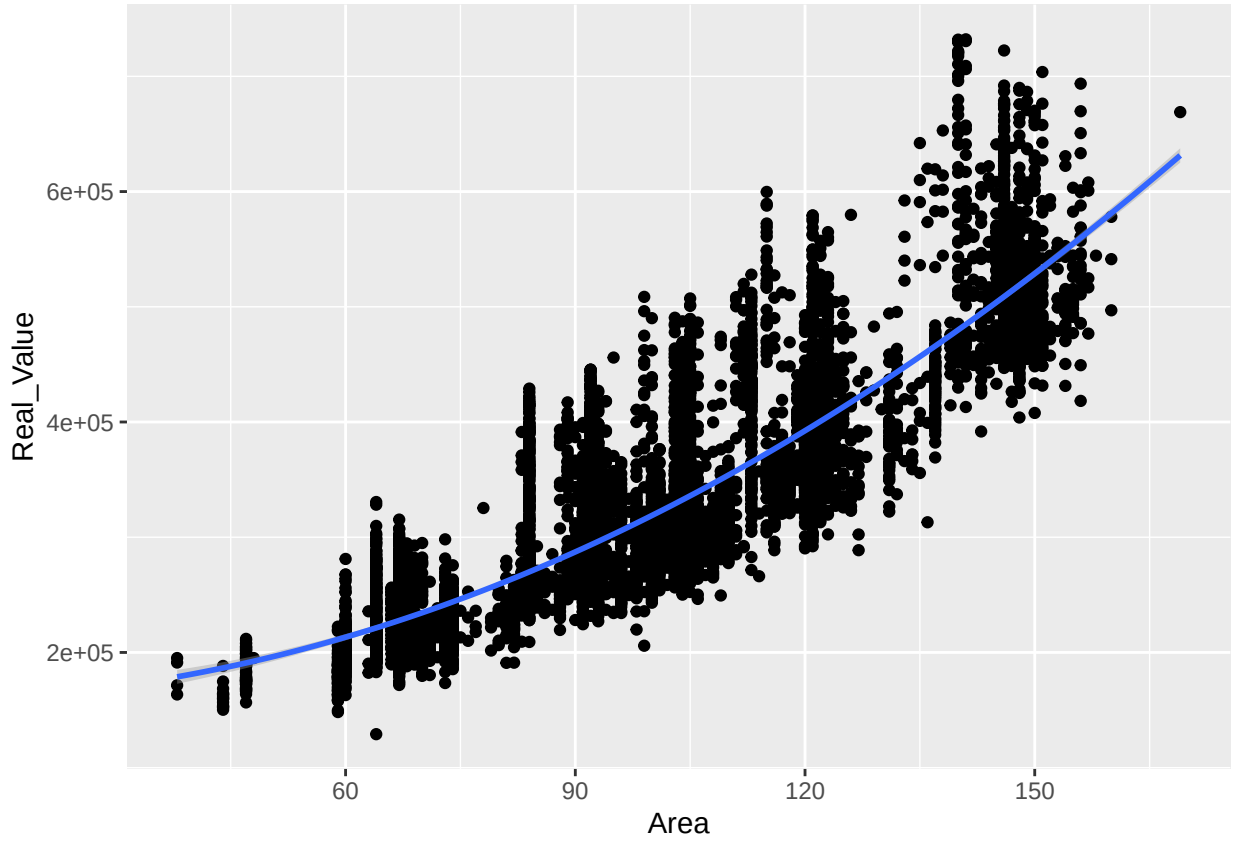

```
ggplot(train, aes(Lease_End, Real_Value, color = Type)) + geom_point() + geom_smooth(formula = y ~ x)
```

```
## `geom_smooth()` using method = 'gam'
```



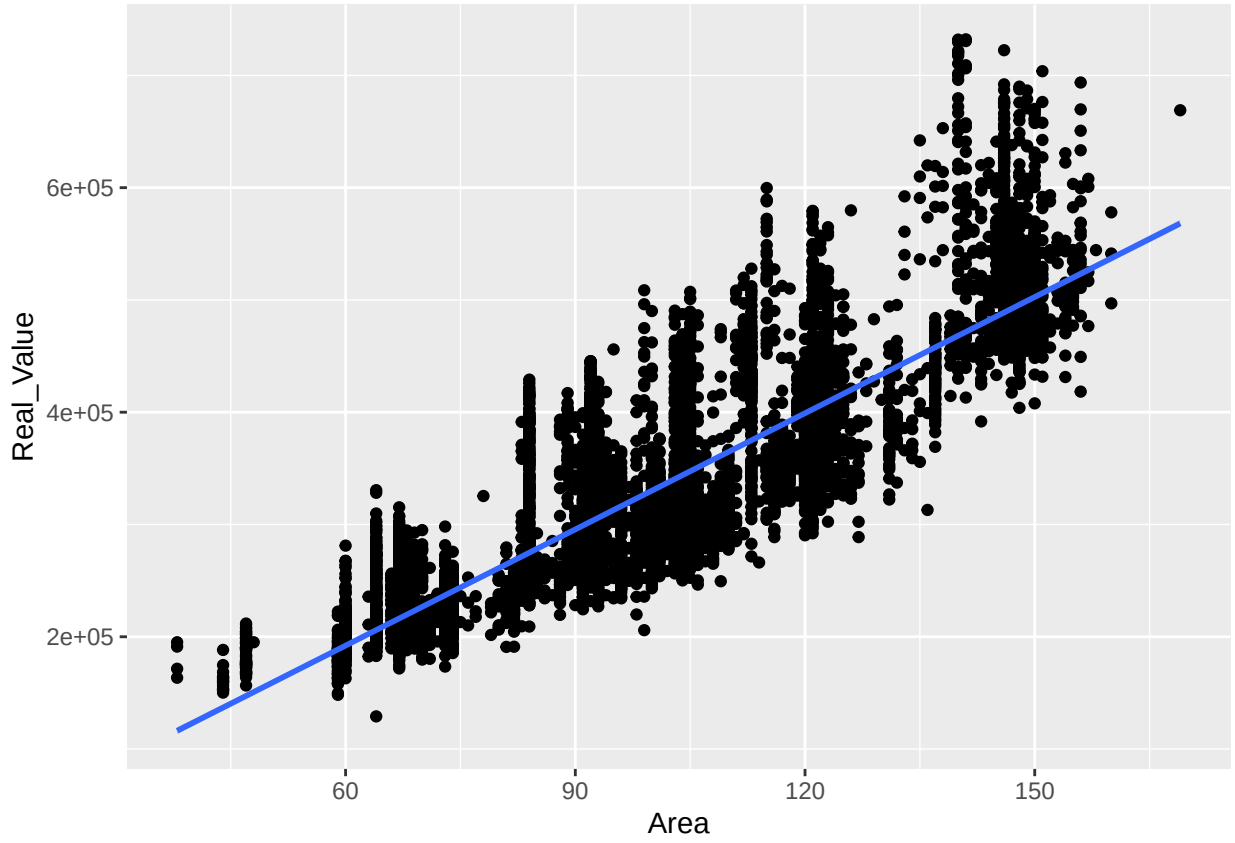

```
ggplot(train, aes(Area, Real_Value)) + geom_point() + geom_smooth(formula = y ~ poly(x,2))
```

```
## `geom_smooth()` using method = 'gam'
```




```
ggplot(train, aes(Area, Real_Value)) + geom_point() + geom_smooth(formula = y ~ x)
```

```
## `geom_smooth()` using method = 'gam'
```




```
#Poly function overfits to the training data hence better not to use it

#Best Subset Selection
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
set.seed(6789)
k = 5

data_area = subset(train, select = -c(Type))
data_type = subset(train, select = -c(Area, Model))

predict.regsubsets <- function(object, newdata, id) {
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form, newdata)
  coefi <- coef(object, id = id)
  xvars <- names(coefi)
  mat[, xvars] %*% coefi
}

folds <- sample(1:k, nrow(data_area), replace = TRUE)
cv.errors <- matrix(NA, k, 14, dimnames = list(NULL, paste(1:14)))
for (j in 1:k){
```

```
  best.fit1 <- regsubsets(Real_Value ~ ., data = data_area[folds != j,], nvmax = 14) #Function for perf
  for (i in 1:14) {
    pred <- predict.regsubsets(best.fit1, data_area[folds ==j,], id = i)
    cv.errors[j, i] <- mean((train$Real_Value[folds ==j] - pred)^2) #Function for making prediction for
  }
}
mean.cv1 <- apply(cv.errors, 2, mean) #apply function averages over the columns of the matrix
aa <- which.min(mean.cv1)

folds <- sample(1:k, nrow(data_type), replace = TRUE)
cv.errors <- matrix(NA, k, 12, dimnames = list(NULL, paste(1:12)))
for (j in 1:k){
  best.fit2 <- regsubsets(Real_Value ~ ., data = data_type[folds != j,], nvmax = 12) #Function for perf
  for (i in 1:12) {
    pred <- predict.regsubsets(best.fit2, data_type[folds ==j,], id = i)
    cv.errors[j, i] <- mean((train$Real_Value[folds ==j] - pred)^2) #Function for making prediction for
  }
}
mean.cv2 <- apply(cv.errors, 2, mean) #apply function averages over the columns of the matrix
bb <- which.min(mean.cv2)

folds <- sample(1:k, nrow(data_area), replace = TRUE)
cv.errors <- matrix(NA, k, 16, dimnames = list(NULL, paste(1:16)))
for (j in 1:k){
  best.fit3 <- regsubsets(Real_Value ~ . - Mrt_Distance - Lease_End + poly(Mrt_Distance, 2) + poly(Lease
  for (i in 1:16) {
    pred <- predict.regsubsets(best.fit3, data_area[folds ==j,], id = i)
    cv.errors[j, i] <- mean((train$Real_Value[folds ==j] - pred)^2) #Function for making prediction for
  }
}
mean.cv3 <- apply(cv.errors, 2, mean) #apply function averages over the columns of the matrix
cc <- which.min(mean.cv3)

folds <- sample(1:k, nrow(data_type), replace = TRUE)
cv.errors <- matrix(NA, k, 14, dimnames = list(NULL, paste(1:14)))
for (j in 1:k){
  best.fit4 <- regsubsets(Real_Value ~ . - Mrt_Distance - Lease_End + poly(Mrt_Distance, 2) + poly(Lease
  for (i in 1:14) {
    pred <- predict.regsubsets(best.fit4, data_type[folds ==j,], id = i)
    cv.errors[j, i] <- mean((train$Real_Value[folds ==j] - pred)^2) #Function for making prediction for
  }
}
mean.cv4 <- apply(cv.errors, 2, mean) #apply function averages over the columns of the matrix
dd <- which.min(mean.cv4) #finds which regression model has the lowest MSE

c(mean.cv1[aa], mean.cv2[bb], mean.cv3[cc], mean.cv4[dd])
```

```
##         14         12         16         14
##  878744872 1060288005 1769168788 1899790763
```

```
#First model is the best and all variables are used
coef(best.fit1, aa) #[Base Model MSE: 863774121]
```

```
##         (Intercept)       TownSERANGOON      Storey04 TO 06      Storey07 TO 09
##       -3608849.331           61732.105           14688.683           24512.176
##      Storey10 TO 12      Storey13 TO 15      Storey16 TO 18                Area
##           30280.755           36584.286           50863.613            2837.988
##       ModelImproved       ModelModel A ModelNew Generation     ModelSimplified
##          -64448.288          -80090.536          -72605.065          -62493.218
##         Lease_Begin           Lease_End        Mrt_Distance
##            1836.223            1210.195          -63782.957
```

```
#Interaction Effects
#Variables: Town, Storey, Area, Model, Lease_End, Mrt_Distance

#Initial Best Model
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
set.seed(6789)

lr.mod1 = glm(Real_Value ~ . - Type, data = train)
cv.err1 = cv.glm(train, lr.mod1, K = 5)
cv.err1$delta[1] #[Initial RMSE]
```

```
## [1] 877583541
```

```
#Town:Area#Town:Areacolours()
lr.mod2 = glm(Real_Value ~ . - Type + Town:Area, data = train)
cv.err2 = cv.glm(train, lr.mod2, K = 5)
cv.err2$delta[1] #[Large improvement in RMSE]
```

```
## [1] 822692320
```

```
#Town:Model
lr.mod3 = glm(Real_Value ~ . - Type + Town:Area + Town:Model, data = train)
cv.err3 = cv.glm(train, lr.mod3, K = 5)
cv.err3$delta[1] #[Good improvement in RMSE]
```

```
## [1] 794332759
```

```
#Storey:Area
lr.mod4 = glm(Real_Value ~ . - Type + Town:Area + Town:Model + Storey:Area, data = train)
cv.err4 = cv.glm(train, lr.mod4, K = 5)
cv.err4$delta[1] #[Marginal improvement in RMSE]
```

```
## [1] 774288968
```

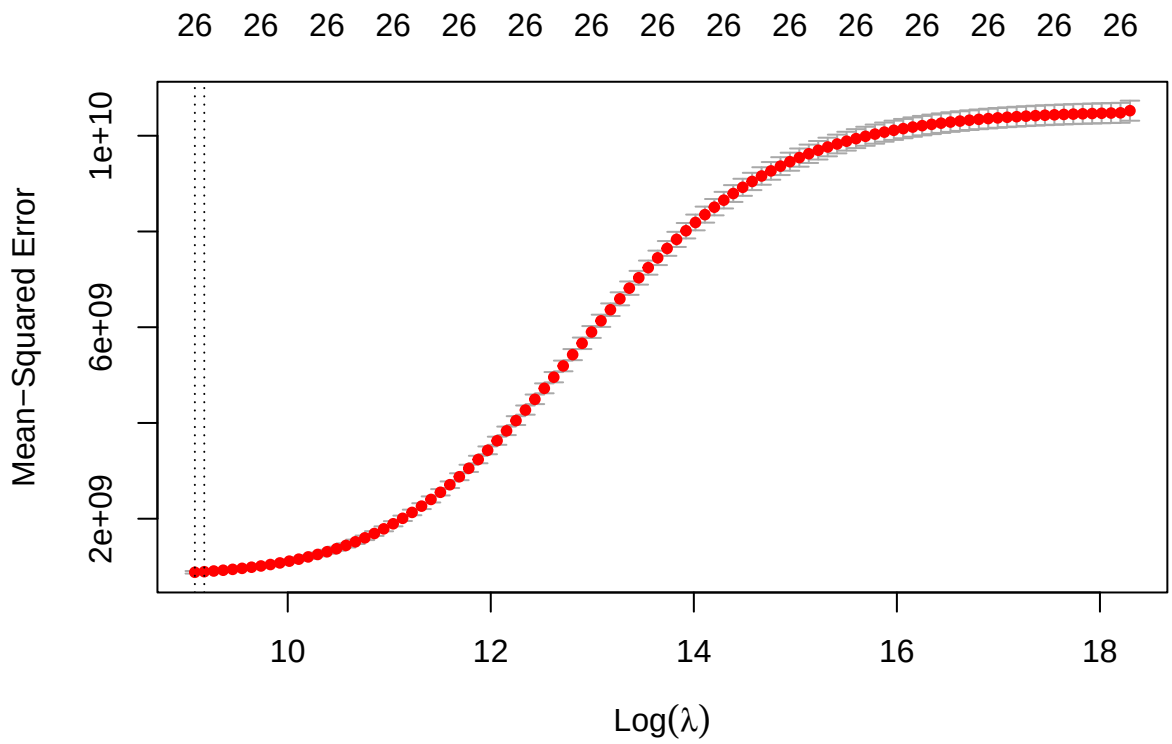
```
#Out-of-sample performance
test_var = subset(test, select = -c(Real_Value))
lr.pred = predict(lr.mod4, newdata = test_var)
mean((test$Real_Value - lr.pred)^2) #LR OOS Perf: 768447891
```

```
## [1] 790438708
```

```
#Normalization
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

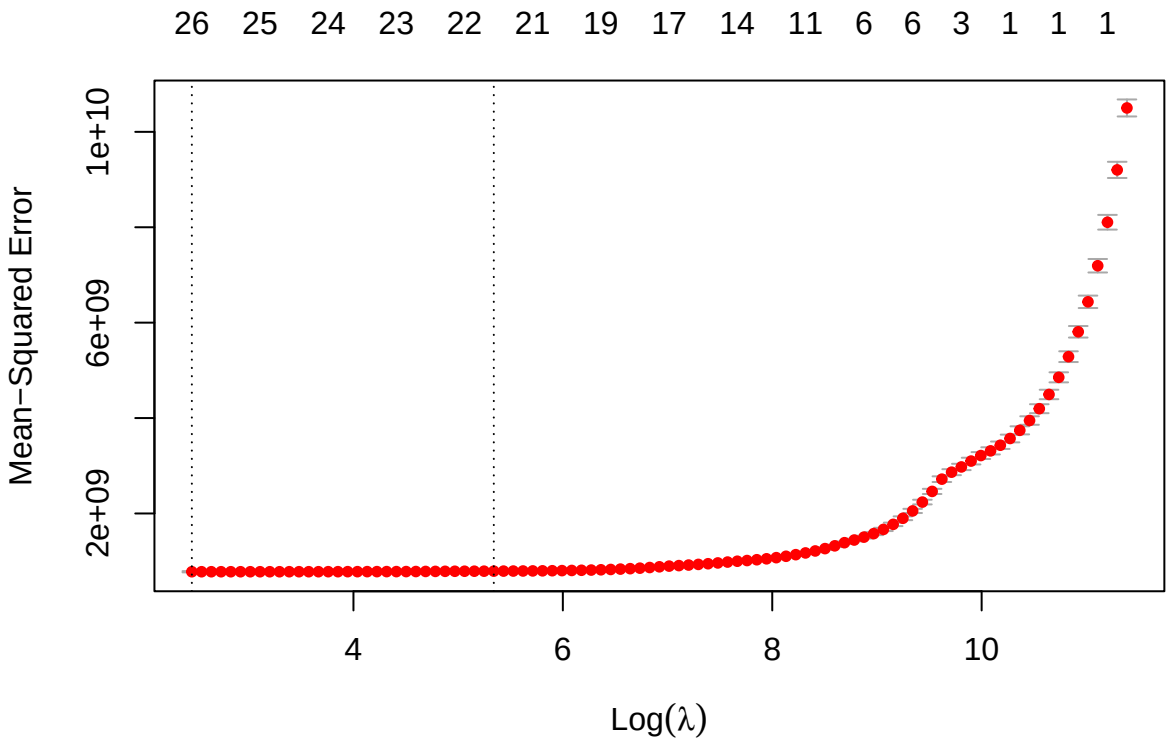
```
set.seed(6789)

#Train-Test Generation
x.train <- model.matrix(Real_Value ~ . - Type + Town:Area + Town:Model + Storey:Area, data = train)[, -
y.train <- train$Real_Value

x.test <- model.matrix(Real_Value ~ . - Type + Town:Area + Town:Model + Storey:Area, data = test)[, -1]
y.test <- test$Real_Value

#Ridge Regression
ridge.mod <- cv.glmnet(x.train, y.train, alpha = 0)
plot(ridge.mod)
```




```
bestlam_ridge <- ridge.mod$lambda.min
ridge.pred <- predict(ridge.mod, s = bestlam_ridge, newx = x.test)
mean((y.test - ridge.pred)^2)
```

```
## [1] 937487731
```

```
#Lasso Regression
lasso.mod <- cv.glmnet(x.train, y.train, alpha = 1)
plot(lasso.mod)
```




```
bestlam_lasso <- lasso.mod$lambda.min
lasso.pred <- predict(lasso.mod, s = bestlam_lasso, newx = x.test)
mean((y.test - lasso.pred)^2)
```

```
## [1] 790672509
```

```
#Ridge OOS Perf: 878373145
#Lasso OOS Perf: 774889012

#RandomForest
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
set.seed(6789)

rf.mod <- randomForest(Real_Value ~ . - Type + Town:Area + Town:Model + Storey:Area, data = train, mtry
```

```r
rf.pred = predict(rf.mod, newdata = test_var)
mean((test$Real_Value - rf.pred)^2) #[Greatly improved RMSE]
```

```
## [1] 442844911
```

```r
#RF OOS Perf: 427348172

#Test Predictions
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

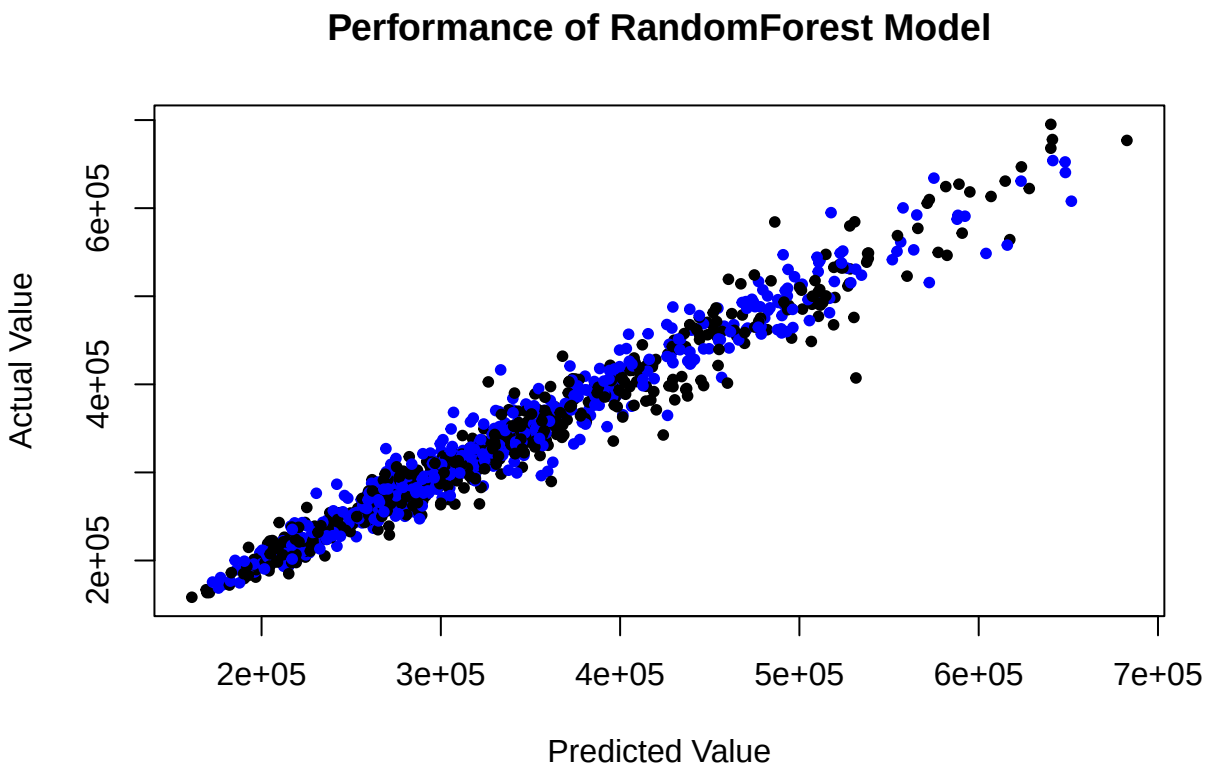
```r
set.seed(6789)

final_test_var = subset(final_test, select = -c(month, ID, Type, Real_Value))
final_lr.mod = lm(Real_Value ~ . + Town:Area + Town:Model + Storey:Area, data = final_train[, -c(1,2,4,8
final_lr.pred = predict(final_lr.mod, newdata = final_test_var)
mean((final_test$Real_Value - final_lr.pred)^2)
```

```
## [1] 783364623
```

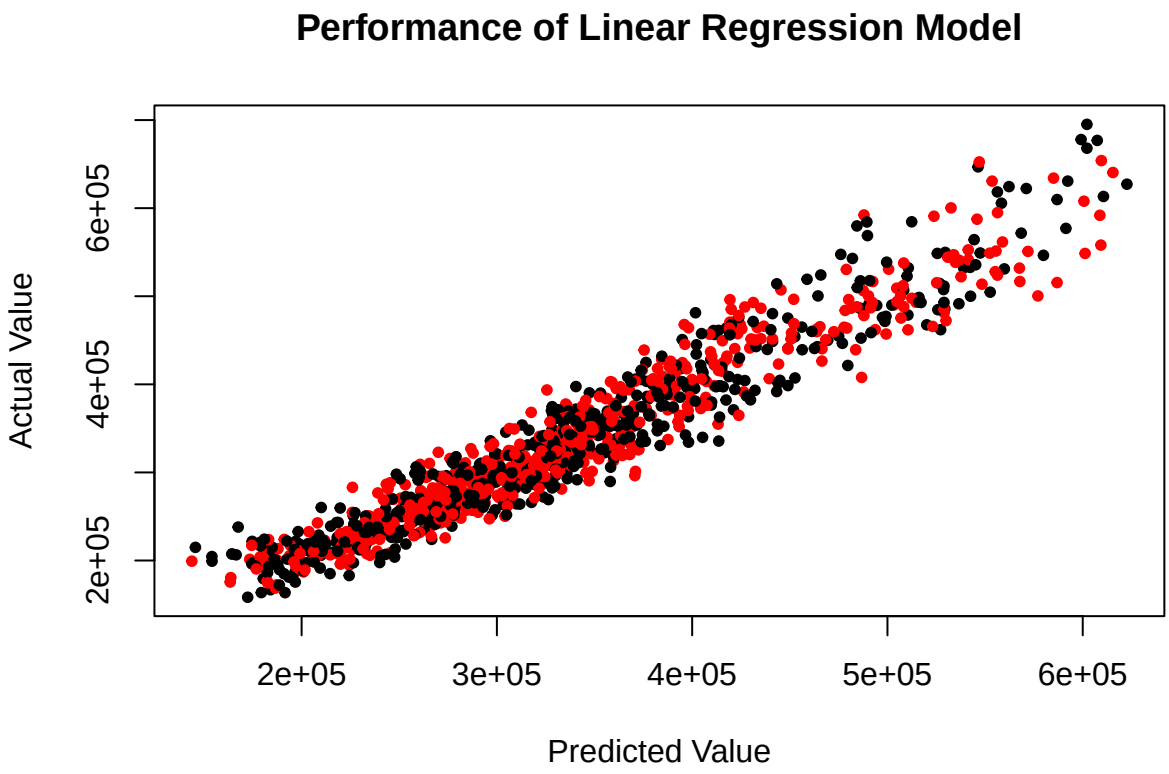
```r
final_rf.mod = randomForest(Real_Value ~ . + Town:Area + Town:Model + Storey:Area, data = final_train[,
final_rf.pred = predict(final_rf.mod, newdata = final_test_var)
mean((final_rf.pred - final_test$Real_Value)^2)
```

```
## [1] 400409313
```

```r
#Final LR perf: 857393146
#Final RF perf: 450740537

#Ploting the Predictions
plot(final_rf.pred, final_test$Real_Value, col = c("blue","black"), pch = 20, xlab = "Predicted Value",
```




```
plot(final_lr.pred, final_test$Real_Value, col = c("red","black"), pch = 20, xlab = "Predicted Value", y
```


Performance of Linear Regression Model
Actual Value
2e+05
4e+05
6e+05
2e+05
3e+05
4e+05
5e+05
6e+05
Predicted Value