Melbourne Housing Snapshot - Price Prediction

Yohan Chandrasukmana - 01112190011

4/25/2022

Contents

"Saya Yohan Chandrasukmana, menyatakan bahwa saya mengerjakan soal-soal ini secara mandiri tanpa bantuan orang lain ataupun memberikan bantuan kepada orang lain. Jika saya terbukti melakukan kecurangan tersebut (mendapat bantuan dari orang lain ataupun memberi bantuan kepada orang lain) maka saya bersedia untuk tidak lulus dalam mata kuliah ini."

0.1 Loading Library

Libraries used in this project are as follows.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## Warning: package 'dplyr' was built under R version 4.1.2

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

```
library(caret) # for splitting train-test set and train-eval set
```

```
## Warning: package 'caret' was built under R version 4.1.2

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(ggpubr) # for arranging ggplot
```

```
## Warning: package 'ggpubr' was built under R version 4.1.2
```

```
library(gam) # for GAM
```

```
## Warning: package 'gam' was built under R version 4.1.2
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 4.1.2
```

```
##
## Attaching package: 'foreach'
```

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

```
## Loaded gam 1.20
```

```
library(car) # for calculating collinearity with vif
```

```
## Warning: package 'car' was built under R version 4.1.2
```

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

```
## Warning: package 'carData' was built under R version 4.1.2
```

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

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

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(gbm) # for GBM
```

```
## Warning: package 'gbm' was built under R version 4.1.3
```

```
## Loaded gbm 2.1.8
```

```
library(tictoc) # to measure time
library(reshape2) # melt()
```

```
## Warning: package 'reshape2' was built under R version 4.1.2
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

---

1 Data Description

Data used in this study case are Melbourne real estate prices from the Melbourne Housing Snapshot dataset. Accessed on April 14, 2022, https://www.kaggle.com/datasets/dansbecker/melbourne-housingsnapshot

1.1 Data Loading and Prerequisites

Data saved in the CSV files will be loaded using the function `read.csv`.

```
rm(list=ls())   # clean up workspace
melb <- read.csv('melb_data_clean.csv')
head(melb)
```

```
##   ID     Suburb         Address Rooms Type   Price Method SellerG      Date
## 1  1 Abbotsford 25 Bloomburg St     2    h 1035000      S  Biggin 4/02/2016
## 2  2 Abbotsford    5 Charles St     3    h 1465000     SP  Biggin 4/03/2017
## 3  3 Abbotsford     55a Park St     4    h 1600000     VB  Nelson 4/06/2016
## 4  4 Abbotsford    124 Yarra St     3    h 1876000      S  Nelson 7/05/2016
## 5  5 Abbotsford   98 Charles St     2    h 1636000      S  Nelson 8/10/2016
## 6  6 Abbotsford   10 Valiant St     2    h 1097000      S  Biggin 8/10/2016
##   Distance Postcode Bedroom2 Bathroom Car Landsize BuildingArea YearBuilt
## 1      2.5     3067        2        1   0      156           79      1900
## 2      2.5     3067        3        2   0      134          150      1900
## 3      2.5     3067        3        1   2      120          142      2014
## 4      2.5     3067        4        2   0      245          210      1910
## 5      2.5     3067        2        1   2      256          107      1890
## 6      2.5     3067        3        1   2      220           75      1900
##   CouncilArea Lattitude Longtitude             Regionname Propertycount
## 1       Yarra  -37.8079   144.9934 Northern Metropolitan          4019
## 2       Yarra  -37.8093   144.9944 Northern Metropolitan          4019
## 3       Yarra  -37.8072   144.9941 Northern Metropolitan          4019
## 4       Yarra  -37.8024   144.9993 Northern Metropolitan          4019
## 5       Yarra  -37.8060   144.9954 Northern Metropolitan          4019
## 6       Yarra  -37.8010   144.9989 Northern Metropolitan          4019
##   SalesYear EffAge
```

```
## 1      2016    100
## 2      2017    100
## 3      2016      2
## 4      2016    100
## 5      2016    100
## 6      2016    100
```

---

2 Data Exploration

```
summary(melb)
```

```
##        ID           Suburb             Address              Rooms
##  Min.   :   1   Length:5179        Length:5179        Min.   :1.000
##  1st Qu.:1296   Class :character   Class :character   1st Qu.:3.000
##  Median :2590   Mode  :character   Mode  :character   Median :3.000
##  Mean   :2590                                         Mean   :3.119
##  3rd Qu.:3884                                         3rd Qu.:4.000
##  Max.   :5179                                         Max.   :8.000
##      Type                Price            Method            SellerG
##  Length:5179        Min.   : 131000   Length:5179        Length:5179
##  Class :character   1st Qu.: 700000   Class :character   Class :character
##  Mode  :character   Median : 960000   Mode  :character   Mode  :character
##                     Mean   :1156211
##                     3rd Qu.:1416375
##                     Max.   :9000000
##      Date              Distance       Postcode       Bedroom2
##  Length:5179        Min.   : 0.7   Min.   :3000   Min.   :0.000
##  Class :character   1st Qu.: 6.6   1st Qu.:3043   1st Qu.:3.000
##  Mode  :character   Median : 9.7   Median :3079   Median :3.000
##                     Mean   :10.4   Mean   :3101   Mean   :3.086
##                     3rd Qu.:13.0   3rd Qu.:3146   3rd Qu.:4.000
##                     Max.   :47.4   Max.   :3977   Max.   :9.000
##     Bathroom         Car              Landsize        BuildingArea
##  Min.   :1.000   Length:5179        Min.   :    1.0   Min.   :   1.0
##  1st Qu.:1.000   Class :character   1st Qu.:  248.0   1st Qu.: 104.0
##  Median :2.000   Mode  :character   Median :  488.0   Median : 133.0
##  Mean   :1.641                      Mean   :  563.4   Mean   : 153.1
##  3rd Qu.:2.000                      3rd Qu.:  660.0   3rd Qu.: 180.3
##  Max.   :8.000                      Max.   :37000.0   Max.   :3112.0
##    YearBuilt    CouncilArea          Lattitude        Longtitude
##  Min.   :1196   Length:5179        Min.   :-38.16   Min.   :144.5
##  1st Qu.:1930   Class :character   1st Qu.:-37.85   1st Qu.:144.9
##  Median :1965   Mode  :character   Median :-37.80   Median :145.0
##  Mean   :1961                      Mean   :-37.80   Mean   :145.0
##  3rd Qu.:1997                      3rd Qu.:-37.75   3rd Qu.:145.1
##  Max.   :2018                      Max.   :-37.46   Max.   :145.5
##   Regionname        Propertycount     SalesYear        EffAge
##  Length:5179        Min.   :  389   Min.   :2016   Min.   :  0.00
##  Class :character   1st Qu.: 4019   1st Qu.:2016   1st Qu.: 19.00
```

```
##  Mode  :character   Median : 6482   Median :2016   Median : 52.00
##                     Mean   : 7252   Mean   :2016   Mean   : 52.33
##                     3rd Qu.: 9264   3rd Qu.:2017   3rd Qu.: 86.00
##                     Max.   :21650   Max.   :2017   Max.   :100.00
```

There are some variables that are not going to be used in the study.

```
# Removing Unused Variables
unused <- c("Suburb", "Address", "Method", "SellerG", "Date",
            "Postcode", "Bedroom2", "Bathroom", "YearBuilt", "Lattitude",
            "Longtitude", "Propertycount", "SalesYear")
melb <- melb[, !names(melb) %in% unused]
```

Next, since the ID variable contains unique values for each entry will also be removed.

```
melb <- melb[, !names(melb) %in% "ID"]
```

2.1 Type Casting

```
# Numerical Variable
numerical <- c('Distance', 'Landsize', 'BuildingArea', 'EffAge')

# Categorical Variable
categorical <- c('Rooms', 'Type', 'Car', 'Regionname', 'CouncilArea')
melb[, categorical] = lapply(melb[, categorical], factor)

# Changing the order of levels for 'Car'
melb$Car <- factor(melb$Car, levels=c("0","1","2","3",">4"))

summary(melb)
```

```
##      Rooms      Type         Price            Distance    Car
##  3      :2394   h:4012   Min.   : 131000   Min.   : 0.7   0 : 377
##  4      :1251   t: 516   1st Qu.: 700000   1st Qu.: 6.6   1 :1940
##  2      :1118   u: 651   Median : 960000   Median : 9.7   2 :2302
##  5      : 281            Mean   :1156211   Mean   :10.4   3 : 293
##  1      :  99            3rd Qu.:1416375   3rd Qu.:13.0   >4: 267
##  6      :  27            Max.   :9000000   Max.   :47.4
##  (Other):   9
##     Landsize        BuildingArea          CouncilArea
##  Min.   :    1.0   Min.   :   1.0   Moreland     : 574
##  1st Qu.:  248.0   1st Qu.: 104.0   Boroondara   : 494
##  Median :  488.0   Median : 133.0   Moonee Valley: 423
##  Mean   :  563.4   Mean   : 153.1   Darebin      : 381
##  3rd Qu.:  660.0   3rd Qu.: 180.3   Maribyrnong  : 350
##  Max.   :37000.0   Max.   :3112.0   Glen Eira    : 348
##                                     (Other)      :2609
##                       Regionname       EffAge
##  Southern Metropolitan     :1659   Min.   :  0.00
##  Northern Metropolitan     :1519   1st Qu.: 19.00
##  Western Metropolitan      :1242   Median : 52.00
```

```
##  Eastern Metropolitan      : 551   Mean   : 52.33
##  South-Eastern Metropolitan: 152   3rd Qu.: 86.00
##  Eastern Victoria          :  23   Max.   :100.00
##  (Other)                   :  33
```

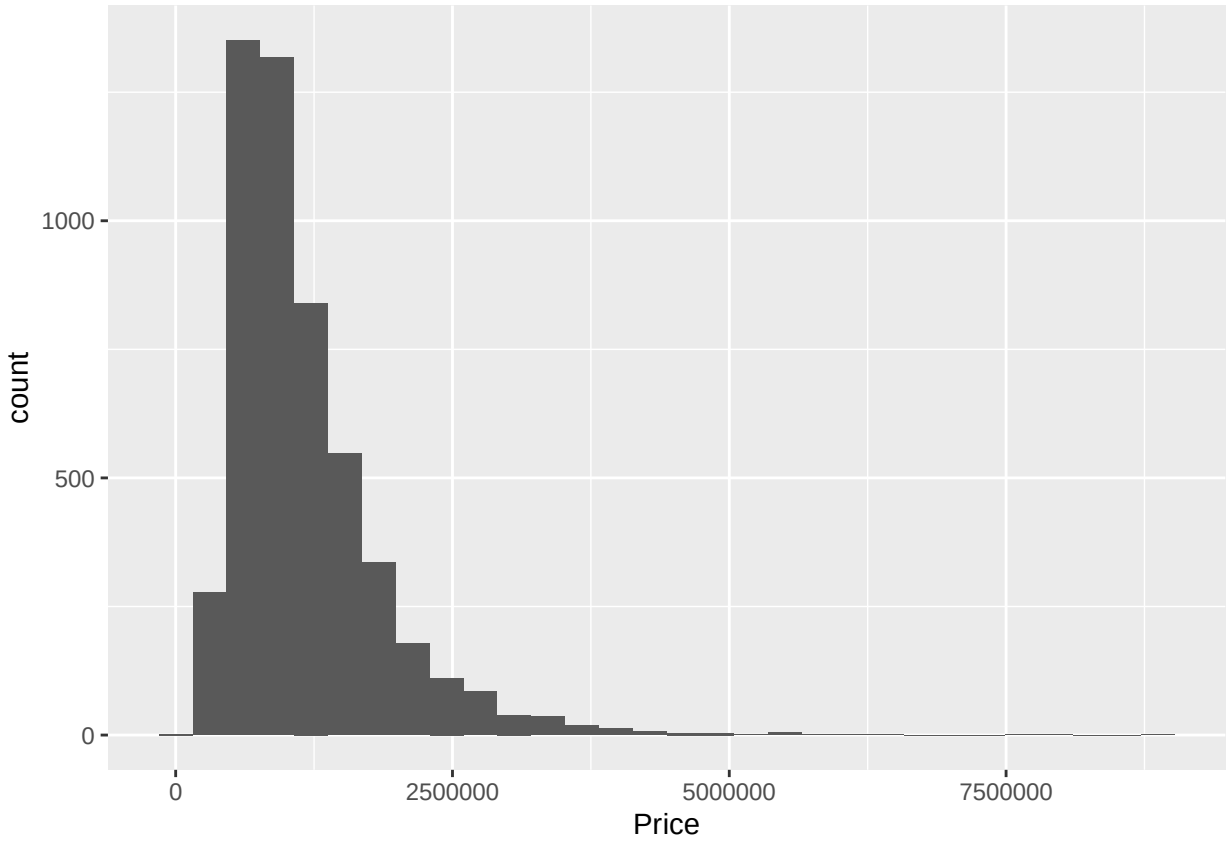
2.2 Dependent Variable Analysis - Price

We will now explore the `Price` variable.

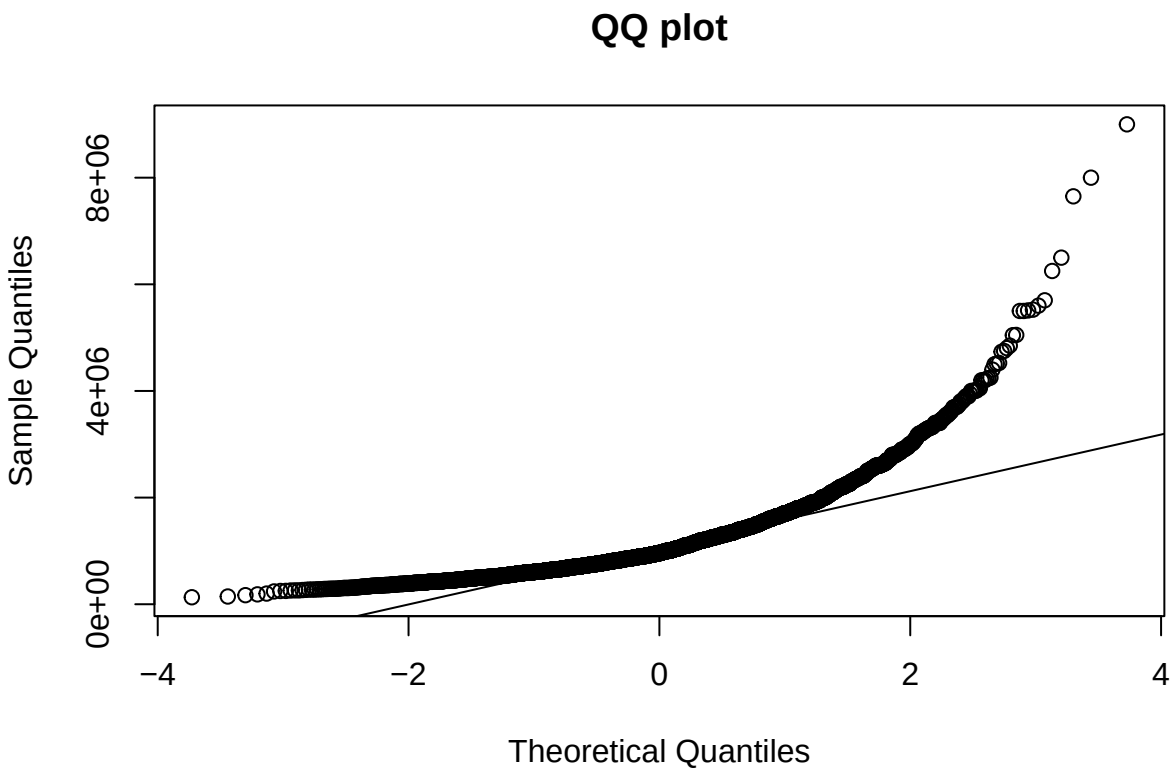
```
summary(melb$Price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  131000  700000  960000 1156211 1416375 9000000
```

```
ggplot(melb) + geom_histogram(aes(x = Price), bins=30)
```




```
qqnorm(melb$Price,main="QQ plot"); qqline(melb$Price)
```




```
ks.test(melb$Price, "pnorm")
```

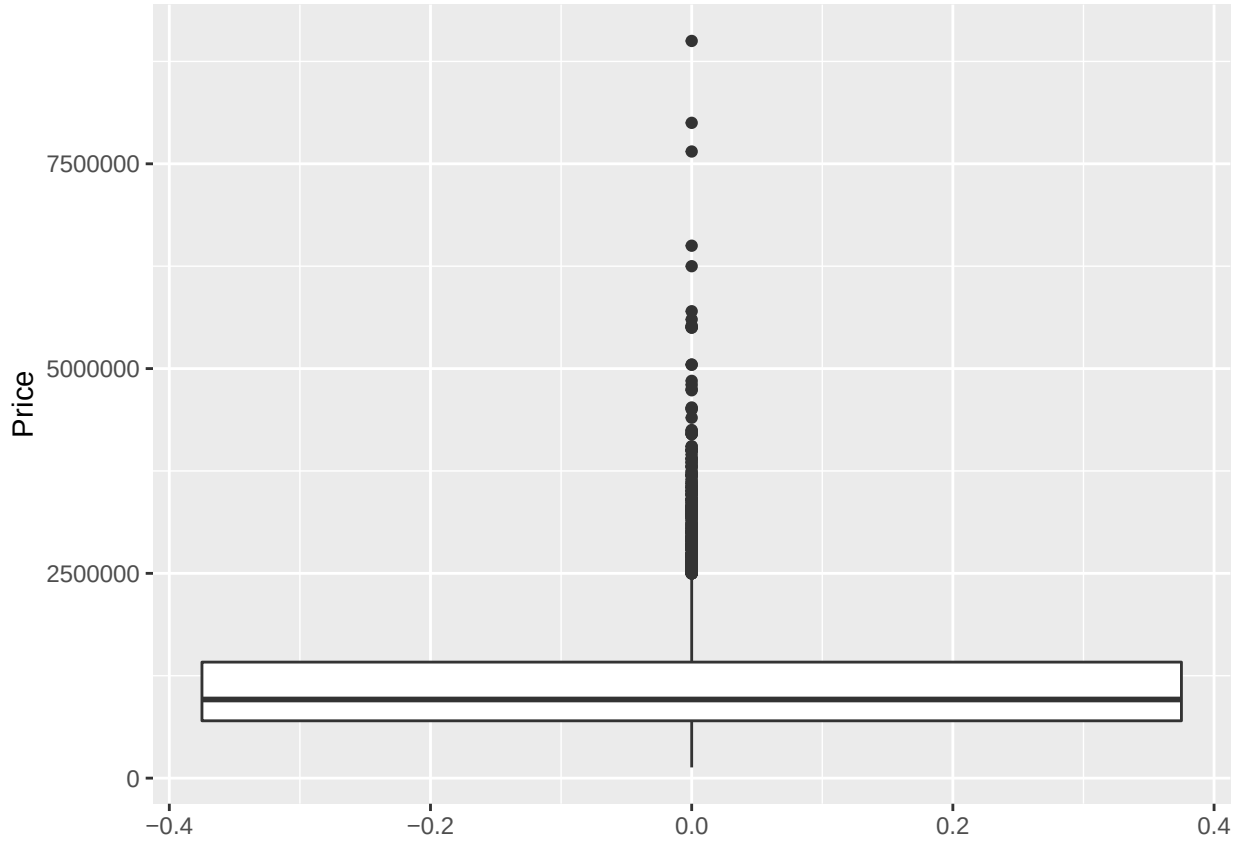
```
## Warning in ks.test(melb$Price, "pnorm"): ties should not be present for the
## Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  melb$Price
## D = 1, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

From the histogram, it can be clearly seen that the data is strongly right-skewed. It can be further observed from the quantile-quantile plot and Kolmogorov-Smirnov test with p-value of 2.2e-16 $<$ 0.05, it can be concluded that the data is not normally distributed.

This distribution may be caused due to the existence of outliers in the data.

```
ggplot(melb) + geom_boxplot(aes(y = Price))
```



It can be observed that there are outliers present in the data. These outliers will be removed to obtain a better model.

A certain price will be considered as an outlier with the following formula:

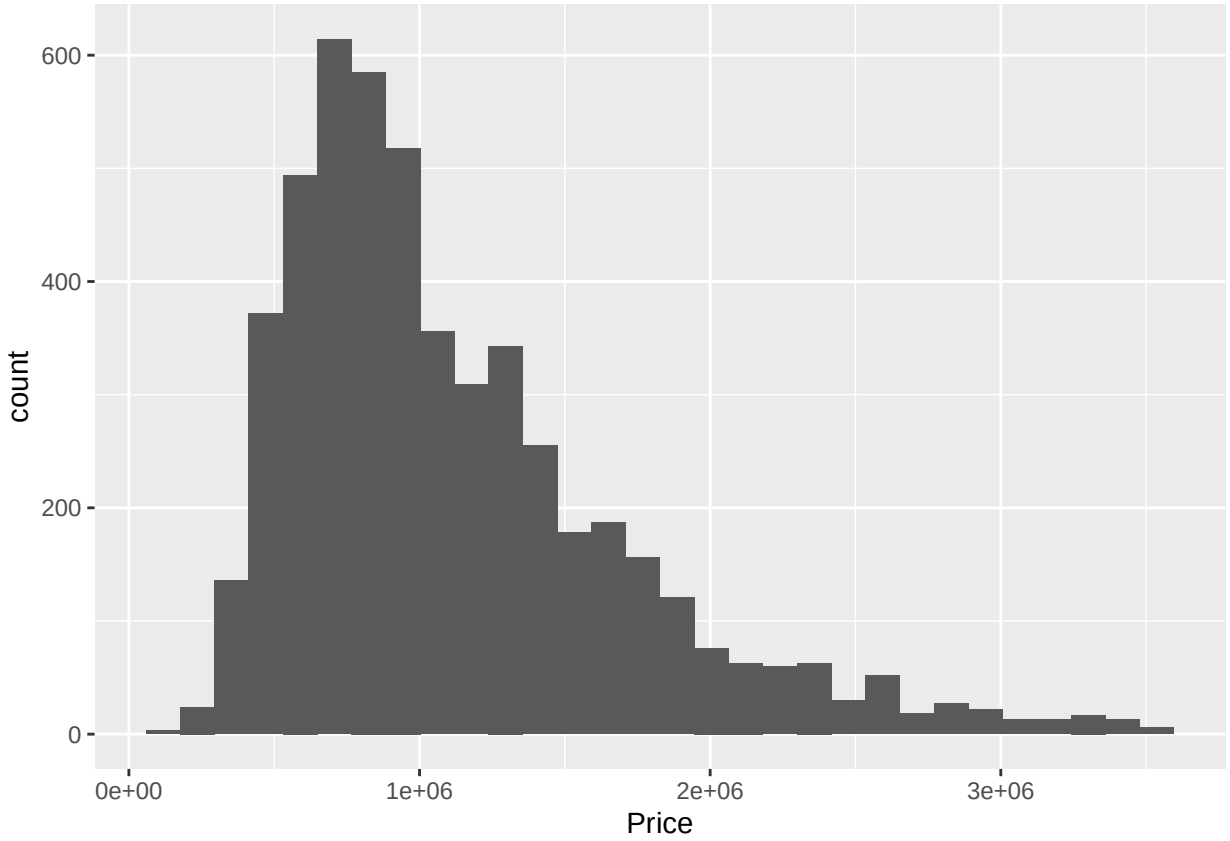
$$outliers = 3 * InterquartileRange/1.5$$

```
# Total number of outliers:
paste("outliers:", length(boxplot.stats(melb$Price, coef = 3)$out))
```

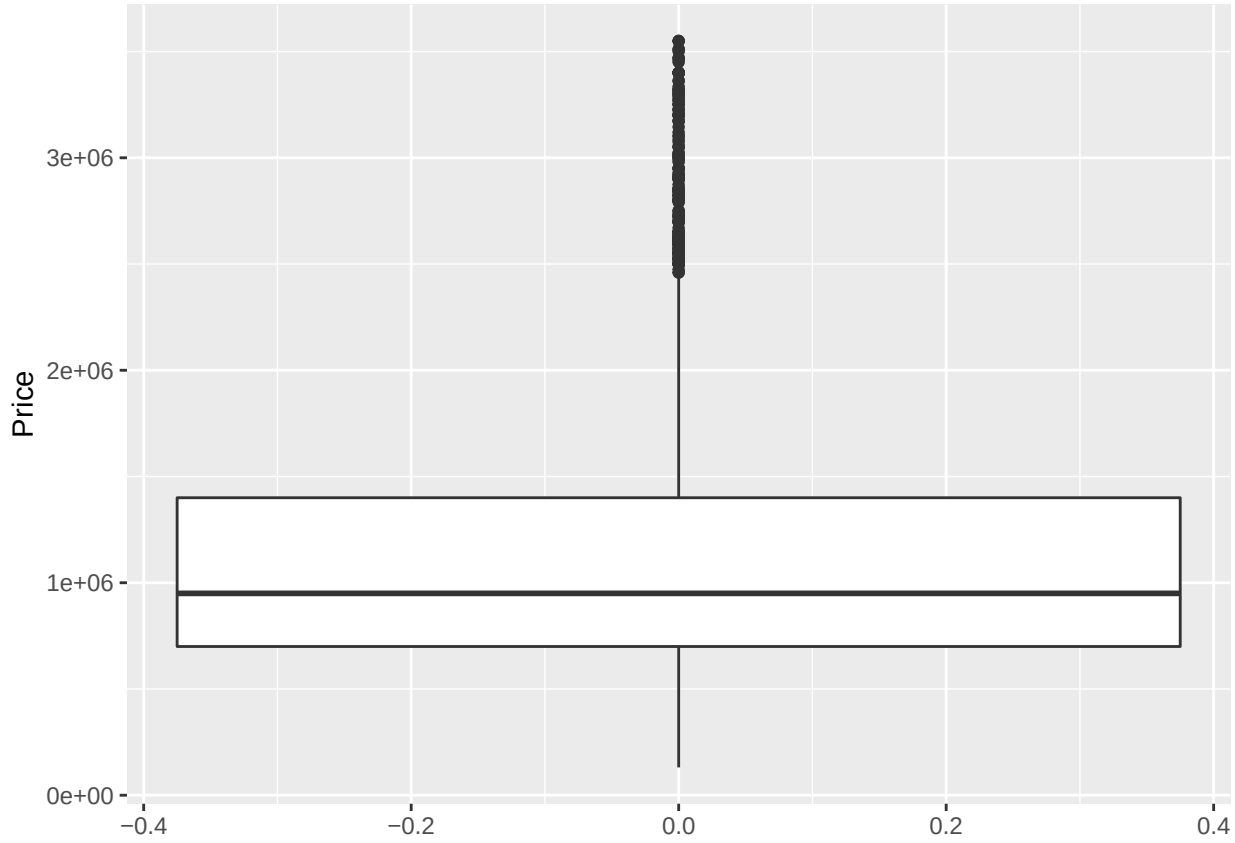
```
## [1] "outliers: 55"
```

```
out = boxplot.stats(melb$Price, coef = 3)$out
out_key = which(melb$Price %in% c(out))
melb = melb[-out_key, ]
```

```
# Checking the histogram and boxplot of the data with removed outliers
ggplot(melb) + geom_histogram(aes(x = Price), bins=30)
```




```
ggplot(melb) + geom_boxplot(aes(y = Price))
```



After the removal of the outliers, both the histogram and the boxplot are seemingly better with the distribution being less skewed.

2.3 Independent Variable Analysis - Numerical Variable

```
summary(melb$Distance)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.70    6.60    9.70   10.44   13.00   47.40
```

```
summary(melb$Landsize)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   245.8   480.0   559.8   657.0 37000.0
```

```
summary(melb$BuildingArea)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   103.0   132.0   151.3   180.0  3112.0
```

```
summary(melb$EffAge)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   19.00   52.00   52.22   86.00  100.00
```

From the summaries above, it can be concluded that there might not be irregularities in the values of the independent variables.

Next, the correlation between the independent variables towards the Price will be tested.

```
cor(melb$Price, melb$Distance)
```

```
## [1] -0.2697733
```

```
cor(melb$Price, melb$Landsize)
```

```
## [1] -0.006769088
```

```
cor(melb$Price, melb$BuildingArea)
```

```
## [1] 0.4760642
```

```
cor(melb$Price, melb$EffAge)
```

```
## [1] 0.3121283
```

From the correlation results, it can be inferred that BuildingArea has the highest correlation followed by EffAge, Distance, and Landsize. However, none of results showed a strong correlation with BuildingArea being only moderately positively correlated with the Price with a correlation value of 0.476. Furthermore, Landsize with a correlation value of -0.007 indicate that it does not affect the real estate prices.

2.4 Independent Variable Analysis - Categorical Variable

```
table(melb$Rooms)
```

```
##
##    1    2    3    4    5    6    7    8
##   99 1118 2387 1226  262   23    6    3
```

```
table(melb$Type)
```

```
##
##    h    t    u
## 3957  516  651
```

```
table(melb$Car)
```

```
##
##    0    1    2    3   >4
##  376 1938 2273  285  252
```

```
table(melb$Regionname)
```

```
##
##       Eastern Metropolitan           Eastern Victoria
##                        548                         23
##      Northern Metropolitan          Northern Victoria
##                       1518                         19
## South-Eastern Metropolitan      Southern Metropolitan
##                        151                       1610
##       Western Metropolitan           Western Victoria
##                       1241                         14
```

```
table(melb$CouncilArea)
```

```
##
##           Banyule            Bayside        Boroondara          Brimbank
##               269                188                475               190
##          Cardinia             Casey           Darebin         Frankston
##                 5                 16                381                30
##         Glen Eira Greater Dandenong       Hobsons Bay             Hume
##               348                 21                198                94
##          Kingston              Knox    Macedon Ranges        Manningham
##               105                 42                 5               146
##       Maribyrnong         Maroondah         Melbourne            Melton
##               350                 34                117                41
##            Monash     Moonee Valley          Moreland         Nillumbik
##               163                422                574                18
##      Port Phillip       Stonnington        Whitehorse        Whittlesea
##               183                190                129                89
##           Wyndham             Yarra      Yarra Ranges
##                47                244                 10
```

From the proportions of the categorical variables, it can be observed that there are only a few data in some regions and council areas. Therefore, a stratified random sampling based on these variables can be considered in the train-test split so that these variables are well represented in the taken sample,

```
# Boxplot
lapply(categorical, function(x) ggplot(melb, aes_string(x=x, y="Price", color=x)) + geom_boxplot())
```

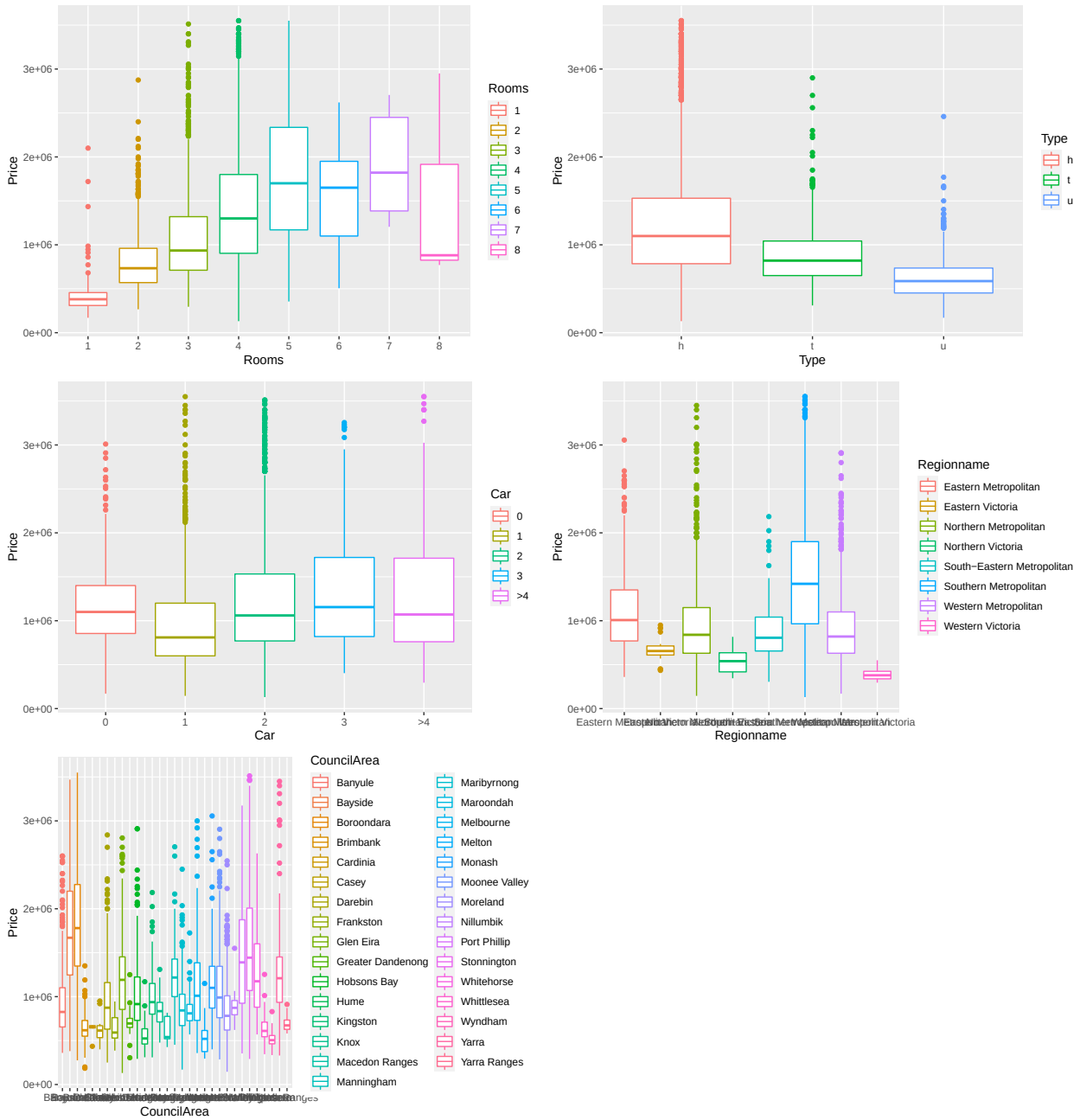
```
## [[1]]
```

```
##
## [[2]]
```

```
##
## [[3]]

##
## [[4]]

##
## [[5]]
```

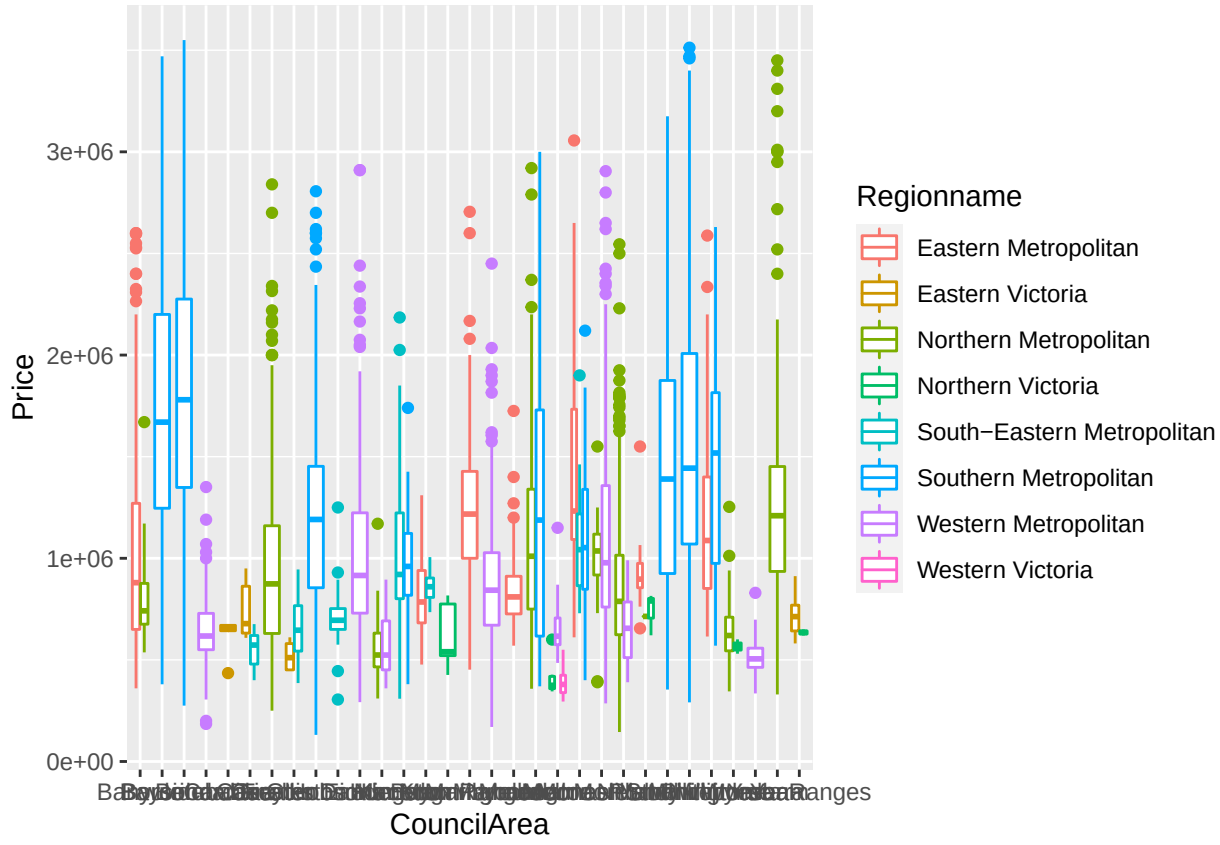


From the barplots above, there are indications of how each categorical variable relates towards the prices. It seems that as the number of rooms increases, the price increases as well. However, this only holds for until 5 total number of rooms as the boxplot fluctuates from 6-8 and only increased in 7 total number of rooms.

Next, the real estate type also seems to affect the prices with h, t, and u being the most to least expensive. For the total number of car, there seems to be no indication of total number of cars towards the price of the house. In both region name and council area, prices differ in different locations.

Since both region name and council area indicate the location of the real estate, their relationship will now be analyzed through the following boxplot.

```
ggplot(melb) + geom_boxplot(aes(x=CouncilArea, y = Price, color=Regionname))
```



---

3 Modelling - Clustering

In this section, the variables in the data will be used to create 3 clusters with k-means clustering which will be used in the modelling in the next sections. In order to use the categorical variables, they must first be transformed as numeric for the clustering to work.

```
melb_kmeans <- melb
melb_kmeans[, categorical] <- lapply(melb[, categorical], as.numeric)
summary(melb_kmeans[, categorical])
```

```
##      Rooms             Type             Car          Regionname
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:3.000   1st Qu.:1.000   1st Qu.:2.000   1st Qu.:3.000
```

```
##  Median :3.000   Median :1.000   Median :3.000   Median :6.000
##  Mean   :3.106   Mean   :1.355   Mean   :2.629   Mean   :4.769
##  3rd Qu.:4.000   3rd Qu.:1.000   3rd Qu.:3.000   3rd Qu.:6.000
##  Max.   :8.000   Max.   :3.000   Max.   :5.000   Max.   :8.000
##   CouncilArea
##  Min.   : 1.00
##  1st Qu.: 7.00
##  Median :17.00
##  Mean   :15.09
##  3rd Qu.:23.00
##  Max.   :31.00
```

Now, the data is ready to be used in k-means clustering. The parameter will be used is nstart which specifies the initial centroid in the beginning. It will be set to the same amount of clusters that are going to be created.

3.1 With Regionname and CouncilArea

```
mod_cluster = kmeans(melb_kmeans, 3, nstart=3)

# Saving Clusters into a Variable
clusters1 = as.factor(mod_cluster$cluster)
```

The following are the centroids' location in the final iteration.

```
mod_cluster$centers
```

```
##      Rooms     Type     Price  Distance      Car Landsize BuildingArea
## 1 3.376562 1.092802 1435553.7  8.746520 2.666270 497.8531     169.0529
## 2 3.982646 1.021692 2487892.4  8.219306 3.034707 661.4620     256.5871
## 3 2.817572 1.553991  730317.7 11.734474 2.545272 578.9209     124.9408
##   CouncilArea Regionname   EffAge
## 1    15.52290   4.814396 64.66686
## 2    10.98482   5.481562 65.35141
## 3    15.47317   4.633803 43.17136
```

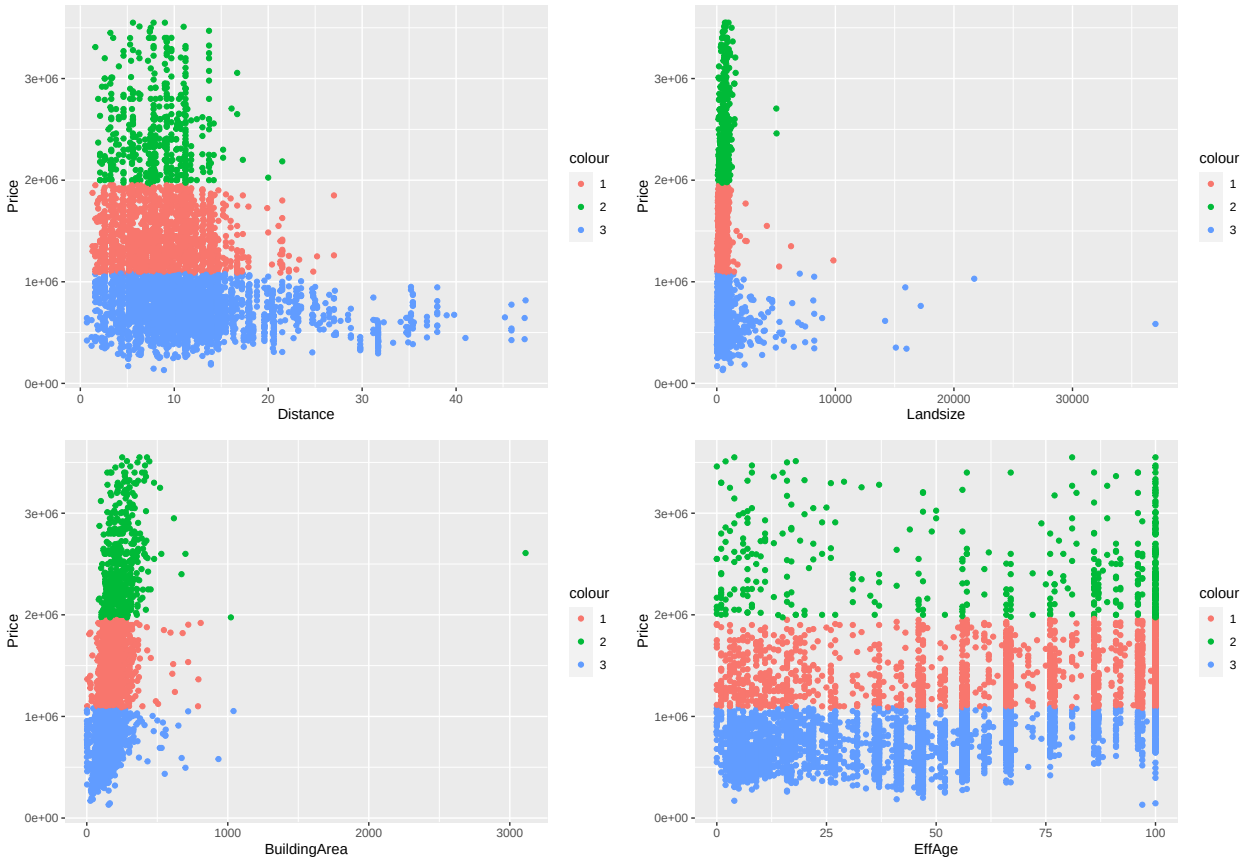
```
lapply(numerical, function(x) ggplot(melb, aes_string(x=x, y="Price", color=clusters1)) + geom_point())
```

```
## [[1]]

##
## [[2]]

##
## [[3]]

##
## [[4]]
```




```
lapply(categorical, function(x) ggplot(melb, aes_string(x=x, y="Price", color=clusters1)) + geom_point()
```

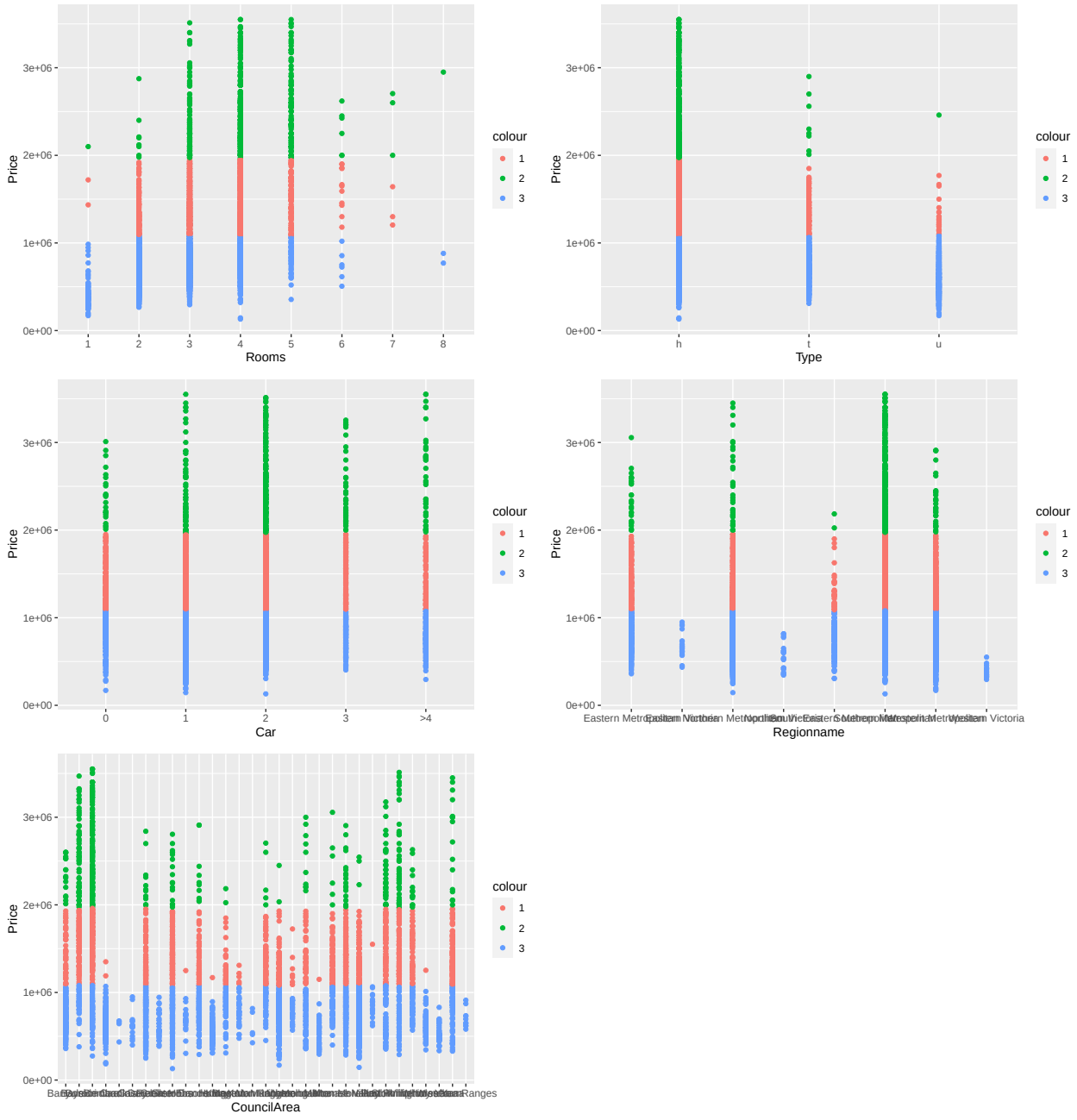
```
## [[1]]

##
## [[2]]

##
## [[3]]

##
## [[4]]

##
## [[5]]
```



3.2 Without `Regionname` and `CouncilArea`

```
mod_cluster = kmeans(melb_kmeans[, !names(melb_kmeans) %in% c("Regionname", "CouncilArea")],
                     3, nstart=3)

# Saving clusters into a Variable
clusters2 = as.factor(mod_cluster$cluster)
```

The following are the centroids' location in the final iteration.

```
mod_cluster$centers
```

```
##      Rooms     Type     Price  Distance      Car Landsize BuildingArea   EffAge
## 1 2.817572 1.553991  730317.7 11.734474 2.545272 578.9209     124.9408 43.17136
## 2 3.376562 1.092802 1435553.7  8.746520 2.666270 497.8531     169.0529 64.66686
## 3 3.982646 1.021692 2487892.4  8.219306 3.034707 661.4620     256.5871 65.35141
```

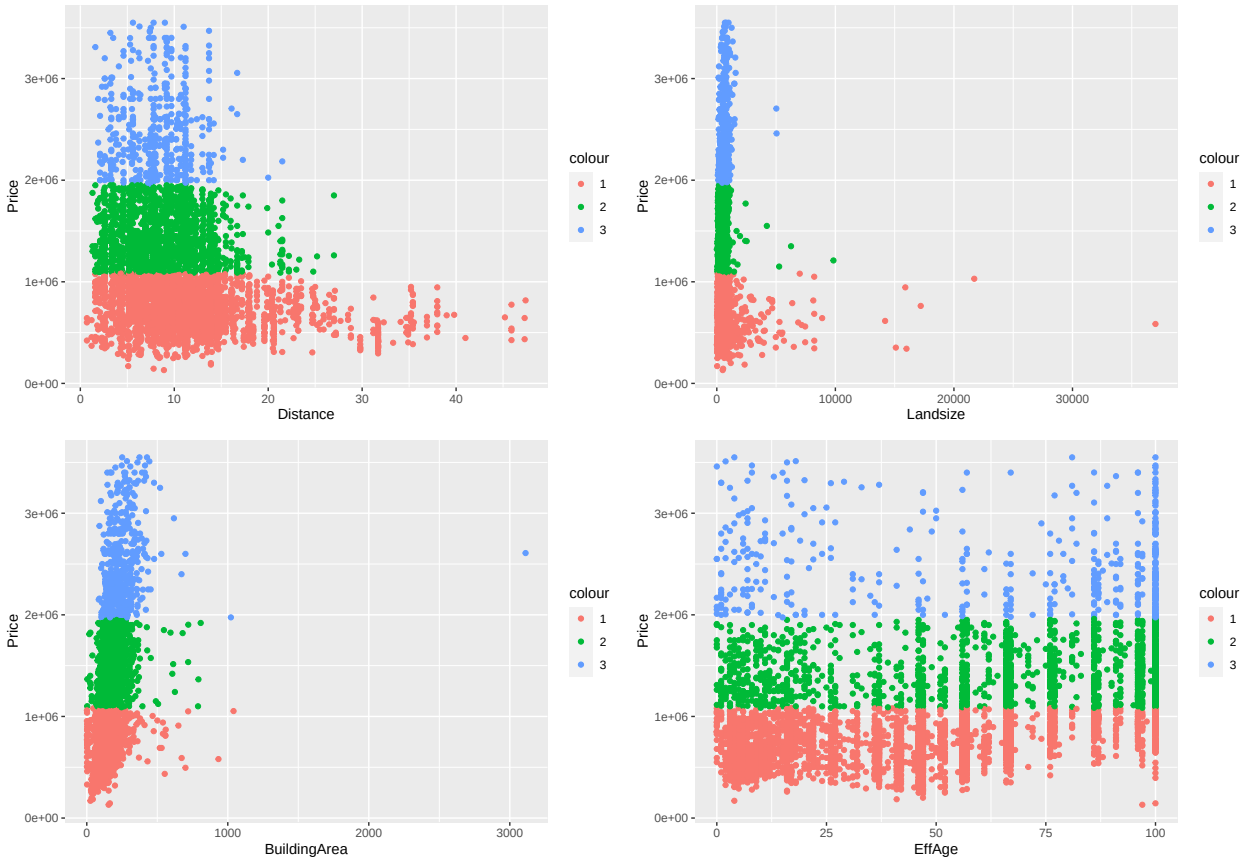
```
lapply(numerical, function(x) ggplot(melb, aes_string(x=x, y="Price", color=clusters2)) + geom_point())
```

```
## [[1]]

##
## [[2]]

##
## [[3]]

##
## [[4]]
```




```
lapply(categorical, function(x) ggplot(melb, aes_string(x=x, y="Price", color=clusters2)) + geom_point()
```

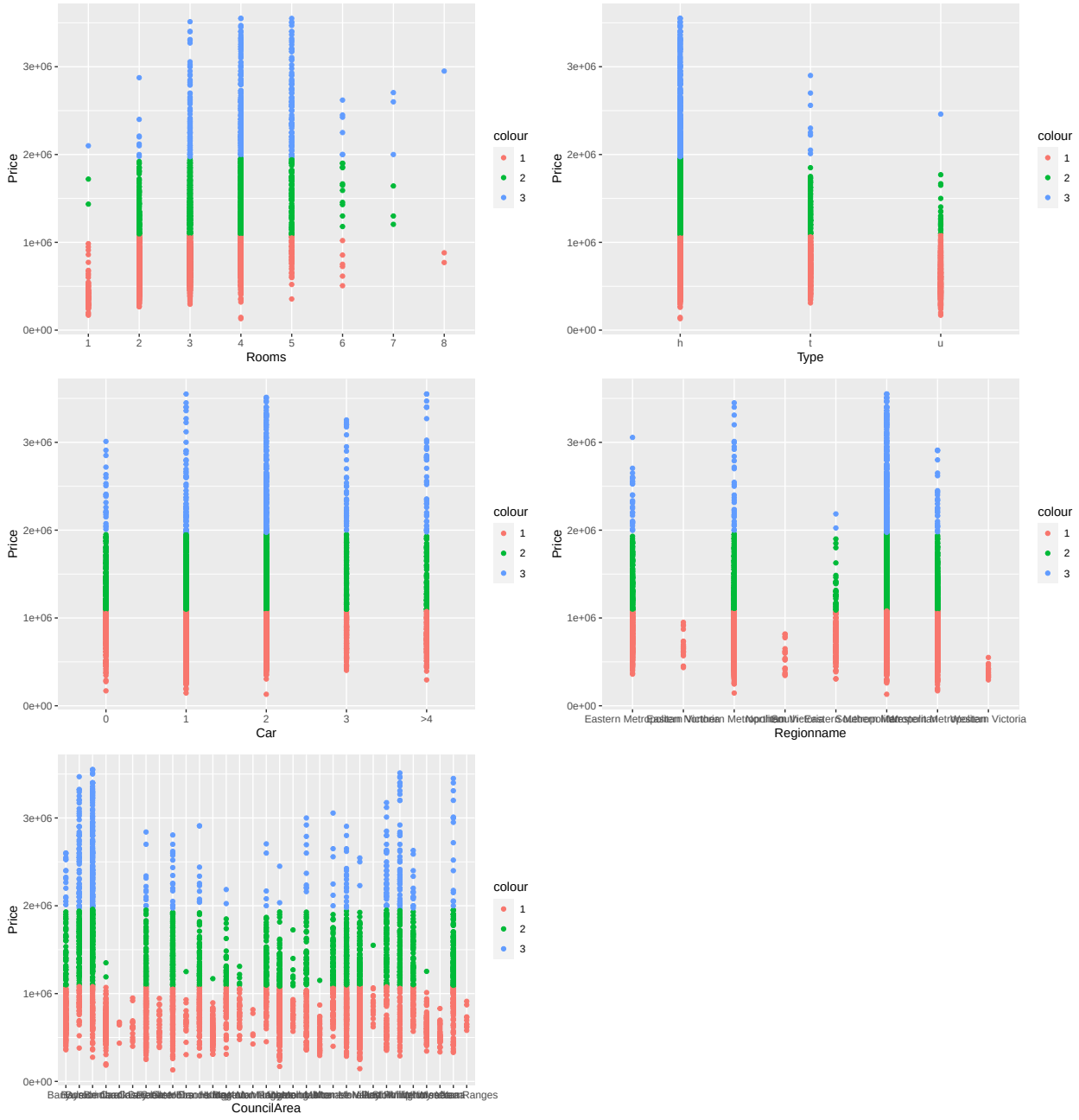
```
## [[1]]
```

```
##
## [[2]]

##
## [[3]]

##
## [[4]]

##
## [[5]]
```


Price
Rooms
Type
Car
Regionname
CouncilArea
colour
1
2
3
0e+00
1e+06
2e+06
3e+06
1
2
3
4
5
6
7
8
h
t
u
0
1
2
3
>4

3.3 Cluster Results

```
summary(clusters1)
```

```
##    1    2    3
## 1681  461 2982
```

```
summary(clusters2)
```

```
##    1    2    3
## 2982 1681  461
```

From both clustering models, the data is divided into three clusters with the same amount of data. However, the order of the clusters is changed. From the plots of the relationship between each independent variable, price, and cluster, the results are similar with the clusters only dividing based on the price. It could be concluded that both models result in the same clusters with different order.

The cluster results of the first model will be kept into the data set as a variable which will be used in other models for prediction. The reason for the choice of this cluster is due to the order of clusters from 1 to 3 which corresponds to how high the price is as well.

```
melb$Cluster <- clusters1
summary(melb$Cluster)
```

```
##    1    2    3
## 1681  461 2982
```

In total, there are 2982, 1681, and 461 data in Clusters 1 to 3.

---

4 Train-Test Split

The splitting will be done using `CreateDataPartition()` function from the caret package. The train-test split ratio will be 80:20 and stratified splitting will be done based on the clusters made. Additionally, the train-test set must also include some regions and council areas with a few entries.

```
set.seed(1)
train.idx <- createDataPartition(y = melb$Cluster, p = 0.8, list = FALSE)
train <- melb[train.idx, ]
test <- melb[-train.idx, ]
```

```
# Checking the proportion of the Cluster categorical variable
# in the train and test set in comparison to the original data set
rbind("Data Set" = prop.table(table(melb$Cluster)),
      "Train" = prop.table(table(train$Cluster)),
      "Test" = prop.table(table(test$Cluster)))
```

```
##                  1         2         3
## Data Set 0.3280640 0.08996877 0.5819672
## Train    0.3280488 0.09000000 0.5819512
## Test     0.3281250 0.08984375 0.5820312
```

```
# Checking the proportion of the region name categorical variable
# in the train and test set in comparison to the original data set
rbind("Data Set" = prop.table(table(melb$Regionname)),
      "Train" = prop.table(table(train$Regionname)),
      "Test" = prop.table(table(test$Regionname)))
```

```
##          Eastern Metropolitan Eastern Victoria Northern Metropolitan
## Data Set            0.1069477      0.004488681             0.2962529
## Train               0.1041463      0.004878049             0.2992683
## Test                0.1181641      0.002929688             0.2841797
##          Northern Victoria South-Eastern Metropolitan Southern Metropolitan
## Data Set       0.003708041                 0.02946916             0.3142077
## Train          0.003902439                 0.02902439             0.3143902
## Test           0.002929688                 0.03125000             0.3134766
##          Western Metropolitan Western Victoria
## Data Set            0.2421936     0.0027322404
## Train               0.2412195     0.0031707317
## Test                0.2460938     0.0009765625
```

```
# Checking the proportion of the council area categorical variable
# in the train and test set in comparison to the original data set
rbind("Data Set" = prop.table(table(melb$CouncilArea)),
      "Train" = prop.table(table(train$CouncilArea)),
      "Test" = prop.table(table(test$CouncilArea)))
```

```
##             Banyule    Bayside Boroondara   Brimbank     Cardinia       Casey
## Data Set 0.05249805 0.03669009 0.09270101 0.03708041 0.0009758002 0.003122560
## Train    0.04951220 0.03585366 0.09536585 0.03585366 0.0012195122 0.003170732
## Test     0.06445312 0.04003906 0.08203125 0.04199219 0.0000000000 0.002929688
##             Darebin   Frankston  Glen Eira Greater Dandenong Hobsons Bay
## Data Set 0.07435597 0.005854801 0.06791569       0.004098361  0.03864169
## Train    0.07487805 0.005121951 0.06731707       0.004146341  0.03829268
## Test     0.07226562 0.008789062 0.07031250       0.003906250  0.04003906
##                Hume   Kingston        Knox Macedon Ranges Manningham
## Data Set 0.01834504 0.02049180 0.008196721   0.0009758002 0.02849336
## Train    0.01853659 0.02048780 0.007560976   0.0012195122 0.02804878
## Test     0.01757812 0.02050781 0.010742188   0.0000000000 0.03027344
##          Maribyrnong   Maroondah  Melbourne      Melton     Monash
## Data Set  0.06830601 0.006635441 0.02283372 0.008001561 0.03181109
## Train     0.07000000 0.007317073 0.02292683 0.008780488 0.03292683
## Test      0.06152344 0.003906250 0.02246094 0.004882812 0.02734375
##          Moonee Valley  Moreland   Nillumbik Port Phillip Stonnington
## Data Set    0.08235753 0.1120219 0.003512881   0.03571429  0.03708041
## Train       0.08195122 0.1148780 0.003414634   0.03682927  0.03439024
## Test        0.08398438 0.1005859 0.003906250   0.03125000  0.04785156
##          Whitehorse Whittlesea     Wyndham      Yarra Yarra Ranges
## Data Set 0.02517564 0.01736924 0.009172521 0.04761905 0.0019516003
## Train    0.02414634 0.01756098 0.009268293 0.04682927 0.0021951220
## Test     0.02929688 0.01660156 0.008789062 0.05078125 0.0009765625
```

From the stratified random sampling, the train-test sample obtained did not include the Macedon Ranges council area. Some other proportions both in the region and council names differ from the original proportion. However, the sampling has quite represented the overall proportions of the region and council area

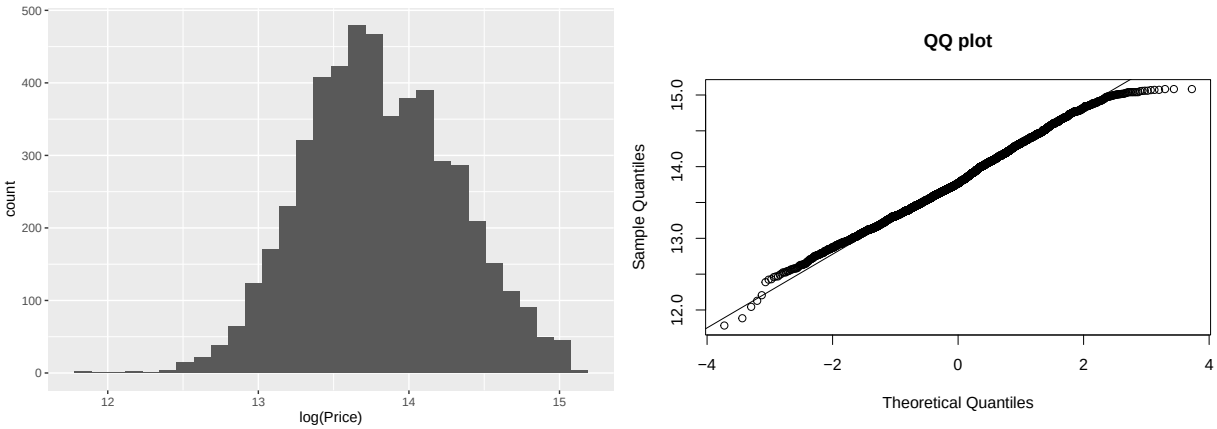
---

5 Preparation For Modelling

5.1 Prices Normality

From the data exploration, it was found that the distribution of the data is strongly skewed, even after removal of outliers. Therefore, a log transformation towards the `Price` variable will be attempted in order to make the distribution of the `Price` variable resemble a normal distribution even more.

```
ggplot(melb) + geom_histogram(aes(x = log(Price)), bins=30)
qqnorm(log(melb$Price),main="QQ plot"); qqline(log(melb$Price))
```



On both the histogram and qqplot, the log(Price) seems to follow a normal distribution. Therefore, this transformation will be used in future modelling and this may result in a better outcome during model fitting.

5.2 Modelling Functions

```
options(scipen=999) # Disable Scientific Notation

# Plots relationships between Log Sale Price, Prediction, and Residuals on the train set
# Function also returns a bin which contains indices of extreme residual data
eval.train_init <- function(model, train, outliers){
  zresid = data.frame(x=rstandard(model))
  title = ifelse(outliers == TRUE, "Outliers not Removed", "Outliers Removed")

  print(ggarrange(
    ggplot() + geom_point(aes(x=model$fitted.values, y=log(train$Price))) +
      geom_abline(aes(intercept = 0, slope = 1), colour = "blue") +
      ggtitle(paste("Log SalePrice vs Prediction - Training Set,", title)) +
      theme(plot.title = element_text(hjust = 0.5)) +
```

```
      labs(x = "Prediction on Train Data", y ="Log Sales Price"),
    ggplot() + geom_point(aes(x=model$fitted.values, y=zresid$x)) +
      geom_abline(aes(intercept = 0, slope = 0), colour = "blue") +
      ggtitle(paste("Residual vs Prediction - Training Set,", title)) +
      theme(plot.title = element_text(hjust = 0.5)) +
      labs(x = "Prediction on Train Data", y ="Residual"),
    nrow = 2, align = "v"))

  if(outliers == TRUE){
    bin = which(abs(zresid)>3)
    return(bin)
  }
}

# Updates train data by removing residuals in the model.
eval.train_update <- function(model, train, bin){
  if(length(bin)>0) {
    train.outliers = train
    train.outliers$outliers = 0
    train.outliers$outliers[bin] = 1
    train.outliers$pred = model$fitted.values
    train.outliers$pred.dollar = exp(train.outliers$pred)
    train.2 = train[-bin,]
  } else {
    train.2 = train
  }

  return(train.2)
}

# Plots relationships between Log Sale Price and Prediction, and Residuals in the test set
# Function also returns the comp function for predictions in the test set
# (with respect to normal Price, not log(Price))
eval.test <- function(pred, obs){
  # Residual for the test set (must be scaled)
  zresid.test = scale(pred - log(obs))

  print(ggarrange(
    ggplot() + geom_point(aes(x=pred, y=log(obs))) +
        geom_abline(aes(intercept = 0, slope = 1), colour = "blue") +
        ggtitle("Log SalePrice vs Prediction - Testing Set") +
        theme(plot.title = element_text(hjust = 0.5)) +
        labs(x = "Prediction on Test Data", y ="Log Sales Price"),
    ggplot() + geom_point(aes(x=pred, y=zresid.test)) +
      geom_abline(aes(intercept = 0, slope = 0), colour = "blue") +
      ggtitle("Residual vs Prediction - Testing Set") +
      theme(plot.title = element_text(hjust = 0.5)) +
      labs(x = "Prediction on Test Data", y ="Residual"),
    nrow = 2, align = "v"))

  comp.test = comp(exp(pred), obs)
  print(comp.test)
```

```
  return(comp.test)
}

# Compare predicted and observed data
# Function returns a list of metrics used for comparison
comp <- function(pred, obs){
  n = length(obs)
  rsq = cor(pred,obs)^2
  mse = sum((pred - obs)^2)/n
  semse = sd((pred - obs)^2) / sqrt(n)
  rmse = sqrt(mse)
  se = sd(pred-obs) / sqrt(n)
  mae = sum(abs(pred-obs))/n
  mape = sum(abs(pred-obs)/obs)/n*100
  return(list("n"=n,"R2"=rsq,"MSE"=mse,"SEMSE"=semse,"RMSE"=rmse,"SE"=se,"MAE"=mae,"MAPE"=mape))
}
```

---

6 Modelling - GAM

Generalized Additive Model (GAM) is a model that assumes that the explanatory variable may not have a linear relationship with the response variable. For each independent variable, GAM will create a spline instead of only weighting them linearly in Generalized Linear Model (GLM). GAM creates partitions for a certain variable and within each partition, GAM fits a function for that suits the respective variable and continues doing so in the next partition, in which the functions combined as a result is called a spline. GAM does the process of fitting splines automatically. However, splines will not be created for categorical variables.

In the `gam` library in R, we can determine a variable to be fitted with a spline by using the `s()` function during the modelling.

6.1 Model Assumptions

The GAM model assumes that there are no interactions between independent variables. Therefore, an independent variable must not be a linear combination of other independent variables. This will be evaluated using inner variance inflation factor (VIF) which measures multicollinearity of independent variables.

6.2 Modelling Preparation

As previously mentioned, we will assume that all numerical variables do not have a linear relationship with the price and add the `s()` for the modelling.

```
# Preparing outcome and independent variables for the models
# We also transform the outcome with logarithm transformation.
outcome <- "log(Price)"
s.numerical <- paste("s(", numerical, ")", sep = "")

(f <- as.formula(paste(outcome,
                       paste(c(s.numerical, categorical), collapse = "+"),
                       sep = "~")))
```

```
## log(Price) ~ s(Distance) + s(Landsize) + s(BuildingArea) + s(EffAge) +
##     Rooms + Type + Car + Regionname + CouncilArea
```

```
(f2 <- as.formula(paste(outcome,
                        paste(c(s.numerical, categorical, "Cluster"), collapse = "+"),
                        sep = "~")))
```

```
## log(Price) ~ s(Distance) + s(Landsize) + s(BuildingArea) + s(EffAge) +
##     Rooms + Type + Car + Regionname + CouncilArea + Cluster
```

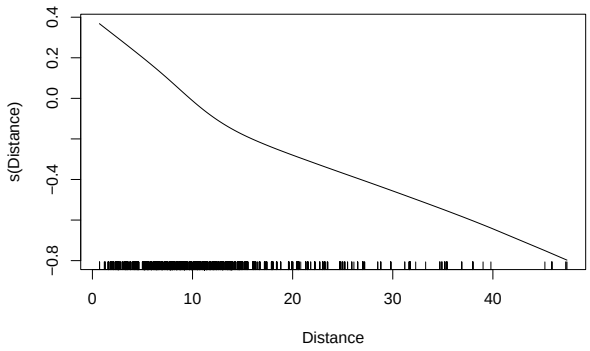
6.3 GAM Without Clusters

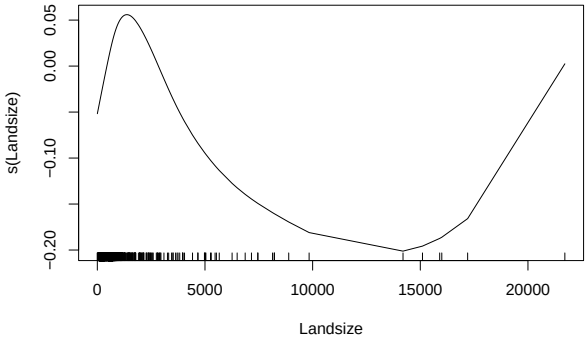
```
mod_gam1 <- gam(f, data=train)
summary(mod_gam1)
```

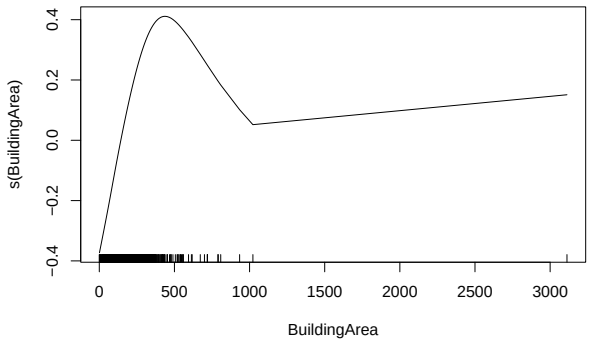
```
##
## Call: gam(formula = f, data = train)
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -2.575033 -0.126449 -0.003049  0.128376  1.171266
##
## (Dispersion Parameter for gaussian family taken to be 0.0469)
##
##     Null Deviance: 1003.513 on 4099 degrees of freedom
## Residual Deviance: 189.0734 on 4033 degrees of freedom
## AIC: -842.7924
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                   Df  Sum Sq Mean Sq  F value                Pr(>F)
## s(Distance)        1  91.613  91.613 1954.138 < 0.00000000000000022 ***
## s(Landsize)        1   0.958   0.958   20.435           0.000006346 ***
## s(BuildingArea)    1 221.638 221.638 4727.607 < 0.00000000000000022 ***
## s(EffAge)          1  77.592  77.592 1655.063 < 0.00000000000000022 ***
## Rooms              7  66.910   9.559  203.888 < 0.00000000000000022 ***
## Type               2   6.422   3.211   68.497 < 0.00000000000000022 ***
## Car                4   9.136   2.284   48.719 < 0.00000000000000022 ***
## Regionname         7 148.472  21.210  452.424 < 0.00000000000000022 ***
## CouncilArea       30  43.512   1.450   30.938 < 0.00000000000000022 ***
## Residuals       4033 189.073   0.047
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                  Npar Df  Npar F                 Pr(F)
## (Intercept)
## s(Distance)            3  24.467   0.00000000000000111 ***
## s(Landsize)            3  30.640 < 0.00000000000000022 ***
## s(BuildingArea)        3 185.346 < 0.00000000000000022 ***
## s(EffAge)              3  49.066 < 0.00000000000000022 ***
```

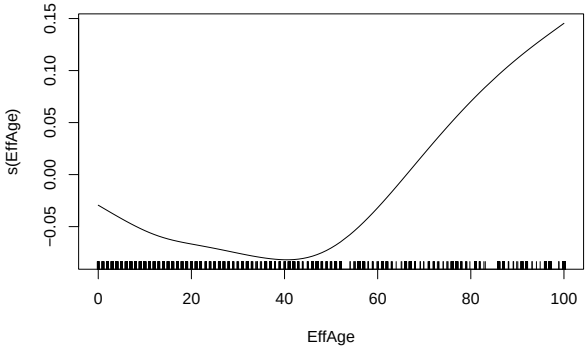
```
## Rooms
## Type
## Car
## Regionname
## CouncilArea
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

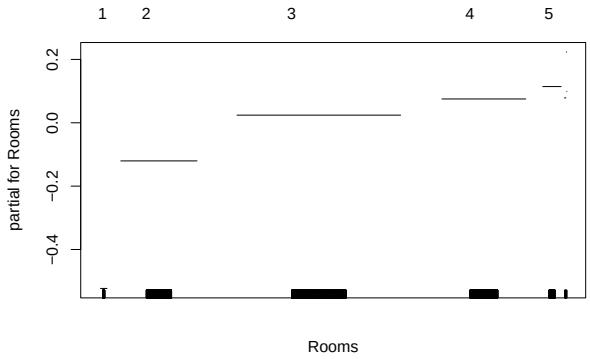
```
plot(mod_gam1)
```

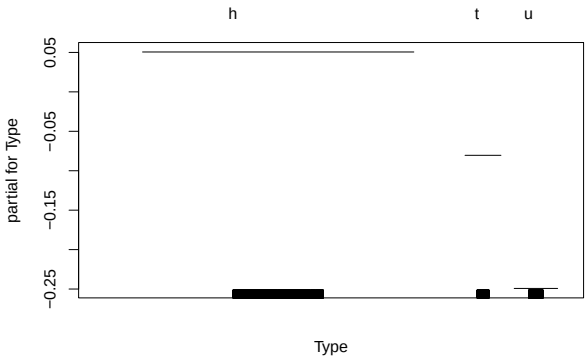


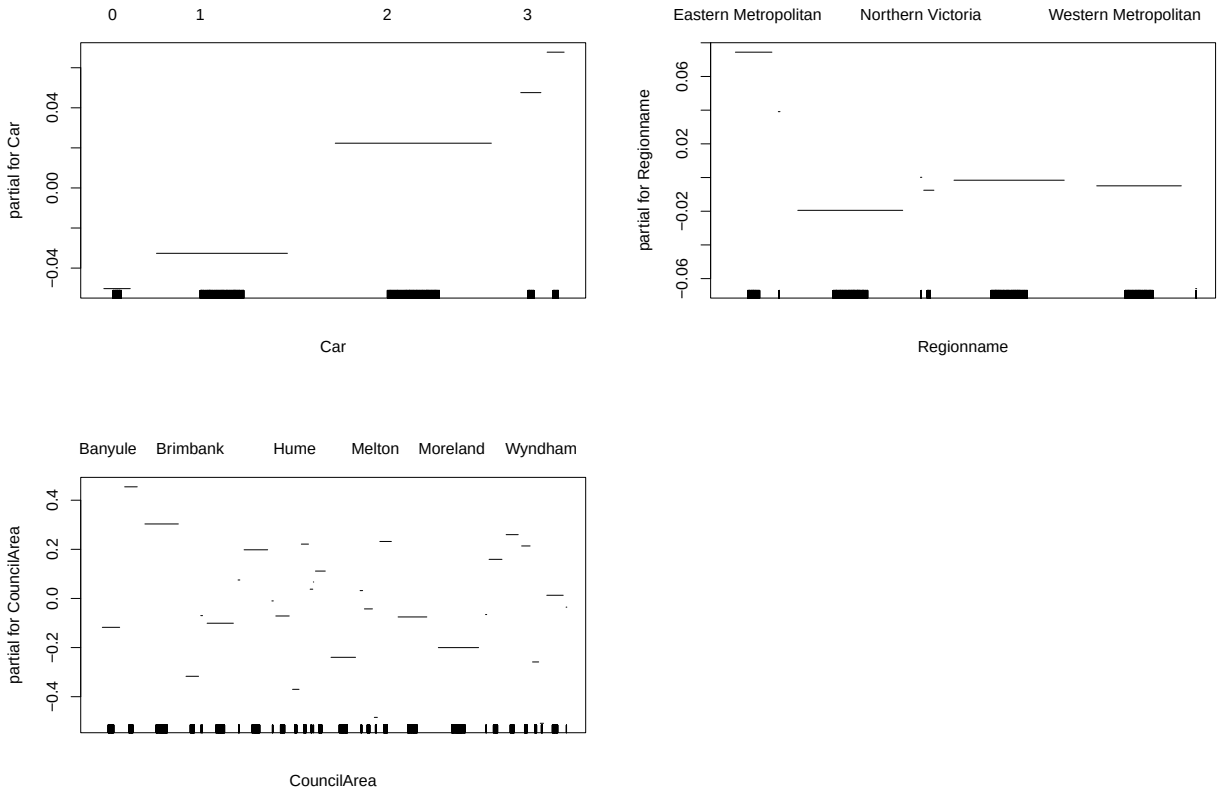












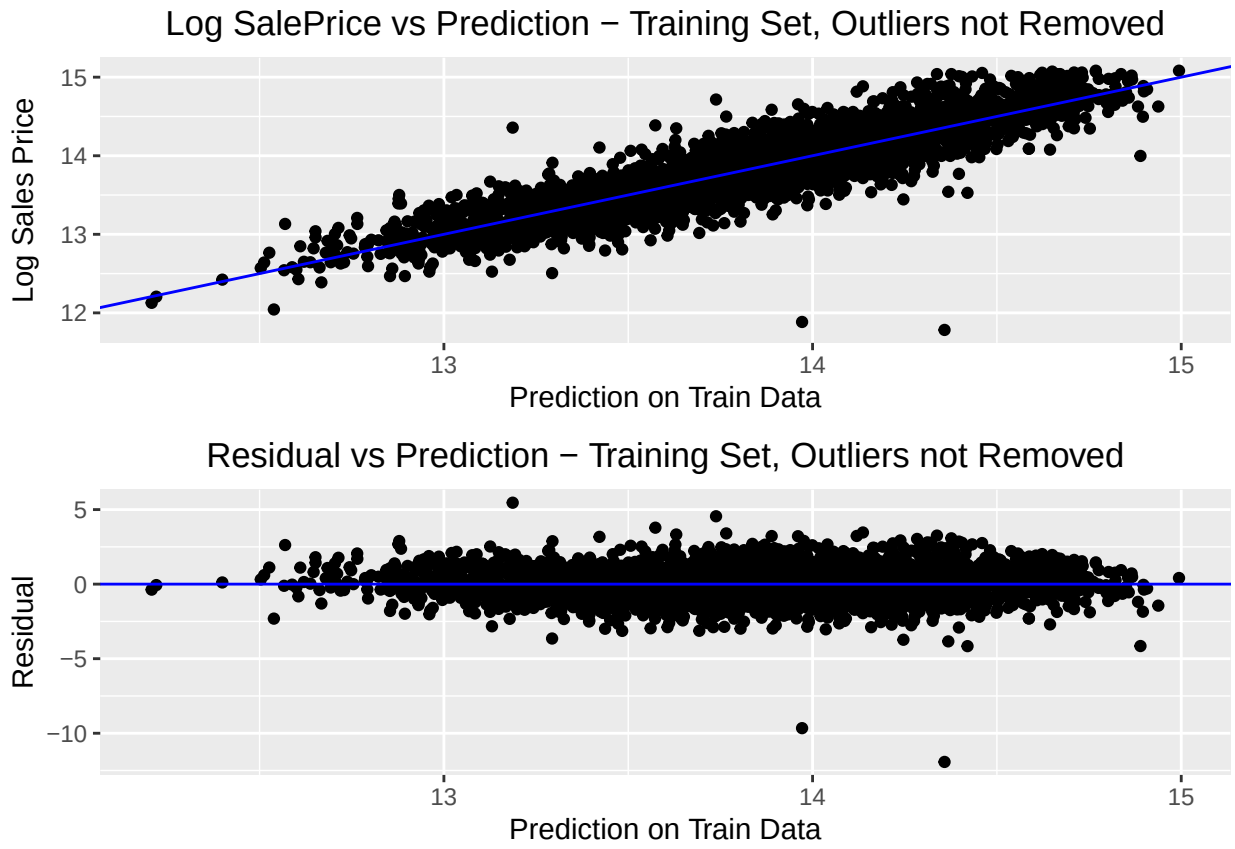
From the spline plots above, almost all of the numerical independent variables are visibly nonlinear. This can be observed through the spline plot lines that are curvy and have one or more inflection points such Landsize, BuildingArea, and EffAge. Although Distance does not seem to have an inflection point on the spline plot, it is still visible that it does not have a straight linear line.

6.3.1 Model Evaluation 1 - AIC & Residual Analysis

```
mod_gam1$aic
```

```
## [1] -842.7924
```

```
bin_gam = eval.train_init(mod_gam1, train, outliers = TRUE)
```




```
train_gam = eval.train_update(mod_gam1, train, bin_gam)
```

In the model's summary, that the model received an AIC value of -857.1206, which will be used to compare with the next model with removed outliers in the train set.

From the graph above, we can see from the first graph that the predictions made by the model are spread around the y=x line, which shows that the predictions are close to the actual values. However, we can see that there are a couple of wrong predictions. The second graph highlights these wrong outcomes as we can see that there are some residuals that are far from the predictions. Residuals are the difference between each predicted data and the actual value of the data. The standardized residuals with the `rstandard()` function will be evaluated and data whose corresponding residual values are greater than 3 will be removed in the following step.

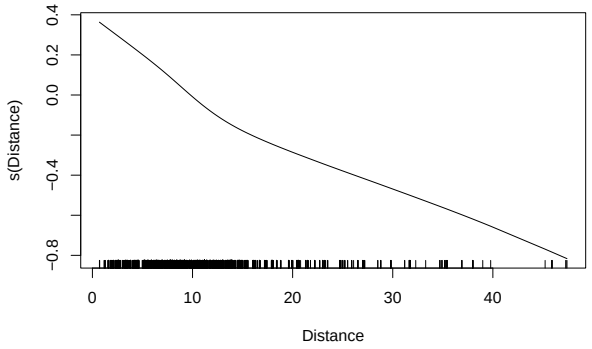
6.3.2 Remodelling With New Train Set

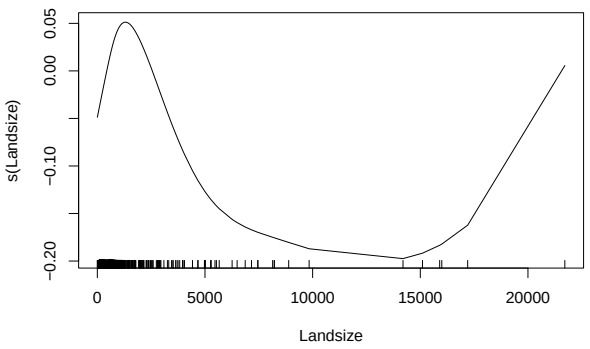
```
mod_gam1.final <- gam(f, data = train_gam)
summary(mod_gam1.final)
```

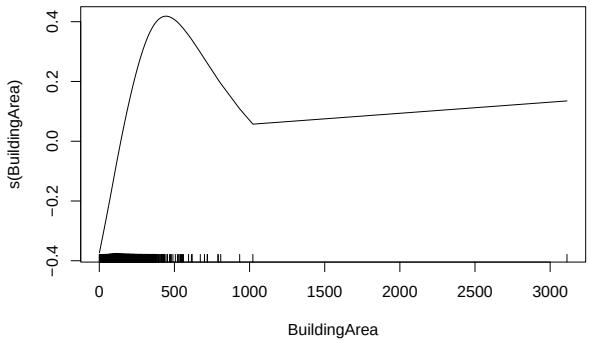
```
##
## Call: gam(formula = f, data = train_gam)
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -0.641093 -0.126236 -0.003961  0.127495  0.631001
##
```

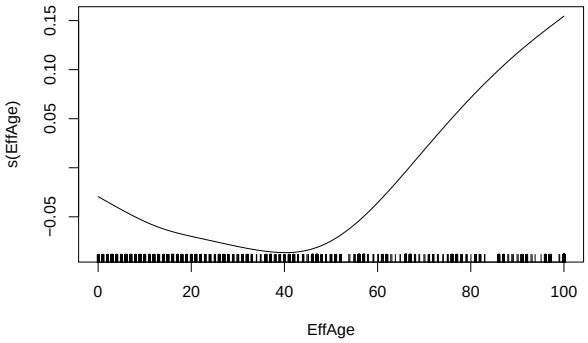
```
## (Dispersion Parameter for gaussian family taken to be 0.0412)
##
##     Null Deviance: 981.7732 on 4076 degrees of freedom
## Residual Deviance: 165.061 on 4010 degrees of freedom
## AIC: -1368.104
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                   Df  Sum Sq Mean Sq  F value                Pr(>F)
## s(Distance)        1  89.076  89.076 2164.024 < 0.00000000000000022 ***
## s(Landsize)        1   1.125   1.125   27.325          0.0000001808 ***
## s(BuildingArea)    1 221.984 221.984 5392.883 < 0.00000000000000022 ***
## s(EffAge)          1  80.236  80.236 1949.262 < 0.00000000000000022 ***
## Rooms              7  67.310   9.616  233.604 < 0.00000000000000022 ***
## Type               2   5.762   2.881   69.995 < 0.00000000000000022 ***
## Car                4   8.960   2.240   54.420 < 0.00000000000000022 ***
## Regionname         7 149.889  21.413  520.202 < 0.00000000000000022 ***
## CouncilArea       30  41.374   1.379   33.505 < 0.00000000000000022 ***
## Residuals       4010 165.061   0.041
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                  Npar Df  Npar F                  Pr(F)
## (Intercept)
## s(Distance)           3  25.406  0.000000000000000222 ***
## s(Landsize)           3  34.494 < 0.00000000000000022 ***
## s(BuildingArea)       3 208.905 < 0.00000000000000022 ***
## s(EffAge)             3  63.023 < 0.00000000000000022 ***
## Rooms
## Type
## Car
## Regionname
## CouncilArea
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

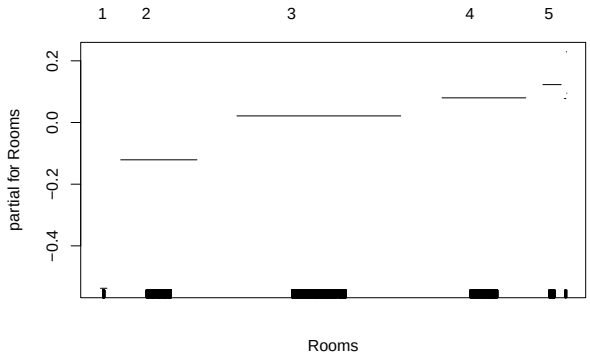
```
plot(mod_gam1.final)
```

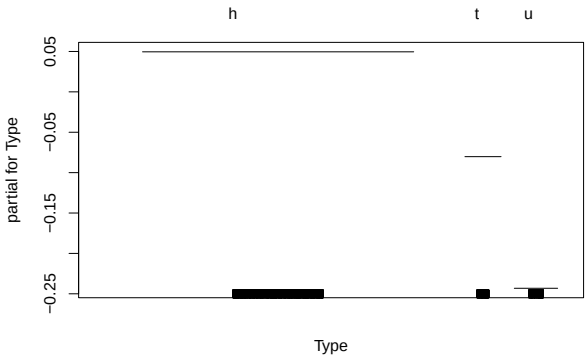


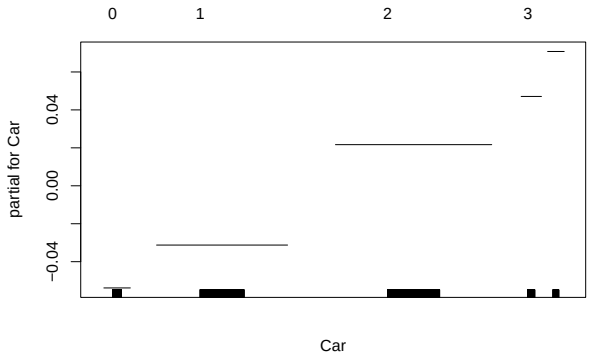


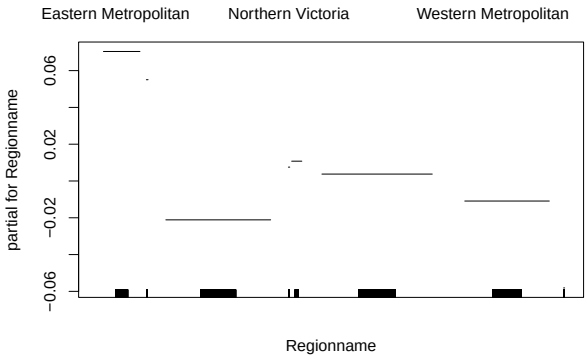

s(BuildingArea)
0.4
0.2
0.0
−0.2
−0.4
0
500
1000
1500
2000
2500
3000
BuildingArea

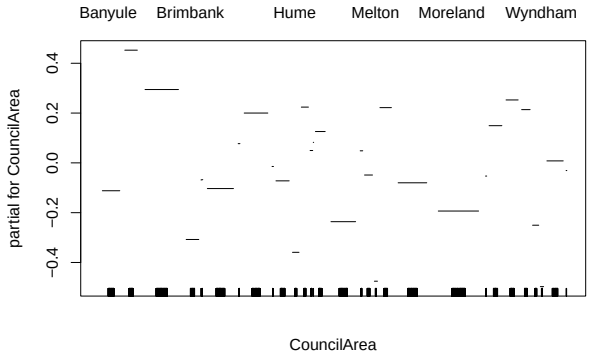

s(EffAge)
0.15
0.10
0.05
−0.05
0
20
40
60
80
100
EffAge


1
2
3
4
5
partial for Rooms
0.2
0.0
−0.2
−0.4
Rooms


h
t
u
partial for Type
0.05
−0.05
−0.15
−0.25
Type


0
1
2
3
partial for Car
0.04
0.00
−0.04
Car


Eastern Metropolitan
Northern Victoria
Western Metropolitan
partial for Regionname
0.06
0.02
−0.02
−0.06
Regionname



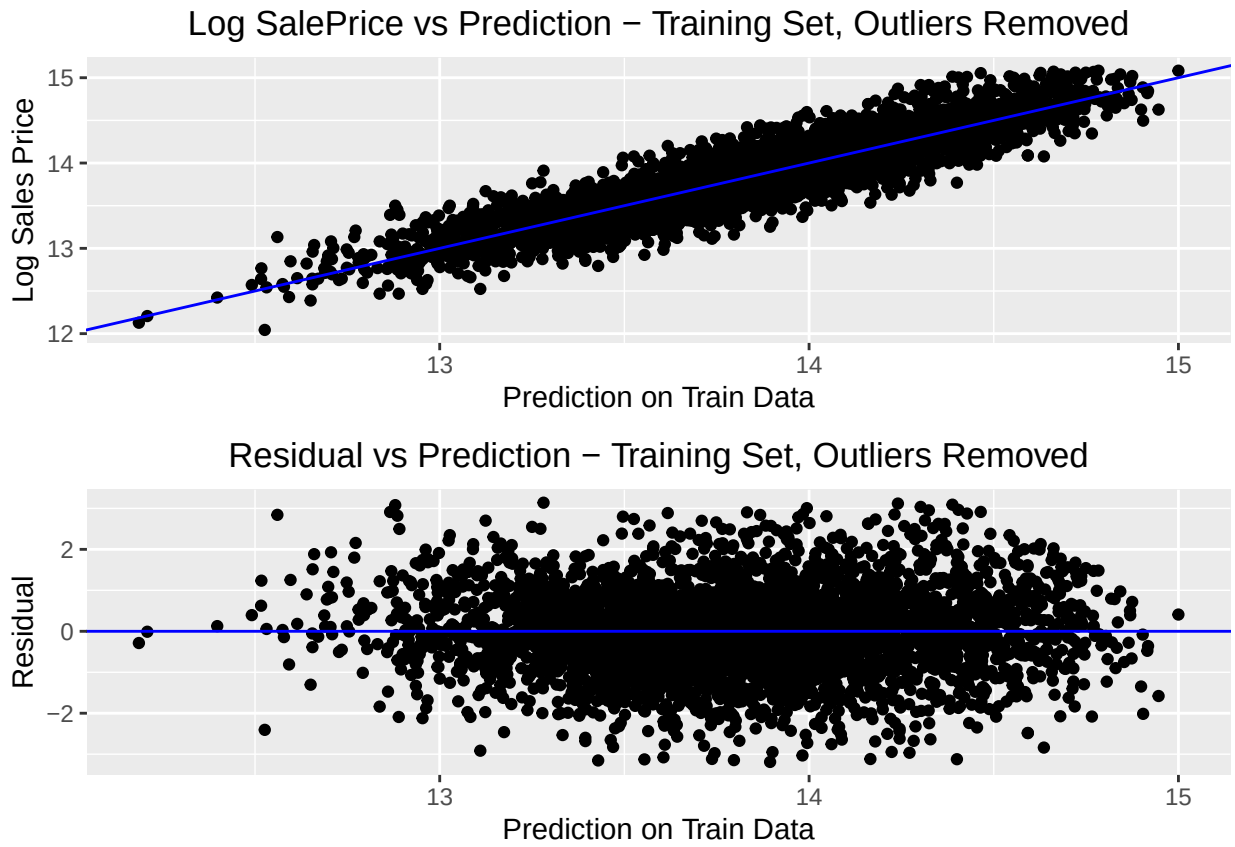
These spline plots as a result of fitting with the outliers removed are consistent with the previous spline plots.

6.3.3 Model Evaluation 2 - New AIC, Residual Analysis, and Error Analysis

```
data.frame("Outliers Not Removed" = mod_gam1$aic,
           "Outliers Removed" = mod_gam1.final$aic)
```

```
##   Outliers.Not.Removed Outliers.Removed
## 1            -842.7924        -1368.104
```

```
eval.train_init(mod_gam1.final, train_gam, outliers = FALSE)
```

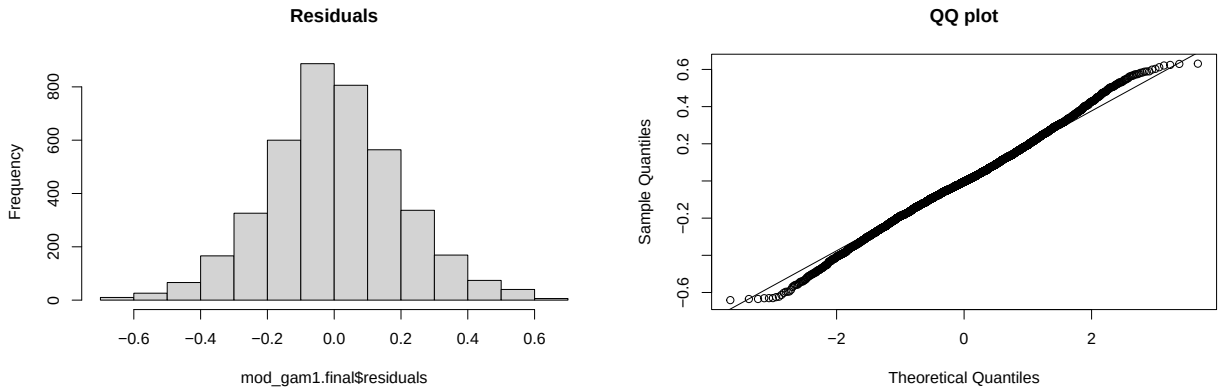


After we have removed the outliers in the training set, we can see that there is a highly significant decrease in the AIC of the model. This means that the model is now performing much better with outliers removed.

From the graphs, it can be observed that the predictions are now closer the actual values and the residuals are also closer to 0. It is also worth noting that there are no significant patterns present in the residuals vs prediction plot; the residuals are randomly distributed.

Furthermore, the normality of the model's residuals will be analyzed.

```
hist(mod_gam1.final$residuals, main="Residuals")
qqnorm(mod_gam1.final$residuals, main="QQ plot"); qqline(mod_gam1.final$residuals)
```



It can also be observed from the histogram and qqplot that the residuals resemble a normal distribution.

Now, we are going to analyze the error in the predictions of the train set. We will also compare the metrics with the previous model with outliers not removed.

```
merge(stack(comp(exp(mod_gam1$fitted.values), exp(mod_gam1$y))),
      stack(comp(exp(mod_gam1.final$fitted.values), exp(mod_gam1.final$y))),
      by = "ind", sort = FALSE)
```

```
##     ind            values.x            values.y
## 1     n        4100.0000000        4077.0000000
## 2    R2           0.7826011           0.8003092
## 3   MSE 75758058367.1153259 68349219784.0920258
## 4 SEMSE  3285952588.4090986  2716400905.1081471
## 5  RMSE      275241.8179840      261436.8370832
## 6    SE        4274.7742434        4074.1702155
## 7   MAE      182940.0802572      177835.6084416
## 8  MAPE          16.6399279          15.8486395
```

From the results, we can conclude that out of the 4078 data in the train set with outliers removed, the model receives an RMSE of 261981.666 and an average absolute error of 15.927%. These values show an increase in performance from the previous model with outliers intact. This can be concluded from the significant decrease in RMSE and MAPE values.

6.3.4 Multicollinearity

```
vif(mod_gam1.final)
```

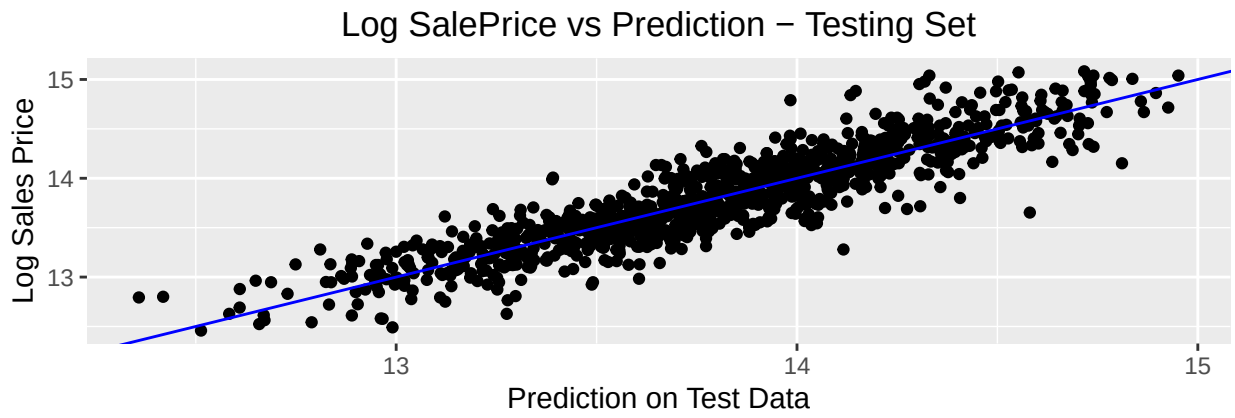
```
##                          GVIF Df GVIF^(1/(2*Df))
## s(Distance)          7.025371  1        2.650542
## s(Landsize)          1.086069  1        1.042146
## s(BuildingArea)      1.618603  1        1.272243
## s(EffAge)            1.772856  1        1.331486
## Rooms                2.458019  7        1.066348
## Type                 2.229720  2        1.221976
## Car                  1.758083  4        1.073075
## Regionname       44249.569443  7        2.147092
## CouncilArea     171347.217518 30        1.222451
```

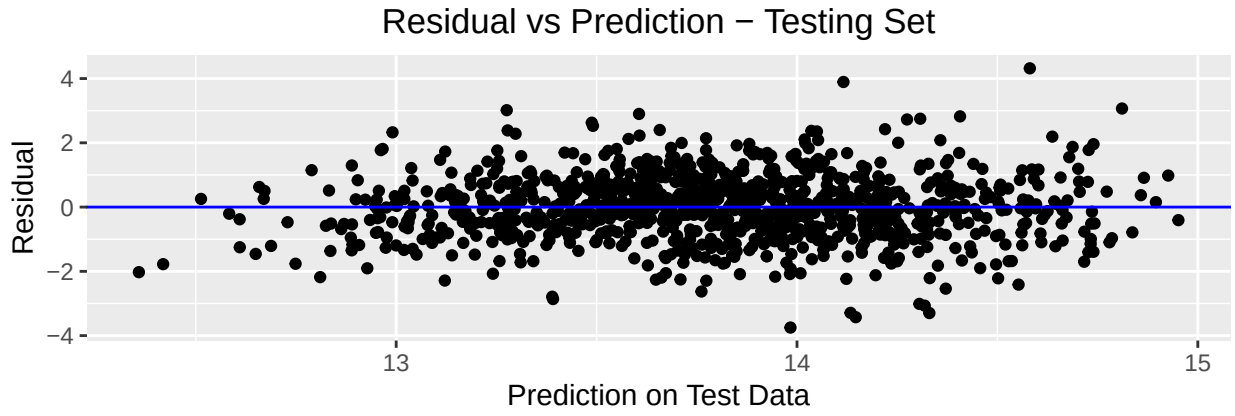
From the VIF scores, we can see that almost all numerical variables have a relatively small VIF with values below 4. Therefore, we can conclude that they are not linear combinations of other independent variables. However, Distance received a VIF of 6.675 which is relatively more than other VIF values. Since Distance do not seem to be a linear combination of other numerical variables and its VIF value does not have a relatively big difference with other VIF values, Distance can still be considered to be used in this model. On the other hand, the VIF values of the categorical variables will be ignored.

6.3.5 Prediction and Error Analysis

```
pred_gam1 = predict(mod_gam1.final, newdata = test, type = "response")
```

```
eval_gam1 = eval.test(pred_gam1, test$Price)
```






```
## $n
## [1] 1024
##
## $R2
## [1] 0.7744738
##
## $MSE
## [1] 82523687398
##
## $SEMSE
## [1] 7674597838
##
## $RMSE
## [1] 287269.4
##
## $SE
## [1] 8939.286
##
## $MAE
## [1] 188612.3
##
## $MAPE
## [1] 16.8877
```

From the graph, we can see that there are some predictions that are far away from the y=x line, which means that the model has some inaccurate predictions in the test set. The second graph further highlights these findings.

From the metrics results, we can conclude that out of the 1024 data in the test set, the model successfully predicted the prices with RMSE of 287900.1 and average absolute error of 16.71%. These metrics and the model's AIC will be kept for further comparison with the results from other models.

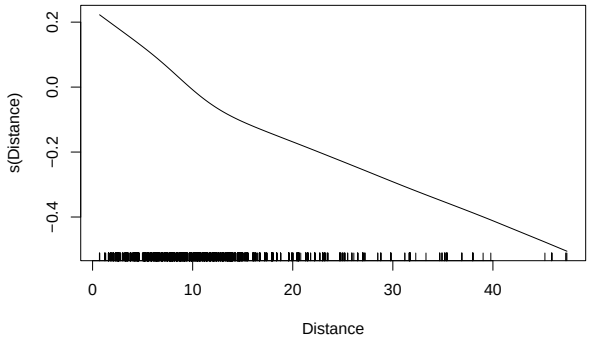
6.4 GAM With Clusters

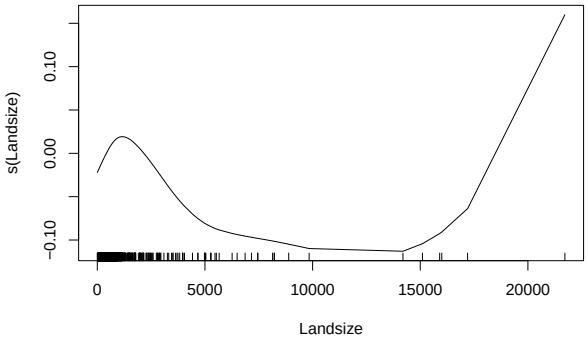
```
mod_gam2 <- gam(f2, data=train)
summary(mod_gam2)
```

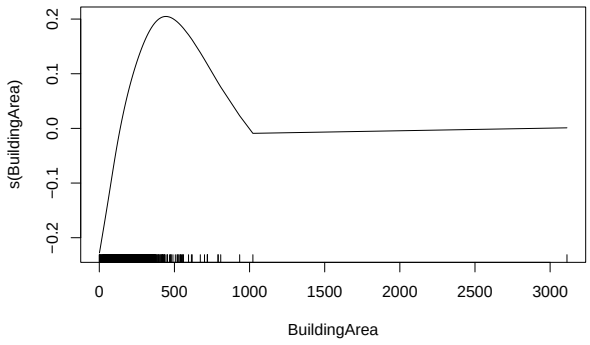
```
##
## Call: gam(formula = f2, data = train)
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -2.169878 -0.101439  0.002833  0.107932  0.865192
##
## (Dispersion Parameter for gaussian family taken to be 0.0275)
##
##     Null Deviance: 1003.513 on 4099 degrees of freedom
## Residual Deviance: 110.6999 on 4031 degrees of freedom
## AIC: -3033.571
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                   Df  Sum Sq Mean Sq  F value                Pr(>F)
## s(Distance)        1  91.281  91.281 3323.900 < 0.00000000000000022 ***
## s(Landsize)        1   1.055   1.055   38.419        0.000000000628 ***
## s(BuildingArea)    1 232.153 232.153 8453.556 < 0.00000000000000022 ***
## s(EffAge)          1  78.735  78.735 2867.034 < 0.00000000000000022 ***
## Rooms              7  88.634  12.662  461.071 < 0.00000000000000022 ***
## Type               2   9.032   4.516  164.447 < 0.00000000000000022 ***
## Car                4   9.679   2.420   88.117 < 0.00000000000000022 ***
## Regionname         7 152.807  21.830  794.894 < 0.00000000000000022 ***
## CouncilArea       30  45.799   1.527   55.590 < 0.00000000000000022 ***
## Cluster            2  90.868  45.434 1654.431 < 0.00000000000000022 ***
## Residuals       4031 110.700   0.027
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                  Npar Df Npar F                 Pr(F)
## (Intercept)
## s(Distance)           3 17.204     0.000000000042415 ***
## s(Landsize)           3 13.685     0.000000007020174 ***
## s(BuildingArea)       3 94.570 < 0.00000000000000022 ***
## s(EffAge)             3 18.658     0.000000000005122 ***
## Rooms
## Type
```

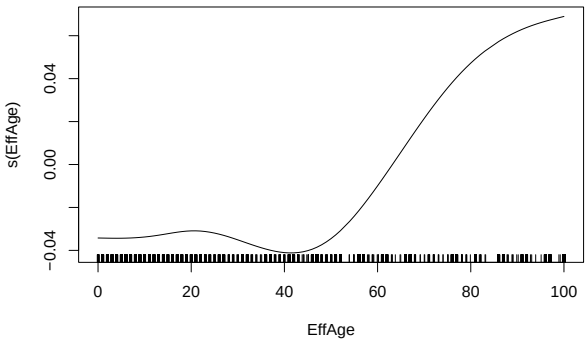
```
## Car
## Regionname
## CouncilArea
## Cluster
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

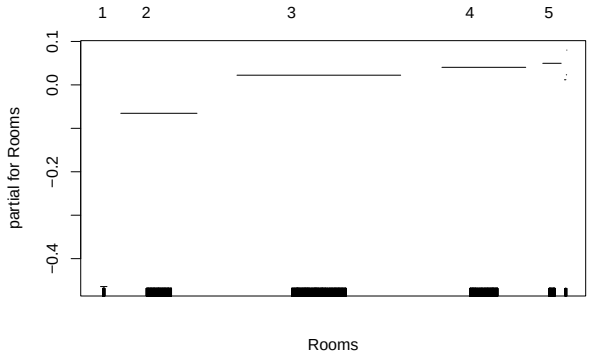
```
plot(mod_gam2)
```

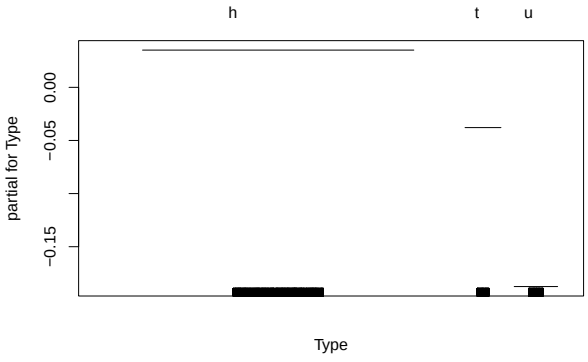


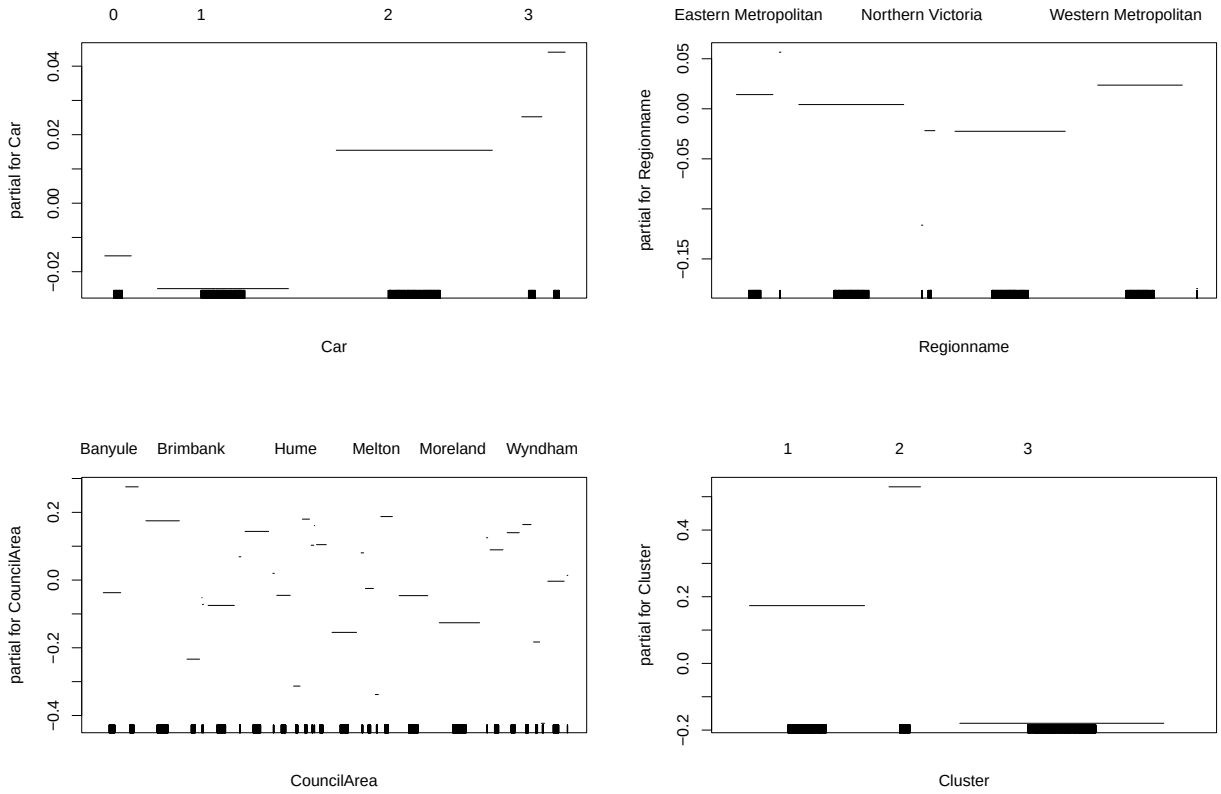












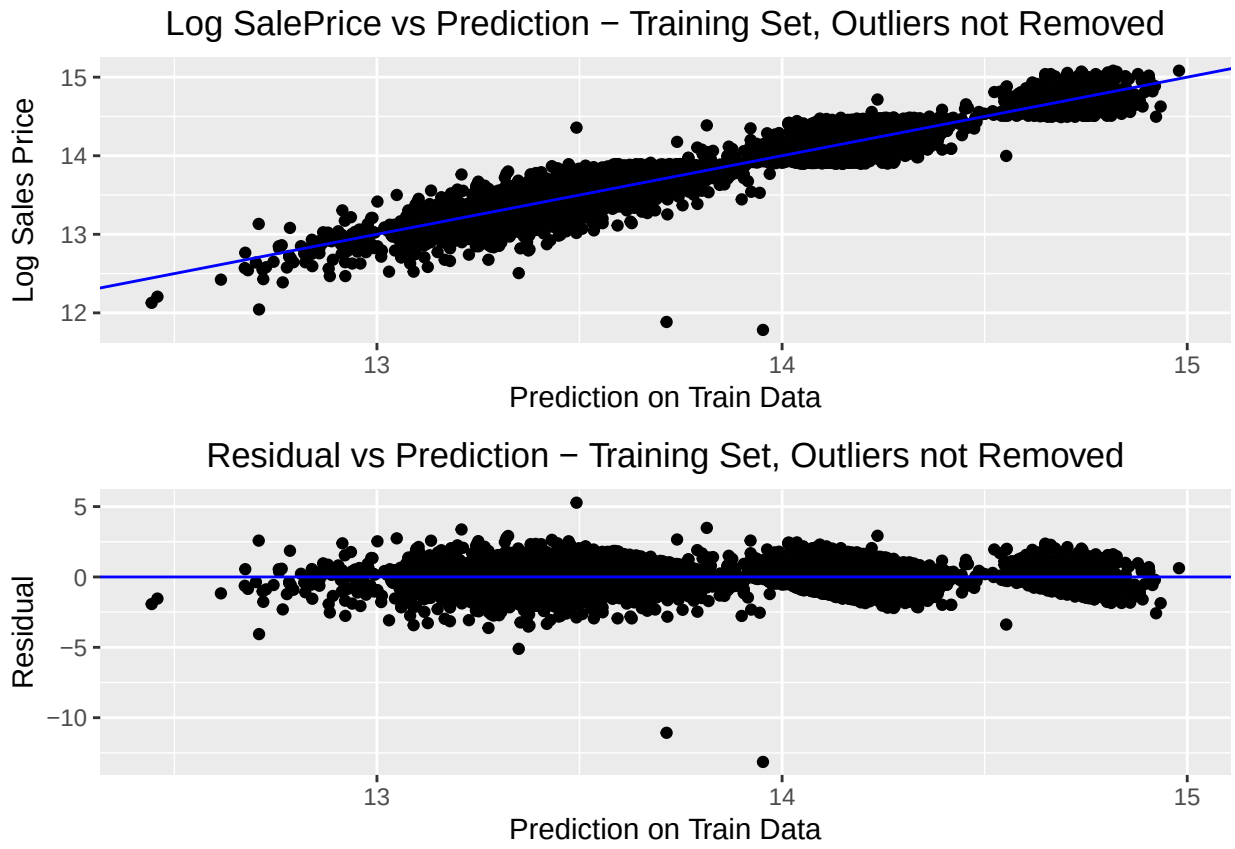
From the spline plots above, almost all of the numerical independent variables are visibly nonlinear. This can be observed through the spline plot lines that are curvy and have one or more inflection points such Landsize, BuildingArea, and EffAge. Although Distance does not seem to have an inflection point on the spline plot, it is still visible that it does not have a straight linear line. The spline plots are consistent with the previous model without clusters. However there are some differences such in the curves, a visible one being the Landsize.

6.4.1 Model Evaluation 1 - AIC & Residual Analysis

```
mod_gam2$aic
```

```
## [1] -3033.571
```

```
bin_gam = eval.train_init(mod_gam2, train, outliers = TRUE)
```




```
train_gam = eval.train_update(mod_gam2, train, bin_gam)
```

In the model's summary, that the model received an AIC value of -3035.398, which will be used to compare with the next model with removed outliers in the train set.

From the graph above, we can see from the first graph that the predictions made by the model are spread around the y=x line, which shows that the predictions are close to the actual values. However, we can see that there are a couple of wrong predictions. The second graph highlights these wrong outcomes as we can see that there are some residuals that are far from the predictions. Residuals are the difference between each predicted data and the actual value of the data. The standardized residuals with the `rstandard()` function will be evaluated and data whose corresponding residual values are greater than 3 will be removed in the following step.

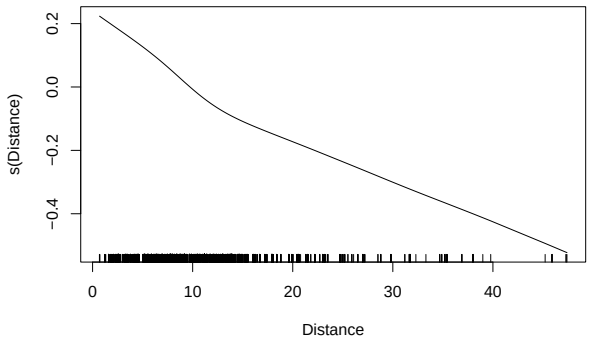
6.4.2 Remodelling With New Train Set

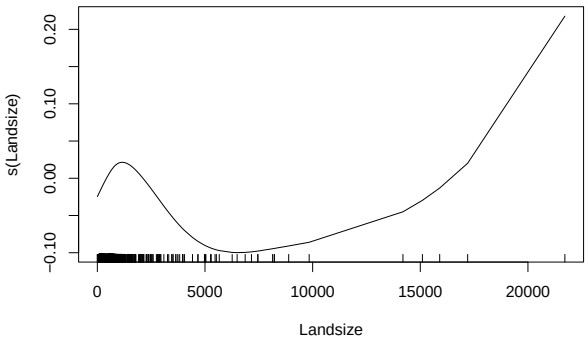
```
mod_gam2.final <- gam(f2, data = train_gam)
summary(mod_gam2.final)
```

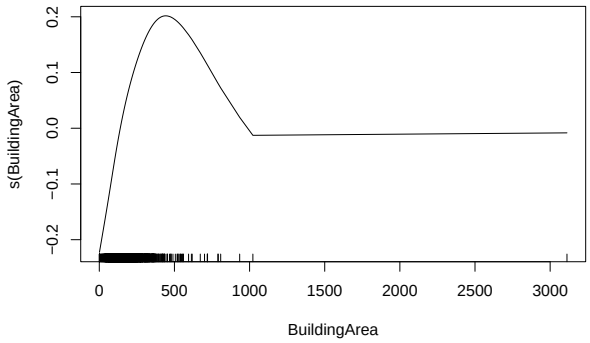
```
##
## Call: gam(formula = f2, data = train_gam)
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -0.506793 -0.101786  0.001691  0.103177  0.492376
##
```

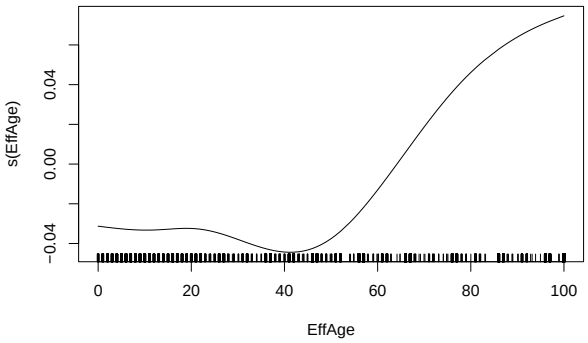
```
## (Dispersion Parameter for gaussian family taken to be 0.024)
##
##     Null Deviance: 977.037 on 4080 degrees of freedom
## Residual Deviance: 96.2918 on 4012 degrees of freedom
## AIC: -3568.966
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                    Df  Sum Sq Mean Sq  F value                Pr(>F)
## s(Distance)         1  92.786  92.786 3865.930 < 0.00000000000000022 ***
## s(Landsize)         1   0.507   0.507   21.110           0.000004471 ***
## s(BuildingArea)     1 225.340 225.340 9388.797 < 0.00000000000000022 ***
## s(EffAge)           1  78.057  78.057 3252.229 < 0.00000000000000022 ***
## Rooms               7  88.147  12.592  524.666 < 0.00000000000000022 ***
## Type                2   7.980   3.990  166.248 < 0.00000000000000022 ***
## Car                 4   9.367   2.342   97.565 < 0.00000000000000022 ***
## Regionname          7 153.177  21.882  911.732 < 0.00000000000000022 ***
## CouncilArea        30  44.506   1.484   61.811 < 0.00000000000000022 ***
## Cluster             2  86.673  43.336 1805.608 < 0.00000000000000022 ***
## Residuals        4012  96.292   0.024
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                  Npar Df  Npar F                 Pr(F)
## (Intercept)
## s(Distance)           3  19.923 0.0000000000008144596 ***
## s(Landsize)           3  18.263 0.0000000000091011643 ***
## s(BuildingArea)       3 105.372 < 0.00000000000000022 ***
## s(EffAge)             3  24.567 0.0000000000000008882 ***
## Rooms
## Type
## Car
## Regionname
## CouncilArea
## Cluster
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

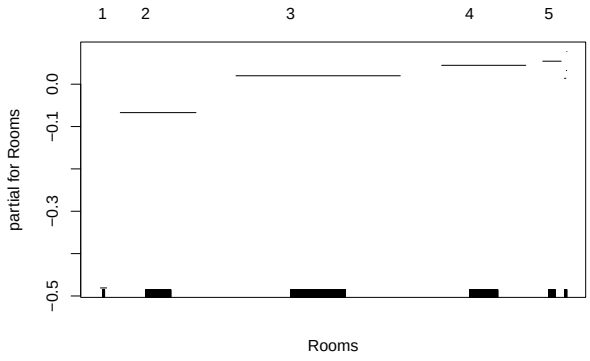
```
plot(mod_gam2.final)
```

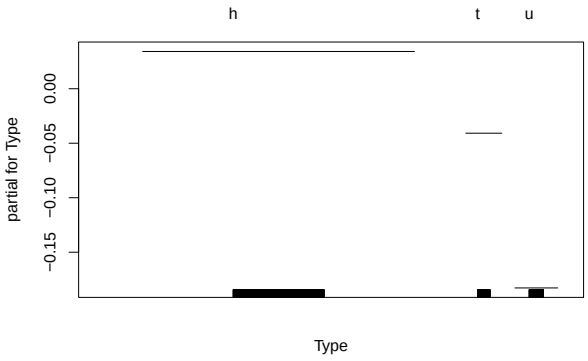

s(Distance)
0.2
0.0
−0.2
−0.4
0
10
20
30
40
Distance

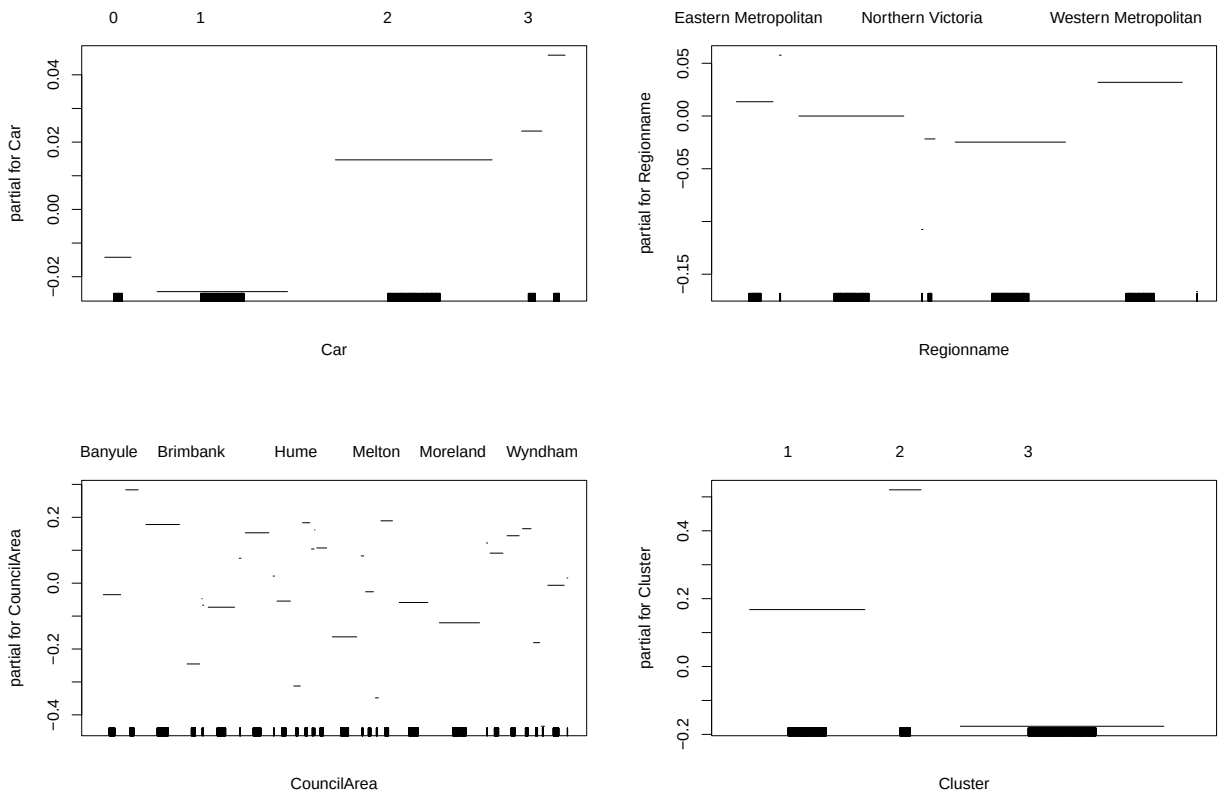

s(Landsize)
0.20
0.10
0.00
−0.10
0
5000
10000
15000
20000
Landsize


s(BuildingArea)
0.2
0.1
0.0
−0.1
−0.2
0
500
1000
1500
2000
2500
3000
BuildingArea


s(EffAge)
0.04
0.00
−0.04
0
20
40
60
80
100
EffAge


1
2
3
4
5
partial for Rooms
0.0
−0.1
−0.3
−0.5
Rooms


h
t
u
partial for Type
0.00
−0.05
−0.10
−0.15
Type



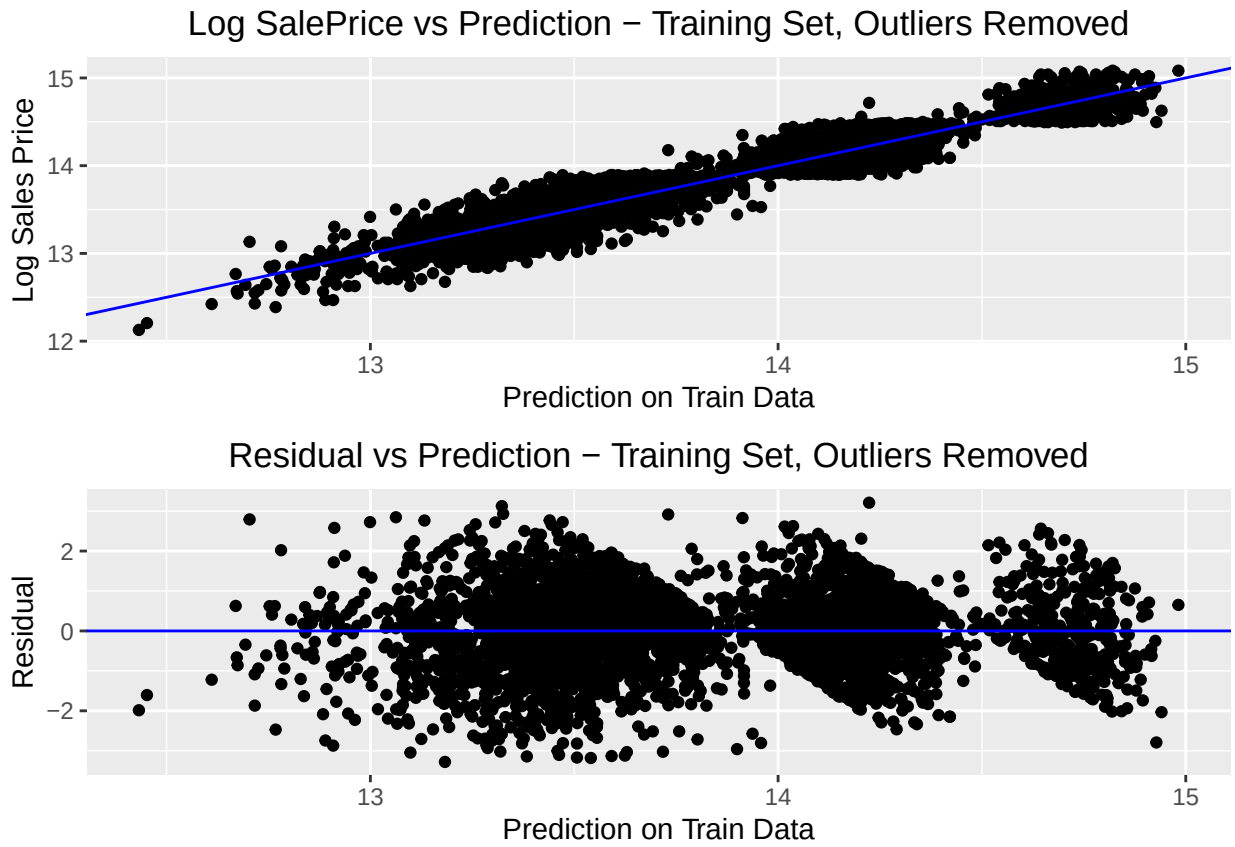
These spline plots as a result of fitting with the outliers removed are consistent with the previous spline plots. Additionally, Landsize has a steep declining line, unlike the previous plot.

6.4.3 Model Evaluation 2 - New AIC, Residual Analysis, and Error Analysis

```
data.frame("Outliers Not Removed" = mod_gam2$aic,
           "Outliers Removed" = mod_gam2.final$aic)
```

```
##   Outliers.Not.Removed Outliers.Removed
## 1            -3033.571        -3568.966
```

```
eval.train_init(mod_gam2.final, train_gam, outliers = FALSE)
```

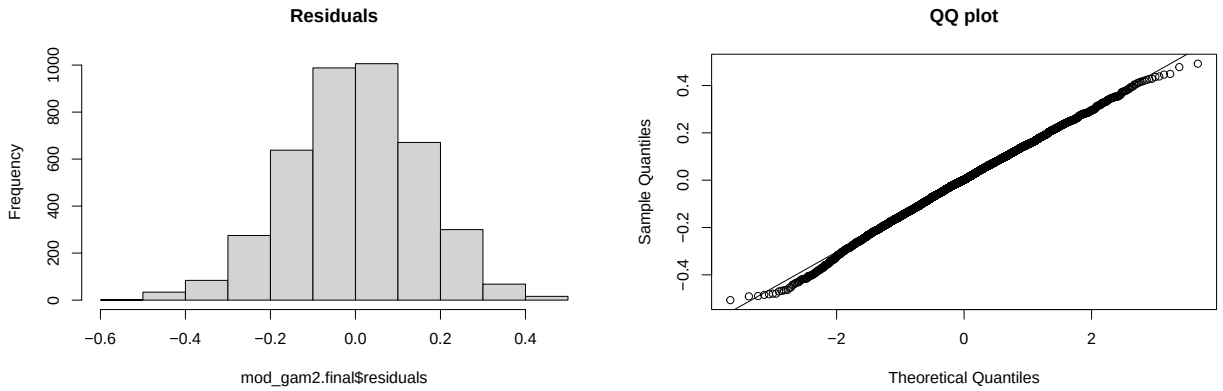


After we have removed the outliers in the training set, we can see that there is a highly significant decrease in the AIC of the model. This means that the model is now performing much better with outliers removed.

From the graphs, it can be observed that the predictions are now closer the actual values and the residuals are also closer to 0. Moreover, there seems to be a pattern in the residuals. Further investigation can be considered on this issue.

Furthermore, the normality of the model's residuals will be analyzed.

```
hist(mod_gam2.final$residuals, main="Residuals")
qqnorm(mod_gam2.final$residuals, main="QQ plot"); qqline(mod_gam2.final$residuals)
```



It can also be observed from the histogram and qqplot that the residuals resemble a normal distribution.

Now, we are going to analyze the error in the predictions of the train set. We will also compare the metrics with the previous model with outliers not removed.

```
merge(stack(comp(exp(mod_gam2$fitted.values), exp(mod_gam2$y))),
      stack(comp(exp(mod_gam2.final$fitted.values), exp(mod_gam2.final$y))),
      by = "ind", sort = FALSE)
```

```
##     ind            values.x            values.y
## 1     n        4100.0000000        4081.0000000
## 2    R2           0.8966523           0.8992087
## 3   MSE 35411985487.2865601 34398283792.4767990
## 4 SEMSE  1352647446.0247760  1302163185.1231389
## 5  RMSE      188180.7255999      185467.7432668
## 6    SE        2933.4417838        2898.5740287
## 7   MAE      134216.5039632      132643.9522756
## 8  MAPE          12.8979329          12.3336649
```

From the results, we can conclude that out of the 4082 data in the train set with outliers removed, the model receives an RMSE of 187181.731 and an average absolute error of 12.383%. These values show an increase in performance from the previous model with outliers intact. This can be concluded from the significant decrease in RMSE and MAPE values.

6.4.4 Multicollinearity

```
vif(mod_gam2.final)
```

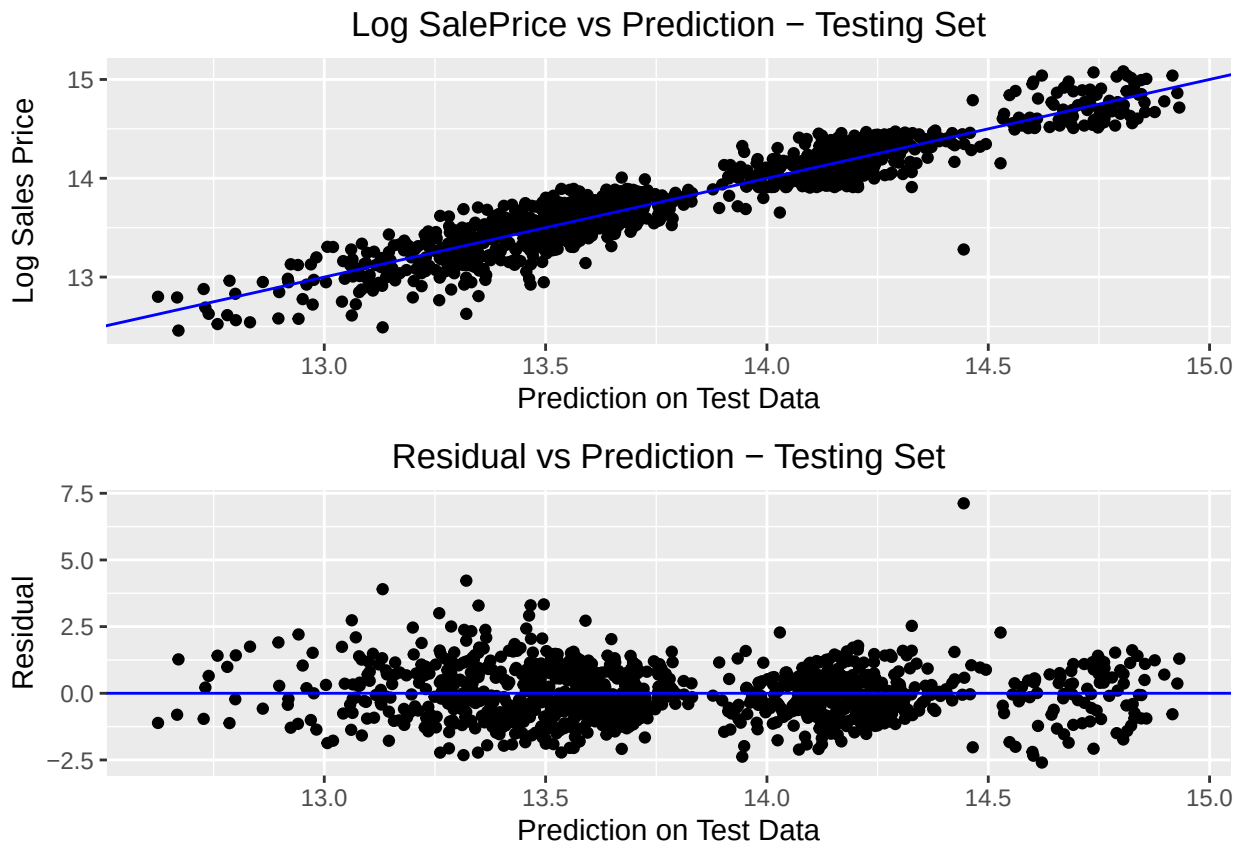
```
##                          GVIF Df GVIF^(1/(2*Df))
## s(Distance)          7.397451  1        2.719826
## s(Landsize)          1.090257  1        1.044154
## s(BuildingArea)      1.704539  1        1.305580
## s(EffAge)            1.801456  1        1.342183
## Rooms                2.704969  7        1.073665
## Type                 2.311983  2        1.233094
## Car                  1.773371  4        1.074237
## Regionname       45540.413091  7        2.151506
## CouncilArea     200964.591253 30        1.225703
## Cluster              2.750939  2        1.287865
```

From the VIF scores, we can see that almost all numerical variables have a relatively small VIF with values below 4. Therefore, we can conclude that they are not linear combinations of other independent variables. However, Distance received a VIF of 7.035 which is relatively more than other VIF values. Since Distance do not seem to be a linear combination of other numerical variables and its VIF value does not have a relatively big difference with other VIF values, Distance can still be considered to be used in this model. On the other hand, the VIF values of the categorical variables will be ignored.

6.4.5 Prediction and Error Analysis

```
pred_gam2 = predict(mod_gam2.final, newdata = test, type = "response")
```

```
eval_gam2 = eval.test(pred_gam2, test$Price)
```




```
## $n
## [1] 1024
##
## $R2
## [1] 0.8924015
##
## $MSE
## [1] 38881830222
##
## $SEMSE
## [1] 3396699672
##
## $RMSE
## [1] 197184.8
##
## $SE
## [1] 6158.396
##
## $MAE
## [1] 138779.7
##
## $MAPE
## [1] 13.18041
```

From the graph, we can see that there are some predictions that are far away from the y=x line, which means that the model has some inaccurate predictions in the test set. The second graph further highlights these findings.

From the results, we can conclude that out of the 1024 data in the test set, the model successfully predicted the prices with RMSE of 193924.5 and average absolute error of 12.75%. These metrics and the model's AIC will be kept for further comparison with the results from other models.

---

7 Modelling - GBM

Gradient boosting method (GBM) is a tree-based machine learning algorithm which combines the predictions of multiple decision trees generated into a certain final prediction. The generation of trees are done iteratively based on the residuals or errors from the previous iteration.

7.1 Model Assumption

The GBM model may have an assumption that the encoded integer value for each variable has an ordinal relation. Therefore, the selection of clusters in the previous section may have been correct. Other than that, the GBM model does not require any other model assumptions.

7.2 Train-Eval Split for Cross Validation

For the gradient boosting method model, a 10-fold cross validation will be used. Stratified sampling will be applied in the train-test set split with `createDataPartition()` with a split ratio of 80:20. Folds for cross validation in the train set will be done with `createFolds()` with an amount of 10 folds.

```
set.seed(1)
n.folds <- 10
folds <- createFolds(y=train$Cluster, k=n.folds, list=T, returnTrain=F)
```

Parameter optimization will be done with 10-Fold Cross Validation. The parameters to be tuned include `n.trees` and `depth`. Moreover, the GBM model will be created using the parameters `shrinkage=0.01` and `n.minobsinnode=5`.

```
try_ntrees = c(5000, 10000, 20000, 30000, 50000)
try_depths = c(3,5,7,9)
```

7.3 GBM Without Clusters

7.3.1 Cross Validation

```
cv_MAPE1 <- NULL
cv_MAPE1 <- matrix(nrow = length(try_ntrees), ncol = length(try_depths))
rownames(cv_MAPE1) = try_ntrees
colnames(cv_MAPE1) = try_depths
```

```
# Note: this code will not be run for knitting purposes
# Results are presented in the next chunk
tic("GBM 1 CV")
for (n in try_ntrees){
  MAPE.ave <- NULL;
    for (d in try_depths){
    MAPE <- NULL; i=1
    for(fold in folds){
      print(paste(n,d,i)); i=i+1
      ## GBM
      mod = gbm(log(Price)~.-Cluster, data=train[-fold, ],
                n.trees=n, interaction.depth=d,
                shrinkage=0.01, n.minobsinnode=5, verbose=F)

      ## Predicting in the validation set
      pred = predict(mod, newdata=train[fold, ], type="response")

      MAPE = c(MAPE, comp(exp(pred), train[fold, ]$Price)$MAPE)

      ## Freeing Memory
      rm(mod); gc()
    }
    MAPE.ave = c(MAPE.ave, mean(MAPE));
  }
  cv_MAPE1[paste(n), ] = MAPE.ave
  print(cv_MAPE1[paste(n), ])
}
toc()
```

```
# Cross Validation GBM 1 Results
# Normal Price - Runtime 4.31403 hours
cv_MAPE1["5000", ] <- c(15.50530,15.20302,15.10528,15.03173)
cv_MAPE1["10000", ] <- c(15.37814,15.19685,15.07715,15.12083)
cv_MAPE1["20000", ] <- c(15.41938,15.37908,15.44239,15.52413)
cv_MAPE1["30000", ] <- c(15.55725,15.50775,15.62089,15.68667)
cv_MAPE1["50000", ] <- c(15.84067,15.84754,15.93141,15.94548)
avg_MAPE11 = mean(cv_MAPE1)
print(cv_MAPE1)
```

```
##              3        5        7        9
## 5000  15.50530 15.20302 15.10528 15.03173
## 10000 15.37814 15.19685 15.07715 15.12083
## 20000 15.41938 15.37908 15.44239 15.52413
## 30000 15.55725 15.50775 15.62089 15.68667
## 50000 15.84067 15.84754 15.93141 15.94548
```

```
# Log Price - Runtime 3.950008 hours
cv_MAPE1["5000", ] <- c(14.51370,14.19185,14.09420,14.11684)
cv_MAPE1["10000", ] <- c(14.27702,14.17983,14.19976,14.27024)
cv_MAPE1["20000", ] <- c(14.28147,14.37412,14.48387,14.56592)
cv_MAPE1["30000", ] <- c(14.36344,14.56086,14.71708,14.76850)
cv_MAPE1["50000", ] <- c(14.59353,14.82618,14.97162,15.00474)
avg_MAPE12 = mean(cv_MAPE1)
print(cv_MAPE1)
```

```
##              3        5        7        9
## 5000  14.51370 14.19185 14.09420 14.11684
## 10000 14.27702 14.17983 14.19976 14.27024
## 20000 14.28147 14.37412 14.48387 14.56592
## 30000 14.36344 14.56086 14.71708 14.76850
## 50000 14.59353 14.82618 14.97162 15.00474
```

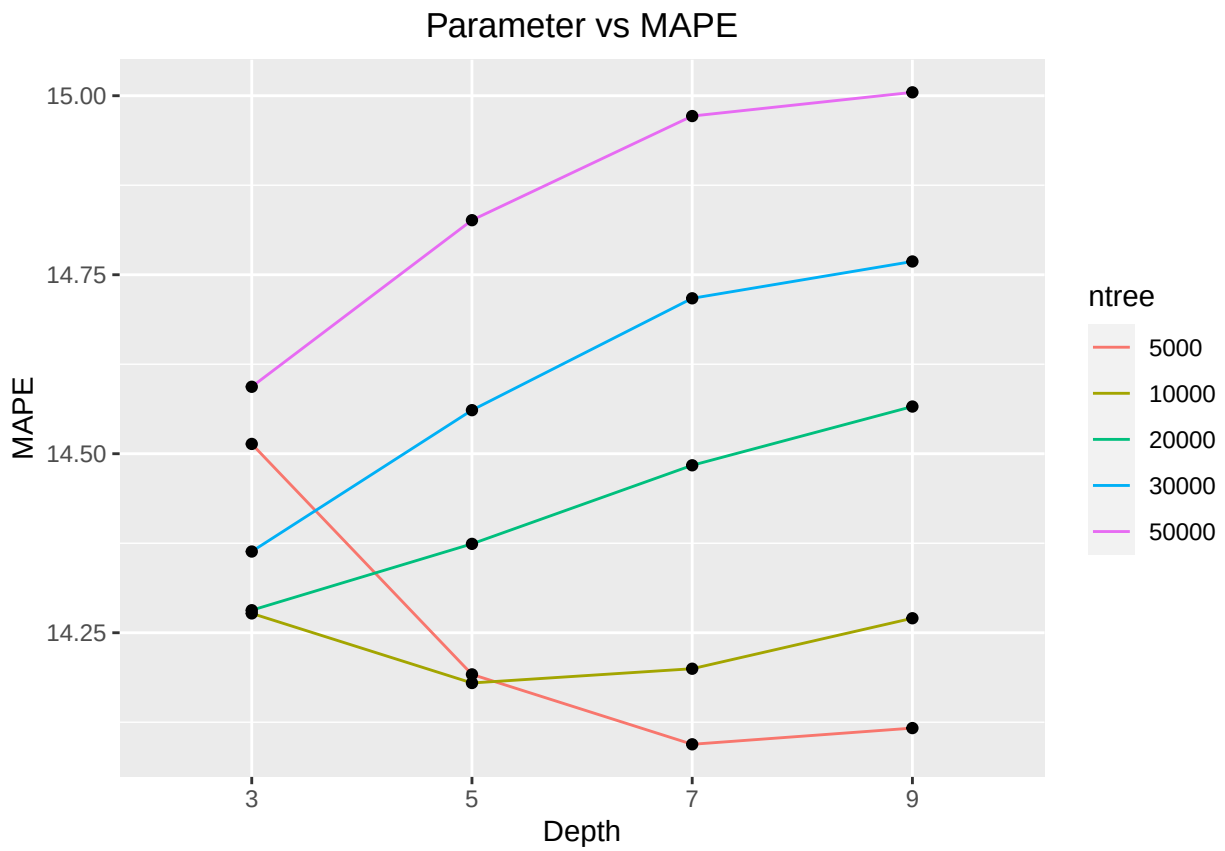
```
data.frame("Without Transformation" = avg_MAPE11,
           "Log Transform" = avg_MAPE12)
```

```
##   Without.Transformation Log.Transform
## 1               15.46605      14.46774
```

The cross validation was done twice for a model without transformation and with the logarithm transform on Price. The MAPE was calculated based on the non-transformed variable. It was found that the model performs better on average in the cross validation with the log transformation.

```
cv_MAPE1 = melt(cv_MAPE1)
cv_MAPE1$Var1 = as.factor(cv_MAPE1$Var1)
cv_MAPE1$Var2 = as.factor(cv_MAPE1$Var2)
```

```
ggplot(cv_MAPE1, aes(x = Var2, y = value)) +
  geom_line(aes(color = Var1, group = Var1)) +
  geom_point()+
  ggtitle("Parameter vs MAPE") +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x = "Depth", y = "MAPE", color = "ntree")
```



From the graph above, there are some patterns that could be seen. In the case of ntree=5000, MAPE decreases as the number of depth increases. However, the MAPE increases at depth=9. The same does not hold for other ntrees since the MAPE increases as the depth increases for most cases. Furthermore, MAPE also tends to increase with the amount of ntree. This may be due to overfitting with too many trees generated.

Since an ideal model would be a model with less complexity and high accuracy, the parameters used will be n.trees=5000 and interaction.depth=7.

```
best_ntree1 = 5000
best_depth1 = 7
```

7.3.2 Modelling with Tuned Parameters

A model will now be created with optimized parameters obtained from the cross validation.

```
tic("GBM 1 Best")
set.seed(1)
mod_gbm1_best = gbm(log(Price)~.-Cluster, data=train,
                n.trees=best_ntree1, interaction.depth=best_depth1,
                shrinkage=0.01, n.minobsinnode=5, verbose=F)
```

```
## Distribution not specified, assuming gaussian ...
```

```
toc()
```

```
## GBM 1 Best: 19.97 sec elapsed
```

Model's prediction on the train set is as follows.

```
yhat = predict(mod_gbm1_best, newdata=train, type="response")
```

```
## Using 5000 trees...
```

```
comp(exp(yhat), train$Price)
```

```
## $n
## [1] 4100
##
## $R2
## [1] 0.9254843
##
## $MSE
## [1] 26291054950
##
## $SEMSE
## [1] 1131725245
##
## $RMSE
## [1] 162145.2
##
## $SE
## [1] 2521.555
##
## $MAE
## [1] 109699.9
##
## $MAPE
## [1] 9.577577
```

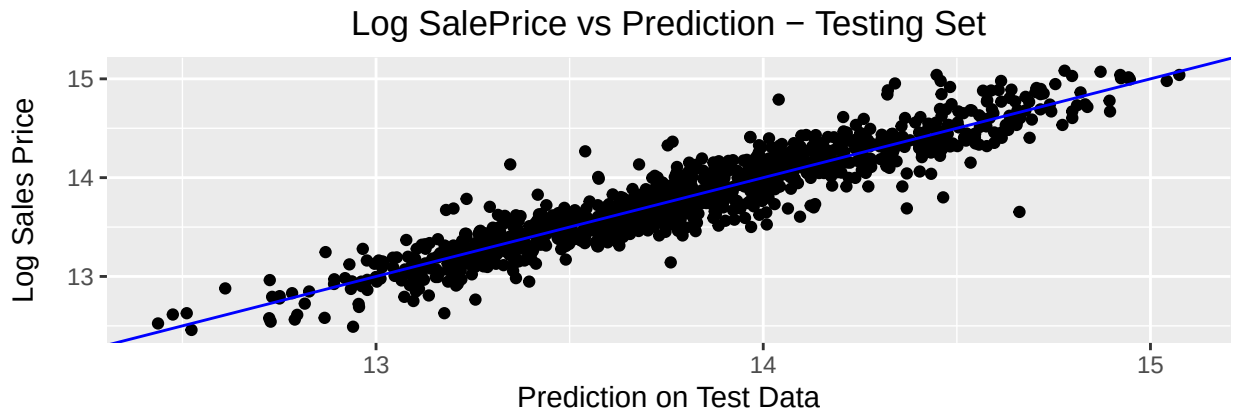
The model received an RMSE of 162145.2 and MAPE of 9.578%. These values indicate a good performance of the model.

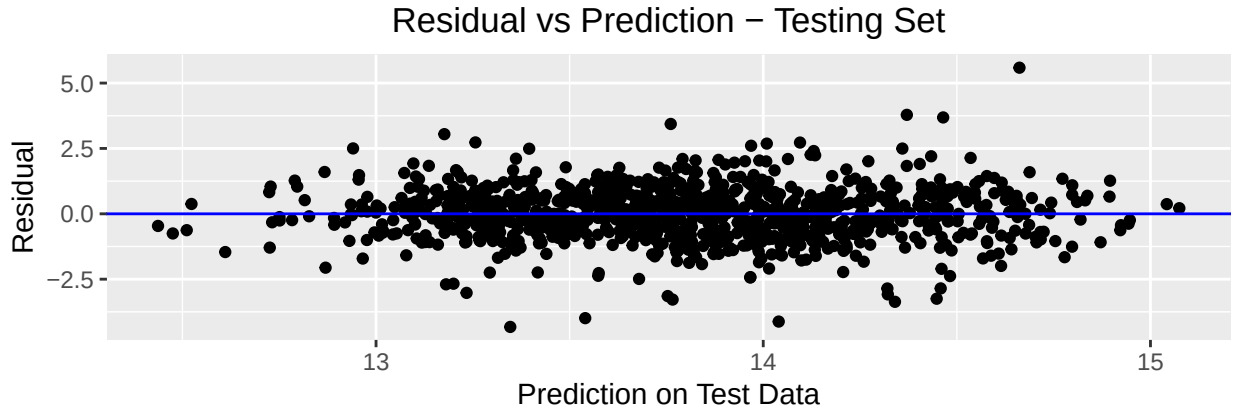
7.3.3 Prediction and Model Evaluation

```
## Predicting the test set
pred_gbm1 = predict(mod_gbm1_best, newdata=test, type="response")
```

```
## Using 5000 trees...
```

```
eval_gbm1 = eval.test(pred_gbm1, test$Price)
```



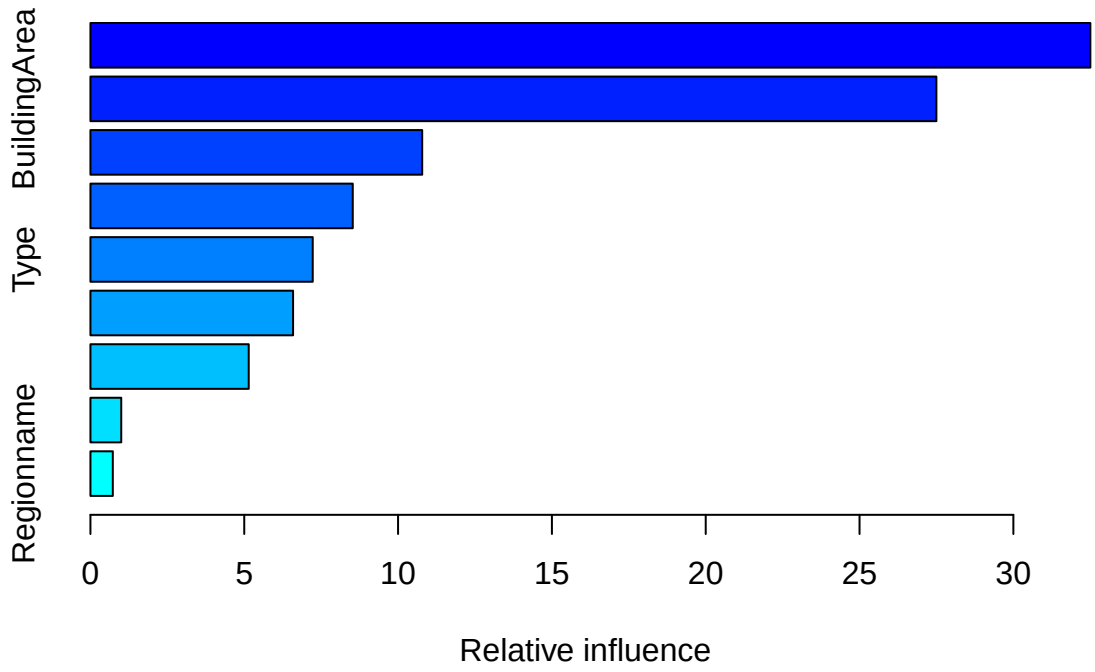



```
## $n
## [1] 1024
##
## $R2
## [1] 0.8408373
##
## $MSE
## [1] 58115574870
##
## $SEMSE
## [1] 5759957780
##
## $RMSE
## [1] 241071.7
##
## $SE
## [1] 7499.777
##
## $MAE
## [1] 158586.3
##
## $MAPE
## [1] 13.94643
```

The predictions of the model receive an RMSE value of 241071.7 and MAPE value of 13.946%. These results indicate a good performance in the model and will be compared with the results from other models.

From the residual plots, there are some points far from the lines which indicate that the model did not accurately predict the data with the corresponding residual. However, there does not seem to be any pattern in the residuals and therefore it is random.

```
(imp_gbm1 <- summary(mod_gbm1_best))
```




```
##                        var    rel.inf
## CouncilArea   CouncilArea 32.5029101
## BuildingArea BuildingArea 27.4976683
## EffAge             EffAge 10.7841618
## Distance         Distance  8.5271242
## Type                 Type  7.2261585
## Landsize         Landsize  6.5872612
## Rooms               Rooms  5.1456625
## Car                   Car  1.0015622
## Regionname     Regionname  0.7274911
```

From the variable importance, the three main variables that have a high importance towards the dependent variable include CouncilArea, BuildingArea, and EffAge. Additionally, number of rooms, car parking lots, and region of the house received the lowest variable importance. These findings are consistent with the exploratory data analysis, where it was found that BuildingArea and EffAge has the highest correlation values. On the other hand, Regionname may have become unimportant as it also explains the location of the house, which is similar to CouncilArea. This may result in the redundancy of independent variables.

7.4 GBM With Clusters

7.4.1 Cross Validation

```
cv_MAPE2 <- NULL
cv_MAPE2 <- matrix(nrow = length(try_ntrees), ncol = length(try_depths))
rownames(cv_MAPE2) = try_ntrees
colnames(cv_MAPE2) = try_depths
```

```
# Note: this code will not be run for knitting purposes
# Results are presented in the next chunk
tic("GBM 2 CV")
for (n in try_ntrees){
  MAPE.ave <- NULL;
    for (d in try_depths){
    MAPE <- NULL; i=1
    for(fold in folds){
      print(paste(n,d,i)); i=i+1
      ## GBM
      mod = gbm(log(Price)~., data=train[-fold, ],
                n.trees=n, interaction.depth=d,
                shrinkage=0.01, n.minobsinnode=5, verbose=F)

      ## Predicting in the validation set
      pred = predict(mod, newdata=train[fold, ], type="response")

      MAPE = c(MAPE, comp(exp(pred), train[fold, ]$Price)$MAPE)

      ## Freeing Memory
      rm(mod); gc()
    }
    MAPE.ave = c(MAPE.ave, mean(MAPE));
  }
  cv_MAPE2[paste(n), ] = MAPE.ave
  print(cv_MAPE2[paste(n), ])
}
toc()
```

```
# Cross Validation GBM 2 Results
# Normal Price - Runtime 5.090928 hours
cv_MAPE2["5000", ] <- c(12.04435,11.74558,11.60044,11.53538)
cv_MAPE2["10000", ] <- c(11.87802,11.70784,11.65690,11.66012)
cv_MAPE2["20000", ] <- c(11.82695,11.83290,11.88897,11.98451)
cv_MAPE2["30000", ] <- c(11.93501,12.00012,12.07607,12.16454)
cv_MAPE2["50000", ] <- c(12.14636,12.27087,12.30897,12.33763)
avg_MAPE21 = mean(cv_MAPE2)
print(cv_MAPE2)
```

```
##              3        5        7        9
## 5000  12.04435 11.74558 11.60044 11.53538
## 10000 11.87802 11.70784 11.65690 11.66012
## 20000 11.82695 11.83290 11.88897 11.98451
```

```
## 30000 11.93501 12.00012 12.07607 12.16454
## 50000 12.14636 12.27087 12.30897 12.33763
```

```
# Log Price - Runtime 3.968056 hours
cv_MAPE2["5000", ] <- c(11.56440,11.30679,11.25243,11.24295)
cv_MAPE2["10000", ] <- c(11.43970,11.30901,11.34839,11.39803)
cv_MAPE2["20000", ] <- c(11.49090,11.49785,11.62484,11.70467)
cv_MAPE2["30000", ] <- c(11.54603,11.67585,11.79794,11.89088)
cv_MAPE2["50000", ] <- c(11.76255,11.92083,12.02956,12.05461)
avg_MAPE22 = mean(cv_MAPE2)
print(cv_MAPE2)
```

```
##              3        5        7        9
## 5000  11.56440 11.30679 11.25243 11.24295
## 10000 11.43970 11.30901 11.34839 11.39803
## 20000 11.49090 11.49785 11.62484 11.70467
## 30000 11.54603 11.67585 11.79794 11.89088
## 50000 11.76255 11.92083 12.02956 12.05461
```

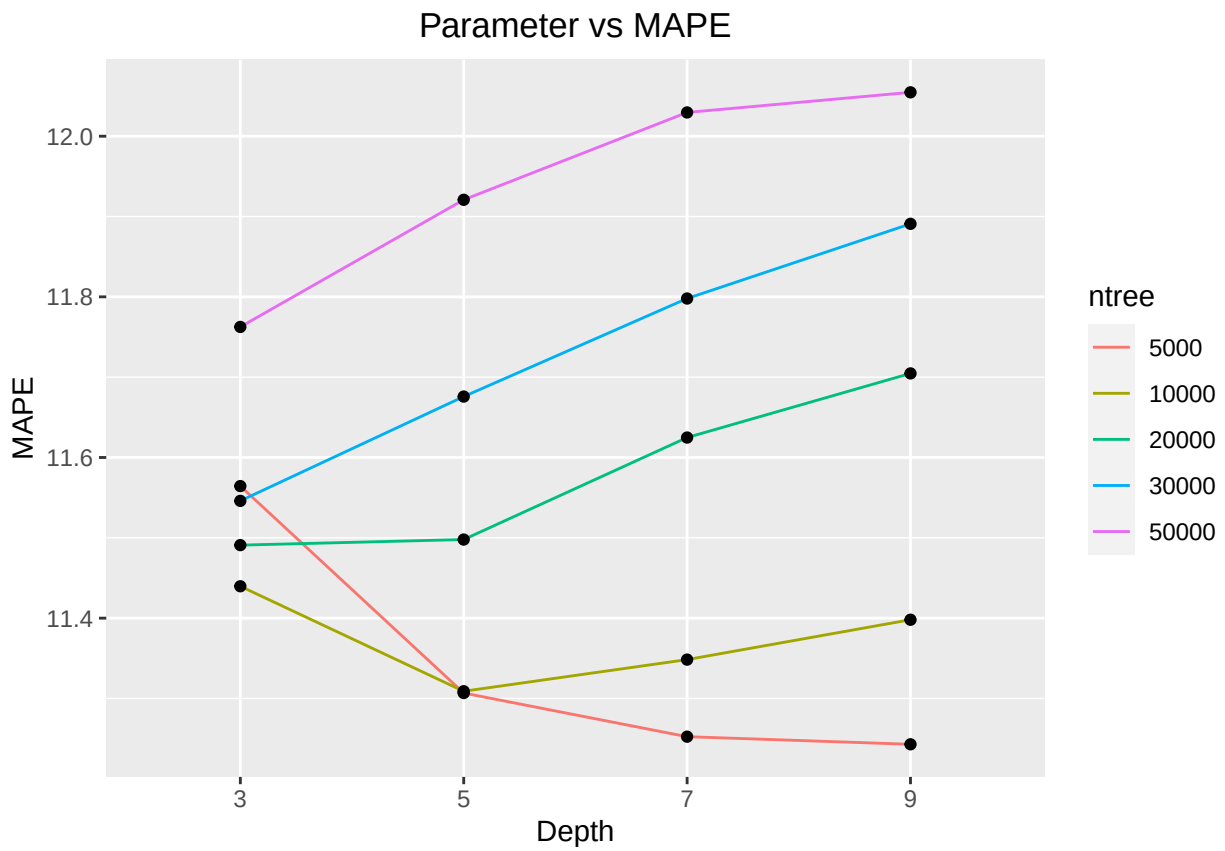
```
data.frame("Without Transformation" = avg_MAPE21,
           "Log Transform" = avg_MAPE22)
```

```
##   Without.Transformation Log.Transform
## 1               11.93008      11.59291
```

The cross validation was done twice for a model without transformation and with the logarithm transform on Price. The MAPE was calculated based on the non-transformed variable. It was found that the model performs better on average in the cross validation with the log transformation. These are consistent with the previous GBM model

```
cv_MAPE2 = melt(cv_MAPE2)
cv_MAPE2$Var1 = as.factor(cv_MAPE2$Var1)
cv_MAPE2$Var2 = as.factor(cv_MAPE2$Var2)
```

```
ggplot(cv_MAPE2, aes(x = Var2, y = value)) +
  geom_line(aes(color = Var1, group = Var1)) +
  geom_point()+
  ggtitle("Parameter vs MAPE") +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x = "Depth", y = "MAPE", color = "ntree")
```



From the graph above, there are some patterns that could be seen. In the case of ntree=5000, MAPE decreases as the number of depth increases. The same does not hold for other ntrees since the MAPE increases as the depth increases for most cases. Furthermore, MAPE also tends to increase with the amount of ntree. This may be due to overfitting with too many trees generated. These results are also similar to the previous GBM cross validation.

Since an ideal model would be a model with less complexity and high accuracy, the parameters used will be n.trees=5000 and interaction.depth=9.

```
best_ntree2 = 5000
best_depth2 = 9
```

7.4.2 Modelling with Tuned Parameters

A model will now be created with optimized parameters obtained from the cross validation.

```
tic("GBM 2 Best")
set.seed(1)
mod_gbm2_best = gbm(log(Price)~., data=train,
                n.trees=best_ntree2, interaction.depth=best_depth2,
                shrinkage=0.01, n.minobsinnode=5, verbose=F)
```

```
## Distribution not specified, assuming gaussian ...
```

```
toc()
```

```
## GBM 2 Best: 27.54 sec elapsed
```

Model's prediction on the train set is as follows.

```
yhat = predict(mod_gbm2_best, newdata=train, type="response")
```

```
## Using 5000 trees...
```

```
comp(exp(yhat), train$Price)
```

```
## $n
## [1] 4100
##
## $R2
## [1] 0.9643244
##
## $MSE
## [1] 12323742929
##
## $SEMSE
## [1] 486698061
##
## $RMSE
## [1] 111012.4
##
## $SE
## [1] 1730.398
##
## $MAE
## [1] 77962.76
##
## $MAPE
## [1] 6.949578
```

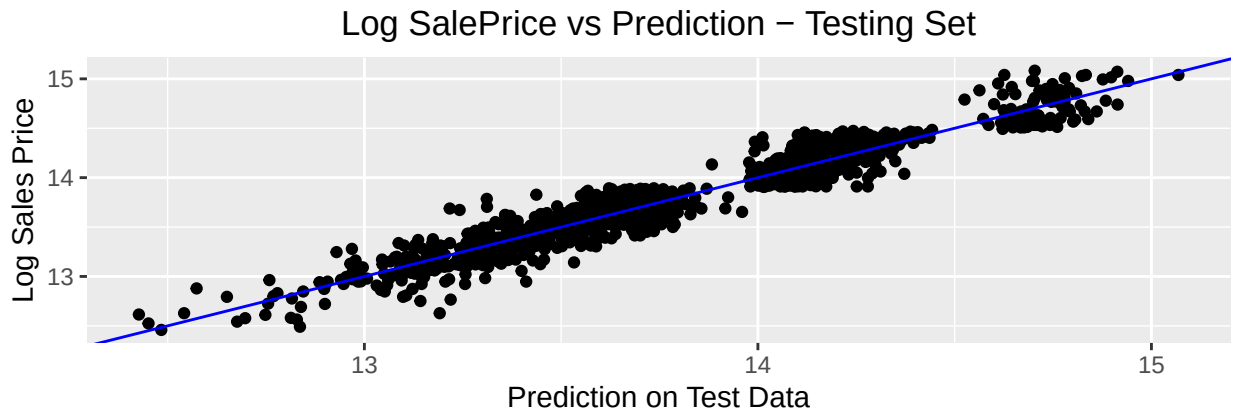
The model received an RMSE of 111012.4 and MAPE of 6.950%. These values might indicate a good performance of the model.

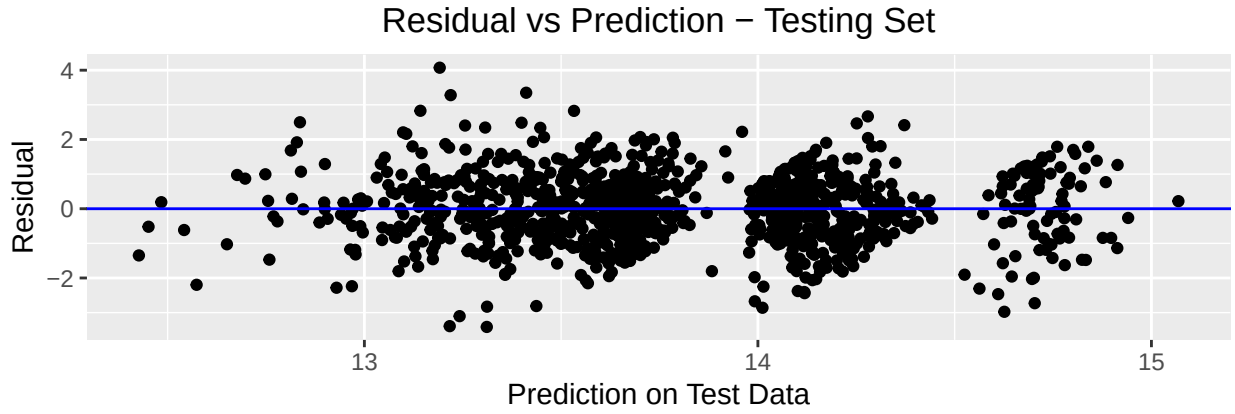
7.4.3 Prediction and Model Evaluation

```
## Predicting the test set
pred_gbm2 = predict(mod_gbm2_best, newdata=test, type="response")
```

```
## Using 5000 trees...
```

```
eval_gbm2 = eval.test(pred_gbm2, test$Price)
```



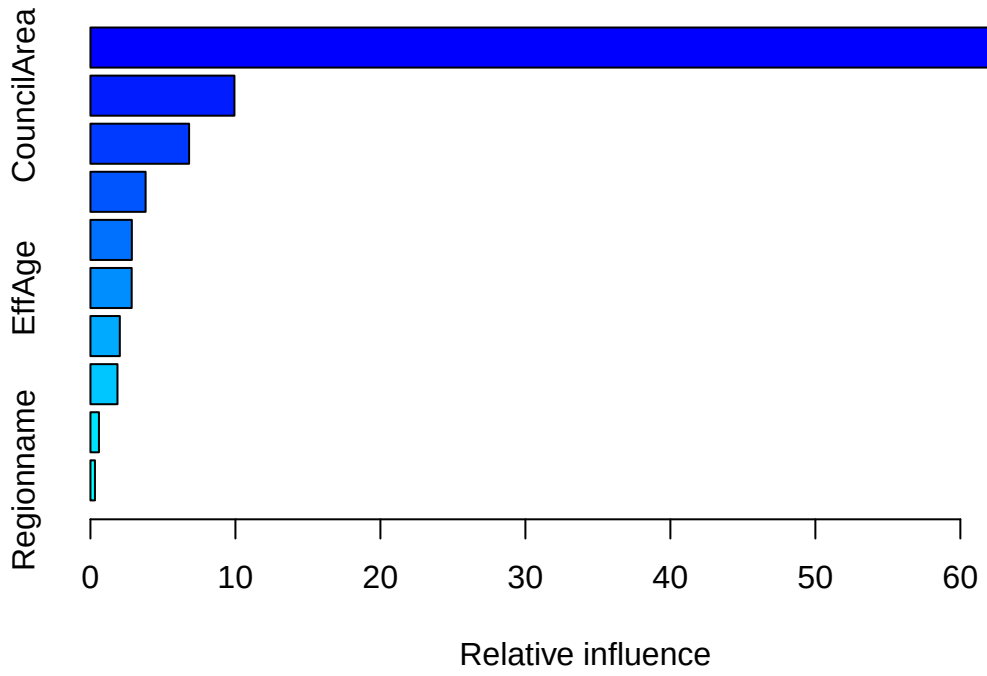



```
## $n
## [1] 1024
##
## $R2
## [1] 0.9109614
##
## $MSE
## [1] 32275983477
##
## $SEMSE
## [1] 2778887788
##
## $RMSE
## [1] 179655.2
##
## $SE
## [1] 5604.127
##
## $MAE
## [1] 123780.8
##
## $MAPE
## [1] 11.06097
```

The predictions of the model receive an RMSE value of 179655.2 and MAPE value of 11.061%. These results indicate a good performance in the model and will be compared with the results from other models.

From the residual plots, there are some points far from the lines which indicate that the model did not accurately predict the data with the corresponding residual. Additionally, patterns can be found in the residuals, just like the results found in the GAM model with the Cluster variable. Further investigation can be done on this problem.

```
(imp_gbm2 <- summary(mod_gbm2_best))
```




```
##                        var    rel.inf
## Cluster             Cluster 68.9658377
## CouncilArea     CouncilArea  9.9311180
## BuildingArea BuildingArea  6.8049245
## Landsize           Landsize  3.8052965
## Distance           Distance  2.8571922
## EffAge               EffAge  2.8458683
## Rooms                Rooms  2.0266998
## Type                  Type  1.8656697
## Car                    Car  0.5871591
## Regionname     Regionname  0.3102341
```

From the variable importance, the Cluster variable dominates its importance in comparison to other variables. The next variables have the same order of importance with the variable importance obtained from the previous GBM model. These variable importance results further highlights the affect of clustering in modelling.

8 Model Comparison

The metrics from the predictions will be compared in this section.

```
cbind("GAM Without Clustering"=eval_gam1,
      "GAM With Clustering"=eval_gam2,
      "GBM Without Clustering"=eval_gbm1,
      "GBM With Clustering"=eval_gbm2)
```

```
##       GAM Without Clustering GAM With Clustering GBM Without Clustering
## n     1024                   1024                1024
## R2    0.7744738              0.8924015           0.8408373
## MSE   82523687398            38881830222         58115574870
## SEMSE 7674597838             3396699672          5759957780
## RMSE  287269.4               197184.8            241071.7
## SE    8939.286               6158.396            7499.777
## MAE   188612.3               138779.7            158586.3
## MAPE  16.8877                13.18041            13.94643
##       GBM With Clustering
## n     1024
## R2    0.9109614
## MSE   32275983477
## SEMSE 2778887788
## RMSE  179655.2
## SE    5604.127
## MAE   123780.8
## MAPE  11.06097
```

From the results above, the distinction in performance between the models is clearly shown. Models that were fitted without clustering perform worse than models with clusters as an independent variable. This can be seen through the lower RMSE and MAPE values. As a comparison, the average of the price in the test set is 1116454. The same also holds in all other metrics. Therefore, it can be assumed that clustering before modelling may increase the performance of a model.

Furthermore, similar contrast of GAM and GBM models can also be inferred from these metrics. GAM models tend to perform poorly than GBM models. Both RMSE and MAPE models in both with and without the clustering variable are lower in GBM. GBM also receives better indication of performance based the other metrics as well. Therefore, GBM, a tree based model, performs better than GAM, a linear based model, in the case of this dataset.

---

9 Conclusion

In conclusion, this project has explored the Melbourne Housing Snapshot dataset, transformed the dependent variable with log transformation, created clustering from the data, and modelled using the transformed dependent variable and clusters. It was found that the transformation and clustering affected the models' performance positively. The models used include GAM and GBM with GBM being a better model for this dataset.

From the variable importance, it was found out that the council area, building area, and age of the house highly affects the house's price. Therefore, real estate investors and people who are interested in buying houses should consider taking these factors into context before buying real estates.

For future studies, there could be more investigation on the patterns that exist in the residuals. Variable selection can also be done with ridge regression or lasso regression and feature engineering from the independent variables.