CYO London Housing

Toby Wong

Jun 2021

1 Introduction

This report is part of the "Create Your Own" project for the capstone course of Harvardx Data Science Professional Certificate.

The objective of this project is to find the best performing machine learning algorithm for housing price prediction and determine whether demographic data (population, crime rate and unemployment rate) can improve the accuracy of the housing price prediction.

The performance of the machine learning algorithms is tested by using residual mean squared error (RMSE) score and it is computed by using the formula below:

$$RMSE = \sqrt{\frac{1}{N}\sum(y_t - \hat{y}_p)^2}$$

Being:
N = number of samples
$\hat{y}_p$ = predicted value
y_t = target value

```
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2))
  }
```

2 London Housing Prices Data Set

The housing prices data in London is used for this project and the data was originally downloaded from Kaggle (https://www.kaggle.com/arnavkulkarni/housing-prices-in-london). Additional demographic data is downloaded from London Data Store (https://data.london.gov.uk).

The following process was performed to prepare the data-set for the machine learning models:

1. Download London housing prices data and demographic data

2. Verify and add the borough data using PostcodesioR package
3. Combine housing data with demographic data
4. Calculate price per square foot (ppsf), crime rate and unemployment rate
5. Simplify the types of property
6. Change the data to appropriate classes
7. Tidy up the data-set and remove unneeded data
8. Name the resulted data-set as "london"

```
# Load Packages
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.
if(!require(PostcodesioR)) install.packages("PostcodesioR", repos = "http://cran.us.r-project.
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
if(!require(readxl)) install.packages("readxl", repos = "http://cran.us.r-project.org")

# Download London housing dataset
dl <- tempfile()
download.file("https://github.com/tobwon/CYO/raw/main/London.csv.zip", dl)
data<-unz(dl, "London.csv")
data<-read.csv(data, header = TRUE, sep = ",")
data<-as.data.frame(data)
rm(dl)

# Add borough data and verify location of properties to by using postcode with PostcodesioR pa
# This will take some time and internet connection is needed
x<-1:3480
geo_info<-sapply(x, function(n){
  postcode_lookup(as.character(data[n,11]))
             })
geo_info<-as.data.frame(t(geo_info))

# Tidy up the London housing data-set and convert the varibles to appropriate types
temp_data<-data%>%
  mutate(price=as.numeric(Price),
         type=as.factor(House.Type),
         area=as.numeric(Area.in.sq.ft),
         bedroom=as.numeric(No..of.Bedrooms),
         bathroom=as.numeric(No..of.Bathrooms),
         reception=as.numeric(No..of.Receptions),
         ID = row_number())%>%
  select(ID, price, type, area, bedroom, bathroom, reception)

# Tidy up the postcode data-set and convert the varibles to appropriate types
```

```r
temp_info<-as.data.frame(geo_info)%>%
  mutate(postcode=as.character(postcode),
         region=as.character(region),
         borough=as.character(admin_district),
         ID = row_number())%>%
  select(ID, postcode, region, borough)

# Tidy up the data-set by removing properties not in London and simplify the property types to
london <- left_join(temp_data, temp_info, by="ID")%>%
  filter(str_detect(region, "London"))
london$type<-str_replace_all(london$type, c("New development"="Apartment", "Flat / Apartment"="

# Convert to price per square feet
london <- london%>%mutate(ppsf=price/area)

# Download London unemployment data
temp = tempfile(fileext = ".xlsx")
download.file("https://data.london.gov.uk/download/employment-rates-by-ethnicity/cf8a5d62-6918-

# Remove un-needed data and tidy up the unemployment data-set
data3 <- read_xlsx(temp, sheet=4)
data3 <-as.data.frame(data3)
london_unemployment<-data3[-c(1:3, 36:57), c(2, 3, 4, 7, 8, 11, 12, 15, 16, 19, 20, 23, 24, 27
london_unemployment[london_unemployment == "!"] <- "0"
london_unemployment[london_unemployment == "#"] <- "0"
london_unemployment[, c(2:15)] <- sapply(london_unemployment[, c(2:15)], as.numeric)

# Replace City of London with Hackeny
london$borough<-str_replace_all(london$borough, c("City of London"="Hackney"))

# Calculate the unemployment rate in different boroughs
london_unemployment$population<-london_unemployment$...4+london_unemployment$...8+london_unempl
london_unemployment$employment<-london_unemployment$`working age employment rate - white`+londo
london_unemployment$unemployment<-london_unemployment$population-london_unemployment$employment
london_unemployment$unemployment_rate<-london_unemployment$unemployment/london_unemployment$pop
london_unemployment<-london_unemployment%>%rename(borough=...2)%>%select(borough, population, u

# Download London crime data
data2 <- read.csv("https://data.london.gov.uk/download/recorded_crime_summary/d2e9ccfc-a054-41

# Tidy up the London crime data-set
data2 <- as.data.frame(data2)
london_crime<-data2%>%mutate(sum=rowSums(.[4:27]))%>%select(c("LookUp_BoroughName", "sum"))%>%g

# Combine London employment and crime data-sets
london_e_c<-left_join(london_crime, london_unemployment, by="borough")
```

```r
# Calculate crime rate in different boroughs
london_e_c<-london_e_c%>%mutate(crime_rate=crime/population)%>%select(c("borough", "population"

# Combine city of London and Hackney boroughs and add population, unemployment rate, and crime
london <-left_join(london, london_e_c, by="borough")
london$borough<-as.factor(london$borough)

# Remove un-needed columns
london<-subset(london, select=c(-ID, -region, -postcode, -price))

# Rearrange columns to put ppsf as the first column
london<-london[,c(7,1,2,3,4,5,6,8,9,10)]

# Remove boroughs which have less than 10 records
# london<-london[london$borough != "Croydon" & london$borough!= "Bromley" & london$borough!= "

# Remove temporary files
# rm(temp_data, temp_info, x, london_e_c, london_crime, london_unemployment)
```

3 Data Exploration

Before building the machine learning algorithm models, data exploration is carried out on "london" data-set to understand the structure and the data it contained. This will help to determine how the models will be built for the analysis.

```r
# Summary of "london" data-set
summary(london)
```

```
      ppsf                type           area            bedroom
 Min.   : 241.6   Apartment:1872   Min.   :  274.0   Min.   : 0.00
 1st Qu.: 763.2   House    :1251   1st Qu.:  815.5   1st Qu.: 2.00
 Median : 935.3   Penthouse:  96   Median : 1250.0   Median : 3.00
 Mean   :1101.7                    Mean   : 1601.5   Mean   : 3.01
 3rd Qu.:1249.6                    3rd Qu.: 2061.5   3rd Qu.: 4.00
 Max.   :7069.2                    Max.   :14358.0   Max.   :10.00

    bathroom       reception                          borough       population
 Min.   : 0.00   Min.   : 0.00   Wandsworth             : 676   Min.   :133000
 1st Qu.: 2.00   1st Qu.: 2.00   Westminster            : 441   1st Qu.:181600
 Median : 3.00   Median : 3.00   Kensington and Chelsea: 297   Median :262400
 Mean   : 3.01   Mean   : 3.01   Hammersmith and Fulham: 247   Mean   :240488
 3rd Qu.: 4.00   3rd Qu.: 4.00   Islington              : 227   3rd Qu.:283300
 Max.   :10.00   Max.   :10.00   Richmond upon Thames   : 227   Max.   :383400
                                 (Other)                :1104
 unemployment_rate   crime_rate
```

```
 Min.   :0.1648    Min.   :0.1330
 1st Qu.:0.2316    1st Qu.:0.1783
 Median :0.2570    Median :0.2044
 Mean   :0.2625    Mean   :0.2463
 3rd Qu.:0.3135    3rd Qu.:0.2549
 Max.   :0.3525    Max.   :0.4791
```

From the summary above, "london" data-set is noted to be a data.frame and it contains 10 columns/variables and 3219 rows/observations. The columns in "london" are:

- "ppsf" - price per square foot
- "type" - property type (apartment, house, penthouse)
- "area" - size of the property
- "bedroom" - number of bedroom
- "bathroom" - number of bathroom
- "reception" - number of reception
- "borough" - the borough which the property is located
- "population" - population of the borough
- "unemployment_rate" - unemployment rate of the borough
- "crime_rate" - crime rate of the borough

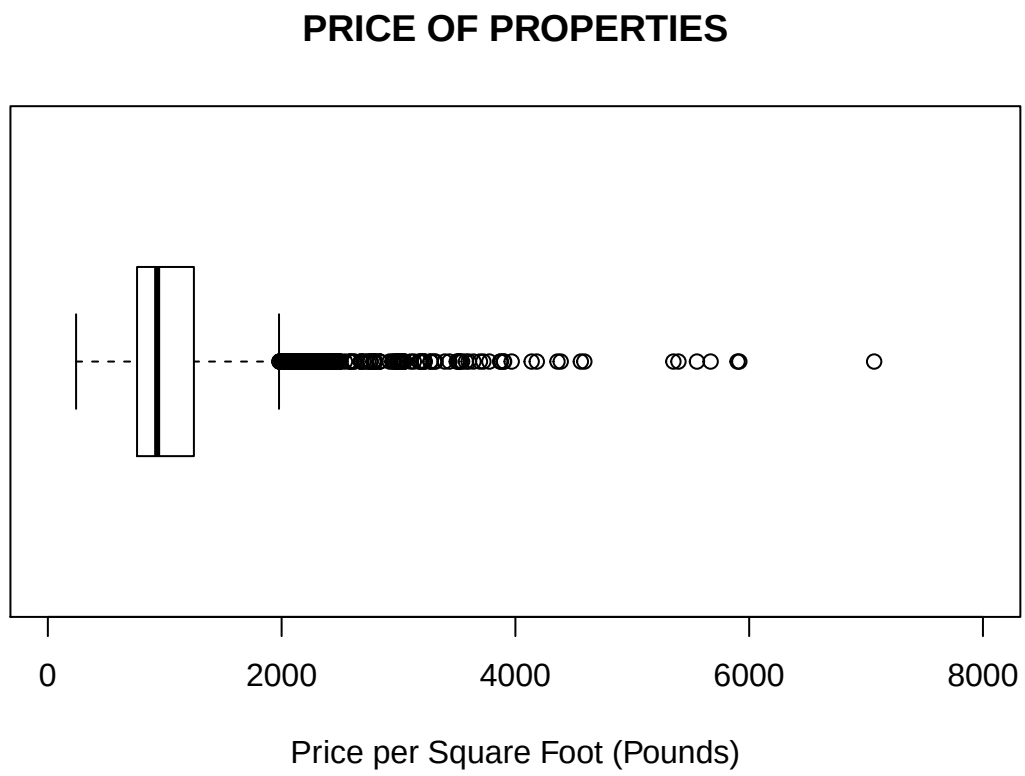
```
# Display the first 5 rows of "edx"
head(london)
```

```
       ppsf      type area bedroom bathroom reception     borough population
1  616.7158     House 2716       5        5         5      Merton     181600
2  798.5258 Apartment  814       2        2         2   Islington     222300
3  965.8344 Apartment  761       2        2         2  Wandsworth     283300
4  888.7210     House 1986       4        4         4  Wandsworth     283300
5  964.2857 Apartment  700       2        2         2  Wandsworth     283300
6 1042.1836 Apartment  403       1        1         1 Westminster     233200
  unemployment_rate crime_rate
1         0.2312775  0.1498789
2         0.2550607  0.2362798
3         0.1648429  0.1783445
4         0.1648429  0.1783445
5         0.1648429  0.1783445
6         0.3524871  0.4790523
```

The information above shown that except "type" and "borough" are in factor class, all other variables are in numeric class.

The data-set contains price per square foot (ppsf) variable and ppsf will be used in this project as the price prediction and this will take away the effect of the size of the property (area) when analyzing.

```r
#Plot the price range
boxplot(london$ppsf, horizontal = TRUE, xlab = "Price per Square Foot (Pounds)", main = "PRICE
```



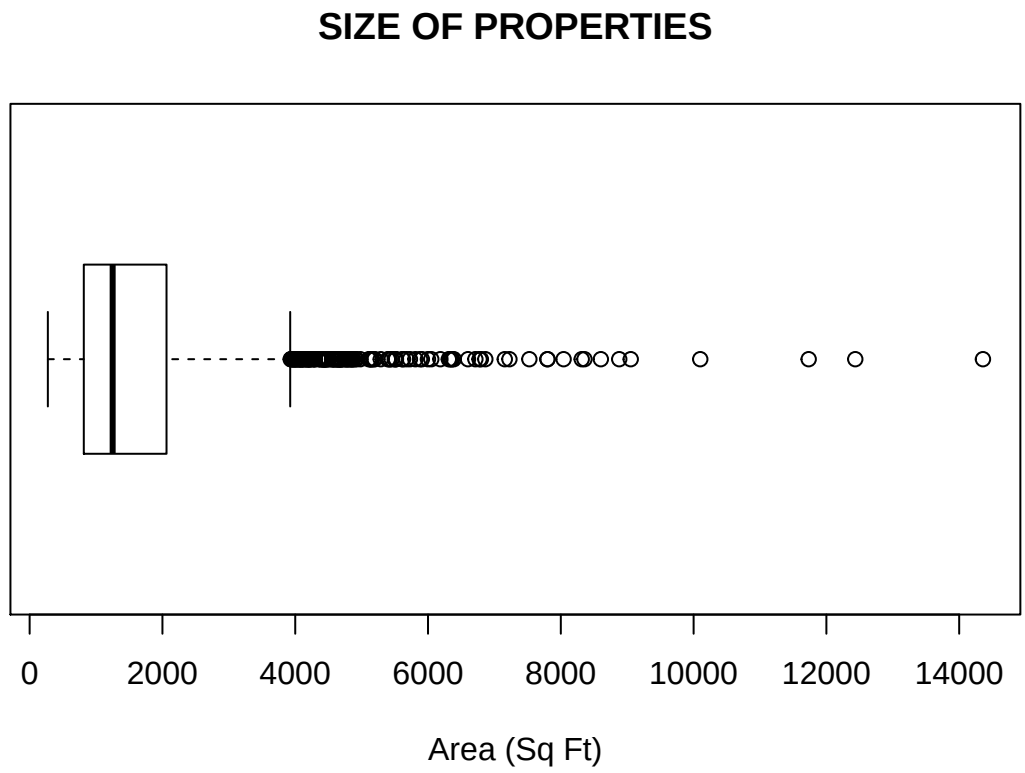
The ppsf has a relatively wide range from 241.6107383to 7069.1801529 and as shown in the boxplot above, majority of the property is less than 2,000 pounds per sq ft.

```r
# Types of property
london%>%group_by(type) %>%
  summarize(count = n(), .groups='drop') %>%
  arrange(desc(count))
```

```
# A tibble: 3 x 2
  type      count
  <fct>     <int>
1 Apartment  1872
2 House      1251
3 Penthouse    96
```

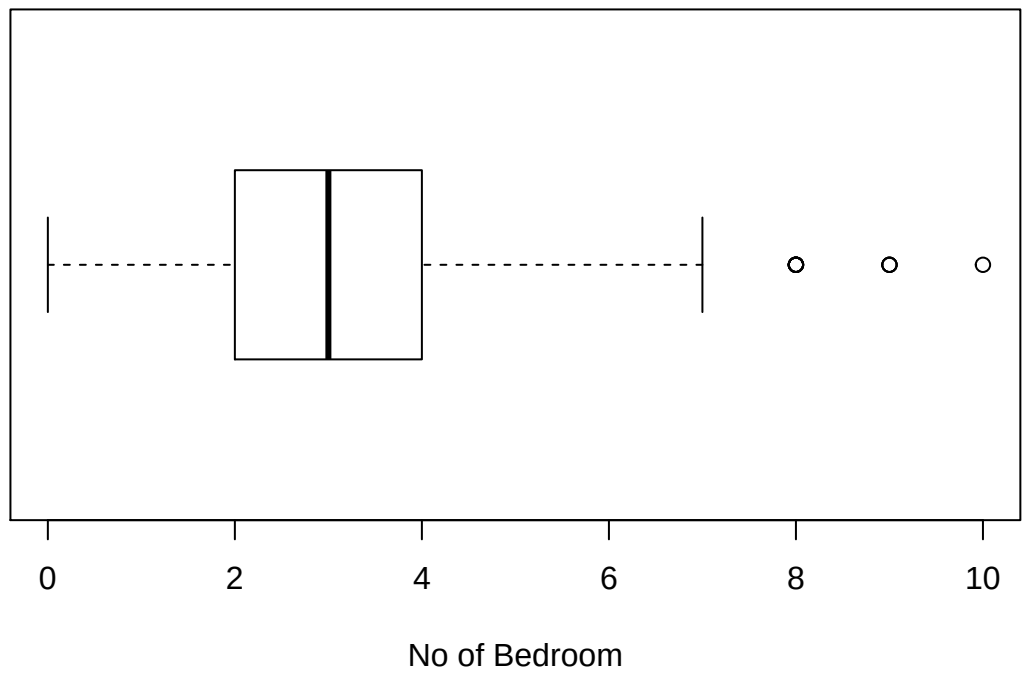
The table above shown the properties were sorted into three different types as shown in the table below. Apartment and House are the two predominant types.

```
boxplot(london$area, horizontal = TRUE, main = "SIZE OF PROPERTIES", xlab = "Area (Sq Ft)")
```



As shown in the boxplot above, majority of the property is less than 2000 sq ft. The range of the property size is from 274`to 14358`.

```
boxplot(london$bedroom, horizontal = TRUE, main = "BEDROOM NUMBER", xlab = "No of Bedroom")
```



The boxplot demonstrated that most of the properties has two to four bedrooms and some larger ones has up to ten bedrooms.

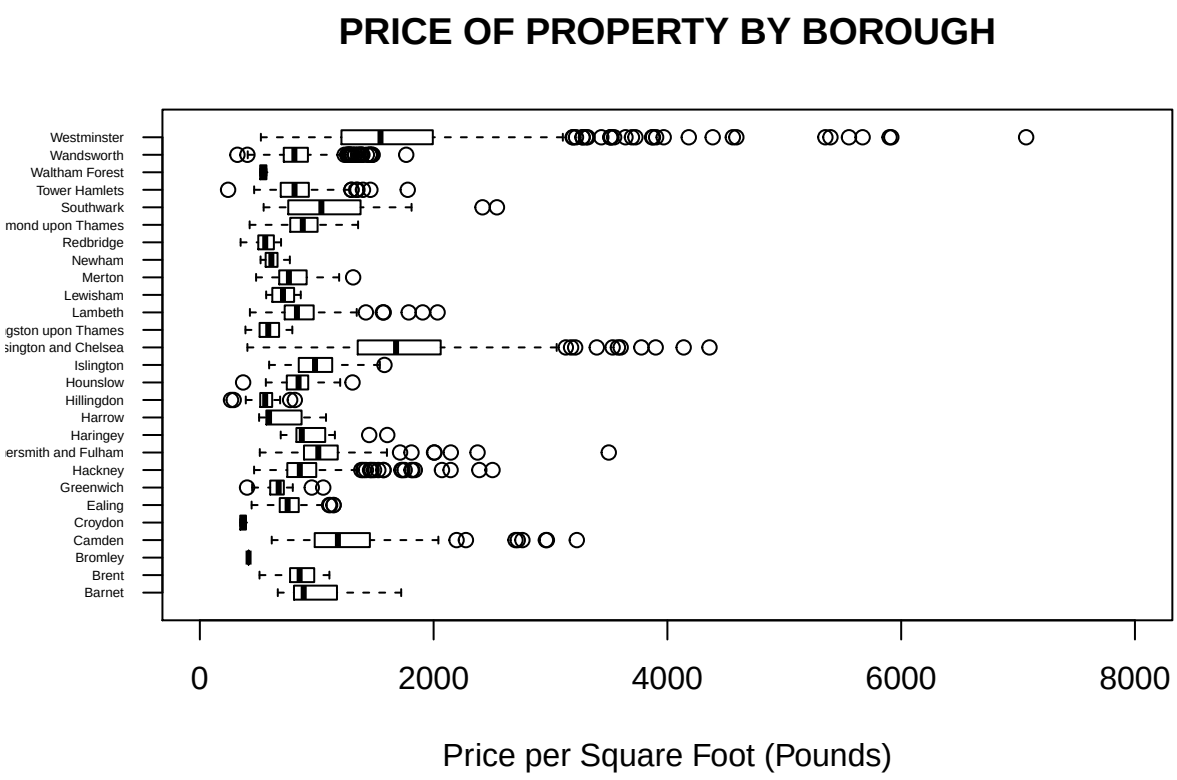
```
london%>%group_by(borough) %>%
  summarize(count = n(), .groups='drop') %>%
  arrange(desc(count))
```

```
# A tibble: 27 x 2
   borough                count
   <fct>                  <int>
 1 Wandsworth               676
 2 Westminster              441
 3 Kensington and Chelsea   297
 4 Hammersmith and Fulham   247
 5 Islington                227
 6 Richmond upon Thames     227
 7 Camden                   180
 8 Hackney                  165
 9 Tower Hamlets            156
10 Lambeth                  121
# ... with 17 more rows
```

The properties in the data-set is located in 27 different boroughs and “Wandsworth” has most of

the property transactions while “Bromley” has only one.

```
boxplot(london$ppsf ~ london$borough, horizontal = TRUE,  ylab= NULL, xlab = "Price per Square
axis(1,cex.axis=1)
```



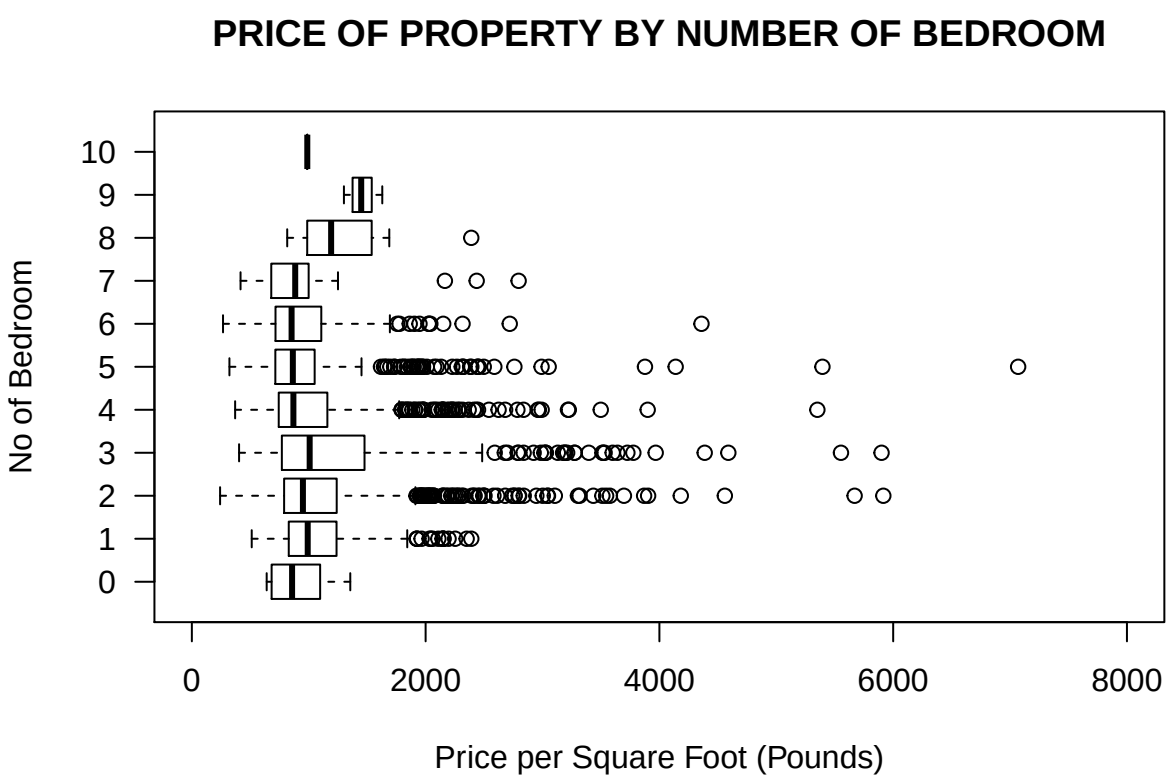
The above boxplot evaluated ppsf against different boroughs and it shown properties located in Westminster and Kensington and Chelsea generally have higher prices.

The borough which has the highest priced property is Westminster and the borough which has the lowest priced property is Tower Hamlets.

```
boxplot(london$ppsf ~ london$type, horizontal = TRUE, ylab = NULL, xlab = "Price per Square Foo
```

As shown in the boxplot above, "Penthouse" and "House" generally commanded higher prices than "Apartment" however the most expensive property is an apartment.

```
boxplot(london$ppsf ~ london$bedroom, horizontal = TRUE, ylab = "No of Bedroom", xlab = "Price
```



The boxplot above analysed ppsf with number of bedroom and it shown there is a positive correlation between the number of bedroom and property price as the higher the number of bedroom the higher the property price.

```
# Produce population table
temp<-london%>%select(borough, population)%>%distinct()
temp[order(temp$population),]
```

```
                borough population
6  Kensington and Chelsea     133000
11   Kingston upon Thames     147500
9    Richmond upon Thames     151100
5  Hammersmith and Fulham     160300
1                  Merton     181600
2               Islington     222300
4             Westminster     233200
23                 Harrow     242100
7                Haringey     242400
26         Waltham Forest     243700
27                Bromley     257900
8                  Camden     262400
12               Hounslow     266600
17              Greenwich     280800
```

```
3              Wandsworth     283300
13                Hackney     285300
21              Lewisham      296600
14             Hillingdon     303700
10                Lambeth     315000
18              Redbridge     316100
20              Southwark     321200
15          Tower Hamlets     332600
22                Croydon     341300
16                 Ealing     342200
19                 Barnet     343200
25                  Brent     358400
24                 Newham     383400
```

```
rm(temp)
```

The borough which has the lowest population is Kensington and Chelsea and the borough which has the highest population is 'r london$borough[which.max(london$population)]'. The population does not correspond with the highest and lowest property price boroughs.

```
# Produce population table
temp<-london%>%select(borough, unemployment_rate)%>%distinct()
temp[order(temp$unemployment_rate),]
```

```
                 borough unemployment_rate
3              Wandsworth         0.1648429
1                  Merton         0.2312775
9    Richmond upon Thames         0.2316347
27                Bromley         0.2318728
21              Lewisham         0.2329737
11   Kingston upon Thames         0.2359322
20              Southwark         0.2378580
17              Greenwich         0.2400285
10                Lambeth         0.2482540
22                Croydon         0.2510987
14             Hillingdon         0.2545275
2               Islington         0.2550607
5  Hammersmith and Fulham         0.2570181
7                Haringey         0.2578383
19                 Barnet         0.2578671
12              Hounslow         0.2636909
16                 Ealing         0.2670953
18              Redbridge         0.2745966
24                 Newham         0.2856025
23                 Harrow         0.2866584
15          Tower Hamlets         0.3024654
13                Hackney         0.3028391
```

```
26         Waltham Forest         0.3065244
6  Kensington and Chelsea         0.3135338
25                  Brent         0.3147321
8                  Camden         0.3399390
4              Westminster         0.3524871
```

```
rm(temp)
```

The borough which has the lowest unemployment rate is Wandsworth and the borough which has the highest unemployment rate is Westminster. The unemployment rates do not correspond with the highest and lowest property price boroughs.

```
# Produce crime rate
temp<-london%>%select(borough, crime_rate)%>%distinct()
temp[order(temp$crime_rate),]
```

```
                  borough crime_rate
23                 Harrow  0.1329698
18              Redbridge  0.1478804
1                  Merton  0.1498789
11   Kingston upon Thames  0.1576475
25                  Brent  0.1602037
9    Richmond upon Thames  0.1605559
19                 Barnet  0.1659237
14             Hillingdon  0.1666480
16                 Ealing  0.1740152
24                 Newham  0.1765310
27                Bromley  0.1772431
3              Wandsworth  0.1783445
12               Hounslow  0.1852551
21               Lewisham  0.1867195
17              Greenwich  0.1908226
26         Waltham Forest  0.1919573
22                Croydon  0.1948139
15          Tower Hamlets  0.1960824
10                Lambeth  0.2044032
20              Southwark  0.2120766
13                Hackney  0.2190361
2               Islington  0.2362798
8                  Camden  0.2398056
7                Haringey  0.2433663
5  Hammersmith and Fulham  0.2549220
6  Kensington and Chelsea  0.3016541
4              Westminster  0.4790523
```

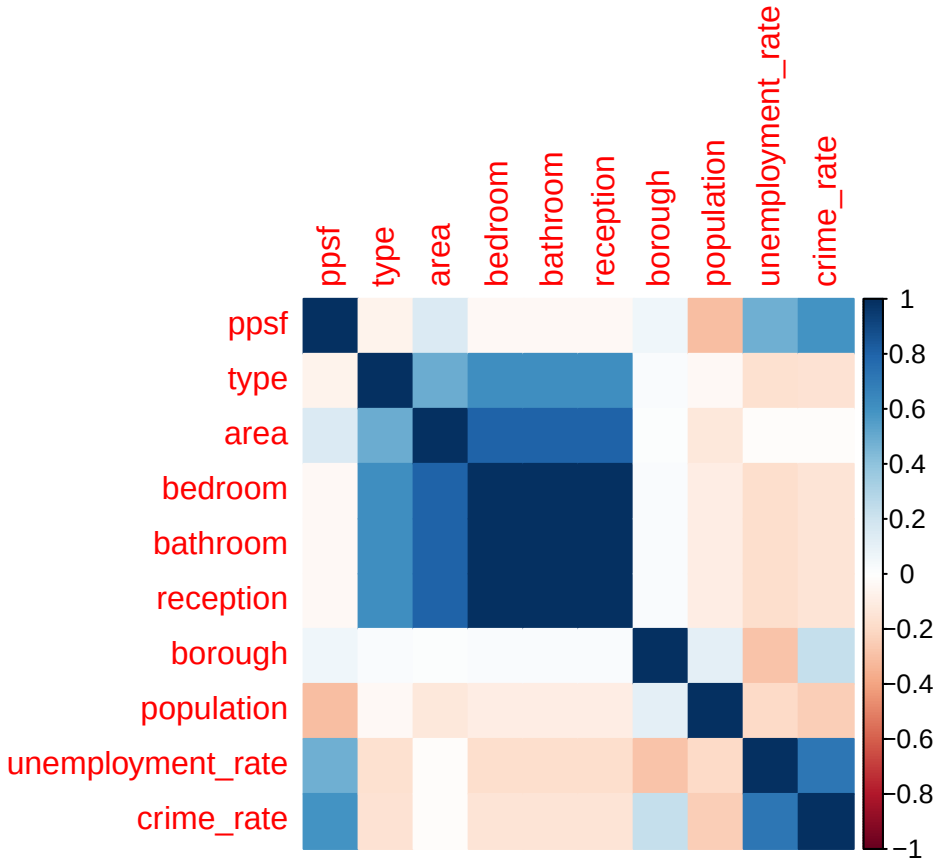
```
rm(temp)
```

The borough which has the lowest crime rate is Harrow and the borough which has the highest crime rate is Westminster. The crime rates do not correspond with the highest and lowest property price boroughs.

After explored the data in "london" data-set, a correlation matrix is produced for a preliminary analysis of the correlation coefficients between different variables using " corrplot" package.

```
#library(ggcorrplot)
#model.matrix(~0+., data=london) %>%
  #cor(use="pairwise.complete.obs") %>%
  #ggcorrplot(show.diag = F, type="lower", lab=F)

# Prepare data set for coorelation matrix and assign number to type and borough
london_cor<-london
#london_cor$type<-as.numeric(london_cor$type)
#london_cor$borough<-as.numeric(london_cor$borough)

# london_cor$type[london_cor$type == "Apartment"] <- 1
# london_cor$type[london_cor$type == "Penthouse"] <- 2
# london_cor$type[london_cor$type == "House"] <- 3

# london_cor$borough[london_cor$borough == "Barnet"] <- 1
# london_cor$borough[london_cor$borough == "Brent"] <- 2
# london_cor$borough[london_cor$borough == "Bromley"] <- 3
# london_cor$borough[london_cor$borough == "Camden"] <- 4
# london_cor$borough[london_cor$borough == "Croydon"] <- 5
# london_cor$borough[london_cor$borough == "Ealing"] <- 6
# london_cor$borough[london_cor$borough == "Greenwich"] <- 7
# london_cor$borough[london_cor$borough == "Hackney"] <- 8
# london_cor$borough[london_cor$borough == "Hammersmith and Fulham"] <- 9
# london_cor$borough[london_cor$borough == "Haringey"] <- 10
# london_cor$borough[london_cor$borough == "Harrow"] <- 11
# london_cor$borough[london_cor$borough == "Hillingdon"] <- 12
# london_cor$borough[london_cor$borough == "Hounslow"] <- 13
# london_cor$borough[london_cor$borough == "Islington"] <- 14
# london_cor$borough[london_cor$borough == "Kensington and Chelsea"] <- 15
# london_cor$borough[london_cor$borough == "Kingston upon Thames"] <- 16
# london_cor$borough[london_cor$borough == "Lambeth"] <- 17
# london_cor$borough[london_cor$borough == "Lewisham"] <- 18
# london_cor$borough[london_cor$borough == "Merton"] <- 19
# london_cor$borough[london_cor$borough == "Newham"] <- 20
# london_cor$borough[london_cor$borough == "Redbridge"] <- 21
# london_cor$borough[london_cor$borough == "Richmond upon Thames"] <- 22
# london_cor$borough[london_cor$borough == "Southwark"] <- 23
# london_cor$borough[london_cor$borough == "Tower Hamlets"] <- 24
# london_cor$borough[london_cor$borough == "Waltham Forest"] <- 25
```

```r
# london_cor$borough[london_cor$borough == "Wandsworth"] <- 26
# london_cor$borough[london_cor$borough == "Westminster"] <- 27


#Process Correlation Matrix
london_cor$type<-as.numeric(london_cor$type)
london_cor$borough<-as.numeric(london_cor$borough)
corlondon<-cor(london_cor)
corrplot(corlondon, method="color")
```



The correlation matrix above shown that the price is positively affected by the crime rate and unemployment rate followed by area and borough. Population has negative correlation with the price and other variables, number of bedroom, bathroom and reception had no effect on the price according to the correlation matrix.

#4 Model Analysis

In order to test whether demographic data affects the price of the properties and hence the accuracy of the prediction, the models are divided into two sets for the analysis: one without the demographic data and another with the demographic data.

The two sets of models will use the following algorithms:

1. Linear Regression

2. Decision Tree
3. Random Forrest

The RMSE scores generated from the two sets of models using the three algorithms will be compared and analysed to deduce the findings for this project.

4.1 Data Prepration

The London data-set is split into "test_set" and "train set". "train_set" will be used to construct the models and then "test_set" will be used to verify the models and compute the RMSE scores. "test_set" is 10% of "london" while "train_set" is 90%.

```
# Splitting london data-set into traning set and test set.  Test set is 10% of london data-set
set.seed(2)
test_index <- createDataPartition(london$ppsf, times = 1, p = 0.1, list = FALSE)
train_set <- london[-test_index,]
test_set <- london[test_index,]

# Remove temporary files to tidy environment
rm(test_index)
```

4.2 Baseline Model

Baseline Model is produced by calculating the average ppsf and its RMSE is used as the baseline for the results.

```
###################
# Baseline Model #
###################

mu<-mean(train_set$ppsf)
rmse0<-RMSE(test_set$ppsf, mu)
```

The average ppsf is 1102.6519188 and it resulted a RMSE score of 518.3801574.

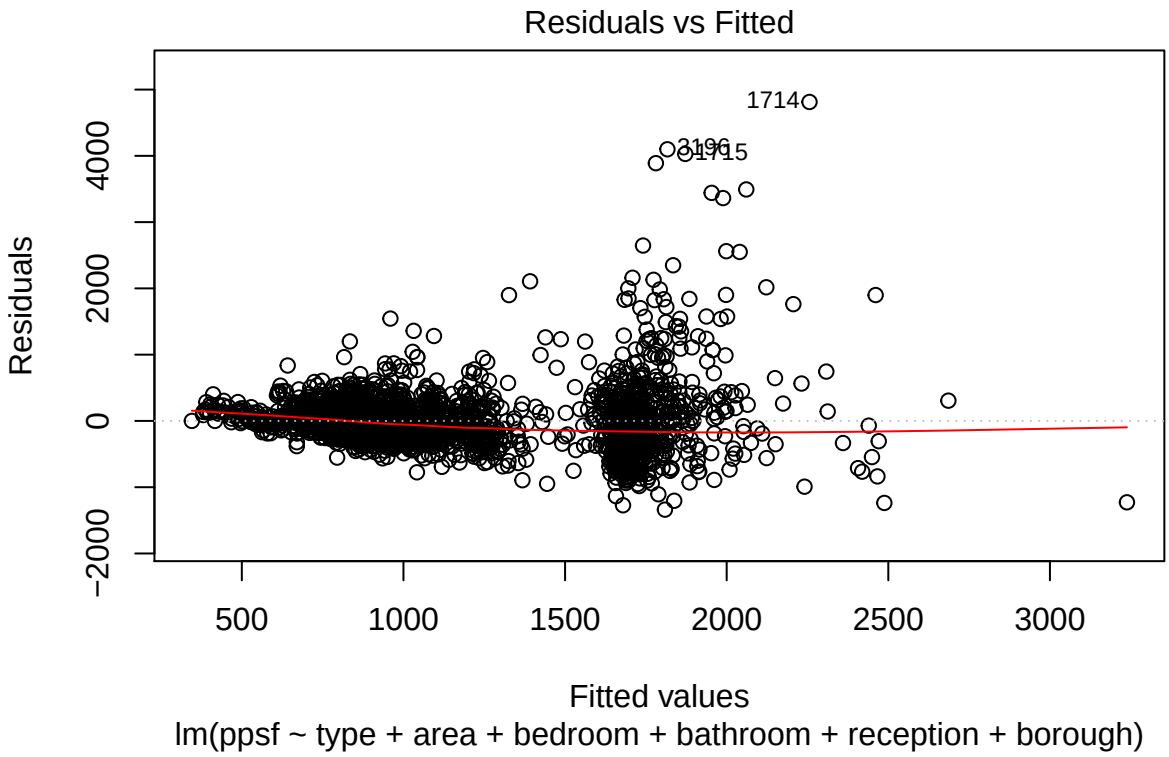
##4.3 Models Without Demographic Data

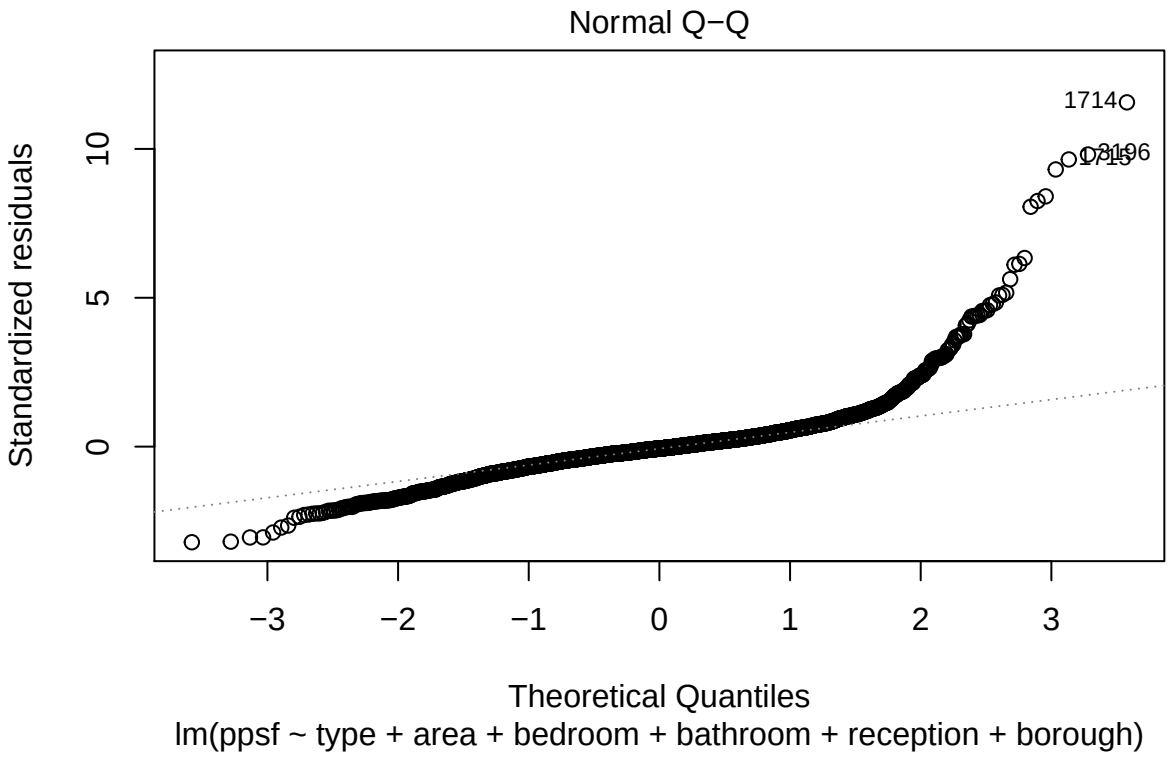
The first set of models - Model 1A, 1B and 1C will be constructed without using the demographic data.

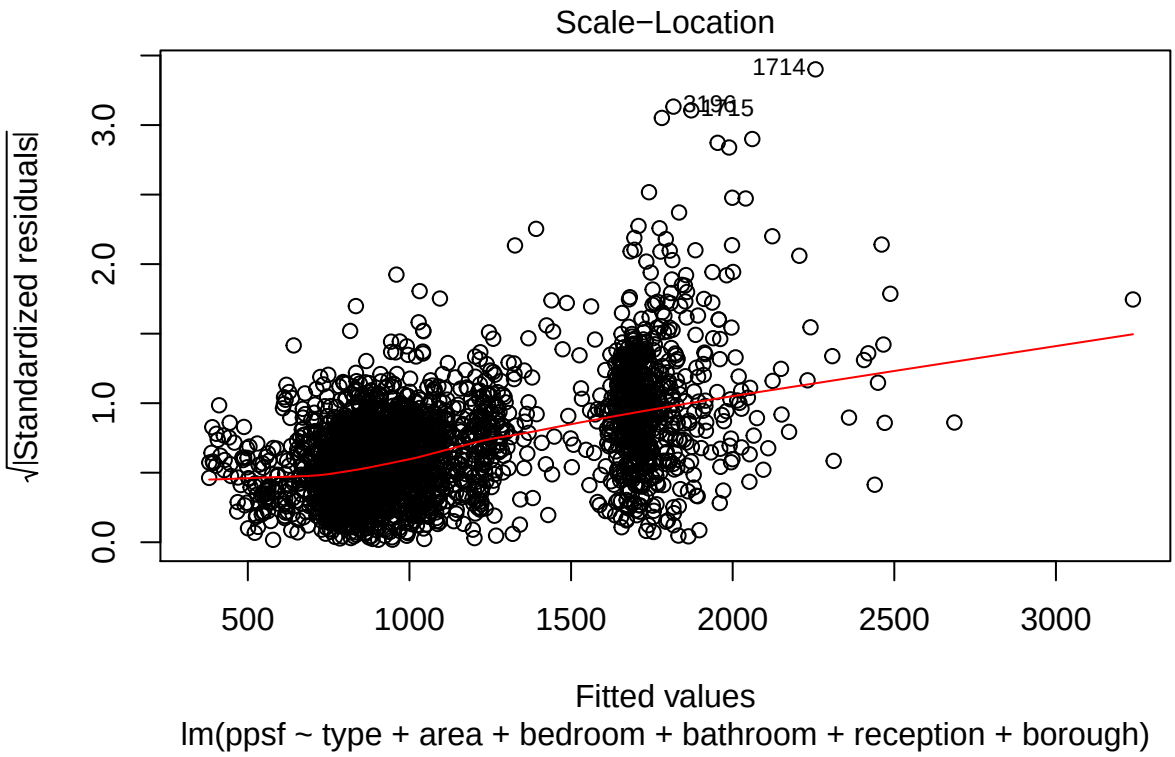
###4.3.1 Model 1A Linear Regression Without Demographic Data

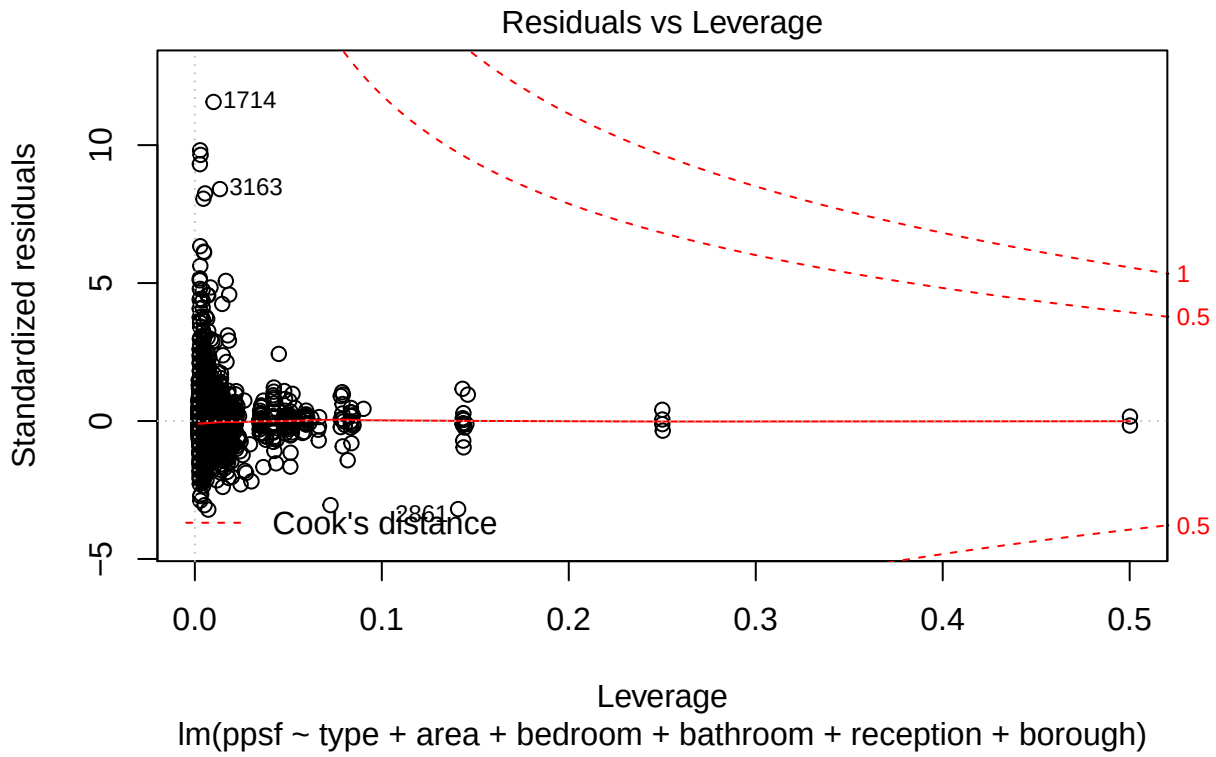
Model 1A used linear regression as the machine algorithm.

```
# Model 1A Linear Regression Without Demographic DataLinear Regression Computation
lr <- lm(ppsf ~ type+area+bedroom+bathroom+reception+borough, data = train_set)
summary(lr)
plot(lr)
```


Residuals vs Fitted
1714
3196
1715
Residuals
4000
2000
0
−2000
500
1000
1500
2000
2500
3000
Fitted values
lm(ppsf ~ type + area + bedroom + bathroom + reception + borough)


Normal Q−Q
Standardized residuals
0
5
10
1714
1715
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(ppsf ~ type + area + bedroom + bathroom + reception + borough)


Scale−Location
1714
3196
1715
√|Standardized residuals|
0.0
1.0
2.0
3.0
500
1000
1500
2000
2500
3000
Fitted values
lm(ppsf ~ type + area + bedroom + bathroom + reception + borough)




```
model1a <- predict(lr, newdata = test_set)

# Compute RMSE for Model 1A
rmse1a<-RMSE(test_set$ppsf, model1a)

#Computate the accuracy for Model 1
#acc1<-confusionMatrix(model1, test_set$price, type="response")
#acc1<-acc1$overall["Accuracy"]
```

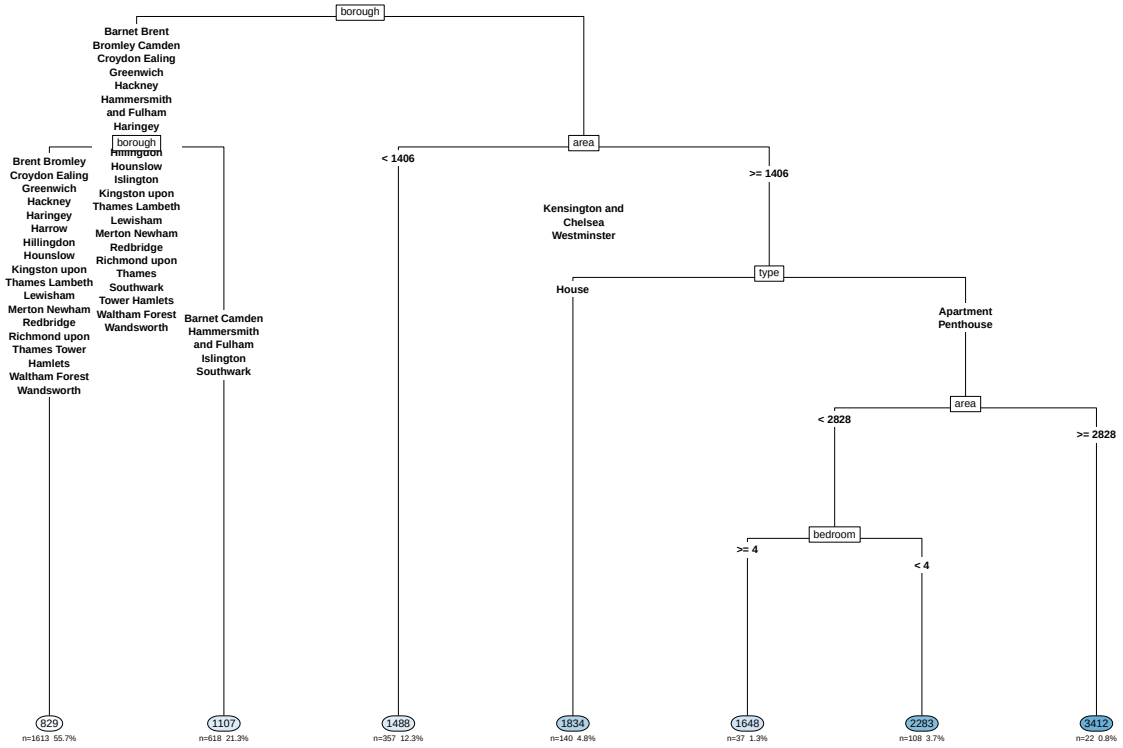
The RMSE score from Model 1A is 333.304403 which is better than the baseline model as expected.

4.3.2 Model 1B Decision Tree Without Demographic Data

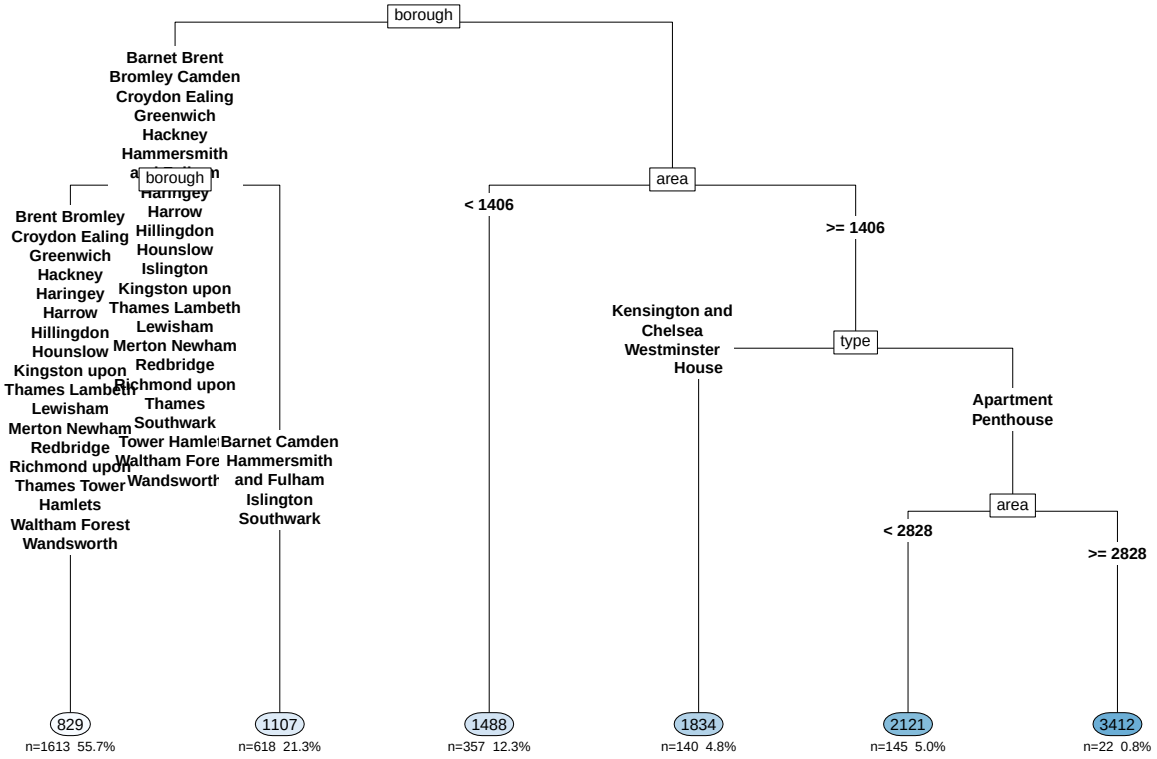
Model 1B used Decision Tree as the machine learning algorithm. The algorithm included pruning process to reduce over-fitting and to try to improve the accuracy.

```
# Model 1B Decision Tree Without Demographic Data Computation

# Model 1B Decision Tree Computation
train_rpart1b <- rpart(ppsf ~ type+area+bedroom+bathroom+reception+borough, data = train_set)
summary(train_rpart1b)
```

```
# Split text display
split.fun <- function(x, labs, digits, varlen, faclen)
{
# replace commas with space
labs <- gsub(",", " ", labs)
for(i in 1:length(labs)) {

# split labs[i] into multiple lines
labs[i] <- paste(strwrap(labs[i], width = 15), collapse = "\n")
}
labs
}

rpart.plot(train_rpart1b, faclen = 0, type=5, yesno=T, clip.facs = TRUE, under = T, digits=-3,
```




```
# Prune the Tree
pfit1b<- prune(train_rpart1b, cp=train_rpart1b$cptable[which.min(train_rpart1b$cptable[,"xerro
rpart.plot(pfit1b, faclen = 0, type=5, yesno=T, clip.facs = TRUE, under = T, digits=-3, extra=
```




```
model1b <- predict(pfit1b, newdata=test_set)

# Compute RMSE for Model 1B
rmse1b<-RMSE(test_set$ppsf, model1b)

#Computate the accuracy for Model 2
#acc2<-confusionMatrix(as.factor(model2), as.factor(test_set$ppsf))
#acc2
```

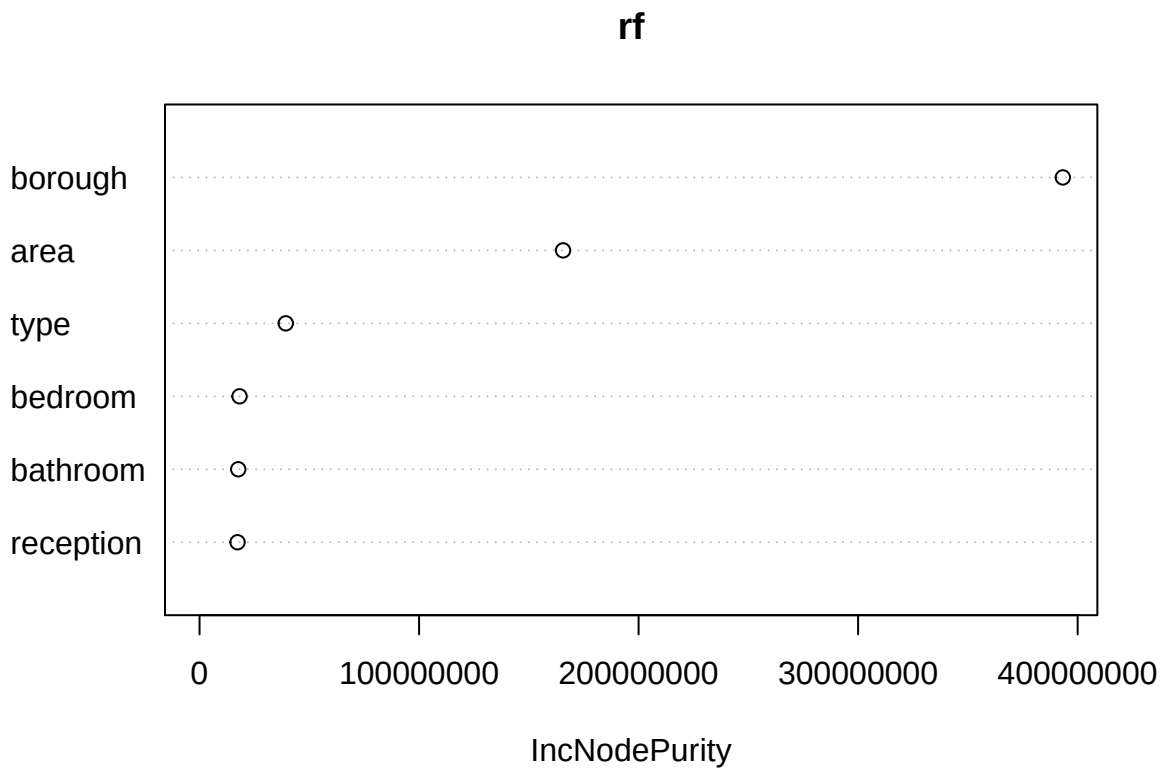
Model 1B achieved a RMSE score of 337.5090796. The performance of Model 1B is not as accuracy as Model 1A and the resulted decision tree shown "borough" as the most significant variable followed by "area" and "type". It is also noted that the pruning had no effect on Model 1B.

4.3.3 Model 1C Random Forest Without Demographic Data

Model 1C used Random Forest as the algorithm.

```
#Model 1C Random Forest Without Demographic Data Computation

rf<-randomForest(ppsf~type+area+bedroom+bathroom+reception+borough, data=train_set)
model1c <- predict(rf, newdata=test_set)
```

```
importance  <- importance(rf)
varImpPlot(rf)
```




```
# Compute RMSE for Model 1C
rmse1c<-RMSE(test_set$ppsf, model1c)

#Computate the accuracy for Model 3
#acc3<-confusionMatrix(model3, test_set$price)
#acc3
```

Model 1C resulted a RMSE score of 324.4682626. It performed better than previous models. From the result, "borough" is the most significant variable followed by "area" and "type".

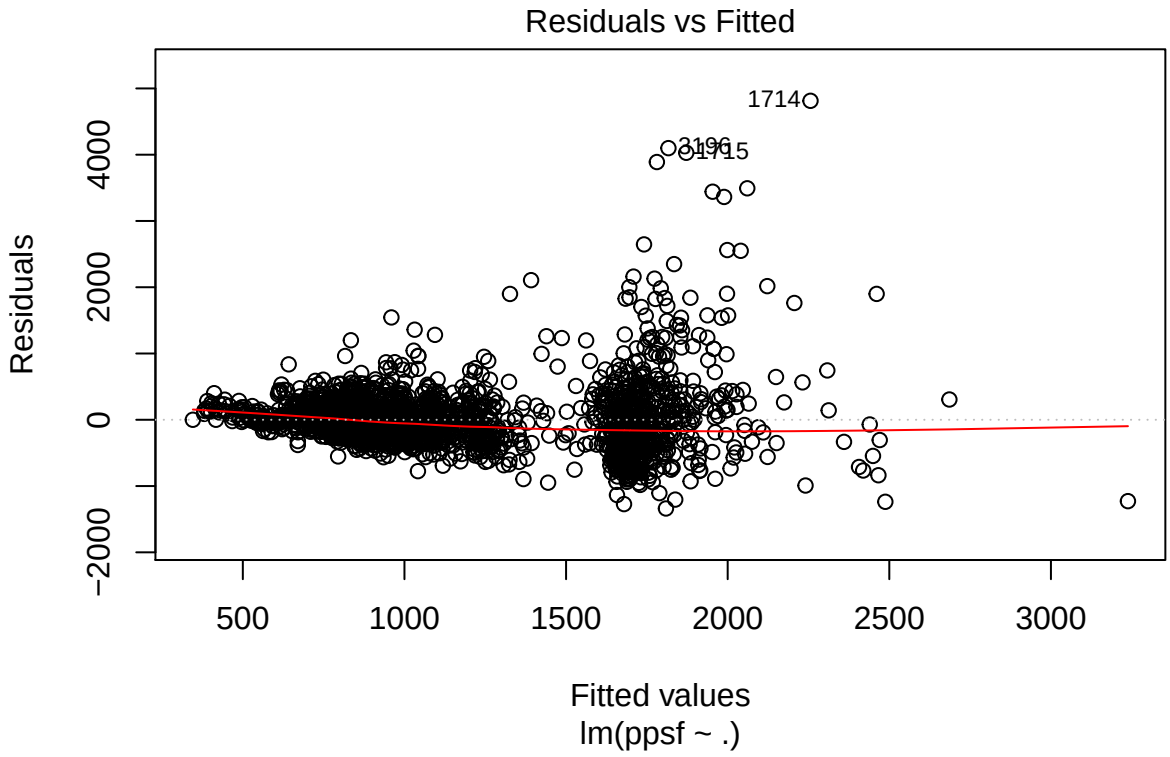
4.4 Model With Demographic Data

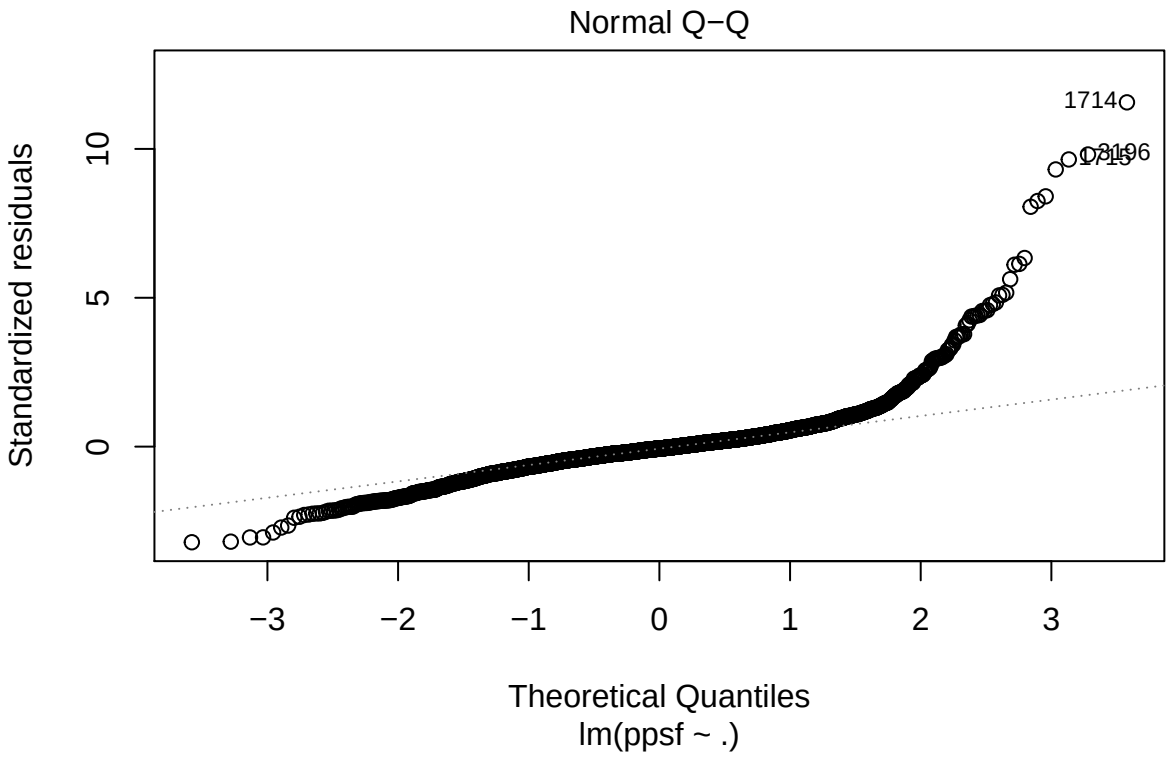
The second set of models, Model 2A, Model 2B and Model 2C, included demographic data in the machine learning algorithms.

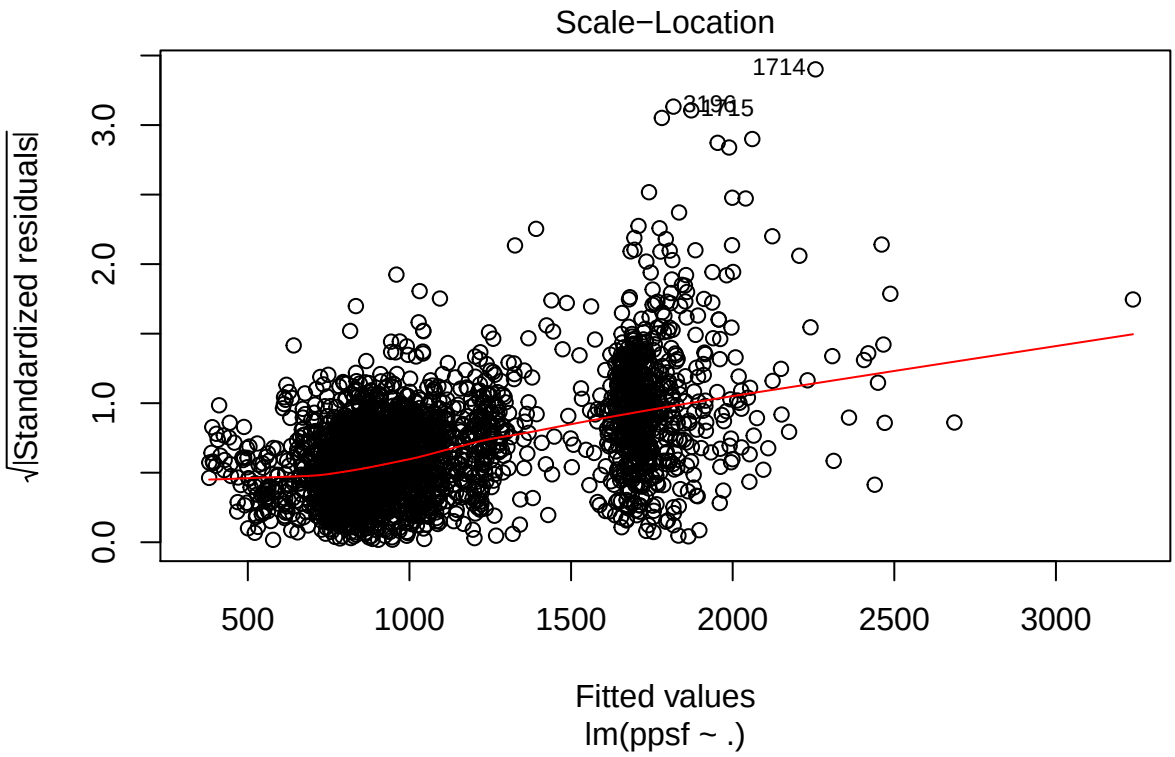
4.4.1 Model 2A Linear Regression With Demographic Data

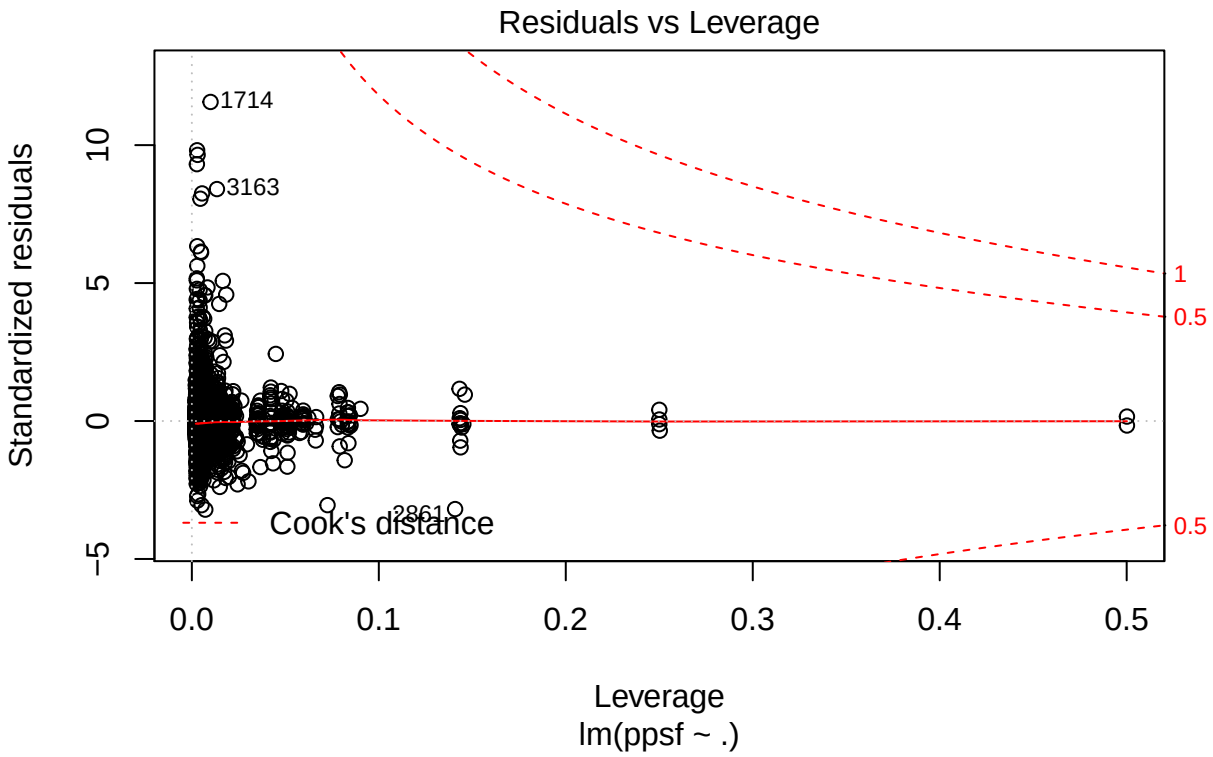
Model 2A used linear regression as the machine algorithm.

```r
# Model 2A  - Linear Regression With Demographic Data Computation
lr <- lm(ppsf ~ ., data = train_set)
summary(lr)
plot(lr)
```




Normal Q−Q
Standardized residuals
0
5
10
1714
2196
1715
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(ppsf ~ .)


Scale−Location
1714
3196
1715
√|Standardized residuals|
0.0
1.0
2.0
3.0
500
1000
1500
2000
2500
3000
Fitted values
lm(ppsf ~ .)




```
model2a <- predict(lr, newdata = test_set)

# Compute RMSE for Model 2A
rmse2a<-RMSE(test_set$ppsf, model2a)

# Computate the accuracy for Model 1
#acc1<-confusionMatrix(model1, test_set$price, type="response")
#acc1<-acc1$overall["Accuracy"]
```

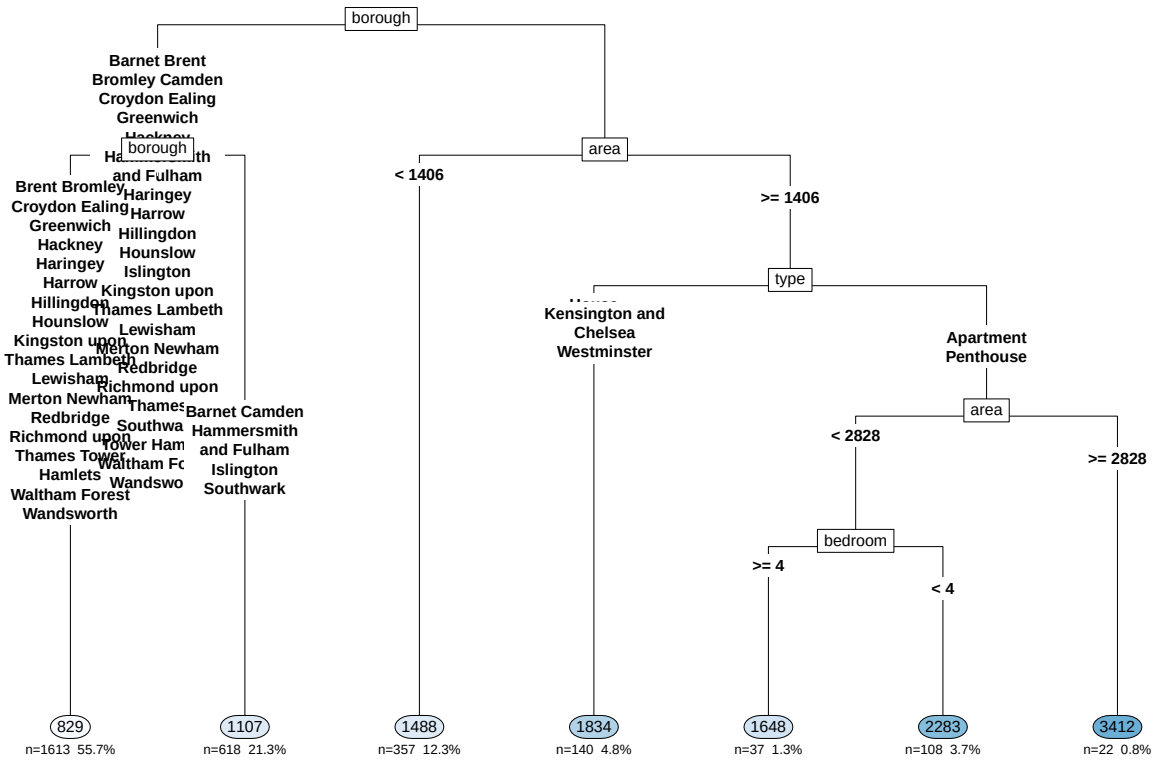
The RMSE score from Model 2A is 333.304403 which is exactly the same as Model 1A. Demographic data had no effect on the computation of linear regression.

4.4.2 Model 2B Decision Tree With Demographic Data

Model 2B used Decision Tree with pruning for the algorithm.

```
# Model 2B - Decision Tree With Demographic Data Computation

# Model 2B Decision Tree Computation
train_rpart2b <- rpart(ppsf ~ ., data = train_set)
summary(train_rpart2b)
rpart.plot(train_rpart2b, faclen = 0, type=5, yesno=T, clip.facs = TRUE, under = T, digits=-3,
```

```
# Prune the Tree
pfit2b<- prune(train_rpart2b, cp=train_rpart2b$cptable[which.min(train_rpart2b$cptable[,"xerro
rpart.plot(pfit2b, faclen = 0, type=5, yesno=T, clip.facs = TRUE, under = T, digits=-3, extra=
```



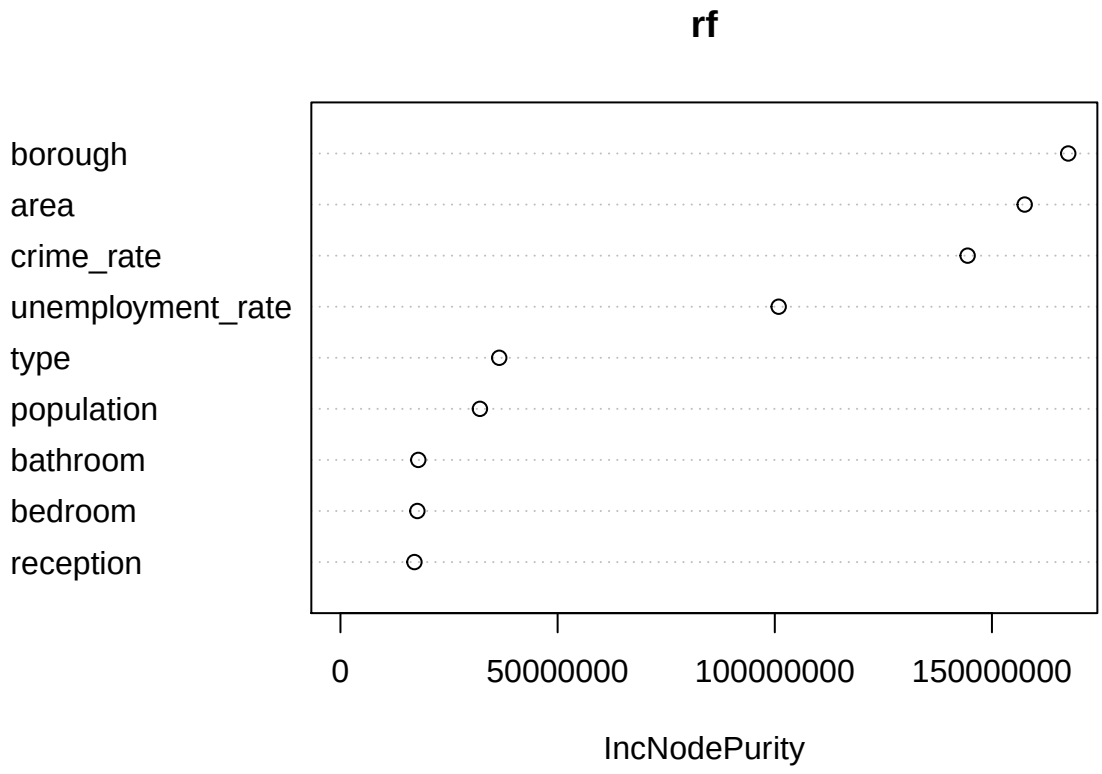

```
model2b <- predict(pfit2b, newdata=test_set)

# Compute RMSE for Model 2B
rmse2b<-RMSE(test_set$ppsf, model2b)

#Computate the accuracy for Model 2
#acc2<-confusionMatrix(as.factor(model2), as.factor(test_set$ppsf))
#acc2
```

The resulted RMSE score for Model 2B is 331.7379732. The result is better than the decision tree model without demographic data - Model 1B. The significant variables are “borough”, “area”, “type” and “bedroom” which are similar to Model 1B and the demographic data is not significant.

4.4.3 Model 2C Random Forest With Demographic Data

Model 2C used Random Forest for the algorithm.

```r
#Model 2C - Random Forest With Demographic Data Computation
rf<-randomForest(ppsf~., data=train_set)
model2c <- predict(rf, newdata=test_set)
importance <- importance(rf)
varImpPlot(rf)
```




```r
rmse2c<-RMSE(test_set$ppsf, model2c)

#Computate the accuracy for Model 2C
#acc3<-confusionMatrix(model3, test_set$price)
#acc3
```

Model 2C achieved a RMSE score of 329.0970403 which is the best in the models with demographic data. From the result it is show "borough" as the most significant variable in the prediction followed by "area", "crime_rate", "unemployment_rate". "population" and "type" also had reasonable in the algorithm.

#5 Results

```r
#Comply results into a summary table
summary <- tibble(Model = c("Baseline Model", "Model 1A Linear Regression Without Demographic
summary
```

```
# A tibble: 7 x 3
  Model                                          RMSE `Difference to Baseline Mo~
  <chr>                                         <dbl>                        <dbl>
1 Baseline Model                                 518.                           0
2 Model 1A Linear Regression Without Demograp~  333.                         185.
3 Model 1B Decision Tree Without Demographic ~  338.                         181.
4 Model 1C Random Forest Without Demographic ~  324.                         194.
5 Model 2A Linear Regression With Demographic~  333.                         185.
6 Model 2B Decision Tree With Demographic Data  332.                         187.
7 Model 2C Random Forest With Demographic Data  329.                         189.
```

The results are shown in the summary table above and the key findings are:

- The most accurate prediction is Random Forest Without Demographic Data
- Demographic data made no difference in linear regression model and therefore it is irrelevant in the algorithm
- Demographic data improved the accuracy in Decision Tree model
- Although the demographic data shown as important variables in Random Forest model however the accuracy reduced
- Borough is the most important variable which confirmed the location contribute to housing price
- Despite the size factor was taken away by using price per sq ft, however the result shows the larger the property the higher in price per sq ft.
- All the models did not result satisfactory accuracy as the varies in the range of GBP300 is large considering the least expensive property is only GBP241 per sq ft.

#6 Conclusion

From the three different machine learning models tested, random Forest Model provided the most accurate prediction. It is also noted that demographic data performed differently in different algorithms as it helped to improve the prediction with decision tree algorithm but reduced the accuracy with Random Forest.

Overall the performances of all the algorithms do not give accurate predictions and this can be improved by increasing the number of transaction record.