

# CYO London Housing

Toby Wong

Jun 2021

## 1 Introduction

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

The objective of this project is to compare the performance of selected machine learning algorithms for housing price prediction and determine whether the inclusion of demographic data (population, unemployment rate and crime rate) can improve the accuracy of the housing price prediction.

Residual mean squared error (RMSE) score is used as the performance indicator for different algorithms 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))  
}
```

This report is divided into the following sections: 1 Introduction 2 London Housing Data-set 3 Data Exploration 4 Model Analysis 5 Results 6 Conclusion

## 2 London Housing Prices Data Set

This project uses the housing prices data in London for the algorithms and the data was originally downloaded from Kaggle (<https://www.kaggle.com/arnavkulkarni/housing-prices-in-london>) and subsequently uploaded to Github. Additional demographic data is downloaded from London Data Store (<https://data.london.gov.uk>).

After downloaded the data, the following process is performed to prepare the data-set for the machine learning models:

1. Verify and add the borough data using PostcodesioR package
2. Combine housing data with demographic data
3. Calculate price per square foot (ppsf), unemployment rate and crime rate
4. Simplify the types of property
5. Change the data to appropriate classes
6. Tidy up the data-set and remove un-needed data
7. 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("dplyr", 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.org")
if(!require(PostcodesioR)) install.packages("PostcodesioR", repos = "http://cran.us.r-project.org")
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 package
# 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 variables 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 variables to appropriate types
```

```

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 Hackney
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$...8+london_unemp
london_unemployment$employment<-london_unemployment$'working age employment rate - white'+lond
london_unemployment$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"))%>%

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

```

```

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

# Add population, unemployment rate, and crime rate to "London" data-set
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 temporary files
rm(temp_data, temp_info, x, london_e_c, london_crime, london_unemployment)

```

### 3 Data Exploration

Before building the machine learning models, data exploration is carried out on “london” data-set to understand the data structure and the variables it contained. This process helps to determine how to build the models for the analysis.

```

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

```

ppsf	type	area	bedroom	bathroom
Min. : 242	Apartment:1872	Min. : 274	Min. : 0	Min. : 0
1st Qu.: 763	House :1251	1st Qu.: 816	1st Qu.: 2	1st Qu.: 2
Median : 935	Penthouse: 96	Median : 1250	Median : 3	Median : 3
Mean :1102		Mean : 1601	Mean : 3	Mean : 3
3rd Qu.:1250		3rd Qu.: 2062	3rd Qu.: 4	3rd Qu.: 4
Max. :7069		Max. :14358	Max. :10	Max. :10

reception	borough	population	unemployment_rate
Min. : 0	Wandsworth : 676	Min. :133000	Min. :0.16
1st Qu.: 2	Westminster : 441	1st Qu.:181600	1st Qu.:0.23
Median : 3	Kensington and Chelsea: 297	Median :262400	Median :0.26
Mean : 3	Hammersmith and Fulham: 247	Mean :240488	Mean :0.26
3rd Qu.: 4	Islington : 227	3rd Qu.:283300	3rd Qu.:0.31
Max. :10	Richmond upon Thames : 227	Max. :383400	Max. :0.35
	(Other) :1104		

crime_rate
Min. :0.13
1st Qu.:0.18
Median :0.20

```
Mean      :0.25
3rd Qu.   :0.25
Max.      :0.48
```

From the summary above, it is noted that “london” data-set 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 6 rows of "london"
head(london)
```

	ppsf	type	area	bedroom	bathroom	reception	borough	population
1	617	House	2716	5	5	5	Merton	181600
2	799	Apartment	814	2	2	2	Islington	222300
3	966	Apartment	761	2	2	2	Wandsworth	283300
4	889	House	1986	4	4	4	Wandsworth	283300
5	964	Apartment	700	2	2	2	Wandsworth	283300
6	1042	Apartment	403	1	1	1	Westminster	233200

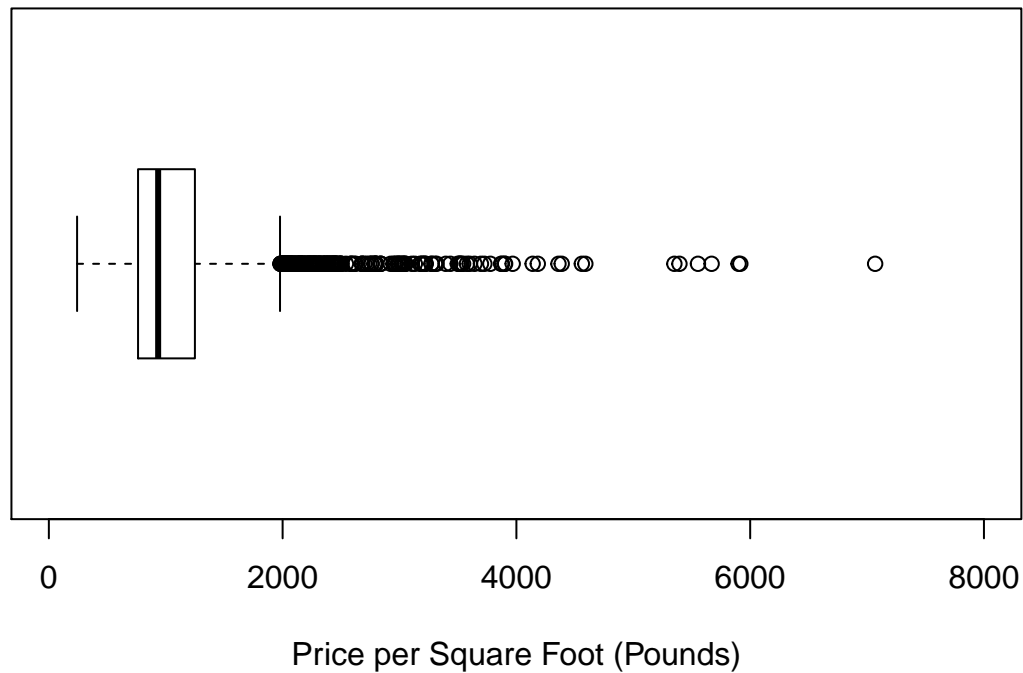
	unemployment_rate	crime_rate
1	0.23	0.15
2	0.26	0.24
3	0.16	0.18
4	0.16	0.18
5	0.16	0.18
6	0.35	0.48

The first 6 rows shown that all variables are in numeric class except “type” and “borough”, which are in factor class.

The most important variable in the data-set is price per square foot (ppsf) and it is the prediction of the algorithms used in this project. By using ppsf, the prediction takes away the factor of the size of the property to provide a more accurate reflection of the property price.

```
# Boxplot of ppsf
boxplot(london$ppsf, horizontal = TRUE, xlab = "Price per Square Foot (Pounds)", main = "PRICE
```

## PRICE OF PROPERTIES



The ppsf boxplot above shown a range from GBP 241.61 per sq ft to GMP 7069.18 per sq ft and it also shown majority of the property is around GBP 1000 per sq ft.

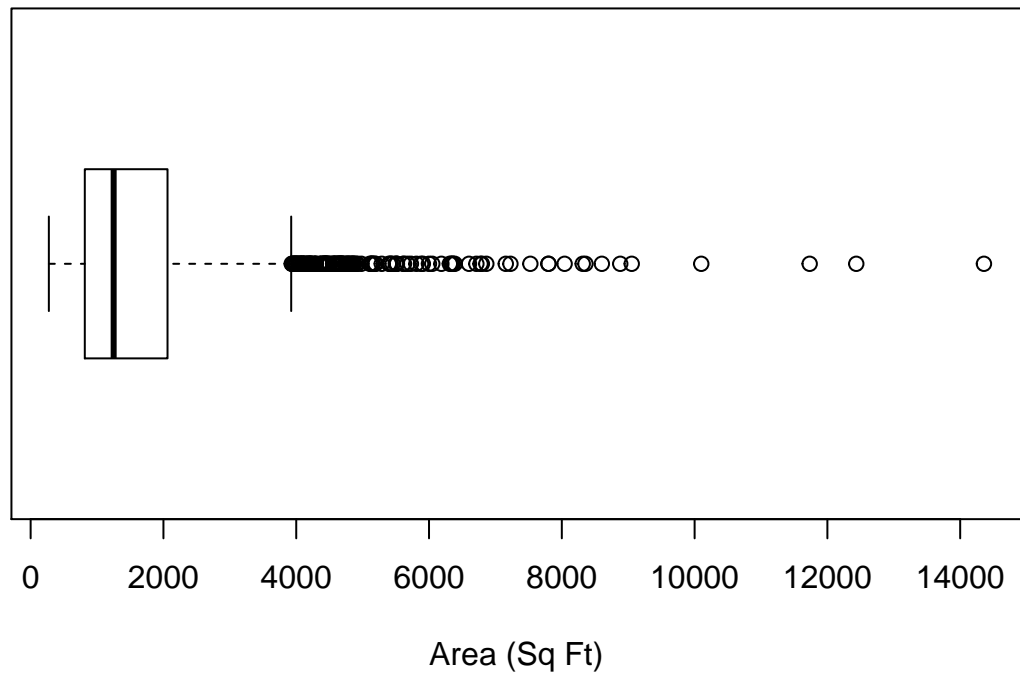
```
# Types of property
london%>%group_by(type) %>%
  summarize(count = n(), .groups='drop') %>%
  arrange(desc(count))
```

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

The properties are sorted into three different types as shown in the table above. “Apartment” and “House” are the two predominant types while much lesser properties are “Penthouse” in the data-set.

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

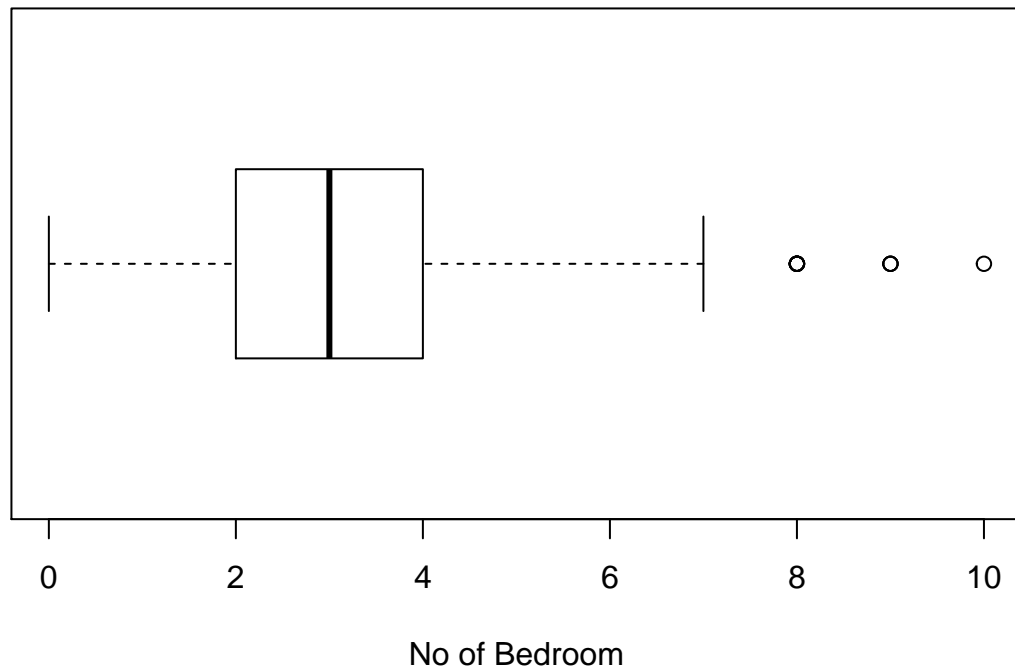
## SIZE OF PROPERTIES



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

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

## BEDROOM NUMBER



The bedroom boxplot demonstrated that most of the properties has two to four bedrooms and the maximum number of bedroom is ten, minimum is zero.

```
# Property Transactions in Different Boroughs
london%>%group_by(borough) %>%
  summarize(count = n(), .groups='drop') %>%
  arrange(desc(count))
```

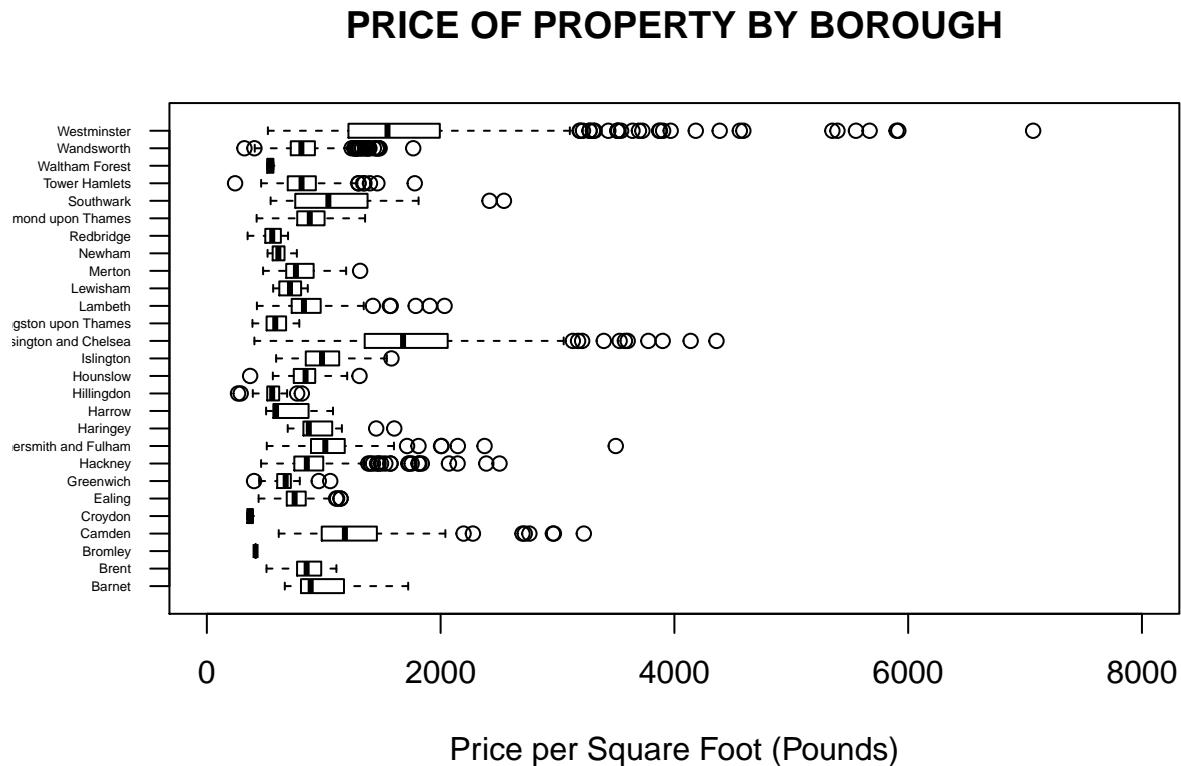
```
# A tibble: 27 x 2
  borough      count
  <fct>      <int>
1 Wandsworth    676
2 Westminster   441
3 Kensington and Chelsea 297
4 Hammersmith and Fulham 247
5 Islington     227
6 Richmond upon Thames 227
7 Camden        180
8 Hackney       165
9 Tower Hamlets 156
10 Lambeth      121
# ... with 17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



The properties in the data-set is located in 27 different boroughs. “Wandsworth” has most of the property transactions while “Bromley” has only one.

```
# ppsf of Property by Borough
```

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



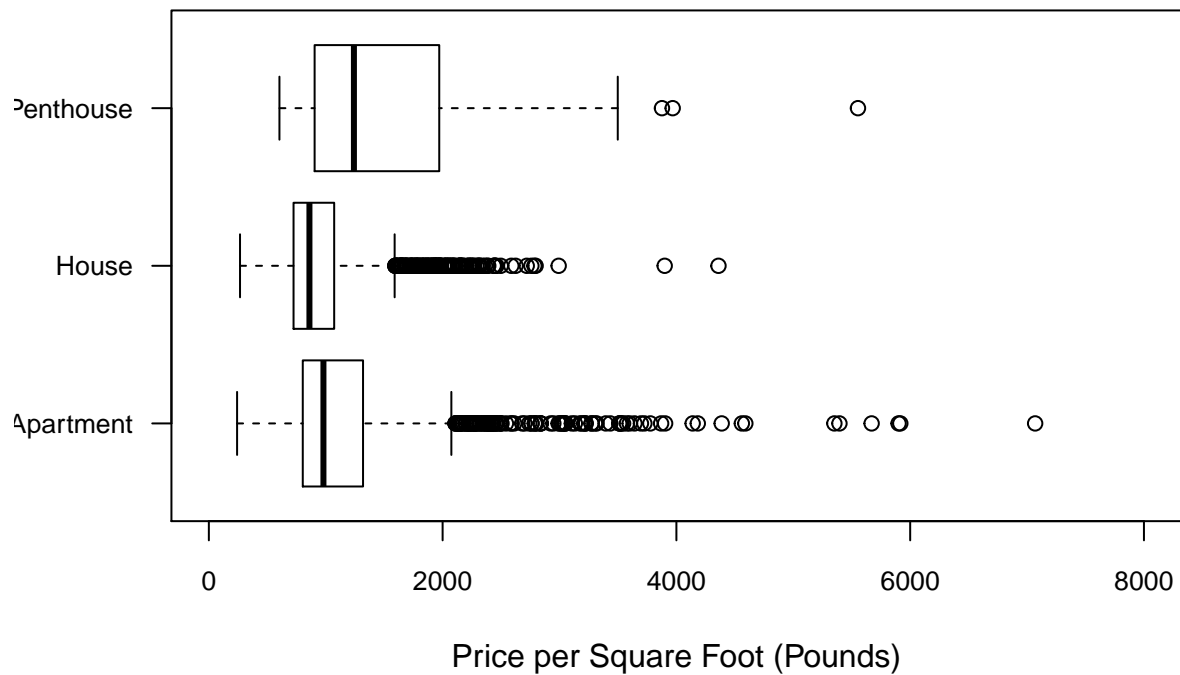
The above boxplot shown ppsf in different boroughs and properties located in “Westminster” and “Kensington and Chelsea” generally have higher prices.

It also shown the highest priced property is located in “Westminster” and the borough which has the lowest priced property is “Tower Hamlets”.

```
# ppsf by Property Type
```

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

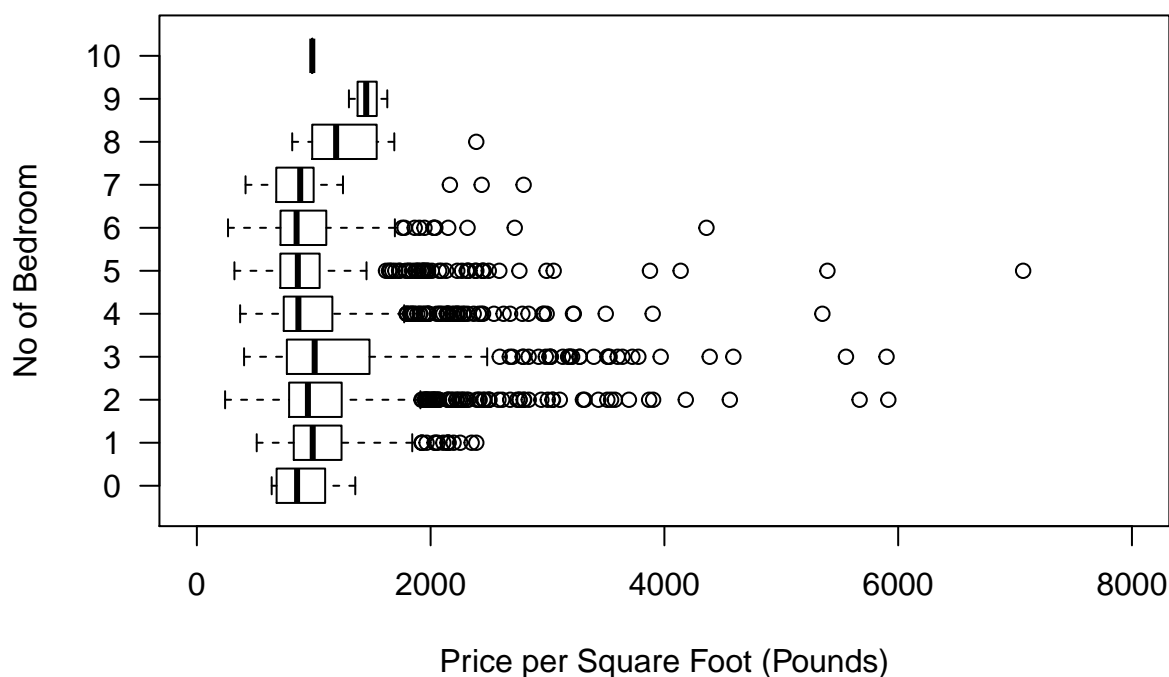
## PRICE OF PROPERTY BY PROPERTY TYPE



The boxplot above shown ppsf with different property types. “Penthouse” and “Apartment” are generally more expensive than “House” and the most expensive property is “Apartment”.

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

## PRICE OF PROPERTY BY NUMBER OF BEDROOM



The price by number of bedroom boxplot above shown majority of the property prices are not affected by the number of bedroom. It is also noted that highest priced property has five bedrooms.

*# Population in different boroughs*

```
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” while the borough which has the highest population is “Newham”. The population does not seem to correspond to the property prices.

```
# Unemployment Rate in Different Boroughs
```

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

	borough	unemployment_rate
3	Wandsworth	0.16
1	Merton	0.23
9	Richmond upon Thames	0.23
27	Bromley	0.23
21	Lewisham	0.23
11	Kingston upon Thames	0.24
20	Southwark	0.24
17	Greenwich	0.24
10	Lambeth	0.25
22	Croydon	0.25
14	Hillingdon	0.25
2	Islington	0.26
5	Hammersmith and Fulham	0.26
7	Haringey	0.26
19	Barnet	0.26
12	Hounslow	0.26
16	Ealing	0.27
18	Redbridge	0.27
24	Newham	0.29
23	Harrow	0.29
15	Tower Hamlets	0.30
13	Hackney	0.30
26	Waltham Forest	0.31

6	Kensington and Chelsea	0.31
25	Brent	0.31
8	Camden	0.34
4	Westminster	0.35

```
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 also do not seem to correspond with the property prices.

```
# Crime Rate in Different Boroughs
```

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

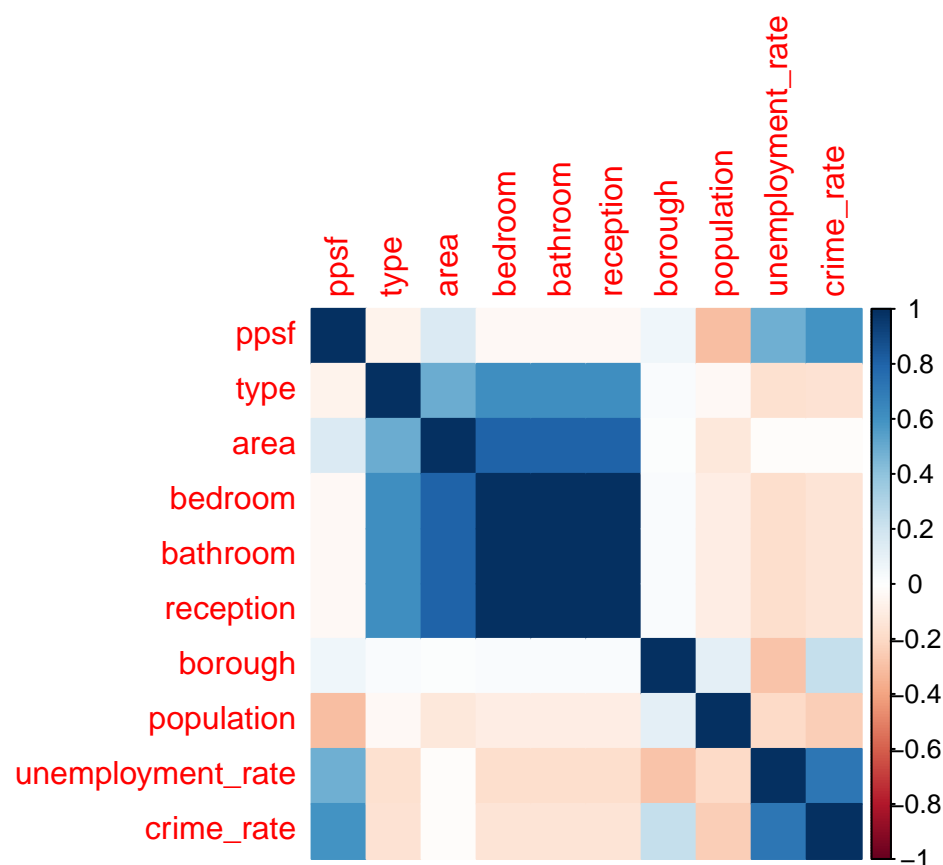
	borough	crime_rate
23	Harrow	0.13
18	Redbridge	0.15
1	Merton	0.15
11	Kingston upon Thames	0.16
25	Brent	0.16
9	Richmond upon Thames	0.16
19	Barnet	0.17
14	Hillingdon	0.17
16	Ealing	0.17
24	Newham	0.18
27	Bromley	0.18
3	Wandsworth	0.18
12	Hounslow	0.19
21	Lewisham	0.19
17	Greenwich	0.19
26	Waltham Forest	0.19
22	Croydon	0.19
15	Tower Hamlets	0.20
10	Lambeth	0.20
20	Southwark	0.21
13	Hackney	0.22
2	Islington	0.24
8	Camden	0.24
7	Haringey	0.24
5	Hammersmith and Fulham	0.25
6	Kensington and Chelsea	0.30
4	Westminster	0.48

```
rm(temp)
```

The borough which has the lowest crime rate is “Harrow” and “Westminster” has the highest crime rate. The crime rates do not seem to correspond with the boroughs which have the highest and lowest property prices.

A correlation matrix is produced by using “corrplot” package to provide a preliminary analysis of the correlation coefficients between different variables.

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



According to the correlation matrix of “London” data-set, ppsf is positively affected by “crime rate” and “unemployment rate” followed by “area” and “borough”. “Population” has negative correlation with the price while other variables, number of bedroom, bathroom and reception have no effect on ppsf.

## 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: one without the demographic data and

another with the demographic data.

The following algorithms are used in the machine learning models:

A. Linear Regression B. Decision Tree C. Random Forrest

RMSE scores are computed for the models in the two sets of algorithms and they are compared and analysed to deduce the findings for this project.

## 4.1 Data Preparation

In order to verify the models, “London” data-set is split into “test\_set” and “train set”. “train\_set” is used in the machine learning algorithms and “test\_set” is used to verify the models and compute the RMSE scores. “test\_set” is 10% of “london” data-set while “train\_set” is 90%.

```
# Splitting "london" data-set into training set and test set. Test set is 10% of "london" data
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 file
rm(test_index)
```

## 4.2 Baseline Model

After to process to create “train\_set” and “test\_set” is completed , a baseline Model is produced by calculating the average ppsf of “train\_set” and its RMSE score 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 GMP 1102.65 per sq ft and it resulted a RMSE score of 518.38.

## 4.3 Models Without Demographic Data

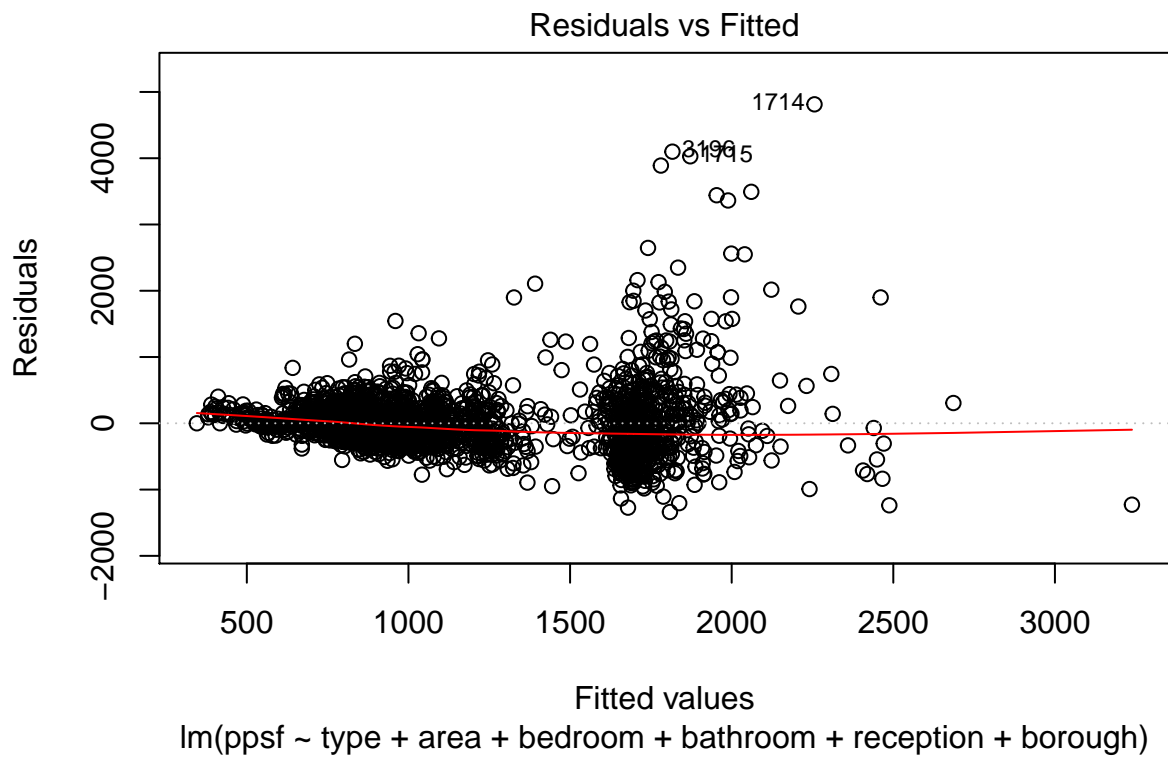
The first set of models, Model 1A, 1B and 1C, are constructed without using the demographic data.

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

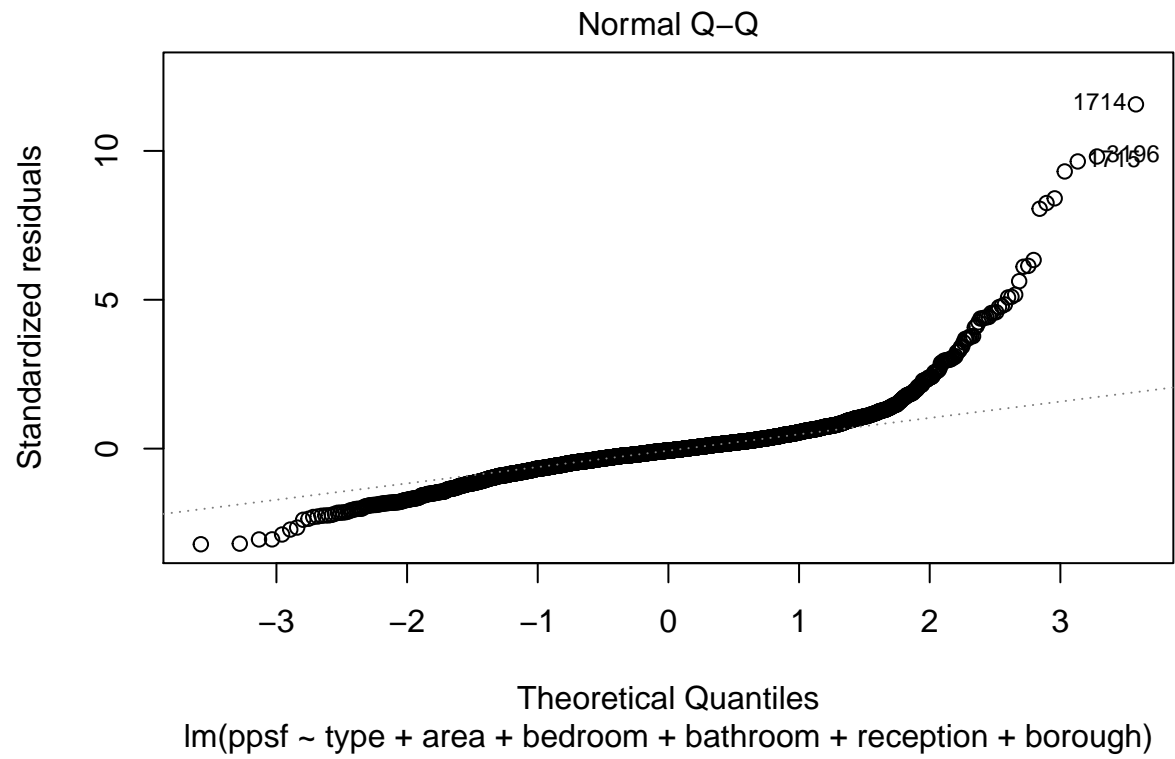
Model 1A is constructed using linear regression as the algorithm.

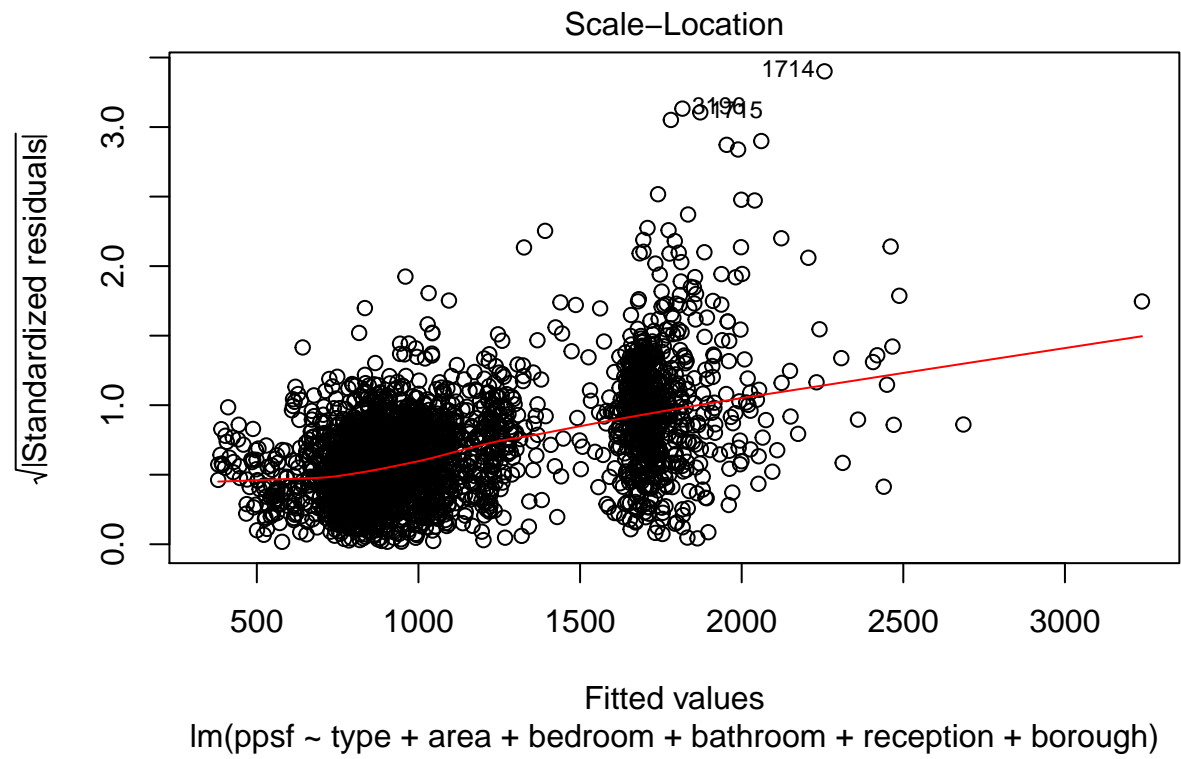
```
#####
# Model 1A Linear Regression Without Demographic Data #
#####

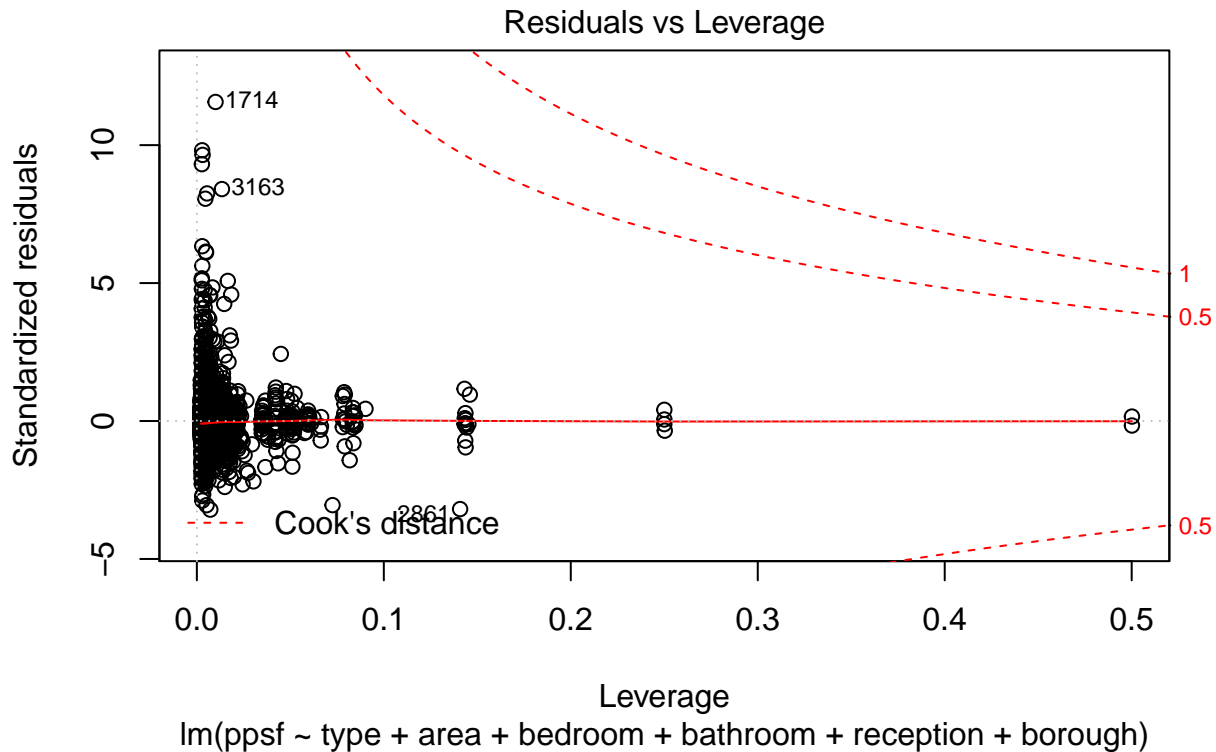
# Linear Regression Computation
lr <- lm(ppsf ~ type+area+bedroom+bathroom+reception+borough, data = train_set)
summary(lr)
plot(lr)
```











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

```
# Compute RMSE for Model 1A
```

```
rmse1a <- RMSE(test_set$ppsf, model1a)
```

The RMSE score from Model 1A is 333.3 which is better than the baseline model as expected.

#### 4.3.2 Model 1B Decision Tree Without Demographic Data

Model 1B uses Decision Tree as the machine learning algorithm and included pruning to try to improve the accuracy by reducing over-fitting.

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

# 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)
```

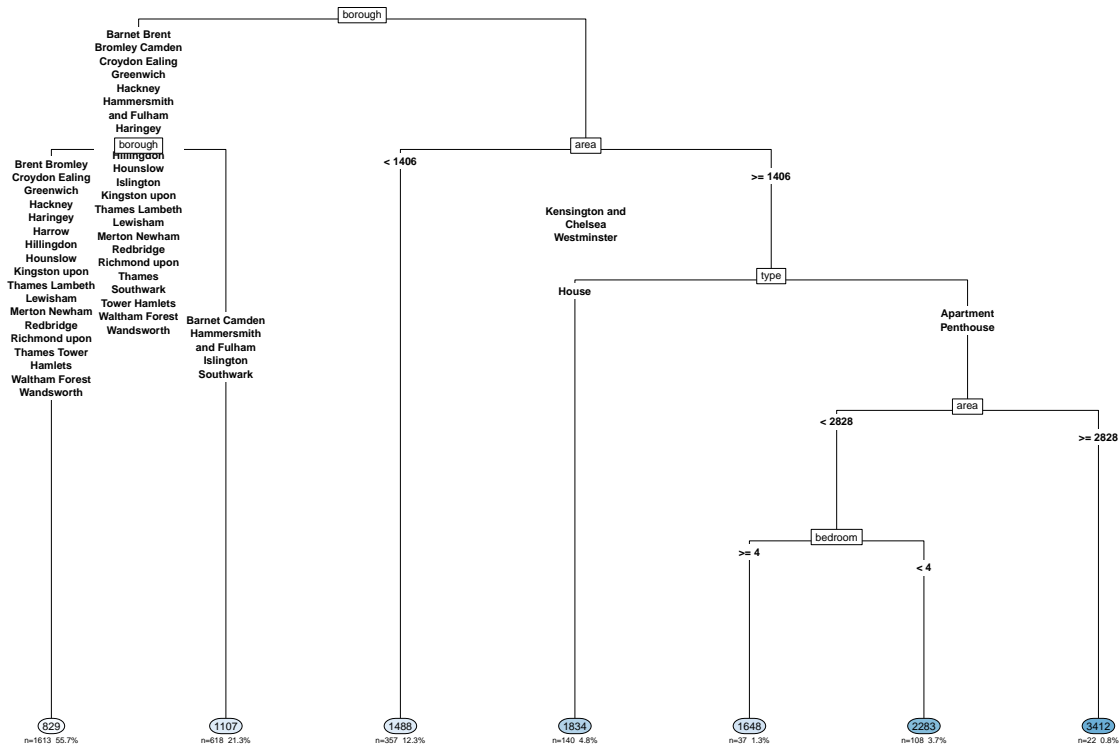
```

{
  # 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,

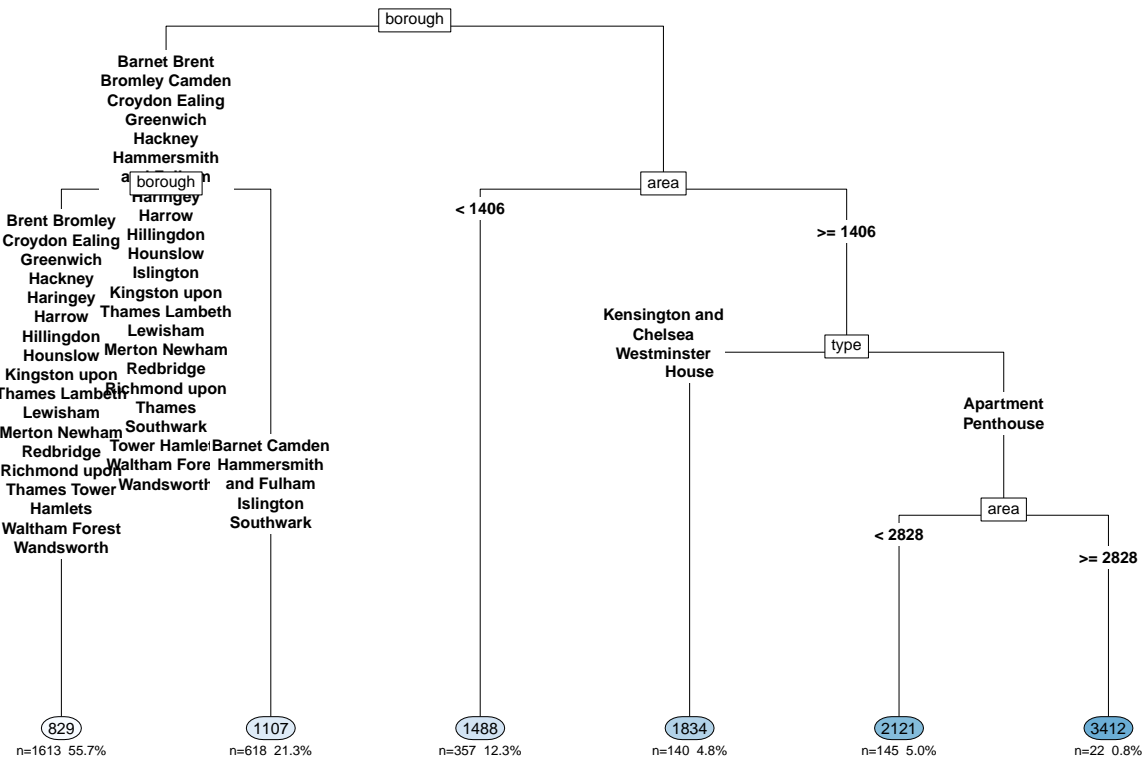
```



```

# Prune the Tree
pfit1b<- prune(train_rpart1b, cp=train_rpart1b$cptable[which.min(train_rpart1b$cptable[, "xerror"],
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)
```

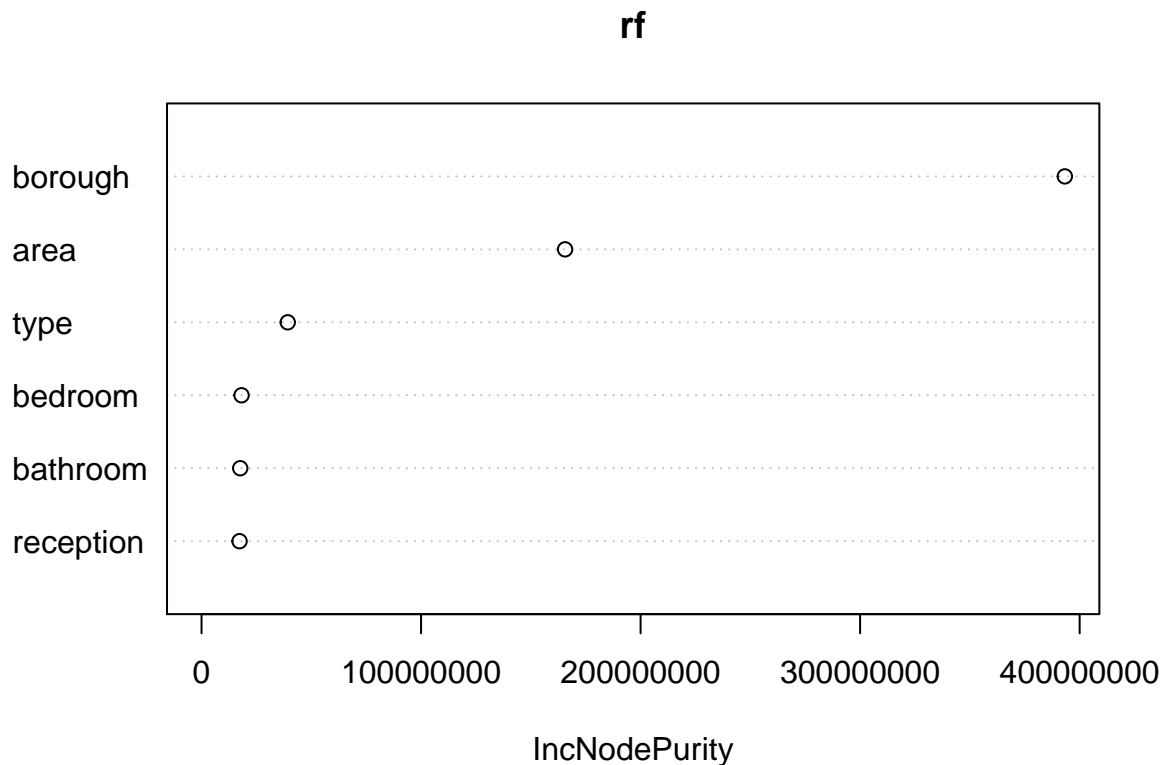
Model 1B achieved a RMSE score of 337.51 and shown Model 1B is not as accurate as Model 1A. The resulted decision tree shown “borough” as the most significant variable followed by “area” and “type”. It is also noted that bedroom is also one of the more significant variable before the pruning.

### 4.3.3 Model 1C Random Forest Without Demographic Data

Model 1C uses Random Forest as the machine learning algorithm.

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

# Random Forest 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)
```

Model 1C resulted a RMSE score of 324.47. It performed better than previous models and it shown “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.

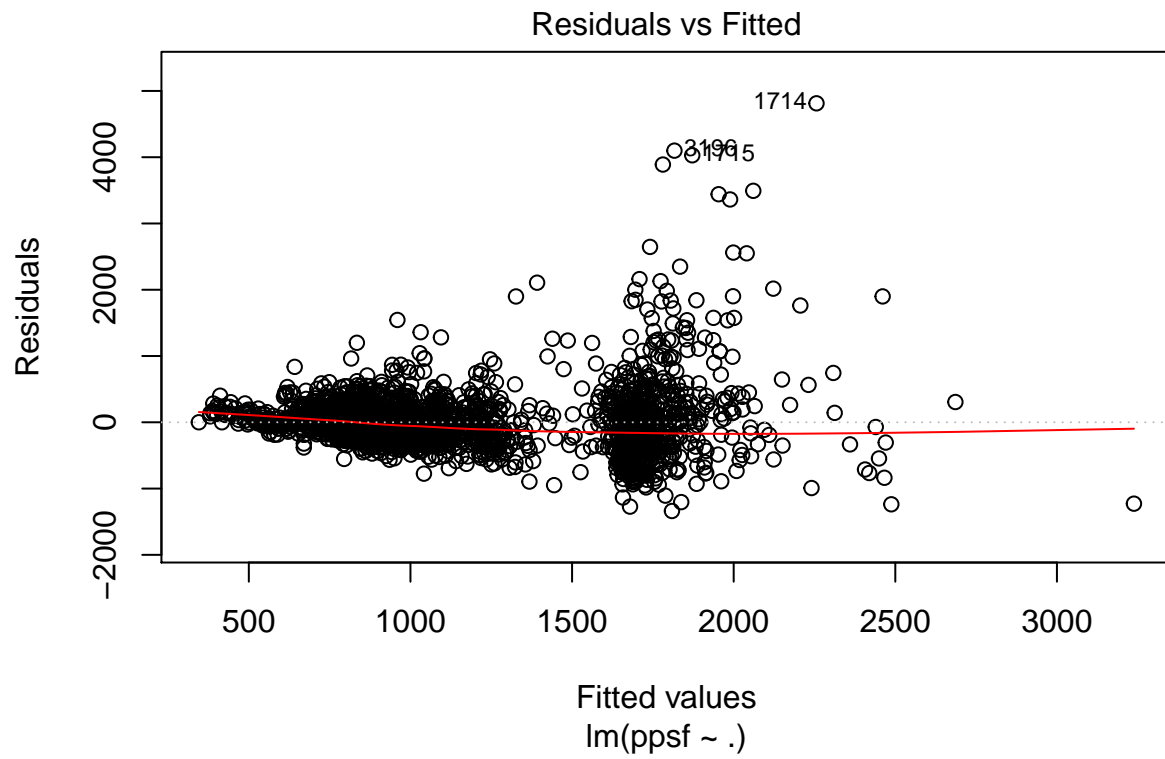
### 4.4.1 Model 2A Linear Regression With Demographic Data

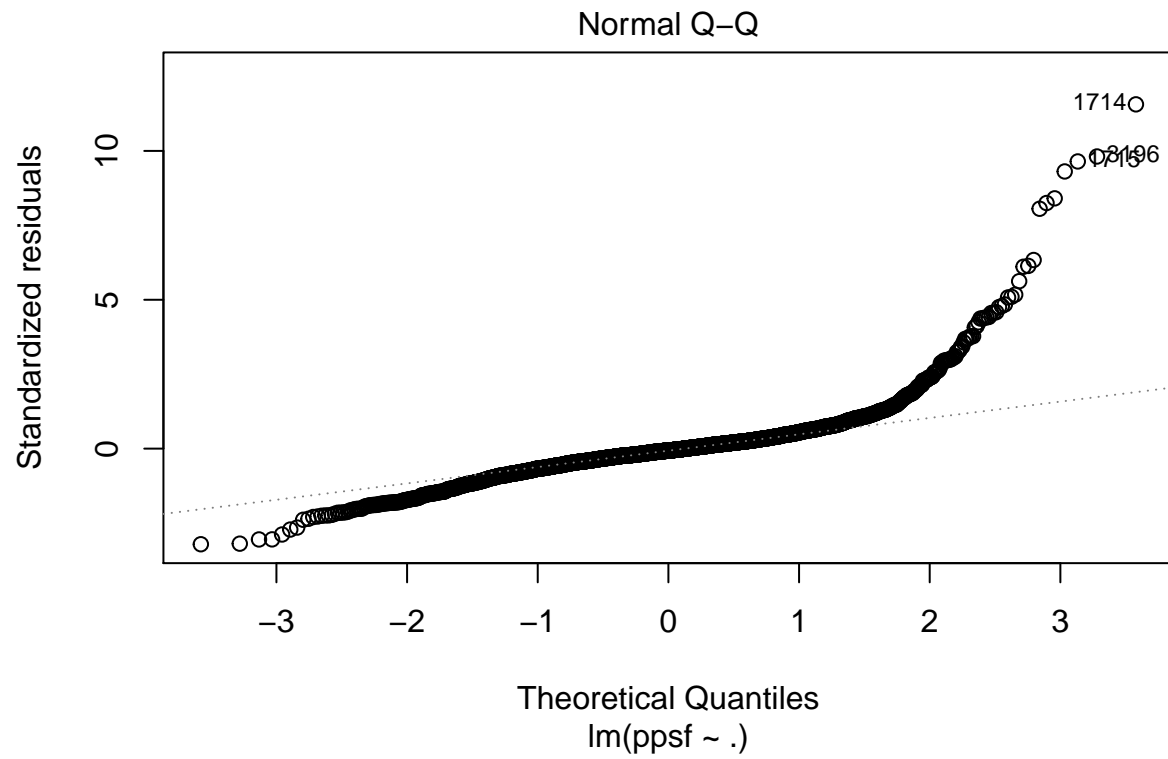
Model 2A, similar to Model 1A, uses linear regression as the machine learning algorithm.

```
#####
# Model 2A - Linear Regression With Demographic Data #
#####

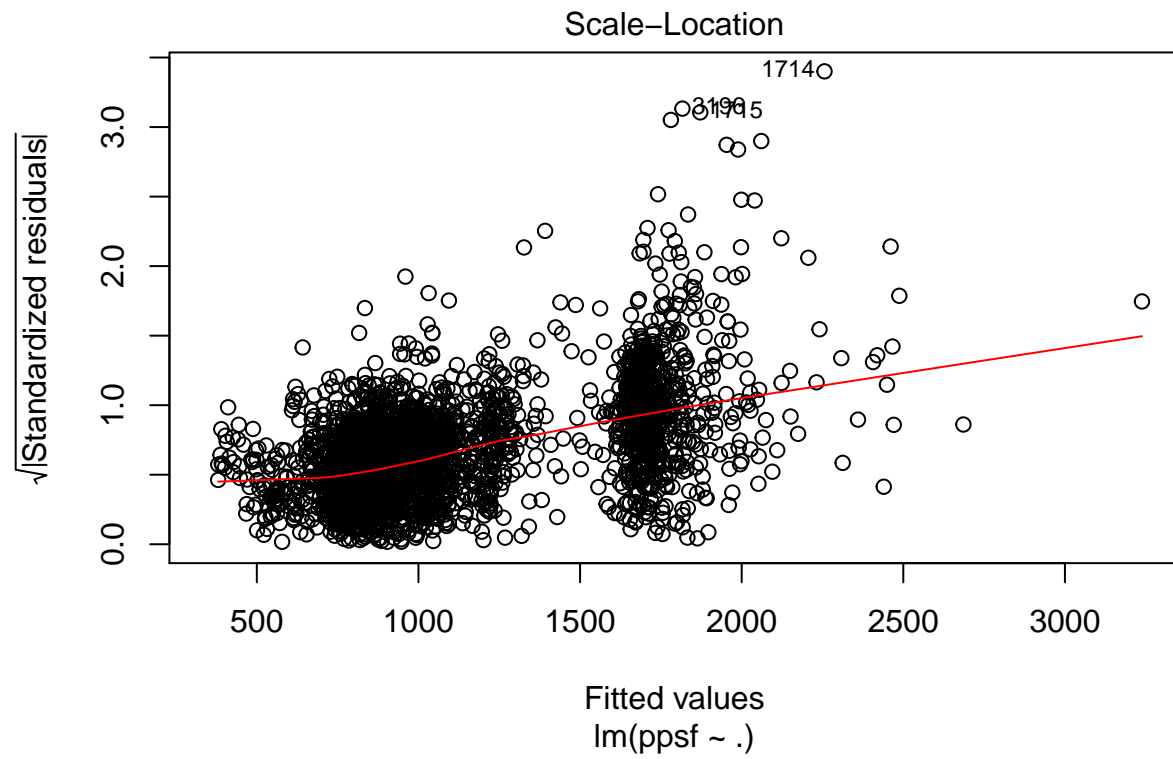
# Linear Regression Computation
lr <- lm(ppsf ~ ., data = train_set)
```

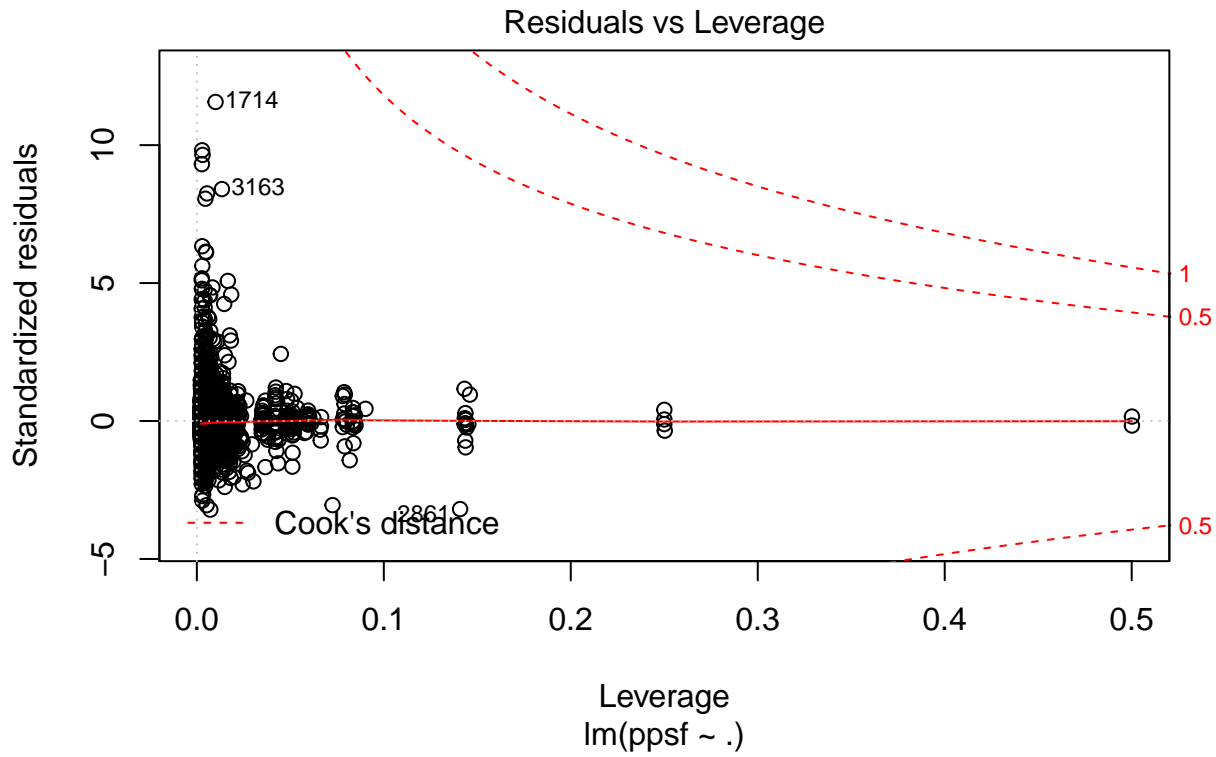
```
summary(lr)  
plot(lr)
```











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

```
# Compute RMSE for Model 2A
```

```
rmse2a<-RMSE(test_set$ppsf, model2a)
```

The RMSE score from Model 2A is 333.3 which is exactly the same as Model 1A. It shown Demographic data had no effect in the computation of linear regression model.

#### 4.4.2 Model 2B Decision Tree With Demographic Data

Model 2B uses Decision Tree with pruning as the machine learning algorithm.

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

```
# Decision Tree Computation
```

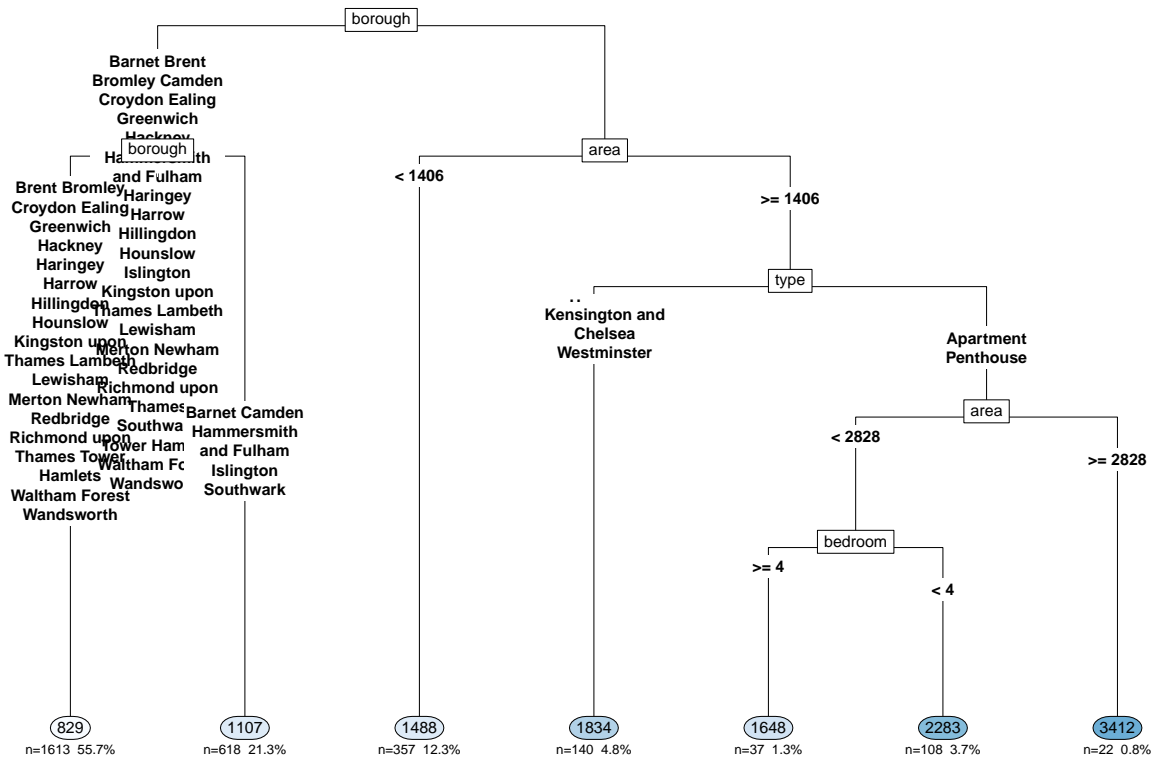
```
train_rpart2b <- rpart(ppsf ~ ., data = train_set)
```

```
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[, "xerror"]), "cp"])
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)
```

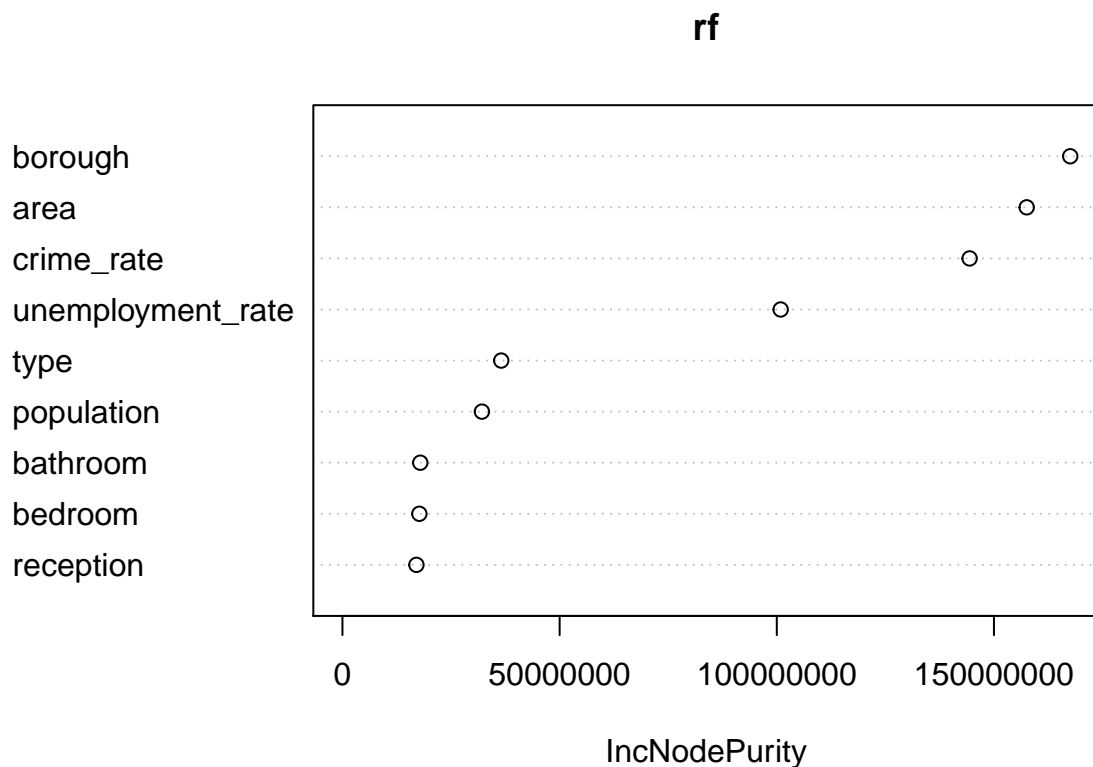
The RMSE score for Model 2B is 331.74 and it performed better than Model 1B, decision tree without demographic data. The significant variables are “borough”, “area”, “type” and “bedroom” which are similar to Model 1B. Although the demographic data is deemed not significant in the algorithm, it still improved the RMSE score nevertheless. It also noted that pruning made no difference in the outcome.

#### 4.4.3 Model 2C Random Forest With Demographic Data

Model 2C uses Random Forest in the machine learning model.

```
#####
# Model 1C Random Forest With Demographic Data Computation #
#####
```

```
# Random Forest Computation
rf<-randomForest(ppsf~., data=train_set)
model2c <- predict(rf, newdata=test_set)
importance <- importance(rf)
varImpPlot(rf)
```



```
# Compute RMSE for Model 2C
rmse2c<-RMSE(test_set$ppsf, model2c)
```

Model 2C achieved a RMSE score of 329.1 which is the best in the models with demographic data. It shown “borough” as the most significant variable in the prediction followed by “area”, “crime rate” and “unemployment rate”. “population” and “type” also had reasonable effect in the algorithm.

## 5 Results

```
#Compile the results into a summary table
summary <- tibble(Model = c("Baseline Model", "Model 1A Linear Regression without Demographic I
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 compiled in the summary table above and the key findings are:

- The most accurate prediction is Model 1C Random Forest Without Demographic Data
- Demographic data made no difference in linear regression model and it is irrelevant in the algorithm
- Demographic data improved the accuracy in Decision Tree model as shown in Model 2B
- Although the demographic data shown as important variables in Model 2C Random Forest with Demographic Data however the accuracy is lower than Model 1C Random Forest Model without Demographic Data
- Borough is the most important variable which confirmed the location factor contributes the most to housing price
- Despite the size factor is taken away by using ppsf, however the results shown the larger properties also have higher ppsf

## 6 Conclusion

From the two sets of machine learning models tested, it is clear that Random Forest algorithm outperformed Linear Regression and Decision Tree in housing price prediction. It is less clearer whether the inclusion of demographic data (population, unemployment rate and crime rate) improves the performance of the algorithms as the results varied and inconclusive.