Ashley Obeng and Maria P. Vargas

December 05, 2022

HOUSE RENT PREDICTION

Introduction

Many people rent instead of buying homes because of individual circumstances and generational trends. The question is how do we determine the renting price of a house? Some millennials are burdened with high student loan debt and face stagnant incomes, making it harder to save a down payment or satisfy the income-to-debt ratio needed to qualify for a mortgage. Others may want the flexibility renting offers and the freedom to move on from a job or city without the burden of having to sell a home. Against this background it will be beneficial to be able to use a machine learning algorithm to predict house rent given certain measurable factors regarded by renters as good indicators for house rent.

Objectives

From the problem statement, our project will seek to come up with a predictive model that can best predict house rent with least amount of error based on the features of a house.

Overview of the Data

In this project, we will consider a house rent prediction data set obtained from the kaggle Repository. This data has 12 variables and 4747 observations.There are four numeric variables with the rest being character(categorical) variables. The ‘rent price’ variable will be used as the target or response variable while the remaining variables will be considered as potential predictors.

Variable Description

- BHK: Number of Bedrooms, Hall, Kitchen.
- Rent: Rent of the Houses/Apartments/Flats.
- Size: Size of the Houses/Apartments/Flats in Square Feet.
- Floor: Houses/Apartments/Flats situated in which Floor and Total Number of Floors (Example: Ground out of 2, 3 out of 5, etc.)
- Area Type: Size of the Houses/Apartments/Flats calculated on either Super Area or Carpet Area or Build Area.
- Area Locality: Locality of the Houses/Apartments/Flats.
- City: City where the Houses/Apartments/Flats are Located.
- Furnishing Status: Furnishing Status of the Houses/Apartments/Flats, either it is Furnished or Semi-Furnished or Unfurnished.
- Tenant Preferred: Type of Tenant Preferred by the Owner or Agent.
- Bathroom: Number of Bathrooms.

- Point of Contact: Whom should you contact for more information regarding the Houses/Apartments/Flats.

Exploratory Data Analysis

Before preceeding to fit the models, it is important to gain some intial insight about the data. To this end,we look at the distribution of out target variable(rent) and the predictor variables.

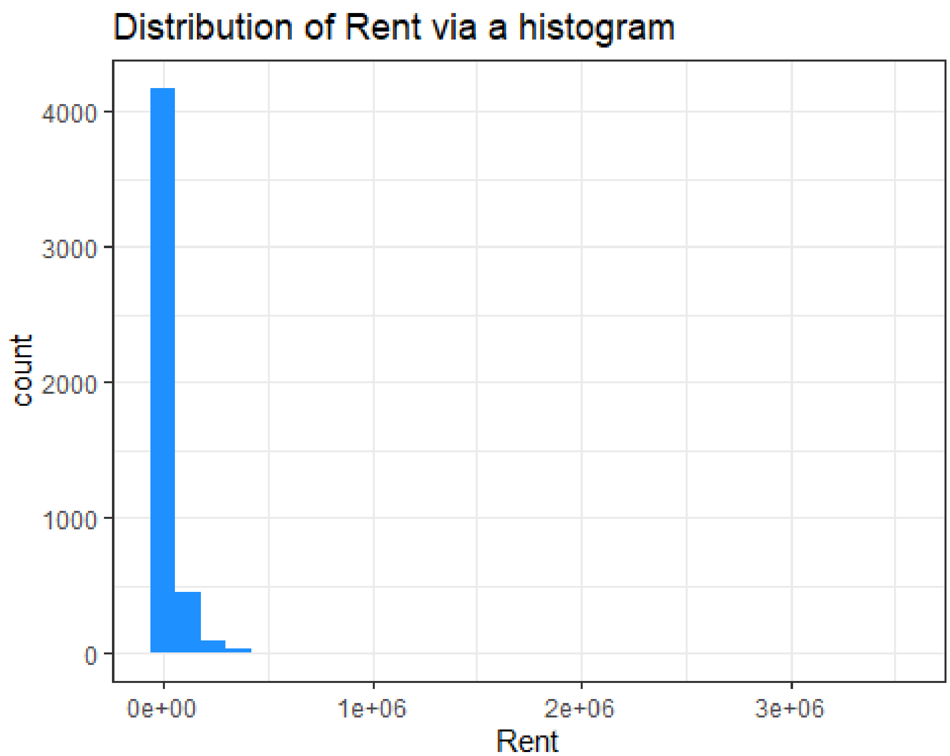
```
##   Posted.On              BHK             Rent              Size       
##  Length:4746        Min.   :1.000   Min.   :   1200   Min.   :  10.0  
##  Class :character   1st Qu.:2.000   1st Qu.:  10000   1st Qu.: 550.0  
##  Mode  :character   Median :2.000   Median :  16000   Median : 850.0  
##                     Mean   :2.084   Mean   :  34993   Mean   : 967.5  
##                     3rd Qu.:3.000   3rd Qu.:  33000   3rd Qu.:1200.0  
##                     Max.   :6.000   Max.   :3500000   Max.   :8000.0  
##      Floor          Totalfloors      Area.Type         Area.Locality     
##  Min.   : 0.000   Min.   : 1.000   Length:4746        Length:4746       
##  1st Qu.: 1.000   1st Qu.: 2.000   Class :character   Class :character  
##  Median : 2.000   Median : 4.000   Mode  :character   Mode  :character  
##  Mean   : 3.641   Mean   : 6.969                                        
##  3rd Qu.: 3.000   3rd Qu.: 6.000                                        
##  Max.   :76.000   Max.   :89.000                                        
##      City           Furnishing.Status  Tenant.Preferred      Bathroom     
##  Length:4746        Length:4746        Length:4746        Min.   : 1.000  
##  Class :character   Class :character   Class :character   1st Qu.: 1.000  
##  Mode  :character   Mode  :character   Mode  :character   Median : 2.000  
##                                                            Mean   : 1.966  
##                                                            3rd Qu.: 2.000  
##                                                            Max.   :10.000  
##  Point.of.Contact  
##  Length:4746       
##  Class :character  
##  Mode  :character  
##                    
##                    
##                    

##   Posted.On              BHK             Rent              Size       
##  Length:4746        Min.   :1.000   Min.   :   1200   Min.   :  10.0  
##  Class :character   1st Qu.:2.000   1st Qu.:  10000   1st Qu.: 550.0  
##  Mode  :character   Median :2.000   Median :  16000   Median : 850.0  
##                     Mean   :2.084   Mean   :  34993   Mean   : 967.5  
##                     3rd Qu.:3.000   3rd Qu.:  33000   3rd Qu.:1200.0  
##                     Max.   :6.000   Max.   :3500000   Max.   :8000.0  
##      Floor          Totalfloors      Area.Type         Area.Locality     
##  Min.   : 0.000   Min.   : 1.000   Length:4746        Length:4746       
##  1st Qu.: 1.000   1st Qu.: 2.000   Class :character   Class :character  
##  Median : 2.000   Median : 4.000   Mode  :character   Mode  :character  
##  Mean   : 3.641   Mean   : 6.969                                        
##  3rd Qu.: 3.000   3rd Qu.: 6.000                                        
```

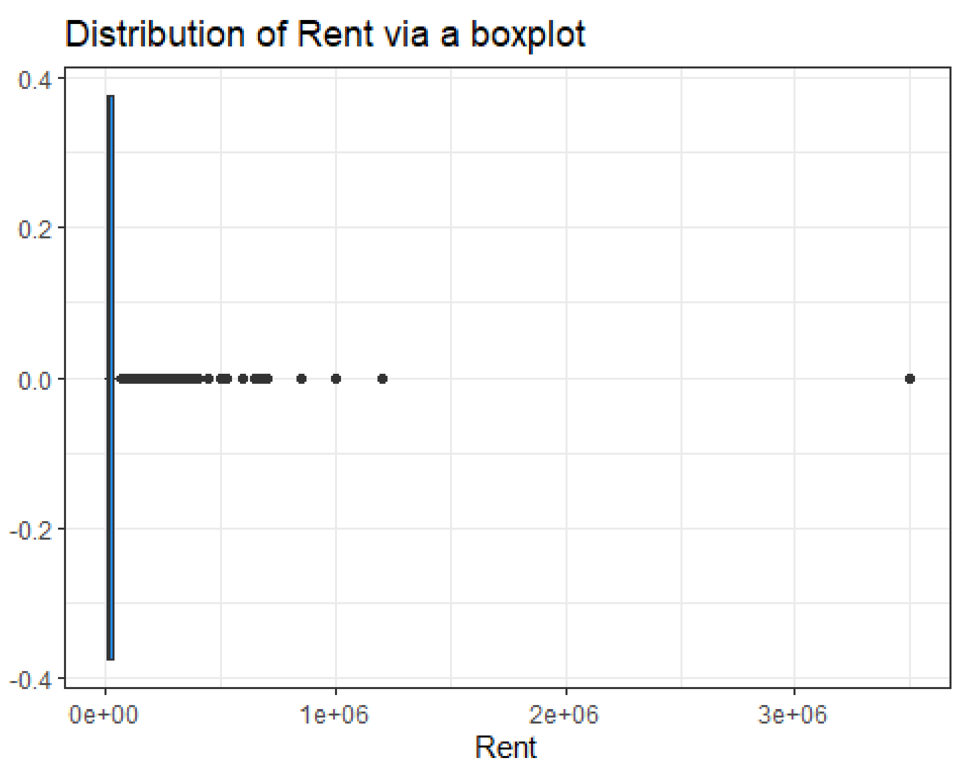
```
##  Max.   :76.000   Max.   :89.000
##     City           Furnishing.Status  Tenant.Preferred      Bathroom
##  Length:4746       Length:4746        Length:4746        Min.   : 1.000
##  Class :character   Class :character   Class :character   1st Qu.: 1.000
##  Mode  :character   Mode  :character   Mode  :character   Median : 2.000
##                                                          Mean   : 1.966
##                                                          3rd Qu.: 2.000
##                                                          Max.   :10.000
##  Point.of.Contact
##  Length:4746
##  Class :character
##  Mode  :character
##
##
##
```

```
##         Posted.On               BHK              Rent              Size
##                 0                 0                 0                 0
##             Floor       Totalfloors         Area.Type     Area.Locality
##                 0                 0                 0                 0
##              City Furnishing.Status  Tenant.Preferred          Bathroom
##                 0                 0                 0                 0
##  Point.of.Contact
##                 0
```

From the above result, there was one missing value for Totalfloors.

Distribution of Target variable (Rent)

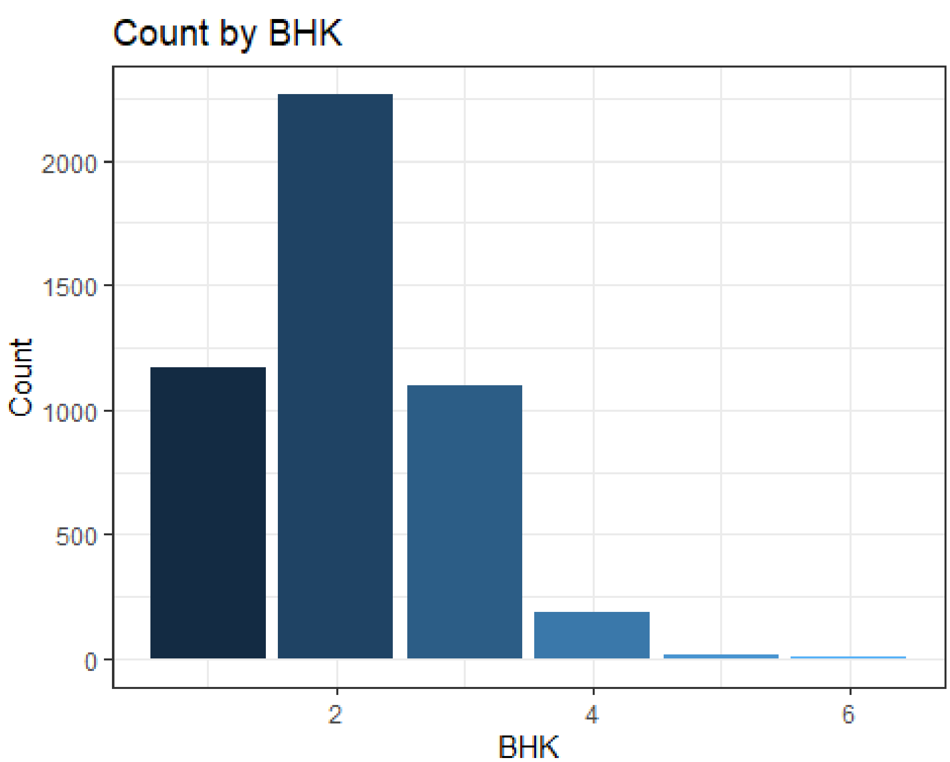


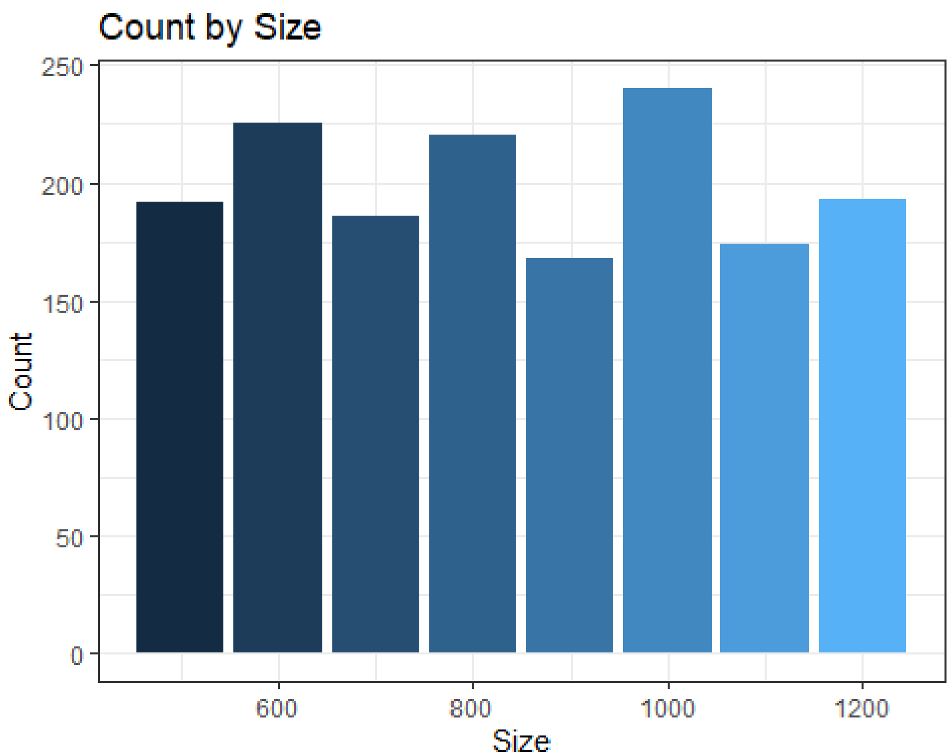


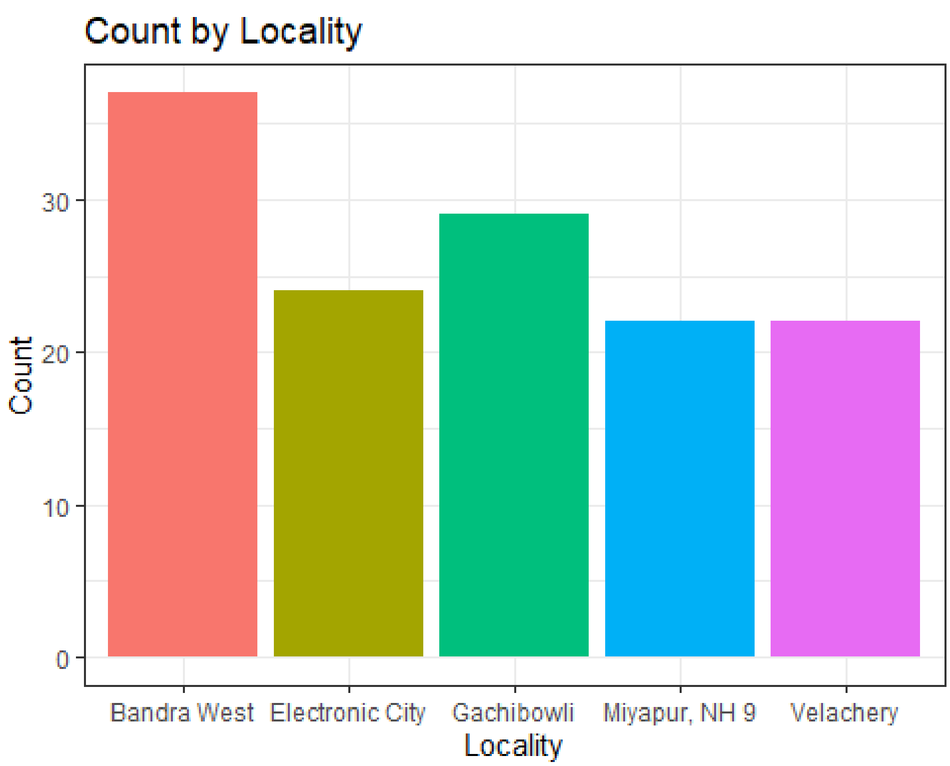
The histogram and boxplot distribution of the rent data reveal significantly that the rent variable is very skewed to the right. Thus,the rent data set is not normally distributed. In addition, it can be observed that there are a few outlying observations in the data set.

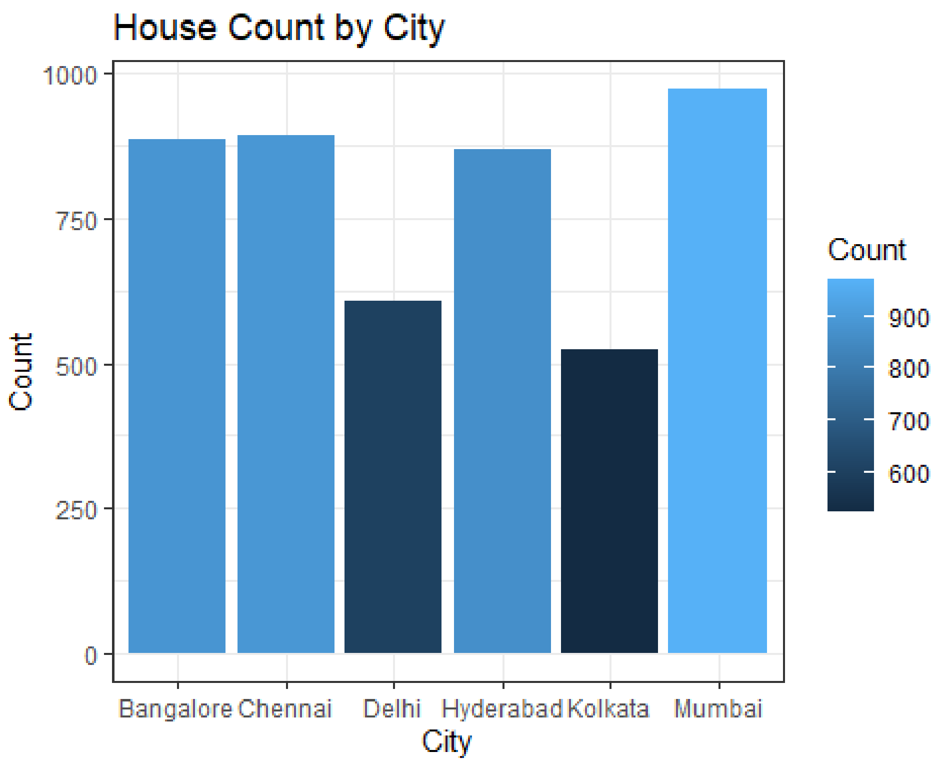
Univariate Distributions of predictors

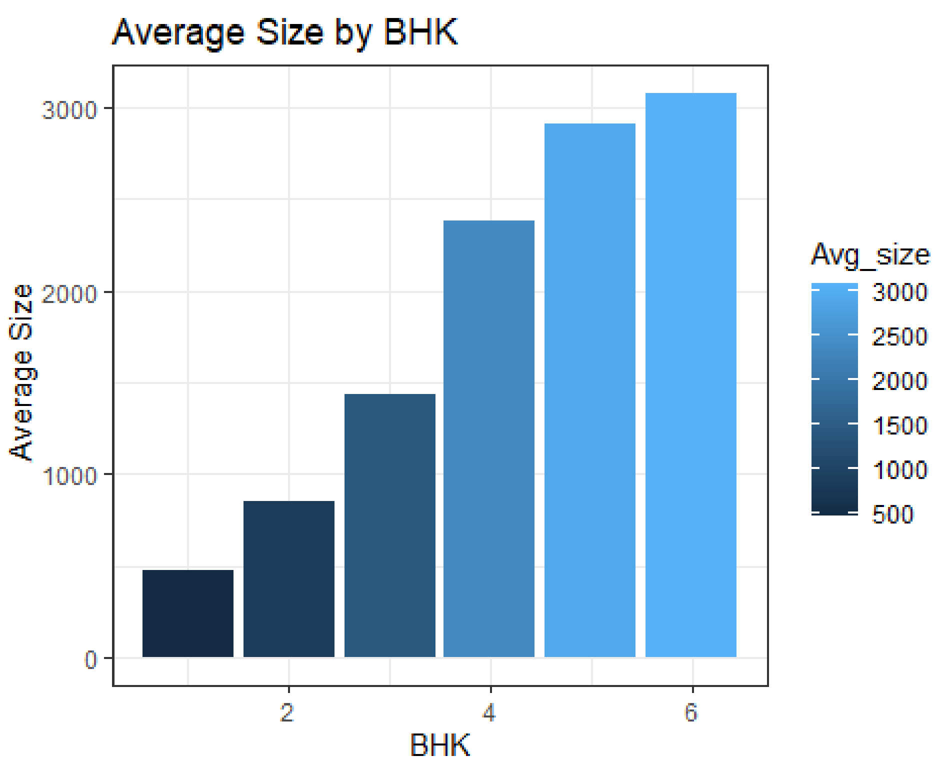
In this section we explore the distribution of individual predictor variables.

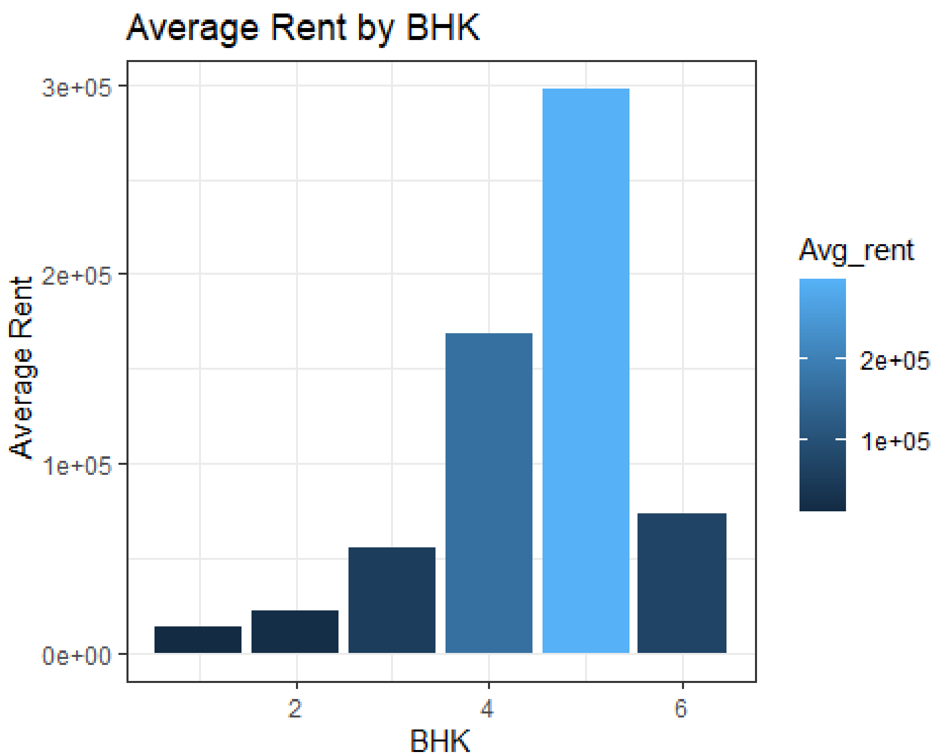


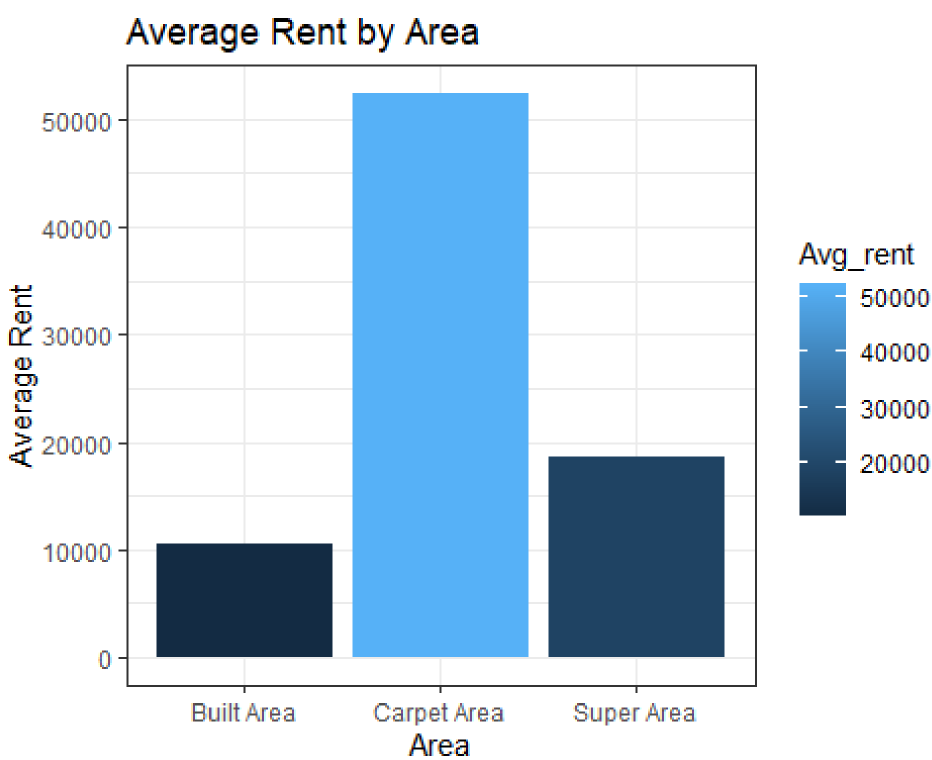

Count by Size
250
200
150
100
50
0
Count
600
800
1000
1200
Size

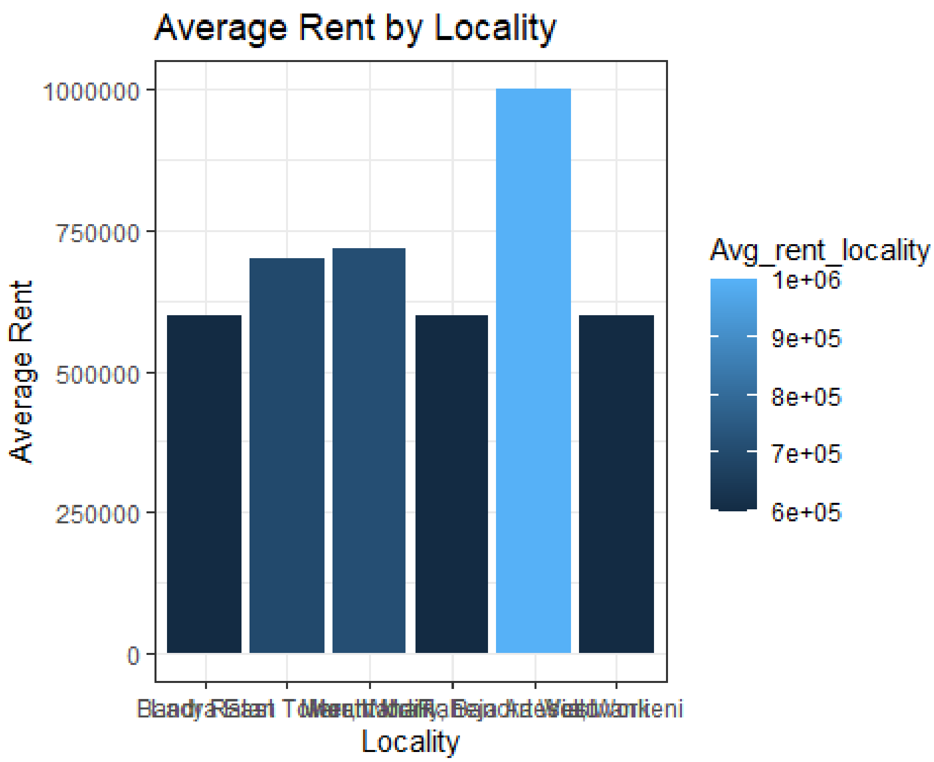

Count by Locality
30
20
10
0
Count
Bandra West
Electronic City
Gachibowli
Miyapur, NH 9
Velachery
Locality

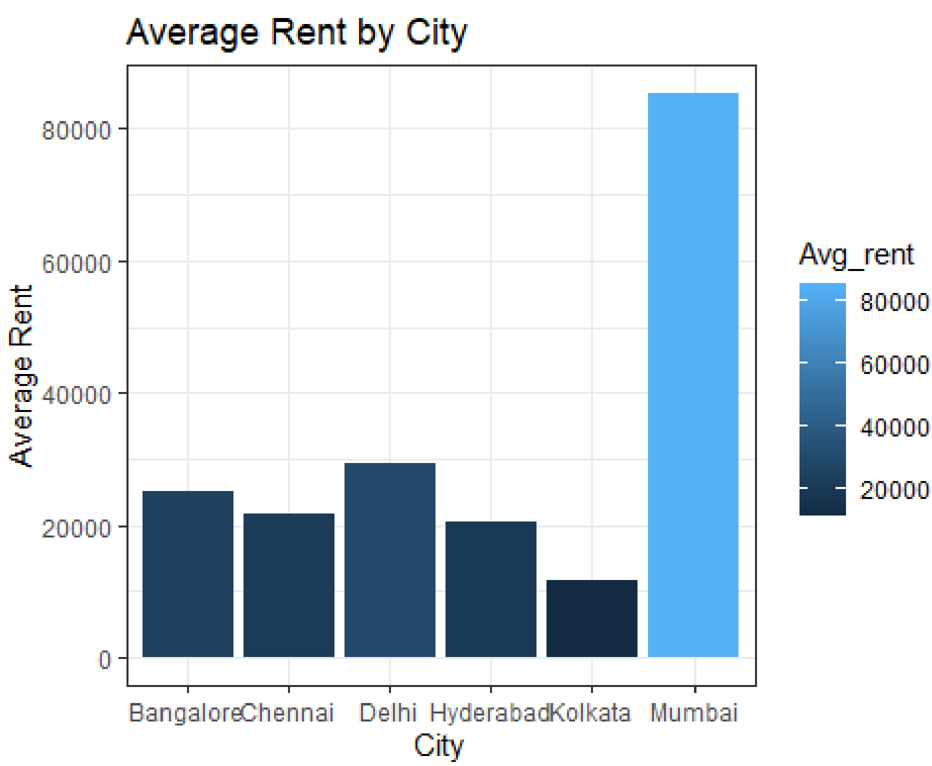

House Count by City
1000
750
500
250
0
Count
Bangalore
Chennai
Delhi
Hyderabad
Kolkata
Mumbai
City
Count
900
800
700
600

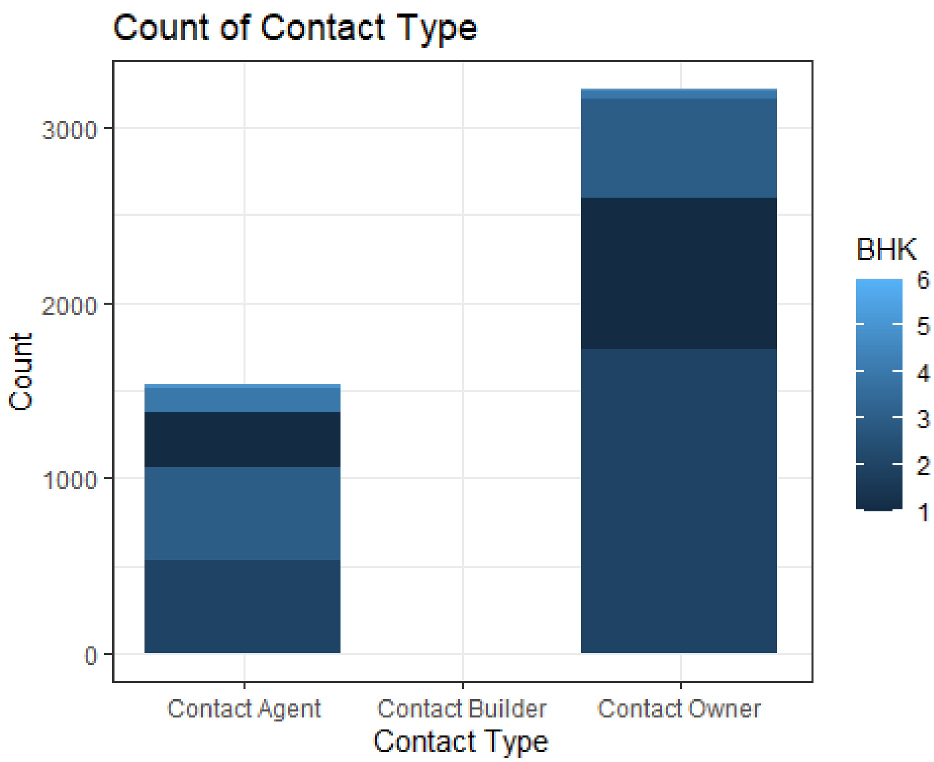

Average Size by BHK
3000
2000
1000
0
Average Size
2
4
6
BHK
Avg_size
3000
2500
2000
1500
1000
500

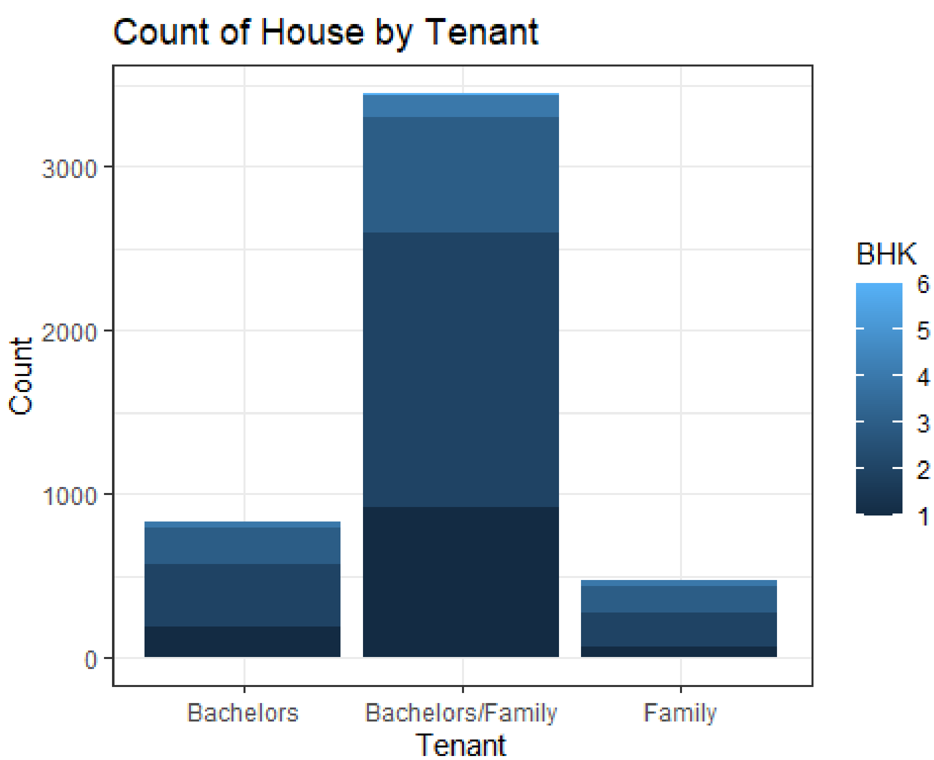

Average Rent by BHK
Average Rent
3e+05
2e+05
1e+05
0e+00
2
4
6
BHK
Avg_rent
2e+05
1e+05


Average Rent by Area
Average Rent
50000
40000
30000
20000
10000
0
Built Area
Carpet Area
Super Area
Area
Avg_rent
50000
40000
30000
20000


Average Rent by Locality
1000000
750000
500000
250000
0
Average Rent
Locality
Avg_rent_locality
1e+06
9e+05
8e+05
7e+05
6e+05


Average Rent by City
80000
60000
40000
20000
0
Average Rent
Bangalore
Chennai
Delhi
Hyderabad
Kolkata
Mumbai
City
Avg_rent
80000
60000
40000
20000



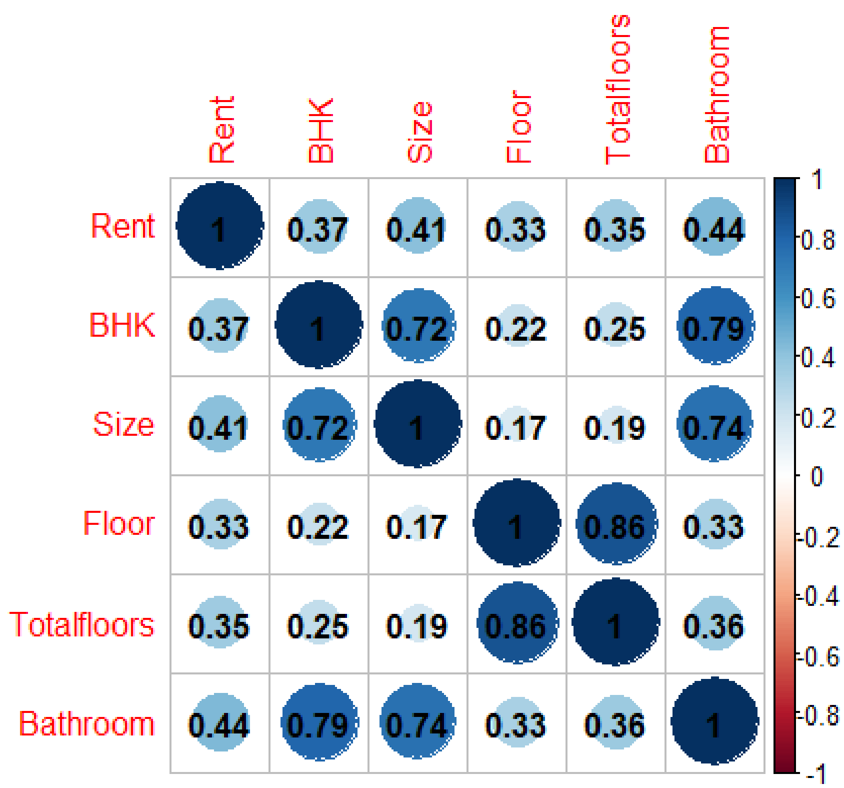


Furnishing.Status	Rent
Furnished	850000
Semi-Furnished	3500000

Furnishing.Status	Rent
Unfurnished	600000

Furnishing.Status	Rent
Furnished	38155008
Semi-Furnished	87156043
Unfurnished	40767869

Correlation among variables



Generally, there appears to be a weak association between rent and each of the continuous predictors, BHK, Size, Floor and Bathroom. Looking at the correlation among the predictors, we observe a high correlation between BHK and Size, BHK and Bathroom, and Bathroom and Size. This may lead to the issue of multicollinearity in our predictive modelling. However, the correlation between BHK and Bathroom is the highest since bathroom is a direct component of BHK(Bathroom, Hall and Kitchen) and hence it may be appropriate to exclude Bathroom from the set of predictors.

Predictive modelling

Data partitioning

For the purpose of model validation, we partitioned our data into 60% training data and 40% test data. The “built area” and “contact builder” levels of the Area.Type and Point.of.Contact, respectively, were removed because they just had 1 and 2 observations to avoid the issue of class imbalance.

```
## [1] 2845   10

## [1] 1898   10
```

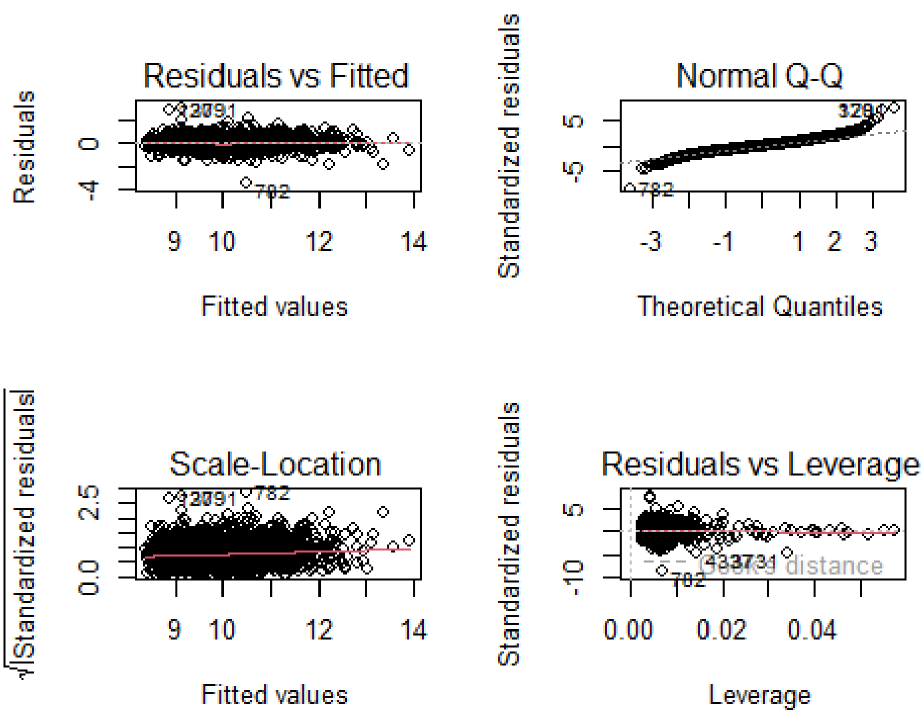
Multiple Linear Regression

we fitted the Multiple Linear Regression using the log transformation of the target variable(Rent) since it was heavily skewed to the right.

```
##
## Call:
## lm(formula = log(Rent) ~ ., data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4193 -0.2327 -0.0072  0.2236  2.9850
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       9.075e+00  4.210e-02 215.561  < 2e-16
***
## BHK                               3.139e-01  1.396e-02  22.482  < 2e-16
***
## Size                              4.906e-04  1.955e-05  25.089  < 2e-16
***
## Floor                             4.529e-03  2.622e-03   1.727   0.0842 .
## Totalfloors                       4.325e-03  1.710e-03   2.529   0.0115 *
## Area.TypeSuper Area              -4.089e-02  1.958e-02  -2.088   0.0369 *
## CityChennai                      -2.395e-02  2.506e-02  -0.956   0.3392
## CityDelhi                         2.037e-01  2.947e-02   6.913 5.84e-12
***
## CityHyderabad                    -1.409e-01  2.566e-02  -5.492 4.33e-08
***
## CityKolkata                      -3.383e-01  2.983e-02 -11.339  < 2e-16
***
## CityMumbai                        9.456e-01  3.117e-02  30.334  < 2e-16
***
## Furnishing.StatusSemi-Furnished  -2.001e-01  2.384e-02  -8.396  < 2e-16
***
## Furnishing.StatusUnfurnished     -3.096e-01  2.474e-02 -12.511  < 2e-16
***
## Tenant.PreferredBachelors/Family -4.589e-02  2.224e-02  -2.064   0.0391 *
```

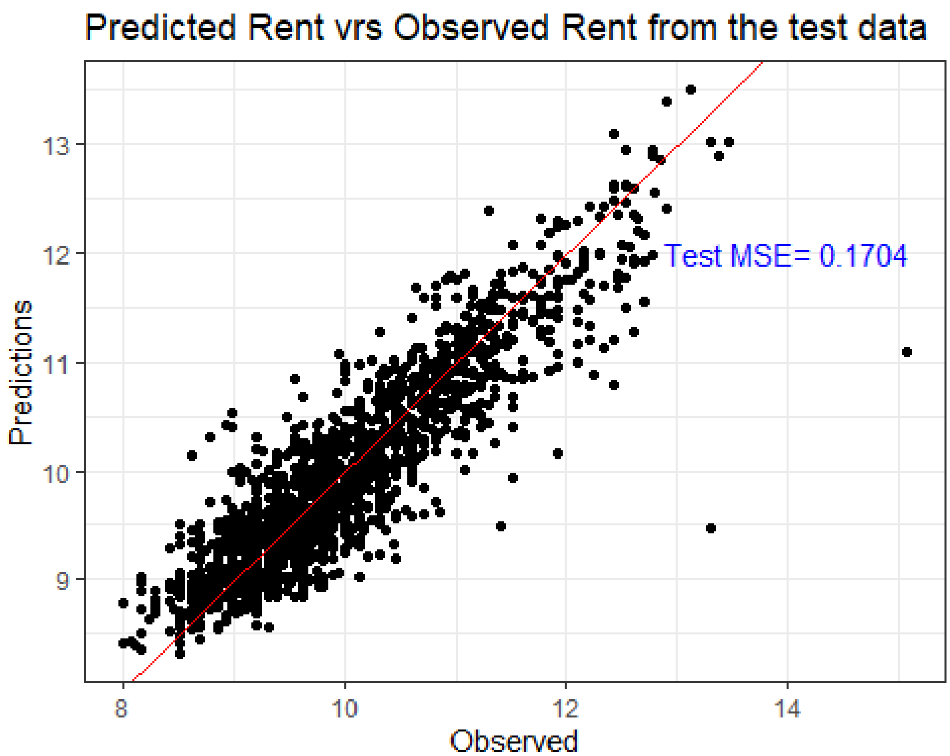
```
## Tenant.PreferredFamily           -1.233e-01  3.123e-02  -3.949 8.02e-05
***
## Point.of.ContactContact Owner    -3.687e-01  2.334e-02 -15.800  < 2e-16
***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4068 on 2829 degrees of freedom
## Multiple R-squared:  0.8098, Adjusted R-squared:  0.8088
## F-statistic: 802.9 on 15 and 2829 DF,  p-value: < 2.2e-16
```

From the summary output of the fitted model,it can be observed that the Pvalues of almost all the predictor variables are significant and also based on the value of the R-square adjusted more than 80% of the variation in the response is jointly accounted for by the predictor variables.



From the the Normal Q-Q plot we can see that the error terms are normally distributed and also there is a linear relationship between rent and the predictor variables. From the residual vrs fitted plot, the error terms have constant variance.

```
## [1] 0.1703918
```



Though we don’t have a perfect prediction, it is clear that the model did reasonably well on the test data as most of the points are near the red line. This is an indication that the model will have a good predictive power even on unseen data.

Regression Tree

- Estimate the parameters:

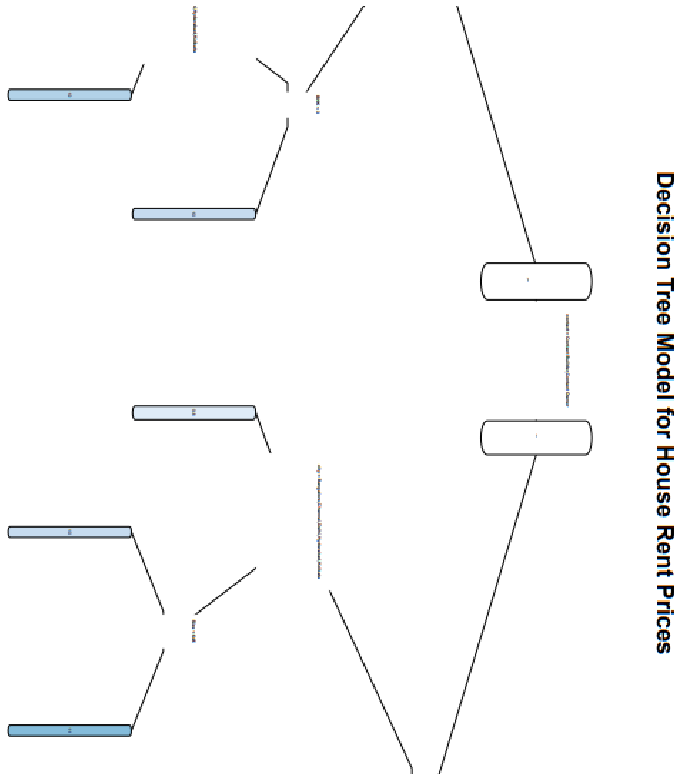
$$\beta_{<} = \frac{\sum_i \mathbb{I}(y_i = 1) \cdot \mathbb{I}(x_i \in I_{<})}{\sum_i \mathbb{I}(x_i \in I_{<})} \text{ and } \beta_{>} = \frac{\sum_i \mathbb{I}(y_i = 1) \cdot \mathbb{I}(x_i \in I_{>})}{\sum_i \mathbb{I}(x_i \in I_{>})}$$

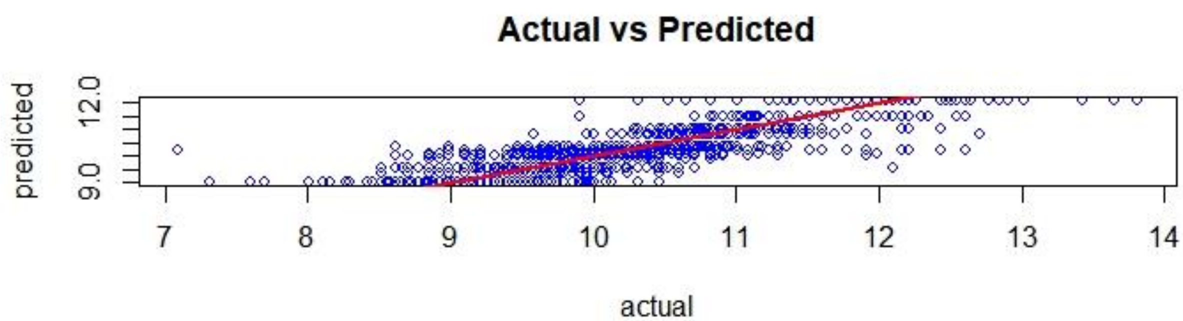
Regression tree is a non-parametric predictive method. It creates a binary tree by recursively splitting the data on the predictor values. The splits are selected so that the two child nodes have smaller variability around their average value than the parent node. To fit our regression tree, we used the rpart package that uses the Gini index as its class purity metric.

Summary of initial tree without pruning:

```
Call:
rpart(formula = lrent ~ ., data = dat.training)
  n= 3164

           CP nsplit rel error    xerror       xstd
1  0.39356957      0 1.0000000 1.0006292 0.03014597
2  0.10466808      1 0.6064304 0.6073482 0.02062739
3  0.07348349      2 0.5017624 0.5028539 0.01658539
4  0.04810246      3 0.4282789 0.4293648 0.01493677
5  0.02851707      4 0.3801764 0.3812212 0.01433126
6  0.02645705      5 0.3516593 0.3644237 0.01425995
7  0.01707198      6 0.3252023 0.3336795 0.01349208
8  0.01451244      7 0.3081303 0.3197633 0.01328464
9  0.01363367      8 0.2936179 0.3072745 0.01344116
10 0.01231613      9 0.2799842 0.2937381 0.01320393
11 0.01229751     10 0.2676681 0.2753901 0.01281540
12 0.01000000     11 0.2553705 0.2639749 0.01254396

Variable importance
    contact    Bathroom        city Totalfloors         BHK        Size
         21          17          14          12          10           9
      Floor   area_type
          9           7

Node number 1: 3164 observations,    complexity param=0.3935696
```

We used cross-validation to prune our tree. And the best size of the tree is 11, similar to the original one but the complexity parameter is 0.0123.





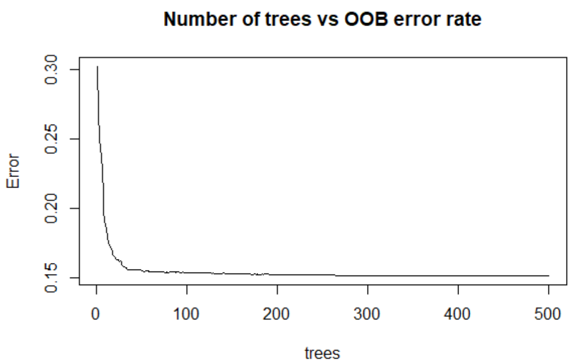
The MSE for the tree is 0.2355.

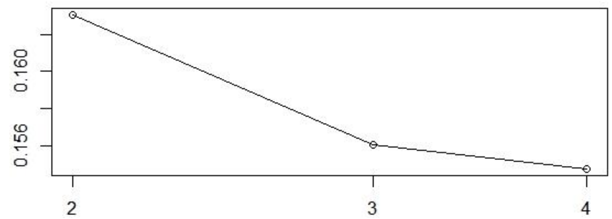
Random Forest

$$\hat{y} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} W_j(x_i, x')\, y_i = \sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m} W_j(x_i, x')\right)\, y_i .$$

The difference between a single regression tree and a random forest regression tree is that the random forest is an ensemble of decision trees. And it only uses a restricted number of features at each node and it uses different sample of features at each split.

We started with a tree using all the features and use the out of bag error vs number of trees plot to determine the ideal number of trees to be used. Then we tune our model to find the best m, or the best number of features that should be use in each split. It came out to be 4, as shown in the plot.

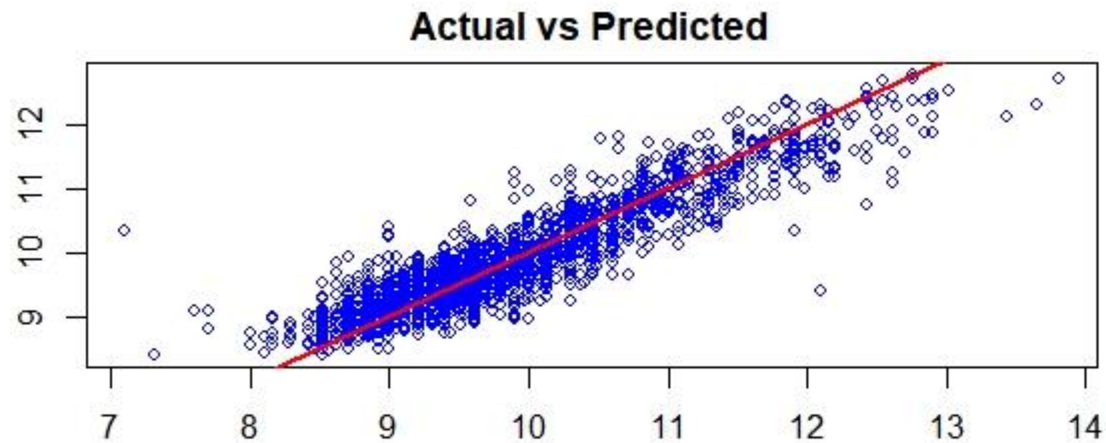




Using the ideal number of parameters, we fit out random forest model again.

```
Call:
 randomForest(formula = lrent ~ ., data = dat.training, mtry = best.
m,      ntree = 100, keep.forest = TRUE, importance = TRUE, proximity
 = TRUE,      oob.prox = FALSE)
               Type of random forest: regression
                     Number of trees: 100
No. of variables tried at each split: 4

          Mean of squared residuals: 0.1550959
                    % Var explained: 81.55
```

```
              %IncMSE IncNodePurity
month         2.456337      42.65800
BHK          14.474514     187.83707
Size         24.641692     367.46355
Floor        10.692078     102.68117
Totalfloors 13.952119     283.55050
Bathroom     17.603951     455.35343
area_type    10.998424      27.66971
city         34.279430     355.13720
furn         14.062845      54.33069
ten_pref      5.116203      28.86916
contact      22.855922     392.94895
```

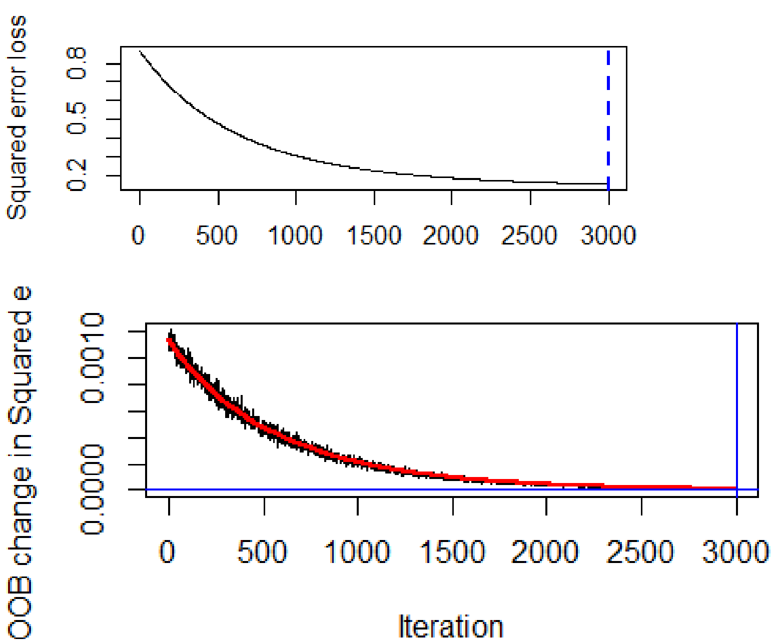


The MSE for the random forest model is 0.147, smaller than the linear regression and the regression tree.

Boosting

Initialize $\hat{f}(\mathbf{x})$ to be a constant, $\hat{f}(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^N \Psi(y_i, \rho)$
For t in $1, \ldots, T$ do

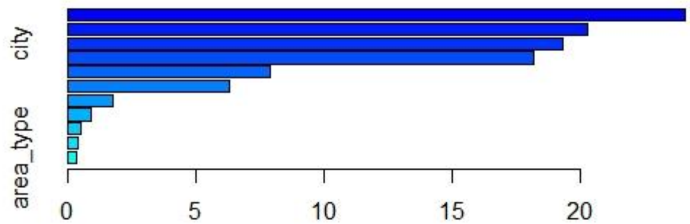
1. Compute the negative gradient as the working response
$$z_i = -\frac{\partial}{\partial f(\mathbf{x}_i)} \Psi(y_i, f(\mathbf{x}_i)) \Big|_{f(\mathbf{x}_i)=\hat{f}(\mathbf{x}_i)} \tag{13}$$
2. Randomly select $p \times N$ cases from the dataset
3. Fit a regression tree with K terminal nodes, $g(\mathbf{x}) = \mathrm{E}(z|\mathbf{x})$. This tree is fit using only those randomly selected observations
4. Compute the optimal terminal node predictions, $\rho_1, \ldots, \rho_K$, as
$$\rho_k = \arg\min_\rho \sum_{\mathbf{x}_i \in S_k} \Psi(y_i, \hat{f}(\mathbf{x}_i) + \rho) \tag{14}$$
where S_k is the set of xs that define terminal node k. Again this step uses only the randomly selected observations.
5. Update $\hat{f}(\mathbf{x})$ as
$$\hat{f}(\mathbf{x}) \leftarrow \hat{f}(\mathbf{x}) + \lambda \rho_{k(\mathbf{x})} \tag{15}$$



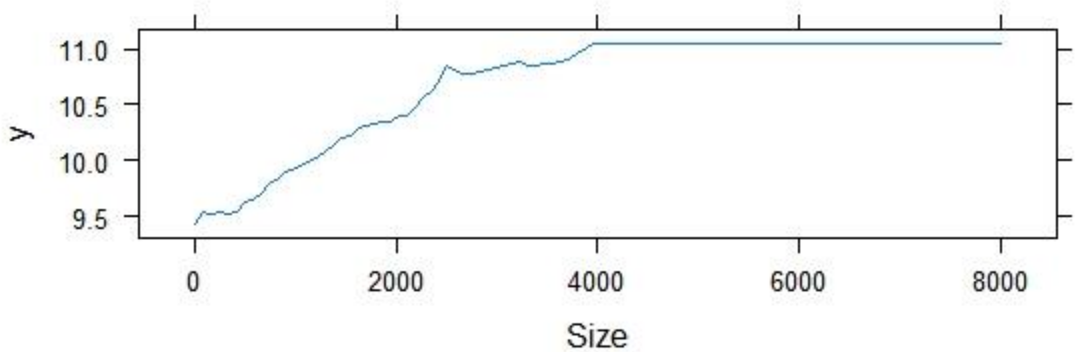
According to the plots there is not much difference in the out of bag error and in the square error loss after the 3000 iterations. We started with 10000 iterations and a shrinkage parameter of 0.001 to minimize the error. We used cross validation to find the best parameters.

The variables of importance in order are:

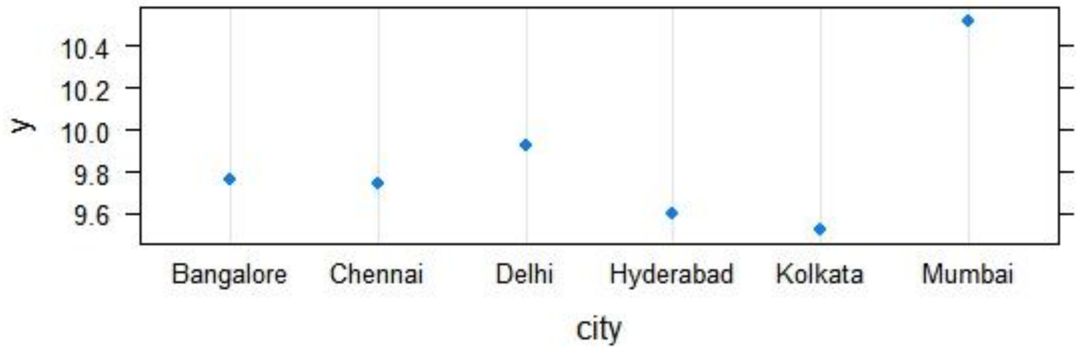
```
                    var    rel.inf
contact         contact 24.0610029
Size               Size 20.2922607
city               city 19.2825705
Bathroom       Bathroom 18.1562254
Totalfloors Totalfloors  7.9060981
BHK                 BHK  6.3237123
furn               furn  1.7745468
Floor             Floor  0.9165826
month             month  0.5340339
ten_pref       ten_pref  0.4089946
area_type     area_type  0.3439722
```



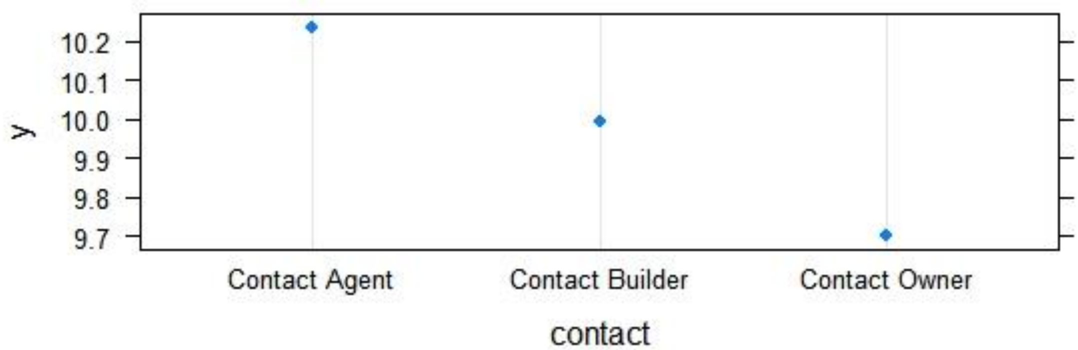
Exploring the partial plots of some of the most influential predictor variables with respect to the logrent price.



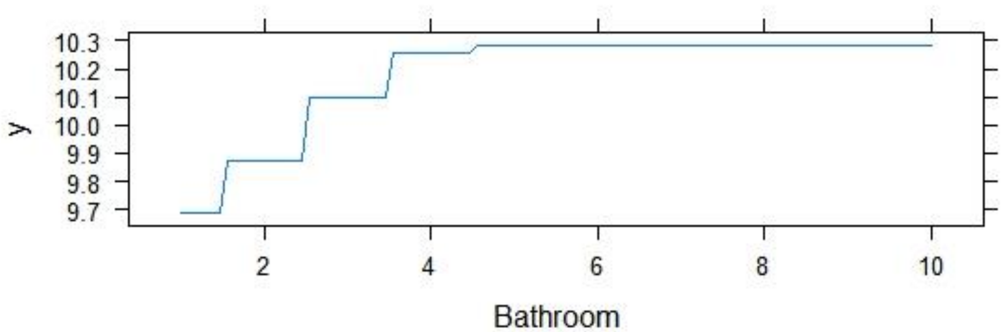
We can observe the increasing trend of the logprice of the property with respect to the size. After the 4000sqft. The price does not vary much.



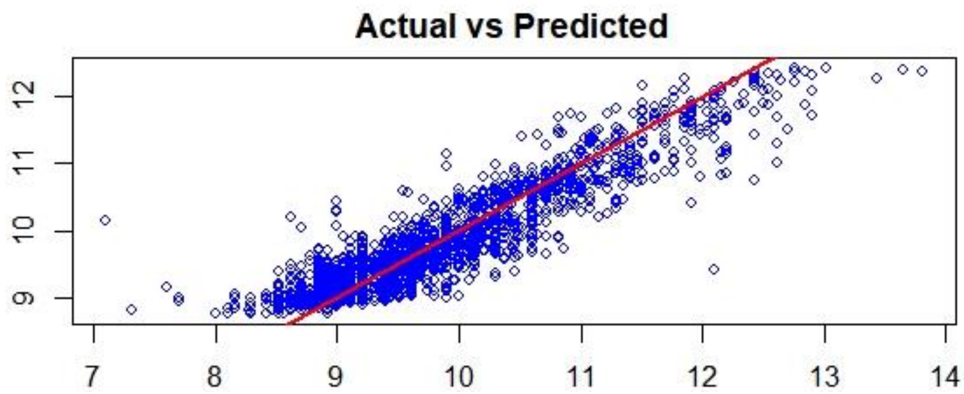
In the plot we can observe that the cities with higher rent prices are Mumbai and then Dheli. And the cheapest cities are Kolkata and Hyderabad.



If the property is rented through a contact agent the rent is higher than if it is rented by the builder or the owner.



We can observe that from 1 to 1.5 bathrooms there is an increase in price, then from 2.5 to 3.5 and a small increase at 4.5 after that the price of the house is uniform and does not increase if the number of bathrooms increases.

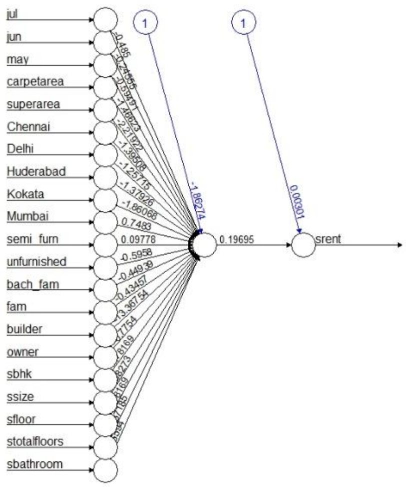


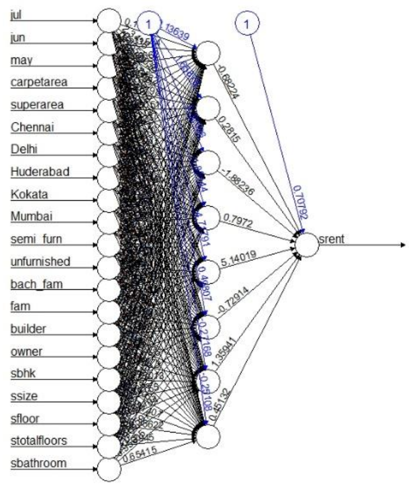
The MSE for the boosting model is 0.15 a little higher than the Random Forest model.

MODEL	MSE
Linear Regression	0.1704
Tree	0.2355
Random Forest	0.147
Boosting	0.15

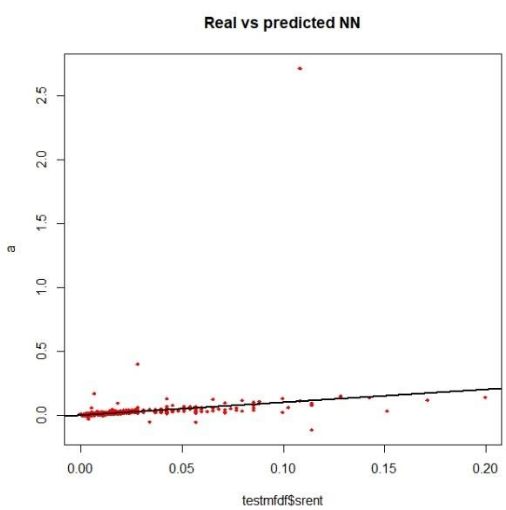
Neural Network

A neural network is a series of algorithms that mimics the operations of an animal brain to recognize relationships between vast amounts of data. In this dataset we used neural network first with one node and then with eight nodes, in both cases we used one hidden layer.





The model needs more toning, but by the actual vs predicted plot we can visually see the accuracy of the model.



Appendix

```
####################################################################
#################    House Rent in India Data Set    #############
####################################################################

library("dplyr")
library("readr")
library("rpart")
library("ggplot2")
library("rattle")
library("rpart.plot")
library("RColorBrewer")
library("randomForest")
library("ROCR")
library("reprtree")
library("caret")
library("gbm")
#Reading data into R
df <- read_csv("HR1.csv")
#EDA
#Looking for missing data and transforming variables
summary(df)
# No misssing data
df$month <- as.factor(df$month)
df$area_type <- as.factor(df$`Area Type`)
#df$area_loc <- as.factor(df$`Area Locality`)
```

```
df$city <- as.factor(df$City)

df$furn <- as.factor(df$`Furnishing Status`)

df$ten_pref <- as.factor(df$`Tenant Preferred`)

df$contact <- as.factor(df$`Point of Contact`)

df <- select(df, -c(`Area Type`,`Area Locality`,`City`,`Furnishing Status`,`Tenant Preferred`,`Point of
Contact`,Montherase))


#Log of Rent

df$lrent <- log(df$Rent)

df<- select(df, -c(Rent,`Posted On`))

summary(df)


#==================================================================

#  Dividing the data in to training and test set

#==================================================================

n <- NROW(df)

set.seed(123)

id.test<- sample(1:n, size = n*0.6)

dat.training<-df[id.test,]

dat.test<- df[-id.test,]


#====================================================================

#  Fit the Tree

#====================================================================

tre0 <- rpart(lrent~.,data = dat.training)

summary(tre0)

par(mfrow=c(1,1),mar=c(4,1,1,2))

plot(tre0); text(tre0)

par(mfrow=c(2,1), mar=c(4, 6, 4, 6))

rsq.rpart(tre0)
```

```
printcp(tre0)




# ----------------------------------
# OBTAINING THE BEST TREE
# ----------------------------------
cv.error <- (tre0$cptable)[,4]
a0 <- 1   # SORT OF ARBITRARY; MAY CHANGE TO ANOTHER VALUE
SE1 <- min(cv.error) + a0*((tre0$cptable)[,5])[which.min(cv.error)]      # 1SE
position <- min((1:length(cv.error))[cv.error <= SE1])
n.size  <- (tre0$cptable)[,2] + 1  # TREE SIZE IS ONE PLUS NUMBER OF SPLITS.
best.size.1SE <- n.size[position]; best.size.1SE
best.cp <-  sqrt(prod(tre0$cptable[(position-1):position,1]))
best.cp

# ---------------------------
# EXPLORE THE BEST TREE MODEL
# ---------------------------
best.tree <- prune(tre0, cp=best.cp)
best.tree
summary(best.tree)
# TREE PLOTS
par(mar = c(1,1,1,1))
prp(best.tree, main="Decision Tree Model for House Rent Prices",
   type=0, box.palette="auto",  # auto color the nodes based on the model type
   faclen=0)
```

```
#-----------------------------------------------------------------------
####################Prediction####################################
#-----------------------------------------------------------------------
yhattr=predict(best.tree,dat.test)
plot(dat.test$lrent,yhattr,xlab="actual",ylab="predicted",main="Actual vs Predicted",col="blue",cex=0.8)
abline(0,1, col="red",lwd=2)
mean((yhattr-dat.test$lrent)^2)

#######################
# RANDOM FOREST
#######################
set.seed(123)
rfmodel=randomForest(lrent~. , data = dat.training,mtry=11)
par(mar=c(2,1,1,2))
plot(rfmodel, main = "Number of trees vs OOB error rate")
round(importance(rfmodel), 2)
par(mar=c(1,1,1,1))
varImpPlot(rfmodel, main="Variable Importance Ranking")

# SEARCH THE BEST mtry PARAMETER, NUMBER OF VARIABLES RANDOMLY SAMPLED AT EACH SPLIT
#x<-as.matrix(dat.training[-c(12,7,10)])
#y<-as.vector(dat.training[,12])
par(mar=c(2,2,2,2))
m.try <- tuneRF(dat.training[-12],dat.training$lrent , ntreeTry=500, stepFactor=1.5,
            improve=0.01, trace=TRUE, plot=TRUE, dobest=FALSE)
best.m <- m.try[m.try[, 2] == min(m.try[, 2]), 1]; best.m

# RANDOM FOREST
#Best m.try=4
```

```
bestrf <- randomForest(lrent ~ .,
                data=dat.training,  mtry=best.m, ntree=100, keep.forest=TRUE,
                importance=TRUE, proximity=TRUE, oob.prox=FALSE)
print(bestrf)
round(importance(bestrf), 2)
par(mar=c(2,2,2,2))
varImpPlot(bestrf, main="Variable Importance Ranking")
importance(bestrf)
getTree(bestrf, 1, labelVar=TRUE)
par(mar=c(1,1,1,1))
MDSplot(bestrf,dat.training$lrent)
#par(mar = c(0.0001,0,0,0))
#reprtree:::plot.getTree(bestrf)
# PREDICTION
yhatrf <- predict(bestrf, newdata=dat.test)
par(mar=c(2,2,2,2))
plot(dat.test$lrent, yhatrf,xlab="actual",ylab="predicted",main="Actual vs Predicted", col="blue", cex=0.8)
abline(a=0, b=1, col="red", lwd=2)
mean((yhatrf-dat.test$lrent)^2)

################################################################################
                BOOSTING
################################################################################
set.seed(123)
boostmodel=gbm(lrent~.,data=dat.training,distribution="gaussian", n.trees = 10000,shrinkage = 0.001,
interaction.depth = 6, cv.folds = 5)
par(mar=c(4,4,1,15))
gbm.perf(boostmodel,plot.it = TRUE,oobag.curve=TRUE,overlay = FALSE,method="cv")
par(mar=c(3,5,3,3))
summary.gbm(boostmodel) #variables of importance
```

```
#partial plot of variables with greater relative influence

par(mfrow=c(1,2))

plot(boostmodel,i="Size")

plot(boostmodel,i="city")

plot(boostmodel,i="contact")

plot(boostmodel,i="Bathroom")


# PREDICTION

yhatg =predict(boostmodel, newdata = dat.test, n.trees = 3000)

mean((yhatg-dat.test$lrent)^2) #MSE0.154


#Modifying the shrinkage parameter

boostmodel2=gbm(lrent~.,data = dat.test,distribution = "gaussian", n.trees=3000, interaction.depth = 6, shrinkage
= 0.001)

par(mar=c(2,3,0.01,3))

summary.gbm(boostmodel2,plotit = TRUE)

plot(boostmodel2,i="Size")

plot(boostmodel2,i="city")

plot(boostmodel2,i="contact")

plot(boostmodel2,i="Totalfloors")

names(boostmodel2)

#Variables of importance

#summary.gbm(boostmodel2)

yhatg2 =predict(boostmodel2, newdata = dat.test, n.trees =3000)

mean((yhatg2-dat.test$lrent)^2)

par(mar=c(2,2,2,2))

plot(dat.test$lrent, yhatg2,xlab="actual",ylab="predicted",main="Actual vs Predicted", col="blue", cex=0.8)

abline(a=0, b=1, col="red", lwd=2)

# Second model, with lambda 0.001 smaller MSE 0.15
```

Neural Network

```
using CSV, DataFrames, RCall
HR1= CSV.File("C:\\Users\\MPVC_\\OneDrive\\Escritorio\\PhD\\Stat 5428\\HR1.csv") |> DataFrame
describe(HR1)

#Erase coloumns that are not going to be used
df = select!(HR1,Not(:"Posted On"))
df = select!(df,Not(:"Montherase"))
df= select!(df,Not(:"Area Locality"))
df
using CategoricalArrays, StatsModels

## converting strings to categorical variables
df.month=CategoricalArray(df.month)
df."Area Type"=CategoricalArray(df."Area Type")
df.City=CategoricalArray(df.City)
df."Furnishing Status"=CategoricalArray(df."Furnishing Status")
df."Tenant Preferred"=CategoricalArray(df."Tenant Preferred")
df."Point of Contact"=CategoricalArray(df."Point of Contact")
df

#Renaming coloumns
cnames=["month", "bhk", "rent", "size", "floor", "totalfloors", "areatype", "city",
"furnstat","tenantpref","bathroom","contact"]
rename!(df,Symbol.(cnames))

## coding categories into dummies
mf = ModelFrame(@formula(bhk ~ 1 + month+areatype+city+furnstat+tenantpref+contact+contact), df, contrasts
= Dict(:x => DummyCoding(base="abr"), :x=>DummyCoding(base="Super Area"),
:x=>DummyCoding(base="Hyderabad"),:x=>DummyCoding(base="Unfurnished"),
:x=>DummyCoding(base="Family"), :x=>DummyCoding(base="Contact Owner")))

# matrix of dummies
mm=ModelMatrix(mf).m
mnames=coefnames(mf)
mfdf=DataFrame(mm,Symbol.(mnames))
println(mnames)

# changing the variable names to suitable
newnames=["intc","jul","jun","may","carpetarea","superarea","Chennai","Delhi","Huderabad","Kokata","Mumbai"
,"semi_furn","unfurnished","bach_fam","fam","builder","owner"]
rename!(mfdf,newnames)
describe(mfdf)
```

```
### Standardization by min and max for continuous variables
#Min Max Normalization
mfdf.sbhk= (df.bhk .- minimum(df.bhk)) ./
            (maximum(df.bhk) - minimum(df.bhk))
mfdf.srent = (df.rent .- minimum(df.rent)) ./ (maximum(df.rent) .- minimum(df.rent))
mfdf.ssize = (df.size .- minimum(df.size)) ./
            (maximum(df.size) .- minimum(df.size))
mfdf.sfloor = (df.floor .- minimum(df.floor)) ./
            (maximum(df.floor) .- minimum(df.floor))
mfdf.stotalfloors = (df.totalfloors .- minimum(df.totalfloors)) ./
            (maximum(df.totalfloors) .- minimum(df.totalfloors))
mfdf.sbathroom = (df.bathroom .- minimum(df.bathroom)) ./
            (maximum(df.bathroom) .- minimum(df.bathroom))
describe(mfdf)

# to get the right hand side formula
join(names(mfdf),"+")

## Consider a sample of size 60% for training data
using Random, Statistics, StatsBase, RDatasets
Random.seed!(23123)

function partitionTrainTest(mfdf, at = 0.6)
  n = nrow(mfdf)
  idx = shuffle(1:n)
  train_idx = view(idx, 1:floor(Int, at*n))
  test_idx = view(idx, (floor(Int, at*n)+1):n)
  mfdf[train_idx,:], mfdf[test_idx,:]
end

using RDatasets,RCall

trmfdf,testmfdf = partitionTrainTest(mfdf, 0.7) # 70% train

## loading neuralnet package from R
@rimport neuralnet as nnet

### model formula
nformu=@formula(srent~jul+jun+may+carpetarea+superarea+Chennai+Delhi+Huderabad+Kokata+Mumbai+semi_
furn+unfurnished+bach_fam+fam+builder+owner+sbhk+ssize+sfloor+stotalfloors+sbathroom)

#Model 0
nmodel = nnet.neuralnet(nformu,data=trmfdf,hidden=1,threshold = 0.01,  algorithm =
"rprop+",var"linear.output"=true);
nmodel["result.matrix"]
```

```
@rimport base as rbase
rbase.plot(nmodel)
#nnet.gwplot(nmodel, var"selected.covariate"="sage")

#Model 1
nmodel1 = nnet.neuralnet(nformu,data=trmfdf,hidden=8,threshold = 0.01,  algorithm =
"rprop+",var"linear.output"=true);
nmodel1["result.matrix"]
@rimport base as rbase
rbase.plot(nmodel1,rep='best')
#nnet.gwplot(nmodel1, var"selected.covariate"="sage")

####################################################
##### predictions on test Data
########################################################

#checkpred=nnet.prediction(nmodel1);
checkpred=nnet.compute(nmodel1,trmfdf, rep=1);

# categorical outcome
tpred= nnet.compute(nmodel1, testmfdf,rep=1);
names(tpred)
a=tpred[Symbol("net.result")];
#[a[3500:3521],a[7021:7042]]

#MSE
#test.r <- (testmfdf$srent)*(max(df$rent)-min(df$rent))+min(df$rent)
MSEnn=sum((testmfdf.srent - tpred.net.result)^2)/nrow(testmfdf)

#Plot Actual vs predicted
@rput testmfdf tpred a
R" par(mfrow=c(1,1))"
R"plot(testmfdf$srent,a,col='red',main='Real vs predicted NN',pch=18,cex=0.7)"
R" abline(0,1,lwd=2)"
R"legend('bottomright',legend='NN',pch=18,col='red', bty='n')"
```