Predicting_bangalore_house_price

Loading data

```
Train_bangalore <- read.csv("Train_banglore_machinehack.csv")
str(Train_bangalore)
```

```
## 'data.frame':    13320 obs. of  9 variables:
##  $ area_type   : Factor w/ 4 levels "Built-up  Area",..: 4 3 1 4 4 4 4 4 4 3 ...
##  $ availability: Factor w/ 81 levels "14-Jul","14-Nov",..: 41 81 81 81 81 81 35 81 81 81 ...
##  $ location    : Factor w/ 1306 levels ""," Anekal"," Banaswadi",..: 432 322 1213 776 729 128
8 917 988 815 444 ...
##  $ size        : Factor w/ 32 levels "","1 Bedroom",..: 16 20 19 19 16 16 21 21 19 25 ...
##  $ society     : Factor w/ 2689 levels "","3Codeli","7 ise P",..: 463 2440 1 2149 1 609 929 3
54 1 1 ...
##  $ total_sqft  : Factor w/ 2117 levels "1","1.25Acres",..: 71 1289 515 603 240 206 1320 1467
370 32 ...
##  $ bath        : int  2 5 2 3 2 2 4 4 3 6 ...
##  $ balcony     : int  1 3 3 1 1 1 NA NA 1 NA ...
##  $ price       : num  39.1 120 62 95 51 ...
```

Data Exploration

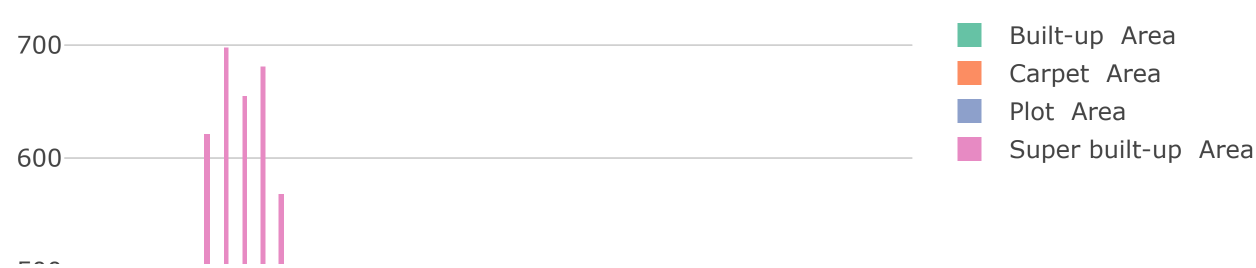
```
#we will start by looking at price skewness of all houses in bangalore as per data given
summary(Train_bangalore$price)
```

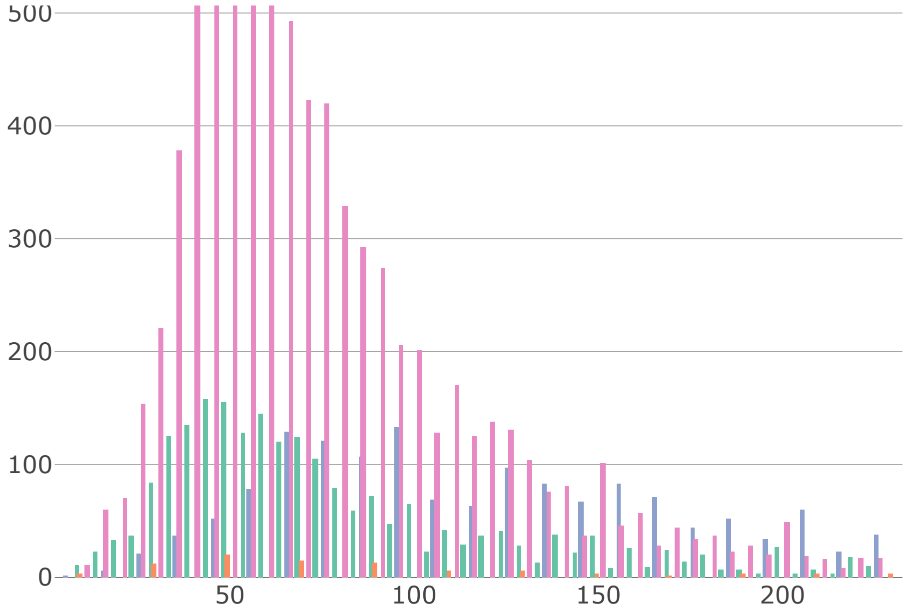
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     8.0    50.0    72.0   112.6   120.0  3600.0
```

So the price starts from 8lakh and goes upto 3600lakhs.the data is right skewed,where 75% of the houses are less than 1.2crore in price range.

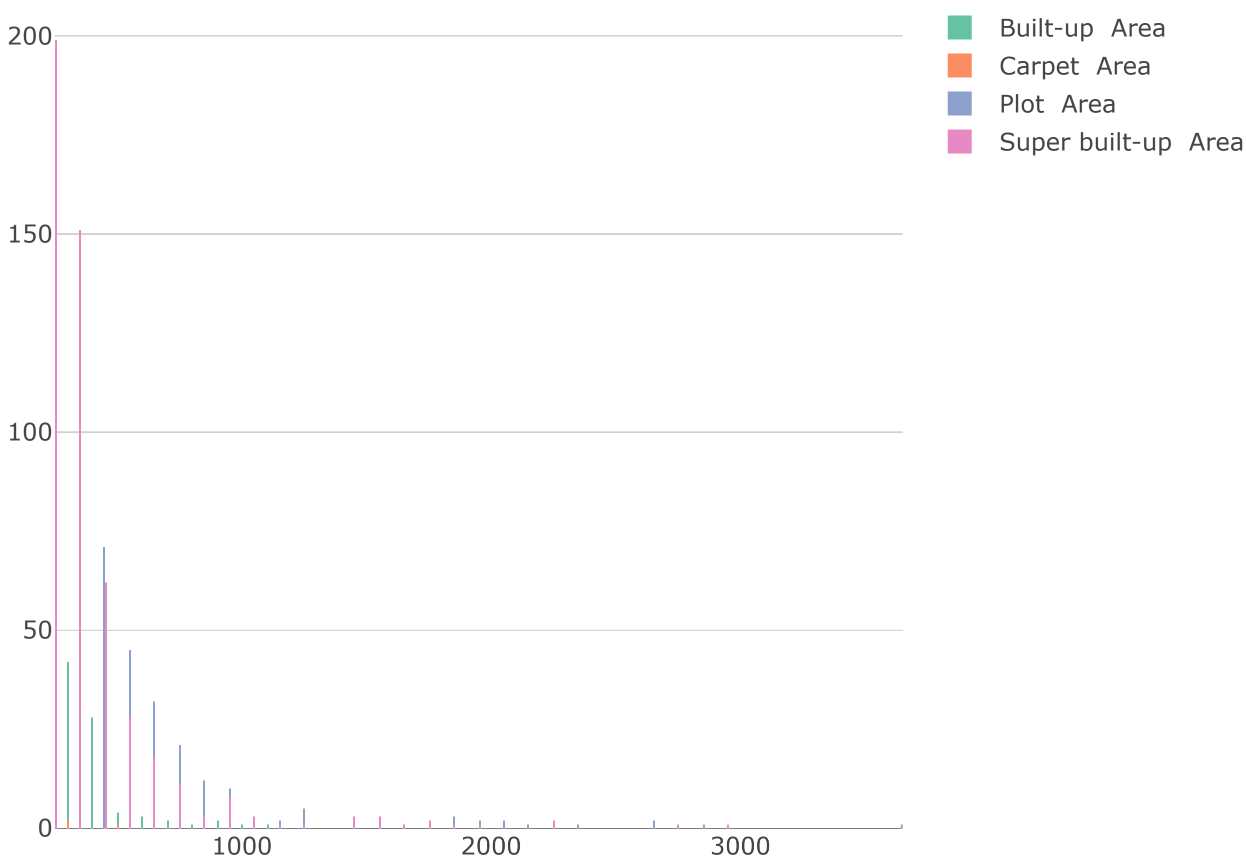
```
#how is the price distribution among diffferent areas in bangalore? without outliers
plot_ly(x=Train_bangalore$price[!Train_bangalore$price %in% boxplot.stats(Train_bangalore$price)
$out],type="histogram",color = Train_bangalore$area_type[!Train_bangalore$price %in% boxplot.sta
ts(Train_bangalore$price)$out])
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.2
```






```
#only outliers
plot_ly(x=Train_bangalore$price[Train_bangalore$price %in% boxplot.stats(Train_bangalore$price)
$out],type="histogram",color = Train_bangalore$area_type[Train_bangalore$price %in% boxplot.stat
s(Train_bangalore$price)$out])
```



We created two plot one with outliers and and other without.From the first(without outlier) we see that most of the houses are in range 50-100lakhs and number is high for type ,superbuilt-up Area

Since Bangalore is an area where lots of working professional live with a better income,Super Built up area provide amenities like covered community centres/ clubs, other covered common facilities.

Dealing Missing values

```
#whether belongs to other society
Train_bangalore$Othersocety<-ifelse(Train_bangalore$society=="","1","0")
Train_bangalore[Train_bangalore==""]<-NA
miss_val<-sapply(Train_bangalore,function(x) sum(is.na(x)))
miss_val
```

```
##    area_type availability     location         size      society 
##            0            0            1           16         5502 
##   total_sqft         bath      balcony        price  Othersocety 
##            0           73          609            0            0
```

Percentage of Missing values

```
percent_mis<-as.data.frame(round((miss_val/nrow(Train_bangalore))*100,1))
names(percent_mis)<-c("Percentage")
percent_mis
```

```
##              Percentage
## area_type           0.0
## availability        0.0
## location            0.0
## size                0.1
## society            41.3
## total_sqft          0.0
## bath                0.5
## balcony             4.6
## price               0.0
## Othersocety         0.0
```

41 percent of missing values are in Society,assuming missing values here as the following property are not in any society but some other place.Replacing Na's in society with other

```
Train_bangalore$society<-as.character(Train_bangalore$society)
Train_bangalore$society[is.na(Train_bangalore$society)]<-"Other"
Train_bangalore$society<-as.factor(Train_bangalore$society)
length(Train_bangalore$society[is.na(Train_bangalore$society)])
```

```
## [1] 0
```

Balconies?

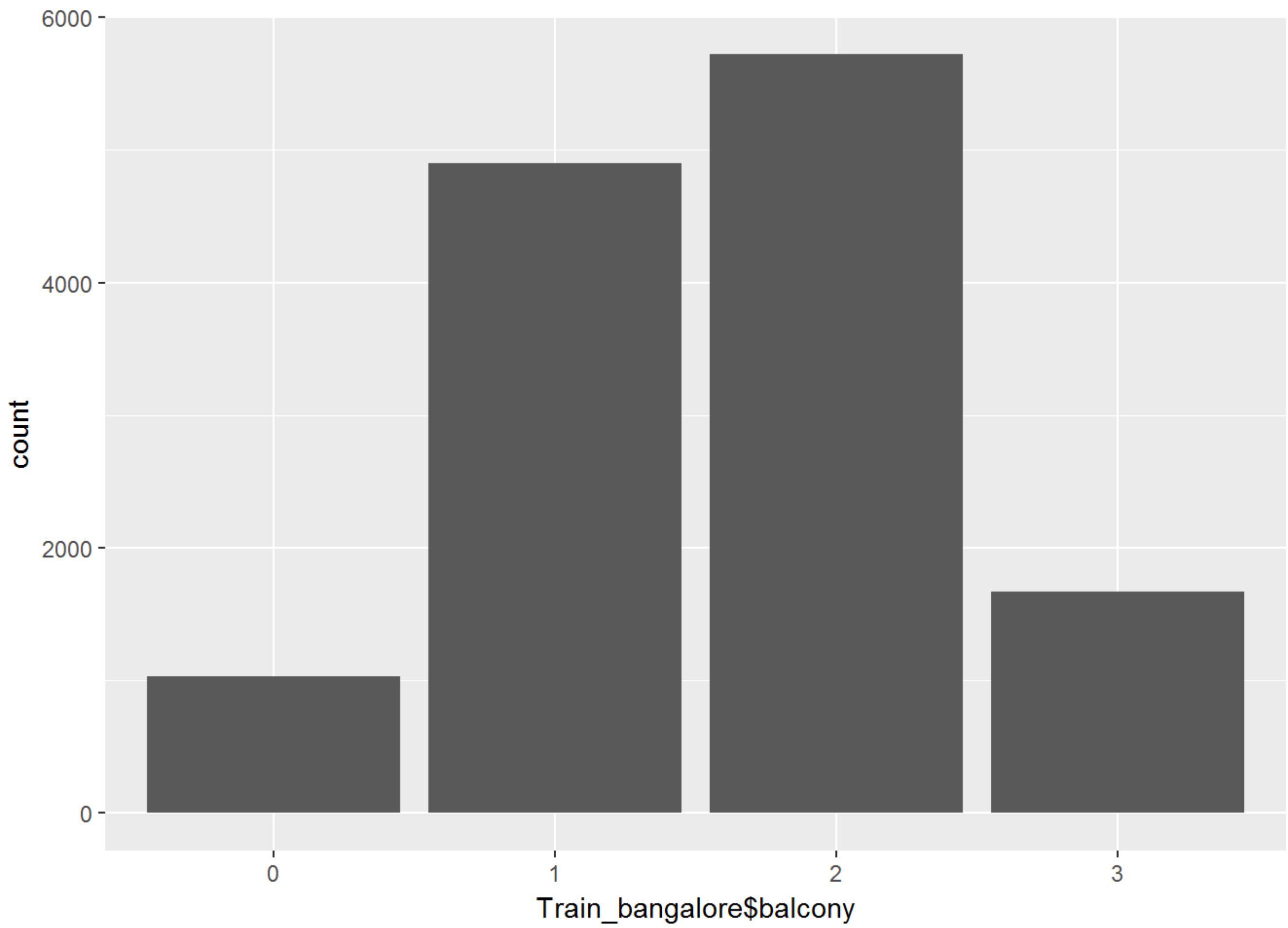
From the plot, we see that number of balconies are one or two in maximum properties,with two slightly higher.

Since missing values are less than .05 percent ,replacing with mode

```
#imputing missing value  for balcony with mode

Train_bangalore$balcony<-as.character(Train_bangalore$balcony)

Train_bangalore$balcony[is.na(Train_bangalore$balcony)]<-"2"

Train_bangalore$balcony<-as.factor(Train_bangalore$balcony)

table(Train_bangalore$balcony)
```

```
##
##    0    1    2    3
## 1029 4897 5722 1672
```

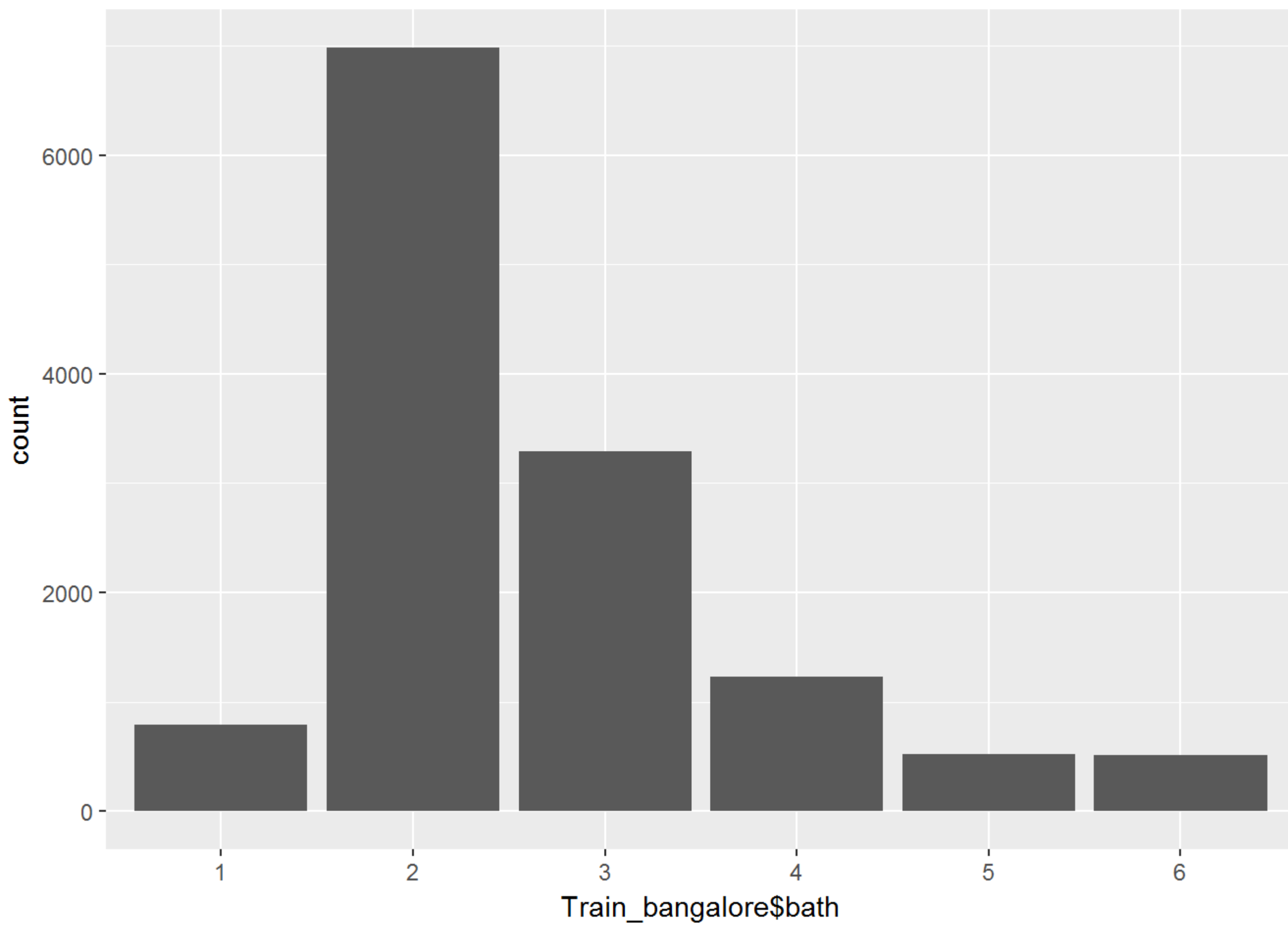
```
ggplot(Train_bangalore,aes(Train_bangalore$balcony))+geom_bar()
```



```
Train_bangalore$bath[Train_bangalore$bath>=6]<-6

Train_bangalore$bath<-as.character(Train_bangalore$bath)

#replacing NA with mode value
Train_bangalore$bath[is.na(Train_bangalore$bath)]<-"2"

Train_bangalore$bath<-as.factor(Train_bangalore$bath)

table(Train_bangalore$bath)
```

```
##
##    1    2    3    4    5    6
##  788 6981 3286 1226  524  515
```

```
ggplot(Train_bangalore,aes(Train_bangalore$bath))+geom_bar()
```



Area

Size and price

```
names(Train_bangalore[,1:4])
```

```
## [1] "area_type"    "availability" "location"     "size"
```

I made a new variable which separate society and not in society,price seems same for each area type ,some of them which are not in societies have prices when area is super-built area.

```
#converting size variable into character variable
Train_bangalore$size<-as.character(Train_bangalore$size)
```

Which locations has highest mean price ?

```
mean_price_per_location<-as.data.frame(arrange(aggregate(Train_bangalore$price,by=list(Train_ban
galore$location),mean),desc(x)))

names(mean_price_per_location)<-c("location","Price_Mean")
#grouping based on pricing
mean_price_per_location$Price_Range_group<-cut(mean_price_per_location$Price_Mean,c(0,100,200,50
0,1000,2000),labels = c("lowprice","averagepriced","HighPrice","veryHigh","extreme"))
#places where price is higher
head(mean_price_per_location[mean_price_per_location$Price_Range_group=="extreme"||mean_price_pe
r_location$Price_Range_group=="veryHigh"],20)
```

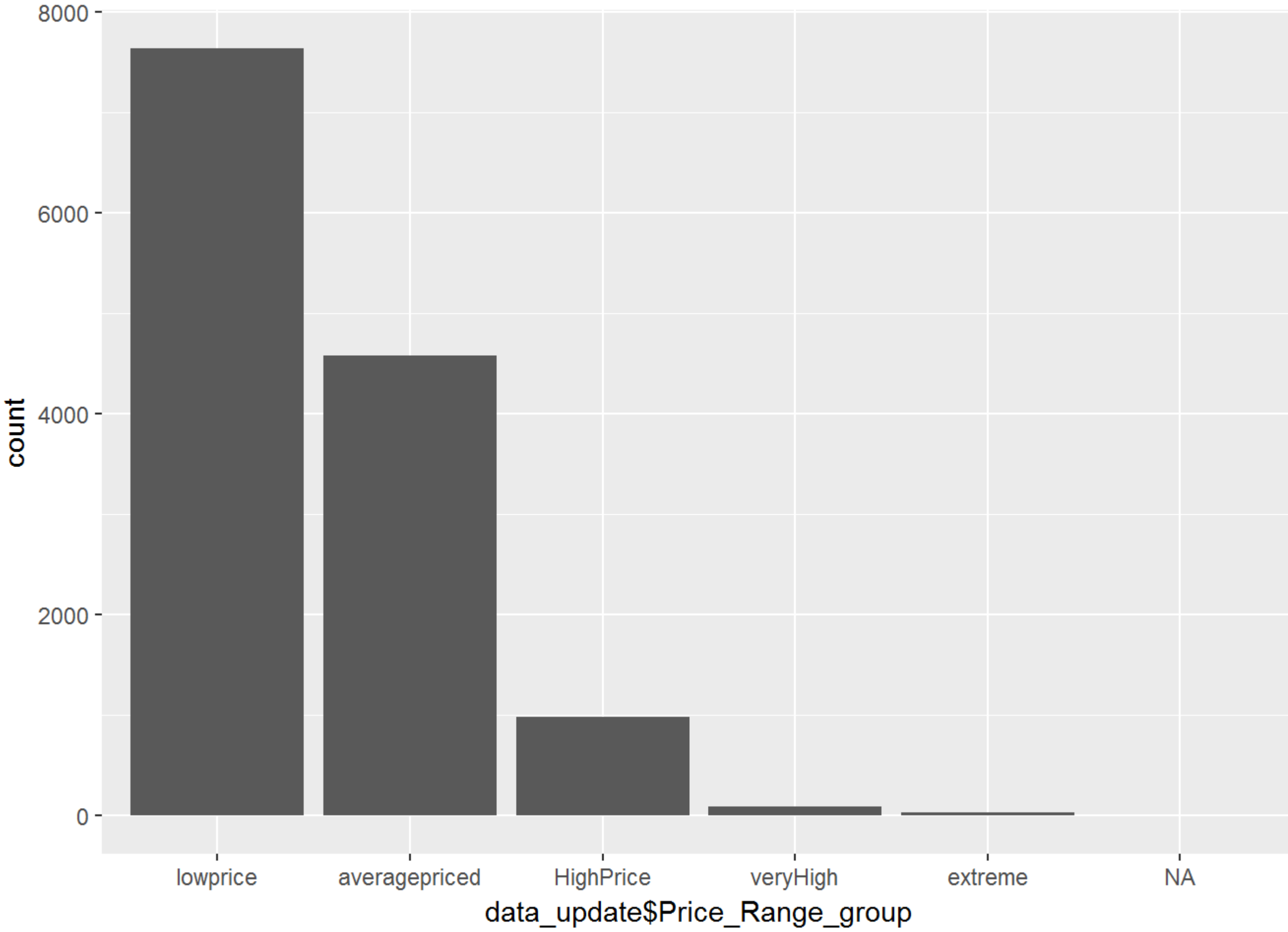
```
##                 location Price_Mean Price_Range_group
## 1            Cubbon Road  1900.0000           extreme
## 2            Ashok Nagar  1486.0000           extreme
## 3         Defence Colony  1167.7143           extreme
## 4                 Yemlur  1093.3889           extreme
## 5          Church Street  1068.0000           extreme
## 6         D Souza Layout  1015.0000           extreme
## 7        Sadashiva Nagar  1011.1000           extreme
## 8          Sindhi Colony   988.0000          veryHigh
## 9        Srinivas Colony   922.0000          veryHigh
## 10   5th Block Jayanagar   905.0000          veryHigh
## 11           Binnamangala   900.0000          veryHigh
## 12       Cunningham Road   824.3846          veryHigh
## 13     Hunasamaranahalli   787.5000          veryHigh
## 14 2nd Block Koramangala   761.0000          veryHigh
## 15       Shanthala Nagar   754.0000          veryHigh
## 16        Dollars Colony   744.3750          veryHigh
## 17           Kathreguppe   725.0000          veryHigh
## 18   Sector 4 HSR Layout   700.0000          veryHigh
## 19       Rest House Road   690.0000          veryHigh
## 20 Ramakrishnappa Layout   662.8571          veryHigh
```

merging the new coloumn.

```
data_update<-left_join(Train_bangalore,mean_price_per_location,by="location")
Train_bangalore<-left_join(Train_bangalore,mean_price_per_location,by="location")
```

Distribution of houses based on prices in bangalore

```
ggplot(data_update,aes(data_update$Price_Range_group))+geom_bar()
```



Around 80% of houses are in range less than a crore.

Cleanig Variable Total_sqft

How much is the area of a house or plot. For any analysis we need to clean the variable in a single format i.e. Squarefeet

```
ll<-Train_bangalore$total_sqft

ll<-as.data.frame(ll)

ll$ll<-as.character(ll$ll)
c<-grep("Acres",ll$ll,ignore.case = T)
b<-grep("Sq. Yards",ll$ll,ignore.case = T)
d<-grep("Sq. Meter",ll$ll,ignore.case = T)
e<-grep("Cents",ll$ll,ignore.case = T)
f<-grep("-",ll$ll,ignore.case = T)
g<-grep("Perch",ll$ll,ignore.case = T)


ll[c,]<-gsub("Acres","",ll[c,],ignore.case = T)
ll[b,]<-gsub("Sq. Yards","",ll[b,],ignore.case = T)
ll[d,]<-gsub("Sq. Meter","",ll[d,],ignore.case = T)
ll[e,]<-gsub("Cents","",ll[e,],ignore.case = T)
ll[g,]<-gsub("Perch","",ll[g,],ignore.case = T)

# for values noted as for example 1600-1700, are replaced with their mean values
rr<-as.data.frame(ll[f,])
names(rr)<-"name"
head(rr)
```

```
##          name
## 1 2100 - 2850
## 2 3010 - 3410
## 3 2957 - 3450
## 4 3067 - 8156
## 5 1042 - 1105
## 6 1145 - 1340
```

```
rr$name<-as.character(rr$name)
pp<-separate(rr,name,into = c("firstnumber","secondnumber"),sep = " - ")
head(pp)
```

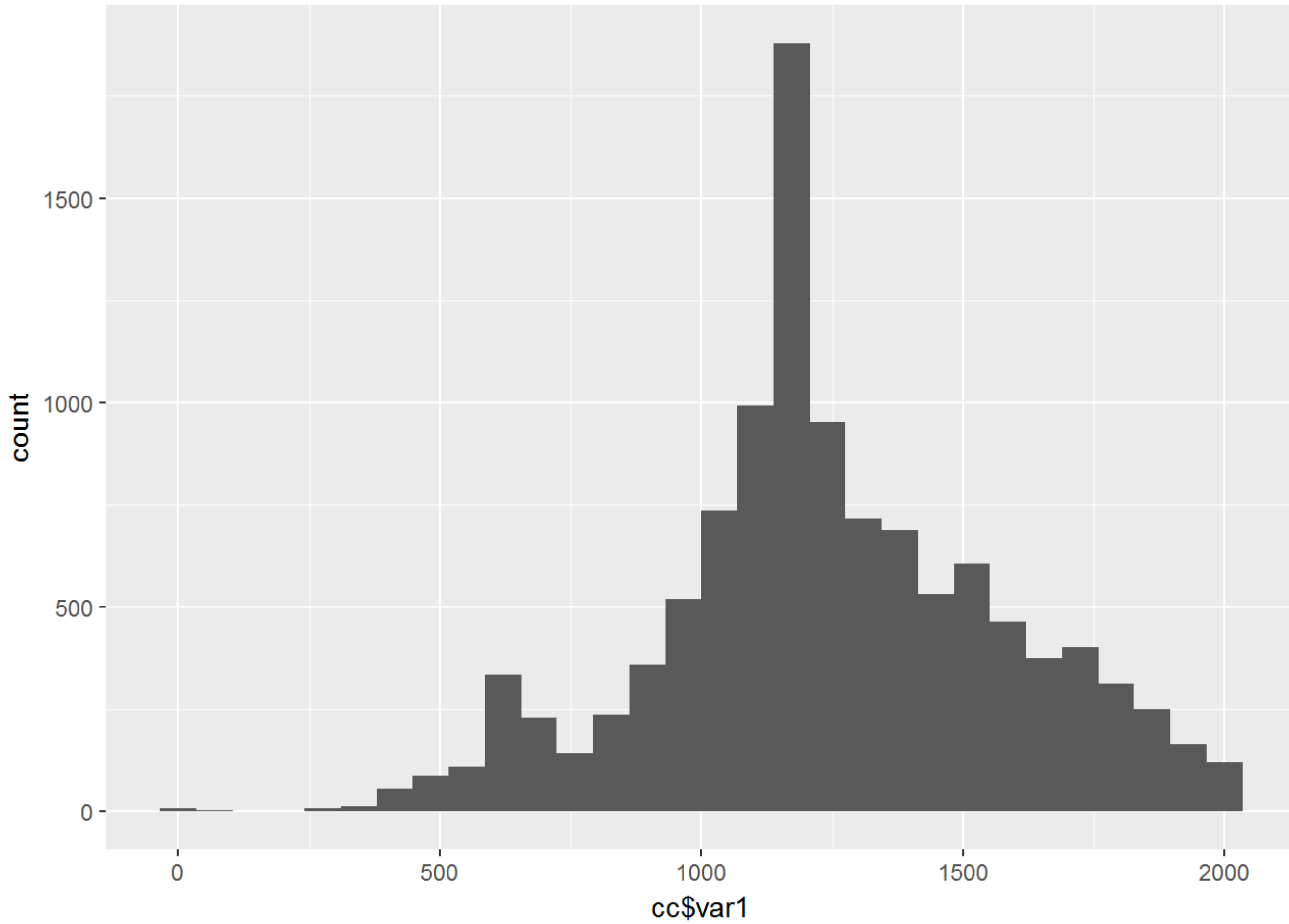
```
##   firstnumber secondnumber
## 1        2100         2850
## 2        3010         3410
## 3        2957         3450
## 4        3067         8156
## 5        1042         1105
## 6        1145         1340
```

```
pp$firstnumber<-as.numeric(pp$firstnumber)
pp$secondnumber<-as.numeric(pp$secondnumber)
final_pp<-(pp$firstnumber+pp$secondnumber)/2
#final conversions
ll[f,]<-final_pp
ll$ll<-as.numeric(ll$ll)
```

```
## Warning: NAs introduced by coercion
```

```
ll[which(is.na(ll)),]<-0

# 1 Acres=43560 sqft

ll[c,]<-ll[c,]*4350

#1sq yard = 9 sqft

ll[b,]<-ll[b,]*9

# 1sq.meter=10.76 sqft

ll[d,]<-ll[d,]*10.76

#1 cents =435.61 sqft

ll[e,]<-ll[e,]*435.61

#1 perch =272.75 sqft

ll[g,]<-ll[g,]*435.61

#replacing for values like 1600 -1900 with mean of there values

ll[f,]<-final_pp

Train_bangalore$total_sqft<-ll$ll
Train_bangalore$total_sqft<-as.numeric(Train_bangalore$total_sqft)

# histogram without outliers

cc<-as.data.frame(Train_bangalore$total_sqft[Train_bangalore$total_sqft<=2000])
names(cc)<-"var1"

ggplot(cc,aes(cc$var1,))+geom_histogram()
```

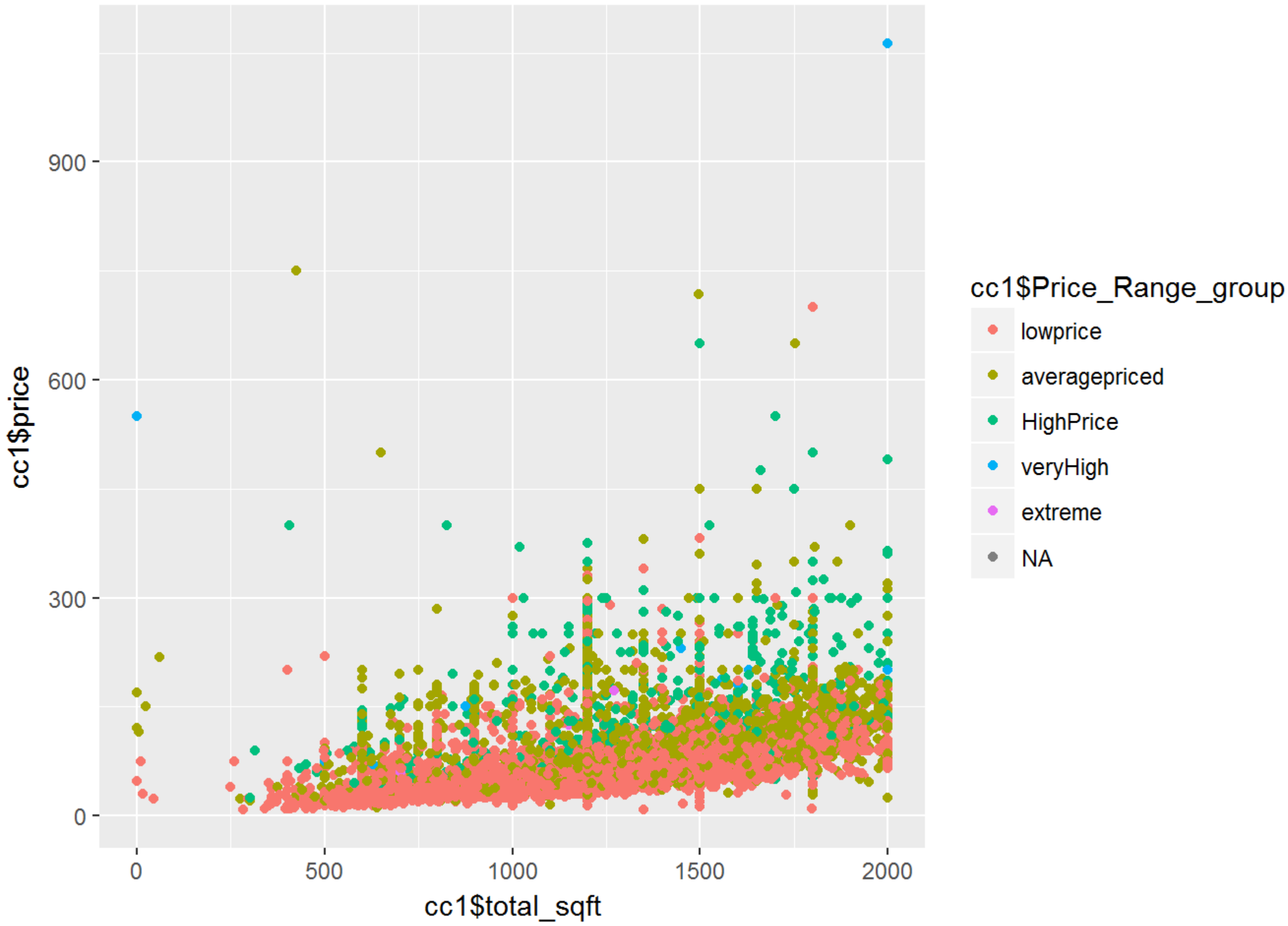
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



most of the plots are in the range from 1000 to 1500 square foot.

Price and and size relationship

```
cc1<-Train_bangalore[Train_bangalore$total_sqft<=2000,]
ggplot(cc1,aes(cc1$total_sqft,cc1$price,color=cc1$Price_Range_group))+geom_point()
```

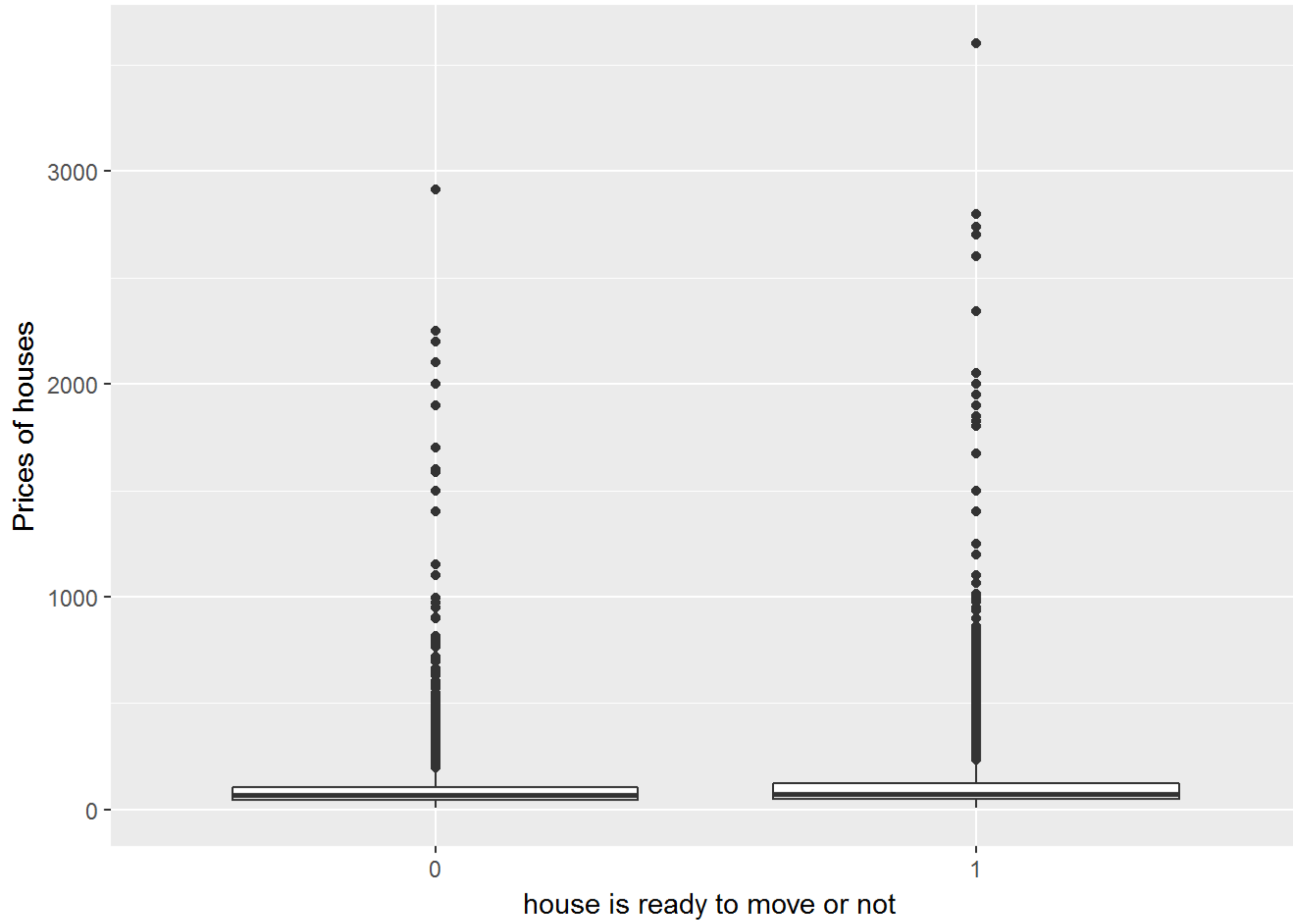


Now we would like to check for the availability and price change. Hypothesis here is that nearer the house available to move in, higher should be the price .Let's analyse the variable Availability

```
head(Train_bangalore$availability)
```

```
## [1] 19-Dec        Ready To Move Ready To Move Ready To Move Ready To Move
## [6] Ready To Move
## 81 Levels: 14-Jul 14-Nov 15-Aug 15-Dec 15-Jun 15-Nov 15-Oct ... Ready To Move
```

```
Train_bangalore$AvailibityCheck<-ifelse(Train_bangalore$availability=="Ready To Move",1,0)
Train_bangalore$AvailibityCheck<-as.factor(Train_bangalore$AvailibityCheck)
ggplot(Train_bangalore,aes(Train_bangalore$AvailibityCheck,Train_bangalore$price))+geom_boxplot
()+ylab("Prices of houses")+xlab(" house is ready to move or not")
```




```
#Cleaning size variable
ss<-Train_bangalore$size
y<-sapply(strsplit(ss," "),head,1)
y<-as.numeric(y)
Train_bangalore$size<-y
Train_bangalore$Othersocety<-as.factor(Train_bangalore$Othersocety)
```

Model Builing

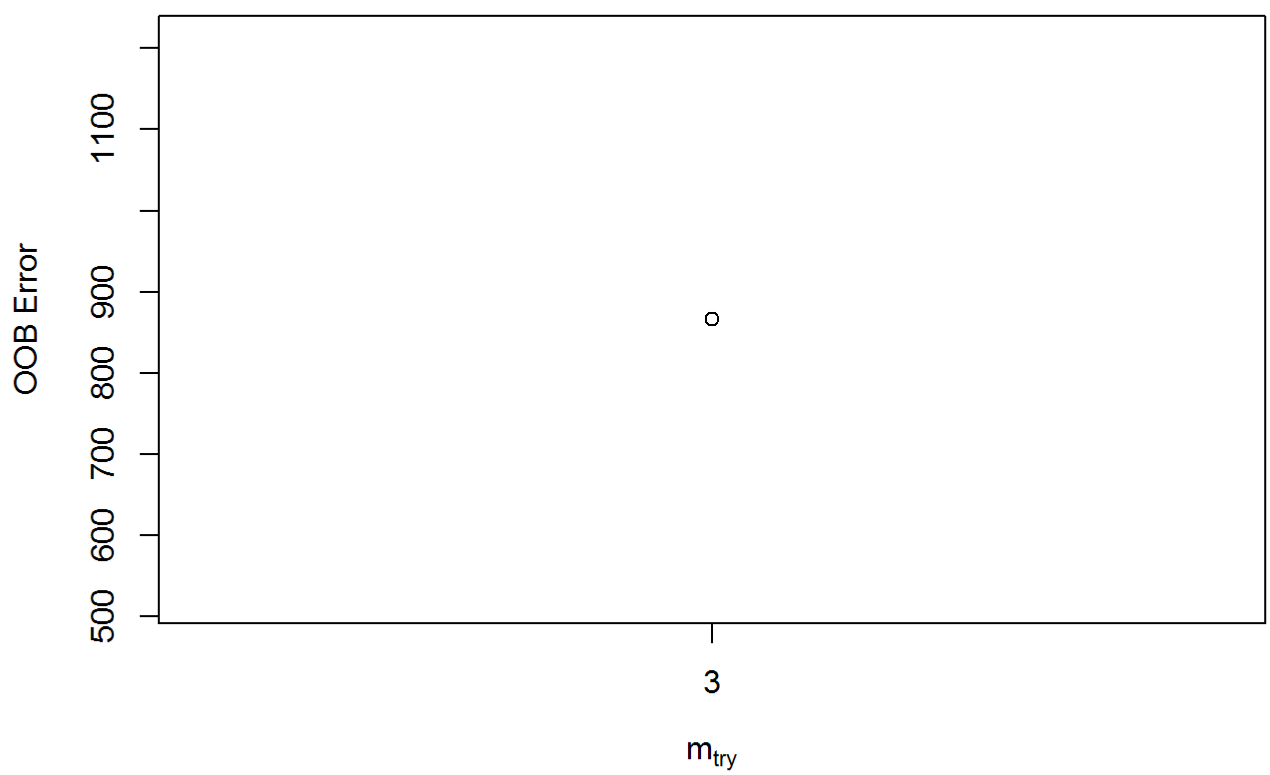
Partitioning data into Train and test set for validation

```
Train_bangalore<-na.omit(Train_bangalore)

Train_bangalore1<-Train_bangalore[,-c(2,3,5)]

Train_bangalore1$size<-as.factor(Train_bangalore1$size)

index<-sample(2,nrow(Train_bangalore1),prob = c(0.7,0.3),replace = TRUE)

bangalore_train<-Train_bangalore1[index==1,]

bangalore_test<-Train_bangalore1[index==2,]
```

Parameter tuning for RandomForest

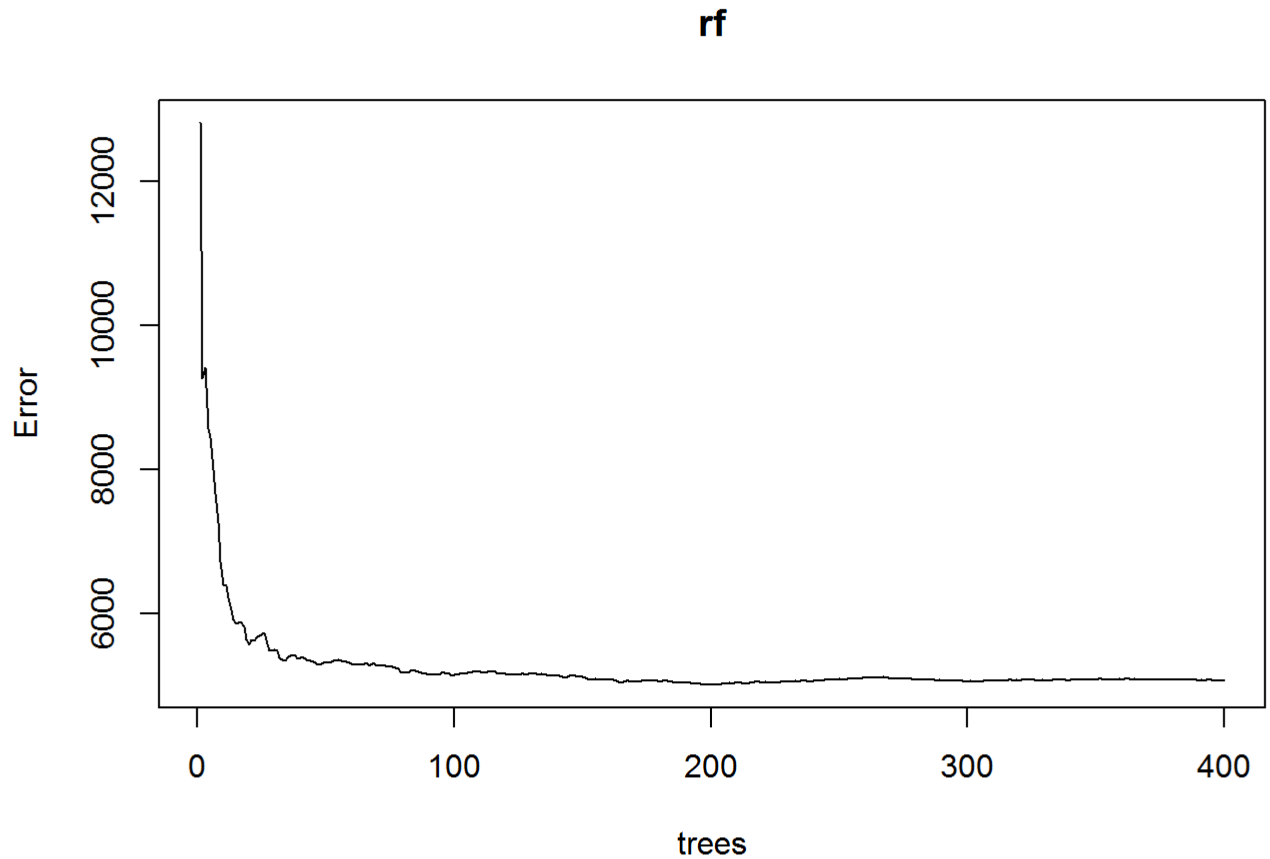
```
#optimised value of mtry

optimalvalue<-tuneRF(bangalore_train,bangalore_train$price,stepFactor = 1.2,improve = 0.1,trace
 = T,plot = T)
```

```
## mtry = 3  OOB error = 865.9057
## Searching left ...
## Searching right ...
```



Training Random Forest model

```
rf<-randomForest(bangalore_train$price~.,data = bangalore_train,ntree=400,mtry=3,importance=TRUE
)

plot(rf)
```



making prediction

```
#making predictions for test data
pred<-predict(rf,bangalore_test)

temp<-as.data.frame(cbind(pred,bangalore_test$price))

names(temp)<-c("predicted","actual")

temp$difference<-temp$actual-temp$predicted

#function to calcultae RMSE
rmse <- function(error)
{
  sqrt(mean(error^2))
}
rmse(temp$difference)  #evaluation
```

```
## [1] 69.11329
```

Using Linear regression

```
linearModel<-glm(bangalore_train$price~.,data = bangalore_train)
summary(linearModel)
```

```
##
## Call:
## glm(formula = bangalore_train$price ~ ., data = bangalore_train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1066.14    -22.72     -1.03     17.30   1873.31
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -1.601e+01  6.960e+00  -2.301  0.02142 *
## area_typeCarpet  Area           2.115e+00  1.410e+01   0.150  0.88074
## area_typePlot  Area             3.484e+01  4.087e+00   8.525  < 2e-16 ***
## area_typeSuper built-up  Area  -8.430e-01  2.921e+00  -0.289  0.77292
## size2                           7.680e+00  8.925e+00   0.861  0.38953
## size3                           2.597e+01  9.480e+00   2.740  0.00616 **
## size4                           9.202e+01  1.076e+01   8.551  < 2e-16 ***
## size5                           8.020e+01  1.272e+01   6.305 3.01e-10 ***
## size6                          -5.787e+00  1.459e+01  -0.397  0.69164
## size7                          -1.861e+01  1.724e+01  -1.080  0.28031
## size8                          -3.037e+01  1.888e+01  -1.608  0.10782
## size9                          -1.940e+01  2.102e+01  -0.923  0.35604
## size10                          1.246e+01  3.172e+01   0.393  0.69441
## size11                         -9.906e+01  7.401e+01  -1.338  0.18080
## size12                          8.345e+01  1.037e+02   0.805  0.42107
## size14                         -6.678e+01  1.037e+02  -0.644  0.51964
## size16                         -1.090e+02  1.045e+02  -1.044  0.29670
## size19                          4.726e+00  1.040e+02   0.045  0.96375
## total_sqft                      1.797e-04  5.783e-05   3.107  0.00190 **
## bath2                          -1.568e+00  8.241e+00  -0.190  0.84910
## bath3                           1.747e+01  9.024e+00   1.935  0.05297 .
## bath4                           5.224e+01  1.010e+01   5.171 2.37e-07 ***
## bath5                           1.088e+02  1.142e+01   9.529  < 2e-16 ***
## bath6                           1.514e+02  1.304e+01  11.605  < 2e-16 ***
## balcony1                        1.273e+01  4.397e+00   2.896  0.00378 **
## balcony2                        1.924e+01  4.464e+00   4.309 1.65e-05 ***
## balcony3                        1.451e+01  5.164e+00   2.809  0.00498 **
## Othersocety1                   -1.454e+01  2.407e+00  -6.040 1.60e-09 ***
## Price_Mean                      6.957e-01  3.683e-02  18.889  < 2e-16 ***
## Price_Range_groupaveragepriced -9.121e+00  3.143e+00  -2.902  0.00372 **
## Price_Range_groupHighPrice      1.108e+01  8.741e+00   1.267  0.20513
## Price_Range_groupveryHigh       1.243e+02  2.618e+01   4.747 2.10e-06 ***
## Price_Range_groupextreme        1.110e+02  4.346e+01   2.555  0.01063 *
## AvailibityCheck1               -6.264e+00  2.722e+00  -2.302  0.02137 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 10553.35)
##
##     Null deviance: 211038905  on 9403  degrees of freedom
## Residual deviance:  98884926  on 9370  degrees of freedom
## AIC: 113844
```

```
##
## Number of Fisher Scoring iterations: 2
```

XGBOOST model

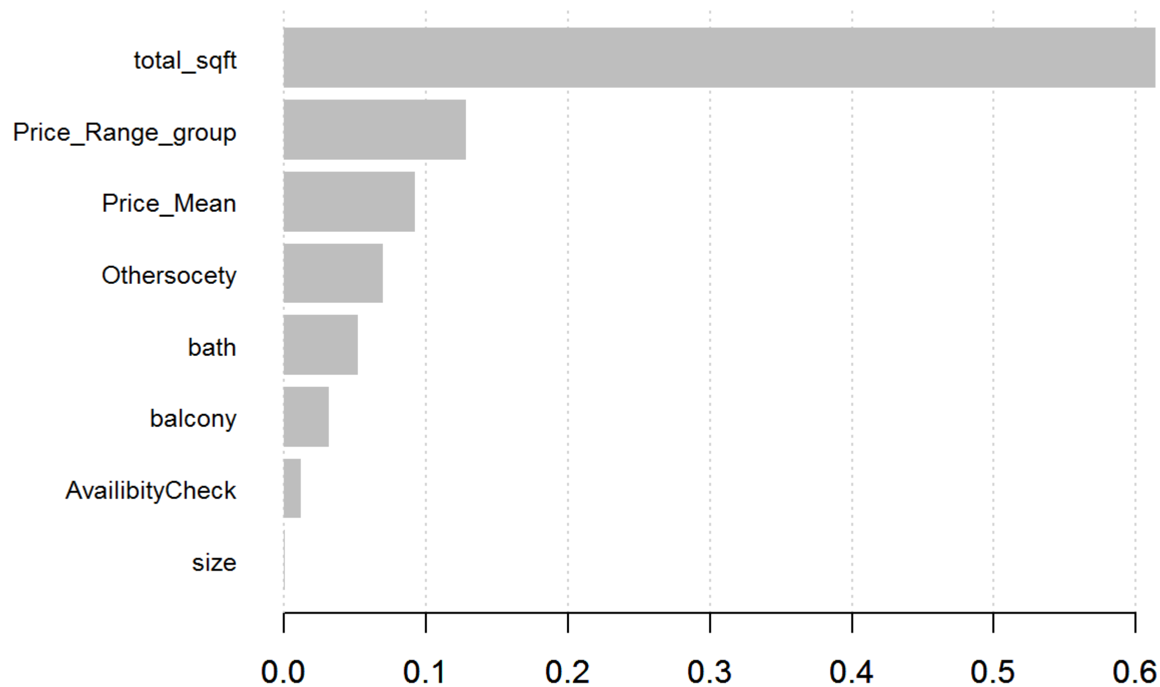
```
#One -Hot coding for factor variables
trainl<-sparse.model.matrix(price ~ .,data = bangalore_train)

#train label
train_label<-bangalore_train[,"price"]
train_matrix<-xgb.DMatrix(data = as.matrix(trainl),label= train_label)

#repeating steps for test
testl<-sparse.model.matrix(price ~ .,data = bangalore_test)
test_label<-bangalore_test[,"price"]
test_matrix<-xgb.DMatrix(data = as.matrix(testl),label= test_label)

#Parameters

xgb_params<-list(objective="reg:linear",eval_metric="rmse",eta=0.1,max_depth=5,lambda=2)
watchlist<-list(train=train_matrix,test=test_matrix)
modelXGB<-xgb.train(params = xgb_params,data = train_matrix,nrounds = 500,watchlist = watchlist,
maximize = FALSE,early_stopping_rounds = 40,print_every_n = 5)
```

```
## [1]  train-rmse:172.369385   test-rmse:170.400787
## Multiple eval metrics are present. Will use test_rmse for early stopping.
## Will train until test_rmse hasn't improved in 40 rounds.
##
## [6]  train-rmse:118.713539   test-rmse:121.178062
## [11] train-rmse:88.508797    test-rmse:94.146538
## [16] train-rmse:71.871666    test-rmse:82.068016
## [21] train-rmse:62.311623    test-rmse:75.281883
## [26] train-rmse:56.657112    test-rmse:71.428947
## [31] train-rmse:53.477612    test-rmse:70.729263
## [36] train-rmse:51.089413    test-rmse:70.333046
## [41] train-rmse:49.434509    test-rmse:69.940620
## [46] train-rmse:48.211140    test-rmse:69.946251
## [51] train-rmse:47.193668    test-rmse:69.779182
## [56] train-rmse:46.404305    test-rmse:69.653145
## [61] train-rmse:45.566044    test-rmse:69.573441
## [66] train-rmse:45.045650    test-rmse:69.470741
## [71] train-rmse:44.305092    test-rmse:69.382362
## [76] train-rmse:43.910839    test-rmse:69.346840
## [81] train-rmse:43.084015    test-rmse:69.292374
## [86] train-rmse:42.736874    test-rmse:69.294815
## [91] train-rmse:42.626606    test-rmse:69.271133
## [96] train-rmse:42.131073    test-rmse:69.273514
## [101]    train-rmse:41.813507    test-rmse:69.244514
## [106]    train-rmse:41.650127    test-rmse:69.254387
## [111]    train-rmse:41.410351    test-rmse:69.104622
## [116]    train-rmse:41.016613    test-rmse:69.189117
## [121]    train-rmse:40.376797    test-rmse:69.293488
## [126]    train-rmse:39.897789    test-rmse:69.290001
## [131]    train-rmse:39.532181    test-rmse:69.252182
## [136]    train-rmse:39.261616    test-rmse:69.200127
## [141]    train-rmse:39.049305    test-rmse:69.193222
## [146]    train-rmse:38.849602    test-rmse:69.202484
## [151]    train-rmse:38.688995    test-rmse:69.082817
## [156]    train-rmse:38.368317    test-rmse:69.003769
## [161]    train-rmse:38.111477    test-rmse:69.027664
## [166]    train-rmse:37.831203    test-rmse:69.042122
## [171]    train-rmse:37.631725    test-rmse:69.109375
## [176]    train-rmse:37.344601    test-rmse:69.105873
## [181]    train-rmse:37.069244    test-rmse:69.098663
## [186]    train-rmse:36.845310    test-rmse:69.048615
## [191]    train-rmse:36.699360    test-rmse:69.061630
## [196]    train-rmse:36.449409    test-rmse:68.981567
## [201]    train-rmse:36.269928    test-rmse:68.966957
## [206]    train-rmse:36.124897    test-rmse:69.026230
## [211]    train-rmse:36.035755    test-rmse:69.049965
## [216]    train-rmse:35.821648    test-rmse:68.983086
## [221]    train-rmse:35.578175    test-rmse:69.050995
## [226]    train-rmse:35.430328    test-rmse:69.034195
## [231]    train-rmse:35.232521    test-rmse:69.031723
## [236]    train-rmse:35.094696    test-rmse:69.073586
## [241]    train-rmse:34.832310    test-rmse:69.050201
```

```
## Stopping. Best iteration:
## [201]    train-rmse:36.269928    test-rmse:68.966957
```

Feature important from XGBoost

```
predictors<-names(bangalore_train)[!names(bangalore_train) %in% c("price")]
gg<-xgb.plot.importance(xgb.importance(model = modelXGB,feature_names = predictors))
```



We get same important variable from XGBoost also.