Apartment Selling Price Model For Queens

Final Project For Math 390.4 Data Science At Queens College

May 24, 2020

By: Sakif Shadman

Abstract: Machine Learning algorithm are playing very important in any walk of life. Machine learning have good predict power and widely used by manufactures and industrialist to promote their business and products. In our proposed project, we used the machine learning algorithms for sale prediction of the houses in Queens, New York. Random Forest , Linear Regression and Regression tree model algorithm are used to predict the selling price of the houses. Selling prices of the house is the regression task so the proposed algorithm can handle both regression and classification problems. The dataset used in this study is taken from MLSI website. The dataset is in raw format and needs too much preprocessing. The supervised machine learning algorithm out perform on the proposed dataset and shows a very good results in form of R2 and MSE. Random Forest and decision fit well over the dataset and does not show any over-fitting or under-fitting problem. Machine Learning algorithms are commonly used for sales analysis for future forecasting and predictive modeling.

Introduction: The sale price analysis is very important and necessary step for the promotion of business. According to view of sale prices prediction, stoke-holders and business man invest their money on those types of business where they have knowledge that they will get large profit and turnout. The sale future forecasting and analysis of sale prices is mostly examined through the help of Machine Learning based algorithms. Machine learning has three main branches 1. supervised 2. unsupervised and 3. reinforcement learning. Supervised machine learning has two main categories classification and regression. Unsupervised machine learning deals the clustering problem. Classification problems are used when the target variable is discrete and regression technique is used when the target variable is continuous. Our target variable sale price is continuous. So our problem is related to regression task. We used three machine learning algorithms Random Forest, Linear Regression and Regression Tree modeling. Linear Regression model is very famous for regression problem as and its ability to solve only regression tasks. Random Forest is ensemble algorithm that is combination of different Regression Tree modeling. Random Forest can handle classification as well as regression problem. Regression Tree modeling is able to handle the classification as well as regression problem. In Regression Tree modeling, over-fitting and under-fitting can be handled through the pruning method. The sale prices of house prediction model dataset is raw data. It contain multiple unnecessary attributes that are irrelevant categorical features. We extract only relevant features from the main dataset and build subset of new dataset that contain only important features. In the subset dataset, our target variable is sale_price of the house. we split the dataset into training and test with the ratio 3:7. Then the model is trained on the training data and predicted on the test data. The model performance is evaluated on the metrics used for regression problems like MSE, Residual, RMSE and R2.

Dataset Description: Dataset is extracted from MLSI website which provide the update row information of house sale prices based on observed variables. The dataset contain 55 column with 2230 instances. The dataset contain lot of categorical features that unnecessary. The dataset also contain the outlier and values in the form of others. The dataset contain the lot of missing values. Each attribute contain different types of invalid values. So handling such type of dataset is also a challenging task.

Featurization: This is important step to be performed on the attribute before passing values to machine learning model. The mostly people do mistake in conversion of attributes. They just convert the attribute direct into numeric form in which original value of attribute does not remain same. We adopt the different strategy for each attribute so that the original information of attribute does not loss.

1. We removed the dollar sign from the attributes and does not directly convert into numeric form.
2. After removing the dollar sign the attribute contain the character values, we then used the numeric function to convert into double value based function
3. The categorical based function are directly converted into numeric form as if we performed the as.charcter function before as.numeric function, all the attribute loss their values and attribute only contain the missing values.
4. Some attributes were actually as factor but they were represented as numeric. So convert their data type from numeric to factor like num_floor_in_building, num_bedrooms etc.

Errors and Miss values: We performed two method for handling the missing values.

1. In first method we observed that attribute who has small number of missing values, we removed the records on the bases of those attributes.
2. In second strategy, The attribute that contain large number of missing values, we used mean method to maintain their values. For the validation checking of the dataset we used the sapply function to checking the missing values and summary function to check the correct values in the dataset.

Modeling: We used three algorithm in the proposed project. Regression Tree modeling, Linear Regression model and Random Forest. These algorithms have ability to handle the classification as well as regression problem. First we split the dataset into two parts training and test part. The training size is 70 while test is 30. We then trained the model on training data and evaluate the result on test data. We also get information about the importance feature of each algorithms. Then we passed these importance features to the model as and trained on these feature and shows the good result. Most of the machine learning applications in real estate price estimation are based on Artificial Neural Networks (ANN) algorithms. Random forest used for classification and regression based on a forest of trees using random inputs, caret for data splitting and generating stratified bootstrap samples, gstat for cross validation, psych for principal component analysis. Regression Tree modeling also have ability to the classification and regression problems. Regression Tree modeling perform well for sale pice os the houses. Used the Regression Tree modeling technique for exploring the relationship between house prices and housing characteristics, which aided the determination of the most important variables of housing prices and predicted housing prices.

Regression Tree model: This model has the ability to regression problem of sale_price prediction based on num_total_rooms, num_full_bathrooms, num_floors_in_building, num_bedrooms. The variable importance plot is drawn as shown in code through barplot and shown which features are important for predicting the model.

Linear Regression Model: Linear regression has just ability to solve the the regression problem. It is very famous for solving regression tasks. In this project linear regression model is training on the following features kitchen_type, num_floors_in_building, num_bedrooms, coop_condo, total_taxes. The OLS summary of Linear regression model shows that model is fitted well over the training. The residual error is 77100 that is very good and have low loss function and P value is also less than five. Our algorithm shows that it rejects the null hypothesis. The R-Squared error 0.2333 that shows the model have good accuracy. F-statistics of proposed model is also high that 17.54. The very importance feature model shows is cope_condo and num-bedrooms.

Random Forest Model: It is ensemble based algorithm that is made from different algorithms. It can handle regression and classification tasks efficiently. Random forest is parametric model whole number of estimates varies from 10 to 100. The model shows the highest R squired value that shows that model is very fitted over the training data and test data. The model does not overfitting and under-fitting problem. The random forest model show the importance feature is Kitchen type thats value is above 4 and has major parts in predicting the random forest.

Random Forest Results: R2 value shows that our model have very good accuracy and fitted well on train and test data. MSE is mean squired error that how many times our algorithm provide the wrong results while its actual result was different. The rout mean squared error is just square of MSE. We used default number of parameter 100 so that iteration should take place automatically.

MSE: 4400174.

RMSE: 6633381.

R2: 0.4324472.

Conclusion: From this we get knowledge that how to handle large row dataset, how to features the attributes and how handle the model. We also get knowledge that machine learning are how to be trained on regression problem. we get more information about the importance feature of algorithms and shows their results visualization graphs. Our algorithm outperformed on such raw dataset and get insight knowledge from dataset. Random Forest performance is better than decision tree for prediction of sale prices.

Acknowledgments: I learned most of the stuffs from Professors lecture and it was not possible if all the lectures video were not available at slack. Once a time I was lost but most of the lecture videos were available at slack. So, it helped me to go through all the videos and understand the material. But also I want to give credit to my friend to help me finish this prediction model on Machine Learning Algorithms. He helped me with some great resources to understand this machine learning material more.

References:

[1]. Chiarazzo, V.; Caggiani, L.; Marinelli, M.; Ottomanelli, M. A Neural Network based model for real estate price estimation considering environmental quality of property location. Transp. Res. Proc. 2014, 3, 810–817.[CrossRef]

[2]. Yalpir, S.; Durduran, S.S.; Unel, F.B.; Yolcu, M. Creating a Valuation Map in GIS Through Artificial Neural Network Methodology: A Case Study. Acta Montan. Slovaca 2014, 19, 89–99.

[3].`Liaw, A.; Wiener, M. Classification and regression by random Forest. R News 2002, 2, 18–22.ISPRS Int. J. Geo-Inf. 2018, 7, 168.

[4]. Kuhn, M. Caret package. J. Stat. Softw. 2008, 28, 1–16.

[5]. Fan, G.Z.; Ong, S.E.; Koh, H.C. Determinants of house price: A decision tree approach. Urban. Stud. 2006, 43, 2301–2315. [CrossRef]

Project Code

####Libraries used in this project

```
library(ggplot2)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

getwd()

## [1] "/Users/sakif/Desktop/Final Project"
```

#Importing the dataset

```
df<-read.csv("housing_data_2016_2017.csv", na.strings = c("") )
```

#Checking shape of the dataset

```
dim(df)

## [1] 2230   55
```

#Checking name of the columns in dataset

```
names(df)

##  [1] "HITId"                          "HITTypeId"
##  [3] "Title"                          "Description"
##  [5] "Keywords"                       "Reward"
##  [7] "CreationTime"                   "MaxAssignments"
##  [9] "RequesterAnnotation"            "AssignmentDurationInSeconds"
## [11] "AutoApprovalDelayInSeconds"     "Expiration"
## [13] "NumberOfSimilarHITs"            "LifetimeInSeconds"
```

```
## [15] "AssignmentId"                     "WorkerId"
## [17] "AssignmentStatus"                 "AcceptTime"
## [19] "SubmitTime"                       "AutoApprovalTime"
## [21] "ApprovalTime"                     "RejectionTime"
## [23] "RequesterFeedback"                "WorkTimeInSeconds"
## [25] "LifetimeApprovalRate"             "Last30DaysApprovalRate"
## [27] "Last7DaysApprovalRate"            "URL"
## [29] "approx_year_built"                "cats_allowed"
## [31] "common_charges"                   "community_district_num"
## [33] "coop_condo"                       "date_of_sale"
## [35] "dining_room_type"                 "dogs_allowed"
## [37] "fuel_type"                        "full_address_or_zip_code"
## [39] "garage_exists"                    "kitchen_type"
## [41] "maintenance_cost"                 "model_type"
## [43] "num_bedrooms"                     "num_floors_in_building"
## [45] "num_full_bathrooms"               "num_half_bathrooms"
## [47] "num_total_rooms"                  "parking_charges"
## [49] "pct_tax_deductibl"                "sale_price"
## [51] "sq_footage"                       "total_taxes"
## [53] "walk_score"
"listing_price_to_nearest_1000"
## [55] "url"
```

#Extracting the columns names that are useful for predictive model

```
mdf = names(df) %in%
c("WorkTimeInSeconds","approx_year_built","community_district_num","co
op_condo", "fuel_type",
                      "kitchen_type", "maintenance_cost" ,
"num_bedrooms" , "num_floors_in_building",
                      "num_full_bathrooms","num_total_rooms",
                     "sale_price", "sq_footage", "total_taxes",
"walk_score" )  #
```

#Making new data-frame from old one

```
data = df[mdf] #include the above columns
```

#Checking new data

```
head(data)
```

```
##   WorkTimeInSeconds approx_year_built community_district_num
coop_condo
## 1               181              1955                     25
co-op
## 2               121              1955                     25
co-op
## 3               120              2004                     24
condo
## 4               160              2002                     25
condo
## 5               136              1949                     26
co-op
## 6               249              1938                     28
co-op
##   fuel_type kitchen_type maintenance_cost num_bedrooms
num_floors_in_building
## 1       gas       eat in               NA            2
6
## 2       oil       eat in            $604             1
7
## 3        NA   efficiency               NA            1
1
## 4       gas       eat in               NA            3
NA
## 5       gas       eat in            $660             2
2
## 6       oil       eat in            $932             2
6
##   num_full_bathrooms num_total_rooms sale_price sq_footage
total_taxes
## 1                  1               5  $228,000         NA
NA
## 2                  1               4  $235,500        890
NA
## 3                  1               3  $137,550        550
$5,500
## 4                  2               5  $545,000         NA
$2,260
## 5                  1               4  $241,700        675
NA
## 6                  1               4  $250,000       1000
NA
```

```
##   walk_score
## 1         82
## 2         89
## 3         90
## 4         94
## 5         71
## 6         90
```

#Removing $ from the attributes values of dataset

```
data$maintenance_cost = gsub("[\\$,]", "", data$maintenance_cost)
data$sale_price = gsub("[\\$,]", "", data$sale_price)
data$total_taxes = gsub("[\\$,]", "", data$total_taxes)


head(data)
```

```
##   WorkTimeInSeconds approx_year_built community_district_num
coop_condo
## 1               181              1955                     25
co-op
## 2               121              1955                     25
co-op
## 3               120              2004                     24
condo
## 4               160              2002                     25
condo
## 5               136              1949                     26
co-op
## 6               249              1938                     28
co-op
##   fuel_type kitchen_type maintenance_cost num_bedrooms
num_floors_in_building
## 1       gas       eat in               NA            2
6
## 2       oil       eat in             604            1
7
## 3        NA   efficiency               NA            1
1
## 4       gas       eat in               NA            3
NA
## 5       gas       eat in             660            2
```

```
2
## 6       oil       eat in             932             2
6
##   num_full_bathrooms num_total_rooms sale_price sq_footage
total_taxes
## 1                  1               5    228000          NA
NA
## 2                  1               4    235500         890
NA
## 3                  1               3    137550         550
5500
## 4                  2               5    545000          NA
2260
## 5                  1               4    241700         675
NA
## 6                  1               4    250000        1000
NA
##   walk_score
## 1         82
## 2         89
## 3         90
## 4         94
## 5         71
## 6         90
```

#Changing the data types of categorical variables

```
data$maintenance_cost = as.numeric(data$maintenance_cost)

## Warning: NAs introduced by coercion

data$sale_price = as.numeric(data$sale_price)

## Warning: NAs introduced by coercion

data$total_taxes = as.numeric(data$total_taxes)

## Warning: NAs introduced by coercion

data$WorkTimeInSeconds =
as.numeric(as.character(data$WorkTimeInSeconds))

## Warning: NAs introduced by coercion
```

```
head(data)

##   WorkTimeInSeconds approx_year_built community_district_num
coop_condo
## 1               181              1955                     25
co-op
## 2               121              1955                     25
co-op
## 3               120              2004                     24
condo
## 4               160              2002                     25
condo
## 5               136              1949                     26
co-op
## 6               249              1938                     28
co-op
##   fuel_type kitchen_type maintenance_cost num_bedrooms
num_floors_in_building
## 1       gas       eat in               NA            2
6
## 2       oil       eat in              604            1
7
## 3        NA   efficiency               NA            1
1
## 4       gas       eat in               NA            3
NA
## 5       gas       eat in              660            2
2
## 6       oil       eat in              932            2
6
##   num_full_bathrooms num_total_rooms sale_price sq_footage
total_taxes
## 1                  1               5     228000         NA
NA
## 2                  1               4     235500        890
NA
## 3                  1               3     137550        550
5500
## 4                  2               5     545000         NA
2260
## 5                  1               4     241700        675
NA
```

```
## 6                  1               4     250000       1000
NA
##   walk_score
## 1         82
## 2         89
## 3         90
## 4         94
## 5         71
## 6         90
```

#Converting data types of attributes according to their relevant formation.

```r
data$coop_condo <- as.numeric(data$coop_condo)
data$fuel_type <- as.numeric(data$fuel_type)
data$kitchen_type <- as.numeric(data$kitchen_type)
data$approx_year_built <- factor(data$approx_year_built)
data$num_bedrooms <- factor(data$num_bedrooms)
data$num_floors_in_building <- factor(data$num_floors_in_building)
data$num_full_bathrooms <- factor(data$num_full_bathrooms)
data$num_total_rooms <- factor(data$num_total_rooms)
data$sq_footage = as.numeric(data$sq_footage)
```

#Checking again the missing values

```r
sapply(data, function(x) sum(is.na(x)))

##      WorkTimeInSeconds      approx_year_built
community_district_num
##                    758                      0
0
##             coop_condo              fuel_type
kitchen_type
##                      0                      0
0
##       maintenance_cost           num_bedrooms
num_floors_in_building
##                    623                      0
0
##     num_full_bathrooms        num_total_rooms
sale_price
##                      0                      0
1702
##             sq_footage            total_taxes
```

```
walk_score
##                        0                    1646
0
```

#Droping the records that contain very low amount of missing values

```
data = data[!is.na(data$approx_year_built),]
data = data[!is.na(data$fuel_type),]
data = data[!is.na(data$kitchen_type),]
data = data[!is.na(data$num_total_rooms), ]
data = data[!is.na(data$num_bedrooms),]
data = data[!is.na(data$community_district_num),]

sapply(data, function(x) sum(is.na(x)))
```

```
##      WorkTimeInSeconds      approx_year_built
community_district_num
##                      758                      0
0
##             coop_condo              fuel_type
kitchen_type
##                        0                      0
0
##       maintenance_cost           num_bedrooms
num_floors_in_building
##                      623                      0
0
##     num_full_bathrooms        num_total_rooms
sale_price
##                        0                      0
1702
##             sq_footage            total_taxes
walk_score
##                        0                   1646
0
```

#Impute the missing values with mean

```
meanmaint = mean(data$maintenance_cost, na.rm = T)
data$maintenance_cost = ifelse(is.na(data$maintenance_cost),
meanmaint,data$maintenance_cost)

meansale = mean(data$sale_price, na.rm = T)
```

```
data$sale_price = ifelse(is.na(data$sale_price),
meansale,data$sale_price)

meantax = mean(data$total_taxes, na.rm = T)
data$total_taxes = ifelse(is.na(data$total_taxes),
meantax,data$total_taxes)

meanfootage = mean(data$sq_footage, na.rm = T)
data$sq_footage = ifelse(is.na(data$sq_footage),
meanfootage,data$sq_footage)

meanwork = mean(data$WorkTimeInSeconds, na.rm = T)
data$WorkTimeInSeconds = ifelse(is.na(data$WorkTimeInSeconds),
meanwork,data$WorkTimeInSeconds)

sapply(data, function(x) sum(is.na(x)))

##      WorkTimeInSeconds      approx_year_built
community_district_num
##                      0                      0
0
##             coop_condo              fuel_type
kitchen_type
##                      0                      0
0
##       maintenance_cost           num_bedrooms
num_floors_in_building
##                      0                      0
0
##     num_full_bathrooms        num_total_rooms
sale_price
##                      0                      0
0
##             sq_footage            total_taxes
walk_score
##                      0                      0
0
```

#Impute the missing values with mean

```
summary(data)
```

```
##  WorkTimeInSeconds approx_year_built community_district_num
coop_condo
##  Min.   : 22.0     1950   : 311      25     :616            Min.
:1.000
##  1st Qu.:106.0     1955   : 149      28     :583            1st
Qu.:1.000
##  Median :162.4     1960   : 136      26     :356            Median
:1.000
##  Mean   :162.4     1965   :  82      30     :226            Mean
:1.255
##  3rd Qu.:162.4     1952   :  80      24     :190            3rd
Qu.:2.000
##  Max.   :815.0     1964   :  63      27     :111            Max.
:2.000
##                    (Other):1409      (Other):148
##    fuel_type      kitchen_type    maintenance_cost  num_bedrooms
##  Min.   :1.000   Min.   : 1.000   Min.   : 155.0   1      :958
##  1st Qu.:2.000   1st Qu.: 4.000   1st Qu.: 676.0   2      :872
##  Median :2.000   Median : 7.000   Median : 858.9   3      :243
##  Mean   :2.992   Mean   : 6.967   Mean   : 858.9   NA     :115
##  3rd Qu.:5.000   3rd Qu.: 9.000   3rd Qu.: 880.0   0      : 28
##  Max.   :7.000   Max.   :14.000   Max.   :4659.0   4      :  6
##                                                    (Other):  8
##  num_floors_in_building num_full_bathrooms num_total_rooms
sale_price
##  NA     :650            1:1736             4      :668     Min.   :
55000
##  6      :492            2: 472             3      :618     1st
Qu.:314957
##  2      :284            3:  22             5      :495     Median
:314957
##  7      :142                               6      :223     Mean
:314957
##  3      :105                               2      :111     3rd
Qu.:314957
##  1      : 85                               7      : 60     Max.
:999999
##  (Other):472                               (Other): 55
##    sq_footage   total_taxes     walk_score
##  Min.   :  1   Min.   :  11   Min.   : 7.00
##  1st Qu.:131   1st Qu.:2226   1st Qu.:77.00
```

```
##  Median :231   Median :2226   Median :89.00
##  Mean   :177   Mean   :2226   Mean   :83.92
##  3rd Qu.:231   3rd Qu.:2226   3rd Qu.:95.00
##  Max.   :231   Max.   :9300   Max.   :99.00
##
```

#Creating Training and Test part

```
set.seed(1966)
trainIndex <- caret::createDataPartition(data$sale_price, p = 0.7,
list = FALSE)
train <- data[trainIndex, ]
test <- data[-trainIndex, ]
```

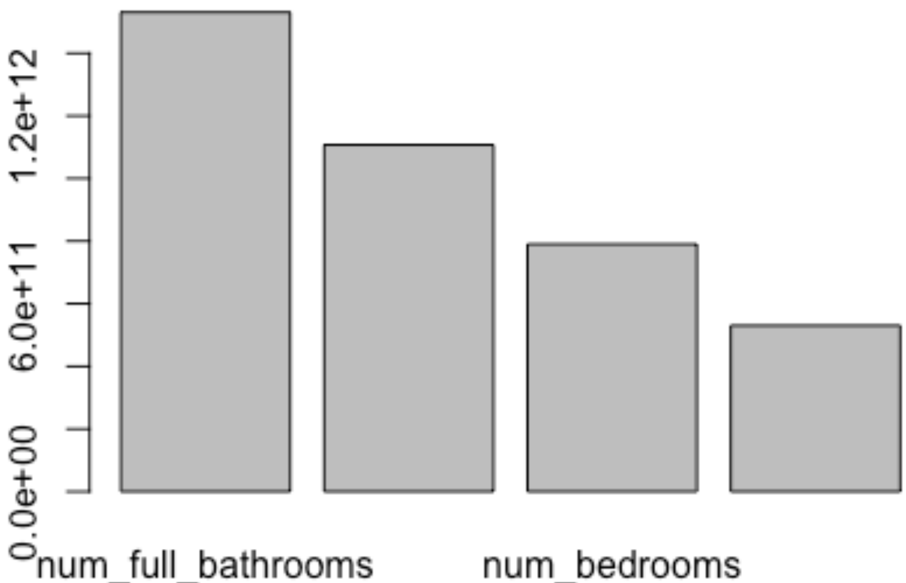
Regression Tree Model

#Regression Tree modeling

```
library(rpart)
dt <- rpart(data$sale_price ~ num_total_rooms + num_full_bathrooms +
num_floors_in_building + num_bedrooms, data = data)
```

#Importance variables

```
dt$variable.importance

##     num_full_bathrooms num_floors_in_building
num_bedrooms
##           1.530804e+12           1.107221e+12
7.901744e+11
##        num_total_rooms
##           5.303075e+11

barplot(dt$variable.importance)
```

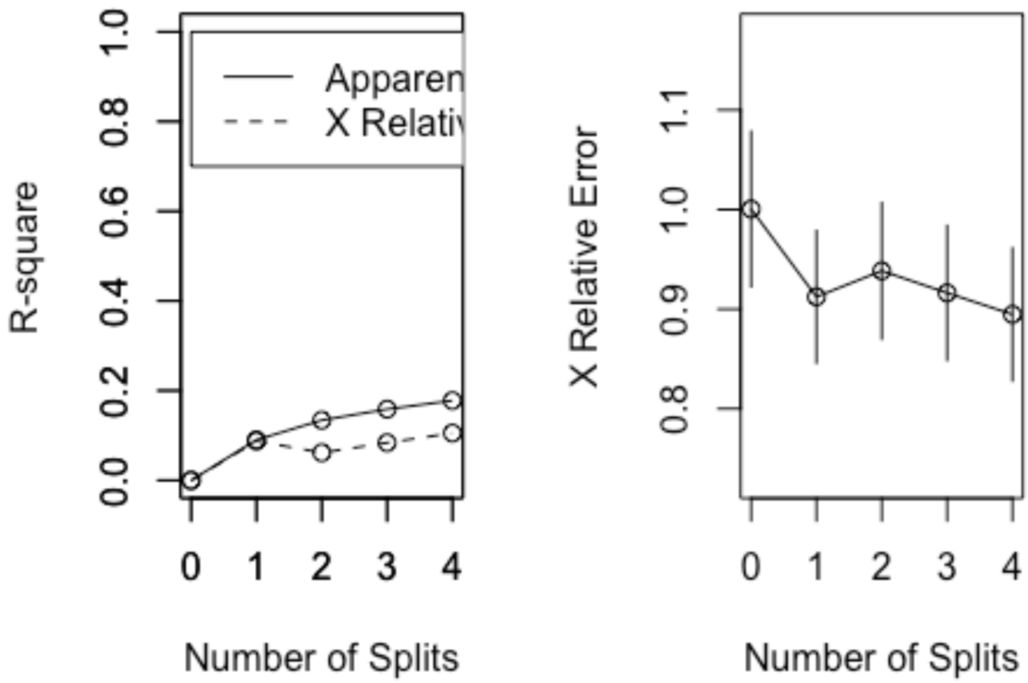


#Checking the train test split

```
par(mfrow = c(1,2))
rsq.rpart(dt)
```

```
##
## Regression tree:
## rpart(formula = data$sale_price ~ num_total_rooms +
num_full_bathrooms +
##     num_floors_in_building + num_bedrooms, data = data)
##
## Variables actually used in tree construction:
## [1] num_bedrooms           num_floors_in_building
num_full_bathrooms
##
## Root node error: 1.6985e+13/2230 = 7616638518
```

```
##
## n= 2230
##
##          CP nsplit rel error  xerror     xstd
## 1 0.090126      0   1.00000 1.00051 0.077978
## 2 0.043895      1   0.90987 0.91235 0.066416
## 3 0.024566      2   0.86598 0.93842 0.068363
## 4 0.019103      3   0.84141 0.91624 0.067573
## 5 0.010000      4   0.82231 0.89470 0.066623
```



#Summary of the Regression Tree model

```
summary(dt)

## Call:
## rpart(formula = data$sale_price ~ num_total_rooms +
```

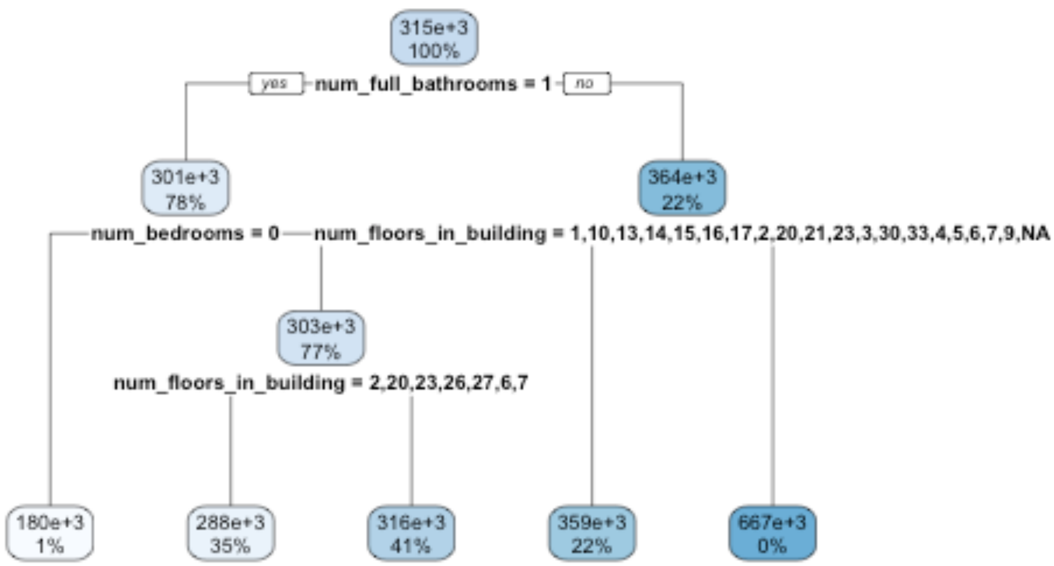
```
num_full_bathrooms +
##     num_floors_in_building + num_bedrooms, data = data)
##   n= 2230
##
##           CP nsplit rel error    xerror       xstd
## 1 0.09012626      0 1.0000000 1.0005144 0.07797753
## 2 0.04389520      1 0.9098737 0.9123493 0.06641579
## 3 0.02456553      2 0.8659785 0.9384169 0.06836283
## 4 0.01910327      3 0.8414130 0.9162434 0.06757267
## 5 0.01000000      4 0.8223097 0.8947033 0.06662334
##
## Variable importance
##     num_full_bathrooms num_floors_in_building
num_bedrooms
##                     39                     28
20
##        num_total_rooms
##                     13
##
## Node number 1: 2230 observations,    complexity param=0.09012626
##   mean=314956.6, MSE=7.616639e+09
##   left son=2 (1736 obs) right son=3 (494 obs)
##   Primary splits:
##       num_full_bathrooms     splits as  LRR, improve=0.09012626, (0
missing)
##       num_bedrooms           splits as  LLRRRRRR,
improve=0.05324203, (0 missing)
##       num_total_rooms        splits as  LLLLLLLLRRRRLL,
improve=0.05058150, (0 missing)
##       num_floors_in_building splits as
LRLLRRLLLLRRRRLLLRRRRRRRLLRLL, improve=0.04491708, (0 missing)
##   Surrogate splits:
##       num_total_rooms        splits as  LLRRRLLLLRRRRL,
agree=0.855, adj=0.346, (0 split)
##       num_bedrooms           splits as  LLLRRRRL, agree=0.829,
adj=0.229, (0 split)
##       num_floors_in_building splits as
LLLLLLLLLLRRRRLLLLRLLLLLLLLLL, agree=0.784, adj=0.024, (0 split)
##
## Node number 2: 1736 observations,    complexity param=0.02456553
##   mean=300980.1, MSE=4.861356e+09
```

```
##   left son=4 (28 obs) right son=5 (1708 obs)
##   Primary splits:
##       num_bedrooms           splits as  LRRRR--R,
improve=0.04944100, (0 missing)
##       num_floors_in_building splits as
RRRRRRLRRLLRRRRLLRRRRRRRLLRRR, improve=0.04134875, (0 missing)
##       num_total_rooms        splits as  RL---LLRRRRR-R,
improve=0.01993979, (0 missing)
##
## Node number 3: 494 observations,    complexity param=0.0438952
##   mean=364072.1, MSE=1.420038e+10
##   left son=6 (486 obs) right son=7 (8 obs)
##   Primary splits:
##       num_floors_in_building splits as  LL--LLLLLLLLRL---
RLLLRLLLLRLL, improve=0.106281700, (0 missing)
##       num_bedrooms           splits as  -LLRLLLL,
improve=0.008171751, (0 missing)
##       num_full_bathrooms     splits as  -LR, improve=0.005187827,
(0 missing)
##       num_total_rooms        splits as  L-LLL-RLLLRLL-,
improve=0.003657032, (0 missing)
##
## Node number 4: 28 observations
##   mean=179896, MSE=1.018865e+10
##
## Node number 5: 1708 observations,    complexity param=0.01910327
##   mean=302965.1, MSE=4.529733e+09
##   left son=10 (784 obs) right son=11 (924 obs)
##   Primary splits:
##       num_floors_in_building splits as
RRRRRRRRRLLRRLRLLRRRRRRRLLRRR, improve=0.04193875, (0 missing)
##       num_bedrooms           splits as  -LRRR--R,
improve=0.01667167, (0 missing)
##       num_total_rooms        splits as  RR---RLRRRRR-R,
improve=0.01658317, (0 missing)
##   Surrogate splits:
##       num_bedrooms    splits as  -RLRL--R, agree=0.573, adj=0.070,
(0 split)
##       num_total_rooms splits as  RR---RRRRLRR-R, agree=0.542,
adj=0.001, (0 split)
##
```

```
## Node number 6: 486 observations
##   mean=359087.8, MSE=1.253414e+10
##
## Node number 7: 8 observations
##   mean=666869.6, MSE=2.222843e+10
##
## Node number 10: 784 observations
##   mean=288002, MSE=4.203757e+09
##
## Node number 11: 924 observations
##   mean=315661.1, MSE=4.455159e+09
```

#Regression Tree model diagram

```
library(rpart.plot)
rpart.plot(dt)
```



#Model Prediction

```
fittedvalues = predict(dt, test)
```

#Now we can get the root mean squared error, a standardized measure of how off we were with our predicted values

```
results <- cbind(fittedvalues, test$sale_price)
colnames(results) <- c('pred','real')
results <- as.data.frame(results)
```

#Now let's take care of negative predictions! Lot's of ways to this, here's a more complicated way, but its a good example of creating a custom function for a custom problem:

```
to_zero <- function(x){
    if  (x < 0){
        return(0)
    }else{
        return(x)
    }
}
results$pred <- sapply(results$pred,to_zero)
```

#MSE (mean squared error):

```
mse <- mean((results$real - results$pred) ^ 2)
print(mse)

## [1] 6102260805
```

#Root mean squared error

```
mse ^ 0.5

## [1] 78116.97
```

#Just the R-Squared Value for our model (just for the predictions)

```
SSE = sum((results$pred - results$real) ^ 2)
SST = sum((mean(data$sale_price) - results$real) ^ 2)

R2 = 1 - SSE/SST
R2
```

```
## [1] 0.2129051
```

```
Linear Regression Model
```

#Linear Regresson Model

```
lr = lm(data$sale_price ~ kitchen_type + num_floors_in_building +
num_bedrooms + coop_condo + total_taxes, data = data)
```

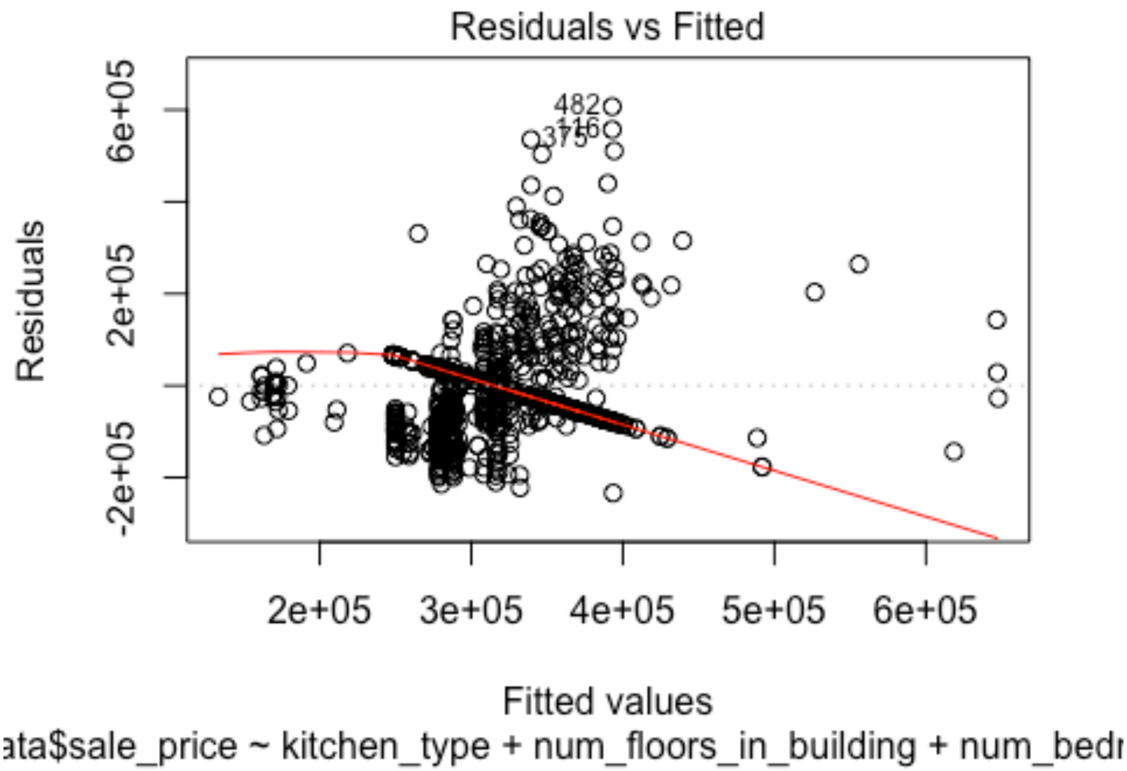
#Summary of the model (OLS)

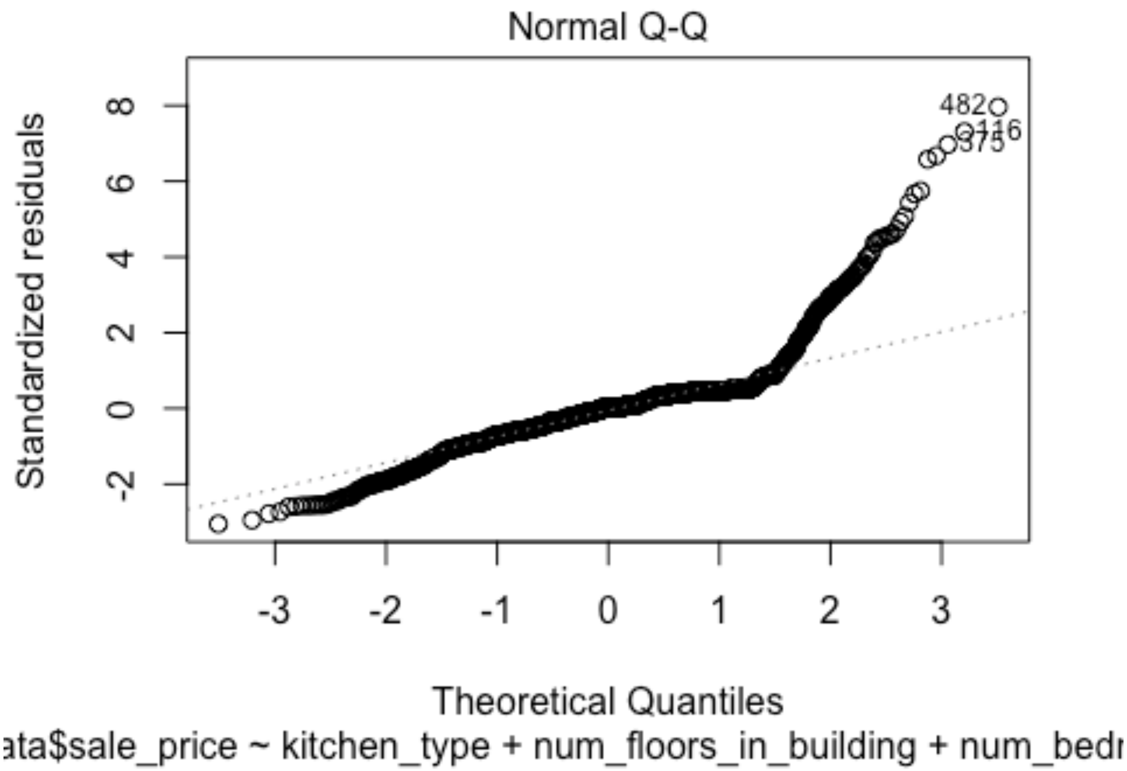
```
summary(lr)

##
## Call:
## lm(formula = data$sale_price ~ kitchen_type +
num_floors_in_building +
##     num_bedrooms + coop_condo + total_taxes, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -233665  -39790    1663   31840  607003
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               1.396e+05  1.857e+04   7.516 8.19e-14 ***
## kitchen_type             -1.609e+03  5.836e+02  -2.757 0.005888 **
## num_floors_in_building10  5.716e+04  3.551e+04   1.609 0.107659
## num_floors_in_building11 -4.581e+04  4.543e+04  -1.008 0.313359
## num_floors_in_building12 -1.282e+04  2.105e+04  -0.609 0.542547
## num_floors_in_building13  6.161e+03  2.410e+04   0.256 0.798241
## num_floors_in_building14  3.102e+04  1.960e+04   1.582 0.113763
## num_floors_in_building15  1.323e+03  1.475e+04   0.090 0.928501
## num_floors_in_building16  4.251e+03  2.105e+04   0.202 0.839989
## num_floors_in_building17  2.075e+04  1.362e+04   1.524 0.127729
## num_floors_in_building2  -3.747e+04  9.580e+03  -3.911 9.45e-05 ***
## num_floors_in_building20  8.884e+03  2.307e+04   0.385 0.700178
## num_floors_in_building21  4.072e+04  1.764e+04   2.309 0.021045 *
## num_floors_in_building22  2.771e+05  3.996e+04   6.934 5.35e-12 ***
```

```
## num_floors_in_building23  6.731e+04  3.572e+04   1.884 0.059653 .
## num_floors_in_building25  1.137e+04  3.557e+04   0.320 0.749202
## num_floors_in_building26 -1.181e+05  7.762e+04  -1.522 0.128191
## num_floors_in_building27 -1.631e+04  3.050e+04  -0.535 0.592919
## num_floors_in_building29  9.301e+04  3.553e+04   2.618 0.008919 **
## num_floors_in_building3  -9.598e+03  1.171e+04  -0.819 0.412658
## num_floors_in_building30  6.699e+04  2.594e+04   2.582 0.009886 **
## num_floors_in_building33  3.844e+04  1.307e+04   2.942 0.003290 **
## num_floors_in_building34  2.012e+05  3.551e+04   5.666 1.66e-08 ***
## num_floors_in_building4  -1.465e+04  1.348e+04  -1.087 0.277253
## num_floors_in_building5   1.684e+04  1.406e+04   1.198 0.231159
## num_floors_in_building6  -9.123e+03  9.088e+03  -1.004 0.315569
## num_floors_in_building7  -8.788e+03  1.059e+04  -0.830 0.406824
## num_floors_in_building8   3.667e+04  1.828e+04   2.006 0.045006 *
## num_floors_in_building9   3.966e+02  1.486e+04   0.027 0.978702
## num_floors_in_buildingNA -7.294e+03  8.913e+03  -0.818 0.413211
## num_bedrooms1             1.169e+05  1.494e+04   7.823 7.94e-15 ***
## num_bedrooms2             1.468e+05  1.499e+04   9.792  < 2e-16 ***
## num_bedrooms3             1.759e+05  1.562e+04  11.266  < 2e-16 ***
## num_bedrooms4             1.380e+05  3.497e+04   3.948 8.13e-05 ***
## num_bedrooms5             1.224e+05  3.581e+04   3.418 0.000643 ***
## num_bedrooms6             1.056e+05  5.721e+04   1.847 0.064922 .
## num_bedroomsNA            1.367e+05  1.643e+04   8.319  < 2e-16 ***
## coop_condo                4.758e+04  4.304e+03  11.056  < 2e-16 ***
## total_taxes              -9.419e-01  1.856e+00  -0.507 0.611955
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 77100 on 2191 degrees of freedom
## Multiple R-squared:  0.2333, Adjusted R-squared:   0.22
## F-statistic: 17.54 on 38 and 2191 DF,  p-value: < 2.2e-16
```

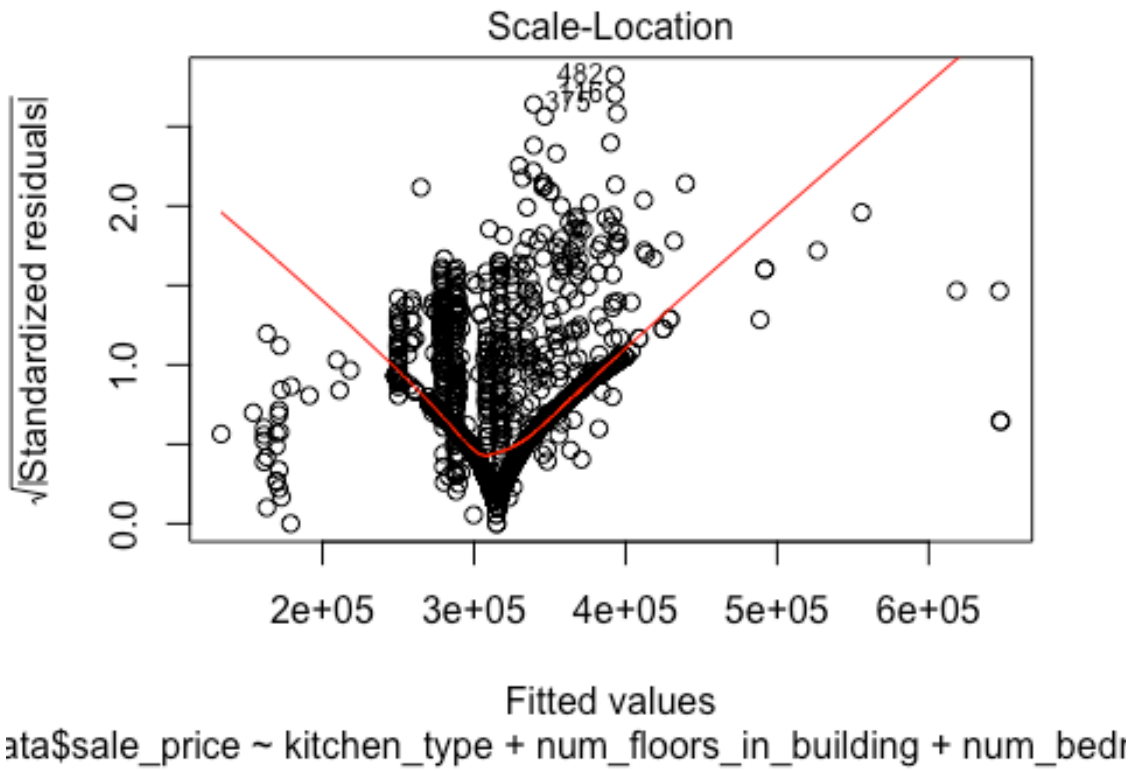
#Predicting the Random Forest Model

```
plot(lr)
```

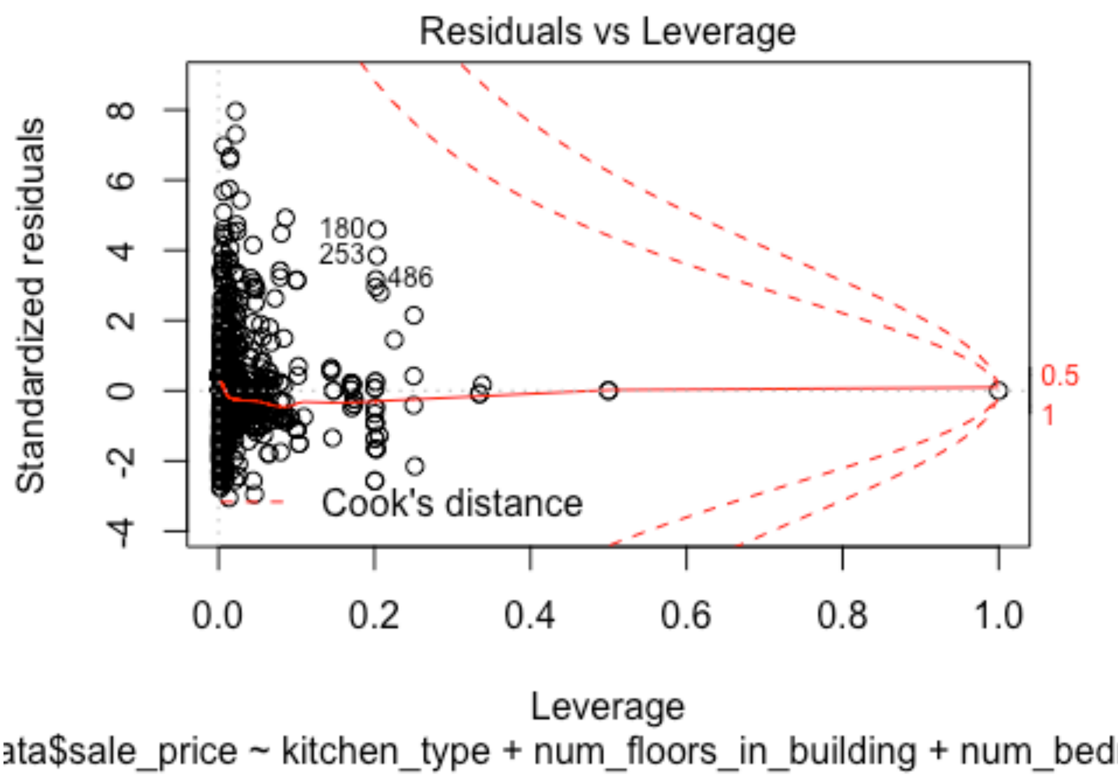

Residuals vs Fitted
482
116
375
6e+05
2e+05
-2e+05
Residuals
2e+05
3e+05
4e+05
5e+05
6e+05
Fitted values
ata$sale_price ~ kitchen_type + num_floors_in_building + num_bedr



Normal Q-Q
Standardized residuals
8
6
4
2
0
-2
482
116
375
-3
-2
-1
0
1
2
3
Theoretical Quantiles
ata$sale_price ~ kitchen_type + num_floors_in_building + num_bedr




```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced

## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```



Random Forest Model

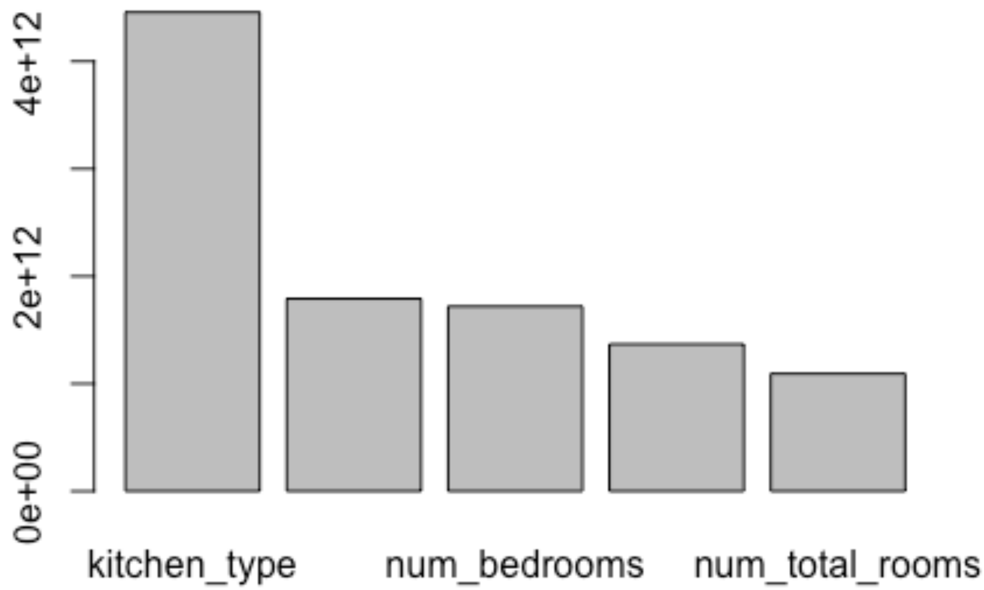
#Fitting the Random Forest model

```
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'
```

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

## The following object is masked from 'package:ggplot2':
##
##     margin

library(rpart)
tree_fit <- rpart::rpart(data$sale_price ~ kitchen_type +
num_floors_in_building + num_bedrooms + coop_condo + num_total_rooms,
data = data)
```

#Checking the importance of variable

```
tree_fit$variable.importance

##           kitchen_type num_floors_in_building
num_bedrooms
##           4.456851e+12           1.790879e+12
1.719627e+12
##             coop_condo        num_total_rooms
##           1.368057e+12           1.093195e+12

barplot(tree_fit$variable.importance)
```



#Summary of the Random Forest Model

```
summary(tree_fit)

## Call:
## rpart::rpart(formula = data$sale_price ~ kitchen_type +
num_floors_in_building +
##     num_bedrooms + coop_condo + num_total_rooms, data = data)
##   n= 2230
##
##           CP nsplit rel error    xerror       xstd
## 1 0.08326561      0 1.0000000 1.0005240 0.07798219
## 2 0.03939569      3 0.7502032 0.7527325 0.06496202
## 3 0.02775556      4 0.7108075 0.7283511 0.06151385
## 4 0.02447076      6 0.6552964 0.7486798 0.06265108
## 5 0.01961643      9 0.5818841 0.7045988 0.06269327
```

```
## 6 0.01719795     11 0.5426512 0.6817598 0.06272763
## 7 0.01177415     12 0.5254533 0.6670817 0.05667872
## 8 0.01000000     13 0.5136791 0.6347481 0.05267817
##
## Variable importance
##           kitchen_type num_floors_in_building
num_bedrooms
##                     43                     17
16
##             coop_condo        num_total_rooms
##                     13                     10
##
## Node number 1: 2230 observations,    complexity param=0.08326561
##   mean=314956.6, MSE=7.616639e+09
##   left son=2 (1661 obs) right son=3 (569 obs)
##   Primary splits:
##       coop_condo             < 1.5  to the left,
improve=0.08054450, (0 missing)
##       num_bedrooms           splits as  LLRRRRRR,
improve=0.05324203, (0 missing)
##       num_total_rooms        splits as  LLLLLLLLRRRRLL,
improve=0.05058150, (0 missing)
##       num_floors_in_building splits as
LRLLRRLLLLRRRRLLLRRRRRRRLLRLL, improve=0.04491708, (0 missing)
##       kitchen_type           < 6.5  to the right,
improve=0.01697508, (0 missing)
##   Surrogate splits:
##       num_floors_in_building splits as
LLRLRLLLLLLLRRLLLLRLLLRRLLLLL, agree=0.796, adj=0.200, (0 split)
##       num_bedrooms           splits as  LLLLLRRL, agree=0.748,
adj=0.014, (0 split)
##       num_total_rooms        splits as  LLRRRLLLLLLLRL,
agree=0.748, adj=0.012, (0 split)
##       kitchen_type           < 13.5 to the left,  agree=0.747,
adj=0.009, (0 split)
##
## Node number 2: 1661 observations,    complexity param=0.03939569
##   mean=300459.8, MSE=5.648825e+09
##   left son=4 (790 obs) right son=5 (871 obs)
##   Primary splits:
##       num_bedrooms           splits as  LLRRR--R,
```

```
improve=0.07131635, (0 missing)
##       num_total_rooms        splits as  RL---LLLRRRR-R,
improve=0.05900775, (0 missing)
##       num_floors_in_building splits as  RRRL-RLRRLLL-
LLLLRLRRRRRLLLRL, improve=0.04000199, (0 missing)
##       kitchen_type           < 9.5  to the left,
improve=0.01195874, (0 missing)
##   Surrogate splits:
##       num_total_rooms        splits as  RL---RLRRRRR-R,
agree=0.794, adj=0.567, (0 split)
##       num_floors_in_building splits as  RRRR-RRLLRLR-
LLLLRRRLLLRLRLLR, agree=0.583, adj=0.123, (0 split)
##       kitchen_type           < 7.5  to the right, agree=0.559,
adj=0.072, (0 split)
##
## Node number 3: 569 observations,    complexity param=0.08326561
##   mean=357274.9, MSE=1.095668e+10
##   left son=6 (360 obs) right son=7 (209 obs)
##   Primary splits:
##       kitchen_type           < 6.5  to the right,
improve=0.11647690, (0 missing)
##       num_floors_in_building splits as  LRLLLLLL-LLRRL--L-L---
LLLLLLL, improve=0.08586306, (0 missing)
##       num_bedrooms           splits as  LLRRLLLL,
improve=0.01848383, (0 missing)
##       num_total_rooms        splits as  L-LLLLLLLLRLL-,
improve=0.01832502, (0 missing)
##   Surrogate splits:
##       num_floors_in_building splits as  LLLRLLLR-RLRRL--L-L---
LLLLLLL, agree=0.650, adj=0.048, (0 split)
##       num_total_rooms        splits as  L-LLLLLLLLRLL-,
agree=0.636, adj=0.010, (0 split)
##
## Node number 4: 790 observations,    complexity param=0.02447076
##   mean=279384.7, MSE=4.726681e+09
##   left son=8 (24 obs) right son=9 (766 obs)
##   Primary splits:
##       num_bedrooms           splits as  LR------,
improve=0.10830260, (0 missing)
##       kitchen_type           < 9.5  to the left,
improve=0.09517635, (0 missing)
```

```
##       num_floors_in_building splits as  RR-L-RLRRLLR-
RRLLRRRRRRRRRRRR, improve=0.07321018, (0 missing)
##       num_total_rooms        splits as  RR---LRRRR----,
improve=0.04036530, (0 missing)
##
## Node number 5: 871 observations,    complexity param=0.02775556
##   mean=319575, MSE=5.716971e+09
##   left son=10 (818 obs) right son=11 (53 obs)
##   Primary splits:
##       num_floors_in_building splits as  LLLL-RLLLLRL-----
RLRRRRLLLLLL, improve=0.077150540, (0 missing)
##       num_bedrooms           splits as  --LRL--L,
improve=0.047875650, (0 missing)
##       num_total_rooms        splits as  LL---LLLLRRR-L,
improve=0.027171150, (0 missing)
##       kitchen_type           < 6.5  to the right,
improve=0.006966599, (0 missing)
##
## Node number 6: 360 observations
##   mean=330055.3, MSE=4.416051e+09
##
## Node number 7: 209 observations,    complexity param=0.08326561
##   mean=404160.2, MSE=1.874839e+10
##   left son=14 (130 obs) right son=15 (79 obs)
##   Primary splits:
##       kitchen_type           < 2.5  to the left,
improve=0.54833720, (0 missing)
##       num_floors_in_building splits as  L--LLL-L-L-RR-----L---
LRLLRLL, improve=0.18143280, (0 missing)
##       num_total_rooms        splits as  -----LLLLLRL--,
improve=0.03677242, (0 missing)
##       num_bedrooms           splits as  RLLR-L-L,
improve=0.01082739, (0 missing)
##   Surrogate splits:
##       num_floors_in_building splits as  L--LLR-L-L-RR-----R---
LRLLRLL, agree=0.713, adj=0.241, (0 split)
##       num_bedrooms           splits as  RLLL-L-L, agree=0.632,
adj=0.025, (0 split)
##
## Node number 8: 24 observations
##   mean=151562.5, MSE=1.516923e+09
```

```
##
## Node number 9: 766 observations,    complexity param=0.02447076
##   mean=283389.6, MSE=4.299297e+09
##   left son=18 (546 obs) right son=19 (220 obs)
##   Primary splits:
##       kitchen_type           < 9.5  to the left,
improve=0.08511195, (0 missing)
##       num_floors_in_building splits as  RR-L-RRRRLRR-
RRLRRRRRRRRRRRRR, improve=0.07652537, (0 missing)
##       num_total_rooms        splits as  RR---LLRRL----,
improve=0.00700507, (0 missing)
##   Surrogate splits:
##       num_floors_in_building splits as  LL-L-LLRLLLL-
LLLLRLLLLLLLLRLL, agree=0.725, adj=0.041, (0 split)
##       num_total_rooms        splits as  RL---LLLLL----,
agree=0.715, adj=0.009, (0 split)
##
## Node number 10: 818 observations
##   mean=314229.2, MSE=3.674325e+09
##
## Node number 11: 53 observations,    complexity param=0.02775556
##   mean=402082, MSE=2.99946e+10
##   left son=22 (44 obs) right son=23 (9 obs)
##   Primary splits:
##       num_total_rooms        splits as  ------LLLRR---,
improve=0.3514422, (0 missing)
##       num_bedrooms           splits as  --LR---L,
improve=0.2966391, (0 missing)
##       kitchen_type           < 5.5  to the right,
improve=0.1368355, (0 missing)
##       num_floors_in_building splits as  -----L----L------R-
LLRR------, improve=0.1001989, (0 missing)
##   Surrogate splits:
##       num_bedrooms splits as  --LR---L, agree=0.943, adj=0.667, (0
split)
##
## Node number 14: 130 observations
##   mean=325120.1, MSE=2.343701e+09
##
## Node number 15: 79 observations,    complexity param=0.01177415
##   mean=534226.3, MSE=1.854582e+10
```

```
##   left son=30 (31 obs) right son=31 (48 obs)
##   Primary splits:
##       num_bedrooms           splits as  LLRR----,
improve=0.136497400, (0 missing)
##       num_total_rooms        splits as  -----LLLLLR---,
improve=0.121686900, (0 missing)
##       num_floors_in_building splits as  L----L---L-RR-----R---
LRLLRLR, improve=0.090553720, (0 missing)
##       kitchen_type           < 5    to the right,
improve=0.002359503, (0 missing)
##   Surrogate splits:
##       num_total_rooms        splits as  -----LLRRRR---,
agree=0.810, adj=0.516, (0 split)
##       num_floors_in_building splits as  L----L---R-RR-----R---
RRLRLLR, agree=0.696, adj=0.226, (0 split)
##       kitchen_type           < 3.5  to the left,  agree=0.633,
adj=0.065, (0 split)
##
## Node number 18: 546 observations,    complexity param=0.02447076
##   mean=271247.1, MSE=5.425492e+09
##   left son=36 (154 obs) right son=37 (392 obs)
##   Primary splits:
##       kitchen_type           < 8.5  to the right,
improve=0.18978690, (0 missing)
##       num_floors_in_building splits as  RR-L-RLRRLRR-RRLR-
RRRRRRRRLRR, improve=0.11001080, (0 missing)
##       num_total_rooms        splits as  -R---LLRRL----,
improve=0.01277479, (0 missing)
##   Surrogate splits:
##       num_floors_in_building splits as  RR-L-RRRRLRR-RRRR-
LRRRRRRRLRR, agree=0.74, adj=0.078, (0 split)
##
## Node number 19: 220 observations
##   mean=313525.1, MSE=2.30213e+08
##
## Node number 22: 44 observations
##   mean=355647.3, MSE=9.579619e+09
##
## Node number 23: 9 observations
##   mean=629096.5, MSE=6.772422e+10
##
```

```
## Node number 30: 31 observations
##   mean=471619, MSE=1.820971e+10
##
## Node number 31: 48 observations
##   mean=574660.2, MSE=1.459654e+10
##
## Node number 36: 154 observations,    complexity param=0.01719795
##   mean=220051.2, MSE=7.625282e+09
##   left son=72 (137 obs) right son=73 (17 obs)
##   Primary splits:
##       num_floors_in_building splits as  R--L--L-RL------R-L-
RRLLLLLLL, improve=0.24875290, (0 missing)
##       num_total_rooms        splits as  -L---LLR------,
improve=0.02539205, (0 missing)
##
## Node number 37: 392 observations,    complexity param=0.01961643
##   mean=291359.8, MSE=3.127082e+09
##   left son=74 (169 obs) right son=75 (223 obs)
##   Primary splits:
##       kitchen_type           < 6.5  to the left,
improve=0.234955400, (0 missing)
##       num_floors_in_building splits as  RR---RRRRLLR-RRLL-
RRRRRRRRLRR, improve=0.074164930, (0 missing)
##       num_total_rooms        splits as  -R---RLRLL----,
improve=0.002681498, (0 missing)
##   Surrogate splits:
##       num_floors_in_building splits as  RR---RRLRLRR-LLLL-
RLRRRRLRLLR, agree=0.622, adj=0.124, (0 split)
##       num_total_rooms        splits as  -R---LRRRL----,
agree=0.599, adj=0.071, (0 split)
##
## Node number 72: 137 observations
##   mean=204709.4, MSE=6.005354e+09
##
## Node number 73: 17 observations
##   mean=343688, MSE=3.497117e+09
##
## Node number 74: 169 observations,    complexity param=0.01961643
##   mean=260223.1, MSE=5.549136e+09
##   left son=148 (75 obs) right son=149 (94 obs)
```

```
##   Primary splits:
##       kitchen_type           < 2.5  to the right, 
improve=0.40345540, (0 missing)
##       num_floors_in_building splits as  R----R-RRLL--RRLL-RR--L-
RLLRR, improve=0.14099830, (0 missing)
##       num_total_rooms        splits as  -----RRLLR----, 
improve=0.01755341, (0 missing)
##   Surrogate splits:
##       num_floors_in_building splits as  R----R-RRLL--LRLL-RR--L-
LLLRR, agree=0.639, adj=0.187, (0 split)
##       num_total_rooms        splits as  -----RRRLR----, 
agree=0.562, adj=0.013, (0 split)
## 
## Node number 75: 223 observations
##   mean=314956.6, MSE=0 
## 
## Node number 148: 75 observations
##   mean=207251.5, MSE=4.874463e+09 
## 
## Node number 149: 94 observations
##   mean=302487.8, MSE=2.06231e+09
```

#Visualize the Model

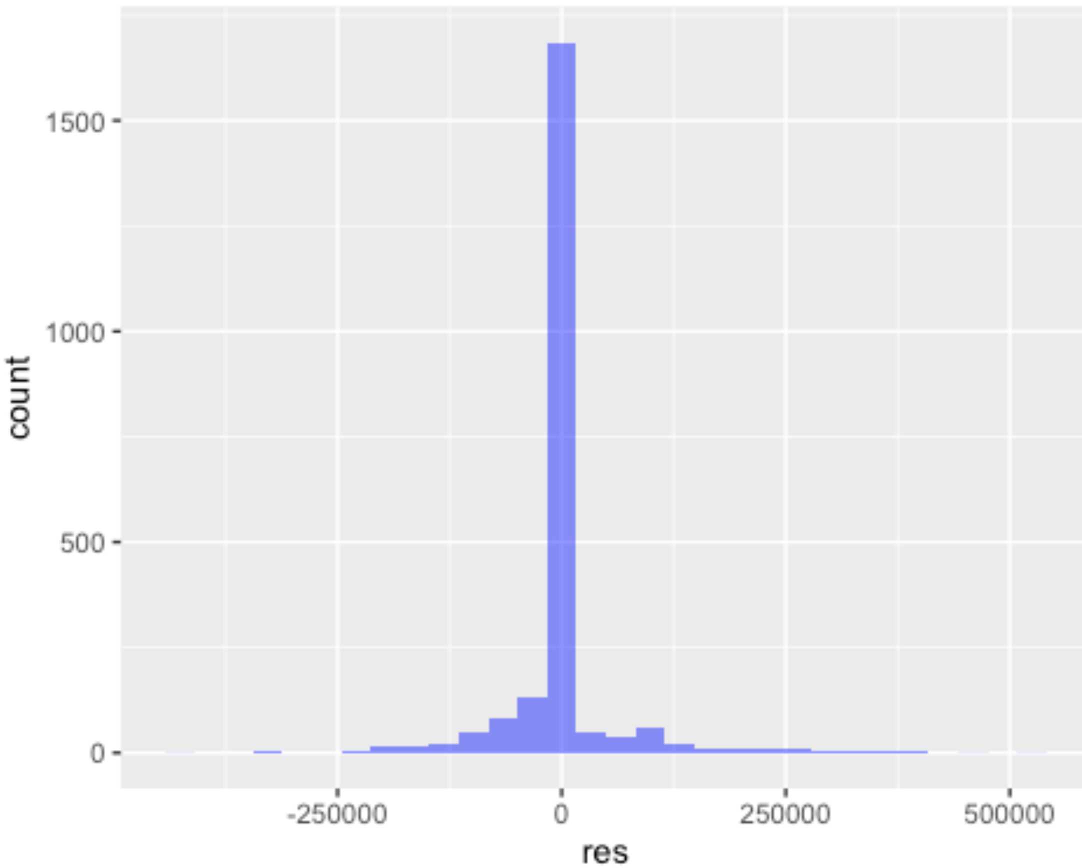
```
# Grab residuals
res <- residuals(tree_fit)
# Convert to DataFrame for gglpot
res <- as.data.frame(res)

head(res)
```

```
##          res
## 1  -86229.16
## 2   28248.53
## 3 -192505.31
## 4  -29660.17
## 5  -72529.16
## 6  -64229.16
```

#Histogram of residuals

```
ggplot(res, aes(res)) +  geom_histogram(fill = 'blue', alpha = 0.5)
```

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

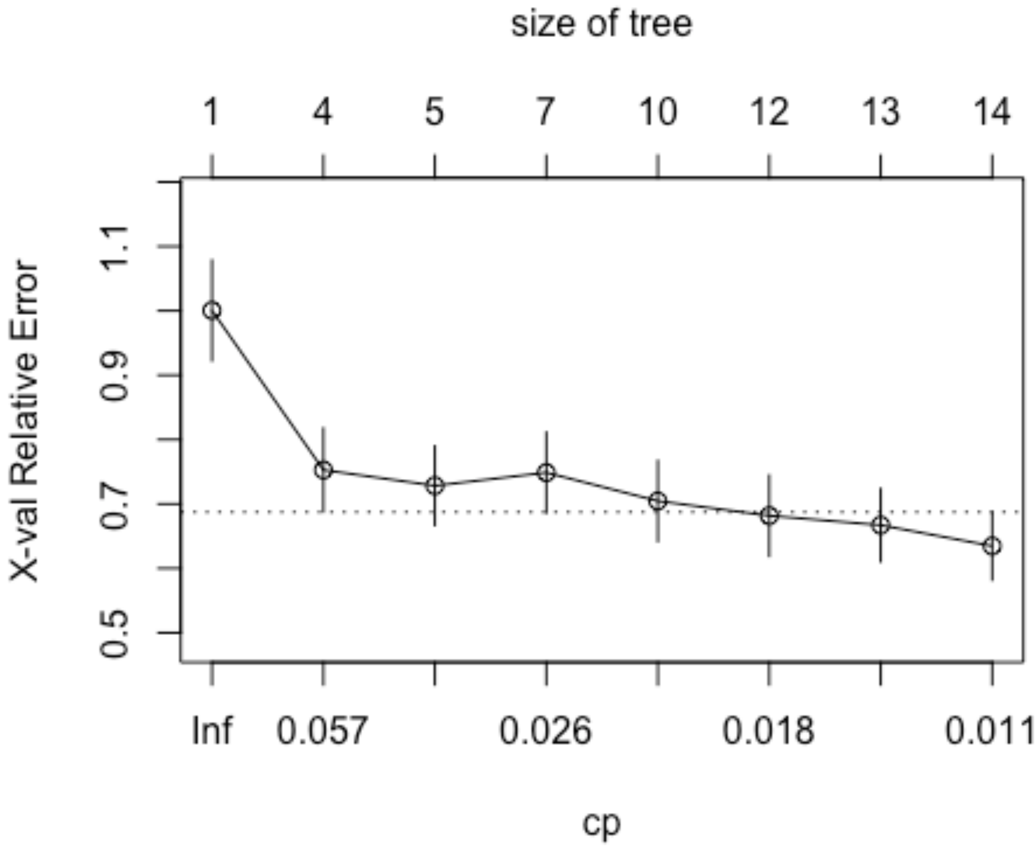


#Random forest CP Table

```
tree_fit$cptable

##           CP nsplit rel error    xerror       xstd
## 1 0.08326561      0 1.0000000 1.0005240 0.07798219
## 2 0.03939569      3 0.7502032 0.7527325 0.06496202
## 3 0.02775556      4 0.7108075 0.7283511 0.06151385
## 4 0.02447076      6 0.6552964 0.7486798 0.06265108
## 5 0.01961643      9 0.5818841 0.7045988 0.06269327
## 6 0.01719795     11 0.5426512 0.6817598 0.06272763
## 7 0.01177415     12 0.5254533 0.6670817 0.05667872
## 8 0.01000000     13 0.5136791 0.6347481 0.05267817
```

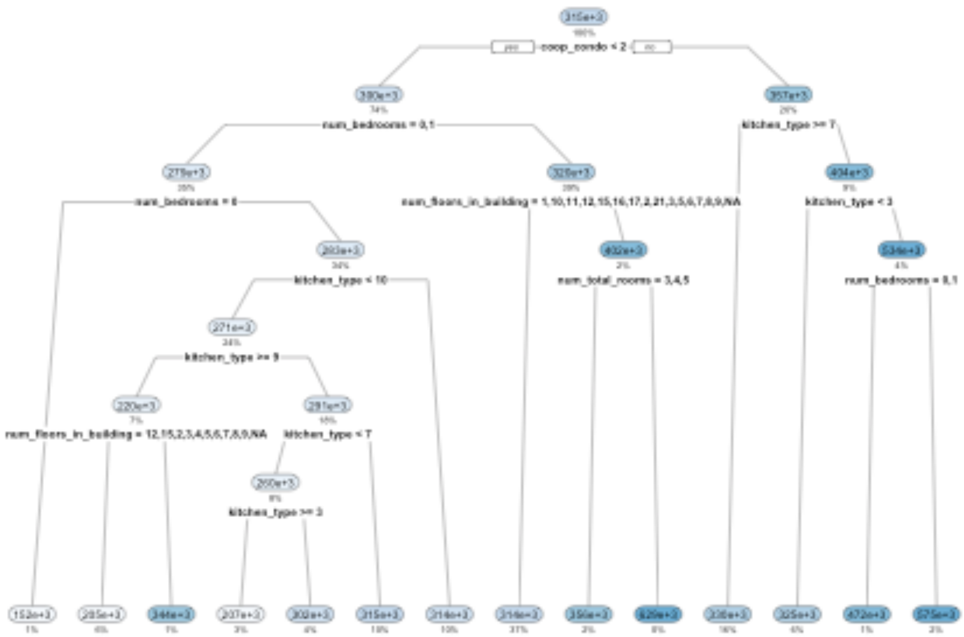
#Visualization of Random Forest CP table

```r
rpart::plotcp(tree_fit)
```



#Diagram of Random Forest

```r
library(rpart.plot)
rpart.plot::rpart.plot(
tree_fit,
type = 2,
branch = .8,
under = TRUE)
```



#Predicting the Random Forest Model

```
rf.pred <- predict(tree_fit, test)
```

#Now we can get the root mean squared error, a standardized measure of how off we were with our predicted values

```
results <- cbind(rf.pred, test$sale_price)
colnames(results) <- c('pred','real')
results <- as.data.frame(results)
```

#Now let's take care of negative predictions! Lot's of ways to this, here's a more complicated way, but its a good example of creating a custom function for a custom problem:

```
to_zero <- function(x){
    if  (x < 0){
```

```
        return(0)
    }else{
        return(x)
    }
}
results$pred <- sapply(results$pred, to_zero)
```

#MSE (mean squared error):

```
mse <- mean((results$real - results$pred) ^ 2)
print(mse)

## [1] 4400174855
```

#Root mean squared error

```
mse ^ 0.5

## [1] 66333.81
```

#Just the R-Squared Value for our model (just for the predictions)

```
SSE = sum((results$pred - results$real) ^ 2)
SST = sum((mean(data$sale_price) - results$real) ^ 2)

R2 = 1 - SSE/SST
R2

## [1] 0.4324472
```