Regression

Naomi Zilber

25 March 2023

Overview

In this notebook, I perform linear, kNN, and decision tree regression models on the data set. The data set used in this notebook is from this link. (https://www.kaggle.com/datasets/syuzai/perth-house-prices)

Load data

Read in the data of house_pricing

```
df <- read.csv("house_pricing.csv", header=TRUE)
str(df)
```

```
## 'data.frame':    33656 obs. of  19 variables:
##  $ ADDRESS         : chr  "1 Acorn Place" "1 Addis Way" "1 Ainsley Court"
"1 Albert Street" ...
##  $ SUBURB          : chr  "South Lake" "Wandi" "Camillo" "Bellevue" ...
##  $ PRICE           : int  565000 365000 287000 255000 325000 409000 400000 3
70000 565000 685000 ...
##  $ BEDROOMS        : int  4 3 3 2 4 4 3 4 4 3 ...
##  $ BATHROOMS       : int  2 2 1 1 1 2 2 2 2 2 ...
##  $ GARAGE          : chr  "2" "2" "1" "2" ...
##  $ LAND_AREA       : int  600 351 719 651 466 759 386 468 875 552 ...
##  $ FLOOR_AREA      : int  160 139 86 59 131 118 132 158 168 126 ...
##  $ BUILD_YEAR      : chr  "2003" "2013" "1979" "1953" ...
##  $ CBD_DIST        : int  18300 26900 22600 17900 11200 27300 28200 41700 12
100 5900 ...
##  $ NEAREST_STN     : chr  "Cockburn Central Station" "Kwinana Station" "Chal
lis Station" "Midland Station" ...
##  $ NEAREST_STN_DIST: int  1800 4900 1900 3600 2000 1000 3700 1100 2500 50
8 ...
##  $ DATE_SOLD       : chr  "09-2018\n" "02-2019\n" "06-2015\n" "07-2018
\n" ...
##  $ POSTCODE        : int  6164 6167 6111 6056 6054 6112 6112 6169 6022 605
3 ...
##  $ LATITUDE        : num  -32.1 -32.2 -32.1 -31.9 -31.9 ...
##  $ LONGITUDE       : num  116 116 116 116 116 ...
##  $ NEAREST_SCH     : chr  "LAKELAND SENIOR HIGH SCHOOL" "ATWELL COLLEGE" "KE
LMSCOTT SENIOR HIGH SCHOOL" "SWAN VIEW SENIOR HIGH SCHOOL" ...
##  $ NEAREST_SCH_DIST: num  0.828 5.524 1.649 1.571 1.515 ...
##  $ NEAREST_SCH_RANK: int  NA 129 113 NA NA NA NA NA NA 29 ...
```

Data Cleaning

Convert garage and build_year to be integers and get rid of features that will not be used for the regression models.

```
df <- df[,c(-1,-2,-11,-13,-14,-15,-16,-17)]
df$GARAGE <- as.integer(df$GARAGE)
```

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

```
df$BUILD_YEAR <- as.integer(df$BUILD_YEAR)
```

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

```
str(df)
```

```
## 'data.frame':    33656 obs. of  11 variables:
##  $ PRICE           : int  565000 365000 287000 255000 325000 409000 400000 3
70000 565000 685000 ...
##  $ BEDROOMS        : int  4 3 3 2 4 4 3 4 4 3 ...
##  $ BATHROOMS       : int  2 2 1 1 1 2 2 2 2 2 ...
##  $ GARAGE          : int  2 2 1 2 2 1 2 2 3 8 ...
##  $ LAND_AREA       : int  600 351 719 651 466 759 386 468 875 552 ...
##  $ FLOOR_AREA      : int  160 139 86 59 131 118 132 158 168 126 ...
##  $ BUILD_YEAR      : int  2003 2013 1979 1953 1998 1991 2014 2013 1983 199
9 ...
##  $ CBD_DIST        : int  18300 26900 22600 17900 11200 27300 28200 41700 12
100 5900 ...
##  $ NEAREST_STN_DIST: int  1800 4900 1900 3600 2000 1000 3700 1100 2500 50
8 ...
##  $ NEAREST_SCH_DIST: num  0.828 5.524 1.649 1.571 1.515 ...
##  $ NEAREST_SCH_RANK: int  NA 129 113 NA NA NA NA NA NA 29 ...
```

Handle missing values

Since school rank has many NAs, I decided to simply get rid of it. For the garage numbers, I assumed that an NA meant no garage space, so I replaced all NAs with zeroes. Lastly, for the build_year, I decided to replace all NAs with the mean of all of the build_year values.

```
sapply(df, function(x) sum(is.na(x)==TRUE))
```

```
##            PRICE         BEDROOMS        BATHROOMS           GARAGE
##                0                0                0             2478
##        LAND_AREA       FLOOR_AREA       BUILD_YEAR         CBD_DIST
##                0                0             3155                0
## NEAREST_STN_DIST NEAREST_SCH_DIST NEAREST_SCH_RANK
##                0                0            10952
```

```
df <- df[,-11]
df$GARAGE[is.na(df$GARAGE)] <- 0
df$BUILD_YEAR[is.na(df$BUILD_YEAR)] <- mean(df$BUILD_YEAR, na.rm=TRUE)
str(df)
```

```
## 'data.frame':    33656 obs. of  10 variables:
##  $ PRICE           : int  565000 365000 287000 255000 325000 409000 400000 3
70000 565000 685000 ...
##  $ BEDROOMS        : int  4 3 3 2 4 4 3 4 4 3 ...
##  $ BATHROOMS       : int  2 2 1 1 1 2 2 2 2 2 ...
##  $ GARAGE          : num  2 2 1 2 2 1 2 2 3 8 ...
##  $ LAND_AREA       : int  600 351 719 651 466 759 386 468 875 552 ...
##  $ FLOOR_AREA      : int  160 139 86 59 131 118 132 158 168 126 ...
##  $ BUILD_YEAR      : num  2003 2013 1979 1953 1998 ...
##  $ CBD_DIST        : int  18300 26900 22600 17900 11200 27300 28200 41700 12
100 5900 ...
##  $ NEAREST_STN_DIST: int  1800 4900 1900 3600 2000 1000 3700 1100 2500 50
8 ...
##  $ NEAREST_SCH_DIST: num  0.828 5.524 1.649 1.571 1.515 ...
```

Getting Rid of Outliers

It looks like there are two garage numbers that are most likely outliers. A garage size of 99 and 50 seems extremely unrealistic, so I removed them. I also removed other potential outliers, including an observation with 16 bathrooms and observations with land area of more than 600,000 squared meters. Removing these observations could be harmful since there is no way of knowing whether they really are outliers, but I assumed that they were outliers and therefore removed them so that they won't skew my models.

```
range(df$GARAGE)
```

```
## [1]  0 99
```

```
df[df$GARAGE>40,]
```

	PRICE	BEDR…	BATHR…	GA…	LAND_A…	FLOOR_…	BUILD_Y…	CBD_…
	<int>	<int>	<int>	<dbl>	<int>	<int>	<dbl>	<in
17287	375000	3	2	99	2916	126	2006	3470
30287	1180000	3	2	50	22367	257	1990	2260

2 rows | 1-10 of 11 columns

```
df <- df[df$GARAGE<50,]

range(df$BATHROOMS)
```

```
## [1]  1 16
```

```
df[df$BATHROOMS>10,]
```

	PRICE	BEDR…	BATHR…	GA…	LAND_A…	FLOOR_…	BUILD_Y…	CBD_…
	<int>	<int>	<int>	<dbl>	<int>	<int>	<dbl>	<int>
28426	300000	4	16	1	745	95	1977	12000

1 row | 1-10 of 11 columns

```
df <- df[df$BATHROOMS<10,]

range(df$LAND_AREA)
```

```
## [1]     61 999999
```

```
df[df$LAND_AREA>600000,]
```

	PRICE	BEDR…	BATHR…	GA…	LAND_A…	FLOOR_…	BUILD_Y…	CBD_…
	<int>	<int>	<int>	<dbl>	<int>	<int>	<dbl>	<int>
2685	375000	4	2	2	999999	175	2016	21400
3576	385000	3	2	2	999999	140	2015	22400
16792	545000	4	2	2	983690	216	2015	39500
27620	365000	4	2	2	999999	172	2015	21400

4 rows | 1-10 of 11 columns

```
df <- df[df$LAND_AREA<600000,]
```

Divide into train and test data

Divide the data to 80% train data and 20% test data

```
set.seed(1234)
i <- sample(1:nrow(df), 0.8*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Data Exploration

Explore data statistically and graphically. I look at the first 4 observations to get an idea of what my data looks like. I find the range of building year of the houses in the data set, and find the average price of the houses. Lastly, I look at the correlations between all of the features in the data set.

```
head(train, n=4)
```

	PRICE	BEDR…	BATHR…	GA…	LAND_A…	FLOOR_…	BUILD_Y…	CBD_…
	<int>	<int>	<int>	<dbl>	<int>	<int>	<dbl>	<int>
15243	290000	3	2	1	187	87	2016	18300
17491	850000	4	1	2	1054	93	1975	16300
15222	495000	4	2	1	1087	220	1978	17200
19842	225000	4	1	0	480	188	2015	26000

4 rows | 1-10 of 11 columns

```
range(train$BUILD_YEAR)
```

```
## [1] 1870 2017
```

```
mean(train$PRICE)
```

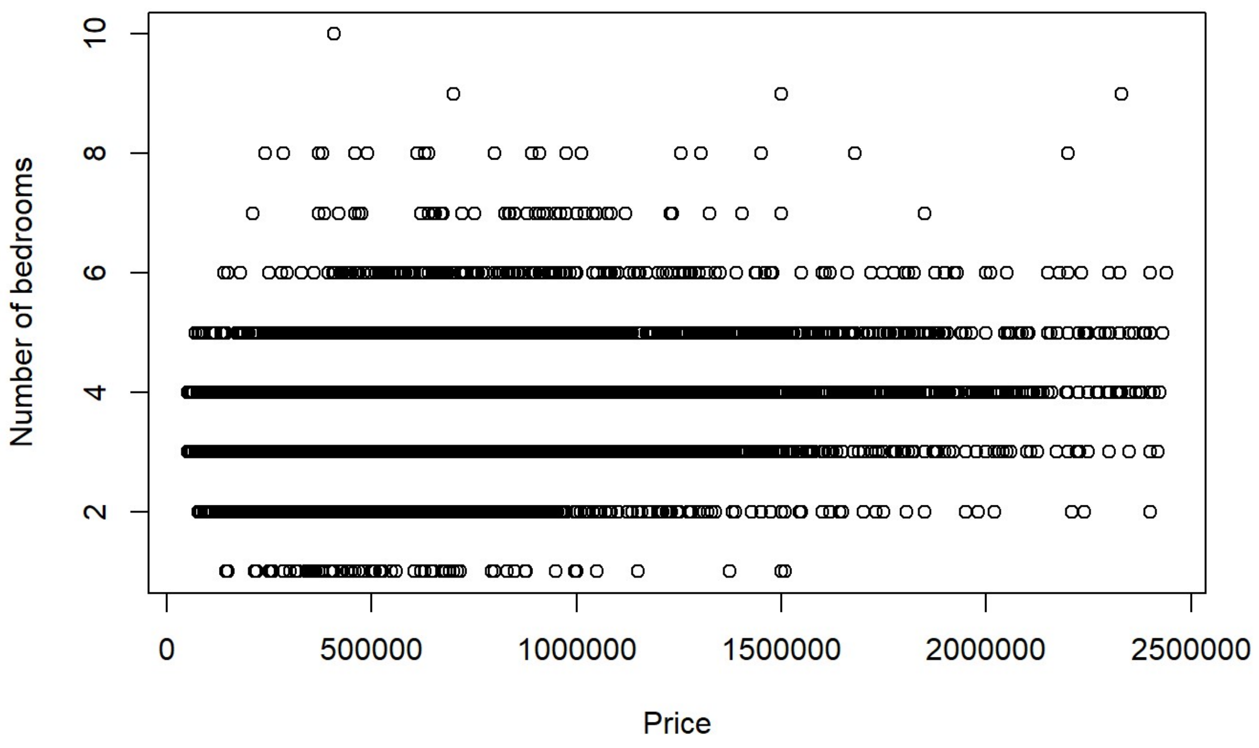
```
## [1] 636542.5
```

```
cor(train)
```

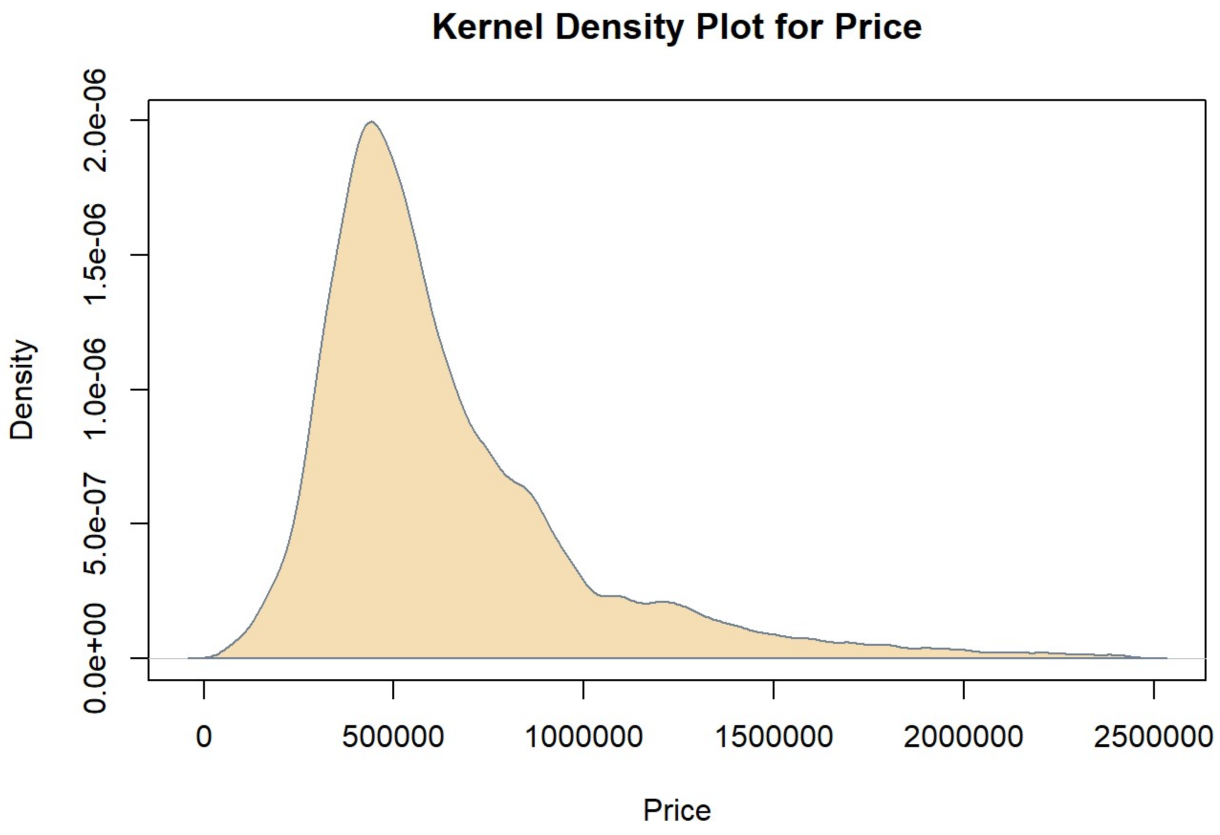
```
##                        PRICE   BEDROOMS  BATHROOMS     GARAGE    LAND_AREA
## PRICE             1.00000000 0.25436907 0.37847218 0.14675079  0.076927622
## BEDROOMS          0.25436907 1.00000000 0.55844942 0.21752504  0.065047055
## BATHROOMS         0.37847218 0.55844942 1.00000000 0.22847475  0.030241272
## GARAGE            0.14675079 0.21752504 0.22847475 1.00000000  0.043591978
## LAND_AREA         0.07692762 0.06504706 0.03024127 0.04359198  1.000000000
## FLOOR_AREA        0.54353244 0.53738008 0.55738957 0.19109708  0.096478077
## BUILD_YEAR       -0.14509322 0.21972258 0.33267185 0.05502331 -0.003405534
## CBD_DIST         -0.35529295 0.12021576 0.03160690 0.03230603  0.172990675
## NEAREST_STN_DIST -0.09316058 0.09898531 0.03514560 0.08394312  0.305551670
## NEAREST_SCH_DIST -0.02104145 0.08672268 0.05665383 0.07361362  0.384521782
##                  FLOOR_AREA   BUILD_YEAR    CBD_DIST NEAREST_STN_DIST
## PRICE            0.54353244 -0.145093216 -0.35529295      -0.09316058
## BEDROOMS         0.53738008  0.219722577  0.12021576       0.09898531
## BATHROOMS        0.55738957  0.332671853  0.03160690       0.03514560
## GARAGE           0.19109708  0.055023313  0.03230603       0.08394312
## LAND_AREA        0.09647808 -0.003405534  0.17299067       0.30555167
## FLOOR_AREA       1.00000000  0.217732957  0.02351493       0.10912987
## BUILD_YEAR       0.21773296  1.000000000  0.25370416       0.09646308
## CBD_DIST         0.02351493  0.253704162  1.00000000       0.44664256
## NEAREST_STN_DIST 0.10912987  0.096463080  0.44664256       1.00000000
## NEAREST_SCH_DIST 0.12068270  0.098950415  0.38041905       0.63736425
##                  NEAREST_SCH_DIST
## PRICE                 -0.02104145
## BEDROOMS               0.08672268
## BATHROOMS              0.05665383
## GARAGE                 0.07361362
## LAND_AREA              0.38452178
## FLOOR_AREA             0.12068270
## BUILD_YEAR             0.09895042
## CBD_DIST               0.38041905
## NEAREST_STN_DIST       0.63736425
## NEAREST_SCH_DIST       1.00000000
```

The first plot returns a graph that shows the price vs the number of bedrooms in the house, which shows that surprisingly it doesn’t seem like in price and number of bedrooms have a high correlation. The density plot represents the price. The boxplot shows the range of the price of houses. It looks like R believes that prices above around 1400000 are outliers. The next plot shows price vs floor area, which shows that these do seems to have a relationship where as floor area increases so does the price. Lastly, the histogram shows the distribution of number of garages. It appears like most houses don’t have many garages.

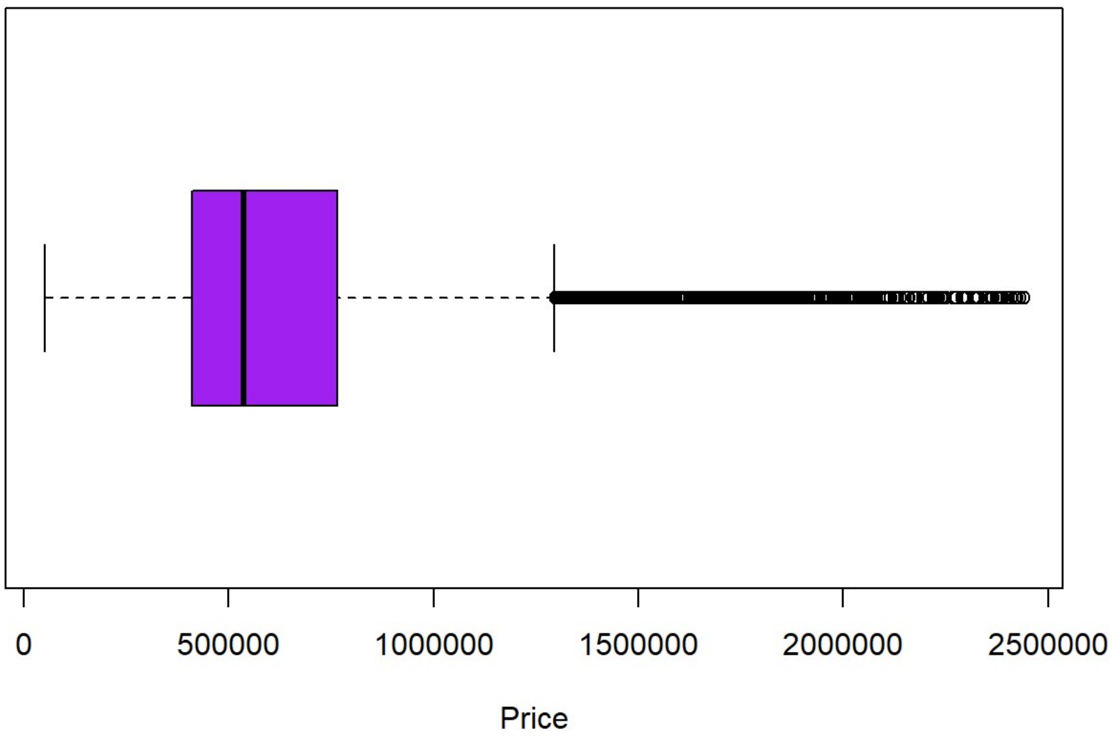
```
plot(train$PRICE, train$BEDROOMS, xlab="Price", ylab="Number of bedrooms")
```



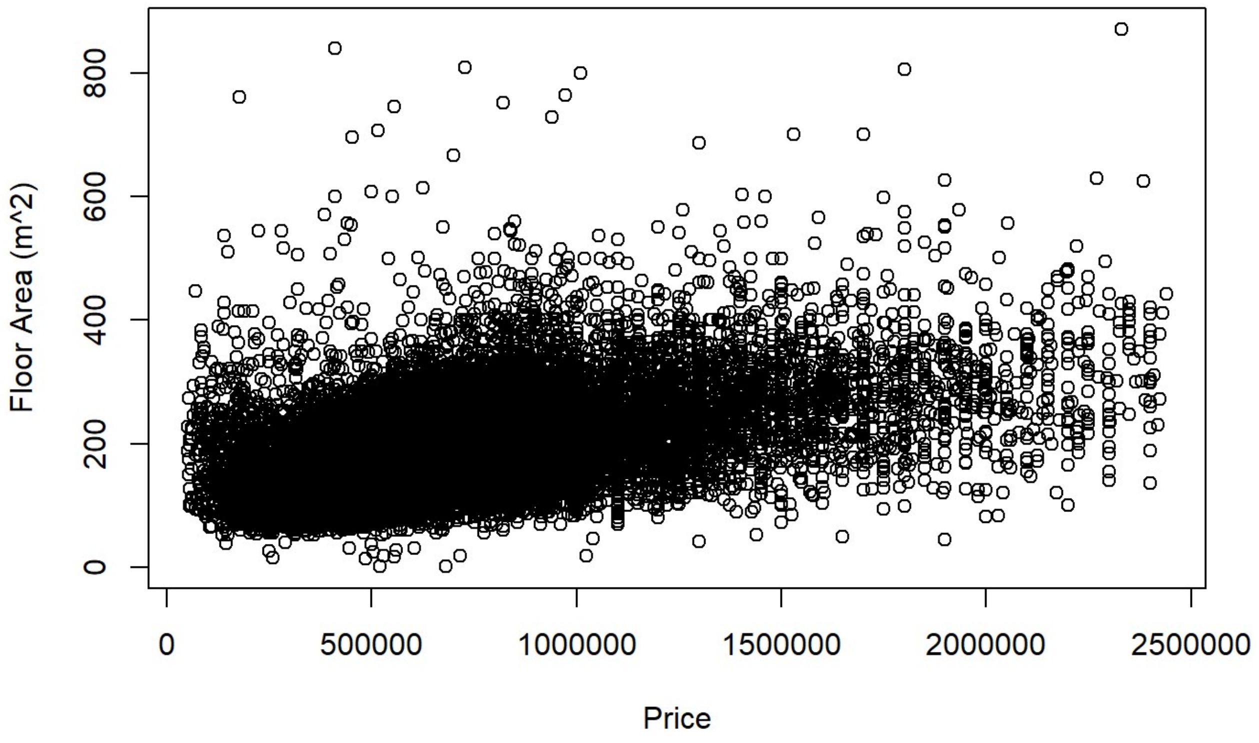
```
d <- density(train$PRICE, na.rm=TRUE)
plot(d, main="Kernel Density Plot for Price", xlab="Price")
polygon(d, col="wheat", border="slategray")
```



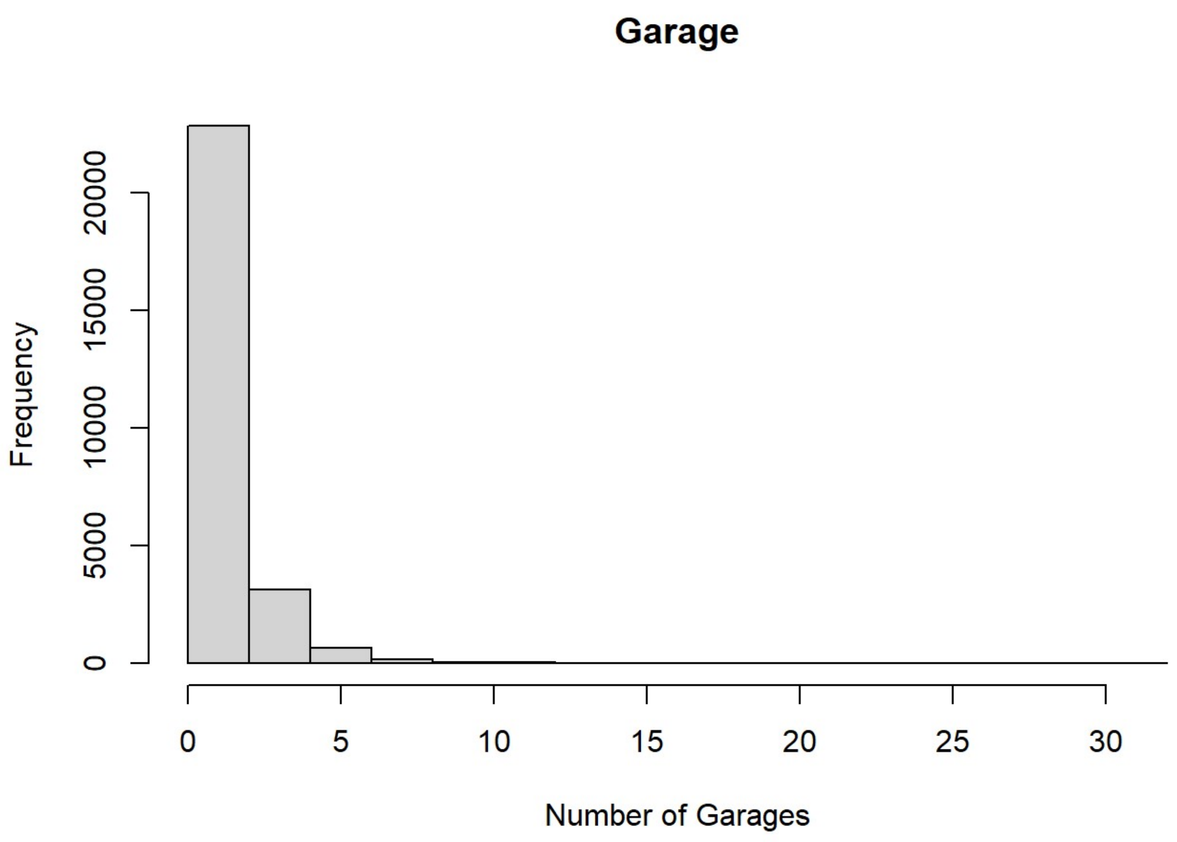
```
boxplot(train$PRICE, col="purple", horizontal=TRUE, xlab="Price")
```



```
plot(train$PRICE, train$FLOOR_AREA, xlab="Price", ylab="Floor Area (m^2)")
```

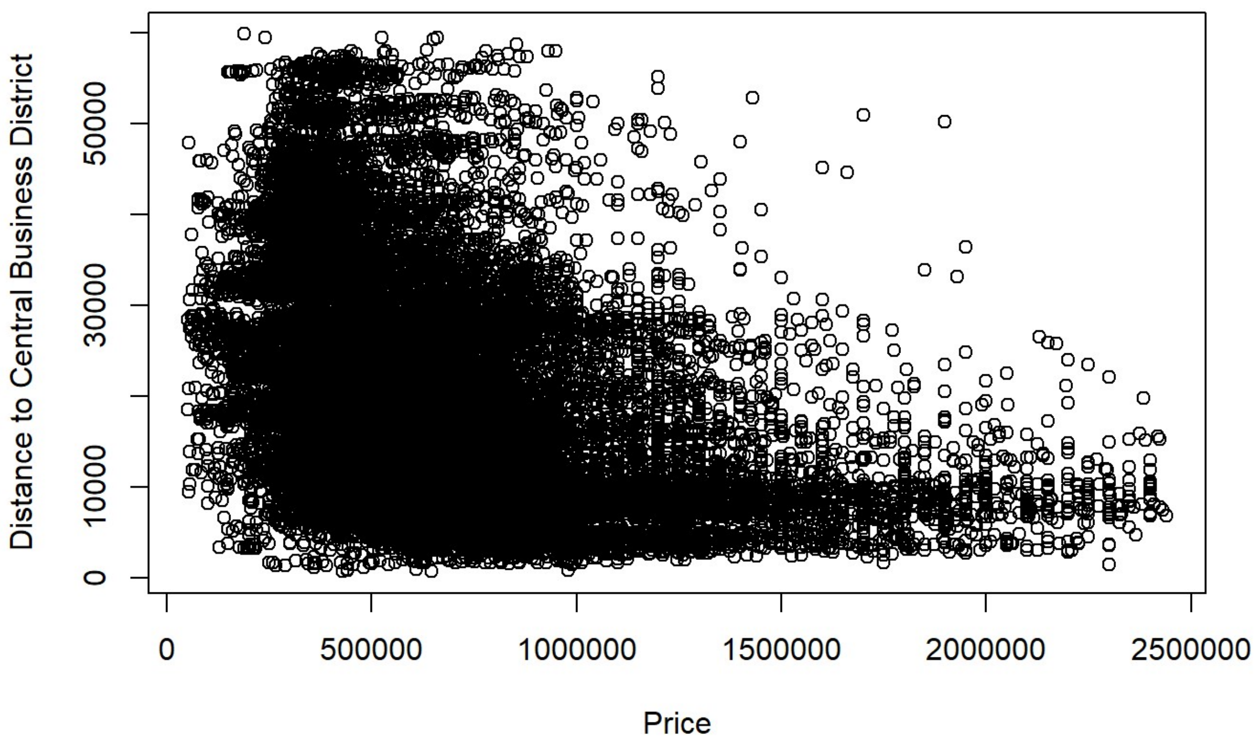


```
hist(train$GARAGE, main="Garage", xlab="Number of Garages")
```

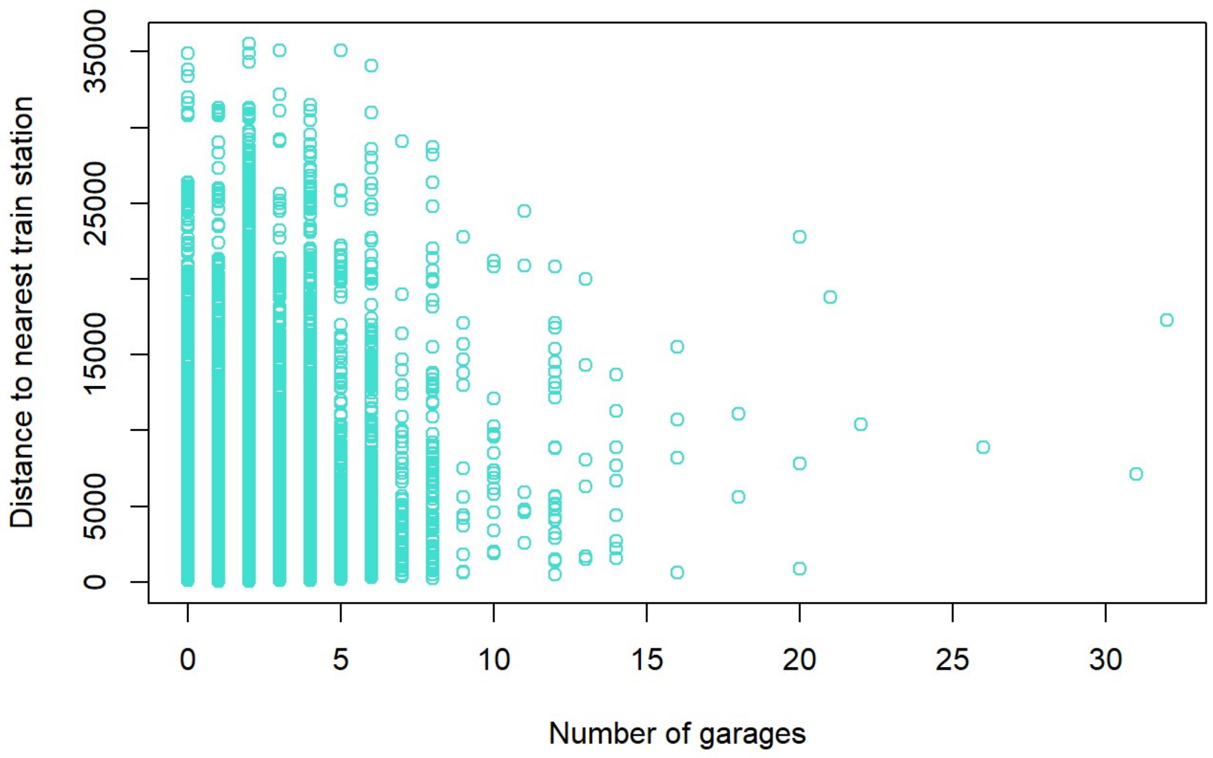


The first plot shows price vs distance to central business district, which show that the smaller the distance to the cbd, the more expensive the house. The second plot shows the number of garages vs the distance to the nearest train station and the third plot shows the build year of the house vs the price of the house. These plots let you get a general idea of how the data looks like.

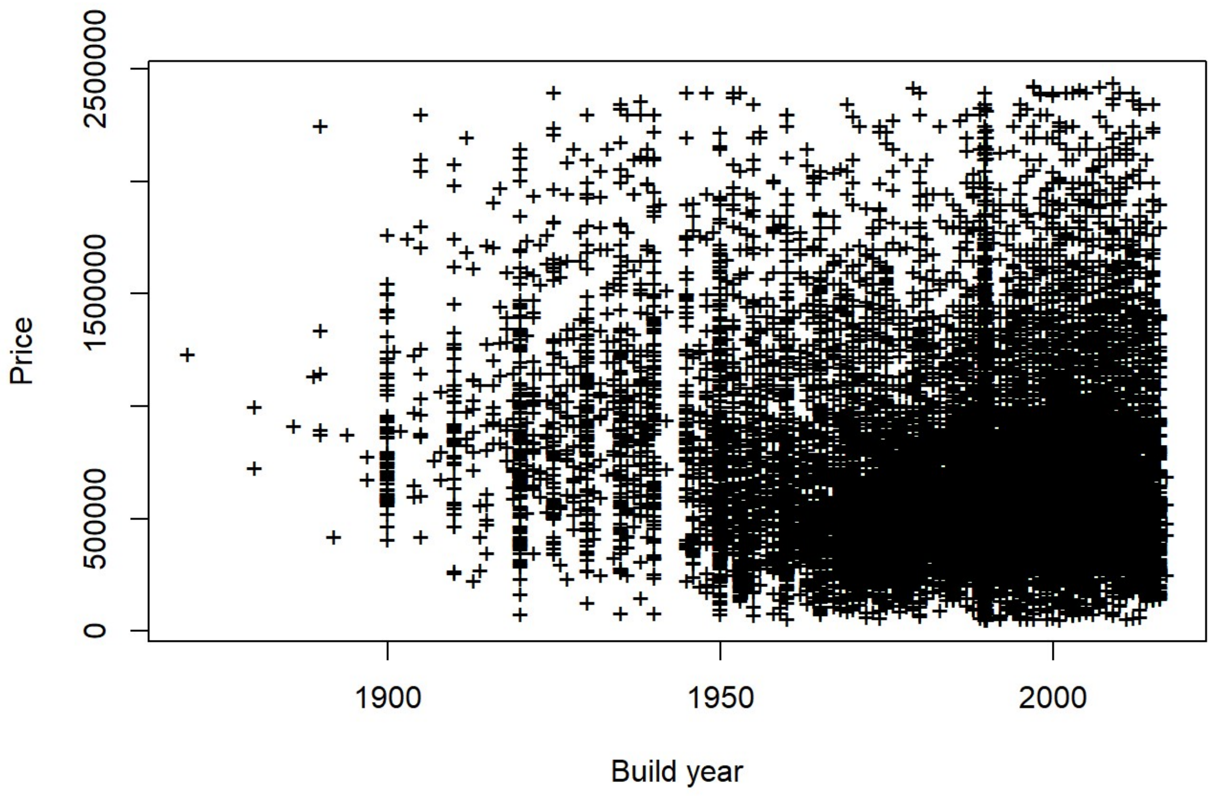
```
plot(train$PRICE, train$CBD_DIST, xlab="Price", ylab="Distance to Central Busin
ess District")
```



```
plot(train$GARAGE, train$NEAREST_STN_DIST, col="turquoise", xlab="Number of gar
ages", ylab="Distance to nearest train station")
```



```
plot(train$BUILD_YEAR, train$PRICE, pch='+', xlab="Build year", ylab="Price")
```



The strongest relationships seem to be between price and floor_area, build_year, and cbd_dist. Overall it seems like there are mainly weak relationships between the price and the rest of the features.

Linear Regression Model

Build a linear regression model to predict the price of the house from all of the features, and print a summary of the model.

The summary statistics show that the model has an R-squared of 0.4979 which is realtively good, meaning that the variance in the model is explained relatively well by the predictors. The RSE is high as 250400, which means that the model is relatively far off from the data. Lastly, the F-statistics is much larger than 1 with a small associated p-value, which indicates that R is relatively confident in the model.

```
lm1 <- lm(PRICE~., data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = PRICE ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2092630  -133444   -33619    94325  1787788
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       8.361e+06  1.653e+05  50.596  < 2e-16 ***
## BEDROOMS         -2.394e+04  2.615e+03  -9.157  < 2e-16 ***
## BATHROOMS         1.241e+05  3.547e+03  34.995  < 2e-16 ***
## GARAGE            9.543e+03  1.207e+03   7.904 2.81e-15 ***
## LAND_AREA         1.972e+00  1.298e-01  15.184  < 2e-16 ***
## FLOOR_AREA        2.447e+03  2.728e+01  89.701  < 2e-16 ***
## BUILD_YEAR       -4.093e+03  8.421e+01 -48.600  < 2e-16 ***
## CBD_DIST         -1.006e+01  1.573e-01 -63.938  < 2e-16 ***
## NEAREST_STN_DIST -3.314e+00  4.603e-01  -7.200 6.16e-13 ***
## NEAREST_SCH_DIST  1.100e+04  1.183e+03   9.302  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 250400 on 26909 degrees of freedom
## Multiple R-squared:  0.4979, Adjusted R-squared:  0.4978
## F-statistic:  2965 on 9 and 26909 DF,  p-value: < 2.2e-16
```

Predict

Correlation is very good but MSE looks extremely terrible

```
pred1 <- predict(lm1, newdata=test)
cor_lm <- cor(pred1, test$PRICE)
mse_lm <- mean((pred1 - test$PRICE)^2)

print(paste("cor = ", cor_lm))
```

```
## [1] "cor =  0.725175008930211"
```

```
print(paste("mse = ", mse_lm))
```

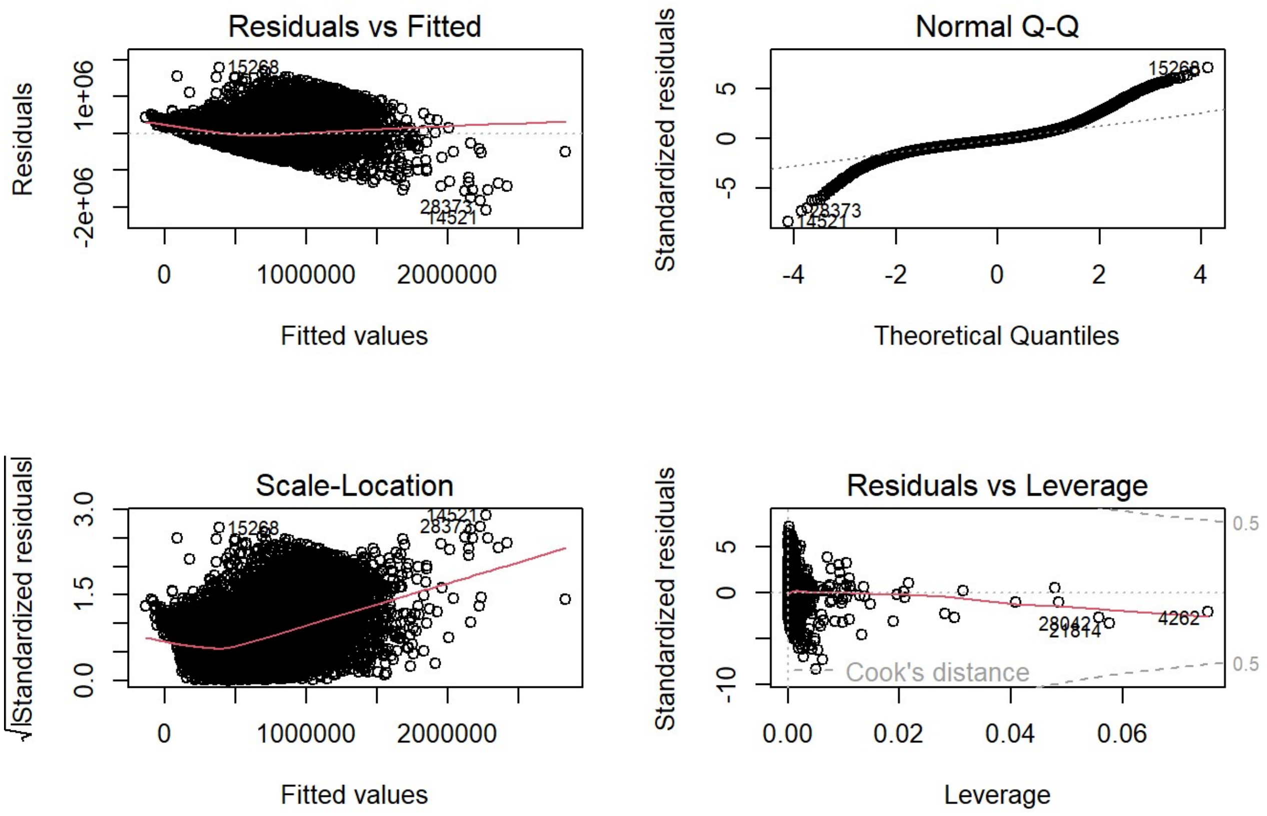
```
## [1] "mse =  63565285672.2517"
```

Plot residuals

The 4 residual plots tell us the following:

- Residuals vs Fitted - the residuals seem to have a slight non-linear pattern of a convex parabola which the linear model didn't entirely capture
- Normal Q-Q - the residuals seem to not follow the dashed line very well, so the residuals are not very well normally distributed
- Scale-Location - the line is not horizontal at all and it doesn't look like the residuals are equally distributed around the line
- Residuals vs Leverage - it seems that there are a few leverage points that are influencing the regression line

```
par(mfrow=c(2,2))
plot(lm1)
```



KNN Model

Before I can perform kNN, I need to find the best k.

Finding the Best k

Based on the results, the best k value is 5.

```r
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
cor_k <- rep(0, 7)
mse_k <- rep(0, 7)
i <- 1
for (k in seq(1, 7, 2)){
  fit_k <- knnreg(train[,2:10], train[,1], k=k)
  pred_k <- predict(fit_k, test[,2:10])
  cor_k[i] <- cor(pred_k, test$PRICE)
  mse_k[i] <- mean((pred_k - test$PRICE)^2)
  print(paste("k=", k, cor_k[i], mse_k[i]))
  i <- i + 1
}
```

```
## [1] "k= 1 0.653589115346965 88588097128.7254"
## [1] "k= 3 0.72414414483886 64518883945.4048"
## [1] "k= 5 0.730957622149009 62341026701.0593"
## [1] "k= 7 0.729294615757618 62689934827.2831"
```

Build a kNN regression model to predict the price of the house from all of the features.

```r
fit <- knnreg(train[,2:10], train[,1], k=5)
```

Predict for Non-Scaled kNN

Correlation is good but mse is bad.

```r
pred2 <- predict(fit, test[,2:10])
cor_knn <- cor(pred2, test$PRICE)
mse_knn <- mean((pred2 - test$PRICE)^2)

print(paste("cor = ", cor_knn))
```

```
## [1] "cor =  0.730957622149009"
```

```r
print(paste("mse = ", mse_knn))
```

```
## [1] "mse =  62341026701.0593"
```

Scaling

Build a kNN regression model like before but with scaled data.

Scale the data.

```
train_scaled <- train[,2:10]
means <- sapply(train_scaled, mean)
stdvs <- sapply(train_scaled, sd)
train_scaled <- scale(train_scaled, center=means, scale=stdvs)
test_scaled <- scale(test[,2:10], center=means, scale=stdvs)
```

Predict for Scaled kNN

The correlation and mse values are better than when the data was not scaled, as expected.

```
fit2 <- knnreg(train_scaled, train$PRICE, k=5)
predictions <- predict(fit2, test_scaled)
cor_knn2 <- cor(predictions, test$PRICE)
mse_knn2 <- mean((predictions - test$PRICE)^2)

print(paste("cor = ", cor_knn2))
```

```
## [1] "cor =  0.765752464715431"
```

```
print(paste("mse = ", mse_knn2))
```

```
## [1] "mse =  55517099799.4977"
```

Decision Tree Regression Model

Build a decision tree regression model to predict the price of the house from all of the features, and print a summary of the model. The summary shows that the tree used floor_area, cbd_dist, and build_year as the decisive parameters, and that the tree has 8 terminal nodes.

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.2.3
```

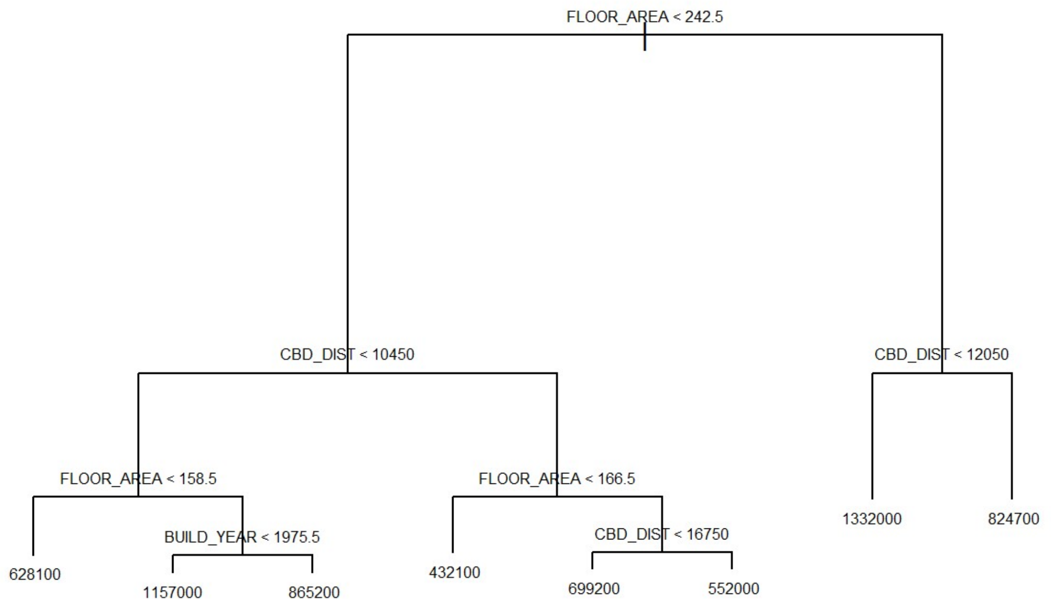
```
library(MASS)

tree1 <- tree(PRICE~., data=train)
summary(tree1)
```

```
##
## Regression tree:
## tree(formula = PRICE ~ ., data = train)
## Variables actually used in tree construction:
## [1] "FLOOR_AREA" "CBD_DIST"   "BUILD_YEAR"
## Number of terminal nodes:  8
## Residual mean deviance:  6.678e+10 = 1.797e+15 / 26910
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1247000  -139100   -33010        0    92980  1968000
```

Plot the tree

```
plot(tree1)
text(tree1, cex=0.5, pretty=0)
```



Predict

The correlation is relatively good but the mse is extremely bad.

```
pred3 <- predict(tree1, newdata=test)
cor_tree <- cor(pred3, test$PRICE)
mse_tree <- mean((pred3 - test$PRICE)^2)

print(paste("cor = ", cor_tree))
```

```
## [1] "cor =  0.686624023855767"
```

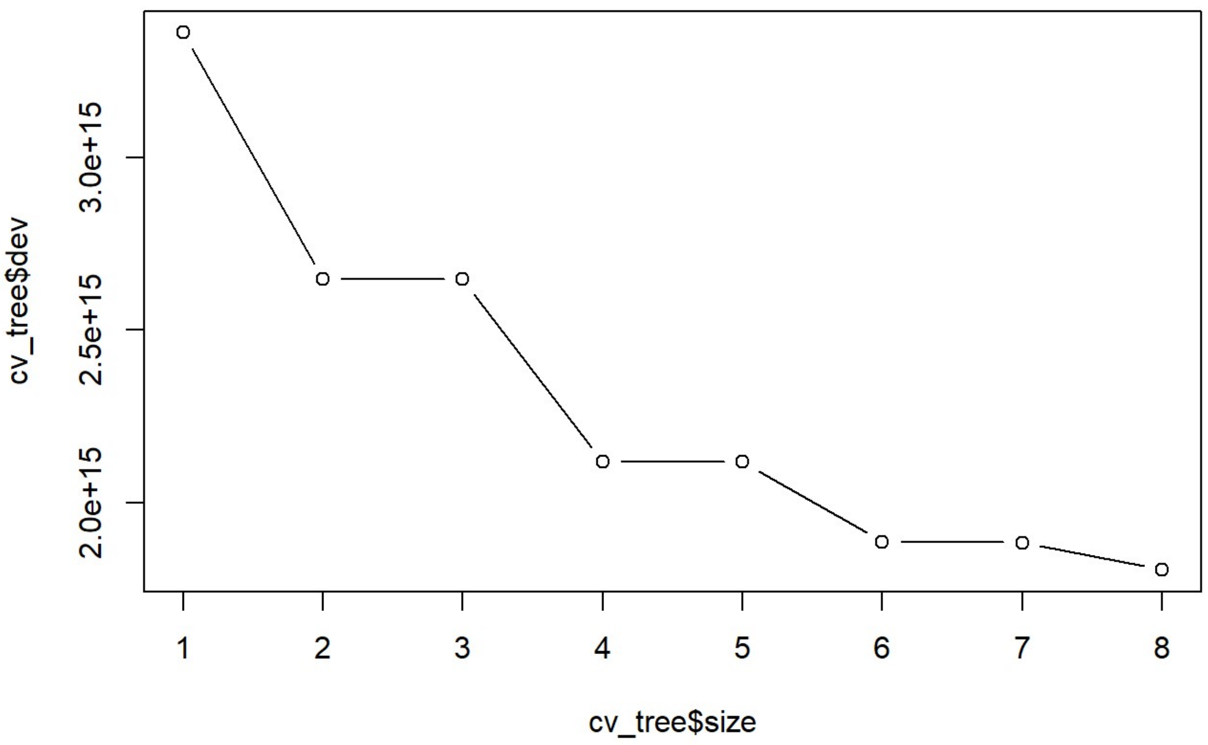
```
print(paste("mse = ", mse_tree))
```

```
## [1] "mse =  70770862278.3649"
```

Tree Pruning

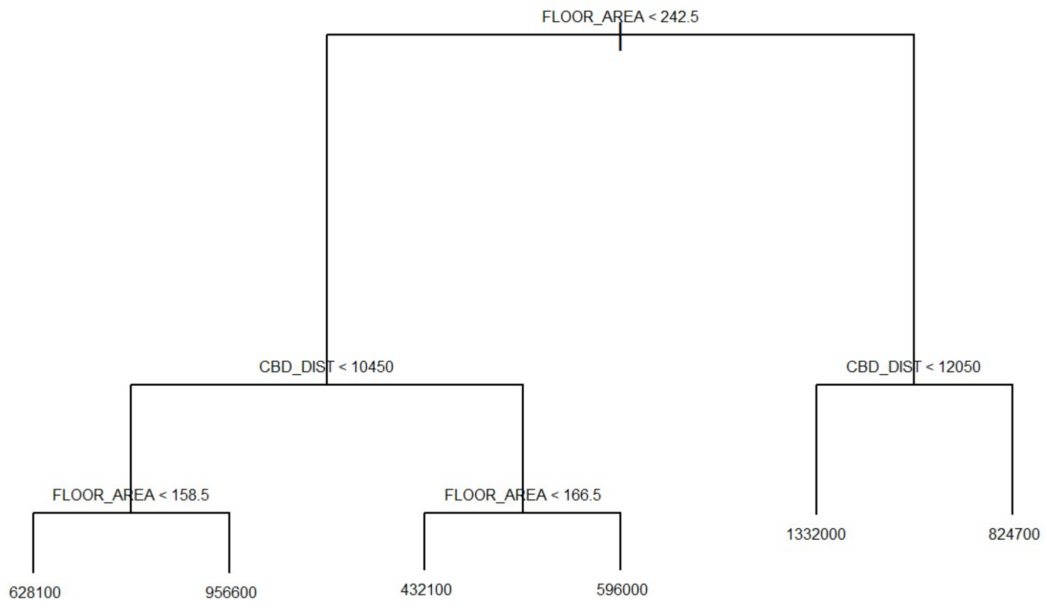
There is no clear curve in this plot, but it looks like the best choice would be to prune the tree to 5 or 6 terminal nodes.

```
cv_tree <- cv.tree(tree1)
plot(cv_tree$size, cv_tree$dev, type='b')
```



The tree is pruned to 6 terminal nodes.

```
tree_pruned <- prune.tree(tree1, best=6)
plot(tree_pruned)
text(tree_pruned, cex=0.5, pretty=0)
```



Predict on the Pruned Tree

It looks like the correlation and mse actually got worse then before pruning the tree. Therefore, in this case, pruning the tree didn’t improve the results, but the tree is now simpler and easier to interpret.

```
pred_pruned <- predict(tree_pruned, newdata=test)
cor_pruned <- cor(pred_pruned, test$PRICE)
mse_pruned <- mean((pred_pruned-test$PRICE)^2)

print(paste("cor = ", cor_pruned))
```

```
## [1] "cor =  0.673037614218809"
```

```
print(paste("mse = ", mse_pruned))
```

```
## [1] "mse =  73263699894.2188"
```

Evaluation

The results are as follows:

- Linear regression - correlation = 0.7252 and mse = 63565285672.2517
- Non-scaled kNN regression - correlation = 0.7310 and mse = 62341026701.0593
- Scaled kNN regression - correlation = 0.7658 and mse = 55517099799.4977
- Decision tree regression - correlation = 0.6866 and mse = 70770862278.3649

Based on these results, it looks like the scaled kNN regression model performed the best out of all of the models since it got the highest correlation and the lowest MSE value. After the scaled kNN regression model, the non-scaled kNN regression model performed the best, for the same reasons as before, though the linear regression model has a very close correlation to it but its mse is much higher. Therefore, the scaled kNN model performed best, with the non-scaled and linear models are close as second a third best models, while the decision tree model did the worst out of them all.

Models Description

Linear regression is a supervised regression technique in which the target is a real number variable and the predictors could be any combination of quantitative or qualitative variables. Some strengths of linear regression are:

- It is a relatively simple and intuitive algorithm
- It works well if the data has a linear pattern
- It has low variance

Some weaknesses are the fact that linear regression has a high bias because it assumes that there is a linear relationship between the target and the predictors (that the data has a linear shape), and thereby tend to underfit the data.

kNN regression is a a supervised regression technique but it doesn’t form a model of the input data, it instead stores all of the training observations in memory and then compares any new observation to existing ones to evaluate it and find the closest k neighbors. Some strengths of kNN regression are:

- It doesn’t assume the shape of the data
- It performs well in low dimensions
- It can be used for both classification and regression

Some weaknesses are:

- It can’t handle high dimensions
- K must be chosen
- The data needs to be scaled for best performance
- It is difficult to interpret

Decision trees have an algorithm that recursively split the input observations into partitions until the observations in a given partition are uniform, and the algorithm is greedy and doesn't go back and reconsider earlier splits. Performance wise, decision trees aren't the best, but they have the advantage of being highly interpretable and give insight into the data. Additionally, decision trees are sensitive to the distribution of predictors so slightly different data can result in very different trees.

Results Explanation

Based on the information above, it can be concluded that the results acquired happened for the following reasons:

- The linear regression model most likely outperformed the decision tree because the data might have a linear trend rather then a not linear and complex relationship between the predictors and the target.
- The scaled kNN regression obviously outperformed the non-scaled kNN regression because scaling improves the results for this algorithm
- The scaled kNN regression most likely outperformed the decision tree because decision trees are sensitive to the distribution of predictors, and therefore that could have made the decision tree perform worse.
- The scaled and non-scaled kNN regression both outperformed the linear regression model because they didn't assume the shape of the data while the linear regression model assumed that the data has a linear pattern, which might have not been the case.