Similarity/Regression

Saugat Gyawali/Bishal Neupane/Spencer Gray/Micheal Stinnett

10/08/2022

Source of data set is here

Description:

Comparison between linear regression and Decision tree Linear regression supports only linear solutions, whereas decision trees supports non linearity solutions too. Also, decision trees handles colinearity better than that of linear regression. Decison trees are better than linear regression for categorical independent variables.

Comparison between linear regression and KNN Linear regression is parametric model, whereas KNN is a non-parametric model. kNN is a slow model, because it need to find the neighbor nodes. But linear regression can easily extract output finding the weights.

Reading the csv file from kaggle data.

```
data <- read.csv("kc_house_data.csv")
```

Dividing the data into train and test data.

We divide the data in 80:20 ratio meaning, 80 percentage is for training and 20% of data is for testing purpose.

```
set.seed(1234)
i <- sample(1:nrow(data), nrow(data) * 0.80, replace=FALSE)
train <- data[i,]
test <- data[-i,]
```

Some of the data exploration of training datasets

```
names(train)
```

```
##  [1] "id"            "date"          "price"         "bedrooms"
##  [5] "bathrooms"     "sqft_living"   "sqft_lot"      "floors"
##  [9] "waterfront"    "view"          "condition"     "grade"
## [13] "sqft_above"    "sqft_basement" "yr_built"      "yr_renovated"
```

```
## [17] "zipcode"       "lat"           "long"          "sqft_living15"
## [21] "sqft_lot15"
```

```
dim(train)
```

```
## [1] 17290    21
```

```
summary(train)
```

```
##        id                date               price            bedrooms
##  Min.   :1.000e+06   Length:17290       Min.   :  75000   Min.   : 0.000
##  1st Qu.:2.116e+09   Class :character   1st Qu.: 320900   1st Qu.: 3.000
##  Median :3.902e+09   Mode  :character   Median : 450000   Median : 3.000
##  Mean   :4.564e+09                      Mean   : 541038   Mean   : 3.371
##  3rd Qu.:7.300e+09                      3rd Qu.: 645000   3rd Qu.: 4.000
##  Max.   :9.900e+09                      Max.   :6885000   Max.   :11.000
##    bathrooms      sqft_living       sqft_lot           floors
##  Min.   :0.000   Min.   :  290   Min.   :    520   Min.   :1.000
##  1st Qu.:1.750   1st Qu.: 1430   1st Qu.:   5034   1st Qu.:1.000
##  Median :2.250   Median : 1910   Median :   7616   Median :1.500
##  Mean   :2.117   Mean   : 2082   Mean   :  15175   Mean   :1.497
##  3rd Qu.:2.500   3rd Qu.: 2550   3rd Qu.:  10686   3rd Qu.:2.000
##  Max.   :8.000   Max.   :13540   Max.   :1651359   Max.   :3.500
##    waterfront            view          condition         grade
##  Min.   :0.000000   Min.   :0.0000   Min.   :1.000   Min.   : 1.000
##  1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.: 7.000
##  Median :0.000000   Median :0.0000   Median :3.000   Median : 7.000
##  Mean   :0.007808   Mean   :0.2403   Mean   :3.409   Mean   : 7.655
##  3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.: 8.000
##  Max.   :1.000000   Max.   :4.0000   Max.   :5.000   Max.   :13.000
##    sqft_above   sqft_basement       yr_built     yr_renovated
##  Min.   : 290   Min.   :   0.0   Min.   :1900   Min.   :   0.00
##  1st Qu.:1190   1st Qu.:   0.0   1st Qu.:1951   1st Qu.:   0.00
##  Median :1560   Median :   0.0   Median :1975   Median :   0.00
##  Mean   :1790   Mean   : 292.2   Mean   :1971   Mean   :  85.29
##  3rd Qu.:2210   3rd Qu.: 560.0   3rd Qu.:1997   3rd Qu.:   0.00
##  Max.   :9410   Max.   :4820.0   Max.   :2015   Max.   :2015.00
##     zipcode          lat             long        sqft_living15
##  Min.   :98001   Min.   :47.16   Min.   :-122.5   Min.   : 399
##  1st Qu.:98033   1st Qu.:47.47   1st Qu.:-122.3   1st Qu.:1486
##  Median :98065   Median :47.57   Median :-122.2   Median :1840
##  Mean   :98078   Mean   :47.56   Mean   :-122.2   Mean   :1987
##  3rd Qu.:98118   3rd Qu.:47.68   3rd Qu.:-122.1   3rd Qu.:2370
##  Max.   :98199   Max.   :47.78   Max.   :-121.3   Max.   :6210
##    sqft_lot15
##  Min.   :   659
##  1st Qu.:  5100
##  Median :  7620
##  Mean   : 12807
##  3rd Qu.: 10087
##  Max.   :871200
```

```r
str(train)
```

```
## 'data.frame':    17290 obs. of  21 variables:
##  $ id           : num  7.00e+09 3.89e+09 1.04e+09 8.66e+09 7.94e+09 ...
##  $ date         : chr  "20140715T000000" "20150304T000000" "20150312T000000" "20150330T000000" ...
##  $ price        : num  600000 606000 660000 537000 975000 ...
##  $ bedrooms     : int  3 3 3 4 3 3 4 4 3 3 ...
##  $ bathrooms    : num  1 2 3.5 2.5 2.5 1.5 2.5 1.5 2.25 1.5 ...
##  $ sqft_living  : int  940 1980 2740 1990 2530 1210 2320 1840 1560 2290 ...
##  $ sqft_lot     : int  19000 7680 3785 2660 7000 10588 9264 7076 35026 9600 ...
##  $ floors       : num  1 1.5 2 2 2.5 1 2 1.5 1 1 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 4 0 0 0 0 0 ...
##  $ condition    : int  3 4 3 3 3 4 3 3 3 4 ...
##  $ grade        : int  6 6 9 8 9 7 8 7 7 7 ...
##  $ sqft_above   : int  940 1070 2190 1990 2530 1210 2320 1840 1290 2290 ...
##  $ sqft_basement: int  0 910 550 0 0 0 0 0 270 0 ...
##  $ yr_built     : int  1945 1911 2001 2012 1915 1958 1994 1957 1985 1967 ...
##  $ yr_renovated : int  0 0 0 0 1999 0 0 0 0 0 ...
##  $ zipcode      : int  98004 98033 98034 98034 98136 98002 98188 98106 98092 98042 ...
##  $ lat          : num  47.6 47.7 47.7 47.7 47.5 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  2280 1330 2060 1990 2380 1408 2320 1510 1660 1310 ...
##  $ sqft_lot15   : int  19000 8704 3457 2665 7000 10588 9129 7320 35160 9600 ...
```

```r
print(head(train))
```

```
##              id            date  price bedrooms bathrooms sqft_living sqft_lot
## 7452 7000100635 20140715T000000 600000        3       1.0         940    19000
## 8016 3886903155 20150304T000000 606000        3       2.0        1980     7680
## 7162 1036450170 20150312T000000 660000        3       3.5        2740     3785
## 8086 8663240180 20150330T000000 537000        4       2.5        1990     2660
## 9196 7935000625 20150409T000000 975000        3       2.5        2530     7000
## 623  9500900135 20141021T000000 200000        3       1.5        1210    10588
##      floors waterfront view condition grade sqft_above sqft_basement yr_built
## 7452    1.0          0    0         3     6        940             0     1945
## 8016    1.5          0    0         4     6       1070           910     1911
## 7162    2.0          0    0         3     9       2190           550     2001
## 8086    2.0          0    0         3     8       1990             0     2012
## 9196    2.5          0    4         3     9       2530             0     1915
## 623     1.0          0    0         4     7       1210             0     1958
##      yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 7452            0   98004 47.5828 -122.190          2280      19000
## 8016            0   98033 47.6839 -122.195          1330       8704
## 7162            0   98034 47.7195 -122.182          2060       3457
## 8086            0   98034 47.7320 -122.178          1990       2665
## 9196         1999   98136 47.5465 -122.398          2380       7000
## 623             0   98002 47.2876 -122.212          1408      10588
```

```r
print(tail(train))
```

```
##               id            date   price bedrooms bathrooms sqft_living
```

```
## 6345  7276100020 20150414T000000  505000        4      1.00        1480
## 17565 8127700210 20150427T000000  600000        2      1.75        1560
## 8500  1722059021 20141217T000000  336500        3      2.00        1830
## 1830  7101100055 20150303T000000  753000        3      1.75        2360
## 657   3760500116 20141120T000000 3070000        3      2.50        3930
## 15486 2873000920 20150331T000000  257000        3      1.75        1430
##       sqft_lot floors waterfront view condition grade sqft_above sqft_basement
## 6345     12675    1.5          0    0         4     7       1480             0
## 17565     3200    1.0          0    0         5     7        880           680
## 8500     12891    1.0          0    0         3     7       1830             0
## 1830      8290    1.0          0    0         4     7       1180          1180
## 657      55867    1.0          1    4         4     8       2330          1600
## 15486     7210    1.0          0    0         3     7       1430             0
##       yr_built yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 6345      1929            0   98133 47.7630 -122.342          1820       7995
## 17565     1946            0   98199 47.6419 -122.394          2060       4940
## 8500      1994            0   98031 47.3924 -122.192          2320       8709
## 1830      1950            0   98115 47.6738 -122.281          1880       7670
## 657       1957            0   98034 47.7022 -122.224          2730      26324
## 15486     1975            0   98031 47.4189 -122.168          1220       7777
```

```
sum(is.na(train))
```

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

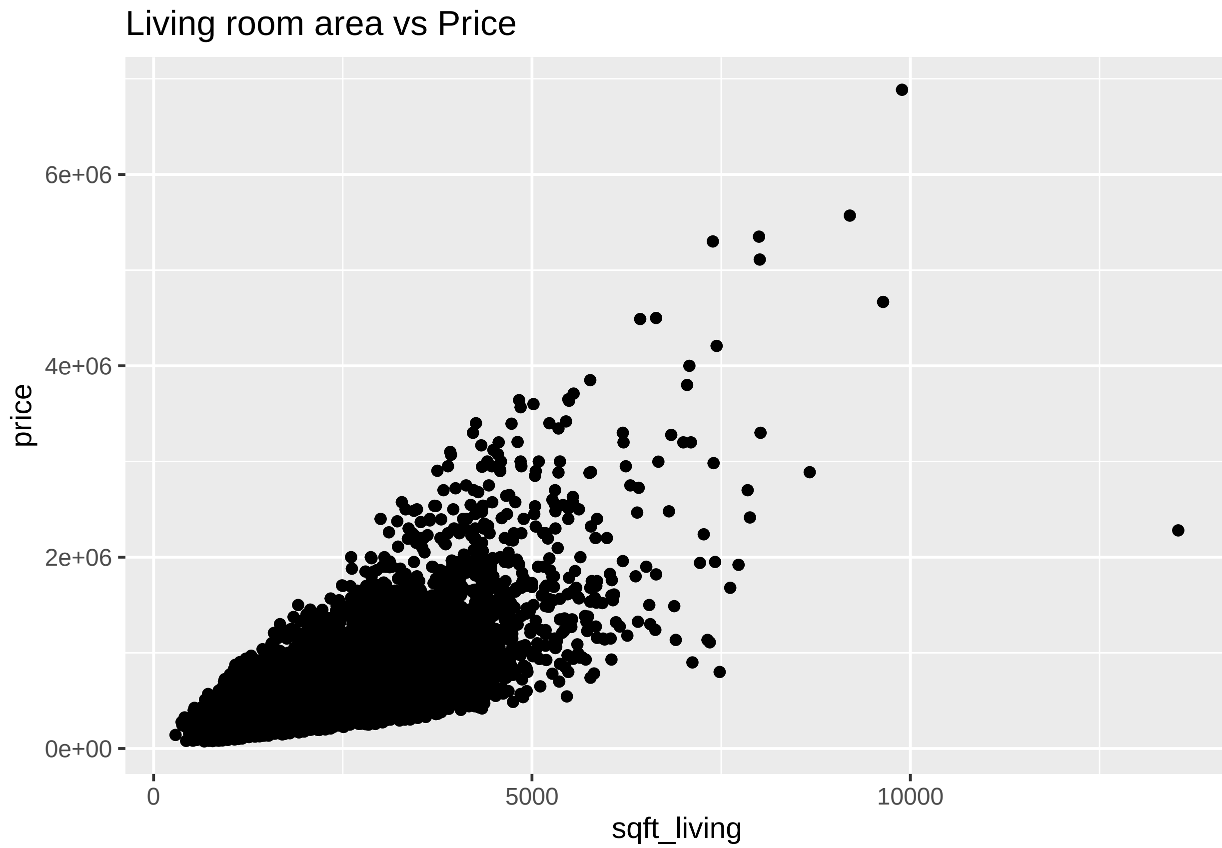
Some informative graphs

```


library(tidyverse)
```

Price vs Area of living room

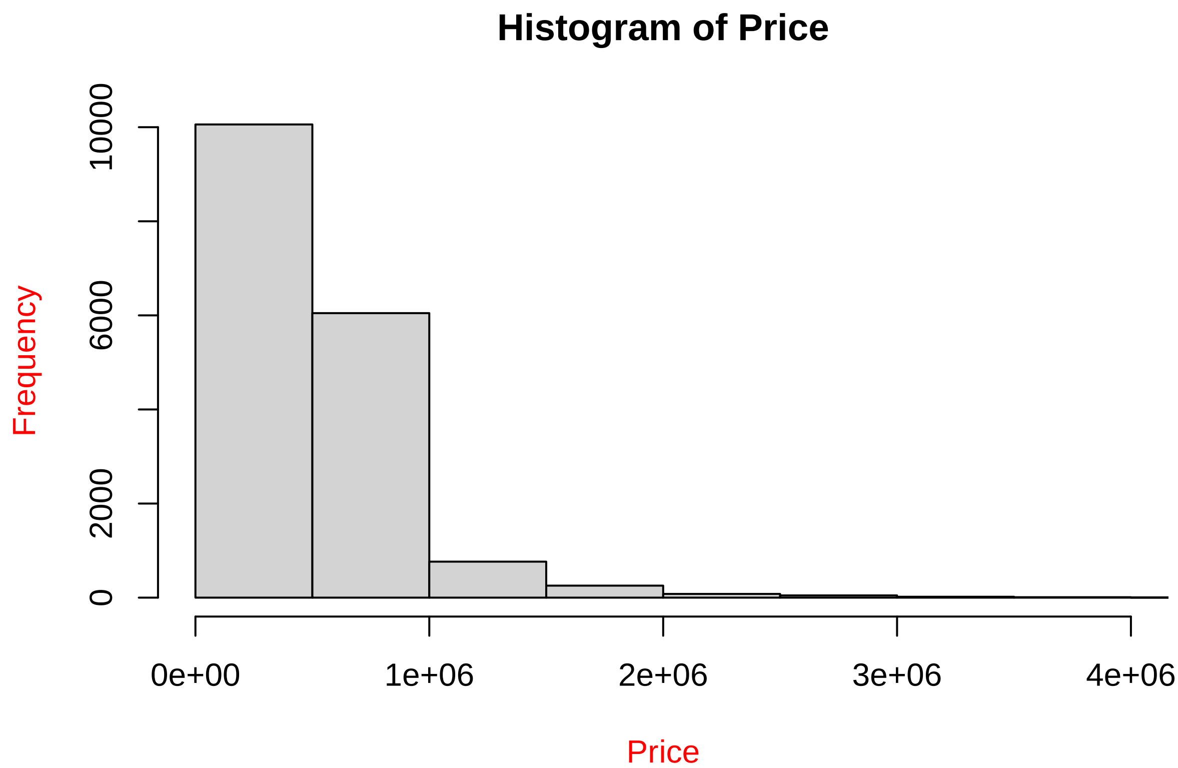
```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
ggplot(data=train, mapping=aes(x=sqft_living,y=price)) + ggtitle("Living room area vs Price") + geom_po
```



Histogram of Price

```
Price <- train$price
hist(Price, col.lab="red", xlim=c(0e+00, 4e+06))
```

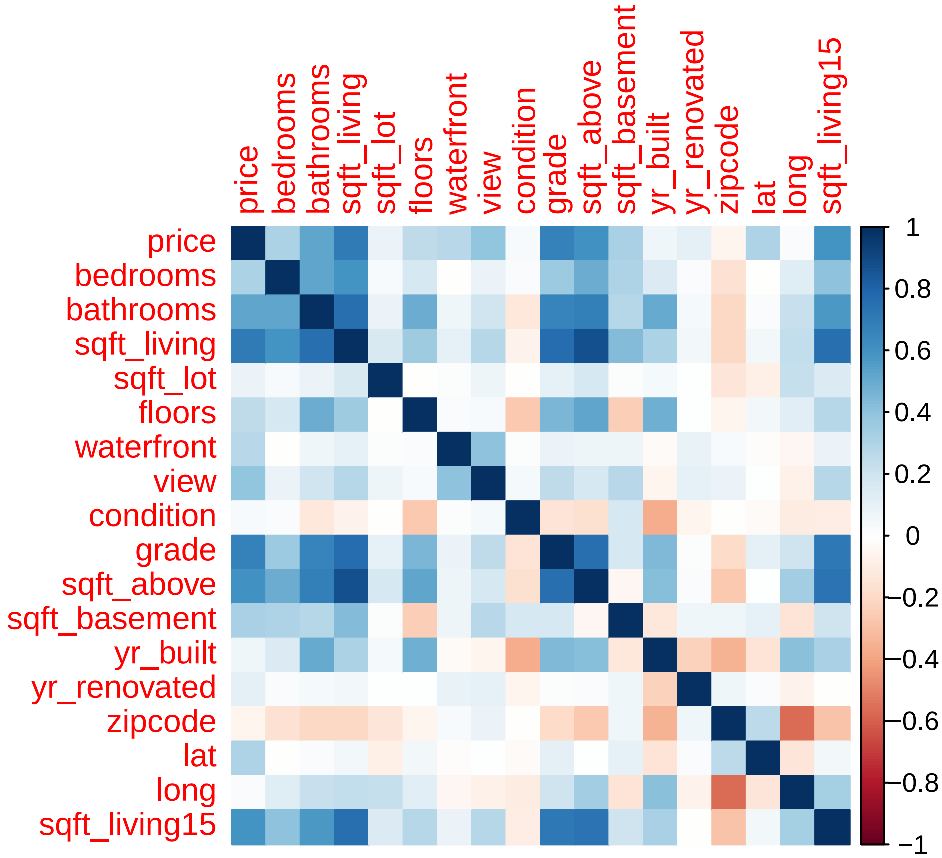



```r
#install.packages("corrplot")
library(corrplot)
```

Comparison of correlation between different parameters

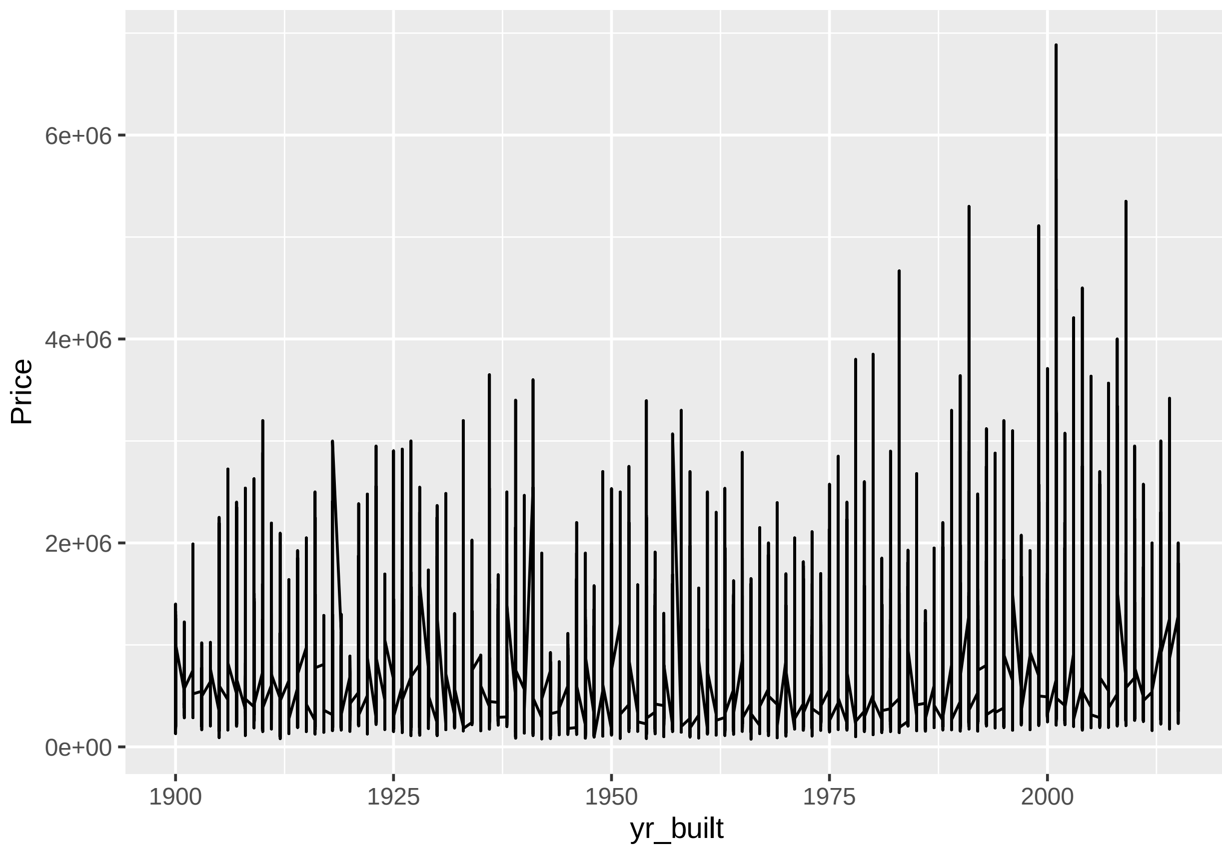
```
## corrplot 0.92 loaded
```

```r
trainData <- train[, 3:20]

M <- cor(trainData)

corrplot(M, method="color")
```



Finding trend of price based on year built

```
library(tidyverse)
ggplot(data=train, mapping=aes(x=yr_built,y=Price)) + geom_line()
```



Performing linear regression

```
lm1 <- lm(price~sqft_above, data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = price ~ sqft_above, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -890409 -165563  -41915  108900 4445909
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 60514.233   5274.673   11.47   <2e-16 ***
## sqft_above    268.462      2.673  100.42   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 291800 on 17288 degrees of freedom
## Multiple R-squared:  0.3684, Adjusted R-squared:  0.3684
## F-statistic: 1.008e+04 on 1 and 17288 DF,  p-value: < 2.2e-16
```

Adding multiple predictors

```
lm2 <- lm(price~sqft_living + sqft_above + grade + bathrooms, data = train)
summary(lm2)
```

```
##
## Call:
## lm(formula = price ~ sqft_living + sqft_above + grade + bathrooms,
##     data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1054889  -134355   -23558    98775  4521034
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.576e+05  1.511e+04 -43.509   <2e-16 ***
## sqft_living  2.463e+02  4.832e+00  50.985   <2e-16 ***
## sqft_above  -7.499e+01  4.912e+00 -15.269   <2e-16 ***
## grade        1.166e+05  2.645e+03  44.097   <2e-16 ***
## bathrooms   -3.437e+04  3.804e+03  -9.035   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 247800 on 17285 degrees of freedom
## Multiple R-squared:  0.5447, Adjusted R-squared:  0.5446
## F-statistic:  5169 on 4 and 17285 DF,  p-value: < 2.2e-16
```

Predicting using the test datasets

```
pred2 <- predict(lm2, newdata=test)
cor_lr <- cor(pred2, test$price)
mse_lr <- mean((pred2-test$price)^2)
rmse_lr <- sqrt(mse_lr)
```

Using kNN Regression

When k = 3

```
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
fit <- knnreg(train[,c('sqft_living', 'sqft_above', 'grade', 'bathrooms')], train[, c('price')], k = 3)
predictions_kequal3 <- predict(fit, test[,c('sqft_living', 'sqft_above', 'grade', 'bathrooms')])
cor_kequal3 <- cor(predictions_kequal3, test$price)
mse_kequal3 <- mean((predictions_kequal3 - test$price)^2)
rmse_kequal3 <- sqrt(mse_kequal3)
```

Finding the best K

```
cor_k <- rep(0, 20)
mse_k <- rep(0, 20)
i <- 1
for (k in seq(1, 39, 2)){
fit_k <- knnreg(train[,c('sqft_living', 'sqft_above', 'grade', 'bathrooms')], train[, c('price')],k=k)
pred_k <- predict(fit_k, test[,c('sqft_living', 'sqft_above', 'grade', 'bathrooms')] )
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 0.627611528752512 92553018402.9019"
## [1] "k= 3 0.717728308991263 66667524828.139"
## [1] "k= 5 0.726576272343119 63838996585.9256"
## [1] "k= 7 0.731275045571293 62710305828.2803"
## [1] "k= 9 0.737419711803711 61478436698.6668"
## [1] "k= 11 0.736812661475036 61636018489.6152"
## [1] "k= 13 0.737967203169501 61436777043.7364"
## [1] "k= 15 0.736325425536694 61755178005.3559"
## [1] "k= 17 0.73375275882837 62249446648.0346"
## [1] "k= 19 0.735612760318792 61923192326.6537"
## [1] "k= 21 0.732616160242743 62539492564.2755"
## [1] "k= 23 0.731820866600733 62704255741.9309"
## [1] "k= 25 0.733236532435928 62426109993.6211"
## [1] "k= 27 0.734927749339924 62125668075.6036"
## [1] "k= 29 0.734629118612551 62207871338.6651"
## [1] "k= 31 0.733206018925886 62497900117.01"
## [1] "k= 33 0.730537461351237 63012729373.0356"
## [1] "k= 35 0.728336976696355 63452560974.8172"
## [1] "k= 37 0.728278780287115 63487773914.2245"
## [1] "k= 39 0.728212445564998 63508294711.6873"
```

```
min_mse <- which.min(mse_k)
max_cor <- which.max(cor_k)
print(paste("Min mse = ", min_mse))
```

```
## [1] "Min mse =  7"
```

```
print(paste("Max cor_k = ", max_cor))
```

```
## [1] "Max cor_k =  7"
```

'7' is found to be the best k, while checking with minimum mse and maximum cor_k. Now, again implementing kNN regression using k = 7

When k = 7

```
library(caret)
fit <- knnreg(train[,c('sqft_living', 'sqft_above', 'grade', 'bathrooms')], train[, c('price')], k = 7)
predictions_kequal7 <- predict(fit, test[,c('sqft_living', 'sqft_above', 'grade', 'bathrooms')])
cor_kequals7 <- cor(predictions_kequal7, test$price)
mse_kequals7 <- mean((predictions_kequal7 - test$price)^2)
rmse_kequals7 <- sqrt(mse_kequals7)
```

We didn't get the better result yet. Now, we can scale the data so that it might produce the better result.

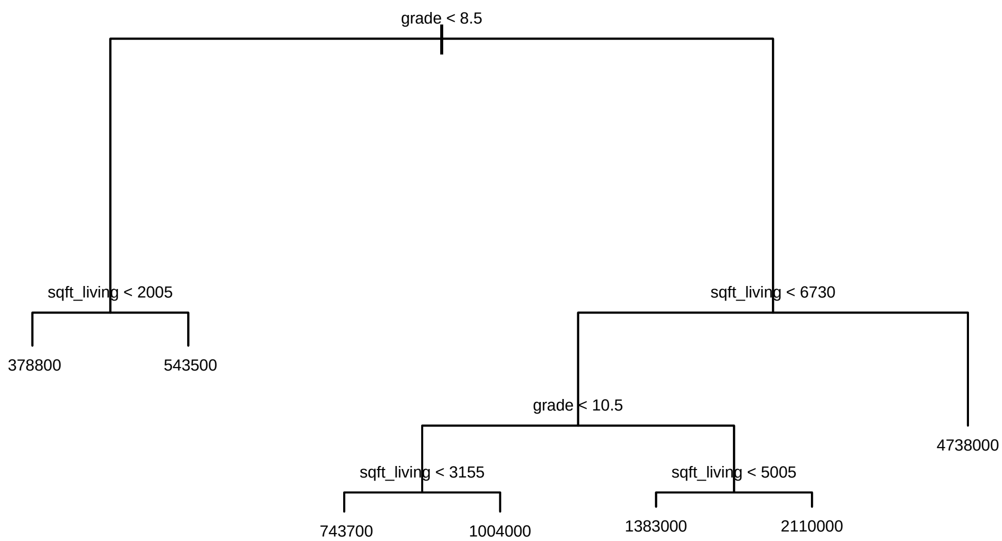
---

kNN Regression by normalizing the data

```
library(caret)

normalize <- function(x){
  return ((x-min(x))/(max(x)-min(x)))
}


#Creating a new dataframe
#For training
dfnew1 <- data.frame(train$sqft_living, train$sqft_above, train$grade, train$bathrooms, train$price)

#For test
dfnew2 <- data.frame(test$sqft_living, test$sqft_above, test$grade, test$bathrooms, test$price)


names(dfnew1) <- c("sqft_living", "sqft_above", "grade", "bathrooms", "price")
names(dfnew2) <- c("sqft_living", "sqft_above", "grade", "bathrooms", "price")
dfnew1_scaled <- as.data.frame(lapply(dfnew1,normalize))
dfnew2_scaled <- as.data.frame(lapply(dfnew2,normalize))
fit <- knnreg(dfnew1_scaled[,1:4], dfnew1_scaled[,5], k = 7)
predictions_normalizing <- predict(fit, dfnew2_scaled[, 1:4])
cor_normalizing <- cor(predictions_normalizing, dfnew2_scaled[,5])
mse_normalizing <- mean((predictions_normalizing - dfnew2_scaled[,5])^2)
rmse_normalizing <- sqrt(mse_normalizing)
```

Decision tree regression

```
library(tree)
tree_prices <- tree(price~., data=dfnew2)
plot(tree_prices)
text(tree_prices, cex=0.5, pretty=0)
```

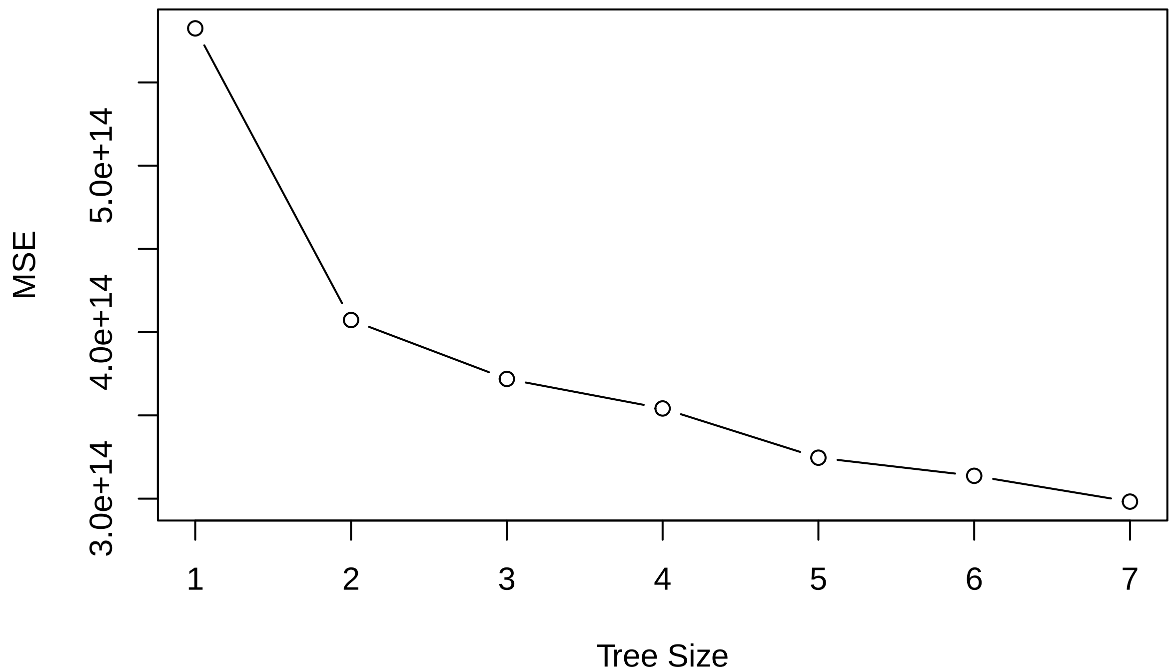


Predicting using the test data set!

```
decisiontree_pred <- predict(tree_prices, dfnew2)
mse_decisiontree <- mean((decisiontree_pred-test$price)^2)
cor_decisiontree <- cor(decisiontree_pred, dfnew2$price)
rmse_decisiontree <- sqrt(mse_decisiontree)
```

Cross validation for pruning the tree

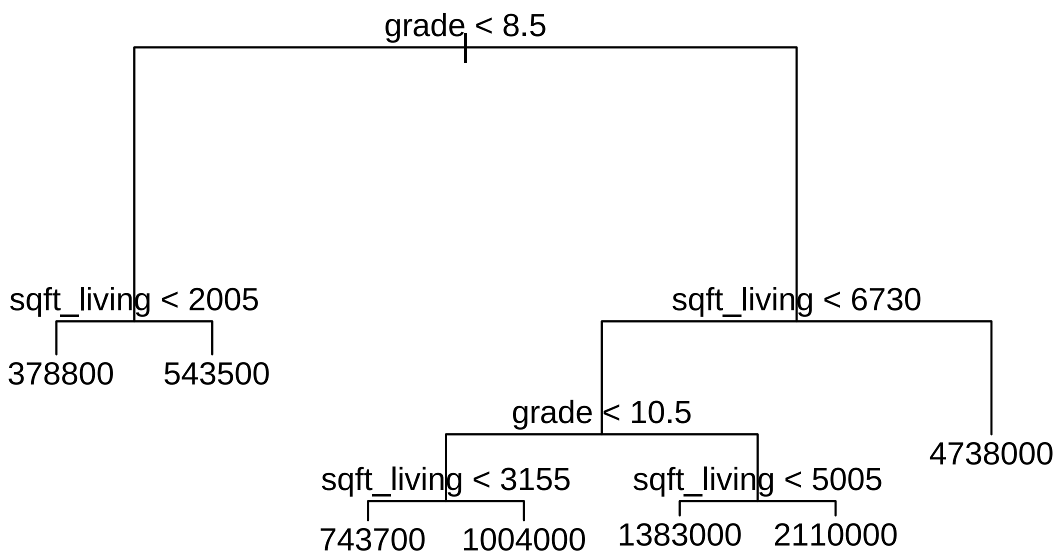
```
cv_tree <- cv.tree(tree_prices)
plot(cv_tree$size, cv_tree$dev, type="b", xlab="Tree Size", ylab="MSE")
```




```
min <- which.min(cv_tree$dev)
print(paste("For minimum MSE chose Tree Size = ", cv_tree$size[1]))
```

```
## [1] "For minimum MSE chose Tree Size =  7"
```

```
tree_pruned <- prune.tree(tree_prices, best = 7)
plot(tree_pruned)
text(tree_pruned, pretty=0)
```



***

We do not need to check the accuracy for the pruned model because the pruning does not help here.

```
temp_pred <- predict(tree_pruned, dfnew2)
temp_mse <- mean((temp_pred-dfnew2$price)^2)
cor <- cor(temp_pred, dfnew2$price)
```

Comparing the result

For linear regression

```
print(paste('correlation:', cor_lr))
```

```
## [1] "correlation: 0.734105665778279"
```

```
print(paste('mse:',mse_lr))
```

```
## [1] "mse: 62132083493.4327"
```

```r
print(paste('rmse:', rmse_lr))
```

```
## [1] "rmse: 249263.080887308"
```

For KNN regression when k = 3

```r
print(paste('correlation:', cor_kequal3))
```

```
## [1] "correlation: 0.717728308991263"
```

```r
print(paste('mse:',mse_kequal3))
```

```
## [1] "mse: 66667524828.139"
```

```r
print(paste('rmse:', rmse_kequal3))
```

```
## [1] "rmse: 258200.551564359"
```

For kNN regression when k = 7

```r
print(paste('correlation:', cor_kequals7))
```

```
## [1] "correlation: 0.731275045571293"
```

```r
print(paste('mse:',mse_kequals7))
```

```
## [1] "mse: 62710305828.2803"
```

```r
print(paste('rmse:', rmse_kequals7))
```

```
## [1] "rmse: 250420.258422278"
```

For kNN regression when k = 7 and normalizing the data

```r
print(paste('correlation:', cor_normalizing))
```

```
## [1] "correlation: 0.755511636636128"
```

```r
print(paste('mse:',mse_normalizing))
```

```
## [1] "mse: 0.0010228727696344"
```

```
print(paste('rmse:', rmse_normalizing))
```

```
## [1] "rmse: 0.0319823821757292"
```

For Decision tree

```
print(paste('correlation:', cor_decisiontree))
```

```
## [1] "correlation: 0.757303745801109"
```

```
print(paste('mse:',mse_decisiontree))
```

```
## [1] "mse: 57432543575.4491"
```

```
print(paste('rmse:', rmse_decisiontree))
```

```
## [1] "rmse: 239650.878520086"
```

Analysis of the result

For linear regression Linear regression works good for linear relationship. We determine the price of the house based first by using single predictor "sqft_above". Price is a dependent variable and square foot above is independent variable. For multivariable regression, there is addition of different predictors like "sqft_living", "sqft_above", "grade", "bathrooms". For multiple variable regression: price = w0 + w1 * sqft_living + w2 * sqft_above + w3 * grade + w4 * bathrooms

Our task is to find w0, w1, w2, w3, w4 in such a way that we minimize the rmse value and achieving the best line. For this we use gradient descent. Main idea is to put at first random value for each weight and updating the values till the cost function reaches minimum value.

For kNN regression kNN is a supervised machine learning algorith which says that similar things exist in close proximity. kNN uses the idea of similarity by finding the euclidian distance between each other. In kNN regression, we fit the training data, which is classified into groups. Now, when new datasets or test data is given, we can observe what group its nearest neighbors it belong to by finding the minimum euclidian distance. k in kNN regression is kept odd number. We can find the k in sucha a way that there is less error and high correlation, so that it will be good model.

For decision trees It recursively split the input observations into partitions until there is observations in a given partition. When we use linear regression model, our aim is to decrease the error over all the data, but in decision trees we want to minimize RSS within each region. We use top-down, greedy approach to partition the data. To start, all predictors are examined to see if they can make the good splits, and for each predictor the numerical value at which the split must be determined. First split will divide into two regions. It is divided till spliting threshold is reached.

Conclusion

By comparing with different algorithms, we found that decision trees algorithm better for these dataset. This might be because of missing features. We know that the price of the house does not only depend on the area how much it is occupied but also the locality where is it, at which state and many other factor. Since, the linear regression is mainly used for linear relationship, a price of the house is not linearly dependent with the predictors here. kNN didn't beat decision trees algorithm for this datasets, it might be because kNN is very sensitive for bad features. Chosing the other features might enhance the result of kNN.