Luis Pulido

Project 2

Dataset 1

This dataset contains house sale prices for King County, which includes Seattle. I want to see if I can create some models that will be useful in predicting the prices of houses around the area. It includes homes sold between May 2014 and May 2015.

This data was found on:

https://www.kaggle.com/harlfoxem/housesalesprediction/data (https://www.kaggle.com/harlfoxem/housesalesprediction/data)

```
house <- read.csv('/home/luis/R/kc_house_data.csv')
```

The R functions I used to explore the data set are dim(), head(), pairs(), names(), and str().

This set has 21613 rows which is perfect for this project.

```
dim(house)
```

```
## [1] 21613    21
```

Quick glance at some of the data in this set.

```
head(house, n =5)
```

```
##           id            date  price bedrooms bathrooms sqft_living
## 1 7129300520 20141013T000000 221900        3      1.00        1180
## 2 6414100192 20141209T000000 538000        3      2.25        2570
## 3 5631500400 20150225T000000 180000        2      1.00         770
## 4 2487200875 20141209T000000 604000        4      3.00        1960
## 5 1954400510 20150218T000000 510000        3      2.00        1680
##   sqft_lot floors waterfront view condition grade sqft_above sqft_basement
## 1     5650      1          0    0         3     7       1180             0
## 2     7242      2          0    0         3     7       2170           400
## 3    10000      1          0    0         3     6        770             0
## 4     5000      1          0    0         5     7       1050           910
## 5     8080      1          0    0         3     8       1680             0
##   yr_built yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 1     1955            0   98178 47.5112 -122.257          1340       5650
## 2     1951         1991   98125 47.7210 -122.319          1690       7639
## 3     1933            0   98028 47.7379 -122.233          2720       8062
## 4     1965            0   98136 47.5208 -122.393          1360       5000
## 5     1987            0   98074 47.6168 -122.045          1800       7503
```

List of the names

```
names(house)
```

```
##  [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"
```

id - a notation for a house date - Date house was sold price - Price is prediction target bedrooms - Number of Bedrooms/House bathrooms - Number of bathrooms/bedrooms sqft_living - square footage of the home sqft_lot - square footage of the lot floors - Total floors (levels) in house waterfront - House which has a view to a waterfront view - Has been viewed condition - How good the condition is (Overall) grade - overall grade given to the housing unit, based on King County grading system sqft_above - square footage of house apart from basement sqft_basement - square footage of the basement yr_built - Built Year yr_renovated - Year when house was renovated zipcode - zipcode of houses lat - Latitude coordinate long - Longitude coordinate sqft_living15 - Living room area in 2015(implies– some renovations) This might or might not have affected the lotsize area sqft_lot15 - lotSize area in 2015(implies– some renovations)

I now use this to begin see which algorithms would work best and what I want to remove that I don’t want in this data.

```
str(house)
```

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : Factor w/ 372 levels "20140502T000000",..: 165 221 291 2
21 284 11 57 252 340 306 ...
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ..
.
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 656
0 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ..
.
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 .
..
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 981
46 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 .
..
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570
...
```

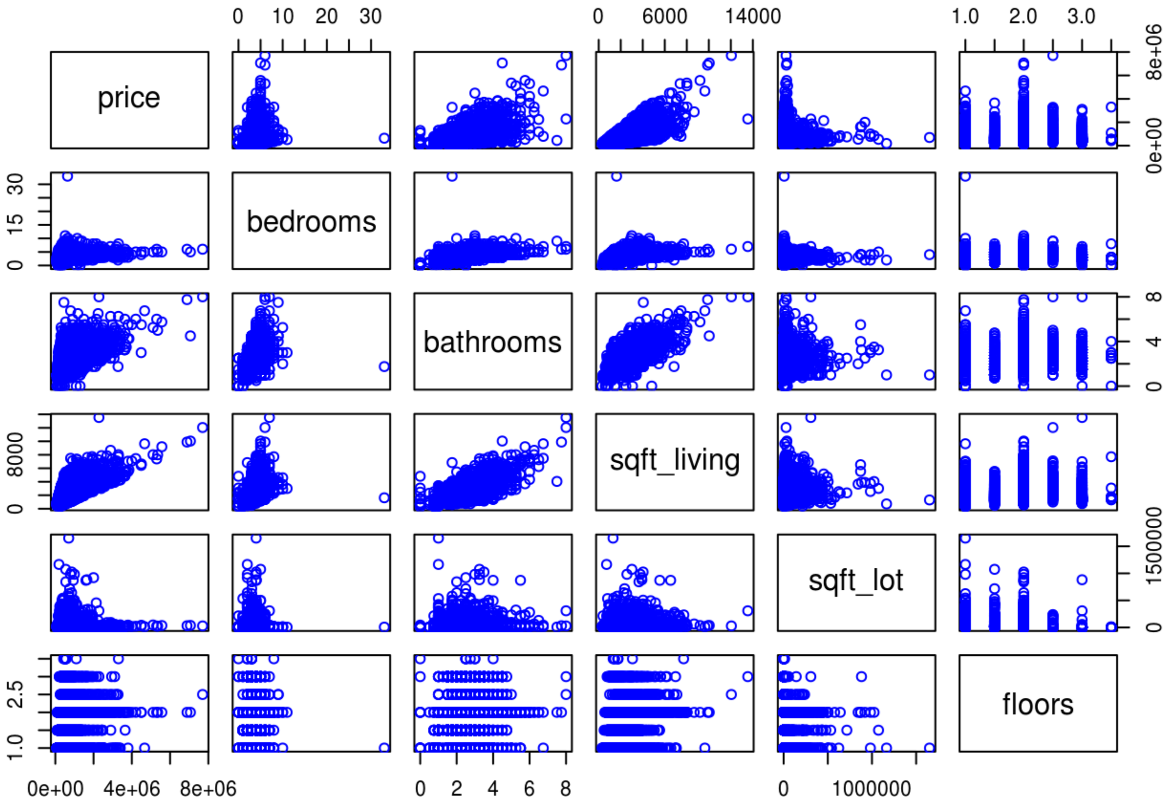
I start by getting rid of the id, date, lattitude, and longitute columns. They won’t be useful for what I have planned.

```
houses <- house[,-c(1,2,18,19)]
names(houses)
```

```
##  [1] "price"         "bedrooms"      "bathrooms"     "sqft_living"  
##  [5] "sqft_lot"      "floors"        "waterfront"    "view"         
##  [9] "condition"     "grade"         "sqft_above"    "sqft_basement"
## [13] "yr_built"      "yr_renovated"  "zipcode"       "sqft_living15"
## [17] "sqft_lot15"
```

A quick glance at this data. Price and sqft_lving seem to have a strong correlation. Bathrooms and sqft_living also seem to have a strong correlation. I think these would be great for linear regression.

```
pairs(houses[,c(1:6)], col="blue")
```



I split my data into tran and test sets here. I make sure of remove columns I don't want such as zipcode, sqft_above, and sqft_basement. sqft_basement was removed for having “NA” in the data

```
avgprice <- mean(houses$price)
avgprice
```

```
## [1] 540088.1
```

```
houses$avgprice <- ifelse(houses$price >= avgprice, 1, 0)
#A quick check to see if I did this correctly.
head(houses[c(1,18)], n = 5)
```

```
##    price avgprice
## 1 221900        0
## 2 538000        0
## 3 180000        0
## 4 604000        1
## 5 510000        0
```

```
houses$yr_built <- as.numeric(houses$yr_built)
houses$grade <- as.numeric(houses$grade)
houses$condition <- as.numeric(houses$condition)
houses$floors <- as.numeric(houses$floors)
houses$bedrooms <- as.numeric(houses$bedrooms)
houses$yr_renovated <- as.numeric(houses$yr_renovated)
houses$sqft_lot <- as.numeric(houses$sqft_lot)
houses$waterfront <- as.numeric(houses$waterfront)
houses$view <- as.numeric(houses$view)

set.seed(1234)
f <- sample(2, nrow(houses), replace=TRUE, prob=c(.75,.25))
train <- houses[f==1, -c(15,16,17,11,12)]
test <- houses[f==2, -c(15,16,17,11,12)]
```

Algorithm 1

Linear regression

Response:price

Predictors: all except avgprice

I decided to start with a linear regression since the data seems to be very easily linearly sperable. After running a summary it makes sense that bedrooms, bathrooms, and sqft_living would be significant to the price. People usually expect the more those increase the more expensive a home would be.

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

```
##
## Call:
## lm(formula = price ~ . - avgprice, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1358737  -110006    -8532    89749  4194651
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.278e+06  1.594e+05  39.394  < 2e-16 ***
## bedrooms     -4.174e+04  2.442e+03 -17.092  < 2e-16 ***
## bathrooms     5.120e+04  3.992e+03  12.827  < 2e-16 ***
## sqft_living   1.726e+02  3.791e+00  45.517  < 2e-16 ***
## sqft_lot     -2.711e-01  4.290e-02  -6.319 2.71e-10 ***
## floors        2.246e+04  3.988e+03   5.632 1.81e-08 ***
## waterfront    5.875e+05  2.158e+04  27.227  < 2e-16 ***
## view          4.722e+04  2.551e+03  18.515  < 2e-16 ***
## condition     1.999e+04  2.914e+03   6.859 7.18e-12 ***
## grade         1.232e+05  2.482e+03  49.627  < 2e-16 ***
## yr_built     -3.613e+03  8.172e+01 -44.212  < 2e-16 ***
## yr_renovated  9.239e+00  4.519e+00   2.045   0.0409 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 217700 on 16234 degrees of freedom
## Multiple R-squared:  0.6543, Adjusted R-squared:  0.6541
## F-statistic:  2793 on 11 and 16234 DF,  p-value: < 2.2e-16
```

This algorithm actually performed pretty well. Almost all of my predictors were very significant to the data. This model is able to explain 65% of the data due to the R^2 value.

```
pred <- predict(lm1, newdata=test)
cor(pred, test$price)
```

```
## [1] 0.8038731
```

```
residuals <- (pred - test$price)
mse <- mean(residuals^2)
mse
```

```
## [1] 45493137091
```

The model had a correlation of 80.39%. The reason being was I used all predictors.

Below are some of the plots I felt were interesting.

```
par(mfrow=c(2,2))
plot(price~bathrooms, data=houses, pch=2, col="purple", xlab="Bathrooms", yla
b="Prices", main="Housing Data")
abline(lm1, col="red")
```

```
## Warning in abline(lm1, col = "red"): only using the first two of 12
## regression coefficients
```

```
plot(price~sqft_living, data=houses, pch=2, col="purple", xlab="Living Sqft",
ylab="Prices", main="Housing Data")
abline(lm1, col="red")
```

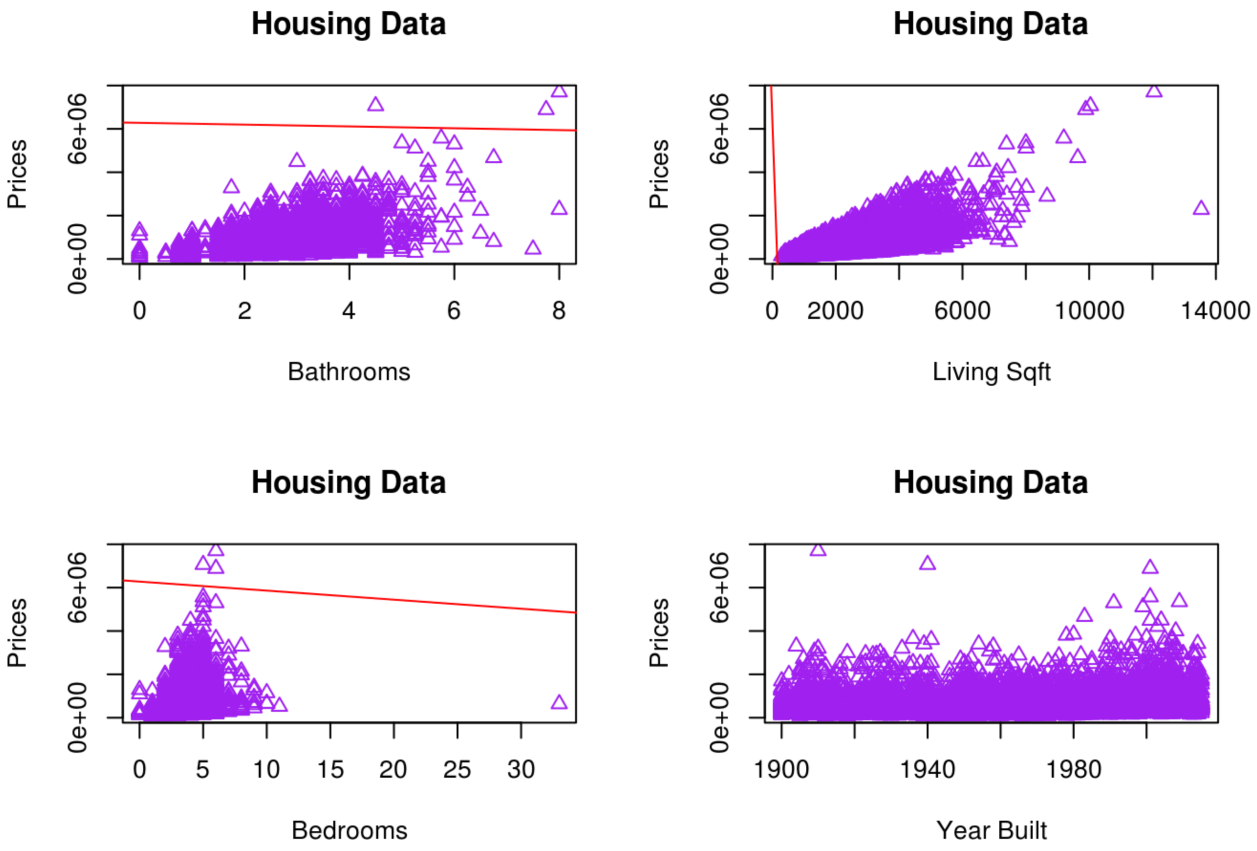
```
## Warning in abline(lm1, col = "red"): only using the first two of 12
## regression coefficients
```

```
plot(price~bedrooms, data=houses, pch=2, col="purple", xlab="Bedrooms", ylab=
"Prices", main="Housing Data")
abline(lm1, col="red")
```

```
## Warning in abline(lm1, col = "red"): only using the first two of 12
## regression coefficients
```

```
plot(price~yr_built, data=houses, pch=2, col="purple", xlab="Year Built", yla
b="Prices", main="Housing Data")
abline(lm1, col="red")
```

```
## Warning in abline(lm1, col = "red"): only using the first two of 12
## regression coefficients
```



```
#I had first plotted all the charts with each predictors and picked out the a
bove 4.
#plot(price~., data=houses)
#abline(lm1, col="red")
```

The linear model doesn’t seem to fit the data very well according to my plots. It seems the prices in this area seem to be pretty steady over the years which I found surprising. I became curious about the prices in the set. It seems most of the houses in this set are very expensive. Most houses have 2-4 bathrooms and 5 bedrooms. They are between 2000-6000 living square feet large.

```
median(houses$price)
```

```
## [1] 450000
```

```
mean(houses$price)
```

```
## [1] 540088.1
```

A second linear regression model to see if I can beat the previous one using less predictors.

Response: price

Predictors: Interaction of bathrooms & sqft_living, yr_built, grade, condition, bedrooms & bathrooms interaction, floors, yr_built & yr_renovated interaction, sqft_lot, waterfront

```
lm2 <-lm(price~bathrooms*sqft_living+yr_built+grade+condition+bedrooms*bathro
oms+floors+yr_built*yr_renovated+sqft_lot+waterfront-bathrooms-bedrooms-sqft_
living, data=train)
summary(lm2)
```

```
##
## Call:
## lm(formula = price ~ bathrooms * sqft_living + yr_built + grade +
##     condition + bedrooms * bathrooms + floors + yr_built * yr_renovated +
##     sqft_lot + waterfront - bathrooms - bedrooms - sqft_living,
##     data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3724362  -106888   -10474    84864  3219498
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            6.374e+06  1.484e+05  42.957  < 2e-16 ***
## yr_built              -3.635e+03  7.606e+01 -47.791  < 2e-16 ***
## grade                  1.347e+05  2.168e+03  62.121  < 2e-16 ***
## condition              2.760e+04  2.809e+03   9.825  < 2e-16 ***
## floors                 2.723e+04  3.789e+03   7.188 6.88e-13 ***
## yr_renovated          -2.427e+03  3.494e+02  -6.946 3.91e-12 ***
## sqft_lot              -2.767e-01  4.133e-02  -6.694 2.24e-11 ***
## waterfront             6.995e+05  1.938e+04  36.090  < 2e-16 ***
## bathrooms:sqft_living  5.582e+01  8.937e-01  62.457  < 2e-16 ***
## bathrooms:bedrooms    -1.510e+04  7.638e+02 -19.767  < 2e-16 ***
## yr_built:yr_renovated  1.259e+00  1.800e-01   6.994 2.78e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 210300 on 16235 degrees of freedom
## Multiple R-squared:  0.6774, Adjusted R-squared:  0.6772
## F-statistic:  3409 on 10 and 16235 DF,  p-value: < 2.2e-16
```

```
prednew <- predict(lm2, newdata=test)
cor(prednew, test$price)
```

```
## [1] 0.8185723
```

```
residuals2 <- (prednew - test$price)
mse2 <- mean(residuals2^2)
mse2
```

```
## [1] 42302624392
```

This new model has a correlation of 81.86%. It also explains 67% of the data.

Algorithm 2

Decision Trees

Response: price

Predictors: grade, view, sqft_living, yr_built

For my second algorithm I wanted to try a regression decision tree. I decided to split the data into train and test sets again.

```
library(tree)
set.seed(1234)
train2 = sample(1:nrow(houses), nrow(houses)/2)

tree.houses = tree(price~grade+view+sqft_living+yr_built-avgprice, houses, su
bset = train2)
#tree.houses = tree(price~., houses)
summary(tree.houses)
```

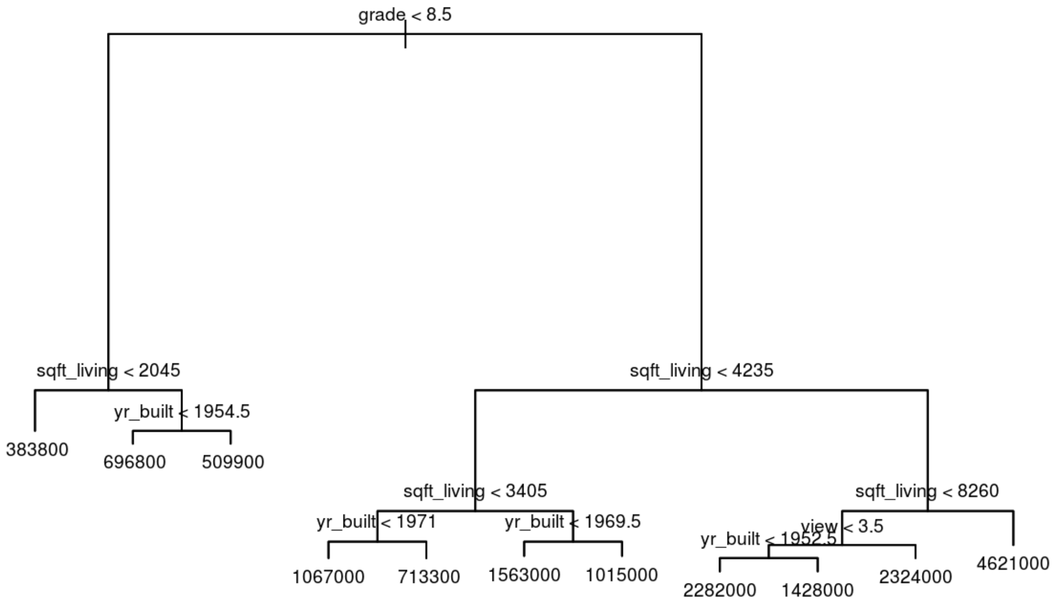
```
##
## Regression tree:
## tree(formula = price ~ grade + view + sqft_living + yr_built -
##     avgprice, data = houses, subset = train2)
## Number of terminal nodes:  11
## Residual mean deviance:  5.25e+10 = 5.667e+14 / 10800
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2341000  -133800   -28760        0    98570  3079000
```

I found it intresting that grade(quality of the house), sqft_living, and if it had a view were used to construct this tree.

If the grade of the house was less than 8.5 with a sqft living space greater than 2045 and build before 1954 the price predicted is $696,750.90

If the house or grade was rated over 8.5 with a living space greater than 8260 square feet the price predicted is $4,621,200.00

```
plot(tree.houses)
text(tree.houses, cex=.6, pretty=1)
```



```
tree.houses
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 10806 1.416e+15  541300
##    2) grade < 8.5 8660 3.281e+14  438200
##      4) sqft_living < 2045 5829 1.320e+14  383800 *
##      5) sqft_living > 2045 2831 1.433e+14  550300
##       10) yr_built < 1954.5 612 4.562e+13  696800 *
##       11) yr_built > 1954.5 2219 8.089e+13  509900 *
##    3) grade > 8.5 2146 6.239e+14  957400
##      6) sqft_living < 4235 1878 2.636e+14  855200
##       12) sqft_living < 3405 1362 1.193e+14  765600
##         24) yr_built < 1971 201 2.582e+13 1067000 *
##         25) yr_built > 1971 1161 7.205e+13  713300 *
##       13) sqft_living > 3405 516 1.045e+14 1092000
##         26) yr_built < 1969.5 72 1.926e+13 1563000 *
##         27) yr_built > 1969.5 444 6.661e+13 1015000 *
##      7) sqft_living > 4235 268 2.030e+14 1674000
##       14) sqft_living < 8260 263 1.399e+14 1618000
##         28) view < 3.5 233 9.925e+13 1527000
##           56) yr_built < 1952.5 27 8.233e+12 2282000 *
##           57) yr_built > 1952.5 206 7.363e+13 1428000 *
##         29) view > 3.5 30 2.379e+13 2324000 *
##       15) sqft_living > 8260 5 1.887e+13 4621000 *
```

```
yhat = predict(tree.houses, newdata=houses[-train2,])
house.test = houses[-train2, "price"]
mse <-mean((yhat - house.test)^2)
mse
```

```
## [1] 58433075757
```

```
sqrt(mse)
```

```
## [1] 241729.3
```

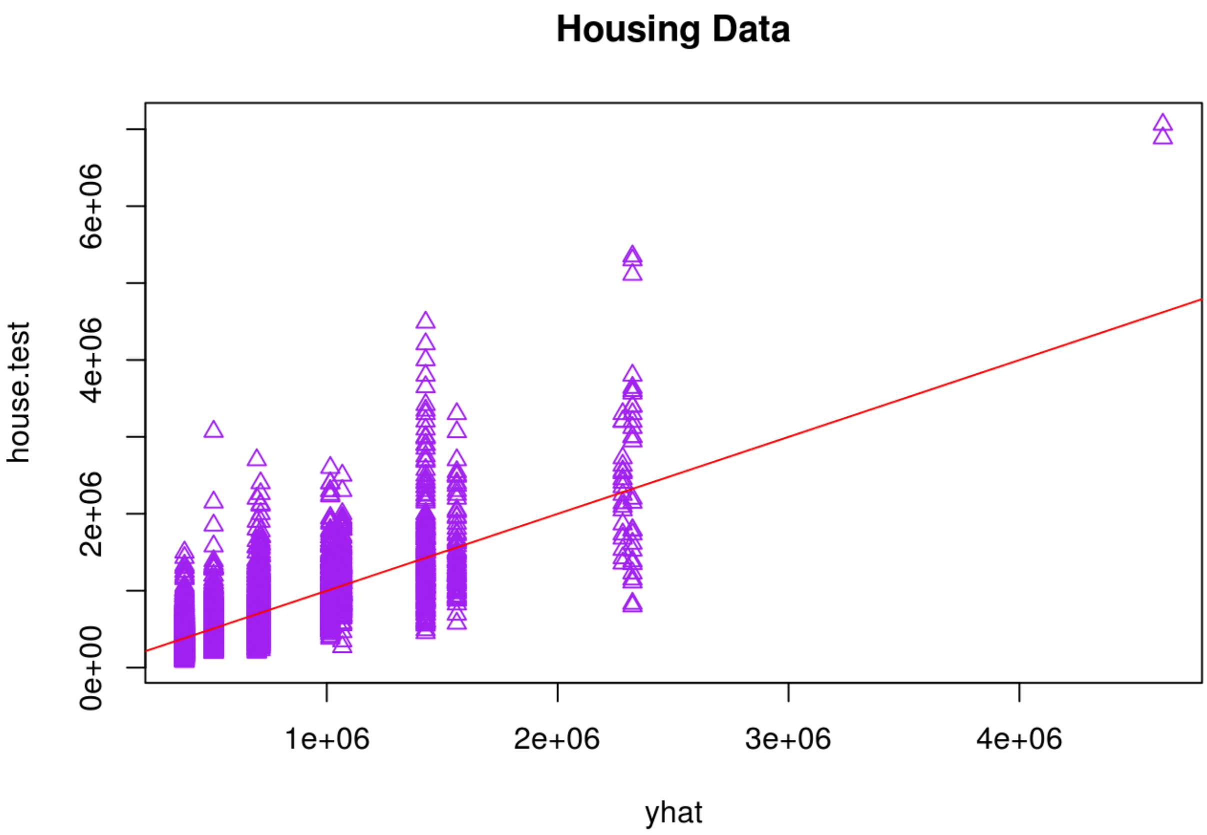
I thought this was very odd but considering these are expensive houses it makes a little sense. We have houses in the set worth over $7 million so being 241,729.34 off isn’t

too outrageous. Still this is pretty bad.

```
max(houses$price)
```

```
## [1] 7700000
```

```
plot(yhat, house.test, pch=2, col="purple", xlab="yhat", ylab="house.test", m
ain="Housing Data")
abline(0,1, col="red")
```



I decided to try a random forest to see if I could improve the really high value of the mse.

Random forest

Response: price

Predictors: All predictors except avgprice

```r
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```r
house.price <- randomForest(price~.-avgprice, data=houses, mtry=3, importance
=TRUE, na.action=na.omit)
house.price
```

```
##
## Call:
##  randomForest(formula = price ~ . - avgprice, data = houses, mtry = 3,
importance = TRUE, na.action = na.omit)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 27763760199
##                     % Var explained: 79.4
```

```r
yhat2 = predict(house.price, newdata=houses[-train2,])
house.test2 = houses[-train2, "price"]
mserand <-mean((yhat2 - house.test2)^2)
mserand
```

```
## [1] 6159510870
```

```r
sqrt(mserand)
```

```
## [1] 78482.55
```

Even with a random forest this leads us to predictions that are $78,482.55 off. This is much much better than before but still too high.

Algorithm 3

Neural Network

I wanted to use a neural network. I wanted to construct this as a classification neural net to make it easier to predict on it. I decided to use the mean as my metric for the new column I created because I was more intrested in the average price of a house in this set. I had originally planned to only use regression here but I was having some difficulties that I eventually solved so I kept both models. Here I now use avgprice I created before splitting the data into train and test sets.

Now using my new attribute as a response I was able to create a neural network with 1 hidden layers. I chose to go with 1 layer due to how linear the data seemed to be.

Classification Neural Net

Response: avgprice

Predictors: bedrooms, sqft_living, yr_built, condition

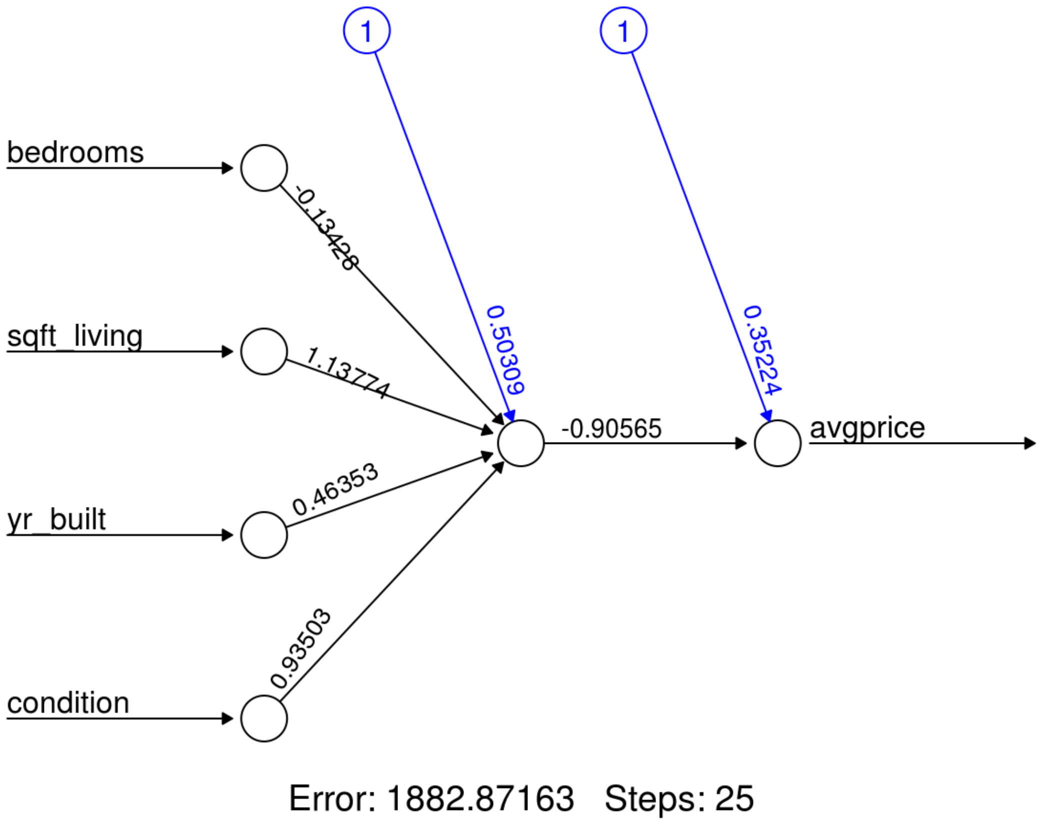
```
library(neuralnet)
#nn1 <- neuralnet(price~bathrooms+floors+sqft_living+grade+condition, data=tr
ain)
#plot(nn1)

housenet <- neuralnet(avgprice~bedrooms+sqft_living+yr_built+condition, train
, hidden=1, lifesign="minimal", linear.output=FALSE, threshold=0.1)
```

```
## hidden: 1    thresh: 0.1    rep: 1/1    steps:      25   error: 1882.87163
time: 0.14 secs
```

I then plotted the network. I used predictors I felt are important to the price

```
plot(housenet, rep="best")
```



I will now predict on this network.

```
temp_test <- subset(test, select=c("bedrooms", "condition", "sqft_living", "yr_built"))
housenet.results <- compute(housenet, temp_test)
results <- data.frame(actual=test$avgprice, prediction=housenet.results$net.result)
```

```
results$round <- round(results$prediction)
mean(results$round == results$actual)
```

```
## [1] 0.6295882243
```

Using a neural network while very exciting wasn’t as helpful as I would have expected. Even when given what I assumed would be the easier task of 1 or 0 as the response my model only was able to predict with 62.96% accuracy.

Here I built a regression neural network. I also normalized the data and retrained it.

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

house_norm <- as.data.frame(lapply(houses, normalize))

set.seed(1234)
i <- sample(1:nrow(houses), 0.70*nrow(houses), replace=FALSE)
trainn <- house_norm[i,]
testn <- house_norm[-i,]
```

Regression Neural Net

Response: price

Predictors: bathrooms, sqft_living, condition, yr_built

```
nnh <- neuralnet(price~bathrooms+sqft_living+condition+yr_built, data=trainn,
hidden=3)
plot(nnh)
```

```
resultnh <- compute(nnh, testn[,c(3,4,9,13)])
prednh <- resultnh$net.result
cor(prednh, testn$price)
```

```
##              [,1]
## [1,] 0.7572895386
```

```
residualsnn <- (prednh - testn$price)
mse2 <- mean(residualsnn^2)
mse2
```

```
## [1] 0.000963440485
```

Using a neural network for regression with price as my predictor I was able to obtain a correlation of 76.08%.

Algorithm 4

kNN for Regression

Response: price

Predictors: bedrooms, bathrooms, sqft_living, floors, condition, grade

Finally I wanted to try nearest neighbor but instead of classification I wanted to do it using regression to see if I could beat my linear model. I used price as my target column here again. After messing with the amount of neighbors I found 10 to be optimal and giving me the best accuracy.

```
library(caret)
```

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

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

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
fit <- knnreg(train[,c(2,3,4,6,9,10)], train[,c(1)], k=10)
predictions <- predict(fit, test[,c(2,3,4,6,9,10)])
cor(predictions, test$price)
```

```
## [1] 0.7324963143
```

```
residualsknn <- (predictions - test$price)
mseknn <- mean(residualsknn^2)
mseknn
```

```
## [1] 59680314759
```

The kNN for regression model had a correlation of 73.25%. Perhaps scaling even further would improve this. While less predictors improves accuracy I found that using all the ones I selected gave me the highest correlation prediction.

In Conclusion

Since I wanted to predict a numeric predictor “price” I chose linear algorithms. I ran Linear Regression, Decision Tree including a Random Forest, Neural Network for classification and regression, and kNN for regression.

Linear Regression had 81.86% Neural Network had 76.08% kNN for Regression had 73.25% Decision tree $78,482.55 off

MSE from algorithms:

45,493,137,091 lm1 42,302,624,392 lm2 58,433,075,757 decision tree 6,128,176,116 random forest 59,680,314,759 knn 0.000963440485 neural network

Based on my MSE my neural network did the best.

When comparing my correlations the simpler algorithm did the best here. My second linear regression model did the best with 81.85%. My normalized neural network did second best with 76.08%. Third best was kNN for regression with 73.25%. The worst model was my decision tree. However after using a random forest the mse was now the second lowest. It is able to make predictions that are $78,482.55 off. Neural network had the best mse.

A linear regression model did best here because as we saw above in the scatterplot most of the data was linearly seperable. There were a lot of strong correlations. Linear Regression was able to account for interactions which helped boost it’s accuracy. It’s also the reason why my regression neural net came in second.

The feature selection I tried was mostly using features I felt had strong correlations or would make sense to influence the price such as the number of bathrooms and size of living space. I usually start my models by using “.” all predictors and working from there .

The metrics I used to evaluate were predicting on the correlations in linear regression. Predicting on the yhat and finding the square root of the mse in my decision trees. In my regression neural net I predicted based on the correlations again. I did the same thing in my nearest neighbor algorithm.

What I learned from the data is that sometimes simple algorithms work best. When you have a huge dataset sometimes the simplest thing can give you the best results. Where you collect data also makes a difference. When I ran “unique(houses$price)” in the console I learned there were 4,028 unique prices in the set. A reason for why the accuracy seems to be low here is in my opinion probably due to how many unique prices are in this set.