

# Sunayu Tech Task

## Multivariate Linear Regression: Predicting House Prices

Michael Vatt

08 Dec 20

## Contents

<b>Overview of Multivariate Linear Regression Housing Prediction</b>	<b>2</b>
Objective . . . . .	3
Dataset . . . . .	3
<b>Familiarization</b>	<b>3</b>
Preprocessing . . . . .	5
<b>Methods and Analysis</b>	<b>6</b>
First Model . . . . .	6
Exploratory Data Analysis . . . . .	7
Second Model . . . . .	16
Influential Data Points . . . . .	17
Third Model . . . . .	19
<b>Results</b>	<b>20</b>
Accuracy of the Models . . . . .	20
<b>Conclusion</b>	<b>22</b>

```
# Create a Function to Check for Installed Packages and Install if They Are Not Installed

install <- function(packages){
  new.packages <- packages[!(packages %in% installed.packages() [, "Package"])]
  if (length(new.packages))
    install.packages(new.packages, dependencies = TRUE, repos = "http://cran.us.r-project.org")
  sapply(packages, require, character.only = TRUE)
```

```

}

# Install

packages <- c("caTools", "car", "caret", "cluster", "Clustering", "corpus", "corrplot", "data.table",
            "dendextend", "doParallel", "dplyr", "e1071", "factoextra", "FactoMineR", "fpc",
            "GGally", "ggplot2", "ggthemes", "gridExtra", "kableExtra", "knitr", "ldatuning",
            "magrittr", "mclust", "NbClust", "petro.One", "plotly", "plotrix", "qdap", "qdapTools",
            "quanteda", "randomForest", "readxl", "RColorBrewer", "rlist", "RWeka", "scales",
            "SentimentAnalysis", "sentimentr", "SnowballC", "stm", "stringr", "syuzhet",
            "tensorflow", "tidyverse", "tidytext", "tidyverse", "tm", "topicmodels", "viridisLite",
            "wordcloud", "xlsx", "zoo")

install(packages)

```

```

# Call the installed packages

library(plyr) # plyr is required to be loaded before dplyr or issues may arise
library(dplyr) # dplyr needed for efficient loading of loadApp()

loadApp <- function() {

my_library <- c("caTools", "car", "caret", "cluster", "Clustering", "corpus", "corrplot", "data.table",
              "dendextend", "doParallel", "dplyr", "e1071", "factoextra", "FactoMineR", "fpc",
              "GGally", "ggplot2", "ggthemes", "gridExtra", "kableExtra", "knitr", "ldatuning",
              "magrittr", "mclust", "NbClust", "petro.One", "plotly", "plotrix", "qdap", "qdapTools",
              "quanteda", "randomForest", "readxl", "RColorBrewer", "rlist", "RWeka", "scales",
              "SentimentAnalysis", "sentimentr", "SnowballC", "stm", "stringr", "syuzhet",
              "tensorflow", "tidyverse", "tidytext", "tidyverse", "tm", "topicmodels", "viridisLite",
              "wordcloud", "xlsx", "zoo")

install.lib <- my_library[!my_library %>% installed.packages()]

for(lib in install.lib) install.packages(lib, dependencies = TRUE)

sapply(my_library, require, character = TRUE)
}

loadApp()

```

## Overview of Multivariate Linear Regression Housing Prediction

This data set includes the following variables:

Variables
Income
House_Age
Number_Rooms
Number_Bedrooms
Area_Population
Price
Address

## Objective

This was a task issued by Sunayu as an evaluation of approach, style, ability, and performance. The R script will analyze a house price data set to predict the price of a house via different independent variables. It will move through data familiarization, preprocessing, splitting the training and testing data, exploratory data analysis, model evaluation, predictions, and results. The R Markdown report is the same script, maybe a little more refined, prettier, and a one stop shop to read the results.

## Dataset

```
split <- detectCores(TRUE)
cl <- makePSOCKcluster(split)
registerDoParallel(cl)
```

```
dataset <- read.csv("https://raw.githubusercontent.com/huzaifsayed/Linear-Regression-Model-for-House-Pr...
```

## Familiarization

Let's take an initial look to familiarize with the data set and to ensure it was properly read in.

```
class(dataset) # Need to coerce into data frame if it is not
## [1] "data.frame"

dim(dataset) # For familiarity and a quick visualization to determine if something went wrong
## [1] 5000    7

summary(dataset) # Does anything look strange? Any variable(s) requiring pre-processing?
```

```

##  Avg..Area.Income Avg..Area.House.Age Avg..Area.Number.of.Rooms
##  Min.    : 17797   Min.    :2.644      Min.    : 3.236
##  1st Qu.: 61481   1st Qu.:5.322      1st Qu.: 6.299
##  Median  : 68804   Median  :5.970      Median  : 7.003
##  Mean    : 68583   Mean    :5.977      Mean    : 6.988
##  3rd Qu.: 75783   3rd Qu.:6.651      3rd Qu.: 7.666
##  Max.    :107702   Max.    :9.519      Max.    :10.760
##
##  Avg..Area.Number.of.Bedrooms Area.Population      Price
##  Min.    :2.000              Min.    : 172.6     Min.    : 15939
##  1st Qu.:3.140              1st Qu.:29403.9   1st Qu.: 997577
##  Median  :4.050              Median  :36199.4   Median  :1232669
##  Mean    :3.981              Mean    :36163.5   Mean    :1232073
##  3rd Qu.:4.490              3rd Qu.:42861.3   3rd Qu.:1471210
##  Max.    :6.500              Max.    :69621.7   Max.    :2469066
##
##  Address
##  000 Adkins Crescent\nSouth Teresa, AS 49642-1348:  1
##  000 Todd Pines\nAshleyberg, KY 90207-1179       :  1
##  001 Steve Plaza\nJessicastad, UT 25190        :  1
##  0010 Gregory Loaf\nSouth Ericfort, VA 34651-0718:  1
##  00149 Raymond Knolls\nNew Jason, UT 75026       :  1
##  002 Katherine Flat\nHartmanland, AZ 37973-3049  :  1
##  (Other)                                         :4994

```

```
str(dataset) # There are 6 variables of numerical and 1 variable factor
```

```

## 'data.frame': 5000 obs. of 7 variables:
## $ Avg..Area.Income      : num  79545 79249 61287 63345 59982 ...
## $ Avg..Area.House.Age   : num  5.68 6 5.87 7.19 5.04 ...
## $ Avg..Area.Number.of.Rooms: num  7.01 6.73 8.51 5.59 7.84 ...
## $ Avg..Area.Number.of.Bedrooms: num  4.09 3.09 5.13 3.26 4.23 4.04 3.41 2.42 2.3 6.1 ...
## $ $ Area.Population     : num  23087 40173 36882 34310 26354 ...
## $ $ Price                : num  1059034 1505891 1058988 1260617 630943 ...
## $ $ Address              : Factor w/ 5000 levels "000 Adkins Crescent\nSouth Teresa, AS 49642-1348",...
```

```
head(dataset) # Numerical variables need scaling? Should we not include any variables?
```

```

##  Avg..Area.Income Avg..Area.House.Age Avg..Area.Number.of.Rooms
##  1            79545.46          5.682861          7.009188
##  2            79248.64          6.002900          6.730821
##  3            61287.07          5.865890          8.512727
##  4            63345.24          7.188236          5.586729
##  5            59982.20          5.040555          7.839388
##  6            80175.75          4.988408          6.104512
##
##  Avg..Area.Number.of.Bedrooms Area.Population      Price
##  1                      4.09          23086.80 1059033.6
##  2                      3.09          40173.07 1505890.9
##  3                      5.13          36882.16 1058988.0
##  4                      3.26          34310.24 1260616.8
##  5                      4.23          26354.11 630943.5
##  6                      4.04          26748.43 1068138.1
##
##  Address

```

```

## 1 208 Michael Ferry Apt. 674\nLaurabury, NE 37010-5101
## 2 188 Johnson Views Suite 079\nLake Kathleen, CA 48958
## 3 9127 Elizabeth Stravenue\nDanieltown, WI 06482-3489
## 4                               USS Barnett\nFPO AP 44820
## 5                               USNS Raymond\nFPO AE 09386
## 6 06039 Jennifer Islands Apt. 443\nTracyport, KS 16077

colnames(dataset) <- c("Income", "House_Age", "Number_Rooms", "Number_Bedrooms", "Area_Population",
                      "Price", "Address") # For easier handling and better visualization
names(dataset) # Ensure that the variable names changed

## [1] "Income"          "House_Age"        "Number_Rooms"      "Number_Bedrooms"
## [5] "Area_Population" "Price"            "Address"

```

## Preprocessing

Here, we will limit our regression model to the first 6 variable columns and remove *Address* for the predictions. *Address* would require encoding but since it would create 1 dummy variable per observation (5000), encoding would be trivial and inconsequential for a prediction.

```

# Let's Check for Missing Data

data.frame(count_na_values <- colSums(is.na(dataset))) # Need to appraise if any greater than 0.

##           count_na_values....colSums.is.na.dataset..
## Income                           0
## House_Age                         0
## Number_Rooms                      0
## Number_Bedrooms                   0
## Area_Population                  0
## Price                            0
## Address                          0

# Split the Data set into a Training Set and Test Set

set.seed(1234)
val_ind <- createDataPartition(dataset$Price, times = 1, p = 0.2, list = FALSE) # Split 80/20
train_set <- dataset[-val_ind,] # 4000 observations for training data
test_set <- dataset[val_ind,] # 1000 observations to test upon

# Let's Make Sure the Split Operated Correctly

dim(train_set); dim(test_set) # 80% and 20% of the data set respectively

## [1] 4000    7

## [1] 1000    7

```

```

class(train_set); class(test_set) # Should be data.frames

## [1] "data.frame"

## [1] "data.frame"

# Remove Address From Data set for Prediction

train_set <- train_set[, (1:6)] # Only keep numerical variables significant to the prediction
test_set <- test_set[, (1:6)] # Only keep numerical variables significant to the prediction
names(train_set); names(test_set) # Ensure proper variables were kept

## [1] "Income"           "House_Age"        "Number_Rooms"      "Number_Bedrooms"
## [5] "Area_Population" "Price"

## [1] "Income"           "House_Age"        "Number_Rooms"      "Number_Bedrooms"
## [5] "Area_Population" "Price"

```

## Methods and Analysis

### First Model

```

## Model Will Not Need to Perform Feature Scaling - Linear Regression Function lm()
## Handles that Inherently

# Fitting Multiple Linear Regression to the Training Set

model1 <- lm(formula = Price ~ ., data = train_set) # Price is the dependent variable as a function
                                                       # of all dependent variables via '.'

## 
## Call:
## lm(formula = Price ~ ., data = train_set)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -327301 -70435     367   68818  346229 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.630e+06 1.890e+04 -139.151 <2e-16 ***
## Income       2.157e+01 1.508e-01  143.026 <2e-16 ***
## House_Age    1.649e+05 1.601e+03  103.047 <2e-16 ***
## Number_Rooms 1.211e+05 1.768e+03   68.498 <2e-16 ***
## Number_Bedrooms 1.696e+03 1.458e+03    1.163   0.245  
## Area_Population 1.508e+01 1.607e-01   93.845 <2e-16 ***

```

```

## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 100600 on 3994 degrees of freedom
## Multiple R-squared:  0.9191, Adjusted R-squared:  0.919
## F-statistic:  9079 on 5 and 3994 DF,  p-value: < 2.2e-16

## We see that Number_Bedrooms is not statistically significant
## Would removing this variable aid prediction accuracy?

# Using Fitted model1 for Predicting the Test Set Results

y_pred1 <- predict(model1, newdata = test_set)
visual_comparison <- data.frame(test_set$Price, y_pred1, residuals = test_set$Price - y_pred1)
visual_comparison[1:10,]

##      test_set.Price    y_pred1  residuals
## 18      528485.2  490640.3   37844.98
## 29     1240763.8 1405751.4  -164987.61
## 34      534305.1  614277.9   -79972.79
## 38     1081150.1 1167072.0   -85921.85
## 45     1153871.5  944404.6   209466.87
## 58     1186688.5 1009474.6   177213.92
## 61     1111085.0  935228.2   175856.81
## 71     1146532.5 1052267.4   94265.01
## 75     1534479.9 1357417.2   177062.70
## 78     1204598.0 1349876.4  -145278.38

```

Despite the model returning statistically significant p-value and R-squared values, we can see the model did not fit quite well on the test data set. Many of the predicted values have a noticeable difference. It is possible to make the model more robust and vigorous through more advanced models. Checking for multicollinearity may also help improve the algorithm.

## Exploratory Data Analysis

```

# Let's try to Evaluate Associations between Variables

correlation <- cor(train_set)
corrplot(correlation, method = "color")

```



```

## Price is positively correlated with Income, House_Age, Number_Rooms, and Area_Population
## Price has a slight positive correlation with Number_Bedrooms

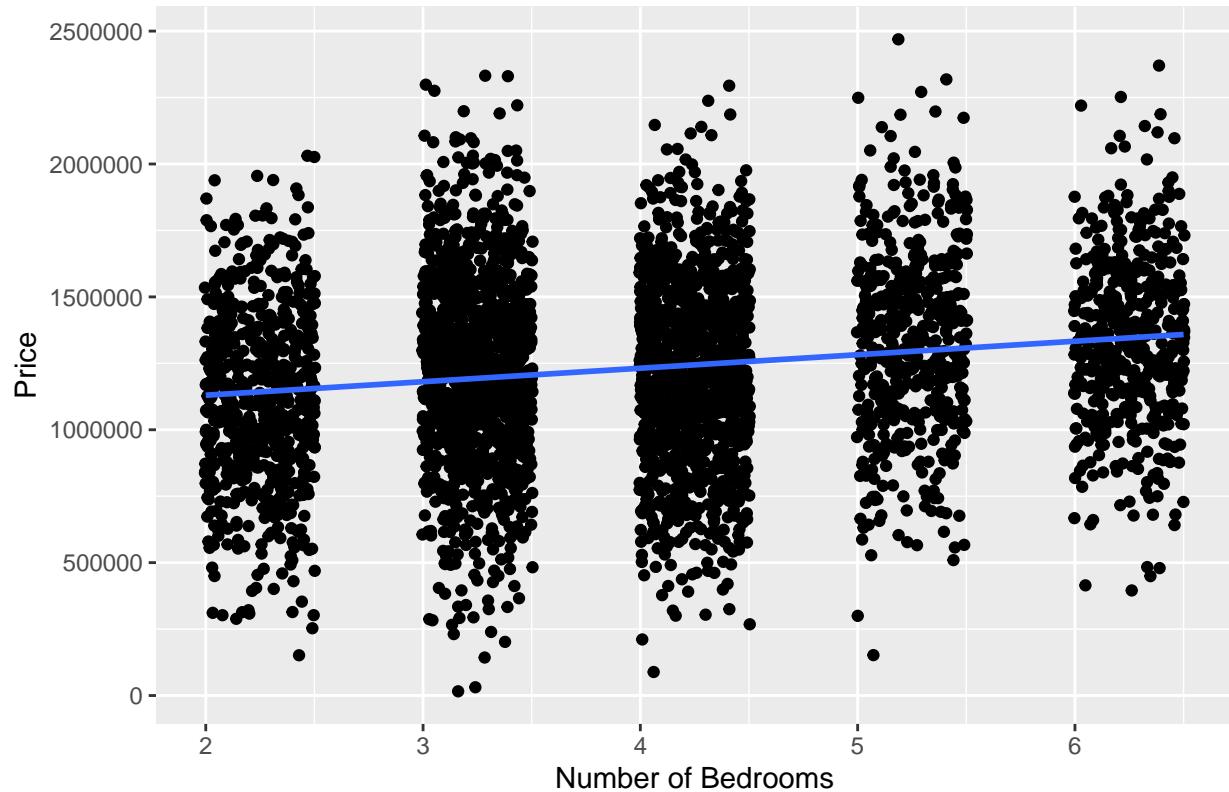
# Let's Make Scatter Plots to Determine the Relationship(s) between the Variables

bedroom_plot <- ggplot(data = train_set, aes(x = Number_Bedrooms, y = Price)) + geom_jitter() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Scatter plot of Bedrooms and Price", x = "Number of Bedrooms", y = "Price")

bedroom_plot # We see that the relationship is linear

```

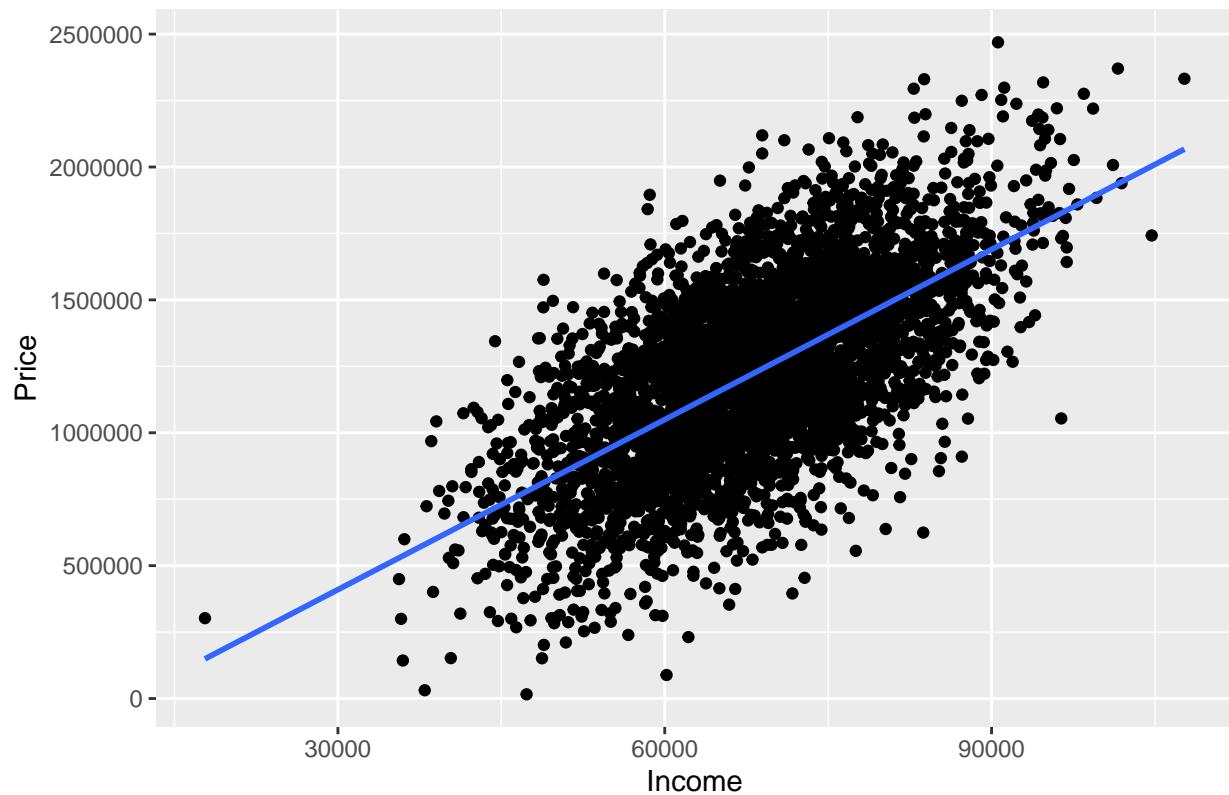
Scatter plot of Bedrooms and Price



```
income_plot <- ggplot(data = train_set, aes(x = Income, y = Price)) + geom_jitter() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Scatter plot of Income and Price", x = "Income", y = "Price")

income_plot # We see that the relationship is linear
```

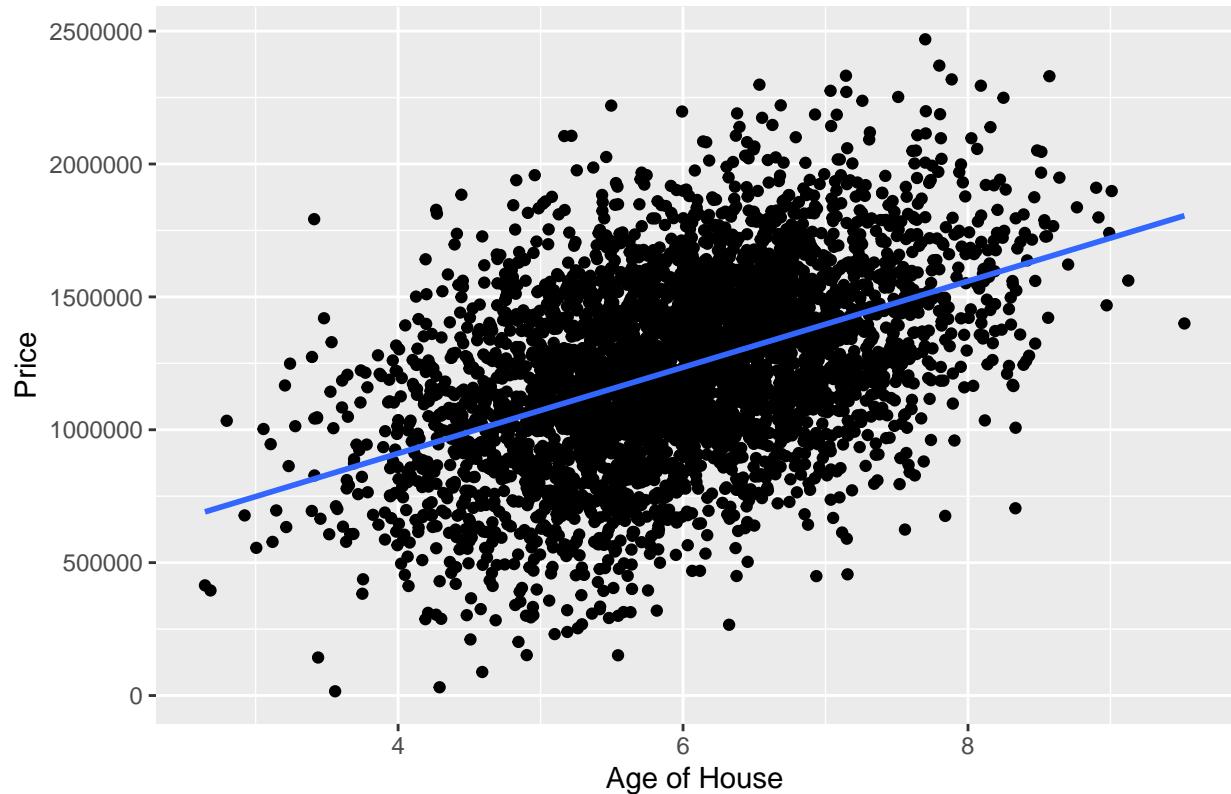
Scatter plot of Income and Price



```
age_plot <- ggplot(data = train_set, aes(x = House_Age, y = Price)) + geom_jitter() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Scatter plot of House Age and Price", x = "Age of House", y = "Price")

age_plot # We see that the relationship is linear
```

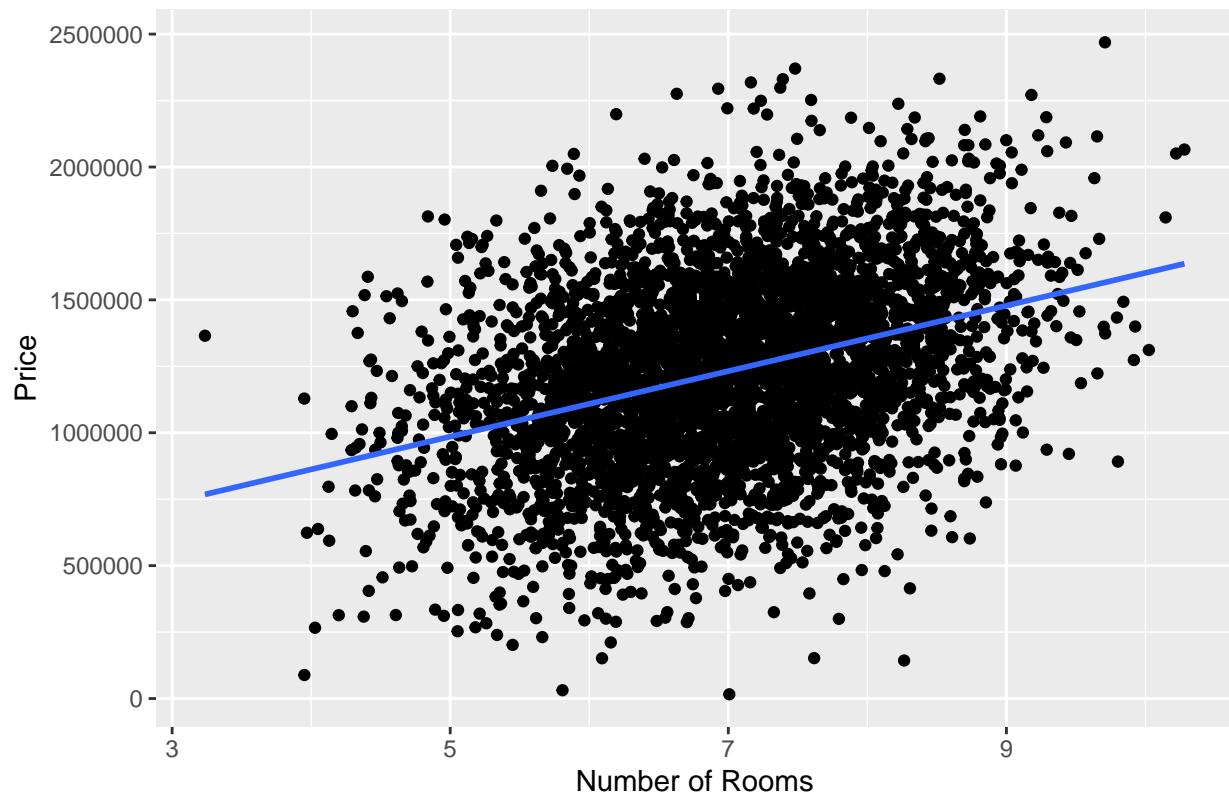
Scatter plot of House Age and Price



```
room_plot <- ggplot(data = train_set, aes(x = Number_Rooms, y = Price)) + geom_jitter() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Scatter plot of Rooms and Price", x = "Number of Rooms", y = "Price")

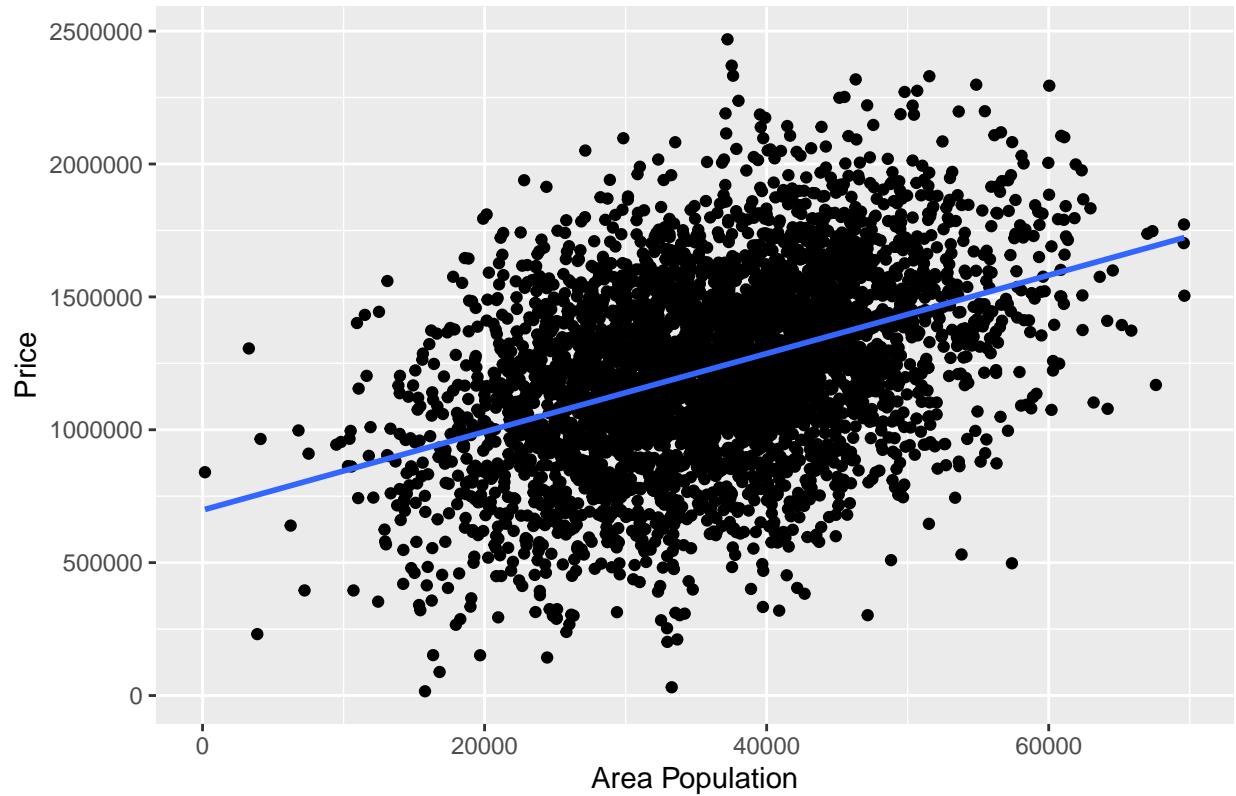
room_plot # We see that the relationship is linear
```

Scatter plot of Rooms and Price

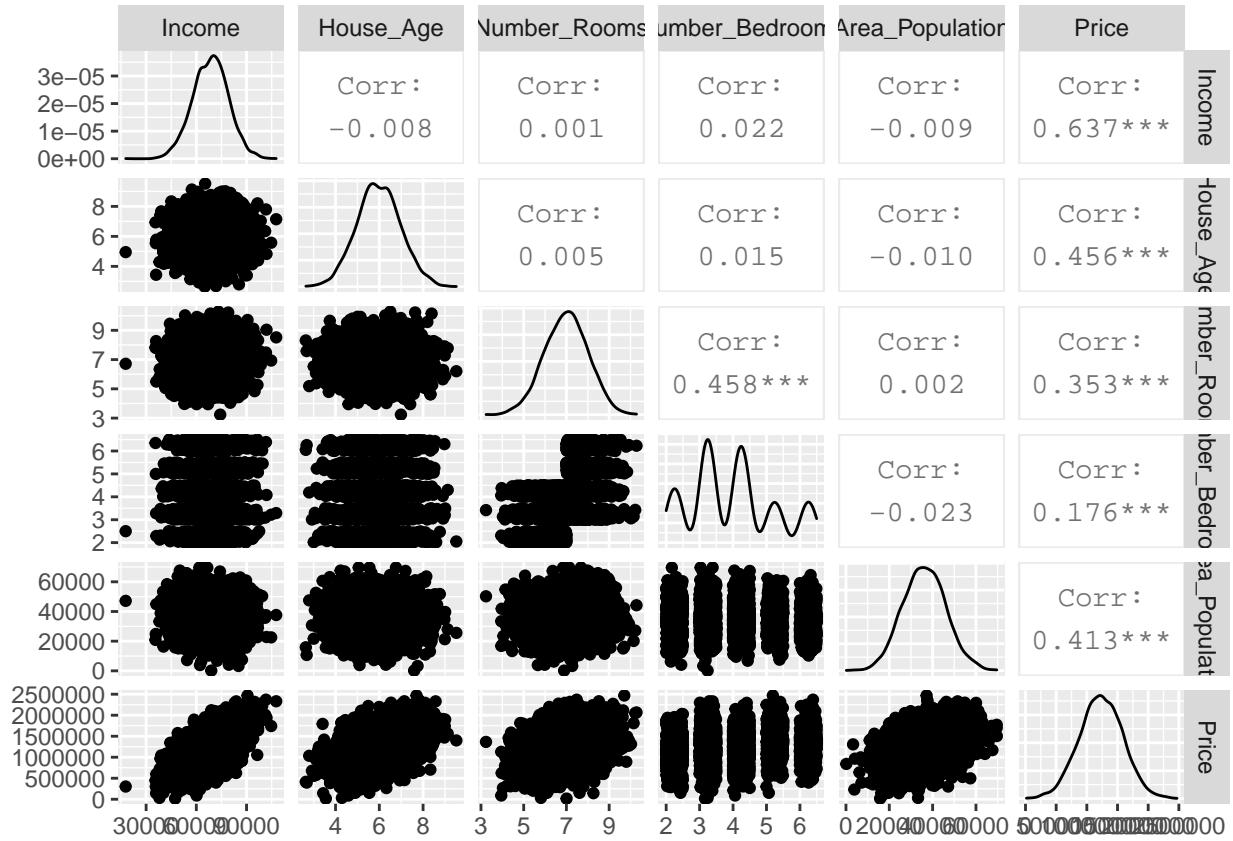


```
pop_plot <- ggplot(data = train_set, aes(x = Area_Population, y = Price)) + geom_jitter() +  
  geom_smooth(method = "lm", se = FALSE) +  
  labs(title = "Scatter plot of Area Population and Price", x = "Area Population", y = "Price")  
  
pop_plot # We see that the relationship is linear
```

Scatter plot of Area Population and Price



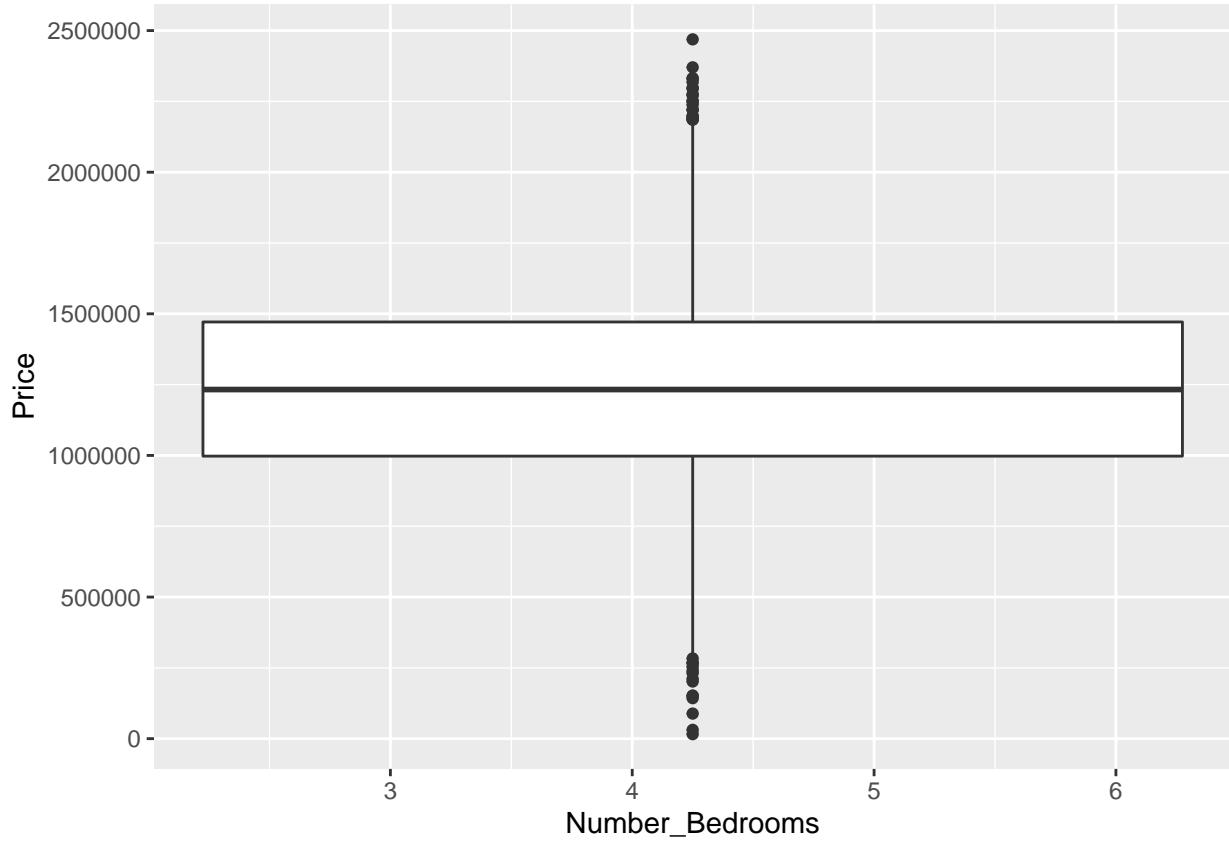
```
# Evaluate Correlations Amongst Variables  
ggpairs(train_set, columns = c(1:6))
```



```
## Correlation evaluation appears to support that Number_Bedrooms has a much lower correlation
## and statistical significance towards Price.
```

```
# Check for Outliers
```

```
ggplot(data = train_set) + geom_boxplot(aes(x = Number_Bedrooms, y = Price)) # Some outliers exist
```



```

train_set_without_outliers <- train_set[-which(train_set$Price %in%
                                             boxplot(train_set$Price, plot = FALSE)$out),]

## There were 34 outliers, uncertain if their removal will significantly impact overall prediction

# Visualize Outlier Differences

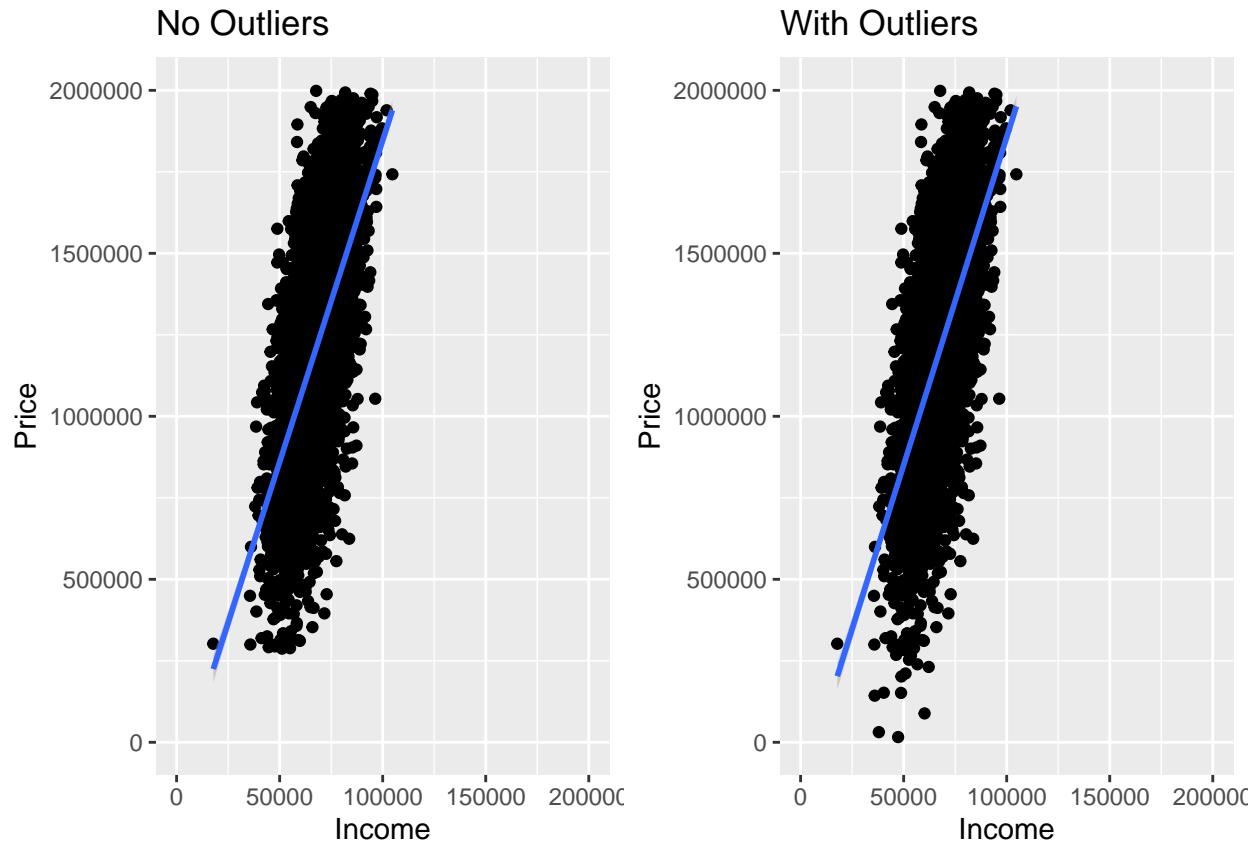
rounded_without_outliers <- round(train_set_without_outliers)
rounded_with_outliers <- round(train_set)

plot1 <- ggplot(data = rounded_without_outliers, aes(x = Income, y = Price)) +
  geom_point() + geom_smooth(method = lm) + xlim(0, 200000) + ylim(0, 2000000) +
  ggtitle("No Outliers")

plot2 <- ggplot(data = rounded_with_outliers, aes(x = Income, y = Price)) +
  geom_point() + geom_smooth(method = lm) + xlim(0, 200000) + ylim(0, 2000000) +
  ggtitle("With Outliers")

grid.arrange(plot1, plot2, ncol = 2)

```



## Second Model

Let's try our second model and see if it performs any better.

```
# Fitting Multiple Linear Regression to the Training set Without Number_Bedrooms

train_set_without_outliers <- train_set_without_outliers[,c(1:3,5:6)] # Remove Number_Bedrooms

model2 <- lm(formula = Price ~ ., data = train_set_without_outliers) # Price is the dependent
summary(model2) # variable as a function of all dependent variables via '.'

## Call:
## lm(formula = Price ~ ., data = train_set_without_outliers)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -325927  -70218     -77    68094  344474 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.608e+06  1.947e+04 -133.97  <2e-16 ***
## Income       2.143e+01  1.535e-01   139.60  <2e-16 ***
## House_Age    1.642e+05  1.612e+03   101.85  <2e-16 ***
```

```

## Number_Rooms      1.213e+05  1.581e+03   76.72   <2e-16 ***
## Area_Population  1.499e+01   1.617e-01   92.69   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 100400 on 3961 degrees of freedom
## Multiple R-squared:  0.9138, Adjusted R-squared:  0.9137
## F-statistic: 1.05e+04 on 4 and 3961 DF, p-value: < 2.2e-16

## All independent variables are now statistically dependent. R-squared and p-values remain
## statistically significant, however, they both actually decreased by approximately 0.005

# Using Fitted model2 for Predicting the Test Set Results

y_pred2 <- predict(model2, newdata = test_set)
visual_comparison2 <- data.frame(test_set$Price, y_pred2, residuals = test_set$Price - y_pred2)
visual_comparison2[1:10,]

##      test_set.Price    y_pred2    residuals
## 18      528485.2  492630.8   35854.40
## 29     1240763.8 1403825.0 -163061.19
## 34      534305.1  614597.5  -80292.33
## 38     1081150.1 1168880.7 -87730.62
## 45      1153871.5  942585.7  211285.78
## 58     1186688.5 1009277.6  177410.94
## 61     1111085.0  939507.8  171577.19
## 71     1146532.5 1054453.6  92078.83
## 75     1534479.9 1353139.0  181340.90
## 78     1204598.0 1349202.8 -144604.77

```

Predictions were slightly different but essentially returned no improvement.

## Influential Data Points

Since there was essentially no improvement, let's detect influential data points with Cook's Distance to see if this may improve our model.

```

# Let's Detect Influential Data Points with Cook's Distance
# These are more influential than simple outliers

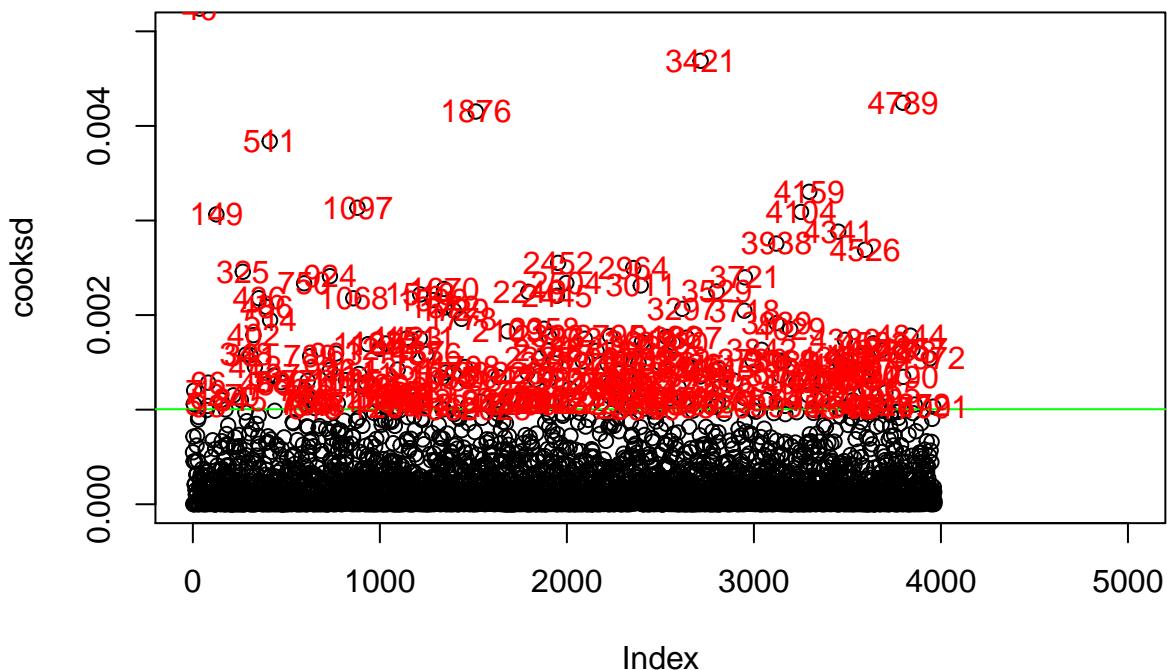
cooksd <- cooks.distance(model2)
mean(cooksd)

## [1] 0.0002511019

plot(cooksd, main = "Influential Observations by Cook's Distance",
      xlim = c(0,5000), ylim = c(0,0.005))
abline(h = 4 * mean(cooksd, na.rm=T), col = "green") # Creates cutoff to find influential points
text(x = 1:length(cooksd) + 1, y = cooksd,
      labels = ifelse(cooksd > 4 * mean(cooksd, na.rm = T), names(cooksd), ""), col = "red")

```

## Influential Observations by Cook's Distance



```

influential <- as.numeric(names(cooksd)[(cooksd > 4 * mean(cooksd, na.rm=T))])
# influential row numbers
head(train_set_without_outliers[influential, ]) # Take a peek at a few influntial observations

##      Income House_Age Number_Rooms Area_Population Price
## 5    59982.20   5.040555     7.839388    26354.11 630943.5
## 7    64698.46   6.025336     8.147760    60828.25 1502055.8
## 44   70421.48   6.907083     7.634319    43183.93 1744932.2
## 107  78633.97   5.465393     7.531143    43728.80 1555320.5
## 111  55472.65   4.822147     5.855972    15353.96 340605.2
## 181  61526.97   6.593963     9.180401    27307.95 1381430.6

train_set_without_outliers_influential <- train_set_without_outliers[-influential, ]
nrow(train_set_without_outliers_influential)

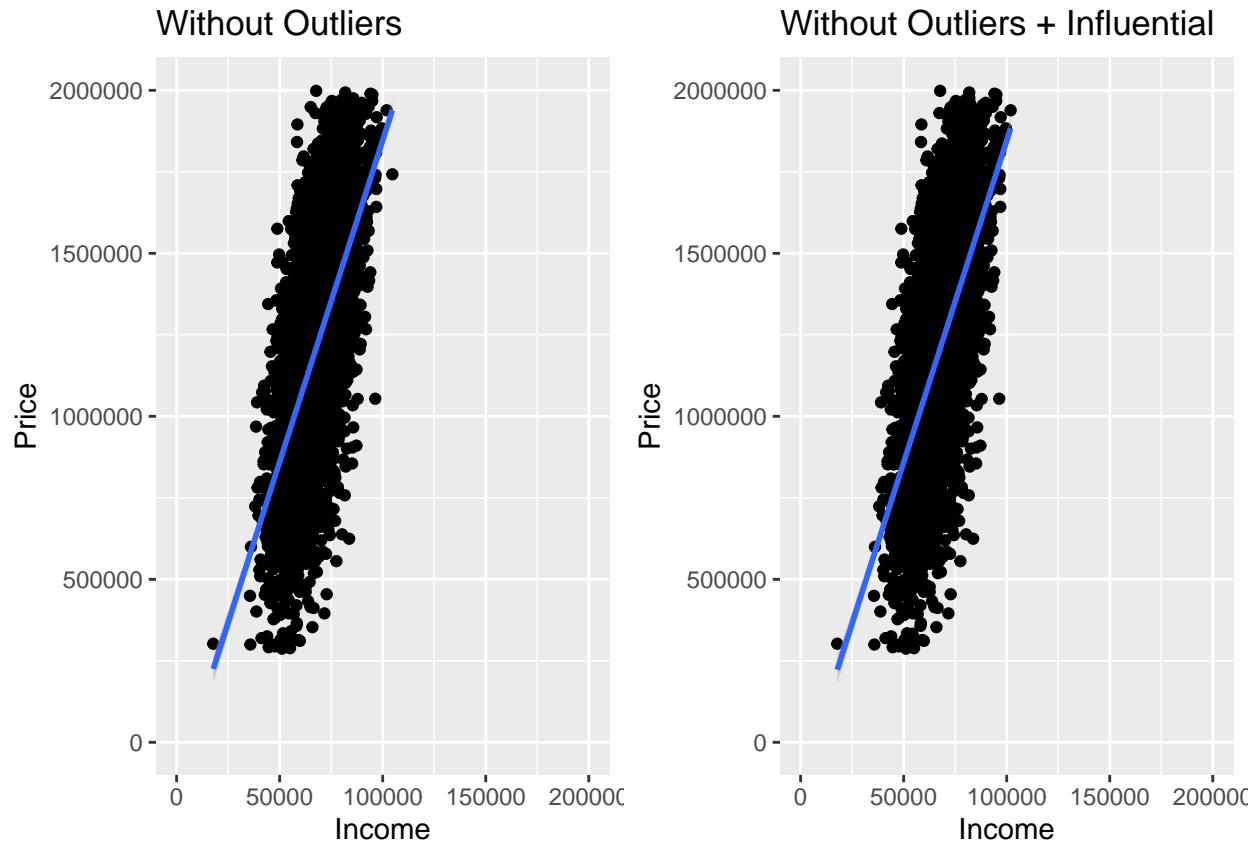
## [1] 3807

plot3 <- ggplot(data = rounded_without_outliers, aes(x = Income, y = Price)) + geom_point() +
  geom_smooth(method = lm) + xlim(0, 200000) + ylim(0, 2000000) + ggtitle("Without Outliers")

plot4 <- ggplot(data = train_set_without_outliers_influential, aes(x = Income, y = Price)) +
  geom_point() + geom_smooth(method = lm) + xlim(0, 200000) + ylim(0, 2000000) +
  ggtitle("Without Outliers + Influential")

grid.arrange(plot3, plot4, ncol = 2)

```



### Third Model

Let's try our third model and see if it performs any better.

```
# Fitting Multiple Linear Regression to the train_set3

model3 <- lm(formula = Price ~ ., data = train_set_without_outliers_influential) # Price is the
summary(model3) #dependent variable as a function of all dependent variables via '.'

## 
## Call:
## lm(formula = Price ~ ., data = train_set_without_outliers_influential)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -324600  -70410    331  68563  344430 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.602e+06  1.988e+04 -130.90   <2e-16 ***
## Income       2.136e+01  1.567e-01  136.33   <2e-16 ***
## House_Age    1.641e+05  1.649e+03   99.49   <2e-16 ***
## Number_Rooms 1.214e+05  1.621e+03   74.94   <2e-16 ***
## Area_Population 1.493e+01  1.649e-01   90.56   <2e-16 ***
```

```

## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 100500 on 3802 degrees of freedom
## Multiple R-squared:  0.9136, Adjusted R-squared:  0.9135
## F-statistic: 1.005e+04 on 4 and 3802 DF,  p-value: < 2.2e-16

## Initial review, model3 virtually fit the same as model2

# Predicting the Test Set Results on train_set3

y_pred3 <- predict(model3, newdata = test_set)
visual_comparison3 <- data.frame(test_set$Price, y_pred3, residuals = test_set$Price - y_pred3)
visual_comparison3[1:10,]

##      test_set.Price    y_pred3    residuals
## 18      528485.2  493336.9   35148.38
## 29     1240763.8 1401712.6 -160948.87
## 34      534305.1  616754.6  -82449.42
## 38     1081150.1 1168679.7  -87529.60
## 45     1153871.5  943443.6  210427.83
## 58     1186688.5 1010048.6  176639.87
## 61     1111085.0  939620.9  171464.13
## 71     1146532.5 1054445.7  92086.75
## 75     1534479.9 1352694.3  181785.65
## 78     1204598.0 1348511.3 -143913.31

```

Predictions were slightly different but essentially no improvement still. This is likely due to the fact that we had 4000 observations in the training set where outliers and influential only accounted for 193 total observations.

- (model1 = 4000; model2 = 3966; model3 = 3807)

Thus not enough change to influence the overall algorithm. Other models may be even more accurate if we attempted to remove other independent variables as well.

## Results

### Accuracy of the Models

Since removing variables can be intensive and time-consuming, let's evaluate our models to see how accurately they performed.

```

# Accuracy of the Models on the Different Training Sets

pred1 <- model1$fitted.values
pred2 <- model2$fitted.values
pred3 <- model3$fitted.values

tally_table <- data.frame(actual = train_set$Price, predicted = pred1)

```

```

mape <- mean(abs(tally_table$actual - tally_table$predicted) / tally_table$actual)
accuracy1 <- 1 - mape

tally_table <- data.frame(actual = train_set_without_outliers$Price, predicted = pred2)
mape <- mean(abs(tally_table$actual - tally_table$predicted) / tally_table$actual)
accuracy2 <- 1 - mape

tally_table <- data.frame(actual = train_set_without_outliers_influential$Price, predicted = pred3)
mape <- mean(abs(tally_table$actual - tally_table$predicted) / tally_table$actual)
accuracy3 <- 1 - mape

mape_preds <- data.frame(Method_on_Training_Set =
                           c("Full Training Set", "Without Outliers", "Without Outliers + Influential"),
                           MAPE = percent(c(accuracy1, accuracy2, accuracy3)))
kable(mape_preds, booktabs = T, col.names = c("Method", "MAPE")) %>%
  kable_styling(latex_options = "striped")

```

Method	MAPE
Full Training Set	92.346%
Without Outliers	92.644%
Without Outliers + Influential	92.611%

We see that model2 returned the greatest accuracy on the training set.

```

# Accuracy on test_set

tally_table_2 <- data.frame(actual = test_set$Price, predicted = y_pred1)
mape_test <- mean(abs(tally_table_2$actual - tally_table_2$predicted) / tally_table_2$actual)
accuracy_test1 <- 1 - mape_test

tally_table_2 <- data.frame(actual = test_set$Price, predicted = y_pred2)
mape_test <- mean(abs(tally_table_2$actual - tally_table_2$predicted) / tally_table_2$actual)
accuracy_test2 <- 1 - mape_test

tally_table_2 <- data.frame(actual = test_set$Price, predicted = y_pred3)
mape_test <- mean(abs(tally_table_2$actual - tally_table_2$predicted) / tally_table_2$actual)
accuracy_test3 <- 1 - mape_test

mape_preds_test <- data.frame(Method_on_Test_Set = c("Full Training Set", "Without Outliers",
                                                       "Without Outliers + Influential"),
                                 MAPE = percent(c(accuracy_test1, accuracy_test2, accuracy_test3)))
kable(mape_preds_test, booktabs = T, col.names = c("Method", "MAPE")) %>%
  kable_styling(latex_options = "striped")

```

Method	MAPE
Full Training Set	92.3903%
Without Outliers	92.3780%
Without Outliers + Influential	92.3711%

## Conclusion

We see that the accuracy values on the test set were virtually the same for each model. Since the accuracy was relatively high, it may not be worth the time spent on testing other models with different independent variable combinations.

```
sessionInfo()

## R version 3.6.3 (2020-02-29)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] parallel stats      graphics grDevices utils      datasets methods
## [8] base
##
## other attached packages:
##  [1] zoo_1.8-8           xlsx_0.6.4.2        wordcloud_2.6
##  [4] viridisLite_0.3.0   topicmodels_0.2-11   tm_0.7-7
##  [7] NLP_0.2-0          forcats_0.5.0       purrr_0.3.4
## [10] readr_1.3.1         tibble_3.0.1        tidyverse_1.3.0
## [13] tidytext_0.2.6       tidyverse_1.3.0     tensorflow_2.2.0
## [16] syuzhet_1.0.4       stringr_1.4.0       stm_1.3.5
## [19] SnowballC_0.7.0     sentimentr_2.7.1    SentimentAnalysis_1.3-3
## [22] scales_1.1.1        RWeka_0.4-43       rlist_0.4.6.1
## [25] readxl_1.3.1        randomForest_4.6-14  quantada_2.1.1
## [28] qdap_2.4.3          RColorBrewer_1.1-2   qdapTools_1.3.5
## [31] qdapRegex_0.7.2     qdapDictionaries_1.0.7 plotrix_3.7-8
## [34] plotly_4.9.2.1      petro.One_0.2.3     NbClust_3.0
## [37] mclust_5.4.6        magrittr_1.5         ldatuning_1.0.2
## [40] knitr_1.30          kableExtra_1.2.1    gridExtra_2.3
## [43] ggthemes_4.2.0       GGally_2.0.0        fpc_2.2-8
## [46] FactoMineR_2.3      factoextra_1.0.7   e1071_1.7-3
## [49] dplyr_1.0.0          doParallel_1.0.15   iterators_1.0.12
## [52] foreach_1.5.0        dendextend_1.14.0   data.table_1.12.8
## [55] corrplot_0.84        corpus_0.10.1      Clustering_1.6
## [58] cluster_2.1.0        caret_6.0-86       ggplot2_3.3.2
## [61] lattice_0.20-41     car_3.0-10        carData_3.0-4
## [64] caTools_1.18.0

##
## loaded via a namespace (and not attached):
##  [1] ClusterR_1.2.2      prabclus_2.3-2       ModelMetrics_1.2.2.2
##  [4] stopwords_2.0         bit64_4.0.5         rpart_4.1-15
```

```

## [ 7] RCurl_1.98-1.2      generics_0.0.2      openNLP_0.2-7
## [10] usethis_1.6.3        gama_1.0.3       RSQLite_2.2.0
## [13] openNLPdata_1.5.3-4   chron_2.3-56     bit_4.0.4
## [16] tokenizers_0.2.1     webshot_0.5.2     xml2_1.3.2
## [19] lubridate_1.7.9      assertthat_0.2.1 viridis_0.5.1
## [22] gower_0.2.1         amap_0.8-18      xfun_0.17
## [25] hms_0.5.3           rJava_0.9-12     evaluate_0.14
## [28] DEoptimR_1.0-8       fansi_0.4.1      dbplyr_1.4.4
## [31] igraph_1.2.5         DBI_1.1.0       htmlwidgets_1.5.1
## [34] reshape_0.8.8         apcluster_1.4.8 stats4_3.6.3
## [37] ellipsis_0.3.1      backports_1.1.10 RcppParallel_5.0.2
## [40] vctrs_0.3.2          abind_1.4-5      withr_2.2.0
## [43] robustbase_0.93-6    lazyeval_0.2.2    crayon_1.3.4
## [46] labeling_0.3          recipes_0.1.12   pkgconfig_2.0.3
## [49] slam_0.1-47          nlme_3.1-148    nnet_7.3-14
## [52] rlang_0.4.7           diptest_0.75-7   lifecycle_0.2.0
## [55] lexicon_1.2.1        modelr_0.1.8     cellranger_1.1.0
## [58] Matrix_1.2-18         reprex_0.3.0     base64enc_0.1-3
## [61] whisker_0.4           bitops_1.0-6     advclust_0.4
## [64] pROC_1.16.2          blob_1.2.1      venneuler_1.1-0
## [67] leaps_3.1             memoise_1.1.0    plyr_1.8.6
## [70] compiler_3.6.3        pvclust_2.2-0    clue_0.3-57
## [73] cli_2.0.2            janeaustenr_0.1.5 mgcv_1.8-31
## [76] MASS_7.3-51.6         tidyselect_1.1.0 stringi_1.4.6
## [79] yaml_2.2.1            ggrepel_0.8.2    grid_3.6.3
## [82] fastmatch_1.1-0      tools_3.6.3     rio_0.5.16
## [85] rstudioapi_0.11       foreign_0.8-75   prodlim_2019.11.13
## [88] farver_2.0.3          scatterplot3d_0.3-41 digest_0.6.25
## [91] pracma_2.2.9          lava_1.6.7      proto_1.0.0
## [94] Rcpp_1.0.5             broom_0.7.0     gender_0.5.4
## [97] httr_1.4.2            kernlab_0.9-29  colorspace_1.4-1
## [100] rvest_0.3.6          XML_3.99-0.3    fs_1.5.0
## [103] reticulate_1.16       splines_3.6.3    RWekajars_3.9.3-2
## [106] xlsxjars_0.6.1        flexmix_2.3-17   xtable_1.8-4
## [109] gmp_0.6-0              jsonlite_1.7.1   timeDate_3043.102
## [112] flashClust_1.01-2    modeltools_0.2-23 ipred_0.9-9
## [115] R6_2.4.1              gsubfn_0.7      pillar_1.4.4
## [118] htmltools_0.5.0        glue_1.4.1      class_7.3-17
## [121] codetools_0.2-16      utf8_1.1.4     sqldf_0.4-11
## [124] curl_4.3               tfruns_1.4      gtools_3.8.2
## [127] zip_2.1.1              openxlsx_4.1.5  survival_3.2-3
## [130] textclean_0.9.3        rmarkdown_2.3    munsell_0.5.0
## [133] haven_2.3.1            reshape2_1.4.4  gtable_0.3.0

## Time difference of 18.89066 secs

```