Advaced R - Individual Assignment

Tomas Tello

22 de mayo de 2019

House Prices Prediction Problem

This report shows the methodology followed to analyze a House Prices dataset to build a model that predicts the prices on a test set according to explanatory variables.

Data Loading

First step: Load the dataset from local storage into data table objects

```
raw_data_train <-fread('data/house_price_train.csv', stringsAsFactors = F)
raw_data_test <-fread('data/house_price_test.csv', stringsAsFactors = F)

str(raw_data_train)
```

```
## Classes 'data.table' and 'data.frame':   17277 obs. of  21 variables:
##  $ id           :integer64 9183703376 464000600 2224079050 6163901283 6392003810 7974200948 2
426059124 2115510300 ...
##  $ date         : chr  "5/13/2014" "8/27/2014" "7/18/2014" "1/30/2015" ...
##  $ price        : num  225000 641250 810000 330000 530000 ...
##  $ bedrooms     : int  3 3 4 4 4 4 4 3 4 3 ...
##  $ bathrooms    : num  1.5 2.5 3.5 1.5 1.75 3.5 3.25 2.25 2.5 1.5 ...
##  $ sqft_living  : int  1250 2220 3980 1890 1814 3120 4160 1440 2250 2540 ...
##  $ sqft_lot     : int  7500 2550 209523 7540 5000 5086 47480 10500 6840 9520 ...
##  $ floors       : num  1 3 2 1 1 2 2 1 2 1 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 2 2 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 4 4 3 3 3 3 3 ...
##  $ grade        : int  7 10 9 7 7 9 10 8 9 8 ...
##  $ sqft_above   : int  1250 2220 3980 1890 944 2480 4160 1130 2250 1500 ...
##  $ sqft_basement: int  0 0 0 0 870 640 0 310 0 1040 ...
##  $ yr_built     : int  1967 1990 2006 1967 1951 2008 1995 1983 1987 1959 ...
##  $ yr_renovated : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98030 98117 98024 98155 98115 98115 98072 98023 98058 98115 ...
##  $ lat          : num  47.4 47.7 47.6 47.8 47.7 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1260 2200 2220 1890 1290 1880 3400 1510 2480 1870 ...
##  $ sqft_lot15   : int  7563 5610 65775 8515 5000 5092 40428 8125 7386 6800 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Data Cleaning and Preparation

Every transformation is declared within a function, so it can be replicated in the train and test sets.

```r
transformations <- function(df){
  #convert to date format
  df$date <- as.Date(df$date, "%m/%d/%Y")
  #convert discrete variables to factors
  df$zipcode <- as.factor(df$zipcode)
  #convert all integers to numeric
  df[ , names(df)[sapply(df, is.integer)]:=
             lapply(.SD,as.numeric),.SDcols =
             names(df)[sapply(df, is.integer)]]

  return(df)
}

train_data <- transformations(raw_data_train)
test_data <- transformations(raw_data_test)

str(train_data)
```

```
## Classes 'data.table' and 'data.frame':   17277 obs. of  21 variables:
##  $ id           :integer64 9183703376 464000600 2224079050 6163901283 6392003810 7974200948 2
426059124 2115510300 ...
##  $ date         : Date, format: "2014-05-13" "2014-08-27" ...
##  $ price        : num  225000 641250 810000 330000 530000 ...
##  $ bedrooms     : num  3 3 4 4 4 4 4 3 4 3 ...
##  $ bathrooms    : num  1.5 2.5 3.5 1.5 1.75 3.5 3.25 2.25 2.5 1.5 ...
##  $ sqft_living  : num  1250 2220 3980 1890 1814 ...
##  $ sqft_lot     : num  7500 2550 209523 7540 5000 ...
##  $ floors       : num  1 3 2 1 1 2 2 1 2 1 ...
##  $ waterfront   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : num  0 2 2 0 0 0 0 0 0 0 ...
##  $ condition    : num  3 3 3 4 4 3 3 3 3 3 ...
##  $ grade        : num  7 10 9 7 7 9 10 8 9 8 ...
##  $ sqft_above   : num  1250 2220 3980 1890 944 2480 4160 1130 2250 1500 ...
##  $ sqft_basement: num  0 0 0 0 870 640 0 310 0 1040 ...
##  $ yr_built     : num  1967 1990 2006 1967 1951 ...
##  $ yr_renovated : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : Factor w/ 70 levels "98001","98002",..: 19 52 15 63 50 50 37 14 33 50 ...
##  $ lat          : num  47.4 47.7 47.6 47.8 47.7 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: num  1260 2200 2220 1890 1290 1880 3400 1510 2480 1870 ...
##  $ sqft_lot15   : num  7563 5610 65775 8515 5000 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Data Exploration

Using the DataExplorer library, we can can do a quick analysis of the variables:

1. Basic metrics (including NA detection)

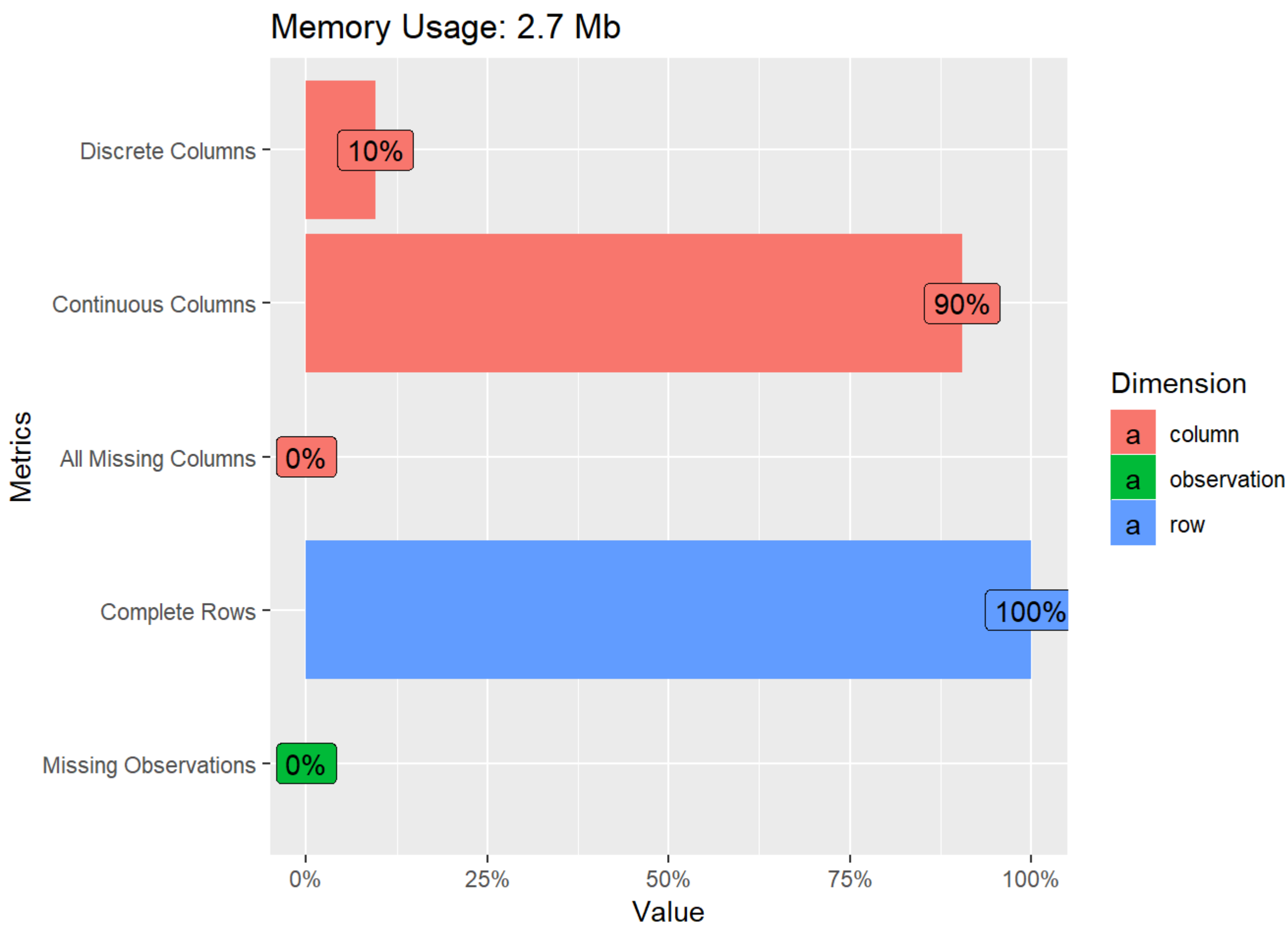
```r
summary(train_data)
```

```
##        id                  date                 price        
##  Min.   :   1000102   Min.   :2014-05-02   Min.   :  78000  
##  1st Qu.:2113701080   1st Qu.:2014-07-21   1st Qu.: 320000  
##  Median :3902100205   Median :2014-10-16   Median : 450000  
##  Mean   :4566440237   Mean   :2014-10-28   Mean   : 539865  
##  3rd Qu.:7302900090   3rd Qu.:2015-02-17   3rd Qu.: 645500  
##  Max.   :9900000190   Max.   :2015-05-27   Max.   :7700000  
##                                                             
##     bedrooms        bathrooms      sqft_living       sqft_lot      
##  Min.   : 1.000   Min.   :0.500   Min.   :  370   Min.   :    520  
##  1st Qu.: 3.000   1st Qu.:1.750   1st Qu.: 1430   1st Qu.:   5050  
##  Median : 3.000   Median :2.250   Median : 1910   Median :   7620  
##  Mean   : 3.369   Mean   :2.114   Mean   : 2080   Mean   :  15186  
##  3rd Qu.: 4.000   3rd Qu.:2.500   3rd Qu.: 2550   3rd Qu.:  10695  
##  Max.   :33.000   Max.   :8.000   Max.   :13540   Max.   :1164794  
##                                                                    
##      floors        waterfront            view          condition    
##  Min.   :1.000   Min.   :0.000000   Min.   :0.0000   Min.   :1.000  
##  1st Qu.:1.000   1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:3.000  
##  Median :1.500   Median :0.000000   Median :0.0000   Median :3.000  
##  Mean   :1.493   Mean   :0.007467   Mean   :0.2335   Mean   :3.413  
##  3rd Qu.:2.000   3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:4.000  
##  Max.   :3.500   Max.   :1.000000   Max.   :4.0000   Max.   :5.000  
##                                                                     
##      grade          sqft_above   sqft_basement       yr_built   
##  Min.   : 3.00   Min.   : 370   Min.   :   0.0   Min.   :1900  
##  1st Qu.: 7.00   1st Qu.:1190   1st Qu.:   0.0   1st Qu.:1951  
##  Median : 7.00   Median :1564   Median :   0.0   Median :1975  
##  Mean   : 7.66   Mean   :1791   Mean   : 289.4   Mean   :1971  
##  3rd Qu.: 8.00   3rd Qu.:2210   3rd Qu.: 556.0   3rd Qu.:1997  
##  Max.   :13.00   Max.   :9410   Max.   :4820.0   Max.   :2015  
##                                                                
##   yr_renovated        zipcode           lat             long       
##  Min.   :   0.00   98103  :  498   Min.   :47.16   Min.   :-122.5  
##  1st Qu.:   0.00   98038  :  482   1st Qu.:47.47   1st Qu.:-122.3  
##  Median :   0.00   98052  :  465   Median :47.57   Median :-122.2  
##  Mean   :  85.35   98115  :  460   Mean   :47.56   Mean   :-122.2  
##  3rd Qu.:   0.00   98117  :  440   3rd Qu.:47.68   3rd Qu.:-122.1  
##  Max.   :2015.00   98042  :  437   Max.   :47.78   Max.   :-121.3  
##                    (Other):14495                                   
##  sqft_living15    sqft_lot15    
##  Min.   : 460   Min.   :   659  
##  1st Qu.:1490   1st Qu.:  5100  
##  Median :1840   Median :  7639  
##  Mean   :1986   Mean   : 12826  
##  3rd Qu.:2360   3rd Qu.: 10080  
##  Max.   :6210   Max.   :871200  
## 
```

```
introduce(train_data)
```

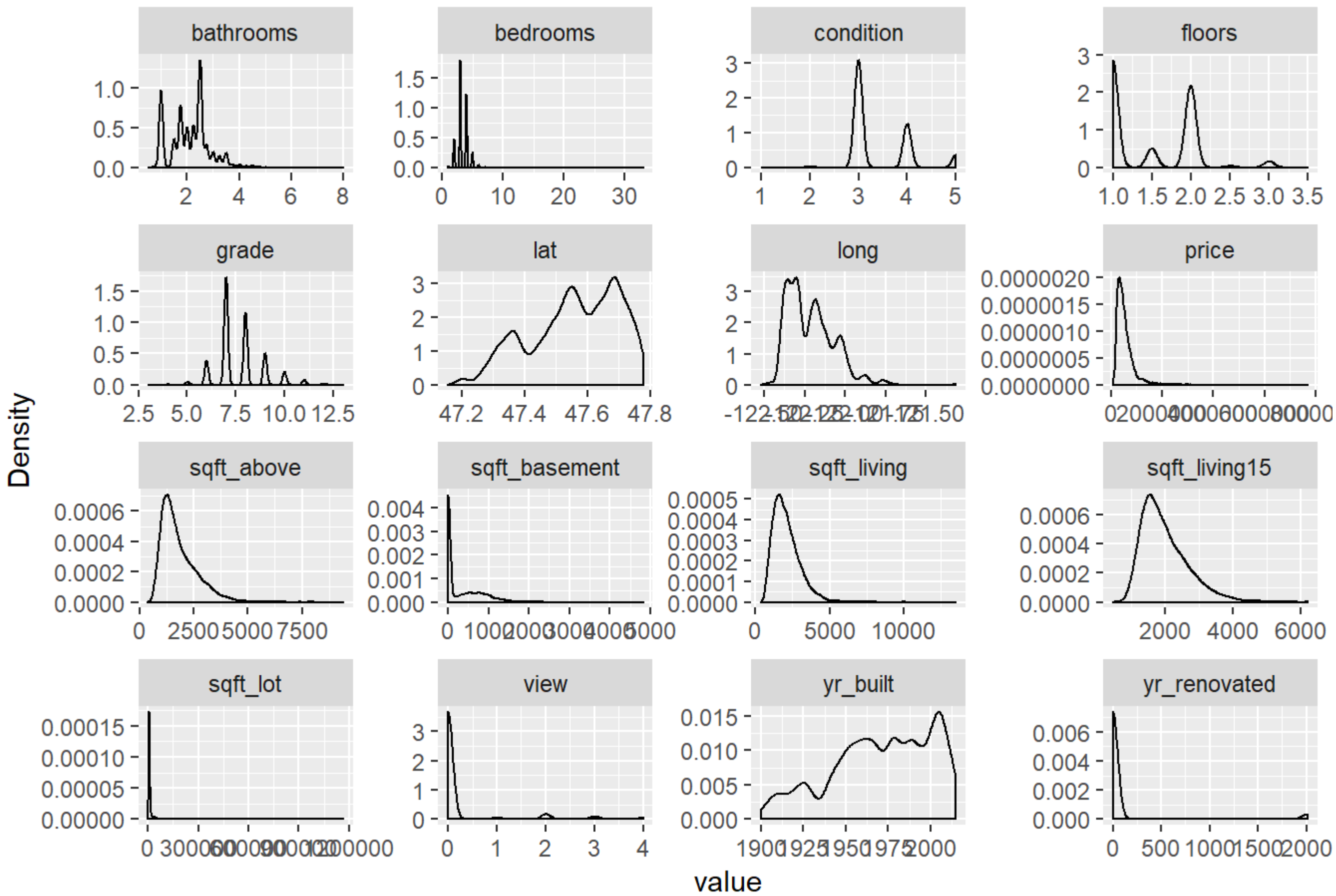
```
##     rows columns discrete_columns continuous_columns all_missing_columns
## 1: 17277      21                2                 19                   0
##    total_missing_values complete_rows total_observations memory_usage
## 1:                    0         17277             362817      2843776
```

```
plot_intro(train_data)
```

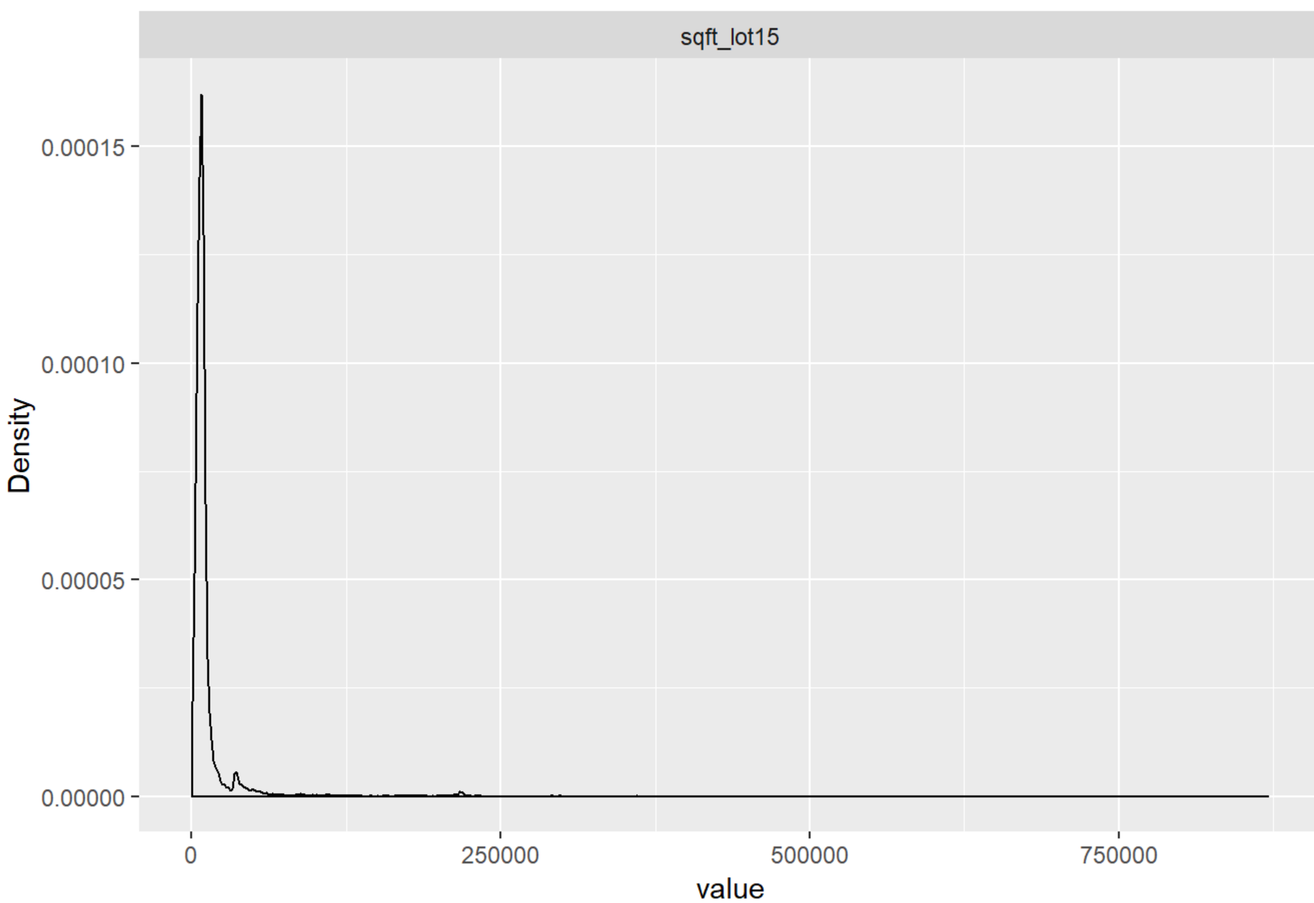


2. Continuous variables analysis and how they relate to the target variable

```
plot_density(train_data[,-c('id')])
```

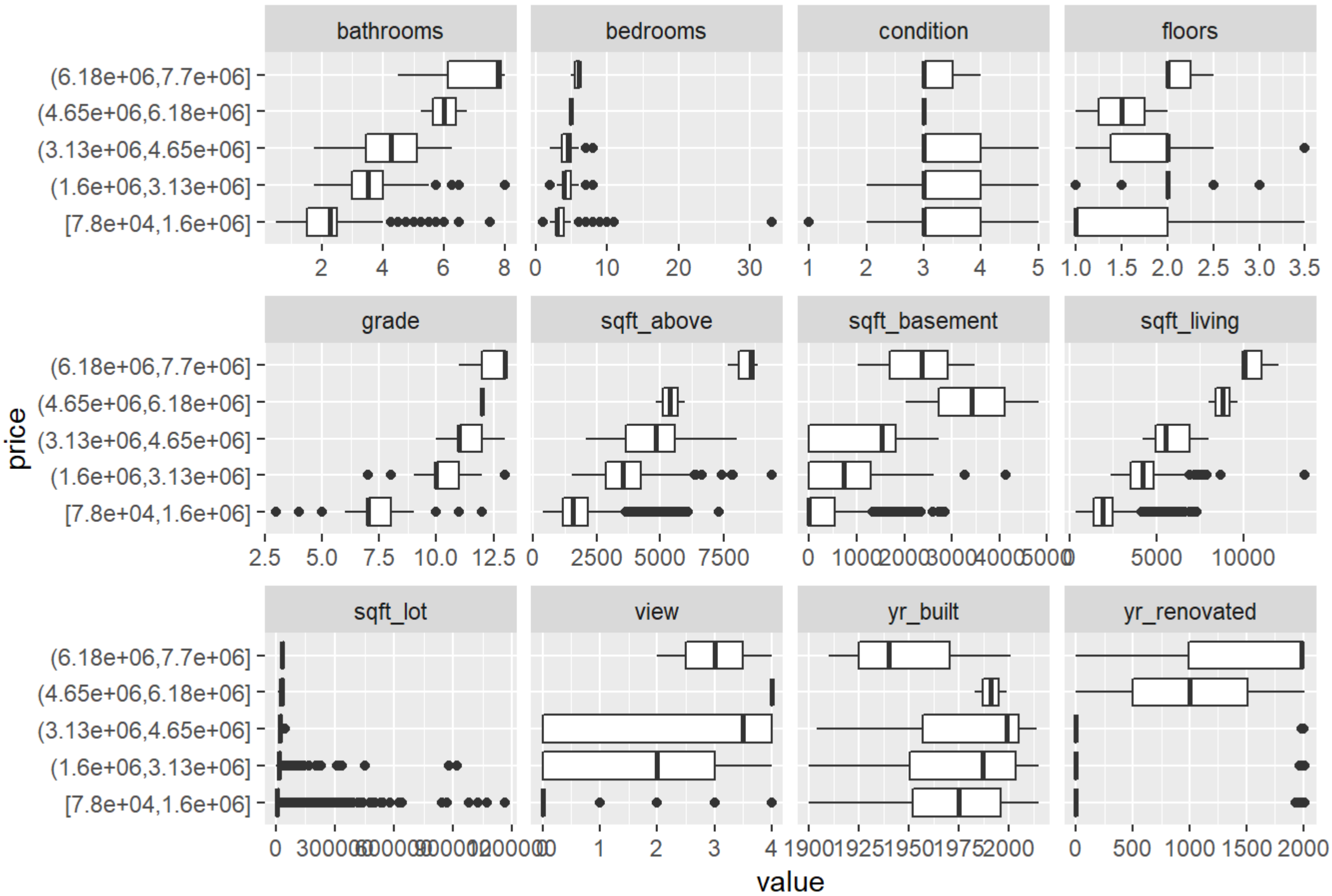

bathrooms
bedrooms
condition
floors
grade
lat
long
price
sqft_above
sqft_basement
sqft_living
sqft_living15
sqft_lot
view
yr_built
yr_renovated
Density
value

Page 1

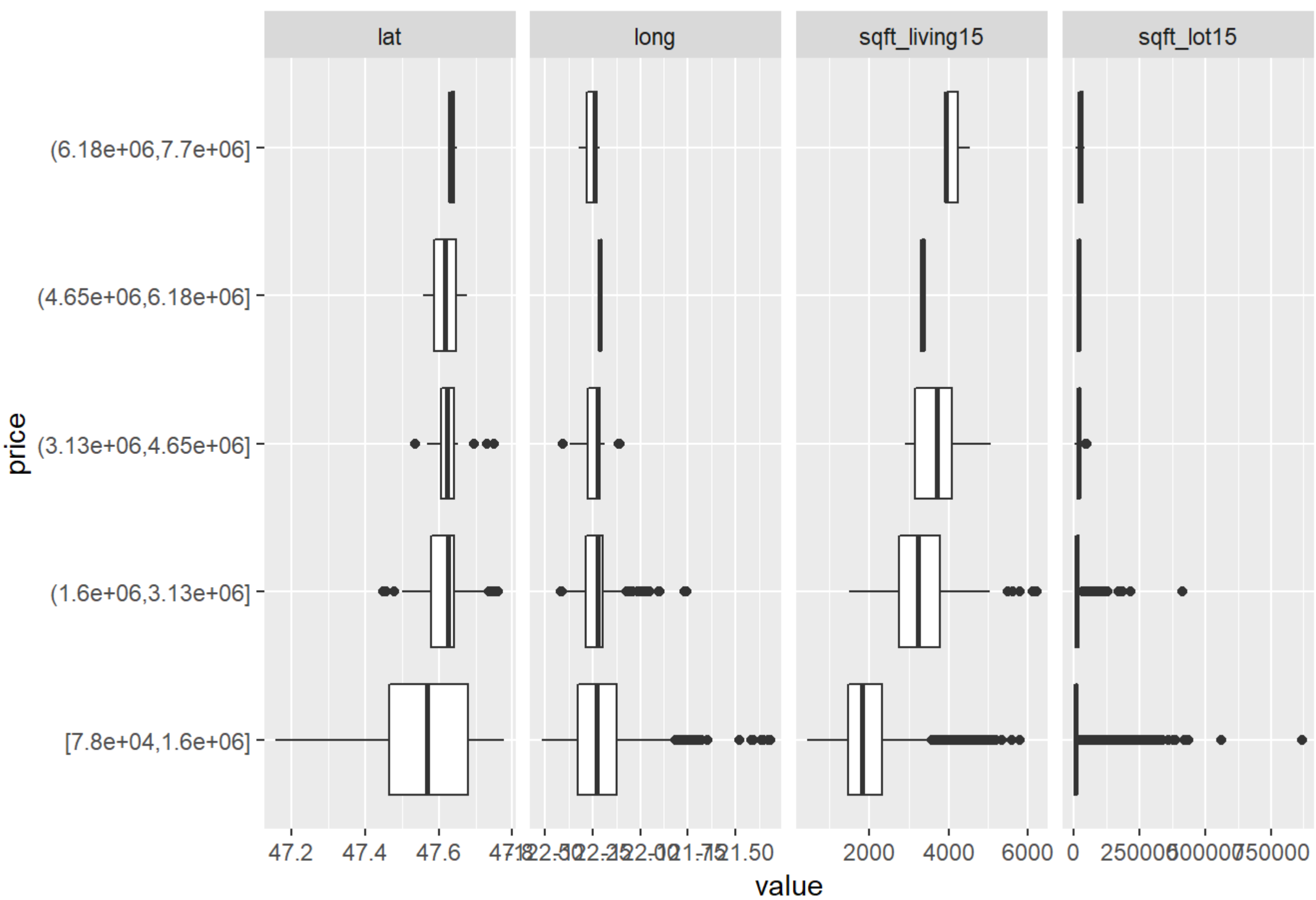

sqft_lot15
0.00015
0.00010
0.00005
0.00000
Density
0
250000
500000
750000
value

Page 2

```
plot_boxplot(train_data[,-c('id')], by = "price")
```


bathrooms
bedrooms
condition
floors
grade
sqft_above
sqft_basement
sqft_living
sqft_lot
view
yr_built
yr_renovated
price
(6.18e+06,7.7e+06]
(4.65e+06,6.18e+06]
(3.13e+06,4.65e+06]
(1.6e+06,3.13e+06]
[7.8e+04,1.6e+06]
value

Page 1


lat
long
sqft_living15
sqft_lot15
price
(6.18e+06,7.7e+06]
(4.65e+06,6.18e+06]
(3.13e+06,4.65e+06]
(1.6e+06,3.13e+06]
[7.8e+04,1.6e+06]
value

Page 2

Looking at the density plots and box plots, we can conclude that the variables “view”, “condition” and “floors” can be considered categorical, as they are not continuous and they show no ordinality in relation to the target variables

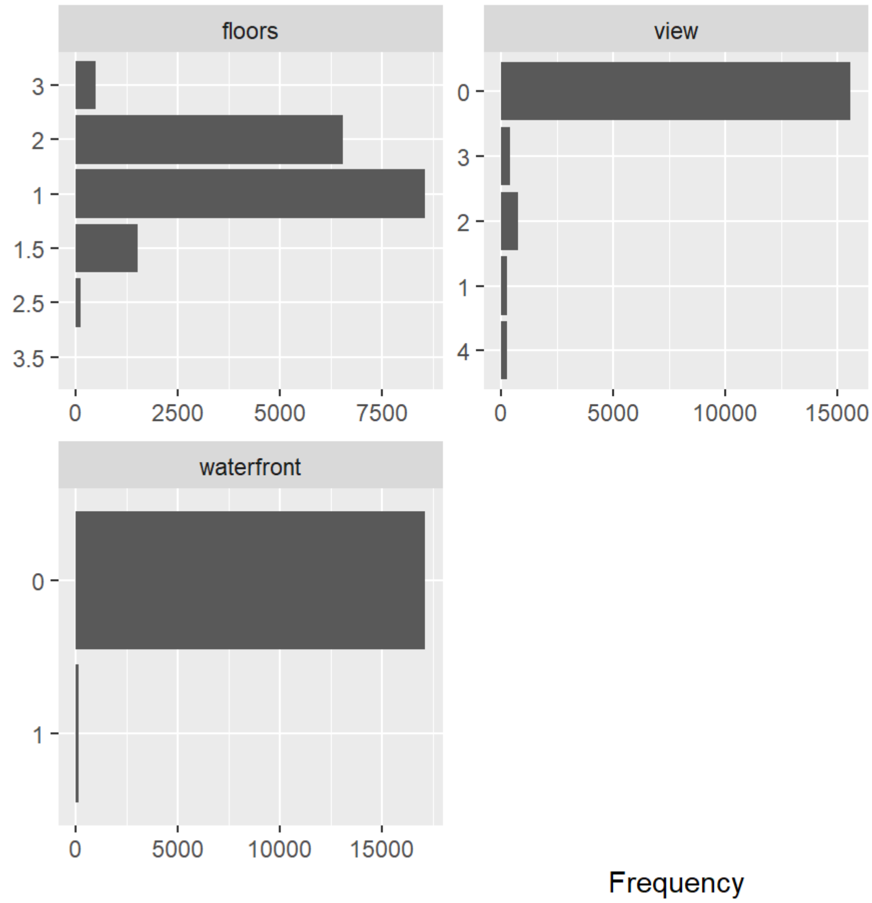
We then update the transformations function and apply it to the dataset:

```
transformations <- function(df){
  df1 <- data.table(df)

  #convert to date format
  df1$date <- as.Date(df1$date, "%m/%d/%Y")
  #convert discrete variables to factors
  df1$zipcode <- as.factor(df1$zipcode)
  df1$condition <- as.factor(df1$condition)
  df1$view <- as.factor(df1$view)
  df1$floors <- as.factor(df1$floors)
  #convert all integers to numeric
  df1[ , names(df1)[sapply(df1, is.integer)]:=
             lapply(.SD,as.numeric),.SDcols =
             names(df1)[sapply(df1, is.integer)]]

  return(df1)
}

scale_df <- function(df){
  df1 <- data.table(df)

  numeric_vars <- names(df1)[sapply(df1, is.numeric)]
  numeric_vars <- numeric_vars[!numeric_vars %in% c('price')]
  ## Scale
  df1[, (numeric_vars) := lapply(.SD, scale), .SDcols=numeric_vars]

  return(df1)

}

train_data <- transformations(raw_data_train)
test_data <- transformations(raw_data_test)

str(train_data)
```

```
## Classes 'data.table' and 'data.frame':   17277 obs. of  21 variables:
##  $ id           :integer64 9183703376 464000600 2224079050 6163901283 6392003810 7974200948 2
426059124 2115510300 ...
##  $ date         : Date, format: "2014-05-13" "2014-08-27" ...
##  $ price        : num  225000 641250 810000 330000 530000 ...
##  $ bedrooms     : num  3 3 4 4 4 4 4 3 4 3 ...
##  $ bathrooms    : num  1.5 2.5 3.5 1.5 1.75 3.5 3.25 2.25 2.5 1.5 ...
##  $ sqft_living  : num  1250 2220 3980 1890 1814 ...
##  $ sqft_lot     : num  7500 2550 209523 7540 5000 ...
##  $ floors       : Factor w/ 6 levels "1","1.5","2",..: 1 5 3 1 1 3 3 1 3 1 ...
##  $ waterfront   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : Factor w/ 5 levels "0","1","2","3",..: 1 3 3 1 1 1 1 1 1 1 ...
##  $ condition    : Factor w/ 5 levels "1","2","3","4",..: 3 3 3 4 4 3 3 3 3 3 ...
##  $ grade        : num  7 10 9 7 7 9 10 8 9 8 ...
##  $ sqft_above   : num  1250 2220 3980 1890 944 2480 4160 1130 2250 1500 ...
##  $ sqft_basement: num  0 0 0 0 870 640 0 310 0 1040 ...
##  $ yr_built     : num  1967 1990 2006 1967 1951 ...
##  $ yr_renovated : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : Factor w/ 70 levels "98001","98002",..: 19 52 15 63 50 50 37 14 33 50 ...
##  $ lat          : num  47.4 47.7 47.6 47.8 47.7 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: num  1260 2200 2220 1890 1290 1880 3400 1510 2480 1870 ...
##  $ sqft_lot15   : num  7563 5610 65775 8515 5000 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

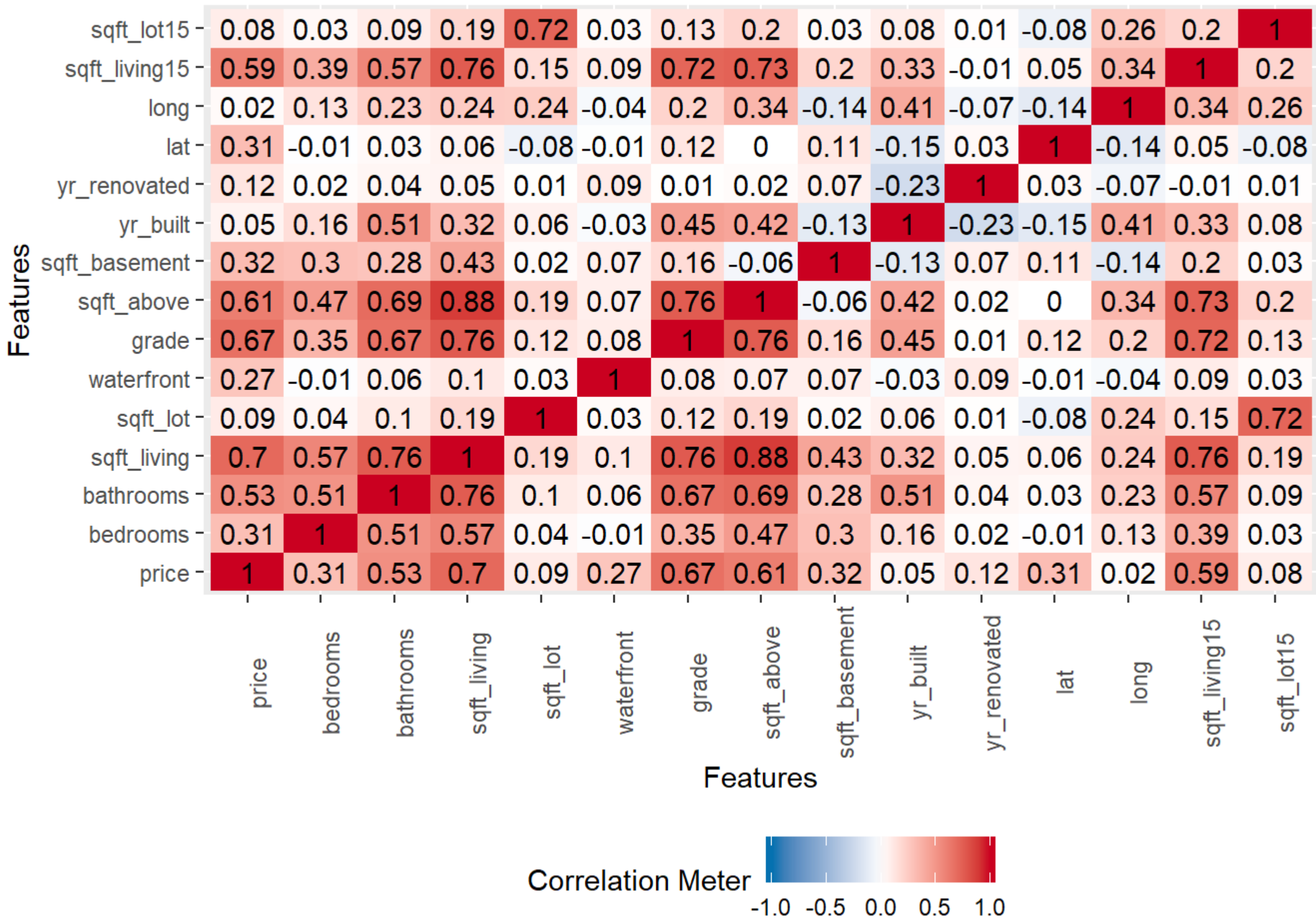
2. Categorical variables analysis

```
plot_bar(train_data)
```



3. Correlation analysis for continuous variables

```
plot_correlation(train_data[,-c('id')], type = "c")
```



Using the correlation matrix, we can start to do some feature selection, removing highly correlated variables. In this case: sqft_above has a 0.88 correlation with sqft_living, so we can remove this variable to avoid redundancy.

Also sqft_living15 and sqft_lot15 are highly correlated to their original counterparts, so we will remove these variables for our analysis.

```
train_data_sub <- train_data[,-c('id', 'sqft_living15', 'sqft_lot15', 'sqft_above', 'date')]
test_data_sub <- test_data[,-c('id', 'sqft_living15', 'sqft_lot15', 'sqft_above', 'date')]

str(train_data_sub)
```

```
## Classes 'data.table' and 'data.frame':   17277 obs. of  16 variables:
##  $ price        : num  225000 641250 810000 330000 530000 ...
##  $ bedrooms     : num  3 3 4 4 4 4 4 3 4 3 ...
##  $ bathrooms    : num  1.5 2.5 3.5 1.5 1.75 3.5 3.25 2.25 2.5 1.5 ...
##  $ sqft_living  : num  1250 2220 3980 1890 1814 ...
##  $ sqft_lot     : num  7500 2550 209523 7540 5000 ...
##  $ floors       : Factor w/ 6 levels "1","1.5","2",..: 1 5 3 1 1 3 3 1 3 1 ...
##  $ waterfront   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : Factor w/ 5 levels "0","1","2","3",..: 1 3 3 1 1 1 1 1 1 1 ...
##  $ condition    : Factor w/ 5 levels "1","2","3","4",..: 3 3 3 4 4 3 3 3 3 3 ...
##  $ grade        : num  7 10 9 7 7 9 10 8 9 8 ...
##  $ sqft_basement: num  0 0 0 0 870 640 0 310 0 1040 ...
##  $ yr_built     : num  1967 1990 2006 1967 1951 ...
##  $ yr_renovated : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : Factor w/ 70 levels "98001","98002",..: 19 52 15 63 50 50 37 14 33 50 ...
##  $ lat          : num  47.4 47.7 47.6 47.8 47.7 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Baseline Model

To run a first model (linear regression), we will first one hot encode all categorical variables. To do so, we need to stack both train and test datasets to make sure both datasets follow the same encoding.

```
train_data_sub$train <- 1
test_data_sub$train <- 0

stacked <- rbind(train_data_sub, test_data_sub, fill = TRUE)

#Dummify the stacked data table
stacked_dum <- dummify(stacked, maxcat = 70)

#Split again based on 'train' flag
train_data_sub <- stacked_dum[stacked_dum$train == 1, -'train']
test_data_sub <- stacked_dum[stacked_dum$train == 0, -'train']
```

To evaluate our baseline model and the feature engineering process, we will take a holdout (validation data) from the train dataset using the f_partition script:

```
train_val <- f_partition(train_data_sub, seed = 1414)
```

Now we train our baseline model. The metric for our predictions will be the Mean Absolute Percent Error (MAPE). After obtaining our baseline score, we will perform some feature engineering and evaluate if these new features help reducing the MAPE.

```
baseline <- lm(price ~ ., data=scale_df(train_val$train))

test_lm<-predict(baseline, newdata = scale_df(train_val$test))
```

```
## Warning in predict.lm(baseline, newdata = scale_df(train_val$test)):
## prediction from a rank-deficient fit may be misleading
```

```
mape_lm<-mape(real=train_val$test$price, predicted = test_lm)
mape_lm
```

```
## [1] 0.2023602
```

Feature Engineering

Data preparation pipeline

Before the feature engineering process, we will define every step prev

```
feat_select <- function(df){
    df_sub <- df[, -c('id', 'sqft_living15', 'sqft_lot15', 'sqft_above', 'date')]

    return(df_sub)
}

encode <- function(df_train, df_test){

  df_train$train <- 1
  df_test$train <- 0

  stacked <- rbind(df_train, df_test, fill = TRUE)

  #Dummify the stacked data table
  stacked_dum <- dummify(stacked, maxcat = 100)

  #Split again based on 'train' flag
  df_train <- stacked_dum[stacked_dum$train == 1, -'train']
  df_test <- stacked_dum[stacked_dum$train == 0, -'train']

  return(list("train" = df_train, "test" = df_test))
}
```

House age

The first feature to be created is the house age which will be obtained from the difference between the date of sale and the date of construction.

```
train_data <- transformations(raw_data_train)
test_data <- transformations(raw_data_test)

train_data$house_age <- year(train_data$date) - train_data$yr_built
test_data$house_age <- year(test_data$date) - test_data$yr_built


train_data_sub <- feat_select(train_data)
test_data_sub <- feat_select(test_data)

#train_data_enc <- encode(train_data_sub, test_data_sub)

train_val <- f_partition(train_data_sub, seed = 1414)

baseline <- lm(price ~ ., data= scale_df(train_val$train))

test_lm<-predict(baseline, newdata = scale_df(train_val$test))
mape_lm<-mape(real=train_val$test$price, predicted = test_lm)
mape_lm
```

```
## [1] 0.1986157
```

Calculating the house age already improved the MAPE by a little, so we are keeping this feature

House age after renovation

Similar to the last feature, now we will calculate the age of the house after renovation. If it has not been renewed, we will keep this value as 0.

```r
renovation_age <- function(df){
  df1 <- data.table(df)

  df1$renov_age <- 0

  for (i in 1:nrow(df)){
    if(df1$yr_renovated[i]>0){
      df1$renov_age[i] <- year(df1$date[i]) - df1$yr_renovated[i]
    }
  }

  return(df1)
}

train_data <- renovation_age(train_data)
test_data <- renovation_age(test_data)

feat_select <- function(df){
    df_sub <- df[, -c('id', 'sqft_living15', 'sqft_lot15', 'sqft_above', 'date')]

    return(df_sub)
}

train_data_sub <- feat_select(train_data)
test_data_sub <- feat_select(test_data)

#train_data_enc <- encode(train_data_sub, test_data_sub)

train_val <- f_partition(train_data_sub, seed = 1414)

baseline <- lm(price ~ ., data= scale_df(train_val$train))

test_lm<-predict(baseline, newdata = scale_df(train_val$test))
mape_lm<-mape(real=train_val$test$price, predicted = test_lm)
mape_lm
```

```
## [1] 0.1982341
```

Although there is no significant improvement in the score, we will keep this feature too.

Property size features

The dataset contains several features related to the property size. We will try to generate new features trying linear combinations of these size features so explore which ones improve our score:

```r
size_features <- function(df){
  df1 <- data.table(df)

  df1$living_lot_ratio <- df1$sqft_living/df1$sqft_lot
  df1$above_lot_ratio <- df1$sqft_above/df1$sqft_lot
  df1$basement_lot_ratio <- df1$sqft_basement/df1$sqft_lot
  df1$basement_living_ratio <- df1$sqft_basement/df1$sqft_living

  return(df1)
}

train_data <- size_features(train_data)
test_data <- size_features(test_data)

train_data_sub <- feat_select(train_data)
test_data_sub <- feat_select(test_data)

#train_data_enc <- encode(train_data_sub, test_data_sub)

train_val <- f_partition(train_data_sub, seed = 1414)

baseline <- lm(price ~ ., data= scale_df(train_val$train))

test_lm<-predict(baseline, newdata = scale_df(train_val$test))
```

```
## Warning in predict.lm(baseline, newdata = scale_df(train_val$test)):
## prediction from a rank-deficient fit may be misleading
```

```r
mape_lm<-mape(real=train_val$test$price, predicted = test_lm)
mape_lm
```

```
## [1] 0.1930074
```

These new features show some improvement to our MAPE score so we will keep them.

Clustering

The final step in our feature engineering process will be to create clusters based on location and size feature from our dataset

```
clustering <- function(df_train, df_test){

  df_train$train <- 1
  df_test$train <- 0

  df1 <- rbind(df_train, df_test, fill = TRUE)

  set.seed(1912)
  clusters <- kmeans(scale(df1[,c("lat", "long", "living_lot_ratio", "basement_living_ratio")]),
50)

  df1$cluster <- as.factor(clusters$cluster)

  df_train <- df1[df1$train == 1, -'train']
  df_test <- df1[df1$train == 0, -'train']

  return(list("train" = df_train, "test" = df_test))

}

clustered <- clustering(train_data, test_data)

train_data <- clustered$train
test_data <- clustered$test

train_data_sub <- feat_select(train_data)
test_data_sub <- feat_select(test_data)

#train_data_enc <- encode(train_data_sub, test_data_sub)

train_val <- f_partition(train_data_sub, seed = 1414)

baseline <- lm(price ~ ., data= scale_df(train_val$train))

test_lm<-predict(baseline, newdata = scale_df(train_val$test))
```

```
## Warning in predict.lm(baseline, newdata = scale_df(train_val$test)):
## prediction from a rank-deficient fit may be misleading
```

```
mape_lm<-mape(real=train_val$test$price, predicted = test_lm)
mape_lm
```

```
## [1] 0.1914659
```

The score after the clustering show also some improvement so we will keep these new features.

Now that we have an improved dataset with new features, we will proceed to try different models and test them on our validation set, and later do cross validation, to select a model that can predict best the house prices with the features from our dataset.

Modeling

Linear model with stepwise feature selection

First we will try a linear regression with stepwise feature selection. This will help us both improve our score and also detect which variables are the most important ones to be used for our analysis.

```
stepwise <- stepAIC(lm(price ~ ., data= scale_df(train_val$train)), trace = F)

summary(stepwise)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + floors +
##     waterfront + view + condition + grade + sqft_basement + yr_built +
##     yr_renovated + zipcode + lat + long + house_age + renov_age +
##     above_lot_ratio + basement_living_ratio + cluster, data = scale_df(train_val$train))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1300125   -65673     1294    60772  3927299
##
## Coefficients:
##                        Estimate Std. Error t value             Pr(>|t|)
## (Intercept)            381967.0    49518.3   7.714 0.000000000000013069
## bedrooms               -23185.9     1782.5 -13.007 < 0.0000000000000002
## bathrooms               17989.9     2495.5   7.209 0.000000000000593150
## sqft_living            180539.6     3437.7  52.518 < 0.0000000000000002
## floors1.5              -23933.8     5554.9  -4.309 0.000016544212779268
## floors2                -44207.1     4828.9  -9.155 < 0.0000000000000002
## floors2.5              117620.1    17405.6   6.758 0.000000000014598395
## floors3                -49723.6    13668.4  -3.638             0.000276
## floors3.5              187147.9    78976.1   2.370             0.017817
## waterfront              54106.3     1707.8  31.681 < 0.0000000000000002
## view1                   67656.9    10949.9   6.179 0.000000000664417746
## view2                   58836.3     6764.7   8.698 < 0.0000000000000002
## view3                  149102.4     9166.4  16.266 < 0.0000000000000002
## view4                  259006.6    14460.2  17.912 < 0.0000000000000002
## condition2               5460.8    42036.0   0.130             0.896641
## condition3              11600.0    39309.7   0.295             0.767927
## condition4              31686.4    39312.1   0.806             0.420243
## condition5              80607.8    39532.5   2.039             0.041467
## grade                   69340.4     2550.6  27.186 < 0.0000000000000002
## sqft_basement           34319.9     4781.9   7.177 0.000000000000749175
## yr_built               812679.4    84025.9   9.672 < 0.0000000000000002
## yr_renovated            18714.8     2139.7   8.747 < 0.0000000000000002
## zipcode98002            48188.7    17449.2   2.762             0.005758
## zipcode98003           -27987.2    16830.4  -1.663             0.096356
## zipcode98004           704821.6    31616.6  22.293 < 0.0000000000000002
## zipcode98005           245177.4    33693.5   7.277 0.000000000000360701
## zipcode98006           229503.9    28491.5   8.055 0.000000000000000859
## zipcode98007           199287.9    34725.0   5.739 0.000000009724289506
## zipcode98008           222207.4    33561.1   6.621 0.000000000037016086
## zipcode98010           104970.5    30631.6   3.427             0.000612
## zipcode98011            38726.6    41024.9   0.944             0.345197
## zipcode98014           114038.2    47568.0   2.397             0.016526
## zipcode98019            76434.5    46895.7   1.630             0.103150
## zipcode98022            49520.2    28207.7   1.756             0.079186
## zipcode98023           -72650.2    17387.7  -4.178 0.000029557473030755
## zipcode98024           177984.2    40210.8   4.426 0.000009661435129006
## zipcode98027           164649.6    30033.9   5.482 0.000000042768678018
## zipcode98028            23070.5    39388.1   0.586             0.558070
## zipcode98029           213099.7    33689.2   6.325 0.000000000260346650
## zipcode98030             5938.9    20660.5   0.287             0.773769
```

```
## zipcode98031              4995.3    21284.8   0.235              0.814453
## zipcode98032             -1735.9    22784.6  -0.076              0.939271
## zipcode98033            299342.6    34758.7   8.612 < 0.0000000000000002
## zipcode98034            120916.2    37146.9   3.255              0.001136
## zipcode98038             61962.1    24582.9   2.521              0.011729
## zipcode98039           1087160.6    41626.1  26.117 < 0.0000000000000002
## zipcode98040            410839.2    28509.2  14.411 < 0.0000000000000002
## zipcode98042              8806.7    20447.0   0.431              0.666688
## zipcode98045            126391.6    42395.4   2.981              0.002876
## zipcode98052            183401.9    35526.3   5.162 0.000000247177020787
## zipcode98053            155205.2    39089.2   3.971 0.000072078122458905
## zipcode98055             24109.2    23829.8   1.012              0.311687
## zipcode98056             65140.5    25747.5   2.530              0.011418
## zipcode98058             15708.2    23230.2   0.676              0.498926
## zipcode98059             74880.3    25704.6   2.913              0.003584
## zipcode98065            114904.9    43101.2   2.666              0.007686
## zipcode98070           -148150.0    27276.0  -5.432 0.000000056833025532
## zipcode98072             79839.9    40776.4   1.958              0.050251
## zipcode98074            152525.3    35071.8   4.349 0.000013778132218384
## zipcode98075            152818.7    34166.9   4.473 0.000007785859487564
## zipcode98077             64496.5    43075.1   1.497              0.134337
## zipcode98092            -12946.0    17210.3  -0.752              0.451931
## zipcode98102            423848.9    36497.5  11.613 < 0.0000000000000002
## zipcode98103            234445.8    34048.6   6.886 0.000000000006004539
## zipcode98105            375486.7    34666.4  10.831 < 0.0000000000000002
## zipcode98106             43913.1    26999.4   1.626              0.103877
## zipcode98107            242089.4    35215.6   6.874 0.000000000006490685
## zipcode98108             32783.3    28927.6   1.133              0.257112
## zipcode98109            431704.7    36607.1  11.793 < 0.0000000000000002
## zipcode98112            505307.9    32276.7  15.655 < 0.0000000000000002
## zipcode98115            242236.1    34293.2   7.064 0.000000000001699872
## zipcode98116            164613.1    29329.4   5.613 0.000000020322786812
## zipcode98117            197166.5    34943.5   5.642 0.000000017098877622
## zipcode98118             88502.1    26069.3   3.395              0.000689
## zipcode98119            399249.1    34413.7  11.601 < 0.0000000000000002
## zipcode98122            250499.8    30986.4   8.084 0.000000000000000678
## zipcode98125             93752.5    36673.5   2.556              0.010586
## zipcode98126             75886.7    27262.5   2.784              0.005384
## zipcode98133             49971.5    37826.1   1.321              0.186495
## zipcode98136            129488.5    28148.7   4.600 0.000004259570277919
## zipcode98144            197626.3    29314.7   6.742 0.000000000016299915
## zipcode98146            -34347.1    26083.8  -1.317              0.187928
## zipcode98148            -18438.9    29144.8  -0.633              0.526963
## zipcode98155             27728.3    39147.3   0.708              0.478767
## zipcode98166            -53032.0    24295.6  -2.183              0.029069
## zipcode98168            -30658.8    24937.3  -1.229              0.218929
## zipcode98177             69443.7    39658.7   1.751              0.079963
## zipcode98178            -27641.4    25609.7  -1.079              0.280458
## zipcode98188            -45238.6    25619.0  -1.766              0.077448
## zipcode98198            -39161.4    18931.4  -2.069              0.038603
## zipcode98199            256518.4    33742.9   7.602 0.000000000000031021
## lat                      26420.8    11627.8   2.272              0.023089
## long                    -49929.2     9165.4  -5.448 0.000000051951838172
## house_age               819517.1    83979.9   9.758 < 0.0000000000000002
```

```
## renov_age              -10950.5     2062.5  -5.309 0.000000111675975068
## above_lot_ratio        -39748.3     4190.2  -9.486 < 0.0000000000000002
## basement_living_ratio  -81042.3     7106.4 -11.404 < 0.0000000000000002
## cluster2               -16482.6    26116.8  -0.631             0.527978
## cluster3               165419.8    34346.1   4.816 0.000001478355440799
## cluster4                 3254.5    23971.2   0.136             0.892008
## cluster5                47518.6    37158.2   1.279             0.200983
## cluster6                 7463.6    27710.4   0.269             0.787671
## cluster7                10993.9    24494.3   0.449             0.653559
## cluster8                 3570.5    19415.6   0.184             0.854094
## cluster9                15338.2    22241.1   0.690             0.490437
## cluster10              -16643.0    25715.3  -0.647             0.517513
## cluster11               40744.1    20746.3   1.964             0.049560
## cluster12              -17000.4    25619.9  -0.664             0.506981
## cluster13               28403.3    27624.0   1.028             0.303869
## cluster14              106408.4    20459.8   5.201 0.000000201252344549
## cluster15               32937.6    23812.7   1.383             0.166628
## cluster16               -7727.2    20751.4  -0.372             0.709624
## cluster17               48956.1    24391.3   2.007             0.044757
## cluster18               54918.2    20958.2   2.620             0.008793
## cluster19               85973.0    36964.9   2.326             0.020044
## cluster20                8362.2    29610.1   0.282             0.777633
## cluster21              -41296.2    27588.7  -1.497             0.134454
## cluster22               69571.4    21540.9   3.230             0.001242
## cluster23               -6762.3    23550.5  -0.287             0.774010
## cluster24               -3137.5    19761.0  -0.159             0.873851
## cluster25              -39680.2    28759.5  -1.380             0.167694
## cluster26                9837.8    23088.8   0.426             0.670052
## cluster27               73561.2    20447.8   3.598             0.000322
## cluster28               25584.9    19723.9   1.297             0.194600
## cluster29               87963.4    25755.9   3.415             0.000639
## cluster30                1818.1    19691.4   0.092             0.926436
## cluster31              141112.7    33143.0   4.258 0.000020792194251857
## cluster32              -28737.2    24908.6  -1.154             0.248641
## cluster33               18212.1    19610.6   0.929             0.353070
## cluster34                7294.1    17188.8   0.424             0.671318
## cluster35                 799.7    23230.3   0.034             0.972540
## cluster36               48752.2    24147.3   2.019             0.043512
## cluster37               16933.9    27785.8   0.609             0.542241
## cluster38              -25706.4    27489.6  -0.935             0.349736
## cluster39              -27897.6    23404.0  -1.192             0.233281
## cluster40               25636.4    20357.6   1.259             0.207942
## cluster41               22548.1    15417.3   1.463             0.143621
## cluster42               -7636.4    21628.1  -0.353             0.724035
## cluster43               87567.6    22397.4   3.910 0.000092846586191939
## cluster44               13039.0    28518.3   0.457             0.647525
## cluster45               10176.2    26104.7   0.390             0.696673
## cluster46              -23716.4    19617.8  -1.209             0.226712
## cluster47               20186.7    18634.2   1.083             0.278689
## cluster48               48915.2    20194.0   2.422             0.015437
## cluster49               28112.4    23380.4   1.202             0.229231
## cluster50               -1000.9    23890.9  -0.042             0.966583
##                           
## (Intercept)            ***
```

```
## bedrooms             ***
## bathrooms            ***
## sqft_living          ***
## floors1.5            ***
## floors2              ***
## floors2.5            ***
## floors3              ***
## floors3.5            *
## waterfront           ***
## view1                ***
## view2                ***
## view3                ***
## view4                ***
## condition2
## condition3
## condition4
## condition5           *
## grade                ***
## sqft_basement        ***
## yr_built             ***
## yr_renovated         ***
## zipcode98002         **
## zipcode98003         .
## zipcode98004         ***
## zipcode98005         ***
## zipcode98006         ***
## zipcode98007         ***
## zipcode98008         ***
## zipcode98010         ***
## zipcode98011
## zipcode98014         *
## zipcode98019
## zipcode98022         .
## zipcode98023         ***
## zipcode98024         ***
## zipcode98027         ***
## zipcode98028
## zipcode98029         ***
## zipcode98030
## zipcode98031
## zipcode98032
## zipcode98033         ***
## zipcode98034         **
## zipcode98038         *
## zipcode98039         ***
## zipcode98040         ***
## zipcode98042
## zipcode98045         **
## zipcode98052         ***
## zipcode98053         ***
## zipcode98055
## zipcode98056         *
## zipcode98058
## zipcode98059         **
```

```
## zipcode98065         **
## zipcode98070         ***
## zipcode98072         .
## zipcode98074         ***
## zipcode98075         ***
## zipcode98077
## zipcode98092
## zipcode98102         ***
## zipcode98103         ***
## zipcode98105         ***
## zipcode98106
## zipcode98107         ***
## zipcode98108
## zipcode98109         ***
## zipcode98112         ***
## zipcode98115         ***
## zipcode98116         ***
## zipcode98117         ***
## zipcode98118         ***
## zipcode98119         ***
## zipcode98122         ***
## zipcode98125         *
## zipcode98126         **
## zipcode98133
## zipcode98136         ***
## zipcode98144         ***
## zipcode98146
## zipcode98148
## zipcode98155
## zipcode98166         *
## zipcode98168
## zipcode98177         .
## zipcode98178
## zipcode98188         .
## zipcode98198         *
## zipcode98199         ***
## lat                  *
## long                 ***
## house_age            ***
## renov_age            ***
## above_lot_ratio      ***
## basement_living_ratio ***
## cluster2
## cluster3             ***
## cluster4
## cluster5
## cluster6
## cluster7
## cluster8
## cluster9
## cluster10
## cluster11            *
## cluster12
## cluster13
```

```
## cluster14             ***
## cluster15
## cluster16
## cluster17             *
## cluster18             **
## cluster19             *
## cluster20
## cluster21
## cluster22             **
## cluster23
## cluster24
## cluster25
## cluster26
## cluster27             ***
## cluster28
## cluster29             ***
## cluster30
## cluster31             ***
## cluster32
## cluster33
## cluster34
## cluster35
## cluster36             *
## cluster37
## cluster38
## cluster39
## cluster40
## cluster41
## cluster42
## cluster43             ***
## cluster44
## cluster45
## cluster46
## cluster47
## cluster48             *
## cluster49
## cluster50
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 155400 on 13675 degrees of freedom
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8196
## F-statistic:   434 on 145 and 13675 DF,  p-value: < 0.00000000000000022
```

```
test_lms<-predict(stepwise, newdata = scale_df(train_val$test))
mape_lms<-mape(real=train_val$test$price, predicted = test_lms)
mape_lms
```

```
## [1] 0.1914916
```

This model did not improve much our score so we will try with another family of models (random forest and XGBoost)

Random Forest Regression

```
RF <- randomForest(price ~ ., data= train_val$train[,-c('zipcode')])



test_rf<-predict(RF, newdata = train_val$test[,-c('zipcode')])
mape_rf<-mape(real=train_val$test$price, predicted = test_rf)
mape_rf
```

```
## [1] 0.1393642
```

Random Forest already shows to be a much better model than linear regression. Now we will compare this model's performance with XGBoost:

XGBoost (tree)

```
train_data_enc <- encode(train_data_sub, test_data_sub)
train_val <- f_partition(train_data_enc$train, seed = 1414)

xgb_0<-xgboost(booster='gbtree',
               data=as.matrix(train_val$train[, -c('price'), with = F]),
               label=train_val$train$price,
               nrounds = 200,
               objective='reg:linear')
```

```
## [1]  train-rmse:473789.031250
## [2]  train-rmse:351371.875000
## [3]  train-rmse:266862.781250
## [4]  train-rmse:210026.515625
## [5]  train-rmse:172137.828125
## [6]  train-rmse:146907.250000
## [7]  train-rmse:131609.765625
## [8]  train-rmse:121147.132812
## [9]  train-rmse:113967.750000
## [10] train-rmse:109222.242188
## [11] train-rmse:106099.851562
## [12] train-rmse:103466.726562
## [13] train-rmse:100932.640625
## [14] train-rmse:99239.210938
## [15] train-rmse:97386.609375
## [16] train-rmse:95306.500000
## [17] train-rmse:93860.914062
## [18] train-rmse:92137.867188
## [19] train-rmse:91319.453125
## [20] train-rmse:89978.562500
## [21] train-rmse:89192.656250
## [22] train-rmse:87909.101562
## [23] train-rmse:86798.828125
## [24] train-rmse:86128.835938
## [25] train-rmse:85408.023438
## [26] train-rmse:84838.953125
## [27] train-rmse:83814.890625
## [28] train-rmse:83377.234375
## [29] train-rmse:82497.523438
## [30] train-rmse:82159.820312
## [31] train-rmse:81364.164062
## [32] train-rmse:80853.546875
## [33] train-rmse:80371.421875
## [34] train-rmse:79901.835938
## [35] train-rmse:78588.664062
## [36] train-rmse:77774.710938
## [37] train-rmse:77077.117188
## [38] train-rmse:76199.179688
## [39] train-rmse:75790.492188
## [40] train-rmse:74668.507812
## [41] train-rmse:74293.781250
## [42] train-rmse:73751.773438
## [43] train-rmse:73350.132812
## [44] train-rmse:72954.171875
## [45] train-rmse:72614.031250
## [46] train-rmse:72235.031250
## [47] train-rmse:71976.101562
## [48] train-rmse:71547.273438
## [49] train-rmse:71251.390625
## [50] train-rmse:70807.109375
## [51] train-rmse:70380.164062
## [52] train-rmse:69804.546875
## [53] train-rmse:69199.429688
```

```
## [54] train-rmse:68787.609375
## [55] train-rmse:68516.406250
## [56] train-rmse:68081.765625
## [57] train-rmse:67716.843750
## [58] train-rmse:67344.562500
## [59] train-rmse:66918.960938
## [60] train-rmse:66717.812500
## [61] train-rmse:66479.546875
## [62] train-rmse:66181.437500
## [63] train-rmse:65704.335938
## [64] train-rmse:65526.847656
## [65] train-rmse:64901.535156
## [66] train-rmse:64562.789062
## [67] train-rmse:64312.882812
## [68] train-rmse:63945.535156
## [69] train-rmse:63788.687500
## [70] train-rmse:63669.523438
## [71] train-rmse:63330.351562
## [72] train-rmse:62930.242188
## [73] train-rmse:62647.652344
## [74] train-rmse:62435.625000
## [75] train-rmse:62018.207031
## [76] train-rmse:61696.101562
## [77] train-rmse:61430.570312
## [78] train-rmse:61299.855469
## [79] train-rmse:60988.093750
## [80] train-rmse:60636.531250
## [81] train-rmse:60484.207031
## [82] train-rmse:60056.117188
## [83] train-rmse:59890.898438
## [84] train-rmse:59630.875000
## [85] train-rmse:59212.617188
## [86] train-rmse:58990.972656
## [87] train-rmse:58914.523438
## [88] train-rmse:58815.738281
## [89] train-rmse:58679.941406
## [90] train-rmse:58335.257812
## [91] train-rmse:58000.582031
## [92] train-rmse:57770.875000
## [93] train-rmse:57683.148438
## [94] train-rmse:57446.726562
## [95] train-rmse:57272.789062
## [96] train-rmse:57012.554688
## [97] train-rmse:56924.316406
## [98] train-rmse:56732.238281
## [99] train-rmse:56482.246094
## [100]    train-rmse:56317.300781
## [101]    train-rmse:56073.582031
## [102]    train-rmse:55913.800781
## [103]    train-rmse:55772.304688
## [104]    train-rmse:55527.863281
## [105]    train-rmse:55264.070312
## [106]    train-rmse:55154.003906
## [107]    train-rmse:54823.808594
```

```
## [108]	train-rmse:54500.562500
## [109]	train-rmse:54183.511719
## [110]	train-rmse:53984.726562
## [111]	train-rmse:53834.500000
## [112]	train-rmse:53725.480469
## [113]	train-rmse:53646.316406
## [114]	train-rmse:53446.484375
## [115]	train-rmse:53196.007812
## [116]	train-rmse:52959.382812
## [117]	train-rmse:52815.273438
## [118]	train-rmse:52644.183594
## [119]	train-rmse:52306.523438
## [120]	train-rmse:52066.472656
## [121]	train-rmse:51909.792969
## [122]	train-rmse:51799.167969
## [123]	train-rmse:51639.886719
## [124]	train-rmse:51414.718750
## [125]	train-rmse:51297.722656
## [126]	train-rmse:51046.523438
## [127]	train-rmse:51000.660156
## [128]	train-rmse:50822.277344
## [129]	train-rmse:50750.937500
## [130]	train-rmse:50617.863281
## [131]	train-rmse:50404.070312
## [132]	train-rmse:50254.863281
## [133]	train-rmse:50081.511719
## [134]	train-rmse:49991.187500
## [135]	train-rmse:49737.097656
## [136]	train-rmse:49630.875000
## [137]	train-rmse:49374.660156
## [138]	train-rmse:49226.585938
## [139]	train-rmse:49122.039062
## [140]	train-rmse:49012.878906
## [141]	train-rmse:48837.867188
## [142]	train-rmse:48723.027344
## [143]	train-rmse:48477.382812
## [144]	train-rmse:48358.441406
## [145]	train-rmse:48277.535156
## [146]	train-rmse:48030.527344
## [147]	train-rmse:47880.421875
## [148]	train-rmse:47733.164062
## [149]	train-rmse:47565.167969
## [150]	train-rmse:47417.531250
## [151]	train-rmse:47252.492188
## [152]	train-rmse:46994.765625
## [153]	train-rmse:46788.894531
## [154]	train-rmse:46687.359375
## [155]	train-rmse:46572.898438
## [156]	train-rmse:46364.085938
## [157]	train-rmse:46094.769531
## [158]	train-rmse:45914.929688
## [159]	train-rmse:45793.750000
## [160]	train-rmse:45729.089844
## [161]	train-rmse:45687.429688
```

```
## [162]	train-rmse:45589.828125
## [163]	train-rmse:45415.964844
## [164]	train-rmse:45202.183594
## [165]	train-rmse:45145.261719
## [166]	train-rmse:45080.460938
## [167]	train-rmse:44905.625000
## [168]	train-rmse:44826.906250
## [169]	train-rmse:44692.488281
## [170]	train-rmse:44570.601562
## [171]	train-rmse:44433.984375
## [172]	train-rmse:44402.554688
## [173]	train-rmse:44248.851562
## [174]	train-rmse:44105.398438
## [175]	train-rmse:44002.035156
## [176]	train-rmse:43907.902344
## [177]	train-rmse:43580.765625
## [178]	train-rmse:43439.250000
## [179]	train-rmse:43350.074219
## [180]	train-rmse:43249.957031
## [181]	train-rmse:43216.414062
## [182]	train-rmse:43129.339844
## [183]	train-rmse:42994.515625
## [184]	train-rmse:42838.886719
## [185]	train-rmse:42781.218750
## [186]	train-rmse:42633.312500
## [187]	train-rmse:42510.886719
## [188]	train-rmse:42355.644531
## [189]	train-rmse:42266.628906
## [190]	train-rmse:42160.734375
## [191]	train-rmse:42060.175781
## [192]	train-rmse:41870.968750
## [193]	train-rmse:41834.578125
## [194]	train-rmse:41701.714844
## [195]	train-rmse:41528.265625
## [196]	train-rmse:41398.449219
## [197]	train-rmse:41366.851562
## [198]	train-rmse:41274.835938
## [199]	train-rmse:41132.949219
## [200]	train-rmse:40995.863281
```

```
test_xgb<-predict(xgb_0, newdata = as.matrix(train_val$test[, !'price', with=F]), type='respons
e')
mape_rf<-mape(real=train_val$test$price, predicted = test_xgb)
mape_rf
```

```
## [1] 0.1293443
```

Using XGBoost with 200 rounds already proved to perform better than Random Forest with a MAPE score on the holdout of 12.9

XGBoost - log scale (tree)

Now we will try the same model but predicting on the target variable in a logarithmic scale:

```
xgb_0_log<-xgboost(booster='gbtree',
               data=as.matrix(train_val$train[, -c('price'), with = F]),
               label= log(train_val$train$price),
               nrounds = 200,
               objective='reg:linear')
```

```
## [1]  train-rmse:8.795152
## [2]  train-rmse:6.161392
## [3]  train-rmse:4.318283
## [4]  train-rmse:3.029167
## [5]  train-rmse:2.127673
## [6]  train-rmse:1.498084
## [7]  train-rmse:1.059144
## [8]  train-rmse:0.754962
## [9]  train-rmse:0.546143
## [10] train-rmse:0.404825
## [11] train-rmse:0.311152
## [12] train-rmse:0.251701
## [13] train-rmse:0.213532
## [14] train-rmse:0.192525
## [15] train-rmse:0.180899
## [16] train-rmse:0.173805
## [17] train-rmse:0.169336
## [18] train-rmse:0.165615
## [19] train-rmse:0.162838
## [20] train-rmse:0.160766
## [21] train-rmse:0.159505
## [22] train-rmse:0.157872
## [23] train-rmse:0.156711
## [24] train-rmse:0.155705
## [25] train-rmse:0.154409
## [26] train-rmse:0.153335
## [27] train-rmse:0.152550
## [28] train-rmse:0.151836
## [29] train-rmse:0.150905
## [30] train-rmse:0.149989
## [31] train-rmse:0.148087
## [32] train-rmse:0.147041
## [33] train-rmse:0.146170
## [34] train-rmse:0.145264
## [35] train-rmse:0.144125
## [36] train-rmse:0.143359
## [37] train-rmse:0.142124
## [38] train-rmse:0.141606
## [39] train-rmse:0.140913
## [40] train-rmse:0.140226
## [41] train-rmse:0.139189
## [42] train-rmse:0.138524
## [43] train-rmse:0.137912
## [44] train-rmse:0.137567
## [45] train-rmse:0.137013
## [46] train-rmse:0.136607
## [47] train-rmse:0.136024
## [48] train-rmse:0.135581
## [49] train-rmse:0.134809
## [50] train-rmse:0.134274
## [51] train-rmse:0.133923
## [52] train-rmse:0.133246
## [53] train-rmse:0.132092
```

```
## [54] train-rmse:0.131063
## [55] train-rmse:0.130625
## [56] train-rmse:0.129866
## [57] train-rmse:0.129385
## [58] train-rmse:0.128959
## [59] train-rmse:0.128155
## [60] train-rmse:0.127297
## [61] train-rmse:0.126874
## [62] train-rmse:0.126184
## [63] train-rmse:0.125589
## [64] train-rmse:0.125161
## [65] train-rmse:0.125029
## [66] train-rmse:0.124515
## [67] train-rmse:0.123910
## [68] train-rmse:0.123455
## [69] train-rmse:0.123200
## [70] train-rmse:0.122986
## [71] train-rmse:0.122786
## [72] train-rmse:0.122096
## [73] train-rmse:0.121540
## [74] train-rmse:0.120984
## [75] train-rmse:0.120499
## [76] train-rmse:0.120339
## [77] train-rmse:0.120118
## [78] train-rmse:0.119521
## [79] train-rmse:0.119373
## [80] train-rmse:0.118846
## [81] train-rmse:0.118354
## [82] train-rmse:0.117955
## [83] train-rmse:0.117704
## [84] train-rmse:0.117398
## [85] train-rmse:0.116906
## [86] train-rmse:0.116442
## [87] train-rmse:0.115790
## [88] train-rmse:0.114642
## [89] train-rmse:0.114341
## [90] train-rmse:0.114076
## [91] train-rmse:0.113889
## [92] train-rmse:0.113491
## [93] train-rmse:0.113327
## [94] train-rmse:0.113138
## [95] train-rmse:0.112863
## [96] train-rmse:0.112813
## [97] train-rmse:0.112323
## [98] train-rmse:0.111967
## [99] train-rmse:0.111285
## [100]    train-rmse:0.110792
## [101]    train-rmse:0.110395
## [102]    train-rmse:0.110132
## [103]    train-rmse:0.109957
## [104]    train-rmse:0.109777
## [105]    train-rmse:0.109352
## [106]    train-rmse:0.109082
## [107]    train-rmse:0.108772
```

```
## [108]	train-rmse:0.108620
## [109]	train-rmse:0.108285
## [110]	train-rmse:0.107886
## [111]	train-rmse:0.107679
## [112]	train-rmse:0.107431
## [113]	train-rmse:0.107226
## [114]	train-rmse:0.107138
## [115]	train-rmse:0.106774
## [116]	train-rmse:0.106496
## [117]	train-rmse:0.106277
## [118]	train-rmse:0.105982
## [119]	train-rmse:0.105814
## [120]	train-rmse:0.105616
## [121]	train-rmse:0.105386
## [122]	train-rmse:0.105133
## [123]	train-rmse:0.104753
## [124]	train-rmse:0.104571
## [125]	train-rmse:0.104521
## [126]	train-rmse:0.104288
## [127]	train-rmse:0.104116
## [128]	train-rmse:0.103711
## [129]	train-rmse:0.103565
## [130]	train-rmse:0.103429
## [131]	train-rmse:0.103101
## [132]	train-rmse:0.102547
## [133]	train-rmse:0.102282
## [134]	train-rmse:0.101891
## [135]	train-rmse:0.101690
## [136]	train-rmse:0.101430
## [137]	train-rmse:0.101047
## [138]	train-rmse:0.100889
## [139]	train-rmse:0.100703
## [140]	train-rmse:0.100532
## [141]	train-rmse:0.100398
## [142]	train-rmse:0.100023
## [143]	train-rmse:0.099943
## [144]	train-rmse:0.099594
## [145]	train-rmse:0.099281
## [146]	train-rmse:0.098725
## [147]	train-rmse:0.098160
## [148]	train-rmse:0.097852
## [149]	train-rmse:0.097695
## [150]	train-rmse:0.097474
## [151]	train-rmse:0.096901
## [152]	train-rmse:0.096632
## [153]	train-rmse:0.096516
## [154]	train-rmse:0.096221
## [155]	train-rmse:0.096162
## [156]	train-rmse:0.095835
## [157]	train-rmse:0.095455
## [158]	train-rmse:0.095356
## [159]	train-rmse:0.095043
## [160]	train-rmse:0.094855
## [161]	train-rmse:0.094512
```

```
## [162]    train-rmse:0.094340
## [163]    train-rmse:0.094025
## [164]    train-rmse:0.093574
## [165]    train-rmse:0.093309
## [166]    train-rmse:0.092937
## [167]    train-rmse:0.092428
## [168]    train-rmse:0.092267
## [169]    train-rmse:0.091930
## [170]    train-rmse:0.091686
## [171]    train-rmse:0.091519
## [172]    train-rmse:0.091343
## [173]    train-rmse:0.091222
## [174]    train-rmse:0.090940
## [175]    train-rmse:0.090590
## [176]    train-rmse:0.090524
## [177]    train-rmse:0.090260
## [178]    train-rmse:0.090008
## [179]    train-rmse:0.089892
## [180]    train-rmse:0.089695
## [181]    train-rmse:0.089533
## [182]    train-rmse:0.089361
## [183]    train-rmse:0.089104
## [184]    train-rmse:0.088754
## [185]    train-rmse:0.088368
## [186]    train-rmse:0.088046
## [187]    train-rmse:0.087925
## [188]    train-rmse:0.087635
## [189]    train-rmse:0.087498
## [190]    train-rmse:0.087365
## [191]    train-rmse:0.087121
## [192]    train-rmse:0.086864
## [193]    train-rmse:0.086679
## [194]    train-rmse:0.086465
## [195]    train-rmse:0.086155
## [196]    train-rmse:0.085647
## [197]    train-rmse:0.085473
## [198]    train-rmse:0.085324
## [199]    train-rmse:0.085035
## [200]    train-rmse:0.084830
```

```
test_xgb_log<-predict(xgb_0, newdata = as.matrix(train_val$test[, !'price', with=F]), type='resp
onse')
mape_rf_log<-mape(real=train_val$test$price, predicted = exp(test_xgb_log))
mape_rf_log
```

```
## [1] Inf
```

Boosting Regression

```
xgb_1<-xgboost(booster='gblinear',
               data=as.matrix(train_val$train[, -c('price'), with = F]),
               label=train_val$train$price,
               nrounds = 200,
               objective='reg:linear')
```

```
## [1]  train-rmse:226284.421875
## [2]  train-rmse:204104.531250
## [3]  train-rmse:197392.718750
## [4]  train-rmse:193646.437500
## [5]  train-rmse:191035.109375
## [6]  train-rmse:188946.171875
## [7]  train-rmse:187299.234375
## [8]  train-rmse:185758.343750
## [9]  train-rmse:184546.171875
## [10] train-rmse:183248.421875
## [11] train-rmse:182157.625000
## [12] train-rmse:181231.328125
## [13] train-rmse:180189.421875
## [14] train-rmse:179244.187500
## [15] train-rmse:178361.500000
## [16] train-rmse:177543.078125
## [17] train-rmse:176768.593750
## [18] train-rmse:176036.406250
## [19] train-rmse:175385.984375
## [20] train-rmse:174732.125000
## [21] train-rmse:174110.875000
## [22] train-rmse:173525.468750
## [23] train-rmse:172971.453125
## [24] train-rmse:172455.234375
## [25] train-rmse:171981.375000
## [26] train-rmse:171509.046875
## [27] train-rmse:171046.906250
## [28] train-rmse:170602.578125
## [29] train-rmse:170175.234375
## [30] train-rmse:169796.359375
## [31] train-rmse:169394.562500
## [32] train-rmse:169014.734375
## [33] train-rmse:168649.046875
## [34] train-rmse:168296.593750
## [35] train-rmse:167956.234375
## [36] train-rmse:167628.453125
## [37] train-rmse:167311.203125
## [38] train-rmse:167005.140625
## [39] train-rmse:166710.781250
## [40] train-rmse:166457.578125
## [41] train-rmse:166178.578125
## [42] train-rmse:165909.968750
## [43] train-rmse:165649.687500
## [44] train-rmse:165398.234375
## [45] train-rmse:165152.281250
## [46] train-rmse:164917.375000
## [47] train-rmse:164703.750000
## [48] train-rmse:164481.781250
## [49] train-rmse:164266.218750
## [50] train-rmse:164056.281250
## [51] train-rmse:163852.187500
## [52] train-rmse:163655.125000
## [53] train-rmse:163461.531250
```

```
## [54] train-rmse:163273.562500
## [55] train-rmse:163090.812500
## [56] train-rmse:162922.046875
## [57] train-rmse:162765.765625
## [58] train-rmse:162590.890625
## [59] train-rmse:162429.406250
## [60] train-rmse:162271.718750
## [61] train-rmse:162117.953125
## [62] train-rmse:161968.140625
## [63] train-rmse:161822.078125
## [64] train-rmse:161679.890625
## [65] train-rmse:161541.343750
## [66] train-rmse:161407.062500
## [67] train-rmse:161275.625000
## [68] train-rmse:161147.640625
## [69] train-rmse:161023.421875
## [70] train-rmse:160902.718750
## [71] train-rmse:160788.234375
## [72] train-rmse:160673.781250
## [73] train-rmse:160561.968750
## [74] train-rmse:160452.640625
## [75] train-rmse:160346.593750
## [76] train-rmse:160243.328125
## [77] train-rmse:160143.140625
## [78] train-rmse:160045.046875
## [79] train-rmse:159949.687500
## [80] train-rmse:159856.671875
## [81] train-rmse:159766.187500
## [82] train-rmse:159678.078125
## [83] train-rmse:159592.500000
## [84] train-rmse:159508.890625
## [85] train-rmse:159430.218750
## [86] train-rmse:159351.062500
## [87] train-rmse:159273.968750
## [88] train-rmse:159198.953125
## [89] train-rmse:159125.953125
## [90] train-rmse:159058.218750
## [91] train-rmse:158990.375000
## [92] train-rmse:158925.000000
## [93] train-rmse:158860.281250
## [94] train-rmse:158797.187500
## [95] train-rmse:158735.562500
## [96] train-rmse:158674.906250
## [97] train-rmse:158610.625000
## [98] train-rmse:158552.328125
## [99] train-rmse:158496.562500
## [100]    train-rmse:158442.578125
## [101]    train-rmse:158389.750000
## [102]    train-rmse:158337.937500
## [103]    train-rmse:158287.218750
## [104]    train-rmse:158237.906250
## [105]    train-rmse:158189.718750
## [106]    train-rmse:158142.703125
## [107]    train-rmse:158096.828125
```

```
## [108]	train-rmse:158052.125000
## [109]	train-rmse:158008.484375
## [110]	train-rmse:157965.781250
## [111]	train-rmse:157925.359375
## [112]	train-rmse:157884.656250
## [113]	train-rmse:157844.921875
## [114]	train-rmse:157806.250000
## [115]	train-rmse:157768.531250
## [116]	train-rmse:157731.765625
## [117]	train-rmse:157691.921875
## [118]	train-rmse:157657.890625
## [119]	train-rmse:157624.015625
## [120]	train-rmse:157590.984375
## [121]	train-rmse:157558.562500
## [122]	train-rmse:157526.812500
## [123]	train-rmse:157495.812500
## [124]	train-rmse:157465.421875
## [125]	train-rmse:157435.781250
## [126]	train-rmse:157407.343750
## [127]	train-rmse:157378.953125
## [128]	train-rmse:157351.312500
## [129]	train-rmse:157324.328125
## [130]	train-rmse:157297.937500
## [131]	train-rmse:157272.187500
## [132]	train-rmse:157246.953125
## [133]	train-rmse:157222.343750
## [134]	train-rmse:157195.406250
## [135]	train-rmse:157172.312500
## [136]	train-rmse:157149.593750
## [137]	train-rmse:157127.312500
## [138]	train-rmse:157105.437500
## [139]	train-rmse:157083.328125
## [140]	train-rmse:157062.515625
## [141]	train-rmse:157042.109375
## [142]	train-rmse:157022.031250
## [143]	train-rmse:157002.343750
## [144]	train-rmse:156983.015625
## [145]	train-rmse:156964.125000
## [146]	train-rmse:156945.500000
## [147]	train-rmse:156927.406250
## [148]	train-rmse:156909.656250
## [149]	train-rmse:156892.109375
## [150]	train-rmse:156874.937500
## [151]	train-rmse:156858.203125
## [152]	train-rmse:156841.765625
## [153]	train-rmse:156826.187500
## [154]	train-rmse:156810.625000
## [155]	train-rmse:156795.265625
## [156]	train-rmse:156780.140625
## [157]	train-rmse:156763.375000
## [158]	train-rmse:156749.171875
## [159]	train-rmse:156735.062500
## [160]	train-rmse:156721.140625
## [161]	train-rmse:156707.406250
```

```
## [162]	train-rmse:156693.859375
## [163]	train-rmse:156680.562500
## [164]	train-rmse:156666.828125
## [165]	train-rmse:156654.125000
## [166]	train-rmse:156641.625000
## [167]	train-rmse:156629.890625
## [168]	train-rmse:156617.890625
## [169]	train-rmse:156606.156250
## [170]	train-rmse:156594.546875
## [171]	train-rmse:156583.093750
## [172]	train-rmse:156571.796875
## [173]	train-rmse:156560.671875
## [174]	train-rmse:156548.156250
## [175]	train-rmse:156537.625000
## [176]	train-rmse:156527.140625
## [177]	train-rmse:156516.765625
## [178]	train-rmse:156506.484375
## [179]	train-rmse:156496.437500
## [180]	train-rmse:156486.406250
## [181]	train-rmse:156476.656250
## [182]	train-rmse:156466.953125
## [183]	train-rmse:156457.625000
## [184]	train-rmse:156448.218750
## [185]	train-rmse:156438.953125
## [186]	train-rmse:156429.796875
## [187]	train-rmse:156420.781250
## [188]	train-rmse:156411.953125
## [189]	train-rmse:156403.734375
## [190]	train-rmse:156395.078125
## [191]	train-rmse:156385.468750
## [192]	train-rmse:156376.187500
## [193]	train-rmse:156368.171875
## [194]	train-rmse:156360.140625
## [195]	train-rmse:156352.156250
## [196]	train-rmse:156344.343750
## [197]	train-rmse:156336.015625
## [198]	train-rmse:156328.406250
## [199]	train-rmse:156320.828125
## [200]	train-rmse:156313.328125
```

```
test_xgb1<-predict(xgb_1, newdata = as.matrix(train_val$test[, !'price', with=F]), type='respons
e')
mape_rf1<-mape(real=train_val$test$price, predicted = test_xgb1)
mape_rf1
```

```
## [1] 0.1919694
```

After trying several models, boosting tree in the logarihtmic scale seems to be the one showing a better performance.

Thus, this will be the model we will tune with Cross-Validation and Hyper Parameter Tuning before running the model on our test set.

Model Tuning

```
#defining the grid
tune_grid <- expand.grid(
  nrounds = seq(from = 200, to = 500, by = 100),
  eta = c(0.025, 0.05, 0.1),
  max_depth = c(2, 3, 4),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

tune_control <- caret::trainControl(
  method = "cv", # cross-validation
  number = 5, # with n folds
  #index = createFolds(tr_treated$Id_clean), # fix the folds
  verboseIter = FALSE, # no training log
  allowParallel = TRUE # FALSE for reproducible results
)

xgb_tune <- caret::train(
  x = as.matrix(train_val$train[, -c('price'), with = F]),
  y = log(train_val$train$price),
  trControl = tune_control,
  tuneGrid = tune_grid,
  method = "xgbTree",
  verbose = TRUE
)


test_xgb_log_tune<-predict(xgb_tune$finalModel, newdata = as.matrix(train_val$test[, !'price', w
ith=F]), type='response')
mape_rf_log_tune<-mape(real=train_val$test$price, predicted = exp(test_xgb_log_tune))
mape_rf_log_tune
```

```
## [1] 0.1205633
```

The best model after Tuning the XGBoot Tree gives us a MAPE of 12.05 on our validation set. Now we will use this model to train on the whole training data and generate the predictions on our test dataset.

Final Model

```
xgb_tune_final <- caret::train(
  x = as.matrix(train_data_enc$train[, -c('price'), with = F]),
  y = log(train_data_enc$train$price),
  trControl = tune_control,
  tuneGrid = tune_grid,
  method = "xgbTree",
  verbose = TRUE
)


test_final<-predict(xgb_tune_final$finalModel, newdata = as.matrix(train_data_enc$test[, !'pric
e', with=F]), type='response')
```

Exporting final predictions

```
write.csv(exp(test_final), file = "predictions_tomas_tello.csv")
```