Basic Data Exploration and Prediction

```
setwd("C:/Users/baoji/OneDrive/IVLE/ST4240 Data Mining_2017/Predictive Modelling Assignment")


set.seed(1)  #for reproducibility
library(plyr)
library(dplyr, warn.conflicts = FALSE)
library(ggplot2)
library(corrplot)
library(e1071)
library(hexbin)
library(gbm)
```

```
## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.1
```

```
library(xgboost)
```

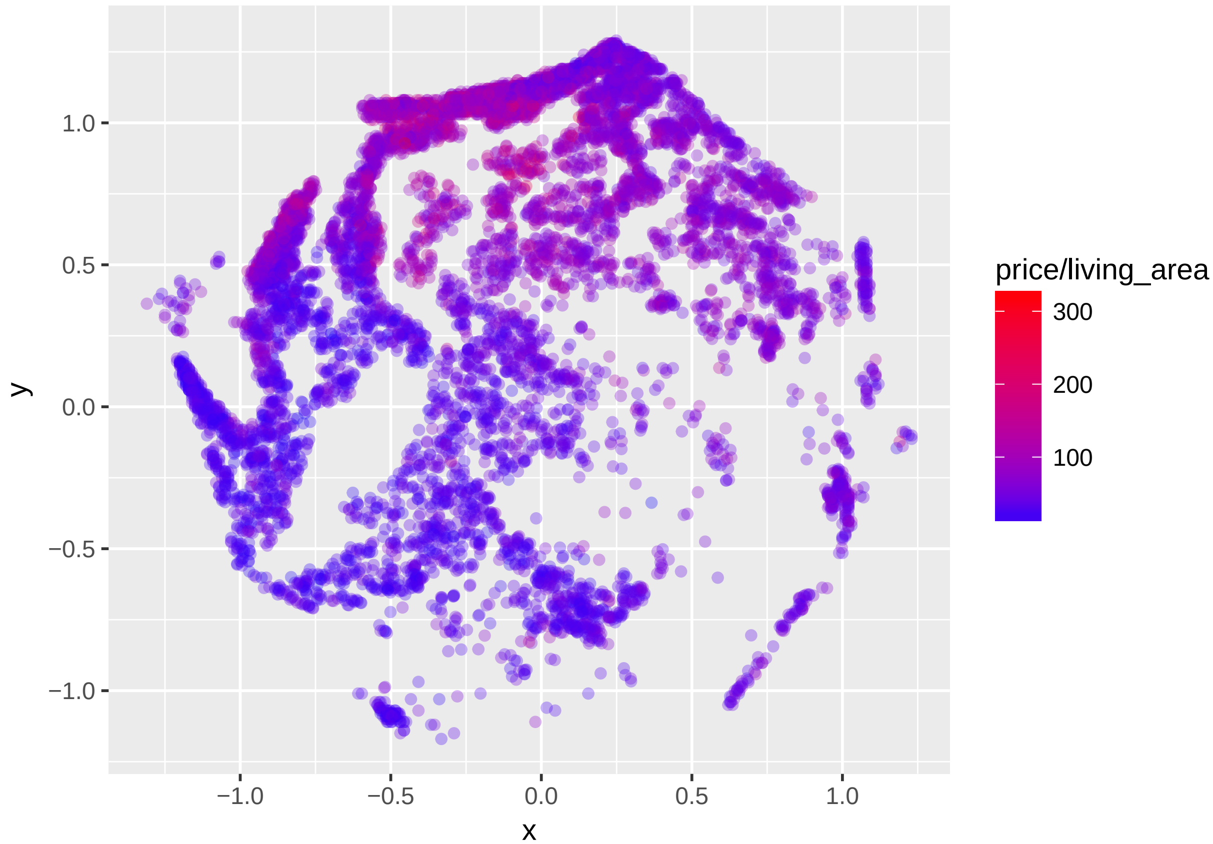
```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(lars)
```

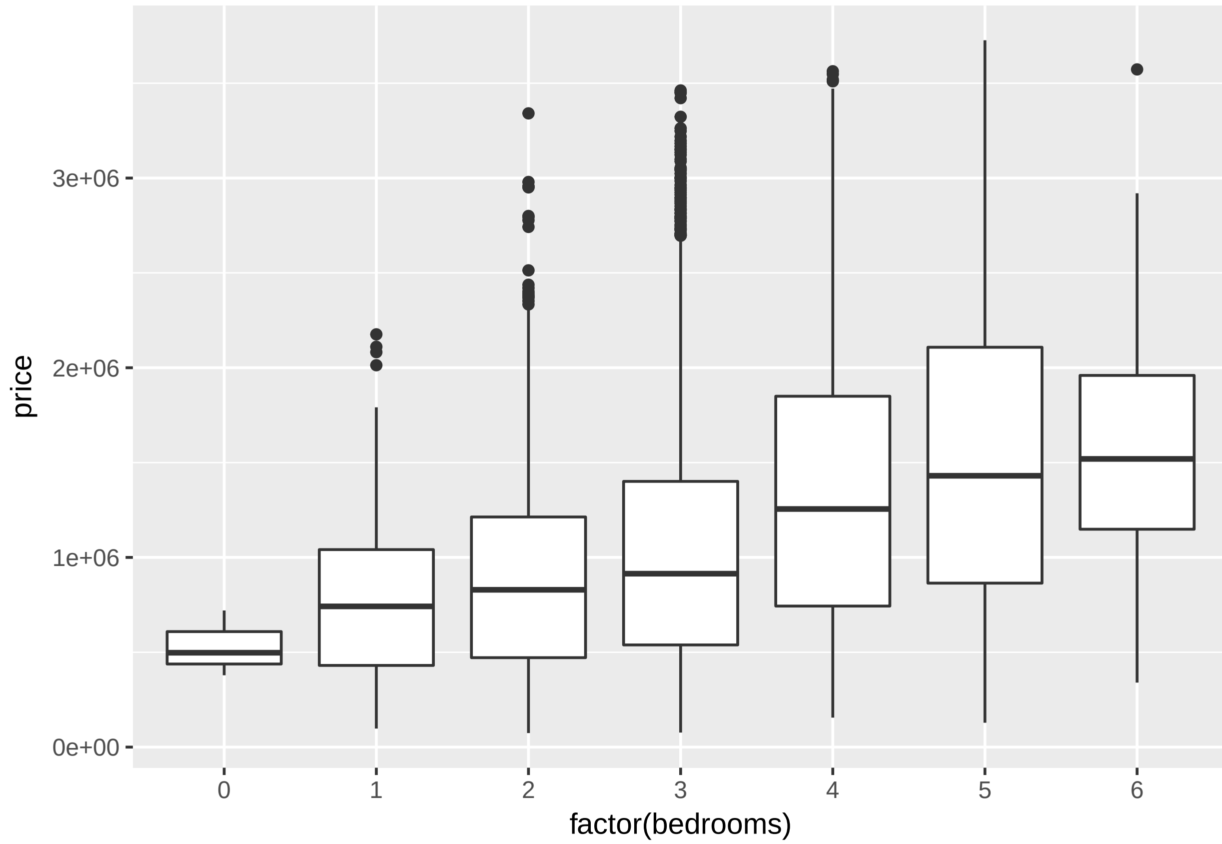
```
## Loaded lars 1.2
```

```
library(Matrix)
```

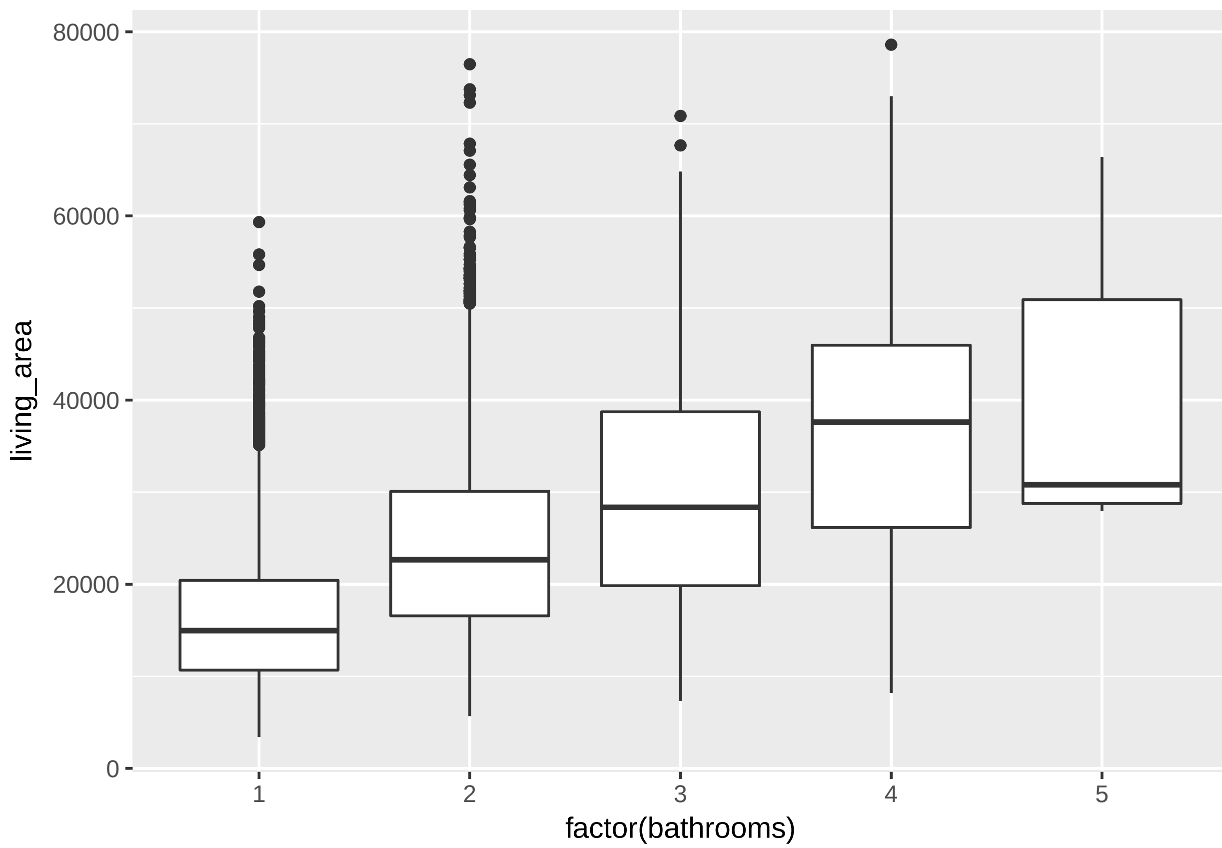
```
filename_train = "train.csv"
data = read.csv(file = filename_train)

data %>%
  ggplot() + geom_point(aes(x=x, y=y, colour=price/living_area),alpha=0.3) +
  scale_colour_gradient(low = "blue", high="red")
```



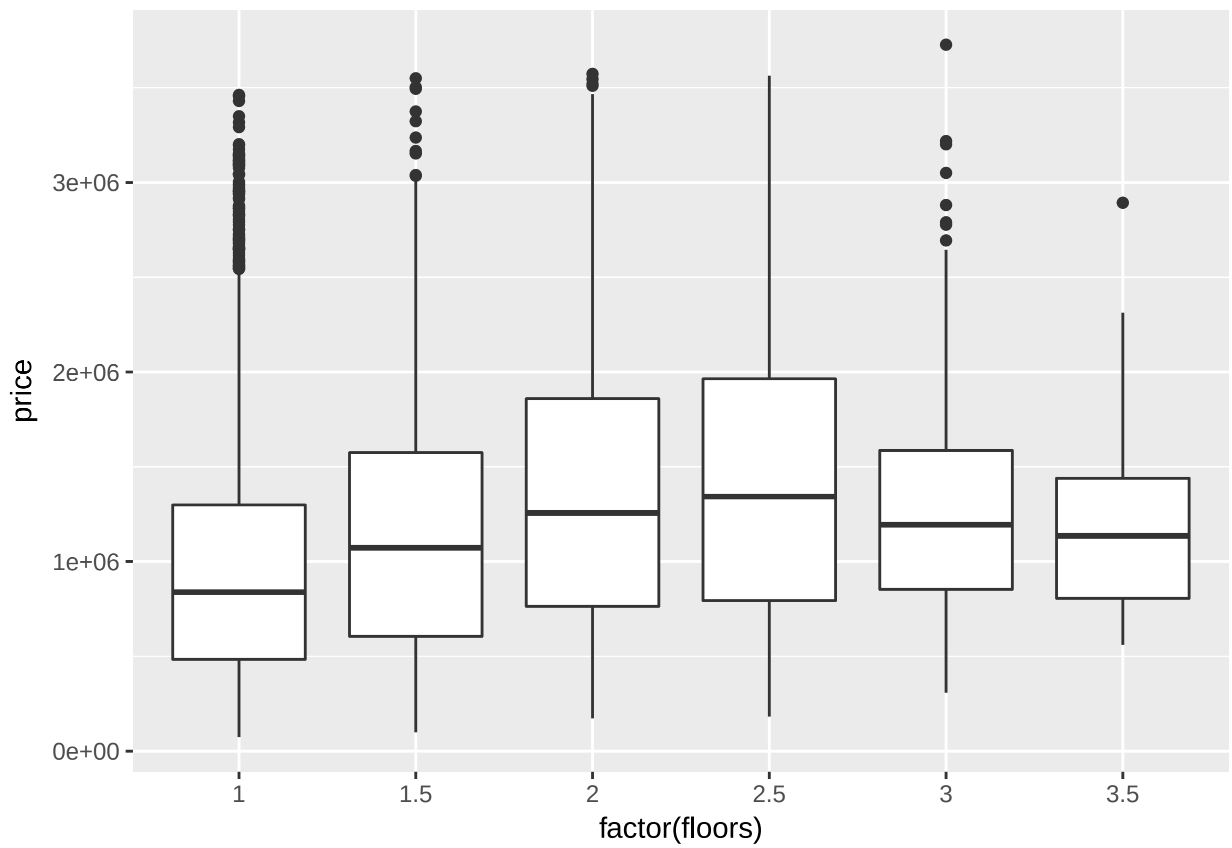

```
data %>%
  ggplot(aes(x=factor(bedrooms), y=price)) +
  geom_boxplot()
```



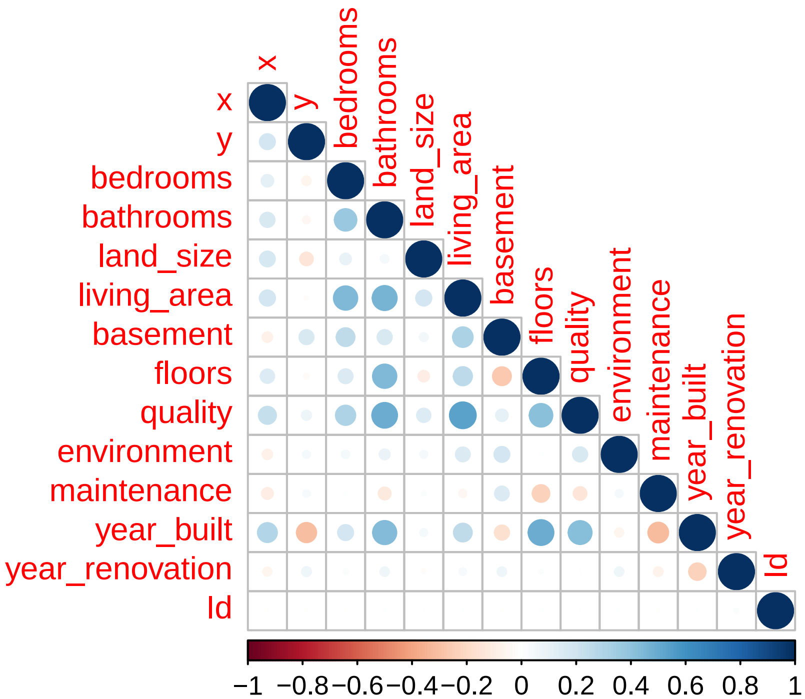

```r
data %>%
  ggplot(aes(x=factor(bathrooms), y=living_area)) +
  geom_boxplot()
```



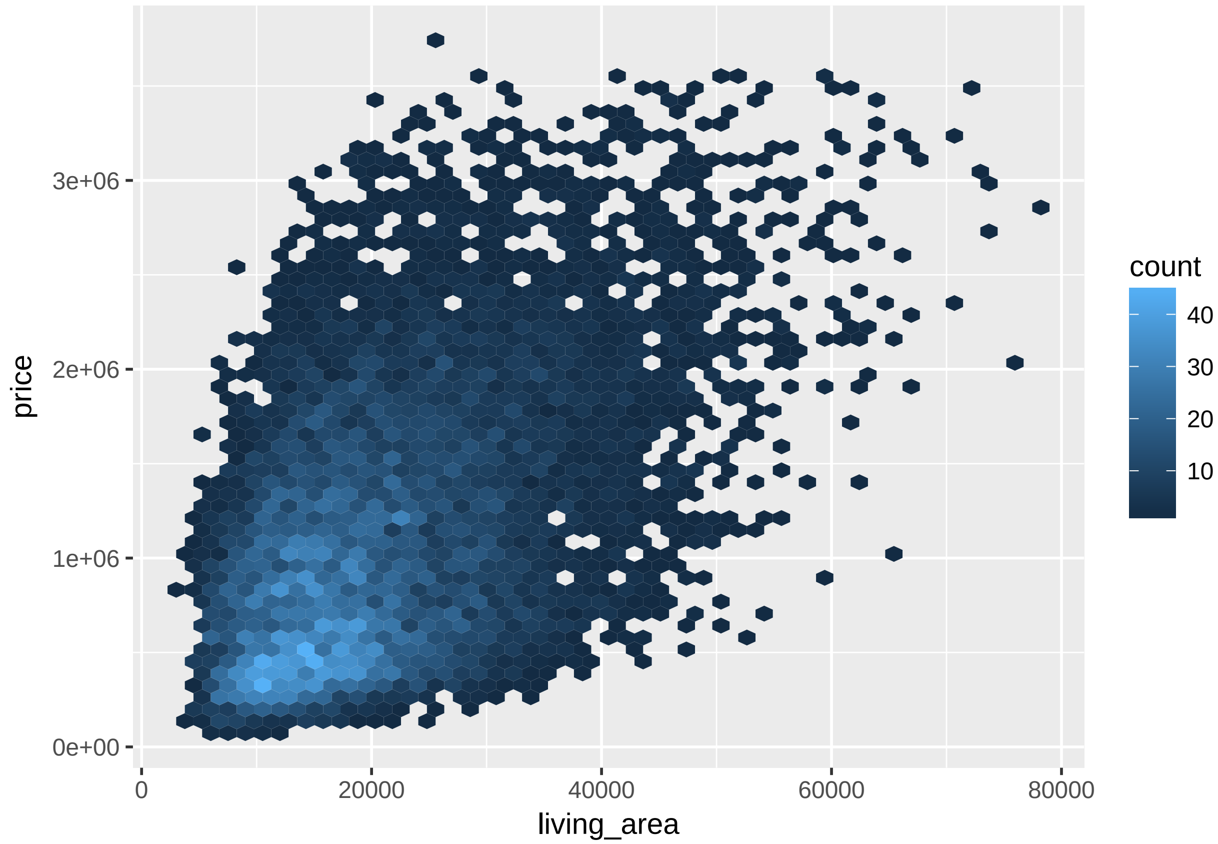

```r
data %>%
  ggplot(aes(x=factor(floors), y=price)) +
  geom_boxplot()
```




```r
# data_skewneww_score <- sapply(data, skewness)
# data_skewneww_score

correlations<- cor(data[,-1],use="everything")
corrplot(correlations, method="circle", type="lower",  sig.level = 0.01, insig = "blank")
```




```r
# ggplot(data,aes(x= year_built,y=price))+geom_point()+geom_smooth()
# ggplot(data,aes(x= year_renovation,y=price))+geom_point()+geom_smooth()

data %>%
  ggplot(aes(x=living_area,y=price)) +
  stat_binhex(bins=50)
```



Data Preprocessing

```
x_data <- data
x_data$price <- NULL
y_data <- data$price

n = nrow(data)

filename_output = "test.csv"
data_output = read.csv(file = filename_output)
output_id = data_output$Id

ALL_X_DATA = rbind(x_data,data_output)

renovation_year_generator <- function(x) {
  if (x['year_renovation'] == -1){
    return (x['year_built'])
  }
  else{
    return (2017-x['year_renovation'])
  }
}

ETL <- function(data){

  # change year built to age of the house
  data$year_built = 2017-data$year_built
```

```r
  # a dummy variable which indicates if the house has been renovated
  data$is_renovated = as.numeric(data$year_renovation != -1)

  # years since last renovation
  data$year_renovation = apply(data, 1, renovation_year_generator)

  # indicate if the house as a basement
  data$has_basement = as.numeric(data$basement != 0)

  # area per floor
  data$area_per_floor = data$living_area/data$floors

  # indicate the house is old (older than 75 years)
  data$is_old = as.numeric(data$year_built >= 75)

  # taking squre of the scores
  data$quality_2 = data$quality^2
  data$environment_2 = data$environment^2
  data$maintenance_2 = data$maintenance^2
  data <- data[ , !(names(data) %in% c("Id"))]

  #clustering based on location (15 areas)
  location_data <- data[,c('x','y')]
  set.seed(1)  #for reproducibility
  location_fit <- kmeans(location_data, 15)
  location.f = factor(location_fit$cluster)
  location_dummies = model.matrix(~location.f)[,-1]
  data <- cbind(data,location_dummies)

  #clustering based on room type
  type_data <- data[,c('bedrooms','bathrooms','living_area','floors','has_basement','area_per_floor')]
  set.seed(1)  #for reproducibility
  type_fit <- kmeans(type_data, 3)
  type.f = factor(type_fit$cluster)
  type_dummies = model.matrix(~type.f)[,-1]
  data <- cbind(data,type_dummies)

  return (data)
}

ALL_X_DATA <- ETL(ALL_X_DATA)
x_data = ALL_X_DATA[1:n,]
data <- x_data
data$price <- y_data

data_output = ALL_X_DATA[(n+1):nrow(ALL_X_DATA),]
```

```r
# create a sample vector of test values
set.seed(1)  #for reproducibility
test.n <- sample(1:nrow(data), nrow(data)/5, replace = F)

# test dataset
y_test <- y_data[test.n]
```

```r
x_test <- x_data[test.n,]
test <- x_test
test$price <- y_test
# x_test <- test[ , !(names(test) %in% c("price"))]

# test <- data
# y_test <- test$price
# x_test <- test[ , !(names(cleaned_data) %in% c("price"))]

# train dataset
y_train <- y_data[-test.n]
x_train <- x_data[-test.n,]
train <- x_train
train$price <- y_train
```

```r
RMSPE <- function(pred_y,true_y){
    a <- sqrt(mean(((pred_y-true_y)/true_y)^2))
    return(a)
}

result_output <- function(y){
  # y <- expm1(y)
  y <- round_any(y,100)
  return(y)
}

xg_eval_mae <- function (yhat, dtrain) {
  y = getinfo(dtrain, "label")
  y = expm1(y)
  yhat = expm1(yhat)
  err= RMSPE(yhat, y)
  return (list(metric = "error", value = err))
}
```

linear regression (stepwise) Since prices of the house are skewed, we take the log the price to reduce skewness.

```r
linear_model = lm(log1p(price) ~ ., data = train)
linear_model <- step(linear_model)
```

```
## Start:  AIC=-17546.5
## log1p(price) ~ x + y + bedrooms + bathrooms + land_size + living_area +
##     basement + floors + quality + environment + maintenance +
##     year_built + year_renovation + is_renovated + has_basement +
##     area_per_floor + is_old + quality_2 + environment_2 + maintenance_2 +
##     location.f2 + location.f3 + location.f4 + location.f5 + location.f6 +
##     location.f7 + location.f8 + location.f9 + location.f10 +
##     location.f11 + location.f12 + location.f13 + location.f14 +
##     location.f15 + type.f2 + type.f3
##
##                   Df Sum of Sq    RSS    AIC
## - location.f11     1     0.007 884.16 -17548
## - floors           1     0.012 884.17 -17548
## - location.f6      1     0.044 884.20 -17548
## - environment      1     0.074 884.23 -17548
## - maintenance      1     0.074 884.23 -17548
```

```
## <none>                          884.16 -17547
## - area_per_floor   1     0.249 884.41 -17546
## - basement         1     0.388 884.54 -17545
## - location.f14     1     0.424 884.58 -17545
## - year_renovation  1     0.592 884.75 -17543
## - year_built       1     0.785 884.94 -17541
## - has_basement     1     1.029 885.19 -17539
## - type.f3          1     1.207 885.36 -17538
## - maintenance_2    1     1.552 885.71 -17535
## - is_renovated     1     1.667 885.82 -17533
## - location.f9      1     2.625 886.78 -17525
## - x                1     2.645 886.80 -17525
## - location.f13     1     2.996 887.15 -17521
## - type.f2          1     3.123 887.28 -17520
## - environment_2    1     3.328 887.48 -17518
## - location.f12     1     3.464 887.62 -17517
## - quality_2        1     4.489 888.65 -17508
## - location.f8      1     5.471 889.63 -17499
## - is_old           1     5.569 889.73 -17498
## - location.f7      1     6.168 890.32 -17493
## - bathrooms        1     8.375 892.53 -17473
## - location.f10     1     9.627 893.78 -17462
## - land_size        1    10.114 894.27 -17458
## - location.f4      1    10.298 894.45 -17456
## - living_area      1    13.170 897.33 -17430
## - location.f2      1    15.846 900.00 -17406
## - bedrooms         1    18.954 903.11 -17379
## - y                1    21.425 905.58 -17357
## - location.f3      1    27.189 911.35 -17306
## - location.f15     1    28.420 912.58 -17295
## - location.f5      1    33.840 918.00 -17248
## - quality          1    34.882 919.04 -17239
##
## Step:  AIC=-17548.44
## log1p(price) ~ x + y + bedrooms + bathrooms + land_size + living_area +
##     basement + floors + quality + environment + maintenance +
##     year_built + year_renovation + is_renovated + has_basement +
##     area_per_floor + is_old + quality_2 + environment_2 + maintenance_2 +
##     location.f2 + location.f3 + location.f4 + location.f5 + location.f6 +
##     location.f7 + location.f8 + location.f9 + location.f10 +
##     location.f12 + location.f13 + location.f14 + location.f15 +
##     type.f2 + type.f3
##
##                   Df Sum of Sq    RSS    AIC
## - floors           1     0.012 884.18 -17550
## - environment      1     0.074 884.24 -17550
## - maintenance      1     0.075 884.24 -17550
## - location.f6      1     0.076 884.24 -17550
## <none>                          884.16 -17548
## - area_per_floor   1     0.250 884.41 -17548
## - basement         1     0.387 884.55 -17547
## - year_renovation  1     0.592 884.76 -17545
## - location.f14     1     0.687 884.85 -17544
## - year_built       1     0.785 884.95 -17543
```

```
## - has_basement     1     1.041 885.20 -17541
## - type.f3          1     1.213 885.38 -17540
## - maintenance_2    1     1.552 885.72 -17536
## - is_renovated     1     1.667 885.83 -17535
## - type.f2          1     3.134 887.30 -17522
## - environment_2    1     3.330 887.49 -17520
## - x                1     3.651 887.81 -17518
## - quality_2        1     4.489 888.65 -17510
## - location.f9      1     5.035 889.20 -17505
## - location.f12     1     5.454 889.62 -17501
## - is_old           1     5.678 889.84 -17499
## - location.f13     1     5.743 889.91 -17499
## - location.f8      1     7.800 891.96 -17480
## - bathrooms        1     8.378 892.54 -17475
## - land_size        1    10.108 894.27 -17460
## - location.f4      1    10.593 894.76 -17455
## - location.f7      1    10.666 894.83 -17455
## - location.f10     1    12.391 896.56 -17439
## - living_area      1    13.182 897.35 -17432
## - bedrooms         1    18.948 903.11 -17381
## - y                1    24.987 909.15 -17328
## - location.f2      1    27.483 911.65 -17306
## - location.f3      1    28.861 913.03 -17294
## - quality          1    34.876 919.04 -17241
## - location.f15     1    35.868 920.03 -17232
## - location.f5      1    53.077 937.24 -17084
##
## Step:  AIC=-17550.33
## log1p(price) ~ x + y + bedrooms + bathrooms + land_size + living_area +
##     basement + quality + environment + maintenance + year_built +
##     year_renovation + is_renovated + has_basement + area_per_floor +
##     is_old + quality_2 + environment_2 + maintenance_2 + location.f2 +
##     location.f3 + location.f4 + location.f5 + location.f6 + location.f7 +
##     location.f8 + location.f9 + location.f10 + location.f12 +
##     location.f13 + location.f14 + location.f15 + type.f2 + type.f3
##
##                   Df Sum of Sq    RSS    AIC
## - location.f6      1     0.073 884.25 -17552
## - maintenance      1     0.074 884.25 -17552
## - environment      1     0.075 884.25 -17552
## <none>                         884.18 -17550
## - basement         1     0.378 884.55 -17549
## - area_per_floor   1     0.412 884.59 -17549
## - year_renovation  1     0.592 884.77 -17547
## - location.f14     1     0.697 884.87 -17546
## - year_built       1     0.774 884.95 -17545
## - has_basement     1     1.081 885.26 -17543
## - type.f3          1     1.206 885.38 -17541
## - maintenance_2    1     1.546 885.72 -17538
## - is_renovated     1     1.661 885.84 -17537
## - type.f2          1     3.124 887.30 -17524
## - environment_2    1     3.319 887.49 -17522
## - x                1     3.717 887.89 -17519
## - quality_2        1     4.533 888.71 -17511
```

```
## - location.f9      1     5.065 889.24 -17507
## - location.f12     1     5.472 889.65 -17503
## - is_old           1     5.687 889.86 -17501
## - location.f13     1     5.739 889.91 -17501
## - location.f8      1     7.836 892.01 -17482
## - bathrooms        1     8.372 892.55 -17477
## - land_size        1    10.103 894.28 -17461
## - location.f4      1    10.590 894.77 -17457
## - location.f7      1    10.657 894.83 -17457
## - location.f10     1    12.391 896.57 -17441
## - bedrooms         1    18.939 903.11 -17383
## - living_area      1    22.551 906.73 -17351
## - y                1    25.045 909.22 -17329
## - location.f2      1    27.474 911.65 -17308
## - location.f3      1    28.898 913.07 -17295
## - quality          1    35.551 919.73 -17237
## - location.f15     1    35.857 920.03 -17234
## - location.f5      1    53.113 937.29 -17086
##
## Step:  AIC=-17551.67
## log1p(price) ~ x + y + bedrooms + bathrooms + land_size + living_area +
##     basement + quality + environment + maintenance + year_built +
##     year_renovation + is_renovated + has_basement + area_per_floor +
##     is_old + quality_2 + environment_2 + maintenance_2 + location.f2 +
##     location.f3 + location.f4 + location.f5 + location.f7 + location.f8 +
##     location.f9 + location.f10 + location.f12 + location.f13 +
##     location.f14 + location.f15 + type.f2 + type.f3
##
##                  Df Sum of Sq    RSS    AIC
## - maintenance      1     0.072 884.32 -17553
## - environment      1     0.075 884.32 -17553
## <none>                         884.25 -17552
## - basement         1     0.370 884.62 -17550
## - area_per_floor   1     0.415 884.66 -17550
## - year_renovation  1     0.591 884.84 -17548
## - location.f14     1     0.688 884.94 -17548
## - year_built       1     0.772 885.02 -17547
## - has_basement     1     1.085 885.33 -17544
## - type.f3          1     1.220 885.47 -17543
## - maintenance_2    1     1.537 885.79 -17540
## - is_renovated     1     1.659 885.91 -17539
## - type.f2          1     3.138 887.39 -17525
## - environment_2    1     3.315 887.56 -17524
## - x                1     3.893 888.14 -17519
## - quality_2        1     4.589 888.84 -17512
## - location.f9      1     5.326 889.57 -17506
## - location.f12     1     5.406 889.65 -17505
## - is_old           1     5.877 890.13 -17501
## - location.f13     1     6.404 890.65 -17496
## - bathrooms        1     8.396 892.64 -17478
## - land_size        1    10.081 894.33 -17463
## - location.f4      1    10.829 895.08 -17456
## - location.f8      1    11.131 895.38 -17454
## - location.f10     1    12.448 896.70 -17442
```

```
## - location.f7      1    12.596 896.84 -17441
## - bedrooms         1    18.911 903.16 -17384
## - living_area      1    22.596 906.84 -17352
## - location.f2      1    27.415 911.66 -17309
## - location.f3      1    28.987 913.23 -17296
## - y                1    29.354 913.60 -17292
## - quality          1    35.679 919.93 -17237
## - location.f15     1    36.300 920.55 -17232
## - location.f5      1    68.299 952.55 -16959
##
## Step:  AIC=-17553.02
## log1p(price) ~ x + y + bedrooms + bathrooms + land_size + living_area +
##     basement + quality + environment + year_built + year_renovation +
##     is_renovated + has_basement + area_per_floor + is_old + quality_2 +
##     environment_2 + maintenance_2 + location.f2 + location.f3 +
##     location.f4 + location.f5 + location.f7 + location.f8 + location.f9 +
##     location.f10 + location.f12 + location.f13 + location.f14 +
##     location.f15 + type.f2 + type.f3
##
##                   Df Sum of Sq    RSS    AIC
## - environment      1     0.075 884.40 -17554
## <none>                         884.32 -17553
## - basement         1     0.371 884.69 -17552
## - area_per_floor   1     0.419 884.74 -17551
## - year_renovation  1     0.589 884.91 -17550
## - location.f14     1     0.680 885.00 -17549
## - year_built       1     0.777 885.10 -17548
## - has_basement     1     1.080 885.40 -17545
## - type.f3          1     1.217 885.54 -17544
## - is_renovated     1     1.664 885.98 -17540
## - type.f2          1     3.139 887.46 -17527
## - environment_2    1     3.319 887.64 -17525
## - x                1     3.895 888.21 -17520
## - quality_2        1     4.579 888.90 -17514
## - location.f9      1     5.304 889.62 -17507
## - location.f12     1     5.375 889.69 -17507
## - is_old           1     5.945 890.27 -17501
## - location.f13     1     6.386 890.71 -17498
## - bathrooms        1     8.413 892.73 -17479
## - maintenance_2    1     8.429 892.75 -17479
## - land_size        1    10.065 894.38 -17465
## - location.f4      1    10.793 895.11 -17458
## - location.f8      1    11.147 895.47 -17455
## - location.f10     1    12.411 896.73 -17444
## - location.f7      1    12.604 896.92 -17442
## - bedrooms         1    18.935 903.26 -17386
## - living_area      1    22.603 906.92 -17353
## - location.f2      1    27.366 911.69 -17311
## - location.f3      1    28.950 913.27 -17297
## - y                1    29.488 913.81 -17293
## - quality          1    35.645 919.97 -17239
## - location.f15     1    36.235 920.56 -17234
## - location.f5      1    68.312 952.63 -16960
##
```

```
## Step:  AIC=-17554.34
## log1p(price) ~ x + y + bedrooms + bathrooms + land_size + living_area +
##     basement + quality + year_built + year_renovation + is_renovated +
##     has_basement + area_per_floor + is_old + quality_2 + environment_2 +
##     maintenance_2 + location.f2 + location.f3 + location.f4 +
##     location.f5 + location.f7 + location.f8 + location.f9 + location.f10 +
##     location.f12 + location.f13 + location.f14 + location.f15 +
##     type.f2 + type.f3
##
##                   Df Sum of Sq    RSS    AIC
## <none>                         884.40 -17554
## - basement         1     0.367 884.76 -17553
## - area_per_floor   1     0.429 884.82 -17553
## - year_renovation  1     0.583 884.98 -17551
## - location.f14     1     0.678 885.07 -17550
## - year_built       1     0.770 885.17 -17549
## - has_basement     1     1.086 885.48 -17547
## - type.f3          1     1.214 885.61 -17545
## - is_renovated     1     1.654 886.05 -17541
## - type.f2          1     3.145 887.54 -17528
## - x                1     3.890 888.29 -17521
## - quality_2        1     4.612 889.01 -17515
## - location.f9      1     5.293 889.69 -17509
## - location.f12     1     5.367 889.76 -17508
## - is_old           1     5.941 890.34 -17503
## - location.f13     1     6.373 890.77 -17499
## - bathrooms        1     8.405 892.80 -17481
## - maintenance_2    1     8.439 892.83 -17480
## - land_size        1    10.091 894.49 -17466
## - location.f4      1    10.795 895.19 -17459
## - location.f8      1    11.130 895.53 -17456
## - location.f10     1    12.402 896.80 -17445
## - location.f7      1    12.603 897.00 -17443
## - bedrooms         1    18.915 903.31 -17387
## - living_area      1    22.652 907.05 -17354
## - environment_2    1    24.026 908.42 -17342
## - location.f2      1    27.343 911.74 -17313
## - location.f3      1    28.951 913.35 -17299
## - y                1    29.506 913.90 -17294
## - quality          1    35.728 920.12 -17240
## - location.f15     1    36.184 920.58 -17236
## - location.f5      1    68.327 952.72 -16961
```

```r
summary(linear_model)
```

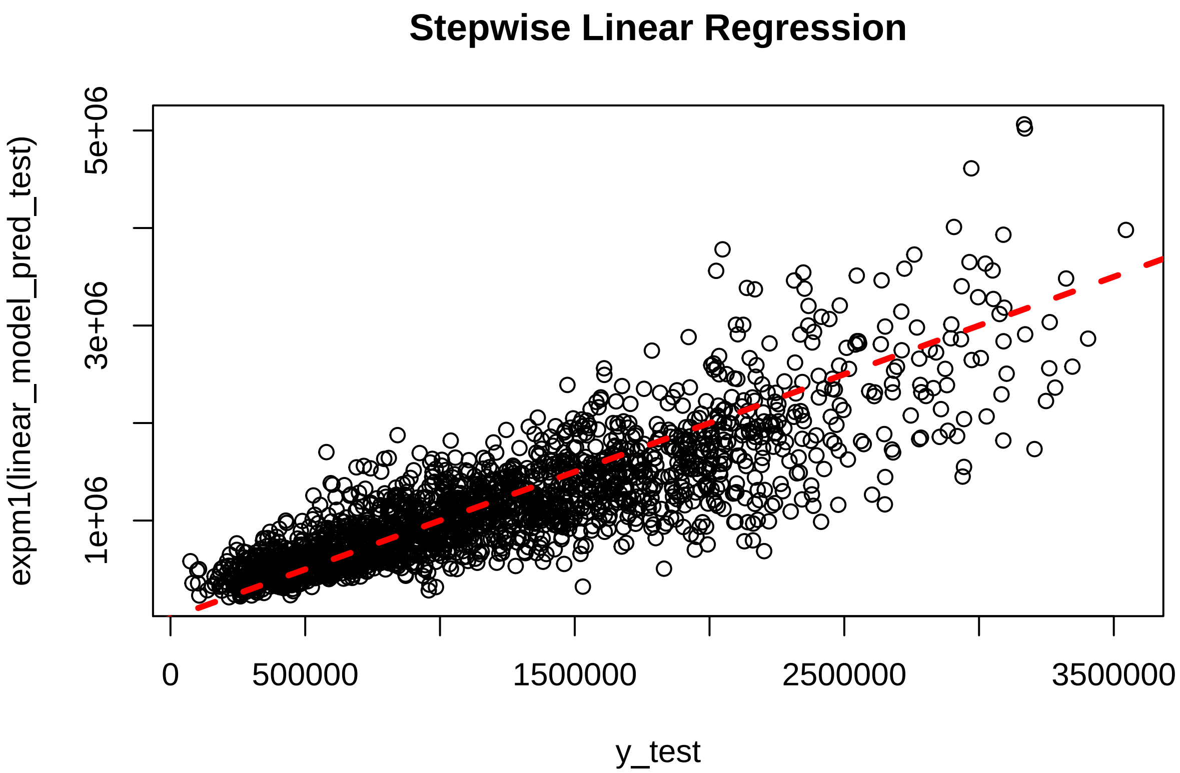
```
##
## Call:
## lm(formula = log1p(price) ~ x + y + bedrooms + bathrooms + land_size +
##     living_area + basement + quality + year_built + year_renovation +
##     is_renovated + has_basement + area_per_floor + is_old + quality_2 +
##     environment_2 + maintenance_2 + location.f2 + location.f3 +
##     location.f4 + location.f5 + location.f7 + location.f8 + location.f9 +
##     location.f10 + location.f12 + location.f13 + location.f14 +
##     location.f15 + type.f2 + type.f3, data = train)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.89594 -0.18539  0.01287  0.20356  1.55089
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1.239e+01  5.320e-02 232.950  < 2e-16 ***
## x                1.502e-01  2.538e-02   5.920 3.36e-09 ***
## y                3.983e-01  2.443e-02  16.305  < 2e-16 ***
## bedrooms         6.536e-02  5.007e-03  13.054  < 2e-16 ***
## bathrooms        6.178e-02  7.099e-03   8.702  < 2e-16 ***
## land_size        3.143e-07  3.296e-08   9.535  < 2e-16 ***
## living_area      1.294e-05  9.059e-07  14.286  < 2e-16 ***
## basement         2.577e-06  1.416e-06   1.819 0.068931 .
## quality          1.948e+00  1.086e-01  17.941  < 2e-16 ***
## year_built      -3.183e-03  1.208e-03  -2.634 0.008445 **
## year_renovation  2.689e-03  1.173e-03   2.293 0.021884 *
## is_renovated     2.742e-01  7.102e-02   3.860 0.000114 ***
## has_basement     4.046e-02  1.294e-02   3.128 0.001769 **
## area_per_floor  -1.933e-06  9.835e-07  -1.965 0.049395 *
## is_old           1.215e-01  1.661e-02   7.316 2.80e-13 ***
## quality_2       -7.030e-01  1.091e-01  -6.446 1.22e-10 ***
## environment_2    5.655e-01  3.844e-02  14.713  < 2e-16 ***
## maintenance_2    1.896e-01  2.174e-02   8.720  < 2e-16 ***
## location.f2     -4.195e-01  2.672e-02 -15.695  < 2e-16 ***
## location.f3     -5.671e-01  3.512e-02 -16.150  < 2e-16 ***
## location.f4     -2.307e-01  2.339e-02  -9.862  < 2e-16 ***
## location.f5     -5.217e-01  2.103e-02 -24.811  < 2e-16 ***
## location.f7     -2.578e-01  2.419e-02 -10.656  < 2e-16 ***
## location.f8     -1.697e-01  1.694e-02 -10.014  < 2e-16 ***
## location.f9     -2.431e-01  3.521e-02  -6.906 5.38e-12 ***
## location.f10    -4.279e-01  4.048e-02 -10.571  < 2e-16 ***
## location.f12    -2.928e-01  4.210e-02  -6.954 3.84e-12 ***
## location.f13    -2.143e-01  2.828e-02  -7.577 3.92e-14 ***
## location.f14    -1.138e-01  4.605e-02  -2.471 0.013478 *
## location.f15    -4.068e-01  2.253e-02 -18.056  < 2e-16 ***
## type.f2          9.489e-02  1.782e-02   5.323 1.05e-07 ***
## type.f3          8.578e-02  2.594e-02   3.307 0.000946 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3332 on 7968 degrees of freedom
## Multiple R-squared:  0.7403, Adjusted R-squared:  0.7393
## F-statistic: 732.8 on 31 and 7968 DF,  p-value: < 2.2e-16
```

```r
linear_model_pred_train = predict.lm(linear_model, newdata = train)
linear_model_pred_test = predict.lm(linear_model, newdata = test)
RMSPE(expm1(linear_model_pred_test),y_test)
```

```
## [1] 0.423323
```

```r
plot(y_test, expm1(linear_model_pred_test),
     main="Stepwise Linear Regression")
abline(a = 0, b=1, col="red", lwd=3, lty=2)
```




```
#everything is roughly fine. There is a nice positive correlation
#(even though some predicted price are negative!)
```

```
set.seed(1)
dtrain=xgb.DMatrix(as.matrix(x_train),label= log1p(y_train))


best_param = list()
best_seednumber = 1234
best_rmspe = Inf
best_rmspe_index = 0

for (iter in 1:100) {
      param <- list(objective = "reg:linear",
          max_depth = sample(6:20, 1),
          eta = runif(1, .01, .3),
          gamma = runif(1, 0, 2),
          subsample = runif(1, .6, .9),
          colsample_bytree = runif(1, .5, .8),
          min_child_weight = sample(1:40, 1)
          )
    cv.nround = 150
    cv.nfold = 5
    seed.number = sample.int(10000, 1)[[1]]
    set.seed(seed.number)
    mdcv <- xgb.cv(data=dtrain, params = param, nthread=6,
```

```r
                     nfold=cv.nfold, nrounds=cv.nround,
                     verbose = F, early_stopping_rounds=8, maximize=FALSE,
                     feval = xg_eval_mae)

    min_rmspe = min(mdcv$evaluation_log[, test_error_mean])
    min_rmspe_index = which.min(mdcv$evaluation_log[, test_error_mean])

    if (min_rmspe < best_rmspe) {
        best_rmspe = min_rmspe
        best_rmspe_index = min_rmspe_index
        best_seednumber = seed.number
        best_param = param
    }
}
```

```
## Warning in sqrt(rowMeans(msg^2) - bst_evaluation^2): NaNs produced

## Warning in sqrt(rowMeans(msg^2) - bst_evaluation^2): NaNs produced
```

```r
nround = best_rmspe_index

set.seed(best_seednumber)
md <- xgb.train(data=dtrain, params=best_param, nrounds=150, nthread=6)
# md <- xgb.train(data=dtrain, nrounds=10, nthread=6)


importance <- xgb.importance(feature_names = names(x_data), model = md)
importance
```

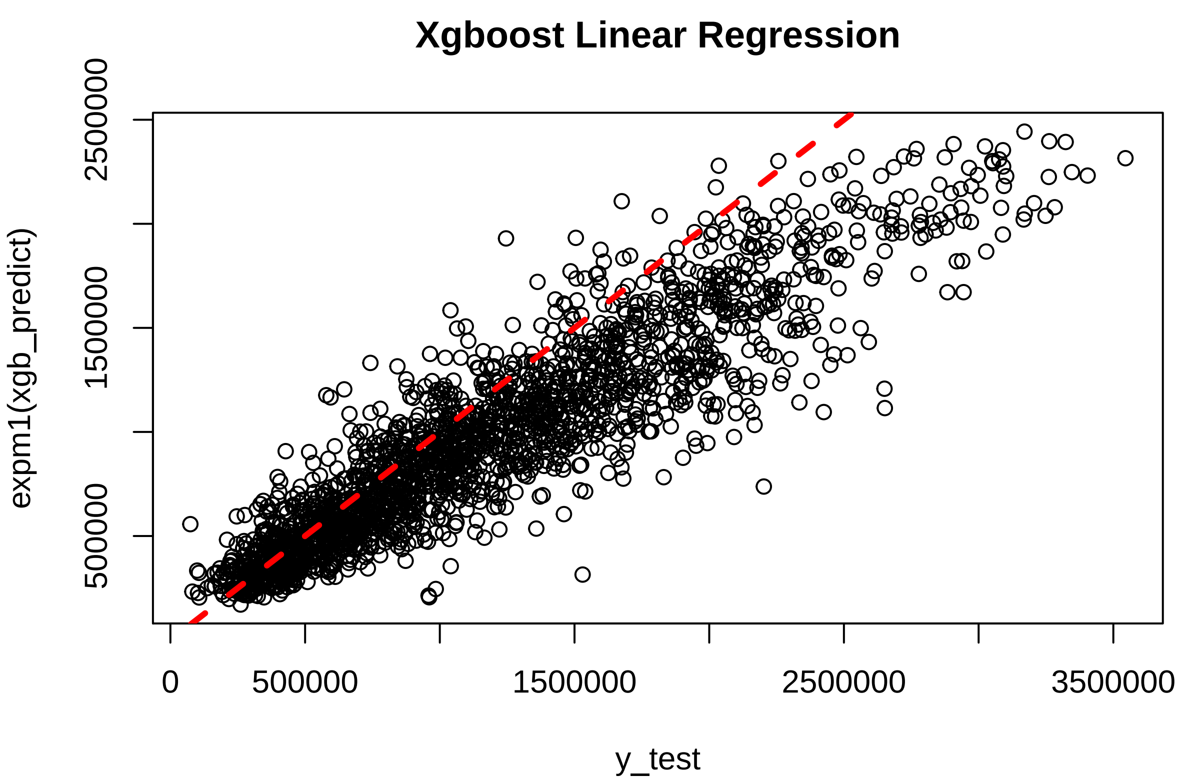
```
##              Feature         Gain        Cover    Frequency
##  1:                y 0.4479872547 0.2132322296 0.1844444444
##  2:          quality 0.1772495363 0.0947182614 0.0662222222
##  3:                x 0.0983593846 0.1350865891 0.1637777778
##  4:      living_area 0.0813651198 0.0810304353 0.0664444444
##  5:        quality_2 0.0433882807 0.0245767035 0.0188888889
##  6:        land_size 0.0220204466 0.0687431029 0.0988888889
##  7:        bathrooms 0.0179614185 0.0324691180 0.0124444444
##  8:      environment 0.0160290980 0.0454187573 0.0380000000
##  9:       year_built 0.0145541843 0.0366422854 0.0560000000
## 10:         bedrooms 0.0141377252 0.0427980698 0.0244444444
## 11: year_renovation 0.0125296411 0.0459485575 0.0588888889
## 12:         basement 0.0097307429 0.0284452188 0.0244444444
## 13:      maintenance 0.0065748014 0.0272741769 0.0442222222
## 14:    environment_2 0.0050339018 0.0145879654 0.0126666667
## 15:     location.f10 0.0047879992 0.0099799043 0.0057777778
## 16:   area_per_floor 0.0040088609 0.0133265472 0.0351111111
## 17:    maintenance_2 0.0030541353 0.0152847013 0.0206666667
## 18:      location.f3 0.0028003117 0.0037591124 0.0026666667
## 19:      location.f5 0.0027662759 0.0079873210 0.0066666667
## 20:           floors 0.0025339207 0.0127374021 0.0126666667
## 21:      location.f6 0.0021770329 0.0098396964 0.0062222222
## 22:     location.f12 0.0020967002 0.0075651117 0.0046666667
## 23:     location.f15 0.0018509612 0.0068391564 0.0077777778
## 24:          type.f3 0.0012223868 0.0006491694 0.0004444444
## 25:     has_basement 0.0009864849 0.0023463874 0.0031111111
```

```
## 26:    location.f14 0.0009467910 0.0011941456 0.0015555556
## 27:     location.f8 0.0007438852 0.0034071203 0.0042222222
## 28:     location.f4 0.0006946140 0.0011930131 0.0035555556
## 29:     location.f2 0.0005220565 0.0019119467 0.0028888889
## 30:     location.f7 0.0004291631 0.0017619990 0.0024444444
## 31:    is_renovated 0.0003806944 0.0038771226 0.0024444444
## 32:    location.f11 0.0003188989 0.0009157683 0.0022222222
## 33:     location.f9 0.0002573668 0.0020988151 0.0015555556
## 34:    location.f13 0.0002084679 0.0009603902 0.0015555556
## 35:         type.f2 0.0001748538 0.0011452201 0.0015555556
## 36:          is_old 0.0001166028 0.0002484783 0.0004444444
##              Feature         Gain        Cover    Frequency
```

```r
important_features <- head(importance,25)$Feature

dtest=xgb.DMatrix(as.matrix(x_test))
xgb_predict = predict(md,dtest)
#RMSPE(round_any(xgb_predict,100),y_test)
RMSPE(expm1(xgb_predict),y_test)
```

```
## [1] 0.3223504
```

```r
plot(y_test, expm1(xgb_predict),
     main="Xgboost Linear Regression")
abline(a = 0, b=1, col="red", lwd=3, lty=2)
```



Train with the whole dataset

```r
ddata=xgb.DMatrix(as.matrix(x_data),label= log1p(y_data))

best_param <- list(
  objective = "reg:linear",
  max_depth = 20,
  eta = 0.03860384,
  gamma = 0.170282,
  subsample =  0.6801184,
  colsample_bytree = 0.7937623,
  min_child_weight = 25
)

best_seednumber <- 6971
best_rmspe_index <- 44

param <- best_param
cv.nround = 150
cv.nfold = 10
set.seed(best_seednumber)
all_mdcv <- xgb.cv(data=ddata, params = param, nthread=6,
               nfold=cv.nfold, nrounds=cv.nround,
               verbose = T, early_stopping_rounds=8, maximize=FALSE,
               feval = xg_eval_mae)
```

```
## [1]  train-error:0.999998+0.000000   test-error:0.999998+0.000000
## Multiple eval metrics are present. Will use test_error for early stopping.
## Will train until test_error hasn't improved in 8 rounds.
##
## [2]  train-error:0.999995+0.000000   test-error:0.999995+0.000000
## [3]  train-error:0.999992+0.000000   test-error:0.999992+0.000000
## [4]  train-error:0.999987+0.000000   test-error:0.999987+0.000000
## [5]  train-error:0.999979+0.000000   test-error:0.999979+0.000000
## [6]  train-error:0.999967+0.000000   test-error:0.999967+0.000001
## [7]  train-error:0.999951+0.000000   test-error:0.999951+0.000001
## [8]  train-error:0.999927+0.000000   test-error:0.999927+0.000002
## [9]  train-error:0.999894+0.000000   test-error:0.999894+0.000002
## [10] train-error:0.999848+0.000000   test-error:0.999848+0.000003
## [11] train-error:0.999786+0.000001   test-error:0.999786+0.000004
## [12] train-error:0.999702+0.000001   test-error:0.999702+0.000006
## [13] train-error:0.999591+0.000001   test-error:0.999591+0.000008
## [14] train-error:0.999446+0.000001   test-error:0.999446+0.000011
## [15] train-error:0.999259+0.000002   test-error:0.999258+0.000014
## [16] train-error:0.999019+0.000002   test-error:0.999018+0.000019
## [17] train-error:0.998716+0.000002   test-error:0.998715+0.000023
## [18] train-error:0.998338+0.000003   test-error:0.998336+0.000030
## [19] train-error:0.997868+0.000004   test-error:0.997866+0.000038
## [20] train-error:0.997293+0.000005   test-error:0.997291+0.000047
## [21] train-error:0.996596+0.000005   test-error:0.996592+0.000058
## [22] train-error:0.995755+0.000006   test-error:0.995751+0.000070
## [23] train-error:0.994752+0.000009   test-error:0.994748+0.000084
## [24] train-error:0.993566+0.000010   test-error:0.993560+0.000101
## [25] train-error:0.992173+0.000011   test-error:0.992165+0.000120
## [26] train-error:0.990551+0.000013   test-error:0.990541+0.000143
## [27] train-error:0.988677+0.000016   test-error:0.988664+0.000168
```

```
## [28] train-error:0.986523+0.000017   test-error:0.986510+0.000197
## [29] train-error:0.984071+0.000022   test-error:0.984054+0.000226
## [30] train-error:0.981294+0.000025   test-error:0.981274+0.000260
## [31] train-error:0.978167+0.000031   test-error:0.978143+0.000296
## [32] train-error:0.974672+0.000037   test-error:0.974642+0.000334
## [33] train-error:0.970787+0.000041   test-error:0.970755+0.000376
## [34] train-error:0.966493+0.000045   test-error:0.966451+0.000419
## [35] train-error:0.961771+0.000049   test-error:0.961719+0.000462
## [36] train-error:0.956608+0.000057   test-error:0.956546+0.000508
## [37] train-error:0.950994+0.000062   test-error:0.950922+0.000554
## [38] train-error:0.944917+0.000063   test-error:0.944839+0.000609
## [39] train-error:0.938371+0.000070   test-error:0.938281+0.000660
## [40] train-error:0.931347+0.000085   test-error:0.931243+0.000708
## [41] train-error:0.923859+0.000092   test-error:0.923743+0.000761
## [42] train-error:0.915892+0.000095   test-error:0.915764+0.000820
## [43] train-error:0.907457+0.000098   test-error:0.907319+0.000880
## [44] train-error:0.898565+0.000103   test-error:0.898418+0.000937
## [45] train-error:0.889223+0.000099   test-error:0.889059+0.000996
## [46] train-error:0.879433+0.000100   test-error:0.879254+0.001057
## [47] train-error:0.869244+0.000094   test-error:0.869059+0.001124
## [48] train-error:0.858638+0.000120   test-error:0.858431+0.001168
## [49] train-error:0.847634+0.000121   test-error:0.847410+0.001229
## [50] train-error:0.836289+0.000131   test-error:0.836060+0.001271
## [51] train-error:0.824583+0.000144   test-error:0.824357+0.001331
## [52] train-error:0.812564+0.000145   test-error:0.812336+0.001367
## [53] train-error:0.800263+0.000149   test-error:0.800021+0.001433
## [54] train-error:0.787695+0.000153   test-error:0.787449+0.001468
## [55] train-error:0.774879+0.000153   test-error:0.774649+0.001544
## [56] train-error:0.761886+0.000153   test-error:0.761652+0.001597
## [57] train-error:0.748700+0.000156   test-error:0.748486+0.001619
## [58] train-error:0.735355+0.000145   test-error:0.735164+0.001696
## [59] train-error:0.721911+0.000148   test-error:0.721753+0.001779
## [60] train-error:0.708362+0.000159   test-error:0.708231+0.001850
## [61] train-error:0.694768+0.000178   test-error:0.694715+0.001903
## [62] train-error:0.681124+0.000181   test-error:0.681149+0.002004
## [63] train-error:0.667467+0.000188   test-error:0.667581+0.002146
## [64] train-error:0.653823+0.000201   test-error:0.654031+0.002319
## [65] train-error:0.640249+0.000215   test-error:0.640570+0.002499
## [66] train-error:0.626735+0.000251   test-error:0.627175+0.002724
## [67] train-error:0.613310+0.000284   test-error:0.613925+0.002971
## [68] train-error:0.600015+0.000313   test-error:0.600773+0.003228
## [69] train-error:0.586845+0.000338   test-error:0.587796+0.003547
## [70] train-error:0.573844+0.000379   test-error:0.575010+0.003850
## [71] train-error:0.561020+0.000401   test-error:0.562441+0.004159
## [72] train-error:0.548390+0.000428   test-error:0.550122+0.004547
## [73] train-error:0.535981+0.000460   test-error:0.538061+0.004983
## [74] train-error:0.523799+0.000496   test-error:0.526236+0.005354
## [75] train-error:0.511839+0.000551   test-error:0.514662+0.005769
## [76] train-error:0.500145+0.000616   test-error:0.503417+0.006215
## [77] train-error:0.488741+0.000683   test-error:0.492472+0.006725
## [78] train-error:0.477598+0.000709   test-error:0.481756+0.007254
## [79] train-error:0.466735+0.000754   test-error:0.471426+0.007763
## [80] train-error:0.456182+0.000798   test-error:0.461479+0.008391
## [81] train-error:0.445890+0.000851   test-error:0.451806+0.008926
```

```
## [82] train-error:0.435934+0.000912   test-error:0.442456+0.009496
## [83] train-error:0.426271+0.000977   test-error:0.433555+0.010153
## [84] train-error:0.416926+0.000999   test-error:0.424972+0.010822
## [85] train-error:0.407838+0.001069   test-error:0.416677+0.011392
## [86] train-error:0.399112+0.001140   test-error:0.408820+0.012120
## [87] train-error:0.390680+0.001165   test-error:0.401221+0.012679
## [88] train-error:0.382551+0.001220   test-error:0.394065+0.013419
## [89] train-error:0.374719+0.001274   test-error:0.387231+0.014208
## [90] train-error:0.367220+0.001314   test-error:0.380855+0.015028
## [91] train-error:0.360067+0.001334   test-error:0.374733+0.015669
## [92] train-error:0.353133+0.001381   test-error:0.368864+0.016223
## [93] train-error:0.346586+0.001426   test-error:0.363373+0.016880
## [94] train-error:0.340314+0.001486   test-error:0.358276+0.017642
## [95] train-error:0.334280+0.001482   test-error:0.353376+0.018296
## [96] train-error:0.328500+0.001550   test-error:0.348877+0.018849
## [97] train-error:0.323096+0.001583   test-error:0.344738+0.019632
## [98] train-error:0.317859+0.001648   test-error:0.340796+0.020109
## [99] train-error:0.312942+0.001687   test-error:0.337243+0.020907
## [100]    train-error:0.308262+0.001780   test-error:0.333889+0.021568
## [101]    train-error:0.303735+0.001817   test-error:0.330618+0.022102
## [102]    train-error:0.299505+0.001838   test-error:0.327815+0.022851
## [103]    train-error:0.295493+0.001932   test-error:0.325082+0.023341
## [104]    train-error:0.291698+0.001992   test-error:0.322625+0.023947
## [105]    train-error:0.288124+0.002018   test-error:0.320387+0.024404
## [106]    train-error:0.284804+0.002083   test-error:0.318439+0.024963
## [107]    train-error:0.281635+0.002127   test-error:0.316712+0.025503
## [108]    train-error:0.278647+0.002201   test-error:0.315093+0.025959
## [109]    train-error:0.275821+0.002312   test-error:0.313724+0.026343
## [110]    train-error:0.273148+0.002379   test-error:0.312489+0.026816
## [111]    train-error:0.270616+0.002397   test-error:0.311482+0.027413
## [112]    train-error:0.268206+0.002384   test-error:0.310615+0.028016
## [113]    train-error:0.265961+0.002452   test-error:0.309899+0.028472
## [114]    train-error:0.263851+0.002528   test-error:0.309167+0.028785
## [115]    train-error:0.261845+0.002589   test-error:0.308606+0.029213
## [116]    train-error:0.259964+0.002625   test-error:0.308252+0.029708
## [117]    train-error:0.258101+0.002603   test-error:0.307856+0.030168
## [118]    train-error:0.256486+0.002565   test-error:0.307593+0.030409
## [119]    train-error:0.254953+0.002552   test-error:0.307333+0.030752
## [120]    train-error:0.253520+0.002593   test-error:0.307176+0.031080
## [121]    train-error:0.252068+0.002508   test-error:0.307123+0.031446
## [122]    train-error:0.250691+0.002458   test-error:0.307029+0.031817
## [123]    train-error:0.249483+0.002436   test-error:0.307019+0.031937
## [124]    train-error:0.248272+0.002407   test-error:0.307076+0.032172
## [125]    train-error:0.247101+0.002423   test-error:0.307123+0.032406
## [126]    train-error:0.246024+0.002389   test-error:0.307345+0.032636
## [127]    train-error:0.244967+0.002394   test-error:0.307646+0.032967
## [128]    train-error:0.243939+0.002387   test-error:0.307853+0.033171
## [129]    train-error:0.243072+0.002371   test-error:0.308086+0.033347
## [130]    train-error:0.242170+0.002355   test-error:0.308292+0.033357
## [131]    train-error:0.241313+0.002407   test-error:0.308598+0.033498
## Stopping. Best iteration:
## [123]    train-error:0.249483+0.002436   test-error:0.307019+0.031937
```

```
all_best_rmspe = min(all_mdcv$evaluation_log[, test_error_mean])
all_best_rmspe_index = which.min(all_mdcv$evaluation_log[, test_error_mean])
nround = all_best_rmspe_index
set.seed(best_seednumber)
all_md <- xgb.train(data=ddata, params=best_param, nrounds=150, nthread=6)
importance <- xgb.importance(feature_names = names(x_data), model = all_md)
importance
```

```
##              Feature         Gain        Cover    Frequency
##  1:                y 0.4453769553 0.1827461384 0.1530024168
##  2:          quality 0.1689849655 0.0884615552 0.0817066369
##  3:                x 0.1045758388 0.1397985976 0.1430563302
##  4:      living_area 0.0824370275 0.0820664751 0.0729689533
##  5:        quality_2 0.0328558391 0.0113812619 0.0082729132
##  6:        land_size 0.0253054028 0.0974236222 0.1049451571
##  7:      environment 0.0198923476 0.0493807236 0.0510317903
##  8:        bathrooms 0.0185779152 0.0233716523 0.0109685815
##  9:         bedrooms 0.0148959693 0.0387375233 0.0224948875
## 10:       year_built 0.0145923150 0.0446905106 0.0622792341
## 11:      maintenance 0.0118730750 0.0468660797 0.0676705707
## 12: year_renovation 0.0111992019 0.0433118890 0.0462911322
## 13:         basement 0.0100211645 0.0330178985 0.0356943670
## 14:   area_per_floor 0.0094144268 0.0312483723 0.0652537646
## 15:     location.f10 0.0055795520 0.0065054110 0.0038111173
## 16:      location.f3 0.0040939363 0.0042329817 0.0025097602
## 17:    environment_2 0.0028742012 0.0089088260 0.0123628927
## 18:           floors 0.0024270269 0.0110713094 0.0097601785
## 19:     location.f12 0.0021790845 0.0029486544 0.0022308979
## 20:     location.f14 0.0020033729 0.0023488911 0.0012084030
## 21:      location.f5 0.0018563061 0.0059619677 0.0035322551
## 22:     location.f15 0.0014255436 0.0040610709 0.0037181632
## 23:    maintenance_2 0.0013745047 0.0042936636 0.0073433724
## 24:      location.f6 0.0010538668 0.0068682196 0.0032533928
## 25:     is_renovated 0.0007342050 0.0099035991 0.0016731735
## 26:      location.f2 0.0007299815 0.0029371081 0.0027886224
## 27:      location.f4 0.0007273756 0.0025945695 0.0030674847
## 28:      location.f8 0.0005451127 0.0034655413 0.0026027143
## 29:     has_basement 0.0005098912 0.0011665556 0.0023238520
## 30:      location.f9 0.0003186369 0.0039457368 0.0021379439
## 31:     location.f13 0.0003023557 0.0014131321 0.0018590816
## 32:     location.f11 0.0003007062 0.0014159545 0.0020449898
## 33:          type.f2 0.0002834165 0.0008303033 0.0027886224
## 34:      location.f7 0.0002519195 0.0010658467 0.0009295408
## 35:           is_old 0.0002168308 0.0006632676 0.0007436326
## 36:          type.f3 0.0002097291 0.0008950906 0.0016731735
##              Feature         Gain        Cover    Frequency
```

```
important_features <- importance$Feature

xgb_predict = predict(all_md,dtest)
#RMSPE(round_any(xgb_predict,100),y_test)
all_best_rmspe
```

```
## [1] 0.3070193
```

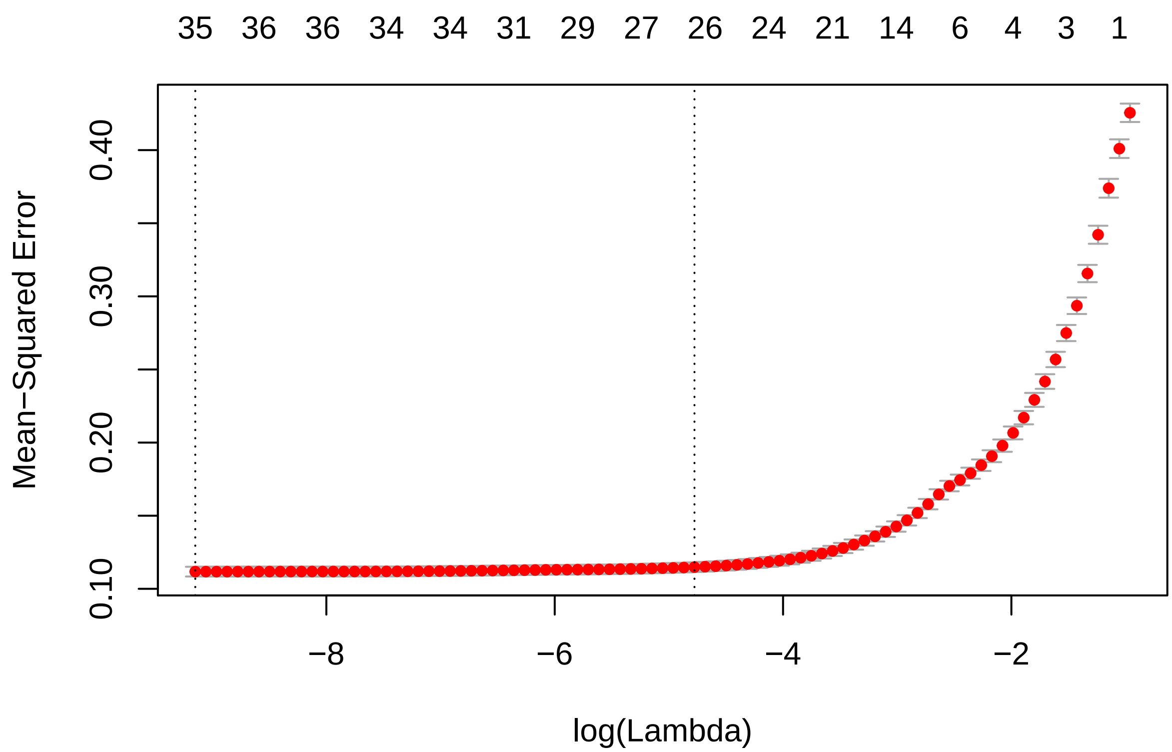
Lasso Regression

```r
library(glmnet)
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

```r
# lasso_fit_trial <-glmnet(x = as.matrix(x_train), y = y_train, alpha = 1)
# plot(lasso_fit_trial, xvar = "lambda")
crossval <-  cv.glmnet(x = as.matrix(x_train), y = log1p(y_train))
plot(crossval)
```




```r
penalty <- crossval$lambda.min #optimal lambda
penalty #minimal shrinkage
```

```
## [1] 0.0001064018
```

```r
lasso_fit <-glmnet(x = as.matrix(x_train), y = log1p(y_train), alpha = 1, lambda = penalty ) #estimate
coef(lasso_fit)
```

```
## 37 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)      1.239180e+01
## x                1.392552e-01
## y                3.980781e-01
## bedrooms         6.540991e-02
## bathrooms        6.192725e-02
## land_size        3.120557e-07
```

```
## living_area      1.278545e-05
## basement         2.506082e-06
## floors           .
## quality          1.915023e+00
## environment      5.133925e-02
## maintenance     -3.493324e-02
## year_built      -1.881513e-03
## year_renovation  1.449444e-03
## is_renovated     2.011766e-01
## has_basement     4.061432e-02
## area_per_floor  -1.907771e-06
## is_old           1.155028e-01
## quality_2       -6.639640e-01
## environment_2    4.987565e-01
## maintenance_2    2.274912e-01
## location.f2     -4.068558e-01
## location.f3     -5.646341e-01
## location.f4     -2.263765e-01
## location.f5     -5.057094e-01
## location.f6      1.898373e-02
## location.f7     -2.402610e-01
## location.f8     -1.564742e-01
## location.f9     -2.205265e-01
## location.f10    -4.194866e-01
## location.f11     6.586722e-03
## location.f12    -2.773149e-01
## location.f13    -1.945964e-01
## location.f14    -8.931695e-02
## location.f15    -3.999123e-01
## type.f2          9.176298e-02
## type.f3          8.076675e-02
```

```r
lasso_predict = predict(lasso_fit, newx = as.matrix(x_test),s = penalty)
RMSPE(expm1(lasso_predict),y_test)
```

```
## [1] 0.4233612
```

After comparision, XGBOOST linear regression has the lowest RMSPE. Final output is made based on the regression model with best parameters.

```r
#let's create a submission file

doutput=xgb.DMatrix(as.matrix(data_output))
output = expm1(predict(md,doutput))
submission = data.frame(Id = output_id,
                            Prediction = output)


#submission_linear_model = data.frame(Id = data_output[,"Id"],
#                                     Prediction = linear_model_pred_test)
#let us create the submission file
write.table(x = submission,
            file = "prediction_8_md.csv",
            sep = ",", row.names = FALSE, col.names = TRUE)
```