

hw_house_price_india

DataSloth

2023-08-15

version: nb

This project will predict the price of Indian house using linear regression by R Programming

Source : data.world
type of source: xlsx

Start with load library

```
#load library
library(tidyverse)
library(caret)
library(mlbench)
library(readxl)
library(ggplot2)
```

Load data file to dataframe("df1 = sheet1, df2 = sheet2")

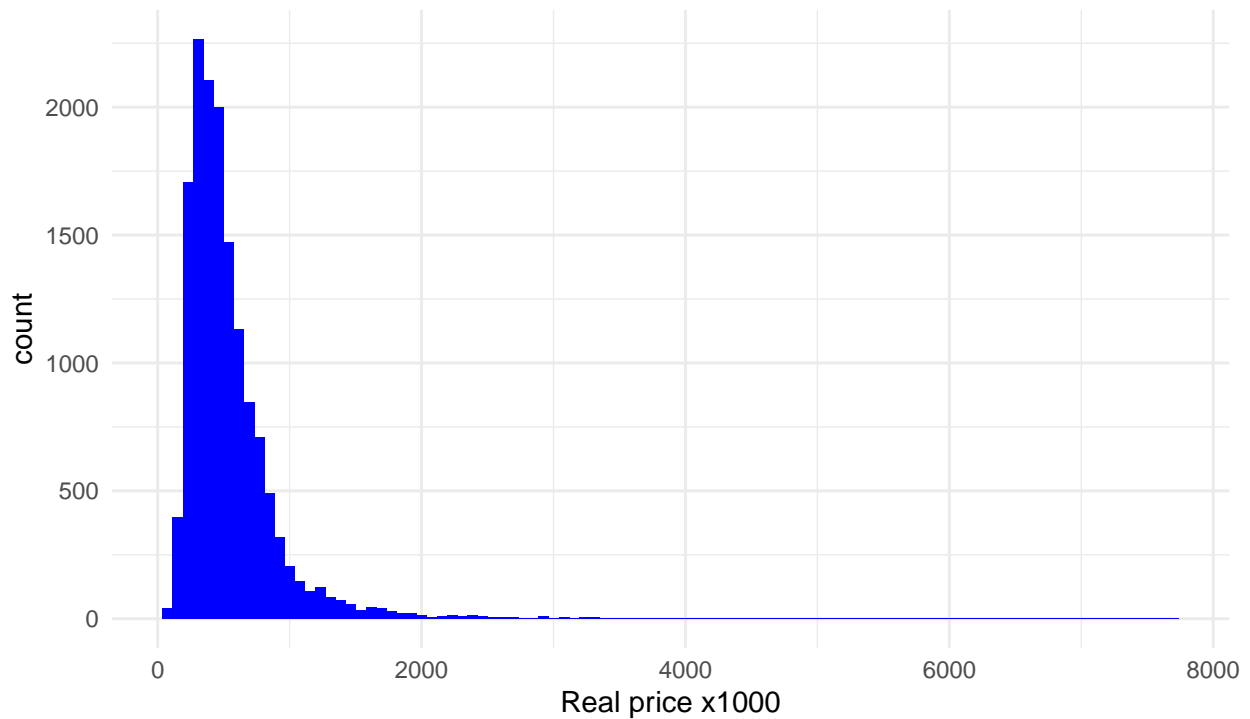
```
# Load data
df1<-read_excel("hpi.xlsx", sheet = 1)
df2<-read_excel("hpi.xlsx", sheet = 2)
```

Visualized data(df1) ~ Price

```
ggplot(df1, aes(Price/1000)) +
  geom_histogram(bins = 100, fill="blue") +
  theme_minimal() +
  labs(
    title = "Visualized Real price by histogram",
    subtitle = "Right skew",
    x = "Real price x1000",
    caption = "Source: data.world"
  )
```

Visualized Real price by histogram

Right skew



Source: data.world

Note: Right skew distribution, not proper for build model but try it.

Create function split_data

```
split_data <- function(df) {  
  set.seed(42)  
  n <- nrow(df)  
  id <- sample(1:n, size = 0.8*n)  
  train_df <- df[id, ]  
  test_df <- df[-id, ]  
  list(train_df, test_df)  
}
```

1. Use full data and real price to build model

1.1 Split full df1 and real price

```
prep_data <- split_data(df1)  
train_data <- prep_data[[1]]  
test_data <- prep_data[[2]]
```

1.2 Train model full df1 and real price

1.2.1 Train

```
model <- train(Price ~ .,  
              data = train_data[ ,-c(1,2)], #remove id and date  
              method = "lm")
```

1.2.2 Show model and summary model

```
model ; summary(model)
```

```
## Linear Regression  
##  
## 11696 samples  
##    20 predictor  
##  
## No pre-processing  
## Resampling: Bootstrapped (25 reps)  
## Summary of sample sizes: 11696, 11696, 11696, 11696, 11696, 11696, ...  
## Resampling results:  
##  
##    RMSE      Rsquared   MAE  
## 192549.8  0.711037 123219.8  
##  
## Tuning parameter 'intercept' was held constant at a value of TRUE  
  
##  
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -1111112  -96339   -8097    75756  3896879  
##  
## Coefficients: (1 not defined because of singularities)  
##  
##              Estimate Std. Error t value  
## (Intercept)      -6.807e+07  1.241e+07  -5.485  
## '\\number of bedrooms\\'  
##      -3.545e+04  2.577e+03 -13.759  
## '\\number of bathrooms\\'  
##      4.235e+04  4.250e+03   9.966  
## '\\living area\\'  
##      1.409e+02  5.716e+00  24.652  
## '\\lot area\\'  
##     -5.479e-03  6.445e-02  -0.085  
## '\\number of floors\\'  
##      6.220e+03  4.677e+03   1.330  
## '\\waterfront present\\'  
##      6.331e+05  2.217e+04  28.558  
## '\\number of views\\'  
##      4.595e+04  2.800e+03  16.408  
## '\\condition of the house\\'  
##      3.138e+04  2.994e+03  10.479  
## '\\grade of the house\\'  
##      1.003e+05  2.814e+03  35.648  
## '\\Area of the house(excluding basement)\\'  
##      3.560e+01  5.671e+00   6.278  
## '\\Area of the basement\\'  
##              NA              NA              NA  
## '\\Built Year\\'  
##     -2.509e+03  9.334e+01 -26.879
```

```
## '\\Renovation Year\\' 2.295e+01 4.574e+00 5.016
## '\\Postal Code\\' 2.610e+02 1.006e+02 2.596
## Latitude 5.509e+05 1.429e+04 38.557
## Longitude -9.900e+04 1.566e+04 -6.323
## living_area_renov 2.529e+01 4.512e+00 5.606
## lot_area_renov -3.178e-01 9.726e-02 -3.268
## '\\Number of schools nearby\\' 3.513e+03 2.178e+03 1.613
## '\\Distance from the airport\\' 1.344e+01 1.993e+02 0.067
## Pr(>|t|)
## (Intercept) 4.22e-08 ***
## '\\number of bedrooms\\' < 2e-16 ***
## '\\number of bathrooms\\' < 2e-16 ***
## '\\living area\\' < 2e-16 ***
## '\\lot area\\' 0.93226
## '\\number of floors\\' 0.18359
## '\\waterfront present\\' < 2e-16 ***
## '\\number of views\\' < 2e-16 ***
## '\\condition of the house\\' < 2e-16 ***
## '\\grade of the house\\' < 2e-16 ***
## '\\Area of the house(excluding basement)\\' 3.56e-10 ***
## '\\Area of the basement\\' NA
## '\\Built Year\\' < 2e-16 ***
## '\\Renovation Year\\' 5.34e-07 ***
## '\\Postal Code\\' 0.00946 **
## Latitude < 2e-16 ***
## Longitude 2.67e-10 ***
## living_area_renov 2.12e-08 ***
## lot_area_renov 0.00109 **
## '\\Number of schools nearby\\' 0.10684
## '\\Distance from the airport\\' 0.94624
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 192300 on 11676 degrees of freedom
## Multiple R-squared:  0.7136, Adjusted R-squared:  0.7131
## F-statistic: 1531 on 19 and 11676 DF, p-value: < 2.2e-16
```

1.3 score~predict model

```
p <- predict(model, newdata = test_data)
```

1.4 Evaluate model

Create function to calculate MAE, MSE, RMSE

```
cal_mae <- function(actual, predict) {
  error <- actual - predict
  mean(abs(error))
}
```

```

}

cal_mse <- function(actual, predict) {
  error <- actual - predict
  mean(error**2)
}

cal_rmse <- function(actual, predict) {
  error <- actual - predict
  sqrt(mean(error**2))
}

r_train <- function(A,P,M = model) {
  mae_log <- cal_mae(A$log_price, P)
  mse_log <- cal_mse(A$log_price, P)
  rmse_log <- cal_rmse(A$log_price, P)
  mae_expo <- cal_mae(exp(A$log_price), exp(P))
  mse_expo <- cal_mse(exp(A$log_price), exp(P))
  rmse_expo <- cal_rmse(exp(A$log_price), exp(P))
  print("--Evaulation of TRAIN--");
  print(paste("MAE_log_train : ",mae_log)) ;
  print(paste("MSE_log_train : ",mse_log)) ;
  print(paste("RMSE_log_train : ",rmse_log)) ;
  print(paste("MAE_expo_train : ",mae_expo)) ;
  print(paste("MSE_expo_train : ",mse_expo)) ;
  print(paste("RMSE_expo_train : ",rmse_expo)) ;
  print(paste("MAE_model : " , M[[4]][[4]])) ;
  print(paste("Rsquared_model : " , M[[4]][[3]])) ;
  print(paste("RMSE_model : " , M[[4]][[2]])) ;
  list(mae_log, mse_log, rmse_log, mae_expo, mse_expo, rmse_expo)
}

r_test <- function(A,P,M = model) {
  mae_log <- cal_mae(A$log_price, P)
  mse_log <- cal_mse(A$log_price, P)
  rmse_log <- cal_rmse(A$log_price, P)
  mae_expo <- cal_mae(exp(A$log_price), exp(P))
  mse_expo <- cal_mse(exp(A$log_price), exp(P))
  rmse_expo <- cal_rmse(exp(A$log_price), exp(P))
  print("--Evaulation of TEST--");
  print(paste("MAE_log_test : ",mae_log)) ;
  print(paste("MSE_log_test : ",mse_log)) ;
  print(paste("RMSE_log_test : ",rmse_log)) ;
  print(paste("MAE_expo_test : ",mae_expo)) ;
  print(paste("MSE_expo_test : ",mse_expo)) ;
  print(paste("RMSE_expo_test : ",rmse_expo)) ;
  print(paste("MAE_model : " , M[[4]][[4]])) ;
  print(paste("Rsquared_model : " , M[[4]][[3]])) ;
  print(paste("RMSE_model : " , M[[4]][[2]])) ;
  list(mae_log, mse_log, rmse_log, mae_expo, mse_expo, rmse_expo)
}

```

Create function to show error of model and save result to list, note: error in real price and logarithm price

1.5 Show error of model full df1 and real price

```
print(paste("MAE_test : " , cal_mae(test_data$Price, p)))
```

```
## [1] "MAE_test : 128122.818842742"
```

```
print(paste("MSE_test : " , cal_mse(test_data$Price, p)))
```

```
## [1] "MSE_test : 53518979494.8183"
```

```
print(paste("RMSE_test : " , cal_rmse(test_data$Price, p)))
```

```
## [1] "RMSE_test : 231341.694242128"
```

```
print(paste("MAE_model : " , model[[4]][[4]]))
```

```
## [1] "MAE_model : 123219.80420048"
```

```
print(paste("Rsquared_model : " , model[[4]][[3]]))
```

```
## [1] "Rsquared_model : 0.711036950308182"
```

```
print(paste("RMSE_model : " , model[[4]][[2]]))
```

```
## [1] "RMSE_model : 192549.824849158"
```

The model(full data, real price) : high error compare train and test, Rsquared_model : 0.711036950308182

So correct the Right skew distribution by take logarithm price(log_price)

2. Use full data and logarithm price to build model

2.1 Create log_price and Split full df1 with log_price to df1_log

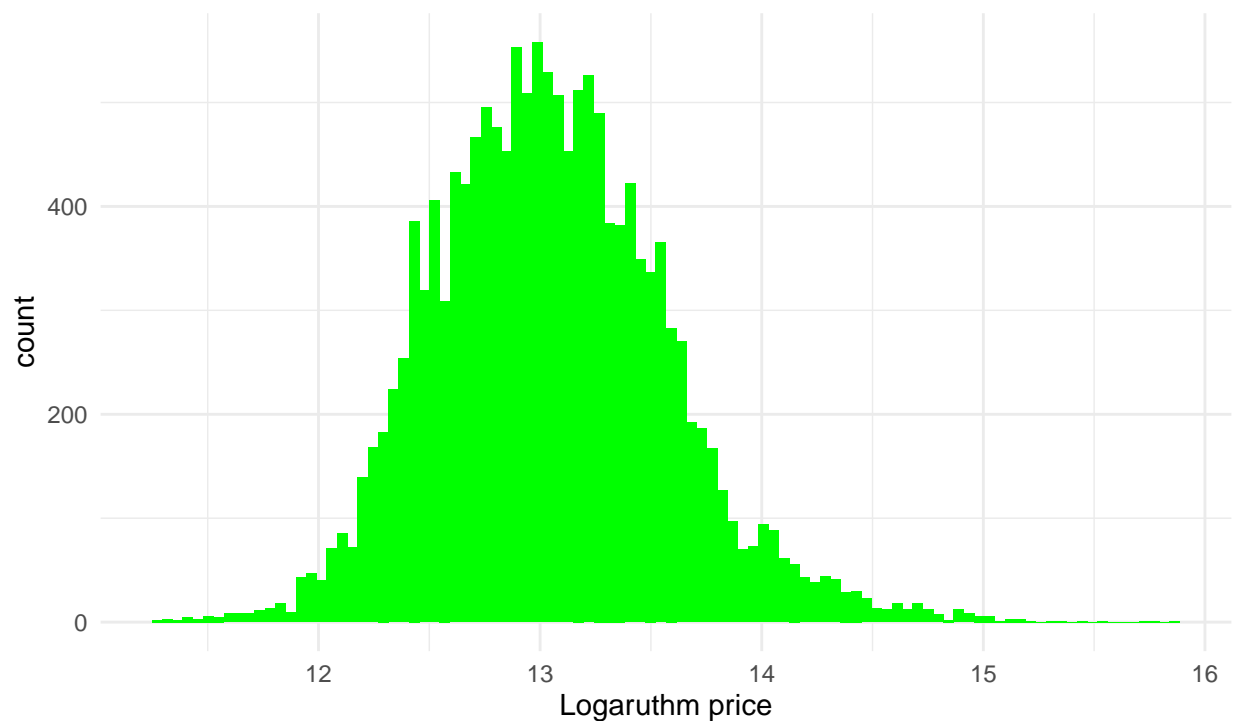
```
df1_log <- df1 %>%  
  mutate(log_price = log(Price))  
  
# Split full df1_log  
  
prep_data_log <- split_data(df1_log)  
train_data_log <- prep_data_log[[1]]  
test_data_log <- prep_data_log[[2]]
```

Visualized data(df1_log) ~ log_price

```
ggplot(df1_log, aes(log_price)) +  
  geom_histogram(bins = 100, fill="green") +  
  theme_minimal() +  
  labs(  
    title = "Visualized logarithm price by histogram",  
    subtitle = "Turn to normal distribution",  
    x = "Logaruthm price",  
    caption = "Source: data.world"  
  )
```

Visualized logarithm price by histogram

Turn to normal distribution



Source: data.world

2.2 Train model df1_log and log_price

2.2.1 Train

```
set.seed(42)  
model_log <- train(log_price ~ .,  
  data = train_data_log[, -c(1,2,23)], #remove price,id,date  
  method = "lm")
```

2.2.2 calculate train error

2.2.2.1 predict train model_log

```
p_log_train <- predict(model_log, newdata = train_data_log)
```

2.2.2.2 calculate train error from model_log with logarithm price and real price_exp(log)

```
model_log_r_train <- r_train(train_data_log, p_log_train, model_log)
```

```
## [1] "--Evaluation of TRAIN--"
## [1] "MAE_log_train : 0.186876611630186"
## [1] "MSE_log_train : 0.0592234337771878"
## [1] "RMSE_log_train : 0.243358652562813"
## [1] "MAE_expo_train : 106108.300922603"
## [1] "MSE_expo_train : 32179140733.112"
## [1] "RMSE_expo_train : 179385.45295846"
## [1] "MAE_model : 0.188182348161632"
## [1] "Rsquared_model : 0.783296783181617"
## [1] "RMSE_model : 0.245141773832881"
```

2.3 score~predict model(model_log)

```
p_log_test <- predict(model_log, newdata = test_data_log)
```

2.4 Evaluate model(model_log)

```
model_log_r_test <- r_test(test_data_log, p_log_test, model_log)
```

```
## [1] "--Evaluation of TEST--"
## [1] "MAE_log_test : 0.193446178197938"
## [1] "MSE_log_test : 0.0648345743548279"
## [1] "RMSE_log_test : 0.254626342617624"
## [1] "MAE_expo_test : 119667.914904593"
## [1] "MSE_expo_test : 125501870360.535"
## [1] "RMSE_expo_test : 354262.431483406"
## [1] "MAE_model : 0.188182348161632"
## [1] "Rsquared_model : 0.783296783181617"
## [1] "RMSE_model : 0.245141773832881"
```

After train model use real price vs logarithm price

find out the important variable for tuning the new model(model_log_s) for lower error or high accuracy

```
## [1] "variable importance of --> model"
```



```
## lm variable importance
##
## Overall
## Latitude 100.00000
## '\\grade of the house\\' 92.44341
## '\\waterfront present\\' 74.02204
## '\\Built Year\\' 69.66113
## '\\living area\\' 63.87331
## '\\number of views\\' 42.45423
## '\\number of bedrooms\\' 35.57289
## '\\condition of the house\\' 27.05106
## '\\number of bathrooms\\' 25.71746
## Longitude 16.25183
## '\\Area of the house(excluding basement)\\' 16.13526
## living_area_renov 14.38876
## '\\Renovation Year\\' 12.85816
## lot_area_renov 8.31543
## '\\Postal Code\\' 6.56831
## '\\Number of schools nearby\\' 4.01476
## '\\number of floors\\' 3.27998
## '\\lot area\\' 0.04566
## '\\Distance from the airport\\' 0.00000
```

```
## [1] "variable importance of --> model_log"
```

```
## lm variable importance
##
## Overall
## Latitude 100.00000
## '\\grade of the house\\' 63.49513
## '\\Built Year\\' 39.30658
## '\\living area\\' 30.85679
## '\\condition of the house\\' 25.85637
## living_area_renov 24.40844
## '\\number of views\\' 23.01281
## '\\waterfront present\\' 19.73784
## '\\number of bathrooms\\' 18.30983
## '\\Postal Code\\' 18.06680
## '\\number of floors\\' 16.05220
## '\\Renovation Year\\' 12.04673
## '\\lot area\\' 7.21650
## '\\number of bedrooms\\' 6.90855
## Longitude 4.33343
## '\\Area of the house(excluding basement)\\' 1.72684
## lot_area_renov 1.61028
## '\\Distance from the airport\\' 0.05296
## '\\Number of schools nearby\\' 0.00000
```

3. Select 5 same important variable from varImp model and model_log

3.1 Subset data from df1 to df1_s and split data

```
# select varImp from above model for create new model
# subset data to df1_s

df1_s <- df1_log %>%
  select(lat = Latitude,
         grade = `grade of the house`,
         bld_yr = `Built Year`,
         lv_area = `living area`,
         no_view = `number of views`,
         log_price, Price)

prep_data_s <- split_data(df1_s)
train_data_s <- prep_data_s[[1]]
test_data_s <- prep_data_s[[2]]
```

3.2 Train model df1_s and log_price

3.2.1 Train

```
model_log_s <- train(log_price ~ lat + grade + bld_yr + lv_area + no_view,
                    data = train_data_s,
                    method = "lm")
```

3.2.2 calculate train error

3.2.2.1 predict train model_log_s

```
p_log_s_train <- predict(model_log_s, newdata = train_data_s)
```

3.2.2.2 calculate train error from model_log with logarithm price and real price_exp(log)

```
## evaluate train model_log_s
model_log_s_r_train <- r_train(train_data_s, p_log_s_train, model_log_s)

## [1] "--Evaulation of TRAIN--"
## [1] "MAE_log_train : 0.200017418153185"
## [1] "MSE_log_train : 0.0665572466155991"
## [1] "RMSE_log_train : 0.257986911713752"
## [1] "MAE_expo_train : 113400.360686776"
## [1] "MSE_expo_train : 35852543842.0516"
## [1] "RMSE_expo_train : 189347.679790515"
## [1] "MAE_model : 0.200652129732564"
## [1] "Rsquared_model : 0.757465703755575"
## [1] "RMSE_model : 0.258473837659618"
```

3.3 score~predict model(model_log_s)

```
p_log_s_test <- predict(model_log_s, newdata = test_data_s)
```

3.4 Evaluate model(model_log_s)

```
model_log_s_r_test <- r_test(test_data_s, p_log_s_test, model_log_s)
```

```
## [1] "--Evaulation of TEST--"
## [1] "MAE_log_test : 0.205768259455702"
## [1] "MSE_log_test : 0.0725033088154798"
## [1] "RMSE_log_test : 0.269264384602717"
## [1] "MAE_expo_test : 128244.175354101"
## [1] "MSE_expo_test : 196314300729.85"
## [1] "RMSE_expo_test : 443073.696725331"
## [1] "MAE_model : 0.200652129732564"
## [1] "Rsquared_model : 0.757465703755575"
## [1] "RMSE_model : 0.258473837659618"
```

Now we build model from important variable, calculate error for evaluate model, Next tuning the parameter, train control and resample technic

4. The final model

4.1 TrainControl repeatCV

```
set.seed(42)
ctrl <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 5
  #verboseIter = TRUE
)
```

4.2 Train final model df1_s and log_price

4.2.1 Train

```
model_lm_final <- train(log_price ~ lat + grade + bld_yr + lv_area + no_view,
  data = train_data_s,
  method = "lm",
  preProcess = c("center", "scale"),
  trControl = ctrl)
```

4.2.2 calculate train error

4.2.2.1 predict train model_lm_final

```
p_lm_final_train <- predict(model_lm_final, newdata = train_data_s)
```

4.2.2.2 calculate train error from model_lm_final with logarithm price and real price_exp(log)

```
model_lm_final_r_train <- r_train(train_data_s, p_lm_final_train, model_lm_final)
```

```
## [1] "--Evaulation of TRAIN--"
## [1] "MAE_log_train : 0.200017418153186"
## [1] "MSE_log_train : 0.0665572466155991"
## [1] "RMSE_log_train : 0.257986911713752"
## [1] "MAE_expo_train : 113400.360686775"
## [1] "MSE_expo_train : 35852543842.0519"
## [1] "RMSE_expo_train : 189347.679790516"
## [1] "MAE_model : 0.200122500185454"
## [1] "Rsquared_model : 0.7591336843618"
## [1] "RMSE_model : 0.258095431757117"
```

4.3 score~predict model(model_lm_final)

```
# test model_lm_final
p_lm_final_test <- predict(model_lm_final, newdata = test_data_s)
```

4.4 Evaluate model(model_lm_final)

```
model_lm_final_r_test <- r_test(test_data_s, p_lm_final_test, model_lm_final)
```

```
## [1] "--Evaulation of TEST--"
## [1] "MAE_log_test : 0.205768259455706"
## [1] "MSE_log_test : 0.0725033088154796"
## [1] "RMSE_log_test : 0.269264384602717"
## [1] "MAE_expo_test : 128244.175354096"
## [1] "MSE_expo_test : 196314300729.781"
## [1] "RMSE_expo_test : 443073.696725252"
## [1] "MAE_model : 0.200122500185454"
## [1] "Rsquared_model : 0.7591336843618"
## [1] "RMSE_model : 0.258095431757117"
```

Switch method to knn

```
model_knn_final <- train(log_price ~ lat + grade + bld_yr + lv_area + no_view,
  data = train_data_s,
  method = "knn",
  tuneGrid = data.frame(k=c(5, 7, 13)),
  preProcess = c("center", "scale"),
  #preProcess = c("range", "zv", "nzv"),
  trControl = ctrl)
```

predict train model_knn_final

```
p_knn_final_train <- predict(model_knn_final, newdata = train_data_s)
```

evaluate train model_knn_final

```
model_knn_final_r_train <- r_train(train_data_s, p_knn_final_train, model_knn_final)
```

```
## [1] "--Evaulation of TRAIN--"
## [1] "MAE_log_train : 0.140665174852055"
## [1] "MSE_log_train : 0.0371433299380822"
## [1] "RMSE_log_train : 0.192726048934964"
## [1] "MAE_expo_train : 81946.2331718462"
## [1] "MSE_expo_train : 25552644789.7944"
## [1] "RMSE_expo_train : 159851.946468582"
## [1] "MAE_model : 0.158835378279188" "MAE_model : 0.155883783285012"
## [3] "MAE_model : 0.153229130059616"
## [1] "Rsquared_model : 0.831110269286084" "Rsquared_model : 0.836307528904411"
## [3] "Rsquared_model : 0.841479039253118"
## [1] "RMSE_model : 0.21640668756905" "RMSE_model : 0.212880120531821"
## [3] "RMSE_model : 0.209466472413462"
```

test model_knn_final

```
p_knn_final_test <- predict(model_knn_final, newdata = test_data_s)
```

evaluate test model_knn_final

```
model_knn_final_r_test <- r_test(test_data_s, p_knn_final_test, model_knn_final)
```

```
## [1] "--Evaulation of TEST--"
## [1] "MAE_log_test : 0.158201565950789"
## [1] "MSE_log_test : 0.048164704298204"
## [1] "RMSE_log_test : 0.219464585521683"
## [1] "MAE_expo_test : 93951.1047196991"
## [1] "MSE_expo_test : 44100247868.9621"
```

```
## [1] "RMSE_expo_test : 210000.590163366"
## [1] "MAE_model : 0.158835378279188" "MAE_model : 0.155883783285012"
## [3] "MAE_model : 0.153229130059616"
## [1] "Rsquared_model : 0.831110269286084" "Rsquared_model : 0.836307528904411"
## [3] "Rsquared_model : 0.841479039253118"
## [1] "RMSE_model : 0.21640668756905" "RMSE_model : 0.212880120531821"
## [3] "RMSE_model : 0.209466472413462"
```

Summary Evaluation

create vector model, model_log, model_log_s, model_lm_final, model_knn_final

```
model_name <- c("model",
               "model_log",
               "model_log_s",
               "model_lm_final",
               "model_knn_final")
```

create vector MAE-train test, MSE-train test, RMSE-train test, Rsquared train

```
mae_train_log <- c(0,
                  model_log_r_train[[1]],
                  model_log_s_r_train[[1]],
                  model_lm_final_r_train[[1]],
                  model_knn_final_r_train[[1]])

mae_train <- c(model[[4]][[4]],
              model_log_r_train[[4]],
              model_log_s_r_train[[4]],
              model_lm_final_r_train[[4]],
              model_knn_final_r_train[[4]])

mae_test_log <- c(0,
                 model_log_r_test[[1]],
                 model_log_s_r_test[[1]],
                 model_lm_final_r_test[[1]],
                 model_knn_final_r_test[[1]])

mae_test <- c(model[[4]][[4]],
             model_log_r_test[[4]],
             model_log_s_r_test[[4]],
             model_lm_final_r_test[[4]],
             model_knn_final_r_test[[4]])

mse_train_log <- c(0,
                  model_log_r_train[[2]],
                  model_log_s_r_train[[2]],
                  model_lm_final_r_train[[2]],
                  model_knn_final_r_train[[2]])
```

```

mse_train <- c(0,
              model_log_r_train[[5]],
              model_log_s_r_train[[5]],
              model_lm_final_r_train[[5]],
              model_knn_final_r_train[[5]])

mse_test_log <- c(0,
                 model_log_r_test[[2]],
                 model_log_s_r_test[[2]],
                 model_lm_final_r_test[[2]],
                 model_knn_final_r_test[[2]])

mse_test <- c(0,
              model_log_r_test[[5]],
              model_log_s_r_test[[5]],
              model_lm_final_r_test[[5]],
              model_knn_final_r_test[[5]])

rmse_train_log <- c(0,
                   model_log_r_train[[3]],
                   model_log_s_r_train[[3]],
                   model_lm_final_r_train[[3]],
                   model_knn_final_r_train[[3]])

rmse_train <- c(model[[4]][[2]],
               model_log_r_train[[6]],
               model_log_s_r_train[[6]],
               model_lm_final_r_train[[6]],
               model_knn_final_r_train[[6]])

rmse_test_log <- c(0,
                  model_log_r_test[[3]],
                  model_log_s_r_test[[3]],
                  model_lm_final_r_test[[3]],
                  model_knn_final_r_test[[3]])

rmse_test <- c(model[[4]][[2]],
               model_log_r_test[[6]],
               model_log_s_r_test[[6]],
               model_lm_final_r_test[[6]],
               model_knn_final_r_test[[6]])

rsquared_model_train <- c(model[[4]][[3]],
                         model_log[[4]][[3]],
                         model_log_s[[4]][[3]],
                         model_lm_final[[4]][[3]],
                         model_knn_final[[4]][[3,3]])

```

Then create df to compare evaluation

```

sum_eva <- data.frame(model_name,
                      mae_train_log, mae_train,

```

```

mae_test_log, mae_test,
mse_train_log, mse_train,
mse_test_log, mse_test,
rmse_train_log, rmse_train,
rmse_test_log, rmse_test,
rsquared_model_train)

```

Show compare error evaluation all model

```

##          model_name mae_train_log mae_train mae_test_log mae_test
## 1          model      0.0000000 123219.80   0.0000000 123219.8
## 2      model_log      0.1868766 106108.30   0.1934462 119667.9
## 3      model_log_s      0.2000174 113400.36   0.2057683 128244.2
## 4 model_lm_final      0.2000174 113400.36   0.2057683 128244.2
## 5 model_knn_final      0.1406652  81946.23   0.1582016  93951.1

##          model_name mse_train_log   mse_train mse_test_log   mse_test
## 1          model      0.00000000         0   0.00000000         0
## 2      model_log      0.05922343 32179140733   0.06483457 125501870361
## 3      model_log_s      0.06655725 35852543842   0.07250331 196314300730
## 4 model_lm_final      0.06655725 35852543842   0.07250331 196314300730
## 5 model_knn_final      0.03714333 25552644790   0.04816470 44100247869

##          model_name rmse_train_log rmse_train rmse_test_log rmse_test
## 1          model      0.0000000    192549.8   0.0000000    192549.8
## 2      model_log      0.2433587    179385.5   0.2546263    354262.4
## 3      model_log_s      0.2579869    189347.7   0.2692644    443073.7
## 4 model_lm_final      0.2579869    189347.7   0.2692644    443073.7
## 5 model_knn_final      0.1927260    159851.9   0.2194646    210000.6

##          model_name rsquared_model_train
## 1          model              0.7110370
## 2      model_log              0.7832968
## 3      model_log_s              0.7574657
## 4 model_lm_final              0.7591337
## 5 model_knn_final              0.8414790

```

CONCLUSION

Final model use K-Nearest Neighbors method and train control for minimize all error and maximize Rsquared as below. Tuning the parameter, resample, train control or switch method may build the lower error model.

```

##          model_name mae_train_log mae_train mae_test_log mae_test
## 5 model_knn_final      0.1406652  81946.23   0.1582016  93951.1

##          model_name mse_train_log   mse_train mse_test_log   mse_test
## 5 model_knn_final      0.03714333 25552644790   0.0481647 44100247869

##          model_name rmse_train_log rmse_train rmse_test_log rmse_test
## 5 model_knn_final      0.192726    159851.9   0.2194646    210000.6

```



```
##          model_name rsquared_model_train
## 5 model_knn_final      0.841479
```