ML Homework - House Price India

Naphon Seeluang

2023-08-10

Instruction

Trial to generate regression model to predict House Price India dataset.

Load Library that use in this homework

```
library(caret)
```

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

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

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.1     v tidyr     1.3.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(readxl)
library(ggplot2)
```

Import and check House Price India File

```
house_price_india <- read_excel("House Price India.xlsx")
head(house_price_india)
```

```
## # A tibble: 6 x 23
##           id  Date `number of bedrooms` `number of bathrooms` `living area`
##        <dbl> <dbl>                <dbl>                 <dbl>         <dbl>
## 1 6762810145 42491                    5                  2.5           3650
## 2 6762810635 42491                    4                  2.5           2920
## 3 6762810998 42491                    5                  2.75          2910
## 4 6762812605 42491                    4                  2.5           3310
## 5 6762812919 42491                    3                  2             2710
## 6 6762813105 42491                    3                  2.5           2600
## # i 18 more variables: `lot area` <dbl>, `number of floors` <dbl>,
```

```
## #   `waterfront present` <dbl>, `number of views` <dbl>,
## #   `condition of the house` <dbl>, `grade of the house` <dbl>,
## #   `Area of the house(excluding basement)` <dbl>,
## #   `Area of the basement` <dbl>, `Built Year` <dbl>, `Renovation Year` <dbl>,
## #   `Postal Code` <dbl>, Lattitude <dbl>, Longitude <dbl>,
## #   living_area_renov <dbl>, lot_area_renov <dbl>, ...
```

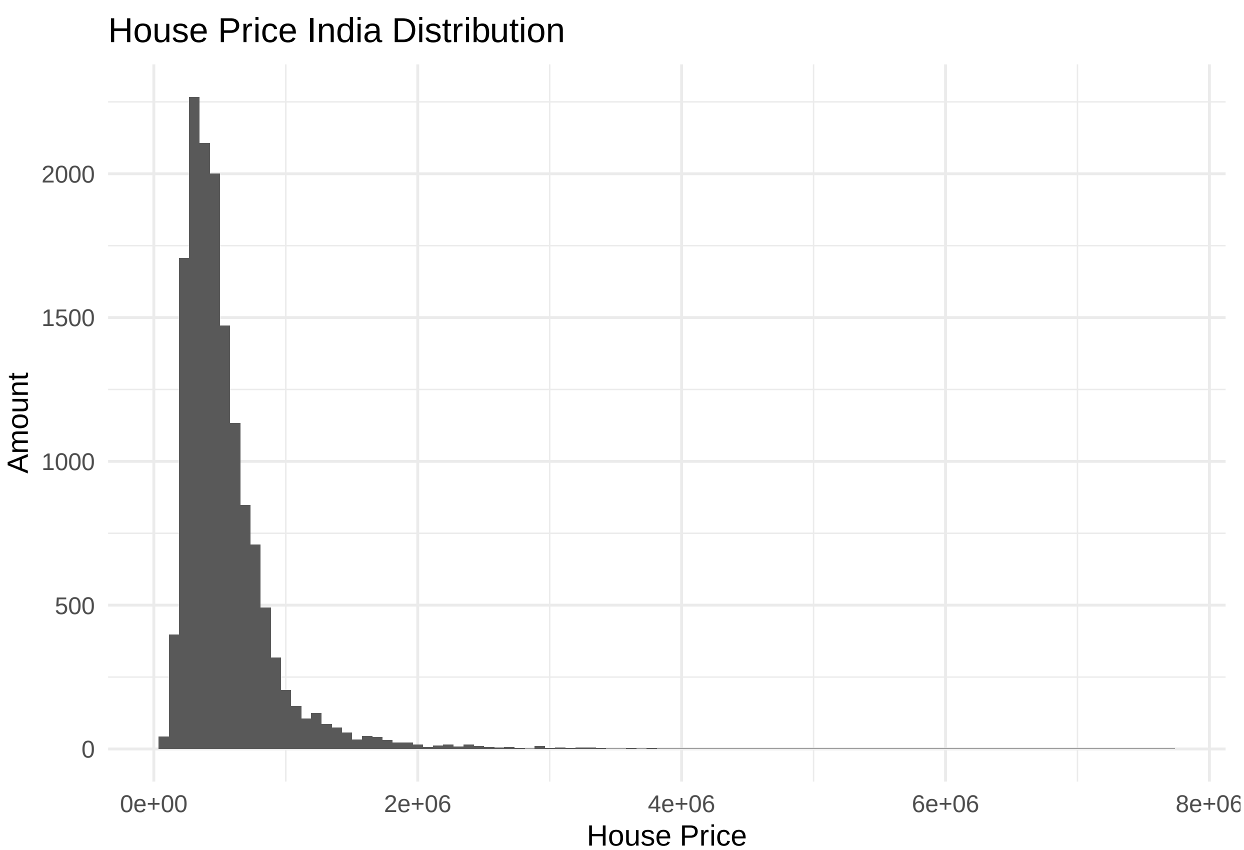
Subset columns and check missing value

```
# subset all column
full_df <- house_price_india

# check NA
full_df %>%
  complete.cases() %>%
  mean()
```

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

```
# check distribution of price
ggplot(full_df, aes(Price)) +
  geom_histogram(bins = 100) +
  theme_minimal() +
  labs(
    title = "House Price India Distribution",
    x = "House Price",
    y = "Amount"
  )
```

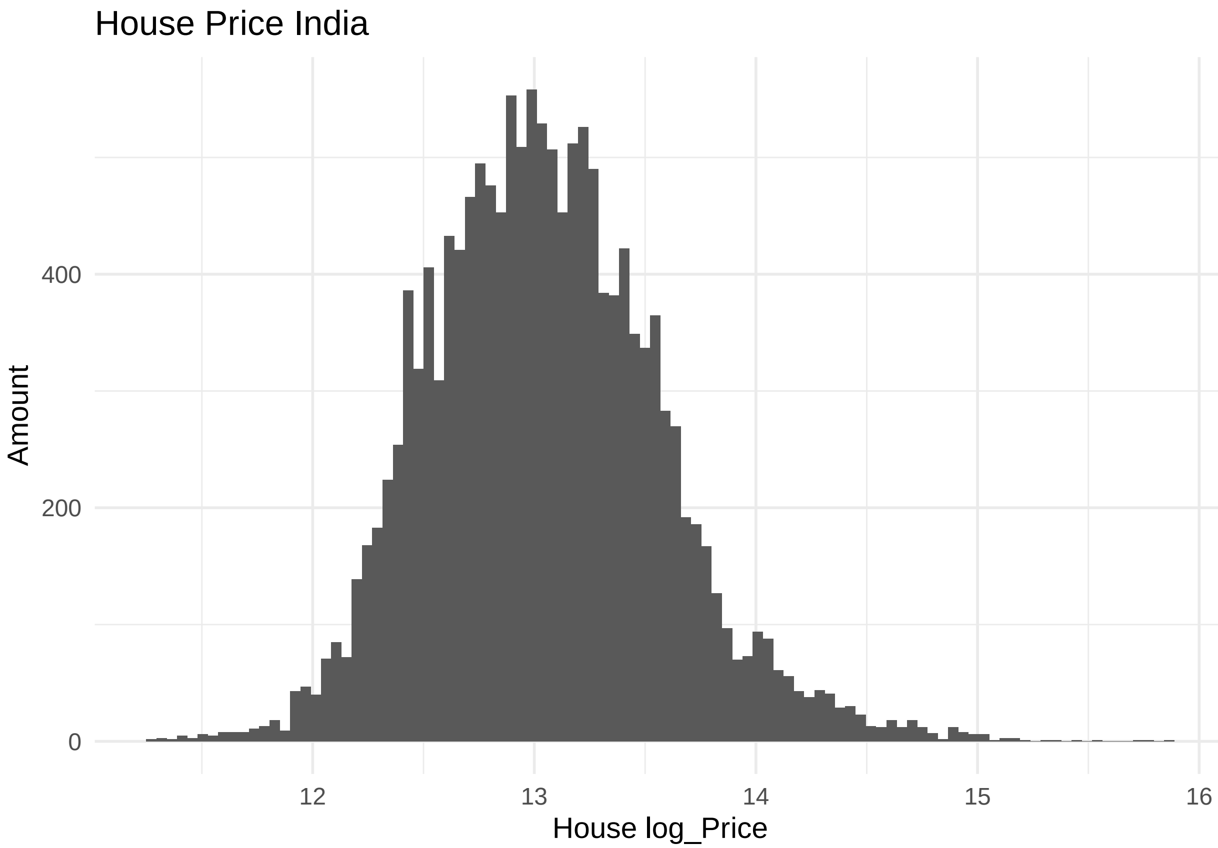


Price is right skewed.

Take log to Price to make it normal distribution

```
# prep data
clean_df <- full_df %>%
  mutate(log_price = log(Price))

# right skewed to normal dist
ggplot(clean_df, aes(log_price)) +
  geom_histogram(bins = 100) +
  theme_minimal() +
  labs(
    title = "House Price India",
    x = "House log_Price",
    y = "Amount"
  )
```



Price is normal distribution.

Process ML

```
# 1. split data 80% train, 20% test
split_data <- function(df) {
  set.seed(44)
  n <- nrow(df)
```

```r
  train_id <- sample(1:n, size = 0.8*n)
  train_df <- df[train_id, ]
  test_df <- df[-train_id, ]
  # return
  list(training = train_df,
       testing = test_df)
}

prep_data <- split_data(clean_df)
train_df <- prep_data[[1]]
test_df <- prep_data[[2]]
```

Check accuracy before take log to Price

```r
# Normal Price
# 2.1 train model
set.seed(44)
lm_model <- train(Price ~ .,
                  data = train_df,
                  # ML algorithm
                  method = "lm")

lm_model
```

```
## Linear Regression
##
## 11696 samples
##    23 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11696, 11696, 11696, 11696, 11696, 11696, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   120058.8  0.8978819  62364.01
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# 3.1 score model
p <- predict(lm_model, newdata=test_df)

# 4.1 evaluate model
# mean absolute error
mae <- mean(abs(p - test_df$Price))
mae
```

```
## [1] 59875.77
```

```r
# root mean square error
rmse <- sqrt(mean((p - test_df$Price)**2))
rmse
```

```
## [1] 96862.66
```

Result after take log to Price

```r
# Take log to Price
# 2.2 train model
lm_model_log <- train(log_price ~ .,
                  data = train_df,
                  # ML algorithm
                  method = "lm")

lm_model_log
```

```
## Linear Regression
##
## 11696 samples
##    23 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11696, 11696, 11696, 11696, 11696, 11696, ...
## Resampling results:
##
##   RMSE        Rsquared   MAE
##   0.07413856  0.9803169  0.04449027
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# 3.2 score model
p_log <- predict(lm_model_log, newdata=test_df)


# 4. evaluate model
# change log value back to normal value
# mean absolute error
mae <- mean(abs(exp(p_log) - exp(test_df$log_price)))
mae
```

```
## [1] 21762.25
```

```r
# root mean square error
rmse <- sqrt(mean((exp(p_log) - exp(test_df$log_price))**2))
rmse
```

```
## [1] 35446.14
```

Check features that most affect to Price

```r
varImp(lm_model_log)
```

```
## lm variable importance
##
##   only 20 most important variables shown (out of 22)
##
##                                                     Overall
## id                                                 100.0000
## Price                                               54.3876
## Lattitude                                            4.4488
```

```
## `\\`condition of the house\\``                 3.4773
## `\\`grade of the house\\``                     3.0518
## `\\`Built Year\\``                             2.3999
## `\\`number of bathrooms\\``                    2.1992
## `\\`number of bedrooms\\``                     1.8331
## `\\`Renovation Year\\``                        1.6179
## living_area_renov                              1.5649
## Longitude                                      1.5263
## `\\`number of views\\``                        1.5213
## `\\`Area of the house(excluding basement)\\``  1.3109
## `\\`number of floors\\``                       0.8649
## Date                                           0.8361
## `\\`Number of schools nearby\\``               0.5293
## `\\`living area\\``                            0.4050
## `\\`waterfront present\\``                     0.3765
## `\\`lot area\\``                               0.2833
## `\\`Postal Code\\``                            0.2709
```

Prep data by subset the most columns that affect to Price

```
clean_df <- full_df %>%
  select(Price, id, Lattitude,
         house_condition = `condition of the house`,
         house_grade = `grade of the house`,
         built_year = `Built Year`) %>%
  mutate(log_price = log(Price))
```

Compared prediction of the result

```
# 1. split data 80% train, 20% test
split_data <- function(df) {
  set.seed(44)
  n <- nrow(df)
  train_id <- sample(1:n, size = 0.8*n)
  train_df <- df[train_id, ]
  test_df <- df[-train_id, ]
  # return
  list(training = train_df,
       testing = test_df)
}

prep_data <- split_data(clean_df)
train_df <- prep_data[[1]]
test_df <- prep_data[[2]]

# Normal Price
# 2.1 train model
set.seed(44)
lm_model <- train(Price ~ .,
                  data = train_df,
                  # ML algorithm
                  method = "lm")
```

```r
lm_model
```

```
## Linear Regression
##
## 11696 samples
##     6 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11696, 11696, 11696, 11696, 11696, 11696, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   125336.6  0.8886798  60212.62
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# 3.1 score model
p <- predict(lm_model, newdata=test_df)

# 4.1 evaluate model
# mean absolute error
mae <- mean(abs(p - test_df$Price))
mae
```

```
## [1] 55500.18
```

```r
# root mean square error
rmse <- sqrt(mean((p - test_df$Price)**2))
rmse
```

```
## [1] 97246.86
```

```r
# Take log to Price
# 2.2 train model
lm_model_log <- train(log_price ~ .,
                  data = train_df,
                  # ML algorithm
                  method = "lm")

lm_model_log
```

```
## Linear Regression
##
## 11696 samples
##     6 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11696, 11696, 11696, 11696, 11696, 11696, ...
## Resampling results:
##
##   RMSE        Rsquared   MAE
##   0.07455094  0.9800978  0.0447776
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# 3.2 score model
p_log <- predict(lm_model_log, newdata=test_df)


# 4. evaluate model
# change log value back to normal value
# mean absolute error
mae <- mean(abs(exp(p_log) - exp(test_df$log_price)))
mae
```

```
## [1] 21890.39
```

```r
# root mean square error
rmse <- sqrt(mean((exp(p_log) - exp(test_df$log_price))**2))
rmse
```

```
## [1] 36167.21
```

Summary

1. Price is right skewed decrease accuracy and increase error metrics of the model
2. Normal distribution from taking log to Price help to increase accuracy and decrease error metrics of the model