Predict house price in India

Pariya

In this project, I use the linear regression model in R language to predict the price because this model is fast to train and can be interpreted.

I used the data from this link https://data.world/dataindianset2000/house-price-india

Load library

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
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     1.0.2     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 error
```

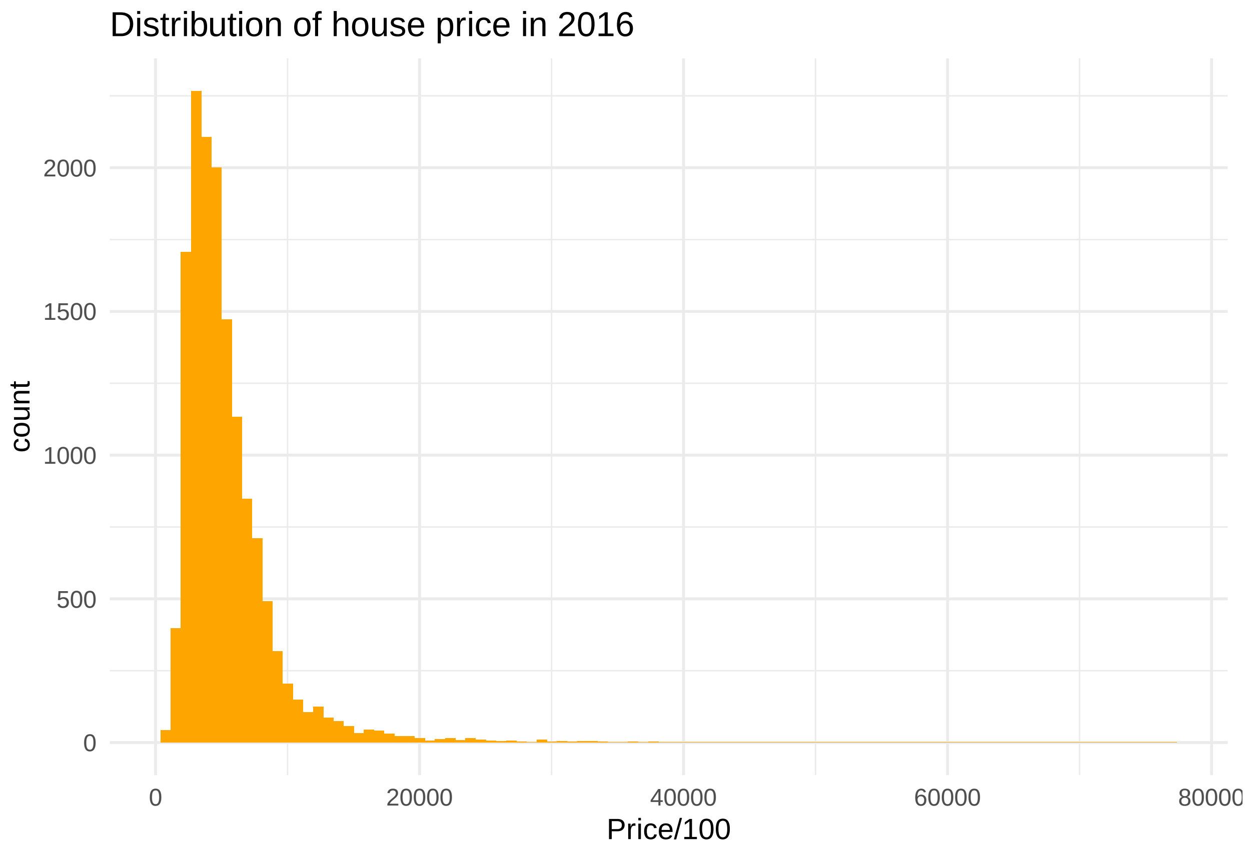
```r
library(mlbench)
library(readxl)
library(ggplot2)
```

Load the house_price_india excel file source from data.world.

```r
df2016 <- read_excel("House_Price_India.xlsx", sheet = 1)
```

Plot a histogram to find the distribution of house prices in 2016.

```r
ggplot(df2016, aes(Price/100)) +
  geom_histogram(bins = 100,
                 fill = "orange") +
  theme_minimal() +
  labs(title = "Distribution of house price in 2016",
       x = "Price/100")
```



The distribution is right-skew. It is not good to use this data to predict without changing to a normal distribution, but I will use two methods to show the difference between the right skew distribution and the normal distribution.

Split data for train and test

```
split_data <- function(df) {
  set.seed(2)
  n <- nrow(df)
  train_id <- sample(1:n, size = 0.8*n)
  train_df <- df[train_id, ]
  test_df <- df[-train_id, ]
  list(train_df, test_df)
}

prep_data <- split_data(df2016)
train_data <- prep_data[[1]]
test_data <- prep_data[[2]]
```

Train model

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

Predict

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

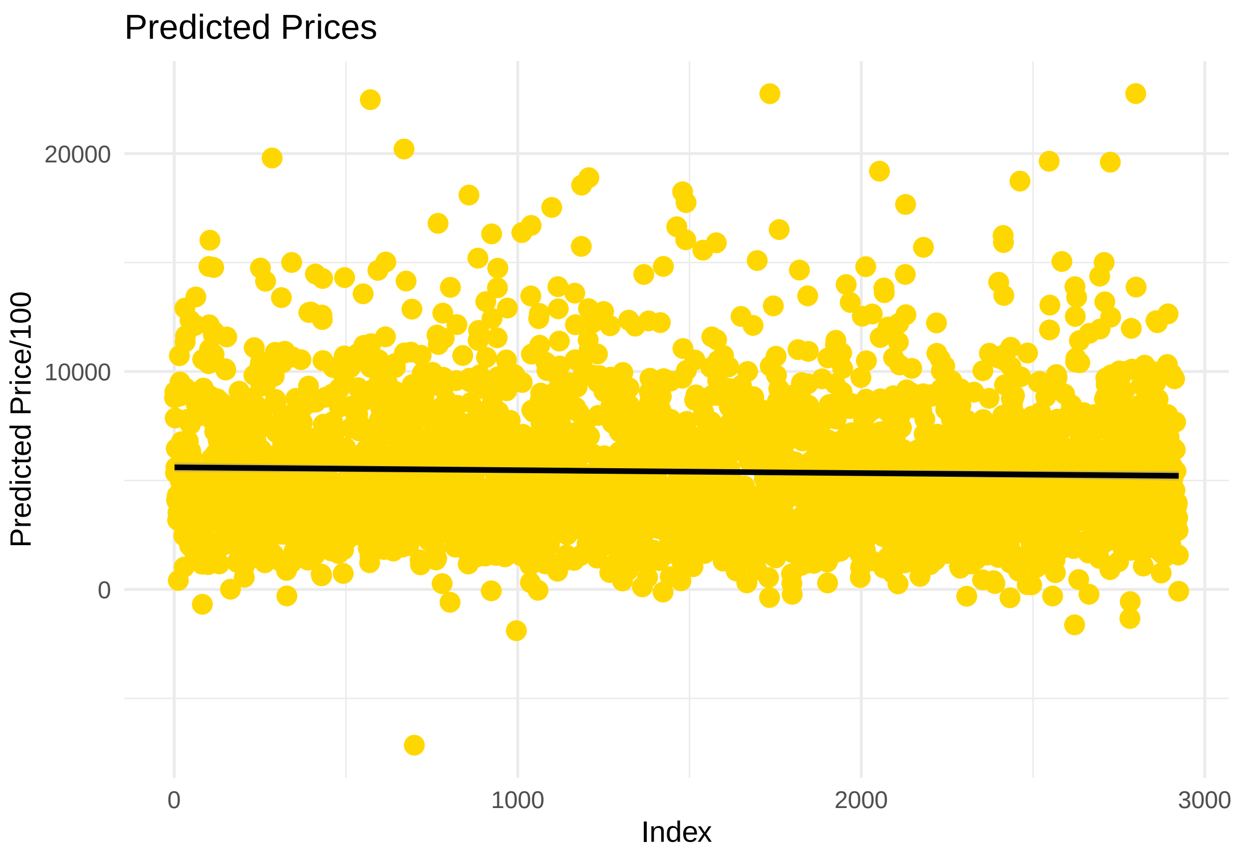
Create a data frame with predicted prices

```
predict_data_before <- data.frame(Predicted_Price = p/100)
```

Visualize predicted prices

```
ggplot(predict_data_before, aes(x = 1:length(Predicted_Price), y = Predicted_Price)) +
  geom_point(color = "gold", size = 3) +
  geom_smooth(method = "lm",
              col = "black") +
  labs(title = "Predicted Prices",
       x = "Index",
       y = "Predicted Price/100") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Evaluate model (find mae, mse, rmse, mape)

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

```
}

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

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

cal_mape <- function(actual, predict) {
  error <- predict - actual
  mean(abs(error/actual)) * 100
}

result_mae <- cal_mae(test_data$Price, p)
result_mse <- cal_mse(test_data$Price, p)
result_rmse <- cal_rmse(test_data$Price, p)
result_mape <- cal_mape(test_data$Price, p)

cat("MAE:", result_mae, "\n")
```

```
## MAE: 125264.8
```

```
cat("MSE:", result_mse, "\n")
```

```
## MSE: 36116190734
```

```
cat("RMSE:", result_rmse, "\n")
```

```
## RMSE: 190042.6
```

```
cat("MAPE:", result_mape, "\n")
```

```
## MAPE: 25.13159
```

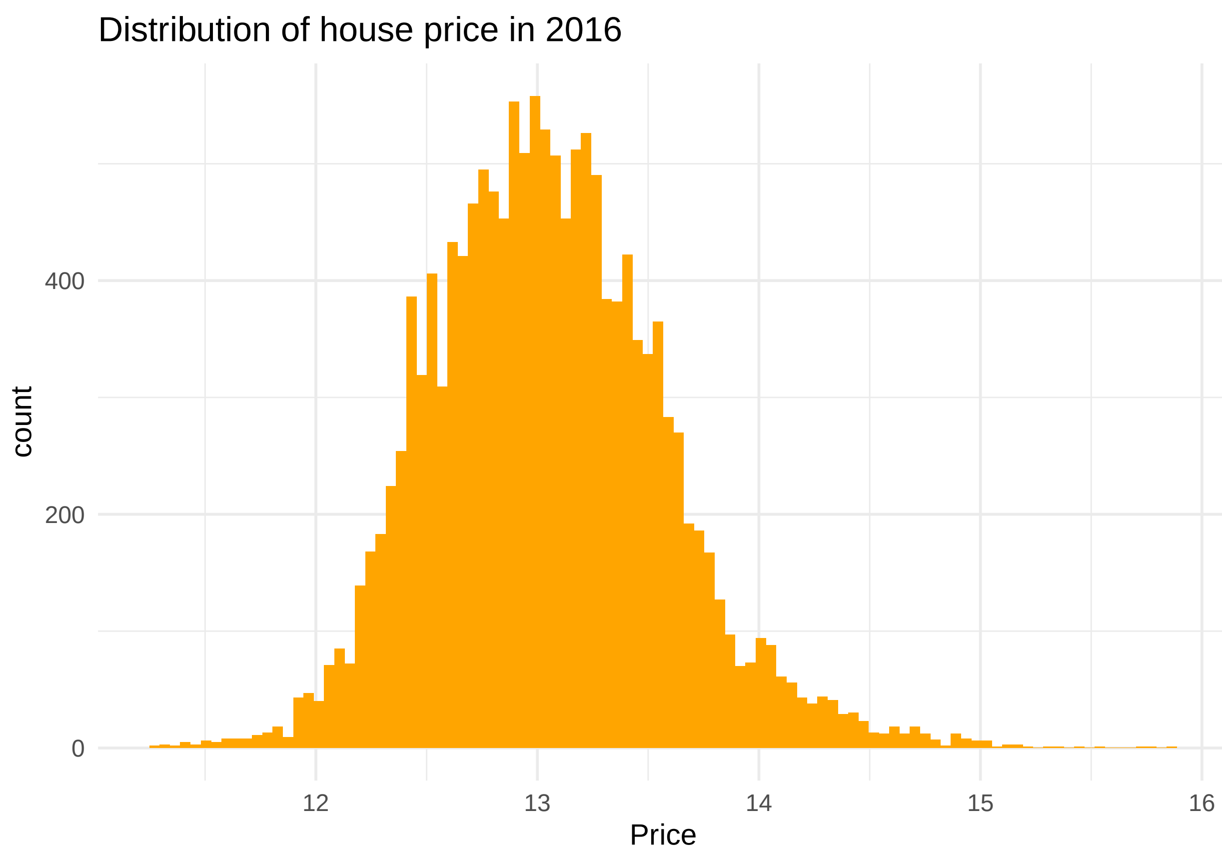
The second method changes to a normal distribution.

Plot a histogram to find the distribution of house prices in 2016.

Change to normal distribution

```
df2016_log <- df2016 %>%
  mutate(log_price = log(Price))

ggplot(df2016_log, aes(log_price)) +
  geom_histogram(bins = 100,
                 fill = "orange") +
  theme_minimal() +
  labs(title = "Distribution of house price in 2016",
       x = "Price")
```



No need to split data again use the data that has been split in the first method.

```
prep_data_log <- split_data(df2016_log)
train_data_log <- prep_data_log[[1]]
test_data_log <- prep_data_log[[2]]
```

Train model

```
model_log <- train(log_price ~ .,
                   data = train_data_log[ ,-c(1,2,23)],
                   method = "lm")
```

Predict

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

Create a data frame with predicted prices

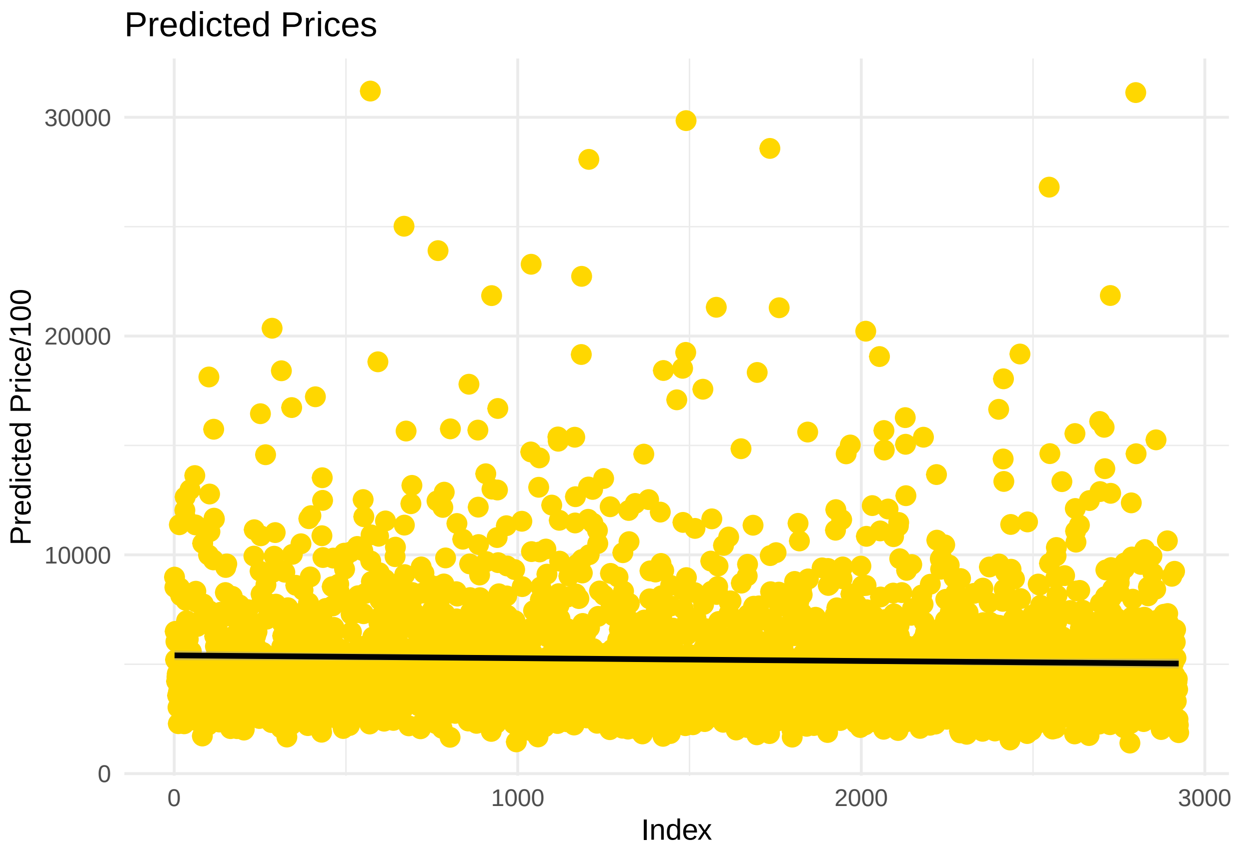
```
predict_data_after <- data.frame(Predicted_Price = exp(p_log)/100)
```

Visualize predicted prices

```
ggplot(predict_data_after, aes(x = 1:length(Predicted_Price), y = Predicted_Price)) +
  geom_point(color = "gold", size = 3) +
  geom_smooth(method = "lm",
```

```
              col = "black") +
  labs(title = "Predicted Prices",
       x = "Index",
       y = "Predicted Price/100") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Evaluate model (find mae, mse, rmse, mape)

```
cal_mae_log <- function(actual, predict) {
  error <- exp(actual) - exp(predict)
  mean(abs(error))
}
cal_mse_log <- function(actual, predict) {
  error <- exp(actual) - exp(predict)
  mean(error^2)
}

cal_rmse_log <- function(actual, predict) {
  error <- exp(actual) - exp(predict)
  sqrt(mean(error^2))
}

cal_mape_log <- function(actual, predict) {
  error <- exp(predict) - exp(actual)
  mean(abs(error / exp(actual))) * 100
```

```r
}

result_mae_normal <- cal_mae_log(test_data_log$log_price, p_log)
result_mse_normal <- cal_mse_log(test_data_log$log_price, p_log)
result_rmse_normal <- cal_rmse_log(test_data_log$log_price, p_log)
result_mape_normal <- cal_mape_log(test_data_log$log_price, p_log)

cat("MAE_normal:", result_mae_normal, "\n")
```

```
## MAE_normal: 109146.6
```

```r
cat("MSE_normal:", result_mse_normal, "\n")
```

```
## MSE_normal: 32242671242
```

```r
cat("RMSE_normal:", result_rmse_normal, "\n")
```

```
## RMSE_normal: 179562.4
```

```r
cat("MAPE_normal:", result_mape_normal, "\n")
```

```
## MAPE_normal: 19.21835
```

Conclusion

According to the results of MAE, MSE, RMSE, MAPE the second method, which normalizes the data first and then performs the LM process, is better because MAE, MSE, RMSE, MAPE from the second method are close to zero compared to the first method, which does not normalize the data.