Document - Machine Learning to predict India house price between 2016 and 2017

Source data: https://data.world/dataindianset2000/house-price-india (https://data.world/dataindianset2000/house-price-india)

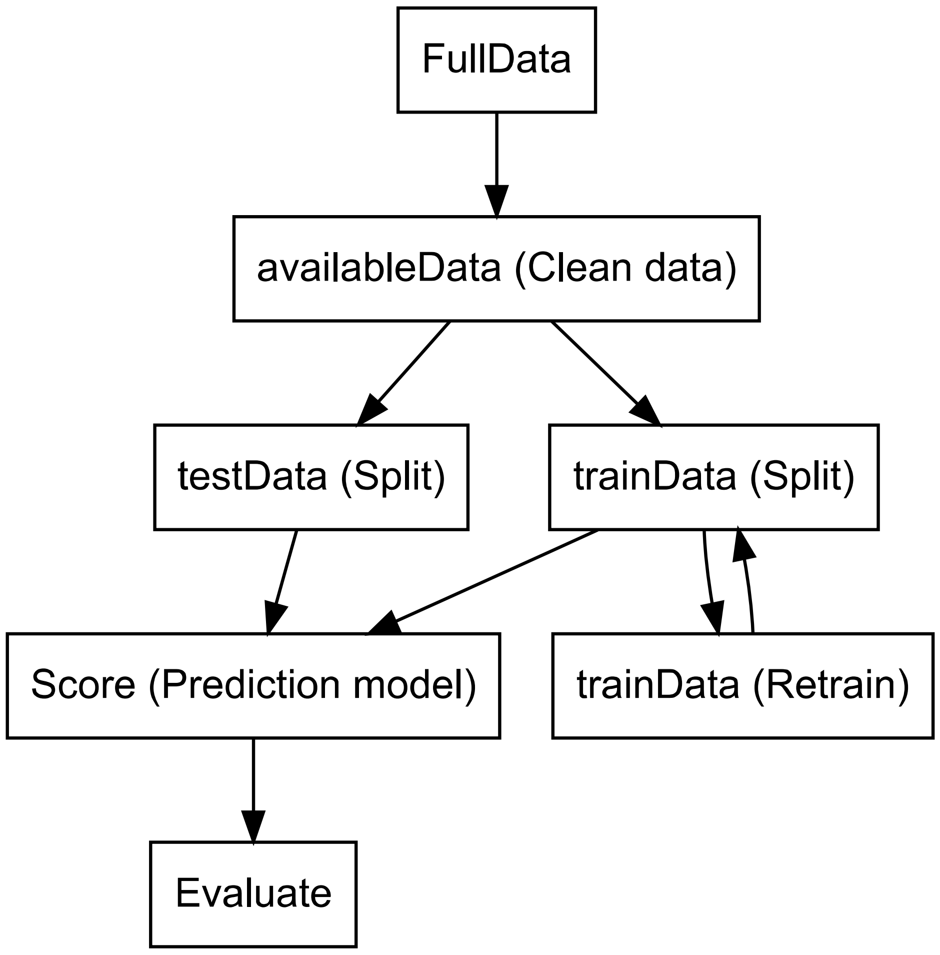
Phattharachai Maichin

2023-10-02

Before start

This document I created to cover how to use "R programming" to manipulate and analyst data to find relationships among variables to predict India house price based on data in year 2016 and 2017.

Introduction to Machine Learning: I have learnt about Basic Machine Learning Workflow and I summarized it in the picture below to demonstrate a basic machine learning workflow.



Install packages

Packages have been used in this project are “tidyverse” for manage data, “caret” for analyst data, and “readxl” for read excel file.

Preview dataset

There are many ways to preview dataset such as View(), head(), glimpse(), str() and other.

```
# Original data is stored as shown in folder name
df2016 <- read_excel("C:/Users/phatt/Desktop/Project data analyst/Document - ML to predict house
price in India between 2016 and 2017/House Price India.xlsx", sheet = "House 2016")
df2017 <- read_excel("C:/Users/phatt/Desktop/Project data analyst/Document - ML to predict house
price in India between 2016 and 2017/House Price India.xlsx", sheet = "House 2017")

# Preview dataset
str(df2016)
```

```
## tibble [14,620 × 23] (S3: tbl_df/tbl/data.frame)
##  $ id                                   : num [1:14620] 6.76e+09 6.76e+09 6.76e+09 6.76e+09
6.76e+09 ...
##  $ Date                                 : num [1:14620] 42491 42491 42491 42491 42491 ...
##  $ number of bedrooms                   : num [1:14620] 5 4 5 4 3 3 5 3 3 4 ...
##  $ number of bathrooms                  : num [1:14620] 2.5 2.5 2.75 2.5 2 2.5 3.25 1.75 2.5
2.25 ...
##  $ living area                          : num [1:14620] 3650 2920 2910 3310 2710 2600 3660 22
40 2390 2200 ...
##  $ lot area                             : num [1:14620] 9050 4000 9480 42998 4500 ...
##  $ number of floors                     : num [1:14620] 2 1.5 1.5 2 1.5 1 2 2 1 1.5 ...
##  $ waterfront present                   : num [1:14620] 0 0 0 0 0 0 0 0 0 0 ...
##  $ number of views                      : num [1:14620] 4 0 0 0 0 0 2 0 2 0 ...
##  $ condition of the house               : num [1:14620] 5 5 3 3 4 4 3 5 4 5 ...
##  $ grade of the house                   : num [1:14620] 10 8 8 9 8 9 10 8 8 7 ...
##  $ Area of the house(excluding basement): num [1:14620] 3370 1910 2910 3310 1880 1700 3660 15
50 1440 1300 ...
##  $ Area of the basement                 : num [1:14620] 280 1010 0 0 830 900 0 690 950 900
...
##  $ Built Year                           : num [1:14620] 1921 1909 1939 2001 1929 ...
##  $ Renovation Year                      : num [1:14620] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Postal Code                          : num [1:14620] 122003 122004 122004 122005 122006
...
##  $ Lattitude                            : num [1:14620] 52.9 52.9 52.9 53 52.9 ...
##  $ Longitude                            : num [1:14620] -115 -114 -114 -114 -114 ...
##  $ living_area_renov                    : num [1:14620] 2880 2470 2940 3350 2060 2380 3320 15
70 2010 2320 ...
##  $ lot_area_renov                       : num [1:14620] 5400 4000 6600 42847 4500 ...
##  $ Number of schools nearby             : num [1:14620] 2 2 1 3 1 1 3 3 1 2 ...
##  $ Distance from the airport            : num [1:14620] 58 51 53 76 51 67 72 71 73 53 ...
##  $ Price                                : num [1:14620] 2380000 1400000 1200000 838000 805000
790000 785000 750000 750000 698000 ...
```

As data structure showed that “Date” column is in numeric format, so there are many ways to manage it such as (1) format Date in Excel file and load file age (2) format in R using “convert_date” in “datetimeutils” package.

```
# (1) format Date in Excel file and load file age
## After change date format in excel load file age
### df2016 <- read_excel("C:/Users/phatt/Desktop/Project data analyst/Document - ML to predict h
ouse price in India between 2016 and 2017/House Price India format date.xlsx", sheet = "House 20
16")
### df2017 <- read_excel("C:/Users/phatt/Desktop/Project data analyst/Document - ML to predict h
ouse price in India between 2016 and 2017/House Price India format date.xlsx", sheet = "House 20
17")

# (2) format in R using "convert_date" in "datetimeutils" package
## install.packages("datetimeutils")
library(datetimeutils)
```

```
##
## Attaching package: 'datetimeutils'
```

```
## The following objects are masked from 'package:lubridate':
##
##     hour, mday, mday<-, minute, month, second, year
```

```
df2016$Date <- df2016$Date %>%
  convert_date(type = "Excel", fraction = FALSE, tz = "UTC")

df2017$Date <- df2017$Date %>%
  convert_date(type = "Excel", fraction = FALSE, tz = "UTC")

# Preview dataset again
str(df2016)
```

```
## tibble [14,620 × 23] (S3: tbl_df/tbl/data.frame)
##  $ id                                   : num [1:14620] 6.76e+09 6.76e+09 6.76e+09 6.76e+09
6.76e+09 ...
##  $ Date                                 : Date[1:14620], format: "2016-05-01" "2016-05-01"
...
##  $ number of bedrooms                   : num [1:14620] 5 4 5 4 3 3 5 3 3 4 ...
##  $ number of bathrooms                  : num [1:14620] 2.5 2.5 2.75 2.5 2 2.5 3.25 1.75 2.5
2.25 ...
##  $ living area                          : num [1:14620] 3650 2920 2910 3310 2710 2600 3660 22
40 2390 2200 ...
##  $ lot area                             : num [1:14620] 9050 4000 9480 42998 4500 ...
##  $ number of floors                     : num [1:14620] 2 1.5 1.5 2 1.5 1 2 2 1 1.5 ...
##  $ waterfront present                   : num [1:14620] 0 0 0 0 0 0 0 0 0 0 ...
##  $ number of views                      : num [1:14620] 4 0 0 0 0 0 2 0 2 0 ...
##  $ condition of the house               : num [1:14620] 5 5 3 3 4 4 3 5 4 5 ...
##  $ grade of the house                   : num [1:14620] 10 8 8 9 8 9 10 8 8 7 ...
##  $ Area of the house(excluding basement): num [1:14620] 3370 1910 2910 3310 1880 1700 3660 15
50 1440 1300 ...
##  $ Area of the basement                 : num [1:14620] 280 1010 0 0 830 900 0 690 950 900
...
##  $ Built Year                           : num [1:14620] 1921 1909 1939 2001 1929 ...
##  $ Renovation Year                      : num [1:14620] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Postal Code                          : num [1:14620] 122003 122004 122004 122005 122006
...
##  $ Lattitude                            : num [1:14620] 52.9 52.9 52.9 53 52.9 ...
##  $ Longitude                            : num [1:14620] -115 -114 -114 -114 -114 ...
##  $ living_area_renov                    : num [1:14620] 2880 2470 2940 3350 2060 2380 3320 15
70 2010 2320 ...
##  $ lot_area_renov                       : num [1:14620] 5400 4000 6600 42847 4500 ...
##  $ Number of schools nearby             : num [1:14620] 2 2 1 3 1 1 3 3 1 2 ...
##  $ Distance from the airport            : num [1:14620] 58 51 53 76 51 67 72 71 73 53 ...
##  $ Price                                : num [1:14620] 2380000 1400000 1200000 838000 805000
790000 785000 750000 750000 698000 ...
```

Preprocess: combine 2016 and 2017 data as one table and check

data integrity

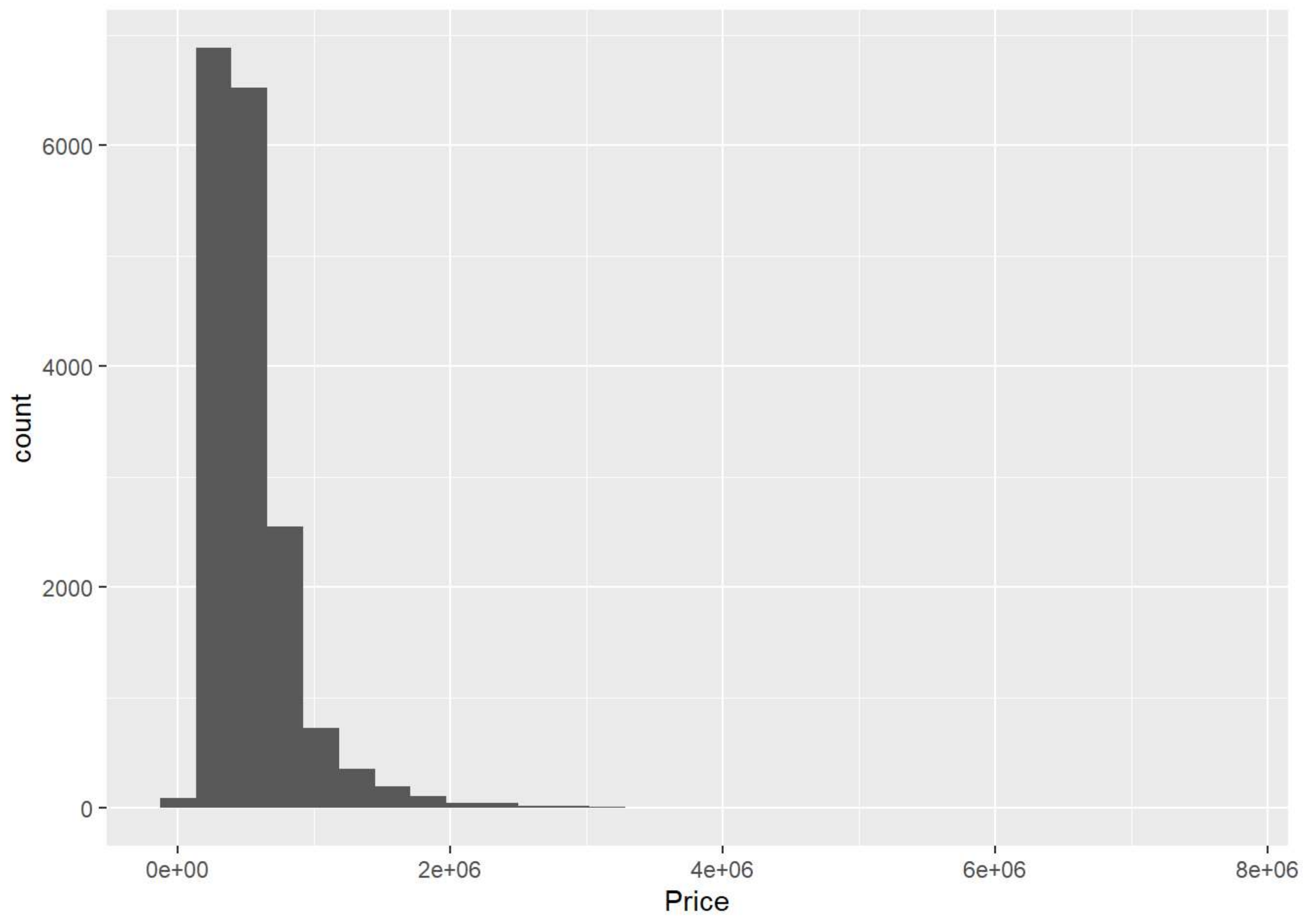
```r
# Combine data using bind_rows
full_df <- df2016 %>%
  bind_rows(df2017)

# check data integrity by searching missing values is an important step for data integrity.
data_inegrity <- function(dataset){
  percent_existData <- dataset %>%
                      complete.cases() %>%
                      mean()*100
  nrowoffulldata <<- nrow(dataset)
  ## write the result
  cat(paste("Original data row:", nrowoffulldata, "\n"))
  cat(paste("Percent exist data:", as.character(percent_existData), "% \n"))
  if (percent_existData == 100){
  print("Percent exist data is 100%, the data is ready")
  } else {
  print("Percent exist data is not 100%, need to clean data")
  }
}

data_inegrity(full_df) # It returns "Percent exist data is 100%, the data is ready".
```

```
## Original data row: 17594
## Percent exist data: 100 %
## [1] "Percent exist data is 100%, the data is ready"
```

Preprocess II: Explore data especially the “dependent Variable (AKA y)”, before analysis

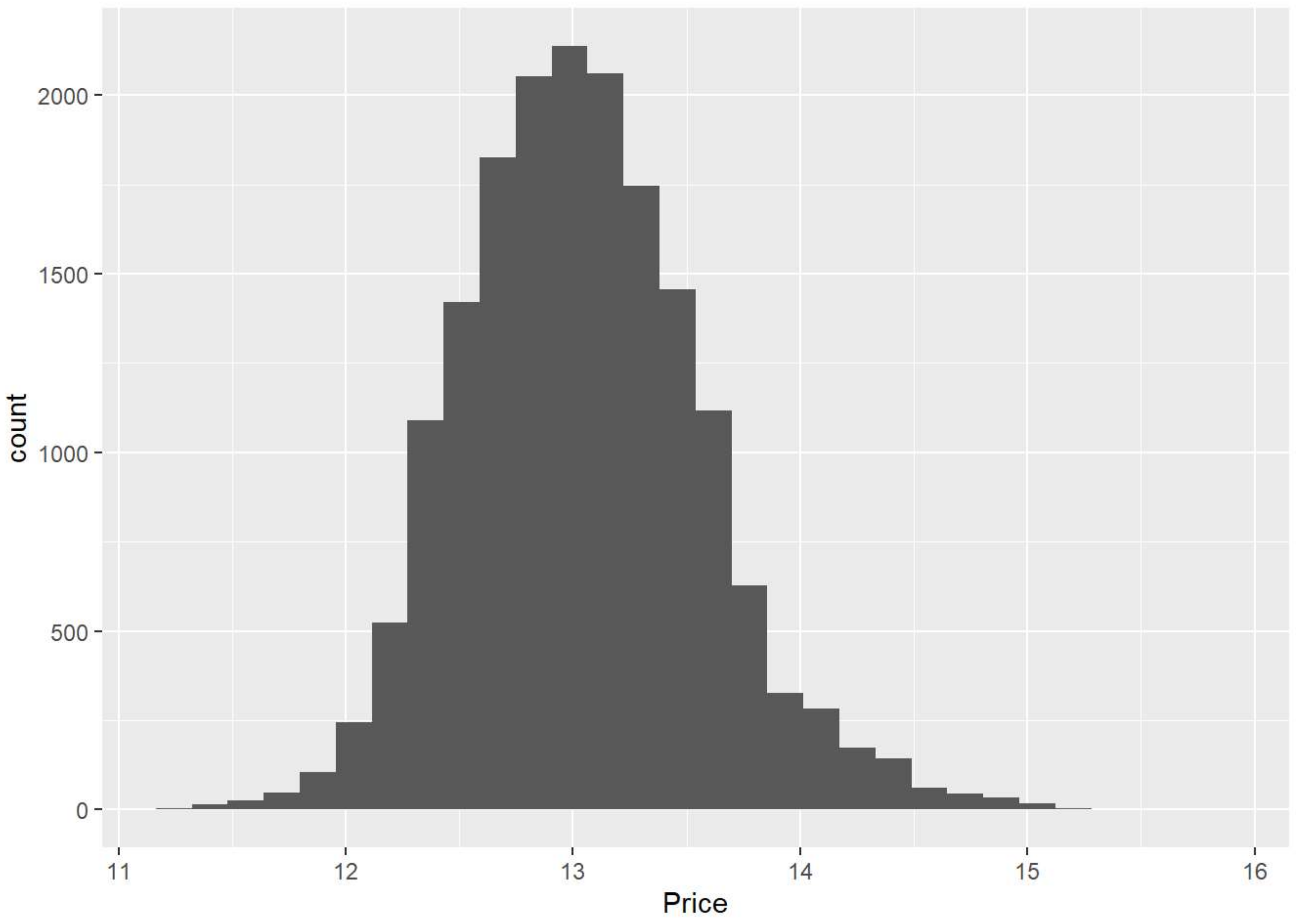
```r
# Use "ggplot" to visualize
ggplot(data = full_df, mapping = aes(x=Price)) +
  geom_histogram() ## It showed right skew
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
# Log transformation may be applied to normalize it.
full_df$Price <- log(full_df$Price)

# Use "ggplot" to visualize again
ggplot(data = full_df, mapping = aes(x=Price)) +
  geom_histogram() ## It showed normal distribution
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
## Note that after analysis does forget to take log out!!!
```

Preprocess III: Explore data to select “related independent variables”

There are many factor that can lead to increase the price such as number of bedrooms, number of bathrooms, living area, Built Year, and Number of schools nearby. These related independent variables have been considered to predict the India house price.

```
select_df <- full_df %>%
  select('number of bedrooms', 'number of bathrooms','living area', 'Built Year', 'Number of sch
ools nearby', 'Price')
```

Split data into train_data and test_data

```
# Then, function below is used to split data into train_data and test_data
## Seed is set to keep the random sampling
## function need two input 1) Dataset 2) sample size by percent (from 0 - 1)
split_data <- function(availableData, percent_sample){
  set.seed(38)
  id <- sample(nrowoffulldata, size = percent_sample*nrowoffulldata)
  train_data <<- availableData[id, ]
  test_data <<- availableData[-id, ]
}

# call split_data(cleanData, 0.7) function to use cleanData that split 70% to train and 30% to t
est set
split_data(select_df, 0.7)

# Just to see split data
(train_data); (test_data)
```

```
## # A tibble: 12,315 × 6
##    `number of bedrooms` `number of bathrooms` `living area` `Built Year`
##                   <dbl>                 <dbl>         <dbl>        <dbl>
##  1                    3                  1             1430         1919
##  2                    3                  1.75          1100         1967
##  3                    4                  1.5           1410         1966
##  4                    4                  3             1850         1991
##  5                    3                  2.5           1950         1990
##  6                    3                  1.75          1750         1976
##  7                    3                  1.5           1270         2007
##  8                    4                  1.75          2350         1961
##  9                    5                  5.25          8010         1999
## 10                    2                  2             1100         1912
## # ℹ 12,305 more rows
## # ℹ 2 more variables: `Number of schools nearby` <dbl>, Price <dbl>
```

```
## # A tibble: 5,279 × 6
##    `number of bedrooms` `number of bathrooms` `living area` `Built Year`
##                   <dbl>                 <dbl>         <dbl>        <dbl>
##  1                    5                  2.5           3650         1921
##  2                    4                  2.5           3310         2001
##  3                    3                  2             2710         1929
##  4                    4                  2.25          2200         1920
##  5                    4                  2.75          2710         2000
##  6                    4                  2.5           2820         2014
##  7                    3                  1.75          2360         1948
##  8                    3                  1.75          2330         1980
##  9                    4                  1.75          1600         1959
## 10                    3                  1.75          1710         1948
## # ℹ 5,269 more rows
## # ℹ 2 more variables: `Number of schools nearby` <dbl>, Price <dbl>
```

Resampling techniques

Resampling techniques are statistical methods used to generate new data points in a dataset by randomly picking data points from the existing dataset. They help in creating new synthetic datasets for training machine learning models and to estimate the properties of a dataset when the dataset is unknown, difficult to estimate, or when the sample size of the dataset is small.

Common methods of resampling are leave one out cross-validation, K-fold cross-validation and bootstrap.

```
# create train control object
# for bootstrap
ctrl_boot <- trainControl(
  method = "boot",   #bootstrap
  number = 10,
  verboseIter = TRUE #print log, when done each iteration
)
# for leave one out cross-validation
ctrl_loocv <- trainControl( # this is not suitable for large data.
  method = "LOOCV",   #leave one out cross-validation
  verboseIter = TRUE #print log, when done each iteration
)
# for K-fold cross-validation
ctrl_cv <- trainControl(
  method = "cv",      #K-fold cross-validation
  number = 5,         #Number of fold, normally we used 5 or 10
  verboseIter = TRUE #print log, when done each iteration
)
```

Train Models (With resampling techniques)

There are many model that can be used to train data. There are two main group (1) Regression and (2) Classification. In this document, the price is numeric value, so regression models will be used.

Test Models (AKA prediction)

Evaluate Models

```
## [1] "model_lm_boot"
```

```
## Mean absolute error of this model is  0.29
```

```
## Root mean square error of this model is  0.36
```

```
## r-square of this model is  0.53
```

```
## [1] "model_lm_cv"
## Mean absolute error of this model is  0.29
```

```
## Root mean square error of this model is  0.36
```

```
## r-square of this model is  0.53
```

```
## [1] "model_rf_wt_cv"
```

```
## Mean absolute error of this model is  0.28
```

```
## Root mean square error of this model is  0.35
```

```
## r-square of this model is  0.55
```

```
## [1] "model_knn_wt_cv"
```

```
## Mean absolute error of this model is  0.3
```

```
## Root mean square error of this model is  0.37
```

```
## r-square of this model is  0.5
```

Compare Multiple Models

```
## 
## Call:
## summary.resamples(object = result)
## 
## Models: liner, randomForest, knn 
## Number of resamples: 5 
## 
## MAE 
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## liner        0.2850235 0.2870264 0.2877657 0.2890170 0.2925188 0.2927505    0
## randomForest 0.2719767 0.2744725 0.2744756 0.2766836 0.2763763 0.2861170    0
## knn          0.2929949 0.2951761 0.2959638 0.2958684 0.2967925 0.2984147    0
## 
## RMSE 
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## liner        0.3557553 0.3571822 0.3611317 0.3602331 0.3615773 0.3655188    0
## randomForest 0.3442604 0.3466425 0.3477281 0.3496724 0.3478971 0.3618337    0
## knn          0.3655600 0.3691793 0.3712648 0.3702882 0.3714094 0.3740272    0
## 
## Rsquared 
##                   Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## liner        0.5278767 0.5298743 0.5301345 0.5338544 0.5346300 0.5467566    0
## randomForest 0.5528393 0.5604457 0.5609402 0.5612136 0.5651074 0.5667353    0
## knn          0.4855783 0.5040071 0.5063267 0.5093214 0.5196092 0.5310855    0
```

Summary

Model ‘randomForest’ is the most usable model to predict India house price between 2016 and 2017 data with the highest Rsquared.

Note

If used model to predict, don’t forget to take LOG out.