stepwise regression process to find best var

Steven

2024-02-23

Data source and original descriptions

data preparation and cleaning

```
# Load necessary libraries
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyve
rse 2.0.0 ──
## ✔ dplyr     1.1.1     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.4     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_co
nflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to for
ce all conflicts to become errors

if (!requireNamespace("caret", quietly = TRUE)) {
  install.packages("caret")
}

library(caret)

## 载入需要的程辑包：lattice
##
## 载入程辑包：'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift

# Load the data
data <- read_csv(file.choose())  # Open and Load file

## Warning: One or more parsing issues, call `problems()` on your data
frame for details,
## e.g.:
```

```
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 34857 Columns: 21
## ── Column specification ────────────────────────────────────────────
────────
## Delimiter: ","
## chr  (8): Suburb, Address, Type, Method, SellerG, Date, CouncilArea,
 Regionname
## dbl (13): Rooms, Price, Distance, Postcode, Bedroom2, Bathroom, Car,
 Landsiz...
##
## ℹ Use `spec()` to retrieve the full column specification for this da
ta.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

```
# Correct column names
names(data)[names(data) == "Lattitude"] <- "Latitude"
names(data)[names(data) == "Longtitude"] <- "Longitude"

# Remove unnecessary columns using dplyr's select function
data_clean <- data %>%
  dplyr::select(Suburb, Rooms, Type, Price, Distance, Bedroom2, Bathroo
m, Car, Landsize, BuildingArea, YearBuilt, CouncilArea, Latitude, Longi
tude, Propertycount, Date)

# Convert 'Date' to date type
data_clean$Date <- as.Date(data_clean$Date, format = "%d/%m/%Y")

# Calculate 'YearsAfterBuilt'
data_clean$YearsAfterBuilt <- as.numeric(format(data_clean$Date, "%Y"))
 - data_clean$YearBuilt

# Calculate PricePerBuildingArea
data_clean$PricePerBuildingArea <- data_clean$Price / data_clean$Buildi
ngArea

# Remove rows with missing values
data_clean <- na.omit(data_clean)

# Drop the "Price", "Longitude", "Latitude", "YearBuilt" and columns fr
om the dataset
data_clean <- subset(data_clean, select = -c(Price, Longitude, Latitude
, YearBuilt))


# Convert categorical variables to factors
cat_vars <- c("Suburb", "Type", "CouncilArea")  # Add categorical varia
```

```
bles here
data_clean[cat_vars] <- lapply(data_clean[cat_vars], as.factor)

# Convert non-categorical variables to numeric
non_cat_vars <- setdiff(names(data_clean), c(cat_vars, "PricePerBuildin
gArea"))
data_clean[non_cat_vars] <- lapply(data_clean[non_cat_vars], as.numeric
)

# Standardize non-categorical variables
data_clean[non_cat_vars] <- scale(data_clean[non_cat_vars])

# Separate predictors and target variable
predictors <- setdiff(names(data_clean), "PricePerBuildingArea")

# Split data into training and testing sets
set.seed(123)
indexes <- createDataPartition(data_clean$PricePerBuildingArea, p = 0.8
, list = FALSE)
train_data <- data_clean[indexes, ]
test_data <- data_clean[-indexes, ]
```

Model training and AIC process

```
# Remove rows with NA, NaN, or Inf values in the target variable
train_data <- train_data[!is.na(train_data$PricePerBuildingArea) & !is.
nan(train_data$PricePerBuildingArea) & !is.infinite(train_data$PricePer
BuildingArea), ]

# Train stepwise regression model
model <- step(lm(PricePerBuildingArea ~ ., data = train_data[, c(predic
tors, "PricePerBuildingArea")]), direction = "backward")

## Start:  AIC=146189.8
## PricePerBuildingArea ~ Suburb + Rooms + Type + Distance + Bedroom2 +

##     Bathroom + Car + Landsize + BuildingArea + CouncilArea +
##     Propertycount + Date + YearsAfterBuilt
##
##
## Step:  AIC=146189.8
## PricePerBuildingArea ~ Suburb + Rooms + Type + Distance + Bedroom2 +

##     Bathroom + Car + Landsize + BuildingArea + CouncilArea +
##     Date + YearsAfterBuilt
##
##
## Step:  AIC=146189.8
## PricePerBuildingArea ~ Suburb + Rooms + Type + Distance + Bedroom2 +
```

```
##     Bathroom + Car + Landsize + BuildingArea + Date + YearsAfterBuil
t
##
##                    Df  Sum of Sq        RSS    AIC
## - Suburb          305 2.1970e+11 6.3092e+12 145831
## - Landsize          1 4.9944e+07 6.0895e+12 146188
## - Bathroom          1 7.9612e+07 6.0896e+12 146188
## - Date              1 5.0891e+08 6.0900e+12 146188
## - YearsAfterBuilt   1 1.0302e+09 6.0905e+12 146189
## <none>                           6.0895e+12 146190
## - Car               1 2.9181e+09 6.0924e+12 146191
## - Rooms             1 3.3703e+09 6.0928e+12 146192
## - Distance          1 4.5451e+09 6.0940e+12 146193
## - Bedroom2          1 1.0378e+10 6.0999e+12 146200
## - Type              2 4.0714e+10 6.1302e+12 146233
## - BuildingArea      1 1.1391e+11 6.2034e+12 146319
##
## Step:  AIC=145830.6
## PricePerBuildingArea ~ Rooms + Type + Distance + Bedroom2 + Bathroom
 +
##     Car + Landsize + BuildingArea + Date + YearsAfterBuilt
##
##                   Df  Sum of Sq        RSS    AIC
## - Date             1 1.5248e+07 6.3092e+12 145829
## - Landsize         1 7.3854e+07 6.3092e+12 145829
## <none>                          6.3092e+12 145831
## - Bathroom         1 2.5749e+09 6.3117e+12 145831
## - Car              1 3.0252e+09 6.3122e+12 145832
## - Rooms            1 4.0337e+09 6.3132e+12 145833
## - Bedroom2         1 9.7407e+09 6.3189e+12 145839
## - YearsAfterBuilt  1 1.0803e+10 6.3200e+12 145841
## - Distance         1 1.3718e+10 6.3229e+12 145844
## - Type             2 4.0314e+10 6.3495e+12 145872
## - BuildingArea     1 9.5400e+10 6.4046e+12 145935
##
## Step:  AIC=145828.6
## PricePerBuildingArea ~ Rooms + Type + Distance + Bedroom2 + Bathroom
 +
##     Car + Landsize + BuildingArea + YearsAfterBuilt
##
##                   Df  Sum of Sq        RSS    AIC
## - Landsize         1 7.6641e+07 6.3093e+12 145827
## <none>                          6.3092e+12 145829
## - Bathroom         1 2.5787e+09 6.3118e+12 145829
## - Car              1 3.0414e+09 6.3122e+12 145830
## - Rooms            1 4.1124e+09 6.3133e+12 145831
## - Bedroom2         1 9.9601e+09 6.3191e+12 145838
## - YearsAfterBuilt  1 1.0788e+10 6.3200e+12 145839
## - Distance         1 1.4390e+10 6.3236e+12 145843
```

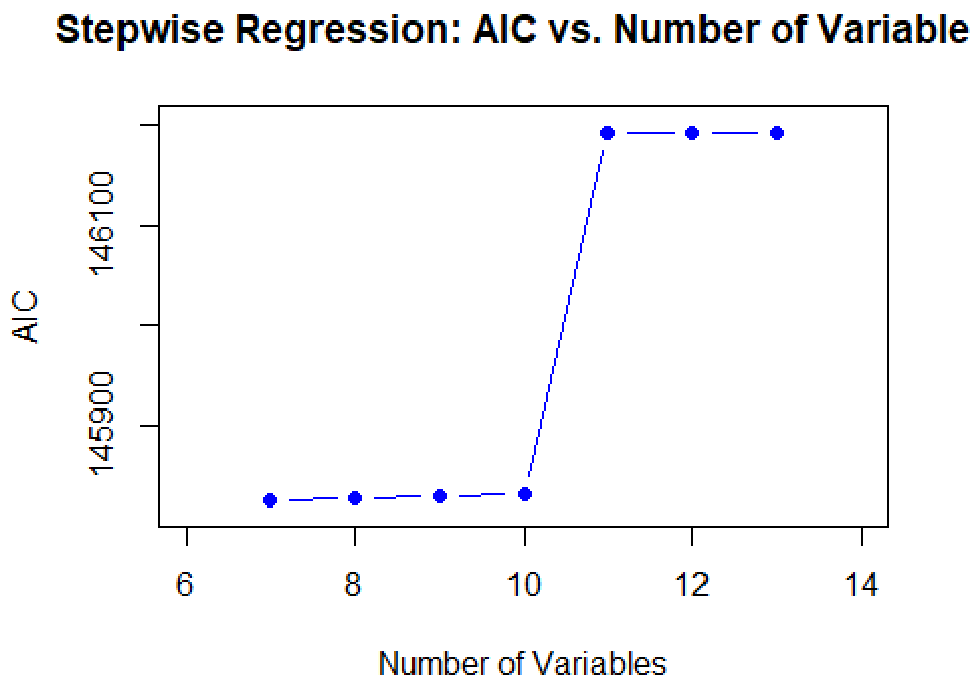
```
## - Type              2 4.0321e+10 6.3495e+12 145870
## - BuildingArea      1 9.5635e+10 6.4048e+12 145933
##
## Step:  AIC=145826.7
## PricePerBuildingArea ~ Rooms + Type + Distance + Bedroom2 + Bathroom
 +
##     Car + BuildingArea + YearsAfterBuilt
##
##                    Df  Sum of Sq        RSS    AIC
## <none>                           6.3093e+12 145827
## - Bathroom          1 2.5938e+09 6.3119e+12 145828
## - Car               1 3.1298e+09 6.3124e+12 145828
## - Rooms             1 4.1043e+09 6.3134e+12 145829
## - Bedroom2          1 9.9565e+09 6.3192e+12 145836
## - YearsAfterBuilt   1 1.0813e+10 6.3201e+12 145837
## - Distance          1 1.4324e+10 6.3236e+12 145841
## - Type              2 4.0248e+10 6.3495e+12 145868
## - BuildingArea      1 9.5562e+10 6.4048e+12 145931
```

```
# Make predictions on test data
predictions <- predict(model, newdata = test_data)

# Evaluate the model
rmse <- sqrt(mean((predictions - test_data$PricePerBuildingArea)^2))
print(paste("RMSE: ", rmse))
```

```
## [1] "RMSE:  Inf"
```

Displaying AIC value graph

```
# AIC values from the stepwise regression process
aic_values <- c(146191.6, 146191.6, 146191.6, 145831.9, 145830, 145828.
6, 145826.6)

# Number of variables in each step
num_variables <- c(13, 12, 11, 10, 9, 8, 7)

# Plotting the AIC values against the number of variables
plot(num_variables, aic_values, type = "b",
     xlab = "Number of Variables",
     ylab = "AIC",
     main = "Stepwise Regression: AIC vs. Number of Variables",
     xlim = c(min(num_variables) - 1, max(num_variables) + 1),
     ylim = c(min(aic_values) - 10, max(aic_values) + 10),
     col = "blue",
     pch = 19)
```



final model summary

```
# Train the final model based on the selected predictors
final_model <- lm(PricePerBuildingArea ~ Rooms + Type + Distance + Bedr
oom2 +
                    Bathroom + Car + BuildingArea + YearsAfterBuilt, dat
a = train_data)

# Print the summary of the final model
summary(final_model)

##
## Call:
## lm(formula = PricePerBuildingArea ~ Rooms + Type + Distance +
##     Bedroom2 + Bathroom + Car + BuildingArea + YearsAfterBuilt,
##     data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -31541   -3928   -1932     471 1034369
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         8685.3      451.6  19.233  < 2e-16 ***
## Rooms              -3085.5     1439.3  -2.144 0.032083 *
## Typet               9392.8     1455.1   6.455 1.15e-10 ***
```

```
## Typeu                461.1     1229.1   0.375 0.707573    
## Distance           -1655.3      413.3  -4.005 6.27e-05 ***
## Bedroom2            4794.0     1435.7   3.339 0.000845 ***
## Bathroom             851.7      499.7   1.704 0.088379 .  
## Car                  745.5      398.2   1.872 0.061235 .  
## BuildingArea       -4803.9      464.4 -10.344  < 2e-16 ***
## YearsAfterBuilt     1495.5      429.8   3.480 0.000505 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 29880 on 7065 degrees of freedom
## Multiple R-squared:  0.02958,    Adjusted R-squared:  0.02834 
## F-statistic: 23.93 on 9 and 7065 DF,  p-value: < 2.2e-16
```

variable correlation check

```
if (!requireNamespace("GGally", quietly = TRUE)) {
    install.packages("GGally")
}
```

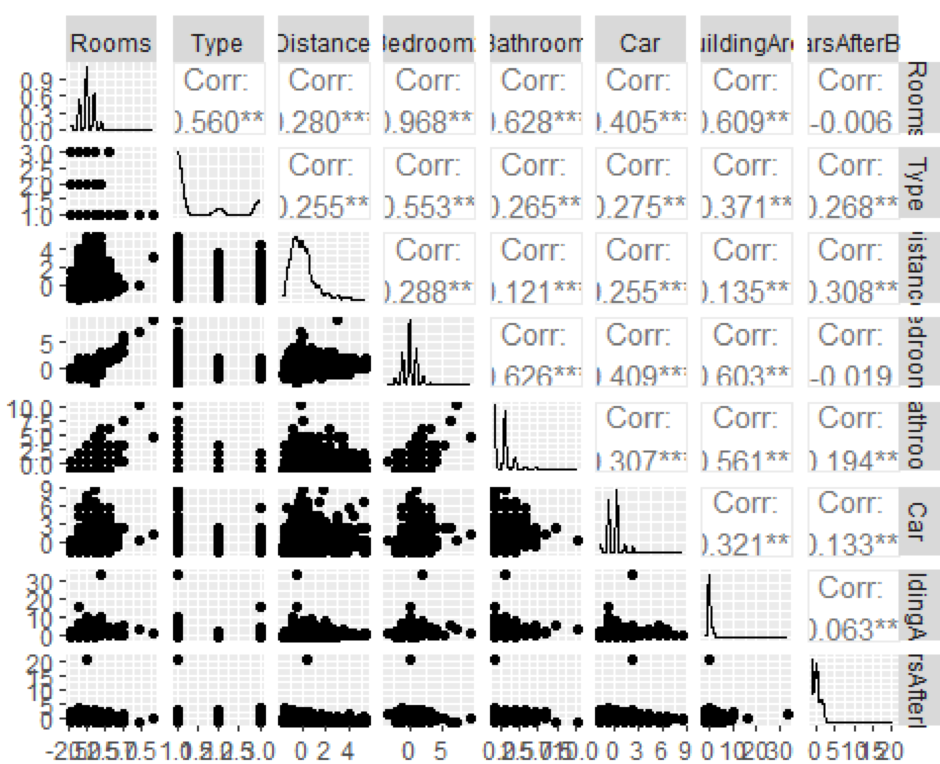
```
## Registered S3 method overwritten by 'GGally':
##   method from   
##   +.gg   ggplot2
```

```
# Load the GGally package
library(GGally)

# Select predictors for correlation analysis
cor_data <- train_data[, c("Rooms", "Type", "Distance", "Bedroom2", "Ba
throom", "Car", "BuildingArea", "YearsAfterBuilt")]

# Convert non-numeric columns to numeric
cor_data_numeric <- as.data.frame(sapply(cor_data, as.numeric))

# Compute pairwise correlations
correlation_matrix <- cor(cor_data_numeric)

# Print pairwise correlations
print(correlation_matrix)
```

```
##                        Rooms       Type   Distance    Bedroom2   Bat
hroom
## Rooms            1.000000000 -0.5597444  0.2800834  0.96801199  0.62
76422
## Type            -0.559744386  1.0000000 -0.2545656 -0.55332318 -0.26
52077
## Distance         0.280083407 -0.2545656  1.0000000  0.28843088  0.12
09617
## Bedroom2         0.968011994 -0.5533232  0.2884309  1.00000000  0.62
61735
```

```
## Bathroom         0.627642191 -0.2652077  0.1209617  0.62617354  1.00
00000
## Car              0.405160513 -0.2754376  0.2553177  0.40921543  0.30
73153
## BuildingArea     0.609242417 -0.3708441  0.1348149  0.60297412  0.56
13127
## YearsAfterBuilt -0.006047909 -0.2679670 -0.3078093 -0.01937879 -0.19
36693
##                        Car BuildingArea YearsAfterBuilt
## Rooms            0.4051605   0.60924242    -0.006047909
## Type            -0.2754376  -0.37084408    -0.267966986
## Distance         0.2553177   0.13481485    -0.307809292
## Bedroom2         0.4092154   0.60297412    -0.019378790
## Bathroom         0.3073153   0.56131273    -0.193669268
## Car              1.0000000   0.32071519    -0.133360609
## BuildingArea     0.3207152   1.00000000    -0.063251087
## YearsAfterBuilt -0.1333606  -0.06325109     1.000000000
```

```r
# Create a histogram grid for visualization
ggpairs(cor_data_numeric)
```



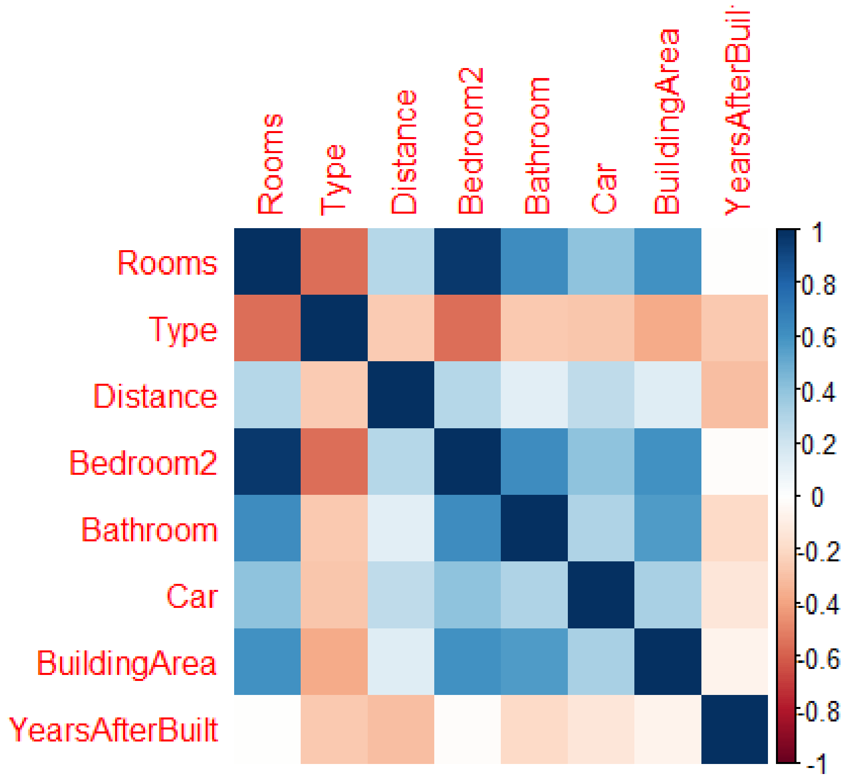
correlation graph display

```r
# Load necessary libraries
library(corrplot)

## corrplot 0.92 loaded
```

```r
# Convert non-numeric columns to numeric
cor_data_numeric <- as.data.frame(sapply(cor_data, as.numeric))

# Compute pairwise correlations
correlation_matrix <- cor(cor_data_numeric)

# Print pairwise correlations
print(correlation_matrix)
```

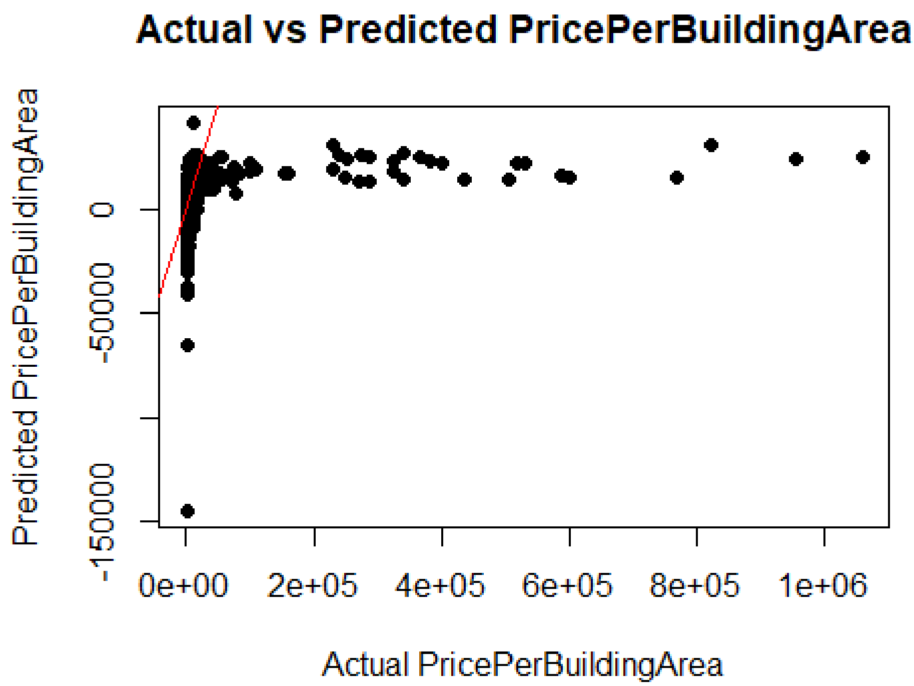
```
##                       Rooms       Type   Distance    Bedroom2   Bat
hroom
## Rooms            1.000000000 -0.5597444  0.2800834  0.96801199  0.62
76422
## Type            -0.559744386  1.0000000 -0.2545656 -0.55332318 -0.26
52077
## Distance         0.280083407 -0.2545656  1.0000000  0.28843088  0.12
09617
## Bedroom2         0.968011994 -0.5533232  0.2884309  1.00000000  0.62
61735
## Bathroom         0.627642191 -0.2652077  0.1209617  0.62617354  1.00
00000
## Car              0.405160513 -0.2754376  0.2553177  0.40921543  0.30
73153
## BuildingArea     0.609242417 -0.3708441  0.1348149  0.60297412  0.56
13127
## YearsAfterBuilt -0.006047909 -0.2679670 -0.3078093 -0.01937879 -0.19
36693
##                       Car BuildingArea YearsAfterBuilt
## Rooms            0.4051605   0.60924242    -0.006047909
## Type            -0.2754376  -0.37084408    -0.267966986
## Distance         0.2553177   0.13481485    -0.307809292
## Bedroom2         0.4092154   0.60297412    -0.019378790
## Bathroom         0.3073153   0.56131273    -0.193669268
## Car              1.0000000   0.32071519    -0.133360609
## BuildingArea     0.3207152   1.00000000    -0.063251087
## YearsAfterBuilt -0.1333606  -0.06325109     1.000000000
```

```r
# Create a correlation plot with color
corrplot(correlation_matrix, method = "color")
```



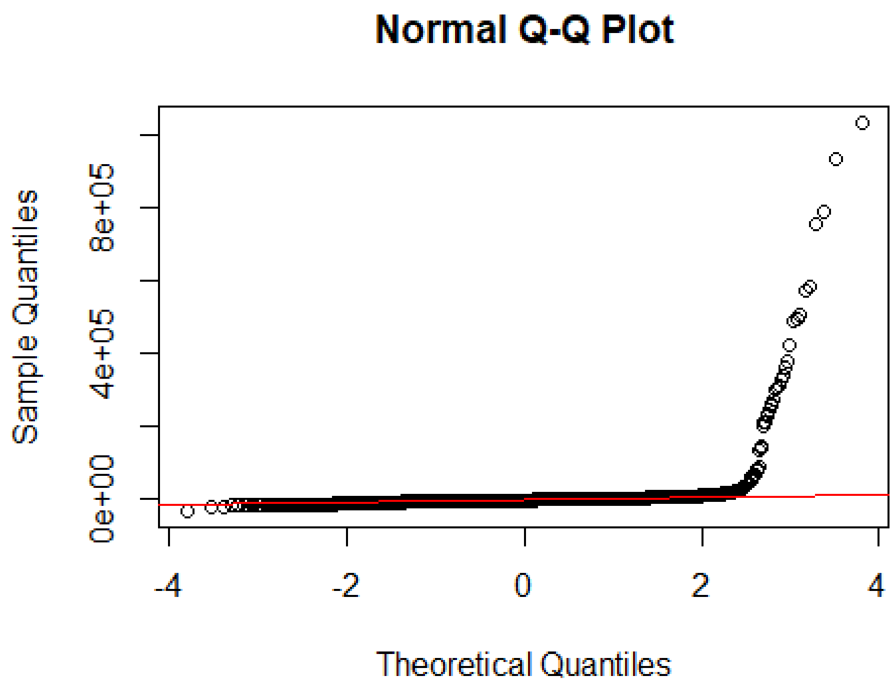
Actual vs predicted graph

```
# Calculate predicted values using the model
predicted_values <- predict(model, newdata = train_data)

# Ensure that only rows with no missing values in the relevant columns are used
valid_rows <- complete.cases(train_data[c("PricePerBuildingArea", "Rooms", "Type", "Distance", "Bedroom2", "Bathroom", "Car", "BuildingArea", "YearsAfterBuilt")])
actual_values <- train_data$PricePerBuildingArea[valid_rows]
predicted_values <- predicted_values[valid_rows]

# Now plot actual vs predicted values
plot(actual_values, predicted_values,
     main = "Actual vs Predicted PricePerBuildingArea",
     xlab = "Actual PricePerBuildingArea",
     ylab = "Predicted PricePerBuildingArea",
     pch = 19)  # pch = 19 for solid circles

# Add a line of perfect fit
abline(a = 0, b = 1, col = "red")
```

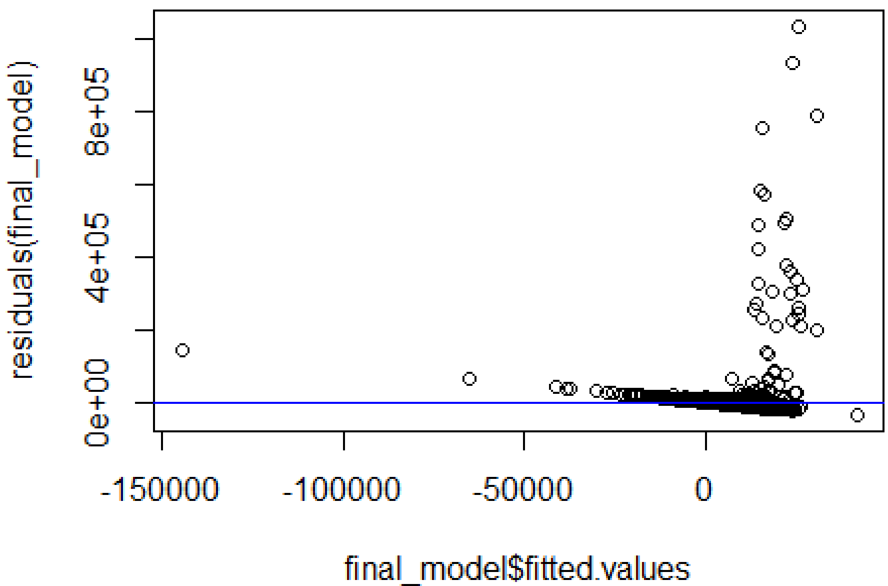


qq and residual plots

```
# QQ plot for the first model
qqnorm(residuals(final_model))
qqline(residuals(final_model), col = "red")
```




```
# Residual plot for the first model
plot(final_model$fitted.values, residuals(final_model))
abline(h = 0, col = "blue")
```



Comment of results # Comment of business implications