classification, regression with outlier exclusion

Steven

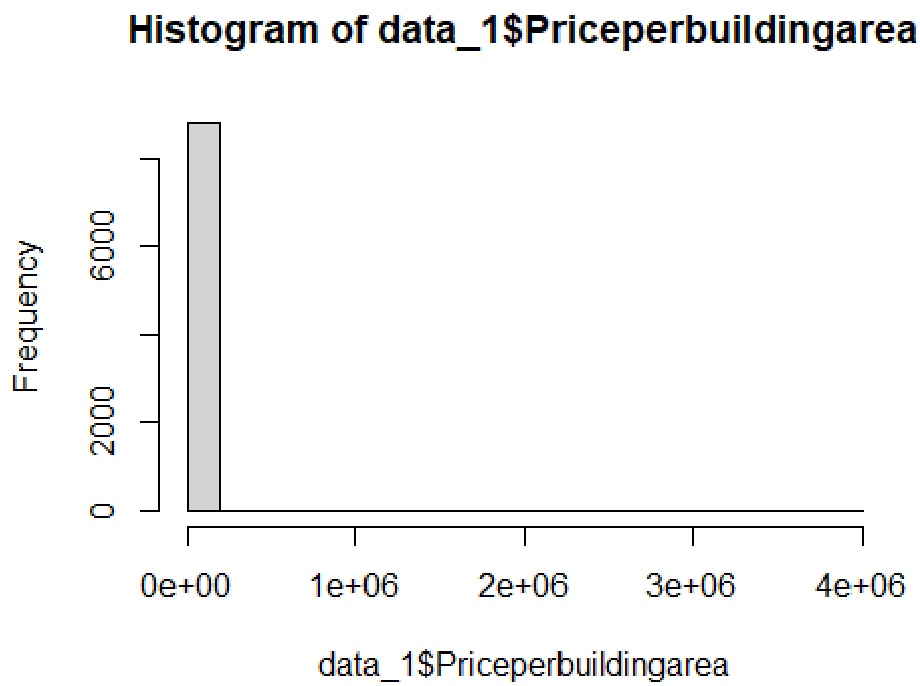
2024-02-16

Data source and original descriptions

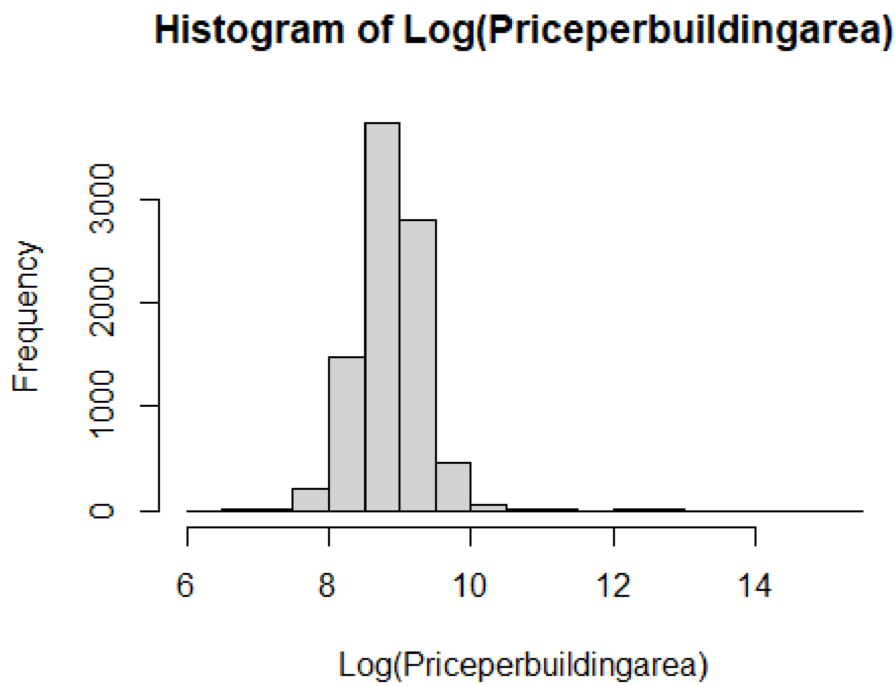
Step 1: Data Cleaning and Preparation

```
# Load necessary libraries
library(dplyr)

##
## 载入程辑包：'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(readr)

# Step 1: Load the dataset

data_1 <- read_csv(file.choose())  # select the file interactively

## 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
data.
```

```
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet
this message.

# Step 2: Correct typos in column names
names(data_1)[names(data_1) == "Lattitude"] <- "Latitude"
names(data_1)[names(data_1) == "Longtitude"] <- "Longitude"

# Step 3: Calculate features
# Convert 'Date' from character to Date type.

data_1$Date <- as.Date(data_1$Date, format = "%d/%m/%Y")

# Extract the year from the 'Date' column
data_1$YearOfSale <- as.numeric(format(data_1$Date, "%Y"))

# Calculate 'YearsSinceBuilt' and 'Priceperbuildingarea'
data_1$YearsSinceBuilt <- data_1$YearOfSale - data_1$YearBuilt
data_1$Priceperbuildingarea <- with(data_1, Price / BuildingArea)

# Step 4: Clean the data (Eliminate `NA` and `Inf` values)
data_1 <- na.omit(data_1) # Remove rows with NA values

#Identify numeric columns
numeric_cols <- sapply(data_1, is.numeric)

# Apply is.infinite only to numeric columns and then reduce to rows
with any Inf values
rows_with_inf <- apply(data_1[, numeric_cols], 1, function(x)
any(is.infinite(x)))

# Remove rows with Inf values
data_1 <- data_1[!rows_with_inf, ]

# hist 1
hist(data_1$Priceperbuildingarea)
```




```
# Step 5: Apply log transformation to 'Priceperbuildingarea'
data_1 <- data_1 %>%
  mutate(LogPriceperbuildingarea = log(Priceperbuildingarea + 1)) #
Adding 1 to avoid log(0)

# hist 2 using the transformed data
hist(data_1$LogPriceperbuildingarea, main = "Histogram of
Log(Priceperbuildingarea)", xlab = "Log(Priceperbuildingarea)")
```




```r
# Using names() function
column_names <- names(data_1)
print(column_names)
```

```
##  [1] "Suburb"                    "Address"
##  [3] "Rooms"                     "Type"
##  [5] "Price"                     "Method"
##  [7] "SellerG"                   "Date"
##  [9] "Distance"                  "Postcode"
## [11] "Bedroom2"                  "Bathroom"
## [13] "Car"                       "Landsize"
## [15] "BuildingArea"              "YearBuilt"
## [17] "CouncilArea"               "Latitude"
## [19] "Longitude"                 "Regionname"
## [21] "Propertycount"             "YearOfSale"
## [23] "YearsSinceBuilt"           "Priceperbuildingarea"
## [25] "LogPriceperbuildingarea"
```

```r
# Identify numeric columns
numeric_columns <- sapply(data_1, is.numeric)

# Select only numeric columns
selected_data <- data_1[, numeric_columns]

# Calculate correlation matrix
correlation_matrix <- cor(selected_data, use = "pairwise.complete.obs")
```

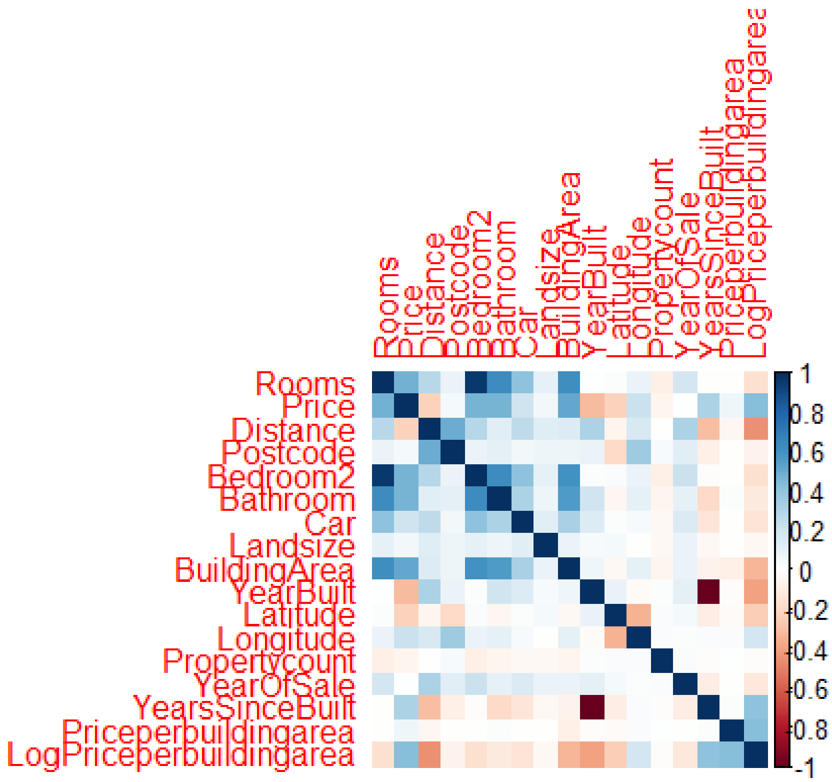
```r
# Print correlation matrix
print(correlation_matrix)
```

```
##                              Rooms        Price    Distance
Postcode
## Rooms                  1.000000000  0.475788014  0.27661114
0.083664315
## Price                  0.475788014  1.000000000 -0.22963983
0.046351970
## Distance               0.276611142 -0.229639827  1.00000000
0.490500587
## Postcode               0.083664315  0.046351970  0.49050059
1.000000000
## Bedroom2               0.964304575  0.461531262  0.28349594
0.086720994
## Bathroom               0.624417153  0.464244561  0.12268190
0.111536273
## Car                    0.402285556  0.209237577  0.25969899
0.055035220
## Landsize               0.101915842  0.058808330  0.13940588
0.070376075
## BuildingArea           0.616344949  0.513429470  0.14005136
0.079538384
## YearBuilt              0.005821426 -0.313353156  0.31296092
0.089175637
## Latitude               0.017253305 -0.223094700 -0.05842611 -
0.197490730
## Longitude              0.082678962  0.211962497  0.16416876
0.358670645
## Propertycount         -0.083545295 -0.059954573 -0.00281781
0.035163212
## YearOfSale             0.188538571  0.001047698  0.31863357
0.126318306
## YearsSinceBuilt       -0.002479590  0.313946247 -0.30786886 -
0.087092985
## Priceperbuildingarea  -0.007444964  0.064399251 -0.04485344 -
0.003500867
## LogPriceperbuildingarea -0.167717250  0.424581159 -0.45739928 -
0.061431297
##                           Bedroom2    Bathroom         Car
Landsize
## Rooms                  0.964304575  0.62441715  0.40228556
0.101915842
## Price                  0.461531262  0.46424456  0.20923758
0.058808330
## Distance               0.283495944  0.12268190  0.25969899
0.139405884
## Postcode               0.086720994  0.11153627  0.05503522
0.070376075
## Bedroom2               1.000000000  0.62685703  0.40645388
```

```
0.101784661
## Bathroom                 0.626857029  1.00000000  0.31117338
0.076769871
## Car                      0.406453875  0.31117338  1.00000000
0.123995235
## Landsize                 0.101784661  0.07676987  0.12399523
1.000000000
## BuildingArea             0.604994169  0.56043192  0.32113364
0.084773959
## YearBuilt                0.015205914  0.19268179  0.14013929
0.037637573
## Latitude                 0.021230920 -0.04290418  0.01622768
0.042897478
## Longitude                0.082319155  0.10980527  0.03509597 -
0.008379047
## Propertycount           -0.082157036 -0.05854791 -0.03061361 -
0.032519305
## YearOfSale               0.210630732  0.10864483  0.15239101
0.084124074
## YearsSinceBuilt         -0.011488450 -0.19110316 -0.13768645 -
0.036210716
## Priceperbuildingarea    -0.002980874  0.01344173 -0.00975971 -
0.007748458
## LogPriceperbuildingarea -0.168785203 -0.11435093 -0.14191911 -
0.037554395
##                         BuildingArea    YearBuilt    Latitude
Longitude
## Rooms                     0.61634495  0.005821426  0.01725331
0.082678962
## Price                     0.51342947 -0.313353156 -0.22309470
0.211962497
## Distance                  0.14005136  0.312960918 -0.05842611
0.164168761
## Postcode                  0.07953838  0.089175637 -0.19749073
0.358670645
## Bedroom2                  0.60499417  0.015205914  0.02123092
0.082319155
## Bathroom                  0.56043192  0.192681793 -0.04290418
0.109805266
## Car                       0.32113364  0.140139288  0.01622768
0.035095966
## Landsize                  0.08477396  0.037637573  0.04289748 -
0.008379047
## BuildingArea              1.00000000  0.063350861 -0.03167111
0.100045394
## YearBuilt                 0.06335086  1.000000000  0.09770358 -
0.026426855
## Latitude                 -0.03167111  0.097703584  1.00000000 -
0.347251372
## Longitude                 0.10004539 -0.026426855 -0.34725137
```

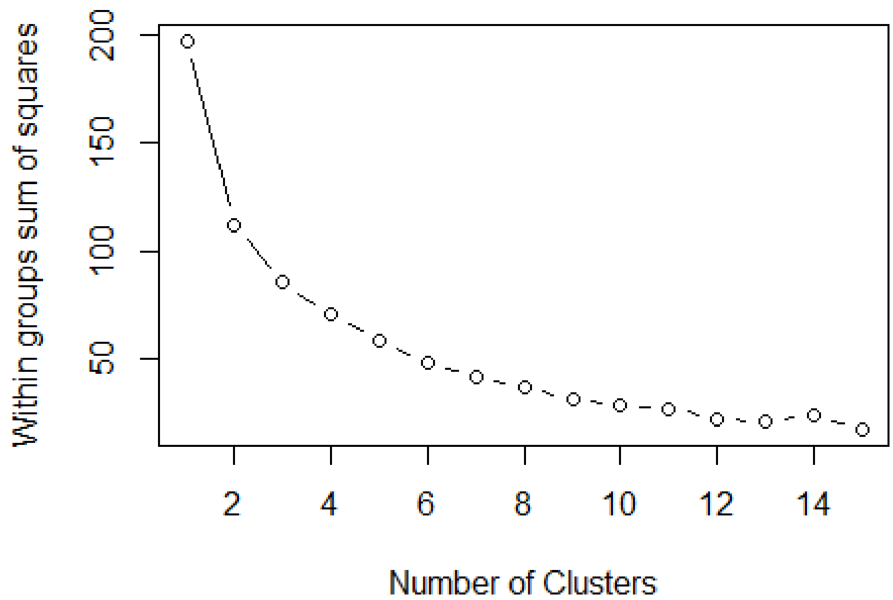
```
1.000000000
## Propertycount            -0.05791083  0.017048600  0.02797651
0.028108628
## YearOfSale                0.08757188  0.111879852  0.04863390
0.018476138
## YearsSinceBuilt          -0.06190984 -0.999843875 -0.09701792
0.026803837
## Priceperbuildingarea     -0.08794680 -0.015589360 -0.02141496
0.025620643
## LogPriceperbuildingarea  -0.33768122 -0.404750293 -0.24126901
0.180911679
##                         Propertycount   YearOfSale YearsSinceBuilt
## Rooms                    -0.083545295  0.188538571     -0.00247959
## Price                    -0.059954573  0.001047698      0.31394625
## Distance                 -0.002817810  0.318633569     -0.30786886
## Postcode                  0.035163212  0.126318306     -0.08709298
## Bedroom2                 -0.082157036  0.210630732     -0.01148845
## Bathroom                 -0.058547912  0.108644828     -0.19110316
## Car                      -0.030613606  0.152391008     -0.13768645
## Landsize                 -0.032519305  0.084124074     -0.03621072
## BuildingArea             -0.057910826  0.087571885     -0.06190984
## YearBuilt                 0.017048600  0.111879852     -0.99984387
## Latitude                  0.027976514  0.048633902     -0.09701792
## Longitude                 0.028108628  0.018476138      0.02680384
## Propertycount             1.000000000  0.020648756     -0.01671269
## YearOfSale                0.020648756  1.000000000     -0.09430341
## YearsSinceBuilt          -0.016712687 -0.094303406      1.00000000
## Priceperbuildingarea      0.004128484 -0.008957342      0.01545866
## LogPriceperbuildingarea  -0.016350379 -0.128436449      0.40320851
##                         Priceperbuildingarea LogPriceperbuildingarea
## Rooms                           -0.007444964             -0.16771725
## Price                            0.064399251              0.42458116
## Distance                        -0.044853444             -0.45739928
## Postcode                        -0.003500867             -0.06143130
## Bedroom2                        -0.002980874             -0.16878520
## Bathroom                         0.013441731             -0.11435093
## Car                             -0.009759710             -0.14191911
## Landsize                        -0.007748458             -0.03755440
## BuildingArea                    -0.087946797             -0.33768122
## YearBuilt                       -0.015589360             -0.40475029
## Latitude                        -0.021414957             -0.24126901
## Longitude                        0.025620643              0.18091168
## Propertycount                    0.004128484             -0.01635038
## YearOfSale                      -0.008957342             -0.12843645
## YearsSinceBuilt                  0.015458664              0.40320851
## Priceperbuildingarea             1.000000000              0.42905488
## LogPriceperbuildingarea          0.429054882              1.00000000

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

```
## corrplot 0.92 loaded

# Calculate the correlation matrix
correlation_matrix <- cor(selected_data, use = "pairwise.complete.obs")

# Create the colored correlation grid
corrplot(correlation_matrix, method = "color")
```

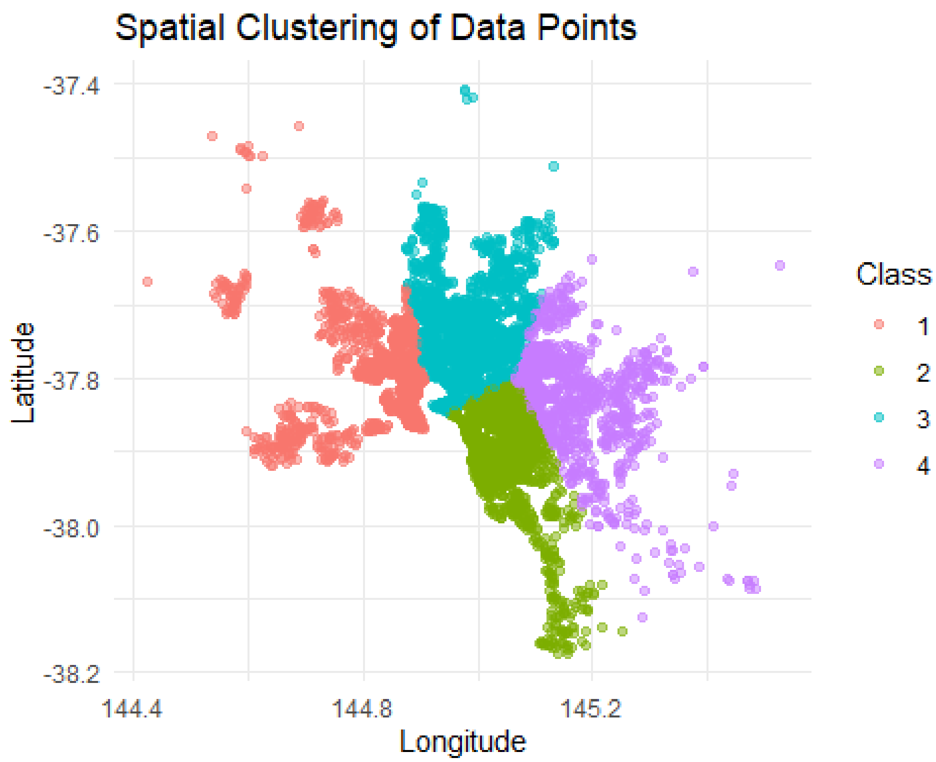


Step 2: Classification with K-Means Clustering:

```
library(dplyr)
library(ggplot2)
library(cluster) # For clustering

# 'data_1' has been loaded and contains 'Longitude' and 'Latitude'
coords <- data_1 %>% select(Longitude, Latitude)

# Determine the optimal number of clusters (optional, for illustration)
# This step can be computationally intensive for large datasets
wss <- (nrow(coords)-1)*sum(apply(coords,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(coords, centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters", ylab="Within
groups sum of squares")
```

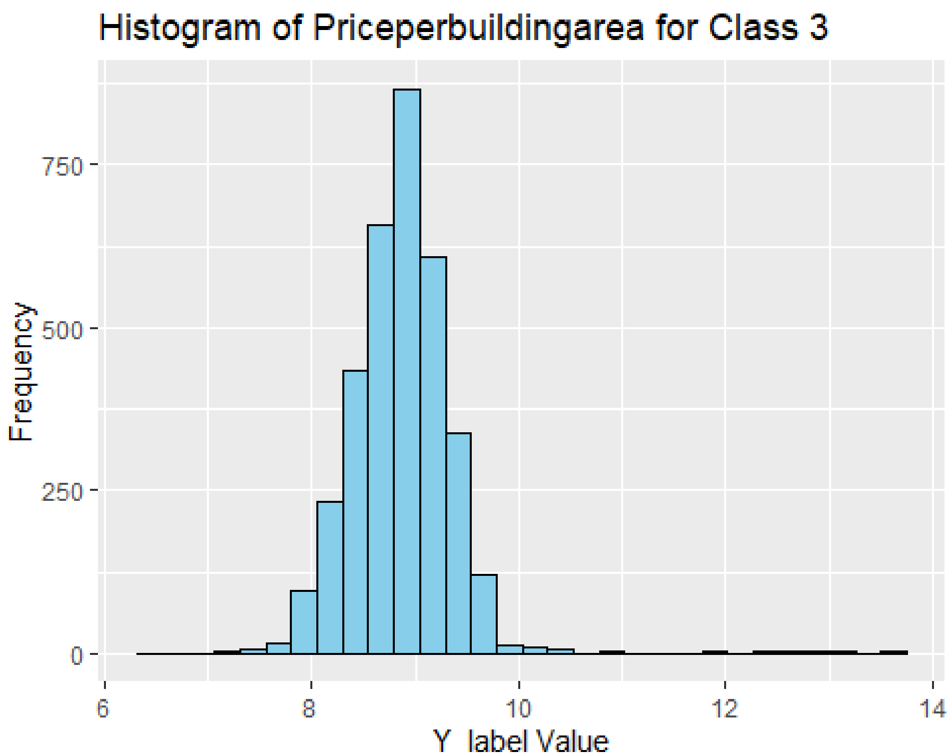


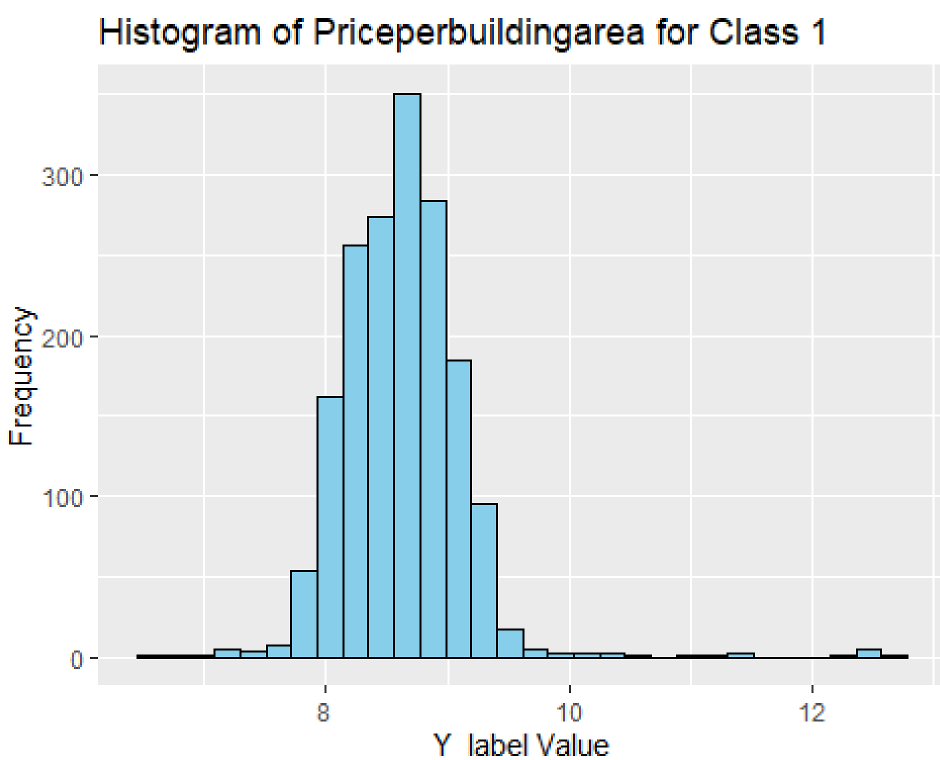

```
# K-Means Clustering
set.seed(123) # For reproducibility
k <- 4 # Choose based on analysis, e.g., using the Elbow Method above
km_res <- kmeans(coords, centers = k)

# Assign class numbers to the original data and factorize
data_1$Class <- km_res$cluster
data_1$Class <- factor(data_1$Class)

# Step 4: Visualize on a Map
library(ggmap)

## ℹ Google's Terms of Service: <https://mapsplatform.google.com>

## ℹ Please cite ggmap if you use it! Use `citation("ggmap")` for
details.

library(ggplot2)

# Basic plot with ggplot2
ggplot(data_1, aes(x = Longitude, y = Latitude, color = factor(Class)))
+
  geom_point(alpha = 0.5) +
  labs(title = "Spatial Clustering of Data Points", color = "Class") +
  theme_minimal()
```

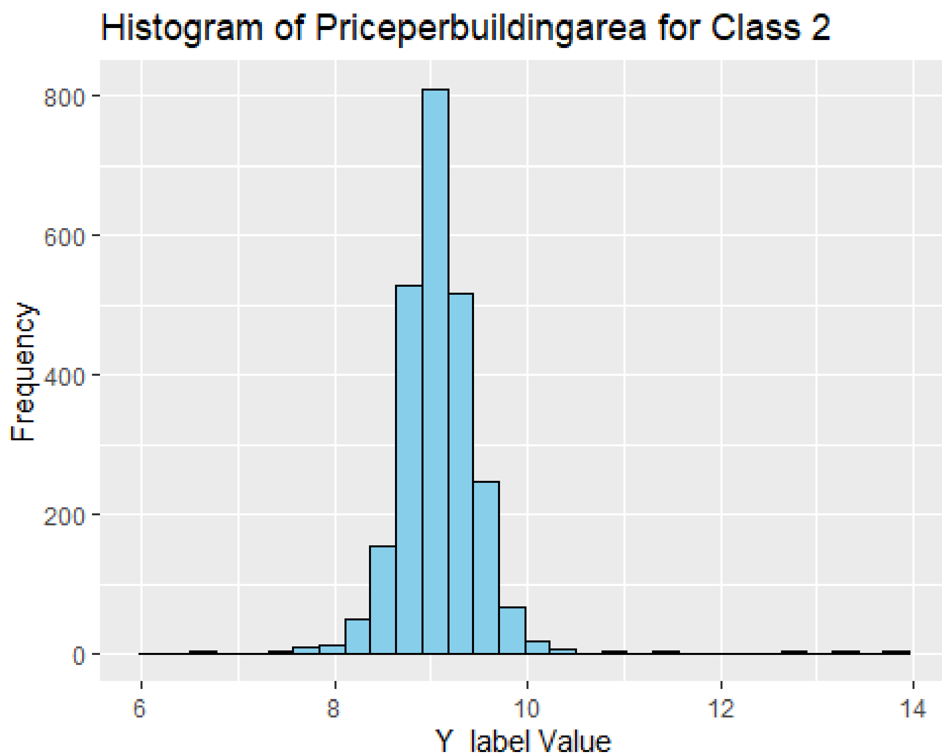


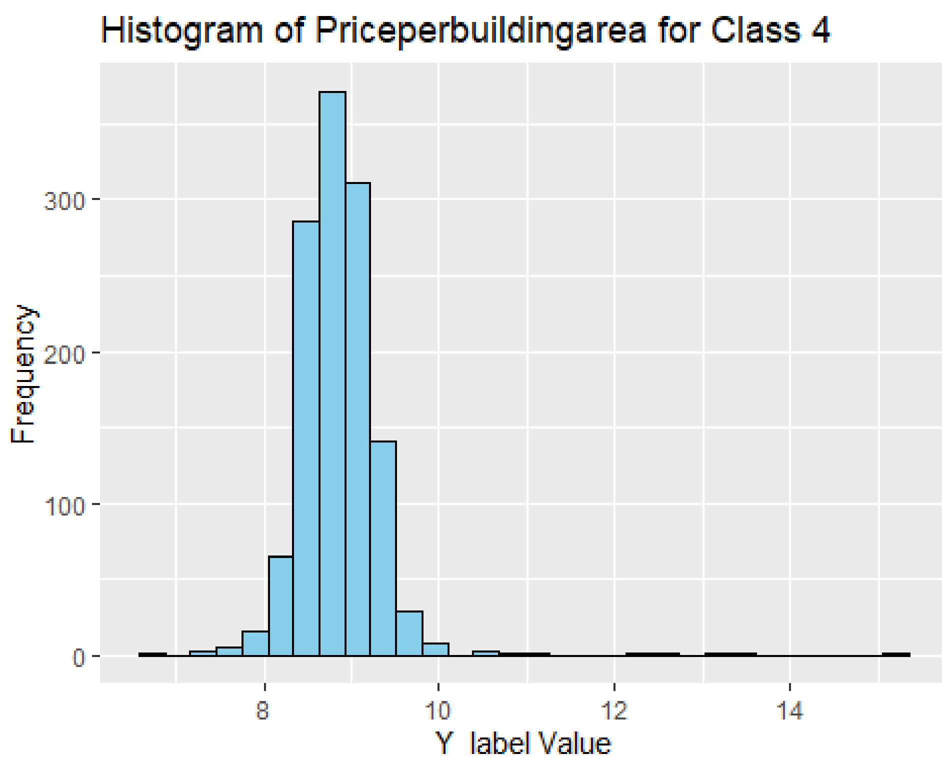
Step 3: visualize histograms:

```
library(ggplot2)
library(dplyr)

# 'data_1' contains 'Y_label' and 'Class' columns
# Loop through each class and plot a histogram
unique_classes <- unique(data_1$Class)

# Create a list to store plots
plot_list <- list()

for(class in unique_classes) {
  plot <- data_1 %>%
    filter(Class == class) %>%
    ggplot(aes(x = LogPriceperbuildingarea)) +
    geom_histogram(bins = 30, fill = "skyblue", color = "black") +
    ggtitle(paste("Histogram of Priceperbuildingarea for Class",
class)) +
    xlab("Y_label Value") +
    ylab("Frequency")

  print(plot) # Display the plot
  plot_list[[as.character(class)]] <- plot # Store the plot in a list
}
```


Histogram of Priceperbuildingarea for Class 3
Frequency
750
500
250
0
6
8
10
12
14
Y_label Value


Histogram of Priceperbuildingarea for Class 1
Frequency
300
200
100
0
8
10
12
Y_label Value

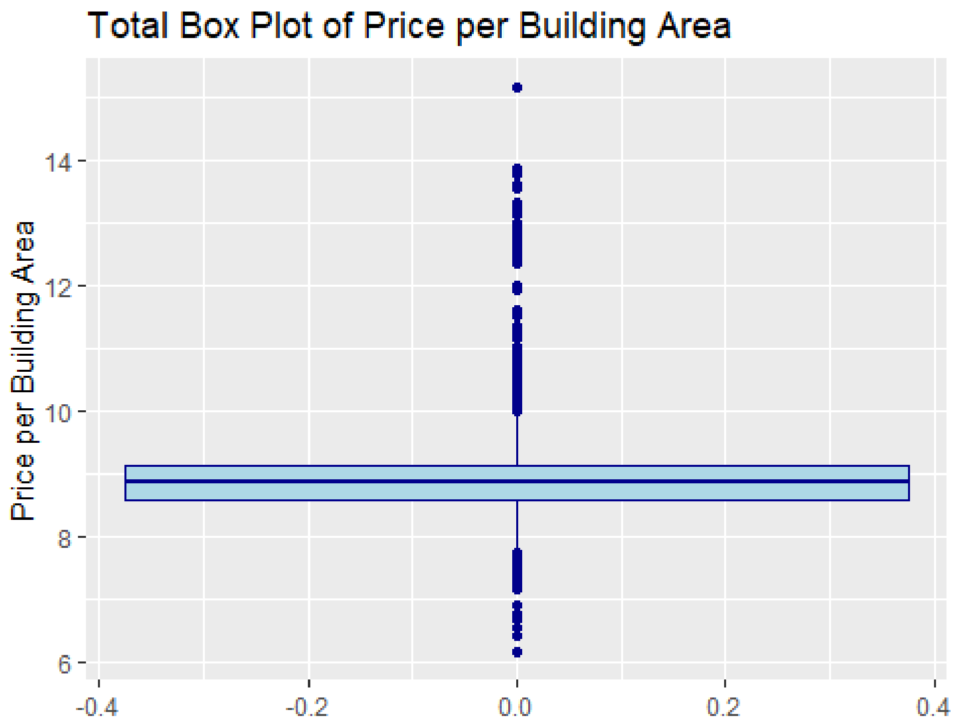




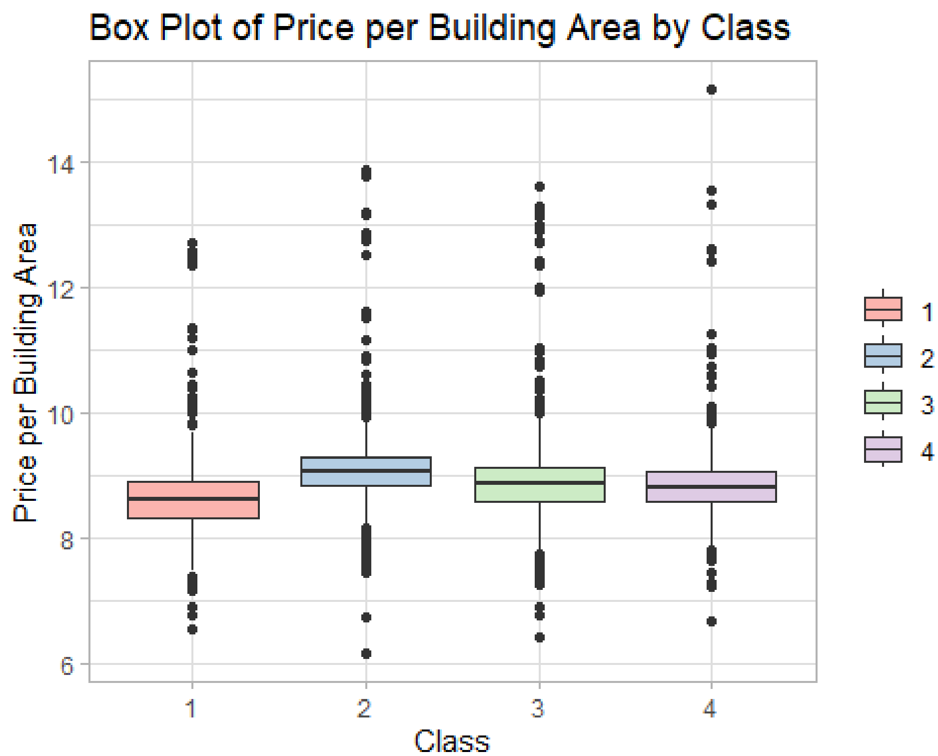
Step 4: Box plots

```
library(ggplot2)
```

```r
# Total Box Plot for 'LogPriceperbuildingarea'
ggplot(data_1, aes(y = LogPriceperbuildingarea)) +
  geom_boxplot(fill = "lightblue", color = "darkblue") +
  ggtitle("Total Box Plot of Price per Building Area") +
  ylab("Price per Building Area") +
  xlab("")
```

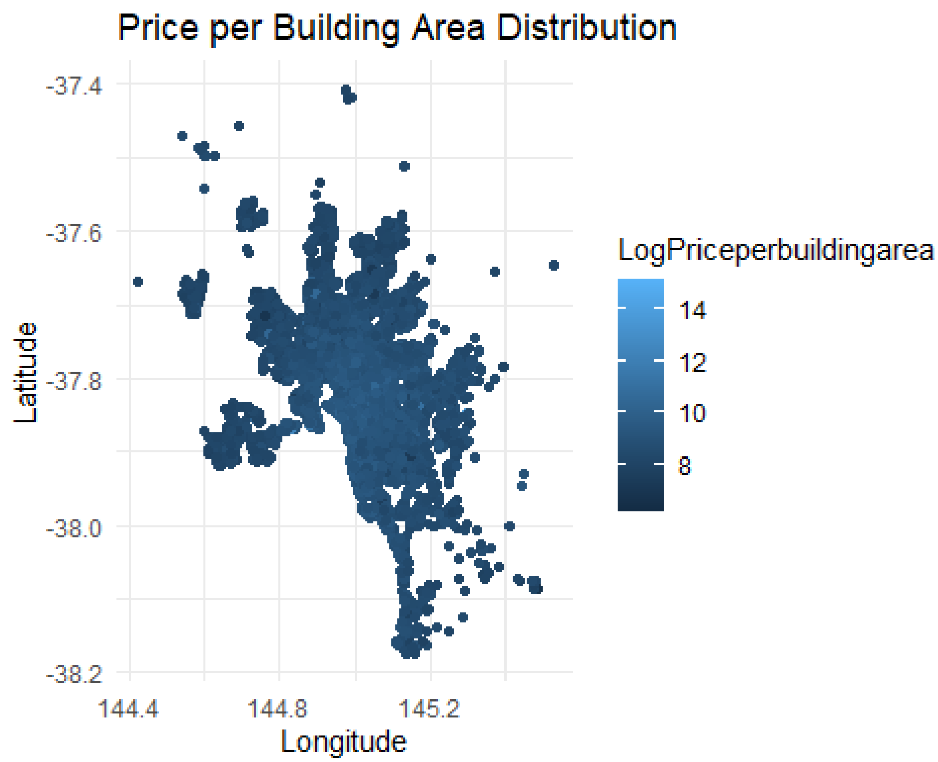



```r
# Box Plots for 'LogPriceperbuildingarea' by Class
ggplot(data_1, aes(x = factor(Class), y = LogPriceperbuildingarea, fill
= factor(Class))) +
  geom_boxplot() +
  scale_fill_brewer(palette = "Pastel1") +  # Color scheme
  ggtitle("Box Plot of Price per Building Area by Class") +
  xlab("Class") +
  ylab("Price per Building Area") +
  theme_light() +
  theme(legend.title = element_blank()) # Remove the legend title
```



Step 5: Visualization of Price distribution

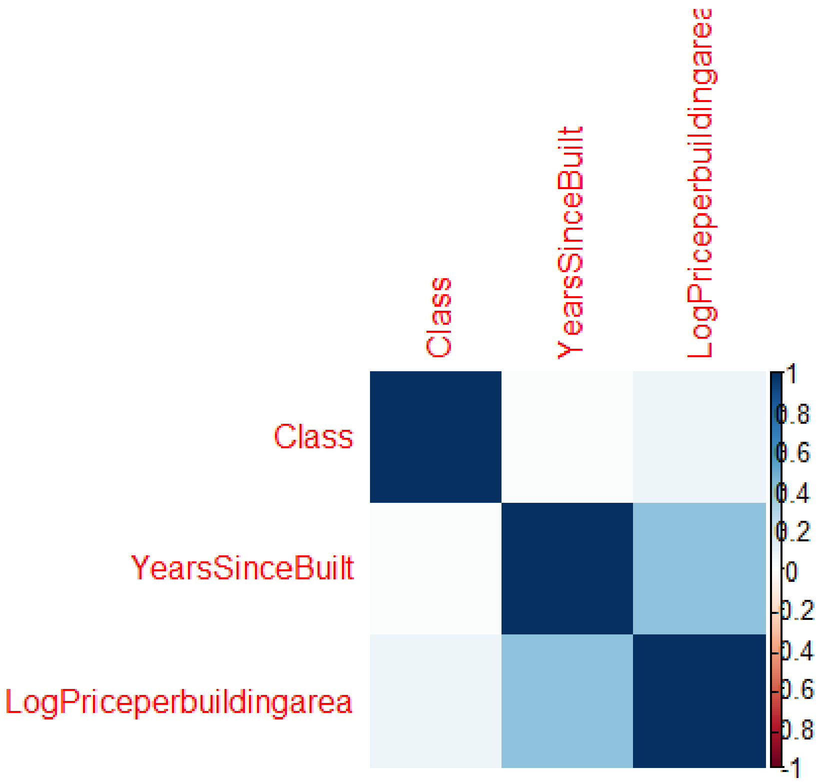
```
library(ggplot2)
library(ggmap)

# PricePerBuildingArea, visualize classification based on it
ggplot(data_1, aes(x = Longitude, y = Latitude, color =
LogPriceperbuildingarea)) + geom_point() + theme_minimal() +
ggtitle("Price per Building Area Distribution")
```



Step 6: Split Data into Training and Testing

```
library(caret)

## 载入需要的程辑包: lattice

set.seed(123) # For reproducibility
index <- createDataPartition(data_1$LogPriceperbuildingarea, p = 0.8,
list = FALSE)
trainData_1 <- data_1[index, ]
testData_1 <- data_1[-index, ]

trainData_trimmed_1 <- subset(trainData_1, select = c(Class,
YearsSinceBuilt, LogPriceperbuildingarea))
#trainData_trimmed$Class <- factor(trainData_trimmed$Class)

testData_trimmed_1 <- subset(testData_1, select = c(Class,
YearsSinceBuilt, LogPriceperbuildingarea))
#testData_trimmed$Class <- factor(testData_trimmed$Class)

# Convert the columns to numeric
selected_data <- data_1[, c("Class", "YearsSinceBuilt",
"LogPriceperbuildingarea")]
selected_data <- sapply(selected_data, as.numeric)

# Check if there are any missing values
if (anyNA(selected_data)) {
```

```
  # Handle missing values as needed
  selected_data <- na.omit(selected_data)
}

# Calculate the correlation matrix
correlation_matrix <- cor(selected_data, use = "pairwise.complete.obs")

# Create the colored correlation grid
corrplot(correlation_matrix, method = "color")
```

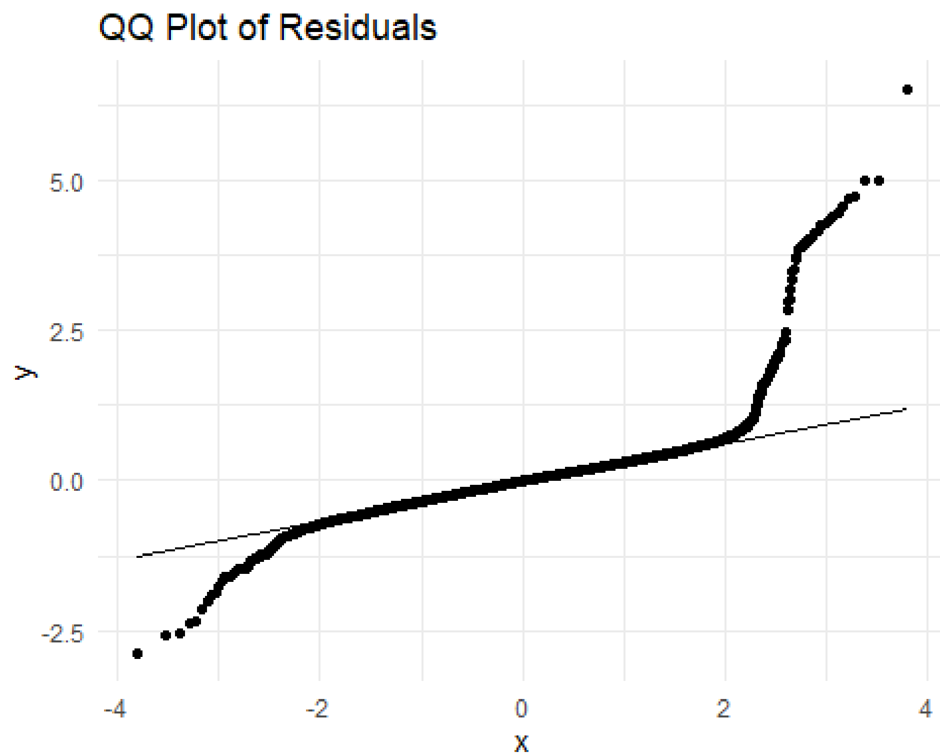


Step 7: Run Regression

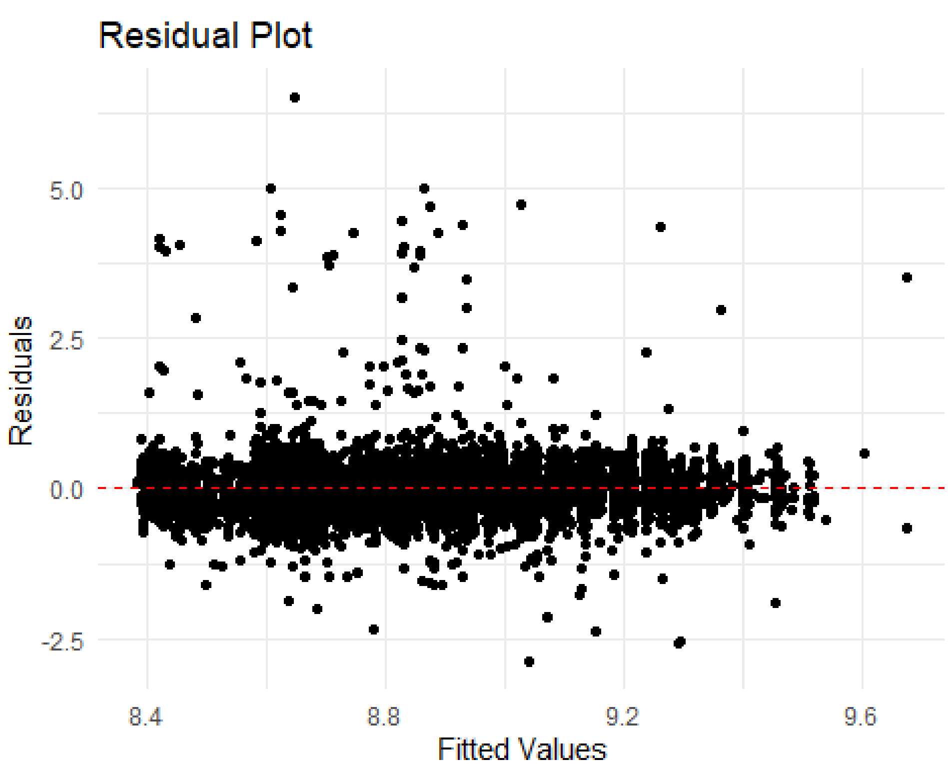
```
model_1<- lm(LogPriceperbuildingarea ~ ., data = trainData_trimmed_1)
summary(model_1)

##
## Call:
## lm(formula = LogPriceperbuildingarea ~ ., data =
trainData_trimmed_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8777 -0.2358 -0.0054  0.1996  6.5075
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        8.3889617  0.0141821  591.52   <2e-16 ***
```

```
## Class2          0.3830909  0.0164309   23.32   <2e-16 ***
## Class3          0.1903375  0.0154050   12.36   <2e-16 ***
## Class4          0.2301495  0.0193324   11.90   <2e-16 ***
## YearsSinceBuilt 0.0054086  0.0001555   34.77   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.464 on 7069 degrees of freedom
## Multiple R-squared:  0.2201, Adjusted R-squared:  0.2197
## F-statistic: 498.7 on 4 and 7069 DF,  p-value: < 2.2e-16
```

```r
# Load necessary libraries
library(ggplot2)

# Extract residuals and fitted values
residuals_1 <- residuals(model_1)
fitted_values_1 <- fitted(model_1)

# Create a data frame
data_df_1 <- data.frame(residuals_1 = residuals_1, fitted_values_1 =
fitted_values_1)

# QQ Plot
qqplot <- ggplot(data.frame(residuals_1 = residuals_1), aes(sample =
residuals_1)) +
  geom_qq() +
  geom_qq_line() +
  ggtitle("QQ Plot of Residuals") +
  theme_minimal()

# Residual Plot
residual_plot <- ggplot(data_df_1, aes(x = fitted_values_1, y =
residuals_1)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  ggtitle("Residual Plot") +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme_minimal()

# Show plots
print(qqplot)
```




```
print(residual_plot)
```



Step 8: Show Algorithm Metrics

```
predictions_1 <- predict(model_1, testData_trimmed_1)
actual_1 <- testData_trimmed_1$Priceperbuildingarea

## Warning: Unknown or uninitialised column: `Priceperbuildingarea`.

# Calculate RMSE and MAE
RMSE_1 <- sqrt(mean((predictions_1 - actual_1) ^ 2))
MAE_1 <- mean(abs(predictions_1 - actual_1))

# Print metrics
print(paste("RMSE:", RMSE_1))

## [1] "RMSE: NaN"

print(paste("MAE:", MAE_1))

## [1] "MAE: NaN"
```

refine the model by introducing distance from center

Step 9: calculate geo center

```
data_2=data_1

# Calculate the center point
center_longitude <- mean(data_2$Longitude)
center_latitude <- mean(data_2$Latitude)
# Print the center point
cat("Center Longitude:", center_longitude, "\n")

## Center Longitude: 144.9913

cat("Center Latitude:", center_latitude, "\n")

## Center Latitude: -37.80468
```

Step 10: calculate distance

```
# Function to calculate distance between two points given their
longitude and latitude
haversine_distance <- function(lon1, lat1, lon2, lat2) {
  # Convert latitude and longitude from degrees to radians
  lon1 <- lon1 * pi / 180
  lat1 <- lat1 * pi / 180
  lon2 <- lon2 * pi / 180
  lat2 <- lat2 * pi / 180

  # Haversine formula
  dlon <- lon2 - lon1
  dlat <- lat2 - lat1
  a <- sin(dlat/2)^2 + cos(lat1) * cos(lat2) * sin(dlon/2)^2
  c <- 2 * asin(sqrt(a))
```

```
  # Radius of the Earth in kilometers
  R <- 6371

  # Calculate the distance
  distance <- R * c
  return(distance)
}

# Calculate the center point
center_longitude <- mean(data_2$Longitude)
center_latitude <- mean(data_2$Latitude)
# Print the center point
cat("Center Longitude:", center_longitude, "\n")

## Center Longitude: 144.9913

cat("Center Latitude:", center_latitude, "\n")

## Center Latitude: -37.80468

# Calculate distance from center for each data point
data_2$distance_from_center <- apply(data_2, 1, function(row) {
  haversine_distance(as.numeric(row["Longitude"]),
as.numeric(row["Latitude"]), center_longitude, center_latitude)
})

# Print the updated data frame
print(data_2)

## # A tibble: 8,842 × 27
##    Suburb Address Rooms Type   Price Method SellerG Date
Distance Postcode
##    <chr>  <chr>   <dbl> <chr>  <dbl> <chr>  <chr>   <date>
<dbl>    <dbl>
##  1 Abbot… 25 Blo…     2 h     1.03e6 S      Biggin  2016-02-04
2.5      3067
##  2 Abbot… 5 Char…     3 h     1.46e6 SP     Biggin  2017-03-04
2.5      3067
##  3 Abbot… 55a Pa…     4 h     1.6 e6 VB     Nelson  2016-06-04
2.5      3067
##  4 Abbot… 124 Ya…     3 h     1.88e6 S      Nelson  2016-05-07
2.5      3067
##  5 Abbot… 98 Cha…     2 h     1.64e6 S      Nelson  2016-10-08
2.5      3067
##  6 Abbot… 10 Val…     2 h     1.10e6 S      Biggin  2016-10-08
2.5      3067
##  7 Abbot… 40 Nic…     3 h     1.35e6 VB     Nelson  2016-11-12
2.5      3067
##  8 Abbot… 123/56…     2 u     7.5 e5 S      Biggin  2016-11-12
2.5      3067
```

```
## 9 Abbot… 16 Wil…     2 h     1.31e6 S      Jellis  2016-10-15
2.5     3067
## 10 Abbot… 42 Hen…     3 h     1.20e6 S      Jellis  2016-07-16
2.5     3067
## # ℹ 8,832 more rows
## # ℹ 17 more variables: Bedroom2 <dbl>, Bathroom <dbl>, Car <dbl>,
## #   Landsize <dbl>, BuildingArea <dbl>, YearBuilt <dbl>, CouncilArea
<chr>,
## #   Latitude <dbl>, Longitude <dbl>, Regionname <chr>, Propertycount
<dbl>,
## #   YearOfSale <dbl>, YearsSinceBuilt <dbl>, Priceperbuildingarea
<dbl>,
## #   LogPriceperbuildingarea <dbl>, Class <fct>, distance_from_center
<dbl>
```

Step 11: Split Data into Training and Testing

```
library(caret)
set.seed(123) # For reproducibility
index <- createDataPartition(data_2$LogPriceperbuildingarea, p=0.8,
list=FALSE)
trainData_2 <- data_2[index, ]
testData_2 <- data_2[-index, ]

trainData_trimmed_2=subset(trainData_2, select =
c(distance_from_center, YearsSinceBuilt, LogPriceperbuildingarea))
#trainData_trimmed$Class <- factor(trainData_trimmed$Class)

testData_trimmed_2=subset(testData_2, select = c(distance_from_center,
YearsSinceBuilt, LogPriceperbuildingarea))
#testData_trimmed$Class <- factor(testData_trimed$Class)
```

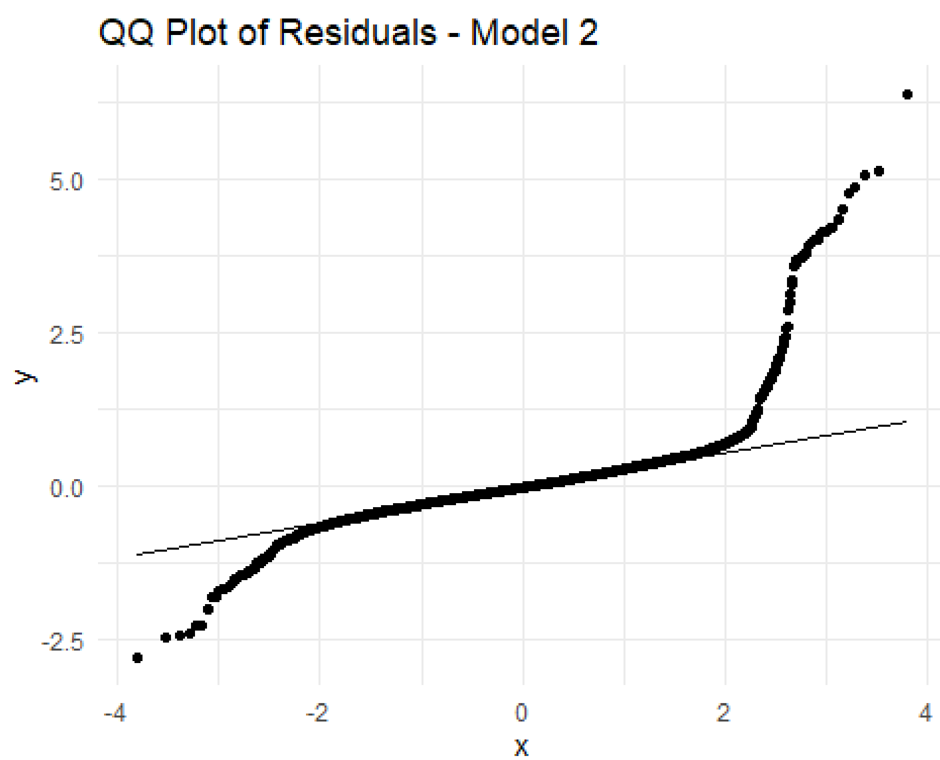
Step 12: Run Regression

```
model_2 <- lm(LogPriceperbuildingarea ~ ., data = trainData_trimmed_2)
summary(model_2)
```

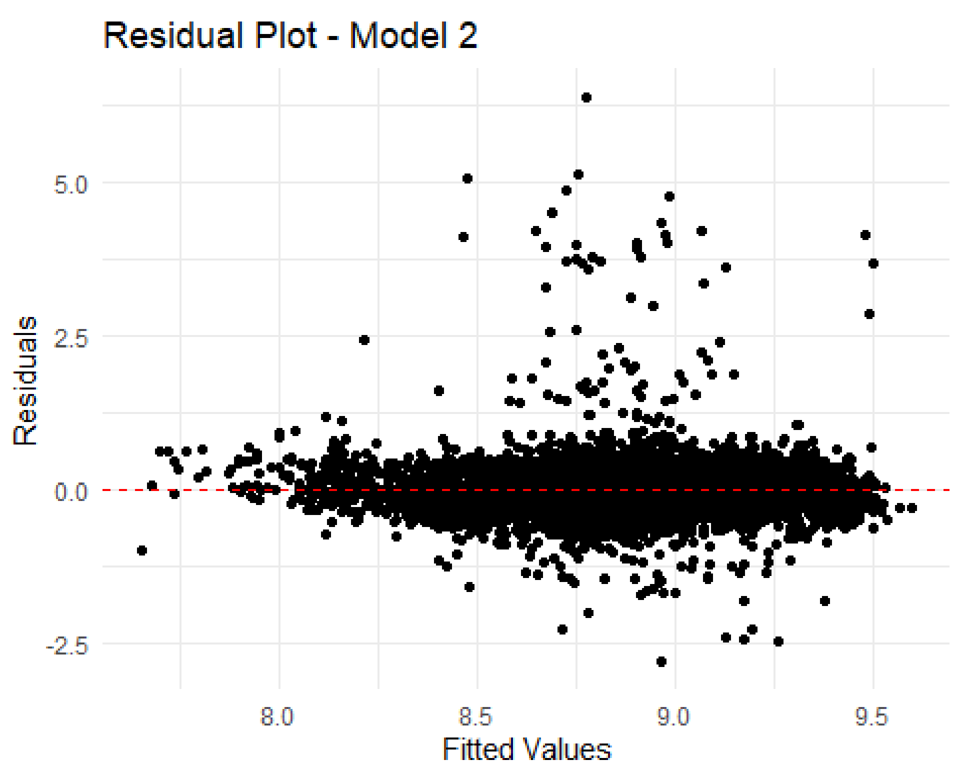
```
##
## Call:
## lm(formula = LogPriceperbuildingarea ~ ., data =
trainData_trimmed_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7966 -0.2100 -0.0236  0.1743  6.3809
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          8.9899464  0.0144129  623.74   <2e-16 ***
## distance_from_center -0.0259987  0.0007086  -36.69   <2e-16 ***
## YearsSinceBuilt       0.0039851  0.0001545   25.79   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4416 on 7071 degrees of freedom
## Multiple R-squared:  0.2934, Adjusted R-squared:  0.2932
## F-statistic:  1468 on 2 and 7071 DF,  p-value: < 2.2e-16
```

```r
# Extract residuals and fitted values for model2
residuals_2 <- residuals(model_2)
fitted_values_2 <- fitted(model_2)

# Create a data frame for model2
data_df_model_2 <- data.frame(residuals_2 = residuals_2,
fitted_values_2 = fitted_values_2)

# QQ Plot for model2
qqplot_model <- ggplot(data.frame(residuals_2 = residuals_2),
aes(sample = residuals_2)) +
  geom_qq() +
  geom_qq_line() +
  ggtitle("QQ Plot of Residuals - Model 2") +
  theme_minimal()

# Residual Plot for model2
residual_plot_model <- ggplot(data_df_model_2, aes(x = fitted_values_2,
y = residuals_2)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  ggtitle("Residual Plot - Model 2") +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme_minimal()

# Show plots for model1
print(qqplot_model)
```

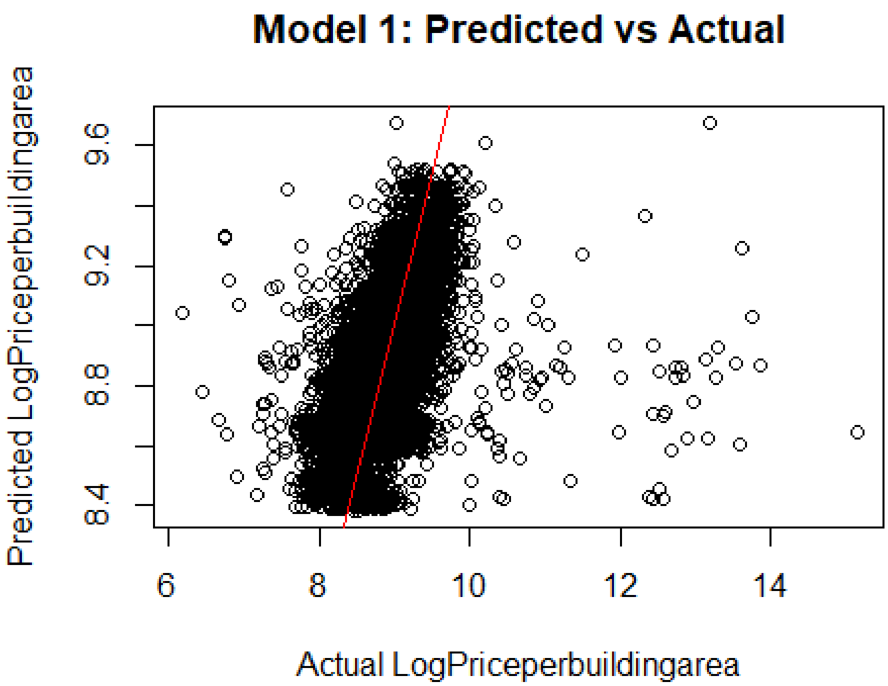



```
print(residual_plot_model)
```

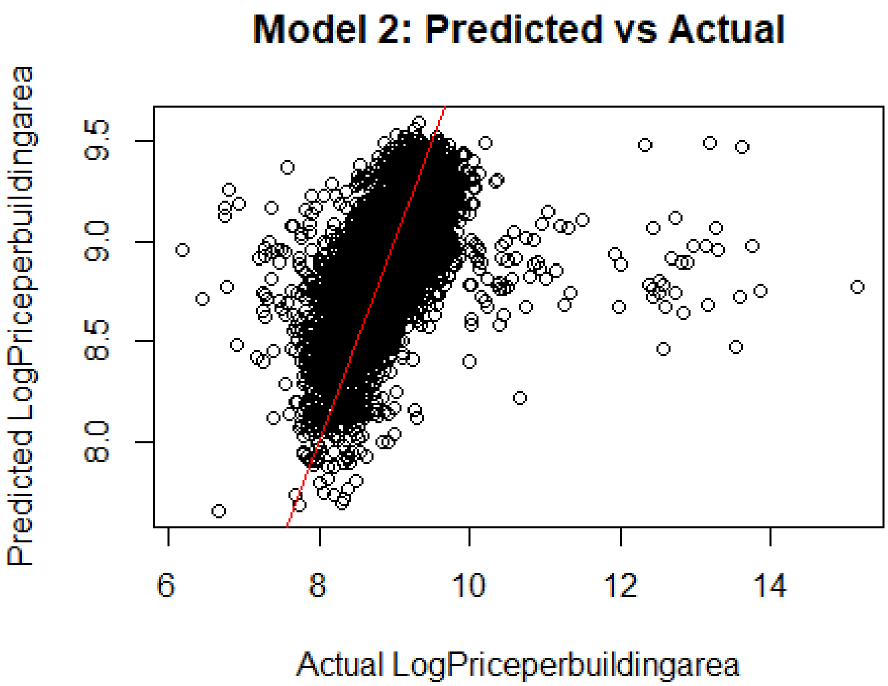


Step 13: visual comparison of both models

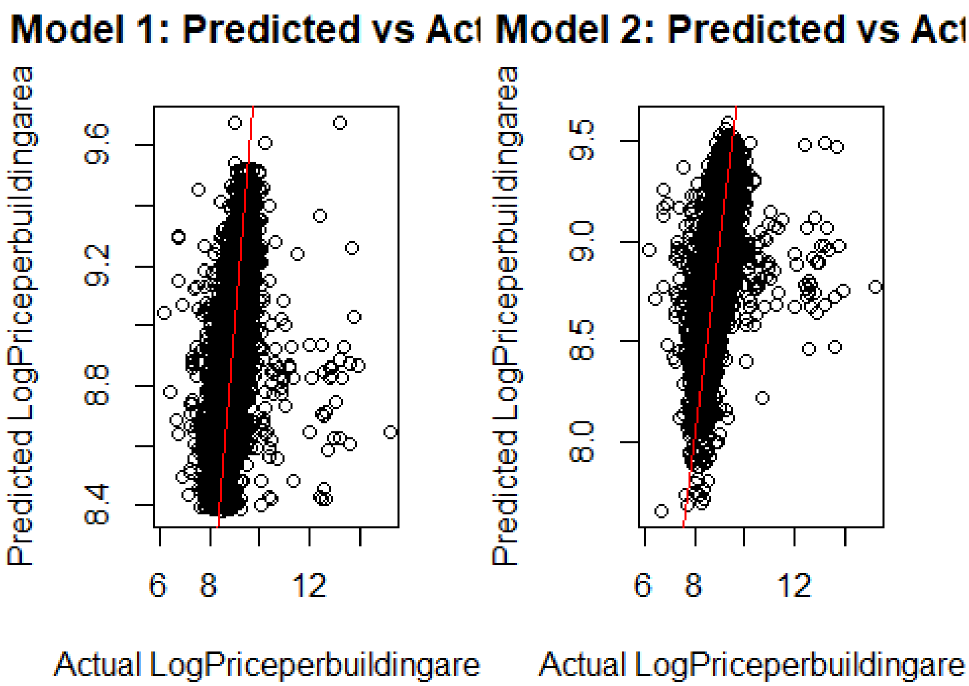
```
# Predicted vs Actual values for model 1(distance based model)
plot(trainData_trimmed_1$LogPriceperbuildingarea, predict(model_1),
     xlab = "Actual LogPriceperbuildingarea", ylab = "Predicted
LogPriceperbuildingarea",
     main = "Model 1: Predicted vs Actual")

# Add a reference line with slope = 1
abline(0, 1, col = "red")
```




```
# Predicted vs Actual values for model 2(class based model)
plot(trainData_trimmed_2$LogPriceperbuildingarea, predict(model_2),
     xlab = "Actual LogPriceperbuildingarea", ylab = "Predicted
LogPriceperbuildingarea",
     main = "Model 2: Predicted vs Actual")

# Add a reference line with slope = 1
abline(0, 1, col = "red")
```

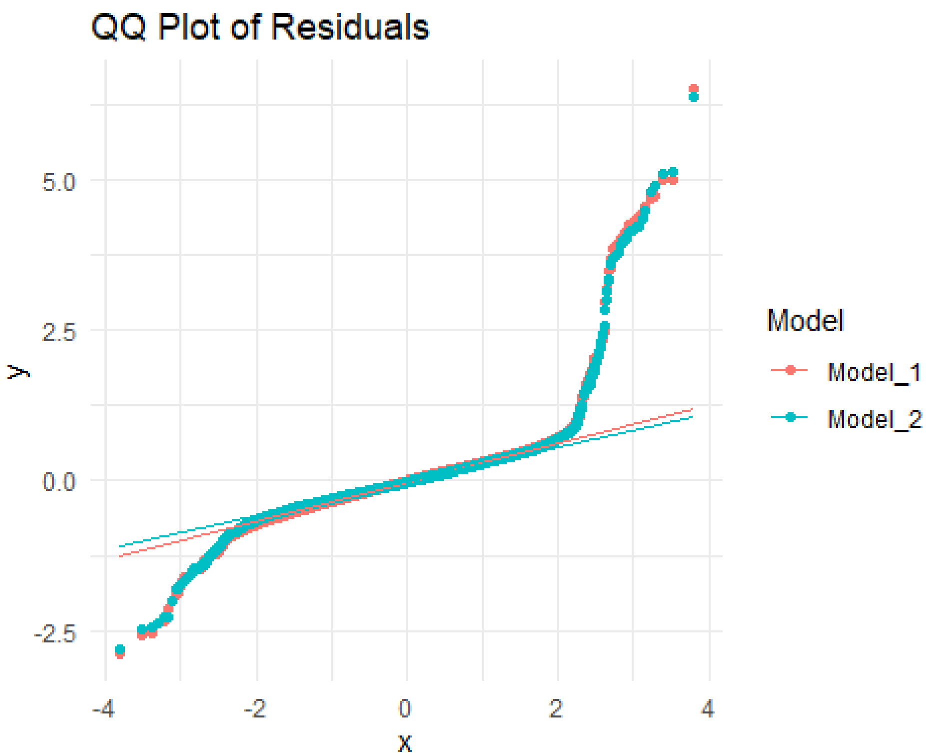



```r
# Set up the plotting area
par(mfrow = c(1, 2))

# Predicted vs Actual values for model 1(distance based model)
plot(trainData_trimmed_1$LogPriceperbuildingarea, predict(model_1),
     xlab = "Actual LogPriceperbuildingarea", ylab = "Predicted
LogPriceperbuildingarea",
     main = "Model 1: Predicted vs Actual")
abline(0, 1, col = "red")  # Add a reference line with slope = 1

# Predicted vs Actual values for model 2(class based model)
plot(trainData_trimmed_2$LogPriceperbuildingarea, predict(model_2),
     xlab = "Actual LogPriceperbuildingarea", ylab = "Predicted
LogPriceperbuildingarea",
     main = "Model 2: Predicted vs Actual")
abline(0, 1, col = "red")  # Add a reference line with slope = 1
```

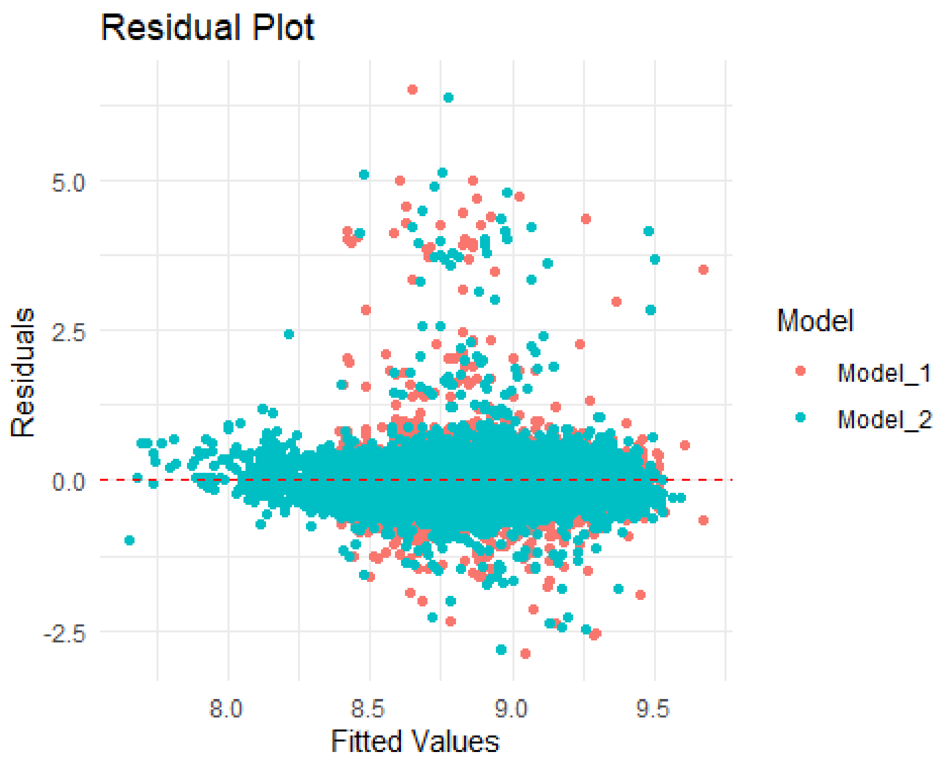



```r
# Combine data frames for both models
combined_data <- rbind(
  data.frame(Model = "Model_1", residuals = residuals_1, fitted_values
= fitted_values_1),
  data.frame(Model = "Model_2", residuals = residuals_2, fitted_values
= fitted_values_2)
)

# QQ Plot
qqplot_combined <- ggplot(combined_data, aes(sample = residuals, color
= Model)) +
  geom_qq() +
  geom_qq_line() +
  ggtitle("QQ Plot of Residuals") +
  theme_minimal()

# Residual Plot
residual_plot_combined <- ggplot(combined_data, aes(x = fitted_values,
y = residuals, color = Model)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  ggtitle("Residual Plot") +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme_minimal()
```

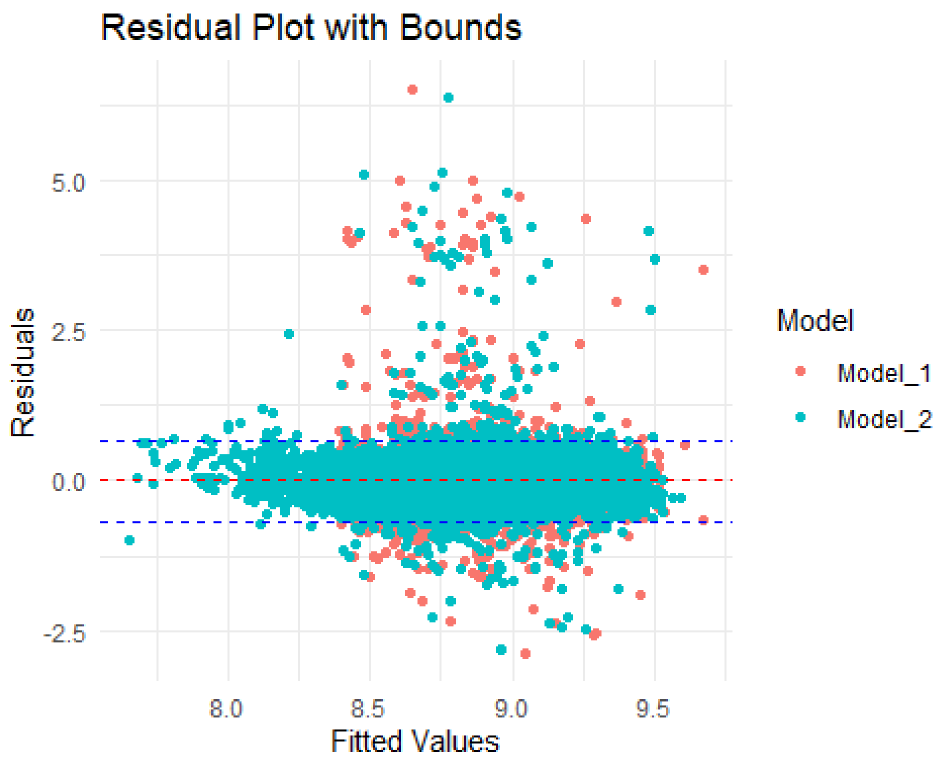
```r
# Show combined plots
print(qqplot_combined)
```




```r
print(residual_plot_combined)
```

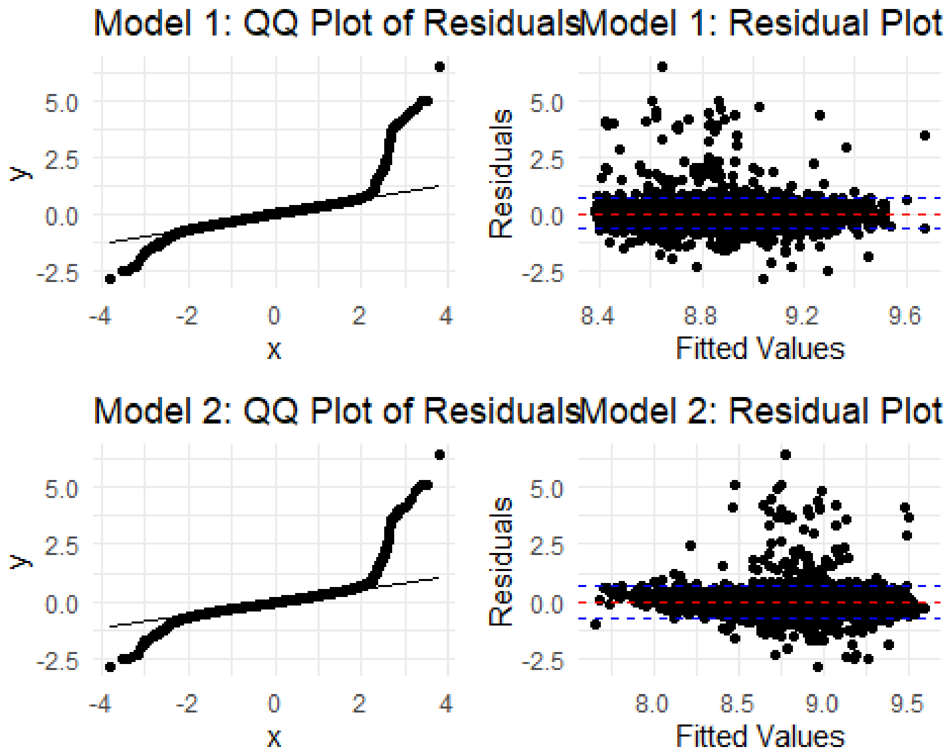



```r
# Calculate lower and upper bounds for residuals (e.g., 95% confidence
interval)
lower_bound <- quantile(combined_data$residuals, 0.025)
upper_bound <- quantile(combined_data$residuals, 0.975)

# Residual Plot with bounds
residual_plot_combined <- ggplot(combined_data, aes(x = fitted_values,
y = residuals, color = Model)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  geom_hline(yintercept = lower_bound, linetype = "dashed", color =
"blue") +
  geom_hline(yintercept = upper_bound, linetype = "dashed", color =
"blue") +
  ggtitle("Residual Plot with Bounds") +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme_minimal()

# Show the residual plot with bounds
print(residual_plot_combined)
```




```
library(gridExtra)

##
## 载入程辑包: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine

# Calculate lower and upper bounds for residuals (e.g., 95% confidence
interval)
lower_bound <- quantile(combined_data$residuals, 0.025)
upper_bound <- quantile(combined_data$residuals, 0.975)

# QQ Plot for Model 1
qqplot_model1 <- ggplot(combined_data[combined_data$Model == "Model_1",
], aes(sample = residuals)) +
  geom_qq() +
  geom_qq_line() +
  ggtitle("Model 1: QQ Plot of Residuals") +
  theme_minimal()

# Residual Plot for Model 1
residual_plot_model1 <- ggplot(combined_data[combined_data$Model ==
"Model_1", ], aes(x = fitted_values, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
```

```r
  geom_hline(yintercept = lower_bound, linetype = "dashed", color =
"blue") +
  geom_hline(yintercept = upper_bound, linetype = "dashed", color =
"blue") +
  ggtitle("Model 1: Residual Plot") +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme_minimal()

# QQ Plot for Model 2
qqplot_model2 <- ggplot(combined_data[combined_data$Model == "Model_2",
], aes(sample = residuals)) +
  geom_qq() +
  geom_qq_line() +
  ggtitle("Model 2: QQ Plot of Residuals") +
  theme_minimal()

# Residual Plot for Model 2
residual_plot_model2 <- ggplot(combined_data[combined_data$Model ==
"Model_2", ], aes(x = fitted_values, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  geom_hline(yintercept = lower_bound, linetype = "dashed", color =
"blue") +
  geom_hline(yintercept = upper_bound, linetype = "dashed", color =
"blue") +
  ggtitle("Model 2: Residual Plot") +
  xlab("Fitted Values") +
  ylab("Residuals") +
  theme_minimal()

# Arrange plots in a 2x2 grid
grid.arrange(qqplot_model1, residual_plot_model1, qqplot_model2,
residual_plot_model2, ncol = 2, nrow = 2)
```



Step 14: Comparison of metrics

```
summary(model_1)

##
## Call:
## lm(formula = LogPriceperbuildingarea ~ ., data =
trainData_trimmed_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8777 -0.2358 -0.0054  0.1996  6.5075
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    8.3889617  0.0141821  591.52   <2e-16 ***
## Class2         0.3830909  0.0164309   23.32   <2e-16 ***
## Class3         0.1903375  0.0154050   12.36   <2e-16 ***
## Class4         0.2301495  0.0193324   11.90   <2e-16 ***
## YearsSinceBuilt 0.0054086  0.0001555   34.77   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.464 on 7069 degrees of freedom
## Multiple R-squared:  0.2201, Adjusted R-squared:  0.2197
## F-statistic: 498.7 on 4 and 7069 DF,  p-value: < 2.2e-16

summary(model_2)
```

```
##
## Call:
## lm(formula = LogPriceperbuildingarea ~ ., data =
trainData_trimmed_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7966 -0.2100 -0.0236  0.1743  6.3809
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           8.9899464  0.0144129  623.74   <2e-16 ***
## distance_from_center -0.0259987  0.0007086  -36.69   <2e-16 ***
## YearsSinceBuilt       0.0039851  0.0001545   25.79   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4416 on 7071 degrees of freedom
## Multiple R-squared:  0.2934, Adjusted R-squared:  0.2932
## F-statistic:  1468 on 2 and 7071 DF,  p-value: < 2.2e-16

# Metrics for model 1
metrics_model_1 <- summary(model_1)
r_squared_model_1 <- metrics_model_1$r.squared
adj_r_squared_model_1 <- metrics_model_1$adj.r.squared
residual_std_error_model_1 <- sqrt(metrics_model_1$sigma^2)
f_statistic_model_1 <- metrics_model_1$fstatistic[1]

# Metrics for model 2
metrics_model_2 <- summary(model_2)
r_squared_model_2 <- metrics_model_2$r.squared
adj_r_squared_model_2 <- metrics_model_2$adj.r.squared
residual_std_error_model_2 <- sqrt(metrics_model_2$sigma^2)
f_statistic_model_2 <- metrics_model_2$fstatistic[1]

# Print metrics for both models
cat("Model 1 Metrics:\n")

## Model 1 Metrics:

cat("R-squared:", r_squared_model_1, "\n")

## R-squared: 0.2201009

cat("Adjusted R-squared:", adj_r_squared_model_1, "\n")

## Adjusted R-squared: 0.2196596

cat("Residual Standard Error:", residual_std_error_model_1, "\n")

## Residual Standard Error: 0.4639518
```

```r
cat("F-statistic:", f_statistic_model_1, "\n\n")

## F-statistic: 498.7481

cat("Model 2 Metrics:\n")

## Model 2 Metrics:

cat("R-squared:", r_squared_model_2, "\n")

## R-squared: 0.2933877

cat("Adjusted R-squared:", adj_r_squared_model_2, "\n")

## Adjusted R-squared: 0.2931878

cat("Residual Standard Error:", residual_std_error_model_2, "\n")

## Residual Standard Error: 0.441553

cat("F-statistic:", f_statistic_model_2, "\n")

## F-statistic: 1467.951
```

Step 15: Final comment

Step 16: Business application