classification, regression with outlier exclusion

Steven

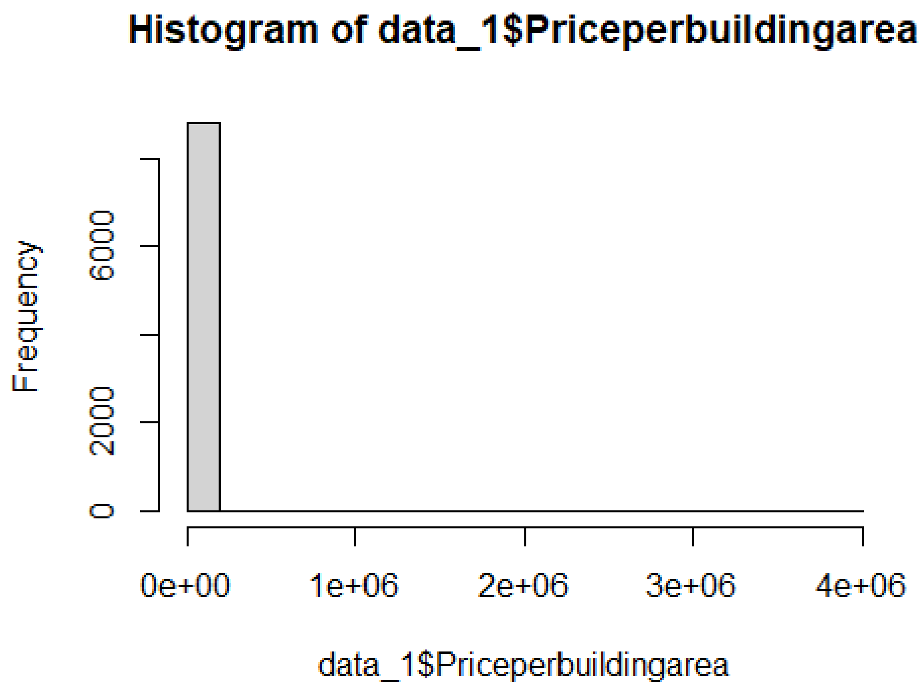
2024-02-16

Data source and original descriptions

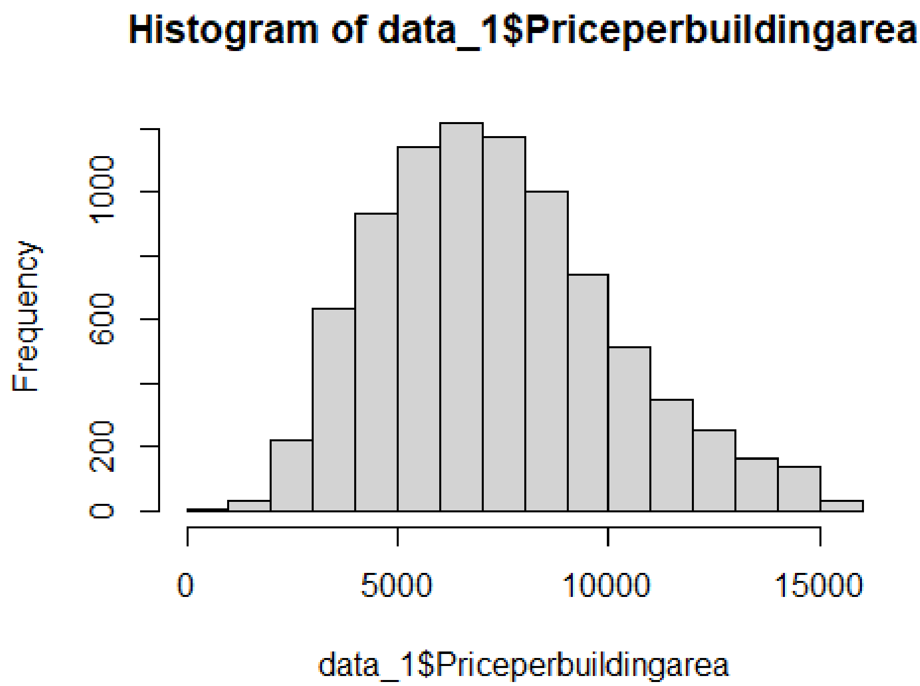
Step 1: Data Cleaning and Preparation

```
# 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)
```




```r
# Step 5: Eliminate outliers in 'Priceperbuildingarea'
Q1 <- quantile(data_1$Priceperbuildingarea, 0.25, na.rm = TRUE)
Q3 <- quantile(data_1$Priceperbuildingarea, 0.75, na.rm = TRUE)
IQR <- Q3 - Q1
data_1 <- data_1 %>%
  filter(Priceperbuildingarea >= (Q1 - 1.5 * IQR) &
Priceperbuildingarea <= (Q3 + 1.5 * IQR))

# hist 2
hist(data_1$Priceperbuildingarea)
```




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

```
##  [1] "Suburb"               "Address"              "Rooms"
##  [4] "Type"                 "Price"                "Method"
##  [7] "SellerG"              "Date"                 "Distance"
## [10] "Postcode"             "Bedroom2"             "Bathroom"
## [13] "Car"                  "Landsize"             "BuildingArea"
## [16] "YearBuilt"            "CouncilArea"          "Latitude"
## [19] "Longitude"            "Regionname"           "Propertycount"
## [22] "YearOfSale"           "YearsSinceBuilt"
"Priceperbuildingarea"
```

```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")

# Print correlation matrix
print(correlation_matrix)
```

```
##                              Rooms         Price     Distance
Postcode
## Rooms                  1.0000000000  0.501094726  0.279584628
0.08667307
## Price                  0.5010947257  1.000000000 -0.220490168
0.03541692
## Distance               0.2795846276 -0.220490168  1.000000000
0.50293962
## Postcode               0.0866730687  0.035416925  0.502939623
1.00000000
## Bedroom2               0.9642190596  0.485052942  0.286713808
0.08980185
## Bathroom               0.6218748893  0.488440346  0.124599926
0.11424239
## Car                    0.4021756258  0.216086293  0.260624274
0.06013886
## Landsize               0.0990786454  0.058431506  0.138053610
0.07187845
## BuildingArea           0.6245141720  0.561318543  0.136755737
0.08186941
## YearBuilt              0.0007745297 -0.295673010  0.297603488
0.10279907
## Latitude               0.0175199666 -0.224470815 -0.066282456 -
0.19444725
## Longitude              0.0822794916  0.224735399  0.165473824
0.36046687
## Propertycount         -0.0836318090 -0.063615943 -0.003382938
0.03570636
## YearOfSale             0.1913029831 -0.002704971  0.327247947
0.12848419
## YearsSinceBuilt        0.0026839596  0.296210171 -0.292274590 -
0.10067900
## Priceperbuildingarea -0.1837209402  0.513547389 -0.514936406 -
0.05173650
##                          Bedroom2    Bathroom          Car
Landsize
## Rooms                  0.964219060  0.62187489  0.40217563
0.099078645
## Price                  0.485052942  0.48844035  0.21608629
0.058431506
## Distance               0.286713808  0.12459993  0.26062427
0.138053610
## Postcode               0.089801854  0.11424239  0.06013886
0.071878455
## Bedroom2               1.000000000  0.62524398  0.40534655
0.099037744
## Bathroom               0.625243979  1.00000000  0.31099793
0.075464539
## Car                    0.405346551  0.31099793  1.00000000
0.120537396
```

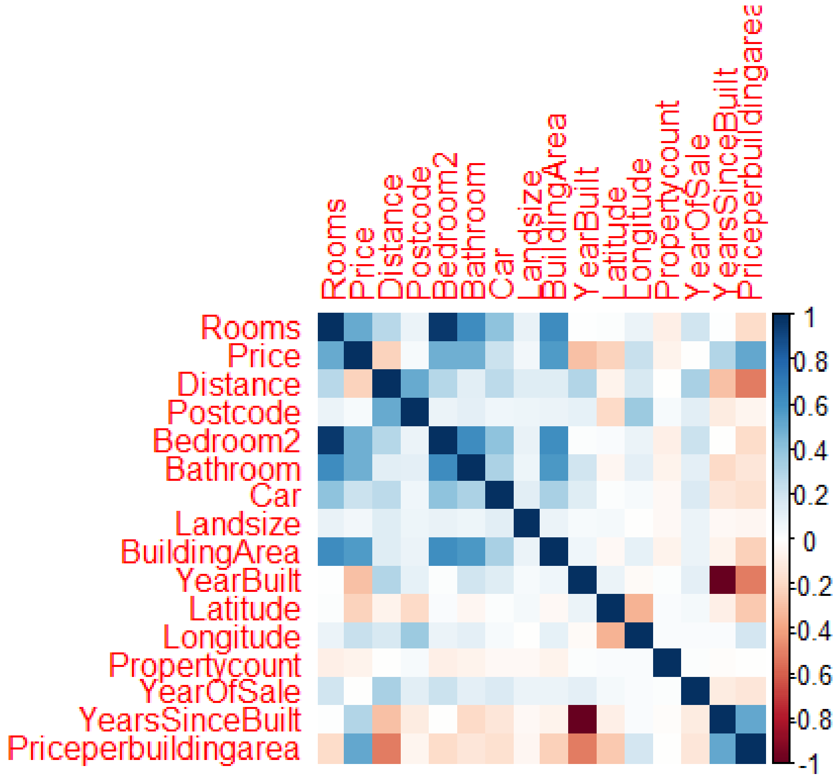
```
## Landsize              0.099037744  0.07546454  0.12053740
1.000000000
## BuildingArea          0.613469920  0.57272213  0.32271057
0.081978978
## YearBuilt             0.011124211  0.19707483  0.13496208
0.034314694
## Latitude              0.021817673 -0.04430443  0.01646177
0.042805418
## Longitude             0.081787849  0.11170203  0.03390007 -
0.009794645
## Propertycount        -0.081790074 -0.06013173 -0.03065207 -
0.032234320
## YearOfSale            0.213542728  0.11221420  0.15686142
0.085526603
## YearsSinceBuilt      -0.007283995 -0.19543590 -0.13239251 -
0.032835826
## Priceperbuildingarea -0.187782815 -0.13925201 -0.16947349 -
0.040829514
##                      BuildingArea     YearBuilt    Latitude
Longitude
## Rooms                  0.62451417  0.0007745297  0.01751997
0.082279492
## Price                  0.56131854 -0.2956730101 -0.22447081
0.224735399
## Distance               0.13675574  0.2976034883 -0.06628246
0.165473824
## Postcode               0.08186941  0.1027990711 -0.19444725
0.360466868
## Bedroom2               0.61346992  0.0111242112  0.02181767
0.081787849
## Bathroom               0.57272213  0.1970748307 -0.04430443
0.111702026
## Car                    0.32271057  0.1349620821  0.01646177
0.033900071
## Landsize               0.08197898  0.0343146942  0.04280542 -
0.009794645
## BuildingArea           1.00000000  0.0623103272 -0.03368642
0.101937067
## YearBuilt              0.06231033  1.0000000000  0.08946128 -
0.024585573
## Latitude              -0.03368642  0.0894612827  1.00000000 -
0.347901002
## Longitude              0.10193707 -0.0245855731 -0.34790100
1.000000000
## Propertycount         -0.06024646  0.0170254136  0.02877662
0.025165982
## YearOfSale             0.08701842  0.1185121940  0.04888162
0.020664425
## YearsSinceBuilt       -0.06085997 -0.9998387213 -0.08875451
0.025008056
```

```
## Priceperbuildingarea  -0.23622285 -0.5193228334 -0.26808344
0.186735548
##                       Propertycount   YearOfSale YearsSinceBuilt
## Rooms                  -0.083631809  0.191302983     0.002683960
## Price                  -0.063615943 -0.002704971     0.296210171
## Distance               -0.003382938  0.327247947    -0.292274590
## Postcode                0.035706359  0.128484185    -0.100678995
## Bedroom2               -0.081790074  0.213542728    -0.007283995
## Bathroom               -0.060131728  0.112214200    -0.195435898
## Car                    -0.030652066  0.156861415    -0.132392512
## Landsize               -0.032234320  0.085526603    -0.032835826
## BuildingArea           -0.060246459  0.087018419    -0.060859970
## YearBuilt               0.017025414  0.118512194    -0.999838721
## Latitude                0.028776623  0.048881622    -0.088754510
## Longitude               0.025165982  0.020664425     0.025008056
## Propertycount           1.000000000  0.019590133    -0.016704842
## YearOfSale              0.019590133  1.000000000    -0.100660488
## YearsSinceBuilt        -0.016704842 -0.100660488     1.000000000
## Priceperbuildingarea   -0.006301737 -0.132182942     0.517961496
##                       Priceperbuildingarea
## Rooms                          -0.183720940
## Price                           0.513547389
## Distance                       -0.514936406
## Postcode                       -0.051736503
## Bedroom2                       -0.187782815
## Bathroom                       -0.139252010
## Car                            -0.169473491
## Landsize                       -0.040829514
## BuildingArea                   -0.236222853
## YearBuilt                      -0.519322833
## Latitude                       -0.268083437
## Longitude                       0.186735548
## Propertycount                  -0.006301737
## YearOfSale                     -0.132182942
## YearsSinceBuilt                 0.517961496
## Priceperbuildingarea            1.000000000
```

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

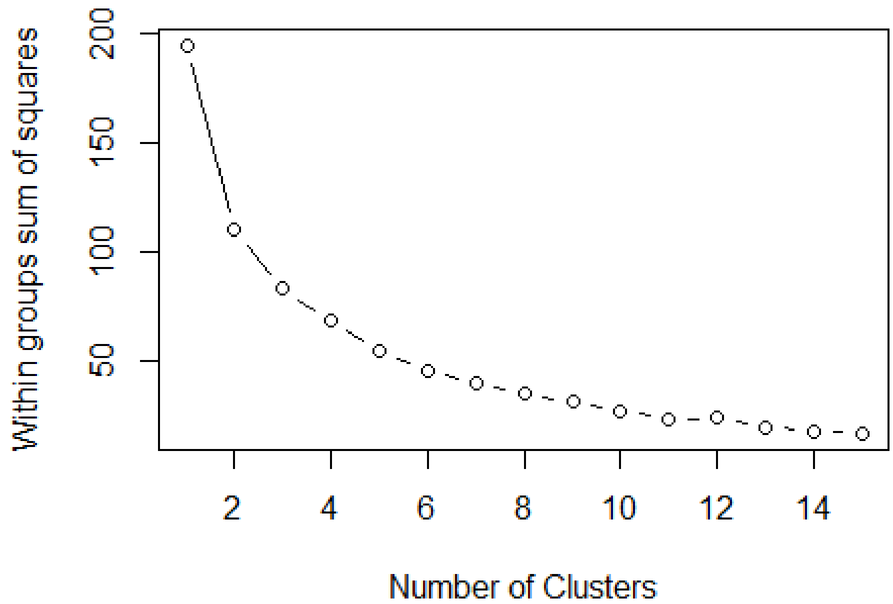
```
## corrplot 0.92 loaded
```

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

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

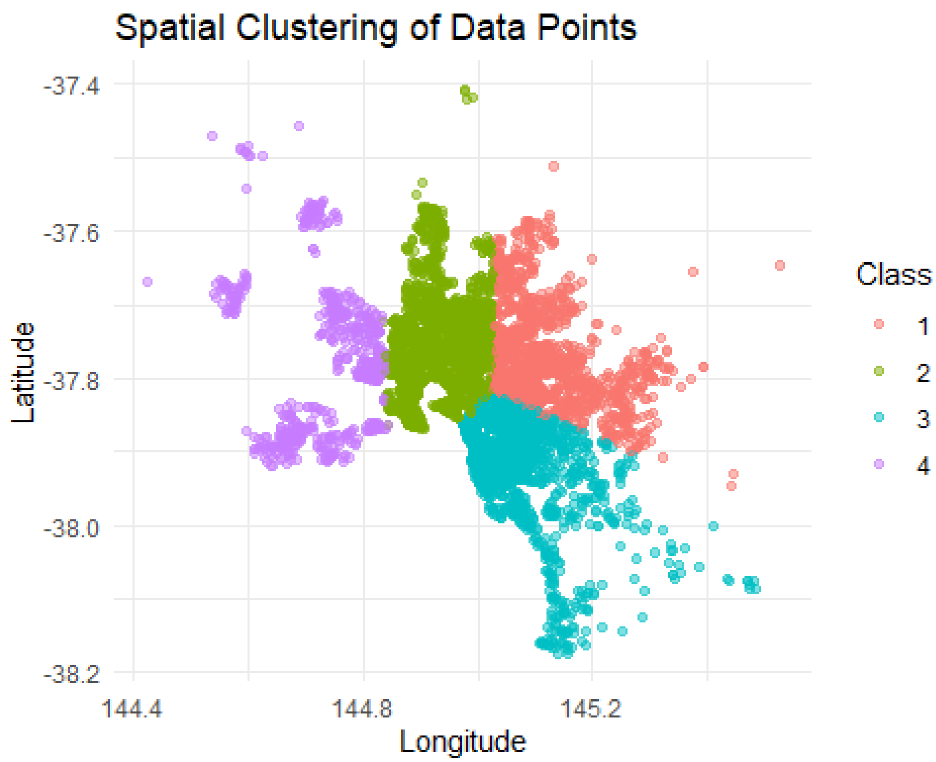


Step 2: Classification with K-Means Clustering:

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

# coordinates
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")
```



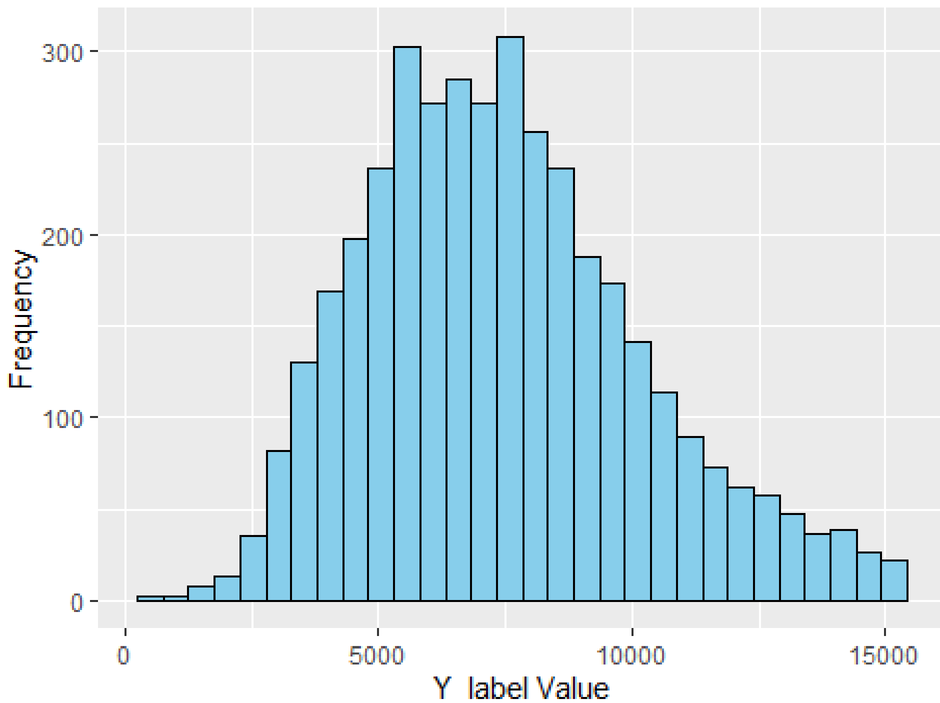

```
# K-Means Clustering
set.seed(123) # For reproducibility
k <- 4 # Choose based on analysis, e.g., using the Elbow Method above
km_res <- kmeans(coords, centers = k)

# 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()
```



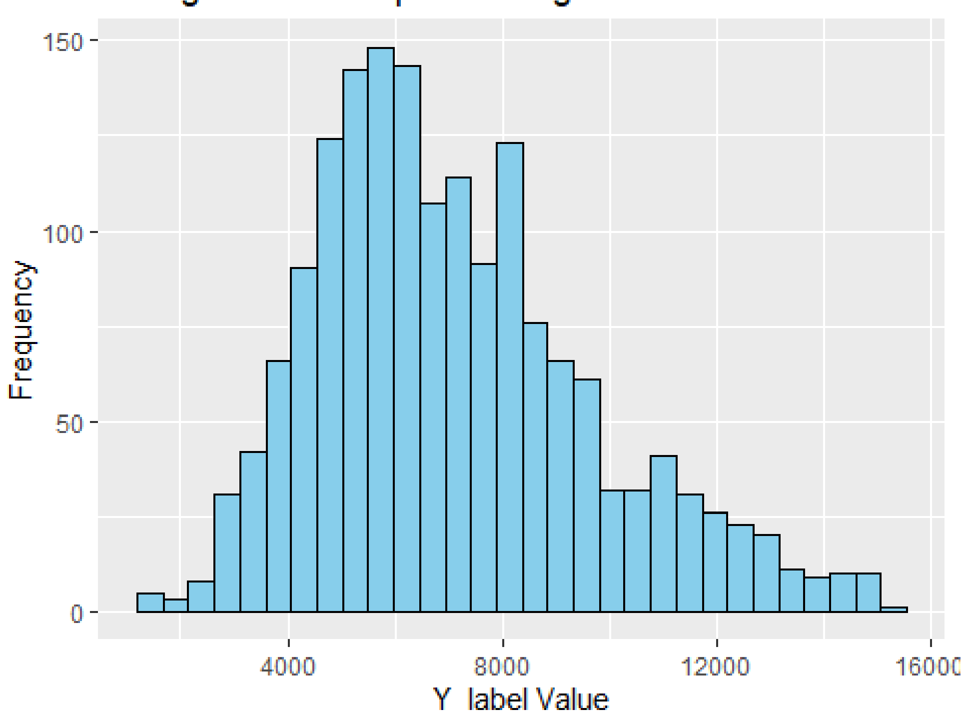
Step 3: visualize histograms:

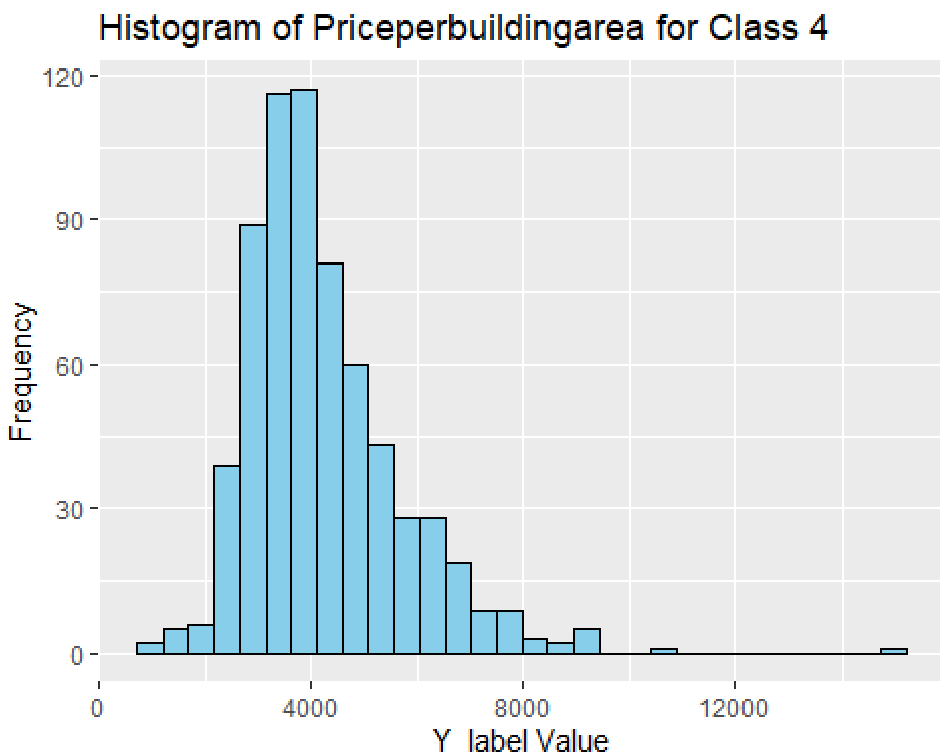
```
library(ggplot2)
library(dplyr)

# '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 = Priceperbuildingarea)) +
    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
}
```

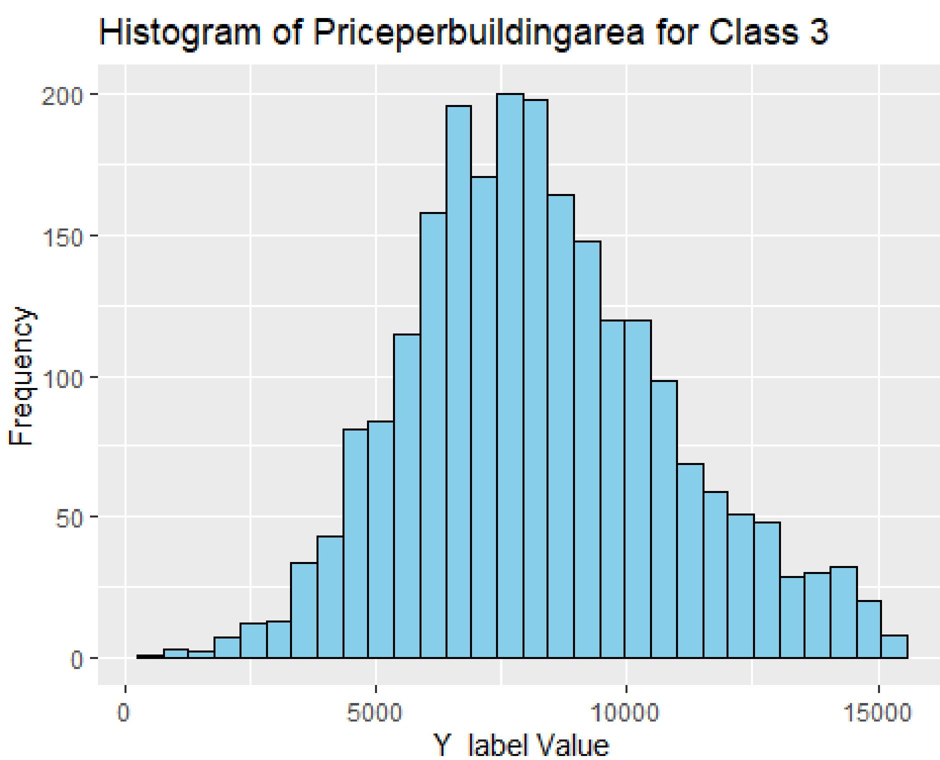
Histogram of Priceperbuildingarea for Class 2



Histogram of Priceperbuildingarea for Class 1



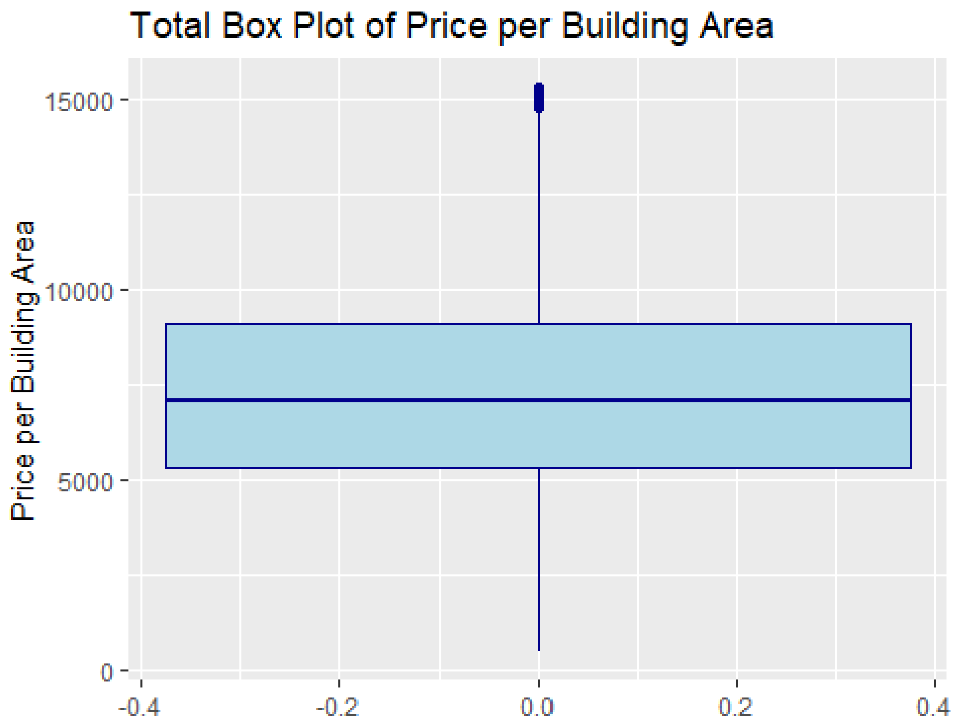




Step 4: Box plots

```
library(ggplot2)
```

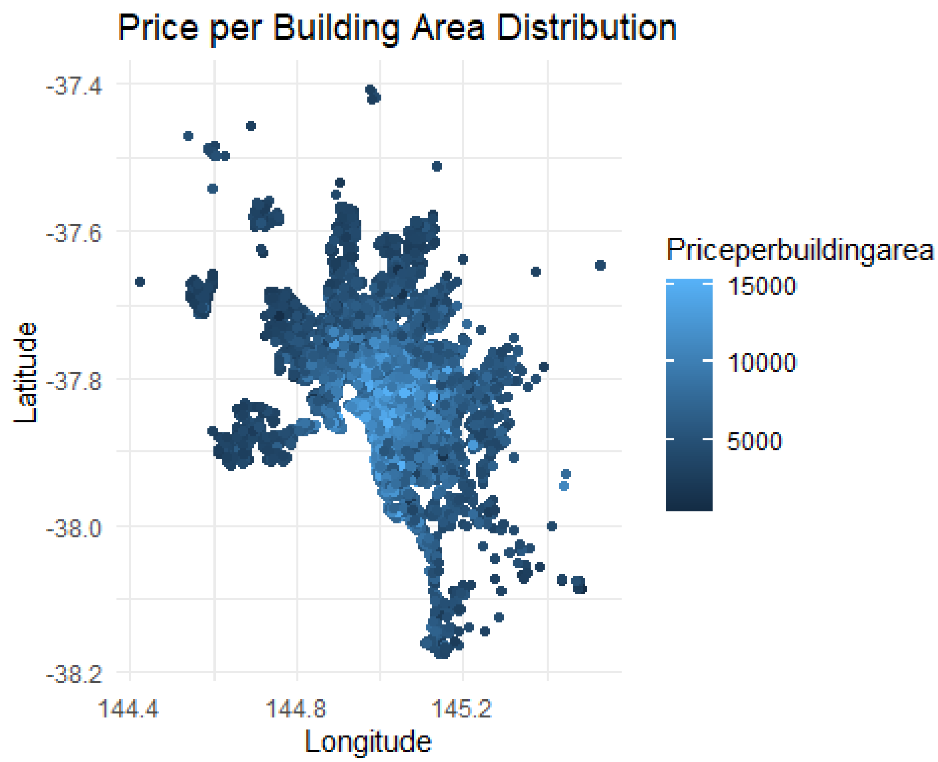
```r
# Total Box Plot for 'Priceperbuildingarea'
ggplot(data_1, aes(y = Priceperbuildingarea)) +
  geom_boxplot(fill = "lightblue", color = "darkblue") +
  ggtitle("Total Box Plot of Price per Building Area") +
  ylab("Price per Building Area") +
  xlab("")
```




```r
# Box Plots for 'Priceperbuildingarea' by Class
ggplot(data_1, aes(x = factor(Class), y = Priceperbuildingarea, 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

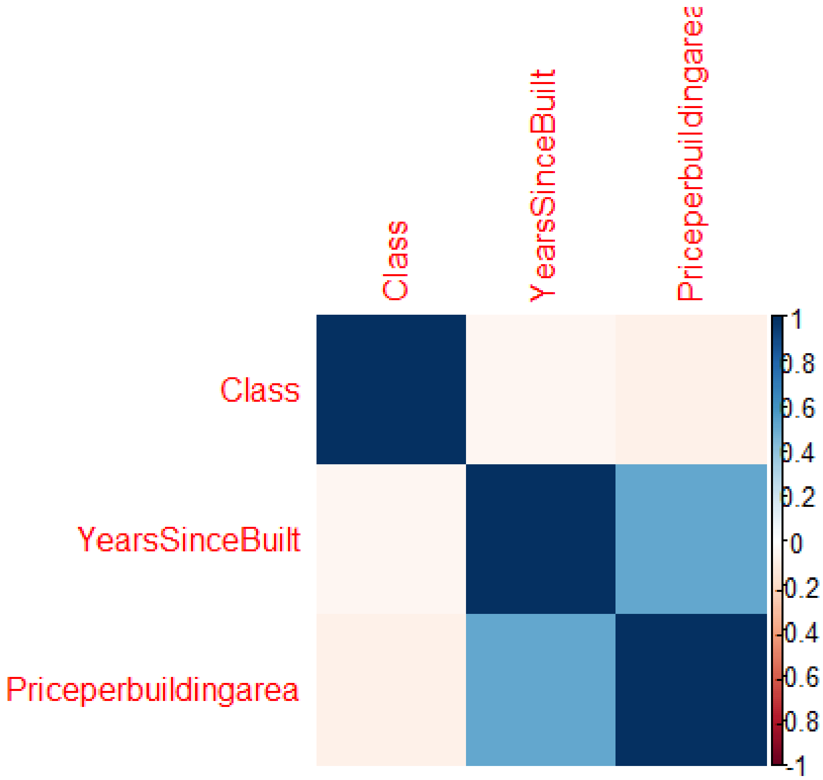
```
library(ggplot2)
library(ggmap)

# PricePerBuildingArea, visualize classification based on it
ggplot(data_1, aes(x = Longitude, y = Latitude, color =
Priceperbuildingarea)) + 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$Priceperbuildingarea, 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, Priceperbuildingarea))
#trainData_trimmed$Class <- factor(trainData_trimmed$Class)

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

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

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

```r
  # 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")
```

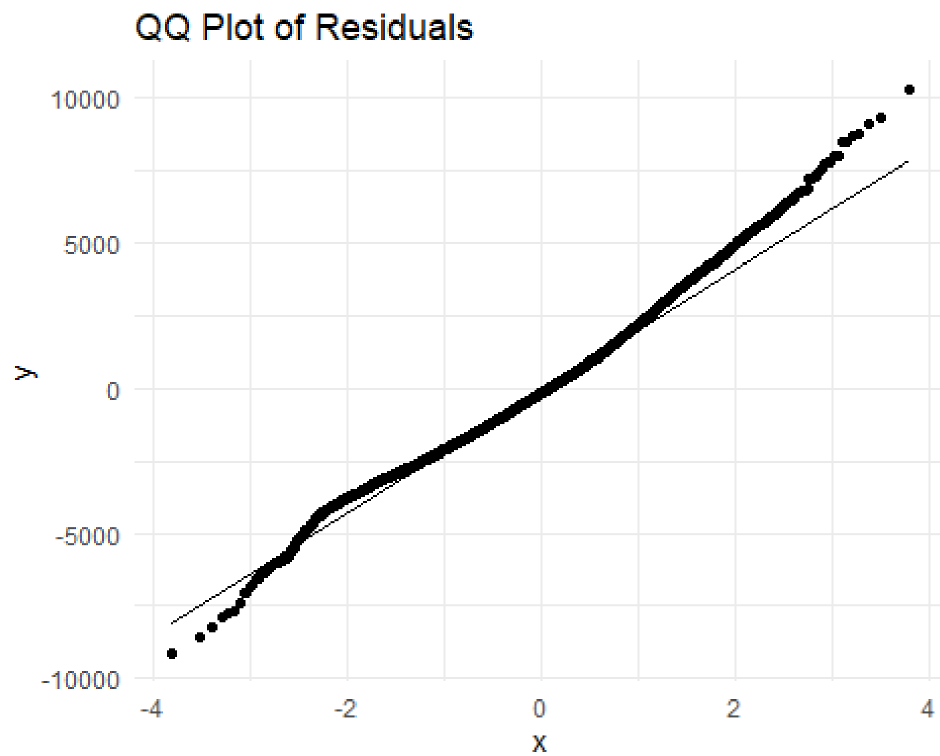


Step 7: Run Regression

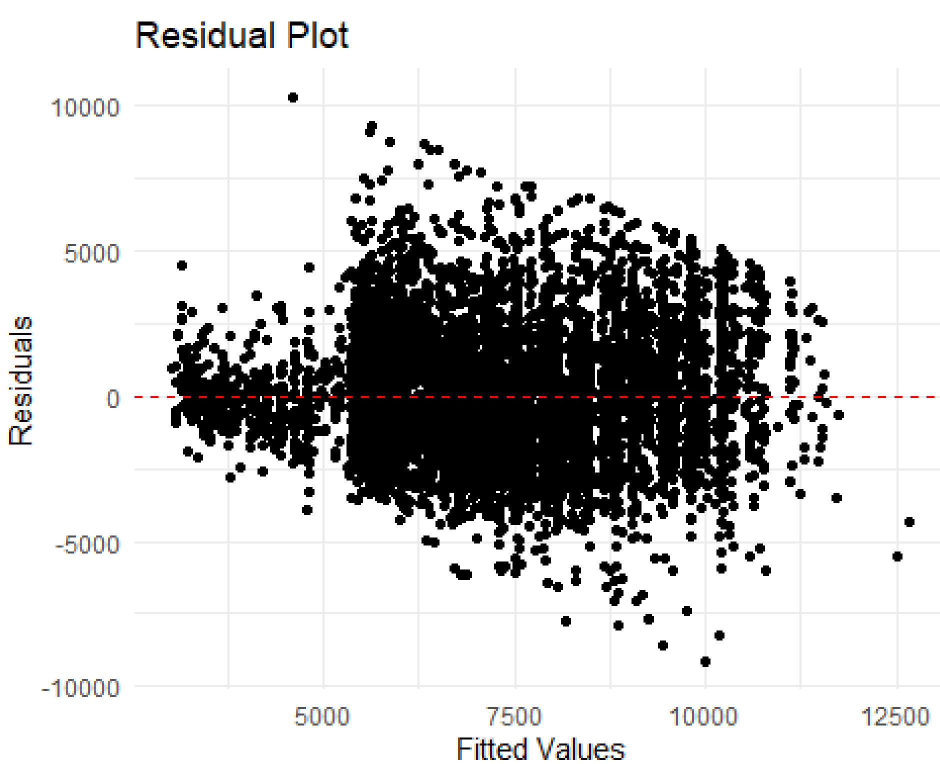
```r
model_1<- lm(Priceperbuildingarea ~ ., data = trainData_trimmed_1)
summary(model_1)

##
## Call:
## lm(formula = Priceperbuildingarea ~ ., data = trainData_trimmed_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9142.2 -1537.9  -169.2  1298.5 10299.7
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        5354.0584    69.6539  76.867   <2e-16 ***
## Class2               26.4614    72.5400   0.365    0.715
```

```
## Class3            917.8225    79.4400  11.554   <2e-16 ***
## Class4          -2329.5063   114.2920 -20.382   <2e-16 ***
## YearsSinceBuilt    38.3039     0.7665  49.972   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2211 on 6827 degrees of freedom
## Multiple R-squared:  0.3621, Adjusted R-squared:  0.3618
## F-statistic: 968.9 on 4 and 6827 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

# 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: 2305.1990809635"

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

## [1] "MAE: 1738.2692701769"
```

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: 144.9909

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

## Center Latitude: -37.80384
```

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))
```

```r
  # 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: 144.9909

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

## Center Latitude: -37.80384

# 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: 8,537 × 26
##    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 8,527 more rows
## # ℹ 16 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>,
## #   Class <fct>, distance_from_center <dbl>
```

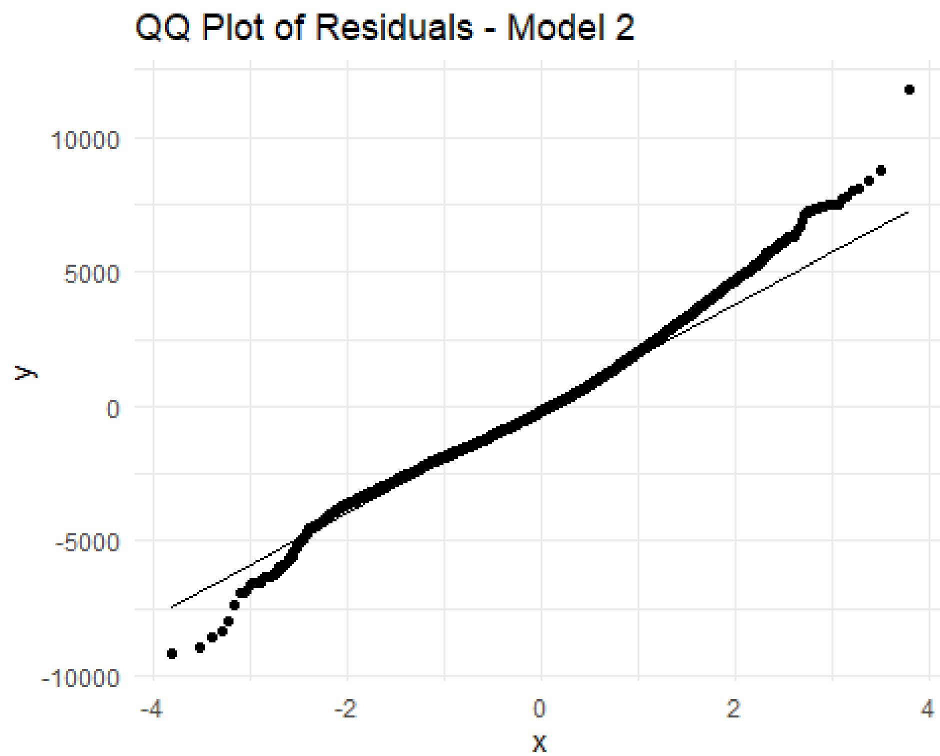
Step 11: Split Data into Training and Testing

```
library(caret)
set.seed(123) # For reproducibility
index <- createDataPartition(data_2$Priceperbuildingarea, 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, Priceperbuildingarea))
#trainData_trimmed$Class <- factor(trainData_trimmed$Class)

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

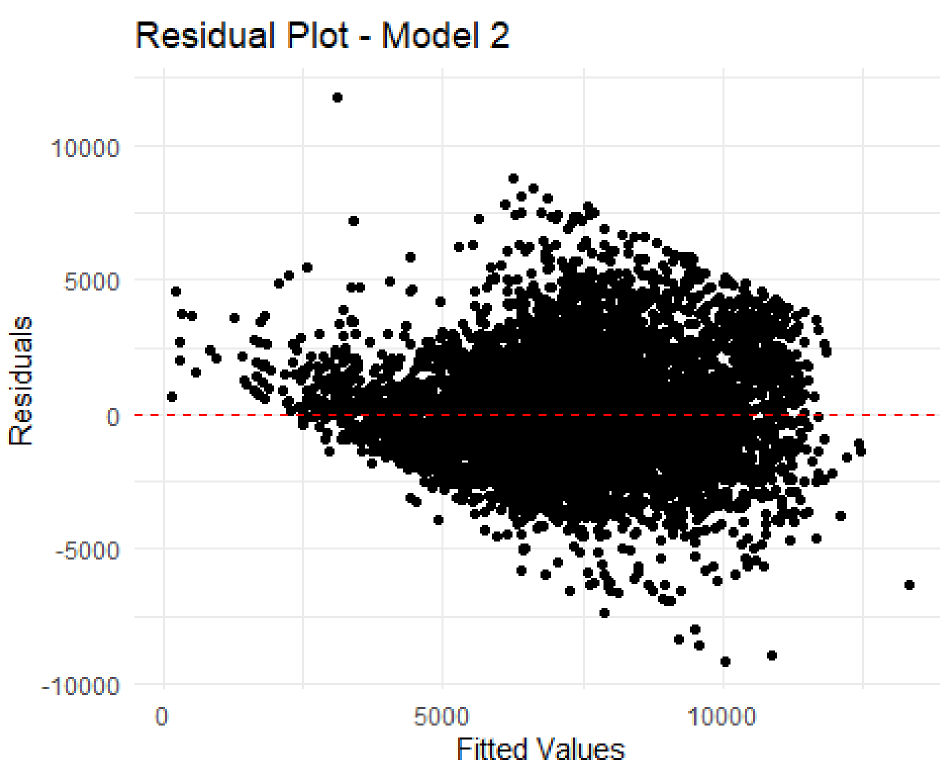
Step 12: Run Regression

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

##
## Call:
## lm(formula = Priceperbuildingarea ~ ., data = trainData_trimmed_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9173.5 -1389.2  -185.5  1214.2 11771.7
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          7594.6911    67.6008  112.35   <2e-16 ***
## distance_from_center -146.4118     3.3025  -44.33   <2e-16 ***
## YearsSinceBuilt        31.2631     0.7414   42.17   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2070 on 6829 degrees of freedom
```

```
## Multiple R-squared:  0.441,  Adjusted R-squared:  0.4408
## F-statistic:  2693 on 2 and 6829 DF,  p-value: < 2.2e-16

# 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)
```

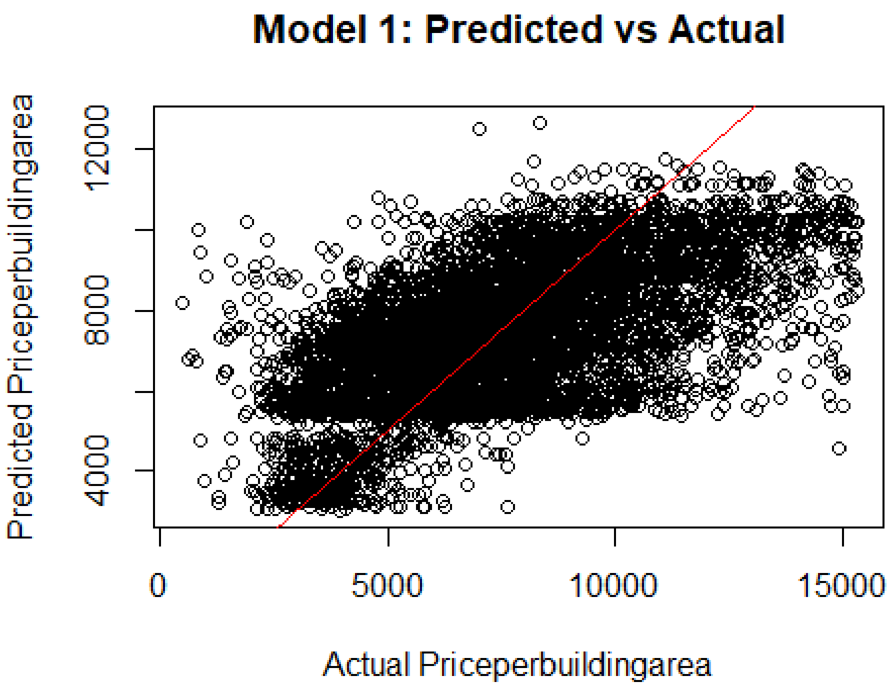



```
print(residual_plot_model)
```

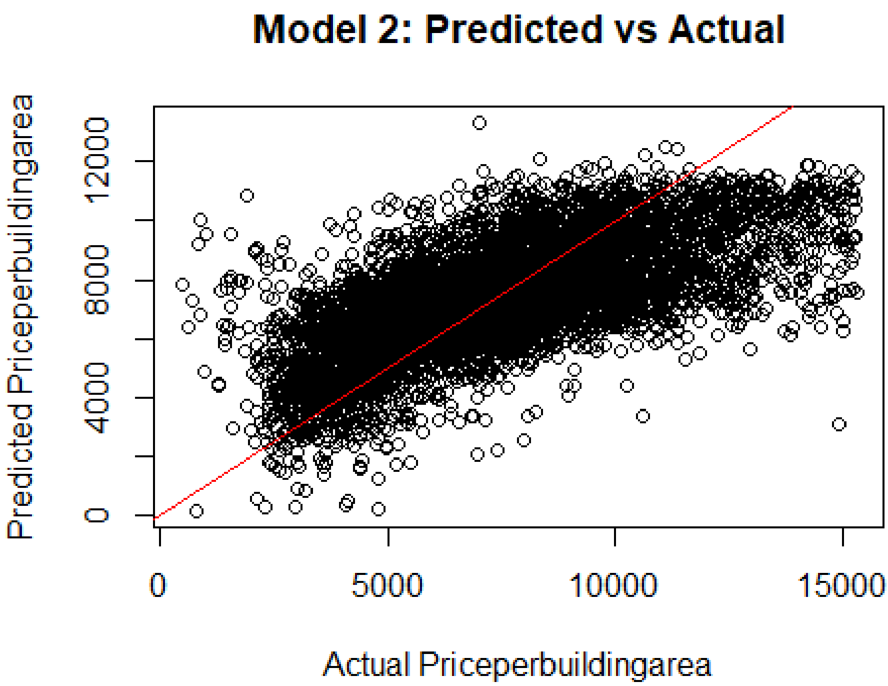


Step 13: visual comparison of both models

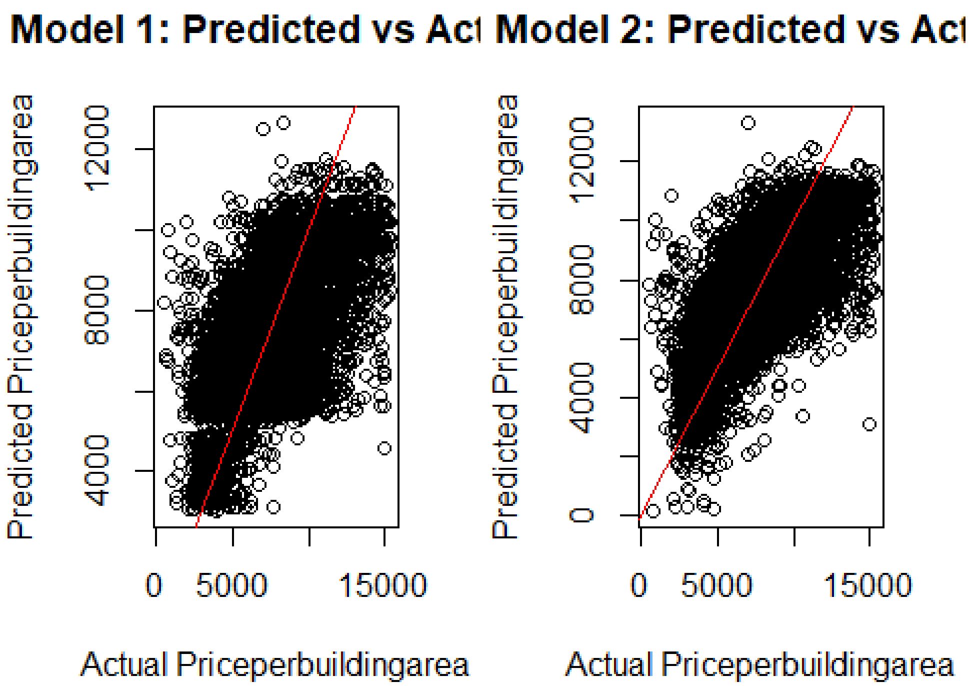
```r
# Predicted vs Actual values for model 1(distance based model)
plot(trainData_trimmed_1$Priceperbuildingarea, predict(model_1),
     xlab = "Actual Priceperbuildingarea", ylab = "Predicted
Priceperbuildingarea",
     main = "Model 1: Predicted vs Actual")

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




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

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

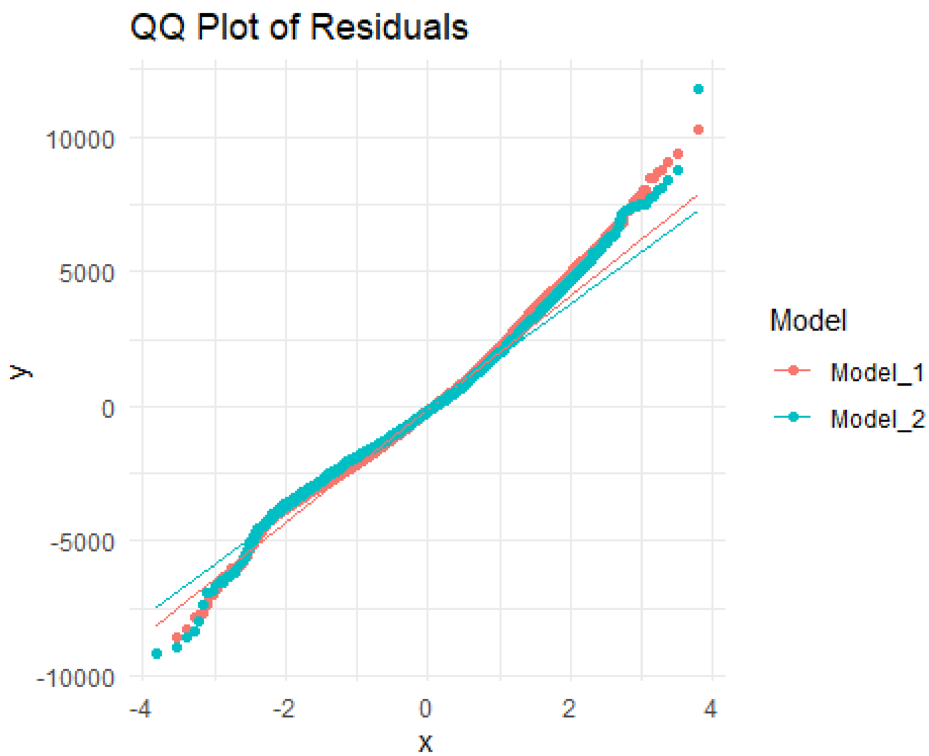



```r
# Set up the plotting area
par(mfrow = c(1, 2))

# Predicted vs Actual values for model 1(distance based model)
plot(trainData_trimmed_1$Priceperbuildingarea, predict(model_1),
     xlab = "Actual Priceperbuildingarea", ylab = "Predicted
Priceperbuildingarea",
     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$Priceperbuildingarea, predict(model_2),
     xlab = "Actual Priceperbuildingarea", ylab = "Predicted
Priceperbuildingarea",
     main = "Model 2: Predicted vs Actual")
abline(0, 1, col = "red")  # Add a reference line with slope = 1
```

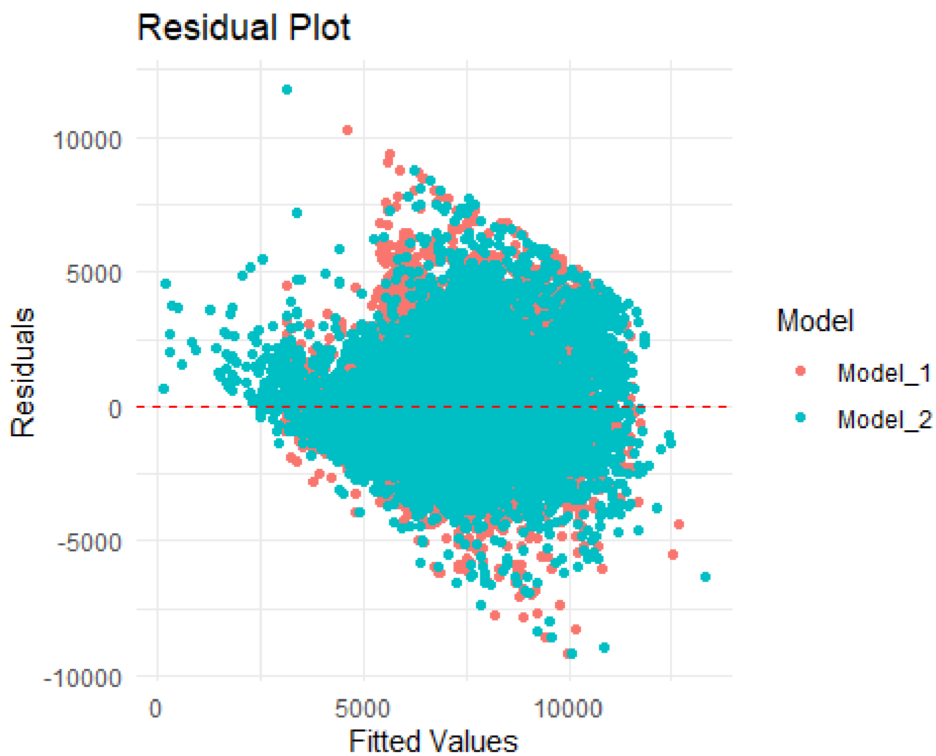



```
# Combine data frames for both models
combined_data <- rbind(
  data.frame(Model = "Model_1", residuals = residuals_1, fitted_values
= fitted_values_1),
  data.frame(Model = "Model_2", residuals = residuals_2, fitted_values
= fitted_values_2)
)

# 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()
```

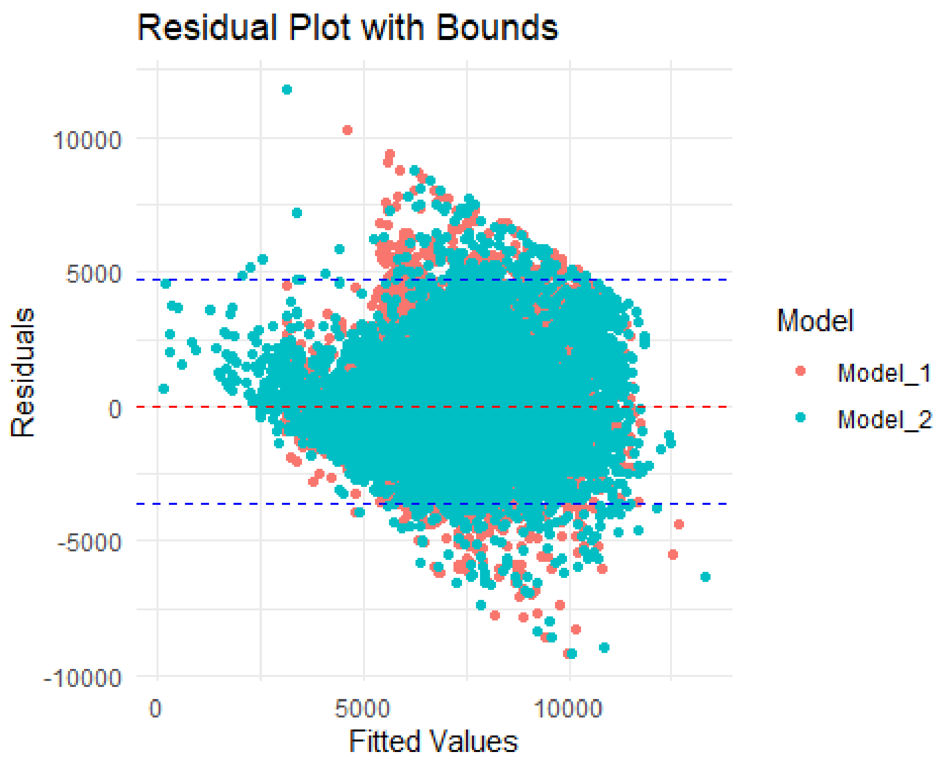
```r
# Show combined plots
print(qqplot_combined)
```




```r
print(residual_plot_combined)
```

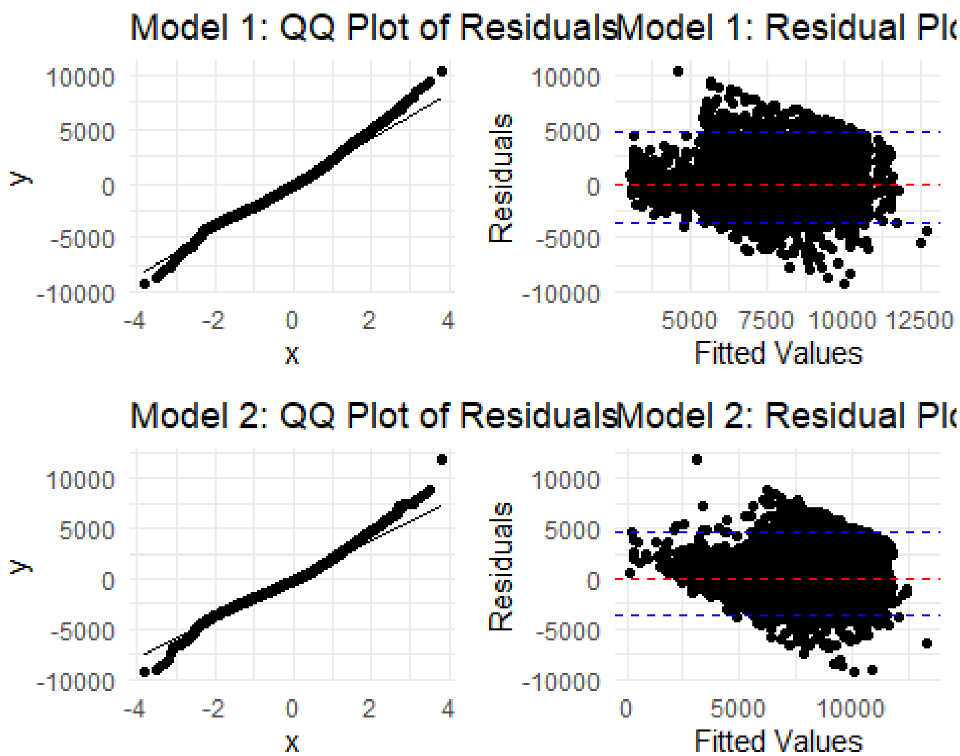



```r
# Calculate lower and upper bounds for residuals (e.g., 95% confidence
interval)
lower_bound <- quantile(combined_data$residuals, 0.025)
upper_bound <- quantile(combined_data$residuals, 0.975)

# 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 = Priceperbuildingarea ~ ., data = trainData_trimmed_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9142.2 -1537.9  -169.2  1298.5 10299.7
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     5354.0584    69.6539  76.867   <2e-16 ***
## Class2            26.4614    72.5400   0.365    0.715
## Class3           917.8225    79.4400  11.554   <2e-16 ***
## Class4         -2329.5063   114.2920 -20.382   <2e-16 ***
## YearsSinceBuilt   38.3039     0.7665  49.972   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2211 on 6827 degrees of freedom
## Multiple R-squared:  0.3621, Adjusted R-squared:  0.3618
## F-statistic: 968.9 on 4 and 6827 DF,  p-value: < 2.2e-16

summary(model_2)
```

```
##
## Call:
## lm(formula = Priceperbuildingarea ~ ., data = trainData_trimmed_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9173.5 -1389.2  -185.5  1214.2 11771.7
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          7594.6911    67.6008  112.35   <2e-16 ***
## distance_from_center -146.4118     3.3025  -44.33   <2e-16 ***
## YearsSinceBuilt        31.2631     0.7414   42.17   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2070 on 6829 degrees of freedom
## Multiple R-squared:  0.441,  Adjusted R-squared:  0.4408
## F-statistic:  2693 on 2 and 6829 DF,  p-value: < 2.2e-16
```

```r
# 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:
```

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

```
## R-squared: 0.3621294
```

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

```
## Adjusted R-squared: 0.3617557
```

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

```
## Residual Standard Error: 2211.019
```

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

```
## F-statistic: 968.9494
cat("Model 2 Metrics:\n")
## Model 2 Metrics:
cat("R-squared:", r_squared_model_2, "\n")
## R-squared: 0.4409704
cat("Adjusted R-squared:", adj_r_squared_model_2, "\n")
## Adjusted R-squared: 0.4408067
cat("Residual Standard Error:", residual_std_error_model_2, "\n")
## Residual Standard Error: 2069.569
cat("F-statistic:", f_statistic_model_2, "\n")
## F-statistic: 2693.406
```

Step 15: Final comment

Step 16: Business application