House Price Prediction Project - GLM

Yohan

Contents

Loading Library

Library used in this study case are as follows.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## Warning: package 'dplyr' was built under R version 4.1.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret) # for splitting train-test set
```

```
## Warning: package 'caret' was built under R version 4.1.2

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(ggpubr) # for arranging ggplot
```

```
## Warning: package 'ggpubr' was built under R version 4.1.2
```

```
library(car) # for calculating vif
```

```
## Warning: package 'car' was built under R version 4.1.2
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 4.1.2
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

Data Description

Data used in this study case are house sales data.

Data Loading and Prerequisites

Data saved in the CSV files will be loaded using the function `read.csv`. A seed will also be set for the project so that we will obtain the same results in each run.

```
rm(list=ls())   # clean up workspace
set.seed(1)
sales <- read.csv('salesData19.csv')
head(sales)
```

```
##   ID City Province     SALEDT  Price   SQFT RMTOT RMBED BATH FLR1AREA BSMT
## 1  1   54        2 2017-05-12 893000  22100     5     5    4     2130    7
## 2  2   54        2 2013-07-05 209000   3276     2     2    1      738    7
## 3  3   33        3 2016-05-11 165000  10000     3     3    1     1293    7
## 4  4   54        2 2013-06-20 449900  83876     5     5    3     1781    7
## 5  5   54        2 2017-02-22 396764  69019     3     3    2     1788    7
## 6  6   54        2 2014-06-30 533754 105484     4     4    3     1891    5
##   HEATSYS ATTIC SFLA GRADE SALE_YEAR SALE_MONTH EFF_AGE Style
## 1       5     1 4829     B        -1          5      11     U
## 2       3     1  738     C        -5          7      50     U
## 3       3     1 1293     C        -2          5      35     U
## 4       5     1 3470     C        -5          6      11     U
## 5       5     0 2326     C        -1          2       1     U
## 6       5     0 4510     B        -4          6       2     U
```

Data Cleanup

First, the data set will be cleaned.

```
# Checking for missing values & removing them
table(is.na(sales))
```

```
##
##  FALSE
## 570266
```

```
sales = na.omit(sales)

# Data summary
summary(sales)
```

```
##        ID              City          Province        SALEDT
##  Min.   :    1   Min.   :20.00   Min.   :1.000   Length:30014
##  1st Qu.: 7504   1st Qu.:54.00   1st Qu.:2.000   Class :character
##  Median :15008   Median :54.00   Median :2.000   Mode  :character
##  Mean   :15008   Mean   :56.02   Mean   :2.022
##  3rd Qu.:22511   3rd Qu.:65.00   3rd Qu.:2.000
##  Max.   :30014   Max.   :98.00   Max.   :3.000
##      Price              SQFT               RMTOT            RMBED
##  Min.   :   1200   Min.   :0.000e+00   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 133000   1st Qu.:6.022e+03   1st Qu.: 3.000   1st Qu.: 3.000
##  Median : 205000   Median :1.145e+04   Median : 3.000   Median : 3.000
##  Mean   : 229836   Mean   :2.308e+05   Mean   : 3.234   Mean   : 3.193
##  3rd Qu.: 291500   3rd Qu.:4.066e+04   3rd Qu.: 4.000   3rd Qu.: 4.000
##  Max.   :5000000   Max.   :2.123e+09   Max.   :18.000   Max.   :13.000
##       BATH           FLR1AREA          BSMT           HEATSYS
##  Min.   : 1.000   Min.   : 192   Min.   :1.000   Min.   :1.000
##  1st Qu.: 1.000   1st Qu.: 750   1st Qu.:6.000   1st Qu.:1.000
##  Median : 1.000   Median :1008   Median :7.000   Median :2.000
##  Mean   : 1.637   Mean   :1081   Mean   :5.963   Mean   :2.446
##  3rd Qu.: 2.000   3rd Qu.:1286   3rd Qu.:7.000   3rd Qu.:3.000
##  Max.   :12.000   Max.   :5366   Max.   :7.000   Max.   :6.000
##      ATTIC             SFLA          GRADE             SALE_YEAR
##  Min.   :0.0000   Min.   :  192   Length:30014       Min.   :-5.000
##  1st Qu.:1.0000   1st Qu.: 1204   Class :character   1st Qu.:-4.000
##  Median :1.0000   Median : 1610   Mode  :character   Median :-2.000
##  Mean   :0.8773   Mean   : 1711                      Mean   :-2.765
##  3rd Qu.:1.0000   3rd Qu.: 2070                      3rd Qu.:-1.000
##  Max.   :1.0000   Max.   :10784                      Max.   :-1.000
##    SALE_MONTH        EFF_AGE          Style
##  Min.   : 1.000   Min.   :  0.00   Length:30014
##  1st Qu.: 5.000   1st Qu.: 15.00   Class :character
##  Median : 7.000   Median : 29.00   Mode  :character
##  Mean   : 7.029   Mean   : 51.98
##  3rd Qu.: 9.000   3rd Qu.: 40.00
##  Max.   :12.000   Max.   :468.00
```

```
# Dropping ID variable since we have the same values in the data frame's index
sales <- subset(sales, select = -c(ID))

# Changing SALEDT variable into date format in R and renaming the column
sales$SALEDT <- as.Date(sales$SALEDT)
names(sales)[names(sales) == 'SALEDT'] <- 'DATE'

# Formatting categorical variables as factors
categorical <- c('City',
                 'Province',
                 'BSMT',
                 'HEATSYS',
                 'ATTIC',
                 'GRADE',
                 'Style')
sales[, categorical] = lapply(sales[, categorical], factor)
```

Exploratory Data Analysis

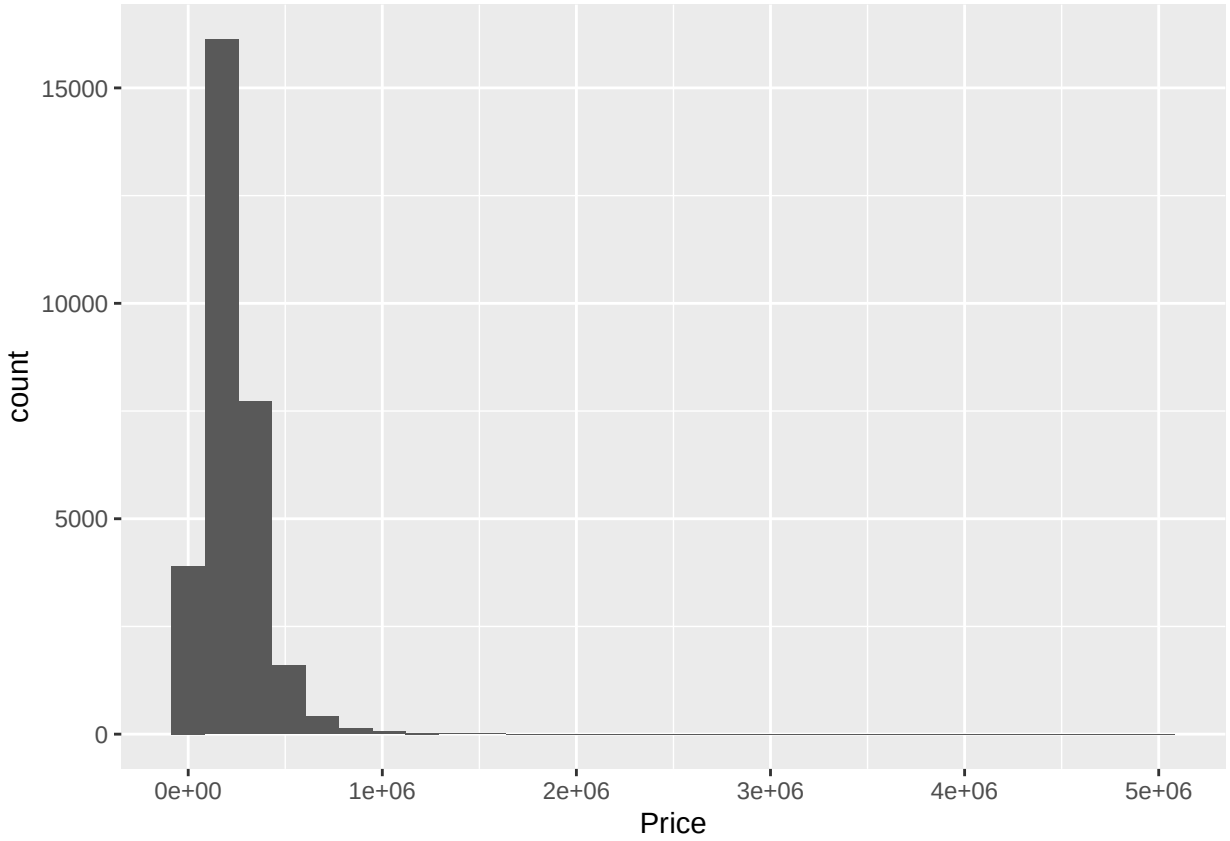
We shall now explore the data set. The dependent variable in this study case is `Price`. We will analyze its distribution and check for outliers in the data.

```
summary(sales$Price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1200  133000  205000  229836  291500 5000000
```

```
ggplot(sales) + geom_histogram(aes(x = Price))
```

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




```
ks.test(sales$Price, "pnorm")
```

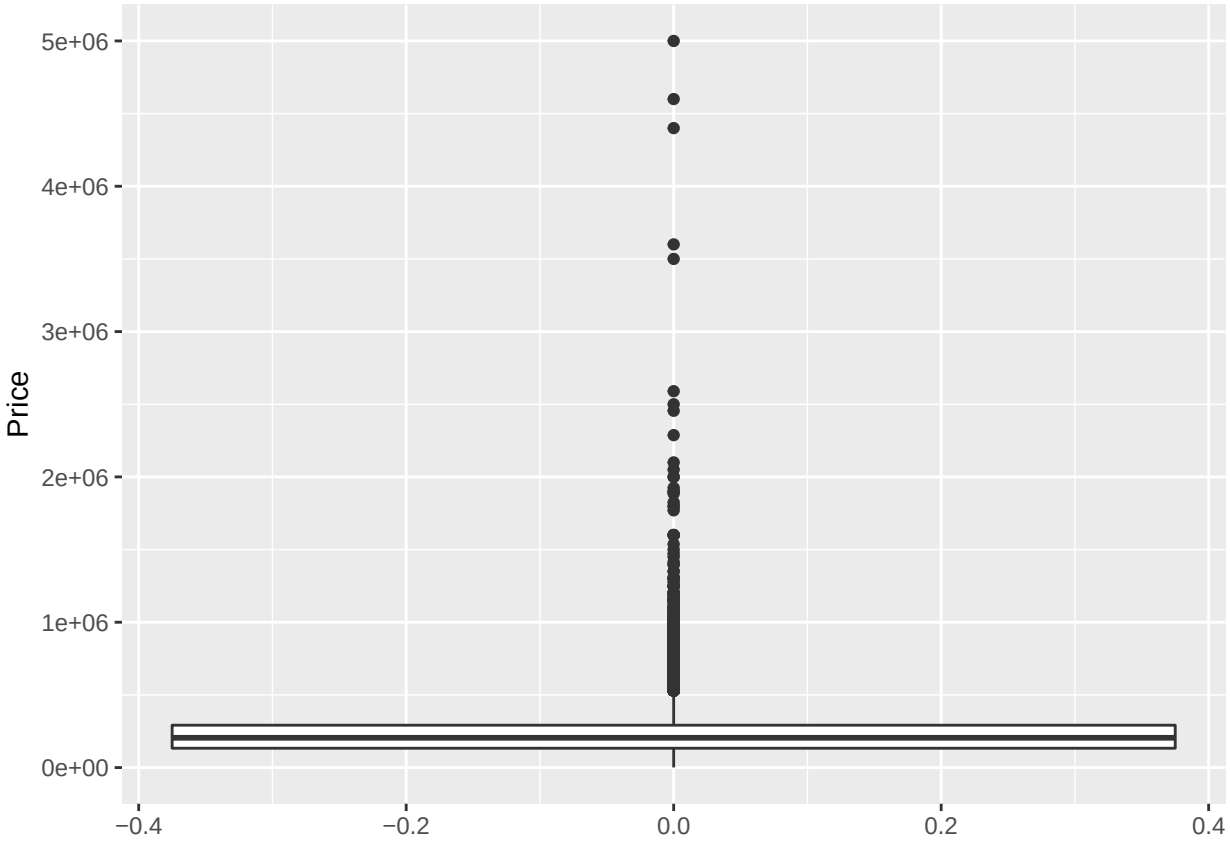
```
## Warning in ks.test(sales$Price, "pnorm"): ties should not be present for the
## Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sales$Price
## D = 1, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

From the histogram, we can see that the data is strongly skewed and not normally distributed. With a p-value of 2.2*10^(-6), we also obtain the same conclusion that the data is not from a normal distribution from the Kolmogorov-Smirnov test.

Now, we are going to analyze outliers that might exist in the dependent variable.

```
ggplot(sales) + geom_boxplot(aes(y = Price))
```



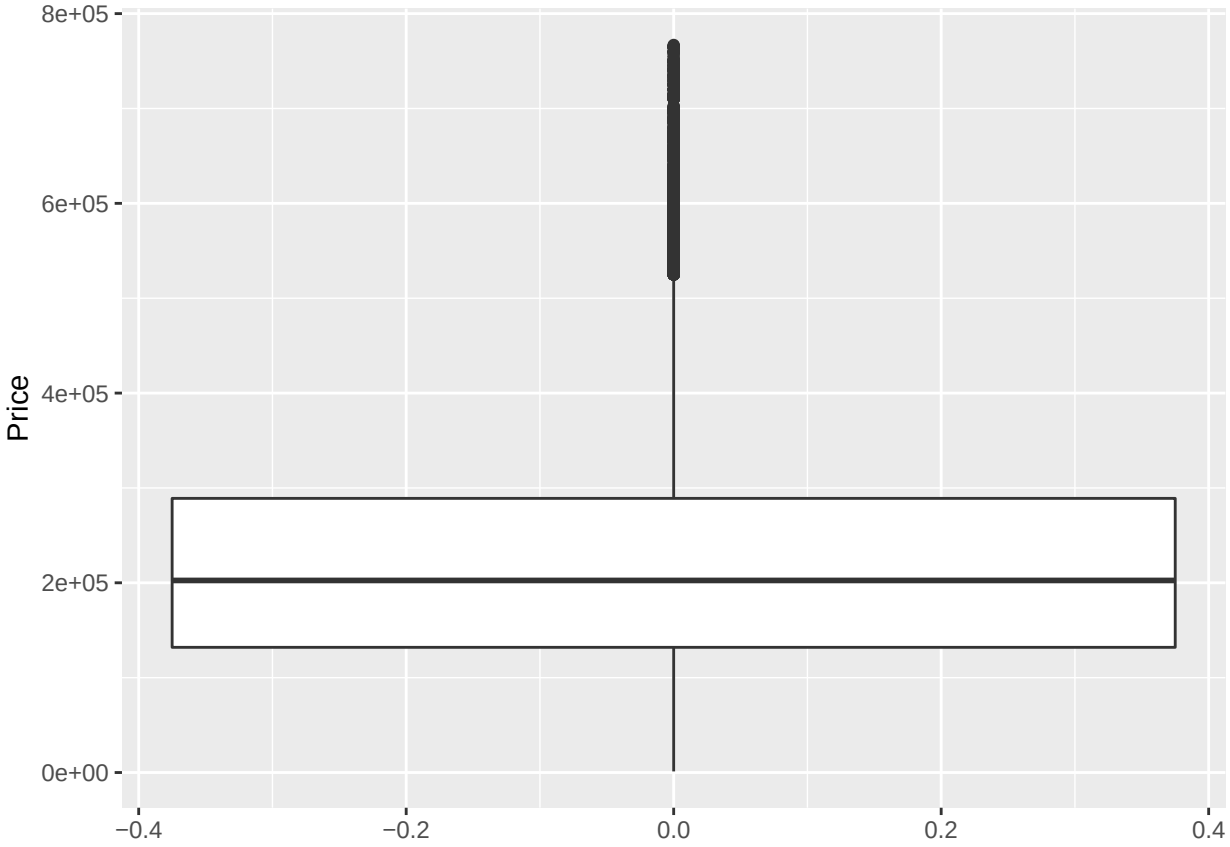
Here, we can conclude that we have outliers present in the response variable. This might be the reason why the distribution of the data is strongly skewed. As these outliers might affect our models, we will remove them. An entry of the data is considered as an outlier with the following formula: `outliers = 3*Interquartile Range / 1.5`.

```
# Total number of outliers:
paste("outliers:", length(boxplot.stats(sales$Price, coef = 3)$out))
```

```
## [1] "outliers: 273"
```

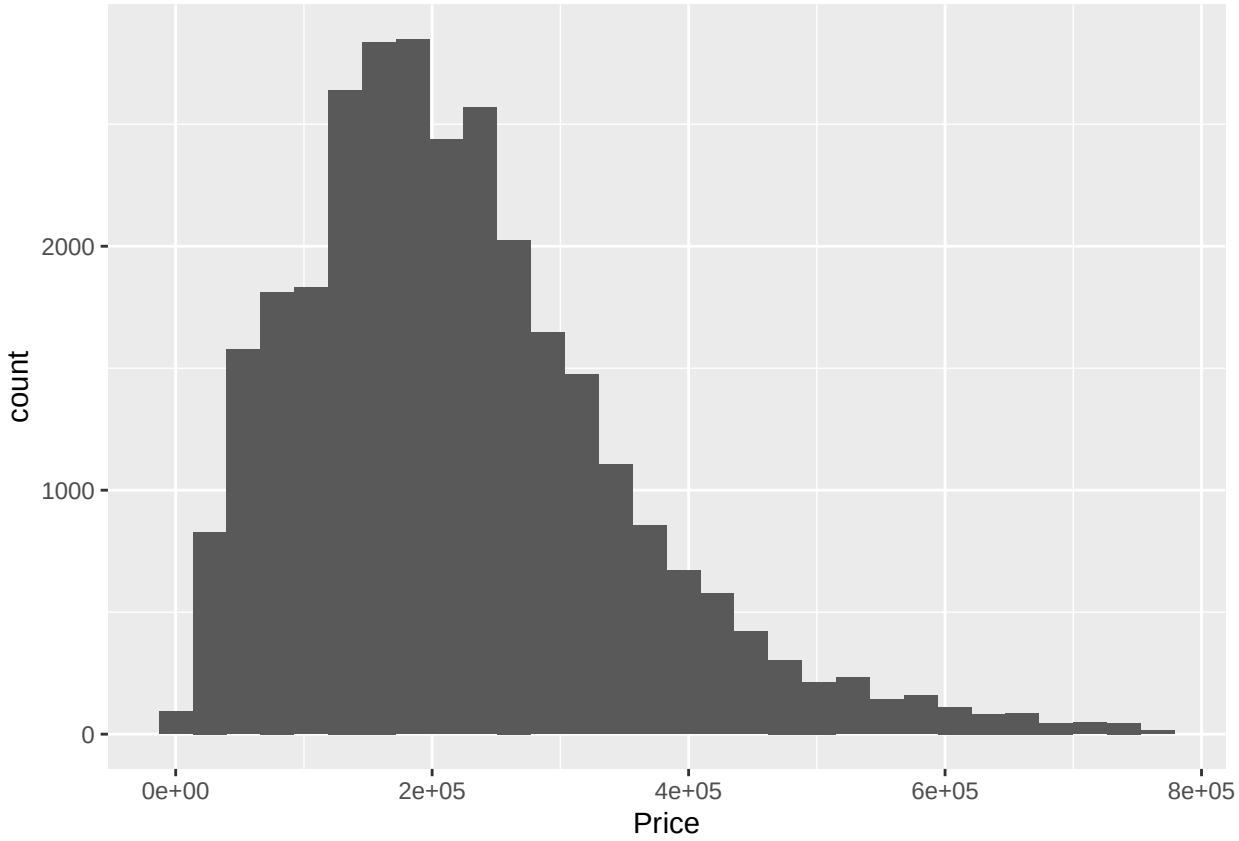
```
out = boxplot.stats(sales$Price, coef = 3)$out
out_key = which(sales$Price %in% c(out))
sales = sales[-out_key, ]

# Checking the boxplot of the data with removed outliers
ggplot(sales) + geom_boxplot(aes(y = Price))
```




```
ggplot(sales) + geom_histogram(aes(x = Price))
```

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



We can see that the boxplot is now slightly better and the histogram of the prices is also less skewed.

After analyzing the response variable, we can now analyze the independent variables, which include `SQFT, FLR1AREA, SFLA, EFF_AGE, GRADE`.

```
table(sales$GRADE)
```

```
##
##     A     B     C     D     E
##    44  2530 23248  3098   821
```

```
summary(sales$SQFT)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.000e+00 6.018e+03 1.140e+04 2.325e+05 4.060e+04 2.123e+09
```

```
summary(sales$FLR1AREA)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     192     750    1008    1074    1279    5366
```

```
summary(sales$SFLA)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     192    1200    1602    1691    2055    8580
```

```
min(sales$SQFT)
```

```
## [1] 0
```

```
# Since it is impossible to have a house with 0 square feet, we will remove them.
sales = sales[sales$SQFT != 0,]
```

We will now see the correlation between the dependent variable and the numeric independent variables.

```
cor(sales$Price, sales$SQFT)
```

```
## [1] -0.01112671
```

```
cor(sales$Price, sales$FLR1AREA)
```

```
## [1] 0.4389079
```

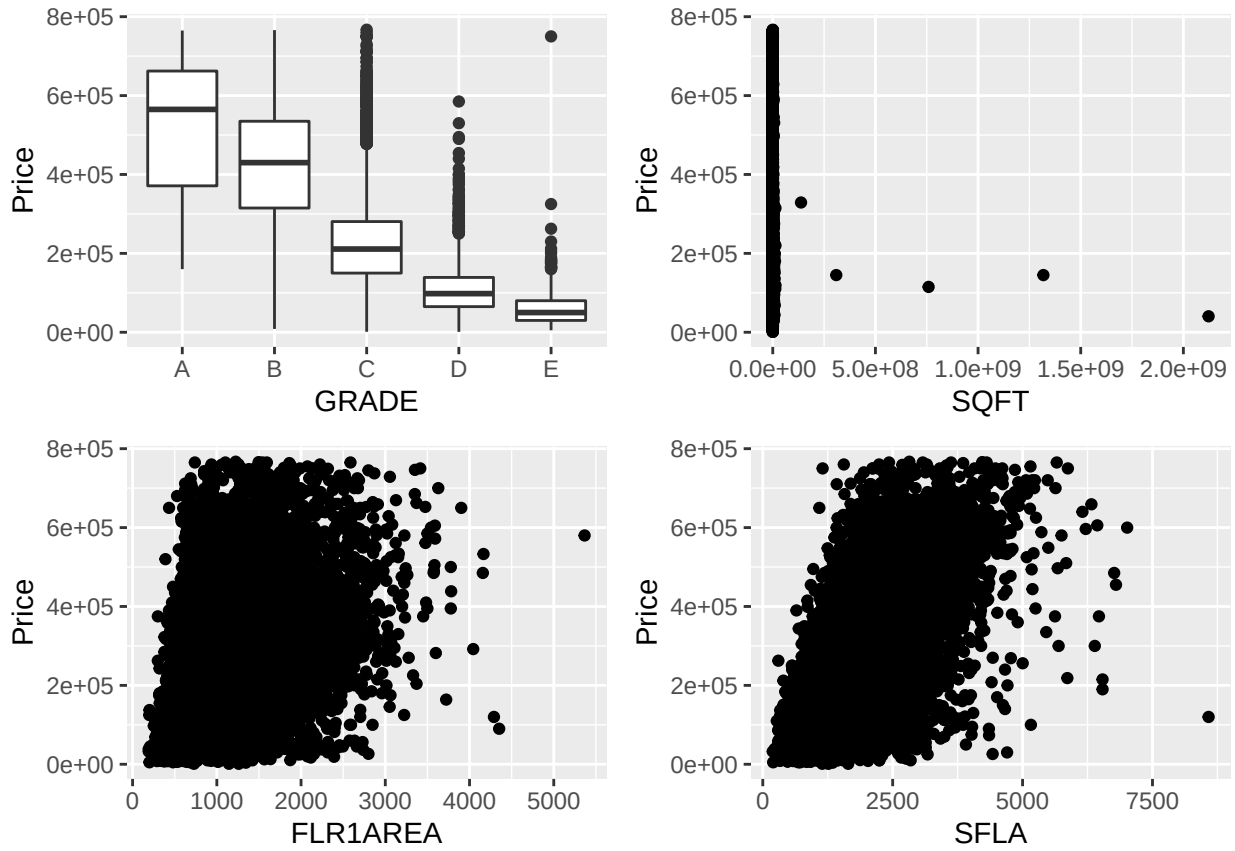
```
cor(sales$Price, sales$SFLA)
```

```
## [1] 0.7160272
```

We can see that the relationship between `Price` and `SQFT` is not significant, while the results also show that there is a moderate correlation between `Price` and `FLR1AREA`. On the other hand, we can see that the correlation of the `SFLA` variable is significant. Therefore, we can assume that `SFLA` will be significant in our models as well.

We will now compare the correlation between the variables with scatterplots and box plot for the categorical independent variable.

```
ggarrange(
  ggplot(sales) + geom_boxplot(aes(GRADE, Price)),
  ggplot(sales) + geom_point(aes(SQFT, Price)),
  ggplot(sales) + geom_point(aes(FLR1AREA, Price)),
  ggplot(sales) + geom_point(aes(SFLA, Price)))
```



From the plots, we can conclude the same conclusion as before based on the correlation coefficients, such as the relationship between `SQFT` and `Price` which shows almost no correlation between the two. Additionally, we can see that the average price of the houses decreases with the `GRADE`. We can also observe the existence of some outliers in each category, one of them being a data point that is far on top in Category 'E'.

Train Test Splitting

We are now ready to split the data set into train and test set. The splitting will be done using `CreateDataPartition` function from the `caret` package. The train-test split ratio will be 80:20 and stratified splitting will be done based on `GRADE` as it is the categorical independent variable which will be modeled in the GLM Model.

```
train.idx <- createDataPartition(y = sales$GRADE, p = 0.8, list = FALSE)
train <- sales[train.idx, ]
test <- sales[-train.idx, ]

# Checking the proportion of the GRADE categorical variable
# in the train and test set in comparison to the original data set
rbind("Data Set" = table(sales$GRADE),
      "Train" = table(train$GRADE),
      "Test" = table(test$GRADE))
```

```
##            A    B     C    D   E
## Data Set 44 2474 22832 3041 805
```

```
## Train    36 1980 18266 2433 644
## Test      8  494  4566  608 161
```

```
rbind("Data Set" = prop.table(table(sales$GRADE)),
      "Train" = prop.table(table(train$GRADE)),
      "Test" = prop.table(table(test$GRADE)))
```

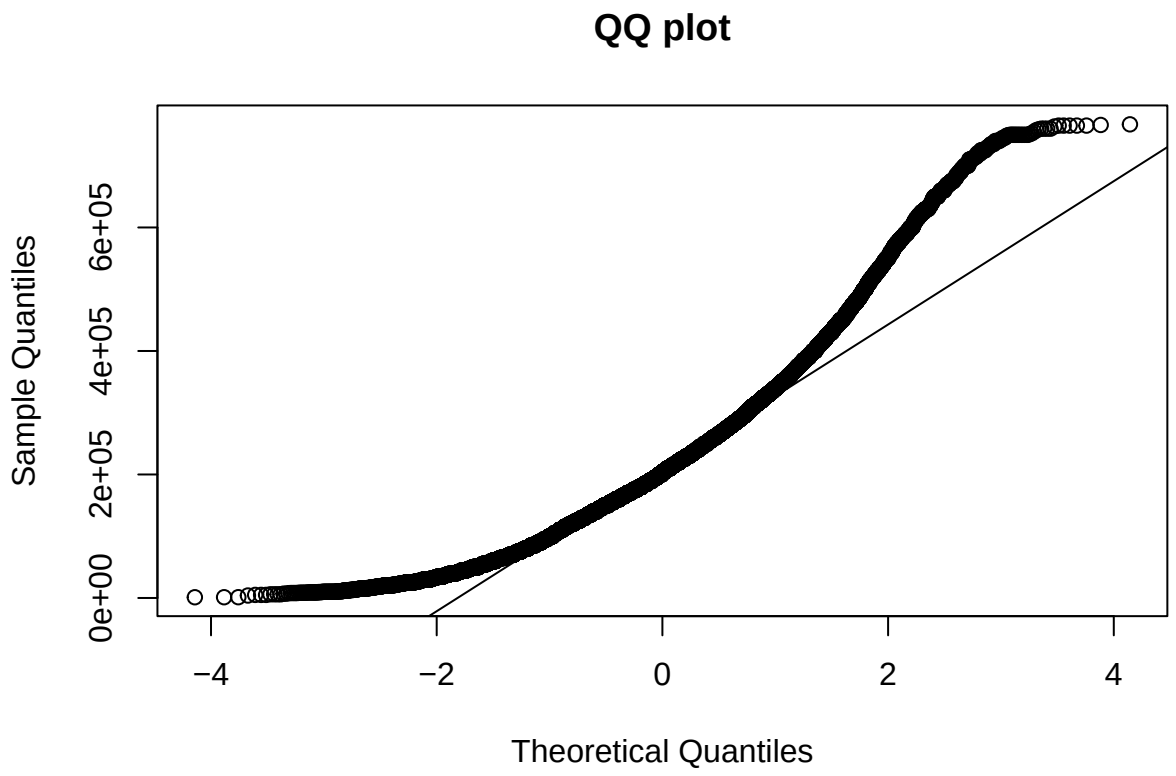
```
##                    A          B         C         D          E
## Data Set 0.001507056 0.08473764 0.7820249 0.1041581 0.02757227
## Train    0.001541162 0.08476390 0.7819684 0.1041569 0.02756967
## Test     0.001370567 0.08463252 0.7822512 0.1041631 0.02758266
```

We can see that all categories in `GRADE` exist in both train and test set with a similar proportion to the original data.

Generalized Linear Model (GLM)

Before we start our GLM modelling, we must first choose a distribution that is suitable to the dependent variable.

```
# Checking the distribution of the dependent variable Price by plotting its histogram
qqnorm(sales$Price,main="QQ plot")
qqline(sales$Price)
```



In addition to the previous normality test with Kolmogorov-Smirnov, we can see through the QQ plot that the data is skewed and is not normally distributed. Therefore, we will attempt to create models based on other distributions, other than the normal/gaussian distribution.

GLM - Choosing Models

There are a couple of family distributions provided in the R distribution, which include the following.

1. Gaussian
2. Gamma
3. Binomial
4. Poisson
5. Inverse Gaussian

The distributions which we will use is the gaussian, gamma, and inverse gaussian family distributions.

We can still use the gaussian distribution as it is the usual multiple linear regression model and that the outcome variable is also continuous. Even though our dependent variable is not normally distributed, fitting a normal multiple linear regression is still useful for comparison with other models.

Then, since the outcome, `Price`, is skewed and always positive, it can be modeled using the gamma distribution. The values are always positive, not only because of the given data, but the sale price of a house is not negative.

As previously mentioned, since the `Price` data is positive, could be assumed as continuous, and has a positively skewed distribution, then the data can be fit into the inverse gaussian distribution which is a family of distribution that includes positive and continuous values with a positively skewed distribution.

On the other hand, the binomial distribution glm is unsuitable since our outcome is not categorical nor a binary outcome. In other words, the value of Prices must be between 0 or 1. We also do not attempt to fit the data into the poisson distribution glm as it is suited for count values; prices are not count outcomes and not discrete.

Modelling Preparation

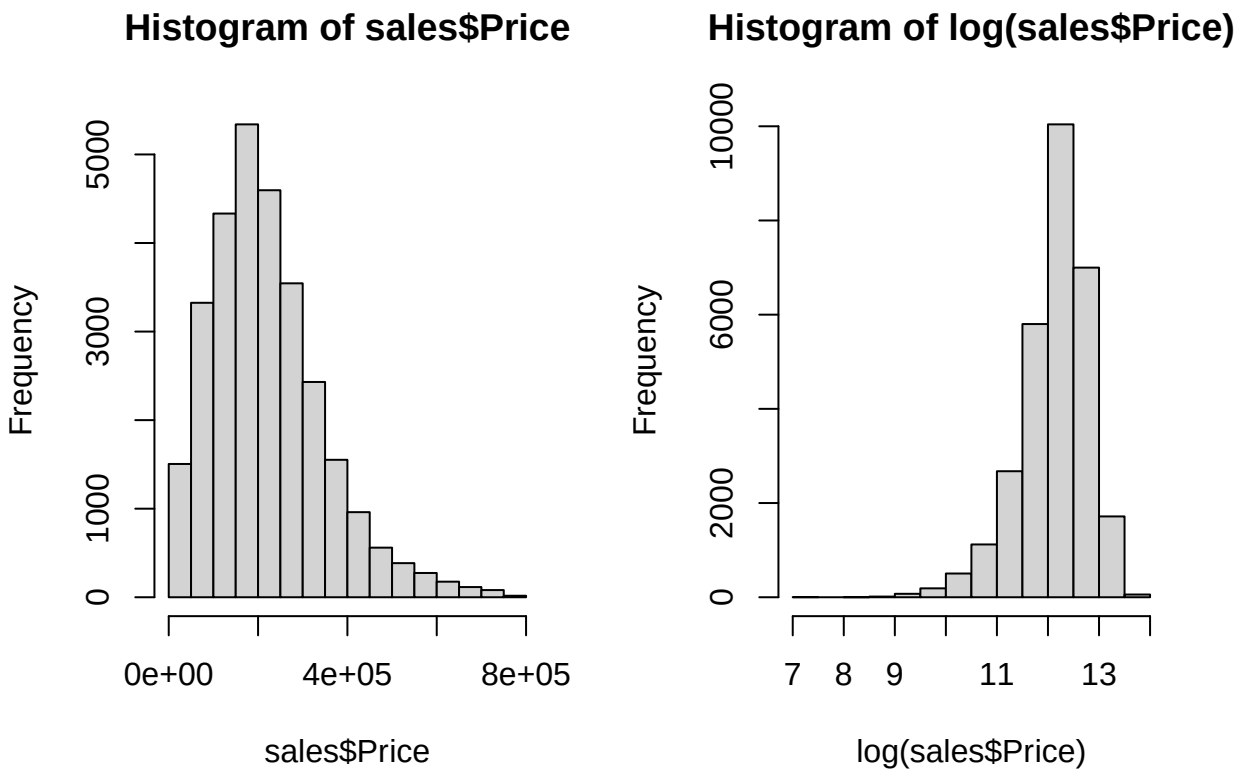
We will now create GLM models with different distributions and find the most suitable model for our data. Once again, the independent variables are as follows:

Independent variables: SQFT, FLR1AREA, SFLA, EFF_AGE, GRADE

```
# Preparing outcome and independent variables for the models
# We also transform the outcome with logarithm transformation.
outcome <- "log(Price)"
variables <- c("SQFT", "FLR1AREA", "SFLA", "EFF_AGE", "GRADE")
f <- as.formula(paste(outcome,
                      paste(variables, collapse = "+"),
                      sep = "~"))
f
```

```
## log(Price) ~ SQFT + FLR1AREA + SFLA + EFF_AGE + GRADE
```

```
par(mfrow=c(1,2))
hist(sales$Price)
hist(log(sales$Price))
```



By transforming the response variable with the logarithm function, we can see that the range of data values become smaller and the distribution of the data becomes less skewed, although it does not completely resemble a normal distribution. This transformation allows a better outcome during model fitting.

Model Evaluation Functions

To analyze and evaluate our models, we will use the following functions.

```
options(scipen=999) # Disable Scientific Notation

# Plot relationships between Log Sale Price, Prediction, and Residuals
# Function also returns a bin which contains indices of extreme residual data
eval.train_init <- function(model, train, outliers){
  zresid = data.frame(x=rstandard(model))
  title = ifelse(outliers == TRUE, "Outliers not Removed", "Outliers Removed")

  print(ggarrange(
    ggplot() + geom_point(aes(x=model$fitted.values, y=log(train$Price))) +
      geom_abline(aes(intercept = 0, slope = 1), colour = "blue") +
      ggtitle(paste("Log SalePrice vs Prediction - Training Set,", title)) +
      theme(plot.title = element_text(hjust = 0.5)) +
      labs(x = "Prediction on Train Data", y ="Log Sales Price"),
    ggplot() + geom_point(aes(x=model$fitted.values, y=zresid$x)) +
      geom_abline(aes(intercept = 0, slope = 0), colour = "blue") +
      ggtitle(paste("Residual vs Prediction - Training Set,", title)) +
```

```
      theme(plot.title = element_text(hjust = 0.5)) +
      labs(x = "Prediction on Train Data", y ="Residual"),
    nrow = 2, align = "v"))

  if(outliers == TRUE){
    bin = which(abs(zresid)>3)
    return(bin)
  }
}

# Updates train data by removing residuals in the model.
eval.train_update <- function(model, train, bin){
  if(length(bin)>0) {
    train.outliers = train
    train.outliers$outliers = 0
    train.outliers$outliers[bin] = 1
    train.outliers$pred = model$fitted.values
    train.outliers$pred.dollar = exp(train.outliers$pred)
    train.2 = train[-bin,]
  } else {
    train.2 = train
  }

  return(train.2)
}

# Compare predicted and observed data
# Function returns a list of metrics used for comparison
comp <- function(pred, obs){
  n = length(obs)
  rsq = cor(pred,obs)^2
  mse = sum((pred - obs)^2)/n
  semse = sd((pred - obs)^2) / sqrt(n)
  rmse = sqrt(mse)
  se = sd(pred-obs) / sqrt(n)
  mae = sum(abs(pred-obs))/n
  mape = sum(abs(pred-obs)/obs)/n*100
  return(list("n"=n,"R2"=rsq,"MSE"=mse,"SEMSE"=semse,"RMSE"=rmse,"SE"=se,"MAE"=mae,"MAPE"=mape))
}

# Plot relationships between Log Sale Price and Prediction in the test set
# Function also returns the comp function for predictions in the test set
eval.test <- function(pred, obs){
  print(
    ggplot() + geom_point(aes(x=pred, y=log(obs))) +
        geom_abline(aes(intercept = 0, slope = 1), colour = "blue") +
        ggtitle("Log SalePrice vs Prediction - Testing Set") +
        theme(plot.title = element_text(hjust = 0.5)) +
        labs(x = "Prediction on Test Data", y ="Log Sales Price"))

  comp.test = comp(pred, log(obs))
  print(comp.test)
  return(comp.test)
```

```
}
```

Gaussian Distribution GLM (Normal LM)

```
mod.gaussian <- glm(f, data = train, family = gaussian)
summary(mod.gaussian)
```

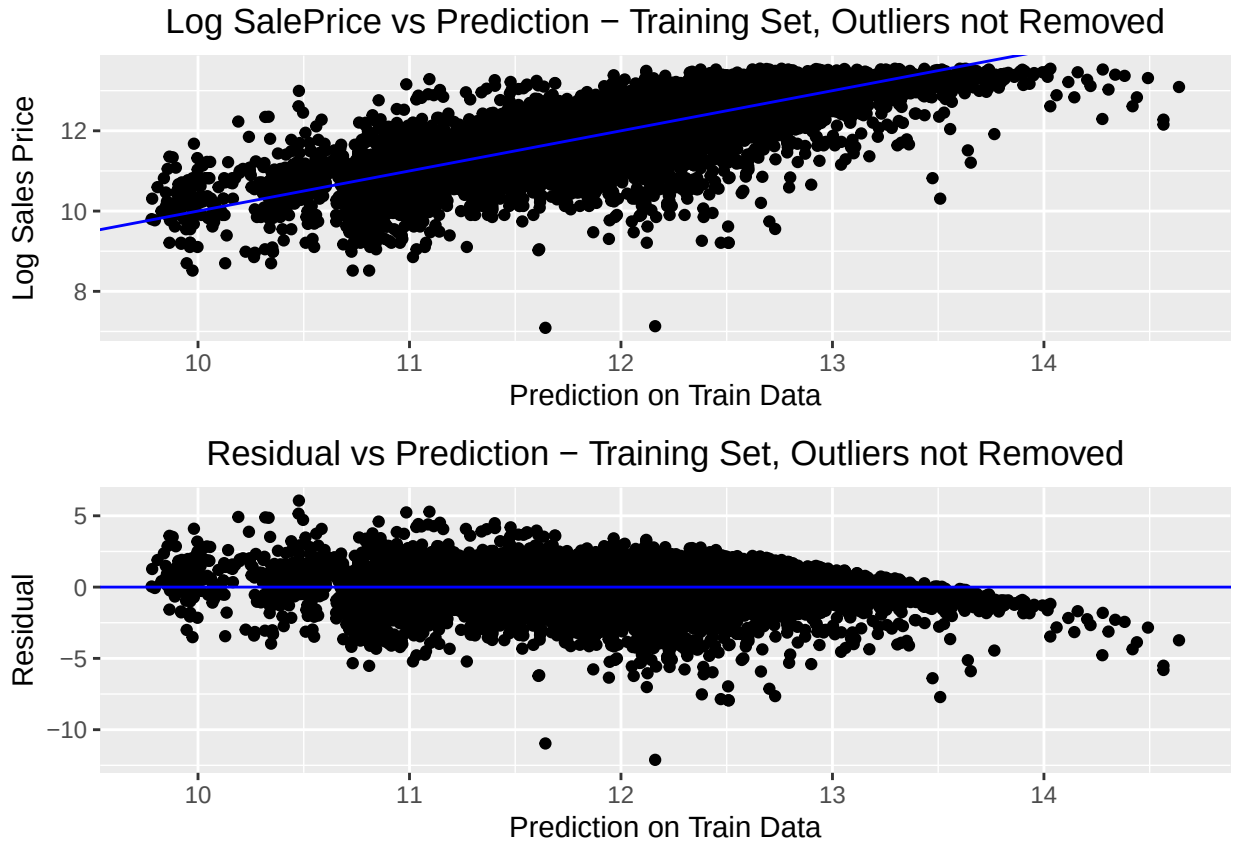
```
##
## Call:
## glm(formula = f, family = gaussian, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.0307  -0.1941   0.0408   0.2347   2.5173
##
## Coefficients:
##                       Estimate        Std. Error t value             Pr(>|t|)
## (Intercept) 11.75136063855805  0.07131559526327 164.780 < 0.0000000000000002
## SQFT        -0.00000000005586  0.00000000015795  -0.354             0.723571
## FLR1AREA     0.00002632304055  0.00000773794097   3.402             0.000671
## SFLA         0.00040475402504  0.00000545359086  74.218 < 0.0000000000000002
## EFF_AGE     -0.00254534095242  0.00002838527893 -89.671 < 0.0000000000000002
## GRADEB       0.10680428937185  0.06998445865396   1.526             0.126995
## GRADEC      -0.17072572427612  0.06980933993040  -2.446             0.014468
## GRADED      -0.54750085582853  0.07064877790138  -7.750  0.00000000000000959
## GRADEE      -0.93169396174532  0.07233037034800 -12.881 < 0.0000000000000002
##
## (Intercept) ***
## SQFT
## FLR1AREA    ***
## SFLA        ***
## EFF_AGE     ***
## GRADEB
## GRADEC      *
## GRADED      ***
## GRADEE      ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1724586)
##
##     Null deviance: 10651.8  on 23358  degrees of freedom
## Residual deviance:  4026.9  on 23350  degrees of freedom
## AIC: 25245
##
## Number of Fisher Scoring iterations: 2
```

Model Evaluation 1 - AIC & Residual Analysis

```
mod.gaussian$aic
```

```
## [1] 25245.23
```

```
bin.gaussian = eval.train_init(mod.gaussian, train, outliers = TRUE)
```




```
train.gaussian = eval.train_update(mod.gaussian, train, bin.gaussian)
```

In the model's summary, that the model received an AIC value of 25245.23, which will be used to compare with the next model with removed outliers in the train set.

From the graph above, we can see from the first graph that the predictions made by the model are spread around the y=x line, which shows that the predictions are close to the actual values. However, we can see that there are a couple of wrong predictions. The second graph highlights these wrong outcomes as we can see that there are some residuals that are far from the predictions. Residuals are the difference between each predicted data and the actual value of the data. Data considered as residuals were evaluated by the `rstandard()` function and values that are greater than 3 will be removed in the following step.

Remodelling With New Train Set

```r
mod.gaussian.final <- glm(f, data = train.gaussian, family = gaussian)
summary(mod.gaussian.final)
```

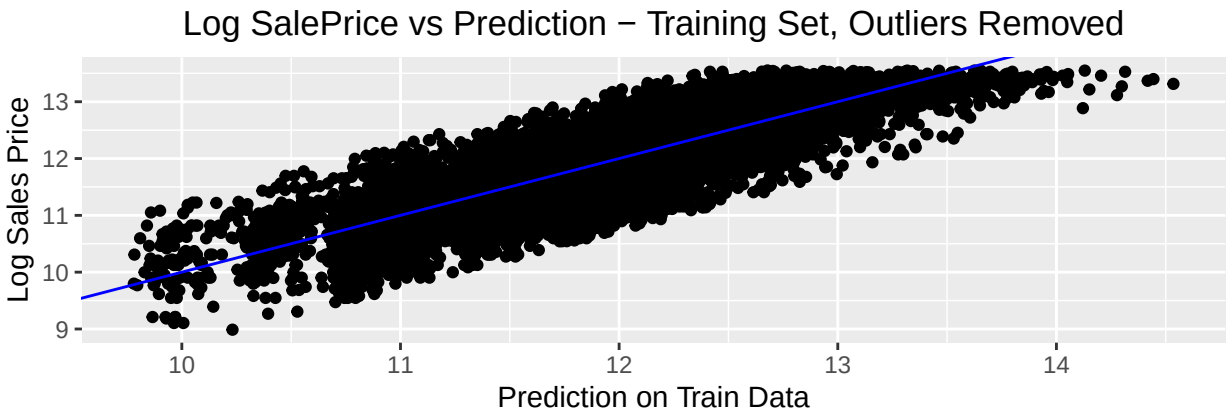
```
##
## Call:
## glm(formula = f, family = gaussian, data = train.gaussian)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.27035  -0.19991   0.02671   0.21786   1.25206
##
## Coefficients:
##                       Estimate        Std. Error  t value             Pr(>|t|)
## (Intercept) 11.79856382080365  0.06333915103924  186.276 < 0.0000000000000002
## SQFT        -0.00000000006489  0.00000000013673   -0.475             0.635095
## FLR1AREA     0.00004510932630  0.00000677097547    6.662      0.0000000000276
## SFLA         0.00040562096804  0.00000479740077   84.550 < 0.0000000000000002
## EFF_AGE     -0.00259335160888  0.00002542581828 -101.997 < 0.0000000000000002
## GRADEB       0.05068061671458  0.06228206236282    0.814             0.415810
## GRADEC      -0.22042117325364  0.06209236514656   -3.550             0.000386
## GRADED      -0.59821388925324  0.06279367386729   -9.527 < 0.0000000000000002
## GRADEE      -0.96278003548929  0.06426982991632  -14.980 < 0.0000000000000002
##
## (Intercept) ***
## SQFT
## FLR1AREA    ***
## SFLA        ***
## EFF_AGE     ***
## GRADEB
## GRADEC      ***
## GRADED      ***
## GRADEE      ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1292314)
##
##     Null deviance: 9433.4  on 23015  degrees of freedom
## Residual deviance: 2973.2  on 23007  degrees of freedom
## AIC: 18233
##
## Number of Fisher Scoring iterations: 2
```

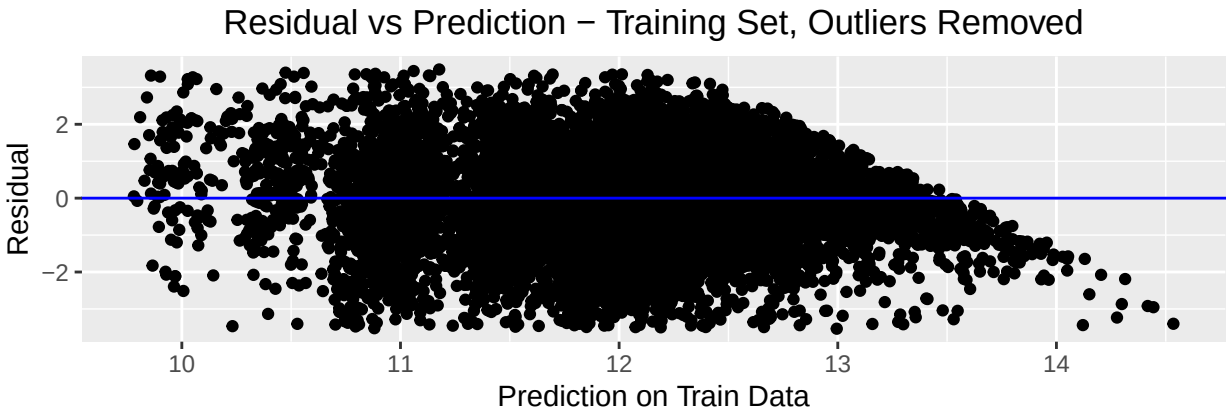
Model Evaluation 2 - New AIC, Residual Analysis, and Error Analysis

```r
data.frame("Outliers Not Removed" = mod.gaussian$aic,
           "Outliers Removed" = mod.gaussian.final$aic)
```

```
##   Outliers.Not.Removed Outliers.Removed
## 1             25245.23         18233.36
```

```
eval.train_init(mod.gaussian.final, train.gaussian, outliers = FALSE)
```





After we have removed the outliers in the training set, we can see that there is a significant decrease in the AIC of the model. This means that the model is now performing better with outliers removed.

From the graphs, we can now see that the predictions are now closer the actual values and the residuals are also closer to 0. It is also worth noting that there are no significant patterns present in the residuals vs prediction plot; the residuals are randomly distributed.

Now, we are going to analyze the error in the predictions of the train set. We will also compare the metrics with the previous model with outliers not removed.

```
merge(stack(comp(mod.gaussian$fitted.values, mod.gaussian$y)),
      stack(comp(mod.gaussian.final$fitted.values, mod.gaussian.final$y)),
      by = "ind", sort = FALSE)
```

```
##     ind        values.x        values.y
## 1     n 23359.000000000 23016.000000000
## 2    R2     0.621949288     0.684820659
## 3   MSE     0.172392133     0.129180822
## 4 SEMSE     0.003217362     0.001427166
## 5  RMSE     0.415201316     0.359417337
## 6    SE     0.002716695     0.002369155
## 7   MAE     0.295183683     0.272584262
## 8  MAPE     2.497062935     2.281900594
```

From the results, we can conclude that out of the 23016 data in the train set with outliers removed, the model receives an RMSE of 0.3594 and an average absolute error of 2.282%. These values show an increase in performance from the previous model with outliers intact. This can be concluded from the lower RMSE and MAPE values.

Multicollinearity

```
vif(mod.gaussian.final)
```

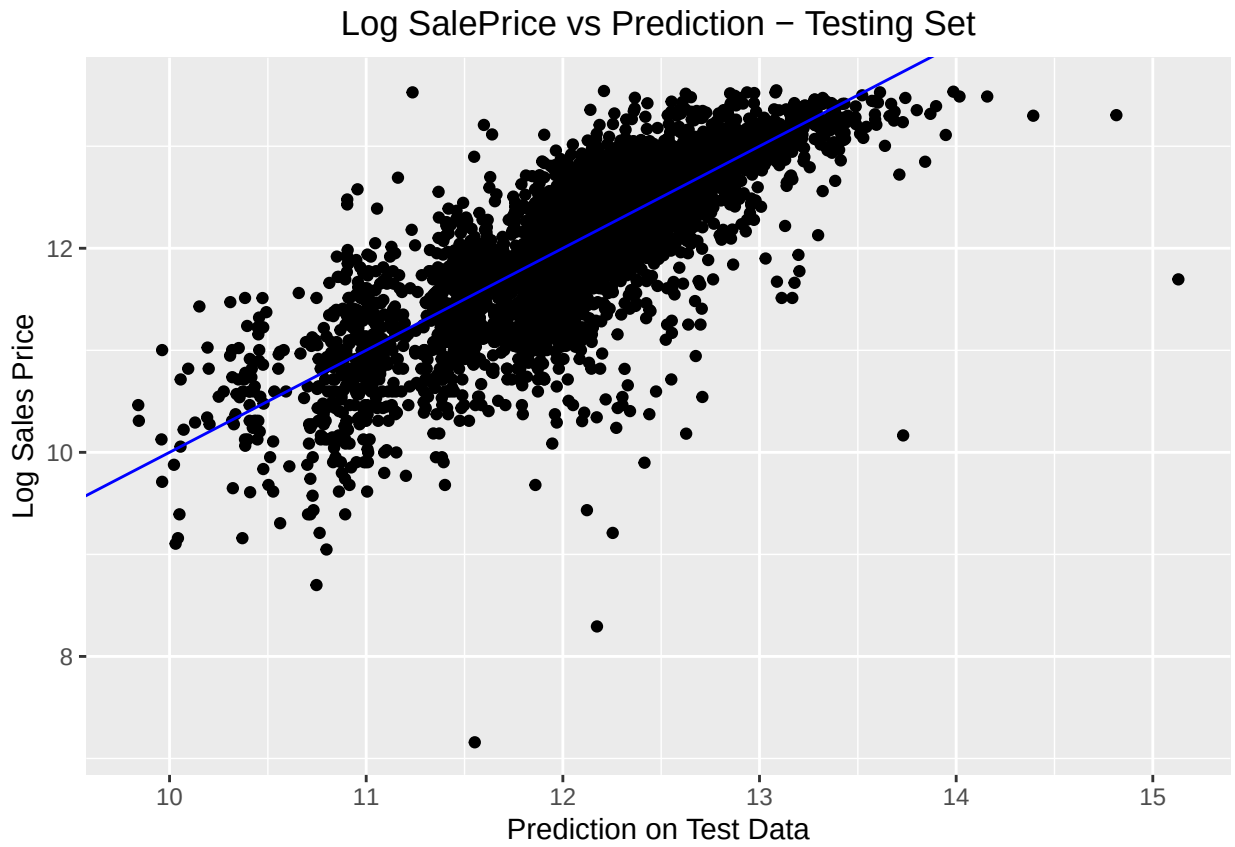
```
##              GVIF Df GVIF^(1/(2*Df))
## SQFT     1.000792  1        1.000396
## FLR1AREA 1.532477  1        1.237933
## SFLA     1.944660  1        1.394511
## EFF_AGE  1.059951  1        1.029539
## GRADE    1.427640  4        1.045508
```

From the VIF scores, we can see that all of them have a relatively small VIF with values below 4. Therefore, we can conclude that each independent variable is not a linear combination of other independent variables. In other words, it is unlikely that there exists a relationship between the independent variables.

Prediction and Error Analysis

```
test.gaussian = test
test.gaussian$prediction = predict(mod.gaussian.final, newdata = test.gaussian, type = "response")
```

```
error.gaussian = eval.test(test.gaussian$prediction, test.gaussian$Price)
```




```
## $n
## [1] 5837
##
## $R2
## [1] 0.6342521
##
## $MSE
## [1] 0.1719461
##
## $SEMSE
## [1] 0.007261306
##
## $RMSE
## [1] 0.4146638
##
## $SE
## [1] 0.005418054
##
## $MAE
## [1] 0.2918277
##
## $MAPE
## [1] 2.480887
```

From the graph, we can see that there are still a lot of predictions that are far away from the y=x line, which means that the model has some inaccurate predictions in the test set.

From the results, we can conclude that out of the 5837 data in the test set, the model successfully predicted the prices with RMSE of 0.4147 and average absolute error of 2.481%. These metrics and the model's AIC will be kept for further comparison with the results from other models.

Gamma Distribution GLM

```
# Comparing 2 Link Functions of Gamma Family
mod.gamma.inv <- glm(f, data = train, family = Gamma)
mod.gamma.ide <- glm(f, data = train, family = Gamma(link = "identity"))

c("Inverse" = mod.gamma.inv$aic, "Identity" = mod.gamma.ide$aic)
```

```
##  Inverse Identity
## 26988.58 26775.78
```

```
# The chosen link function is identity as the fitted model receives a smaller AIC in comparison with th
mod.gamma <- mod.gamma.ide
summary(mod.gamma)
```

```
##
## Call:
## glm(formula = f, family = Gamma(link = "identity"), data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.49011  -0.01595   0.00336   0.01914   0.22102
##
## Coefficients:
##                       Estimate        Std. Error t value             Pr(>|t|)
## (Intercept) 11.71424843831243  0.07818018977929 149.837 < 0.0000000000000002
## SQFT        -0.00000000002383  0.00000000014493  -0.164               0.8694
## FLR1AREA     0.00003173656077  0.00000804100083   3.947      0.00007942057612
## SFLA         0.00041910028308  0.00000571662077  73.313 < 0.0000000000000002
## EFF_AGE     -0.00249059678148  0.00002627889326 -94.776 < 0.0000000000000002
## GRADEB       0.10920612765226  0.07697156366807   1.419               0.1560
## GRADEC      -0.16845133562669  0.07673311658064  -2.195               0.0282
## GRADED      -0.53248634230034  0.07747738473442  -6.873      0.00000000000645
## GRADEE      -0.90637527206309  0.07876253869980 -11.508 < 0.0000000000000002
##
## (Intercept) ***
## SQFT
## FLR1AREA    ***
## SFLA        ***
## EFF_AGE     ***
## GRADEB
## GRADEC      *
## GRADED      ***
## GRADEE      ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

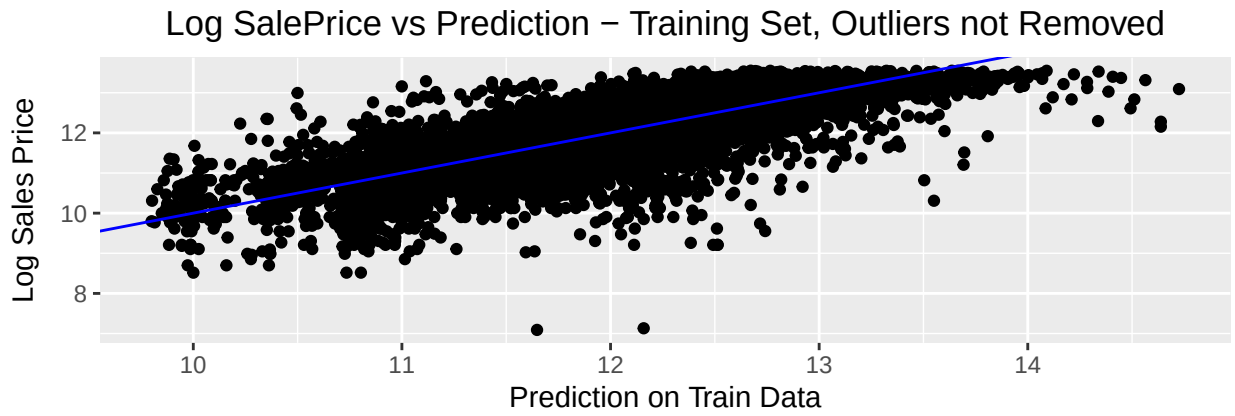
```
##
## (Dispersion parameter for Gamma family taken to be 0.001221869)
##
##     Null deviance: 75.806  on 23358  degrees of freedom
## Residual deviance: 29.341  on 23350  degrees of freedom
## AIC: 26776
##
## Number of Fisher Scoring iterations: 4
```

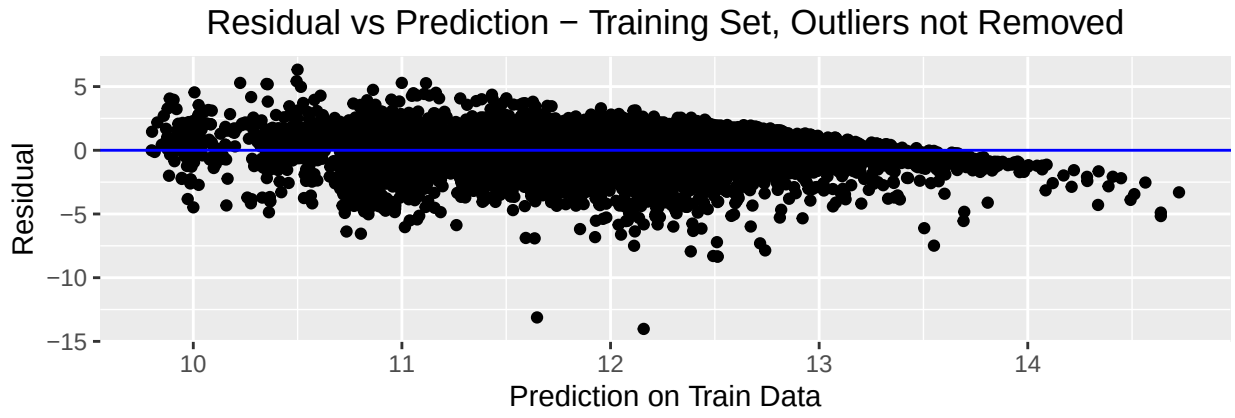
Model Evaluation 1 - AIC & Residual Analysis

```
mod.gamma$aic
```

```
## [1] 26775.78
```

```
bin.gamma = eval.train_init(mod.gamma, train, outliers = TRUE)
```






```
train.gamma = eval.train_update(mod.gamma, train, bin.gamma)
```

In the model's summary, the model received an AIC value of 26775.78, which will be used to compare with the next model with removed outliers in the train set.

From the graph above, we can see that from the first graph, the predictions made by the model are spread around the y=x line, which shows that the predictions are close to the actual values. However, we can see

that there are a couple of wrong predictions. The second graph highlights these wrong outcomes as we can see that there are some residuals that are far from the predictions. Residuals were processed and removed the same way as before in the previous model.

Remodelling With New Train Set

```
mod.gamma.final <- glm(f, data = train.gamma, family = Gamma(link = "identity"))
summary(mod.gamma.final)
```

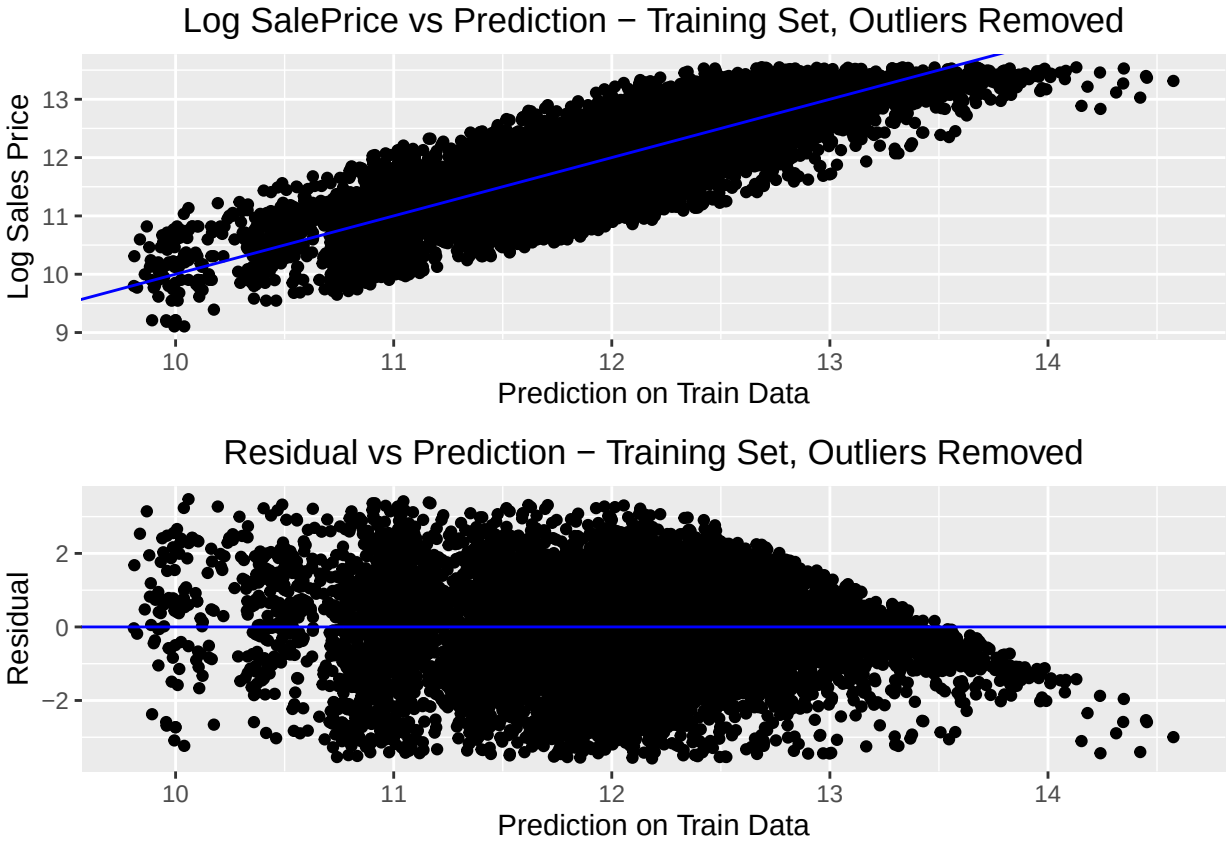
```
##
## Call:
## glm(formula = f, family = Gamma(link = "identity"), data = train.gamma)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.106452  -0.016512   0.002194   0.017653   0.103311
##
## Coefficients:
##                       Estimate        Std. Error  t value             Pr(>|t|)
## (Intercept) 11.74207247665828  0.06754346636108  173.845 < 0.0000000000000002
## SQFT        -0.00000000004068  0.00000000012321   -0.330               0.74130
## FLR1AREA     0.00004816191723  0.00000693885312    6.941       0.000000000004
## SFLA         0.00041382434492  0.00000494753677   83.643 < 0.0000000000000002
## EFF_AGE     -0.00251596771466  0.00002346946010 -107.202 < 0.0000000000000002
## GRADEB       0.08557008207067  0.06652194920436    1.286               0.19834
## GRADEC      -0.18513153378825  0.06630422407248   -2.792               0.00524
## GRADED      -0.55067305144482  0.06694194295563   -8.226 < 0.0000000000000002
## GRADEE      -0.91744773354436  0.06810131843537  -13.472 < 0.0000000000000002
##
## (Intercept) ***
## SQFT
## FLR1AREA    ***
## SFLA        ***
## EFF_AGE     ***
## GRADEB
## GRADEC      **
## GRADED      ***
## GRADEE      ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.000886253)
##
##     Null deviance: 64.386  on 22964  degrees of freedom
## Residual deviance: 20.485  on 22956  degrees of freedom
## AIC: 18569
##
## Number of Fisher Scoring iterations: 4
```

Model Evaluation 2 - New AIC, Residual Analysis, and Error Analysis

```
data.frame("Outliers Not Removed" = mod.gamma$aic,
           "Outliers Removed" = mod.gamma.final$aic)
```

```
##   Outliers.Not.Removed Outliers.Removed
## 1             26775.78         18568.78
```

```
eval.train_init(mod.gamma.final, train.gamma, outliers = FALSE)
```



Just as we have also previously concluded in the previous model, we can see that there is a significant decrease in the AIC of the model after removing the residuals in the training set. This means that the model is now performing better with the outliers removed.

From the graphs, we can now see that the predictions are now closer the actual values and the residuals are also closer to 0 as well. It is also worth noting that there are no significant patterns present in the residuals vs prediction plot; the residuals are randomly distributed.

Now, we are going to analyze the error in the predictions of the train set.

```
merge(stack(comp(mod.gamma$fitted.values, mod.gamma$y)),
      stack(comp(mod.gamma.final$fitted.values, mod.gamma.final$y)),
      by = "ind", sort = FALSE)
```

```
##     ind        values.x        values.y
```

```
## 1     n 23359.000000000 22965.000000000
## 2    R2     0.621689000     0.684529422
## 3   MSE     0.172519106     0.126446538
## 4 SEMSE     0.003222990     0.001375051
## 5  RMSE     0.415354193     0.355593220
## 6    SE     0.002717695     0.002346549
## 7   MAE     0.295274921     0.270727200
## 8  MAPE     2.497114308     2.261930412
```

From the results, we can conclude that out of the 22965 data in the train set with outliers removed, the model's predictions receive an RMSE of 0.3556 and an average absolute error of 2.262%. These values show an increase in performance from the previous model with outliers intact. This can be concluded from the lower RMSE and MAPE values.

Multicollinearity

```
vif(mod.gamma.final)
```

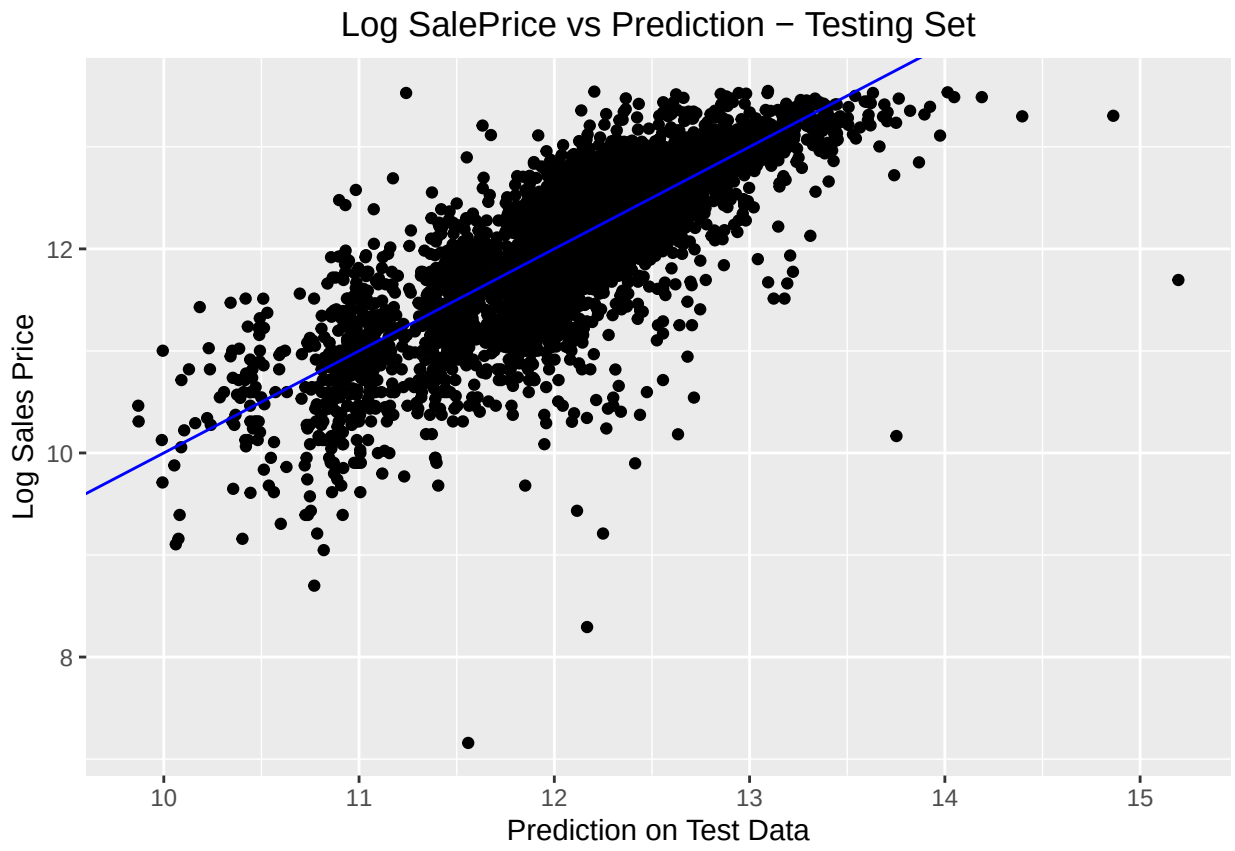
```
##              GVIF Df GVIF^(1/(2*Df))
## SQFT     1.000948  1        1.000474
## FLR1AREA 1.534878  1        1.238902
## SFLA     1.953324  1        1.397614
## EFF_AGE  1.064685  1        1.031836
## GRADE    1.439983  4        1.046634
```

From the VIF scores, we can see that all of them have a relatively small VIF with values below 4. Therefore, we can conclude that each independent variable is not a linear combination of other independent variables. In other words, it is unlikely that there exists a relationship between the independent variables.

Prediction and Error Analysis

```
test.gamma = test
test.gamma$prediction = predict(mod.gamma.final, newdata = test.gamma, type = "response")
```

```
error.gamma = eval.test(test.gamma$prediction, test.gamma$Price)
```




```
## $n
## [1] 5837
##
## $R2
## [1] 0.6340733
##
## $MSE
## [1] 0.1721358
##
## $SEMSE
## [1] 0.007294248
##
## $RMSE
## [1] 0.4148926
##
## $SE
## [1] 0.005419834
##
## $MAE
## [1] 0.2919618
##
## $MAPE
## [1] 2.482133
```

From the results, we can conclude that out of the 5837 data in the test set, the model has predicted with RMSE of 0.4149 and average absolute error of 2.482%. These metrics will be kept for further comparison with the results from other models.

Inverse Gaussian Distribution GLM

```
# Comparing 2 Link Functions of Inverse Gaussian Family
mod.inv.gsn.mu2 <- glm(f, data = train, family = inverse.gaussian)
mod.inv.gsn.ide <- glm(f, data = train, family = inverse.gaussian(link = "identity"))

c("1/mu^2" = mod.inv.gsn.mu2$aic, "Identity" = mod.inv.gsn.ide$aic)
```

```
##   1/mu^2 Identity
## 28046.29 27701.45
```

```
# The chosen link function is identity as the fitted model receives a smaller AIC in comparison with th
mod.inv.gsn <- mod.inv.gsn.ide
summary(mod.inv.gsn)
```

```
##
## Call:
## glm(formula = f, family = inverse.gaussian(link = "identity"),
##     data = train)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.154837  -0.004619   0.000964   0.005456   0.065572
##
## Coefficients:
##                       Estimate        Std. Error t value             Pr(>|t|)
## (Intercept) 11.69506389013096  0.08208525686653 142.475 < 0.0000000000000002
## SQFT        -0.00000000001037  0.00000000013917  -0.075               0.9406
## FLR1AREA     0.00003495357117  0.00000820984715   4.258       0.000020751617
## SFLA         0.00042637570891  0.00000586094554  72.749 < 0.0000000000000002
## EFF_AGE     -0.00246580301233  0.00002535374755 -97.256 < 0.0000000000000002
## GRADEB       0.11076631749992  0.08093895206639   1.369               0.1712
## GRADEC      -0.16723023689074  0.08066752135851  -2.073               0.0382
## GRADED      -0.52470745189549  0.08136924688779  -6.448       0.000000000115
## GRADEE      -0.89342295168902  0.08249600466655 -10.830 < 0.0000000000000002
##
## (Intercept) ***
## SQFT
## FLR1AREA    ***
## SFLA        ***
## EFF_AGE     ***
## GRADEB
## GRADEC      *
## GRADED      ***
## GRADEE      ***
## ---
```

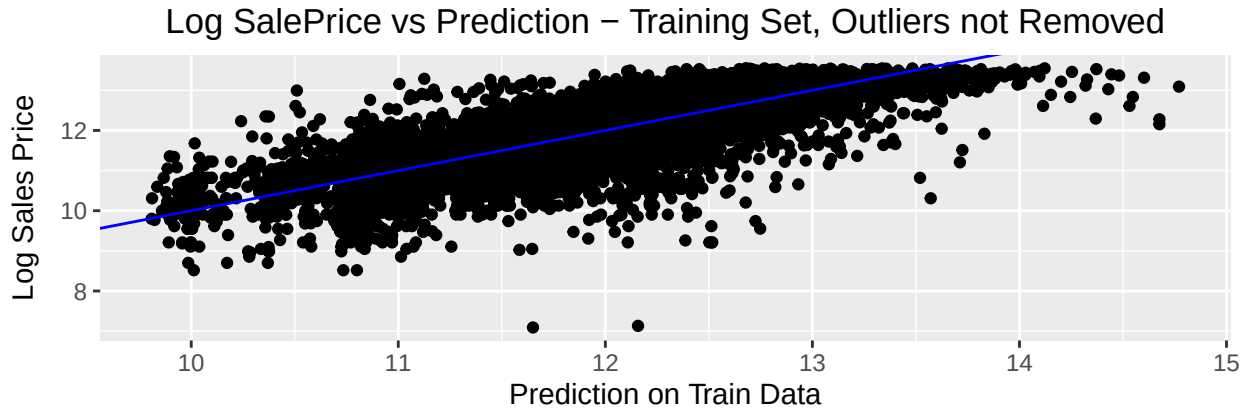
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for inverse.gaussian family taken to be 0.0001033406)
##
##     Null deviance: 6.4210  on 23358  degrees of freedom
## Residual deviance: 2.5226  on 23350  degrees of freedom
## AIC: 27701
##
## Number of Fisher Scoring iterations: 4
```

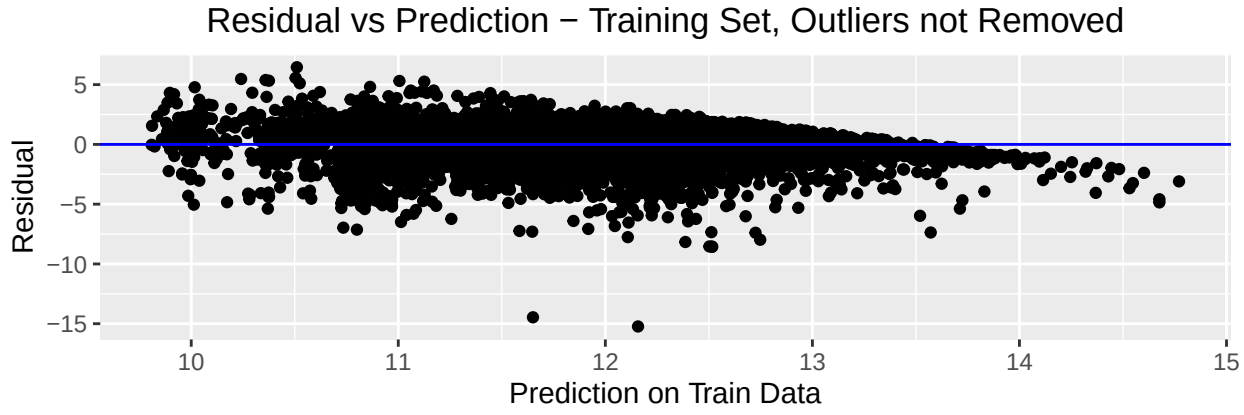
Model Evaluation 1 - AIC & Residual Analysis

```
mod.inv.gsn$aic
```

```
## [1] 27701.45
```

```
bin.inv.gsn = eval.train_init(mod.inv.gsn, train, outliers = TRUE)
```






```
train.inv.gsn = eval.train_update(mod.inv.gsn, train, bin.inv.gsn)
```

In the model's summary, the model received an AIC value of 27701.45, which will be used to compare with the next model with removed outliers in the train set.

From the graph above, we can see that from the first graph, the predictions made by the model are spread around the y=x line, which shows that the predictions are close to the actual values. However, we can see that there are a couple of wrong predictions. The second graph highlights these wrong outcomes as we can see that there are some residuals that are far from the predictions. Residuals were processed and removed the same way as before in the previous model.

Remodelling With New Train Set

```
mod.inv.gsn.final <- glm(f, data = train.inv.gsn, family = Gamma(link = "identity"))
summary(mod.inv.gsn.final)
```

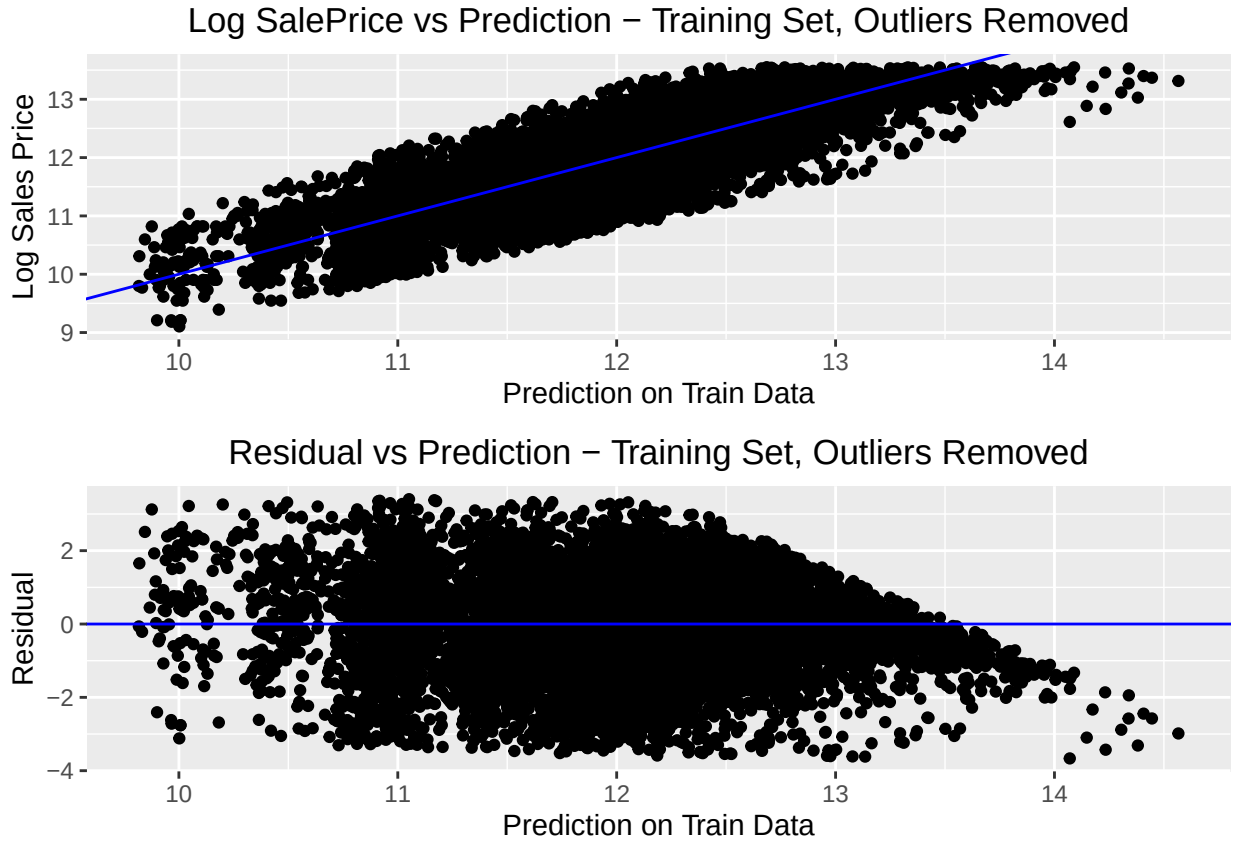
```
##
## Call:
## glm(formula = f, family = Gamma(link = "identity"), data = train.inv.gsn)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.107497  -0.016528   0.002191   0.017642   0.101047
##
## Coefficients:
##                       Estimate        Std. Error  t value             Pr(>|t|)
## (Intercept) 11.71202120256290  0.06647351420353  176.191 < 0.0000000000000002
## SQFT        -0.00000000004317  0.00000000012290   -0.351               0.7254
## FLR1AREA     0.00004843486514  0.00000692061323    6.999  0.00000000000265579
## SFLA         0.00041172723205  0.00000493244862   83.473 < 0.0000000000000002
## EFF_AGE     -0.00250237405594  0.00002350102656 -106.479 < 0.0000000000000002
## GRADEB       0.11937284111668  0.06540892435902    1.825               0.0680
## GRADEC      -0.15189550526945  0.06520175242573   -2.330               0.0198
## GRADED      -0.51943366459415  0.06585630943438   -7.887  0.00000000000000322
## GRADEE      -0.88464307698428  0.06704242998252  -13.195 < 0.0000000000000002
##
## (Intercept) ***
## SQFT
## FLR1AREA    ***
## SFLA        ***
## EFF_AGE     ***
## GRADEB      .
## GRADEC      *
## GRADED      ***
## GRADEE      ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.0008818494)
##
##     Null deviance: 63.866  on 22953  degrees of freedom
## Residual deviance: 20.372  on 22945  degrees of freedom
## AIC: 18450
##
## Number of Fisher Scoring iterations: 4
```

Model Evaluation 2 - New AIC, Residual Analysis, and Error Analysis

```
data.frame("Outliers Not Removed" = mod.inv.gsn$aic,
           "Outliers Removed" = mod.inv.gsn.final$aic)
```

```
##   Outliers.Not.Removed Outliers.Removed
## 1             27701.45         18449.73
```

```
eval.train_init(mod.inv.gsn.final, train.inv.gsn, outliers = FALSE)
```



Just as we have also previously concluded in the 2 previous models, we can conclude that there is a significant decrease in the AIC of the model after removing the outliers in the training set. This means that the model is now performing better with the outliers removed.

From the graphs, we can now see that the predictions are now closer the actual values and the residuals are also closer to 0 as well. It is also worth noting that there are no significant patterns present in the residuals vs prediction plot; the residuals are randomly distributed.

Now, we are going to analyze the error in the predictions of the train set.

```
merge(stack(comp(mod.inv.gsn$fitted.values, mod.inv.gsn$y)),
      stack(comp(mod.inv.gsn.final$fitted.values, mod.inv.gsn.final$y)),
      by = "ind", sort = FALSE)
```

```
##     ind        values.x        values.y
```

```
## 1     n 23359.000000000 22954.000000000
## 2    R2     0.621365073     0.683384883
## 3   MSE     0.172681746     0.126041077
## 4 SEMSE     0.003227415     0.001371682
## 5  RMSE     0.415549932     0.355022643
## 6    SE     0.002718976     0.002343345
## 7   MAE     0.295421490     0.270387611
## 8  MAPE     2.497945671     2.258122196
```

From the results, we can conclude that out of the 22954 data in the train set with outliers removed, the model's predictions receive an RMSE of 0.3550 and an average absolute error of 2.258%. These values show an increase in performance from the previous model with outliers intact. This can be concluded from the lower RMSE and MAPE values.

Multicollinearity

```
vif(mod.inv.gsn.final)
```

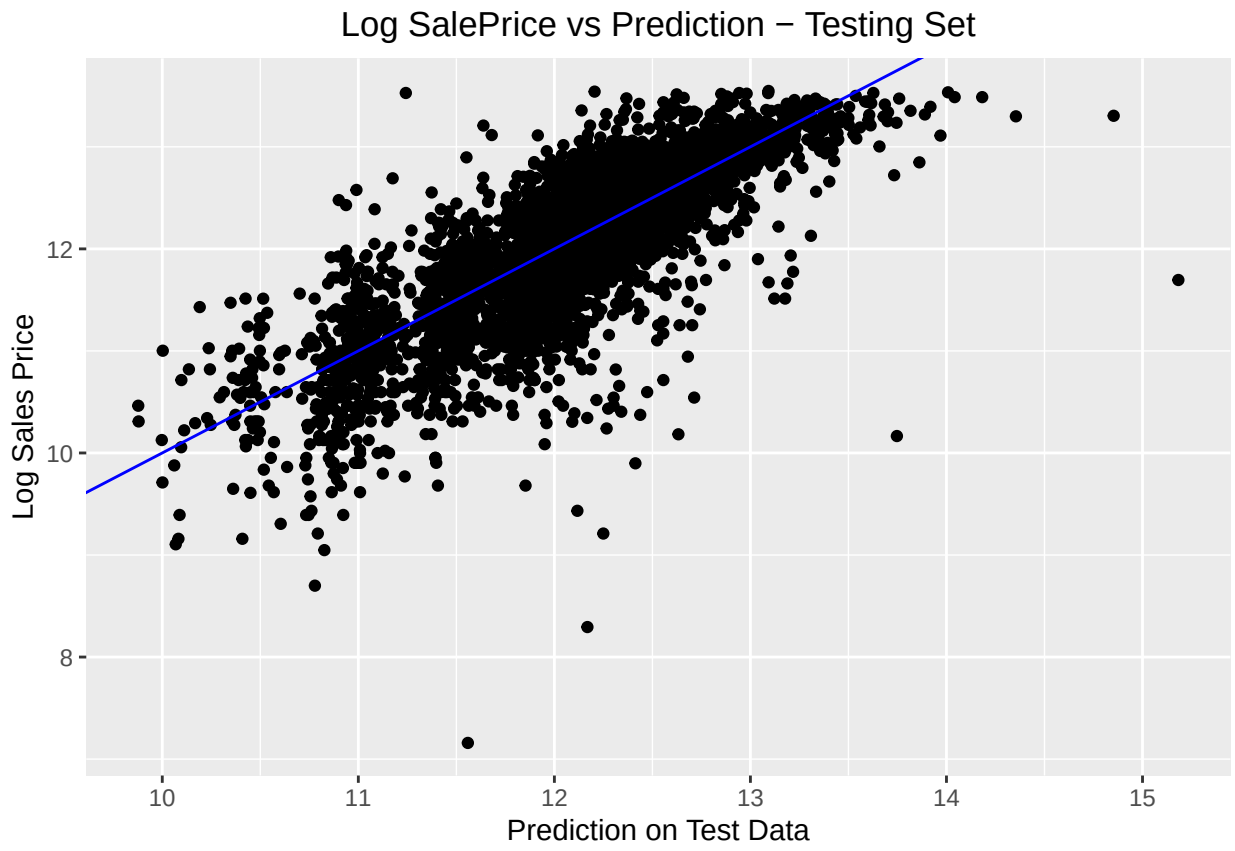
```
##              GVIF Df GVIF^(1/(2*Df))
## SQFT     1.000953  1        1.000476
## FLR1AREA 1.533930  1        1.238519
## SFLA     1.953078  1        1.397526
## EFF_AGE  1.064323  1        1.031660
## GRADE    1.440720  4        1.046701
```

From the VIF scores, we can see that all of them have a relatively small VIF with values below 4. Therefore, we can conclude that each independent variable is not a linear combination of other independent variables. In other words, it is unlikely that there exists a relationship between the independent variables.

Prediction and Error Analysis

```
test.inv.gsn = test
test.inv.gsn$prediction = predict(mod.inv.gsn.final, newdata = test.inv.gsn, type = "response")
```

```
error.inv.gsn = eval.test(test.inv.gsn$prediction, test.inv.gsn$Price)
```




```
## $n
## [1] 5837
##
## $R2
## [1] 0.6340485
##
## $MSE
## [1] 0.1722045
##
## $SEMSE
## [1] 0.007289626
##
## $RMSE
## [1] 0.4149753
##
## $SE
## [1] 0.005420554
##
## $MAE
## [1] 0.2920642
##
## $MAPE
## [1] 2.483237
```

From the results, we can conclude that out of the 5837 data in the test set, the model has predicted with RMSE of 0.4150 and average absolute error of 2.483%. These metrics will be kept for further comparison with the results from other models.

Comparison and Conclusion

After creating 3 models with the gaussian, gamma, and inverse gaussian distribution, we are now going to compare the performance of the models using the computed metrics.

```
comp.aic <- data.frame("Metric" = "AIC",
                       "Gaussian" = mod.gaussian.final$aic,
                       "Gamma" = mod.gamma.final$aic,
                       "Inverse Gaussian" = mod.inv.gsn.final$aic)
comp.error <- merge(stack(error.gaussian), stack(error.gamma), by="ind", sort = FALSE)
comp.error <- merge(comp.error, stack(error.inv.gsn), by="ind", sort = FALSE)
names(comp.error) <- names(comp.aic)
comp.df <- rbind(comp.aic, comp.error)

comp.df
```

```
##   Metric        Gaussian           Gamma Inverse.Gaussian
## 1    AIC 18233.364752997 18568.778379527  18449.734131337
## 2      n  5837.000000000  5837.000000000   5837.000000000
## 3     R2     0.634252090     0.634073257      0.634048482
## 4    MSE     0.171946072     0.172135841      0.172204504
## 5  SEMSE     0.007261306     0.007294248      0.007289626
## 6   RMSE     0.414663805     0.414892566      0.414975305
## 7     SE     0.005418054     0.005419834      0.005420554
## 8    MAE     0.291827727     0.291961823      0.292064170
## 9   MAPE     2.480886534     2.482132884      2.483236768
```

From the results above, we can see that the AIC of the gaussian glm is lower than that of gamma glm's and inverse gaussian glm's. This indicates that there are more residues in the gamma and inverse gaussian glm which in turn affects the fitted values which affects the AIC value. Therefore, we can conclude that the gaussian glm is slightly a better model in terms of fitted values and its residues.

Similar conclusions can also be taken by evaluating other metrics, such as RMSE and MAPE. The gaussian glm outperforms the gamma and inverse gaussian glm in both metrics: the RMSE is smaller by 0.000228761 and 0.0003115 consecutively while the MAPE is smaller by 0.00124635% and 0.002350234%. This means that the square root of the average of the squared error and the average absolute percentage of error in the prediction of gaussian glm is slightly less than the values obtained from the gamma and inverse gaussian model. Therefore, the gaussian glm is more accurate than gamma glm and inverse gaussian glm. In addition, the gamma glm also outperforms the inverse gaussian in both of the metrics discussed.

Therefore, it can be concluded that even though the `Prices` data is skewed, we can still fit the data into a gaussian glm or the regular multiple linear model. In this case, the gaussian glm is the best model for fitting the data with the highest accuracy, followed by the gamma and inverse gaussian glm. Although, it is worth noting that the difference between the evaluating metrics are fairly small.