# Homework5.Rmd

2023-10-10

```
#install.packages("car")
library(caret)
```

```
library(base)
library(car)
```

```
library(ModelMetrics)
```

```
library(pls)
```

```
library(glmnet)
```

```
library(MASS)
library(earth)
```

```
library(ggplot2)
```

```
#Reading the data
housingData <- read.csv("housingData.csv")
#Converting data to string
#str(housingData)
#Printing the first five rows
#head(housingData)
#Printing the summary of data
#summary(housingData)
```

## Question 1(a)-i

### Creating a hold-out validationSet using the first 100 observations in the data and the trainingSet using the remaining 900 observations form the data set “housingData”

```
validationSet<- housingData[1:100,]
trainingSet<- housingData[101:1000,]
```

### Creating the ols model using lm for remaining 100 observations

```
ols_model5<-lm(log(SalePrice)~Foundation+ CentralAir +PavedDrive +BsmtQual
+ExterQual +KitchenQual +TotRmsAbvGrd +GarageArea +Neighborhood +YearBuilt
+OverallQual, data=trainingSet)
```

### Printing the summary of ols model

```
summary(ols_model5)
```

```
##
## Call:
## lm(formula = log(SalePrice) ~ Foundation + CentralAir + PavedDrive +
##     BsmtQual + ExterQual + KitchenQual + TotRmsAbvGrd + GarageArea +
##     Neighborhood + YearBuilt + OverallQual, data = trainingSet)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.69777 -0.08719 -0.00670  0.08497  0.52607
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          1.065e+01  8.487e-01  12.545  < 2e-16 ***
## FoundationCBlock     9.037e-02  2.268e-02   3.985 7.33e-05 ***
## Foundationother     -3.387e-02  6.503e-02  -0.521 0.602615
## FoundationPConc      7.517e-02  2.601e-02   2.890 0.003951 **
## CentralAirY          1.498e-01  2.645e-02   5.664 2.03e-08 ***
## PavedDriveP         -6.863e-03  3.914e-02  -0.175 0.860853
## PavedDriveY          2.369e-02  2.708e-02   0.875 0.382092
## BsmtQualAvg         -6.446e-02  1.711e-02  -3.768 0.000176 ***
## BsmtQualBelowAvg    -1.213e-01  3.910e-02  -3.101 0.001993 **
## ExterQualAvg        -3.584e-02  1.776e-02  -2.018 0.043956 *
## ExterQualBelowAvg   -1.017e-01  7.221e-02  -1.409 0.159269
## KitchenQualAvg      -8.390e-02  1.501e-02  -5.588 3.11e-08 ***
## KitchenQualBelowAvg -9.225e-02  3.841e-02  -2.402 0.016544 *
## TotRmsAbvGrd         5.451e-02  3.803e-03  14.334  < 2e-16 ***
## GarageArea           3.651e-04  3.369e-05  10.838  < 2e-16 ***
## NeighborhoodClearCr  1.911e-01  4.515e-02   4.233 2.56e-05 ***
## NeighborhoodCollgCr -6.342e-02  3.677e-02  -1.725 0.084949 .
## NeighborhoodCrawfor  1.371e-01  3.908e-02   3.508 0.000475 ***
## NeighborhoodEdwards -7.294e-02  3.368e-02  -2.165 0.030635 *
## NeighborhoodGilbert -2.754e-02  4.055e-02  -0.679 0.497186
## NeighborhoodIDOTRR  -8.027e-02  4.666e-02  -1.720 0.085772 .
## NeighborhoodMitchel -2.036e-02  4.076e-02  -0.500 0.617533
## NeighborhoodNAmes   -3.338e-02  3.191e-02  -1.046 0.295937
## NeighborhoodNoRidge  1.349e-01  4.609e-02   2.927 0.003518 **
## NeighborhoodNridgHt  1.313e-02  4.418e-02   0.297 0.766406
## NeighborhoodNWAmes  -1.051e-02  3.811e-02  -0.276 0.782724
## NeighborhoodOldTown -1.231e-01  3.248e-02  -3.789 0.000162 ***
## Neighborhoodother   -9.099e-02  3.392e-02  -2.683 0.007446 **
## NeighborhoodSawyer  -3.657e-02  3.661e-02  -0.999 0.318054
## NeighborhoodSawyerW -8.177e-02  3.929e-02  -2.081 0.037727 *
## NeighborhoodSomerst -8.957e-02  4.207e-02  -2.129 0.033563 *
## NeighborhoodTimber   4.339e-02  4.850e-02   0.895 0.371270
## YearBuilt            6.562e-05  4.400e-04   0.149 0.881481
## OverallQual          1.030e-01  6.587e-03  15.635  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1481 on 836 degrees of freedom
```

```
##   (30 observations deleted due to missingness)
## Multiple R-squared:  0.8321, Adjusted R-squared:  0.8255
## F-statistic: 125.6 on 33 and 836 DF,  p-value: < 2.2e-16
```

### Printing the residuals of ols model

```
residuals5= residuals(ols_model5)

anova(ols_model5)

## Analysis of Variance Table
##
## Response: log(SalePrice)
##              Df  Sum Sq Mean Sq  F value  Pr(>F)
## Foundation    3 29.1679  9.7226 443.0732 < 2e-16 ***
## CentralAir    1  4.9040  4.9040 223.4807 < 2e-16 ***
## PavedDrive    2  1.9450  0.9725  44.3188 < 2e-16 ***
## BsmtQual      2  8.0926  4.0463 184.3956 < 2e-16 ***
## ExterQual     2  9.6435  4.8217 219.7327 < 2e-16 ***
## KitchenQual   2  3.3169  1.6585  75.5777 < 2e-16 ***
## TotRmsAbvGrd  1 17.4002 17.4002 792.9496 < 2e-16 ***
## GarageArea    1  6.0085  6.0085 273.8148 < 2e-16 ***
## Neighborhood 17  4.9680  0.2922  13.3174 < 2e-16 ***
## YearBuilt     1  0.1054  0.1054   4.8051 0.02865 *
## OverallQual   1  5.3639  5.3639 244.4407 < 2e-16 ***
## Residuals   836 18.3449  0.0219
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

residualSE5<-sqrt(anova(ols_model5)[[3]][5])
residualSE5

## [1] 2.195846

#Calculating the RMSE value
RMSE_5<- sqrt(mean(ols_model5$residuals^2))
RMSE_5

## [1] 0.1452105

#str(ols_model5)
summary(ols_model5)

##
## Call:
## lm(formula = log(SalePrice) ~ Foundation + CentralAir + PavedDrive +
##     BsmtQual + ExterQual + KitchenQual + TotRmsAbvGrd + GarageArea +
##     Neighborhood + YearBuilt + OverallQual, data = trainingSet)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.69777 -0.08719 -0.00670  0.08497  0.52607
```

```
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)    
## (Intercept)          1.065e+01  8.487e-01  12.545  < 2e-16 ***
## FoundationCBlock     9.037e-02  2.268e-02   3.985 7.33e-05 ***
## Foundationother     -3.387e-02  6.503e-02  -0.521 0.602615    
## FoundationPConc      7.517e-02  2.601e-02   2.890 0.003951 ** 
## CentralAirY          1.498e-01  2.645e-02   5.664 2.03e-08 ***
## PavedDriveP         -6.863e-03  3.914e-02  -0.175 0.860853    
## PavedDriveY          2.369e-02  2.708e-02   0.875 0.382092    
## BsmtQualAvg         -6.446e-02  1.711e-02  -3.768 0.000176 ***
## BsmtQualBelowAvg    -1.213e-01  3.910e-02  -3.101 0.001993 ** 
## ExterQualAvg        -3.584e-02  1.776e-02  -2.018 0.043956 *  
## ExterQualBelowAvg   -1.017e-01  7.221e-02  -1.409 0.159269    
## KitchenQualAvg      -8.390e-02  1.501e-02  -5.588 3.11e-08 ***
## KitchenQualBelowAvg -9.225e-02  3.841e-02  -2.402 0.016544 *  
## TotRmsAbvGrd         5.451e-02  3.803e-03  14.334  < 2e-16 ***
## GarageArea           3.651e-04  3.369e-05  10.838  < 2e-16 ***
## NeighborhoodClearCr  1.911e-01  4.515e-02   4.233 2.56e-05 ***
## NeighborhoodCollgCr -6.342e-02  3.677e-02  -1.725 0.084949 .  
## NeighborhoodCrawfor  1.371e-01  3.908e-02   3.508 0.000475 ***
## NeighborhoodEdwards -7.294e-02  3.368e-02  -2.165 0.030635 *  
## NeighborhoodGilbert -2.754e-02  4.055e-02  -0.679 0.497186    
## NeighborhoodIDOTRR  -8.027e-02  4.666e-02  -1.720 0.085772 .  
## NeighborhoodMitchel -2.036e-02  4.076e-02  -0.500 0.617533    
## NeighborhoodNAmes   -3.338e-02  3.191e-02  -1.046 0.295937    
## NeighborhoodNoRidge  1.349e-01  4.609e-02   2.927 0.003518 ** 
## NeighborhoodNridgHt  1.313e-02  4.418e-02   0.297 0.766406    
## NeighborhoodNWAmes  -1.051e-02  3.811e-02  -0.276 0.782724    
## NeighborhoodOldTown -1.231e-01  3.248e-02  -3.789 0.000162 ***
## Neighborhoodother   -9.099e-02  3.392e-02  -2.683 0.007446 ** 
## NeighborhoodSawyer  -3.657e-02  3.661e-02  -0.999 0.318054    
## NeighborhoodSawyerW -8.177e-02  3.929e-02  -2.081 0.037727 *  
## NeighborhoodSomerst -8.957e-02  4.207e-02  -2.129 0.033563 *  
## NeighborhoodTimber   4.339e-02  4.850e-02   0.895 0.371270    
## YearBuilt            6.562e-05  4.400e-04   0.149 0.881481    
## OverallQual          1.030e-01  6.587e-03  15.635  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1481 on 836 degrees of freedom
##   (30 observations deleted due to missingness)
## Multiple R-squared:  0.8321, Adjusted R-squared:  0.8255 
## F-statistic: 125.6 on 33 and 836 DF,  p-value: < 2.2e-16

#The AIC value for ols model
AIC(ols_model5)

## [1] -818.5005
```

```r
#The BIC value for ols model
BIC(ols_model5)

## [1] -651.6032

#The VIF value for ols model
vif_values5<-vif(ols_model5)
```

**ols model [5] is taken as the best model among the five models as it has the best rmse value among all other five models.**

**The RMSE value(residualSE5) for ols model- [5], is 0.1452105 is and the p- value and adjusted R^2 values are 2.2e^-16 and 0.8255. We take these values as the final values as they are much better.**

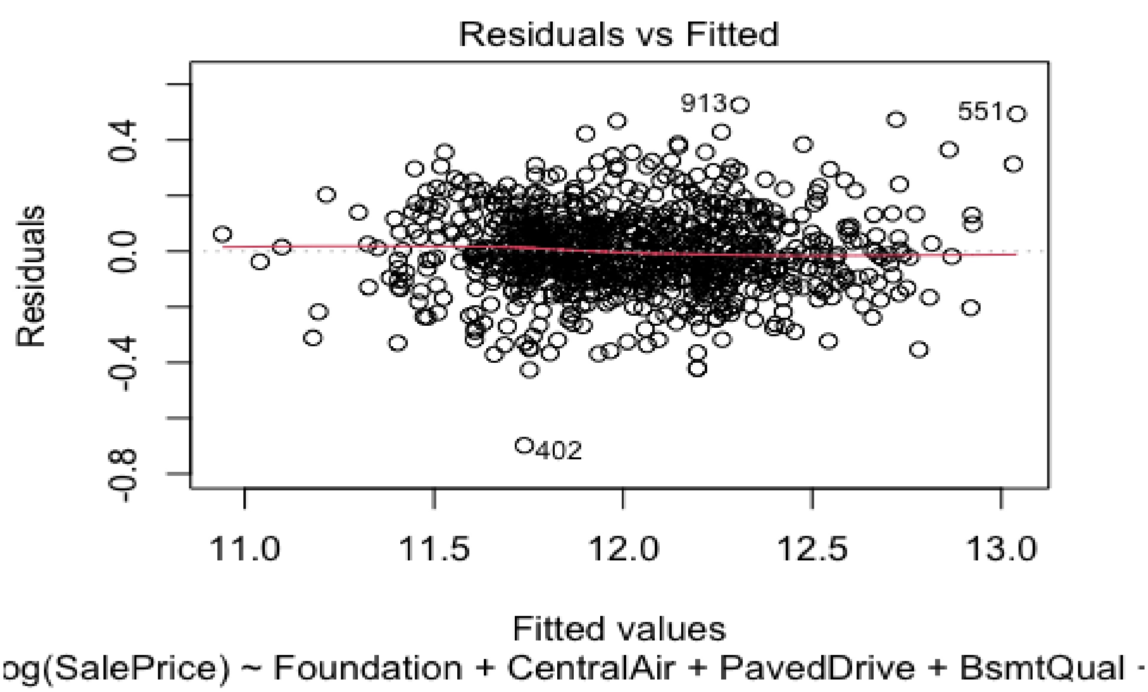
## Question 1(a)-ii

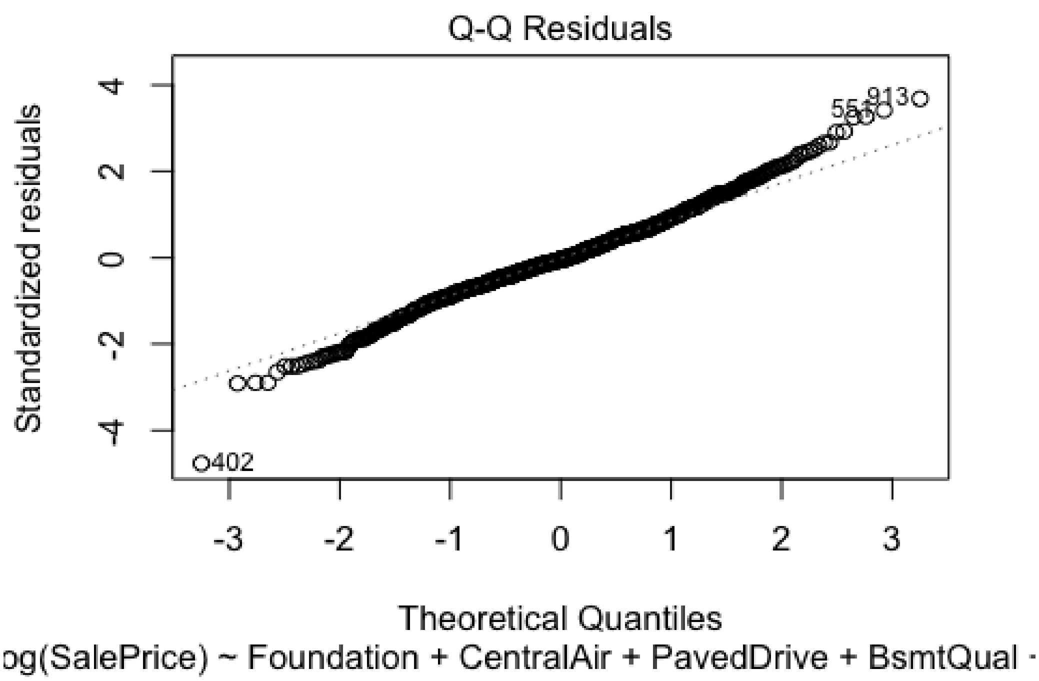
### Residuals for ols model 5

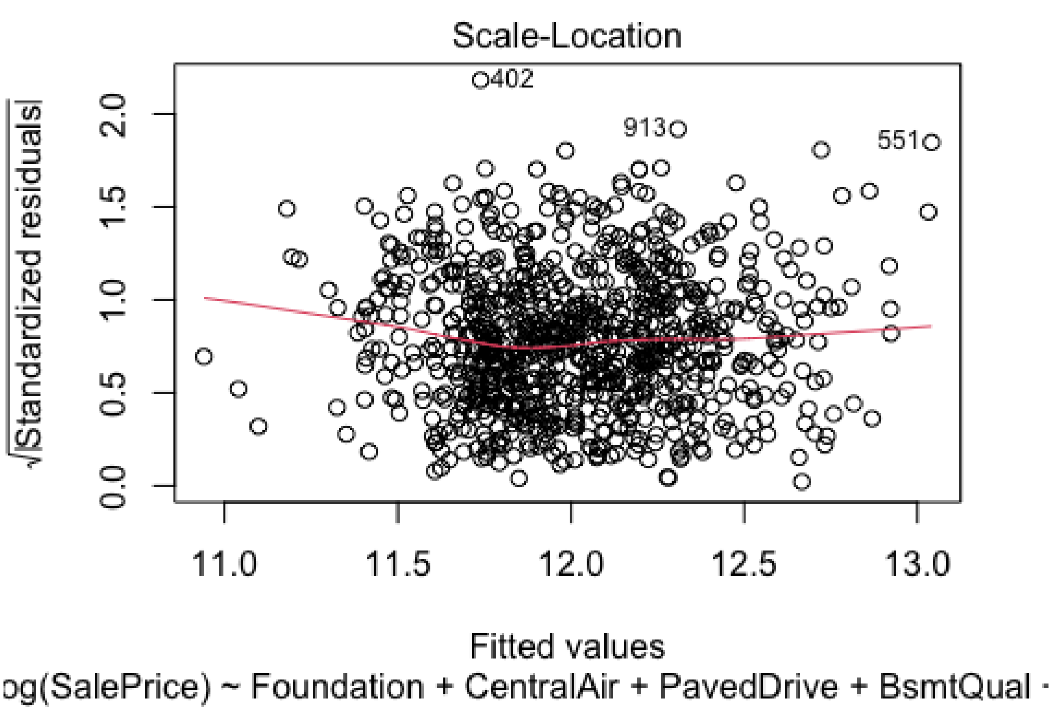
```r
residuals_ols5<-residuals(ols_model5)
```

### Plotting the ols models

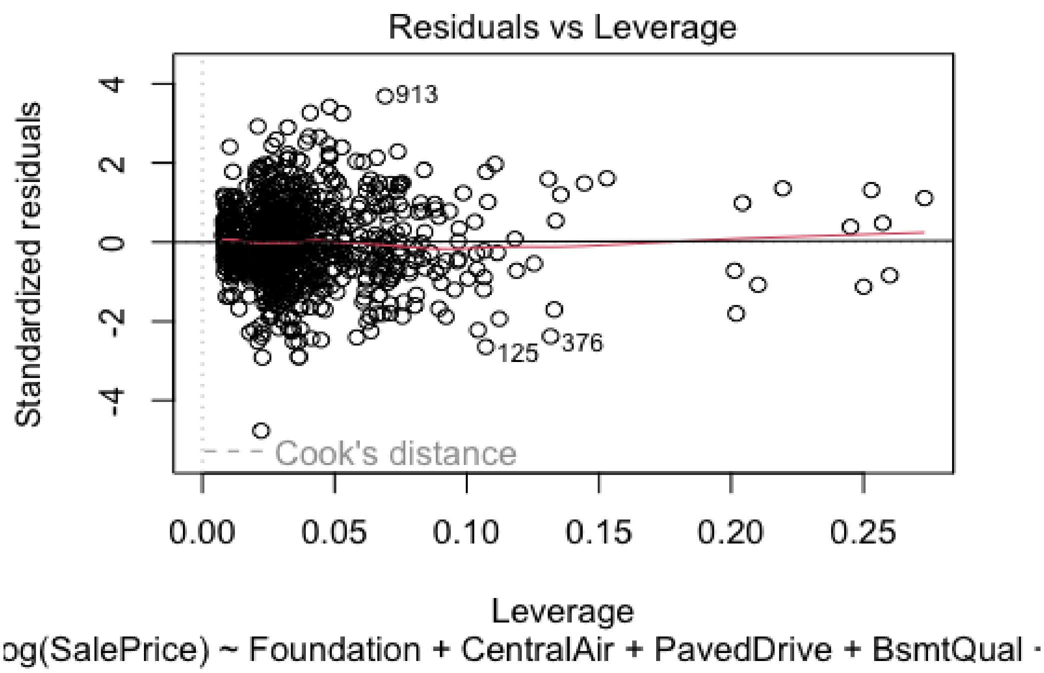
```r
# We take OLS model [5] as the best model
plot(ols_model5)
```



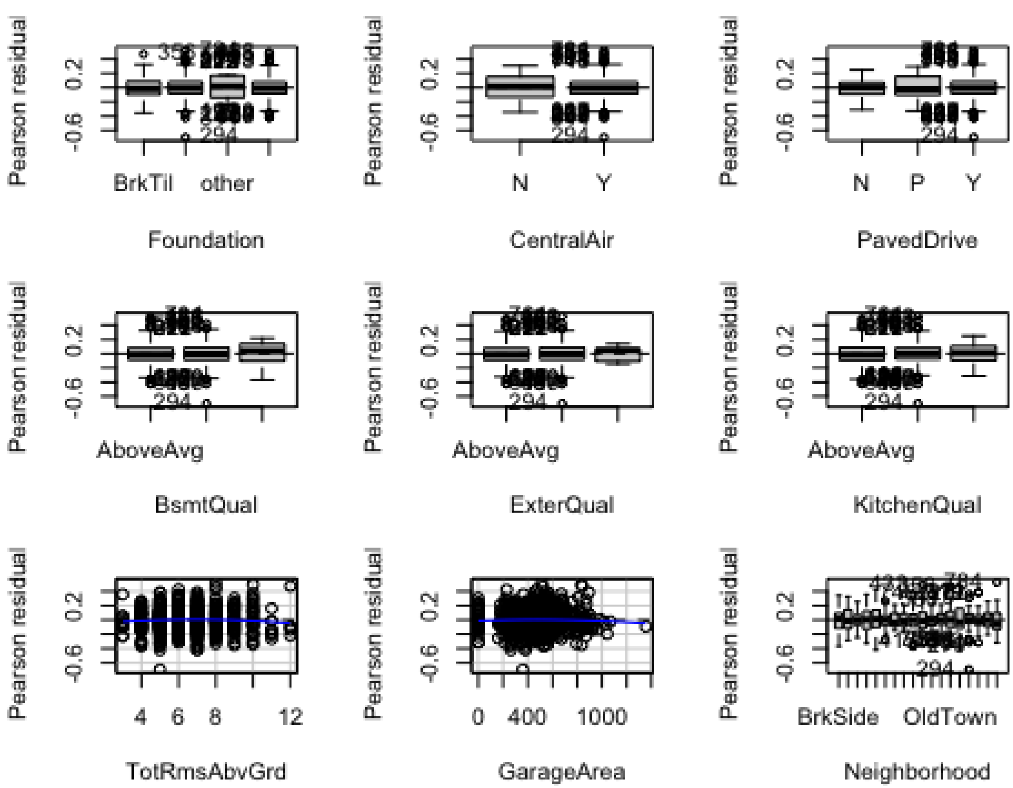


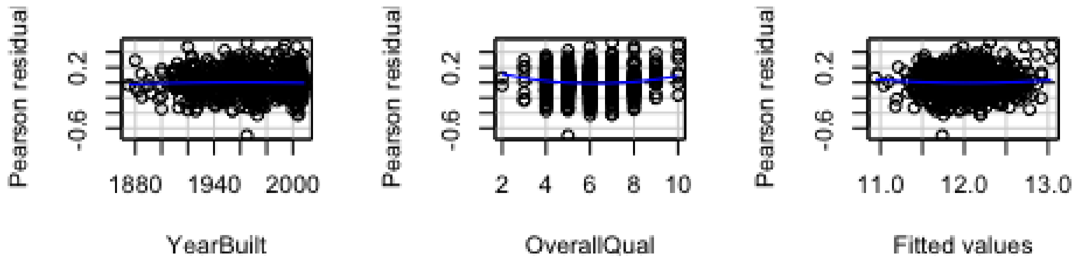




```
qqline(ols_model5$residuals)
```




```
residualPlots(ols_model5)
```






```
##                Test stat Pr(>|Test stat|)
## Foundation
## CentralAir
## PavedDrive
## BsmtQual
## ExterQual
## KitchenQual
## TotRmsAbvGrd   -1.6807        0.0931882 .
## GarageArea     -0.9460        0.3444192
## Neighborhood
## YearBuilt      -1.1870        0.2355569
## OverallQual     3.7616        0.0001806 ***
## Tukey test      1.5499        0.1211669
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

**OlS -[5] is a strong model by depicting the residuals and when we break down the residuals by factor , most of them are evenly distributed above and below the horizontal axis with little bit non linear relation ship of 'OverallQual'.**

**If we transform the OverallQual variable , we may get improved model performance , and also to make the model more efficient , outlier handling can be considered.**

## Question 1 (b)

### Finding the count of missing values

```
missing_count <- sapply(housingData, function(x)
sum(length(which(is.na(x)))))
max_missing <- missing_count[missing_count>200]
#Removing the missing values from data
housingData1 <- housingData[,!names(housingData) %in% names(max_missing)]
act_housingData <- na.omit(housingData1)
```

### Choosing number of components as 8 as best by analysing pls_model1

```
#Creating pls model for the best components with crossvalidation and method
as oscorespls
pls_model2 <- plsr(log(SalePrice)~., 8, data = act_housingData, method =
"oscorespls", validation = "CV")
#Creating the summary of PLS model 2
summary(pls_model2)

## Data:    X dimension: 915 153
##  Y dimension: 915 1
## Fit method: oscorespls
## Number of components considered: 8
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.3403   0.3307   0.1882   0.1835   0.1755   0.1717   0.1690
## adjCV       0.3403   0.3303   0.1882   0.1834   0.1754   0.1716   0.1689
##        7 comps  8 comps
## CV      0.1647   0.1523
## adjCV   0.1656   0.1516
##
## TRAINING: % variance explained
##                   1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7
comps
## X                  98.877    99.24    99.50    99.61    99.80    99.84
99.9
## log(SalePrice)      8.743    69.60    71.32    74.18    75.28    76.10
77.2
```

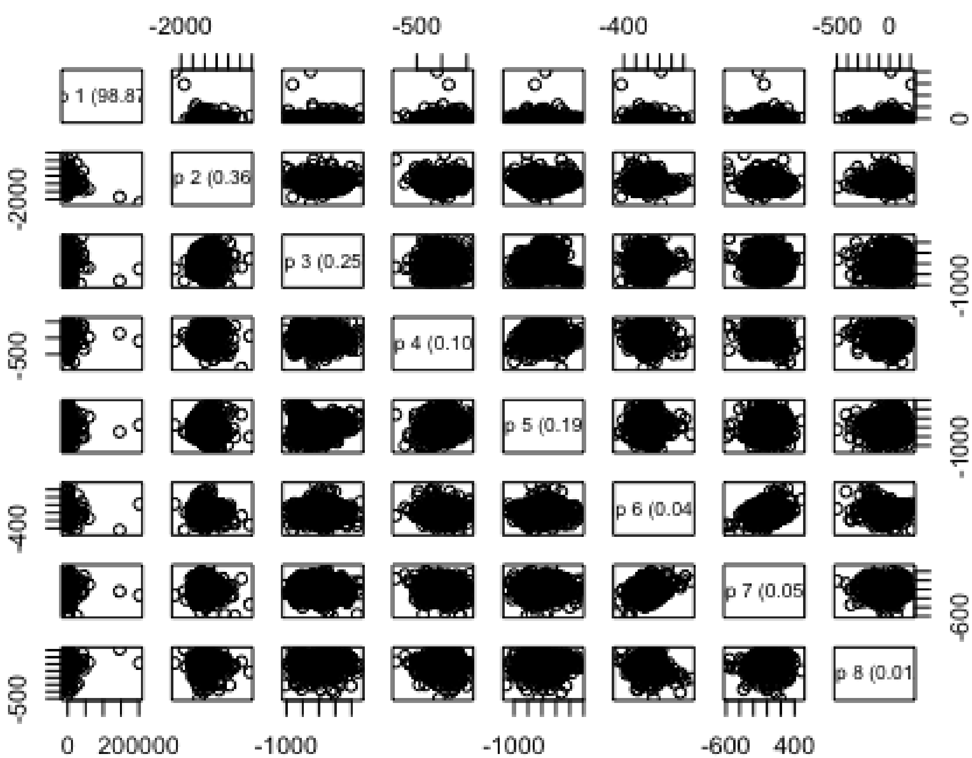
```
##                    8 comps
## X                    99.91
## log(SalePrice)       81.56
```

```
R2(pls_model2)
```

```
## (Intercept)      1 comps      2 comps      3 comps      4 comps      5 
comps
##   -0.002189     0.053420     0.693357     0.708593     0.733269     
0.744705
##     6 comps      7 comps      8 comps
##    0.752731     0.765146     0.799154
```

## Plotting the pls model 2

```
#Plotting the pls model 2
plot(pls_model2, plottype = "scores", comps = 1:8)
```




##

Finding the beta values

```
beta_pls <- drop(coef(pls_model2))
#beta_pls
```

### Finding the residuals

```
residuals_pls <- drop(pls_model2$resid)[,8]
#residuals_pls
```

### The RMSE value with 8 components is 0.1512

### The number of components are 8

## Question - 1(c)

### performing a Lasso regression using the "lars" package in R

```
#install.packages("lars")
library(lars)

## Loaded lars 1.3

#Here removing specified columns From HousingData dataset which are not
useful and adding saleprice column to the dataset.
reducedHD = na.omit(housingData[,c(5,23,35,43, 74)])
reducedHD$SalePrice = log(reducedHD$SalePrice)

#Here X contains predictor variables of the columns 1 to 4 and Y contains 5th
column of reducedHD
y <- as.numeric(reducedHD[,5])      #target variable
x <- as.matrix(reducedHD[,1:4])     #predictors
```

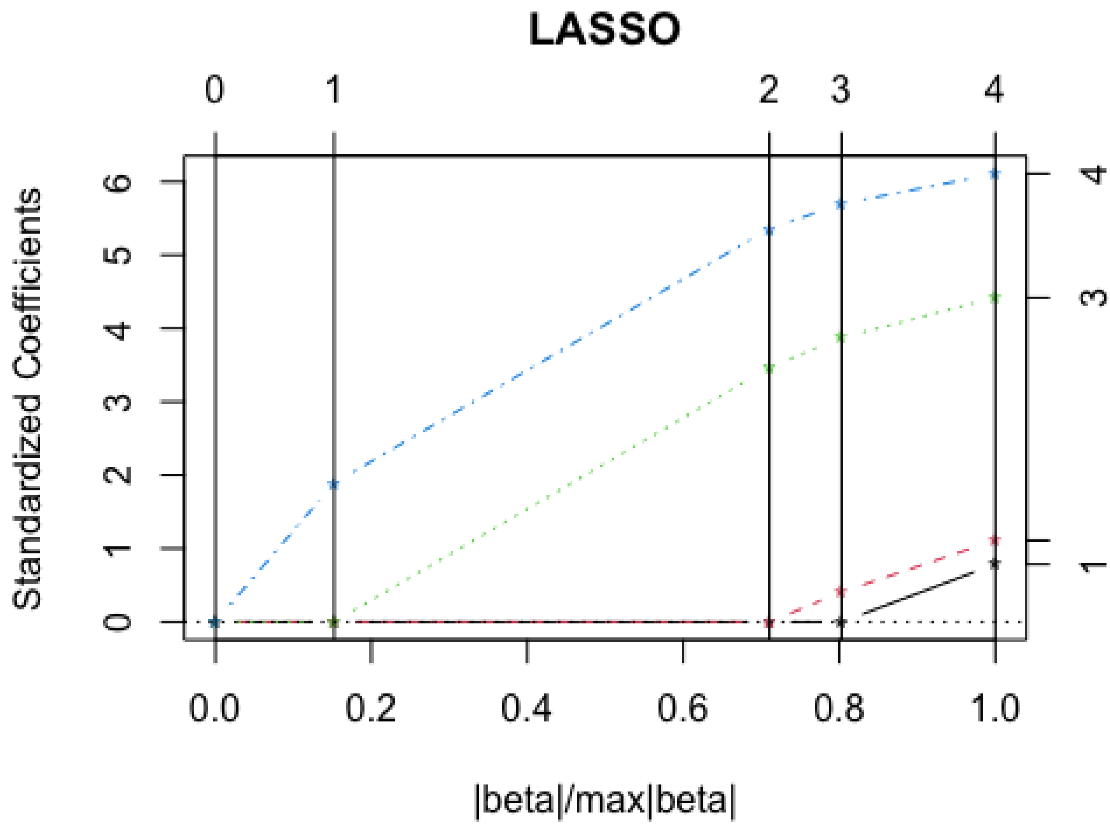
### Lasso model is applied using lars function which best fits a lasso regression model.

```
lasso <- lars(x, y, type="lasso")
```

### Reduce the plot margins and set the size

```
par(mar = c(3, 3, 2, 2))  # Set smaller margins (bottom, left, top, right)
options(repr.plot.width = 6, repr.plot.height = 4)  # Set the plot size
(adjust as needed)

#Plotting of coefficients of predictor variables as the function parameter
lambda.
plot(lasso)
```




```
#It shows the coefficients of the predictor variables change with different
lambda values
lasso$lambda

## [1] 8.374890 6.483601 1.704205 1.065226

summary(lasso)

## LARS/LASSO
## Call: lars(x = x, y = y, type = "lasso")
##   Df     Rss       Cp
## 0  1 131.457 2202.338
## 1  2 103.356 1521.056
## 2  3  46.545  141.730
## 3  4  43.366   66.432
## 4  5  40.757    5.000

lambda.lasso <- c(lasso$lambda,0)
beta <- coef(lasso)

#str(lasso)

colors <- rainbow(4)
```

**lambda values are very high. need to adjust parameters below to show lamdas values**

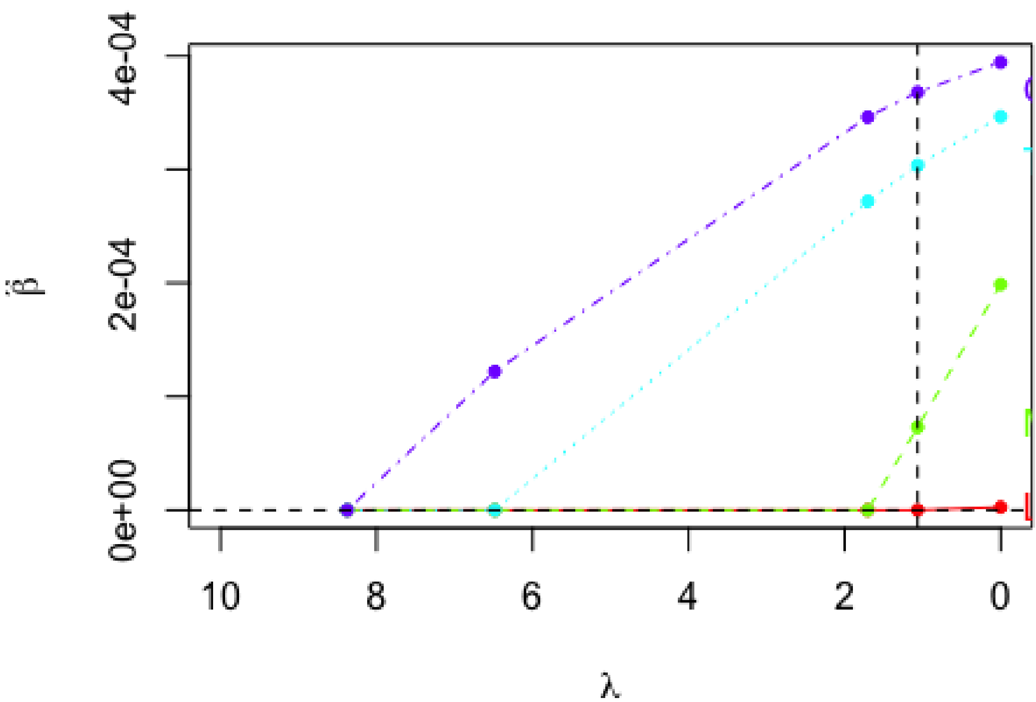
**finish the second part of the problem (do the same for question above)**

**matplot lambda on the x-axis and the corresponding coefficients on the y-axis.**

```
matplot(lambda.lasso, beta, xlim=c(10,0), type="o", pch=20,
xlab=expression(lambda),
        ylab=expression(hat(beta)), col=colors)

#The text function creates lables for the plots
text(rep(-0, 4), beta[4,], colnames(x), pos=4, col=colors)

#abline is used to add vertical and horizontal lines to highlight specific
lambda values and coefficients.
abline(v=lambda.lasso[4], lty=2)
abline(h=0, lty=2)
```




\## Calculating residualsum of the squares values for the lasso model

### It contains the coefficients at a specific lambda value (e.g., lambda = 4).

```
beta.lasso <- beta[4,]

#calculates the residuals by subtracting the predicted values (at lambda = 4)
from the actual target variable 'y.'
resid.lasso <- y - predict(lasso, as.matrix(reducedHD[, 1:4]), s = 4, type =
"fit")$fit
```

### calculates the residual sum of squares.

```
rss.lasso <- sum(resid.lasso^2)/(67-4)
```

### Calculate RMSE manually

```
rmse_lasso <- sqrt(mean(resid.lasso^2))
cat("Lasso Regression RMSE:", rmse_lasso, "\n")

## Lasso Regression RMSE: 0.2086632
```

### Calculate the total sum of squares (TSS)

```
tss <- sum((y - mean(y))^2)
```

### Calculate the residual sum of squares (RSS)

```
rss <- sum(resid.lasso^2)
```

### Calculate R-squared (R2)

```
rsquared_lasso <- 1 - (rss / tss)
cat("Lasso Regression R-squared (R2):", rsquared_lasso, "\n")

## Lasso Regression R-squared (R2): 0.6701125
```

## Question -1(D)

```
library(Metrics)  # Load the Metrics package for RMSE calculation

##
## Attaching package: 'Metrics'

## The following objects are masked from 'package:ModelMetrics':
##
##     auc, ce, logLoss, mae, mse, msle, precision, recall, rmse, rmsle

## The following objects are masked from 'package:caret':
##
##     precision, recall

#Here removing specified columns From HousingData dataset which are not
useful and adding saleprice column to the dataset.
reducedHD = na.omit(housingData[,c(5,23,35,43, 74)])
reducedHD$SalePrice = log(reducedHD$SalePrice)
```

## Ridge regression Model

```
library(Metrics)  # Load the Metrics package for RMSE calculation

# Split your data into training and validation sets (e.g., 70% training, 30% validation)
set.seed(123)  # For reproducibility
sample_indices <- sample(nrow(reducedHD), 0.7 * nrow(reducedHD))
train_data <- reducedHD[sample_indices, ]
validation_data <- reducedHD[-sample_indices, ]
```

## Fit the ridge regression model on the training data with the optimal lambda

```
optimal_lambda <- 4.4  # Replace with your optimal lambda
ridge_model <- lm.ridge(SalePrice ~ ., data = train_data, lambda = optimal_lambda)
```

### Extract coefficients from the ridge model

```
ridge_coefs <- coef(ridge_model)
```

### Create a matrix of predictors for the validation data

```
x_validation <- model.matrix(SalePrice ~ ., data = validation_data)

# Calculate predictions manually
predictions_ridge <- x_validation %*% ridge_coefs

# Extract the actual sale prices from the validation data
actual_sale_prices <- validation_data$SalePrice
```

## Calculate RMSE manually

```
rmse_ridge <- sqrt(mean((predictions_ridge - actual_sale_prices)^2))
cat("Ridge Regression RMSE:", rmse_ridge, "\n")

## Ridge Regression RMSE: 0.2043166
```

## Calculate the square of RMSE

```
rmse_square <- rmse_ridge^2
cat("Square of RMSE:", rmse_square, "\n")

## Square of RMSE: 0.04174528
```

# elasticnet regression

## Load the necessary library

```
library(glmnet)

# Convert the target variable to log scale
reducedHD$SalePrice <- log(reducedHD$SalePrice)
```

## Split your data into training and validation sets (e.g., 70% training, 30% validation)

```
set.seed(123)  # For reproducibility
sample_indices <- sample(nrow(reducedHD), 0.7 * nrow(reducedHD))
train_data <- reducedHD[sample_indices, ]
validation_data <- reducedHD[-sample_indices, ]
```

## Define the predictor variables and response variable

## Define the values of alpha and lambda for Elastic Net

```
# Define the predictor variables and response variable
X <- as.matrix(train_data[, -(5)])  # Predictor variables (excluding the 5th
column, SalePrice)
y <- train_data$SalePrice  # Response variable
elasticNet_grid <- expand.grid(alpha = c(0, 1), lambda = seq(0.001, 1, length
= 100))
## Initialize empty lists to store the models

elasticNet_models <- list()
## Loop through the alpha values and fit Elastic Net models
for (alpha_val in unique(elasticNet_grid$alpha)) {
  # Filter the grid for the current alpha value
  grid_subset <- elasticNet_grid[elasticNet_grid$alpha == alpha_val, ]
### Fit the Elastic Net model using cross-validation
  elasticNet_model <- cv.glmnet(
    x = X,
    y = y,
    alpha = alpha_val,
    lambda = grid_subset$lambda
  )
  # Store the model in the list
  elasticNet_models[[as.character(alpha_val)]] <- elasticNet_model
}

# Access the models using elasticNet_models$`0` and elasticNet_models$`1`
```

### Get the optimal lambda and alpha values

```
optimal_lambda <- elasticNet_model$lambda.min
optimal_alpha <- elasticNet_model$alpha.min
```

### Extract coefficients for the optimal model

```
coef(elasticNet_model, s = "lambda.min")

## 5 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) 2.405823e+00
## LotArea     1.143643e-07
## MasVnrArea  1.019667e-05
```

```
## TotalBsmtSF 2.794386e-05
## GrLivArea   3.240312e-05
```

## Create a matrix of predictors for the validation data

```
X_validation <- as.matrix(validation_data[, !(colnames(validation_data) ==
"SalePrice")])
```

### Make predictions on the validation data

```
predictions_elasticNet <- predict(elasticNet_model, s = optimal_lambda, newx
= X_validation)
```

### Extract the actual sale prices from the validation data

```
actual_sale_prices <- validation_data$SalePrice
```

### Calculate RMSE

```
rmse_elasticNet <- sqrt(mean((predictions_elasticNet -
actual_sale_prices)^2))
cat("ElasticNet Regression RMSE:", rmse_elasticNet, "\n")

## ElasticNet Regression RMSE: 0.0173047
```

### Calculate the square of RMSE

```
rmse_square_elasticNet <- rmse_elasticNet^2
cat("Square of RMSE:", rmse_square_elasticNet, "\n")

## Square of RMSE: 0.0002994527
```

## SVR Model

### Load the required libraries (if not already loaded)

```
library(e1071)  # For SVR
library(Metrics)  # For RMSE calculation
```

### Convert the target variable to log scale

```
# Convert the target variable to log scale
reducedHD$SalePrice <- log(reducedHD$SalePrice)
```

### Split your data into training and validation sets (e.g., 70% training, 30% validation)

```
set.seed(123)  # For reproducibility
sample_indices <- sample(nrow(reducedHD), 0.7 * nrow(reducedHD))
train_data <- reducedHD[sample_indices, ]
validation_data <- reducedHD[-sample_indices, ]
```

## Define the predictor variables and response variable

```
X_train <- train_data[, !(names(train_data) %in% "SalePrice")]  # Exclude the
SalePrice column
y_train <- train_data$SalePrice  # Response variable

X_validation <- validation_data[, !(names(validation_data) %in% "SalePrice")]
```

```r
# Exclude the SalePrice column for validation
y_validation <- validation_data$SalePrice  # Actual SalePrice for validation
```

**Add the SalePrice column to the validation data**

```r
# Add the SalePrice column to the validation data
validation_data_with_SalePrice <- cbind(X_validation, SalePrice =
y_validation)
```

**Fit the SVR model on the training data**

```r
svr_model <- svm(y_train ~ ., data = train_data, kernel = "linear", cost =
7.5, epsilon = 0.125)
```

**Make predictions on the validation data**

```r
predictions_svr <- predict(svr_model, newdata =
validation_data_with_SalePrice)
```

**Calculate RMSE for SVR**

```r
rmse_svr <- sqrt(mean((predictions_svr - y_validation)^2))
```

**Calculate R-squared for SVR**

```r
ssr <- sum((predictions_svr - mean(y_validation))^2)
sst <- sum((y_validation - mean(y_validation))^2)
rsquared_svr <- 1 - (ssr / sst)

cat("SVR RMSE:", rmse_svr, "\n")

## SVR RMSE: 0.0007029618

cat("SVR R-squared (R²):", rsquared_svr, "\n")

## SVR R-squared (R²): 0.04587419
```

| MODEL | NOTES | HYPERPARAMETERS | CV RMSE | CV R^2 |
|---|---|---|---|---|
| OLS | lm | N/A | 0.1452105 | 0.8255 |
| PLS | pls | N Components = 8<br>Method = OSCOREPLS | 0.1512 | 0.800179 |
| LASSO | | N/A | 0.20866 | 0.67011 |
| ELASTIC NET | `glmnet` | NA | 0.01730 | 0.000299 |
| SVR | `Metrics+e1071` | epsilon = 0.125 + cost =7.5 | 0.000702 | 0.04587 |
| RIDGE REGRESSION | Metrics | lambda = 4.4 | 0.20431 | 0.041 |