STATISTICS FOR BUSINESS ANALYTICS I - Assignment 2
VRETTEAS STELIOS p2822003

1) Read the "usdata" dataset and use str() to understand its structure

Dataset file **usdata** has no specific format and R reads all variables as integers.
Further data cleaning will be needed in order to define the variables into numeric and factors.
Clearely, the PRICE, SQFT, AGE and FEATS variables are numeric and NE, COR are factors.
Below the Str and summary function outputs. (no missing values appear)

```
> str(data)
'data.frame':   63 obs. of  6 variables:
 $ PRICE: int  2050 2150 2150 1999 1900 1800 1560 1449 1375 1270 ...
 $ SQFT : int  2650 2664 2921 2580 2580 2774 1920 1710 1837 1880 ...
 $ AGE  : int  3 28 17 20 20 10 2 2 20 30 ...
 $ FEATS: int  7 5 6 4 4 4 5 3 5 6 ...
 $ NE   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ COR  : int  0 0 0 0 0 0 0 0 0 0 ...
>
```

```
> summary(data)
     PRICE           SQFT           AGE            FEATS            NE              COR
 Min.   : 580   Min.   : 970   Min.   : 2.00   Min.   :1.000   Min.   :0.000   Min.   :0.0000
 1st Qu.: 910   1st Qu.:1400   1st Qu.: 7.00   1st Qu.:3.000   1st Qu.:0.000   1st Qu.:0.0000
 Median :1049   Median :1680   Median :20.00   Median :4.000   Median :1.000   Median :0.0000
 Mean   :1158   Mean   :1730   Mean   :17.46   Mean   :3.952   Mean   :0.619   Mean   :0.2222
 3rd Qu.:1250   3rd Qu.:1920   3rd Qu.:27.50   3rd Qu.:4.000   3rd Qu.:1.000   3rd Qu.:0.0000
 Max.   :2150   Max.   :2931   Max.   :31.00   Max.   :8.000   Max.   :1.000   Max.   :1.0000
>
```

2) Convert the variables PRICE, SQFT, AGE, FEATS to be numeric variables and NE, COR to be factors.

With the help of ***lapply***, ***as.numeric*** *and* ***as.factor*** functions we transform the first four variables into numeric and the other two as factors in order to examine them separately.

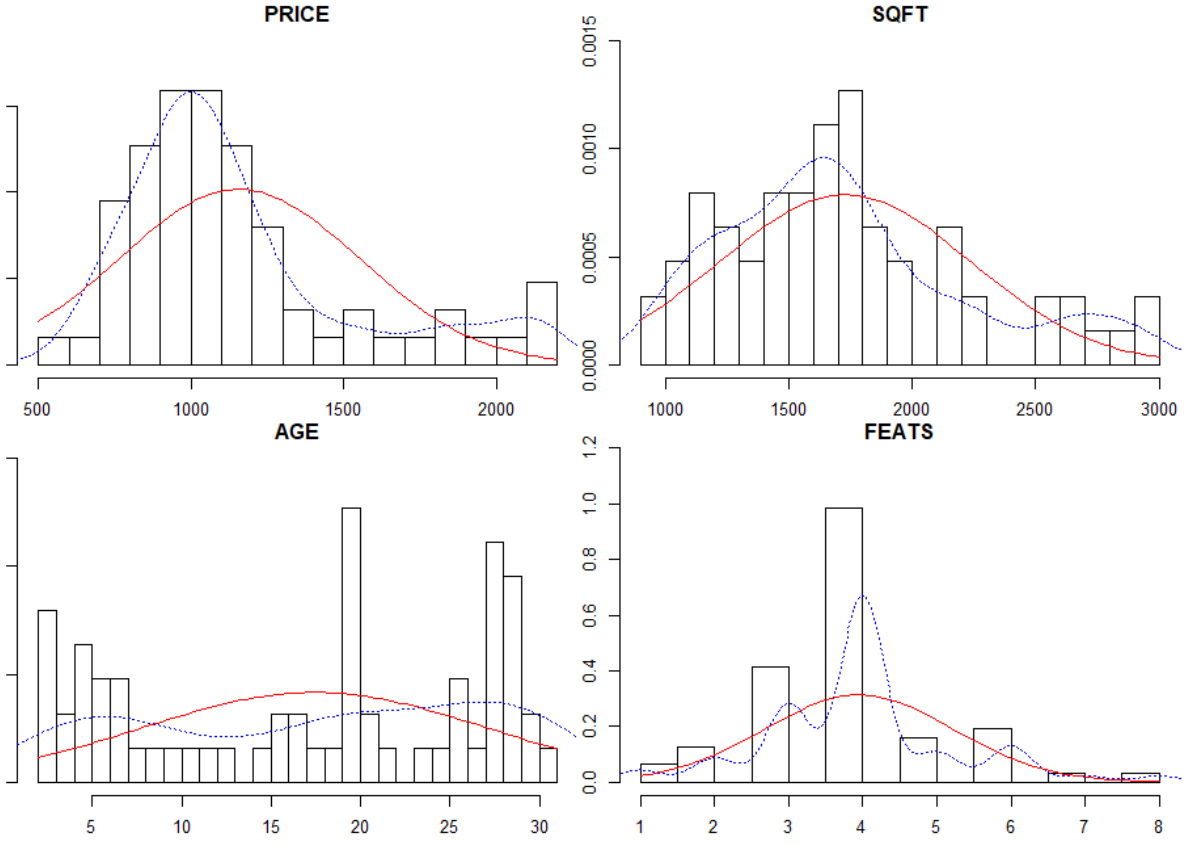
3) Perform descriptive analysis and visualization for each variable to get an initial insight of what the data looks like. Comment on your findings.

First, we have to separate the numeric variables from the factors, so we create a new dataset data_num for the numeric and data_fac for the factors.
With the help of describe and summary we have a first look for the distributions of the data. We retrieve information about the descriptive measures of the variables, (min, max, median , mean skew, kurtosis etc.)

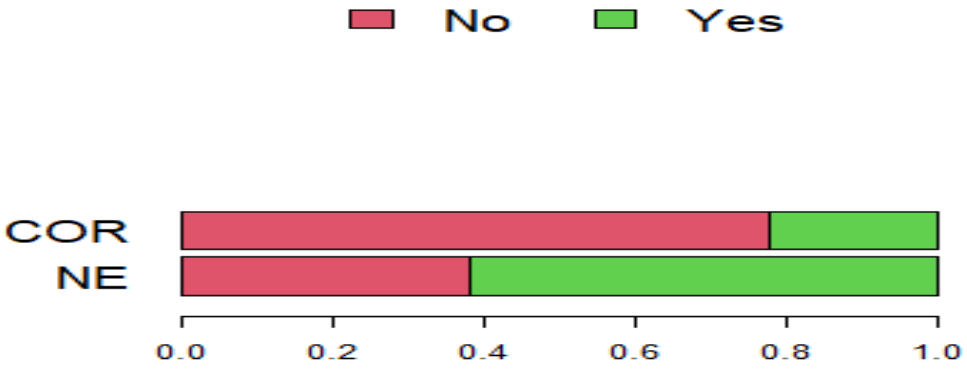
```
> round(t(describe(data_num)),2) # describe numeric variables
           PRICE    SQFT   AGE FEATS
vars        1.00    2.00  3.00  4.00
n          63.00   63.00 63.00 63.00
mean     1158.41 1729.54 17.46  3.95
sd        392.71  506.70  9.60  1.28
median   1049.00 1680.00 20.00  4.00
trimmed  1105.96 1685.18 17.75  3.92
mad       262.42  392.89 11.86  1.48
min       580.00  970.00  2.00  1.00
max      2150.00 2931.00 31.00  8.00
range    1570.00 1961.00 29.00  7.00
skew        1.18    0.74 -0.21  0.45
kurtosis    0.54   -0.16 -1.47  1.12
se         49.48   63.84  1.21  0.16
>
```

```
> round(sapply(data_num, summary),2) # summary numeric variables
          PRICE    SQFT   AGE FEATS
Min.     580.00  970.00  2.00  1.00
1st Qu.  910.00 1400.00  7.00  3.00
Median  1049.00 1680.00 20.00  4.00
Mean    1158.41 1729.54 17.46  3.95
3rd Qu. 1250.00 1920.00 27.50  4.00
Max.    2150.00 2931.00 31.00  8.00
>
```

Firstly, we create density histograms for the numeric variables in order to have first look for their distributions. The probability density function is the blue line while the normal curve is the normal distribution line for each variable.



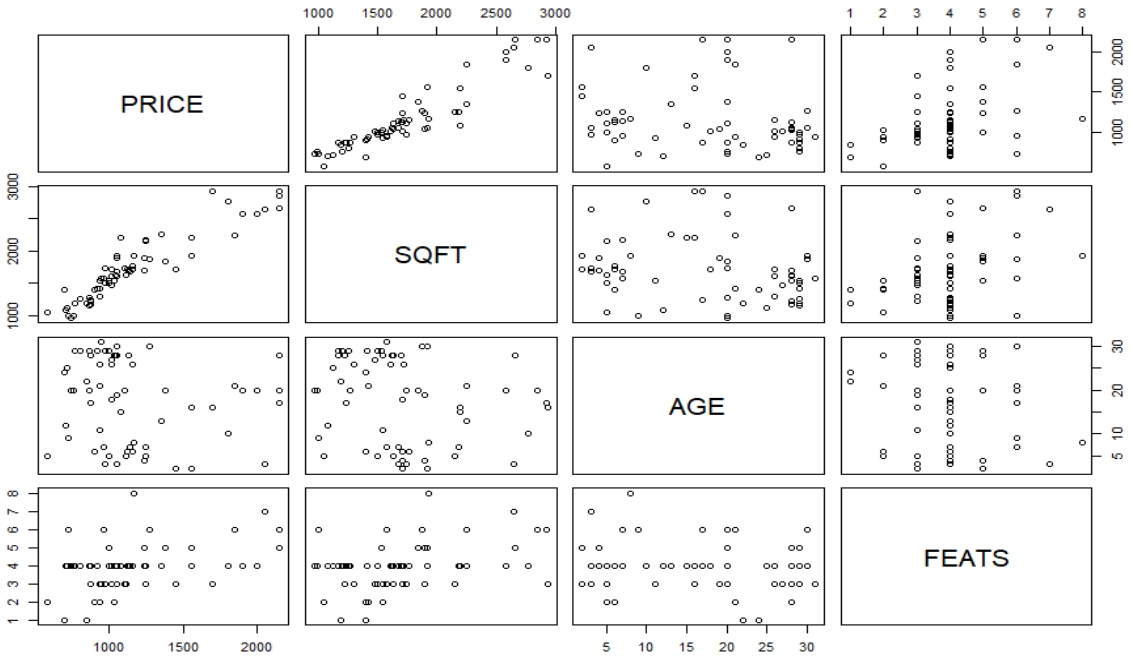
For the factor variables, we create the below barplots to show the frequencies between "yes" and "no" responses in a 0 to 1 scale.



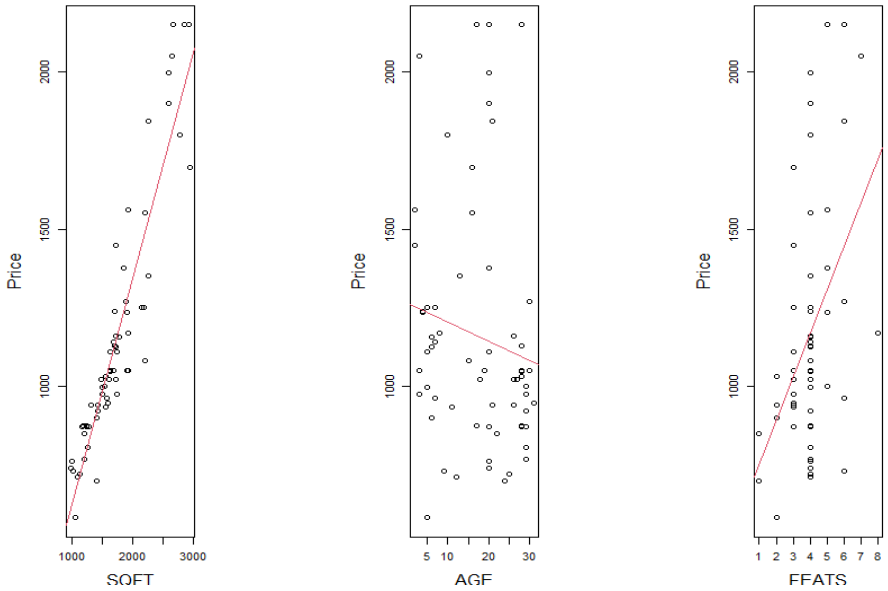
4) **Conduct pairwise comparisons between the variables in the dataset to investigate if there are any associations implied by the dataset. (Hint: Plot variables against one another and use correlation plots and measures for the numerical variables.). Comment on your findings.**
Is there a linear relationship between PRICE and any of the variables in the dataset?

First, we try to see with visuals, the associations between the variables.
Then we will try to see how the price our response relates with the numeric variable and then with the factors. Then we use the Pearson correlation coefficient to measure the linear correlation between the variables.

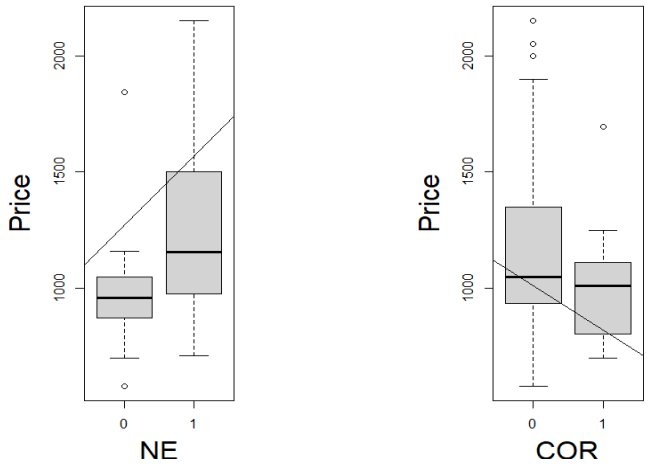
Visualizations of bivariate associations.



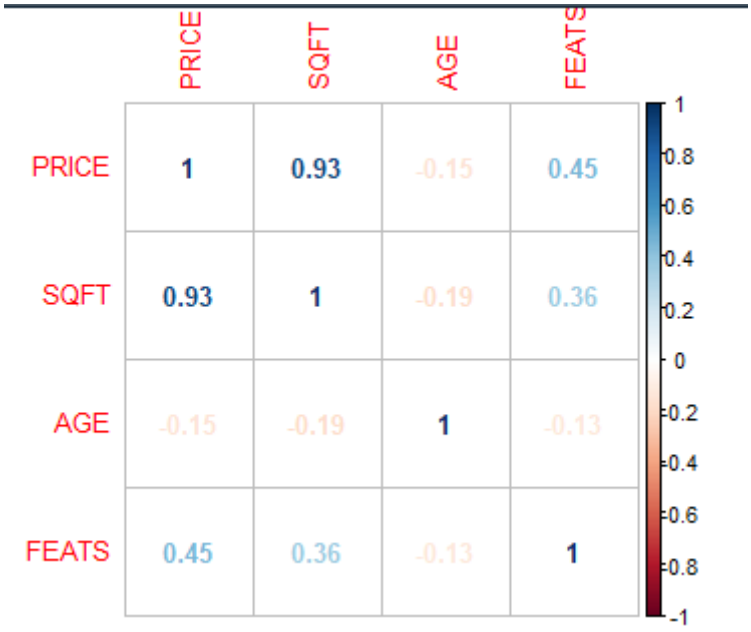
Price (our response) relationships between the numerical variables



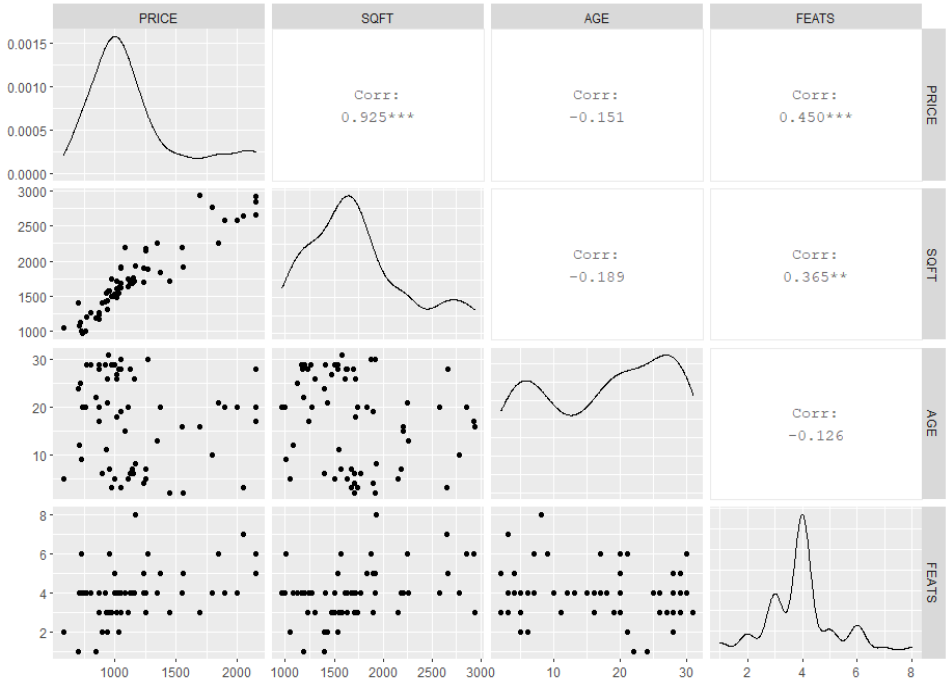
Price (our response) relationships between the factor variables



Correlation matrix



Summary plot of analysis



From the above we can assume a perfect linear relationship between price our response variable and SQFT of the numeric variables since the Pearson coefficient is above 0.7 for this pair and indicates strong linear relationship at **0.93**. This means that the bigger the homes are (since the SQFT is Square Feet of living space) the higher the price will be and vice versa. Furthermore, price with variable FEATS seem to have a positive linear relationship but we cannot assume more since the Pearson correlation value for this pair is 0.45.

From the above , it seems that there is no multicollinearity effect since there is not a strong linear relationship between the predictor variables .

5) **Construct a model for the expected selling prices (PRICE) according to the remaining features. (hint:Conduct multiple regression having PRICE as a response and all the other variables as predictors). Does this linear model fit well to the data? (Hint: Comment on R^2 adj).**

With the help of R and the use of lm function , we will construct a multiple regression model, having PRICE as our response value and the other variables as predictors.

```
> model1 <- lm(formula = data$PRICE~., data = data)
> summary(model1)

Call:
lm(formula = data$PRICE ~ ., data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-416.11  -71.03  -15.26   83.02  347.77

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -193.34926   94.52382  -2.046   0.0454 *
SQFT           0.67662    0.04098  16.509   <2e-16 ***
AGE            2.22907    2.28626   0.975   0.3337
FEATS         34.36573   16.27114   2.112   0.0391 *
NE1           30.00446   47.93940   0.626   0.5339
COR1         -53.07940   46.15653  -1.150   0.2550
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 144.8 on 57 degrees of freedom
Multiple R-squared:  0.8749,    Adjusted R-squared:  0.864
F-statistic: 79.76 on 5 and 57 DF,  p-value: < 2.2e-16
```

Above is the summary of our model1 which contains all variables.
R2 adj value is 0.86, this means that the model1 explains the 86% of the variation of the observed data which is pretty good.

6) **Find the best model for predicting the selling prices (PRICE). Select the appropriate features using stepwise methods. (Hint: Use Forward, Backward or Stepwise procedure according to AIC or BIC to choose which variables appear to be more significant for predicting selling PRICES).**

Since we want to get the best "predictive model" for PRICE we choose to implement the AIC method (Akaike Information Criterion) which is the most appropriate method for this than the BIC. We will do this through the step function using the method "both" in order to implement a stepwise procedure. Below the R output and explanation of procedure.

```
> # AIC
> step(model1, direction= "both")
Start:  AIC=632.62
data$PRICE ~ SQFT + AGE + FEATS + NE + COR

         Df Sum of Sq     RSS    AIC
- NE      1      8218 1203977 631.05
- AGE     1     19942 1215701 631.66
- COR     1     27743 1223502 632.07
<none>                1195759 632.62
- FEATS   1     93580 1289339 635.37
- SQFT    1   5717835 6913594 741.17

Step:  AIC=631.05
data$PRICE ~ SQFT + AGE + FEATS + COR

         Df Sum of Sq     RSS    AIC
- AGE     1     12171 1216147 629.69
- COR     1     25099 1229076 630.35
<none>                1203977 631.05
+ NE      1      8218 1195759 632.62
- FEATS   1    106953 1310930 634.42
- SQFT    1   6288869 7492846 744.24

Step:  AIC=629.69
data$PRICE ~ SQFT + FEATS + COR

         Df Sum of Sq     RSS    AIC
- COR     1     22454 1238602 628.84
<none>                1216147 629.69
+ AGE     1     12171 1203977 631.05
+ NE      1       447 1215701 631.66
- FEATS   1    104259 1320407 632.87
- SQFT    1   6352036 7568184 742.87

Step:  AIC=628.84
data$PRICE ~ SQFT + FEATS

         Df Sum of Sq     RSS    AIC
<none>                1238602 628.84
+ COR     1     22454 1216147 629.69
+ AGE     1      9526 1229076 630.35
+ NE      1       218 1238384 630.83
- FEATS   1    138761 1377363 633.53
- SQFT    1   6389899 7628501 741.37

Call:
lm(formula = data$PRICE ~ SQFT + FEATS, data = data)

Coefficients:
(Intercept)         SQFT        FEATS
  -175.9276       0.6805      39.8369

> |
```

The stepwise regression (or stepwise selection) consists of iteratively adding and removing predictors, in the predictive model, in order to find the subset of variables in the data set resulting in the best predictive model, that is a model that lowers prediction error.

At the beginning our model (with all variables) gets an AIC rate of 632.62. Then the AIC algorithm will start to remove and add variables (the previously removed ones) and try all different combinations of variables in order to find the model with the lowest AIC rate.

So, for starters, the algorithm suggests that the variable NE should be removed from the model since the removal of NE will give the model with the lowest AIC rate (631.05) than the removal of the others.

After the removal of NE, the algorithm continues the procedure and indicates that the removal of AGE will give the model with the lowest AIC rate (629.69).

Finally, in the last part of the output the algorithm suggestion with the "none" choice results in the model with the lowest AIC rate, this means that we should not add or remove anything else.

In our case the stepwise procedure indicates that the best model for predicting the price has predictors the SQFT and FEATS variables with AIC rate of 628.84

The other two procedures are the backward and the forward.

7) **Get the summary of your final model, (the model that you ended up having after conducting the stepwise procedure) and comment on the output. Interpret the coefficients. Comment on the significance of each coefficient and write down the mathematical formulation of the model (e.g. PRICES = Intercept + coef1*Variable1 + coef2*Variable2 +…. + ε where ε ~ N(0, ...)). Should the intercept be excluded from our model?**

The above variable selection stepwise procedure indicated that the best predictive model for PRICE as response has two predictors the SQFT and the FEATS variables respectively.
So below, we construct model2 with these two variables as predictors and we will explain the summary output of this model.

```
> model2<- lm(PRICE ~ SQFT + FEATS, data=data)
> summary(model2)

Call:
lm(formula = PRICE ~ SQFT + FEATS, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-400.44  -71.70  -11.21   93.12  341.82

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -175.92760   74.34207  -2.366   0.0212 *
SQFT           0.68046    0.03868  17.594   <2e-16 ***
FEATS         39.83687   15.36531   2.593   0.0119 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 143.7 on 60 degrees of freedom
Multiple R-squared:  0.8705,    Adjusted R-squared:  0.8661
F-statistic: 201.6 on 2 and 60 DF,  p-value: < 2.2e-16
```

Output explained

Call: Our model formula – lm(PRICE ~ SQFT + FEATS, data = data)

Residuals: Residuals are essentially the difference between the actual observed response values (Price in our case) and the response values that the model predicted. R output shows the summary of the distribution of these values. When we want to have a first look of how well the model fits the data, we should look for a symmetrical distribution across these points. We will do further testing for the residuals later in order to check basic assumptions of our model.

Coefficients - Intercept – estimation:

Intercept or constant is the expected value of our response when all other variables are zero. Graphically it is the point when the regression lies crosses the y-axis.
If the predictors never equal zero then the intercept has no practical meaning and does not tell anything about the relationship between the response and the predictors. When this happens, in order to fix it and give our intercept a meaning we can rescale the predictors to the center of their distribution.
If we do this then the intercept now has a meaning and It is the mean value of Y at the chosen value of X it gives us information about the typical observation.

Coefficients indicate the mean change of our response variable for one-unit change in the predictor variable while holding the other predictors in the model stable.

This value is the **Estimate** column in the above R output.
We have to be careful though how to interpretate this change because we have to consider the units our data are represented.

So, in our case where the response **PRICE** is represented into hundreds of dollars and the SQFT into square feet, a good interpretation would be the following.

Taken into consideration that the range of the SQFT variable is 970 sq. ft to 2931 a logical difference between the houses would be a 100 sq.ft. change.
So, we could say that a difference of 100 sq. ft between two houses could result into a 6800$ dollars in the price in favor of the bigger one.
(0.68x100sqft = 68, then 68 x 100$ = 6800$)

For the FEATS variable, we could say that for every additional feature a house is equipped with then the average change of price in dollars would be 3983$, (39.83x100 = 3983$)

Coefficients - Intercept – significance tests.

Interpretation of the p-values in R output

The p-value for each coefficient estimator tests the null hypothesis that the coefficient is equal to zero where the alternative hypothesis is that the coefficient estimator value is not zero. A low p-value (<0.05) indicates that we cannot reject the null hypothesis thus we should not exclude the predictor from the model. So, a predictor with low p-value seems to be a meaningful addition to the model since the changes in the predictor value will relate into the changes of the response value.

In our case, all predictors including the intercept do not fail the significance test and can be considered statistically significant to the model.

About the intercept, I choose to include it into the model and not to exclude it since it passes the significance test. In this form with the negative value it has no meaning but if we rescale the predictor values with the use of scale function in R then in the “new model” the intercept will represent the price of the typical house.

Mathematical Formula of the model

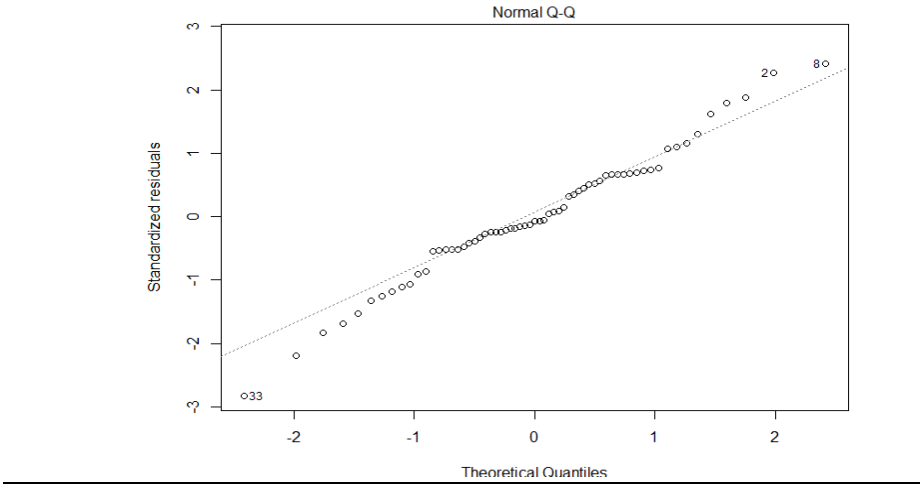
$$\mathbf{PRICE = -175.92 + 0.68 \times SQFT + 39.83 \times FEATS + \varepsilon,\ \varepsilon \sim N(0, 143.7^2) – R2\ adj = 86.61\%}$$

8) Check the assumptions of your final model. Are the assumptions satisfied? If not, what is the impact of the violation of the assumption not satisfied in terms of inference? What could someone do about it?

After conducting the final model, we should check if the model satisfies the multiple linear regression model assumptions. These are the following.

Normality of the residuals

This means that the errors between the observed and the predicted values (residuals of regression) should be normally distributed.



From the first look of the above qq-plot it seems that the residuals satisfy the normality assumption, but we will implement Kolmogorov Smirnov and Shapiro Wilks normality tests in order to be sure.

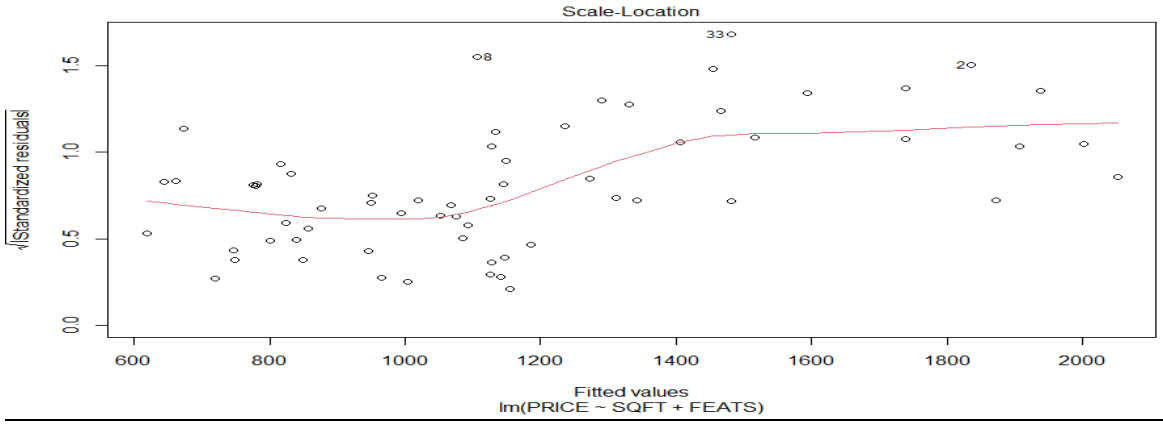
Testing for the normality of residuals Assumption

```
> lillie.test(residuals(model2))

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  residuals(model2)
D = 0.10234, p-value = 0.09854

> shapiro.test(residuals(model2))

        Shapiro-Wilk normality test

data:  residuals(model2)
W = 0.98483, p-value = 0.6303
```

Both tests for normality succeeded. P-value in both tests is greater than the 0.05 value so we do not reject Ho and we assume normality for the residuals.

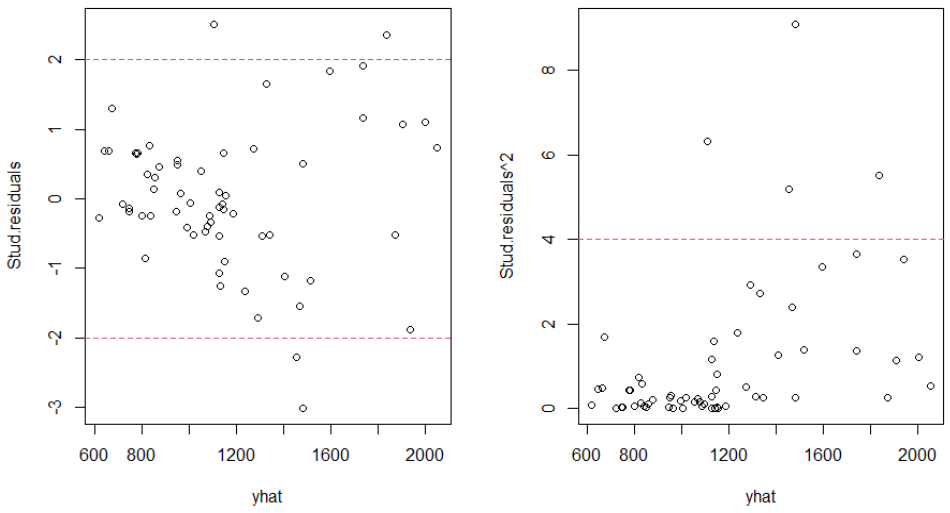
Constant Variance – Homoscedasticity

This assumption means that when you plot the residuals against the predicted values, the variances of the residuals predicted values should be constant.



This plot is a scale-location plot (square rooted standardized residual vs. predicted value). In this particular plot we are checking to see if there is a pattern in the residuals. If the red line is flat and horizontal with equally and randomly spread data points then the assumption is satisfied. If the red line has a positive slope to it, or if your data points are not randomly spread out, the model violates this assumption. In our case it is clear that the model does not satisfies the homoscedasticity assumption.

In the below diagrams we see the residuals that they are thin spread, in the left plot we would like the points to be as close as possible to center and, in the left as close as possible down to zero (because they are the squared residuals).



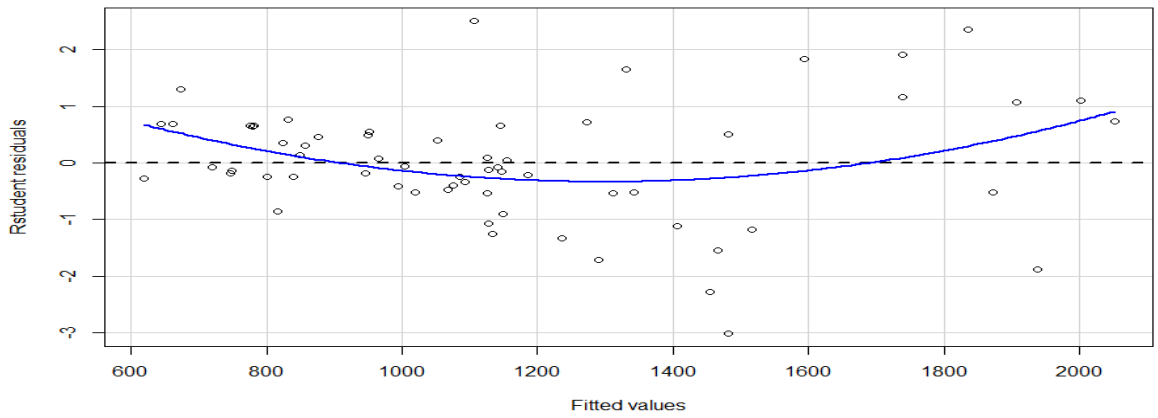
Testing the Homoscedasticity Assumption

```
> library(car)
> ncvTest(model2)
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 14.99402, Df = 1, p = 0.00010785
> yhat.quantiles<-cut(yhat, breaks=quantile(yhat, probs=seq(0,1,0.25)), dig.lab=6)
> table(yhat.quantiles)
yhat.quantiles
(618.914,853.023]  (853.023,1126.6]  (1126.6,1336.06] (1336.06,2050.73]
               15                17                 14                16
> leveneTest(rstudent(model2)~yhat.quantiles)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value    Pr(>F)
group  3  9.9191 2.249e-05 ***
      58
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

In order to check further the homoscedasticity assumption, we implement ncvTest and Levene tests.

Both tests fail p-values <0.05 so we can assume that we have unequal variances and we face and Heteroscedasticity problem.

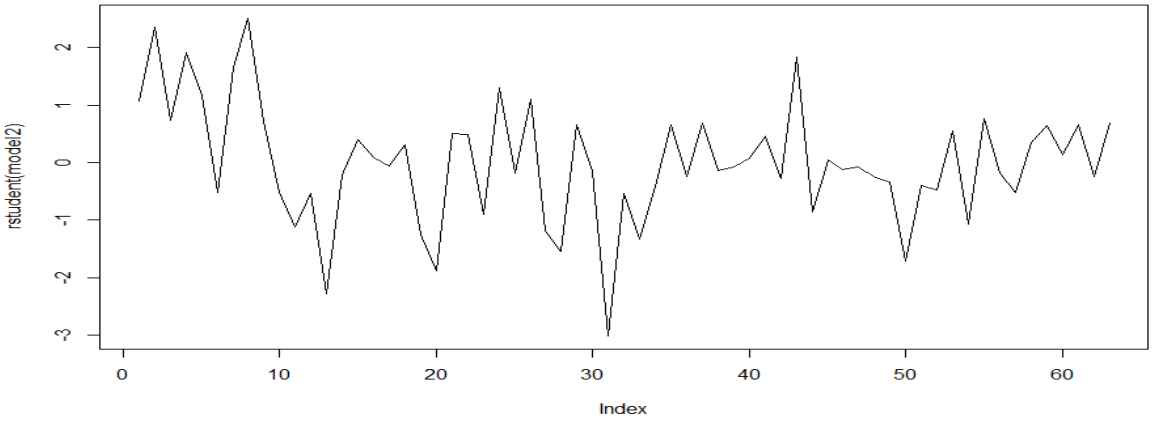
Non-Linearity Assumption



From the above plot it is clear that there the non-Linearity assumption is violated.
We would expect a straight line and the points to be close to it to satisfy the linearity assumption.

Independence of errors – Autocorrelation

Finally, we check the last assumption about the independence of errors.
From the plot we that there is no stability in the distribution of the errors so we can first assume that the assumption is no violated.



Testing the Independence of Errors Assumption

```
> durbinWatsonTest(model2)
 lag Autocorrelation D-W Statistic p-value
   1       0.2012826       1.573363    0.07
 Alternative hypothesis: rho != 0
>
```

We implement Durbin-Watson test to check further the autocorrelation assumption, the null hypothesis for this test is that there is no autocorrelation between the errors while the alternative indicates the opposite.

From the above result test, we see that D-W statistic is 1.57 (values closer to 2 are better) and p-value is 0.07 is greater than 0.05 so we do not reject the null hypothesis (no autocorrelation) thus we can assume there is no autocorrelation.

Finally, from the above we can say that the model satisfies or dissatisfies the below assumptions.

1. Normality of the residuals - Satisfies
2. Constant of variance (Homoscedasticity) – Fails
3. Non linearity – Fails
4. Independence of errors (Autocorrelation) – Satisfies

In order to fix the above we can try to implement log transformations into the model.
If we do this, the new model satisfies both the above assumptions which previously failed.

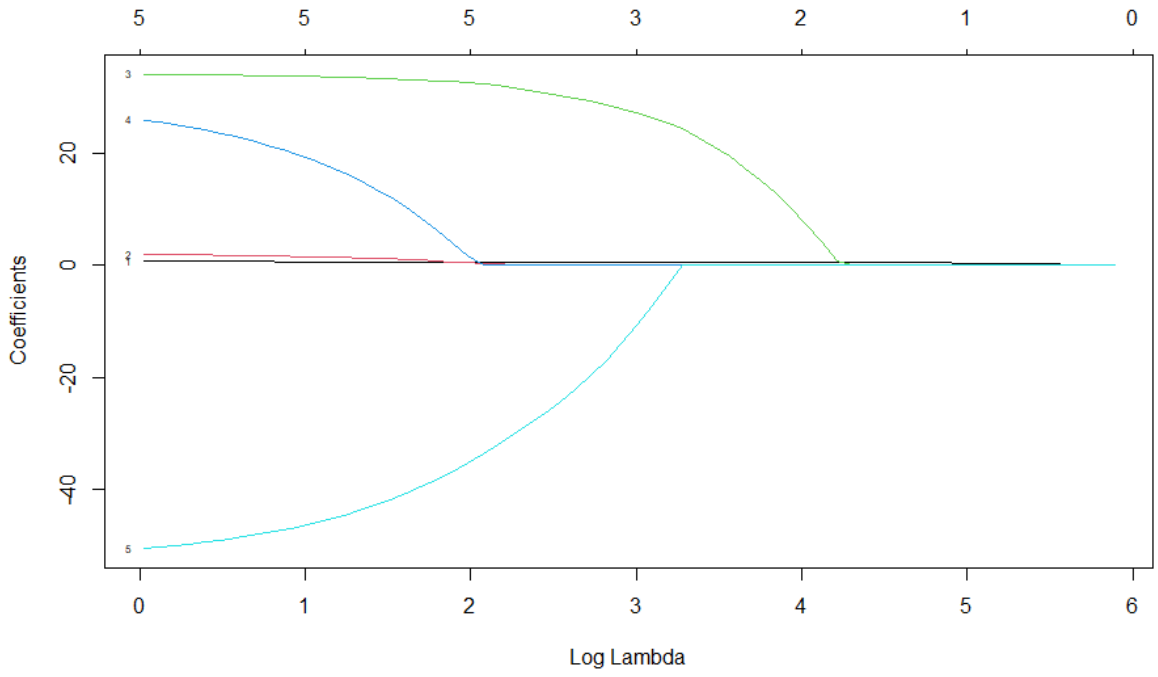
9. **Conduct LASSO as a variable selection technique and compare the variables that you end up having using LASSO to the variables that you ended up having using stepwise methods in (VI). Are you getting the same results? Comment.**

Lasso is a regression analysis method that performs both variable selection and regularization. What LASSO does is to force the sum of the absolute value of the coefficients to be less than a fixed value which results certain coefficients set to zero, removing them from the model, this makes the model simpler and more interpretable. The regularization parameter is λ and measures the degree where the coefficients are penalized.

We fit the model into an R object, using the glmnet function.
As input, we load our predictors data as input X and the response as a vector

```
> library(glmnet)
> X<- model.matrix(model1)[,-1] # lasso model fitting
> lasso <- glmnet(X, data$PRICE)
> par(mfrow=c(1,1))
> plot(lasso, xvar = "lambda", label = T)
>
```

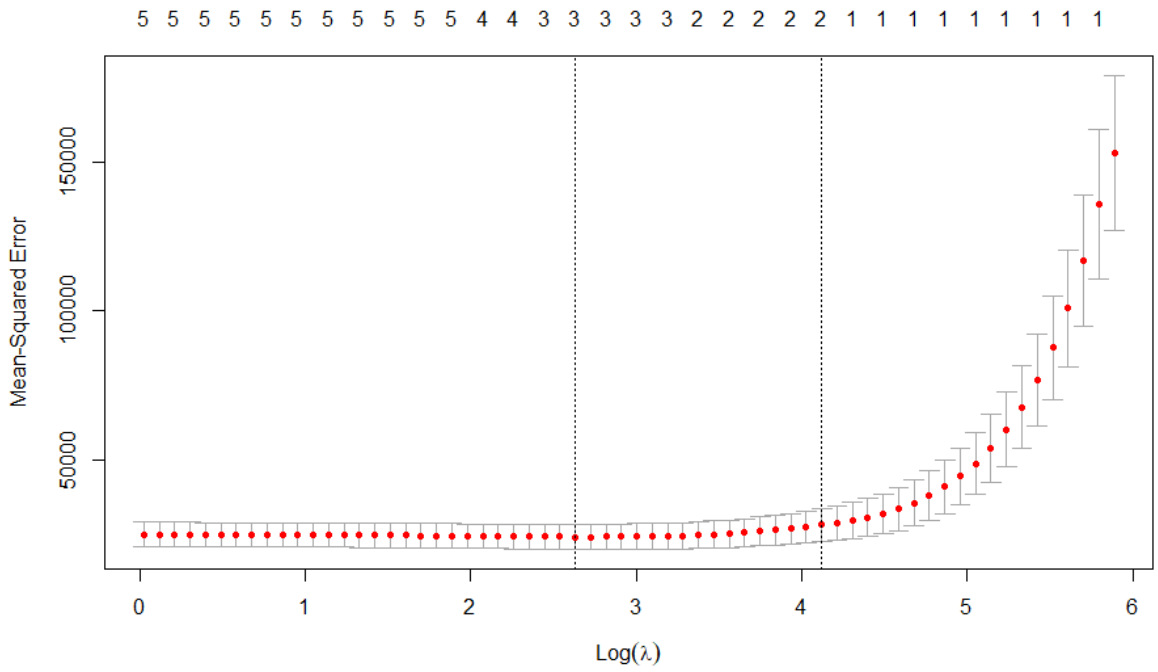
. Then we visualize the coefficients using the plot function.



Each curve corresponds to a variable. It shows the path of its coefficient against the ℓ1-norm of the whole coefficient vector for different levels of λ .
The axis above indicates the number of nonzero coefficients at the current λ
Lambda is the weight given to the regularization term (the L1 norm), so as lambda approaches zero, the loss function of your model approaches the OLS loss function
When lambda is very small, all coefficients are in the model.
As lambda grows, the regularization term has greater effect and fewer variables will be in the model (because more and more coefficients will be zero valued).

Furthermore, we do the cross-validation method in order to select the best λ for our model.
We perform cross-validation with the use of the cv.glmnet function.
We define the elastic net "a" value to 1 , because we want to perform the LASSO method.

We plot the object with all possible values of λ .



We see the two vertical lines, that the algorithm suggests for the value of λ.

The lambda.min is the value of λ that results into minimum error.

The other value, lambda.1se gives the most regularized model within the same level of mean squared error with the lambda.min

We test the two values of **λ** in order to get the results.

```
> # Rebuilding the model with  lamda values min and 1se respectively
> lasso_min <- glmnet(X,data$PRICE, alpha = 1, lambda = lasso1$lambda.min)
> lasso_1se <- glmnet(X,data$PRICE, alpha = 1, lambda = lasso1$lambda.1se)
> coefficients(lasso_min)
6 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept) -84.8228785
SQFT          0.6555256
AGE           .
FEATS        28.6552998
NE1           .
COR1        -17.0038257
> coefficients(lasso_1se)
6 x 1 sparse Matrix of class "dgCMatrix"
                     s0
(Intercept) 93.770884
SQFT         0.598732
AGE          .
FEATS        7.365442
NE1          .
COR1         .
>
```

Finally, we choose the lambda.1se value because it results into a simpler model with least coefficients and because it does not differ from the lambda.min value in terms of MSE .

We get the same model with the AIC variable selection method, although the coefficients differ.

R CODE – REFERENCE

```
# task 1
data<- read.table(file.choose())
str(data)
summary(data)

# task 2
data[1:4]<- lapply(data[1:4], as.numeric) # columns 1 to 4 are the numeric
data[5:6]<- lapply(data[5:6], as.factor) # columns 5 to 6 are the factors
str(data)

# task 3
library(psych)
num_only<- sapply(data, class) == "numeric" # in order to separate the numeric
data_num<- data[ , num_only]
data_fac<- data[ , !num_only] # in order to separate the factors
round(t(describe(data_num)),2) # describe numeric variables
round(sapply(data_num, summary),2) # summary numeric variables
multi.hist(data_num,dcol= c("blue","red"),dlty=c("dotted", "solid")) # for numeric
- density histograms
n<- nrow(data_num)
par(mai=c(0.5,1.5,0.5,0.5)) # for factors - barplots
barplot(sapply(data_fac,table)/n, horiz=T, las=1, col=2:3, ylim=c(0,8),
cex.names=1.5)
legend('top', fil=2:3, legend=c("No","Yes"), ncol=2, bty='n',cex=1.5)
```

```r
# task 4
library(ggplot2)
library(GGally)
pairs(data_num) # Visualization of bivariate associations
ggpairs(data_num) # better summary graph

par(mfrow=c(1,3))
#Price (our response) on numeric variables
for(j in 2:4){
  plot(data_num[,j], data_num[,1], xlab=names(data_num)[j],
ylab='Price',cex.lab=1.5)
  abline(lm(data_num[,1]~data_num[,j]),col=2)
}

par(mfrow=c(1,2))
#Price (our response) on factor variables
for(j in 1:2){
  boxplot(data_num[,1]~data_fac[,j], xlab=names(data_fac)[j],
ylab='Price',cex.lab=2.0)
  abline(lm(data_num[,1]~data_fac[,j]))
}

library(corrplot)
corrplot(round(cor(data_num),2), method = "number") # correlation matrix
round(cor(data_num),2)
cor(data_num$PRICE, data_num$SQFT, method = "pearson") # correlation - pearson


# task 5
model1 <- lm(formula = data$PRICE~., data = data) # Fitting the regression model -
all variables
summary(model1)

# task 6
step(model1, direction= "both") # AIC

# task 7
model2<- lm(PRICE ~ SQFT + FEATS, data=data) # model with lowest AIC rate
summary(model2)

# task 8
# check normality of the residuals
library(nortest)
par(mfrow=c(1,1))
plot(model2, which=2)
lillie.test(residuals(model2))
shapiro.test(residuals(model2))

# check constant variance (homoscedasticity) assumption with plots
par(mfrow=c(1,1))
plot(model2, which=3)
Stud.residuals<-rstudent(model2)
yhat <- fitted(model2)
par(mfrow=c(1,2))
plot(yhat, Stud.residuals) # stud.residuals
abline(h=c(-2,2), col=2, lty=2)
plot(yhat, Stud.residuals^2) # squared stud.residuals
abline(h=4, col=2, lty=2)

# check constant variance (homoscedasticity) assumption with tests
library(car)
ncvTest(model2)
yhat.quantiles<-cut(yhat, breaks=quantile(yhat, probs=seq(0,1,0.25)), dig.lab=6)
table(yhat.quantiles)
leveneTest(rstudent(model2)~yhat.quantiles)
```

```r
# non linearity assumption
par(mfrow=c(1,1))
library(car)
residualPlot(model2, type='rstudent')

# Independence of errors assumption
plot(rstudent(model2), type='l')
durbinWatsonTest(model2)

# task 9
library(glmnet)
X<- model.matrix(model1)[,-1] # lasso model fitting
lasso <- glmnet(X, data$PRICE)
par(mfrow=c(1,1))
plot(lasso, xvar = "lambda", label = T)

# cross valitation to find a reasonable value for lambda
lasso1<- cv.glmnet(X, data$PRICE, alpha = 1)
lasso1$lambda
lasso1$lambda.min # lambda min
lasso1$lambda.1se # lambda.1se
plot(lasso1)

# Rebuilding the model with  lamda values min and 1se respectively
lasso_min <- glmnet(X,data$PRICE, alpha = 1, lambda = lasso1$lambda.min)
lasso_1se <- glmnet(X,data$PRICE, alpha = 1, lambda = lasso1$lambda.1se)
coefficients(lasso_min)
coefficients(lasso_1se)
```