Regression Assignment

Akash C R

2023-04-27

#Question 1

```
library('xlsx')
qn1 = read.csv("housing.csv")
```

We can see that ocean_proximity is having string variables. Lets convert it to numericals before we perform the correlation analysis.

```
unique(qn1$ocean_proximity)
```

```
## [1] "NEAR BAY"   "<1H OCEAN"  "INLAND"     "NEAR OCEAN" "ISLAND"
```

Among these 5 unique values, Near Bay, Near Ocean mean the same thing. So, we can map it to the same number.

```
qn1$ocean_proximity[qn1$ocean_proximity == 'NEAR BAY'] = 1
qn1$ocean_proximity[qn1$ocean_proximity == 'NEAR OCEAN'] = 1
qn1$ocean_proximity[qn1$ocean_proximity == '<1H OCEAN'] = 2
qn1$ocean_proximity[qn1$ocean_proximity == 'INLAND'] = 3
qn1$ocean_proximity[qn1$ocean_proximity == 'ISLAND'] = 4

qn1$ocean_proximity = as.numeric(qn1$ocean_proximity)
```

```
#while plotting correlation map we got NA values corresponding to total_bedrooms which #essentially mea
nacount = colSums(is.na(data.frame(qn1$total_bedrooms)))
print(nacount)
```

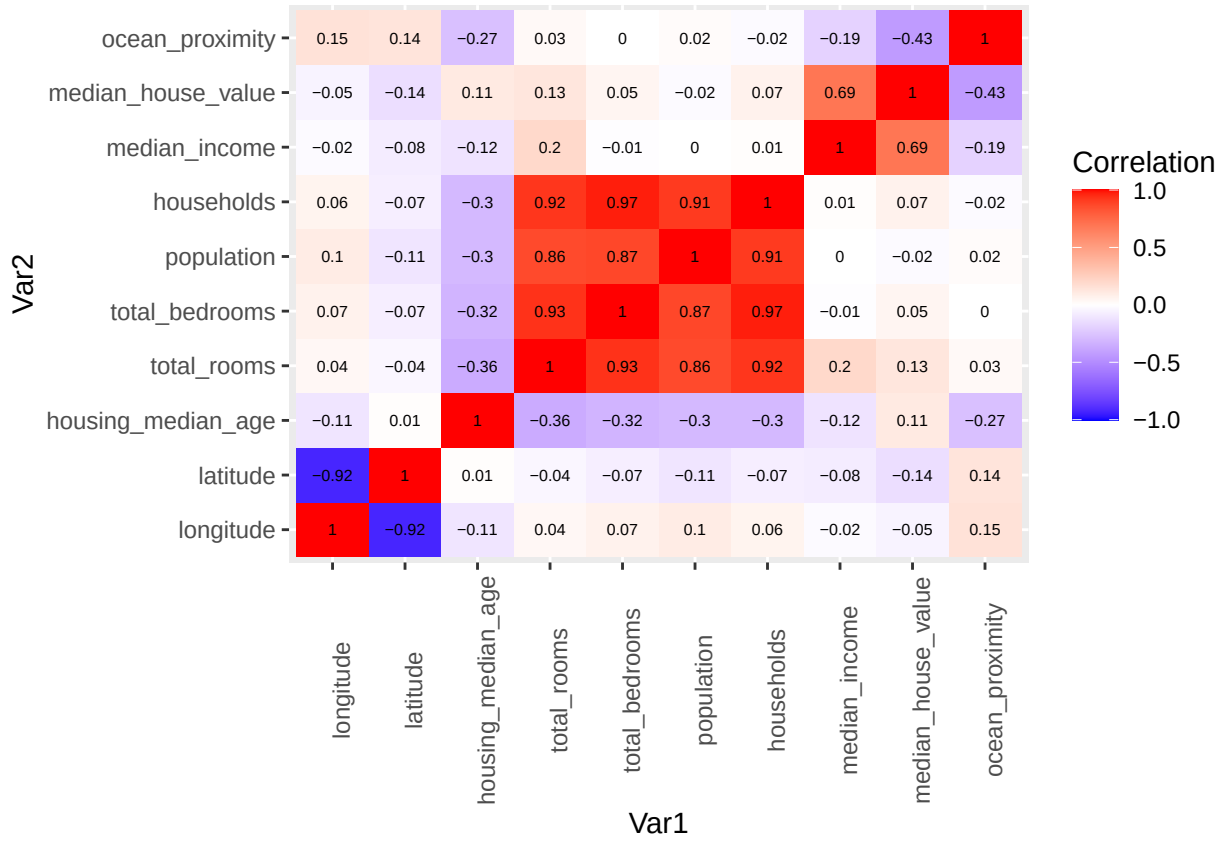
```
## qn1.total_bedrooms
##                207
```

```
#so replace them with avg value
meanval = mean(qn1$total_bedrooms, na.rm = TRUE)
qn1$total_bedrooms = ifelse(is.na(qn1$total_bedrooms), meanval, qn1$total_bedrooms)
```

Visualise the correlation between variables in the data set.

```
library(reshape2)
cor1 = round(cor(qn1), 2)
melted_cor = melt(cor1)

library(ggplot2)
ggplot(data = melted_cor, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  geom_text(aes(label = value), size = 2) +
  scale_fill_gradient2(low = "blue", high = "red",
                       limit = c(-1,1), name="Correlation") +
  theme(axis.text.x = element_text(angle = 90))
```



We can clearly see some of the variables are highly correlated, now lets perform a correlation test to confirm the collinearity before building the model.

```
#true correlation is greater than 0
cor.test(qn1$total_bedrooms, qn1$total_rooms, alternative = "greater")
```

```
##
##  Pearson's product-moment correlation
##
## data:  qn1$total_bedrooms and qn1$total_rooms
## t = 355.76, df = 20638, p-value < 2.2e-16
## alternative hypothesis: true correlation is greater than 0
## 95 percent confidence interval:
```

```
##  0.9256302 1.0000000
## sample estimates:
##       cor
## 0.9272527
```

```
cor.test(qn1$households, qn1$population, alternative = "greater")
```

```
##
##  Pearson's product-moment correlation
##
## data:  qn1$households and qn1$population
## t = 309.83, df = 20638, p-value < 2.2e-16
## alternative hypothesis: true correlation is greater than 0
## 95 percent confidence interval:
##  0.905175 1.000000
## sample estimates:
##       cor
## 0.9072223
```

```
cor.test(qn1$longitude, qn1$latitude, alternative = "less")
```

```
##
##  Pearson's product-moment correlation
##
## data:  qn1$longitude and qn1$latitude
## t = -348.85, df = 20638, p-value < 2.2e-16
## alternative hypothesis: true correlation is less than 0
## 95 percent confidence interval:
##  -1.0000000 -0.9229865
## sample estimates:
##        cor
## -0.9246644
```

Now from the above correlation tests we can see whenever p-value is less than 0.05, it is significant correlation and hence using one of such two variables is enough while building the model.

Pick 2 linear regression models to predict median house value.

Regression model 1.

```
f1 = median_house_value ~ longitude + housing_median_age + total_rooms + households + median_income + o

r1 = lm(f1, data = qn1)
summary(r1)
```

```
##
## Call:
## lm(formula = f1, data = qn1)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -559071  -44980  -13072   30571  485322
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.536e+05  3.138e+04   4.894 9.94e-07 ***
## longitude           6.056e+02  2.596e+02   2.333   0.0196 *
## housing_median_age  1.196e+03  4.573e+01  26.156  < 2e-16 ***
## total_rooms        -1.052e+01  6.997e-01 -15.033  < 2e-16 ***
## households          8.233e+01  3.880e+00  21.220  < 2e-16 ***
## median_income       4.198e+04  3.198e+02 131.255  < 2e-16 ***
## ocean_proximity    -4.064e+04  7.607e+02 -53.416  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 73560 on 20633 degrees of freedom
## Multiple R-squared:  0.5938, Adjusted R-squared:  0.5937
## F-statistic:  5027 on 6 and 20633 DF,  p-value: < 2.2e-16
```

Regression model 2

```
f2 = median_house_value ~ latitude + housing_median_age + total_bedrooms + population + median_income +

r2 = lm(f2, data = qn1)
summary(r2)
```

```
##
## Call:
## lm(formula = f2, data = qn1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -538413  -43353  -10613   30829  614372
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         2.171e+05  8.901e+03   24.39   <2e-16 ***
## latitude           -3.547e+03  2.384e+02  -14.88   <2e-16 ***
## housing_median_age  1.288e+03  4.441e+01   29.01   <2e-16 ***
## total_bedrooms      9.879e+01  2.486e+00   39.73   <2e-16 ***
## population         -3.067e+01  9.138e-01  -33.56   <2e-16 ***
## median_income       3.981e+04  2.728e+02  145.93   <2e-16 ***
## ocean_proximity    -4.054e+04  7.261e+02  -55.83   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 71720 on 20633 degrees of freedom
## Multiple R-squared:  0.6139, Adjusted R-squared:  0.6137
## F-statistic:  5467 on 6 and 20633 DF,  p-value: < 2.2e-16
```

Regression 3

We can see that ocean_proximity and median_income are highly correlated with median house value where ocean_proximity is negatively correlated.

```
f3 = median_house_value ~ median_income + ocean_proximity

r3 = lm(f3, data = qn1)
summary(r3)
```

```
##
## Call:
## lm(formula = f3, data = qn1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -504739  -44692  -12973   29313  467485
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     161553.6     2077.0   77.78   <2e-16 ***
## median_income    38216.2      281.8  135.62   <2e-16 ***
## ocean_proximity -49380.5      720.6  -68.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 75580 on 20637 degrees of freedom
## Multiple R-squared:  0.5711, Adjusted R-squared:  0.571
## F-statistic: 1.374e+04 on 2 and 20637 DF,  p-value: < 2.2e-16
```

So we built three linear regression models using only one among highly correlated variable thus removing three dimensions in both of the models. And in third model we used only 2 variables which are having high absolute correlation values. And in all the cases the p-value is less than 0.05 and hence we can say our model fits well to the data.

Check for collinearity using VIF to remove highly correlated variables from the model.

VIF is used to check multicollinearity, so if VIF is above 5 then it indicates high multicollinearity.

```
library('car')

vif(r1)
```

```
##          longitude housing_median_age        total_rooms         households
##           1.031655           1.263301           8.887635           8.393098
##      median_income    ocean_proximity
##           1.408296           1.218576
```

We know that VIF higher than 5 is bit problematic. So in our case we can remove any one of the total_rooms and households which are highly correlated to each other to reduce the multicollinearity.

Update regression1 model1

```
f1 = median_house_value ~ longitude + housing_median_age + households + median_income + ocean_proximity

r1_modified = lm(f1, data = qn1)
summary(r1_modified)
```

```
##
## Call:
## lm(formula = f1, data = qn1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -534939  -45341  -13015   30789  479074
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        189879.306  31456.608   6.036  1.6e-09 ***
## longitude             817.682    260.593   3.138   0.0017 **
## housing_median_age   1279.882     45.631  28.049  < 2e-16 ***
## households             28.030      1.424  19.677  < 2e-16 ***
## median_income       39629.281    280.558 141.252  < 2e-16 ***
## ocean_proximity    -42972.233    748.735 -57.393  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 73960 on 20634 degrees of freedom
## Multiple R-squared:  0.5893, Adjusted R-squared:  0.5892
## F-statistic:  5923 on 5 and 20634 DF,  p-value: < 2.2e-16
```

```
vif(r1_modified)
```

```
##          longitude housing_median_age         households      median_income
##           1.028609           1.244484           1.119253           1.072017
##    ocean_proximity
##           1.167709
```

As we can see from the vif scores, after removing total_rooms feature, multicollinearity of all variables are significantly less than 5.

Now lets try the same for Regression Model 2.

```
vif(r2)
```

```
##           latitude housing_median_age     total_bedrooms         population
##           1.040785           1.253307           4.360473           4.296936
##      median_income    ocean_proximity
##           1.077909           1.167920
```

And since all the values are less than 5, there is no need to remove the features.

Lets try the same for Regression Model 3.

```
vif(r3)
```

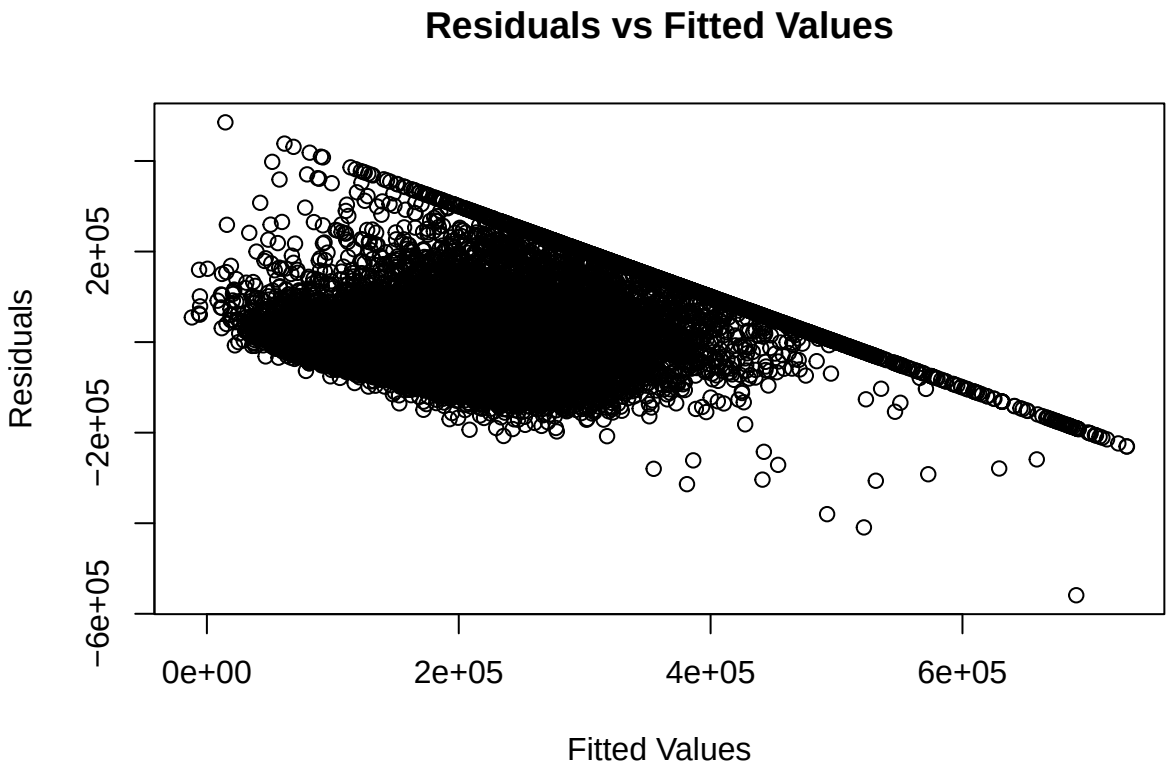
```
##   median_income ocean_proximity
##        1.035542        1.035542
```

Regression model 3 also does not have problem of multicollinearity so no need to remove any features.

Plot the distribution of residuals against the fitted values to check for heteroscedasticity.

Lets try for the Regression model 1.

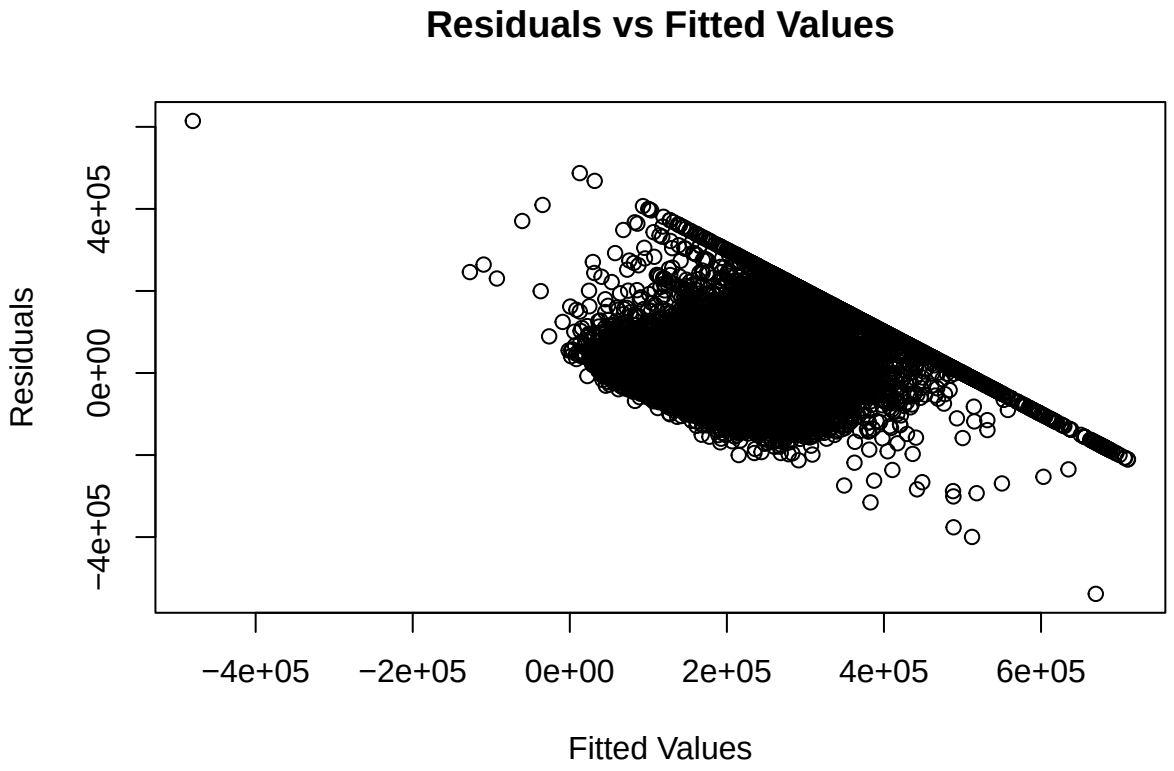
```
my_resid = resid(r1)
my_fitted = fitted(r1)
plot(my_fitted, my_resid, main = "Residuals vs Fitted Values", xlab = "Fitted Values", ylab = "Residuals
```



Since plot of residuals against fitted values is not constant, it means that there is heteroscedasticity in our data.

Now lets try the same for Model2.

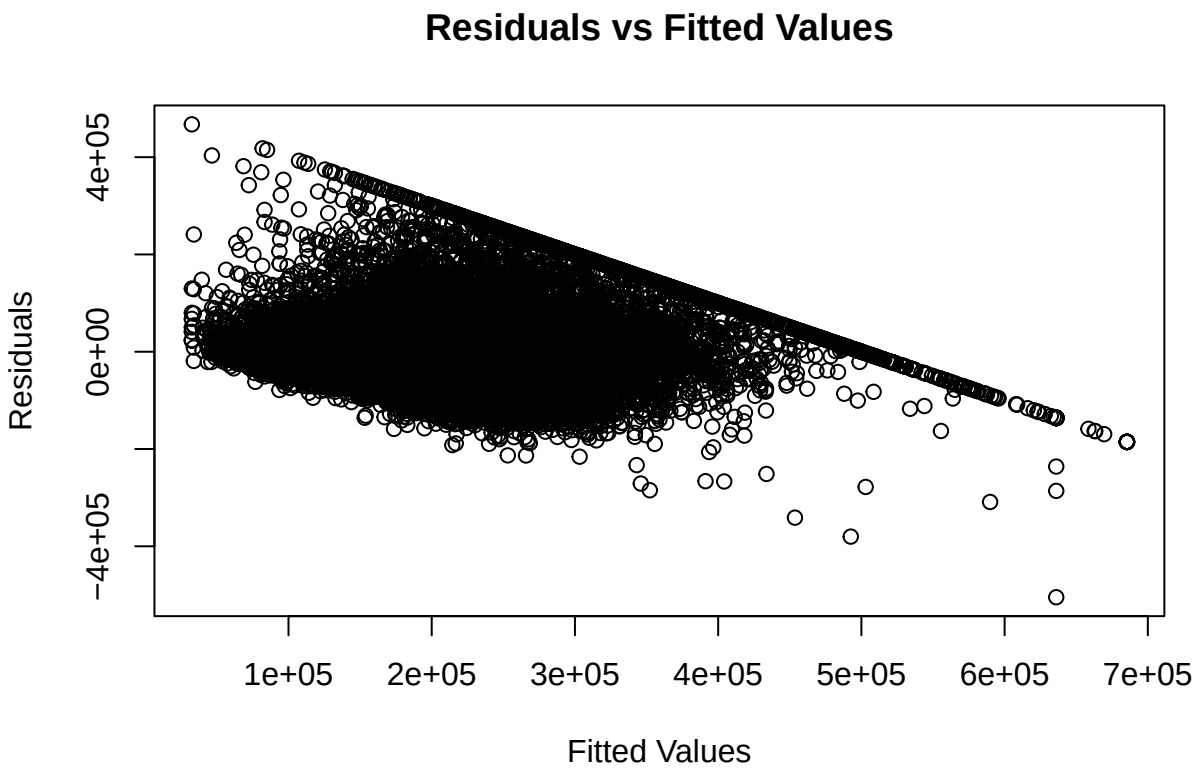
```
my_resid = resid(r2)
my_fitted = fitted(r2)
plot(my_fitted, my_resid, main = "Residuals vs Fitted Values", xlab = "Fitted Values", ylab = "Residuals
```



Again we can say that the data has heteroscedasticity.

Now lets try for model3.

```
my_resid = resid(r3)
my_fitted = fitted(r3)
plot(my_fitted, my_resid, main = "Residuals vs Fitted Values", xlab = "Fitted Values", ylab = "Residuals
```



This has the same result again.

So all three models have significant heteroscedasticity.

Use ncvTest or equivalent to test for heteroscedasticity

```
ncvTest(r1)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1230.522, Df = 1, p = < 2.22e-16
```

We can clearly see that p value is far less that 0.05 which indicates that null hypothesis of constant variance is rejected and there is a evidence of heteroscedasticity in the residuals.

Now lets perform the same on model2.

```
ncvTest(r2)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1177.469, Df = 1, p = < 2.22e-16
```

Again a very small p-value indicating there is evidence of heteroscedasticity in the residuals, meaning that the variance of the residuals is not constant across the range of fitted values.

```
ncvTest(r3)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1014.636, Df = 1, p = < 2.22e-16
```

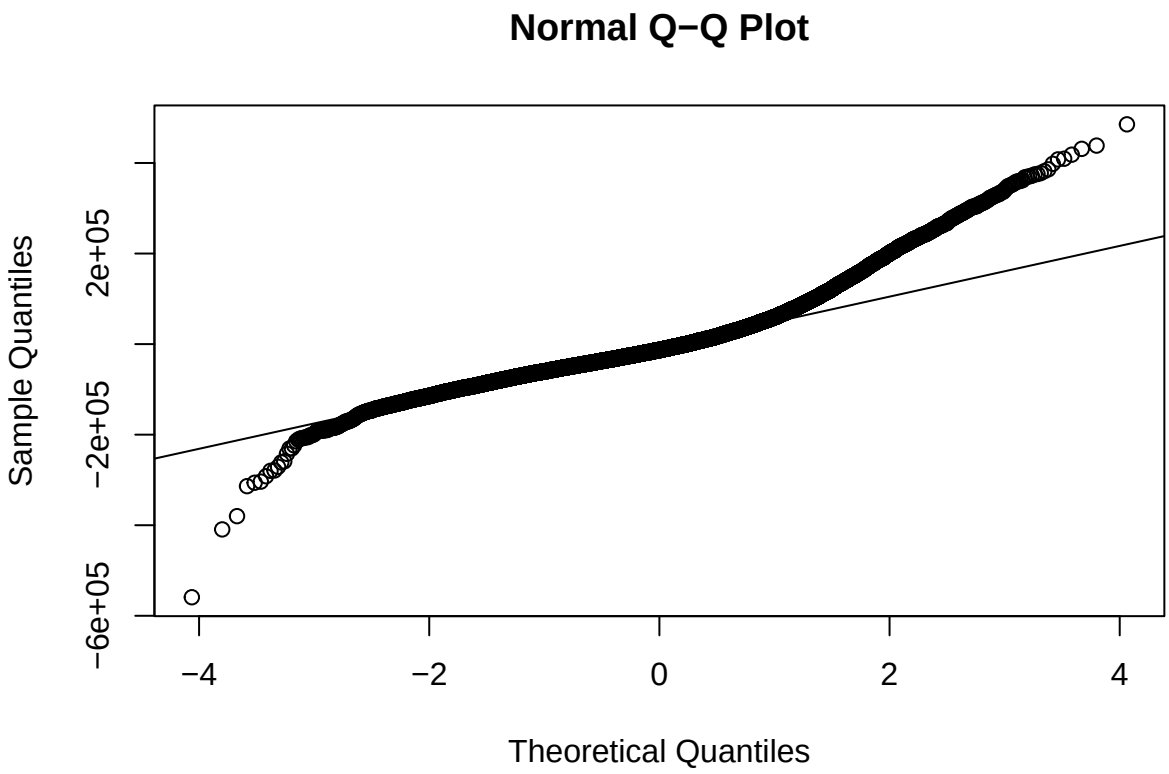
Again the proof of heteroscedasticity. So we can clearly see that both the tests ncvTest as well as plot of residuals vs fitted are consistent and provide evidence for the presence of heteroscedasticity.

Test for normality of the residuals.

Lets use qqplot to determine the normality of the residuals.

For model 1

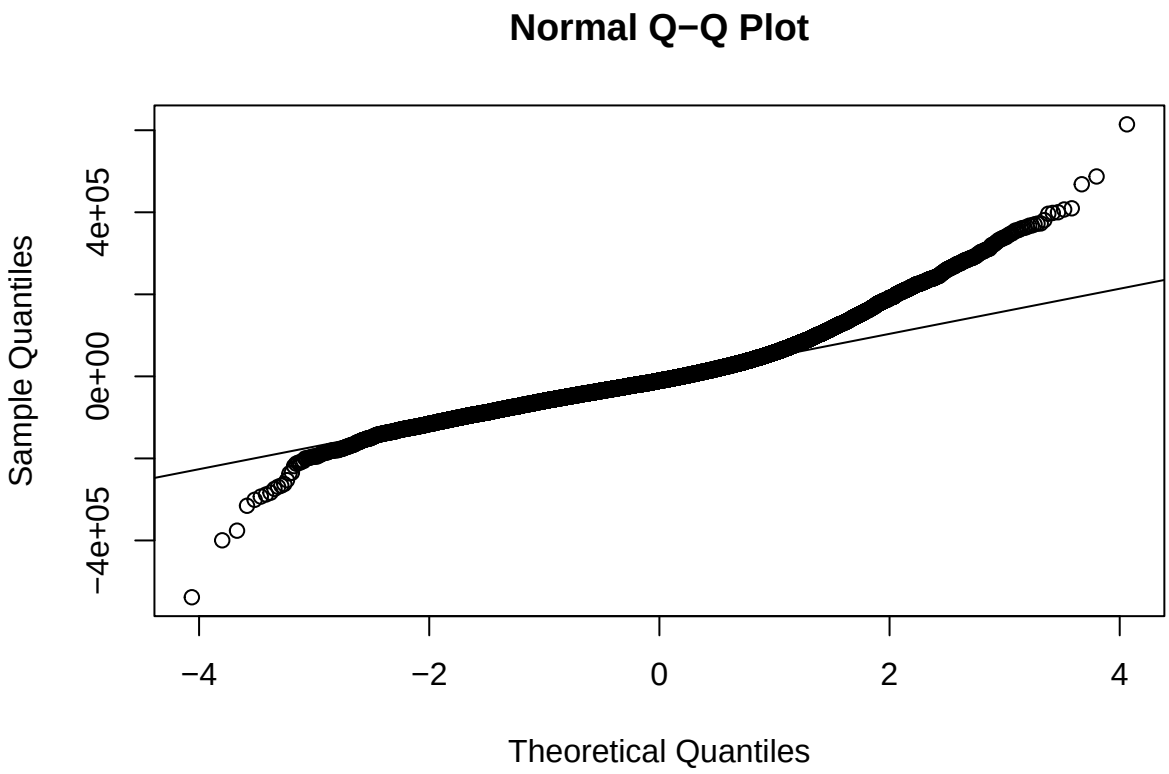
```
qqnorm(r1$residuals)
qqline(r1$residuals)
```



As we can cleary see from the graph that points deviate from the straight line significanlty, we can say that the residuals are not normally distributed.

For model2

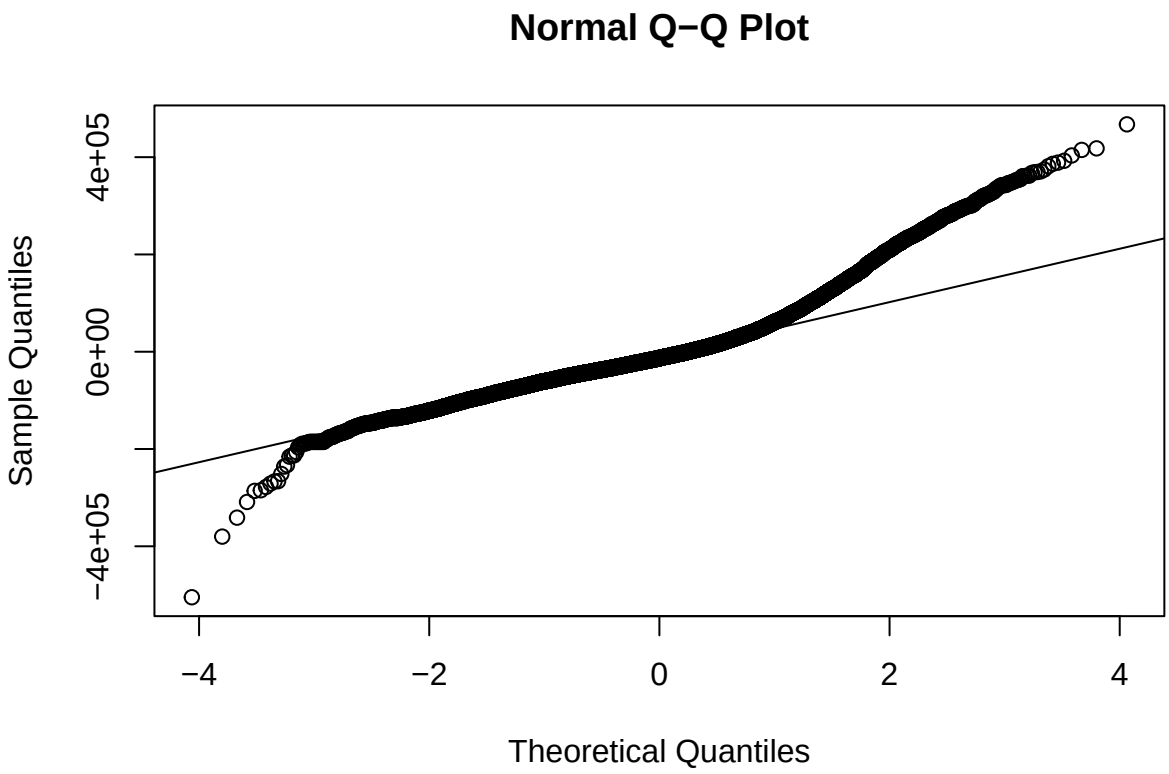
```
qqnorm(r2$residuals)
qqline(r2$residuals)
```



Again the residuals are not normally distributed.

Now testing for model 3

```
qqnorm(r3$residuals)
qqline(r3$residuals)
```



Again we can see the deviation and say that the residuals are not normally distributed.

So we can reject the null hypothesis that our data came from normally distributed data.

Compare two models using AIC and choose the best model.

```
AIC(r1)
```

```
## [1] 521158.8
```

```
AIC(r2)
```

```
## [1] 520113.5
```

```
AIC(r3)
```

```
## [1] 522274.9
```

As we can see from the above AIC score, model 2 has the lowest AIC. So, we can choose model2 be the better model among the three.

Report the coefficients of the winning model and their statistics (including confidence intervals) and interpret the resulting model coefficients.

The model 2 is the best model.

```
summary(r2)
```

```
##
## Call:
## lm(formula = f2, data = qn1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -538413  -43353  -10613   30829  614372
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         2.171e+05  8.901e+03   24.39   <2e-16 ***
## latitude           -3.547e+03  2.384e+02  -14.88   <2e-16 ***
## housing_median_age  1.288e+03  4.441e+01   29.01   <2e-16 ***
## total_bedrooms      9.879e+01  2.486e+00   39.73   <2e-16 ***
## population         -3.067e+01  9.138e-01  -33.56   <2e-16 ***
## median_income       3.981e+04  2.728e+02  145.93   <2e-16 ***
## ocean_proximity    -4.054e+04  7.261e+02  -55.83   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 71720 on 20633 degrees of freedom
## Multiple R-squared:  0.6139, Adjusted R-squared:  0.6137
## F-statistic:  5467 on 6 and 20633 DF,  p-value: < 2.2e-16
```

R-square value of 0.61 says that the model explains 61% of the variation in the response variable.

And also higher t-values whose absolute value is greater than 2 imply that the estimate of regression coefficient is significant. Higher the absolute t-value higher the significance that variables are related.

Confidence interval of 95%

```
confint(object = r2, level=0.95)
```

```
##                          2.5 %       97.5 %
## (Intercept)        199623.01991 234514.67356
```

```
## latitude            -4014.37036  -3079.66499
## housing_median_age   1201.33681   1375.41408
## total_bedrooms         93.91834    103.66514
## population            -32.45687    -28.87474
## median_income       39275.71281  40345.17169
## ocean_proximity    -41960.06486 -39113.51581
```

Question 2

```
qn2 = read.csv("binary.csv")
```

Normalising data using min-max normalisation

```
qn2$gre = (qn2$gre - min(qn2$gre))/(max(qn2$gre) - min(qn2$gre))
qn2$gpa = (qn2$gpa - min(qn2$gpa))/(max(qn2$gpa) - min(qn2$gpa))
qn2$rank = (qn2$rank - min(qn2$rank))/(max(qn2$rank) - min(qn2$rank))
```

Train/test set

```
n_row = nrow(qn2)
total_row = 0.75 * n_row
train_sample <- 1: total_row

train_set = qn2[train_sample, ]
test_set = qn2[-train_sample, ]

dim(train_set)
```

```
## [1] 300   4
```

```
dim(test_set)
```

```
## [1] 100   4
```

---

building the model.

```
r formula1
<- admit~gre
+ gpa + rank
l1 <-
glm(formula1,
data =
train_set,
family =
'binomial')
```

Reporting the statistics.

confusion matrix

```
predict <- predict(l1, test_set, type = 'response')

# confusion matrix
matrix <- table(test_set$admit, predict > 0.5)
print(matrix)
```

```
##
##     FALSE TRUE
##   0    59    6
##   1    26    9
```

Accuracytest

(tp + tn)/total.

```
accuracy_Test <- sum(diag(matrix)) / sum(matrix)
accuracy_Test
```

```
## [1] 0.68
```

Reporting statistics

```
summary(l1)
```

```
##
## Call:
## glm(formula = formula1, family = "binomial", data = train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6983  -0.8580  -0.5868   1.0876   2.3188
```

```
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.0729     0.6094  -3.401 0.000671 ***
## gre           1.7870     0.7447   2.400 0.016415 *
## gpa           1.4838     0.6885   2.155 0.031164 *
## rank         -1.9269     0.4496  -4.286 1.82e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 369.84  on 299  degrees of freedom
## Residual deviance: 329.33  on 296  degrees of freedom
## AIC: 337.33
##
## Number of Fisher Scoring iterations: 4
```

We can see that rank and GPA are most significant variables for prediction.

Explain in terms odds by exponentiating the coefficients? TODO ## Confidence Intervals

```
confint(l1,level = 0.95)
```

```
##                  2.5 %     97.5 %
## (Intercept) -3.2999690 -0.9030962
## gre          0.3466938  3.2755420
## gpa          0.1505509  2.8577594
## rank        -2.8349821 -1.0669166
```

adding new column

```
df2 = qn2
df2$new = df2$gpa * df2$rank
```

Train/test set

```
#set.seed(123)

n_row = nrow(df2)
total_row = 0.75 * n_row
train_sample <- 1: total_row

data_train2 = df2[train_sample, ]
data_test2 = df2[-train_sample, ]

dim(data_train2)
```

```
## [1] 300   5
```

```
dim(data_test2)
```

```
## [1] 100   5
```

build the model

```
formula2 <- admit~gpa+gre+rank+new
l2 <- glm(formula2, data = data_train2, family = 'binomial')
```

confusion matrix

```
predict <- predict(l2, data_test2, type = 'response')

# confusion matrix
matrix <- table(data_test2$admit, predict > 0.5)
print(matrix)
```

```
##
##     FALSE TRUE
##   0    57    8
##   1    26    9
```

Accuracytest

```
accuracy_Test2 <- sum(diag(matrix)) / sum(matrix)
accuracy_Test2
```

```
## [1] 0.66
```

Reporting statistics

```
summary(l2)
```

```
##
## Call:
## glm(formula = formula2, family = "binomial", data = data_train2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9089  -0.8206  -0.5914   1.0312   2.2327
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.0959     0.9360  -3.308 0.000941 ***
```

```
## gpa           3.0059     1.2496   2.406 0.016149 *
## gre           1.7873     0.7496   2.384 0.017108 *
## rank          0.3212     1.5232   0.211 0.833002
## new          -3.3396     2.2025  -1.516 0.129440
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 369.84  on 299  degrees of freedom
## Residual deviance: 327.04  on 295  degrees of freedom
## AIC: 337.04
##
## Number of Fisher Scoring iterations: 4
```

The coefficient corresponding to 'new'(gpa*rank) variable is very high compared to all the other variable coefficients.Thus,This variable has more affect on the prediction.

Confidence Intervals

```
confint(l2,level =.99)
```

```
##                   0.5 %     99.5 %
## (Intercept) -5.6397805 -0.7824478
## gpa         -0.1030651  6.3787410
## gre         -0.1128429  3.7706252
## rank        -3.7444490  4.1662339
## new         -9.0381881  2.3771373
```