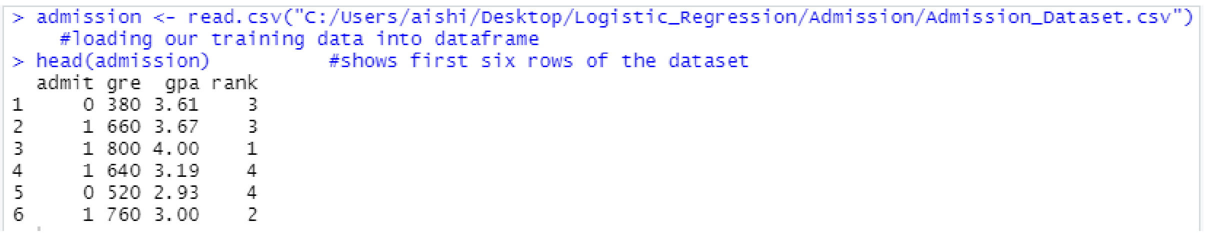
<u>LOGISTIC REGRESSION</u>

1. SELECTION OF DATASET:

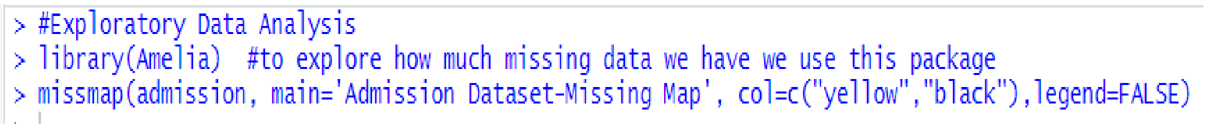
The dataset chosen here is ‘Admission Dataset’. Our goal is to how variables, such as GRE (Graduate Record Exam scores), GPA (grade point average) and prestige of the undergraduate institution, effect admission into graduate school. The response variable, admit/don’t admit, is a binary variable.

Dimension: 400 rows and 4 columns

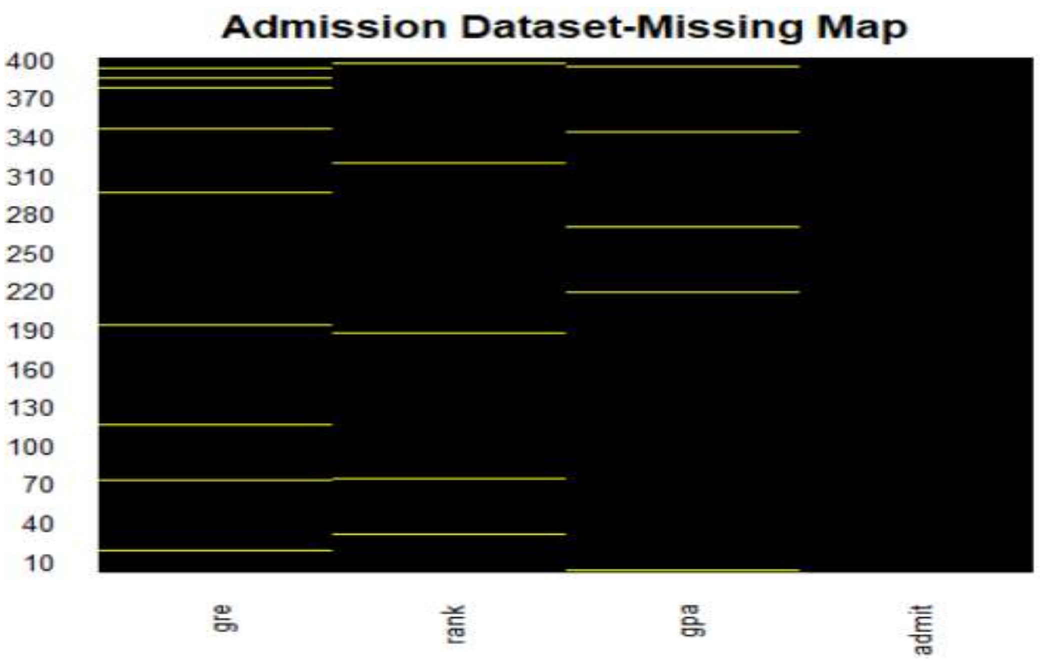


```
> admission <- read.csv("C:/Users/aishi/Desktop/Logistic_Regression/Admission/Admission_Dataset.csv")
   #loading our training data into dataframe
> head(admission)           #shows first six rows of the dataset
  admit gre  gpa rank
1     0 380 3.61    3
2     1 660 3.67    3
3     1 800 4.00    1
4     1 640 3.19    4
5     0 520 2.93    4
6     1 760 3.00    2
```

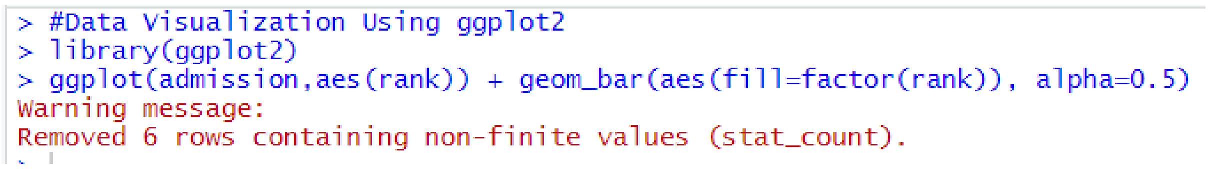
2. EXPLORATORY DATA ANALYSIS:



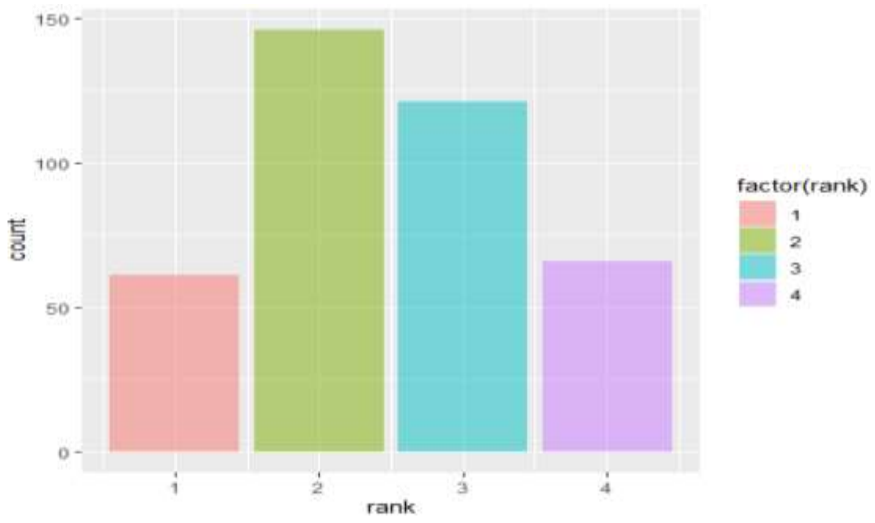
```
> #Exploratory Data Analysis
> library(Amelia)  #to explore how much missing data we have we use this package
> missmap(admission, main='Admission Dataset-Missing Map', col=c("yellow","black"),legend=FALSE)
```



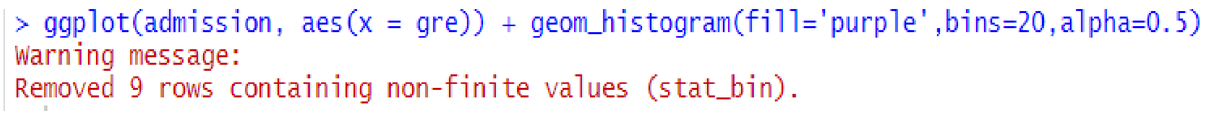
Here yellow represents the missing data. We can see that roughly some percent of the ‘gre’, ‘gpa’ and ‘rank’ data are missing. Now we have to replace the missing data ‘gre’, ‘gpa’ and ‘rank’ with some imputations.



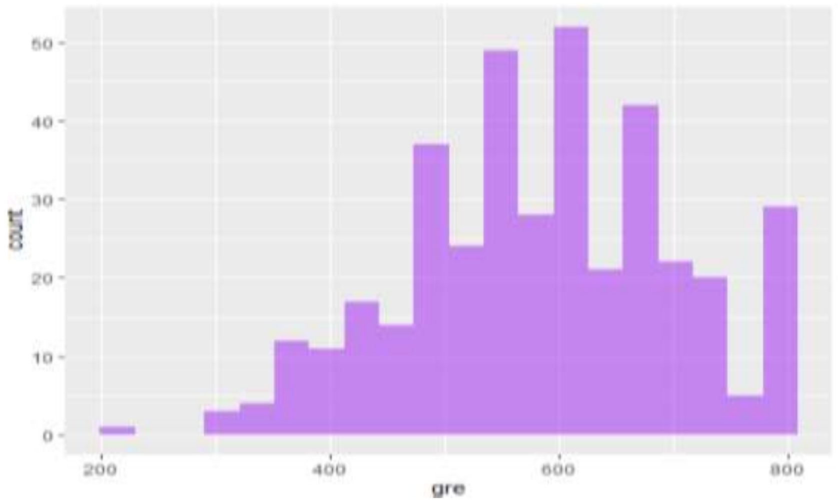
```
> #Data Visualization Using ggplot2
> library(ggplot2)
> ggplot(admission,aes(rank)) + geom_bar(aes(fill=factor(rank)), alpha=0.5)
Warning message:
Removed 6 rows containing non-finite values (stat_count).
```



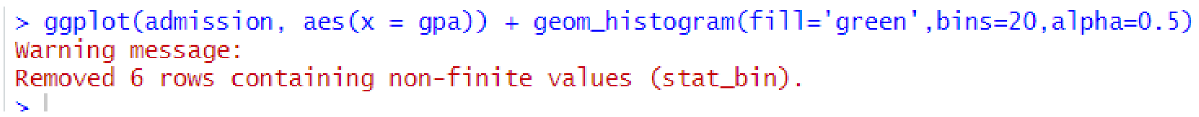
The students with good gre and gpa fall under rank 1. From this graph we can see that the rank 1 amount is very less so we can say that there are not many students who have received good gre and gpa. Whereas the students with average gre and gpa fall under rank 2. And in the graph we can see that rank 2 is highest so we can say that the students mostly got average gre and gpa.



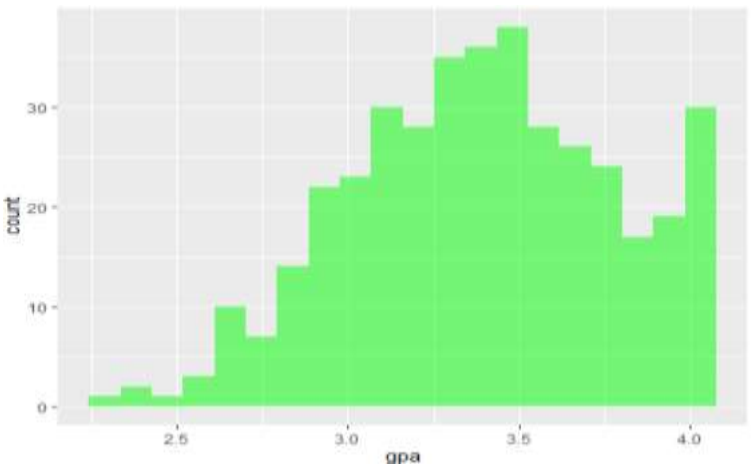
```
> ggplot(admission, aes(x = gre)) + geom_histogram(fill='purple',bins=20,alpha=0.5)
Warning message:
Removed 9 rows containing non-finite values (stat_bin).
```



We can see from this histogram that in many areas gre is more. So we can say that the graph is a scattered one.



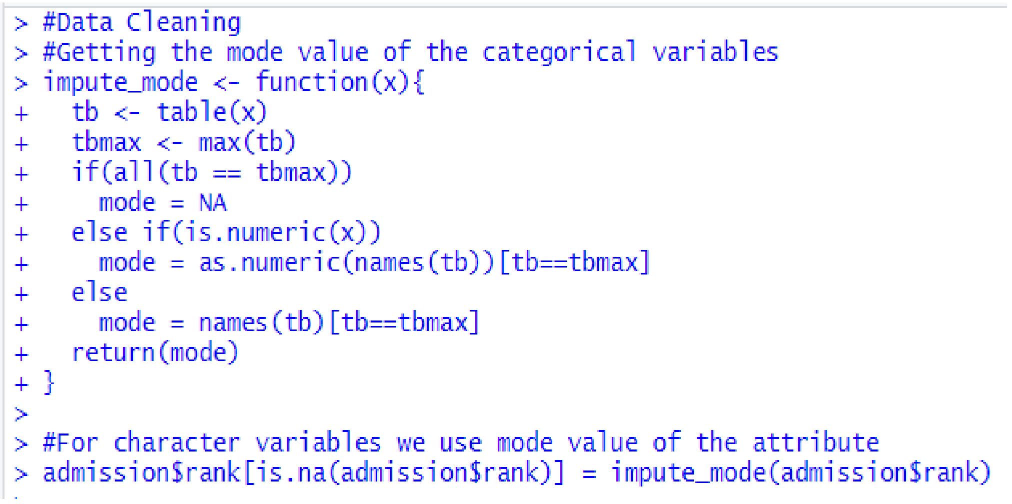
```
> ggplot(admission, aes(x = gpa)) + geom_histogram(fill='green',bins=20,alpha=0.5)
Warning message:
Removed 6 rows containing non-finite values (stat_bin).
>
```



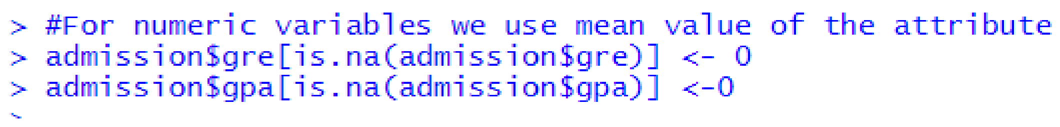
We can see from the graph that the ‘gpa’ graph is also a scattered one and the observations are higher in many areas.

3. DATA CLEANING:

We create a function to treat NA values in categorical attributes i.e. ‘rank’. We treat NA values of numerical attributes with zero because here for attributes ‘gre’ and ‘gpa’ we can’t take the mean value.

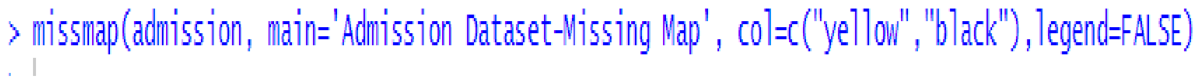


```
> #Data Cleaning
> #Getting the mode value of the categorical variables
> impute_mode <- function(x){
+   tb <- table(x)
+   tbmax <- max(tb)
+   if(all(tb == tbmax))
+     mode = NA
+   else if(is.numeric(x))
+     mode = as.numeric(names(tb))[tb==tbmax]
+   else
+     mode = names(tb)[tb==tbmax]
+   return(mode)
+ }
>
> #For character variables we use mode value of the attribute
> admission$rank[is.na(admission$rank)] = impute_mode(admission$rank)
```

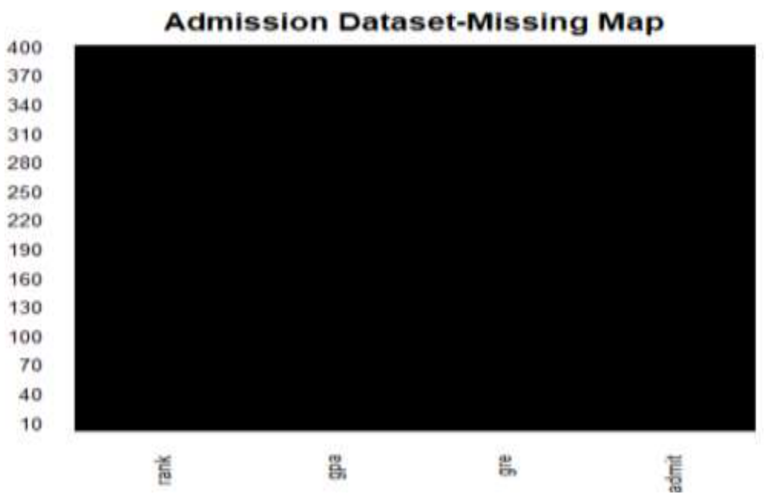


```
> #For numeric variables we use mean value of the attribute
> admission$gre[is.na(admission$gre)] <- 0
> admission$gpa[is.na(admission$gpa)] <-0
```

Now we will see whether the missing values got replaced or not.



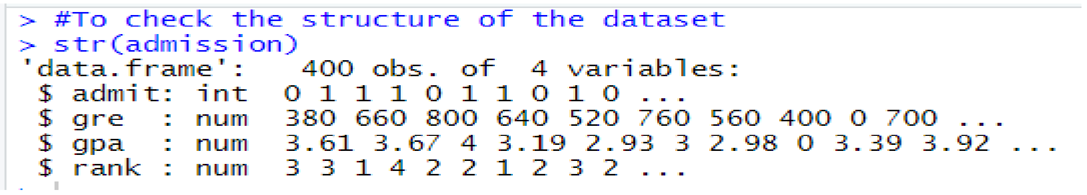
```
> missmap(admission, main='Admission Dataset-Missing Map', col=c("yellow","black"),legend=FALSE)
```



We can see that all the missing values got replaced either by mode or mean value.

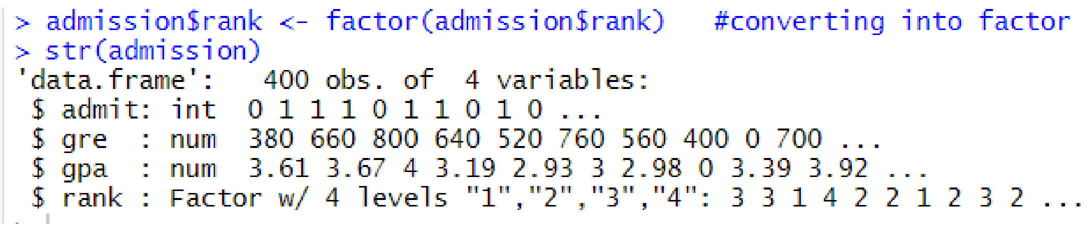
4. BUILDING A LOGISTIC REGRESSION MODEL:

We have to build the model. We will make the features which we would be using of the correct data type.



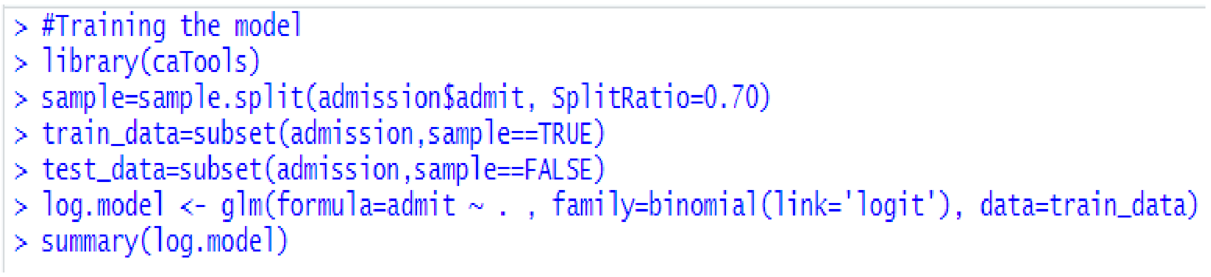
```
> #To check the structure of the dataset
> str(admission)
'data.frame':   400 obs. of  4 variables:
 $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
 $ gre  : num  380 660 800 640 520 760 560 400 0 700 ...
 $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 0 3.39 3.92 ...
 $ rank : num  3 3 1 4 2 2 1 2 3 2 ...
```

We will set factor columns and check the structure again.

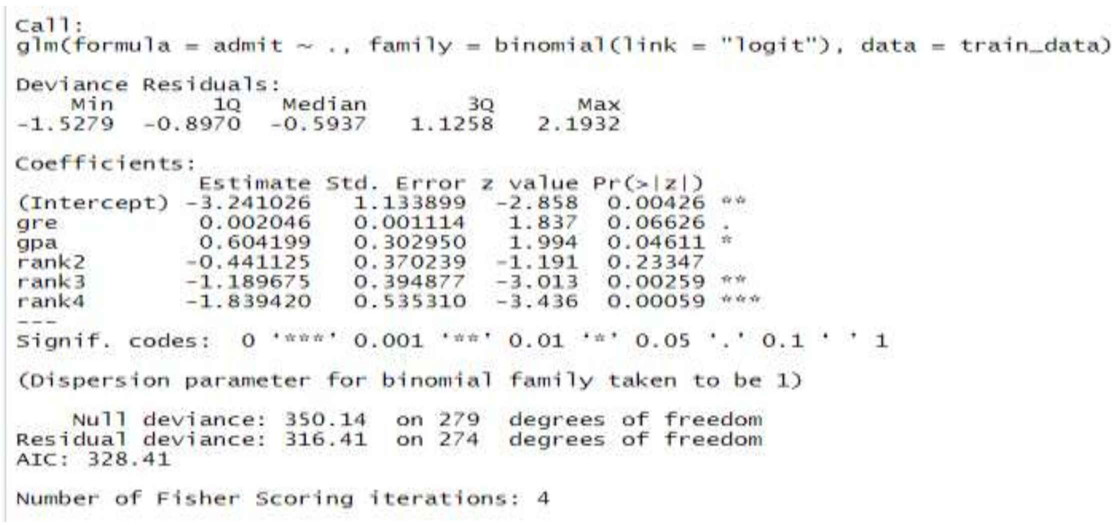


```
> admission$rank <- factor(admission$rank)   #converting into factor
> str(admission)
'data.frame':   400 obs. of  4 variables:
 $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
 $ gre  : num  380 660 800 640 520 760 560 400 0 700 ...
 $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 0 3.39 3.92 ...
 $ rank : Factor w/ 4 levels "1","2","3","4": 3 3 1 4 2 2 1 2 3 2 ...
```

Now we will train our model.



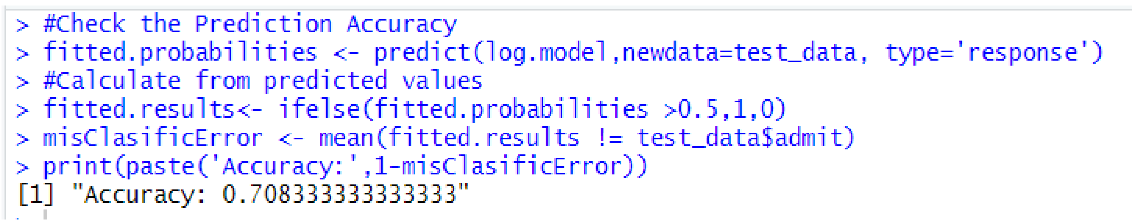
```
> #Training the model
> library(caTools)
> sample=sample.split(admission$admit, SplitRatio=0.70)
> train_data=subset(admission,sample==TRUE)
> test_data=subset(admission,sample==FALSE)
> log.model <- glm(formula=admit ~ . , family=binomial(link='logit'), data=train_data)
> summary(log.model)
```



```
Call:
glm(formula = admit ~ ., family = binomial(link = "logit"), data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5279  -0.8970  -0.5937   1.1258   2.1932

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.241026   1.133899  -2.858  0.00426 **
gre          0.002046   0.001114   1.837  0.06626 .
gpa          0.604199   0.302950   1.994  0.04611 *
rank2       -0.441125   0.370239  -1.191  0.23347
rank3       -1.189675   0.394877  -3.013  0.00259 **
rank4       -1.839420   0.535310  -3.436  0.00059 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 350.14  on 279  degrees of freedom
Residual deviance: 316.41  on 274  degrees of freedom
AIC: 328.41

Number of Fisher Scoring iterations: 4
```

We can clearly see that rank4 have the most significant features. rank3 and gpa have less significant feature. The other feature gre and rank2 have no significant features in the prediction.

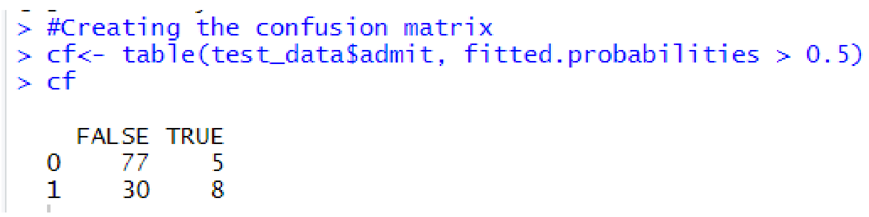
Now we check the prediction of our model.



```
> #Check the Prediction Accuracy
> fitted.probabilities <- predict(log.model,newdata=test_data, type='response')
> #Calculate from predicted values
> fitted.results<- ifelse(fitted.probabilities >0.5,1,0)
> misClasificError <- mean(fitted.results != test_data$admit)
> print(paste('Accuracy:',1-misClasificError))
[1] "Accuracy: 0.708333333333333"
```

We can say that the model has achieved 70.83% accuracy.

Now we evaluate predictions on the training set using confusion matrix.



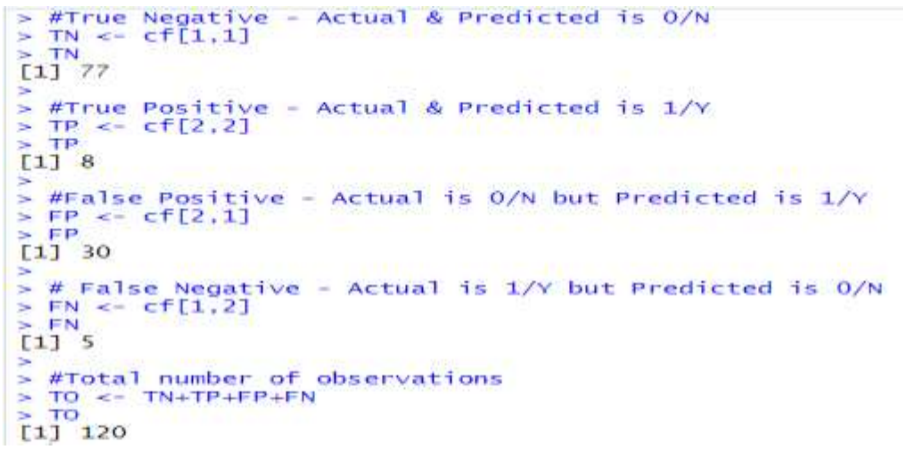
```
> #Creating the confusion matrix
> cf<- table(test_data$admit, fitted.probabilities > 0.5)
> cf

    FALSE TRUE
  0    77    5
  1    30    8
```

From the confusion matrix we can infer that:

- 77 people were predicted not to take admission in the graduate school and they have actually not got admitted.
- 5 people were predicted not to take admission in the graduate school and they have actually got admitted.
- 30 people were predicted to take admission in the graduate school and they have actually not got admitted.
- 8 people were predicted to take admission in the graduate school and they have actually got admitted.

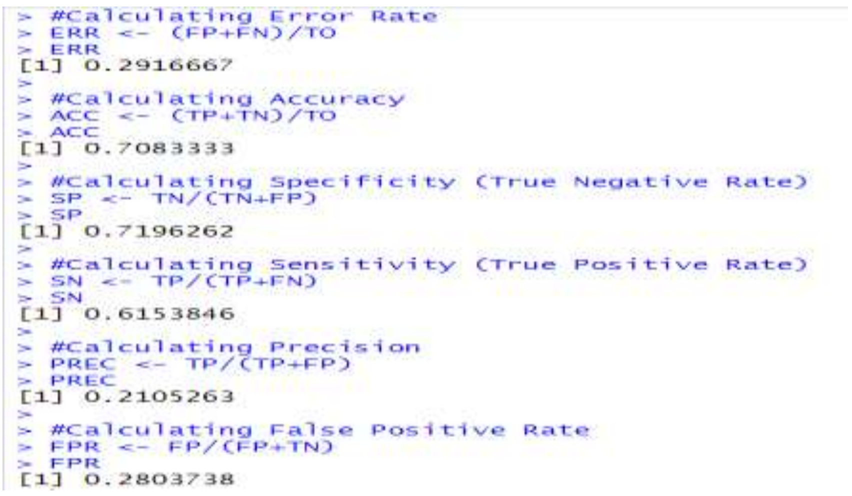
5. ROC CURVE OF THE MODEL:

From the confusion matrix we get some basic terminologies.



```
> #True Negative - Actual & Predicted is 0/N
> TN <- cf[1,1]
> TN
[1] 77
>
> #True Positive - Actual & Predicted is 1/Y
> TP <- cf[2,2]
> TP
[1] 8
>
> #False Positive - Actual is 0/N but Predicted is 1/Y
> FP <- cf[2,1]
> FP
[1] 30
>
> # False Negative - Actual is 1/Y but Predicted is 0/N
> FN <- cf[1,2]
> FN
[1] 5
>
> #Total number of observations
> TO <- TN+TP+FP+FN
> TO
[1] 120
```

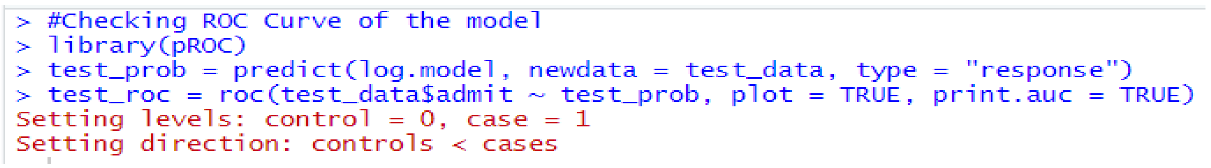
Next we calculate some basic measure derived from confusion matrix.



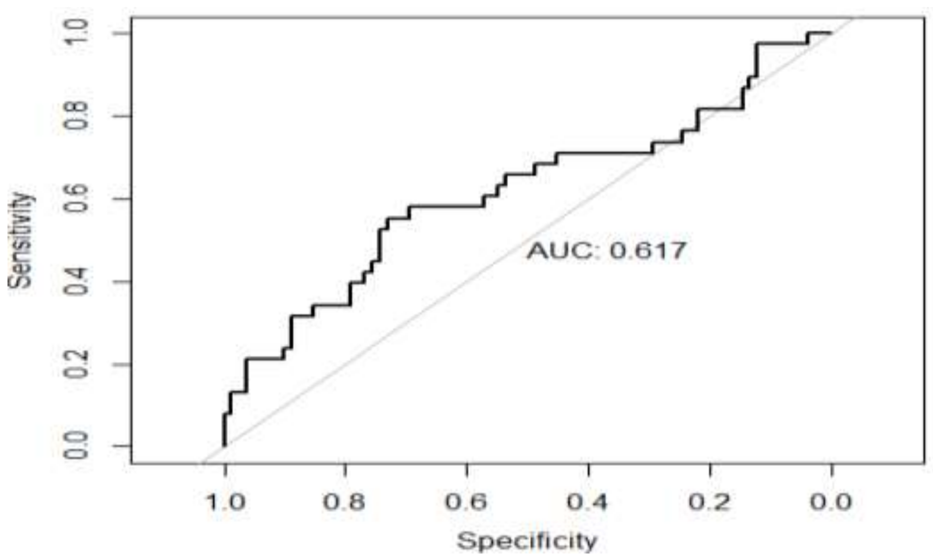
```
> #Calculating Error Rate
> ERR <- (FP+FN)/TO
> ERR
[1] 0.2916667
>
> #Calculating Accuracy
> ACC <- (TP+TN)/TO
> ACC
[1] 0.7083333
>
> #Calculating Specificity (True Negative Rate)
> SP <- TN/(TN+FP)
> SP
[1] 0.7196262
>
> #Calculating Sensitivity (True Positive Rate)
> SN <- TP/(TP+FN)
> SN
[1] 0.6153846
>
> #Calculating Precision
> PREC <- TP/(TP+FP)
> PREC
[1] 0.2105263
>
> #Calculating False Positive Rate
> FPR <- FP/(FP+TN)
> FPR
[1] 0.2803738
```

- Error rate (ERR) that is the number of all incorrect predictions divided by the total number of observations in the dataset is 0.291. It is near 0.0 so we can say that the error rate is good.
- Accuracy (ACC) which is the number of all correct predictions divided by the total number of observations in the dataset is 0.708. It is near 1.0 so we can say that the accuracy is better.
- Specificity (SP) which is the number of correct negative predictions divided by the total number of negatives is 0.719. It is close to 1.0 so we can say that the specificity is better.
- Sensitivity (SN) which is the number of correct positive predictions divided by the total number of positives is 0.615. It is in middle so we can say that the sensitivity is good.
- Precision (PREC) which is the number of correct positive predictions divided by the total number of positive predictions is 0.21. It is far from 1.0 so we can say that the precision is not at all good. PREC is also known as Positive Predictive Value.
- False positive rate (FPR) which is the number of incorrect positive predictions divided by the total number of negatives is 0.28. It is near 0.0 so we can say that the FPR is better.

Now we check the ROC-AUC Curve of the model.



```
> #Checking ROC Curve of the model
> library(pROC)
> test_prob = predict(log.model, newdata = test_data, type = "response")
> test_roc = roc(test_data$admit ~ test_prob, plot = TRUE, print.auc = TRUE)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```



Here the AUC is closely near to 1 that is 0.617 which means it has average measure of separability. This also means there is 61.7% chance that the model will be able to distinguish between positive class and negative class.