LOGISTIC REGRESSION

1. SELECTION OF DATASET:

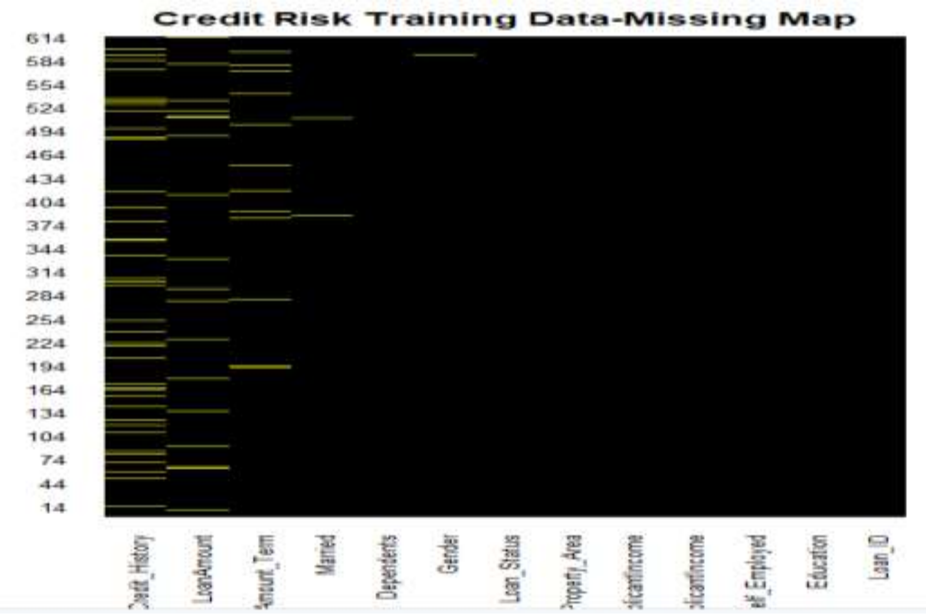
The dataset chosen here is 'Credit Risk Dataset'. Our goal is to properly classify people who have defaulted based on dataset parameters.

Dimension: 614 rows and 13 columns

```
> credit <- read.csv("C:/Users/aishi/Desktop/Logistic_Regression/Credit Risk/Credit_Risk_Train.csv")
  #loading our training data into dataframe
> head(credit)       #shows first six rows of the dataset
   Loan_ID Gender Married Dependents     Education Self_Employed ApplicantIncome
1 LP001002   Male      No          0      Graduate            No            5849
2 LP001003   Male     Yes          1      Graduate            No            4583
3 LP001005   Male     Yes          0      Graduate           Yes            3000
4 LP001006   Male     Yes          0  Not Graduate            No            2583
5 LP001008   Male      No          0      Graduate            No            6000
6 LP001011   Male     Yes          2      Graduate           Yes            5417
  CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History Property_Area
1                 0         NA              360              1         Urban
2              1508        128              360              1         Rural
3                 0         66              360              1         Urban
4              2358        120              360              1         Urban
5                 0        141              360              1         Urban
6              4196        267              360              1         Urban
  Loan_Status
1           Y
2           N
3           Y
4           Y
5           Y
6           Y
```

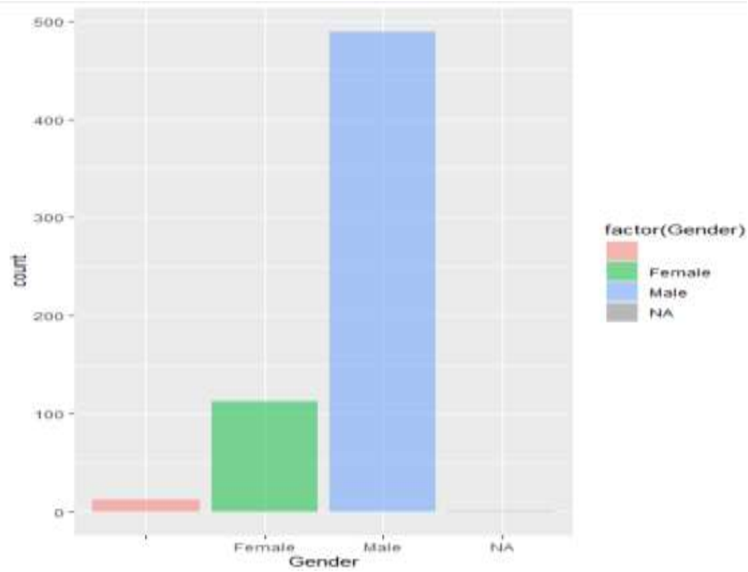
2. EXPLORATORY DATA ANALYSIS:

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



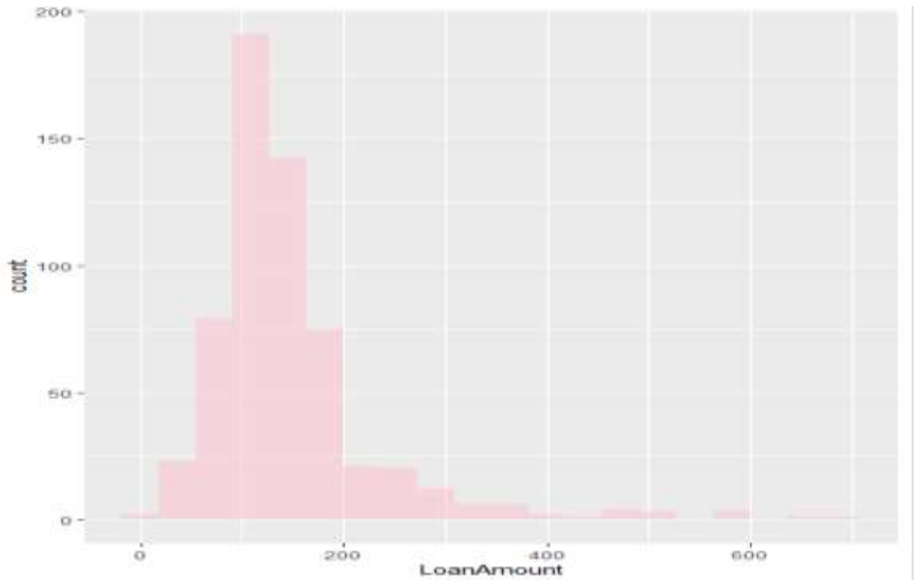
Here yellow represents the missing data. We can see that roughly some percent of the 'Credit_History', 'Loan_Amount' , 'Loan_Amount_Term', 'Married' and 'Gender'. data are missing. Now we have to replace the missing 'Credit_History', 'Loan_Amount', 'Loan_Amount_Term', 'Married' and 'Gender' data with some imputations.

```
> #Data Visualization Using ggplot2
> library(ggplot2)
> ggplot(credit,aes(Gender)) + geom_bar(aes(fill=factor(Gender)), alpha=0.5)
```



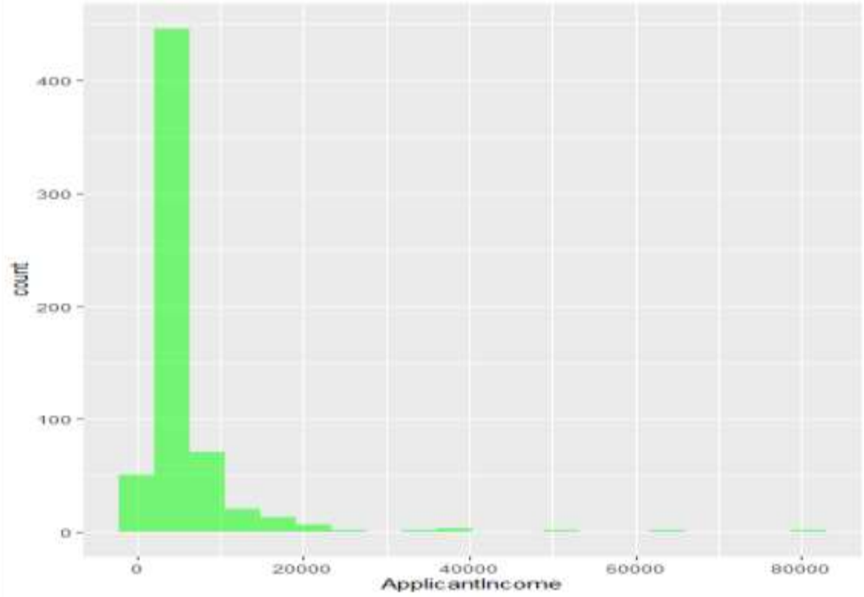
We can see from the graph that there are more number of males who takes loans rather than female. Here the pink bar represents the null values in the 'Gender' column.

```
> ggplot(credit, aes(x = LoanAmount)) + geom_histogram(fill='pink',bins=20,alpha=0.5)
Warning message:
Removed 22 rows containing non-finite values (stat_bin).
```



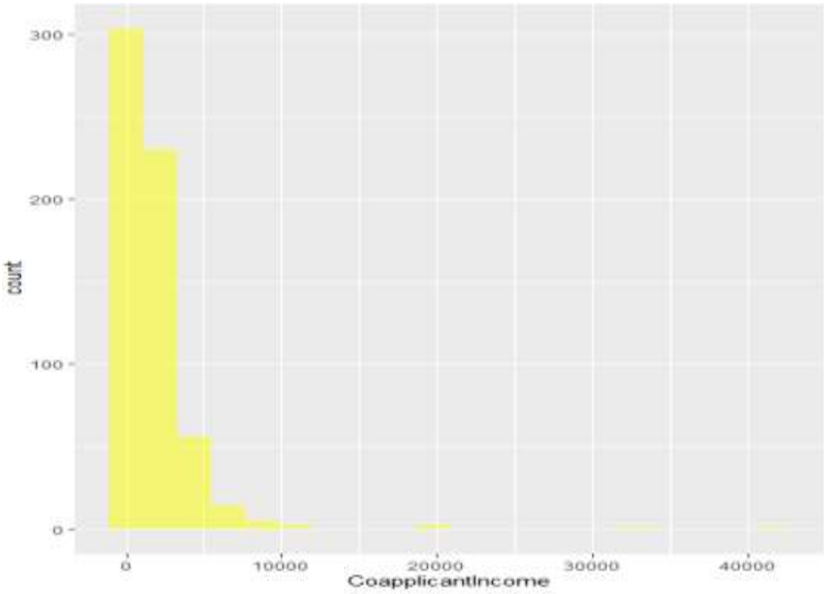
We can see from the graph that the loan amount is high at 100. People may have taken loan of this amount more.

```
> ggplot(credit, aes(x = ApplicantIncome)) + geom_histogram(fill='green',bins=20,alpha=0.5)
```



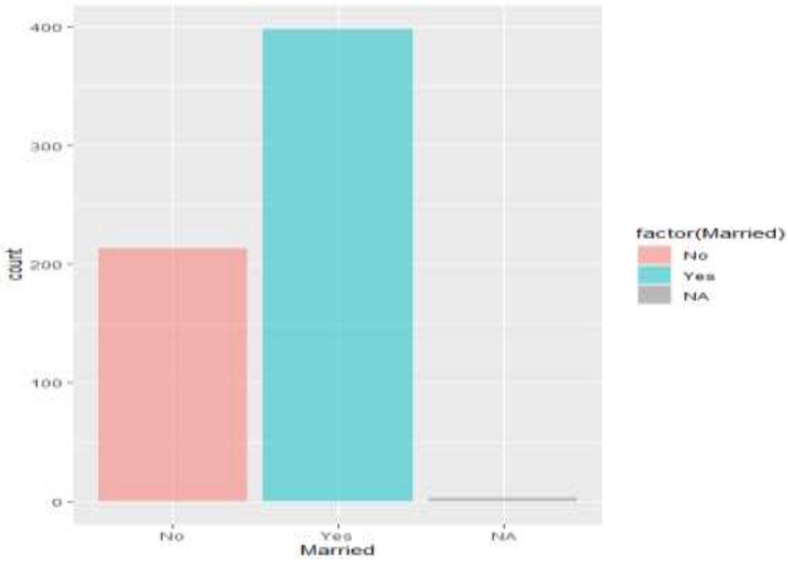
We can see from this graph that the people who applies for loan has income between 5000 and 6000 more. And all the other places the income is very less.

```
> ggplot(credit, aes(x = CoapplicantIncome)) + geom_histogram(fill='yellow',bins=20,alpha=0.5)
```



While applying for a loan in any bank along with the applicant's name we also need to mention a co-applicant. So here in this graph we can see that the co-applicant's income is very less, almost zero. The highest frequency is between 0 and 2000.

```
> ggplot(credit,aes(Married)) + geom_bar(aes(fill=factor(Married)), alpha=0.5)
```



We can see from the graph that the people who takes loan are mostly married. Here grey represents missing value in the 'Married' column.

3. DATA CLEANING:

We create a function to treat NA values in categorical attributes. We treat NA values of numerical attributes with the mean of numerical variable.

```
> #Data Cleaning
> #Getting the mode value of the character 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
> credit$Gender[is.na(credit$Gender)] = impute_mode(credit$Gender)
> credit$Married[is.na(credit$Married)] = impute_mode(credit$Married)
> credit$Credit_History[is.na(credit$Credit_History)] = impute_mode(credit$Credit_History)
```

```
> #For numeric variables we use mean value of the attribute
> credit$LoanAmount[is.na(credit$LoanAmount)] <- mean(credit$LoanAmount, na.rm = TRUE)
> credit$Loan_Amount_Term[is.na(credit$Loan_Amount_Term)] <- mean(credit$Loan_Amount_Term, na.rm = TRUE)
```

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

```
> missmap(credit, main='Credit Risk Training Data-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 handle the categorical attributes as regression model can handle only numeric attributes. So we create dummy variables for categorical attributes which will be used for regression model. If there are only 2 unique values in attribute then we create a dummy variable with 1/0. If there are more than 2 unique values in attribute then we create a dummy variable for each value with 1/0.

```
> cr=credit
> #2 Unique values treatment
> cr$Dummy_Gender=ifelse(credit$Gender=="Male",1,0)
> cr$Dummy_Married=ifelse(credit$Married=="Yes",1,0)
> cr$Dummy_Education=ifelse(credit$Education=="Graduate",1,0)
> cr$Dummy_Self_employed=ifelse(credit$Self_Employed=="Yes",1,0)
>
> #More than 2 unique values treatment
> cr$Dummy_Urban=ifelse(credit$Property_Area=="Urban",1,0)
> cr$Dummy_Rural=ifelse(credit$Property_Area=="Rural",1,0)
> cr$Dummy_Semiurban=ifelse(credit$Property_Area=="Semiurban",1,0)
>
> # Taking first character each of them
> cr$Dummy_Dep=as.numeric(substr(credit$Dependents,1,1))
>
> #Target response variable
> cr$Loan_Status=ifelse(credit$Loan_Status=="Y",1,0)
```

```
> #Checking the transformed dataset
> head(cr)
   Loan_ID Gender Married Dependents    Education Self_Employed ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History
1 LP001002   Male      No          0     Graduate            No            5849                 0   146.4122              360              1
2 LP001003   Male     Yes          1     Graduate            No            4583              1508   128.0000              360              1
3 LP001005   Male     Yes          0     Graduate           Yes            3000                 0    66.0000              360              1
4 LP001006   Male     Yes          0 Not Graduate            No            2583              2358   120.0000              360              1
5 LP001008   Male      No          0     Graduate            No            6000                 0   141.0000              360              1
6 LP001011   Male     Yes          2     Graduate           Yes            5417              4196   267.0000              360              1
  Property_Area Loan_Status Dummy_Gender Dummy_Married Dummy_Education Dummy_Self_employed Dummy_Urban Dummy_Rural Dummy_Semiurban Dummy_Dep
1         Urban           1            1             0               1                   0           1           0               0         0
2         Rural           0            1             1               1                   0           0           1               0         1
3         Urban           1            1             1               1                   1           1           0               0         0
4         Urban           1            1             1               0                   0           1           0               0         0
5         Urban           1            1             0               1                   0           1           0               0         0
6         Urban           1            1             1               1                   1           1           0               0         2
```

We will select only the required columns for training.

```
> library(dplyr)
> cr<- select(cr,-Loan_ID, -Gender,-Married, -Education, -Self_Employed, -Property_Area, -Dependents)
> head(cr,3)
  ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History Loan_Status Dummy_Gender Dummy_Married Dummy_Education
1            5849                 0   146.4122              360              1           1            1             0               1
2            4583              1508   128.0000              360              1           0            1             1               1
3            3000                 0    66.0000              360              1           1            1             1               1
  Dummy_Self_employed Dummy_Urban Dummy_Rural Dummy_Semiurban Dummy_Dep
1                   0           1           0               0         0
2                   0           0           1               0         1
3                   1           1           0               0         0
```

We will check the structure of the columns.

```
> str(cr)
'data.frame':	614 obs. of  14 variables:
 $ ApplicantIncome    : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
 $ CoapplicantIncome  : num  0 1508 0 2358 0 ...
 $ LoanAmount         : num  146 128 66 120 141 ...
 $ Loan_Amount_Term   : num  360 360 360 360 360 360 360 360 360 360 ...
 $ Credit_History     : num  1 1 1 1 1 1 1 0 1 1 ...
 $ Loan_Status        : num  1 0 1 1 1 1 1 0 1 0 ...
 $ Dummy_Gender       : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Dummy_Married      : num  0 1 1 1 0 1 1 1 1 1 ...
 $ Dummy_Education    : num  1 1 1 0 1 1 0 1 1 1 ...
 $ Dummy_Self_employed: num  0 0 1 0 0 1 0 0 0 0 ...
 $ Dummy_Urban        : num  1 0 1 1 1 1 1 0 1 0 ...
 $ Dummy_Rural        : num  0 1 0 0 0 0 0 0 0 0 ...
 $ Dummy_Semiurban    : num  0 0 0 0 0 0 0 1 0 1 ...
 $ Dummy_Dep          : num  0 1 0 0 0 2 0 3 2 1 ...
```

```
> #Training the model
> library(caTools)
> sample<- sample.split(cr$Loan_Status, SplitRatio = 0.70)
> train = subset(cr, sample == TRUE)
> test = subset(cr, sample == FALSE)
> logistic_model <- glm(formula=Loan_Status ~ . , family=binomial(link='logit'), data=train)
> summary(logistic_model)
```

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

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.2391  -0.3316   0.5316   0.7077   2.4903  

Coefficients: (1 not defined because of singularities)
                     Estimate Std. Error z value Pr(>|z|)    
(Intercept)        -1.680e+00  1.022e+00  -1.644   0.1003    
ApplicantIncome     1.474e-05  3.482e-05   0.424   0.6719    
CoapplicantIncome  -4.924e-05  4.522e-05  -1.089   0.2762    
LoanAmount         -3.207e-03  2.037e-03  -1.574   0.1154    
Loan_Amount_Term   -2.893e-03  2.354e-03  -1.229   0.2190    
Credit_History      4.142e+00  5.453e-01   7.596 3.06e-14 ***
Dummy_Gender        2.234e-01  3.348e-01   0.667   0.5047    
Dummy_Married       4.054e-01  2.973e-01   1.363   0.1727    
Dummy_Education     5.172e-01  3.171e-01   1.631   0.1029    
Dummy_Self_employed 5.299e-02  3.644e-01   0.145   0.8844    
Dummy_Urban        -7.000e-01  3.212e-01  -2.179   0.0293 *  
Dummy_Rural        -7.884e-01  3.216e-01  -2.452   0.0142 *  
Dummy_Semiurban            NA         NA      NA       NA    
Dummy_Dep           1.232e-02  1.471e-01   0.084   0.9333    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 532.79  on 428  degrees of freedom
Residual deviance: 384.69  on 416  degrees of freedom
AIC: 410.69

Number of Fisher Scoring iterations: 5
```

We can clearly see that Credit_History have the most significant features. Dummy_Urban and Dummy_Rural have less significant feature. The other features have no significant features in the prediction.

```
> #Check the Prediction Accuracy
> fitted_probabilities <- predict(logistic_model, newdata=test, type='response')
Warning message:
In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
> #Calculate from predicted values
> fitted_results<- ifelse(fitted_probabilities >=0.5,1,0)
> misClasificError <- mean(fitted_results != test$Loan_Status)
> print(paste('Accuracy:',1-misClasificError))
[1] "Accuracy: 0.805405405405405"
```

We can say that the model has achieved 80.54% accuracy.

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

```
> #Creating the confusion matrix
> cf <- table(test$Loan_Status, fitted_probabilities > 0.5)
> cf

    FALSE TRUE
  0    25   33
  1     3  124
```

From the confusion matrix we can infer that:

- 25 people were predicted not to get a loan and they have actually not received the loan.
- 33 people were predicted not to get a loan and they have actually received the loan.
- 3 people were predicted to get a loan and they have not received the loan.
- 124 people were predicted to get a loan and they have actually received the loan.

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] 25
>
> #True Positive - Actual & Predicted is 1/Y
> TP <- cf[2,2]
> TP
[1] 124
>
> #False Positive - Actual is 0/N but Predicted is 1/Y
> FP <- cf[2,1]
> FP
[1] 3
>
> # False Nefgative - Actual is 1/Y but Predicted is 0/N
> FN <- cf[1,2]
> FN
[1] 33
>
> #Total number of observations
> TO <- TN+TP+FP+FN
```

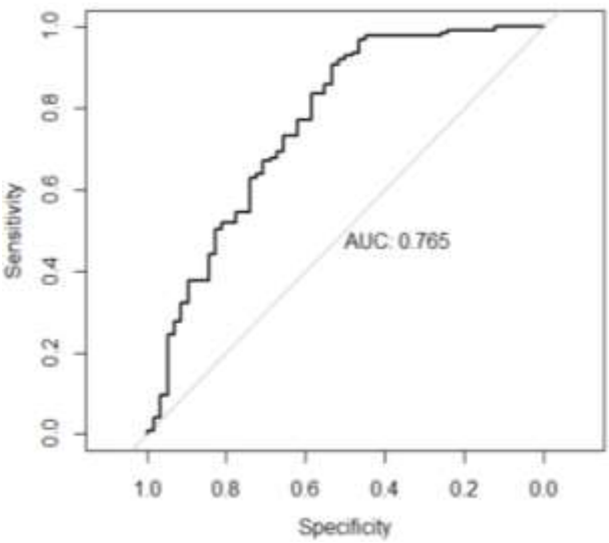
Next we calculate some basic measure derived from confusion matrix.

```
> #Calculating Error Rate
> ERR <- (FP+FN)/TO
> ERR
[1] 0.1945946
>
> #Calculating Accuracy
> ACC <- (TP+TN)/TO
> ACC
[1] 0.8054054
>
> #Calculating Specificity (True Negative Rate)
> SP <- TN/(TN+FP)
> SP
[1] 0.8928571
>
> #Calculating Sensitivity (True Positive Rate)
> SN <- TP/(TP+FN)
> SN
[1] 0.7898089
>
> #Calculating Precision
> PREC <- TP/(TP+FP)
> PREC
[1] 0.976378
>
> #Calculating False Positive Rate
> FPR <- FP/(FP+TN)
> FPR
[1] 0.1071429
```

- Error rate (ERR) that is the number of all incorrect predictions divided by the total number of observations in the dataset is 0.19. 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 0.805. 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 0.89. 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 0.789. It is near 1.0 so we can say that the sensitivity is better.
- Precision (PREC) which is the number of correct positive predictions divided by the total number of positive predictions is 0.97. It is close to 1.0 so we can say that the precision is best. 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 0.10. 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(logistic_model, newdata = test, type = "response")
Warning message:
In predict.lm(object, newdata, se.fit, scale = 1, type = if (type ==  :
  prediction from a rank-deficient fit may be misleading
> test_roc = roc(test$Loan_Status ~ test_prob, plot = TRUE, print.auc = TRUE)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```



Here the AUC is near to 1 that is 0.765 which means it has good measure of separability. This also means there is 76.5% chance that the model will be able to distinguish between positive class and negative class.