**Chance of Admission for Higher Studies**

**Import data from web storage. Name the dataset and now do Logistic**

**Regression to find out relation between variables that are affecting the**

**admission of a student in a institute based on his or her GRE score, GPA**

**obtained and rank of the student. Also check if the model is fit or not.**

**require (foreign), require (MASS).**

Predict the chances of admission of a student to a Graduate program based on:

GRE Scores (290 to 340)

TOEFL Scores (92 to 120) //Named as Rank in dataset

University Rating (1 to 5)

Statement of Purpose (1 to 5)

Letter of Recommendation Strength (1 to 5)

Undergraduate CGPA (6.8 to 9.92) // Named as GPA in dataset

Research Experience (0 or 1)

Chance of Admit (0.34 to 0.97) // Named as Admission in dataset

# Load necessary libraries

require(foreign)

require(MASS)

# Step 1: Import the dataset from web storage

# Replace 'url\_to\_dataset.csv' with the actual URL or file path

url <- "url\_to\_dataset.csv"

data <- read.csv("C:/Users/Girija/ Data Science Using R Programming FDP DATA/Dataset/Admission\_Predict")

# Display the first few rows of the dataset

head(data)

# Step 2: Prepare the data

# Ensure the dataset has the following columns: 'GRE', 'GPA', 'Rank', and 'Admission'

# If 'Admission' is not already a factor (binary), convert it

data$Admission <- as.factor(data$Admission)

# Step 3: Fit the Logistic Regression Model

# Fit the model: Admission ~ GRE + GPA + Rank

model <- glm(Admission ~ GRE + GPA + Rank, data = data, family = binomial)

# Display the summary of the model

summary(model)

# Step 4: Evaluate the Model

# 1. Check the model's coefficients and their significance

coefficients(model)

# 2. Make predictions and evaluate the model fit

# Add predicted probabilities to the data

data$predicted\_prob <- predict(model, type = "response")

# Use the cut-off of 0.5 for classification (adjust if necessary)

data$predicted\_class <- ifelse(data$predicted\_prob > 0.5, 1, 0)

# Check the confusion matrix

table(Predicted = data$predicted\_class, Actual = data$Admission)

# 3. Evaluate performance metrics: Accuracy, Precision, Recall, F1 Score

library(caret)

confusion\_matrix <- confusionMatrix(factor(data$predicted\_class), data$Admission)

print(confusion\_matrix)

# 4. Check the model's goodness of fit

# Deviance Residuals and Pearson Residuals

deviance(model)

pearson.residuals <- residuals(model, type = "pearson")

sum(pearson.residuals^2)

# ROC Curve and AUC

library(pROC)

roc\_curve <- roc(data$Admission, data$predicted\_prob)

plot(roc\_curve)

auc(roc\_curve)

# Load necessary libraries

require(MASS)

require(caret)

require(pROC)

# Step 1: Import the dataset (replace 'dataset.csv' with your actual dataset file)

# For demonstration purposes, we'll create a sample dataset

# In practice, you should load your dataset with actual values

data <- data.frame(

GRE = c(300, 320, 310, 330, 315),

TOEFL = c(95, 105, 100, 110, 102),

University\_Rating = c(3, 4, 3, 5, 4),

SOP = c(3, 4, 3, 5, 4),

LOR = c(2, 4, 3, 5, 4),

CGPA = c(8.5, 9.0, 8.7, 9.5, 9.2),

Research = c(1, 1, 0, 1, 0),

Chance\_of\_Admit = c(0.6, 0.8, 0.5, 0.9, 0.7)

)

# Display the first few rows of the dataset

head(data)

# Step 2: Prepare the data

# Ensure 'Chance\_of\_Admit' is treated as a numeric variable

data$Chance\_of\_Admit <- as.numeric(data$Chance\_of\_Admit)

# Fit the logistic regression model

# Using 'Chance\_of\_Admit' as the response variable

model <- glm(Chance\_of\_Admit ~ GRE + TOEFL + University\_Rating + SOP + LOR + CGPA + Research,

data = data, family = binomial)

# Display the summary of the model

summary(model)

# Step 3: Predicting the chance of admission for new data

# Create a new data frame with student data to predict

new\_data <- data.frame(

GRE = 310,

TOEFL = 100,

University\_Rating = 3,

SOP = 3,

LOR = 3,

CGPA = 8.9,

Research = 1

)

# Predict the chance of admission

predicted\_prob <- predict(model, newdata = new\_data, type = "response")

predicted\_prob

# If you want to classify the chance (e.g., if it's above a certain threshold), you can set a cutoff

# Here, we'll use 0.5 as the cutoff

predicted\_class <- ifelse(predicted\_prob > 0.5, 1, 0)

predicted\_class

# Display the prediction

cat("Predicted probability of admission:", predicted\_prob, "\n")

cat("Predicted class (1 if admitted, 0 if not):", predicted\_class, "\n")

OUTPUT:

> require(foreign)

> require(MASS)

> data <- read.csv("C:/Users/MGIT/Downloads/Data Science Using R Programming FDP DATA/Dataset/Admission\_Predict.csv")

> head(data)

Serial.No. GRE Rank University.Rating SOP LOR GPA

1 1 337 118 4 4.5 4.5 9.65

2 2 324 107 4 4.0 4.5 8.87

3 3 316 104 3 3.0 3.5 8.00

4 4 322 110 3 3.5 2.5 8.67

5 5 314 103 2 2.0 3.0 8.21

6 6 330 115 5 4.5 3.0 9.34

Research Admission

1 1 0.92

2 1 0.76

3 1 0.72

4 1 0.80

5 0 0.65

6 1 0.90

> data$Admission <- as.factor(data$Admission)

> model <- glm(Admission ~ GRE + GPA + Rank, data = data, family = binomial)

> summary(model)

Call:

glm(formula = Admission ~ GRE + GPA + Rank, family = binomial,

data = data)

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -75.3890 51.3780 -1.467 0.142

GRE 0.1885 0.2033 0.927 0.354

GPA -0.6445 2.8172 -0.229 0.819

Rank 0.2830 0.2658 1.065 0.287

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 25.183 on 399 degrees of freedom

Residual deviance: 16.752 on 396 degrees of freedom

AIC: 24.752

Number of Fisher Scoring iterations: 11

> coefficients(model)

(Intercept) GRE GPA Rank

-75.3889938 0.1884639 -0.6445444 0.2830341

> data$predicted\_prob <- predict(model, type = "response")

> data$predicted\_class <- ifelse(data$predicted\_prob > 0.5, 1, 0)

> table(Predicted = data$predicted\_class, Actual = data$Admission)

Actual

Predicted 0.34 0.36 0.38 0.39 0.42 0.43 0.44 0.45

1 2 2 2 1 3 1 3 2

Actual

Predicted 0.46 0.47 0.48 0.49 0.5 0.51 0.52 0.53

1 5 5 3 4 2 1 5 3

Actual

Predicted 0.54 0.55 0.56 0.57 0.58 0.59 0.6 0.61

1 5 1 6 8 5 4 1 7

Actual

Predicted 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69

1 9 6 17 9 7 7 10 7

Actual

Predicted 0.7 0.71 0.72 0.73 0.74 0.75 0.76 0.77

1 12 16 15 13 11 8 12 8

Actual

Predicted 0.78 0.79 0.8 0.81 0.82 0.83 0.84 0.85

1 12 12 11 8 8 3 9 6

Actual

Predicted 0.86 0.87 0.88 0.89 0.9 0.91 0.92 0.93

1 8 5 4 9 8 7 6 9

Actual

Predicted 0.94 0.95 0.96 0.97

1 12 4 7 4

> library(caret)

> confusion\_matrix <- confusionMatrix(factor(data$predicted\_class), data$Admission)

Error in confusionMatrix.default(factor(data$predicted\_class), data$Admission) :

The data must contain some levels that overlap the reference.

> install.packages("caret")

The following package(s) will be installed:

- caret [6.0-94]

These packages will be installed into "C:/Users/MGIT/Desktop/3201/StudentData/renv/library/windows/R-4.4/x86\_64-w64-mingw32".

Do you want to proceed? [Y/n]: Y

# Installing packages -------------------------------

- Installing caret ... OK [linked from cache]

Successfully installed 1 package in 16 milliseconds.

> library(caret)

> confusion\_matrix <- confusionMatrix(factor(data$predicted\_class), data$Admission)

Error in confusionMatrix.default(factor(data$predicted\_class), data$Admission) :

The data must contain some levels that overlap the reference.

> print(confusion\_matrix)

Error: object 'confusion\_matrix' not found

> deviance(model)

[1] 16.75226

> pearson.residuals <- residuals(model, type = "pearson")

> sum(pearson.residuals^2)

[1] 49.57867

> install.packages("pROC")

The following package(s) will be installed:

- pROC [1.18.5]

These packages will be installed into "C:/Users/MGIT/Desktop/3201/StudentData/renv/library/windows/R-4.4/x86\_64-w64-mingw32".

Do you want to proceed? [Y/n]: library(pROC)

- Unrecognized response: please enter 'y' or 'n', or type Ctrl + C to cancel.

Do you want to proceed? [Y/n]: Y

# Installing packages -------------------------------

- Installing pROC ... OK [linked from cache]

Successfully installed 1 package in 15 milliseconds.

> roc\_curve <- roc(data$Admission, data$predicted\_prob)

Setting levels: control = 0.34, case = 0.36

Setting direction: controls < cases

Warning message:

In roc.default(data$Admission, data$predicted\_prob) :

'response' has more than two levels. Consider setting 'levels' explicitly or using 'multiclass.roc' instead

> plot(roc\_curve)

> auc(roc\_curve)

Area under the curve: 1

> require(MASS)

> require(caret)

> require(pROC)

> data <- data.frame(

+ GRE = c(300, 320, 310, 330, 315),

+ Rank = c(95, 105, 100, 110, 102),

+ University\_Rating = c(3, 4, 3, 5, 4),

+ SOP = c(3, 4, 3, 5, 4),

+ LOR = c(2, 4, 3, 5, 4),

+ GPA = c(8.5, 9.0, 8.7, 9.5, 9.2),

+ Research = c(1, 1, 0, 1, 0),

+ Admission = c(0.6, 0.8, 0.5, 0.9, 0.7)

+ )

> head(data)

GRE Rank University\_Rating SOP LOR GPA Research

1 300 95 3 3 2 8.5 1

2 320 105 4 4 4 9.0 1

3 310 100 3 3 3 8.7 0

4 330 110 5 5 5 9.5 1

5 315 102 4 4 4 9.2 0

Admission

1 0.6

2 0.8

3 0.5

4 0.9

5 0.7

> data$Admission <- as.numeric(data$Admission)

> model <- glm(Admission ~ GRE + Rank + University\_Rating + SOP + LOR + GPA + Research,

+ data = data, family = binomial)

Warning message:

In eval(family$initialize) : non-integer #successes in a binomial glm!

> summary(model)

Call:

glm(formula = Admission ~ GRE + Rank + University\_Rating + SOP +

LOR + GPA + Research, family = binomial, data = data)

Coefficients: (3 not defined because of singularities)

Estimate Std. Error z value

(Intercept) 0.1783 790.4685 0.000

GRE 0.1383 8.3040 0.017

Rank -0.2426 15.4666 -0.016

University\_Rating 2.0794 7.5462 0.276

SOP NA NA NA

LOR NA NA NA

GPA -2.8768 31.6667 -0.091

Research NA NA NA

Pr(>|z|)

(Intercept) 1.000

GRE 0.987

Rank 0.987

University\_Rating 0.783

SOP NA

LOR NA

GPA 0.928

Research NA

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 5.0363e-01 on 4 degrees of freedom

Residual deviance: -2.2204e-17 on 0 degrees of freedom

AIC: 13.778

Number of Fisher Scoring iterations: 5

> new\_data <- data.frame(

+ GRE = 310,

+ Rank = 100,

+ University\_Rating = 3,

+ SOP = 3,

+ LOR = 3,

+ GPA = 8.9,

+ Research = 1

+ )

> predicted\_prob <- predict(model, newdata = new\_data, type = "response")

Warning message:

In predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :

prediction from rank-deficient fit; attr(\*, "non-estim") has doubtful cases

> predicted\_prob

1

0.36

attr(,"non-estim")

1

1

> predicted\_class <- ifelse(predicted\_prob > 0.5, 1, 0)

> predicted\_class

1

0

> cat("Predicted probability of admission:", predicted\_prob, "\n")

Predicted probability of admission: 0.36

> cat("Predicted class (1 if admitted, 0 if not):", predicted\_class, "\n")

Predicted class (1 if admitted, 0 if not): 0