REGULARIZATION

ALY6015: Intermediate Analytics

DHRUV VIJAY GUJRATHI

Northeastern University

DHRUV VIJAY GUJRATHI, College of Professional Studies, Northeastern University, Boston, MA 02115.

This report is regarding regularization.

Dhruv Gujrathi is now a student at Department of Analytics, Northeastern University

Contact: gujrathi.d@northeastrn.edu

(NUID: 001029464)

INTRODUCTION

Regularization of data can be defined as a very important concept which is used to avoid overfitting data especially when the trained and tested data are varying. Regularization is implemented by adding a "penalty" term to the best fit derived from the trained data, in order to achieve a lesser variance with the tested data and also restricts the influence of predictor variables over the output variable by compressing their coefficients.

Ridge Regression: Ridge regression is a method which is implemented by summing up bias to a multilinear regression model to get an accurate regression with tested data at a cost of losing accuracy for the training data. The general equation of a best-fit line for multilinear regression is

$$\mathbf{y = \beta 0 + \beta 1x1 + \beta 2x2 + \cdots \beta kxk}$$

where y is the output variable and x1, x2…xk are predictor variables.

The penalty term for ridge regression is $\lambda(\text{slope})^2$, where lambda denotes the degree of deflection from the original curve by restricting the coefficients of predictor variables but never makes them zero. Therefore, the equation for ridge regression is

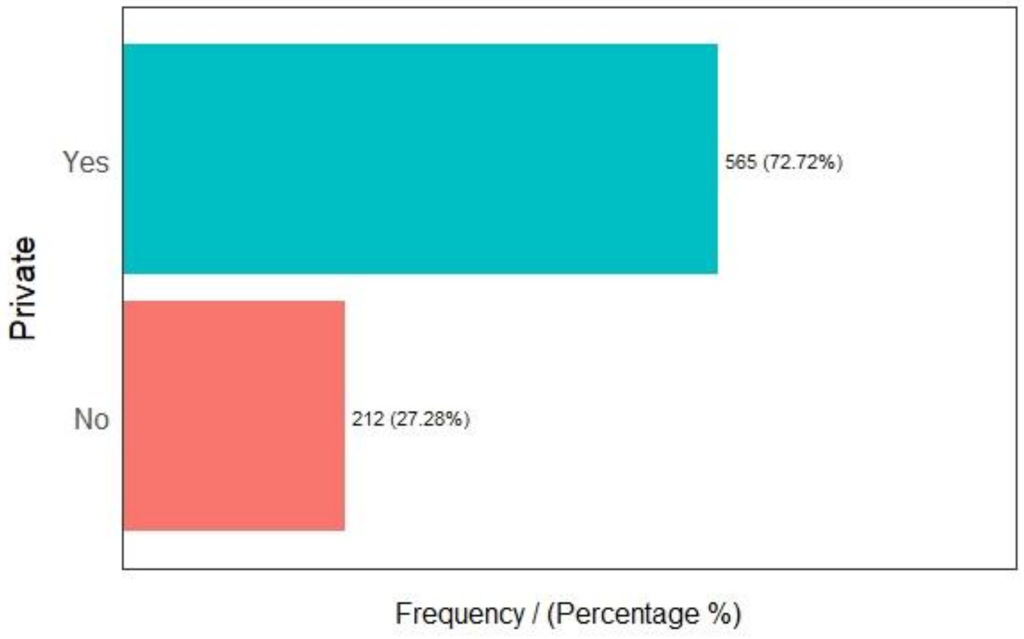
$$\mathbf{y = \beta 0 + \beta 1x1 + \beta 2x2 + \cdots \beta kxk + \lambda(slope)^2}$$

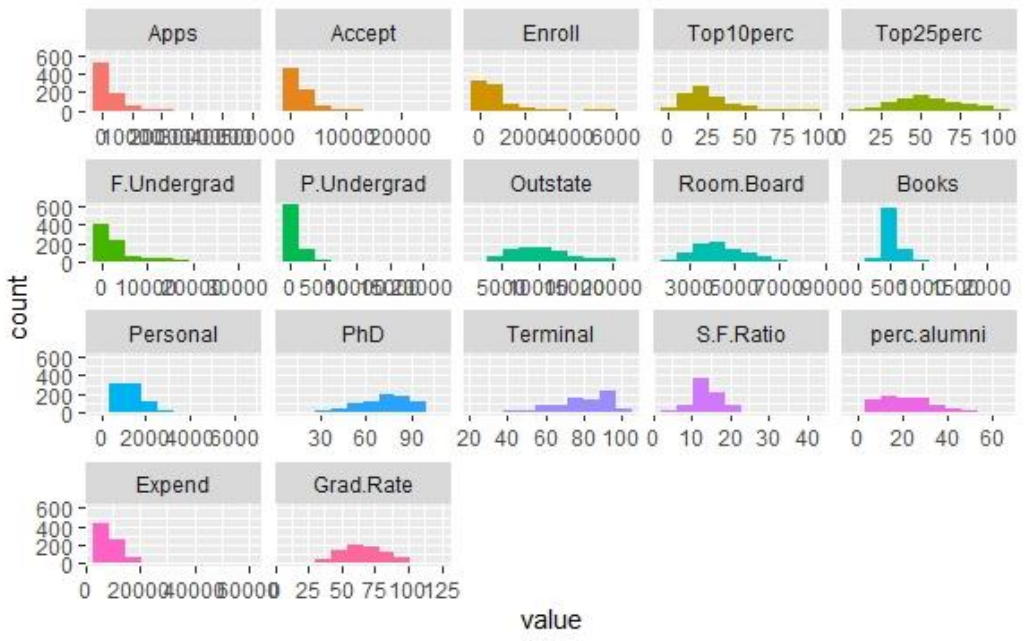
Lasso Regression: Lasso regression is like ridge regression, but it only differs in the penalty term. The penalty for lasso regression is $\lambda|\text{slope}|$. It can eliminate the variables by equating their coefficients to zero therefore removing the variables that have high covariance with other predictor variables. The equation for lasso regression is

$$\mathbf{y = \beta 0 + \beta 1x1 + \beta 2x2 + \cdots \beta kxk + \lambda|slope|}$$

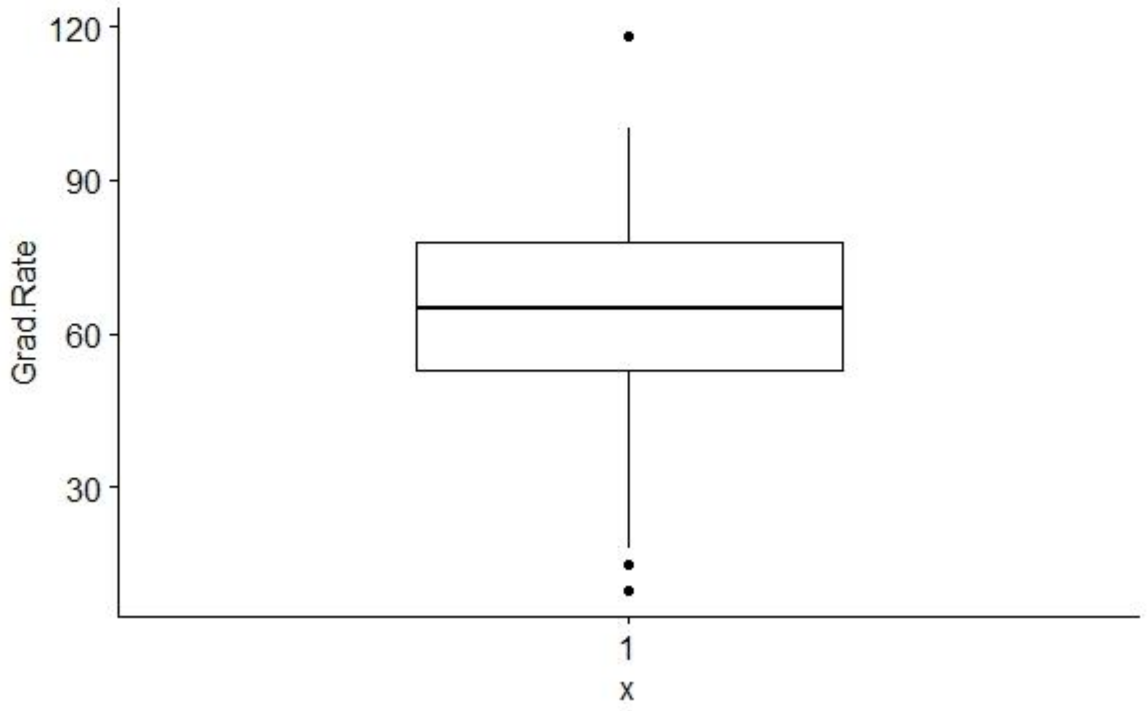
ANALYSIS

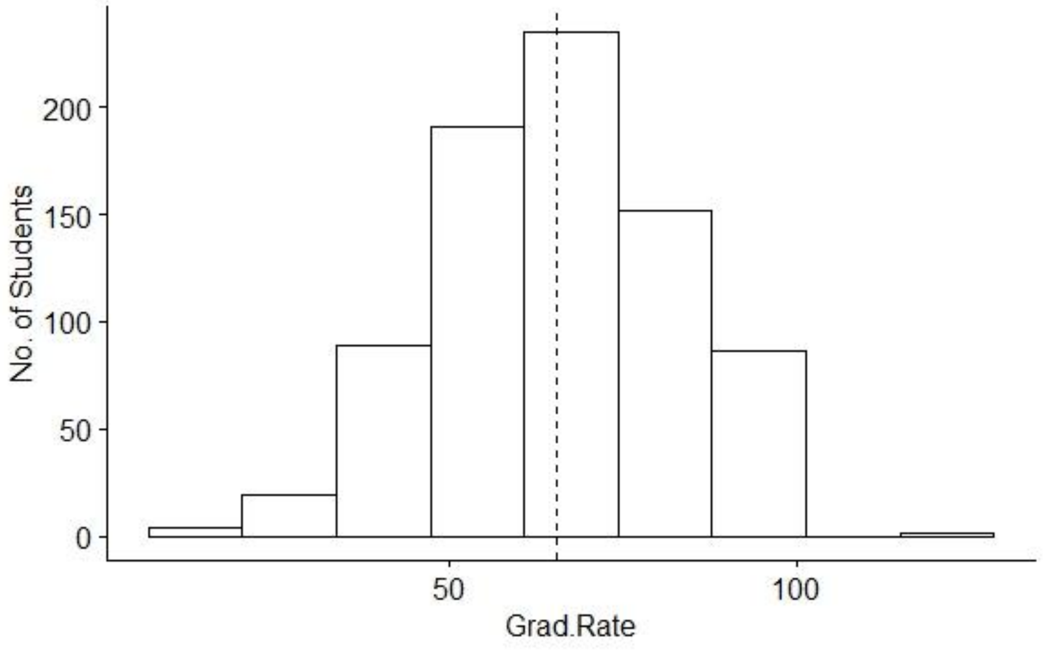
1. We install the "ISLR" package which is a collection of datasets used in the book 'An Introduction to Statistical Learning with Applications in R'. The "College" dataset is a part of the ISLR package and are the Statistics for many US Colleges from the 1995 issue of US News and World Report. EDA stands for 'Exploratory Data Analysis' and is a process used to interpret, analyse, and visualize the dataset to get a better understanding of it. To perform EDA, we basically first analyze/interpret the data and then visualize it to get better insights into it. To analyze the data we use different functions that help us to get to know the data and its variables. After analyzing the data, we visualize the data to find the number of occurrences in each variable which helps us to calculate the values manually.

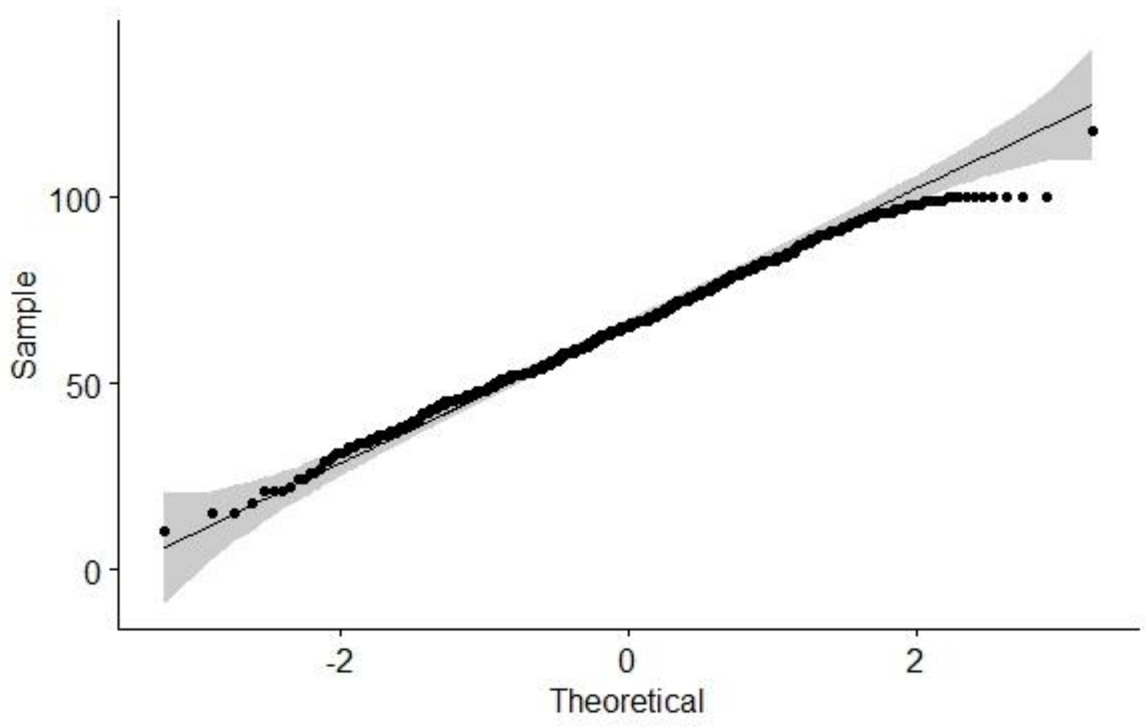


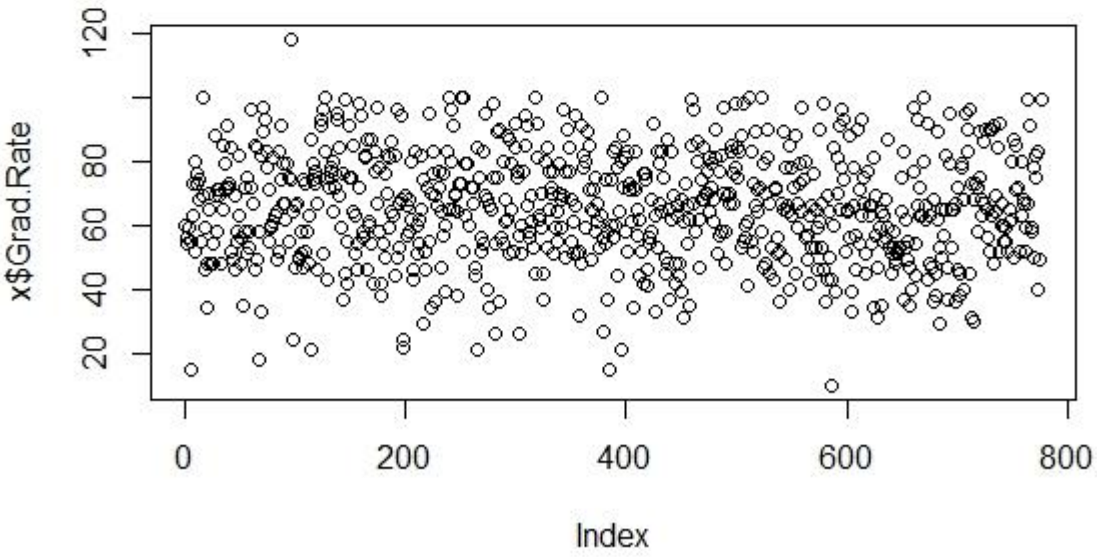


2. Descriptive statistics is used to describe the dataset using certain functions about the data in a study. It provides a simple summary of all the features and samples. They are very useful in doing a quantitative data analysis. Through descriptive statistics, we determine the dataset features like mean, median, mode, quantile, range standard deviation, variance, etc. After finding the essential features of the dataset, we determine the summary and then go on to visualize the statistical parameters that we have determined after completing the descriptive statistics. The package used to conduct a descriptive statistics procedure on the dataset is "Hmisc" and "tidyverse". The mean of the dataset is 65.46, median is 65, range is from 10 to 118, variance is 295.07 and the standard deviation is 17.17.









3. Separating data into training and testing sets is an important part of evaluating data mining models. By using similar data for training and testing, you can minimize the effects of data discrepancies and better understand the characteristics of the model. If one fits the model on the training dataset, then they implicitly minimize error or find correct responses. The fitted model provides a good prediction on the training dataset. Then one can test the model on the test dataset. To split the dataset, we firstly set seed so that the sample can be reproduced in the future also. Then, by using the ‘sample’ function, we can split the dataset in the required percentage values.

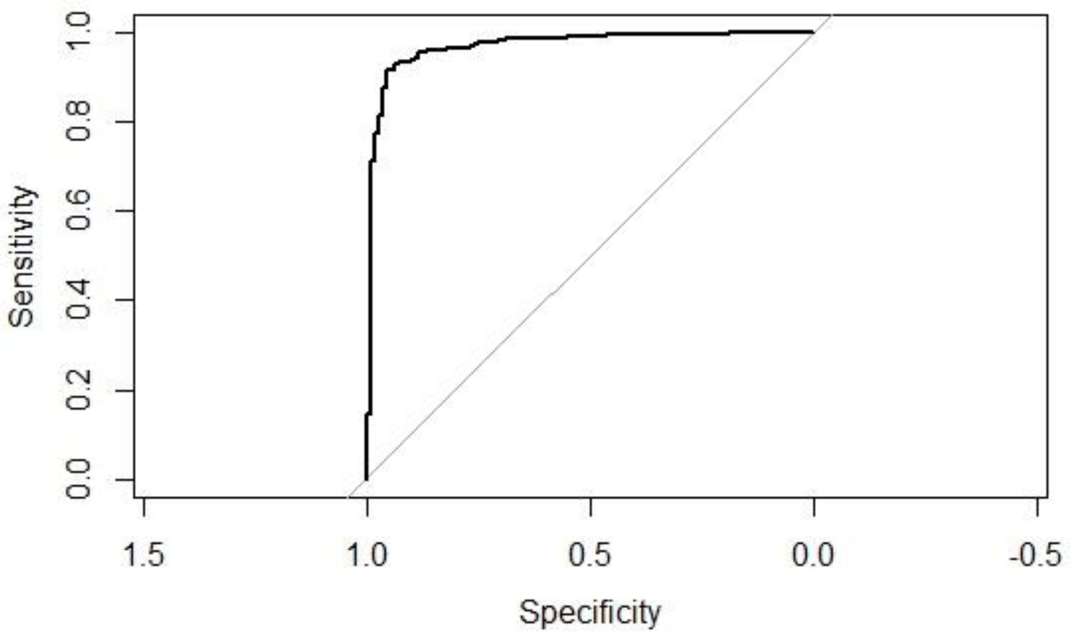
4. We use the ‘stats’ package to call the ‘glm’ function. The stats package is used to carry out the statistical functions on the dataset. This can include determining the linear regression to fit model, working the residuals determining the deviance, the null deviance, etc. The ‘glm’ function is used to fit the generalized linear models, specified by the giving a symbolic description of the linear predictor and a description of the error distribution. The ‘glm’ function returns an object of class inheriting from ‘glm’ which inherits from the class "lm". If a non-standard method is used, the object will also inherit from the class returned by that function.
5. A confusion matrix is a technique for summarizing the performance of a classification algorithm. Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset. Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making. In simple words, it calculates a cross-tabulation of the observed and predicted classes with associated statistics.

6. The confusion matrix is as follows:

```
   No Yes
 0 87   9
 1 27 266
```

From the values in the code, we can clearly see that the precision is 90%, which means that labels of 90% universities are ‘No’ and the remaining are ‘Yes’. The specificity is 90%, the recall value is 76% and the accuracy is 90%.

7. This function returns the ROC curve and computes the area under the curve (AUC) for binary classifiers. his is the main function of the pROC package. It builds a ROC curve and returns a roc object, a list of class roc. This object can be printed, plotted, or passed to the functions auc, ci, smooth.roc and coords. Additionally, two roc objects can be compared with roc.test. In simple words, the main job of this function is to build a ROC object.



   From the plot/graph above, we can understand that closer the curve to the top left corner of the graph axes, better the performance of the ROC curve. We also need to note that ROC does not depend on the class of the distribution and that is why is best used for predicting events like disasters or diseases.
8. The AUC can be defined as the probability that the fit model will score a randomly drawn positive sample higher than a randomly drawn negative sample. This function computes the numeric value of area under the ROC curve (AUC) with the trapezoidal rule. Two syntaxes are possible: one object of class "roc", or either two vectors (response, predictor) or a formula (response~predictor) as in the roc function. By default, the total AUC is computed, but a portion of the ROC curve can be specified with partial.auc. From the graph/plot above we can conclude that the area under the curve is used to summarize the performance of each classifier into a singular measure. It is basically a measure of predictive accuracy. In this case, the AUC is 0.9694.

CONCLUSION

In this project, we used the regularization methods for models to describe the relationship between the given variables. We also developed a prediction model using the confusion matrix and ultimately by plotting the ROC curve which depicts if the class of variables will affect the precision, accuracy or specificity of the prediction model which is developed. We used the glm() function to fit a regression model where the ridge and LASSO functions were implemented over the lambda values of regression.

REFERENCES:

1. Tran, J. (2019, December 20). Exploratory Data Analysis in R for beginners. Retrieved November 11, 2020, from https://towardsdatascience.com/exploratory-data-analysis-in-r-for-beginners-fe031add7072
2. Descriptive Statistics and Graphics. (n.d.). Retrieved November 18, 2020, from http://www.sthda.com/english/wiki/descriptive-statistics-and-graphics

APPENDIX:

CODE:

```
install.packages("ISLR") #Package containing 'College' dataset

install.packages("tidyverse") #Data Manipulation & Visualization

install.packages("Hmisc") #Imputing missing values, high level graphics, etc.

install.packages("funModeling")

install.packages("stats")

library(ISLR)

library(tidyverse)

library(Hmisc)

library(funModeling)

library(stats)

data("College") #Selecting dataset 'College'

# defining this function

getdata <- function(ISLR)

{

  e <- new.env()

  name <- data(College, envir = e)[1]

  e[[name]]

}


#Loading the dataset calling getdata()

x <- getdata("College")

x
```

```
#Performing EDA
basic_eda <- function(x) #Performing exploratory data analysis
{
  glimpse(x)
  print(status(x))
  freq(x)
  print(profiling_num(x))
  plot_num(x)
  describe(x)
}
basic_eda(x)
#Performing Descriptive Statistics
summary(x)
mean(x$Grad.Rate)
median(x$Grad.Rate)
install.packages("modeest")
library(modeest)
mfv((x$Grad.Rate))
min(x$Grad.Rate)
max(x$Grad.Rate)
range(x$Grad.Rate)
var(x$Grad.Rate)
sd(x$Grad.Rate)
#Plotting the descriptive statistical characteristics
```

```
install.packages("ggpubr")

library(ggpubr)

ggboxplot(x, y = "Grad.Rate", width = 0.5)

gghistogram(x, x = "Grad.Rate", ylab = "No. of Students", bins = 9,

          add = "mean")

ggqqplot(x, x = "Grad.Rate")

hist(x$Grad.Rate) #Plotting Histogram

plot(x$Grad.Rate) #Plotting Scatterplot


# Creating Train and Test set - maintaining % of event rate (70/30 split)
library(caret)

set.seed(3456) #Setting the seed for future use

trainIndex <- sample(seq_len(nrow(College)), size = smp_size) # as it wouldn't
balance the classes. Need upSample/downSample

college_train <- x[ trainIndex,] #Splitting the dataset

college_test <- x[ -trainIndex,] #Splitting the dataset

cat("Created college_train with ", nrow(college_train), " rows\n" ) #Created
training set

cat("Created college_test with ", nrow(college_test), " rows" ) #Created Testing
Set


table_train <- table(college_train$Private)

cat("\nTable of labels for train is = ", table_train) #table(college_train$Private))


cat("\nTable of labels for test is = ", table(college_test$Private))
#table(college_test$Private))
```

```
# Fitting the regression model using 'glm' function

formula <-  Private ~ F.Undergrad + Outstate + Terminal + P.Undergrad + Apps

college_model = glm(formula = formula, family = "binomial", data =
college_train, control = list(maxit = 50))

summary(college_model)


# Prediction model

predicted_test <- predict(college_model, college_test, type="response" )

predicted_test

install.packages("caret")

library(caret)

install.packages("InformationValue")

library(InformationValue)


# calculate Confusion Matrix with the optimal cutoff

optCutOff <- optimalCutoff(college_test$Private, predicted_test)[1]

cat("\nOptimal cutoff (threshold) is = ", optCutOff)


#Creating Confusion matrix

confusion_matrix                                                          <-
InformationValue::confusionMatrix(college_test$Private,predicted_test,
threshold = 0.5) #optCutOff) # try 0.5, explain tradeoff

confusion_matrix # predicted (columns), actual (rows)

#Creating Precision set
```

```
precision = confusion_matrix[1,1] / (confusion_matrix[1,2] + confusion_matrix[1,1])

cat("\nPrecision ", precision)

#Creating Specificity set

specificity = confusion_matrix[2,2] / (confusion_matrix[2,2] + confusion_matrix[2,1])

cat("\nSpecificity   ", specificity  )

#Creating Recall set

recall = confusion_matrix[1,1] / (confusion_matrix[2,1] + confusion_matrix[1,1])

cat("\nRecall ", recall)

#Creating Accuracy set

accuracy = (confusion_matrix[1,1] + confusion_matrix[2,2]) / sum(confusion_matrix)

cat("\nAccuracy ", accuracy)

#Installing ggplot2 package

install.packages("ggplot2")

library(ggplot2)

#Installing pROC package

install.packages("pROC")

library(pROC)

#Plotting the ROC Curve

plot(roc(college_test$Private,predicted_test, legacy.axes=TRUE))

#Determining the AUC

auc(roc(college_test$Private,predicted_test, legacy.axes=TRUE))
```