Logistic Regression on Graduate Admission Data

Joshua Jose

2023-03-14

Aim

To identify if GRE(Graduate Record Exam scores), GPA (grade point average) and prestige of the undergraduate institution, affect admission into graduate school. The response variable, admit/don't admit, is a binary variable.

Data Description

The data set contains the variables:

- admit - Admission status(Admit=1, didn't admit=0)
- gre - Graduate Record Exam scores.
- gpa - Grade point average.
- rank - Rank/Tier of the undergraduate institution.

Theory

Since the dependent variable here is binary, we are planning on using Logistic Regression to identify the student gets admitted or not. Logistic regression is a special type of the Generalized Linear Models.Here, the bivariate outcome Y has a Bernoulli distribution with parameter π (success probability $\pi \in (0,1)$). Recall that $E[Y] = \pi$. The logit link function is

$$logit(E[Y]) = log\frac{E[Y]}{1-E[Y]} = log\frac{\pi}{1-\pi} = log\frac{P(Y=1)}{P(Y=0)}$$

Then, the logit of the mean (aka log odds) is modeled as a linear combination of the covariates (regressors) X, i.e., we have a linear predictor

$$logit(E[\mathbf{Y}]) = \mathbf{X}\beta$$

where β is a vector of unknown parameters. The maximum likelihood based approach is used for the parameter estimation.In order to find this estimator we'll have to use Fisher's Scoring method which can be seen in http://hua-zhou.github.io/teaching/biostatm280-2017spring/slides/18-newton/newton.html. Predicted probability can be obtained and it is

$$\hat{P}(Y_i = 1) = \frac{1}{1 + exp(-X_i^T\beta)}$$

Exploratory Data Analysis

Loading the libraries that we'll require.

```
library("aod")
library("ggplot2")
library("ROCR")
library("twinning")
```

Loading the data set

```
admission<- read.csv("sud.csv")
head(admission)
```

```
##   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
```

Checking how the data looks

```
summary(admission)
```

```
##      admit             gre             gpa             rank
##  Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000
##  Median :0.0000   Median :580.0   Median :3.395   Median :2.000
##  Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :2.485
##  3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000
##  Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :4.000
```

```
dim(admission)
```

```
## [1] 400   4
```

```
str(admission)
```

```
## 'data.frame':    400 obs. of  4 variables:
##  $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
```

converting rank and admit to a factor as rank and admit should be treated as a categorical variable.

```
admission$rank <- factor(admission$rank)
admission$admit <- factor(admission$admit)
```

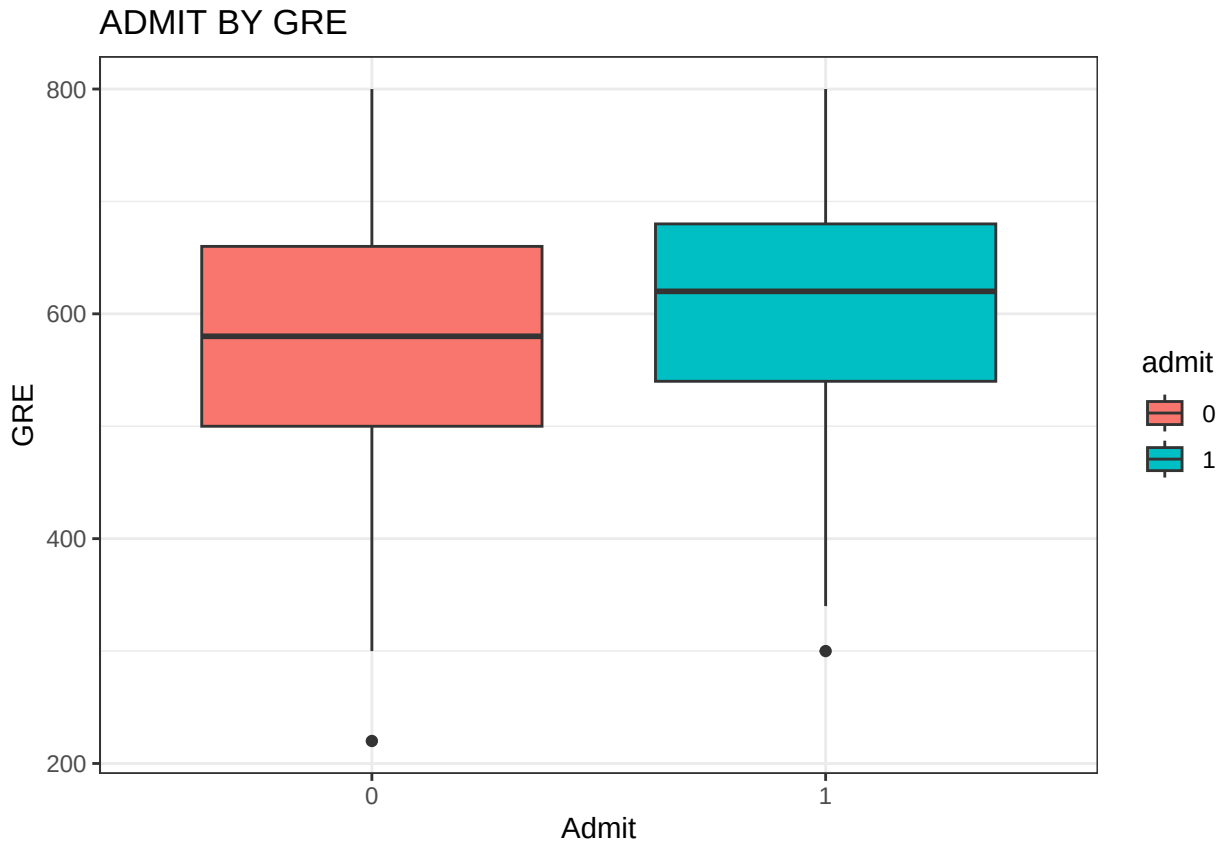
A two-way contingency table of categorical outcome and predictors we want

```
xtabs(~admit + rank, data = admission)
```

```
##       rank
## admit  1  2  3  4
##     0 28 97 93 55
##     1 33 54 28 12
```

We will explore the relationship between dependent and independent variables by way of visualization. Since gre is numeric variable and dependent variable is factor variable, we plot a box plot.

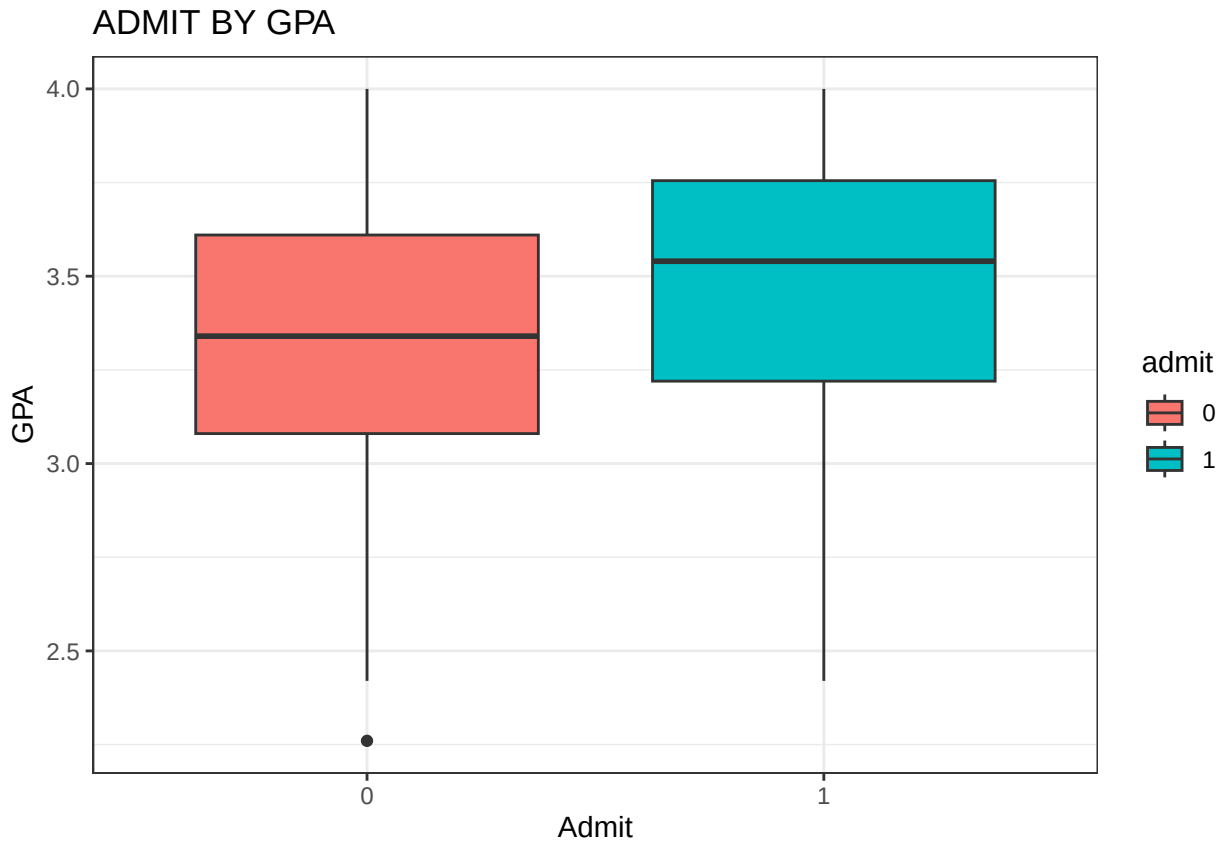
```
ggplot(admission,aes(admit,gre,fill=admit))+
  geom_boxplot()+
  theme_bw()+
  xlab("Admit")+
  ylab("GRE")+
  ggtitle("ADMIT BY GRE")
```



The two box plots are different in terms of displacement, and hence gre seems significant(at least graphically). We try the same thing for gpa

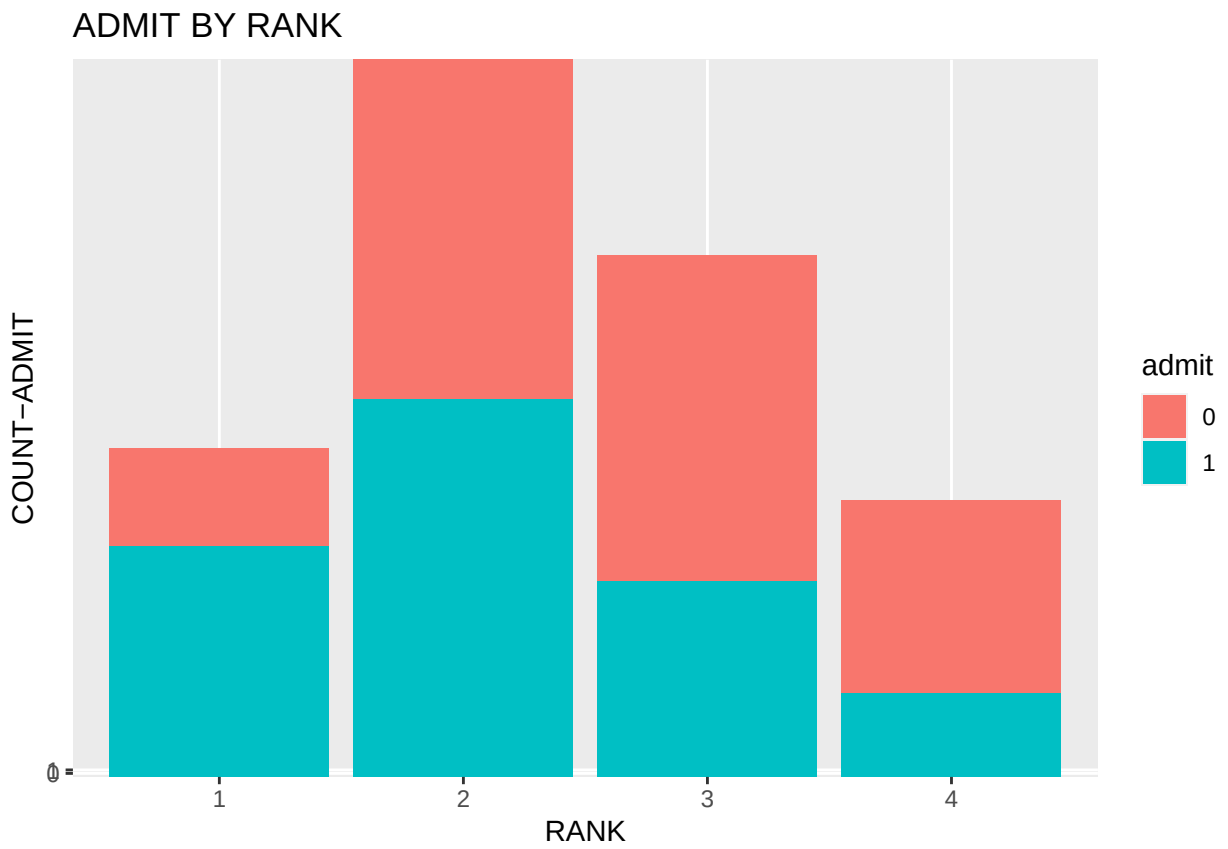
```
ggplot(admission,aes(admit,gpa,fill=admit))+
  geom_boxplot()+
  theme_bw()+
```

```
  xlab("Admit")+
  ylab("GPA")+
  ggtitle("ADMIT BY GPA")
```



There is clear difference in displacement between the two box plots, hence gpa is an important predictor. rank is a factor variable and since the dependent variable is a factor variable we plot a bar plot.

```
ggplot(admission,aes(rank,admit,fill=admit))+
  geom_col()+
  xlab("RANK")+
  ylab("COUNT-ADMIT")+
  ggtitle("ADMIT BY RANK")
```



There is a clear pattern; as rank goes from 1 to 4 the possibility of a student being admitted decreases.

Data Partition and fitting the model

Partitioning the data set into testing and training set.We do this using the 'twinning' package. twin() implements the twinning algorithm presented in Vakayil and Joseph (2022). A partition of the dataset is returned, such that the resulting two disjoint sets, termed as *twins*, are distributed similar to each other, as well as the whole dataset. Such a partition is an optimal training-testing split (Joseph and Vakayil, 2021) for training and testing statistical and machine learning models, and is model-independent. The statistical similarity also allows one to treat either of the twins as a compression (lossy) of the dataset for tractable model building on Big Data.The model is built on the training set, but its validated using the test set.

```
set.seed(385)
twin_indices = twin(admission, r=5)
admission_test = admission[twin_indices, ]
admission_train = admission[-twin_indices, ]
```

Training the model.

```
fit<- glm(admit ~ gre + gpa + rank, data = admission_train, family = "binomial")
summary(fit)
```

```
##
## Call:
```

```
## glm(formula = admit ~ gre + gpa + rank, family = "binomial",
##     data = admission_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5871  -0.8763  -0.6509   1.1739   2.0576
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.786705   1.258835  -3.008 0.002629 **
## gre          0.002077   0.001220   1.702 0.088684 .
## gpa          0.768489   0.370232   2.076 0.037922 *
## rank2       -0.640023   0.352183  -1.817 0.069171 .
## rank3       -1.265511   0.383234  -3.302 0.000959 ***
## rank4       -1.514861   0.460619  -3.289 0.001006 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 400.59  on 319  degrees of freedom
## Residual deviance: 370.53  on 314  degrees of freedom
## AIC: 382.53
##
## Number of Fisher Scoring iterations: 4
```

We see that the coeffecients of rank2, rank3,rank4 are all in comparison to rank1.

Some interpretations from the model are

- For every one unit change in gre, the log odds of admission (versus non-admission) increases by 0.002.
- For a one unit increase in gpa, the log odds of being admitted to graduate school increases by 0.768.
- The indicator variables for rank have a slightly different interpretation. For example, having attended an undergraduate institution with rank of 2, versus an institution with a rank of 1, changes the log odds of admission by -0.640.

Confidence Intervals using profiled log-likelihood as follows:

```
confint(fit)
```

```
## Waiting for profiling to be done...
```

```
##                     2.5 %      97.5 %
## (Intercept) -6.3112950218 -1.36203408
## gre         -0.0002944199  0.00450102
## gpa          0.0503752513  1.50608347
## rank2       -1.3362673558  0.04911035
## rank3       -2.0290914687 -0.52171030
## rank4       -2.4533035912 -0.63632729
```

Confidence Intervals using standard errors as follows:

```
confint.default(fit)
```

```
##                     2.5 %       97.5 %
## (Intercept) -6.2539759113 -1.319434361
## gre         -0.0003142221  0.004467669
## gpa          0.0428473330  1.494131229
## rank2       -1.3302891293  0.050242775
## rank3       -2.0166367700 -0.514386221
## rank4       -2.4176575572 -0.612064253
```

Testing

We can test for an overall effect any variable on the model using Wald test. Here we are looking at the effect of each rank

```
wald.test(b = coef(fit), Sigma = vcov(fit), Terms = 4:6)
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 15.7, df = 3, P(> X2) = 0.0013
```

The chi-squared test statistic of 15.7, with three degrees of freedom is associated with a p-value of 0.0013 indicating that the overall effect of rank is statistically significant.

If we want to test the difference (subtraction) of the terms for rank=2 and rank=3

```
l <- cbind(0, 0, 0, 1, -1, 0)
wald.test(b = coef(fit), Sigma = vcov(fit), L = l)
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 4.0, df = 1, P(> X2) = 0.047
```

The chi-squared test statistic of 4.0 with 1 degree of freedom is associated with a p-value of 0.047, indicating that the difference between the coefficient for rank=2 and the coefficient for rank=3 is statistically significant.

The odds-ratio and the 95% CI for the odds-ratio is

```
exp(cbind(OR = coef(fit), confint(fit)))
```

```
## Waiting for profiling to be done...
```

```
##                      OR      2.5 %    97.5 %
## (Intercept) 0.02267017 0.00181568 0.2561392
## gre         1.00207888 0.99970562 1.0045112
## gpa         2.15650592 1.05166566 4.5090364
## rank2       0.52728020 0.26282487 1.0503362
## rank3       0.28209497 0.13145490 0.5935046
## rank4       0.21983876 0.08600898 0.5292326
```

We can now calculate the predicted probability of admission at each value of rank, holding gre and gpa at their means.

```
admission_train1 <- with(admission_train, data.frame(gre = mean(gre),
                                                     gpa = mean(gpa),
                                                     rank = factor(1:4)))
admission_train1$rankP <- predict(fit, newdata = admission_train1, type = "response")
admission_train1
```

```
##       gre      gpa rank     rankP
## 1 588.125 3.390906    1 0.5101344
## 2 588.125 3.390906    2 0.3544626
## 3 588.125 3.390906    3 0.2270633
## 4 588.125 3.390906    4 0.1862872
```

In the above output we see that the predicted probability of being accepted into a graduate program is 0.51 for students from the highest prestige undergraduate institutions (rank=1), and 0.186 for students from the lowest ranked institutions (rank=4), holding gre and gpa at their means.

We can do something very similar to create a table of predicted probabilities varying the value of gre and rank. We are going to plot these, so we will create 100 values of gre between 200 and 800, at each value of rank (i.e., 1, 2, 3, and 4).

```
admission_train2 <- with(admission_train,data.frame(gre=rep(seq(from=200,
                                                                to=800,
                                                                length.out=100),4)
                                                    ,gpa=mean(gpa),
                                                    rank=factor(rep(1:4, each = 100))))
head(admission_train2)
```

```
##        gre      gpa rank
## 1 200.0000 3.390906    1
## 2 206.0606 3.390906    1
## 3 212.1212 3.390906    1
## 4 218.1818 3.390906    1
## 5 224.2424 3.390906    1
## 6 230.3030 3.390906    1
```

The code to generate the predicted probabilities (the first line below) is the same as before, except we are also going to ask for standard errors so we can plot a confidence interval.

We get the estimates on the link scale(so our relations will look linear when graphed) and back transform both the predicted values and confidence limits into probabilities.

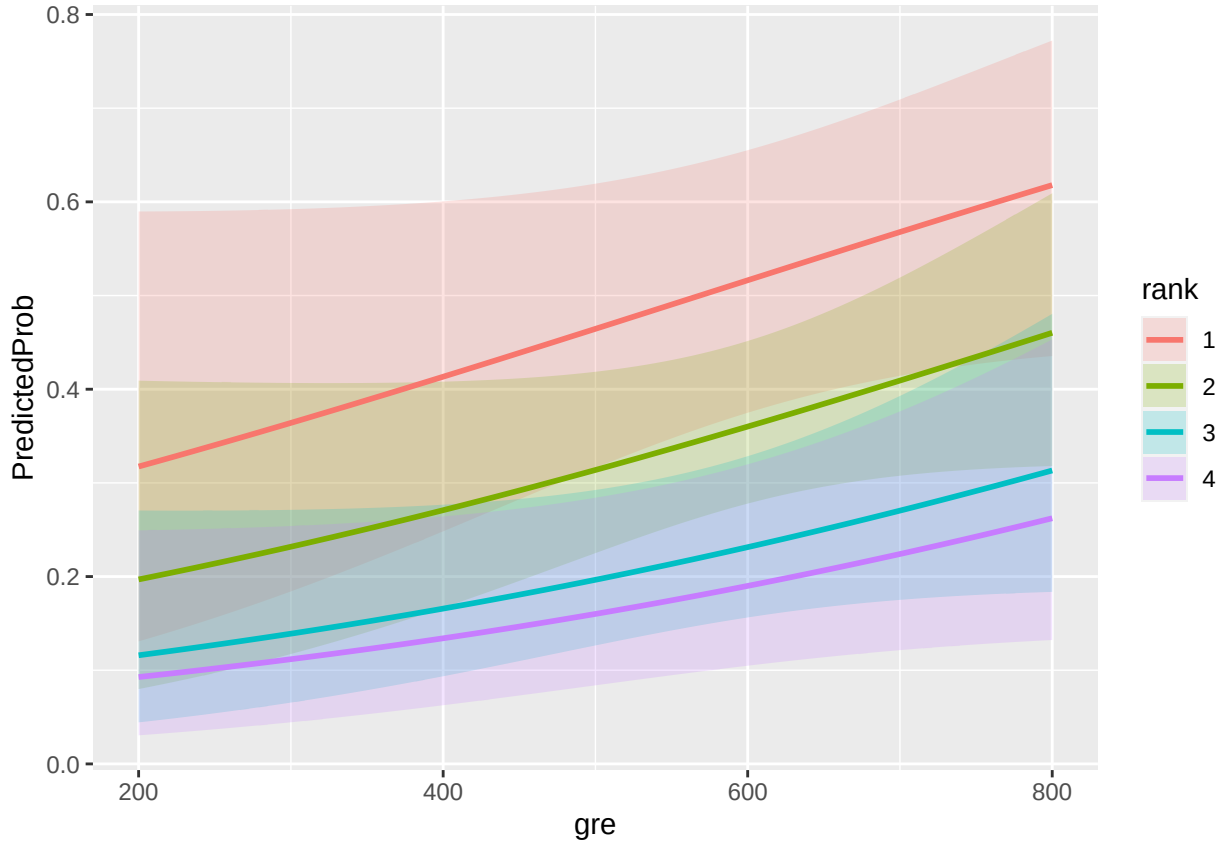
```
admission_train3 <- cbind(admission_train2, predict(fit,
                                                    newdata = admission_train2,
                                                    type="link", se=TRUE))
admission_train3 <- within(admission_train3, {
  PredictedProb <- plogis(fit)
  LL <- plogis(fit - (1.96 * se.fit))
  UL <- plogis(fit + (1.96 * se.fit))
})
head(admission_train3)
```

```
##        gre      gpa rank        fit    se.fit residual.scale        UL
## 1 200.0000 3.390906    1 -0.7654853 0.5757086              1 0.5897431
## 2 206.0606 3.390906    1 -0.7528991 0.5693655              1 0.5897803
## 3 212.1212 3.390906    1 -0.7403129 0.5630481              1 0.5898297
## 4 218.1818 3.390906    1 -0.7277267 0.5567572              1 0.5898916
## 5 224.2424 3.390906    1 -0.7151405 0.5504936              1 0.5899665
## 6 230.3030 3.390906    1 -0.7025543 0.5442585              1 0.5900549
##          LL PredictedProb
## 1 0.1308034     0.3174565
## 2 0.1336742     0.3201899
## 3 0.1365922     0.3229357
## 4 0.1395574     0.3256938
## 5 0.1425698     0.3284640
## 6 0.1456293     0.3312462
```

It can also be helpful to use graphs of predicted probabilities to understand and/or present the model.

```
ggplot(admission_train3, aes(x = gre, y = PredictedProb)) + geom_ribbon(aes(ymin = LL,
  ymax = UL, fill = rank), alpha = 0.2) + geom_line(aes(colour = rank),
  size = 1)
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
```



We may also wish to see measures of how well our model fits. This can be particularly useful when comparing competing models.

One measure of model fit is the significance of the overall model. This test asks whether the model with predictors fits significantly better than a model with just an intercept (i.e., a null model).The test statistic is the difference between the residual deviance for the model with predictors and the null model.The test statistic is distributed chi-squared with degrees of freedom equal to the differences in degrees of freedom between the current and the null model (i.e., the number of predictor variables in the model).

```
with(fit, null.deviance - deviance)
```

```
## [1] 30.05753
```

```
with(fit, df.null - df.residual)
```

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

```
with(fit, pchisq(null.deviance - deviance, df.null - df.residual,lower.tail = FALSE))
```

```
## [1] 1.436897e-05
```

The chi-square of 30.06 with 5 degrees of freedom and an associated p-value of less than 0.001 tells us that our model as a whole fits significantly better than an empty model.

Modeling and Validation

We can get the predicted probabilities π_i's and try classifying the data points in the testing set. Before that we need to find the optimal threshold value from the using the model in the training set.

```
pred_train <- predict(fit,admission_train,type = "response")
op_thresh=0
op_accuracy=0
thresh_values = seq(0.10, 0.90, by=0.01)
for(i in thresh_values){
  new_pred_train= pred_train
  new_pred_train<- ifelse(new_pred_train >i, 1, 0)
  missing_classerr <- mean(new_pred_train != admission_train$admit)
  Accuracy= 1 - missing_classerr
  if(Accuracy>op_accuracy){
    op_thresh=i
    op_accuracy=Accuracy
  }
}
print(paste('Optimal Threshold value(upto two decimal points) =', op_thresh))
```

```
## [1] "Optimal Threshold value(upto two decimal points) = 0.48"
```

There are libraries that have in-built functions that find the optimal threshold, anybody reading this is free to find one and try it, it most likely will give a better threshold value.

Now we vaidate the model using the test set.

```
pred_test <- predict(fit,admission_test,type = "response")
pred_test <- ifelse(pred_test> op_thresh,1,0)
table(admission_test$admit, pred_test)
```

```
##    pred_test
##      0  1
##   0 51  4
##   1 17  8
```

```
missing_classerr <- mean(pred_test != admission_test$admit)
print(paste('Accuracy =', 1 - missing_classerr))
```

```
## [1] "Accuracy = 0.7375"
```

From the confusion matrix we have overall accurracy of 73.75%.

Now if we plot the ROC Curve to see how good our model predicts on the test set

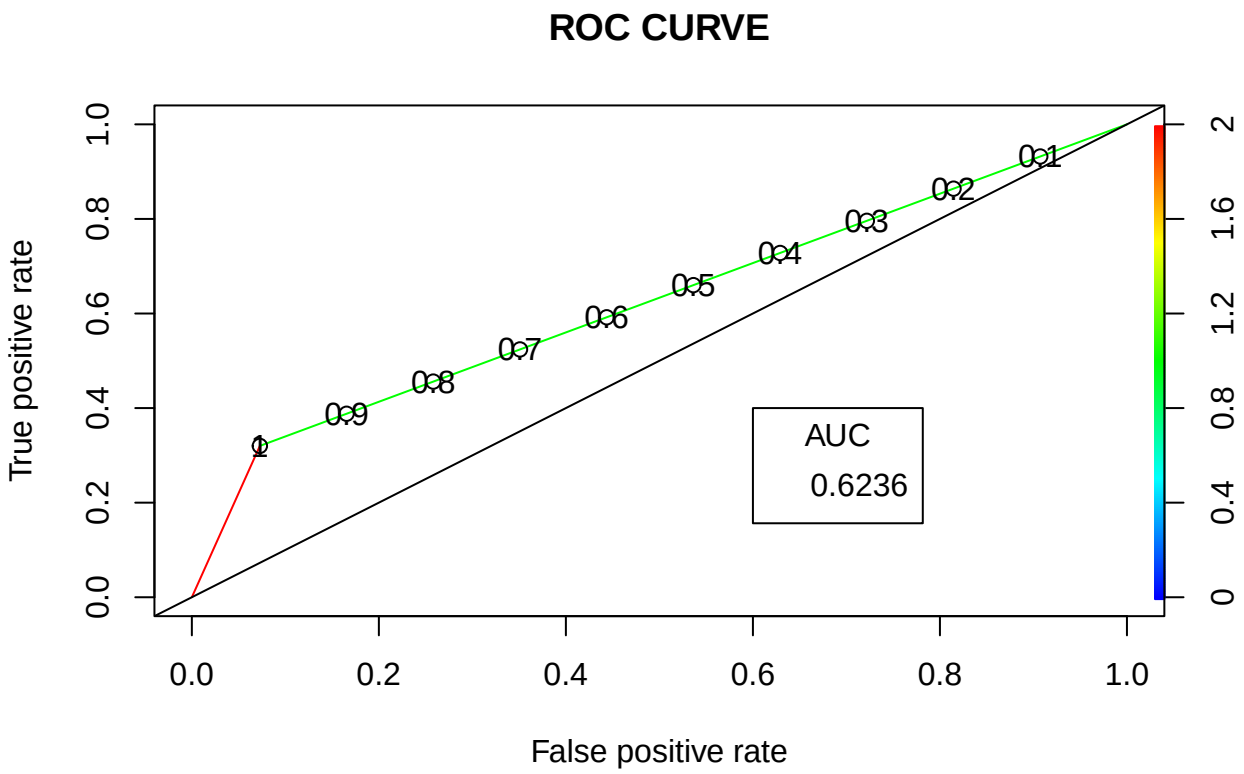
```
ROCPred <- prediction(pred_test, admission_test$admit)
ROCPer <- performance(ROCPred, measure = "tpr",
                      x.measure = "fpr")

auc <- performance(ROCPred, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.6236364
```

```
plot(ROCPer, colorize = TRUE,
     print.cutoffs.at = seq(0.1, by = 0.1),
     main = "ROC CURVE")
abline(a = 0, b = 1)

auc <- round(auc, 4)
legend(.6, .4, auc, title = "AUC", cex = 1)
```



We have an AUC of about 62.4%, which considerably good bearing in mind that our data set was small and imbalanced response variable.