Business Data Mining (IDS 572)

Linear and Logistic Regression

(sources: http://www.ats.ucla.edu/stat/r/dae/logit.htm and R and Data Mining: Examples and Case Studies book by Y. Zhao)

Regression is to build a function of independent variables (also known as predictors) to predict a dependent variable (also called response). For example, banks can assess the risk of home-loan applicants based on their age, income, expenses, occupation, number of dependents, total credit limit, etc using a regression model.

Linear Regression

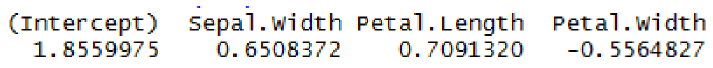
Linear regression is an approach for modeling the linear relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted by X. If there is only one explanatory variable, the regression model is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression.
To run a regression model in R we can use "lm()" function which stands for "Linear Model". To look at an example, we consider the "iris" data set. Suppose we want to predict the numerical variable "Sepal.Length" using Sepal.Width, Petal.Length, and Petal.Width as input variables.

```
> # lm(formula, data) where formula is the model description such as y ~ x and data indicates
  which data set should be used.
> fit = lm(Sepal.Length~Sepal.Width + Petal.Length + Petal.Width, data= iris)
> summary(fit)
```

```
Call:
lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width,
    data = iris)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.82816 -0.21989  0.01875  0.19709  0.84570 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)   1.85600    0.25078   7.401 9.85e-12 ***
Sepal.Width   0.65084    0.06665   9.765  < 2e-16 ***
Petal.Length  0.70913    0.05672  12.502  < 2e-16 ***
Petal.Width  -0.55648    0.12755  -4.363 2.41e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3145 on 146 degrees of freedom
Multiple R-squared:  0.8586,	Adjusted R-squared:  0.8557 
F-statistic: 295.5 on 3 and 146 DF,  p-value: < 2.2e-16
```

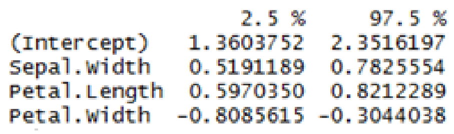
Notice that the values from the variable coefficients fall under the column Estimate. The r^2 of the fit is found by looking at Multiple R-Squared (in this case, it is 0.8586).

> # Other useful functions
> coefficients(fit) # returns model coefficients



```
(Intercept)  Sepal.Width Petal.Length  Petal.Width
  1.8559975    0.6508372    0.7091320   -0.5564827
```

>confint(fit, level=0.95) # returns 95% Confidence Intervals for model parameters



```
                  2.5 %     97.5 %
(Intercept)   1.3603752  2.3516197
Sepal.Width   0.5191189  0.7825554
Petal.Length  0.5970350  0.8212289
Petal.Width  -0.8085615 -0.3044038
```

> fitted(fit) # returns predicted values for each observation; you can also get this using fit$fitted

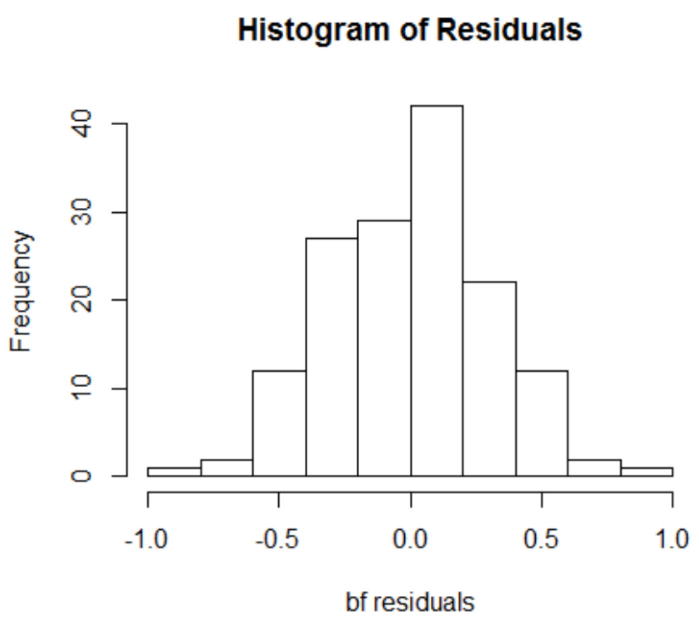
```
      1        2        3        4        5        6        7        8        9       10       11
2
.015416 4.689997 4.749251 4.825994 5.080499 5.377194 4.894684 5.021245 4.624913 4.881642 5.216496
.092158
     13       14       15       16       17       18       19       20       21       22       23
4
.745645 4.532906 5.199008 5.560786 5.093541 4.959767 5.367758 5.225932 5.163072 5.105200 4.796847
.931043
     25       26       27       28       29       30       31       32       33       34       35
5
.304898 4.831824 4.980862 5.086329 4.950332 4.961991 4.896907 4.909949 5.532480 5.471002 4.825994
.678338
```

> residuals(fit) # returns residuals of predictions

```
            1             2             3             4             5             6             7 
 0.0845842387  0.2100028184 -0.0492514176 -0.2259940935 -0.0804994772  0.0228063193 -0.2946837793 
            8             9            10            11            12            13            14 
-0.0212452413 -0.2249134657  0.0183576405  0.1835036110 -0.2921584372  0.0543545524 -0.2329058599 
           15            16            17            18            19            20            21 
 0.6009920509  0.1392141315  0.3064591029  0.1402325047  0.3322417692 -0.1259318390  0.2369283669 
           22            23            24            25            26            27            28 
-0.0051998570 -0.1968466935  0.1689568809 -0.5048980249  0.1681764266  0.0191380949  0.1136710428 
           29            30            31            32            33            34            35 
 0.2496679547 -0.2619910053 -0.0969072894  0.4900512908 -0.3324795188  0.0289982272  0.0740059065 
           36            37            38            39            40            41            42 
 0.3216617783  0.5554974346 -0.2361477432 -0.2190839857  0.0787547587  0.1111457007  0.3921502918 
           43            44            45            46            47            48            49 
-0.3492514176  0.0653509110 -0.3539363566  0.1656510844 -0.2524933009 -0.2201646135  0.0835036110 
           50            51            52            53            54            55            56 
 0.1147516706  0.5074791136  0.1049537714  0.3863847037  0.0339766623  0.3943754392 -0.4460078969 
```

Recall that residual = observed value – predicted value. Therefore, if the residual is positive, we know the observed value was larger than the predicted. If a residual is negative, we know the observed value was smaller than the predicted one.

> hist(fit$residuals, main="Histogram of Residuals", xlab = "bf residuals")



The histogram has Normal distribution shape.

> anova(fit) # returns anova table

```
Analysis of Variance Table

Response: Sepal.Length
              Df Sum Sq Mean Sq F value    Pr(>F)    
Sepal.Width    1  1.412   1.412  14.274 0.0002296 ***
Petal.Length   1 84.427  84.427 853.309 < 2.2e-16 ***
Petal.Width    1  1.883   1.883  19.035 2.413e-05 ***
Residuals    146 14.445   0.099                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

>vcov(fit) # returns the covariance matrix for the model parameters

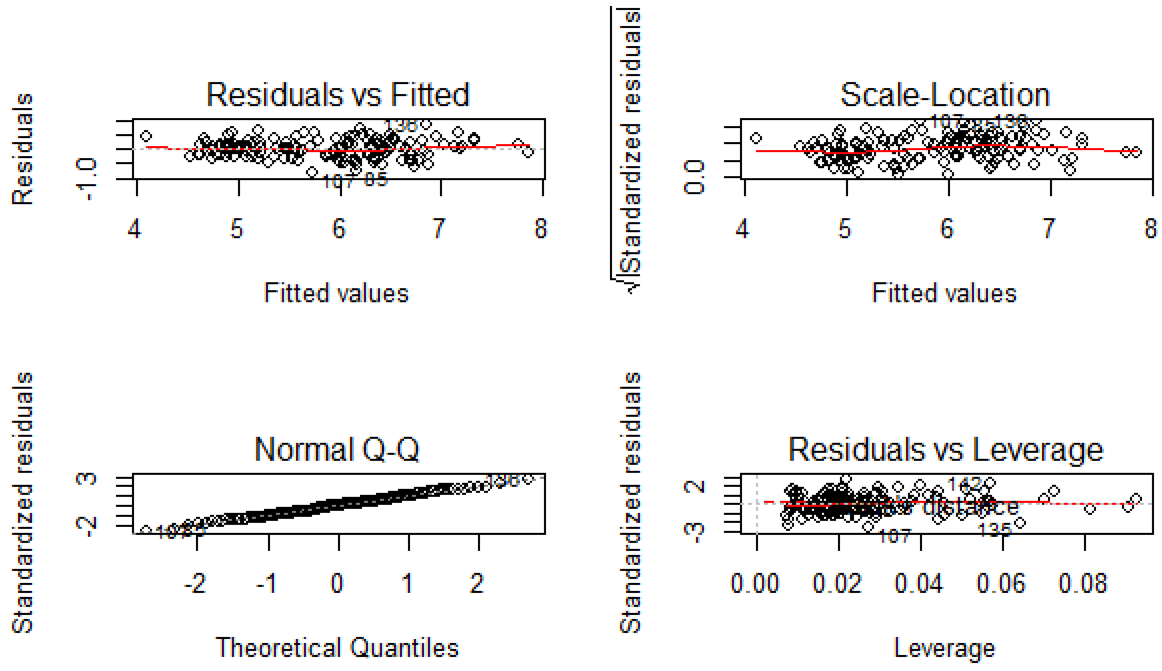
```
              (Intercept)  Sepal.Width Petal.Length  Petal.Width
(Intercept)    0.062889160 -0.015933225 -0.007264687  0.011493320
Sepal.Width   -0.015933225  0.004441875  0.001142174 -0.001617029
Petal.Length  -0.007264687  0.001142174  0.003217078 -0.006934766
Petal.Width    0.011493320 -0.001617029 -0.006934766  0.016268479
```

The function “influence()”provides the basic quantities which are used in forming a wide variety of diagnostics for checking the quality of regression fits.

> influence(fit) # regression diagnostics

The output of influence() function contains the following measures:
hat: hat matrix maps the vector of response values (dependent variable values) to the vector of fitted values (or predicted values). It describes the influence each response value has on each fitted value.
coefficients: (unless do.coef is false) is a matrix whose ith tow contains the change in the estimated coefficients which results when the ith case is dropped from the regression.
sigm: is a vector whose ith elements contains the estimate of the residual standard deviation obtained when the ith case is dropped from the regression.

```
> # plot fit
> layout(matrix(c(1,2,3,4),2,2)) #We use layout get all the four plots of fit in one picture
> plot(fit) # To view the plots, hit ‘return’
```



The first plot “ Residuals vs Fitted” gives an idea of whether there are any curvature in the data. If the red line is strongly curved, a quadratic or etc, other model may be a better choice. In this case, the curvature is not strong. The second plot “Normal Q-Q” is to check whether the residuals are normally distributed.
The third plot “Scale-Location” is used to check if the variance is constant (i.e. if the standard deviation among the residuals appears to be about constant). If the red line is strongly tilted up/down, that is a red flag. There are no issues with that in this example – the variance appears constant (the red line will always move up/down a little because of inherent randomness). The last plot “Residuals vs Leverage” is used to check to see if there were any overly influential points.

To compare two linear models we can use the “anova()” function that uses the “Chi-Square” test. anova() tests whether reduction in the residual sum of square are statistically significant or not.

```
> #compare models
> fit2 = lm(Sepal.Length~Sepal.Width , data=iris)
> anova(fit,fit2)
```

```
Analysis of Variance Table

Model 1: Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Widtl
Model 2: Sepal.Length ~ Sepal.Width
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)    
1    146  14.445                                  
2    148 100.756 -2   -86.311 436.17 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The R function step() (or the stepAIC() function from the MASS package) can be used to perform variable selection. To perform forward selection for example we need to begin by specifying a starting model and the range of models which we want to examine in the search.

```
> # forward selection
> null = lm(Sepal.Length~Sepal.Width, data= iris) # only includes one variable
> full = lm(Sepal.Length~Sepal.Width + Petal.Length + Petal.Width, data= iris) # includes all the
  variables
> # We can perform forward selection using the command:
> step(null, scope=list(lower=null, upper=full), direction="forward")
```

This tells R to start with the null model and search through models lying in the range between the null and full model using the forward selection algorithm. It gives rise to the following output:

```
Start:  AIC=-55.69
Sepal.Length ~ Sepal.Width

               Df Sum of Sq     RSS     AIC
+ Petal.Length  1    84.427  16.329 -326.66
+ Petal.Width   1    70.845  29.911 -235.86
<none>                      100.756  -55.69

Step:  AIC=-326.66
Sepal.Length ~ Sepal.Width + Petal.Length

              Df Sum of Sq    RSS     AIC
+ Petal.Width  1    1.8834 14.445 -343.04
<none>                     16.329 -326.66

Step:  AIC=-343.04
Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width


Call:
lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width, 
    data = iris)

Coefficients:
 (Intercept)   Sepal.Width  Petal.Length   Petal.Width  
      1.8560        0.6508        0.7091       -0.5565  
```

According to this procedure, the best model is the one that includes the variables Sepal.width, Petal.Length, and Petal.Width.
Notice that AIC (Akaike Information Criterion) is -2 log(likelihood) + $2p$ where p is the number of potential variables to be included in the model. AIC tries to find a model that fits the data well but has lower number of variables. In other words, we want to minimize AIC. Larger models will fit better and so have smaller RSS but use more parameters. Thus, the best choice of model will balance fit (likelihood) with model size (p).

We can perform backward elimination on the same data set using the command:

```
> step(full, data=iris, direction="backward")
```

and stepwise regression using the command:

```
> step(null, scope = list(upper=full), data=iris, direction="both")
```

Logistic Regression

Logistic regression is used to predict the probability of occurrence of an event by fitting data to a logistic curve. A logistic regression model is built as the following equation:

$$logit(y) = b_0 + b_1x_1 + b_2x_2 + \cdots + b_kx_k,$$

where $x_1, \ldots, x_k$ are predictors, y is a response to predict, and $logit(y) = \ln\left(\frac{y}{1-y}\right)$. The above equation can also be written as

$$y = \frac{1}{1 + e^{-(b_0+b_1x_1+\cdots+b_kx_k)}}$$

Logistic regression, or logit regression, or logit model, is a regression model where the dependent variable is categorical.

```
> # packages required
> library(aod)
> library(ggplot2)
> library(Rcpp)
```

Suppose we are interested in how variables such as GRE, GPA, and prestige of undergraduate institutions effect admission into graduate schools. The target variable, admit/don’t admit is a binary variable. For our data analysis below, we are going to consider a generated hypothetical data, which can be found from the website below.

```
> mydata = read.csv("http://www.ats.ucla.edu/stat/data/binary.csv")
> ## view the first few rows of the data
> head(mydata)
```

```
  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
```

This dataset has a binary response (outcome, dependent) variable called admit. There are three predictor variables: gre, gpa and rank. We will treat the variables gre and gpa as continuous. The variable rank takes on the values 1 through 4. Institutions with a rank of 1 have the highest prestige, while those with a rank of 4 have the lowest. We can get basic descriptive for the entire data set by using summary. To get the standard deviations, we use sapply() function to apply the sd function to each variable in the dataset.

```
> #summary of the data
> summary(mydata)
```

```
    admit             gre             gpa        rank
Min.   :0.0000   Min.   :220.0   Min.   :2.260   1: 61
1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   2:151
Median :0.0000   Median :580.0   Median :3.395   3:121
Mean   :0.3175   Mean   :587.7   Mean   :3.390   4: 67
3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670
Max.   :1.0000   Max.   :800.0   Max.   :4.000
```

```
> sapply(mydata, sd)
```

```
    admit         gre         gpa        rank
0.4660867 115.5165364   0.3805668   0.9444602
```

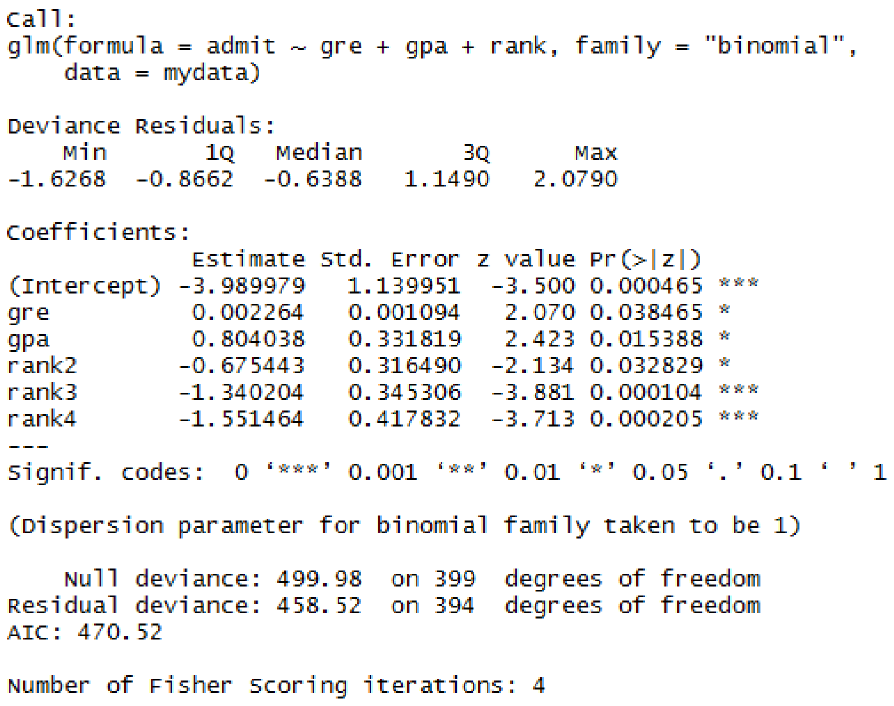
The code below estimates a logistic regression model using the “glm()” (generalized linear model) function. First, we convert the variable “rank” to a factor to indicate that rank should be treated as a categorical variable.

```
> # build model
> mydata$rank = factor(mydata$rank)
> mylogit = glm(admit ~ gre + gpa + rank, data = mydata, family = "binomial")
```

The argument family is used to describe the error distribution. This option is set to binomial, which tells R to perform logistic regression.

To get the results we can use the summary() function.

```
> # summary of model
> summary(mylogit)
```



```
Call:
glm(formula = admit ~ gre + gpa + rank, family = "binomial",
    data = mydata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6268  -0.8662  -0.6388   1.1490   2.0790

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.989979   1.139951  -3.500 0.000465 ***
gre          0.002264   0.001094   2.070 0.038465 *
gpa          0.804038   0.331819   2.423 0.015388 *
rank2       -0.675443   0.316490  -2.134 0.032829 *
rank3       -1.340204   0.345306  -3.881 0.000104 ***
rank4       -1.551464   0.417832  -3.713 0.000205 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 499.98  on 399  degrees of freedom
Residual deviance: 458.52  on 394  degrees of freedom
AIC: 470.52

Number of Fisher Scoring iterations: 4
```

- In the output above, the first thing we see is the call; this is R reminding us what the model we ran was and what options we specified, etc.
- Next, we see the deviance residuals, which are a measure of model fit. This part of output shows the distribution of the deviance residuals for individual cases used in the model. Below we discuss how to use summaries of the deviance statistic to assess model fit.
- The next part of the output shows the coefficients, their standard errors, the z-statistic (sometimes called a Wald z-statistic), and the associated p-values. Both gre and gpa are statistically significant, as are the three terms for rank. The logistic regression coefficients give the change in the log odds of the outcome for a one-unit increase in the predictor variable.
    - 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 0.804.
    - 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 -0.675.
- To test $H_0: \beta_1 = 0$, we use z = 2.070 (p-value=0.038465). Hence, the gre appears to have a significant impact on the probability of admission, while controlling for the gpa and rank.
- Below the table of coefficients are fit indices, including the null and deviance residuals and the AIC. Later we show an example of how you can use these values to help assess model fit.

We can use the confint function to obtain confidence intervals for the coefficient estimates. Note that for logistic models, confidence intervals are based on the profiled log-likelihood function. We can also get CIs based on just the standard errors by using the default method.

```
> ## CIs using profiled log-likelihood
> confint(mylogit)
```

```
##                  2.5 %   97.5 %
## (Intercept) -6.271620 -1.79255
## gre          0.000138  0.00444
## gpa          0.160296  1.46414
## rank2       -1.300889 -0.05675
## rank3       -2.027671 -0.67037
## rank4       -2.400027 -0.75354
```

```
> ## CIs using standard errors
> confint.default(mylogit)
```

```
##                 2.5 %   97.5 %
## (Intercept) -6.22424 -1.75572
## gre          0.00012  0.00441
## gpa          0.15368  1.45439
## rank2       -1.29575 -0.05513
## rank3       -2.01699 -0.66342
## rank4       -2.37040 -0.73253
```

We can also exponentiate the coefficients and interpret them as odds-ratios. To get the exponentiated coefficients, we tell R that you want to exponentiate (exp), and that the object you want to exponentiate is called coefficients and it is part of mylogit (coef(mylogit)). We can use the same logic to get odds ratios and their confidence intervals, by exponentiating the confidence intervals from before. To put it all in one table, we use cbind to bind the coefficients and confidence intervals column-wise.

```
> # odd ratio only
> exp(coef(mylogit))
```

```
## (Intercept)         gre         gpa       rank2       rank3       rank4
##      0.0185      1.0023      2.2345      0.5089      0.2618      0.2119
```

```
> # odd rations and 95% CI
> exp(cbind(OR = coef(mylogit), confint(mylogit)))
```

```
##                 OR   2.5 % 97.5 %
## (Intercept) 0.0185 0.00189  0.167
## gre         1.0023 1.00014  1.004
## gpa         2.2345 1.17386  4.324
## rank2       0.5089 0.27229  0.945
## rank3       0.2618 0.13164  0.512
## rank4       0.2119 0.09072  0.471
```

Now we can say that for a one unit increase in gpa, the odds of being admitted to graduate school (versus not being admitted) increase by a factor of 2.23.

We are often interested in using the fitted logistic regression curve to estimate probabilities and construct confidence intervals for these estimates. We can do this using the function "predict.glm()". The usage is similar to that of the function predict() which we previously used. The main difference is the option type, which tells R which type of prediction is required. The default predictions are given on the logit scale (i.e. predictions are made in terms of the log odds), while using type = "response" gives the predicted probabilities.

```
> # we first construct a new data set
> newdata1 = with(mydata, data.frame(gre = mean(gre), gpa = mean(gpa), rank = factor(1:4)))
> newdata1
```

```
##   gre  gpa rank
## 1 588 3.39    1
## 2 588 3.39    2
## 3 588 3.39    3
## 4 588 3.39    4
```

```
> newdata1$rankP = predict(mylogit, newdata = newdata1, type = “response”)
```

```
##   gre  gpa rank rankP
## 1 588 3.39    1 0.517
## 2 588 3.39    2 0.352
## 3 588 3.39    3 0.219
## 4 588 3.39    4 0.185
```

In the above output we see that the predicted probability of being accepted into a graduate program is 0.52 for students from the highest prestige undergraduate institutions (rank=1), and 0.18 for students from the lowest ranked institutions (rank=4), holding gre and gpa at their means.

We may also wish to see measures of how well our model fits. This can be particularly useful when comparing competing models. The output produced by summary(mylogit) included indices of fit (shown below the coefficients), including the null and deviance residuals and the AIC. 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). To find the difference in deviance for the two models (i.e., the test statistic) we can use the command:

```
> with(mylogit, null.deviance - deviance)
[1] 41.45903
```

The degrees of freedom for the difference between the two models is equal to the number of predictor variables in the mode, and can be obtained using:

```
> with(mylogit, df.null, df.residual)
[1] 399
```

Finally, the p-value can be obtained using

```
> with(mylogit, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail = FALSE))
[1] 7.58e-08
```

The chi-square of 41.46 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. This is sometimes called a likelihood ratio test (the deviance residual is -2*log likelihood).

Look at the example provided in the following link for the code related to multinomial logistic regression model.

http://www.ats.ucla.edu/stat/r/dae/mlogit.htm