**R: Logistic Regression**

*Goal*: Perform logistic regression using logit model for binary classification. The log odds of the outcome are modeled as a linear combination of the predictor variables.

*Data*: binary.csv available in <https://stats.idre.ucla.edu/stat/data/binary.csv> .

**Table of Contents**

1 ---- Setup and Data Description

4 ---- Logistic Regression Models (Training, Testing, and Full)

5 ---- Profiling and Interpretation

7 ---- Confusion Table (Accuracy, Sensitivity, Specificity, PPV, NPV)

9 ---- S-Curve

10 --- ROC / AUC

**Setup and Data Description**

First, install the following packages (Caret may take a while and may require installing <https://cran.rstudio.com/bin/windows/Rtools/>). Don’t forget to uncomment installation lines if running the code for the first time!

The packages we use are as follows:

* **ggplot**: allows easier plotting of data
* **pROC**: used for plotting ROC curves automatically

Then, read in the dataset from the start of the instruction set. The variables (attributes) in the dataset include the following:

1. **admit**: whether student was accepted to UCLA (0 = rejected, 1 = accepted)
2. **gre**: score on the GRE standardized test
3. **gpa**: grade point average
4. **rank**: class ranking scheme (1, 2, 3, or 4)

#Install the following packages (Caret may take a while and may require

#you to install https://cran.rstudio.com/bin/windows/Rtools/ #install.packages("ggplot2")

library(ggplot2)

# install.packages("pROC")

library(pROC) # library to draw ROC

mydata <- read.csv("https://stats.idre.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

##

# set seed for reproducibility

Random.seed <- c('Mersenne-Twister', 1)

set.seed(1)

# Convert rank to factor to indicate that rank should be treated as a categorical variable

mydata$rank <- factor(mydata$rank)

Now that we’ve read in the data, let us describe the data through visualization. First, we use bar charts to summarize the number of each class in the categorical variables **admit** and **rank**. Then, we plot histograms of the numeric variables **gre** and **gpa**, and compare the two against our target variable (admit) using a scatterplot.

# Plot the admit counts

ggplot(mydata) + geom\_bar(aes(x = factor(admit)), fill='lightblue', color='black', stat='count') +

geom\_text(stat='count',aes(x = factor(admit), label=..count..), vjust=2) +

xlab('Acceptance') + ylab('Count')

Chart

Description automatically generated

# Plot the rank counts

ggplot(mydata) + geom\_bar(aes(x = rank), fill='lightblue', color='black', stat='count') +

geom\_text(stat='count',aes(x = rank, label=..count..), vjust=5) +

xlab('Rank') + ylab('Count') + theme()

Chart, bar chart

Description automatically generated

# Plot histograms and scatter plots for gre and gpa

ggplot(mydata) + geom\_histogram(aes(x=gre, y=..density..), color="black",fill="lightblue", binwidth=40) + geom\_density(aes(x=gre), color="blue")

Chart, histogram

Description automatically generated

ggplot(mydata) + geom\_histogram(aes(x=gpa, y=..density..), color="black",fill="lightblue", binwidth=0.2) + geom\_density(aes(x=gpa), color="blue")

Chart, histogram

Description automatically generated

ggplot(mydata) + geom\_point(aes(x=gpa, y=gre, color=factor(admit))) + labs(x="GPA", y="GRE", color="Admittance") + scale\_color\_manual(labels = c("Rejected", "Accepted"), values = c("red", "blue"))

Background pattern, scatter chart

Description automatically generated

**Logistic Regression Models**

To create a logistic regression, we use the **glm** (generalized linear model) function in R. First, we split the data into training and testing, and train the model only on the training data. Then, we create a separate model fitted on all of the data. This lets us interpret the model’s overall significance and determine which individual coefficients are significant.

From the data, we see that all three predictors are significant (p < 0.05) so the parsimonious model is equal to the model with all parameters.

# Split the data into training (e.g.80%) and testing (e.g., 20%)

indices <-sample(1:nrow(mydata), 0.8 \* nrow(mydata), replace = FALSE)

train <-mydata[indices,]

test <-mydata[-indices,]

# Fit Train Model

logit.train <- glm(admit ~ gre + gpa + rank, data = train, family = "binomial")

# Fit Full Model

logit.all <- glm(admit ~ gre + gpa + rank, data = mydata, family = "binomial")

summary(logit.all)

##

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##

**Profiling and Interpretation**

Interpreting the summary above from the full model reveals the estimated model and its fit. The indices of fit include the null deviance, residual deviance, and the Akaike information criterion (AIC, the smaller the better).

The coefficients measure the effect of each independent variable. Here, we see that:

1. A one-unit increase in gre is associated with an increase in the log-odds of UCLA graduate school admittance by 0.002264
2. A one-unit increase in gre is associated with an increase in the log-odds of admittance by 0.804038
3. Having rank 2 is associated with a decrease in the log-odds of admittance by -0.675443
4. etc…

The formula for this model is then:

**logit(p) = -3.909 + 0.002\*gre + 0.804\*gpa - 0.675\*rank2 - 1.340\*rank3 - 1.551\*rank4**

**The null and residual deviance** are a measure of model fit. **Null deviance** tells us how well the target (admit) is predicted by ONLY the intercept term, while **residual deviance** informs how well the model fits the data when including independent variables (gre, gpa, and rank).

The deviance residual is -2\*log likelihood. We can view the loglikelihood of the model using:

# Calculate loglikelihood

logLik(logit.all)

#'log Lik.' -229.2587 (df=6)

We can also use these measures to test the significance of the model - whether the model with predictors fits significantly better than a model with just an intercept (null model). This is done by using a chi-square test, implemented in the code below.

The smaller the p-value the more significance the model. The p-value for this model is less than 0.0001 which means the model fits significantly better than an empty model.

# Obtain p-value for chi-square test

with(logit.all, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail = FALSE))

[1] 7.578194e-08

**Note: 7.578194e-08 = 0.00000007578194; you only need to report p < .0001.**

We can also construct a model with just an intercept for comparison to see how incorporating predictors improves prediction. Here, low p-value represents statistically significant results, while Akaike information criterion (AIC) estimates prediction error (lower error is better).

The table below compares the model with just the intercept and the model that incorporates all predictors (independent variables):

|  |  |  |
| --- | --- | --- |
|  | Intercept only | Full model |
| AIC (the smaller the better) | 501.98 | 470.52 |
| p-value | 1 | < .0001 |

logit.none <- glm(admit ~ 1, data = mydata, family = "binomial")

with(logit.none, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail = FALSE))

[1] 1

summary(logit.none)

Call:

glm(formula = admit ~ 1, family = "binomial", data = mydata)

Deviance Residuals:

Min 1Q Median 3Q Max

-0.8741 -0.8741 -0.8741 1.5148 1.5148

Coefficients:

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

(Intercept) -0.7653 0.1074 -7.125 1.04e-12 \*\*\*

---

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: 499.98 on 399 degrees of freedom

**AIC: 501.98**

Number of Fisher Scoring iterations: 4

Another metric is the **odds ratio** which can be computed by calculating e^(coefficients), as in the code below.The odd ratios show that:

1. For every unit change in gre, the log odds of being admitted to graduate school increases by a factor of 1.0022670 (log odds \* 1.0022670)
2. For every unit change in gpa, the log odds of being admitted to graduate school increases by a factor of 2.2345448
3. For every unit change in rank2, the log odds of being admitted to graduate school increases by a factor of 0.5089310
4. etc…

Furthermore, we can also compute the confidence intervals of each predictor as well to provide a range of possible values with a certain level of certainty.

# Calculate odds ratio

exp(coef(logit.all))

(Intercept) gre gpa rank2 rank3 rank4

0.0185001 1.0022670 2.2345448 0.5089310 0.2617923 0.2119375

# Confidence Intervals

confint(logit.all)

##

Waiting for profiling to be done...

2.5 % 97.5 %

(Intercept) -6.2716202334 -1.792547080

gre 0.0001375921 0.004435874

gpa 0.1602959439 1.464142727

rank2 -1.3008888002 -0.056745722

rank3 -2.0276713127 -0.670372346

rank4 -2.4000265384 -0.753542605

##

**Confusion Table**

The confusion table contains the number of true/false positive and true/false negative predictions. These can be used to calculate accuracy, sensitivity, specificity, positive predictive value (PPV), and negative predictive value (NPV).

However, logistic regression outputs a probability, not a discrete prediction, so to calculate the confusion table, we must first decide which probability corresponds to which prediction. Here, we use **ifelse** to set probabilities >0.5 as a predicted value of 1 and <=0.5 as a predicted value of 0.

Here, we create confusion tables using the test set, as well as the train set and the full dataset.

## Test set

# Evaluate on test set

predictions <- predict(logit.train, test, type="response")

# Discretize predictions; set probabilities >0.5 to 1 and <=0.5 to 0

predictions.binary <- ifelse(predictions > 0.5, 1, 0)

# Create confusion table

cftable <- table(predictions.binary, test$admit)

cftable

predictions.binary 0 1

0 47 20

1 6 7

# Calculate Accuracy

accuracy <- sum(diag(cftable))/sum(cftable)

accuracy

[1] 0.675

# Calculate Sensitivity

sensitivity<-cftable[1]/(cftable[1] + cftable[2])

sensitivity

[1] 0.8867925

# Calculate Specificity

specificity <- cftable[4]/(cftable[3] + cftable[4])

specificity

[1] 0.2592593

# Calculate Positive Predictive Value

ppv <- cftable[1]/(cftable[1] + cftable[3])

ppv

[1] 0.7014925

# Calculate Negative Predictive Value

npv <- cftable[4]/(cftable[2] + cftable[4])

npv

[1] 0.5384615

## Training set

pred.train <- predict(logit.train, train, type="response")

pred.train.binary <- ifelse(pred.train > 0.5, 1, 0)

# Create confusion table

cft.train <- table(pred.train.binary, train$admit)

# Calculate Accuracy

accuracy <- sum(diag(cft.train))/sum(cft.train)

accuracy

[1] 0.715625

# Calculate Sensitivity

sensitivity <- cft.train[1]/(cft.train[1] + cft.train[2])

sensitivity

[1] 0.9227273

# Calculate Specificity

specificity <- cft.train[4]/(cft.train[3] + cft.train[4])

specificity

[1] 0.26

# Calculate Positive Predictive Value

ppv <- cft.train[1]/(cft.train[1] + cft.train[3])

ppv

[1] 0.732852

# Calculate Negative Predictive Value

npv <- cft.train[4]/(cft.train[2] + cft.train[4])

npv

[1] 0.6046512

## Full dataset

pred.all <- predict(logit.all, mydata, type="response")

pred.all.binary <- ifelse(pred.all > 0.5, 1, 0)

# Create confusion table

cft.all <- table(pred.all.binary, mydata$admit)

# Calculate accuracy

accuracy <- sum(diag(cft.all))/sum(cft.all)

accuracy

[1] 0.71

# Calculate sensitivity.

sensitivity <- cft.all[1]/(cft.all[1] + cft.all[2])

sensitivity

[1] 0.9304029

# Calculate specificity

specificity <- cft.all[4]/(cft.all[3] + cft.all[4])

specificity

[1] 0.2362205

# Calculate Positive Predictive Value

ppv <- cft.all[1]/(cft.all[1] + cft.all[3])

ppv

[1] 0.7236467

# Calculate Negative Predictive Value

npv <- cft.all[4]/(cft.all[2] + cft.all[4])

npv

[1] 0.6122449

**S-Curve**

The S-curve plots the logistic curve against the data. This plot contains a wealth of information:

* The orange | characters are the data,(xi,yi).
* The blue “curve” is the predicted probabilities given by the fitted logistic regression. That is, p(x)=(Y=1∣X=x)
* The solid vertical black line represents the [**decision boundary**](https://en.wikipedia.org/wiki/Decision_boundary), the midpoint of S-curve where the GRE that obtains a predicted probability of 0.5.

The decision boundary is found by solving for points that satisfy

=(Yadmit=1∣X=x) = 0.5

This is equivalent to point that satisfy

+x1=0.

Thus, for logistic regression with a single predictor, the decision boundary, the midpoint of S-curve is given by corresponding x-value

x1=− / .

The code below plots the S-Curve:

gre\_mdl <- glm(admit ~ gre, data = mydata, family = "binomial")

plot(admit ~ gre, data = mydata,

col = "darkorange", pch = "|", xlim = c(0, 2500), ylim = c(0, 1),

main = "Using Logistic Regression for Classification")

abline(h = 0, lty = 3)

abline(h = 1, lty = 3)

abline(h = 0.5, lty = 2)

curve(predict(gre\_mdl, data.frame(gre = x), type = "response"),

add = TRUE, lwd = 3, col = "dodgerblue")

abline(v = -coef(gre\_mdl)[1] / coef(gre\_mdl)[2], lwd = 3)

Chart

Description automatically generated

#Note: To better see the midpoint, you can change the range of x, here GRE, to xlim = c(-1000, 2500); if you can’t see a nice s-curve, remember to adjust the xlim.

gre\_mdl <- glm(admit ~ gre, data = mydata, family = "binomial")

plot(admit ~ gre, data = mydata,

col = "darkorange", pch = "|", **xlim = c(-1000, 2500),** ylim = c(0, 1),

main = "Using Logistic Regression for Classification")

abline(h = 0, lty = 3)

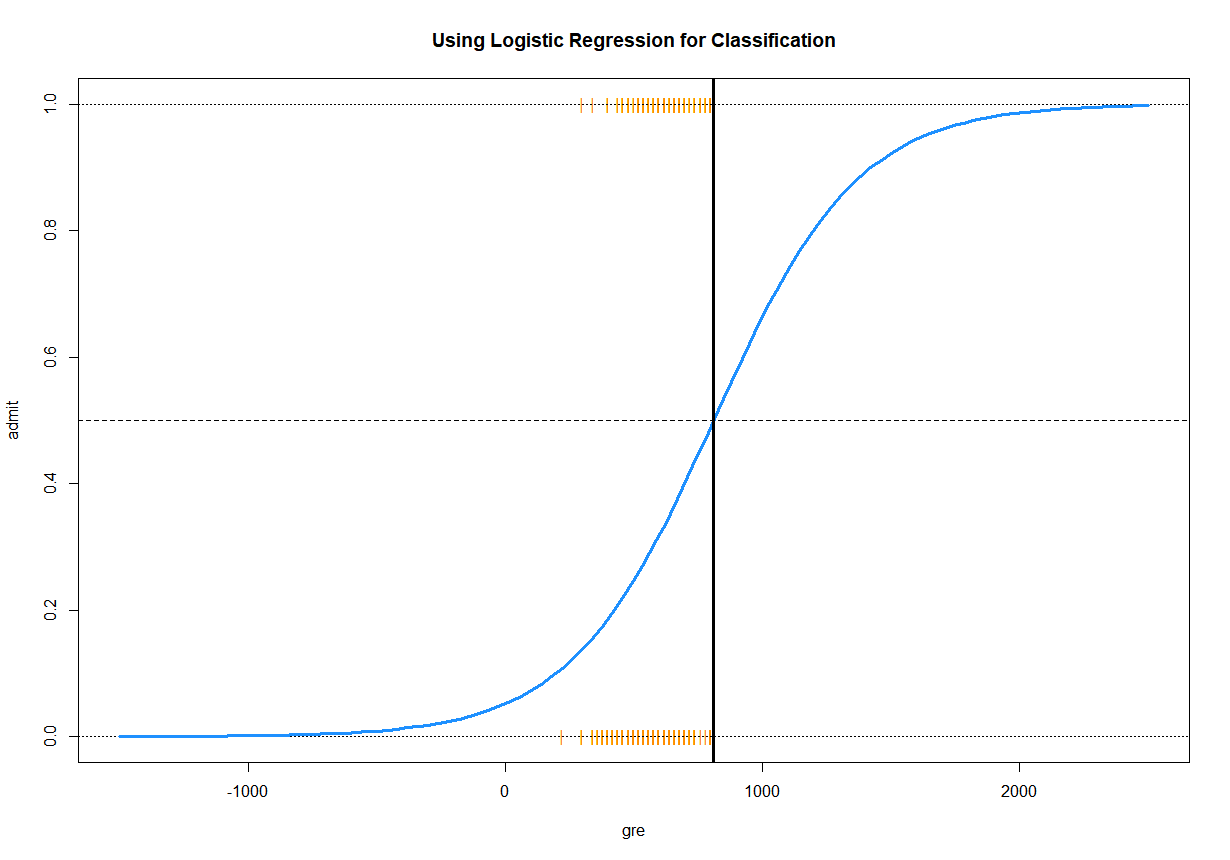
abline(h = 1, lty = 3)

abline(h = 0.5, lty = 2)

curve(predict(gre\_mdl, data.frame(gre = x), type = "response"),

add = TRUE, lwd = 3, col = "dodgerblue")

abline(v = -coef(gre\_mdl)[1] / coef(gre\_mdl)[2], lwd = 3)



**ROC Curve**

Previously, we used a discrimination threshold, the midpoint of 0.5, to distinguish between predictions of 0 (rejection) and 1 (acceptance) for the **admit** variable. However, we can also use other values.

The Receiver Operating Character Curve (ROC) measures how well the logistic regression can predict if we change the discrimination threshold. The area under the ROC (AUC) serves as another useful metric for evaluating the predictive accuracy of the logistic regression, generally classification methods.

Below, we calculate AUC for each model and plot the ROC for the test set.

# Plot ROC curve

test.prob = predict(logit.train, newdata = test, type = "response")

test.roc = roc(test$admit, test.prob)

plot.roc(test.roc, col=par("fg"),plot=TRUE,print.auc = FALSE, legacy.axes = TRUE, asp =NA)

test.roc

##

Call:

roc.default(response = test$admit, predictor = test.prob)

Data: **test.prob** in 53 controls (test$admit 0) < 27 cases (test$admit 1).

Area under the curve: **0.6349**

# Calculate AUC for train and full datasets separately

# Train

train.prob = predict(logit.train, newdata = train, type = "response")

roc(train$admit, train.prob)

#roc(test$admit, test.prob)

##

Call:

roc.default(response = train$admit, predictor = train.prob)

Data: **train.prob** in 220 controls (train$admit 0) < 100 cases (train$admit 1).

Area under the curve: **0.7067**

##

# Full model

all.prob = predict(logit.train, newdata = mydata, type = "response")

roc(mydata$admit, all.prob)

##

Call:

roc.default(response = mydata$admit, predictor = all.prob)

Data: all.prob in 273 controls (mydata$admit 0) < 127 cases (mydata$admit 1).

**Area under the curve: 0.6924**

##

# Add a smoothed ROC:

plot.roc(smooth(test.roc), col="blue", add = TRUE,plot=TRUE,print.auc = TRUE, legacy.axes = TRUE, asp =NA)

legend("bottomright", legend=c("Empirical", "Smoothed"),

col=c(par("fg"), "blue"), lwd=2)

Chart

Description automatically generated

Table summarizing the results of each model:

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Testing | Training | Full Dataset  (All Variables: In this example, the parsimonious model is the same as the full model) |
| Accuracy | 0.68 | 0.72 | 0.71 |
| Sensitivity | 0.89 | 0.92 | 0.93 |
| Specificity | 0.26 | 0.26 | 0.24 |
| PPV | 0.70 | 0.73 | 0.72 |
| NPV | 0.54 | 0.60 | 0.61 |
| AUC | 0.64 | 0.71 | 0.69 |

Link: <https://stats.idre.ucla.edu/r/dae/logit-regression/>