Game of Two Halves

Rob Carver

November 2015

This is an example of using R Markdown to share R code AND results

setwd("~/Dropbox/MyRWorkDSC201/Game2Halves")  
  
#Read file: Betting.csv contains case data  
mydata<-read.csv("Betting.csv",header=T)  
  
# Summarize the outcomes of all observed matches  
table(mydata$Match\_O)

##   
## Draw Loss Win   
## 404 392 724

This is just a tally of actual observed Draws, Wins and Losses in the data

Now we go on to create estimate the ***Multinomial Logit Model*** -- fancy term for a logistic regression in which the Y variable has more than 2 categories. In this case, we know that the Match Outcome can be "Draw", "Loss", or "Win"

We need a new package to run the procedure; the package is ***nnet***. Once it is installed and invoked, we can give the relevant command. In the following code we re-order the default sequence of the Y variable, then run the logistic regression. Following that, we display the coefficient estimates, compute p-values (significance levels) and exponentiate the coefficients to match the "Exp(B)" column in **Exhibit 7** of the case

library(nnet)  
  
mydata$result<-relevel(mydata$Match\_O, ref="Loss")  
# show that "result" just re-orders "Match\_O"  
table(mydata$Match\_O, mydata$result)

##   
## Loss Draw Win  
## Draw 0 404 0  
## Loss 392 0 0  
## Win 0 0 724

# Synatx of multinomial command includes target ~ x1 + x2 +  
# etc., data = dataframe  
test<-multinom(result~HTGD+RED.H+RED.A+POINTS\_H+POINTS\_A+  
 TOTAL\_H\_P+TOTAL\_A\_P+FGS.0+FGS.1,   
 data=mydata)

## # weights: 33 (20 variable)  
## initial value 1669.890679   
## iter 10 value 1350.231305  
## iter 20 value 1141.100550  
## iter 30 value 1133.951679  
## final value 1133.951565   
## converged

summary(test) # display estimation results

## Call:  
## multinom(formula = result ~ HTGD + RED.H + RED.A + POINTS\_H +   
## POINTS\_A + TOTAL\_H\_P + TOTAL\_A\_P + FGS.0 + FGS.1, data = mydata)  
##   
## **Coefficients**:  
## (Intercept) HTGD RED.H RED.A POINTS\_H POINTS\_A  
## Draw 3.534989 0.5111854 0.3005473 0.4629235 0.02403643 -0.01779833  
## Win 3.313188 1.6183962 -0.8102867 0.9832042 0.03520821 -0.03451276  
## TOTAL\_H\_P TOTAL\_A\_P FGS.0 FGS.1  
## Draw -0.0001797641 -0.01023752 -3.520911 -2.819490  
## Win 0.0102033252 -0.01468249 -3.319533 -2.473055  
##   
## **Std. Errors**:  
## (Intercept) HTGD RED.H RED.A POINTS\_H POINTS\_A  
## Draw 0.4620808 0.1201270 0.5722288 0.5396210 0.008885585 0.007916140  
## Win 0.4703446 0.1434815 0.7431552 0.5465152 0.009217549 0.008585501  
## TOTAL\_H\_P TOTAL\_A\_P FGS.0 FGS.1  
## Draw 0.003570172 0.003823573 0.409595 0.4261340  
## Win 0.003795461 0.003954150 0.413159 0.4299782  
##   
## Residual Deviance: 2267.903   
## AIC: 2307.903

#2-tailed z test—compute the significance levels here.  
z <- summary(test)$coefficients/summary(test)$standard.errors  
p <- (1 - pnorm(abs(z), 0, 1)) \* 2  
p

## (Intercept) HTGD RED.H RED.A POINTS\_H  
## Draw 1.998401e-14 2.08698e-05 0.5994287 0.39096550 0.0068282980  
## Win 1.865619e-12 0.00000e+00 0.2755665 0.07201193 0.0001336174  
## POINTS\_A TOTAL\_H\_P TOTAL\_A\_P FGS.0 FGS.1  
## Draw 2.455325e-02 0.95984215 0.0074179603 0.000000e+00 3.679546e-11  
## Win 5.822547e-05 0.00718175 0.0002046679 8.881784e-16 8.841182e-09

#Exponentiate  
exp(coef(test))

## (Intercept) HTGD RED.H RED.A POINTS\_H POINTS\_A TOTAL\_H\_P  
## Draw 34.29465 1.667266 1.3505978 1.588712 1.024328 0.9823591 0.9998203  
## Win 27.47257 5.044992 0.4447306 2.673007 1.035835 0.9660760 1.0102556  
## TOTAL\_A\_P FGS.0 FGS.1  
## Draw 0.9898147 0.02957249 0.05963635  
## Win 0.9854248 0.03616970 0.08432683

One standard way to assess a model is to create the *Confusion Matrix* and calculate the "misclassification rate" -- how often did our model assign a case to the wrong class?

#Misclassification error -- Confustion matrix and misclassification rate  
tab1<-table(predict(test), mydata$Match\_O)  
print(tab1)

##   
## Draw Loss Win  
## Loss 120 313 75  
## Draw 103 10 78  
## Win 181 69 571

1-sum(diag(tab1))/sum(tab1)

## [1] 0.5388158

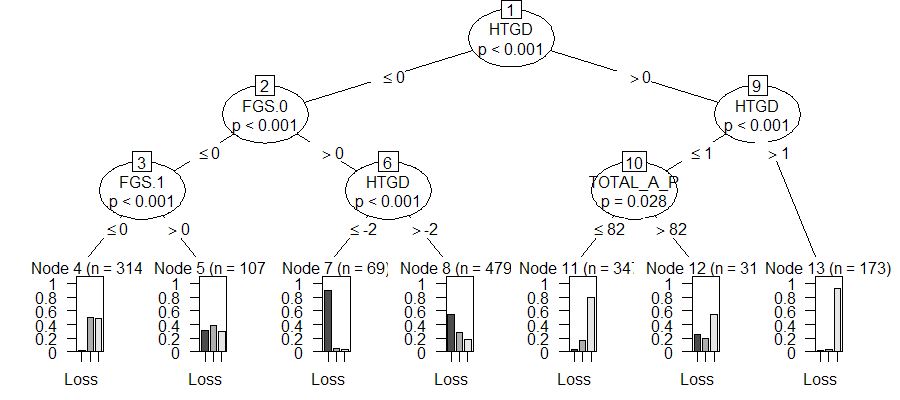
So, this model does not do a spectacular job! It is wrong more than half of the time. Let's move on to a "Decision Tree" approach

# Decision tree with party. First invoke package party, which performs recursive partitioning, an alternative to regression models  
  
library(party)

# Create a decision tree using same variables as in Exhibit 9 of case. 1st create an object (mymodel) containing the target and explanatory variables  
mymodel <-result ~ HTGD + TOTAL\_H\_P + TOTAL\_A\_P + FGS.0 + FGS.1   
  
partytree <- ctree(mymodel, data=mydata, controls=ctree\_control(mincriterion=0.90,  
 minsplit=250, maxdepth=3)) # specify some criteria for splitting  
# Note that this tree does not perfectly match the one in the case  
  
print(partytree) # print the decision rules

## Response: result   
## Inputs: HTGD, TOTAL\_H\_P, TOTAL\_A\_P, FGS.0, FGS.1   
## Number of observations: 1520   
##   
## 1) HTGD <= 0; criterion = 1, statistic = 512.524  
## 2) FGS.0 <= 0; criterion = 1, statistic = 254.288  
## 3) FGS.1 <= 0; criterion = 1, statistic = 76.495  
## 4)\* weights = 314   
## 3) FGS.1 > 0  
## 5)\* weights = 107   
## 2) FGS.0 > 0  
## 6) HTGD <= -2; criterion = 1, statistic = 31.087  
## 7)\* weights = 69   
## 6) HTGD > -2  
## 8)\* weights = 479   
## 1) HTGD > 0  
## 9) HTGD <= 1; criterion = 1, statistic = 19.558  
## 10) TOTAL\_A\_P <= 82; criterion = 0.972, statistic = 10.339  
## 11)\* weights = 347   
## 10) TOTAL\_A\_P > 82  
## 12)\* weights = 31   
## 9) HTGD > 1  
## 13)\* weights = 173

plot(partytree) # display graphically



#Misclassification error -- Confusion matrix and misclassification rate  
  
tab<-table(predict(partytree), mydata$result)   
print(tab)

##   
## Loss Draw Win  
## Loss 324 137 87  
## Draw 40 198 183  
## Win 28 69 454

1-sum(diag(tab))/sum(tab)

## [1] 0.3578947

So, this model does considerably better: It is wrong only about 36% of the time.