Logistic Regression - Book Marketing

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.0.5 v dplyr 1.0.3  
## v tidyr 1.1.2 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

books <- read.csv("C:/Users/Amara Diallo/Desktop/SPRING 2019/Predictive 353/Exam/BookMarketing.csv")  
str(books)

## 'data.frame': 50000 obs. of 13 variables:  
## $ obs\_num: int 1 2 3 4 5 6 7 8 9 10 ...  
## $ buyer\_f: int 0 0 0 0 0 1 0 0 1 0 ...  
## $ gender : int 1 1 0 0 0 0 1 0 0 0 ...  
## $ l\_pur : int 30 28 16 8 16 8 26 2 6 12 ...  
## $ n\_pur : int 10 3 2 1 1 1 1 11 11 1 ...  
## $ pur3 : int 2 0 2 0 1 0 0 2 3 0 ...  
## $ pur4 : int 0 1 0 0 0 0 1 3 2 0 ...  
## $ pur5 : int 1 0 0 1 0 0 0 0 0 0 ...  
## $ pur6 : int 0 0 0 0 0 0 0 0 3 0 ...  
## $ pur7 : int 2 1 0 0 0 0 0 3 1 0 ...  
## $ cook\_f : int 0 0 0 0 0 0 0 1 0 0 ...  
## $ atlas\_f: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ art\_f : int 0 0 0 0 0 0 0 0 0 0 ...

table(books$buyer\_f) # 45478 0's and 4522 1's

##   
## 0 1   
## 45478 4522

summary(books$buyer\_f) # mean = 0.09044 indicates proportion of 1's

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00000 0.00000 0.00000 0.09044 0.00000 1.00000

sum(books$buyer\_f) # get 4522

## [1] 4522

0.09044\*50000 # verifies 4522

## [1] 4522

In this project we are acting as marketer of books. We are interested in pushing sales for a book called, “Art History of Florence.”Our data contains 50,000 customers and the const of mailing an offer is $2 and the profit from a sal of a book is $20. Our job is to find a way reduce expenses by maximizing our profit:

## Shotgun vs Rifle

Shotgun Approach –> **We will Send offer to all 50,000 –>**

## Option 1 (Shotgun): This is what will happen if we Mail offer to everyone

* Cost of offer = $2(50,000) = $100,000
* Response rate = 9.04% (i.e. 4522 buyers)
* Profit = $20(4522) = $90,440
* Net Profit = -$9,560

## Option 2 (Rifle): Mail offer to only those who will buy

* In this case, we do a test mailing on say 10,000 customers to develop the model. Model development includes selecting the best variables and using them to put together the regression equation.
* Next, the model is applied to the remaining 40,000 assuming the researcher is blind to the results.
* Evaluate the costs and profits arising from misprediction in the model.

**Total net profit = 18*buyers + (-2)*nonbuyers –>**

18\*4522 + 45478\*(-2) # gives -9560.

## [1] -9560

45478+4522

## [1] 50000

4522/50000 # 0.09044

## [1] 0.09044

45478/50000 # 0.90956

## [1] 0.90956

## Strategy - pilot test on a subset

We will Choose a random sample of 10,000 cases from the sample then perform an analysis on the selected 10,000 cases.Based on this, build a model that can help predict who will respond; After, we apply the results of selected sample to the remaining 40,000 cases to test how well the model fits (“Validation”)

##Let’s start Splitting our Data We will Randomly split this data by using function sample.split, this will divide data into 2 sets… The first set should have 20% of the data (training) and the second set (test data) will have 80% of the data. Sample.split will make sure similar propotion of buyers are in both the first and the second set (done by specifying books$buyer\_f as the first argument)

# Install an load caTools package  
# install.packages("caTools")  
library(caTools) # for function sample.split  
  
set.seed(99)  
split <- sample.split(books$buyer\_f, SplitRatio = 0.20)  
split[1:20]

## [1] FALSE TRUE FALSE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE TRUE  
## [13] FALSE FALSE TRUE FALSE FALSE FALSE FALSE TRUE

Given a vector and a ratio value, yields a logical vector where the given ratio of the elements are assigned the value TRUE, and the remaining elements are assigned the value FALSE. This will Preserve relative ratios of categories in books$buyer\_f , for instance, splits helps so that both subsets have similar proportion of bookbuyers.

summary(split) # gives 40000 FALSE and 10000 TRUE

## Mode FALSE TRUE   
## logical 40000 10000

train <- subset(books, split ==1) # equivalent to split == TRUE  
test <- subset(books, split==0)  
table(train$buyer\_f)

##   
## 0 1   
## 9096 904

As we can see, we have 9096 0’s and 904 1’s (0.0904 = proportion of buyers)  
Baseline model, for each observation, prob(buy) = 0.0904. We wil predict buy if prob(buy) > 0.5; So for each observation, predict not buy. We would guess for each person, they are not buyers… Number of actual non-buyers = 9096 For these people, our guess would be correct. However,for 904 buyers, our guess would not be correct.  
*overall accuracy = Number of correctly predicted/total Number observations* ==>

mean(train$buyer\_f) #0.0904

## [1] 0.0904

mean(test$buyer\_f) #0.09045

## [1] 0.09045

## Logistic Regression Model

In this section we will Run logistic regression in order to predict buyer\_f based on all variables except for observation numbers (row numbers)

$$The below equation means that predictors are all variables except #buyer\_f$$

model <- glm(buyer\_f ~ .-obs\_num, data=train, family=binomial)  
summary(model)

##   
## Call:  
## glm(formula = buyer\_f ~ . - obs\_num, family = binomial, data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.0247 -0.3946 -0.2570 -0.1602 3.3524   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.267344 0.092669 -24.467 < 2e-16 \*\*\*  
## gender 1.041013 0.080634 12.910 < 2e-16 \*\*\*  
## l\_pur -0.101097 0.006399 -15.800 < 2e-16 \*\*\*  
## n\_pur -0.186712 0.031797 -5.872 4.30e-09 \*\*\*  
## pur3 -0.086199 0.055146 -1.563 0.118032   
## pur4 -0.228619 0.070129 -3.260 0.001114 \*\*   
## pur5 0.411544 0.077903 5.283 1.27e-07 \*\*\*  
## pur6 1.413970 0.069727 20.279 < 2e-16 \*\*\*  
## pur7 0.864689 0.058853 14.692 < 2e-16 \*\*\*  
## cook\_f -0.043470 0.156375 -0.278 0.781023   
## atlas\_f 0.699247 0.200725 3.484 0.000495 \*\*\*  
## art\_f 0.581233 0.153988 3.775 0.000160 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 6069.2 on 9999 degrees of freedom  
## Residual deviance: 4581.1 on 9988 degrees of freedom  
## AIC: 4605.1  
##   
## Number of Fisher Scoring iterations: 6

# model <- glm(buyer\_f ~ ., data=train, family=binomial)  
# This would mean predictors are all variables except #buyer\_f

### syntax: glm instead of lm, need to indicate family = binomial

### Let’s try all features except -obs\_num

##cook\_f has p value 0.781023, highest p value, therefore we will delete this

model <- glm(buyer\_f ~ .-obs\_num -cook\_f, data=train, family=binomial)  
summary(model)

##   
## Call:  
## glm(formula = buyer\_f ~ . - obs\_num - cook\_f, family = binomial,   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.0250 -0.3946 -0.2569 -0.1602 3.3448   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.265900 0.092520 -24.491 < 2e-16 \*\*\*  
## gender 1.040821 0.080628 12.909 < 2e-16 \*\*\*  
## l\_pur -0.101147 0.006397 -15.811 < 2e-16 \*\*\*  
## n\_pur -0.187521 0.031671 -5.921 3.20e-09 \*\*\*  
## pur3 -0.089689 0.053751 -1.669 0.095197 .   
## pur4 -0.227942 0.070085 -3.252 0.001145 \*\*   
## pur5 0.413631 0.077555 5.333 9.64e-08 \*\*\*  
## pur6 1.416603 0.069097 20.502 < 2e-16 \*\*\*  
## pur7 0.864868 0.058851 14.696 < 2e-16 \*\*\*  
## atlas\_f 0.682973 0.191944 3.558 0.000373 \*\*\*  
## art\_f 0.562584 0.138609 4.059 4.93e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 6069.2 on 9999 degrees of freedom  
## Residual deviance: 4581.2 on 9989 degrees of freedom  
## AIC: 4603.2  
##   
## Number of Fisher Scoring iterations: 6

### pur3 has p value of 0.095197 (> 0.05), not bad, but we will try deleting since there are many other variables that have important p value.

(model <- glm(buyer\_f ~ .-obs\_num -cook\_f -pur3, data=train, family=binomial))

##   
## Call: glm(formula = buyer\_f ~ . - obs\_num - cook\_f - pur3, family = binomial,   
## data = train)  
##   
## Coefficients:  
## (Intercept) gender l\_pur n\_pur pur4 pur5   
## -2.2653 1.0419 -0.1012 -0.2257 -0.1899 0.4533   
## pur6 pur7 atlas\_f art\_f   
## 1.4520 0.9032 0.6715 0.5681   
##   
## Degrees of Freedom: 9999 Total (i.e. Null); 9990 Residual  
## Null Deviance: 6069   
## Residual Deviance: 4584 AIC: 4604

glm(buyer\_f ~ .-obs\_num -cook\_f -pur3, data=train, family=binomial)

##   
## Call: glm(formula = buyer\_f ~ . - obs\_num - cook\_f - pur3, family = binomial,   
## data = train)  
##   
## Coefficients:  
## (Intercept) gender l\_pur n\_pur pur4 pur5   
## -2.2653 1.0419 -0.1012 -0.2257 -0.1899 0.4533   
## pur6 pur7 atlas\_f art\_f   
## 1.4520 0.9032 0.6715 0.5681   
##   
## Degrees of Freedom: 9999 Total (i.e. Null); 9990 Residual  
## Null Deviance: 6069   
## Residual Deviance: 4584 AIC: 4604

summary(model)

##   
## Call:  
## glm(formula = buyer\_f ~ . - obs\_num - cook\_f - pur3, family = binomial,   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.9842 -0.4003 -0.2552 -0.1613 3.3667   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.265292 0.092486 -24.493 < 2e-16 \*\*\*  
## gender 1.041880 0.080588 12.928 < 2e-16 \*\*\*  
## l\_pur -0.101156 0.006392 -15.824 < 2e-16 \*\*\*  
## n\_pur -0.225674 0.022368 -10.089 < 2e-16 \*\*\*  
## pur4 -0.189939 0.066301 -2.865 0.004173 \*\*   
## pur5 0.453320 0.073987 6.127 8.95e-10 \*\*\*  
## pur6 1.451975 0.066054 21.982 < 2e-16 \*\*\*  
## pur7 0.903236 0.054474 16.581 < 2e-16 \*\*\*  
## atlas\_f 0.671506 0.191822 3.501 0.000464 \*\*\*  
## art\_f 0.568051 0.138506 4.101 4.11e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 6069.2 on 9999 degrees of freedom  
## Residual deviance: 4584.0 on 9990 degrees of freedom  
## AIC: 4604  
##   
## Number of Fisher Scoring iterations: 6

## Overall model significant?

Null deviance = deviance of baseline (no predictors, has only intercept term) (sort of like SST) deg of freedom = n - 1 Residual deviance = deviance of this model (sort of like SSE) df = n - k - 1 Measure of improvement = chidiff = modeldeviance This has chi square distribution with deg of freedom = df of null - df of residual = modeldf.residual Get p value with this: pchisq(chidiff, dfdiff, lower.tail = F)

(chidiff = model$null.deviance - model$deviance)

## [1] 1485.232

(dfdiff = model$df.null - model$df.residual)

## [1] 9

pchisq(chidiff, dfdiff, lower.tail = F)

## [1] 2.951946e-314

## Very small p value.

### What is AIC? Can use to compare models

Akaike Information Criterion - this is like adj R^2 But not like adj R^2, lower AIC is better

* Check for multicollinearity

library(car)

## Loading required package: carData

##   
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':  
##   
## recode

## The following object is masked from 'package:purrr':  
##   
## some

vif(model)

## gender l\_pur n\_pur pur4 pur5 pur6 pur7 atlas\_f   
## 1.030244 1.051020 4.519903 1.406932 1.674383 2.061617 2.032236 1.484666   
## art\_f   
## 1.375640

vif > 5 - potential problem with multicollinearity, but no such problem here

## Make prediction

Fred: l\_pur=8, n\_pur=3, pur3=1, pur7=2, cook\_f=0, atlas\_f=0, art\_f=0

model <- glm(buyer\_f ~ gender + l\_pur + n\_pur + pur4 + pur5 + pur6 + pur7 + atlas\_f + art\_f, data=train, family=binomial)  
fred <- data.frame(obs\_num = 100, gender = 1, l\_pur=8, n\_pur=3, pur3=1, pur4 = 0, pur5 = 0, pur6 = 0, pur7=2, cook\_f=0, atlas\_f=0, art\_f=0)  
predict(model, type = "response", newdata = fred)

## 1   
## 0.2883919

model$coeff

## (Intercept) gender l\_pur n\_pur pur4 pur5   
## -2.2652915 1.0418797 -0.1011555 -0.2256743 -0.1899388 0.4533197   
## pur6 pur7 atlas\_f art\_f   
## 1.4519755 0.9032362 0.6715060 0.5680505

model$coeff\*c(1, 1, 8, 3, 0, 0, 0, 2, 0, 0)

## (Intercept) gender l\_pur n\_pur pur4 pur5   
## -2.2652915 1.0418797 -0.8092443 -0.6770230 0.0000000 0.0000000   
## pur6 pur7 atlas\_f art\_f   
## 0.0000000 1.8064723 0.0000000 0.0000000

z <- sum(model$coeff\*c(1, 1, 8, 3, 0, 0, 0, 2, 0, 0))  
prob <- 1/(1+exp(-z))  
(predict(model, newdata = fred)) # this gives the logit too

## 1   
## -0.9032068

(odds <- prob/(1-prob))

## [1] 0.405268

Bob same as Fred but has purchased “Italian Art”  
Coeff here is 0.568  
Bob’s z = Fred’s z + 0.568  
Recall z = ln of odds  
Odds = e^z  
Bob’s odds = exp(Fred’s z + 0.568) = Fred’s odds \* exp(0.568)  
= 0.405\*exp(0.568)

exp(0.568051)

## [1] 1.764824

odds\*exp(0.568051)

## [1] 0.7152267

### direct calculation

bob <- data.frame(gender = 1, l\_pur=8, n\_pur=3, pur3=1, pur4 = 0, pur5 = 0, pur6 = 0, pur7=2, cook\_f=0, atlas\_f=0, art\_f=1)  
(prob <- predict(model, type = "response", newdata = bob))

## 1   
## 0.4169866

(odds <- prob/(1-prob))

## 1   
## 0.7152263

for each person whose prob. > 0.5,we will classify them as potential buyer

model$fitted.values[1:10] # Show first 10 probabilities

## 2 6 9 12 15 20 21   
## 0.01764514 0.03556398 0.38334900 0.02461448 0.01664058 0.01925566 0.06378567   
## 22 24 29   
## 0.02518167 0.36268262 0.26269087

predictTrain <- model$fitted.values  
# or alternatively, we can do  
# predictTrain <- predict(model, type = "response")   
table(train$buyer\_f, predictTrain > 0.5)

##   
## FALSE TRUE  
## 0 9020 76  
## 1 719 185

# for each person whose prob. > 0.5, classify them as buyer

## This is a classification matrix (also called confusion matrix).

The rows correspond to observed counts (“actual”) and the columns correspond to predicted counts resulting from the model.

(9020+185)/10000 # 0.9205 = overall accuracy (increased from 0.9096)

## [1] 0.9205

9020+76 # actual 0's

## [1] 9096

719+185 # actual 1's

## [1] 904

185/904 # tpr = true positive/actual positive = 0.205 (called sensitivity)

## [1] 0.204646

9020/9096 # tnr = true negative/actual negative = 0.992 (called specificity)

## [1] 0.9916447

1-9020/9096 # fpr = false positive/actual negative = 0.0084

## [1] 0.008355321

* Overall accuracy = proportion of sample that is correctly classified
* Sensitivity = Number true positives / Number actual positives = proportion of buyers that are correctly identified
* Specificity = Number true negatives / Number actual negatives = proportion of non-buyers that are correctly identified  
  Hit rate (in marketing) = Number of true positives / Number predicted positives = proportion of predicted buyers that are actual buyers

## What if threshold is higher? (e.g., classify as buyer only if prob > 0.7)

fewer predicted positive, more predicted negative

## Let’s Make prediction on the test data (of 40,000 observations)

This will gives probabilities for test data with the model built on train data

predictTest <- predict(model, type = "response", newdata=test)   
# gives probabilities for test data with the model built on train data

### We will Build a classification matrix with the prediction on the test data

$$ The below equation is the number of predicted to buy & actually buy (True positives)$$

table(test$buyer\_f, predictTest > 0.5)

##   
## FALSE TRUE  
## 0 35946 436  
## 1 2892 726

sum(predictTest > 0.5) # predicted to buy

## [1] 1162

sum(test$buyer\_f) # actual buyers

## [1] 3618

sum(predictTest > 0.5 & test$buyer\_f) # predicted to buy & actually buy (True positives)

## [1] 726

## Economic Analysis

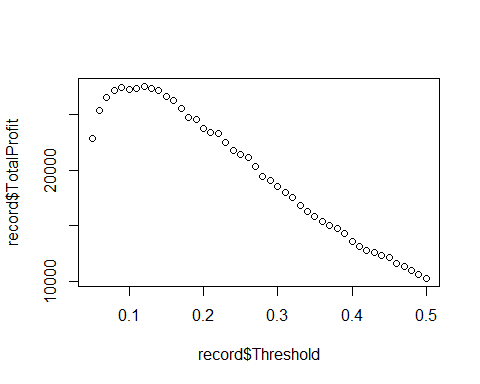
Total net profit: Profit from training data of 10,000 + profit from testing data of 40,000

profit1 <- 20\*sum(train$buyer\_f) # profit from buyers in the training group  
cost1 <- 2\*nrow(train) # cost of sending offer to everyone in the training group  
predBuyer <- predictTest > 0.4   
 # predBuyer is logical vector of who are predicted to buy from the test group  
cost2 <- 2\*sum(predBuyer) # cost of sending offer to those predicted to buy  
profit2 <- 20\*sum(predBuyer & test$buyer\_f)   
 # profit from the actual buyers who were predicted to buy  
(totalNet <- profit1 - cost1 + profit2 - cost2) # ( ) lets print the resulting totalNet value

## [1] 13622

Lowering the threshold from 0.5 seems to yield higher total net profit. We will Calculate total net profit for different threshold values: 0.5, 0.49, 0.48, …, 0.05. Use *record* to store the results.

record <- data.frame(Threshold = numeric(), TotalProfit = numeric())  
for (t in seq(0.5, 0.05, -0.01)) {  
 profit1 <- 20\*sum(train$buyer\_f)   
 cost1 <- 2\*nrow(train)   
 predBuyer <- predictTest > t   
 cost2 <- 2\*sum(predBuyer)   
 profit2 <- 20\*sum(predBuyer & test$buyer\_f)   
 totalNet <- profit1 - cost1 + profit2 - cost2  
 record <- rbind(record, data.frame(Threshold = t, TotalProfit = totalNet))  
}  
plot(record$Threshold, record$TotalProfit)



record[which.max(record$TotalProfit),] # identify the threshold with highest net profit

## Threshold TotalProfit  
## 39 0.12 27562

The optimal threshold is 0.12 with maximum total profit of $27,562. This says send the offer to the customers whose probability of purchase is higher than 0.12.

## Receiver Operator Characteristic (ROC) Curve

ROC curve can be used to (1) select a good threshold value and to (2) measure model performance

table(train$buyer\_f, predictTrain > 0.7)

##   
## FALSE TRUE  
## 0 9077 19  
## 1 817 87

9077+87 # 9164 = # correct predictions, overall accuracy = 0.9164

## [1] 9164

87/(817+87) # tpr = 0.0962

## [1] 0.09623894

19/(9077+19)

## [1] 0.00208883

* false positive = 19
* actual negative = 9077+19

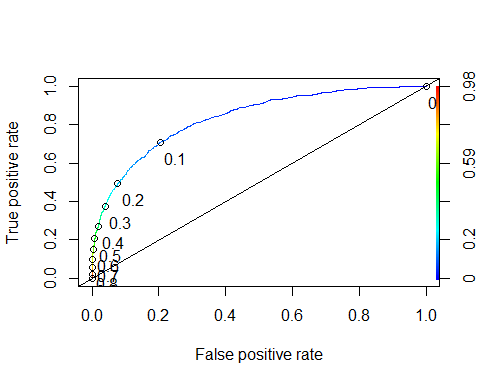
0.0021

* Increasing tpr leads to increasing fpr  
  ## What is the acceptable threshold?  
  Plot tpr and fpr for various threshold values

#install.packages("ROCR")  
library(ROCR)  
ROCRpred <- prediction(predictTrain, train$buyer\_f)

Given a vector prob’s (predictTrain) and label of 1’s and 0’s (train$buyer\_f), makes prediction for various values of thresholds

ROCRperf <- performance(ROCRpred, "tpr", "fpr")  
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0, 1, by=0.1), text.adj=c(-0.2,1.7))   
abline(a=0, b=1)



text.adj: first value - horizontal adj (neg = right), second value - vertical adj (neg = above)

If threshold t = 0, every observation will satisfy predTrain > 0, so every observation will be classfied as 1. Then tpr = 1 and fpr = 1. As t is increased from 0 to 1, both tpr and fpr decrease.  
### Choose t where tpr is relatively high with fpr acceptably low.

performance(ROCRpred, measure = "auc")@y.values

## [[1]]  
## [1] 0.8343852

0.8343852 = area under the curve This measures the performance of the model, closer to 1 the better. Minimum is 0.5 - this is equivalent to random guessing.