On the Prediction of Deliquencies

# Introduction

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

# Libraries

library(smbinning)  
library(InformationValue)  
library(randomForest)  
set.seed = 1

# Read Data

data <- read.csv(file = "mart\_delinquency.csv",  
 colClasses=c(  
 "ORDER\_DT\_HOUR" ="factor",  
 "ORDER\_DT\_MONTH" ="factor",  
 "PROFILEDESC" ="factor",  
 "TIME\_BETWEEN\_REG\_FIRST\_BIN" ="factor",  
 "DEM\_BIN" ="factor",  
 "DIS\_BIN" ="factor",  
 "SHIPTYPE" ="factor",  
 "STATE" ="factor",  
 "STATE\_BIN" ="factor",  
 "STATE\_BIN\_2" ="factor",  
 "ORDER\_DT\_HOUR\_BIN" ="factor",  
 "USERGROUP" ="factor"  
 )  
 )

# Creating Training and Test sets

# Create Training Data  
input\_ones <- data[which(data$OUTCOME == 1), ] # all 1's  
input\_zeros <-data[which(data$OUTCOME == 0), ] # all 0's  
# Create Test Data  
input\_ones\_train\_rows <- sample(1:nrow(input\_ones), 0.7 \* nrow(input\_ones)) # 1's for training  
input\_zeros\_train\_rows <- sample(1:nrow(input\_zeros), 0.7 \* nrow(input\_zeros)) # 0's for training. Pick as many 0's as 1's  
#   
training\_ones <- input\_ones[input\_ones\_train\_rows, ]   
training\_zeros <- input\_zeros[input\_zeros\_train\_rows, ]  
data\_train <- rbind(training\_ones, training\_zeros) # row bind the 1's and 0's   
#   
test\_ones <- input\_ones[-input\_ones\_train\_rows, ]  
test\_zeros <- input\_zeros[-input\_zeros\_train\_rows, ]  
data\_test <- rbind(test\_ones, test\_zeros) # row bind the 1's and 0's

# Calculating the Information Value (IV) of independent variables

In this section we calculate the IV of each independent variable. The IV is a measure of the relationship between an independent and the dependent variable.As a rule of thumb, independent variables with an IV less than 0.02 are ignored. First we select the names of variables to be investigated.

vars <- c("PROFILEDESC",   
 "TIME\_BETWEEN\_REG\_FIRST\_BIN",   
 "DEM\_BIN",   
 "DIS\_BIN",   
 "SHIPTYPE",   
 "STATE\_BIN\_2",   
 "ORDER\_DT\_HOUR\_BIN",  
 "STATE",  
 "USERGROUP",  
   
 "ORDER\_DT\_MONTH\_VAL",   
 "DEPT\_WOA",   
 "SUBCLASS\_VAL",  
 "USERGROUP\_WOA",  
 "STATE\_WOA"  
)

And we calculate the corresponding IVs using the smbinning package.

vars <- c(vars)  
df.iv <- data.frame(vars=vars, IV=numeric(length(vars)))  
for(var in vars){  
 #smb <- smbinning.factor(data\_train, y="OUTCOME", x=var, maxcat = 60) # WOE table  
 #if(class(smb) != "character"){ # heck if some error occured  
 # df.iv[df.iv$vars == var, "IV"] <- smb$iv  
 #}  
}  
#df.iv <- df.iv[order(-df.iv$IV),]  
#df.iv

## Order Attributes

### Product Type

A particular order may have multiple products. Products can be categorized to (i) Departments (Dept), (ii) Classes (Class), and (iii) Subclasses (Subclass).We review the IV of each of these attributes. We first look at the Product Department. The column DEPT\_WOA contains the Weight of Evidence (WoE) for each department. The column DEPT\_VAL contains the average deliquency for each department.

var = 'DEPT\_WOA'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.0378

var = 'DEPT\_VAL'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.041

We repeat the same exercise for class.

var = 'CLASS\_WOA'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.0236

var = 'CLASS\_VAL'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.0236

We repeat the same exercise for Subclass.

var = 'SUBCLASS\_WOA'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.0087

var = 'SUBCLASS\_VAL'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.0108

### Order Value

We investigate the Order Value and perform supervised binning. The central idea is to find those cutpoints that maximize the difference between the groups. Using ‘smbinning’ package we can quickly find the optimal cutpoints in seconds and evaluate the relationship with the target variable using metrics such as Weight of Evidence and Information Value.

var <- 'DEM'  
smb <- smbinning(data\_train, y="OUTCOME", x=var, p = 0.05)  
smb$ctree

##   
## Model formula:  
## OUTCOME ~ DEM  
##   
## Fitted party:  
## [1] root  
## | [2] DEM <= 339.2  
## | | [3] DEM <= 253.57: 0.108 (n = 7492, err = 720.1)  
## | | [4] DEM > 253.57: 0.136 (n = 7229, err = 848.6)  
## | [5] DEM > 339.2: 0.168 (n = 94480, err = 13177.7)  
##   
## Number of inner nodes: 2  
## Number of terminal nodes: 3

smb$iv

## [1] 0.017

data\_train <- smbinning.gen(data\_train, smb, chrname = "DEM\_SMBIN")  
data\_test <- smbinning.gen(data\_test , smb, chrname = "DEM\_SMBIN")

### Discount

The variable DIS represent the % of discount applied to the order. We perform a form of supervised binning as follows:

var <- 'DIS'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$ctree

##   
## Model formula:  
## OUTCOME ~ DIS  
##   
## Fitted party:  
## [1] root  
## | [2] DIS <= 0.0813: 0.179 (n = 61362, err = 9019.1)  
## | [3] DIS > 0.0813  
## | | [4] DIS <= 0.1861: 0.148 (n = 31695, err = 4007.3)  
## | | [5] DIS > 0.1861: 0.119 (n = 16144, err = 1697.0)  
##   
## Number of inner nodes: 2  
## Number of terminal nodes: 3

smb$iv

## [1] 0.028

data\_train <- smbinning.gen(data\_train, smb, chrname = "DIS\_SMBIN")  
data\_test <- smbinning.gen(data\_test , smb, chrname = "DIS\_SMBIN")

### Order Ship Type

var <- 'SHIPTYPE'  
smb <- smbinning.factor(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.0094

### Time

var = 'ORDER\_DT\_HOUR'  
smb <- smbinning.factor(data\_train, y="OUTCOME", x=var, maxcat = 50)  
smb$iv

## [1] 0.005

var = 'ORDER\_DT\_MONTH'  
smb <- smbinning.factor(data\_train, y="OUTCOME", x=var, maxcat = 50)  
smb$ctree

## NULL

smb$iv

## [1] 0.0123

## Buyer Attributes

### Time between Registration and First Purchase

var = 'TIME\_BETWEEN\_REG\_FIRST'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.0872

data\_train <- smbinning.gen(data\_train, smb, chrname = "TIME\_BETWEEN\_REG\_FIRST\_SMBIN")  
data\_test <- smbinning.gen(data\_test , smb, chrname = "TIME\_BETWEEN\_REG\_FIRST\_SMBIN")

### Geography

#var = 'STATE'  
#smb <- smbinning.factor(data\_train, y="OUTCOME", x=var, maxcat = 100)  
#smb$iv  
  
#var = 'STATE\_BIN'  
#smb <- smbinning.factor(data\_train, y="OUTCOME", x=var, maxcat = 100)  
#smb$iv  
  
var <- 'STATE\_WOA'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.0165

smb$ctree

##   
## Model formula:  
## OUTCOME ~ STATE\_WOA  
##   
## Fitted party:  
## [1] root  
## | [2] STATE\_WOA <= -0.0364  
## | | [3] STATE\_WOA <= -0.1358: 0.138 (n = 22830, err = 2709.6)  
## | | [4] STATE\_WOA > -0.1358: 0.151 (n = 26868, err = 3445.8)  
## | [5] STATE\_WOA > -0.0364  
## | | [6] STATE\_WOA <= 0.1348: 0.169 (n = 44242, err = 6216.7)  
## | | [7] STATE\_WOA > 0.1348: 0.192 (n = 15261, err = 2371.2)  
##   
## Number of inner nodes: 3  
## Number of terminal nodes: 4

data\_train <- smbinning.gen(data\_train, smb, chrname = "STATE\_WOA\_SMBIN")  
data\_test <- smbinning.gen(data\_test , smb, chrname = "STATE\_WOA\_SMBIN")

### UserGroup

var <- 'USERGROUP'  
smb <- smbinning.factor(data\_train, y="OUTCOME", x=var, maxcat = 100)  
smb$iv

## [1] 0.04

var <- 'USERGROUP\_WOA'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.0252

smb$ctree

##   
## Model formula:  
## OUTCOME ~ USERGROUP\_WOA  
##   
## Fitted party:  
## [1] root  
## | [2] USERGROUP\_WOA <= 0.128  
## | | [3] USERGROUP\_WOA <= -0.182: 0.120 (n = 7721, err = 814.2)  
## | | [4] USERGROUP\_WOA > -0.182  
## | | | [5] USERGROUP\_WOA <= -0.135  
## | | | | [6] USERGROUP\_WOA <= -0.137: 0.154 (n = 8076, err = 1052.4)  
## | | | | [7] USERGROUP\_WOA > -0.137: 0.186 (n = 9449, err = 1428.4)  
## | | | [8] USERGROUP\_WOA > -0.135: 0.154 (n = 70196, err = 9168.1)  
## | [9] USERGROUP\_WOA > 0.128  
## | | [10] USERGROUP\_WOA <= 0.413: 0.210 (n = 5485, err = 908.9)  
## | | [11] USERGROUP\_WOA > 0.413: 0.206 (n = 8274, err = 1352.5)  
##   
## Number of inner nodes: 5  
## Number of terminal nodes: 6

data\_train <- smbinning.factor.gen(data\_train, smb, chrname = "USERGROUP\_WOA\_SMBIN")  
data\_test <- smbinning.factor.gen(data\_test , smb, chrname = "USERGROUP\_WOA\_SMBIN")

### Days since Customer Launch

var <- 'DAYS\_SINCE\_LAUNCH'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.2205

smb$ctree

##   
## Model formula:  
## OUTCOME ~ DAYS\_SINCE\_LAUNCH  
##   
## Fitted party:  
## [1] root  
## | [2] DAYS\_SINCE\_LAUNCH <= 49: 0.121 (n = 5471, err = 580.4)  
## | [3] DAYS\_SINCE\_LAUNCH > 49: 0.169 (n = 100352, err = 14091.7)  
##   
## Number of inner nodes: 1  
## Number of terminal nodes: 2

data\_train <- smbinning.gen(data\_train, smb, chrname = "DAYS\_SINCE\_LAUNCH\_SMBIN")  
data\_test <- smbinning.gen(data\_test , smb, chrname = "DAYS\_SINCE\_LAUNCH\_SMBIN")

### Client Eligibles

var <- 'ELIGIBLES'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.0081

smb$ctree

##   
## Model formula:  
## OUTCOME ~ ELIGIBLES  
##   
## Fitted party:  
## [1] root  
## | [2] ELIGIBLES <= 149243  
## | | [3] ELIGIBLES <= 135014: 0.167 (n = 89652, err = 12465.7)  
## | | [4] ELIGIBLES > 135014: 0.201 (n = 6199, err = 994.4)  
## | [5] ELIGIBLES > 149243: 0.140 (n = 10050, err = 1213.6)  
##   
## Number of inner nodes: 2  
## Number of terminal nodes: 3

data\_train <- smbinning.gen(data\_train, smb, chrname = "ELIGIBLES\_SMBIN")  
data\_test <- smbinning.gen(data\_test , smb, chrname = "ELIGIBLES\_SMBIN")

### Verified Salary

var = 'VERIFIEDSALARY'  
smb <- smbinning(data\_train, y="OUTCOME", x=var)  
smb$iv

## [1] 0.4151

smb$ctree

##   
## Model formula:  
## OUTCOME ~ VERIFIEDSALARY  
##   
## Fitted party:  
## [1] root  
## | [2] VERIFIEDSALARY <= 33395.18  
## | | [3] VERIFIEDSALARY <= 26000  
## | | | [4] VERIFIEDSALARY <= 21832: 0.306 (n = 5488, err = 1164.5)  
## | | | [5] VERIFIEDSALARY > 21832: 0.334 (n = 8295, err = 1846.0)  
## | | [6] VERIFIEDSALARY > 26000  
## | | | [7] VERIFIEDSALARY <= 31220: 0.259 (n = 15760, err = 3022.8)  
## | | | [8] VERIFIEDSALARY > 31220: 0.211 (n = 7209, err = 1198.4)  
## | [9] VERIFIEDSALARY > 33395.18  
## | | [10] VERIFIEDSALARY <= 37512.09: 0.162 (n = 14194, err = 1928.7)  
## | | [11] VERIFIEDSALARY > 37512.09: 0.096 (n = 51723, err = 4501.3)  
##   
## Number of inner nodes: 5  
## Number of terminal nodes: 6

data\_train <- smbinning.gen(data\_train, smb, chrname = "VERIFIEDSALARY\_SMBIN")  
data\_test <- smbinning.gen(data\_test , smb, chrname = "VERIFIEDSALARY\_SMBIN")

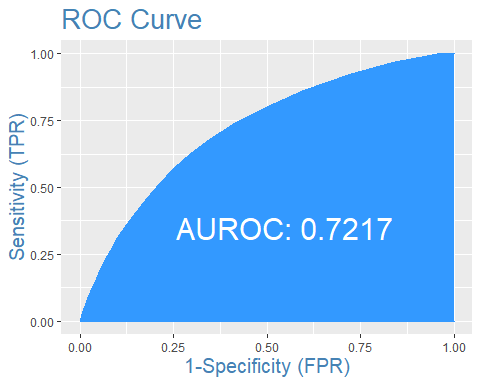
# Fitting a Logistic Regression Model

logitMod <- glm(OUTCOME ~ ORDER\_DT\_MONTH +   
 DEPT\_WOA +  
 CLASS\_WOA +  
 DEM\_SMBIN +   
 DIS\_SMBIN +  
 TIME\_BETWEEN\_REG\_FIRST\_SMBIN +  
 PROFILEDESC +  
 STATE\_WOA\_SMBIN +  
 USERGROUP +  
 ELIGIBLES\_SMBIN +  
 DAYS\_SINCE\_LAUNCH\_SMBIN +  
 VERIFIEDSALARY\_SMBIN  
 ,   
 data=data\_train,   
 family=binomial(link="logit")  
 )  
summary(logitMod)

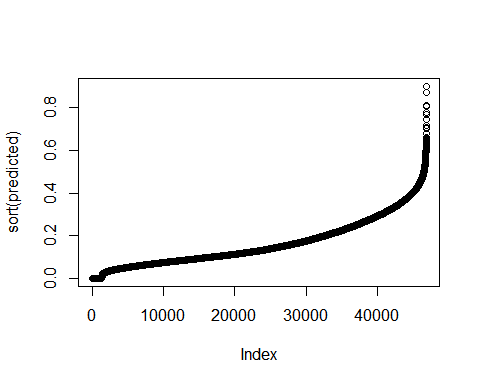
##   
## Call:  
## glm(formula = OUTCOME ~ ORDER\_DT\_MONTH + DEPT\_WOA + CLASS\_WOA +   
## DEM\_SMBIN + DIS\_SMBIN + TIME\_BETWEEN\_REG\_FIRST\_SMBIN + PROFILEDESC +   
## STATE\_WOA\_SMBIN + USERGROUP + ELIGIBLES\_SMBIN + DAYS\_SINCE\_LAUNCH\_SMBIN +   
## VERIFIEDSALARY\_SMBIN, family = binomial(link = "logit"),   
## data = data\_train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.9195 -0.6245 -0.4584 -0.3098 2.8477   
##   
## Coefficients:  
## Estimate Std. Error z value  
## (Intercept) -16.456998 41.170517 -0.400  
## ORDER\_DT\_MONTH10 -0.069205 0.047659 -1.452  
## ORDER\_DT\_MONTH11 -0.239609 0.042408 -5.650  
## ORDER\_DT\_MONTH12 -0.205438 0.042510 -4.833  
## ORDER\_DT\_MONTH2 0.029354 0.051430 0.571  
## ORDER\_DT\_MONTH3 -0.061587 0.049169 -1.253  
## ORDER\_DT\_MONTH4 0.068528 0.048708 1.407  
## ORDER\_DT\_MONTH5 0.059286 0.048228 1.229  
## ORDER\_DT\_MONTH6 0.061874 0.049094 1.260  
## ORDER\_DT\_MONTH7 0.028990 0.046943 0.618  
## ORDER\_DT\_MONTH8 0.008069 0.046850 0.172  
## ORDER\_DT\_MONTH9 -0.063937 0.048938 -1.306  
## DEPT\_WOA 1.353775 0.099490 13.607  
## CLASS\_WOA 1.786833 0.191694 9.321  
## DEM\_SMBIN02 <= 339.2 0.261767 0.052587 4.978  
## DEM\_SMBIN03 > 339.2 0.663892 0.039899 16.639  
## DIS\_SMBIN02 <= 0.1861 -0.166702 0.020661 -8.068  
## DIS\_SMBIN03 > 0.1861 -0.340073 0.028965 -11.741  
## TIME\_BETWEEN\_REG\_FIRST\_SMBIN02 <= 371 0.041047 0.022115 1.856  
## TIME\_BETWEEN\_REG\_FIRST\_SMBIN03 <= 658 -0.218436 0.034216 -6.384  
## TIME\_BETWEEN\_REG\_FIRST\_SMBIN04 <= 1082 -0.535195 0.047436 -11.283  
## TIME\_BETWEEN\_REG\_FIRST\_SMBIN05 > 1082 -0.820367 0.042244 -19.420  
## PROFILEDESCBalanced Bliss 0.213564 0.033073 6.457  
## PROFILEDESCBudget Boomers 0.240254 0.029828 8.055  
## PROFILEDESCCultural Mix 0.209481 0.043343 4.833  
## PROFILEDESCFlourishing Family 0.140446 0.038786 3.621  
## PROFILEDESCSavvy Singles 0.442660 0.030903 14.324  
## PROFILEDESCSettled in the City 0.202525 0.045787 4.423  
## PROFILEDESCUrban Crew 0.545968 0.032022 17.050  
## STATE\_WOA\_SMBIN02 <= -0.0364 -0.029674 0.027964 -1.061  
## STATE\_WOA\_SMBIN03 <= 0.1348 0.046402 0.024963 1.859  
## STATE\_WOA\_SMBIN04 > 0.1348 0.126377 0.030646 4.124  
## USERGROUPAJG -0.674411 0.209032 -3.226  
## USERGROUPARIZONA 0.274565 0.312888 0.878  
## USERGROUPASEA -0.922686 0.167819 -5.498  
## USERGROUPBOISE\_CASCADE -0.618346 0.117490 -5.263  
## USERGROUPCEHA -0.847198 0.187364 -4.522  
## USERGROUPCHS -0.908432 0.189102 -4.804  
## USERGROUPCSWG -0.274821 0.205069 -1.340  
## USERGROUPDELL\_ONLY -1.384897 0.203993 -6.789  
## USERGROUPFCG -1.311509 0.376013 -3.488  
## USERGROUPFED\_GOV -0.273898 0.181923 -1.506  
## USERGROUPKOHLS -0.606148 0.195078 -3.107  
## USERGROUPLEVI -0.785994 0.180628 -4.351  
## USERGROUPPC\_ONLY -0.872428 0.195304 -4.467  
## USERGROUPPC\_ONLY\_18\_MONTH -0.786461 0.202901 -3.876  
## USERGROUPPC\_PLUS\_APPS -0.387380 0.221011 -1.753  
## USERGROUPQUEST -1.193741 0.200854 -5.943  
## USERGROUPRETAIL 0.066047 0.435250 0.152  
## USERGROUPSTANDARD -0.940463 0.357011 -2.634  
## USERGROUPSTD\_PLUS\_APPS -1.393113 0.531592 -2.621  
## USERGROUPTBK 1.605291 0.289005 5.555  
## USERGROUPTWC -0.713594 0.183413 -3.891  
## USERGROUPTYSON -0.577423 0.185135 -3.119  
## ELIGIBLES\_SMBIN01 <= 135014 12.632861 41.176457 0.307  
## ELIGIBLES\_SMBIN02 <= 149243 12.676221 41.176476 0.308  
## ELIGIBLES\_SMBIN03 > 149243 12.033869 41.176703 0.292  
## DAYS\_SINCE\_LAUNCH\_SMBIN01 <= 49 1.332849 0.723586 1.842  
## DAYS\_SINCE\_LAUNCH\_SMBIN02 > 49 1.618908 0.722299 2.241  
## VERIFIEDSALARY\_SMBIN01 <= 21832 1.633256 0.071882 22.721  
## VERIFIEDSALARY\_SMBIN02 <= 26000 1.620979 0.068916 23.521  
## VERIFIEDSALARY\_SMBIN03 <= 31220 1.212468 0.067085 18.074  
## VERIFIEDSALARY\_SMBIN04 <= 33395.18 0.941716 0.070915 13.280  
## VERIFIEDSALARY\_SMBIN05 <= 37512.09 0.618861 0.068409 9.047  
## VERIFIEDSALARY\_SMBIN06 > 37512.09 0.057121 0.066071 0.865  
## Pr(>|z|)   
## (Intercept) 0.689357   
## ORDER\_DT\_MONTH10 0.146482   
## ORDER\_DT\_MONTH11 0.000000016040866744 \*\*\*  
## ORDER\_DT\_MONTH12 0.000001347135813053 \*\*\*  
## ORDER\_DT\_MONTH2 0.568165   
## ORDER\_DT\_MONTH3 0.210371   
## ORDER\_DT\_MONTH4 0.159456   
## ORDER\_DT\_MONTH5 0.218961   
## ORDER\_DT\_MONTH6 0.207557   
## ORDER\_DT\_MONTH7 0.536873   
## ORDER\_DT\_MONTH8 0.863259   
## ORDER\_DT\_MONTH9 0.191388   
## DEPT\_WOA < 0.0000000000000002 \*\*\*  
## CLASS\_WOA < 0.0000000000000002 \*\*\*  
## DEM\_SMBIN02 <= 339.2 0.000000643104214176 \*\*\*  
## DEM\_SMBIN03 > 339.2 < 0.0000000000000002 \*\*\*  
## DIS\_SMBIN02 <= 0.1861 0.000000000000000712 \*\*\*  
## DIS\_SMBIN03 > 0.1861 < 0.0000000000000002 \*\*\*  
## TIME\_BETWEEN\_REG\_FIRST\_SMBIN02 <= 371 0.063443 .   
## TIME\_BETWEEN\_REG\_FIRST\_SMBIN03 <= 658 0.000000000172431247 \*\*\*  
## TIME\_BETWEEN\_REG\_FIRST\_SMBIN04 <= 1082 < 0.0000000000000002 \*\*\*  
## TIME\_BETWEEN\_REG\_FIRST\_SMBIN05 > 1082 < 0.0000000000000002 \*\*\*  
## PROFILEDESCBalanced Bliss 0.000000000106569270 \*\*\*  
## PROFILEDESCBudget Boomers 0.000000000000000797 \*\*\*  
## PROFILEDESCCultural Mix 0.000001344244560088 \*\*\*  
## PROFILEDESCFlourishing Family 0.000293 \*\*\*  
## PROFILEDESCSavvy Singles < 0.0000000000000002 \*\*\*  
## PROFILEDESCSettled in the City 0.000009722785148561 \*\*\*  
## PROFILEDESCUrban Crew < 0.0000000000000002 \*\*\*  
## STATE\_WOA\_SMBIN02 <= -0.0364 0.288617   
## STATE\_WOA\_SMBIN03 <= 0.1348 0.063052 .   
## STATE\_WOA\_SMBIN04 > 0.1348 0.000037272679262275 \*\*\*  
## USERGROUPAJG 0.001254 \*\*   
## USERGROUPARIZONA 0.380205   
## USERGROUPASEA 0.000000038388496980 \*\*\*  
## USERGROUPBOISE\_CASCADE 0.000000141761620309 \*\*\*  
## USERGROUPCEHA 0.000006135194368989 \*\*\*  
## USERGROUPCHS 0.000001555844208690 \*\*\*  
## USERGROUPCSWG 0.180199   
## USERGROUPDELL\_ONLY 0.000000000011295277 \*\*\*  
## USERGROUPFCG 0.000487 \*\*\*  
## USERGROUPFED\_GOV 0.132178   
## USERGROUPKOHLS 0.001889 \*\*   
## USERGROUPLEVI 0.000013523321222624 \*\*\*  
## USERGROUPPC\_ONLY 0.000007931640395871 \*\*\*  
## USERGROUPPC\_ONLY\_18\_MONTH 0.000106 \*\*\*  
## USERGROUPPC\_PLUS\_APPS 0.079643 .   
## USERGROUPQUEST 0.000000002792851090 \*\*\*  
## USERGROUPRETAIL 0.879389   
## USERGROUPSTANDARD 0.008432 \*\*   
## USERGROUPSTD\_PLUS\_APPS 0.008776 \*\*   
## USERGROUPTBK 0.000000027832428476 \*\*\*  
## USERGROUPTWC 0.000099980089204833 \*\*\*  
## USERGROUPTYSON 0.001815 \*\*   
## ELIGIBLES\_SMBIN01 <= 135014 0.758997   
## ELIGIBLES\_SMBIN02 <= 149243 0.758196   
## ELIGIBLES\_SMBIN03 > 149243 0.770096   
## DAYS\_SINCE\_LAUNCH\_SMBIN01 <= 49 0.065474 .   
## DAYS\_SINCE\_LAUNCH\_SMBIN02 > 49 0.025005 \*   
## VERIFIEDSALARY\_SMBIN01 <= 21832 < 0.0000000000000002 \*\*\*  
## VERIFIEDSALARY\_SMBIN02 <= 26000 < 0.0000000000000002 \*\*\*  
## VERIFIEDSALARY\_SMBIN03 <= 31220 < 0.0000000000000002 \*\*\*  
## VERIFIEDSALARY\_SMBIN04 <= 33395.18 < 0.0000000000000002 \*\*\*  
## VERIFIEDSALARY\_SMBIN05 <= 37512.09 < 0.0000000000000002 \*\*\*  
## VERIFIEDSALARY\_SMBIN06 > 37512.09 0.387297   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 96510 on 109199 degrees of freedom  
## Residual deviance: 86765 on 109135 degrees of freedom  
## (1 observation deleted due to missingness)  
## AIC: 86895  
##   
## Number of Fisher Scoring iterations: 15

predicted <- predict(logitMod, data\_test, type="response") # predicted scores

plotROC(data\_test$OUTCOME, predicted)



plot(sort(predicted))



### Validation

df <- data.frame(OUTCOME=data\_test$OUTCOME, predicted)   
b <- seq(0, 1, 0.05)  
df$Bins <- cut(df$predicted, breaks = b)  
  
df.aggr.1 <- aggregate(df$OUTCOME, by=list(df$Bins), FUN = mean)  
df.aggr.2 <- aggregate(df$OUTCOME, by=list(df$Bins), FUN = length)

How many orders do we capture for scores above 35%?

v <- df.aggr.1$x \* df.aggr.2$x  
df.aggr.1$v <- v/sum(v)  
df.aggr.1

## Group.1 x v  
## 1 (0,0.05] 0.03056492 0.0192002119  
## 2 (0.05,0.1] 0.07422749 0.1186440678  
## 3 (0.1,0.15] 0.11763517 0.1541313559  
## 4 (0.15,0.2] 0.17324819 0.1365201271  
## 5 (0.2,0.25] 0.23322610 0.1344014831  
## 6 (0.25,0.3] 0.26810345 0.1235434322  
## 7 (0.3,0.35] 0.33126695 0.1132150424  
## 8 (0.35,0.4] 0.37744277 0.0895127119  
## 9 (0.4,0.45] 0.40238313 0.0581302966  
## 10 (0.45,0.5] 0.46462715 0.0321769068  
## 11 (0.5,0.55] 0.47500000 0.0125794492  
## 12 (0.55,0.6] 0.43750000 0.0046345339  
## 13 (0.6,0.65] 0.50000000 0.0015889831  
## 14 (0.65,0.7] 0.66666667 0.0005296610  
## 15 (0.7,0.75] 0.75000000 0.0003972458  
## 16 (0.75,0.8] 1.00000000 0.0002648305  
## 17 (0.8,0.85] 1.00000000 0.0002648305  
## 18 (0.85,0.9] 1.00000000 0.0002648305

i <- as.character(df.aggr.1$Group.1) >= "(0.3,"  
sum(df.aggr.1[i,]$v)

## [1] 0.3135593

### Meaningful Thresholds

Assume that the probability of default produced by the model is valid. What is a breakeven threshold, that is, the probability where the expected losses are equal to the expected benefit? It turns out that the break-even probability is equal to the Margin which is about 37.5%.

Cancelling orders with a score above 37.5% should have a positive impact. So what percent of scores are above 37.5%?

i <- !is.na(predicted) & predicted >= 0.375  
sum(i)

## [1] 2729

sum(i)/length(i)

## [1] 0.05830947

Moving this further, the score can help us quantify the expected losses for each order. That would help us prioritize orders not just by score but by the expected losses thus making our process more efficient.