## 运行环境:

linux Deepin (Ubuntu) X64 0.353

R 3.0.10

packages: caret,dplyr,plyr,xgboost,methods

## 代码:

library(caret)  
library(dplyr)  
  
##################################读取数据  
  
product = read.table("data/product.final.txt",  
 header = T,sep = "\t")   
  
product\_final = select(product,-is\_p2p) %>%  
 mutate(.,  
 early\_repayment\_na = is.na(early\_repayment)+0,  
 penalty\_na = is.na(penalty)+0,  
 apply\_ratio = fangkuan\_num/apply\_num)  
product\_final$apply\_ratio[is.na(product\_final$apply\_ratio)] <- 0   
  
apply(product\_final,2,function(x) sum(is.na(x)))  
save(product\_final,file = "newdata/product.Rdata")  
  
  
rm(list = ls())  
  
  
##########################################quality  
quality = read.table("data/quality.final.txt",  
 header = T,sep = "\t")  
  
  
quality = group\_by(quality,user\_id) %>%  
 summarise(.,  
 city\_id = first(city\_id),  
 application\_type = first(application\_type),   
 application\_term = first(application\_term),   
 application\_limit = first(application\_limit),   
 op\_type = first(op\_type),  
 col\_type = first(col\_type),   
 user\_loan\_experience = first(user\_loan\_experience),   
 user\_has\_car = first(user\_has\_car),   
 user\_income\_by\_card = first(user\_income\_by\_card),   
 user\_work\_period = first(user\_work\_period),   
 col\_value = first(col\_value),  
 house\_payment\_records = first(house\_payment\_records),   
 car\_value = first(car\_value),  
 col\_has\_mortgage = first(col\_has\_mortgage),  
 reapply\_count = all(is.na(reapply\_count)),   
 product\_type = first(product\_type),   
 apply\_from = first(apply\_from),   
 platform = first(platform),   
 spam\_score = first(spam\_score),   
 mobile\_verify = first(mobile\_verify),   
 source = first(source),   
 medium = first(medium),   
 mobile\_source = first(mobile\_source),   
 mobile\_medium = first(mobile\_medium),   
 bank\_id = first(bank\_id),  
 quality\_amount = n()   
 )  
  
  
quality\_final = quality  
save(quality\_final,file = "newdata/quality.Rdata")  
##load("newdata/quality.Rdata")  
  
user = read.table("data/user.final.txt",head = T,sep ="\t")  
me = first  
# function(x){  
# mean(x,na.rm=T)  
# }  
# name.user = names(user)  
user = group\_by(user,user\_id) %>%  
 summarise(.,pv = me(pv),  
 pv\_index\_loan = me(pv\_index\_loan),   
 pv\_apply\_total = me(pv\_apply\_total),   
 pv\_ask = me(pv\_ask),   
 pv\_calculator = me(pv\_calculator),   
 order\_count\_loan = me(order\_count\_loan),   
 pv\_daikuan = me(pv\_daikuan),   
 pv\_credit = me(pv\_credit),   
 pv\_search\_daikuan = me(pv\_search\_daikuan),   
 pv\_detail\_daikuan = me(pv\_detail\_daikuan),   
 pv\_date = me(date),user\_amount = n())  
# names(user) = c(name.user,"user\_amount")  
user\_final = user  
save(user\_final,file = "newdata/user.Rdata")  
  
rm(list = ls())  
load("newdata/product.Rdata")  
load("newdata/quality.Rdata")  
load("newdata/user.Rdata")  
train = read.table("data/order\_train.txt",header = T,sep = "\t")  
test = read.table("data/order\_test\_no\_label.txt",header = T,sep = "\t")  
  
  
  
  
train\_final = left\_join(train,user\_final,by = "user\_id") %>%  
 left\_join(.,quality\_final,by = "user\_id") %>%  
 left\_join(.,product\_final,by = "product\_id")   
  
test\_final = left\_join(test,user\_final,by = "user\_id") %>%  
 left\_join(.,quality\_final,by = "user\_id") %>%  
 left\_join(.,product\_final,by = "product\_id")   
  
  
## All completed cases  
dim(train\_final)  
# [1] 143152 59  
dim(test\_final)  
# [2] 36108 58  
  
  
yun = rbind(train\_final[,-6],test\_final)  
  
##################### 1 city\_fit  
  
city\_fit = yun %>%   
 group\_by(.,city\_id.x,city\_id.y) %>%  
 summarise(.,n = n()) %>%   
 arrange(.,desc(n)) %>%  
 summarise(.,cityFit = first(as.character(city\_id.y)))  
  
yun = left\_join(yun,city\_fit,by = 'city\_id.x') %>%  
 mutate(.,city\_fit = 0+(as.character(city\_id.y) == cityFit),  
 city\_blank = is.na(city\_id.x)) %>%  
 select(.,-c(city\_id.y,cityFit)) %>%  
 rename(.,city\_id = city\_id.x)  
   
######################### 2 limit  
  
yun = mutate(yun,big\_limit = limit>100 & limit!=200,  
 med\_limit = limit<100 & limit>50 & limit%%10!=0,  
 dig\_limit = round(limit) != limit &   
 !(limit %in% c(2.5,3.5,1.5,4.5) ))  
  
  
######################### house  
  
  
house\_function = function(col\_type,house\_payment\_records,  
 col\_has\_mortgage,col\_value){  
 a1 = col\_type %in% c(1,2,3,4,5,6,8,10,12,14,16,100)   
 a2 = house\_payment\_records == 1   
 a3 = col\_has\_mortgage == 2   
 a4 = col\_value != 0  
 any(a1,a2,a3,a4,na.rm = T)  
}  
  
yun = mutate(yun,house\_1 = col\_type %in% c(1,2,3,4,5,6,8,10,12,14,16,100),  
 house\_2 = col\_value != 0)  
  
  
table(yun$house\_1[1:143152],train\_final$result)  
  
  
train\_final = cbind(yun[1:143152,],train$result)  
test\_final = yun[143153:179260,]  
AddVariable = function(data = train,yun = yun){  
 require(dplyr)  
 prepare1 = yun %>% group\_by(.,city\_id) %>%   
 summarise(.,city\_amount = n())  
 prepare2 = yun %>% group\_by(.,bank\_id.y) %>%   
 summarise(.,bank\_amount = n())  
   
 data = left\_join(data,prepare1,by = "city\_id") %>%  
 left\_join(.,prepare2,by = "bank\_id.y")  
   
 mutate(data,weekday = factor(date%%7),  
 month = factor(date%%365%/%31),  
 fit\_user = is.na(user\_amount),  
 fit\_quality = is.na(quality\_amount),  
 big\_city = city\_amount >1000,  
 med\_bank = bank\_amount > 30,  
 big\_bank = bank\_amount >100,  
 house\_3 = house\*house\_1,  
 house\_4 = house\*house\_2\*house\_1  
 )   
   
}  
  
  
  
test\_final = AddVariable(data = test\_final,  
 yun = yun) %>%   
 select (.,-c(bank\_id.y,  
 product\_id,user\_id))  
  
  
train\_final = AddVariable(data = train\_final,  
 yun = yun) %>%   
 select (.,-c(bank\_id.y,  
 product\_id,user\_id))  
  
train\_final[is.na(train\_final)] = -10000  
test\_final[is.na(test\_final)] = -10000  
  
levels = table(trainx$city\_id) %>%  
 sort(.,decreasing = T)  
  
  
  
train\_final = cbind(train\_final,j1)  
test\_final = cbind(test\_final,j2)  
  
dim(train\_final)[1] -> n  
  
index = round(n\*0.8):n  
  
trainx = train\_final[-index,]  
testx = train\_final[index,]  
save(train\_final,test\_final,trainx,testx,file = "tr4.Rdata")  
rm(list = ls())  
  
load("tr4.Rdata")  
require(xgboost)  
require(methods)  
require(plyr)  
  
apply(trainx,2,function(x) sum(x ==-10000))  
  
load("newdata//user.Rdata")  
name = names(user\_final)  
names(trainx) %in% name  
trainx = trainx[,!names(trainx) %in% name]  
testx = testx[,!names(testx) %in% name]  
train\_final = train\_final[,!names(train\_final) %in% name]  
test\_final = test\_final[,!names(test\_final) %in% name]  
  
  
fc = function(pre=res,labels = train$result){  
 tp = sum(pre == 1 & labels == 1)/sum(pre == 1)  
 fp = sum(pre == 1 & labels == 1)/sum(labels == 1)   
 2\*tp\*fp/(tp+fp)  
}  
  
  
change = function(x){  
 as.numeric(x)  
}  
  
  
  
  
label <- as.numeric(as.character(trainx[,69]))  
  
data <- as.matrix(colwise(change)(trainx[,-69]))  
  
data2 <- as.matrix(colwise(change)(testx[,-69]))  
label2 = as.numeric(as.character(testx[,69]))  
# weight <- as.numeric(dtrain[[32]]) \* testsize / length(label)  
  
xgmat <- xgb.DMatrix(data, label = label, missing = -10000)  
param <- list("objective" = "binary:logistic",  
 "bst:eta" = 0.05,  
 "bst:max\_depth" = 5,  
 "eval\_metric" = "logloss",  
 "gamma" = 1,  
 "silent" = 1,  
 "nthread" = 16 ,  
 "min\_child\_weight" =1.45  
)  
watchlist <- list("train" = xgmat)  
nround =300  
print ("loading data end, start to boost trees")  
  
  
  
  
  
label3 <- as.numeric(as.character(train\_final[,69]))  
data3 <- as.matrix(colwise(as.numeric)(train\_final[,-69]))  
  
  
data4 <- as.matrix(colwise(as.numeric)(test\_final))  
  
xgmat <- xgb.DMatrix(data3, label = label3, missing = -10000)  
  
bst2 = xgb.train(param, xgmat, nround, watchlist);  
# bst.cv = xgb.cv(param, xgmat, nround,nfold = 10,watchlist)  
pre3 = predict(bst2,data3)  
  
ans1 = rep(0,999)  
for (i in 1:999){  
 j = 0.001\*i   
 res = pre3>j  
 ans1[i] = fc(pre=res,labels = label3)  
}  
summary(ans1)  
  
which.max(ans1)  
  
pre.final = predict(bst2,data4)  
out = pre.final>0.23  
writeLines(as.character(out),"submit/4.14.1.txt")