Week2\_NT

library(data.table)  
library(plyr)  
library(magrittr)  
library(tidyverse)  
library(lubridate)  
library(MLmetrics) # logloss를 사용하기 위한 패키지

## 챕터 0

#### **0 - 1 : [데이터1.csv]를 불러오고 [event\_datetime]의 기준으로 train set과 test set을 나누어라**

data <- fread('데이터1.csv')  
data <- data %>% mutate(date = data$event\_datetime %>% day %>%   
 str\_pad(2, side = 'left', pad = '0'))

train\_data <-   
 data %>% filter(date != 10)  
train\_data$date %>% unique()

## [1] "01" "02" "03" "04" "05" "06" "07" "08" "09"

test\_data <-   
 data %>% filter(date == 10)

#### **0 - 2 : 데이터가 나누어 졌다면, event\_datetime의 변수를 제거하라**

train\_data %<>% select(-event\_datetime, -date)  
test\_data %<>% select(-event\_datetime, -date)

#### **0 - 3 : 문자형 변수를 모두 범주형 변수로 바꾸어 주어라**

train\_data %<>% mutate\_if(is.character, as.factor)  
test\_data %<>% mutate\_if(is.character, as.factor)  
  
rm(data) # 메모리 아끼자

## 챕터 1 모델링

### **로지스틱**

**1-1 : [device\_model], [predicted\_house\_price]의 변수를 제외한 모든 X 변수를 Y 변수에 대하여 모델 적합 시켜보자**

model\_glm\_1\_1 <- glm(click ~ . , data = train\_data %>%   
 select(-c(device\_model, predicted\_house\_price)))  
#model\_glm\_1\_1 %>% summary

**1-2 : 위의 X 변수에서 [predicted\_house\_price]을 추가하여 모델 적합해보자**

model\_glm\_1\_2 <- glm(click ~ . , data = train\_data %>%   
 select(-device\_model))  
#model\_glm\_1\_2 %>% summary

**1-3 : 위의 X 변수에서 [device\_model]을 추가하여 모델 적합해보자**

model\_glm\_1\_3 <- glm(click ~ . , data = train\_data %>%   
 select(-predicted\_house\_price))  
#model\_glm\_1\_3 %>% summary

**1-4 : 위에서 만든 모델 3개의 결과를 test set에 대하여 predict 해보고 이에 대한 logloss를 구해보자. 그리고 이중에 오류가 나는 모델은 무엇이고 왜 그 오류가 나는지 서술해보자**

pred\_1\_1 <-   
 predict(model\_glm\_1\_1, newdata = test\_data %>%   
 select(-c(device\_model, predicted\_house\_price)), family = 'binomial')

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

LogLoss(pred\_1\_1, test\_data$click)

## [1] 0.3105291

pred\_1\_2 <-   
 predict(model\_glm\_1\_2, newdata = test\_data %>%   
 select(-device\_model), family = 'binomial')  
LogLoss(pred\_1\_2, test\_data$click)  
  
pred\_1\_3 <-   
 predict(model\_glm\_1\_3, newdata = test\_data %>%   
 select(-predicted\_house\_price), family = 'binomial')  
LogLoss(pred\_1\_3, test\_data$click)

2번 모델은 predicted\_house\_price에 NA가 있어서 반환값에도 NA가 많은 것 같다. 3번 모델은 트레인 데이터에는 없는 device\_model의 종류가 테스트 데이터에는 있어서 오류가 발생한다. 걍 둘다 지워버리자…

또한 AIC의 관점에서도 첫번째 모델로 충분하다! 첫번째 모델의 AIC값이 제일 작음!

logloss\_glm <- LogLoss(pred\_1\_1, test\_data$click)

그래서 두개를 지운 모델의 로그로스를 이렇게 한다.

**1-5 : [device\_model], [predicted\_house\_price]의 변수를 제외한 모든 X 변수를 Y 변수에 대하여 연결 함수를 cloglog로 바꾸어서 모델 적합 시켜보자**

model\_glm\_2 <- glm(click ~ . , data = train\_data %>%   
 select(-c(device\_model, predicted\_house\_price)), family = 'binomial'(link = cloglog))  
#model\_glm\_2 %>% summary

complementary log log 방식으로 glm을 피팅할 수 있다. probit으로도 가능하고! 근데 probit은 연산이 불편하고, cloglog의 특징은 까먹었다! 쨋든 로짓이 가장 해석이나 계산면에서 좋다고 알고 있다!

## 문제 2 : train set과 test set에서 [device\_model], [predicted\_house\_price] 변수를 삭제하자

train\_data %<>% select(-c(device\_model, predicted\_house\_price))  
test\_data %<>% select(-c(device\_model, predicted\_house\_price))

### Tree Model

library(tree)

## Warning: package 'tree' was built under R version 3.6.3

## Registered S3 method overwritten by 'tree':  
## method from  
## print.tree cli

library(rpart)  
# 라이브러리 가져오자!

set.seed(1) #시드 지정해주자!

**문제 3-1 : rpart 패키지를 이용하여 모든 X 변수를 이용하여 Y 변수에 대해 트리모델을 적합 시켜보아라**

model\_tree\_1 <- rpart(click ~ ., data = train\_data)  
model\_tree\_1

## n= 19742   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 19742 1528.7330 0.08459123   
## 2) ssp\_id=4rKKqEXsKn,5vVGidhbcl,8hNBdRzeNs,A6E0SZLhXP,CD3hRiI3bN,ddtzah8tWp,JRMcxPd0gO,ki6yt82lyq,SrN77Arvqh,tDmR2RkEPK,Uox85xVMSC,VKAHCb2KFB,wWGPlLy4jH 11755 348.0361 0.03054020 \*  
## 3) ssp\_id=k5ASnQBVC3,M6QaRvdZ8h,nwf1A3O5cO,y7QKxSwhwV 7987 1095.8100 0.16414170   
## 6) advertisement\_id=7Jed7iZc4g,9gyihRjPVK,9XoBTDdtSl,B12nM0QBme,Hy6SSlFrrj,K6ZwviDgnR,kPHyEU7nLt,OPLghzPS1i,PrrNeme8gU,RDcrDg5GGQ,T36AMrQrkF,TbkcVoisoR,tmj7wCMWB5,VMHkGmzhR9,xDjSV8yqPL 5296 606.7415 0.13198640 \*  
## 7) advertisement\_id=7dyzy9aZoJ,BD8n5KtMc3,CCWN3Vsn9P,dChiDwvVws,kZP6ssxfMO,qTUiMFHa1k,thYsZvmQHI,zdUcuwnbGE 2691 472.8161 0.22742470 \*

pred\_tree\_1 <- predict(model\_tree\_1, newdata = test\_data)  
LogLoss(pred\_tree\_1, test\_data$click)

## [1] 0.2551373

2번 가지를 만들었고, logloss는 0.2551

**문제 3-2 : tree 패키지를 이용하여 모든 X 변수를 이용하여 Y 변수에 대해 트리모델을 적합 시켜보아라**

model\_tree\_2 <- tree(click ~ ., data = train\_data)  
model\_tree\_2

## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 19742 1529.0 0.08459   
## 2) ssp\_id: 4rKKqEXsKn,5vVGidhbcl,8hNBdRzeNs,A6E0SZLhXP,CD3hRiI3bN,ddtzah8tWp,JRMcxPd0gO,ki6yt82lyq,SrN77Arvqh,tDmR2RkEPK,Uox85xVMSC,VKAHCb2KFB,wWGPlLy4jH 11755 348.0 0.03054 \*  
## 3) ssp\_id: k5ASnQBVC3,M6QaRvdZ8h,nwf1A3O5cO,y7QKxSwhwV 7987 1096.0 0.16410   
## 6) advertisement\_id: 7Jed7iZc4g,9gyihRjPVK,9XoBTDdtSl,B12nM0QBme,Hy6SSlFrrj,K6ZwviDgnR,kPHyEU7nLt,OPLghzPS1i,PrrNeme8gU,RDcrDg5GGQ,T36AMrQrkF,TbkcVoisoR,tmj7wCMWB5,VMHkGmzhR9,xDjSV8yqPL 5296 606.7 0.13200 \*  
## 7) advertisement\_id: 7dyzy9aZoJ,BD8n5KtMc3,CCWN3Vsn9P,dChiDwvVws,kZP6ssxfMO,qTUiMFHa1k,thYsZvmQHI,zdUcuwnbGE 2691 472.8 0.22740 \*

pred\_tree\_2 <- predict(model\_tree\_2, newdata = test\_data)  
LogLoss(pred\_tree\_2, test\_data$click)

## [1] 0.2560239

노드를 안나눠버림! 클래스 불균형이라서! 18072 vs 1670이니까 걍 다 0이라고 예측해도 91%의 정분류율을 가진다.

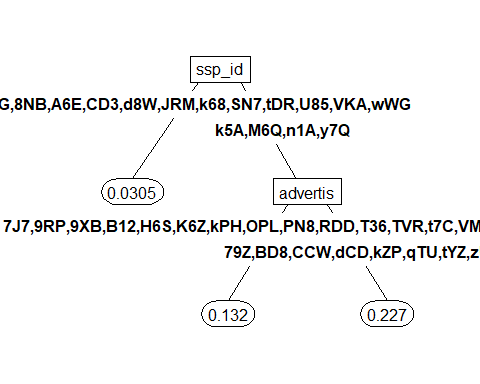
*logloss는 0.2560*

**[추가 문제] 문제 3-3 : 위의 만들어진 2 트리모델의 plot을 그려 보아라**

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.6.3

prp(model\_tree\_1, type = 5, digits = 3)



plot(model\_tree\_2) 로 두번째 트리를 그려볼 수 있다. 하지만 노드를 안나눴는데 그릴필요조차 없겠지? 따라서 여기서는 rpart가 더 좋겠다!

logloss\_tree <- LogLoss(pred\_tree\_1, test\_data$click)

## 랜덤포레스트

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.6.3

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

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

## The following object is masked from 'package:ggplot2':  
##   
## margin

**문제 4-1 : train과 test의 Y 변수가 숫자형(0, 1) 범주형으로 바꾸어 주어라**

train\_data$click <- factor(train\_data$click)  
test\_data$click <- factor(test\_data$click)

**문제 4-2 : 랜덤 포레스트는 train set과 test set의 범주레벨을 맞추지 않으면 오류가 난다.2개의 데이터 셋 레벨을 맞추어 주자**

*▶ 힌트 : 의외로 rbind를 사용하면 간단하게 할 수 있다*

dim(train\_data) # 19742개 행

## [1] 19742 23

data <- rbind(train\_data, test\_data)  
data %>% dim # 22294행

## [1] 22294 23

train\_data <- data[1:19742, ]  
test\_data <- data[19743:22294,]

**문제 4-3 : 모든 X 변수를 이용하여 Y 변수에 대해 랜덤 포레스트 모델을 적합 시켜보아라**

*▶ 모델 변수 이름 : model\_rf\_1* *▶ 시드 번호 : 1* *▶ 파라미터 : 기본값*

set.seed(1)  
model\_rf\_1 <- randomForest(as.factor(click) ~ ., data = train\_data)  
pred\_rf\_1 <- predict(model\_rf\_1, newdata = test\_data, type = 'prob')  
logloss\_rf <- LogLoss(pred\_rf\_1[, 2], ifelse(as.integer(test\_data$click) == 1, 0, 1))  
logloss\_rf

## [1] 0.2767536

# logloss 값은 0.2767

**문제 5 : 로지스틱회귀, 트리, 랜덤 포레스트에서 만들어진 모든 모델의 test에 대한 Logloss를 구하고 이를 비교하라**

logloss\_glm

## [1] 0.3105291

logloss\_tree

## [1] 0.2551373

logloss\_rf

## [1] 0.2767536

트리 성능이 제일 좋네!

## 챕터 2 – Cross Validation

library(caret)

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following objects are masked from 'package:MLmetrics':  
##   
## MAE, RMSE

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

**문제 1 : 아래와 같은 조건으로 랜덤포레스트 모델의 파라미터를 튜닝하고, 어떤 파라미터가 가장 좋은지 확인해라**

*5 Fold CV 방식 파라미터 튜닝* *파라미터 : mtry = 3, 4, 5* *척도 : logloss* *시드 번호 : 1*

set.seed(1)  
train\_data\_x <- train\_data %>% select(-click)  
train\_data\_y <- train\_data %>% select(click)  
  
n\_split = 5  
cv <- caret::createFolds(train\_data$click, k = n\_split)  
tune\_para <- expand.grid(mtry = 3:5)  
tune\_para$logloss <- NA  
  
rfmodel <- list(NULL)  
  
for (k in 1:NROW(tune\_para)) {  
 logloss\_result <- c()  
 for (i in 1:n\_split) {  
   
 idx <- cv[[i]]  
   
 train\_x <- train\_data\_x[-idx, ]  
 train\_y <- train\_data\_y[-idx, ]  
 val\_x <- train\_data\_x[idx, ]  
 val\_y <- train\_data\_y[idx, ]  
   
 set.seed(1)  
 rfmodel[[k]] <- randomForest(x = train\_x, y = train\_y, mtry = tune\_para[k, 'mtry'])  
 cv\_pred <- predict(rfmodel[[k]], newdata = cbind(val\_y, val\_x), type = 'prob')  
 cv\_logloss <- LogLoss(cv\_pred[, 2], ifelse(as.integer(val\_y) == 1, 0, 1))  
 logloss\_result <- c(logloss\_result, cv\_logloss)  
 }  
 tune\_para[k, 'logloss'] <- mean(logloss\_result)  
 print(k)  
}

## [1] 1  
## [1] 2  
## [1] 3

tune\_para # 3일때 최고다!

## mtry logloss  
## 1 3 0.2594486  
## 2 4 0.2628726  
## 3 5 0.2611207

동영님이 올려준 코드를 기반으로 짰습니다! expand.grid 갓갓!

**문제 2 : 최종적으로 그 파라미터를 사용한 모델로 test를 예측하였을 때 logloss를 구하여라. 그리고 이를 위의 랜덤포레스트 [model\_rf\_1] 와 결과를 비교하여라**

*▶ 모델 변수 이름 : model\_rf\_2*

model\_rf\_2 <- randomForest(as.factor(click) ~ ., data = train\_data, mtry = 3)  
pred\_rf\_2 <- predict(model\_rf\_2, newdata = test\_data, type = 'prob')  
logloss\_rf\_2 <- LogLoss(pred\_rf\_2[, 2], ifelse(as.integer(test\_data$click) == 1, 0, 1))  
  
logloss\_rf

## [1] 0.2767536

logloss\_rf\_2

## [1] 0.2771863

처음 랜포 모델이 더 좋았는데 너무 미미하다. 랜포 자체가 랜덤성을 가지기 때문에, cv를 통해 선택된 동일한 mtry를 갖는다고 하더라도,실제 각각 트리들의 구조들을 다를것이다.그래서 glm과 같이 동질적인 구조가 나오기는 어려울 것이고, 대신 랜포는 그자체로 n을 늘리면 늘 분산은 감소하기 때문에(과적합 걱정 없다!) 그냥 경험적인 mtry선택과 감당한수 있는 연산속도의 n을 선택해주면 되지 않는가 싶다.

## *끗*