경영데이터분석2 | 1차 과제 | 이동통신사 고객 이탈 데이터 분석

숙명여자대학교 경제학부 1815118학번 강수연

[데이터 분석 결과 요약 보고]

실제 이탈하지 않을 이동통신사 가입 고객을 적합하게 예측하여 마케팅하는 것을 기대손익 측면에서 핵심 문제로 정의하고 데이터 분석을 수행하였습니다. 따라서 churn 데이터의 이탈 여부를 예측하고자 하는 target 변수로 두고 0(No), 1(Yes)를 분류하였습니다.

다만, 데이터 라벨의 0과 1이 73 : 26의 비율로 클래스 불균형으로 인한 모델 성능 저하가 우려되었습니다. 즉, 이탈하지 않는 고객 데이터의 영향으로 상대적으로 적은 이탈하는 고객 데이터를 분류하는데 문제가 발생할 수 있는 상황이었습니다.

따라서 정밀도와 재현율을 조화평균하여 고려하는 F-Beta 점수를 평가지표로 선정하되 데이터 분석 목적에 따라 "이탈한다고 예측한 고객이 실제로 이탈할 비율", 정밀도에 더 높은 가중치 부여하였습니다. 참고로 Beta값이 1.0보다 작으면, Precision에 비중을 두고 평가지표를 산정합니다.

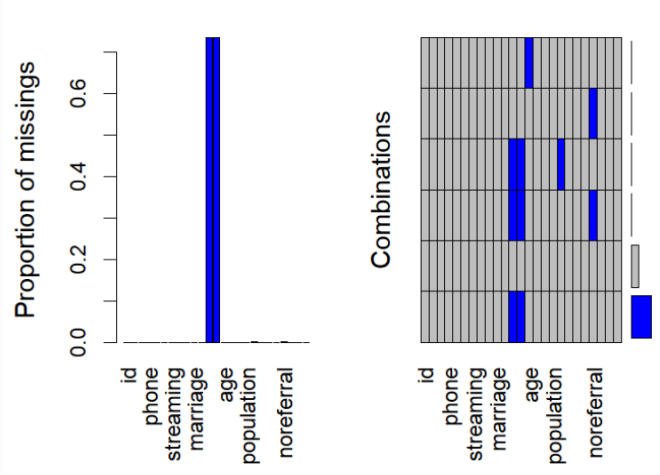
실제로 이탈하지 않을 고객을 이탈할 고객으로 분류하고 이동통신서비스 멤버십 혜택 관련 마케팅 프로모션을 수행하지 않는 오분류가 발생하여, 현업에서 잃는 기대수익(오차비용)을 줄이기 위해 거짓 양성(FP)을 최소화하는 머신러닝 분류모델 성능지표로 F-0.7 점수를 선정하였습니다.
신규 고객 데이터에 학습한 최적화 모델을 적용한 결과 모두 이탈하지 않을 고객으로 예측되었습니다.
모델 최적화 결과는 다음의 표와 같습니다.

	KNN1	KNN2	KNN2의 임계값 조정
하이퍼파라미터	k = 15, 17	k(n_neighbors) = 26 가까운 거리 내 이웃 점의 가중치를 부여하는 기준, 커널 함수 = optimal	k = 26 kernel = optimal
이탈 여부 class 변수의 0(No)와 1(Yes)을 분류하는 임계값 (threshold)	0.5	0.5	0.29 (0.05부터 0.95까지 0.01씩 증가하는 수열로 최적화되는 임계값 그리드 서치)
정확도, 코헨의 카파값	accuracy = 0.8622 Kappa = 0.5976	accuracy = 0.8665 Kappa = 0.622	accuracy = 0.8513 Kappa = 0.6367
F-Beta 점수 (Beta = 0.7)	`73.2%`	`75.11%`	`75.90%`

출처 : https://medium.com/@douglaspsteen/beyond-the-f-1-score-a-look-at-the-f-beta-score-3743ac2ef6e3

STEP1: 데이터프레임 준비 및 전처리

```
library(readr)
library(dplyr)
library(ggplot2)
library(VIM)
library(caret)
library(kknn)
library(MLmetrics)
churn <- read_csv("churn.csv")
```




```
# 1.1. 데이터 전처리: NA 처리
mice_plot <- aggr(churn, col = c("grey", "blue"), bars = TRUE, prop = TRUE, only.miss = FALSE)
  # [결측값 시각화]
mean(churn$age, na.rm = T)
churn$age <- ifelse(is.na(churn$age), 46.505, churn$age)
mean(churn$monthly_charge, na.rm = T)
churn$monthly_charge <- ifelse(is.na(churn$monthly_charge), 65.533,
churn$monthly_charge)
churn$noreferral <- ifelse(is.na(churn$noreferral), 0, churn$noreferral)

# 1.2. class 변수인 churn의 척도를 범주형 변경
table(churn$churn)
churn$churn <- as.factor(churn$churn)

# 1.3. churn_z 만들기
churn_z <- as.data.frame(scale(churn[14:25]))
churn_z$churn <- churn$churn


# STEP2: train과 test dataset을 7:3 분할 ####

churn_split <- createDataPartition(churn_z$churn, times = 1, p = 0.7, list = FALSE)
train <- churn_z[churn_split, ]
test <- churn_z[-churn_split, ]

table(churn_z$churn == '1') / nrow(churn_z)
  # 전체 데이터 class 비율 [0.7346301 : 0.2653699] False : True
table(train$churn == '1') / nrow(train)
  # 훈련용 데이터 class 비율 [0.7345366 : 0.2654634]
table(test$churn == '1') / nrow(test)
  # 검증용 데이터 class 비율 [0.7348485 : 0.2651515]

# STEP3: 최적의 K값 선정 및 KNN 모델 학습 ####
```

```
grid <- expand.grid(k = 3:20)
  # [근접한 이웃 후보군 3 ~ 20 (n_neighbors) 설정]
control <- trainControl(method = "repeatedcv", number = 10, repeats = 10)
  # [k-폴드 교차검증할 k = 10으로 설정]
set.seed(2022)
KNN1 <- train(churn ~ ., data = train, method = "knn", trControl = control, tuneGrid = grid)
KNN1$bestTune
  # [최적의 K값은 15, 17] 정확도 및 코헨의 카파(정확도-기대정확도 / 1-기대정확도)가 가장 높음


# STEP4: KNN 모델 성능 평가 ####

pred <- predict(KNN1, newdata = test, type = "prob")
pred$churn <- predict(KNN1, newdata = test, method = "class")
  # [default는 0.5 이상인 class로 분류]
confusionMatrix(test$churn, pred$churn, positive = "1")
  # [Accuracy : 0.8622]
  # [Kappa : 0.5976]
FBeta_Score(test$churn, pred$churn, positive = "1", beta = 0.7)
  # [실제 이탈하지 않을 고객을 적합하게 예측하여 마케팅하는 것을 기대손익 측면의 핵심 문제로 정의]
  # [0, 1이 73:26으로 클래스 불균형으로 인한 모델 성능 저하를 우려, F-Beta 점수를 평가지표로 선정]
  # [이탈한다고 예측한 고객이 실제로 이탈할 비율, 정밀도에 더 높은 가중치 부여]
  # [Beta값이 1.0보다 작으면, Precision에 비중을 두고 계산] 거짓 양성을 최소화
  # [F-Beta score = 73.2%]


# STEP5: KNN 모델 성능 개선 ####

# 5.1. 가까운 거리 내 이웃 data point에 가중치 부여
set.seed(2022)
KNN2 <- train.kknn(churn ~ ., data = train,
                   kmax = 30, distance = 2, # [L2 distance]
                   kernel = c("rectangular", "triangular", "Epanechnikov", "biweight",
"triweight", "cosine", "inversion", "Gaussian", "rank", "optimal"))
KNN2$best.parameters
  # [kernel = optimal]
  # [K = 26]

rm(KNN2)
KNN2 <- train.kknn(churn ~ ., data = train,
                   ks = 26, distance = 2, kernel = "optimal",
)
pred2 <- predict(KNN2, newdata = test, type = "prob")
pred2 <- data.frame(pred2)
colnames(pred2) <- c("0", "1")
```

```
pred2$churn <- predict(KNN2, newdata = test, method = "class")
  # [default는 0.5 이상인 class로 분류]
confusionMatrix(test$churn, pred2$churn, positive = "1")
  # [Accuracy : 0.8665]
  # [Kappa : 0.622]
FBeta_Score(test$churn, pred2$churn, positive = "1", beta = 0.7)
  # [F-Beta score = 75.11%]

# 5.2. class인 churn변수를 이진분류하는 임계값 0.5를 조정하여 최적의 임계값 그리드 서치
perform <- data.frame(threshold = seq(0.05, 0.95, 0.01),
                      TP = 0, TN = 0, FN = 0, FP = 0)
threshold <- seq(0.05, 0.95, 0.05)
  # [class 분류를 위한 0.05부터 0.95까지 0.05씩 증가하는 임계값 수열]
i = 0
  # [threshold별 performance measure값의 행 번호 생성]
test[, 14:133] <- 0
  # [test 데이터 셋에 빈 공간 벡터 생성]

for (c in threshold) {
  i <- i + 1
  test[, i + 13] <- ifelse(pred2$`1` >= c, "1", "0")
  test[, i + 13] <- as.factor(test[, i + 13])
    # [각 임계값별 분류한 class를 빈 벡터에 할당 및 범주형 척도 변경]
  perform[i, 2] <- c(sum((test[, i + 13] == "1") & (test$churn == "1"))) # [TP]
  perform[i, 3] <- c(sum((test[, i + 13] == "0") & (test$churn == "0"))) # [TN]
  perform[i, 4] <- c(sum((test[, i + 13] == "0") & (test$churn == "1"))) # [FN]
  perform[i, 5] <- c(sum((test[, i + 13] == "1") & (test$churn == "0"))) # [FP]
}
perform$accuracy <- (perform$TP + perform$TN)/2112
ggplot(data = perform, aes(x = threshold, y = accuracy)) + geom_line(col = "blue") +
geom_point(col = "blue", alpha = 0.3)
  # [정확도] (1*TP + 1*TN + 0* FN + 0*FP) / total
perform$precision <- (perform$TP)/(perform$TP + perform$FP)
  # [정밀도]
perform$recall <- (perform$TP)/(perform$TP + perform$FN)
  # [재현율]

perform$FBetaScore <- (1.49 * perform$precision * perform$recall)/
(0.49*perform$precision + perform$recall)
ggplot(data = perform, aes(x = threshold, y = FBetaScore)) + geom_line(col = "gold") +
geom_point(col = "gold", alpha = 0.3)
  # [F-Beta score] {(70%**2 + 1) * Precision * Recall}/(70%**2*Precision + Recall)
which.max(perform$FBetaScore)
  # [threshold = 0.29]
  # F-Beta score =  75.9%, Accuracy = 0.8513, Kappa = 0.6367
```

```
pred2$churn_fitting <- ifelse(pred2$`1` >= 0.29, "1", "0")
pred2$churn_fitting <- as.factor(pred2$churn_fitting)
confusionMatrix(test$churn, pred2$churn_fitting, positive = "1")
FBeta_Score(test$churn, pred2$churn_fitting, beta = 0.7, positive = "1")

# STEP6: 예측 ####

churn_new <- read.csv("churn predict.csv")
pred_new <- predict(KNN2, newdata = churn_new[, -1], method = "class")
pred_new <- data.frame(pred_new)
pred_new <- pred_new %>% rename(churn = pred_new)

pred_prob <- predict(KNN2, newdata = churn_new[, -1], type = "prob")
pred_new$`0` <- pred_prob[1]
pred_new$churn_fitting <- ifelse(pred_new$`1` >= 0.29, "1", "0")
  # [모두 class = 0(이탈 여부 No)인 이동통신사 서비스 가입고객으로 예측]
```