Chapter 6

메모라이제이션

이 장에서 다룰 내용

- 단일 변수 모델 구축하기
- 교차 유효성 변수 선택하기
- 기본 다항변수 모델 구축하기
- 의사결정나무, 최근접 이웃 모델 시작하기

1. 사용된 데이터

- KDD Cup 2009 : Customer Relationship Prediction
- 프랑스 텔레콤 회사 Orange의 대형 마케팅 데이터베이스를 활용하여
- 고객의 이동 (churn),
- 새로운 제품 또는 서비스 (appetency)를 구매,
- 추가적으로 판매하는 성향을 예측하여 판매 수익을 높이는 것(up-selling)
- 을 예측할 수 있는 기회를 제공합니다.

☝ 훈련데이터로 데이터를 부분 샘플링하여 디버깅 하는데 들이는 시간 줄이기

2. 단일 변수 모델 구축하기

- 범주형 변수 - 테이블로 설명
- 숫자형 변수 - 범주화 하여 단일 변수 모델 구축

2-1 범주형 단일 변수 모델링

- 변수 218의 레벨로 이탈 그래프 만들기
- 변수 218의 코드값으로 이탈율 그룹화

```
table218 <- table(
   Var218=dTrain[,'Var218'],     # Note: 1
   churn=dTrain[,outcome],   # Note: 2
   useNA='ifany')    # Note: 3
print(table218)
```

```
##        churn
## Var218    -1     1
##   cJvF 19245  1220
##   UYBR 17860  1618
##   <NA>   423   152
```

```
print(table218[,2]/(table218[,1]+table218[,2]))
```

```
##        cJvF        UYBR        <NA>
## 0.05961398 0.08306808 0.26434783
```

2-1 범주형 단일 변수 모델링

- 범주형 변수를 위한 단일 변수 모델 함수

```
mkPredC <- function(outCol,varCol,appCol) {      # Note: 1
    pPos <- sum(outCol==pos)/length(outCol)   # Note: 2
    naTab <- table(as.factor(outCol[is.na(varCol)]))
    pPosWna <- (naTab/sum(naTab))[pos]    # Note: 3
    vTab <- table(as.factor(outCol),varCol)
    pPosWv <- (vTab[pos,]+1.0e-3*pPos)/(colSums(vTab)+1.0e-3)
    pred <- pPosWv[appCol]    # Note: 5
    pred[is.na(appCol)] <- pPosWna    # Note: 6
    pred[is.na(pred)] <- pPos     # Note: 7
    pred      # Note: 8
}
```

2-1 범주형 단일 변수 모델링

- 단일 범주형 변수 모델을 모든 데이터셋에 적용

```
for(v in catVars) {
  pi <- paste('pred',v,sep='')
  dTrain[,pi] <- mkPredC(dTrain[,outcome],dTrain[,v],dTrain[,v])
  dCal[,pi] <- mkPredC(dTrain[,outcome],dTrain[,v],dCal[,v])
  dTest[,pi] <- mkPredC(dTrain[,outcome],dTrain[,v],dTest[,v])
}
```

2-1 범주형 단일 변수 모델링

- AUC로 범주형 변수 스코어링하기

```
calcAUC <- function(predcol,outcol) {
    perf <- performance(prediction(predcol,outcol==pos),'auc')
    as.numeric(perf@y.values)
 }

for(v in catVars) {
   pi <- paste('pred',v,sep='')
   aucTrain <- calcAUC(dTrain[,pi],dTrain[,outcome])
   if(aucTrain>=0.8) {
      aucCal <- calcAUC(dCal[,pi],dCal[,outcome])
      print(sprintf("%s, trainAUC: %4.3f calibrationAUC: %4.3f",
        pi,aucTrain,aucCal))
   }
 }
```

```
## [1] "predVar200, trainAUC: 0.830 calibrationAUC: 0.565"
## [1] "predVar202, trainAUC: 0.827 calibrationAUC: 0.525"
## [1] "predVar214, trainAUC: 0.830 calibrationAUC: 0.565"
## [1] "predVar217, trainAUC: 0.897 calibrationAUC: 0.553"
```

2-2 숫자형 단일 변수 모델링

- 숫자의 범위로 새로운 범주형 변수 만들기

```
mkPredN <- function(outCol,varCol,appCol) {
    cuts <- unique(as.numeric(quantile(varCol,
       probs=seq(0, 1, 0.1),na.rm=T)))
    varC <- cut(varCol,cuts)
    appC <- cut(appCol,cuts)
    mkPredC(outCol,varC,appC)
}
```

2-2 숫자형 단일 변수 모델링

- AUC로 숫자형 변수 스코어링하기

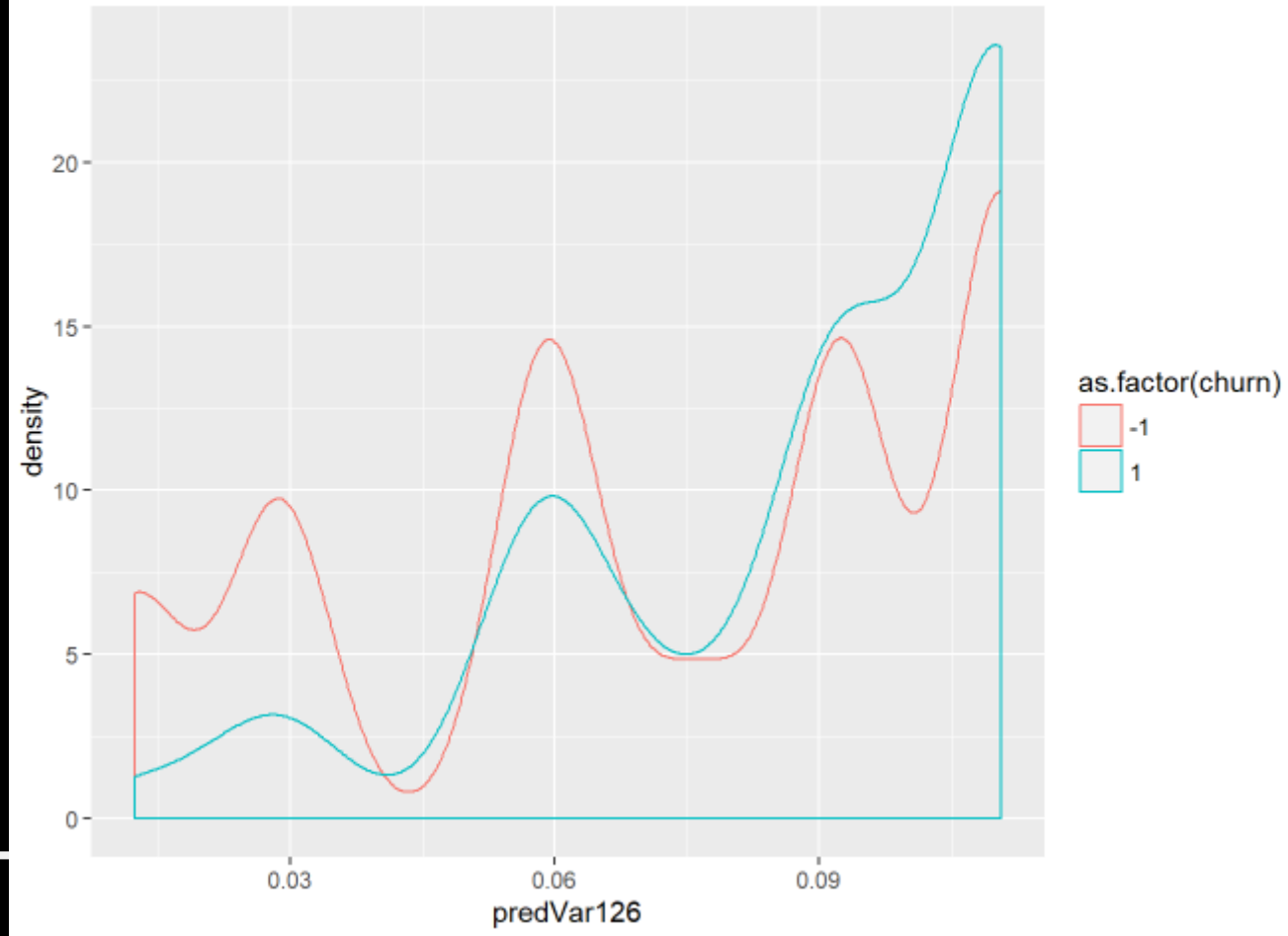
```
for(v in numericVars) {
   pi <- paste('pred',v,sep='')
   dTrain[,pi] <- mkPredN(dTrain[,outcome],dTrain[,v],dTrain[,v])
   dTest[,pi] <- mkPredN(dTrain[,outcome],dTrain[,v],dTest[,v])
   dCal[,pi] <- mkPredN(dTrain[,outcome],dTrain[,v],dCal[,v])
   aucTrain <- calcAUC(dTrain[,pi],dTrain[,outcome])
   if(aucTrain>=0.55) {
      aucCal <- calcAUC(dCal[,pi],dCal[,outcome])
      print(sprintf("%s, trainAUC: %4.3f calibrationAUC: %4.3f",
        pi,aucTrain,aucCal))
   }
 }
```

```
## [1] "predVar6, trainAUC: 0.557 calibrationAUC: 0.554"
## [1] "predVar7, trainAUC: 0.555 calibrationAUC: 0.565"
## [1] "predVar13, trainAUC: 0.568 calibrationAUC: 0.553"
## [1] "predVar73, trainAUC: 0.608 calibrationAUC: 0.616"
## [1] "predVar74, trainAUC: 0.574 calibrationAUC: 0.566"
## [1] "predVar81, trainAUC: 0.558 calibrationAUC: 0.542"
```

2-2 숫자형 단일 변수 모델링

- 변수 성능을 그래프로 표현하기

```
ggplot(data=dCal) +
   geom_density(aes(x=predVar126,color=as.factor(churn)))
```



2-3 교차 검증으로 과적합 정도 측정

- 반복 교차 검증 실험 수행

```
var <- 'Var217'
aucs <- rep(0,100)
for(rep in 1:length(aucs)) {    # Note: 1
   useForCalRep <- rbinom(n=dim(dTrainAll)[[1]],size=1,prob=0.1)>0      # Note: 2
   predRep <- mkPredC(dTrainAll[!useForCalRep,outcome],     # Note: 3
      dTrainAll[!useForCalRep,var],
      dTrainAll[useForCalRep,var])
   aucs[rep] <- calcAUC(predRep,dTrainAll[useForCalRep,outcome])    # Note: 4
 }
mean(aucs)
```

```
## [1] 0.5548469
```

```
## [1] 0.5556656
sd(aucs)
```

```
## [1] 0.01569641
```

2-4 다항변수를 이용한 모델 구축

- AIC-inspired 스코어에 따른 변수 선택

```
for(v in catVars) {     # Note: 2
  pi <- paste('pred',v,sep='')
  liCheck <- 2*((logLikelyhood(dCal[,outcome],dCal[,pi]) -
      baseRateCheck))
  if(liCheck>minStep) {
     print(sprintf("%s, calibrationScore: %g",
        pi,liCheck))
     selVars <- c(selVars,pi)
  }
}
```

```
## [1] "predVar194, calibrationScore: 5.25759"
## [1] "predVar201, calibrationScore: 5.25521"
## [1] "predVar204, calibrationScore: 5.37414"
## [1] "predVar205, calibrationScore: 24.2323"
## [1] "predVar206, calibrationScore: 34.4434"
## [1] "predVar210, calibrationScore: 10.6681"
## [1] "predVar212, calibrationScore: 6.23409"
## [1] "predVar218, calibrationScore: 13.2455"
## [1] "predVar221, calibrationScore: 12.4098"
## [1] "predVar225, calibrationScore: 22.9074"
## [1] "predVar226, calibrationScore: 6.68931"
## [1] "predVar228, calibrationScore: 15.9644"
## [1] "predVar229, calibrationScore: 24.4946"
```

```
for(v in numericVars) {     # Note: 3
  pi <- paste('pred',v,sep='')
  liCheck <- 2*((logLikelyhood(dCal[,outcome],dCal[,pi]) -
      baseRateCheck))
  if(liCheck>=minStep) {
     print(sprintf("%s, calibrationScore: %g",
        pi,liCheck))
     selVars <- c(selVars,pi)
  }
}
```

```
## [1] "predVar6, calibrationScore: 13.2431"
## [1] "predVar7, calibrationScore: 18.685"
## [1] "predVar13, calibrationScore: 10.0632"
## [1] "predVar28, calibrationScore: 11.3864"
## [1] "predVar65, calibrationScore: 9.96938"
## [1] "predVar72, calibrationScore: 12.5353"
## [1] "predVar73, calibrationScore: 48.2524"
## [1] "predVar74, calibrationScore: 19.6324"
## [1] "predVar81, calibrationScore: 8.8741"
## [1] "predVar113, calibrationScore: 23.136"
## [1] "predVar125, calibrationScore: 6.06029"
## [1] "predVar126, calibrationScore: 74.9556"
## [1] "predVar134, calibrationScore: 5.68144"
## [1] "predVar140, calibrationScore: 16.1816"
## [1] "predVar144, calibrationScore: 15.9858"
## [1] "predVar189, calibrationScore: 42.3059"
```

2-4 다항변수를 이용한 모델 구축

- 결정나무 모델에 적용하기1

```
library('rpart')
fV <- paste(outcome,'>0 ~ ',
   paste(c(catVars,numericVars),collapse=' + '),sep='')
tmodel <- rpart(fV,data=dTrain)
print(calcAUC(predict(tmodel,newdata=dTrain),dTrain[,outcome]))
```

```
## [1] 0.9241265
```

```
## [1] 0.9241265
print(calcAUC(predict(tmodel,newdata=dTest),dTest[,outcome]))
```

```
## [1] 0.5266172
```

```
## [1] 0.5266172
print(calcAUC(predict(tmodel,newdata=dCal),dCal[,outcome]))
```

```
## [1] 0.5126917
```

```
## [1] 0.5126917
```

```
tVars <- paste('pred',c(catVars,numericVars),sep='')
fV2 <- paste(outcome,'>0 ~ ',paste(tVars,collapse=' + '),sep='')
tmodel <- rpart(fV2,data=dTrain)
print(calcAUC(predict(tmodel,newdata=dTrain),dTrain[,outcome]))
```

```
## [1] 0.928669
```

```
## [1] 0.928669
print(calcAUC(predict(tmodel,newdata=dTest),dTest[,outcome]))
```

```
## [1] 0.5390648
```

```
## [1] 0.5390648
print(calcAUC(predict(tmodel,newdata=dCal),dCal[,outcome]))
```

```
## [1] 0.5384152
```

```
## [1] 0.5384152
```

2-4 다항변수를 이용한 모델 구축

- 결정나무 모델에 적용하기2

```
tmodel <- rpart(fV2,data=dTrain,
   control=rpart.control(cp=0.001,minsplit=1000,
      minbucket=1000,maxdepth=5)
 )
print(calcAUC(predict(tmodel,newdata=dTrain),dTrain[,outcome]))
```

```
## [1] 0.9421195
```

```
## [1] 0.9421195
print(calcAUC(predict(tmodel,newdata=dTest),dTest[,outcome]))
```

```
## [1] 0.5794633
```

```
## [1] 0.5794633
print(calcAUC(predict(tmodel,newdata=dCal),dCal[,outcome]))
```

```
## [1] 0.547967
```

```
f <- paste(outcome,'>0 ~ ',paste(selVars,collapse=' + '),sep='')
tmodel <- rpart(f,data=dTrain,
   control=rpart.control(cp=0.001,minsplit=1000,
      minbucket=1000,maxdepth=5)
 )
print(calcAUC(predict(tmodel,newdata=dTrain),dTrain[,outcome]))
```

```
## [1] 0.6906852
```

```
## [1] 0.6906852
print(calcAUC(predict(tmodel,newdata=dTest),dTest[,outcome]))
```

```
## [1] 0.6843595
```

```
## [1] 0.6843595
print(calcAUC(predict(tmodel,newdata=dCal),dCal[,outcome]))
```

```
## [1] 0.6669301
```

2-4 다항변수를 이용한 모델 구축

- 결정나무 출력하기

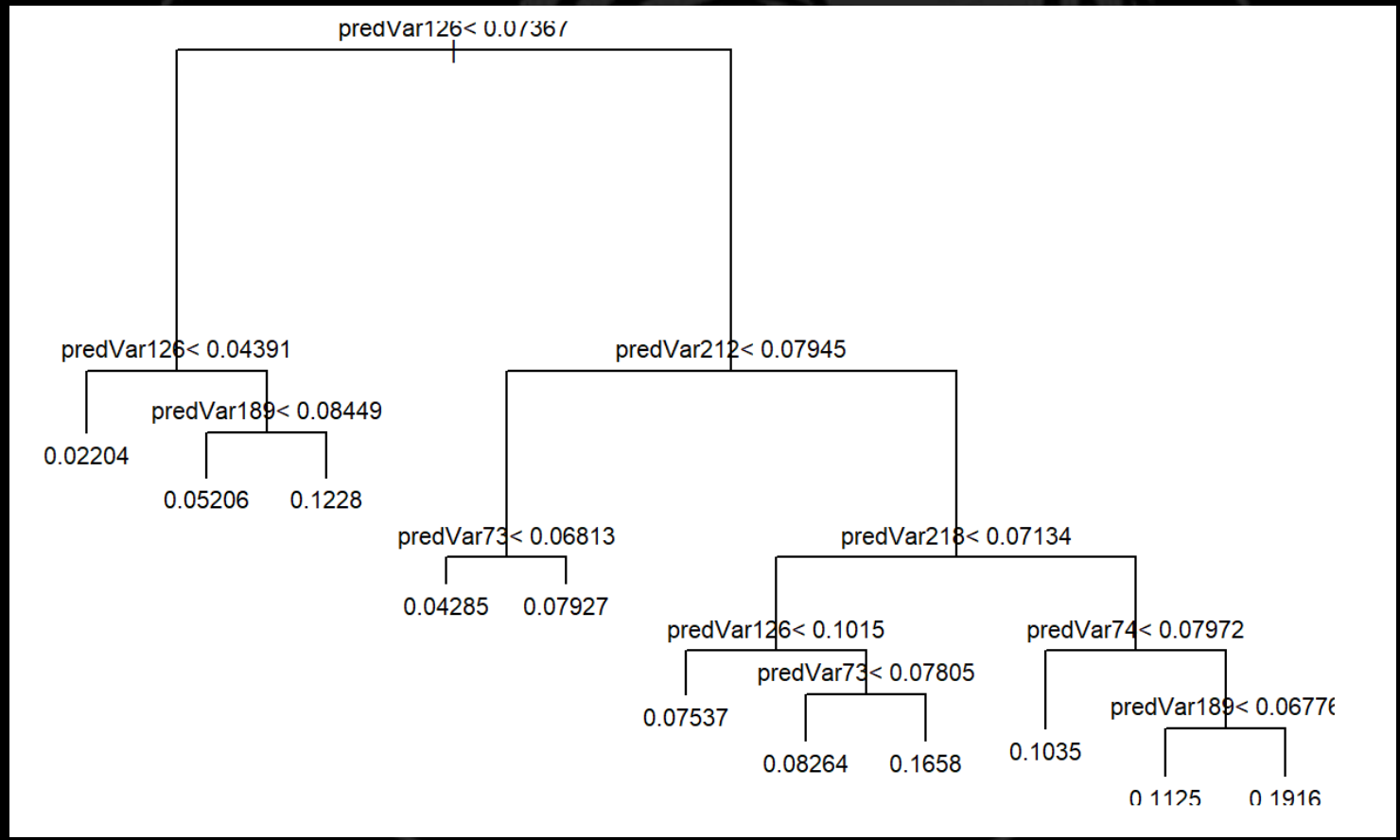
```
print(tmodel)
```

```
## n= 40518
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 40518 2769.3550 0.07379436
##    2) predVar126< 0.07366888 18188  726.4097 0.04167583
##      4) predVar126< 0.04391312 8804  189.7251 0.02203544 *
##      5) predVar126>=0.04391312 9384  530.1023 0.06010230
##       10) predVar189< 0.08449448 8317  410.4571 0.05206204 *
##       11) predVar189>=0.08449448 1067  114.9166 0.12277410 *
##    3) predVar126>=0.07366888 22330 2008.9000 0.09995522
##      6) predVar212< 0.07944508 8386  484.2499 0.06153112
##       12) predVar73< 0.06813291 4084  167.5012 0.04285015 *
##       13) predVar73>=0.06813291 4302  313.9705 0.07926546 *
##      7) predVar212>=0.07944508 13944 1504.8230 0.12306370
##       14) predVar218< 0.07134103 6728  580.7390 0.09542212
##         28) predVar126< 0.1015407 3901  271.8426 0.07536529 *
##         29) predVar126>=0.1015407 2827  305.1617 0.12309870
##           58) predVar73< 0.07804522 1452  110.0826 0.08264463 *
##           59) predVar73>=0.07804522 1375  190.1935 0.16581820 *
##       15) predVar218>=0.07134103 7216  914.1502 0.14883590
##         30) predVar74< 0.0797246 2579  239.3579 0.10352850 *
##         31) predVar74>=0.0797246 4637  666.5538 0.17403490
##           62) predVar189< 0.06775545 1031  102.9486 0.11251210 *
##           63) predVar189>=0.06775545 3606  558.5871 0.19162510 *
```

2-4 다항변수를 이용한 모델 구축

- 결정나무 그리기

```
par(cex=0.7)
plot(tmodel)
text(tmodel)
```



2-4 다항변수를 이용한 모델 구축

- K 최근접 이웃 메서드(KNN; K-nearest neighbot)

```
library('class')
nK <- 200
knnTrain <- dTrain[,selVars]     # Note: 1
knnCl <- dTrain[,outcome]==pos   # Note: 2
knnPred <- function(df) {    # Note: 3
    knnDecision <- knn(knnTrain,df,knnCl,k=nK,prob=T)
    ifelse(knnDecision==TRUE,    # Note: 4
       attributes(knnDecision)$prob,
       1-(attributes(knnDecision)$prob))
}
print(calcAUC(knnPred(dTrain[,selVars]),dTrain[,outcome]))
```

```
## [1] 0.7437617
```

```
## [1] 0.7443927
print(calcAUC(knnPred(dCal[,selVars]),dCal[,outcome]))
```

```
## [1] 0.7131476
```

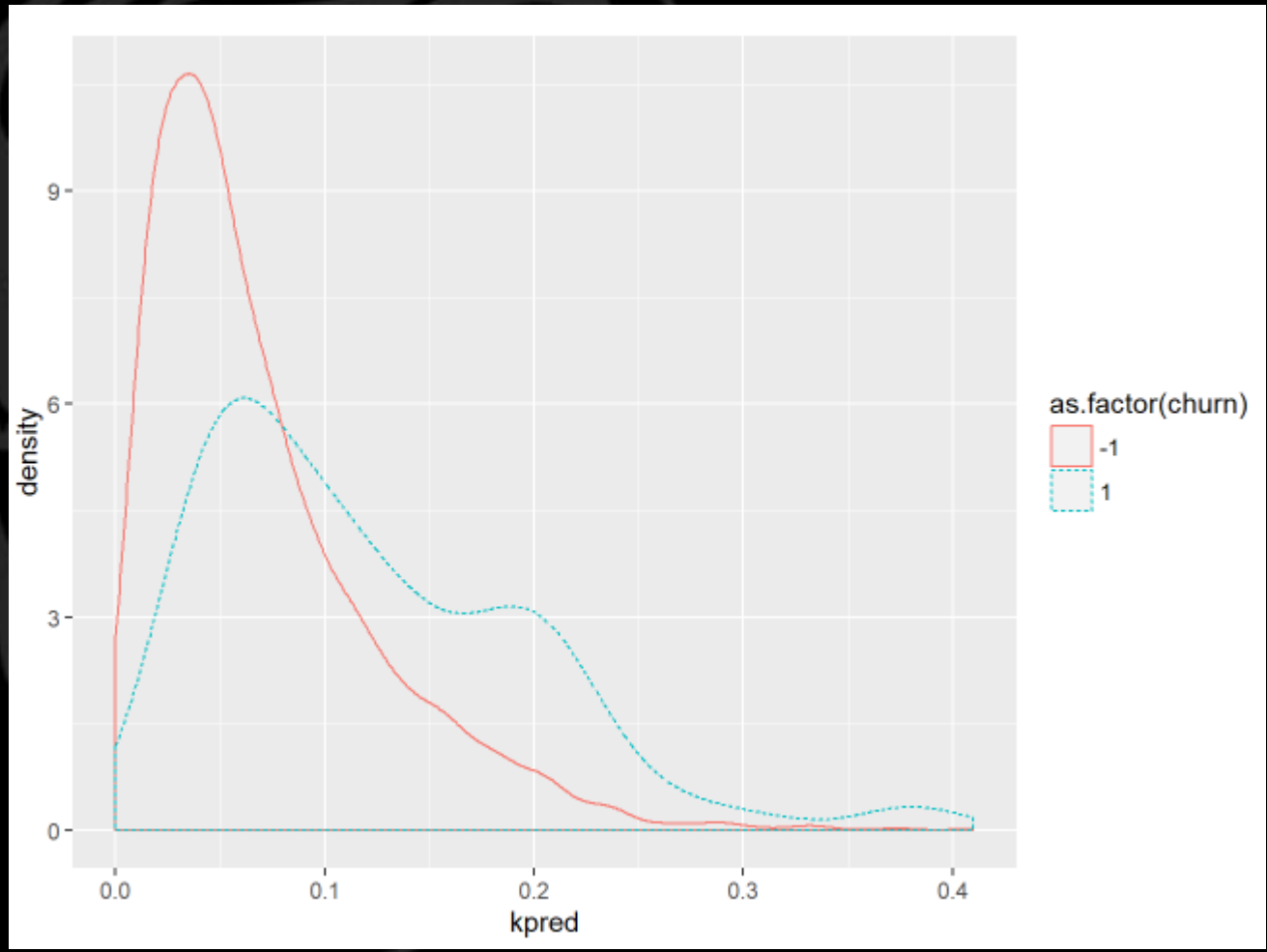
```
## [1] 0.7119394
print(calcAUC(knnPred(dTest[,selVars]),dTest[,outcome]))
```

```
## [1] 0.7179175
```

2-4 다항변수를 이용한 모델 구축

- 밀도그래프를 이용한 KNN의 성능

```
dCal$kpred <- knnPred(dCal[,selVars])
ggplot(data=dCal) +
   geom_density(aes(x=kpred,
      color=as.factor(churn),linetype=as.factor(churn)))
```



2-4 다항변수를 이용한 모델 구축

- 검증 데이터를 사용한 KNN의 ROC 곡선

```
plotROC <- function(predcol,outcol) {
   perf <- performance(prediction(predcol,outcol==pos),'tpr','fpr')
   pf <- data.frame(
      FalsePositiveRate=perf@x.values[[1]],
      TruePositiveRate=perf@y.values[[1]])
   ggplot() +
      geom_line(data=pf,aes(x=FalsePositiveRate,y=TruePositiveRate)) +
      geom_line(aes(x=c(0,1),y=c(0,1)))
}
print(plotROC(knnPred(dTest[,selVars]),dTest[,outcome]))
```