11장 titanic 생존자 예측

jihyun

2019 8 28

타이타닉 데이터를 사용한 기계학습 실습

01 타이타닉 데이터 받기

타이타닉 데이터는 http://biostat.mc.vanderbilt.edu/wiki/Main/DataSets 에서 titanic3.csv 를 다운받으면 된다.

02 데이터 불러오기

read.csv() 를 통해 titanic3.csv를 불러올 수 있다. 불러온 뒤에 사용할 column만 분리했다.

```
titanic = read.csv("titanic3.csv")
titanic <- titanic[, !names(titanic) %in% c("home.dest", "boat", "body")]
str(titanic)
```

```
## 'data.frame':    1309 obs. of  11 variables:
##  $ pclass  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ survived: int  1 1 0 0 0 1 1 0 1 0 ...
##  $ name    : Factor w/ 1307 levels "Abbing, Mr. Anthony",..: 22 24 25 26 27 31 46 47 51 55 ...
##  $ sex     : Factor w/ 2 levels "female","male": 1 2 1 2 1 2 1 2 1 2 ...
##  $ age     : num  29 0.92 2 30 25 48 63 39 53 71 ...
##  $ sibsp   : int  0 1 1 1 1 0 1 0 2 0 ...
##  $ parch   : int  0 2 2 2 2 0 0 0 0 0 ...
##  $ ticket  : Factor w/ 929 levels "110152","110413",..: 188 50 50 50 50 125 93 16 77 826 ...
##  $ fare    : num  211 152 152 152 152 ...
##  $ cabin   : Factor w/ 187 levels "","A10","A11",..: 45 81 81 81 81 151 147 17 63 1 ...
##  $ embarked: Factor w/ 4 levels "","C","Q","S": 4 4 4 4 4 4 4 4 4 2 ...
```

변수설명 pclass : 1,2,3 등석 정보 survived : 생존여부(survived, dead) name : 이름 sex : 성별 age : 나이 sibsp : 함께 탑승한 형제, 또는 배우자의 수 parch : 함께 탑승한 부모, 또는 자녀의 수 ticket : 티켓 번호 fare : 티켓 요금 cabin : 선실 번호 embarcked : 탑승한곳 C: cherbourg, Q : Queenstown, S : Southampton

03 데이터 타입 변경

str(titanic)의 결과를 보면 pclass가 int로 저장되어있다. 그러나 pclass는숫자로 취급하기 보다는 범주형 변수인 factor로 취급하는것이 낫다. 또한 survived또한 factor로 취급하는 것이 좋다. int로 되어있으면 분류 알고리즘이 아니라 회귀분석 알고리즘이 실행되게 된다.

마지막으로, name, ticket, cabin은 factor가 아니라 단순 문자열로 나타내는것이 더 옳다.

```
titanic$pclass <- as.factor(titanic$pclass)
titanic$name <- as.character(titanic$name)
titanic$ticket <- as.character(titanic$ticket)
titanic$cabin <- as.character(titanic$cabin)
titanic$survived <- factor(titanic$survived, levels=c(0,1), labels=c("dead","survived"))
str(titanic)
```

```
## 'data.frame':    1309 obs. of  11 variables:
##  $ pclass  : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ survived: Factor w/ 2 levels "dead","survived": 2 2 1 1 1 2 2 1 2 1 ...
##  $ name    : chr  "Allen, Miss. Elisabeth Walton" "Allison, Master. Hudson Trevor" "Allison, Miss. H
##  $ sex     : Factor w/ 2 levels "female","male": 1 2 1 2 1 2 1 2 1 2 ...
##  $ age     : num  29 0.92 2 30 25 48 63 39 53 71 ...
##  $ sibsp   : int  0 1 1 1 1 0 1 0 2 0 ...
##  $ parch   : int  0 2 2 2 2 0 0 0 0 0 ...
##  $ ticket  : chr  "24160" "113781" "113781" "113781" ...
##  $ fare    : num  211 152 152 152 152 ...
##  $ cabin   : chr  "B5" "C22 C26" "C22 C26" "C22 C26" ...
##  $ embarked: Factor w/ 4 levels "","C","Q","S": 4 4 4 4 4 4 4 4 4 2 ...
```

str을 자세히 보면 embarked에 "" 값이있는 것을 볼 수 있다. table을 통해 보면

```
table(titanic$embarked)
```

```
##
##       C   Q   S
##   2 270 123 914
```

"" 인 값이 2개가 있는 것을 알 수 있고, 이것은 NA를 의미한다. ""를 NA로 수정해 보자.

```
levels(titanic$embarked)[1] <- NA
table(titanic$embarked, useNA = "always")# table    NA          na      always      .
```

```
##
##    C    Q    S <NA>
##  270  123  914    2
```

cabin컬럼에도 빈 문자열이 저장되어있다. 이 값 역시 NA로 바꿔보자.

```
titanic$cabin <- ifelse(titanic$cabin == "", NA, titanic$cabin)
str(titanic)
```

```
## 'data.frame':    1309 obs. of  11 variables:
##  $ pclass  : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ survived: Factor w/ 2 levels "dead","survived": 2 2 1 1 1 2 2 1 2 1 ...
##  $ name    : chr  "Allen, Miss. Elisabeth Walton" "Allison, Master. Hudson Trevor" "Allison, Miss. H
##  $ sex     : Factor w/ 2 levels "female","male": 1 2 1 2 1 2 1 2 1 2 ...
##  $ age     : num  29 0.92 2 30 25 48 63 39 53 71 ...
##  $ sibsp   : int  0 1 1 1 1 0 1 0 2 0 ...
##  $ parch   : int  0 2 2 2 2 0 0 0 0 0 ...
##  $ ticket  : chr  "24160" "113781" "113781" "113781" ...
##  $ fare    : num  211 152 152 152 152 ...
##  $ cabin   : chr  "B5" "C22 C26" "C22 C26" "C22 C26" ...
##  $ embarked: Factor w/ 3 levels "C","Q","S": 3 3 3 3 3 3 3 3 3 1 ...
```

(Factor 와 character에 na해주는 방법에 차이가 있다.)

04. 테스트 데이터의 분리.

test데이터와 train데이터를 분리해 보자. createDataPartion()을 통해 데이터를 나누면, test, train데이터 간의생존자 수와 사망자 수의 비율을 고려하여 데이터를 분리 할 수 있다.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
set.seed(137)
test_idx <- createDataPartition(titanic$survived, p=0.1)$Resample1
titanic.test <- titanic[test_idx,]
titanic.train <- titanic[-test_idx,]
```

데이터 분리가 끝나면 이후 단계에서 사용이 편리하도록 저장해 놓는다.

아래 코드는 titanic, titanic.test, titanic.train 데이터를 titanic.RData라는 파일에 저장한다.

```
save(titanic, titanic.test, titanic.train, file = "titanic.RData")
```

05. 교차 검증 준비

caret 패키지의 creatFolds()를 이용해 데이터를 분리해 보자.

```
#(createFolds(titanic.train$survived, k=10))
```

수행 결과, 10개의 fold가 만들어 져서 리스트에 저장되어 있는 것을 알 수 있다.

```
create_ten_fold_cv <- function(){
  set.seed(137)
  lapply(createFolds(titanic.train$survived, k=10), function(idx){
    return(list(train=titanic.train[-idx, ],
                validation = titanic.train[idx, ]))
  })
}
```

이 함수는 Fold01, Fold02 ..를 포함하는 리스트를 반환하며, 각 폴드에는 train, validatoin이라는 이름에 훈련 데이터와 검증 데이터가 저장된다.

06. 데이터 탐색

모델을 작성하기 전에 데이터가 어떤 모습을 하고 있는지 살펴보면 모델을 세울 방법에 관한 많은 힌트를 얻을 수 있다. Hmisc패키지에는 summary에 포뮬러를 지정해 데이터의 요약 정보를 얻을 수 있는 기능이 있다.

```
#install.packages("Hmisc")
library(Hmisc)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
data <- create_ten_fold_cv()[[1]]$train
#reverse       survived
summary(survived ~ pclass + sex + age + sibsp + parch + fare + embarked, data = data, method = "reverse
```

```
##
##
## Descriptive Statistics by survived
##
## +------------+----+----------------------+----------------------+
## |            |N   |dead                  |survived              |
## |            |    |(N=655)               |(N=405)               |
## +------------+----+----------------------+----------------------+
## |pclass : 1  |1060|       15%  ( 99)     |       40%  (163)     |
## +------------+----+----------------------+----------------------+
## |    2       |    |       20%  (131)     |       23%  ( 93)     |
## +------------+----+----------------------+----------------------+
## |    3       |    |       65%  (425)     |       37%  (149)     |
## +------------+----+----------------------+----------------------+
## |sex : male  |1060|       85%  (557)     |       34%  (137)     |
## +------------+----+----------------------+----------------------+
## |age         | 843|   21.00/28.00/39.25  |   19.00/27.00/36.00  |
## +------------+----+----------------------+----------------------+
## |sibsp : 0   |1060|       72%  (471)     |       62%  (253)     |
## +------------+----+----------------------+----------------------+
## |    1       |    |       20%  (128)     |       32%  (131)     |
## +------------+----+----------------------+----------------------+
## |    2       |    |        3%  ( 20)     |        3%  ( 13)     |
## +------------+----+----------------------+----------------------+
## |    3       |    |        2%  ( 10)     |        1%  (  5)     |
## +------------+----+----------------------+----------------------+
## |    4       |    |        2%  ( 15)     |        1%  (  3)     |
## +------------+----+----------------------+----------------------+
## |    5       |    |        1%  (  5)     |        0%  (  0)     |
## +------------+----+----------------------+----------------------+
## |    8       |    |        1%  (  6)     |        0%  (  0)     |
## +------------+----+----------------------+----------------------+
```

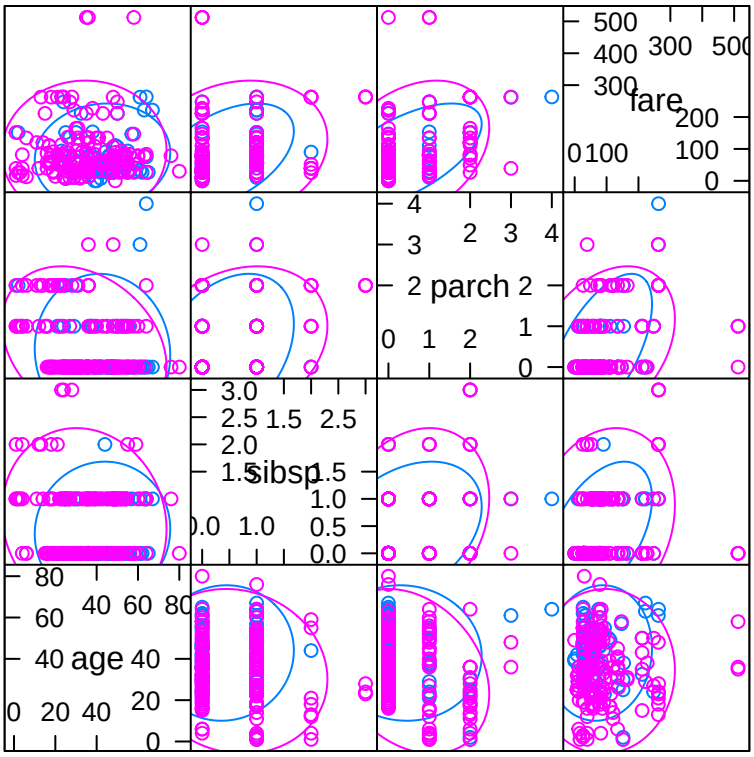
```
## |parch : 0   |1060|       83%  (544)      |       69%  (278)      |
## +------------+----+-----------------------+-----------------------+
## |    1       |    |        8%  ( 55)      |       18%  ( 73)      |
## +------------+----+-----------------------+-----------------------+
## |    2       |    |        7%  ( 44)      |       12%  ( 49)      |
## +------------+----+-----------------------+-----------------------+
## |    3       |    |        0%  (  3)      |        1%  (  4)      |
## +------------+----+-----------------------+-----------------------+
## |    4       |    |        1%  (  5)      |        0%  (  0)      |
## +------------+----+-----------------------+-----------------------+
## |    5       |    |        0%  (  3)      |        0%  (  1)      |
## +------------+----+-----------------------+-----------------------+
## |    6       |    |        0%  (  1)      |        0%  (  0)      |
## +------------+----+-----------------------+-----------------------+
## |fare        |1060| 7.8646/10.5000/26.0000|11.1333/26.0000/56.9292|
## +------------+----+-----------------------+-----------------------+
## |embarked : C|1058|       14%  ( 94)      |       31%  (125)      |
## +------------+----+-----------------------+-----------------------+
## |    Q       |    |        9%  ( 62)      |        9%  ( 35)      |
## +------------+----+-----------------------+-----------------------+
## |    S       |    |       76%  (499)      |       60%  (243)      |
## +------------+----+-----------------------+-----------------------+
```

이번에는 caret::featurePlot()을 사용해 데이터를 시각해 보자. featurePlot()은 NA가 하나라도 있으면 차트가 제대로 그려지지 않으므로, NA 부터 제거해야 한다. 이때 complete.cases()를 사용하여 na가 있는지 쉽게 볼 수 있다.

```
#install.packages("ellipse")
library(ellipse)
```

```
##
## Attaching package: 'ellipse'

## The following object is masked from 'package:graphics':
##
##     pairs
```

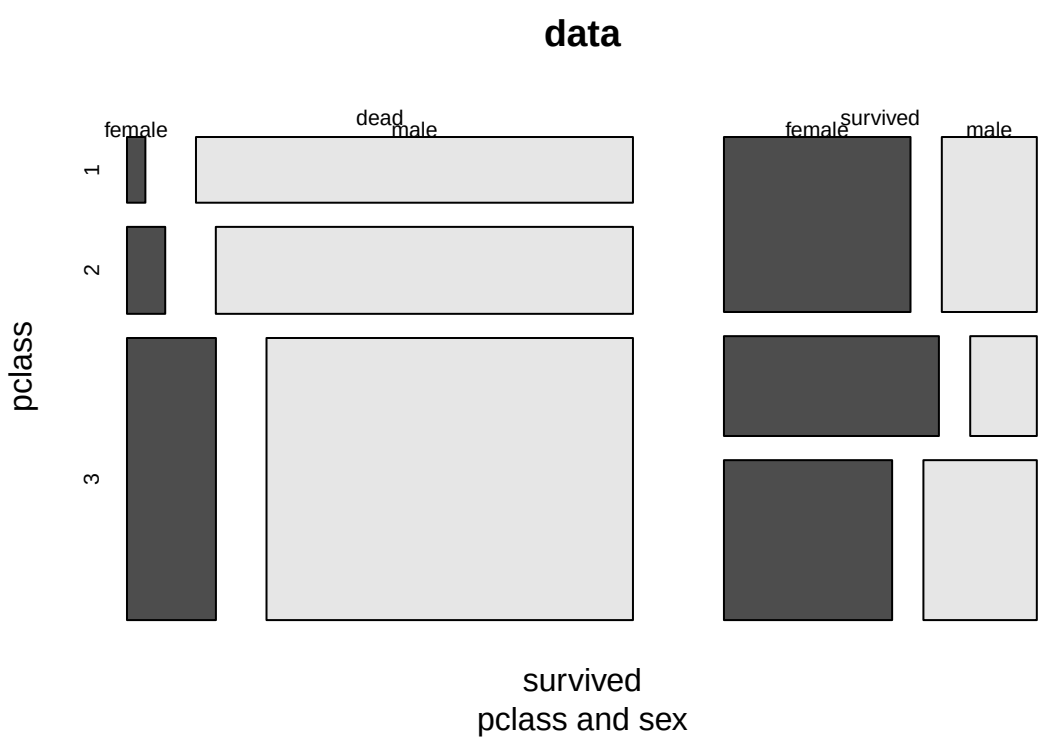
```
data <- create_ten_fold_cv()[[1]]$train
data.complete <- data[complete.cases(data),]# na         .

featurePlot(
  data.complete[
    ,sapply(names(data.complete),
            function(n){is.numeric(data.complete[,n])})],# numeric
  data.complete[,c("survived")],
  "ellipse"
)
```



Scatter Plot Matrix

팩터 타입의 차트에는 모자이크 플롯을 사용할 수 있다.

```
mosaicplot(survived ~pclass + sex, data = data, color = TRUE,
           main <- "pclass and sex")
```



분활표를 사용 할 수도 있다. 아래 분활표는 생존자 수와 관계없이, 성별과 class를 나타낸 분활표이다.

```
xtabs(~ sex + pclass, data = data)
```

```
##         pclass
## sex        1   2   3
##   female 112  82 172
##   male   150 142 402
```

아래 분활표는 생존 자 중에서 성별과 class를 나타낸 분활표이다.

```
xtabs(survived == "survived" ~ sex + pclass, data = data)# survived == "survived"
```

```
##         pclass
## sex        1   2   3
##   female 108  71  89
##   male    55  22  60
```

두결과를 조합해서 생존율을 구할 수 있다.

```
xtabs(survived == "survived" ~ sex+ pclass, data=data) / xtabs( ~sex + pclass, data = data)
```

```
##         pclass
## sex             1         2         3
##   female 0.9642857 0.8658537 0.5174419
##   male   0.3666667 0.1549296 0.1492537
```

07. 평가 메트릭

탑승객 생존 여부 예측 모델의 성능은 정확도(accuracy)로 하기로 한다. 정확도는 예측값이 true 이든 false이든 정확히 예측한 값의 비율을 뜻한다.

08. 의사결정 나무 모델

의사결정나무 모델은 다양한 변수의 상호 작용을 잘 표현해 준다. 또 타이타닉 데이터에는 na값이aksgdmsep, rpart는 이를 대리변수로 처리해준다. 대리변수란 노드에서 가지치기를 할 때 사용된 변수를 대신할 수있는 다른 변수를 말한다. 예를들어 age변수로 가지치기를 해야하는데 age변수가 na값이라면 height를 통해 age를 유추해서 가지치기를 하는 방식이다.

의사결정 나무 모델에 적합하지 않은 name(이름), ticket(티켓번호), cabin(방 번호)를 제외하고 모델을 만들어 보자.

```
library(rpart)
m <- rpart (
  survived ~pclass + sex + age + sibsp + parch + fare + embarked,
  data = titanic.train)

p <- predict(m, newdata = titanic.test, type = "class")
head(p)
```

```
##        1       12       13       40       41       42
## survived survived survived     dead     dead survived
## Levels: dead survived
```

09. rpart의 교차검증

교차 테스트 데이터에 대한 예측값을 구해보자.

```
library(rpart)
library(foreach)
folds <- create_ten_fold_cv()

rpart_result <- foreach(f=folds) %do% {
  model_rpart <- rpart(
    survived ~ pclass + sex + age+ sibsp + parch + fare + embarked,
    data = f$train)
  predicted <- predict(model_rpart, newdata = f$validation,
                       type = "class")
  return(list(actual = f$validation$survived, predicted = predicted))
}
```

rpart_result에는 각 fold에 대한 예측값과 실제값이 들어가게 된다.

10. 정확도 평가

rpart_result가 리스트인 것을 감안하여 정확도를 계산하는 함수를 만들어 보자.(이 함수는 다른 모델의 결과값에도 적용 될 수 있다.)

```
evaluation <- function(lst){
  accuracy <- sapply(lst, function(one_result){
    return(sum(one_result$predicted == one_result$actual)/NROW(one_result$actual))
  })

  print(sprintf("MEAN +/- SD : %.3f +/- %.3f",
                mean(accuracy), sd(accuracy)))
  return(accuracy)
}

(rpart_accuracy <- evaluation(rpart_result))
```

```
## [1] "MEAN +/- SD : 0.801 +/- 0.029"
```

```
##  [1] 0.7881356 0.8050847 0.7966102 0.8547009 0.8305085 0.8305085 0.7796610
##  [8] 0.7796610 0.7796610 0.7606838
```

출력결과 모델의 성능은 80.1%로 나타났다.

11. 모델 향상시키기 - 조건부 추론 나무

조건부 추론 나무를 이용하여 정확도를 높혀보자.

```
library(party)
```

```
## Loading required package: grid
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
ctree_result <- foreach(f=folds) %do%{
  model_ctree <- ctree(
    survived ~ pclass + sex + age + sibsp + parch + fare + embarked,
    data = f$train)
  predicted <- predict(model_ctree, newdata=f$validation,
                       type = "response")
  return(list(actual = f$validation$survived, predicted = predicted))
}

(ctree_accuracy <- evaluation(ctree_result))
```

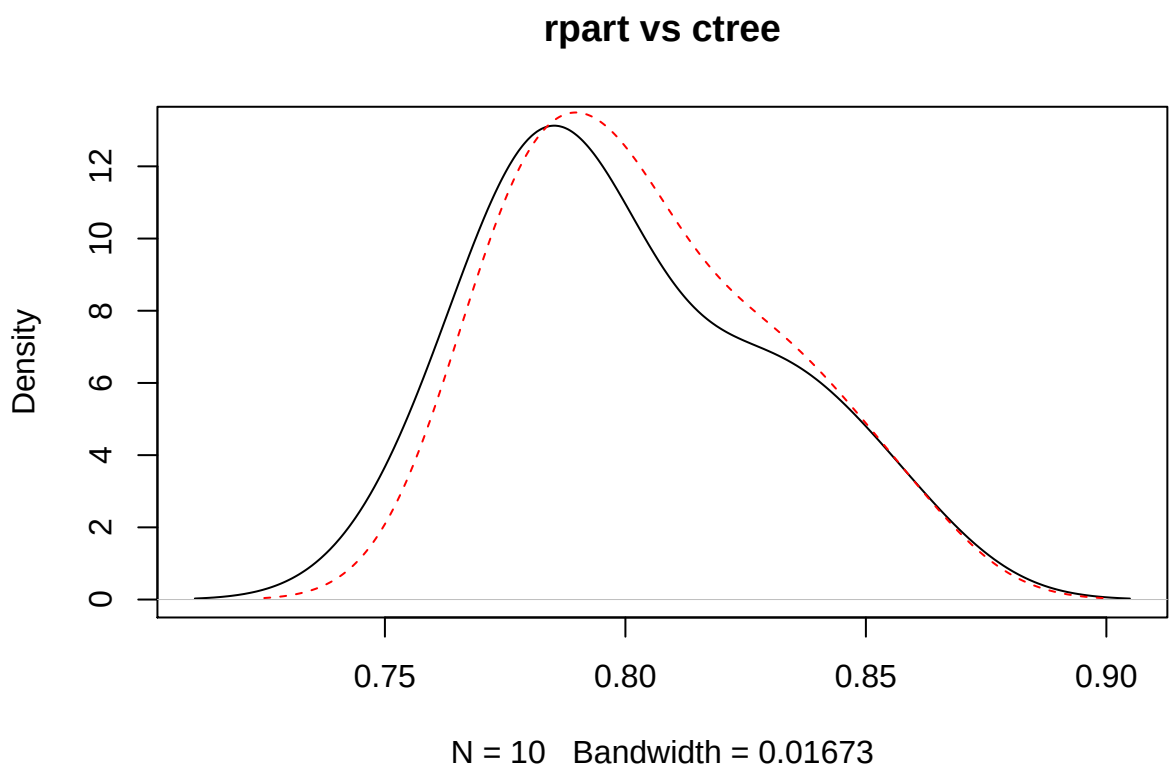
```
## [1] "MEAN +/- SD : 0.804 +/- 0.027"
```

```
##  [1] 0.7881356 0.8050847 0.7966102 0.8547009 0.8305085 0.8305085 0.8050847
##  [8] 0.7711864 0.7796610 0.7777778
```

출력결과 조건부 추론 나무의 성능은 80.4%로 의사결정나무 모델보다 약간 정확도가 높은 것을 알 수 있다.

또는 다음과 같이 밀도그림을 그려 정확도의 분포를 볼 수 있다.

```
plot(density(rpart_accuracy), main = "rpart vs ctree")+lines(density(ctree_accuracy), col ="red", lty="
```




```
## integer(0)
```

12. 모델 향상시키기 - 가족id만들기

가족의 생존여부에 따라서 각 개인의 생존여부에 영향이 갈 수도있기 때문에, 가족 변수를 만들어 보자. 티켓 id가 같으면 같은 가족일 것이라고 가정하고 가족 id를 많들어 보자.

그러나 훈련데이터와 검증데이터에 가족구성원이 나눠져 있으면 제대로 예측이 되지 않고, 타이타닉 데이터의 수가 작기 때문에, 훈련데이터와 검증 데이터 모두를 이용하여 가족id를 만들자.

먼저는 ticket이 가족을 찾는 데 얼마나 유용한지 보자.

아래는 titanic.train데이터를 ticket아이디에 따라 정렬해 표시한 것이다.

```
head(titanic.train[order(titanic.train$ticket),
               c("ticket","parch", "name","cabin","embarked")],10)
```

```
##     ticket parch                                                 name
## 68  110152     0                                 Cherry, Miss. Gladys
## 196 110152     0                                Maioni, Miss. Roberta
## 246 110152     0 Rothes, the Countess. of (Lucy Noel Martha Dyer-Edwards)
## 290 110413     2                                  Taussig, Miss. Ruth
## 291 110413     1                                    Taussig, Mr. Emil
## 75  110465     0                           Clifford, Mr. George Quincy
## 233 110465     0                         Porter, Mr. Walter Chamberlain
## 195 110469     0                              Maguire, Mr. John Edward
## 35  110489     0                              Borebank, Mr. John James
## 30  110564     0           Bjornstrom-Steffansson, Mr. Mauritz Hakan
##     cabin embarked
## 68    B77        S
## 196   B79        S
## 246   B77        S
## 290   E68        S
## 291   E67        S
## 75    A14        S
## 233  C110        S
## 195  C106        S
## 35    D22        S
## 30    C52        S
```

위의 결과를 보면 Taussig성을 가진 3명의 사람들이 비슷한 선실, 같은 탑승위치에 탄 것으로 보아, 가족임을 알 수 있다. 이러한 방식을 ticket을 통해 가족을 유추 해 낼 수 있는것을 알 수 있다.

13. 교차검증 데이터 준비하기(각 탑승객의 생존확률 구하기)

위에서 한 것 처럼 교차검증을 위한 데이터를 준비한다. 하지만 가족관계를 위해서 처음에는 validation과 train데이터를 합쳐서 가족관계를 파악 한 후 다시 둘을 분리해야 한다. 따라서 type변수를 두어 나중에 분리할 수 있게 하였고, prob변수를 두어 각 사람의 생존률을 저장할 수 있게 하였다.(이전에는 type = “class”를 통해 바로 생존 결과를 출력했다면 이번에는 type = “prob”로 두어 확률을 저장했다.)

```
family_result <- foreach(f=folds) %do%{
  f$train$type = "T"
  f$validation$type = "V"

  all<- rbind(f$train, f$validation)
  ctree_model <- ctree(
```

```
    survived ~ pclass + sex + age + sibsp + parch + fare + embarked,
    data = f$train)

  all$prob <- sapply(
    predict(ctree_model, type = "prob", newdata = all),
    function(result){result[1]}#result[2]
  )
}
```

14. 가족ID부여

생존 확률의 예측값을 가족 단위로 모으기 위해 어떤 탑승객이 누구와 가족인지를 알아보자. 티켓 번호별로 가족ID룰 부여하자.

```
#install.packages('plyr')
library(plyr)
```

```
##
## Attaching package: 'plyr'

## The following object is masked from 'package:modeltools':
##
##     empty

## The following objects are masked from 'package:Hmisc':
##
##     is.discrete, summarize
```

```
family_idx <- 0
ticket_based_family_id <- ddply(all, .(ticket), function(rows){#   ticket            .  functino  rows
  family_idx <<- family_idx+1
  return(data.frame(family_id = paste0("TICKET_", family_idx)))#    row   family_id  "TICKET_1"
})
str(all)
```

```
## 'data.frame':    1178 obs. of  13 variables:
##  $ pclass  : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ survived: Factor w/ 2 levels "dead","survived": 2 1 1 2 2 1 2 1 2 2 ...
##  $ name    : chr  "Allison, Master. Hudson Trevor" "Allison, Miss. Helen Loraine" "Allison, Mrs. Huds
##  $ sex     : Factor w/ 2 levels "female","male": 2 1 1 2 1 2 1 2 1 2 ...
##  $ age     : num  0.92 2 25 48 63 39 53 47 26 80 ...
##  $ sibsp   : int  1 1 1 0 1 0 2 1 0 0 ...
##  $ parch   : int  2 2 2 0 0 0 0 0 0 0 ...
##  $ ticket  : chr  "113781" "113781" "113781" "19952" ...
##  $ fare    : num  151.6 151.6 151.6 26.6 78 ...
##  $ cabin   : chr  "C22 C26" "C22 C26" "C22 C26" "E12" ...
##  $ embarked: Factor w/ 3 levels "C","Q","S": 3 3 3 3 3 3 3 1 3 3 ...
##  $ type    : chr  "T" "T" "T" "T" ...
##  $ prob    : num  0.4783 0.0707 0.0707 0.75 0.0707 ...
```

이제 all 데이터 프레임에 ticket값에 따라 family id 를 추가해 보자.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following objects are masked from 'package:Hmisc':
##
##     src, summarize
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
all <- adply(all,
             1,#      , 2
             function(row){
               family_id <-NA
               if(!is.na(row$ticket)){
                 family_id <- subset(ticket_based_family_id, ticket == row$ticket)$family_id

               }
               return(data.frame(family_id = family_id))
             })
str(all)
```

```
## 'data.frame':    1178 obs. of  14 variables:
##  $ pclass   : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ survived : Factor w/ 2 levels "dead","survived": 2 1 1 2 2 1 2 1 2 2 ...
##  $ name     : chr  "Allison, Master. Hudson Trevor" "Allison, Miss. Helen Loraine" "Allison, Mrs. Hu
##  $ sex      : Factor w/ 2 levels "female","male": 2 1 1 2 1 2 1 2 1 2 ...
##  $ age      : num  0.92 2 25 48 63 39 53 47 26 80 ...
##  $ sibsp    : int  1 1 1 0 1 0 2 1 0 0 ...
##  $ parch    : int  2 2 2 0 0 0 0 0 0 0 ...
##  $ ticket   : chr  "113781" "113781" "113781" "19952" ...
##  $ fare     : num  151.6 151.6 151.6 26.6 78 ...
##  $ cabin    : chr  "C22 C26" "C22 C26" "C22 C26" "E12" ...
##  $ embarked : Factor w/ 3 levels "C","Q","S": 3 3 3 3 3 3 3 1 3 3 ...
##  $ type     : chr  "T" "T" "T" "T" ...
##  $ prob     : num  0.4783 0.0707 0.0707 0.75 0.0707 ...
##  $ family_id: Factor w/ 860 levels "TICKET_1","TICKET_2",..: 49 49 49 119 91 16 75 771 114 280 ...
```

15. 가족 구성원 생존 확률의 병합

다음과 같은 변수를 데이터 프레임에 추가해보자.

avg_prob : 가족 구성원의 평균 생존 확률 maybe_parent / maybe_child : 특정 탑승객이 부모인지 자녀인지 여부 parent_prob / child_prob : 부모의 평균 생존율과 자녀의 평균 생존율

가족 구성원의 평균 생존율은 ddply()로 쉽게 구할 수 있다.

```
all <- ddply(all,
             .(family_id),
             function(rows){
               rows$avg_prob <- mean(rows$prob)
               return(rows)
             })
```

다음은 각 탑승객이 부모 또는 자녀 중 어느 쪽에 속하는지를 알아볼 차례이다. 부모인지 자녀인지 여부는 maybe_parent, maybe_child에 저장될 것이며, 부모 자녀를 판단하는 기준으로는 나이(age)를 사용한다.

```
all <- ddply(all,.(family_id), function(rows){
  rows$maybe_parent <- FALSE
  rows$maybe_child <- FALSE
  #
  if(NROW(rows) == 1 ||#              ,      .
     sum(rows$parch)==0 ||#  ,        0    ,      .
     NROW(rows) == sum(is.na(rows$age))){#               .(  na ) ->
    return(rows)
  }

  max_age <- max(rows$age, na.rm = TRUE)
  min_age <- min(rows$age, na.rm = TRUE)
  return(adply(rows, 1, function(row){
    if(!is.na(row$age) && !is.na(row$sex)){
      row$maybe_parent <- (max_age - row$age)<10#      10          ==    .
      row$maybe_child <- (row$age - min_age)<10#       10          ==    .
    }
    return(row)
  }))
})
```

이렇게 부모 자녀 여부를 판단하고 나면 부모의 생존 확률과 자녀의 평균 생존 확률을 구할 수 있다.

```
all <- ddply(all, .(family_id), function(rows){
  rows$avg_parent_prob <- rows$avg_prob #
  rows$avg_child_prob <- rows$avg_prob #
  if(NROW(rows)==1 || sum(rows$parch == 0)){#         return
    return(rows)
  }

  #
  parent_prob <- subset(rows, maybe_parent == TRUE)[,"prob"]
  if(NROW(parent_prob>0)){
    rows$avg_parent_prob <- mean(parent_prob)
  }
```

```
  #
  child_prob <- c(subset(rows, maybe_child == TRUE)[,"prob"])

  if(NROW(child_prob) > 0){
    rows$avg_child_prob <- mean(child_prob)
  }

  return(rows)
})
str(all)
```

```
## 'data.frame':    1178 obs. of  19 variables:
##  $ pclass         : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ survived       : Factor w/ 2 levels "dead","survived": 2 2 2 2 1 1 1 1 1 2 ...
##  $ name           : chr  "Cherry, Miss. Gladys" "Maioni, Miss. Roberta" "Rothes, the Countess. of (Lu
##  $ sex            : Factor w/ 2 levels "female","male": 1 1 1 1 2 2 2 2 2 2 ...
##  $ age            : num  30 16 33 18 52 NA 47 30 42 28 ...
##  $ sibsp          : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ parch          : int  0 0 0 2 1 0 0 0 0 0 ...
##  $ ticket         : chr  "110152" "110152" "110152" "110413" ...
##  $ fare           : num  86.5 86.5 86.5 79.7 79.7 ...
##  $ cabin          : chr  "B77" "B79" "B77" "E68" ...
##  $ embarked       : Factor w/ 3 levels "C","Q","S": 3 3 3 3 3 3 3 3 3 3 ...
##  $ type           : chr  "T" "T" "T" "T" ...
##  $ prob           : num  0.0707 0.0707 0.0707 0.0707 0.75 ...
##  $ family_id      : Factor w/ 860 levels "TICKET_1","TICKET_2",..: 1 1 1 2 2 3 3 4 5 6 ...
##  $ avg_prob       : num  0.0707 0.0707 0.0707 0.4104 0.4104 ...
##  $ maybe_parent   : logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
##  $ maybe_child    : logi  FALSE FALSE FALSE TRUE FALSE FALSE ...
##  $ avg_parent_prob: num  0.0707 0.0707 0.0707 0.75 0.75 ...
##  $ avg_child_prob : num  0.0707 0.0707 0.0707 0.0707 0.0707 ...
```

16.가족 정보를 사용한 ctree()모델링

이제 all데이터를 사용해 모델을 만들고 성능을 평가해 보자. type, avg_prob, maybe_parent, maybe_child, avg_parent_prob, avg_child_prob를 사용할 것이다.

all 에 있는 정보 중 훈련데이터를 사용해 ctree()를 수행하고 이를 검증 데이터에 적용해보자.

```
f$train <- subset(all, type="T")
f$validation <- subset(all, type = "V")

(m<- ctree(survived ~ pclass + sex + age + sibsp + parch + fare + embarked +
             maybe_parent + maybe_child + avg_prob + avg_parent_prob + avg_child_prob,
           data = f$train))
```

```
##
##   Conditional inference tree with 7 terminal nodes
##
## Response:  survived
## Inputs:  pclass, sex, age, sibsp, parch, fare, embarked, maybe_parent, maybe_child, avg_prob, avg_pa
## Number of observations:  1178
```

```
##
## 1) avg_prob <= 0.576093; criterion = 1, statistic = 401.181
##   2) sex == {male}; criterion = 1, statistic = 62.756
##     3) age <= 13; criterion = 1, statistic = 24.541
##       4)*  weights = 26
##     3) age > 13
##       5) pclass == {1, 3}; criterion = 0.996, statistic = 15.864
##         6)*  weights = 86
##       5) pclass == {2}
##         7)*  weights = 30
##   2) sex == {female}
##     8) pclass == {1, 2}; criterion = 1, statistic = 63.276
##       9)*  weights = 219
##     8) pclass == {3}
##       10)*  weights = 139
## 1) avg_prob > 0.576093
##   11) avg_child_prob <= 0.75; criterion = 1, statistic = 23.221
##     12)*  weights = 125
##   11) avg_child_prob > 0.75
##     13)*  weights = 553
```

```
print(m)
```

```
##
##   Conditional inference tree with 7 terminal nodes
##
## Response:  survived
## Inputs:  pclass, sex, age, sibsp, parch, fare, embarked, maybe_parent, maybe_child, avg_prob, avg_pa
## Number of observations:  1178
##
## 1) avg_prob <= 0.576093; criterion = 1, statistic = 401.181
##   2) sex == {male}; criterion = 1, statistic = 62.756
##     3) age <= 13; criterion = 1, statistic = 24.541
##       4)*  weights = 26
##     3) age > 13
##       5) pclass == {1, 3}; criterion = 0.996, statistic = 15.864
##         6)*  weights = 86
##       5) pclass == {2}
##         7)*  weights = 30
##   2) sex == {female}
##     8) pclass == {1, 2}; criterion = 1, statistic = 63.276
##       9)*  weights = 219
##     8) pclass == {3}
##       10)*  weights = 139
## 1) avg_prob > 0.576093
##   11) avg_child_prob <= 0.75; criterion = 1, statistic = 23.221
##     12)*  weights = 125
##   11) avg_child_prob > 0.75
##     13)*  weights = 553
```

```
predicted <- predict(m, newdata = f$validation)
```

확인결과, avg_prob, avg_child_prob이 유용하게 사용되고 있는것을 알 수 있다.(일찍 사용되었다.)