Ch6Report

Madi Kassymbekov

4/9/2021

Question 1

```
# Data preprocessing
jasa$subject <- 1:nrow(jasa)
tdata <- with(jasa, data.frame(subject = subject,
futime= pmax(.5, fu.date - accept.dt),
txtime= ifelse(tx.date== fu.date,
(tx.date -accept.dt) -.5,
(tx.date - accept.dt)),
fustat = fustat
))
sdata <- tmerge(jasa, tdata, id=subject, death = event(futime, fustat), trt = tdc(txtime),
options= list(idname="subject"))
sdata$year <- as.numeric(sdata$accept.dt - as.Date("1967-10-01"))/365.25
sdata=sdata[,c(6,7,15,16,17,18,19)]
```

a. Construct this new dataset and provide a summary using the simple command summary

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.4

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

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

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

```
intervaldata <- sdata %>% group_by(subject) %>% top_n(1, tstop)
summary(intervaldata)
```

```
##     surgery            age            subject          tstart
##  Min.   :0.0000   Min.   : 8.786   Min.   :  1.0   Min.   :  0.00
##  1st Qu.:0.0000   1st Qu.:41.180   1st Qu.: 26.5   1st Qu.:  0.00
##  Median :0.0000   Median :47.751   Median : 52.0   Median :  9.00
##  Mean   :0.1553   Mean   :45.169   Mean   : 52.0   Mean   : 25.24
##  3rd Qu.:0.0000   3rd Qu.:52.059   3rd Qu.: 77.5   3rd Qu.: 31.50
##  Max.   :1.0000   Max.   :64.408   Max.   :103.0   Max.   :309.00
##      tstop             death             trt
##  Min.   :   0.5   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:  32.5   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :  89.0   Median :1.0000   Median :1.0000
##  Mean   : 309.2   Mean   :0.7282   Mean   :0.6699
##  3rd Qu.: 411.5   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1799.0   Max.   :1.0000   Max.   :1.0000
```

Answer: Summary statistics are above.

b. Fit a Cox model to this data using the variables age , trt and surgery and the R function coxph . The interval censoring is defined with the function Surv that we used in the case of right censoring. Use the help of this function to understand how to code it. Comment on the significance of the variables.

```
fitcoxinterval <- coxph(Surv(tstart, tstop, death)~age+trt+surgery, data=intervaldata)
summary(fitcoxinterval)
```

```
## Call:
## coxph(formula = Surv(tstart, tstop, death) ~ age + trt + surgery,
##     data = intervaldata)
##
##   n= 103, number of events= 75
##
##             coef exp(coef) se(coef)      z Pr(>|z|)
## age      0.05065   1.05195  0.01451  3.491 0.000482 ***
## trt     -1.05867   0.34692  0.31442 -3.367 0.000760 ***
## surgery -0.41156   0.66261  0.37204 -1.106 0.268628
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##         exp(coef) exp(-coef) lower .95 upper .95
## age        1.0520     0.9506    1.0225    1.0823
## trt        0.3469     2.8825    0.1873    0.6425
## surgery    0.6626     1.5092    0.3196    1.3739
##
## Concordance= 0.657  (se = 0.03 )
## Likelihood ratio test= 21.45  on 3 df,   p=8e-05
## Wald test            = 20.17  on 3 df,   p=2e-04
## Score (logrank) test = 20.72  on 3 df,   p=1e-04
```

Answer: Based on the summary of fitted Cox model, increase in age by one year multiplies the risk of death by 1.052, transplant decreases the risk of death by a multiple of 0.347 and surgery variable is way above 0.05 significance which indicates that surgery is not a significant variable for explaining the risk of death.

c. Obtain the risk fitted values for all subjects in the dataset and print the values for the first 10 subjects

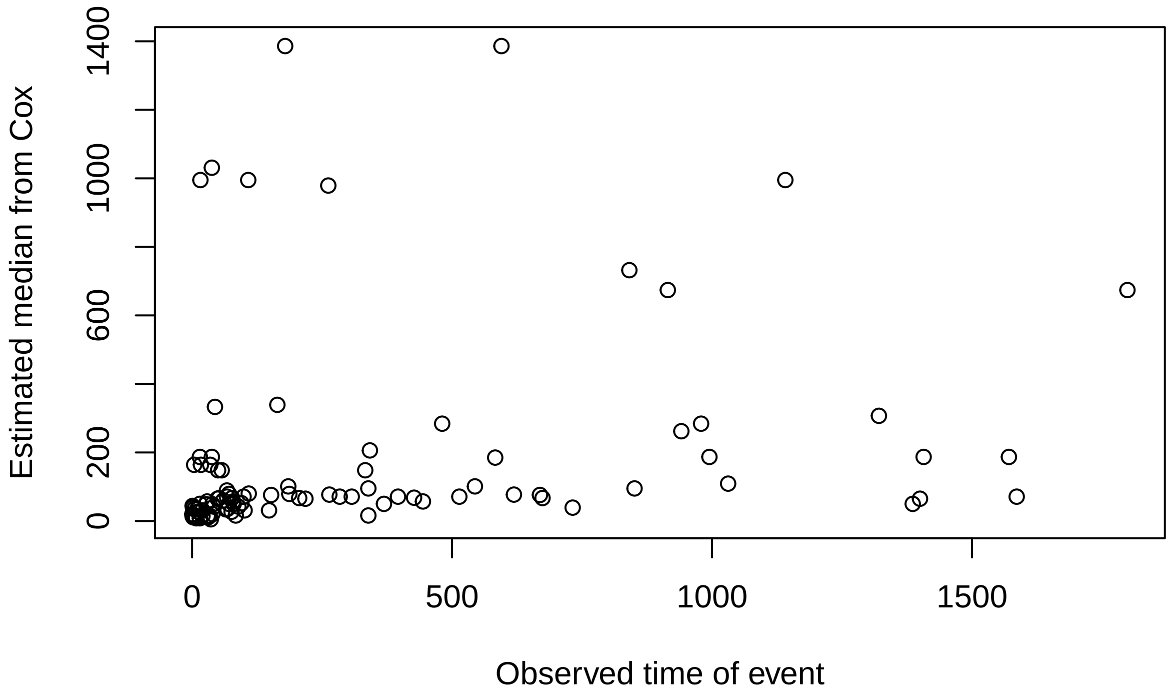
```
predcoxriskinterval=predict(fitcoxinterval, newdata=intervaldata, type="risk")
print(predcoxriskinterval[1:10])
```

```
##         1         2         3         4         5         6         7         8
## 0.4840872 1.4016068 0.5507928 0.2705848 0.2908555 1.6118603 0.4630134 1.0091746
##         9        10
## 1.1061947 0.3030839
```

Answer: From the fitted values for first 10 subjects subject 6,2, 8 and 9 have risk coefficient multiples higher than 1 as they are older than 40 and had not received any surgery or transplant.

d. Compare the predictions of the median survival times with the observed survival times (there are several simple ways to do it)

```
coxsurvtime=survfit(fitcoxinterval,newdata= intervaldata)
coxsurvtime<-summary(coxsurvtime)$table[, "median"]
intervaldata$Median<-coxsurvtime

plot(intervaldata$tstop , intervaldata$Median,
     xlab = "Observed time of event" , ylab = "Estimated median from Cox")
```



```
coxMSE <- mean((intervaldata$tstop[!is.na(coxsurvtime)] - coxsurvtime[!is.na(coxsurvtime)])^2)
```

Answer: By looking at the comparison plot of median survival time vs observed survival time, we see that cox model is underestimating survival time which is proved by large MSE of 2.3052249×10^5.

2. We are now going to work with the time-varying dataset sdata . Fit a Cox model to this data (use the command coxph as it is) and compare the results with the previous question 1b).

```
fitcoxsdata = coxph(Surv(tstart, tstop,death)~age+trt+surgery, data=sdata)
summary(fitcoxsdata)
```

```
## Call:
## coxph(formula = Surv(tstart, tstop, death) ~ age + trt + surgery,
##     data = sdata)
##
##   n= 170, number of events= 75
##
##               coef exp(coef) se(coef)      z Pr(>|z|)
## age      0.03055   1.03103  0.01389  2.199   0.0279 *
## trt      0.01405   1.01415  0.30822  0.046   0.9636
## surgery -0.77326   0.46150  0.35966 -2.150   0.0316 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##         exp(coef) exp(-coef) lower .95 upper .95
## age        1.0310     0.9699    1.0033    1.0595
## trt        1.0142     0.9860    0.5543    1.8555
## surgery    0.4615     2.1668    0.2280    0.9339
##
## Concordance= 0.599  (se = 0.036 )
## Likelihood ratio test= 10.72  on 3 df,   p=0.01
## Wald test            = 9.68  on 3 df,   p=0.02
## Score (logrank) test = 10  on 3 df,   p=0.02
```

Answer: Compared to previous dataset in 1b, sdata cox model identifies transplant trt variable as non-significant variable with p-value of 0.9636 for predicting risk of death while surgery variable became significant which is opposite to the summary of cox model in 1b.

Question 2

```r
library(survival)
data("colon")
```

```
## Warning in data("colon"): data set 'colon' not found
```

```r
mycolon=colon[colon$etype==2,]
mycolon.train=mycolon[1:500,-c(1,2,16)]
mycolon.test=mycolon[501:929,-c(1,2,16)]
id.train=mycolon[1:500,1]
id.test=mycolon[501:929,1]
#Omit NA rows from test set
mycolon.test=na.omit(mycolon.test)
mycolon.train=na.omit(mycolon.train)
```

```r
#Function to compute C-index with survival data from the notes
cindexsurvival=function(riskhat,t,status)
{
  # riskhat = risk prediction (higher values mean more at risk)
  # t = observed time
  # status = censoring indicator (1=dead; 0=alive)
  n=length(riskhat)
  cc=0
  npair=0
  for(i in 1:(n-1))
  {
    for(j in (i+1):n)
    {
      if(((t[i]<t[j]) & status[i]==1) | ((t[j]<t[i]) & status[j]==1))
      {
        cc=cc+(riskhat[i] > riskhat[j])*(t[i]<t[j])+(riskhat[i] < riskhat[j])*(t[i]>t[j])
        npair=npair+1
      }
    }
  }
  cc/npair
}
```

1. Compute the predicted survival time on the test set obtained from a Cox model (include all variables in each model) and compute the C-index based on these predicted values

```r
fitcox=coxph(Surv(time, status)~., data=mycolon.train)
predcoxrisktest=predict(fitcox, newdata=mycolon.test,type="risk")
c_index_Cox<- cindexsurvival(predcoxrisktest,mycolon.test[,"time"],mycolon.test[,"status"])
print(c_index_Cox)
```

```
##      1001
## 0.6443206
```

Answer: C-index calculated on the test sets' predicted survival time from cox model, shows that c-index is 0.6443206

2. Compute the predicted survival time on the test set obtained from a AFT model assuming a log-normal distribution (include all variables in each model) and compute the C-index based on these predicted values

```r
fitaft = survreg(Surv(time, status)~., data=mycolon.train, dist="lognormal")
predaft = predict(fitaft, newdata=mycolon.test, type="response")
#Survival time needs to be converted to risk for c-index, so reciprocal of survtime = risk
c_index_AFT<-cindexsurvival(1/predaft,mycolon.test[,"time"],mycolon.test[,"status"])
print(c_index_AFT)
```

```
##      1001
## 0.6431993
```

Answer: C-index calculated for the predicted values of test set from AFT model is 0.6431993 which is similar to cox model's one.

3. Compute the predicted survival time on the test set obtained from a survival random forest with 400 trees (include all variables in each model) and compute the C-index based on these predicted values

```r
library(randomForestSRC)
fitsrc=rfsrc(Surv(time,status)~.,data=mycolon.train,ntree=400)
predsrc=predict(fitsrc,newdata=mycolon.test)
#Survival time needs to be converted to risk for c-index, so reciprocal of survtime = risk
cindexsurvival(1/predsrc$predicted,mycolon.test[,"time"],mycolon.test[,"status"])
```

```
## [1] 0.3368727
```

4. Compare the three C-index values and comment Answer: Based on c-indexes of 3 different models, cox and AFT models' c-indexes are similar while survival random forest's one is much lower which suggests the underestimation by the forest model compared to observations. It should be noted, that c-index is not sufficient to tell whether model is good for the fitted data, it only indicates how model is able to identify subjects who are more at risk than others. Predicted survival time may not match the actual one and therefore further analysis is required to pick the best method.

Question 3

1. Re-run the three methods we used in the uplift modeling chapter on the simulated dataset used in this chapter. Evaluate the agreement between the three methods with respect to the ranking of the test-set subjects based on the predicted lift. You can use the strategy of your choice

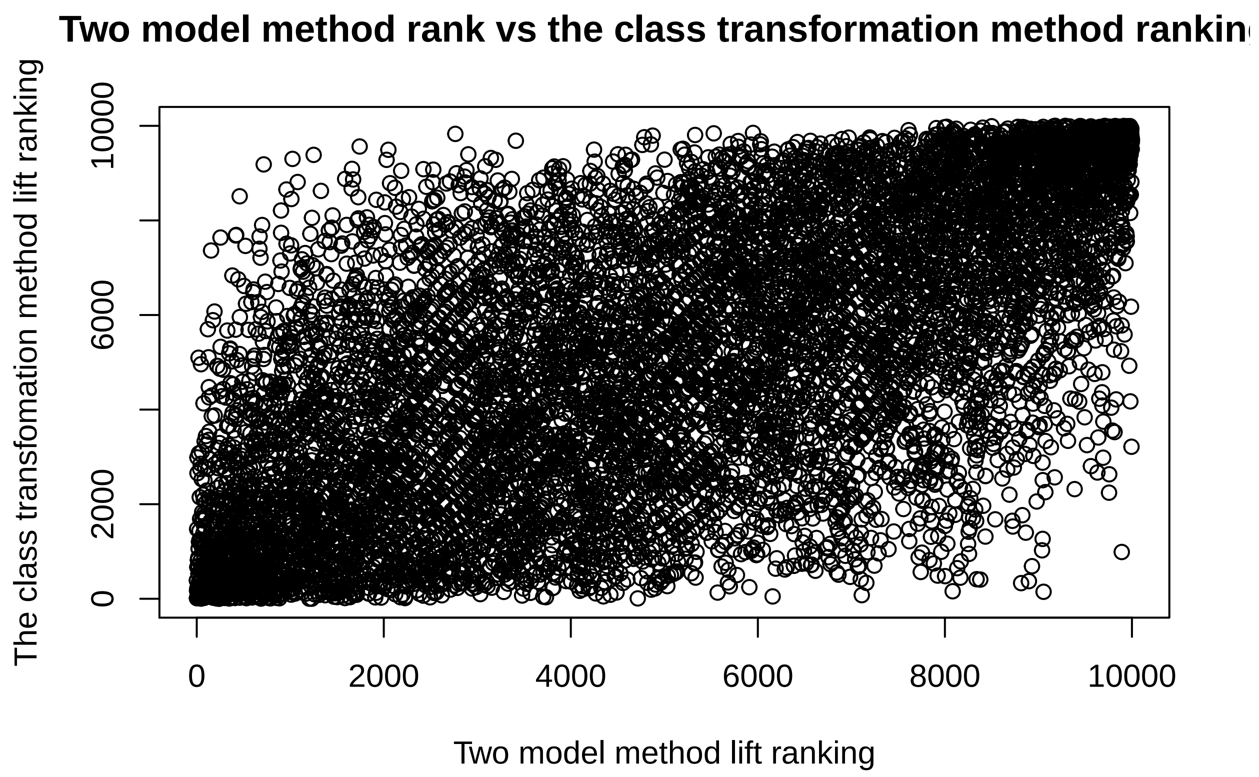
```r
genlift=function(nt,nc)
{
# nt =sample size for treatment
# nc =sample size for control
# output:
# pc = P(Y=1) if control and pt = P(Y=1) if treatment
# truep = P(Y=1) for this subject. # (it is either pc or pt)
# lift = lift (pt-pc)
# treat = treatment indicator (1=treatment; 0=control)
# yf = response (0 or 1) (factor version)
# y = response (0 or 1) (numeric version)
n=nt+nc
treat=c(rep(1,nt),rep(0,nc))
x=matrix(rnorm(n*10),ncol=10)
pc=1/(1+exp(-(-4+.3*(x[,1]+3)^2+x[,2]+.5*x[,2]*x[,3])))
pt=1/(1+exp(-(-4+.5*(x[,1]+3)^2+1.3*x[,2]+.8*x[,2]*x[,3]+
1.5*abs(x[,4]))))
lift=pt-pc
truep=pc*(1-treat)+pt*treat
y=rbinom(n,1,truep)
yf=factor(y)
out=data.frame(pc,pt,truep,lift,x,treat,y,yf)
names(out)=c("pc","pt","truep","lift",
"x1","x2","x3","x4","x5","x6","x7","x8","x9","x10",
"treat","y","yf")
out
}

set.seed(123456789)
dattrain=genlift(1000,1000)
dattest=genlift(5000,5000)


# two models approach
dattrainc=dattrain[dattrain$treat==0,]
rffitc=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainc,ntree=100)
predc=predict(rffitc,newdata=dattest)$predicted[,2]

dattraint=dattrain[dattrain$treat==1,]
rffitt=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattraint,ntree=100)
predt=predict(rffitt,newdata=dattest)$predicted[,2]
lift2models=predt-predc

# Class transformation method
dattrain$z=factor((dattrain$y == dattrain$treat))
classrf=rfsrc(z~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrain,ntree=100)
liftclass=2*predict(classrf,newdata=dattest)$predicted[,2]-1

# Tree based uplift forest
uprf=upliftRF(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+trt(treat),
data=dattrain,split_method ="ED",ntree=100)
preduprf=predict(uprf,newdata=dattest)
liftuprf=preduprf[,1]-preduprf[,2]
```

```r
c(mean(abs(dattest$lift-lift2models)),
+ mean(abs(dattest$lift-liftuprf)),
+ mean(abs(dattest$lift-liftclass)))
```

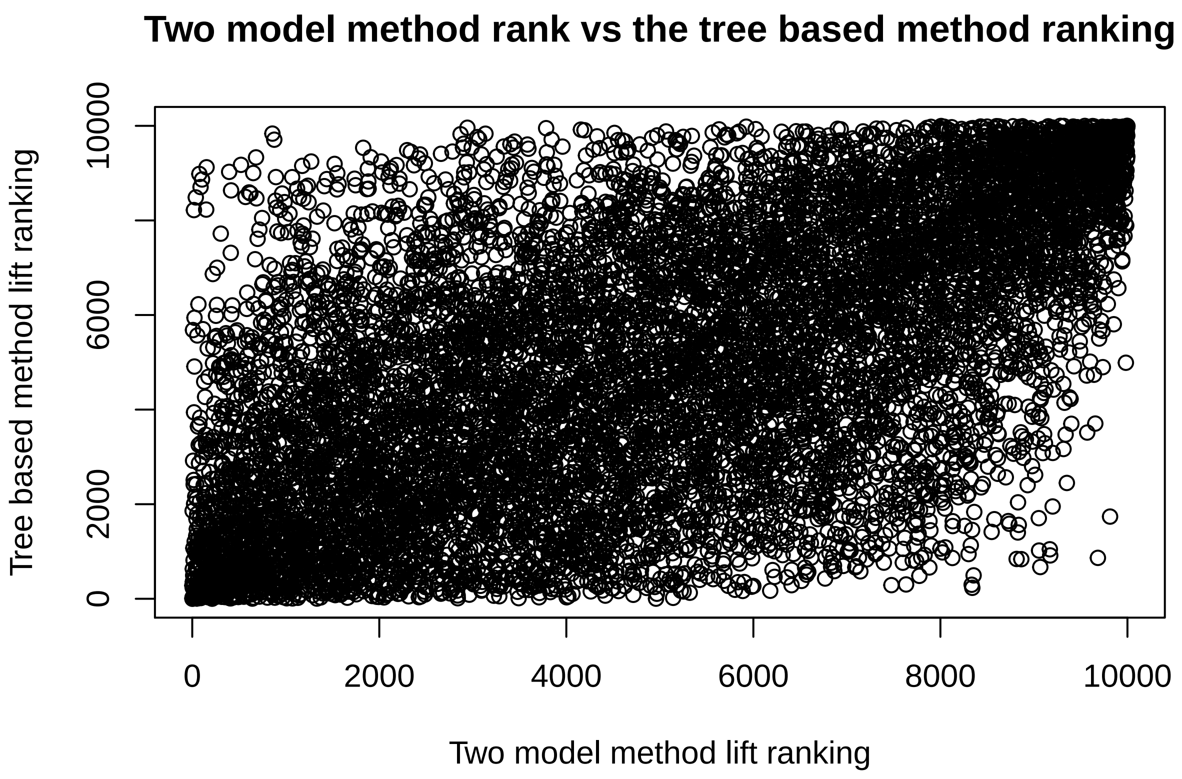
```
## [1] 0.1299159 0.1750095 0.1771534
```

```r
#Ranking list subjects
subjectid <- seq(1 , nrow(dattest) , 1 )
```

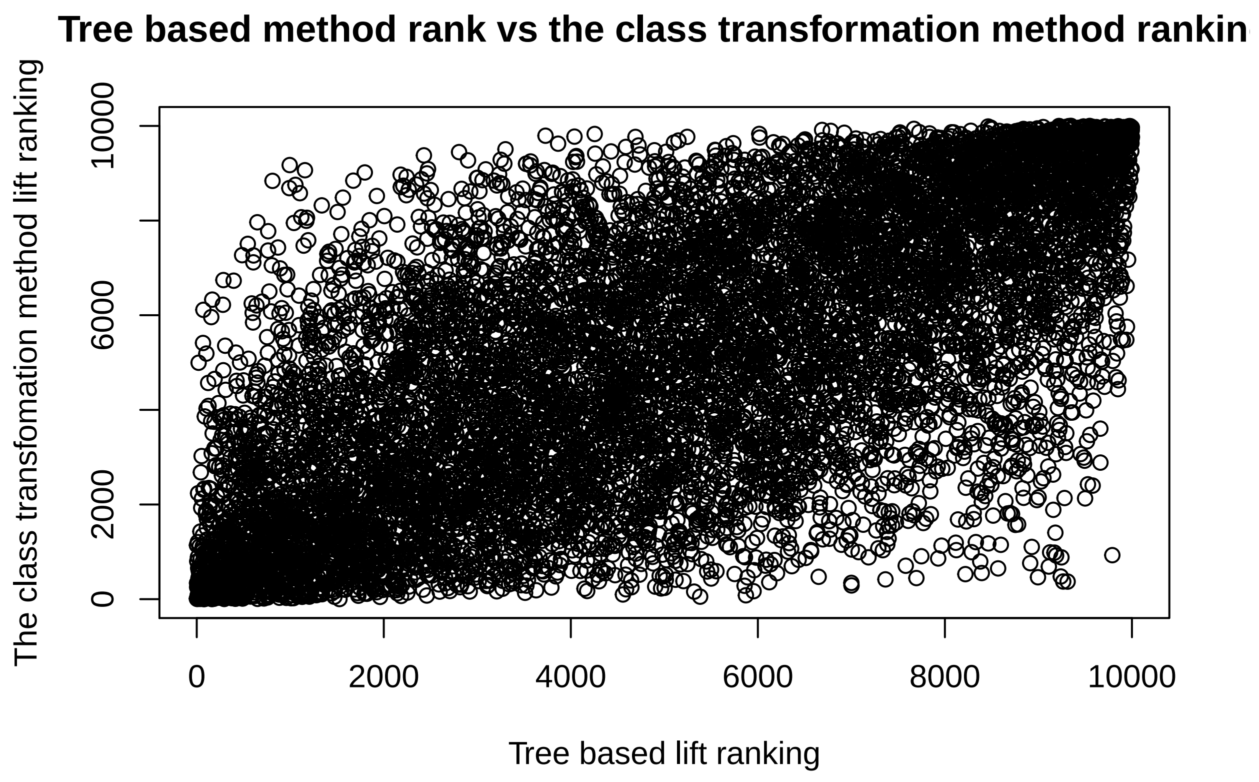
```r
#True ranking
trueranking <- as.data.frame(subjectid)
trueranking$truelift <- dattest$lift
trueranking <- trueranking[order(trueranking$truelift , decreasing = TRUE), ]
trueranking$truerank <- subjectid

#Rank of two models method
twomodelranking <- as.data.frame(subjectid)
twomodelranking$twomodellift <- lift2models
twomodelranking <- twomodelranking[order(twomodelranking$twomodellift , decreasing = TRUE), ]
twomodelranking$twomodelrank <- subjectid

#Rank of class transformation
classranking <- as.data.frame(subjectid)
classranking$classlift <- liftclass
classranking <- classranking[order(classranking$classlift , decreasing = TRUE), ]
classranking$classrank <- subjectid

#Rank of uplift forest
treeranking <- as.data.frame(subjectid)
treeranking$treelift <- liftuprf
treeranking <- treeranking[order(treeranking$treelift , decreasing = TRUE), ]
treeranking$treerank <- subjectid


topliftranking <- merge(twomodelranking, classranking, by="subjectid")
topliftranking <- merge(topliftranking, treeranking, by="subjectid")
topliftranking <- merge(topliftranking, trueranking, by="subjectid")

#MAE for measuring rank error rate for each model
MAE_twomodel <- mean(abs(topliftranking$twomodelrank - topliftranking$truerank))
MAE_class <- mean(abs(topliftranking$classrank - topliftranking$truerank))
MAE_upliftforest <- mean(abs(topliftranking$treerank - topliftranking$truerank))


plot(topliftranking$twomodelrank , topliftranking$classrank ,
     main="Two model method rank vs the class transformation method ranking",
     xlab="Two model method lift ranking",
     ylab="The class transfomation method lift ranking")
```



```
plot(topliftranking$twomodelrank , topliftranking$treerank ,
     main="Two model method rank vs the tree based method ranking",
     xlab="Two model method lift ranking",
     ylab="Tree based method lift ranking")
```



```
plot(topliftranking$treerank , topliftranking$classrank ,
     main="Tree based method rank vs the class transformation method ranking",
     xlab="Tree based lift ranking",
     ylab="The class transfomation method lift ranking")
```



```
MAE_results <- data.frame(MAE_twomodel, MAE_class, MAE_upliftforest)
names(MAE_results) <- c("TwoModel MAE", "Uplift Forest MAE", "Class MAE")
knitr::kable(MAE_results, caption="MAE values of three models on lift ranking")
```

Table 1: MAE values of three models on lift ranking

TwoModel MAE	Uplift Forest MAE	Class MAE
1648.341	2281.452	2234.849

Answer: To evaluate test set ranking based on the predicted lift of three models, comparison plots were used to assess whether there is an agreement. By looking at ranking plots for each method, there is an agreement between methods about top and lowest ranks in the test set, however agreement is much less noticeable towards the subjects in the middle where we see less bold points which indicates a match between methods. Additional metric to evaluate ranking performance is to calculate MAE for each method compared to true lift ranking. Based on MAE, two model method was the best with MAE 1648.3408 while class and tree based MAE were significantly higher.

2. The dataset used in class was simulated such that there is a perfect balance between groups (50% are cases and 50% are controls). By resimulating the data under different scenarios, evaluate if a disequilibrium between the group size (keeping the same total size) has an impact on the performance of the methods.

```
set.seed(5984736)
#10% treatment 90% control
dattrainunbal=genlift(200,1800)
dattestunbal=genlift(1000,9000)
```

```r
dattrainc=dattrainunbal[dattrainunbal$treat==0,]
rffitc=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainc,ntree=100)
predc=predict(rffitc,newdata=dattestunbal)$predicted[,2]

dattraint=dattrainunbal[dattrainunbal$treat==1,]
rffitt=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattraint,ntree=100)
predt=predict(rffitt,newdata=dattestunbal)$predicted[,2]
lift2models=predt-predc

#T equation instead of Z as data is unbalanced
p_treat = 0.1
dattrainunbal$t=factor(dattrainunbal$y*((dattrain$treat/p_treat)-
                                          (1-dattrain$treat)/(1-p_treat)))
classrf=rfsrc(t~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainunbal,ntree=100)
liftclass=p_treat*predict(classrf,newdata=dattestunbal)$predicted[,1]+(1-p_treat)*
  predict(classrf,newdata=dattestunbal)$predicted[,3]

uprf=upliftRF(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+trt(treat),
data=dattrainunbal,split_method ="ED",ntree=100)
preduprf=predict(uprf,newdata=dattestunbal)
liftuprf=preduprf[,1]-preduprf[,2]

results <- data.frame("10% treatment, 20% control",
mean(abs(dattestunbal$lift-lift2models)), mean(abs(dattestunbal$lift-liftuprf)),
mean(abs(dattestunbal$lift-liftclass)))
names(results) <- c("Simulated data", "TwoModel MAE",
                    "Uplift Forest MAE", "Class MAE")

#20% treatment 80% control
dattrainunbal=genlift(400,1600)
dattestunbal=genlift(2000,8000)

dattrainc=dattrainunbal[dattrainunbal$treat==0,]
rffitc=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainc,ntree=100)
predc=predict(rffitc,newdata=dattestunbal)$predicted[,2]

dattraint=dattrainunbal[dattrainunbal$treat==1,]
rffitt=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattraint,ntree=100)
predt=predict(rffitt,newdata=dattestunbal)$predicted[,2]
lift2models=predt-predc

#T equation instead of Z as data is unbalanced
p_treat = 0.2
dattrainunbal$t=factor(dattrainunbal$y*((dattrain$treat/p_treat)-
                                          (1-dattrain$treat)/(1-p_treat)))
classrf=rfsrc(t~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainunbal,ntree=100)
liftclass=p_treat*predict(classrf,newdata=dattestunbal)$predicted[,1]+
  (1-p_treat)*predict(classrf,newdata=dattestunbal)$predicted[,3]

uprf=upliftRF(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+trt(treat),
data=dattrainunbal,split_method ="ED",ntree=100)
preduprf=predict(uprf,newdata=dattestunbal)
liftuprf=preduprf[,1]-preduprf[,2]
```

```r
results[nrow(results) + 1,] = list(Name="20% treatment 80% control",
                                   mean(abs(dattestunbal$lift-lift2models)),
                                   mean(abs(dattestunbal$lift-liftuprf)),
                                   mean(abs(dattestunbal$lift-liftclass)))

#30% treatment, 70% control
dattrainunbal=genlift(600,1400)
dattestunbal=genlift(3000,7000)

dattrainc=dattrainunbal[dattrainunbal$treat==0,]
rffitc=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainc,ntree=100)
predc=predict(rffitc,newdata=dattestunbal)$predicted[,2]

dattraint=dattrainunbal[dattrainunbal$treat==1,]
rffitt=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattraint,ntree=100)
predt=predict(rffitt,newdata=dattestunbal)$predicted[,2]
lift2models=predt-predc

#T equation instead of Z as data is unbalanced
p_treat = 0.3
dattrainunbal$t=factor(dattrainunbal$y*((dattrain$treat/p_treat)-
                                         (1-dattrain$treat)/(1-p_treat)))
classrf=rfsrc(t~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainunbal,ntree=100)
liftclass=p_treat*predict(classrf,newdata=dattestunbal)$predicted[,1]+
  (1-p_treat)*predict(classrf,newdata=dattestunbal)$predicted[,3]


uprf=upliftRF(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+trt(treat),
data=dattrainunbal,split_method ="ED",ntree=100)
preduprf=predict(uprf,newdata=dattestunbal)
liftuprf=preduprf[,1]-preduprf[,2]

results[nrow(results) + 1,] =list(Name="30% treatment 70% control",
                                   mean(abs(dattestunbal$lift-lift2models)),
                                   mean(abs(dattestunbal$lift-liftuprf)),
                                   mean(abs(dattestunbal$lift-liftclass)))

#40% treatment, 60% control
dattrainunbal=genlift(800,1200)
dattestunbal=genlift(4000,6000)

dattrainc=dattrainunbal[dattrainunbal$treat==0,]
rffitc=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainc,ntree=100)
predc=predict(rffitc,newdata=dattestunbal)$predicted[,2]

dattraint=dattrainunbal[dattrainunbal$treat==1,]
rffitt=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattraint,ntree=100)
predt=predict(rffitt,newdata=dattestunbal)$predicted[,2]
lift2models=predt-predc

#T equation instead of Z as data is unbalanced
p_treat = 0.4
dattrainunbal$t=factor(dattrainunbal$y*((dattrain$treat/p_treat)-
                                          (1-dattrain$treat)/(1-p_treat)))
classrf=rfsrc(t~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainunbal,ntree=100)
liftclass=p_treat*predict(classrf,newdata=dattestunbal)$predicted[,1]+
  (1-p_treat)*predict(classrf,newdata=dattestunbal)$predicted[,3]

uprf=upliftRF(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+trt(treat),
data=dattrainunbal,split_method ="ED",ntree=100)
preduprf=predict(uprf,newdata=dattestunbal)
liftuprf=preduprf[,1]-preduprf[,2]

results[nrow(results) + 1,] = list(Name="40% treatment, 60% control",
                                   mean(abs(dattestunbal$lift-lift2models)),
                                   mean(abs(dattestunbal$lift-liftuprf)),
                                   mean(abs(dattestunbal$lift-liftclass)))

#60% treatment, 40% control
dattrainunbal=genlift(1200,800)
dattestunbal=genlift(6000,4000)

dattrainc=dattrainunbal[dattrainunbal$treat==0,]
rffitc=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainc,ntree=100)
predc=predict(rffitc,newdata=dattestunbal)$predicted[,2]

dattraint=dattrainunbal[dattrainunbal$treat==1,]
rffitt=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattraint,ntree=100)
predt=predict(rffitt,newdata=dattestunbal)$predicted[,2]
lift2models=predt-predc

#T equation instead of Z as data is unbalanced
p_treat = 0.6
dattrainunbal$t=factor(dattrainunbal$y*((dattrain$treat/p_treat)-
                                          (1-dattrain$treat)/(1-p_treat)))
classrf=rfsrc(t~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainunbal,ntree=100)
liftclass=p_treat*predict(classrf,newdata=dattestunbal)$predicted[,1]+
  (1-p_treat)*predict(classrf,newdata=dattestunbal)$predicted[,3]

uprf=upliftRF(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+trt(treat),
data=dattrainunbal,split_method ="ED",ntree=100)
preduprf=predict(uprf,newdata=dattestunbal)
liftuprf=preduprf[,1]-preduprf[,2]

results[nrow(results) + 1,] = list(Name="60% treatment, 40% control",
                                   mean(abs(dattestunbal$lift-lift2models)),
                                   mean(abs(dattestunbal$lift-liftuprf)),
                                   mean(abs(dattestunbal$lift-liftclass)))

#70% treatment, 30% control
dattrainunbal=genlift(1400,600)
dattestunbal=genlift(7000,3000)

dattrainc=dattrainunbal[dattrainunbal$treat==0,]
rffitc=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainc,ntree=100)
predc=predict(rffitc,newdata=dattestunbal)$predicted[,2]
```

```
dattraint=dattrainunbal[dattrainunbal$treat==1,]
rffitt=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattraint,ntree=100)
predt=predict(rffitt,newdata=dattestunbal)$predicted[,2]
lift2models=predt-predc

#T equation instead of Z as data is unbalanced
p_treat = 0.7
dattrainunbal$t=factor(dattrainunbal$y*((dattrain$treat/p_treat)-
                                        (1-dattrain$treat)/(1-p_treat)))
classrf=rfsrc(t~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainunbal,ntree=100)
liftclass=p_treat*predict(classrf,newdata=dattestunbal)$predicted[,1]+
  (1-p_treat)*predict(classrf,newdata=dattestunbal)$predicted[,3]

uprf=upliftRF(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+trt(treat),
data=dattrainunbal,split_method ="ED",ntree=100)
preduprf=predict(uprf,newdata=dattestunbal)
liftuprf=preduprf[,1]-preduprf[,2]

results[nrow(results) + 1,] = list(Name="70% treatment, 30% control",
                                   mean(abs(dattestunbal$lift-lift2models)),
                                   mean(abs(dattestunbal$lift-liftuprf)),
                                   mean(abs(dattestunbal$lift-liftclass)))


#80% treatment 20% control
dattrainunbal=genlift(1600,400)
dattestunbal=genlift(8000,2000)

dattrainc=dattrainunbal[dattrainunbal$treat==0,]
rffitc=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainc,ntree=100)
predc=predict(rffitc,newdata=dattestunbal)$predicted[,2]

dattraint=dattrainunbal[dattrainunbal$treat==1,]
rffitt=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattraint,ntree=100)
predt=predict(rffitt,newdata=dattestunbal)$predicted[,2]
lift2models=predt-predc

#T equation instead of Z as data is unbalanced
p_treat = 0.8
dattrainunbal$t=factor(dattrainunbal$y*((dattrain$treat/p_treat)-
                                        (1-dattrain$treat)/(1-p_treat)))
classrf=rfsrc(t~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainunbal,ntree=100)
liftclass=p_treat*predict(classrf,newdata=dattestunbal)$predicted[,1]+
  (1-p_treat)*predict(classrf,newdata=dattestunbal)$predicted[,3]

uprf=upliftRF(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+trt(treat),
data=dattrainunbal,split_method ="ED",ntree=100)
preduprf=predict(uprf,newdata=dattestunbal)
liftuprf=preduprf[,1]-preduprf[,2]

results[nrow(results) + 1,] = list(Name="80% treatment 20% control",
                                   mean(abs(dattestunbal$lift-lift2models)),
                                   mean(abs(dattestunbal$lift-liftuprf)),
                                    mean(abs(dattestunbal$lift-liftclass)))

#90% treatment 10% control
dattrainunbal=genlift(1800,200)
dattestunbal=genlift(9000,1000)

dattrainc=dattrainunbal[dattrainunbal$treat==0,]
rffitc=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainc,ntree=100)
predc=predict(rffitc,newdata=dattestunbal)$predicted[,2]

dattraint=dattrainunbal[dattrainunbal$treat==1,]
rffitt=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattraint,ntree=100)
predt=predict(rffitt,newdata=dattestunbal)$predicted[,2]
lift2models=predt-predc

#T equation instead of Z as data is unbalanced
p_treat = 0.9
dattrainunbal$t=factor(dattrainunbal$y*((dattrain$treat/p_treat)-
                                           (1-dattrain$treat)/(1-p_treat)))
classrf=rfsrc(t~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainunbal,ntree=100)
liftclass=p_treat*predict(classrf,newdata=dattestunbal)$predicted[,1]+
  (1-p_treat)*predict(classrf,newdata=dattestunbal)$predicted[,3]

uprf=upliftRF(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+trt(treat),
data=dattrainunbal,split_method ="ED",ntree=100)
preduprf=predict(uprf,newdata=dattestunbal)
liftuprf=preduprf[,1]-preduprf[,2]

results[nrow(results) + 1,] = list(Name="90% treatment 10% control",
                                    mean(abs(dattestunbal$lift-lift2models)),
                                    mean(abs(dattestunbal$lift-liftuprf)),
                                    mean(abs(dattestunbal$lift-liftclass)))

#Balanced Dataset for comparison
dattrain=genlift(1000,1000)
dattest=genlift(5000,5000)
# two models approach
dattrainc=dattrain[dattrain$treat==0,]
rffitc=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrainc,ntree=100)
predc=predict(rffitc,newdata=dattest)$predicted[,2]

dattraint=dattrain[dattrain$treat==1,]
rffitt=rfsrc(yf~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattraint,ntree=100)
predt=predict(rffitt,newdata=dattest)$predicted[,2]
lift2models=predt-predc

# Class transformation method
dattrain$z=factor((dattrain$y == dattrain$treat))
classrf=rfsrc(z~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10,data=dattrain,ntree=100)
liftclass=2*predict(classrf,newdata=dattest)$predicted[,2]-1

# Tree based uplift forest
uprf=upliftRF(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+trt(treat),
data=dattrain,split_method ="ED",ntree=100)
preduprf=predict(uprf,newdata=dattest)
liftuprf=preduprf[,1]-preduprf[,2]

results[nrow(results) + 1,] = list(Name="50% treatment 50% control",
                                   mean(abs(dattest$lift-lift2models)),
                                   mean(abs(dattest$lift-liftuprf)),
                                   mean(abs(dattest$lift-liftclass)))

knitr::kable(results, caption="MAE values of three models dependent on simulated dataset")
```

Table 2: MAE values of three models dependent on simulated dataset

Simulated data	TwoModel MAE	Uplift Forest MAE	Class MAE
10% treatment, 20% control	0.1584105	0.1853349	0.2515624
20% treatment 80% control	0.1350670	0.1707161	0.2437861
30% treatment 70% control	0.1264400	0.1530570	0.2283331
40% treatment, 60% control	0.1266484	0.1505748	0.2250364
60% treatment, 40% control	0.1389875	0.1576035	0.2240221
70% treatment, 30% control	0.1316140	0.1605344	0.2063678
80% treatment 20% control	0.1384107	0.1585081	0.2005763
90% treatment 10% control	0.1412500	0.1777551	0.1944865
50% treatment 50% control	0.1198635	0.1542940	0.1715069

Answer: For most of the unbalanced sets, balanced set MAE was better regardless of the model, however, in two occasion (30%-40% treatments) tree based MAE outperformed balanced MAE. It is noticeable that two model and tree based MAE are in close range to balanced one while for Class transformation method, MAE is significantly higher compared to the balanced one. In general, the more imbalanced groups are, the bigger MAE measure becomes, however there are small MAE fluctuations in 30-40, and 60-80% treatment groups.