#1. load library####

library(dplyr)

library(readxl)

library(na.tools)

library(caret)

library(openxlsx)

library("glmnet")

library(pROC)

library(xgboost)

library(Matrix)

library(dplyr)

library(Boruta)

library(randomForestSRC)

library(rms)

# rm(list = ls())

#2. Read Radiomic Data####

# Load Radiomic Data

develop<-read.csv("develop\_01.csv")

validation<-read.csv("validation\_01.csv")

colnames(develop)[1] <-"id"

colnames(develop)[2] <-"Label"

colnames(validation)[1] <-"id"

colnames(validation)[2] <-"Label"

#3. DATA Tidying####

# Data Tidying based on CustomLabel

radiomics\_end\_index <- length(develop)

radiomics\_full <- develop

radiomics\_full\_v<-validation

Feature\_full<-develop[3:length(develop)]

Feature\_full\_v<-validation[3:length(validation)]

Feature\_full\_v[1,0:2]

#4. Extract Features####

Features\_develop <- Feature\_full[1:length(Feature\_full)]

Features\_validation <- Feature\_full\_v[1:length(Feature\_full\_v)]

Labels <- as.factor(radiomics\_full$Label)

Labels\_v <- as.factor(radiomics\_full\_v$Label)

id\_v<-validation$id

id\_d<-develop$id

which\_na(develop)

which\_na(validation)

data\_Develop\_ori <- cbind(Labels, Features\_develop)

data\_Develop\_ori\_full <- cbind(id\_d,Labels, Features\_develop)

data\_validation\_ori <- cbind(Labels\_v, Features\_validation)

data\_validation\_ori\_full <- cbind(id\_v,Labels\_v, Features\_validation)

total\_deve\_id<-data.frame(unique(data\_Develop\_ori\_full$id\_d))

sized1= round(0.7\*length(total\_deve\_id[,1]))

sized2=length(total\_deve\_id[,1])-sized1

#5. Boruta+RandomForest: model construction####

library(caret)

library(openxlsx)

library("glmnet")

library(pROC)

library(xgboost)

library(Matrix)

library(dplyr)

library(Boruta)

seed <- NULL

counter <- 1

currtree <- NULL

currsize <- NULL

currdepth <- NULL

group <- NULL

train\_result <- NULL

train\_se <- NULL

train\_sp <- NULL

test\_result <- NULL

test\_se <- NULL

test\_sp <- NULL

currseed <- NULL

currboruta <- NULL

currsplit <- NULL

for (cboruta in c(floor(runif(10)\*1000))){

for (csplit in c(floor(runif(10)\*1000))){

set.seed(csplit)

train\_split\_id<-sample(total\_deve\_id[,1], sized1, replace = FALSE, prob = NULL)

train\_split\_id

train\_full<-data\_Develop\_ori\_full[which(data\_Develop\_ori\_full$id\_d %in% train\_split\_id),]

train\_feature<-train\_full[5:length(train\_full)]

y\_train <- as.numeric(train\_full$Labels)

y\_train <- train\_full$Labels

train\_feature\_full<-cbind(y\_train,train\_feature)

test\_full<-data\_Develop\_ori\_full[-which(data\_Develop\_ori\_full$id\_d %in% train\_split\_id),]

test\_feature<-test\_full[5:length(test\_full)]

y\_test <- as.numeric(test\_full$Labels)

y\_test <- test\_full$Labels

test\_feature\_full<-cbind(y\_test,test\_feature)

set.seed(cboruta)

boruta\_train <- Boruta(y\_train~., data = train\_feature, doTrace = 0,maxRuns=15)

boruta\_df <- attStats(boruta\_train)

boruta\_features<-boruta\_df[order(abs(boruta\_df$meanImp),decreasing=TRUE),]

Selected\_Features\_train<-train\_feature %>%dplyr:::select(rownames(boruta\_features))

Selected\_Features\_test <-test\_feature %>%dplyr:::select(rownames(boruta\_features))

SELECTDATA\_train <- cbind(y\_train, Selected\_Features\_train)

SELECTDATA\_test <- cbind(y\_test, Selected\_Features\_test)

library(randomForestSRC)

for (groupindex in c(1:5)){

for (tree in 1:10){

for (size in 1:10){

for (depth in 1:5){

currseed[counter]<-seedindex

currsplit[counter] <- csplit

currboruta[counter] <- cboruta

currtree[counter] <- tree

currsize[counter] <- size

currdepth[counter] <- depth

group[counter] <- groupindex

set.seed(groupindex)

rf <- rfsrc(y\_train~., data =SELECTDATA\_train, ntree=tree, nodesize =size,nodedepth = depth,importance = TRUE)

train\_class <- predict.rfsrc(rf,SELECTDATA\_train)$predicted[,1]

Train\_roc\_data <- data.frame(y\_train,train\_class)

train\_result[counter] <- roc(Train\_roc\_data[,1], Train\_roc\_data[,2])$auc

train\_se[counter] <-coords (roc(Train\_roc\_data[,1], Train\_roc\_data[,2]),

"best")$sensitivity

train\_sp[counter] <-coords (roc(Train\_roc\_data[,1], Train\_roc\_data[,2]),

"best")$specificity

test\_class <- predict.rfsrc(rf,SELECTDATA\_test)$predicted[,1]

Test\_roc\_data <- data.frame(y\_test,test\_class)

test\_result[counter] <- roc(Test\_roc\_data[,1], Test\_roc\_data[,2])$auc

test\_se[counter] <-coords(roc(Test\_roc\_data[,1], Test\_roc\_data[,2]),coords (roc(Train\_roc\_data[,1], Train\_roc\_data[,2]),"best")$threshold)$sensitivity

test\_sp[counter] <-coords(roc(Test\_roc\_data[,1], Test\_roc\_data[,2]),coords (roc(Train\_roc\_data[,1], Train\_roc\_data[,2]),"best")$threshold)$specificity

counter <- counter + 1

}

}

}

}

}

}

summary <- data.frame(currsplit,currboruta,group, currtree,currsize,currdepth,train\_result,test\_result,train\_se,test\_se,train\_sp,test\_sp)

bestparam <- summary %>% filter(train\_result > 0.9)%>%filter(test\_result > 0.9)

bestparam[dim(bestparam)[1],] -> bestparam

bestparam

#6. Applying Boruta + RandomForest####

set.seed(bestparam$currsplit)

train\_split\_id<-sample(total\_deve\_id[,1], sized1, replace = FALSE, prob = NULL)

train\_full<-data\_Develop\_ori\_full[which(data\_Develop\_ori\_full$id\_d %in% train\_split\_id),]

train\_feature<-train\_full[5:length(train\_full)]

y\_train <- as.numeric(train\_full$Labels)

y\_train <- train\_full$Labels

train\_feature\_full<-cbind(y\_train,train\_feature)

test\_full<-data\_Develop\_ori\_full[-which(data\_Develop\_ori\_full$id\_d %in% train\_split\_id),]

test\_feature<-test\_full[5:length(test\_full)]

y\_test <- as.numeric(test\_full$Labels)

y\_test <- test\_full$Labels

test\_feature\_full<-cbind(y\_test,test\_feature)

y\_validation <-as.numeric(data\_validation\_ori\_full$Labels\_v)

y\_validation <-data\_validation\_ori\_full$Labels\_v

set.seed(bestparam$currboruta)

boruta\_train <- Boruta(y\_train~., data = train\_feature, doTrace = 0,maxRuns=15)

boruta\_df <- attStats(boruta\_train)

boruta\_features<-boruta\_df[order(abs(boruta\_df$meanImp),decreasing=TRUE),]

Selected\_Features\_train <-train\_feature %>%dplyr:::select(rownames(boruta\_features))

Selected\_Features\_test <-test\_feature %>%dplyr:::select(rownames(boruta\_features))

Selected\_Features\_v <- Features\_validation%>% dplyr:::select(rownames(boruta\_features))

SELECTDATA\_validation <- cbind(y\_validation, Selected\_Features\_v)

SELECTDATA\_train <- cbind(y\_train, Selected\_Features\_train)

SELECTDATA\_test <- cbind(y\_test, Selected\_Features\_test)

set.seed(bestparam$group)

rf <- rfsrc(y\_train~ ., data =SELECTDATA\_train, ntree=bestparam$currtree, nodesize=bestparam$currsize,nodedepth = bestparam$currdepth,importance = TRUE)

rf$predicted

rf$class

library(ggRandomForests)

gg0 <- gg\_vimp(rf)

gg<-gg0[gg0$vimp!=0,]

gg<-gg[gg$set=="all",]

plot(gg)

train\_class <- predict.rfsrc(rf,SELECTDATA\_train)$predicted[,1]

test\_class <- predict.rfsrc(rf,SELECTDATA\_test)$predicted[,1]

validation\_class <- predict.rfsrc(rf,SELECTDATA\_validation)$predicted[,1]

vali\_full<-cbind("id\_d"=validation$id,"Labels"=validation$Label,Feature\_full\_v)

data\_train<-cbind(train\_full,train\_class)

data\_test<-cbind(test\_full,test\_class)

data\_vali<-cbind(vali\_full,validation\_class)

data\_train <- transform(data\_train, group = 0)

data\_test <- transform(data\_test, group = 1)

data\_vali <- transform(data\_vali, group = 2)

write.csv(data\_train,"Train data.csv")

write.csv(data\_test,"Test\_dataf.csv")

write.csv(data\_vali,"validation\_data.csv")

#Train roc data

library("pROC")

Train\_roc\_data <- data.frame(y\_train,train\_class)

names(Train\_roc\_data)<- c("y\_train", "rfsrc\_pred")

rocobj\_train <-roc(Train\_roc\_data$y\_train, Train\_roc\_data$rfsrc\_pred)

print("Training Dataset")

rocobj\_train$auc

ci.auc(rocobj\_train)##CI

roc\_summary\_train <-coords (rocobj\_train,

"best"

, ret=c ("threshold", "specificity", "sensitivity", "accuracy","precision","recall","ppv","npv"),transpose = TRUE)

roc\_summary\_train

roc\_summary\_train1 <-coords (rocobj\_train,

"best")$specificity

# Test roc data

library("pROC")

Test\_roc\_data <- data.frame(y\_test,test\_class)

names(Test\_roc\_data)<- c("y\_test", "rfsrc\_pred\_test")

rocobj\_test <- roc(Test\_roc\_data[,1], Test\_roc\_data[,2])

print("Testing Dataset")

rocobj\_test$auc

ci.auc (rocobj\_test)##CI

roc\_summary\_test<-coords (rocobj\_test,roc\_summary\_train[1], ret=c ("threshold", "specificity", "sensitivity", "accuracy","precision","recall","ppv","npv"),transpose = TRUE)

roc.test (rocobj\_train, rocobj\_test, method= "delong")

roc\_summary\_test

# Validation roc data

library("pROC")

Validation\_roc\_data <- data.frame(y\_validation,validation\_class)

names(Validation\_roc\_data)<- c("y\_validation", "rfsrc\_pred\_validation")

rocobj\_validation <- roc(Validation\_roc\_data$y\_validation, Validation\_roc\_data$rfsrc\_pred\_validation)

print("Validation Dataset")

rocobj\_validation$thresholds

roc\_summary\_vali<-coords (rocobj\_validation,roc\_summary\_train[1], ret=c ("threshold", "specificity", "sensitivity", "accuracy","precision","recall","ppv","npv"),transpose = TRUE)

roc.test (rocobj\_train, rocobj\_validation, method= "delong")