zg21696\_SectionC\_code

Ruinan Wang

13/12/2021

## Importing Data and pretreatment

breast\_cancer\_ds <- read.csv("data.csv", header = TRUE)  
## The column id and X don't affect the prediction, so delete them from dataset  
breast\_cancer\_ds <- breast\_cancer\_ds %>% select(-c(id,X))   
## The number of features and samples  
dim(breast\_cancer\_ds)

## [1] 569 31

## Convert the value of the categorical variable "diagnosis" to a numerical value where M = 1 and B = 0.  
breast\_cancer\_ds$diagnosis[which(breast\_cancer\_ds$diagnosis == 'M')] <- 1  
breast\_cancer\_ds$diagnosis[which(breast\_cancer\_ds$diagnosis == 'B')] <- 0  
  
## The magnitude for some features have a significant gap, like "radius\_mean" and "area\_mean"  
head(select(breast\_cancer\_ds,symmetry\_mean,area\_mean),3)

## symmetry\_mean area\_mean  
## 1 0.2419 1001  
## 2 0.1812 1326  
## 3 0.2069 1203

## The feature-scaling is necessary  
## here we used normalization through creating a function  
standardization <- function (x){  
 return((x-mean(x))/(sd(x)))  
}  
bc\_df <- as.data.frame(lapply(breast\_cancer\_ds[2:31],standardization))  
breast\_cancer\_ds <- cbind(breast\_cancer\_ds[1],bc\_df)  
  
## Then the dataset need to do the train-validation-test split for the following RandomizedSearchCV task  
num\_total <- breast\_cancer\_ds %>% nrow()  
num\_test <- ceiling(0.25\*num\_total)  
set.seed(12)  
breast\_cancer\_ds <- breast\_cancer\_ds %>% sample\_n(size=nrow(.))  
test\_inds <- seq(num\_total-num\_test+1,num\_total)  
test\_data <- breast\_cancer\_ds %>% filter (row\_number()%in%test\_inds)  
train\_validation\_data <- breast\_cancer\_ds %>% filter(!row\_number() %in% test\_inds)

## Hyperparameter Tuning through RandomizedSearchCV

In this part, an external package named RandomSearchR by MOUSELIMIS is used for implementing RandomizedSearchCV to tune hyperparameter.

library(RandomSearchR)  
## list all choices of hyperparameter  
grid\_kknn = list(k = 3:20,   
 distance = 1:5,  
 kernel = c("rectangular", "triangular", "epanechnikov", "biweight", "triweight","cos", "inv", "gaussian", "rank", "optimal"))  
  
## make diagnosis results the numerical value  
diagnosis\_vec = train\_validation\_data[,1]  
diagnosis\_vec = c(1:length(unique(diagnosis\_vec)))[match(diagnosis\_vec, sort(unique(diagnosis\_vec)))]  
data\_vec = train\_validation\_data[,-1]  
## create a formula for the following random\_search\_resample function  
form <- as.formula(paste('diagnosis ~', paste(names(data\_vec),collapse = '+')))  
  
ALL\_DATA = train\_validation\_data  
ALL\_DATA$diagnosis = as.factor(diagnosis\_vec)  
  
## execute the random search CV  
res\_knn = random\_search\_resample(  
 as.factor(diagnosis\_vec),  
 tune\_iters = 30,  
 resampling\_method = list(method = 'cross\_validation',  
 repeats=NULL,   
 sample\_rate = NULL,  
 folds = 5),  
 ALGORITHM = list(package = require(kknn), algorithm = kknn),  
 grid\_params = grid\_kknn,  
 DATA = list(formula = form, train = ALL\_DATA),  
 Args = NULL,  
 regression = FALSE,  
 re\_run\_params=FALSE  
)

##   
## grid random search starts ..   
##   
## | | | 0% | |== | 3% | |===== | 7% | |======= | 10% | |========== | 14% | |============ | 17% | |============== | 21% | |================= | 24% | |=================== | 28% | |====================== | 31% | |======================== | 34% | |=========================== | 38% | |============================= | 41% | |=============================== | 45% | |================================== | 48% | |==================================== | 52% | |======================================= | 55% | |========================================= | 59% | |=========================================== | 62% | |============================================== | 66% | |================================================ | 69% | |=================================================== | 72% | |===================================================== | 76% | |======================================================== | 79% | |========================================================== | 83% | |============================================================ | 86% | |=============================================================== | 90% | |================================================================= | 93% | |==================================================================== | 97% | |======================================================================| 100%

acc = function(y\_true, preds) {   
 out = table(y\_true, max.col(preds, ties.method = "random"))  
 acc = sum(diag(out))/sum(out)  
 acc  
}  
  
perf = performance\_measures(list\_objects = list(kknn = res\_knn),  
 eval\_metric = acc,  
 sort = list(variable = 'Median', decreasing = TRUE))  
head(perf$test\_params$kknn,10)

## k distance kernel Min. 1st Qu. Median Mean 3rd Qu.  
## 10 8 3 gaussian 0.9294118 0.9647059 0.9651163 0.9694938 0.9882353  
## 21 8 3 gaussian 0.9294118 0.9647059 0.9651163 0.9694938 0.9882353  
## 1 20 5 cos 0.8941176 0.9534884 0.9647059 0.9554036 0.9764706  
## 2 6 1 triangular 0.9411765 0.9418605 0.9647059 0.9648427 0.9764706  
## 4 7 2 inv 0.9294118 0.9534884 0.9647059 0.9624624 0.9647059  
## 5 13 3 epanechnikov 0.9294118 0.9534884 0.9647059 0.9624624 0.9764706  
## 7 12 5 gaussian 0.8941176 0.9534884 0.9647059 0.9530506 0.9647059  
## 11 13 1 optimal 0.9294118 0.9418605 0.9647059 0.9601368 0.9764706  
## 12 4 1 rectangular 0.9411765 0.9534884 0.9647059 0.9695212 0.9882353  
## 13 4 2 inv 0.9294118 0.9418605 0.9647059 0.9577839 0.9647059  
## Max.  
## 10 1.0000000  
## 21 1.0000000  
## 1 0.9882353  
## 2 1.0000000  
## 4 1.0000000  
## 5 0.9882353  
## 7 0.9882353  
## 11 0.9882353  
## 12 1.0000000  
## 13 0.9882353