R programming Code

#load package

library(randomForest)

library(ggplot2)

library("caret")

library(lattice)

library(ggplot2)

library(kernlab)

library(caret)

data.all<- read.csv("F:\\data\\shunxu1.0.csv")

set.seed(4)

a<-c()

for(i in 1:10){

traindata<- data.all[-folds[[i]],]

testdata<- data.all[folds[[i]],]

# Random Forest

# hyper-parameters tuning

for(j in 2:6){

for(k in seq(from=500,to=2000,by=250)){

m1=randomForest(Y~.,traindata,mtry=j,ntree=k,proximity=TRUE, importance=TRUE)

result<-c(i,j,k,sum(diag(table(observed=testdata$Y,predicted=predict(m1, testdata))))/sum(table(observed=testdata$Y,predicted=predict(m1, testdata))))

a<-rbind(a,result)

}}}

a[which(a[,4]==max(a[,4])),]

#10-folds cross-validation

folds<-createFolds(y=data.all$Y,k=10)

re1<-{}

re2<-{}

re3<-{}

for(i in 1:10){

traindata<- data.all[-folds[[i]],]

testdata<- data.all[folds[[i]],]

m2<-randomForest(Y~.,traindata,mtry=3,ntree=1500,proximity=TRUE, importance=TRUE)

re1=c(re1,length(testdata$Y[which(predict(m2,newdata=testdata)== testdata$Y)])/length(testdata$Y))

re2=c(re2,length(traindata$Y[which(predict(m2,newdata=traindata)== traindata$Y)])/length(traindata$Y))

re3=c(re3,length(data.all$Y[which(predict(m2,newdata=data.all)==data.all$Y)])/length(data.all$Y))

}

mean(re1)

re1

mean(re2)

re2

mean(re3)

re3

#Training model

traindata<- data.all[-folds[[2]],]

testdata<- data.all[folds[[2]],]

m2<-randomForest(Y~.,traindata,mtry=3,ntree=1500,proximity=TRUE, importance=TRUE)

#Test dataset-based prediction

pre.te <-predict(m2,testdata)

table(observed=testdata$Y,predicted= pre.te)

sum(diag(table(observed=testdata$Y,predicted=pre.te)))/sum(table(observed=testdata$Y,predicted= pre.te))

#Train dataset-based prediction

pre.tr<- predict(m2,traindata)

table(observed=traindata$Y,predicted=pre.tr)

sum(diag(table(observed=traindata$Y,predicted=pre.tr)))/sum(table(observed=traindata$Y,predicted=pre.tr))

#Overall dataset-based prediction

pre.all <-predict(m2, data.all)

table(observed= data.all$Y,predicted=pre.all)

sum(diag(table(observed=data.all$Y,predicted=pre.all)))/sum(table(observed=data.all$Y,predicted=pre.all))

#Factors ranking

importance<-importance(x=m2)

importance

set.seed(4)

varImpPlot(m2)

# SVM

# hyper-parameters tuning

set.seed(4)

a<-c()

for(i in 1:10){

traindata<-fff[-folds[[i]],]

testdata<-fff[folds[[i]],]

for(j in seq(from=0.1,to=0.5,by=0.1)){

for(k in 1:15){

m1<-ksvm(Y~.,data=traindata, type = "C-svc", kernel="rbfdot", kpar=list(sigma=j),C= k, prob.model = TRUE)

result<-c(i,j,k,sum(diag(table(observed=testdata$Y,predicted=predict(m1, testdata))))/sum(table(observed=testdata$Y,predicted=predict(m1, testdata))))

a<-rbind(a,result)

}}}

a[which(a[,4]==max(a[,4])),]

#Training model

traindata<-fff[-folds[[7]],]

testdata<-fff[folds[[7]],]

m2<-ksvm(Y~.,data=traindata,type="C-svc",kernel="rbfdot",kpar=list(sigma=0.1),C =5, prob.model = TRUE)

#Test dataset-based prediction

pre.te <-predict(m2,testdata)

table(observed=testdata$Y,predicted= pre.te)

sum(diag(table(observed=testdata$Y,predicted=pre.te)))/sum(table(observed=testdata$Y,predicted= pre.te))

#Train dataset-based prediction

pre.tr<- predict(m2,traindata)

table(observed=traindata$Y,predicted=pre.tr)

sum(diag(table(observed=traindata$Y,predicted=pre.tr)))/sum(table(observed=traindata$Y,predicted=pre.tr))

#Overall dataset-based prediction

pre.all <-predict(m2,fff)

table(observed=fff$Y,predicted=pre.all)

sum(diag(table(observed=fff$Y,predicted=pre.all)))/sum(table(observed=fff$Y,predicted=pre.all))