Allen Phelps and Dave Hiltbrand

Dr. Cuixian Chen

STT 592

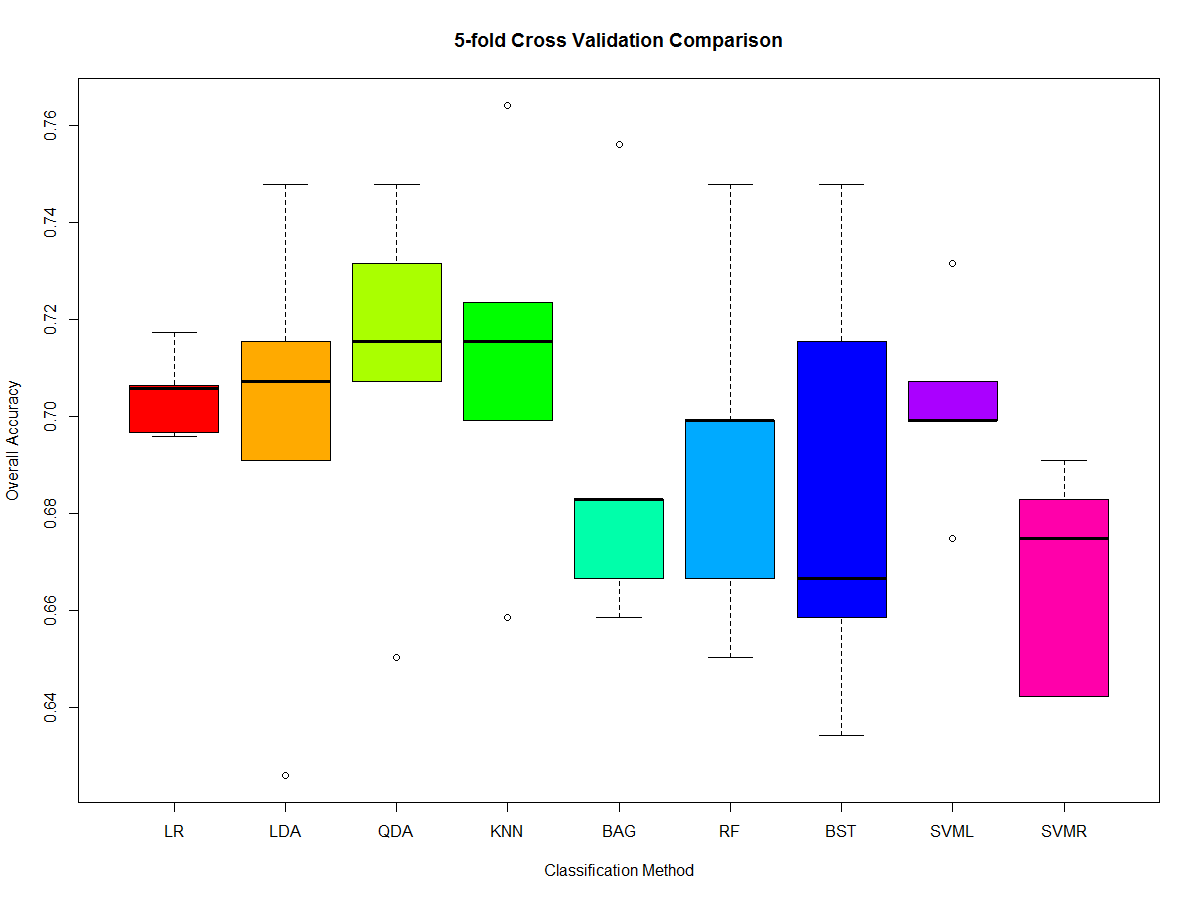
8 November 2016

Group Report 6

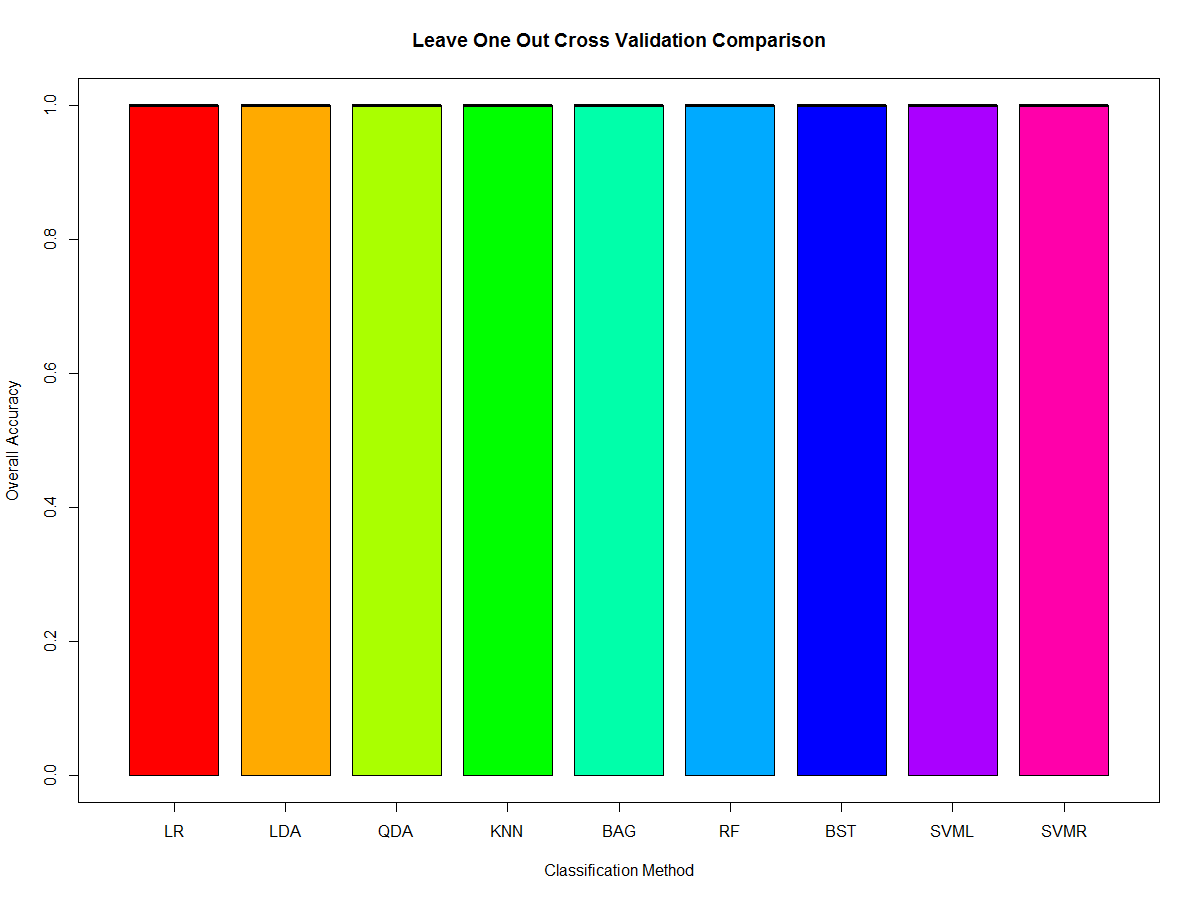
Thus far our data mining project has given our group seven types of classification modeling to predict whether individuals from the TBI data set will be diagnosed with PTSD by a clinician by undergoing a series of evaluations such as the Trauma Symptom Inventory (TSI) and Minnesota Multiphasic Personality Inventory (MMPI). In keeping with the best practice of research, we again try another method of modeling known as Support Vector Machines (SVM).

The basic idea of support vector machines is to divide observations into separable classes with linear, polynomial, or radial methods. Simple data can often be separated by a straight line (linear) but it can come at a cost. In developing our best model to use with support vector machines we ran both Leave-One-Out Cross Validation and K-Fold 5 Cross Validation while using the built in tuning function of the “e1071” package. Information leaking can occur if the data set is tuned before cross validation begins so to prevent seeing the leak we ran our tuning function inside a “for” loop for each type of cross validation. The tuning function itself seeks to find the best cost parameter for linear SVM and both cost and gamma parameters for radial SVM. We created a vector to capture the best cost and gamma associated with each round of tests to allow us the ability to create future models with the best parameters.

The results indicated linear SVM supports our data better with an accuracy of 71.22% and standard deviation of 0.02040634 while radial SVM resulted in an accuracy of 66.67% and standard deviation of 0.471800. As we had seen previously within logistic regression and linear discriminant analysis we have good separation among our classes and this again is apparent with the results higher linear SVM accuracy. One discovery made during this round of modeling as displayed in Table 3 is the gross amount of time involved in running certain models when using LOOCV. The test for LOOCV Radial SVM took our computer over 8 hours to run which must be taken into consideration when making adjustments to parameters. This supports our decision to run the tuning function inside the validation loop for efficiency.



**Figure 1**. The Boxplot displays the prediction accuracy, from 0 to 1, for all nine classifiers applying 5-fold CV: Logistic Regression, LDA, QDA, K-NN, Bagging, Random Forest, Boosting, Linear Support Vector Machines and Radial Support Vector Machines.



**Figure 2**. The Boxplot displays the prediction accuracy, from 0 to 1, for all nine classifiers applying Leave-One-Out CV: Logistic Regression, LDA, QDA, K-NN, Bagging, Random Forest, Boosting, Linear Support Vector Machines and Radial Support Vector Machines.

|  |  |  |
| --- | --- | --- |
| Classification Method (5-fold CV) | Mean | SD |
| Logistic Regression | 0.7190422 | 0.01809779 |
| Linear Discriminant Analysis | 0.7089431 | 0.01941030 |
| Quadratic Discriminant Analysis | 0.6813008 | 0.01563852 |
| K Nearest Neighbors | 0.7024390 | 0.03752189 |
| Bagging | 0.6780488 | 0.03477927 |
| Random Forest | 0.7056911 | 0.03009229 |
| Boosting | 0.7121951 | 0.03180099 |
| Linear Support Vector Machines | 0.7024390 | 0.02040634 |
| Radial Support Vector Machines | 0.6666667 | 0.02299534 |

**Table 1**. The prediction accuracy means and standard deviations for 5-fold Cross Validation for all nine classifiers.

|  |  |  |
| --- | --- | --- |
| Classification Method (LOOCV) | Mean | SD |
| Logistic Regression | 0.6294581 | 0.4829513 |
| Linear Discriminant Analysis | 0.7089431 | 0.4546195 |
| Quadratic Discriminant Analysis | 0.6813008 | 0.4663514 |
| K Nearest Neighbors | 0.7056911 | 0.4561025 |
| Bagging | 0.6829268 | 0.4657150 |
| Random Forest | 0.6975610 | 0.4596882 |
| Boosting | 0.6991870 | 0.4589848 |
| Linear Support Vector Machines | 0.71219510 | 0.4531082 |
| Radial Support Vector Machines | 0.6667000 | 0.471800 |

**Table 2**. The prediction accuracy means and standard deviations for Leave-One-Out Cross Validation for all nine classifiers.

|  |  |  |
| --- | --- | --- |
|  | **Time Elapsed** | |
| **Classification Method** | *5-fold CV* | *LOOCV* |
| *Logistic Regression* | 0.0880 s | 10.6868 s |
| *Linear Discriminant Analysis* | 2.6595 s | 0.0655 s |
| *Quadratic Discriminant Analysis* | 2.6443 s | 0.0620 s |
| *K Nearest Neighbors* | 0.9521 s | 0.0691 s |
| *Bagging* | 4.1257 s | 11.3656 m |
| *Random Forest* | 3.7435 s | 9.3066 m |
| *Boosting* | 2.7221 s | 6.2322 m |
| *Linear Support Vector Machines* | 1.0427 m | 2.9414 h |
| *Radial Support Vector Machines* | 2.6971 m | 8.1302 h |

**Table 3**. The time elapsed to run each model in R as stated in minutes or hours.

**Appendix: R Code**

install.packages('e1071')

install.packages('gbm')

install.packages('klaR')

install.packages('leaps')

install.packages('nortest')

install.packages('randomForest')

install.packages('tree')

library(boot)

library(class)

library(e1071)

library(gbm)

library(klaR)

library(leaps)

library(MASS)

library(nortest)

library(randomForest)

library(tree)

########################################

#Subset data for Logistic Regression/KNN

########################################

dat=read.csv("stt592dat.csv", header=T)

names(dat)

dat=data.frame(dat[,13],dat[,22],dat[,26],dat[,22]\*dat[,26])

dat=dat[complete.cases(dat),]

names(dat)=c('ptsd','aa','da','aada')

##########################

#Logistic Regression LOOCV

##########################

begin=Sys.time()

n=dim(dat)[1]

set.seed(1)

foldi=sample(rep(1:n,length.out=n))

table(foldi)

lr1out=NULL

for(k in 1:n)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[!train,]

ytrain=glm(ptsd~aa\*da,data=dat,family=binomial)

ptrain=predict(ytrain,test[,2:4],type='response')

ptrain=ifelse(ptrain<=0.5,FALSE,TRUE)

accuracy=(ptrain==test[,1])

lr1out=c(lr1out,accuracy)

}

print(lr1out)

mean(lr1out) #0.6097

sd(lr1out) #0.4878

boxplot(lr1out,col='green')

end=Sys.time()

time=end-begin; time #10.68677 secs

##############################

#Logistic Regression 5-fold CV

##############################

begin=Sys.time()

nfolds=5

set.seed(1)

foldi=sample(rep(1:nfolds,length.out=dim(dat)[1]))

table(foldi)

lr5out=NULL

for(k in 1:nfolds)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[!train,]

ytrain=glm(ptsd~aa\*da,data=dat,family=binomial)

ptrain=predict(ytrain,test[,2:4],type='response')

ptrain=ifelse(ptrain<=0.5,FALSE,TRUE)

accuracy=mean(ptrain==test[,1])

lr5out=c(lr5out,accuracy)

}

print(lr5out)

mean(lr5out) #0.7045

sd(lr5out) #0.0088

boxplot(lr5out,col='lightyellow',main='5-fold Cross Validation',ylab='Overall Accuracy')

end=Sys.time()

time=end-begin;time #0.08801389 secs

##########

#KNN LOOCV

##########

begin=Sys.time()

set.seed(1)

folds\_i <- sample(rep(1:n, length.out = n))

table(folds\_i)

OUT.KNN1=NULL

for (j in 1:n)

{

test.ID <- which(folds\_i == j)

train\_X <- dat[-test.ID, c("aa","da",'aada')]

train\_Y <- dat[-test.ID, 1]

test\_X <- dat[test.ID, c("aa","da",'aada')]

test\_Y <- dat[test.ID, 1]

knn.pred=knn(train\_X, test\_X, train\_Y, k=12)

table(knn.pred,test\_Y)

Accuracy=mean(knn.pred==test\_Y)

OUT.KNN1=c(OUT.KNN1, Accuracy)

}

print(OUT.KNN1)

mean(OUT.KNN1) #0.6894

sd(OUT.KNN1) #0.4631

boxplot(OUT.KNN1,col="orange")

end=Sys.time()

time=end-begin;time #0.952101 secs

##############

#KNN 5-fold CV

##############

begin=Sys.time()

n\_fold<-5;

rep(1:n\_fold, length.out = n)

set.seed(1)

folds\_i <- sample(rep(1:n\_fold, length.out = n))

table(folds\_i)

OUT.KNN=NULL

for (j in 1:n\_fold)

{

test.ID <- which(folds\_i == j)

train\_X <- dat[-test.ID, c("aa","da",'aada')]

train\_Y <- dat[-test.ID, 1]

test\_X <- dat[test.ID, c("aa","da",'aada')]

test\_Y <- dat[test.ID, 1]

knn.pred=knn(train\_X, test\_X, train\_Y, k=12)

table(knn.pred,test\_Y)

Accuracy=mean(knn.pred==test\_Y)

OUT.KNN=c(OUT.KNN, Accuracy)

}

print(OUT.KNN)

mean(OUT.KNN) #0.7122

sd(OUT.KNN) #0.0384

boxplot(OUT.KNN,col="lavender",main='5-fold Cross Validation',ylab='Overall Accuracy')

end=Sys.time()

time=end-begin;time #0.06912684 secs

########################

#Subset Data for LDA/QDA

########################

dat=read.csv("stt592dat.csv", header=T)

names(dat)

dat=data.frame(dat[,13],dat[,25],dat[,25])

dat=dat[complete.cases(dat),]

names(dat)=c('ptsd','ie','iedummy')

##########

#LDA LOOCV

##########

begin=Sys.time()

n=dim(dat)[1]

set.seed(1)

foldi=sample(rep(1:n,length.out=dim(dat)[1]))

table(foldi)

lda1out=NULL

for(k in 1:n)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

l=lda(ptsd~ie,data=dat)

lpred=predict(l,test[,2:3])

lclass=lpred$class

table(lclass,test[,1])

accuracy=mean(lclass==test[,1])

lda1out=c(lda1out,accuracy)

}

print(lda1out)

mean(lda1out) #0.6976

sd(lda1out) #0.4597

boxplot(lda1out,col='green')

end=Sys.time()

time=end-begin;time #2.659549 secs

##############

#LDA 5-fold CV

##############

begin=Sys.time()

nfolds=5

set.seed(1)

foldi=sample(rep(1:nfolds,length.out=dim(dat)[1]))

table(foldi)

lda5out=NULL

for(k in 1:nfolds)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

l=lda(ptsd~ie,data=dat)

lpred=predict(l,test[,2:3])

lclass=lpred$class

table(lclass,test[,1])

accuracy=mean(lclass==test[,1])

lda5out=c(lda5out,accuracy)

}

print(lda5out)

mean(lda5out) #0.6976

sd(lda5out) #0.0450

boxplot(lda5out,col='lightgreen',main='5-fold Cross Validation',ylab='Overall Accuracy')

end=Sys.time()

time=end-begin;time #0.06552315 secs

##########

#QDA LOOCV

##########

begin=Sys.time()

n=dim(dat)[1]

set.seed(1)

foldi=sample(rep(1:n,length.out=dim(dat)[1]))

table(foldi)

qda1out=NULL

for(k in 1:n)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

q=qda(ptsd~ie,data=dat)

qpred=predict(q,test[,2:3])

qclass=qpred$class

table(qclass,test[,1])

accuracy=mean(qclass==test[,1])

qda1out=c(qda1out,accuracy)

}

print(qda1out)

mean(qda1out) #0.7106

sd(qda1out) #0.4539

boxplot(qda1out,col='green')

end=Sys.time()

time=end-begin;time #2.644317 secs

##############

#QDA 5-fold CV

##############

begin=Sys.time()

nfolds=5

set.seed(1)

foldi=sample(rep(1:nfolds,length.out=dim(dat)[1]))

table(foldi)

qda5out=NULL

for(k in 1:nfolds)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

q=qda(ptsd~ie,data=dat)

qpred=predict(q,test[,2:3])

qclass=qpred$class

table(qclass,test[,1])

accuracy=mean(qclass==test[,1])

qda5out=c(qda5out,accuracy)

}

print(qda5out)

mean(qda5out) #0.7106

sd(qda5out) #0.0371

boxplot(qda5out,col='lightblue',main='5-fold Cross Validation',ylab='Overall Accuracy')

end=Sys.time()

time=end-begin;time #0.0619719 secs

####################################################

#Subset Data for Decision Tree/Bagging/Random Forest

####################################################

dat=read.csv("stt592dat.csv", header=T)

names(dat)

ptsd=ifelse(dat[,13]==TRUE,'Yes','No')

dat=data.frame(ptsd,dat[,19:31])

dat=dat[complete.cases(dat),]

##############

#Bagging LOOCV

##############

begin=Sys.time()

n=dim(dat)[1]

set.seed(1)

foldi=sample(rep(1:n,length.out=n))

table(foldi)

bag1out=NULL

for(k in 1:n)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

ybag=randomForest(ptsd~.,data=train,mtry=13,ntree=500,importance=T)

bagpred=predict(ybag,test)

accuracy=mean(bagpred==test[,1])

bag1out=c(bag1out,accuracy)

}

print(bag1out)

mean(bag1out) #0.6894

sd(bag1out) #0.4631

boxplot(bag1out,col='green')

end=Sys.time()

time=end-begin;time #11.36555 mins

##################

#Bagging 5-fold CV

##################

begin=Sys.time()

nfolds=5

set.seed(1)

foldi=sample(rep(1:nfolds,length.out=dim(dat)[1]))

table(foldi)

bag5out=NULL

for(k in 1:nfolds)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

ybag=randomForest(ptsd~.,data=train,mtry=13,ntree=500,importance=T)

bagpred=predict(ybag,test)

accuracy=mean(bagpred==test[,1])

bag5out=c(bag5out,accuracy)

}

print(bag5out)

mean(bag5out) #0.6894

sd(bag5out) #0.0387

boxplot(bag5out,col='green',main='5-fold Cross Validation',ylab='Overall Accuracy')

varImpPlot(ybag,type=1)

end=Sys.time()

time=end-begin;time #4.125689 secs

####################

#Random Forest LOOCV

####################

begin=Sys.time()

n=dim(dat)[1]

set.seed(1)

foldi=sample(rep(1:n,length.out=n))

table(foldi)

rf1out=NULL

for(k in 1:n)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

yrf=randomForest(ptsd~.,data=train,mtry=4,ntree=500,importance=T)

rfpred=predict(yrf,test)

accuracy=mean(rfpred==test[,1])

rf1out=c(rf1out,accuracy)

}

print(rf1out)

mean(rf1out) #0.6894

sd(rf1out) #0.4631

boxplot(rf1out,col='green')

end=Sys.time()

time=end-begin;time #9.30658 mins

########################

#Random Forest 5-fold CV

########################

begin=Sys.time()

nfolds=5

set.seed(1)

foldi=sample(rep(1:nfolds,length.out=dim(dat)[1]))

table(foldi)

rf5out=NULL

for(k in 1:nfolds)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

yrf=randomForest(ptsd~.,data=train,mtry=4,ntree=500,importance=T)

rfpred=predict(yrf,test)

accuracy=mean(rfpred==test[,1])

rf5out=c(rf5out,accuracy)

}

print(rf5out)

mean(rf5out) #0.6927

sd(rf5out) #0.0374

boxplot(rf5out,col='blue',main='5-fold Cross Validation',ylab='Overall Accuracy')

varImpPlot(yrf,type=1)

end=Sys.time()

time=end-begin;time #3.743508 secs

#########################

#Subset Data for Boosting

#########################

dat=read.csv("stt592dat.csv", header=T)

names(dat)

ptsd=ifelse(dat[,13]==TRUE,1,0)

dat=data.frame(ptsd,dat[,19:31])

dat=dat[complete.cases(dat),]

###############

#Boosting LOOCV

###############

begin=Sys.time()

n=dim(dat)[1]

set.seed(1)

foldi=sample(rep(1:n,length.out=n))

table(foldi)

boost1out=NULL

for(k in 1:n)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

yboost=gbm(ptsd~.,data=train,distribution='bernoulli',interaction.depth=2,n.trees=1000,shrinkage=0.01)

boostpred=predict(yboost,test,n.trees=1000,type='response')

boostpred=factor(ifelse(boostpred<=0.5,0,1))

accuracy=mean(boostpred==test[,1])

boost1out=c(boost1out,accuracy)

}

print(boost1out)

mean(boost1out) #0.7008

sd(boost1out) #0.4583

boxplot(boost1out,col='green')

end=Sys.time()

time=end-begin;time #6.23216 mins

###################

#Boosting 5-fold CV

###################

begin=Sys.time()

nfolds=5

set.seed(1)

foldi=sample(rep(1:nfolds,length.out=dim(dat)[1]))

table(foldi)

boost5out=NULL

for(k in 1:nfolds)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

yboost=gbm(ptsd~.,data=train,distribution='bernoulli',interaction.depth=2,n.trees=1000,shrinkage=0.01)

boostpred=predict(yboost,test,n.trees=1000,type='response')

boostpred=factor(ifelse(boostpred<=0.5,0,1))

accuracy=mean(boostpred==test[,1])

boost5out=c(boost5out,accuracy)

}

print(boost5out)

mean(boost5out) #0.6846

sd(boost5out) #0.0461

boxplot(boost5out,col='purple',main='5-fold Cross Validation',ylab='Overall Accuracy')

summary(yboost)

end=Sys.time()

time=end-begin;time #2.722134 secs

####################

#Subset Data for SVM

####################

dat=read.csv("stt592dat.csv", header=T)

names(dat)

ptsd=dat[,13]

dat=data.frame(ptsd,dat[,19:31])

dat=dat[complete.cases(dat),]

########################

#Linear Kernel SVM LOOCV

########################

begin=Sys.time()

n=dim(dat)[1]

set.seed(1)

foldi=sample(rep(1:n,length.out=n))

table(foldi)

svml1out=NULL

for(k in 1:n)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

set.seed(1)

ytune=tune(svm,as.factor(ptsd)~.,data=train,kernel="linear",ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))

ybest=ytune$best.model

ypred=predict(ybest,test)

accuracy=mean(ypred==test[,1])

svml1out=c(svml1out,accuracy)

}

print(svml1out)

mean(svml1out) #0.7121

sd(svml1out) #0.4531

boxplot(svml1out,col='green')

end=Sys.time()

time=end-begin;time #2.941356 hours

############################

#Linear Kernel SVM 5-fold CV

############################

begin=Sys.time()

nfolds=5

set.seed(1)

foldi=sample(rep(1:nfolds,length.out=n))

table(foldi)

svml5out=NULL

cbest=NULL

for(k in 1:nfolds)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

set.seed(1)

ytune=tune(svm,as.factor(ptsd)~.,data=train,kernel="linear",ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))

ybest=ytune$best.model

ypred=predict(ybest,test)

accuracy=mean(ypred==test[,1])

svml5out=c(svml5out,accuracy)

cost=ytune$best.parameters[[1]]

cbest=c(cbest,cost)

}

print(svml5out);print(cbest)

mean(svml5out)

sd(svml5out)

boxplot(svml5out,col='green')

end=Sys.time()

time=end-begin;time #1.042653 mins

########################

#Radial Kernel SVM LOOCV

########################

begin=Sys.time()

n=dim(dat)[1]

set.seed(1)

foldi=sample(rep(1:n,length.out=n))

table(foldi)

svmr1out=NULL

for(k in 1:n)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

set.seed(1)

ytune=tune(svm,as.factor(ptsd)~.,data=train,kernel="radial",ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100),

gamma=c(0.5,1,2,3,4)))

ybest=ytune$best.model

ypred=predict(ybest,test)

accuracy=mean(ypred==test[,1])

svmr1out=c(svmr1out,accuracy)

}

print(svmr1out)

mean(svmr1out)

sd(svmr1out)

boxplot(svmr1out,col='green')

end=Sys.time()

time=end-begin;time

############################

#Radial Kernel SVM 5-fold CV

############################

begin=Sys.time()

nfolds=5

set.seed(1)

foldi=sample(rep(1:nfolds,length.out=n))

table(foldi)

svmr5out=NULL

cbest=NULL

gbest=NULL

for(k in 1:nfolds)

{

testi=which(foldi==k)

train=dat[-testi,]

test=dat[testi,]

set.seed(1)

ytune=tune(svm,as.factor(ptsd)~.,data=train,kernel="radial",ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100),

gamma=c(0.5,1,2,3,4)))

ybest=ytune$best.model

ypred=predict(ybest,test)

accuracy=mean(ypred==test[,1])

svmr5out=c(svmr5out,accuracy)

cost=ytune$best.parameters[[1]];gamma=ytune$best.parameters[[2]]

cbest=c(cbest,cost);gbest=c(gbest,gamma)

}

print(svmr5out);print(cbest);print(gbest)

mean(svmr5out) #0.6667

sd(svmr5out) #0.0230

boxplot(svmr5out,col='green')

end=Sys.time()

time=end-begin;time #2.697095 mins

######################

#Side-by-side Boxplots

######################

boxplot(lr1out,lda1out,qda1out,OUT.KNN1,bag1out,rf1out,boost1out,svml1out,svmr1out,col=c(rainbow(9)),

main='Leave One Out Cross Validation Comparison',

names=c('LR','LDA','QDA','KNN','BAG','RF','BST','SVML','SVMR'),

xlab='Classification Method',ylab='Overall Accuracy')

boxplot(lr5out,lda5out,qda5out,OUT.KNN,bag5out,rf5out,boost5out,svml5out,svmr5out,col=c(rainbow(9)),

main='5-fold Cross Validation Comparison',names=c('LR','LDA','QDA','KNN','BAG','RF','BST','SVML','SVMR'),

xlab='Classification Method',ylab='Overall Accuracy')