Project 2

Jakob Jarecki

3/15/2021

knitr::opts\_chunk$set(echo = TRUE)

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

#Package which has dataset- BreastCancer  
require(mlbench)

## Loading required package: mlbench

## Warning: package 'mlbench' was built under R version 4.0.4

#frontload all packages for analysis  
library(caTools) #Used to split data into training and test data

## Warning: package 'caTools' was built under R version 4.0.3

library(caret) #Used for training and plotting models

## Warning: package 'caret' was built under R version 4.0.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.0.4

library(mice) #Used to remove NA value in dataset

## Warning: package 'mice' was built under R version 4.0.4

##   
## Attaching package: 'mice'

## The following object is masked from 'package:stats':  
##   
## filter

## The following objects are masked from 'package:base':  
##   
## cbind, rbind

library(klaR) #Used to implement naiveBayes classification algorithm

## Warning: package 'klaR' was built under R version 4.0.3

## Loading required package: MASS

library(e1071) #Used to implement support vector machine classification

## Warning: package 'e1071' was built under R version 4.0.3

library(rpart) #Used to implement tree algorithm

## Warning: package 'rpart' was built under R version 4.0.3

library(nnet) #Used to implement neural net classifier  
library(randomForest) #Used to implement Random Forest classifier

## Warning: package 'randomForest' was built under R version 4.0.3

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(dplyr) #Used in conversion of categorical data to numerical data

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':  
##   
## combine

## 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

library(hablar) #Used in conversion from factors to integers

## Warning: package 'hablar' was built under R version 4.0.4

##   
## Attaching package: 'hablar'

## The following object is masked from 'package:dplyr':  
##   
## na\_if

## The following object is masked from 'package:mice':  
##   
## squeeze

library(adabag) #used in ensemble techniques

## Warning: package 'adabag' was built under R version 4.0.4

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 4.0.3

## Loading required package: doParallel

## Warning: package 'doParallel' was built under R version 4.0.4

## Loading required package: iterators

## Warning: package 'iterators' was built under R version 4.0.3

## Loading required package: parallel

library(caretEnsemble)#used to make an ensemble algorithm/classifier

## Warning: package 'caretEnsemble' was built under R version 4.0.4

##   
## Attaching package: 'caretEnsemble'

## The following object is masked from 'package:ggplot2':  
##   
## autoplot

#Load dataset  
data(BreastCancer)  
#Check load   
head(BreastCancer)

## Id Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size  
## 1 1000025 5 1 1 1 2  
## 2 1002945 5 4 4 5 7  
## 3 1015425 3 1 1 1 2  
## 4 1016277 6 8 8 1 3  
## 5 1017023 4 1 1 3 2  
## 6 1017122 8 10 10 8 7  
## Bare.nuclei Bl.cromatin Normal.nucleoli Mitoses Class  
## 1 1 3 1 1 benign  
## 2 10 3 2 1 benign  
## 3 2 3 1 1 benign  
## 4 4 3 7 1 benign  
## 5 1 3 1 1 benign  
## 6 10 9 7 1 malignant

#Verify structure  
str(BreastCancer)

## 'data.frame': 699 obs. of 11 variables:  
## $ Id : chr "1000025" "1002945" "1015425" "1016277" ...  
## $ Cl.thickness : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 5 5 3 6 4 8 1 2 2 4 ...  
## $ Cell.size : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 1 1 2 ...  
## $ Cell.shape : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 2 1 1 ...  
## $ Marg.adhesion : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 5 1 1 3 8 1 1 1 1 ...  
## $ Epith.c.size : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 2 7 2 3 2 7 2 2 2 2 ...  
## $ Bare.nuclei : Factor w/ 10 levels "1","2","3","4",..: 1 10 2 4 1 10 10 1 1 1 ...  
## $ Bl.cromatin : Factor w/ 10 levels "1","2","3","4",..: 3 3 3 3 3 9 3 3 1 2 ...  
## $ Normal.nucleoli: Factor w/ 10 levels "1","2","3","4",..: 1 2 1 7 1 7 1 1 1 1 ...  
## $ Mitoses : Factor w/ 9 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 5 1 ...  
## $ Class : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...

summary(BreastCancer)

## Id Cl.thickness Cell.size Cell.shape Marg.adhesion  
## Length:699 1 :145 1 :384 1 :353 1 :407   
## Class :character 5 :130 10 : 67 2 : 59 2 : 58   
## Mode :character 3 :108 3 : 52 10 : 58 3 : 58   
## 4 : 80 2 : 45 3 : 56 10 : 55   
## 10 : 69 4 : 40 4 : 44 4 : 33   
## 2 : 50 5 : 30 5 : 34 8 : 25   
## (Other):117 (Other): 81 (Other): 95 (Other): 63   
## Epith.c.size Bare.nuclei Bl.cromatin Normal.nucleoli Mitoses   
## 2 :386 1 :402 2 :166 1 :443 1 :579   
## 3 : 72 10 :132 3 :165 10 : 61 2 : 35   
## 4 : 48 2 : 30 1 :152 3 : 44 3 : 33   
## 1 : 47 5 : 30 7 : 73 2 : 36 10 : 14   
## 6 : 41 3 : 28 4 : 40 8 : 24 4 : 12   
## 5 : 39 (Other): 61 5 : 34 6 : 22 7 : 9   
## (Other): 66 NA's : 16 (Other): 69 (Other): 69 (Other): 17   
## Class   
## benign :458   
## malignant:241   
##   
##   
##   
##   
##

#Correct NAs  
dataset\_impute <- mice(BreastCancer[,2:10], print = FALSE) #Removing NA values and ID(1st column) from dataset  
BreastCancer <- cbind(BreastCancer[,11, drop = FALSE], mice::complete(dataset\_impute, 1)) #Adding Target class to the imputed dataset without NA

#Split data into a training set and a holdout set  
set.seed(123)#Sets a seed so the split it reproducible  
dt = sort(sample(nrow(BreastCancer), nrow(BreastCancer)\*.6))#Selects 60% of rows  
train<-BreastCancer[dt,]#Moves 60% of all records into a training set  
test<-BreastCancer[-dt,]#Moves 40% of all records into a testing set

#Implements the support vector model  
mysvm <- svm(Class ~ ., train)  
mysvm.pred <- predict(mysvm, train)  
#Build confusion matrix  
swvmtable<-confusionMatrix(mysvm.pred, reference = train$Class)  
#Begin new table for comparison  
key<- c('Recall', 'Precision', 'F1 Measure')  
SVMMethod<-c(swvmtable$byClass["Recall"],swvmtable$byClass["Precision"],swvmtable$byClass["F1"])

#Implements the naive bayes classifier  
mynb <- NaiveBayes(Class ~ ., train)  
mynb.pred <- predict(mynb,train)

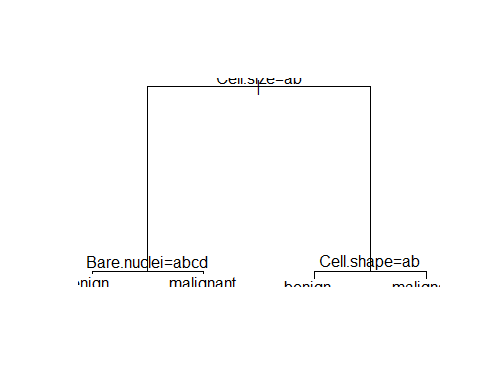
##

mynnet.pred <- predict(mynnet,train,type="class")  
#Build confusion matrix  
table(mynnet.pred,train$Class)

##   
## mynnet.pred benign malignant  
## benign 264 1  
## malignant 3 151

#Manual calculation of scores  
TP=as.numeric("151")  
FN=as.numeric("2")  
FP=as.numeric("1")  
TN=as.numeric("265")  
NNPr=(TP/(TP+FP))  
NNR = (TP/(TP+FN))  
NNF = (2\*NNPr\*NNR)/(NNPr+NNR)  
#Continue building rows for comparison table  
NNMethod<-c(NNR,NNPr,NNF)

#implements decision tree classifier  
mytree <- rpart(Class ~ ., train)  
  
plot(mytree); text(mytree) #Plots decision trees



summary(mytree)

## Call:  
## rpart(formula = Class ~ ., data = train)  
## n= 419   
##   
## CP nsplit rel error xerror xstd  
## 1 0.82236842 0 1.0000000 1.0000000 0.06474807  
## 2 0.03289474 1 0.1776316 0.1776316 0.03306544  
## 3 0.01315789 2 0.1447368 0.1578947 0.03129346  
## 4 0.01000000 3 0.1315789 0.1710526 0.03248870  
##   
## Variable importance  
## Cell.size Cell.shape Bare.nuclei Bl.cromatin Epith.c.size   
## 21 18 16 16 15   
## Normal.nucleoli   
## 15   
##   
## Node number 1: 419 observations, complexity param=0.8223684  
## predicted class=benign expected loss=0.3627685 P(node) =1  
## class counts: 267 152  
## probabilities: 0.637 0.363   
## left son=2 (254 obs) right son=3 (165 obs)  
## Primary splits:  
## Cell.size splits as LLRRRRRRRR, improve=144.9527, (0 missing)  
## Cell.shape splits as LLRRRRRRRR, improve=140.4064, (0 missing)  
## Bl.cromatin splits as LLLRRRRRRR, improve=126.2930, (0 missing)  
## Bare.nuclei splits as LLRRRRRRRR, improve=125.5699, (0 missing)  
## Epith.c.size splits as LLRRRRRRRR, improve=120.3733, (0 missing)  
## Surrogate splits:  
## Cell.shape splits as LLRRRRRRRR, agree=0.921, adj=0.800, (0 split)  
## Epith.c.size splits as LLRRRRRRRR, agree=0.897, adj=0.739, (0 split)  
## Bl.cromatin splits as LLLRRRRRRR, agree=0.895, adj=0.733, (0 split)  
## Bare.nuclei splits as LLRRRRRRRR, agree=0.888, adj=0.715, (0 split)  
## Normal.nucleoli splits as LLRRRRRRRR, agree=0.881, adj=0.697, (0 split)  
##   
## Node number 2: 254 observations, complexity param=0.01315789  
## predicted class=benign expected loss=0.02755906 P(node) =0.6062053  
## class counts: 247 7  
## probabilities: 0.972 0.028   
## left son=4 (244 obs) right son=5 (10 obs)  
## Primary splits:  
## Bare.nuclei splits as LLLLRRRR-R, improve=6.822370, (0 missing)  
## Bl.cromatin splits as LLLLR-R---, improve=3.243198, (0 missing)  
## Cell.shape splits as LLRRRRRRRR, improve=2.425849, (0 missing)  
## Cl.thickness splits as LLLLLRRRRR, improve=2.355220, (0 missing)  
## Epith.c.size splits as LLLRRRRRRR, improve=1.994255, (0 missing)  
## Surrogate splits:  
## Bl.cromatin splits as LLLLR-L---, agree=0.976, adj=0.4, (0 split)  
## Normal.nucleoli splits as LLLR-R---R, agree=0.976, adj=0.4, (0 split)  
## Mitoses splits as LLR---L--, agree=0.965, adj=0.1, (0 split)  
##   
## Node number 3: 165 observations, complexity param=0.03289474  
## predicted class=malignant expected loss=0.1212121 P(node) =0.3937947  
## class counts: 20 145  
## probabilities: 0.121 0.879   
## left son=6 (15 obs) right son=7 (150 obs)  
## Primary splits:  
## Cell.shape splits as LLRRRRRRRR, improve=9.818182, (0 missing)  
## Cell.size splits as LLLRRRRRRR, improve=7.938272, (0 missing)  
## Marg.adhesion splits as LLRRRRRRRR, improve=7.575758, (0 missing)  
## Bare.nuclei splits as LLLRLLLRRR, improve=6.594010, (0 missing)  
## Cl.thickness splits as LLLLRRRRRR, improve=6.450857, (0 missing)  
## Surrogate splits:  
## Bl.cromatin splits as LRRRRRRRRR, agree=0.921, adj=0.133, (0 split)  
##   
## Node number 4: 244 observations  
## predicted class=benign expected loss=0.004098361 P(node) =0.5823389  
## class counts: 243 1  
## probabilities: 0.996 0.004   
##   
## Node number 5: 10 observations  
## predicted class=malignant expected loss=0.4 P(node) =0.02386635  
## class counts: 4 6  
## probabilities: 0.400 0.600   
##   
## Node number 6: 15 observations  
## predicted class=benign expected loss=0.3333333 P(node) =0.03579952  
## class counts: 10 5  
## probabilities: 0.667 0.333   
##   
## Node number 7: 150 observations  
## predicted class=malignant expected loss=0.06666667 P(node) =0.3579952  
## class counts: 10 140  
## probabilities: 0.067 0.933

mytree.pred <- predict(mytree,train,type="class")  
#Build confusion matrix  
treetable<-confusionMatrix(mytree.pred, reference = train$Class)  
#Continue building rows for comparison table  
TreeMethod<-c(treetable$byClass["Recall"],treetable$byClass["Precision"],treetable$byClass["F1"])

#Implements Leave One Out Cross Validation Model  
ans <- numeric(length(train[,1]))  
for (i in 1:length(train[,1])) {  
 mytree <- rpart(Class~ ., train[-i,])  
 mytree.pred <- predict(mytree,train[i,],type="class")  
 ans[i] <- mytree.pred  
}  
ans <- factor(ans,labels=levels(train$Class))  
#Build confusion matrix  
loocvtable<-confusionMatrix(ans, reference = train$Class)  
loocvtable

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction benign malignant  
## benign 253 16  
## malignant 14 136  
##   
## Accuracy : 0.9284   
## 95% CI : (0.8994, 0.9512)  
## No Information Rate : 0.6372   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8447   
##   
## Mcnemar's Test P-Value : 0.8551   
##   
## Sensitivity : 0.9476   
## Specificity : 0.8947   
## Pos Pred Value : 0.9405   
## Neg Pred Value : 0.9067   
## Prevalence : 0.6372   
## Detection Rate : 0.6038   
## Detection Prevalence : 0.6420   
## Balanced Accuracy : 0.9212   
##   
## 'Positive' Class : benign   
##

#Continue building rows for comparison table  
loovcMethod<-c(loocvtable$byClass["Recall"],loocvtable$byClass["Precision"],loocvtable$byClass["F1"])

#Create new set to convert to numeric functions  
df<-train  
#Use convert to force the class to a numerical  
df %>% convert(num(Class:Mitoses))

#String all variables as integer  
df$Cl.thickness<-as.integer(df$Cl.thickness)  
df$Class<-as.integer(df$Class)  
df$Mitoses<-as.integer(df$Mitoses)  
df$Cell.size<-as.integer(df$Cell.size)  
df$Cell.shape<-as.integer(df$Cell.shape)  
df$Marg.adhesion<-as.integer(df$Marg.adhesion)  
df$Epith.c.size<-as.integer(df$Epith.c.size)  
df$Bare.nuclei<-as.integer(df$Bare.nuclei)  
df$Bl.cromatin<-as.integer(df$Bl.cromatin)  
df$Normal.nucleoli<-as.integer(df$Normal.nucleoli)  
#Implements the QDA method  
myqda <- qda(Class ~ ., data=df)  
myqda.pred <- predict(myqda, df)  
head(myqda.pred)

## $class  
## [1] 1 1 1 1 1 1 1 2 2 1 2 1 2 1 2 1 1 1 2 1 1 2 1 2 2 2 2 2 1 2 2 1 2 2 2 2 2  
## [38] 1 2 2 1 2 1 1 1 1 1 2 2 1 1 1 1 1 1 2 1 2 2 2 2 2 2 1 2 2 1 1 1 2 1 1 1 1  
## [75] 2 2 2 2 1 1 1 1 1 1 2 1 1 1 2 2 1 1 1 2 1 2 1 1 2 1 1 2 2 2 1 2 1 2 2 2 1  
## [112] 2 2 2 1 1 2 1 1 1 2 1 1 2 1 1 2 2 1 1 2 2 1 2 2 1 1 2 2 2 2 1 2 2 2 1 1 1  
## [149] 1 2 1 2 2 1 1 2 2 2 2 2 1 1 1 1 2 1 1 2 2 2 2 1 1 2 1 2 1 2 1 2 1 2 2 1 2  
## [186] 1 1 1 2 2 1 1 1 2 2 2 1 2 1 1 1 1 1 1 1 2 2 2 2 1 2 2 2 2 1 2 1 1 2 2 1 1  
## [223] 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 2 2 1 1 2 1 2 2 1 1 1 1  
## [260] 1 2 2 1 2 1 2 1 2 1 2 2 2 1 1 1 2 2 1 1 1 1 1 1 1 1 1 2 1 2 2 1 2 2 2 1 1  
## [297] 1 1 1 1 1 1 1 1 1 2 1 1 1 2 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1  
## [334] 1 2 1 2 1 2 1 1 1 2 1 2 1 2 2 2 1 1 1 1 1 1 1 1 2 2 1 1 2 2 1 1 1 1 1 1 1  
## [371] 1 1 1 2 1 1 1 1 2 2 1 1 1 1 1 1 2 1 1 1 1 1 2 2 1 1 1 1 1 2 2 2 1 1 1 1 1  
## [408] 2 2 1 1 1 1 1 1 1 1 2 2  
## Levels: 1 2  
##   
## $posterior  
## 1 2  
## 5 9.999945e-01 5.526323e-06  
##

#Build confusion matrix  
table(myqda.pred$class,df$Class)

##   
## 1 2  
## 1 254 3  
## 2 13 149

#Manual calculation of scores  
#table was built manually and scores can be calculated manually  
TP2=as.numeric("149")  
FN2=as.numeric("2")  
FP2=as.numeric("13")  
TN2=as.numeric("254")  
QDAP =(TP2/(TP2+FP2))  
QDAR = (TP2/(TP2+FN2))  
QDAF = (2\*QDAP\*QDAR)/(QDAP+QDAR)  
#Continue building rows for comparison table  
QDAMethod<-c(QDAR,QDAP,QDAF)

#Implements the RDA method  
myrda <- rda(Class ~ ., train)  
myrda.pred <- predict(myrda, train)  
#Build Confusion Matrix  
rdatable<-confusionMatrix(myrda.pred$class, reference = train$Class)  
#Continue building rows for comparison table  
RDAMethod<-c(rdatable$byClass["Recall"],rdatable$byClass["Precision"],rdatable$byClass["F1"])

#Implements the randomforest method  
myrf <- randomForest(Class ~ .,train)  
myrf.pred <- predict(myrf, train)  
#Build Confusion Matrix  
rftable<-confusionMatrix(myrf.pred, reference = train$Class)  
#Continue building rows for comparison table  
RFMethod<-c(rftable$byClass["Recall"],rftable$byClass["Precision"],rftable$byClass["F1"])

# boosting  
boost <- boosting(Class ~ ., data = train)  
predboost <- predict(boost, train, type = "class")  
predboost$prob

## [,1] [,2]  
## [1,] 0.911099168 0.088900832

predboost$confusion

## Observed Class  
## Predicted Class benign malignant  
## benign 267 0  
## malignant 0 152

predboost$error

## [1] 0

boostable<-confusionMatrix(as.factor(predboost$class), as.factor(train$Class))  
#Continue building rows for comparison table  
BOOMethod<-c(boostable$byClass["Recall"],boostable$byClass["Precision"],boostable$byClass["F1"])

Stats.DF<-data.frame("Scores for support vector machine"=SVMMethod,"Scores for Naive Bayes"=NBMethod, "Scores for support decision trees"=TreeMethod,"Scores for LOOVC"=loovcMethod ,"Scores for Neural Network" = NNMethod, "Scores for QDA"=QDAMethod,"Scores for RDA"=RDAMethod, "Scores for RF"=RFMethod,"Scores for Boosting"=BOOMethod)   
Stats.DF #this table helps assess what models should be used for informing creation of stacked ensembles

## Scores.for.support.vector.machine Scores.for.Naive.Bayes  
## Recall 0.9737828 0.9737828  
## Precision 0.9848485 0.9923664  
## F1 0.9792844 0.9829868  
## Scores.for.support.decision.trees Scores.for.LOOVC  
## Recall 0.9475655 0.9475655  
## Precision 0.9768340 0.9405204  
## F1 0.9619772 0.9440299  
## Scores.for.Neural.Network Scores.for.QDA Scores.for.RDA Scores.for.RF  
## Recall 0.9869281 0.9867550 0.9737828 1  
## Precision 0.9934211 0.9197531 0.9923664 1  
## F1 0.9901639 0.9520767 0.9829868 1  
## Scores.for.Boosting  
## Recall 1  
## Precision 1  
## F1 1

#this is one technique for setting an ensemble. it is slightly more complicated and requires control mechanisms compared to majority vote. It does a good job at predicting, but it is tough to do a 1 to 1 comparison to other models unless every model is done with train controls as well. This model was included as an alternate possibility if majority vote has issues.  
#Set control  
control <- trainControl(method="repeatedcv", number=10, repeats=3, savePredictions=TRUE, classProbs=TRUE)  
#Pick algorithms based on criteria   
#algorithms selected are done to maximize usefulness against training set  
algorithmList <- c('rpart', 'naive\_bayes','nnet')  
set.seed(123)  
#Run combined algorithm  
models <- caretList(Class~., data=train, trControl=control, methodList=algorithmList)

## Warning in trControlCheck(x = trControl, y = target): x$savePredictions == TRUE  
## is depreciated. Setting to 'final' instead.

## stopped after 100 iterations

results <- resamples(models)  
#Summary of results for comparing to other models  
summary(results)

##   
## Call:  
## summary.resamples(object = results)  
##   
## Models: rpart, naive\_bayes, nnet   
## Number of resamples: 30   
##   
## Accuracy   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## rpart 0.8571429 0.9119399 0.9285714 0.9355063 0.9534884 0.9761905 0  
## naive\_bayes 0.9024390 0.9523810 0.9534884 0.9657321 0.9766058 1.0000000 0  
## nnet 0.9069767 0.9512195 0.9639954 0.9625548 0.9761905 1.0000000 0  
##   
## Kappa   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## rpart 0.6692913 0.8095298 0.8467153 0.8584784 0.9029345 0.9489051 0  
## naive\_bayes 0.7836412 0.8962963 0.9011076 0.9258360 0.9503691 1.0000000 0  
## nnet 0.7957245 0.8931298 0.9229823 0.9184154 0.9489051 1.0000000 0

#sergios method  
combine.classes<-data.frame(myrf.pred, myrda.pred$class,  
mytree.pred,mynnet.pred,mysvm.pred, mynb.pred$class)  
combine.classes$myrf.pred<-ifelse(combine.classes$myrf.pred=="benign", 0, 1)  
combine.classes[,2]<-ifelse(combine.classes[,2]=="benign", 0, 1)  
combine.classes[,3]<-ifelse(combine.classes[,3]=="benign", 0, 1)  
combine.classes[,4]<-ifelse(combine.classes[,4]=="benign", 0, 1)  
combine.classes[,5]<-ifelse(combine.classes[,5]=="benign", 0, 1)  
combine.classes[,6]<-ifelse(combine.classes[,6]=="benign", 0, 1)  
majority.vote=rowSums(combine.classes)  
#head(majority.vote)  
combine.classes[,7]<-rowSums(combine.classes)  
combine.classes[,8]<-ifelse(combine.classes[,7]>=4, "malignant", "benign")  
table(combine.classes[,8], train$Class)

##   
## benign malignant  
## benign 260 2  
## malignant 7 150

mvtable<-confusionMatrix(as.factor(combine.classes[,8]), reference = as.factor(train$Class))  
MVMethod<-c(mvtable$byClass["Recall"],mvtable$byClass["Precision"],mvtable$byClass["F1"])  
MVMethod

## Recall Precision F1   
## 0.9737828 0.9923664 0.9829868

StatsTrain.DF<-data.frame("Scores for support vector machine"=SVMMethod,"Scores for Naive Bayes"=NBMethod, "Scores for support decision trees"=TreeMethod,"Scores for LOOVC"=loovcMethod ,"Scores for Neural Network" = NNMethod, "Scores for QDA"=QDAMethod,"Scores for RDA"=RDAMethod, "Scores for RF"=RFMethod,"Scores for Boosting"=BOOMethod, "Scores for Majority Vote"=MVMethod)   
StatsTrain.DF #use this to determine what to run against test classes

## Scores.for.support.vector.machine Scores.for.Naive.Bayes  
## Recall 0.9737828 0.9737828  
## Precision 0.9848485 0.9923664  
## F1 0.9792844 0.9829868  
## Scores.for.support.decision.trees Scores.for.LOOVC  
## Recall 0.9475655 0.9475655  
## Precision 0.9768340 0.9405204  
## F1 0.9619772 0.9440299  
## Scores.for.Neural.Network Scores.for.QDA Scores.for.RDA Scores.for.RF  
## Recall 0.9869281 0.9867550 0.9737828 1  
## Precision 0.9934211 0.9197531 0.9923664 1  
## F1 0.9901639 0.9520767 0.9829868 1  
## Scores.for.Boosting Scores.for.Majority.Vote  
## Recall 1 0.9737828  
## Precision 1 0.9923664  
## F1 1 0.9829868

#we will test Neural Network, RDA, Majority Vote   
#Using Majority Vote means running multiple algorithms to create majority vote algorithm  
#SVM Testing for Majority Vote  
svmtest <- svm(Class ~ ., test)  
svmtest.pred <- predict(svmtest, test)  
#Naive Bayes Testing for Majority Vote  
tbtest <- NaiveBayes(Class ~ ., test)  
tbtest.pred <- predict(tbtest,test)

#Neural Network for Majority Vote and comparison  
mytnnet <- nnet(Class ~ ., test, size=1)

## # weights: 83  
## initial value 205.934568   
## iter 10 value 18.525643  
## iter 20 value 1.259686  
## iter 30 value 0.012055  
## iter 40 value 0.003774  
## iter 50 value 0.001344  
## iter 60 value 0.000349  
## final value 0.000070   
## converged

mytnnet.pred <- predict(mytnnet,test,type="class")  
#Build confusion matrix  
table(mytnnet.pred,test$Class)

##   
## mytnnet.pred benign malignant  
## benign 191 0  
## malignant 0 89

#Manual calculation of scores  
TPTN=as.numeric("191")  
FNTN=as.numeric("0")  
FPTN=as.numeric("0")  
TNTN=as.numeric("89")  
NNPrTN=(TPTN/(TPTN+FPTN))  
NNRTN = (TPTN/(TPTN+FNTN))  
NNFTN = (2\*NNPrTN\*NNRTN)/(NNPrTN+NNRTN)  
#Continue building rows for comparison table  
NNTNMethod<-c(NNRTN,NNPrTN,NNFTN)  
#RDA for Majority Vote and comparison  
testrda <- rda(Class ~ ., test)  
testrda.pred <- predict(testrda, test)  
#Build Confusion Matrix  
trdatable<-confusionMatrix(testrda.pred$class, reference = test$Class)  
#Continue building rows for comparison table  
RDATMethod<-c(trdatable$byClass["Recall"],trdatable$byClass["Precision"],trdatable$byClass["F1"])  
#RF for Majority Vote  
testrf <- randomForest(Class ~ .,test)  
testrf.pred <- predict(testrf, test)  
#loovc for Majority Vote  
anst <- numeric(length(test[,1]))  
for (i in 1:length(test[,1])) {  
 mytreet <- rpart(Class~ ., test[-i,])  
 mytreet.pred <- predict(mytreet,test[i,],type="class")  
 anst[i] <- mytreet.pred  
}  
anst <- factor(ans,labels=levels(test$Class))  
#majority vote  
combine.classes<-data.frame(testrf.pred, testrda.pred$class,  
mytreet.pred,mytnnet.pred,svmtest.pred, tbtest.pred$class)  
combine.classes$testrf.pred<-ifelse(combine.classes$testrf.pred=="benign", 0, 1)  
combine.classes[,2]<-ifelse(combine.classes[,2]=="benign", 0, 1)  
combine.classes[,3]<-ifelse(combine.classes[,3]=="benign", 0, 1)  
combine.classes[,4]<-ifelse(combine.classes[,4]=="benign", 0, 1)  
combine.classes[,5]<-ifelse(combine.classes[,5]=="benign", 0, 1)  
combine.classes[,6]<-ifelse(combine.classes[,6]=="benign", 0, 1)  
majority.vote=rowSums(combine.classes)  
#head(majority.vote)  
combine.classes[,7]<-rowSums(combine.classes)  
combine.classes[,8]<-ifelse(combine.classes[,7]>=4, "malignant", "benign")  
table(combine.classes[,8], test$Class)

##   
## benign malignant  
## benign 189 0  
## malignant 2 89

mvttable<-confusionMatrix(as.factor(combine.classes[,8]), reference = as.factor(test$Class))  
MVMTethod<-c(mvttable$byClass["Recall"],mvttable$byClass["Precision"],mvttable$byClass["F1"])  
MVMTethod

## Recall Precision F1   
## 0.9895288 1.0000000 0.9947368

StatsFinal.DF<-data.frame("Scores for Train Neural Network" = NNMethod, "Scores for Test Neural Network" = NNTNMethod, "Scores for Test RDA"=RDATMethod, "Scores for Train RDA"=RDAMethod, "Scores for Train Majority Vote"=MVMethod, "Scores for Test Majority Vote"=MVMTethod)   
StatsFinal.DF #This table will determine best algorithm

## Scores.for.Train.Neural.Network Scores.for.Test.Neural.Network  
## Recall 0.9869281 1  
## Precision 0.9934211 1  
## F1 0.9901639 1  
## Scores.for.Test.RDA Scores.for.Train.RDA  
## Recall 0.9895288 0.9737828  
## Precision 0.9947368 0.9923664  
## F1 0.9921260 0.9829868  
## Scores.for.Train.Majority.Vote Scores.for.Test.Majority.Vote  
## Recall 0.9737828 0.9895288  
## Precision 0.9923664 1.0000000  
## F1 0.9829868 0.9947368

#simple RDA performs well enough in comparison to Majority vote that gain by majority vote may not be worth effort. would need larger data set to determine  
#neural network learned data and overfit the test model. As neural network is a feeder for Majority vote, would have to reexamine majority vote algorithm choices