APPENDIX

R CODE:

```
#Loading the dataset and checking the definition of variables

load("C:/Users/sonne/Downloads/backpain.RData")
codeVariables

# Comparing classifiers using training, validation and test

# Load the required libraries
library(rpart)
library(partykit)
library(randomForest)
library(adabag)
library(nnet)
# Load the kernlab package
library(kernlab)
# Load the vignette for the package
vignette("kernlab")


# creating the index for each type dataset,by applying bootstrap
# on train dataset

N <- nrow(dat)
set.seed(1) #setting seed to reproduce the results
indtrain <- sample(1:N,size=N,replace=TRUE)
indtrain <- sort(indtrain)
indvalid <- sample(setdiff(1:N,indtrain),size=0.2*N)
indvalid <- sort(indvalid)
indtest <- setdiff(1:N,union(indtrain,indvalid))

# Splitting the dataset into train validation and test
train=dat[indtrain,]
valid=dat[indvalid,]
test=dat[indtest,]


# Fit a classifier to only the training data
fit.r <- rpart(PainDiagnosis~.,data=train)

# Fit a logistic regression to the training data only too
```

```
# First load the nnet package

library(nnet)
fit.l <- multinom(PainDiagnosis~., data=train)

# fir random forest to the training data
fit.ran=randomForest(PainDiagnosis~., data=train)

# fit bagging of classification trees on the same data.
fit.bag<-bagging(PainDiagnosis~.,data=train)

# fit svm to the training dataset
fit.svm=ksvm(PainDiagnosis~.,kernel="rbfdot",data=train)

# Boosting
fit.boost=boosting(PainDiagnosis~.,data=train,coeflearn = "Breiman",
                   boos=F,mfinal = 30)


# Classify for ALL of the observations in validation dataset by diffrent models
pred.r <- predict(fit.r,type="class",newdata=valid)
pred.l <- predict(fit.l,type="class",newdata=valid)
pred.ran=predict(fit.ran,type="class",newdata=valid)
pred.bag<-predict(fit.bag,type="class",newdata=valid)
pred.svm<-predict(fit.svm,type="response",newdata=valid)
pred.boost=predict.boosting(fit.boost,newdata = valid)

# Look at table for the validation data only (rows=truth, cols=prediction)
tab.r <- table(valid$PainDiagnosis,pred.r)
tab.r
tab.l <- table(valid$PainDiagnosis,pred.l)
tab.l
tab.ran <- table(valid$PainDiagnosis,pred.ran)
tab.ran
tab.bag=pred.bag$confusion
tab.bag
tab.svm=table(valid$PainDiagnosis,pred.svm)
tab.svm
tab.boost=pred.boost$confusion
tab.boost
# Work out the accuracy
acc.r <- sum(diag(tab.r))/sum(tab.r)
acc.l <- sum(diag(tab.l))/sum(tab.l)
acc.ran=sum(diag(tab.ran))/sum(tab.ran)
acc.bag=(sum(tab.bag)-sum(diag(tab.bag)))/sum(tab.bag)
acc.svm=sum(diag(tab.svm))/sum(tab.svm)
```

```
acc.boost=(sum(tab.boost)-sum(diag(tab.boost)))/sum(tab.boost)
acc.r
acc.l
acc.ran
acc.bag
acc.boost
acc.svm

#Finding specificity and sensitivity

# R part
c1=confusionMatrix(valid$PainDiagnosis,pred.r)
sensitivity_r=c1$byClass[[1]]
specificity_r=c1$byClass[[2]]

# Random forest
c2=confusionMatrix(valid$PainDiagnosis,pred.ran)
sensitivity_ran=c2$byClass[[1]]
specificity_ran=c2$byClass[[2]]

# Logistic regression
c3=confusionMatrix(valid$PainDiagnosis,pred.l)
sensitivity_l=c3$byClass[[1]]
specificity_l=c3$byClass[[2]]

# SVM
c5=confusionMatrix(valid$PainDiagnosis,pred.svm)
sensitivity_svm=c5$byClass[[1]]
specificity_svm=c5$byClass[[2]]

# Plotting ROC curves
library(pROC)
plot(roc(as.numeric(valid$PainDiagnosis),as.numeric(pred.r), direction="<"),
    col="yellow", lwd=2, main="ROC curves for different model")
lines(roc(as.numeric(valid$PainDiagnosis),as.numeric(pred.ran), direction="<"),
    col="blue", lwd=2)
lines(roc(as.numeric(valid$PainDiagnosis),as.numeric(pred.l), direction="<"),
    col="red", lwd=2)
lines(roc(as.numeric(valid$PainDiagnosis),
        as.numeric(as.factor(pred.bag$class)), direction=">"),
    col="black", lwd=2, main="ROC curve for Bagging model")
lines(roc(as.numeric(valid$PainDiagnosis),
        as.numeric(as.factor(pred.boost$class)), direction=">"),
    col="green", lwd=2, main="ROC curve for Boosting model")
lines(roc(as.numeric(valid$PainDiagnosis),as.numeric(pred.svm), direction="<"),
    col="violet", lwd=2, main="ROC curve for SVM model")
```

```
legend("topleft",c("R part","Random forest","Logistic regression","Bagging",
       "Boosting","SVM"),col = c("yellow","blue","red","black","green",
                                 "violet"),lty = c(1,1,1,1,1,1),cex=0.6)

# Look at the method that did best on the validation data
# and apply the same to the test data

# Classify for ALL of the observations of test dataset using Randomforest fit

pred.ran <- predict(fit.ran,type="class",newdata=test)
tab <- table(test$PainDiagnosis,pred.ran)
tab
#accuracy
sum(diag(tab))/sum(tab)


# From the error statistics, it is clear that random forest is the better model

# Variable importance plot for random forests
varImpPlot(fit.ran)
# Values of the variable importance criterion
varImp(fit.ran)
# Important variables are Painlocation and Criterion8

#extraction of rules
library(inTrees)
treeList <- RF2List(fit.ran)  # transform rf object to an inTrees' format
exec <- extractRules(treeList, dat[,-1])
ruleMetric <- getRuleMetric(exec,dat[,-1],dat[,1]) # get rule metrics
ruleMetric <- pruneRule(ruleMetric, dat[,-1], dat[,1]) #prune the rule
ruleMetric <- selectRuleRRF(ruleMetric, dat[,-1], dat[,1])
ruleMetric[1:5,]


# Let's repeat this process to see if we were just unlucky!

# Set up res to store results

res<-matrix(NA,100,7)

# Start simulation to look at this
iterlim <- 100
for (iter in 1:iterlim)
{
  # creating the index for each type dataset
  N <- nrow(dat)
```

```
indtrain <- sample(1:N,size=N,replace=TRUE)
indtrain <- sort(indtrain)
indvalid <- sample(setdiff(1:N,indtrain),size=0.25*N)
indvalid <- sort(indvalid)
indtest <- setdiff(1:N,union(indtrain,indvalid))

# Splitting the dataset into train validation and test
train=dat[indtrain,]
valid=dat[indvalid,]
test=dat[indtest,]


# Fit a classifier to only the training data
fit.r <- rpart(PainDiagnosis~.,data=train)
#plot(as.party(fit.r))

# Fit a logistic regression to the training data only too
fit.l <- multinom(PainDiagnosis~., data=train)

# fir random forest to the training data
fit.ran=randomForest(PainDiagnosis~., data=train)


# fit svm to the training dataset
fit.svm=ksvm(PainDiagnosis~.,data=train)

# fit boosting to the training dataset
fit.boost=boosting(PainDiagnosis~.,data=train,coeflearn = "Breiman",
                   boos=F,mfinal = 30)


# Classify for ALL of the observations in validation dataset
pred.r <- predict(fit.r,type="class",newdata=valid)
pred.l <- predict(fit.l,type="class",newdata=valid)
pred.ran=predict(fit.ran,type="class",newdata=valid)
pred.svm<-predict(fit.svm,type="response",newdata=valid)
pred.boost=predict.boosting(fit.boost,newdata = valid)

# Look at table for the validation data only (rows=truth, cols=prediction)
tab.r <- table(valid$PainDiagnosis,pred.r)
tab.l <- table(valid$PainDiagnosis,pred.l)
tab.ran <- table(valid$PainDiagnosis,pred.ran)
tab.svm=table(valid$PainDiagnosis,pred.svm)
tab.boost=pred.boost$confusion

# Work out the accuracy
```

```
acc.r <- sum(diag(tab.r))/sum(tab.r)
acc.l <- sum(diag(tab.l))/sum(tab.l)
acc.ran=sum(diag(tab.ran))/sum(tab.ran)
acc.svm=sum(diag(tab.svm))/sum(tab.svm)
acc.boost=(sum(tab.boost)-sum(diag(tab.boost)))/sum(tab.boost)

# Store the results
res[iter,1] <- acc.r
res[iter,2] <- acc.l
res[iter,3] <- acc.ran
res[iter,4] <- acc.svm
res[iter,5] <- acc.boost

# Look at the method that did best on the validation data
# when applied to the test data
lis=c(acc.r,acc.l,acc.svm,acc.ran,acc.boost)
chosen=which(max(lis)==lis)
if(chosen==5){
  testpred=predict.boosting(fit.boost,newdata=test)
  tab <- pred.boost$confusion
  acc <- (sum(tab.boost)-sum(diag(tab.boost)))/sum(tab.boost)
  res[iter,6] <- 5
  res[iter,7] <- acc
}else if(chosen==3){
  testpred=predict(fit.ran,type="class",newdata=test)
  tab <- table(test$PainDiagnosis,testpred)
  acc <- sum(diag(tab))/sum(tab)
  res[iter,6] <- 3
  res[iter,7] <- acc
}else if(chosen==4){
  testpred=predict(fit.svm,type="response",newdata=test)
  tab <- table(test$PainDiagnosis,testpred)
  acc <- sum(diag(tab))/sum(tab)
  res[iter,6] <- 4
  res[iter,7] <- acc
}else if(chosen==1){
  testpred=predict(fit.r,type="class",newdata=test)
  tab <- table(test$PainDiagnosis,testpred)
  acc <- sum(diag(tab))/sum(tab)
  res[iter,6] <- 1
  res[iter,7] <- acc
}else if(chosen==2){
  testpred=predict(fit.l,type="class",newdata=test)
  tab <- table(test$PainDiagnosis,testpred)
  acc <- sum(diag(tab))/sum(tab)
  res[iter,6] <- 2
```

```
    res[iter,7] <- acc
  }
} # iter


# Check out the error rate summary statistics.
colnames(res)<-c("valid.r","valid.l","valid.ran","valid.svm","valid.boost",
                 "chosen","test")

apply(res[,-6],2,summary)

tab <- table(res[,6])
names(tab)<- c("Rpart","Logistic","RandomForest","SVM","Boosting")
tab
```