HW3

Michael Cho

February 25, 2017

## Problem 1

# a.

For n fold cross validation, the best k is 8 with 0.33948 MSE and 50.47 . For 10 fold cross validation, the best k is 7 with with 0.3477054 mse AND 49.27 . Some pros and cons is that n fold is not more computationally expensive than 10 fold cross validation, since we are not fitting a model. N fold cross validation produces unbiased estimates, but is more highly variable than k fold validation and is generally a worse estimate of test set error when trying to generalize to unseen data.

# b.

There is no prediction error, since we are using n fold cross validation and all other outside observations of the iteration fold are considered as candidates for nearest neighbors and there is no need to run replicates, since the same nearest neighbors would be used for individual observations all the time.

# c.

Using 8 neighbors and looking through the whole heart data set, we find that row indexes 517,307,622,478,313,778,273, and 80 are the closest neighbors and the predicted average of their log10 cost is 3.1449.

## Problem 2

# a.

It seems the intvn and ervis has the biggest effect based on plots in the appendix. Please note that variables where a smoother was not applied will not appear in the plot output.

# b.

The 10 fold cross validation standard deviation of prediction error is 0.0016. N fold cross validation is more computationally expensive than 10 fold cross validation and while it produces unbiased estimates of test set error, it is more highly variable while 10 fold cross validation is more balanced between variance and bias often producing lower test set error. There also seems to be an issue with applying a smoother to variables a low number of levels. For example, drugs and comp seems to have a low range of values and therefore a smoother cannot be applied in some scenarios where data values are sparse. This would be exacerbated in a 10 fold versus a n fold cross validation scenario as n fold just uses leave one out methodology.

# c.

The predicted cost from the model is 3.556478.

## Problem 3

# a.

The 10 fold cross validation indicates that we get the lowest mean square cv error of 0.221 and 67.6 with degree 1 and span 0.5.

# b.

The lowest cp is 0.2239008 with degree 1 and span 0.5 with agrees with the optimal parameters from the 10 fold cross validation.

# c.

The prediction error standard deviation of the best 10 fold cv error is 0.001.

# d.

The predicted cost from the model is 3.625035.

## Problem 4

# a.

The 10 fold cross validation indicates the optimal number of terms is 2.

# b.

The prediction error standard deviation of the best 10 fold cv error is 0.002. Please refer to the appendix for the summary of the fitted model as well as plots of component function. Term1 has a coefficient of 0.7348 and term2 has a coefficient of 0.1591, so term 1 seems to be more important. Comp (0.925) and intvn (0.29) weigh the most for term1 and gend(0.603) and comp(0.785) weigh the most for term2.

# c.

The predicted cost from the model is 3.614803.

## Problem 5

# a.

The best knn model according to 10 fold cv is with k=3 with 0.13 misclass rate.

# b.

For gam, I didn't apply a smoothing paramaeter to Ri, Ca, and Mg according to gam.check which does a check on the degrees of freedom used as indicated by k in the smoothing function. 10 fold cv indicates a 0.17 misclass rate.

# c.

The best neural network model according to 10 fold cv is of size 25 and decay 0.005 with a misclass rate of 0.14. So it seems comparable to knn but better than gams.

## Problem 6

# a.

Please check appendix for plot of cv vs test squared error loss. The best number of trees seem to be around 53.

# b.

Based on the summary with the best number of trees, we find that intvn has the most importance, followed by comorb, and dur.

# c.

The predicted cost is 3.392167.

## Problem 7

# a.

The out of bag is 66.85 and doesn't seem to vary much between replicates.

# b.

Yes, the number of trees grown seem to be appropriate since the error seems to be lowest a lot lower than 500 trees.

# c.

The most important variables seem to be intvn, comorb, dur, and comp. This seems to generally agree with variable importance measures from other methods.

# d.

age - negative nonlinear gend - negative linear intvn - positive nonlinear drugs - nonlinear ervis - positive nonlinear comp - positive nonlinear comorb - positive nonlinear dur - positive nonlinear This agrees with the variable importance measures as we can see by the y axis the scale from lowest to highest variable values. Intvn effect on cost from from 2 to 4 while comorb ranges from 2.5 to 3.2. These partial dependence plots also agree with the partial dependence plots, but random forests seems to be a lot smoother, while the boosted tree dependence plots look more like piecewise functions. Gender is also not linear in boosted trees. The partial dependence plots is found in the appendix code for problem 6.

# e.

The predicted cost is 3.624023.

# f.

The model with mtry=2 seems to have the highest of 66.91. In an individual tree, mtry is the number of variables that is considered at each split. This will be different for each individual tree in the ensemble of trees.

# g.

Based on cross validated values, I would choose boosted trees at 68.78% .

## Appendix

# Problem 1

setwd("C:/Users/mcho/Desktop/pred\_anal2")  
heart<-read.csv("HW2dat.csv")  
heart$cost<-log10(heart$cost)

library(yaImpute)  
  
CVInd <- function(n,K) { #n is sample size; K is number of parts; returns K-length list of indices for each part  
 m<-floor(n/K) #approximate size of each part  
 r<-n-m\*K   
 I<-sample(n,n) #random reordering of the indices  
 Ind<-list() #will be list of indices for all K parts  
 length(Ind)<-K  
 for (k in 1:K) {  
 if (k <= r) kpart <- ((m+1)\*(k-1)+1):((m+1)\*k)   
 else kpart<-((m+1)\*r+m\*(k-r-1)+1):((m+1)\*r+m\*(k-r))  
 Ind[[k]] <- I[kpart] #indices for kth part of data  
 }  
 Ind  
}  
  
# n fold cv  
n=nrow(heart)  
K<-n #K-fold CV on each replicate  
n.models = 4 #number of different models to fit  
y<-heart$cost  
yhat=matrix(0,n,n.models)   
MSE<-matrix(0,1,n.models)  
  
Ind<-CVInd(n,K)  
 for (k in 1:K) {  
 train<-as.matrix(heart[-Ind[[k]],c(2:9)])  
 test<-as.matrix(heart[Ind[[k]],c(2:9)])  
 ytrain<-heart[-Ind[[k]],1]  
 K1=9; K2=8; K3=7; K4=6;  
 out<-ann(train,test,K1,verbose=F)  
 ind<-as.matrix(t(out$knnIndexDist[,1:K1]))  
 yhat[Ind[[k]],1]<-apply(ind,1,function(x) mean(ytrain[x]))  
 out<-ann(train,test,K2,verbose=F)  
 ind<-as.matrix(t(out$knnIndexDist[,1:K2]))  
 yhat[Ind[[k]],2]<-apply(ind,1,function(x) mean(ytrain[x]))  
 out<-ann(train,test,K3,verbose=F)  
 ind<-as.matrix(t(out$knnIndexDist[,1:K3]))  
 yhat[Ind[[k]],3]<-apply(ind,1,function(x) mean(ytrain[x]))  
 out<-ann(train,test,K4,verbose=F)  
 ind<-as.matrix(t(out$knnIndexDist[,1:K4]))  
 yhat[Ind[[k]],4]<-apply(ind,1,function(x) mean(ytrain[x]))  
 } #end of k loop  
MSE[1,]=apply(yhat,2,function(x) sum((y-x)^2))/n; MSE #mean square CV error

## [,1] [,2] [,3] [,4]  
## [1,] 0.34552 0.3394838 0.3443251 0.3505619

r2<-1-MSE/var(y); r2 #CV r^2

## [,1] [,2] [,3] [,4]  
## [1,] 0.4959313 0.5047373 0.4976744 0.4885758

# k=8 is best here  
  
# predicted cost  
finalknn<-ann(as.matrix(heart[,c(2:9)]),as.matrix(data.frame(age=59,gend=0,intvn=10,drugs=0,ervis=3,comp=0,comorb=4,dur=300)),8,verbose=F)  
finalknn$knnIndexDist

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]  
## [1,] 517 307 622 478 313 778 273 80 35 38 65 67 67  
## [,14] [,15] [,16]  
## [1,] 69 79 80

mean(heart[finalknn$knnIndexDist[,1:8], 1])

## [1] 3.144949

# 10 fold cv  
Nrep<-50 #number of replicates of CV  
K<-10 #K-fold CV on each replicate  
n=nrow(heart)  
n.models = 4 #number of different models to fit  
y<-heart$cost  
yhat=matrix(0,n,n.models)   
MSE<-matrix(0,Nrep,n.models)  
for (j in 1:Nrep) {  
 Ind<-CVInd(n,K)  
 for (k in 1:K) {  
 train<-as.matrix(heart[-Ind[[k]],c(2:9)])  
 test<-as.matrix(heart[Ind[[k]],c(2:9)])  
 ytrain<-heart[-Ind[[k]],1]  
 K1=9; K2=8; K3=7; K4=6;  
 out<-ann(train,test,K1,verbose=F)  
 ind<-as.matrix(out$knnIndexDist[,1:K1])  
 yhat[Ind[[k]],1]<-apply(ind,1,function(x) mean(ytrain[x]))  
 out<-ann(train,test,K2,verbose=F)  
 ind<-as.matrix(out$knnIndexDist[,1:K2])  
 yhat[Ind[[k]],2]<-apply(ind,1,function(x) mean(ytrain[x]))  
 out<-ann(train,test,K3,verbose=F)  
 ind<-as.matrix(out$knnIndexDist[,1:K3])  
 yhat[Ind[[k]],3]<-apply(ind,1,function(x) mean(ytrain[x]))  
 out<-ann(train,test,K4,verbose=F)  
 ind<-as.matrix(out$knnIndexDist[,1:K4])  
 yhat[Ind[[k]],4]<-apply(ind,1,function(x) mean(ytrain[x]))  
 } #end of k loop  
 MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n  
} #end of j loop  
MSEAve<- apply(MSE,2,mean); MSEAve #averaged mean square CV error

## [1] 0.3553032 0.3500018 0.3490524 0.3535880

MSEsd <- apply(MSE,2,sd); MSEsd #SD of mean square CV error

## [1] 0.003287207 0.004058691 0.005184606 0.005238700

r2<-1-MSEAve/var(y); r2 #CV r^2

## [1] 0.4816588 0.4893929 0.4907779 0.4841611

# k=7 best here

# Problem 2

library(mgcv)  
out<-gam(cost~s(age)+s(intvn) + gend + comp + drugs + s(ervis) + s(comorb) + s(dur), data=heart,family=gaussian())   
summary(out)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## cost ~ s(age) + s(intvn) + gend + comp + drugs + s(ervis) + s(comorb) +   
## s(dur)  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.74355 0.02121 129.366 < 2e-16 \*\*\*  
## gend -0.07083 0.04010 -1.766 0.0777 .   
## comp 0.28958 0.06971 4.154 3.63e-05 \*\*\*  
## drugs -0.02730 0.01950 -1.400 0.1619   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(age) 1.000 1.000 3.617 0.05757 .   
## s(intvn) 4.610 5.563 136.168 < 2e-16 \*\*\*  
## s(ervis) 4.450 5.435 3.101 0.00606 \*\*   
## s(comorb) 3.924 4.809 17.023 2.72e-15 \*\*\*  
## s(dur) 5.917 7.043 5.514 3.11e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.685 Deviance explained = 69.4%  
## GCV = 0.22296 Scale est. = 0.2162 n = 788

par(mfrow=c(2,3))  
plot(out)  
  
# predicted cost  
predict(out,data.frame(age=59,gend=0,intvn=10,drugs=0,ervis=3,comp=0,comorb=4,dur=300))

## 1   
## 3.556478

# n fold cv  
n=nrow(heart)  
K<-n #K-fold CV on each replicate  
n.models = 1 #number of different models to fit  
y<-heart$cost  
yhat=matrix(0,n,n.models)   
MSE<-matrix(0,1,n.models)  
  
Ind<-CVInd(n,K)  
 for (k in 1:K) {  
 out<-gam(cost~s(age)+s(intvn) + gend + comp + drugs  
 + s(ervis) + s(comorb) + s(dur),   
 data=heart[-Ind[[k]],],family=gaussian())   
 yhat[Ind[[k]],1]<-predict(out,heart[Ind[[k]],])  
 } #end of k loop  
MSE[1,]=apply(yhat,2,function(x) sum((y-x)^2))/n; MSE #mean square CV error

## [,1]  
## [1,] 0.2253335

r2<-1-MSE/var(y); r2 #CV r^2

## [,1]  
## [1,] 0.6712677

# 10 fold cv  
Nrep<-50 #number of replicates of CV  
K<-10 #K-fold CV on each replicate  
n=nrow(heart)  
n.models = 1 #number of different models to fit  
y<-heart$cost  
yhat=matrix(0,n,n.models)   
MSE<-matrix(0,Nrep,n.models)  
for (j in 1:Nrep) {  
 Ind<-CVInd(n,K)  
 for (k in 1:K) {  
 out<-gam(cost~s(age)+s(intvn) + gend + comp + drugs  
 + s(ervis) + s(comorb) + s(dur),   
 data=heart[-Ind[[k]],],family=gaussian())   
 yhat[Ind[[k]],1]<-predict(out,heart[Ind[[k]],])  
 } #end of k loop  
 MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n  
} #end of j loop  
MSEAve<- apply(MSE,2,mean); MSEAve #averaged mean square CV error

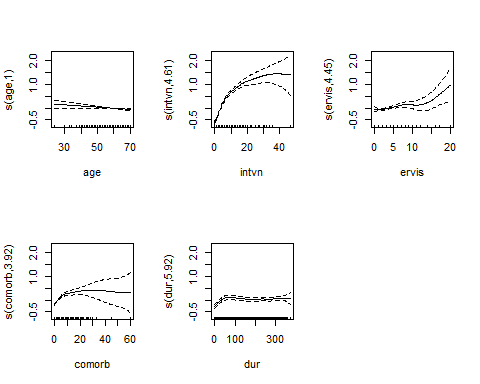
## [1] 0.2260036

MSEsd <- apply(MSE,2,sd); MSEsd #SD of mean square CV error

## [1] 0.001713345

r2<-1-MSEAve/var(y); r2 #CV r^2

## [1] 0.6702901



# Problem 3

# 10 fold cv  
Nrep<-50 #number of replicates of CV  
K<-10 #K-fold CV on each replicate  
n=nrow(heart)  
n.models = 9#number of different models to fit  
y<-heart$cost  
yhat=matrix(0,n,n.models)   
MSE<-matrix(0,Nrep,n.models)  
for (j in 1:Nrep) {  
 Ind<-CVInd(n,K)  
 for (k in 1:K) {  
 out<- loess(cost~intvn + comorb + dur + ervis, data=heart[-Ind[[k]],],  
 control=loess.control(surface = "direct"), span=0.1, degree=0)   
 yhat[Ind[[k]],1]<-predict(out,heart[Ind[[k]],])  
 out<- loess(cost~intvn + comorb + dur + ervis, data=heart[-Ind[[k]],],   
 control=loess.control(surface = "direct"), span=0.5, degree=0)   
 yhat[Ind[[k]],2]<-predict(out,heart[Ind[[k]],])  
 out<- loess(cost~intvn + comorb + dur + ervis, data=heart[-Ind[[k]],],   
 control=loess.control(surface = "direct"), span=1, degree=0)   
 yhat[Ind[[k]],3]<-predict(out,heart[Ind[[k]],])  
 out<- loess(cost~intvn + comorb + dur + ervis, data=heart[-Ind[[k]],],   
 control=loess.control(surface = "direct"), span=0.1, degree=1)   
 yhat[Ind[[k]],4]<-predict(out,heart[Ind[[k]],])  
 out<- loess(cost~intvn + comorb + dur + ervis, data=heart[-Ind[[k]],],   
 control=loess.control(surface = "direct"), span=0.5, degree=1)   
 yhat[Ind[[k]],5]<-predict(out,heart[Ind[[k]],])  
 out<- loess(cost~intvn + comorb + dur + ervis, data=heart[-Ind[[k]],],   
 control=loess.control(surface = "direct"), span=1, degree=1)   
 yhat[Ind[[k]],6]<-predict(out,heart[Ind[[k]],])  
 out<- loess(cost~intvn + comorb + dur + ervis, data=heart[-Ind[[k]],],   
 control=loess.control(surface = "direct"),span=0.1, degree=2)   
 yhat[Ind[[k]],7]<-predict(out,heart[Ind[[k]],])  
 out<- loess(cost~intvn + comorb + dur + ervis, data=heart[-Ind[[k]],],   
 control=loess.control(surface = "direct"), span=0.5, degree=2)   
 yhat[Ind[[k]],8]<-predict(out,heart[Ind[[k]],])  
 out<- loess(cost~intvn + comorb + dur + ervis, data=heart[-Ind[[k]],],   
 control=loess.control(surface = "direct"),span=1, degree=2)   
 yhat[Ind[[k]],9]<-predict(out,heart[Ind[[k]],])  
 } #end of k loop  
 MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n  
} #end of j loop  
MSEAve<- apply(MSE,2,mean); MSEAve #averaged mean square CV error

## [1] 0.2276983 0.2989229 0.6542799 0.2287382 0.2219651 0.2806409 0.2986728  
## [8] 0.2255190 0.2353081

MSEsd <- apply(MSE,2,sd); MSEsd #SD of mean square CV error

## [1] 0.0012833103 0.0006815753 0.0014620084 0.0042285681 0.0014487006  
## [6] 0.0013104484 0.0120664741 0.0063771735 0.0021627597

r2<-1-MSEAve/var(y); r2 #CV r^2

## [1] 0.66781784 0.56391043 0.04549076 0.66630078 0.67618184 0.59058150  
## [7] 0.56427521 0.67099709 0.65671611

# cp   
for (lambda in seq(.02,.2,.02)) {out<-loess(cost~intvn + comorb + dur + ervis, heart,degree=0, span=lambda); print(c(lambda,out$s))}

## [1] 0.0200000 0.4844904  
## [1] 0.0400000 0.4821727  
## [1] 0.0600000 0.4886634  
## [1] 0.0800000 0.4960235  
## [1] 0.1000000 0.5017093  
## [1] 0.1200000 0.5183665  
## [1] 0.1400000 0.5279226  
## [1] 0.1600000 0.5343597  
## [1] 0.1800000 0.5492153  
## [1] 0.2000000 0.5588878

sig\_hat0<-0.4821727  
  
for (lambda in c(seq(.01,.05,.01), seq(.1,1,.2))) {out<-loess(cost~intvn + comorb + dur + ervis, heart,degree=0, span=lambda); SSE<-sum((heart$cost-out$fitted)^2); Cp <- (SSE+2\*out$trace.hat\*sig\_hat0^2)/n; print(c(lambda,Cp))}

## [1] 0.0100000 0.3447391  
## [1] 0.020000 0.250741  
## [1] 0.0300000 0.2447149  
## [1] 0.040000 0.243106  
## [1] 0.0500000 0.2447364  
## [1] 0.1000000 0.2558221  
## [1] 0.3000000 0.3592051  
## [1] 0.500000 0.445103  
## [1] 0.7000000 0.5055038  
## [1] 0.90000 0.54365

# degree 0 with lambda 0.04, Cp=0.243106  
  
for (lambda in seq(.02,.2,.02)) {out<-loess(cost~intvn + comorb + dur + ervis, heart,degree=1, span=lambda); print(c(lambda,out$s))}

## [1] 0.0200000 0.7217457  
## [1] 0.0400000 0.4908207  
## [1] 0.060000 0.477706  
## [1] 0.0800000 0.4696464  
## [1] 0.1000000 0.4671513  
## [1] 0.1200000 0.4676536  
## [1] 0.1400000 0.4728385  
## [1] 0.1600000 0.4711656  
## [1] 0.1800000 0.4710862  
## [1] 0.2000000 0.4708937

sig\_hat1<-0.4671513  
  
for (lambda in c(seq(.01,.05,.01), seq(.1,1,.2))) {out<-loess(cost~intvn + comorb + dur + ervis, heart,degree=1, span=lambda); SSE<-sum((heart$cost-out$fitted)^2); Cp <- (SSE+2\*out$trace.hat\*sig\_hat1^2)/n; print(c(lambda,Cp))}

## [1] 0.010000 3.919603  
## [1] 0.0200000 0.4633957  
## [1] 0.0300000 0.2895165  
## [1] 0.0400000 0.2636108  
## [1] 0.0500000 0.2526826  
## [1] 0.1000000 0.2320746  
## [1] 0.3000000 0.2254268  
## [1] 0.5000000 0.2239008  
## [1] 0.7000000 0.2280489  
## [1] 0.9000000 0.2393453

# degree 1 with lambda 0.5, Cp=0.2239008  
  
for (lambda in seq(.02,.2,.02)) {out<-loess(cost~intvn + comorb + dur + ervis, heart,degree=2, span=lambda); print(c(lambda,out$s))}

## [1] 0.02000 81.95271  
## [1] 0.040000 6.941557  
## [1] 0.060000 1.008625  
## [1] 0.080000 0.749715  
## [1] 0.1000000 0.6802507  
## [1] 0.1200000 0.6951063  
## [1] 0.1400000 0.6351406  
## [1] 0.1600000 0.6129484  
## [1] 0.1800000 0.6529255  
## [1] 0.2000000 0.6234012

sig\_hat2<-0.6129484  
  
for (lambda in c(seq(.01,.05,.01), seq(.1,2,.2))) {out<-loess(cost~intvn + comorb + dur + ervis, heart,degree=2, span=lambda); SSE<-sum((heart$cost-out$fitted)^2); Cp <- (SSE+2\*out$trace.hat\*sig\_hat2^2)/n; print(c(lambda,Cp))}

## [1] 0.01 49511260.64  
## [1] 0.020 4095.139  
## [1] 0.03000 17.41063  
## [1] 0.04000 29.98221  
## [1] 0.05000 3.29391  
## [1] 0.1000000 0.5006148  
## [1] 0.3000000 0.3427353  
## [1] 0.5000000 0.2812468  
## [1] 0.7000000 0.2598593  
## [1] 0.9000000 0.2553736  
## [1] 1.1000000 0.2404817  
## [1] 1.3000000 0.2414684  
## [1] 1.5000000 0.2410224  
## [1] 1.7000000 0.2407992  
## [1] 1.9000000 0.2407165

# degree 2 with lambda 1.1, Cp=0.2404817  
  
# predicted cost   
out<-loess(cost~intvn + comorb + dur + ervis, heart,degree=1, span=0.5)  
predict(out,data.frame(age=59,gend=0,intvn=10,drugs=0,ervis=3,comp=0,comorb=4,dur=300))

## 1   
## 3.625035

# Problem 4

library(stats)  
  
# n fold cv  
n=nrow(heart)  
K<-n #K-fold CV on each replicate  
n.models = 3 #number of different models to fit  
y<-heart$cost  
yhat=matrix(0,n,n.models)   
MSE<-matrix(0,1,n.models)  
  
Ind<-CVInd(n,K)  
 for (k in 1:K) {  
 out<- ppr(cost~., data=heart[-Ind[[k]],], nterms=3)   
 yhat[Ind[[k]],1]<-predict(out,heart[Ind[[k]],])  
 out<- ppr(cost~., data=heart[-Ind[[k]],], nterms=2)   
 yhat[Ind[[k]],2]<-predict(out,heart[Ind[[k]],])  
 out<- ppr(cost~., data=heart[-Ind[[k]],], nterms=1)   
 yhat[Ind[[k]],3]<-predict(out,heart[Ind[[k]],])  
 } #end of k loop  
MSE[1,]=apply(yhat,2,function(x) sum((y-x)^2))/n; MSE #mean square CV error

## [,1] [,2] [,3]  
## [1,] 0.2199436 0.2172311 0.218855

r2<-1-MSE/var(y); r2 #CV r^2

## [,1] [,2] [,3]  
## [1,] 0.6791309 0.683088 0.6807191

# optimal number of n terms is 2  
  
# 10 fold cv  
Nrep<-50 #number of replicates of CV  
K<-10 #K-fold CV on each replicate  
n=nrow(heart)  
n.models = 3#number of different models to fit  
y<-heart$cost  
yhat=matrix(0,n,n.models)   
MSE<-matrix(0,Nrep,n.models)  
for (j in 1:Nrep) {  
 Ind<-CVInd(n,K)  
 for (k in 1:K) {  
 out<- ppr(cost~., data=heart[-Ind[[k]],], nterms=3)   
 yhat[Ind[[k]],1]<-predict(out,heart[Ind[[k]],])  
 out<- ppr(cost~., data=heart[-Ind[[k]],], nterms=2)   
 yhat[Ind[[k]],2]<-predict(out,heart[Ind[[k]],])  
 out<- ppr(cost~., data=heart[-Ind[[k]],], nterms=1)   
 yhat[Ind[[k]],3]<-predict(out,heart[Ind[[k]],])  
 } #end of k loop  
 MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n  
} #end of j loop  
MSEAve<- apply(MSE,2,mean); MSEAve #averaged mean square CV error

## [1] 0.2230094 0.2188478 0.2195882

MSEsd <- apply(MSE,2,sd); MSEsd #SD of mean square CV error

## [1] 0.004169366 0.002537754 0.001287023

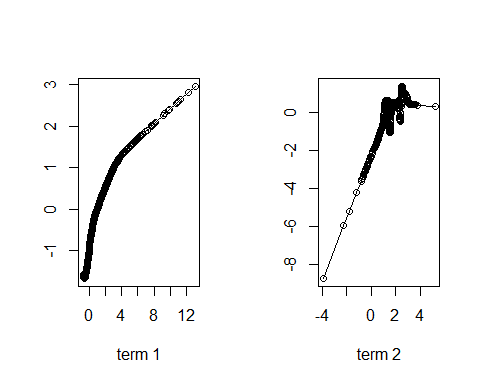
r2<-1-MSEAve/var(y); r2 #CV r^2

## [1] 0.6746583 0.6807296 0.6796494

# fitted model interpretation and predicted cost use nterms=2  
out<-ppr(cost~., data=heart, nterms=2)   
summary(out)

## Call:  
## ppr(formula = cost ~ ., data = heart, nterms = 2)  
##   
## Goodness of fit:  
## 2 terms   
## 151.8006   
##   
## Projection direction vectors:  
## term 1 term 2   
## age -0.011897310 0.033386867  
## gend -0.151610398 0.603046978  
## intvn 0.290111026 0.008185856  
## drugs -0.111364027 0.065095812  
## ervis 0.064720395 0.066472939  
## comp 0.925599430 0.785962008  
## comorb 0.139219151 -0.093566942  
## dur 0.001175057 -0.002341940  
##   
## Coefficients of ridge terms:  
## term 1 term 2   
## 0.7348637 0.1591152

par(mfrow=c(1,2)); plot(out)



predict(out,data.frame(age=59,gend=0,intvn=10,drugs=0,ervis=3,comp=0,comorb=4,dur=300))

## 1   
## 3.614803

par(mfrow=c(1,1))

# Problem 5

glass<-read.csv("HW2datb.csv")  
glass$type<-factor(ifelse(glass$type=="WinF"|glass$type=="WinNF","Win","Other"))  
  
# 10 fold cv for knn  
Nrep<-3 #number of replicates of CV  
K<-10 #K-fold CV on each replicate  
n=nrow(glass)  
n.models = 4 #number of different models to fit  
y<-glass$type  
yhat=matrix(0,n,n.models)   
MSC<-matrix(0,Nrep,n.models)  
for (j in 1:Nrep) {  
 Ind<-CVInd(n,K)  
 for (k in 1:K) {  
 train<-as.matrix(glass[-Ind[[k]],c(1:9)])  
 test<-as.matrix(glass[Ind[[k]],c(1:9)])  
 ytrain<-glass[-Ind[[k]],10]  
 K1=5; K2=4; K3=3; K4=2;  
 out<-ann(train,test,K1,verbose=F)  
 ind<-as.matrix(out$knnIndexDist[,1:K1])  
 phat<-apply(ind,1,function(x) sum(ytrain[x]=="Win")/length(ytrain[x]))  
 yhat[Ind[[k]],1]<-ifelse(phat>0.5, "Win","Other")  
 out<-ann(train,test,K2,verbose=F)  
 ind<-as.matrix(out$knnIndexDist[,1:K2])  
 phat<-apply(ind,1,function(x) sum(ytrain[x]=="Win")/length(ytrain[x]))  
 yhat[Ind[[k]],2]<-ifelse(phat>0.5, "Win","Other")  
 out<-ann(train,test,K3,verbose=F)  
 ind<-as.matrix(out$knnIndexDist[,1:K3])  
 phat<-apply(ind,1,function(x) sum(ytrain[x]=="Win")/length(ytrain[x]))  
 yhat[Ind[[k]],3]<-ifelse(phat>0.5, "Win","Other")  
 out<-ann(train,test,K4,verbose=F)  
 ind<-as.matrix(out$knnIndexDist[,1:K4])  
 phat<-apply(ind,1,function(x) sum(ytrain[x]=="Win")/length(ytrain[x]))  
 yhat[Ind[[k]],4]<-ifelse(phat>0.5, "Win","Other")  
 } #end of k loop  
 MSC[j,]=apply(yhat,2,function(x) sum(x!=y)/n)  
} #end of j loop  
MSCAve<- apply(MSC,2,mean); MSCAve #averaged mean square CV misclass rate

## [1] 0.1495327 0.1386293 0.1261682 0.1401869

MSCsd <- apply(MSC,2,sd); MSCsd #SD of mean square CV misclass rate

## [1] 0.004672897 0.002697898 0.004672897 0.012363324

# k =3 best here  
  
# 10 fold cv for gam  
# used gam.check(out) to not apply smoothing to variables where close to k  
Nrep<-3 #number of replicates of CV  
K<-10 #K-fold CV on each replicate  
n=nrow(glass)  
n.models = 1 #number of different models to fit  
y<-glass$type  
yhat=matrix(0,n,n.models)   
MSC<-matrix(0,Nrep,n.models)  
for (j in 1:Nrep) {  
 Ind<-CVInd(n,K)  
 for (k in 1:K) {  
 out<-gam(type~RI+s(Na)+Mg+s(Al)+s(Si)+s(K)+Ca+s(Ba)+s(Fe),  
 data=glass[-Ind[[k]],], family=binomial())  
 #print(gam.check(out))  
 phat<-predict(out,glass[Ind[[k]],], type="response")  
 yhat[Ind[[k]],1]<-ifelse(phat>0.5, "Win","Other")  
 } #end of k loop  
 MSC[j,]=apply(yhat,2,function(x) sum(x!=y)/n)  
} #end of j loop  
MSCAve<- apply(MSC,2,mean); MSCAve #averaged mean square CV misclass rate

## [1] 0.1775701

MSCsd <- apply(MSC,2,sd); MSCsd #SD of mean square CV miscalss rate

## [1] 0.008093695

# 10 fold cv for nnet  
standglass<-glass  
standglass[ , c(1:9)]<-scale(standglass[ , c(1:9)], center=T, scale=T)  
library(nnet)  
Nrep<-3 #number of replicates of CV  
K<-10 #K-fold CV on each replicate  
n=nrow(glass)  
n.models = 9 #number of different models to fit  
y<-glass$type  
yhat=matrix(0,n,n.models)   
MSC<-matrix(0,Nrep,n.models)  
for (j in 1:Nrep) {  
 Ind<-CVInd(n,K)  
 for (k in 1:K) {  
 out<-nnet(type~.,standglass[-Ind[[k]],], linout=F, skip=F,   
 size=20, decay=0.005, maxit=1000, trace=F)  
 yhat[Ind[[k]],1]<-as.character(predict(out,standglass[Ind[[k]],], type="class"))  
 out<-nnet(type~.,standglass[-Ind[[k]],], linout=F, skip=F,   
 size=20, decay=0.01, maxit=100, trace=F)  
 yhat[Ind[[k]],2]<-as.character(predict(out,standglass[Ind[[k]],], type="class"))  
 out<-nnet(type~.,standglass[-Ind[[k]],], linout=F, skip=F,   
 size=20, decay=0.05, maxit=100, trace=F)  
 yhat[Ind[[k]],3]<-as.character(predict(out,standglass[Ind[[k]],], type="class"))  
 out<-nnet(type~.,standglass[-Ind[[k]],], linout=F, skip=F,   
 size=25, decay=0.005, maxit=100, trace=F)  
 yhat[Ind[[k]],4]<-as.character(predict(out,standglass[Ind[[k]],], type="class"))  
 out<-nnet(type~.,standglass[-Ind[[k]],], linout=F, skip=F,   
 size=25, decay=0.01, maxit=100, trace=F)  
 yhat[Ind[[k]],5]<-as.character(predict(out,standglass[Ind[[k]],], type="class"))  
 out<-nnet(type~.,standglass[-Ind[[k]],], linout=F, skip=F,   
 size=25, decay=0.05, maxit=100, trace=F)  
 yhat[Ind[[k]],6]<-as.character(predict(out,standglass[Ind[[k]],], type="class"))  
 out<-nnet(type~.,standglass[-Ind[[k]],], linout=F, skip=F,   
 size=30, decay=0.005, maxit=100, trace=F)  
 yhat[Ind[[k]],7]<-as.character(predict(out,standglass[Ind[[k]],], type="class"))  
 out<-nnet(type~.,standglass[-Ind[[k]],], linout=F, skip=F,   
 size=30, decay=0.01, maxit=100, trace=F)  
 yhat[Ind[[k]],8]<-as.character(predict(out,standglass[Ind[[k]],], type="class"))  
 out<-nnet(type~.,standglass[-Ind[[k]],], linout=F, skip=F,   
 size=30, decay=0.05, maxit=100, trace=F)  
 yhat[Ind[[k]],9]<-as.character(predict(out,standglass[Ind[[k]],], type="class"))  
 } #end of k loop  
 MSC[j,]=apply(yhat,2,function(x) sum(y!=x)/n)  
} #end of j loop  
MSCAve<- apply(MSC,2,mean); MSCAve #averaged mean misclassification rate

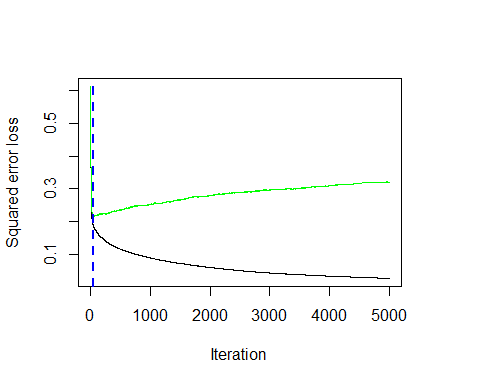
## [1] 0.1542056 0.1619938 0.1635514 0.1713396 0.1526480 0.1619938 0.1651090  
## [8] 0.1728972 0.1526480

MSCsd<-apply(MSC,2,sd);MSCsd

## [1] 0.009345794 0.011759867 0.008093695 0.011759867 0.005395797 0.007137968  
## [7] 0.005395797 0.009345794 0.009727411

# Problem 6

#install.packages("gbm")  
set.seed(135)  
library(gbm)  
gbm1 <- gbm(cost~., data=heart, var.monotone=rep(0,8), distribution="gaussian", n.trees=5000, shrinkage=0.1, interaction.depth=3, bag.fraction = .5, train.fraction = 1, n.minobsinnode = 10, cv.folds = 10, keep.data=TRUE, verbose=FALSE)  
best.iter <- gbm.perf(gbm1,method="cv");best.iter



## [1] 52

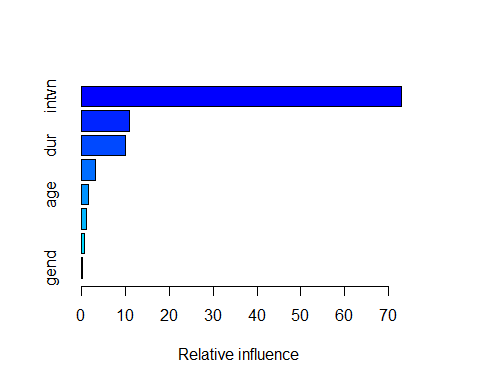
sqrt(gbm1$cv.error[best.iter]) #CV error SD

## [1] 0.4642951

1-gbm1$cv.error[best.iter]/var(heart$cost) #CV r^2

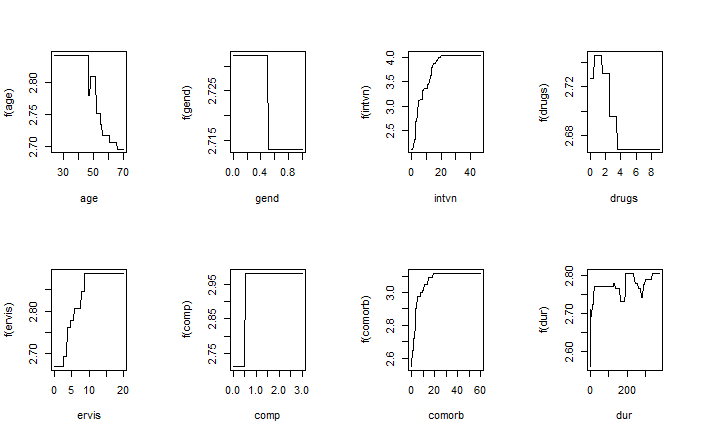
## [1] 0.6855116

summary(gbm1,n.trees=best.iter) # based on the optimal number of trees



## var rel.inf  
## intvn intvn 72.9124913  
## comorb comorb 10.7529278  
## dur dur 9.9119777  
## ervis ervis 3.1880071  
## age age 1.4655200  
## comp comp 1.0406340  
## drugs drugs 0.5249077  
## gend gend 0.2035346

# partial dependence plots  
par(mfrow=c(2,4))  
plot(gbm1, i.var = 1, n.trees = best.iter)  
plot(gbm1, i.var = 2, n.trees = best.iter)  
plot(gbm1, i.var = 3, n.trees = best.iter)  
plot(gbm1, i.var = 4, n.trees = best.iter)  
plot(gbm1, i.var = 5, n.trees = best.iter)  
plot(gbm1, i.var = 6, n.trees = best.iter)  
plot(gbm1, i.var = 7, n.trees = best.iter)  
plot(gbm1, i.var = 8, n.trees = best.iter)

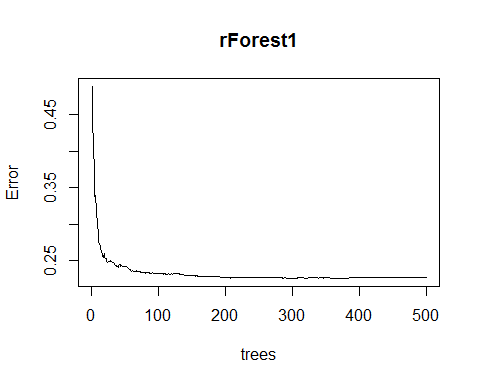


# prediction  
predict(gbm1,data.frame(age=59,gend=0,intvn=10,drugs=0,ervis=3,comp=0,comorb=4,dur=300), n.trees = best.iter)

## [1] 3.506312

# Problem 7

#install.packages("randomForest")  
set.seed(1234)  
library(randomForest)  
rForest1 <- randomForest(cost~., data=heart, mtry=3, ntree = 500, importance = TRUE)  
par(mfrow=c(1,1))  
plot(rForest1) #plots OOB mse vs # trees

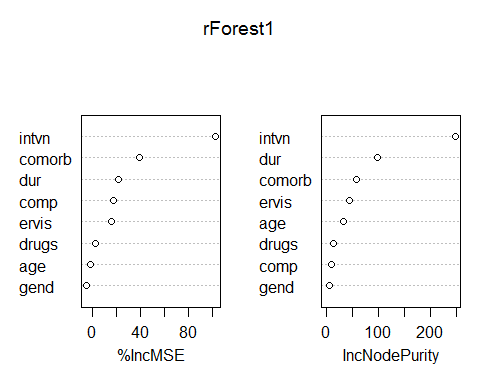


rForest1 #check the OOB mse and r^2

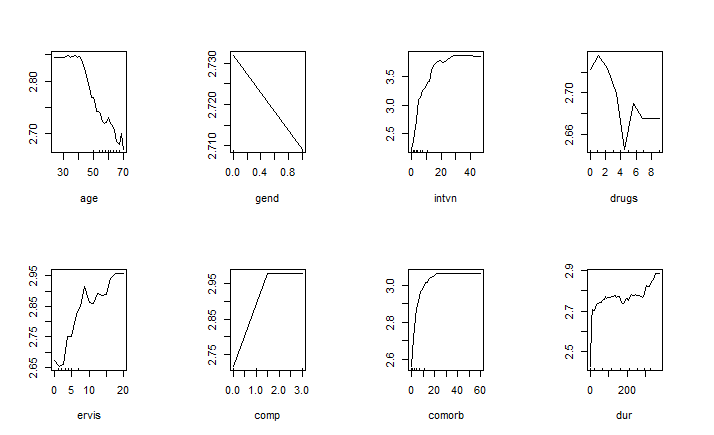
##   
## Call:  
## randomForest(formula = cost ~ ., data = heart, mtry = 3, ntree = 500, importance = TRUE)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## Mean of squared residuals: 0.2269444  
## % Var explained: 66.85

importance(rForest1); varImpPlot(rForest1)

## %IncMSE IncNodePurity  
## age -1.805068 32.491460  
## gend -4.917792 5.558582  
## intvn 102.666692 249.234914  
## drugs 2.156468 11.777113  
## ervis 15.440700 44.492357  
## comp 17.170477 8.762452  
## comorb 39.378343 56.272071  
## dur 21.274893 98.186049



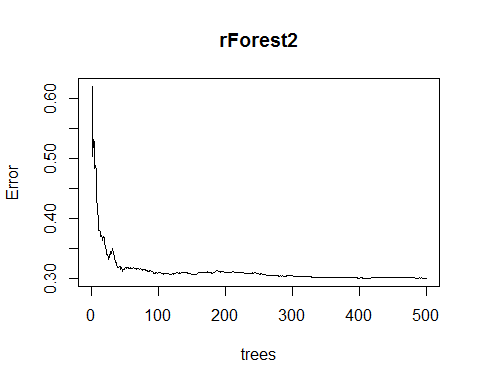
par(mfrow=c(2,4))  
for (i in c(2:9)) partialPlot(rForest1, pred.data=heart, x.var = names(heart)[i], xlab = names(heart)[i], main=NULL) #creates "partial dependence" plots



# prediction   
predict(rForest1, data.frame(age=59,gend=0,intvn=10,drugs=0,ervis=3,comp=0,comorb=4,dur=300))

## 1   
## 3.624023

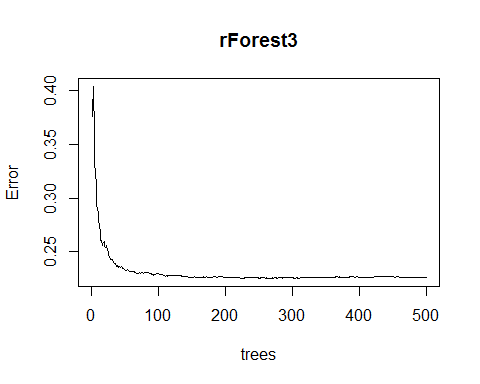
# mtry 1 and 2  
set.seed(1234)  
rForest2 <- randomForest(cost~., data=heart, mtry=1, ntree = 500, importance = TRUE)  
par(mfrow=c(1,1))  
plot(rForest2) #plots OOB mse vs # trees



rForest2 #check the OOB mse and r^2

##   
## Call:  
## randomForest(formula = cost ~ ., data = heart, mtry = 1, ntree = 500, importance = TRUE)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 1  
##   
## Mean of squared residuals: 0.3013911  
## % Var explained: 55.98

set.seed(1234)  
rForest3 <- randomForest(cost~., data=heart, mtry=2, ntree = 500, importance = TRUE)  
par(mfrow=c(1,1))  
plot(rForest3) #plots OOB mse vs # trees



rForest3 #check the OOB mse and r^2

##   
## Call:  
## randomForest(formula = cost ~ ., data = heart, mtry = 2, ntree = 500, importance = TRUE)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## Mean of squared residuals: 0.2265548  
## % Var explained: 66.91