HW2

Michael Cho

February 4, 2017

## Please note that I made unstandardized and standardized versions of the datasets. The standardized datsets are for neural networks and the unstandardized are used for other models. The MSE of different models should therefore be directly comparable as well as the response variable is the same for all models problems 1-3, and the same for all models for problem 4.

## Problem 1

## a.

The is 55% with a training MSE of 0.3008. This is a very low value and later on, we will show that compared to neural networks and regression trees, the linear regression model had the least predictive power.

# b.

The correlation matrix and scatterplot diagrams show that the total number of interventions/procedures (intvn) has the highest effect on cost (r=0.70), followed by number of days of treatment condition (comorb, r=0.35) and number of emergency room visits (comp, r=0.31). Only the variables intvn, ervis, comp, and comorb are significant in the regression output.

# c.

The variables drugs, number of comorbidites, and number of days of treatment are heavily right skewed, suggesting we need a log transform. Here we have been instructed to not transform these predictors. While the normality plot seems to fit roughly well, except for outliers, the residual vs. fitted plot shows that there continues to be nonlinearity in the model as the residuals are not arranged in a parallel band as linear regression requires errors by i.i.d. If we were continuing with a linear regression model, we might want to consider a transformation of the response variable and remove outliers accordings to Cook’s distance.

## Problem 2

# a.

Please refer to appendix for k-fold. Combinations of size 25,30,35 were mixed with decay of 0.5,1,2 to find the optimal mix. It seems the best combination is size 25 and decay of 1.

# b.

The cross validation shows that of the optimal mix is 67.7%, with a cv MSE of 0.2213 and a training MSE of 0.2236. This is a good improvement compared to the linear model.

# c.

Please refer to appendix for ALEplots of the main and second order effects. The main effect of the predictors are below, calculated from taking the difference between the high and low value of these variables on cost. It seems that just like the linear output, the number of interventions has the highest effect on cost (2) followed by comorb (1.5) and drugs (-0.5). The rest of the variables have a moderate effect on cost (0.25), except for age and gender which has the lowest abs value. Most second order interactions don't have much effect on cost, except for intvn and comp, intvn and comorb, drugs and comp, and comp and comorb. Matrices of these second order interactions are in the appendix as well.

age= -0.08, gender= - -0.01, intvn= 2, drugs= - 0.5, ervis = 0.25, comp = 0.25, comorb = 1.5

# d.

The residual plot shows pretty random errors. The neural network seems to capture most of the nonlinearities. Please look at the appendix for the appropriate plot.

## Problem 3

# a.

Please refer to appendix for k-fold. A regression tree with complexity parameter of 0.0035195 seems to be the most optimal.

# b.

The cross validation shows that of the optimal tree size is approximately 65% with a MSE of 0.35, while the training is 70.5% and MSE is 0.2012. The model is better than the linear regression, but the cv and training results are quite different indicating issues with fitting.

# c.

Please refer to the appendix for the text and graphic output of the regression tree.

Any tree starts by taking variables which would produce the most significant split in the data. In the case of a regression tree, the criteria for a split is finding the variable value/factor which decreases SSE. Intvn has the largest impact reducing SSE by 323.41 followed by comorb 49.55. The variable importance is the total SSE reduced wherever the variable was used in a binary split and printed below.

## intvn comorb ervis comp age   
## 323.406063 49.545544 3.655447 2.206868 2.032086

# d.

The residual plot seems to have normal errors and most nonlinearities seem to be captured.

# e.

I would choose the neural network. Although both regression tree and neural networks have better and MSE compared to the linear model, the neural network has more comparable results between its cross validation and training and MSE results compared to the regression tree. In addition, regression trees have less predictive power when fitted to a set of data points that indeed have some sort of linear trend.

## Problem 4

Please note that model comparisons were based on misclassification rate, while growing the classification tree is based on improvements in the gini index.

# a.

Please refer to appendix for the neural network 10 fold cv. The grid of model parameters was size of 20,30,40 and decay of 0.05, 0.1, 0.15. Optimal mix was found to be size 30 and decay 0.05. The corresponding cv misclass rate was around 0.27 and training misclass rate around 0.30.

# b.

Please refer to appendix for the classification tree 10 fold cv. Optimal complexity parameter was 0.0144928 with a cv misclass rate of 0.29775 and training misclass rate of 0.228972.

# c.

Please refer to appendix for the multinomial logistic regression. The cv misclass rate was 0.38 and training misclass rate was 0.266. Please note that the summary is comparing the log odds of other categories compared to Containers. A one unit increase in the predictors increases the log odss of head and table vs cont except for K, CA, and Ba. With other categories, a one unit increases decreases log odds compared to Container except for Fe.

# d.

Both the classification tree and multinomial logistic regression are more interpretable than the neural network since the former just conducts recursive partitioning of the predictor space into easy to follow rules and multinomial logistic regression provides the change in log odds of particular classes with a unit change in the predictor space. Both the neural network and classification tree are better than the multinomial logistic regression in their cv misclass rate, while the neural network and classification tree are quite similar in their results. So with results being relatively similar, I would choose the classification tree model since it is more interpretable than the neural network.

## Appendix

# Read in data

# Problem 1

setwd("C:/Users/Michael/Desktop/pred\_anal2")  
# install.packages("xlsx")  
# install.packages('rJava')  
options(java.home="C:\\Program Files\\Java\\jre7\\")  
library(rJava)  
library(xlsx)  
heart<-read.xlsx("HW2\_data.xls", sheetName= 1)  
  
# first column has row index get rid of it  
heart<-heart[,2:9]  
  
# cost to log10 cost response  
heart$cost<-log10(heart$cost)  
  
# standardized predictor dataset and unstandardized predictor dataset  
standheart<-heart  
standheart[ , c(2,4:8)]<-scale(standheart[ ,c(2,4:8)], center=T, scale=T)  
unstandheart<-heart

# a and b.

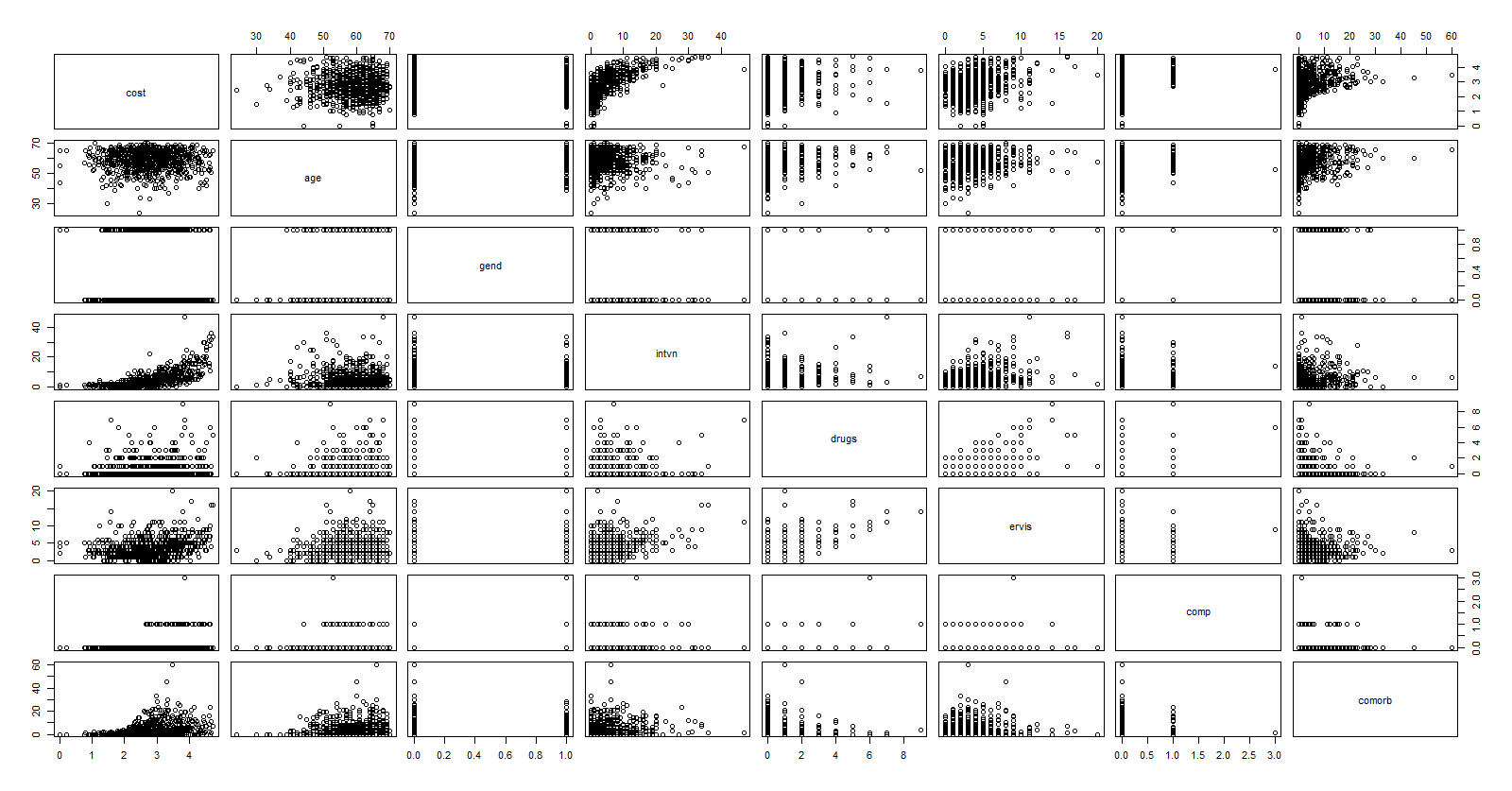
regfit<-lm(cost~., unstandheart)  
summary(regfit)

##   
## Call:  
## lm(formula = cost ~ ., data = unstandheart)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.63926 -0.29784 0.02001 0.31735 1.58525   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.227871 0.174331 12.780 < 2e-16 \*\*\*  
## age -0.002380 0.002939 -0.810 0.41833   
## gend -0.055611 0.047170 -1.179 0.23878   
## intvn 0.091151 0.003873 23.537 < 2e-16 \*\*\*  
## drugs -0.022834 0.021924 -1.042 0.29796   
## ervis 0.025739 0.009281 2.773 0.00568 \*\*   
## comp 0.342639 0.081487 4.205 2.92e-05 \*\*\*  
## comorb 0.034449 0.003375 10.207 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.5513 on 780 degrees of freedom  
## Multiple R-squared: 0.5605, Adjusted R-squared: 0.5566   
## F-statistic: 142.1 on 7 and 780 DF, p-value: < 2.2e-16

n<-nrow(unstandheart)  
# training MSE 0.30  
sum((unstandheart$cost-predict(regfit, unstandheart))^2)/n

## [1] 0.3008608

pairs(unstandheart)



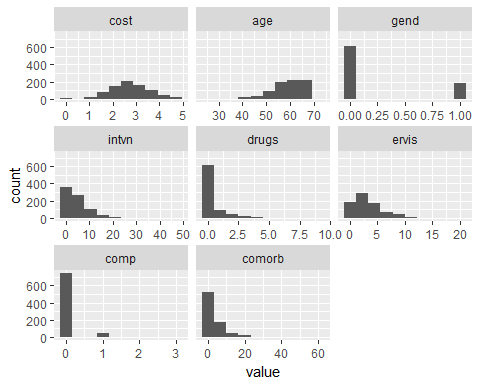
round(cor(unstandheart),2)

## cost age gend intvn drugs ervis comp comorb  
## cost 1.00 -0.02 -0.01 0.70 0.16 0.31 0.24 0.35  
## age -0.02 1.00 -0.02 -0.03 0.03 0.06 -0.04 0.09  
## gend -0.01 -0.02 1.00 0.03 0.03 0.11 0.03 -0.04  
## intvn 0.70 -0.03 0.03 1.00 0.23 0.37 0.20 0.15  
## drugs 0.16 0.03 0.03 0.23 1.00 0.53 0.15 -0.05  
## ervis 0.31 0.06 0.11 0.37 0.53 1.00 0.16 0.03  
## comp 0.24 -0.04 0.03 0.20 0.15 0.16 1.00 0.03  
## comorb 0.35 0.09 -0.04 0.15 -0.05 0.03 0.03 1.00

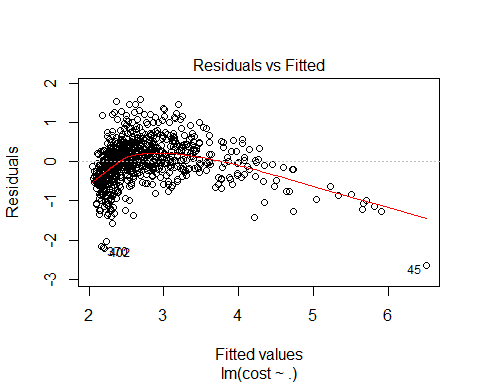
# the number of interventions/procedures carried out and number of days of treatment condition  
# seemed to have the biggest effect on cost

# c.

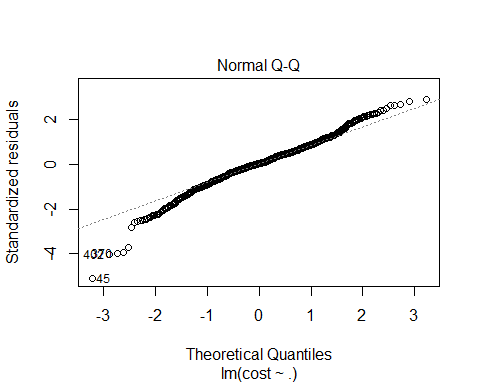
library(reshape2)  
library(ggplot2)  
ggplot(data = melt(unstandheart), mapping = aes(x = value)) +   
 geom\_histogram(bins = 10) + facet\_wrap(~variable, scales = 'free\_x')



# drugs, number of comorbidites, and number of days of treatment are heavily right skewed  
plot(regfit, which=1)



plot(regfit, which=2)



# Problem 2

# a and b.

library(nnet)  
  
# linout=T for continous values  
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  
}  
  
Nrep<-3 #number of replicates of CV  
K<-10 #K-fold CV on each replicate  
n.models = 9 #number of different models to fit  
n=nrow(standheart)  
y<-standheart$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<-nnet(cost~.,standheart[-Ind[[k]],], linout=T, skip=F, size=25,   
 decay=0.5, maxit=100, trace=F)  
 yhat[Ind[[k]],1]<-as.numeric(predict(out,standheart[Ind[[k]],]))  
 out<-nnet(cost~.,standheart[-Ind[[k]],], linout=T, skip=F, size=25,   
 decay=1, maxit=100, trace=F)  
 yhat[Ind[[k]],2]<-as.numeric(predict(out,standheart[Ind[[k]],]))  
 out<-nnet(cost~.,standheart[-Ind[[k]],], linout=T, skip=F, size=25,   
 decay=2, maxit=100, trace=F)  
 yhat[Ind[[k]],3]<-as.numeric(predict(out,standheart[Ind[[k]],]))  
 out<-nnet(cost~.,standheart[-Ind[[k]],], linout=T, skip=F, size=30,   
 decay=0.5, maxit=100, trace=F)  
 yhat[Ind[[k]],4]<-as.numeric(predict(out,standheart[Ind[[k]],]))  
 out<-nnet(cost~.,standheart[-Ind[[k]],], linout=T, skip=F, size=30,   
 decay=1, maxit=100, trace=F)  
 yhat[Ind[[k]],5]<-as.numeric(predict(out,standheart[Ind[[k]],]))  
 out<-nnet(cost~.,standheart[-Ind[[k]],], linout=T, skip=F, size=30,   
 decay=2, maxit=100, trace=F)  
 yhat[Ind[[k]],6]<-as.numeric(predict(out,standheart[Ind[[k]],]))  
 out<-nnet(cost~.,standheart[-Ind[[k]],], linout=T, skip=F, size=35,   
 decay=0.5, maxit=100, trace=F)  
 yhat[Ind[[k]],7]<-as.numeric(predict(out,standheart[Ind[[k]],]))  
 out<-nnet(cost~.,standheart[-Ind[[k]],], linout=T, skip=F, size=35,   
 decay=1, maxit=100, trace=F)  
 yhat[Ind[[k]],8]<-as.numeric(predict(out,standheart[Ind[[k]],]))  
 out<-nnet(cost~.,standheart[-Ind[[k]],], linout=T, skip=F, size=35,   
 decay=2, maxit=100, trace=F)  
 yhat[Ind[[k]],9]<-as.numeric(predict(out,standheart[Ind[[k]],]))  
 } #end of k loop  
 MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n  
} #end of j loop  
MSE

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] 0.2276366 0.2233624 0.2248628 0.2218990 0.2252605 0.2261612 0.2242273  
## [2,] 0.2238317 0.2208508 0.2238957 0.2216429 0.2213259 0.2236388 0.2208366  
## [3,] 0.2243286 0.2199329 0.2231988 0.2237933 0.2207438 0.2236446 0.2256156  
## [,8] [,9]  
## [1,] 0.2234988 0.2250161  
## [2,] 0.2212479 0.2237993  
## [3,] 0.2209733 0.2238925

MSEAve<- apply(MSE,2,mean); MSEAve #averaged mean square CV error

## [1] 0.2252656 0.2213820 0.2239858 0.2224451 0.2224434 0.2244815 0.2235598  
## [8] 0.2219066 0.2242360

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

## [1] 0.0020683064 0.0017753872 0.0008356742 0.0011746153 0.0024569575  
## [6] 0.0014546660 0.0024583832 0.0013856768 0.0006771902

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

## [1] 0.6713668 0.6770324 0.6732339 0.6754816 0.6754840 0.6725106 0.6738553  
## [8] 0.6762671 0.6728689

# lowest MSE around 0.2213 with 68% R^2  
  
# best model parameters are size=25 and decay=1  
nn.fit<-nnet(cost~.,standheart, linout=T, skip=F, size=25, decay=1, maxit=10, trace=F)  
# training MSE of 0.224  
sum((standheart$cost-predict(nn.fit, standheart))^2)/n

## [1] 0.2236191

# c.

# install.packages("yaImpute")  
# install.packages("ALEPlot")  
# path <- './ALEPlot\_1.0.tar.gz'  
# install.packages(path, repos = NULL, type="source")  
library(ALEPlot)  
yhat <- function(X.model, newdata) as.numeric(predict(X.model, newdata))  
par(mfrow=c(2,4))  
for (j in 1:7) {ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=j, K=50, NA.plot = TRUE)  
 rug(heart[,j]) } ## This creates main effect ALE plots for all 7 predictors  
  
# main effects   
# age= -0.08, gender=-0.01, intvn=2,drugs=-0.5,ervis=0.25,comp=0.25,comorb= 1.5  
age<-matrix(rep(c(0.06,-0.02),2),nrow=2)  
age[2,1]-age[1,1]

## [1] -0.08

gender<-matrix(rep(c(0.005, -0.005),2),nrow=2)  
gender[2,1]-gender[1,1]

## [1] -0.01

intvn<-matrix(rep(c(-0.5, 1.5),2),nrow=2)  
intvn[2,1]-intvn[1,1]

## [1] 2

drugs<-matrix(rep(c(0, -0.5),2),nrow=2)  
drugs[2,1]-drugs[1,1]

## [1] -0.5

ervis<-matrix(rep(c(-0.05, 0.2),2),nrow=2)  
ervis[2,1]-ervis[1,1]

## [1] 0.25

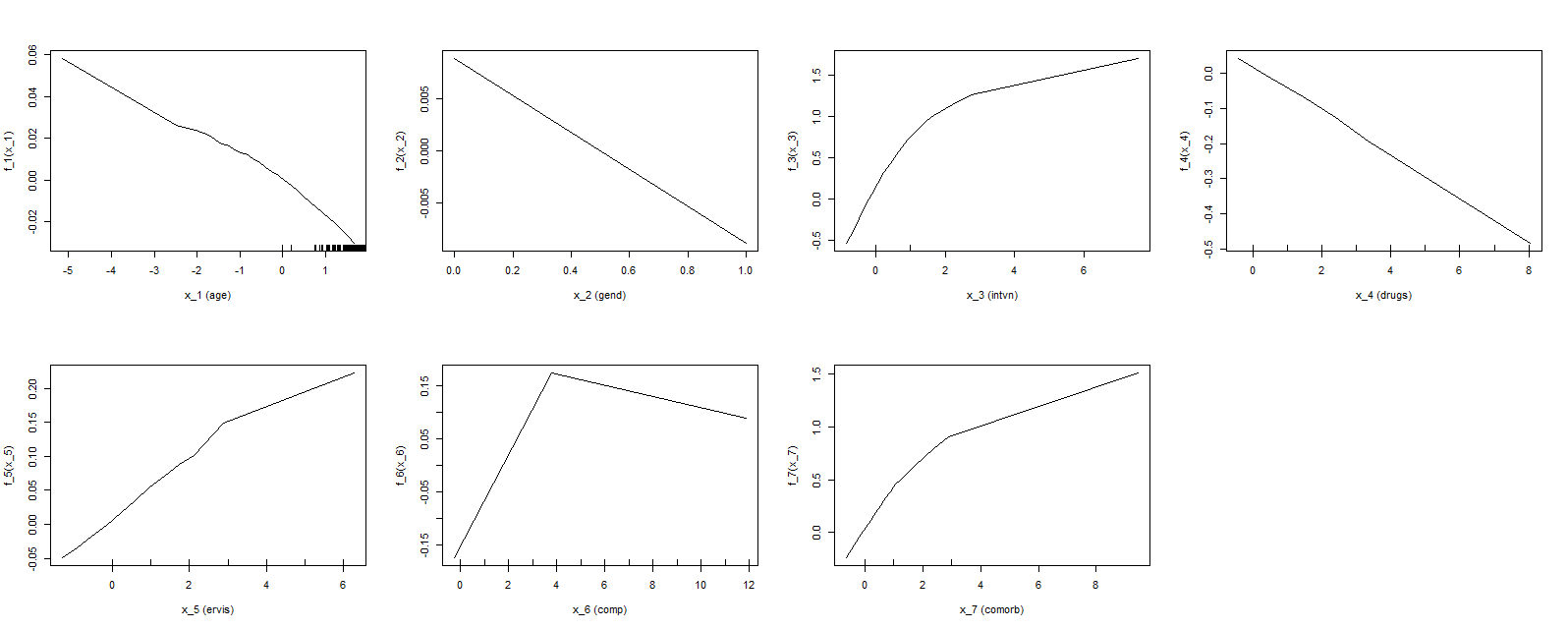
comp<-matrix(rep(c(-0.15, 0.1),2),nrow=2)  
comp[2,1]-comp[1,1]

## [1] 0.25

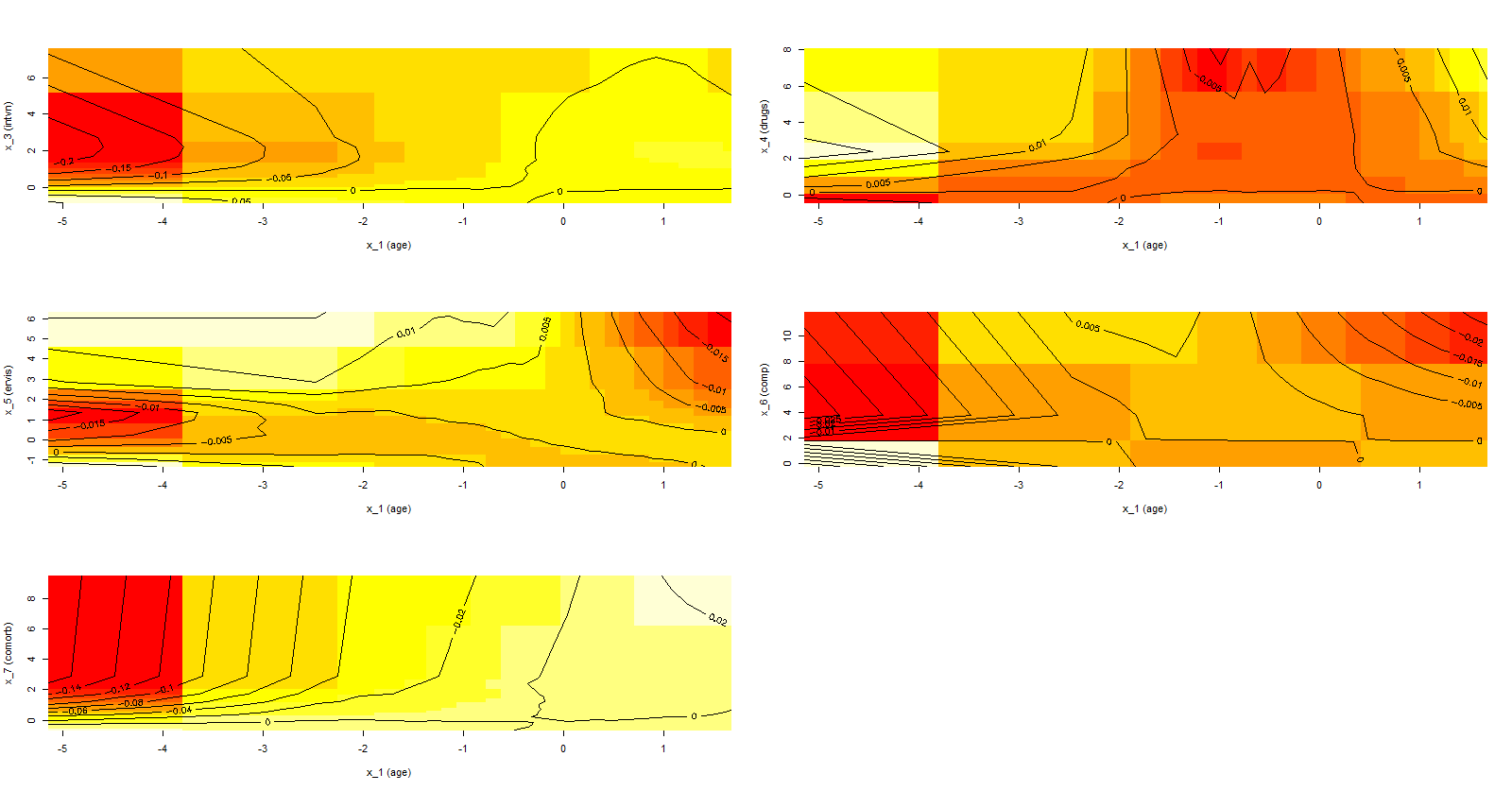
comorb<-matrix(rep(c(0, 1.5),2),nrow=2)  
comorb[2,1]-comorb[1,1]

## [1] 1.5

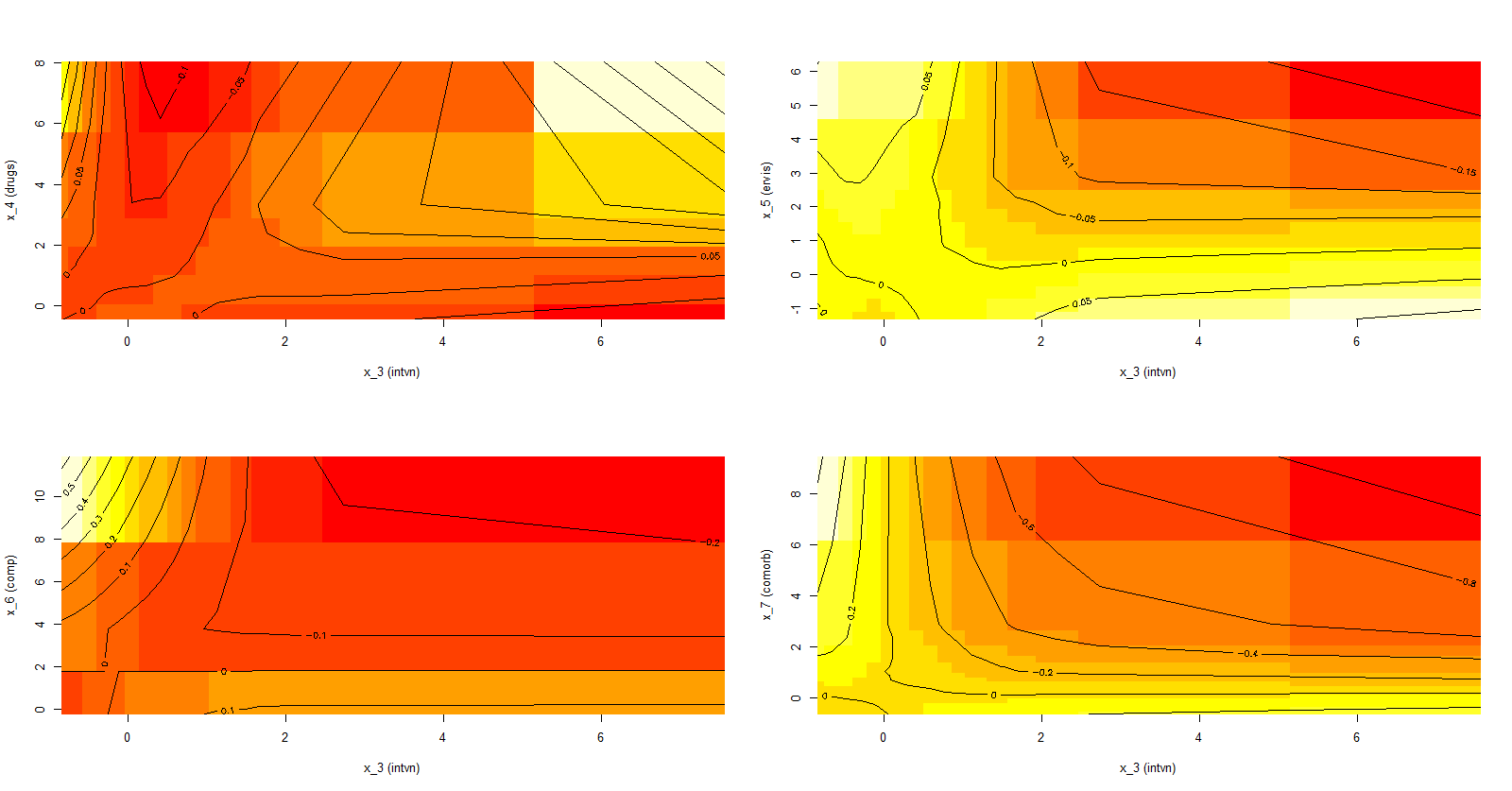
# second order interactions  
par(mfrow=c(3,2)) ## This creates 2nd-order interaction ALE plots



a<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(1,3), K=50, NA.plot = TRUE)  
b<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(1,4), K=50, NA.plot = TRUE)  
c<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(1,5), K=50, NA.plot = TRUE)  
d<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(1,6), K=50, NA.plot = TRUE)  
e<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(1,7), K=50, NA.plot = TRUE)  
  
par(mfrow=c(2,2))



a<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(3,4), K=50, NA.plot = TRUE)  
b<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(3,5), K=50, NA.plot = TRUE)  
c<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(3,6), K=50, NA.plot = TRUE)  
d<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(3,7), K=50, NA.plot = TRUE)



intvncomp<-matrix(c(0,0.4, 0.1, -0.2),byrow=T, nrow=2)  
intvncomp

## [,1] [,2]  
## [1,] 0.0 0.4  
## [2,] 0.1 -0.2

intvncomp[2,2]-intvncomp[1,2]

## [1] -0.6

intvncomp[2,1]-intvncomp[1,1]

## [1] 0.1

# invtn and high comp has negative effect on cost  
  
intvncomorb<-matrix(c(0,0.6, 0.2, -1),byrow=T, nrow=2)  
intvncomorb

## [,1] [,2]  
## [1,] 0.0 0.6  
## [2,] 0.2 -1.0

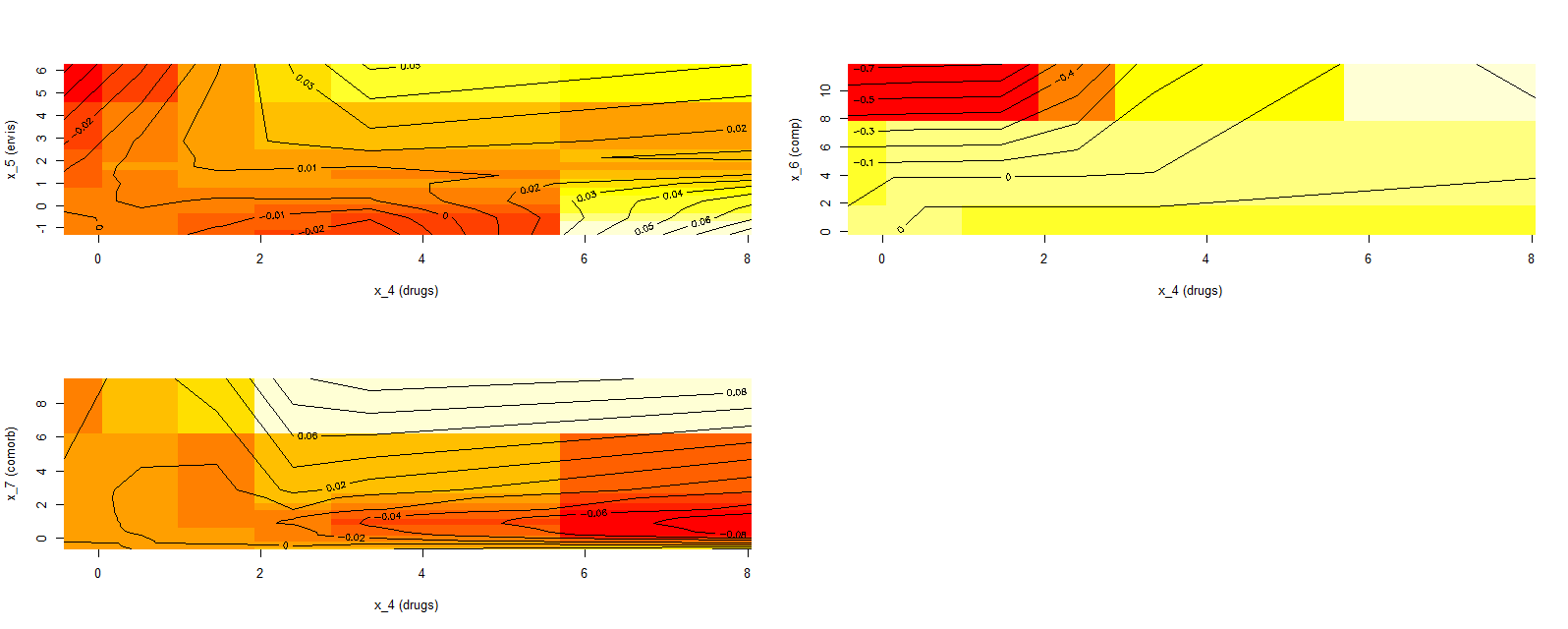
intvncomorb[2,2]-intvncomorb[1,2]

## [1] -1.6

intvncomorb[2,1]-intvncomorb[1,1]

## [1] 0.2

# invtn and comp has negative effect on cost  
  
par(mfrow=c(2,2))  
a<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(4,5), K=50, NA.plot = TRUE)  
b<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(4,6), K=50, NA.plot = TRUE)  
c<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(4,7), K=50, NA.plot = TRUE)  
  
par(mfrow=c(1,2))



drugscomp<-matrix(c(0,0.7, 0, 0),byrow=T, nrow=2)  
drugscomp

## [,1] [,2]  
## [1,] 0 0.7  
## [2,] 0 0.0

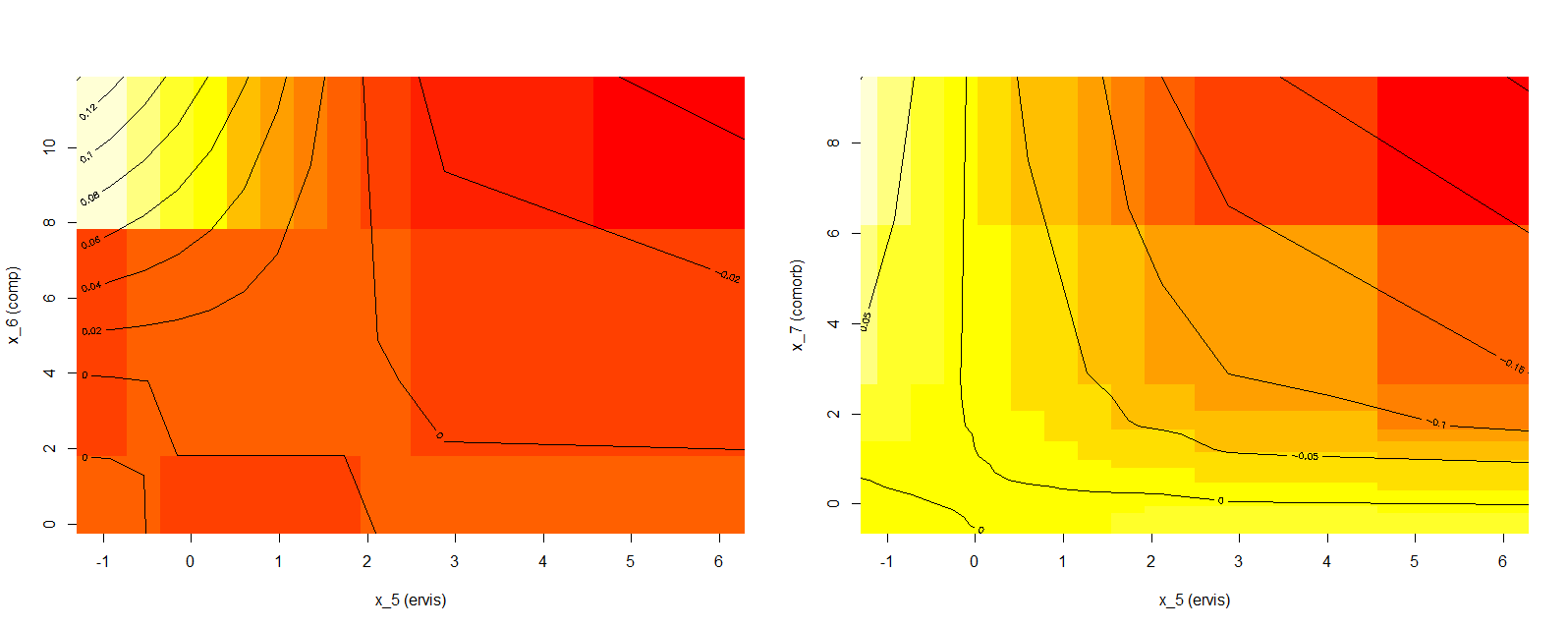
drugscomp [2,2]- drugscomp [1,2]

## [1] -0.7

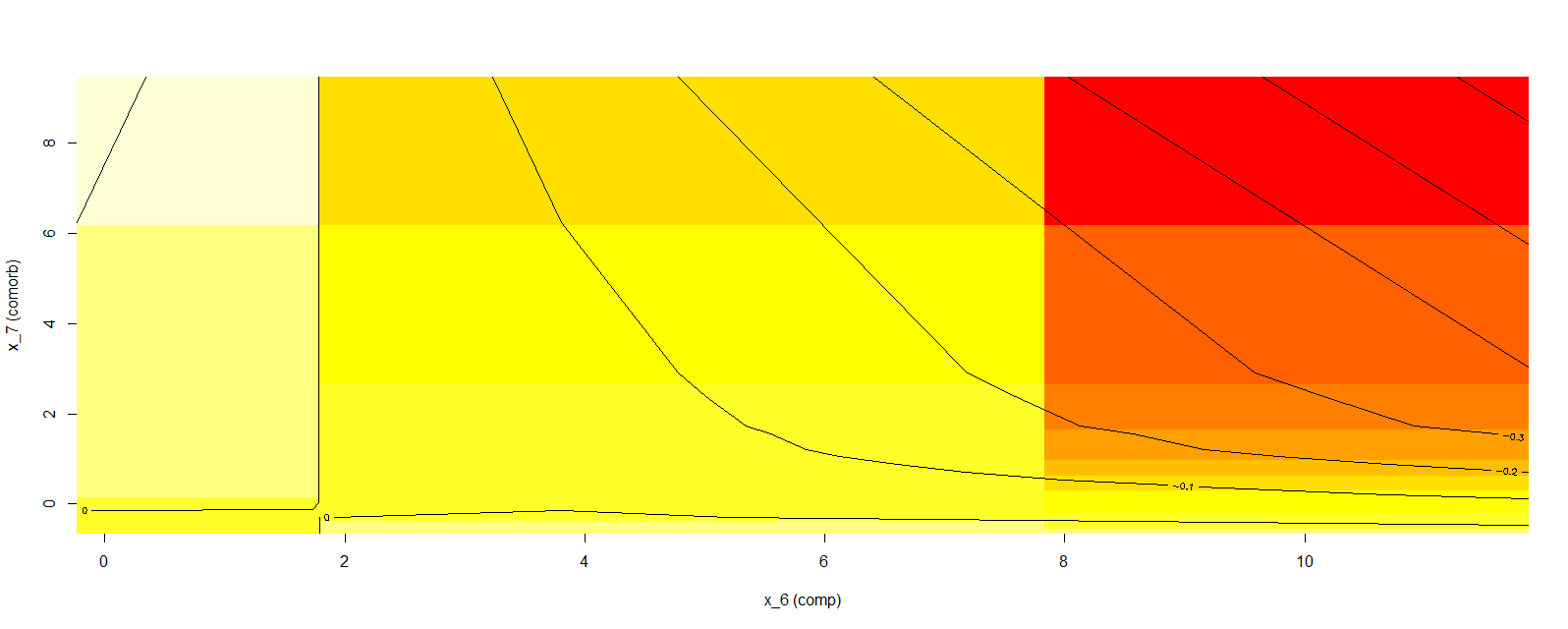
drugscomp [2,1]- drugscomp [1,1]

## [1] 0

a<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(5,6), K=50, NA.plot = TRUE)  
b<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(5,7), K=50, NA.plot = TRUE)



par(mfrow=c(1,1))  
a<-ALEPlot(standheart[,2:8], nn.fit, pred.fun=yhat, J=c(6,7), K=50, NA.plot = TRUE)



compcomorb<-matrix(c(0,0, 0, -0.6),byrow=T, nrow=2)  
compcomorb

## [,1] [,2]  
## [1,] 0 0.0  
## [2,] 0 -0.6

compcomorb[2,2]-compcomorb[1,2]

## [1] -0.6

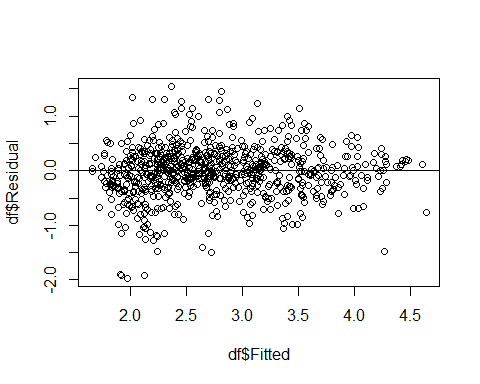
compcomorb[2,1]-compcomorb[1,1]

## [1] 0

# comp and comorb has negative effect on cost

# d.

par(mfrow=c(1,1))  
resid<-standheart$cost-predict(nn.fit, standheart)  
df<-data.frame(cbind(resid,predict(nn.fit, standheart)))  
colnames(df)<-c("Residual", "Fitted")  
df <- df[order(df$Residual),]  
plot(df$Fitted, df$Residual)  
abline(h=0)

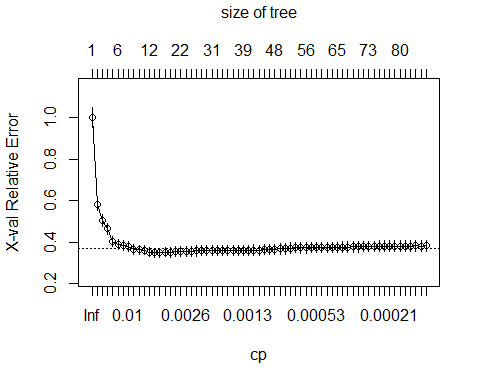


# the variance seems to be random

# Problem 3

# a.,b.,c.

library(rpart)  
control <- rpart.control(minbucket = 5, cp = 0.0001, maxsurrogate = 0, usesurrogate = 0, xval = 10)   
# minbucket minimum number of obsv, cp is complexity parameter  
orgtree <- rpart(cost ~ .,unstandheart, method = "anova", control = control)  
plotcp(orgtree) #plot of CV r^2 vs. size



Nrep<-100 #number of replicates of CV  
n<-nrow(orgtree$cptable)  
y<-unstandheart$cost  
xerror=matrix(0,n,Nrep)  
rownames(xerror)<-round(orgtree$cptable[,1],8)  
# rownames are cp paramter  
  
#running 10 replicates of 10-fold cross validation  
for (j in 1:Nrep){  
 orgtreecv <- rpart(cost ~ .,unstandheart, method = "anova", control = control)   
 df<-data.frame(orgtreecv$cptable)  
 xerror[ ,j]<-df[ ,4] # fill xerror table of 10 replicates  
}  
xerrorAve<-apply(xerror,1,mean); xerrorAve #averaged mean square CV error

## 0.4393807 0.07144171 0.05828524 0.05516294 0.01767365 0.01260299   
## 1.0029009 0.5852951 0.5024934 0.4603152 0.4072424 0.3901684   
## 0.01166647 0.00873489 0.00677614 0.00664712 0.00622922 0.00409089   
## 0.3845768 0.3724062 0.3635302 0.3595428 0.3582976 0.3502537   
## 0.0037669 0.00351925 0.00334926 0.00289757 0.00263329 0.00262504   
## 0.3494911 0.3494625 0.3497921 0.3527146 0.3554427 0.3564236   
## 0.00250947 0.00230362 0.0021852 0.0020556 0.00196347 0.00189675   
## 0.3563962 0.3569404 0.3572820 0.3569391 0.3576197 0.3577270   
## 0.00171146 0.00155697 0.00151823 0.00146593 0.00145187 0.00137418   
## 0.3589468 0.3605224 0.3621892 0.3627721 0.3630399 0.3639785   
## 0.00132513 0.00118913 0.00114698 0.0010195 0.0009154 0.00090817   
## 0.3648849 0.3657253 0.3669340 0.3681248 0.3701045 0.3711485   
## 0.00081261 0.00066945 0.00060069 0.00059844 0.00058423 0.00055128   
## 0.3718753 0.3746626 0.3768320 0.3777902 0.3779401 0.3787469   
## 0.00053289 0.00052382 0.00051826 0.00047742 0.00047174 0.00040168   
## 0.3793237 0.3796081 0.3797229 0.3803224 0.3806945 0.3815406   
## 0.00038333 0.00036051 0.00034433 0.00033608 0.00031622 0.00027854   
## 0.3823028 0.3826707 0.3829598 0.3832647 0.3836108 0.3842125   
## 0.00026468 0.00023864 0.00021642 0.00019866 0.00018737 0.00018468   
## 0.3847090 0.3850154 0.3854550 0.3858237 0.3860584 0.3861700   
## 0.00018094 0.00017588 0.00015448 0.00012712 1e-04   
## 0.3862264 0.3862572 0.3864372 0.3866088 0.3867746

xerrorAve[which(xerrorAve==min(xerrorAve))]

## 0.00351925   
## 0.3494625

xerrorsd<-apply(xerror,1,sd); xerrorsd

## 0.4393807 0.07144171 0.05828524 0.05516294 0.01767365 0.01260299   
## 0.001375674 0.008716364 0.012900649 0.013630304 0.008209924 0.006886988   
## 0.01166647 0.00873489 0.00677614 0.00664712 0.00622922 0.00409089   
## 0.007588257 0.008171864 0.006745424 0.006123641 0.005839698 0.006472298   
## 0.0037669 0.00351925 0.00334926 0.00289757 0.00263329 0.00262504   
## 0.006850948 0.006838220 0.006889681 0.007184136 0.006951037 0.007566021   
## 0.00250947 0.00230362 0.0021852 0.0020556 0.00196347 0.00189675   
## 0.007466944 0.007911979 0.008520085 0.007947231 0.008243247 0.008545998   
## 0.00171146 0.00155697 0.00151823 0.00146593 0.00145187 0.00137418   
## 0.008724519 0.009021326 0.009092789 0.009218568 0.009144522 0.009194246   
## 0.00132513 0.00118913 0.00114698 0.0010195 0.0009154 0.00090817   
## 0.008873610 0.009170735 0.009754347 0.009282113 0.009220108 0.009229411   
## 0.00081261 0.00066945 0.00060069 0.00059844 0.00058423 0.00055128   
## 0.009306097 0.009472182 0.009701805 0.009772278 0.009676426 0.009806289   
## 0.00053289 0.00052382 0.00051826 0.00047742 0.00047174 0.00040168   
## 0.009711185 0.009679421 0.009601095 0.009659902 0.009732011 0.009766899   
## 0.00038333 0.00036051 0.00034433 0.00033608 0.00031622 0.00027854   
## 0.009982371 0.009976910 0.009826202 0.009794237 0.009848978 0.009934242   
## 0.00026468 0.00023864 0.00021642 0.00019866 0.00018737 0.00018468   
## 0.009969685 0.009983853 0.009889153 0.009880594 0.009859584 0.009814557   
## 0.00018094 0.00017588 0.00015448 0.00012712 1e-04   
## 0.009820089 0.009828633 0.009865921 0.009858072 0.009841410

r2<-1-xerrorAve; r2[which(xerrorAve==min(xerrorAve))] #CV r^2

## 0.00351925   
## 0.6505375

# cv r2 of 65%  
  
#prune back to optimal size, according to plot of CV 1-r^2  
orgtree2 <- prune(orgtree, cp=0.00351925) #approximately the best is with complexity parameter of 0.0035195  
orgtree2$variable.importance

## intvn comorb ervis comp age   
## 323.406063 49.545544 3.655447 2.206868 2.032086

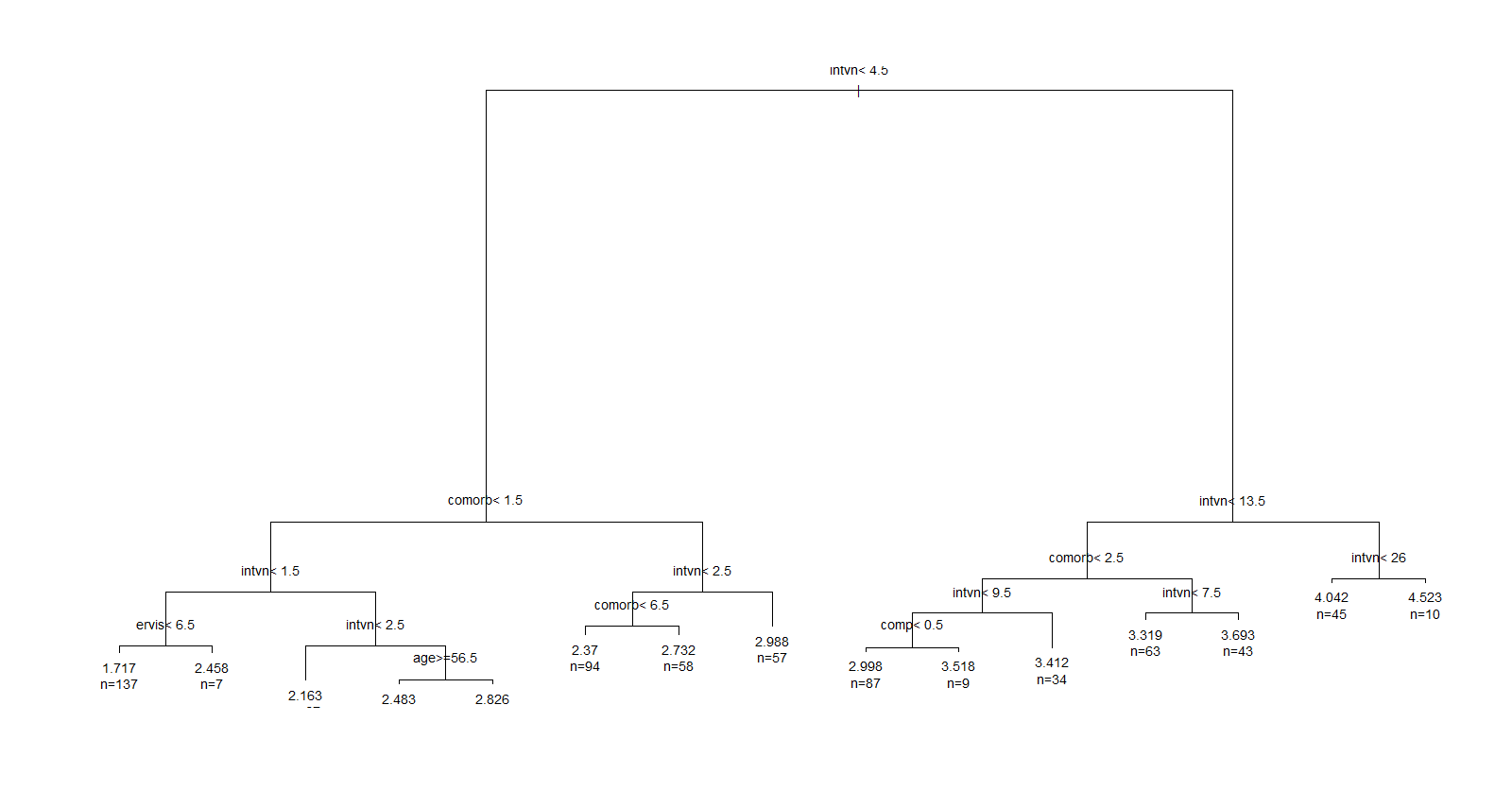
# training r2 of 70.597  
1-orgtree2$cptable[nrow(orgtree2$cptable),3] #shows training and CV 1-r^2, and other things

## [1] 0.7059781

# pruned tree plot and text  
orgtree2

## n= 788   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 788 539.4587000 2.731718   
## 2) intvn< 4.5 497 202.2420000 2.312051   
## 4) comorb< 1.5 288 123.4154000 2.074830   
## 8) intvn< 1.5 144 47.9762700 1.753385   
## 16) ervis< 6.5 137 41.5799900 1.717370 \*  
## 17) ervis>=6.5 7 2.7408280 2.458241 \*  
## 9) intvn>=1.5 144 45.6809900 2.396275   
## 18) intvn< 2.5 67 19.9185100 2.163335 \*  
## 19) intvn>=2.5 77 18.9636800 2.598962   
## 38) age>=56.5 51 11.7883100 2.482970 \*  
## 39) age< 56.5 26 5.1432840 2.826484 \*  
## 5) comorb>=1.5 209 40.2867600 2.638940   
## 10) intvn< 2.5 152 20.2993700 2.508147   
## 20) comorb< 6.5 94 12.3044400 2.369842 \*  
## 21) comorb>=6.5 58 3.2828250 2.732295 \*  
## 11) intvn>=2.5 57 10.4531800 2.987721 \*  
## 3) intvn>=4.5 291 100.1889000 3.448469   
## 6) intvn< 13.5 236 59.5997500 3.289783   
## 12) comorb< 2.5 130 34.6069800 3.142323   
## 24) intvn< 9.5 96 22.9374600 3.046642   
## 48) comp< 0.5 87 18.7280300 2.997876 \*  
## 49) comp>=0.5 9 2.0025560 3.518043 \*  
## 25) intvn>=9.5 34 8.3091180 3.412483 \*  
## 13) comorb>=2.5 106 18.6991900 3.470630   
## 26) intvn< 7.5 63 8.9787070 3.318678 \*  
## 27) intvn>=7.5 43 6.1346320 3.693258 \*  
## 7) intvn>=13.5 55 9.1467180 4.129374   
## 14) intvn< 26 45 6.7027050 4.041791 \*  
## 15) intvn>=26 10 0.5455236 4.523494 \*

par(cex=.9); plot(orgtree2, uniform=F); text(orgtree2, use.n = T); par(cex=1)

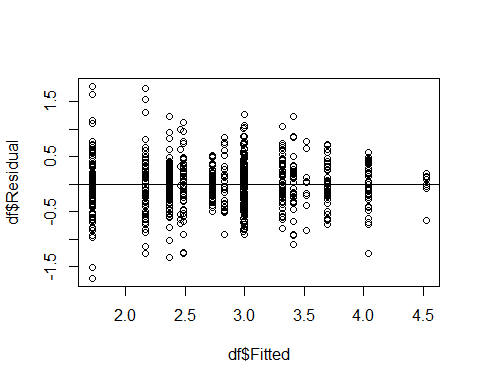


# training MSE 0.2012  
sum((unstandheart$cost-predict(orgtree2, unstandheart))^2)/nrow(unstandheart)

## [1] 0.2012851

# d.

par(mfrow=c(1,1))  
resid<-unstandheart$cost-predict(orgtree2, unstandheart)  
df<-data.frame(cbind(resid,predict(orgtree2, unstandheart)))  
colnames(df)<-c("Residual", "Fitted")  
df <- df[order(df$Residual),]  
plot(df$Fitted, df$Residual)  
abline(h=0)



# errors seems centered around 0

# Problem 4

# a.

glass<-read.xlsx("HW2\_data.xls", sheetName= 2)  
# first column has row index get rid of it  
glass<-glass[,2:11]  
  
# dataset of standardized predictors and unstandardized predictors  
unstandglass<-glass  
standglass<-glass  
standglass[ , c(1:9)]<-scale(standglass[ , c(1:9)], center=T, scale=T)  
  
# using misclassification rate to decide between models,   
# use class in predict function   
Nrep<-10 #number of replicates of CV  
K<-10 #K-fold CV on each replicate  
n.models = 9 #number of different models to fit  
n=nrow(standglass)  
y<-standglass$type  
yhat=matrix('',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=0.05, maxit=100, 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=0.1, 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=0.15, 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=30, decay=0.05, 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=30, decay=0.1, 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=30, decay=0.15, 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=40, decay=0.05, 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=40, decay=0.1, 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=40, decay=0.15, 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 lowest of 0.2735981

## [1] 0.2728972 0.2906542 0.3032710 0.2686916 0.2808411 0.3140187 0.2719626  
## [8] 0.2855140 0.3028037

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

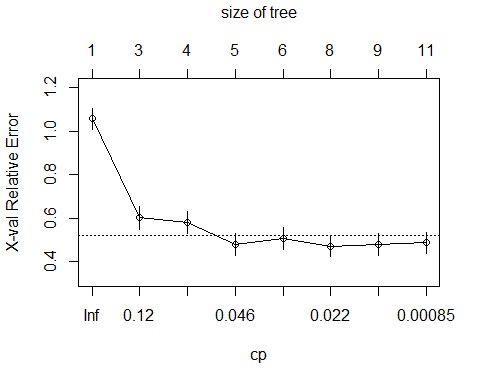
## [1] 0.01709142 0.01689152 0.01726796 0.01149908 0.01517393 0.01389701  
## [7] 0.01703455 0.01517393 0.01407052

# best model parameters are size=30 and decay=0.05  
nn.fit2<-nnet(type~.,standglass, linout=F, skip=F, size=30, decay=0.05, maxit=10, trace=F)  
# training misclass rate 30.37  
1-sum(glass$type==predict(nn.fit2,standglass, type="class"))/nrow(standglass)

## [1] 0.3037383

# b.

control <- rpart.control(minbucket = 5, cp = 0.0001, maxsurrogate = 0, usesurrogate = 0, xval = 10)   
# minbucket minimum number of obsv, cp is complexity parameter  
classtree <- rpart(type ~ .,unstandglass, method = "class", control = control)  
plotcp(classtree) #plot of CV r^2 vs. size



Nrep<-100 #number of replicates of CV  
n<-nrow(classtree$cptable)  
y<-unstandglass$type  
xerror=matrix(0,n,Nrep)  
rownames(xerror)<-round(classtree$cptable[,1],7)  
# rownames are cp paramter  
  
#running 10 replicates of 10-fold cross validation  
for (j in 1:Nrep){  
 classtreecv <- rpart(type ~ .,unstandglass, method = "class", control = control)  
 df<-data.frame(classtreecv$cptable)  
 xerror[ ,j]<-df[ ,4] # fill xerror table of 10 replicates  
}  
xerrorAve<-apply(xerror,1,mean); xerrorAve #averaged mean square CV error

## 0.2065217 0.0724638 0.057971 0.0362319 0.0326087 0.0144928 0.0072464   
## 1.0473188 0.5986957 0.5823913 0.5031884 0.5085507 0.4617391 0.4732609   
## 1e-04   
## 0.4732609

xerrorAve[which(xerrorAve==min(xerrorAve))]

## 0.0144928   
## 0.4617391

xerrorsd<-apply(xerror,1,sd); xerrorsd

## 0.2065217 0.0724638 0.057971 0.0362319 0.0326087 0.0144928   
## 0.041126755 0.005986157 0.016975679 0.022829717 0.020762731 0.023713138   
## 0.0072464 1e-04   
## 0.024322975 0.025804334

# cv misclass rate is 0.29775 with optimal cp is 0.0144928  
xerrorAve\*(1-max(table(y))/nrow(unstandglass))

## 0.2065217 0.0724638 0.057971 0.0362319 0.0326087 0.0144928 0.0072464   
## 0.6753738 0.3860748 0.3755607 0.3244860 0.3279439 0.2977570 0.3051869   
## 1e-04   
## 0.3051869

#prune back to optimal size, according to plot of CV 1-r^2  
classtree2 <- prune(classtree, cp=0.0144928) #approximately the best size pruned tree is with complexity  
# parameter of 0.0144928  
# how much gini decreases by  
classtree2$variable.importance

## Ba Mg Al Ca Na RI   
## 26.044912 17.020468 16.085776 8.668054 6.116667 5.534158

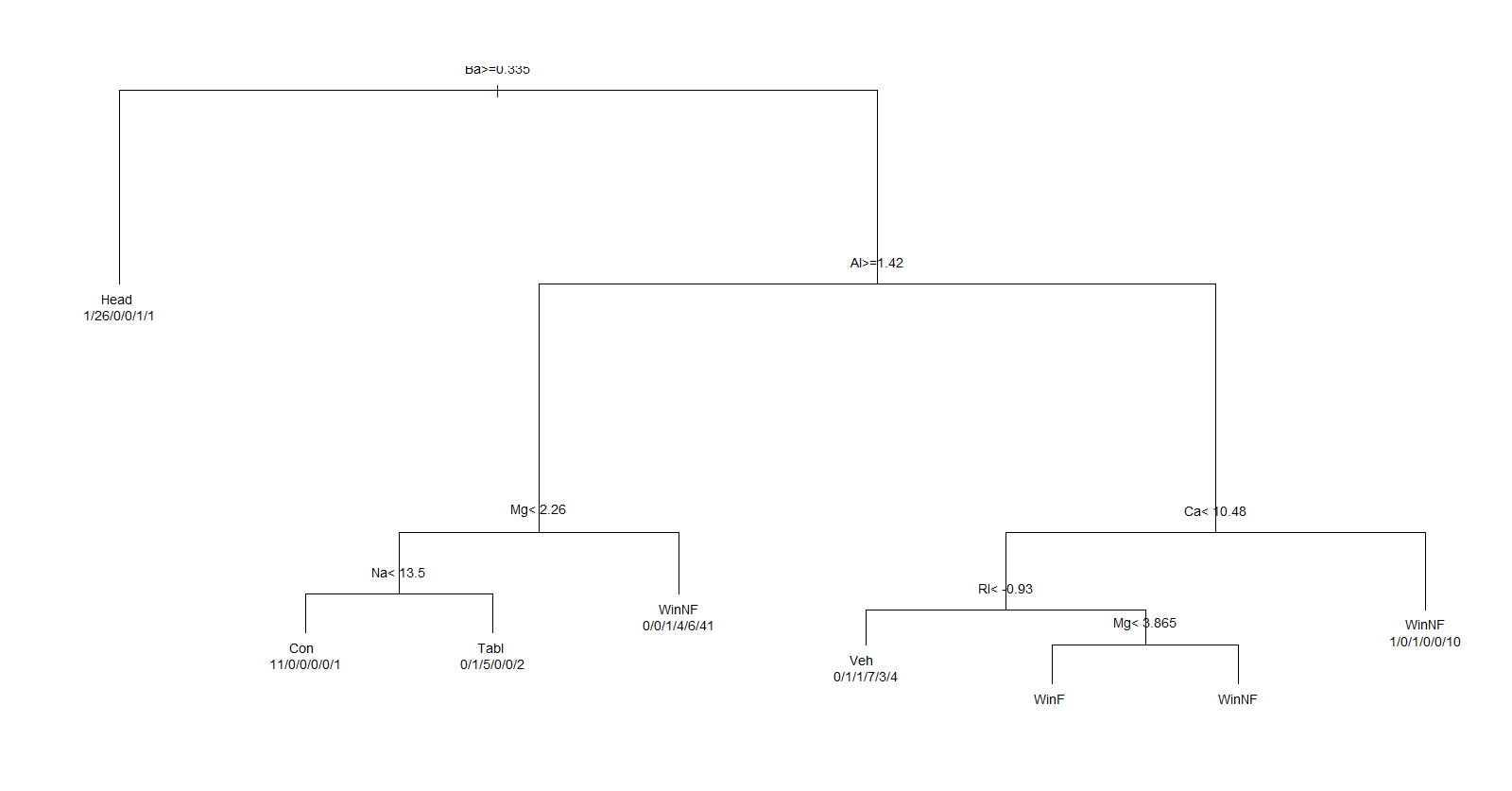
# training misclass % is 0.228972  
classtree2$cptable[nrow(classtree2$cptable),3]\*(1-max(table(y))/nrow(unstandglass))

## [1] 0.228972

# plot and text of pruned tree  
classtree2

## n= 214   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 214 138 WinNF (0.061 0.14 0.042 0.079 0.33 0.36)   
## 2) Ba>=0.335 29 3 Head (0.034 0.9 0 0 0.034 0.034) \*  
## 3) Ba< 0.335 185 110 WinNF (0.065 0.016 0.049 0.092 0.37 0.41)   
## 6) Al>=1.42 72 28 WinNF (0.15 0.014 0.083 0.056 0.083 0.61)   
## 12) Mg< 2.26 20 9 Con (0.55 0.05 0.25 0 0 0.15)   
## 24) Na< 13.495 12 1 Con (0.92 0 0 0 0 0.083) \*  
## 25) Na>=13.495 8 3 Tabl (0 0.12 0.62 0 0 0.25) \*  
## 13) Mg>=2.26 52 11 WinNF (0 0 0.019 0.077 0.12 0.79) \*  
## 7) Al< 1.42 113 50 WinF (0.0088 0.018 0.027 0.12 0.56 0.27)   
## 14) Ca< 10.48 101 38 WinF (0 0.02 0.02 0.13 0.62 0.21)   
## 28) RI< -0.93 16 9 Veh (0 0.062 0.062 0.44 0.19 0.25) \*  
## 29) RI>=-0.93 85 25 WinF (0 0.012 0.012 0.071 0.71 0.2)   
## 58) Mg< 3.865 77 18 WinF (0 0.013 0.013 0.065 0.77 0.14) \*  
## 59) Mg>=3.865 8 2 WinNF (0 0 0 0.12 0.12 0.75) \*  
## 15) Ca>=10.48 12 2 WinNF (0.083 0 0.083 0 0 0.83) \*

par(cex=.9); plot(classtree2, uniform=F); text(classtree2, use.n = T); par(cex=1)



# c.

library(MASS)  
Nrep<-20 #number of replicates of CV  
K<-10 #K-fold CV on each replicate  
n.models = 1 #number of different models to fit  
n=nrow(unstandglass)  
y<-unstandglass$type  
yhat=matrix('',n,n.models)  
MSC<-matrix(0,Nrep,n.models)  
  
for (j in 1:Nrep) {  
 Ind<-CVInd(n,K)  
 for (k in 1:K) {  
 out<-multinom(type~., unstandglass[-Ind[[k]],], trace=F)  
 yhat[Ind[[k]],1]<-as.character(predict(out,unstandglass[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

## [1] 0.3827103

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

## [1] 0.01258444

# cv 0.38 misclassified  
  
multi<-multinom(type~., unstandglass, trace=F)  
summary(multi)

## Call:  
## multinom(formula = type ~ ., data = unstandglass, trace = F)  
##   
## Coefficients:  
## (Intercept) RI Na Mg Al Si  
## Head -23.86095 11.4561798 11.532853 -31.328209 -10.08402 5.1109902  
## Tabl -19.10899 7.5881758 25.653266 -16.414504 -1.60811 0.2063631  
## Veh 34.08724 -1.8620049 3.866093 10.923929 -17.32159 -2.1412740  
## WinF -149.41546 -0.3477599 3.658852 10.029066 -17.99907 0.7919274  
## WinNF 140.13953 -0.1193227 1.362369 5.416969 -16.36071 -2.1766024  
## K Ca Ba Fe  
## Head -19.109099 -46.035166 -18.618012 -316.76298  
## Tabl -196.817343 -31.466105 -54.217377 -564.01014  
## Veh 2.924413 7.516091 4.823409 15.35464  
## WinF 5.505606 4.933656 5.619909 14.80819  
## WinNF 2.861293 1.638547 1.881901 17.18487  
##   
## Std. Errors:  
## (Intercept) RI Na Mg Al Si K  
## Head 0.08914746 1.4624187 3.218191 2.685662 4.356877 0.8545688 4.1004300  
## Tabl 0.11907880 3.4707838 7.082766 2.904807 4.366013 1.6523381 0.2821726  
## Veh 0.08524063 1.0693860 1.489607 2.514686 3.959844 0.5713682 4.3948929  
## WinF 0.07253254 0.9943932 1.379273 2.167435 3.817599 0.5067949 4.1398757  
## WinNF 0.09390767 0.9628696 1.296499 2.038527 3.675437 0.4793301 3.8422423  
## Ca Ba Fe  
## Head 3.340707 3.932839 1.286248e-01  
## Tabl 2.843409 3.569727 6.852118e-14  
## Veh 2.487713 4.548661 4.271674e+00  
## WinF 2.156048 3.109834 3.699431e+00  
## WinNF 2.021691 2.044751 3.404173e+00  
##   
## Residual Deviance: 255.7437   
## AIC: 355.7437

# training 26.64% misclassified  
1-sum(glass$type==predict(multi, unstandglass,type="class"))/nrow(unstandglass)

## [1] 0.2663551