HW2

2023-02-06

Question 1

a

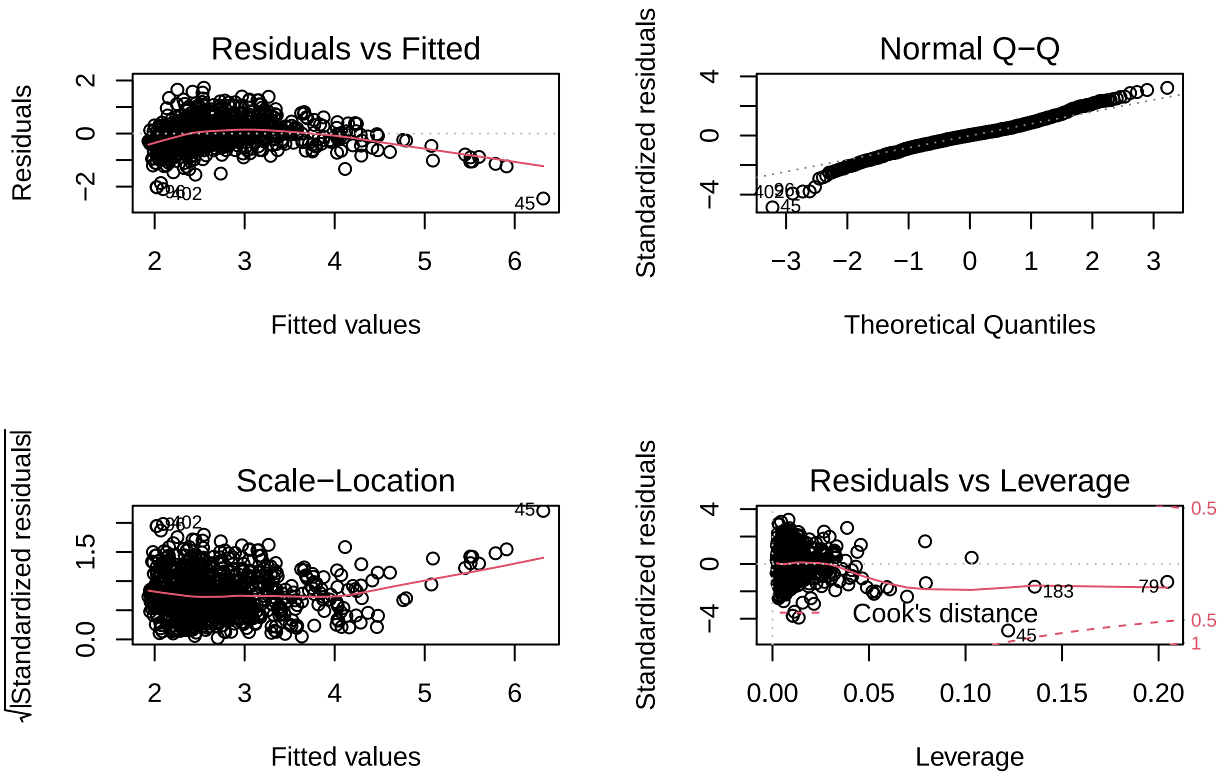
```
## New names:
## * `` -> ...1

##      age     gend    intvn    drugs    ervis     comp   comorb      dur
## 1.032749 1.018121 1.238005 1.409274 1.556264 1.058985 1.349956 1.399433

##
## Call:
## lm(formula = cost ~ ., data = dat_scaled)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.44852 -0.30093  0.01049  0.28276  1.72581
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.73172    0.01914 142.723  < 2e-16 ***
## age         -0.02981    0.01946  -1.532   0.1260
## gend        -0.02811    0.01932  -1.455   0.1462
## intvn        0.49125    0.02131  23.053  < 2e-16 ***
## drugs       -0.02736    0.02274  -1.203   0.2291
## ervis        0.05917    0.02389   2.477   0.0135 *
## comp         0.08114    0.01971   4.117 4.25e-05 ***
## comorb       0.13619    0.02225   6.120 1.48e-09 ***
## dur          0.14729    0.02266   6.501 1.43e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5373 on 779 degrees of freedom
## Multiple R-squared:  0.5831, Adjusted R-squared:  0.5789
## F-statistic: 136.2 on 8 and 779 DF,  p-value: < 2.2e-16
```

As seen from the above model output, the R-squared is 0.5831, which means that out model is able to explain 58% of the variation in the data. The model seems to fit fairly well to the data, but some predictors are insignificant.

b

As seen from the above model output, dur, the number of comorbidities, and the number of interventions or procedures carried out are the top 3 variables with the highest influence on cost as their coefficients are the largest. Note the VIF output for each variable is close to 1, meaning little to no multicolinearity. Also note from the normal QQ plot below that our residuals are normally distributed. This makes our estimates of the regression coefficients reliable

c



As seen from the diagnostic plots above, the residual vs fitted plot shows that the residuals have non-linear patterns, which means that the linear model does not explain the non-linear relationships that was in the data, which is expected. Looking at the scale-location plot, we can see that there is a clear trend, which means that we have are violating the assumption of equal variance of the residuals for every level of independent variables.

Question 2

a

```
##   size decay Average CV Error over 3 replicates
## 1    1 0.001                          0.2232222
## 2    1 0.010                          0.2215376
## 3    1 0.100                          0.2245123
## 4    1 0.000                          0.2403943
## 5    2 0.001                          0.2127500
## 6    2 0.010                          0.2183798
## 7    2 0.100                          0.2135844
## 8    2 0.000                          0.2164361
## 9      3 0.001                          0.2241562
## 10     3 0.010                          0.2193953
## 11     3 0.100                          0.2236197
## 12     3 0.000                          0.2457677
```

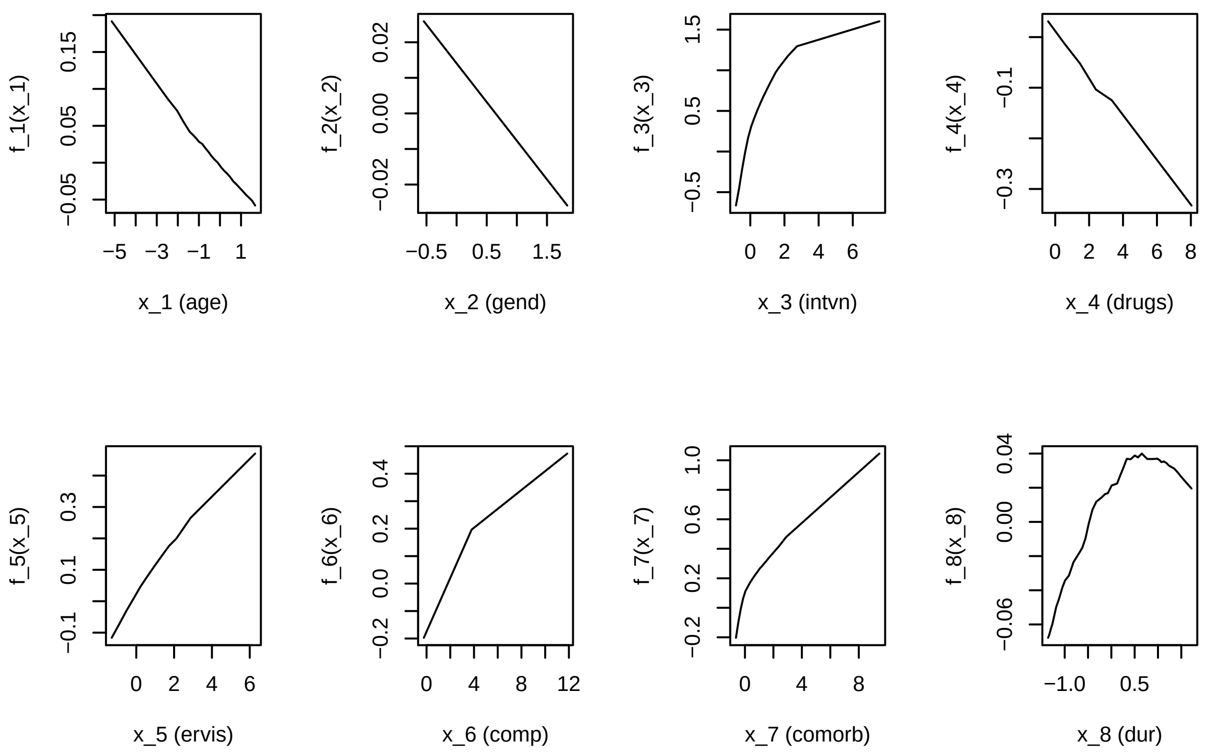
b

```
## [1] 0.7090515
```

As seen above, with the size = 2 and decay = 0.001 from CV, the neural network is able to explain 70.9051485 % of the variance.

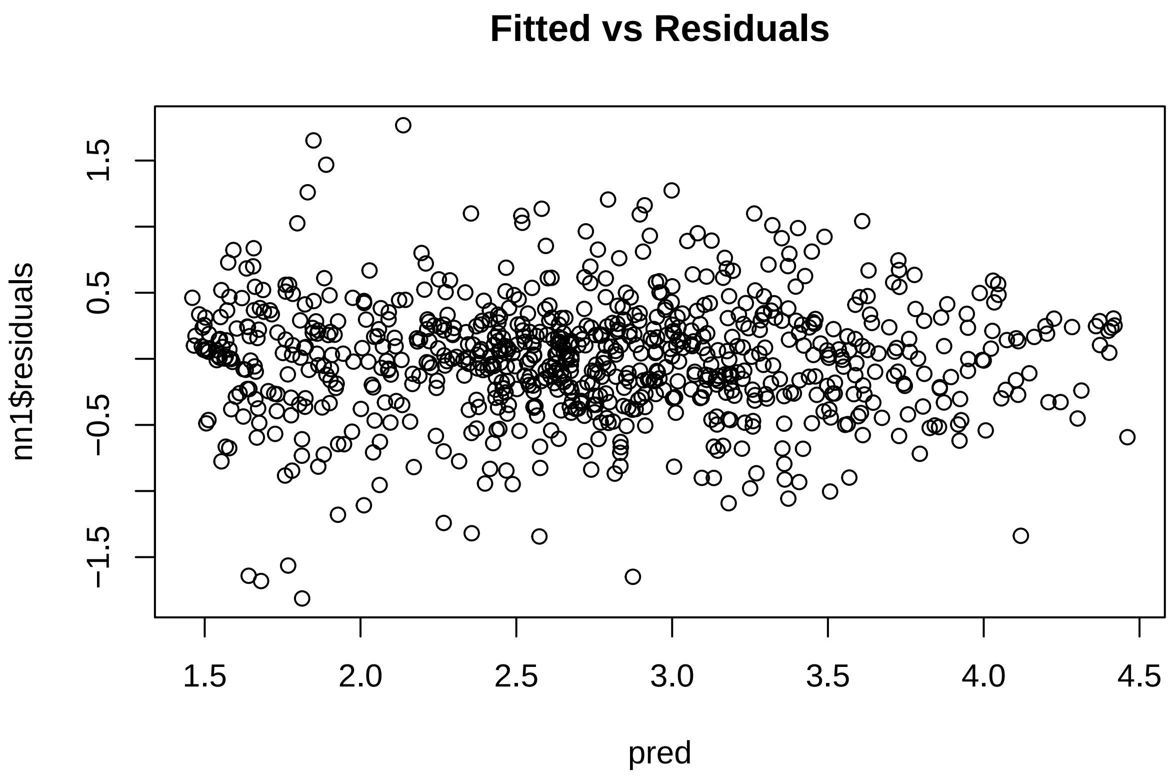
c

```
## Warning: package 'ALEPlot' was built under R version 4.0.5
```



As seen from the ALE plots, ervis, comp, and comorb, and intvn have the most significant effect on cost. They all have a positive relationship with cost.

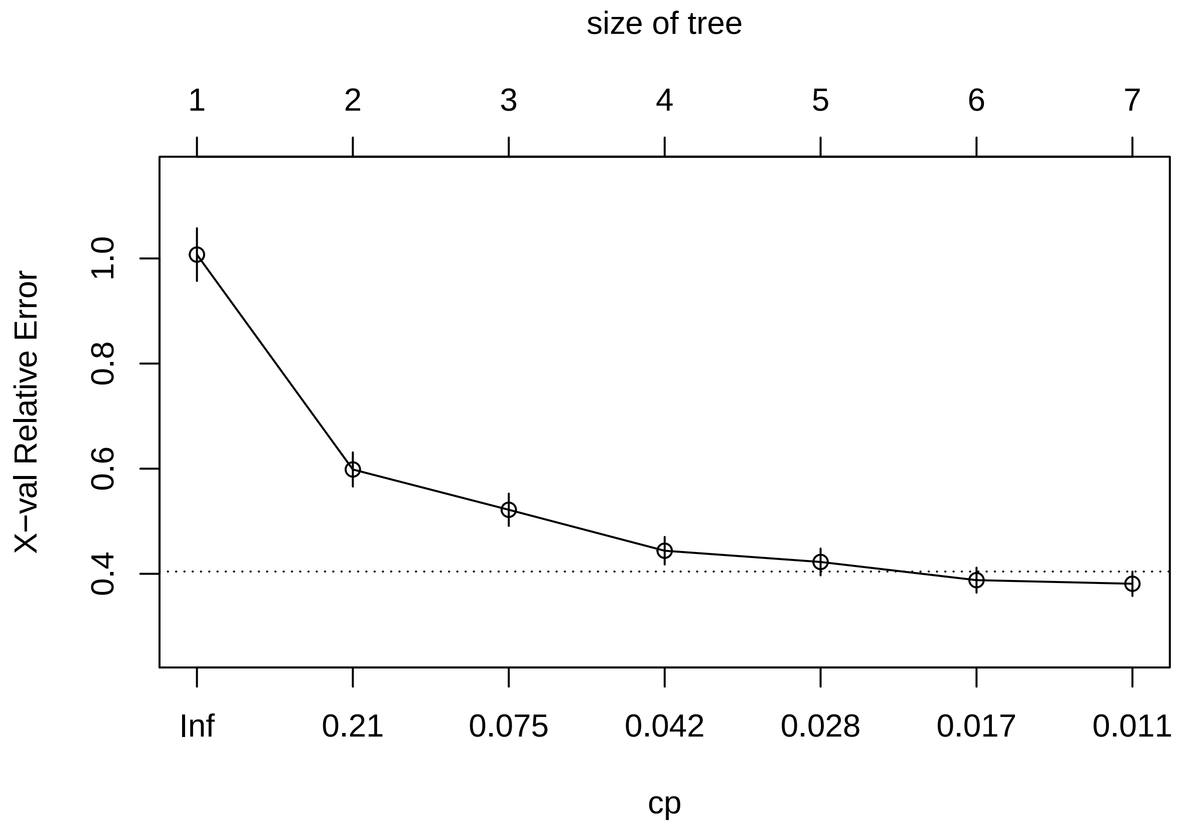
d



AS seen from the plot above, all non-linear relationships have been captured.

Question 3

a



```
## [1] 0.2403334
```

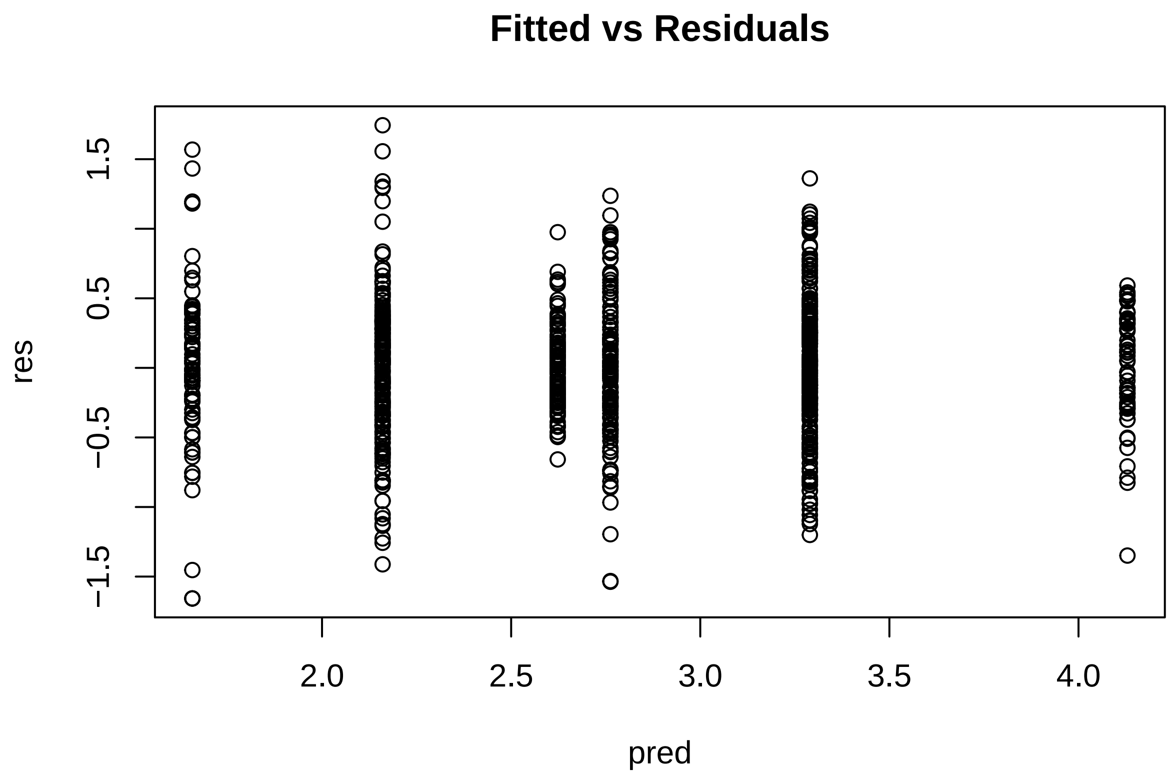
As seen above, the best cp is 0.01.

b

```
## [1] 0.6493849
```

As seen from the above plots and Rˆ2, the decision tree algorithm explains 64.9384886 % of the variation in the data. It is still very good, but not as good as the neural network.

c



d

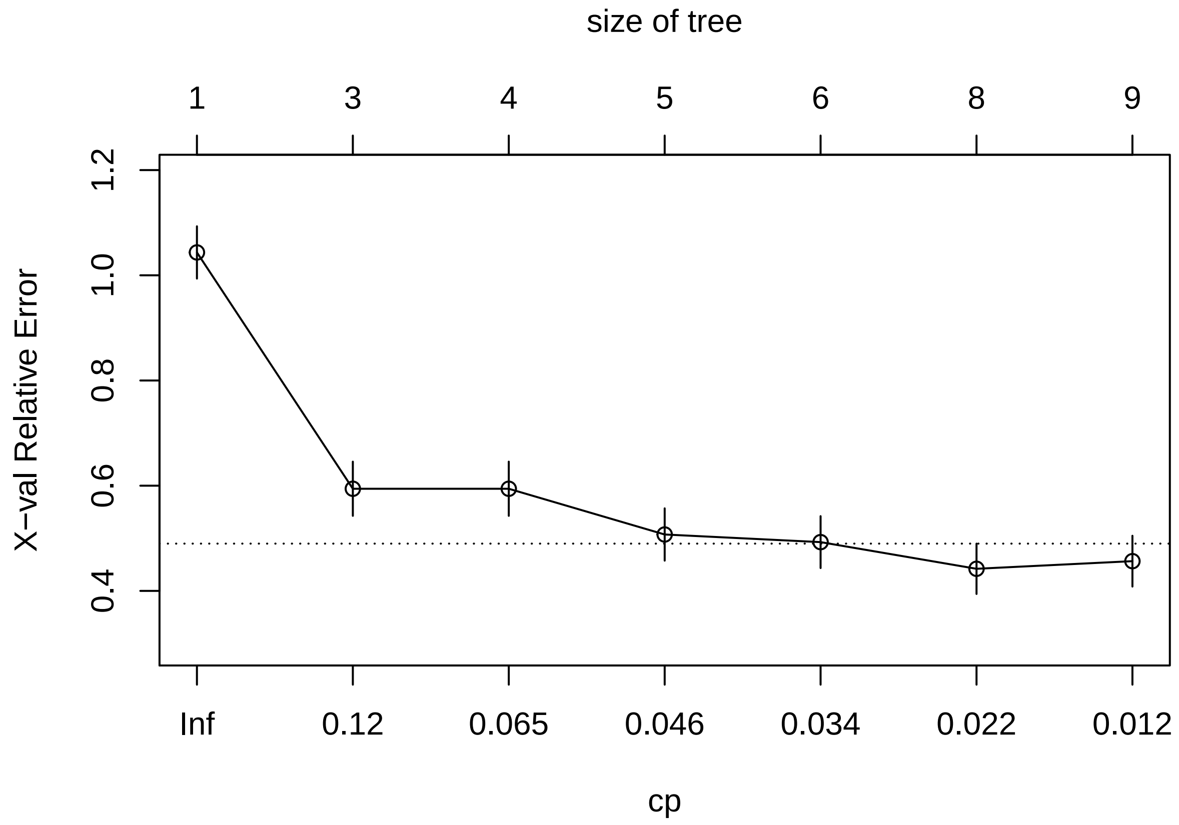
As seen above, the neural network can explain the most amount of variance in this dataset, which means it has the least MSE. Thus, we choose neural network.

Question 4

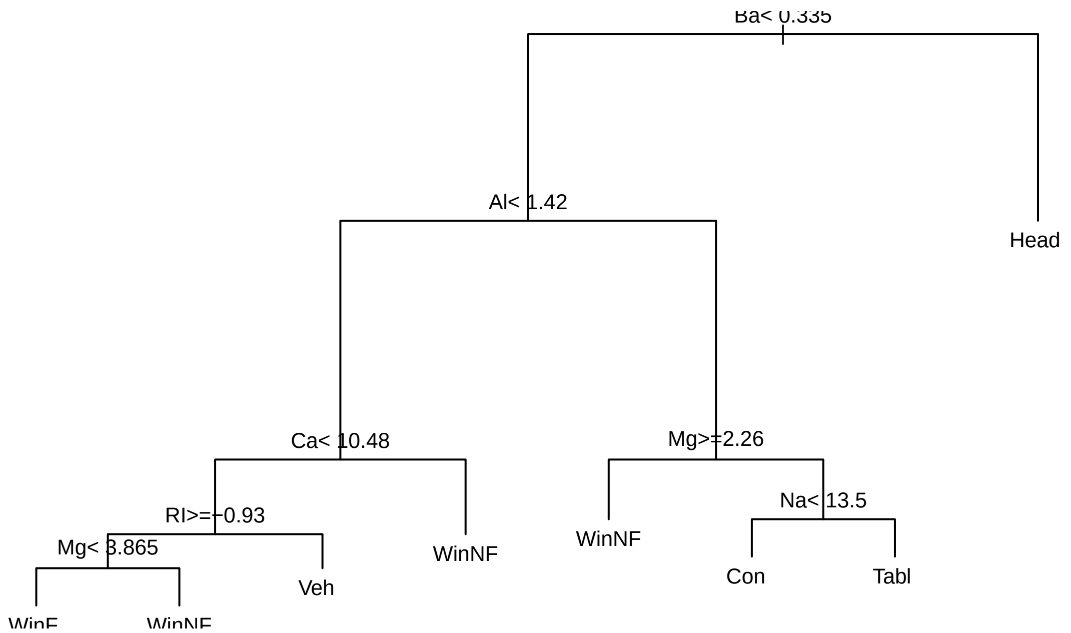
a

```
##   size decay Average CV Missclassification rate over 3 replicates
## 1    5 0.001                                             0.4112150
## 2    5 0.010                                             0.2897196
## 3    5 0.100                                             0.3411215
## 4    5 0.000                                             0.5981308
## 5   10 0.001                                             0.3084112
## 6   10 0.010                                             0.2663551
## 7   10 0.100                                             0.2757009
## 8   10 0.000                                             0.5654206
```

b



```
##
## Classification tree:
## rpart(formula = type ~ ., data = dat, method = "class", control = control)
##
## Variables actually used in tree construction:
## [1] Al Ba Ca Mg Na RI Si
##
## Root node error: 138/214 = 0.64486
##
## n= 214
##
##         CP nsplit rel error  xerror     xstd
## 1 0.206522      0   1.00000 1.04348 0.049733
## 2 0.072464      2   0.58696 0.59420 0.051536
## 3 0.057971      3   0.51449 0.59420 0.051536
## 4 0.036232      4   0.45652 0.50725 0.049733
## 5 0.032609      5   0.42029 0.49275 0.049357
## 6 0.014493      7   0.35507 0.44203 0.047855
## 7 0.010000      8   0.34058 0.45652 0.048314
```



```
##        Mg        Al        Ca        Ba        RI        Na         K        Si
## 32.375811 30.321865 28.990878 26.044912 23.469294 17.706035 15.305120  7.086511
```

```
##         CP     nsplit  rel error     xerror       xstd
## 0.02200000 7.00000000 0.35507246 0.44202899 0.04785471
```

```
##  rel error     xerror
## 0.01493295 0.01859000
```

```
## [1] 0.228972
```

c

```
## [1] 0.2616822
```

d

```
## [1] 0.2585670 0.2943925 0.3613707
```

As seen above, the best model is neural network with the smallest missclassification error

Appendix

```r
labs = knitr::all_labels()
labs = labs[!labs %in% c("setup","getlabels", "allcode")]
```

```r
########### QUESTION 1 begins here ###################
dat <- read_excel("HW2_data.xls")
dat <- dat[2:10]
dat$cost <- log10(dat$cost)

dat_scaled <- as.data.frame(cbind(dat$cost, scale(dat[2:9])))
names(dat_scaled) <- names(dat)

m1 <- lm(cost ~., data = dat_scaled)
vif(m1)
summary(m1)
par(mfrow = c(2, 2))
plot(m1)
########### QUESTION 2 begins here ###################
set.seed(123)
Nrep <- 3 #number of replicates of CV
K <- 10  #K-fold CV on each replicate
n.models <- 12 #number of different models to fit
size <- c(1,2,3)
decay <- c(0.001, 0.01, 0.1, 0)
n <- nrow(dat_scaled)
y <- dat_scaled$cost
yhat <- matrix(0, n, n.models)
MSE <- matrix(0, Nrep, n.models)
output <- data.frame()
model_num <- 1
for (s in size) {
  for (d in decay) {
    for (j in 1:Nrep) {
      cv_idx <- createFolds(dat_scaled$cost, k = K)
      for (k in 1:K) {
        out <-
          nnet(
            cost ~ .,
            data = dat_scaled[-cv_idx[[k]],],
            linout = T,
            skip = F,
            size = s,
            decay = d,
            maxit = 1000,
            trace = F
          )
         yhat[cv_idx[[k]], model_num] <- as.numeric(predict(out, dat_scaled[cv_idx[[k]], ]))
      } #end of k loop
      MSE[j, ] <- apply(yhat, 2, function(x)
        sum((y - x) ^ 2)) / n
    }#end of j loop
    output <- rbind(output, c(s, d))
    model_num <- model_num + 1
  }
}
MSEAve <- apply(MSE,2,mean)
output <- cbind(output, MSEAve)
names(output) <- c("size", "decay", "Average CV Error over 3 replicates")
output
set.seed(123)
nn1 <- nnet(cost ~ ., data = dat_scaled, linout=T, skip=F, size=2, decay=0.001, maxit=1000, trace=F)
yhat <- as.numeric(predict(nn1))
y <- dat_scaled$cost
mse <- mean((y - yhat)^2)
r2 <- 1 - mse/var(y); r2
library(ALEPlot)
yhat <- function(X.model, newdata) as.numeric(predict(X.model, newdata))
nn1 <- nnet(cost ~ ., data = dat_scaled, linout=T, skip=F, size=2, decay=0.001, maxit=1000, trace=F)
par(mfrow=c(2,4))
for (j in 1:8) {
  ALEPlot(data.frame(dat_scaled[, 2:9]), nn1, pred.fun=yhat, J=j, K=50, NA.plot = TRUE)
  }
 ## This creates main effect ALE plots for all 8 predictors
pred <- predict(nn1, dat_scaled)
res <- dat_scaled$cost - pred
plot(pred, nn1$residuals, main = "Fitted vs Residuals")
########### QUESTION 3 begins here ###################
set.seed(55)
control <- rpart.control(minbucket = 5, cp = 0.01)
out <- rpart(cost ~ ., data = dat_scaled, method = "anova", control = control)
plotcp(out)
control_opti <- rpart.control(minbucket = 5, cp = 0.017)
out_opti <- rpart(cost ~ ., data = dat_scaled, method = "anova", control = control)
Nrep <- 3 #number of replicates of CV
K <- 10  #K-fold CV on each replicate
n.models <- 1 #number of different models to fit
cp <- c(0.001,0.01,0.1,0)
n <- nrow(dat_scaled)
y <- dat_scaled$cost
yhat <- matrix(0, n, n.models)
MSE <- matrix(0, Nrep, n.models)
model_num <- 1
for (j in 1:Nrep) {
  cv_idx <- createFolds(dat_scaled$cost, k = K)
  for (k in 1:K) {
    control_opti <- rpart.control(minbucket = 5, cp = 0.017)
    out_opti <- rpart(cost ~ ., data = dat_scaled, method = "anova", control = control_opti)
    yhat[cv_idx[[k]], model_num] <- as.numeric(predict(out_opti, dat_scaled[cv_idx[[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)
```

```r
MSEAve
r2 <- 1 - MSEAve/var(y); r2
pred <- predict(out_opti, dat_scaled)
res <- dat_scaled$cost - pred
plot(pred, res, main = "Fitted vs Residuals")
########### QUESTION 4 begins here ###################
dat <- fgl
set.seed(123)
Nrep <- 3 #number of replicates of CV
K <- 10  #K-fold CV on each replicate
n.models <- 8 #number of different models to fit
size <- c(5, 10)
decay <- c(0.001, 0.01, 0.1, 0)
n <- nrow(dat)
y <- dat$type
yhat <- matrix(0, n, n.models)
missclass <- matrix(0, Nrep, n.models)
output <- data.frame()
model_num <- 1
for (s in size) {
  for (d in decay) {
    for (j in 1:Nrep) {
      cv_idx <- createFolds(dat$type, k = K)
      for (k in 1:K) {
        out <-
          nnet(
            type ~ .,
            data = dat[-cv_idx[[k]],],
            linout = F,
            skip = F,
            size = s,
            decay = d,
            maxit = 1000,
            trace = F
          )
        yhat[cv_idx[[k]], model_num] <- predict(out, dat[cv_idx[[k]], ], type="class")
      } #end of k loop
      missclass[j, ] <- apply(yhat, 2, function(x) sum(y != x) / length(y))
    }#end of j loop
    output <- rbind(output, c(s, d))
    model_num <- model_num + 1
  }
}
missclass_avg <- apply(missclass,2,mean)
output <- cbind(output, missclass_avg)
names(output) <- c("size", "decay", "Average CV Missclassification rate over 3 replicates")
output
set.seed(123)
dat <- fgl
control <- rpart.control(minbucket = 5, maxdepth = 5)
out <- rpart(type ~ ., data = dat, method = "class", control = control)
plotcp(out)
printcp(out)
```

```
#prune back to optimal size, according to plot of CV r^2
out <- prune(out, cp=0.022)  #approximately the cp corresponding to the best size
par(cex=.7); plot(out, uniform=F); text(out, use.n = F); par(cex=1)
out$variable.importance
out$cptable[nrow(out$cptable),]
out$cptable[nrow(out$cptable),][c(3,4)]*min(table(dat$type)/nrow(dat))  #training and cv misclass rates
yhat<-predict(out, type="class"); sum(yhat != dat$type)/nrow(dat) #check training misclass rate
m2 <- multinom(type ~., data= dat, trace=FALSE)
yhat<-predict(m2, type="class")
sum(yhat != dat$type)/nrow(dat)
dat <- fgl
set.seed(123)
Nrep <- 3 #number of replicates of CV
K <- 10 #K-fold CV on each replicate
n.models <-3 #number of different models to fit
size <- c(5, 10)
decay <- c(0.001, 0.01, 0.1, 0)
n <- nrow(dat)
y <- as.numeric(as.factor(dat$type))
yhat <- matrix(0, n, n.models)
missclass <- matrix(0, Nrep, n.models)
output <- data.frame()

for (j in 1:Nrep) {
  cv_idx <- createFolds(dat$type, k = K)
  for (k in 1:K) {
    out <-
      nnet(
        type ~ .,
        data = dat[-cv_idx[[k]], ],
        linout = F,
        skip = F,
        size = 10,
        decay = 0.01,
        maxit = 1000,
        trace = F,
        method = "class"
      )
    yhat[cv_idx[[k]], 1] <- as.numeric(factor(predict(out, dat[cv_idx[[k]], ], type="class"), levels = 

    control <- rpart.control(minbucket = 5, cp = 0.022)
    out <- rpart(type ~ ., data = dat[-cv_idx[[k]], ], method = "class", control = control)
    yhat[cv_idx[[k]], 2] <- predict(out, dat[cv_idx[[k]], ], type="class")

    out <- multinom(type ~., data = dat[-cv_idx[[k]], ], trace=FALSE)
    yhat[cv_idx[[k]], 3] <- predict(out, dat[cv_idx[[k]], ], type="class")
  } #end of k loop
  missclass[j, ] <- apply(yhat, 2, function(x) sum(y != x) / length(y))
}#end of j loop

missclass_avg <- apply(missclass, 2, mean)
missclass_avg
```