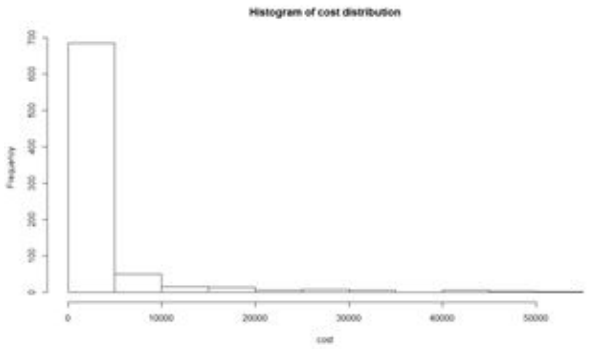
Predictive Analytics Homework 2

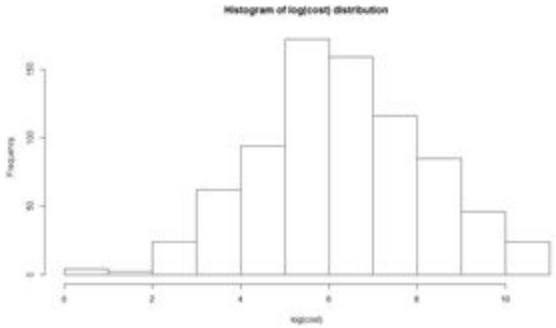
Winter 2015

Team Members: Shiyi Chen, Steven Lin, Shawn Li, Ameer Khan, Sanjeevni Wanchoo

Problem 1

a) We began by performing a log transformation on the response variable, to remedy its skewed distribution. The response variable (cost) distribution before and after the transformation is shown below.





We first fit a basic model, without any interactions, and performed stepwise regression, which gave us the model below:

```
Call:
lm(formula = log_cost ~ intvn + dur + comorb + comp + ervis +
    age + gend, data = heart_data)

Residuals:
    Min      1Q  Median      3Q     Max
-5.9119 -0.6833  0.0222  0.6740  3.9272

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.1356102  0.3910499  13.133  < 2e-16 ***
intvn        0.2017159  0.0087644  23.015  < 2e-16 ***
dur          0.0027940  0.0004315   6.475 1.67e-10 ***
comorb       0.0536119  0.0085787   6.249 6.77e-10 ***
comp         0.7374629  0.1825272   4.040 5.87e-05 ***
ervis        0.0395799  0.0182899   2.164   0.0308 *
age         -0.0101994  0.0066372  -1.537   0.1248
gend        -0.1491000  0.1058738  -1.408   0.1594
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.238 on 780 degrees of freedom
Multiple R-squared:  0.5824,   Adjusted R-squared:  0.5786
F-statistic: 155.4 on 7 and 780 DF,  p-value: < 2.2e-16
```

As can be seen above, # of drugs was seen as insignificant predictor, and was removed from the model during stepwise regression. Furthermore, p-values for age and gender were insignificant.
We then followed the same procedure to fit a model, but this time including the interaction terms, which (after stepwise regression), gave us the model below:

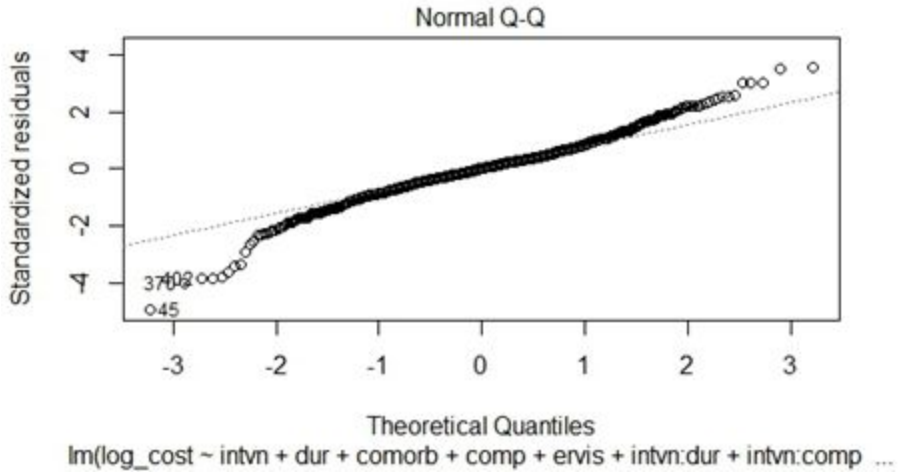
```
Call:
lm(formula = log_cost ~ intvn + dur + comorb + comp + ervis +
    drugs + gend + age + intvn:dur + intvn:comp + dur:comorb +
    intvn:comorb + intvn:ervis + dur:ervis + intvn:drugs + dur:gend,
    data = heart_data)

Residuals:
     Min       1Q   Median       3Q      Max
-4.6099 -0.5980  0.0301  0.6024  3.9810

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.431e+00  3.697e-01  11.984  < 2e-16 ***
intvn        3.744e-01  2.009e-02  18.633  < 2e-16 ***
dur          2.933e-03  6.754e-04   4.343 1.59e-05 ***
comorb       1.923e-01  2.986e-02   6.439 2.11e-10 ***
comp         1.427e+00  2.902e-01   4.916 1.08e-06 ***
ervis        9.314e-02  3.224e-02   2.889 0.003976 **
drugs       -3.081e-02  6.126e-02  -0.503 0.615186
gend        -3.593e-01  1.677e-01  -2.142 0.032513 *
age         -8.848e-03  6.130e-03  -1.443 0.149290
intvn:dur   -3.182e-04  8.815e-05  -3.609 0.000327 ***
intvn:comp  -7.634e-02  2.476e-02  -3.083 0.002123 **
dur:comorb  -3.622e-04  9.333e-05  -3.881 0.000113 ***
intvn:comorb -5.999e-03  1.458e-03  -4.115 4.30e-05 ***
intvn:ervis  -1.160e-02  2.461e-03  -4.713 2.89e-06 ***
dur:ervis    2.620e-04  1.470e-04   1.782 0.075107 .
intvn:drugs  -7.968e-03  4.859e-03  -1.640 0.101431
dur:gend     1.272e-03  8.140e-04   1.563 0.118489
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 1.137 on 771 degrees of freedom
Multiple R-squared:  0.6517,   Adjusted R-squared:  0.6445
F-statistic: 90.16 on 16 and 771 DF,  p-value: < 2.2e-16
```

Using interactions improved the R-squared. The adjusted R-squared went from .5786 to .6445. Here, age and drugs are not significant. Additionally, only a single interaction with drugs (intervn:drugs) was included in the model, and was also not significant (at α = 0.05). Therefore, to simplify the model further, drugs and age predictors were removed during the modeling process. All other predictors and their interactions were retained, and stepwise regression was performed. This gave us the final model below:

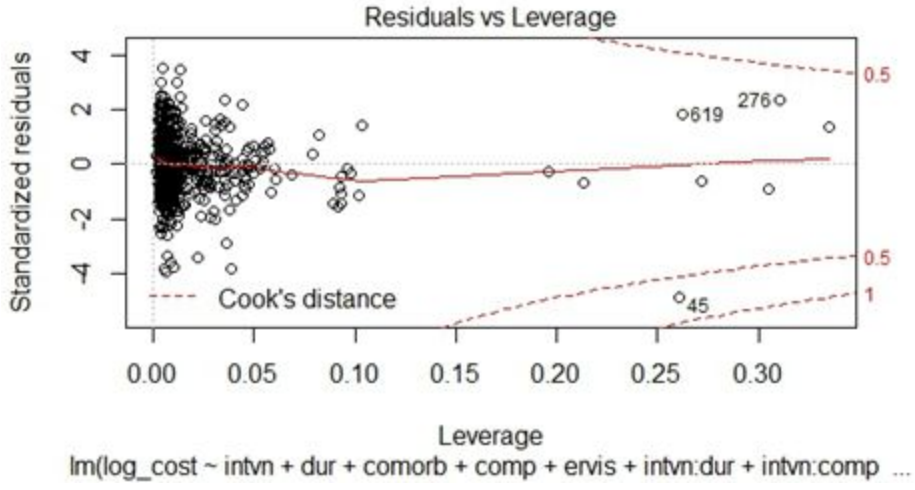
```
Call:
lm(formula = log_cost ~ intvn + dur + comorb + comp + ervis +
    intvn:dur + intvn:comp + dur:comorb + intvn:comorb + intvn:ervis +
    dur:ervis, data = heart_data)

Residuals:
     Min       1Q   Median       3Q      Max
-4.8189 -0.5882  0.0365  0.5931  4.0697

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.889e+00  1.239e-01  31.380  < 2e-16 ***
intvn        3.692e-01  1.999e-02  18.466  < 2e-16 ***
dur          2.935e-03  6.554e-04   4.478 8.67e-06 ***
comorb       2.004e-01  2.992e-02   6.696 4.10e-11 ***
comp         1.429e+00  2.909e-01   4.912 1.10e-06 ***
ervis        7.988e-02  3.089e-02   2.586 0.009888 **
intvn:dur   -2.705e-04  8.715e-05  -3.104 0.001977 **
intvn:comp  -7.816e-02  2.481e-02  -3.150 0.001697 **
dur:comorb  -3.939e-04  9.340e-05  -4.218 2.76e-05 ***
intvn:comorb -5.600e-03  1.446e-03  -3.873 0.000117 ***
intvn:ervis  -1.416e-02  1.952e-03  -7.253 9.89e-13 ***
dur:ervis    2.949e-04  1.433e-04   2.058 0.039943 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.143 on 776 degrees of freedom
Multiple R-squared:  0.6454,   Adjusted R-squared:  0.6404
F-statistic: 128.4 on 11 and 776 DF,  p-value: < 2.2e-16
```

The adjusted R-squared of this model is close to the previous model, and all the predictors above have significant p-values.
In order to ascertain the predictive power of this model, cross-validation was performed by using this model on the heart data. This gave us the cross-validation mean **R-squared of 0.6247**. This means that about 62.47% of the variance in data is explained by our model. The mean cross-validation SSE was 1073.495.
b) Number of interventions (intvn) , duration (dur) , comorbidities (comorb), complications (comp) and ER visits (ervis) appear to have the most influence on the cost. Age, number of drugs, and gender don't appear as important.
c) We start by checking the normality assumption. To check the assumption, we can look at the Q-Q Plot, as shown below:

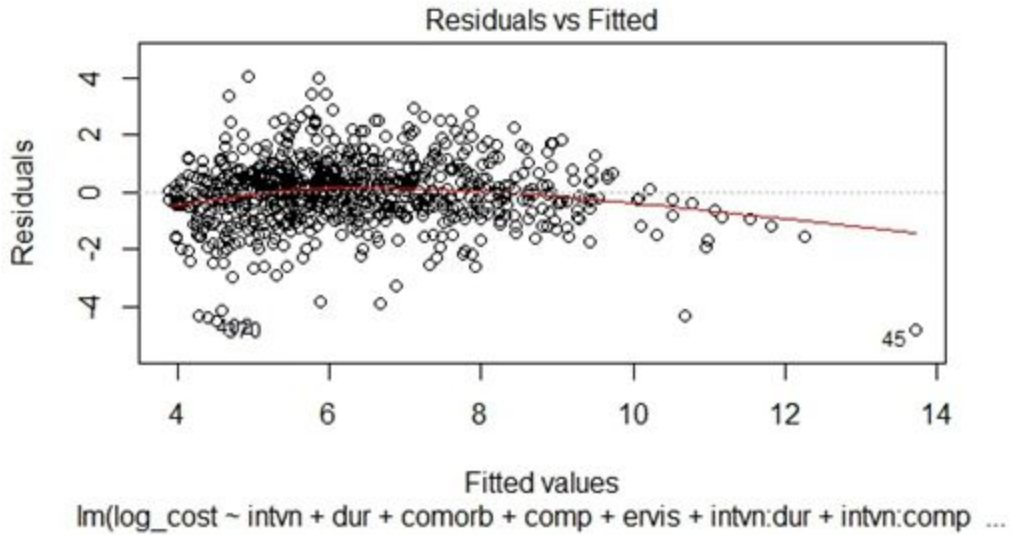


The QQ plot suggests that the data are close to normally distributed.

The plot of outliers and influential observations is shown below.



Next, we can look at the residual plots to assess the fit.



As can be seen, the residuals are mostly random distributed. The slightly non-linear trend at the end may be due to the impact of outliers and influential observations. The model can be improved on by removing these observations and refitting the model.

Problem 2

(a) Use 10-fold cross-validation to find the best combination of shrinkage parameter λ and number of hidden nodes.

Before fitting any models, the predictors and response were standardized. A few neural network models were run to narrow down the set of optimal shrinkage parameters and number of nodes. Then, a 10-fold cross validation for number of hidden nodes 1 to 6 and shrinkage parameter 0.05 to 0.10 in increments of 0.01 was conducted with 20 repetitions for each combination and averaging the R2 CV across the repetitions. The results show that the best combination resulting in the highest average R2 CV is: number of hidden nodes = 2 , shrinkage parameter λ = 0.08.

size	lambda	R2_avg
1	0.05	0.673005
1	0.06	0.672578
1	0.07	0.671923

1	0.08	0.67115
1	0.09	0.671149
1	0.1	0.670552
2	0.05	0.683855
2	0.06	0.682234
2	0.07	0.682801
2	**0.08**	**0.684414**
2	0.09	0.683648
2	0.10	0.684385
3	0.05	0.675818
3	0.06	0.678661
3	0.07	0.675509
3	0.08	0.677269
3	0.09	0.676607
3	0.10	0.676318
4	0.05	0.669463
4	0.06	0.669929
4	0.07	0.668089
4	0.08	0.671594
4	0.09	0.668953
4	0.10	0.668993
5	0.05	0.659384
5	0.06	0.658592
5	0.07	0.657605
5	0.08	0.660263
5	0.09	0.660595
5	0.10	0.662411
6	0.05	0.64854
6	0.06	0.64723
6	0.07	0.646848

6	0.08	0.648553
6	0.09	0.64695
6	0.1	0.648505

(b) Fit the final best model and discuss how good you think the model is, in terms of its predictive power.

Based on the 10-fold cross validation results, the best neural network model with number of hidden nodes = 2 , shrinkage parameter λ = 0.08 was fitted. The R output is shown below.

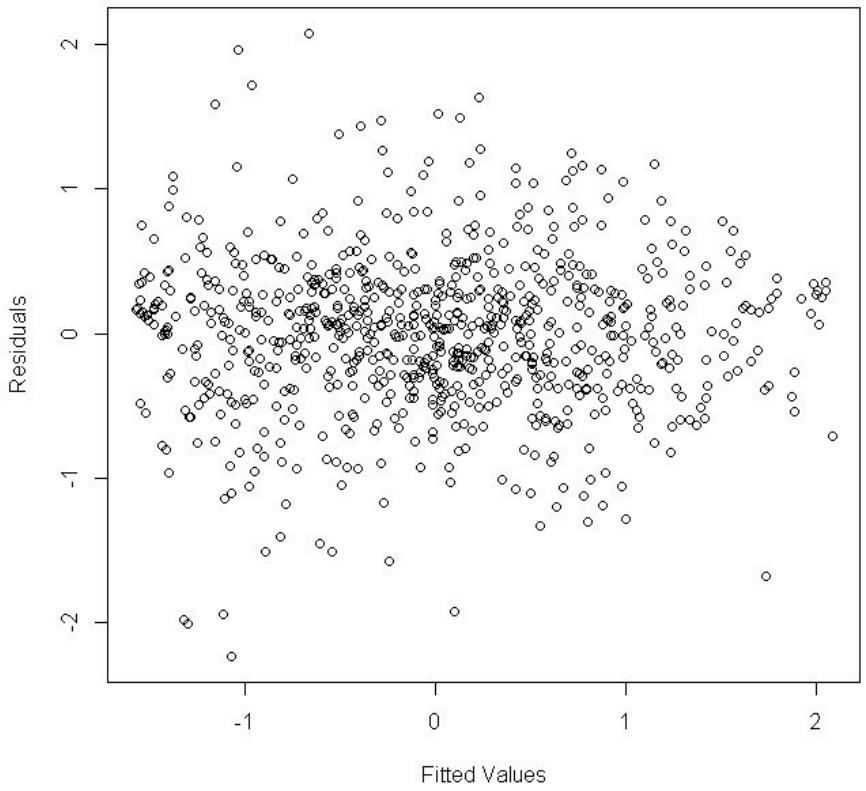
```
a 8-2-1 network with 21 weights
options were - linear output units  decay=0.08
 b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1
  0.23   0.01   0.03  -0.68  -0.01  -0.17  -0.12  -0.13   0.14
 b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2 i8->h2
  2.99  -0.12  -0.03   2.47  -0.27   0.08   0.17   1.68   0.48
 b->o h1->o h2->o
-0.14 -2.89  2.24
```

(c) Which variables appear to have the most influence on the cost?

Neural network models are not good for explanatory purposes since results are difficult to interpret. However, by looking at the output (columns of predictors and magnitude of coefficients), it seems that predictors number 3 (interventions) and 7 (comorbidities) seem to have the most influence on the cost.

(d) Construct appropriate residual plots to assess whether there remains any nonlinearity not captured by the neural network model.

The residuals plot looks random, indicating that there **does not remain** any nonlinearity that was not captured by the neural network model (i.e. all non-linearity was captured by the model).


Residuals
2
1
0
-1
-2
-1
0
1
2
Fitted Values

Problem 3

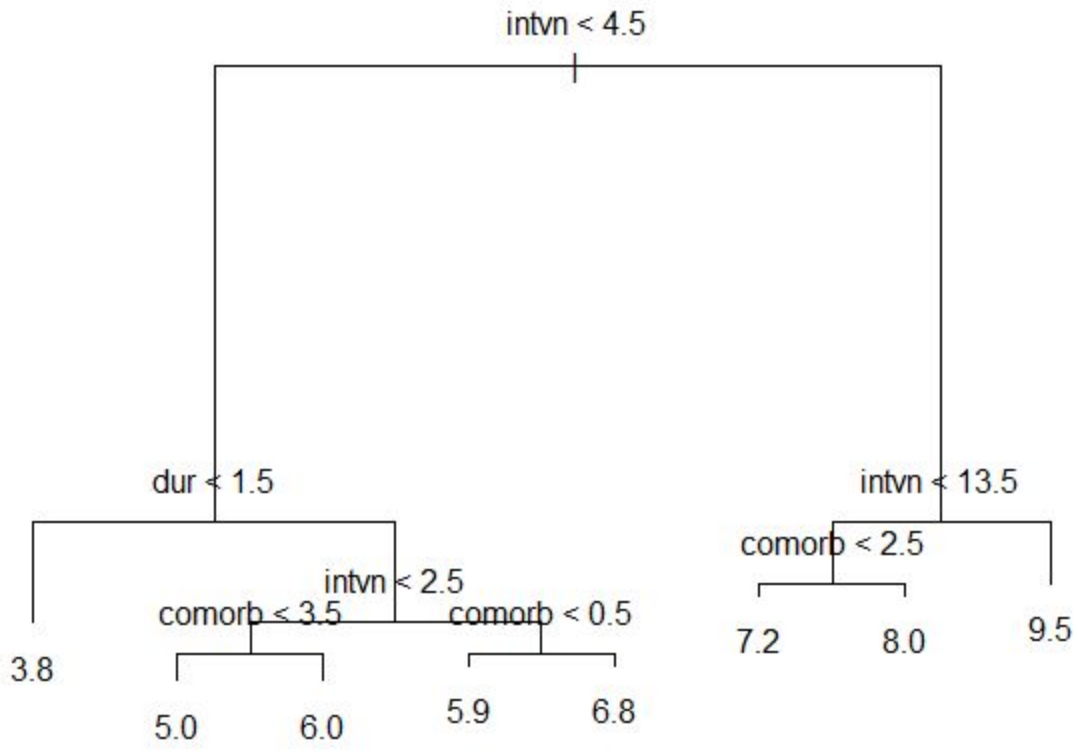
(a) Use 10-fold cross-validation to find the best tree size or complexity parameter value.

In order to determine the best tree size, a full model was first fit using the entire dataset. This tree model was then pruned using 10-fold cross-validation, and the lowest value of tree size (number of leaf nodes) corresponding to the minimum deviance were selected as the best size. 100 repetitions of the cross-validation procedure were performed, and the most frequent value of the smallest tree size that minimized the deviance was reported as the best tree size.

The best tree size obtained was 8.

(b) Fit the final best model and discuss how good you think the model is, in terms of its predictive power.

The best model was obtained by pruning the original tree model down to the best size - 8 leaf nodes. The tree model looks like:



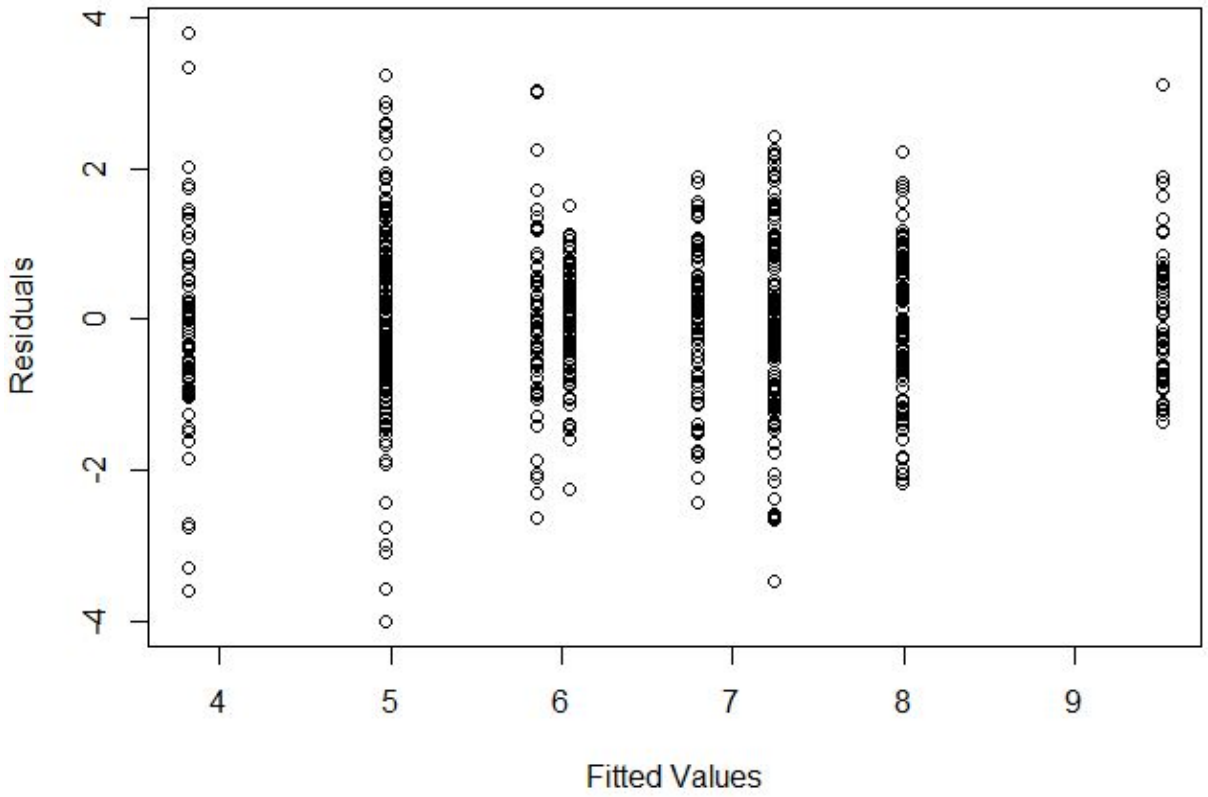
The predictive power of the tree is estimated using 10-fold cross-validation and computing the cross-validation SSE and the cross-validation R^2 (obtained using 1 - var(residuals) / var(y)). For 10 repetitions, the mean prediction R^2 obtained is 63.48%.

(c) Which variables appear to have the most influence on the cost?

The length of the branches in the above tree gives the relative importance of the corresponding predictor. From the tree representation, intvn, dur and comorb are the most important variables, in that order.

(d) Construct appropriate residual plots to assess whether there remains any linearity not captured by the regression tree model.

The plot of the residuals against the fitted values is given below:



Since the residuals are fairly randomly distributed, there isn't any lingering linearity not captured by the model. The fitted values increment only in steps as the tree assigns a fixed value to each leaf node.

(e) Which model (the linear regression, neural network, or tree) would you recommend for this data set, and why?

The best model is decided on the basis of its prediction power. The highest prediction R2 value using 10-fold cross-validation was obtained for the neural network model (~68%) as opposed to the regression model (~62%) and the decision tree (~63%). The same cross-validation partitions were used for computing the cross-validation R2 for the three models. In terms of pure prediction power, the neural network is the best. If explanatory power is to be considered as

well, then the decision tree is the easiest to interpret, with only a marginal decrease in prediction performance.

Problem 4

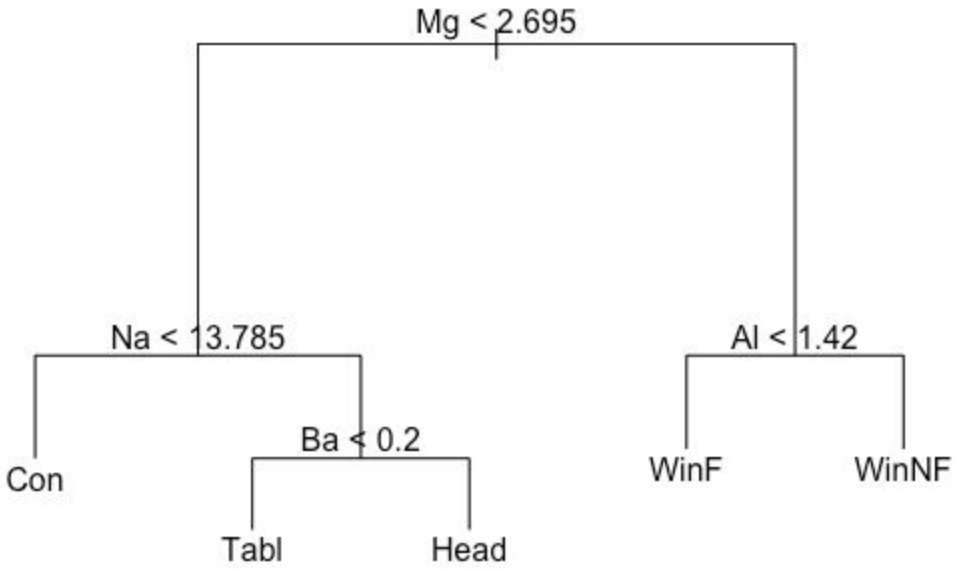
(a) Use 10-fold cross-validation to find the best neural network model for classifying the class type.

We first standardized the predictor variables to fulfill the requirement of the neural network model. The response variable was not rescaled for it is categorical. Then we ran 10-fold cross validation for the number of hidden nodes 5 to 20 and λ 0 to 0.2 with 0.05 increment and calculated misclassification rate. The results below show that the combination of 19 hidden nodes and λ = 0.1 gives the lowest misclassification rate: **0.247**.

	lambda 0	**lambda 0.05**	**lambda 0.1**	**lambda 0.15**	**lambda 0.2**
size 5	0.3131	0.3224	0.3411	0.3411	0.3505
size 6	0.3551	0.2991	0.2991	0.3084	0.3458
size 7	0.3645	0.3271	0.3131	0.3037	0.3271
size 8	0.3318	0.2850	0.3224	0.3037	0.3084
size 9	0.3551	0.2944	0.2850	0.3271	0.3131
size 10	0.3598	0.3645	0.2897	0.3084	0.3037
size 11	0.3598	0.2804	0.2804	0.3178	0.3131
size 12	0.3832	0.2570	0.2710	0.2850	0.2804
size 13	0.3785	0.2710	0.2897	0.3224	0.3037
size 14	0.3411	0.3037	0.2710	0.2944	0.3318
size 15	0.3879	0.2850	0.2944	0.2944	0.3364
size 16	0.3411	0.2991	0.3037	0.3037	0.3037
size 17	0.3738	0.2710	0.2617	0.2897	0.3084
size 18	0.3972	0.2897	0.2897	0.2991	0.3037
size 19	0.3832	0.2617	**0.2477**	0.3318	0.2991
size 20	0.3645	0.2757	0.3037	0.2804	0.3271

(b) Use 10-fold cross-validation to find the best classification tree model for classifying the class type.

For a tree model we do not need to do any variable transformation. Therefore we first ran 10-fold cross-validation with cv.tree function 100 times to determine the best size of a tree. 10-fold cross-validation suggests us that 5 is the optimal size. Then we calculated misclassification rate for the tree with size 5 by running 10-fold cross-validation 100 times and averaging misclassification rate across 100 repetitions. The result shows that the best classification tree model has misclassification rate **0.389.** The plot of the best classification tree is displayed below.



(c) Fit a multinomial model and discuss the results.

We first encoded the response variable “type” as a factor and then we fitted a multinomial model. Following is the result of the fitting model.

We used cross validation to choose the best multinomial model by evaluating a misclassification rate. Since different partitions give different results, we fitted the model for 10 times and got a 10 different misclassification rates. We get the mean value of the 10 results to be the final classification rate of the multinomial model, which is **0.3864**.

```
Call:
multinom(formula = type ~ ., data = data, maxit = 1000)

Coefficients:
      (Intercept)         RI          Na         Mg         Al         Si          K          Ca          Ba          Fe
Head   -117.16321 15.7761898 20.40013311 -44.636113 -15.55451  7.623556 -26.2279698 -65.4073909  -25.067857 -334.732984
Tabl    -26.70279 21.4675715 33.52535913 -50.794834  17.89681  7.156916 -272.4164964 -89.0846221 -246.976201 -683.299664
Veh     175.66039 -1.8440801  1.89927845   7.824761 -16.80545 -3.312461    0.8600173   5.0995220    3.664816    9.821937
WinF   -315.69345 -0.3658734  4.76416894  10.210424 -14.51400  2.692985    7.0681291   5.6186720    8.024848   10.924882
WinNF   225.31141 -0.1365759  0.05875039   3.021057 -15.53320 -2.814580    2.0922886  -0.1644049    1.301319   12.882593

Std. Errors:
      (Intercept)        RI        Na       Mg       Al        Si        K        Ca        Ba         Fe
Head   0.03001096 0.6639113 2.1329815 1.093490 3.740305 0.4006438 1.9649179 0.8064869 2.4813902 0.02672328
Tabl   0.05592167 4.6407082 2.1645212 5.296269 2.590322 0.1994511 0.0892803 4.4830515 0.4084623 0.03605334
Veh    0.07335335 0.8064206 1.1601923 1.922142 3.122923 0.4444306 3.0118868 2.0518127 4.5927995 3.12072679
WinF   0.05606164 0.7135256 1.0443353 1.530458 2.940196 0.3627298 2.6792460 1.6491620 2.9393278 2.39360870
WinNF  0.04976264 0.6703784 0.9224886 1.302801 2.710065 0.3223242 2.1224752 1.4685504 1.6159217 2.02625832

Residual Deviance: 249.5311
AIC: 349.5311
```

```
> misclass_reg
 [1] 0.3738318 0.4065421 0.3878505 0.3785047 0.3785047 0.3878505 0.3785047 0.3971963 0.3878505 0.3878505

> mean( misclass_reg );
[1] 0.3864486
```

(d) Compare the three models from parts (a)—(c) in terms of their predictive ability and interpretability. Which model do you think is the most appropriate for predicting glass type?

Based on results from previous 3 questions, we can get the following misclassification rate table.

	Neural Network	Classification Tree	Multinomial model
Misclassification rate	0.247	0.389	0.386

Following table is the pros and cons based on predictive ability and interpretability for these 3 models.

	Neural Network	Classification Tree	Multinomial model
Predictive ability	Neural network has the **lowest misclassification rate**. It shows Neural network model has the strongest predictive ability	Classification tree has a very similar misclassification rate with multinomial model. The predictive abilities of these two	Multinomial model has a very similar misclassification rate with classification tree. The predictive

	among these 3 models.	models are close.	abilities of these two models are close.
Interpretability	Difficult to interpret.	Classification tree is simple to understand and interpret. It's easy to understand what variables are important in making the prediction	Multinomial model is simple to understand and interpret. It has output telling which of the independent variables are significantly related to the response variable by the Likelihood ratio test.

Based on both predictive ability and interpretability, we think classification tree would be the most appropriate for predicting glass type amount these 3 models.

A**ppendix**

```
####Problem 1 ################################################
library( car );


# Function for performing cross-validation
cvIndex = function ( n, k ) {
  m = n %/% k;
  r = n %% k;

  index_vector = sample( n, n );
  index = list( );

  for ( i in 1:k ) {
    if ( i <= r ) {
      kpart = ( ( m + 1 ) * ( i - 1 ) + 1 ):( ( m + 1 ) * i );
    }
    else {
      kpart = ( ( m + 1 ) * r + m * ( i - r - 1 ) + 1 ):( ( m + 1 ) * r + m * ( i - r ) );
    }

    index[[ i ]] = index_vector[ kpart ];
  }

  return( index );
};

# Read data
heart_data = read.csv( "HW2_data.csv", header = T );

n = nrow( heart_data );
p = ncol( heart_data );

# Examine cost (response variable) distribution
hist( heart_data[ , "cost" ] , xlab = "cost" , main = "Histogram of cost distribution" );
hist( log( heart_data[ , "cost" ] ) , xlab = "log(cost)" , main = "Histogram of log(cost) distribution"
);

heart_data = heart_data[ , 2:p ];
p = p - 1;

# Perform log transformation on the cost variable
```

```
heart_data[ , "cost" ] = log( heart_data[ , "cost" ] );
colnames( heart_data )[ 1 ] = "log_cost";

# par( mfrow = c( 2, 1 ) )
plot( jitter( heart_data$intvn, 3 ), jitter( heart_data$log_cost, 3 ), pch = 16, cex = 0.5 );
plot( jitter( heart_data$comorb, 3 ), jitter( heart_data$log_cost, 3 ), pch = 16, cex = 0.5 );


fit_null = lm( log_cost ~ 1, data = heart_data );

fit_full = lm( log_cost ~ ., data = heart_data );
fit_step = step( fit_null, scope = formula( fit_full ) , direction = "both" );

fit_full = lm( log_cost ~ ( . ) ^ 2, data = heart_data );
fit_step = step( fit_null, scope = formula( fit_full ) , direction = "both" );

fit_full = lm( log_cost ~ ( . - age - drugs ) ^ 2, data = heart_data );
fit_step = step( fit_null, scope = formula( fit_full ) , direction = "both" );

summary( fit_step );

plot( fit_step );
avPlots( fit_step );


# Cross-Validation to obtain R-squared

k = 10;
N = 10;

SSE_lm = c( );
Rsq_lm = c( );

set.seed( 1 );

for ( j in 1:N ) {
   indexCV = cvIndex( n, k );
   yhat = rep( NA, n );

   for ( i in 1:k ) {
     fit_cv = lm( formula = formula( fit_step ), data = heart_data[ -indexCV[[ i ]], ] );
     yhat[ indexCV[[ i ]] ] = as.numeric( predict( fit_cv, heart_data[ indexCV[[ i ]], ] ) );
   }
```

```
    e = yhat - heart_data[ , "log_cost" ];

    SSE_lm = c( SSE_lm, sum( e ^ 2 ) );
    Rsq_lm = c( Rsq_lm, 1 - var( e ) / var( heart_data[ , "log_cost" ] ) );
}

mean( Rsq_lm );

#############################################################

#### Problem 2 ##############################################

#### Load data

# My PC
main = "C:/Users/Steven/Documents/Academics/3_Graduate School/2014-2015 ~ NU/"

# Aginity
#main = "\\\\nas1/labuser169"

course = "MSIA_420_Predictive_Analytics"
datafolder = "Data"
setwd(file.path(main,course, datafolder))

filename = "HW2_data.csv"
mydata = read.csv(filename,header = T)

#### Process data

head(mydata)
mydata = mydata[-1]
head(mydata) # drop ID column

# convert rsponse to log10 (change 0 to 1)
hist(mydata$total_cost)
mydata$total_cost[mydata$total_cost == 0] = 1
mydata$total_cost = log10(mydata$total_cost)
hist(mydata$total_cost)


# Use 10-fold cross-validation to find the best combination of # shrinkage parameter lambda
and number of hidden nodes
```

```
k = ncol(mydata)  #number of columns (response is in first column)
mydata1 = mydata  #will be standardized and scaled version of data
mydata1[1:k] = sapply(mydata1[1:k], function(x) (x-mean(x))/sd(x)) #standardize predictors and
need to standardize response


pairs(mydata1, cex=.5, pch=16)

#### A function to determine the indices in a CV partition
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
}

library(nnet)
##Now use the same CV partition to compare Neural Net
sdreps = matrix(0,0,3) #run this line once. Run the remaining lines multiple
colnames(sdreps) = c("size","lambda","R2_avg")
replicates
maxSize = 6
minSize = 1
lambdas = seq(0.05,0.1, by =0.01)
#lambdas = c(0.1)
maxReps = 20

# Start the clock!
ptm <- proc.time()

for (node in minSize:maxSize){
  for (lambda in lambdas){

          results = c()
```

```
        for (rep in 1:maxReps){
        Ind = CVInd(n=nrow(mydata1),10)
        K = length(Ind)
        y = mydata1$total_cost
        yhat = y

        for (k in 1:K) {
        out = nnet(total_cost~.,mydata1[-Ind[[k]],],
                linout=T, skip=F, size=node, decay=lambda, maxit=1000, trace=F)
        yhat[Ind[[k]]] = as.numeric(predict(out,mydata1[Ind[[k]],]))
        }

        e=y-yhat
        R2 = 1 - var(e)/var(y)
        results = c(results,R2)

        }
        R2_avg = mean(results)
        sdreps = rbind(sdreps,c(node,lambda,R2_avg))
 }

}

# Stop the clock
proc.time() - ptm

best = sdreps[which.max(sdreps[,"R2_avg"]),]
best # size =2, lambda = 0.08

write.table(sdreps,"Hw2_problem2.csv", sep=",")

# fit model

fitNNET= nnet(total_cost~.,mydata1,
        linout=T, skip=F, size=best["size"], decay=best["lambda"], maxit=1000, trace=F)

# output
summary(fitNNET)

# residual plots
y = mydata1$total_cost
yhat = y
```

```
yhat = as.numeric(predict(fitNNET,mydata1))
e=y-yhat
plot(e~yhat,xlab="Fitted Values",ylab="Residuals")

#### Problem 3 ################################################

library( tree );

heart_data = read.csv( "HW2_data.csv", header = T );
n = nrow( heart_data );
p = ncol( heart_data );

heart_data = heart_data[ , 2:p ];

p = p - 1;
heart_data[ , "cost" ] = log( heart_data[ , "cost" ] );
colnames( heart_data )[ 1 ] = "log_cost";

control = tree.control( nobs = n, mincut = 5, minsize = 10, mindev = 0.002 );

tree_fit = tree( log_cost ~ ., heart_data, control = control );
plot( tree_fit );
text( tree_fit, digits = 2 );

tree_prune = prune.tree( tree_fit );
plot( tree_prune );

N = 100;

minSize = c( );

for ( j in 1:N ) {
  tree_cv = cv.tree( tree_fit, FUN = prune.tree, K = 10 );
  minSize = c( minSize, min( tree_cv$size[ which( tree_cv$dev == min( tree_cv$dev ) ) ] ) );
}

# Obtain the most frequent value for the minimum size
bestSize = as.numeric( names( sort( table( minSize ), decreasing = T )[ 1 ] ) );

tree_best = prune.tree( tree_fit, best = bestSize );
plot( tree_best );
text( tree_best, digits = 2 );
```

```
N = 10;

SSE_tree = c( );
Rsq_tree = c( );

set.seed( 1 );

for ( j in 1:N ) {
  indexCV = cvIndex( n, k );
  yhat = rep( NA, n );

  for ( i in 1:k ) {
    tree_cv = tree( log_cost ~ ., heart_data[ -indexCV[[ i ]], ], control = control );
    tree_prune_cv = prune.tree( tree_cv, best = bestSize );
    yhat[ indexCV[[ i ]] ] = as.numeric( predict( tree_prune_cv, heart_data[ indexCV[[ i ]], ] ) );
  }

  e = yhat - heart_data[ , "log_cost" ];

  SSE_tree = c( SSE_tree, sum( e ^ 2 ) );
  Rsq_tree = c( Rsq_tree, 1 - var( e ) / var( heart_data[ , "log_cost" ] ) );
}

mean( Rsq_tree );

fitted = predict( tree_best, newdata = NULL );
e = fitted - heart_data[ , "log_cost" ];

plot( fitted, e, xlab = "Fitted Values", ylab = "Residuals" );
plot( heart_data[ , "intvn" ], e );


####Problem 4 ################################################
library( MASS );
library( nnet );
library( tree );


cvIndex = function ( n, k ) {
  m = n %/% k;
  r = n %% k;

  index_vector = sample( n, n );
```

```
  index = list( );

  for ( i in 1:k ) {
    if ( i <= r ) {
      kpart = ( ( m + 1 ) * ( i - 1 ) + 1 ):( ( m + 1 ) * i );
    }
    else {
      kpart = ( ( m + 1 ) * r + m * ( i - r - 1 ) + 1 ):( ( m + 1 ) * r + m * ( i - r ) );
    }

    index[[ i ]] = index_vector[ kpart ];
  }

  return( index );
};

p = ncol( fgl );
n = nrow( fgl );

# Neural Network
# Standardize predictors
data = as.data.frame( scale( fgl[ , 1: p - 1 ] ) );
data[ , "type" ] = as.factor( fgl[ , p ] );


hiddenNodes = seq( 5, 20 );
lambda = seq( 0, 0.2, by = 0.05 );

k = 10;

misclass_nn = matrix( , nrow = length( hiddenNodes ), ncol = length( lambda ) );

set.seed( 1 );

for ( indexNodes in 1:length( hiddenNodes ) ) {
  for ( indexLambda in 1:length( lambda ) ) {
    indexCV = cvIndex( n, k )
    yhat = rep( NA, n );
    # phat_max = rep( NA, n );

    for ( i in 1:k ) {
      fit_nn = nnet( type ~ ., data[ -indexCV[[ i ]], ], linout = F, skip = F,
```

```
                size = hiddenNodes[ indexNodes ], decay = lambda[ indexLambda ], maxit = 1000,
trace = F );

    yhat[ indexCV[[ i ]] ] = predict( fit_nn, data[ indexCV[[ i ]], ], type = "class" );

    # phat = predict( fit_nn, data[ indexCV[[ i ]], ], type = "raw" );
    # phat_max[ indexCV[[ i ]] ] = apply( phat, 1, max );
   }

   misclass_nn[ indexNodes, indexLambda ] = sum( yhat != data[ , "type" ] ) / n;
  }
}

indexMinMCE = which.min( misclass_nn );
nodesMin = 1 + ( indexMinMCE - 1 ) %% ( length( hiddenNodes ) );
lambdaMin = 1 + ( indexMinMCE - 1 ) %/% ( length( hiddenNodes ) );

hiddenNodes[ nodesMin ];
lambda[ lambdaMin ];

misclass_nn[ nodesMin, lambdaMin ];

write(misclass_nn, "misclass.csv")

# Tree
data = fgl;
data[ , "type" ] = as.factor( data[ , "type" ] );

control = tree.control( nobs = n, mincut = 5, minsize = 10, mindev = 0.0001 );

tree_fit = tree( type ~ ., data, control = control );
plot( tree_fit );
text( tree_fit, digits = 2 );

tree_prune = prune.tree( tree_fit );
plot( tree_prune )

N = 100;

minSize = c( );

for ( j in 1:N ) {
  tree_cv = cv.tree( tree_fit, FUN = prune.tree, K = 10 );
```

```
  minSize = c( minSize, min( tree_cv$size[ which( tree_cv$dev == min( tree_cv$dev ) ) ] ) );
}

bestSize = as.numeric( names( table( minSize ) ) [ which( table( minSize ) == max( table(
minSize ) ) ) ] );

tree_best = prune.tree( tree_fit, best = bestSize );
plot( tree_best )
text( tree_best, digits = 2 )

k = 10;
N = 10;

misclass_tree = c( );

set.seed( 1 );

for ( j in 1:N ) {
  indexCV = cvIndex( n, k )
  yhat = rep( NA, n );
  yhat = as.factor( yhat );
  levels( yhat ) = levels( data[ , "type" ] );

  for ( i in 1:k ) {
    tree_cv = tree( type ~ ., data[ -indexCV[[ i ]], ], control = control );
    tree_prune_cv = prune.tree( tree_cv, best = bestSize );
    yhat[ indexCV[[ i ]] ] = predict( tree_prune_cv, data[ indexCV[[ i ]], ], type = "class" );
  }

  misclass_tree = c( misclass_tree, sum( yhat != data[ , "type" ] ) / n );
}

mean( misclass_tree );

# Multinomial Logistic Regression

data = fgl;
data[ , "type" ] = as.factor( data[ , "type" ] );

fit_null = multinom( type ~ 1, data );
fit_full = multinom( type ~ ., data, maxit = 1000  );

misclass_reg = c( );
```

```
set.seed( 1 );

for ( j in 1:N ) {
  indexCV = cvIndex( n, k )
  yhat = rep( NA, n );
  yhat = as.factor( yhat );
  levels( yhat ) = levels( data[ , "type" ] );

  for ( i in 1:k ) {
    fit_cv = multinom( formula = formula( fit_full ), data = data[ -indexCV[[ i ]], ], maxit = 1000,
trace = F );
    yhat[ indexCV[[ i ]] ] = predict( fit_cv, data[ indexCV[[ i ]], ], type = "class" );
  }

  misclass_reg = c( misclass_reg, sum( yhat != data[ , "type" ] ) / n );
}

mean( misclass_reg );
```