Predictive Analytics: HW # 2

Ayush Agarwal

Question 1

```
Call:
lm(formula = log10(cost) ~ ., data = df1)

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
```

a) The adjusted r2 = 57.89 % which means 58 % of the variability in the log(cost) is explained by the linear dependence on the predictors. We can say that response is somewhat predictable using linear model, but non-linear models can be explored in this case for more better fit on the data.

b) Let’s first check the VIF values of the predictors to check for multi-collinearity before analyzing the p-values of the predictors.

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

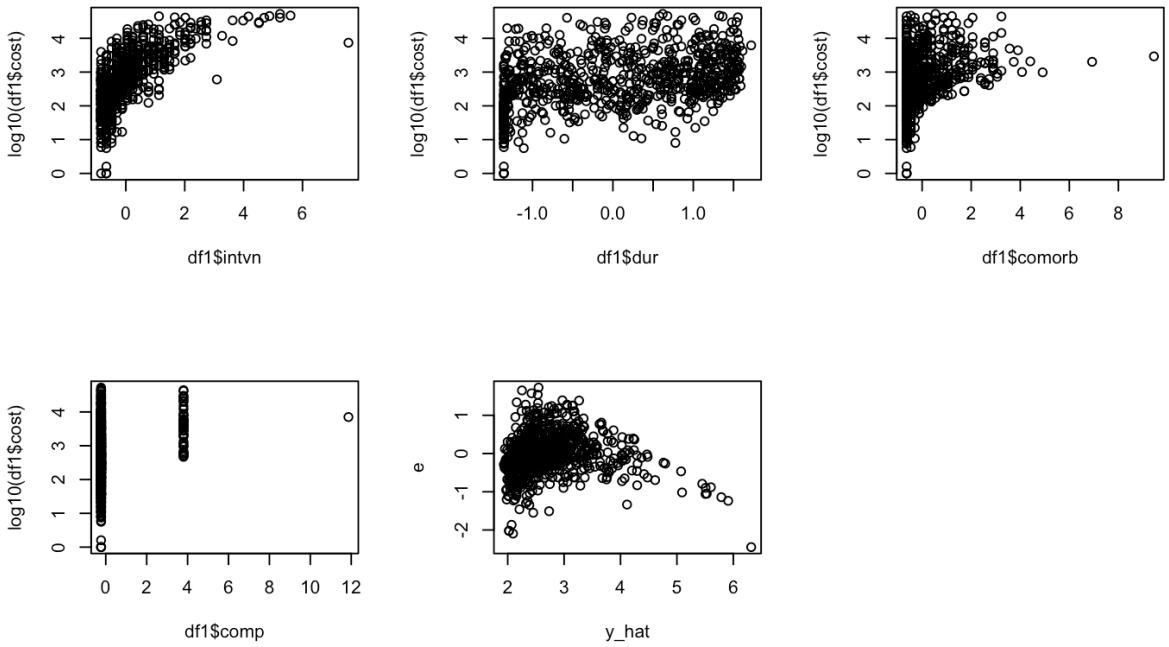
None of the VIF values are large therefore no sign of multi-collinearity is there. By analyzing the P-values of the predictors, intvn, comp, comorb and dur are having p-values less than 5%, therefore they are statistically significant.

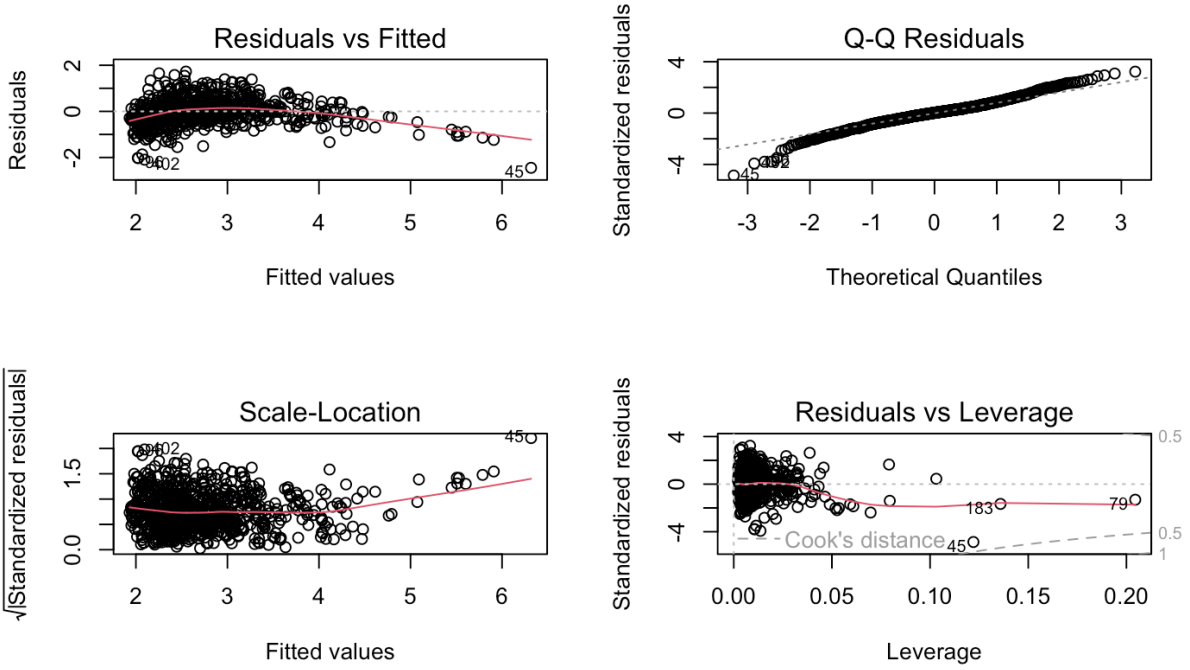
Since the predictors are standardized, the coefficients are on same scale, therefore we can compare their values directly by using their magnitudes.

Order of Importance

intvn > dur > comorb > comp

c) Let's analyze and conclude from the plots below.





- I have plotted the pair plots between significant variables and log10(cost). It can be clearly seen that non-linearity is the issue here. No predictor is linearly related with the response variable.

- Also, the errors are heteroscedastic and follow non-linear patterns. Therefore, there is a need to evaluate the non-linear models in this case.
- The Q-Q plots also have some tails which shows the errors are also not normally distributed as well.

Question 2:

a) I have done 10-fold cross validation with 5 replicates and 12 models. Here is the result summary.

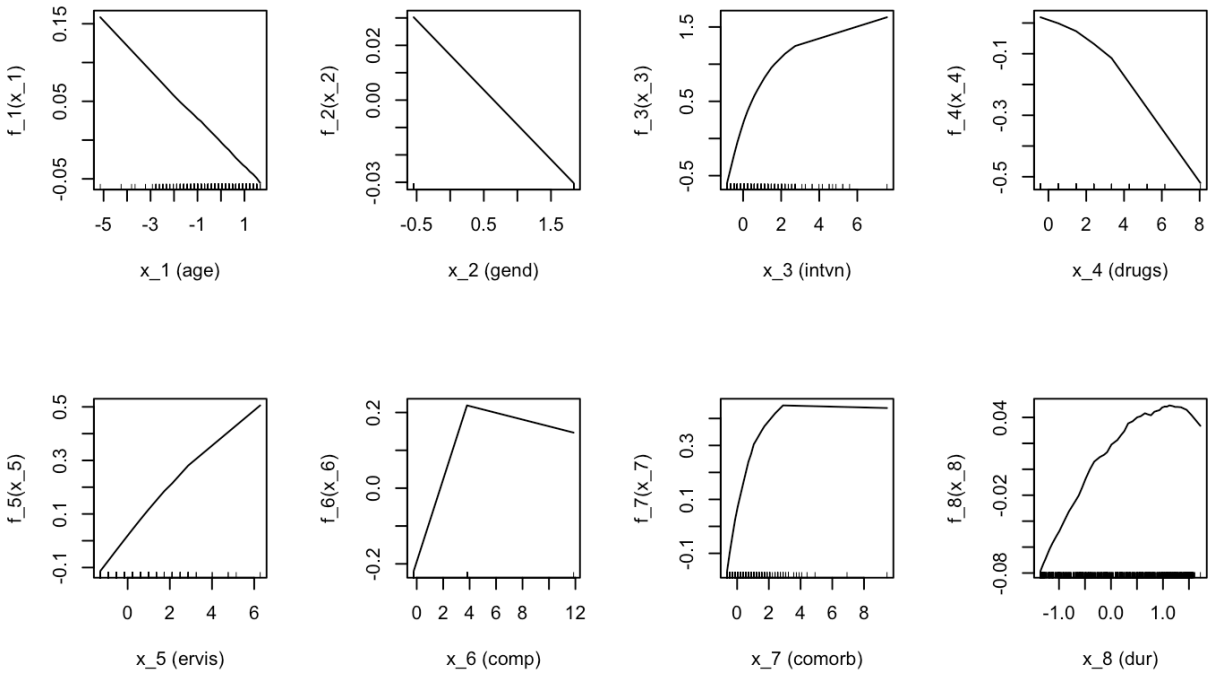
Skip	Size	Decay	Avg CV r2
F	10	1	0.6836031
F	10	2	0.6784083
F	10	0.1	0.6210518
F	20	1	0.6818906
F	20	5	0.6583066
F	5	1	0.6822078
F	5	0.5	0.6786005
T	10	0.1	0.5772591
T	10	1	0.6797126
T	10	10	0.6163756
T	20	1	0.6788410
T	20	0.01	0.2952480

The best model is found with hyper parameters Skip = F, Size = 10 and Decay = 1 with Average CV MSE = 0.6836031

b) Fitting the best Neural Net and Linear Regression with 10-fold cross validation and 20 replicates
    - For Neural Net, Average CV r2 = 0.6825088
    - For Linear Regression, Average CV r2 = 0.5836692

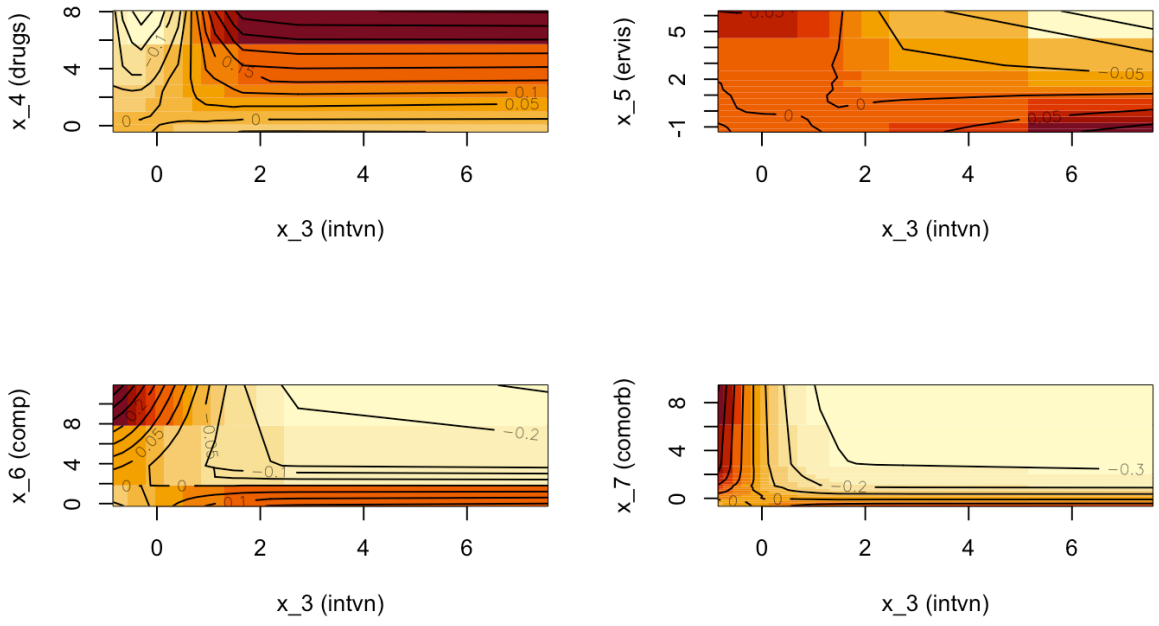
    Clearly, it can be concluded that Neural Network has higher CV R2 than Linear Regression, which means it does a better job in explaining the variability in response variable using the non-linear relationships among the predictors.

c) Summary of a neural net is hard to interpret, ALE plots can help in determining the importance.



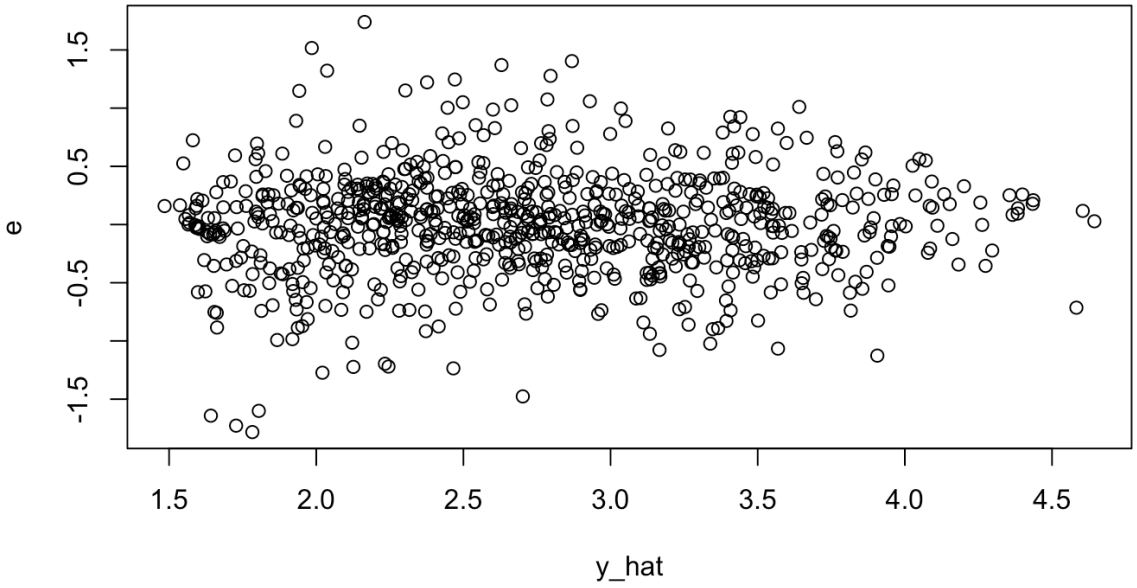
From the ALE plots it can be observed that “intvn” is still of highest importance followed by drugs, ervis, comorb and comp which are the next most important predictors having nearly same importance value.

Let’s analyze the interaction plots of the intvn with drugs, ervis, comorb and comp to see if there are interaction effects in addition to main effects



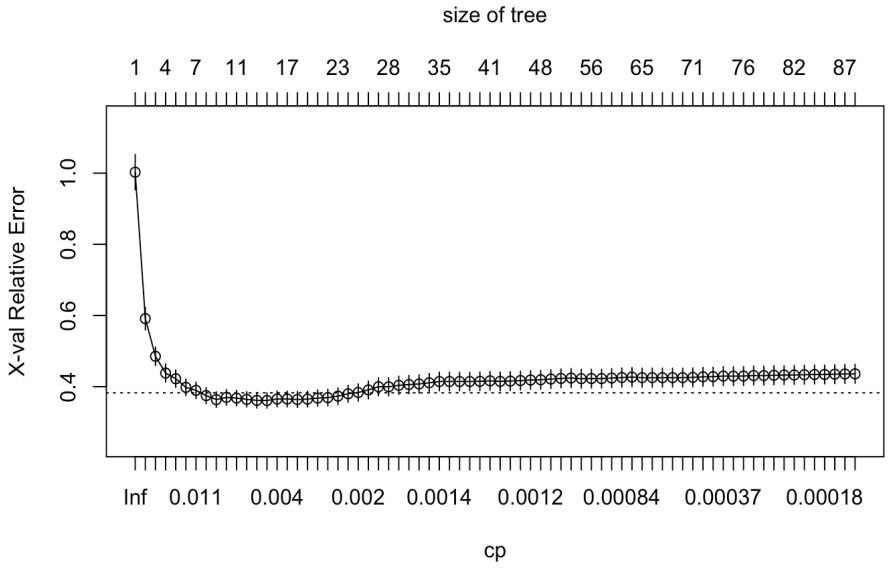
From the above plots, we can clearly see that, intvn is having sizeable interactions with drugs, ervis, comp and comorb as compared to main effects, therefore the interactions should also be considered.

d) Let's plot the residuals vs predicted values. The errors are looking completely scattered now with no non-linear patterns observed. Therefore it can be concluded that neural net is able to detect all non-linearity in the data .



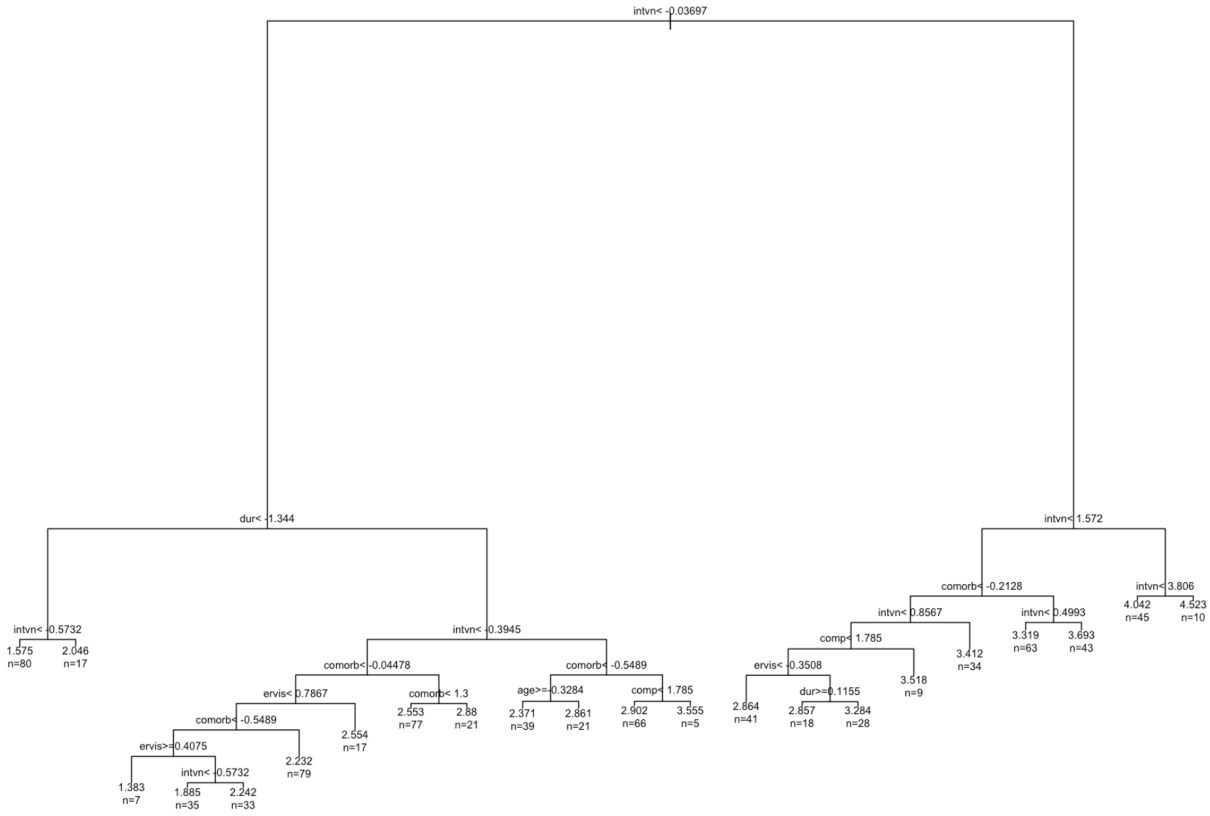
Question 3

a) Below is the plot Validation R2 Error vs CP Values



From the above plot and reading the CP Table, we can conclude that optimum decay value is **0.003140835** with a size of **19 nodes**.

b) Below is the plot of Best Fitted Tree



Training R2 = 0.734 [*This may be inflated due to overfitting, better to Check CV r2*]
Cross Validation R2 = 0.657
For Neural Networks we found that it was Cross Validation R2 was 68.2 % which is better than 65.7 % . Hence we can conclude that Decision Tree did a fairly good job at in predicting cost but neural network is little better here.

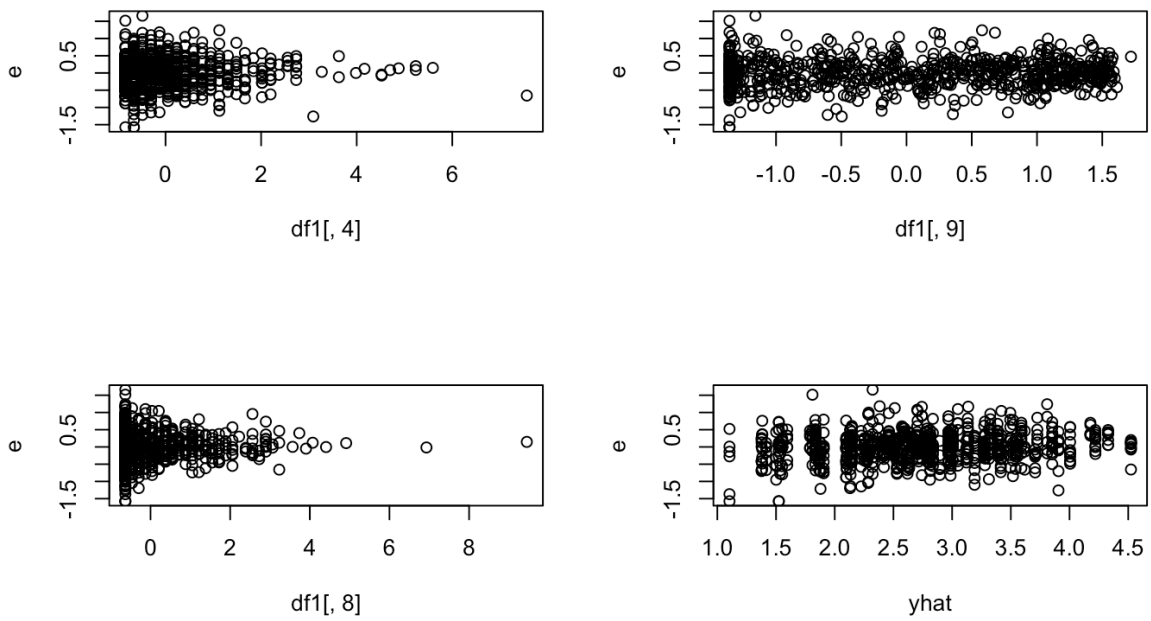
c) Intvn , dur , comorb, ervis , comp and age are the predictors used in constructing decision tree

intvn	dur	comorb	ervis	comp	age
299.146502	53.693377	28.910902	7.213502	4.190102	3.278480

From the feature importance table, it can be clearly seen that intvn is most important followed by dur, comorb while ervis , comp and age are not so important

Intvn >>> dur > comorb > ervis > comp > age

d) Below are the residuals plot b/w residual and top 3 important variables and y_hat



From the above plots, there are no non-linear patterns observed in the data. Hence we can conclude that there are no non-linear patterns that are not captured by the fitted decision tree.

e) Lets compare the cross Validation R2 for Decision Tree, Neural Network and Linear Regression

Decision Tree = 65.7 %
Neural Network = 68.2 %
Linear Regression = 58.36 %

By comparing the R2 values, it can be clearly interpreted that Neural Network seems to work best with highest r2 value of 68.2 %. Decision tree also seem to work well with r2 of 65.7 % which is not very less than neural net while linear regression performs worst. Hence in terms of suggestion,

- **In terms of Predictive Power**
  Neural Net > Decision Tree > Linear Regression (Not Recommended)
- In terms of explainability
  Linear Regression > Decision Tree > Neural Net

Question 4

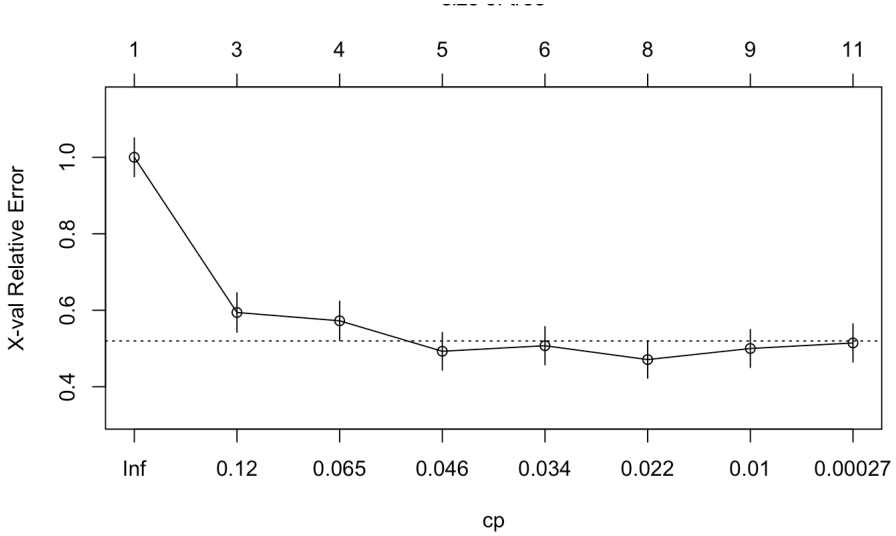
a) I have done 10-fold cross validation with 20 replicates and 9 models. Here is the result summary.

Skip	Size	Decay	Avg CV Misclass Rate
T	10	0.03	0.2815421
T	10	0.1	0.2890187
T	10	1	0.3504673
F	20	0.01	0.2892523
F	20	0.1	0.2866822
F	20	1	0.3721963
F	10	0.01	0.3107477
F	10	0.1	0.2915888
F	5	1	0.4084112

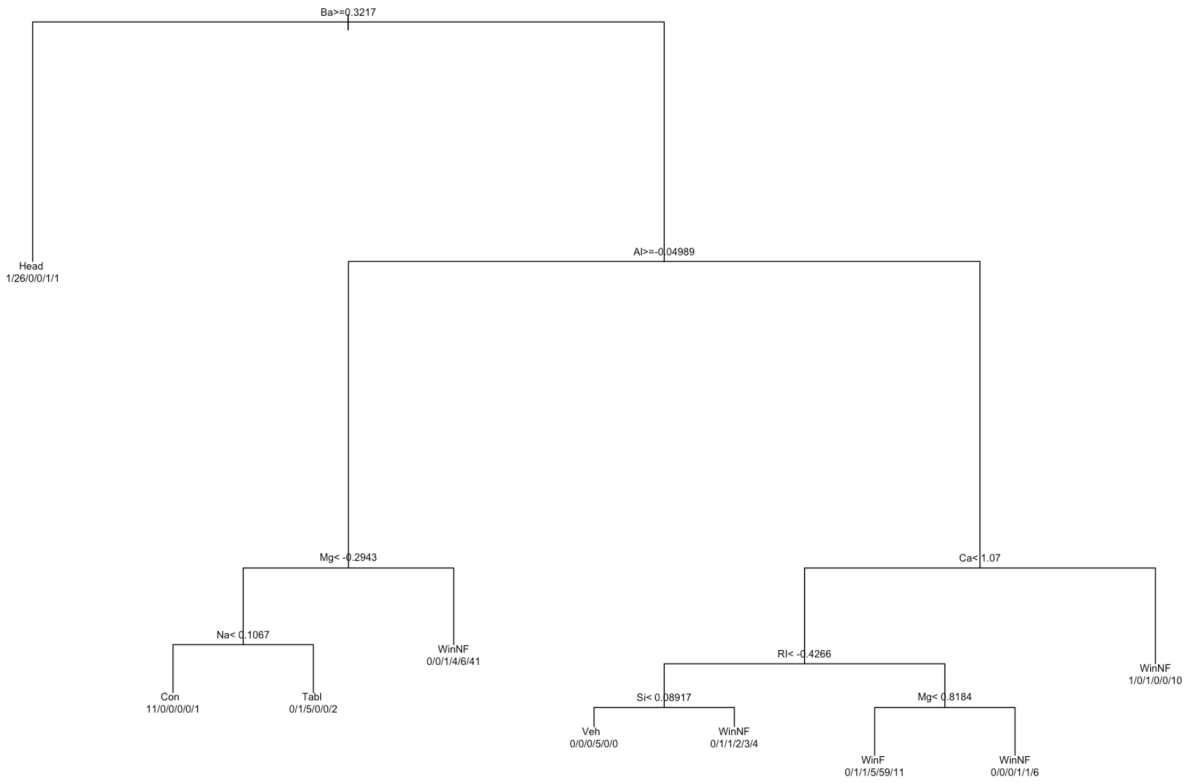
Best Model Hyperparameters with CV Misclass Rate = 0.2815
Skip = T, Size = 10, Decay = 0.03

After fitting the best model, training miss class rate is 2.8 % which is substantially lower than 28.15 %. This means that Training Misclass rate is not that reliable as it can be prone to overfitting. Therefore, looking at cross validation misclass rate is a recommended way to judge the performance of any model. Also, the model is fairly good enough having 28 % miss class rate with 6 categories as classification with high number of classes is always challenging.

b) After doing 10-fold cross validation, below is the plot of relative error vs decay parameter



From the above plot and cp table output, optimal decay parameter is **0.01449275** with **number of nodes as 7**.



Upon fitting the best tree, above plot is the final structure of the decision tree.

Performance Metrics:
Training Misclass Rate: 21.9 %
CV Misclass Rate: 30.37 %

However, training misclass rate is lower here but that can happen due to overfitting. So, it will be fair enough to compare the models based on cross validation misclass rate only. Decision tree is having 30.37 % misclass rate which is little higher than the neural network above with 28.15 % misclass rate. However the difference is not very significant.

c) After using 10-fold cross validation to fit a multinomial regression we can see that the cross validation misclassification rate for the multinomial model is 37.75% which is substantially higher than the other two models (DTrees and Neural Nets) indicating that multinomial is not the best model to fit for this dataset.

d)

- In terms of predictability, models in order of best to worst by comparing cross validation misclass rate are.
  Neural Network > Decision Trees > multinomial Regression
- The tree is most likely the easiest to understand (Mg is the main predictor variable). Because there are multiple sets of coefficients in the multinomial model as opposed to binary logistic regression, it is still rather challenging to analyze.

Bias Fairness Questions

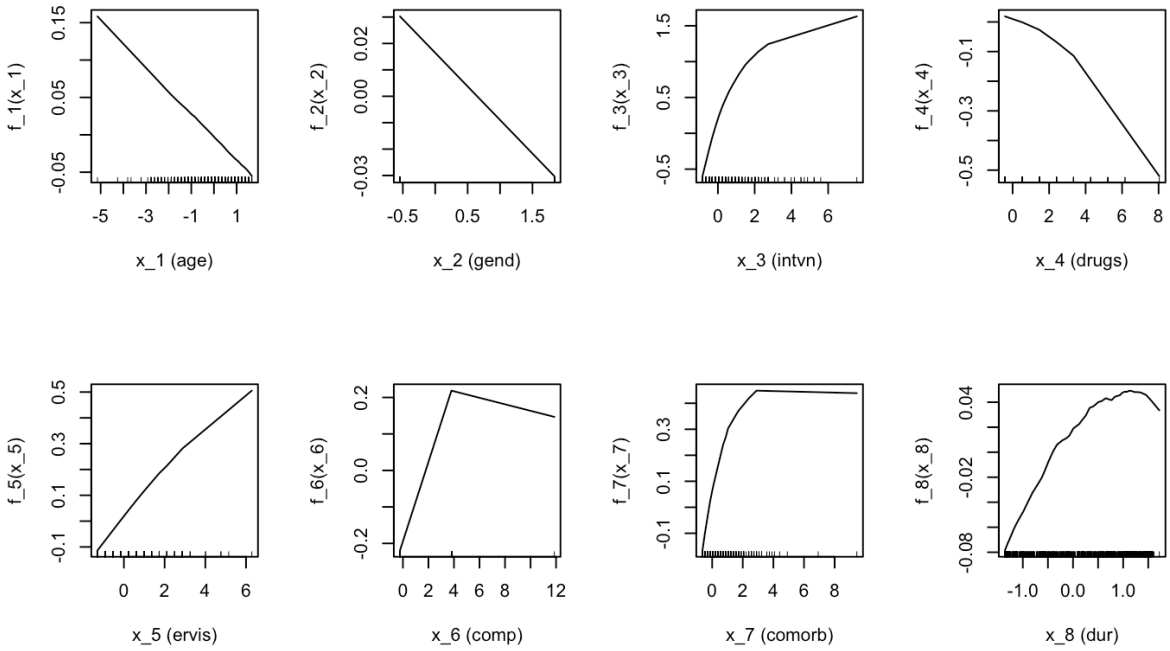
Question 1: Below are three business purposes along with bias/fairness issues that can arise

a) Insurance Premium Calculation:
   a. **Purpose**: Predictive models are used by insurers to evaluate risk and set premium costs for various policyholders. In insurance, age and gender are considered conventional risk variables because it is assumed that they are related to the possibility and expense of claims.
   b. **Bias/Fairness Issues**: Relying excessively on age and gender may result in unfair premium increases for some demographic groups, hence constituting discriminatory practices. This might not fairly represent the risk profiles of individual people and could maintain current societal disparities.

b) Healthcare Management:
   a. **Purpose**: These models could be used by insurers and healthcare providers to control total costs of treatment, identify high-risk patients for focused therapies, and more effectively allocate resources.
   b. **Bias/Fairness Issues**: Models that fail to sufficiently take into consideration the intricate interactions among age, gender, and health outcomes may misrepresent the risk for particular populations, which could result in ineffective or inappropriate care treatments.

c) Policy Development:
   a. **Purpose**: More broadly, these models can help policymakers decide how best to distribute resources, subsidies, or benefits among various demographic groups in order to better control the total cost burden.
   b. **Bias/Fairness Issues**: Policies based on skewed models have the potential to worsen already-existing imbalances by redistributing resources away from the most vulnerable or solidifying long-standing injustices.

Question 2: Below is the ALE plot from the best neural net fitted on the data

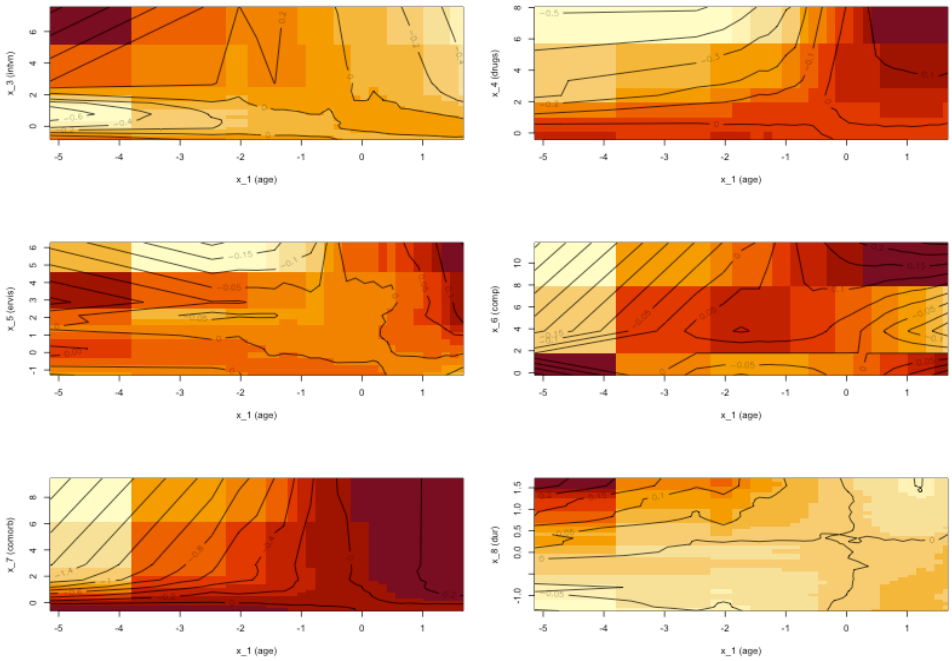


Variable Importance Table:

intvn	dur	comorb	ervis	comp	age
299.146502	53.693377	28.910902	7.213502	4.190102	3.278480

After fitting different versions of Neural Nets and Decision Trees and by analyzing the ALE plots and Variable Importance table, it can be inferred that both age and gender does not have any significant effect on predicting cost and hence there is very less influence of these two predictors on the response variable. Also, from the analysis done in Questions 1-3 , they don't have any significant interactions with any of the major predictors

Further, as we can see from these ALEplots, there is very high interaction between age and all the major variables of the model such as intvn, comp and comorb. So, there is an effect of multicollinearity on how age affects the overall model



Question 3:
Analyzing the impact of age and gender on the cost predictions of a model without considering other variables may not reveal the full extent of bias or fairness issues. Age and gender could contribute to bias in the following ways:

- **Indirect Influence**: Age and gender may not be primary predictors but could still influence the cost indirectly through correlations with other variables.
- **Proxy Variables**: Other variables in the model might act as proxies for age or gender, leading to hidden biases.
- **Group-Level Disparities**: Even if age and gender are not significant predictors, the model might still make unfair predictions for certain age or gender groups, which can be revealed by analyzing the distribution of errors or outcomes across these groups.

To ensure fairness, it's crucial to examine the model's performance across different age and gender groups and not just their direct influence as predictors. Mitigation could involve techniques like error analysis, fairness metrics evaluation, and potentially adjusting the model to account for disparate impacts.

Question 4

There can be several risks of Limited Analysis with only gender and age as predictor variables and cost as response. Some of the risks are highlighted below:

- **Confounding Variables**: Other predictors not included in the simple analysis might influence both the response variable and the sensitive predictors. Ignoring these confounders can create or hide apparent associations, leading to incorrect inferences about the relationship between the sensitive predictors and the outcome.
- **Omitted Variable Bias**: Excluding relevant predictors from the analysis can bias the estimated effects of the included variables. This can lead to over- or underestimating the relationship between the sensitive predictors and the response variable, potentially masking true bias/fairness issues.
- **Lack of Interaction Effects**: The relationship between sensitive predictors and the response variable might depend on the levels of other variables. For example, the impact of age on the cost might differ by gender, geographic location, or health status. Analyzing sensitive predictors in isolation ignores potential interaction effects that could be crucial for understanding the full picture.

However, if we include all the predictors in the model, there can be many upsides of the comprehensive model as well. Some of them are listed below:

- **Control for Confounders**: By including a broader set of predictors, the model can control for confounding variables, allowing for a clearer understanding of the direct relationship between sensitive predictors and the response variable.
- **Identify Interaction Effects**: A comprehensive model can include interaction terms to capture how the relationship between sensitive predictors and the response variable might change across levels of other variables, providing a more nuanced understanding of the data.
- **More Accurate Estimates**: Including all relevant predictors helps to reduce omitted variable bias, leading to more accurate estimates of the effects of sensitive predictors on the response variable.
- **Fairness Metrics Evaluation**: With a comprehensive model, it's possible to evaluate fairness metrics more accurately, considering the complex interactions and relationships between all predictors and the response variable. This can inform more effective bias mitigation strategies.

Appendix

Ayush Agarwal

2024-02-07

Question 1

Part 1.a)

```
df <- read_excel('HW2_data.xls')
df <- df[,-1]
df1 <- df
# standardize the predictors
df1[2:9] <- sapply(df1[2:9], function(x) ((x - mean(x)) / sd(x)))

# Fitting Linear Regression Model
out <- lm(log10(cost) ~ . , df1)
summary(out)
```

The adjusted r2 = 57.89 % which means 58 % of the variability in the log(cost) is exlained by the linear dependence on the predictors. We can say that response is somewhat predictable using linear model but non-linear models can be explored in this case for more better fit on the data

Part 1.b)

```
# Lets check the VIF Values first to check multi-colinearity before testing for p-values of predictors
vif(out)
```

None of the VIF values are large therefore no sign of multi-colinearity is there. By analysing the P-values of the predictors, intvn, comp, comorb and dur are having p-values less than 5% , therefore they are statistically significant. Since the predictors are standardized, the coefficients are on same scale, therefore we can compare their values directly by using their magnitudes. Order of Importance intvn > dur > comorb > comp

Part 1.c)

```
y_hat <- predict(out)
y <- log10(df1$cost)
e <- y-y_hat

par(mfrow = c(2, 3))
plot(df1$intvn,log10(df1$cost))
```

```
plot(df1$dur,log10(df1$cost))
plot(df1$comorb,log10(df1$cost))
plot(df1$comp,log10(df1$cost))
plot(y_hat,e)

par(mfrow = c(2, 2))
plot(out)
```

I have plotted the pair plots between significant variables and log10(cost). It can be clearly seen that non-linearity is the issue here. No predictor is linearly related with the response variable. Also the error are heterodastic and follow non-linear patterns. Therefore there is a need to evaluate the non-linear models in this case.

Question 2

```
set.seed(1911)
df <- read_excel('HW2_data.xls')
df <- df[,-1]
df1 <- df
# standardize the predictors
df1[2:9] <- sapply(df1[2:9], function(x) ((x - mean(x)) / sd(x)))

CVInd <- function(n,K) { #n is sample size; K is number of parts; returns K-length list of indices for
  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
}
```

Part 2.a

```
set.seed(1911)
##Now use multiple reps of CV to compare Neural Nets and linear reg models###
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 13 #number of different models to fit
n=nrow(df1)
y<-log10(df1$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) {
    # model 1
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=F, size=10, decay=1, maxit=1000, trace=F)
    yhat[Ind[[k]],1]<-as.numeric(predict(out,df1[Ind[[k]],]))
    # model 2
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=F, size=10, decay=2, maxit=1000, trace=F)
    yhat[Ind[[k]],2]<-as.numeric(predict(out,df1[Ind[[k]],]))
    # model 3
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=F, size=10, decay=0.1, maxit=1000, trace=F)
    yhat[Ind[[k]],3]<-as.numeric(predict(out,df1[Ind[[k]],]))
    # model 4
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=F, size=20, decay=1, maxit=1000, trace=F)
    yhat[Ind[[k]],4]<-as.numeric(predict(out,df1[Ind[[k]],]))
    # model 5
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=F, size=20, decay=5, maxit=1000, trace=F)
    yhat[Ind[[k]],5]<-as.numeric(predict(out,df1[Ind[[k]],]))
    # model 6
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=F, size=5, decay=1, maxit=1000, trace=F)
    yhat[Ind[[k]],6]<-as.numeric(predict(out,df1[Ind[[k]],]))
    # model 7
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=F, size=5, decay=0.5, maxit=1000, trace=F)
    yhat[Ind[[k]],7]<-as.numeric(predict(out,df1[Ind[[k]],]))
    # model 8
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=T, size=10, decay=0.1, maxit=1000, trace=F)
    yhat[Ind[[k]],8]<-as.numeric(predict(out,df1[Ind[[k]],]))
    # model 9
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=T, size=10, decay=1, maxit=1000, trace=F)
    yhat[Ind[[k]],9]<-as.numeric(predict(out,df1[Ind[[k]],]))
    # model 10
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=T, size=10, decay=10, maxit=1000, trace=F)
    yhat[Ind[[k]],10]<-as.numeric(predict(out,df1[Ind[[k]],]))
    # model 11
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=T, size=20, decay=1, maxit=1000, trace=F)
    yhat[Ind[[k]],11]<-as.numeric(predict(out,df1[Ind[[k]],]))
    # model 12
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=T, size=20, decay=0.01, maxit=1000, trace=F
    yhat[Ind[[k]],12]<-as.numeric(predict(out,df1[Ind[[k]],]))

    # Linear Regression Model
    out <- lm(log10(cost) ~ . , df1)
    yhat[Ind[[k]],13]<-as.numeric(predict(out,df1[Ind[[k]],]))
  } #end of k loop
  MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n
} #end of j loop
MSE
MSEAve<- apply(MSE,2,mean); MSEAve #averaged mean square CV error
MSEsd <- apply(MSE,2,sd); MSEsd   #SD of mean square CV error
r2<-1-MSEAve/var(y); r2  #CV r^2
```

Part 2.b)

```
set.seed(1911)
# Fitting the best Model
Nrep<-20 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 2 #number of different models to fit
n=nrow(df1)
y<-log10(df1$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) {
    # model 1
    out<-nnet(log10(cost)~.,df1[-Ind[[k]],], linout=T, skip=F, size=10, decay=1, maxit=1000, trace=F)
    yhat[Ind[[k]],1]<-as.numeric(predict(out,df1[Ind[[k]],]))

    # Linear Regression Model
    out <- lm(log10(cost) ~ . , df1)
    yhat[Ind[[k]],2]<-as.numeric(predict(out,df1[Ind[[k]],]))
  } #end of k loop
  MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n
} #end of j loop
MSE
MSEAve<- apply(MSE,2,mean); MSEAve #averaged mean square CV error
MSEsd <- apply(MSE,2,sd); MSEsd   #SD of mean square CV error
r2<-1-MSEAve/var(y); r2  #CV r^2
```

Part 2.C

```
set.seed(1911)
out<-nnet(log10(cost)~.,df1, linout=T, skip=F, size=10, decay=1, maxit=1000, trace=F)
summary(out)

df1 = data.frame(df1)
yhat <- function(X.model, newdata) as.numeric(predict(X.model, newdata))
par(mfrow=c(2,4))
for (j in 1:8) {ALEPlot(df1[,2:9], out, pred.fun=yhat, J=j, K=50, NA.plot = TRUE)
rug(df1[,j+1]) } ## This creates main effect ALE plots for all 8 predictors
par(mfrow=c(1,1))

par(mfrow=c(2,2))
ALEPlot(df1[,2:9], out, pred.fun=yhat, J=c(3,4), K=50, NA.plot = TRUE)
ALEPlot(df1[,2:9], out, pred.fun=yhat, J=c(3,5), K=50, NA.plot = TRUE)
ALEPlot(df1[,2:9], out, pred.fun=yhat, J=c(3,6), K=50, NA.plot = TRUE)
ALEPlot(df1[,2:9], out, pred.fun=yhat, J=c(3,7), K=50, NA.plot = TRUE)
par(mfrow=c(1,1))
```

Part 2.D

```
# Plotting the residuals against predicted values
y_hat = predict(out)
e = y - y_hat
plot(y_hat, e)
```

Problem 3

```
set.seed(1911)
df <- read_excel('HW2_data.xls')
df <- df[,-1]
df1 <- df
# standardize the predictors
df1[2:9] <- sapply(df1[2:9], function(x) ((x - mean(x)) / sd(x)))
head(df1)
```

Part 3.a

```
set.seed(1911)
control = rpart.control(minbucket = 5, cp = 0.0001, maxsurrogate = 0, usesurrogate = 0, xval = 10)
df1.tr <- rpart(log10(cost)~., df1,method="anova", control = control)

# Plot of CV R2 vs Size
plotcp(df1.tr)

cptable <- data.frame(printcp(df1.tr)); cptable;

#Checking for the values with the least cv error
least_value_cp <- cptable[which.min(cptable$xerror), ]
least_value_cp$CP #optimum decay parameter
least_value_cp$nsplit #optimum size of tree
```

Part 3.b

```
df1.tr1 <- prune(df1.tr, cp=0.003140835)

png("Best_tree_plot.png", width=2000, height=1500, res=150)

# Set the margins and plot
par(mar=c(7, 5, 4, 2) + 0.1)
plot(df1.tr1, uniform=FALSE)
text(df1.tr1, use.n=TRUE, cex=0.6)

# Close the device
dev.off()
```

```r
# Checking the predictive power of the fit
yhat_dt<-predict(df1.tr1); e<-y-yhat_dt
# Training R square
c(1-var(e)/var(y), 1-df1.tr1$cptable[nrow(df1.tr1$cptable),3])
# Cross Validation R Square
1-df1.tr1$cptable[nrow(df1.tr1$cptable),4]
```

Part 3.c

```r
df1.tr1$variable.importance
```

Part 3.d

```r
# Plot Error Plots with respect to top 3 most important predictors
# intvn(4), dur(9), comorb(8)
par(mfrow=c(2,2))
plot(df1[,4],e)
plot(df1[,9],e)
plot(df1[,8],e)
plot(yhat,e)
```

Problem 4

```r
set.seed(1234)
data2 <- read.delim("fgl.txt")

#Inspecting the data
head(data2)

#Standardizing the predictor variables and making sure type is a factor
data2 = data2 %>%
  mutate(across(
    .cols = where(is.numeric),
    .fns = ~ ( . - mean(., na.rm = TRUE)) / sd(., na.rm = TRUE)
  )) %>%
  mutate(type = as.factor(type))

table(data2$type)
```

Part 4.a

```r
set.seed(1234)
# Fitting the best Model
```

```
Nrep<-20 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 9 #number of different models to fit
n=nrow(data2)
y<-data2$type
yhat=matrix(0,n,n.models)
CV.MisclassRate<-matrix(0,Nrep,n.models)



for (j in 1:Nrep) {
  Ind<-CVInd(n,K)
  for (k in 1:K) {
    out<-nnet(type~.,data2[-Ind[[k]],],linout=F,skip=T,size=10,decay=.03, maxit=1000,trace=F)
    yhat[Ind[[k]],1]<-predict(out,data2[Ind[[k]],],type = 'class')

    out<-nnet(type~.,data2[-Ind[[k]],],linout=F,skip=T,size=10,decay=0.1, maxit=1000,trace=F)
    yhat[Ind[[k]],2]<-predict(out,data2[Ind[[k]],],type = 'class')

    out<-nnet(type~.,data2[-Ind[[k]],],linout=F,skip=T,size=10,decay=1, maxit=1000,trace=F)
    yhat[Ind[[k]],3]<-predict(out,data2[Ind[[k]],],type = 'class')

    out<-nnet(type~.,data2[-Ind[[k]],],linout=F,skip=F,size=20,decay=.01, maxit=1000,trace=F)
    yhat[Ind[[k]],4]<-predict(out,data2[Ind[[k]],],type = 'class')

    out<-nnet(type~.,data2[-Ind[[k]],],linout=F,skip=F,size=20,decay=.1, maxit=1000,trace=F)
    yhat[Ind[[k]],5]<-predict(out,data2[Ind[[k]],],type = 'class')

    out<-nnet(type~.,data2[-Ind[[k]],],linout=F,skip=F,size=20,decay=1, maxit=1000,trace=F)
    yhat[Ind[[k]],6]<-predict(out,data2[Ind[[k]],],type = 'class')

    out<-nnet(type~.,data2[-Ind[[k]],],linout=F,skip=F,size=10,decay=.01, maxit=1000,trace=F)
    yhat[Ind[[k]],7]<-predict(out,data2[Ind[[k]],],type = 'class')

    out<-nnet(type~.,data2[-Ind[[k]],],linout=F,skip=F,size=10,decay=.1, maxit=1000,trace=F)
    yhat[Ind[[k]],8]<-predict(out,data2[Ind[[k]],],type = 'class')

    out<-nnet(type~.,data2[-Ind[[k]],],linout=F,skip=F,size=5,decay=1, maxit=1000,trace=F)
    yhat[Ind[[k]],9]<-predict(out,data2[Ind[[k]],],type = 'class')
  } #end of k loop
  CV.MisclassRate[j,]=apply(yhat,2,function(x) sum(y != x)/n)
} #end of j loop
CV.MisclassRate
CV.MisclassRateAve<- apply(CV.MisclassRate,2,mean); CV.MisclassRateAve #averaged CV misclass rate
CV.MisclassRateSD <- apply(CV.MisclassRate,2,sd); CV.MisclassRateSD   #SD of CV misclass rate
```

Part 4.a.1

```
#Fitting the best neural network model
out<-nnet(type~.,data2,linout=F,skip=T,size=10,decay=.03, maxit=1000,trace=F)
yhat<-predict(out,type="class")
sum(y != yhat)/length(y)  #training misclassification rate
```

Part 4.b

```
set.seed(1234)
control <- rpart.control(minbucket = 5, cp = 0.00001, maxsurrogate = 0, usesurrogate = 0, xval = 10)
data2.tr <- rpart(type~.,data2, method = "class", control = control)

plotcp(data2.tr) #plotting the x-val relative error vs cp
cptable = data.frame(printcp(data2.tr)); cptable; #printing out the table too

#Checking for the values with the least cv error
least_value_cp <- cptable[which.min(cptable$xerror), ]
least_value_cp$CP #optimum decay parameter
least_value_cp$nsplit #optimum size of tree

# 0.01449275, 7
#prune to optimal size
data2.tr1 <- prune(data2.tr, cp=0.01449275)  #approximately the cp corresponding to the optimal size
data2.tr1

# Save to a PNG file with increased dimensions and higher resolution
png("prune_best_tree_plot2.png", width=2000, height=1500, res=150)  # Width and height in pixels, res i

# Set the margins and plot
par(mar=c(7, 5, 4, 2) + 0.1)
plot(data2.tr1, uniform=FALSE)
text(data2.tr1, use.n=TRUE, cex=0.6)

# Close the device
dev.off()

# Checking the predictive power of the fit
yhat<-predict(data2.tr1, type="class")
sum(yhat != y)/nrow(data2) #training misclassification rate

#We need to calculate the misclassification rate from the cp table
cv_misclass_rate = 0.64486 * least_value_cp$xerror
cv_misclass_rate
```

Part 4.c

```
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(data2)
y<-as.numeric(data2$type)
yhat=matrix(0,n,n.models)
CV.rate<-matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
```

```
  Ind<-CVInd(n,K)
  for (k in 1:K) {
    out = multinom(type ~.,data2[-Ind[[k]],],trace = F)
    yhat[Ind[[k]],1]<-predict(out,data2[Ind[[k]],],type = 'class')
  } #end of k loop
  CV.rate[j,]=apply(yhat,2,function(x) sum(y != x)/n)
} #end of j loop
CV.rate
CV.rateAve<- apply(CV.rate,2,mean); CV.rateAve #averaged CV misclass rate
CV.rateSD <- apply(CV.rate,2,sd); CV.rateSD   #SD of CV misclass rate
```