hw02

Samuel Swain

2023-02-10

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

Problem 1

1(a)

```
## MSE:
##  - Average: 1.947016
##  - SD 0.02844145
```

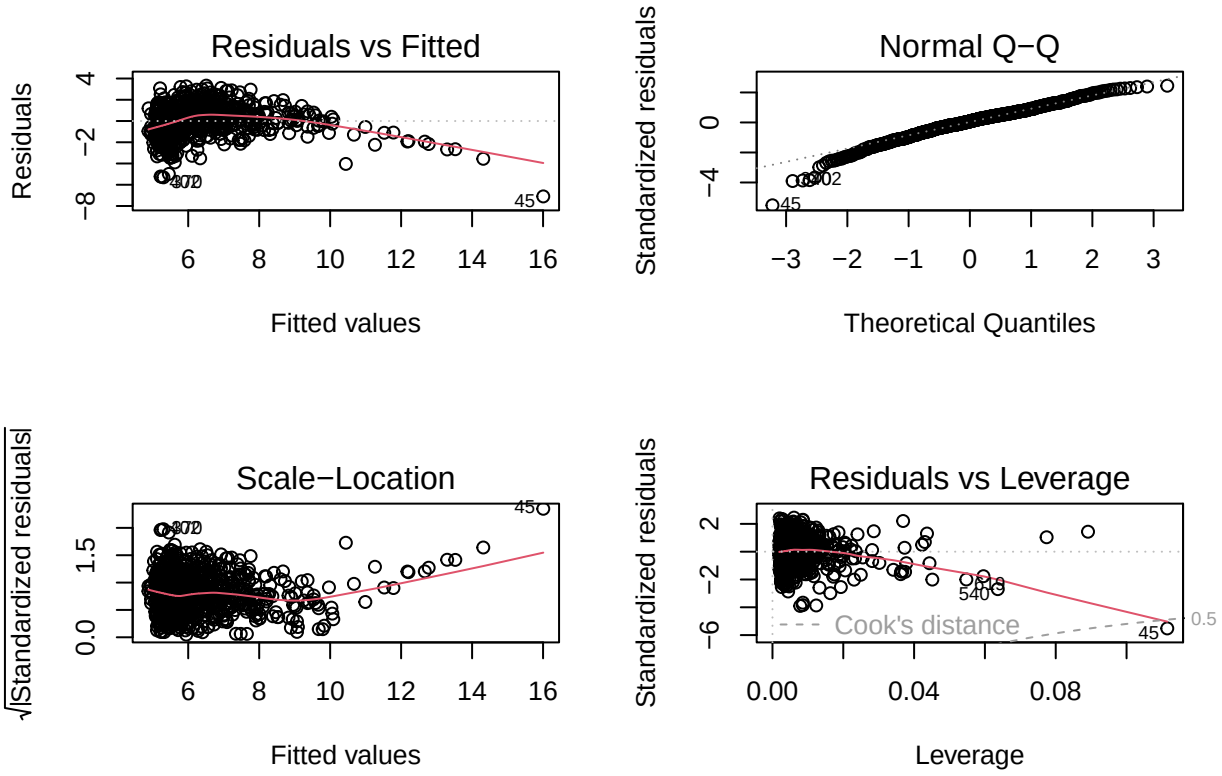
As we can see above, the average error is 1.9470162 per prediction with a standard deviation of 0.0284414. The predictive power of this model can definitely be improved or we can explore other models that will do a better job at predicting cost.

1(b)

```
##
## Call:
## lm(formula = cost ~ age + gend + intvn + drugs + ervis, data = df_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.0979 -0.7699  0.1101  0.8554  3.3293
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.290013   0.048606 129.408  < 2e-16 ***
## age          0.001477   0.048832   0.030  0.97587
## gend        -0.071141   0.048982  -1.452  0.14680
## intvn        1.283944   0.052426  24.491  < 2e-16 ***
## drugs       -0.088402   0.057351  -1.541  0.12362
## ervis        0.178760   0.060489   2.955  0.00322 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.364 on 782 degrees of freedom
## Multiple R-squared:  0.491,  Adjusted R-squared:  0.4877
## F-statistic: 150.9 on 5 and 782 DF,  p-value: < 2.2e-16
```

Total number of interventions and number emergency room visits seem to have the highest effect on the cost of the subscriber. Increasing intvn by only one will increase the cost of the subscriber by about 812.08 whereas increasing ervis by just one will also increase the cost of the subscriber by about 376.53 dollars.

1(c)



As we can see from the plots above, there are a lot of problems with using a linear model to attain the relationship between the features and the cost. The problems are listed below: - Non-linearity of the data based on the Residuals vs Fitted plot - A few outliers based on the Residuals vs Leverage plot

Problem 2

2(a)

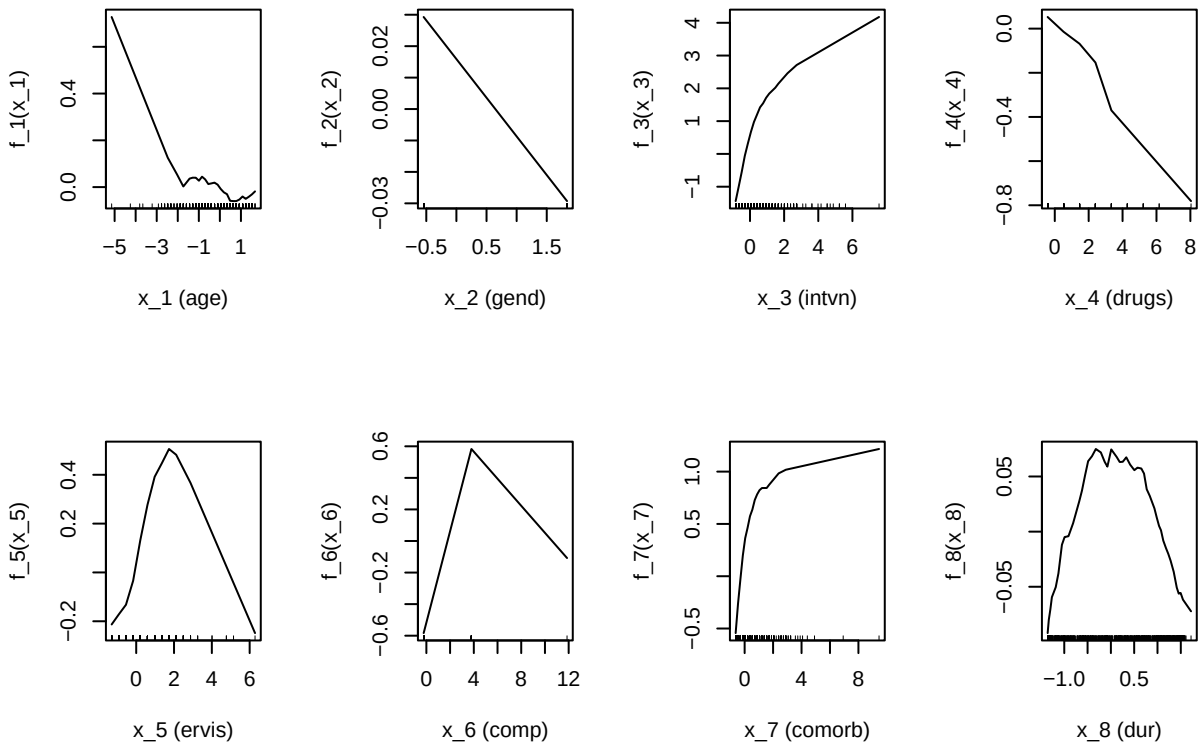
```
## Best NNet:
##  Model: 1
##  Decay: 15
##  Size: 0.5
```

2(b)

```
## MSE:
##  - Average: 1.458738
##  - SD 0.04090414
```

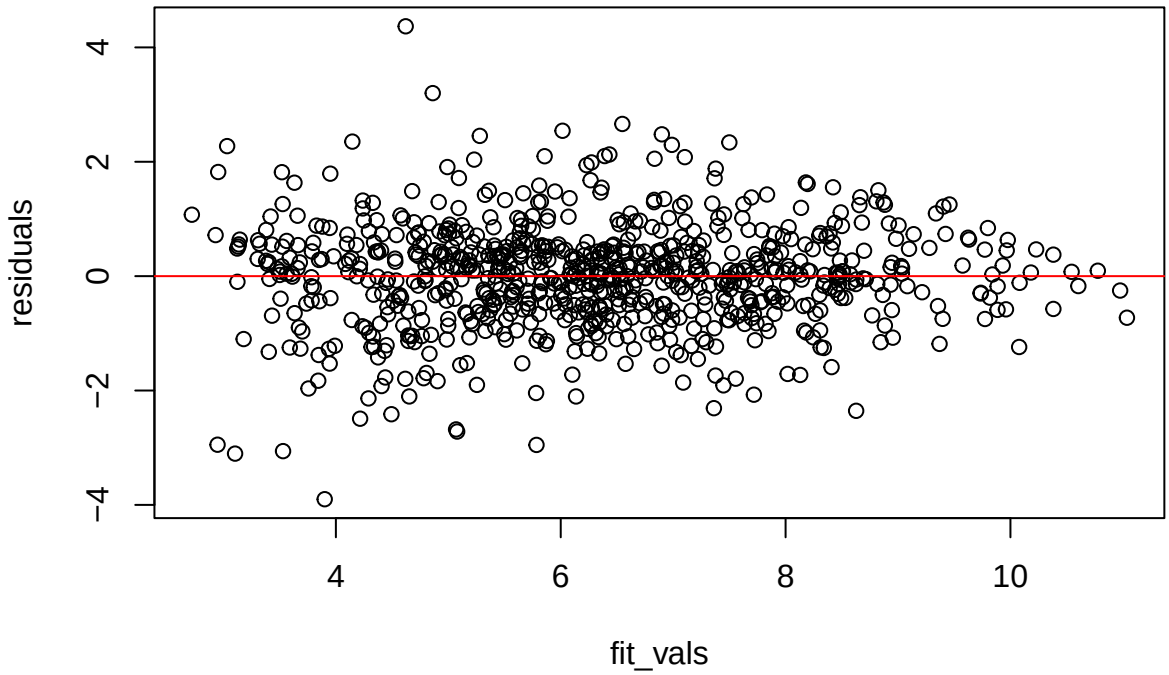
The best Neural Net attained above has a size of 15 and a decay rate of 0.5. In terms of predictive power, I think the model is very good. With an average CV MSE of 1.9754511, it performs better than the linear model.

2(c)



Below are the top three features ranked by strength of effect: - Intervention (intvn): Positive - Complications (comp): Positive for the first 25% and then negative for the last 75% - Drugs (drugs): Negative for the first 20% and then negative for the last 80%

2(d)



Looking at the fitted versus residual plot above, we can see pretty much all of the non linearity has been captured.

Problem 3

3(a)

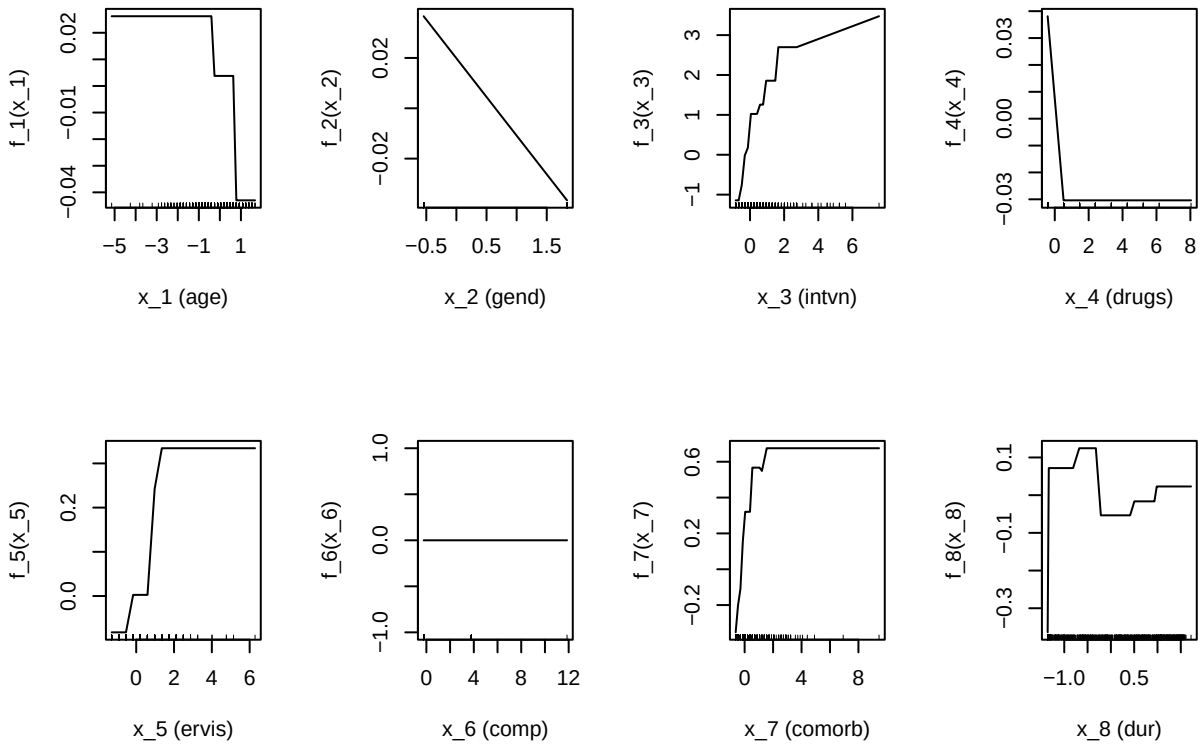
```
## Best Tree:
##  Model: 2
##  Minbucket: 10
##  CP: 0.001
```

3(b)

```
## MSE:
##  - Average: 1.281127
##  - SD 0.0268458
```

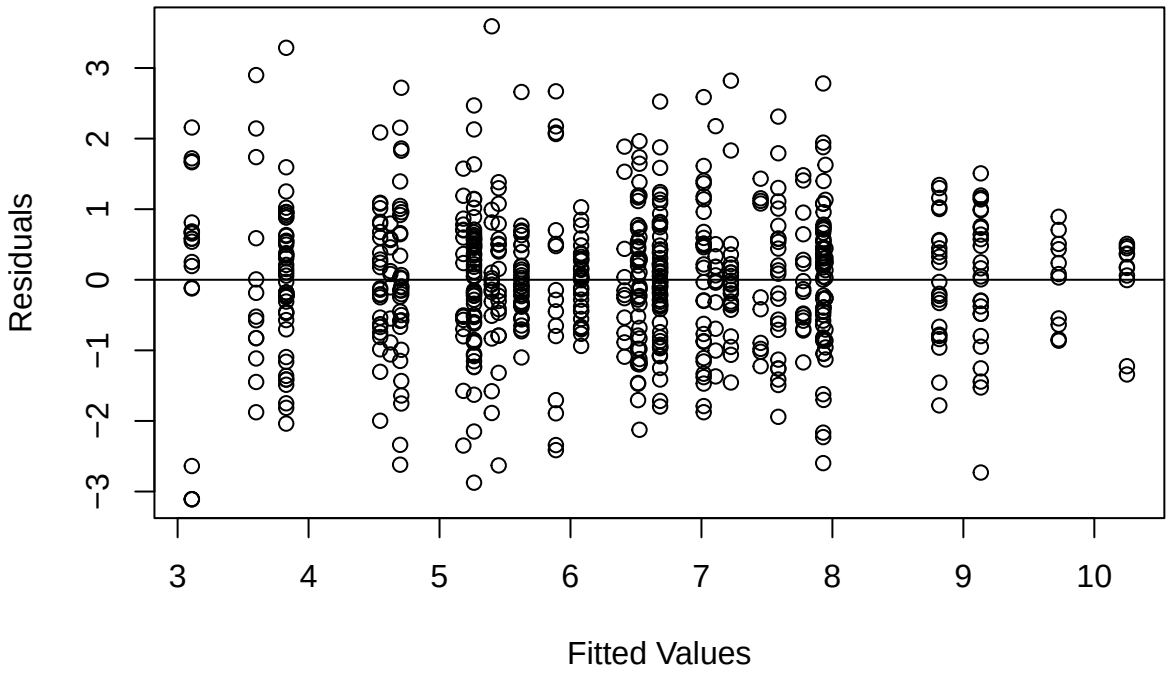
The best Tree attained above has a min-bucket of 10 and a decay rate of 0.001. In terms of predictive power, I think the model is really good. It's better at predicting than the linear and N Net model given average CV MSE of 1.2811271.

3(c)



As we can see above, intervention (intvn) is again the most important variable when fitting the tree. The cost of the subscriber goes up for each additional intervention until two, then the effect levels off. Another important variable is comorb. We can see that the effect is not as big as intvn but it still explains about 1.5 total variation in cost.

3(d)



The pattern of this tree Fitted vs Residual plot is similar to the N Net plot. It seems to have captured most of the non linearity. There is no trend down or up as we increase fitted values.

3(e)

I would use the tree model as it produces the lowest MSE out of the three models. It also is able to capture all of the non linearity in the data.

Question 4

4(a)

```
## Best NNet:
##  Model: 2
##  Size: 20
##  Decay: 0.1
```

4(b)

```
## Best Tree:
##  Model: 2
```

```
##  CP: 1e-04
##  Minbucket: 5
```

4(c)

```
## [INFO] -- MNLogit model fit
```

4(d)

```
## Models
##  - NNet: 0.6137072
##  - Tree: 0.7009346
##  - MNLogit: 0.5934579
```

The logit model is more interpretable than the nnet but about the same as the tree model. The only extra piece of information the logit model gives us is access to the exact change in likelihoods / probabilities from looking at the coefficients. The tree model has a higher cross validation accuracy as well as similar interpretability. We should choose the tree model over the nnet all day. When looking at the MNlogit model we see that it gets about 9% less predictions correct. Given the slightly higher interpretability of the MNlogit model versus the much higher prediction accuracy of the Tree model, using the Tree model will be more appropriate for predicting glass type.

Appendix

Commented out code is the output used to attain the results above. Commended to shorted final submission pdf.

```
# Libraries
library(boot)
library(nnet)
library(glmnet)
library(ALEPlot)
library(rpart)

# Cross Validation
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
}
```

```r
# Standardize Function
standardize_predictors <- function(data) {
  predictors <- data
  standardized_predictors <- scale(predictors)
  data <- standardized_predictors
  return(data)
}
```

Problem 1

```r
library(readxl)
df_1 = read_excel("HW2_data.xls")
df_1 = df_1[, -c(1)]

# Vars
df_1$cost<-log(df_1$cost)
X_1 = df_1[, -c(1)]
df_1[, c(2, 3, 4, 5, 6, 7, 8, 9)] = standardize_predictors(X_1)
```

1(a)

code:

```r
fit_1a <- lm(cost ~ age + gend + intvn + drugs + ervis, data = df_1)

##Now use multiple reps of CV to compare Neural Nets and linear reg models###
Nrep<-5 #number of replicates of CV
K<-3  #K-fold CV on each replicate
n.models = 1 #number of different models to fit
n=nrow(df_1)
y<-df_1$cost
yhat=matrix(0,n,n.models)
MSE<-matrix(0,Nrep,n.models)
RMSE<-matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
  Ind<-CVInd(n,K)
  for (k in 1:K) {
    out<-lm(cost ~ age + gend + intvn + drugs + ervis, data = df_1[-Ind[[k]],])
    yhat[Ind[[k]],1]<-as.numeric(predict(out,df_1[Ind[[k]],]))
  } #end of k loop
  MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n
  RMSE[j,]=apply(yhat,2,function(x) sqrt(sum((y-x)^2))/n)
} #end of j loop
MSEAve<- apply(MSE,2,mean) #averaged mean square CV error
MSEsd <- apply(MSE,2,sd)   #SD of mean square CV error
RMSEAve<- apply(RMSE,2,mean) #averaged mean square CV error
RMSEsd <- apply(RMSE,2,sd)   #SD of mean square CV error

# cat("MSE:", "\n", "- Average:", MSEAve, "\n", "- SD", MSEsd)
```

1(b)

code:

```
# summary(fit_1a)
```

1(c)

code:

```
# par(mfrow = c(2,2))
# plot(fit_1a)
```

Problem 2

2(a)

code:

```
##Now use multiple reps of CV to compare Neural Nets and linear reg models###
Nrep<-3 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 4 #number of different models to fit
n=nrow(df_1)
y<-df_1$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~.,df_1[-Ind[[k]],], linout=T, skip=F, size=15, decay=.5, maxit=1000, trace=F)
    yhat[Ind[[k]],1]<-as.numeric(predict(out,df_1[Ind[[k]],]))
    out<-nnet(cost~.,df_1[-Ind[[k]],], linout=T, skip=F, size=20, decay=.1, maxit=1000, trace=F)
    yhat[Ind[[k]],2]<-as.numeric(predict(out,df_1[Ind[[k]],]))
    out<-nnet(cost~.,df_1[-Ind[[k]],], linout=T, skip=F, size=25, decay=.01, maxit=1000, trace=F)
    yhat[Ind[[k]],3]<-as.numeric(predict(out,df_1[Ind[[k]],]))
    out<-nnet(cost~.,df_1[-Ind[[k]],], linout=T, skip=F, size=30, decay=0, maxit=1000, trace=F)
    yhat[Ind[[k]],4]<-as.numeric(predict(out,df_1[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) #averaged mean square CV error
MSEsd <- apply(MSE,2,sd)  #SD of mean square CV error

cat("Best NNet:", "\n", "Model:", which.min(MSEAve), "\n",
    "Decay:", c(15, 20, 25, 30)[which.min(MSEAve)], "\n",
    "Size:", c(0.5, 0.1, 0.01, 0)[which.min(MSEAve)])
```

```
## Best NNet:
##  Model: 1
```

```
##  Decay: 15
##  Size: 0.5
```

2(b)

code:

```
best_nnet_2b <- nnet(cost~., df_1, linout=T, skip=F,
                     size=c(15, 20, 25, 30)[which.min(MSEAve)],
                     decay=c(0.5, 0.1, 0.01, 0)[which.min(MSEAve)],
                     maxit=1000, trace=F)
cat("MSE:", "\n", "- Average:", MSEAve[which.min(MSEAve)], "\n", "- SD", MSEsd[which.min(MSEAve)])
```

```
## MSE:
##  - Average: 1.433667
##  - SD 0.0200601
```

2(c)

code:

```
df_newdata = df_1[2:9]
df_newdata <- as.data.frame(df_newdata)

yhat <- function(X.model, newdata) as.numeric(predict(X.model, newdata))

# par(mfrow=c(2,4))
# for (j in 1:8)  {ALEPlot(df_newdata, best_nnet_2b, pred.fun=yhat, J=j, K=50, NA.plot = TRUE)
#   rug(df_newdata[,j]) }  ## This creates main effect ALE plots for all 8 predictors
# par(mfrow=c(1,1))
```

2(d)

code:

```
# Extract the fitted values and residuals
fit_vals <- fitted(best_nnet_2b)
residuals <- residuals(best_nnet_2b)

# Plot the fitted values against the residuals
# plot(fit_vals, residuals)
# abline(h=0, col="red")
```

Problem 3

3(a)

code:

```
##Now use multiple reps of CV to compare Neural Nets and linear reg models###
Nrep<-3 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 4 #number of different models to fit
n=nrow(df_1)
y<-df_1$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) {
    control <- rpart.control(minbucket = 10, cp = 0.0001, maxsurrogate = 0, usesurrogate = 0, xval = 0)
    out <- rpart(cost~.,df_1[-Ind[[k]],], method = "anova", control = control)
    yhat[Ind[[k]],1] <- as.numeric(predict(out,df_1[Ind[[k]],]))
    control <- rpart.control(minbucket = 10, cp = 0.001, maxsurrogate = 0, usesurrogate = 0, xval = 0)
    out <- rpart(cost~.,df_1[-Ind[[k]],], method = "anova", control = control)
    yhat[Ind[[k]],2]<-as.numeric(predict(out,df_1[Ind[[k]],]))
    control <- rpart.control(minbucket = 15, cp = 0.01, maxsurrogate = 0, usesurrogate = 0, xval = 0)
    out <- rpart(cost~.,df_1[-Ind[[k]],], method = "anova", control = control)
    yhat[Ind[[k]],3]<-as.numeric(predict(out,df_1[Ind[[k]],]))
    control <- rpart.control(minbucket = 20, cp = 0.1, maxsurrogate = 0, usesurrogate = 0, xval = 0)
    out <- rpart(cost~.,df_1[-Ind[[k]],], method = "anova", control = control)
    yhat[Ind[[k]],4]<-as.numeric(predict(out,df_1[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) #averaged mean square CV error
MSEsd <- apply(MSE,2,sd)  #SD of mean square CV error

# cat("Best Tree:", "\n", "Model:", which.min(MSEAve), "\n", "Minbucket:", c(10, 10, 15, 20)[which.min(
```

3(b)

code:

```
control <- rpart.control(minbucket = c(10, 10, 15, 20)[which.min(MSEAve)], cp = c(0.0001, 0.001, 0.01, 
best_tree_3b <- rpart(cost~.,df_1[-Ind[[k]],], method = "anova", control = control)

# cat("MSE:", "\n", "- Average:", MSEAve[which.min(MSEAve)], "\n", "- SD", MSEsd[which.min(MSEAve)], "\
```

3(c)

code:

```
yhat <- function(X.model, newdata) as.numeric(predict(X.model, newdata))

# par(mfrow=c(2,4))
# for (j in 1:8)  {ALEPlot(df_newdata, best_tree_3b, pred.fun=yhat, J=j, K=50, NA.plot = TRUE)
#   rug(df_newdata[,j]) }  ## This creates main effect ALE plots for all 8 predictors
# par(mfrow=c(1,1))
```

3(d)

code:

```
# Get fitted values and residuals
fitted_values <- predict(best_tree_3b)
residuals <- df_1[-Ind[[k]], "cost"] - fitted_values

# Plot fitted versus residuals
# plot(fitted_values, residuals$cost, xlab = "Fitted Values", ylab = "Residuals")
# abline(0, 0)
```

Question 4

```
library(readxl)
df_4 = read_excel("HW2_data.xls", sheet = "FGL data")
df_4 = df_4[, -c(1)]
df_4[, -c(10)] = scale(df_4[, -c(10)])
df_4$type <- as.factor(df_4$type)

set.seed(420)
```

4(a)

code:

```
Nrep <- 3 #number of replicates of CV
K <- 10  #K-fold CV on each replicate
n.models <- 4 #number of different models to fit
n <- nrow(df_4)
y <- df_4$type
yhat <- matrix(0, n, n.models)
Accuracy <- matrix(0, Nrep, n.models)

for (j in 1:Nrep) {
  Ind <- CVInd(n, K)
  for (k in 1:K) {
    out <- nnet(type ~ ., df_4[-Ind[[k]],], linout=F, skip=F, size=10, decay=.5, maxit=10, trace=F) # M
    yhat[Ind[[k]], 1] <- predict(out, df_4[Ind[[k]],], type="class")
    out <- nnet(type ~ ., df_4[-Ind[[k]],], linout=F, skip=F, size=15, decay=.5, maxit=10, trace=F) # M
    yhat[Ind[[k]], 2] <- predict(out, df_4[Ind[[k]],], type="class")
    out <- nnet(type ~ ., df_4[-Ind[[k]],], linout=F, skip=F, size=20, decay=.5, maxit=10, trace=F) # M
    yhat[Ind[[k]], 3] <- predict(out, df_4[Ind[[k]],], type="class")
    out <- nnet(type ~ ., df_4[-Ind[[k]],], linout=F, skip=F, size=25, decay=.5, maxit=10, trace=F) # M
    yhat[Ind[[k]], 4] <- predict(out, df_4[Ind[[k]],], type="class")
  } #end of k loop
  Accuracy[j,] <- apply(yhat, 2, function(x) mean(x == y)) # Accuracy calculation
} #end of j loop

AccuracyAve <- apply(Accuracy, 2, mean) #averaged mean accuracy
```

```
AccuracySd <- apply(Accuracy, 2, sd)  #SD of mean accuracy

best_nn_vars = c(c(15, 20, 25, 30)[which.max(AccuracyAve)], c(0.5, 0.1, 0.01, 0)[which.max(AccuracyAve)]

# cat("Best NNet:", "\n", "Model:", which.max(AccuracyAve), "\n",
#     "Size:", c(15, 20, 25, 30)[which.max(AccuracyAve)], "\n",
#     "Decay:", c(0.5, 0.1, 0.01, 0)[which.max(AccuracyAve)])
```

4(b)

code:

```
Nrep <- 3 #number of replicates of CV
K <- 10  #K-fold CV on each replicate
n.models <- 4 #number of different models to fit
n <- nrow(df_4)
y <- y <- as.numeric(df_4$type)
yhat <- matrix(0, n, n.models)
Accuracy <- matrix(0, Nrep, n.models)

for (j in 1:Nrep) {
  Ind <- CVInd(n, K)
  for (k in 1:K) {
    control <- rpart.control(minbucket = 3, cp = 0.00001, maxsurrogate = 0, usesurrogate = 0, xval = 0)
    out <- rpart(type~.,df_4[-Ind[[k]],], method = "class", control = control)
    yhat[Ind[[k]],1] <- predict(out,df_4[Ind[[k]],], type = "class")
    control <- rpart.control(minbucket = 5, cp = 0.0001, maxsurrogate = 0, usesurrogate = 0, xval = 0)
    out <- rpart(type~.,df_4[-Ind[[k]],], method = "class", control = control)
    yhat[Ind[[k]],2]<-predict(out,df_4[Ind[[k]],], type = "class")
    control <- rpart.control(minbucket = 7, cp = 0.001, maxsurrogate = 0, usesurrogate = 0, xval = 0)
    out <- rpart(type~.,df_4[-Ind[[k]],], method = "class", control = control)
    yhat[Ind[[k]],3]<-predict(out,df_4[Ind[[k]],], type = "class")
    control <- rpart.control(minbucket = 10, cp = 0.01, maxsurrogate = 0, usesurrogate = 0, xval = 0)
    out <- rpart(type~.,df_4[-Ind[[k]],], method = "class", control = control)
    yhat[Ind[[k]],4]<-predict(out,df_4[Ind[[k]],], type = "class")
  } #end of k loop
  Accuracy[j,] <- apply(yhat, 2, function(x) mean(x == y)) # Accuracy calculation
} #end of j loop

AccuracyAve <- apply(Accuracy, 2, mean) #averaged mean accuracy
AccuracySd <- apply(Accuracy, 2, sd)  #SD of mean accuracy

best_tree_vars = c(c(0.00001, 0.0001, 0.001, 0.01)[which.max(AccuracyAve)], c(3, 5, 7, 10)[which.max(Ac

# cat("Best Tree:", "\n", "Model:", which.max(AccuracyAve), "\n",
#     "CP:", c(0.00001, 0.0001, 0.001, 0.01)[which.max(AccuracyAve)], "\n",
#     "Minbucket:", c(3, 5, 7, 10)[which.max(AccuracyAve)])
```

4(c)

code:

```
Nrep <- 3 #number of replicates of CV
K <- 3  #K-fold CV on each replicate
n.models <- 1 #number of different models to fit
n <- nrow(df_4)
y <- as.numeric(df_4$type)
yhat <- matrix(0, n, n.models)
Accuracy <- matrix(0, Nrep, n.models)

for (j in 1:Nrep) {
  Ind <- CVInd(n, K)
  for (k in 1:K) {
    model4 <- multinom(type~., df_4[-Ind[[k]],], maxit=1000, trace=F)
    yhat[Ind[[k]],1] <- predict(model4, df_4[Ind[[k]],], type = "class")
  } #end of k loop
  Accuracy[j,] <- apply(yhat, 2, function(x) mean(x == y)) # Accuracy calculation
} #end of j loop

AccuracyAve <- apply(Accuracy, 2, mean) #averaged mean accuracy
AccuracySd <- apply(Accuracy, 2, sd)  #SD of mean accuracy
```

4(d)

code:

```
Nrep <- 3 #number of replicates of CV
K <- 4  #K-fold CV on each replicate
n.models <- 3 #number of different models to fit
n <- nrow(df_4)
y <- as.numeric(df_4$type)
yhat <- matrix(0, n, n.models)
Accuracy <- matrix(0, Nrep, n.models)

check_acc <- function(x) {
  return(mean(x == y))
}

for (j in 1:Nrep) {
  Ind <- CVInd(n, K)
  for (k in 1:K) {
    out <- nnet(type ~ ., df_4[-Ind[[k]],], linout=F, skip=F, size=best_nn_vars[1], decay=best_nn_vars[
    yhat[Ind[[k]], 1] <- as.factor(predict(out, df_4[Ind[[k]],], type="class"))

    control <- rpart.control(minbucket = best_tree_vars[2], cp = best_tree_vars[1], maxsurrogate = 0, us
    out <- rpart(type~.,df_4[-Ind[[k]],], method = "class", control = control)
    prune(tree = out, cp = best_tree_vars[1])
    yhat[Ind[[k]], 2] <- predict(out,df_4[Ind[[k]],], type = "class")

    model4 <- multinom(type~., df_4[-Ind[[k]],], maxit=1000, trace=F)
    yhat[Ind[[k]], 3] <- predict(model4, df_4[Ind[[k]],], type = "class")

  } #end of k loop

  Accuracy[j,] <- apply(yhat, 2, check_acc) # Accuracy calculation
```

```
} #end of j loop

AccuracyAve <- apply(Accuracy, 2, mean) #averaged mean accuracy
AccuracySd <- apply(Accuracy, 2, sd)  #SD of mean accuracy

# cat("Models", "\n",
#     "- NNet:", AccuracyAve[1], "\n",
#     "- Tree:", AccuracyAve[2], "\n",
#     "- MNLogit:", AccuracyAve[3])
```