Predictive Analytics II: Assignment # 3

Question 1

a) Below is the table of K vs Cross Validation SSE

K	**CV SSE**
1	393
5	233
8	223
10	224
15	231
20	234

Best K = 8 (CV SSE = 223)

Using N-fold cross-validation (CV) typically provides a more precise estimate of the test error variance compared to K-fold CV when K is less than n. However, for most methods, especially when n is large, n-fold CV requires significantly more computation time since it necessitates training n different models. In the case of nearest neighbor algorithms, there is no actual training phase, hence n-fold CV does not demand more computational effort than K-fold CV.

b) For K=8, CV estimate of the prediction error standard deviation = 0.529

c) Predicted cost for a person with age=59, gend=0, intvn=10, drugs=0,ervis=3, comp=0, comorb=4, and dur=300:

Log10(cost) = 3.379
Cost = 2393

Question 2

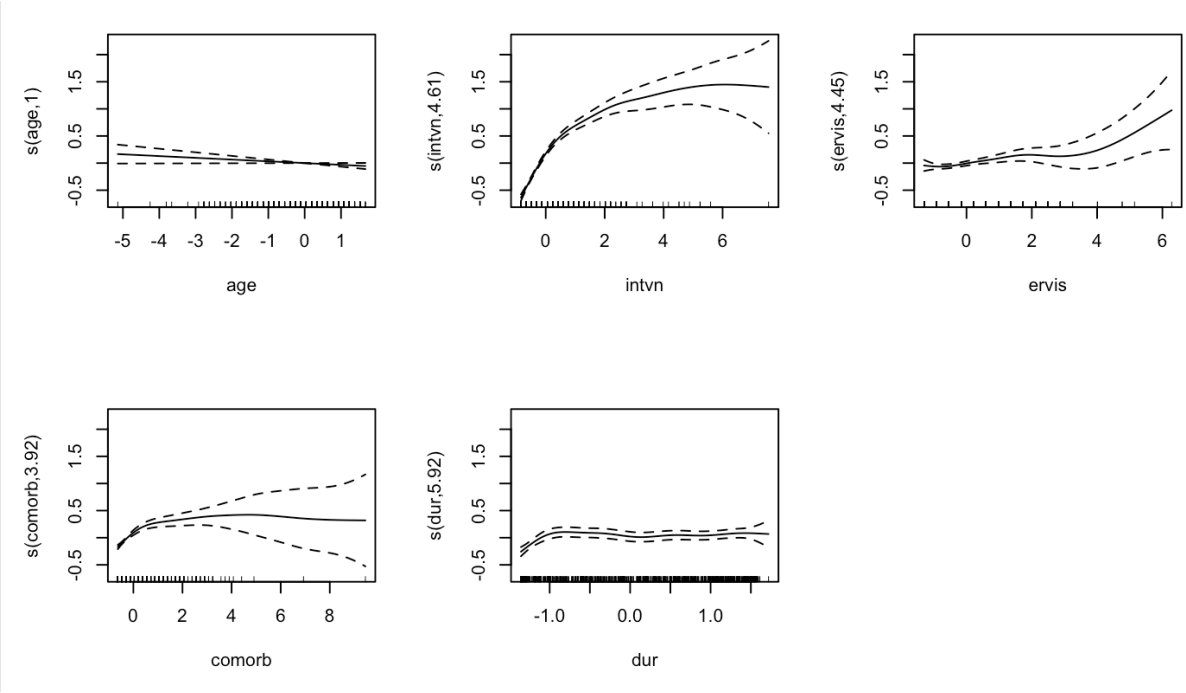
a) Note that since gend, drugs, and comp are discrete with only a very few categories, it is not possible to fit a nonparametric smoother to them, and so the gam() function will give an error. Therefore I have not used them with smoother function. Here is the output and Plot of the GAM Model

```
Formula:
log10(cost) ~ s(age) + gend + s(intvn) + drugs + s(ervis) + comp +
    s(comorb) + s(dur)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.73172    0.01656 164.919  < 2e-16 ***
gend        -0.02975    0.01685  -1.766   0.0777 .
drugs       -0.02904    0.02074  -1.400   0.1619
comp         0.07184    0.01729   4.154 3.63e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
            edf Ref.df       F  p-value
s(age)    1.000  1.000   3.617  0.05757 .
s(intvn)  4.610  5.563 136.168  < 2e-16 ***
s(ervis)  4.450  5.435   3.101  0.00609 **
s(comorb) 3.924  4.809  17.023  < 2e-16 ***
s(dur)    5.917  7.043   5.514 3.52e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.685   Deviance explained = 69.4%
GCV = 0.22296  Scale est. = 0.2162    n = 788
```



From the model output, we can observe from the p-values that intvn, comorb, dur and comp are significant. The plot is also consistent with showing intvn, comorb, dur and comp being the most important predictors.

b) For the model above
CV SSE = 177.5628
CV Estimate of Standard Deviation = sqrt(177.5628/n) = 0.4746

Pros and cons of using n-fold CV, versus say 10-fold CV, for GAMs

When discussing the trade-offs between n-fold and K-fold cross-validation in the context of computational modeling, excluding specific methods like nearest neighbors and local kernel weighting, it's evident that n-fold cross-validation (often leave-one-out cross-validation when n is the sample size) can be significantly more computationally intensive than K-fold, particularly when K is less than n. This is because n-fold requires the model to be trained on nearly the entire dataset n times, which, for complex models like Generalized Additive Models (GAMs), translates into a considerable amount of computational resources and time. Although n-fold cross-validation is lauded for its precise estimation of test error variance, thus potentially offering a more accurate assessment of model performance, the marginal gains in accuracy might not always be sufficient to outweigh the substantial increase in computational

burden. In practical settings, especially when dealing with large datasets or under computational constraints, 10-fold cross-validation offers a more pragmatic approach. It provides a balance by delivering reliable error estimates and maintaining a more manageable computational load, thus often being the preferred method in applied predictive modeling.

c) For a person with age=59, gend=0, intvn=10, drugs=0,ervis=3, comp=0, comorb=4, and dur=300

   Log10(predicted cost) = 3.5564
   Predicted cost = 10 ^ 3.5564 = 3600.80

Question 3

a) I used 5 replicates of 10-fold cross CV. The best combination of parameters is **span = 0.6 and degree = 1 and MSEcv = 0.2459 and SSEcv=194**. This model has the best MSEcv among all the models I have fit.
b) Using Cp Method, the best set of parameters found is span=0.1 and degree=0.1. This closely aligns with what we have got using 10-fold CV
c) CV Estimate of Prediction Error is = 0.4958
d) for a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300
   log10(predicted cost) = 3.645
   predicted cost = 10 ^ 3.645 = 4415

Question 4

a) Below is the comparison of cross validation SSE for 20 replicates of 10-fold CV for PPR using different values of nterms

nterms	**CV SSE**
1	172
2	173
3	176
30	244

Best Number of nterms = 1 as it has lowest SSE = 172

b) From the above table, lowest SSE comes with nterms = 1. So, fitted a best model with nterms=1

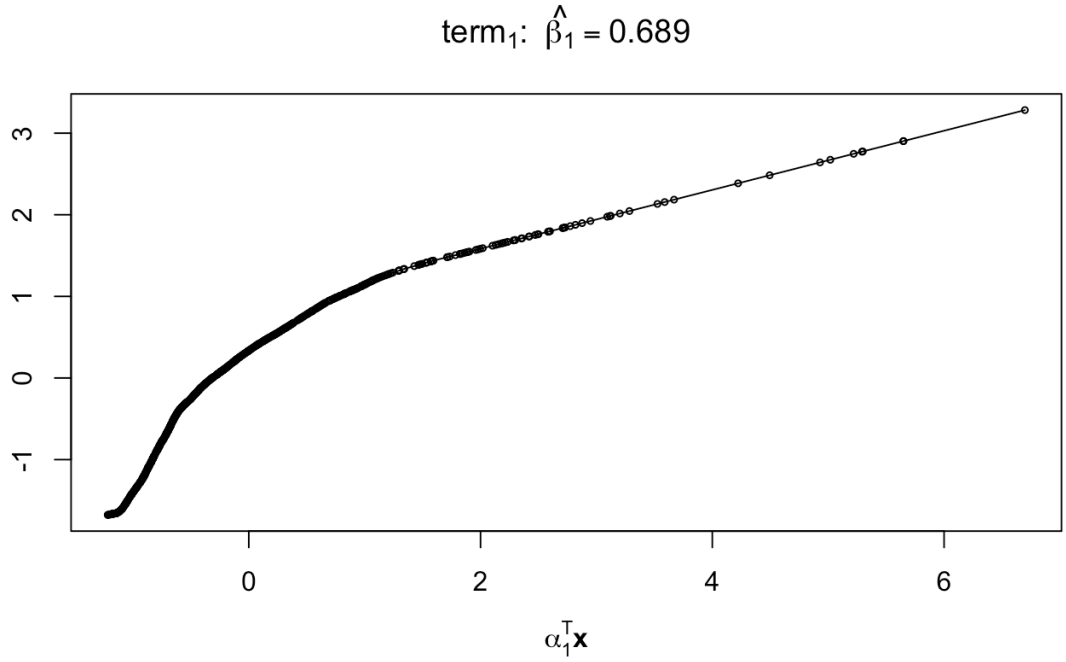
Here is the model Summary Output and output plot.

```
Call:
ppr(formula = log10(cost) ~ ., data = data4, nterms = 1)

Goodness of fit:
 1 terms
165.5263

Projection direction vectors ('alpha'):
        age        gend       intvn       drugs       ervis        comp      comorb         dur
-0.02759078 -0.02686806  0.93275393 -0.07100287  0.10038411  0.15520531  0.28667863  0.08423339

Coefficients of ridge terms ('beta'):
   term 1
0.6888639
```



CV SSE = 172
CV Estimate of Prediction Error = sqrt(172/n) where n=788 => **0.467**

The above shows the projection direction coefficients (the b's from class), and the shape of the nonparametric component function. The largest direction coefficients are for intvn, comorb, ervis, and dur, which agrees with the previous results

c) For a person with age=59, gend=0, intvn=10, drugs=0,ervis=3, comp=0, comorb=4, and dur=300

Log10(predicted cost) = 3.510637
Predicted cost = 10 ^ 3.510637 = 3240.68

Question 5

a) I have used n-fold CV with 5 replicas to find the best K.

K	Misclass Rate (CV)
2	0.1700
4	0.1607
6	0.1644
7	0.1495
8	0.1654
10	0.6822

From the above table, we can see that K=7 is the best with MisclassCv = 14.95 %

b) On applying 10-fold CV with 5 replicates to find the **$MisClass_{cv}$** of GAM which is equal to **16.07%** (since question does not specify we must use n-fold CV, hence applied 10-fold CV for computational efficiency)

c) After doing a 10-fold CV with 5 replicates on 9 models and comparing the cross validation misclassification rate for all our models we can see the neural network with the combination of **10 hidden nodes** or **10 hidden nodes** and **decay of 1** performs with best with **$MisClass_{cv}$** of **15.51%.**

Question 6

a) Below is the 10 fold CV estimated SSE table for various combinations of n.trees, shrinkage and n.depth

n.trees	Shrinkage	n.depth	CV SSE
1000	0.1	3	168
1000	0.05	3	170
1000	**0.02**	**3**	**167**
1000	0.01	3	169
2000	0.005	3	169

1000	0.02	2	169
1000	0.02	4	168

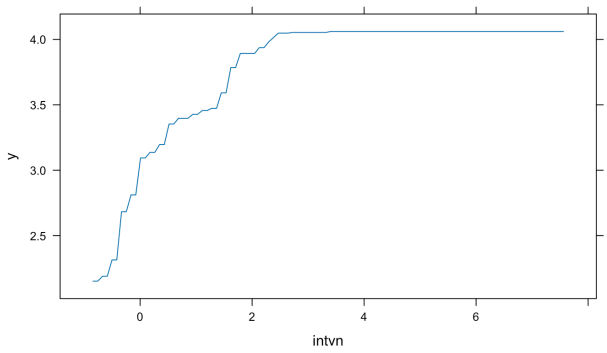
From the table, the best choice of parameters having lowest SSE of 167 is n.trees = 1000, shrinkage = 0.02 and n.depth = 3

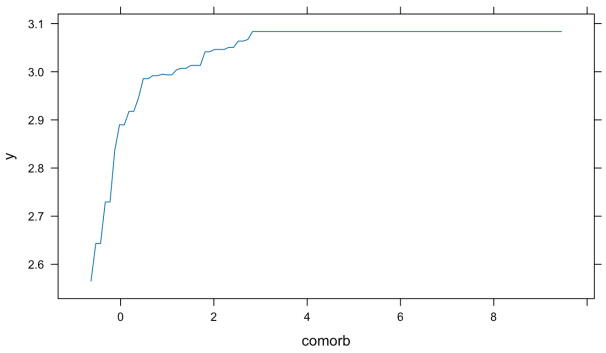
b) Below is the relative importance table of the different variables

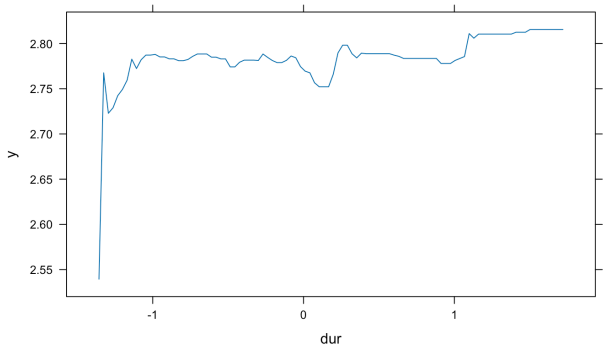
	var <chr>	rel.inf <dbl>
intvn	intvn	71.9423646
comorb	comorb	10.7646449
dur	dur	9.7811147
ervis	ervis	3.5434270
comp	comp	1.5064888
age	age	1.4086645
drugs	drugs	0.5760261
gend	gend	0.4772694

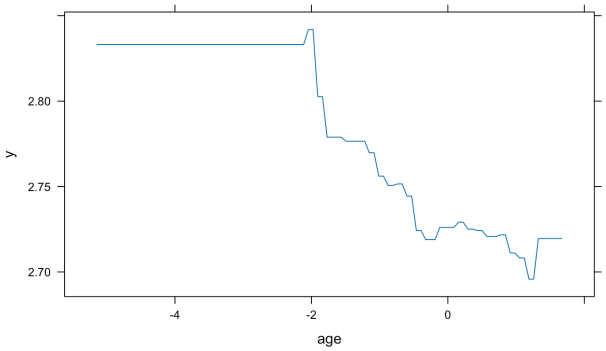
Based on the Relative importance table , the variable exerting the greatest influence is the count of interventions, with the comorbidity index, which reflects the quantity of additional ailments a patient possesses, and the length of treatment trailing closely behind as significant factors. In contrast, the frequency of emergency room encounters, the incidence of complications during treatment, and the patient's age hold less sway over outcomes. The quantity of medications administered and the patient's sex appear to have a negligible impact on costs. These findings are consistent with those obtained through alternative methodologies.

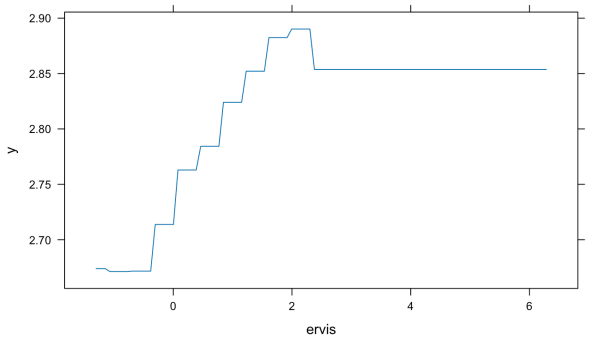
The below marginal plots also shows that intvn, comorb and dur have the largest y-range which is directly proportional to importance. Ervis, Comp and Age have comparatively lesser range than the intvn, comorb and dur.

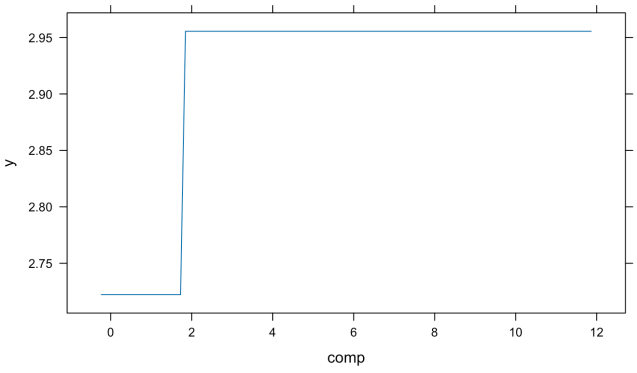












Conclusion: The variable importance measures and marginal plots largely agree with the results of the other methods

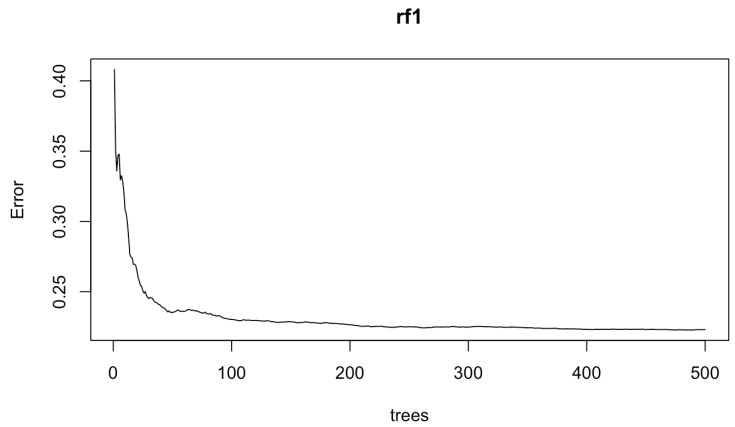
c) For a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300

Log10(predicted cost) = 3.50079
Predicted cost = 10 ^ 3.50079 = 3168.03

Question 7

a) I took 10 replicas of the randomForest training and the typical OOB R2 values observed are 0.6702529 0.6722604 0.6694926 0.6703733 0.6703342 0.674993 0.6722597 0.6692238 0.6687674 0.6742285.
The average OOB R2 value is **0.6712.**
Similarly, average OOB MSE Value is **0.225**

b) Here is the plot CV Error vs Number of Trees

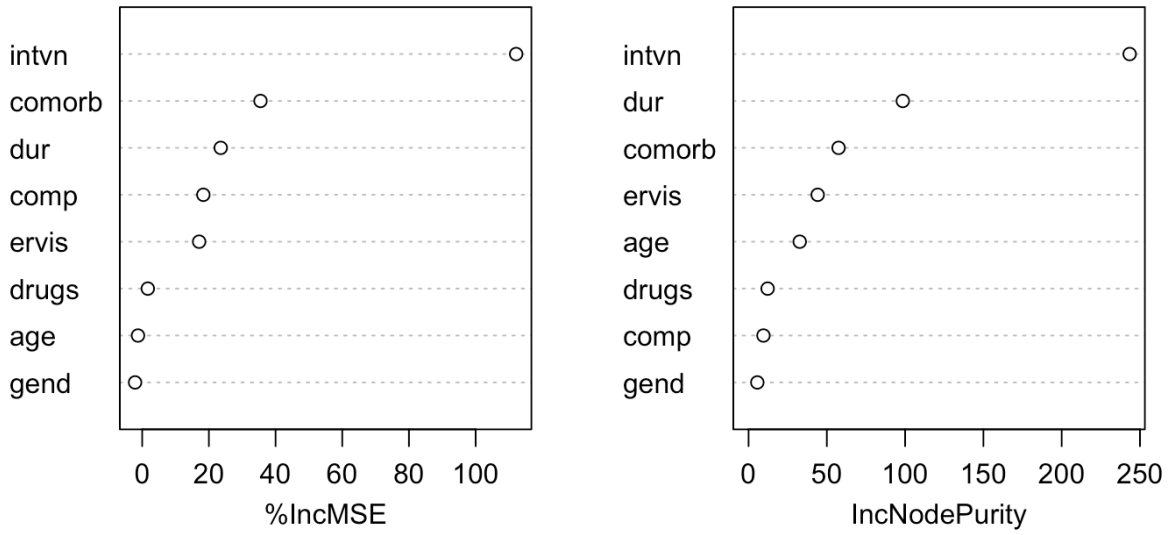


Yes, in my opinion, ntree = 500 is large enough, as we can see from the plot, the OOB CV MSE has converged to a constant value well before the number of trees has reached to 500.

c) Below is the variable importance table and plot

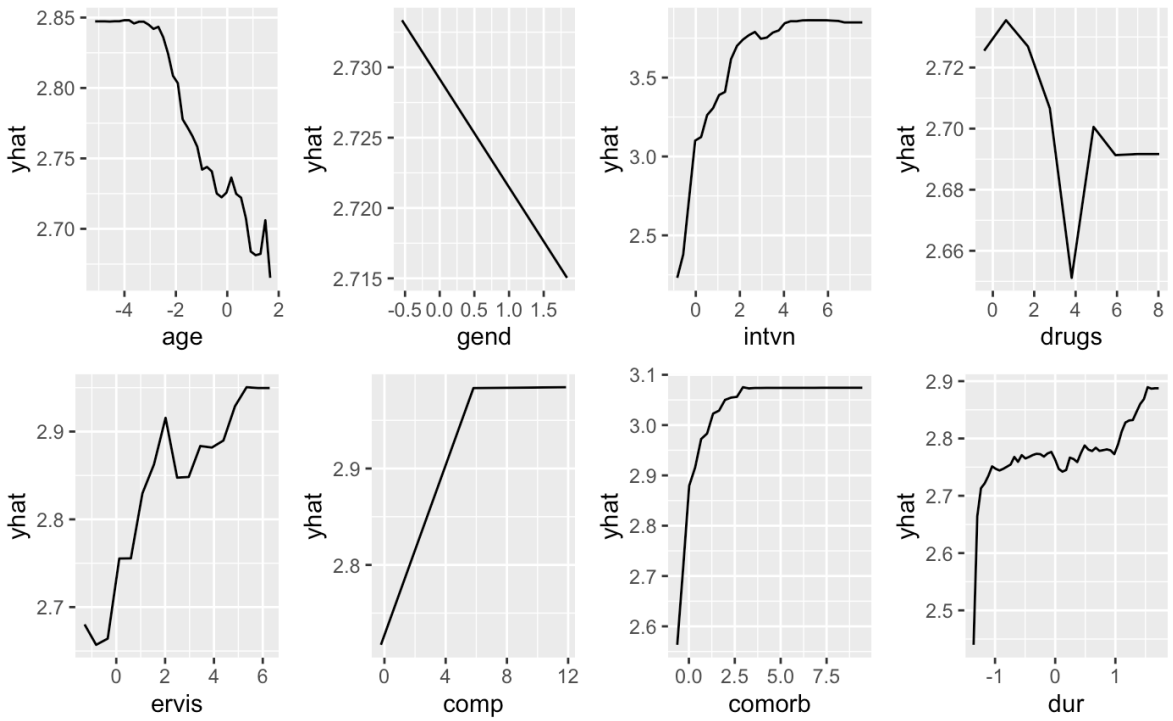
	%IncMSE	IncNodePurity
age	-1.258711	32.670953
gend	-2.130723	5.587874
intvn	112.147301	243.394535
drugs	1.684757	12.174843
ervis	17.079696	44.145740
comp	18.340100	9.500543
comorb	35.476984	57.484795
dur	23.555269	98.522281

rf1



As we can see from the table above, by both importance measures, intvn has the highest importance followed by the dur and comorb (order reversed). The boosted tree also had intvn by far the most important, followed by comorb and dur.

d) Below are the partial plots of all the variables



Based on the plot, intvn, comorb and dur are having very non-linear effects. They have monotonically positive effects, as they are increasing sharply as the predictor value increases and then plateauing. Overall, we can say that the effect is pretty like that of boosted trees.

e) For x1 = 3.0, x2 = 28, x3 = 1.0
Predicted Response = 3.58

f) Here is the OOB R2 Values for different mtry

Mtry	OOB R2
1	56.35 %
2	67.12 %
3	67.12 %

We can see that mtry=1 is worst and mtry=2 and mtry=3 are almost similar. So, we can take mtry=3 to be the best model.

The **mtry** parameter in random forest algorithms is crucial for defining how many predictor variables are considered for splitting at each node of the trees. It serves to inject randomness, helping to prevent overfitting by ensuring trees are less

correlated and that dominant variables don't overshadow others. Finding the right mtry value is a balance act; too low, and the model may miss key information, too high, and it risks overfitting. The ideal **mtry** is typically found through model tuning techniques like cross-validation, impacting the model's bias-variance trade-off and overall effectiveness.

g) Here is the Comparison of different models by comparing CV SSE

Model	**CV SSE**
Tree	198
Neural Net	172
KNN	223
GAM	178
Local Linear Regression	194
PPR	172
Linear Regression	233
Boosted Trees	167
Random Forest	178

From this table, we can conclude that boosted trees have worked best on the given data followed by PPR, Neural Net, GAM and RandomForest.

PA2_HW3

Ayush Agarwal

2024-02-29

Question 1

```
set.seed(1234)
data <- read_excel('HW3_data.xls')
data  <- data[,-1]
data1 <- data
# standardize the predictors
data1[2:9] <- sapply(data1[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
}
data1
```

Part a)

You can also embed plots, for example:

```
set.seed(1234)
####CV to choose the best K
Nrep<-1 #number of replicates of CV
n=nrow(data1)
K<-n #K-fold CV on each replicate
n.models = 6 #number of different models to fit
y<-log10(data1$cost)
yhat=matrix(0,n,n.models)
MSE<-matrix(0,Nrep,n.models)
SSE<-matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
```

```
  Ind<-CVInd(n,K)
  for (k in 1:K) {
    train<-as.matrix(data1[-Ind[[k]],2:9])
    test<-as.matrix(data1[Ind[[k]],2:9])
    ytrain<-y[-Ind[[k]]]
    K1=1;K2=5;K3=8;K4=10;K5=15;K6=20

    # K1
    out<-ann(train,test,K1,verbose=F)
    ind<-as.matrix(out$knnIndexDist[,1:K1])
    yhat[Ind[[k]],1]<-apply(t(ind),1,function(x) mean(ytrain[x]))

    # K2
    out<-ann(train,test,K2,verbose=F)
    ind<-as.matrix(out$knnIndexDist[,1:K2])
    yhat[Ind[[k]],2]<-apply(t(ind),1,function(x) mean(ytrain[x]))

    # K3
    out<-ann(train,test,K3,verbose=F)
    ind<-as.matrix(out$knnIndexDist[,1:K3])
    yhat[Ind[[k]],3]<-apply(t(ind),1,function(x) mean(ytrain[x]))

    # k4
    out<-ann(train,test,K4,verbose=F)
    ind<-as.matrix(out$knnIndexDist[,1:K4])
    yhat[Ind[[k]],4]<-apply(t(ind),1,function(x) mean(ytrain[x]))

    # k5
    out<-ann(train,test,K5,verbose=F)
    ind<-as.matrix(out$knnIndexDist[,1:K5])
    yhat[Ind[[k]],5]<-apply(t(ind),1,function(x) mean(ytrain[x]))

    # k6
    out<-ann(train,test,K6,verbose=F)
    ind<-as.matrix(out$knnIndexDist[,1:K6])
    yhat[Ind[[k]],6]<-apply(t(ind),1,function(x) mean(ytrain[x]))

  } #end of k loop
  MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2)) / n
  SSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))
} #end of j loop
MSEAve<- apply(MSE,2,mean); MSEAve #averaged mean square CV error
SSEAve<- apply(SSE,2,mean); SSEAve #averaged square CV error
MSEsd <- apply(MSE,2,sd); MSEsd   #SD of mean square CV error
r2<-1-MSEAve/var(y); r2  #CV r^2
plot(yhat[,2],y)
```

Part b)

```
set.seed(1234)
####CV to choose the best K
```

```
Nrep<-1 #number of replicates of CV
n=nrow(data1)
K<-n #K-fold CV on each replicate
n.models = 1 #number of different models to fit
y<-log10(data1$cost)
yhat=matrix(0,n,n.models)
MSE<-matrix(0,Nrep,n.models)
SSE<-matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
  Ind<-CVInd(n,K)
  for (k in 1:K) {
    train<-as.matrix(data1[-Ind[[k]],2:9])
    test<-as.matrix(data1[Ind[[k]],2:9])
    ytrain<-y[-Ind[[k]]]
    K1=8

    # K1
    out<-ann(train,test,K1,verbose=F)
    ind<-as.matrix(out$knnIndexDist[,1:K1])
    yhat[Ind[[k]],1]<-apply(t(ind),1,function(x) mean(ytrain[x]))

  } #end of k loop
  MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2)) / n
  SSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))
} #end of j loop
MSEAve<- apply(MSE,2,mean)
SSEAve<- apply(SSE,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
plot(yhat[,1],y)
standard_error<-sd(y-yhat)
cat('Cross Validation MSE:->', MSEAve,'\n')
cat('Cross Validation SSE:->', SSEAve,'\n')
cat('Cross Validation R2:->', r2,'\n')
cat('Standard Error:->',standard_error,'\n')
```

Part c)

```
set.seed(1234)
# Testing on new Data Point
test<-c(59, 0, 10, 0, 3, 0, 4, 300)
xbar<-apply(as.matrix(data[,2:9]),2,mean)
std<-apply(as.matrix(data[,2:9]),2,sd)
test<-(test-xbar)/std #must standardize

test<-matrix(test,1,8)
K<-8
out<-ann(as.matrix(data1[,2:9]),test,K,verbose=F)
ind<-matrix(out$knnIndexDist[,1:K],nrow(test),K)
cat("Nearest Neighbour Indices:->",ind,'\n')
```

```
yhat<-mean(log10(data1$cost[ind]))
cat("Predicted Yhat:->",yhat,'\n')
```

Question 2

```
data2 <- data
# standardize the predictors
data2[2:9] <- sapply(data2[2:9], function(x) ((x - mean(x)) / sd(x)))
head(data2)
```

Part a)

```
set.seed(1234)
out<-gam(log10(cost)~ s(age) + gend + s(intvn) + drugs + s(ervis) + comp + s(comorb) + s(dur), data=data
summary(out)
par(mfrow=c(2,3))
plot(out) #plot component functions
```

Part B

```
set.seed(1234)
Nrep<-1 #number of replicates of CV
K<-nrow(data2) #K-fold CV on each replicate
n=nrow(data2)
y<-log10(data2$cost)
SSE<-matrix(0,Nrep,1)
for (j in 1:Nrep) {
  Ind<-CVInd(n,K)
  yhat1<-y;
  yhat2<-y;
  for (k in 1:K) {
  out<- gam(log10(cost)~ s(age) + gend + s(intvn) + drugs + s(ervis) + comp + s(comorb) + s(dur), data=d
  yhat1[Ind[[k]]]<-as.numeric(predict(out,data2[Ind[[k]],]))
  } #end of k loop
  SSE[j,]=sum((y-yhat1)^2)
  } #end of j loop
RMSE <- sqrt(SSE/n)
SSE
RMSE
```

```
set.seed(1234)
out<-gam(log10(cost)~ s(age) + gend + s(intvn) + drugs + s(ervis) + comp + s(comorb) + s(dur), data=data
test<-c(59, 0, 10, 0, 3, 0, 4, 300)
xbar<-apply(as.matrix(data[,2:9]),2,mean)
std<-apply(as.matrix(data[,2:9]),2,sd)
test<-(test-xbar)/std #must standardize
```

```r
X<-data.frame(matrix(test,1,8))
names(X)<-names(data2[2:9])
yhat<-predict(out,newdata=X)
yhat
```

Question 3

```r
data3 <- data
# standardize the predictors
data3[2:9] <- sapply(data3[2:9], function(x) ((x - mean(x)) / sd(x)))
head(data3)
```

```r
set.seed(1234)
Nrep <- 5 # Number of replicates of CV
K <- 10  # n-fold CV on each replicate
n <- nrow(data3)

#Hyperparameter values
lambda_seq <- seq(0.1, 1, 0.1) # Sequence of lambda values
degrees <- c(0, 1, 2) # Degrees to test

#Intialise a dataframe to store the results
results <- expand.grid(lambda = lambda_seq, degree = degrees, Replicate = 1:Nrep, MSE = NA_real_) # Pre

for (j in 1:Nrep) {
  Ind <- CVInd(n, K)
  for (lambda in lambda_seq) {
    for (degree in degrees) {
      cv_mse <- numeric(K)
      for (k in 1:K) {
        train <- data3[-Ind[[k]], ]
        test <- data3[Ind[[k]], ]

        # Fit the model
        set.seed(1234)
        out <- loess(log10(cost) ~ intvn + comorb + dur + ervis, data = train, degree = degree, span = 

        # Predict and calculate MSE
        yhat <- predict(out, newdata = test)
        cv_mse[k] <- mean((log10(test$cost) - yhat)^2)
      }

      # Store the average MSE across all folds for this lambda and degree
      results$MSE[results$lambda == lambda & results$degree == degree & results$Replicate == j] <- mean
    }
  }
}

#Aggregate the results to find the average MSE across all replicates for each lambda and degree combina
overall_MSE <- aggregate(MSE ~ lambda + degree, data = results, mean)
```

```r
# Identify the combination with the lowest average MSE
best_combination <- overall_MSE[which.min(overall_MSE$MSE), ]
best_combination
```

Part b

```r
#b.Using Cp method to find the optimum value of span and degree
# Initialize a data frame to store lambda, degree, and sigma
results_sigma <- data.frame(lambda = numeric(), degree = integer(), sigma = numeric())

# Loop through lambda and degree combinations
for (lambda in lambda_seq) {
  for (degree in degrees) {
    # Fit the model
    out <- loess(log10(cost) ~ intvn + comorb + dur + ervis, data = data3, degree = degree, span = lambd
    results_sigma <- rbind(results_sigma, data.frame(lambda = lambda, degree = degree, sigma = out$s))
  }
}

# Find the row with the minimum sigma value
best_combination1 <- results_sigma[which.min(results_sigma$sigma), ]
best_combination1
```

Part c

```r
RMSEcv = sqrt(best_combination$MSE)
RMSEcv
```

Part d

```r
# Predicting on new data
test<-c(59, 0, 10, 0, 3, 0, 4, 300)
xbar<-apply(as.matrix(data[,2:9]),2,mean)
std<-apply(as.matrix(data[,2:9]),2,sd)
test<-(test-xbar)/std #must standardize
X<-data.frame(matrix(test,1,8))
names(X)<-names(data[2:9])

#Fitting the optimum model
fit2 <- loess(log10(cost) ~ intvn + comorb + dur + ervis, data = data3, degree = 1, span = 0.1)
yhat = as.numeric(predict(fit2,newdata = X));yhat
```

Question 4

```
data4 <- data
data4[2:9] <- sapply(data4[2:9], function(x) ((x - mean(x)) / sd(x)))
head(data4)
```

```
set.seed(1234)
Nrep<-20 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 4 #number of different models to fit
n=nrow(data4)
y<-log10(data4$cost)
yhat=matrix(0,n,n.models)
MSE<-matrix(0,Nrep,n.models)
SSE<-matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
  Ind<-CVInd(n,K)
  for (k in 1:K) {
    out<- ppr(log10(cost) ~., data=data4[-Ind[[k]],], nterms=1)
    yhat[Ind[[k]],1]<-as.numeric(predict(out,data4[Ind[[k]],]))

    out<- ppr(log10(cost) ~., data=data4[-Ind[[k]],], nterms=2)
    yhat[Ind[[k]],2]<-as.numeric(predict(out,data4[Ind[[k]],]))

    out<- ppr(log10(cost) ~., data=data4[-Ind[[k]],], nterms=3)
    yhat[Ind[[k]],3]<-as.numeric(predict(out,data4[Ind[[k]],]))

    out<- ppr(log10(cost) ~., data=data4[-Ind[[k]],], nterms=40)
    yhat[Ind[[k]],4]<-as.numeric(predict(out,data4[Ind[[k]],]))
  } #end of k loop
  MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n
  SSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))
} #end of j loop
MSEAve<- apply(MSE,2,mean); MSEAve #averaged mean square CV error
SSEAve<- apply(SSE,2,mean); SSEAve #averaged mean square CV error
RMSEAve<- sqrt(MSEAve); RMSEAve #averaged root 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 B

```
out<-ppr(log10(cost)~., data=data4, nterms=1)
summary(out)
par(mfrow=c(1,1))
plot(out)
```

Part c

```
test<-c(59, 0, 10, 0, 3, 0, 4, 300)
xbar<-apply(as.matrix(data[,2:9]),2,mean)
```

```
std<-apply(as.matrix(data[,2:9]),2,sd)
test<-(test-xbar)/std #must standardize
X<-data.frame(matrix(test,1,8))
names(X)<-names(data4[2:9])
yhat<-predict(out,newdata=X)
yhat
```

Question 5

```
data5 <- read.delim("fgl.txt")

#Inspecting the data
head(data5)

#Making it a 2-class problem
z<-(data5$type == "WinF") | (data5$type == "WinNF")
y<-as.character(data5$type)
y[z]<-"Win"; y[!z]<-"Other"
data5<-data.frame(data5,"type_bin"=as.factor(y))  #add a binary factor response column
y[y == "Win"]<-1;y[y == "Other"]<-0;
data5<-data.frame(data5,"type01"=as.numeric(y))  #also add a binary numeric response column

#Selecting the required columns
data5 = data5 %>%
  dplyr::select(-c(type, type_bin)) %>%
  mutate(type01 = as.factor(type01))

#Standardizing the predictor variables and making sure type is a factor
data5_final = data5 %>%
  mutate(across(
    .cols = setdiff(names(.), c("type01")),
    .fns = ~ ( . - mean(., na.rm = TRUE)) / sd(., na.rm = TRUE)
  ))
```

Part a)

```
#Using n-fold CV to find the optimum value of K
Nrep<-5 #number of replicates of CV
n=nrow(data5_final)
K<-n  #n-fold CV on each replicate
n.models = 6 #number of different models to fit
y<-data5_final$type01

yhat=matrix(0,n,n.models)
CV_Misclass <-matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
  Ind<-CVInd(n,K)
  for (k in 1:K) {
    train<-as.matrix(data5_final[-Ind[[k]],1:9])
```

```
    test<-as.matrix(data5_final[Ind[[k]],1:9])
    ytrain <- as.vector(data5_final[-Ind[[k]], ][[10]])

    K1=2;K2=4;K3=6;K4=7;K5=8;K6=10; #different K's to try
    out<-knn(train, test, ytrain, k = K1, prob = T)
    yhat[Ind[[k]],1]<-as.numeric(as.character(out))

    out<-knn(train, test, ytrain, k = K2, prob = T)
    yhat[Ind[[k]],2]<-as.numeric(as.character(out))

    out<-knn(train, test, ytrain, k = K3, prob = T)
    yhat[Ind[[k]],3]<-as.numeric(as.character(out))

    out<-knn(train, test, ytrain, k = K4, prob = T)
    yhat[Ind[[k]],4]<-as.numeric(as.character(out))

    out<-knn(train, test, ytrain, k = K5, prob = T)
    yhat[Ind[[k]],5]<-as.numeric(as.character(out))

    out<-knn(train, test, ytrain, k = K6, prob = T)
    yhat[Ind[[k]],5]<-as.numeric(as.character(out))
  } #end of k loop
  CV_Misclass[j,]=apply(yhat,2,function(x) sum(y != x)/n)
} #end of j loop
CV_Misclass_Ave<- apply(CV_Misclass,2,mean); CV_Misclass_Ave #averaged CV misclass rate
CV_Misclass_SD <- apply(CV_MSE,2,sd); CV_Misclass_SD   #SD of CV misclass rate
```

Part b)

```
set.seed(1234)
#Using n-fold CV to find the CV Misclass Rate
Nrep<-5 #number of replicates of CV
n=nrow(data5_final)
K<-10  #n-fold CV on each replicate
n.models = 1 #number of different models to fit
y<-data5_final$type01

yhat=matrix(0,n,n.models)
CV_Misclass_Rate<-matrix(0,Nrep,n.models)

for (j in 1:Nrep) {
  Ind<-CVInd(n,K)
  for (k in 1:K) {
    fit4 = gam(type01~s(RI)+s(Na)+s(Mg)+s(Al)+ s(Si) + s(K) + s(Ca)
               + s(Ba)+ s(Fe), data=data5_final[-Ind[[k]],], family=binomial)
    out <- as.numeric(predict(fit4, data5_final[Ind[[k]],1:9],type = 'response'))
    yhat[Ind[[k]]] = ifelse(out > 0.5, 1, 0)
  } #end of k loop
  CV_Misclass[j,]=apply(yhat,2,function(x) sum(y != x)/n)
} #end of j loop
CV_MisclassAve<- apply(CV_Misclass,2,mean); CV_MisclassAve #averaged CV misclass rate
```

```
CV_MisclassSD <- apply(CV.rate,2,sd); CV_MisclassSD   #SD of CV misclass rate
```

```
# Fitting 9 separate neural networks to find the optimum parameters values
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 9 #number of different models to fit
n=nrow(data5_final)
y<-data5_final$type01
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<-nnet(type01~.,data5_final[-Ind[[k]],],linout=F,skip=F,size=25,decay=.01, maxit=1000,trace=F)
    yhat[Ind[[k]],1]<-predict(out,data5_final[Ind[[k]],],type = 'class')

    out<-nnet(type01~.,data5_final[-Ind[[k]],],linout=F,skip=F,size=25,decay=.1, maxit=1000,trace=F)
    yhat[Ind[[k]],2]<-predict(out,data5_final[Ind[[k]],],type = 'class')

    out<-nnet(type01~.,data5_final[-Ind[[k]],],linout=F,skip=F,size=25,decay=1, maxit=1000,trace=F)
    yhat[Ind[[k]],3]<-predict(out,data5_final[Ind[[k]],],type = 'class')

    out<-nnet(type01~.,data5_final[-Ind[[k]],],linout=F,skip=F,size=15,decay=.01, maxit=1000,trace=F)
    yhat[Ind[[k]],4]<-predict(out,data5_final[Ind[[k]],],type = 'class')

    out<-nnet(type01~.,data5_final[-Ind[[k]],],linout=F,skip=F,size=15,decay=.1, maxit=1000,trace=F)
    yhat[Ind[[k]],5]<-predict(out,data5_final[Ind[[k]],],type = 'class')

    out<-nnet(type01~.,data5_final[-Ind[[k]],],linout=F,skip=F,size=15,decay=1, maxit=1000,trace=F)
    yhat[Ind[[k]],6]<-predict(out,data5_final[Ind[[k]],],type = 'class')

    out<-nnet(type01~.,data5_final[-Ind[[k]],],linout=F,skip=F,size=10,decay=.01, maxit=1000,trace=F)
    yhat[Ind[[k]],7]<-predict(out,data5_final[Ind[[k]],],type = 'class')

    out<-nnet(type01~.,data5_final[-Ind[[k]],],linout=F,skip=F,size=10,decay=.1, maxit=1000,trace=F)
    yhat[Ind[[k]],8]<-predict(out,data5_final[Ind[[k]],],type = 'class')

    out<-nnet(type01~.,data5_final[-Ind[[k]],],linout=F,skip=F,size=10,decay=1, maxit=1000,trace=F)
    yhat[Ind[[k]],9]<-predict(out,data5_final[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.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
```

Question 6

```
data6 <- data
data6[2:9] <- sapply(data6[2:9], function(x) ((x - mean(x)) / sd(x)))
head(data6)
```

Part a

```
set.seed(1234)

combinations <- data.frame(
  n.trees = c(1000, 1000, 1000, 1000, 2000, 1000, 1000),
  shrinkage = c(0.1, 0.05, 0.02, 0.01, 0.005, 0.02, 0.02),
  int.depth = c(3, 3, 3, 3, 3, 2, 4)
)

# Create an empty data frame to store the results
results <- data.frame(n.trees=integer(), shrinkage=numeric(), int.depth=integer(), CV_SSE=numeric())

# Loop over the specified combinations
for(i in 1:nrow(combinations)) {
  # Extract the parameters for the current combination
  ntrees <- combinations$n.trees[i]
  shrinkage <- combinations$shrinkage[i]
  depth <- combinations$int.depth[i]
  set.seed(1234)
  # Fit the GBM model with the current combination of parameters
  gbm_model <- gbm(log10(cost)~., data=data6, var.monotone=rep(0,8), distribution="gaussian",
             n.trees=ntrees, shrinkage=shrinkage, interaction.depth=depth,bag.fraction = .5,
             train.fraction = 1, n.minobsinnode = 10, cv.folds = 10, keep.data=TRUE, verbose=FALSE)

  # Determine the best iteration based on CV
  best.iter <- gbm.perf(gbm_model, method="cv")

  # Calculate the SSECV
  SSECV <- gbm_model$cv.error[best.iter] * nrow(data6)

  # Add the current results to the results data frame
  results <- rbind(results, c(ntrees, shrinkage, depth, SSECV))
}

# Name the columns of the results data frame
colnames(results) <- c("ntrees", "shrinkage", "depth", "CV_SSE")

# Print the results
print(results)
```

```
set.seed(1234)
gbm1 <- gbm(log10(cost)~., data=data6, var.monotone=rep(0,8), distribution="gaussian",
            n.trees=1000, shrinkage=0.02, interaction.depth=3,bag.fraction = .5,
            train.fraction = 1, n.minobsinnode = 10, cv.folds = 10, keep.data=TRUE, verbose=FALSE)
best.iter <- gbm.perf(gbm1,method="cv");best.iter
SSECV<-gbm1$cv.error[best.iter]*nrow(data6);SSECV


summary(gbm1,n.trees=best.iter) # based on the optimal number of trees
```

```r
# Plotting the output
par(mfrow = c(2, 3))
plot(gbm1, i.var = 3, n.trees = best.iter)
plot(gbm1, i.var = 7, n.trees = best.iter)
plot(gbm1, i.var = 8, n.trees = best.iter)


plot(gbm1, i.var = 1, n.trees = best.iter)
plot(gbm1, i.var = 5, n.trees = best.iter)
plot(gbm1, i.var = 6, n.trees = best.iter)
```

```r
# Predicting on new data
test<-c(59, 0, 10, 0, 3, 0, 4, 300)
xbar<-apply(as.matrix(data[,2:9]),2,mean)
std<-apply(as.matrix(data[,2:9]),2,sd)
test<-(test-xbar)/std #must standardize
X<-data.frame(matrix(test,1,8))
names(X)<-names(data[2:9])
yhat<-predict(gbm1,newdata=X,n.trees=best.iter);yhat
```

Question 7

```r
data7 <- data
data7[2:9] <- sapply(data7[2:9], function(x) ((x - mean(x)) / sd(x)))
head(data7)
```

Part a

```r
# Initialize lists to store the OOB R^2 and MSE values
r2_list <- list()
mse_list <- list()

# Loop to run the randomForest model 10 times
for (i in 1:10) {
  set.seed(i) # Set a seed for reproducibility
  rf1 <- randomForest(log10(cost) ~ ., data = data7, mtry = 3, ntree = 500, importance = TRUE)

  # Retrieve and store the final OOB R^2 and MSE
  r2_list[[i]] <- tail(rf1$rsq, 1)
  mse_list[[i]] <- tail(rf1$mse, 1)
}

mean_r2 <- mean(unlist(r2_list))
mean_mse <- mean(unlist(mse_list))

cat("OOB R2:", unlist(r2_list), "\n")
cat("OOB MSE:", unlist(mse_list), "\n")
```

```
cat("Mean OOB R2:", mean_r2, "\n")
cat("Mean OOB MSE:", mean_mse, "\n")
```

Part B

```
# Plotting the rf1 to analyse test error vs num_trees
plot(rf1)
```

Part c)

```
importance(rf1)
varImpPlot(rf1)
```

Part d

```
# Creating PartialPlots
plots <- lapply(1:8, function(i) {
  pdp::partial(rf1, pred.var = names(data7)[i+1], pred.data = data7, plot = TRUE, plot.engine = "ggplot
})

# arrange the plots in a 2 x 4 grid
grid.arrange(grobs = plots, ncol = 4)
```

Part e

```
# Prediction on new data
test<-c(59, 0, 10, 0, 3, 0, 4, 300)
xbar<-apply(as.matrix(data[,2:9]),2,mean)
std<-apply(as.matrix(data[,2:9]),2,sd)
test<-(test-xbar)/std #must standardize
X<-data.frame(matrix(test,1,8))
names(X)<-names(data[2:9])
yhat<-predict(rf1,newdata=X);yhat
```

Part f

```
r2_list_m1 <- list()
mse_list_m1 <- list()

r2_list_m2 <- list()
mse_list_m2 <- list()
```

```r
# mtry 1
# Loop to run the randomForest model 10 times
for (i in 1:10) {
  set.seed(i) # Set a seed for reproducibility
  rf_m1 <- randomForest(log10(cost) ~ ., data = data7, mtry = 1, ntree = 500, importance = TRUE)
  # Retrieve and store the final OOB R^2 and MSE
  r2_list_m1[[i]] <- tail(rf_m1$rsq, 1)
  mse_list_m1[[i]] <- tail(rf_m1$mse, 1)
}
mean_r2_m1 <- mean(unlist(r2_list_m1))
mean_mse_m1 <- mean(unlist(mse_list_m1))

cat("Mean OOB R2 for mtry=1:", mean_r2_m1, "\n")
cat("Mean OOB MSE for mtry=1:", mean_mse_m1, "\n")

# mtry 2
# Loop to run the randomForest model 10 times
for (i in 1:10) {
  set.seed(i+1) # Set a seed for reproducibility
  rf_m2 <- randomForest(log10(cost) ~ ., data = data7, mtry = 2, ntree = 500, importance = TRUE)
  # Retrieve and store the final OOB R^2 and MSE
  r2_list_m2[[i]] <- tail(rf_m2$rsq, 1)
  mse_list_m2[[i]] <- tail(rf_m2$mse, 1)
}
mean_r2_m2 <- mean(unlist(r2_list_m2))
mean_mse_m2 <- mean(unlist(mse_list_m2))

cat("Mean OOB R2 for mtry=2:", mean_r2, "\n")
cat("Mean OOB MSE for mtry=2:", mean_mse, "\n")
```