Assignment 5 – Shivani Bhoite

#This file contains all the questions (in black) and generated output (in blue).
#When the output of a model is too long. The truncated output is pasted in the file.
###First the entire code for the question and sub question is pasted and the entire out for
###that question, and sub question is pasted

#Imports

```
library(caret)
library(gbm)
library('RANN')
#install.packages("klaR")
library('klaR')
#install.packages("ggpubr")
library(ggpubr)
data(scat)
str(scat)
```

#Ouput

```
> library(caret)
> library(gbm)
> library('RANN')
> #install.packages("klaR")
> library('klaR')
> #install.packages("ggpubr")
> library(ggpubr)
> data(scat)
> str(scat)
'data.frame':	110 obs. of  19 variables:
 $ Species  : Factor w/ 3 levels "bobcat","coyote",..: 2 2 1 2 2 2 1 1 1 1 ...
 $ Month    : Factor w/ 9 levels "April","August",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ Year     : int  2012 2012 2012 2012 2012 2012 2012 2012 2012 2012 ...
 $ Site     : Factor w/ 2 levels "ANNU","YOLA": 2 2 2 2 2 2 1 1 1 1 ...
 $ Location : Factor w/ 3 levels "edge","middle",..: 1 1 2 2 1 1 3 3 3 2 ...
 $ Age      : int  5 3 3 5 5 5 1 3 5 5 ...
 $ Number   : int  2 2 2 2 4 3 5 7 2 1 ...
 $ Length   : num  9.5 14 9 8.5 8 9 6 5.5 11 20.5 ...
 $ Diameter : num  25.7 25.4 18.8 18.1 20.7 21.2 15.7 21.9 17.5 18 ...
 $ Taper    : num  41.9 37.1 16.5 24.7 20.1 28.5 8.2 19.3 29.1 21.4 ...
 $ TI       : num  1.63 1.46 0.88 1.36 0.97 1.34 0.52 0.88 1.66 1.19 ...
 $ Mass     : num  15.9 17.6 8.4 7.4 25.4 ...
 $ d13C     : num  -26.9 -29.6 -28.7 -20.1 -23.2 ...
 $ d15N     : num  6.94 9.87 8.52 5.79 7.01 8.28 4.2 3.89 7.34 6.06 ...
```

```
 $ CN      : num  8.5 11.3 8.1 11.5 10.6 9 5.4 5.6 5.8 7.7 ...
 $ ropey    : int  0 0 1 1 0 1 1 0 0 1 ...
 $ segmented: int  0 0 1 0 1 0 1 1 1 1 ...
 $ flat     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ scrape   : int  0 0 1 0 0 0 1 0 0 0 ...
```

```
########## 1 Set the Species column as the target/outcome and convert it to numeric. (5 points)
scat_2<-scat
scat_2$Species<-as.factor(scat_2$Species)
target<-scat_2$Species
target_class<-factor(target)
scat_2$Species<-unclass(scat_2$Species)
#Converted to numeric - printing the data below
print(scat_2)
```

```
Species    Month Year Site Location Age Number Length Diameter Taper   TI  Mass   d13C  d15N
CN ropey segmented flat scrape
1       2  January 2012 YOLA     edge   5     2    9.5     25.7  41.9 1.63 15.89 -26.85  6.94  8.50     0
0    0      0
2       2  January 2012 YOLA     edge   3     2   14.0     25.4  37.1 1.46 17.61 -29.62  9.87 11.30     0
0    0      0
3       1  January 2012 YOLA   middle   3     2    9.0     18.8  16.5 0.88  8.40 -28.73  8.52  8.10     1
1    0      1
4       2  January 2012 YOLA   middle   5     2    8.5     18.1  24.7 1.36  7.40 -20.07  5.79 11.50     1
0    0      0
6       2  January 2012 YOLA     edge   5     4    8.0     20.7  20.1 0.97 25.45 -23.24  7.01 10.60     0
1    0      0
7       2  January 2012 YOLA     edge   5     3    9.0     21.2  28.5 1.34 14.14 -29.00  8.28  9.00     1
0    0      0
8       1  January 2012 ANNU off_edge   1     5    6.0     15.7   8.2 0.52 14.82 -28.06  4.20  5.40     1
1    0      1
9       1  January 2012 ANNU off_edge   3     7    5.5     21.9  19.3 0.88 26.41 -27.60  3.89  5.60     0
1    0      0
10      1  January 2012 ANNU off_edge   5     2   11.0     17.5  29.1 1.66 16.24 -28.64  7.34  5.80
0       1    0      0
13      1  January 2012 ANNU   middle   5     1   20.5     18.0  21.4 1.19 11.22 -27.35  6.06  7.70
1       1    0      0
14      3  January 2012 ANNU   middle   3     1    8.0       NA    NA   NA  2.51 -25.79  7.83 20.50     0
0    1      0
15      3  January 2012 ANNU   middle   1     1    8.0     12.9  14.7 1.14  8.55 -25.71  8.47 18.10     1
0    0      0
16      3  January 2012 ANNU   middle   3     1   12.0       NA    NA   NA 18.14 -25.18 10.10 15.50
0       0    1      0
```

18	3	January	2012	ANNU	middle	3	1	11.5	NA	NA	NA	8.17	-25.73	9.72	18.90	0	0	1	0
19	3	January	2012	ANNU	middle	1	1	8.5	NA	NA	NA	3.43	-26.17	8.07	19.90	0	0	1	0
20	3	January	2012	ANNU	middle	5	1	10.5	12.1	11.9	0.98	3.10	-26.88	6.70	7.00	1	1	0	0
21	1	February	2013	ANNU	edge	5	7	5.0	13.0	37.6	2.89	9.75	-27.92	7.57	5.80	1	1	0	0
23	2	February	2013	ANNU	edge	5	6	6.5	24.0	23.1	0.96	33.00	-27.66	12.88	7.70	1	1	0	0
24	1	February	2013	ANNU	edge	5	4	10.5	15.5	38.2	2.46	12.76	-25.77	3.88	5.70	1	0	0	0
25	1	February	2013	ANNU	off_edge	3	3	11.0	16.5	25.8	1.56	18.75	-28.91	6.36	6.00	1	1	0	0
26	1	February	2013	ANNU	off_edge	5	4	11.5	17.5	18.9	1.08	14.08	-27.30	6.61	6.90	1	1	0	0
28	1	February	2013	ANNU	off_edge	5	5	7.5	18.0	32.1	1.78	21.69	-27.20	9.07	5.80	1	1	0	0
29	1	April	2012	YOLA	middle	4	4	5.5	21.7	18.9	0.87	19.15	-29.12	8.72	6.05	0	1	0	0
30	1	April	2012	YOLA	middle	3	1	9.0	21.6	11.4	0.53	9.74	-29.85	10.32	7.48	0	1	0	0
31	2	April	2012	YOLA	middle	5	2	17.0	22.1	5.0	0.23	24.69	-27.82	7.95	7.30	0	1	0	0
32	2	April	2012	YOLA	middle	3	2	6.0	12.5	21.4	1.71	5.05	-29.52	8.91	7.50	1	0	0	0
33	1	April	2012	YOLA	off_edge	5	2	16.5	18.7	23.0	1.23	10.21	-28.53	5.59	7.84	1	1	0	0
34	1	April	2012	ANNU	off_edge	4	1	16.5	16.5	28.9	1.83	7.11	-27.66	8.06	6.20	1	1	0	0
35	3	April	2012	ANNU	middle	1	1	10.0	NA	NA	NA	5.53	-26.58	8.17	18.90	0	0	1	0
36	3	April	2012	ANNU	off_edge	4	3	13.5	19.1	12.8	0.67	9.23	-28.17	5.88	7.70	0	1	0	0
37	3	April	2012	ANNU	middle	4	2	9.5	18.4	28.8	1.57	4.37	-27.37	4.90	6.60	1	0	0	0
38	3	April	2012	ANNU	off_edge	5	1	12.0	14.7	34.7	2.38	7.97	-26.99	5.87	9.20	1	0	0	0
39	1	April	2012	ANNU	off_edge	3	2	7.0	15.7	20.9	1.33	8.98	-28.17	7.08	6.20	1	1	0	0
40	3	April	2012	ANNU	edge	5	1	10.5	17.7	7.5	0.42	7.14	-25.94	7.56	7.70	0	1	0	0
41	1	April	2012	ANNU	edge	3	6	4.0	20.8	12.4	0.60	25.73	-29.55	8.56	7.00	0	1	0	0

```
42      1    April 2012 ANNU off_edge   4     5    5.0     13.8  18.8 1.36  5.50 -27.72  1.84  6.20     1
1    0     0
43      1     May 2013 ANNU   middle   3     2    9.0     17.7  44.7 2.53  6.46 -28.48  6.29  5.60     1
1    0     0
44      1     May 2013 ANNU     edge   3     3   12.0     22.0  38.9 1.77 23.73 -28.92  5.17  6.20     1
1    0     0
45      1     May 2013 ANNU     edge   4     2   15.0     21.7  53.4 2.46 18.00 -27.25  6.94  5.50     1
1    0     0
46      1     May 2013 ANNU     edge   4     1   11.0     16.5  16.0 0.97 11.25 -27.92  8.38  7.70     1
1    0     0
47      1     May 2013 ANNU off_edge   5     4    6.0     15.9  52.8 3.32 10.60 -28.73  7.74  8.40     0
1    0     0
48      1     May 2013 ANNU     edge   3     4   12.0     21.4  12.0 0.56 19.04 -28.72  6.94  6.80     0
1    0     0
49      1     May 2013 ANNU off_edge   5     3   10.5     22.0  37.2 1.69 18.90 -28.00  6.42  6.90
1       0    0     0
50      1     May 2013 ANNU     edge   3     2    4.5     18.2  37.8 2.08  3.82 -25.76  3.50  7.30     1
0    0     0
51      1     May 2013 ANNU     edge   3     2    7.5     16.1  32.0 1.99  6.70 -26.70  7.52  6.00     1
0    0     0
52      1    June 2012 ANNU     edge   1     1    5.5     19.6  25.0 1.28  1.50 -27.40  8.89  4.90     0
0    0     0
53      3    June 2012 ANNU   middle   2     3    6.5     15.1  37.4 2.48  4.31 -28.47  6.39 10.10     1
0    0     0
55      3    June 2012 ANNU     edge   1     1   11.5     10.3  30.8 2.99  2.23 -28.04  6.51  7.20     1
0    0     0
56      1    June 2012 ANNU   middle   2     5    9.5     21.3  18.3 0.86 16.33 -28.56  7.54  6.70     0
1    0     0
57      1    June 2012 ANNU     edge   5     3   12.0     18.7  30.3 1.62 13.19 -27.72  7.25  5.00     1
1    0     0
58      1    June 2012 ANNU   middle   3     3   10.0     24.1    NA    NA 26.89 -27.15  3.46  5.50     0
1    0     0
59      2    June 2012 ANNU     edge   3     2   12.0     23.1  39.1 1.69 22.59 -22.19 18.00  6.00     1
1    0     0
 [ reached 'max' / getOption("max.print") -- omitted 58 rows ]
```

```
########## 2 Remove the Month, Year, Site, Location features. (5 points)
#Before removing
head(scat)
scat_subset <- subset(scat, select = - c(Month, Year, Site, Location))
#After removing
head(scat_subset)
```

```
> ########## 2 Remove the Month, Year, Site, Location features. (5 points)
> #Before removing
> head(scat)
  Species   Month Year Site Location Age Number Length Diameter Taper   TI  Mass   d13C d15N
CN ropey segmented flat scrape
1  coyote January 2012 YOLA     edge   5     2    9.5     25.7  41.9 1.63 15.89 -26.85 6.94  8.5     0
0    0      0
2  coyote January 2012 YOLA     edge   3     2   14.0     25.4  37.1 1.46 17.61 -29.62 9.87 11.3     0
0    0      0
3  bobcat January 2012 YOLA   middle   3     2    9.0     18.8  16.5 0.88  8.40 -28.73 8.52  8.1     1
1    0      1
4  coyote January 2012 YOLA   middle   5     2    8.5     18.1  24.7 1.36  7.40 -20.07 5.79 11.5     1
0    0      0
6  coyote January 2012 YOLA     edge   5     4    8.0     20.7  20.1 0.97 25.45 -23.24 7.01 10.6     0
1    0      0
7  coyote January 2012 YOLA     edge   5     3    9.0     21.2  28.5 1.34 14.14 -29.00 8.28  9.0     1
0    0      0
> scat_subset <- subset(scat, select = - c(Month, Year, Site, Location))
> #After removing
> head(scat_subset)
  Species Age Number Length Diameter Taper   TI  Mass   d13C d15N   CN ropey segmented flat
scrape
1  coyote   5     2    9.5     25.7  41.9 1.63 15.89 -26.85 6.94  8.5     0         0    0      0
2  coyote   3     2   14.0     25.4  37.1 1.46 17.61 -29.62 9.87 11.3     0         0    0      0
3  bobcat   3     2    9.0     18.8  16.5 0.88  8.40 -28.73 8.52  8.1     1         1    0      1
4  coyote   5     2    8.5     18.1  24.7 1.36  7.40 -20.07 5.79 11.5     1         0    0      0
6  coyote   5     4    8.0     20.7  20.1 0.97 25.45 -23.24 7.01 10.6     0         1    0      0
7  coyote   5     3    9.0     21.2  28.5 1.34 14.14 -29.00 8.28  9.0     1         0    0      0
```

```
########## 3 Check if any values are null. If there are, impute missing values using KNN. (10
points)
sum(is.na(scat_subset))
preProcValues <- preProcess(scat_subset, method = c("knnImpute","center","scale"))
scat_processed <- predict(preProcValues, scat_subset)
sum(is.na(scat_processed))
str(scat_processed)
```

```
> ########## 3 Check if any values are null. If there are, impute missing values using KNN. (10
points)
> sum(is.na(scat_subset))
[1] 47
> preProcValues <- preProcess(scat_subset, method = c("knnImpute","center","scale"))
> scat_processed <- predict(preProcValues, scat_subset)
```

```
> sum(is.na(scat_processed))
[1] 0
> str(scat_processed)
'data.frame':	110 obs. of  15 variables:
 $ Species  : Factor w/ 3 levels "bobcat","coyote",..: 2 2 1 2 2 2 1 1 1 1 ...
 $ Age      : num  1.207 -0.252 -0.252 1.207 1.207 ...
 $ Number   : num  -0.433 -0.433 -0.433 -0.433 0.968 ...
 $ Length   : num  0.0587 1.3679 -0.0867 -0.2322 -0.3777 ...
 $ Diameter : num  1.8396 1.7623 0.0622 -0.1181 0.5516 ...
 $ Taper    : num  0.961 0.642 -0.726 -0.182 -0.487 ...
 $ TI       : num  0.0283 -0.1406 -0.7171 -0.24 -0.6277 ...
 $ Mass     : num  0.388 0.583 -0.458 -0.571 1.469 ...
 $ d13C     : num  0.00468 -1.26856 -0.85947 3.12113 1.66403 ...
 $ d15N     : num  -0.165 0.807 0.359 -0.546 -0.141 ...
 $ CN       : num  0.0276 0.7922 -0.0816 0.8468 0.6011 ...
 $ ropey    : num  -1.131 -1.131 0.876 0.876 -1.131 ...
 $ segmented: num  -1.131 -1.131 0.876 -1.131 0.876 ...
 $ flat     : num  -0.239 -0.239 -0.239 -0.239 -0.239 ...
 $ scrape   : num  -0.217 -0.217 4.562 -0.217 -0.217 ...

########## 4 Converting every categorical variable to numerical (if needed). (5 points)
#Not needed

########## 5 With a seed of 100, 75% training, 25% testing.
########### Build the following models: randomforest, neural net, naive bayes and GBM.
scat_processed$Species<-as.factor(scat_processed$Species)
#print(scat_processed)
#Building Models
#Spliting training set into two parts based on outcome: 75% and 25%
set.seed(100)
index <- createDataPartition(scat_processed$Species, p=0.75, list=FALSE)
str(index)
trainSet <- scat_processed[ index,]
testSet <- scat_processed[-index,]

#feature selection
control <- rfeControl(functions = rfFuncs,method = "repeatedcv", repeats = 3,verbose = FALSE)
outcomeName<-'Species'
predictors<-names(trainSet)[!names(trainSet) %in% outcomeName]
str(predictors)

Species_Pred_Profile <- rfe(trainSet[,predictors], trainSet[,outcomeName],rfeControl = control)
Species_Pred_Profile
#names(getModelInfo())
```

```
#Making models

#1)GBM
#As there are more than 2 categories for prediction in GBM the distribution has to be changed from bernoulli to multinomial
model_gbm<-
train(trainSet[,predictors],trainSet[,outcomeName],method='gbm',distribution='multinomial')

#2)Random Forest
model_rf<-train(trainSet[,predictors],trainSet[,outcomeName],method='rf',importance=T)

#3) Neural Network
model_nnet<-
train(trainSet[,predictors],trainSet[,outcomeName],method='nnet',importance=T)

#4) Naive Bayes
model_nbayes<-train(trainSet[,predictors],trainSet[,outcomeName],method='naive_bayes')


###Model Summarization
#1)GBM
print(model_gbm)

#2)Random Forest
print(model_rf)

#3) Neural Network
print(model_nnet)

#4) Naive Bayes
print(model_nbayes)

###Plot Variable Importance
#GBM
plot(varImp(object=model_gbm),main="GBM - Variable Importance")

#RF
plot(varImp(object=model_rf),main="RF - Variable Importance")


#NNET
#for ploting the variable importance of
```

```
df1<-as.data.frame(varImp(object=model_nnet)$importance)
print(df1)
df2 = data.frame(name = c("d15N
","d13C","Mass","CN","Length","ropey","flat","Diameter","Number","Age","TI","segmented","T
aper","scrape"))
cbinded_df<-cbind(df1,df2)

p<-ggplot(data=cbinded_df, aes(x=name, y=Overall)) +
  geom_bar(stat="identity")+ggtitle('Neural Net - Variable Importance')
nnet_var_imp<-p + coord_flip()
nnet_var_imp

#Naive Bayes
plot(varImp(object=model_nbayes),main="Naive Bayes - Variable Importance")

###Confusion Matrix
#GBM
predictions<-predict.train(object=model_gbm,testSet[,predictors],type="raw")
table(predictions)
confusionMatrix(predictions,testSet[,outcomeName])


#RF
predictions<-predict.train(object=model_rf,testSet[,predictors],type="raw")
table(predictions)
confusionMatrix(predictions,testSet[,outcomeName])


#Neural Network
predictions<-predict.train(object=model_nnet,testSet[,predictors],type="raw")
table(predictions)
confusionMatrix(predictions,testSet[,outcomeName])


#Naive Bayes
predictions<-predict.train(object=model_nbayes,testSet[,predictors],type="raw")
table(predictions)
confusionMatrix(predictions,testSet[,outcomeName])

> #print(scat_processed)
> #Building Models
> #Spliting training set into two parts based on outcome: 75% and 25%
> set.seed(100)
```

```
> index <- createDataPartition(scat_processed$Species, p=0.75, list=FALSE)
> str(index)
 int [1:83, 1] 1 3 4 5 6 7 9 13 14 15 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr "Resample1"
> trainSet <- scat_processed[ index,]
> testSet <- scat_processed[-index,]
> #feature selection
> control <- rfeControl(functions = rfFuncs,method = "repeatedcv", repeats = 3,verbose =
FALSE)
> outcomeName<-'Species'
> predictors<-names(trainSet)[!names(trainSet) %in% outcomeName]
> str(predictors)
 chr [1:14] "Age" "Number" "Length" "Diameter" "Taper" "TI" "Mass" "d13C" "d15N" "CN"
"ropey" "segmented" "flat" "scrape"
> Species_Pred_Profile <- rfe(trainSet[,predictors], trainSet[,outcomeName],rfeControl =
control)
> Species_Pred_Profile

Recursive feature selection

Outer resampling method: Cross-Validated (10 fold, repeated 3 times)

Resampling performance over subset size:

 Variables Accuracy  Kappa AccuracySD KappaSD Selected
        4   0.6987 0.4890     0.1359  0.2359        *
        8   0.6876 0.4588     0.1391  0.2554
       14   0.6709 0.4357     0.1395  0.2462

The top 4 variables (out of 4):
   CN, d13C, d15N, Mass

  7       0.7311           nan     0.1000    0.0098
    8       0.7056           nan     0.1000    0.0024
    9       0.6905           nan     0.1000   -0.0052
   10       0.6704           nan     0.1000    0.0137
   20       0.5275           nan     0.1000   -0.0064
   40       0.3920           nan     0.1000   -0.0402
   50       0.3442           nan     0.1000   -0.0192

Warning messages:
1: In (function (x, y, offset = NULL, misc = NULL, distribution = "bernoulli",  :
```

```
  variable 14: scrape has no variation.
2: In (function (x, y, offset = NULL, misc = NULL, distribution = "bernoulli",  :
  variable 14: scrape has no variation.
3: In (function (x, y, offset = NULL, misc = NULL, distribution = "bernoulli",  :
  variable 14: scrape has no variation.
> #2)Random Forest
> model_rf<-train(trainSet[,predictors],trainSet[,outcomeName],method='rf',importance=T)

# weights:  93
initial  value 94.937508
iter  10 value 12.945597
iter  20 value 0.779583
iter  30 value 0.299806
iter  40 value 0.254957
iter  50 value 0.226537
iter  60 value 0.204354
iter  70 value 0.187119
iter  80 value 0.169009
iter  90 value 0.149862
iter 100 value 0.142856
final  value 0.142856
stopped after 100 iterations
# weights:  93
initial  value 92.922438
iter  10 value 34.593676
iter  20 value 26.552082
iter  30 value 24.483406
iter  40 value 24.045519
iter  50 value 23.821458
iter  60 value 23.792807
iter  70 value 23.791236
iter  80 value 23.791194
iter  80 value 23.791193
iter  80 value 23.791193
final  value 23.791193
converged
>
> #4) Naive Bayes
> model_nbayes<-train(trainSet[,predictors],trainSet[,outcomeName],method='naive_bayes')
>
> ###Model Summarization
> #1)GBM
> print(model_gbm)
Stochastic Gradient Boosting
```

83 samples
14 predictors
3 classes: 'bobcat', 'coyote', 'gray_fox'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 83, 83, 83, 83, 83, 83, ...
Resampling results across tuning parameters:

interaction.depth	n.trees	Accuracy	Kappa
1	50	0.6387128	0.3914874
1	100	0.6380097	0.3952067
1	150	0.6194960	0.3650139
2	50	0.6357713	0.3944541
2	100	0.6304535	0.3844917
2	150	0.6160130	0.3633467
3	50	0.6341745	0.3942939
3	100	0.6235783	0.3737189
3	150	0.6176815	0.3652989

Tuning parameter 'shrinkage' was held constant at a value of 0.1
Tuning parameter 'n.minobsinnode' was held constant at a value of 10
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were n.trees = 50, interaction.depth = 1, shrinkage = 0.1 and n.minobsinnode = 10.
>> #2)Random Forest
> print(model_rf)
Random Forest

83 samples
14 predictors
3 classes: 'bobcat', 'coyote', 'gray_fox'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 83, 83, 83, 83, 83, 83, ...
Resampling results across tuning parameters:

mtry	Accuracy	Kappa
2	0.6640078	0.4357575
8	0.6596277	0.4418873
14	0.6429827	0.4203673

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
>> #3) Neural Network
> print(model_nnet)
Neural Network

83 samples
14 predictors
3 classes: 'bobcat', 'coyote', 'gray_fox'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 83, 83, 83, 83, 83, 83, ...
Resampling results across tuning parameters:

size	decay	Accuracy	Kappa
1	0e+00	0.5729286	0.2937404
1	1e-04	0.5562225	0.3127517
1	1e-01	0.6215010	0.3787888
3	0e+00	0.6371756	0.4185840
3	1e-04	0.6819096	0.4828334
3	1e-01	0.6945499	0.4963025
5	0e+00	0.6577248	0.4445735
5	1e-04	0.6691592	0.4733851
5	1e-01	0.7042263	0.5123361

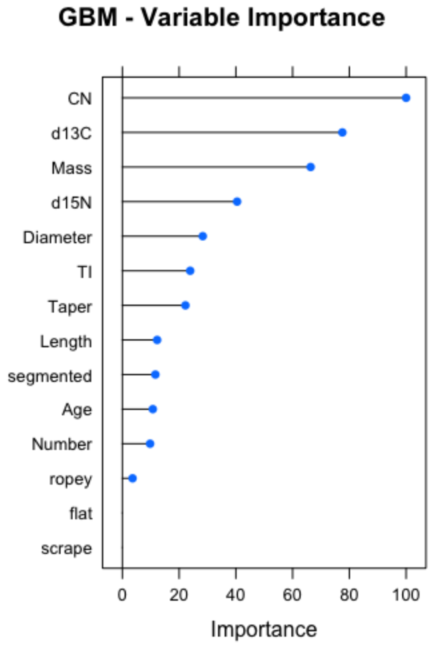
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were size = 5 and decay = 0.1.
> #4) Naive Bayes
> print(model_nbayes)
Naive Bayes

83 samples
14 predictors
3 classes: 'bobcat', 'coyote', 'gray_fox'

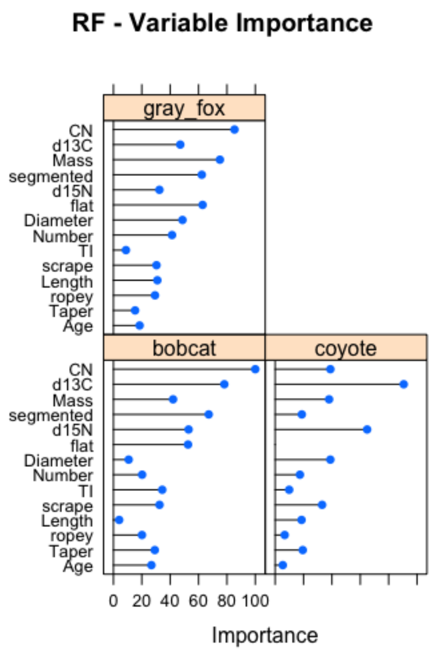
No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 83, 83, 83, 83, 83, 83, ...
Resampling results across tuning parameters:

usekernel	Accuracy	Kappa
FALSE	0.5071348	0.2760894
TRUE	0.6643524	0.4282045

Tuning parameter 'laplace' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were laplace = 0, usekernel = TRUE and adjust = 1.
>###Plot Variable Importance
#GBM
plot(varImp(object=model_gbm),main="GBM - Variable Importance")



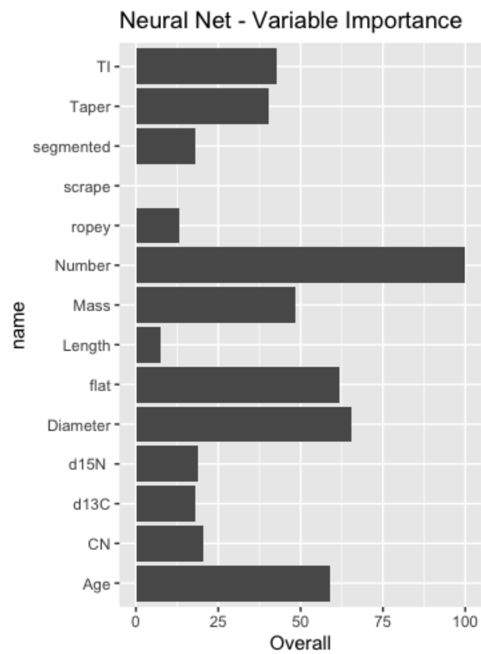
> #RF
> plot(varImp(object=model_rf),main="RF - Variable Importance")



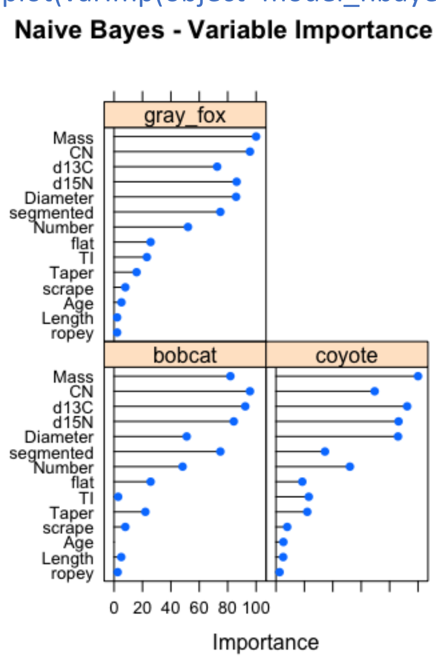
>

```
> df1<-as.data.frame(varImp(object=model_nnet)$importance)
> print(df1)
               Overall       bobcat       coyote    gray_fox
Age        1.890322e+01 1.890322e+01 1.890322e+01  18.903219
Number     1.797657e+01 1.797657e+01 1.797657e+01  17.976571
Length     4.826772e+01 4.826772e+01 4.826772e+01  48.267722
Diameter   2.051976e+01 2.051976e+01 2.051976e+01  20.519761
Taper      7.534487e+00 7.534487e+00 7.534487e+00   7.534487
TI         1.314275e+01 1.314275e+01 1.314275e+01  13.142750
Mass       6.197148e+01 6.197148e+01 6.197148e+01  61.971484
d13C       6.532158e+01 6.532158e+01 6.532158e+01  65.321576
d15N       1.000000e+02 1.000000e+02 1.000000e+02 100.000000
CN         5.893423e+01 5.893423e+01 5.893423e+01  58.934234
ropey      4.269846e+01 4.269846e+01 4.269846e+01  42.698458
segmented  1.802613e+01 1.802613e+01 1.802613e+01  18.026128
flat       4.027148e+01 4.027148e+01 4.027148e+01  40.271483
scrape     1.514744e-15 1.514744e-15 1.514744e-15   0.000000
> df2 = data.frame(name = c("d15N
","d13C","Mass","CN","Length","ropey","flat","Diameter","Number","Age","TI","segmented","T
aper","scrape"))
> cbinded_df<-cbind(df1,df2)
> p<-ggplot(data=cbinded_df, aes(x=name, y=Overall)) +
+   geom_bar(stat="identity")+ggtitle('Neural Net - Variable Importance')
> nnet_var_imp<-p + coord_flip()
> nnet_var_imp
> df1<-as.data.frame(varImp(object=model_nnet)$importance)
> print(df1)
               Overall       bobcat       coyote    gray_fox
Age        1.890322e+01 1.890322e+01 1.890322e+01  18.903219
Number     1.797657e+01 1.797657e+01 1.797657e+01  17.976571
Length     4.826772e+01 4.826772e+01 4.826772e+01  48.267722
Diameter   2.051976e+01 2.051976e+01 2.051976e+01  20.519761
Taper      7.534487e+00 7.534487e+00 7.534487e+00   7.534487
TI         1.314275e+01 1.314275e+01 1.314275e+01  13.142750
Mass       6.197148e+01 6.197148e+01 6.197148e+01  61.971484
d13C       6.532158e+01 6.532158e+01 6.532158e+01  65.321576
d15N       1.000000e+02 1.000000e+02 1.000000e+02 100.000000
CN         5.893423e+01 5.893423e+01 5.893423e+01  58.934234
ropey      4.269846e+01 4.269846e+01 4.269846e+01  42.698458
segmented  1.802613e+01 1.802613e+01 1.802613e+01  18.026128
flat       4.027148e+01 4.027148e+01 4.027148e+01  40.271483
scrape     1.514744e-15 1.514744e-15 1.514744e-15   0.000000
```

```
> df2 = data.frame(name = c("d15N
","d13C","Mass","CN","Length","ropey","flat","Diameter","Number","Age","TI","segmented","T
aper","scrape"))
> cbinded_df<-cbind(df1,df2)
> p<-ggplot(data=cbinded_df, aes(x=name, y=Overall)) +
+   geom_bar(stat="identity")+ggtitle('Neural Net - Variable Importance')
> nnet_var_imp<-p + coord_flip()
> nnet_var_imp
```




```
> #Naive Bayes
> plot(varImp(object=model_nbayes),main="Naive Bayes - Variable Importance")
```



```
> ###Confusion Matrix
> #GBM
> predictions<-predict.train(object=model_gbm,testSet[,predictors],type="raw")
> table(predictions)
predictions
 bobcat  coyote gray_fox 
     17       5       5 
> confusionMatrix(predictions,testSet[,outcomeName])
Confusion Matrix and Statistics

          Reference
Prediction bobcat coyote gray_fox
  bobcat       14      1        2
  coyote        0      5        0
  gray_fox      0      1        4

Overall Statistics
                                          
               Accuracy : 0.8519          
                 95% CI : (0.6627, 0.9581)
    No Information Rate : 0.5185          
    P-Value [Acc > NIR] : 0.0003126       
                                          
                  Kappa : 0.7465          
                                          
 Mcnemar's Test P-Value : 0.2614641       

Statistics by Class:

                     Class: bobcat Class: coyote Class: gray_fox
Sensitivity                 1.0000        0.7143          0.6667
Specificity                 0.7692        1.0000          0.9524
Pos Pred Value              0.8235        1.0000          0.8000
Neg Pred Value              1.0000        0.9091          0.9091
Prevalence                  0.5185        0.2593          0.2222
Detection Rate              0.5185        0.1852          0.1481
Detection Prevalence        0.6296        0.1852          0.1852
Balanced Accuracy           0.8846        0.8571          0.8095
> 
> #RF
> predictions<-predict.train(object=model_rf,testSet[,predictors],type="raw")
> table(predictions)
predictions
```

```
 bobcat  coyote gray_fox 
    18       5       4 
> confusionMatrix(predictions,testSet[,outcomeName])
Confusion Matrix and Statistics

          Reference
Prediction bobcat coyote gray_fox
  bobcat       14      2        2
  coyote        0      5        0
  gray_fox      0      0        4

Overall Statistics
                                          
               Accuracy : 0.8519          
                 95% CI : (0.6627, 0.9581)
    No Information Rate : 0.5185          
    P-Value [Acc > NIR] : 0.0003126       
                                          
                  Kappa : 0.7416          
                                          
 Mcnemar's Test P-Value : NA              

Statistics by Class:

                     Class: bobcat Class: coyote Class: gray_fox
Sensitivity                 1.0000        0.7143          0.6667
Specificity                 0.6923        1.0000          1.0000
Pos Pred Value              0.7778        1.0000          1.0000
Neg Pred Value              1.0000        0.9091          0.9130
Prevalence                  0.5185        0.2593          0.2222
Detection Rate              0.5185        0.1852          0.1481
Detection Prevalence        0.6667        0.1852          0.1481
Balanced Accuracy           0.8462        0.8571          0.8333
> 
> #Neural Network
> predictions<-predict.train(object=model_nnet,testSet[,predictors],type="raw")
> table(predictions)
predictions
 bobcat  coyote gray_fox 
    14       7       6 
> confusionMatrix(predictions,testSet[,outcomeName])
Confusion Matrix and Statistics

          Reference
```

```
Prediction bobcat coyote gray_fox
  bobcat      13      0       1
  coyote       1      5       1
  gray_fox     0      2       4

Overall Statistics

               Accuracy : 0.8148
                 95% CI : (0.6192, 0.937)
    No Information Rate : 0.5185
    P-Value [Acc > NIR] : 0.001421

                  Kappa : 0.6987

 Mcnemar's Test P-Value : 0.506165

Statistics by Class:

                     Class: bobcat Class: coyote Class: gray_fox
Sensitivity                 0.9286        0.7143          0.6667
Specificity                 0.9231        0.9000          0.9048
Pos Pred Value              0.9286        0.7143          0.6667
Neg Pred Value              0.9231        0.9000          0.9048
Prevalence                  0.5185        0.2593          0.2222
Detection Rate              0.4815        0.1852          0.1481
Detection Prevalence        0.5185        0.2593          0.2222
Balanced Accuracy           0.9258        0.8071          0.7857
>
> #Naive Bayes
> predictions<-predict.train(object=model_nbayes,testSet[,predictors],type="raw")
> table(predictions)
predictions
  bobcat   coyote gray_fox
      18        5        4
> confusionMatrix(predictions,testSet[,outcomeName])
Confusion Matrix and Statistics

          Reference
Prediction bobcat coyote gray_fox
  bobcat      14      2       2
  coyote       0      5       0
  gray_fox     0      0       4

Overall Statistics
```

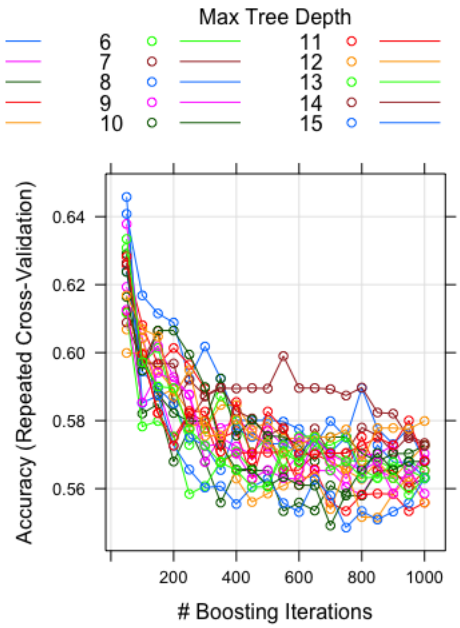
```
            Accuracy : 0.8519
              95% CI : (0.6627, 0.9581)
    No Information Rate : 0.5185
    P-Value [Acc > NIR] : 0.0003126

                  Kappa : 0.7416

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: bobcat Class: coyote Class: gray_fox
Sensitivity                 1.0000        0.7143          0.6667
Specificity                 0.6923        1.0000          1.0000
Pos Pred Value              0.7778        1.0000          1.0000
Neg Pred Value              1.0000        0.9091          0.9130
Prevalence                  0.5185        0.2593          0.2222
Detection Rate              0.5185        0.1852          0.1481
Detection Prevalence        0.6667        0.1852          0.1481
Balanced Accuracy           0.8462        0.8571          0.8333
>
> gbm_df <- data.frame("Experiment" = 'GBM', "Accuracy" = model_gbm$results$Accuracy,
"Kappa" = model_gbm$results$Kappa)
> gbm_df <-gbm_df[order(-gbm_df$Accuracy),]
> rf_df <- data.frame("Experiment" = 'Random Forest', "Accuracy" =
model_rf$results$Accuracy, "Kappa" = model_rf$results$Kappa)
> rf_df <-rf_df[order(-rf_df$Accuracy),]
> nnet_df <- data.frame("Experiment" = 'Neural Network', "Accuracy" =
model_nnet$results$Accuracy, "Kappa" = model_nnet$results$Kappa)
> nnet_df <-nnet_df[order(-nnet_df$Accuracy),]
> nb_df <- data.frame("Experiment" = 'Naive Bayes', "Accuracy" =
model_nbayes$results$Accuracy, "Kappa" = model_nbayes$results$Kappa)
> nb_df <-nb_df[order(-nb_df$Accuracy),]
> total <- rbind(gbm_df[1,], rf_df[1,],nnet_df[1,],nb_df[1,])
> total <-total[order(-total$Accuracy),]
> print(total)
       Experiment  Accuracy     Kappa
9  Neural Network 0.7042263 0.5123361
21    Naive Bayes 0.6643524 0.4282045
2   Random Forest 0.6640078 0.4357575
1             GBM 0.6387128 0.3914874
>
    7       0.7574           nan     0.1000    0.0259
```

8	0.7261	nan	0.1000	0.0104
9	0.6969	nan	0.1000	0.0003
10	0.6651	nan	0.1000	-0.0139
20	0.4711	nan	0.1000	-0.0011
40	0.2955	nan	0.1000	-0.0106
60	0.1944	nan	0.1000	-0.0067
80	0.1380	nan	0.1000	-0.0123
100	0.0875	nan	0.1000	-0.0124
120	0.0629	nan	0.1000	-0.0055
140	0.0471	nan	0.1000	-0.0044
160	0.0301	nan	0.1000	-0.0017
540	0.0001	nan	0.1000	-0.0000
560	0.0001	nan	0.1000	-0.0000
580	0.0000	nan	0.1000	-0.0000
600	0.0000	nan	0.1000	-0.0000
620	0.0000	nan	0.1000	-0.0000
640	0.0000	nan	0.1000	-0.0000
660	0.0000	nan	0.1000	-0.0000
680	0.0000	nan	0.1000	-0.0000
700	0.0000	nan	0.1000	-0.0000
720	0.0000	nan	0.1000	-0.0000
740	0.0000	nan	0.1000	-0.0000
760	0.0000	nan	0.1000	-0.0000
780	0.0000	nan	0.1000	-0.0000
800	0.0000	nan	0.1000	-0.0000
820	0.0000	nan	0.1000	-0.0000

540	0.0001	nan	0.1000	-0.0000
560	0.0001	nan	0.1000	-0.0000
580	0.0000	nan	0.1000	-0.0000
600	0.0000	nan	0.1000	-0.0000
620	0.0000	nan	0.1000	-0.0000
640	0.0000	nan	0.1000	-0.0000
660	0.0000	nan	0.1000	-0.0000
680	0.0000	nan	0.1000	-0.0000
700	0.0000	nan	0.1000	-0.0000
720	0.0000	nan	0.1000	-0.0000
740	0.0000	nan	0.1000	-0.0000
760	0.0000	nan	0.1000	-0.0000
780	0.0000	nan	0.1000	-0.0000
800	0.0000	nan	0.1000	-0.0000
820	0.0000	nan	0.1000	-0.0000
840	0.0000	nan	0.1000	-0.0000
860	0.0000	nan	0.1000	-0.0000

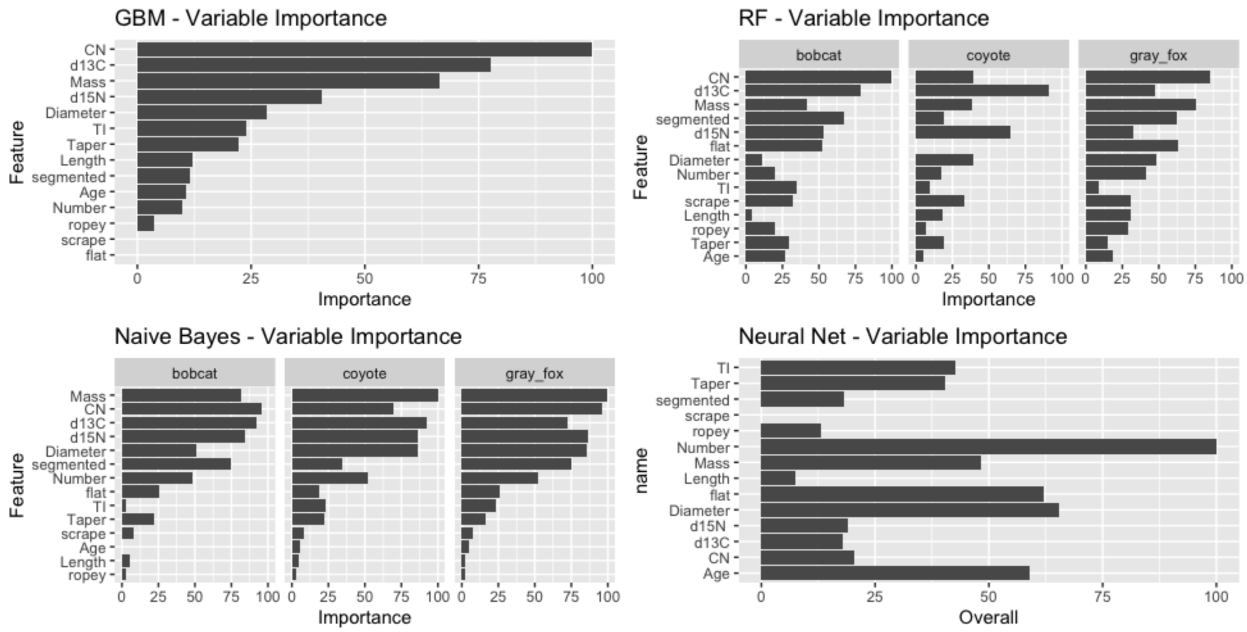
```
   880        0.0000             nan     0.1000   -0.0000
   900        0.0000             nan     0.1000   -0.0000
   920        0.0000             nan     0.1000   -0.0000
   940        0.0000             nan     0.1000   -0.0000
   960        0.0000             nan     0.1000   -0.0000
   980        0.0000             nan     0.1000   -0.0000
  1000        0.0000             nan     0.1000   -0.0000

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1        1.0986             nan     0.1000    0.1484
     2        0.9869             nan     0.1000    0.0916
     3        0.9080             nan     0.1000    0.0335
     4        0.8536             nan     0.1000    0.0703
     5        0.7955             nan     0.1000    0.0303
     6        0.7431             nan     0.1000    0.0349
     7        0.6966             nan     0.1000    0.0066
     8        0.6692             nan     0.1000   -0.0109
     9        0.6473             nan     0.1000   -0.0015
    10        0.6251             nan     0.1000    0.0185
    20        0.4853             nan     0.1000   -0.0025
    40        0.2706             nan     0.1000   -0.0081
    50        0.2303             nan     0.1000    0.0007

>
> print(model_gbm_tune_7)
Stochastic Gradient Boosting

83 samples
14 predictors
 3 classes: 'bobcat', 'coyote', 'gray_fox'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 67, 66, 66, 66, 67, 66, ...
Resampling results across tuning parameters:

  interaction.depth  n.trees  Accuracy   Kappa
  1                    50     0.6408039  0.4077184
  1                   100     0.6168333  0.3667855
  1                   150     0.6115196  0.3585280
  1                   200     0.6088529  0.3537218
  1                  1000     0.5678039  0.2880220
  2                    50     0.6193333  0.3637752
  2                   100     0.6043333  0.3452883
```

```
  2                150    0.5897549  0.3306283
  2                200    0.5751765  0.3085288
  2                250    0.5795882  0.3178003
  2                300    0.5776765  0.3126046
  2                350    0.5778235  0.3114129
 11                 900    0.5583922  0.2813404
 11                 950    0.5533922  0.2730906
 11                1000    0.5558922  0.2761914
 12                  50    0.5999020  0.3328887
 12                 100    0.5997549  0.3415974
 12                 150    0.5947549  0.3356803
 13                  50    0.6116667  0.3628699
 13                 100    0.5782843  0.3084900
 13                 150    0.5798824  0.3146955
 13                 200    0.5753235  0.3057935
 13                 250    0.5584118  0.2747114
 13                 300    0.5607647  0.2770250
 13                 350    0.5679706  0.2924458
 13                 400    0.5709314  0.3017999
 13                 450    0.5654706  0.2908100
 13                 500    0.5606176  0.2851557
 [ reached getOption("max.print") -- omitted 150 rows ]

Tuning parameter 'shrinkage' was held constant at a value of 0.1
Tuning parameter 'n.minobsinnode' was held constant at a value of 10
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were n.trees = 50, interaction.depth = 15, shrinkage = 0.1
and n.minobsinnode = 10.
>
```



########## 8 Using GGplot and gridExtra to plot all variable of importance plots into one single plot. (10 points)

```
> ########## 8 Using GGplot and gridExtra to plot all variable of importance plots into one single plot. (10 points)
> #GBM
> gbm_var_imp<-ggplot(varImp(object=model_gbm))+ggtitle('GBM - Variable Importance')
> #RF
> rf_var_imp<-ggplot(varImp(object=model_rf))+ggtitle('RF - Variable Importance')
> #NNET
> df1<-as.data.frame(varImp(object=model_nnet)$importance)
> #print(df1)
> df2 = data.frame(name = c("d15N
","d13C","Mass","CN","Length","ropey","flat","Diameter","Number","Age","TI","segmented","Taper","scrape"))
> cbinded_df<-cbind(df1,df2)
> p<-ggplot(data=cbinded_df, aes(x=name, y=Overall)) +
+   geom_bar(stat="identity")+ggtitle('Neural Net - Variable Importance')
> nnet_var_imp<-p + coord_flip()
> #NB
> nb_var_imp<-ggplot(varImp(object=model_nbayes))+ggtitle('Naive Bayes - Variable Importance')
> #Combining the data
> grid.arrange(gbm_var_imp, rf_var_imp,nb_var_imp,nnet_var_imp)
>
```



######### 9 Which model performs the best? and why do you think this is the case?
#Can we accurately predict species on this dataset? (10 points)

print(total)

#The Neural Network performs the best with an accuracy of 70%.
Neural networks model is the best as it shows the ability to learn on non-linear relationships and complex
relationships like seen in the dataset we have.
Neural network here takes into consideration all the other features and builds a weighted relationship in between them
Hence this relationship helps in acheieving the highest accuracy
YES we can predict the species with this model

```
> print(total)
      Experiment  Accuracy     Kappa
9  Neural Network 0.7042263 0.5123361
21    Naive Bayes 0.6643524 0.4282045
2   Random Forest 0.6640078 0.4357575
1             GBM 0.6387128 0.3914874
```

#################### Graduate Question
#Using feature selection with rfe in caret and the repeatedcv method: Find the top 3
#predictors and build the same models as in 6 and 8 with the same parameters. (20 points)

```
> control <- rfeControl(functions = rfFuncs,
+                   method = "repeatedcv",
+                   repeats = 3,
+                   verbose = FALSE)
> outcomeName<-'Species'
> predictors<-names(trainSet)[!names(trainSet) %in% outcomeName]
> Species_Pred_Profile <- rfe(trainSet[,predictors], trainSet[,outcomeName],rfeControl =
control)
> Species_Pred_Profile

Recursive feature selection

Outer resampling method: Cross-Validated (10 fold, repeated 3 times)

Resampling performance over subset size:

 Variables Accuracy  Kappa AccuracySD KappaSD Selected
         4   0.6950 0.4807     0.1466  0.2478        *
         8   0.6917 0.4685     0.1610  0.2701
        14   0.6759 0.4365     0.1528  0.2598

The top 4 variables (out of 4):
   CN, d13C, d15N, Mass

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1        1.0986             nan     0.1000    0.1201
     2        1.0215             nan     0.1000    0.0707
     3        0.9598             nan     0.1000    0.0761
     4        0.9021             nan     0.1000    0.0335
     5        0.8707             nan     0.1000    0.0025
     6        0.8386             nan     0.1000    0.0501
     7        0.8043             nan     0.1000    0.0257
     8        0.7734             nan     0.1000    0.0158
     9        0.7520             nan     0.1000    0.0164
    10        0.7273             nan     0.1000    0.0210
    20        0.6156             nan     0.1000   -0.0286
    40        0.5303             nan     0.1000   -0.0282
    50        0.5033             nan     0.1000   -0.0171

>> model_rf_10<-train(trainSet[,predictors_top3],trainSet[,outcomeName],method='rf',
importance=T)
note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .

>iter  80 value 18.517038
```

```
iter  90 value 17.745407
iter 100 value 17.159709
final  value 17.159709
stopped after 100 iterations
# weights:  24
initial  value 120.658672
iter  10 value 54.565501
iter  20 value 51.233331
iter  30 value 50.769308
iter  40 value 50.768222
iter  40 value 50.768222
iter  40 value 50.768222
final  value 50.768222
converged
>> model_nbayes_10<-
train(trainSet[,predictors_top3],trainSet[,outcomeName],method='naive_bayes',importance=T)
There were 50 or more warnings (use warnings() to see the first 50)
>
> model_rf_10_tune<-train(trainSet[,predictors],trainSet[,outcomeName],method='rf',
importance=T,trControl=fitControl,tuneLength=20)
note: only 13 unique complexity parameters in default grid. Truncating the grid to 13 .

>inal  value 0.080192
stopped after 100 iterations
# weights:  57
initial  value 121.916980
iter  10 value 46.220121
iter  20 value 34.492048
iter  30 value 32.988396
iter  40 value 31.958370
iter  50 value 30.817493
iter  60 value 30.526914
iter  70 value 30.152870
iter  80 value 30.106078
final  value 30.106039
converged
>
> model_nbayes_10_tune<-
train(trainSet[,predictors],trainSet[,outcomeName],method='naive_bayes',importance=T,trCon
trol=fitControl,tuneLength=20)
There were 50 or more warnings (use warnings() to see the first 50)
>520       0.0221           nan     0.1000   -0.0040
   540       0.0213           nan     0.1000   -0.0034
   560       0.0215           nan     0.1000   -0.0093
```

```
   580      0.0204          nan    0.1000   -0.0086
   600      0.0196          nan    0.1000   -0.0001
   620      0.0186          nan    0.1000   -0.0089
   640      0.0189          nan    0.1000   -0.0033
   660      0.0152          nan    0.1000   -0.0006
   680      0.0146          nan    0.1000   -0.0063
   700      0.0143          nan    0.1000   -0.0001
   720      0.0150          nan    0.1000    0.0002
   740      0.0147          nan    0.1000   -0.0008
   760      0.0150          nan    0.1000   -0.0020
   780      0.0144          nan    0.1000   -0.0029
   800      0.0120          nan    0.1000   -0.0000
   820      0.0116          nan    0.1000   -0.0033
   840      0.0114          nan    0.1000   -0.0036
   860      0.0106          nan    0.1000   -0.0034
   880      0.0096          nan    0.1000   -0.0025
   900      0.0103          nan    0.1000   -0.0045

>> model_rf_10_tune_top3<-
train(trainSet[,predictors_top3],trainSet[,outcomeName],method='rf',
importance=T,trControl=fitControl,tuneLength=20)
note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .

stopped after 100 iterations
# weights:  52
initial  value 93.717098
iter  10 value 50.877757
iter  20 value 49.776962
iter  30 value 49.615422
iter  40 value 49.582843
iter  50 value 49.580839
iter  60 value 49.579469
iter  70 value 49.578307
iter  80 value 49.578100
final  value 49.578080
converged
>
> model_nbayes_10_tune_top3<-
train(trainSet[,predictors_top3],trainSet[,outcomeName],method='naive_bayes',importance=T,
trControl=fitControl,tuneLength=20)
There were 50 or more warnings (use warnings() to see the first 50)
>
> #For models using top 3 predictors
```

```
> gbm_df_10 <- data.frame("Experiment" = 'GBM with top 3 Features', "Accuracy" =
model_gbm_10$results$Accuracy, "Kappa" = model_gbm_10$results$Kappa)
> gbm_df_10 <-gbm_df_10[order(-gbm_df_10$Accuracy),]
> rf_df_10 <- data.frame("Experiment" = 'Random Forest with top 3 Features', "Accuracy" =
model_rf_10$results$Accuracy, "Kappa" = model_rf_10$results$Kappa)
> rf_df_10 <-rf_df_10[order(-rf_df_10$Accuracy),]
> nnet_df_10 <- data.frame("Experiment" = 'Neural Network with top 3 Features', "Accuracy" =
model_nnet_10$results$Accuracy, "Kappa" = model_nnet_10$results$Kappa)
> nnet_df_10 <-nnet_df_10[order(-nnet_df_10$Accuracy),]
> nb_df_10 <- data.frame("Experiment" = 'Naive Bayes with top 3 Features', "Accuracy" =
model_nbayes_10$results$Accuracy, "Kappa" = model_nbayes_10$results$Kappa)
> nb_df_10 <-nb_df_10[order(-nb_df_10$Accuracy),]
> #For models using tuning for all features
> gbm_df_10_tune <- data.frame("Experiment" = 'GBM with Tune for all features', "Accuracy" =
model_gbm_10_tune$results$Accuracy, "Kappa" = model_gbm_10_tune$results$Kappa)
> gbm_df_10_tune <-gbm_df_10_tune[order(-gbm_df_10_tune$Accuracy),]
> rf_df_10_tune <- data.frame("Experiment" = 'Random Forest with Tune for all features',
"Accuracy" = model_rf_10_tune$results$Accuracy, "Kappa" =
model_rf_10_tune$results$Kappa)
> rf_df_10_tune <-rf_df_10_tune[order(-rf_df_10_tune$Accuracy),]
> nnet_df_10_tune <- data.frame("Experiment" = 'Neural Network with Tune for all features',
"Accuracy" = model_nnet_10_tune$results$Accuracy, "Kappa" =
model_nnet_10_tune$results$Kappa)
> nnet_df_10_tune <-nnet_df_10_tune[order(-nnet_df_10_tune$Accuracy),]
> nb_df_10_tune <- data.frame("Experiment" = 'Naive Bayes with Tune for all features',
"Accuracy" = model_nbayes_10_tune$results$Accuracy, "Kappa" =
model_nbayes_10_tune$results$Kappa)
> nb_df_10_tune <-nb_df_10_tune[order(-nb_df_10_tune$Accuracy),]
> #For models using tuning with top 3 features
> gbm_df_10_tune_top3 <- data.frame("Experiment" = 'GBM with Tune for top 3 Features',
"Accuracy" = model_gbm_10_tune_top3$results$Accuracy, "Kappa" =
model_gbm_10_tune_top3$results$Kappa)
> gbm_df_10_tune_top3 <-gbm_df_10_tune_top3[order(-gbm_df_10_tune_top3$Accuracy),]
> rf_df_10_tune_top3 <- data.frame("Experiment" = 'Random Forest with Tune for top 3
Features', "Accuracy" = model_rf_10_tune_top3$results$Accuracy, "Kappa" =
model_rf_10_tune_top3$results$Kappa)
> rf_df_10_tune_top3 <-rf_df_10_tune_top3[order(-rf_df_10_tune_top3$Accuracy),]
> nnet_df_10_tune_top3 <- data.frame("Experiment" = 'Neural Network with Tune for top 3
Features', "Accuracy" = model_nnet_10_tune_top3$results$Accuracy, "Kappa" =
model_nnet_10_tune_top3$results$Kappa)
> nnet_df_10_tune_top3 <-nnet_df_10_tune_top3[order(-nnet_df_10_tune_top3$Accuracy),]
> nb_df_10_tune_top3 <- data.frame("Experiment" = 'Naive Bayes with Tune for top 3
Features', "Accuracy" = model_nbayes_10_tune_top3$results$Accuracy, "Kappa" =
model_nbayes_10_tune_top3$results$Kappa)
```

```
> nb_df_10_tune_top3 <-nb_df_10_tune_top3[order(-nb_df_10_tune_top3$Accuracy),]
> total_10 <- rbind(gbm_df_10[1,],
rf_df_10[1,],nnet_df_10[1,],nb_df_10[1,],gbm_df_10_tune[1,],
rf_df_10_tune[1,],nnet_df_10_tune[1,],nb_df_10_tune[1,],gbm_df_10_tune_top3[1,],rf_df_10
_tune_top3[1,],nnet_df_10_tune_top3[1,],nb_df_10_tune_top3[1,])
> total_10 <-total_10[order(-total_10$Accuracy),]
>
> print(total_10)
                                   Experiment  Accuracy     Kappa
80  Neural Network with Tune for top 3 Features 0.7655752 0.6010872
11             Naive Bayes with top 3 Features 0.7564246 0.5765330
40    Neural Network with Tune for all features 0.7378105 0.5660206
22     Naive Bayes with Tune for top 3 Features 0.7360327 0.5345680
6           Neural Network with top 3 Features 0.7298132 0.5449672
12   Random Forest with Tune for top 3 Features 0.7145033 0.5096485
4      Random Forest with Tune for all features 0.7029183 0.4847383
2            Random Forest with top 3 Features 0.6926276 0.4668292
353           GBM with Tune for top 3 Features 0.6809118 0.4729273
21       Naive Bayes with Tune for all features 0.6803497 0.4573714
15              GBM with Tune for all features 0.6416111 0.4058856
1                      GBM with top 3 Features 0.6395396 0.3831167
>
```

#c. Which model performs the best? and why do you think this is the case?
#Can we accurately predict species on this dataset? (10 points)
#Ans---The Neural Network model with top 3 parameters and parameter tunning works the best acheieving
upto accuracy of 77%.
Neural networks model is the best as it shows the ability to learn on non-linear relationships and complex
relationships like seen in the dataset we have.
Plus the neural network here is using the best of 3 features and tuning them. Hence the accuracy is higher than the previous.
Yes, We can predict the species with using this model.