```
library(caret)
library(gbm)
library(e1071)

data(scat)

str(scat)
```

Output:

```
'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.**

```
> outcomeName<-'Species'
> #Converting outcome variable to numeric
> scat$Species<-ifelse(scat$Species=="bobcat",0,ifelse(scat$Species=="coyote", 1, 2))
> # After converting outcome variable to numeric
> str(scat)
'data.frame':	110 obs. of  19 variables:
 $ Species  : num  1 1 0 1 1 1 0 0 0 0 ...
 $ 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 ...
>
```

> **# 2. Remove the Month, Year, Site, Location features.**

```
> scat[ ,c('Month', 'Year', 'Site', 'Location')] <- list(NULL)
> str(scat)
'data.frame':   110 obs. of  15 variables:
 $ Species  : num  1 1 0 1 1 1 0 0 0 0 ...
 $ 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 ...
>
```

> **# 3. Check if any values are null.**
> # If there are, impute missing values using KNN.

```
> sum(is.na(scat))
[1] 47
> preProcValues <- preProcess(scat[,c('Age', 'Number', 'Length', 'Diameter','Taper','TI','Mass','d13C','d15N','CN','ropey','segmented','flat','scrape')], method = c("knnImpute","center","scale"))
> library('RANN')
```

```
> scat_processed <- predict(preProcValues, scat)
> sum(is.na(scat_processed))
[1] 0
>
```

 # 4. Converting every categorical variable to numerical (if needed).

```
> str(scat_processed) # after processing
'data.frame':   110 obs. of  15 variables:
 $ Species  : num  1 1 0 1 1 1 0 0 0 0 ...
 $ 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 ...
>
```

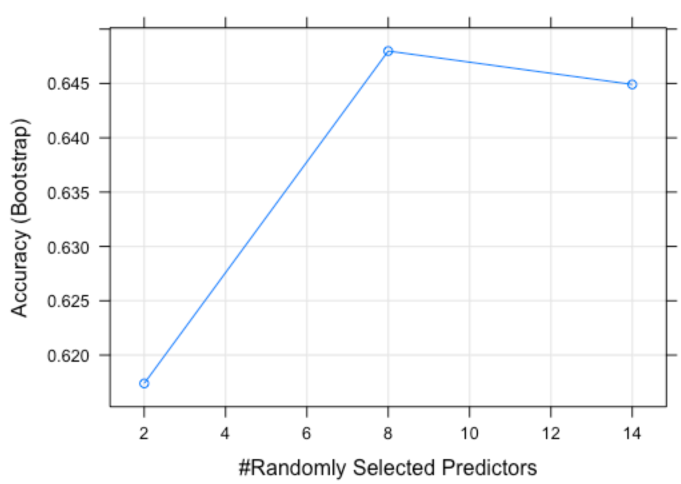
 # Answer - After removing columns in Question 2 and converting outcome column
```
# to numeric, there are no more categorical variables.
# There is no need for any conversion at this point.
```

 # 5. With a seed of 100, 75% training, 25% testing.
 # Build the following models: randomforest, neural net, naive bayes and GBM.
 # a. For these models display a)model summarization and
 # b) plot variable of importance, for the predictions (use the prediction set) display
 # c) confusion matrix (60 points)

```
> #Converting the dependent variable back to categorical
> scat_processed$Species<-as.factor(scat_processed$Species)
> str(scat_processed)
'data.frame':   110 obs. of  15 variables:
 $ Species  : Factor w/ 3 levels "0","1","2": 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 ...
```

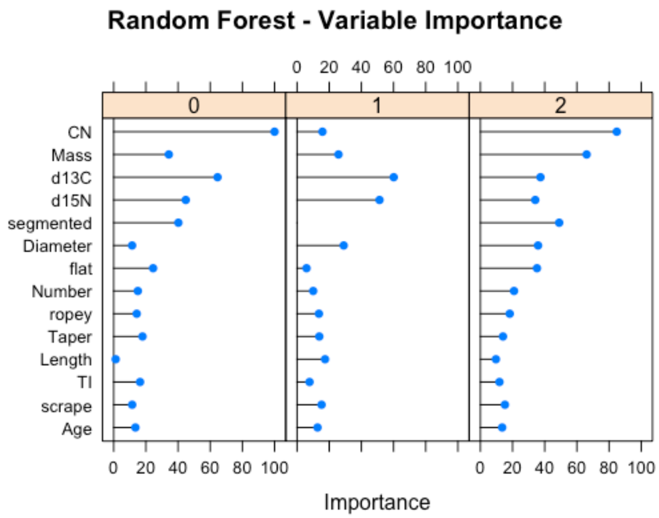
```
 $ Length   : num  0.0587 1.3679 -0.0867 -0.2322 -0.3777 ...
 $ Diameter : num  1.8396 1.7623 0.0622 -0.1181 0.5516 ...
 $ Taper    : num  0.961 0.642 -0.726 -0.182 -0.487 ...
 $ TI       : num  0.0283 -0.1406 -0.7171 -0.24 -0.6277 ...
 $ Mass     : num  0.388 0.583 -0.458 -0.571 1.469 ...
 $ d13C     : 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 ...
>

> #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)
> trainSet <- scat_processed[ index,]
> testSet <- scat_processed[-index,]
> #Checking the structure of trainSet
> str(trainSet)
'data.frame':	83 obs. of  15 variables:
 $ Species  : Factor w/ 3 levels "0","1","2": 2 1 2 2 2 1 1 3 3 3 ...
 $ Age      : num  1.207 -0.252 1.207 1.207 1.207 ...
 $ Number   : num  -0.433 -0.433 -0.433 0.968 0.268 ...
 $ Length   : num  0.0587 -0.0867 -0.2322 -0.3777 -0.0867 ...
 $ Diameter : num  1.8396 0.0622 -0.1181 0.5516 0.6804 ...
 $ Taper    : num  0.9609 -0.7262 -0.1816 -0.4871 0.0709 ...
 $ TI       : num  0.0283 -0.7171 -0.24 -0.6277 -0.2599 ...
 $ Mass     : num  0.388 -0.458 -0.571 1.469 0.19 ...
 $ d13C     : num  0.00468 -0.85947 3.12113 1.66403 -0.98357 ...
 $ d15N     : num  -0.165 0.359 -0.546 -0.141 0.28 ...
 $ CN       : num  0.0276 -0.0816 0.8468 0.6011 0.1642 ...
 $ ropey    : num  -1.131 0.876 0.876 -1.131 0.876 ...
 $ segmented: num  -1.131 0.876 -1.131 0.876 -1.131 ...
 $ flat     : num  -0.239 -0.239 -0.239 -0.239 -0.239 ...
 $ scrape   : num  -0.217 4.562 -0.217 -0.217 -0.217 ...
> # all variables as predictors
> predictors<-c("Age", "Number", "Length", "Diameter", "Taper", "TI", "Mass", "d13C",
+               "d15N", "CN", "ropey", "segmented", "flat", "scrape")
>

> # ##### randomforest ######
> model_rf<-train(trainSet[,predictors],trainSet[,outcomeName],method='rf', importance=T)
```

```
> # summarizing the model
> print(model_rf)
Random Forest

83 samples
14 predictors
 3 classes: '0', '1', '2'

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    0.6173904  0.3456397
   8    0.6479826  0.4143276
  14    0.6449064  0.4170319

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 8.
>

> # Visualizing the models
> plot(model_rf)
>
```




```
> varImp(object=model_rf)
rf variable importance

  variables are sorted by maximum importance across the classes
            0     1     2
```

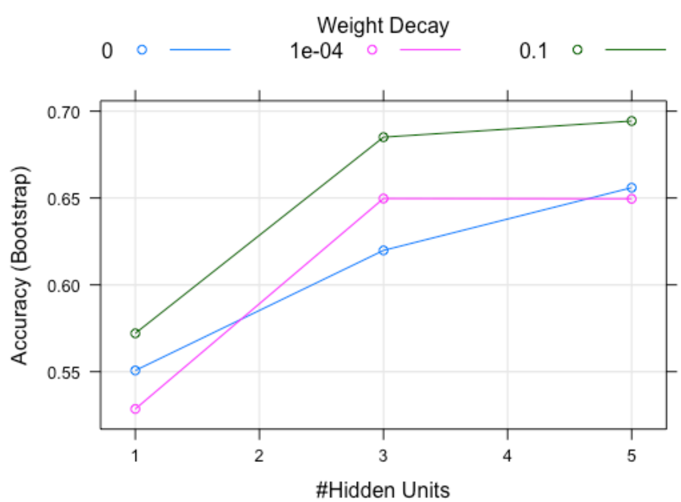
```
CN        100.000 15.788 84.801
Mass       34.267 25.760 66.057
d13C       64.580 60.099 37.362
d15N       44.807 51.268 34.111
segmented  40.202  0.000 48.987
Diameter   11.442 29.001 35.856
flat       24.498  5.886 35.177
Number     15.000 10.028 20.908
ropey      14.312 13.604 18.244
Taper      17.905 13.834 14.042
Length      1.228 17.413  9.645
TI         16.384  7.761 11.851
scrape     11.487 15.256 15.256
Age        13.491 12.742 13.537
>

> #Plotting Varianle importance for Random Forest
> plot(varImp(object=model_rf),main="Random Forest - Variable Importance")
>
```




```
> #Predictions
> predictions_rf<-predict.train(object=model_rf,testSet[,predictors],type="raw")
> table(predictions_rf)
predictions_rf
 0  1  2
18  5  4
> #Confusion Matrix and Statistics
> confusionMatrix(predictions_rf,testSet[,outcomeName])
Confusion Matrix and Statistics

          Reference
```

```
Prediction  0  1  2
       0 14  2  2
       1  0  5  0
       2  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: 0 Class: 1 Class: 2
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 Net #######
> model_nnet<-train(trainSet[,predictors],trainSet[,outcomeName],method='nnet',
importance=T)
> # summarizing the model
> print(model_nnet)
Neural Network

83 samples
14 predictors
 3 classes: '0', '1', '2'

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  0.5506913  0.2859758
 1     1e-04  0.5285263  0.2388859
 1     1e-01  0.5720865  0.3012730
 3     0e+00  0.6198346  0.3975515
 3     1e-04  0.6497167  0.4392089
 3     1e-01  0.6850518  0.4921780
 5     0e+00  0.6559566  0.4472659
 5     1e-04  0.6495575  0.4435142
 5     1e-01  0.6943589  0.5100720

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.
> # Visualizing the models
> plot(model_nnet)
```



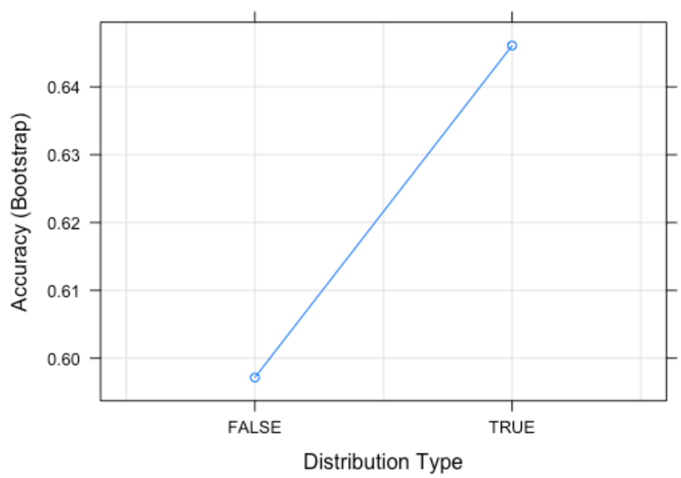

```
> #Variable Importance
> nnet_varImp<- varImp(object=model_nnet)
> #Plotting Variable importance for Neural Net
> plot(varImp(object=model_nnet),main="Neural Net - Variable Importance")
Error in rep.int(factor(names(x), unique(names(x))), lengths(x)) :
 invalid 'times' value
> #Predictions
> predictions_nnet<-predict.train(object=model_nnet,testSet[,predictors],type="raw")
> table(predictions_nnet)
predictions_nnet
 0  1  2
15  7  5
> #Confusion Matrix and Statistics
> confusionMatrix(predictions_nnet,testSet[,outcomeName])
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2
         0 13  1  1
         1  1  5  1
         2  0  1  4

Overall Statistics

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

                  Kappa : 0.6932

 Mcnemar's Test P-Value : 0.801252

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity            0.9286   0.7143   0.6667
Specificity            0.8462   0.9000   0.9524
Pos Pred Value         0.8667   0.7143   0.8000
Neg Pred Value         0.9167   0.9000   0.9091
Prevalence             0.5185   0.2593   0.2222
Detection Rate         0.4815   0.1852   0.1481
Detection Prevalence   0.5556   0.2593   0.1852
Balanced Accuracy      0.8874   0.8071   0.8095
>

> # ########### Naive Bayes ###########
> model_nb<-train(trainSet[,predictors],trainSet[,outcomeName],method='naive_bayes',
importance=T)
There were 50 or more warnings (use warnings() to see the first 50)
> # summarizing the model
> print(model_nb)
Naive Bayes

83 samples
14 predictors
 3 classes: '0', '1', '2'
```

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      0.5971475  0.3907572
   TRUE      0.6461044  0.4180012
```

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.
> # Visualizing the models
> plot(model_nb)
>

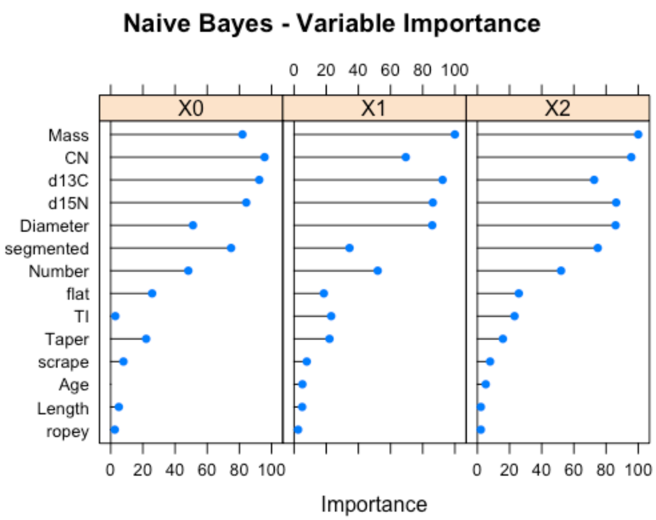


> #Variable Importance
> varImp(object=model_nb)
ROC curve variable importance

  variables are sorted by maximum importance across the classes

	X0	X1	X2
Mass	81.911	100.000	100.000
CN	95.670	69.479	95.670
d13C	92.359	92.359	72.587
d15N	84.284	86.294	86.294
Diameter	51.142	85.913	85.913
segmented	74.845	34.487	74.845
Number	48.256	52.028	52.028

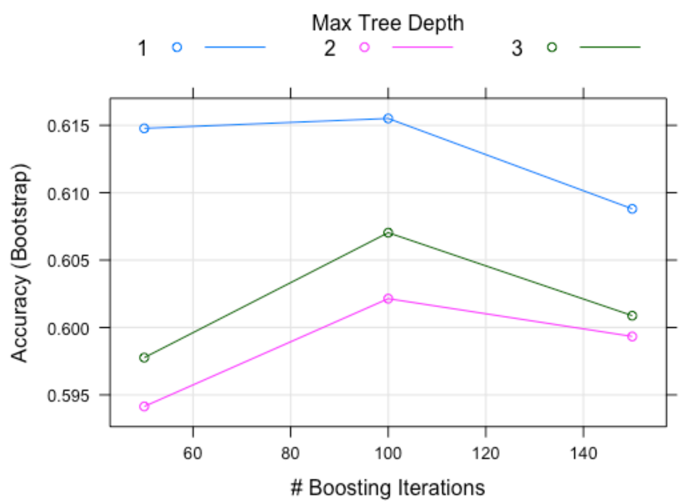
```
flat      25.757  18.523  25.757
TI         2.860  23.092  23.092
Taper     22.038  22.038  15.858
scrape     7.907   7.907   7.907
Age        0.000   5.197   5.197
Length     5.047   5.047   2.152
ropey      2.523   2.523   2.143
> #Plotting Variable importance for Naive Bayes
> plot(varImp(object=model_nb),main="Naive Bayes - Variable Importance")
>
```




```
> #Predictions
> predictions_nb<-predict.train(object=model_nb,testSet[,predictors],type="raw")
> table(predictions_nb)
predictions_nb
 0  1  2
18  5  4
> #Confusion Matrix and Statistics
> confusionMatrix(predictions_nb,testSet[,outcomeName])
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2
         0 14  2  2
         1  0  5  0
         2  0  0  4

Overall Statistics

               Accuracy : 0.8519
                 95% CI : (0.6627, 0.9581)
```

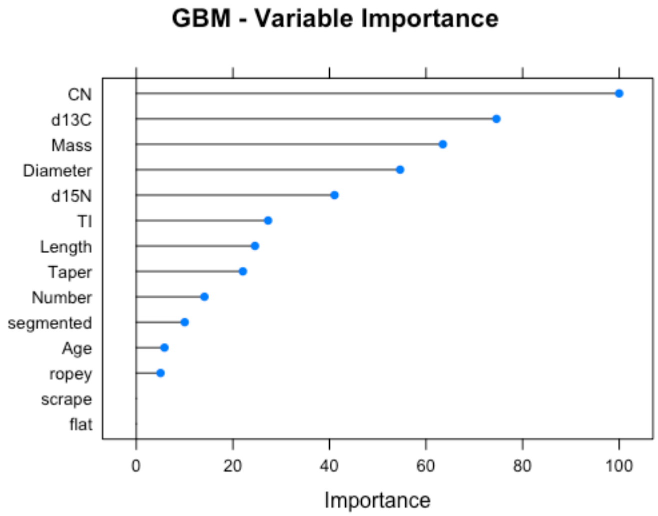
```
   No Information Rate : 0.5185
   P-Value [Acc > NIR] : 0.0003126

              Kappa : 0.7416

 Mcnemar's Test P-Value : NA

Statistics by Class:

                 Class: 0 Class: 1 Class: 2
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 ##########
> model_gbm1<-train(trainSet[,predictors],trainSet[,outcomeName],method='gbm')

> # summarizing the model
> print(model_gbm1)
Stochastic Gradient Boosting

83 samples
14 predictors
 3 classes: '0', '1', '2'

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     0.6147733  0.3651472
  1                  100     0.6155064  0.3710730
  1                  150     0.6088021  0.3618473
  2                   50     0.5941397  0.3363546
  2                  100     0.6021387  0.3509252
  2                  150     0.5993344  0.3452358
  3                   50     0.5977627  0.3374063
```

```
 3              100     0.6070331  0.3523725
 3              150     0.6008749  0.3491138

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 = 100, interaction.depth = 1, shrinkage
 = 0.1 and n.minobsinnode = 10.
> # Visualizing the models
> plot(model_gbm1)
>
```




```
> #Variable Importance
> varImp(object=model_gbm1)
gbm variable importance

          Overall
CN        100.000
d13C       74.587
Mass       63.478
Diameter   54.631
d15N       41.043
TI         27.289
Length     24.577
Taper      22.064
Number     14.108
segmented  10.024
Age         5.829
ropey       5.020
scrape      0.000
```

```
flat      0.000
> #Plotting Variable importance for GBM
> plot(varImp(object=model_gbm1),main="GBM - Variable Importance")
```




```
> #Predictions
> predictions_gbm<-predict.train(object=model_gbm1,testSet[,predictors],type="raw")
> table(predictions_gbm)
predictions_gbm
 0  1  2 
17  5  5 
> #Confusion Matrix and Statistics
> confusionMatrix(predictions_gbm,testSet[,outcomeName])
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2
         0 14  1  2
         1  0  5  0
         2  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: 0 Class: 1 Class: 2
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
>
```

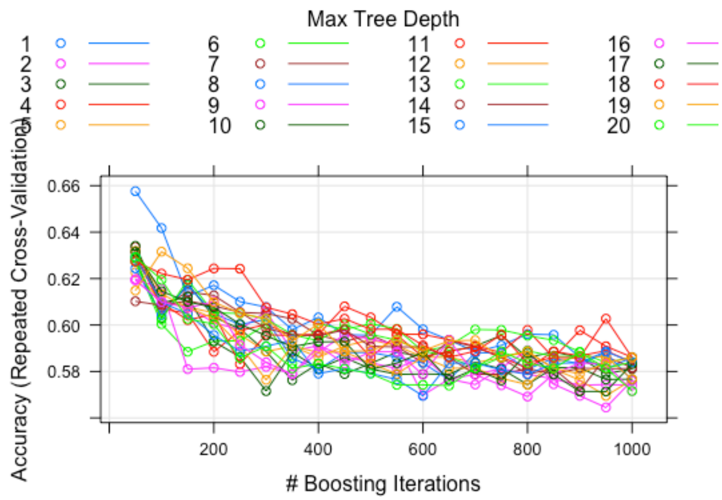
 # 6. For the BEST performing models of each (randomforest, neural net, naive bayes and gbm)
 # create and display a data frame that has the following columns:
 # ExperimentName, accuracy, kappa.
 # Sort the data frame by accuracy.

```
> experimentName<-c("Random Forest" , "Neural Net", "Naive Bayes",  "GBM")
> accuracyDetails<-c(max(model_rf$results$Accuracy), max(model_nnet$results$Accuracy),
+                   max(model_nb$results$Accuracy), max(model_gbm1$results$Accuracy))
> kappaDetails<-c(max(model_rf$results$Kappa), max(model_nnet$results$Kappa),
+                   max(model_nb$results$Kappa), max(model_gbm1$results$Kappa))
> bestModelDf<-data.frame(ExperimentName=experimentName, Accuracy=accuracyDetails,
Kappa=kappaDetails)
> print(bestModelDf[order(-bestModelDf$Accuracy),])
  ExperimentName  Accuracy     Kappa
2     Neural Net 0.6943589 0.5100720
1  Random Forest 0.6479826 0.4170319
3    Naive Bayes 0.6461044 0.4180012
4            GBM 0.6155064 0.3710730
>
```

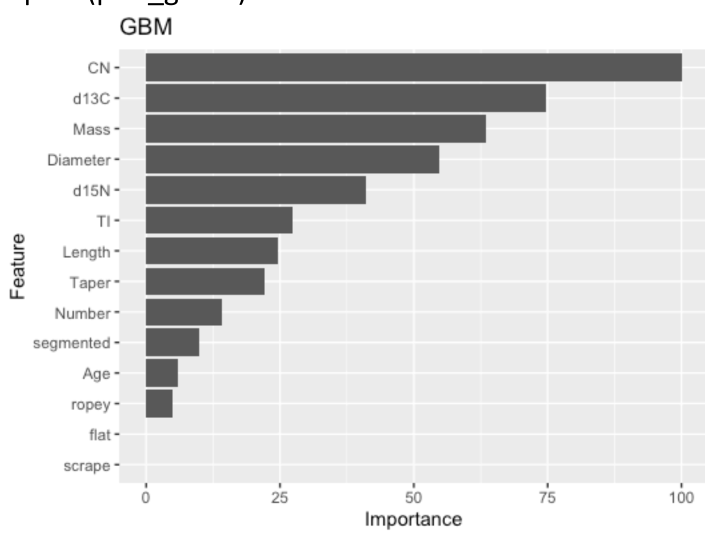
> # 7. Tune the GBM model using tune length = 20 and:
> # a) print the model summary and b) plot the models. (20 points)

```
> #using tune length
> fitControl <- trainControl(
+   method = "repeatedcv",
+   number = 5,
+   repeats = 5)
> ### Using tuneGrid ####
> modelLookup(model='gbm')
  model         parameter                   label forReg forClass probModel
```

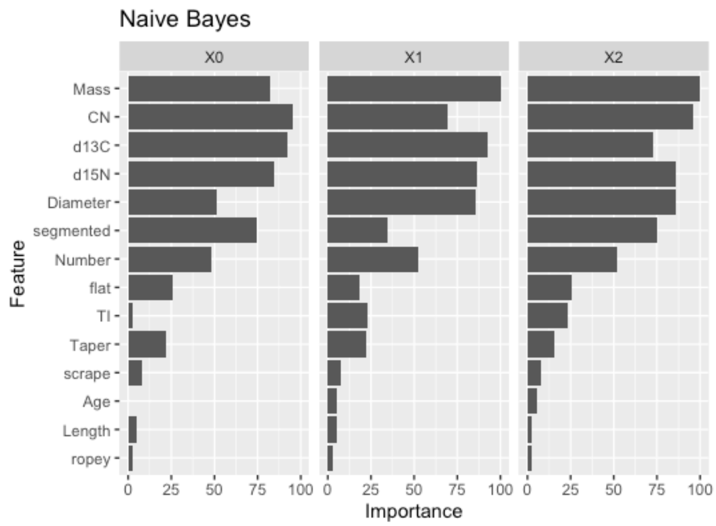
```
1   gbm          n.trees   # Boosting Iterations   TRUE     TRUE      TRUE
2   gbm interaction.depth         Max Tree Depth   TRUE     TRUE      TRUE
3   gbm        shrinkage              Shrinkage   TRUE     TRUE      TRUE
4   gbm    n.minobsinnode Min. Terminal Node Size   TRUE     TRUE      TRUE
> model_gbm2<-
train(trainSet[,predictors],trainSet[,outcomeName],method='gbm',trControl=fitControl,tuneLen
gth=20)
> # a) print the model summary
> print(model_gbm2)
Stochastic Gradient Boosting

83 samples
14 predictors
 3 classes: '0', '1', '2'

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

  interaction.depth  n.trees  Accuracy   Kappa
   1                  50     0.6576340  0.4311423
   1                 100     0.6417516  0.4060450
   1                 150     0.6124150  0.3621097
   1                 200     0.6171209  0.3680670
   1                 250     0.6100621  0.3549479
   1                 300     0.6077255  0.3583588
   1                 350     0.5980033  0.3379037
....
> #b) plot the models.
> plot(model_gbm2)
>
```



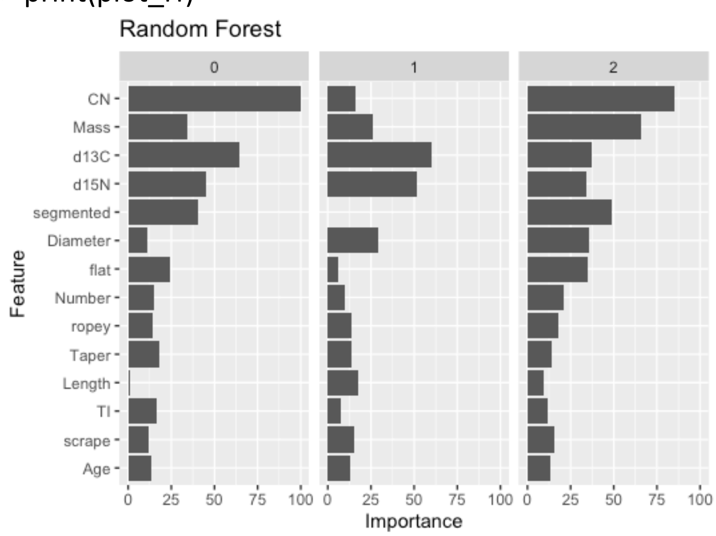
> **# 8. Using GGplot and gridExtra to plot all variable of**
> **# importance plots into one single plot. (10 points)**
> library(ggplot2)
> library(gridExtra)
> plot_gbm1 <- ggplot(data=varImp(object=model_gbm1)) + ggtitle("GBM")
> print(plot_gbm1)



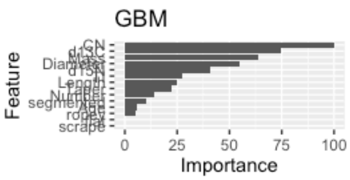
> plot_nb <- ggplot(data=varImp(object=model_nb)) + ggtitle("Naive Bayes")
> print(plot_nb)

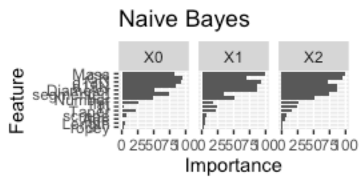


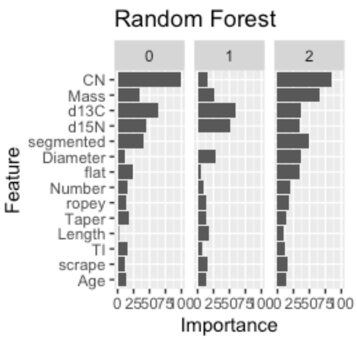

```
> plot_nnet <- ggplot(data=varImp(object=model_nnet)) + ggtitle("Neural Network")
Error in rep.int(factor(names(x), unique(names(x))), lengths(x)) :
  invalid 'times' value
> print(plot_nnet)
Error in print(plot_nnet) : object 'plot_nnet' not found
> plot_rf <- ggplot(data=varImp(object=model_rf)) + ggtitle("Random Forest")
> print(plot_rf)
```




```
> grid.arrange(plot_gbm1, plot_nb, plot_rf, nrow = 2, ncol=2, heights = c(0.35, 0.65))
```







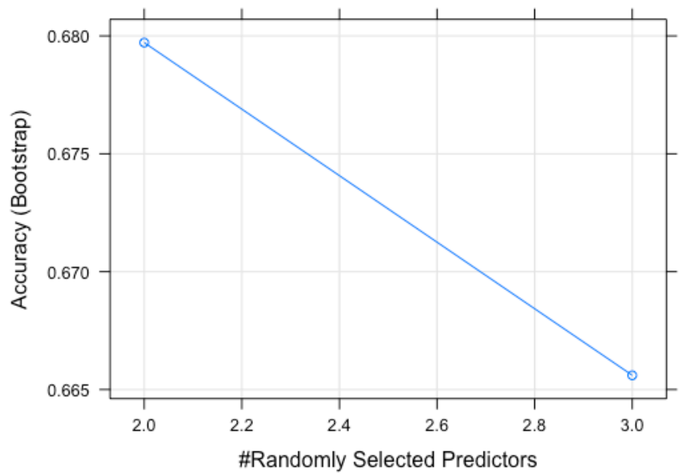
 # 9. Which model performs the best? and why do you think this is the case?
 # Can we accurately predict species on this dataset? (10 points)

 # Answer -
As per accuracy & Kappa values of the models, Neural Network model performs the best.
ExperimentName   Accuracy   Kappa
2   Neural Net 0.6943589   0.5100720
1 Random Forest 0.6479826   0.4170319
3   Naive Bayes 0.6461044   0.4180012
4   GBM 0.6155064   0.3710730
Neural Network model uses memory to store previous read values and
does classification based on correlation between the read values.
Whereas, other models like Naive Bayes considers attributes are independent of each other.
Species from the data set can be predicted with less than 70 % accuracy
i.e. 69.4% using Neural Networks model, 64.8% using Random Forest model can be accurately predicted.

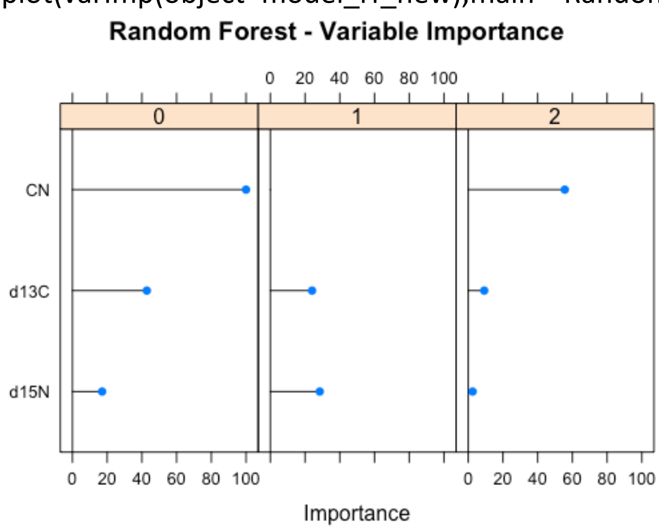
 # 10. Graduate Student questions:
 # a. Using feature selection with rfe in caret and the repeatedcv method: Find the top 3
 # predictors and build the same models as in 6 and 8 with the same parameters. (20 points)

```
> #Feature selection using rfe in caret
> control <- rfeControl(functions = rfFuncs,
+                     method = "repeatedcv",
+                     repeats = 3,
+                     verbose = FALSE)
> predictors<-names(trainSet)[!names(trainSet) %in% outcomeName]
> Species_Pred <- rfe(trainSet[,predictors], trainSet[,outcomeName],rfeControl = control)
> Species_Pred
```

```
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   0.7012 0.4856     0.1523  0.2653        *
         8   0.6927 0.4629     0.1643  0.2865
        14   0.6870 0.4461     0.1518  0.2794

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



> #Taking only the top 3 predictors
> predictorsTop3<-c("CN", "d13C", "d15N")
> # ######## randomforest  with top-3 selected features ###########
> model_rf_new<-train(trainSet[,predictorsTop3],trainSet[,outcomeName],method='rf',
importance=T)
note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .

> # summarizing the model
> print(model_rf_new)
Random Forest

83 samples
 3 predictor
 3 classes: '0', '1', '2'

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     0.6797157  0.4496038
  3     0.6656016  0.4308129

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
> # Visualizing the models
> plot(model_rf_new)
```




```
> #Variable Importance
> varImp(object=model_rf_new)
rf variable importance

  variables are sorted by maximum importance across the classes
        0     1      2
CN   100.00  0.00 55.615
d13C  42.83 24.03  9.208
d15N  17.08 28.46  2.506
> #Plotting Varianle importance for Random Forest
> plot(varImp(object=model_rf_new),main="Random Forest - Variable Importance")
```



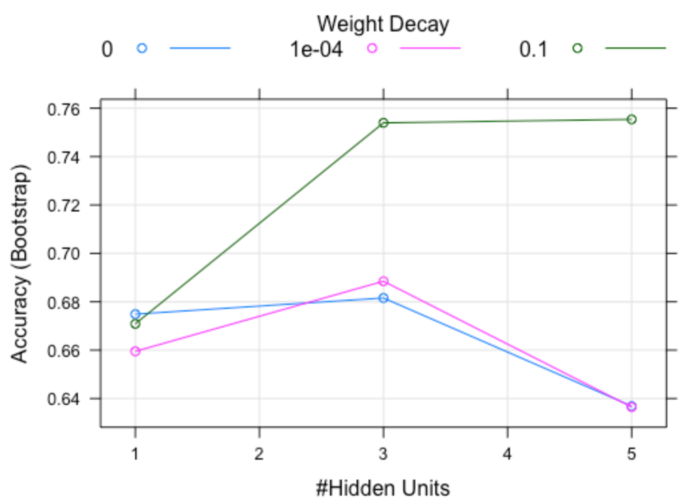

```
> #Predictions
> predictions_rf_new<-
predict.train(object=model_rf_new,testSet[,predictorsTop3],type="raw")
> table(predictions_rf_new)
predictions_rf_new
 0  1  2
```

```
15  6  6
> #Confusion Matrix and Statistics
> confusionMatrix(predictions_rf_new,testSet[,outcomeName])
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2
         0 11  1  3
         1  1  5  0
         2  2  1  3

Overall Statistics

               Accuracy : 0.7037
                 95% CI : (0.4982, 0.8625)
    No Information Rate : 0.5185
    P-Value [Acc > NIR] : 0.04012

                  Kappa : 0.5102

 Mcnemar's Test P-Value : 0.75300

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity            0.7857   0.7143   0.5000
Specificity            0.6923   0.9500   0.8571
Pos Pred Value         0.7333   0.8333   0.5000
Neg Pred Value         0.7500   0.9048   0.8571
Prevalence             0.5185   0.2593   0.2222
Detection Rate         0.4074   0.1852   0.1111
Detection Prevalence   0.5556   0.2222   0.2222
Balanced Accuracy      0.7390   0.8321   0.6786
> # ####### Neural Net with top-3 selected features ###########
> model_nnet_new<-train(trainSet[,predictorsTop3],trainSet[,outcomeName],method='nnet',
importance=T)
> # summarizing the model
> print(model_nnet_new)
Neural Network

83 samples
 3 predictor
 3 classes: '0', '1', '2'
```

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	0.6748811	0.4172198
1	1e-04	0.6594939	0.3943782
1	1e-01	0.6708776	0.4121009
3	0e+00	0.6815514	0.4592315
3	1e-04	0.6884762	0.4660593
3	1e-01	0.7539512	0.5732017
5	0e+00	0.6368239	0.3989197
5	1e-04	0.6364624	0.3947993
5	1e-01	0.7553847	0.5772432

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.
> # Visualizing the models
> plot(model_nnet_new)



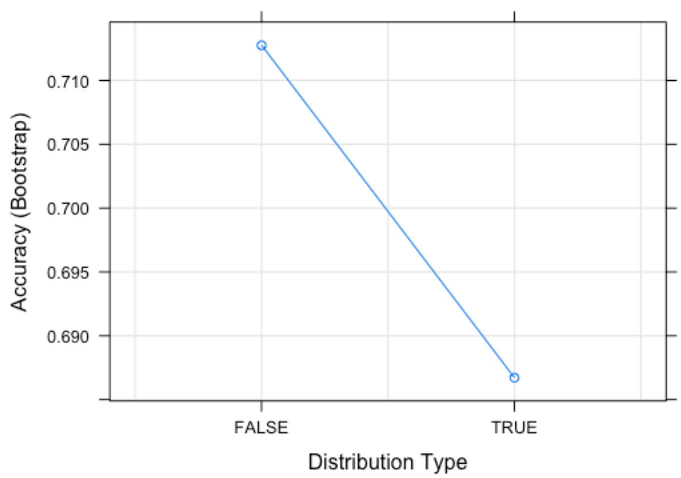
> #Plotting Variable importance for Neural Net
> plot(varImp(object=model_nnet_new),main="Neural Net - Variable Importance")
Error in rep.int(factor(names(x), unique(names(x))), lengths(x)) :
invalid 'times' value
> #Predictions
> predictions_nnet_new<-
predict.train(object=model_nnet_new,testSet[,predictorsTop3],type="raw")
> table(predictions_nnet_new)
predictions_nnet_new
0 1 2

```
17  5  5
> #Confusion Matrix and Statistics
> confusionMatrix(predictions_nnet_new,testSet[,outcomeName])
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2
         0 13  1  3
         1  1  4  0
         2  0  2  3

Overall Statistics

               Accuracy : 0.7407
                 95% CI : (0.5372, 0.8889)
    No Information Rate : 0.5185
    P-Value [Acc > NIR] : 0.01571

                  Kappa : 0.5563

 Mcnemar's Test P-Value : 0.17180

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity            0.9286   0.5714   0.5000
Specificity            0.6923   0.9500   0.9048
Pos Pred Value         0.7647   0.8000   0.6000
Neg Pred Value         0.9000   0.8636   0.8636
Prevalence             0.5185   0.2593   0.2222
Detection Rate         0.4815   0.1481   0.1111
Detection Prevalence   0.6296   0.1852   0.1852
Balanced Accuracy      0.8104   0.7607   0.7024
>> # ############# Naive Bayes with top-3 selected features ###########
> model_nb_new<-
train(trainSet[,predictorsTop3],trainSet[,outcomeName],method='naive_bayes', importance=T)
There were 50 or more warnings (use warnings() to see the first 50)
> # summarizing the model
> print(model_nb_new)
Naive Bayes

83 samples
 3 predictor
 3 classes: '0', '1', '2'
```

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      0.7127563  0.4979608
   TRUE      0.6867141  0.4609503
```

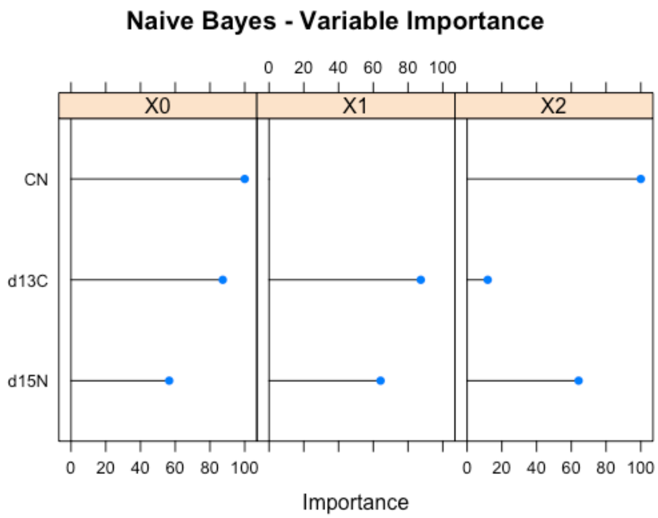
Tuning parameter 'laplace' was held constant at a value of 0
Tuning parameter 'adjust'
 was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were laplace = 0, usekernel = FALSE and adjust = 1.
> # Visualizing the models
> plot(model_nb_new)



> varImp(object=model_nb_new)
ROC curve variable importance

```
  variables are sorted by maximum importance across the classes
       X0    X1     X2
CN   100.00  0.00 100.00
d13C  87.36 87.36  11.87
d15N  56.52 64.20  64.20
```

> #Plotting Variable importance for Naive Bayes
> plot(varImp(object=model_nb_new),main="Naive Bayes - Variable Importance")



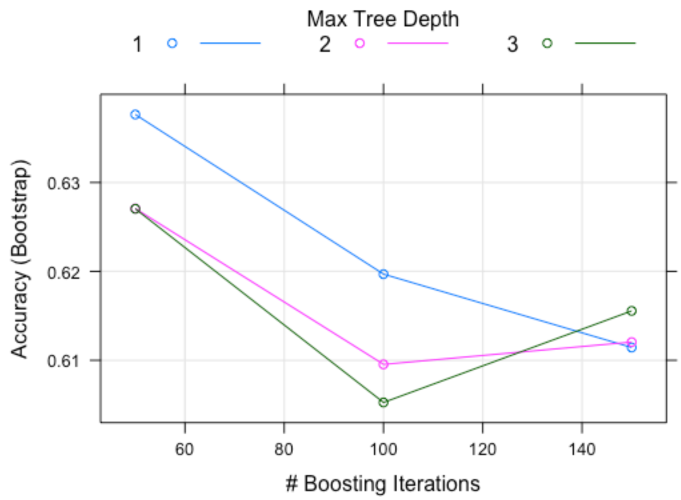

```
> #Predictions
> predictions_nb_new<-
predict.train(object=model_nb_new,testSet[,predictorsTop3],type="raw")
> table(predictions_nb_new)
predictions_nb_new
 0  1  2
18  5  4
> #Confusion Matrix and Statistics
> confusionMatrix(predictions_nb_new,testSet[,outcomeName])
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2
         0 14  1  3
         1  0  5  0
         2  0  1  3

Overall Statistics

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

                  Kappa : 0.677

 Mcnemar's Test P-Value : 0.171797

Statistics by Class:
```

	Class: 0	Class: 1	Class: 2
Sensitivity	1.0000	0.7143	0.5000
Specificity	0.6923	1.0000	0.9524
Pos Pred Value	0.7778	1.0000	0.7500
Neg Pred Value	1.0000	0.9091	0.8696
Prevalence	0.5185	0.2593	0.2222
Detection Rate	0.5185	0.1852	0.1111
Detection Prevalence	0.6667	0.1852	0.1481
Balanced Accuracy	0.8462	0.8571	0.7262

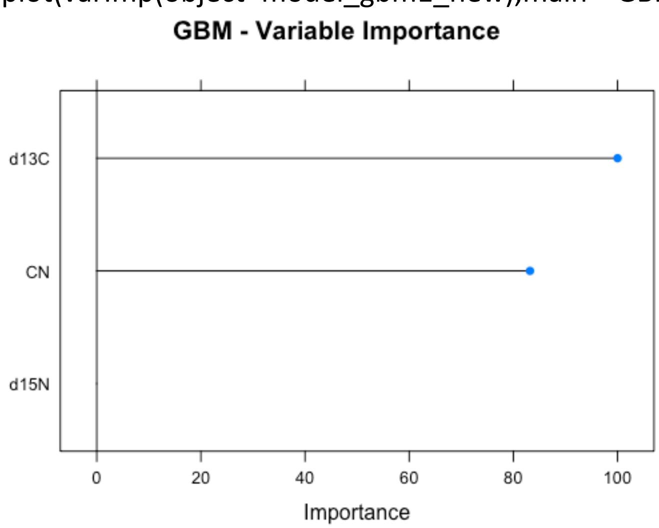
```
> # ############### GBM with top-3 selected features ###########
> model_gbm1_new<-train(trainSet[,predictorsTop3],trainSet[,outcomeName],method='gbm')
> # summarizing the model
> print(model_gbm1_new)
Stochastic Gradient Boosting

83 samples
 3 predictor
 3 classes: '0', '1', '2'

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	0.6376478	0.4006285
1	100	0.6196917	0.3705796
1	150	0.6114344	0.3551965
2	50	0.6270669	0.3856325
2	100	0.6095393	0.3571104
2	150	0.6120484	0.3609020
3	50	0.6270374	0.3859090
3	100	0.6052552	0.3501068
3	150	0.6155551	0.3654858

```
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.
> # Visualizing the models
> plot(model_gbm1_new)
```




```
> #Variable Importance
> varImp(object=model_gbm1_new)
gbm variable importance

     Overall
d13C  100.00
CN     83.19
d15N    0.00
> #Plotting Variable importance for GBM
> plot(varImp(object=model_gbm1_new),main="GBM - Variable Importance")
```




```
> #Predictions
> predictions_gbm_new<-
predict.train(object=model_gbm1_new,testSet[,predictorsTop3],type="raw")
> table(predictions_gbm_new)
predictions_gbm_new
 0  1  2
17  7  3
```

```
> #Confusion Matrix and Statistics
> confusionMatrix(predictions_gbm_new,testSet[,outcomeName])
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2
         0 13  1  3
         1  0  5  2
         2  1  1  1

Overall Statistics

               Accuracy : 0.7037
                 95% CI : (0.4982, 0.8625)
    No Information Rate : 0.5185
    P-Value [Acc > NIR] : 0.04012

                  Kappa : 0.4906

 Mcnemar's Test P-Value : 0.50617

Statistics by Class:
```

	Class: 0	Class: 1	Class: 2
Sensitivity	0.9286	0.7143	0.16667
Specificity	0.6923	0.9000	0.90476
Pos Pred Value	0.7647	0.7143	0.33333
Neg Pred Value	0.9000	0.9000	0.79167
Prevalence	0.5185	0.2593	0.22222
Detection Rate	0.4815	0.1852	0.03704
Detection Prevalence	0.6296	0.2593	0.11111
Balanced Accuracy	0.8104	0.8071	0.53571

 # 10. b. Create a dataframe that compares the non-feature selected models (the same as on 7)
 # and add the best BEST performing models of each (randomforest, neural net, naive bayes and gbm) and
 # display the data frame that has the following columns: ExperimentName, accuracy, kappa.
 # Sort the data frame by accuracy. (40 points)

```
> experimentName_new<-c("Random Forest New" , "Neural Net New", "Naive Bayes New", "GBM New")
> accuracyDetails_new<-c(max(model_rf_new$results$Accuracy),
max(model_nnet_new$results$Accuracy),
```

```
+              max(model_nb_new$results$Accuracy),
max(model_gbm1_new$results$Accuracy))
> kappaDetails_new<-c(max(model_rf_new$results$Kappa),
max(model_nnet_new$results$Kappa),
+            max(model_nb_new$results$Kappa), max(model_gbm1_new$results$Kappa))
> bestModelDf_new<-data.frame(ExperimentName=c(experimentName,
experimentName_new),
+                    Accuracy=c(accuracyDetails, accuracyDetails_new),
+                    Kappa=c(kappaDetails, kappaDetails_new))
> print(bestModelDf_new[order(-bestModelDf_new$Accuracy),])
   ExperimentName       Accuracy     Kappa
6   Neural Net New      0.7553847    0.5772432
7  Naive Bayes New      0.7127563    0.4979608
2      Neural Net       0.6943589    0.5100720
5 Random Forest New     0.6797157    0.4496038
1    Random Forest      0.6479826    0.4170319
3     Naive Bayes       0.6461044    0.4180012
8        GBM New        0.6376478    0.4006285
4          GBM          0.6155064    0.3710730
```

 # NOTE::: "New" refers to the latest models with Top 3 predictors in
 # "Random Forest New" , "Neural Net New", "Naive Bayes New", "GBM New".

 # 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)

 # Answer -
Output of Best Model dataframe :::

```
#    ExperimentName            Accuracy    Kappa
# 6   Neural Net New           0.7522766   0.5693890
# 7  Naive Bayes New           0.7254481   0.5135670
# 2      Neural Net            0.6943589   0.5100720
# 5 Random Forest New          0.6861781   0.4607807
# 1    Random Forest           0.6479826   0.4170319
# 3     Naive Bayes            0.6461044   0.4180012
# 8        GBM New             0.6352902   0.3809075
# 4          GBM               0.6155064   0.3710730
```

From the Best model printed above with based on accuracy, it is evident that Neural Network New
(with top 3 predictors) performed the best with an accuracy of 75.22% and Kappa value of 56.9%.
Followed by Naive Bayes(with top 3 predictors) and the old Neural network model (with non-feature selected)

with accuracies of 72.54 & 69.43 % respectively.
With the new Neural Network model with top 3 selected features, prediction accuracy increased from 69.4 % to 75.22 %.
Prediction using this model is improved.
Yes we can predict Species from the dataset using Neural Network New model with 75.22% accuracy.