Homework 5 Notebook - Henry Woodyard

Code

Run this only if the machine learning packages have not yet been installed.

Hide

```
#install.packages("caret")
#install.packages("randomForest")
#install.packages("RANN")
#install.packages("gbm")
```

Loading packages and data

Hide

```
library(caret)
```

```
Loading required package: lattice
Loading required package: ggplot2
```

Hide

```
library(RANN)
library(randomForest)
```

```
randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 惏牷randomForest惏怍

The following object is masked from 惏牷package:ggplot2惏怍:

    margin
```

Hide

```
library(ggplot2)
library(gridExtra)
```

```
Attaching package: 惣梅gridExtra惣怍

The following object is masked from 惣梅package:randomForest惣怍:

    combine
```

Hide

```
library(gbm)
```

```
Loaded gbm 2.1.5
```

Hide

```
data(scat)
```

Question 1: Set the Species column as the target/outcome and convert it to numeric. (5 points)

Hide

```
scat$Species <- as.factor(scat$Species)
```

Question 2: Remove the Month, Year, Site, Location features. (5 points)

Hide

```
removal <- c("Month", "Year", "Site", "Location")
scat <- scat[,-which(colnames(scat) %in% removal)]; rm(removal)
```

3. Check if any values are null. If there are, impute missing values using KNN. (10 points)

Hide

```
print("Missing values before imputing")
```

```
[1] "Missing values before imputing"
```

Hide

```
missing <- colSums(is.na(scat))
cols_with_missing <- which(missing > 0)
cols_to_impute <- scat[,cols_with_missing]
print(missing)
```

```
  Species       Age    Number    Length  Diameter     Taper        TI 
        0         0         0         0         6        17        17 
     Mass      d13C      d15N        CN     ropey segmented      flat 
        1         2         2         2         0         0         0 
   scrape 
        0 
```

Hide

```
imputeValues <- preProcess(cols_to_impute, method = c("knnImpute", "center", "scale"))
scat[,cols_with_missing] <- predict(imputeValues, scat[, cols_with_missing])

print("Missing values after imputing")
```

```
[1] "Missing values after imputing"
```

Hide

```
colSums(is.na(scat))
```

```
  Species       Age    Number    Length  Diameter     Taper        TI 
        0         0         0         0         0         0         0 
     Mass      d13C      d15N        CN     ropey segmented      flat 
        0         0         0         0         0         0         0 
   scrape 
        0 
```

Hide

```
rm(imputeValues, cols_to_impute, cols_with_missing, missing)
```

Hide

```
imputeValues <- preProcess(scat, method = c("knnImpute", "center", "scale"))
scat <- predict(imputeValues, scat)
```

4. Converting every categorical variable to numerical (if needed). (5 points)

Hide

```
str(scat)
```

```
'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  1.893 1.8141 0.0775 -0.1067 0.5774 ...
 $ Taper    : num  1 0.659 -0.804 -0.222 -0.549 ...
 $ TI       : num  0.0352 -0.1475 -0.771 -0.255 -0.6742 ...
 $ Mass     : num  0.396 0.59 -0.453 -0.566 1.478 ...
 $ d13C     : num  0.00637 -1.27798 -0.86532 3.15002 1.6802 ...
 $ d15N     : num  -0.16 0.819 0.368 -0.544 -0.137 ...
 $ CN       : num  0.0295 0.7988 -0.0805 0.8538 0.6065 ...
 $ 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 ...
```

Hide

```
# All variables are numeric or integer. Conversion not needed.
```

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)

Set random seed to 100 and partition the data.

Hide

```
set.seed(100)
index <- createDataPartition(scat$Species, p = .75, list = FALSE)
trainSet <- scat[index,]
testSet <- scat[-index,]
rm(index)
```

Save the names of our outcome and predictor variables for easy reference.

Hide

```
outcomeName <- "Species"
predictors <- names(scat)[names(scat) != outcomeName]
```

Train a random forest model.

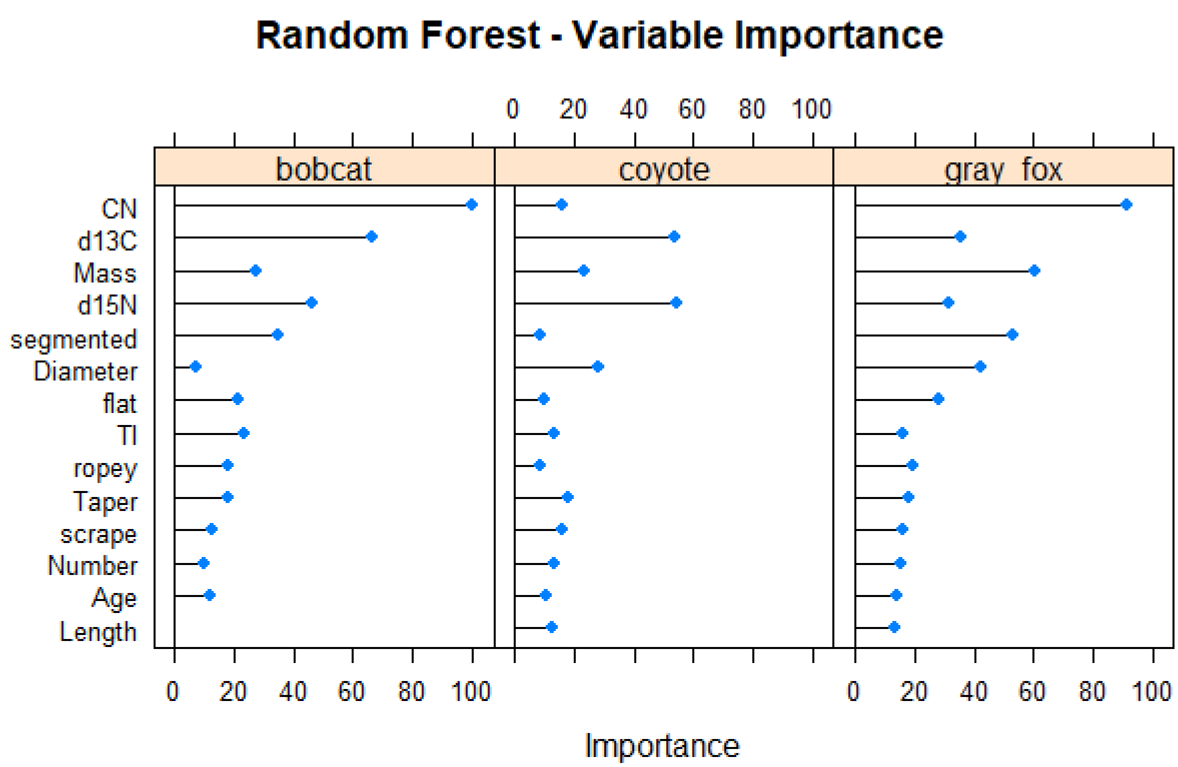
Hide

```
model_rf <- train(x = trainSet[,predictors],
                  y = trainSet[,outcomeName],
                  method = 'rf',
                  importance = T,
                  verbose = F)
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    0.6449190  0.3884346
   8    0.6664501  0.4459848
  14    0.6659344  0.4489932

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

Hide

```
plot(varImp(object = model_rf), main = "Random Forest - Variable Importance")
```



Hide

```
predict_rf <- predict.train(model_rf, testSet[,predictors])
results_rf <- confusionMatrix(predict_rf, testSet[,outcomeName])
print(results_rf)
```

```
Confusion Matrix and Statistics

          Reference
Prediction bobcat coyote gray_fox
  bobcat       14      2        3
  coyote        0      5        0
  gray_fox      0      0        3

Overall Statistics

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

                  Kappa : 0.6707

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: bobcat Class: coyote Class: gray_fox
Sensitivity                 1.0000        0.7143          0.5000
Specificity                 0.6154        1.0000          1.0000
Pos Pred Value              0.7368        1.0000          1.0000
Neg Pred Value              1.0000        0.9091          0.8750
Prevalence                  0.5185        0.2593          0.2222
Detection Rate              0.5185        0.1852          0.1111
Detection Prevalence        0.7037        0.1852          0.1111
Balanced Accuracy           0.8077        0.8571          0.7500
```

Train a neural network.

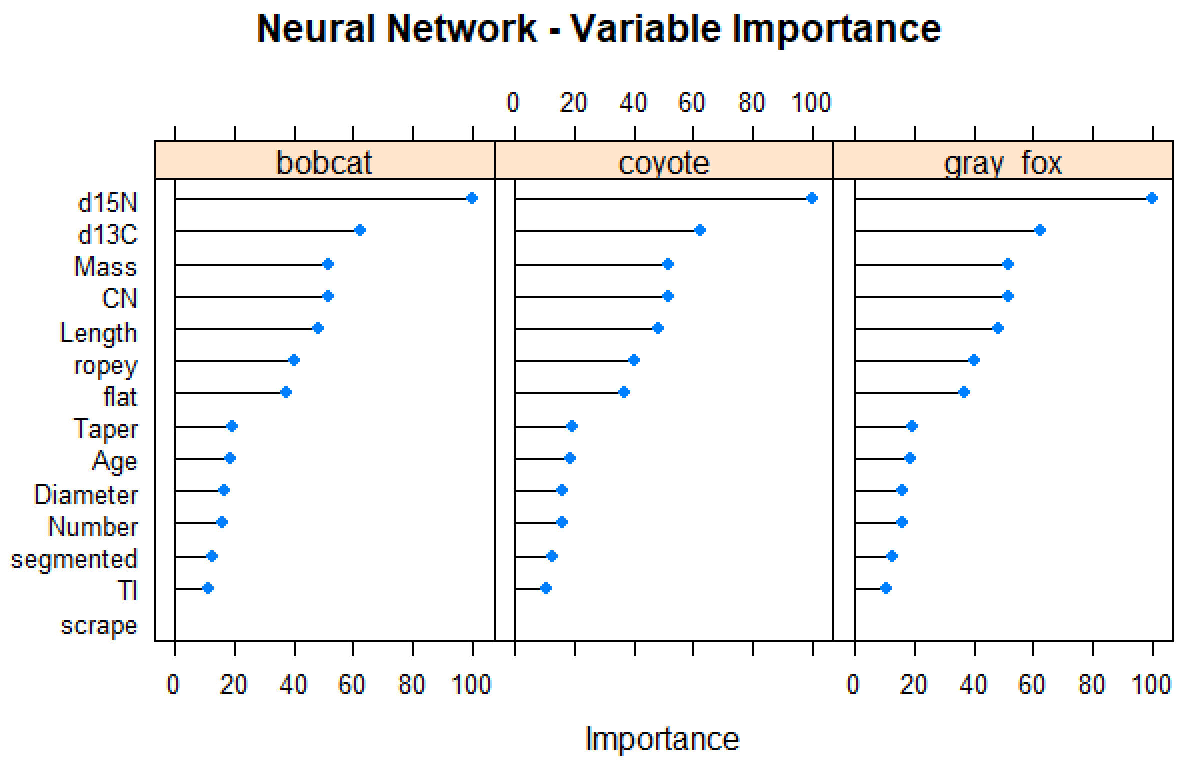
Hide

```
model_nn <- train(x = trainSet[,predictors],
                  y = trainSet[,outcomeName],
                  method = 'nnet',
                  importance = T,
                  trace = F)
print(model_nn)
```

```
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  0.5602687  0.2971229
  1     1e-04  0.5277275  0.2489196
  1     1e-01  0.6037359  0.3411895
  3     0e+00  0.6391312  0.4188356
  3     1e-04  0.6401082  0.4230570
  3     1e-01  0.6635934  0.4478497
  5     0e+00  0.6648931  0.4542429
  5     1e-04  0.6598657  0.4478332
  5     1e-01  0.6654134  0.4527684

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.
```

Hide

```
# NN importance is reported by default in a way that throws an error. Converting to a dataframe
 and removing the "overall" column fixes this.
imp <- varImp(model_nn)
imp$importance <- as.data.frame(imp$importance)[,-1]
plot(imp, main = "Neural Network - Variable Importance")
```



Hide

```
predict_nn <- predict.train(model_nn, testSet[,predictors])
results_nn <- confusionMatrix(predict_nn, testSet[,outcomeName])
print(results_nn)
```

```
Confusion Matrix and Statistics

          Reference
Prediction bobcat coyote gray_fox
  bobcat       13      0        0
  coyote        1      5        1
  gray_fox      0      2        5

Overall Statistics

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

                  Kappa : 0.7632

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: bobcat Class: coyote Class: gray_fox
Sensitivity                 0.9286        0.7143          0.8333
Specificity                 1.0000        0.9000          0.9048
Pos Pred Value              1.0000        0.7143          0.7143
Neg Pred Value              0.9286        0.9000          0.9500
Prevalence                  0.5185        0.2593          0.2222
Detection Rate              0.4815        0.1852          0.1852
Detection Prevalence        0.4815        0.2593          0.2593
Balanced Accuracy           0.9643        0.8071          0.8690
```

Train a naive bayes model.

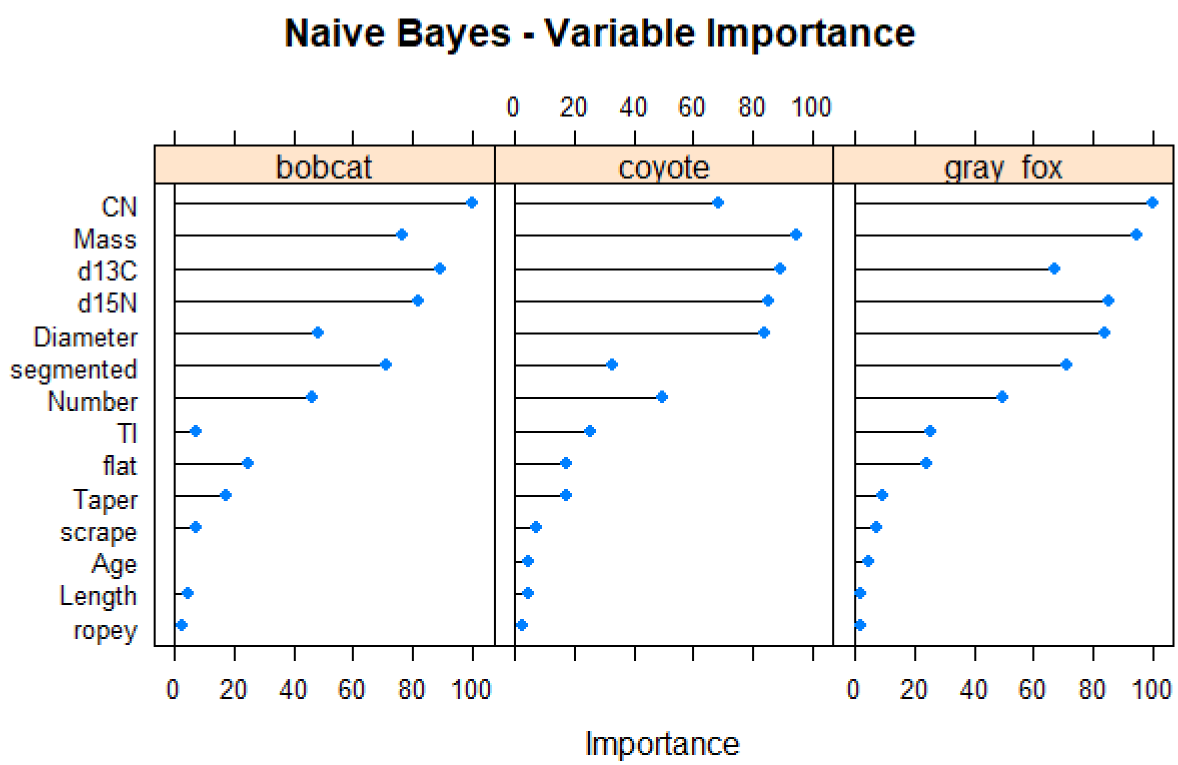
Hide

```
model_nb <- train(x = trainSet[,predictors],
                  y = trainSet[,outcomeName],
                  method = 'naive_bayes')
print(model_nb)
```

```
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      0.6184025  0.4186515
   TRUE      0.6628904  0.4514779

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.
```

Hide

```
plot(varImp(object = model_nb), main = "Naive Bayes - Variable Importance")
```



Hide

```
predict_nb <- predict.train(model_nb, testSet[,predictors])
results_nb <- confusionMatrix(predict_nb, testSet[,outcomeName])
print(results_nb)
```

```
Confusion Matrix and Statistics

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

Overall Statistics

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

                  Kappa : 0.6831

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: bobcat Class: coyote Class: gray_fox
Sensitivity                 0.9286        0.7143          0.6667
Specificity                 0.6923        1.0000          0.9524
Pos Pred Value              0.7647        1.0000          0.8000
Neg Pred Value              0.9000        0.9091          0.9091
Prevalence                  0.5185        0.2593          0.2222
Detection Rate              0.4815        0.1852          0.1481
Detection Prevalence        0.6296        0.1852          0.1852
Balanced Accuracy           0.8104        0.8571          0.8095
```

Train a Gradient Boosting Machines (GBM) model.

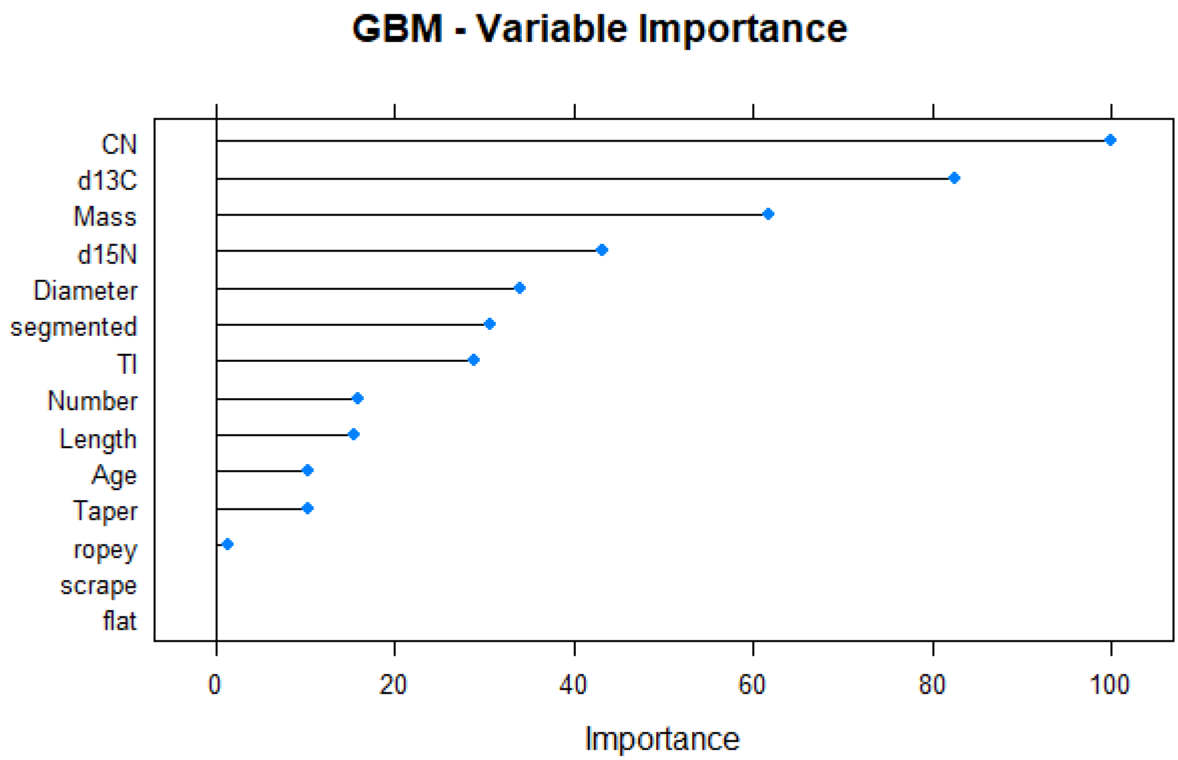
Hide

```
model_gbm <- train(x = trainSet[,predictors],
                  y = trainSet[,outcomeName],
                  method = 'gbm',
                  verbose = F)
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      0.6543485  0.4205283
  1                  100      0.6464551  0.4090252
  1                  150      0.6289119  0.3801202
  2                   50      0.6452228  0.4040488
  2                  100      0.6331374  0.3859193
  2                  150      0.6346581  0.3899366
  3                   50      0.6232588  0.3699732
  3                  100      0.6188623  0.3618511
  3                  150      0.6154565  0.3538589

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.
```

Hide

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



Hide

```
predict_gbm <- predict.train(model_gbm, testSet[,predictors])
results_gbm <- confusionMatrix(predict_gbm, testSet[,outcomeName])
print(results_gbm)
```

```
Confusion Matrix and Statistics

          Reference
Prediction bobcat coyote gray_fox
  bobcat       14      1        2
  coyote        0      5        1
  gray_fox      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 : 0.6824

 Mcnemar's Test P-Value : 0.391625

Statistics by Class:

                     Class: bobcat Class: coyote Class: gray_fox
Sensitivity                 1.0000        0.7143          0.5000
Specificity                 0.7692        0.9500          0.9524
Pos Pred Value              0.8235        0.8333          0.7500
Neg Pred Value              1.0000        0.9048          0.8696
Prevalence                  0.5185        0.2593          0.2222
Detection Rate              0.5185        0.1852          0.1111
Detection Prevalence        0.6296        0.2222          0.1481
Balanced Accuracy           0.8846        0.8321          0.7262
```

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. (15 points)

Note: I'm a little confused about your phrasing here. You say for the "best performing models of each", but when predicting, the model already uses the best parameters (i.e. bestTune). Perhaps you wanted us to get the accuracy from the model directly, rather than from the prediction; however, it seems more robust to use the accuracy from the testing and thus I'm using that.

Using the results from the confusion matrices earlier, I take the accuracy and kappa for each model and combine them into a dataframe.

Hide

```
results_all <- as.data.frame(rbind(results_nn$overall[c("Accuracy", "Kappa")],
                             results_nb$overall[c("Accuracy", "Kappa")],
                             results_rf$overall[c("Accuracy", "Kappa")],
                             results_gbm$overall[c("Accuracy", "Kappa")]))
results_all <- cbind(c("Neural Net", "Naive Bayes", "Random Forest", "GBM"),
                     results_all)
colnames(results_all)[1] <- "ExperimentName"
print(results_all[order(results_all$Accuracy, decreasing = T),])
```

	ExperimentName <fctr>	Accuracy <dbl>	Kappa <dbl>
1	Neural Net	0.8518519	0.7631579
2	Naive Bayes	0.8148148	0.6830986
3	Random Forest	0.8148148	0.6707317
4	GBM	0.8148148	0.6823529

4 rows

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

This gives warning for zero variance in the scrape variable. I assume this is because some random subsamples result in scrape being constant. Annoying to see but empirically fine.

Hide

```
model_gbm_tuned <- train(x = trainSet[,predictors],
                         y = trainSet[,outcomeName],
                         method = 'gbm',
                         tuneLength = 20,
                         verbose = F)
```

```
variable 14: scrape has no variation.variable 14: scrape has no variation.variable 14: scrape ha
s no variation.variable 14: scrape has no variation.variable 14: scrape has no variation.variabl
e 14: scrape has no variation.variable 14: scrape has no variation.variable 14: scrape has no va
riation.variable 14: scrape has no variation.variable 14: scrape has no variation.variable 14: s
crape has no variation.variable 14: scrape has no variation.variable 14: scrape has no variatio
n.variable 14: scrape has no variation.variable 14: scrape has no variation.variable 14: scrape
has no variation.variable 14: scrape has no variation.variable 14: scrape has no variation.varia
ble 14: scrape has no variation.variable 14: scrape has no variation.
```

Hide

```
print(model_gbm_tuned)
```

```
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     0.6544860  0.4329368
   1                  100     0.6478794  0.4240464
   1                  150     0.6413565  0.4153538
   1                  200     0.6281235  0.3904565
   1                  250     0.6283308  0.3922084
   1                  300     0.6215263  0.3802714
   1                  350     0.6297663  0.3940760
   1                  400     0.6315559  0.3985401
   1                  450     0.6268682  0.3909514
   1                  500     0.6239534  0.3871706
   1                  550     0.6273703  0.3903794
   1                  600     0.6164212  0.3733550
   1                  650     0.6150046  0.3720943
   1                  700     0.6164946  0.3722923
   1                  750     0.6191175  0.3771790
   1                  800     0.6189124  0.3772179
   1                  850     0.6205399  0.3789572
   1                  900     0.6160672  0.3719492
   1                  950     0.6190342  0.3776456
   1                 1000     0.6158465  0.3731862
   2                   50     0.6455649  0.4155581
   2                  100     0.6418848  0.4140376
   2                  150     0.6330193  0.4012279
   2                  200     0.6190148  0.3826711
   2                  250     0.6161080  0.3753404
   2                  300     0.6190009  0.3812209
   2                  350     0.6108654  0.3679395
   2                  400     0.6178837  0.3804358
   2                  450     0.6203581  0.3827920
   2                  500     0.6177338  0.3800135
   2                  550     0.6162983  0.3790783
   2                  600     0.6174156  0.3815298
   2                  650     0.6175754  0.3814272
   2                  700     0.6092720  0.3680430
   2                  750     0.6094099  0.3681720
   2                  800     0.6081139  0.3675109
   2                  850     0.6029220  0.3592596
   2                  900     0.6066083  0.3638201
   2                  950     0.6065250  0.3642037
   2                 1000     0.6068541  0.3644228
   3                   50     0.6464910  0.4197312
```

```
 3       100   0.6240793  0.3881133
 3       150   0.6196791  0.3790104
 3       200   0.6091445  0.3607727
 3       250   0.6216297  0.3840099
 3       300   0.6213963  0.3831918
 3       350   0.6159956  0.3737283
 3       400   0.6146225  0.3695983
 3       450   0.6188581  0.3766414
 3       500   0.6130553  0.3699184
 3       550   0.6148459  0.3722637
 3       600   0.6107825  0.3681002
 3       650   0.6134390  0.3712089
 3       700   0.6041187  0.3569594
 3       750   0.6085012  0.3643280
 3       800   0.6097026  0.3651829
 3       850   0.6125102  0.3690260
 3       900   0.6105568  0.3648996
 3       950   0.6036994  0.3562760
 3      1000   0.6076620  0.3619111
 4        50   0.6410578  0.4091416
 4       100   0.6252401  0.3865402
 4       150   0.6191769  0.3760532
 4       200   0.6225311  0.3827756
 4       250   0.6235127  0.3848849
 4       300   0.6174631  0.3737045
 4       350   0.6162829  0.3729554
 4       400   0.6191867  0.3761575
 4       450   0.6125574  0.3681416
 4       500   0.6142012  0.3697225
 4       550   0.6171620  0.3743705
 4       600   0.6127649  0.3669362
 4       650   0.6056960  0.3555083
 4       700   0.6081169  0.3616810
 4       750   0.6098317  0.3644140
 4       800   0.6110671  0.3677982
 4       850   0.6086066  0.3643343
 4       900   0.6067138  0.3609209
 4       950   0.6093179  0.3624806
 4      1000   0.6067534  0.3592581
 5        50   0.6496160  0.4272039
 5       100   0.6456973  0.4213646
 5       150   0.6288253  0.3943930
 5       200   0.6221745  0.3850672
 5       250   0.6158191  0.3752162
 5       300   0.6145724  0.3735270
 5       350   0.6044535  0.3558580
 5       400   0.6056223  0.3552434
 5       450   0.6028144  0.3503889
 5       500   0.6031820  0.3500628
 5       550   0.5949396  0.3380624
 5       600   0.5932311  0.3360037
 5       650   0.5894256  0.3309657
 5       700   0.5985433  0.3474177
 5       750   0.5933047  0.3393025
```

5	800	0.5884997	0.3289923
5	850	0.5887705	0.3301695
5	900	0.5913881	0.3332751
5	950	0.5941500	0.3368543
5	1000	0.5900106	0.3304351
6	50	0.6482430	0.4250268
6	100	0.6493792	0.4281636
6	150	0.6298055	0.3948456
6	200	0.6278745	0.3913656
6	250	0.6221715	0.3816637
6	300	0.6141558	0.3701310
6	350	0.6119318	0.3687588
6	400	0.6119612	0.3677011
6	450	0.6060115	0.3541150
6	500	0.6032420	0.3500434
6	550	0.6035679	0.3507564
6	600	0.5980841	0.3434402
6	650	0.6009725	0.3472014
6	700	0.6006255	0.3478253
6	750	0.5965914	0.3421960
6	800	0.5922206	0.3358904
6	850	0.5972707	0.3429697
6	900	0.5923434	0.3373253
6	950	0.5992015	0.3475415
6	1000	0.5951394	0.3405009
7	50	0.6365216	0.4058972
7	100	0.6258954	0.3898293
7	150	0.6246196	0.3893335
7	200	0.6177830	0.3788253
7	250	0.6164364	0.3778553
7	300	0.6118015	0.3687223
7	350	0.6197601	0.3835367
7	400	0.6174874	0.3801508
7	450	0.6051990	0.3627360
7	500	0.6034718	0.3599242
7	550	0.6061245	0.3643098
7	600	0.6021996	0.3575742
7	650	0.5995283	0.3531244
7	700	0.5983730	0.3516067
7	750	0.6007783	0.3550354
7	800	0.5971098	0.3493776
7	850	0.5981630	0.3512029
7	900	0.5970156	0.3481119
7	950	0.5968664	0.3484390
7	1000	0.5911741	0.3396673
8	50	0.6534403	0.4335198
8	100	0.6375952	0.4111208
8	150	0.6320311	0.4030418
8	200	0.6228513	0.3890996
8	250	0.6291210	0.3974176
8	300	0.6296007	0.3962627
8	350	0.6231121	0.3852976
8	400	0.6102759	0.3667793
8	450	0.6124781	0.3708278

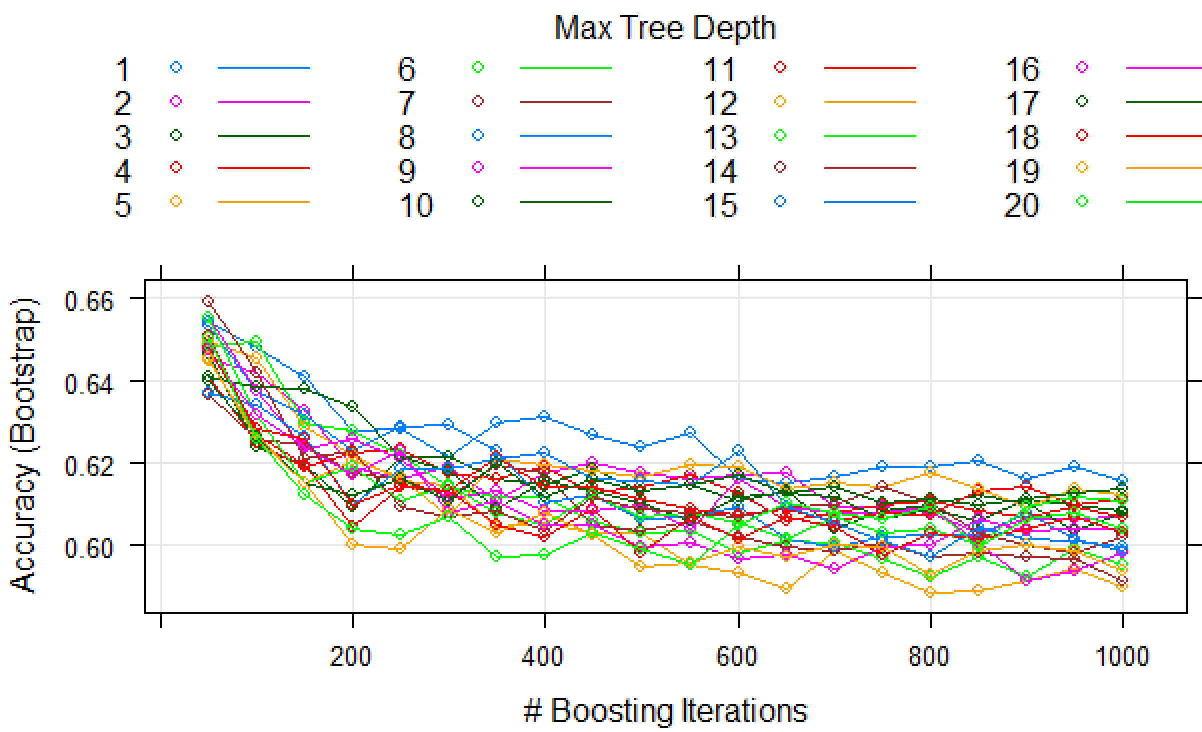
8	500	0.6066181	0.3640990
8	550	0.6067190	0.3613311
8	600	0.6092904	0.3659836
8	650	0.6013533	0.3524035
8	700	0.5996854	0.3524540
8	750	0.6022438	0.3560380
8	800	0.6027493	0.3563419
8	850	0.6025770	0.3557241
8	900	0.6081035	0.3633086
8	950	0.6013511	0.3541077
8	1000	0.5987864	0.3507780
9	50	0.6552520	0.4360529
9	100	0.6374962	0.4085252
9	150	0.6233265	0.3872269
9	200	0.6261910	0.3907434
9	250	0.6211230	0.3785583
9	300	0.6115006	0.3667249
9	350	0.6134798	0.3714560
9	400	0.6083053	0.3640224
9	450	0.6081677	0.3621760
9	500	0.6098811	0.3643488
9	550	0.6038152	0.3567461
9	600	0.6161863	0.3758666
9	650	0.6091472	0.3640170
9	700	0.6078592	0.3616650
9	750	0.6076819	0.3596654
9	800	0.6092335	0.3654923
9	850	0.6007227	0.3516878
9	900	0.5914021	0.3362290
9	950	0.5935638	0.3388249
9	1000	0.5982012	0.3463642
10	50	0.6412283	0.4125134
10	100	0.6384875	0.4080833
10	150	0.6383455	0.4120723
10	200	0.6338311	0.4028566
10	250	0.6213635	0.3823179
10	300	0.6182492	0.3755332
10	350	0.6090261	0.3606007
10	400	0.6155867	0.3730632
10	450	0.6132853	0.3683405
10	500	0.6103911	0.3656883
10	550	0.6060714	0.3576591
10	600	0.6119421	0.3681905
10	650	0.6128788	0.3718184
10	700	0.6118537	0.3691957
10	750	0.6065529	0.3605301
10	800	0.6086671	0.3624328
10	850	0.6059708	0.3570052
10	900	0.6112345	0.3669116
10	950	0.6097629	0.3647125
10	1000	0.6084296	0.3633550
11	50	0.6507124	0.4289611
11	100	0.6268978	0.3911950
11	150	0.6198657	0.3784753

```
  11                200     0.6045664  0.3546480
  11                250     0.6141726  0.3719702
  11                300     0.6133435  0.3696174
  11                350     0.6215217  0.3825874
  11                400     0.6141780  0.3722399
  11                450     0.6141715  0.3711845
  11                500     0.6114185  0.3681869
  11                550     0.6086629  0.3639070
  11                600     0.6076067  0.3613462
  11                650     0.6087158  0.3615347
  11                700     0.6060978  0.3593209
  11                750     0.6080837  0.3635802
  11                800     0.6075284  0.3599694
  11                850     0.6130287  0.3701176
  11                900     0.6141309  0.3716178
  11                950     0.6096757  0.3642440
  11               1000     0.6113484  0.3686408
  12                 50     0.6449416  0.4193296
  12                100     0.6270633  0.3920543
  12                150     0.6152220  0.3696926
  12                200     0.6203953  0.3782565
  12                250     0.6169003  0.3757165
  12                300     0.6068208  0.3588728
  12                350     0.6210473  0.3811554
  12                400     0.6196088  0.3781627
  12                450     0.6179647  0.3766453
  12                500     0.6167601  0.3753489
  12                550     0.6195095  0.3758202
  12                600     0.6192294  0.3785228
  12                650     0.6136202  0.3698144
  12                700     0.6152498  0.3723942
  12                750     0.6140733  0.3690369
  12                800     0.6176823  0.3754697
  12                850     0.6138734  0.3698250
  12                900     0.6095580  0.3635545
  12                950     0.6138946  0.3710268
  12               1000     0.6124258  0.3700798
  13                 50     0.6551513  0.4320009
  13                100     0.6316593  0.3964733
  13                150     0.6150700  0.3742939
  13                200     0.6193090  0.3813280
  13                250     0.6109922  0.3658247
  13                300     0.6149522  0.3734547
  13                350     0.6081313  0.3632731
  13                400     0.6050551  0.3578121
  13                450     0.6122801  0.3690805
  13                500     0.6071381  0.3616324
 [ 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
 = 14, shrinkage = 0.1 and n.minobsinnode = 10.
```

Hide

```
plot(model_gbm_tuned)
```



Hide

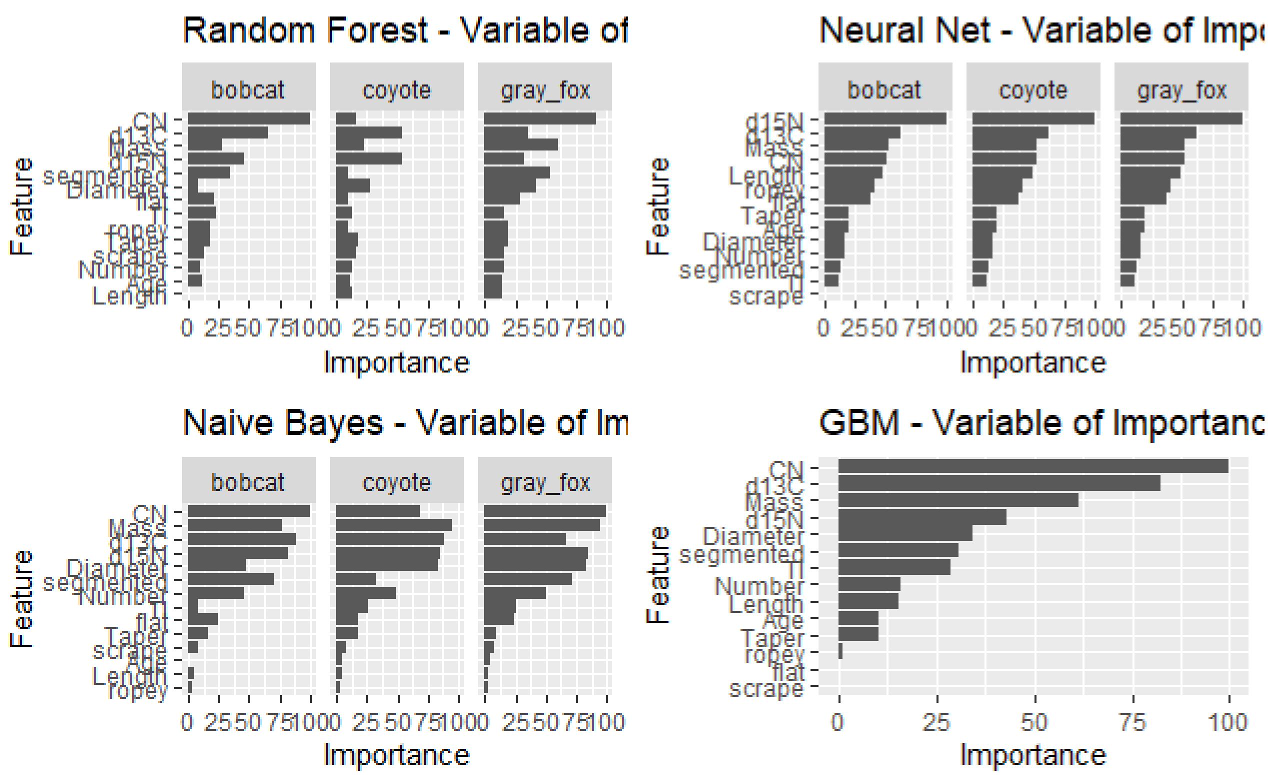
```
predict_gbm_tuned <- predict.train(model_gbm_tuned, testSet[,predictors])
results_gbm_tuned <- confusionMatrix(predict_gbm_tuned, testSet[,outcomeName])
```

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

Create ggplots for variable importance of each model, and combine them using grid.arrange(). This looks a bit messy in a notebook - should best be viewed as its own window.

Hide

```
rf_imp <- ggplot(data = varImp(object = model_rf))+ggtitle("Random Forest - Variable of Importance")

nn_imp <- ggplot(data = imp)+ggtitle("Neural Net - Variable of Importance")

nb_imp <- ggplot(data = varImp(object = model_nb))+ggtitle("Naive Bayes - Variable of Importance")

gbm_imp <- ggplot(data = varImp(object = model_gbm))+ggtitle("GBM - Variable of Importance")

grid.arrange(rf_imp, nn_imp, nb_imp, gbm_imp, ncol = 2)
```



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)

From comparing the accuracy of models earlier, we see that the neural network performs the best with an accuracy of about 85%. While we would like to have a higher accuracy, this is still significantly better than random guessing. This is decent considering our training set has only 83 observations. Also, because our test set has only 27 observations, the accuracy can only be measured in multiples of 1/27. Thus the other models all have an accuracy of .815.

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)

Hide

```
control <- rfeControl(functions = rfFuncs,
                      method = "repeatedcv",
                      repeats = 3,
                      verbose = F)
Scat_Pred_Profile <- rfe(trainSet[,predictors],
                         trainSet[,outcomeName],
                         rfeControl = control,
                         sizes = c(3, 6, 9, 12))
print(Scat_Pred_Profile)
```

```
Recursive feature selection

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

Resampling performance over subset size:
```

	Variables <S3: AsIs>	Accuracy <S3: AsIs>	Kappa <S3: AsIs>	AccuracySD <S3: AsIs>	KappaSD <S3: AsIs>	Selected <S3: AsIs>
1	3	0.7614	0.5883	0.1405	0.2391	*
2	6	0.7259	0.5339	0.1702	0.2772	
3	9	0.7310	0.5422	0.1547	0.2590	
4	12	0.7018	0.4838	0.1568	0.2634	
5	14	0.7017	0.4803	0.1427	0.2457	

5 rows

```
The top 3 variables (out of 3):
   CN, d13C, d15N
```

Our best variables are CN, d13C, and d15N. Now we restrict to only those features and train the appropriate models.

Hide

```
predictors <- c("CN", "d13C", "d15N")
model_rf_rfe <- train(x = trainSet[,predictors],
                      y = trainSet[,outcomeName],
                      method = 'rf',
                      importance = T,
                      verbose = F)
```

```
note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .
```

Hide

```
model_nn_rfe <- train(x = trainSet[,predictors],
                      y = trainSet[,outcomeName],
                      method = 'nnet',
                      importance = T,
                      trace = F)
model_nb_rfe <- train(x = trainSet[,predictors],
                      y = trainSet[,outcomeName],
                      method = 'naive_bayes')
model_gbm_rfe <- train(x = trainSet[,predictors],
                       y = trainSet[,outcomeName],
                       method = 'gbm',
                       verbose = F)
model_gbm_tuned_rfe <- train(x = trainSet[,predictors],
                             y = trainSet[,outcomeName],
                             method = 'gbm',
                             tuneLength = 20,
                             verbose = F)
```

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)

Hide

```
results_gbm_rfe <- confusionMatrix(predict.train(model_gbm_rfe,
                                             testSet[,predictors]),
                                   testSet[,outcomeName])
results_gbm_tuned_rfe <- confusionMatrix(predict.train(model_gbm_tuned_rfe,
                                             testSet[,predictors]),
                                         testSet[,outcomeName])
results_nn_rfe <- confusionMatrix(predict.train(model_nn_rfe,
                                             testSet[,predictors]),
                                  testSet[,outcomeName])
results_nb_rfe <- confusionMatrix(predict.train(model_nb_rfe,
                                             testSet[,predictors]),
                                  testSet[,outcomeName])
results_rf_rfe <- confusionMatrix(predict.train(model_rf_rfe,
                                             testSet[,predictors]),
                                  testSet[,outcomeName])

results_all_rfe <- as.data.frame(rbind(results_gbm_rfe$overall[c("Accuracy","Kappa")],
                     results_gbm_tuned_rfe$overall[c("Accuracy", "Kappa")],
                     results_nn_rfe$overall[c("Accuracy", "Kappa")],
                     results_rf_rfe$overall[c("Accuracy", "Kappa")],
                     results_nb_rfe$overall[c("Accuracy", "Kappa")]))
results_all_rfe <- cbind(c("GBM - RFE",
                       "GBM - RFE - Tuned",
                       "Neural Net - RFE",
                       "Random Forest - RFE",
                       "Naive Bayes - RFE"),
                     results_all_rfe)
colnames(results_all_rfe)[1] <- "ExperimentName"
results_all <- rbind(results_all, results_all_rfe)
results_all <- results_all[order(results_all$Accuracy, decreasing = TRUE),]
print(results_all)
```

	ExperimentName <fctr>	Accuracy <dbl>	Kappa <dbl>
1	Neural Net	0.8518519	0.7631579
2	Naive Bayes	0.8148148	0.6830986
3	Random Forest	0.8148148	0.6707317
4	GBM	0.8148148	0.6823529
9	Naive Bayes - RFE	0.8148148	0.6770335
7	Neural Net - RFE	0.7407407	0.5563380
8	Random Forest - RFE	0.7037037	0.5102041
5	GBM - RFE	0.6666667	0.4361949
6	GBM - RFE - Tuned	0.6296296	0.4013304

9 rows

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)

Once again, the neural network performs the best when comparing predictions to the test data. Using RFE actually reduced the performance of all of our models, which could be because of our low sample size. My answer holds from before - 85% accuracy is pretty good and would allow for fairly accurate prediction.