Question - 3 Extra

December 15, 2022

Arunaggiri Pandian Karunanidhi

0.0.1 Question - 3

H. EXTRA CREDIT (5 points): Using gradient boosting decision trees with R's xgboost library, generate a machine learning model for the wheat seed classification based on the features provided in the data. Generate a confusion matrix for the test data set to demonstrate the accuracy of the model. Based on your model, classify the beans provided in the unlabeled wheat-unknown.csv data set. Indicate which classification has been assigned to each of the unlabeled seeds. How do the results with xgboost compare to the support vector machine model?

```
In [1]: df = read.csv("/public/bmort/R/wheat.csv")
```

```
In [2]: # finding mode value

        find_mode <- function(x) {
          u <- unique(x)
          tab <- tabulate(match(x, u))
          u[tab == max(tab)]
        }

        mode_val = find_mode(df$width)
```

```
In [3]: # replacing null value with mode value

        # which(is.na(df$width))
        df$width[8] = mode_val
```

```
In [4]: sapply(df, function(x) sum(is.na(x)))
```

area 0 **perimeter** 0 **compactness** 0 **length** 0 **width** 0 **asymmetry** 0 **groove** 0 **type** 0

```
In [5]: # normalize data

        normalize <- function(x) {
        return ((x - min(x)) / (max(x) - min(x)))
        }

        df$area = normalize(df$area)
```

```
df$perimeter = normalize(df$perimeter)
df$length = normalize(df$length)
df$width = normalize(df$width)
df$asymmetry = normalize(df$asymmetry)
df$groove = normalize(df$groove)
df$compactness = normalize(df$compactness)
```

In [6]:
```
summary(df)
```

```
      area          perimeter       compactness         length      
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.1688   1st Qu.:0.2190   1st Qu.:0.4555   1st Qu.:0.2017  
 Median :0.3602   Median :0.4070   Median :0.6025   Median :0.3575  
 Mean   :0.4112   Mean   :0.4524   Mean   :0.5804   Mean   :0.4164  
 3rd Qu.:0.6438   3rd Qu.:0.6968   3rd Qu.:0.7244   3rd Qu.:0.6249  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
     width          asymmetry          groove       type  
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   A:68  
 1st Qu.:0.2324   1st Qu.:0.2247   1st Qu.:0.2578   B:69  
 Median :0.4324   Median :0.3675   Median :0.3481   C:63  
 Mean   :0.4528   Mean   :0.3768   Mean   :0.4408         
 3rd Qu.:0.6627   3rd Qu.:0.5122   3rd Qu.:0.6696         
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000         
```

In [7]:
```
# Replacing outlier values with NaN values

for (x in c('compactness','asymmetry'))
{
  value = df[,x][df[,x] %in% boxplot.stats(df[,x])$out]
  df[,x][df[,x] %in% value] = NA
}
```

In [8]:
```
# now let's remove the rows that has null values

#Removing the null values

library(tidyr)
df = drop_na(df)
as.data.frame(colSums(is.na(df)))
# sapply(df, function(x) sum(is.na(x)))
```

A data.frame: 8 Œ 1

	colSums(is.na(df)) <dbl>
area	0
perimeter	0
compactness	0
length	0
width	0
asymmetry	0
groove	0
type	0

```
In [9]: library(caret)
        library(xgboost)
```

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

```
In [10]: intrain = createDataPartition(y = df$type, p= 0.8, list = FALSE)
         train = df[intrain,]
         test = df[-intrain,]
```

```
In [11]: # Fit the model on the training set

         set.seed(123)
         model = train(
           type ~., data = train, method = "xgbTree",
           trControl = trainControl("cv", number = 5)
           )
         # Best tuning parameter
         model$bestTune
```

A data.frame: 1 Œ 7

	nrounds <dbl>	max_depth <int>	eta <dbl>	gamma <dbl>	colsample_bytree <dbl>	min_child_weight <dbl>
65	100	1	0.4	0	0.8	1

```
In [15]: model
```

```
eXtreme Gradient Boosting

158 samples
  7 predictor
  3 classes: 'A', 'B', 'C'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 127, 126, 127, 126, 126
Resampling results across tuning parameters:
```

eta	max_depth	colsample_bytree	subsample	nrounds	Accuracy	Kappa
0.3	1	0.6	0.50	50	0.8983871	0.8470061
0.3	1	0.6	0.50	100	0.8921371	0.8375391
0.3	1	0.6	0.50	150	0.8921371	0.8375391
0.3	1	0.6	0.75	50	0.8983871	0.8470061
0.3	1	0.6	0.75	100	0.9048387	0.8567546
0.3	1	0.6	0.75	150	0.9048387	0.8567546
0.3	1	0.6	1.00	50	0.8983871	0.8470061
0.3	1	0.6	1.00	100	0.8985887	0.8473843
0.3	1	0.6	1.00	150	0.8985887	0.8473843
0.3	1	0.8	0.50	50	0.8983871	0.8470061
0.3	1	0.8	0.50	100	0.9048387	0.8567546
0.3	1	0.8	0.50	150	0.9112903	0.8665030
0.3	1	0.8	0.75	50	0.9048387	0.8567546
0.3	1	0.8	0.75	100	0.8985887	0.8472875
0.3	1	0.8	0.75	150	0.8985887	0.8472875
0.3	1	0.8	1.00	50	0.8985887	0.8473843
0.3	1	0.8	1.00	100	0.8985887	0.8473843
0.3	1	0.8	1.00	150	0.8985887	0.8473843
0.3	2	0.6	0.50	50	0.9046371	0.8563491
0.3	2	0.6	0.50	100	0.9046371	0.8563491
0.3	2	0.6	0.50	150	0.9110887	0.8660975
0.3	2	0.6	0.75	50	0.9048387	0.8567546
0.3	2	0.6	0.75	100	0.9112903	0.8665030
0.3	2	0.6	0.75	150	0.9112903	0.8665030
0.3	2	0.6	1.00	50	0.9048387	0.8567546
0.3	2	0.6	1.00	100	0.9048387	0.8567546
0.3	2	0.6	1.00	150	0.9048387	0.8567546
0.3	2	0.8	0.50	50	0.9048387	0.8567546
0.3	2	0.8	0.50	100	0.9108871	0.8657884
0.3	2	0.8	0.50	150	0.9108871	0.8657884
0.3	2	0.8	0.75	50	0.8983871	0.8470061
0.3	2	0.8	0.75	100	0.8983871	0.8470061
0.3	2	0.8	0.75	150	0.9048387	0.8567546
0.3	2	0.8	1.00	50	0.9048387	0.8567546
0.3	2	0.8	1.00	100	0.9048387	0.8567546
0.3	2	0.8	1.00	150	0.9048387	0.8567546
0.3	3	0.6	0.50	50	0.9112903	0.8665030
0.3	3	0.6	0.50	100	0.9175403	0.8759423
0.3	3	0.6	0.50	150	0.9175403	0.8759423
0.3	3	0.6	0.75	50	0.9048387	0.8567546
0.3	3	0.6	0.75	100	0.8983871	0.8470061
0.3	3	0.6	0.75	150	0.8983871	0.8470061
0.3	3	0.6	1.00	50	0.9048387	0.8567546
0.3	3	0.6	1.00	100	0.9048387	0.8567546
0.3	3	0.6	1.00	150	0.9048387	0.8567546
0.3	3	0.8	0.50	50	0.9048387	0.8567546
0.3	3	0.8	0.50	100	0.9110887	0.8660975

0.3	3	0.8	0.50	150	0.9110887	0.8661939
0.3	3	0.8	0.75	50	0.8983871	0.8470061
0.3	3	0.8	0.75	100	0.8983871	0.8470061
0.3	3	0.8	0.75	150	0.8983871	0.8470061
0.3	3	0.8	1.00	50	0.9048387	0.8567546
0.3	3	0.8	1.00	100	0.9048387	0.8567546
0.3	3	0.8	1.00	150	0.9048387	0.8567546
0.4	1	0.6	0.50	50	0.9048387	0.8567546
0.4	1	0.6	0.50	100	0.9046371	0.8564454
0.4	1	0.6	0.50	150	0.9110887	0.8661939
0.4	1	0.6	0.75	50	0.8983871	0.8470061
0.4	1	0.6	0.75	100	0.9048387	0.8567546
0.4	1	0.6	0.75	150	0.8985887	0.8472875
0.4	1	0.6	1.00	50	0.8985887	0.8473843
0.4	1	0.6	1.00	100	0.8923387	0.8379173
0.4	1	0.6	1.00	150	0.8987903	0.8477262
0.4	1	0.8	0.50	50	0.9046371	0.8564454
0.4	1	0.8	0.50	100	0.9237903	0.8852852
0.4	1	0.8	0.50	150	0.9175403	0.8758459
0.4	1	0.8	0.75	50	0.8921371	0.8375391
0.4	1	0.8	0.75	100	0.9048387	0.8567546
0.4	1	0.8	0.75	150	0.9048387	0.8567546
0.4	1	0.8	1.00	50	0.9048387	0.8567546
0.4	1	0.8	1.00	100	0.9048387	0.8567546
0.4	1	0.8	1.00	150	0.8923387	0.8379589
0.4	2	0.6	0.50	50	0.9175403	0.8759423
0.4	2	0.6	0.50	100	0.9175403	0.8759423
0.4	2	0.6	0.50	150	0.9175403	0.8759423
0.4	2	0.6	0.75	50	0.8983871	0.8470061
0.4	2	0.6	0.75	100	0.9112903	0.8665635
0.4	2	0.6	0.75	150	0.9048387	0.8568151
0.4	2	0.6	1.00	50	0.9048387	0.8568151
0.4	2	0.6	1.00	100	0.9048387	0.8568151
0.4	2	0.6	1.00	150	0.9048387	0.8568151
0.4	2	0.8	0.50	50	0.9110887	0.8660975
0.4	2	0.8	0.50	100	0.9110887	0.8660975
0.4	2	0.8	0.50	150	0.9175403	0.8759065
0.4	2	0.8	0.75	50	0.9112903	0.8665030
0.4	2	0.8	0.75	100	0.9112903	0.8665030
0.4	2	0.8	0.75	150	0.9050403	0.8571327
0.4	2	0.8	1.00	50	0.9048387	0.8567546
0.4	2	0.8	1.00	100	0.9048387	0.8567546
0.4	2	0.8	1.00	150	0.8985887	0.8473843
0.4	3	0.6	0.50	50	0.9048387	0.8567546
0.4	3	0.6	0.50	100	0.9048387	0.8567546
0.4	3	0.6	0.50	150	0.9048387	0.8567546
0.4	3	0.6	0.75	50	0.9048387	0.8567546
0.4	3	0.6	0.75	100	0.9048387	0.8567546

```
  0.4  3          0.6               0.75      150       0.9048387  0.8567546
  0.4  3          0.6               1.00       50       0.8983871  0.8470061
  0.4  3          0.6               1.00      100       0.8983871  0.8470061
  0.4  3          0.6               1.00      150       0.8983871  0.8470061
  0.4  3          0.8               0.50       50       0.9110887  0.8660975
  0.4  3          0.8               0.50      100       0.9110887  0.8660975
  0.4  3          0.8               0.50      150       0.9110887  0.8660975
  0.4  3          0.8               0.75       50       0.9048387  0.8567546
  0.4  3          0.8               0.75      100       0.8983871  0.8470061
  0.4  3          0.8               0.75      150       0.8983871  0.8470061
  0.4  3          0.8               1.00       50       0.8983871  0.8470061
  0.4  3          0.8               1.00      100       0.8983871  0.8470061
  0.4  3          0.8               1.00      150       0.8983871  0.8470061

Tuning parameter 'gamma' was held constant at a value of 0
Tuning
 parameter 'min_child_weight' 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 nrounds = 100, max_depth = 1, eta
 = 0.4, gamma = 0, colsample_bytree = 0.8, min_child_weight = 1 and subsample
 = 0.5.
```

In [16]:
```
# let's apply our model to the test set

test_pred <- predict(model, newdata = test)
test_pred
```

1. A 2. A 3. A 4. A 5. A 6. A 7. C 8. A 9. A 10. A 11. A 12. A 13. A 14. B 15. B 16. B 17. B 18. B 19. B 20. B 21. B 22. B 23. B 24. B 25. B 26. B 27. A 28. C 29. C 30. C 31. C 32. C 33. C 34. C 35. C 36. C 37. C

Levels: 1. 'A' 2. 'B' 3. 'C'

In [17]:
```
# Compute model prediction accuracy rate

mean(test_pred == test$type)
```

0.945945945945946

In [18]:
```
confusionMatrix(table(test_pred, test$type))
```

```
Confusion Matrix and Statistics


test_pred  A  B  C
        A 12  0  1
        B  0 13  0
        C  1  0 10
```

```
Overall Statistics

               Accuracy : 0.9459
                 95% CI : (0.8181, 0.9934)
    No Information Rate : 0.3514
    P-Value [Acc > NIR] : 3.643e-14

                  Kappa : 0.9187

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C
Sensitivity            0.9231   1.0000   0.9091
Specificity            0.9583   1.0000   0.9615
Pos Pred Value         0.9231   1.0000   0.9091
Neg Pred Value         0.9583   1.0000   0.9615
Prevalence             0.3514   0.3514   0.2973
Detection Rate         0.3243   0.3514   0.2703
Detection Prevalence   0.3514   0.3514   0.2973
Balanced Accuracy      0.9407   1.0000   0.9353
```

```
In [13]: dim(train)
```

1. 158 2. 8

```
In [14]: dim(test)
```

1. 37 2. 8

```
In [12]: # Make predictions on the test data

         predicted.classes = model %>% predict(test)
         head(predicted.classes)
```

1. A 2. A 3. A 4. A 5. A 6. A
Levels: 1. 'A' 2. 'B' 3. 'C'

```
In [23]: # Compute model prediction accuracy rate
         mean(predicted.classes == test$type)
```

0.972972972972973

```
In [19]: # let's apply our model to the test set

         test_pred <- predict(model, newdata = test)
         test_pred
```

1. A 2. A 3. A 4. A 5. A 6. A 7. C 8. A 9. A 10. A 11. A 12. A 13. A 14. B 15. B 16. B 17. B 18. B 19. B 20. B 21. B 22. B 23. B 24. B 25. B 26. B 27. A 28. C 29. C 30. C 31. C 32. C 33. C 34. C 35. C 36. C 37. C

Levels: 1. 'A' 2. 'B' 3. 'C'

In [20]:
```
# Compute model prediction accuracy rate

mean(test_pred == test$type)
```

0.945945945945946

In [26]:
```
# variable importance

varImp(model)
```

```
xgbTree variable importance

             Overall
groove      100.00000
area         73.20171
perimeter    28.03824
asymmetry    15.26741
width         7.04142
length        0.01702
compactness   0.00000
```

In [27]:
```
confusionMatrix(table(test_pred, test$type))
```

```
Confusion Matrix and Statistics


test_pred  A  B  C
        A 13  0  1
        B  0 13  0
        C  0  0 10

Overall Statistics

               Accuracy : 0.973
                 95% CI : (0.8584, 0.9993)
    No Information Rate : 0.3514
    P-Value [Acc > NIR] : 1.079e-15

                  Kappa : 0.9593

 Mcnemar's Test P-Value : NA

Statistics by Class:
```

```
                      Class: A Class: B Class: C
Sensitivity             1.0000   1.0000   0.9091
Specificity             0.9583   1.0000   1.0000
Pos Pred Value          0.9286   1.0000   1.0000
Neg Pred Value          1.0000   1.0000   0.9630
Prevalence              0.3514   0.3514   0.2973
Detection Rate          0.3514   0.3514   0.2703
Detection Prevalence    0.3784   0.3514   0.2703
Balanced Accuracy       0.9792   1.0000   0.9545
```

In [22]:
```
unknown = read.csv('/public/bmort/R/wheat-unknown.csv')
dim(unknown)
```

1. 10 2. 7

In [23]:
```
test_pred <- predict(model, newdata = unknown)
test_pred
```

1. B 2. B 3. B 4. B 5. B 6. B 7. B 8. B 9. B 10. B
Levels: 1. 'A' 2. 'B' 3. 'C'

In [24]:
```
unknown$type = test_pred
unknown
```

A data.frame: 10 Œ 8

area <dbl>	perimeter <dbl>	compactness <dbl>	length <dbl>	width <dbl>	asymmetry <dbl>	groove <dbl>	type <fct>
11.56	13.31	0.8198	5.363	2.683	4.062	5.182	B
14.79	14.52	0.8819	5.545	3.291	2.704	5.111	B
10.82	12.83	0.8256	5.180	2.630	4.853	5.089	B
13.32	13.94	0.8613	5.541	3.073	7.035	5.440	B
11.49	13.22	0.8263	5.304	2.695	5.388	5.310	B
10.83	12.96	0.8099	5.278	2.641	5.182	5.185	B
15.11	14.54	0.8986	5.579	3.462	3.128	5.180	B
11.19	13.05	0.8253	5.250	2.675	5.813	5.219	B
12.02	13.33	0.8503	5.350	2.810	4.271	5.308	B
17.99	15.86	0.8992	5.890	3.694	2.068	5.837	B