QUESTION 3H

H. 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 [8]:

```
install.packages('xgboost')
library(xgboost)
install.packages('caret')
library(caret)
install.packages('ggplot2')
install.packages('lattice')
library(ggplot2)
library(lattice)
```

In [2]:

```
wheat_x <- read.csv('/public/bmort/R/wheat.csv')
head(wheat_x,5)
```

A data.frame: 5 × 8

area	perimeter	compactness	length	width	asymmetry	groove	type
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>
15.26	14.84	0.8710	5.763	3.312	2.221	5.220	A
14.88	14.57	0.8811	5.554	3.333	1.018	4.956	A
14.29	14.09	0.9050	5.291	3.337	2.699	4.825	A
13.84	13.94	0.8955	5.324	3.379	2.259	4.805	A
16.14	14.99	0.9034	5.658	3.562	1.355	5.175	A

In [3]:

```
## Splitting the data into training and testing data
splitt <- createDataPartition(y= wheat_x$type, p= 0.80, list = FALSE)
```

In [38]:

```
## Training and testing data
trainn <- wheat_x[splitt, ]
testt <- wheat_x[-splitt,]
```

In [39]:

```
#determine predictor and target variables in training set
train_X <-data.matrix(trainn[,-8])
train_y <- trainn[,8]
```

In [40]:

```
#determine predictor and response variables in testing set
test_X <- data.matrix(testt[,-8])
test_y <- testt[,8]
```

In [41]:

```
## Final training and testing data
# converting the train and test data into xgboost matrix type.
Xgb_train <- xgb.DMatrix(data = train_X, label = train_y)
Xgb_test <- xgb.DMatrix(data = test_X, label = test_y)
```

In [46]:

```
## defining watchlist
watch_list <- list(train= Xgb_train,test= Xgb_test)

## Creating the model
xg_model <- xgb.train(data = Xgb_train, max.depth = 3, watchlist = watch_list, nrounds
= 70)
# summary(xg_model)
```

```
[1]	train-rmse:1.209089	test-rmse:1.208659
[2]	train-rmse:0.873718	test-rmse:0.871835
[3]	train-rmse:0.640466	test-rmse:0.648591
[4]	train-rmse:0.479072	test-rmse:0.505488
[5]	train-rmse:0.366879	test-rmse:0.407975
[6]	train-rmse:0.286911	test-rmse:0.356380
[7]	train-rmse:0.231867	test-rmse:0.320092
[8]	train-rmse:0.194603	test-rmse:0.308039
[9]	train-rmse:0.168215	test-rmse:0.304311
[10]	train-rmse:0.150808	test-rmse:0.299737
[11]	train-rmse:0.140158	test-rmse:0.298941
[12]	train-rmse:0.129001	test-rmse:0.303604
[13]	train-rmse:0.122792	test-rmse:0.301365
[14]	train-rmse:0.115298	test-rmse:0.298990
[15]	train-rmse:0.113144	test-rmse:0.300021
[16]	train-rmse:0.111610	test-rmse:0.300074
[17]	train-rmse:0.108576	test-rmse:0.302950
[18]	train-rmse:0.102985	test-rmse:0.303109
[19]	train-rmse:0.099516	test-rmse:0.302980
[20]	train-rmse:0.095539	test-rmse:0.302861
[21]	train-rmse:0.093030	test-rmse:0.302945
[22]	train-rmse:0.089520	test-rmse:0.302915
[23]	train-rmse:0.085736	test-rmse:0.303466
[24]	train-rmse:0.081723	test-rmse:0.307019
[25]	train-rmse:0.079704	test-rmse:0.306318
[26]	train-rmse:0.075999	test-rmse:0.304814
[27]	train-rmse:0.074217	test-rmse:0.304922
[28]	train-rmse:0.070865	test-rmse:0.304198
[29]	train-rmse:0.068277	test-rmse:0.304071
[30]	train-rmse:0.065921	test-rmse:0.303537
[31]	train-rmse:0.063231	test-rmse:0.305327
[32]	train-rmse:0.061330	test-rmse:0.305064
[33]	train-rmse:0.060070	test-rmse:0.304929
[34]	train-rmse:0.056920	test-rmse:0.305270
[35]	train-rmse:0.054811	test-rmse:0.304541
[36]	train-rmse:0.052888	test-rmse:0.304622
[37]	train-rmse:0.049996	test-rmse:0.300759
[38]	train-rmse:0.047264	test-rmse:0.300882
[39]	train-rmse:0.045981	test-rmse:0.300983
[40]	train-rmse:0.044827	test-rmse:0.300787
[41]	train-rmse:0.043348	test-rmse:0.301096
[42]	train-rmse:0.041247	test-rmse:0.301317
[43]	train-rmse:0.039651	test-rmse:0.300635
[44]	train-rmse:0.038232	test-rmse:0.300291
[45]	train-rmse:0.036961	test-rmse:0.300564
[46]	train-rmse:0.035821	test-rmse:0.300235
[47]	train-rmse:0.035473	test-rmse:0.300143
[48]	train-rmse:0.034385	test-rmse:0.300060
[49]	train-rmse:0.032380	test-rmse:0.297359
[50]	train-rmse:0.031419	test-rmse:0.296653
[51]	train-rmse:0.030496	test-rmse:0.297024
[52]	train-rmse:0.029773	test-rmse:0.296946
[53]	train-rmse:0.029425	test-rmse:0.296782
[54]	train-rmse:0.028883	test-rmse:0.296843
[55]	train-rmse:0.028564	test-rmse:0.296923
[56]	train-rmse:0.028234	test-rmse:0.296804
[57]	train-rmse:0.027929	test-rmse:0.296774
[58]	train-rmse:0.027339	test-rmse:0.296692
[59]	train-rmse:0.026817	test-rmse:0.296387
[60]	train-rmse:0.026610	test-rmse:0.296445
[61]	train-rmse:0.026143	test-rmse:0.296381
```

```
[62]	train-rmse:0.025321	test-rmse:0.296064
[63]	train-rmse:0.024299	test-rmse:0.296329
[64]	train-rmse:0.023492	test-rmse:0.296354
[65]	train-rmse:0.022822	test-rmse:0.296251
[66]	train-rmse:0.022185	test-rmse:0.296303
[67]	train-rmse:0.021488	test-rmse:0.296845
[68]	train-rmse:0.020897	test-rmse:0.296682
[69]	train-rmse:0.020374	test-rmse:0.296637
[70]	train-rmse:0.019961	test-rmse:0.296785
```

In [47]:

```
## Defining a the final model
f_model <- xgboost(data = Xgb_train, max.depth =3, nrounds = 50)
```

```
[1]	train-rmse:1.209089
[2]	train-rmse:0.873718
[3]	train-rmse:0.640466
[4]	train-rmse:0.479072
[5]	train-rmse:0.366879
[6]	train-rmse:0.286911
[7]	train-rmse:0.231867
[8]	train-rmse:0.194603
[9]	train-rmse:0.168215
[10]	train-rmse:0.150808
[11]	train-rmse:0.140158
[12]	train-rmse:0.129001
[13]	train-rmse:0.122792
[14]	train-rmse:0.115298
[15]	train-rmse:0.113144
[16]	train-rmse:0.111610
[17]	train-rmse:0.108576
[18]	train-rmse:0.102985
[19]	train-rmse:0.099516
[20]	train-rmse:0.095539
[21]	train-rmse:0.093030
[22]	train-rmse:0.089520
[23]	train-rmse:0.085736
[24]	train-rmse:0.081723
[25]	train-rmse:0.079704
[26]	train-rmse:0.075999
[27]	train-rmse:0.074217
[28]	train-rmse:0.070865
[29]	train-rmse:0.068277
[30]	train-rmse:0.065921
[31]	train-rmse:0.063231
[32]	train-rmse:0.061330
[33]	train-rmse:0.060070
[34]	train-rmse:0.056920
[35]	train-rmse:0.054811
[36]	train-rmse:0.052888
[37]	train-rmse:0.049996
[38]	train-rmse:0.047264
[39]	train-rmse:0.045981
[40]	train-rmse:0.044827
[41]	train-rmse:0.043348
[42]	train-rmse:0.041247
[43]	train-rmse:0.039651
[44]	train-rmse:0.038232
[45]	train-rmse:0.036961
[46]	train-rmse:0.035821
[47]	train-rmse:0.035473
[48]	train-rmse:0.034385
[49]	train-rmse:0.032380
[50]	train-rmse:0.031419
```

In [62]:

```
## Predicting the test data
pred_model <- predict(f_model, Xgb_test, reshape = TRUE)
pred_model
```

1.07092070579529 1.08245182037354 1.34180295467377
0.87090277671814 2.41850447654724 1.34151339530945
1.54925894737244 1.26427865028381 1.14174437522888
0.993972659111023 1.14301800727844 1.26121783256531
1.09366059303284 1.96713089942932 2.0047287940979
2.04513692855835 2.00014972686768 2.00773334503174
1.94364893436432 2.02451372146606 1.96146869659424
2.01924133300781 1.97906804084778 1.82517683506012
1.88484585285187 1.87115859985352 2.57206106185913
2.90848565101624 2.99162435531616 2.98265147209167
3.00836515426636 3.00075268745422 3.01530051231384
2.57041525840759 2.97980642318726 2.97694706916809
2.75333571434021 3.26959133148193

In [60]:

```
## Converting the predictions to factors
pred_model[(pred_model>3)] = 3
pred_y = as.factor((levels(test_y))[round(pred_model)])
print(pred_y)
```

```
 [1] A A A A B A B A A A A A A B B B B B B B B B B B B B C C C C C C C C C
C C C
Levels: A B C
```

In [65]:

```
## Creating the confusion matrix
conf_matrix <- confusionMatrix(test_y, pred_y )
conf_matrix
```

```
Confusion Matrix and Statistics

          Reference
Prediction  A  B  C
         A 11  2  0
         B  0 13  0
         C  0  0 12

Overall Statistics

               Accuracy : 0.9474
                 95% CI : (0.8225, 0.9936)
    No Information Rate : 0.3947
    P-Value [Acc > NIR] : 7.82e-13

                  Kappa : 0.921

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C
Sensitivity            1.0000   0.8667   1.0000
Specificity            0.9259   1.0000   1.0000
Pos Pred Value         0.8462   1.0000   1.0000
Neg Pred Value         1.0000   0.9200   1.0000
Prevalence             0.2895   0.3947   0.3158
Detection Rate         0.2895   0.3421   0.3158
Detection Prevalence   0.3421   0.3421   0.3158
Balanced Accuracy      0.9630   0.9333   1.0000
```

This xgboost model has an accuracy of 94.74% classification rate. From the confusion matrix, we can see that model correctly classifies 11 type A wheat seeds as type A and falsely classified 2 type A wheat seeds as type B. It also, correctly classified 13 type B wheat seeds as type B. Finally it correctly classified 12 type C as type C.

In [68]:

```
## Predicting the new dataset
wheatx_new <- read.csv('/public/bmort/R/wheat-unknown.csv')
wheatx_new
```

A data.frame: 10 × 7

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

In [76]:

```
## Prediction
pred_new <- predict(f_model, as.matrix(wheatx_new))
pred_new[(pred_new > 3)] =3
predy_new <- as.factor((levels(test_y))[round(pred_new)])
predy_new
```

C A C C C C A C C B

▶ **Levels**:

Comparing the SVM model and the XGBoost model

- Both models have a 90% + accuracy rate.
- Also, both tend to give the same predicitions for our new unlabeled dataset [C,A,C, C, C, C, A, C, C, B]
- Therefore any of these two models can be used to make predictions.
- But looking at the complex nature of the XGBoost model, the SVM should be used because it is easy to implement and takes a short time to run.