Classifier Performance and Cross Validation

Objective: to predict the age in years of abalone shells based on physical measurements

Data set: Abalone dataset https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/

First, I prepared the environment, set the working directory and loaded the libraries using the following:

```
> ###   Classifier Performance and Cross Validation
>
> rm(list=ls()) #Clear the environment
> setwd("YOUR_PATH") #Set working directory for the assignment
> getwd() #Check working directory
[1] "YOUR_PATH"
>
> #Age prediction for abalone shells based on physical measurements
> #Abalone dataset https://archive.ics.uci.edu/ml/datasets/Abalone
>
> #Load packages
> library(caret)
> library(e1071)
> library(rpart)
```

Next, I loaded the abalone dataset from the previously downloaded data file (https://archive.ics.uci.edu/ml/datasets/Abalone) using the following code:

```
> #Load data
> abalone <- read.table("abalone.data", header = FALSE, sep = ",")
```

To make sure that the data was loaded correctly, I looked at the internal structure of the data frame:

```
> str(abalone) #Check internal structure of the data frame
'data.frame':      4177 obs. of  9 variables:
 $ V1: Factor w/ 3 levels "F","I","M": 3 3 1 3 2 2 1 1 3 1 ...
 $ V2: num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 ...
 $ V3: num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44 ...
 $ V4: num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15 ...
 $ V5: num  0.514 0.226 0.677 0.516 0.205 ...
 $ V6: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
 $ V7: num  0.101 0.0485 0.1415 0.114 0.0395 ...
 $ V8: num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ...
 $ V9: int  15 7 9 10 7 8 20 16 9 19 ...
```

The data was loaded correctly.

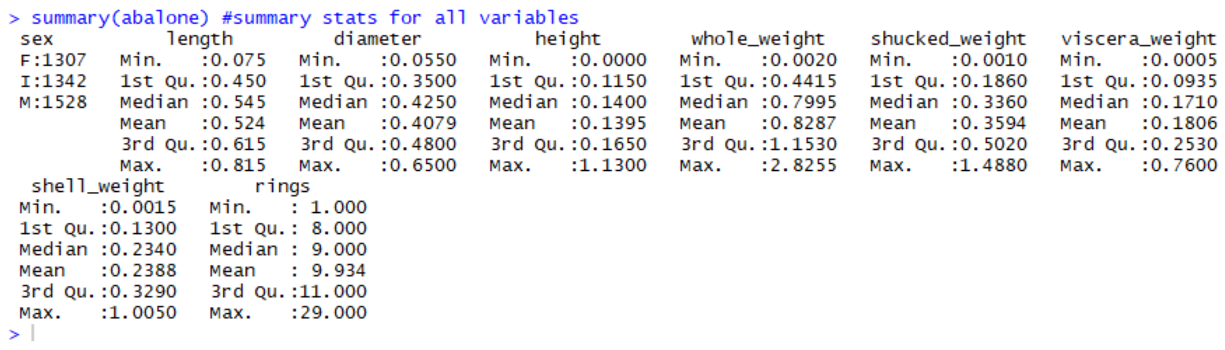
The dataset contains 4177 observations of 9 variables. The variable for sex was loaded as a factor with three levels (“male”, “female”, “infant”) and the rest of the variables were represented by numerical values and integers (for the number of rings).

First, for convenience purposes I added the appropriate column names:

```
> #Add column names
> names(abalone) <- c("sex", "length", "diameter", "height", "whole_weight", "shucked_weight", "v
iscera_weight", "shell_weight", "rings")
> str(abalone) #Check results
'data.frame':	4177 obs. of  9 variables:
 $ sex           : Factor w/ 3 levels "F","I","M": 3 3 1 3 2 2 1 1 3 1 ...
 $ length        : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 ...
 $ diameter      : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44 ...
 $ height        : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15 ...
 $ whole_weight  : num  0.514 0.226 0.677 0.516 0.205 ...
 $ shucked_weight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
 $ viscera_weight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
 $ shell_weight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ...
 $ rings         : int  15 7 9 10 7 8 20 16 9 19 ...
```

Next, to familiarize myself with the data, I looked at the first few records in the data frame and at the summary statistics for all variables in the dataset:

```
> ###EDA
>
> head(abalone) #First few rows
  sex length diameter height whole_weight shucked_weight viscera_weight shell_weight rings
1   M  0.455    0.365  0.095       0.5140         0.2245         0.1010        0.150    15
2   M  0.350    0.265  0.090       0.2255         0.0995         0.0485        0.070     7
3   F  0.530    0.420  0.135       0.6770         0.2565         0.1415        0.210     9
4   M  0.440    0.365  0.125       0.5160         0.2155         0.1140        0.155    10
5   I  0.330    0.255  0.080       0.2050         0.0895         0.0395        0.055     7
6   I  0.425    0.300  0.095       0.3515         0.1410         0.0775        0.120     8
```



```
> summary(abalone) #summary stats for all variables
 sex          length         diameter          height         whole_weight     shucked_weight    viscera_weight  
 F:1307   Min.   :0.075   Min.   :0.0550   Min.   :0.0000   Min.   :0.0020   Min.   :0.0010   Min.   :0.0005  
 I:1342   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150   1st Qu.:0.4415   1st Qu.:0.1860   1st Qu.:0.0935  
 M:1528   Median :0.545   Median :0.4250   Median :0.1400   Median :0.7995   Median :0.3360   Median :0.1710  
          Mean   :0.524   Mean   :0.4079   Mean   :0.1395   Mean   :0.8287   Mean   :0.3594   Mean   :0.1806  
          3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650   3rd Qu.:1.1530   3rd Qu.:0.5020   3rd Qu.:0.2530  
          Max.   :0.815   Max.   :0.6500   Max.   :1.1300   Max.   :2.8255   Max.   :1.4880   Max.   :0.7600  
  shell_weight         rings       
 Min.   :0.0015   Min.   : 1.000  
 1st Qu.:0.1300   1st Qu.: 8.000  
 Median :0.2340   Median : 9.000  
 Mean   :0.2388   Mean   : 9.934  
 3rd Qu.:0.3290   3rd Qu.:11.000  
 Max.   :1.0050   Max.   :29.000  
> 
```

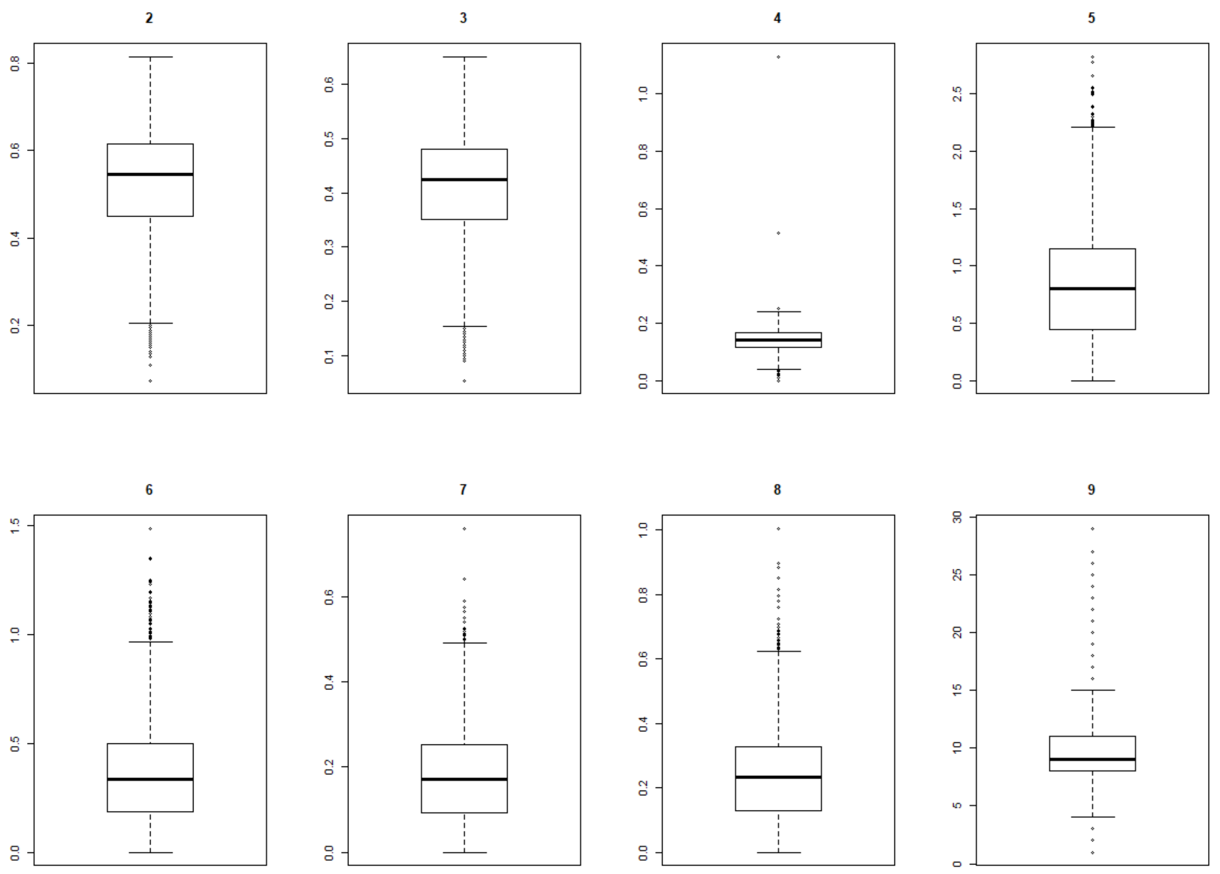
I have not noticed any missing data, but to doublecheck I used the following code, which returned zero missing values:

```
> sum(is.na(abalone)) #Missing values?
[1] 0
```

Next, I looked at boxplots for all numerical variables in the dataset:

```
> #Boxplots for all numerical variables in the dataset
> par(mfrow=c(2,4))
> for (i in 2:9){
+   boxplot(abalone[i], main=i)
+ }
```

The output is below:



The box plots show that none of the variables is normally distributed. They all represent skewed distributions and contain outliers. These plots suggest that in order to use any type of classification algorithms based on distance measurements, the observations will need to be scaled.

Next step – data preprocessing. Since I was planning on using classification algorithms based on distance measurements, I needed to encode the sex variable (currently a factor with three levels. So, I introduced new dummy variables female and infant. The data set includes 1307 female samples, 1528 male samples and 1342 samples marked "infant".

```
> #Data Pre-Processing
> #Encoding sex variable
> table(abalone$sex)

   F    I    M 
1307 1342 1528 
```

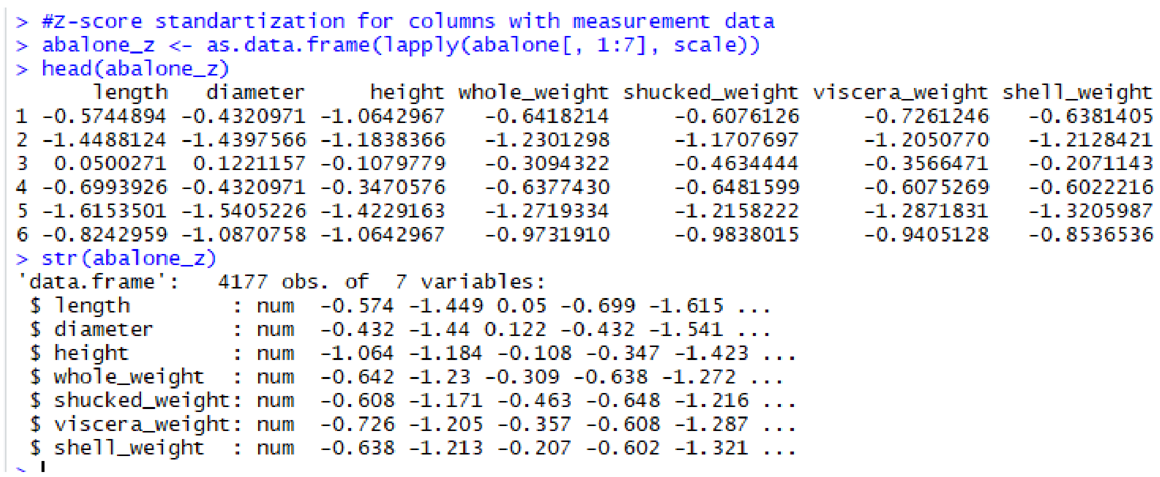
So, I added a new variable female equal to 1 for all samples where sex is “F”, and a new variable infant for all samples where sex = “I”:

```
> abalone$female <- ifelse(abalone$sex == "F", 1, 0)
> abalone$infant <- ifelse(abalone$sex == "I", 1, 0)
>
> #Check results
> table(abalone$female)

   0    1
2870 1307
> table(abalone$infant)

   0    1
2835 1342
> str(abalone)
'data.frame':      4177 obs. of  11 variables:
 $ sex           : Factor w/ 3 levels "F","I","M": 3 3 1 3 2 2 1 1 3 1 ...
 $ length        : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 ...
 $ diameter      : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44 ...
 $ height        : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15 ...
 $ whole_weight  : num  0.514 0.226 0.677 0.516 0.205 ...
 $ shucked_weight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
 $ viscera_weight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
 $ shell_weight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ...
 $ rings         : int  15 7 9 10 7 8 20 16 9 19 ...
 $ female        : num  0 0 1 0 0 0 1 1 0 1 ...
 $ infant        : num  0 0 0 0 1 1 0 0 0 0 ...
```

After verifying results, I deleted the initial sex variable:

```
> #delete sex column
> abalone <- abalone[,-1]
```

The next step – is data standardization, since the data frame contains highly variable measurements in different units (distance and weight). I applied scale() function to all numerical fields, which yielded the following results:



```
> #Z-score standartization for columns with measurement data
> abalone_z <- as.data.frame(lapply(abalone[, 1:7], scale))
> head(abalone_z)
      length   diameter     height whole_weight shucked_weight viscera_weight shell_weight
1 -0.5744894 -0.4320971 -1.0642967   -0.6418214     -0.6076126     -0.7261246   -0.6381405
2 -1.4488124 -1.4397566 -1.1838366   -1.2301298     -1.1707697     -1.2050770   -1.2128421
3  0.0500271  0.1221157 -0.1079779   -0.3094322     -0.4634444     -0.3566471   -0.2071143
4 -0.6993926 -0.4320971 -0.3470576   -0.6377430     -0.6481599     -0.6075269   -0.6022216
5 -1.6153501 -1.5405226 -1.4229163   -1.2719334     -1.2158222     -1.2871831   -1.3205987
6 -0.8242959 -1.0870758 -1.0642967   -0.9731910     -0.9838015     -0.9405128   -0.8536536
> str(abalone_z)
'data.frame':   4177 obs. of  7 variables:
 $ length        : num  -0.574 -1.449 0.05 -0.699 -1.615 ...
 $ diameter      : num  -0.432 -1.44 0.122 -0.432 -1.541 ...
 $ height        : num  -1.064 -1.184 -0.108 -0.347 -1.423 ...
 $ whole_weight  : num  -0.642 -1.23 -0.309 -0.638 -1.272 ...
 $ shucked_weight: num  -0.608 -1.171 -0.463 -0.648 -1.216 ...
 $ viscera_weight: num  -0.726 -1.205 -0.357 -0.608 -1.287 ...
 $ shell_weight  : num  -0.638 -1.213 -0.207 -0.602 -1.321 ...
```

The next step in preparing the data is to add back the rings, female and infant columns:

```
> #attach rings, female and infant columns
> abalone_fullz <- cbind(abalone_z, abalone$rings, abalone$female, abalone$infant)
> names(abalone_fullz)[8:10] <- c("rings", "female", "infant")
```

Next, I added a column for age calculated as the number of rings plus 1.5:

```
> #add age column instead of rings
> abalone_fullz$age <- abalone_fullz$rings + 1.5
```

Finally, I grouped the age observations into three categories: young, adult, and old:

```
> #assign abalone data to the new object
> abalone_new <-abalone_fullz
> #aggregate into 3 groups
> abalone_new$age <- cut(abalone_new$age, breaks = c(0,7,12,100), labels = c("young", "adult", "o
ld"))
> abalone_new$age <-as.factor(abalone_new$age)
```

And removed the initial rings column:

```
> #remove rings column
> abalone_new <- subset(abalone_new, select = -rings)
```

These preparations resulted in the data frame with 4177 observations of 10 variables fully prepared for model fitting.

```
 > #check results
> #abalone_new - full df ready to fit models
> str(abalone_new)
```

```
'data.frame':	4177 obs. of  10 variables:
 $ length        : num  -0.574 -1.449 0.05 -0.699 -1.615 ...
 $ diameter      : num  -0.432 -1.44 0.122 -0.432 -1.541 ...
 $ height        : num  -1.064 -1.184 -0.108 -0.347 -1.423 ...
 $ whole_weight  : num  -0.642 -1.23 -0.309 -0.638 -1.272 ...
 $ shucked_weight: num  -0.608 -1.171 -0.463 -0.648 -1.216 ...
 $ viscera_weight: num  -0.726 -1.205 -0.357 -0.608 -1.287 ...
 $ shell_weight  : num  -0.638 -1.213 -0.207 -0.602 -1.321 ...
 $ female        : num  0 0 1 0 0 0 1 1 0 1 ...
 $ infant        : num  0 0 0 0 1 1 0 0 0 0 ...
 $ age           : Factor w/ 3 levels "young","adult",..: 3 2 2 2 2 2 3 3 2 3 ...
```

Next step is to split the dataset into training and testing parts using the following code:

```
> #Split the dataset into 80% training and  20% testing data
> set.seed(123)
> split = 0.08
> trainIndex <- createDataPartition(abalone_new$age, p=split, list = FALSE)
> abalone_test<- abalone_new[trainIndex, ]
> abalone_train <- abalone_new[-trainIndex, ]
```

The resulting training set contains 3841 observation and test dataset 336 observations of 10 variables.

```
> #Test dataset without the age column
> abalone_testnoage <- abalone_test[-10]
```

I used the same training and testing sets to fit and evaluate two classification algorithms – Naïve Bayes and recursive tree partitioning, as those algorithms support multiclass classifications.

First, Naïve Bayes, fitting the model and generating predictions:

```
> ####Naive Bayes
> set.seed(123)
> nb_model <- naiveBayes(age ~ ., data=abalone_train)
> #Generate predictions
> nb_prediction <- predict(nb_model,newdata = abalone_testnoage)
```

Applying the model to the testing set resulted in the following predictions:

```
> #generate classification table
> nb_table1 <- table(nb_prediction, abalone_test$age)
> nb_table1

nb_prediction young adult old
        young    15    24   0
        adult      1   110  42
        old        0    70  74
```

I used confusionMatrix() to generate performance indicators:

```
> #Evaluate model performance
> confusionMatrix(nb_table1)
Confusion Matrix and Statistics


nb_prediction young adult old
        young    15    24   0
        adult     1   110  42
        old       0    70  74

Overall Statistics

               Accuracy : 0.5923
                 95% CI : (0.5376, 0.6453)
    No Information Rate : 0.6071
    P-Value [Acc > NIR] : 0.7313

                  Kappa : 0.2847
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: young Class: adult Class: old
Sensitivity               0.93750       0.5392     0.6379
Specificity               0.92500       0.6742     0.6818
Pos Pred Value            0.38462       0.7190     0.5139
Neg Pred Value            0.99663       0.4863     0.7812
Prevalence                0.04762       0.6071     0.3452
Detection Rate            0.04464       0.3274     0.2202
Detection Prevalence      0.11607       0.4554     0.4286
Balanced Accuracy         0.93125       0.6067     0.6599
```

The overall classification accuracy of the model was only 59.23%. It means that misclassification rate was quite high - 40.77% of test observations were misclassified (error rate = 1 – accuracy rate). Kappa coefficient reflects the accuracy of the algorithm, adjusted for baseline random chance in the dataset. For this classifier is was equal to only 28.47%.

If we look at each one of the classes, adults class has the highest precision, or positive predictive value, of 71.90%. The young class has the lowest precision rate of 38.462%. The sensitivity, or recall, expresses the probability of correctly labeled observations in a particular class. The model sensitivity was the highest for the young class (93.750%). The model had the most problem identifying observations belonging to the adult class, with the sensitivity of 53.92%. The specificity, which measures the true negative rate, was also highest for the young class (92.50%), and relatively close for the adult (67.42%) and old classes (68.18%).

In the case of the multiclass classification, the interpretation of the performance metrics is not as straightforward as in case of binary classifications, but overall the Naïve Bayes model was not very accurate in classifying observations, especially belonging to the adult and old classes.

Next, I used a recursive partitioning algorithm to build a classification tree for the abalone dataset:

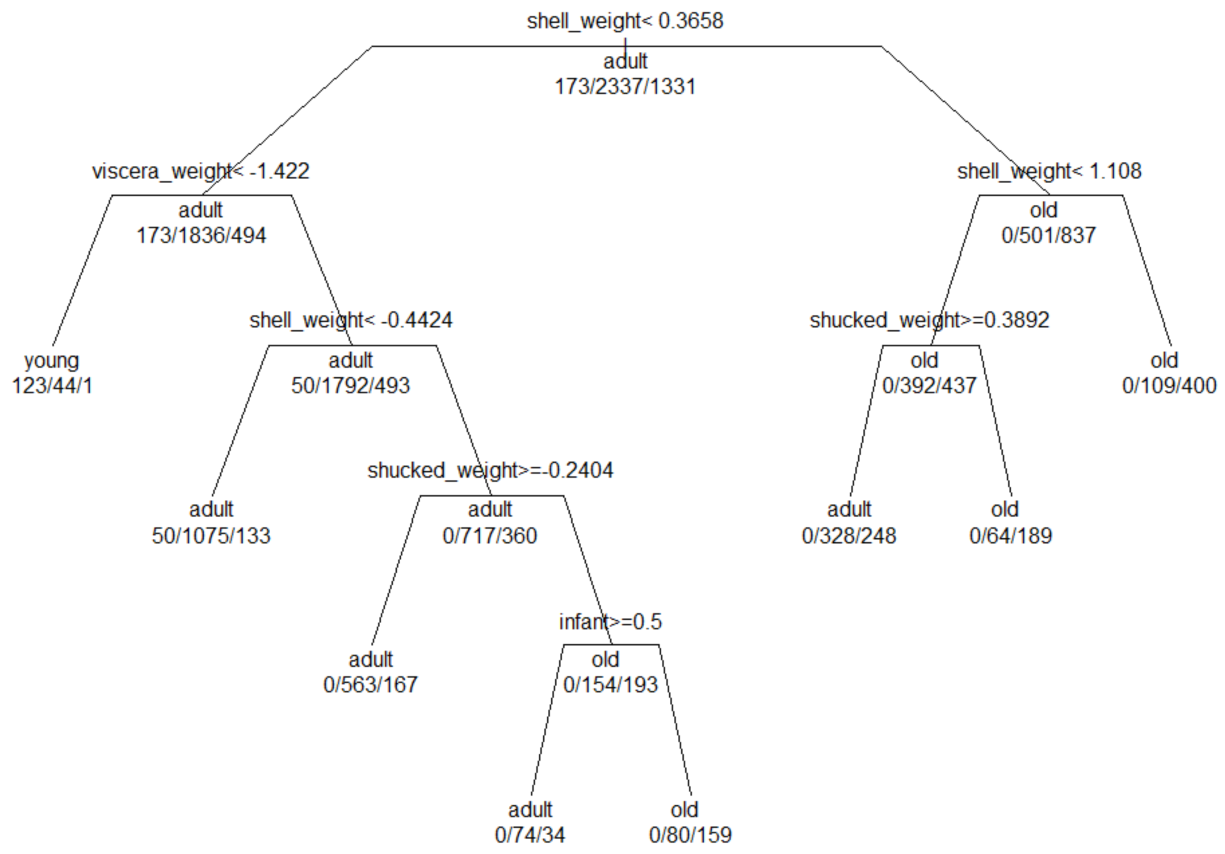
```
> ###Build classification tree model
> set.seed(234)
> abalone_rp <- rpart(age ~ ., data = abalone_train)
```

It resulted in the following model:

```
> #Examine the model
> summary(abalone_rp)
Call:
rpart(formula = age ~ ., data = abalone_train)
  n= 3841

          CP nsplit rel error    xerror       xstd
1 0.22340426      0 1.0000000 1.0000000 0.02011330
2 0.05252660      1 0.7765957 0.7825798 0.01899701
3 0.02659574      2 0.7240691 0.7386968 0.01868394
4 0.01750887      4 0.6708777 0.7234043 0.01856725
5 0.01000000      7 0.6183511 0.6748670 0.01816976

Variable importance
  shell_weight   whole_weight viscera_weight       diameter         length shucked_weight 
height
            19             17             16             15             15              9 
8
        infant
             1
```

(The rest of the output is omitted.)

Tree visualization:



Next, I used the model to generate the prediction on the test dataset:

```
> #Generate predictions
> predict_rp1 <- predict(abalone_rp, abalone_test, type="class")
> table(abalone_test$age, predict_rp1)
       predict_rp1
        young adult old
  young    12     4   0
  adult     4   175  25
  old       0    47  69
```

I used confusionMatrix() function to evaluate the performance of the tree algorithm:

```
> #generate confusion matrix using caret package
> confusionMatrix(table(predict_rp1, abalone_test$age))
Confusion Matrix and Statistics


predict_rp1 young adult old
      young    12     4   0
      adult     4   175  47
      old       0    25  69

Overall Statistics

               Accuracy : 0.7619
                 95% CI : (0.7127, 0.8064)
    No Information Rate : 0.6071
    P-Value [Acc > NIR] : 1.386e-09

                  Kappa : 0.5168
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: young Class: adult Class: old
Sensitivity               0.75000       0.8578     0.5948
Specificity               0.98750       0.6136     0.8864
Pos Pred Value            0.75000       0.7743     0.7340
Neg Pred Value            0.98750       0.7364     0.8058
Prevalence                0.04762       0.6071     0.3452
Detection Rate            0.03571       0.5208     0.2054
Detection Prevalence      0.04762       0.6726     0.2798
Balanced Accuracy         0.86875       0.7357     0.7406
```

Overall, recursive partitioning algorithm showed better results than Naïve Bayes. Rpart model was able to classify abalone data with overall 76.19% accuracy, which means the error rate of 23.81. Precision, or positive predictive value, for this model, was close for all three classes – 75.00% for young, 77.43% for adults and 73.40% for old abalone shells. The probability of correctly labeled observations (sensitivity) was the highest for the adult class 985.78%, for the young class it was 75.00%. However, the sensitivity fell down to 59.48 for the old class. The specificity, or the true negative rate, was high for the young class (98.75%), and lower for the old (88.64%) and the adult (61.36%) classes.

While the decision tree model showed better results on the abalone data set relative to the Naïve Bayes model, it is likely that on a different dataset results might have been different. Outliers in the data affect classification algorithms in various ways. In addition, it is possible that a different grouping of the training data between the tree classes (young, adult and old) might yield different results as well. The above discussion about the accuracy indicators relates only to the performance demonstrated on the abalone dataset.

Next, I applied 10-fold cross-validation to the same Naïve Bayes and recursive partitioning algorithms.

I used the following code for fitting the Naïve Bayes model:

```
> ###Cross Validation
>
> train_control <- trainControl(method="cv", number =10)
>
> #cross-validation with Naive Bayes
> set.seed(345)
> nb_modelCV <- train(age ~ ., data = abalone_train, method = "nb",  trControl=train_control)
There were 50 or more warnings (use warnings() to see the first 50)
>
> nb_modelCV
Naive Bayes

3841 samples
   9 predictor
   3 classes: 'young', 'adult', 'old'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3457, 3457, 3457, 3457, 3457, 3458, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.6053113  0.3118130
   TRUE      0.6217128  0.3386246

Tuning parameter 'fL' 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 fL = 0, usekernel = TRUE and adjust = 1.
```

After that I generated predictions using the new model:

```
> #Generate predictions
> nb_predictionCV <- predict(nb_modelCV,newdata = abalone_testnoage)
Warning messages:
1: In FUN(X[[i]], ...) :
  Numerical 0 probability for all classes with observation 115
2: In FUN(X[[i]], ...) :
  Numerical 0 probability for all classes with observation 216
```

```
3: In FUN(X[[i]], ...) :
  Numerical 0 probability for all classes with observation 248
```

Below are the resulting predictions

```
> #generate classification table
> nb_tableCV <- table(nb_predictionCV, abalone_test$age)
> nb_tableCV

nb_predictionCV young adult old
          young    15    20   0
          adult     1   111  36
          old       0    73  80
```

I used confusionMatrix() evaluate the model's performance:

```
> #Evaluate model performance
> confusionMatrix(nb_tableCV)
Confusion Matrix and Statistics


nb_predictionCV young adult old
          young    15    20   0
          adult     1   111  36
          old       0    73  80

Overall Statistics

               Accuracy : 0.6131
                 95% CI : (0.5587, 0.6655)
    No Information Rate : 0.6071
    P-Value [Acc > NIR] : 0.435

                  Kappa : 0.3217
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: young Class: adult Class: old
Sensitivity               0.93750       0.5441     0.6897
Specificity               0.93750       0.7197     0.6682
Pos Pred Value            0.42857       0.7500     0.5229
Neg Pred Value            0.99668       0.5053     0.8033
Prevalence                0.04762       0.6071     0.3452
Detection Rate            0.04464       0.3304     0.2381
Detection Prevalence      0.10417       0.4405     0.4554
Balanced Accuracy         0.93750       0.6319     0.6789
```

10-fold cross-validation allowed to increase the overall performance of the Naïve Bayes model from 59.23% to 61.31%. Two classes mostly benefited from this increase performance due to more training data available – the adult and the old classes. For the adult class the precision increased to 75.00% (from 71.90%) sensitivity increased to 54.41 % (from 53.92%) and specificity to 71.97% (from 67.42%). There were similar changes for the old class: precision rose to 52.29% (from 51.39%), sensitivity increased to 68.97% (from 63.79%), however, specificity was 66.82% (compared to 68.18 without cross validation).

Next step – using 10-fold cross-validation with the decision tree algorithm. I used the following code to fit the model:

```
> ###Cross Validation with rpart
> set.seed(456)
> rpr_modelCV <- train(age~ ., data =abalone_train, method = "rpart", trControl = train_control)
> rpr_modelCV
CART

3841 samples
   9 predictor
   3 classes: 'young', 'adult', 'old'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3457, 3455, 3456, 3458, 3457, 3457, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.02659574  0.7141364  0.4231724
  0.05252660  0.6964436  0.3850168
  0.22340426  0.6305551  0.1018521

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.02659574.
```

I used the new decision tree model to generate predictions for the testing dataset:

```
> #generate predictions
> rpr_predictionCV <-predict(rpr_modelCV, newdata = abalone_testnoage)
> #generate classification table
> rpr_tableCV <- table(rpr_predictionCV, abalone_test$age)
> rpr_tableCV

rpr_predictionCV young adult old
           young    12     4   0
           adult     4   157  56
           old       0    43  60
```

As the above output shows, misclassifications are still common. For example, the model classified 217 observations as adult class, however, only 157 of them were classified correctly. 4 actually represented the young class, and 56 observations were actually from the old class.

To my surprise, a look at the confusion matrix revealed that the overall accuracy of the model using 10 -fold cross validation decreased compared to the model without cross-validation.

```
> #Evaluate model performance
> confusionMatrix(rpr_tableCV)
Confusion Matrix and Statistics


rpr_predictionCV young adult old
           young    12     4   0
           adult     4   157  56
           old       0    43  60

Overall Statistics

               Accuracy : 0.6815
                 95% CI : (0.6288, 0.7311)
    No Information Rate : 0.6071
    P-Value [Acc > NIR] : 0.002818

                  Kappa : 0.3628
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: young Class: adult Class: old
Sensitivity               0.75000       0.7696     0.5172
Specificity               0.98750       0.5455     0.8045
Pos Pred Value            0.75000       0.7235     0.5825
Neg Pred Value            0.98750       0.6050     0.7597
Prevalence                0.04762       0.6071     0.3452
Detection Rate            0.03571       0.4673     0.1786
Detection Prevalence      0.04762       0.6458     0.3065
Balanced Accuracy         0.86875       0.6575     0.6609
```

As the above output shows, the overall accuracy went down to 68.15 % (from 76.18% earlier). The statistics by class show that the decrease in performance was most significant for the two classes that didn’t fare originally – the adult and the old classes. The balances accuracy for the old class went down to 66.09% (from 74.06) and for the adult class to 65.75% (from 73.57%).

These negative changes in the model performance might be potentially connected with the sensitivity of the chosen decision tree algorithm to the outliers/noise, and the quality of the observations. In this case, increasing the size of the training data available did not result in improvements.

Theoretically, increasing the size of training data by increasing the number of folds, should provide a better model fit. However, I think that this statement would be correct only for the models with a linear learning curve. For the algorithms that I used in this assignment increasing the number of folds (to the K-1) might not result in the significant improvements.

Another potential direction for improving the results would be to look at the initial data and evaluate the potential of grouping the ages differently, treating outliers, applying different models etc.