Evaluating Performance Measures

Code

The objective of this assignment is to compare two different classification algorithms using accuracy and performance metrics. I'm going to use K-NN and Naive Bayes to to predict the age of an abalone using abalone features. Abalone are shell fish that are popular to eat in many countries, especially raw in a sashimi spread. The rings attribute corresponds to an abalone's age in years (after the abalone reaches 1 to 1.5 years of age). The process of determining an abalone's age is tedious and time consuming, so using classification machine learning might be useful for predicting an abalone's age.

Data was taken from this website https://archive.ics.uci.edu/ml/datasets/Abalone (https://archive.ics.uci.edu/ml/datasets/Abalone)

Here are the attribute descriptions:

Sex / nominal / – / M, F, and I (infant) Length / continuous / mm / Longest shell measurement Diameter / continuous / mm / perpendicular to length Height / continuous / mm / with meat in shell Whole weight / continuous / grams / whole abalone Shucked weight / continuous / grams / weight of meat Viscera weight / continuous / grams / gut weight (after bleeding) Shell weight / continuous / grams / after being dried Rings / integer / – / +1.5 gives the age in years

Hide

Hide

```
 abalone <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data"), header = FALSE, sep = ",")
colnames(abalone) <- c("sex", "length", 'diameter', 'height', 'whole_weight', 'shucked_wieght', 'viscera_wieght', 'shell_weight', 'rings' )
```

Hide

Hide

```
summary(abalone)
```

```
 sex          length         diameter          height
 F:1307   Min.   :0.075   Min.   :0.0550   Min.   :0.0000
 I:1342   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150
 M:1528   Median :0.545   Median :0.4250   Median :0.1400
          Mean   :0.524   Mean   :0.4079   Mean   :0.1395
          3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650
          Max.   :0.815   Max.   :0.6500   Max.   :1.1300
  whole_weight    shucked_wieght   viscera_wieght
 Min.   :0.0020   Min.   :0.0010   Min.   :0.0005
 1st Qu.:0.4415   1st Qu.:0.1860   1st Qu.:0.0935
 Median :0.7995   Median :0.3360   Median :0.1710
 Mean   :0.8287   Mean   :0.3594   Mean   :0.1806
 3rd Qu.:1.1530   3rd Qu.:0.5020   3rd Qu.:0.2530
 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
```

Hide

Hide

```
str(abalone)
```

```
'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_wieght: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
 $ viscera_wieght: 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 ...
```

Hide

Hide

```
summary(abalone$rings)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   8.000   9.000   9.934  11.000  29.000 
```

As shown above, the “rings” variable has a range between 1-29. This is the variable that we want to predict, and predicting this many levels might not give us the insight we’re looking for. I suspect that there’s an optimal age range for harvesting abalones for consumption. While I don’t know this age range, this project could be adjusted with the sought-after age range inserted. For now, we’ll break the rings variable into 3 levels" “young” for abalones less than 8, “adult” for abalones between 8-11, and “old” for abalones older than 11.

Hide

Hide

```
abalone$rings <- as.numeric(abalone$rings)
abalone$rings <- cut(abalone$rings, br=c(-1,8,11,35), labels = c("young", 'adult', 'o
ld'))
abalone$rings <- as.factor(abalone$rings)
summary(abalone$rings)
```

```
young adult   old
 1407  1810   960
```

I’m going to create a couple of different classification models, and then compare them using accuracy and performance metrics. I’ll start with a KNN classification algorithm. Because KNN requires all numeric variables for prediction, I’m going to remove the “sex” variable.

Hide

Hide

```
z <- abalone
z$sex <- NULL
```

I’ll now normalize the data using min max normalization

Hide

Hide

```
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
z[1:7] <- as.data.frame(lapply(z[1:7], normalize))
summary(z$shucked_wieght)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.1244  0.2253  0.2410  0.3369  1.0000
```

Now each variable has a min of 0 and a max of 1. We’ll now split the data into training and testing sets.

Hide

Hide

```
ind <- sample(2, nrow(z), replace=TRUE, prob=c(0.7, 0.3))
KNNtrain <- z[ind==1,]
KNNtest <- z[ind==2,]
```

Now we run the model. I'm going to make k equal to the square root of 2918, the number of observations in the training set.

Hide

Hide

```
library(class)
KNNpred <- knn(train = KNNtrain[1:7], test = KNNtest[1:7], cl = KNNtrain$rings, k = 5
4)
```

Let's see how the model does on the test data.

Hide

Hide

```
library("gmodels")
CrossTable(x = KNNtest$rings, y = KNNpred, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1276


              | KNNpred
KNNtest$rings |      young |      adult |        old |  Row Total |
--------------|------------|------------|------------|------------|
        young |        325 |         93 |          0 |        418 |
              |      0.778 |      0.222 |      0.000 |      0.328 |
              |      0.747 |      0.135 |      0.000 |            |
              |      0.255 |      0.073 |      0.000 |            |
--------------|------------|------------|------------|------------|
        adult |         90 |        418 |         35 |        543 |
              |      0.166 |      0.770 |      0.064 |      0.426 |
              |      0.207 |      0.607 |      0.230 |            |
              |      0.071 |      0.328 |      0.027 |            |
--------------|------------|------------|------------|------------|
          old |         20 |        178 |        117 |        315 |
              |      0.063 |      0.565 |      0.371 |      0.247 |
              |      0.046 |      0.258 |      0.770 |            |
              |      0.016 |      0.139 |      0.092 |            |
--------------|------------|------------|------------|------------|
 Column Total |        435 |        689 |        152 |       1276 |
              |      0.341 |      0.540 |      0.119 |            |
--------------|------------|------------|------------|------------|
```

Hide

Hide

```
(328+451+97)/((84+2+26+91+21+159)+(328+451+97))
```

```
[1] 0.6957903
```

This KNN classifier predicted the abalone age with 69% accuracy - likely not accurate enough for an abalone harvester to trust. Before moving on to more specific accuracy and performance tests I'm going to try a smaller k value and see if it improves the accuracy.

Hide

Hide

```
library(class)
KNNpred <- knn(train = KNNtrain[1:7], test = KNNtest[1:7], cl = KNNtrain$rings, k = 1
0)
```

Hide

Hide

```
library("gmodels")
CrossTable(x = KNNtest$rings, y = KNNpred, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1276


              | KNNpred
KNNtest$rings |     young |     adult |       old | Row Total |
--------------|-----------|-----------|-----------|-----------|
        young |       318 |        96 |         4 |       418 |
              |     0.761 |     0.230 |     0.010 |     0.328 |
              |     0.741 |     0.150 |     0.019 |           |
              |     0.249 |     0.075 |     0.003 |           |
--------------|-----------|-----------|-----------|-----------|
        adult |        94 |       382 |        67 |       543 |
              |     0.173 |     0.703 |     0.123 |     0.426 |
              |     0.219 |     0.597 |     0.324 |           |
              |     0.074 |     0.299 |     0.053 |           |
--------------|-----------|-----------|-----------|-----------|
          old |        17 |       162 |       136 |       315 |
              |     0.054 |     0.514 |     0.432 |     0.247 |
              |     0.040 |     0.253 |     0.657 |           |
              |     0.013 |     0.127 |     0.107 |           |
--------------|-----------|-----------|-----------|-----------|
 Column Total |       429 |       640 |       207 |      1276 |
              |     0.336 |     0.502 |     0.162 |           |
--------------|-----------|-----------|-----------|-----------|
```

Hide

Hide

```
(313+422+135)/((313+422+135)+(94+7+57+89+17+125))
```

```
[1] 0.6910246
```

This model has just about the same predictive power on the test set. This can also be shown in the confusion matrix below:

Hide

Hide

```
library(caret)
```

```
package ‘caret’ was built under R version 3.3.2Loading required package: lattice
Loading required package: ggplot2
package ‘ggplot2’ was built under R version 3.3.2
```

Hide

Hide

```
confusionMatrix(KNNpred, KNNtest$rings)
```

```
Confusion Matrix and Statistics

          Reference
Prediction young adult old
     young   318    94  17
     adult    96   382 162
     old       4    67 136

Overall Statistics

               Accuracy : 0.6552
                 95% CI : (0.6284, 0.6813)
    No Information Rate : 0.4255
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4581
 Mcnemar's Test P-Value : 2.749e-10

Statistics by Class:

                     Class: young Class: adult Class: old
Sensitivity                0.7608       0.7035     0.4317
Specificity                0.8706       0.6480     0.9261
Pos Pred Value             0.7413       0.5969     0.6570
Neg Pred Value             0.8819       0.7469     0.8326
Prevalence                 0.3276       0.4255     0.2469
Detection Rate             0.2492       0.2994     0.1066
Detection Prevalence       0.3362       0.5016     0.1622
Balanced Accuracy          0.8157       0.6758     0.6789
```

The misclassification rate is 1 minus the accuracy, shown below.

Hide

Hide

```
1-0.691
```

```
[1] 0.309
```

Let's now create a naive bayes classifier for the same data.

Hide

Hide

```
NBtrain <- KNNtrain
NBtest <- KNNtest
```

Hide

Hide

```
library(e1071)
```

```
package 'e1071' was built under R version 3.3.2
```

Hide

Hide

```
model <- naiveBayes(rings ~., data = NBtrain)
model
```

```


Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
    young     adult       old 
0.3409169 0.4367459 0.2223371 

Conditional probabilities:
       length
Y            [,1]      [,2]
  young 0.4716203 0.1490067
  adult 0.6712654 0.1177351
  old   0.6912424 0.1092433

       diameter
Y            [,1]      [,2]
  young 0.4507906 0.1502746
```

```
  adult 0.6580688 0.1193139
  old   0.6863657 0.1123703

      height
Y            [,1]       [,2]
  young 0.09493365 0.04008618
  adult 0.13448953 0.02663204
  old   0.14618234 0.02554316

      whole_weight
Y           [,1]      [,2]
  young 0.1544802 0.1083561
  adult 0.3494055 0.1509557
  old   0.3924864 0.1602648

      shucked_wieght
Y           [,1]       [,2]
  young 0.1337806 0.09876195
  adult 0.2970106 0.14060814
  old   0.2962226 0.13836111

      viscera_wieght
Y           [,1]       [,2]
  young 0.1241098 0.09026025
  adult 0.2866922 0.12776408
  old   0.3145439 0.13646942

      shell_weight
Y           [,1]       [,2]
  young 0.1206878 0.08019425
  adult 0.2733141 0.10873391
  old   0.3365036 0.13167884
```

Hide

Hide

```
pred <- predict(model, NBtest)
print(confusionMatrix(pred,NBtest$rings))
```

```
Confusion Matrix and Statistics

          Reference
Prediction young adult old
     young   336   126  47
     adult    76   285 139
     old       6   132 129

Overall Statistics

               Accuracy : 0.5878
                 95% CI : (0.5602, 0.6149)
    No Information Rate : 0.4255
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3667
 Mcnemar's Test P-Value : 1.32e-09

Statistics by Class:

                     Class: young Class: adult Class: old
Sensitivity                0.8038       0.5249     0.4095
Specificity                0.7984       0.7067     0.8564
Pos Pred Value             0.6601       0.5700     0.4831
Neg Pred Value             0.8931       0.6675     0.8157
Prevalence                 0.3276       0.4255     0.2469
Detection Rate             0.2633       0.2234     0.1011
Detection Prevalence       0.3989       0.3918     0.2092
Balanced Accuracy          0.8011       0.6158     0.6330
```

The accuracy rate for the naive bayes model predicting the test set is only about 59%, which makes the misclassification rate approx. 41%.

While it's likely that neither algorithm is adequate for predicting the abalone age, the KNN model is more accurate so far.

Let's try a bootstrapping method for further model evaluation.

Hide

Hide

```
library(caret)
train_control <- trainControl(method='boot', number = 100)

trModel <- train(rings~., data = z, trControl=train_control, method="nb")
```

Hide

Hide

```
print(trModel)
```

```
Naive Bayes

4177 samples
   7 predictor
   3 classes: 'young', 'adult', 'old'

No pre-processing
Resampling: Bootstrapped (100 reps)
Summary of sample sizes: 4177, 4177, 4177, 4177, 4177, 4177, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.5810706  0.3529160
   TRUE      0.6069231  0.3799894

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.
```

Hide

Hide

```
trModel2 <- train(rings~., data = z, trControl=train_control, method="knn")
```

Hide

Hide

```
print(trModel2)
```

```
k-Nearest Neighbors

4177 samples
   7 predictor
   3 classes: 'young', 'adult', 'old'

No pre-processing
Resampling: Bootstrapped (100 reps)
Summary of sample sizes: 4177, 4177, 4177, 4177, 4177, 4177, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.6222626  0.4091962
  7  0.6347705  0.4266619
  9  0.6447592  0.4406755

Accuracy was used to select the optimal model using  the
 largest value.
The final value used for the model was k = 9.
```

The bootstrapping method indicates that the KNN model might be slightly more accurate for classifying the data. It was also estimated that the most effective k value for the KNN model would be 9. We used 10 for our model.

Let's now do 10-fold cross validation to evaluate the models.

Hide

Hide

```
control = trainControl(method="repeatedcv", number=10, repeats=3)
model5 <- train(rings~., data = KNNtrain, method = "knn", preProcess="scale", trContr
ol=control)
model5
```

```
k-Nearest Neighbors

2901 samples
   7 predictor
   3 classes: 'young', 'adult', 'old'

Pre-processing: scaled (7)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2612, 2610, 2610, 2610, 2611, 2611, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.6564605  0.4565599
  7  0.6689747  0.4743428
  9  0.6789763  0.4889676

Accuracy was used to select the optimal model using  the
 largest value.
The final value used for the model was k = 9.
```

The 10-fold cross validation method indicates that the optimal model for KNN is one with k = 9 (same as what the bootstrap method predicted).

The 10-fold cross validation method indicates that the optimal model for Naive Bayes is a model with fL = 0, usekernal = TRUE and adjust = 1.

The cross-validation method confirms that the KNN method is more effective for this data set than Naive Bayes.

Let's create the new models with the suggested parameters.

Hide

Hide

```
library(class)
KNNpred <- knn(train = KNNtrain[1:7], test = KNNtest[1:7], cl = KNNtrain$rings, k = 9
)
library(caret)
confusionMatrix(KNNpred, KNNtest$rings)
```

```
Confusion Matrix and Statistics

          Reference
Prediction young adult old
     young   322    96  18
     adult    90   375 164
     old       6    72 133

Overall Statistics

               Accuracy : 0.6505
                 95% CI : (0.6236, 0.6767)
    No Information Rate : 0.4255
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.4517
 Mcnemar's Test P-Value : 3.9e-09

Statistics by Class:

                     Class: young Class: adult Class: old
Sensitivity                0.7703       0.6906     0.4222
Specificity                0.8671       0.6535     0.9188
Pos Pred Value             0.7385       0.5962     0.6303
Neg Pred Value             0.8857       0.7403     0.8291
Prevalence                 0.3276       0.4255     0.2469
Detection Rate             0.2524       0.2939     0.1042
Detection Prevalence       0.3417       0.4929     0.1654
Balanced Accuracy          0.8187       0.6720     0.6705
```

With k = 9 the model was about 69% accurate in predicting the test data set.

Hide

Hide

```
library(e1071)
```

Hide

Hide

```
model <- naiveBayes(rings ~., data = NBtrain, fL = 0, usekernal = TRUE, adjust = 1)
model
```

```


Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace, fL = 0, usekernal = TRUE,
```

```
    adjust = 1)

A-priori probabilities:
Y
    young     adult       old 
0.3409169 0.4367459 0.2223371 

Conditional probabilities:
       length
Y            [,1]      [,2]
  young 0.4716203 0.1490067
  adult 0.6712654 0.1177351
  old   0.6912424 0.1092433

       diameter
Y            [,1]      [,2]
  young 0.4507906 0.1502746
  adult 0.6580688 0.1193139
  old   0.6863657 0.1123703

       height
Y             [,1]       [,2]
  young 0.09493365 0.04008618
  adult 0.13448953 0.02663204
  old   0.14618234 0.02554316

       whole_weight
Y            [,1]      [,2]
  young 0.1544802 0.1083561
  adult 0.3494055 0.1509557
  old   0.3924864 0.1602648

       shucked_wieght
Y            [,1]       [,2]
  young 0.1337806 0.09876195
  adult 0.2970106 0.14060814
  old   0.2962226 0.13836111

       viscera_wieght
Y            [,1]       [,2]
  young 0.1241098 0.09026025
  adult 0.2866922 0.12776408
  old   0.3145439 0.13646942

       shell_weight
Y            [,1]       [,2]
  young 0.1206878 0.08019425
  adult 0.2733141 0.10873391
  old   0.3365036 0.13167884
```

Hide

Hide

Hide

```
pred <- predict(model, NBtest)
print(confusionMatrix(pred,NBtest$rings))
```

```
Confusion Matrix and Statistics

          Reference
Prediction young adult old
     young   336   126  47
     adult    76   285 139
     old       6   132 129

Overall Statistics

               Accuracy : 0.5878
                 95% CI : (0.5602, 0.6149)
    No Information Rate : 0.4255
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3667
 Mcnemar's Test P-Value : 1.32e-09

Statistics by Class:

                     Class: young Class: adult Class: old
Sensitivity                0.8038       0.5249     0.4095
Specificity                0.7984       0.7067     0.8564
Pos Pred Value             0.6601       0.5700     0.4831
Neg Pred Value             0.8931       0.6675     0.8157
Prevalence                 0.3276       0.4255     0.2469
Detection Rate             0.2633       0.2234     0.1011
Detection Prevalence       0.3989       0.3918     0.2092
Balanced Accuracy          0.8011       0.6158     0.6330
```

With the suggested parameters given from the 10-fold validation, the naive bayes algorithm is about 59% accurate.

The models trained by the 10-fold validation have almost equal accuracy to the models I originally created, when testing on the test data set. My concern with this project is that the parameters I originally used didn't differ much from the suggested model in 10-fold validation.

Hide

Hide

```
control14 = trainControl(method="repeatedcv", number=10, repeats=3)
model7 <- train(rings~., data = KNNtrain, method = "rf", preProcess="scale", trContro
l=control14)
```

```
randomForest 4.6-12
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object is masked from 'package:ggplot2':

    margin
```

Hide

Hide

```
model7
```

```
Random Forest

2901 samples
   7 predictor
   3 classes: 'young', 'adult', 'old'

Pre-processing: scaled (7)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2611, 2611, 2611, 2612, 2611, 2610, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.6886180  0.5076597
  4     0.6826480  0.4994915
  7     0.6813857  0.4980003

Accuracy was used to select the optimal model using  the
 largest value.
The final value used for the model was mtry = 2.
```

As shown above, the optimal random forest model is expected to perform at about 67%. It seems like the machine learning algorithms are having a difficult learning enough from the abalone features to accurately predict the age of the abalone. I would suspect that an abalone harvester would want a more accurate model before he or she could trust it in a commercial setting. Therefore, better data might be necessary.