Assignment 7, Evaluating Performance Measures

MSDS 680 - Machine Learning

Alla Topp

Objective: to predict the age in years of abalone shells (rings) using physical measurements such as length diameter, whole weight, etc.

Abalone is a shellfish. Rings are formed in the shell as the abalone grows, which is about one ring per year (except the first year). To get to the rings, shells will be cut, polished, stained, and examined through a microscope. Some rings are difficult to identify so researchers believe that it is reasonable to add 1.5 to the rings count when estimating the age of the abalones. The method is quite time-consuming and tedious.

I'm going to use K-NN and Naive Bayes to predict the age of an abalone using abalone features. Data was taken from this website https://archive.ics.uci.edu/ml/datasets/Abalone

We start with importing the data into RStudio and do some exploration analytics:

```
#Alla Topp
#Week 7, Evaluating Performance Measures
#using libraries
library(class)
library(gmodels)
library(caret)
library(e1071)
library(klaR)

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' )
summary(abalone)
str(abalone)
```

```
> summary(abalone)
 sex          length         diameter          height        whole_weight     shucked_wieght    viscera_wieght
 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             age
 Min.   :0.0015   Min.   : 1.000   Min.   : 2.50
 1st Qu.:0.1300   1st Qu.: 8.000   1st Qu.: 9.50
 Median :0.2340   Median : 9.000   Median :10.50
 Mean   :0.2388   Mean   : 9.934   Mean   :11.43
 3rd Qu.:0.3290   3rd Qu.:11.000   3rd Qu.:12.50
 Max.   :1.0050   Max.   :29.000   Max.   :30.50
> str(abalone)
'data.frame':	4177 obs. of  10 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 ...
 $ age           : num  16.5 8.5 10.5 11.5 8.5 9.5 21.5 17.5 10.5 20.5 ...
```

We want to look for a better and more simple methodology to predict the age of the abalone. From the data set, the rings variable ranges from 1 to 29. We are going to break the rings variable into 3 levels" "young" for abalones less than 7, "adult" for abalones between 7-12, and "old" for abalones older than 12. The age is calculated by adding 1.5 to rings. Also, we ae going to remove after grouping.

```
#creating new column age as was described in the assignment
abalone$age <- abalone$rings+1.5 #create a new variable 'age'
abalone_new <- abalone #assign original abalone to the new object that includes 'age'
abalone_new$age <- cut(abalone_new$age, breaks = c(0,7,12,100),
                       labels = c("young","adult","old")) # aggregrate into 3 groups
abalone_new$age <- as.factor(abalone_new$age)
abalone_new <- subset(abalone_new, select = -rings) # remove Rings otherwise the output will
summary(abalone_new$age)
```

```
> summary(abalone_new$age)
young adult   old
  189  2541  1447
```

	sex	length	diameter	height	whole_weight	shucked_wiegt	viscera_wieght	shell_weight	age
1	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	old
2	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	adult
3	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	adult
4	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	adult

I'll start with a KNN classification algorithm. Because KNN requires all numeric variables for prediction, I'm going to remove the "sex" variable and normalize the data using min max normalization.

```
#Knn
knn_abalone <- abalone_new #creating new df without column sex
knn_abalone$sex <- NULL #removing column sex

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

```
> summary(knn_abalone$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. Our split will be 70/30. 70% is training set and 30% will be testing set.

```
#splitting the data to training and testing set
set.seed(555)
ind <- sample(2, nrow(knn_abalone), replace=TRUE, prob=c(0.7, 0.3)) #splitting to 70/30
KNN_train <- knn_abalone[ind==1,] #training set containing 70% of data
KNN_test <- knn_abalone[ind==2,] #testing set containing 30% data
```

Now we run the model. I'm going to make k equal to the square root of 2924, the number of observations in the training set.

```
#KNN model with k=54 based on the train set
KNN_pred <- knn(train = KNN_train[1:7], test = KNN_test[1:7], cl = KNN_train$age, k = 54)
CrossTable(x = KNN_test$age, y = KNN_pred, prop.chisq = FALSE) #producing cross table
confusionMatrix(table(KNN_test$age, KNN_pred)) #producing confusion matrix
```

```
Total Observations in Table:  1207


             | KNN_pred 
KNN_test$age |     young |     adult |       old | Row Total | 
-------------|-----------|-----------|-----------|-----------|
       young |        27 |        24 |         0 |        51 | 
             |     0.529 |     0.471 |     0.000 |     0.042 | 
             |     0.871 |     0.028 |     0.000 |           | 
             |     0.022 |     0.020 |     0.000 |           | 
-------------|-----------|-----------|-----------|-----------|
       adult |         4 |       638 |        86 |       728 | 
             |     0.005 |     0.876 |     0.118 |     0.603 | 
             |     0.129 |     0.745 |     0.269 |           | 
             |     0.003 |     0.529 |     0.071 |           | 
-------------|-----------|-----------|-----------|-----------|
         old |         0 |       194 |       234 |       428 | 
             |     0.000 |     0.453 |     0.547 |     0.355 | 
             |     0.000 |     0.227 |     0.731 |           | 
             |     0.000 |     0.161 |     0.194 |           | 
-------------|-----------|-----------|-----------|-----------|
Column Total |        31 |       856 |       320 |      1207 | 
             |     0.026 |     0.709 |     0.265 |           | 
-------------|-----------|-----------|-----------|-----------|
```

```
> confusionMatrix(table(KNN_test$age, KNN_pred)) #producing confusion matrix
Confusion Matrix and Statistics

       KNN_pred
        young adult old
  young    27    24   0
  adult     4   638  86
  old       0   194 234

Overall Statistics
                                          
               Accuracy : 0.7448          
                 95% CI : (0.7192, 0.7692)
    No Information Rate : 0.7092          
    P-Value [Acc > NIR] : 0.003229        
                                          
                  Kappa : 0.4652          
 Mcnemar's Test P-Value : NA              

Statistics by Class:

                     Class: young Class: adult Class: old
Sensitivity               0.87097       0.7453     0.7312
Specificity               0.97959       0.7436     0.7813
Pos Pred Value            0.52941       0.8764     0.5467
Neg Pred Value            0.99654       0.5449     0.8896
Prevalence                0.02568       0.7092     0.2651
Detection Rate            0.02237       0.5286     0.1939
Detection Prevalence      0.04225       0.6031     0.3546
Balanced Accuracy         0.92528       0.7445     0.7563
```

This KNN classifier predicted the abalone age with 74% accuracy - likely not accurate enough for an abalone harvester to trust, but it's still pretty high. After this I tried a few other k values but looks like 54 has the best outcome.

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

```
> 1-.7448
[1] 0.2552
```

Let's now create a Naive Bayes classifier for the same data.

```
#Naive Bayes
NB_train <- KNN_train #using the same training set for the model
NB_test <- KNN_test #using the same test set for the NB model

#model
model <- naiveBayes(age ~., data = NB_train)
model
pred <- predict(model, NB_test)
print(confusionMatrix(pred,NB_test$age))
```

```
> model

Naive Bayes Classifier for Discrete Predictors

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

A-priori probabilities:
Y
     young      adult        old 
0.04646465 0.61043771 0.34309764 

Conditional probabilities:
       length
Y            [,1]      [,2]
  young 0.2407462 0.0937278
  adult 0.5857359 0.1427804
  old   0.6963239 0.1130213

       diameter
Y            [,1]       [,2]
  young 0.2225064 0.08958661
  adult 0.5685085 0.14568926
  old   0.6892488 0.11595103
```

```
> print(confusionMatrix(pred,NB_test$age))
Confusion Matrix and Statistics

          Reference
Prediction young adult old
     young    47    95   2
     adult     4   422 130
     old       0   211 296

Overall Statistics

               Accuracy : 0.6338
                 95% CI : (0.6059, 0.661)
    No Information Rate : 0.6031
    P-Value [Acc > NIR] : 0.01557

                  Kappa : 0.3555
 Mcnemar's Test P-Value : < 2e-16

Statistics by Class:

                     Class: young Class: adult Class: old
Sensitivity               0.92157       0.5797     0.6916
Specificity               0.91609       0.7203     0.7291
Pos Pred Value            0.32639       0.7590     0.5838
Neg Pred Value            0.99624       0.5300     0.8114
Prevalence                0.04225       0.6031     0.3546
Detection Rate            0.03894       0.3496     0.2452
Detection Prevalence      0.11930       0.4606     0.4200
Balanced Accuracy         0.91883       0.6500     0.7104
```

The accuracy rate for the naive bayes model predicting the test set is only about 63%, which makes the misclassification rate approximately 37%. While it's likely that neither algorithm is adequate for predicting the abalone age, the KNN model is more accurate so far.

Now we are going to use 10-fold cross validation for training the classifiers.

```
#10-fold cross validation
train_control <- trainControl(method = "cv", number = 10)
cv_model <- train(age~., data = knn_abalone, trControl=train_control,method ="nb")
print(cv_model)
```

```
> print(cv_model)
Naive Bayes

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

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3759, 3760, 3759, 3760, 3760, 3760, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.6143204  0.3214431
   TRUE      0.6296480  0.3444354

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.
```

We can see that the accuracy of the model shows approximately 63% accuracy which is lower than both previous models showed. We will try Repeated 10-fold Cross Validation with 3 repeats to estimate Naive Bayes on our dataset.

```
#Repeated k-fold Cross Validation
train_control2 = trainControl(method="repeatedcv", number=10, repeats=3)
cv_model2 <- train(age~., data = KNN_train, method = "knn", preProcess="scale", trControl=train_control2)
cv_model2
```

```
> cv_model2
k-Nearest Neighbors

2970 samples
   7 predictor
   3 classes: 'young', 'adult', 'old'

Pre-processing: scaled (7)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2673, 2673, 2672, 2673, 2673, 2673, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.7297276  0.4509303
  7  0.7324272  0.4539505
  9  0.7381625  0.4614362

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. The cross-validation method confirms that the KNN method is more effective for this data set than Naive Bayes. 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. Looks like we don`t go over 74% accuracy with this data set which means that the machine learning algorithms are having a difficult learning enough from the abalone features to accurately predict the age of the abalone. Using different data set definitely affect the results of the models, so classifiers would work much better with specific data sets. The number of folds does affect the result and the performance of the classifier.

References

(n.d.). Retrieved from https://archive.ics.uci.edu/ml/datasets/Abalone

Brownlee, J. (2019, August 22). How To Estimate Model Accuracy in R Using The Caret Package. Retrieved from https://machinelearningmastery.com/how-to-estimate-model-accuracy-in-r-using-the-caret-package/

Sign In. (n.d.). Retrieved from https://rpubs.com/Billyhansen6/318406

Yu-Wei, C. (2015). Machine learning with R cookbook explore over 110 recipes to analyze data and build predictive models with the simple and easy-to-use R code. Packt Publishing.