Lipscomb University MSDS 5213: HW1 - KNN

Olivia Samples

6/22/2020

3.1 Dataset description

The description of the data can be found here http://archive.ics.uci.edu/ml/datasets/banknote+authentication

VARIABLE DESCRIPTIONS:

1. variance of Wavelet Transformed image (continuous)
2. skewness of Wavelet Transformed image (continuous)
3. curtosis of Wavelet Transformed image (continuous)
4. entropy of image (continuous)
5. class (integer)

3.2 Your tasks

- split the data into two sets (train (80% and test 20%)
- use the train method from the caret library to figure out which value of k (for knn) will work best with the data (10 points)
- test the best model with the test data (10 points)
- produce a confusion matrix (10 points)

Using RMarkDown, submit your work in .html or .pdf, including list of the commands you have used as well as the output generated with each command if any. (20 points)

R Markdown

Start by loading the class library.

```
library(class)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

Read the csv file:

```
banknote <- read.csv(file='/Users/osamples/Documents/Lipscomb/MSDS 5213/Assignments/banknote.csv', head
```

Let's create 80-20 split of the data. 80% training data and 20% testing data.

```
split_size <- 0.8
train_size <- floor(nrow(banknote) * split_size)
set.seed(1)
train_indices <- sample(1:nrow(banknote), train_size)
```

Extract the class label from the training data. V5 represents the fifth column since we read file without a header.

```
cl = banknote[train_indices,]$V5
```

Extract the training data and the testing data. Then run the knn with $k = 30$.

```
train = subset(banknote[train_indices,], select = c(V1, V2, V3, V4))
test = subset(banknote[-train_indices,], select = c(V1, V2, V3, V4))
pred <- knn(train,test,cl, k = 30, prob = FALSE, use.all = TRUE)
```

To see the confusion matrix, let us compare what is in pred with what is in data[-train_indices;] column 5.

```
confusionMatrix(pred, factor(banknote[-train_indices,]$V5))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 149   0
##          1   3 123
##
##                Accuracy : 0.9891
##                  95% CI : (0.9685, 0.9977)
##     No Information Rate : 0.5527
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.978
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 0.9803
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9762
##              Prevalence : 0.5527
##          Detection Rate : 0.5418
##    Detection Prevalence : 0.5418
##       Balanced Accuracy : 0.9901
##
##        'Positive' Class : 0
##
```

Sometimes we do not know ahead of time the appropriate setup for the value of k to be used. To handle this issue we need to use the train() method from the caret packages.

Now we are ready to find the best k value.

```
ktune <- train(train, factor(cl), method ="knn", tuneGrid =
  data.frame(.k = 1:30), trControl = trainControl(method = "cv"))
```

To see the result of the training print the following:

```
ktune
```

```
## k-Nearest Neighbors
##
## 1097 samples
##    4 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 987, 987, 988, 987, 987, 987, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  1.0000000  1.0000000
##    2  1.0000000  1.0000000
##    3  1.0000000  1.0000000
##    4  1.0000000  1.0000000
##    5  1.0000000  1.0000000
##    6  1.0000000  1.0000000
##    7  1.0000000  1.0000000
##    8  0.9990909  0.9981636
##    9  1.0000000  1.0000000
##   10  0.9981735  0.9963064
##   11  0.9963470  0.9926283
##   12  0.9954379  0.9907919
##   13  0.9936197  0.9871192
##   14  0.9936197  0.9871192
##   15  0.9936197  0.9871192
##   16  0.9936197  0.9871192
##   17  0.9936197  0.9871192
##   18  0.9936197  0.9871192
##   19  0.9936197  0.9871192
##   20  0.9936197  0.9871192
##   21  0.9936197  0.9871192
##   22  0.9936197  0.9871192
##   23  0.9936197  0.9871192
##   24  0.9936197  0.9871192
##   25  0.9936197  0.9871192
##   26  0.9936197  0.9871192
##   27  0.9936197  0.9871192
##   28  0.9936197  0.9871192
##   29  0.9936197  0.9871192
##   30  0.9936197  0.9871192
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

You can use the result of the training to perform prediction.

```
pred2 <- predict(ktune,test)
```

Now, let's compare this prediction to the last by comparing the confusion matrices.

```r
confusionMatrix(pred2, factor(banknote[-train_indices,]$V5))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 152   0
##          1   0 123
##
##                Accuracy : 1
##                  95% CI : (0.9867, 1)
##     No Information Rate : 0.5527
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5527
##          Detection Rate : 0.5527
##    Detection Prevalence : 0.5527
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

In both instances, the model pretty accurately predicted the classes. However, after using the train method from the carrot library to determine which value of k would work best for the data, we received a 100% accurate test. And so, the best value for k is 3, rather than 30.

In R, most of the machine learning models can be saved in this way.

```r
saveRDS(ktune, file="ktune.rds")
```

This allows you to not have to train from the data each time you need to do classification. You can later load the model for use in this way.