Classification K-Nearest-Neighbors

Chris Bentz

16/01/2023

Load Packages

If the libraries are not installed yet, you need to install them using, for example, the command: install.packages("ggplot2"). For the Hrate package this is different, since it comes from github. The devtools library needs to be installed, and then the install_github() function is used.

```
library(ggplot2)
library(dplyr)
library(class)
library(gridExtra)
library(gmodels)
library(caret)
library(ggExtra)
library(ggpubr)
```

Load Data

Load data table with values per text file.

```
# load estimations from stringBase corpus
estimations.df <- read.csv("~/Github/NaLaFi/results/features.csv")
#head(features.csv)
```

Exclude subcorpora (if needed).

```
#selected <- c("random","shuffled")
#estimations.df <- estimations.df[!(estimations.df$subcorpus %in% selected), ]
```

Split into separate files by length of chunks in characters.

```
# choose number of characters
num.char = 10
# subset data frame
estimations.df <- estimations.df[estimations.df$num.char == num.char, ]
nrow(estimations.df)
```

```
## [1] 3361
```

Select relevant columns of the data frame, i.e. the measures to be included in classification and the ''corpus" or ''subcorpus" column.

```
estimations.subset <- estimations.df[c("corpus", "subcorpus",
                                       "huni.chars",
```

```
                                        "hrate.chars",
                                        "ttr.chars",
                                        "rm.chars"
                                        )]
```

Remove NAs (whole row)

```
estimations.subset <- na.omit(estimations.subset)
```

Center and scale the data

```
estimations.scaled <- cbind(estimations.subset[1:2], scale(estimations.subset[3:ncol(estimations.subset)
nrow(estimations.scaled)
```

```
## [1] 3007
```

Create Training and Test Sets

```
# Generating seed
set.seed(1234)
# Randomly generating our training and test samples with a respective ratio of 2/3 and 1/3
datasample <- sample(2, nrow(estimations.scaled), replace = TRUE, prob = c(0.67, 0.33))
# Generate training set
estimations.training <- estimations.scaled[datasample == 1, 3:ncol(estimations.scaled)]
nrow(estimations.training)
```

```
## [1] 2044
```

```
# Generate test set
estimations.test <- estimations.scaled[datasample == 2, 3:ncol(estimations.scaled)]
nrow(estimations.test)
```

```
## [1] 963
```

Get training and test labels

```
# Generate training labels
training.labels <- estimations.scaled[datasample == 1, 1]
# Generate test labels
test.labels <- estimations.scaled[datasample == 2, 1]
```

Initialize data frame

```
knn.results <- data.frame(k = numeric(0), precision = numeric(0),
                          recall = numeric(0), f1 = numeric(0))
```

Building knn classifier

```
# choose maximum number of neighbors n
n = 10
# run a loop over different numbers of neighbors up to n
for (k in 1:n){
  # knn estimation of labels
  predictions.knn <- knn(train = as.data.frame(estimations.training),
                         test = as.data.frame(estimations.test),
                         cl = training.labels, k = k)

  # model evaluation
  # creating a dataframe from known (true) test labels
  test.labels <- data.frame(test.labels)
  # combining predicted and known classes
  class.comparison <- data.frame(predictions.knn, test.labels)
  # giving appropriate column names
  names(class.comparison) <- c("predicted", "observed")
  # inspecting our results table
  head(class.comparison)
  # get confusion matrix
  cm <- confusionMatrix(data = class.comparison$predicted,
                        reference = as.factor(class.comparison$observed))
  # print(cm)
  # get precision, recall, and f1 from the output list of confusionMatrix()
  f1 <- cm[["byClass"]]["F1"]
  recall <- cm[["byClass"]]["Recall"]
  precision <- cm[["byClass"]]["Precision"]

  # prepare data frame with results
  local.results <- data.frame(k, precision, recall, f1, row.names = NULL)
  local.results.rounded <- round(local.results, 2)
  # print(local.results.rounded)
  knn.results <- rbind(knn.results, local.results.rounded)
}
print(knn.results)
```

```
##     k precision recall   f1
## 1   1      0.71   0.72 0.72
## 2   2      0.71   0.70 0.71
## 3   3      0.71   0.69 0.70
## 4   4      0.71   0.68 0.69
## 5   5      0.73   0.69 0.71
## 6   6      0.72   0.70 0.71
## 7   7      0.75   0.69 0.72
## 8   8      0.73   0.69 0.71
## 9   9      0.74   0.68 0.71
## 10 10      0.74   0.68 0.71
```

Write to file.

```
write.csv(knn.results, file = paste("~/Github/NaLaFi/results/knn/knn_results_",
                                    paste(num.char, ".csv", sep =""),
                                    sep =""), row.names = F)
```