Classification with Logistic Regression

Chris Bentz

14/10/2023

Description

Logistic regression analyses of the feature vectors per character string (loaded from NaLaFi/results/features.csv). The results are stored in NaLaFi/results/LR. Note that the number of characters has to be chosen manually (via num.char = " ").

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(dplyr)
library(class)
library(gmodels)
library(caret)
```

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("natural")
#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 = 1000
# subset data frame
estimations.df <- estimations.df[estimations.df$num.char == num.char, ]
```

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",
                                       "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], scale(estimations.subset[2:ncol(estimations.subset)]))
```

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
train <- estimations.scaled[datasample == 1, 1:ncol(estimations.scaled)]
# Generate test set
test <- estimations.scaled[datasample == 2, 1:ncol(estimations.scaled)]
```

Building logistic regression model

The following code to run a logistic regression is adopted from https://datasciencedojo.com/blog/logistic-regression-in-r-tutorial/ (last accessed 16.01.2023).

```
# logistic regression estimation of labels
log.model <- glm(as.factor(corpus) ~., data = train, family = binomial(link = "logit"))
summary(log.model)
```

```
##
## Call:
## glm(formula = as.factor(corpus) ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6470   0.0093   0.2509   0.3375   1.9153
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.5930     0.2143   7.435 1.05e-13 ***
## huni.chars    1.0825     0.5250   2.062   0.0392 *
## hrate.chars  -2.0546     0.3342  -6.147 7.89e-10 ***
## ttr.chars     1.7408     0.7548   2.306   0.0211 *
## rm.chars     -3.8886     0.4215  -9.225  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1386.03  on 1260  degrees of freedom
## Residual deviance:  487.15  on 1256  degrees of freedom
## AIC: 497.15
##
## Number of Fisher Scoring iterations: 7
```

```
# look at the training data and predicted values (y) to check the the dummy coding,
# i.e. "non-writing" is coded as 0, and "writing" as 1
head(train)
```

```
##           corpus huni.chars hrate.chars  ttr.chars   rm.chars
## 7145 non-writing -1.0720886  0.53193108 -0.4813408 -0.1385562
## 7146 non-writing -0.5830817  0.57139926 -0.4813408 -0.4472640
## 7147 non-writing -0.6793608  0.18077197 -0.4813408 -0.5330161
## 7148 non-writing -0.8863766  0.10817215 -0.5019469 -0.5158657
## 7150 non-writing -0.7992368  0.08378230 -0.5225529 -0.4987153
## 7151 non-writing -0.7254698  0.07483311 -0.5431590 -0.5587418
```

```
head(log.model$y)
```

```
## 7145 7146 7147 7148 7150 7151
##    0    0    0    0    0    0
```

Prediction

Make predictions using the logistic regression model with "trained", i.e. estimated coefficients.

```
log.predictions <- predict(log.model, test, type = "response")
head(log.predictions)
```

```
##      7149      7155      7158      7160      7170      7172
## 0.8111904 0.9499524 0.9470035 0.9459057 0.9375979 0.9261141
```

Assign a label according to the rule that the label is "writing" if the prediction probability is >0.5, else assign "non-writing".

```
log.prediction.rd <- ifelse(log.predictions > 0.5, "writing", "non-writing")
head(log.prediction.rd, 10)
```

```
##          7149          7155          7158          7160          7170
##     "writing"     "writing"     "writing"     "writing"     "writing"
##          7172          7173          7180          7183          7184
##     "writing"     "writing" "non-writing" "non-writing" "non-writing"
```

Model evaluation

```
# creating a dataframe from known (true) test labels
test.labels <- data.frame(test$corpus)
# combining predicted and known classes
```

```r
class.comparison <- data.frame(log.prediction.rd, test.labels)
# giving appropriate column names
names(class.comparison) <- c("predicted", "observed")
# inspecting our results table
head(class.comparison)
```

```
##      predicted    observed
## 7149   writing non-writing
## 7155   writing non-writing
## 7158   writing non-writing
## 7160   writing non-writing
## 7170   writing non-writing
## 7172   writing non-writing
```

```r
# get confusion matrix
cm <- confusionMatrix(as.factor(class.comparison$predicted),
                      reference = as.factor(class.comparison$observed))
print(cm)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    non-writing writing
##   non-writing         106       6
##   writing              35     424
##
##                Accuracy : 0.9282
##                  95% CI : (0.9038, 0.948)
##     No Information Rate : 0.7531
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7926
##
##  Mcnemar's Test P-Value : 1.226e-05
##
##             Sensitivity : 0.7518
##             Specificity : 0.9860
##          Pos Pred Value : 0.9464
##          Neg Pred Value : 0.9237
##              Prevalence : 0.2469
##          Detection Rate : 0.1856
##    Detection Prevalence : 0.1961
##       Balanced Accuracy : 0.8689
##
##        'Positive' Class : non-writing
##
```

```r
# get precision, recall, and f1 from the output list of confusionMatrix()
accuracy <- cm$overall['Accuracy']
f1 <- cm[["byClass"]]["F1"]
recall <- cm[["byClass"]]["Recall"]
precision <- cm[["byClass"]]["Precision"]

# prepare data frame with results
lr.results <- data.frame(accuracy, precision, recall, f1, row.names = NULL)
```

```
lr.results.rounded <- round(lr.results, 2)
print(lr.results.rounded)
```

```
##   accuracy precision recall   f1
## 1     0.93      0.95   0.75 0.84
```

Write to file.

```
write.csv(lr.results.rounded, file = paste("~/Github/NaLaFi/results/LR/results_logReg_",
                                           paste(num.char, ".csv", sep = ""),
                                         sep = ""), row.names = F)
```