

# Supervised learning aggregated

Cornelius Erfort

5/10/2021

## Loading packages

```
packages <- c(
  "quanteda", "quanteda.textmodels", "dplyr", "caret", "randomForest", "tm", "beepR", "rmarkdown", "e1071", "haven")

lapply(packages[!(packages %in% rownames(installed.packages()))], install.packages)

if(!("quanteda.classifiers" %in% rownames(installed.packages()))) remotes::install_github("quanteda/quanteda.classifiers")

lapply(c(packages, "quanteda.classifiers"), require, character.only = TRUE)
```

## Loading data

```
sample_germany <- read_dta("../sample_germany.dta")
table(sample_germany$country)

## Warning: Unknown or uninitialised column: `country`.

## < table of extent 0 >

# Correcting classification for three documents
sample_germany$issue[sample_germany$id == 229] <- 191
sample_germany$issue[sample_germany$id == 731] <- 7
sample_germany$issue[sample_germany$id == 902] <- 10

# Subset to relevant vars
germany_textpress <- sample_germany %>% select("header", "text", "issue", "position", "id")

# Distribution of issues in the hand-coded sample
table(germany_textpress$issue)

##
##   1    2    3    4    5    6    7    8    9   10   12   13   14   15   16   17   18   20   23   98
## 175 181 119   99 167 137   84 105 131   74 195 104   32 168 121   68  27  97  19  91
##   99 191 192
##   46 350 152
```

## Merging categories

```
germany_textpress$issue_r1 <- as.numeric(germany_textpress$issue)

germany_textpress <- germany_textpress %>% mutate(issue_r1 = recode(issue_r1,
```

```

      `8` = 7, # Environment & Energy
      `13` = 10, # Transportation & Welfare
      `14` = 10, # Housing & Welfare
      `18` = 15, # Foreign Trade and Domestic Commerce
      `98` = 99, # Non-thematic & Other
      `23` = 99) # Culture: few observations
    )

table(germany_textpress$issue_r1)

```

```

##
##      1      2      3      4      5      6      7      9     10     12     15     16     17     20     99    191    192
## 175 181 119    99 167 137 189 131 210 195 195 121    68    97 156 350 152

```

## Creating the document frequency matrix (dfm)

```
corp_press <- str_c(germany_textpress$header, " ", germany_textpress$text) %>% corpus()
```

```
## Warning: NA is replaced by empty string
```

```
# Add id var to corpus
```

```
docvars(corp_press, "id") <- germany_textpress$id
```

```
docvars(corp_press, "issue_r1") <- germany_textpress$issue_r1
```

```
# Create random sample for test dataset (size: 1/5 of all classified documents)
```

```
set.seed(300)
```

```
id_test <- sample(docvars(corp_press, "id"), round(length(docvars(corp_press, "id"))/5, 0), replace = F)
```

```
# Create training and test set
```

```
dfmat_training <- corpus_subset(corp_press, !(id %in% id_test)) %>%
```

```
  dfm(remove = stopwords("de"), stem = T, remove_punct = T, remove_number = T, remove_symbols = T, remove_digits = T)
```

```
  dfm_trim(min_docfreq = 0.005, max_docfreq = .8, docfreq_type = "prop") # Remove words occurring in less than 0.5% of documents
```

```
dfmat_test <- corpus_subset(corp_press, id %in% id_test) %>%
```

```
  dfm(remove = stopwords("de"), stem = T, remove_punct = T, remove_number = T, remove_symbols = T, remove_digits = T)
```

```
  dfm_trim(min_docfreq = 0.005, max_docfreq = .8, docfreq_type = "prop") # Remove words occurring in less than 0.5% of documents
```

## Naive Bayes classification model

```
tmod_nb_r1 <- textmodel_nb(dfmat_training, dfmat_training$issue_r1)
```

```
# summary(tmod_nb_r1)
```

## Evaluation

```
dfmat_matched <- dfm_match(dfmat_test,
                           features = featnames(dfmat_training))
```

```
actual_class <- docvars(dfmat_matched, "issue_r1")
```

```
predicted_class <- predict(tmod_nb_r1, newdata = dfmat_matched)
```

```
tab_class <- table(actual_class, predicted_class)
```

```
tab_class
```

```
##              predicted_class
```

```
## actual_class  1  2  3  4  5  6  7  9 10 12 15 16 17 20 99 191 192
##          1  23  0  0  0  4  1  4  0  1  1  2  0  1  0  0  0  2
##          2   0 18  2  1  1  1  0  2  0  8  0  1  0  2  1  4  0
##          3   1  1 16  0  0  1  1  0  0  0  0  0  0  0  1  0
##          4   0  0  0 14  0  0  1  0  0  0  1  0  0  0  1  0
##          5   4  0  2  0 23  0  1  1  0  1  1  0  0  0  0  1
##          6   0  0  0  0  2 21  0  0  3  0  0  0  0  0  1  0  0
##          7   0  0  0  4  0  0 28  0  3  1  2  0  1  1  0  1  1
##          9   0  0  0  0  1  0  0 14  0  2  0  0  0  1  1  1  0
##         10   1  1  0  0  3  1  5  1 17  1  7  0  0  1  1  1  0
##         12   1  3  0  1  0  1  1  3  0 19  2  2  1  0  2  2  2
##         15   1  3  1  0  0  1  2  0  3  2 13  0  1  0  1  1  9
##         16   0  2  1  0  0  0  0  0  0  2  1 15  0  2  0  3  0
##         17   0  2  0  0  0  3  0  0  0  1  1  0  3  0  2  0  0
##         20   2  0  0  0  0  2  1  0  0  1  0  1  1  4  0  0  0
##         99   2  0  0  0  0  4  0  0  0  0  0  2  0  4 20  2  0
##        191   0  1  0  1  0  1  3  1  0  1  0  4  0  3  0 55  4
##        192   2  0  0  0  0  0  0  1  0  1  3  0  0  0  2  3 19
```

```
confusionMatrix(tab_class, mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
```

```
##
##          predicted_class
## actual_class  1  2  3  4  5  6  7  9 10 12 15 16 17 20 99 191 192
##          1  23  0  0  0  4  1  4  0  1  1  2  0  1  0  0  0  2
##          2   0 18  2  1  1  1  0  2  0  8  0  1  0  2  1  4  0
##          3   1  1 16  0  0  1  1  0  0  0  0  0  0  0  1  0
##          4   0  0  0 14  0  0  1  0  0  0  1  0  0  0  1  0
##          5   4  0  2  0 23  0  1  1  0  1  1  0  0  0  0  1
##          6   0  0  0  0  2 21  0  0  3  0  0  0  0  0  1  0  0
##          7   0  0  0  4  0  0 28  0  3  1  2  0  1  1  0  1  1
##          9   0  0  0  0  1  0  0 14  0  2  0  0  0  1  1  1  0
##         10   1  1  0  0  3  1  5  1 17  1  7  0  0  1  1  1  0
##         12   1  3  0  1  0  1  1  3  0 19  2  2  1  0  2  2  2
##         15   1  3  1  0  0  1  2  0  3  2 13  0  1  0  1  1  9
##         16   0  2  1  0  0  0  0  0  0  2  1 15  0  2  0  3  0
##         17   0  2  0  0  0  3  0  0  0  1  1  0  3  0  2  0  0
##         20   2  0  0  0  0  2  1  0  0  1  0  1  1  4  0  0  0
##         99   2  0  0  0  0  4  0  0  0  0  0  2  0  4 20  2  0
##        191   0  1  0  1  0  1  3  1  0  1  0  4  0  3  0 55  4
##        192   2  0  0  0  0  0  0  1  0  1  3  0  0  0  2  3 19
```

```
## Overall Statistics
```

```
##
##          Accuracy : 0.5876
##          95% CI : (0.5451, 0.6292)
##    No Information Rate : 0.1369
##    P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##          Kappa : 0.5568
```

```
##
##    McNemar's Test P-Value : NA
```

```
## Statistics by Class:
```

```
##
##          Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Precision      0.58974  0.43902  0.76190  0.82353  0.67647  0.77778
## Recall         0.62162  0.58065  0.72727  0.66667  0.67647  0.56757
## F1             0.60526  0.50000  0.74419  0.73684  0.67647  0.65625
## Prevalence     0.06752  0.05657  0.04015  0.03832  0.06204  0.06752
## Detection Rate 0.04197  0.03285  0.02920  0.02555  0.04197  0.03832
## Detection Prevalence 0.07117  0.07482  0.03832  0.03102  0.06204  0.04927
## Balanced Accuracy 0.79516  0.76808  0.85888  0.83049  0.82753  0.77791
##          Class: 7 Class: 9 Class: 10 Class: 12 Class: 15 Class: 16
## Precision      0.66667  0.70000  0.42500  0.47500  0.34211  0.57692
## Recall         0.59574  0.60870  0.62963  0.46341  0.39394  0.60000
## F1             0.62921  0.65116  0.50746  0.46914  0.36620  0.58824
## Prevalence     0.08577  0.04197  0.04927  0.07482  0.06022  0.04562
## Detection Rate 0.05109  0.02555  0.03102  0.03467  0.02372  0.02737
## Detection Prevalence 0.07664  0.03650  0.07299  0.07299  0.06934  0.04745
## Balanced Accuracy 0.78390  0.79863  0.79274  0.71100  0.67270  0.78948
##          Class: 17 Class: 20 Class: 99 Class: 191 Class: 192
## Precision      0.250000  0.333333  0.58824  0.7432  0.61290
## Recall         0.375000  0.222222  0.64516  0.7333  0.50000
## F1             0.300000  0.266667  0.61538  0.7383  0.55072
## Prevalence     0.014599  0.032847  0.05657  0.1369  0.06934
## Detection Rate 0.005474  0.007299  0.03650  0.1004  0.03467
## Detection Prevalence 0.021898  0.021898  0.06204  0.1350  0.05657
## Balanced Accuracy 0.679167  0.603564  0.80904  0.8466  0.73824
```

```
crossval(tmod_nb_r1, k = 5) # Five-fold cross-validation
```

```
##          precision          recall          f1          accuracy
##          0.6241461          0.6399636          0.6244665          0.6207797
## balanced_accuracy
##          0.6035375
```