ML: NB, RF, SVM

Miras Tolepbergen

2023-11-08

```
rm(list=ls(all=TRUE))
setwd("C:/Users/Miras/Desktop/u_m/1st/big_data_analytics/Labs/projects")
getwd()
```

```
## [1] "C:/Users/Miras/Desktop/u_m/1st/big_data_analytics/Labs/projects"
```

```
library(quanteda)
```

```
## Warning in .recacheSubclasses(def@className, def, env): undefined subclass
## "pcorMatrix" of class "replValueSp"; definition not updated

## Warning in .recacheSubclasses(def@className, def, env): undefined subclass
## "pcorMatrix" of class "xMatrix"; definition not updated

## Warning in .recacheSubclasses(def@className, def, env): undefined subclass
## "pcorMatrix" of class "mMatrix"; definition not updated

## Package version: 3.3.1
## Unicode version: 13.0
## ICU version: 69.1

## Parallel computing: 4 of 4 threads used.

## See https://quanteda.io for tutorials and examples.
```

```
library(readtext)
```

```
##
## Attaching package: 'readtext'

## The following object is masked from 'package:quanteda':
##
##     texts
```

```
library(naivebayes)
```

```
## Warning: package 'naivebayes' was built under R version 4.3.2

## naivebayes 0.9.7 loaded
```

```r
library(ranger)
```

```
## Warning: package 'ranger' was built under R version 4.3.2
```

```r
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.3.2
```

```r
library(reshape2)
library(ggplot2)
```

```r
uk <- read.csv("C:/Users/Miras/Desktop/u_m/1st/big_data_analytics/Labs/projects/uk_train.csv",
               stringsAsFactors=FALSE)
str(uk)
```

```
## 'data.frame':    360 obs. of  6 variables:
##  $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ id         : num  4.71e+17 4.71e+17 4.71e+17 4.71e+17 4.71e+17 ...
##  $ screen_name: chr  "DavidCoburnUKip" "Nospin_43" "WillDuckworthGP" "Andrew_Duff_MEP" ...
##  $ text       : chr  "@benjamincohen @RichardHilton1 On other hands if Faith is reformed such as Qual
##  $ polite     : chr  "polite" "polite" "polite" "polite" ...
##  $ Sentiment  : chr  "neutral" "neutral" "positive" "neutral" ...
```

```r
uk$polite <- ifelse(uk$polite == 'polite', 1, 0)
uk$polite <- factor(uk$polite,  levels=c("0", "1"), labels=c("impolite", "polite"))
str(uk)
```

```
## 'data.frame':    360 obs. of  6 variables:
##  $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ id         : num  4.71e+17 4.71e+17 4.71e+17 4.71e+17 4.71e+17 ...
##  $ screen_name: chr  "DavidCoburnUKip" "Nospin_43" "WillDuckworthGP" "Andrew_Duff_MEP" ...
##  $ text       : chr  "@benjamincohen @RichardHilton1 On other hands if Faith is reformed such as Qual
##  $ polite     : Factor w/ 2 levels "impolite","polite": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Sentiment  : chr  "neutral" "neutral" "positive" "neutral" ...
```

```r
table(uk$polite)
```

```
##
## impolite   polite
##      104      256
```

```r
prop.table(table(uk$polite))
```

```
##
##  impolite    polite
## 0.2888889 0.7111111
```

```r
nrow(uk)
```

```
## [1] 360
```

```r
# DfM for the train-set

myCorpusTwitterTrain <- corpus(uk)
tok2 <- tokens(myCorpusTwitterTrain , remove_punct = TRUE, remove_numbers=TRUE,
               remove_symbols = TRUE, split_hyphens = TRUE, remove_separators = TRUE, remove_URL = TRUE)
```

```
## Warning: remove_URL argument is not used.
```

```r
tok2 <- tokens_remove(tok2, stopwords("en"))
# let's also remove the unicode symbols
tok2 <- tokens_remove(tok2, c("0*", "*@*"))
tok2 <- tokens_wordstem (tok2)
Dfm_train <- dfm(tok2)

# Let's trim the dfm in order to keep only tokens that appear in 2 or more tweets
# and let's keep only features with at least 2 characters
Dfm_train <- dfm_trim(Dfm_train , min_docfreq = 2, verbose=TRUE)
```

```
## Removing features occurring:

##   - in fewer than 2 documents: 1,058

##   Total features removed: 1,058 (71.6%).
```

```r
Dfm_train  <- dfm_remove(Dfm_train , min_nchar = 2)
topfeatures(Dfm_train , 20)  # 20 top words
```

```
##    ukip     get   peopl     amp     can    look    like    time    fffd     one
##      24      22      16      16      15      15      14      14      14      13
##    just    good english     now   think   thank      go   parti   right    make
##      13      12      11      11      11      11      11      11      11      10
```

```r
# DfM for the test-set

uk10 <- read.csv("C:/Users/Miras/Desktop/u_m/1st/big_data_analytics/Labs/projects/uk_test2.csv",
                 stringsAsFactors=FALSE)
str(uk10)
```

```
## 'data.frame':    360 obs. of  6 variables:
##  $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ id         : num  4.71e+17 4.71e+17 4.71e+17 4.71e+17 4.71e+17 ...
##  $ screen_name: chr  "NickiBrooksx" "DavidCoburnUKip" "IainMcGill" "danielrhamilton" ...
##  $ text       : chr  "@LilianGreenwood @FoDS_Group  truly stunning." "@benjamincohen @RichardHilton1
##  $ polite     : logi  NA NA NA NA NA NA ...
##  $ Sentiment  : logi  NA NA NA NA NA NA ...
```

```r
nrow(uk10)
```

```
## [1] 360
```

```r
myCorpusTwitterTest <- corpus(uk10)
tok <- tokens(myCorpusTwitterTest , remove_punct = TRUE, remove_numbers=TRUE,
              remove_symbols = TRUE, split_hyphens = TRUE, remove_separators = TRUE, remove_URL = TRUE)
```

```
## Warning: remove_URL argument is not used.
```

```r
tok <- tokens_remove(tok, stopwords("en"))
tok <- tokens_remove(tok, c("0*", "*@*"))
tok <- tokens_wordstem (tok)
Dfm_test <- dfm(tok)
Dfm_test<- dfm_trim(Dfm_test, min_docfreq = 2, verbose=TRUE)
```

```
## Removing features occurring:

##   - in fewer than 2 documents: 1,010

##   Total features removed: 1,010 (70.7%).
```

```r
Dfm_test<- dfm_remove(Dfm_test, min_nchar = 2)

# with just 0s in its DfM It would be a non-reliable prediction by definition
Dfm_test[ntoken(Dfm_test) == 0,]
```

```
## Document-feature matrix of: 16 documents, 415 features (100.00% sparse) and 5 docvars.
##          features
## docs      mind ukip first repres scotland vote euro kind word dear
##   text1      0    0     0      0        0    0    0    0    0    0
##   text64     0    0     0      0        0    0    0    0    0    0
##   text82     0    0     0      0        0    0    0    0    0    0
##   text118    0    0     0      0        0    0    0    0    0    0
##   text147    0    0     0      0        0    0    0    0    0    0
##   text153    0    0     0      0        0    0    0    0    0    0
## [ reached max_ndoc ... 10 more documents, reached max_nfeat ... 405 more features ]
```

```r
Dfm_test <- Dfm_test[ntoken(Dfm_test) != 0,]
Dfm_test[ntoken(Dfm_test) == 0,]
```

```
## Document-feature matrix of: 0 documents, 415 features (0.00% sparse) and 5 docvars.
## [ reached max_nfeat ... 405 more features ]
```

```r
topfeatures(Dfm_test , 20)  # 20 top words
```

```
##    amp   ukip   vote  thank   make   need  peopl  parti  think  #ukip   fuck
##     26     22     20     20     19     17     17     15     15     15     15
##    can   like   just    one    say racist   know    get     go
##     14     14     14     13     12     12     11     11     11
```

```r
#make the features identical between train and test-set
#by passing Dfm_train to dfm_match() as a pattern

setequal(featnames(Dfm_train), featnames(Dfm_test))
```

```
## [1] FALSE
```

```r
nfeat(Dfm_test)
```

```
## [1] 415
```

```r
nfeat(Dfm_train)
```

```
## [1] 414
```

```r
test_dfm  <- dfm_match(Dfm_test, features = featnames(Dfm_train))
nfeat(test_dfm)
```

```
## [1] 414
```

```r
setequal(featnames(Dfm_train), featnames(test_dfm ))
```

```
## [1] TRUE
```

```r
#convert the two DfMs into matrices for the ML algorithms to work

train <- as(Dfm_train, "dgCMatrix")
test <- as(test_dfm, "dgCMatrix")
```

#Naive Bayes Model

```r
table(Dfm_train@x)
```

```
##
##    1    2    4    6
## 1397   37    2    1
```

```r
str(Dfm_train@docvars$polite)
```

```
##  Factor w/ 2 levels "impolite","polite": 2 2 2 2 2 2 2 2 2 2 ...
```

```r
system.time(NB <- multinomial_naive_bayes(x=train, y=Dfm_train@docvars$polite))
```

```
##    user  system elapsed
##    0.10    0.12    0.24
```

```r
NB
```

```
##
## ============================ Multinomial Naive Bayes ============================
##
##  Call:
## multinomial_naive_bayes(x = train, y = Dfm_train@docvars$polite)
##
## ---------------------------------------------------------------------------------
##
## Laplace smoothing: 0.5
##
## ---------------------------------------------------------------------------------
##
##  A priori probabilities:
##  impolite    polite
## 0.2888889 0.7111111
##
## ---------------------------------------------------------------------------------
##
##           Classes
## Features     impolite      polite
##   faith    0.00078125 0.002779984
##   reform   0.00078125 0.002779984
##   oppos    0.00078125 0.001985703
##   gay      0.00234375 0.005957109
##   marriag  0.00078125 0.006751390
##   english  0.00859375 0.005162828
##   scotland 0.00390625 0.001985703
##   get      0.01015625 0.013105639
##   one      0.00390625 0.009134234
##   agre     0.00078125 0.003574265
##
## ---------------------------------------------------------------------------------
##
## # ... and 404 more features
##
## ---------------------------------------------------------------------------------
```

```r
prop.table(table(Dfm_train@docvars$polite)) # prior probabilities
```

```
##
##  impolite    polite
## 0.2888889 0.7111111
```

```r
head(NB$params, 10) #likelihood of a tweet containing a word 'faith' to be polite
```

```
##            impolite      polite
## faith    0.00078125 0.002779984
## reform   0.00078125 0.002779984
## oppos    0.00078125 0.001985703
## gay      0.00234375 0.005957109
```

```
## marriag  0.00078125 0.006751390
## english  0.00859375 0.005162828
## scotland 0.00390625 0.001985703
## get      0.01015625 0.013105639
## one      0.00390625 0.009134234
## agre     0.00078125 0.003574265
```

```
                    #is much higher than impolite, similar can be said about words like
                    #'reform' and 'marriag', while likelihood of a tweet containing words
                    #'english' and 'scotland' for instance to be impolite is slightly higher
```

```r
head(sort((NB$params[,2]-NB$params[,1]) , decreasing=TRUE), 10)
```

```
##        good        time       peopl       thank     marriag        know
## 0.009147265 0.008379046 0.007610827 0.005996202 0.005970140 0.005970140
##         one         new        word        need
## 0.005227984 0.005175859 0.005175859 0.004381578
```

```r
# the features that present the highest absolute value in the difference between
#the two likelihoods can be considered as among the most important ones in affecting
#the overall performance of the algorithm in predicting the "polite" label in the training-set
head(sort((NB$params[,1]-NB$params[,2]) , decreasing=TRUE), 10) #same with "impolite" feature
```

```
##        fuck        fffd      stupid         use      racist        twat
## 0.016009109 0.012831985 0.008196609 0.008196609 0.008170547 0.006634109
##     everyon        must       speak       enjoy
## 0.006608047 0.006581985 0.005071609 0.005071609
```

```r
# let's predict the test-set

predicted_nb <- predict(NB ,test )
table(predicted_nb )
```

```
## predicted_nb
## impolite   polite
##       65      279
```

```r
prop.table(table(predicted_nb ))
```

```
## predicted_nb
##  impolite    polite
## 0.1889535 0.8110465
```

```r
head(predict(NB ,test))
```

```
## [1] polite polite polite polite polite polite
## Levels: impolite polite
```

```r
head(predict(NB ,test, type="prob" ))
```

```
##         impolite    polite
## text2 0.28888889 0.7111111
## text3 0.28651052 0.7134895
## text4 0.05058303 0.9494170
## text5 0.30642590 0.6935741
## text6 0.01354288 0.9864571
## text7 0.06080461 0.9391954
```

#Random Forrest

```r
set.seed(123)
system.time(RF <- ranger(y= Dfm_train@docvars$polite, x=train, keep.inbag=TRUE))
```

```
##    user  system elapsed
##    1.69    0.02    0.54
```

```r
RF
```

```
## Ranger result
##
## Call:
##  ranger(y = Dfm_train@docvars$polite, x = train, keep.inbag = TRUE)
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      360
## Number of independent variables:  414
## Mtry:                             20
## Target node size:                 1
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error:             25.83 %
```

```r
# see how  observations/texts are in-bag in each tree. Let's see the first (of 500) tree:
RF$inbag.counts[1]
```

```
## [[1]]
##   [1] 1 1 0 2 0 0 3 0 0 2 3 0 5 0 2 0 0 0 2 0 0 1 2 0 0 1 2 1 0 1 1 1 1 1 0 1 1
##  [38] 1 0 0 0 2 3 1 1 2 0 1 0 0 1 1 2 2 1 2 1 0 1 2 1 0 1 0 0 0 2 2 1 1 2 2 3 0
##  [75] 3 0 1 4 0 1 1 0 1 0 1 0 1 2 0 1 0 2 1 2 0 1 2 1 2 0 2 1 0 1 3 1 0 0 1 0 1
## [112] 1 1 0 2 1 2 0 1 1 0 1 1 1 2 2 2 1 2 2 0 0 0 1 1 1 1 1 3 2 0 1 1 1 1 3 1 1
## [149] 3 0 0 1 1 0 3 0 0 2 2 1 2 1 2 1 1 0 1 2 0 0 0 0 0 1 0 0 0 0 2 1 0 0 0 2 0
## [186] 1 1 1 2 1 0 3 1 4 0 1 1 0 1 1 1 2 0 0 2 0 2 2 3 1 0 0 1 2 2 0 0 2 2 2 1 0
## [223] 1 0 3 2 3 0 1 0 0 2 1 2 0 0 2 1 2 0 0 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 2 2 3
## [260] 1 1 1 1 0 2 2 0 1 0 2 1 1 0 0 1 1 1 0 1 1 1 0 1 0 2 0 0 0 1 0 1 2 2 1 1 0
## [297] 0 1 3 0 1 2 2 1 3 2 2 1 0 0 1 0 1 0 0 2 1 2 2 0 2 0 1 1 0 0 1 2 2 0 1 1 3
## [334] 1 1 2 1 0 0 0 2 2 1 2 0 3 1 0 0 1 0 1 1 1 0 2 0 2 2 1
```

```r
sum(unlist(RF$inbag.counts[1]))
```

```
## [1] 360
```

```r
RF
```

```
## Ranger result
##
## Call:
##  ranger(y = Dfm_train@docvars$polite, x = train, keep.inbag = TRUE)
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      360
## Number of independent variables:  414
## Mtry:                             20
## Target node size:                 1
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error:             25.83 %
```

```r
RF$prediction.error
```

```
## [1] 0.2583333
```

```r
1-RF$prediction.error
```

```
## [1] 0.7416667
```

```r
nt <- seq(1, 501, 10)
# We can also plot how the OOB predictions errors change over the number of
#trees we employ. For example, between 1 and 501 trees
nt
```

```
##  [1]   1  11  21  31  41  51  61  71  81  91 101 111 121 131 141 151 161 171 181
## [20] 191 201 211 221 231 241 251 261 271 281 291 301 311 321 331 341 351 361 371
## [39] 381 391 401 411 421 431 441 451 461 471 481 491 501
```

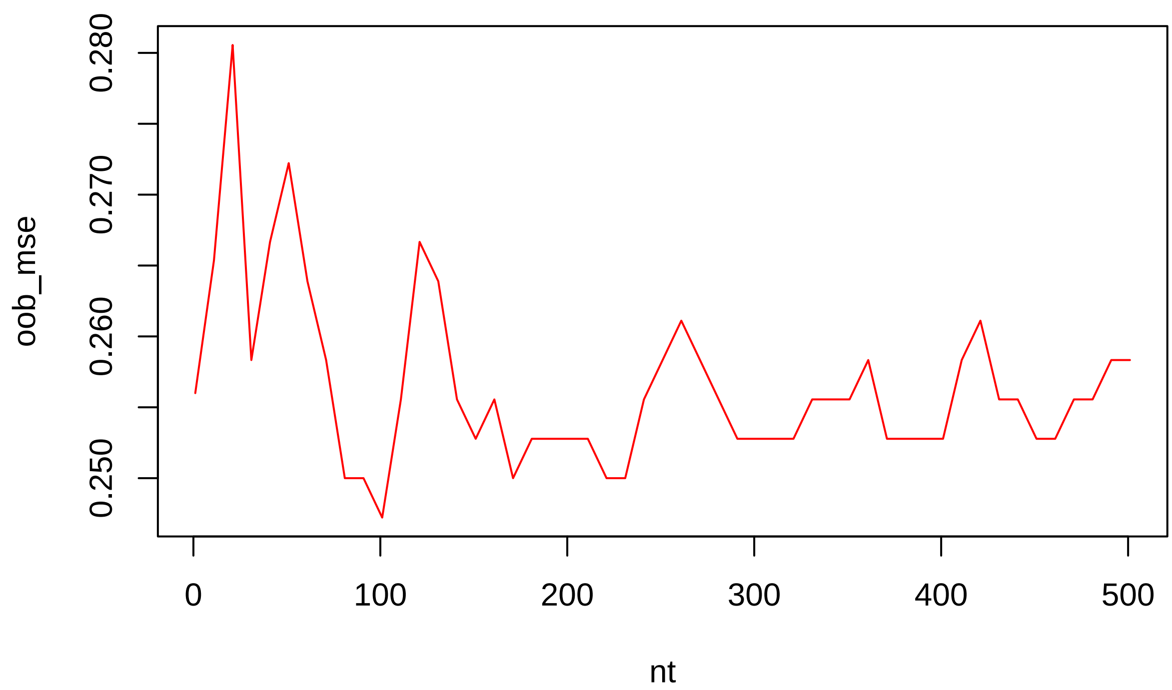
```r
oob_mse <- vector("numeric", length(nt))

for(i in 1:length(nt)){
  set.seed(123)
  rr2 <- ranger(y= Dfm_train@docvars$polite, x=train,  num.threads=4, num.trees = nt[i], write.forest =
  oob_mse[i] <- rr2$prediction.error
}

oob_mse
```

```
##  [1] 0.2560000 0.2653631 0.2805556 0.2583333 0.2666667 0.2722222 0.2638889
##  [8] 0.2583333 0.2500000 0.2500000 0.2472222 0.2555556 0.2666667 0.2638889
## [15] 0.2555556 0.2527778 0.2555556 0.2500000 0.2527778 0.2527778 0.2527778
## [22] 0.2527778 0.2500000 0.2500000 0.2555556 0.2583333 0.2611111 0.2583333
## [29] 0.2555556 0.2527778 0.2527778 0.2527778 0.2527778 0.2555556 0.2555556
## [36] 0.2555556 0.2583333 0.2527778 0.2527778 0.2527778 0.2527778 0.2583333
## [43] 0.2611111 0.2555556 0.2555556 0.2527778 0.2527778 0.2555556 0.2555556
## [50] 0.2583333 0.2583333
```

```r
plot(x = nt, y = oob_mse, col = "red", type = "l")
```




```r
#the lowest prediction error for OOB at about 100 trees
```

```r
set.seed(123)
system.time(RFI <- ranger(y= Dfm_train@docvars$polite, x=train, importance="permutation",
                    scale.permutation.importance = TRUE))
```

```
##    user  system elapsed
##   13.24    0.11    4.00
```

```r
head(RFI $variable.importance)
```

```
##      faith     reform      oppos        gay    marriag    english
##  1.0010015  0.0000000  0.0000000  0.7023113  2.1873484 -2.4972925
```

```r
# 10 most important words
head(sort(RFI$variable.importance , decreasing=TRUE), 10)
```

```
##      fuck    stupid      twat    bloodi      noth       ars     idiot    sicken
## 35.311670 18.766541 16.717028 14.066076 12.199862 11.429539 11.147829 10.244112
##       lol     enjoy
##  9.499127  9.280221
```

```r
#these variables increase the avg. error rate in our prediction relative to
#when we use the actual variable value of that feature
```

```r
#predict test-set

set.seed(123)
system.time(predicted_rf <- predict(RF, test))
```

```
##    user  system elapsed
##    0.33    0.00    0.11
```

```r
str(predicted_rf )
```

```
## List of 5
##  $ predictions              : Factor w/ 2 levels "impolite","polite": 2 2 2 2 1 2 2 2 2 2 ...
##  $ num.trees                : num 500
##  $ num.independent.variables: num 414
##  $ num.samples              : int 2
##  $ treetype                 : chr "Classification"
##  - attr(*, "class")= chr "ranger.prediction"
```

```r
table(predicted_rf$predictions )
```

```
##
## impolite   polite
##       35      309
```

```r
prop.table(table(predicted_rf$predictions ))
```

```
##
##  impolite    polite
## 0.1017442 0.8982558
```

```r
set.seed(1)
system.time(predicted_rf2 <- predict(RF, test))
```

```
##    user  system elapsed
##    0.33    0.00    0.10
```

```r
table(predicted_rf2$predictions )
```

```
##
## impolite   polite
##       35      309
```

```r
set.seed(123)
system.time(RF2 <- ranger(y= Dfm_train@docvars$polite, x=train,  probability=TRUE))
```

```
##    user  system elapsed
##    1.85    0.03    0.55
```

```r
set.seed(123)
system.time(predicted_rf2 <- predict(RF2, test))
```

```
##    user  system elapsed
##    0.38    0.02    0.15
```

```r
str(predicted_rf )
```

```
## List of 5
##  $ predictions              : Factor w/ 2 levels "impolite","polite": 2 2 2 2 1 2 2 2 2 2 ...
##  $ num.trees                : num 500
##  $ num.independent.variables: num 414
##  $ num.samples              : int 2
##  $ treetype                 : chr "Classification"
##  - attr(*, "class")= chr "ranger.prediction"
```

```r
str(predicted_rf2 )
```

```
## List of 5
##  $ predictions              : num [1:344, 1:2] 0.157 0.163 0.113 0.168 0.522 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:2] "impolite" "polite"
##  $ num.trees                : num 500
##  $ num.independent.variables: num 414
##  $ num.samples              : int 2
##  $ treetype                 : chr "Probability estimation"
##  - attr(*, "class")= chr "ranger.prediction"
```

```r
head(predicted_rf2$predictions )
```

```
##        impolite    polite
## [1,] 0.15700870 0.8429913
## [2,] 0.16254956 0.8374504
## [3,] 0.11289453 0.8871055
## [4,] 0.16768562 0.8323144
## [5,] 0.52188015 0.4781198
## [6,] 0.08414686 0.9158531
```

```r
set.seed(123)
system.time(predicted_rfALL <- predict(RF, test, predict.all=TRUE))
```

```
##    user  system elapsed
##    0.27    0.00    0.10
```

```r
str(predicted_rfALL )
```

```
## List of 5
##  $ predictions              : num [1:344, 1:500] 2 2 2 2 2 2 2 2 2 2 ...
##  $ num.trees                : num 500
##  $ num.independent.variables: num 414
##  $ num.samples              : int 2
##  $ treetype                 : chr "Classification"
##  - attr(*, "class")= chr "ranger.prediction"
```

```r
# let's see the prediction of the 500 trees for the first text in the test-set
predicted_rfALL$predictions[1,]
```

```
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [223] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [260] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [297] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [334] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [371] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [408] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [445] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [482] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```r
# this text is classified as 2 i.e., polite
table(predicted_rfALL$predictions[1,])
```

```
## 
##   2 
## 500
```

```r
# let's see the prediction of the 500 trees for the second text in the test-set
predicted_rfALL$predictions[2,]
```

```
##   [1] 2 2 2 2 2 2 1 2 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2
##  [38] 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 1 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 1 2 2 2
## [112] 2 2 2 2 2 2 1 2 2 2 1 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [149] 2 2 2 2 2 2 2 2 2 2 1 2 1 2 1 1 2 2 2 1 2 2 2 2 2 2 2 1 1 1 2 2 1 2 2 2 2
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2
```

```
## [223] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 1
## [260] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [297] 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2
## [334] 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2
## [371] 2 2 2 1 2 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2
## [408] 2 2 2 1 2 1 2 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 1 2 2 2 2 1 2 2 2 2
## [445] 1 1 2 2 1 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1 1 2
## [482] 2 2 2 2 2 2 1 1 2 2 2 2 2 1 2 1 2 2 2
```

```r
# this text is classified between 1-2 i.e. impolite and polite respectively)
table(predicted_rfALL$predictions[2,]) #more polite than impolite though
```

```
##
##   1   2
##  67 433
```

```r
# and indeed:
head(predicted_rf$predictions )
```

```
## [1] polite   polite   polite   polite   impolite polite
## Levels: impolite polite
```

#Support Vector Machines SVM

```r
system.time(SV <- svm(y= Dfm_train@docvars$polite, x=train, kernel='linear'))
```

```
##    user  system elapsed
##    0.09    0.00    0.13
```

```r
SV
```

```
##
## Call:
## svm.default(x = train, y = Dfm_train@docvars$polite, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  244
```

```r
length(SV$index)
```

```
## [1] 244
```

```r
nrow(train) # 244 out of 360 texts in the training-set data
```

```
## [1] 360
```

```r
head(SV$coefs)
```

```
##             [,1]
## [1,] 0.22808807
## [2,] 0.01319214
## [3,] 0.11640339
## [4,] 1.00000000
## [5,] 1.00000000
## [6,] 0.98212158
```

```r
summary(SV$coefs)
```

```
##        V1
##  Min.   :-1.00000
##  1st Qu.:-0.47086
##  Median : 0.07487
##  Mean   : 0.00000
##  3rd Qu.: 0.37383
##  Max.   : 1.00000
```

```r
str(SV) #the decision values in classifying the documents in the training-set
```

```
## List of 29
##  $ call           : language svm.default(x = train, y = Dfm_train@docvars$polite, kernel = "linear")
##  $ type           : num 0
##  $ kernel         : num 0
##  $ cost           : num 1
##  $ degree         : num 3
##  $ gamma          : num 0.00242
##  $ coef0          : num 0
##  $ nu             : num 0.5
##  $ epsilon        : num 0.1
##  $ sparse         : logi TRUE
##  $ scaled         : logi [1:414] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ x.scale        : NULL
##  $ y.scale        : NULL
##  $ nclasses       : int 2
##  $ levels         : chr [1:2] "impolite" "polite"
##  $ tot.nSV        : int 244
##  $ nSV            : int [1:2] 154 90
##  $ labels         : int [1:2] 2 1
##  $ SV             :Formal class 'matrix.csr' [package "SparseM"] with 4 slots
##   .. ..@ ra       : num [1:874] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..@ ja       : int [1:874] 6 7 8 9 10 11 8 9 19 20 ...
##   .. ..@ ia       : int [1:245] 1 7 14 15 15 17 19 20 21 22 ...
##   .. ..@ dimension: int [1:2] 244 414
##  $ index          : int [1:244] 2 4 5 6 7 8 9 11 16 17 ...
##  $ rho            : num -0.883
##  $ compprob       : logi FALSE
##  $ probA          : NULL
##  $ probB          : NULL
##  $ sigma          : NULL
```

```
##  $ coefs          : num [1:244, 1] 0.2281 0.0132 0.1164 1 1 ...
##  $ na.action      : NULL
##  $ fitted         : Factor w/ 2 levels "impolite","polite": 2 2 2 2 2 2 2 2 2 2 ...
##   ..- attr(*, "names")= chr [1:360] "1" "2" "3" "4" ...
##  $ decision.values: num [1:360, 1] 1.38 1 1.34 1 1 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:360] "1" "2" "3" "4" ...
##   .. ..$ : chr "polite/impolite"
##  - attr(*, "class")= chr "svm"
```

```r
head(SV$decision.values)
```

```
##   polite/impolite
## 1       1.3821758
## 2       0.9999428
## 3       1.3378022
## 4       1.0002673
## 5       0.9998761
## 6       0.8834727
```

```r
#positive coeff. means text is classified as 'polite', thus all first  are 'polite'
head(predict(SV , train))
```

```
##      1      2      3      4      5      6
## polite polite polite polite polite polite
## Levels: impolite polite
```

```r
# let's illustrate texts that represent the support vectors in our case
str(uk)
```

```
## 'data.frame':    360 obs. of  6 variables:
##  $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ id         : num  4.71e+17 4.71e+17 4.71e+17 4.71e+17 4.71e+17 ...
##  $ screen_name: chr  "DavidCoburnUKip" "Nospin_43" "WillDuckworthGP" "Andrew_Duff_MEP" ...
##  $ text       : chr  "@benjamincohen @RichardHilton1 On other hands if Faith is reformed such as Qua
##  $ polite     : Factor w/ 2 levels "impolite","polite": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Sentiment  : chr  "neutral" "neutral" "positive" "neutral" ...
```

```r
vectors <- uk[SV$index,]
nrow(vectors)
```

```
## [1] 244
```

```r
str(vectors) # texts 1, 3, 12-15 for example are absent cause they are not supporting vectors!
```

```
## 'data.frame':    244 obs. of  6 variables:
##  $ X          : int  2 4 5 6 7 8 9 11 16 17 ...
##  $ id         : num  4.71e+17 4.71e+17 4.71e+17 4.71e+17 4.71e+17 ...
##  $ screen_name: chr  "Nospin_43" "Andrew_Duff_MEP" "beachthistle" "GawainTowler" ...
##  $ text       : chr  "@DavidCoburnUKip @Vote_UKIP do english servicemen/wm stationed in Scotland get
##  $ polite     : Factor w/ 2 levels "impolite","polite": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Sentiment  : chr  "neutral" "neutral" "neutral" "negative" ...
```

```r
vectors$coefs <- SV$coefs[,1]
str(vectors)
```

```
## 'data.frame':    244 obs. of  7 variables:
##  $ X          : int  2 4 5 6 7 8 9 11 16 17 ...
##  $ id         : num  4.71e+17 4.71e+17 4.71e+17 4.71e+17 4.71e+17 ...
##  $ screen_name: chr  "Nospin_43" "Andrew_Duff_MEP" "beachthistle" "GawainTowler" ...
##  $ text       : chr  "@DavidCoburnUKip @Vote_UKIP do english servicemen/wm stationed in Scotland get
##  $ polite     : Factor w/ 2 levels "impolite","polite": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Sentiment  : chr  "neutral" "neutral" "neutral" "negative" ...
##  $ coefs      : num  0.2281 0.0132 0.1164 1 1 ...
```

```r
summary(vectors$coefs)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.00000 -0.47086  0.07487  0.00000  0.37383  1.00000
```

```r
vectors<- vectors[order(vectors$coef),] # negative coefficient implies 'impolite' text
str(vectors)
```

```
## 'data.frame':    244 obs. of  7 variables:
##  $ X          : int  257 259 260 264 267 275 286 291 298 304 ...
##  $ id         : num  4.63e+17 4.63e+17 4.63e+17 4.64e+17 4.64e+17 ...
##  $ screen_name: chr  "GinnerRodgers" "kymru" "ThatAndyHall" "MrHappySW11" ...
##  $ text       : chr  "A modern jazzy reversion of Mien Kampf @nickgriffinmep #twat http://t.co/wRRKor
##  $ polite     : Factor w/ 2 levels "impolite","polite": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Sentiment  : chr  "negative" "negative" "negative" "negative" ...
##  $ coefs      : num  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
```

```r
strwrap((vectors$text)[1:7])
```

```
##  [1] "A modern jazzy reversion of Mien Kampf @nickgriffinmep #twat"
##  [2] "http://t.co/wRRKomnUB6 via @YouTube"
##  [3] "@Nigel_Farage you forgot the firing squad in front of that billboard"
##  [4] "@Nigel_Farage a plague apon both your houses"
##  [5] "@steveparrott50 @Nigel_Farage typical lefty beeb #hahnotpolitical"
##  [6] "@Nigel_Farage atwat when awake , and a twat when u kip."
##  [7] "http://t.co/0goa58Jee7"
##  [8] "@GoodallGiles are you secretly related to Joss and is your middle name"
##  [9] "'Rosetta'?"
## [10] "@Bruciebabe @Nigel_Farage Perhaps we should change EU in UK to FU"
```

```r
vectors <- vectors[order(-vectors$coef),] # positive coefficient implies 'polite' text
str(vectors)
```

```
## 'data.frame':    244 obs. of  7 variables:
##  $ X          : int  6 7 9 22 31 36 47 54 81 83 ...
##  $ id         : num  4.71e+17 4.71e+17 4.71e+17 4.71e+17 4.71e+17 ...
##  $ screen_name: chr  "GawainTowler" "edwardhayes_1" "GoodallGiles" "TheMockneyRebel" ...
##  $ text       : chr  "@craigmelson not funny..." "@MEPStandingUp4U the people believe in you" "@pswi
##  $ polite     : Factor w/ 2 levels "impolite","polite": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Sentiment  : chr  "negative" "positive" "negative" "neutral" ...
##  $ coefs      : num  1 1 1 1 1 1 1 1 1 1 ...
```

```r
strwrap((vectors$text)[1:7])
```

```
## [1] "@craigmelson not funny..."
## [2] "@MEPStandingUp4U the people believe in you"
## [3] "@pswidlicki @GrillingKippers Couldn't be worse now could it?"
## [4] "@Tim_Aker @GuidoFawkes I attended and I can assure you I was not"
## [5] "handpicked @ElzbietaVine"
## [6] "@JimBobbers cheesy peas!"
## [7] "@imonckton @WhiteGeNOcideHC @Dubdanu @nickgriffinmep @StanCollymore"
## [8] "Racist like the occupation of Palestine by Israel you mean?"
## [9] "@davenellist cheeky 280odd here! #everylittlehelps"
```

```r
# let's predict the test-set
system.time(predicted_svm <- predict(SV , test))
```

```
##    user  system elapsed
##    0.02    0.00    0.01
```

```r
table(predicted_svm )
```

```
## predicted_svm
## impolite   polite
##       68      276
```

```r
prop.table(table(predicted_svm ))
```

```
## predicted_svm
##  impolite    polite
## 0.1976744 0.8023256
```

```r
system.time(SVprob <- svm(y= Dfm_train@docvars$polite, x=train, kernel='linear', probability=TRUE)) #wi
```

```
##    user  system elapsed
##    0.06    0.00    0.08
```

```r
head(predict(SVprob , test))
```

```
##      1      2      3      4      5      6
## polite polite polite polite polite polite
## Levels: impolite polite
```

```r
head(attr(predict(SVprob , test,probability=TRUE),"probabilities"))
```

```
##      polite  impolite
## 1 0.7743053 0.2256947
## 2 0.7014893 0.2985107
## 3 0.8198404 0.1801596
## 4 0.7262152 0.2737848
## 5 0.7537279 0.2462721
## 6 0.7560294 0.2439706
```

#NB, RF, SVM results comparison

```r
prop.table(table(predicted_nb )) #NB
```

```
## predicted_nb
##  impolite    polite
## 0.1889535 0.8110465
```

```r
prop.table(table(predicted_rf$predictions )) #RF
```

```
##
##  impolite    polite
## 0.1017442 0.8982558
```

```r
prop.table(table(predicted_svm )) #SVM
```

```
## predicted_svm
##  impolite    polite
## 0.1976744 0.8023256
```

```r
results <- as.data.frame(rbind(prop.table(table(predicted_nb )), prop.table(table(predicted_rf$predicti
str(results)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ impolite: num  0.189 0.102 0.198
##  $ polite  : num  0.811 0.898 0.802
```

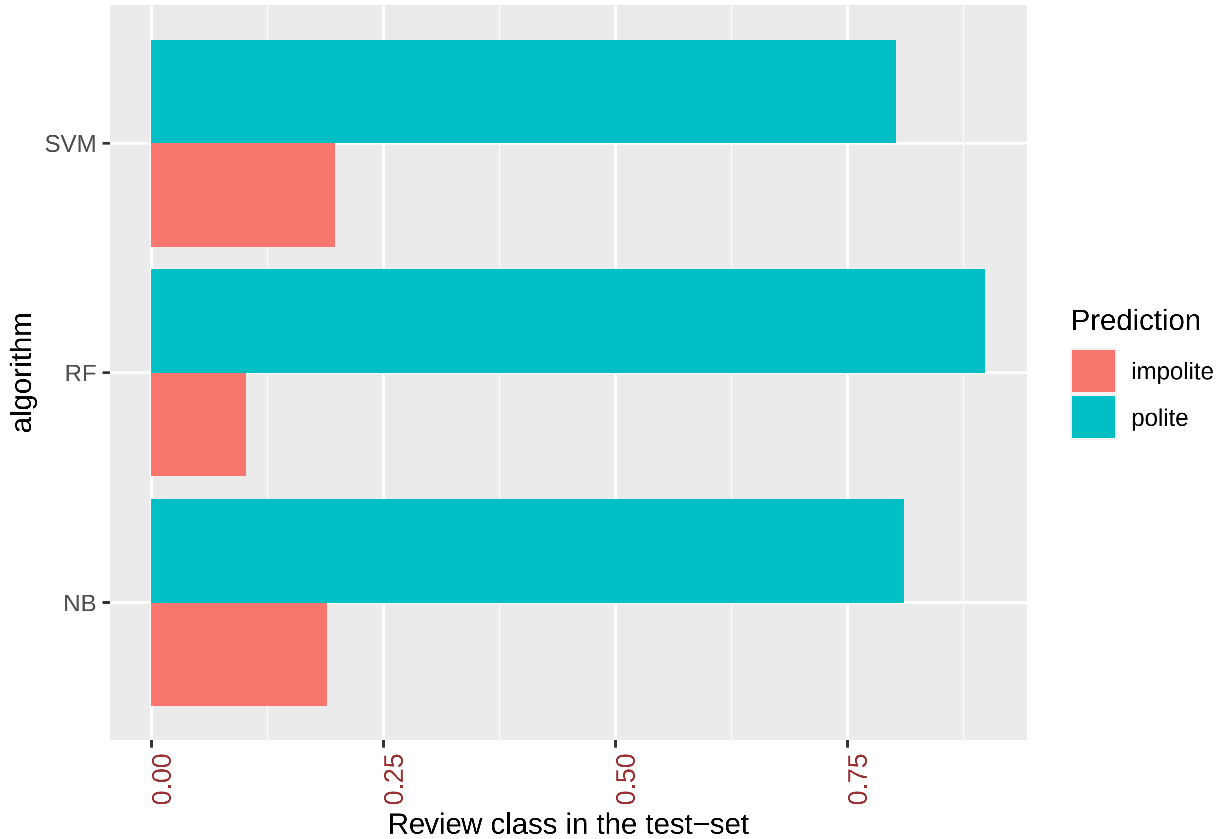
```r
results$algorithm <- c("NB", "RF", "SVM")
str(results)
```

```
## 'data.frame':    3 obs. of  3 variables:
##  $ impolite : num  0.189 0.102 0.198
##  $ polite   : num  0.811 0.898 0.802
##  $ algorithm: chr  "NB" "RF" "SVM"
```

```r
# plot the results
df.long<-melt(results,id.vars=c("algorithm"))
str(df.long)
```

```
## 'data.frame':    6 obs. of  3 variables:
##  $ algorithm: chr  "NB" "RF" "SVM" "NB" ...
##  $ variable : Factor w/ 2 levels "impolite","polite": 1 1 1 2 2 2
##  $ value    : num  0.189 0.102 0.198 0.811 0.898 ...
```

```r
ggplot(df.long,aes(algorithm,value,fill=variable))+
  geom_bar(position="dodge",stat="identity") + theme(axis.text.x = element_text(color="#993333", size=1
  ylab(label="Review class in the test-set") +  xlab("algorithm") + scale_fill_discrete(name = "Predict
```



##RF results are the quite different ones regarding the both classess, while SVM and NB results are very similar