Election Violence Newspaper Article Selection Process Pilot: Evaluate Keyword Selection Method

August 10, 2018

1 Summary

After the code to setup and download data (Sections 2-??) the document below includes: First, an implementation of code to select of articles for coding, where initial evaluation of articles is based on keywords from the boolean searches which return those articles (Sections ??-??). This classification is done using actual searches conducted (effectively 4 keywords) with a Naive Bayes classifier.

Second, the document then provides an evaluation of the keyword selection step based on user classified articles from the platform with Naive Bayes and other classifiers and with 4 and 10 keyword sets (Sections 4-7).

Finally, to provide an alternative way of evaluating the process, the document includes 10 randomly selected cases which are rejected for coding in the implementation code (Section ??).

2 Packages used

```r
library(reportRx) # note using this just for sanitizing LaTex output, not included in durhamevp package
library(tidyverse)
library(quanteda)
library(durhamevp)
library(RTextTools) # not sure if this is included as part of the durhamevp package
```

3 Download Classified Cases

There is an initial set of cases which represent the concept (in this case 'election violence'). We also have a set of cases which do not represent the concept (Note: King et al do not have the non-election violence cases).

I also add indicators that this material is classified (`classified==1`) and that it is not unclassified (`unclass==0`) which will be useful when added to subsequent material.

```r
# In fact we have some classified docs non-EV cases here King's R set doesn't contain any non-cases
# I think we should use our more extensive information here

classdocs<-durhamevp::get_classified_docs()

classdocs<-classdocs %>%
  dplyr::mutate(std_url = sub("download/", "", url)) %>%
  dplyr::mutate(unclass=0, classified=1)
```

4 Assess Naive Bayes selection performance on 4 keywords (binary dfm)

Here there are four keywords (election, riot, disturbance, incident) which are either present (1) or not-present (0). Use these patterns to identify election violence articles with a Naive Bayes classifer.

```
class_corpus<-quanteda::corpus(classdocs[classdocs$election_article==1,], text_field="ocr")
keywords<-c("election", "riot", "incident", "disturbance")
class_dfm_4<-quanteda::dfm(class_corpus, select=keywords)
#class_dfm<-preprocess_corpus(class_corpus, stem=FALSE, min_termfreq = 20)

the_sets_4<-split_dfm(class_dfm_4, n_train = 1000)
classifier<-quanteda::textmodel_nb(the_sets_4$training_set, y=quanteda::docvars(the_sets_4$training_set
quanteda::docvars(the_sets_4$testing_set, "predicted")<-predict(classifier, newdata = the_sets_4$testin
caret::confusionMatrix(data=quanteda::docvars(the_sets_4$testing_set, "predicted"), reference=factor(qu
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  10   8
##          1 134 395
##
##                Accuracy : 0.7404
##                  95% CI : (0.7015, 0.7767)
##     No Information Rate : 0.7367
##     P-Value [Acc > NIR] : 0.4451
##
##                   Kappa : 0.069
##  Mcnemar's Test P-Value : <2e-16
##
##               Precision : 0.7467
##                  Recall : 0.9801
##                      F1 : 0.8476
##              Prevalence : 0.7367
##          Detection Rate : 0.7221
##    Detection Prevalence : 0.9671
##       Balanced Accuracy : 0.5248
##
##        'Positive' Class : 1
##
```

5 How do other classifiers perform on 4 keywords (binary dfm)

Is performance improved by using other classifiers than Naive Bayes? Here there are four keywords (election, riot, disturbance, incident) which are either present (1) or not-present (0). Use these patterns to identify election violence articles.

```
class_corpus<-quanteda::corpus(classdocs[classdocs$election_article==1,], text_field="ocr")
keywords<-c("election", "riot", "incident", "disturbance")
class_dfm<-quanteda::dfm(class_corpus, select=keywords)
class_dfm_bin<-quanteda::dfm_weight(class_dfm, scheme="boolean")
doc_matrix<-quanteda::convert(class_dfm_bin, "tm")
```

```
## Warning in dfm2tm(x):  converted DocumentTermMatrix will not have weight attributes set
correctly
```

```
training_nos<-which(1:nrow(class_dfm_bin) %in% sample(1:nrow(class_dfm_bin), 1000))
testing_nos<-which(!1:nrow(class_dfm_bin) %in% training_nos)
container <- RTextTools::create_container(doc_matrix, quanteda::docvars(class_dfm, "EV_article"), train

models_res<-RTextTools::train_models(container, algorithms=c("MAXENT", "SVM", "BOOSTING", "BAGGING", "R
results_res<-RTextTools::classify_models(container, models_res)
analytics_res<-RTextTools::create_analytics(container, results_res)

class_summary<-analytics_res@document_summary %>%
  tibble::rowid_to_column("document_id") %>%
  tidyr::gather(variable, value, -MANUAL_CODE, -document_id) %>%
  separate(variable, c("variable", "var2")) %>%
  filter(var2 %in% c("CODE")) %>%
  unite(value, var2, value) %>%
  group_by(variable, MANUAL_CODE, value) %>%
  tally() %>%
  spread(value, n)
```

```
## Warning:  attributes are not identical across measure variables;
## they will be dropped
## Warning:  package 'bindrcpp' was built under R version 3.4.4
```

```
class_summary_by_classifier<-analytics_res@document_summary %>%
  tibble::rowid_to_column("document_id") %>%
  tidyr::gather(variable, value, -MANUAL_CODE, -document_id) %>%
  separate(variable, c("variable", "var2")) %>%
  filter(var2=="LABEL") %>%
  unite(value, var2, value) %>%
  group_by(variable, MANUAL_CODE, value) %>%
  tally() %>%
  spread(value, n)
```

```
## Warning:  attributes are not identical across measure variables;
## they will be dropped
```

```
knitr::knit_print(knitr::kable(class_summary))
```

variable	MANUAL_CODE	CODE_0	CODE_1
CONSENSUS	0	7	177
CONSENSUS	1	6	357
PROBABILITY	0	7	177
PROBABILITY	1	6	357

```
knitr::knit_print(knitr::kable(class_summary_by_classifier))
```

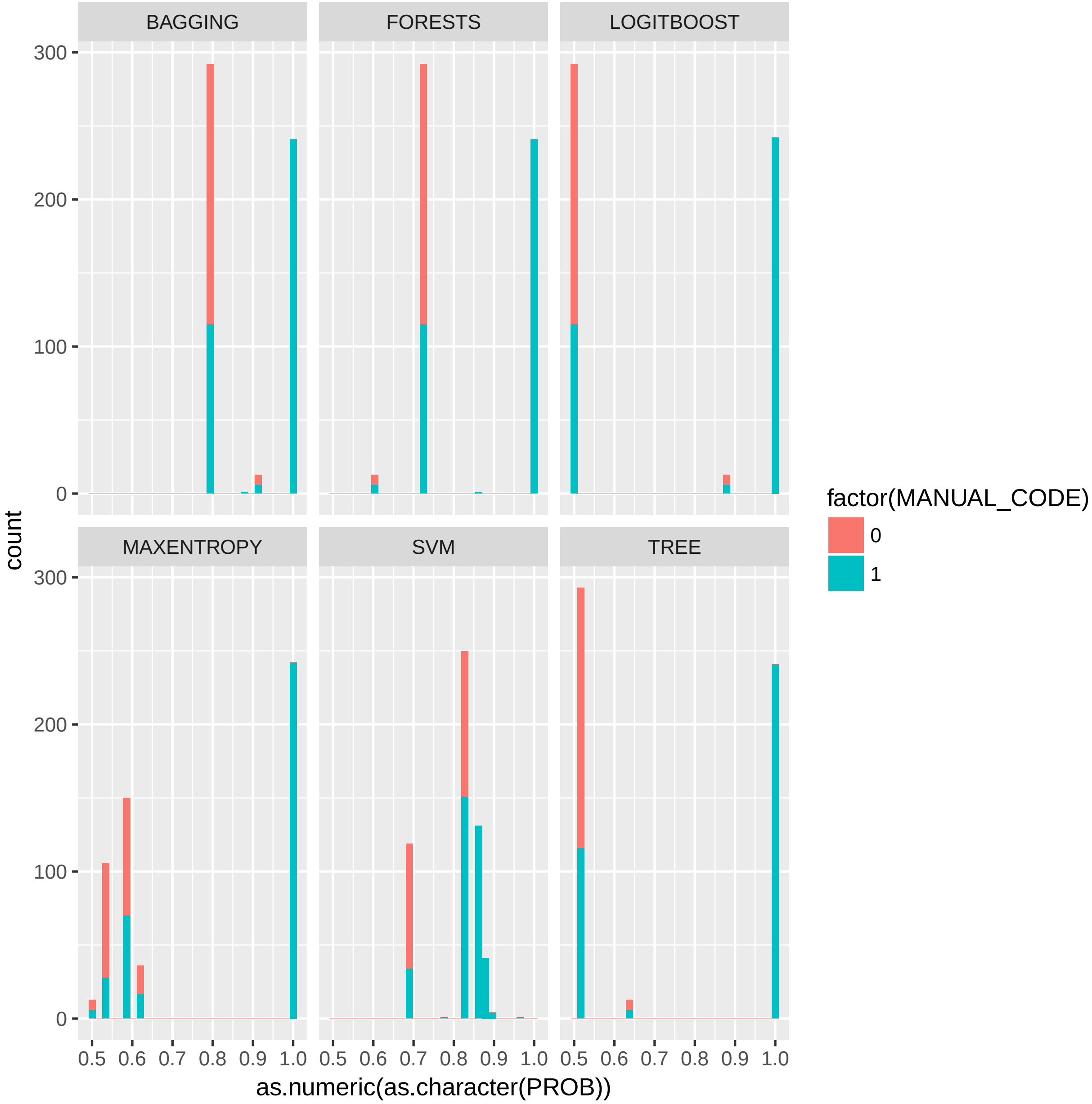
variable	MANUAL_CODE	LABEL_0	LABEL_1
BAGGING	0	7	177
BAGGING	1	6	357
FORESTS	0	NA	184
FORESTS	1	NA	363
LOGITBOOST	0	184	NA
LOGITBOOST	1	121	242
MAXENTROPY	0	7	177
MAXENTROPY	1	6	357
SVM	0	85	99
SVM	1	34	329
TREE	0	7	177
TREE	1	6	357

```
prob_summary<-analytics_res@document_summary %>%
  tibble::rowid_to_column("document_id") %>%
  tidyr::gather(variable, value, -MANUAL_CODE, -document_id) %>%
  separate(variable, c("variable", "var2")) %>%
  filter(var2 %in% c("LABEL", "PROB")) %>%
  spread(var2, value) %>%
  group_by(variable, MANUAL_CODE, LABEL)

## Warning:  attributes are not identical across measure variables;
## they will be dropped

prob_summary%>%
  ggplot(aes(x=as.numeric(as.character(PROB)), fill=factor(MANUAL_CODE)))+
  facet_wrap(~variable)+
  geom_histogram()

## ‘stat_bin()‘ using ‘bins = 30‘.  Pick better
## value with ‘binwidth‘.
```



```
t(analytics_res@algorithm_summary)

##                           0    1
## SVM_PRECISION          0.71 0.77
## SVM_RECALL             0.46 0.91
## SVM_FSCORE             0.56 0.83
## LOGITBOOST_PRECISION 0.60 1.00
## LOGITBOOST_RECALL     1.00 0.67
## LOGITBOOST_FSCORE     0.75 0.80
## BAGGING_PRECISION     0.54 0.67
## BAGGING_RECALL        0.04 0.98
## BAGGING_FSCORE        0.07 0.80
## FORESTS_PRECISION      NaN 0.66
## FORESTS_RECALL        0.00 1.00
## FORESTS_FSCORE        NaN 0.80
## TREE_PRECISION       0.54 0.67
## TREE_RECALL          0.04 0.98
## TREE_FSCORE          0.07 0.80
## MAXENTROPY_PRECISION 0.54 0.67
## MAXENTROPY_RECALL    0.04 0.98
## MAXENTROPY_FSCORE    0.07 0.80
```

```
t(analytics_res@label_summary)
```

```
##                                          0
## NUM_MANUALLY_CODED              184.000000
## NUM_CONSENSUS_CODED              13.000000
## NUM_PROBABILITY_CODED            13.000000
## PCT_CONSENSUS_CODED               7.065217
## PCT_PROBABILITY_CODED             7.065217
## PCT_CORRECTLY_CODED_CONSENSUS     3.804348
## PCT_CORRECTLY_CODED_PROBABILITY   3.804348
##                                         1
## NUM_MANUALLY_CODED              363.00000
## NUM_CONSENSUS_CODED             534.00000
## NUM_PROBABILITY_CODED           534.00000
## PCT_CONSENSUS_CODED             147.10744
## PCT_PROBABILITY_CODED           147.10744
## PCT_CORRECTLY_CODED_CONSENSUS    98.34711
## PCT_CORRECTLY_CODED_PROBABILITY  98.34711
```

```
analytics_res@ensemble_summary
```

```
##        n-ENSEMBLE COVERAGE n-ENSEMBLE RECALL
## n >= 1                1.00              0.67
## n >= 2                1.00              0.67
## n >= 3                1.00              0.67
## n >= 4                1.00              0.67
## n >= 5                0.81              0.76
## n >= 6                0.44              1.00
```

```
summary(analytics_res)
```

```
## ENSEMBLE SUMMARY
##
##        n-ENSEMBLE COVERAGE n-ENSEMBLE RECALL
## n >= 1                1.00              0.67
## n >= 2                1.00              0.67
## n >= 3                1.00              0.67
## n >= 4                1.00              0.67
## n >= 5                0.81              0.76
## n >= 6                0.44              1.00
##
##
## ALGORITHM PERFORMANCE
##
##        SVM_PRECISION           SVM_RECALL
##                0.740                0.685
##           SVM_FSCORE LOGITBOOST_PRECISION
##                0.695                0.800
##    LOGITBOOST_RECALL    LOGITBOOST_FSCORE
##                0.835                0.775
##    BAGGING_PRECISION       BAGGING_RECALL
##                0.605                0.510
##       BAGGING_FSCORE    FORESTS_PRECISION
##                0.435                0.330
##       FORESTS_RECALL       FORESTS_FSCORE
##                0.500                0.400
##       TREE_PRECISION          TREE_RECALL
##                0.605                0.510
##          TREE_FSCORE MAXENTROPY_PRECISION
##                0.435                0.605
##    MAXENTROPY_RECALL    MAXENTROPY_FSCORE
##                0.510                0.435
```

6 Assess Naive Bayes selection performance on 10 keywords (binary dfm)

Here there are ten keywords ((election, riot, disturbance, incident, mob, stone, window, candidate, party, hustings, magistrate) which are either present (1) or not-present (0). Use these patterns to identify election violence articles with a Naive Bayes classifer.

```
cd<-classdocs[classdocs$election_article==1,]
cd<-cd[sample(nrow(cd)),]
#class_corpus<-quanteda::corpus(classdocs[classdocs£election_article==1,], text_field="ocr")
class_corpus<-quanteda::corpus(cd, text_field="ocr")
keywords<-c("election", "riot", "incident", "disturbance", "mob", "stone", "window", "candidate", "party
class_dfm_10<-quanteda::dfm(class_corpus, select=keywords)
#class_dfm<-preprocess_corpus(class_corpus, stem=FALSE, min_termfreq = 20)

the_sets_10<-split_dfm(class_dfm_10, n_train = 1000)
classifier_10<-quanteda::textmodel_nb(the_sets_10$training_set, y=quanteda::docvars(the_sets_10$training
quanteda::docvars(the_sets_10$testing_set, "predicted")<-predict(classifier_10, newdata = the_sets_10$te
caret::confusionMatrix(data=quanteda::docvars(the_sets_10$testing_set, "predicted"), reference=factor(q

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  94  53
##          1  75 325
##
##                Accuracy : 0.766
##                  95% CI : (0.7282, 0.8009)
##     No Information Rate : 0.691
##     P-Value [Acc > NIR] : 6.201e-05
##
##                   Kappa : 0.4315
##  Mcnemar's Test P-Value : 0.06343
##
##               Precision : 0.8125
##                  Recall : 0.8598
##                      F1 : 0.8355
##              Prevalence : 0.6910
##          Detection Rate : 0.5941
##    Detection Prevalence : 0.7313
##       Balanced Accuracy : 0.7080
##
##        'Positive' Class : 1
##
```

7 How do other classifiers perform on 10 keywords (binary dfm)

The same but with ten keywords (election, riot, disturbance, incident, mob, stone, window, candidate, party, hustings, magistrate) which are either present (1) or not-present (0). Use these patterns to identify election violence articles using a range of other classifiers.

```
class_corpus<-quanteda::corpus(classdocs[classdocs$election_article==1,], text_field="ocr")
cd<-classdocs[classdocs$election_article==1,]
cd<-cd[sample(nrow(cd)),]
#class_corpus<-quanteda::corpus(classdocs[classdocs£election_article==1,], text_field="ocr")
class_corpus<-quanteda::corpus(cd, text_field="ocr")
keywords<-c("election", "riot", "incident", "disturbance", "mob", "stone", "window", "candidate", "party
class_dfm<-quanteda::dfm(class_corpus, select=keywords)
class_dfm_bin<-quanteda::dfm_weight(class_dfm, scheme="boolean")
doc_matrix<-quanteda::convert(class_dfm_bin, "tm")

## Warning in dfm2tm(x):  converted DocumentTermMatrix will not have weight attributes set
correctly

training_nos<-which(1:nrow(class_dfm_bin) %in% sample(1:nrow(class_dfm_bin), 1000))
testing_nos<-which(!1:nrow(class_dfm_bin) %in% training_nos)
container <- RTextTools::create_container(doc_matrix, quanteda::docvars(class_dfm_bin, "EV_article"), t

models_res<-RTextTools::train_models(container, algorithms=c("MAXENT", "SVM", "BOOSTING", "BAGGING", "R
results_res<-RTextTools::classify_models(container, models_res)
analytics_res<-RTextTools::create_analytics(container, results_res)

class_summary<-analytics_res@document_summary %>%
  tibble::rowid_to_column("document_id") %>%
  tidyr::gather(variable, value, -MANUAL_CODE, -document_id) %>%
  separate(variable, c("variable", "var2")) %>%
  filter(var2 %in% c("CODE")) %>%
  unite(value, var2, value) %>%
  group_by(variable, MANUAL_CODE, value) %>%
  tally() %>%
  spread(value, n)

## Warning:  attributes are not identical across measure variables;
## they will be dropped

class_summary_by_classifier<-analytics_res@document_summary %>%
  tibble::rowid_to_column("document_id") %>%
  tidyr::gather(variable, value, -MANUAL_CODE, -document_id) %>%
  separate(variable, c("variable", "var2")) %>%
  filter(var2=="LABEL") %>%
  unite(value, var2, value) %>%
  group_by(variable, MANUAL_CODE, value) %>%
  tally() %>%
  spread(value, n)
```

```
## Warning:  attributes are not identical across measure variables;
## they will be dropped
```

```
knitr::knit_print(knitr::kable(class_summary))
```

variable	MANUAL_CODE	CODE_0	CODE_1
CONSENSUS	0	NA	161
CONSENSUS	1	NA	386
PROBABILITY	0	1	160
PROBABILITY	1	NA	386

```
knitr::knit_print(knitr::kable(class_summary_by_classifier))
```

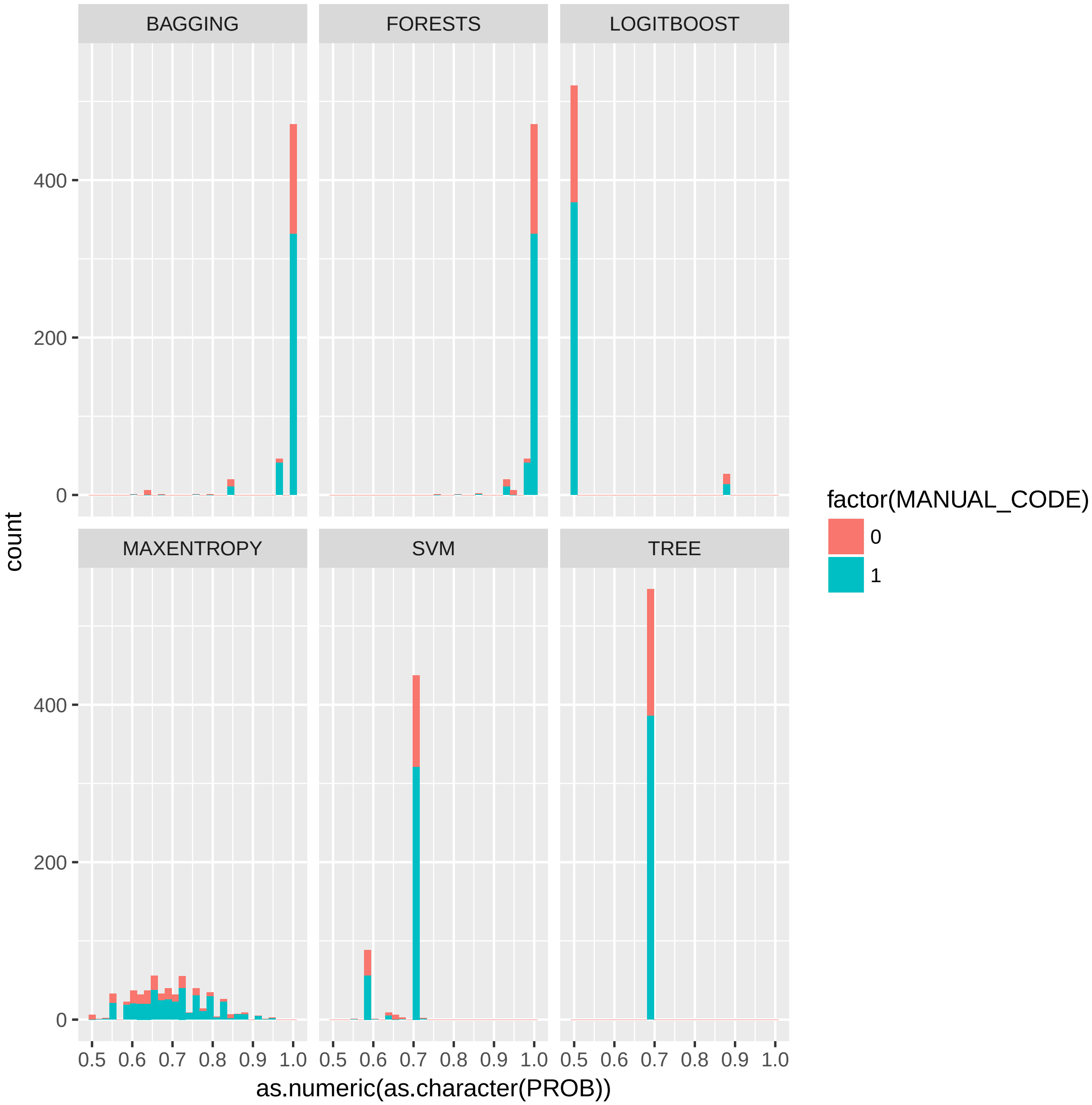
variable	MANUAL_CODE	LABEL_0	LABEL_1
BAGGING	0	8	153
BAGGING	1	1	385
FORESTS	0	NA	161
FORESTS	1	NA	386
LOGITBOOST	0	148	13
LOGITBOOST	1	372	14
MAXENTROPY	0	NA	161
MAXENTROPY	1	NA	386
SVM	0	NA	161
SVM	1	NA	386
TREE	0	NA	161
TREE	1	NA	386

```
prob_summary<-analytics_res@document_summary %>%
  tibble::rowid_to_column("document_id") %>%
  tidyr::gather(variable, value, -MANUAL_CODE, -document_id) %>%
  separate(variable, c("variable", "var2")) %>%
  filter(var2 %in% c("LABEL", "PROB")) %>%
  spread(var2, value) %>%
  group_by(variable, MANUAL_CODE, LABEL)
```

```
## Warning:  attributes are not identical across measure variables;
## they will be dropped
```

```
prob_summary%>%
  ggplot(aes(x=as.numeric(as.character(PROB)), fill=factor(MANUAL_CODE)))+
  facet_wrap(~variable)+
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`.  Pick better
## value with `binwidth`.
```



```
t(analytics_res@algorithm_summary)

##                           0    1
## SVM_PRECISION           NaN 0.71
## SVM_RECALL             0.00 1.00
## SVM_FSCORE              NaN 0.83
## LOGITBOOST_PRECISION   0.28 0.52
## LOGITBOOST_RECALL      0.92 0.04
## LOGITBOOST_FSCORE      0.43 0.07
## BAGGING_PRECISION      0.89 0.72
## BAGGING_RECALL         0.05 1.00
## BAGGING_FSCORE         0.09 0.84
## FORESTS_PRECISION       NaN 0.71
## FORESTS_RECALL         0.00 1.00
## FORESTS_FSCORE        NaN 0.83
## TREE_PRECISION        NaN 0.71
## TREE_RECALL          0.00 1.00
## TREE_FSCORE           NaN 0.83
## MAXENTROPY_PRECISION  NaN 0.71
## MAXENTROPY_RECALL    0.00 1.00
## MAXENTROPY_FSCORE     NaN 0.83
```

```
t(analytics_res@label_summary)
```

```
##                                            0
## NUM_MANUALLY_CODED              161.000000
## NUM_CONSENSUS_CODED               0.000000
## NUM_PROBABILITY_CODED             1.000000
## PCT_CONSENSUS_CODED               0.000000
## PCT_PROBABILITY_CODED             0.621118
## PCT_CORRECTLY_CODED_CONSENSUS     0.000000
## PCT_CORRECTLY_CODED_PROBABILITY   0.621118
##                                          1
## NUM_MANUALLY_CODED              386.0000
## NUM_CONSENSUS_CODED             547.0000
## NUM_PROBABILITY_CODED           546.0000
## PCT_CONSENSUS_CODED             141.7098
## PCT_PROBABILITY_CODED           141.4508
## PCT_CORRECTLY_CODED_CONSENSUS   100.0000
## PCT_CORRECTLY_CODED_PROBABILITY 100.0000
```

```
analytics_res@ensemble_summary
```

```
##        n-ENSEMBLE COVERAGE n-ENSEMBLE RECALL
## n >= 1                1.00              0.71
## n >= 2                1.00              0.71
## n >= 3                1.00              0.71
## n >= 4                1.00              0.71
## n >= 5                0.99              0.71
## n >= 6                0.05              0.54
```

```
summary(analytics_res)
```

```
## ENSEMBLE SUMMARY
##
##        n-ENSEMBLE COVERAGE n-ENSEMBLE RECALL
## n >= 1                1.00              0.71
## n >= 2                1.00              0.71
## n >= 3                1.00              0.71
## n >= 4                1.00              0.71
## n >= 5                0.99              0.71
## n >= 6                0.05              0.54
##
##
## ALGORITHM PERFORMANCE
##
##        SVM_PRECISION           SVM_RECALL
##                0.355                0.500
##           SVM_FSCORE LOGITBOOST_PRECISION
##               0.415              0.400 
##    LOGITBOOST_RECALL    LOGITBOOST_FSCORE 
##               0.480              0.250 
##    BAGGING_PRECISION       BAGGING_RECALL 
##               0.805              0.525 
##       BAGGING_FSCORE    FORESTS_PRECISION 
##               0.465              0.355 
##       FORESTS_RECALL       FORESTS_FSCORE 
##               0.500              0.415 
##       TREE_PRECISION          TREE_RECALL 
##               0.355              0.500 
##          TREE_FSCORE MAXENTROPY_PRECISION 
##               0.415              0.355 
##    MAXENTROPY_RECALL    MAXENTROPY_FSCORE 
##               0.500              0.415
```

8 Assess Naive Bayes selection performance on full set of keywords (binary dfm)

Here we take all the keywords in the ocr as a theoretical maximum which are either present (1) or not-present (0). Use these patterns to identify election violence articles with a Naive Bayes classifer.

```
cd<-classdocs[classdocs$election_article==1,]
cd<-classdocs
cd<-cd[sample(nrow(cd)),]
#class_corpus<-quanteda::corpus(classdocs[classdocs£election_article==1,], text_field="ocr")
class_corpus<-quanteda::corpus(cd, text_field="ocr")
class_dfm_full<-preprocess_corpus(class_corpus, stem=FALSE, min_termfreq = 20, min_docfreq = 20)

the_sets_full<-split_dfm(class_dfm_full, n_train = 1000)
classifier_full<-quanteda::textmodel_nb(the_sets_full$training_set, y=quanteda::docvars(the_sets_full$tr
quanteda::docvars(the_sets_full$testing_set, "predicted")<-predict(classifier_full, newdata = the_sets_
caret::confusionMatrix(data=quanteda::docvars(the_sets_full$testing_set, "predicted"), reference=factor
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   0   1
##          0 333 134
##          1 328 455
##                                          
##                Accuracy : 0.6304         
##                  95% CI : (0.603, 0.6572)
##     No Information Rate : 0.5288         
##     P-Value [Acc > NIR] : 2.589e-13      
##                                          
##                   Kappa : 0.2714         
##  Mcnemar's Test P-Value : < 2.2e-16      
##                                          
##               Precision : 0.5811         
##                  Recall : 0.7725         
##                      F1 : 0.6633
##               Prevalence : 0.4712
##           Detection Rate : 0.3640
##     Detection Prevalence : 0.6264
##        Balanced Accuracy : 0.6381
##
##         'Positive' Class : 1
##
```