Classifier(s) (legacy)

```r
set.seed(42)

library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(party)
```

```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x stringr::boundary() masks strucchange::boundary()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x purrr::lift()       masks caret::lift()
## x dplyr::where()      masks party::where()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.2.0 --
## v broom        1.0.5     v rsample      1.2.1
## v dials        1.3.0     v tune         1.2.1
## v infer        1.0.7     v workflows    1.1.4
## v modeldata    1.4.0     v workflowsets 1.1.0
```

```
## v parsnip      1.2.1     v yardstick    1.3.2
## v recipes      1.1.0
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard()        masks purrr::discard()
## x dplyr::filter()          masks stats::filter()
## x parsnip::fit()           masks infer::fit(), party::fit(), modeltools::fit()
## x recipes::fixed()         masks stringr::fixed()
## x dplyr::lag()             masks stats::lag()
## x purrr::lift()            masks caret::lift()
## x tune::parameters()       masks dials::parameters(), modeltools::parameters()
## x yardstick::precision()   masks caret::precision()
## x yardstick::recall()      masks caret::recall()
## x yardstick::sensitivity() masks caret::sensitivity()
## x yardstick::spec()        masks readr::spec()
## x yardstick::specificity() masks caret::specificity()
## x recipes::step()          masks stats::step()
## x recipes::update()        masks stats4::update(), stats::update()
## x dplyr::where()           masks party::where()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

Load and tidy data

```r
pretty_names <- read_csv("../feat_name_mapping.csv")
```

```
## Rows: 85 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): name_orig, name_pretty
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
data <- read_csv("../measurements/measurements.csv")
```

```
## Rows: 754 Columns: 108
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (20): fpath, KUK_ID, FileName, FileFormat, FolderPath, subcorpus, Source...
## dbl (85): RuleAbstractNouns, RuleAmbiguousRegards, RuleAnaphoricReferences, ...
## lgl  (3): ClarityPursuit, SyllogismBased, Bindingness
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
data_no_nas <- data %>%
  select(!c(
    fpath,
    # KUK_ID,
    # FileName,
    FolderPath,
    # subcorpus,
    DocumentTitle,
    ClarityPursuit,
    Readability,
```

```
    SyllogismBased,
    SourceDB
  )) %>%
  # replace -1s in variation coefficients with NAs
  mutate(across(c(
    `RuleDoubleAdpos.max_allowable_distance.v`,
    `RuleTooManyNegations.max_negation_frac.v`,
    `RuleTooManyNegations.max_allowable_negations.v`,
    `RuleTooManyNominalConstructions.max_noun_frac.v`,
    `RuleTooManyNominalConstructions.max_allowable_nouns.v`,
    `RuleCaseRepetition.max_repetition_count.v`,
    `RuleCaseRepetition.max_repetition_frac.v`,
    `RulePredSubjDistance.max_distance.v`,
    `RulePredObjDistance.max_distance.v`,
    `RuleInfVerbDistance.max_distance.v`,
    `RuleMultiPartVerbs.max_distance.v`,
    `RuleLongSentences.max_length.v`,
    `RulePredAtClauseBeginning.max_order.v`,
    `mattr.v`,
    `maentropy.v`
  ), ~ na_if(.x, -1))) %>%
  # replace NAs with 0s
  replace_na(list(
    RuleGPcoordovs = 0,
    RuleGPdeverbaddr = 0,
    RuleGPpatinstr = 0,
    RuleGPdeverbsubj = 0,
    RuleGPadjective = 0,
    RuleGPpatbenperson = 0,
    RuleGPwordorder = 0,
    RuleDoubleAdpos = 0,
    RuleDoubleAdpos.max_allowable_distance = 0,
    RuleDoubleAdpos.max_allowable_distance.v = 0,
    RuleAmbiguousRegards = 0,
    RuleReflexivePassWithAnimSubj = 0,
    RuleTooManyNegations = 0,
    RuleTooManyNegations.max_negation_frac = 0,
    RuleTooManyNegations.max_negation_frac.v = 0,
    RuleTooManyNegations.max_allowable_negations = 0,
    RuleTooManyNegations.max_allowable_negations.v = 0,
    RuleTooManyNominalConstructions.max_noun_frac.v = 0,
    RuleTooManyNominalConstructions.max_allowable_nouns.v = 0,
    RuleFunctionWordRepetition = 0,
    RuleCaseRepetition.max_repetition_count.v = 0,
    RuleCaseRepetition.max_repetition_frac.v = 0,
    RuleWeakMeaningWords = 0,
    RuleAbstractNouns = 0,
    RuleRelativisticExpressions = 0,
    RuleConfirmationExpressions = 0,
    RuleRedundantExpressions = 0,
    RuleTooLongExpressions = 0,
    RuleAnaphoricReferences = 0,
    RuleLiteraryStyle = 0,
```

```
    RulePassive = 0,
    RulePredSubjDistance = 0,
    RulePredSubjDistance.max_distance = 0,
    RulePredSubjDistance.max_distance.v = 0,
    RulePredObjDistance = 0,
    RulePredObjDistance.max_distance = 0,
    RulePredObjDistance.max_distance.v = 0,
    RuleInfVerbDistance = 0,
    RuleInfVerbDistance.max_distance = 0,
    RuleInfVerbDistance.max_distance.v = 0,
    RuleMultiPartVerbs = 0,
    RuleMultiPartVerbs.max_distance = 0,
    RuleMultiPartVerbs.max_distance.v = 0,
    RuleLongSentences.max_length.v = 0,
    RulePredAtClauseBeginning.max_order.v = 0,
    RuleVerbalNouns = 0,
    RuleDoubleComparison = 0,
    RuleWrongValencyCase = 0,
    RuleWrongVerbonominalCase = 0,
    RuleIncompleteConjunction = 0
  ))

data_clean <- data_no_nas %>%
  # norm data expected to correlate with text length
  mutate(across(c(
    RuleGPcoordovs,
    RuleGPdeverbaddr,
    RuleGPpatinstr,
    RuleGPdeverbsubj,
    RuleGPadjective,
    RuleGPpatbenperson,
    RuleGPwordorder,
    RuleDoubleAdpos,
    RuleAmbiguousRegards,
    RuleFunctionWordRepetition,
    RuleWeakMeaningWords,
    RuleAbstractNouns,
    RuleRelativisticExpressions,
    RuleConfirmationExpressions,
    RuleRedundantExpressions,
    RuleTooLongExpressions,
    RuleAnaphoricReferences,
    RuleLiteraryStyle,
    RulePassive,
    RuleVerbalNouns,
    RuleDoubleComparison,
    RuleWrongValencyCase,
    RuleWrongVerbonominalCase,
    RuleIncompleteConjunction,
    num_hapax,
    RuleReflexivePassWithAnimSubj,
    RuleTooManyNominalConstructions,
    RulePredSubjDistance,
```

```
    RuleMultiPartVerbs,
    RulePredAtClauseBeginning
  ), ~ .x / word_count)) %>%
  mutate(across(c(
    RuleTooFewVerbs,
    RuleTooManyNegations,
    RuleCaseRepetition,
    RuleLongSentences,
    RulePredObjDistance,
    RuleInfVerbDistance
  ), ~ .x / sent_count)) %>%
  # remove variables identified as "u counts"
  select(!c(
    RuleTooFewVerbs,
    RuleTooManyNegations,
    RuleTooManyNominalConstructions,
    RuleCaseRepetition,
    RuleLongSentences,
    RulePredAtClauseBeginning,
    sent_count,
    word_count,
    syllab_count,
    char_count
  )) %>%
  # remove variables identified as unreliable
  select(!c(
    RuleAmbiguousRegards,
    RuleFunctionWordRepetition,
    RuleDoubleComparison,
    RuleWrongValencyCase,
    RuleWrongVerbonominalCase
  )) %>%
  # remove artificially limited variables
  select(!c(
    RuleCaseRepetition.max_repetition_frac,
    RuleCaseRepetition.max_repetition_frac.v
  )) %>%
  # remove further variables belonging to the 'acceptability' category
  select(!c(RuleIncompleteConjunction)) %>%
  unite("strata", c(subcorpus, class), sep = "_", remove = FALSE) %>%
  mutate(across(c(class), ~ as.factor(.x)))

# no NAs should be present now
data_clean[!complete.cases(data_clean), ]
```

```
## # A tibble: 754 x 84
##    KUK_ID          FileName FileFormat strata subcorpus SourceID DocumentVersion
##    <chr>           <chr>    <chr>      <chr>  <chr>     <chr>    <chr>
##  1 673b7a37c6537d~ 002_Kom~ TXT        KUKY_~ KUKY      <NA>     Original
##  2 673b7a37c6537d~ 006_Chc~ TXT        KUKY_~ KUKY      <NA>     Redesign
##  3 673b7a37c6537d~ 004_Nev~ TXT        KUKY_~ KUKY      <NA>     Original
##  4 673b7a37c6537d~ 008_Pol~ TXT        KUKY_~ KUKY      <NA>     Original
##  5 673b7a37c6537d~ 005_Och~ TXT        KUKY_~ KUKY      <NA>     Original
##  6 673b7a37c6537d~ 016_Obc~ TXT        KUKY_~ KUKY      <NA>     Original
```

```
##  7 673b7a37c6537d~ 019_Dět~ TXT        KUKY_~ KUKY      <NA>     Redesign
##  8 673b7a37c6537d~ 007_DŮC~ TXT        KUKY_~ KUKY      <NA>     Redesign
##  9 673b7a37c6537d~ 024_Opa~ TXT        KUKY_~ KUKY      <NA>     Original
## 10 673b7a37c6537d~ 047_Dav~ TXT        KUKY_~ KUKY      <NA>     Original
## # i 744 more rows
## # i 77 more variables: ParentDocumentID <chr>, LegalActType <chr>,
## #   Objectivity <chr>, Bindingness <lgl>, AuthorType <chr>,
## #   RecipientType <chr>, RecipientIndividuation <chr>, Anonymized <chr>,
## #   `Recipient Type` <chr>, class <fct>, RuleAbstractNouns <dbl>,
## #   RuleAnaphoricReferences <dbl>,
## #   RuleCaseRepetition.max_repetition_count <dbl>, ...
```

Filter for features identified as important

This may not be necessary, as the identification was crucial to the EFA above all, so that features irrelevant for readability would not appear in the model. It may be useful to compare the importances of a model trained on all features and on a selected-feature model.

```r
selected_features_tibble <- read_csv("../efa/selected_features.csv")
```

```
## Rows: 67 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): feat_name
## dbl (1): p_value
## lgl (1): selected
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
formula_all <- reformulate(
  selected_features_tibble %>% pull(feat_name), "class"
)
formula_selected <- reformulate(
  selected_features_tibble %>% filter(selected) %>% pull(feat_name), "class"
)
```

Split and folds

```r
.split_prop <- 4 / 5
.no_folds <- 10

split <- data_clean %>% initial_split(prop = .split_prop, strata = strata)

training_set <- training(split)
testing_set <- testing(split)

folds <- vfold_cv(training_set, v = .no_folds, strata = strata)

nrow(training_set)
```

```
## [1] 601
```

```r
training_set %>%
  select(subcorpus, class) %>%
  table()
```

```
##                class
## subcorpus      bad good
##   CzCDC        170    0
##   FrBo          62  183
##   KUKY          65   87
##   LiFRLaw        2    0
##   OmbuFlyers    32    0
```

```r
nrow(testing_set)
```

```
## [1] 153
```

```r
testing_set %>%
  select(subcorpus, class) %>%
  table()
```

```
##                class
## subcorpus      bad good
##   CzCDC         44    0
##   FrBo          16   46
##   KUKY          17   23
##   LiFRLaw        1    0
##   OmbuFlyers     6    0
```

Experimental model

To familiarize myself with the library and CRFs.

```r
training_split <- training_set %>%
  initial_split(prop = .split_prop, strata = strata)
train_subset <- training(training_split)
devtest_subset <- testing(training_split)

model_rf_exp <- cforest(
  formula_selected,
  data = train_subset, controls = cforest_control(ntree = 1000)
)

predictions_exp <- predict(model_rf_exp, newdata = devtest_subset)
confusionMatrix(predictions_exp, devtest_subset$class, positive = "good")
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction bad good
##       bad   49   18
##       good  19   37
##
##                Accuracy : 0.6992
##                  95% CI : (0.61, 0.7786)
##     No Information Rate : 0.5528
##     P-Value [Acc > NIR] : 0.00063
```

```
##
##                   Kappa : 0.3926
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.6727
##             Specificity : 0.7206
##          Pos Pred Value : 0.6607
##          Neg Pred Value : 0.7313
##              Prevalence : 0.4472
##          Detection Rate : 0.3008
##    Detection Prevalence : 0.4553
##       Balanced Accuracy : 0.6967
##
##        'Positive' Class : good
##
```

```
importances_exp <- varimp(model_rf_exp)
# computationally expensive
# cimportances_exp <- varimp(model_rf_exp, conditional = TRUE)
```

MFV model

```
(nrow(data_clean %>% filter(class == "bad")) / nrow(data_clean)) %>%
  round(3)
```

```
## [1] 0.55
```

```
(nrow(training_set %>% filter(class == "bad")) / nrow(training_set)) %>%
  round(3)
```

```
## [1] 0.551
```

```
(nrow(testing_set %>% filter(class == "bad")) / nrow(testing_set)) %>%
  round(3)
```

```
## [1] 0.549
```

Helpers

```
ntree_tune_levels <- 500 + 0:8 * 250

tune_crf <- function(formula, folds, ntree_tune_levels) {
  accuracy_column <- numeric()
  ntree_column <- numeric()
  fold_column <- numeric()

  for (ntree_ in ntree_tune_levels) {
    message(paste0(c("ntree_ ", ntree_), collapse = " "))
    ctrl <- cforest_control(ntree = ntree_)

    for (i in seq_len(nrow(folds))) {
      alldata <- pull(folds[i, 1])[[1]]$data
      trindices <- pull(folds[i, 1])[[1]]$in_id
```

```
      trdata <- alldata[trindices, ]
      tsdata <- alldata[-trindices, ]

      model <- cforest(formula, data = trdata, controls = ctrl)
      pred <- predict(model, newdata = tsdata)

      cm <- confusionMatrix(pred, tsdata$class, positive = "good")

      ntree_column <- c(ntree_column, ntree_)
      fold_column <- c(fold_column, i)
      accuracy_column <- c(accuracy_column, cm$overall["Accuracy"])
    }
  }

  data.frame(
    ntree = ntree_column,
    fold = fold_column,
    accuracy = accuracy_column
  )
}
```

Selected-features model

Tune

```
tune_df_sel <- tune_crf(formula_selected, folds, ntree_tune_levels)
```

```
## ntree_  500

## ntree_  750

## ntree_  1000

## ntree_  1250

## ntree_  1500

## ntree_  1750

## ntree_  2000

## ntree_  2250

## ntree_  2500
```

```
tune_df_sel %>%
  group_by(ntree) %>%
  summarize(mean_acc = mean(accuracy), sd_acc = sd(accuracy))
```

```
## # A tibble: 9 x 3
##   ntree mean_acc sd_acc
##   <dbl>    <dbl>  <dbl>
## 1   500    0.759 0.0226
## 2   750    0.760 0.0361
## 3  1000    0.757 0.0362
## 4  1250    0.759 0.0346
## 5  1500    0.762 0.0308
```

```
## 6  1750    0.762 0.0358
## 7  2000    0.759 0.0336
## 8  2250    0.760 0.0393
## 9  2500    0.760 0.0430
```

```r
tune_df_sel %>%
  group_by(fold) %>%
  summarize(mean_acc = mean(accuracy), sd_acc = sd(accuracy))
```

```
## # A tibble: 10 x 3
##     fold mean_acc  sd_acc
##    <dbl>    <dbl>   <dbl>
##  1     1    0.806 0.0106
##  2     2    0.738 0.00711
##  3     3    0.750 0.00723
##  4     4    0.741 0.0206
##  5     5    0.772 0.00833
##  6     6    0.702 0.0194
##  7     7    0.748 0.0130
##  8     8    0.746 0
##  9     9    0.805 0.00862
## 10    10    0.791 0.00575
```

```r
best_ntree_sel <- tune_df_sel %>%
  group_by(ntree) %>%
  summarize(mean_acc = mean(accuracy)) %>%
  arrange(-mean_acc) %>%
  head(n = 1) %>%
  pull(ntree)
```

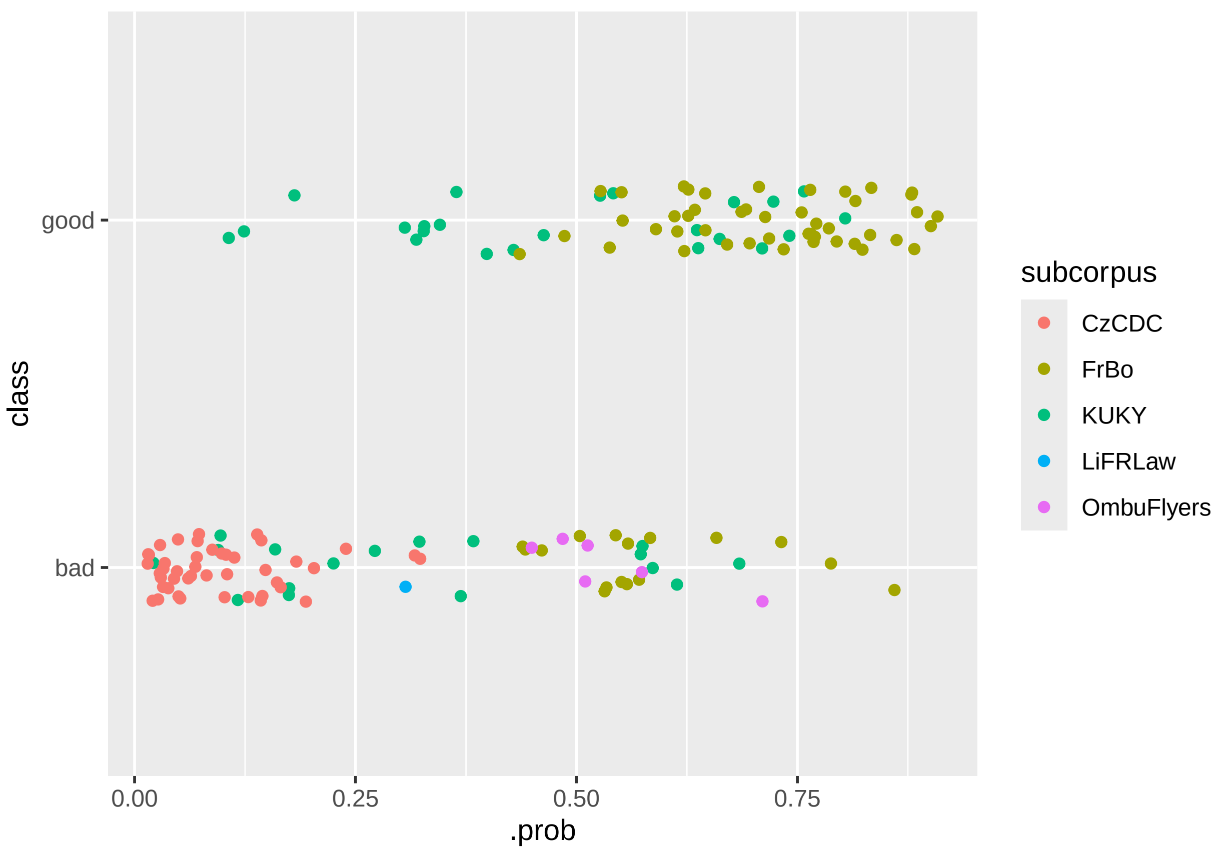
Fit

```r
model_crf_sel <- cforest(
  formula_selected, training_set,
  controls = cforest_control(ntree = best_ntree_sel)
)

predictions_sel_prob <- predict(
  model_crf_sel,
  newdata = testing_set, type = "prob"
) %>%
  map(function(x) x[1, 2]) %>%
  unlist() %>%
  as.vector()
predictions_sel <- if_else(predictions_sel_prob > 0.5, "good", "bad") %>%
  as.factor()

confusionMatrix(predictions_sel, testing_set$class, positive = "good")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   62   14
##       good  22   55
```

```
##
##                Accuracy : 0.7647
##                  95% CI : (0.6894, 0.8294)
##     No Information Rate : 0.549
##     P-Value [Acc > NIR] : 2.577e-08
##
##                   Kappa : 0.5297
##
##  Mcnemar's Test P-Value : 0.2433
##
##             Sensitivity : 0.7971
##             Specificity : 0.7381
##          Pos Pred Value : 0.7143
##          Neg Pred Value : 0.8158
##              Prevalence : 0.4510
##          Detection Rate : 0.3595
##    Detection Prevalence : 0.5033
##       Balanced Accuracy : 0.7676
##
##        'Positive' Class : good
##
```

```
testing_set_sel <- testing_set %>%
  mutate(.prob = predictions_sel_prob, .pred = predictions_sel)
testing_set_sel %>% ggplot(aes(x = .prob, y = class, color = subcorpus)) +
  geom_jitter(width = 0, height = 0.1)
```



```
testing_set_sel %>%
  mutate(abs_dev = abs(0.5 - .prob)) %>%
  filter(class != .pred & abs_dev > 0.25) %>%
  arrange(-abs_dev) %>%
  select(subcorpus, FileName, class, .prob, abs_dev) %>%
  mutate(across(c(.prob, abs_dev), ~ round(.x, 3))) %>%
  as.data.frame()
```

```
##   subcorpus
## 1      KUKY
## 2      KUKY
## 3      FrBo
## 4      KUKY
## 5      FrBo
##                                                                                    FileName
## 1                                                     0217_6Afs_2000035_20210219141328__1_
## 2                                                                              11_vizum_pred
## 3 orig_Co můžete dělat, pokud obec postupuje při prodeji nebo pronájmu pozemků nezákonně_final
## 4                                                                                   Odvolani
## 5                                                               orig_Jak probíhá správní řízení
##   class .prob abs_dev
## 1  good 0.107   0.393
## 2  good 0.124   0.376
## 3   bad 0.860   0.360
## 4  good 0.181   0.319
## 5   bad 0.788   0.288
```

All-features model

Tune

```
tune_df_all <- tune_crf(formula_all, folds, ntree_tune_levels)
```

```
## ntree_  500

## ntree_  750

## ntree_  1000

## ntree_  1250

## ntree_  1500

## ntree_  1750

## ntree_  2000

## ntree_  2250

## ntree_  2500
```

```
tune_df_all %>%
  group_by(ntree) %>%
  summarize(mean_acc = mean(accuracy), sd_acc = sd(accuracy))
```

```
## # A tibble: 9 x 3
##   ntree mean_acc sd_acc
##   <dbl>    <dbl>  <dbl>
```

```
## 1   500    0.757 0.0347
## 2   750    0.759 0.0438
## 3  1000    0.759 0.0382
## 4  1250    0.760 0.0351
## 5  1500    0.757 0.0339
## 6  1750    0.757 0.0382
## 7  2000    0.759 0.0345
## 8  2250    0.760 0.0340
## 9  2500    0.759 0.0330
```

```r
tune_df_all %>%
  group_by(fold) %>%
  summarize(mean_acc = mean(accuracy), sd_acc = sd(accuracy))
```

```
## # A tibble: 10 x 3
##     fold mean_acc  sd_acc
##    <dbl>    <dbl>   <dbl>
##  1     1    0.799 0.00794
##  2     2    0.729 0.00711
##  3     3    0.754 0
##  4     4    0.743 0.0147
##  5     5    0.802 0.0130
##  6     6    0.704 0.0162
##  7     7    0.741 0.00878
##  8     8    0.731 0.0102
##  9     9    0.793 0
## 10    10    0.791 0.00575
```

```r
best_ntree_all <- tune_df_all %>%
  group_by(ntree) %>%
  summarize(mean_acc = mean(accuracy)) %>%
  arrange(-mean_acc) %>%
  head(n = 1) %>%
  pull(ntree)
```

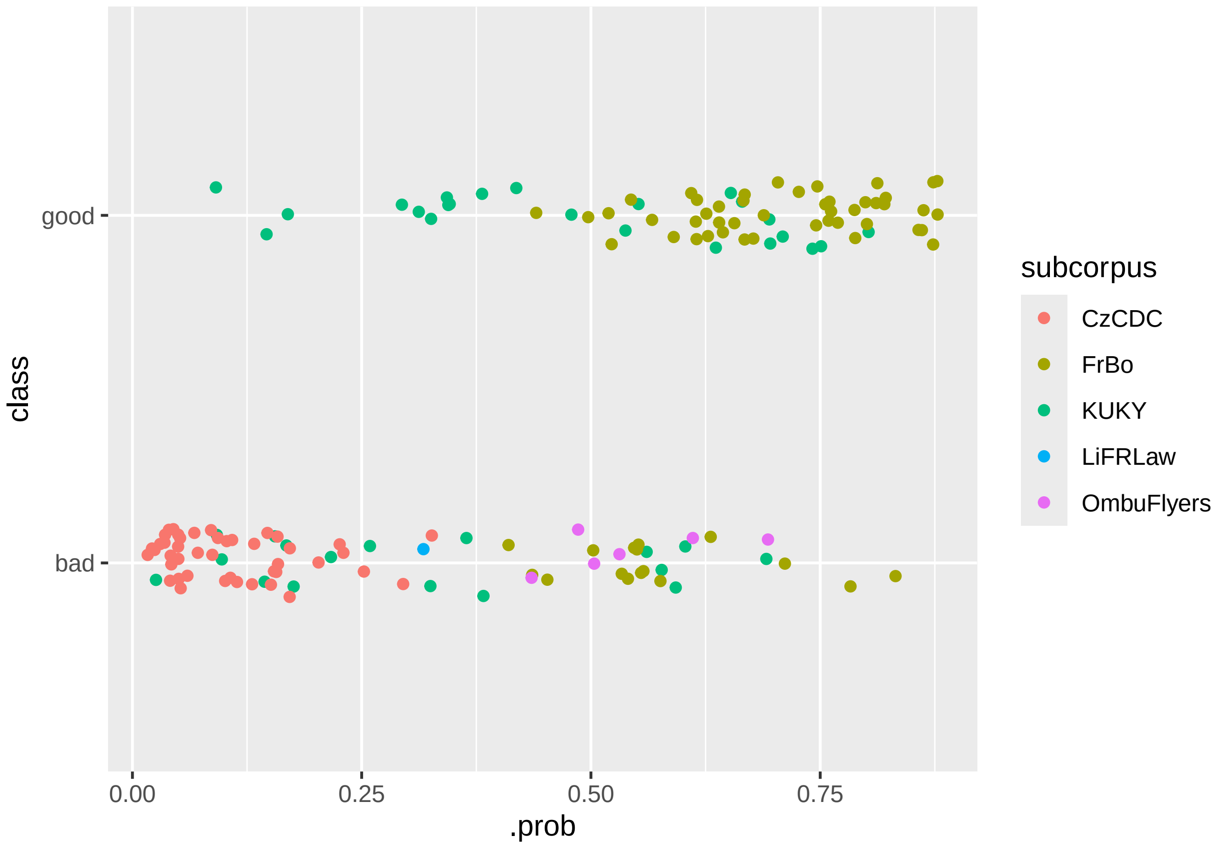
Fit

```r
model_crf_all <- cforest(
  formula_all, training_set,
  controls = cforest_control(ntree = best_ntree_all)
)

predictions_all_prob <- predict(
  model_crf_all,
  newdata = testing_set, type = "prob"
) %>%
  map(function(x) x[1, 2]) %>%
  unlist() %>%
  as.vector()
predictions_all <- if_else(predictions_all_prob > 0.5, "good", "bad") %>%
  as.factor()

confusionMatrix(predictions_all, testing_set$class, positive = "good")
```

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

```
##
##           Reference
## Prediction bad good
##       bad   62   14
##       good  22   55
##
##                Accuracy : 0.7647
##                  95% CI : (0.6894, 0.8294)
##     No Information Rate : 0.549
##     P-Value [Acc > NIR] : 2.577e-08
##
##                   Kappa : 0.5297
##
##  Mcnemar's Test P-Value : 0.2433
##
##             Sensitivity : 0.7971
##             Specificity : 0.7381
##          Pos Pred Value : 0.7143
##          Neg Pred Value : 0.8158
##              Prevalence : 0.4510
##          Detection Rate : 0.3595
##    Detection Prevalence : 0.5033
##       Balanced Accuracy : 0.7676
##
##        'Positive' Class : good
##
```

```r
testing_set_all <- testing_set %>%
  mutate(.prob = predictions_all_prob, .pred = predictions_all)
testing_set_all %>% ggplot(aes(x = .prob, y = class, color = subcorpus)) +
  geom_jitter(width = 0, height = 0.1)
```




```
testing_set_all %>%
  mutate(abs_dev = abs(0.5 - .prob)) %>%
  filter(class != .pred & abs_dev > 0.25) %>%
  arrange(-abs_dev) %>%
  select(subcorpus, FileName, class, .prob, abs_dev) %>%
  mutate(across(c(.prob, abs_dev), ~ round(.x, 3))) %>%
  as.data.frame()
```

```
##   subcorpus
## 1      KUKY
## 2      KUKY
## 3      FrBo
## 4      KUKY
## 5      FrBo
##                                                                            FileName
## 1                                                   0217_6Afs_2000035_20210219141328__1_
## 2                                                                      11_vizum_pred
## 3 orig_Co můžete dělat, pokud obec postupuje při prodeji nebo pronájmu pozemků nezákonně_final
## 4                                                                            Odvolani
## 5                                                          orig_Jak probíhá správní řízení
##   class .prob abs_dev
## 1  good 0.091   0.409
## 2  good 0.146   0.354
## 3   bad 0.832   0.332
## 4  good 0.169   0.331
## 5   bad 0.783   0.283
```