# Classifier

```r
set.seed(42)

library(caret) # highly correlated features removal
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```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 dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```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 recipes::fixed()         masks stringr::fixed()
## x dplyr::lag()             masks stats::lag()
## x purrr::lift()            masks caret::lift()
## 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()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```r
library(e1071)
```

```
##
## Attaching package: 'e1071'
##
```

```
## The following object is masked from 'package:tune':
##
##     tune
##
## The following object is masked from 'package:rsample':
##
##     permutations
##
## The following object is masked from 'package:parsnip':
##
##     tune
```

## Helpers

```
train_svm <- function(
    training_set,
    testing_set,
    columns,
    kernel = "radial",
    gamma = if (is.vector(training_set)) 1 else 1 / ncol(training_set),
    cost = 1) {
  model <- svm(
    training_set[columns],
    training_set$class,
    kernel = kernel, type = "C-classification",
    gamma = gamma,
    cost = cost,
    probability = TRUE,
    cross = 10
  )

  if (is.null(testing_set)) {
    return(list(
      model = model
    ))
  }

  pred <- predict(model, testing_set[columns], probability = TRUE)
  set_with_preds <- testing_set %>%
    mutate(
      pred = pred,
      prob_good = attr(pred, "probabilities")[, "good"],
      prob_bad = attr(pred, "probabilities")[, "bad"]
    )

  cm <- confusionMatrix(
    set_with_preds$pred, set_with_preds$class,
    mode = "everything"
  )

  return(list(
    model = model,
    prediction_set = set_with_preds,
```

```r
    cm = cm
  ))
}

train_glm <- function(training_set, testing_set, columns) {
  formula <- reformulate(colnames(training_set[columns]), "class")
  model <- glm(
    formula,
    training_set,
    family = "binomial"
  )
  pred <- predict(model, testing_set[columns], type = "response")
  set_with_preds <- testing_set %>%
    mutate(
      prob_good = pred,
      prob_bad = 1 - pred,
      pred = if_else(pred > .5, "good", "bad") %>%
        factor(levels = c("bad", "good"))
    )

  cm <- confusionMatrix(
    set_with_preds$pred, set_with_preds$class,
    mode = "everything"
  )

  return(list(
    model = model,
    prediction_set = set_with_preds,
    cm = cm
  ))
}

get_mismatch_details <- function(data_with_predictions) {
  print(
    data_with_predictions %>%
      ggplot(aes(x = prob_good, y = class, color = subcorpus)) +
      geom_jitter(height = 0.2, width = 0)
  )

  cat("Confusion matrices by subcorpora:\n")
  data_with_predictions %>%
    select(pred, class, subcorpus) %>%
    table() %>%
    print()

  cat("\n")

  deviations <- data_with_predictions %>%
    filter(pred != class) %>%
    mutate(abs_dev = abs(prob_good - 0.5)) %>%
    arrange(-abs_dev)

  cat("Greatest deviations:\n")
  deviations %>%
    select(abs_dev, prob_good, class, subcorpus, FileName) %>%
    mutate(across(c(prob_good, abs_dev), ~ round(.x, 3))) %>%
    print(n = round(nrow(data_with_predictions) / 5))

  cat("Names of highest-deviating documents:\n")
  highest_deviation_names <- deviations %>%
    filter(abs_dev >= 0.25) %>%
    arrange(-abs_dev) %>%
    pull(FileName)

  print(highest_deviation_names)

  return(list(
    deviations = deviations, highest_deviations = highest_deviation_names
  ))
}

plot_outlier <- function(doc_name, variable_importances, dataset) {
  important_variables <- sort(variable_importances) %>% tail(n = 9)
  varnames <- names(important_variables)

  dmut <- dataset %>%
    select(KUK_ID, FileName, class, all_of(varnames)) %>%
    mutate(across(all_of(varnames), ~ scale(.x))) %>%
    pivot_longer(
      all_of(varnames),
      names_to = "feature", values_to = "value"
    ) %>%
    mutate(across(value, ~ .x[, 1]))

  cat(
    nrow(dmut %>% filter(value > 5)),
    "observation(s) removed from the plot\n"
  )
  dmutf <- dmut %>% filter(value <= 5)

  dmutf %>%
    ggplot(aes(x = class, y = value)) +
    facet_wrap(~feature) +
    geom_boxplot() +
    geom_point(
      data = dmut %>% filter(FileName == doc_name), color = "red", size = 5
    ) +
    labs(y = "measurements (scaled)")
}
```

## 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
prettify_feat_name <- function(x) {
  name <- pull(pretty_names %>%
    filter(name_orig == x), name_pretty)
  if (length(name) == 1) {
    return(name)
  } else {
    return(x)
  }
}

prettify_feat_name_vector <- function(x) {
  map(
    x,
    prettify_feat_name
  ) %>% unlist()
}


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

```
## Rows: 753 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
.firstnonmetacolumn <- 17

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.v = 0,
    RuleAmbiguousRegards = 0,
    RuleReflexivePassWithAnimSubj = 0,
    RuleTooManyNegations = 0,
    RuleTooManyNegations.max_negation_frac.v = 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.v = 0,
    RulePredObjDistance = 0,
    RulePredObjDistance.max_distance.v = 0,
    RuleInfVerbDistance = 0,
    RuleInfVerbDistance.max_distance.v = 0,
    RuleMultiPartVerbs = 0,
    RuleMultiPartVerbs.max_distance.v = 0,
    RuleLongSentences.max_length.v = 0,
```

```r
    RulePredAtClauseBeginning.max_order.v = 0,
    RuleVerbalNouns = 0,
    RuleDoubleComparison = 0,
    RuleWrongValencyCase = 0,
    RuleWrongVerbonominalCase = 0,
    RuleIncompleteConjunction = 0
  )) %>%
  # replace NAs with medians
  mutate(across(c(
    RuleDoubleAdpos.max_allowable_distance,
    RuleTooManyNegations.max_negation_frac,
    RuleTooManyNegations.max_allowable_negations,
    RulePredSubjDistance.max_distance,
    RulePredObjDistance.max_distance,
    RuleInfVerbDistance.max_distance,
    RuleMultiPartVerbs.max_distance
  ), ~ coalesce(., median(., na.rm = TRUE)))) %>%
  # merge GPs
  mutate(
    GPs = RuleGPcoordovs +
      RuleGPdeverbaddr +
      RuleGPpatinstr +
      RuleGPdeverbsubj +
      RuleGPadjective +
      RuleGPpatbenperson +
      RuleGPwordorder
  ) %>%
  select(!c(
    RuleGPcoordovs,
    RuleGPdeverbaddr,
    RuleGPpatinstr,
    RuleGPdeverbsubj,
    RuleGPadjective,
    RuleGPpatbenperson,
    RuleGPwordorder
  ))

data_clean <- data_no_nas %>%
  # norm data expected to correlate with text length
  mutate(across(c(
    GPs,
    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 text-length dependent
  select(!c(
    RuleTooFewVerbs,
    RuleTooManyNegations,
    RuleTooManyNominalConstructions,
    RuleCaseRepetition,
    RuleLongSentences,
    RulePredAtClauseBeginning,
    syllab_count,
    char_count
  )) %>%
  # remove variables identified as unreliable
  select(!c(
    RuleAmbiguousRegards,
    RuleFunctionWordRepetition,
    RuleDoubleComparison,
    RuleWrongValencyCase,
    RuleWrongVerbonominalCase
  )) %>%
  # remove further variables belonging to the 'acceptability' category
  select(!c(RuleIncompleteConjunction)) %>%
  # remove artificially limited variables
  select(!c(
    RuleCaseRepetition.max_repetition_frac,
    RuleCaseRepetition.max_repetition_frac.v
  )) %>%
  # remove variables with too many NAs
  select(!c(
    RuleDoubleAdpos.max_allowable_distance,
    RuleDoubleAdpos.max_allowable_distance.v
  )) %>%
  mutate(across(c(
    class,
    FileFormat,
    subcorpus,
    DocumentVersion,
    LegalActType,
    Objectivity,
    AuthorType,
    RecipientType,
    RecipientIndividuation,
    Anonymized
  ), ~ as.factor(.x)))

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

```
## # A tibble: 0 x 77
## # i 77 variables: KUK_ID <chr>, FileName <chr>, FileFormat <fct>,
## #   subcorpus <fct>, SourceID <chr>, DocumentVersion <fct>,
## #   ParentDocumentID <chr>, LegalActType <fct>, Objectivity <fct>,
## #   Bindingness <lgl>, AuthorType <fct>, RecipientType <fct>,
## #   RecipientIndividuation <fct>, Anonymized <fct>, Recipient Type <chr>,
## #   class <fct>, RuleAbstractNouns <dbl>, RuleAnaphoricReferences <dbl>,
## #   RuleCaseRepetition.max_repetition_count <dbl>, ...
```

```
colnames(data_clean) <- prettify_feat_name_vector(colnames(data_clean))

data_scaled <- data_clean %>%
  mutate(across(all_of(.firstnonmetacolumn:ncol(data_clean)), ~ scale(.x)[, 1]))

data_stratified <- data_scaled %>%
  unite("strata", c("class", "subcorpus"), remove = FALSE)
```

## Important features identification

```
feature_importances <- read_csv("../importance_measures/featcomp.csv")
```

```
## Rows: 61 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): Variable, Sign
## dbl (15): Importance, p_value, estimate, wilcox_p, wilcox_r, kw_p, kw_chi2, ...
## lgl  (4): selected_pval, wilcox_sel, kw_sel, selected_reg
##
## 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.
```

```
selected_features_names <- feature_importances %>%
  filter(kw_sel) %>%
  pull(Variable)
```

## Formulas

```r
columns_all <- colnames(data_stratified)[
  (.firstnonmetacolumn + 1):ncol(data_stratified)
]
columns_readabilty_forms <- c("ari", "cli", "fkgl", "fre", "gf", "smog")

correlating90 <- findCorrelation(
  cor(data_stratified[columns_all]),
  cutoff = 0.9, verbose = TRUE, names = TRUE
)
```

```
## Compare row 11  and column  42 with corr  0.943
##   Means:  0.349 vs 0.174 so flagging column 11
## Compare row 42  and column  48 with corr  0.978
##   Means:  0.333 vs 0.168 so flagging column 42
## Compare row 48  and column  57 with corr  0.987
##   Means:  0.319 vs 0.163 so flagging column 48
## Compare row 57  and column  46 with corr  0.948
##   Means:  0.303 vs 0.158 so flagging column 57
## Compare row 43  and column  44 with corr  0.96
##   Means:  0.241 vs 0.154 so flagging column 43
## Compare row 60  and column  49 with corr  0.958
##   Means:  0.176 vs 0.151 so flagging column 60
## Compare row 58  and column  55 with corr  0.979
##   Means:  0.168 vs 0.151 so flagging column 58
## Compare row 50  and column  53 with corr  0.964
##   Means:  0.167 vs 0.15 so flagging column 50
## All correlations <= 0.9
```

```r
columns_notcorrelating90 <- c()
for (col in columns_all) {
  if (!(col %in% correlating90)) {
    columns_notcorrelating90 <- c(columns_notcorrelating90, col)
  }
}

correlating85 <- findCorrelation(
  cor(data_stratified[columns_all]),
  cutoff = 0.85, verbose = TRUE, names = TRUE
)
```

```
## Compare row 11  and column  42 with corr  0.943
##   Means:  0.349 vs 0.174 so flagging column 11
## Compare row 42  and column  48 with corr  0.978
##   Means:  0.333 vs 0.168 so flagging column 42
## Compare row 48  and column  57 with corr  0.987
##   Means:  0.319 vs 0.163 so flagging column 48
## Compare row 57  and column  46 with corr  0.948
##   Means:  0.303 vs 0.158 so flagging column 57
## Compare row 46  and column  47 with corr  0.852
##   Means:  0.276 vs 0.154 so flagging column 46
## Compare row 28  and column  41 with corr  0.888
##   Means:  0.273 vs 0.148 so flagging column 28
## Compare row 43  and column  44 with corr  0.96
##   Means:  0.233 vs 0.145 so flagging column 43
## Compare row 60  and column  49 with corr  0.958
##   Means:  0.176 vs 0.142 so flagging column 60
## Compare row 49  and column  58 with corr  0.887
##   Means:  0.163 vs 0.141 so flagging column 49
## Compare row 58  and column  55 with corr  0.979
##   Means:  0.156 vs 0.14 so flagging column 58
## Compare row 50  and column  53 with corr  0.964
##   Means:  0.164 vs 0.139 so flagging column 50
## Compare row 54  and column  51 with corr  0.858
##   Means:  0.117 vs 0.14 so flagging column 51
## All correlations <= 0.85
```

```
columns_notcorrelating85 <- c()
for (col in columns_all) {
  if (!(col %in% correlating85)) {
    columns_notcorrelating85 <- c(columns_notcorrelating85, col)
  }
}

correlating75 <- findCorrelation(
  cor(data_stratified[columns_all]),
  cutoff = 0.75, verbose = TRUE, names = TRUE
)
```

```
## Compare row 11  and column  42 with corr  0.943
##   Means:  0.349 vs 0.174 so flagging column 11
## Compare row 42  and column  48 with corr  0.978
##   Means:  0.333 vs 0.168 so flagging column 42
## Compare row 48  and column  57 with corr  0.987
##   Means:  0.319 vs 0.163 so flagging column 48
## Compare row 57  and column  46 with corr  0.948
##   Means:  0.303 vs 0.158 so flagging column 57
## Compare row 46  and column  47 with corr  0.852
##   Means:  0.276 vs 0.154 so flagging column 46
## Compare row 28  and column  35 with corr  0.816
##   Means:  0.273 vs 0.148 so flagging column 28
## Compare row 35  and column  41 with corr  0.76
##   Means:  0.255 vs 0.144 so flagging column 35
## Compare row 41  and column  59 with corr  0.763
##   Means:  0.238 vs 0.14 so flagging column 41
## Compare row 43  and column  44 with corr  0.96
##   Means:  0.225 vs 0.137 so flagging column 43
## Compare row 56  and column  60 with corr  0.779
##   Means:  0.18 vs 0.134 so flagging column 56
## Compare row 45  and column  60 with corr  0.772
##   Means:  0.187 vs 0.132 so flagging column 45
## Compare row 60  and column  49 with corr  0.958
##   Means:  0.157 vs 0.13 so flagging column 60
## Compare row 49  and column  58 with corr  0.887
##   Means:  0.143 vs 0.129 so flagging column 49
## Compare row 58  and column  55 with corr  0.979
##   Means:  0.139 vs 0.129 so flagging column 58
## Compare row 50  and column  53 with corr  0.964
##   Means:  0.156 vs 0.128 so flagging column 50
## Compare row 54  and column  51 with corr  0.858
##   Means:  0.12 vs 0.128 so flagging column 51
## All correlations <= 0.75
```

```
columns_notcorrelating75 <- c()
for (col in columns_all) {
  if (!(col %in% correlating75)) {
    columns_notcorrelating75 <- c(columns_notcorrelating75, col)
  }
}
```

## Hyperparameters

```
colsids <- c(
  "all", "notcorrelating90",
  "notcorrelating85", "notcorrelating75"
)
colsets <- list(
  columns_all, columns_notcorrelating90,
  columns_notcorrelating85, columns_notcorrelating75
)
```

## Splits and folds

```
.splitprop <- 3 / 4

split <- initial_split(data_stratified, .splitprop, strata = strata)

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

training_set %>%
  select(class) %>%
  table()
```

```
## class
##  bad good
##  310  253
```

```
testing_set %>%
  select(class) %>%
  table()
```

```
## class
##  bad good
##  104   86
```

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

```
##              class
## subcorpus     bad good
##    CzCDC      157    0
##    FrBo        56  171
##    KUKY        64   82
##   LiFRLaw      3    0
##   OmbuFlyers  30    0
```

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

```
##             class
## subcorpus     bad good
##   CzCDC        54    0
##   FrBo         22   58
##   KUKY         20   28
##   LiFRLaw       0    0
##   OmbuFlyers    8    0
```

## Tune

```
tune_res <- tibble(
  columns = character(),
  kernel = character(),
  gamma = numeric(),
  cost = numeric(),
  error = numeric(),
  dispersion = numeric()
)

for (coli in seq_along(colsets)) {
  colsid <- colsids[coli]
  columns <- colsets[[coli]]

  message("tune radial on ", colsid)
  tune_radial <- tune.svm(training_set[columns], training_set$class,
    gamma = 10^(-3:3),
    cost = c(0.01, 0.1, 1, 10, 100, 1000),
    kernel = "radial"
  )
  tune_res <- tune_res %>%
    bind_rows(tune_radial$performances %>%
      mutate(kernel = "radial", columns = colsid))

  message("tune polynomial3 on ", colsid)
  tune_polynomial <- tune.svm(training_set[columns], training_set$class,
    gamma = 10^(-3:3),
    degree = 3,
    cost = c(0.01, 0.1, 1, 10, 100, 1000),
    kernel = "polynomial"
  )
  tune_res <- tune_res %>%
    bind_rows(tune_polynomial$performances %>%
      mutate(kernel = "polynomial3", columns = colsid))


  message("tune polynomial4 on ", colsid)
  tune_polynomial <- tune.svm(training_set[columns], training_set$class,
```

```r
    gamma = 10^(-3:3),
    degree = 4,
    cost = c(0.01, 0.1, 1, 10, 100, 1000),
    kernel = "polynomial"
  )
  tune_res <- tune_res %>%
    bind_rows(tune_polynomial$performances %>%
      mutate(kernel = "polynomial4", columns = colsid))


  message("tune polynomial5 on ", colsid)
  tune_polynomial <- tune.svm(training_set[columns], training_set$class,
    gamma = 10^(-3:3),
    degree = 5,
    cost = c(0.01, 0.1, 1, 10, 100, 1000),
    kernel = "polynomial"
  )
  tune_res <- tune_res %>%
    bind_rows(tune_polynomial$performances %>%
      mutate(kernel = "polynomial5", columns = colsid))


  message("tune sigmoid on ", colsid)
  tune_sigmoid <- tune.svm(training_set[columns], training_set$class,
    gamma = 10^(-3:3),
    cost = c(0.01, 0.1, 1, 10, 100, 1000),
    kernel = "sigmoid"
  )
  tune_res <- tune_res %>%
    bind_rows(tune_sigmoid$performances %>%
      mutate(kernel = "sigmoid", columns = colsid))
}
```

```
## tune radial on all

## tune polynomial3 on all

## tune polynomial4 on all

## tune polynomial5 on all

## tune sigmoid on all

## tune radial on notcorrelating90

## tune polynomial3 on notcorrelating90

## tune polynomial4 on notcorrelating90

## tune polynomial5 on notcorrelating90

## tune sigmoid on notcorrelating90

## tune radial on notcorrelating85

## tune polynomial3 on notcorrelating85

## tune polynomial4 on notcorrelating85

## tune polynomial5 on notcorrelating85
## tune sigmoid on notcorrelating85

## tune radial on notcorrelating75

## tune polynomial3 on notcorrelating75

## tune polynomial4 on notcorrelating75

## tune polynomial5 on notcorrelating75

## tune sigmoid on notcorrelating75
```

```
tune_res %>% write_csv("tune_results.csv")
# tune_res <- read_csv("tune_results.csv")

tune_res %>%
  arrange(error, -dispersion)
```

```
## # A tibble: 840 x 7
##    columns          kernel  gamma  cost error dispersion degree
##    <chr>            <chr>   <dbl> <dbl> <dbl>      <dbl>  <dbl>
##  1 notcorrelating90 radial  0.01      1 0.192     0.0514     NA
##  2 all              radial  0.01      1 0.197     0.0648     NA
##  3 notcorrelating90 radial  0.001   100 0.199     0.0595     NA
##  4 notcorrelating75 radial  0.01      1 0.202     0.0494     NA
##  5 notcorrelating85 radial  0.001     1 0.206     0.0595     NA
##  6 notcorrelating85 sigmoid 0.01      1 0.209     0.0631     NA
##  7 notcorrelating75 radial  0.001    10 0.211     0.0691     NA
##  8 notcorrelating85 radial  0.01      1 0.211     0.0569     NA
##  9 notcorrelating90 radial  0.001    10 0.213     0.0708     NA
## 10 notcorrelating90 sigmoid 0.001   100 0.217     0.0407     NA
## # i 830 more rows
```

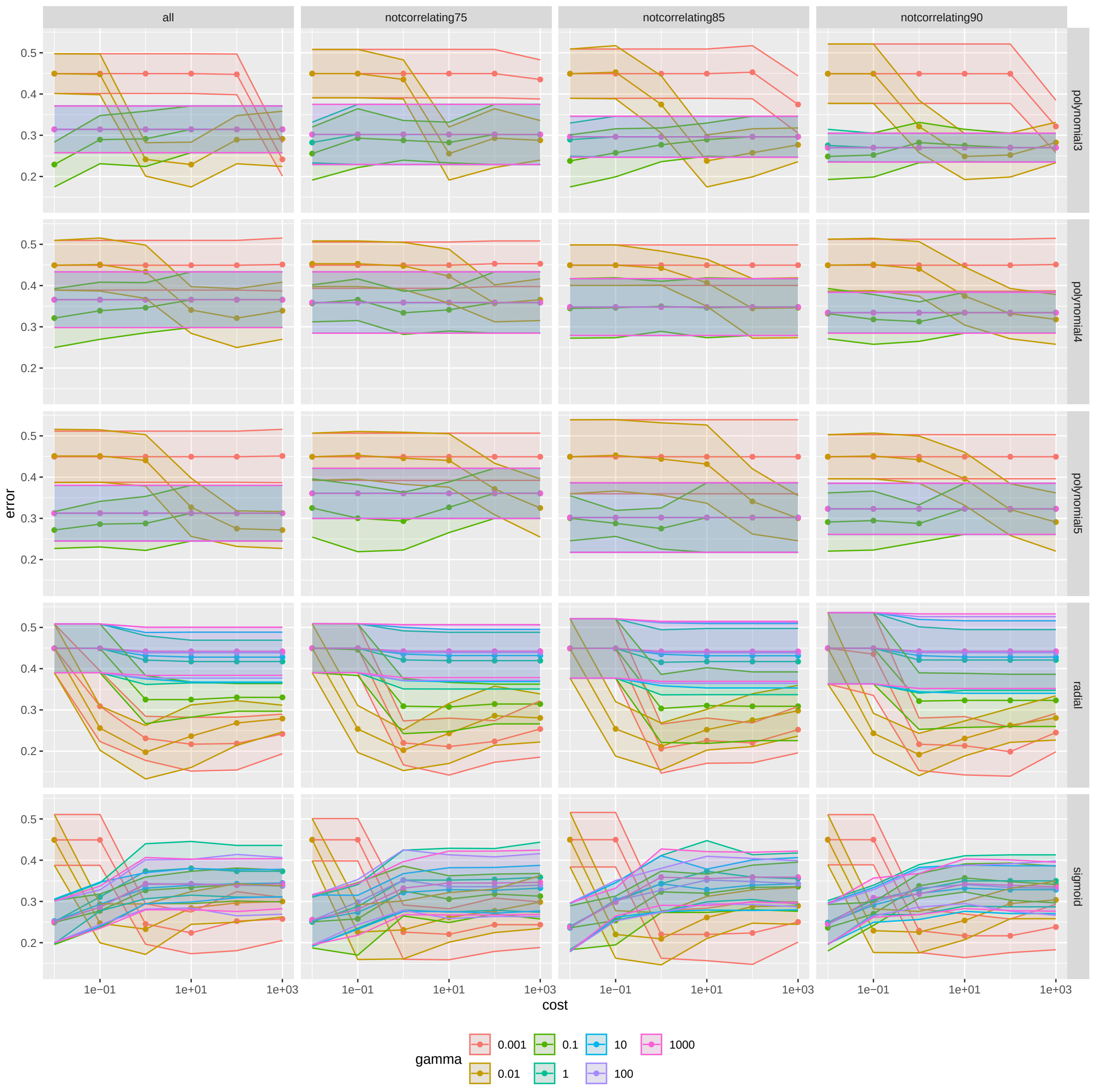
```
tune_res %>%
  filter(columns == "all") %>%
  arrange(error, -dispersion)
```

```
## # A tibble: 210 x 7
##    columns kernel      gamma   cost error dispersion degree
##    <chr>   <chr>       <dbl>  <dbl> <dbl>      <dbl>  <dbl>
##  1 all     radial      0.01    1    0.197     0.0648     NA
##  2 all     radial      0.001  10    0.217     0.0652     NA
##  3 all     radial      0.001 100    0.219     0.0641     NA
##  4 all     sigmoid     0.001  10    0.224     0.0506     NA
##  5 all     polynomial3 0.1     0.01 0.229     0.0543      3
##  6 all     polynomial3 0.01   10    0.229     0.0543      3
##  7 all     radial      0.001   1    0.231     0.0535     NA
##  8 all     sigmoid     0.01    1    0.233     0.0611     NA
##  9 all     radial      0.01   10    0.236     0.0759     NA
## 10 all     polynomial3 0.01    1    0.242     0.0404      3
## # i 200 more rows
```

```
tune_res %>%
  filter(str_detect(columns, "notcorrelating.*")) %>%
  arrange(error, -dispersion)
```

```
## # A tibble: 630 x 7
##    columns          kernel  gamma  cost error dispersion degree
##    <chr>            <chr>   <dbl> <dbl> <dbl>      <dbl>  <dbl>
##  1 notcorrelating90 radial  0.01      1 0.192     0.0514     NA
##  2 notcorrelating90 radial  0.001   100 0.199     0.0595     NA
##  3 notcorrelating75 radial  0.01      1 0.202     0.0494     NA
##  4 notcorrelating85 radial  0.001     1 0.206     0.0595     NA
##  5 notcorrelating85 sigmoid 0.01      1 0.209     0.0631     NA
##  6 notcorrelating75 radial  0.001    10 0.211     0.0691     NA
##  7 notcorrelating85 radial  0.01      1 0.211     0.0569     NA
##  8 notcorrelating90 radial  0.001    10 0.213     0.0708     NA
##  9 notcorrelating90 sigmoid 0.001   100 0.217     0.0407     NA
## 10 notcorrelating90 sigmoid 0.001    10 0.217     0.0528     NA
## # i 620 more rows
```

```r
tune_res %>%
  mutate(across(gamma, as.factor)) %>%
  ggplot(aes(
    x = cost, y = error, ymin = error - dispersion,
    ymax = error + dispersion, color = gamma, fill = gamma
  )) +
  geom_point() +
  geom_line() +
  geom_ribbon(alpha = 0.1) +
  scale_x_log10() +
  facet_grid(kernel ~ columns) +
  theme(legend.position = "bottom")
```



best:

- columns: notcorrelating85
- kernel: radial
- gamma: 0.001
- cost: 100

## SVM

```r
model_all <- train_svm(
  training_set, testing_set, columns_all, "radial",
  gamma = 0.01, cost = 1
)
model_all$cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   83   19
##       good  21   67
##
##                Accuracy : 0.7895
##                  95% CI : (0.7246, 0.8451)
##     No Information Rate : 0.5474
##     P-Value [Acc > NIR] : 2.844e-12
##
##                   Kappa : 0.576
##
##  Mcnemar's Test P-Value : 0.8744
##
##             Sensitivity : 0.7981
##             Specificity : 0.7791
##          Pos Pred Value : 0.8137
##          Neg Pred Value : 0.7614
##               Precision : 0.8137
##                  Recall : 0.7981
##                      F1 : 0.8058
##              Prevalence : 0.5474
##          Detection Rate : 0.4368
##    Detection Prevalence : 0.5368
##       Balanced Accuracy : 0.7886
##
##        'Positive' Class : bad
##
```

## SVM readability formulas

```
model_rf <- train_svm(
  training_set, testing_set, columns_readabilty_forms, "radial",
  gamma = 0.01, cost = 1
)
model_rf$cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   72   23
##       good  32   63
##
##                Accuracy : 0.7105
##                  95% CI : (0.6405, 0.7739)
##     No Information Rate : 0.5474
##     P-Value [Acc > NIR] : 2.942e-06
##
##                   Kappa : 0.4211
```

```
##
##  Mcnemar's Test P-Value : 0.2807
##
##             Sensitivity : 0.6923
##             Specificity : 0.7326
##          Pos Pred Value : 0.7579
##          Neg Pred Value : 0.6632
##               Precision : 0.7579
##                  Recall : 0.6923
##                      F1 : 0.7236
##              Prevalence : 0.5474
##          Detection Rate : 0.3789
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.7124
##
##        'Positive' Class : bad
##
```