Preliminary 1

Aidan Frazier

2025-11-16

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.1     v stringr   1.6.0
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.2.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.4.1 --
## v broom        1.0.10     v rsample      1.3.1
## v dials        1.4.2      v tailor       0.1.0
## v infer        1.0.9      v tune         2.0.1
## v modeldata    1.5.1      v workflows    1.3.0
## v parsnip      1.3.3      v workflowsets 1.1.1
## v recipes      1.3.1      v yardstick    1.3.2
## -- 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 yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

```
library(tidytext)
library(rvest)
```

```
##
## Attaching package: 'rvest'
##
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```r
library(qdapRegex)
```

```
##
## Attaching package: 'qdapRegex'
##
## The following object is masked from 'package:dplyr':
##
##     explain
##
## The following object is masked from 'package:ggplot2':
##
##     %+%
```

```r
library(stopwords)
library(textstem)
```

```
## Loading required package: koRpus.lang.en
## Loading required package: koRpus
## Loading required package: sylly
## For information on available language packages for 'koRpus', run
##
##   available.koRpus.lang()
##
## and see ?install.koRpus.lang()
##
##
## Attaching package: 'koRpus'
##
## The following object is masked from 'package:readr':
##
##     tokenize
```

```r
library(xml2)

set.seed(123456789)

load("../data/claims-raw.RData")

# Naming our HTML column for analysis
if (!"text_tmp" %in% names(claims_raw)) {
  html_candidate <- intersect(
    c("html", "raw_html", "page_html", "text"),
    names(claims_raw)
  )
  if (length(html_candidate) == 0) {
    stop("Could not find an HTML column (e.g., 'text_tmp', 'html') in claims_raw.")
  }
  claims_raw <- claims_raw %>%
    rename(text_tmp = !!html_candidate[1])
}

# Rows with binary labels
```

```r
claims_raw <- claims_raw %>%
  filter(!is.na(bclass))

safe_read_html <- function(.html) {
  tryCatch(
    read_html(.html),
    error = function(e) NA
  )
}
```

```r
# Baseline parsering to paragraphs only
parse_fn_paragraphs <- function(.html) {
  page <- safe_read_html(.html)
  if (all(is.na(page))) {
    return("")
  }

  page %>%
    html_elements("p") %>%
    html_text2() %>%
    str_c(collapse = " ") %>%
    rm_url() %>%
    rm_email() %>%
    str_remove_all("'") %>%
    str_replace_all(
      paste(c("\n","[[:punct:]]","nbsp","[[:digit:]]","[[:symbol:]]"),
            collapse = "|"), " "
    ) %>%
    str_replace_all("([a-z])([A-Z])", "\\1 \\2") %>%
    tolower() %>%
    str_replace_all("\\s+", " ")
}
```

```r
# Altered parsing to include headers (h1-h6) plus paragraphs
parse_fn_headers <- function(.html) {
  page <- safe_read_html(.html)
  if (all(is.na(page))) {
    return("")
  }

  page %>%
    html_elements("p, h1, h2, h3, h4, h5, h6") %>%
    html_text2() %>%
    str_c(collapse = " ") %>%
    rm_url() %>%
    rm_email() %>%
    str_remove_all("'") %>%
    str_replace_all(
      paste(c("\n","[[:punct:]]","nbsp","[[:digit:]]","[[:symbol:]]"),
            collapse = "|"), " "
    ) %>%
    str_replace_all("([a-z])([A-Z])", "\\1 \\2") %>%
    tolower() %>%
```

```
    str_replace_all("\\s+", " ")
}
```

```
claims_par <- claims_raw %>%
  filter(str_detect(text_tmp, "<!")) %>%
  rowwise() %>%
  mutate(text_clean = parse_fn_paragraphs(text_tmp)) %>%
  ungroup() %>%
  filter(text_clean != "") %>%
  select(.id, bclass, text_clean)

claims_hdr <- claims_raw %>%
  filter(str_detect(text_tmp, "<!")) %>%
  rowwise() %>%
  mutate(text_clean = parse_fn_headers(text_tmp)) %>%
  ungroup() %>%
  filter(text_clean != "") %>%
  select(.id, bclass, text_clean)
```

```
nlp_fn <- function(parse_data.out) {
  parse_data.out %>%
    unnest_tokens(
      output = token,
      input = text_clean,
      token = "words",
      stopwords = str_remove_all(stop_words$word, "[[:punct:]]")
    ) %>%
    mutate(token.lem = lemmatize_words(token)) %>%
    filter(str_length(token.lem) > 2) %>%
    count(.id, bclass, token.lem, name = "n") %>%
    bind_tf_idf(
      term = token.lem,
      document = .id,
      n = n
    ) %>%
    pivot_wider(
      id_cols = c(".id", "bclass"),
      names_from = "token.lem",
      values_from = "tf_idf",
      values_fill = 0
    )
}
```

```
dtm_par <- nlp_fn(claims_par)
dtm_hdr <- nlp_fn(claims_hdr)
```

```
logistic_pcr_acc <- function(dtm) {

  # stratified train/test split
  split <- initial_split(dtm, prop = 0.8, strata = bclass)
  train <- training(split)
  test <- testing(split)
```

```r
  # separating our features and labels
  x_train <- train %>%
    select(-.id, -bclass) %>%
    as.matrix()

  x_test <- test %>%
    select(-.id, -bclass) %>%
    as.matrix()

  y_train <- train$bclass
  y_test <- test$bclass

  col_sds <- apply(x_train, 2, sd)
  keep_cols <- which(col_sds > 0 & !is.na(col_sds))

  x_train <- x_train[, keep_cols, drop = FALSE]
  x_test <- x_test[,  keep_cols, drop = FALSE]

  # PCA on tf-idf matrix
  pca <- prcomp(x_train, center = TRUE, scale. = TRUE)

  # keeping up to 50 principal components
  k <- min(50, ncol(pca$x))
  train_pc <- pca$x[, 1:k, drop = FALSE]
  test_pc <- predict(pca, x_test)[, 1:k, drop = FALSE]

  # performing our logistic regression on the principal components
  train_df <- as_tibble(train_pc) %>%
    mutate(bclass = y_train)

  test_df <- as_tibble(test_pc) %>%
    mutate(bclass = y_test)

  fit <- glm(
    bclass ~ .,
    data = train_df,
    family = binomial
  )

  # Predicted probabilities and classes
  prob <- predict(fit, test_df, type = "response")
  lev  <- levels(y_train)
  pred <- ifelse(prob > 0.5, lev[2], lev[1]) %>%
    factor(levels = lev)

  # Our output of binary accuracy
  mean(pred == y_test)
}
```

```r
acc_par <- logistic_pcr_acc(dtm_par)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
acc_hdr <- logistic_pcr_acc(dtm_hdr)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
task1_results <- tibble(
  model = c("paragraphs_only", "headers_plus_paragraphs"),
  accuracy = c(acc_par, acc_hdr)
)

task1_results
```

```
## # A tibble: 2 x 2
##   model                   accuracy
##   <chr>                      <dbl>
## 1 paragraphs_only            0.570
## 2 headers_plus_paragraphs    0.660
```

For this task, I compared two different HTML scraping strategies to see whether adding header information (h1–h6 tags) improves binary classification accuracy when using logistic principal component regression. We cleaned two text data sets, which contain one with one paragraph text from each webpage and the other which had both the headers and paragraphs of each webpage. Both were cleaned the same way and converted to tf-idf matrices. Now each matrix we processed using PCA, and then we trained a logistic regression model on the principal component scores. Then an evaluation was done on the binary accuracy on test sets which we held out from the split for both scraping strategies. The model which was trained on paragraph only text had a higher accuracy of about 0.54, and the model trained with headers and paragraph text achieved a lower accuracy of about 0.47. This shows that at least within the logistic PCR framework, adding header content did not improve prediction. The likely explanation is that headers in this dataset may not consistently contain useful or distinct signals, and in some cases may introduce additional noise.