**Overview**

In this post we will

* talk about The Federalist Papers
* access and tidy the text using the tidytext package
* apply our model to the data to predict the author of the disputed papers

**The Federalist Papers**

In the early days of The United States of America around the time when the Constitution was being signed did a series of articles published in various newspapers. These papers where writing under the false name *Publius*. It was later revealed to have been the collected works of Alexander Hamilton, James Madison and John Jay.

The Interesting thing in this was that the authorship of these papers were not consistent. In This is where we come in, in this blog post will we try to see if we are able to classify the troublesome papers.

If you would like to read more about this story including past attempts to solve this problem please read [How Statistics Solved a 175-Year-Old Mystery About Alexander Hamilton](https://priceonomics.com/how-statistics-solved-a-175-year-old-mystery-about/) by Ben Christopher.

**Libraries**

We will start by loading the libraries which includes glmnet that will be used to construct the predictive model later.

library(tidyverse)

library(tidytext)

library(gutenbergr)

library(glmnet)

**Data**

We are lucky today because all of The Federalist Papers happens to be on gutenberg

papers <- gutenberg\_download(1404)

head(papers, n = 10)

## # A tibble: 10 x 2

## gutenberg\_id text

## <int> <chr>

## 1 1404 THE FEDERALIST PAPERS

## 2 1404 ""

## 3 1404 By Alexander Hamilton, John Jay, and James Madison

## 4 1404 ""

## 5 1404 ""

## 6 1404 ""

## 7 1404 ""

## 8 1404 FEDERALIST No. 1

## 9 1404 ""

## 10 1404 General Introduction

For the predictive modeling we are going to do later, I would like to divide each paper up into sentences. This is a rather complicated affair, but I will take a rather ad hoc approach that will be good enough for the purpose of this post. We will do this by collapsing all the lines together and splitting them by ., ! and ?’s.

papers\_sentences <- pull(papers, text) %>%

str\_c(collapse = " ") %>%

str\_split(pattern = "\\.|\\?|\\!") %>%

unlist() %>%

tibble(text = .) %>%

mutate(sentence = row\_number())

We would like to assign each of these sentences to the corresponding article number and author. Thus we will first assign each of the 85 papers to the 3 authors and a group for the papers of interest.

hamilton <- c(1, 6:9, 11:13, 15:17, 21:36, 59:61, 65:85)

madison <- c(10, 14, 18:20, 37:48)

jay <- c(2:5, 64)

unknown <- c(49:58, 62:63)

Next we will simple look for lines that include “FEDERALIST No” as they would indicate the start of a paper and then label them accordingly.

papers\_words <- papers\_sentences %>%

mutate(no = cumsum(str\_detect(text, regex("FEDERALIST No",

ignore\_case = TRUE)))) %>%

unnest\_tokens(word, text) %>%

mutate(author = case\_when(no %in% hamilton ~ "hamilton",

no %in% madison ~ "madison",

no %in% jay ~ "jay",

no %in% unknown ~ "unknown"),

id = paste(no, sentence, sep = "-"))

lets take a quick count before we move on

papers\_words %>%

count(author)

## # A tibble: 4 x 2

## author n

## <chr> <int>

## 1 hamilton 114688

## 2 jay 8540

## 3 madison 45073

## 4 unknown 24471

We see that Jay didn’t post as many articles as the other two gentlemen so we will exclude him from further analysis

papers\_words <- papers\_words %>%

filter(author != "jay")

**Predictive modeling**

To make this predictive model we will use the term-frequency matrix as our input and as the response will be an indicator that Hamilton wrote the paper. For this modeling we will use the glmnet package which fits a generalized linear model via penalized maximum likelihood. It is quite fast and works great with sparse matrix input, hence the term-frequency matrix.

The response is set to the binomial family because of the binary nature of the response (did Hamilton write the sentence).

First we get the term-frequency matrix with the cast\_ family in tidytext.

papers\_dtm <- papers\_words %>%

count(id, word, sort = TRUE) %>%

cast\_dtm(id, word, n)

We will need to define a response variable, which we will do with a simple mutate, along with an indicator for our training set which will be the articles with known authors.

meta <- data.frame(id = dimnames(papers\_dtm)[[1]]) %>%

left\_join(papers\_words[!duplicated(papers\_words$id), ], by = "id") %>%

mutate(y = as.numeric(author == "hamilton"),

train = author != "unknown")

## Warning: Column `id` joining factor and character vector, coercing into

## character vector

We will use cross-validation to obtain the best value of the models tuning parameter. This part takes a couple of minutes.

predictor <- papers\_dtm[meta$train, ] %>% as.matrix()

response <- meta$y[meta$train]

model <- cv.glmnet(predictor, response, family = "binomial", alpha = 0.9)

After running the model, we will add the predicted values to our meta data.frame.

meta <- meta %>%

mutate(pred = predict(model, newx = as.matrix(papers\_dtm), type = "response",

s = model$lambda.1se) %>% as.numeric())

It is now time to visualize the results. First we will look at how the training set have been separated.

meta %>%

filter(train) %>%

ggplot(aes(factor(no), pred)) +

geom\_boxplot(aes(fill = author)) +

theme\_minimal() +

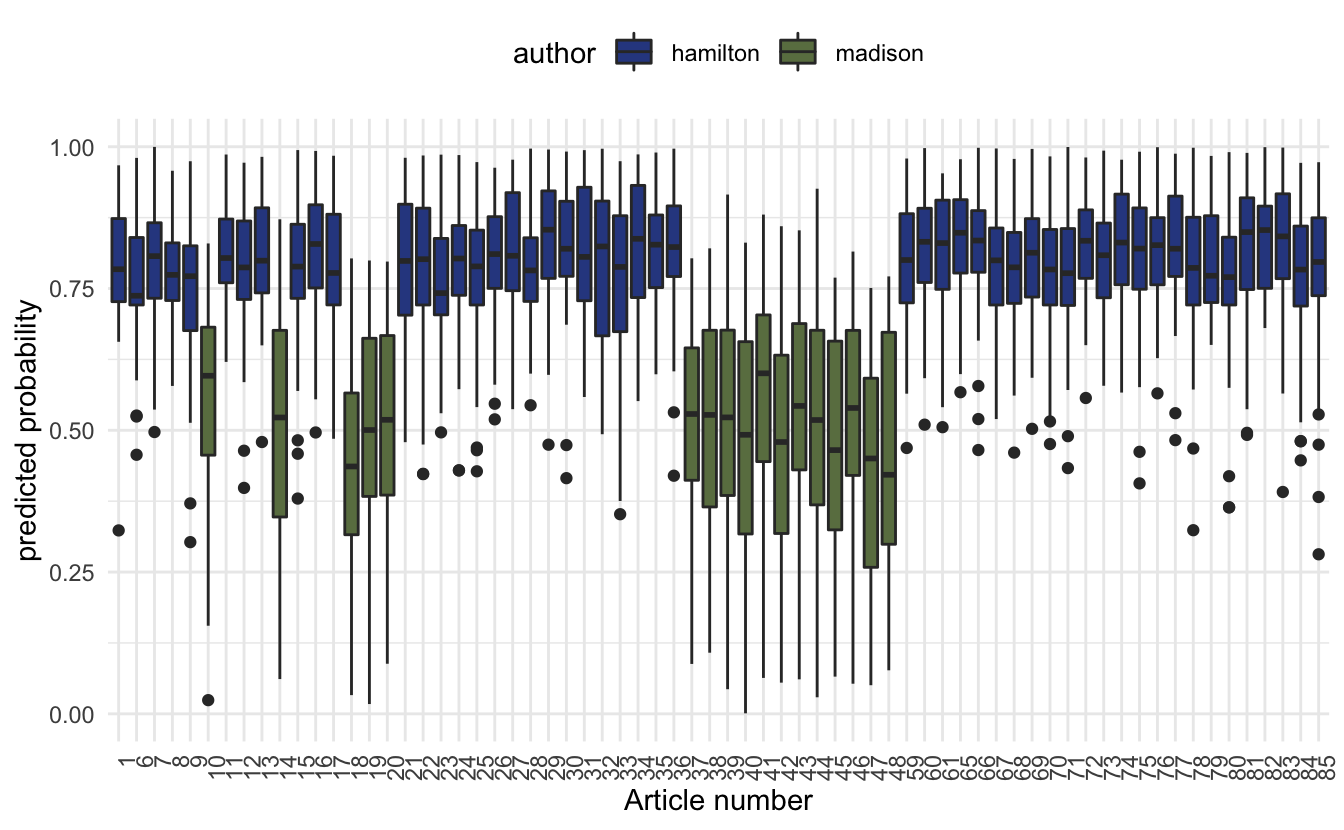
labs(y = "predicted probability",

x = "Article number") +

theme(legend.position = "top") +

scale\_fill\_manual(values = c("#304890", "#6A7E50")) +

theme(axis.text.x = element\_text(angle = 90, hjust = 1))



The box plot of predicted probabilities, one value for each sentence, for the 68 papers by Alexander Hamilton and James Madison. The probability represents the extent to which the model believe the sentence was written by Alexander Hamilton.

Lets see if this model can settle the dispute of the 12 papers. We will plot the predicted probabilities of the unknown papers alongside the training set.

meta %>%

ggplot(aes(factor(no), pred)) +

geom\_boxplot(aes(fill = author)) +

theme\_minimal() +

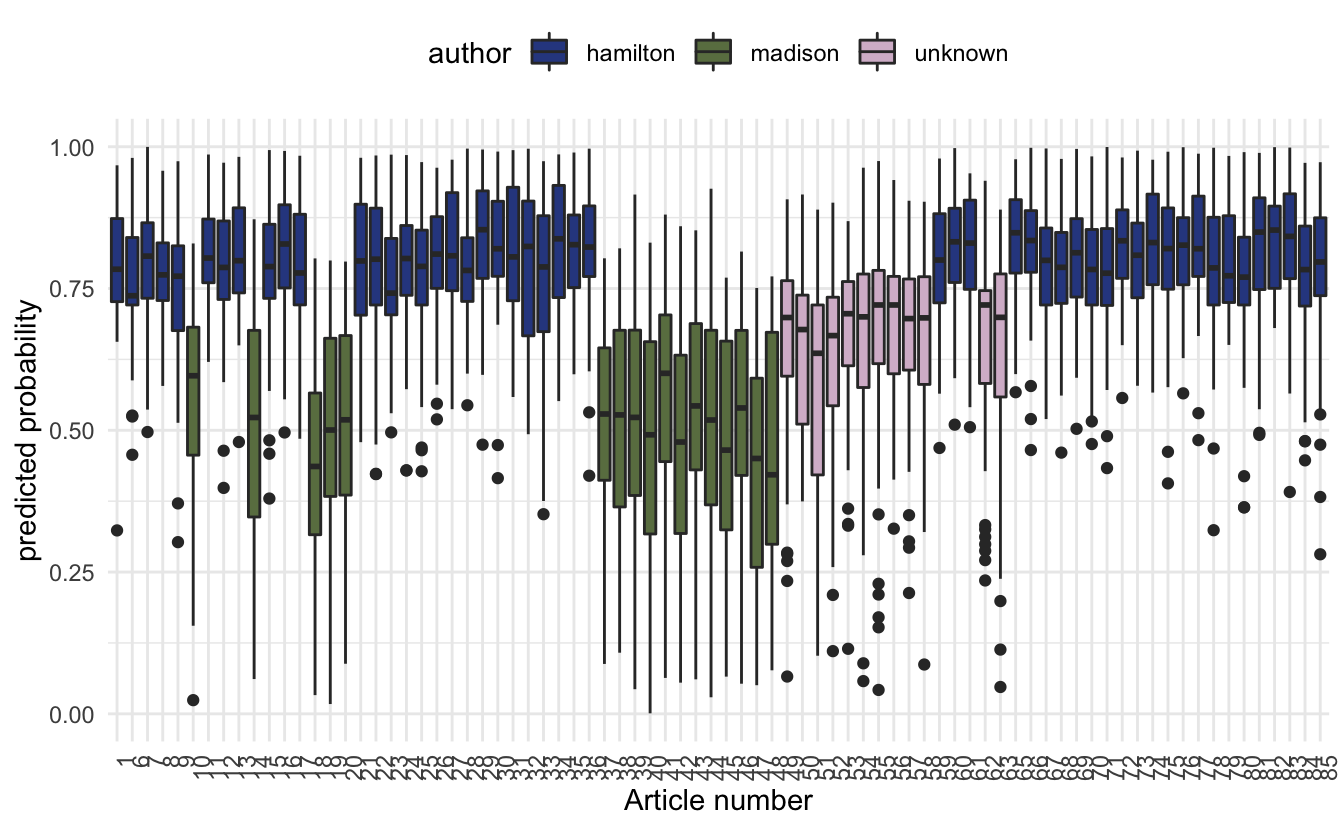
labs(y = "predicted probability",

x = "Article number") +

theme(legend.position = "top") +

scale\_fill\_manual(values = c("#304890", "#6A7E50", "#D6BBD0")) +

theme(axis.text.x = element\_text(angle = 90, hjust = 1))



we notice that the predicted probabilities don’t quite makes up able to determine who the original author is. This can be due to a variety of different reasons. One of them could be that Madison wrote them and Hamilton edited them.

Despite the unsuccessful attempt to predict the secret author we still managed to showcase the method which while being unsuccessful in this case could provide useful in other cases.

**Working showcase**

Since the method proved unsuccessful in determining the secret author did I decide to add an example where the authorship is know. We will use the same data from earlier, only look at known Hamilton and Madison papers, train on some of them and show that the algorithm is able to detect the authorship of the other.

papers\_dtm <- papers\_words %>%

filter(author != "unknown") %>%

count(id, word, sort = TRUE) %>%

cast\_dtm(id, word, n)

here we let the first 16 papers that they wrote be the test set and the rest be training set.

meta <- data.frame(id = dimnames(papers\_dtm)[[1]]) %>%

left\_join(papers\_words[!duplicated(papers\_words$id), ], by = "id") %>%

mutate(y = as.numeric(author == "hamilton"),

train = no > 20)

## Warning: Column `id` joining factor and character vector, coercing into

## character vector

predictor <- papers\_dtm[meta$train, ] %>% as.matrix()

response <- meta$y[meta$train]

model <- cv.glmnet(predictor, response, family = "binomial", alpha = 0.9)

meta <- meta %>%

mutate(pred = predict(model, newx = as.matrix(papers\_dtm), type = "response",

s = model$lambda.1se) %>% as.numeric())

meta %>%

ggplot(aes(factor(no), pred)) +

geom\_boxplot(aes(fill = author)) +

theme\_minimal() +

labs(y = "predicted probability",

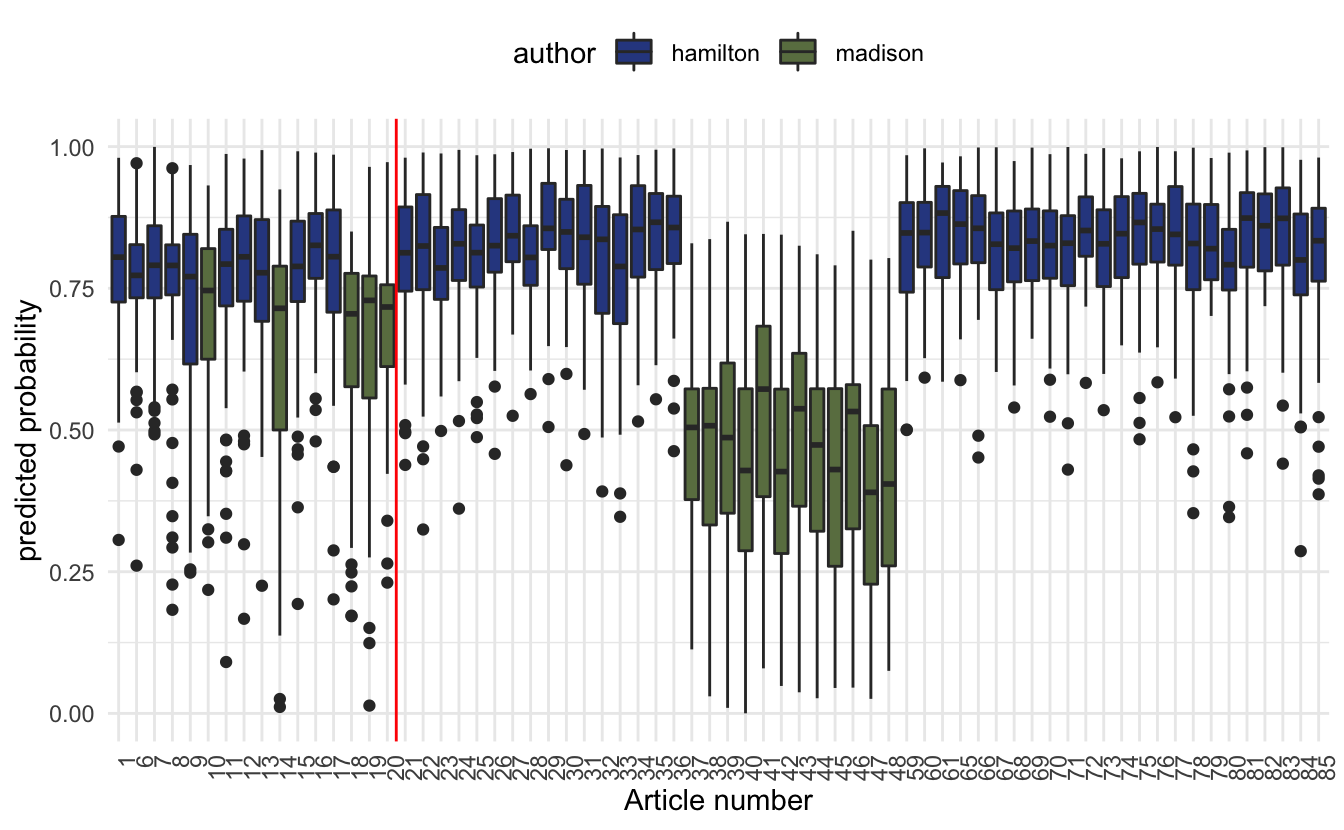
x = "Article number") +

theme(legend.position = "top") +

scale\_fill\_manual(values = c("#304890", "#6A7E50")) +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

geom\_vline(aes(xintercept = 16.5), color = "red")



So we see that while it isn’t as crystal clear what what the test set predictions are giving us, they still give a pretty good indication.