

# Spam filter

Naveen

21/01/2021

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##   filter, lag

## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union
```

```
library(magrittr)
setwd("G:/projects.dataquest/Csv files")
spam <- read_csv("spam.csv")
```

```
##
## -- Column specification -----
## cols(
##   label = col_character(),
##   sms = col_character()
## )
```

```
n <- nrow(spam)
ncol(spam)
```

```
## [1] 2
```

```
dim(spam)
```

```
## [1] 1000    2
```

```
spam_count <- spam %>% filter(label == "spam")
percent_spam <- (150/1000)* 100
percent_nonspam <- (850/1000) * 100
percent_nonspam
```

```
## [1] 85
```

```
percent_spam
```

```
## [1] 15
```

```
set.seed(1)
training_set <- 0.8 * n
cross_valid <- 0.1 * n
test_set <- 0.1 * n

sample_train <- sample(1:n, size = training_set, replace = FALSE)
remaining_indices <- setdiff(1:n, sample_train)
sample_cross <- remaining_indices[1:(length(remaining_indices)/2)]
sample_test <- remaining_indices[(length(remaining_indices)/2)+ 1:length(remaining_indices)]
remaining_indices <- setdiff(1:n, sample_train)

spam_train <- spam[sample_train,]
spam_train
```

```
## # A tibble: 800 x 2
##   label sms
##   <chr> <chr>
## 1 ham   That's fine, have him give me a call if he knows what he wants or has ~
## 2 ham   The Xmas story is peace.. The Xmas msg is love.. The Xmas miracle is j~
## 3 ham   Nah, I'm a perpetual DD
## 4 ham   Well. You know what i mean. Texting
## 5 ham   Same as u... Dun wan... Y u dun like me already ah... Wat u doing now?~
## 6 ham   That's the thing with apes, u can fight to the death to keep something~
## 7 ham   Please da call me any mistake from my side sorry da. Pls da goto docto~
## 8 ham   Hello madam how are you ?
## 9 ham   I am waiting machan. Call me once you free.
## 10 ham  Yes, i'm small kid.. And boost is the secret of my energy..
## # ... with 790 more rows
```

```
spam_cv <- spam[sample_cross,]
spam_test <- spam[sample_test,]

print(mean(spam_train$label == "ham"))
```

```
## [1] 0.85375
```

```
print(mean(spam_cv$label == "ham"))
```

```
## [1] 0.835
```

```
print(mean(spam_test$label == "ham"))
```

```
## [1] 0.85
```

```
library(tidyverse)
```

```
## -- Attaching packages ----- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3      v purrr 0.3.4
## v tibble 3.0.4       v stringr 1.4.0
## v tidyr 1.1.2        v forcats 0.5.0
```

```
## -- Conflicts ----- tidyverse_conflicts() --
```

```
## x tidyr::extract() masks magrittr::extract()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag() masks stats::lag()
## x purrr::set_names() masks magrittr::set_names()
```

```
clean_train <- spam_train %>% mutate(
  sms = str_to_lower(sms) %>% str_squish %>% str_replace_all("[:punct:]", "") %>% str_replace_all("[\\u00a0]", " ") %>% str_replace_all("[:digit:]", "")
clean_train
```

```
## # A tibble: 800 x 2
```

```
##   label sms
```

```
##   <chr> <chr>
```

```
## 1 ham "thats fine have him give me a call if he knows what he wants or has a~
```

```
## 2 ham "the xmas story is peace the xmas msg is love the xmas miracle is jesu~
```

```
## 3 ham "nah im a perpetual dd"
```

```
## 4 ham "well you know what i mean texting"
```

```
## 5 ham "same as u dun wan y u dun like me already ah wat u doing now still ea~
```

```
## 6 ham "thats the thing with apes u can fight to the death to keep something ~
```

```
## 7 ham "please da call me any mistake from my side sorry da pls da goto docto~
```

```
## 8 ham "hello madam how are you "
```

```
## 9 ham "i am waiting machan call me once you free"
```

```
## 10 ham "yes im small kid and boost is the secret of my energy"
```

```
## # ... with 790 more rows
```

```
vocabulary <- NULL
```

```
messages <- clean_train %>% pull(sms)
```

```
for (m in messages) {
```

```
  words <- str_split(m, " ")[[1]]
```

```
  vocabulary <- c(vocabulary, words)
```

```
}
```

```
vocabulary <- vocabulary %>% unique()
```

```
spam_messages <- clean_train %>%
```

```
  filter(label == "spam") %>% pull(sms)
```

```
ham_messages <- clean_train %>% filter(label == "ham") %>% pull(sms)
```

```
spam_vocabulary <- NULL
```

```
for (s in spam_messages) {
```

```

words <- str_split(s, " ")[[1]]
spam_vocabulary <- c(spam_vocabulary, words)

}

ham_vocabulary <- NULL
for (sm in ham_messages) {
  words <- str_split(sm, " ")[[1]]
  ham_vocabulary <- c(ham_vocabulary, words)
}

spam_vocabulary <- spam_vocabulary %>% unique()
ham_vocabulary <- ham_vocabulary %>% unique()

```

```

#counting the numbers of vocab
n_spam <- spam_vocabulary %>% length()
n_ham <- ham_vocabulary %>% length()
n_vocab <- vocabulary %>% length()

#marginal probability of a training message being spam or ham

p_spam <- mean(clean_train$label == "spam")
p_ham <- mean(clean_train$label == "ham")

#creating tibble for counting the spam and ham messages

spam_counts <- tibble(
  word = spam_vocabulary
) %>% mutate(
  spam_count = map_int(word, function(w) {
    map_int(spam_messages, function(s) {
      (str_split(s, " ")[[1]] == w) %>% sum
    }) %>%
    sum
  })
)

ham_counts <- tibble(
  word = ham_vocabulary
) %>% mutate(
  ham_count = map_int(word, function(w) {
    map_int(ham_messages, function(sm) {
      (str_split(sm, " ")[[1]] == w) %>% sum
    }) %>%
    sum
  })
)
ham_counts

```

```
## # A tibble: 3,857 x 2
```

```
##      word  ham_count
##      <chr>      <int>
## 1 thats         38
## 2 fine          12
## 3 have         164
## 4 him           25
## 5 give          24
## 6 me           242
## 7 a             369
## 8 call          111
## 9 if            95
## 10 he           53
## # ... with 3,847 more rows
```

*#joining the tibbles*

```
word_counts <- full_join(spam_counts, ham_counts, by = "word") %>% mutate(
  spam_count = ifelse(is.na(spam_count), 0, spam_count),
  ham_count = ifelse(is.na(ham_count), 0, ham_count)
)
word_counts
```

```
## # A tibble: 4,388 x 3
##      word                spam_count ham_count
##      <chr>                <dbl>      <dbl>
## 1 save                    1          1
## 2 money                    2         17
## 3 on                       48        147
## 4 wedding                  1          0
## 5 lingerie                  1          0
## 6 at                       16        114
## 7 wwwbridalpetticoatdreamscouk 1          0
## 8 choose                    2          6
## 9 from                      30         55
## 10 a                       113        369
## # ... with 4,378 more rows
```

```
classify <- function(message, alpha = 1) {
  clean_message <- str_to_lower(message) %>% str_squish %>%
    str_replace_all("[[:punct:]]", "") %>%
    str_replace_all("[\\u0094\\u0092\\u0096\\n\\t]", "") %>%
    str_replace_all("[[:digit:]]", "")

  words <- str_split(clean_message, " ")[[1]]
  new_words <- setdiff(vocabulary, words)
  new_words_prob <- tibble(
    word = new_words,
    spam_prob = 1,
    ham_prob = 1,
  )

  present_probs <- word_counts %>%
    filter(word %in% words) %>% mutate(
      spam_prob = (spam_count + alpha) / (n_spam + alpha * n_vocab),
```

```
ham_prob = (ham_count + alpha) / (n_ham + alpha * n_vocab)
) %>% bind_rows(new_words_prob) %>%
  pivot_longer(
    cols = c("spam_prob", "ham_prob"),
    names_to = "label",
    values_to = "prob"
  ) %>% group_by(label) %>%
  summarise(
    wi_prob = prod(prob)
  )

p_spam_given_message <- p_spam *(present_probs %>% filter(label == "spam_prob") %>% pull(wi_prob))
p_ham_given_message <- p_ham * (present_probs %>% filter(label == "ham_prob") %>% pull(wi_prob))

# Classify the message based on the probability
ifelse(p_spam_given_message >= p_ham_given_message, "spam", "ham")
}
```

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

```
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```
final_train
```

```
## # A tibble: 800 x 3
##   label sms                                     prediction
##   <chr> <chr>                                     <chr>
## 1 ham   "thats fine have him give me a call if he knows what he wan~ ham
## 2 ham   "the xmas story is peace the xmas msg is love the xmas mira~ ham
## 3 ham   "nah im a perpetual dd"                                     ham
## 4 ham   "well you know what i mean texting"                         ham
## 5 ham   "same as u dun wan y u dun like me already ah wat u doing n~ ham
## 6 ham   "thats the thing with apes u can fight to the death to keep~ ham
## 7 ham   "please da call me any mistake from my side sorry da pls da~ ham
## 8 ham   "hello madam how are you "                                   ham
## 9 ham   "i am waiting machan call me once you free"                 ham
## 10 ham  "yes im small kid and boost is the secret of my energy"     ham
## # ... with 790 more rows
```

```
# Results of classification on training
confusion <- table(final_train$label, final_train$prediction)
accuracy <- (confusion[1,1] + confusion[2,2]) / nrow(final_train)
confusion
```

```
##
##      ham spam
## ham  680   3
## spam  79  38
```

```
accuracy
```

```
## [1] 0.8975
```

```
alpha_grid <- seq(0.05, 1, by = 0.05)
cv_accuracy <- NULL

for (alpha in alpha_grid) {

  cv_probs <- word_counts %>%
```



[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```
# Check out what the best alpha value is
tibble(
  alpha = alpha_grid,
  accuracy = cv_accuracy
)
```

```
## # A tibble: 20 x 2
##   alpha accuracy
##   <dbl>     <dbl>
## 1 0.05     0.93
## 2 0.1      0.93
## 3 0.15     0.93
## 4 0.2      0.92
## 5 0.25     0.915
## 6 0.3      0.91
## 7 0.35     0.905
## 8 0.4      0.9
## 9 0.45     0.9
## 10 0.5     0.895
## 11 0.55     0.89
## 12 0.6     0.89
## 13 0.65     0.89
## 14 0.7     0.875
## 15 0.75     0.87
## 16 0.8     0.865
## 17 0.85     0.865
## 18 0.9     0.865
## 19 0.95     0.86
## 20 1       0.86
```