# Homework 2 - spam classification

PSTAT131-231

## Part 1: spam classification

**Background**: The `spam` data are publicly available on the UCI Machine Learning repository. They comprise 2788 non-spam emails from saved work and personal emails, and 1813 emails filed as spam. Each observation in the dataset corresponds to one email.

The variables in the dataset comprise information about the frequencies of words and characters in the emails, and are as follows.

- 48 word frequency variables named `wf_WORD`, calculated as the percentage of words in the e-mail that match `WORD`.
- 6 character frequency variables named `cf_.` and `cf_1`, ..., `cf_5` for non-alphanumeric characters, calculated as the percentage of characters in the e-mail that match the given character.
- 3 all-capital letter variables: `allcaps_avgrun` gives the average length of strings of consecutive captial letters; `allcaps_maxrun` gives the longest such run; `total_caps` gives the total number of capital letters in the email.
- A factor indicating whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail.

**Objective**: You will build models to classify emails as spam and non-spam. This task has applications in developing spam filters.

```
# spam dataset
load('data/spam.RData')

# center and scale features
spam <- spam %>% mutate(across(-spam, scale))
```

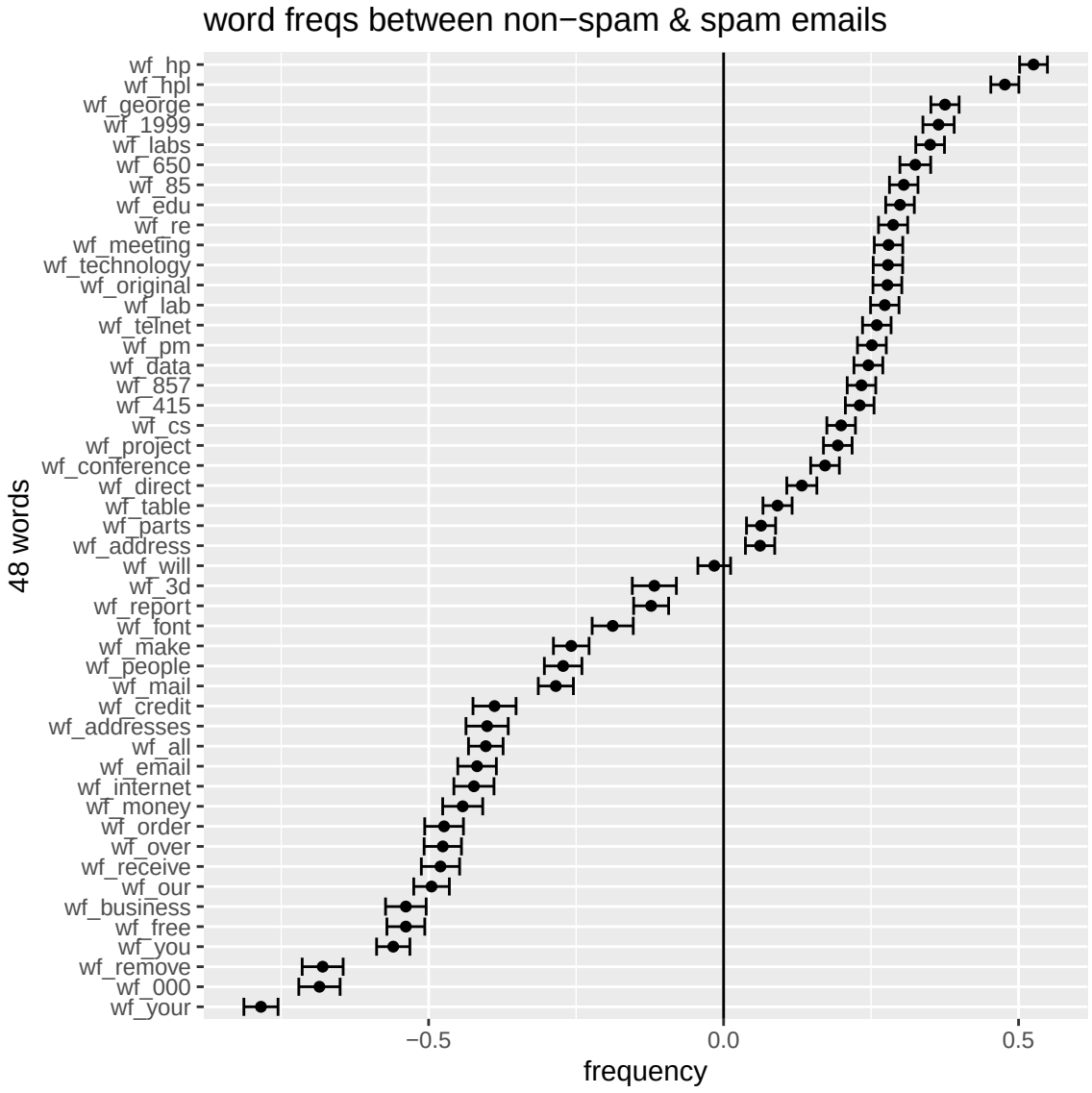
Throughout, you'll compare/select models based on estimates of misclassification error from cross validation. As a preliminary step, create ten random exclusive partitions of the dataset. You'll use the *same* partitions to estimate classification errors for every method to ensure a fair comparison (this avoids confounding variability due to random partitioning with variability due to method).

```
set.seed(11721) # for reproducibility
cv_data <- spam %>% crossv_kfold(k = 6, id = 'fold')
```

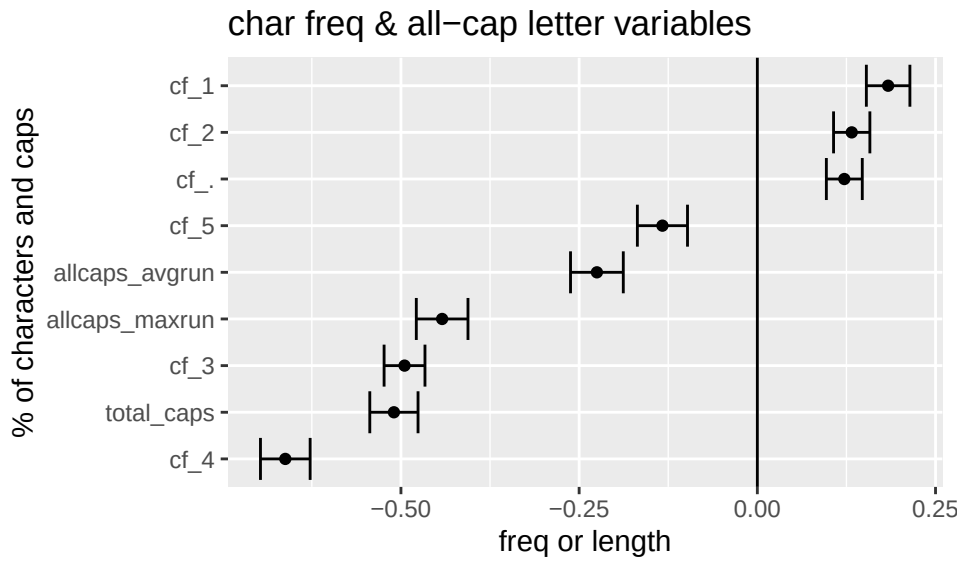
### Q1 Exploratory plots

i) Make a plot of the average difference in word frequencies between spam and non-spam emails for each word variable: plot the average difference on the horizontal axis and the word on the vertical axis.

Order the words on the vertical axis by average frequency difference (see `fct_reorder`). Indicate the vertical line at 0 (`geom_vline`), and add error bars (`geom_errorbarh`). (Hint: first compute the mean and variance of the standardized frequencies for each word by spam classification; then compute the difference and a standard error. You will likely (but not necessarily) follow a select-gather-group-summarize-spread-mutate-plot flow.) Ensure that the axes are appropriately labeled and legible.




ii) Make a similar plot for the character frequency variables and capital-run variables (you can put all 9 on a single axis – no need to make two separate plots, since all variables have been standardized).




iii) Based on the above, do you think that spam email is discernible from non-spam email from these features? Why or why not? Which variable(s) appear to be the strongest indicators of spam/non-spam? Which variables appear to be the weakest indicators?
**Answer:** we think spam email **is** discernible from non-spam email from these features
from the plot we can see there is a mean difference between spam and non-spam emails for certain variables. the standard error for some variables are bigger than others **wf_your**, **wf_000**, **cf_4** appear to be stronger indicators of spam/non-spam
**wf_will** and **wf_address** appear to be the weaker indicators of spam/non-spam

iv) Based on the foregoing, which classification methods do you expect will perform well? Do you think that all available variables should be used for classification? Why or why not?
**Answer: Logistic Regression**
In this case, we have more than 50 predictors. we want to know if a higher/lower frequency of some words or non-alphanumeric_characters or capital_letters means an email is spam or not. the advantage of a logistic regression classifier is that it is easy to interpret. it also works (better) in high dimensions and with noise variables
Therefore, we expect **logistic regression** to perform well
we think **not** all available variables should be used for classification
variables that seem to have near to 0 diff on the plot can possibly superfluous predictors that can cause overfitting or decrease in the performance of our model

**Q2 $k$-nearest neighbors**

Here you'll train a $k$-nearest neighbors classifier and estimate misclassification errors.

i) Train $k$-nearest neighbors classifiers for $k = 1, 5, 10, 15, 20, 25$ on each fold and compute the average (over folds) training and test misclassification error for each $k$. Display the training and test misclassification errors for each $k$.

| k | avg_train_error | avg_test_error |
|---|---|---|
| 1 | 0.0005651 | 0.08954 |
| 5 | 0.06586 | 0.0902 |
| 10 | 0.08081 | 0.09302 |

| k | avg_train_error | avg_test_error |
|---|---|---|
| 15 | 0.08529 | 0.09389 |
| 20 | 0.09033 | 0.1013 |
| 25 | 0.09659 | 0.1017 |

ii) For the best $k$, predict spam classification for each observation by leave-one-out cross validation. Cross-tabulate predicted and true labels and display the result.

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

```
## [1] 0.0823734
```

| | not_spam | spam |
|---|---|---|
| **not_spam** | 2606 | 182 |
| **spam** | 197 | 1616 |

```
##           y_hat_knn
## y            not_spam       spam
##   not_spam 0.93472023 0.06527977
##   spam     0.10865968 0.89134032
```

iii) What does the optimal value of $k$ suggest about separation of the classes in the feature space? Considering this together with the estimated prediction accuracy, make a conjecture about the nature of the relationship between the class labels and word/character frequencies – do you think that spam is discerned primarily by keywords or by word combinations? Does your conjecture coincide with any intuitions you have about the use of written language? What implications does your conjecture have for how features should enter into more structured classifiers?
**Answer:** the optimal value of $k$ is 1. When k = 1, the decision boundary is overly flexible and finds patterns in the data that don't correspond to the Bayes decision boundary. it indicates that there is little to no separation of the classes in the feature space.
predicting class labels simply by word/character frequencies may not produce good accuracy.
We think spam is discerned primarily by word combinations. Our conjecture coincides with our intuitions about the use of written language. A single keyword does not tell us much about the context of a piece of writing, but if we know a short phrase combined with certain characters that appears in this piece, then that gives us some insight
features should enter into more structured classifiers in a way that incorporates word/character combinations

**Q3 Logistic regression**

Here you'll fit a logistic regression model with first-order terms only and estimate the misclassification error using cross validation.

i) Using the same partitioning of the data that you used to select $k$ for the $k$-nearest neighbors classifier, train logistic regression models on each fold and predict the class labels on the test partition using a threshold of $\hat{p} \geq 0.5$. Estimate the training and test misclassification errors for the logistic regression model by averaging the proportion of erroneous classifications on each partition across the folds. How do the estimated misclassification errors compare with those of the KNN classifier for the optimal $k$?

```
## # A tibble: 6 x 3
##   prop_train_errors prop_test_errors  fold
##               <dbl>            <dbl> <int>
## 1            0.0696           0.0639     1
## 2            0.0707           0.0704     2
## 3            0.0608           0.0847     3
## 4            0.0715           0.0717     4
## 5            0.0728           0.0626     5
## 6            0.0678           0.0770     6
```

```
## # A tibble: 1 x 2
##   avg_train_error avg_test_error
##             <dbl>          <dbl>
## 1          0.0689         0.0717
```

the estimated misclassification errors are slightly lower compared with those of the KNN classifier for the optimal $k$ (q2i)

ii) Predict spam classification for each observation in the full dataset by leave-one-out cross validation using the logistic regression model with a probability threshhold of 0.5. Cross-tabulate the predicted and true class labels. How do the error rates of each type compare with the KNN classifier?

```
## [1] 0.0826087
```

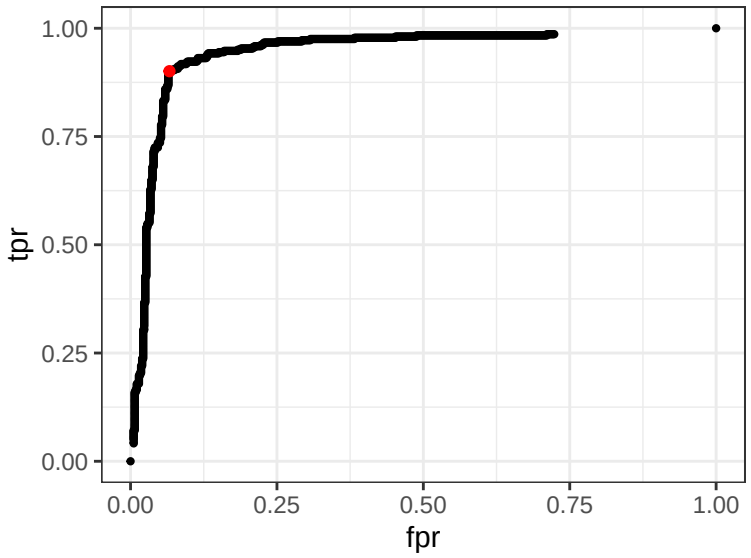
```
##           y_hat_glm
## y          not_spam spam
##   not_spam      521   36
##   spam           40  323
```

```
##           y_hat_glm
## y            not_spam       spam
##   not_spam 0.93536804 0.06463196
##   spam     0.11019284 0.88980716
```

(using 20% of the data) the error rates of each type are similar compared with the KNN classifier (q2ii)

iii) Construct an ROC curve for the logistic regression model and choose an optimal threshold. Repeat the previous part using your optimal threshhold.

```
## # A tibble: 1 x 4
##      fpr   tpr thresh youden
##    <dbl> <dbl>  <dbl>  <dbl>
## 1 0.0664 0.901  0.449  0.834
```




```
## [1] 0.08043478
```

```
##           glm_preds_adj
## y          not_spam spam
##   not_spam      520   37
##   spam           37  326
```

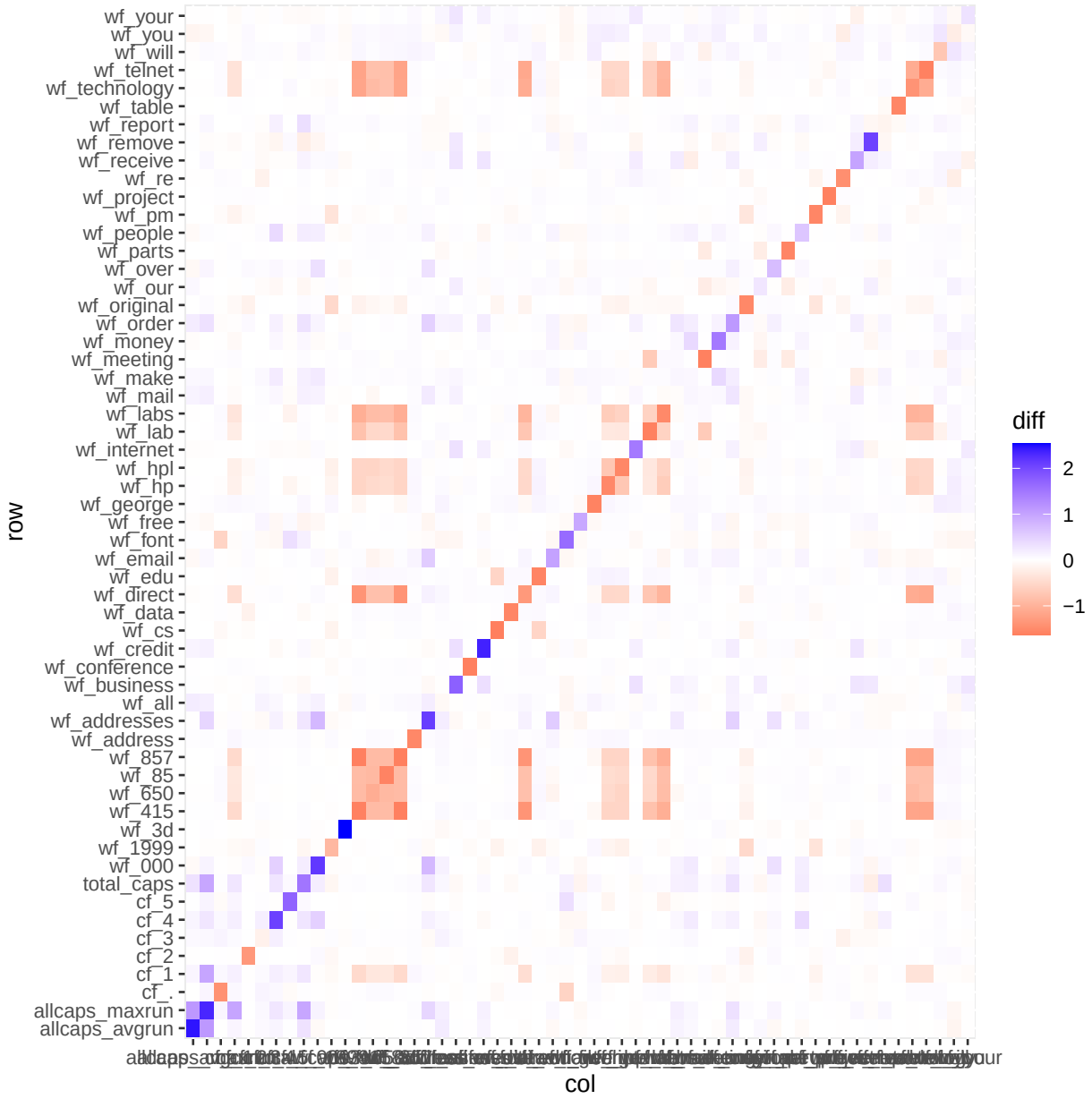
```
##           glm_preds_adj
## y             not_spam       spam
##   not_spam 0.93357271 0.06642729
##   spam     0.10192837 0.89807163
```

the error rates of each type are similar compared with the KNN classifier (q2ii)

iv) Based on your analysis using $k$-nearest neighbors (particularly your answer to the final question from the KNN section), make a general suggestion about what kind of additional model terms might improve the logistic regression classifier. Do you forsee any problems with implementing your suggestion?
**Answer:** Additional model terms like interactions between variables (word combinations) might improve the logistic regression classifier since there are so many variables, implementing my suggestion will be time consuming and can possibly overcomplicate our model. if we are selective about the additional terms, then we would have to bring in some outside presumption.

**Q4 Discriminant analysis**

i) Extract the feature covariance matrices for spam emails and non-spam emails. Construct a heatmap of the difference between the covariance matrices (see `image()` or `geom_raster()`). Do the covariance matrices seem similar? Which discriminant method (linear or quadratic) seems more appropriate?




the covariance matrices seem kind of different
quadratic discriminant method seems more appropriate

ii) Assume that the covariance matrices are similar enough that the LDA assumption is not unreasonable. For each fold, fit an LDA classifier to the training partition and predict class labels on the test partition and compute the training and test misclassification errors. Display these errors for each fold.

Table 3: Table continues below

| sum_train_errors | length_train | sum_test_errors | length_test |
|---|---|---|---|
| 426 | 3834 | 75 | 767 |
| 433 | 3834 | 86 | 767 |

| sum_train_errors | length_train | sum_test_errors | length_test |
|---|---|---|---|
| 409 | 3834 | 82 | 767 |
| 421 | 3834 | 93 | 767 |
| 420 | 3834 | 91 | 767 |
| 427 | 3835 | 87 | 766 |

| avg_train_errors | avg_test_errors | fold |
|---|---|---|
| 0.1111 | 0.09778 | 1 |
| 0.1129 | 0.1121 | 2 |
| 0.1067 | 0.1069 | 3 |
| 0.1098 | 0.1213 | 4 |
| 0.1095 | 0.1186 | 5 |
| 0.1113 | 0.1136 | 6 |

```
## # A tibble: 1 x 2
##   avg_train_error avg_test_error
##             <dbl>          <dbl>
## 1           0.110          0.112
```

**Q5 Method comparison**

Construct a table showing the average training and test misclassification errors across all folds for each method above: $k$-nearest neighbors (with your optimal choice of $k$); logistic regression; and linear discriminant analysis. Which performs best at prediction? Does this conform to your guess in Q1?

```
## # A tibble: 1 x 3
##       k avg_train_error avg_test_error
##   <dbl>           <dbl>          <dbl>
## 1     1        0.000565         0.0895
```

```
## # A tibble: 1 x 2
##   avg_train_error avg_test_error
##             <dbl>          <dbl>
## 1          0.0689         0.0717
```

```
## # A tibble: 1 x 2
##   avg_train_error avg_test_error
##             <dbl>          <dbl>
## 1           0.110          0.112
```

| avg_train_error | avg_test_error | method |
|---|---|---|
| 0.0005651 | 0.08954 | knn_optimal_k |
| 0.06885 | 0.07172 | log_reg_0.5 |
| 0.1102 | 0.1117 | lda_mle |

logistic regression performs best at prediction
this does conform to my guess in Q1

## Codes

```
#setwd('~/pstat131-231/hw/')
library(tidyverse)
library(modelr)
library(pander)
library(class)
library(ISLR)
knitr::opts_chunk$set(echo=F,
                      eval=T,
                      cache=T,
                      results='markup',
                      message=F,
                      warning=F,
                      fig.height=3,
                      fig.width=4,
                      fig.align='center')
# spam dataset
load('data/spam.RData')

# center and scale features
spam <- spam %>% mutate(across(-spam, scale))
set.seed(11721) # for reproducibility
cv_data <- spam %>% crossv_kfold(k = 6, id = 'fold')
# i)
n <- table(spam$spam)
calc <- spam %>%
  #select(contains('wf'), spam) %>%
  gather(key, value = 'freq', -spam) %>%
  group_by(key, spam) %>%
  summarize(across(freq, list(mean = mean, var = var), na.rm = TRUE)) %>%
  pivot_wider(names_from = spam, values_from = c(freq_mean, freq_var)) %>%
  mutate(avg_diff = freq_mean_not_spam - freq_mean_spam,
         stderror = sqrt(freq_var_not_spam/n[1] + freq_var_spam/n[2]))

calc %>% filter(str_detect(key, "wf")) %>%
  ggplot(aes(avg_diff, fct_reorder(key, avg_diff))) +
  geom_point() +
  geom_vline(xintercept = 0,) +
  geom_errorbarh(aes(xmax = avg_diff + stderror, xmin = avg_diff - stderror)) +
  labs(x="frequency", y="48 words", title="word freqs between non-spam & spam emails")
calc %>% filter(str_detect(key, "wf")==FALSE) %>%
  ggplot(aes(avg_diff, fct_reorder(key, avg_diff))) +
  geom_point() +
  geom_vline(xintercept = 0,) +
  geom_errorbarh(aes(xmax = avg_diff + stderror, xmin = avg_diff - stderror)) +
  labs(x="freq or length", y="% of characters and caps", title="char freq & all-cap letter variables")
# function to fit a sequence in k of knn models
knn_fn <- function(train, test, k_seq) {
  train_mx <- as.data.frame(train) %>% select(-spam) %>% as.matrix()
  train_labels <- as.data.frame(train) %>% pull(spam)
  test_mx <- as.data.frame(test) %>% select(-spam) %>% as.matrix()
  test_labels <- as.data.frame(test) %>% pull(spam)
```

```
  out <- tibble(k = k_seq) %>%
    mutate(preds = map(k, ~ knn(train_mx, test_mx, train_labels, .x)),
           fit = map(k, ~ knn(train_mx, train_mx, train_labels, .x)),
           test_labels = map(k, ~ test_labels),
           train_labels = map(k, ~ train_labels),) %>%
    mutate(test_misclass = map2(preds, test_labels,
                                ~ as.numeric(.x) - as.numeric(.y)),
           train_misclass = map2(fit, train_labels,
                                 ~ as.numeric(.x) - as.numeric(.y)) ) %>%
    mutate(test_errors = map(test_misclass, ~ mean(abs(.x))),
           train_errors = map(train_misclass, ~ mean(abs(.x))) )
    return (out)
}
k_values <-  c(1,5,10,15,20,25)
cv_out <- tibble ()
for (i in 1:6) {
  out_i <- knn_fn(cv_data[[1]][[i]], cv_data[[2]][[i]], k_values)
  cv_out <- cv_out %>% bind_rows(out_i)
}

cv_errors <- cv_out %>%
  #unnest(everything())  %>%
  select(k, contains("error")) %>% group_by(k) %>%
  summarize(avg_train_error = mean(unlist(train_errors)),
            avg_test_error = mean(unlist(test_errors)))
cv_errors %>% pander()
# select k
best_k <- cv_errors$k[which.min(abs(cv_errors$avg_test_error))]
best_k

x_mx <- spam %>% select(-spam) %>% as.matrix() # features
y <- spam %>% pull(spam) # response

y_hat_knn <- knn.cv(train = x_mx, cl = y, k = best_k) # leave-one-out CV predictions
mean(y != y_hat_knn)
# cross-tabulate
errors_knn <- table(y, y_hat_knn)
errors_knn %>% pander()
errors_knn/rowSums(errors_knn)
# function to
log_fn <- function(train, test) {
  train_labels <- as.data.frame(train) %>% pull(spam)
  test_labels <- as.data.frame(test) %>% pull(spam)
  # fit logistic regression linear in x1 and x2
  fit_glm <- glm(spam ~ ., family = 'binomial', data = train)

  p_hat_glm <- predict(fit_glm, train, type = 'response') # compute estimated probabilities
  y_hat_train_glm <- factor(p_hat_glm > 0.5, labels = levels(spam$spam)) # bayes classifier

  preds_glm <- predict(fit_glm, test, type = 'response')
  y_hat_test_glm <- factor(preds_glm > 0.5, labels = levels(spam$spam))

  train_out <- tibble(
```

```
    train_labels = train_labels,
    p_hat_glm=p_hat_glm,
    y_hat_train_glm=y_hat_train_glm) %>%
    mutate(train_misclass = map2(y_hat_train_glm, train_labels,
                                 ~ as.numeric(.x) - as.numeric(.y)) ) %>%
    mutate(train_errors = map(train_misclass, ~ mean(abs(.x))) )

  test_out <- tibble(
    test_labels = test_labels,
    preds_glm=preds_glm,
    y_hat_test_glm=y_hat_test_glm) %>%
    mutate(test_misclass = map2(y_hat_test_glm, test_labels,
                                ~ as.numeric(.x) - as.numeric(.y)) ) %>%
    mutate(test_errors = map(test_misclass, ~ mean(abs(.x))) )

  errors <- tibble(prop_train_errors=sum(unlist(train_out$train_errors))/length(train_out$train_errors)
                   prop_test_errors=sum(unlist(test_out$test_errors))/length(test_out$test_errors))
  return (errors)
}

log_out <- tibble ()
for (i in 1:6) {
  out_i <- log_fn(cv_data[[1]][[i]], cv_data[[2]][[i]])
  log_out <- log_out %>% bind_rows(out_i)
}
log_out <- log_out %>% bind_cols(fold=1:6)
log_out
errors_log <- log_out %>%
              summarize(avg_train_error= mean(prop_train_errors),
                        avg_test_error= mean(prop_test_errors) )
errors_log
subset_spam <- slice_sample(spam, prop = 0.2) #nectir

y <- subset_spam %>% pull(spam)
preds_glm <- integer()

for (i in 1:(nrow(subset_spam)) ) {
  one_left <- subset_spam %>% slice(i)
  tmp_spam <- subset_spam %>% slice(-i)
  fit_glm <- glm(spam ~ ., family = 'binomial', data = tmp_spam)
  one_preds_glm <- predict(fit_glm, one_left, type = 'response')
  preds_glm[i] <- one_preds_glm
}
#preds_glm
y_hat_glm <- factor(preds_glm > 0.5, labels = levels(y))
mean(y != y_hat_glm)
error_glm <- table(y, y_hat_glm)
error_glm
error_glm/rowSums(error_glm)
library(ROCR)

prediction_glm <- prediction(predictions = preds_glm, labels = y) # create prediction object
perf_glm <- performance(prediction.obj = prediction_glm, 'tpr', 'fpr') # compute error rates as a funct
```

```
rates_glm <- tibble(fpr = perf_glm@x.values,
                    tpr = perf_glm@y.values,
                    thresh = perf_glm@alpha.values)  # extract rates and threshold from `perf_lda` as a

rates_glm <- rates_glm %>%
  unnest(everything()) %>%
  mutate(youden = (tpr-fpr)) # compute youden's statistic

optimal_thresh <- rates_glm %>% slice_max(youden)
optimal_thresh
# roc curve
rates_glm %>%
  ggplot(aes(x = fpr, y = tpr)) +
  geom_point(size=0.8) +
  geom_point(data=optimal_thresh, color='red') +
  theme_bw()
# recalibrate qda with different probability threshold
glm_preds_adj <- factor(preds_glm > optimal_thresh$thresh, labels = levels(y))
mean(y != glm_preds_adj)
errors_glm_adj <- table(y, glm_preds_adj)
errors_glm_adj
errors_glm_adj/rowSums(errors_glm_adj)
# i) Extract the feature covariance matrices
cov_yes <- spam %>% filter(spam=="spam") %>% select(where(is.numeric)) %>% cov()
cov_not <- spam %>% filter(spam!="spam") %>% select(where(is.numeric)) %>% cov()

# heatmap of differences
diffs <- as_tibble(cov_yes-cov_not)
diffs <- diffs %>% bind_cols(row = names(diffs))
diffs <- gather(diffs, key = 'col', value = 'diff', -58)
diffs %>% ggplot(aes(y = row, x = col), color = "grey50") +
  geom_raster(aes(fill=diff)) +
  scale_fill_gradient2(low="red", high="blue", mid = "white", midpoint = 0)
#library(MASS)
lda_fn <- function(train, test) {
  train_labels <- as.data.frame(train) %>% pull(spam)
  test_labels <- as.data.frame(test) %>% pull(spam)
  lda_fit <- MASS::lda(spam ~ ., method = 'mle', data = train)
  lda_preds_train <- predict(lda_fit, train)
  lda_preds <- predict(lda_fit, test)

  train_out <- tibble(lda_preds_train=lda_preds_train$class,
                      train_labels = train_labels) %>%
        mutate(train_misclass = map2(lda_preds_train, train_labels,
                                ~ as.numeric(.x) - as.numeric(.y)) ) %>%
        mutate(train_errors = map(train_misclass, ~ mean(abs(.x))) )
  test_out <- tibble(lda_preds=lda_preds$class,
                      test_labels = test_labels) %>%
    mutate(test_misclass = map2(lda_preds, test_labels,
                                ~ as.numeric(.x) - as.numeric(.y)) ) %>%
    mutate(test_errors = map(test_misclass, ~ mean(abs(.x))) )
  errors <- tibble(sum_train_errors=sum(unlist(train_out$train_errors)),
                   length_train=length(train_labels),
```

```
                  sum_test_errors=sum(unlist(test_out$test_errors)),
                  length_test=length(test_labels) ) %>%
    mutate(avg_train_errors = sum_train_errors/length_train,
           avg_test_errors = sum_test_errors/length_test)
  return (errors)
}

lda_out <- tibble ()
for (i in 1:6) {
  error_foldi <- lda_fn(cv_data[[1]][[i]], cv_data[[2]][[i]])
  lda_out <- lda_out %>% bind_rows(error_foldi)
}
lda_out <- lda_out %>% bind_cols(fold=1:6)
lda_out %>% pander()# Display these errors for each fold.

errors_lda <- lda_out %>%
              summarize(avg_train_error= mean(avg_train_errors),
                        avg_test_error= mean(avg_test_errors) )
errors_lda
best_k_errors <- cv_errors %>% filter(k==best_k)
best_k_errors
errors_log
errors_lda

errors_all <- bind_rows(best_k_errors[-1],errors_log,errors_lda )
errors_all <- errors_all %>% bind_cols(method=c("knn_optimal_k", "log_reg_0.5", "lda_mle"))
errors_all %>% pander()
```