Classification

Tony Ni

6/24/2020

Libraries

```
library(class) # for kNN methods
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.2
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'ggplot2' was built under R version 3.6.1
```

```
## Warning: package 'tibble' was built under R version 3.6.1
```

```
## Warning: package 'tidyr' was built under R version 3.6.2
```

```
## Warning: package 'purrr' was built under R version 3.6.1
```

```
## Warning: package 'forcats' was built under R version 3.6.1
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Reading in Data

```
setwd("~/harvard-summer-biostats")
df <- read_csv("data/long_illinois.csv") #read in data
```

```
## Parsed with column specification:
## cols(
##   well_id = col_character(),
##   site = col_character(),
##   disposal_area = col_character(),
##   type = col_character(),
##   gradient = col_character(),
##   contaminant = col_character(),
##   concentration = col_double()
## )
```

```
df1 <- na.omit(df) #get rid of na's
```

Introduction

Classification techniques allow you to use explanatory variables to make "rules" which classify observations according to categories of some categorical response variable (it could be quantitative that you made into categorical). In our case, the goal of a classification analysis would be to use the variables to accurately classify the wells as being dangerous or safe. The various classification methods make decision rules to accomplish this.

Since we don't have a variable labelling a well as being dangerous or safe, we have to make a LARGE assumption and assume that wells with contaminant concentration values over the threshold is considered to be dangerous.

We will pull out the observatiosn with concentrations past the threshold and place them within a new dataset.

More Wrangling

```
#creating vector of contaminants
contaminant <- unique(df1$contaminant)

#creating vector of threshold values for each contaminant
threshold <- c(6/1000, 10/1000, 2, 4/1000, 3, 5/1000, NA, NA, 100/1000,
               6/1000, NA, 15/1000, 40/1000, 2/1000, 40/1000, NA, 5,
               50/1000, 500, 2/1000, NA)

contam_t <- cbind(contaminant, threshold) %>%
  na.omit()

#creating function to obtain all observations with values above threshold
getOverThreshold <- function(df){
  datalist = list()
  for(i in 1:nrow(contam_t)){ #for each contaminant i
    df1 <- filter(df, gradient == "Upgradient")
    df2 <- filter(df1, contaminant == contam_t[i])
    data <- filter(df2, concentration > contam_t[nrow(contam_t) + i])
    datalist[[i]] <- data
  }
  toReturn <- do.call(rbind, datalist)

  return(toReturn)
```

```
}



overthreshold_df <- getOverThreshold(df1) #df with all observations over threshold value

df2 <- overthreshold_df %>% #all of the well are over the threshold
  mutate(status = "dangerous")

df3 <- full_join(df1, df2) %>% #join together
    mutate(status = if_else(is.na(status), "safe", status)) #else, under
```

```
## Joining, by = c("well_id", "site", "disposal_area", "type", "gradient", "contaminant", "concentratio
```

```
df4 <- df3 %>% #collapse empty rows
  spread(contaminant, concentration) %>%
  group_by(well_id) %>%
  summarise_each(funs(first(.[!is.na(.)]))) %>%
  filter(gradient == "Upgradient") %>%
  na.omit()
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

There are many approaches to classification that we could take, for example: classification trees, random forests, linear discrminant analysis (LDA), but we decided to go with nearest neighbors classifiers because it is intuitive to understand and is a non-parametric method which allows us to forgo the assumptions that we might have needed to check for the other methods.

NN Classifiers

Using a NN classifier, we mainly want to classify wells based on their "nearest neighbors". For each of our wells, we determine its' r nearest neighbor (by Euclidean distance) and then classify a well according to the majority class of its r NN. For example, if $r = 2$ and the 5 closest neighbors for a well are the classes 3,3,2,5,3 – then we would assign that well to be class 3.

We have to pick the number of NN in advance (which can be done with CV), and there are issues that can arise if some classes are small and if r is too large. The small classes will get "swamped out" meaning it is possible that you may never identify an observation from a small class

```
knitr::kable(table(df4$status), col.names = c("Status", "Frequency"))
```

Status	Frequency
dangerous	44
safe	6

We have 44 wells that are considered dangerous and 6 which are safe. There are DEFINTELY not enough wells in our safe group, but we continued on to see how things would turn out still.

We split our data into a 80% training set, and 20% testing set.

```
set.seed(7271999)

##Generate a random number that is 80% of the total number of rows in dataset.
 ran <- sample(1:nrow(df4), 0.8 * nrow(df4))

##extract training set
 df4_train <- df4[ran,]

##extract testing set
 df4_test <- df4[-ran,]
```

```
well.knn <- knn(df4_train[, c(7:21, 23:26)], df4_test[, c(7:21, 23:26)], df4_train$status, k = 5, prob =
```

In our knn model, we threw out pH and Total Dissolved Solids but used all other contaminants. We used many different values of k and eventually decided on a value of k = 5. Obviously since the majority of the wells in our dataset were said to be dangerous (44) compared to safe (6), we would expect that model won't be as good as classifying safe wells compared to dangerous wells.

```
knitr::kable(table(df4_test$status, well.knn))
```

	dangerous	safe
dangerous	8	0
safe	1	1

We can see from our table that our model correctly classified 8/8 dangerous wells and 1/2 safe wells – thus sadly, what we expected. Some issues with our approach obviously is that fact that we assumed that if a concentration is over the threshold – it is dangerous – when in reality some chemicals may be nontoxic compared to others.