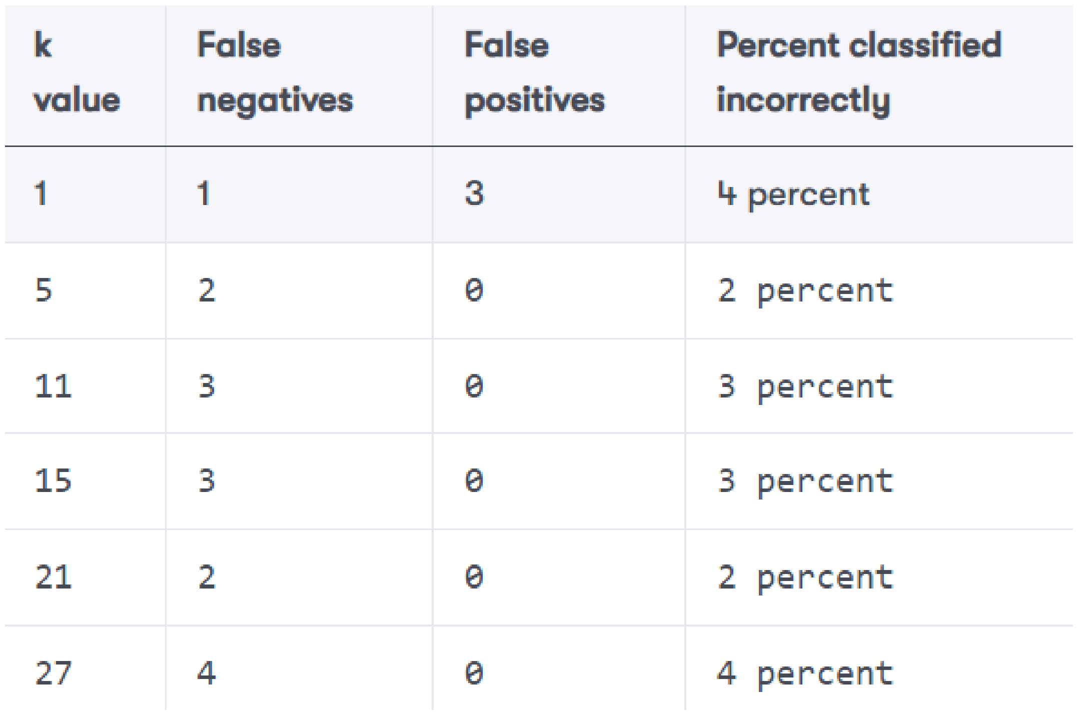
ml

March 7, 2023

Classifying with different `k` values

- Use different values of `k`, check and interpret the predictions: k = 1, 5, 11, 15, 21, 27.



k value	False negatives	False positives	Percent classified incorrectly
1	1	3	4 percent
5	2	0	2 percent
11	3	0	3 percent
15	3	0	3 percent
21	2	0	2 percent
27	4	0	4 percent

Figure 1: Sample results - kNN on wbcd for different k

- Check for different `k` by looping over a function `check_k`:

1. restart the dataset `wbcd`.
2. `normalize` it to obtain `wbcd_n`.

   ```
   normalize <- function(x) {
     return ((x-min(x))/(max(x)-min(x)))
   }
   wbcd_n <- as.data.frame(lapply(wbcd[2:31],FUN=normalize))
   ```

3. split `wbcd_n` in training and test data:

   ```
   wbcd_train <- wbcd_n[1:469,]   # all normalized columns
   wbcd_test <- wbcd_n[470:569,]  # all normalized columns
   ```

4. create training and testing labels

   ```
   wbcd_train_labels <- wbcd[1:469,1]  # from the original dataset
   wbcd_test_labels <- wbcd[470:569,1]  # from the original dataset
   ```

5. define function with `k` as argument

   ```
   library(class)
   wbcd_check_k <- function(x=1) {
     wbcd_test_pred <- knn(train = wbcd_train, # training data
                           test = wbcd_test,  # test data
                           cl = wbcd_train_labels, # class factor
                           k = x)  # nearest neighbors
     print(table(x = wbcd_test_labels,
                 y = wbcd_test_pred))
     mean(wbcd_test_pred==wbcd_test_labels)
   }
   ```

6. loop over the `k` values 1,5,11,15,21,27:

   ```
   ## get the Wisconsin breast cancer data as data frame:
   wbcd <- read.csv(file="http://bit.ly/3khqmkp")
   ## drop the first (ID) column:
   wbcd <- wbcd[-1]
   ## recode target class as labeled 2-level factor
   wbcd$diagnosis |> factor(c("B","M"),c("Benign","Malignant")) -> wbcd$diagnosi
   # wbcd$diagnosis |>  str()
   normalize <- function(x) {
     return ((x-min(x))/(max(x)-min(x)))
   }
   wbcd_n <- as.data.frame(lapply(wbcd[2:31],FUN=normalize))
   ```

```
wbcd_train <- wbcd_n[1:469,]   # all normalized columns
wbcd_test <- wbcd_n[470:569,]  # all normalized columns
wbcd_train_labels <- wbcd[1:469,1]  # from the original dataset
wbcd_test_labels <- wbcd[470:569,1]  # from the original dataset
library(class)
wbcd_check_k <- function(x=1) {
  wbcd_test_pred <- knn(train = wbcd_train, # training data
                      test = wbcd_test,  # test data
                      cl = wbcd_train_labels, # class factor
                      k = x)  # nearest neighbors
  print(table(x = wbcd_test_labels,
              y = wbcd_test_pred))
  mean(wbcd_test_pred==wbcd_test_labels)
}
k_vals <- c(1,5,11,15,21,27)
result <- rep(NA,length(k_vals))
names(result) <- k_vals
j <- 0
for (i in k_vals) {
  j <- j + 1
  result[j] <- wbcd_check_k(i)
  print(paste("Accuracy for k =",i,":",
              format(result[j],digits=6)))
}
## Find best result
## cat("Best result:", format(max(result),digits=3),
##     "for k =", which(result==max(result)),"\n")
```

```
   y
x           Benign Malignant
  Benign        58         3
  Malignant      1        38
[1] "Accuracy for k = 1 : 0.96"
   y
x           Benign Malignant
  Benign        61         0
  Malignant      2        37
[1] "Accuracy for k = 5 : 0.98"
   y
x           Benign Malignant
```

```
  Benign        61         0
  Malignant      3        36
[1] "Accuracy for k = 11 : 0.97"
   y
x           Benign Malignant
  Benign        61         0
  Malignant      3        36
[1] "Accuracy for k = 15 : 0.97"
   y
x           Benign Malignant
  Benign        61         0
  Malignant      2        37
[1] "Accuracy for k = 21 : 0.98"
   y
x           Benign Malignant
  Benign        61         0
  Malignant      4        35
[1] "Accuracy for k = 27 : 0.96"
```

- The Resulting table

k value	False negatives	False positives	% correct
1	1	3	96
5	2	0	98
11	3	0	97
15	3	0	97
21	2	0	98
27	4	0	96

- It is unwise to tailor the approach too closely to the test data: the difference of 100 patient records is not very large compared to the likely test data sets in production.
- To be certain that a learner will generalize to future data, you can create several sets of 100 patients at random and repeatedly retest.

Fixing the algorithm

- Problem in **??**: updating the result vector `r` inside so that the returned vector does not have `k[length(k)] = 7` values but only three:
  - loopindex `i` assumes values `1,5,7`

  - calls `f(i)` and returns a single value
  - I want to put the value into `r` but not into `r[i]` (which is `NA` after the first loop, because `length(r)` is 3.
  - Create dummy index `j` and increase it

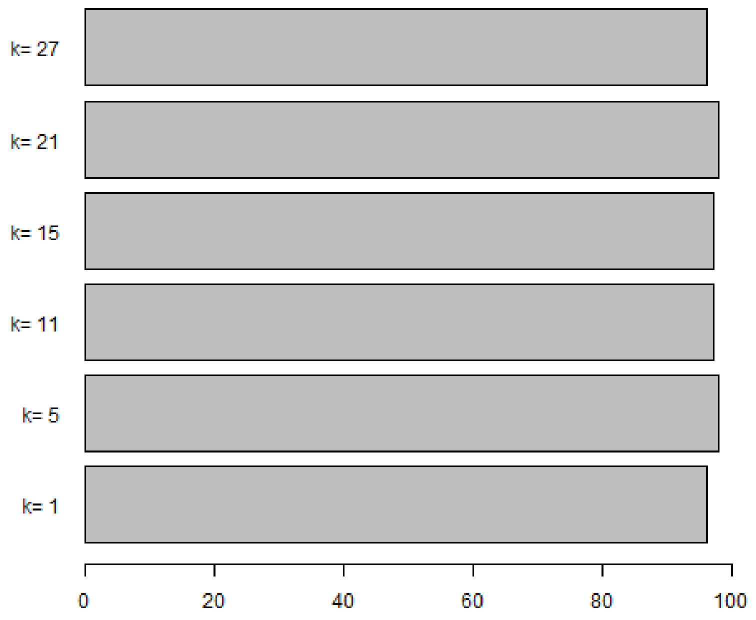
```
f <- function(x=1) {
  return (mean(rnorm(x)))
}
                                        #k <- c(1,5,7)
cat("k:",k,"\n")
r <- rep(NA,length(k))
                                        #cat("r:",r,"\n")
j <- 0
for (i in k) {  # loop runs three times
  j <- j + 1
  r[j] <- f(i)  # assign value of k to r
  print(i)   # print k
  print(r)   # print r
}
r

Error in cat("k:", k, "\n") : object 'k' not found
Error: object 'k' not found
Error in k : object 'k' not found
Error: object 'r' not found
```

Plot results vector (barplot or histogram)

- Plot the `results` vector. Not a very interesting plot.

```
names
barplot(result*100,horiz=TRUE,
        names.arg = paste("k=",k_vals),
        las=1, xlim=c(0,100))
```



Plot cross table results

- Use `capture.output` to stop `gmodels::CrossTable` from printing to the screen, but save the table in `cross_table`, a list.
- Use `table` instead for cleaner, shorter output:

```
cf <- function(x=1) {
  wbcd_test_pred <- knn(train = wbcd_train, # training data
                        test = wbcd_test,  # test data
                        cl = wbcd_train_labels, # class factor
                        k = x)  # nearest neighbors
  cross_table <- table(x = wbcd_test_labels,
                       y = wbcd_test_pred)
  print(cross_table)
  ##      write(cross_table, file='./data/crosstable.txt')
  mean(wbcd_test_pred==wbcd_test_labels)
}
cf(21)
```

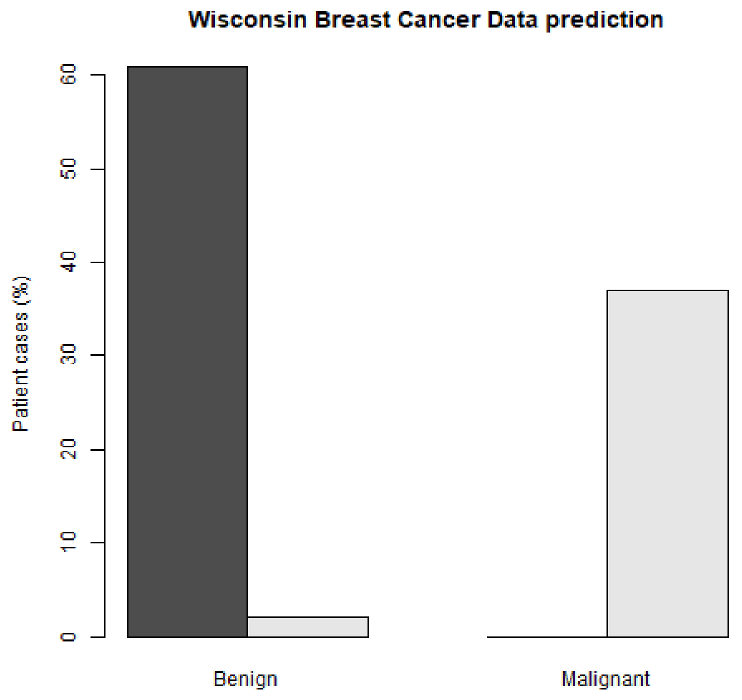
```
           y
x            Benign Malignant
  Benign         61         0
  Malignant       2        37
[1] 0.98
```

- Without function, to test the plotting:

```
wbcd_test_pred <- knn(train = wbcd_train, # training data
                      test = wbcd_test,  # test data
                      cl = wbcd_train_labels, # class factor
                      k = 21)  # nearest neighbors
cross_table <- table(x = wbcd_test_labels,
                     y = wbcd_test_pred)
##      write(cross_table, file='./data/crosstable.txt')
mean(wbcd_test_pred==wbcd_test_labels)
cross_table
```

```
[1] 0.98
           y
x            Benign Malignant
  Benign         61         0
  Malignant       2        37
```

```
barplot(cross_table,
        beside=TRUE,
        ylab="Patient cases (%)",
        main="Wisconsin Breast Cancer Data prediction")
```



- ChatGPT

```
library(gmodels)

my_function <- function(x, y) {
  ## Create a table
  mytable <- table(x, y)
  ## Suppress output of CrossTable, save the result to an R object
  result <- capture.output(gmodels::CrossTable(x = x, y = y, prop.chisq = FALSE))
  ## Return the result
  return(result)
}
## Call the function
x <- c(1, 1, 2, 3, 3, 3)
y <- c("A", "A", "B", "B", "C", "C")
result <- my_function(x, y)
## View the result
```