knn___k_nearest_neighbours.R

win10

2021-05-14

```
# ##########################################################################
rm(list = ls())
options(digits = 5)
# if (!is.null(dev.list())){dev.off()}
# #########################################################################

library(class) # pkg includes knn knearest neighbours
library(caret) # pkg for Classification And REgression Training
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
# read file. Use credit_card_data.txt
my_data = read.delim(file.choose())

# create function for scaling
scale_func = function(a_var) {
  (a_var - min(a_var)) / (max(a_var) - min(a_var))
}

# use lapply to apply function to data
# cast the output of lapply to a data frame
my_data_scaled = as.data.frame(lapply(my_data[, 1:11], scale_func))

# to ensure repeatable results despite random selection
set.seed(123)

# determine test:train split ratio.
split_ratio = 0.7 # e.g. train:test = 0.7 : (1-0.7)

#split data into test and train
random_sampling = sample(
  1:nrow(my_data_scaled),
  size = nrow(my_data_scaled) * split_ratio,
  replace = FALSE
) #randomly select data.

# capture training and testing, predictors/factors/features
train_data = my_data_scaled[random_sampling, ]
test_data = my_data_scaled[-random_sampling, ]

# capture training and testing, responses
train_results = train_data[, 11]
```

```
test_results = test_data[, 11]
# capturing "known" responses is essential for knn; this is the "supervised" part
# because knn is a classifier OR supervised type of machine learning

rows_qty = NROW(train_results)

# to determine the best "k"; use sqrt(nrows)
print( c( "sqrt(nrows)= ", round( sqrt( rows_qty),2)))
```

```
## [1] "sqrt(nrows)= " "21.38"
```

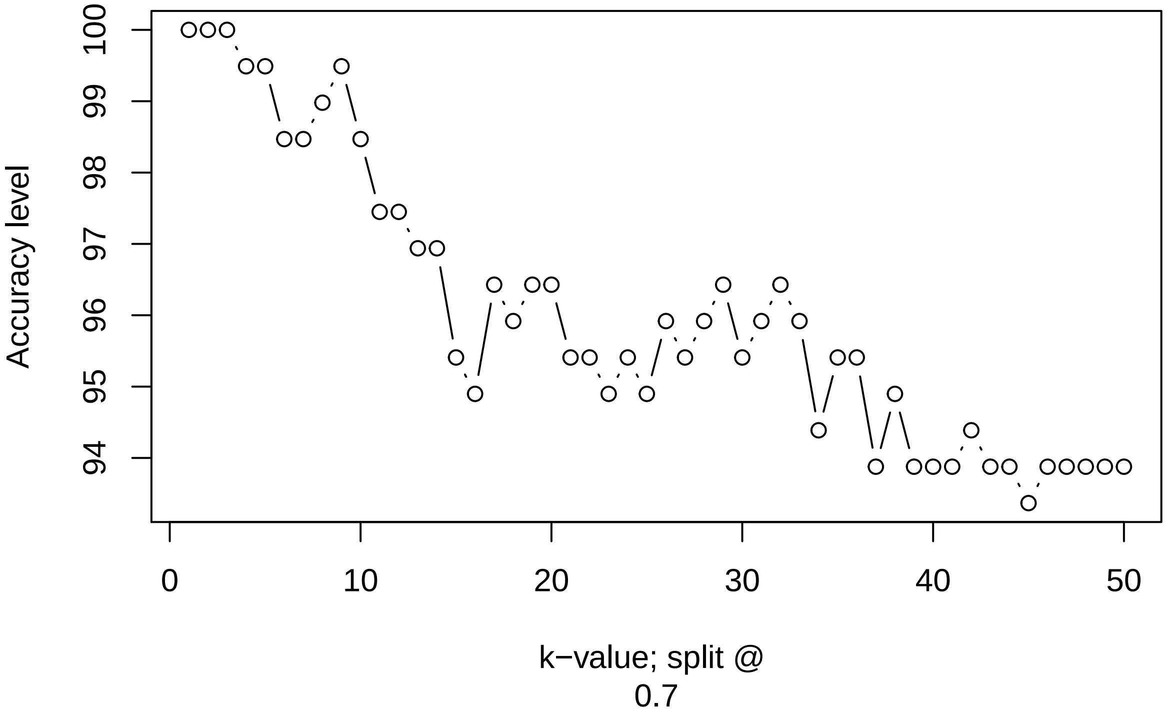
```
# since sqrt is fraction, use one whole number below and above
knn_21 = knn(
  train = train_data,
  test = test_data,
  cl = train_results,
  k = 21 # k-nearest neighbors
)

knn_22 = knn(
  train = train_data,
  test = test_data,
  cl = train_results,
  k = 22 # k-nearest neighbors
)

accu_knn_21 = 100 * sum(test_results == knn_21) / NROW(test_results)
accu_knn_22 = 100 * sum(test_results == knn_22) / NROW(test_results)

i = 1
k_best = 1 # k-nearest neighbors
for (i in 1:50)
{
  knn_mod = knn(
    train = train_data,
    test = test_data,
    cl = train_results,
    k = i
  )
  k_best[i] = 100 * sum(test_results == knn_mod) / NROW(test_results)
  k = i
}

plot(
  k_best,
  type = "b",
  xlab = c("k-value; split @ ", split_ratio),
  ylab = "Accuracy level"
)
```


Accuracy level
100
99
98
97
96
95
94
0
10
20
30
40
50
k−value; split @
0.7