ISYE6501 Homework 2

1/18/2019

Question 3.1

Find a good classifier using ksvm or kknn, include cross-validation and splitting the data sets.

```
# another student suggested this library to split the data, so let's try it
library(caTools)
library(kknn)
rm(list = ls())
set.seed(42) # the answer of life
ccdata <- read.delim("~/Documents/R/GeorgiaTech/Validation/credit_card_data-headers.txt")

# preview data
head(ccdata)
```

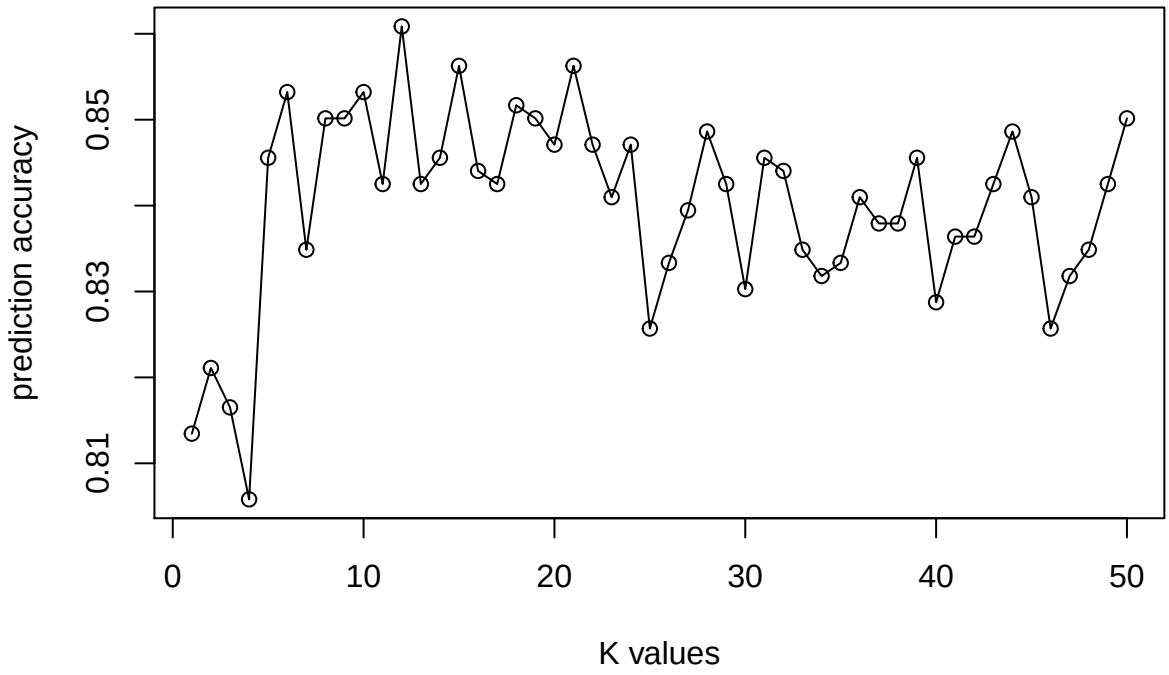
```
##   A1    A2    A3   A8 A9 A10 A11 A12 A14 A15 R1
## 1  1 30.83 0.000 1.25  1   0   1   1 202   0  1
## 2  0 58.67 4.460 3.04  1   0   6   1  43 560  1
## 3  0 24.50 0.500 1.50  1   1   0   1 280 824  1
## 4  1 27.83 1.540 3.75  1   0   5   0 100   3  1
## 5  1 20.17 5.625 1.71  1   1   0   1 120   0  1
## 6  1 32.08 4.000 2.50  1   1   0   0 360   0  1
```

```
# will be used next in a for loop
max_k <- 50

# vector to save predictions
accuracy <- c()
# now calculate the accuracy for kmax neighbors
for (neighbor in 1:max_k) {
    # this package includes cross validation
    model <- cv.kknn(R1~.,ccdata, kcv=10, # use the common k-folds (Dr. Sokol suggested it)
            k=neighbor,
            scale=TRUE)
    # round it to make it comparable
    pred <- round(model[[1]][,2])
    accuracy[neighbor] <- sum(pred == ccdata$R1)/nrow(ccdata)
}
# show accuracies for each k
print (accuracy)
```

```
##  [1] 0.8134557 0.8211009 0.8165138 0.8058104 0.8455657 0.8532110 0.8348624
##  [8] 0.8501529 0.8501529 0.8532110 0.8425076 0.8608563 0.8425076 0.8455657
## [15] 0.8562691 0.8440367 0.8425076 0.8516820 0.8501529 0.8470948 0.8562691
## [22] 0.8470948 0.8409786 0.8470948 0.8256881 0.8333333 0.8394495 0.8486239
## [29] 0.8425076 0.8302752 0.8455657 0.8440367 0.8348624 0.8318043 0.8333333
## [36] 0.8409786 0.8379205 0.8379205 0.8455657 0.8287462 0.8363914 0.8363914
## [43] 0.8425076 0.8486239 0.8409786 0.8256881 0.8318043 0.8348624 0.8425076
## [50] 0.8501529
```

```r
# see how accuracy behaves
plot(accuracy, ylab="prediction accuracy",xlab="K values")
lines(accuracy)
```




```r
print (which.max(accuracy))
```

```
## [1] 12
```

Here we can look the accuracy for each K. The graph shows that accuracy with k < 5 are the worst and then it behaves similarly, being the best K in the range from 10 to 20. Our model determines that the best k is

```r
print (which.max(accuracy))
```

```
## [1] 12
```