knn

Odo Luo

March 6, 2021

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(caret)
```

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

Homework 1

Helpful link: https://rpubs.com/Mentors_Ubiqum/tunegrid_tunelength https://rpubs.com/njvijay/16444

E1: Load the iris dataset and select only entries with the classes iris virginica or iris

versicolor (so we have a binary classification problem).

```
iris <- read.csv("iris.csv",header = TRUE) %>%
  as_tibble %>%
  filter(Species== "virginica" | Species == "versicolor" ) %>%
  mutate(id=row_number())

head(iris)
```

```
## # A tibble: 6 x 6
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species       id
##          <dbl>       <dbl>        <dbl>       <dbl> <chr>      <int>
## 1          7           3.2          4.7         1.4 versicolor     1
## 2          6.4         3.2          4.5         1.5 versicolor     2
## 3          6.9         3.1          4.9         1.5 versicolor     3
## 4          5.5         2.3          4           1.3 versicolor     4
## 5          6.5         2.8          4.6         1.5 versicolor     5
## 6          5.7         2.8          4.5         1.3 versicolor     6
```

E2: Use the kNN-classes of sklearn in Python and the caret package in R with a K of 5

and a train-test-split of 70-30 for an initial classification and calculate the accuracy using the test set.

Splitting data into 70/30 train/test subsets:

```
train <- iris %>%  sample_frac(.70)
test <-  anti_join(iris, train,'id')
train <- train %>% select(-id)
test <- test %>% select(-id)
```

Train knn using caret package with k=5

```
control <- trainControl(method="repeatedcv",repeats=7)
knn <- train(Species~ ., data=train,method="knn",trControl=control,tuneLength=10,tuneGrid=data.frame(k=5
knn
```

```
## k-Nearest Neighbors
##
## 70 samples
##  4 predictor
##  2 classes: 'versicolor', 'virginica'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 7 times)
## Summary of sample sizes: 63, 63, 63, 64, 63, 62, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9576531  0.9165533
##
## Tuning parameter 'k' was held constant at a value of 5
```

Prediction

```
test<-test %>% mutate(Species=as.factor(Species))
prediction<- predict(knn,newdata = test)
confusionMatrix(prediction,test$Species)
```

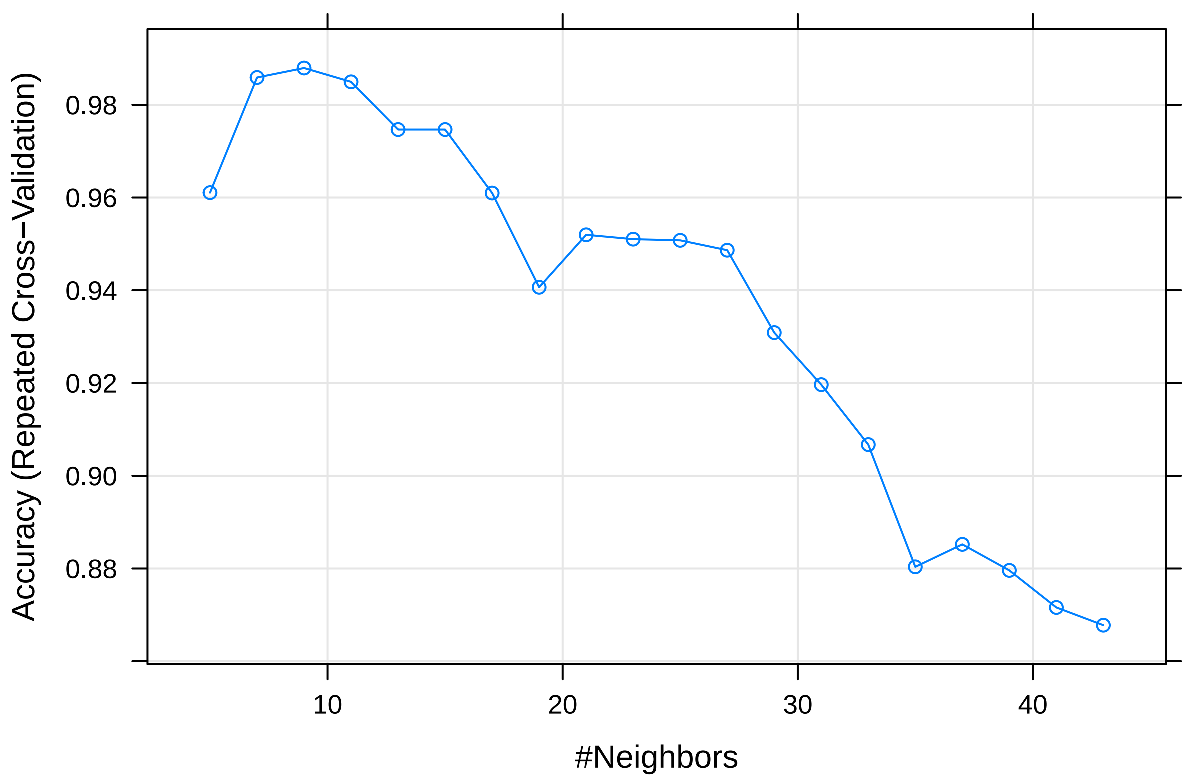
```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   versicolor virginica
##   versicolor         13         1
##   virginica           1        15
##
##                Accuracy : 0.9333
##                  95% CI : (0.7793, 0.9918)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : 2.326e-06
##
##                   Kappa : 0.8661
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9286
##             Specificity : 0.9375
```

```
##          Pos Pred Value : 0.9286
##          Neg Pred Value : 0.9375
##              Prevalence : 0.4667
##          Detection Rate : 0.4333
##    Detection Prevalence : 0.4667
##       Balanced Accuracy : 0.9330
##
##        'Positive' Class : versicolor
##
```

E3: Use the extensive search approach to identify a good k, plot the accuracy for all k

you tried, and explain your choice of a "good" k.

```
kFinding <- train(Species~ ., data=train,method="knn",trControl=control,tuneLength=20)
plot(kFinding)
```



With these data a k of 9 would be a good choice in regards of accuracy because it has max accuracy. Depending on the data, the number of k must be consindered.

E4 (non-coding): Elaborate on the strengths and weaknesses of the kNN-classifier and

give examples where you would not use it.

KNN is a lazy loading model. It advantages lie in the abitily of making real time predictions since new data can constantly be added and its simplicity, since it is easy to understand. The disadvantages are its bad perfomance wiht large data sets. It also is sensitive to outliers, therefore noisy datasets are not recommended to be used.