KNN Classification

Siddharth.S.Chandran (18BCE1003)

04/04/2021

Dataset => Cancer cell dataset

K-NN is a clustering algorithm used to find features in data that are related in natural or hard to understand ways. K-NN is great for finding 'groups' in data and classifying them.

```
data <- read.csv("C:/Users/Siddharth.S.Chandran/Downloads/data.csv", header=TRUE)
head(data)
```

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38         122.80    1001.0
## 2   842517         M       20.57        17.77         132.90    1326.0
## 3 84300903         M       19.69        21.25         130.00    1203.0
## 4 84348301         M       11.42        20.38          77.58     386.1
## 5 84358402         M       20.29        14.34         135.10    1297.0
## 6   843786         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean concavity_mean concave.points_mean
## 1         0.11840          0.27760         0.3001             0.14710
## 2         0.08474          0.07864         0.0869             0.07017
## 3         0.10960          0.15990         0.1974             0.12790
## 4         0.14250          0.28390         0.2414             0.10520
## 5         0.10030          0.13280         0.1980             0.10430
## 6         0.12780          0.17000         0.1578             0.08089
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.2419                0.07871    1.0950     0.9053        8.589
## 2        0.1812                0.05667    0.5435     0.7339        3.398
## 3        0.2069                0.05999    0.7456     0.7869        4.585
## 4        0.2597                0.09744    0.4956     1.1560        3.445
## 5        0.1809                0.05883    0.7572     0.7813        5.438
## 6        0.2087                0.07613    0.3345     0.8902        2.217
##   area_se smoothness_se compactness_se concavity_se concave.points_se
## 1  153.40      0.006399        0.04904      0.05373           0.01587
## 2   74.08      0.005225        0.01308      0.01860           0.01340
## 3   94.03      0.006150        0.04006      0.03832           0.02058
## 4   27.23      0.009110        0.07458      0.05661           0.01867
## 5   94.44      0.011490        0.02461      0.05688           0.01885
## 6   27.19      0.007510        0.03345      0.03672           0.01137
##   symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
## 1     0.03003             0.006193        25.38         17.33          184.60
## 2     0.01389             0.003532        24.99         23.41          158.80
## 3     0.02250             0.004571        23.57         25.53          152.50
## 4     0.05963             0.009208        14.91         26.50           98.87
## 5     0.01756             0.005115        22.54         16.67          152.20
## 6     0.02165             0.005082        15.47         23.75          103.40
##   area_worst smoothness_worst compactness_worst concavity_worst
## 1     2019.0           0.1622            0.6656          0.7119
## 2     1956.0           0.1238            0.1866          0.2416
## 3     1709.0           0.1444            0.4245          0.4504
## 4      567.7           0.2098            0.8663          0.6869
## 5     1575.0           0.1374            0.2050          0.4000
## 6      741.6           0.1791            0.5249          0.5355
##   concave.points_worst symmetry_worst fractal_dimension_worst  X
## 1               0.2654         0.4601                 0.11890 NA
## 2               0.1860         0.2750                 0.08902 NA
## 3               0.2430         0.3613                 0.08758 NA
## 4               0.2575         0.6638                 0.17300 NA
## 5               0.1625         0.2364                 0.07678 NA
## 6               0.1741         0.3985                 0.12440 NA
```

```r
#drop id, can lead to prediction errors if we forget about it
data <- data[-1]
#we care mostly about the diagnosis variable, which is the dependent variable in our model
table(data$diagnosis)
```

```
##
##   B   M
## 357 212
```

```r
#Also make the variable more informative
data$diagnosis <- factor(data$diagnosis, levels = c('B','M'),
                         labels = c('Benign', 'Malignant'))
#look at proportions
round(prop.table(table(data$diagnosis)) * 100, digits = 1)
```

```
##
##    Benign Malignant
##      62.7      37.3
```

```r
#See how the values will react to KNN
summary(data[c('radius_mean','area_mean','smoothness_mean')])
```

```
##   radius_mean       area_mean      smoothness_mean
##  Min.   : 6.981   Min.   : 143.5   Min.   :0.05263
##  1st Qu.:11.700   1st Qu.: 420.3   1st Qu.:0.08637
##  Median :13.370   Median : 551.1   Median :0.09587
##  Mean   :14.127   Mean   : 654.9   Mean   :0.09636
##  3rd Qu.:15.780   3rd Qu.: 782.7   3rd Qu.:0.10530
##  Max.   :28.110   Max.   :2501.0   Max.   :0.16340
```

```r
#Clearly the values will need to be normalized
```

```r
#Normalize the values with Min-Max
#This is a way of making every value in between 0 and 1, so each observation effects the
#classifier in the same way
```

```
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
#Test out the function
normalize(c(1,2,3,4,5))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```
normalize(c(10,20,30,40,50))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```
#Use lapply to normalize each column in the df
data_n <- as.data.frame(lapply(data[2:31], normalize))
summary(data_n[c('radius_mean','area_mean','smoothness_mean')])
```

```
##   radius_mean       area_mean      smoothness_mean
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2233   1st Qu.:0.1174   1st Qu.:0.3046
##  Median :0.3024   Median :0.1729   Median :0.3904
##  Mean   :0.3382   Mean   :0.2169   Mean   :0.3948
##  3rd Qu.:0.4164   3rd Qu.:0.2711   3rd Qu.:0.4755
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```
#Lets try some prediction
library(caret)
```

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

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

```
data_train <- data_n[1:469,]
data_test <- data_n[470:569,]
#Need to give labels to the new sets
data_train_labels <- data[1:469,1]
data_test_labels <- data[470:569,1]
```

```
#KNN implementation

#The K parameter decides how many neighbors we'd like to consider the distances of when
#implementing the classifier. We typically use K=21 b/c sqrt(459) which is the number
#of observations
library(class)
data_test_pred <- knn(train=data_train, test=data_test,
                      cl=data_train_labels, k=21)
```

```
#Evaluate the algorithm
conf_matrix <- table(data_test_labels, data_test_pred)
conf_matrix
```

```
##                   data_test_pred
## data_test_labels Benign Malignant
##        Benign        77         0
##        Malignant      2        21
```

```
performance <- sum(diag(conf_matrix)) / sum(conf_matrix)
performance
```

```
## [1] 0.98
```

```
#Incredible performance
```