Question 3 -KNN

Anil Akyildirim

3/29/2020

```
#load dataset
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------------
```

```
## <U+2713> ggplot2 3.2.1     <U+2713> purrr   0.3.3
## <U+2713> tibble  2.1.3     <U+2713> dplyr   0.8.3
## <U+2713> tidyr   1.0.0     <U+2713> stringr 1.4.0
## <U+2713> readr   1.3.1     <U+2713> forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
knn.data <- read.csv("icu.csv")
head(knn.data)
```

```
##   ID STA AGE SEX RACE SER CAN CRN INF CPR SYS HRA PRE TYP FRA PO2 PH PCO
## 1  8   0  27   1    1   0   0   0   1   0 142  88   0   1   0   0  0   0
## 2 12   0  59   0    1   0   0   0   0   0 112  80   1   1   0   0  0   0
## 3 14   0  77   0    1   1   0   0   0   0 100  70   0   0   0   0  0   0
## 4 28   0  54   0    1   0   0   0   1   0 142 103   0   1   1   0  0   0
## 5 32   0  87   1    1   1   0   0   1   0 110 154   1   1   0   0  0   0
## 6 38   0  69   0    1   0   0   0   1   0 110 132   0   1   0   1  0   0
##   BIC CRE LOC
## 1   0   0   0
## 2   0   0   0
## 3   0   0   0
## 4   0   0   0
## 5   0   0   0
## 6   1   0   0
```

```
#look to see datatypes
str(knn.data)
```

```
## 'data.frame':    200 obs. of  21 variables:
##  $ ID  : int  8 12 14 28 32 38 40 41 42 50 ...
##  $ STA : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AGE : int  27 59 77 54 87 69 63 30 35 70 ...
##  $ SEX : int  1 0 0 0 1 0 0 1 0 1 ...
##  $ RACE: int  1 1 1 1 1 1 1 1 2 1 ...
##  $ SER : int  0 0 1 0 1 0 1 0 0 1 ...
##  $ CAN : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ CRN : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ INF : int  1 0 0 1 1 1 0 0 0 0 ...
```

```
##  $ CPR : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ SYS : int  142 112 100 142 110 110 104 144 108 138 ...
##  $ HRA : int  88 80 70 103 154 132 66 110 60 103 ...
##  $ PRE : int  0 1 0 0 1 0 0 0 0 0 ...
##  $ TYP : int  1 1 0 1 1 1 0 1 1 0 ...
##  $ FRA : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ PO2 : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ PH  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PCO : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ BIC : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ CRE : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ LOC : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
# create COMA feature
knn.data <- mutate(knn.data, COMA=ifelse(LOC==2, 1, 0))
head(knn.data)
```

```
##   ID STA AGE SEX RACE SER CAN CRN INF CPR SYS HRA PRE TYP FRA PO2 PH PCO
## 1  8   0  27   1    1   0   0   0   1   0 142  88   0   1   0   0  0   0
## 2 12   0  59   0    1   0   0   0   0   0 112  80   1   1   0   0  0   0
## 3 14   0  77   0    1   1   0   0   0   0 100  70   0   0   0   0  0   0
## 4 28   0  54   0    1   0   0   0   1   0 142 103   0   1   1   0  0   0
## 5 32   0  87   1    1   1   0   0   1   0 110 154   1   1   0   0  0   0
## 6 38   0  69   0    1   0   0   0   1   0 110 132   0   1   0   1  0   0
##   BIC CRE LOC COMA
## 1   0   0   0    0
## 2   0   0   0    0
## 3   0   0   0    0
## 4   0   0   0    0
## 5   0   0   0    0
## 6   1   0   0    0
```

```
# change class from numerical to categorical
unique(knn.data$STA)
```

```
## [1] 0 1
```

```
knn.data$STA <- factor(knn.data$STA)
```

```
# keep 5 features
features <- c('TYP', 'AGE', 'INF', 'COMA', 'STA')
knn.df <- knn.data[features]
head(knn.df)
```

```
##   TYP AGE INF COMA STA
## 1   1  27   1    0   0
## 2   1  59   0    0   0
## 3   0  77   0    0   0
## 4   1  54   1    0   0
## 5   1  87   1    0   0
## 6   1  69   1    0   0
```

```r
euclideanDist <- function(a, b){
  d = 0
  for(i in c(1:(length(a)) ))
  {
    d = d + (a[[i]]-b[[i]])^2
  }
  d = sqrt(d)
  return(d)
}

knn_predict2 <- function(test_data, train_data, k_value, labelcol){
  pred <- c()  #empty pred vector
  #LOOP-1
  for(i in c(1:nrow(test_data))){   #looping over each record of test data
    eu_dist =c()          #eu_dist & eu_char empty  vector
    eu_char = c()
    good = 0              #good & bad variable initialization with 0 value
    bad = 0

    #LOOP-2-looping over train data
    for(j in c(1:nrow(train_data))){

      #adding euclidean distance b/w test data point and train data to eu_dist vector
      eu_dist <- c(eu_dist, euclideanDist(test_data[i,-c(labelcol)], train_data[j,-c(labelcol)]))

      #adding class variable of training data in eu_char
      eu_char <- c(eu_char, as.character(train_data[j,][[labelcol]]))
    }

    eu <- data.frame(eu_char, eu_dist) #eu dataframe created with eu_char & eu_dist columns

    eu <- eu[order(eu$eu_dist),]       #sorting eu dataframe to gettop K neighbors
    eu <- eu[1:k_value,]               #eu dataframe with top K neighbors

    tbl.sm.df<-table(eu$eu_char)
    cl_label<-  names(tbl.sm.df)[[as.integer(which.max(tbl.sm.df))]]

    pred <- c(pred, cl_label)
    }
    return(pred) #return pred vector
  }


accuracy <- function(test_data,labelcol,predcol){
  correct = 0
  for(i in c(1:nrow(test_data))){
    if(test_data[i,labelcol] == test_data[i,predcol]){
      correct = correct+1
    }
  }
  accu = (correct/nrow(test_data)) * 100
  return(accu)
}
```

```
#load data
labelcol <- 5 # for df it is the fifth col
predictioncol<-labelcol+1
```

```
# create train/test partitions
set.seed(2)
n<-nrow(knn.df)
knn.df<- knn.df[sample(n),]

train.df <- knn.df[1:as.integer(0.7*n),]
```

```
K = 3 # number of neighbors to determine the class
table(train.df[,labelcol])
```

```
##
##   0   1
## 112  28
```

```
test.df <- knn.df[as.integer(0.7*n +1):n,]
table(test.df[,labelcol])
```

```
##
##  0  1
## 48 12
```

```
predictions <- knn_predict2(test.df, train.df, K,labelcol) #calling knn_predict()
test.df[,predictioncol] <- predictions #Adding predictions in test data as 7th column
print(accuracy(test.df,labelcol,predictioncol))
```

```
## [1] 78.33333
```

```
table(test.df[[predictioncol]],test.df[[labelcol]])
```

```
##
##      0  1
##   0 44  9
##   1  4  3
```

```
accuracy_score_3 <- accuracy(test.df,labelcol,predictioncol) # create accuracy score object for further
```

Above outlines the confusion MAtrix and accuracy score(76.67) when K is 3.

```
K = 5 # number of neighbors to determine the class
table(train.df[,labelcol])
```

```
##
##   0   1
## 112  28
```

```
test.df <- knn.df[as.integer(0.7*n +1):n,]
table(test.df[,labelcol])
```

```
##
##  0  1
## 48 12
```

```
predictions <- knn_predict2(test.df, train.df, K,labelcol)
test.df[,predictioncol] <- predictions #Adding predictions in test data as 7th column
print(accuracy(test.df,labelcol,predictioncol))
```

```
## [1] 80
```

```
table(test.df[[predictioncol]],test.df[[labelcol]])
```

```
##
##      0  1
##   0 47 11
##   1  1  1
```

```
accuracy_score_5 <- accuracy(test.df,labelcol,predictioncol) # create accuracy score object for further
```

Above outlines the confusion Matrix and accuracy(80) when K is 5.

```
K = 7 # number of neighbors to determine the class
table(train.df[,labelcol])
```

```
##
##   0   1
## 112  28
```

```
test.df <- knn.df[as.integer(0.7*n +1):n,]
table(test.df[,labelcol])
```

```
##
##  0  1
## 48 12
```

```
predictions <- knn_predict2(test.df, train.df, K,labelcol)
test.df[,predictioncol] <- predictions #Adding predictions in test data as 7th column
print(accuracy(test.df,labelcol,predictioncol))
```

```
## [1] 80
```

```
table(test.df[[predictioncol]],test.df[[labelcol]])
```

```
##
##      0  1
##   0 48 12
```

```
accuracy_score_7 <- accuracy(test.df,labelcol,predictioncol) # create accuracy score object for further
```

Above is the confusion matrix and accuracy score for K=7.

```
K = 15 # number of neighbors to determine the class
table(train.df[,labelcol])
```

```
##
##   0   1
## 112  28
```

```
test.df <- knn.df[as.integer(0.7*n +1):n,]
table(test.df[,labelcol])
```

```
##
##  0  1
## 48 12
```

```
predictions <- knn_predict2(test.df, train.df, K,labelcol)
test.df[,predictioncol] <- predictions #Adding predictions in test data as 7th column
print(accuracy(test.df,labelcol,predictioncol))
```

```
## [1] 80
```

```
table(test.df[[predictioncol]],test.df[[labelcol]])
```

```
##
##      0  1
##   0 48 12
```

```
accuracy_score_15 <- accuracy(test.df,labelcol,predictioncol) # create accuracy score object for furthe
```

Above is the confusion matrix and accuracy score for K=15

```
K = 25 # number of neighbors to determine the class
table(train.df[,labelcol])
```

```
##
##   0   1
## 112  28
```

```
test.df <- knn.df[as.integer(0.7*n +1):n,]
table(test.df[,labelcol])
```

```
##
##  0  1
## 48 12
```

```
predictions <- knn_predict2(test.df, train.df, K,labelcol)
test.df[,predictioncol] <- predictions #Adding predictions in test data as 7th column
print(accuracy(test.df,labelcol,predictioncol))
```

```
## [1] 80
```

```
table(test.df[[predictioncol]],test.df[[labelcol]])
```

```
##
##      0  1
##   0 48 12
```

```
accuracy_score_25 <- accuracy(test.df,labelcol,predictioncol) # create accuracy score object for furthe
```

Above is the confusion matrix and accuracy score for K=25

```
K = 50 # number of neighbors to determine the class
table(train.df[,labelcol])
```

```
##
##   0   1
## 112  28
```

```
test.df <- knn.df[as.integer(0.7*n +1):n,]
table(test.df[,labelcol])
```

```
##
##  0  1
## 48 12
```

```
predictions <- knn_predict2(test.df, train.df, K,labelcol)
test.df[,predictioncol] <- predictions #Adding predictions in test data as 7th column
print(accuracy(test.df,labelcol,predictioncol))
```

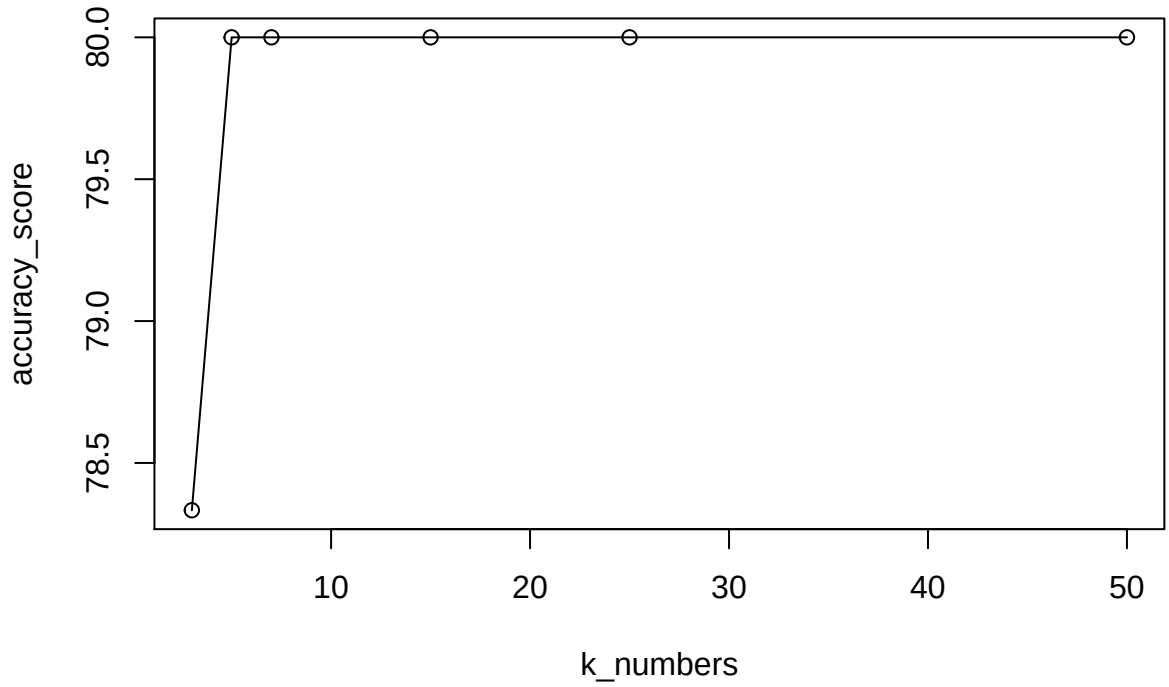
```
## [1] 80
```

```
table(test.df[[predictioncol]],test.df[[labelcol]])
```

```
##
##      0  1
##   0 48 12
```

```
accuracy_score_50 <- accuracy(test.df,labelcol,predictioncol) # create accuracy score object for furthe
```

```
accuracy_score <- c(accuracy_score_3, accuracy_score_5, accuracy_score_7, accuracy_score_15, accuracy_s
k_numbers <- c(3,5,7,15,25,50)
plot(k_numbers, accuracy_score)
lines(k_numbers, accuracy_score)
```



TWe selected TYP, AGE and INF as a descriptive variable from our dataset. We created COMA variable with a defined condition and added to the descriptive variables. Our target variable is STA which has two classes (0 and 1). We defined our KNN model from scratch with distance function (iterate, sort , combine) for prediction and we also defined accuracy function. We run the model for K(3,5,7,15,25,50), note the predicted vs real values(confusionmatrix), accuracy for each K and plot of accuracy vs K. We notice that K>7 (15,25,50), even though accuracy is higher, we are getting wrong predicted values. Ideal K value is 7.