Knn.R

DELL

2024-04-07

#a)  
data = read.csv("C:\\Users\\DELL\\OneDrive\\Desktop\\Desktop\\College\\DATA SCIENCE\\R\\RDS\\Breast Cancer Prediction\\Breast Cancer Wisconsin.csv")  
dim(data) #dimension

## [1] 569 32

str(data) #data type of each variable

## 'data.frame': 569 obs. of 32 variables:  
## $ id : int 842302 842517 84300903 84348301 84358402 843786 844359 84458202 844981 84501001 ...  
## $ diagnosis : chr "M" "M" "M" "M" ...  
## $ radius\_mean : num 18 20.6 19.7 11.4 20.3 ...  
## $ texture\_mean : num 10.4 17.8 21.2 20.4 14.3 ...  
## $ perimeter\_mean : num 122.8 132.9 130 77.6 135.1 ...  
## $ area\_mean : num 1001 1326 1203 386 1297 ...  
## $ smoothness\_mean : num 0.1184 0.0847 0.1096 0.1425 0.1003 ...  
## $ compactness\_mean : num 0.2776 0.0786 0.1599 0.2839 0.1328 ...  
## $ concavity\_mean : num 0.3001 0.0869 0.1974 0.2414 0.198 ...  
## $ concave.points\_mean : num 0.1471 0.0702 0.1279 0.1052 0.1043 ...  
## $ symmetry\_mean : num 0.242 0.181 0.207 0.26 0.181 ...  
## $ fractal\_dimension\_mean : num 0.0787 0.0567 0.06 0.0974 0.0588 ...  
## $ radius\_se : num 1.095 0.543 0.746 0.496 0.757 ...  
## $ texture\_se : num 0.905 0.734 0.787 1.156 0.781 ...  
## $ perimeter\_se : num 8.59 3.4 4.58 3.44 5.44 ...  
## $ area\_se : num 153.4 74.1 94 27.2 94.4 ...  
## $ smoothness\_se : num 0.0064 0.00522 0.00615 0.00911 0.01149 ...  
## $ compactness\_se : num 0.049 0.0131 0.0401 0.0746 0.0246 ...  
## $ concavity\_se : num 0.0537 0.0186 0.0383 0.0566 0.0569 ...  
## $ concave.points\_se : num 0.0159 0.0134 0.0206 0.0187 0.0188 ...  
## $ symmetry\_se : num 0.03 0.0139 0.0225 0.0596 0.0176 ...  
## $ fractal\_dimension\_se : num 0.00619 0.00353 0.00457 0.00921 0.00511 ...  
## $ radius\_worst : num 25.4 25 23.6 14.9 22.5 ...  
## $ texture\_worst : num 17.3 23.4 25.5 26.5 16.7 ...  
## $ perimeter\_worst : num 184.6 158.8 152.5 98.9 152.2 ...  
## $ area\_worst : num 2019 1956 1709 568 1575 ...  
## $ smoothness\_worst : num 0.162 0.124 0.144 0.21 0.137 ...  
## $ compactness\_worst : num 0.666 0.187 0.424 0.866 0.205 ...  
## $ concavity\_worst : num 0.712 0.242 0.45 0.687 0.4 ...  
## $ concave.points\_worst : num 0.265 0.186 0.243 0.258 0.163 ...  
## $ symmetry\_worst : num 0.46 0.275 0.361 0.664 0.236 ...  
## $ fractal\_dimension\_worst: num 0.1189 0.089 0.0876 0.173 0.0768 ...

#b)Splitting the data  
  
set.seed(123)  
test\_size = sample(nrow(data), 100)  
  
data\_test = data[test\_size,]  
  
data\_train = data[-test\_size, ]  
  
  
train\_features = data\_train[, 3:32]  
test\_features = data\_test[, 3:32]  
  
#c)   
  
library(class)  
#scaling the data  
  
  
knn\_model = knn(train\_features, test\_features, cl = data\_train$diagnosis, k = 21)  
knn\_model

## [1] M B B B B M B B B B M B B B M B B M M B B M M B M M B B B B B B B B M M B  
## [38] B B B B B B B B B B M B M B M B M B B B B B B B B B M B B B B M M B B B B  
## [75] B M B B M M B B B M M M B B B M B B B B B M B B B B  
## Levels: B M

confusion\_matrix = table(actual = data\_test$diagnosis,predict = knn\_model)  
  
#Accuracy = (TP+TN)/(TP+TN\_FP+FN)  
accuracy = sum(diag(confusion\_matrix))/sum(confusion\_matrix)  
accuracy

## [1] 0.94

#d) summmarize and note the difference  
  
summary(data$radius\_mean)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 6.981 11.700 13.370 14.127 15.780 28.110

summary(data$area\_mean)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 143.5 420.3 551.1 654.9 782.7 2501.0

summary(data$smoothness\_mean)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.05263 0.08637 0.09587 0.09636 0.10530 0.16340

#The units of measurement are too vast  
  
#e) min-max normalization  
  
#FORMULA: (x - min of x)/ (max of x - min of x)  
normalized\_data = function(data) {  
   
 scaled\_data = data  
   
 for (i in 3:ncol(data)){ #iterating over the columns (there are 30 columns)  
 x\_min = min(data[,i]) #all the rows in each column  
 x\_max = max(data[,i])  
   
 if (x\_max - x\_min != 0){  
 scaled\_data[,i] = (data[,i] - x\_min)/(x\_max-x\_min)  
 }  
 else {  
 scaled\_data[,i] = 0  
 }  
   
 return (scaled\_data)  
 }  
   
}  
  
normalized\_data =normalized\_data(data)  
  
#f) splitting into train and test  
  
normal\_train = normalized\_data[1:469, ]  
  
normal\_test = normalized\_data[470:569, ]  
  
normal\_train\_features = normal\_train[, 3:32]  
normal\_test\_features = normal\_test[, 3:32]  
  
#g) build the model on normalized data  
  
knn\_model\_normalized = knn(normal\_train\_features, normal\_test\_features, cl = normal\_train$diagnosis, k = 21)  
knn\_model\_normalized

## [1] B B B M B B B B B B M B M B B B B B M B M B M M B B B B B M M B M B M B B  
## [38] B B M M B B M B M B M M B B B M B B B B B B B B B B B M B M B B B B B B B  
## [75] B B B B B B B B B B B B B B B B B B B M M M M M M B  
## Levels: B M

confusion\_matrix = table(actual = normal\_test$diagnosis,predict = knn\_model\_normalized)  
  
#Accuracy = (TP+TN)/(TP+TN\_FP+FN)  
accuracy = sum(diag(confusion\_matrix))/sum(confusion\_matrix)  
accuracy

## [1] 0.95

#h) scaling the data according to z-score standardization  
  
train\_scale = scale(data\_train[,3:32])  
test\_scale = scale(data\_test[, 3:32])  
  
knn\_model\_scaled = knn(train\_scale, test\_scale, cl = data\_train$diagnosis, k = 21)  
knn\_model\_scaled

## [1] B B B B M M B B B M M B B B M B B M M B B M M B M M B B B B B B B B M B M  
## [38] M B B B B B M B M B M B M M M B M B B B B B B B B B M B B B B M M B B B B  
## [75] B M B B B M B B B M M M B B B M B B B B B M M B B B  
## Levels: B M

confusion\_matrix = table(actual = data\_test$diagnosis,predict = knn\_model\_scaled)  
  
#Accuracy = (TP+TN)/(TP+TN\_FP+FN)  
accuracy = sum(diag(confusion\_matrix))/sum(confusion\_matrix)  
accuracy

## [1] 0.91