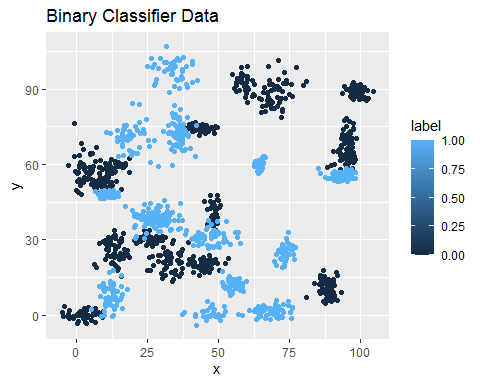
8.2 Assignment: Introduction to Machine Learning

Rachel Nelson

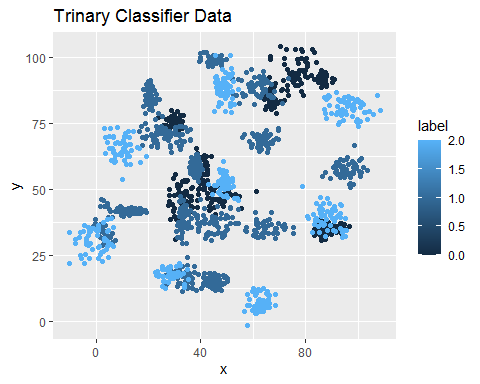
7/25/2020

####a. Plot the data from each dataset using a scatter plot.

ggplot(bcd\_df, aes(x=x, y=y, color=label)) +  
 geom\_point() + ggtitle("Binary Classifier Data")



ggplot(tcd\_df, aes(x=x, y=y, color=label)) +  
 geom\_point() + ggtitle("Trinary Classifier Data")



####b. The k nearest neighbors algorithm categorizes an input value by looking at the labels for the k nearest points and assigning a category based on the most common label. In this problem, you will determine which points are nearest by calculating the Euclidean distance between two points. As a refresher, the Euclidean distance between two points:

####Fitting a model is when you use the input data to create a predictive model. There are various metrics you can use to determine how well your model fits the data. You will learn more about these metrics in later lessons. For this problem, you will focus on a single metric; accuracy. Accuracy is simply the percentage of how often the model predicts the correct result. If the model always predicts the correct result, it is 100% accurate. If the model always predicts the incorrect result, it is 0% accurate.

####Fit a k nearest neighbors model for each dataset for k=3, k=5, k=10, k=15, k=20, and k=25. Compute the accuracy of the resulting models for each value of k. Plot the results in a graph where the x-axis is the different values of k and the y-axis is the accuracy of the model.

# First Value  
head(bcd\_df)

## label x y  
## 1 0 70.88469 83.17702  
## 2 0 74.97176 87.92922  
## 3 0 73.78333 92.20325  
## 4 0 66.40747 81.10617  
## 5 0 69.07399 84.53739  
## 6 0 72.23616 86.38403

str(bcd\_df)

## 'data.frame': 1498 obs. of 3 variables:  
## $ label: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ x : num 70.9 75 73.8 66.4 69.1 ...  
## $ y : num 83.2 87.9 92.2 81.1 84.5 ...

# creating test and training data sets   
bcd.subset <- as.data.frame(bcd\_df)  
head(bcd.subset)

## label x y  
## 1 0 70.88469 83.17702  
## 2 0 74.97176 87.92922  
## 3 0 73.78333 92.20325  
## 4 0 66.40747 81.10617  
## 5 0 69.07399 84.53739  
## 6 0 72.23616 86.38403

nomalize <- function(x) {  
 return ((x - min(x))/(max(x)-min(x)))}  
  
set.seed(123)  
dat.d <- sample(1:nrow(bcd\_df), size=nrow(bcd\_df)\*0.7, replace = FALSE)  
  
train.bcd <- bcd\_df[dat.d,]  
test.bcd <- bcd\_df[-dat.d,]  
  
train.bcd\_labels <- bcd\_df[dat.d,1]  
test.bcd\_labels <- bcd\_df[-dat.d,1]  
  
## KNN model and accuracy  
  
knn.3 <- knn(train=train.bcd, test=test.bcd, cl=train.bcd\_labels, k=3)  
bcd.ACC.3 <-100 \* sum(test.bcd\_labels == knn.3)/NROW(test.bcd\_labels)  
bcd.ACC.3

## [1] 98

knn.5 <- knn(train=train.bcd, test=test.bcd, cl=train.bcd\_labels, k=5)  
bcd.ACC.5 <-100 \* sum(test.bcd\_labels == knn.5)/NROW(test.bcd\_labels)  
bcd.ACC.5

## [1] 97.55556

knn.10 <- knn(train=train.bcd, test=test.bcd, cl=train.bcd\_labels, k=10)  
bcd.ACC.10 <-100 \* sum(test.bcd\_labels == knn.10)/NROW(test.bcd\_labels)  
bcd.ACC.10

## [1] 98

knn.15 <- knn(train=train.bcd, test=test.bcd, cl=train.bcd\_labels, k=15)  
bcd.ACC.15 <-100 \* sum(test.bcd\_labels == knn.15)/NROW(test.bcd\_labels)  
bcd.ACC.15

## [1] 97.55556

knn.20 <- knn(train=train.bcd, test=test.bcd, cl=train.bcd\_labels, k=20)  
bcd.ACC.20 <-100 \* sum(test.bcd\_labels == knn.20)/NROW(test.bcd\_labels)  
bcd.ACC.20

## [1] 97.55556

knn.25 <- knn(train=train.bcd, test=test.bcd, cl=train.bcd\_labels, k=25)  
bcd.ACC.25 <-100 \* sum(test.bcd\_labels == knn.25)/NROW(test.bcd\_labels)  
bcd.ACC.25

## [1] 98.44444

# First Value  
head(tcd\_df)

## label x y  
## 1 0 30.08387 39.63094  
## 2 0 31.27613 51.77511  
## 3 0 34.12138 49.27575  
## 4 0 32.58222 41.23300  
## 5 0 34.65069 45.47956  
## 6 0 33.80513 44.24656

str(tcd\_df)

## 'data.frame': 1568 obs. of 3 variables:  
## $ label: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ x : num 30.1 31.3 34.1 32.6 34.7 ...  
## $ y : num 39.6 51.8 49.3 41.2 45.5 ...

# creating test and training data sets   
tcd.subset <- as.data.frame(tcd\_df)  
head(tcd.subset)

## label x y  
## 1 0 30.08387 39.63094  
## 2 0 31.27613 51.77511  
## 3 0 34.12138 49.27575  
## 4 0 32.58222 41.23300  
## 5 0 34.65069 45.47956  
## 6 0 33.80513 44.24656

nomalize <- function(x) {  
 return ((x - min(x))/(max(x)-min(x)))}  
  
set.seed(123)  
dat.d <- sample(1:nrow(tcd\_df), size=nrow(tcd\_df)\*0.7, replace = FALSE)  
  
train.tcd <- tcd\_df[dat.d,]  
test.tcd <- tcd\_df[-dat.d,]  
  
train.tcd\_labels <- tcd\_df[dat.d,1]  
test.tcd\_labels <- tcd\_df[-dat.d,1]  
  
## KNN model and accuracy  
  
knn.3 <- knn(train=train.tcd, test=test.tcd, cl=train.tcd\_labels, k=3)  
tcd.ACC.3 <-100 \* sum(test.tcd\_labels == knn.3)/NROW(test.tcd\_labels)  
tcd.ACC.3

## [1] 93.20594

knn.5 <- knn(train=train.tcd, test=test.tcd, cl=train.tcd\_labels, k=5)  
tcd.ACC.5 <-100 \* sum(test.tcd\_labels == knn.5)/NROW(test.tcd\_labels)  
tcd.ACC.5

## [1] 92.14437

knn.10 <- knn(train=train.tcd, test=test.tcd, cl=train.tcd\_labels, k=10)  
tcd.ACC.10 <-100 \* sum(test.tcd\_labels == knn.10)/NROW(test.tcd\_labels)  
tcd.ACC.10

## [1] 90.02123

knn.15 <- knn(train=train.tcd, test=test.tcd, cl=train.tcd\_labels, k=15)  
tcd.ACC.15 <-100 \* sum(test.tcd\_labels == knn.15)/NROW(test.tcd\_labels)  
tcd.ACC.15

## [1] 89.17197

knn.20 <- knn(train=train.tcd, test=test.tcd, cl=train.tcd\_labels, k=20)  
tcd.ACC.20 <-100 \* sum(test.tcd\_labels == knn.20)/NROW(test.tcd\_labels)  
tcd.ACC.20

## [1] 87.26115

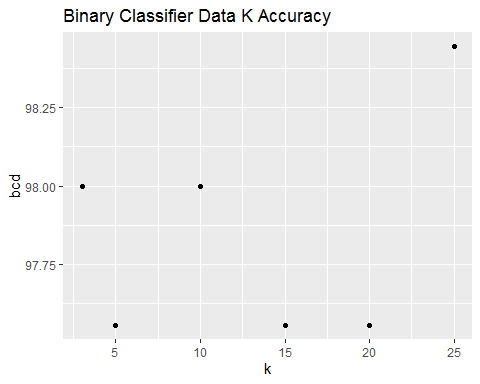
knn.25 <- knn(train=train.tcd, test=test.tcd, cl=train.tcd\_labels, k=25)  
tcd.ACC.25 <-100 \* sum(test.tcd\_labels == knn.25)/NROW(test.tcd\_labels)  
tcd.ACC.25

## [1] 86.83652

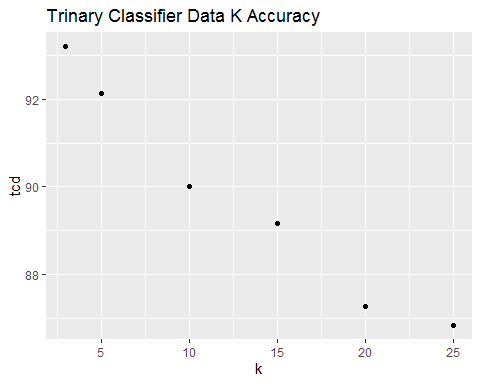
#Plot the results in a graph where the x-axis is the different values of k and the y-axis is the accuracy of the model  
k\_value <- c(3,5,10,15,25)  
bcd.ACC\_value <- c(bcd.ACC.3,bcd.ACC.5,bcd.ACC.10,bcd.ACC.15,bcd.ACC.25)  
tcd.ACC\_value <- c(tcd.ACC.3,tcd.ACC.5,tcd.ACC.10,tcd.ACC.15,tcd.ACC.25)  
  
  
model\_accuracy <- matrix(c(3,bcd.ACC.3,tcd.ACC.3,5,bcd.ACC.5,tcd.ACC.5,10,bcd.ACC.10,tcd.ACC.10,15,bcd.ACC.15,tcd.ACC.15,20,bcd.ACC.20,tcd.ACC.20,25,bcd.ACC.25,tcd.ACC.25),ncol=3,byrow=TRUE)  
colnames(model\_accuracy) <- c("k","bcd","tcd")  
rownames(model\_accuracy) <- c("3","5","10","15","20","25")  
model\_accuracy <- as.data.frame(model\_accuracy)  
model\_accuracy

## k bcd tcd  
## 3 3 98.00000 93.20594  
## 5 5 97.55556 92.14437  
## 10 10 98.00000 90.02123  
## 15 15 97.55556 89.17197  
## 20 20 97.55556 87.26115  
## 25 25 98.44444 86.83652

ggplot(model\_accuracy, aes(x=k, y=bcd)) +  
 geom\_point() + ggtitle("Binary Classifier Data K Accuracy")



ggplot(model\_accuracy, aes(x=k, y=tcd)) +  
 geom\_point() + ggtitle("Trinary Classifier Data K Accuracy")



####c. In later lessons, you will learn about linear classifiers. These algorithms work by defining a decision boundary that separates the different categories. ####Looking back at the plots of the data, do you think a linear classifier would work well on these datasets? Maybe? It’s hard to say for sure because the clusters have more then one k. However, since the data set allows for supervised learning, you could compare the accuracy of the linear classifier to the accuracy of KNN and check to see which model provides the better outcome.