wk9\_Exercise15\_MachineLearning

SumbarajuAditya

2/13/2021

## dataframes for BinaryClassifier and TrinaryClassifier

library(ggplot2)  
options(scipen = 999)  
  
# dataframes for BinaryClassifier and TrinaryClassifier  
BinaryClassifier <- read.csv("C:/BU/DSC520/assignment\_repo/dsc520/data/binary-classifier-data.csv")  
TrinaryClassifier <- read.csv("C:/BU/DSC520/assignment\_repo/dsc520/data/trinary-classifier-data.csv")  
# Structure of Binary Classifier  
str(BinaryClassifier)

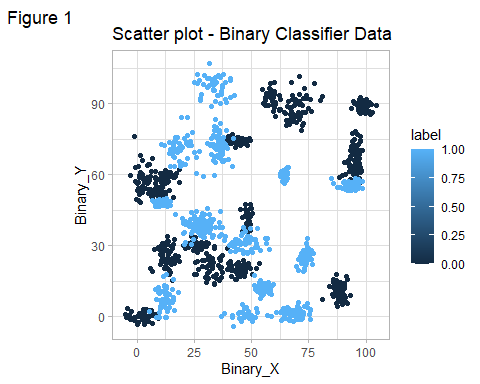
## 'data.frame': 1498 obs. of 3 variables:  
## $ label: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ x : num 70.9 75 73.8 66.4 69.1 ...  
## $ y : num 83.2 87.9 92.2 81.1 84.5 ...

# Structure of Trinary Classifier  
str(TrinaryClassifier)

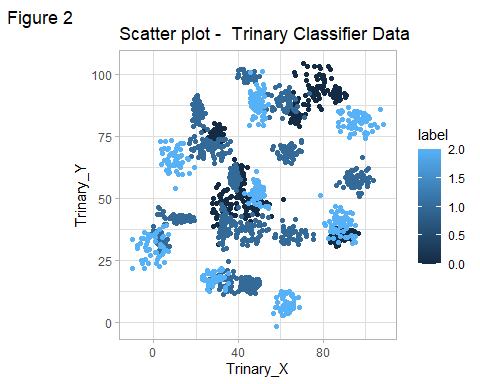
## 'data.frame': 1568 obs. of 3 variables:  
## $ label: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ x : num 30.1 31.3 34.1 32.6 34.7 ...  
## $ y : num 39.6 51.8 49.3 41.2 45.5 ...

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

library("ggplot2")  
BinaryClassiferPlot <- ggplot(BinaryClassifier, aes(x=x, y=y, color=label)) +  
 geom\_point() + labs(title="Scatter plot - Binary Classifier Data", tag = "Figure 1",x = "Binary\_X",  
 y = "Binary\_Y", Color="Binary") + theme\_light()  
  
TrinaryClassiferPlot <- ggplot(TrinaryClassifier, aes(x=x, y=y, color=label)) +  
 geom\_point() + labs(title="Scatter plot - Trinary Classifier Data", tag = "Figure 2", x = "Trinary\_X",  
 y = "Trinary\_Y",Color="Trinary") + theme\_light()  
  
BinaryClassiferPlot



TrinaryClassiferPlot

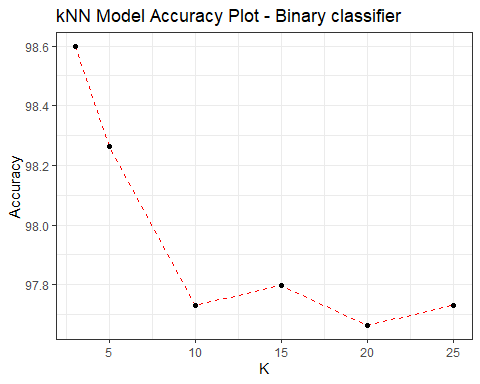


## B. The k nearest neighbors algorithm categorizes an input value by looking at the labels for the k nearest points and assigning a category based on the most common label. In this problem, you will determine which points are nearest by calculating the Euclidean distance between two points. As a refresher,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

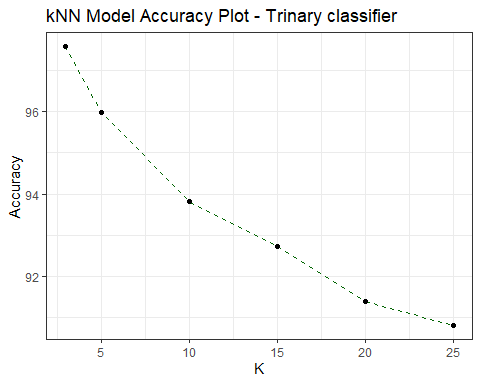
library(caret)

## Loading required package: lattice

library(class)  
library(lattice)  
library("caTools")  
  
set.seed(30)  
## Binary classifier  
BinaryClassifier\_Split<-sample.split(BinaryClassifier, SplitRatio=0.75)  
BinaryClassifier\_Train <- subset(BinaryClassifier, BinaryClassifier\_Split="TRUE")  
BinaryClassifier\_Test <- subset(BinaryClassifier, BinaryClassifier\_Split="FALSE")  
  
  
in\_K\_List = c(3, 5, 10, 15, 20, 25)  
accuracyBin <- c()  
index = 0  
for (i in in\_K\_List) {   
 index = index + 1  
 Bin\_knn <- knn(train=BinaryClassifier\_Train, test=BinaryClassifier\_Test, cl=BinaryClassifier\_Train$label, k=i )  
 accuracyBin[index] <- 100 \* sum(BinaryClassifier\_Test$label == Bin\_knn)/NROW(BinaryClassifier\_Test$label)  
 }  
  
## Trinary classifier  
TrinaryClassifier\_Split<-sample.split(TrinaryClassifier, SplitRatio=0.75)  
TrinaryClassifier\_Train <- subset(TrinaryClassifier, TrinaryClassifier\_Split="TRUE")  
TrinaryClassifier\_Test <- subset(TrinaryClassifier, TrinaryClassifier\_Split="FALSE")  
  
  
in\_K\_List = c(3, 5, 10, 15, 20, 25)  
accuracyTrn <- c()  
index = 0  
for (i in in\_K\_List) {   
 index = index + 1  
 Trn\_knn <- knn(train=TrinaryClassifier\_Train, test=TrinaryClassifier\_Test, cl=TrinaryClassifier\_Train$label, k=i )  
 accuracyTrn[index] <- 100 \* sum(TrinaryClassifier\_Test$label == Trn\_knn)/NROW(TrinaryClassifier\_Test$label)  
}  
  
acc\_bin\_data <- data.frame(in\_K\_List,accuracyBin)  
names(acc\_bin\_data) <- c("K","Accuracy")  
  
acc\_tri\_data <- data.frame(in\_K\_List,accuracyTrn)  
names(acc\_tri\_data) <- c("K","Accuracy")  
  
## Plot Accuracy  
  
BinaryAccuracyPlot <- ggplot(acc\_bin\_data, aes(x = K, y = Accuracy)) +   
 geom\_line(linetype = "dashed", color="red") + geom\_point() +   
 labs(title="kNN Model Accuracy Plot - Binary classifier") + theme\_bw()  
  
TrinaryAccuracyPlot <- ggplot(acc\_tri\_data, aes(x = K, y = Accuracy)) +  
 geom\_line(linetype = "dashed", color="dark green") + geom\_point() +  
 labs(title="kNN Model Accuracy Plot - Trinary classifier") + theme\_bw()  
  
BinaryAccuracyPlot



TrinaryAccuracyPlot



## 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?

From the below mentioned observations; I don’t think linear classifier would work well on these data-sets Plot observations: 1. Data seems to be wide spread. 2. Accuracy gets dropped with an increase value of K.Looks like linear classifier won’t work on these data-sets. 3. In my opinion; As the data is randomly spread across it would be helpful if we use classification method instead of clustering.