Exercise 15

Abhigyan Misra

October 27th 2020

# Exercise 15: Introduction to Machine Learning

Problem Statement : These assignments are here to provide you with an introduction to the “Data Science” use for these tools. This is your future. It may seem confusing and weird right now but it hopefully seems far less so than earlier in the semester. Attempt these homework assignments. You will not be graded on your answer but on your approach. This should be a, “Where am I on learning this stuff” check. If you can’t get it done, please explain why.

Include all of your answers in a R Markdown report.

Regression algorithms are used to predict numeric quantity while classification algorithms predict categorical outcomes. A spam filter is an example use case for a classification algorithm. The input dataset is emails labeled as either spam (i.e. junk emails) or ham (i.e. good emails). The classification algorithm uses features extracted from the emails to learn which emails fall into which category.

In this problem, you will use the nearest neighbors algorithm to fit a model on two simplified datasets. The first dataset (found in binary-classifier-data.csv) contains three variables; label, x, and y. The label variable is either 0 or 1 and is the output we want to predict using the x and y variables. The second dataset (found in trinary-classifier-data.csv) is similar to the first dataset except that the label variable can be 0, 1, or 2.

Note that in real-world datasets, your labels are usually not numbers, but text-based descriptions of the categories (e.g. spam or ham). In practice, you will encode categorical variables into numeric values.

# Solution

## Set the working directory to the root of your DSC 520 directory  
setwd("C:/git-bellevue/dsc520-fork")  
  
## Load the `caTools` library  
library(caTools)

## Warning: package 'caTools' was built under R version 4.0.3

## Load the `data/binary-classifier-data.csv` as df  
binary\_classifier\_df <- read.csv("data/binary-classifier-data.csv")  
head(binary\_classifier\_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

summary(binary\_classifier\_df)

## label x y   
## Min. :0.000 Min. : -5.20 Min. : -4.019   
## 1st Qu.:0.000 1st Qu.: 19.77 1st Qu.: 21.207   
## Median :0.000 Median : 41.76 Median : 44.632   
## Mean :0.488 Mean : 45.07 Mean : 45.011   
## 3rd Qu.:1.000 3rd Qu.: 66.39 3rd Qu.: 68.698   
## Max. :1.000 Max. :104.58 Max. :106.896

## Load the `data/trinary-classifier-data.csv` as df  
trinary\_classifier\_df <- read.csv("data/trinary-classifier-data.csv")  
head(binary\_classifier\_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

summary(binary\_classifier\_df)

## label x y   
## Min. :0.000 Min. : -5.20 Min. : -4.019   
## 1st Qu.:0.000 1st Qu.: 19.77 1st Qu.: 21.207   
## Median :0.000 Median : 41.76 Median : 44.632   
## Mean :0.488 Mean : 45.07 Mean : 45.011   
## 3rd Qu.:1.000 3rd Qu.: 66.39 3rd Qu.: 68.698   
## Max. :1.000 Max. :104.58 Max. :106.896

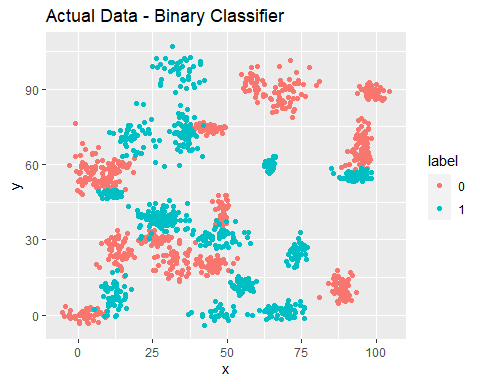
## a. Plot the data from each dataset using a scatter plot.

### Making Labels as Factor, since they are numbers

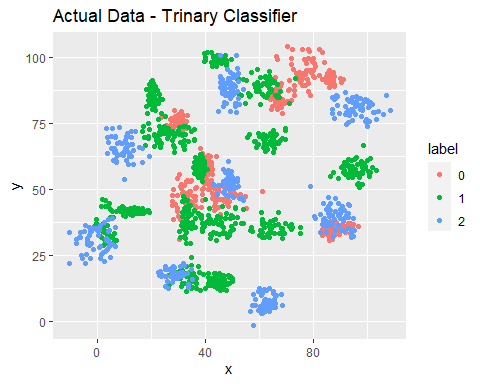
# Since label is number converting to factors, so that it becomes categorical  
binary\_classifier\_df$label <- as.factor(binary\_classifier\_df$label)  
trinary\_classifier\_df$label <- as.factor(trinary\_classifier\_df$label)

### Plotting

library(ggplot2)  
ggplot(data = binary\_classifier\_df, aes(y = y, x = x, color = label)) +   
 geom\_point() + ggtitle("Actual Data - Binary Classifier")



ggplot(data = trinary\_classifier\_df, aes(y = y, x = x, color = label)) +   
 geom\_point() + ggtitle("Actual Data - Trinary Classifier")



# 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:p1=(x1, y1) and p2=(x2,y2) is d

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.

## Splitting the Data into Train and Test dataframes

# Splitting the dataset for the model into train and test datasets.  
myData\_binary <- sample.split(binary\_classifier\_df$label, SplitRatio=0.8)  
train\_binary <- subset(binary\_classifier\_df, myData\_binary==TRUE)  
test\_binary <- subset(binary\_classifier\_df, myData\_binary==FALSE)  
  
myData\_trinary <- sample.split(trinary\_classifier\_df$label, SplitRatio=0.8)  
train\_trinary <- subset(trinary\_classifier\_df, myData\_trinary==TRUE)  
test\_trinary <- subset(trinary\_classifier\_df, myData\_trinary==FALSE)

## Accuracy function

# this function divides the correct predictions by total number of predictions that tell   
# us how accurate the model is.  
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) \* 100}  
#accuracy(tab)

## Fitting K nearest Model for Binary Classifier

Generating KNN Models for Binary Classifier Data and Displaying their accuracy from K=1 to K=25

library(class)  
  
K\_Binary\_Values <- c()  
Accuracy\_Binary <- c()  
error.rate.b <- c()  
  
# Running for multiple K Values  
for(i in 1:25){  
 knnmodel.b.i <- knn(train\_binary[2:3],test\_binary[2:3],k=i,cl=train\_binary$label)  
 table.b.i <- table(knnmodel.b.i,test\_binary$label)  
 accur = accuracy(table.b.i)  
 print(paste("Accuracy for Binary Classifier Data Model with K=", i ," is ", accuracy(table.b.i)))  
 K\_Binary\_Values <- c(K\_Binary\_Values, i)  
 Accuracy\_Binary<- c(Accuracy\_Binary, accur)  
 error.rate.b <- c(error.rate.b, mean(test\_binary$label != knnmodel.b.i))  
}

## [1] "Accuracy for Binary Classifier Data Model with K= 1 is 96.3210702341137"  
## [1] "Accuracy for Binary Classifier Data Model with K= 2 is 96.3210702341137"  
## [1] "Accuracy for Binary Classifier Data Model with K= 3 is 96.6555183946488"  
## [1] "Accuracy for Binary Classifier Data Model with K= 4 is 96.3210702341137"  
## [1] "Accuracy for Binary Classifier Data Model with K= 5 is 96.6555183946488"  
## [1] "Accuracy for Binary Classifier Data Model with K= 6 is 96.3210702341137"  
## [1] "Accuracy for Binary Classifier Data Model with K= 7 is 96.3210702341137"  
## [1] "Accuracy for Binary Classifier Data Model with K= 8 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 9 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 10 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 11 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 12 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 13 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 14 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 15 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 16 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 17 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 18 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 19 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 20 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 21 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 22 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 23 is 97.3244147157191"  
## [1] "Accuracy for Binary Classifier Data Model with K= 24 is 96.989966555184"  
## [1] "Accuracy for Binary Classifier Data Model with K= 25 is 97.3244147157191"

print(K\_Binary\_Values)

## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

print(Accuracy\_Binary)

## [1] 96.32107 96.32107 96.65552 96.32107 96.65552 96.32107 96.32107 96.98997  
## [9] 96.98997 96.98997 96.98997 96.98997 96.98997 96.98997 96.98997 96.98997  
## [17] 96.98997 96.98997 96.98997 96.98997 96.98997 96.98997 97.32441 96.98997  
## [25] 97.32441

print(error.rate.b\*100)

## [1] 3.678930 3.678930 3.344482 3.678930 3.344482 3.678930 3.678930 3.010033  
## [9] 3.010033 3.010033 3.010033 3.010033 3.010033 3.010033 3.010033 3.010033  
## [17] 3.010033 3.010033 3.010033 3.010033 3.010033 3.010033 2.675585 3.010033  
## [25] 2.675585

print(Accuracy\_Binary+error.rate.b\*100)

## [1] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100  
## [20] 100 100 100 100 100 100

accuracy\_binary\_df <- data.frame(K\_Binary\_Values, Accuracy\_Binary, error.rate.b\*100)  
accuracy\_binary\_df

## K\_Binary\_Values Accuracy\_Binary error.rate.b...100  
## 1 1 96.32107 3.678930  
## 2 2 96.32107 3.678930  
## 3 3 96.65552 3.344482  
## 4 4 96.32107 3.678930  
## 5 5 96.65552 3.344482  
## 6 6 96.32107 3.678930  
## 7 7 96.32107 3.678930  
## 8 8 96.98997 3.010033  
## 9 9 96.98997 3.010033  
## 10 10 96.98997 3.010033  
## 11 11 96.98997 3.010033  
## 12 12 96.98997 3.010033  
## 13 13 96.98997 3.010033  
## 14 14 96.98997 3.010033  
## 15 15 96.98997 3.010033  
## 16 16 96.98997 3.010033  
## 17 17 96.98997 3.010033  
## 18 18 96.98997 3.010033  
## 19 19 96.98997 3.010033  
## 20 20 96.98997 3.010033  
## 21 21 96.98997 3.010033  
## 22 22 96.98997 3.010033  
## 23 23 97.32441 2.675585  
## 24 24 96.98997 3.010033  
## 25 25 97.32441 2.675585

## Fitting K nearest Model for Trinary Classifier

Generating KNN Models for Trinary Classifier Data and Displaying their accuracy from K=1 to K=25

K\_Trinary\_Values <- c()  
Accuracy\_Trinary <- c()  
error.rate.t <- c()  
  
# Running for multiple K Values  
for(i in 1:25){  
 knnmodel.t.i <- knn(train\_trinary[2:3],test\_trinary[2:3],k=i,cl=train\_trinary$label)  
 table.t.i <- table(knnmodel.t.i,test\_trinary$label)  
 accur = accuracy(table.t.i)  
 print(paste("Accuracy for Trinary Classifier Data Model with K=", i ," is ", accur))  
 K\_Trinary\_Values <- c(K\_Trinary\_Values, i)  
 Accuracy\_Trinary<- c(Accuracy\_Trinary, accur)  
 error.rate.t <- c(error.rate.t, mean(test\_trinary$label != knnmodel.t.i))  
}

## [1] "Accuracy for Trinary Classifier Data Model with K= 1 is 87.8594249201278"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 2 is 86.9009584664537"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 3 is 90.4153354632588"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 4 is 88.8178913738019"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 5 is 89.1373801916933"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 6 is 89.1373801916933"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 7 is 89.776357827476"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 8 is 89.1373801916933"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 9 is 88.8178913738019"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 10 is 88.4984025559105"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 11 is 89.4568690095847"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 12 is 88.8178913738019"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 13 is 88.4984025559105"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 14 is 87.220447284345"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 15 is 88.1789137380192"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 16 is 88.1789137380192"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 17 is 87.8594249201278"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 18 is 87.5399361022364"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 19 is 88.4984025559105"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 20 is 89.4568690095847"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 21 is 88.1789137380192"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 22 is 88.1789137380192"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 23 is 87.8594249201278"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 24 is 88.1789137380192"  
## [1] "Accuracy for Trinary Classifier Data Model with K= 25 is 87.220447284345"

print(K\_Trinary\_Values)

## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

print(Accuracy\_Trinary)

## [1] 87.85942 86.90096 90.41534 88.81789 89.13738 89.13738 89.77636 89.13738  
## [9] 88.81789 88.49840 89.45687 88.81789 88.49840 87.22045 88.17891 88.17891  
## [17] 87.85942 87.53994 88.49840 89.45687 88.17891 88.17891 87.85942 88.17891  
## [25] 87.22045

print(error.rate.t\*100)

## [1] 12.140575 13.099042 9.584665 11.182109 10.862620 10.862620 10.223642  
## [8] 10.862620 11.182109 11.501597 10.543131 11.182109 11.501597 12.779553  
## [15] 11.821086 11.821086 12.140575 12.460064 11.501597 10.543131 11.821086  
## [22] 11.821086 12.140575 11.821086 12.779553

print(Accuracy\_Trinary+error.rate.t\*100)

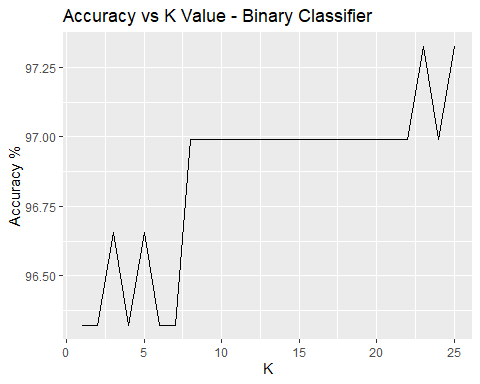
## [1] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100  
## [20] 100 100 100 100 100 100

accuracy\_trinary\_df <- data.frame(K\_Trinary\_Values, Accuracy\_Trinary, error.rate.t\*100)  
accuracy\_trinary\_df

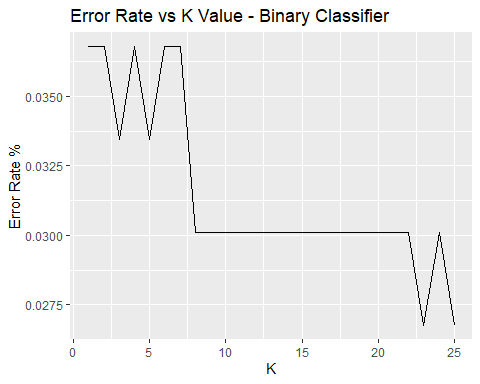
## K\_Trinary\_Values Accuracy\_Trinary error.rate.t...100  
## 1 1 87.85942 12.140575  
## 2 2 86.90096 13.099042  
## 3 3 90.41534 9.584665  
## 4 4 88.81789 11.182109  
## 5 5 89.13738 10.862620  
## 6 6 89.13738 10.862620  
## 7 7 89.77636 10.223642  
## 8 8 89.13738 10.862620  
## 9 9 88.81789 11.182109  
## 10 10 88.49840 11.501597  
## 11 11 89.45687 10.543131  
## 12 12 88.81789 11.182109  
## 13 13 88.49840 11.501597  
## 14 14 87.22045 12.779553  
## 15 15 88.17891 11.821086  
## 16 16 88.17891 11.821086  
## 17 17 87.85942 12.140575  
## 18 18 87.53994 12.460064  
## 19 19 88.49840 11.501597  
## 20 20 89.45687 10.543131  
## 21 21 88.17891 11.821086  
## 22 22 88.17891 11.821086  
## 23 23 87.85942 12.140575  
## 24 24 88.17891 11.821086  
## 25 25 87.22045 12.779553

## Plotting the K Values vs Accuracy

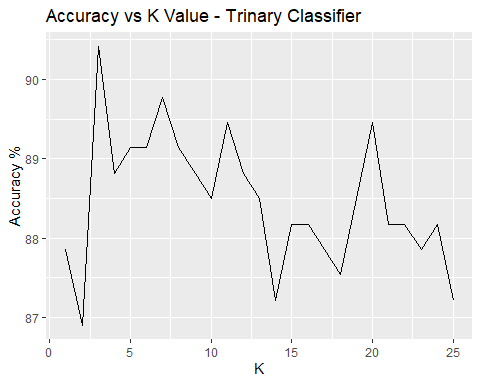
library(ggplot2)  
ggplot(data = accuracy\_binary\_df, aes(y = Accuracy\_Binary, x = K\_Binary\_Values)) +   
 geom\_line() + ggtitle("Accuracy vs K Value - Binary Classifier") +  
 ylab("Accuracy %") + xlab("K")



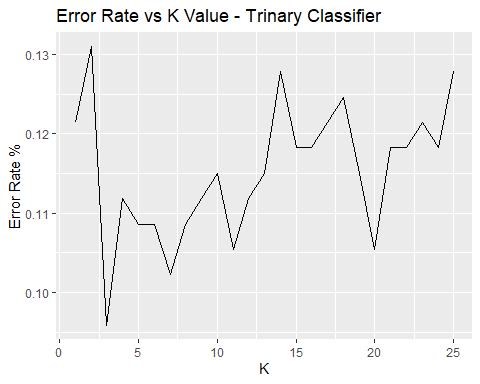
# Inverted Plot of Above  
ggplot(data = accuracy\_binary\_df, aes(y = error.rate.b, x = K\_Binary\_Values)) +   
 geom\_line() + ggtitle("Error Rate vs K Value - Binary Classifier") +  
 ylab("Error Rate %") + xlab("K")



ggplot(data = accuracy\_trinary\_df, aes(y = Accuracy\_Trinary, x = K\_Trinary\_Values)) +   
 geom\_line() + ggtitle("Accuracy vs K Value - Trinary Classifier") +  
 ylab("Accuracy %") + xlab("K")



# Inverted Plot of Above  
ggplot(data = accuracy\_trinary\_df, aes(y = error.rate.t, x = K\_Trinary\_Values)) +   
 geom\_line() + ggtitle("Error Rate vs K Value - Trinary Classifier") +  
 ylab("Error Rate %") + xlab("K")



# 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?

If we look at the Actual Data, its hard to divide the data(colored dots) with a line / linear classifier as the data is spread in different clusters. It is not possible to put the different binary values in opposite direction of this line, So linear classifier will not work with this kind of dataset.

# References

<https://towardsdatascience.com/k-nearest-neighbors-algorithm-with-examples-in-r-simply-explained-knn-1f2c88da405c> <https://blogs.oracle.com/datascience/introduction-to-k-means-clustering> <https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/> <https://www.datacamp.com/community/tutorials/k-means-clustering-r>