Exercise 15: Introduction to Machine Learning

Daniel Angel

February 14, 2021

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 directory to root of directory
setwd("C:/Users/Danny/Documents")

## Load the 'caTools' library
library(caTools)

## Load the 'data/binary-classifier-data.csv' as data frame
bin_class_df <- read.csv("data/binary-classifier-data.csv")
str(bin_class_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 ...
```

```r
## Load the 'data/trinary-classifier-data.csv' as data frame
tri_class_df <- read.csv("data/trinary-classifier-data.csv")
str(tri_class_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 ...
```

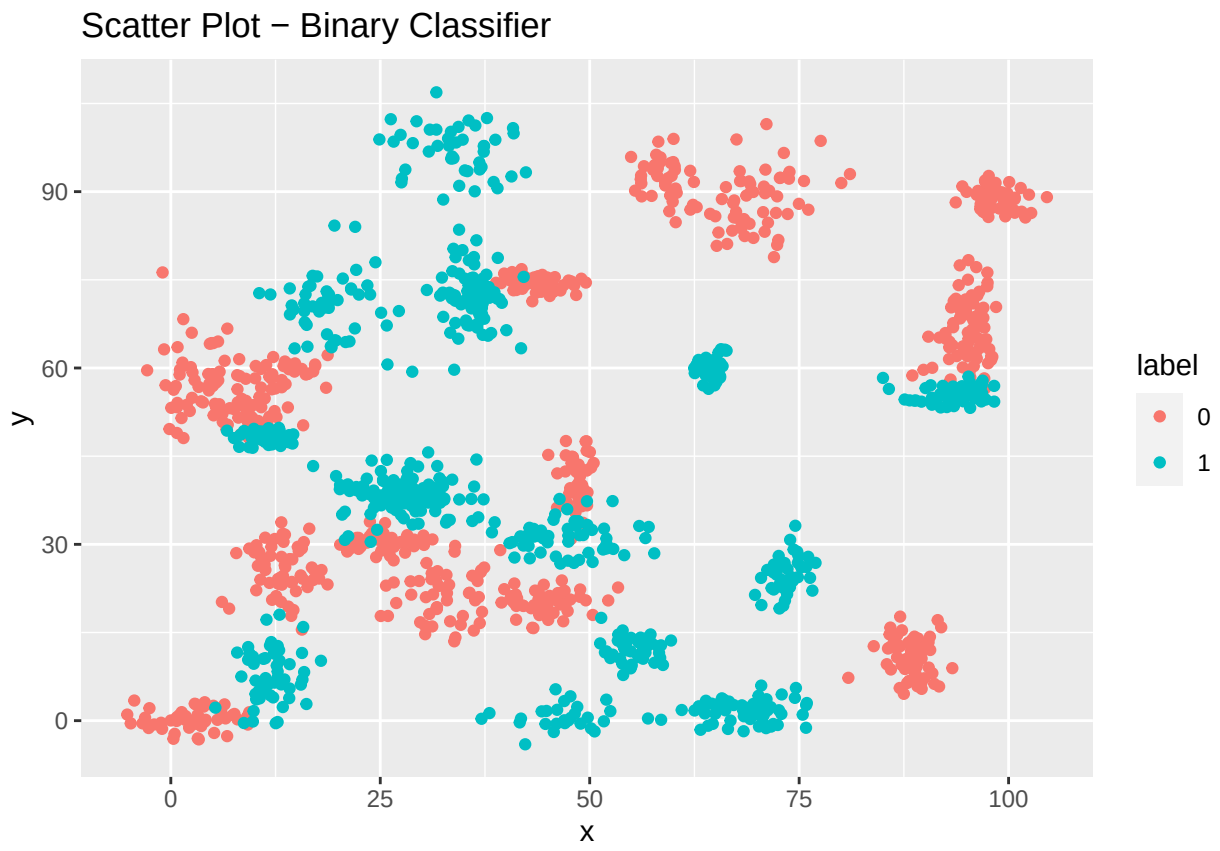
a. Plot the data from each dataset using a scatter plot.

Convert Labels to Factors

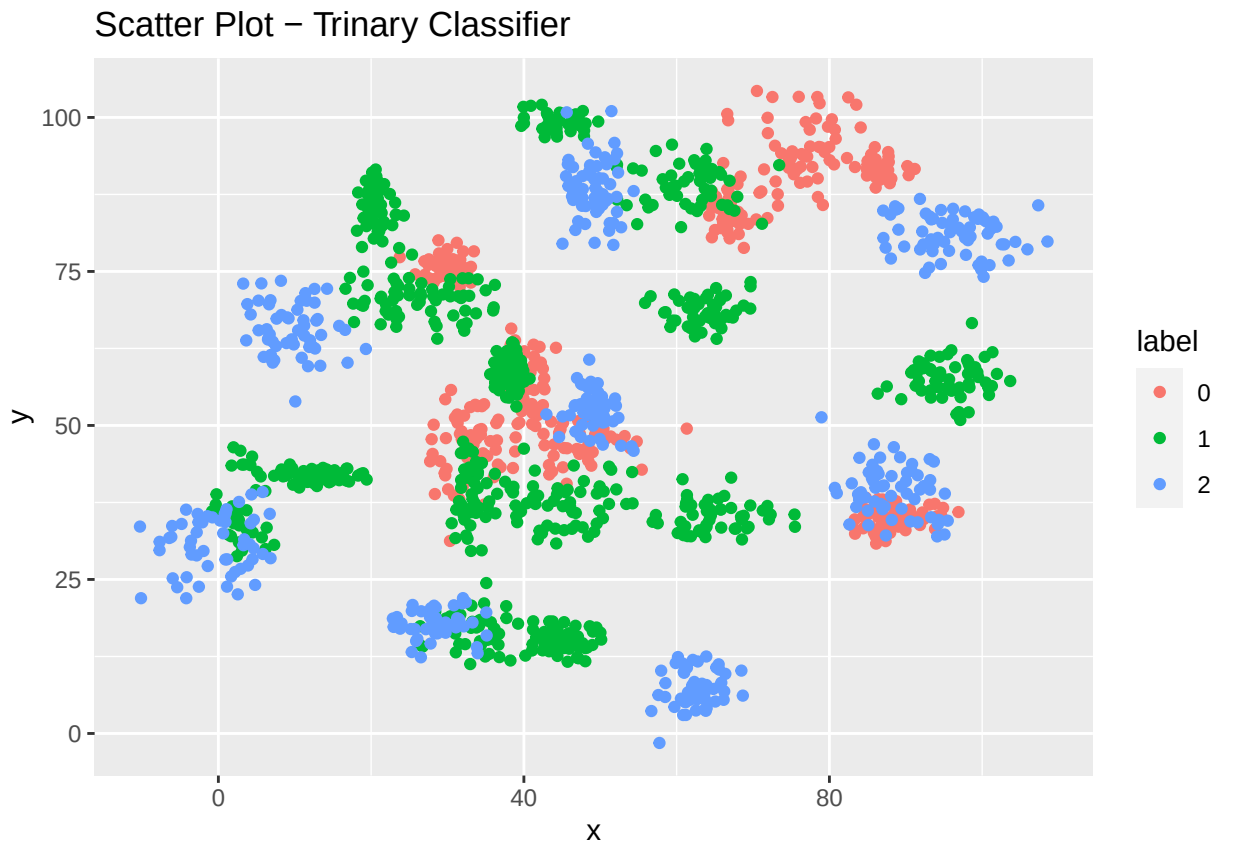
```r
# Since label is number convert to factors, so that it becomes categorical
bin_class_df$label <- as.factor(bin_class_df$label)
tri_class_df$label <- as.factor(tri_class_df$label)
```

Plotting

```r
ggplot(data = bin_class_df, aes(y = y, x = x, color = label)) +
  geom_point() + ggtitle("Scatter Plot - Binary Classifier")
```



```
ggplot(data = tri_class_df, aes(y = y, x = x, color = label)) +
  geom_point() + ggtitle("Scatter Plot - 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 derived from the Pythagorean Theorem

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

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 Training and Testing Subsets

```
# Splitting the dataset for the model into train and test datasets.
splits_bin <- sample.split(bin_class_df$label, SplitRatio=0.8)
trainer_bin <- subset(bin_class_df, splits_bin==TRUE)
tester_bin <- subset(bin_class_df, splits_bin==FALSE)

splits_tri <- sample.split(tri_class_df$label, SplitRatio=0.8)
trainer_tri <- subset(tri_class_df, splits_tri==TRUE)
tester_tri <- subset(tri_class_df, splits_tri==FALSE)
```

Accuracy function

```
# A function to divide the correct number of predictions by total predictions
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=3 to K=25

```
K_Binary_Values <- c()
Accuracy_Binary <- c()

# Running for multiple K Values
for(i in 3:25){
  knn_bi <- knn(trainer_bin[2:3],tester_bin[2:3],k=i,cl=trainer_bin$label)
  table_bi <- table(knn_bi,tester_bin$label)
  accur = accuracy(table_bi)
  K_Binary_Values <- c(K_Binary_Values, i)
  Accuracy_Binary<- c(Accuracy_Binary, accur)

}



acc_bin_df <- data.frame(K_Binary_Values, Accuracy_Binary)
acc_bin_df
```

```
##    K_Binary_Values Accuracy_Binary
## 1                3        98.32776
## 2                4        97.65886
## 3                5        98.66221
## 4                6        98.32776
## 5                7        98.32776
## 6                8        98.66221
## 7                9        97.65886
## 8               10        98.32776
## 9               11        98.32776
```

```
## 10              12        98.32776
## 11              13        97.99331
## 12              14        98.32776
## 13              15        97.99331
## 14              16        97.65886
## 15              17        97.65886
## 16              18        97.99331
## 17              19        97.99331
## 18              20        97.65886
## 19              21        97.99331
## 20              22        97.99331
## 21              23        97.99331
## 22              24        97.65886
## 23              25        97.99331
```

Fitting K nearest Model for Trinary Classifier

Generating KNN Models for Trinary Classifier Data and Displaying their accuracy from K=3 to K=25

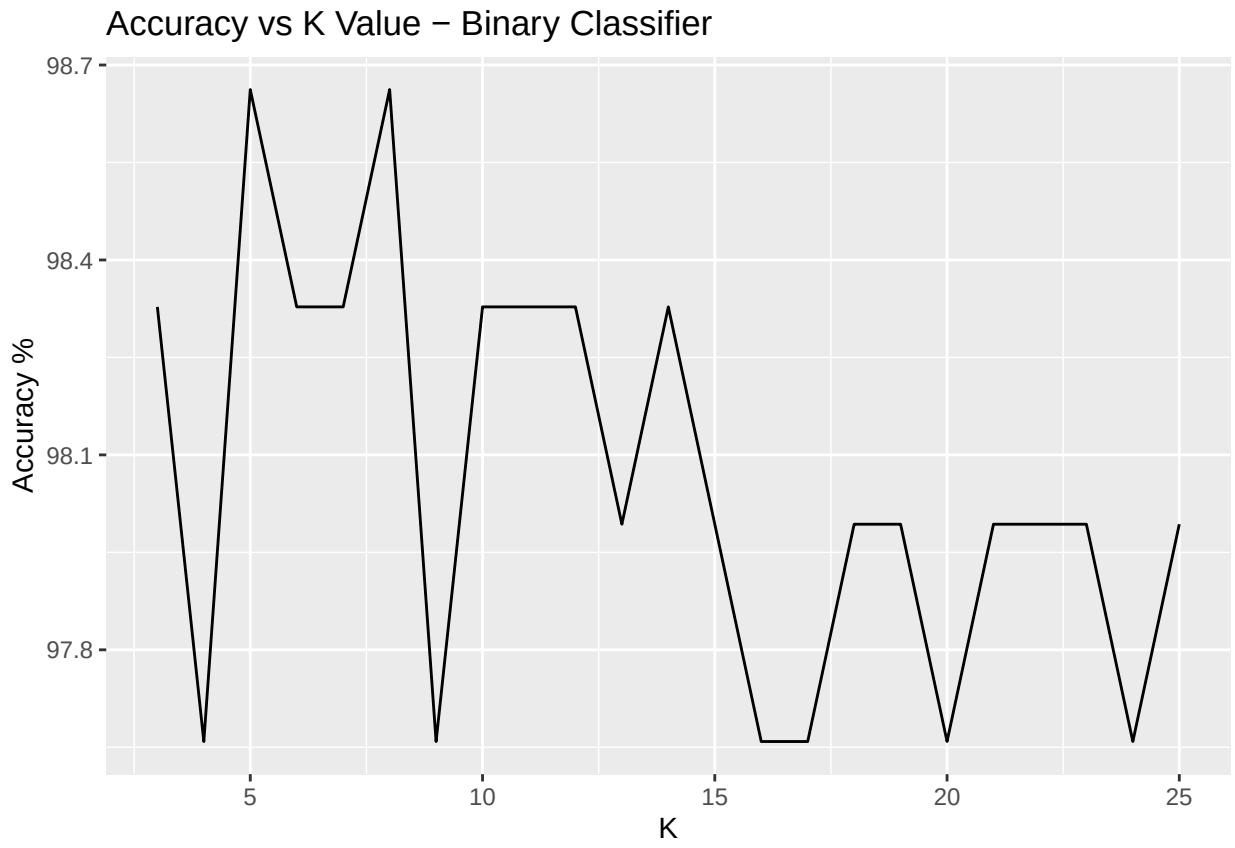
```
K_Trinary_Values <- c()
Accuracy_Trinary <- c()

# Running for multiple K Values
for(i in 3:25){
  knn_tri <- knn(trainer_tri[2:3],tester_tri[2:3],k=i,cl=trainer_tri$label)
  table_tri <- table(knn_tri,tester_tri$label)
  accur = accuracy(table_tri)
  K_Trinary_Values <- c(K_Trinary_Values, i)
  Accuracy_Trinary<- c(Accuracy_Trinary, accur)
}




acc_tri_df <- data.frame(K_Trinary_Values, Accuracy_Trinary)
acc_tri_df
```

```
##    K_Trinary_Values Accuracy_Trinary
## 1                 3         85.94249
## 2                 4         86.58147
## 3                 5         87.53994
## 4                 6         87.53994
## 5                 7         87.22045
## 6                 8         87.85942
## 7                 9         85.94249
## 8                10         84.66454
## 9                11         85.62300
## 10               12         86.26198
## 11               13         86.58147
## 12               14         84.66454
## 13               15         86.26198
## 14               16         86.90096
## 15               17         85.62300
```

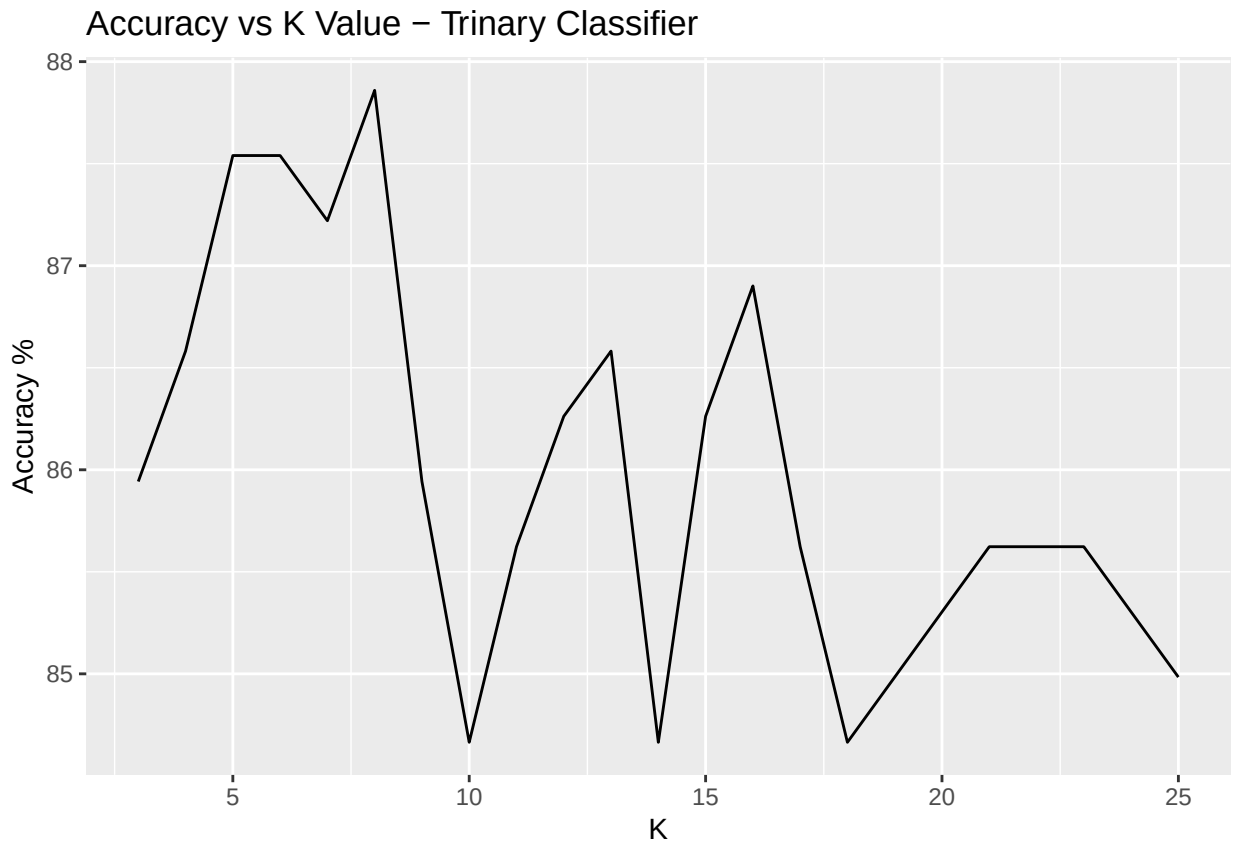
```
## 16              18          84.66454
## 17              19          84.98403
## 18              20          85.30351
## 19              21          85.62300
## 20              22          85.62300
## 21              23          85.62300
## 22              24          85.30351
## 23              25          84.98403
```

Plotting the K Values vs Accuracy

```
ggplot(data = acc_bin_df, aes(y = Accuracy_Binary, x = K_Binary_Values)) +
  geom_line() + ggtitle("Accuracy vs K Value - Binary Classifier") +
  ylab("Accuracy %") + xlab("K")
```




```
ggplot(data = acc_tri_df, aes(y = Accuracy_Trinary, x = K_Trinary_Values)) +
  geom_line() + ggtitle("Accuracy vs K Value - Trinary Classifier") +
  ylab("Accuracy %") + 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?

Looking at the scatter plots you can see that the labels are highly intermingled which would make dividing them simply with a single straight line difficult and in fact would prove nearly impossible.