Exercise 15: Introduction to Machine Learning

Kolekar, Ajit

November 1st, 2020

# Assignment description

**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.**

library('ggplot2')  
library('class')  
  
  
binary\_df <- read.csv("C:/Users/Shilp/Documents/GitHub/dsc520/data/binary-classifier-data.csv")  
  
  
# Split data to use 80% of data to train the model and 20% of data to test the model  
data\_split\_binary <- sample(1:nrow(binary\_df), 0.8 \* nrow(binary\_df))  
train\_binary <- binary\_df[data\_split\_binary,]  
test\_binary <- binary\_df[-data\_split\_binary,]  
  
nrow(binary\_df)

## [1] 1498

nrow(train\_binary)

## [1] 1198

nrow(test\_binary)

## [1] 300

trinary\_df <- read.csv("C:/Users/Shilp/Documents/GitHub/dsc520/data/trinary-classifier-data.csv")  
  
  
# Split data to use 80% of data to train the model and 20% of data to test the model  
data\_split\_trinary <- sample(1:nrow(trinary\_df), 0.8 \* nrow(trinary\_df))  
train\_trinary <- trinary\_df[data\_split\_trinary,]  
test\_trinary <- trinary\_df[-data\_split\_trinary,]  
  
nrow(trinary\_df)

## [1] 1568

nrow(train\_trinary)

## [1] 1254

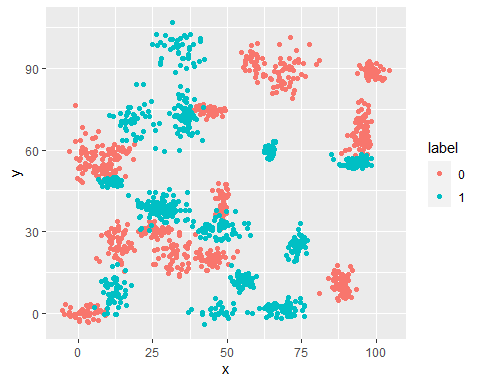
nrow(test\_trinary)

## [1] 314

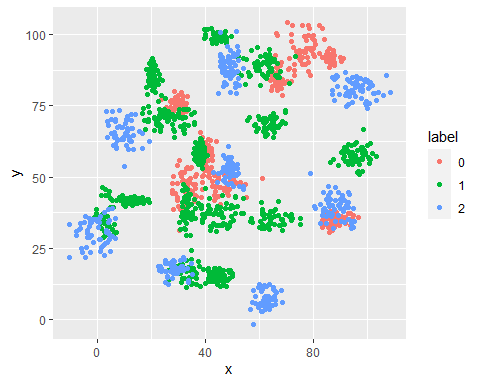
## Question a

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

binary\_factor\_df <- binary\_df  
binary\_factor\_df$label <- as.factor(binary\_factor\_df$label)  
ggplot(binary\_factor\_df, aes(x=x, y=y, color=label)) + geom\_point()



trinary\_factor\_df <- trinary\_df  
trinary\_factor\_df$label <- as.factor(trinary\_factor\_df$label)  
ggplot(trinary\_factor\_df, aes(x=x, y=y, color=label)) + geom\_point()



## Question 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.**

**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.**

Euclidean Distance:

# install.packages('TSdist')  
library('TSdist')

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

## Loading required package: proxy

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

##   
## Attaching package: 'proxy'

## The following objects are masked from 'package:stats':  
##   
## as.dist, dist

## The following object is masked from 'package:base':  
##   
## as.matrix

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

## Loaded TSdist v3.7. See ?TSdist for help, citation("TSdist") for use in publication.

EuclideanDistance(binary\_df$x,binary\_df$y)

## [1] 1411.959

EuclideanDistance(trinary\_df$x,trinary\_df$y)

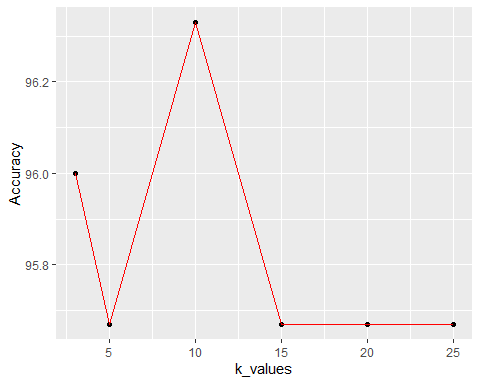
## [1] 1357.734

Binary dataset:

binary\_glm <- glm(label ~ x + y, data=binary\_df, family = binomial)  
  
# extract 1st column of train dataset because it will be used as 'cl' argument in knn function.  
target\_category <- binary\_df[data\_split\_binary,1]  
  
# extract 1st column if test dataset to measure the accuracy  
test\_category <- binary\_df[-data\_split\_binary,1]  
  
k\_values<- c(3, 5, 10, 15, 20, 25)  
Accuracy <- NULL  
  
for (i in 1:length(k\_values))  
{  
 test\_pred <- knn(train\_binary,test\_binary,cl=target\_category,k=k\_values[i])  
 confmatrix <- table(test\_category,test\_pred)  
 accuracy <- (confmatrix[[1,1]] + confmatrix[[2,2]]) / sum(confmatrix)  
 Accuracy <- c(Accuracy, round((accuracy \* 100), digits=2))  
}   
  
binary\_results\_df <- data.frame(k\_values, Accuracy)  
binary\_results\_df

## k\_values Accuracy  
## 1 3 96.00  
## 2 5 95.67  
## 3 10 96.33  
## 4 15 95.67  
## 5 20 95.67  
## 6 25 95.67

ggplot(binary\_results\_df, aes(x=k\_values, y=Accuracy)) + geom\_point() + geom\_line(colour="red")

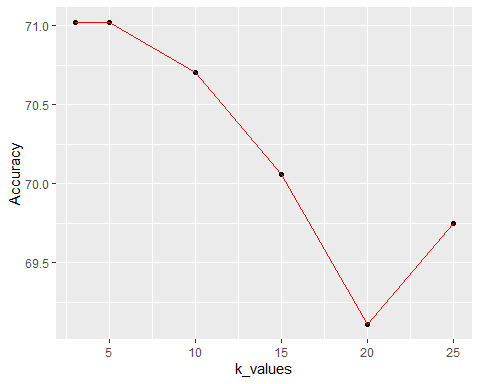


Trinary dataset:

trinary\_glm <- glm(label ~ x + y, data=trinary\_df, family = poisson)  
  
# extract 1st column of train dataset because it will be used as 'cl' argument in knn function.  
target\_category <- trinary\_df[data\_split\_trinary,1]  
  
# extract 1st column if test dataset to measure the accuracy  
test\_category <- trinary\_df[-data\_split\_trinary,1]  
  
k\_values<- c(3, 5, 10, 15, 20, 25)  
Accuracy <- NULL  
  
for (i in 1:length(k\_values))  
{  
 test\_pred <- knn(train\_trinary,test\_trinary,cl=target\_category,k=k\_values[i])  
 confmatrix <- table(test\_category,test\_pred)  
 accuracy <- (confmatrix[[1,1]] + confmatrix[[2,2]]) / sum(confmatrix)  
 Accuracy <- c(Accuracy, round((accuracy \* 100), digits=2))  
}   
  
trinary\_results\_df <- data.frame(k\_values, Accuracy)  
trinary\_results\_df

## k\_values Accuracy  
## 1 3 71.02  
## 2 5 71.02  
## 3 10 70.70  
## 4 15 70.06  
## 5 20 69.11  
## 6 25 69.75

ggplot(trinary\_results\_df, aes(x=k\_values, y=Accuracy)) + geom\_point() + geom\_line(colour="red")



## Question c

**Looking back at the plots of the data, do you think a linear classifier would work well on these datasets?**

By looking at the plots of the data, linear classifier would not work well with both binary data set and trinary data set, since the data points are scattered all over. It is not possible to draw a straight line through the data points that would correctly represent the data. The data points are in clusters, so K-Nearest Neighbors classifier would work well for these data sets.