```
---
title: "ASSIGNMENT 8.2_MachineLearning"
author: "Bhushan Suryawanshi"
date: '2020-07-14'

---
```

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.

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

```r
library("ggplot2")
library("caTools")
```

```
## Warning: package 'caTools' was built under R version 4.0.2
```

```r
library("class")
```

```
## Warning: package 'class' was built under R version 4.0.2
```

```r
library("caret")
```

```
## Warning: package 'caret' was built under R version 4.0.2
```

```
## Loading required package: lattice
```

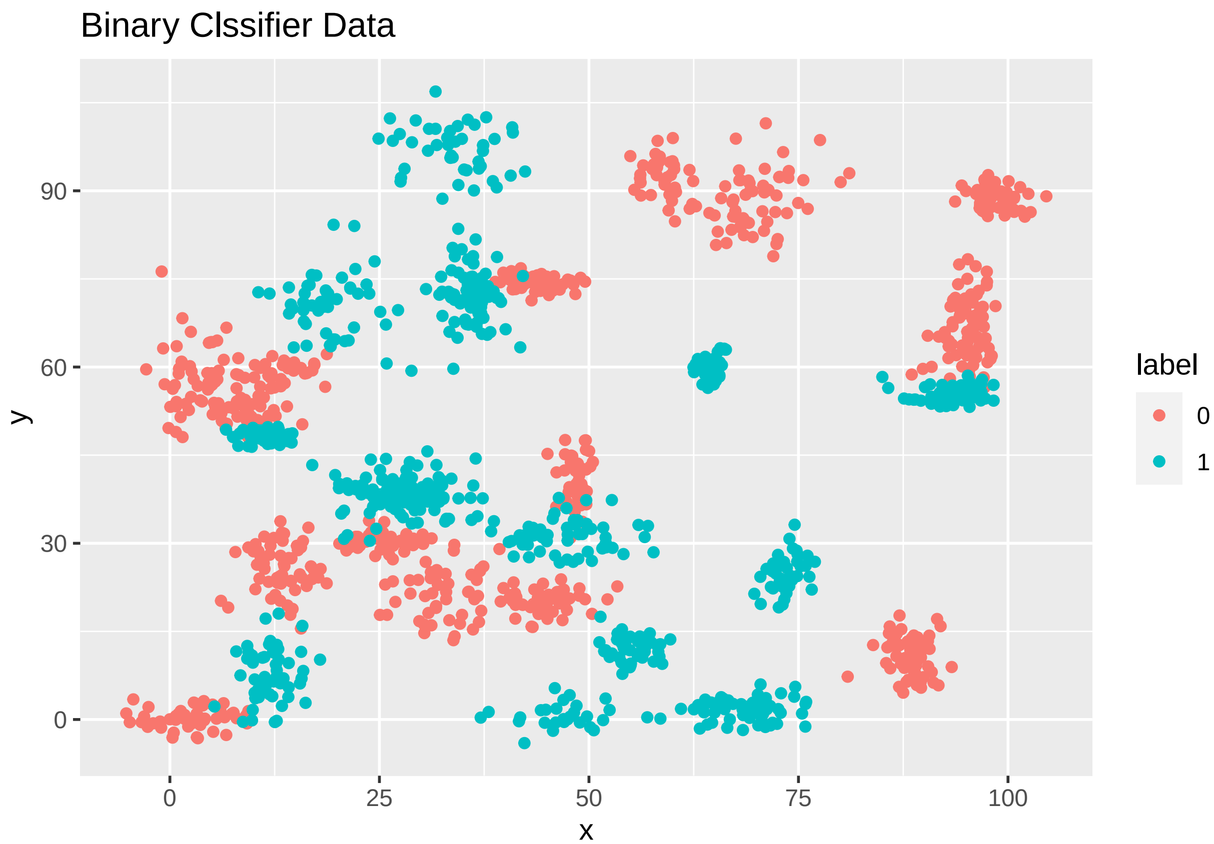
```r
binary_classifier_df <- read.csv("binary-classifier-data.csv")
head(binary_classifier_df)
```

```
##   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
```

```r
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
```

```r
binary_classifier_df$label <- as.factor(binary_classifier_df$label)

ggplot(binary_classifier_df, aes(x=x, y=y, color=label)) + geom_point() + ggtitle('Binary Clssifier Data
```



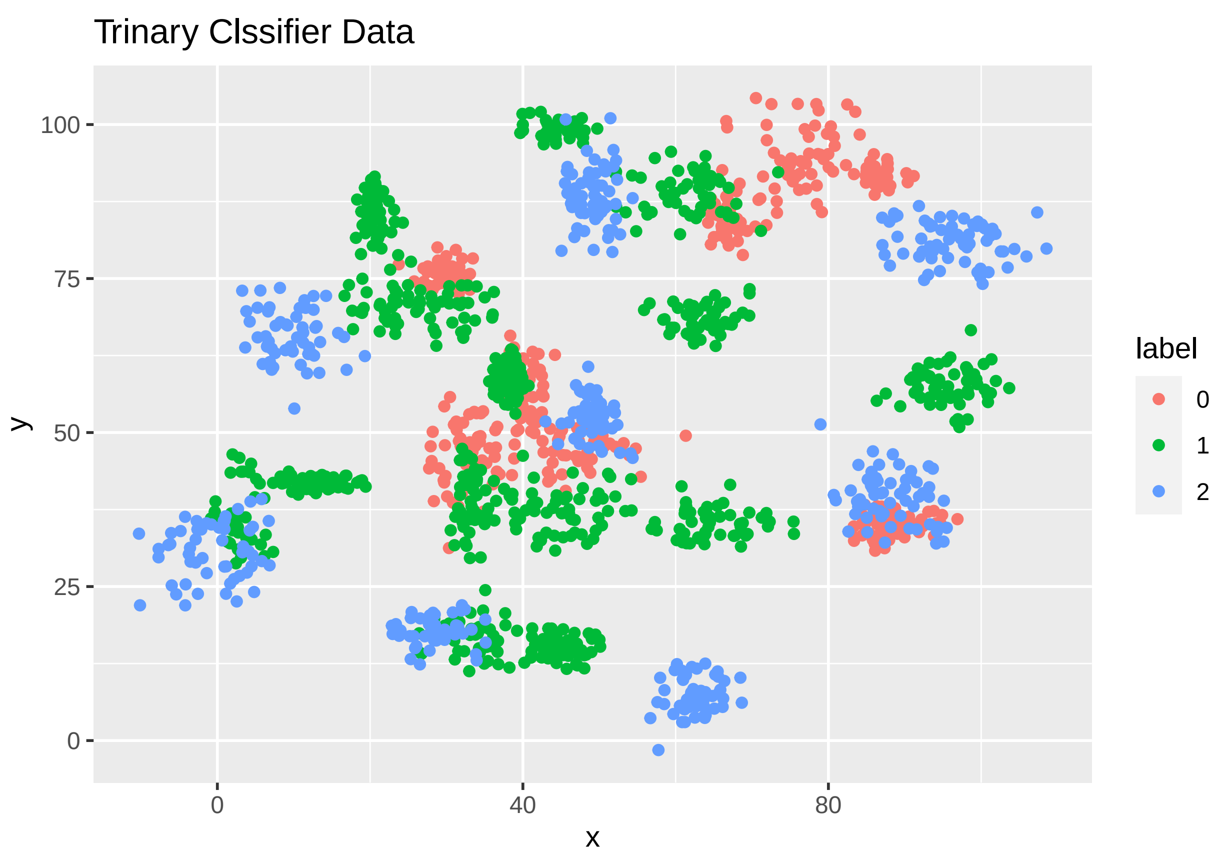

```r
trinary_classifier_df <- read.csv("trinary-classifier-data.csv")
head(trinary_classifier_df)
```

```
##   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
```

```r
summary(trinary_classifier_df)
```

```
##      label             x                y          
##  Min.   :0.000   Min.   :-10.26   Min.   : -1.541  
##  1st Qu.:0.000   1st Qu.: 31.15   1st Qu.: 35.906  
##  Median :1.000   Median : 45.59   Median : 55.073  
##  Mean   :1.037   Mean   : 48.86   Mean   : 55.282  
##  3rd Qu.:2.000   3rd Qu.: 66.27   3rd Qu.: 77.403  
##  Max.   :2.000   Max.   :108.56   Max.   :104.293
```

```r
trinary_classifier_df$label <- as.factor(trinary_classifier_df$label)

ggplot(trinary_classifier_df, aes(x=x, y=y, color=label)) + geom_point() + ggtitle('Trinary Clssifier Da
```



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:

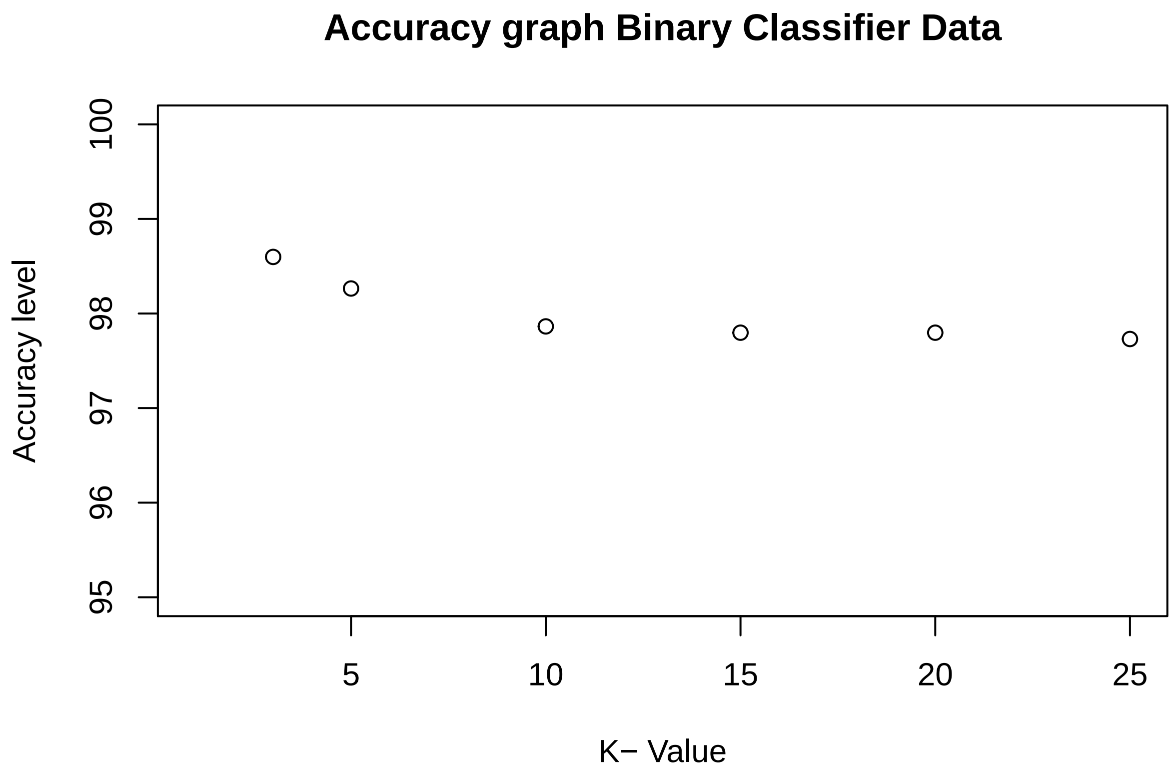
Fitting a model is when you use the input data to create a predictive model. There are various metrics you can use to determine how well your model fits the data. You will learn more about these metrics in later lessons. For this problem, you will focus on a single metric; accuracy. Accuracy is simply 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.

```
set.seed(42)
binary_split<-sample.split(binary_classifier_df, SplitRatio=0.80)
trinary_split<-sample.split(trinary_classifier_df, SplitRatio=0.80)

binary_train <- subset(binary_classifier_df, binary_split="TRUE")
binary_test <- subset(binary_classifier_df, binary_split="FALSE")

trinary_train <- subset(trinary_classifier_df, trinary_split="TRUE")
trinary_test <- subset(trinary_classifier_df, trinary_split="FALSE")

list_of_k <- list(3,5,10,15,20,25)

accuracy_binary = 1

for (i in list_of_k) {

  knn_binary <- knn(train=binary_train, test=binary_test, cl=binary_train$label, k=i )
  accuracy_binary[i] <- 100 * sum(binary_test$label == knn_binary)/nrow(binary_test)


}

accuracy_binary
```

```
##  [1]  1.00000       NA 98.59813       NA 98.26435       NA       NA       NA
##  [9]       NA 97.86382       NA       NA       NA       NA 97.79706       NA
## [17]       NA       NA       NA 97.79706       NA       NA       NA       NA
## [25] 97.73031
```

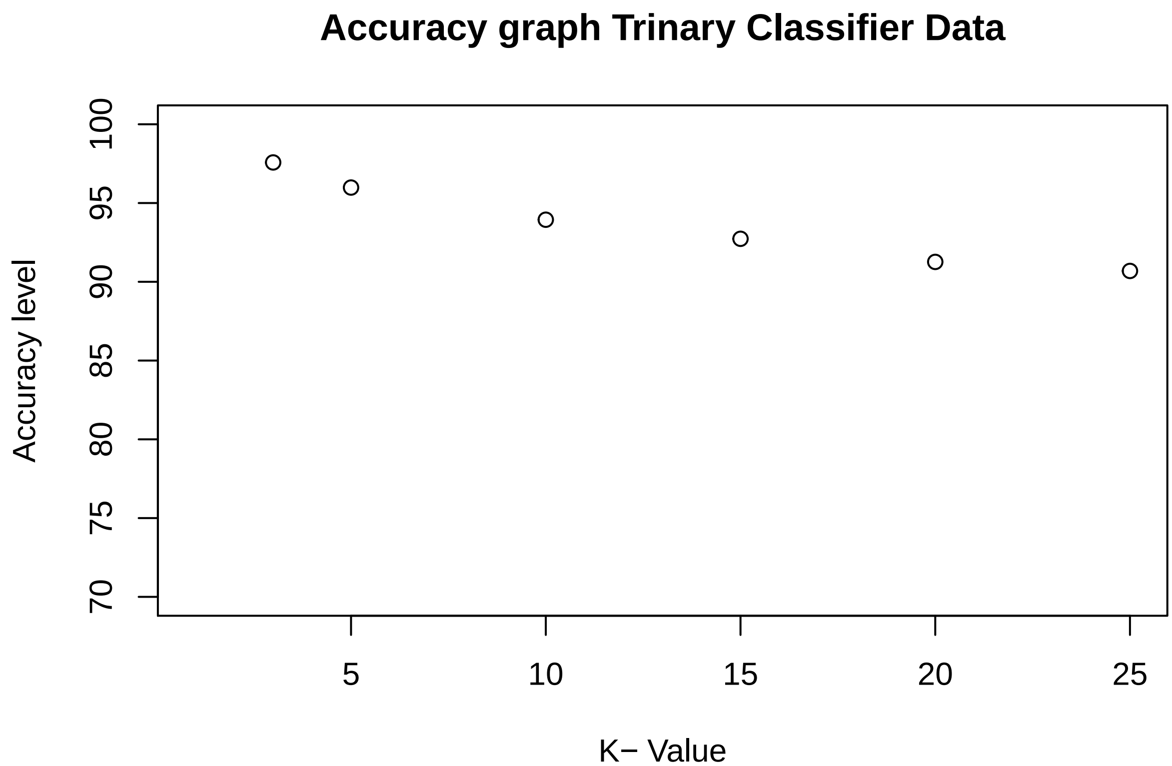
```
plot(accuracy_binary, type="b", xlab="K- Value",ylab="Accuracy level", ylim = c(95,100), main = "Accura
```




```
accuracy_trinary = 1
for (i in list_of_k) {

  knn_trinary <- knn(train=trinary_train, test=trinary_test, cl=trinary_train$label, k=i )
  accuracy_trinary[i] <- 100 * sum(trinary_test$label == knn_trinary)/nrow(trinary_test)

}

accuracy_trinary
```

```
##  [1]  1.00000       NA 97.57653       NA 95.98214       NA       NA       NA
##  [9]       NA 93.94133       NA       NA       NA       NA 92.72959       NA
## [17]       NA       NA       NA 91.26276       NA       NA       NA       NA
## [25] 90.68878
```

```
plot(accuracy_trinary, type="b", xlab="K- Value",ylab="Accuracy level", ylim = c(70,100), main = "Accura
```



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?

Answer Looking at the scattered plot of the data we can see its so wide spread. Also we have seen as value of K increases the accuracy is dropping. Which means instead of using k-means if we use algorithm where we can use classification instead of clustering will help. Linear classifier can be helpful here because instead of forming clusters linear classifier will form a classification boundary based on the characteristics (independent variables).