ASSIGNMENT 11.2.1

2022-06-04

Installing Packages

install.packages("e1071") install.packages("caTools") install.packages("class") install.packages("tidymodels") install.packages("gridExtra") install.packages("kknn") —

Loading package

library(e1071) library(caTools) library(class) library(tidyverse) library(tidymodels) library(gridExtra) library(plyr) library(ggplot2) library(kknn) —

```
# Loading package
library(e1071)
library(caTools)
library(class)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 0.2.0 --
```

```
## v broom        0.7.12     v rsample      0.1.1
## v dials        0.1.1      v tune         0.2.0
## v infer        1.0.0      v workflows    0.2.6
## v modeldata    0.1.1      v workflowsets 0.2.1
## v parsnip      0.2.1      v yardstick    0.0.9
## v recipes      0.2.0
```

```
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard()       masks purrr::discard()
## x dplyr::filter()         masks stats::filter()
## x recipes::fixed()        masks stringr::fixed()
## x dplyr::lag()            masks stats::lag()
## x rsample::permutations() masks e1071::permutations()
## x yardstick::spec()       masks readr::spec()
## x recipes::step()         masks stats::step()
## x tune::tune()            masks parsnip::tune(), e1071::tune()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(plyr)
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:purrr':
##
##     compact
```

```
library(ggplot2)
library(kknn)

set.seed(123)


## Load the `data/binary-classifier-data` to
bin_class_df <- read.csv("/Users/siddharthabhaumik/Documents/GitHub/dsc520/data/binary-classifier-data.

## Viewing Sample data
head(bin_class_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
summary(bin_class_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
```

```
## Check Data Structure of the object
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 ...
```

```
## Load the `data/trinary-classifier-data` to
tri_class_df <- read.csv("/Users/siddharthabhaumik/Documents/GitHub/dsc520/data/trinary-classifier-data

## Viewing Sample data
head(tri_class_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
```

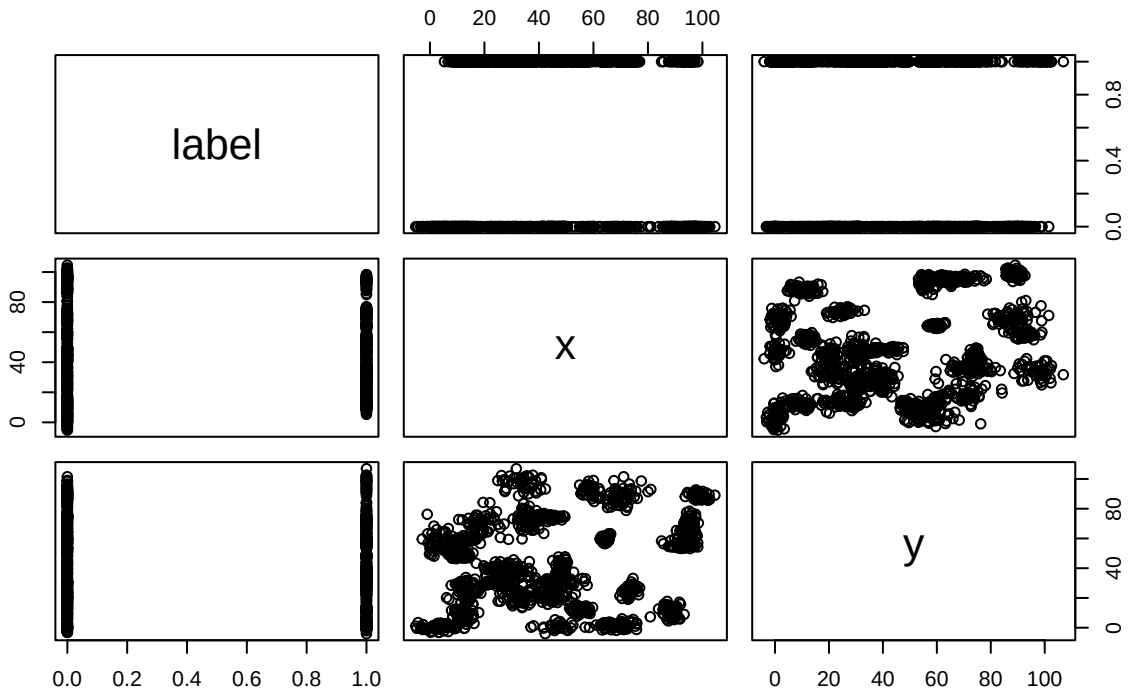
```
## summary
summary(tri_class_df)
```

```
##      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
```

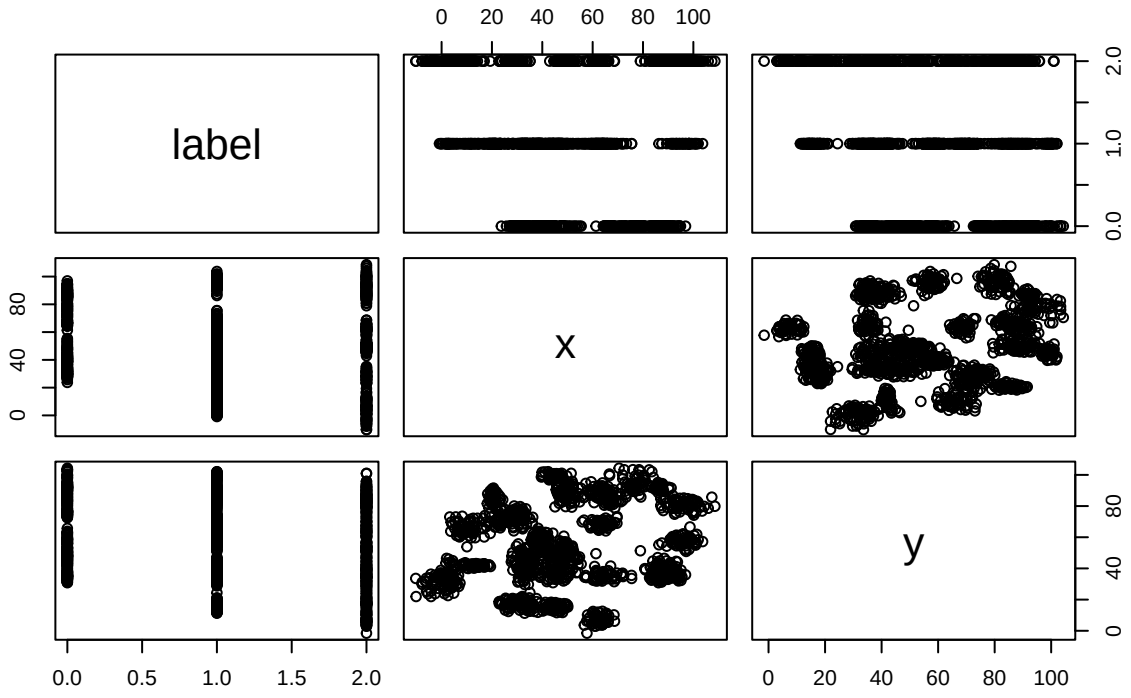
```
## Check Data Structure of the object
str(tri_class_df)
```

```
## '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 ...
```

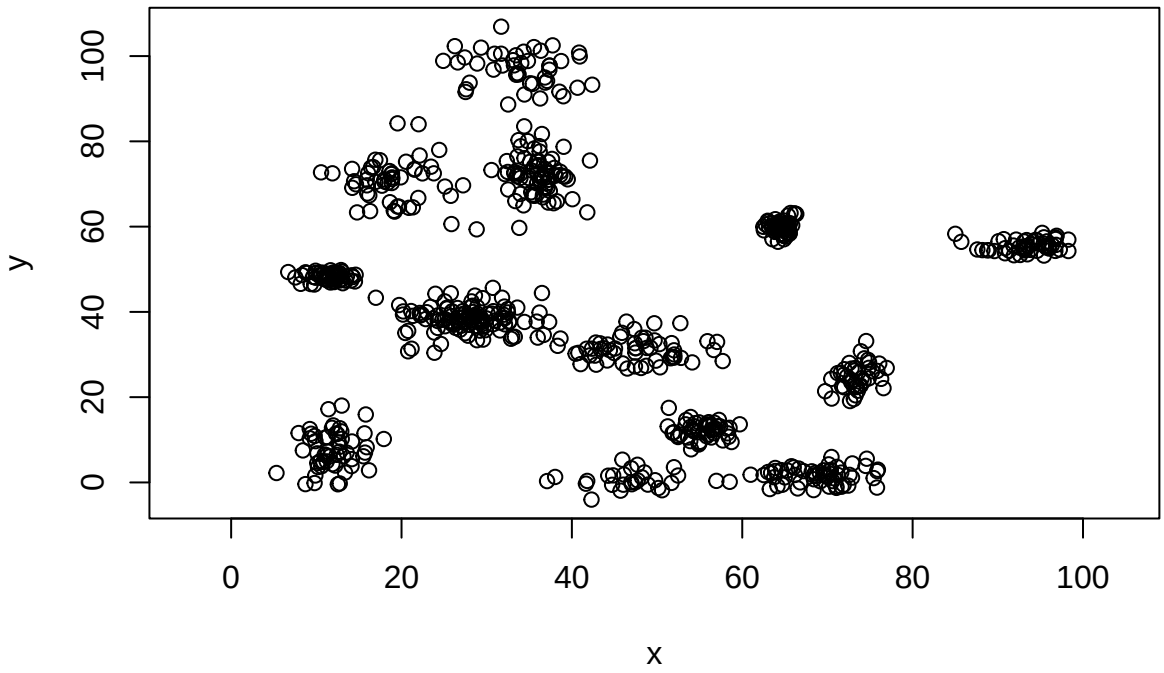
```
# Plot the data from each dataset using a scatter plot.

## Since we have more than two variables and we want to find the correlation
## between one variable versus the remaining ones,I use the scatterplot matrix.

pairs(~label+x+y, data = bin_class_df, main = "Scatterplot Matrix for Binary Classifier")
```



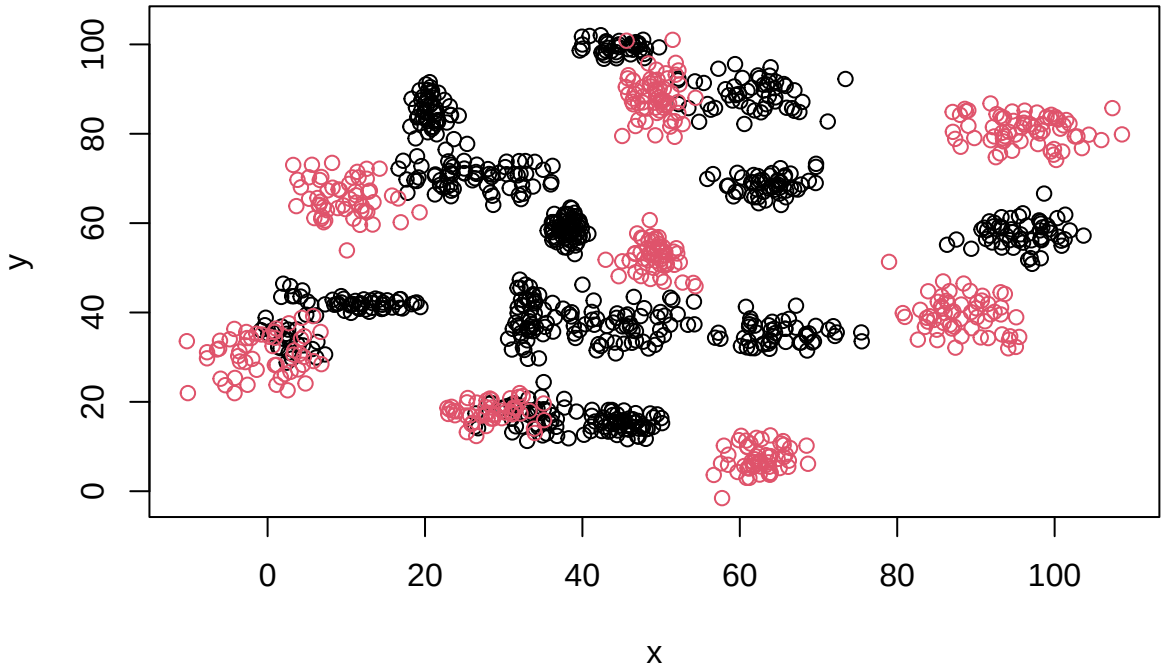

```
pairs(~label+x+y, data = tri_class_df, main = "Scatterplot Matrix for Trinary Classifier")
```




```
## Simple plot for binary classifier
with(bin_class_df,{plot(x,y,col=label)})
```




```
## Simple plot for trinary classifier
with(tri_class_df,{plot(x,y,col=label)})
```




```
# Splitting binary classifier data into train and test data
split <- sample.split(bin_class_df, SplitRatio = 0.7)
train_bin_class <- subset(bin_class_df, split == "TRUE")
test_bin_class <- subset(bin_class_df, split == "FALSE")

# Feature Scaling
train_scale <- scale(train_bin_class[, 1:3])
test_scale <- scale(test_bin_class[, 1:3])
```

```
# Fitting KNN Model for binary_classifier dataset
bin_class_knn <- knn(train = train_scale,
                     test = test_scale,
                     cl = train_bin_class$label,
                     k = 1)
bin_class_knn
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [149] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [186] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [223] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
## [260] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [297] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [334] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [371] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [408] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [445] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [482] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## Levels: 0 1
```

```
# Confusion Matrix
bin_class_cm <- table(test_bin_class$label, bin_class_knn)

# Model Evaluation - Choosing K
# Calculate out of Sample error
misClassError <- mean(bin_class_knn != test_bin_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# K = 3
bin_class_knn <- knn(train = train_scale,
                     test = test_scale,
                     cl = train_bin_class$label,
                     k = 3)
misClassError <- mean(bin_class_knn != test_bin_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
test_accuracy <- paste('Accuracy =', 1-misClassError)

# K = 5
bin_class_knn <- knn(train = train_scale,
                     test = test_scale,
                     cl = train_bin_class$label,
                     k = 5)
misClassError <- mean(bin_class_knn != test_bin_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# K = 10
bin_class_knn <- knn(train = train_scale,
                     test = test_scale,
                     cl = train_bin_class$label,
                     k = 10)
misClassError <- mean(bin_class_knn != test_bin_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# K = 15
bin_class_knn <- knn(train = train_scale,
                     test = test_scale,
                     cl = train_bin_class$label,
                     k = 15)
misClassError <- mean(bin_class_knn != test_bin_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# K = 20
bin_class_knn <- knn(train = train_scale,
                     test = test_scale,
                     cl = train_bin_class$label,
                     k = 20)
misClassError <- mean(bin_class_knn != test_bin_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# K = 25
bin_class_knn <- knn(train = train_scale,
                     test = test_scale,
                     cl = train_bin_class$label,
                     k = 25)
misClassError <- mean(bin_class_knn != test_bin_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# Splitting Trinary classifier data into train and test data
split <- sample.split(tri_class_df, SplitRatio = 0.7)
train_tri_class <- subset(tri_class_df, split == "TRUE")
test_tri_class <- subset(tri_class_df, split == "FALSE")

# Feature Scaling
train_tri_scale <- scale(train_tri_class[, 1:3])
test_tri_scale <- scale(test_tri_class[, 1:3])

# Fitting KNN Model for trinary_classifier dataset
tri_class_knn <- knn(train = train_tri_scale,
                     test = test_tri_scale,
                     cl = train_tri_class$label,
                     k = 1)
tri_class_knn
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [260] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [297] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [334] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [371] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [408] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [445] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [482] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [519] 2 2 2 2 2
## Levels: 0 1 2
```

```
# Confusion Matrix
tri_class_cm <- table(test_tri_class$label, tri_class_knn)

tri_class_cm
```

```
##    tri_class_knn
##       0   1   2
##   0 132   0   0
##   1   0 240   0
##   2   0   0 151
```

```
# Model Evaluation - Choosing K
# Calculate out of Sample error
misClassError <- mean(tri_class_knn != test_tri_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# K = 3
tri_class_knn <- knn(train = train_tri_scale,
                     test = test_tri_scale,
                     cl = train_tri_class$label,
                     k = 3)
misClassError <- mean(tri_class_knn != test_tri_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# K = 5
tri_class_knn <- knn(train = train_tri_scale,
                     test = test_tri_scale,
                     cl = train_tri_class$label,
                     k = 5)
misClassError <- mean(tri_class_knn != test_tri_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# K = 10
tri_class_knn <- knn(train = train_tri_scale,
                     test = test_tri_scale,
                     cl = train_tri_class$label,
```

```
                      k = 10)
misClassError <- mean(tri_class_knn != test_tri_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# K = 15
tri_class_knn <- knn(train = train_tri_scale,
                     test = test_tri_scale,
                     cl = train_tri_class$label,
                     k = 15)
misClassError <- mean(tri_class_knn != test_tri_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# K = 20
tri_class_knn <- knn(train = train_tri_scale,
                     test = test_tri_scale,
                     cl = train_tri_class$label,
                     k = 20)
misClassError <- mean(tri_class_knn != test_tri_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# K = 25
tri_class_knn <- knn(train = train_tri_scale,
                     test = test_tri_scale,
                     cl = train_tri_class$label,
                     k = 25)
misClassError <- mean(tri_class_knn != test_tri_class$label)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 1"
```

```
# Looking back at the plots of the data, do you think a linear classifier would work well on these data

## No, it doesn't look like from the results. The current model accuracy is 100%
## whereas for linear model (done previous week) it was merely 41% accuracy.
## Logistic regression goes very well on high dimensional data sets with a lot of training points,
## but if your data is not linearly separable the algorithm won't work well.

# How does the accuracy of your logistic regression classifier from last week compare?  Why is the accu
## There is a big difference in accuracy between both methods.
## The current method gives 100% accuracy whereas last weeks gives only 41% accuracy.
```