DSC_520_week10_Assignment02

Reenie Christudass

2022-08-11

Contents

Load libraries as needed

```r
if(!require('factoextra')) {
  install.packages("factoextra", repos="http://cran.us.r-project.org")
  library('factoextra')
}
```

```
## Loading required package: factoextra

## Warning: package ’factoextra’ was built under R version 4.2.1

## Loading required package: ggplot2

## Warning: package ’ggplot2’ was built under R version 4.2.1

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
if(!require('cluster')) {
  install.packages("cluster", repos="http://cran.us.r-project.org")
  library('cluster')
}
```

```
## Loading required package: cluster

## Warning: package ’cluster’ was built under R version 4.2.1
```

```r
if(!require('NbClust')) {
  install.packages("NbClust", repos="http://cran.us.r-project.org")
  library('NbClust')
}
```

```
## Loading required package: NbClust
```

```r
if(!require('e1071')) {
  install.packages("e1071", repos="http://cran.us.r-project.org")
  library('e1071')
}
```

```
## Loading required package: e1071

## Warning: package ’e1071’ was built under R version 4.2.1
```

```r
if(!require('caTools')) {
  install.packages("caTools", repos="http://cran.us.r-project.org")
  library('caTools')
}
```

```
## Loading required package: caTools

## Warning: package ’caTools’ was built under R version 4.2.1
```

```
if(!require('class')) {
  install.packages("class", repos="http://cran.us.r-project.org")
  library('class')
}
```

```
## Loading required package: class
```

```
## Warning: package 'class' was built under R version 4.2.1
```

```
if(!require('dplyr')) {
  install.packages("dplyr", repos="http://cran.us.r-project.org")
  library('dplyr')
}
```

```
## Loading required package: dplyr
```

```
## Warning: package 'dplyr' was built under R version 4.2.1
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
rm(list = ls())
## Set the working directory to the root of your DSC 520 directory
setwd("C:/Users/chris/dsc520/data")
```

Read the binary-classifier-data

```
df_binary <- read.csv("C:/Users/chris/dsc520/data/binary-classifier-data.csv")
head(df_binary)
```

```
##   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
```

Preprocess the dataset

```
binary.new<- df_binary[,c(2, 3)]
binary.class<- df_binary[,c(1)]
```

Normalize the dataset

```
normalize <- function(x){
  return ((x-min(x))/(max(x)-min(x)))
}

binary.new$x<- normalize(binary.new$x)
binary.new$y<- normalize(binary.new$y)

head(binary.new)
```

```
##           x         y
## 1 0.6930933 0.7861557
## 2 0.7303243 0.8290011
## 3 0.7194983 0.8675355
## 4 0.6523084 0.7674851
## 5 0.6765989 0.7984207
## 6 0.7054045 0.8150698
```

Apply k-means clustering algorithm for 3 centroids

```
result<- kmeans(binary.new,3) #aplly k-means algorithm with no. of centroids(k)=3
result$size # gives no. of records in each cluster
```

```
## [1] 595 393 510
```

```
result$centers # gives value of cluster center datapoint value(3 centers for k=3)
```

```
##           x         y
## 1 0.1986354 0.3669741
## 2 0.5917467 0.1967395
## 3 0.6574267 0.7186740
```

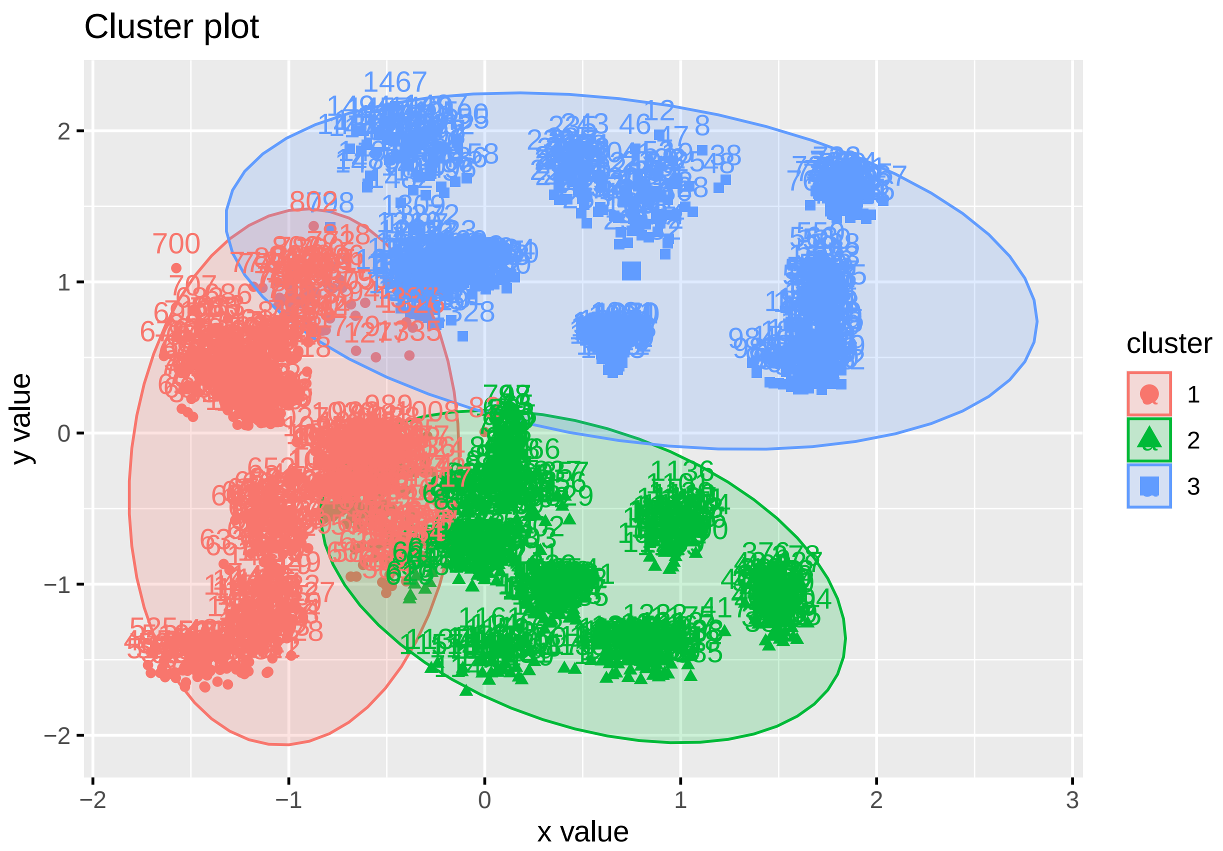
```
result$cluster #gives cluster vector showing the custer where each record falls
```

```
##     [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##    [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##    [75] 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3
##   [112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##   [149] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##   [186] 3 3 3 3 3 3 3 3 3 3 3 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] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##   [260] 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
##   [297] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 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 2
##   [371] 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
##   [408] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 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 1
##   [482] 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
##   [519] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##   [556] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 2 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1
##   [593] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 2 2 1 2 1 1 1 1 1 1
##   [630] 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
##   [667] 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
##   [704] 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##   [741] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1
##   [778] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##   [815] 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2
##   [852] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1
##   [889] 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
##   [926] 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##   [963] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [1000] 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
##  [1037] 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [1074] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [1111] 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
##  [1148] 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
##  [1185] 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
##  [1222] 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
##  [1259] 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [1296] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3
##  [1333] 3 3 1 3 1 3 3 3 3 3 3 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
##  [1370] 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
##  [1407] 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
##  [1444] 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [1481] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```
table(result$cluster,binary.class)
```

```
##    binary.class
##       0   1
##   1 315 280
##   2 164 229
##   3 288 222
```

```
fviz_cluster(result , binary.new, ellipse.type = "norm")
```



### Accuracy for 3 centroids

```
#CONCLUSION
#Total number of correctly classified instances are: 315 + 280 = 595
#Total number of incorrectly classified instances are: 164 + 288 + 229 + 222 = 903
#Accuracy = 595 /(595 + 903) = 0.40   i.e our model has achieved 40% accuracy!
```

Apply k-means clustering algorithm for 5 centroids

```
result<- kmeans(binary.new,5) #aplly k-means algorithm with no. of centroids(k)=5
result$size # gives no. of records in each cluster
```

```
## [1] 225 226 292 246 509
```

```
result$centers # gives value of cluster center datapoint value(5 centers for k=5)
```

```
##           x         y
## 1 0.4279611 0.7710493
## 2 0.1506591 0.5491736
## 3 0.8253709 0.6778152
## 4 0.6705661 0.1389318
## 5 0.2941539 0.2603013
```

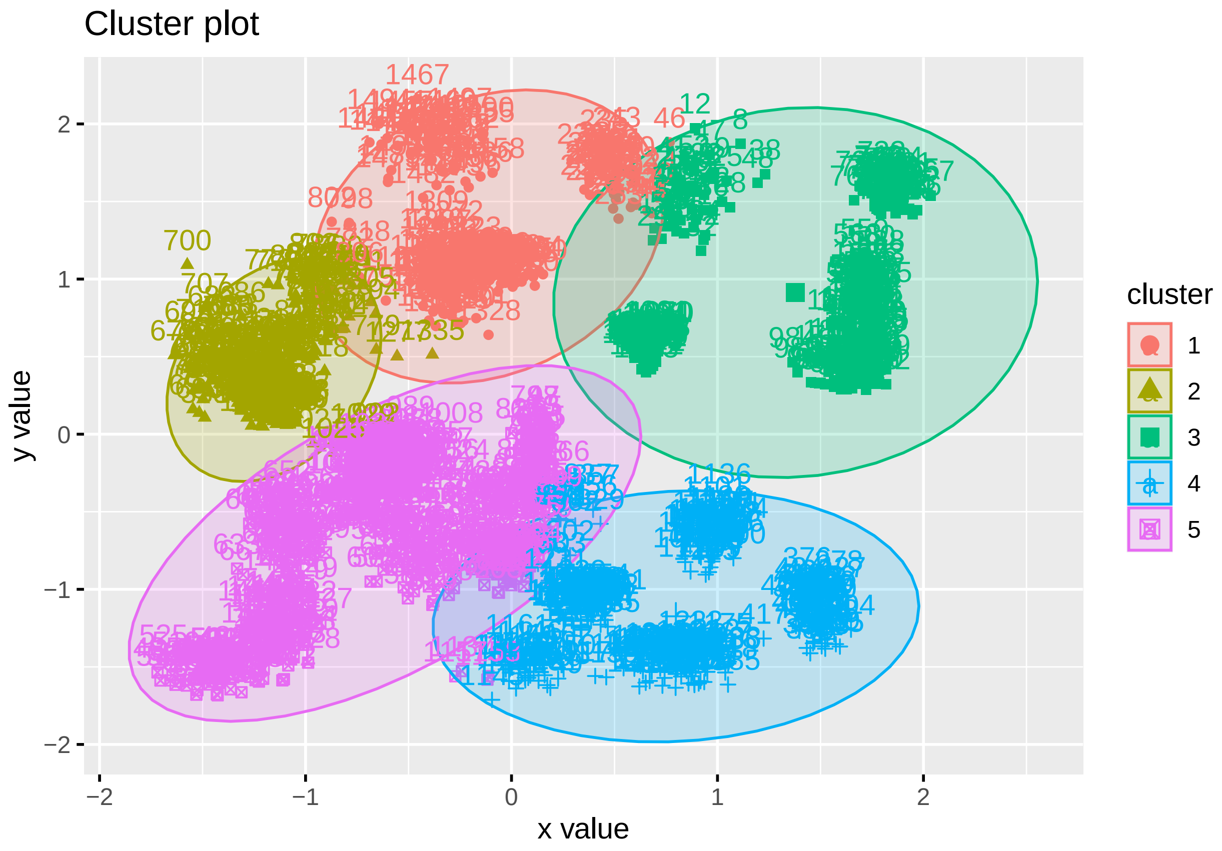
```
result$cluster #gives cluster vector showing the custer where each record falls
```

```
##    [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 1 3 3 3 3 3 3 3
##   [38] 3 3 1 3 3 3 1 3 1 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##   [75] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [112] 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
##  [149] 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [186] 3 3 3 3 3 3 3 3 3 3 3 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
##  [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] 5 5 4 5 5 5 5 5 5 5 5 4 5 5 5 5 5 5 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [297] 5 5 5 5 5 4 4 5 5 5 5 5 5 5 5 5 5 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [334] 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 4
##  [371] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [408] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [445] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [482] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [519] 5 5 5 5 5 5 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [556] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [593] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [630] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [667] 5 5 5 5 5 5 5 5 5 5 5 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
##  [704] 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [741] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 1 2 2 2 2 2
##  [778] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2
##  [815] 2 2 2 1 5 5 5 4 5 5 5 5 5 5 4 4 5 5 5 5 5 5 4 4 5 5 5 5 5 5 5 5 5 5 5 4 4
##  [852] 5 5 5 5 4 4 5 5 5 5 4 5 4 5 5 5 5 5 5 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [889] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [926] 5 5 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [963] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 2 5 5 5 5 5 5 5
## [1000] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 2 5 2 5 2 5 5 5 5 5 5 5
## [1037] 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1074] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [1111] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 4 4 4 5 4 4 4 4 4 4
## [1148] 4 4 4 4 4 4 4 4 4 5 5 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [1185] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [1222] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [1259] 4 4 4 4 4 4 4 4 4 4 4 4 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1296] 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
## [1333] 1 1 2 1 1 1 1 1 1 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
## [1370] 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 5 5 5 5 5 5
## [1407] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [1444] 5 5 5 5 5 5 5 5 5 5 5 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
## [1481] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
table(result$cluster,binary.class)
```

```
##    binary.class
##       0   1
##   1 104 121
##   2 118 108
##   3 184 108
##   4  63 183
##   5 298 211
```

```
fviz_cluster(result , binary.new, ellipse.type = "norm")
```



Accuracy for 5 centroids

```
#CONCLUSION
#Total number of correctly classified instances are: 211 + 227 = 438
#Total number of incorrectly classified instances are: 1060
#Accuracy = 438/(438 + 1060) = 0.29 i.e our model has achieved 29% accuracy!
```

Apply k-means clustering algorithm for 10 centroids

```
result<- kmeans(binary.new,10) #aplly k-means algorithm with no. of centroids(k)=10
result$size # gives no. of records in each cluster
```

```
##  [1] 222 193  60 117 126  92 239  55 236 158
```

```
result$centers # gives value of cluster center datapoint value(10 centers for k=10)
```

```
##             x          y
## 1  0.1485467 0.55195011
## 2  0.4780065 0.64743901
## 3  0.9423037 0.82513023
## 4  0.1178343 0.08395875
## 5  0.9070221 0.59165434
## 6  0.6460839 0.84653981
## 7  0.2741420 0.32907757
## 8  0.3501175 0.88085332
## 9  0.4811728 0.23082896
## 10 0.7494746 0.13525383
```

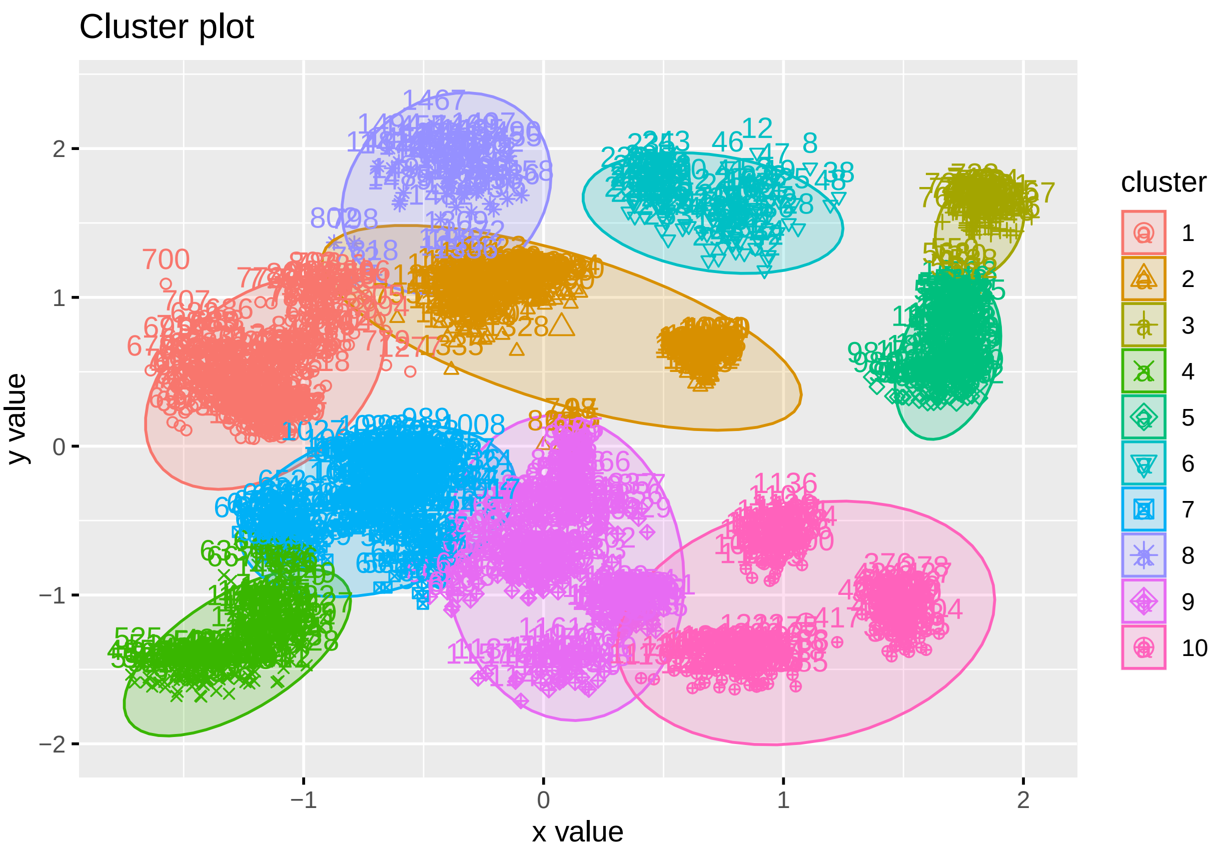
```
result$cluster #gives cluster vector showing the custer where each record falls
```

```
##    [1]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
##   [25]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
##   [49]  6  6  6  6  6  6  6  9  9  9  9  9  9  9  9  2  2  9  9  9  9  9  9  9
##   [73]  2  2  9  9  9  2  9  9  9  2  9  9  9  2  9  9  9  9  9  9  9  9  9  9
##   [97]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
##  [121]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
##  [145]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  5  5  5  5  5  5  5  5
##  [169]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [193]  5  5  5  5  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [217]  1  1  1  1  1  1  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
##  [241]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  9  9  9  9  9
##  [265]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
##  [289]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
##  [313]  9  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [337]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [361]  1  1  1  1  1  1  1  1  1 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##  [385] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##  [409] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  7  7  7  7  7
##  [433]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [457]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  4  4
##  [481]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##  [505]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##  [529]  4  4  4  5  5  5  5  5  5  3  5  5  5  5  5  5  5  5  5  5  5  3  5  5
##  [553]  5  5  5  3  5  5  5  3  5  5  5  5  5  5  5  3  5  5  5  3  5  5  5  7
##  [577]  9  9  7  9  7  7  7  9  7  7  7  7  9  9  7  9  9  9  7  7  7  7  7  7
##  [601]  9  7  9  9  7  7  7  7  7  7  9  7  7  7  7  9  9  7  9  9  9  7  9  7
##  [625]  7  7  7  7  4  7  7  7  4  7  4  7  4  4  4  7  7  7  7  7  7  7  7  4
##  [649]  7  7  7  7  7  7  7  7  7  7  7  7  7  4  7  7  7  7  4  7  7  7  7  7
##  [673]  7  7  7  4  7  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [697]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  3  3
##  [721]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
##  [745]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  1
##  [769]  1  1  1  8  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [793]  1  1  2  1  1  8  1  1  1  8  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [817]  1  8  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  7  9  9  9  9  9
##  [841]  9  9  9  7  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
##  [865]  9  9  9  9  9  9  9  7  9  7  9  9  7  7  7  7  7  7  7  7  7  7  7  7
##  [889]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [913]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  5  5
##  [937]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [961]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [985]  5  5  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [1009]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [1033]  7  7  7  7  7  7  7  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [1057]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [1081]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2 10 10 10 10 10 10 10 10 10
## [1105] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [1129] 10 10 10 10 10 10 10 10  9  9 10  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1153]  9  9  9  9  9  9  9  9  9  9  9  9 10 10 10 10 10 10 10 10 10 10 10 10
## [1177] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [1201] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  9  9
## [1225]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1249]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  2  8
## [1273]  2  2  2  2  1  2  2  2  2  2  2  2  2  2  8  2  2  2  2  2  2  2  2  2
## [1297]  8  2  2  2  2  2  2  2  2  2  2  2  8  8  2  8  2  2  2  2  2  2  2  2
## [1321]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  8  2  2  2  2  2  2  2  1
## [1345]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## [1369]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## [1393]  1  1  1  1  1  1  1  1  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
## [1417]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
## [1441]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  8  8  8  8  8  8  8  8  8  8
## [1465]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
## [1489]  8  8  8  8  8  8  8  8  8  8
```

```
table(result$cluster,binary.class)
```

```
##     binary.class
##        0   1
##   1  118 104
##   2   71 122
##   3   60   0
##   4   63  54
##   5   74  52
##   6   92   0
##   7  124 115
##   8    0  55
##   9  107 129
##   10  58 100
```

```
fviz_cluster(result , binary.new, ellipse.type = "norm")
```



Accuracy for 10 centroids

```
#CONCLUSION
#Total number of correctly classified instances are: 134 + 130 = 264
#Total number of incorrectly classified instances are: 1234
#Accuracy = 264/(264+1234) = 0.18 i.e our model has achieved 18% accuracy!
```

Apply k-means clustering algorithm for 15 centroids

```
result<- kmeans(binary.new,15) #aplly k-means algorithm with no. of centroids(k)=15
result$size # gives no. of records in each cluster
```

```
##  [1] 116 232  55 109  74  71  61 171 146  92  56  54  62 155  44
```

```
result$centers # gives value of cluster center datapoint value(15 centers for k=15)
```

```
##            x          y
## 1  0.1174546 0.08272258
## 2  0.2708633 0.32693866
## 3  0.2126607 0.66942284
## 4  0.4750475 0.35334387
## 5  0.3728073 0.68319543
## 6  0.9019500 0.54831326
## 7  0.9143334 0.65527036
## 8  0.1296211 0.51828235
## 9  0.4831935 0.15584324
## 10 0.6460839 0.84653981
## 11 0.6357047 0.57672761
## 12 0.9446337 0.83619494
## 13 0.4445585 0.70853852
## 14 0.7526923 0.13703427
## 15 0.3568023 0.91075928
```

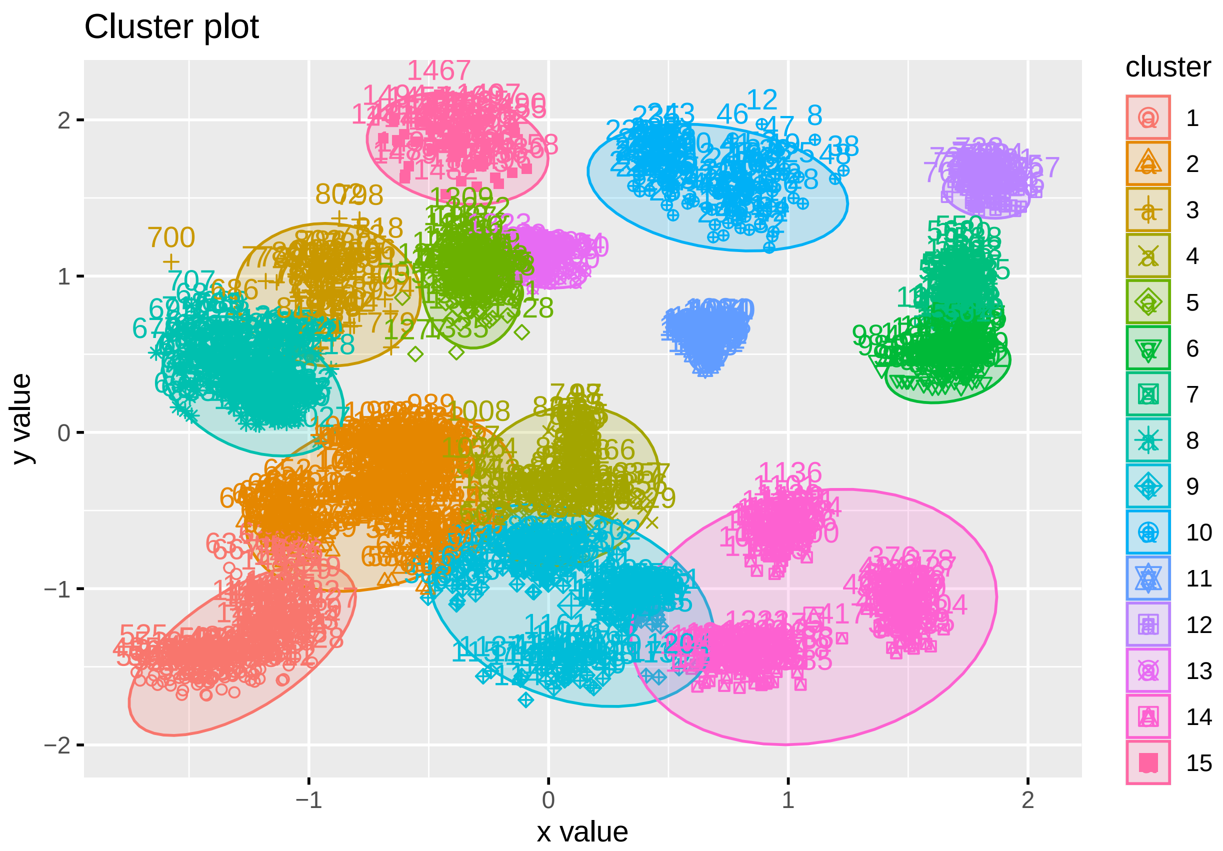
```
result$cluster #gives cluster vector showing the custer where each record falls
```

```
##   [1] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##  [25] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##  [49] 10 10 10 10 10 10 10  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##  [73]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##  [97]  4  4 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13  5 13 13 13
## [121] 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
## [145] 13 13 13 13 13 13 13 13 13 13 13  5 13 13 13 13  7  7  6  7  6  7  6  7
## [169]  6  7  6  6  6  6  6  7  6  6  6  7  7  7  7  6  6  6  7  7  6  7  7  7
## [193]  7  7  6  7  8  3  8  8  8  8  8  8  8  8  8  8  8  8  3  8  8  8  8  8
## [217]  8  8  8  8  3  8 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [241] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  9  9  9  9  9
## [265]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [289]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [313]  9  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
## [337]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
## [361]  8  8  8  8  8  8  8  8  8 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
## [385] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
## [409] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14  2  2  2  2  2
## [433]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [457]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  1  1
## [481]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## [505]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## [529]  1  1  1  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [553]  7  6  7  7  7  7  6  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  2
## [577]  9  9  2  2  2  2  2  9  2  9  2  2  9  9  2  4  4  2  2  2  2  2  2  2
## [601]  9  2  4  9  2  2  2  2  2  2  9  2  2  2  2  9  4  2  9  9  9  2  9  2
## [625]  2  2  2  2  1  2  2  2  1  2  1  2  1  1  2  2  2  2  2  2  2  2  2  1
## [649]  2  2  2  2  2  2  2  2  2  2  2  2  2  1  2  2  2  2  1  2  2  2  2  2
## [673]  2  2  2  1  2  8  8  8  8  8  8  8  8  3  8  8  8  8  8  8  8  8  8  8
## [697]  8  8  8  3  8  8  8  8  8  8  8  8  8  8  8  8  8 12 12 12 12 12 12 12
## [721] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
## [745] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12  3
## [769]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
## [793]  3  3  5  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
## [817]  3  3  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  2  4  4  4  4  4
## [841]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
## [865]  4  4  4  4  4  4  4  4  4  4  4  4  4  2  2  2  2  2  2  2  2  2  2  2
## [889]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [913]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  6  6
## [937]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
##  [961]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
##  [985]  6  6  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  4
## [1009]  2  2  2  2  2  2  2  2  4  2  2  2  2  2  2  4  2  2  8  2  2  2  2  2
## [1033]  2  2  2  4  2  2  2 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
## [1057] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
## [1081] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 14 14 14 14 14 14 14 14 14
## [1105] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
## [1129] 14 14 14 14 14 14 14 14  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1153]  9  9  9  9  9  9  9  9  9  9  9  9 14 14 14 14 14 14 14 14 14 14 14  9
## [1177] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
## [1201] 14 14 14  9 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14  9  9
## [1225]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1249]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  5  5
## [1273]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5 13  5
## [1297]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
## [1321]  5  5 13  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  8
## [1345]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
## [1369]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
## [1393]  8  8  8  8  8  8  8  8  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## [1417]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## [1441]  1  1  1  1  1  1  1  1  1  1  1  1  1  1 15 15 15 15 15 15 15 15 15 15
## [1465] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
## [1489] 15 15 15 15 15 15 15 15 15 15
```

```
table(result$cluster,binary.class)
```

```
##      binary.class
##          0   1
##   1     62  54
##   2    126 106
##   3      5  50
##   4     47  62
##   5      2  72
##   6     19  52
##   7     61   0
##   8    113  58
##   9     68  78
##   10    92   0
##   11     0  56
##   12    54   0
##   13    60   2
##   14    58  97
##   15     0  44
```

```
fviz_cluster(result , binary.new, ellipse.type = "norm")
```



Accuracy for 15 centroids

```
#CONCLUSION
#Total number of correctly classified instances are: 113 + 115 = 228
#Total number of incorrectly classified instances are:  1270
#Accuracy = 228/(228+1270) = 0.15 i.e our model has achieved 15% accuracy!
```

Apply k-means clustering algorithm for 20 centroids

```
result<- kmeans(binary.new,20) #aplly k-means algorithm with no. of centroids(k)=20
result$size # gives no. of records in each cluster
```

```
##  [1]  77  45  41  28  58  35  94  57  65 177 170  86  60 104  52  57  30  70 145
## [20]  47
```

```
result$centers # gives value of cluster center datapoint value(20 centers for k=20)
```

```
##            x          y
## 1  0.5236279 0.10709803
## 2  0.9210728 0.56892938
## 3  0.7165939 0.25955168
## 4  0.8545763 0.10795625
## 5  0.2182266 0.66601358
## 6  0.8807853 0.53735644
## 7  0.4874017 0.35771440
## 8  0.9355533 0.83887061
## 9  0.3295088 0.38083013
## 10 0.3948524 0.74999281
## 11 0.1291946 0.51882062
## 12 0.4220776 0.21867062
## 13 0.1682182 0.26362279
## 14 0.1126293 0.06858707
## 15 0.9141736 0.66331607
## 16 0.6749095 0.05112462
## 17 0.8497296 0.15714777
## 18 0.2867848 0.29312332
## 19 0.6394681 0.74149869
## 20 0.2693045 0.38695645
```

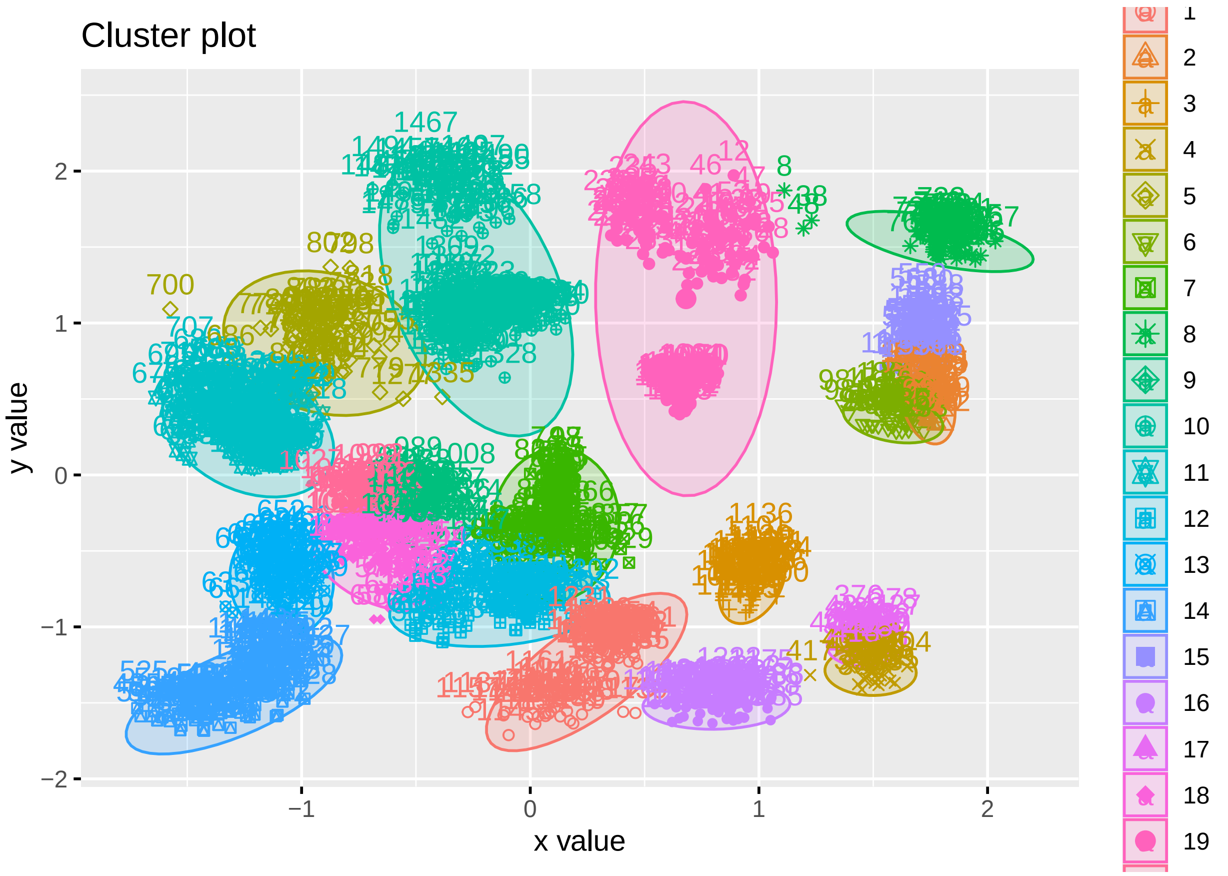
```
result$cluster #gives cluster vector showing the custer where each record falls
```

```
##     [1] 19 19 19 19 19 19 19  8 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
##    [25] 19 19 19 19 19 19 19 19 19 19 19 19 19  8 19 19 19 19 19 19 19 19 19  8
##    [49] 19 19 19 19 19 19 19  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##    [73]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##    [97]  7  7 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##   [121] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##   [145] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  2 15  2  2  2  2  2 15
##   [169]  2  2  6  2  2  2  2 15  2  2  2 15 15 15 15  6  2  6 15  2  6 15 15 15
##   [193] 15 15  2 15 11  5 11 11 11 11 11 11 11 11 11 11 11 11  5 11 11 11 11 11
##   [217] 11 11 11 11  5 11 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
##   [241] 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 12 12 12 12 12
##   [265] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
##   [289] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
##   [313] 12 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
##   [337] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
##   [361] 11 11 11 11 11 11 11 11 11  4 17  4  4 17  4 17 17 17  4 17 17  4  4 17
##   [385]  4  4 17  4 17 17 17  4 17  4 17 17  4  4  4 17 17 17  4  4 17  4 17 17
##   [409] 17  4  4  4  4  4  4 17  4 17  4 17 17 17  4 17 17  4 17 18 18 18 13 18
##   [433] 18 18 18 18 18 18 13 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18
##   [457] 18 18 20 18 18 18 18 18 18 13 18 18 18 18 18 18 18 18 18 18 18 18 14 14
##   [481] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
##   [505] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
##   [529] 14 14 14 15 15 15 15 15 15 15  2 15 15  2  2 15 15 15 15 15 15 15 15 15
##   [553] 15  2 15 15 15 15  2 15  2 15 15 15 15 15 15 15 15 15 15 15 15 15 15 18
##   [577] 12 12 18 12 18 18 12 12 18 12 12 18 12 12 18 12 12 12 12 18 18 18 18 18
##   [601] 12 12 12 12 12 18 12 12 12 18 12 18 18 18 12 12 12 18 12 12 12 18 12 13
##   [625] 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
##   [649] 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
##   [673] 13 13 13 13 13 11 11 11 11 11 11 11 11  5 11 11 11 11 11 11 11 11 11 11
##   [697] 11 11 11  5 11 11 11 11 11 11 11 11 11 11 11 11 11  8  8  8  8  8  8  8
##   [721]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
##   [745]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  5
##   [769]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##   [793]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [817]  5  5  7  7  7  7  7  7  7  7  7  7  7  7  7  7 12 12  9  7  7  7 12  7
##  [841]  7  7  7  9  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [865]  7  7  7  7  7  7  7  9  7  9  7  7  9  9  9 20  9 20  9 20  9  9  9 20
##  [889] 20  9  9  9  9 20  9  9 20  9  9  9 20  9  9  9 20 20  9 20  9  9  9 20
##  [913]  9 20 20  9 20  9  9  9 20  9 20  9 20 20  9  9 20  9 20  9  9  9  2  6
##  [937]  6  2  6  6  6  2  2  6  6  6  2  6  6  2  6  6  6  2  2  2  2  2  2  2
##  [961]  6  6  6  6  6  2  2  6  6  2  6  2  6  6  6  6  6  2  2  6  2  6  2  6
##  [985]  6  6  9  9  9 20  9 20 20 18  9  9  9 20 20 20  9 20  9 18 20 20 20  9
## [1009]  9 20 20  9 20 20  9 18  9 20 18 20  9  9 20  9 20  9 20 20 20 20 20 20
## [1033]  9  9  9  9  9 20  9 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
## [1057] 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
## [1081] 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19  3  3  3  3  3  3  3  3  3
## [1105]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
## [1129]  3  3  3  3  3  3  3  3  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## [1153]  1  1  1  1  1  1  1  1  1  1  1  1 16 16 16 16 16 16 16 16 16 16 16  1
## [1177] 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
## [1201] 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16  1  1
## [1225]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## [1249]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 10 10
## [1273] 10 10 10 10  5 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [1297] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [1321] 10 10 10 10 10 10 10 10 10 10 10 10 10 10  5 10 10 10 10 10 10 10 10 11
## [1345] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
## [1369] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
## [1393] 11 11 11 11 11 11 11 11 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
## [1417] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 13 14
## [1441] 14 14 14 14 14 14 13 14 13 14 14 14 14 14 10 10 10 10 10 10 10 10 10 10
## [1465] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [1489] 10 10 10 10 10 10 10 10 10 10
```

```
table(result$cluster,binary.class)
```

```
##     binary.class
##        0   1
##   1    0  77
##   2   24  21
##   3    0  41
##   4   28   0
##   5    5  53
##   6    4  31
##   7   43  51
##   8   57   0
##   9    0  65
##   10  62 115
##   11 113  57
##   12  83   3
##   13  57   3
##   14  53  51
##   15  52   0
##   16   0  57
##   17  30   0
##   18  66   4
##   19  89  56
##   20   1  46
```

```
fviz_cluster(result , binary.new, ellipse.type = "norm")
```



Accuracy for 20 centroids

```
#CONCLUSION
#Total number of correctly classified instances are: 113 + 104 = 217
#Total number of incorrectly classified instances are: 1281
#Accuracy = 217/(217+1281) = 0.14 i.e our model has achieved 14% accuracy!
```

Apply k-means clustering algorithm for 25 centroids

```
result<- kmeans(binary.new,25) #aplly k-means algorithm with no. of centroids(k)=25
result$size # gives no. of records in each cluster
```

```
##  [1]  24  31  33  92  78  35  42  45  59  89  36  33 108  96  29  89  71  18 134
## [20]  18  13  39  56 125 105
```

```
result$centers # gives value of cluster center datapoint value(25 centers for k=25)
```

```
##             x          y
## 1  0.04284968 0.03618473
## 2  0.16589037 0.29455696
## 3  0.91738454 0.68263737
## 4  0.64608394 0.84653981
## 5  0.10380877 0.56038521
## 6  0.10218781 0.03995692
## 7  0.91350613 0.54036730
## 8  0.15796637 0.10813621
## 9  0.21765227 0.65319859
## 10 0.42239444 0.21445569
## 11 0.93420436 0.84037751
## 12 0.37441815 0.91194559
## 13 0.30382713 0.38432411
## 14 0.48554010 0.35719612
## 15 0.17070654 0.23055522
## 16 0.14668699 0.48137599
## 17 0.28646162 0.29380407
## 18 0.96549249 0.82782980
## 19 0.40705272 0.69589440
## 20 0.86356605 0.54113703
## 21 0.29358810 0.88988596
## 22 0.91352952 0.61381556
## 23 0.63570467 0.57672761
## 24 0.58473603 0.08422274
## 25 0.79241073 0.17838812
```

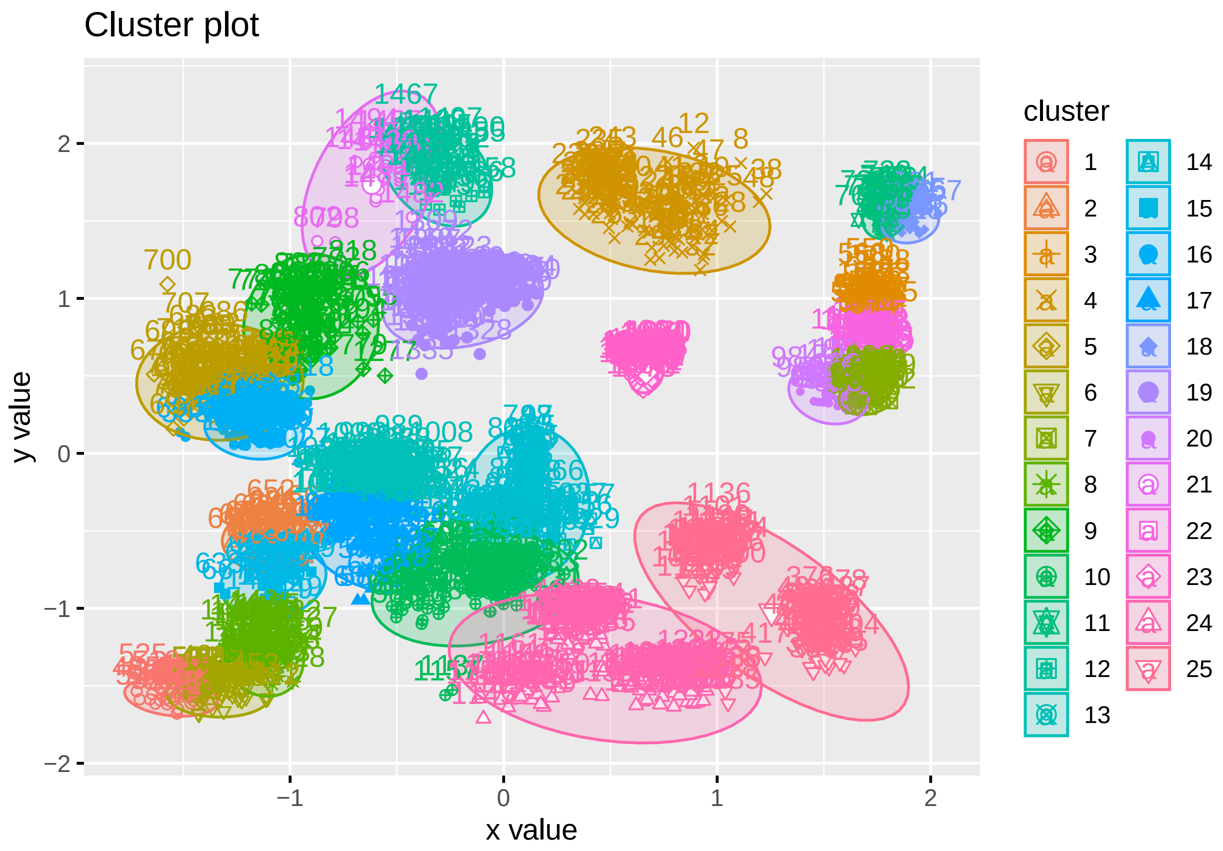
```
result$cluster #gives cluster vector showing the custer where each record falls
```

```
##   [1]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##  [25]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##  [49]  4  4  4  4  4  4  4 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
##  [73] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
##  [97] 14 14 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
## [121] 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
## [145] 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 22 22 22 22 22 22  7  3
## [169] 22 22 20 22 22  7 22 22  7 22 22 22  3 22 22 20 22 20 22 22  7  3 22 22
## [193] 22  3 22 22  5  9  5  9  5  9  5  5  5  5  5  5  5  5  9  5  9  5  5  5
## [217]  9 16  5  9  9  9  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
## [241]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4 10 10 10 10 10
## [265] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [289] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [313] 10  5  5  5  5 16  5 16 16 16 16 16  5 16  5  5  5  5 16 16 16 16 16  5
## [337]  5  5  5  5  5 16 16  5  5  5 16 16  5 16  5  5 16 16 16 16  5 16  5 16
## [361] 16  5 16  5 16  5 16 16  5 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
## [385] 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
## [409] 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 17 17 17  2 17
## [433] 17 17 17 17 17 17  2 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17
## [457] 17 17 17 17 17 17 17 17 17  2 17 17 17 17 17 17 17 17 17 17 17 17  6  6
## [481]  6  6  6  6  1  6  1  1  6  6  6  6  6  1  6  1  6  1  1  1  6  1  6  1
## [505]  6  6  1  6  6  1  1  6  1  1  6  6  6  6  1  1  1  1  6  1  1  1  1  6
## [529]  1  6  6 22  3  3  3  3  3  3 22  3  3 22 22 22 22  3  3 22 22  3  3  3
## [553]  3 22  3  3  3  3  7  3 22  3 22  3 22 22  3  3  3 22  3  3  3  3  3 17
## [577] 10 10 17 10 17 17 10 10 17 10 10 17 10 10 17 10 10 10 10 17 17 17 17 17
##  [601] 10 10 10 10 10 17 10 10 10 17 10 17 17 17 10 10 10 17 10 10 10 17 10  2
##  [625]  2  2  2 15 15  2 15 15 15  2 15  2 15 15 15 15 15  2  2 15  2  2  2 15
##  [649]  2 15 15  2  2  2  2 15  2 15  2 15 15 15  2 15  2 15 15  2 15  2  2 15
##  [673]  2  2  2 15  2  5  5  5  5  5  5  5  5  5  5  5  5 16  5 16  5  5  5  5
##  [697]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5 18 18 11 11 18 11 11
##  [721] 11 11 18 11 11 11 11 18 18 11 18 18 11 11 11 11 11 18 11 18 11 18 11 11
##  [745] 11 18 11 18 11 18 11 11 18 11 11 11 11 18 11 11 11 18 11 11 11 11 18  9
##  [769]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
##  [793]  9  9  9  9  9 21  9  9  9 21  9  9  9  9  9  9  9  9  9  9  9  9  9  9
##  [817]  9  9 14 14 14 14 14 14 14 14 14 14 14 14 14 14 10 10 13 14 14 14 10 14
##  [841] 14 14 14 13 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
##  [865] 14 14 14 14 14 14 14 14 14 13 14 14 13 13 13 13 13 13 13 13 13 13 13 13
##  [889] 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
##  [913] 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13  7 20
##  [937]  7  7  7  7 20  7  7 20 20 20  7  7  7  7 20  7 20  7  7  7  7  7  7  7
##  [961]  7 20  7  7  7  7  7 20 20  7 20  7 20  7  7 20  7  7  7 20  7  7  7 20
##  [985]  7  7 13 13 13 13 13 13 13 17 13 13 13 13 13 13 13 13 13 17 13 13 13 13
## [1009] 13 13 13 13 13 13 13 17 14 13 17 13 13 13 13 13 13 13 16 13 13 13 13 13
## [1033] 13 13 13 13 13 13 13 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
## [1057] 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
## [1081] 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 25 25 25 25 25 25 25 25 25
## [1105] 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
## [1129] 25 25 25 25 25 25 25 25 10 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
## [1153] 24 24 24 24 10 24 24 24 24 24 24 24 24 24 25 24 24 24 24 24 24 24 25 24
## [1177] 24 24 24 24 24 24 24 24 25 25 24 25 24 24 24 24 24 24 24 24 24 24 24 24
## [1201] 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 25 24 24 24 24 24 24
## [1225] 24 24 24 24 24 24 10 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
## [1249] 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 19 19
## [1273] 19 19 19 19  9 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
## [1297] 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
## [1321] 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 16
## [1345] 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
## [1369] 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
## [1393] 16 16 16 16 16 16 16 16  8  8  8  8  8  8  8  8  8  6  6  8  8  8  8  8
## [1417]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  6  8 15  8
## [1441]  8  6  8  6  8  8 15  8 15  8  8  8  6  8 12 12 21 12 12 12 12 12 12 21
## [1465] 12 12 12 12 12 12 12 12 12 21 12 21 12 12 21 12 21 21 12 12 21 12 12 12
## [1489] 21 12 21 12 12 21 12 12 12 12
```

```
table(result$cluster,binary.class)
```

```
##     binary.class
##        0  1
##   1   24  0
##   2   31  0
##   3   33  0
##   4   92  0
##   5   78  0
##   6   29  6
##   7    5 37
##   8    0 45
##   9    9 50
##   10  83  6
##   11  36  0
##    12    0   33
##    13    0  108
##    14   43   53
##    15   26    3
##    16   31   58
##    17   67    4
##    18   18    0
##    19   62   72
##    20    3   15
##    21    0   13
##    22   39    0
##    23    0   56
##    24    0  125
##    25   58   47
```

```
fviz_cluster(result , binary.new, ellipse.type = "norm")
```



Accuracy for 25 centroids

```
#CONCLUSION
#Total number of correctly classified instances are: 113 + 112 = 225
#Total number of incorrectly classified instances are:  1273
#Accuracy = 225/(225+1273) = .15 i.e our model has achieved 15% accuracy!
```

Read the trinary-classifier-data

```
df_trinary <- read.csv("C:/Users/chris/dsc520/data/trinary-classifier-data.csv")
head(df_trinary)
```

```
##   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
```

Preprocess the dataset

```
trinary.new<- df_trinary[,c(2, 3)]
trinary.class<- df_trinary[,c(1)]
```

Normalize the dataset

```
normalize <- function(x){
  return ((x-min(x))/(max(x)-min(x)))
}

trinary.new$x<- normalize(trinary.new$x)
trinary.new$y<- normalize(trinary.new$y)

head(trinary.new)
```

```
##           x         y
## 1 0.3395560 0.3890216
## 2 0.3495902 0.5037685
## 3 0.3735363 0.4801528
## 4 0.3605825 0.4041591
## 5 0.3779910 0.4442837
## 6 0.3708747 0.4326334
```

Apply k-means clustering algorithm for 3 centroids

```
result<- kmeans(trinary.new,3) #aplly k-means algorithm with no. of centroids(k)=3
result$size # gives no. of records in each cluster
```

```
## [1] 455 452 661
```

```r
result$centers # gives value of cluster center datapoint value(3 centers for k=3)
```

```
##           x         y
## 1 0.5790819 0.2681119
## 2 0.6918873 0.7987016
## 3 0.3085831 0.5429129
```

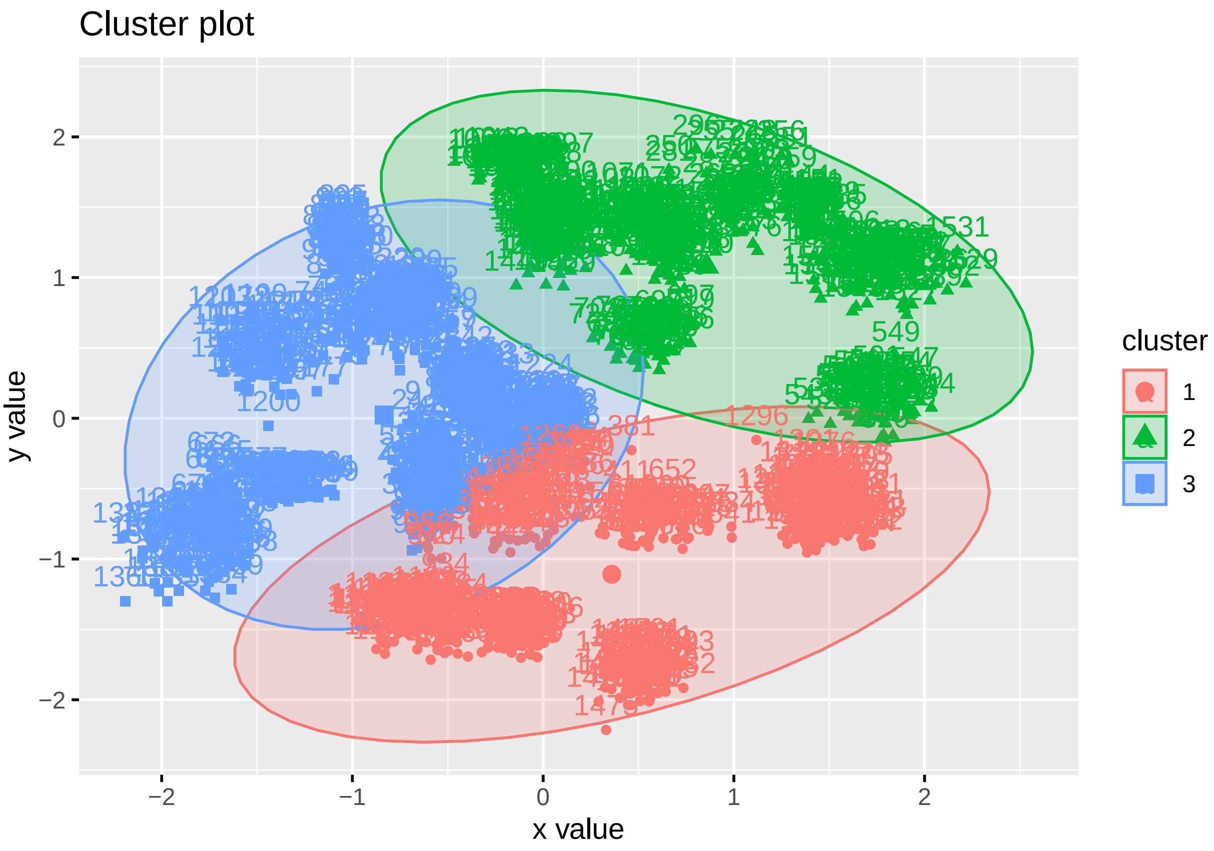
```r
result$cluster #gives cluster vector showing the custer where each record falls
```

```
##    [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##   [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##   [75] 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
##  [112] 1 1 1 1 1 1 1 1 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
##  [149] 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
##  [186] 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [223] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [260] 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
##  [297] 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [334] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 1 3 3 1 3 1 1 3 3 1 3
##  [371] 1 3 3 3 1 1 1 3 1 3 1 3 1 1 1 1 3 3 3 1 3 3 3 3 1 1 3 1 1 1 1 1 1 1 1 3 1
##  [408] 1 3 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 3 1 3 1 3 1 3 1 1
##  [445] 1 3 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 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [519] 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
##  [556] 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [593] 3 3 3 3 3 3 3 3 3 3 3 3 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
##  [630] 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 3 3 3 3 3 3 3 3 3 3 3 3
##  [667] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [704] 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 3 3 3 3
##  [741] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [778] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [815] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [852] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [889] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3
##  [926] 3 3 3 3 3 3 3 3 1 3 3 3 3 3 1 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3
##  [963] 3 3 3 3 3 3 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
## [1000] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1037] 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
## [1074] 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
## [1111] 2 2 2 2 2 2 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
## [1148] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1185] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3
## [1222] 3 3 3 1 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3
## [1259] 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1296] 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
## [1333] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1370] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1407] 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
## [1444] 2 2 2 2 2 2 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
## [1481] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1518] 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
## [1555] 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
table(result$cluster,trinary.class)
```

```
##    trinary.class
##       0   1   2
##   1  83 206 166
##   2 123 201 128
##   3 188 315 158
```

```
fviz_cluster(result , trinary.new, ellipse.type = "norm")
```



### Accuracy for 3 centroids

```
#CONCLUSION
#Total number of correctly classified instances are: 177 + 407 + 212 = 796
#Total number of incorrectly classified instances are: 772
#Accuracy = 796/(796+772) = 0.51 i.e our model has achieved 51% accuracy!
```

Apply k-means clustering algorithm for 5 centroids

```
result<- kmeans(trinary.new,5) #aplly k-means algorithm with no. of centroids(k)=5
result$size # gives no. of records in each cluster
```

```
## [1] 233 516 308 222 289
```

```
result$centers # gives value of cluster center datapoint value(5 centers for k=5)
```

```
##           x         y
## 1 0.4572183 0.1690295
## 2 0.3363702 0.4558994
## 3 0.3550543 0.7834396
## 4 0.8109186 0.4115140
## 5 0.7290976 0.8117163
```

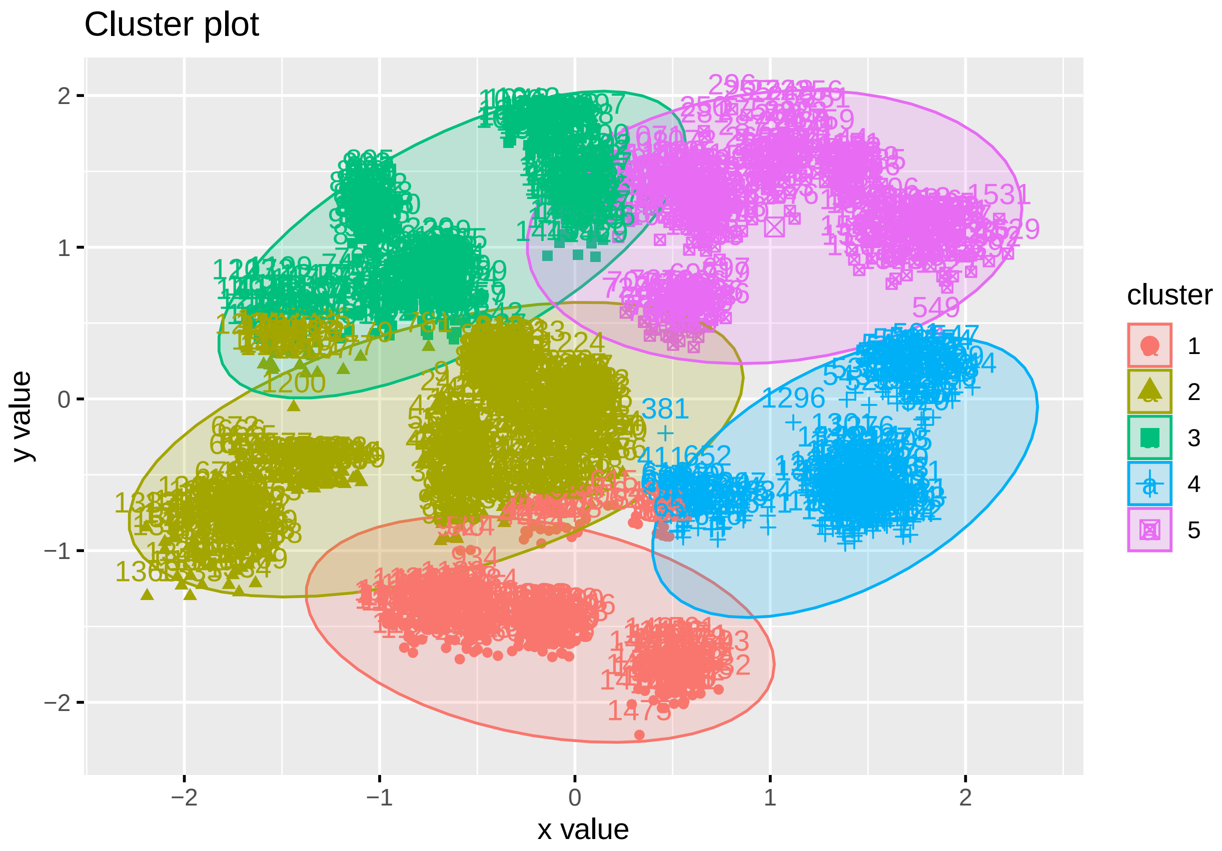
```
result$cluster #gives cluster vector showing the custer where each record falls
```

```
##    [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 2 2
##   [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##   [75] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [112] 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [149] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [186] 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [223] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [260] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [297] 5 5 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
##  [334] 2 2 2 3 2 2 2 2 3 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
##  [371] 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 1 2 2 1
##  [408] 2 2 2 4 1 2 2 1 2 1 2 2 2 1 2 1 2 2 1 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 2 2 1
##  [445] 1 2 1 2 2 2 2 2 2 2 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 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [519] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 4 4 4 4 4 4
##  [556] 4 4 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
##  [593] 2 2 2 2 2 2 2 2 2 2 2 2 1 4 4 4 4 4 4 4 4 4 1 4 4 4 4 1 4 4 1 4 4 1 4 4 4
##  [630] 4 1 1 1 4 4 4 4 4 1 4 4 4 4 4 4 4 4 4 1 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2
##  [667] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
##  [704] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 3 3 3 3
##  [741] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [778] 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2
##  [815] 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
##  [852] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [889] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 1 2
##  [926] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [963] 2 2 2 2 2 2 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
## [1000] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1037] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 3 5 5 5
## [1074] 5 5 5 3 5 5 5 3 5 5 5 5 5 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [1111] 5 5 5 5 5 5 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
## [1148] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 2 2 3 2 3 3 3 3 3 3 2 2 2 3 3 3 2 2
## [1185] 2 3 3 3 3 2 2 2 3 3 2 3 2 2 3 2 2 2 3 2 3 2 2 2 2 3 3 2 3 2 3 3 3 2 3 2 2
## [1222] 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
## [1259] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [1296] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [1333] 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
## [1370] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1407] 3 3 3 3 3 3 3 3 3 3 3 3 3 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1444] 3 3 3 3 3 3 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
## [1481] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [1518] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [1555] 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

```
table(result$cluster,trinary.class)
```

```
##    trinary.class
##       0   1   2
##   1   0 130 103
##   2 148 233 135
##   3  56 162  90
##   4  67  97  58
##   5 123 100  66
```

```
fviz_cluster(result , trinary.new, ellipse.type = "norm")
```



Accuracy for 5 centroids

```
#CONCLUSION
#Total number of correctly classified instances are: 146 + 238 + 127 = 511
#Total number of incorrectly classified instances are:  1057
#Accuracy = 511/(511+1057) = 0.32 i.e our model has achieved 32% accuracy!
```

Apply k-means clustering algorithm for 10 centroids

```
result<- kmeans(trinary.new,10) #aplly k-means algorithm with no. of centroids(k)=10
result$size # gives no. of records in each cluster
```

```
##  [1]  52 133 210 170 377  57 101 226 118 124
```

```
result$centers # gives value of cluster center datapoint value(10 centers for k=10)
```

```
##            x         y
## 1  0.6225631 0.6647173
## 2  0.1280964 0.3556988
## 3  0.4496658 0.1508958
## 4  0.7791655 0.3599500
## 5  0.4299381 0.4759888
## 6  0.8926408 0.5590896
## 7  0.8673340 0.8161890
## 8  0.2718773 0.7133674
## 9  0.4935597 0.8866637
## 10 0.6745039 0.8670831
```

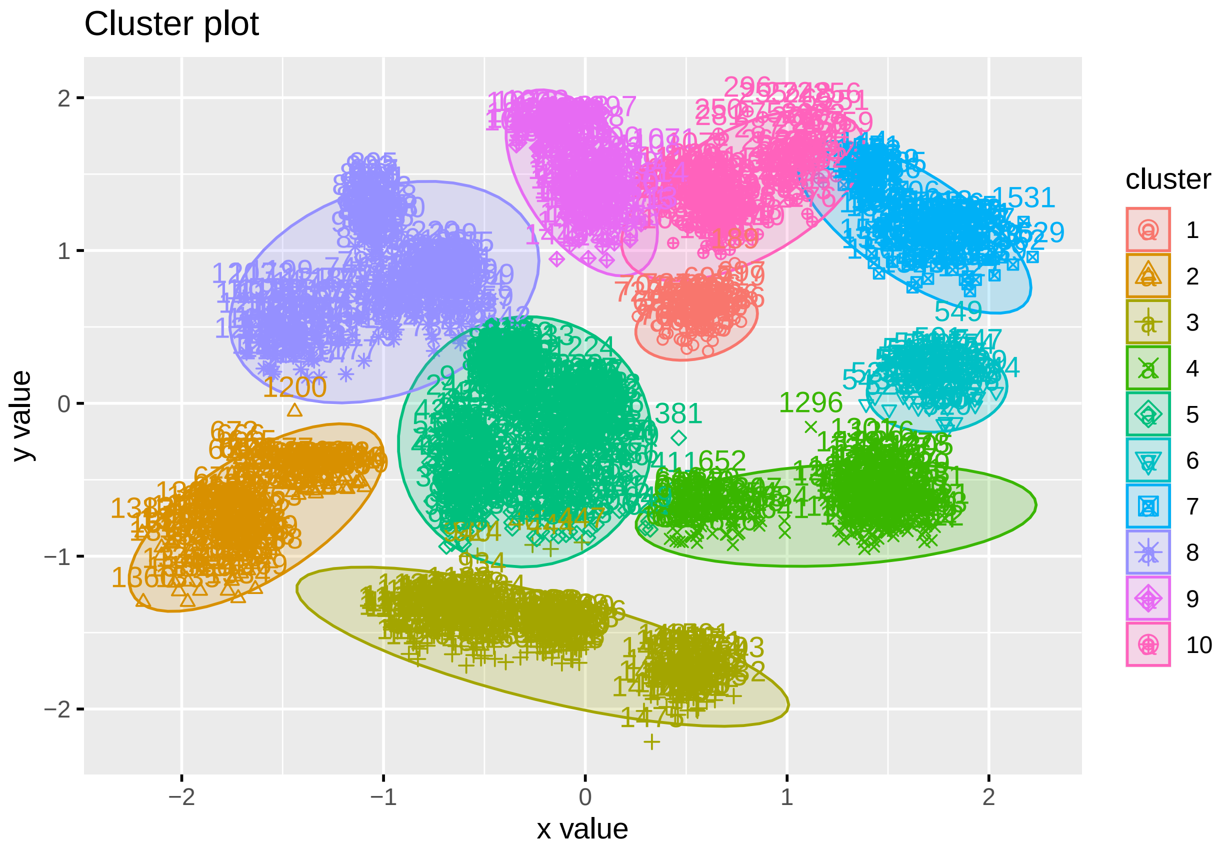
```
result$cluster #gives cluster vector showing the custer where each record falls
```

```
##    [1]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##   [25]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##   [49]  5  5  5  5  5  5  5  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##   [73]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##   [97]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##  [121]  4  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [145]  7  7  7  7  7  7  7  7  7 10  7  7  7 10 10 10 10 10 10 10 10 10 10 10
##  [169] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  1 10 10 10
##  [193]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
##  [217]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
##  [241]  8  8  8  8  8  8 10 10 10 10 10 10 10 10 10 10 10 10 10 10  7 10 10 10
##  [265] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##  [289] 10 10 10 10 10 10 10 10 10 10  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [313]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [337]  5  5  5  5  5  8  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [361]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [385]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  3  5
##  [409]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [433]  5  5  5  5  5  5  5  5  5  5  5  3  5  5  3  5  5  5  5  5  5  5  3  3
##  [457]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
##  [481]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  6  6  6  6
##  [505]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
##  [529]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
##  [553]  6  6  6  6  6  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
##  [577]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
##  [601]  2  2  2  2  4  4  4  4  4  4  4  4  4  4  5  4  4  4  4  5  4  4  4  4
##  [625]  4  4  4  4  4  4  5  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##  [649]  5  4  4  4  4  4  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
##  [673]  2  2  2  2  2  2  2  2  2  2  2  2  2  1  1  1  1  1  1  1  1  1  1  1
##  [697]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [721]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  8  8  8  8  8  8  8  8
##  [745]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
##  [769]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
##  [793]  8  8  8  8  8  8  8  8  8  8  8  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [817]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [841]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [865]  5  5  5  5  5  5  5  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
##  [889]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
##  [913]  8  8  8  8  8  8  8  8  8  8  5  3  5  5  5  5  5  5  5  5  5  3  5  5
##  [937]  5  5  5  3  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [961]  5  5  5  5  5  5  5  5  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
##  [985]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
## [1009]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  9  9  9  9  9  9  9  9  9
## [1033]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1057]  9  9  9  9  9  9  9  9 10 10 10 10 10  9  9 10 10  9 10 10  9  9 10 10
## [1081]  9 10 10 10 10 10  9 10 10 10 10 10  9 10 10 10 10 10 10 10 10 10 10  9
## [1105] 10 10 10  9  9 10 10 10  9  9  9 10  3  3  3  3  3  3  3  3  3  3  3  3
## [1129]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
## [1153]  3  3  3  3  3  3  3  3  3  3  3  3  3  8  8  8  8  8  8  8  8  8  8  8
## [1177]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  2
## [1201]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  5  5  5  5  5
## [1225]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
## [1249]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
## [1273]  5  5  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
## [1297]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
## [1321]  4  4  4  4  4  4  4  4  4  4  4  4  2  2  2  2  2  2  2  2  2  2  2  2
## [1345]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [1369]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9
## [1393]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1417]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1441]  9  9  9  9  9  9  9  9  9  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
## [1465]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
## [1489]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  7  7  7  7  7  7  7  7  7
## [1513]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [1537]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [1561]  7  7  7  7  7  7  7  7
```

```
table(result$cluster,trinary.class)
```

```
##     trinary.class
##        0   1   2
##   1    1  51   0
##   2    0  78  55
##   3    0 107 103
##   4   66  46  58
##   5  150 172  55
##   6    0  57   0
##   7   36   0  65
##   8   55 118  53
##   9    0  55  63
##   10  86  38   0
```

```
fviz_cluster(result , trinary.new, ellipse.type = "norm")
```



Accuracy for 10 centroids

```
#CONCLUSION
#Total number of correctly classified instances are: 136 + 143 + 64 = 343
#Total number of incorrectly classified instances are: 1225
#Accuracy = 343/(343+1225) = 0.22 i.e our model has achieved 22% accuracy!
```

Apply k-means clustering algorithm for 15 centroids

```
result<- kmeans(trinary.new,15) #aplly k-means algorithm with no. of centroids(k)=15
result$size # gives no. of records in each cluster
```

```
##  [1]  60  51 111  75 170  58 363  57 126  65  51 159  81  87  54
```

```
result$centers # gives value of cluster center datapoint value(15 centers for k=15)
```

```
##              x          y
## 1  0.17566040 0.64169886
## 2  0.62172281 0.66286196
## 3  0.48890646 0.88933884
## 4  0.08902092 0.31129055
## 5  0.77916554 0.35995003
## 6  0.17862507 0.41312320
## 7  0.43158249 0.47338142
## 8  0.89264082 0.55908962
## 9  0.33417721 0.69542580
## 10 0.89620225 0.77997327
## 11 0.26087541 0.81895437
## 12 0.39160747 0.17713233
## 13 0.77201452 0.90120278
## 14 0.63321531 0.83823861
## 15 0.61259454 0.08265121
```

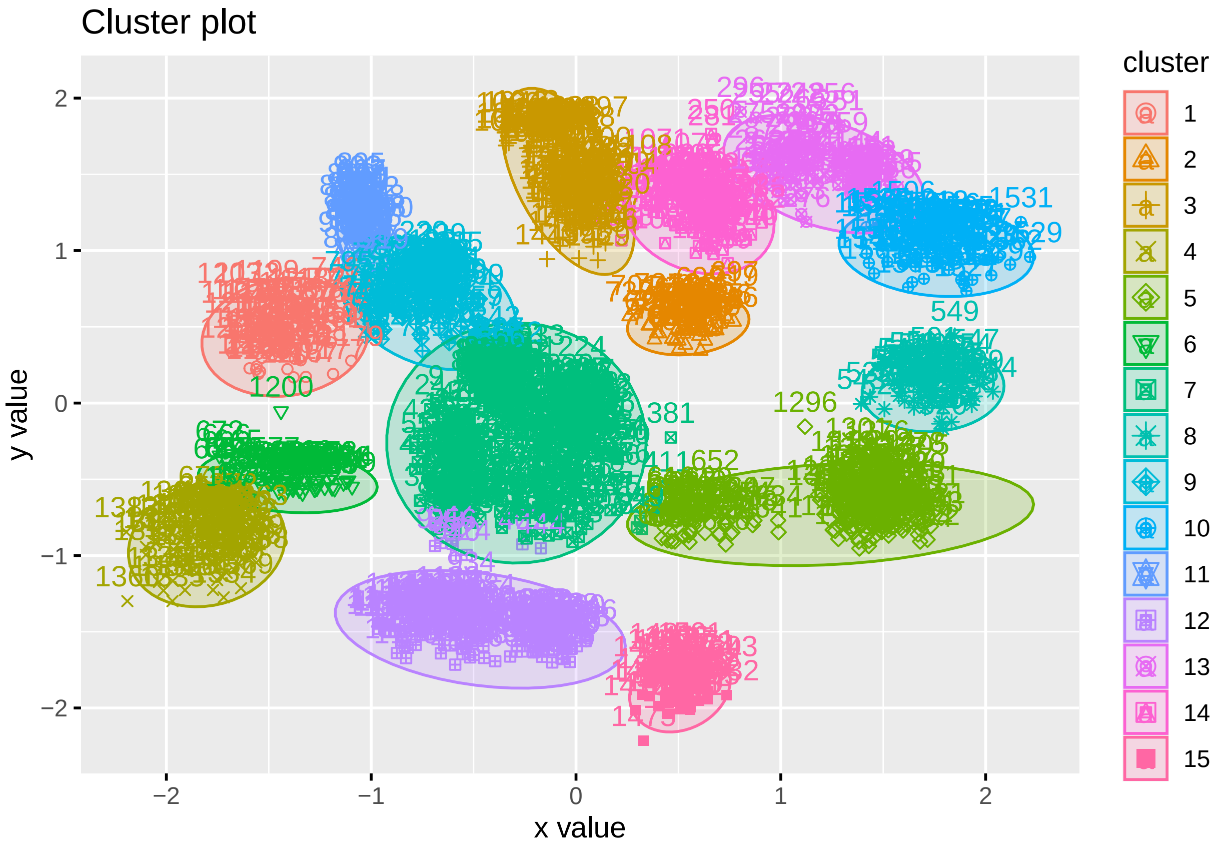
```
result$cluster #gives cluster vector showing the custer where each record falls
```

```
##   [1]  7  7  7  7  7  7  7 12  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [25]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [49]  7  7  7  7  7  7  7  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [73]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [97]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
## [121]  5 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
## [145] 13 13 13 13 13 13 13 13 13 13 13 13 13 14 14 14 14 14 14 14 14 14 14 14
## [169] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
## [193]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [217]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [241]  9  9  9  9  9  9 14 13 13 14 13 13 13 13 13 13 13 13 13 13 13 13 13 14
## [265] 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 14 13 13 13 14 14 13 13
## [289] 14 13 13 14 13 13 13 13 13 13  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [313]  7  7  7  7  7  7  7  7  7  9  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [337]  9  7  7  7  7  9  7  7  7  7  9  7  7  7  7  7  7  7  7  7  7  7  7  7
## [361]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [385]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7 12  7
## [409]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [433]  7  7  7  7  7  7  7  7  7  7  7 12  7  7  7  7  7  7  7  7  7  7 12 12
## [457] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
## [481] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12  8  8  8  8
## [505]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
## [529]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
## [553]  8  8  8  8  8  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
## [577]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
## [601]  6  6  6  6  5  5  5  5  5  5  5  5  5  5  7  5  5  5  5  7  5  5  5  5
## [625]  5  5  5  5  5  5  7  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
## [649]  7  5  5  5  5  5  4  4  6  4  4  4  4  6  4  4  6  6  4  4  4  4  4  6
## [673]  4  6  4  4  6  4  4  4  4  4  4  6  4  2  2  2  2  2  2  2  2  2  2  2
## [697]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [721]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9  9  9
## [745]  9  1  1  9  9  9  9  9  9  9  9  9  9  1  9  9  9  9  9  9  1  9  1  9
## [769]  9  9  1  1  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [793]  9  9  9  9  9  9  9  9  9  9  9  7  7  7  9  7  9  7  7  9  7  7  7  9
## [817]  7  9  7  7  9  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [841]  9  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [865]  7  9  7  7  7  7  7 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
##   [889] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
##   [913] 11 11 11 11 11 11 11 11 11 11 12 12  7  7  7  7  7  7  7  7  7 12  7  7
##   [937]  7  7  7 12  7  7  7 12  7  7  7  7  7  7  7  7  7  7  7 12  7  7  7  7
##   [961]  7  7  7  7  7  7  7  7 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
##   [985] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
##  [1009] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12  3  3  3  3  3  3  3  3  3
##  [1033]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
##  [1057]  3  3  3  3  3  3  3  3 14 14 14 14 14  3 14 14 14 14 14 14  3 14 14 14
##  [1081]  3 14 14 14 14 14  3 14 14 14 14 14 14 14 14 14 14 14 14 14 14 13 14  3
##  [1105] 14 14 14  3  3 14 14 14 14 14 14 14 12 12 12 12 12 12 12 12 12 12 12 12
##  [1129] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
##  [1153] 12 12 12 12 12 12 12 12 12 12 12 12 12  1  1  1  1  1  1  1  1  1  1  1
##  [1177]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  6
##  [1201]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  7  7  7  7  7
##  [1225]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [1249]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [1273]  7  7  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [1297]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [1321]  5  5  5  5  5  5  5  5  5  5  5  5  4  4  4  4  6  4  4  4  4  4  4  4
##  [1345]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##  [1369]  4  4  4  4  4  4  6  4  4  4  4  4  4  4  4  4  4  4  3  3  3  3  3  3
##  [1393]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
##  [1417]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
##  [1441]  3  3  3  3  3  3  3  3  3 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
##  [1465] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
##  [1489] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 10 10 10 10 10 10 10 10 10
##  [1513] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##  [1537] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##  [1561] 10 10 10 10 10 10 10 10
```

```
table(result$cluster,trinary.class)
```

```
##     trinary.class
##        0   1   2
##   1    0   7  53
##   2    0  51   0
##   3    0  48  63
##   4    0  23  52
##   5   66  46  58
##   6    0  55   3
##   7  146 162  55
##   8    0  57   0
##   9   58  68   0
##   10   0   0  65
##   11   0  51   0
##   12   1 109  49
##   13  80   1   0
##   14  43  44   0
##   15   0   0  54
```

```
fviz_cluster(result , trinary.new, ellipse.type = "norm")
```



Accuracy for 15 centroids

```
#CONCLUSION
#Total number of correctly classified instances are: 92 + 109 + 103 = 304
#Total number of incorrectly classified instances are:
#Accuracy = 304/(304+1264) = 0.19 i.e our model has achieved 19% accuracy!
```

Apply k-means clustering algorithm for 20 centroids

```
result<- kmeans(trinary.new,20) #aplly k-means algorithm with no. of centroids(k)=20
result$size # gives no. of records in each cluster
```

```
##  [1] 145 124 122  58  38  29 130 114 118  51 113 101  55  39  61  85  52  51  57
## [20]  25
```

```
result$centers # gives value of cluster center datapoint value(20 centers for k=20)
```

```
##            x          y
## 1  0.4157648 0.55769713
## 2  0.6745039 0.86708309
## 3  0.3769234 0.39809270
## 4  0.4665171 0.15769977
## 5  0.3789309 0.16932862
## 6  0.6052676 0.06313218
## 7  0.1259234 0.35358775
## 8  0.3262357 0.70453376
## 9  0.4935597 0.88666374
## 10 0.2608754 0.81895437
## 11 0.4956468 0.45304854
## 12 0.8673340 0.81618902
## 13 0.3229178 0.18398639
## 14 0.8215093 0.40767867
## 15 0.1755870 0.63976593
## 16 0.8355390 0.34529911
## 17 0.6225631 0.66471735
## 18 0.6322168 0.34783225
## 19 0.8926408 0.55908962
## 20 0.6210939 0.10529330
```

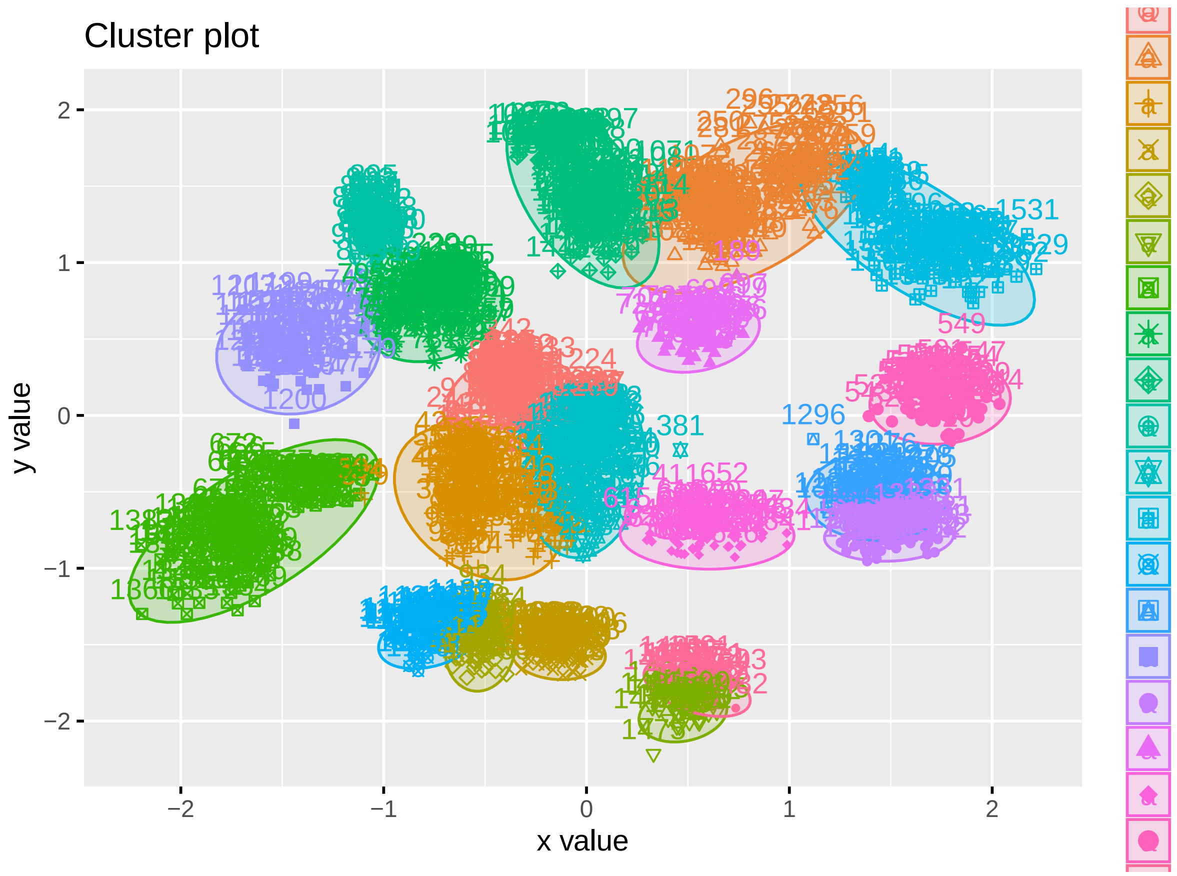
```
result$cluster #gives cluster vector showing the custer where each record falls
```

```
##    [1]  3  1  3  3  3  3  1  3  1  1  3  3  3  3  1  3  3  3  3  3  1  3  3  1
##   [25]  3  3  3  3  3  3  3  3  3  3  1  3  3  3  1  1  1  3  3  3  3  1  3  3
##   [49]  3  3  3  3  3  3  3 14 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
##   [73] 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
##   [97] 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 14 16
##  [121] 16 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
##  [145] 12 12 12 12 12 12 12 12 12  2 12 12 12  2  2  2  2  2  2  2  2  2  2  2
##  [169]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2 17  2  2  2
##  [193]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
##  [217]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
##  [241]  8  8  8  8  8  8  2  2  2  2  2  2  2  2  2  2  2  2  2  2 12  2  2  2
##  [265]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
##  [289]  2  2  2  2  2  2  2  2  2  2 11  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [313]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 11  1  1  1  1  1
##  [337]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 11 11 11 11 11 11 11 11
##  [361]  3 11 11 11 11 11 11 11 11 11 11 11  3 11 11 11 11 11 11 11 11  1 11 11
##  [385] 11 11 11 11 11 11 11 11  1 11  3 11  3 11 11  3 11  3 11 11  3  3  3 11
##  [409]  3 11 18  3 11  3  3 11 11 11 11 11  3 11  3 11  3  3 11  3 11 11 11  3
##  [433] 11  3 11  3  3  3  3  3 11  3 11  3  3  3 11 11  3  3 11 11  3 11 13  5
##  [457]  5  5 13  5  5  5 13  4  5  5  4  5  5  5  5  5 13  5 13 13  5  5 13  5
##  [481]  5  5  5  5 13 13 13  5  4  5  5 13  5  5  5  5  5  5 13  5 19 19 19 19
##  [505] 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
##  [529] 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
##  [553] 19 19 19 19 19  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [577]  7  7  3  7  7  7  7  7  7  7  7  7  7  7  7  7  7  3  7  7  7  7  7  7
##  [601]  7  7  7  7 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18
##  [625] 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18
##  [649] 18 18 18 18 18 18  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [673]  7  7  7  7  7  7  7  7  7  7  7  7  7 17 17 17 17 17 17 17 17 17 17 17
##  [697] 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17
##  [721] 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17  8  8  8  8  8  8  8  8
##  [745]  8 15 15  8  8  8  8  8  8  8  8  8  8 15  8  8  8  8  8  8 15  8 15  8
##  [769]  8  8 15 15  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
##  [793]  8  8  8  8  8  8  8  8  8  8  8  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [817]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [841]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [865]  1  1  1  1  1  1  1 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##  [889] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##  [913] 10 10 10 10 10 10 10 10 10 10  3  3  3  3  3  3  3  3  3  3  3  5  3  3
##  [937]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
##  [961]  3  3  3  3  3  3  3  3  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##  [985]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
## [1009]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  9  9  9  9  9  9  9  9  9
## [1033]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1057]  9  9  9  9  9  9  9  9  2  2  2  2  2  9  9  2  2  9  2  2  9  9  2  2
## [1081]  9  2  2  2  2  2  9  2  2  2  2  2  9  2  2  2  2  2  2  2  2  2  2  9
## [1105]  2  2  2  9  9  2  2  2  9  9  9  2 13 13 13 13 13 13 13 13  5 13 13  5
## [1129] 13 13 13  5 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13  5 13 13
## [1153]  5 13 13 13 13 13 13 13 13  5 13 13 13 15 15 15 15 15 15 15 15 15 15 15
## [1177] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
## [1201] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 11 11 11 11  1
## [1225] 11 11  1 11 11 11 11  1 11 11  1 11 11 11  1 11 11 11  1 11 11 11 11 11
## [1249] 11 11 11 11 11 11 11 11  1 11  1 11  1 11 11 11 11 11  1 11  1 11 11 11
## [1273] 11 11 16 14 14 14 14 14 14 14 14 14 16 14 16 16 16 14 16 16 14 14 14 14
## [1297] 14 16 16 16 14 14 16 16 14 14 16 16 14 16 14 14 14 14 14 14 14 14 14 16
## [1321] 14 16 14 16 14 14 14 16 14 14 16 16  7  7  7  7  7  7  7  7  7  7  7  7
## [1345]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [1369]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  9  9  9  9  9  9
## [1393]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1417]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1441]  9  9  9  9  9  9  9  9  9  6  6 20  6  6  6 20  6 20 20  6  6 20  6 20
## [1465]  6  6 20  6  6  6 20  6  6 20  6  6 20 20  6 20 20 20  6  6 20  6 20 20
## [1489] 20  6  6 20  6 20  6 20 20  6 20  6 20  6 20 12 12 12 12 12 12 12 12 12
## [1513] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
## [1537] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
## [1561] 12 12 12 12 12 12 12 12
```

```
table(result$cluster,trinary.class)
```

```
##     trinary.class
##       0  1  2
##   1  66 68 11
##   2  86 38  0
##   3  45 77  0
##   4   0 58  0
##   5   0 32  6
##   6   0  0 29
##   7   0 76 54
##   8  54 60  0
##   9   0 55 63
##   10  0 51  0
##   11 40 29 44
##   12 36  0 65
##   13  0 12 43
##   14  2  0 37
##   15  0  7 54
##   16 64  0 21
##   17  1 51  0
##    18   0 51   0
##    19   0 57   0
##    20   0   0 25
```

```
fviz_cluster(result , trinary.new, ellipse.type = "norm")
```



Accuracy for 20 centroids

```
#CONCLUSION
#Total number of correctly classified instances are: 66 + 68 + 64 = 198
#Total number of incorrectly classified instances are:
#Accuracy = 198/(304+1370) = .13  i.e our model has achieved 13% accuracy!
```

Apply k-means clustering algorithm for 25 centroids

```
result<- kmeans(trinary.new,25) #aplly k-means algorithm with no. of centroids(k)=25
result$size # gives no. of records in each cluster
```

```
##  [1]  53  65  51  40  58  92  75  25  58  57 114  29  48 116  81 111  87  49  26
## [20]  58  51  53  60  60  51
```

```
result$centers # gives value of cluster center datapoint value(25 centers for k=25)
```

```
##             x          y
## 1  0.37090267 0.17651604
## 2  0.89620225 0.77997327
## 3  0.85670036 0.34957970
## 4  0.31255023 0.17995971
## 5  0.17862507 0.41312320
## 6  0.49390485 0.49407742
## 7  0.08902092 0.31129055
## 8  0.62109386 0.10529330
## 9  0.46651705 0.15769977
## 10 0.89264082 0.55908962
## 11 0.32623575 0.70453376
## 12 0.60526755 0.06313218
## 13 0.63036046 0.34770891
## 14 0.41360235 0.56726479
## 15 0.77201452 0.90120278
## 16 0.48890646 0.88933884
## 17 0.63321531 0.83823861
## 18 0.80330401 0.35054442
## 19 0.82496233 0.42025289
## 20 0.36996473 0.36968415
## 21 0.26087541 0.81895437
## 22 0.47799173 0.37195886
## 23 0.17566040 0.64169886
## 24 0.36862028 0.46273821
## 25 0.62172281 0.66286196
```

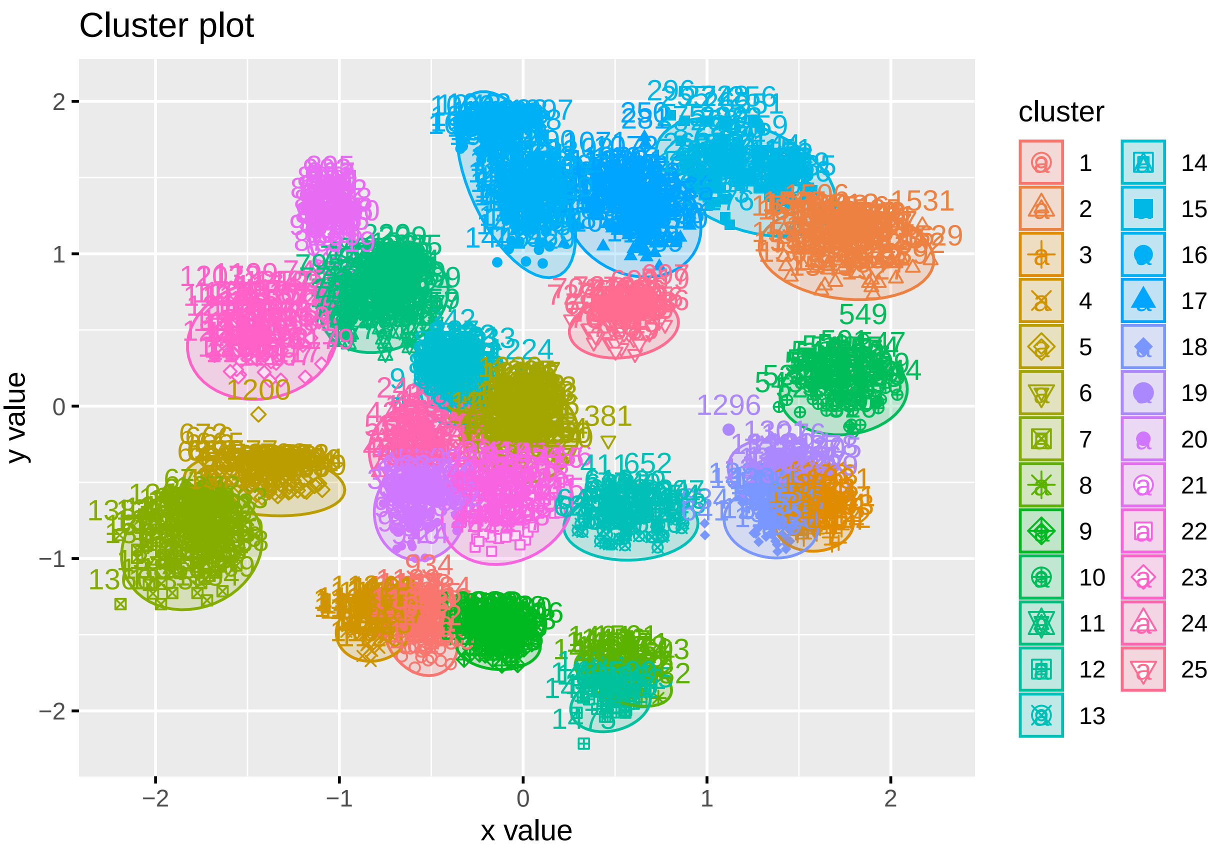
```
result$cluster #gives cluster vector showing the custer where each record falls
```

```
##   [1] 20 24 24 20 24 24 24 20 14 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
##  [25] 24 24 24 24 20 20 20 20 24 24 24 20 24 20 24 24 24 24 24 24 20 24 24 24
##  [49] 24 24 24 20 24 20 24 18 18 18 18  3  3  3  3  3 18 18 18  3  3  3  3  3
##  [73] 18 18  3 18  3  3 18  3 18  3 18  3 18  3  3 18 18  3 18  3 18 18 18  3
##  [97] 18  3  3  3  3 18  3  3 18  3  3  3 18  3  3 18 18 18 18 18  3  3 18  3
## [121]  3 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
## [145] 15 15 15 15 15 15 15 15 15 15 15 15 15 17 17 17 17 17 17 17 17 17 17 17
## [169] 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17
## [193] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
## [217] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
## [241] 11 11 11 11 11 11 17 15 15 17 15 15 15 15 15 15 15 15 15 15 15 15 15 17
## [265] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 17 15 15 15 17 17 15 15
## [289] 17 15 15 17 15 15 15 15 15 15  6 14 14 14 14 14  6 14 14 14 14 14 14 14
## [313] 14 14 14 14 14 14  6 14 14 14 14 14 14  6 14 14 14 14  6 14 14 14 14 14
## [337] 14 14 14 14 24 14 14 24 14 24 14 14 14 14 14 14  6  6  6  6  6  6  6  6
## [361] 24  6  6  6  6  6  6  6 22  6  6  6 24 22  6  6 22  6  6  6  6  6  6  6
## [385] 22 22  6  6 22 22  6  6  6  6 20 22 20 22 22 22 22 22 22 22 20 20 22 22
## [409] 20 22 13 22 22 24 22 22 22 22 22 22 22 22 22 22 20 22 22 22 22 22 22 20
## [433] 22 22 22 20 20 20 20 20 22 20 22 22 22 22 22 22 22 22 22 22 22 22  4  1
## [457]  1  1  1  1  1  1  4  9  1  1  9  1  1  1  1  1  1  1  4  4  1  1  4  1
## [481]  1  1  1  1  1  4  4  1  9  1  1  1  1  1  1  1  1  1  1  1 10 10 10 10
##  [505] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##  [529] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##  [553] 10 10 10 10 10  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [577]  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
##  [601]  5  5  5  5 13 13 13 13 13 13 13 13 13 13 22 13 13 13 13 13 13 13 13 13
##  [625] 13 13 13 13 13 13 13 13 13 18 13 13 13 13 13 13 18 13 13 13 13 13 13 13
##  [649] 13 13 13 13 13 13  7  7  5  7  7  7  7  5  7  7  5  5  7  7  7  7  7  5
##  [673]  7  5  7  7  5  7  7  7  7  7  7  5  7 25 25 25 25 25 25 25 25 25 25 25
##  [697] 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25
##  [721] 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 11 11 11 11 11 11 11 11
##  [745] 11 23 23 11 11 11 11 11 11 11 11 11 11 23 11 11 11 11 11 11 23 11 23 11
##  [769] 11 11 23 23 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
##  [793] 11 11 11 11 11 11 11 11 11 11 11 14 14 14 14 14 14 14 14 14 14 14 14 14
##  [817] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
##  [841] 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
##  [865] 14 14 14 14 14 14 14 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21
##  [889] 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21
##  [913] 21 21 21 21 21 21 21 21 21 21 20 20 24 24 20 20 24 20 20 20 20  1 20 20
##  [937] 20 20 20 20 24 20 24 20 24 20 20 24 20 20 20 20 24 20 24 20 24 20 20 20
##  [961] 24 20 20 20 24 20 20 20  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
##  [985]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1009]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9 16 16 16 16 16 16 16 16 16
## [1033] 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
## [1057] 16 16 16 16 16 16 16 16 17 17 17 17 17 16 17 17 17 17 17 17 16 17 17 17
## [1081] 16 17 17 17 17 17 16 17 17 17 17 17 17 17 17 17 17 17 17 17 17 15 17 16
## [1105] 17 17 17 16 16 17 17 17 17 17 17 17  1  4  1  4  4  4  4  4  1  4  4  1
## [1129]  4  1  4  1  4  4  4  4  4  1  1  4  4  4  4  4  4  4  4  1  4  1  4  4
## [1153]  1  1  1  4  4  4  4  1  4  1  4  1  4 23 23 23 23 23 23 23 23 23 23 23
## [1177] 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23  5
## [1201] 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23  6  6  6  6 14
## [1225]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
## [1249]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
## [1273]  6  6 18 19 19 19 18 19 19 18 19 19 18 19  3 18  3 19 18 18 19 19 19 19
## [1297] 19 18 18 18 19 19  3  3 19 19  3  3 18 18  3 19  3 19  3 18 18 18 19 18
## [1321] 19  3 19  3 19  3 19  3 19 18  3  3  7  7  7  7  5  7  7  7  7  7  7  7
## [1345]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [1369]  7  7  7  7  7  7  5  7  7  7  7  7  7  7  7  7  7  7 16 16 16 16 16 16
## [1393] 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
## [1417] 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16
## [1441] 16 16 16 16 16 16 16 16 16 12 12  8 12 12 12  8 12  8  8 12 12  8 12  8
## [1465] 12 12  8 12 12 12  8 12 12  8 12 12  8  8 12  8  8  8 12 12  8 12  8  8
## [1489]  8 12 12  8 12  8 12  8  8 12  8 12  8 12  8  2  2  2  2  2  2  2  2  2
## [1513]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [1537]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [1561]  2  2  2  2  2  2  2  2
```

```
table(result$cluster,trinary.class)
```

```
##     trinary.class
##       0  1  2
##   1   0 37 16
##   2   0  0 65
##   3  36  0 15
##   4   0  7 33
##    5   0 55  3
##    6  38  0 54
##    7   0 23 52
##    8   0  0 25
##    9   0 58  0
##   10   0 57  0
##   11  54 60  0
##   12   0  0 29
##   13   0 48  0
##   14  47 68  1
##   15  80  1  0
##   16   0 48 63
##   17  43 44  0
##   18  30  2 17
##   19   0  0 26
##   20  12 46  0
##   21   0 51  0
##   22   7 46  0
##   23   0  7 53
##   24  47 13  0
##   25   0 51  0
```

```
fviz_cluster(result , trinary.new, ellipse.type = "norm")
```



Accuracy for 25 centroids

```
#CONCLUSION
#Total number of correctly classified instances are: 80 + 73 + 53 = 206
#Total number of incorrectly classified instances are:
#Accuracy = 206/(206+1362) = 0.13  i.e our model has achieved 13% accuracy!
```