Exercise15

R Exercise 15

```r
setwd("~/MadR/Workspaces/dsc520")
```

Loading data from file

```r
data.binary=read.csv("data/binary-classifier-data.csv")
data.trinary=read.csv("data/trinary-classifier-data.csv")

summary(data.binary)
```

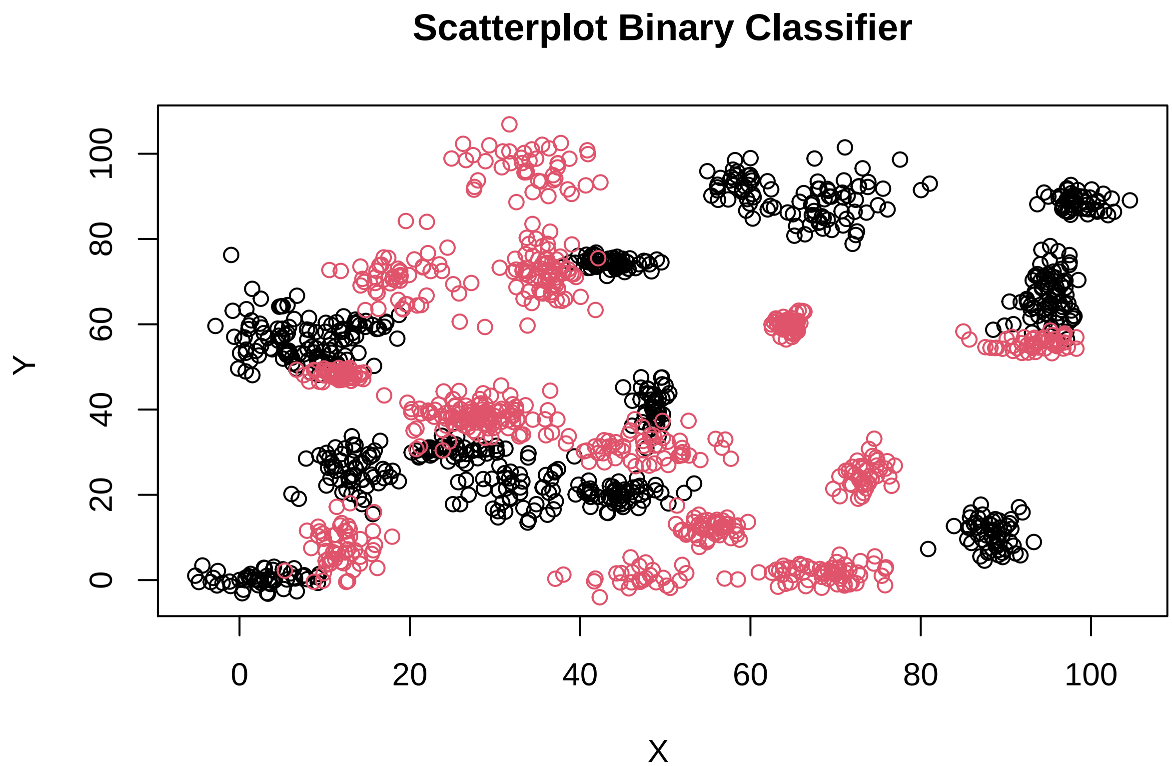
```
##      label             x                 y
##  Min.   :0.000   Min.   : -5.20   Min.   : -4.019
##  1st Qu.:0.000   1st Qu.: 19.77   1st Qu.: 21.207
##  Median :0.000   Median : 41.76   Median : 44.632
##  Mean   :0.488   Mean   : 45.07   Mean   : 45.011
##  3rd Qu.:1.000   3rd Qu.: 66.39   3rd Qu.: 68.698
##  Max.   :1.000   Max.   :104.58   Max.   :106.896
```

```r
summary(data.trinary)
```

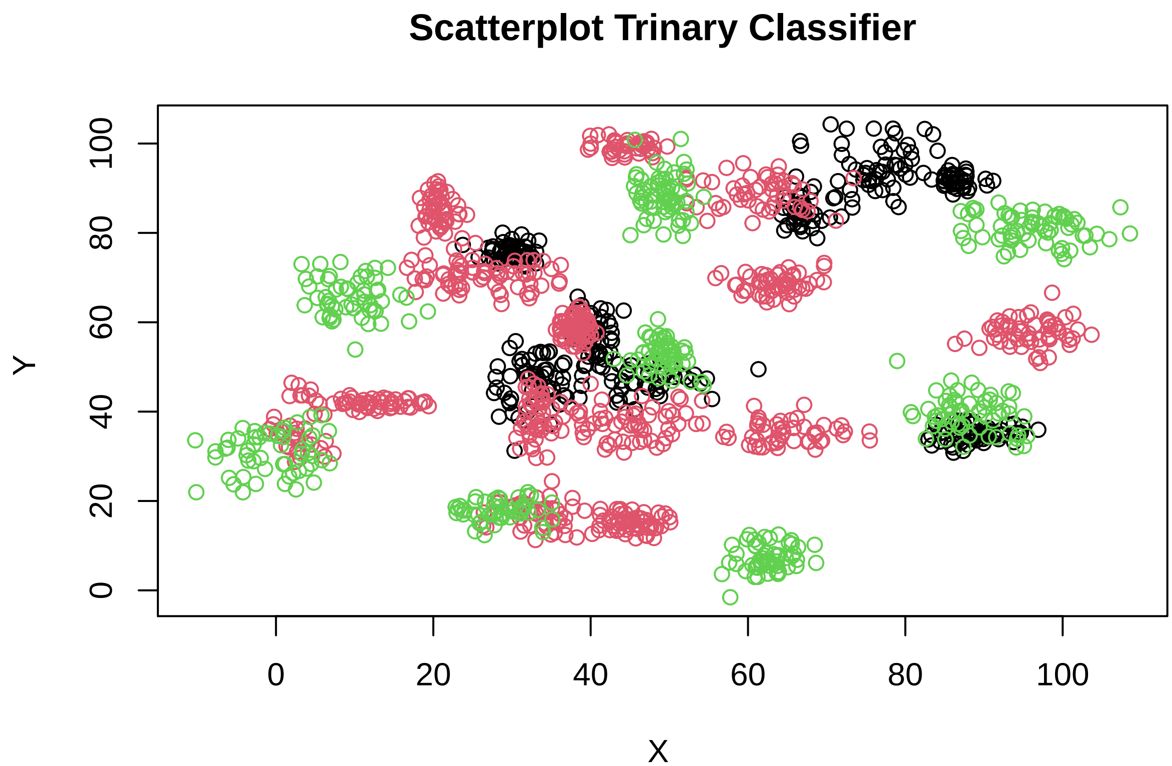
```
##      label             x                 y
##  Min.   :0.000   Min.   :-10.26   Min.   : -1.541
##  1st Qu.:0.000   1st Qu.: 31.15   1st Qu.: 35.906
##  Median :1.000   Median : 45.59   Median : 55.073
##  Mean   :1.037   Mean   : 48.86   Mean   : 55.282
##  3rd Qu.:2.000   3rd Qu.: 66.27   3rd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

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

```r
plot(data.binary$x,data.binary$y,col=data.binary$label +1,
     main="Scatterplot Binary Classifier",   xlab="X ", ylab="Y ")
```




```r
plot(data.trinary$x,data.trinary$y,col=data.trinary$label +1,
     main="Scatterplot Trinary Classifier",   xlab="X ", ylab="Y ")
```



b - Binary data set

```
set.seed(9850)
gp<-runif(nrow(data.binary))
data.binary<-data.binary[order(gp),]
head(data.binary)
```

```
##     label         x        y
## 216     0 12.006713 58.20435
## 405     0 88.024357 13.26384
## 316     0  7.993121 54.15258
## 804     1 16.669075 73.98231
## 103     0 40.565872 74.84798
## 20      0 69.521713 89.94501
```

```
normalize<-function(x){
  return (
            (x - min(x))/max(x)-min(x)
         )
}
data.binary.n<-as.data.frame(lapply(data.binary[,c(2:3)], normalize))

str(data.binary.n)
```

```
## 'data.frame':    1498 obs. of  2 variables:
```

```
##  $ x: num  5.37 6.09 5.33 5.41 5.64 ...
##  $ y: num  4.6 4.18 4.56 4.75 4.76 ...
```

```r
summary(data.binary.n)
```

```
##        x               y
##  Min.   :5.200   Min.   :4.019
##  1st Qu.:5.439   1st Qu.:4.255
##  Median :5.650   Median :4.475
##  Mean   :5.681   Mean   :4.478
##  3rd Qu.:5.885   3rd Qu.:4.700
##  Max.   :6.250   Max.   :5.057
```

```r
r<-round(0.8*nrow(data.binary.n))
l<-nrow(data.binary.n)

data.train<-data.binary.n[1:r,]
data.test<-data.binary.n[r:l,]

data.train.target<-data.binary[1:r,1]
data.test.target<-data.binary[r:l,1]

k_value<-round(sqrt(nrow(data.binary)))
```

calculatig knn and accuracy for each k =3 , 5 ,10 ,15 ,20 , 25 and 39

```r
 library("class")



knn.3<- knn(train = data.train, test=data.test,cl=data.train.target, k=3)
ACC.3<-100 * sum( data.test.target == knn.3) / NROW(data.test.target)

table(data.test.target,knn.3)
```

```
##                  knn.3
## data.test.target   0   1
##                 0 156   2
##                 1   2 141
```

```r
ACC.3
```

```
## [1] 98.6711
```

```r
knn.5<- knn(train = data.train, test=data.test,cl=data.train.target, k=5)
ACC.5<-100 * sum( data.test.target == knn.5) / NROW(data.test.target)

table(data.test.target,knn.5)
```

```
##                  knn.5
## data.test.target   0   1
##                 0 156   2
##                 1   1 142
```

```r
ACC.5
```

```
## [1] 99.00332
```

```r
knn.10<- knn(train = data.train, test=data.test,cl=data.train.target, k=10)
ACC.10<-100 * sum( data.test.target == knn.10) / NROW(data.test.target)

table(data.test.target,knn.10)
```

```
##                  knn.10
## data.test.target   0   1
##                 0 156   2
##                 1   0 143
```

```r
ACC.10
```

```
## [1] 99.33555
```

```r
knn.15<- knn(train = data.train, test=data.test,cl=data.train.target, k=15)
ACC.15<-100 * sum( data.test.target == knn.15) / NROW(data.test.target)

table(data.test.target,knn.15)
```

```
##                  knn.15
## data.test.target   0   1
##                 0 156   2
##                 1   1 142
```

```r
ACC.15
```

```
## [1] 99.00332
```

```r
knn.20<- knn(train = data.train, test=data.test,cl=data.train.target, k=20)
ACC.20<-100 * sum( data.test.target == knn.20) / NROW(data.test.target)

table(data.test.target,knn.20)
```

```
##                  knn.20
## data.test.target   0   1
##                 0 156   2
##                 1   0 143
```

```r
ACC.20
```

```
## [1] 99.33555
```

```r
knn.25<- knn(train = data.train, test=data.test,cl=data.train.target, k=25)
ACC.25<-100 * sum( data.test.target == knn.25) / NROW(data.test.target)

table(data.test.target,knn.25)
```

```
##                knn.25
## data.test.target   0   1
##                0 156   2
##                1   1 142
```

```r
ACC.25
```

```
## [1] 99.00332
```

```r
knn.39<- knn(train = data.train, test=data.test,cl=data.train.target, k=k_value)
ACC.39<-100 * sum( data.test.target == knn.39) / NROW(data.test.target)

table(data.test.target,knn.39)
```

```
##                knn.39
## data.test.target   0   1
##                0 156   2
##                1   1 142
```

```r
ACC.39
```

```
## [1] 99.00332
```

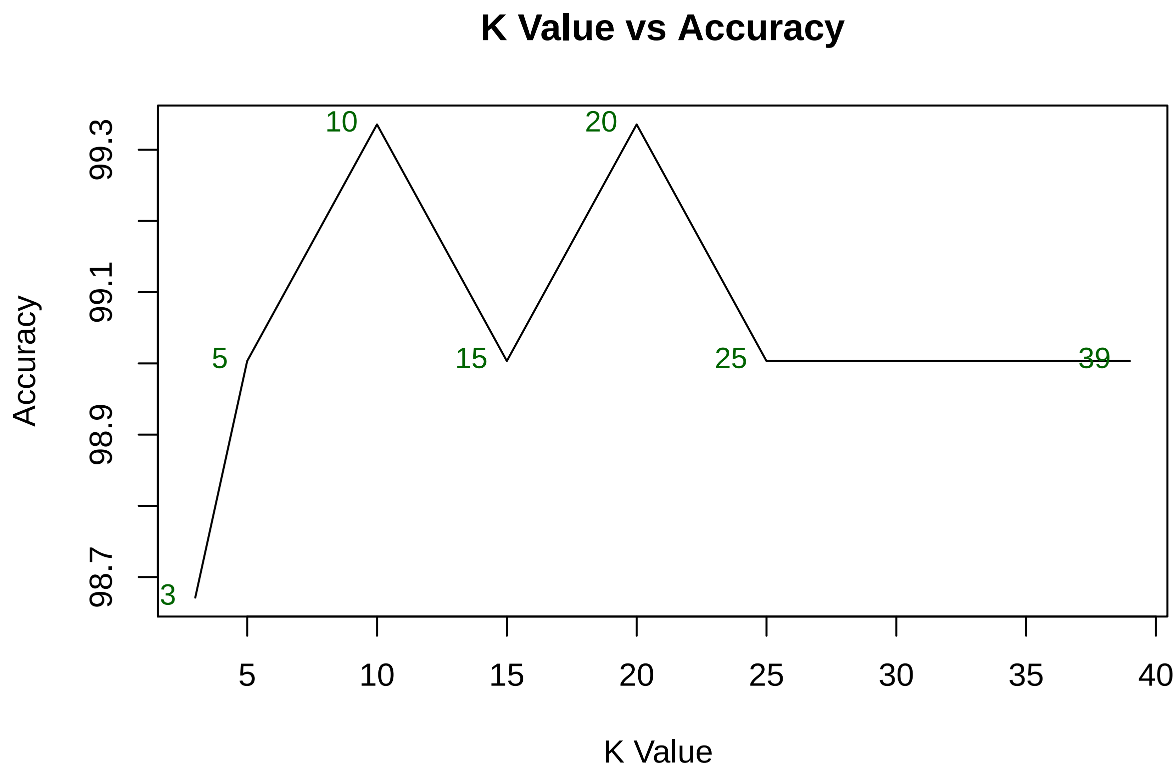
Plot for binary knn data

```r
binary.plot<- data.frame("K.Value"=c(3,5,10,15,20,25,39),
                         "Accuracy"=c(ACC.3,ACC.5,ACC.10,ACC.15,ACC.20,ACC.25,ACC.39)
                        )
library(caret)
```

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

```
## Loading required package: ggplot2
```

```r
plot(binary.plot,
     main="K Value vs Accuracy",
      xlab="K Value ", ylab="Accuracy",type="l"
     )
   text(binary.plot[, 'K.Value'], binary.plot[, 'Accuracy'],  binary.plot$K.Value,
     cex = 0.88, pos = 2, col = "darkgreen")
```



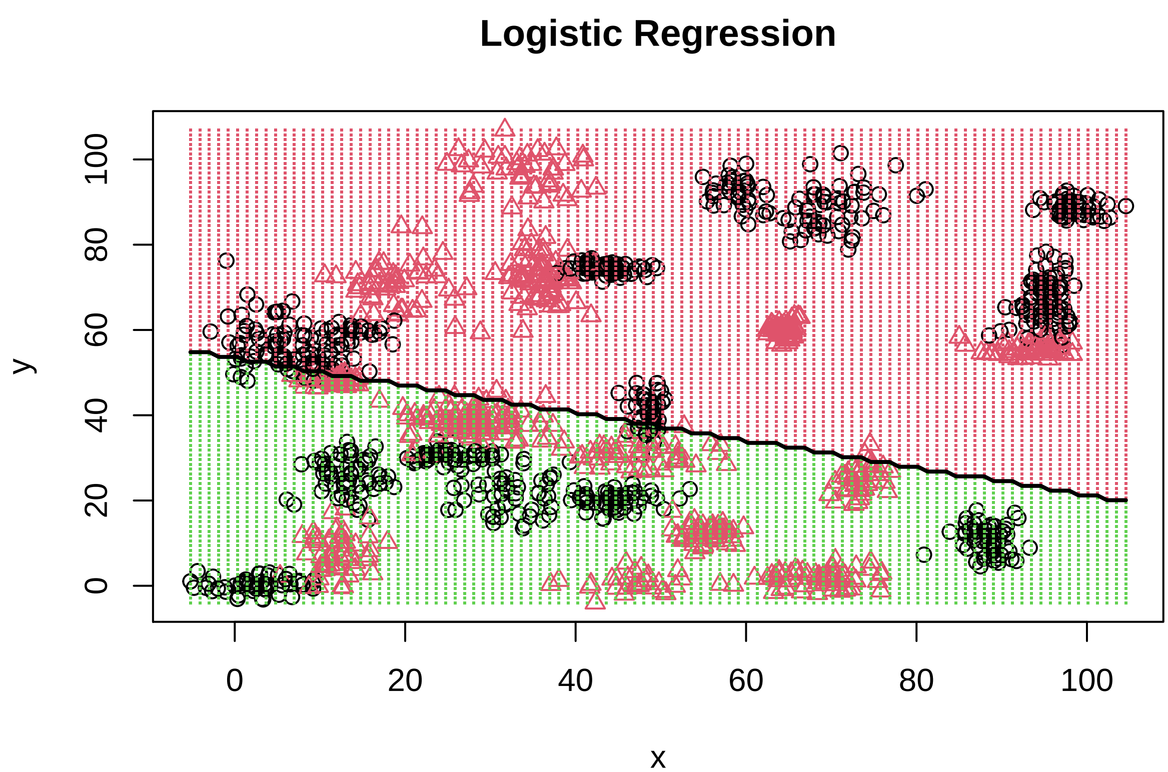
decision plot function

```
decisionplot <- function(model, data, class = NULL, predict_type = "class",
  resolution = 100, showgrid = TRUE, ...) {

  if(!is.null(class)) cl <- data[,class] else cl <- 1
  data <- data[,2:3]
  k <- length(unique(cl))

  plot(data, col = as.integer(cl)+1L, pch = as.integer(cl)+1L, ...)

  # make grid
  r <- sapply(data, range, na.rm = TRUE)
  xs <- seq(r[1,1], r[2,1], length.out = resolution)
  ys <- seq(r[1,2], r[2,2], length.out = resolution)
  g <- cbind(rep(xs, each=resolution), rep(ys, time = resolution))
  colnames(g) <- colnames(r)
  g <- as.data.frame(g)

  ### guess how to get class labels from predict
  ### (unfortunately not very consistent between models)
  p <- predict(model, g, type = predict_type)
  if(is.list(p)) p <- p$class
  p <- as.factor(p)
```

```
  if(showgrid) points(g, col = as.integer(p)+1L, pch = ".")

  z <- matrix(as.integer(p), nrow = resolution, byrow = TRUE)
  contour(xs, ys, z, add = TRUE, drawlabels = FALSE,
    lwd = 2, levels = (1:(k-1))+.5)

  invisible(z)
}
```

c. Decision boundry for binary dataset

```
library(lattice)


model <- glm(label ~., data = data.binary)
class(model) <- c("lr", class(model))
predict.lr <- function(object, newdata, ...)
  predict.glm(object, newdata, type = "response") > .5


decisionplot(model, data.binary, class = "label", main = "Logistic Regression")
```



Trinary

b - Trinary data set

```
set.seed(9850)
gp<-runif(nrow(data.trinary))
data.trinary<-data.trinary[order(gp),]
head(data.trinary)
```

```
##      label        x        y
## 216      0 30.10783 74.34244
## 405      1 38.91000 37.00775
## 316      0 41.34978 61.17379
## 804      1 37.79238 57.31985
## 1554     2 98.14194 80.07052
## 103      0 90.54824 36.35014
```

```
normalize<-function(x){
  return (
            (x - min(x))/max(x)-min(x)
         )
}
data.trinary.n<-as.data.frame(lapply(data.trinary[,c(2:3)], normalize))

str(data.trinary.n)
```

```
## 'data.frame':    1568 obs. of  2 variables:
##  $ x: num  10.6 10.7 10.7 10.7 11.3 ...
##  $ y: num  2.27 1.91 2.14 2.11 2.32 ...
```

```
summary(data.trinary.n)
```

```
##        x               y
##  Min.   :10.26   Min.   :1.541
##  1st Qu.:10.64   1st Qu.:1.900
##  Median :10.78   Median :2.084
##  Mean   :10.81   Mean   :2.086
##  3rd Qu.:10.97   3rd Qu.:2.298
##  Max.   :11.36   Max.   :2.556
```

```
r<-round(0.8*nrow(data.trinary.n))
l<-nrow(data.trinary.n)

data.train<-data.trinary.n[1:r,]
data.test<-data.trinary.n[r:l,]

data.train.target<-data.trinary[1:r,1]
data.test.target<-data.trinary[r:l,1]

k_value<-round(sqrt(nrow(data.trinary)))
```

calculatig knn and accuracy for each k =3 , 5 ,10 ,15 ,20 , 25 and 39

```r
library("class")



knn.3<- knn(train = data.train, test=data.test,cl=data.train.target, k=3)
ACC.3<-100 * sum( data.test.target == knn.3) / NROW(data.test.target)

table(data.test.target,knn.3)
```

```
##                  knn.3
## data.test.target   0   1   2
##                0  65   9   4
##                1   6 126   8
##                2   8   4  85
```

```r
ACC.3
```

```
## [1] 87.61905
```

```r
knn.5<- knn(train = data.train, test=data.test,cl=data.train.target, k=5)
ACC.5<-100 * sum( data.test.target == knn.5) / NROW(data.test.target)

table(data.test.target,knn.5)
```

```
##                  knn.5
## data.test.target   0   1   2
##                0  65   9   4
##                1   6 127   7
##                2   8   5  84
```

```r
ACC.5
```

```
## [1] 87.61905
```

```r
knn.10<- knn(train = data.train, test=data.test,cl=data.train.target, k=10)
ACC.10<-100 * sum( data.test.target == knn.10) / NROW(data.test.target)

table(data.test.target,knn.10)
```

```
##                  knn.10
## data.test.target   0   1   2
##                0  68   8   2
##                1   7 124   9
##                2  10   5  82
```

```r
ACC.10
```

```
## [1] 86.98413
```

```r
knn.15<- knn(train = data.train, test=data.test,cl=data.train.target, k=15)
ACC.15<-100 * sum( data.test.target == knn.15) / NROW(data.test.target)

table(data.test.target,knn.15)
```

```
##                   knn.15
## data.test.target   0   1   2
##                  0  66  10   2
##                  1   6 125   9
##                  2  12   4  81
```

```r
ACC.15
```

```
## [1] 86.34921
```

```r
knn.20<- knn(train = data.train, test=data.test,cl=data.train.target, k=20)
ACC.20<-100 * sum( data.test.target == knn.20) / NROW(data.test.target)

table(data.test.target,knn.20)
```

```
##                   knn.20
## data.test.target   0   1   2
##                  0  65  10   3
##                  1   6 126   8
##                  2  11   3  83
```

```r
ACC.20
```

```
## [1] 86.98413
```

```r
knn.25<- knn(train = data.train, test=data.test,cl=data.train.target, k=25)
ACC.25<-100 * sum( data.test.target == knn.25) / NROW(data.test.target)

table(data.test.target,knn.25)
```

```
##                   knn.25
## data.test.target   0   1   2
##                  0  63  13   2
##                  1   8 124   8
##                  2  10   3  84
```

```r
ACC.25
```

```
## [1] 86.03175
```

```r
knn.39<- knn(train = data.train, test=data.test,cl=data.train.target, k=k_value)
ACC.39<-100 * sum( data.test.target == knn.39) / NROW(data.test.target)

table(data.test.target,knn.39)
```

```
##                  knn.39
## data.test.target   0   1   2
##                0  64  11   3
##                1   9 121  10
##                2  13   1  83
```

```r
ACC.39
```

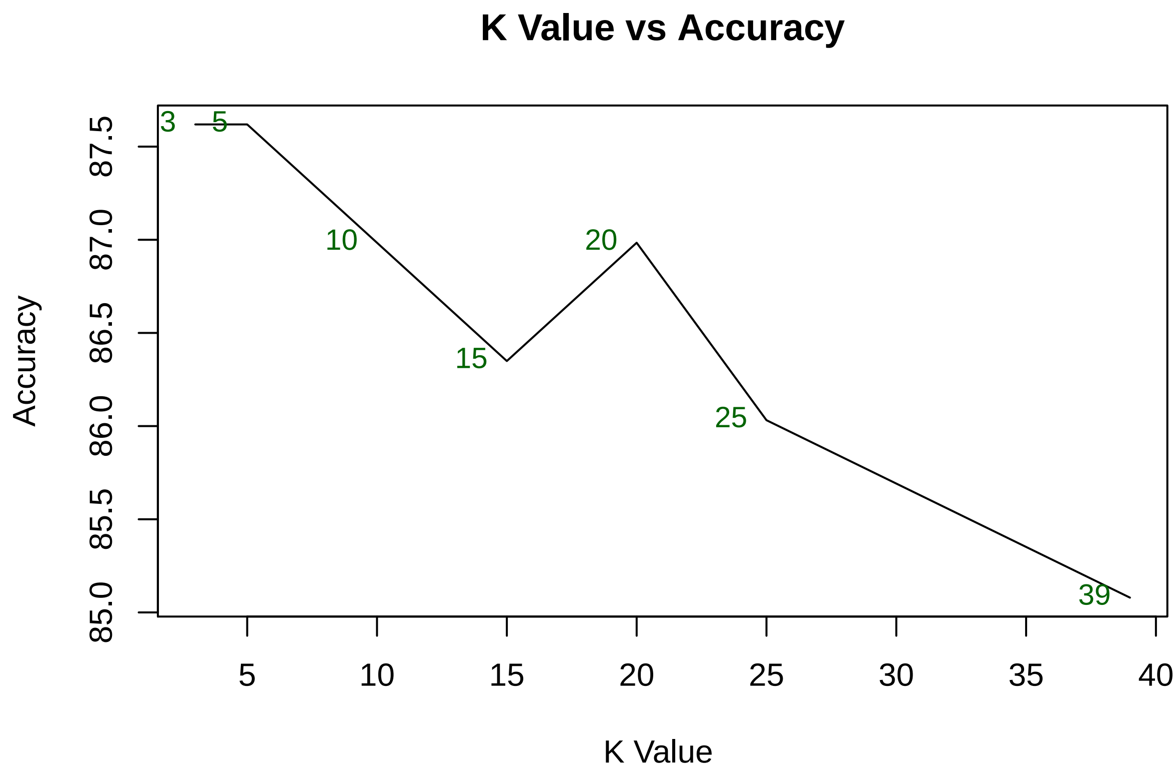
```
## [1] 85.07937
```

Plot for trinary knn data

```r
library(purrr)
```

```
##
## Attaching package: 'purrr'

## The following object is masked from 'package:caret':
##
##     lift
```

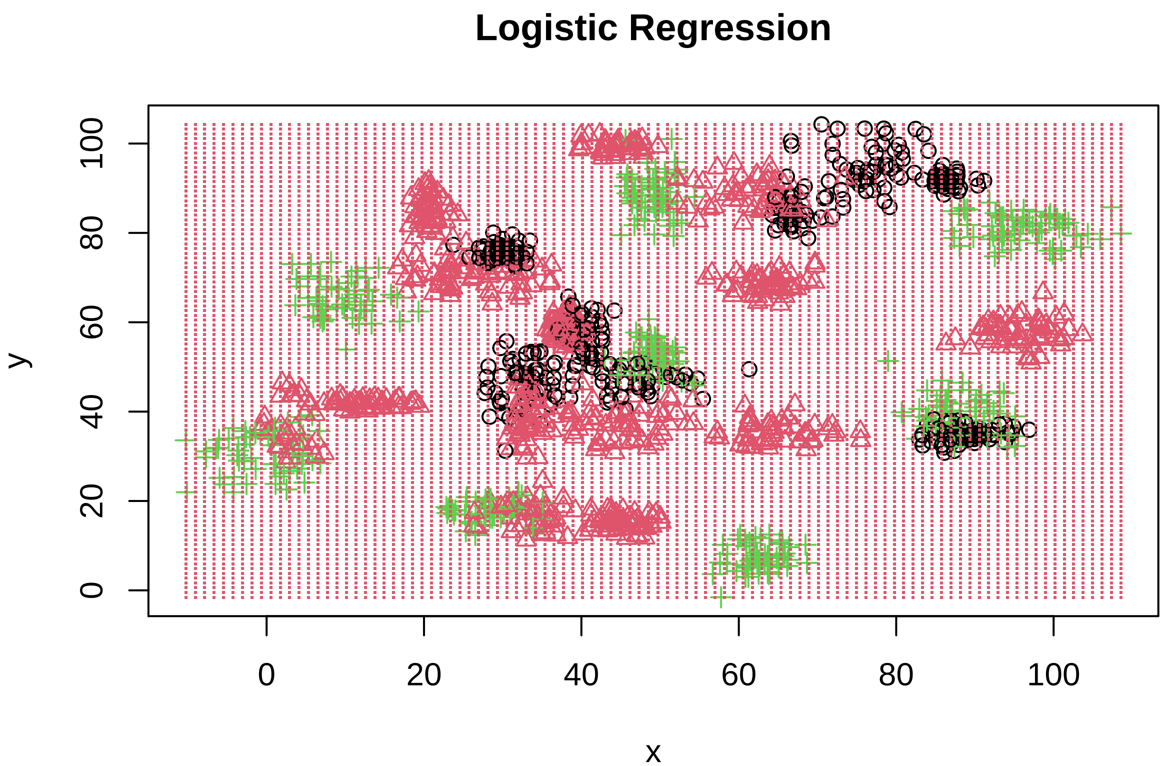
```r
trinary.plot<- data.frame("K.Value"=c(3,5,10,15,20,25,39),
                          "Accuracy"=c(ACC.3,ACC.5,ACC.10,ACC.15,ACC.20,ACC.25,ACC.39)
                         )
library(caret)

plot(trinary.plot,
     main="K Value vs Accuracy",
      xlab="K Value ", ylab="Accuracy",type="l"
     )
   text(trinary.plot[, 'K.Value'], trinary.plot[, 'Accuracy'],
        trinary.plot$K.Value,
     cex = 0.88, pos = 2, col = "darkgreen")
```



Decision boundry for trinary dataset

```
model <- glm(label ~., data = data.trinary)
class(model) <- c("lr", class(model))
predict.lr <- function(object, newdata, ...)
  predict.glm(object, newdata, type = "response") > .5


decisionplot(model, data.trinary, class = "label", main = "Logistic Regression")
```

```
## Warning in contour.default(xs, ys, z, add = TRUE, drawlabels = FALSE, lwd = 2, :
## all z values are equal
```



data doesn't look suitable for linear classifier