DSC520 Week10 Exercise 11.2.1

Venkat Jagadeesh Jampani

March 5th 2022

Project: Fit a Logistic Regression model to Binary Classifier Dataset

```
# Load the packages
library(caTools)
library(ggplot2)
setwd("/Users/Jagadeesh/Documents/GitHub/dsc520")

# Load Binary Classifier Dataset
binary_df <- read.csv("data/binary-classifier-data.csv")

# Check structure of binary_df
str(binary_df)
```

```
## 'data.frame':	1498 obs. of  3 variables:
##  $ label: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ x    : num  70.9 75 73.8 66.4 69.1 ...
##  $ y    : num  83.2 87.9 92.2 81.1 84.5 ...
```

```
# Check sample rows of binary_df
head(binary_df)
```

```
##   label        x        y
## 1     0 70.88469 83.17702
## 2     0 74.97176 87.92922
## 3     0 73.78333 92.20325
## 4     0 66.40747 81.10617
## 5     0 69.07399 84.53739
## 6     0 72.23616 86.38403
```

```
# Load the trinary classifier data set to dataframe
trinary_df <- read.csv("data/trinary-classifier-data.csv")

# Check structure of trinary_df
str(trinary_df)
```

```
## 'data.frame':	1568 obs. of  3 variables:
##  $ label: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ x    : num  30.1 31.3 34.1 32.6 34.7 ...
##  $ y    : num  39.6 51.8 49.3 41.2 45.5 ...
```

```r
# Check sample rows
head(trinary_df)
```

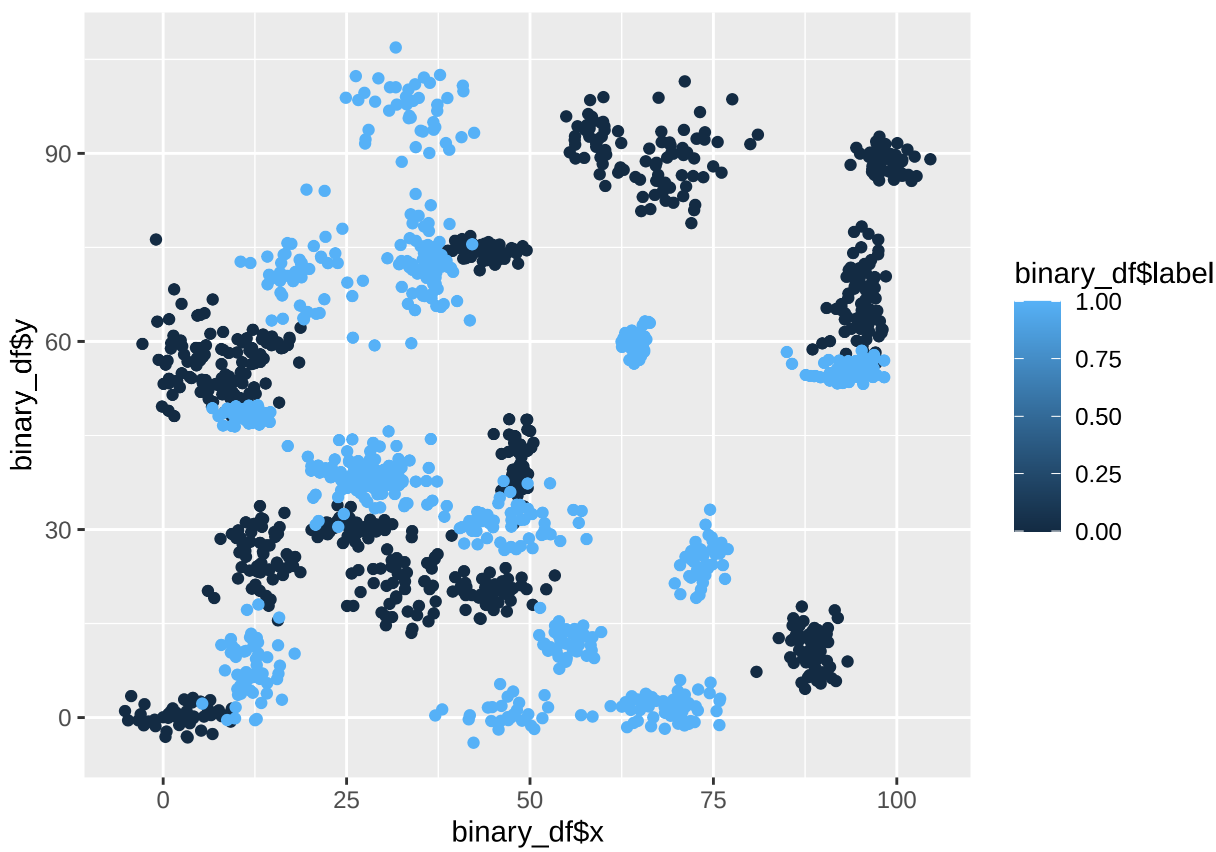
```
##   label        x        y
## 1     0 30.08387 39.63094
## 2     0 31.27613 51.77511
## 3     0 34.12138 49.27575
## 4     0 32.58222 41.23300
## 5     0 34.65069 45.47956
## 6     0 33.80513 44.24656
```

```r
# i. Plot the data from each dataset using scatter plot
ggplot(binary_df, aes(x=binary_df$x, y=binary_df$y)) + geom_point(aes(color = binary_df$label))
```

```
## Warning: Use of `binary_df$label` is discouraged. Use `label` instead.

## Warning: Use of `binary_df$x` is discouraged. Use `x` instead.

## Warning: Use of `binary_df$y` is discouraged. Use `y` instead.
```

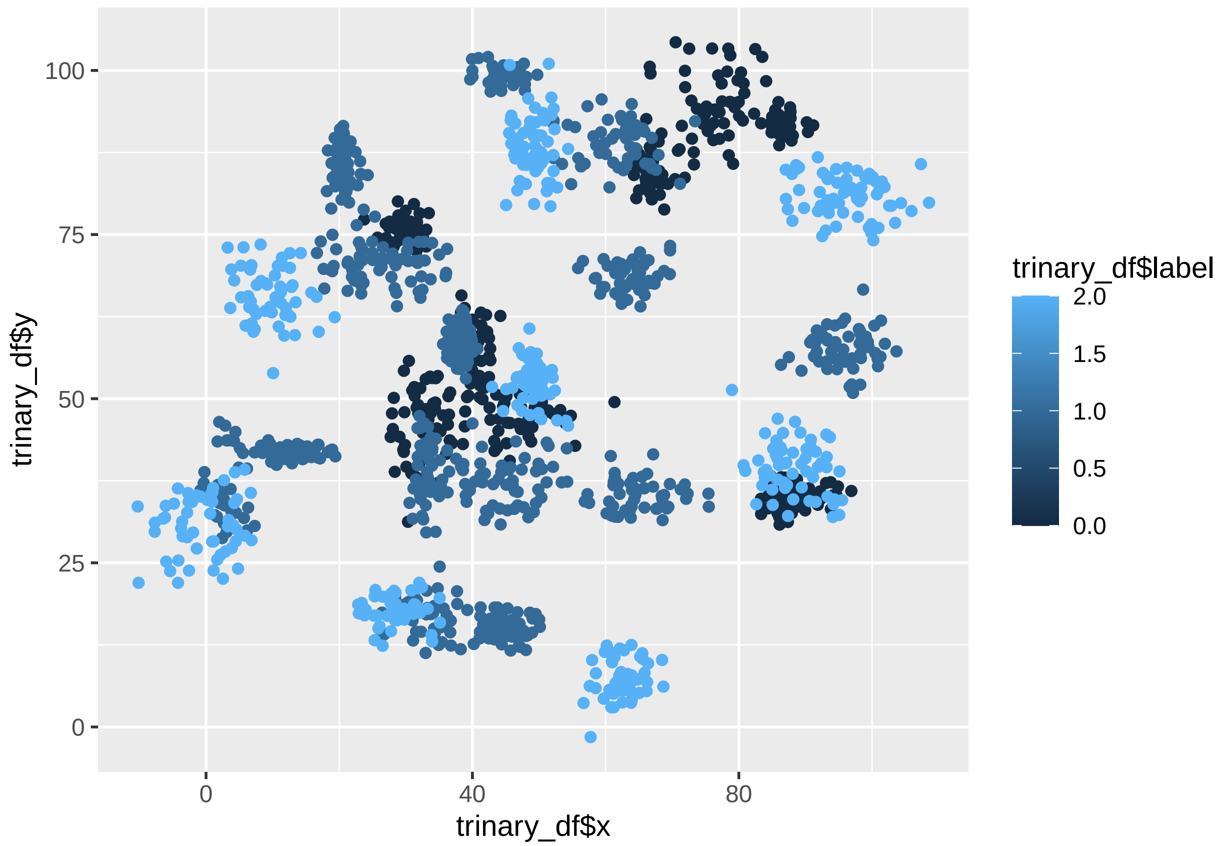



```r
ggplot(trinary_df, aes(x=trinary_df$x, y=trinary_df$y)) + geom_point(aes(color = trinary_df$label))
```

```
## Warning: Use of `trinary_df$label` is discouraged. Use `label` instead.
```

```
## Warning: Use of `trinary_df$x` is discouraged. Use `x` instead.

## Warning: Use of `trinary_df$y` is discouraged. Use `y` instead.
```




```
# Normalization of binary_df
normalize <- function(x) { return ((x - min(x)) / (max(x) - min(x))) }
binary_df.n = as.data.frame(lapply(binary_df[,2:3], normalize))
binary_df.n = as.data.frame(lapply(binary_df[,2:3], normalize))
set.seed(123)

# Random sample of 70% of data
dat.d <- sample(1:nrow(binary_df.n), size = nrow(binary_df.n)*0.7, replace = FALSE)

# Create test and train datasets
train.binary_df <- binary_df[dat.d,]
test.binary_df <- binary_df[-dat.d,]

# Create separate dataframe for label feature
train.binary_df_label <- binary_df[dat.d,1]
test.binary_df_label <- binary_df[-dat.d,1]

# Find no. of observations
NROW(train.binary_df)
```

```
## [1] 1048
```

```r
# From above, we have 700 observations in our training dataset. The square root of 700 is about 26.45.
# Therefore, we'll create 2 models.One with 'K' value 26 and other with 'K' value 27

library(class)
knn.binary_df.1 <- knn(train=train.binary_df, test=test.binary_df, cl=train.binary_df_label, k=1)
# Calculate accuracy of the models
# Caculate the proportion of correct classification for k=32,33
ACC.binary_df.1 <- 100*sum(test.binary_df_label == knn.binary_df.1)/NROW(test.binary_df_label)
ACC.binary_df.1
```

```
## [1] 98.22222
```

```r
# Accuracy is 98.22
# Check prediction against actual value in tabular form for k=32
table(knn.binary_df.1, test.binary_df_label)
```

```
##                  test.binary_df_label
## knn.binary_df.1   0   1
##                 0 227   4
##                 1   4 215
```

```r
# Use confusion matrix to calculate the accuracy.
library(caret)
```

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

```r
confusionMatrix(table(knn.binary_df.1, test.binary_df_label))
```

```
## Confusion Matrix and Statistics
##
##                  test.binary_df_label
## knn.binary_df.1   0   1
##                 0 227   4
##                 1   4 215
##
##                  Accuracy : 0.9822
##                    95% CI : (0.9653, 0.9923)
##     No Information Rate : 0.5133
##     P-Value [Acc > NIR] : <2e-16
##
##                     Kappa : 0.9644
##
##  Mcnemar's Test P-Value : 1
##
##               Sensitivity : 0.9827
##               Specificity : 0.9817
##            Pos Pred Value : 0.9827
##            Neg Pred Value : 0.9817
##                Prevalence : 0.5133
##            Detection Rate : 0.5044
##      Detection Prevalence : 0.5133
```

```
##       Balanced Accuracy : 0.9822
##
##        'Positive' Class : 0
##
```

```r
# Normalization of trinary_df
normalize <- function(x) { return ((x - min(x)) / (max(x) - min(x))) }
trinary_df.n = as.data.frame(lapply(trinary_df[,2:3], normalize))
trinary_df.n = as.data.frame(lapply(trinary_df[,2:3], normalize))
set.seed(123)
# Random sample of 70% of data
dat.d <- sample(1:nrow(trinary_df.n), size = nrow(trinary_df.n)*0.7, replace = FALSE)
# Create test and train datasets
train.trinary_df <- trinary_df[dat.d,]
test.trinary_df <- trinary_df[-dat.d,]
# Create separate dataframe for label feature
train.trinary_df_label <- trinary_df[dat.d,1]
test.trinary_df_label <- trinary_df[-dat.d,1]

# Find no. of observations
NROW(train.trinary_df)
```

```
## [1] 1097
```

```r
library(class)
knn.trinary_df.1 <- knn(train=train.trinary_df, test=test.trinary_df, cl=train.trinary_df_label, k=1)

# Calculate accuracy of the models
# Caculate the proportion of correct classification for k=32,33
ACC.trinary_df.1 <- 100*sum(test.trinary_df_label == knn.trinary_df.1)/NROW(test.trinary_df_label)
ACC.trinary_df.1
```

```
## [1] 95.75372
```

```r
# Accuracy is 95.75
# Check prediction against actual value in tabular form for k=32
table(knn.trinary_df.1, test.trinary_df_label)
```

```
##                  test.trinary_df_label
## knn.trinary_df.1   0   1   2
##                0 131   7   1
##                1   3 185   7
##                2   0   2 135
```

```r
# Use confusion matrix to calculate the accuracy.
library(caret)
confusionMatrix(table(knn.trinary_df.1, test.trinary_df_label))
```

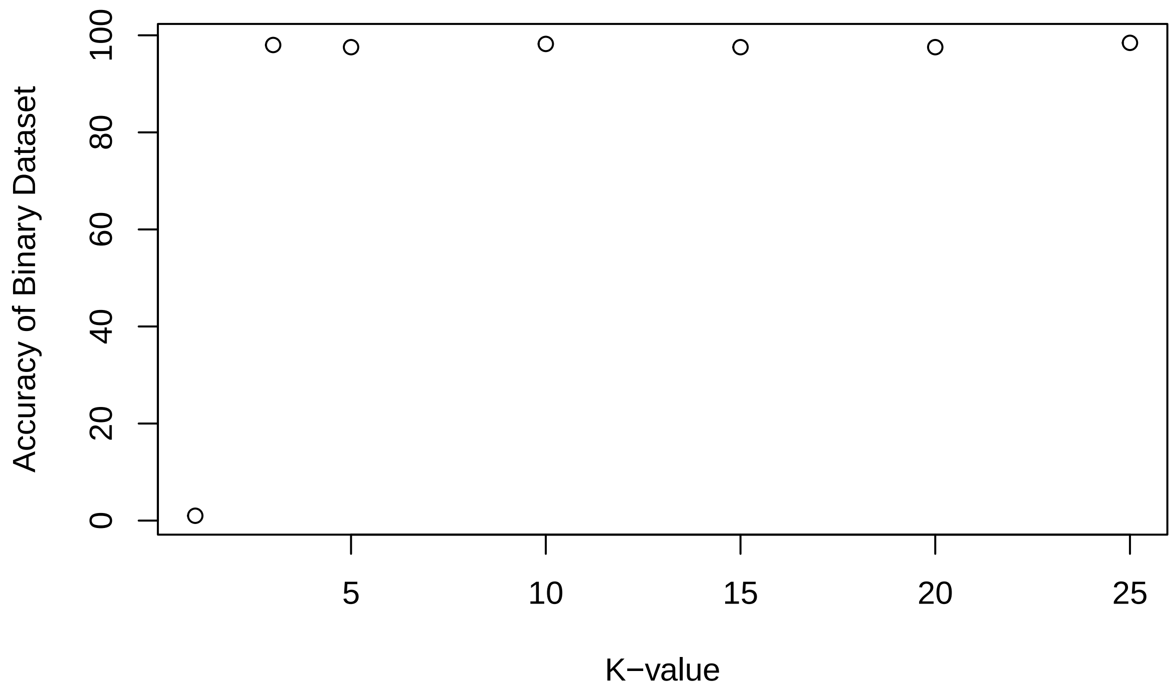
```
## Confusion Matrix and Statistics
##
##                  test.trinary_df_label
```

```
## knn.trinary_df.1   0   1   2
##                0 131   7   1
##                1   3 185   7
##                2   0   2 135
##
## Overall Statistics
##
##                Accuracy : 0.9575
##                  95% CI : (0.9352, 0.9739)
##     No Information Rate : 0.4119
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9354
##
##  Mcnemar's Test P-Value : 0.1461
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2
## Sensitivity            0.9776   0.9536   0.9441
## Specificity            0.9763   0.9639   0.9939
## Pos Pred Value         0.9424   0.9487   0.9854
## Neg Pred Value         0.9910   0.9674   0.9760
## Prevalence             0.2845   0.4119   0.3036
## Detection Rate         0.2781   0.3928   0.2866
## Detection Prevalence   0.2951   0.4140   0.2909
## Balanced Accuracy      0.9769   0.9588   0.9690
```

```r
# ii Fit a k nearest neighbors' model for each dataset for k=3, k=5, k=10, k=15, k=20, and k=25. Comput
# Accuracy level of binary dataset
j <- 1
k.optm <- 1
for (i in c(3,5,10,15,20,25)){
  knn.mod <- knn(train=train.binary_df, test=test.binary_df, cl=train.binary_df_label, k=i)
  k.optm[i] <- 100 * sum(test.binary_df_label == knn.mod)/NROW(test.binary_df_label)
  k <- i
  j <- j + 1
  cat(k,'=',k.optm[i],'')}
```

```
## 3 = 98 5 = 97.55556 10 = 98.22222 15 = 97.55556 20 = 97.55556 25 = 98.44444
```

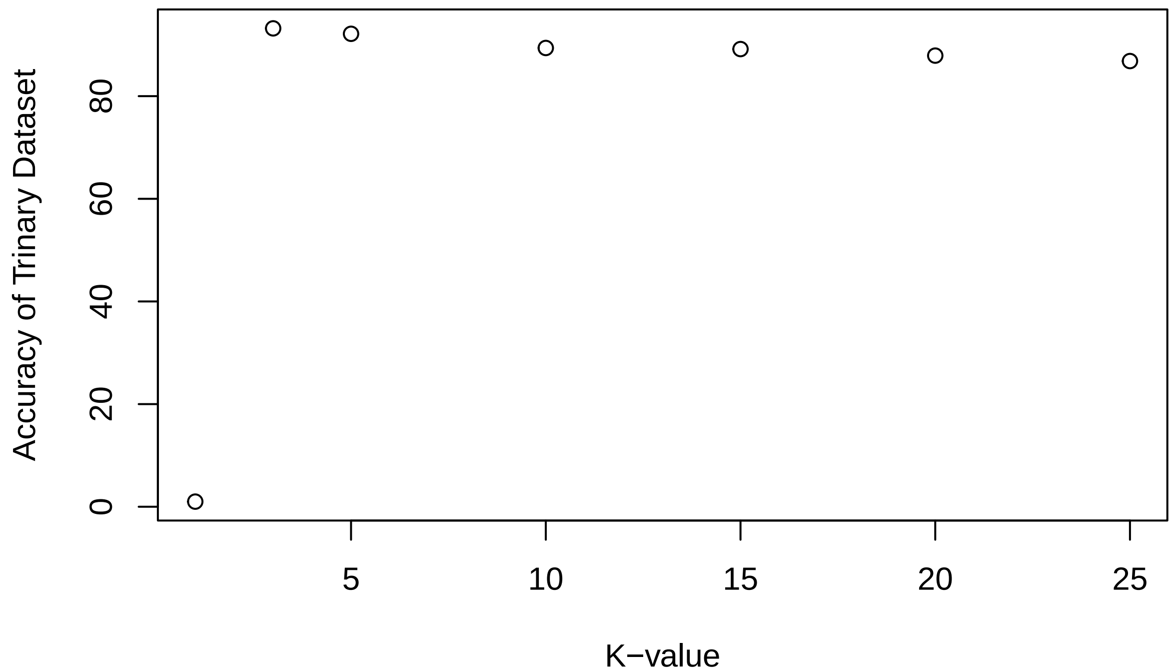
```r
# Accuracy Plot
plot(k.optm, type="b",xlab="K-value",ylab="Accuracy of Binary Dataset")
```




```r
# Accuracy level of trinary dataset
j <- 1
k.optm <- 1
for (i in c(3,5,10,15,20,25)){
  knn.mod <- knn(train=train.trinary_df, test=test.trinary_df, cl=train.trinary_df_label, k=i)
  k.optm[i] <- 100 * sum(test.trinary_df_label == knn.mod)/NROW(test.trinary_df_label)
  k <- i
  j <- j + 1
  cat(k,'=',k.optm[i],'')}
```

```
## 3 = 93.20594 5 = 92.14437 10 = 89.38429 15 = 89.17197 20 = 87.89809 25 = 86.83652
```

```r
# Accuracy Plot
plot(k.optm, type="b",xlab="K-value",ylab="Accuracy of Trinary Dataset")
```

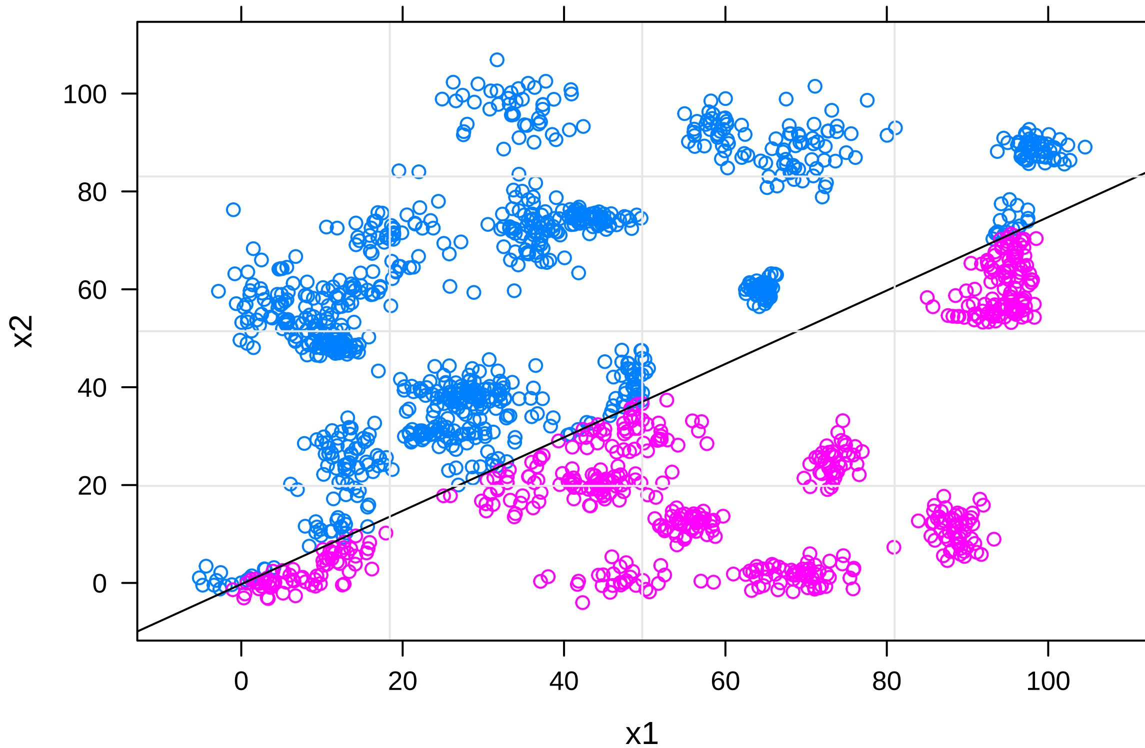



```
# i. Looking back at the plots of the data, do you think a linear classifier would work well on these d
x1=binary_df[2]
x2=binary_df[3]
y <- sign(3 * x1 - 4 * x2 - 1)
y[y == -1] <- 0
df <- cbind.data.frame(y,x1,x2)
names(df)[1] <- 'y'
names(df)[2] <- 'x1'
names(df)[3] <- 'x2'
mdl <- glm(y ~ . , data=df,family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
slope <- coef(mdl)[2]/(-coef(mdl)[3])
intercept <- coef(mdl)[1]/(-coef(mdl)[3])
library(lattice)
xyplot(x2 ~ x1,data=df, groups=y,panel=function(...){
  panel.xyplot(...)
  panel.abline(intercept, slope)
  panel.grid(...)
})
```

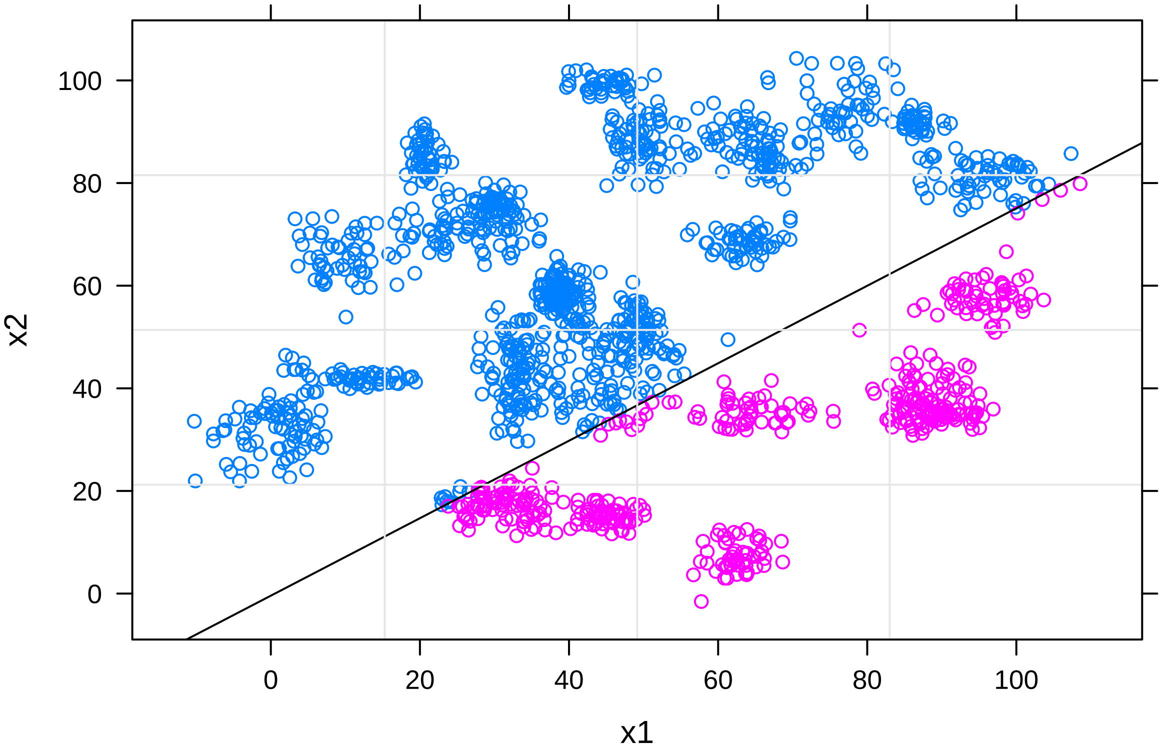



```
x1= trinary_df[2]
x2= trinary_df[3]
y <- sign(3 * x1 - 4 * x2 - 1)
y[y == -1] <- 0
df <- cbind.data.frame(y,x1,x2)
names(df)[1] <- 'y'
names(df)[2] <- 'x1'
names(df)[3] <- 'x2'
mdl <- glm(y ~ . , data=df,family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
slope <- coef(mdl)[2]/(-coef(mdl)[3])
intercept <- coef(mdl)[1]/(-coef(mdl)[3])
library(lattice)
xyplot(x2 ~ x1,data=df, groups=y,panel=function(...){
  panel.xyplot(...)
  panel.abline(intercept, slope)
  panel.grid(...)
})
```




```
# Looking at the plots, I think that the linear classifier would work well on binary data set but not o
# data set
# ii. How does the accuracy of your logistic regression classifier from last week compare?  Why is the 
## The accuracy of logistic regression model was 67% but the accuracy of knn model is 98% of binary data
## The difference in accuracy is due to the non-linearness of the data in the input datasets.
## KNN fits good for the non-linear dataset and hence it is more suitable model in our case.
```