Week11_12_Part1_Introduction to Machine Learning KNN

Janine Par

2022-05-31

```
#Fit a Logistic Regression Model

## Set the working directory to the root of your DSC 520 directory
setwd("C:/Users/janin/OneDrive/Documents/R_repo/dsc520/")

## Load the `data/binaryclassifierdata`
binaryclass_df <- read.csv("data/binary-classifier-data.csv", header=TRUE, comment.char = "@")

## Load the `data/Trinary Classifier Data`
trinaryclass_df <- read.csv("data/trinary-classifier-data.csv", header=TRUE, comment.char = "@")

str(binaryclass_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 ...
```

```
str(trinaryclass_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 ...
```

```
table(binaryclass_df$label)
```

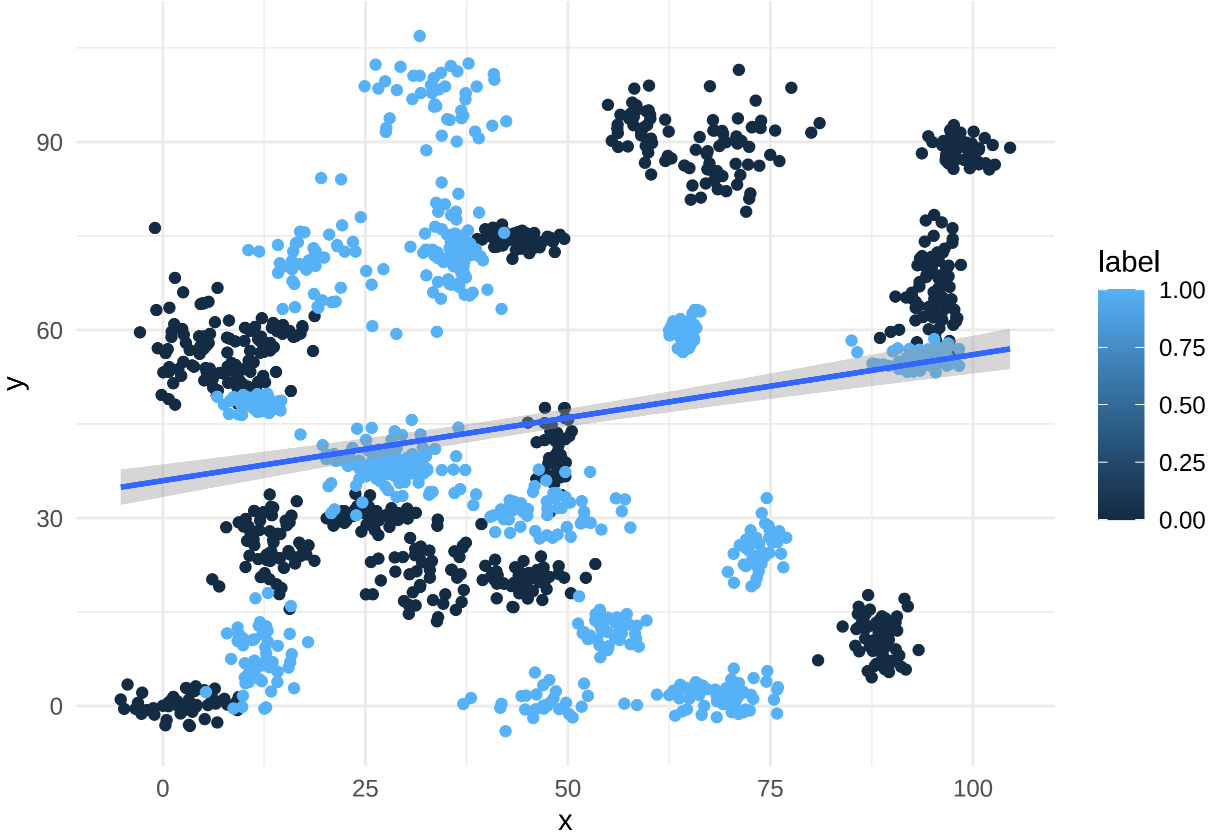
```
##
##   0   1
## 767 731
```

```
table(trinaryclass_df$label)
```

```
##
##   0   1   2
## 394 722 452
```

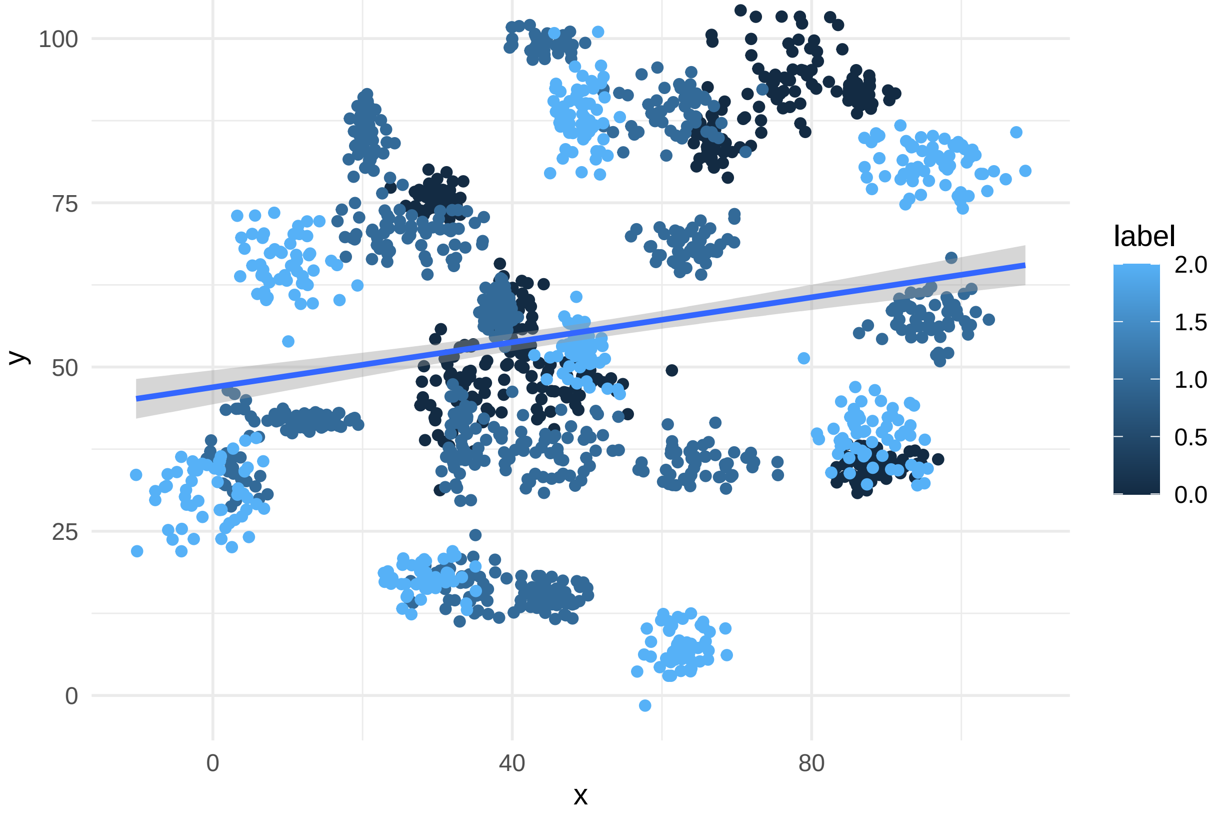
```
#Plot the data from each dataset using a scatter plot.

ggplot(binaryclass_df, aes(x=x, y=y, color=label)) + geom_point() + geom_smooth(method="lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```
ggplot(trinaryclass_df, aes(x=x, y=y, color=label)) + geom_point() + geom_smooth(method="lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



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

binaryclass_df_n <- as.data.frame(lapply(binaryclass_df[2:3], normalize))  #Normalize X and Y

trinaryclass_df_n <- as.data.frame(lapply(trinaryclass_df[2:3], normalize))  #Normalize X and Y
```

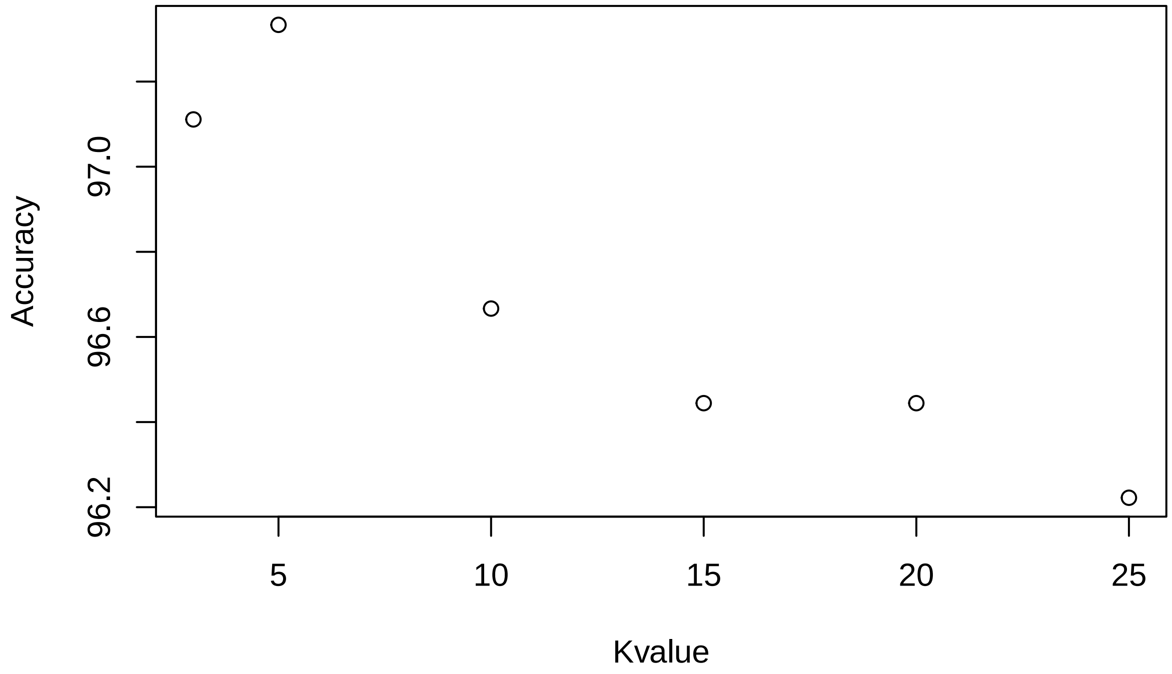
#Work on Binary Class datasets

```r
#Split data

set.seed(1234)

#Get random numbers for Training Data
size <- floor(0.7*nrow(binaryclass_df))

train_ind <- sample(seq_len(nrow(binaryclass_df)), size = size)

bcsplit_train.label <- binaryclass_df[train_ind,1 ]

bcsplit_test.label <- binaryclass_df[-train_ind,1 ]

bcsplit_train <- binaryclass_df_n[train_ind,]

bcsplit_test <- binaryclass_df_n[-train_ind,]
```

```
bcsplit_predict <- knn(train = bcsplit_train,
                       test=bcsplit_test,
                       cl=bcsplit_train.label,
                       k=round(sqrt(nrow(bcsplit_train)))) # K-32 initial run

acc_bcsplit_32 <- 100 * sum(bcsplit_test.label == bcsplit_predict)/NROW(bcsplit_test.label)

#Confusion Matrix
confmatrix <- table(Actual_value=bcsplit_test.label, Predicted_Value= bcsplit_predict)

# Accuracy
(confmatrix [[1,1]] + confmatrix [[2,2]])/sum(confmatrix) * 100
```

```
## [1] 96.44444
```

```
knn_val <- c(3,5,10,15,20,25)

bcs_knn_model <- data.frame()

i=1

for (kv in knn_val)
{
  bcsplit_predict<- knn(train = bcsplit_train,
                        test=bcsplit_test,
                        cl=bcsplit_train.label,
                        k=kv)

  confmatrix <- table(Actual_value=bcsplit_test.label, Predicted_Value= bcsplit_predict)
  kvaccuracy <-   100 * sum(bcsplit_test.label == bcsplit_predict)/NROW(bcsplit_test.label)  #((confmat
  kvalue <- kv
  bcs_knn_model <- rbind(bcs_knn_model, c(kvalue, kvaccuracy))
  names(bcs_knn_model) <- c("Kvalue", "Accuracy")
}

#ggplot(data=bcs_knn_model, aes(x=kvalue, y=accuracy)) + geom_point()

plot(bcs_knn_model)
```



```
# K value and Accuracy
bcs_knn_model
```

```
##   Kvalue Accuracy
## 1      3 97.11111
## 2      5 97.33333
## 3     10 96.66667
## 4     15 96.44444
## 5     20 96.44444
## 6     25 96.22222
```

```
#Binary Class Linear Classifier from last week (week 10)

binaryclass.model <- glm(label~x+y, data=binaryclass_df, family=binomial())

summary (binaryclass.model)
```

```
##
## Call:
## glm(formula = label ~ x + y, family = binomial(), data = binaryclass_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3728  -1.1697  -0.9575   1.1646   1.3989
##
```

```
## Coefficients:
##              Estimate Std. Error z value  Pr(>|z|)    
## (Intercept)  0.424809   0.117224   3.624   0.00029 ***
## x           -0.002571   0.001823  -1.411   0.15836    
## y           -0.007956   0.001869  -4.257 0.0000207 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2075.8  on 1497  degrees of freedom
## Residual deviance: 2052.1  on 1495  degrees of freedom
## AIC: 2058.1
##
## Number of Fisher Scoring iterations: 4
```

```
bcsplit <- sample.split(binaryclass_df, SplitRatio = 0.8)

bcsplit_train <- subset(binaryclass_df,tssplit='True')

bcsplit_test <- subset(binaryclass_df,tssplit='False')

#Predict
res.train <- predict(binaryclass.model,bcsplit_train,type ="response")

res.test <- predict(binaryclass.model,bcsplit_test,type ="response")

confmatrix <- table(Actual_value=bcsplit_train$label, Predicted_Value= res.train > 0.5)

(confmatrix [[1,1]] + confmatrix [[2,2]])/sum(confmatrix)*100
```

```
## [1] 58.34446
```

#Work on Trinary Class datasets

```
#Split data

set.seed(1234)

#Get random numbers for Training Data
size <- floor(0.7*nrow(trinaryclass_df))

train_ind <- sample(seq_len(nrow(trinaryclass_df)), size = size)

trsplit_train.label <- trinaryclass_df[train_ind,1 ]

trsplit_test.label <- trinaryclass_df[-train_ind,1 ]

trsplit_train <- trinaryclass_df_n[train_ind, ]

trsplit_test <- trinaryclass_df_n[-train_ind,]

trsplit_predict <- knn(train = trsplit_train,
                       test=trsplit_test,
                       cl=trsplit_train.label,
                       k=round(sqrt(nrow(trsplit_train)))) # K-32 initial run

summary(trsplit_predict)
```

```
##   0   1   2
## 140 200 131
```

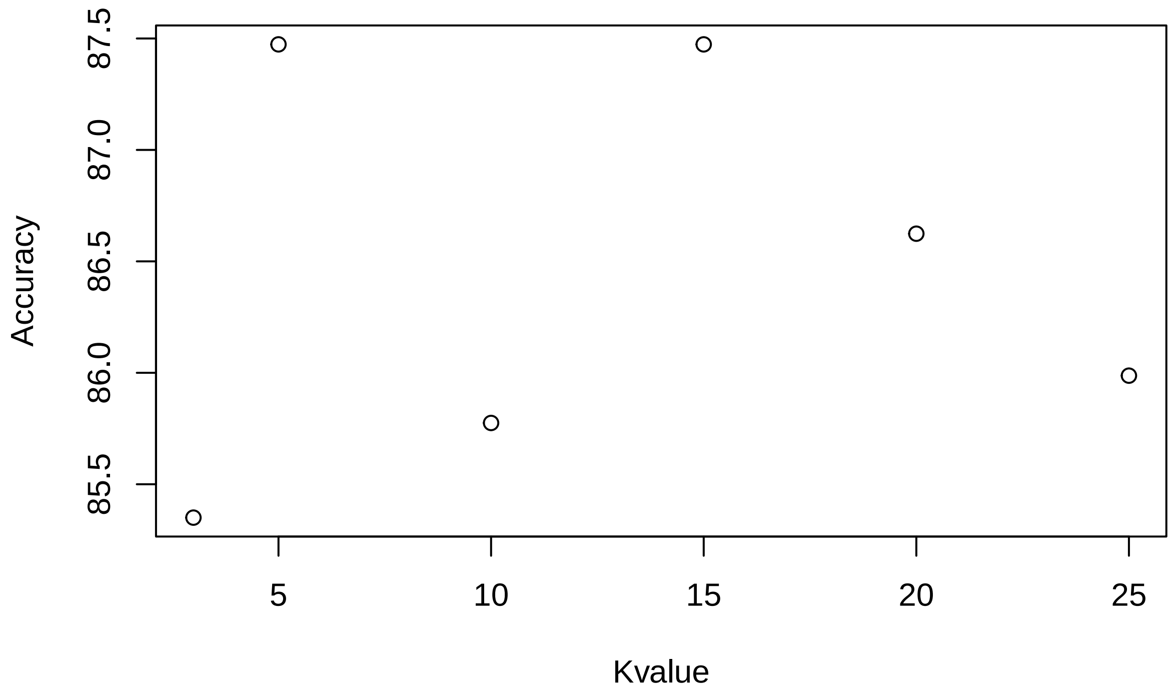
```
#COnfusion Matrix
confmatrix <- table(Actual_value=trsplit_test.label, Predicted_Value= trsplit_predict)

confmatrix
```

```
##             Predicted_Value
## Actual_value   0   1   2
##            0 112  18   3
##            1  12 177  13
##            2  16   5 115
```

```
# Accuracy
(confmatrix [[1,1]] + confmatrix [[2,2]])/sum(confmatrix) * 100
```

```
## [1] 61.35881
```

```
#Fit a k nearest neighbors' model for each dataset for k=3, k=5, k=10, k=15, k=20, and k=25. Compute th

knn_val <- c(3,5,10,15,20,25)

tr_knn_model <- data.frame()

i=1

for (kv in knn_val)
{
  trsplit_predict<- knn(train = trsplit_train,
                        test=trsplit_test,
                        cl=trsplit_train.label,
                        k=kv)

  confmatrix <- table(Actual_value=trsplit_test.label, Predicted_Value= trsplit_predict)
  kvaccuracy <-   100 * sum(trsplit_test.label == trsplit_predict)/NROW(trsplit_test.label)
  kvalue <- kv
  tr_knn_model <- rbind(tr_knn_model, c(kvalue, kvaccuracy))
  names(tr_knn_model) <- c("Kvalue", "Accuracy")
}

#ggplot(data=bcs_knn_model, aes(x=kvalue, y=accuracy)) + geom_point()

plot(tr_knn_model)
```



```
tr_knn_model
```

```
##   Kvalue Accuracy
## 1      3 85.35032
## 2      5 87.47346
## 3     10 85.77495
## 4     15 87.47346
## 5     20 86.62420
## 6     25 85.98726
```

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

I do not think that the linear classifier will work well with the Binary and Trinary dataset because the plot shows non-linearity of the data.

#How does the accuracy of your logistic regression classifier from last week compare? Why is the accuracy different between these two methods?

Last week, Binary dataset has lower accuracy of 58.3% compare to KNN model which resulted to higher accuracy and this is because KNN model fits the data more because of it's Non-Linearity.