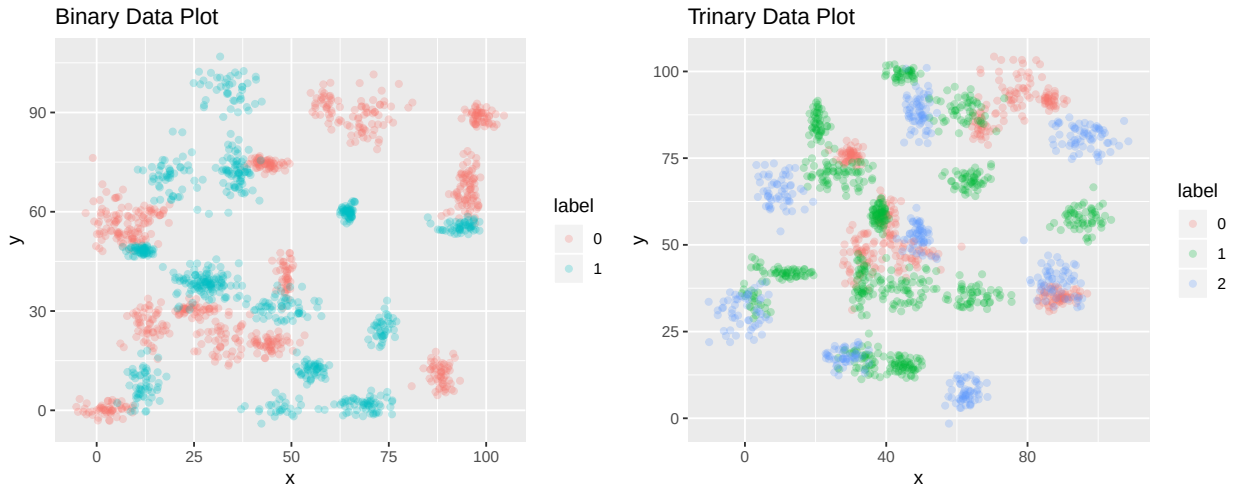
Assignment_9.1_HillZach

Zach Hill

May 11, 2019

A: Plotted data from original datasets



B.1: Constructing Test and Training datasets

First we separate our datasets into training and testing datasets. In most cases we would then normalize the data due to differing units of measure but in this case I have elected not to since the numbers dont really have any defined unit of measure. I have also elected to use a randomly generated dataset each time the datasets are built. This will result in a better picture of the data as more iterations are run as the accuracy will change slightly and eventually regress towards the mean. Additionally if you change the size of the coefficient in the sample and recreate the datasets, different accuracies will show.

```
bin_sample <- sample(seq_len(nrow(bin)), size = floor(.75 * nrow(bin)))
bin_train <- bin[bin_sample, 2:3]
bin_train_label <- bin[bin_sample, 1, drop = TRUE]
bin_test <- bin[-bin_sample, 2:3]
bin_test_label <- bin[-bin_sample, 1, drop = TRUE]

trin_sample <- sample(seq_len(nrow(trin)), size = floor(.75 * nrow(trin)))
trin_train <- trin[trin_sample, 2:3]
trin_train_label <- trin[trin_sample, 1, drop = TRUE]
trin_test <- trin[-trin_sample, 2:3]
trin_test_label <- trin[-trin_sample, 1, drop = TRUE]
```

Next we would run the kNN algorithm (kmeans or knn are two examples I found) against the datasets to predict whether the results would indicate a label of 0 or 1. I elected to use knn as the syntax made more sense to me.

```
bin_k3 <- table(knn(train = bin_train, test = bin_test, cl = bin_train_label, k = 3), bin_test_label)
bin_k5 <- table(knn(train = bin_train, test = bin_test, cl = bin_train_label, k = 5), bin_test_label)
bin_k10 <- table(knn(train = bin_train, test = bin_test, cl = bin_train_label, k = 10), bin_test_label)
bin_k15 <- table(knn(train = bin_train, test = bin_test, cl = bin_train_label, k = 15), bin_test_label)
bin_k20 <- table(knn(train = bin_train, test = bin_test, cl = bin_train_label, k = 20), bin_test_label)
```

```
bin_k25 <- table(knn(train = bin_train, test = bin_test, cl = bin_train_label, k = 25), bin_test_label)

trin_k3 <- table(knn(train = trin_train, test = trin_test, cl = trin_train_label, k = 3), trin_test_labe
trin_k5 <- table(knn(train = trin_train, test = trin_test, cl = trin_train_label, k = 5), trin_test_labe
trin_k10 <- table(knn(train = trin_train, test = trin_test, cl = trin_train_label, k = 10), trin_test_la
trin_k15 <- table(knn(train = trin_train, test = trin_test, cl = trin_train_label, k = 15), trin_test_la
trin_k20 <- table(knn(train = trin_train, test = trin_test, cl = trin_train_label, k = 20), trin_test_la
trin_k25 <- table(knn(train = trin_train, test = trin_test, cl = trin_train_label, k = 25), trin_test_la
```

B.2: Analyzing results of kNN

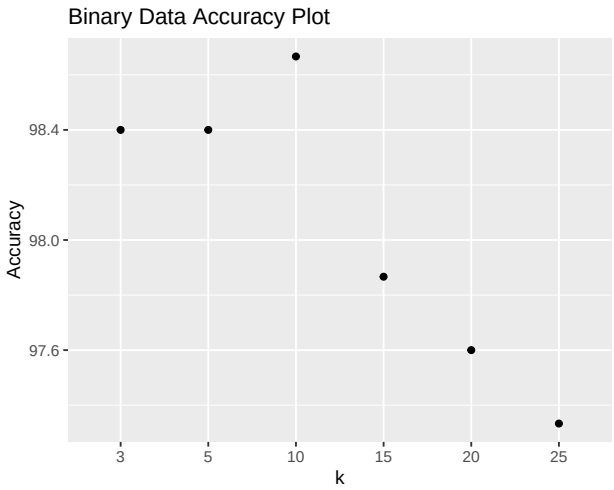
Next we check the accuracy of our models, to see how close our predictions match real world data. I defined a function to make this easier.

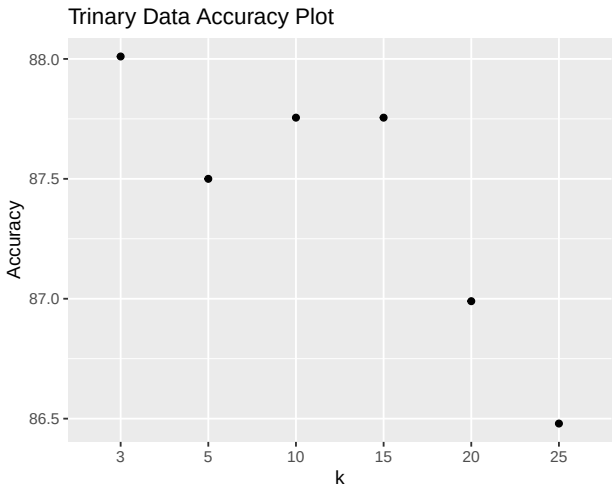
```
acc <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
```

Below are the relative accuracies of kNN for the binary data. These data are plotted to show how changing k changes the accuracy.

```
acc(bin_k3)
## [1] 98.4
acc(bin_k5)
## [1] 98.4
acc(bin_k10)
## [1] 98.66667
acc(bin_k15)
## [1] 97.86667
acc(bin_k20)
## [1] 97.6
acc(bin_k25)
## [1] 97.33333
```

```
acc(trin_k3)
## [1] 88.0102
acc(trin_k5)
## [1] 87.5
acc(trin_k10)
## [1] 87.7551
acc(trin_k15)
## [1] 87.7551
acc(trin_k20)
## [1] 86.9898
acc(trin_k25)
## [1] 86.47959
```





C: Linear Classifiers

These datasets are heavily grouped in random regions leaving me to believe there is no linear classification which might apply.