HW1

Problem 2.1

As someone who is interested in stocks, I would be interested to classify a stock into one of the three categories: 1) Buy 2) Sell 3) Hold.

There are numerous attributes that I could use for this classification – Moving average that indicates the current price trend of that stock, industry that the company operates in, company's revenue growth/decline over the last twelve months, movement of the index that the stock is currently traded in, standard deviation to assess the volatility of the stock and Bollinger Bands.

Another example that comes to my mind is something that grocery stores can use to identify high value customers so that they can target their marketing campaigns at those individuals. Specifically, I would use attributes such as age, gender, marital status, no. of children, salary, amount of money they spent in the last 6 months at their grocery chain, zip code - where they live, distance to the nearest grocery store from their home, no. of grocery stores of their competitors near their home. With these attributes I can identify if someone would be a high value customer and if so, can target specific ads. at them.

Problem 2.2

1) After loading all the packages, I split the data into training and test data sets. This is because if we train and test on the same data set, the model will inherently do well. Also, in real life the test data would be unavailable beforehand and hence this approach is followed. I ran the SVM classifier for varying C values and different kernel types. As shown in the results, the values of C don't affect the accuracy of predictions in a vanilladot classifier. In addition, if I look at the equation of the classifier, I can see that the attributes V5 and V10 are more important than the other ones. The reason is that the coefficients of these attributes are higher when compared to the other ones. There are two possible reasons why the accuracy values don't change for varying C values. 1) The data is already clustered together, and both the classification labels are already separated enough i.e. the margin is already big enough 2) not all the attributes affect the outcome of the classification, as mentioned above only a couple of attributes affect the performance of the classifier. When I look at the accuracies for varying C values in other kernels, I can see that C value does make a difference and the best accuracy is achieved at a C value of 0.1. To summarize, the SVM does really well in classifying the data as an accuracy of 82% is achieved. Also, below is the quation of the classifier: 0.000178 x1 - 0.00022 x2 + 0.00167 x3 + 0.0070 x4 + 0.9911 x5 - 0.0037 x6 + 0.0078 x7 + 0.00086 x8 - 0.0020 x9 + 0.1051 x10 + 0.0819 = 0. The equations of the classifier for each scenario can be found in the results.

2) As mentioned, before I ran the model for three different Kernel types and can see that for these kernel types the value of C does affect the test accuracy. For example, when splinedot kernel is used the best accuracy is achieved at a C value of 0.1 after which the accuracy drops significantly.The same for rbfdot kernel.

3) First, I scaled the data since the ranges for each attribute is significantly different. As shown in the graph that plots the accuracy for different values of n, I can see that the best accuracy is achieved for a n value of 19. As n increases, I can see that the accuracy decreases, which is because the model is not able to learn much from the training data i.e. it is too generalized, hence when it sees the test data the accuracy drops. On the other hand, for low values of n, the test accuracy is also low because the

model learns way too much from the training data and when it sees the test data the accuracy is low because of overfitting. As a rule of thumb, a suitable n value should be equal to the square root of the number of training samples - in our case the training data set size is 490 and the square of root of that is about 22, which is close to our best n value. To summarize, KNN does well on the data with an accuracy of 87%, better than SVM.

```
set.seed(10)
setwd("~/Desktop/Analytics/HW1")

library(kernlab)
```

```
## Warning: package 'kernlab' was built under R version 3.4.4
```

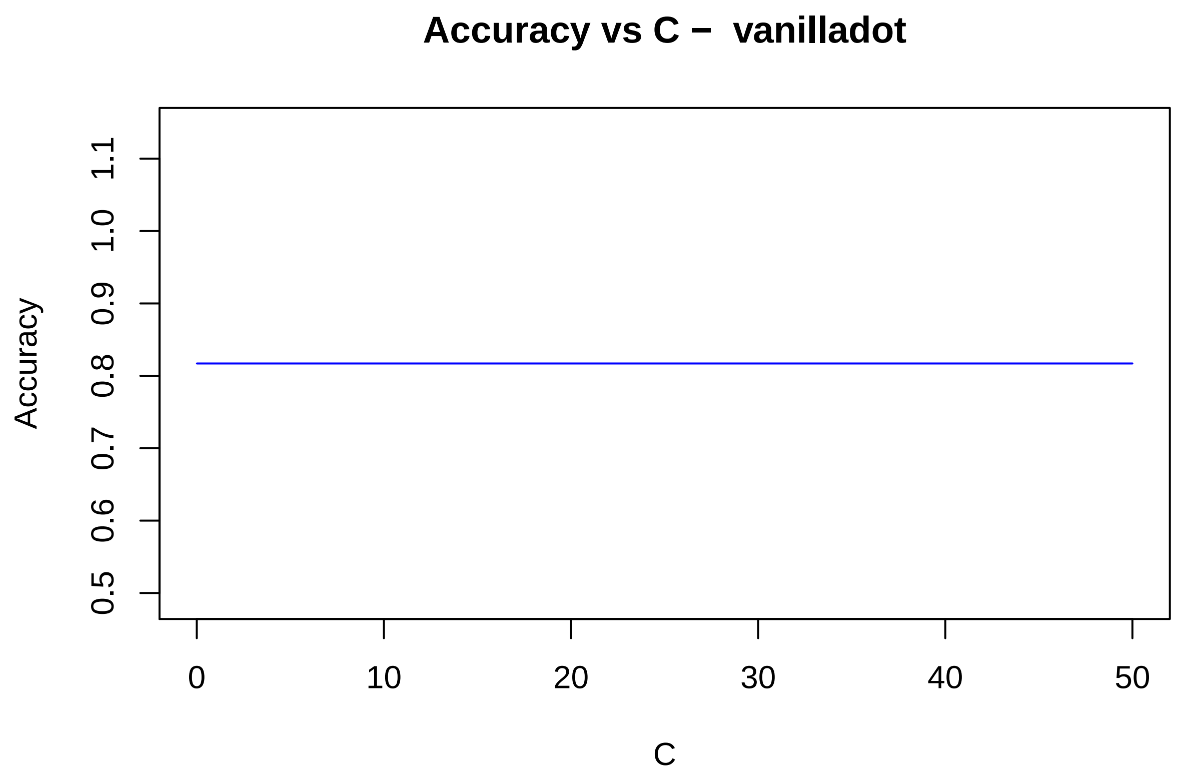
```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4

## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:kernlab':
##
##     alpha
```

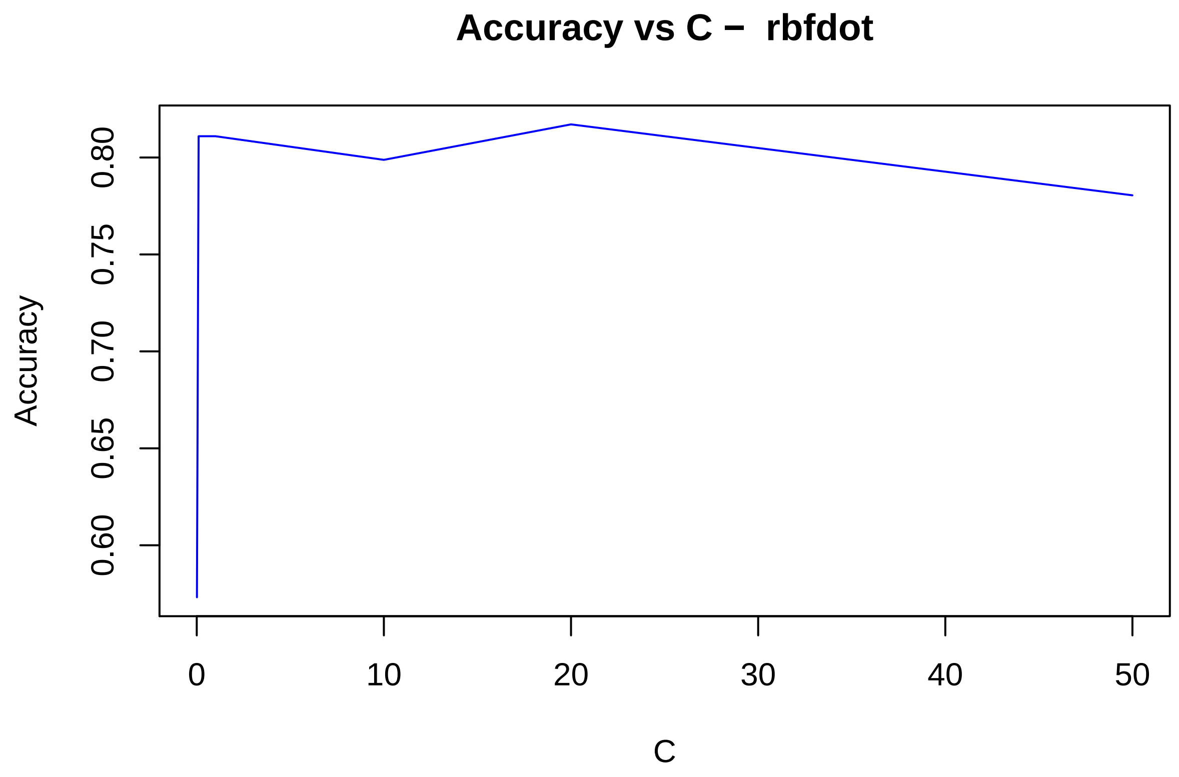
```
credit_data <- read.table("credit_card_data.txt")
samples = sort(sample(nrow(credit_data), nrow(credit_data)*0.75))

train_data = credit_data[samples,]
test_data = credit_data[-samples,]

x = c(0.01,0.1,1.0,10,20,50)
n = c("vanilladot", "rbfdot", "splinedot")
for (j in n){
  alist = c()
  for (i in x) {

    cat("The value of C is ", i, "\n")
    cat("\n")
    model = ksvm(as.matrix(train_data[,1:10]), as.factor(train_data[,11]), type="C-svc", kernel=j, C=i,
    print("Equations - start")
    a = colSums(model@xmatrix[[1]] * model@coef[[1]])
    print(a)
    a0 = -model@b
    print(a0)

    print("Equations - end")
    pred = predict(model, test_data[,1:10])
```

```r
    print(sum(pred == test_data[,11])/nrow(test_data))
    alist = c(alist, sum(pred == test_data[,11])/nrow(test_data))
    cat("Confusion Matrix\n")
    print(table(pred, test_data[,11]))
    cat("\n")
    #print(confusionMatrix(as.factor(pred), as.factor(test_data[,11])))

  }
  plot(x, alist, xlab='C', ylab='Accuracy', main=paste('Accuracy vs C - ', j), type='l', col='blue')
}
```

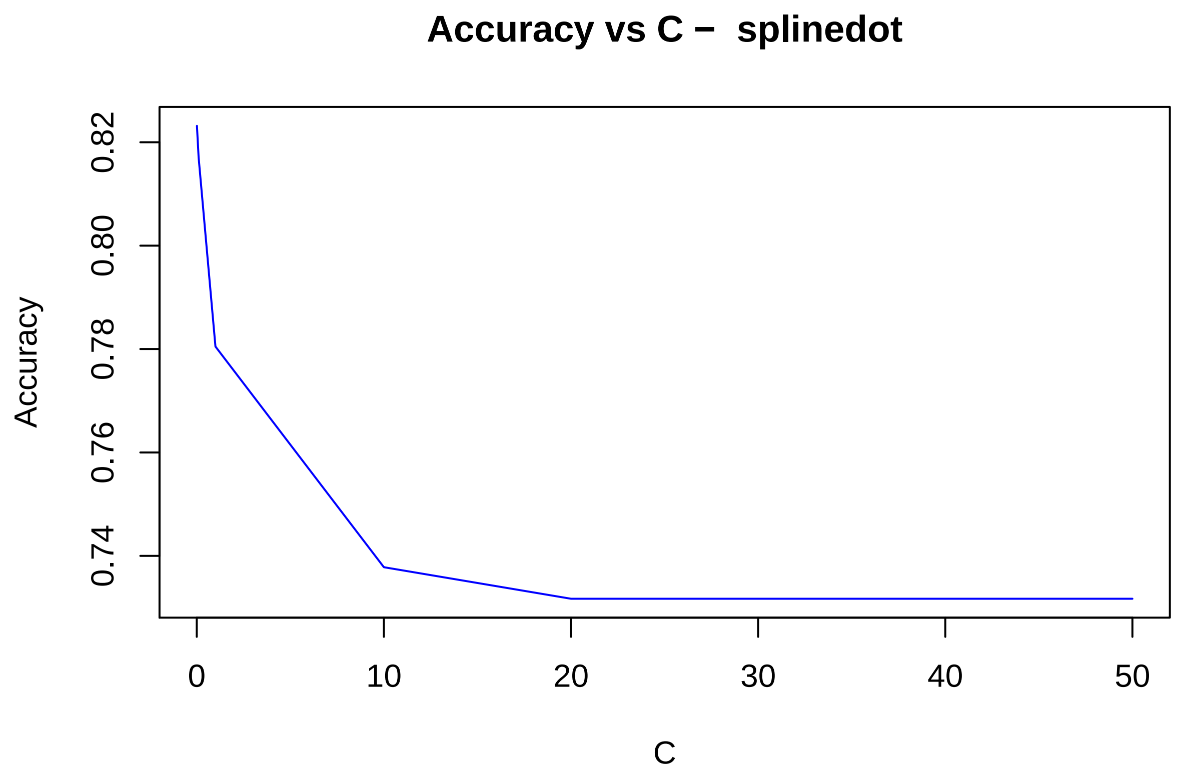
```
## The value of C is  0.01
##
##  Setting default kernel parameters
## [1] "Equations - start"
##            V1            V2            V3            V4            V5
##  0.0001789480 -0.0002225020  0.0016763552  0.0070662766  0.9911101612
##            V6            V7            V8            V9           V10
## -0.0037103615  0.0078519973  0.0008679328 -0.0020892922  0.1051198071
## [1] 0.08191885
## [1] "Equations - end"
## [1] 0.8170732
## Confusion Matrix
##
## pred  0  1
##    0 70  6
##    1 24 64
##
## The value of C is  0.1
##
##  Setting default kernel parameters
## [1] "Equations - start"
##            V1            V2            V3            V4            V5
##  0.0001746787 -0.0016530391 -0.0002251441  0.0011004785  1.0053445194
##            V6            V7            V8            V9           V10
## -0.0009025334  0.0021397070 -0.0003199152 -0.0019082353  0.1194653446
## [1] 0.08393983
## [1] "Equations - end"
## [1] 0.8170732
## Confusion Matrix
##
## pred  0  1
##    0 70  6
##    1 24 64
##
## The value of C is  1
##
##  Setting default kernel parameters
## [1] "Equations - start"
##            V1            V2            V3            V4            V5
##  1.877239e-04 -1.471910e-03 -2.102681e-04  7.705190e-04  1.007888e+00
##            V6            V7            V8            V9           V10
## -1.392269e-03  7.619770e-04  7.169302e-05 -1.678315e-03  1.197184e-01
## [1] 0.08195209
## [1] "Equations - end"
## [1] 0.8170732
## Confusion Matrix
##
## pred  0  1
##    0 70  6
##    1 24 64
##
## The value of C is  10
##
##  Setting default kernel parameters
## [1] "Equations - start"
##            V1            V2            V3            V4            V5
##  3.807114e-05 -1.564005e-03 -3.938874e-04  9.223329e-04  1.007895e+00
##            V6            V7            V8            V9           V10
## -1.249829e-03  8.395660e-04  1.281499e-04 -1.790226e-03  1.197563e-01
## [1] 0.08210904
## [1] "Equations - end"
## [1] 0.8170732
## Confusion Matrix
##
## pred  0  1
##    0 70  6
##    1 24 64
##
## The value of C is  20
##
##  Setting default kernel parameters
## [1] "Equations - start"
##            V1            V2            V3            V4            V5
##  0.0000632361 -0.0014996329 -0.0003422099  0.0009080813  1.0078877850
##            V6            V7            V8            V9           V10
## -0.0012587297  0.0007501059  0.0000907545 -0.0016478413  0.1197161026
## [1] 0.08208607
## [1] "Equations - end"
## [1] 0.8170732
## Confusion Matrix
##
## pred  0  1
##    0 70  6
##    1 24 64
##
## The value of C is  50
##
##  Setting default kernel parameters
## [1] "Equations - start"
##            V1            V2            V3            V4            V5
## -2.551606e-05 -1.381925e-03 -3.930764e-04  8.168232e-04  1.007963e+00
##            V6            V7            V8            V9           V10
## -1.445025e-03  4.397006e-04  5.668497e-05 -1.591906e-03  1.197135e-01
## [1] 0.08213781
## [1] "Equations - end"
## [1] 0.8170732
## Confusion Matrix
##
## pred  0  1
##    0 70  6
##    1 24 64
```



```
## The value of C is  0.01
##
## [1] "Equations - start"
##          V1          V2          V3          V4          V5          V6
##  0.06433365  0.55928102  0.80007891  1.37852862  3.32462364 -1.85716989
##          V7          V8          V9         V10
##  1.70908922 -0.04203112 -0.47262499  0.74756491
## [1] -0.5113298
## [1] "Equations - end"
## [1] 0.5731707
## Confusion Matrix
##
## pred  0  1
##    0 94 70
##    1  0  0
##
## The value of C is  0.1
##
## [1] "Equations - start"
##         V1         V2         V3         V4         V5         V6         V7
##  0.3488602  1.7914520  2.8137692  5.3246608 16.9033486 -3.2694012  5.5198282
##         V8         V9        V10
## -0.6577055 -1.6844585  5.5703865
## [1] 0.4924973
## [1] "Equations - end"
## [1] 0.8109756
## Confusion Matrix
##
## pred  0  1
##    0 69  6
##    1 25 64
##
## The value of C is  1
##
## [1] "Equations - start"
##         V1         V2         V3         V4         V5         V6         V7
## -0.2006794 -1.8618925  5.1216516  6.6100976 28.2627125 -5.9903991 13.0921233
##         V8         V9        V10
## -2.8538249 -7.1855583 19.2839094
## [1] 0.483774
## [1] "Equations - end"
## [1] 0.8109756
## Confusion Matrix
##
## pred  0  1
##    0 69  6
##    1 25 64
##
## The value of C is  10
##
## [1] "Equations - start"
##          V1          V2          V3          V4          V5          V6
##  -0.3567856 -11.9299445   8.2393708  11.0032995  33.0738385  -9.8359662
##          V7          V8          V9         V10
##  14.1005774  -7.0273194 -19.2569881  31.0104983
## [1] 0.4417932
## [1] "Equations - end"
## [1] 0.7987805
## Confusion Matrix
##
## pred  0  1
##    0 71 10
##    1 23 60
##
## The value of C is  20
##
## [1] "Equations - start"
##         V1         V2         V3         V4         V5         V6         V7
##   1.302372 -18.098688   6.346293  14.265367  39.421876  -9.319162  14.918622
##         V8         V9        V10
## -10.061161 -24.583706  34.372319
## [1] 0.470456
## [1] "Equations - end"
## [1] 0.8170732
## Confusion Matrix
##
## pred  0  1
##    0 74 10
##    1 20 60
##
## The value of C is  50
##
## [1] "Equations - start"
##          V1          V2          V3          V4          V5          V6
##   5.3189356 -17.4173133   0.9299892  19.7635112  47.7023766 -15.5200204
##          V7          V8          V9         V10
##  16.1662324 -21.2656676 -33.0828342  34.6169511
## [1] 0.5063747
## [1] "Equations - end"
## [1] 0.7804878
## Confusion Matrix
##
## pred  0  1
##    0 71 13
##    1 23 57
```



```
## The value of C is  0.01
##
##  Setting default kernel parameters
## [1] "Equations - start"
##          V1          V2          V3          V4          V5          V6
## -0.01071833 -0.01139493  0.02098310  0.03337903  0.39940403 -0.08467209
##          V7          V8          V9         V10
##  0.04931244  0.03108432 -0.07886462  0.03146472
## [1] -0.3543269
## [1] "Equations - end"
## [1] 0.8231707
## Confusion Matrix
##
## pred  0  1
##    0 72  7
##    1 22 63
##
## The value of C is  0.1
##
##  Setting default kernel parameters
## [1] "Equations - start"
##         V1         V2         V3         V4         V5         V6         V7
## -0.2147506  0.1165202  0.1261728  0.1199191  0.4222241 -0.2759090  0.1244345
##         V8         V9        V10
##  0.0511223 -0.5051568  0.1055280
## [1] -1.236962
## [1] "Equations - end"
## [1] 0.8170732
## Confusion Matrix
##
## pred  0  1
##    0 73  9
##    1 21 61
##
## The value of C is  1
##
##  Setting default kernel parameters
## [1] "Equations - start"
##          V1          V2          V3          V4          V5          V6
## -0.47061448  3.96772706 -0.62449686  0.15125744 -0.33460750 -0.01332231
##          V7          V8          V9         V10
##  0.15052709  0.49229294 -3.80819674  0.22173936
## [1] -2.829089
## [1] "Equations - end"
## [1] 0.7804878
## Confusion Matrix
##
## pred  0  1
##    0 73 15
##    1 21 55
##
## The value of C is  10
##
##  Setting default kernel parameters
## [1] "Equations - start"
##           V1           V2           V3           V4           V5           V6
##  -0.65406650  36.33067276  -7.50907613   0.89813817  -4.11898523   1.99840549
##           V7           V8           V9          V10
##   0.06582863   7.57291633 -27.93496359   0.76378764
## [1] -8.807861
## [1] "Equations - end"
## [1] 0.7378049
## Confusion Matrix
##
## pred  0  1
##    0 73 22
##    1 21 48
##
## The value of C is  20
##
##  Setting default kernel parameters
## [1] "Equations - start"
##           V1           V2           V3           V4           V5           V6
##  -0.93700627  71.02051972 -14.21849951   2.29560941  -8.09006821   3.63942767
##           V7           V8           V9          V10
##   0.05339299  14.83372765 -53.75686820   1.24195585
## [1] -13.92263
## [1] "Equations - end"
## [1] 0.7317073
## Confusion Matrix
##
## pred  0  1
##    0 73 23
##    1 21 47
##
## The value of C is  50
##
##  Setting default kernel parameters
## [1] "Equations - start"
##            V1            V2            V3            V4            V5
## -1.511463e+00  1.742283e+02 -3.298283e+01  6.053941e+00 -2.071627e+01
##            V6            V7            V8            V9           V10
##  8.665024e+00  9.355199e-03  3.645710e+01 -1.307003e+02  2.750944e+00
## [1] -27.40366
## [1] "Equations - end"
## [1] 0.7317073
## Confusion Matrix
##
## pred  0  1
##    0 74 24
##    1 20 46
```



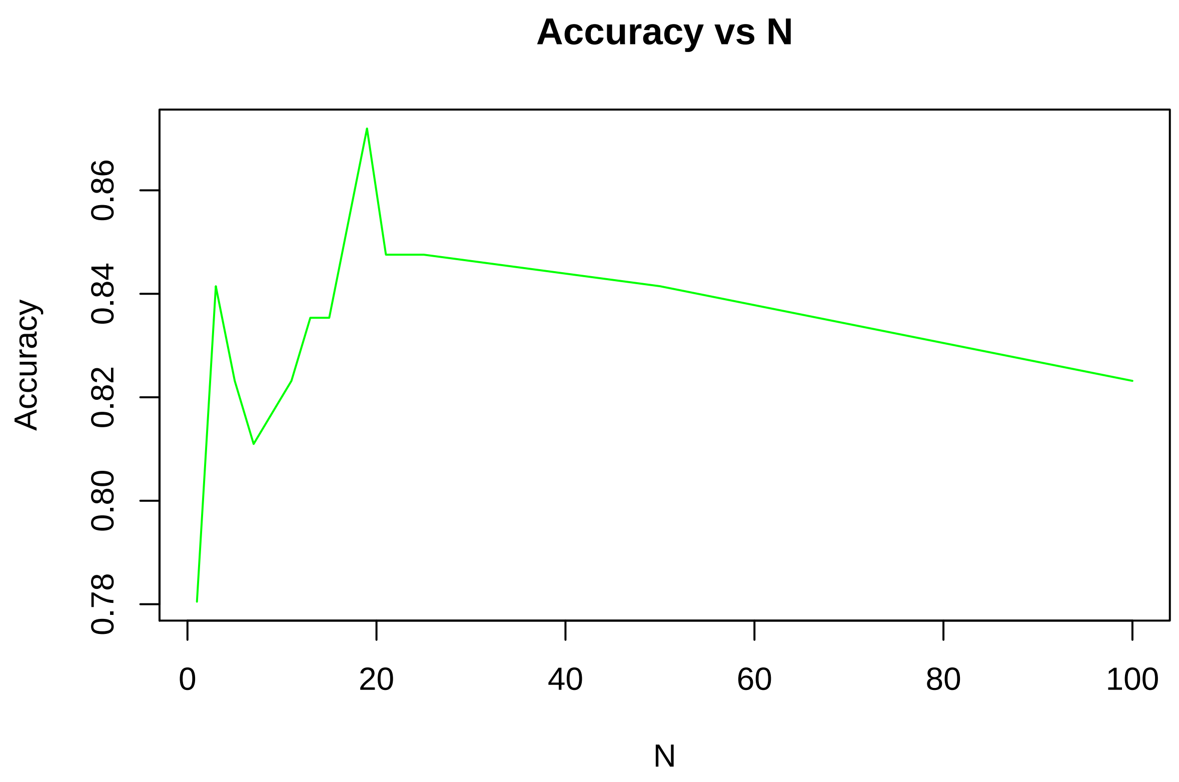
```
print("KNN ---------------------------------------------------")
```

```
## [1] "KNN ---------------------------------------------------"
```

```
library(class)
scaled_train_data = scale(as.matrix(train_data[,1:10], center = TRUE, scale=TRUE))
scaled_test_data = scale(as.matrix(test_data[,1:10], center=TRUE, scale=TRUE))

kn = c(1,3,5,7,9,11,13,15,17,19,21,23,25,50,100)

a = c()
for (i in kn){
  model = knn(scaled_train_data, scaled_test_data, as.factor(train_data[,11]), k=i)
  cat("Confusion Matrix for k value", i)
  tab = table(model,test_data[,11])
  print(tab)

  accuracy = sum(diag(tab))*1.0/(nrow(scaled_test_data))
  a = c(a, accuracy)
  cat("Knn Accuracy for k value of",i, "is ", accuracy, "\n")

  }
```

```
## Confusion Matrix for k value 1
## model   0   1
##     0 75 17
##     1 19 53
## Knn Accuracy for k value of 1 is  0.7804878
## Confusion Matrix for k value 3
## model  0  1
##     0 75  7
##     1 19 63
## Knn Accuracy for k value of 3 is  0.8414634
## Confusion Matrix for k value 5
## model  0  1
##     0 74  9
##     1 20 61
## Knn Accuracy for k value of 5 is  0.8231707
## Confusion Matrix for k value 7
## model  0  1
##     0 73 10
##     1 21 60
## Knn Accuracy for k value of 7 is  0.8109756
## Confusion Matrix for k value 9
## model  0  1
##     0 75 11
##     1 19 59
## Knn Accuracy for k value of 9 is  0.8170732
## Confusion Matrix for k value 11
## model  0  1
##     0 75 10
##     1 19 60
## Knn Accuracy for k value of 11 is  0.8231707
## Confusion Matrix for k value 13
## model  0  1
##     0 76  9
##     1 18 61
## Knn Accuracy for k value of 13 is  0.8353659
## Confusion Matrix for k value 15
## model  0  1
##     0 75  8
##     1 19 62
## Knn Accuracy for k value of 15 is  0.8353659
## Confusion Matrix for k value 17
## model  0  1
##     0 77  7
##     1 17 63
## Knn Accuracy for k value of 17 is  0.8536585
## Confusion Matrix for k value 19
## model  0  1
##     0 80  7
##     1 14 63
## Knn Accuracy for k value of 19 is  0.8719512
## Confusion Matrix for k value 21
## model  0  1
##     0 79 10
##     1 15 60
## Knn Accuracy for k value of 21 is  0.847561
## Confusion Matrix for k value 23
## model  0  1
##     0 79 10
##     1 15 60
## Knn Accuracy for k value of 23 is  0.847561
## Confusion Matrix for k value 25
## model  0  1
##     0 81 12
##     1 13 58
## Knn Accuracy for k value of 25 is  0.847561
## Confusion Matrix for k value 50
## model  0  1
##     0 82 14
##     1 12 56
## Knn Accuracy for k value of 50 is  0.8414634
## Confusion Matrix for k value 100
## model  0  1
##     0 83 18
##     1 11 52
## Knn Accuracy for k value of 100 is  0.8231707
```

```r
plot(kn, a, xlab='N', ylab='Accuracy', main='Accuracy vs N', type='l', col='green')
```


Accuracy vs N
Accuracy
N