HW2

```
set.seed(10)
```

```
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

## Warning: package 'lattice' was built under R version 3.4.4

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

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

```
library(class)
```

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

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

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

3.1 Part A - Cross Validation - KNN

First, I performed cross validation to find out the best k value based on the highest validation accuracy. The best accuracy obtained after cross validation is 85.34%. I then used the same k value, fit a model and used the model on the test data. The test accuracy dropped slightly to 82.37%. The reason for this drop is that when we cross validated the model, the better performance could have been contributed by chance and not because of the model being better. Thus, when we tested the model on the test data, we don't see the same performance.

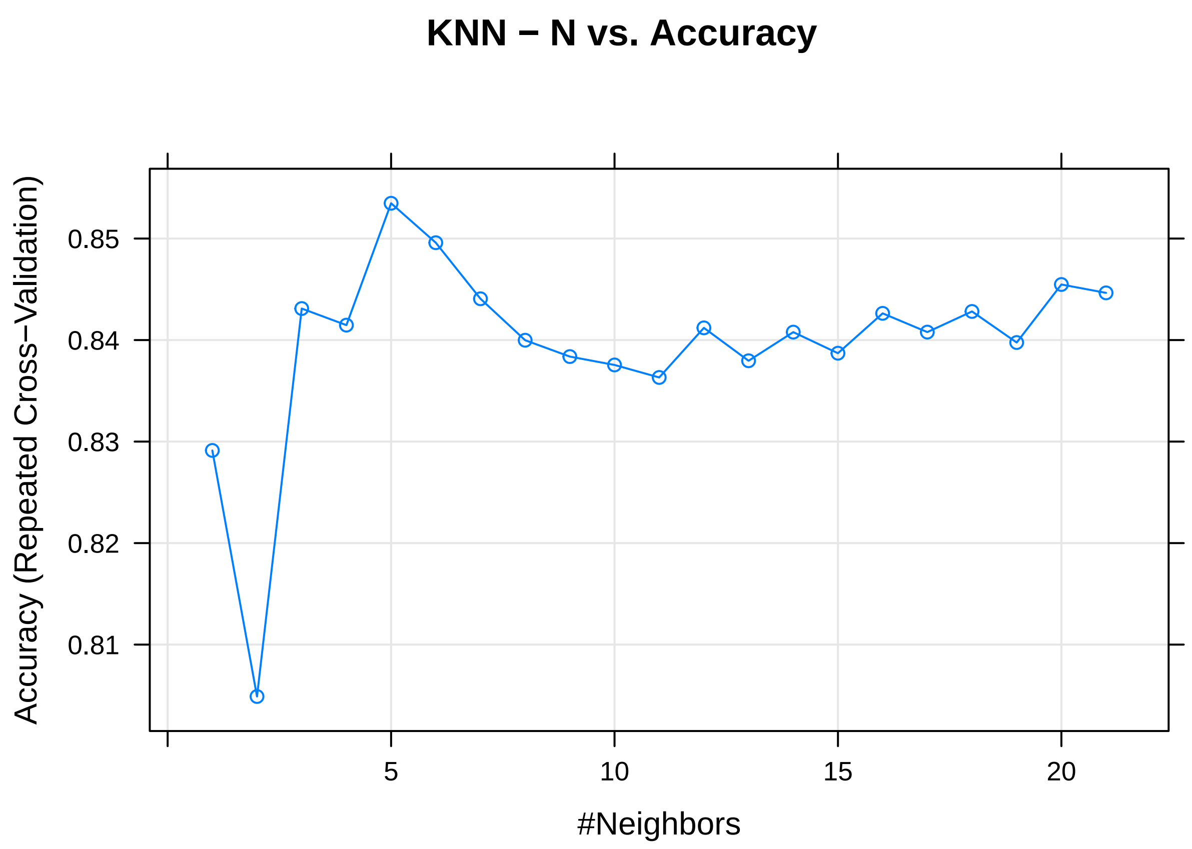
```r
scaled_train_data = scale(as.matrix(train_data[,1:10], center = TRUE, scale=TRUE))
scaled_test_data = scale(as.matrix(test_data[,1:10], center=TRUE, scale=TRUE))

ctrl = trainControl(method="repeatedcv", repeats = 10)

knnmodel = train(scaled_train_data,as.factor(train_data[,11]),method="knn",tuneGrid=expand.grid(k=1:21)
knnmodel
```

```
## k-Nearest Neighbors
##
## 490 samples
##  10 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 441, 441, 441, 441, 442, 440, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.8291269  0.6553604
##    2  0.8048895  0.6059555
##    3  0.8431039  0.6840454
##    4  0.8414702  0.6809192
##    5  0.8534764  0.7055339
##    6  0.8495895  0.6977927
##    7  0.8440699  0.6870846
##    8  0.8399833  0.6786272
##    9  0.8383747  0.6754919
##   10  0.8375498  0.6737643
##   11  0.8363167  0.6713412
##   12  0.8412031  0.6812201
##   13  0.8379626  0.6750442
##   14  0.8407825  0.6802577
##   15  0.8387037  0.6761251
##   16  0.8426311  0.6839426
##   17  0.8407862  0.6800952
##   18  0.8428194  0.6837693
##   19  0.8397575  0.6771221
##   20  0.8454730  0.6884473
##   21  0.8446483  0.6865866
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```r
plot(knnmodel, main="KNN - N vs. Accuracy\n")
```



```
model = knn(scaled_train_data, scaled_test_data, as.factor(train_data[,11]), k=5)
cat("Confusion Matrix")
```

```
## Confusion Matrix
```

```
tab = table(model,test_data[,11])
print(tab)
```

```
##
## model  0  1
##     0 74  9
##     1 20 61
```

```
accuracy = sum(diag(tab))*1.0/(nrow(scaled_test_data))
cat("Test Accuracy", accuracy)
```

```
## Test Accuracy 0.8231707
```

3.1 Part A - Cross Validation - SVM

Next, I performed cross validation using a SVM classifier. The accuracy after cross validation i.e on the validation data set is 87.7% and the same on the test data us 81.7%. The reason for this behaviour is the same as what was explained above for KNN.

```
model = ksvm(as.matrix(train_data[,1:10]), as.factor(train_data[,11]), type="C-svc", kernel="vanilladot
```

```
##  Setting default kernel parameters
```

```
cat("Validation accuracy", 1.0 - model@cross, "\n")
```

```
## Validation accuracy 0.877551
```

```
pred = predict(model, test_data[,1:10])

cat("Test accuracy", sum(pred == test_data[,11])/nrow(test_data), "\n")
```

```
## Test accuracy 0.8170732
```

3.1 - Part B - SVM

In this part, I split the data into training, validation and test data sets and fit the SVM model to determine the validation and test accuracies. I ran the SVM for different values of C and cross validated to obtain the best validation accuracy. I then used the same C value to test the model on the test data. I split the dataset as follows: 60% training, 20% validation and 20% test. As shown below the best C values was 0.05, validation accuracy was 90% and test accuracy was 83.2%.

```
samples = sort(sample(nrow(credit_data), nrow(credit_data)*0.60))


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

samples = sort(sample(nrow(test_data), nrow(test_data)*0.50))
validation_data = test_data[samples,]
test2_data = test_data[-samples,]
x = c(0.0001, 0.001,0.01,0.05,0.1,1,10)
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="vanilla

    pred = predict(model, validation_data[,1:10])


    cat(" Validation accuracy", sum(pred == validation_data[,11])/nrow(validation_data), "\n")
    alist = c(alist, sum(pred == validation_data[,11])/nrow(validation_data))
  }
```

```
##  The value of C is  1e-04
##
##  Setting default kernel parameters
##  Validation accuracy 0.4732824
##  The value of C is  0.001
##
##  Setting default kernel parameters
##  Validation accuracy 0.6335878
##  The value of C is  0.01
##
##  Setting default kernel parameters
##  Validation accuracy 0.9007634
##  The value of C is  0.05
##
##  Setting default kernel parameters
##  Validation accuracy 0.9007634
##  The value of C is  0.1
##
##  Setting default kernel parameters
##  Validation accuracy 0.9007634
##  The value of C is  1
##
##  Setting default kernel parameters
##  Validation accuracy 0.9007634
##  The value of C is  10
##
##  Setting default kernel parameters
##  Validation accuracy 0.9007634
```

```
model = ksvm(as.matrix(train_data[,1:10]), as.factor(train_data[,11]), type="C-svc", kernel="vanilladot
```

```
##  Setting default kernel parameters
```

```
pred = predict(model, test2_data[,1:10])
cat(" Test accuracy", sum(pred == test2_data[,11])/nrow(test2_data))
```

```
##  Test accuracy 0.8320611
```

4.1 - K-Means Example

A clustering algorithm can be used to group similar customers together, so that retail stores can target specific ads depending upon the customer type and their preferences.

The attributes that can be useful are as follows: 1) Age 2) Gender 3) Occupation 4) Income 5) Marital status 6) No. of Children 7) Amount of money spent in that particular retail store type in the last twelve months (i.e. amount spent in Clothing if it is a clothing retailer, amount spent in groceries if the business type is a grocer).

Clustering can also be used in the healthcare industry to group patients with certain types of diseases or patients that exhibit similar symptoms. For example, diabetic, heart disease, thyroid disease patients could each form a cluster. This will enable the medical professional to target the appropriate treatment. There are many attributes that I can think of here: 1) Age 2) Sex 3) Blood Pressure (Lower and Higher could each be an attribute) 4) Heart rate 5) A1C level 6) Fasting blood sugar level 7) Thyroid hormone level 8) Symptom 1 - chest pain 9) Symptom 2 - fatigue 10) Symptom 3 - anxiety.

4.2 - K - Means

1) As shown below in the elbow curve - the best value of k is 3. The elbow graph plots the total sum of squared distance between each point to the corresponding cluster center. At a k value of 3 there is a significant reduction in this value when compared to values 1 and 2. This is the reason why the best k is chosen as 3. For k values greater than 3 there is indeed a reduction in this metric, however that reduction is marginal when compared to the initial reduction.

2) At this value of k=3, Setosa is predicted correctly with all 50 data points in one cluster. Most of Versicolor data is predicted in one cluster with 2 instances predicted in a cluster that's shared with Virginica. To summarize, of the three different labels, Setosa is predicted accurately (100% accuracy), the next is Versicolor (96% accuracy) and the last one is Virginica (72% accuracy).

3) Next, I ran the K-Means algorithm for the combination of a pair of parameters i.e Sepal Length and Sepal Width as one scenario, Petal Length and Petal Width as another scenario and so on to determine the pair that clusters the data the best. As shown below, Petal Length and Petal Width look like two most important attributes that can help in accurately classifying the data.

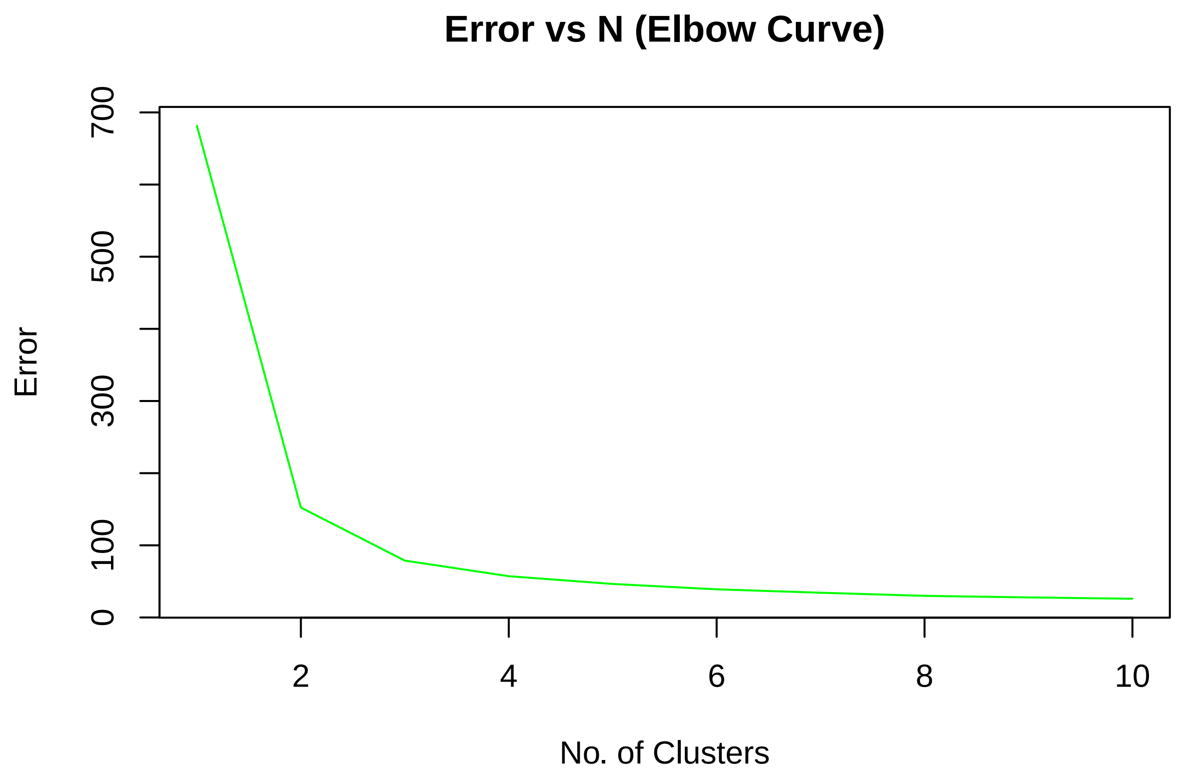
4) I also plotted the clusters against the two Principal Components that explain the most variability in the data. This is easier to visualize as the 4 dimensions are reduced to two.

```
library(cluster)
```

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

```
iris = read.table("iris.txt")
error = c()
km = c()
for (i in 1:10){
  kmod = kmeans(iris[,1:4], i, nstart=25)
  #print(table(kmod$cluster,iris[,5]))
  error = c(error, kmod$tot.withinss)
  km = c(km, i)
  #clusplot(iris[,1:4], kmod$cluster, color=TRUE, shade=TRUE, lines=0, main="Cluster Plot")
}

plot(km, error, xlab='No. of Clusters', ylab='Error', main='Error vs N (Elbow Curve)', type='l', col='g
```



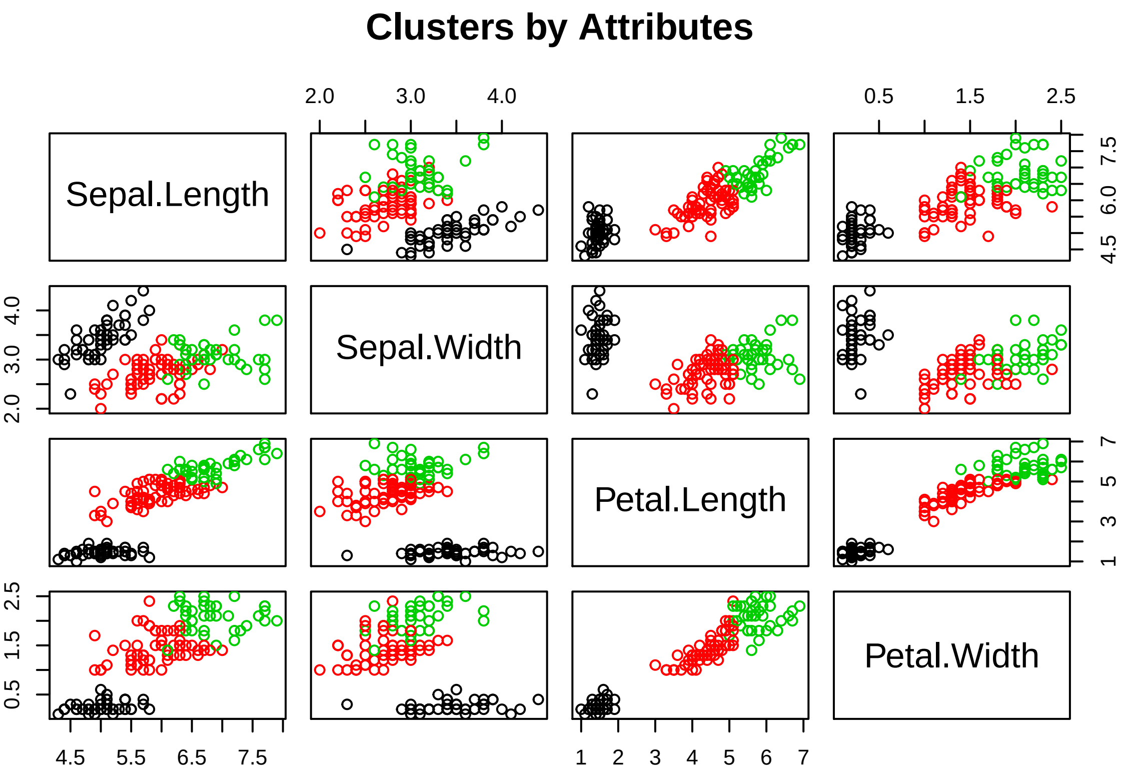
```r
kmod = kmeans(iris[,1:4], 3, nstart=25)
print("Confusion Matrix")
```

```
## [1] "Confusion Matrix"
```

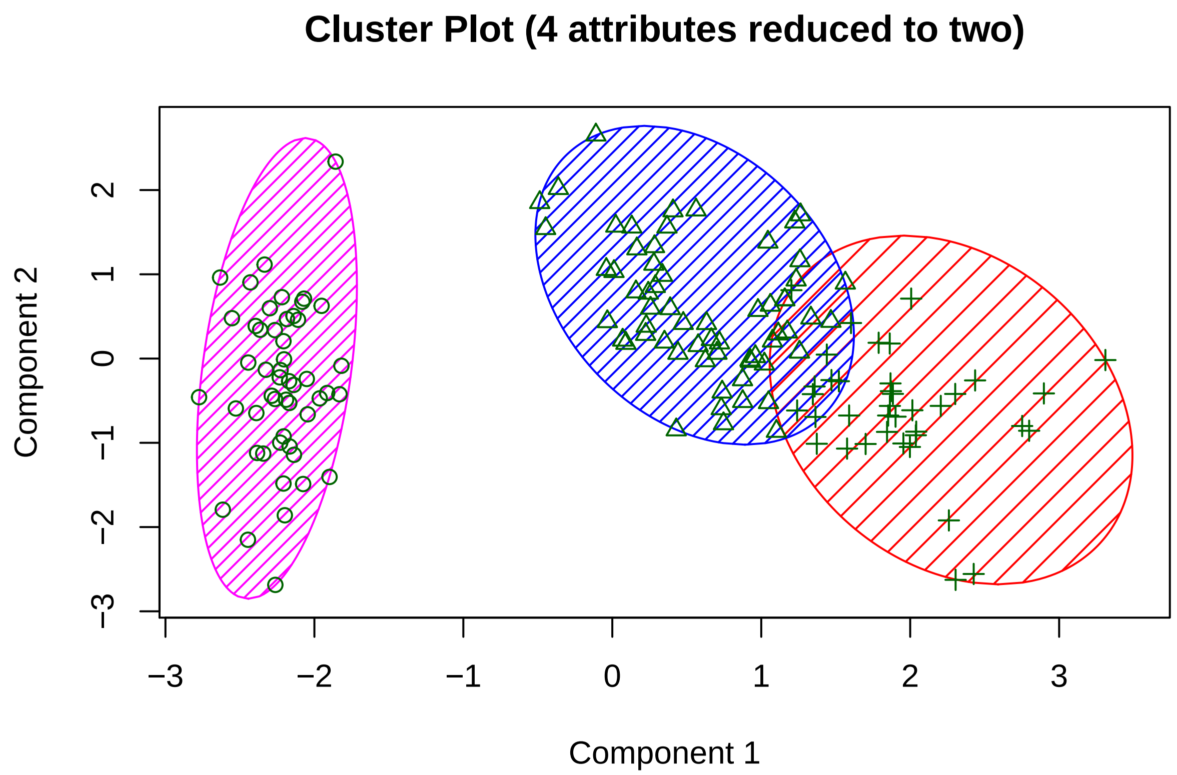
```r
print(table(kmod$cluster,iris[,5]))
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         48        14
##   3      0          2        36
```

```r
plot(iris[,1:4], col=kmod$cluster, main="Clusters by Attributes")
```



```
clusplot(iris[,1:4], kmod$cluster, color=TRUE, shade=TRUE, lines=0, main="Cluster Plot (4 attributes re
```



These two components explain 95.81 % of the point variability.

```
cat("Determine the best factors/attributes")
```

```
## Determine the best factors/attributes
```

```
cat("Sepal Length and Sepal Width")
```

```
## Sepal Length and Sepal Width
```

```
kmod = kmeans(iris[,1:2], 3, nstart=25)

print(table(kmod$cluster,iris[,5]))
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         12        35
##   3      0         38        15
```

```
cat("Petal Length and Petal Width")
```

```
## Petal Length and Petal Width
```

```r
kmod = kmeans(iris[,3:4], 3, nstart=25)
print(table(kmod$cluster,iris[,5]))
```

```
##
##     setosa versicolor virginica
##   1      0         48         4
##   2     50          0         0
##   3      0          2        46
```

```r
cat("Sepal Length and Petal Length")
```

```
## Sepal Length and Petal Length
```

```r
kmod = kmeans(iris[,c(1,3)], 3, nstart=25)
print(table(kmod$cluster,iris[,5]))
```

```
##
##     setosa versicolor virginica
##   1     50          1         0
##   2      0          4        37
##   3      0         45        13
```

```r
cat("Sepal Width and Petal Width")
```

```
## Sepal Width and Petal Width
```

```r
kmod = kmeans(iris[,c(2,4)], 3, nstart=25)

print(table(kmod$cluster,iris[,5]))
```

```
##
##     setosa versicolor virginica
##   1     49          0         0
##   2      1         46         6
##   3      0          4        44
```

```r
cat("Sepal Length and Petal Width")
```

```
## Sepal Length and Petal Width
```

```r
kmod = kmeans(iris[,c(1,4)], 3, nstart=25)
print(table(kmod$cluster,iris[,5]))
```

```
##
##     setosa versicolor virginica
##   1      0          9        35
##   2      0         37        15
##   3     50          4         0
```

```r
cat("Sepal Width and Petal Length")
```

```
## Sepal Width and Petal Length
```

```r
kmod = kmeans(iris[,c(2,3)], 3, nstart=25)
print(table(kmod$cluster,iris[,5]))
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0          2        41
##   3      0         48         9
```