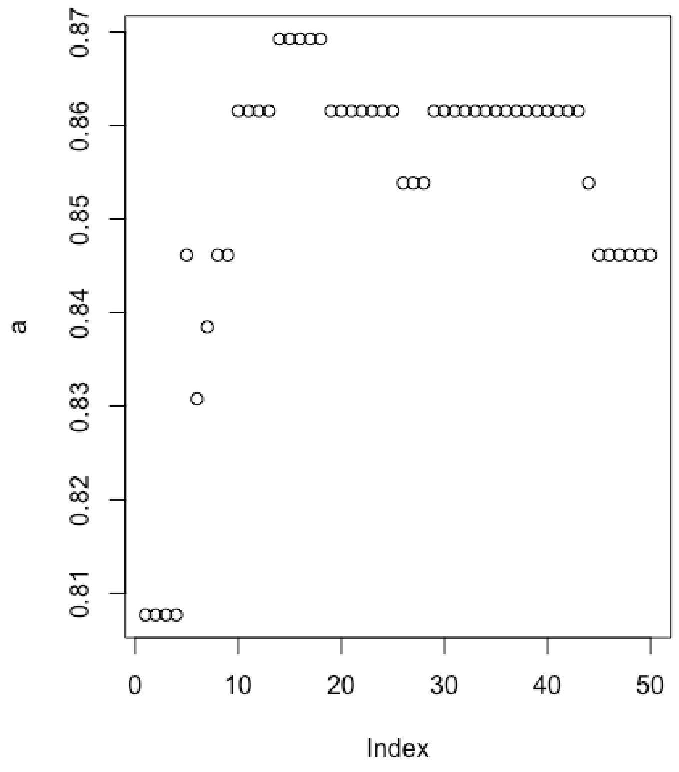
Homework 2

3.1
Using the same data set (credit_card_data.txt or credit_card_data-headers.txt) as in Question 2.2, use the ksvm or kknn function to find a good classifier:
(a) using cross-validation (do this for the k-nearest-neighbors model; SVM is optional); and
(b) splitting the data into training, validation, and test data sets (pick either KNN or SVM; the other is optional).

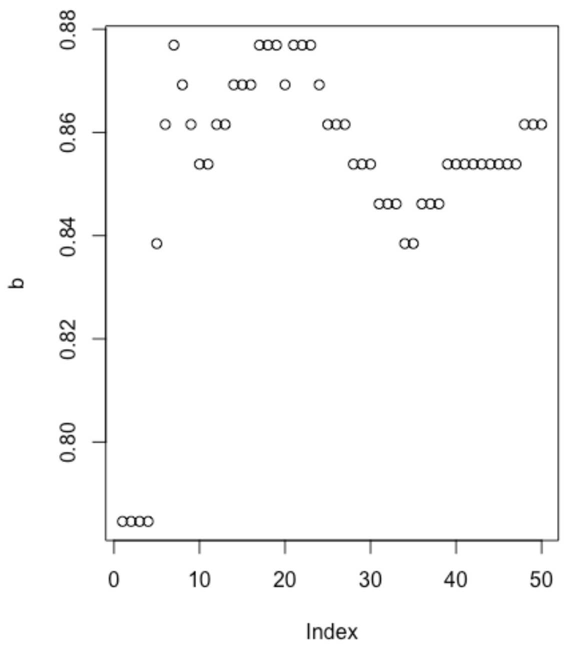
a) #METHOD 1: MANUAL 4 FOLD CROSS VALIDATION

```
library(kernlab)
library(kknn)
set.seed(42)

#Import the data
data<- read.delim("credit_card_data-headers.txt",stringsAsFactors = FALSE,header = TRUE)

#sampling the data
index<- sample(1:nrow(data),520)
data_cv<- data[index,]   # 520 out of 654 data points are set to be cross-validation data set
data_test<- data[-index,] #While remaining 134 data points to be testing data set.

#Split the cv data set into 4 folds (20% for each fold (4 folds-total 80%), 20% for test data set)
data_cv1<-data_cv[1:130,]
data_cv2<-data_cv[131:260,]
data_cv3<-data_cv[261:390,]
data_cv4<-data_cv[391:520,]
data123<- rbind(data_cv1,data_cv2,data_cv3)
data124<- rbind(data_cv1,data_cv2,data_cv4)
data134<- rbind(data_cv1,data_cv3,data_cv4)
data234<- rbind(data_cv2,data_cv3,data_cv4)

#Build a for loop to perform cross validation for different combination and 1:50 k values

# data_cv1 & data_cv2 & data_cv3 as training set, while data_cv4 as validation set
a<-c()
for ( k in 1:50){
m1<- kknn(R1~.,data123,test=data_cv4,k=k,scale=TRUE)
a[k]= sum((round(predict(m1))) == data_cv4$R1)/nrow(data_cv4)
}
plot(a)
```



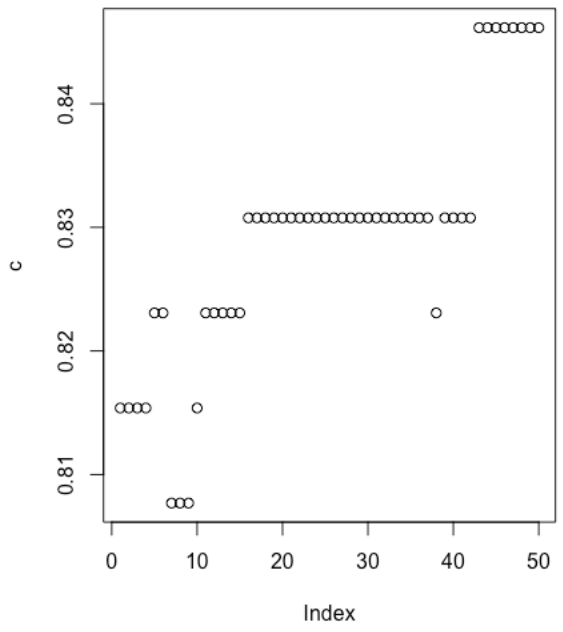
#data_cv1 & data_cv2 & data_cv4 as training set, while data_cv3 as validation set

```
b<-c()
for (k in 1:50){
m2<- kknn(R1~.,data124,data_cv3,k=k,scale=TRUE)
b[k]= sum((round(predict(m2))) == data_cv3$R1)/nrow(data_cv3)
}

plot(b)
```



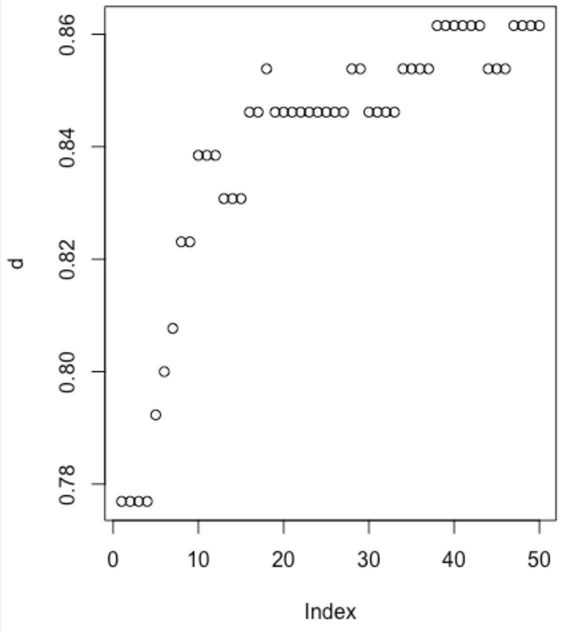
#data_cv1 & data_cv3 & data_cv4 as training set, while data_cv2 as validation set

```
c<-c()
for (k in 1:50){
m3<- kknn(R1~.,data134,data_cv2,k=k,scale=TRUE)
c[k] =sum((round(predict(m3))) == data_cv2$R1)/nrow(data_cv2)
}

plot(c)
```



#data_cv2 & data_cv3 & data_cv4 as training set, while data_cv1 as validation set

```
d<-c()
for (k in 1:50) {
m4<-kknn(R1~.,data234,data_cv1,k=k,scale=TRUE)
d[k] =sum((round(predict(m4))) == data_cv1$R1)/nrow(data_cv1)
}

plot(d)
```



#comparing the mean for different models

```
> mean(a)
[1] 0.8535385
> mean(b)
[1] 0.8530769
> mean(c)
[1] 0.8290769
> mean(d)
[1] 0.8401538
```

#*The average value for combination a is the highest among these 4 folds. So we choose to go for this combination and pick the parameter (k) with the highest accuracy*

```
> which( a == max(a))
[1] 14 15 16 17 18.  # when k = 14 / 15 / 16 /17 /18, the model produces the highest accuracy

> max(a)
[1] 0.8692308.   #highest is the 0.8692308
```

# perform the final testing with entire cross validation data set and optimal k

```
> data_cvcombine<- rbind(data_cv1,data_cv2,data_cv3,data_cv4)
> m_finaltest<- kknn(R1~.,data_cvcombine,data_test,k=14,scale = TRUE)
> accuracy_finaltest<- sum(round(predict(m_finaltest)) == data_test$R1) / nrow(data_test)
> accuracy_finaltest
[1] 0.858209
```

So the testing accuracy is 85.8209%, and the optimal k is 14

a) **Method 2: Using train.kknn**

# create a loop for train.kknn

```
result<- c()
> for (i in 1:25){
+ model <- train.kknn(R1~.,data,kmax=25,scale=TRUE)
+ result[i] <- sum((round(fitted(model)[[i]][1:nrow(data)])) == data$R1)/ nrow(data)
+ }
```

# show the result of the models with 25 different k value

```
> result
 [1] 0.8149847 0.8149847 0.8149847 0.8149847 0.8516820 0.8455657 0.8470948 0.8486239
0.8470948 0.8516820 0.8516820 0.8532110
[13] 0.8516820 0.8516820 0.8532110 0.8532110 0.8532110 0.8516820 0.8501529 0.8501529
0.8486239 0.8470948 0.8440367 0.8455657
[25] 0.8455657

> max(result)
[1] 0.853211
> result_optimal<- which(result == max(result))
> result_optimal
[1] 12 15 16 17
```

So the testing accuracy is 85.3211 % when using train.kknn , and the optimal k is 12 / 15 /16/ 17.

When using train.kknn for cross validation, we assume K fold is number of data points(n-1). So in this case we are executing 653 fold cross validation.

a) **Method 3: Using using cv.kknn**

# create a for loop with cv.kknn models and looping over 25 different k values

```
result_cv<- c()


> for (i in 1:25){
+ m_cvkknn<- cv.kknn(R1~.,data,kcv=10,k=i,scale=TRUE)  # Using 10 folds cross validation.
+ result_cv[i]<- sum((round(m_cvkknn[[1]][,2]))==data$R1)/nrow(data)
+ }

> max(result_cv) # Get the maximum result : accuracy percent.
[1] 0.8562691

> cvkknn_optimal<- which(result_cv == max(result_cv))
> cvkknn_optimal. # Get the optimal k : 10 .
[1] 10
```

the testing accuracy is 85.6291 % when using cv.kknn , and the optimal k is 10.

B） splitting the data into training, validation, and test data sets (pick either KNN or SVM; the other is optional).

```
# data preparation- split the data in 3 sets, d_train,(60%)  d_valid(20%)  and d_test.(20%)

> d_train<- data [1:392,]
> d_valid<- data [393:523,]
> d_test<- data[524:654,]

# train model with d_train and validate with d_valid to get the optimal performance
parameter ( k in 1:50)
f<-c()
for (i in 1:50){
+ reg_val<- kknn(R1~.,d_train,d_valid,k=i,scale=TRUE)
+ f[i]<- sum(round(fitted(reg_val)) == d_valid$R1)/nrow(d_valid)
+ }

# evaluate 50 different models and choose the one with highest accuracy percent
> f
 [1] 0.7938931 0.7938931 0.7938931 0.7938931 0.7938931 0.8015267 0.8015267 0.8015267
0.8015267 0.8015267 0.8015267 0.8015267
```

```
[13] 0.8091603 0.8091603 0.8167939 0.8167939 0.8167939 0.8167939 0.8244275 0.8396947
0.8396947 0.8396947 0.8396947 0.8396947
[25] 0.8473282 0.8473282 0.8473282 0.8549618 0.8473282 0.8473282 0.8473282 0.8473282
0.8473282 0.8549618 0.8549618 0.8549618
[37] 0.8549618 0.8549618 0.8473282 0.8473282 0.8473282 0.8473282 0.8473282 0.8473282
0.8473282 0.8473282 0.8473282 0.8473282
[49] 0.8473282 0.8473282
```

# the highest accuracy rate is 0.8549618

```
> max(f)
[1] 0.8549618
```

# the optimal k is chosen among these values with the same accuracy rate (28 34 35 36 37 38)

```
> optimal_f<- which(f == max(f))
> optimal_f
[1] 28 34 35 36 37 38
```

# Implement the best performance result and report back to testing data set. Loop over each parameter and see which one produces the highest accuracy rate.

```
> op_f <- c(28,34,35,36,37,38)
> for (i in 1:length(op_f)){
+ reg_test<- kknn(R1~.,d_train,d_test,k=op_f[i],scale=TRUE)
+ optimal_f[i]<-sum(round(fitted(reg_test)) == d_test$R1)/nrow(d_test)
+ }
> optimal_f
[1] 0.8244275 0.8167939 0.8167939 0.8167939 0.8167939 0.8167939
```

The optimal K is 28 which produces the highest accuracy rate on testing date set: 0.8244275

Question 4.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

When I was taking online courses from edx or coursera, the platform asked me to fill in a lot of personal information then it will generate a list of courses that you may be interested in. The recommendation courses(Could be response in clustering model) designed for each students could be calculated by clustering model based on various factors.

Factors:
1) Students age.
2) Student's current job
3)Gender
4) Courses already enrolled.
5) Preferred Career path

Question 4.2

#Import data and set seed in order to make the result reproducible.

```
>library(kernlab)
>library(ggplot2)
>set.seed(42)
>data(iris)
```

#mix up the raw data to avoid bias

```
>gp<- runif(nrow(iris))
> iris<- iris[order(gp),]
> head(iris)
   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
112          6.4         2.7          5.3         1.9  virginica
76           6.6         3.0          4.4         1.4 versicolor
41           5.0         3.5          1.3         0.3     setosa
140          6.9         3.1          5.4         2.1  virginica
130          7.2         3.0          5.8         1.6  virginica
19           5.7         3.8          1.7         0.3     setosa
```

#normalize the data (Scaling)
#build a function called "normalize" which aims to scale the data into 0:1

```
> normalize<- function(x) {
+ return((x- min(x))/(max(x)-min(x)))
+ }
```

```
> iris_n <- as.data.frame(lapply(iris[,c(1,2,3,4)],normalize))
```

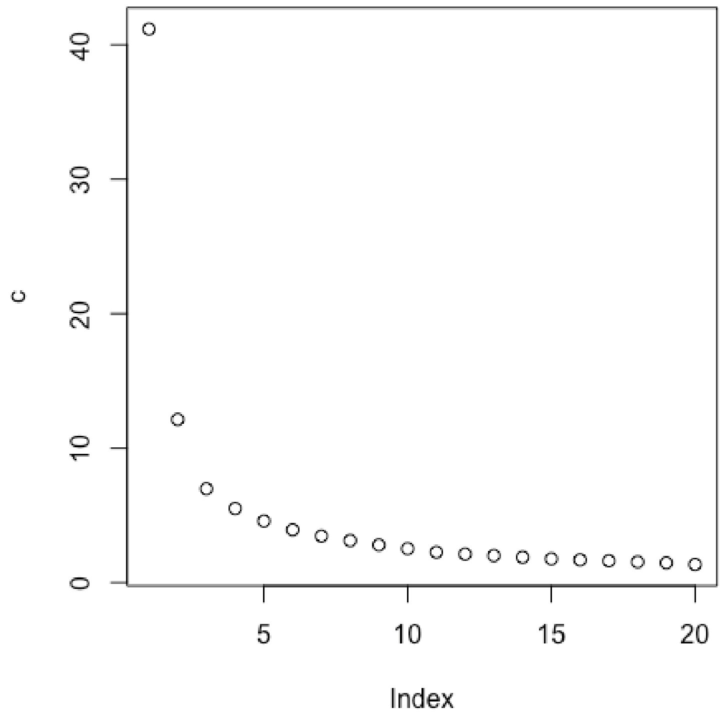
#Build a for loop to try out different numbers of centers using Kmeans model

```
c<-c()
> for (i in 1:20){
+ cluster_finder<- kmeans(iris_n[,1:4],centers=i,nstart=25)
+ c[i]<-cluster_finder$tot.withinss
+ }
```

# Investigate the relationship between center number and tot.withinss

```
> c
 [1] 41.166110 12.127791  6.982216  5.516933  4.580323  3.926713  3.467980  3.130580
2.801430  2.528057  2.266034  2.125728  2.013635  1.890940  1.770455  1.695634  1.624415
1.542890
[19]  1.470108  1.351771
> plot(c)  # the elbow curve took place when centers=3 (diminishing marginal benefit begins)
```

#Optimal Centers = 3



# Specify different combinations of predictors to train kmeans function.

```
#     Sepal.Length(1) /. Sepal.Width(2)/  Petal.Length(3)/  Petal.Width (4)
```

#In total, there are 11 combinations to train the model

Sepal.Length(1) +. Sepal.Width(2)
Sepal.Length(1)+ Petal.Length(3)
Sepal.Length(1) +Petal.Width (4)
Sepal.Width(2)+ Petal.Length(3)
Sepal.Width(2)+ Petal.Width (4)
Petal.Length(3) + Petal.Width (4)

Sepal.Length(1) +. Sepal.Width(2)+ Petal.Length(3)
Sepal.Length(1) +. Sepal.Width(2)+ Petal.Width (4)
Sepal.Width(2)+ Petal.Length(3) + Petal.Width (4)
Sepal.Length(1)+ Petal.Length(3) +Petal.Width (4)

Sepal.Length(1) +. Sepal.Width(2)+ Petal.Length(3)+ Petal.Width (4)

#Below are the results of each combination and the accuracy percentage follows.

```
>cluster1<- kmeans(iris_n[,c(1,2)],centers=3,nstart=25)
> table(cluster1$cluster,iris$Species)

   setosa versicolor virginica
 1      1        37        16
 2      0        13        34
 3     49         0         0
```

Accuracy1 = (49+37+34)/150= 0.8

```
> cluster2<- kmeans(iris_n[,c(1,3)],centers=3,nstart=25)
> table(cluster2$cluster,iris$Species)

   setosa versicolor virginica
 1      0        40        16
 2     50         3         0
 3      0         7        34
```

Accuracy2 = (50+40+34)/150= 0.82667

```
> cluster3<- kmeans(iris_n[,c(1,4)],centers=3,nstart=25)
> table(cluster3$cluster,iris$Species)

   setosa versicolor virginica
 1      0         3        36
 2      0        44        14
 3    50        3       0
```
Accuracy3 = (50+44+36)/ 150= 0.86667

```
> cluster4<- kmeans(iris_n[,c(2,3)],centers=3,nstart=25)
> table(cluster4$cluster,iris$Species)

   setosa versicolor virginica
 1     0       42       14
 2    50        0        0
 3     0        8       36
```
Accuracy4 = (50+42+36)/ 150= 0.85333

```
> cluster5<- kmeans(iris_n[,c(2,4)],centers=3,nstart=25)
> table(cluster5$cluster,iris$Species)

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

Accuracy5 = (49+46+44)/ 150= 0.92667

```
> cluster6<- kmeans(iris_n[,c(3,4)],centers=3,nstart=25)
> table(cluster6$cluster,iris$Species)

   setosa versicolor virginica
 1     0        2       46
 2    50        0        0
 3     0       48        4
```
Accuracy6 = (50+46+48)/ 150= 0.96

```
> cluster7<- kmeans(iris_n[,c(1,2,3)],centers=3,nstart=25)
> table(cluster7$cluster,iris$Species)

   setosa versicolor virginica
 1     0       40       15
 2    50        0        0
 3     0       10       35
```

Accuracy7 = (50+40+35)/ 150= 0.83333

```
> cluster8<- kmeans(iris_n[,c(1,2,4)],centers=3,nstart=25)
> table(cluster8$cluster,iris$Species)

   setosa versicolor virginica
 1     50          0         0
 2      0         11        41
 3      0         39         9
```
Accuracy8 = (50+39+41)/ 150= 0.8667

```
> cluster9<- kmeans(iris_n[,c(2,3,4)],centers=3,nstart=25)
> table(cluster9$cluster,iris$Species)

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

Accuracy9= (50+46+46)/ 150= 0.94667

```
> cluster10<- kmeans(iris_n[,c(1,3,4)],centers=3,nstart=25)
> table(cluster10$cluster,iris$Species)

   setosa versicolor virginica
 1      0          2        37
 2     50          0         0
 3      0         48        13
```

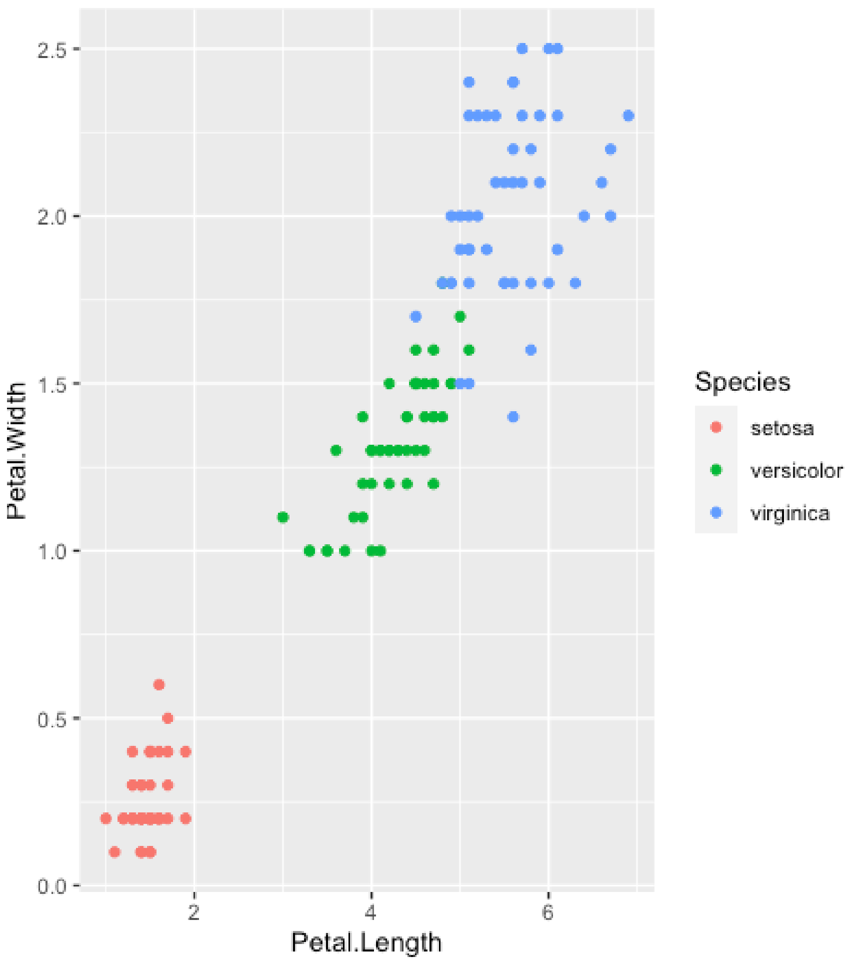
Accuracy10= (50+48+37)/ 150= 0.9

```
> cluster11<- kmeans(iris_n[,c(1,2,3,4)],centers=3,nstart=25)
> table(cluster11$cluster,iris$Species)

   setosa versicolor virginica
 1      0         47        14
 2     50          0         0
 3      0          3        36
```

Accuracy11= (50+47+36)/ 150= 0.88667

Result:

The optimal combination which gives the best accuracy rate is cluster 6. The accuracy rate is 96%. The predictor combination is Petal.Length(3)/ Petal.Width (4). Optimal center # is 3.


Petal.Width
Petal.Length
Species
setosa
versicolor
virginica