1 Question 3.1

We are tasked with finding a good classifier from `credit_card_data-headers.txt` using cross-validation - and splitting the data into training, validation and test data sets.
I have chosen to complete both tasks in one go. Since k-nearest-neighbors is required for (a), then that's what we'll be working with.
There are 654 data points, so I have chosen to use $\frac{2}{3}$ for training data, $\frac{1}{6}$ for validation data and $\frac{1}{6}$ for test data. This means, we'll have roughly 100 points each for validation and test, and 450 for training. I'll be using the rotation method for splitting the data, as I believe there is a lower risk of bias being introduced this way, compared to the risk of skewed data when using randomness.

1.1 The solution

First we load our libraries and import the data:

```
library(kknn)
library(kernlab)

# Load the data
data <- read.csv("credit_card_data-headers.csv")
```

Then we split the data up into our Test matrix and Training/Validation matrices.

```
# Create 5 Training/Validation matrices in a list and 1 Test matrix
# - with the headers from the CSV-file
tv_data <- list(data[0,],data[0,],data[0,],data[0,],data[0,])
test_data <- data[0,]

# Loop through the data and split into training/validation and test
# Using rbind to append lines to the end of each Matrix
for (i in 1:nrow(data)){
    if (i %% 6 == 0) {
        test_data <- rbind(test_data,data[i,])
    } else {
        tv_data[[i%%6]] <- rbind(tv_data[[i%%6]],data[i,])
    }
}
```

We now have 5 matrices for training and validation.
There are 5 different ways, you can create a training set from 4 of the matrices (and use the remaining matrix for validation). For each combination, we'll:

- Find the best k for the training data (trying k $1 \ldots 25$)
- Checking the accuracy for the "'best k"' using the validation data
- Checking if the current k has given a better accuracy than previous attempts

Here's the code to do that:

```
# Now try all 5 kombinations of training/validation data
best_k <- 0
best_acc <- 0
for (i in 1:5) {
    validation_data <- tv_data[[i]]

    # Empty training_data and append 4 of the 5 matrices from tv_data
    training_data <- data[0,]
      for (j in 1:5) {
        if (j != i) {
            training_data <- rbind(training_data,tv_data[[j]])
        }
    }

    # Find best k for current training data
    acc <- rep(0,25)
    for (n in 1:25) {
        p <- rep(0,(nrow(training_data)))
        for (r in 1:nrow(training_data)){
            # Run kknn and save the prediction result for each point
            model=kknn(R1~.,training_data[-r,],training_data[r,],k=n,scale=TRUE)
            p[r] <- as.integer(fitted(model)+0.5)
        }
        # Check the accuracy of the predictions
        acc[n] = sum(p==training_data[,11])/nrow(training_data)
    }

    # Now lets use validation to see how good our k is
    p <- rep(0,(nrow(validation_data)))
    for (r in 1:nrow(validation_data)){
        # Run kknn and save the prediction result for each point,
                # with k= the best result from training session
        model=kknn(R1~.,training_data,validation_data[r,],k=which.max(acc),scale=TRUE)
        p[r] <- as.integer(fitted(model)+0.5)
    }
    # Check the accuracy of the predictions
    val_acc = sum(p==validation_data[,11])/nrow(validation_data)


    if (val_acc > best_acc) {
        # If the best accuracy of this model is better than previous attempts,
                # then save k-value
        best_acc <- val_acc
        best_k <- which.max(acc)
    }
}
```

We know have a "'best k"' based on cross-validation. To find out, how good our model is, we'll test it with the test data:

```
# Build new model data and test with test data and the best_k found
model_data <- rbind(tv_data[[1]],tv_data[[2]],tv_data[[3]],tv_data[[4]],tv_data[[5]])
p <- rep(0,(nrow(test_data)))
for (r in 1:nrow(test_data)){
    model=kknn(R1~.,model_data,test_data[r,],k=best_k,scale=TRUE)
    p[r] <- as.integer(fitted(model)+0.5)
}
# Check the accuracy of the predictions
acc = sum(p==test_data[,11])/nrow(test_data)
acc
best_k
```

The result found is, that for a value of $k = 14$ we expect the model to be right 87% of the time.

2 Question 4.1

Living in a country with multiple political parties (currently 13 different parties in parliment), clustering could be used for predicting which party a person will vote for in the next election. Predictors could be:

- Age
- Income
- Zip code
- Marital status
- Primary method of transportation
- ...

3 Question 4.2

The task is to use `kmeans` to cluster the points from `iris.txt`.
First, we use `kmeans` without any real tweaking:

```
data <- read.csv("iris.csv")

ss = rep(0,10)
for (i in 2:10) {
    model <- kmeans(data[2:5],i)
    ss[i] <- model\$tot.withinss
}

plot(ss[-1], type="b")
```

This produces the plot Figure 1 As expected, the "'elbow"' is at 3 clusters, matching the number of categories in the data.
Let's see how accurate the clustering was. We know from the data, that there

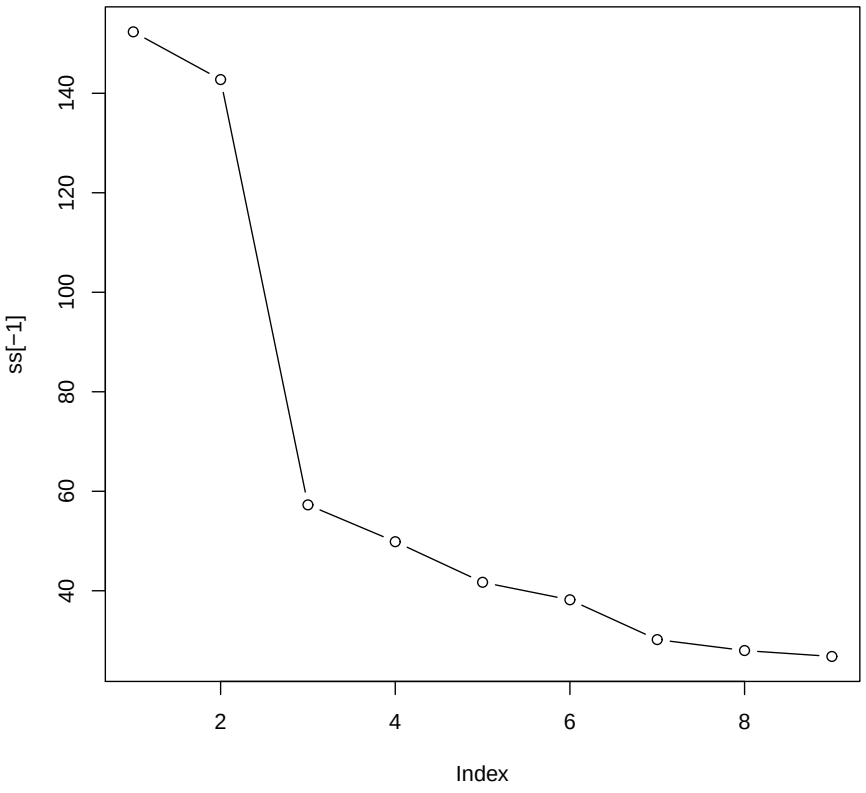

Figure 1: Total distance squared without any tweaking.

should be 50 points in each cluster, and that the categories are ordered. However, we don't know which flower will be which cluster, so we have to run through the 6 iterations.

```
iter <- list(c(1,2,3),c(1,3,2),c(2,1,3),c(2,3,1),c(3,1,2),c(3,2,1))

best <- 0
for (i in 1:6){
    correct = rep(iter[[i]],each=50)
    cor <- sum(model\$cluster == correct)/150
    if (cor > best) {
        best <- cor
    }
}
best
```

We currently get about 89% correct. Let's try only using 1, 2 or 3 of the predictors, and see if that improves our result.

```
predictors <- list(c(2,3,4),c(2,3,5),c(2,4,5),c(3,4,5),c(2,3),c(2,4),c(2,5),c(3,4),c(3,5),
res <- rep(0,length(predictors))
for (j in 1:length(predictors)) {
    model <- kmeans(data[predictors[[j]]],3)
    best <- 0
    for (i in 1:6) {
        correct = rep(iter[[i]],each=50)
        cor <- sum(model\$cluster == correct)/150
        if (cor > best) {
            best <- cor
        }
    }
    res[j] = best
}
max(res)
plot(res)
```

We see (Figure 2 & Table 3), that by only using the Petal.Width(5) we get an accuracy of 96%. We get the same result, by using both Petal.Length(4) and Petal.Width(5).

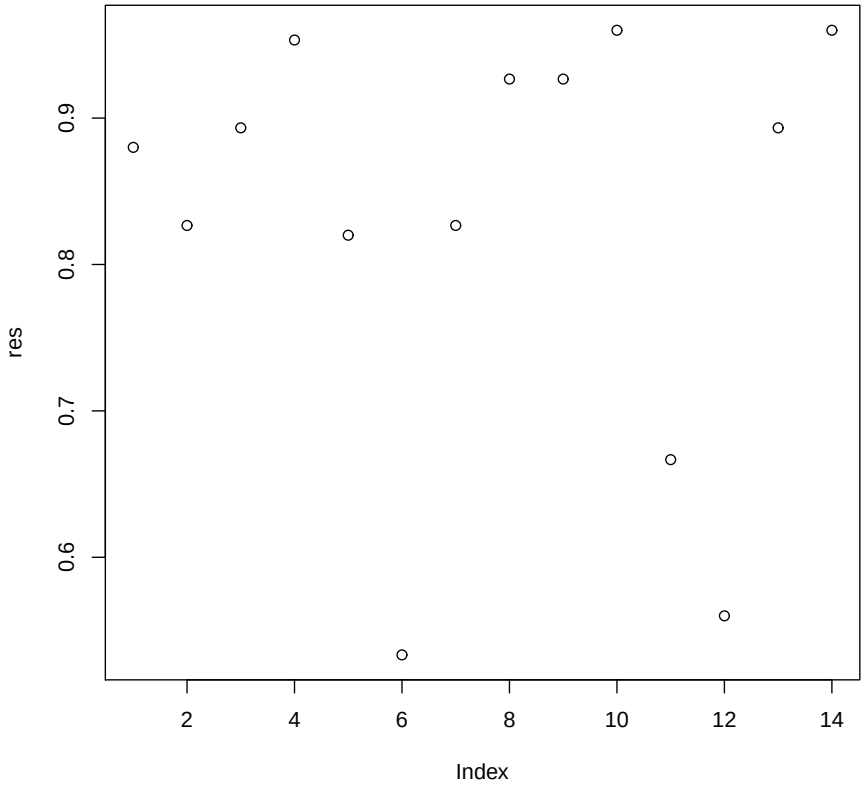

Figure 2: Accuracy depending on which predictors are used.

Index	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	χ	χ	χ	
2	χ	χ		χ
3	χ		χ	χ
4		χ	χ	χ
5	χ	χ		
6	χ		χ	
7	χ			χ
8		χ	χ	
9		χ		χ
10			χ	χ
11	χ			
12		χ		
13			χ	
14				χ

Figure 3: Predictors.