Lab1 - DATA MINING
Bürgi Robin
Sebastianelli Nicola

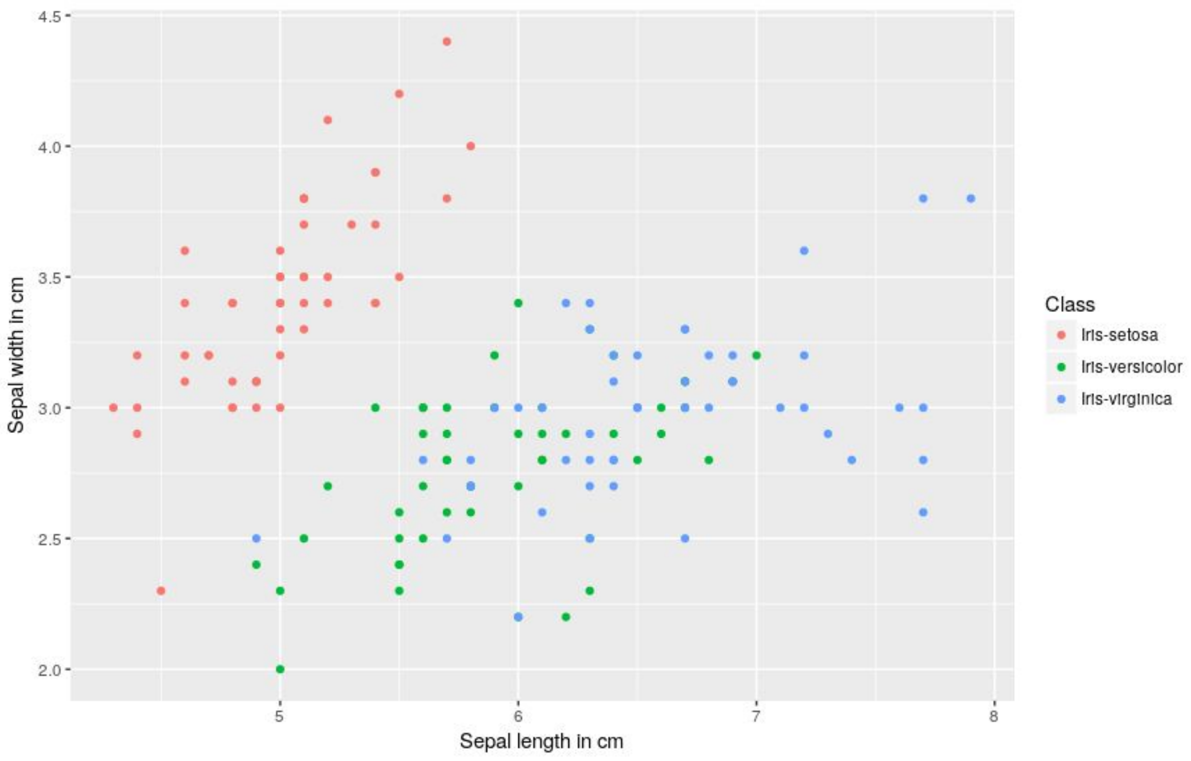
Problem:

The task is to try data mining methods in R with RStudio. This in order to give a deeper knowledge about data mining methods and how to use R and RStudio to apply them to real datasets.
The task will be to use the dataset named Iris and apply both the K-means and K-Nearest-Neighbor(KNN) data mining methods on the dataset. This in order to identify the classes in the Iris dataset.
The Iris dataset is a classical dataset and used in many examples when describing data mining and can be download from: http://archive.ics.uci.edu/ml/datasets/Iris.[1]

Data Exploration:

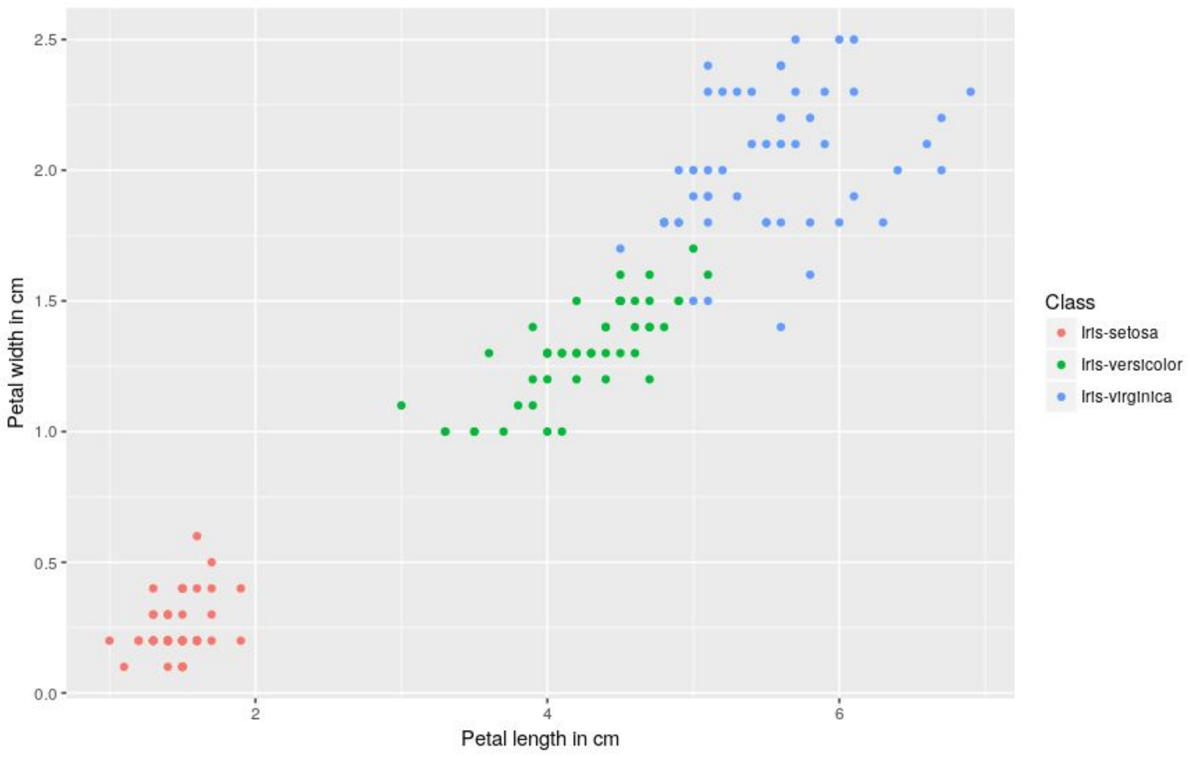
The Iris dataset consists of 150 instances (50 of each class) each containing sepal length in cm, sepal width in cm, petal length in cm, petal width in cm as well as the class.[2]
To explore and visualize the data we have grouped the data by class and created a plot of the sepal as well as petal size using ggplot2 library.

Plot of sepal size:



[1] Problem text taken from lecturer slides
[2] https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.names

Plot of petal size:



As we can see, the classes are much better grouped together using the petal width and length.

Clustering KMEANS:

For the first task we are suppose to apply the K-means clustering algorithm.
To have a better analysis of data we have applied the clustering algorithm one time considering just sepal, one time considering just petal and one time considering both.

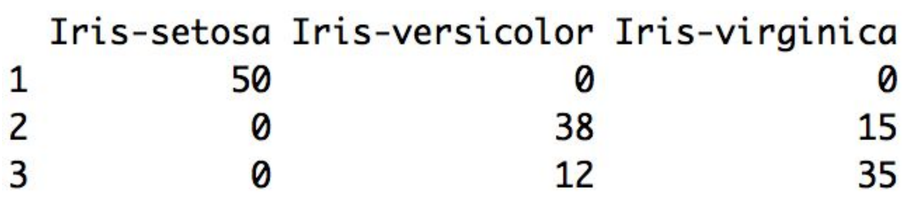
To apply the clustering we have used the R-function kmeans: `kmeans(data,k)`.
The set of data for our three tests is given by selecting every time all the data from the corresponding group e.g. : `irisData[,1:2]` select all the datas from column 1 and 2, that correspond at sepal length and width, from the iris matrix.
The number of clusters (the k value) was set to `length(unique(irisData[,5]))` which counts the number of unique classes in the dataset.

To compare the resulting clusters with the actual classes a comparison table was printed using the command `table(irisClusterSepal$cluster, irisData$C)`
A visualisation of each et result set was then created using ggplot.[3]

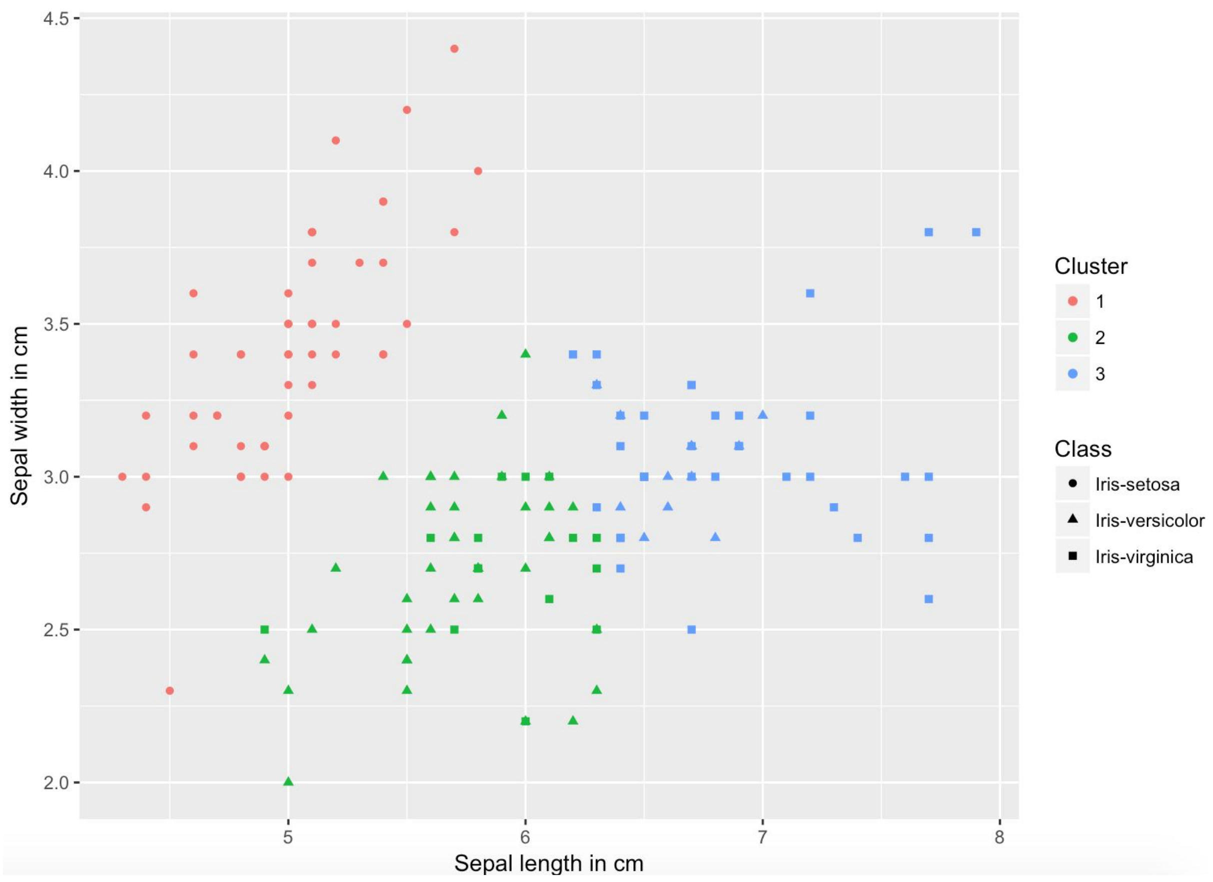
[3] https://www.r-bloggers.com/k-means-clustering-in-r/

Sepal clustering:

```
set.seed(as.numeric(Sys.time()))
irisClusterSepal <- kmeans(irisData[,1:2], length(unique(irisData[,5])))
irisClusterSepal
table(irisClusterSepal$cluster, irisData$C)
irisClusterSepal$cluster <- as.factor(irisClusterSepal$cluster)
ggplot(irisData, aes(SL, SW, color = irisClusterSepal$cluster,shape=C))
+ geom_point() +labs(color='Cluster',shape='Class',x ='Sepal length in
cm', y ='Sepal width in cm')
```



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



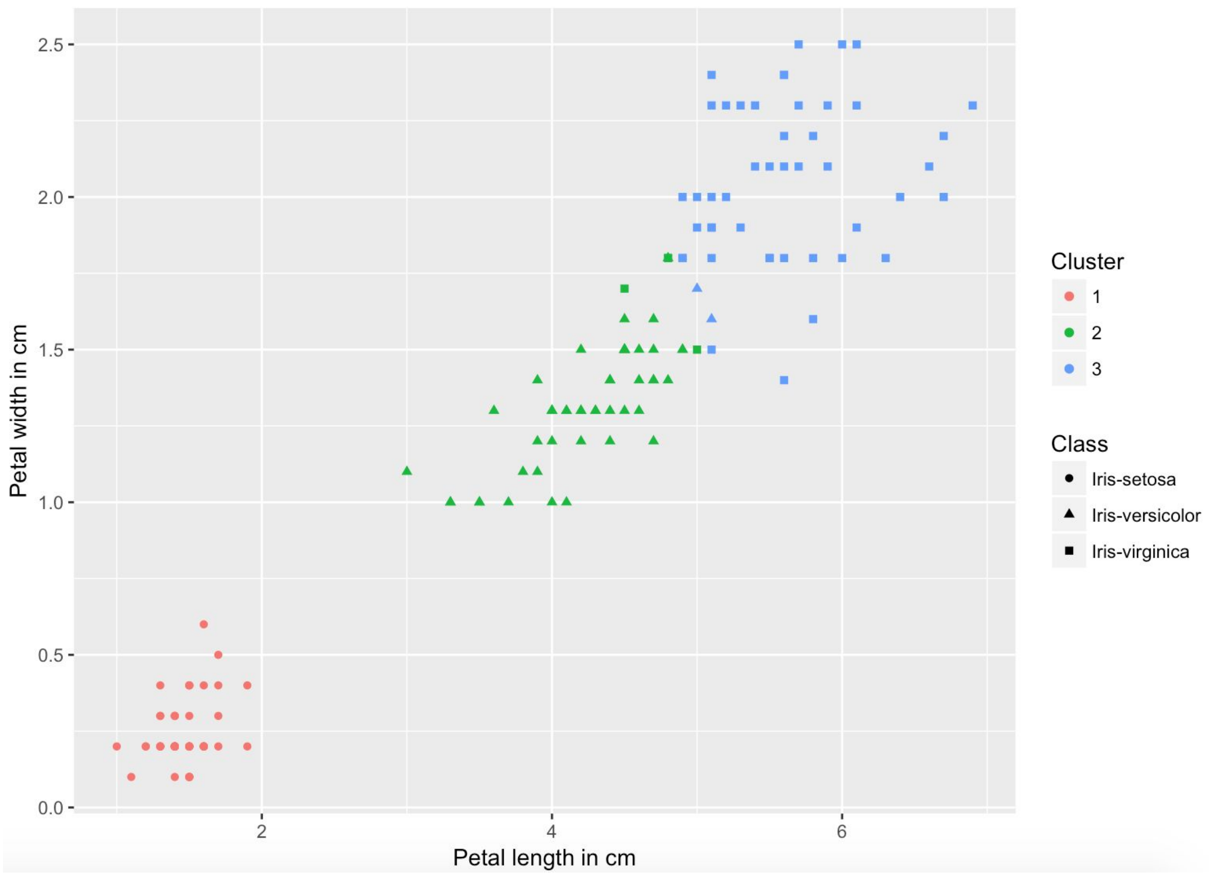
As we can see from the plot and the table, the Iris-setosa are all clustered correctly, but for Iris-versicolor and Iris-virginica there are some errors.

Considering the petal length and width we expect that the clustering will have less errors because as we have seen the classes have better grouping.

Petal clustering:

```
set.seed(as.numeric(Sys.time()))
irisClusterPetal <- kmeans(irisData[,3:4], length(unique(irisData[,5])))
irisClusterPetal
table(irisClusterPetal$cluster, irisData$C)
irisClusterPetal$cluster <- as.factor(irisClusterPetal$cluster)
ggplot(irisData, aes(PL, PW, color = irisClusterPetal$cluster,shape=C))
+ geom_point() +labs(color='Cluster',shape='Class',x ='Petal length in
cm', y ='Petal width in cm')
```

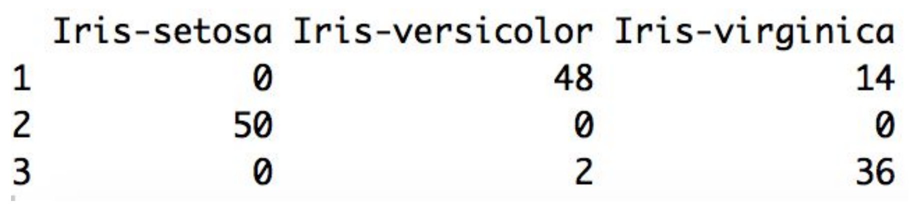
```
    Iris-setosa Iris-versicolor Iris-virginica
  1          50               0              0
  2           0              48              4
  3           0               2             46
```



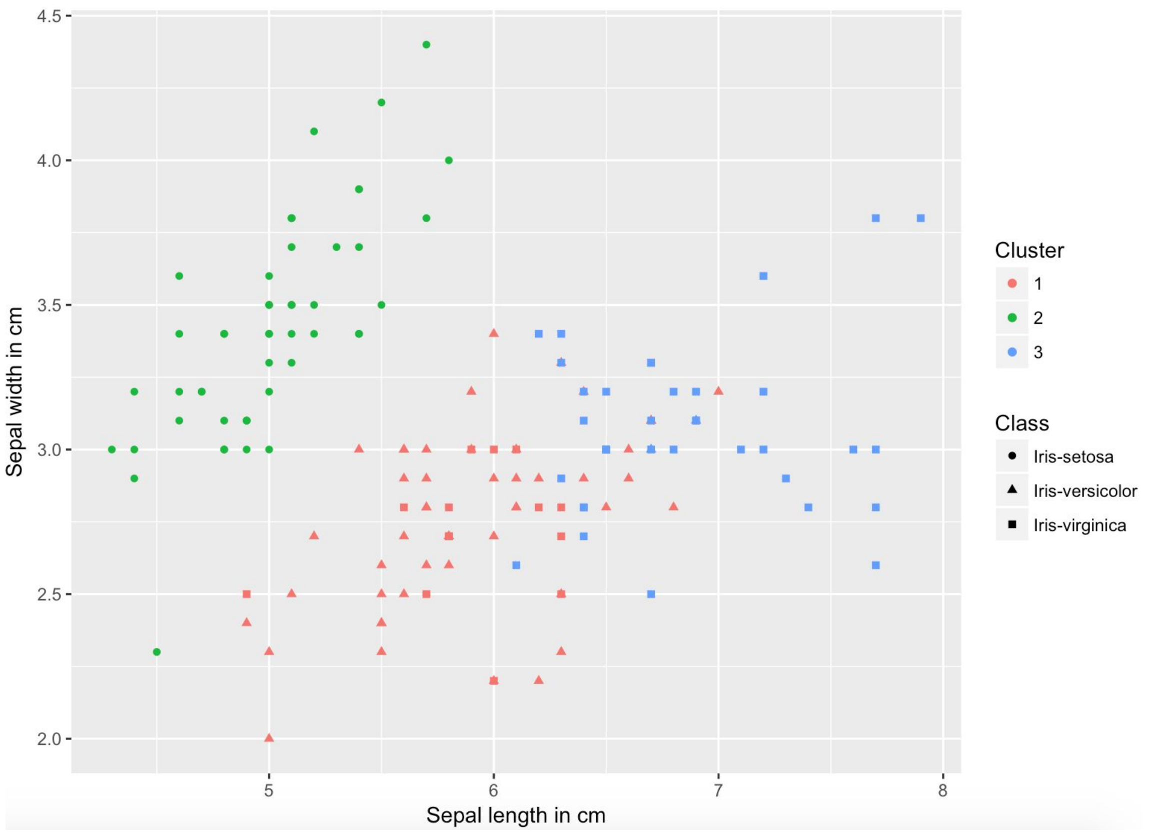
As expected the clustering considering just the petal length and width has less error, just 6 errors instead of 27 from the previous test.

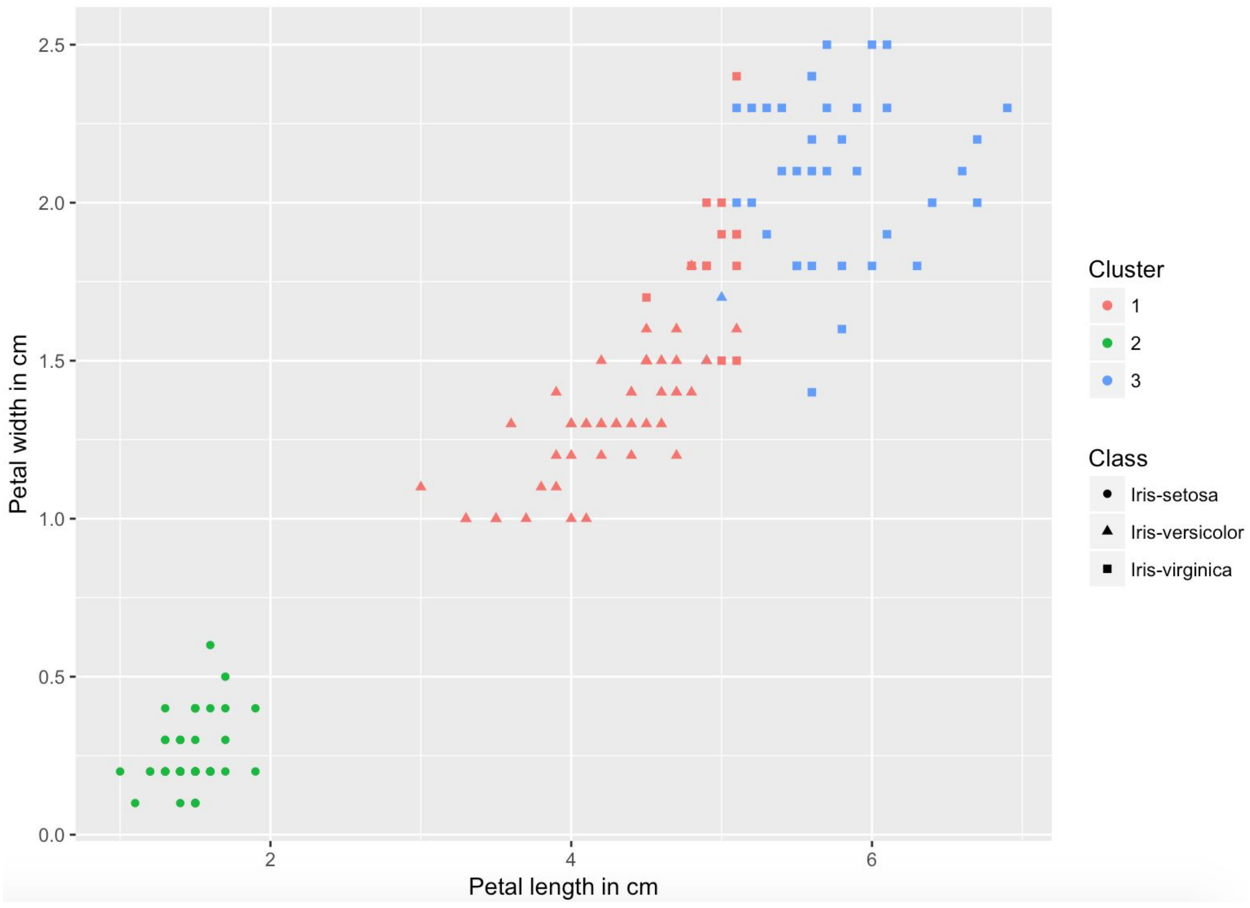
Sepal & Petal clustering:

```
set.seed(as.numeric(Sys.time()))
irisCluster <- kmeans(irisData[,1:4], length(unique(irisData[,5])))
irisCluster
table(irisCluster$cluster, irisData$C)
irisCluster$cluster <- as.factor(irisCluster$cluster)
ggplot(irisData, aes(SL, SW, color = irisCluster$cluster,shape=C)) +
geom_point() +labs(color='Cluster', shape='Class',x ='Sepal length in
cm', y ='Sepal width in cm')
ggplot(irisData, aes(PL, PW, color = irisCluster$cluster,shape=C)) +
geom_point() +labs(color='Cluster', shape='Class',x ='Petal length in
cm', y ='Petal width in cm')
```



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





Considering now both the sepal and petal length and width we can see that the number of flowers wrongly clustered are 16.

We can assume that if we want to cluster Iris flower is better to cluster them just by using petal length and width because of the lower error ratio.

Classification KNN:

The second task was to apply the K-nearest neighbors algorithm to the dataset to create a classifier.

We defined a function to normalize the data and applied it to the iris dataset using `lapply`.

```
normalize <- function(x) { return ((x - min(x)) / (max(x) - min(x))) }
iris_normalized <- as.data.frame(lapply(irisData[1:4], normalize))
summary(iris_normalized$sepal_length) # check if normalized
```

To train the classifier the dataset had to be splitted into a training set and a test set. We decided to use half the data to train and half the data to test the resulting classifier. The `rbind`-function was used to combine multiple data frames together.

```
# creating training data from normalized dataframe
iris_train <- rbind(iris_normalized[1:25,], iris_normalized[51:75,],
iris_normalized[101:125,])
iris_train_labels <- c(irisData[1:25,5], irisData[51:75,5],
irisData[101:125,5]) # create a vector of training labels
iris_test <- rbind(iris_normalized[26:50,], iris_normalized[76:100,],
iris_normalized[126:150,])
iris_test_labels <- c(irisData[26:50,5], irisData[76:100,5],
irisData[126:150,5]) # create a vector of test labels
```

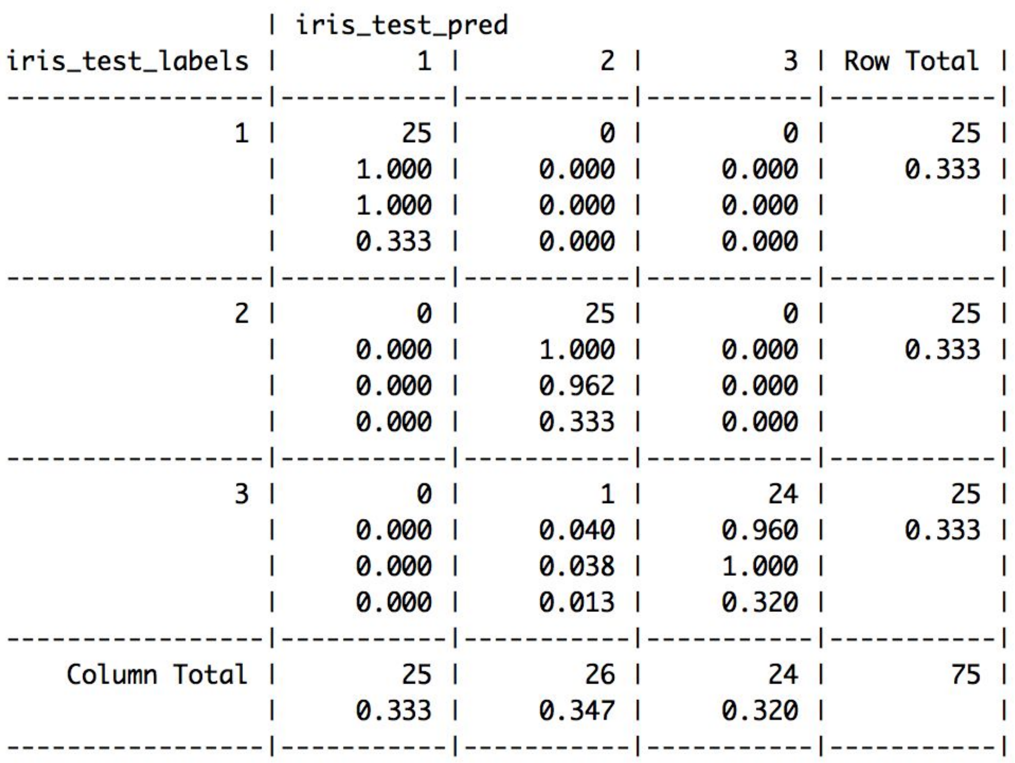
The classifier was now created using the R-function `knn(training_set, test_set, training_labels, k)`.
The k value was defined as the square root of the total number of instances in our dataset as it is said to be a good starting point for most datasets.[4] We used the `CrossTable`-command to evaluate the predictions. Further we created a accuracy function that returns a value from 0 to 1 that represents the accuracy of our predictions.

```
# creating predictions with k=sqrt(total values)
iris_test_pred <- knn(train = iris_train, test = iris_test,cl =
iris_train_labels, k=sqrt(nrow(irisData)))

CrossTable(x = iris_test_labels, y = iris_test_pred, prop.chisq = FALSE)
# print cross table to evaluate predictions

accuracy = function(actual, predicted) { mean(actual == predicted) } #
define function to calculate accuracy
accuracy(actual = iris_test_labels, iris_test_pred) # calculate accuracy
for the prediction above
```

[4] https://www.researchgate.net/post/How_can_we_find_the_optimum_K_in_K-Nearest_Neighbor



```
                 | iris_test_pred 
iris_test_labels |         1 |         2 |         3 | Row Total | 
-----------------|-----------|-----------|-----------|-----------|
               1 |        25 |         0 |         0 |        25 | 
                 |     1.000 |     0.000 |     0.000 |     0.333 | 
                 |     1.000 |     0.000 |     0.000 |           | 
                 |     0.333 |     0.000 |     0.000 |           | 
-----------------|-----------|-----------|-----------|-----------|
               2 |         0 |        25 |         0 |        25 | 
                 |     0.000 |     1.000 |     0.000 |     0.333 | 
                 |     0.000 |     0.962 |     0.000 |           | 
                 |     0.000 |     0.333 |     0.000 |           | 
-----------------|-----------|-----------|-----------|-----------|
               3 |         0 |         1 |        24 |        25 | 
                 |     0.000 |     0.040 |     0.960 |     0.333 | 
                 |     0.000 |     0.038 |     1.000 |           | 
                 |     0.000 |     0.013 |     0.320 |           | 
-----------------|-----------|-----------|-----------|-----------|
    Column Total |        25 |        26 |        24 |        75 | 
                 |     0.333 |     0.347 |     0.320 |           | 
-----------------|-----------|-----------|-----------|-----------|
```

To verify that we chose the best possible k value for our dataset we created the following program that creates a knn classifier for each k value from 1 to 25. Each classifier was then evaluated using our accuracy function. The results were then visualized by plotting the accuracy of the classifiers and the corresponding k values.

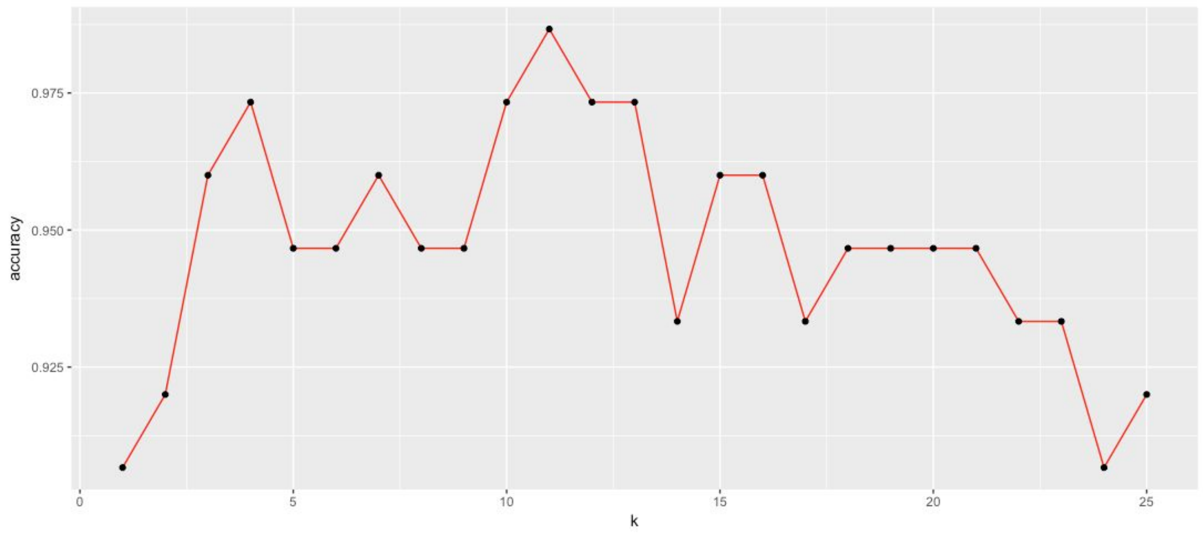
```r
# Verify
k=sqrt(total values) as best value for k
k_max <- nrow(iris_train)/3
results <- vector("list", k_max)

# make knn classifier for each k and write the according accuracy into
the results vector
for (k in 1:k_max){
  iris_test_pred <- knn(train = iris_train, test = iris_test,cl =
iris_train_labels, k=k)
  a <- accuracy(actual = iris_test_labels, iris_test_pred)
  results[k] <- a
}

# create a dataframe from the vector
df <- data.frame(accuracy=matrix(unlist(results), nrow=k_max, byrow=T))
df$k <- as.numeric(row.names(df)) # write index to dataframe as it
represents k in the vector

# plot a line graph of all k values and the corresponding accuracy
```

```
ggplot(data=df, aes(x=k, y=accuracy, group=1)) +
  geom_line(color = "red")+
  geom_point()
```



As can be seen in the graph above, a k value of 11-12 will give us the highest accuracy.

Conclusion:

As we have seen for the clustering process, using the k-mean method, is better to use just the petal length and width because of the grouped data structure and so have minor clustering errors.
For the classification process, using the KNN algorithm, to have an high accuracy is important to find the correct K value. With the previous test we have proved that the best accuracy for classification is given by the square root of the number of instances contained in the dataset.