Data Visualization and Analysis

BINF4245

Prof. Dr. Renato Pajarola

Exercise and Homework Completion Requirements

1. Every **solved** exercise will earn you points, up to 15 in total.
   - *Late submissions (up to one day) will result in "-1" point.*
   - *If the solution works as expected, full points are awarded, else partial points are assessed as outlined in the code skeleton.*
2. The four exercises give rise to the following point distribution: 2 – 3 – 5 – 5.
   - *A **minimum of 7 points** from all four exercises must be achieved to pass the module. Failure to achieve this minimum will result in a failing grade for the entire module.*
   - *Thus at least two exercises must be correctly solved, and one must be from the more advanced ones.*
3. We give **bonus points** for students who have completed more than 8 points from all the exercises.
   - *Thus **7 points** from the exercises is required, **8 points** is still normal passing, and **9 and above** would give 1 or more extra bonus points.*
   - *Only the bonus points can and will be added directly to the final exam.*
4. Do not copy assignments, tools to detect copying and plagiarism will be used.
   - *The exercise results are an integral part of the final course grade and therefore the handed in attempts and solutions to the exercises **must be your personal work**.*

Submission Rules

- Submitted code must run without errors using the indicated Python environment, using the included libraries, packages, and frameworks.
- The whole project source code must be zipped and submitted before the given deadline, including the output results (saved in .html file or as a screenshot picture).
- Submit your .zip archive named *dva_ex1_Firstname_Lastname_MATRIKELNUMBER.zip* (e.g. dva_ex1_Hans_Muster_01_234_567.zip) through the OLAT course page.
- You zip file should contain:
  - Your working code as dva_ex3_Hans_Muster.py file
  - A screenshot or export of your tool as .jpg, .png, .pdf or .html
  - An readme.txt if you needed additional libraries and good reasons why you needed them.
- **Deadline is Sunday, 15 May 2022 at 23:59h**

Exercise 3: Clustering

Clustering methods partition a dataset into different groups based on a given metric, typically the Euclidean distance. In this exercise, you will work with the well-known multivariate data set *Fisher's Iris* with a specific focus on data processing and clustering methods. What you need to do is to first present the dataset as two scatter plots then implement the k-means clustering algorithm together with a slider widget to choose the number of clusters (k) and a select to switch between random or fixed initialization. Detailed requirements for each step are described below.

Step 1: On startup, visualize the dataset (using gray color) in two scatter plots with the following axis:

Axis/Plot	**Plot1**	**Plot2**
X	Petal length	Petal width
Y	Sepal length	Petal length

Step 2: The number of clusters k is given by a slider widget with a range between 2 and 10, the default should be 3. Whenever this slider changes, perform a k-means clustering and display the result by coloring the datapoints according to their assigned cluster.

The goal of any cluster algorithm is to groups by assigning *cluster labels* to each data point of a dataset. Typically, the labels are integers {0, *k* - 1} where *k* is the number of clusters. If a clustering algorithm returns a list [0,1,1] it means that the first data point belonged to cluster 0 while the second and the third belonged to cluster 1.

K: number of clusters, n_iter: the maximal number of iterations

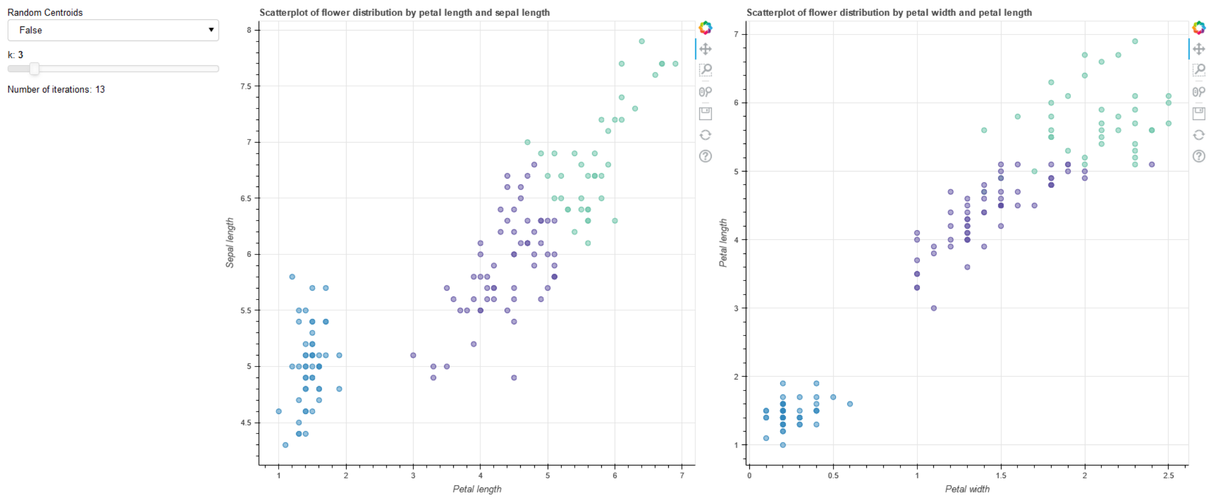
k-Means algorithm:

```
1.  Initialize k random centroids by assigning k random labels to each data point
2.  While i < n_iter:
1.  Compute the new centroids by calculating means of each cluster
2.  If the centroids didn't change, terminate
3.  Else, for each point, find the closest centroid using the Euclidean distance and update the cluster label
```

Step 3: Implement a dashboard with two controls, one to change between a random and preset initialization. The reason for this is that the result of k-means depends on the initial centroids. By using non-random centroids, you should receive the same result as shown in the sample images. If the controls are changed by the users, the kmeans algorithm should be recomputed, and the colors of the data points should be updated.

The second controls widget is a slider to select a *k* between 2 and including 10. If any of the two controls change, update the clustering using a bokeh server callback.

Step 4: Implement a Div which displays the number of iterations before the algorithm terminated

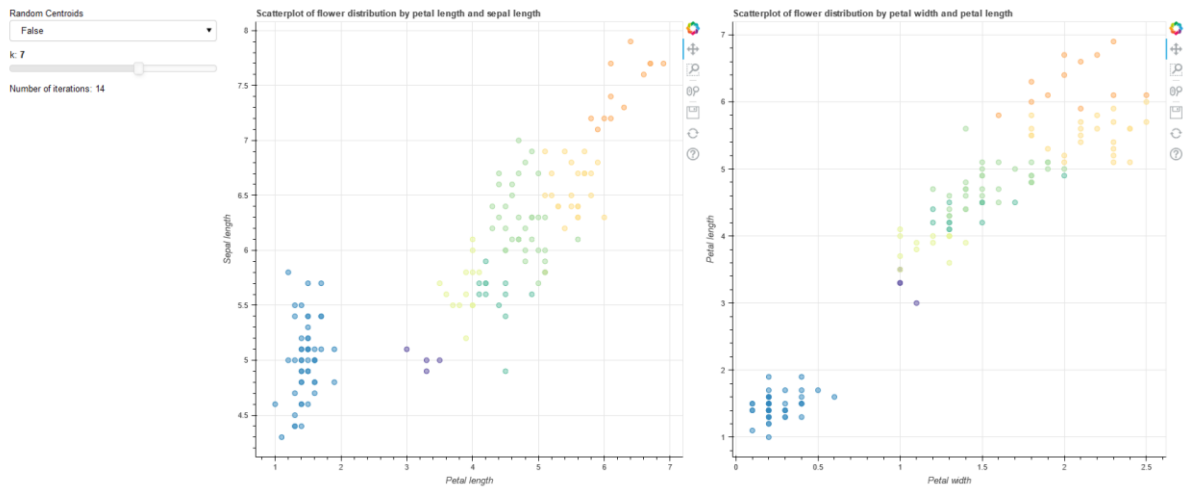

Random Centroids

False

k: 3

Number of iterations: 13

Scatterplot of flower distribution by petal length and sepal length
Sepal length
Petal length
Scatterplot of flower distribution by petal width and petal length
Petal length
Petal width


Random Centroids

False

k: 7

Number of iterations: 14

Scatterplot of flower distribution by petal length and sepal length
Sepal length
Petal length
Scatterplot of flower distribution by petal width and petal length
Petal length
Petal width

Screenshot: How the resulting tool should look.

Remarks:

- Try to make good use of the hints and references provided in the skeleton code. (**very important**)
- Try to google first for any Python related issues/bugs.
- Due to the special situation, we don't arrange in person meeting in this semester. Please contact me (Gaudenz Halter, halter@ifi.uzh.ch) for technical questions regarding the exercise only as a last resort.
- More than one day late submission will not be accepted and graded.
- If the submitted solution works as expected, full points will be rewarded, else, we follow the points indicated in the template
- **Online office hour** will be on **Friday, 6 May 16:00 – 17:00**. The zoom link will be communicated in that week, prior to the office hour, via email.