Data Visualization and Analysis

BINF4245

Prof. Dr. Renato Pajarola

Exercise and Homework Completion Requirements

- Exercises and reading assignments are **mandatory** and they must be completed successfully to finish the course and get a sufficient passing final grade.
- Exercises are graded coarsely into categories **pass** or **fail**.
  - *A **fail** is given to failed submissions and partial solutions, and no points are awarded.*
  - *A **pass** indicates that the exercise is sufficiently good to receive the corresponding points.*
  - *Late submissions (up to one day) will result in "-1" point.*
- The four exercises are allocated to the following point distribution: 2 – 3 – 5 – 5
  - A **minimum of 7 points** must be achieved to pass the module.
  - Thus, at least two exercises must be solved correctly, including at least one from the advanced ones.
  - *Failure to achieve this minimum will result in a failing grade for the entire module*
- We award **bonus points** for students who have collected more than 8 points from all the exercises.
  - Thus, **7 points** from the exercises is required, **8 points** is still normal passing, **9 and above** would give 1 or more extra bonus points.
  - Only the bonus points can and will be added directly to the final grade.
- 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, included libraries, packages and frameworks. If additional libraries/packages are needed, please specify these in a *readme.txt* file together with your submission.
- The whole project source code must be zipped and submitted before the given deadline, including exactly the files indicated in the exercise.
- Submit your .zip archive named *dva_ex2_Firstname_Lastname_MATRIKELNUMBER.zip* (e.g. dva_ex1_Hans_Muster_01234567.zip) through the OLAT course page.
- **Deadline is Monday, 30 November 2020 at 23:59h**

Exercise 3

In this exercise, you will work with the well-known multivariate data set "Fisher's Iris data set", 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-medoids clustering algorithm together with a select widget for random or fixed medoids, and a button to run the algorithm. 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: Plot1 x: Petal length, y: Sepal length. Plot2 x: Petal width, y: Petal length

Step 2: The number of clusters ***k*** in this exercise is fixed to 3. To finish step 2, you need to use the pre-defined medoids (24, 74, 124) and set them as the initial medoids of the 3 clusters. Use the k-medoids algorithm to refine the medoids, assign each data point to a medoid, and assign each cluster a color.

Additional hints and information:

- The fixed medoids (24, 74, 124) are indices into the data array where the actual data points are located.

- Use the following definitions:

1) The distance of a data point to a medoid is the sum of the absolute differences between the parameters petal length, petal width, sepal length and sepal width. This is also called the Manhattan distance.

2) The cost of a single data point is the minimum of its distances to all medoids.

3) The cost of the clustering is the sum of the costs of all data points i.e. the sum of all minimal distances of data points to their closest medoid.

- We use the Partitioning Around Medoids (PAM) implementation. This approach uses a greedy search, which is the reason why random medoid selection can lead to different results.

Working mechanism:

1) Initialize k data points as medoids. We always use k = 3 in this exercise.

2) Assign each data point to the closest medoid and calculate the clustering cost.

3) While the cost of the configuration decreases:

    3.1) For each medoid m and for each non-medoid data point p:

        3.1.1) Swap p with m and recalculate the cost of the clustering with p as new medoid.

        3.1.2) If the new cost is lower than the current cost, remember the combination and revert the changes.

3.2) Swap the best p and m combination, if it decreases the cost. If no cost decreasing swap can be performed, terminate the algorithm.

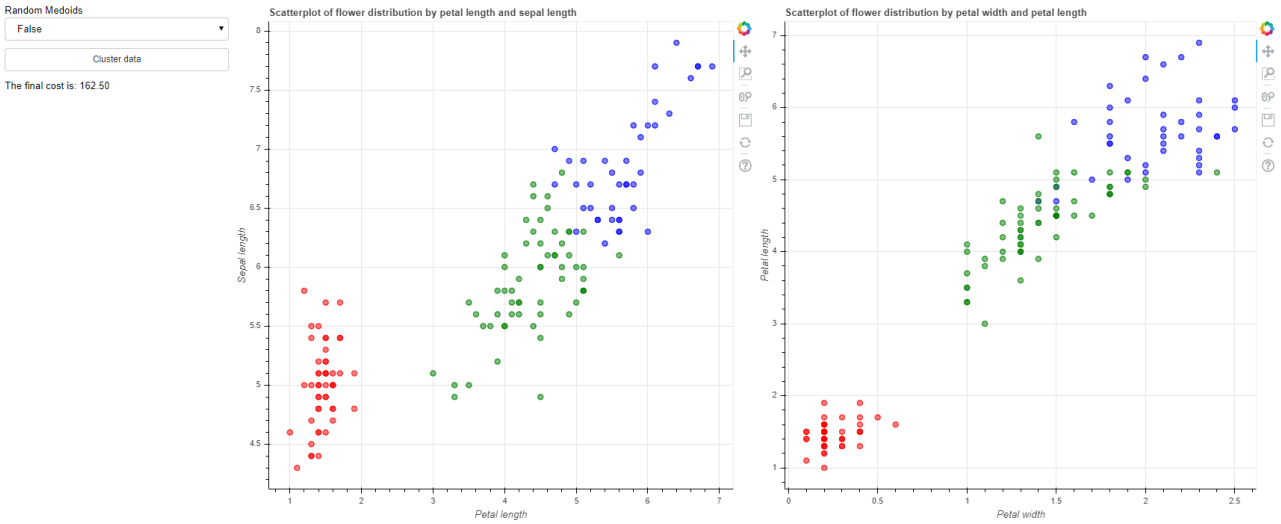
- Additional information can be found here: https://en.wikipedia.org/wiki/K-medoids

- Use numpy array functions to parallelize calculations and significantly improve the speed of your code.

- Plot the glyphs of the scatter plot with an alpha value < 1.0 such that overlapping points are discernible.

Step 3: Extend your algorithm such that either the pre-defined medoids or random medoids can be used. The decision which strategy is used is made by the value of the select widget.

Step 4: Display the final cost of your clustering in the dashboard.



Important: All deliverables of this exercise must be submitted before the deadline. The absence of any required files will automatically lead to a **FAIL**.

Submission:

- clean version of your code file with proper comments (**.py** format. **No** .ipynb files!) that runs without any error using a variation of the "bokeh serve" command.

- readme.txt – Use this file for your comments or remarks (if necessary). If you used any additional libraries that are not imported in the skeleton explain why and for what in this file. This file may be empty if you have no comments and if you used only the provided libraries.

- An export of the final dashboard in .pdf or .jpg format. (A screenshot is also accepted.)

- Put all required files into a .zip archive using the naming scheme detailed on the first page of this document. Put the files directly into the archive and do not package the root folder.