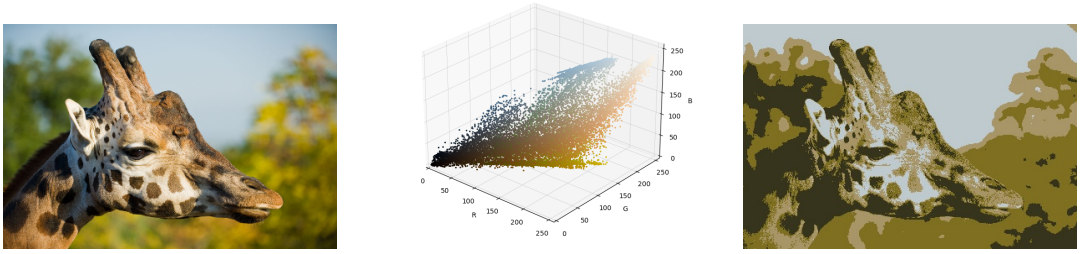
Practical Assignment
Task Sheet

Task 1 – Implementation of Clustering Algorithms

The task deals with the implementation of the two clustering algorithms `k-means` and `DBSCAN` (see [2] and [1]). As a concrete clustering problem, the segmentation of color images using clustering algorithms will be used. Therefore, the objects that are to be clustered are the pixels of any given image. Each pixel is represented by its vector in the RGB color space (3-dimensional vector).
Design and implement a program which takes an input image and clusters its pixels accordingly. The output of the segmentation will be the image with each pixel colored by its cluster's average color. An example can be seen in Figure 1 below.



(a) Original image (b) Pixel vectors in 3D space (c) Segmented image

Figure 1: Example segmentation

List of requirements:

1. Using the source code and potential documentation documents, the program must compile and run without major crashes.
2. The implemented algorithms have to work correctly (according to their underlying theory) and in a reasonable amount of time.
3. For the algorithms, the following parameters should be available and arbitrarily selectable:
   - `k-means`: `k` $\in \mathbb{N}$
   - `DBSCAN`: `eps` $\in \mathbb{R}$, `min_samples` $\in \mathbb{N}$
4. For the distance functions, Euclidean, Manhattan and Maximum should be implemented! It must be possible for the user to choose them freely.
5. The clustering algorithms have to be implemented by you, personally. Detected plagiarism will result in immediate failure of the task.

6. The clustering algorithms should be implemented in a general way, so that they work with any given feature vectors in any dimensionality and not just with specified example (image) data.
7. The program must have a command line interface with which to test it.

Task 2 - Classification Experiments

In this task, you will use classification methods on given datasets. Furthermore, the simple classification method *k-Nearest-Neighbor* will be implemented and compared to other available algorithms and implementations.

List of requirements:

1. Write a Python class named `KNNClassifier` that performs classification using the K-Nearest Neighbors algorithm. Your implementation should include:
   - A `fit(X, y)` method that stores the training data.
   - A `predict(X)` method that predicts the class labels for the provided data points.
2. Choose **two datasets** from the list in Section . Then, train and evaluate your `KNNClassifier` implementation on both datasets. Additionally, do the same with the KNN classifier and a third classifier of your choice from `scikit-learn` for comparison. Choose suitable hyperparameters and split your data into suitable *TR* and *TE*.
3. For each classifier, record the *accuracy* on the test set and the *runtime* for both training and prediction each. Compare the performance of all three classifiers. Also generate a confusion matrix for each of your experiments.
4. Create a brief scientific report summarizing your results (around 2-3 pages). Your report should contain:
   - Experiment setup
     - What datasets and classifiers were used?
     - How were the training / test samples selected?
     - How many different runs were executed for a certain training/test split?
     - How were the parameters for the classification method optimized?

     ...
   - Presentation of results
     - Accuracy scores for each classifier on each dataset
     - Runtime for training and prediction for each classifier on each dataset
     - Visualisations of the trained classification model (if applicable, e. g. plot of a decision tree, ...)

     ...
   - Discussion of the results

   Conclude with a short analysis of the results. Discuss any observed differences in accuracy and runtime between your implementation and the `scikit-learn` versions.

Datasets

This section lists the data that you may use in your experiments and where to retrieve it from.

1. **Iris Dataset**
   Dataset for multiclass classification with 150 samples and 4 features. The goal is to classify the samples into 3 species of iris.
   Source: UCI Machine Learning Repository - Iris Dataset or via `sklearn.datasets.load_iris()`
2. **Breast Cancer Wisconsin Dataset**
   Binary classification dataset with 569 samples and 30 features (e.g., mean radius, texture, perimeter) extracted from cell images. The goal is to predict whether a tumor is benign (Class 0) or malignant (Class 1).
   Source: UCI Machine Learning Repository - Breast Cancer Wisconsin Dataset or via `sklearn.datasets.load_breast_cancer()`
3. **Wine Quality Dataset**
   Dataset with 4,898 samples of red and white wines, each with 11 chemical properties (e.g., alcohol content, pH, residual sugar). The task is to classify the wine quality, with quality ratings as 6 classes ranging from 3 to 8.
   Source: UCI Machine Learning Repository - Wine Quality Dataset
4. **Titanic Survival Dataset**
   Dataset for binary classification, predicting passenger survival (0 for did not survive, 1 for survived). It includes 891 samples and 12 features, such as passenger age, gender, ticket class, and fare paid.
   Source: Kaggle - Titanic: Machine Learning from Disaster
5. **MNIST Dataset**
   Dataset with 60,000 training and 10,000 test grayscale images of handwritten digits (0-9) at a resolution of 28x28 pixels, with each digit as a separate class, making it a 10-class classification task.
   Source: Kaggle - MNIST Dataset

If you are curious to explore a different dataset than the ones listed here, please consult the exercise instructor for permission.

Submission

1. The submission is to be done in Moodle. The submission link deactivates after the deadline.
2. Make sure to not save it as a draft but to really explicitly submit it.
3. Submission is only possible if you are registered in a group.
4. In case the submission on Moodle is not working, email your files to stahlale@b-tu.de instead.
5. Submit all files that are required to run and test your implementation and/or to reproduce your experiments. If necessary, include documentation documents.

Bibliography

[1] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[2] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.