Practical Training - Task Sheet 2
Implementation and Application of Classification Methods

1 Task Description

In this task, you will use classification methods on a given dataset which is described in Section 2. Furthermore, the simple classification method *k-Nearest-Neighbor* will be implemented and compared to other available algorithms and implementations.

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 2. 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.
4. Export a confusion matrix for each of your experiments. (See Appendix A)
5. Create a brief scientific report summarizing your results (around 2-3 pages). For example, your report may 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.

2 Datasets

This section lists the datasets 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: Yann LeCun's Website - MNIST Dataset

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

3 Submission

1. Submission must be done before the submission date given on the Moodle page.
2. The submission is done over Moodle
3. Your digital submission must contain:
   a) Source code of your KNN implementation.
   b) Source code of all used programs/scripts
   c) The results of your experiments in CSV format. See Appendix A for the format requirements.
   d) Your scientific report that discusses the results
   e) A short(!) textual explanation of the installation, execution and operation (README.txt) of your program for the experiments.

Note: Make sure to follow the format requirements listed in the apendix! We will do an initial automated check of all submissions. Only those submissions that pass this check will be evaluated further!

Appendix A: CSV Format of Confusion Matrix

In order to make the files readable for our evaluation scripts, please adhere to these requirements:

File Names:

1. Each file should be named in the scheme
   `g<group number>_d<dataset no.>_c<classifier no.>.csv`, with:
   - `group no.` = your group number with leading zeros to make it three digits
   - `result no.` = number of the explored dataset (corresponds to number in Section 2)
   - `result no.` = number of your used classifier (should be defined in your reportb)
2. Since you have to train three classifiers on two datasets, you should produce a total of six result files.
   For example, group 5's submission could look like this:
   - `g005_d1_c1.csv`
   - `g005_d1_c2.csv`
   - `g005_d1_c3.csv`
   - `g005_d3_c1.csv`
   - `g005_d3_c2.csv`
   - `g005_d3_c3.csv`

Format:

Please use the `sklearn` library to generate and `pandas` to export your confusion matrices. Below is a short description of how to achieve this:

1. Import the required libraries:

```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
```

2. Generate the confusion matrix using `confusion_matrix(y_true, y_pred)` where `y_true` are the true labels and `y_pred` are the predicted labels:

```
y_true = [1, 0, 1, 1, 0, 1, 0]
y_pred = [1, 0, 1, 0, 0, 1, 1]
conf_matrix = confusion_matrix(y_true, y_pred)
```

3. To export the confusion matrix, save it to a text file using `pandas.DataFrame.to_csv`:

```
df = pd.DataFrame(conf_matrix, index=["Actual_0", "Actual_1"], columns=["
    Pred_0", "Pred_1"])
df.to_csv("g001_d2_c3.csv")
```