Coursework Description

(100% of the module assessment)
Submission Deadline: 11am on 7^{th} May 2025

1 Overview

This coursework requires you to apply the skills and knowledge acquired during the module using a real-world dataset for classification. You can access the datasets for this assessment in the folder *Assessment/Dataset* in LEARN.

2 Dataset

The dataset contains images of various locations on earth. The images have been classified according to the use of the land covered by the image. This is the classification problem you will be looking at in this coursework.

3 Coursework Files

As part of this Coursework (CW), you have been provided with the following files:

- **CW description** is this file which contains all the details of this CW. It contains information about the questions you have to answer, how you should prepare your submission and what I expect in your submission.
- **Assessment rubric** provides information about the four categories along which your CW submission will be marked.
- **coc131_cw.py** is the python script in which you should write your code. This is the file that you need to upload in your CW submission along with other material requested.

4 Questions

This CW has six questions. The file coc131_cw.py already contains six functions *q1* to *q6*. Write your responses to *Question 1* in the function *q1* and so on. In all questions (except 1), you also need to produce visualizations. You should

upload these visualizations along with the submission (see Section 5 for file type and naming rules to follow).

1. Load the dataset as specified in the docmentation for the function *q1* in file coc131_cw.py.
2. Write an implementation for standardizing a given input dataset in *q2* according to the documentation of the function.
    - Using a visualization, illustrate the impact of standardization on the dataset.
3. Build a MLP classifier using the given dataset. You should use the holdout method for building and evaluating your model. 30% of the dataset should be reserved for testing.
    - Write the optimal hyperparameters you obtain in the variable *optimal_hyperparam* at the top of the file coc131_cw.py.
    - Generate visualization(s) to summarize the results of your hyperparameter optimization.
4. Study and visualize the impact of the hyperparameter alpha on the parameters and performance of an MLP classifier. By parameters, I mean the weights and biases of the trained model.
    - Create a visualization that shows how different alpha values impact the model parameters and the performance of the model. The visualization(s) should clearly show how alpha affects model parameters and performance.
5. Use hypothesis testing to compare the effect of CV with and without stratification. You should use 5-fold CV for this question.
    - Create a visualization supporting the conclusions from hypothesis testing.
6. Use LocallyLinearEmbedding in sklearn to obtain a 2-dimensional representation of all samples in the dataset. Create a visualization demonstrating the separability of the classes in the 2-dimensional space obtained using LocallyLinearEmbedding.

5 Preparing your Submission

Your submission should be in a zip file named **coc131_cw_<student-number>.zip**. Replace "<student-number>" with your student number. The zip file should contain the following items:

- coc131_cw.py with your implementation.

- Visualization(s) for questions 3, 4, 5 and 6. **All visualizations should be in png format.** Visualizations should be based on the question number and number of visualizations you have for a given question. For example, if your submission for *q4* had 3 visualizations, they should be named *q4-1.png*, *q4-2.png* and *q4-3.png*.
- You should use a Jupyter Notebook to invoke the functions in coc131_cw.py. Upload this notebook as part of your submission.

6 Notes on Expectations

You will be marked according to your overall achievement and the Assessment Rubric that will be provided to you separately from this document. However, below follows a qualitative description of relevant items that may help you understand the general level of expectation associated with this piece of coursework.

1. **Technical mastery of Python and Sklearn** Your programs should show mastery of what you have been taught.
2. **Technical mastery of module curriculum** This include illustration and utilization of items like CV, proper training/testing protocols and visualizations where appropriate.
    - For visualization, you are limited to the use of matplotlib only.
3. **Design** Your programs should be well structured for the task in hand so that it is as easy as possible for:
    - A user to use the program for any likely purpose. **A key strategy is to write you own functions and reuse them across your implementation.**
    - A programmer to understand the code structure and be able to develop it further,
    - A programmer to be able to re-use as much as possible of the code in a related application.
4. **Clarity and Self-Documentation** Given the structure of your programs, they should be as easy to read and understand as possible. Lay your code out so that it can be listed sensibly on a variety of devices: avoid having any lines longer than 80 characters as these may wrap. Sensible names should be chosen for all variables, methods etc. Documentation strings and/or markdown cells should be included for each:
    - **Program** You can provide any special instructions or warnings to the user (or assessor!) or draw attention to any aspects of the program that you are particularly proud of (please don't waste time by writing an excessive amount; max 200 words).

- **Function** State what each function does and explain the roles of its parameters.
  In addition, you should include occasional comments in your code; these may be (a) to introduce a new section in the code, or (b) to explain something that is not obvious. Bear in mind that pointless comments make your code harder to read, not easier.

5. Visualizations should be easy to read and serve its purpose. The questions clearly state what the visualizations should illustrate.

6. The documentation for parameters a function can accept is not perfect **by design**. The documentation for the function itself clearly specifies what it should do but you can exploit the available parameters to do more than that. Thus, under-specification in the description of function parameters is intentional so you can write reusable and multi-functional code.

7. You can expand the function documentation (both function itself and its parameters) to help me better understand your implementation. When you add documentation, keep the following things in mind

   - You documentation should start in a new line following documentation that was already present in the code shared by me. The documentation should start with your initials followed by colon, for example, I would write "SD: ...".
   - You can not change function definition i.e. you can't add/remove parameters and what the function returns. You can't make an optional parameter non-optional and vice-versa.

8. **Read the function documentation carefully.** Some functions have additional requirements on what is returned. Your functions will be tested against these requirements.

9. **No pandas please**. Only numpy should be used for array manipulation.