Python for Applied Machine Learning

Assignment Specification

There are two parts to this assignment. In the first part, you will develop a color-based segmentation system for sweet pepper and in the second part you will develop a classification system to differentiate sweet pepper from background. In achieving this you will have written code that has been:

- divided into modules and be able to import them,
- provide classes for your different solutions,
- produce precision-recall curves for the segmentation and classification systems that you obtain to support your decisions about which method to use, and
- implement machine learning algorithms.

You are encouraged to seek advice to discuss different design choices. However, all design choices are for you, the student, to make. This is because **during the exam interview you will be asked to present and discuss the reasons for your design choices**.

The data to use for this assignment can be found here:

https://uni-bonn.sciebo.de/s/wWisbFKBDnY5QFZ

Part I Segmentation of sweet pepper based on color

In this part of the assignment you will design two sweet pepper **segmentation** systems based on a color (RGB) input. You will be given labelled data to learn from that consists of "red sweet pepper" as class 1 and "not red sweet pepper" as class 0. It will also be divided into separate folders with a defined *training*, *validation* and *evaluation* set.

You will need to develop two systems and you will need to implement one of these yourself; the other classifier can make use of the sklearn library. For the implementation of your own classifier you must implement both the training algorithm (fit) and the evaluation algorithm (predict). The solutions should be coded in an object-oriented manner.

For this part you will need to provide two python scripts, one for each solution, that will process an image given at the command line so that it can be executed at the command line. Something like the following would be acceptable:

- python3 sweet_pepper_segementation_solution1.py image.png
- python3 sweet_pepper_segementation_solution2.py image.png

Your solution will run your sweet pepper segmentation algorithm and count the number of pixels which contain red sweet pepper. It will then provide the percentage of vegetation pixels in the image and print this to the screen. Your solutions will also print out the time taken to process the image.

When you submit your solutions, you will also need to provide, for both classifiers, the precision-recall curves to describe their performance on the *validation* and *evaluation* sets. It is suggested that you use color features from by transforming the RGB data to a different color space (e.g. YCbCr, Lab, etc.).

There is an extension to this part of the assignment. *For the extension you will implement one system that can perform multi-class segmentation into "yellow sweet pepper" as class 2, "red sweet pepper" as class 1 and "not red/yellow sweet pepper" as class 0. For this solution it will process an image given at the command line so that it can be executed at the command line. Something like the following would be acceptable:*

- *python3 sweet_pepper_segementation_solution3.py image.png*

Your solution will print to the screen which class if believes the image belongs to and the time taken to process the image.

Below is a guide for the key objectives that should be met by this part of the assignment.

General Objective	**Notes**
Implement your own classifier	Using the training set
Implement a classifier using sklearn	Using the training set
Produce a precision recall curve for each classifier	On the validation and evaluation sets
Provide the time taken to process an image	Preferably using a decorator or context manager
Provide the code in a modular way for the two classifiers	Separate code into appropriate modules
Provide the code in an object-oriented framework	Separate the code so that we see code re-use where possible
Count the percentage of vegetation in an image	A percentage
Extension**:** Implement the 3-class segmentation system	

Part II Classification of sweet pepper and background

In this part of the assignment you will design a two-class **classification** system to decide if an image is either a "sweet pepper" or "not sweet pepper". You will be given images to learn from that consists of "sweet pepper" and "not sweet pepper". You will need to load this data and assign them the correct labels and use the defined training, validation and evaluation set.

You will need to develop two systems and you will need to implement one of these yourself; the other system can make use of the sklearn library. For the implementation of your own system:

- it can be your own feature extraction algorithm (e.g. bag of visual words) or your own classifier,
- it cannot be the same classifier that you implemented in Part I.

The solutions should be coded in an object-oriented manner.

For this part you will need to provide two python scripts, one for each solution, that will process an image given at the command line so that it can be executed at the command line. Something like the following would be acceptable:

- python3 classification_solution1.py image.png
- python3 classification_solution2.py image.png

Your solution will print to the screen which class it believes the image belongs to and the time taken to process the image.

When you submit your solutions you will also need to provide, for both systems, the precision-recall curves to describe their performance on the validation and evaluation sets.

Below is a guide for the key objectives that should be met by this part of the assignment.

General Objective	**Notes**
Implement your own classifier/feature extractor	Using the training set
Implement a classifier using sklearn	Using the training set
Produce a precision recall curve for each classifier	On the validation and evaluation sets
Provide the time taken to process an image	Preferably using a decorator or context manager
Provide the code in a modular way for the two classifiers	Separate code into appropriate modules
Provide the code in an object-oriented framework	Separate the code so that we see code re-use where possible