Environment setup for running the challenge.py

1) Create a folder ~C://SpectraPlasmonic//challenge1/ /and place all the files in the same folder
2) Change the working directory of your development environment (Spyder) to the folder, created in step 1.
3) Open test.py file
4) Initialize the parameters. Set path2model to path to the trained model for example ~path//C://SpectraPlasmonic//challenge1//model
5) If the testing_data has the same format as the training_data sent, use the path2data parameter and read the file using the data_loader.py otherwise initialize testing_data, and testing_labels. For simplicity the labels are changed from 'A', 'B','C','D' to 0, 1, 2, and 3
6) After setting the parameter and initialization run test.py to test the trained model
7) You must see the confusion matrix and the ROC in the console

Module description

The following modules are provided:

Main.py: The main module used for understanding the data and training the model, initialize the parameter Path2data for running the code and see the step-by-step training process.

data_loader: loads the data, and returns the spectra, class labels from the data dictionary. For simplicity the labels are changed from 'A', 'B','C','D' to 0, 1, 2, and 3. This module also normalizes the spectra to the peak at 601 wavenumber to reduce the effect of concentration.

Feature_extractor.py: Applied principal component analysis (PCA) followed by linear discriminant analysis (LDA) for extracting features from the spectral data. PCA was performed using 5 components (explaining more than 90% of the dataset). Then an LDA is performed to further reduce the featured to only three (one less than the number of classes).

Since the number of samples are small only 73, the extracted PCA-LDA features are added to the spectra itself to increase the accuracy of the model.

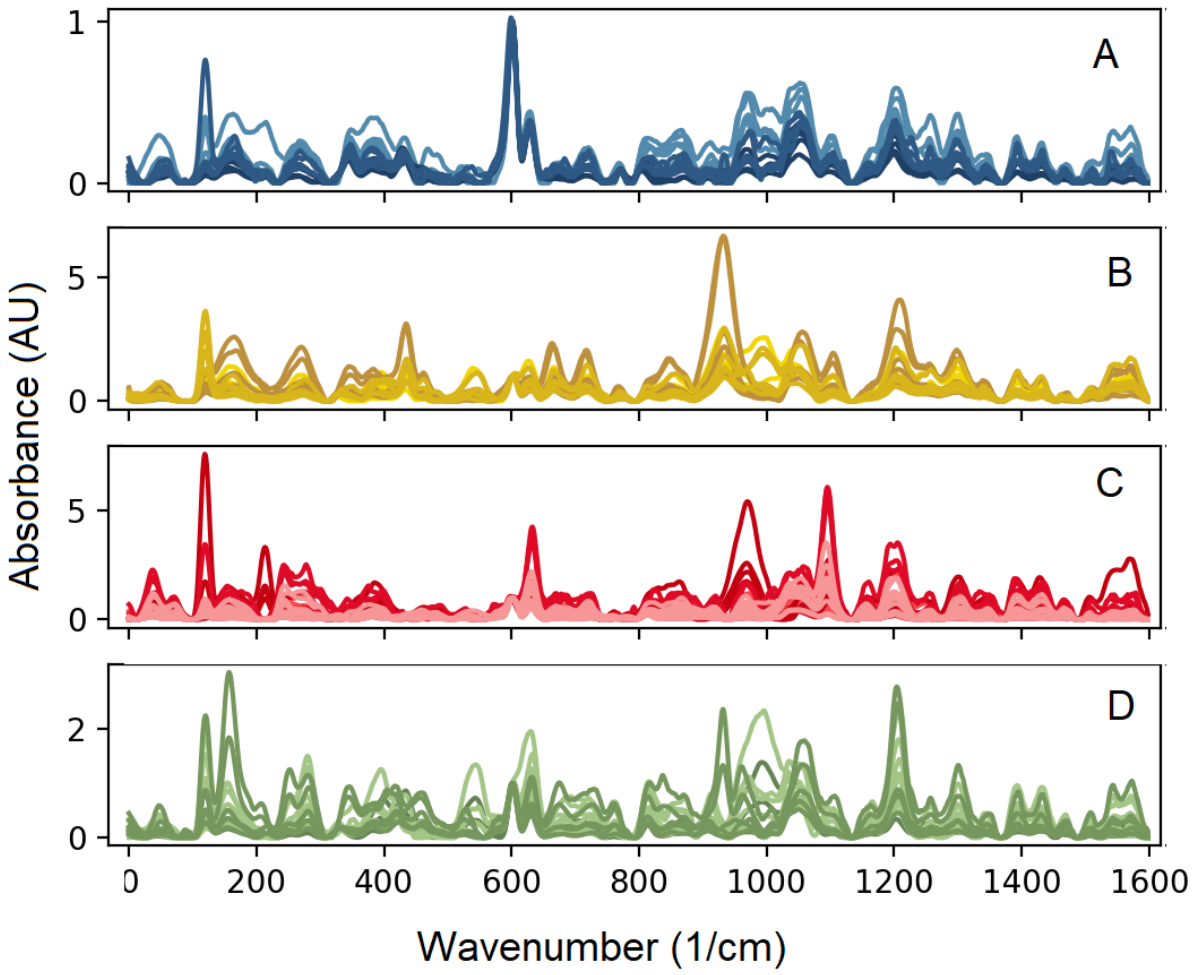
To increase the performance of the model I also added the ration of the peak at 601 wavenumbers to 628 as another feature. Ending up with total of 1605 features.

Performance.py: Evaluate the performance of the model using confusion matrixes and the ROC curves

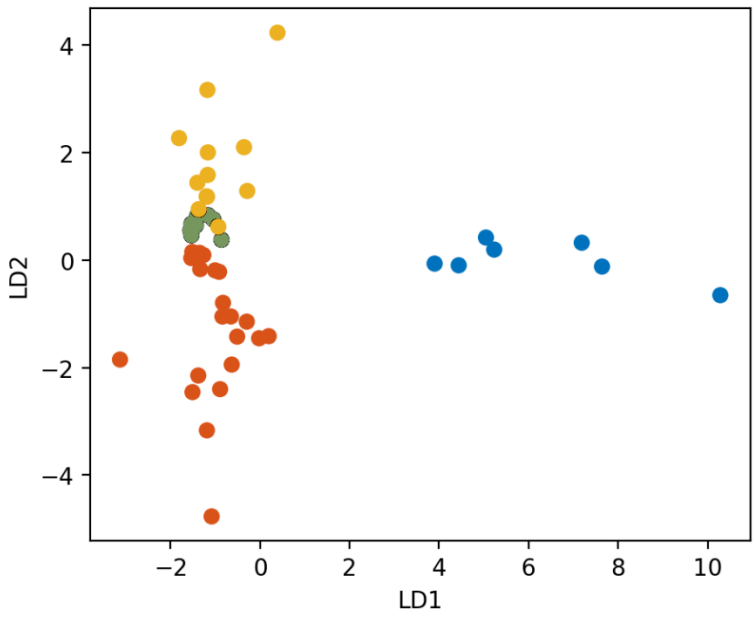
Test.py: Initialize the path2data and path2model and tests the performance

Data Visualization & Problem Understanding

To understand the spectra of each class and the effect of concentration spectra of each class is plotted. The intensity of each color is representative of the concentration of the spectra.



The spectra are clean with no noise. Projecting the spectra to a mathematical coordinate where the maximum variation between the spectra is selected, is the best practice for feature extraction here, so I did a principal component analysis (PCA). Since the data was still mixed in the PCA domin, I performed a LDA as well



Projecting the data into LDA domains shows distinct clusters they are not still perfectly separated, so I trained a logistic regression model to classify the spectra using the only 3 features extracted.

Since the number of samples were small, I had to add the LD features to the 1601 points of spectra to train the logistic regression.

Model Building

I separated the data into two groups for training and testing, I trained the model using 66% of the data and tested it using the other 33% of the data. The further tuned the data using the later 33% so that the model is ready to be tested using your independent test set.

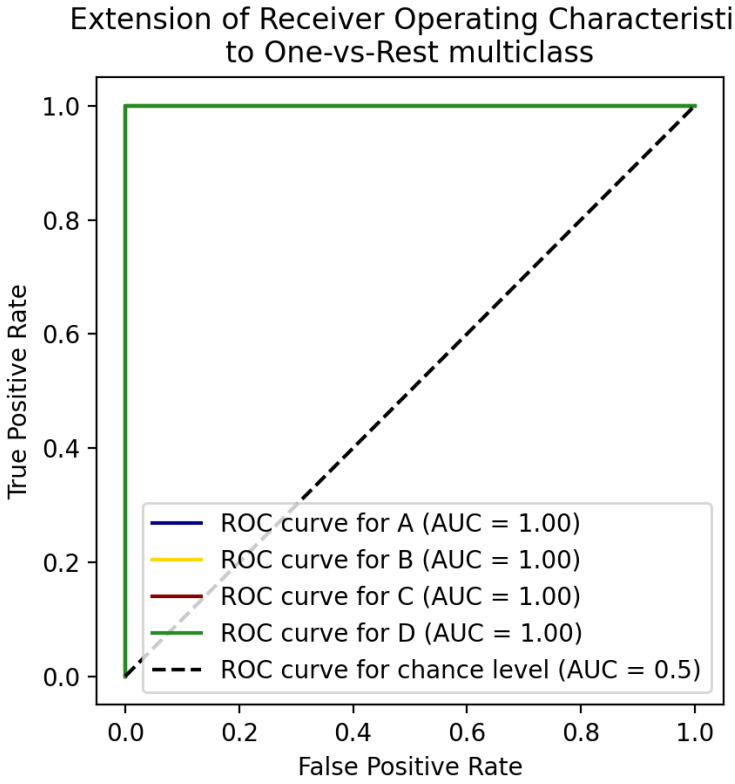
Since the labels were strings A,B,C and D, I changed the label classed to 0 to 3 numbers for simplicity (this label change becomes important later for checking the performance of the model. If I want to use the sklearn commands, the labels must be 0,1,2,3).

The logistic regression model is trained using 1605 features (spectra + LDA extracted features + ratio of 601 to 628 band))

The model is initialized randomly here, so depending on the initialization the model may not converge to the right answer. Since the training data is small and the model is simple, I repeated the training until I got the correct trained model. Finally, the results are evaluated using a confusion matrix and the ROC (one-versus-all) curves.

Because of the distinct variations in the spectra, the model has 100% accuracy on the test set (33% of the received data). The confusion matrix and the ROC curves of the test set are shown below:

	A	B	C	C
A	100%			
B		100%		
C			100%	
D				100%



Model's performance on unseen data

The model performance on unseen data, can be tested using the test.py file. The performance can be increased by feature engineering and increasing the number of features, using peak ratios, areas under the peak, and etc.