**Environment setup for running the challenge.py**

1. Create a folder ~path/P&POptica/interview/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 model for example ~path//P&POptica//interview//challenge1//model
5. If the testing\_data is a .pkl file with the same format as the training\_data.pkl, use the path2data parameter and read the file using the data\_loader otherwise initialize testing\_data, and testing\_labels
6. Run the program
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. It also returns the target labels from the labels.txt file

**MyPCA.py:** Performs PCA analysis on the spectra and returns features

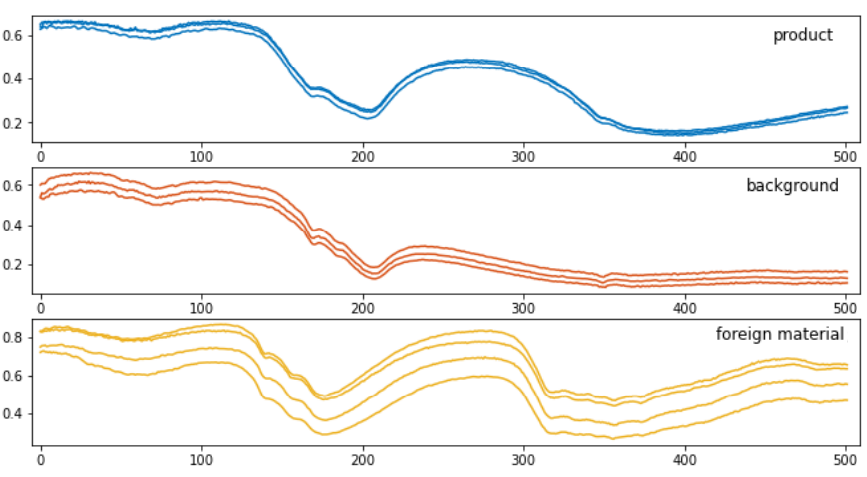
**Performance.py:** Evaluate the performance of the model using confusion matrixes and the ROC curves

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

**Challenge.py**: contains the SpecPredict class

**Data Visualization & Problem Understanding**

1. To understand the spectra of each class, random spectra from each class are plotted



Each class has a distinct spectrum, and the peaks are at very different positions, meaning the spectra of the material are very different. 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).

1. The projected data to the PC coordinate is shown belowChart, scatter chart, bubble chart

   Description automatically generated

Plotting the spectra using only the first two PC components, results in three distinct clusters another sign that each class has a very distinct chemistry. This means that the spectral features can be extracted using PCA, and a simple supervised learning algorithm like linear discriminant analysis can classify the spectra with accurate results.

For further chemometric analysis and reducing the spectral information, we could look at the PC loadings and find out the important chemistry. The first four PC components include more than 99% of the data, so we selected only the first three components.

**Model Building**

In the mind of a spectroscopist, the first thing to try is applying the features to a linear discriminant analysis (LDA) to train an LDA model. I decided to go with a logistic regression model since it does the same thing and is usually the first model that a software engineer would try.

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.

Since the labels were 0,1, and 5, I change the label for class 5 to 2 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).

Then I applied PCA to the spectra to select features. The logistic regression model is trained using the 4 selected PC (features), using 5-fold cross-validation. 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. The confusion matrix and the ROC curves of the test set are shown below:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Product | Background | Foreign Material |
| Product |  |  |  |
| Background |  |  |  |
| Foreign Material |  |  |  |

**Speed optimization**

The training speed may be different from one computer to another, or even using the same computer at different times. Speed optimization becomes important for training, when the training samples are huge. The classification speed becomes important for real-time models and anomaly detection.

We can do the following for reducing the training time:

* Changing your optimization function (solver)
* Using different hyperparameter optimization techniques (grid search, random search, early stopping)
* Parallelize or distribute the training with [joblib](https://joblib.readthedocs.io/en/latest/) and [Ray](https://docs.ray.io/en/master/index.html)

We can do the following for reducing the classification time:

* Reducing the features
* Reducing the size of the model

**Model's performance on unseen data**

The model performance on unseen data is tested using 33% of the data, by plotting the ROCs and confusion matrixes