Projekt Mikro- und Feingeräte
Intelligent Spectral Analysis

Final Report

15.04.2022

Authors:

Adnene Boumessouer
Institution: TU Berlin
a.boumessouer@tu-berlin.de
Matr. number: 374599

Anis Kadri
Institution: TU Berlin
anis.kadri@tu-berlin.de
Matr. number: 365196

Supervised by:

Anja Marckwardt
marckwardt@mfg.tu-berlin.de

Jessica Weber
weber@mfg.tu-berlin.de

Contents

List of Figures

List of Tables

1 Introduction

Raman spectroscopy has been shown to be a powerful analytical technique in the study of biological materials, and it allows rapid, noninvasive and high spatial resolution acquisition of biochemical and structural information through the generation of spectra. Over the past several decades, it has become an important technique in a wide range of fields, including geology and mineralogy, pharmaceuticals, biology, medicine and archeology. Its wide applicability comes from the fact that each band in a Raman spectrum represents inter action of incident light with a vibrational mode in the molecule; it is therefore highly material specific and can be used for identification and structural characterization of unknown samples [1].

1.1 Raman Spectroscopy

In Raman spectroscopy, a physical phenomenon called Raman scattering is utilized to examine materials, where the photons striking the material are inelastically scattered. In this process, energy is transferred either from the photon to the molecule (Stokes Raman scattering) or from the molecule to the photon (anti-Stokes Raman scattering) by transporting the photon to a different vibrational level. This energy transfer leads to a wavelength shift of the scattered photon, which in turn depends on the material's characteristics. This results in characteristic signature from which conclusions can be drawn about the material composition [2].

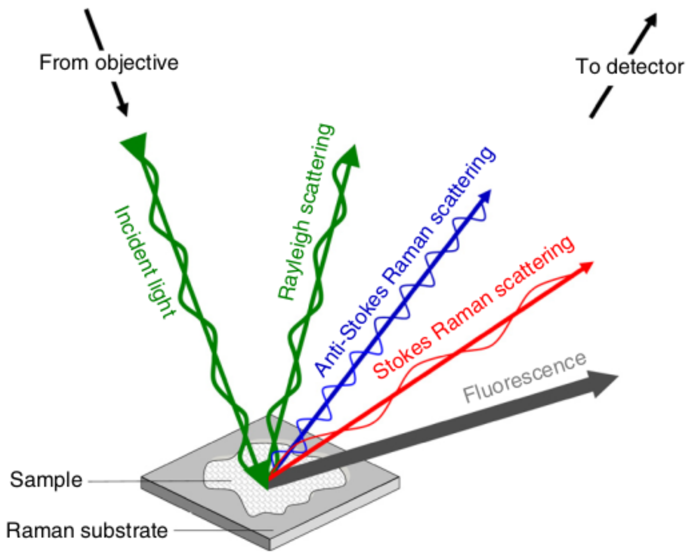


Figure 1: Schematic identifying light scattering after laser exposure on a sample surface.

The device implementing the Raman effect is known as the Raman spectrometer (Figure 2). In a Raman spectrometer, a sample is illuminated with a monochromatic light. Most of the light passes through the sample, a very small portion is reflected by the sample in all spatial directions (elastic scattering). An even much smaller fraction, on the other hand, is scattered inelastically (Raman scattering). When the photons and the molecules interact, the energy levels of the particles change and thus the wavelength of the reflected light also changes. The emitted energy passes through a spectrograph, which divides it into a light spectrum. A charge coupled device (CCD) detector analyzes the spectrum and converts it to a "read out" on a computer. The peaks in the Raman spectrum correlate correlate with the different vibrations in the molecular boundaries and each substance has its own unique spectral fingerprint based on these vibrations[2].

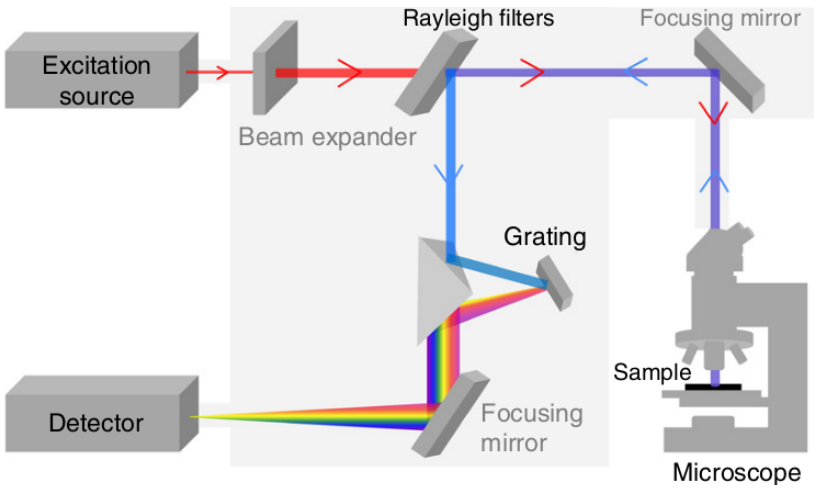


Figure 2: Raman spectroscopic microscope system.

1.2 Objective and Approach

This project proposes an ML-based approach to determine the concentration of a substance, the Pentanediol, dissolved in water. Raman spectroscopy will be used to record light spectra for different concentrations under different settings. The aim is to create a dataset of spectra (described in section 2) that are mapped to known concentration. In order to learn this mapping, a machine learning based approach is developed.

This approach is subdivided into two parts: preprocessing and training. Preprocessing includes trimming and smoothing the spectra (section 2.2), extracting features (section 3.1) in addition to conducting a selection of the best features (section 3.2). The training stage involves conducting a preliminary performance comparison between different regression models using Cross-Validation in order to select a promising model (sections 4.1 and 4.3) and then fine-tuning it with Grid Search (section 4.4). Finally, an evaluation is conducted (section 5), from which conclusions and improvement suggestions are drawn (section 6).

Implementation

The code for this project is available as Jupyter notebooks at:
`https://github.com/AdneneBoumessouer/intelligent_spectral_analysis`
Please read carefully the instructions in the readme file to be able to execute the code and reproduce the experiments.

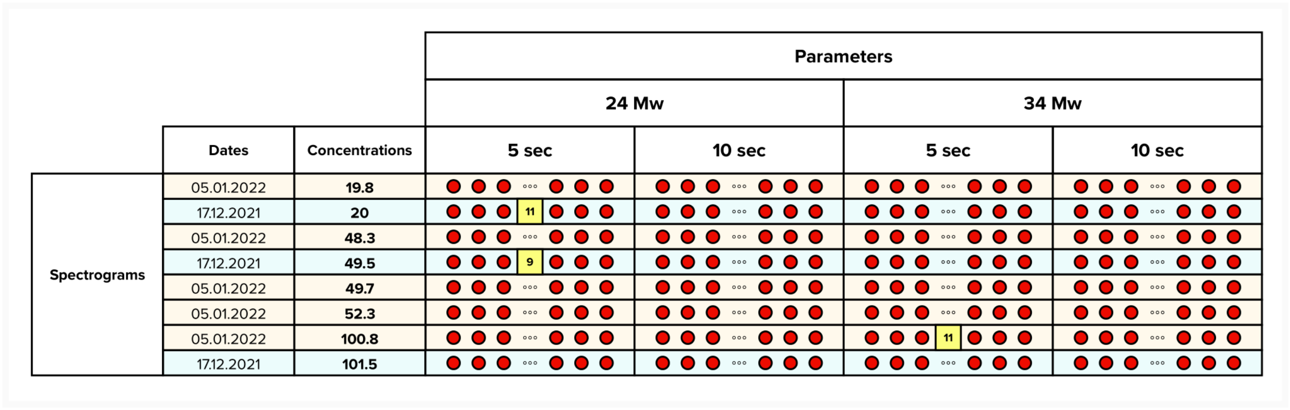
The dataset of spectra used in this project can be downloaded using the following link:
`https://drive.google.com/file/d/16NH1MDH6PU-ZU7xEN8VZw7G6KYfUE8gg/view?usp=sharing`

2 Dataset

2.1 Acquisition Setup & Description

The experiment in which the measurement data are recorded is set up as follows: 8 different samples of Pentanediol dissolved in water are prepared with concentrations 19.8, 20, 48.3, 49.5, 49.7, 52.3, 100.8 and 101.5 mg/mL. In this process, the duration of each recording, further referred to as integration time, and the intensity of the laser light, indirectly the laser power, can take 2 values, forming a total of 4 possible configurations. To determine the laser power precisely, a sensor is also integrated into the experimental setup.

The table in the Figure 3 illustrates the structure of the available measurements, with each individual recording represented by a red dot. Each sample is measured 10 times pro configuration with some exceptions of 9 or 11 measurements each (e.g solutions corresponding to the concentration 20 mg/mL with 5s - 24mV or 49.5 mg/mL with 5s - 24mV). The left column indicates the date on which each solution was prepared, which is considered a possible interference cause in the validation section 2.2.2.



			Parameters			
			24 Mw		34 Mw	
	Dates	Concentrations	5 sec	10 sec	5 sec	10 sec
Spectrograms	05.01.2022	19.8	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●
	17.12.2021	20	● ● ● 11 ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●
	05.01.2022	48.3	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●
	17.12.2021	49.5	● ● ● 9 ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●
	05.01.2022	49.7	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●
	05.01.2022	52.3	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●
	05.01.2022	100.8	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● 11 ● ● ●	● ● ● ◦◦◦ ● ● ●
	17.12.2021	101.5	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●	● ● ● ◦◦◦ ● ● ●

Figure 3: Overview dataset.

The first step is to visualise and examine the data to check the quality of the signals (noise content, extractable information) and validate their consistency.

2.2 Exploration & Validation

2.2.1 Exploration: Trimming and Smoothing

Raman spectras are represented in the x axis in wavenumber [cm^{-1}] instead of wavelength [nm]. The wavenumber is an energy related value and can be obtained by a Raman shift, described by the following formula:

$$\Delta\tilde{\nu} = \frac{1}{\lambda_0} - \frac{1}{\lambda} \tag{1}$$

Where λ_0 is the reference wavelength used by the measurement instrument (522nm in this case). Figure 4 shows the raw data after performing a Raman shift. Since there are 4 possible configurations, the data is presented in 4 graphs accordingly, where each graph represents a particular acquisition setting (i.e combination of integration time and laser power). In each graph, spectra with concentrations in the vicinity of 20 mg/mL are plotted in green, those in the vicinity of 50 mg/mL in blue and those in the vicinity of 100 mg/mL in red.

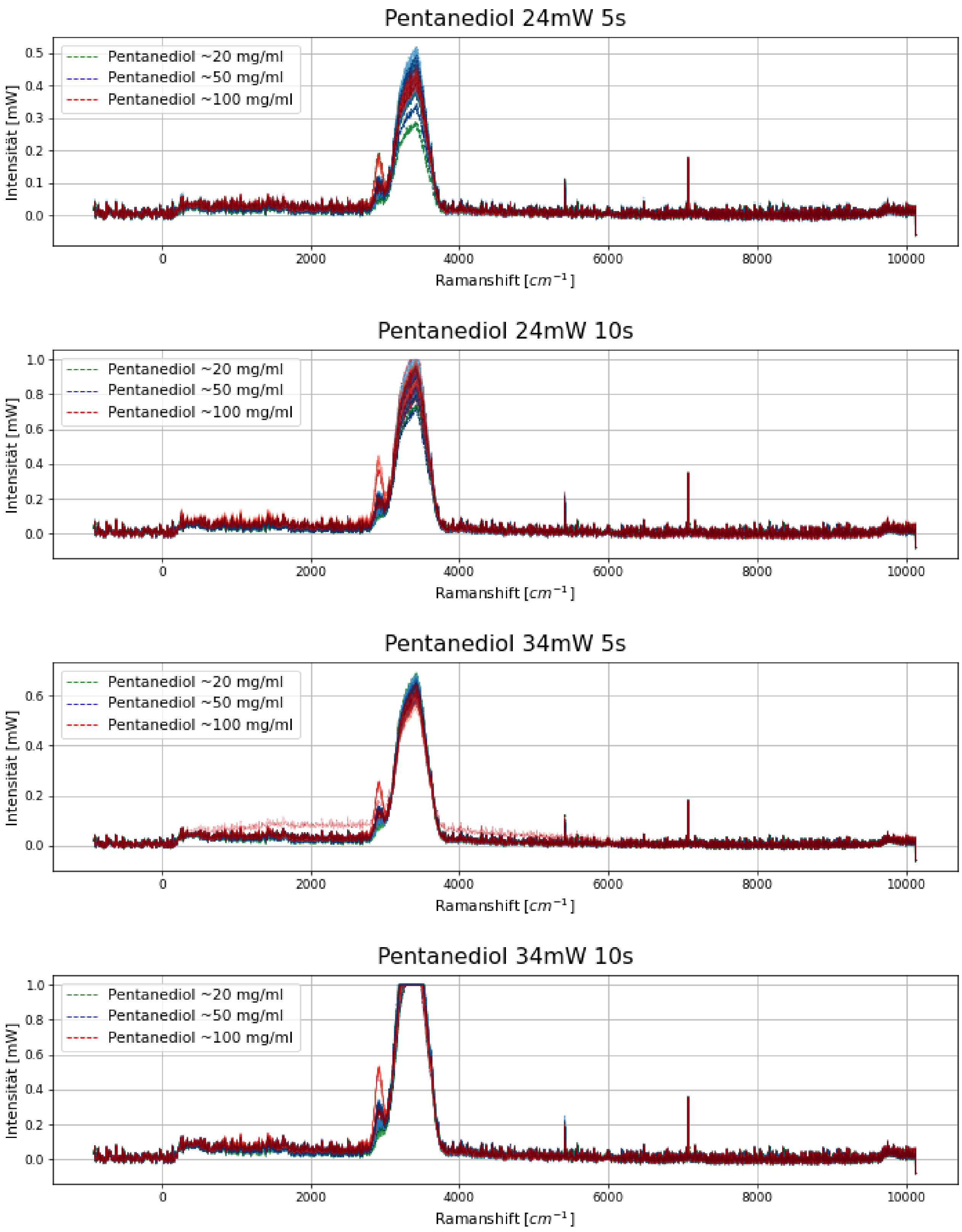


Figure 4: Raw Raman spectra in 4 settings. In each graph, spectra are grouped by concentration vicinity.

There is a clear saturation in the 34 mW ,10s configuration in the second peak between 3000 cm^{-1} and 3750cm^{-1}, which is to be interpreted as a distortion of the measurement and leads to an unusable section. Because of this corruption, it is imperative to trim the data and use either the segment before or after saturation. The plots also show that most of the information tends to be stored in the first interval, where a distinct spike is observed, while the second interval (after the corrupted section) is essentially constant when noise is disregarded. This is also consistent with the characteristic interval of pentanediol where the specific energy peaks for this substance occur, as shown in Figure 5, which is a typical Raman spectrum of pentanediol.

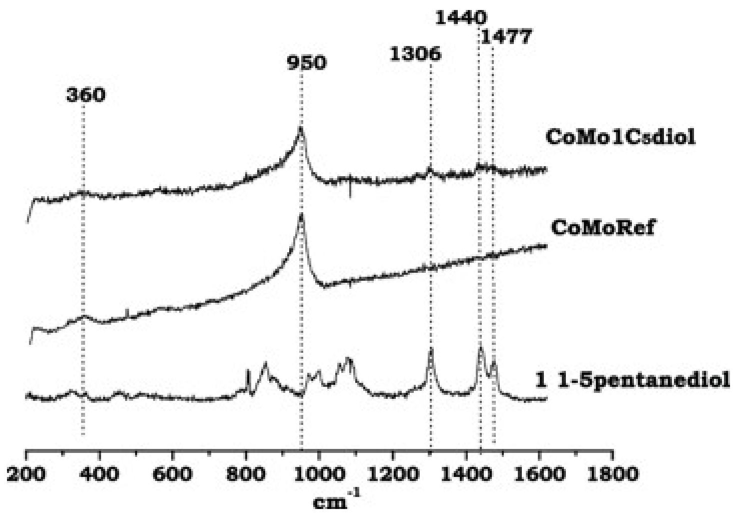


Figure 5: Typical Raman spectra of Pentanediol.

On this basis, the spectra are trimmed up to 3000 cm^{-1} in order to extract the region of interest. In addition, they are smoothed using a Savitzky-Golay filter with polynomial order 5 and a window length of 21 to reduce the noise. Figure 6 shows an example graph after trimming and smoothing.

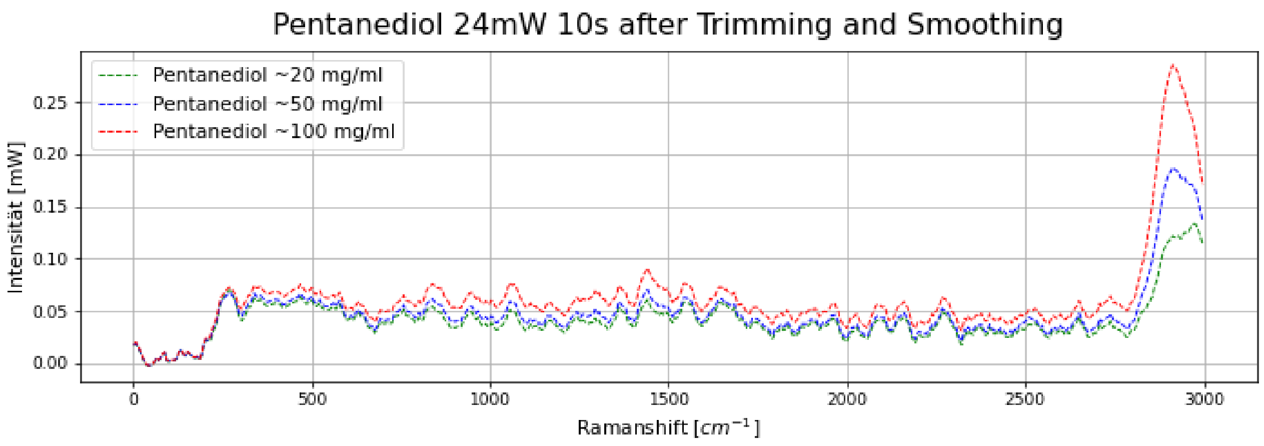


Figure 6: Raman spectra of Pentanediol in our dataset.

Even after cleaning and smoothing, the typical pentanediol peaks seen in Figure 5 are not apparent in the measured signals

2.2.2 Validation: Consistency with respect to date and concentration

Before further pre-processing, validation of the data is required to check consistency with respect to two main factors: The date of preparation and the concentration of the pentanediol. The pentanediol solutions were prepared on two different days (17.12.2021 and 05.01.2022), so it is reasonable to investigate any noise or influence of the date on the measurement. For this purpose, plots were made with the same configuration (laser power and integration time), the same concentration group, but different dates. The expectation here is to find (almost) overlapping signals, since the only difference is the date of preparation. Figure 7 shows one of these Graphs.

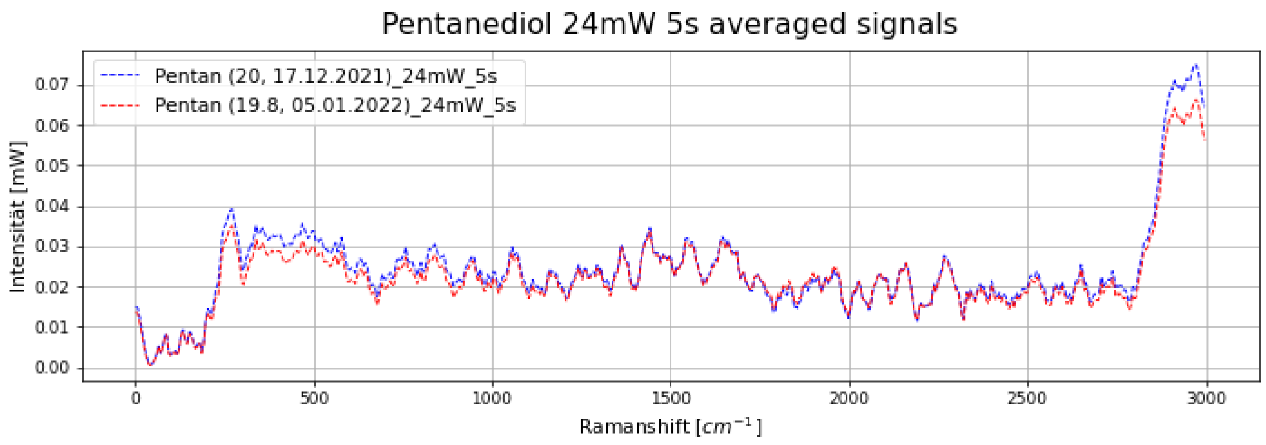


Figure 7: Averaged Raman spectra of Pentanediol for concentrations 20 mg/mL and 19.8 mg/mL under the same setting (24 mW, 5 seconds) for different dates.

Figure 7 shows a good overlap in the first part of the signal (section before 2800cm^{-1}). The second part contains some differences that led to dividing the signal into these two parts in further preprocessing steps. This observation holds also for other concentration groups.

After checking the measurement with respect to the preparation date. Both dates were merged and the spectra were grouped by concentration categories only, as shown in Figure 8.

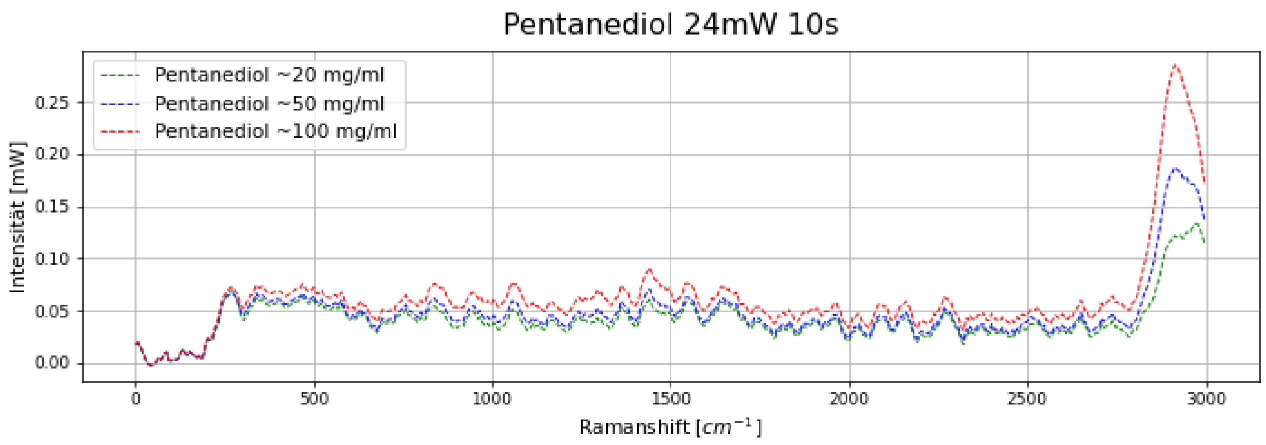


Figure 8: Raman spectra averaged across concentration groups under the setting 24mW, 10 sec.

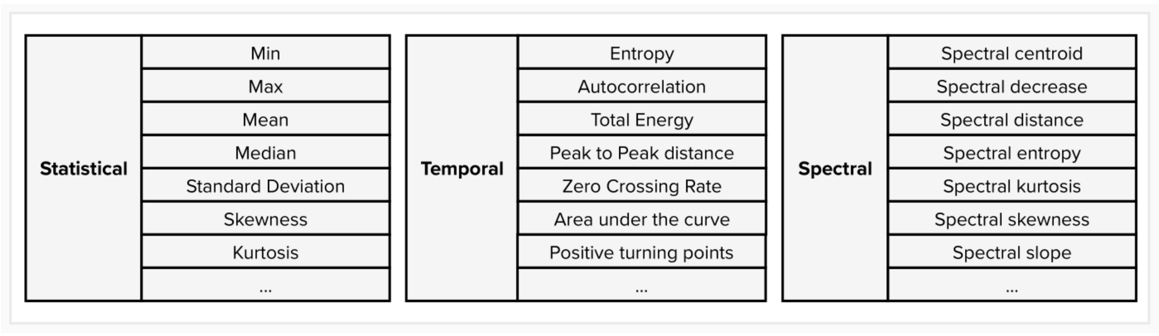
In these graphs, a gradation of intensity (y-values) can be seen, reflecting the increase in concentration. This means that the information of interest can be extracted from the available data.

3 Preprocessing

This section introduces the preprocessing steps needed to transform the smoothed and trimmed spectra into a set of relevant features that will be used for predicting the concentration of pentanediol. After examining the data in section 2.2, it was concluded that features such as peak height and peak width which are traditionally used in Raman spectrosopy to determine the concentration of a substance could not be identified due to the quality of the measurements. In order to overcome this limitation, it seems reasonable not to restrict ourselves to these two features and instead extract other descriptive features from the spectra.

3.1 Feature Extraction with TSFEL

As a preliminary step in conventional machine learning pipelines, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps. To facilitate this process, we resort to the TSFEL python library [3]. This open source library is originally intended to perform feature extraction on time series. Although our dataset contains spectra and not timeseries, we make use of this library, since the spectra can also be considered as a ordered sequence of values. TSFEL simplifies the task by automatically extracting over 270 features, which are divided into three areas: statistical, temporal and spectral features. Figure 9 illustrates a sample of these features. A comprehensive listing and description of the extracted features can be found here:
`https://github.com/AdneneBoumessouer/intelligent_spectral_analysis/blob/main/data/preprocessed/features.csv`



Statistical	Temporal	Spectral
Min	Entropy	Spectral centroid
Max	Autocorrelation	Spectral decrease
Mean	Total Energy	Spectral distance
Median	Peak to Peak distance	Spectral entropy
Standard Deviation	Zero Crossing Rate	Spectral kurtosis
Skewness	Area under the curve	Spectral skewness
Kurtosis	Positive turning points	Spectral slope
...	...	...

Figure 9: Overview of TSFEL library features

When validating the data set (section 2.2.2), a distinction was made between two intervals. A clear influence of the preparation date was observed in the interval between 2800 cm^{-1} and 3000 cm^{-1}, while this factor did not seem to have any influence on the first interval (before 2800 cm^{-1}). Therefore, all data (training and test sets) are split at 2800 cm^{-1} and feature extraction was performed separately for both intervals, which are then concatenated together with the "laser power" and "integration time" to form a comprehensive feature matrix. Figure 10 shows the feature extraction steps from dividing the signals into two intervals to extracting the features in each interval and combining all features for each signal.

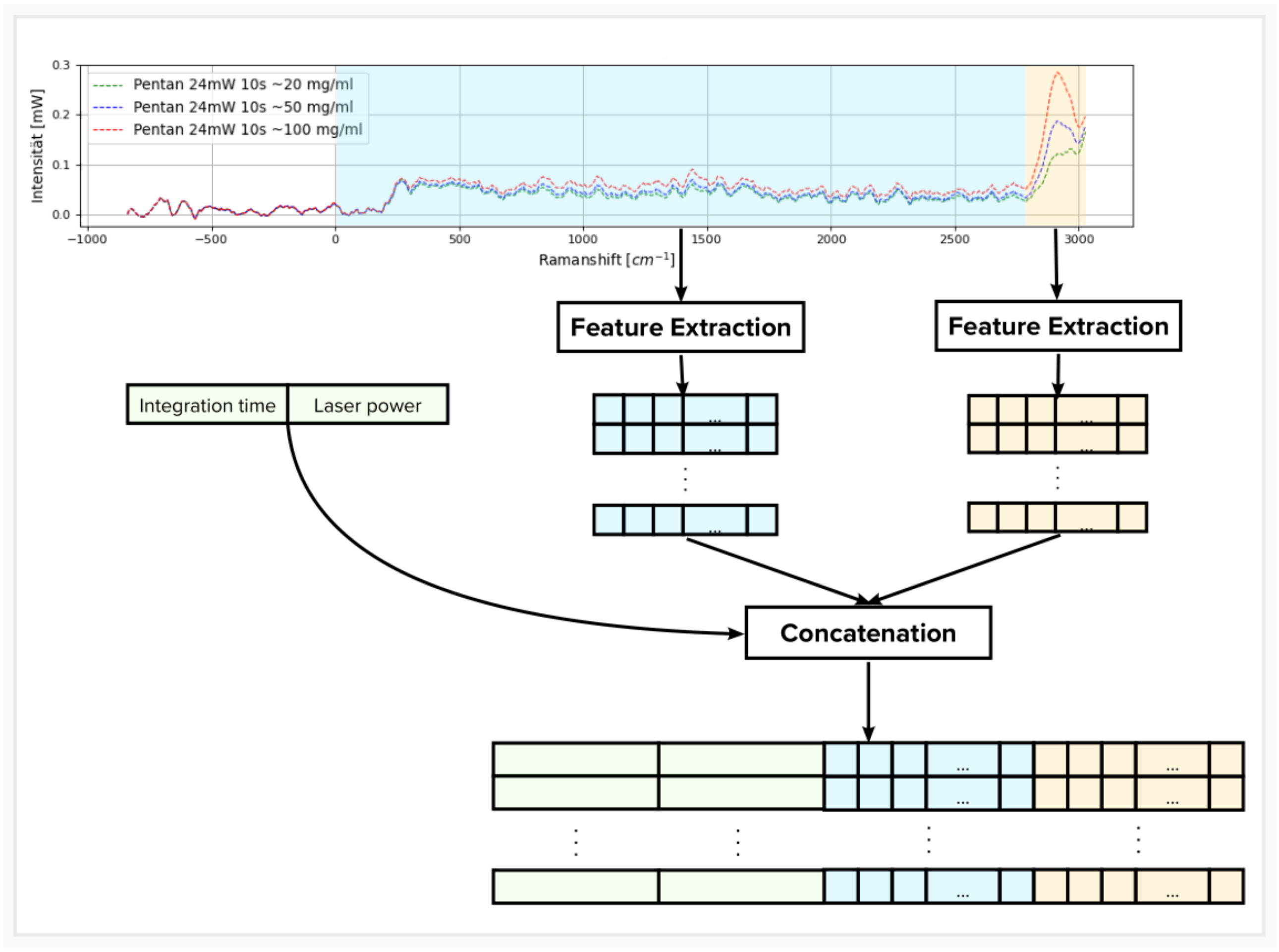


Figure 10: Features extraction Process.

The result is a feature matrix where x rows stand for the different signals and y columns for the features.

A large number of features combined with a limited number of samples is recipe for overfitting, a common problem in machine learning where the model ”memorises” the training set instead of learning to make predictions. This can be recognised by the model performing well on the training data but poorly on unseen data. To avoid overfitting, it is recommended to perform feature selection and only use features that are relevant for the purpose, determine the concentration of pentanediol in this case. This is considered in detail in the next section.

3.2 Feature Selection

Not all extractable features are useful, some of them may be misleading or irrelevant. Determining a subset of the original features is called feature selection. The selected features are expected to contain the relevant information from the input data so that the desired task can be performed using this reduced representation instead of the full output data as shown in Figure 11.

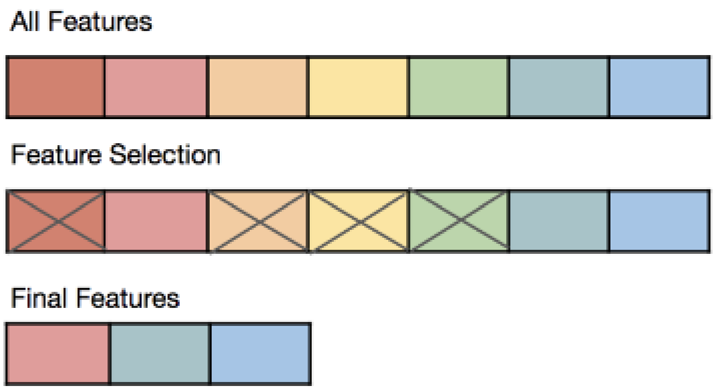


Figure 11: Feature Selection.

To achieve a feature selection that is well suited for the given task, the following methods are used

3.2.1 Removal of highly correlated features

If two or more features contain the same information (e.g. the same measurement in feet and metres or the repetitiveness of images represented as pixels), they are called redundant features and it is reasonable to keep only one of them and discard the others, as they would increase the size of the dataset without adding information and thus slow down the training process. In some cases, they may even act like noise in the prediction. Redundancy shows up in statistics as high correlation. A simple method to detect and discard such features is therefore to remove highly correlated features using a correlation threshold. This is done in a first step when selecting features.

3.2.2 Removal low variance features

A feature that has little or no relation to the label (in this case, concentration) is worthless for training (e.g., "foot size" is not relevant for predicting favourite drink). As for redundant features, insignificant features increase the size of the dataset without adding useful information, hence they are a good candidate to be discarded. These features are characterized by low variance because a change in their values is not associated with a change in the label values. Removing features with a variance of less than the fixed threshold is a good practice to get rid of them and is the second step of the selection process.

3.2.3 Feature Normalization

The features extracted with the TSFEL library are calculated automatically and have a wide range of magnitudes. Direct use of these features may result in some features with large magnitude overshadowing the others, as value changes in the latter would go unnoticed. Normalisation is therefore mandatory to overcome this issue and avoid the loss of information

3.2.4 Mutual Information Regression

Even after performing the previous steps, the number of remaining features is still large and requires further selection. Instead of removing the least relevant features based on a threshold, the idea in this step is to keep only the k-best features based on a ranking metric. This approach allows the selection of a certain number of features. The algorithm used in this step is based on the mutual information between each feature and the label. In probability and information theory, the mutual information (MI) of two random variables is a measure of the interdependence between the two variables. More precisely, it quantifies the "amount of information" obtained about one random variable by observing the other random variable. The concept of mutual information is closely related to that of the entropy of a random variable. Each characteristic is given a score by this algorithm, represented in Figure 12 resulting in a ranked list from which the 20 best are selected and the rest are discarded.

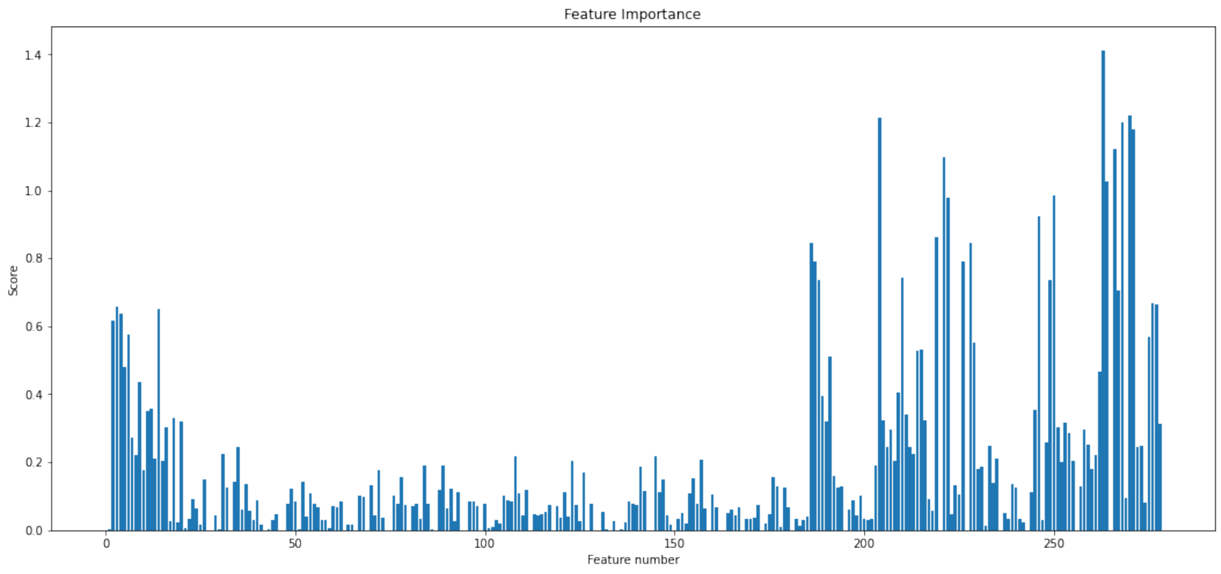


Figure 12: Features score from Mutual Information algorithm.

4 Training

4.1 Methodology

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect training score but would fail to predict anything useful on yet-unseen data. This situation is called overfitting. To avoid it, it is common practice when performing a supervised machine learning experiment to hold out part of the available data as a test set (Figure 13).

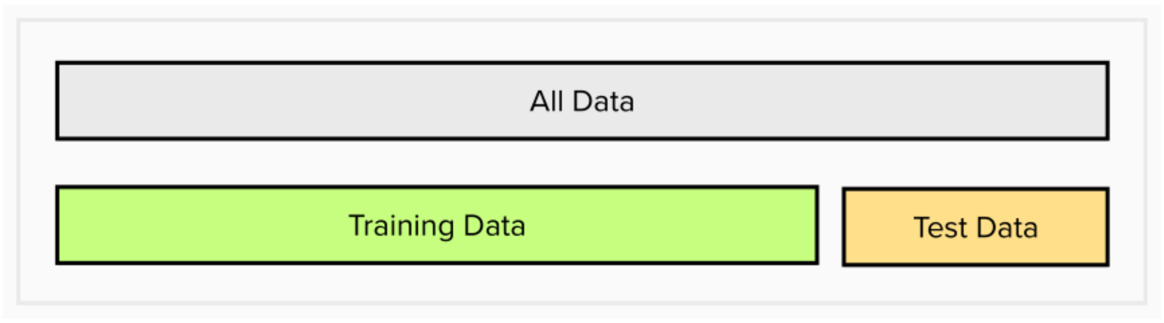


Figure 13: Train test split

When evaluating different settings (”hyperparameters”) for estimators, there is still a risk of overfitting on the test set because the parameters can be tweaked until the estimator performs optimally. This way, knowledge about the test set can ”leak” into the model and evaluation metrics no longer report on generalization performance.

This is because in realistic scenarios we rarely build a model just by training its parameters once. Instead, we are likely to explore many versions of a model through various modeling choices regarding network architecture, learning rates, data augmentation strategies, and other factors we will discuss in upcoming chapters. Many of these choices can be described as choices of hyperparameters. The word reflects that they are parameters about parameters, since they are the higher-level choices that govern the meaning of the weight parameters. The problem is that even though the ordinary training process is looking at only predictions on the training data when it learns values for the weight parameters, the same is not true of us. We, as modelers, are evaluating the model by looking at predictions on the test data when we decide to explore new hyperparameter values. So subsequent versions of the model are, indirectly, shaped by us having seen the test data. Just as the automatic training process is in danger of overfitting the training data, we are in danger of overfitting the test data through human trial and error and exploration.

To solve this problem, yet another part of the dataset can be held out as a so-called ”validation set”: training proceeds on the training set, after which evaluation is done on the validation set, and when the experiment seems to be successful, final evaluation can be done on the test set.

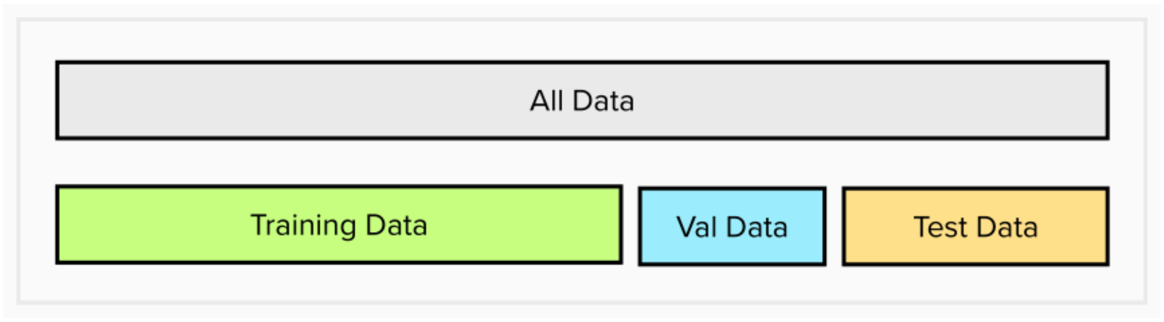


Figure 14: Train test validation split

However, by partitioning the available data into three sets (Figure 14), we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of (train, validation) sets.

A solution to this problem is a procedure called cross-validation (CV for short). A test set should still be held out for final evaluation, but the validation set is no longer needed when doing CV. In the basic approach, called k-fold CV, the training set is split into k smaller sets (other approaches are described below, but generally follow the same principles). The following procedure is followed for each of the k "folds":

- A model is trained using $k-1$ of the folds as training data
- The resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as mean squared error).

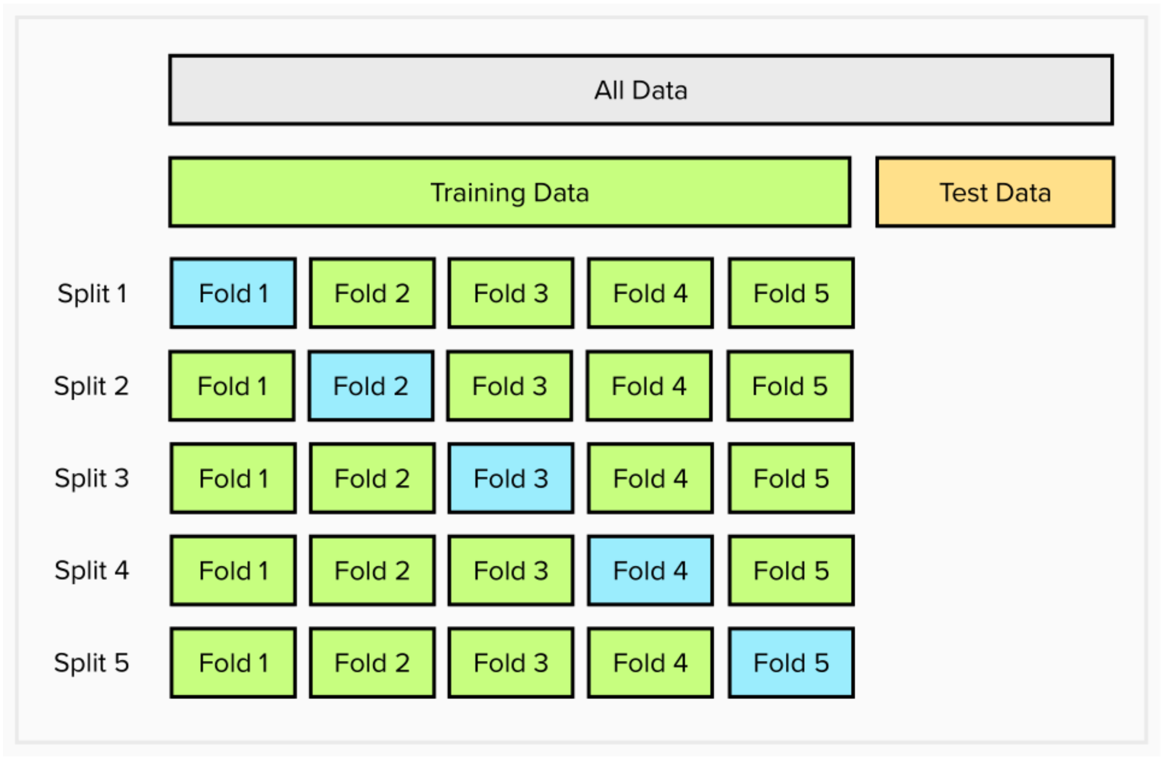


Figure 15: Cross Validation

The performance measures reported by k-fold cross-validation is then the average and the standard deviation of the values computed in the loop, which respectively represent an estimate of the model's performance and a measure of how precise this estimate is. This approach can be computationally expensive, but does not waste too much data (as is the case when fixing an arbitrary validation set), which is a major advantage in problems such as ours, where the number of samples is very small (321 samples in total).

Notice that while the cross-validation procedure is generally used for finetuning (i.e hyperparameter optimization), it can also be used to compare different models based on the average and standard deviation as will be described in section 4.3.

The figure below is a flow chart summarising the different steps for training, which are discussed in detail in subsequent section.

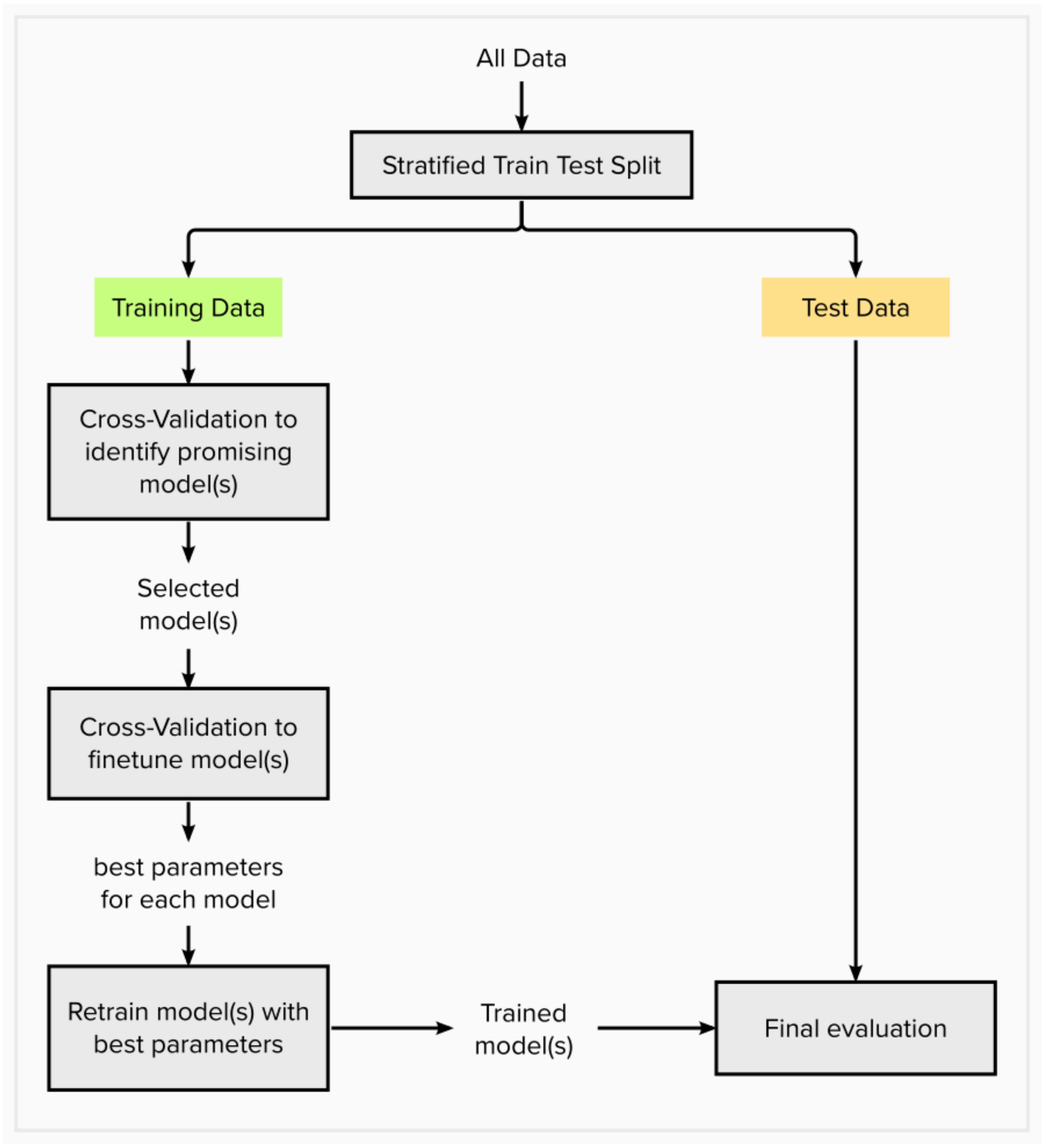

All Data
Stratified Train Test Split
Training Data
Test Data
Cross-Validation to identify promising model(s)
Selected model(s)
Cross-Validation to finetune model(s)
best parameters for each model
Retrain model(s) with best parameters
Trained model(s)
Final evaluation

Figure 16: Training methodology

4.2 Startified Train-Test-Split

Typically, the size of the test set makes 20% of the entire dataset and the particular instances that go into the training and test splits are chosen randomly. However, in our use case, the randomness of the splits may cause an imbalance in the representation of the concentrations in the training set, since some concentrations might end up being heavily under-represented and have very few examples. This means it is more challenging for a model to learn the characteristics of examples from this class, which would make the prediction of the concentration harder. In order to mitigate this problem, a variant of the split is used which is called *Stratified Train Test Split*. As the name suggests, this procedures ensures that the percentage of samples for each concentration is preserved.

4.3 Identifying promising models

After having split the entire preprocessed dataset into train and test subsets, the next stage is to shortlist a few promising models before spending any effort into fine-tuning (i.e hyperparameter optimization) by resorting to cross-validation, as explained in section 4.1.

4.3.1 Tested models

The models selected for the initial comparison are listed below:

1. Linear Models
    - Ridge Regressor: This model solves a regression model where the loss function is the linear least squares function and regularization is given by the L2-norm. Also known as Tikhonov regularization.
    - Lasso Regressor: This model solves a regression model where the loss function is the linear least squares function and regularization is given by the L1-norm.
    - Elastic Net: This model solves a regression model where the loss function is the linear least squares function and a combined L1 and L2 regularization.
2. Ensemble Models
    - Random Forest Regressor: A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.
    - Gradient Boosting Regressor: Gradient Boosting builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.
3. Neural Networks
    - Multi-layer Perceptron: A multilayer perceptron is a class of feedforward artificial neural network and consists of at least three layers of nodes: an input layer, a hidden

layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function.

4.3.2 Performance Measure

In order to compare between models, it is necessary to define a performance measure. In this section and throughout this project, we will use the mean absolute error (MAE), calculated as follows:

$$MAE = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \tag{2}$$

where:

- $\hat{y}_i$: a prediction of the i^{th} input.
- y_i: the true value of the i^{th} input.
- n: the total number of inputs

4.3.3 Cross-Validation

The table 1 below shows the results of a 10-fold Cross-Validation on all the models. The last two rows contain the mean and the standard deviation of the 10 evaluation scores.

	Models					
	Linear Models			Ensemble Models		Neural Network
Fold	Ridge	Lasso	Elastic Net	Random Forest	Gradient Boosting	Multilayer Perceptron
0	11.40	11.58	11.76	3.98	6.25	0.61
1	13.49	13.50	14.61	3.52	4.31	0.23
2	9.95	10.10	9.61	2.70	5.12	0.40
3	12.21	12.22	13.26	3.08	4.38	1.32
4	8.86	8.85	10.74	1.32	3.20	0.46
5	10.57	10.57	11.06	5.28	6.67	0.80
6	15.73	13.66	9.72	3.68	3.73	2.43
7	9.56	9.63	10.67	1.25	2.66	2.10
8	11.19	10.71	11.81	5.84	5.78	1.29
9	9.59	9.44	9.75	4.75	6.76	1.08
avg	11.25	11.02	11.30	3.54	4.89	**1.07**
std	2.09	1.67	1.62	1.52	1.45	**0.73**

Table 1: Result of 10-fold Cross-Validation for identifying promising models.

The table shows that all linear models consisting of Ridge, Lasso and Elastic Net have underperformed with an average score settling around 11, followed by Random Forest and Gradient Boosting Regressors with average scores of 3.54 and 4.89 respectively and a similar std around 1.5. Finally, the MLP outperformed all other models with average and std scores of 1.07 and

0.73 respectively. Based on these results, the MLP is chosen as the only model for further consideration.

4.4 Finetuning

4.4.1 Grid Search CV

After selecting the most promising model from the previous section, we need to fine-tune it in order to extract the best performance out of it. One naive option would be to fiddle with the hyperparameters manually until a great combination is found. This would be very tedious and time-consuming work. Instead, we resort to a procedure called *Grid Search CV*. In this procedure, the user specifies the hyperparameters to experiment with and what values to try out and it will use cross-validation (as described in section 4.1) to evaluate all possible combinations of hyperparameter values.

4.4.2 Round 1

The tested hyperparameter values are compiled in the following table:

Hyperparameter	Description	Tested values
alpha	L2 penalty (regularization term) parameter	0.3, 0.5, 0.9
hidden layer sizes	The ith element represents the number of neurons in the ith hidden layer.	(20,20,20), (40,40,20), (60,60,40)

Table 2: Hyperparameter values tested in Grid Search CV.

Therefore, the Grid Search algorithm will explore all $3 \times 3 = 9$ combinations of alpha and hidden layer sizes specified in table 2. For each combination, it will conduct a cross-validation with $n_{folds} = 10$ and report each time the average and standard deviation scores (avg & std) across all 10 folds. The results are shown in the table below:

Round 1	CV scores		Hyperparameters	
combination	avg	std	alpha	hidden layer sizes
1	0.98	0.69	0.3	(20, 20, 20)
2	0.84	0.60	0.3	(40, 40, 20)
3	0.89	0.63	0.3	(60, 60, 40)
4	1.48	1.53	0.5	(20, 20, 20)
5	0.93	0.59	0.5	(40, 40, 20)
6	**0.59**	**0.61**	0.5	(60, 60, 40)
7	1.07	0.69	0.9	(20, 20, 20)
8	**0.68**	**0.35**	0.9	(40, 40, 20)
9	**0.79**	**0.52**	0.9	(60, 60, 40)

Table 3: Result of Grid Search CV, Round 1

The avg and std scores shown in table 3 suggest that the best hyperparameter combinations (highlighted in bold) correspond with a high value of alpha and a higher number of neurons per hidden layer.

4.4.3 Round 2

A second round of fine-tuning is conducted: for alpha, the values 0.5 and 0.9 are retained and for the hidden layer sizes, (60,60,40) from round 1 is retained, to which two new values (60,60,60) and (120,40,20) are added.

Round 2	CV scores		Hyperparameters	
combination	avg	std	alpha	hidden layer sizes
1	0.59	0.61	0.5	(60, 60, 40)
2	1.10	0.76	0.5	(60, 60, 60)
3	**0.54**	**0.30**	0.5	(120, 40, 20)
4	0.79	0.52	0.9	(60, 60, 40)
5	1.13	0.55	0.9	(60, 60, 60)
6	0.62	0.44	0.9	(120, 40, 20)

Table 4: Result of Grid Search CV, Round 2

Table 4 shows the best solution is obtained by setting alpha to 0.5 and the hidden layer sizes hyperparameter to (120,40,20). The average and standard deviation score for this combination are 0.54 and 0.30 respectively. This result is slightly better than the scores we got by using the default hyperparameter values in Table 1 (which were 1.07 and 0.73 respectively), thereby demonstrating the usefulness of grid search as a means to improve a model's performance during the training phase.

5 Evaluation

5.1 Error distribution

Now is the time to evaluate the final model on the test set. There is nothing special about this process. The MLP is retrained on the entire training set using the best parameters and then used to predict the concentrations on the test set. The errors are calculated using the mean absolute error discussed in previous sections.

The reported training and test mean absolute errors are equal to 0.099 mg/mL and 1.845 mg/mL.

The figure 17 below shows the error distribution on the test.

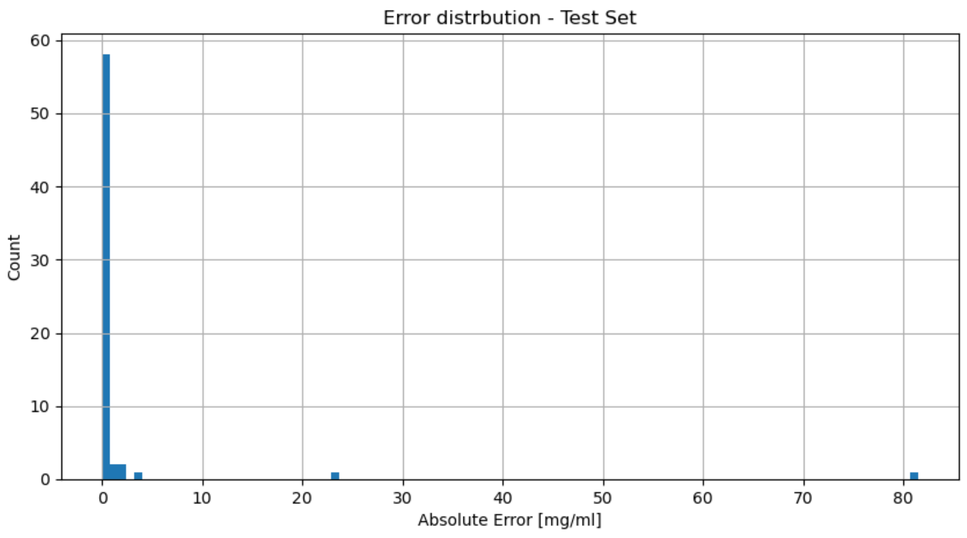


Figure 17: Error distribution on the test set

It is interesting to note that the majority of the errors are between 0 mg/mL and 5 mg/mL except two instances, which will be discussed shortly. For clarity, the same distribution is plotted but restricted to the range [0,5] to get a better histogram resolution.

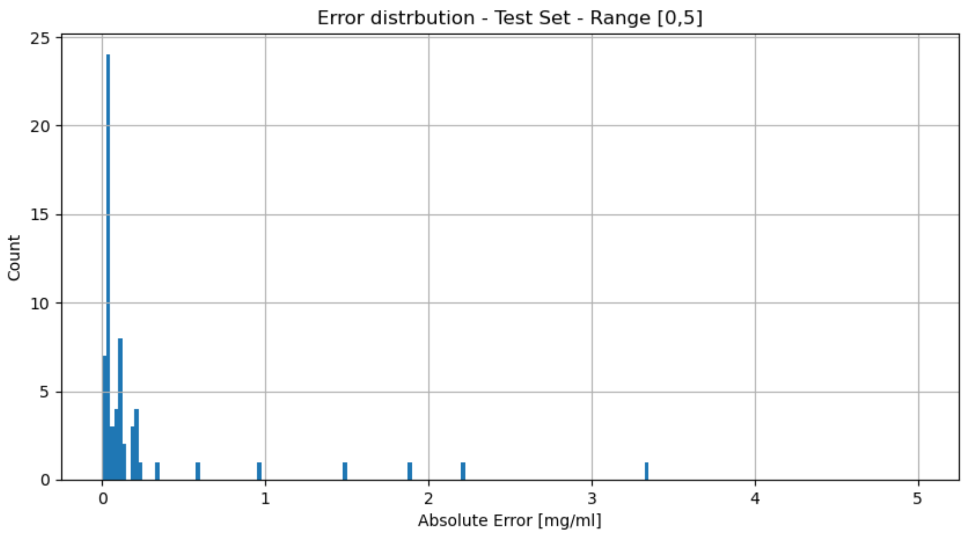


Figure 18: Error distribution on the test set, restricted to range [0,5]

Even in this zoomed view, it is notable that the majority of the errors lie below the threshold of 1 mg/mL and a few others between 1 mg/mL and 5 mg/mL. This observation is also confirmed when we take a look at the 10 instances with the highest error on the test set, which are listed in the table 5 below:

index	concentration	date	power	integration time	run	absolute error
10	20.0	17.12.2021	24	5	11	81.45
95	48.3	05.01.2022	24	10	6	23.34
221	19.8	05.01.2022	34	5	2	3.32
307	100.8	05.01.2022	34	5	8	2.21
243	48.3	05.01.2022	34	5	4	1.87
119	49.7	05.01.2022	24	10	10	1.49
108	49.7	05.01.2022	24	5	9	0.95
103	49.7	05.01.2022	24	5	4	0.59
147	100.8	05.01.2022	24	5	8	0.32
121	52.3	05.01.2022	24	5	2	0.24

Table 5: Instances corresponding to the 10 highest errors on the test set.

5.2 Outlier

The error of 81.45 is absurdly large, which warrants a more detailed investigation. The following plot shows the spectrum of this instance in red as well as other spectra from the same category (24 mW, 5 sec).

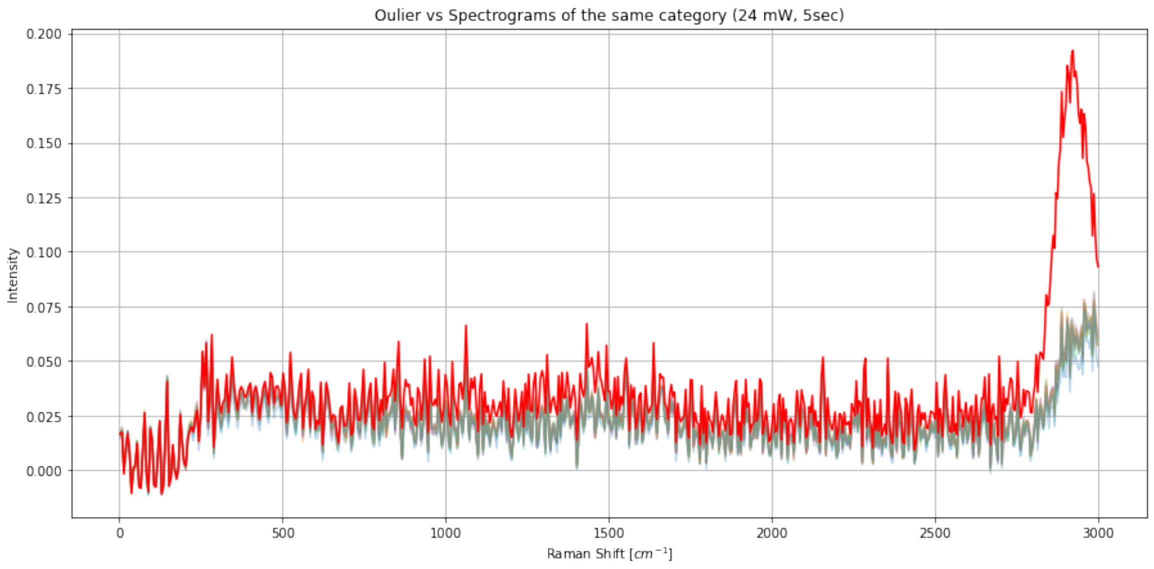


Figure 19: Outlier spectrum.

It is notable that this instance strongly differs with the other spectra in the raman shift interval between 2500 cm^{-1} and 3000 cm^{-1}. This discrepancy in the input signal leads to the calculation of features which diverge those corresponding to other signals from the same category, which ultimately results in an abnormal prediction of the concentration. It is interesting to note that the removal of this outlier from the test set bring the mean absolute error from 1.845 mg/mL down to an acceptable value 0.60 mg/mL.

6 Conclusions

This project proposes an ML-based approach to determine the concentration of pentanediol solutions using spectra of known concentrations. This approach consists in extracting statistical, spectral and temporal features from the spectra (see section 3.1) and conducting a feature selection to retain only valuable features (see section 3.2). Furthermore, it makes use of a set of techniques that are widely used in the domain of machine learning such as the Cross-Validation for drawing performance comparisons between different models (see sections 4.1 and 4.3) and Grid Search (see section 4.4) for hyperparameter optimisation.

Based on our evaluation, the proposed method's prediction accuracy is acceptable (average test error of 0.60 mg/mL without outliers, with predictions errors for most test instances under 1 mg/mL). However, it must be emphasized that this assertion holds as long as the query spectra correspond to concentrations present in the training data set. For a more exhaustive evaluation, the method must also be tested on query spectra whose concentrations are not included in the training data.

This implies that, if the goal is to make predictions for concentrations of a single substance like Pentanediol in a given range with the method in its current state, then the dataset should contain spectra of concentrations with a high enough resolution depending on the targeted accuracy. For example, assuming the desired interval is $[c_{low}, c_{high}]$ and the step is 1mg/mL, then the dataset should contain spectra for concentrations $[c_{low}, c_{low}+1, ..., c_{high}-1, c_{high}]$.

Finally, we suggest adding an outlier detection algorithm at the beginning of the preprocessing pipeline to improve the overall method.

References

[1] Yuki Maruyama Corey J. Cochrane George R. Rossman Jordana Blacksberg, Erik Alerstam. Miniaturized time-resolved raman spectrometer for planetary science based on a fast single photon avalanche diode detector array. *Applied Optics*, 2016. URL `https://opg.optica.org/ao/abstract.cfm?uri=ao-55-4-739`.

[2] Benjamin Bird Gianfelice Cinque Kelly Curtis Jennifer Dorney Karen Esmonde-White Nigel J Fullwood Benjamin Gardner Pierre L Martin-Hirsch Michael J Walsh Martin R McAinsh Nicholas Stone Francis L Martin Holly J Butler, Lorna Ashton. Using raman spectroscopy to characterize biological materials. *Nature Protocols*, 2016. URL `https://www.nature.com/articles/nprot.2016.036`.

[3] Welcome to tsfel documentation! URL `https://tsfel.readthedocs.io/en/latest/`.