PROJECT REPORT ML-IR : A Machine Learning Approach to Automated Infrared Spectra Interpretation

Motivation - The Problem

A Brief Overview of Infrared Spectroscopy for Characterization of Unknown Chemicals

As a chemistry major, one of the early topics covered in Organic Chemistry 1 is the identification and characterization of an unknown chemical. Being able to accurately identify a chemical in an unknown sample is incredibly important in a variety fields including pharmaceutical chemistry (determining if you've made your product of interest, quality control, etc), environmental chemistry (testing for the presence of environmental contaminants in a sample, tracking biomarkers), and astrochemistry (characterizing the gases present in nebulas, alien atmospheres, etc). Infrared (IR) Absorbance spectroscopy also known as Vibrational Spectroscopy has been used for many decades to identify specific functional groups in molecules and plays a key role in the identification of unknown chemical species. In this approach unknown chemical structures can be thought of as Lego sets and the infrared absorbance spectra reveals useful information about which pieces are used. Because each specific "piece" (for instance a Carbon bonded to an Oxygen [C=O]) has a different infrared absorbance (corresponding to the energy needed to vibrate that bond) the infrared spectra can be used to glean valuable information about key features present in a molecular system.

For Example the IR spectra for n-butanol shows a number of useful features corresponding to the different functional groups in Butanol (CH3-CH2-CH2-CH2-OH):

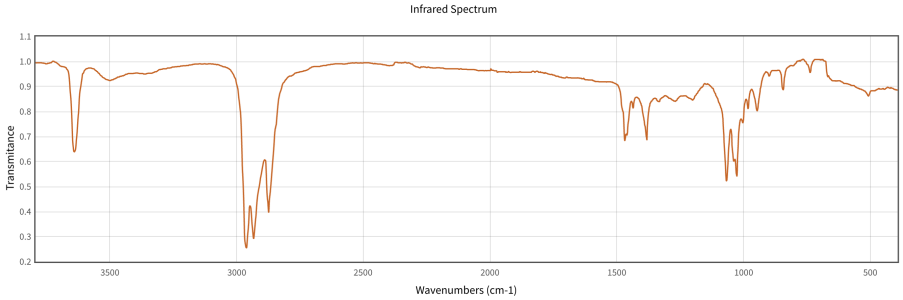

Figure 1: Butanol IR Spectrum

The y-axis of the chart represents the amount of light transmitted with a given energy in units of wave-numbers on the x-axis. For butanol the sharp peak at

~3600 cm^{-1} corresponds to the O-H bond vibration and the broad features at ~3000 cm^{-1} correspond to C-H bond vibrations.

Translation - IR Spectroscopy With Machine Learning

In a ML sense, a functional group is a multi-label classifier (ie. one compounds IR spectra often can indicate the presence of multiple functional groups). Ideally, I should be able to feed in the IR-Absorbance Spectrum as a collection of features and the model should be able to output a multi-label prediction for each suspected functional group. Automated detection of functional groups and chemical unknowns in test samples allows automated testing stations for environmental contaminants or chemical product impurities.

Building the Dataset

In order to train a ML model to recognize IR absorbance features and map them to functional groups I developed a basis set of chemical systems that contain the functional groups of interest. To limit the scope and computational cost of this project I focus on small organic molecules only containing Carbon, Nitrogen, Hydrogen,and Oxygen. I also focus on the following functional groups: C-H, C=C, C≡C, C≡N, C-OH, C=O, C=OOH, N-H, C≡C−H.

A further issue is the lack of mass-downloadable data for IR Spectra, meaning that digitized spectra were downloaded manually from the NIST web-book (8). As such, 101 Spectra were specifically chosen to form the training set (and accurately capture the functional groups of interest). These spectra include 40 Systems with only one main functional group aside from C-H and 60 systems that largely mix different functional groups together. The dataset contains a variety of infrared spectra in both the gas, liquid, and solid phase; however, all spectra are over a similar range of infrared energies (500-4000 wave-numbers). The lowest resolution spectra has ~83 points. Functional group labels were added manually for each spectra and encoded in a binary fashion. Hexanol for instance got the following identification [1,0,0,0,1,0,0,0,0] where index 0 and index 4 represent the C-H and O-H bonds respectively. Due to the small size of the dataset some classes were over-represented (C-H) while others were under-represented in the data (C=OOH). Figure 2 shows the class distribution for this dataset as well as for the train, test, and validation splits (more on this later). In an ideal situation there would be significantly more representation for all classes; however, the dataset is still large enough to perform some proof-of-concept ML modeling.

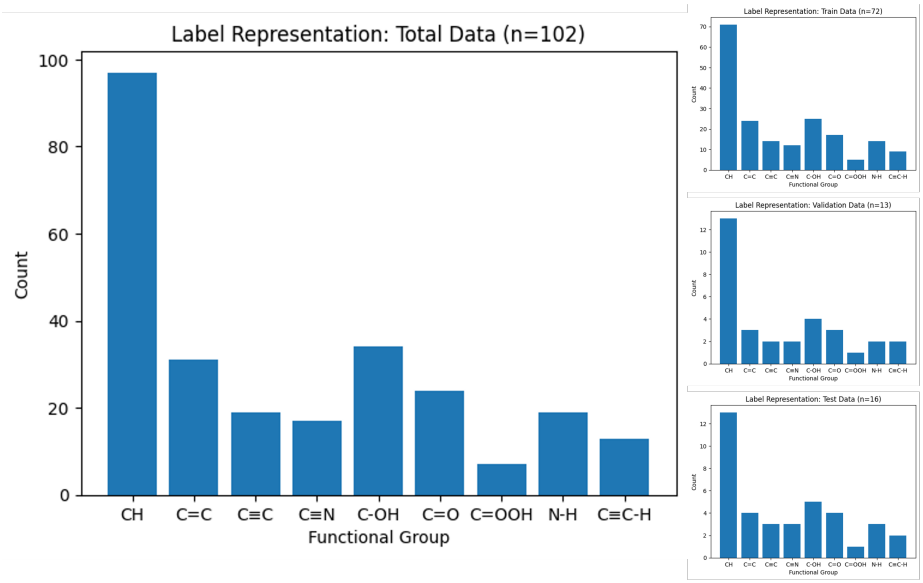

Figure 2: Functional Group Counts in Total Dataset, Train, Validation, and Test Splits

Preprocessing - Converting IR Absorbance Spectrum to Features

IR spectra were individually downloaded in "*.JCAMP" format from the National Institute of Science and Technology (NIST) which hosts free scientific data for a large selection of small organic molecules. Once downloaded, the continuous IR absorbance values were binned and interpolated using scipy's "interp1d" module (9) to generate 100 features ranging from 500-4000 cm^{-1}. Once binned and interpolated the values were normalized for each spectra such that the largest absorbance intensity, I, in each spectra was set to (I/I) 1 and other intensities, i, were scaled to i/I. This normalization was performed to deal with various absorbance units and prevent features with differently scaled intensities from being over-represented in the data. Additionally, spectra with X-units in micrometers as opposed to wave-numbers were converted to wave-numbers. Transmission spectra were converted to absorbance spectra using the formula A = 1 - T where A is light absorbance and T is light transmission. Preprocessed experimental spectra were manually assigned appropriate functional groups based on their chemical structures and written to a master csv file, "Full_File.csv". The master csv file had the following general format where A1-A100 are the binned absorbances ranging from 500-4000cmˆ-1 and C-H . . . are the functional groups/labels:

FileName	A1, A2, A3 ... A100	C-H, C=C ...
hexene	0, 0.1, 1 ... 0.1.	1, 1, ...

These data generation and preprocessing steps are performed in the jupyter notebook “load.ipynb”.

Once compiled into a "csv" file, the data was read in as a pandas data-frame and split into training, testing, and validation sets of size 70, 16, and 13 respectively. Splitting the data presented another challenge, as simply using train_test_split(stratify=y) was not feasible due the number of possible class combinations and small dataset. For nine functional groups there are 2^9 combinations, and stratify requires at least two instances of every class combination in order to evenly distribute instances. With 100 data points it was not possible to capture enough systems with duplicate functional groups for stratify to work. Instead MultilabelStratifiedKFold() was used to at least partially ensure adequate representation for each label in the test, train, and validation sets. The subplots of Figure 2 show the class distribution for the training, test, and validation sets. All three subplots show similar distributions to the total dataset indicating relatively successful data splitting; however, some classes like C=OOH are only represented by 1 or 2 instances in the test and validation sets. This is suboptimal but necessary given the limited size of the dataset.

Model Selection

KNN is one of the only models we have covered that supports this type of multi-label classification problem as a default option, simply choosing the best multi-label based on the most represented K neighbors. However, sklearn(6) provides a solution that allows other classifier models to be used as multi-label classifiers: the OneVsRestClassifer() module. This module takes a classifier model (Decision Tree, SVC, Random Forest, etc) and trains one instance of the classifier model for each label in the data. Using this module, each sub-model only determines whether an individual binary label is applicable or not. Then, the predictions of each sub-model for each class are combined to generate a multi-label prediction (ie. each sub-model contributes a 1 or 0 for each class). In our case there are 9 functional groups (binary labels) so the OneVsRestClassifier() module generates 9 binary classifiers that output a 9 digit binary multi-label where each column corresponds to a different label.

Using the OneVsRestClassifier(), the following models were selected with a goal of employing a comprehensive range of different algorithms used in this course: KNN, Logistic Regression, Decision Tree, Random Forest, Linear SVC, and RBF SVC. Additionally, the “class_weights” parameter was set to “balanced” for applicable models in an attempt to mitigate the severe class imbalances found in the dataset (Figure 2). The balanced parameter weights classes inversely proportional to their representation in the dataset forcing the model to pay more

attention to otherwise under-represented classes.

Evaluation Metrics

The models were evaluated using the accuracy score and the F1 score of the train, test, and validation data. The accuracy score reflects whether the ML model accurately predicted the presence or absence of each of the 9 functional groups. Since there are nine possible labels per spectra, a single incorrect prediction or eight incorrect predictions are treated the same in this approach despite one model clearly outperforming the other. To deal with this limitation of accuracy scores, F1 scores were also used. F1 scores measure the average of the precision and recall. Precision can be defined by the following equation: $P = \frac{True_P}{True_P + False_P}$ and measures the tendency of the model to predict false positives (ie. wrong functional groups). Recall can be defined by: $R = \frac{True_P}{True_P + False_N}$ and is measures the tendency of the model to predict false negatives (ie. miss functional groups). This combination of precision and recall provides a more balanced assessment of the model strength especially for a relatively imbalanced dataset where some functional groups (C-H) are over-represented. By calculating F1 scores for each class and averaging a total F1 score was generated for the data for each model. Individual class performances were also assessed using the individual scores.

Hyperparameter Optimization

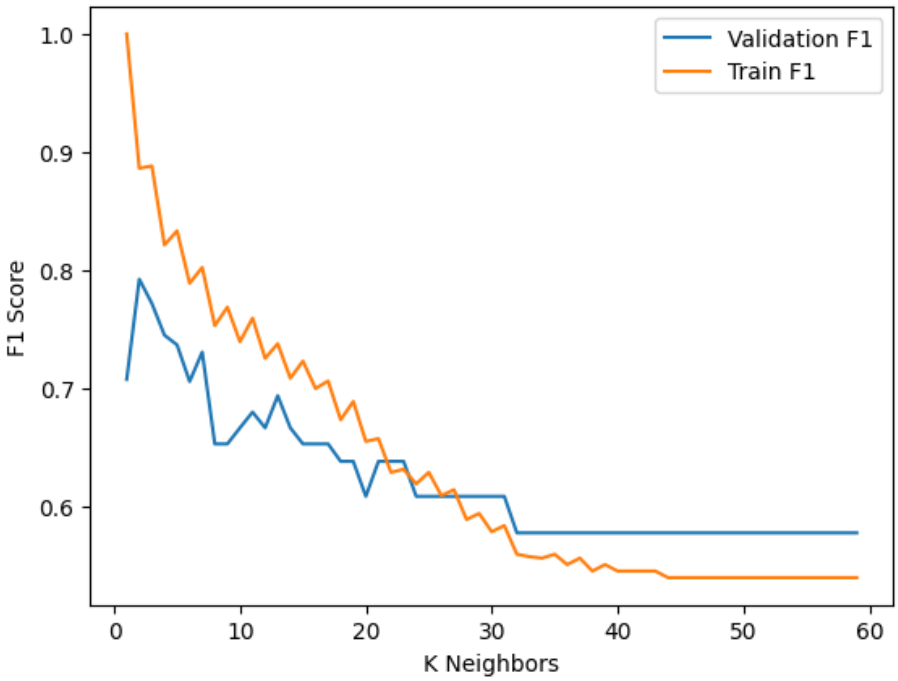

Figure 3: KNN Hyperparameter Optimization

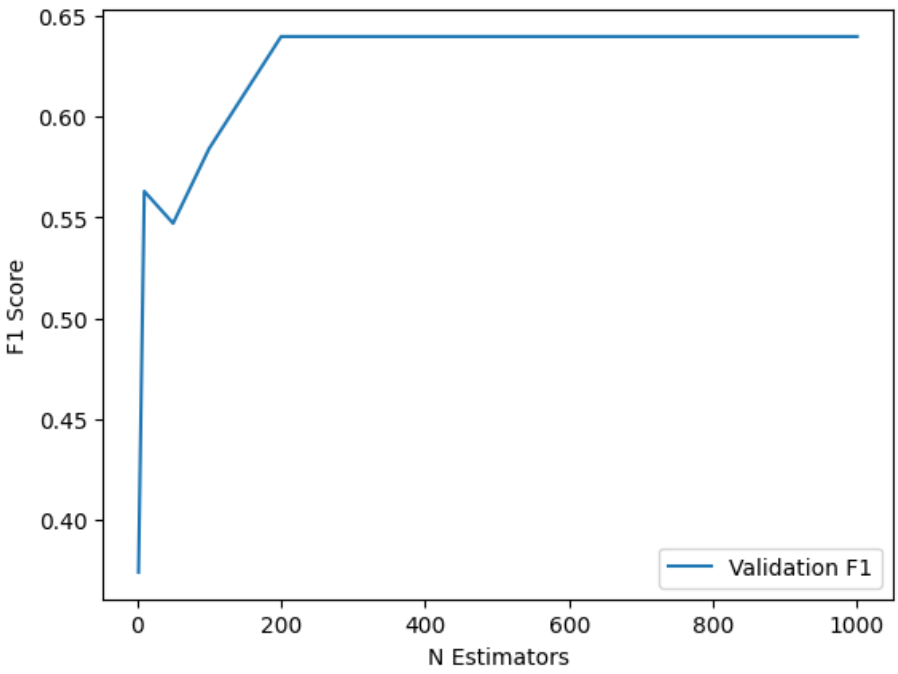

Figure 4: Random Forest Hyperparameter Optimization

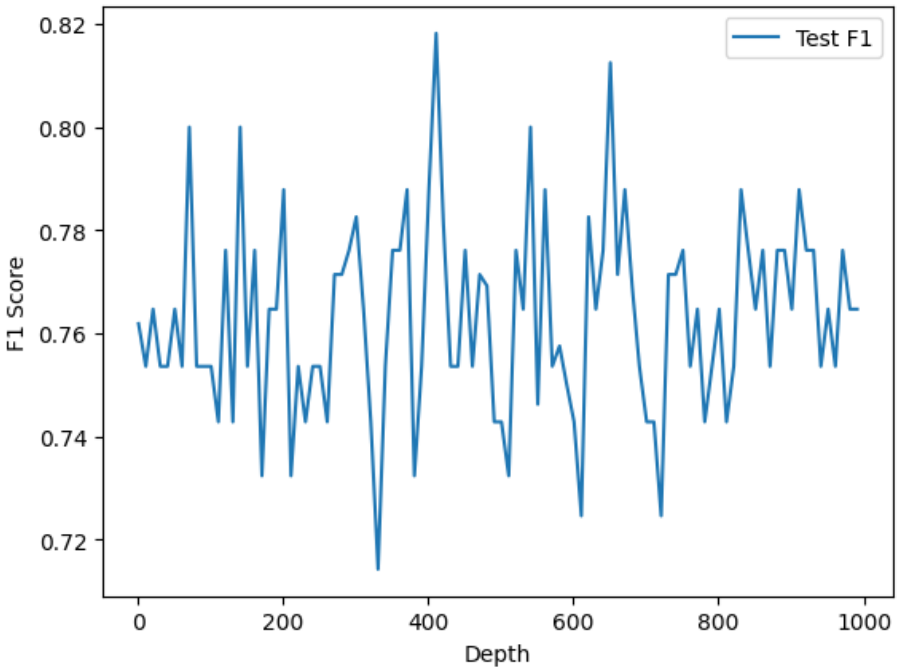

Figure 5: Decision Tree Hyperparameter Optimization

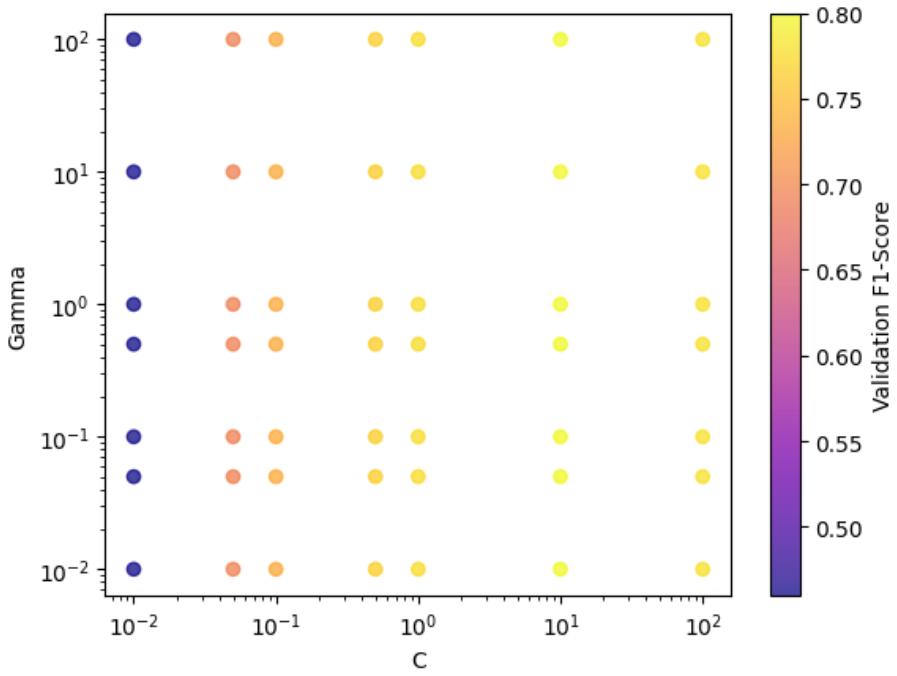

Figure 6: Linear SVC Hyperparameter Optimization

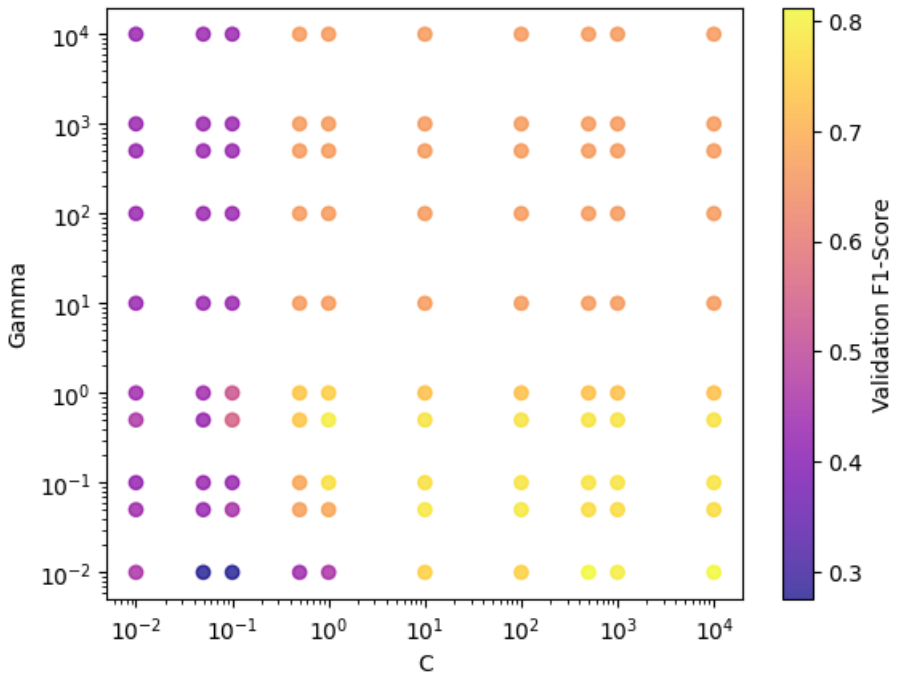

Figure 7: RBF SVC Hyperparameter Optimization

Hyperparameter optimization was performed for Random Forest, Decision Tree, Linear SVC, RBF SVC, and KNN machine learning models. The F1 Score calculated on the validation data was used as the optimization parameter when performing hyperparameter optimization. For models with more than 1 hyperparameter (SVC) a grid search was performed to determine the best combination of hyperparameters. The best set of hyperparameters are summarized in the table below:

Table 1: Hyerparameter Optimization Results Results

Model	Hyperparameters Selected
KNN	n_neighbors=2
Decision Tree	max_depth=420
Random Forest	n_estimators=200
Linear SVC	C=10; Gamma=0.05
RBF SVC	C=500 ; Gamma=0.5

Results and Performance Tables

Overall Model Performance on Validation Data

For average evaluation metrics on the Test Data (Tables 6-7), the top two models were Decision Tree and Random Forest with KNN and RBF SVC in close competition for 3rd (Table 6). RBF SVC had a better Macro F1-Score but KNN performed better with respect to accuracy. There was no one model that outperformed the others across all classes on the test data (Table 7). Moreover, some classes like C-H with a distinct feature at 3000 wave-numbers and high representation in the dataset were far easier to predict than others. Hard to predict labels included C=OOH and C≡N. C=OOH functional group prediction requires recognition of two distinct vibrational features (C=O and -OH) combined with a broadening of the O-H stretch and C≡N prediction requires detection of a very small feature that models are prone to miss as noise.

Ultimately, these predictions show that machine learning models are capable of predicting functional groups of small organic molecules based on their IR absorbance spectra; however, the limited size of the dataset and severe class imbalances make this a proof-of-concept study rather than a deployable model. This limitation is most apparent when looking at the training data vs test data results. On the training data, all models performed moderately well with respect to accuracy and F1 Score with cross model average values of 0.80 and 0.94 respectively (Table 2). Models also performed well with respect to individual labels with the C≡N and C=OOH functional groups having the two lowest cross model average F1 Scores of 0.86 and 0.81 respectively. However, when asked to predict spectra not in the training or validation data all of the models struggled with an average 41% overall cross model accuracy decrease and 26% overall cross model F1-Score decrease. Additionally, individual label performance dropped significantly for each label with C≡N and C=OOH cross model F1 Scores going from 0.86 and 0.81 to 0.15 and 0.00 respectively.

This drop in both overall performance and performance by class is indicative of overfitting to the training data as well as an inherent restraint of the small dataset. The models simply have not seen enough data to perform well (>90% Accuracy/F1 Scores) on unknown test data and this effect was only compounded for under-represented labels such as C=OOH and C≡N.

Table 2. Overall Performance on Training Data

Model	Accuracy Score	Micro F1 Score	Macro F1 Score
KNN	0.5417	0.8863	0.7986
Logistic Regression	0.5694	0.9110	0.8684
Decision Tree	1	1	1
Random Forest	1	1	1
Linear SVC	0.9030	0.9645	0.9820
RBF SVC	0.7634	0.9490	0.9202

Model	Accuracy Score	Micro F1 Score	Macro F1 Score
Cross Model Average	0.7963	0.9518	0.9282

Table 3. Performance by Class on Training Data (F1 Score)

Model	C-H	C=C	C≡C	C≡N	C-OH	C=O	C=O	OH	N-H	C≡C-H
KNN	0.99	0.77	0.92	0.59	0.86	0.94	0.57	0.67	0.88	
Logistic Regression	0.99	0.82	0.84	0.75	0.94	0.97	0.67	0.93	0.9	
Decision Tree	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	
Random Forest	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	
Linear SVC	1.00	0.96	1.00	0.96	0.98	1.00	0.83	1.00	0.95	
RBF SVC	1.00	0.91	0.90	0.83	0.94	1.00	0.83	0.97	0.90	
Cross Model Average	0.996	0.910	0.943	0.855	0.953	0.985	0.817	0.928	0.938	

Table 4. Overall Perforamnce on Validation Data

Model	Accuracy Score	Micro F1 Score	Macro F1 Score
KNN	0.3846	0.7925	0.5778
Logistic Regression	0.3077	0.7826	0.6854
Decision Tree	0.3846	0.7462	0.6384
Random Forest	0.4615	0.8115	0.6400
Linear SVC	0.3846	0.800	0.6471
RBF SVC	0.4615	0.7937	0.6598

Table 5. Performance by Class on Validation Data (F1 Score)

Model	C-H	C=C	C≡C	C≡N	C-OH	C=O	C=O	OH	N-H	C≡C-H
KNN	1.00	0.00	0.67	0.67	0.40	0.80	0.00	1.00	0.67	
Logistic Regression	1.00	0.67	0.80	0.44	0.40	0.86	0.00	1.00	1.00	
Decision Tree	0.96	0.57	1.00	0.29	0.57	0.86	0.00	0.5	1.00	
Random Forest	1.00	0.50	0.67	0.67	0.40	0.86	0.00	1.00	0.67	
Linear SVC	1.00	0.67	0.50	0.67	0.67	0.86	0.00	0.80	0.67	
RBF SVC	1.00	0.80	0.50	0.57	0.40	1.00	0.00	1.00	0.67	

Table 6. Overall Performance on Test Data

Model	Accuracy Score	Micro F1 Score	Macro F1 Score
KNN	0.3750	0.7164	0.5607
Logistic Regression	0.1875	0.7126	0.6412
Decision Tree	0.5625	0.7442	0.6232
Random Forest	0.6250	0.8115	0.6566
Linear SVC	0.2500	0.7058	0.6440
RBF SVC	0.3125	0.7105	0.6100
Cross Model Average	0.3854	0.7335	0.6226

Table 7. Performance by Class on Test Data (F1 SCORE)

Model	C-H	C=C	C≡C	C≡N	C-OH	C=O	C=OOH	N-H	C≡C-H
KNN	0.90	0.40	0.80	0.00	0.75	0.40	0.00	0.80	1.00
Logistic Regression	0.9	0.75	1.00	0.17	0.80	0.86	0.00	0.50	0.80
Decision Tree	0.9	0.80	0.75	0.57	0.67	0.86	0.00	0.40	0.67
Random Forest	0.9	0.86	0.80	0.00	0.89	0.67	0.00	0.8	1.00
Linear SVC	0.90	0.60	0.80	0.18	0.89	0.86	0.0	0.57	1.00
RBF SVC	0.90	0.57	0.67	0.00	0.89	0.67	0.0	0.80	1.00
Cross Model Average	0.90	0.66	0.80	0.15	0.82	0.72	0.00	0.65	0.91

Test Data Confusion Matrices for Each Model on C-H,C=C, C≡N, and C=OOH Functional Groups

Test data confusion matrices were computed for the C=C, C-H, C≡N, and C=OOH functional groups for the three most accurate models: Random Forest, Decision, and KNN. The confusion matrices show stronger model performance for more represented groups (C-H, C=C) as well as the low positive identification rate of the C≡N feature across all models.

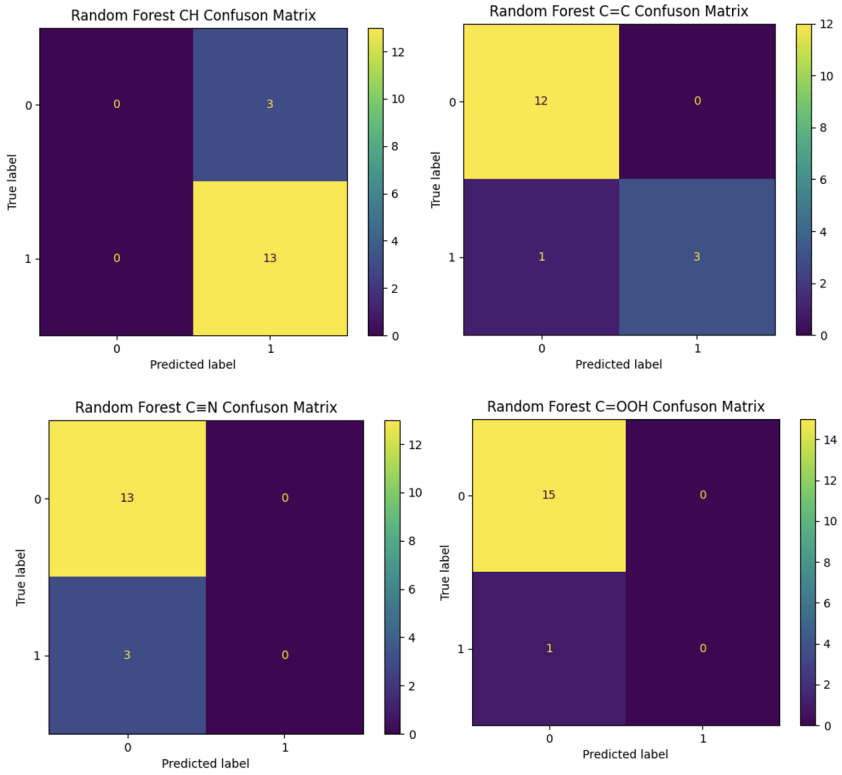

Figure 8: Random Forest Confusion Matrices for the C-H, C=C, C≡N, and C=OOH functional groups

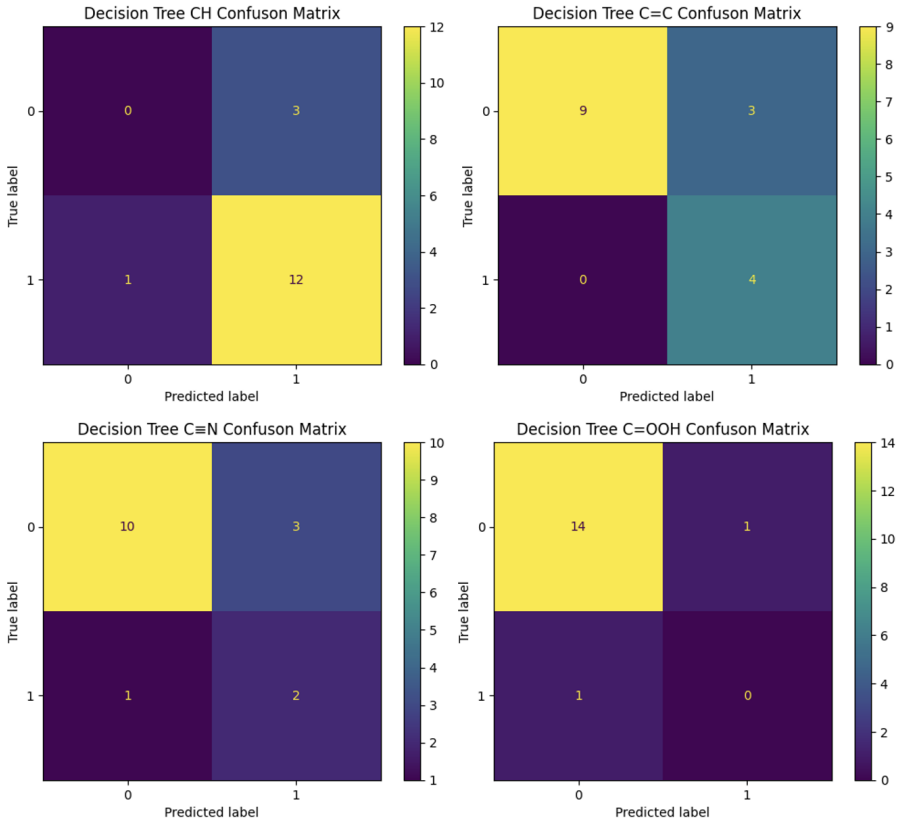

Figure 9: Decision Tree Confusion Matrices for the C-H, C=C, C≡N, and C=OOH functional groups

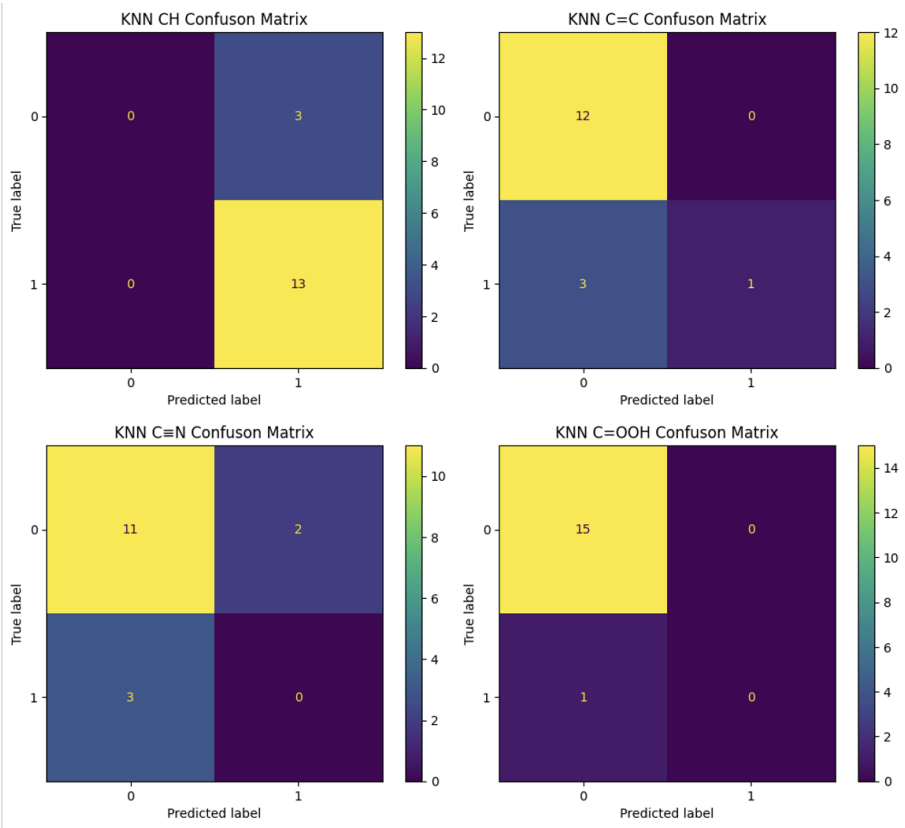

Figure 10: KNN Confusion Matrices for the C-H, C=C, C≡N, and C=OOH functional groups

Interpretation

I have implemented a simple graphical interpretation tool that show the experimental spectra along side the key features in the spectra that led to the identification of a specific peak for the best performing machine learning model: Random Forest. The goal of this tool is to showcase which parts of the spectra the model is using to make its prediction for both debugging and interpretation purposes.

Let's look at an example compound in the test set: Propanamide, which has the C-H, C=O, N-H functional groups Because Random Forest had the best performance on the test set we will use it to interpret the experimental spectrum and assign functional groups. Running the predict function in my jupyter notebook for "propanamide" produces the following output:

"Random Forest Fx Group: ['CH', 'C=O', 'N-H'] Correct Fx Group: ['CH',

'C=O', 'N-H']"

The random forest feature importance scores of the three individual random forest models corresponding to CH, C=O, and NH can then be plotted against the experimental spectrum to show what parts of the data were most important for each positive functional group identification. It is clear from the figure that the Random Forest model was able to identify the characteristic C=O stretch at 1600 cm^{-1}, the early N-H band at ~500 cm^{-1}, and the C-H stretch at ~3000 cm^{-1}.

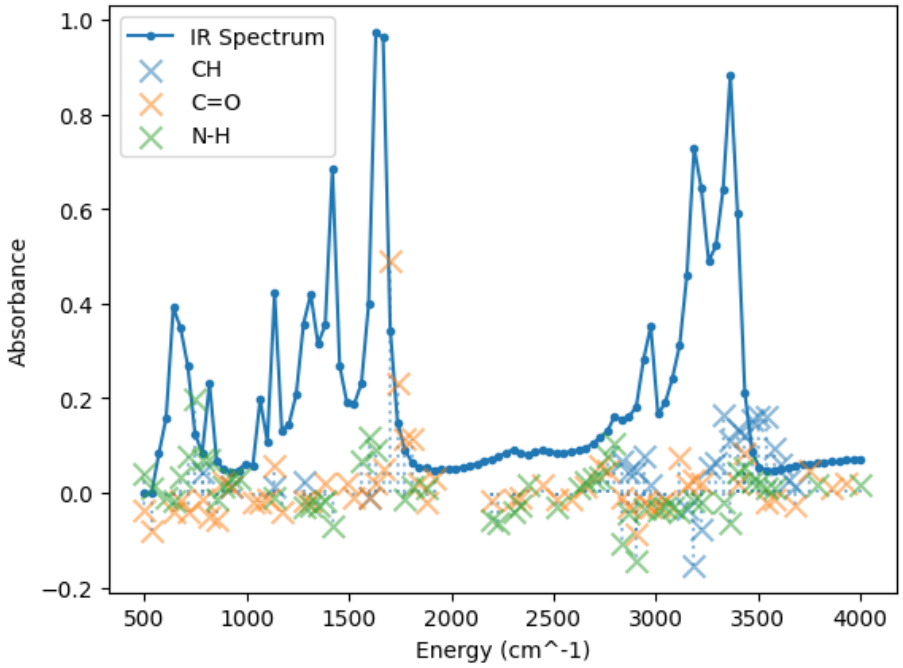

Figure 11: Random Forest Spectral Interpretation

Discussion

It is worth noting, that there are numerous recent research studies using machine learning models to identify functional groups and sometimes entire chemical structures from infrared absorption spectra. Specifically researchers have employed Convolutional Neural Networks(1,2), Deep Learning(4,5), and transformer models(3) to generate predictions with reasonably high accuracy (60-90% depending on the study/model) for a large number of functional groups (40+). By scraping the internet, paying for proprietary datasets, and generating artificial data with ab initio simulations, these researchers have access to training sets with tens of thousands of spectra. In an ideal world I would have tried to gain access to one of these datasets or generate my own scraper, but at my current

level of python experience this was difficult to do within the time constraint of this project.

Despite the numerous available machine learning models for infrared absorbance spectra interpretation, much of this work is still performed by human researchers. Nonetheless machine learning models have become more relevant, being sold to researchers as product by publication companies with large datasets like Wiley. These tools have also been used in autonomous labs where robots mix chemicals and generate IR absorbance spectra which are then analyzed using ML tools and ab initio quantum mechanics simulations(7).

In conclusion, I have implemented a proof-of-concept style suite of machine learning models in an attempt to address a real world problem related to the field of chemistry: the interpretation of infrared absorbance spectra for functional group analysis. Although the small size of the dataset limited model performance, the most accurate model, Random Forest, was able to accurately predict an entire 9-digit multi-label 62.5% of the time for a set of test systems, indicating the potential viability of this approach.

References

1) https://pubs.acs.org/doi/10.1021/acsomega.5c01903
2) https://pmc.ncbi.nlm.nih.gov/articles/PMC10055241/
3) https://www.nature.com/articles/s42004-024-01341-w
4) https://pmc.ncbi.nlm.nih.gov/articles/PMC8152587/
5) https://pubs.acs.org/doi/10.1021/acs.analchem.5c03126
6) https://scikit-learn.org/stable/supervised_learning.html
7) https://www.chinesechemsoc.org/doi/10.31635/ccschem.025.202505768
8) https://webbook.nist.gov/chemistry
9) https://docs.scipy.org/doc/scipy-1.16.2/reference/generated/scipy.interpolate.interp1d.html