Proteomics data analysis using PyOpenMs

Shubhangi Kaushik

Abstract

The goal of the project: The goal of this project was to train two machine learning(ML) methods like Adaboost and XGBoost on top of the proteomics data pre-processed using PyOpenMs, and then evaluating the working of the two models.

Main results of the project: Post binning, heatmaps were generated to visualize the frequency distribution of values for Retention time(RT) and mass by charge(mz) in each bin. After this step, machine learning methods were applied on top of the results, generating accuracy, precision, sensitivity, specificity, confusion matrix, and F-1 score along with the ROC curve to check the performance of the two ML methods. The project helped me learn the importance of pre-existing processing tools like like PyOpenMs. Practical implementation of data binning and data flattening helped with understanding the importance and scope of the concepts.

Goal

The goal of this project was to train two ML methods on top of the two classes i.e. class L L and class H H of the proteomics data[8] provided in the lecture slides. Before conducting the ML analysis, the data from both classes were to be preprocessed using PyOpenMs[6], and features were extracted. To reduce the dimensionality of the data, data binning was implemented and the bins containing the highest amount of RT and mz values were selected. Following it, the selected data from each file of both classes were flattened into a 1D array. Eventually, each file became sample rows for the classes. Data augmentation was conducted on each class separately, and then they were merged and ML methods were trained and evaluated on top of them.

Data

The dataset used in the study is a comprehensive full factorial LC-MS/MS proteomics benchmark dataset[8]. It is made up of Escherichia coli homogenates that have been tryptically digested using chicken ovalbumin (OVA) and bovine carbonic anhydrase II (CA). Two of their classes were chosen for the analysis: the ”LL” class, which represented samples spiked with low levels of both CA and OVA and the ”HH” class, which included samples spiked with high levels of both proteins. Both classes consisted of six samples each.

Liquid chromatography[3] and tandem mass spectrometry[2] are two procedures that are combined to form LC-MS/MS, or Liquid Chromatography-Tandem Mass Spectrometry. The material is initially separated using liquid chromatography, after which it is ionized and fragmented in the mass spectrometer.

Data Preprocessing

The data is publicly available for download in various formats at https://cac.science.ru.nl/research/data/ecoli/. The mzXML files for the twelve samples, six for L L class and six for H H class were downloaded. The features or the peaks were extracted from them using PyOpenMS (Figure 1(a)), which is a Python library that offers access to a feature-rich library of open-source algorithms for proteomics analysis based on mass spectrometry. The mass spectrometry peaks observed in the samples correspond to the ions that were identified and measured by the mass spectrometer during the analysis process. These features are detected in the form of 2-dimensional patterns in mz and time (RT) dimensions.

The identified features were subsequently organized into bins in a 40x40 dimensional array (Figure 1(b)) and visualized in a 2D histogram. The color of each bin corresponded to the frequency of features in those bins. The primary purpose of this binning process was to reduce the dimensionality of each sample. To accomplish this reduction, only the top 100 bins from the binned data(Figure 1(c)), focusing on the most significant and informative bins were selected.

Initially, the number of detected features for each sample exceeded 40,000. However, through binning, the features were reduced by a quarter, hence reducing the data volume effectively.

For each class, the data with significant bins were retrieved from each sample of the class. The data was flattened for both classes and then augmented. For augmentation, two methods were implemented:-

First, the data from both classes were merged and then augmented using a RandomOversampler, and a noise factor of 0.05 was introduced to the augmented data based on the normal values. As the augmentation was conducted after merging both classes together, this caused the augmented data to have less variation within the class and more variation between the two classes. Hence the test data points were very similar to the training data points, causing the ML method to work with 100% accuracy while classifying the test data points. This approach was considered to be biased. The second approach was to augment the data separately using StandardScaler and then combine it together and feed it into the ML methods. The results of this approach were more realistic, as the data augmentation of each class happened independently of the other class.

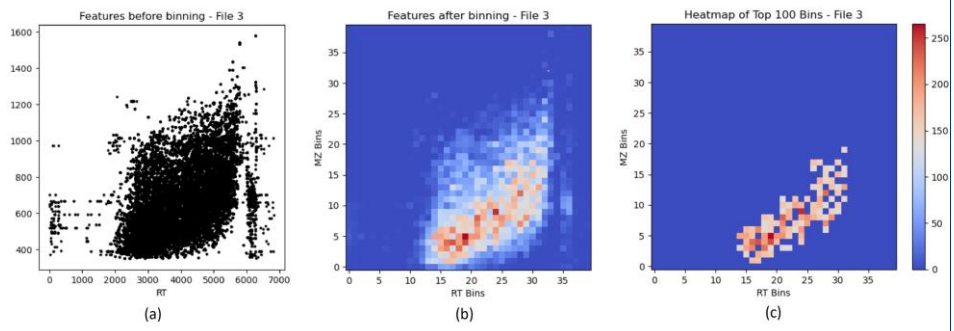

Figure 1: Figure (a) shows the scatter plot between MZ and RT for the Feature detected by pyOpenMS. In Figure (b), the plotted features are displayed after grouping them into bins. The value assigned to each bin is determined by adding up the number of values contained within it. The bin size is represented using a heatmap, where the intensity of the red color indicates the bin with the highest frequency, while the blue color represents the bin with the lowest frequency. Figure (c) shows the filtered plot for the bins with the 100 highest frequencies.

To confirm the data was getting visualized correctly, a few values were added at RT equal to 1000 and mz equal to 1200. While visualizing the data, the added values were visible in the 2D histogram and the scatterplot. Hence, it was confirmed that the data binning was working properly on the data.

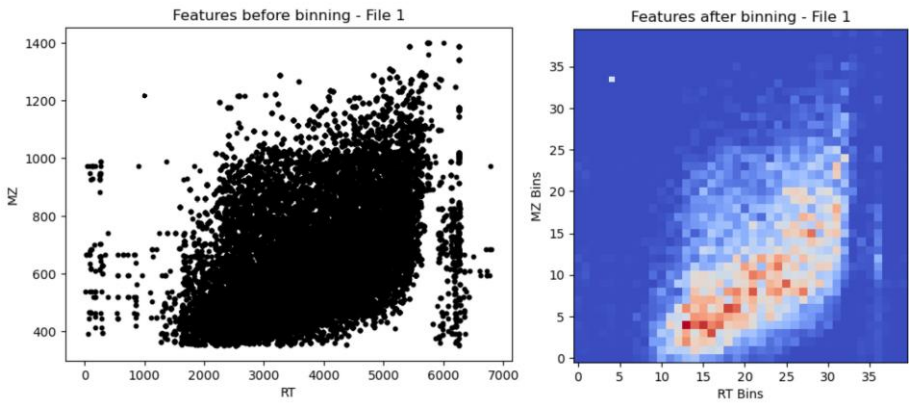

Figure 2: Figure (a) shows the scatter plot between MZ and RT for the Feature detected by pyOpenMS. Along with that at position 1000 RT value and 1200 mz value an artificial spike can be noticed. This spike was added to test the working of data binning. In Figure (b), the plotted features are displayed after grouping them into bins. Here the artificial spike exhibits the color light blue suggesting a medium-strength frequency signal for that particular bin.

Method

Various tools and machine learning processes were used to build the models and evaluate the results. At first, the merged dataset was taken and divided into two datasets X and Y. Where X contained all the values of the merged classes except the label class and dataset Y only contained the labels. X dataset was later scaled and following this both the datasets were divided into training and testing sets.
Which later was fed into the ML methods.

XGBoost

XGBoost[1] or Extreme Gradient Boost is an advanced implementation of gradient boosting. Gradient Boosting[4] trains weaker algorithms like Decision Trees sequentially, such that every later model corrects the mistakes made by the former model by minimizing a loss function. On top of that, XGBoost incorporates regularization techniques like L1 (Lasso) and L2 (Ridge) as well that add penalty terms to the loss function. This controls the complexity of the model and prevents overfitting.

AdaBoost

AdaBoost or Adaptive Boosting[7] is an ensemble method as well that iteratively trains weak learners by adjusting the weights of misclassified instances. It gives higher importance to instances that are challenging to be classified correctly in the next iteration. It can use any weak learner like Decision Slump and focuses on improving overall accuracy with upcoming iterations.

Evaluation

The machine learning model was evaluated using the prediction probabilities calculated using the K-Fold cross-validation technique where the number of folds was set to 5. Based on the prediction probabilities, the class was predicted for both train and test datasets. These predictions were evaluated using the following:

Confusion Matrix

A confusion matrix is a table that summarizes the performance of a classification model by giving the counts of true positive, true negative, false positive, and false negative predictions (Table 2). Based on these values, the scores for Accuracy, Sensitivity, Specificity, Precision and F1 Score are calculated

Predicted/Actual	Positive	Negative
Positive	TP	FP
Negative	FN	TN

Table 1: Confusion Matrix

Accuracy: (TP + TN) / (TP + TN + FP + FN)
Sensitivity (Recall): TP / (TP + FN)
Specificity: TN / (TN + FP)
Precision: TP / (TP + FP)

F1 Score: 2 * (Precision * Sensitivity) / (Precision + Sensitivity)

ROC Curve

(Receiver Operating Characteristic) Curve[5] for each class was also generated using the probabilities. It plots the true positive rate against the false positive rate at different classification thresholds.

Results and Discussion

Initially, the intention was to utilize RT (retention time) values as features and MZ (mass-to-charge ratio) values as the data points. This involved restricting the precision of the RT values to two decimal places and calculating the mean of the MZ values in case there are multiple data points sharing the same RT value. Our objective was to eventually create a consolidated dataset for machine learning analysis encompassing both, L L and H H datasets, for the common RT values. However, common RT points could not be found in the two datasets and hence the plan was dropped. The precision was reduced to 0 decimal places as well, but still could not find any common data points in the two datasets.

Therefore, it was decided to proceed with the flattening approach without picking the common RT values. By flattening the data from the six samples in each dataset, a total of 12 rows were obtained. Out of these, six rows corresponded to the H H class (class 1), while the remaining six rows corresponded to the L L class (class 0). The data was then augmented for each class using standard scalar library and the data for each class was increased up to 100 rows.

After applying the ML methods the results obtained were as follows:-

Model	Accuracy	Sensitivity	Specificity	Precision	F1 Score
AdaBoost(Train)	96.2%	95.53%	96.97%	97%	96.2%
AdaBoost(Test)	94%	93.94%	94.12%	93.9%	93.9%
XGBoost(Train)	95.5%	94.03%	96.97%	96.9%	95.5%
XGBoost(Test)	94%	87.88%	1%	1%	93.5%

Table 2: Accuracy, Sensitivity, and Specificity of the AdaBoost and XGBoost models for both the training and test sets. Based on these metrics, both models seem to work really well. AdaBoost generally performs slightly better than XGBoost when it comes to training data for all 5 metrics. However, for the test set, each algorithm does equally well for accuracy and also the AUC Score(Figure 2) but for Specificity and Precision, XGBoost works better while for Sensitivity and F1 Score, AdaBoost works better.

The sample set which was obtained for each class had different lengths, this could have hindered the analysis, moreover, as obtaining the common RT values was not possible. All the samples which had higher feature lengths were subsetted to the length of the lowest sample. Although this approach can be considered to be flawed, nonetheless it was the only way that seemed to carry the analysis further. Hence, it was implemented.

An important feature analysis was planned to be conducted. However, the approach was not plausible, as the features were the flattened data for MZ and RT values of each

sample. Even though every feature denoted either the MZ or RT value but there was no commonality within each column. Hence it was impossible

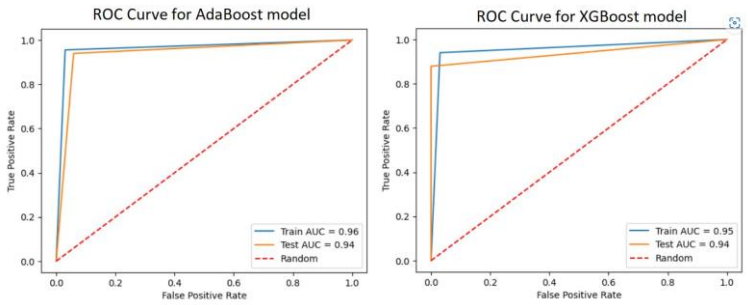

Figure 3: ROC Curve for the AdaBoost and the XGBoost model. In both, the orange line indicates the ROC curve for the test dataset whereas the blue line indicates the ROC curve for the training set. For the training data, it can be observed that the AdaBoost model performs better. For the test data, the higher line and leftward shift in AdaBoost model suggest that it achieves a higher TPR at lower FPR values compared to XGBoost model indicating a more favorable trade-off between TPR and FPR.

to conduct a feature importance study on the flattened data. Nonetheless, if the feature importance study was conducted, it would have retrieved the numerical values depicting mz and retention time, it would have been very hard to get the biological context for these values.

An alternative method for conducting the analysis could have been chosen, which involved employing the initial approach of identifying shared reference time (RT) points in the files. Instead of focusing on individual RT points, RT point bins could have been chosen. In future, I plan to do so.

Overall, it can be concluded that PyOpenMs works well on the proteomics data and could improve the working of ML methods on proteomics data. With proper data to backtrack the features to, a proper Ml analysis and feature importance study can be conducted.

Best Feature in the code

The binning part could be considered a nice part of the code. Although the construction of the graph is not very complicated. However, it is the way it describes the frequency distribution in a 2D array, which is impressive.

Appendix

Author details

References

1. Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.
2. Bruno Domon and Ruedi Aebersold. Mass spectrometry and protein analysis. *science*, 312(5771):212–217, 2006.
3. Salvatore Fanali, Paul R Haddad, Colin Poole, and Marja-Liisa Riekkola. *Liquid chromatography: applications*. Elsevier, 2017.
4. Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.
5. Karimollah Hajian-Tilaki. Receiver operating characteristic (roc) curve analysis for medical diagnostic test evaluation. *Caspian journal of internal medicine*, 4(2):627, 2013.

6. Hannes L R¨ost, Uwe Schmitt, Ruedi Aebersold, and Lars Malmstr¨om. pyopenms: a python-based interface to the openms mass-spectrometry algorithm library. *Proteomics*, 14(1):74–77, 2014.
7. Robert E Schapire. Explaining adaboost. In *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, pages 37–52. Springer, 2013.
8. Hans JCT Wessels, Tom G Bloemberg, Maurice Van Dael, Ron Wehrens, Lutgarde MC Buydens, Lambert P Van den Heuvel, and Jolein Gloerich. A comprehensive full factorial lc-ms/ms proteomics benchmark data set. *Proteomics*, 12(14):2276–2281, 2012.