Protocol to analyze 1D and 2D mass spectrometry data from glioblastoma tissues for cancer diagnosis and immune cell identification

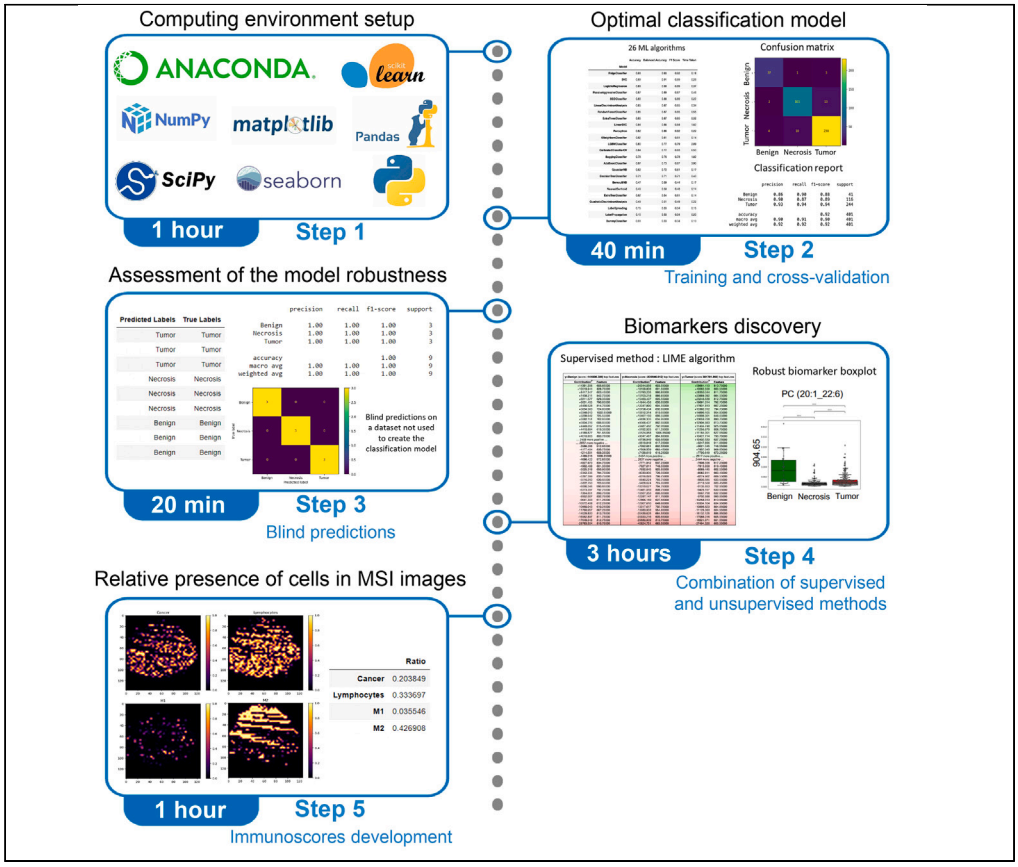
Yanis Zirem, Léa Ledoux, Michel Salzet, Isabelle Fournier

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol to analyze 1D and 2D mass spectrometry data from glioblastoma tissues for cancer diagnosis and immune cell identification



In context of cancer diagnosis-based mass spectrometry (MS), the classification model created is crucial. Moreover, exploration of immune cell infiltration in tissues can offer insights within the tumor microenvironment. Here, we present a protocol to analyze 1D and 2D MS data from glioblastoma tissues for cancer diagnosis and immune cells identification. We describe steps for training the most optimal model and cross-validating it, for discovering robust biomarkers and obtaining their corresponding boxplots as well as creating an immunoscore based on MS-imaging data.

Yanis Zirem, Léa Ledoux, Michel Salzet, Isabelle Fournier

michel.salzet@univ-lille.fr (M.S.)
isabelle.fournier@univ-lille.fr (I.F.)

Highlights

Protocol for analyzing MS data for cancer diagnosis and immune cell identification

Training and cross-validating an accurate and reliable classification model

Steps for generating statistical plots corresponding to robust biomarkers

Localizing immune cells in 2D MS data

Zirem et al., STAR Protocols 5, 103285
September 20, 2024
https://doi.org/10.1016/j.xpro.2024.103285

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol to analyze 1D and 2D mass spectrometry data from glioblastoma tissues for cancer diagnosis and immune cell identification

Yanis Zirem,[1,3,4] Léa Ledoux,[1,3] Michel Salzet,[1,2,*] and Isabelle Fournier[1,2,5,*]

[1]University Lille, Inserm, CHU Lille, U1192 - Protéomique Réponse Inflammatoire Spectrométrie de Masse - PRISM, 59000 Lille, France

[2]Institut Universitaire de France (IUF), Paris, France

[3]These authors contributed equally

[4]Technical contact

[5]Lead contact

*Correspondence: michel.salzet@univ-lille.fr (M.S.), isabelle.fournier@univ-lille.fr (I.F.)
https://doi.org/10.1016/j.xpro.2024.103285

SUMMARY

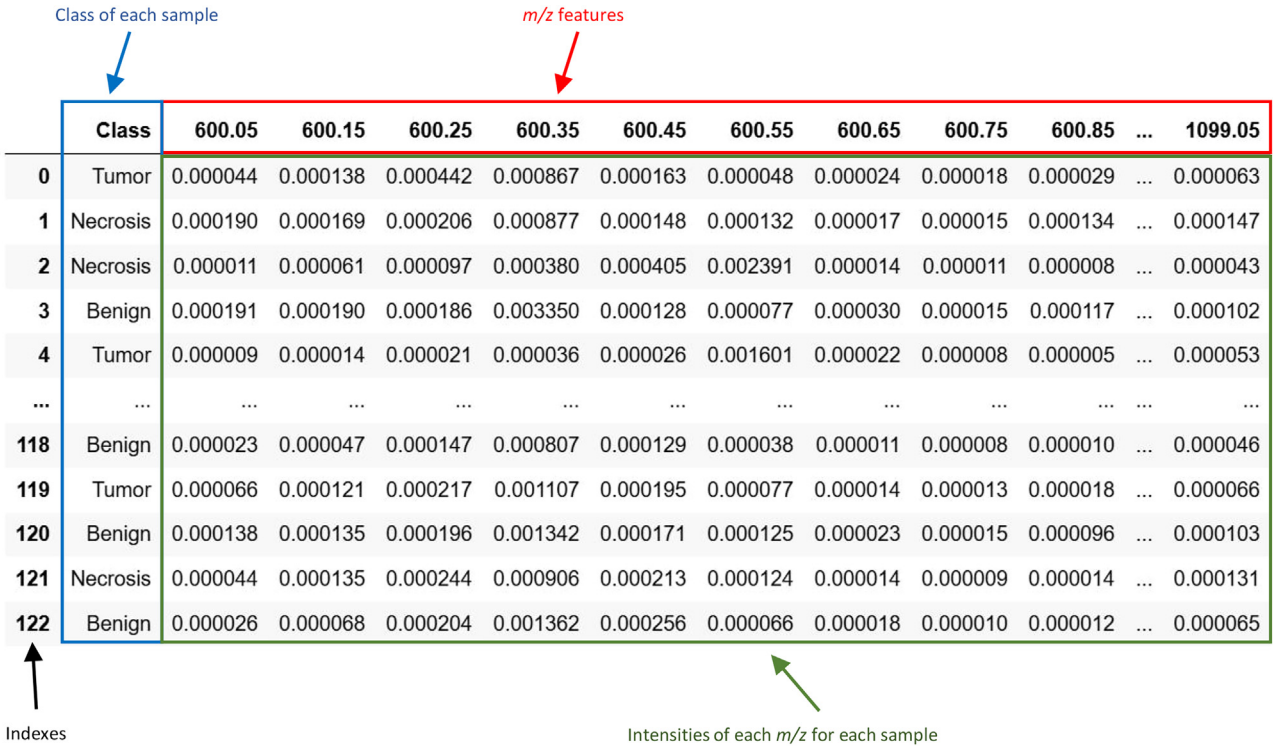
In context of cancer diagnosis-based mass spectrometry (MS), the classification model created is crucial. Moreover, exploration of immune cell infiltration in tissues can offer insights within the tumor microenvironment. Here, we present a protocol to analyze 1D and 2D MS data from glioblastoma tissues for cancer diagnosis and immune cells identification. We describe steps for training the most optimal model and cross-validating it, for discovering robust biomarkers and obtaining their corresponding boxplots as well as creating an immunoscore based on MS-imaging data. For complete details on the use and execution of this protocol, please refer to Zirem et al.[1]

BEFORE YOU BEGIN

In the realm of cancer diagnosis using mass spectrometry (MS), supervised machine learning (ML) algorithms serve as pivotal tools for building precise classification models based on labeled datasets.[2,3] The selection and optimization of an appropriate classification model can significantly influence result accuracy.[4] Additionally, the integration of non-supervised ML approaches for cross-analysis and the interpretability of outcomes are crucial considerations.[5] Therefore, we have created an artificial intelligence (AI) framework aimed at achieving the utmost accuracy and reliability in classification models for each specific dataset, while also identifying robust biomarkers through the use of both supervised and unsupervised methodologies. This pipeline was tailored by leveraging molecular fingerprints from glioblastoma tissues, enabling the development of a 1D classification model capable of distinguishing between benign, necrotic, and tumor tissues. Furthermore, a complementary pipeline, referred to as the 2D pipeline, was established to estimate the relative presence of immune cells in tissues analyzed via mass spectrometry imaging (MSI). Indeed, the composition of the tumor microenvironment (TME) is widely known to significantly influence cancer aggressiveness, type, and patient prognosis.[6–9] While these two pipelines were initially developed using SpiderMass MS and MSI data, they can be readily applied to various types of mass spectrometry data. The only prerequisite is adherence to the prescribed data matrix format outlined below.

Description of datasets

The train dataset required for the 1D pipeline should consist of a CSV file structured such that the initial column ("Class") denotes the label for each analyzed sample, while subsequent columns



	Class	600.05	600.15	600.25	600.35	600.45	600.55	600.65	600.75	600.85	...	1099.05
0	Tumor	0.000044	0.000138	0.000442	0.000867	0.000163	0.000048	0.000024	0.000018	0.000029	...	0.000063
1	Necrosis	0.000190	0.000169	0.000206	0.000877	0.000148	0.000132	0.000017	0.000015	0.000134	...	0.000147
2	Necrosis	0.000011	0.000061	0.000097	0.000380	0.000405	0.002391	0.000014	0.000011	0.000008	...	0.000043
3	Benign	0.000191	0.000190	0.000186	0.003350	0.000128	0.000077	0.000030	0.000015	0.000117	...	0.000102
4	Tumor	0.000009	0.000014	0.000021	0.000036	0.000026	0.001601	0.000022	0.000008	0.000005	...	0.000053
...	...	...	...	...	...	...	...	...	...	...	...	...
118	Benign	0.000023	0.000047	0.000147	0.000807	0.000129	0.000038	0.000011	0.000008	0.000010	...	0.000046
119	Tumor	0.000066	0.000121	0.000217	0.001107	0.000195	0.000077	0.000014	0.000013	0.000018	...	0.000066
120	Benign	0.000138	0.000135	0.000196	0.001342	0.000171	0.000125	0.000023	0.000015	0.000096	...	0.000103
121	Necrosis	0.000044	0.000135	0.000244	0.000906	0.000213	0.000124	0.000014	0.000009	0.000014	...	0.000131
122	Benign	0.000026	0.000068	0.000204	0.001362	0.000256	0.000066	0.000018	0.000010	0.000012	...	0.000065

Figure 1. An example of the required format for the train dataset (Tumor, Necrosis and Benign classes) used in the 1D pipeline

represent the features, specifically the *m/z* variables in our case. Each row of this dataset stores the relative intensities of each sample for every *m/z* feature (refer to Figure 1).

For the blind validation set, the dataset must mirror the train dataset, with samples not used in the train dataset.

Regarding the 2D pipeline, the dataset requirement involves an MSI dataset converted into imzML format using imzML converter.[10]

Example files, i.e., two csv files (training and testing datasets), a pre-trained model and an imzML MSI file, are available, in GitHub, for easy code testing.

The dataset may have undergone pre-processing before being used throughout this protocol. In our case, binning (0.1) and TIC normalization were performed beforehand. However, this is not mandatory, it's essential that MS and MSI data undergo the same pre-processing steps.

Python and package installation

Timing: 1 h

1. Visit the Anaconda website at https://www.anaconda.com/download, obtain the suitable Anaconda installer according to your computer's specifications, and proceed with the installation process.
2. Open the Anaconda-Navigator and launch the Jupyter Notebook.
3. In the Jupyter Notebook interface, run the following codes to import required packages (numpy, pandas, scikit-learn, matplotlib, scipy, seaborn, statannot, lazypredict, joblib, eli5, plotly, pyimzml and lightgbm) in their proper version.

```
> pip install numpy==1.24.3 pandas==2.0.3 scikit-learn==1.2.2 matplotlib==3.7.2 scipy==1.11.1
seaborn==0.12.2 statannot==0.2.3 lazypredict==0.2.12 joblib==1.3.1 eli5==0.13.0 pyimzml==
1.5.3 plotly==5.17.0 lightgbm
```

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
All the original code and files example have been deposited on GitHub.	Our recent paper[1]	https://github.com/yanisZirem/Protocol-to-analyse-1D-and-2D-mass-spec-data-for-cancer-diagnosis-and-immunecell-identification
MSI imzML example file has been deposited on Harvard Dataverse.	Our recent paper[1]	https://doi.org/10.7910/DVN/PRXHIJ
Software and algorithms		
Python	Python v3.11.4	RRID:SCR_008394; https://www.python.org
Anaconda	Anaconda	RRID:SCR_018317; https://www.anaconda.com/download
Jupyter Notebook	Jupyter Notebook v6.5.4	RRID:SCR_018315; https://jupyter.org
NumPy	NumPy v1.24.3	RRID: SCR_008633; https://numpy.org/
Pandas	Pandas v2.0.3	RRID:SCR_018214; https://pandas.pydata.org
scikit-learn	scikit-learn v1.2.2	RRID:SCR_002577; https://scikit-learn.org/stable/
Matplotlib	Matplotlib v3.7.2	RRID:SCR_008624; https://matplotlib.org/
SciPy	SciPy v1.11.1	RRID:SCR_008058; https://scipy.org/
Seaborn	Seaborn v0.12.2	RRID:SCR_018132; https://seaborn.pydata.org/
statannot	statannot v0.2.3	https://pypi.org/project/statannot/
lazypredict	lazypredict v0.2.12	https://pypi.org/project/lazypredict/
joblib	joblib v1.3.1	https://pypi.org/project/joblib/
Eli5	Eli5 v0.13.0	https://pypi.org/project/eli5/
PyimzML	PyimzML v1.5.3	https://pypi.org/project/pyimzML/
plotly	plotly v5.17.0	RRID:SCR_013991; https://pypi.org/project/plotly/

STEP-BY-STEP METHOD DETAILS

In-house modules download

Timing: 5 min

To use the code contained in this protocol, three in-house python modules need to be download. In fact, 3 Python modules are available on the corresponding GitHub (see KRT), containing all the functions required to run the code below.

The 3 modules are as follows:

Supervised.py: Contains functions for supervised learning for the 1D pipeline.

Unsupervised.py: Contains functions for unsupervised learning for the 1D pipeline.

MSI_immunoscoring.py: Contains functions for the 2D pipeline for immunoscoring using MS-imaging data.

Data importation and management

Timing: 15 min

This step involves importing the CSV file and visualizing the spectra contained within the dataset.

1. Import the CSV file (yourfile.csv) into Jupyter Notebook under the dataframe named 'data'. Additionally, it will print out the labels for each class along with their respective number of samples.

```
> import pandas as pd

> data = pd.read_csv('yourfile.csv') #Import the CSV file

> print("Labels :",data.Class.unique()) #Print name of the classes

> print("Number of samples: ",data.Class.value_counts()) #Print number of samples per class

> data
```

Note: The data matrix will also be displayed. Verify that the structure contains only the column named 'Class' with each labeled sample, and the *m/z* features with their corresponding intensities in rows.

2. Display either the average spectra for each labeled class or the spectra of an individual sample.
   a. Choose the colors corresponding to each class.

```
> custom_colors = {'Class1': 'green', 'Class2': 'pink'} #Custom colors for the classes
```

Note: The 'custom_colors' dictionary can be customized according to specific preferences, based on the class number and the desired color scheme. The placeholders 'Class 1' and 'Class 2' should be replaced with the actual names of each class present in the dataset.

   b. Import the 'plot_average_spectra' function from Supervised.py module to showcase the average spectra for each labeled class.

```
> from PRISM_Lib.Supervised import plot_average_spectra

> plot_average_spectra(data, class_column='Class', colors=custom_colors) #Plot the average
spectra
```

Note: The 'plot_average_spectra' function is imported from the Supervised.py module located in the PRISM_Lib folder. The folder name can be changed as required.

   c. If needed, it is also possible to display each individual spectrum from each sample of the dataset.

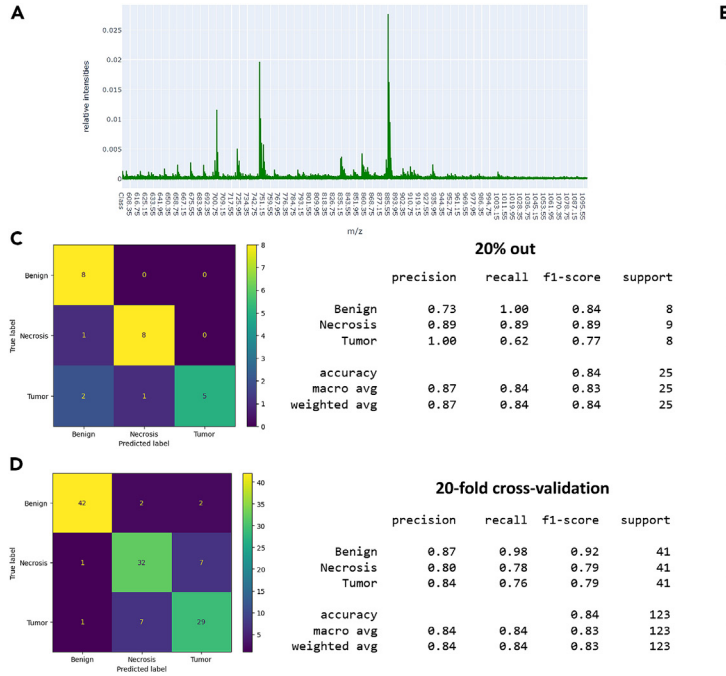
```
> from PRISM_Lib.Supervised import plot_sample_spectrum

> plot_sample_spectrum(data, 122, color='green') #Plot the spectrum of the sample at the
desired index 122
```

Note: The index number of the sample (in this case, sample 122 is used) from which the spectra will be displayed should be adjusted according to preferences. Additionally, the color can be customized as desired (here, green was chosen) (see Figure 2A).

Obtainment of the optimal model

Timing: 40 min

3. Evaluate 26 machine learning algorithms from Lazypredict (https://lazypredict.readthedocs.io/en/latest/), depending of scikit-learn, to determine the most suitable model for the dataset.



B

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
Perceptron	0.88	0.88	None	0.88	0.53
LGBMClassifier	0.84	0.84	None	0.84	6.11
PassiveAggressiveClassifier	0.80	0.81	None	0.80	0.50
LinearSVC	0.80	0.81	None	0.80	1.51
RidgeClassifierCV	0.80	0.80	None	0.79	0.37
RidgeClassifier	0.80	0.80	None	0.79	0.29
ExtraTreeClassifier	0.80	0.80	None	0.80	0.33
ExtraTreesClassifier	0.80	0.80	None	0.80	0.56
SGDClassifier	0.76	0.77	None	0.75	0.31
CalibratedClassifierCV	0.76	0.77	None	0.75	4.35
RandomForestClassifier	0.76	0.76	None	0.76	1.38
BaggingClassifier	0.72	0.73	None	0.71	3.19
KNeighborsClassifier	0.72	0.72	None	0.72	0.39
DecisionTreeClassifier	0.72	0.71	None	0.72	0.55
LogisticRegression	0.68	0.69	None	0.67	1.20
LinearDiscriminantAnalysis	0.68	0.69	None	0.68	3.55
NuSVC	0.68	0.68	None	0.68	0.98
SVC	0.68	0.68	None	0.67	0.96
GaussianNB	0.68	0.67	None	0.66	0.30
NearestCentroid	0.60	0.59	None	0.58	0.28
BernoulliNB	0.56	0.55	None	0.53	0.35
AdaBoostClassifier	0.40	0.42	None	0.35	10.23
QuadraticDiscriminantAnalysis	0.40	0.41	None	0.40	8.88
DummyClassifier	0.32	0.33	None	0.16	0.28
LabelSpreading	0.32	0.33	None	0.16	0.28
LabelPropagation	0.32	0.33	None	0.16	0.31

Figure 2. Obtainment of the optimal classification model specific of each dataset

(A) Example of a spectra from one sample.

(B) Example of a table of 26 classifiers that were trained and evaluated, showing their accuracies, balanced accuracy, ROC AUC, F1 Score and time taken.

(C and D) Example of classification report and confusion matrix of the optimal algorithm with the 20% out set and after 20-fold cross-validation, obtained for the classification model that distinguish benign, necrotic and tumor tissues.

a. Train the 26 algorithms and calculates the accuracy of each classifier for the dataset. The accuracy is calculated by performing a 20% validation split. Indeed, the training dataset is split into two sets: a train set (80%) and a test set (20%).

```
> from PRISM_Lib.Supervised import Train_models

>models,predictions=Train_models(data, target_column='Class', test_size=0.2, random_state=1)
#Train different ML models (26 algorithms)

> models
```

Note: The outcome will be a table with the 26 classifiers that were tested (Figure 2B), along with their respective accuracy, balanced accuracy, F1-score and the time taken for each evaluation.

4. Find and build the optimal model based on the F1-score.

```
> from PRISM_Lib.Supervised import Find_and_build_best_model

> best_model_name, best_model_pipeline = Find_and_build_best_model(data, models, specific_
model=None)
```

Note: Furthermore, note that if desired, a particular algorithm can be specified instead of selecting the best one by replacing 'None' for 'specific_model' with the name (in strings) of the desired algorithm.

Note: Various evaluation parameters are obtained for each model, such as recall, accuracy, precision and F1-score. Selection of the optimal model is based on the F1-score. This

combines precision and recall and is known as the harmonic mean. This score provides a more balanced measure of performance across classes.

5. Obtain the corresponding confusion matrix and classification report (Figure 2C).
   a. Use the 'confusion_matrix_scores_classification_report' function to present the confusion matrix and classification report of the optimal classification model constructed.

```
> from PRISM_Lib.Supervised import confusion_matrix_scores_classification_report

> confusion_matrix_scores_classification_report (best_model_pipeline, data) #Display confusion matrix and classification report for validation data
```

Note: The 'ConfusionMatrixDisplay.from_estimator' is a pre-existing function within the sklearn package's metrics module designed to exhibit the confusion matrix for a split train/test set.

   b. Perform the k-fold cross validation of the model to assess his robustness and display the corresponding confusion matric and classification report (Figure 2D).

```
> from PRISM_Lib.Supervised import cross_validate_and_report

> cross_validate_and_report(best_model_pipeline, data) #Perform cross-validation and report results
```

Note: Please note that the number of iterations can be specified in 'n_splits' (in Supervised.py module); here, a 20-fold cross-validation was employed to thoroughly evaluate the robustness and generalization capabilities of the constructed classification model. Furthermore, each accuracy is presented alongside the mean and standard deviation values obtained for a comprehensive understanding of the performance variability across the folds.

6. Save the classification model as a .pkl file for possible future use.

```
> import joblib

> joblib.dump(best_model_pipeline, "name_model.pkl") #Save the trained optimal model
```

Note: If used without any changes, the classification model will be created as a name_model.pkl file.

Blind predictions

Timing: 20 min

Blind predictions are obtained using samples not included in the creation of the previous optimal classification model. This blind test set is crucial for evaluating the model's robustness. During the blind validation step, a dataset similar to the training set (but with new samples) is employed.

7. Upload the blind validation dataset as a CSV file (validation_blind.csv) under the dataframe named 'val'.

```
> val = pd.read_csv("validation_blind.csv") #Load unknown data for blind predictions

> val_id = val

> val = val.drop(["Class"], axis=1)
```

8. Make the blind predictions.

```
> validation = best_model_pipeline.predict(val) #Validate the model on unknown data

> validation
```

9. Create a dataframe to compare both predicted labels (from validation) and true labels (specified by the "Class" column in the dataset) (Figure 3A).

```
> df = pd.DataFrame(validation)

> df["True Labels"] = val_id["Class"]

> df = df.rename(columns={0 : "Predicted Labels"})

> df #Display a DataFrame that compare predicted and true labels
```

10. Display the classification report to obtain the accuracy, which represents the correct prediction rate of the classification model, in addition to the corresponding confusion matrix (Figures 3B and 3C).

A

	Predicted Labels	True Labels
0	Tumor	Tumor
1	Tumor	Tumor
2	Tumor	Tumor
3	Necrosis	Necrosis
4	Necrosis	Necrosis
5	Necrosis	Necrosis
6	Benign	Benign
7	Benign	Benign
8	Benign	Benign

B

	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	3
Necrosis	1.00	1.00	1.00	3
Tumor	1.00	1.00	1.00	3
accuracy			1.00	9
macro avg	1.00	1.00	1.00	9
weighted avg	1.00	1.00	1.00	9

C

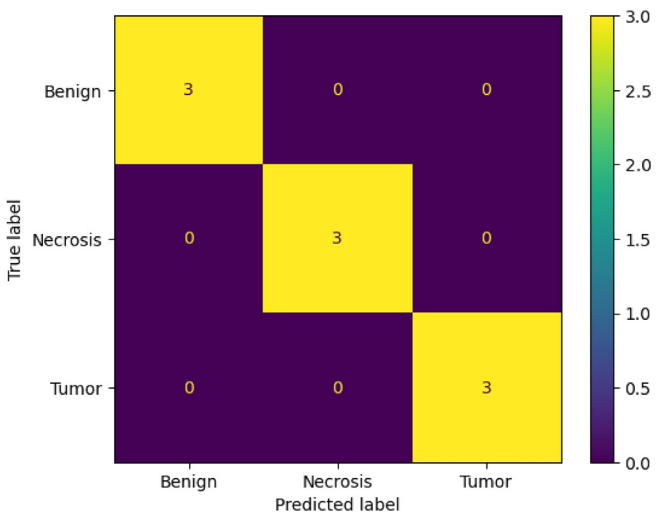


Figure 3. Example of outcomes of blind predictions obtained by the classification model

(A) Overview table providing a comparison between the predicted and true labels.

(B and C) Classification report and confusion matrix displaying various metrics, including accuracy, for the blind predictions conducted on tissue not used in the creation of classification model.

```
> from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrix
Display

> ConfusionMatrixDisplay.from_estimator(best_model_pipeline, val, val_id["Class"])

> print(classification_report(validation, val_id["Class"])) #Display the classification
report and the confusion matrix for the validation set
```

Supervised model explicability

Timing: 15 min

Supervised explicability of the classification model is achieved through the use of the LIME algorithm.[11]

11. Use the 'eli5_feature_importance' function using eli5 package (https://eli5.readthedocs.io/en/latest/overview.html) to identify the *m/z* features that contribute positively or negatively to predicting each class in the classification model (Figure 4A).

```
> from PRISM_Lib.Supervised import eli5_feature_importance

> sample_contribution = eli5_feature_importance(best_model_pipeline, data, top_fea-
tures=40) #Get feature importance thanks to LIME algorithm
```

Note: Here, it will enable to obtain the top contributing 40 contributing *m/z* features for each class, although this number can be specified as desired.

A

y=Benign top features		y=Necrosis top features		y=Tumor top features	
Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature
+59.258	915.55	+48.248	1017.85	+30.680	619.35
+53.057	916.55	+47.110	1019.85	+28.020	621.35
+49.043	632.45	+44.748	706.25	+28.018	773.55
+46.047	699.55	+43.363	1018.85	+27.512	755.65
+43.500	835.55	+40.681	666.15	+27.454	871.65
+43.365	610.45	+39.900	707.25	+26.950	620.35
+42.809	699.85	+39.688	1016.85	+26.841	674.55
+41.895	747.75	+38.543	1047.85	+26.704	725.65
+40.080	790.65	+38.261	669.55	+24.919	725.85
+38.696	628.35	+37.688	1085.85	+24.316	1017.65
+38.255	857.55	+37.486	1046.85	+24.250	673.55
+37.479	937.55	+37.079	747.65	+24.246	725.55
+36.720	918.65	+36.622	832.75	+23.700	1033.65
+36.317	788.65	+36.219	878.05	+22.299	652.15
+36.141	604.35	+36.107	720.95	+22.246	862.85
+36.012	789.65	+35.865	722.15	+21.184	785.75
+35.908	965.55	+35.509	872.05	+20.850	786.75
+35.787	1063.05	+34.742	879.05	+20.541	741.65
+34.853	1063.55	+34.658	904.05	+20.333	688.55
+34.352	759.55	+34.150	1020.85	+19.991	723.75
+34.261	748.05	+34.024	907.65	*... 2462 more positive ...*	
+34.082	1064.05	+33.818	721.95	*... 2499 more negative ...*	
+33.843	747.85	+33.809	963.85	-19.996	917.55
+33.715	748.85	+33.044	639.65	-20.012	1073.85
+33.250	858.55	+33.011	807.65	-20.047	918.05
+32.705	800.65	+32.749	1086.85	-20.154	919.05
+32.651	747.55	+32.278	666.05	-20.847	1063.95
... 1895 more positive ...		+32.158	665.55	-21.149	1057.95
... 3066 more negative ...		+32.028	990.75	-21.264	1063.15
-32.433	687.55	+32.024	906.65	-21.641	870.35
-33.291	737.45	+31.790	708.25	-21.835	1064.15
-33.431	783.65	*... 2491 more positive ...*		-22.353	1088.95
-34.039	952.65	*... 2470 more negative ...*		-22.599	704.25
-35.500	711.55	-32.079	755.65	-22.660	779.55
-36.886	981.55	-32.440	748.75	-23.134	1019.85
-37.558	771.65	-34.241	862.85	-23.766	1087.95
-39.815	753.45	-35.000	871.65	-24.000	<BIAS>
-40.956	981.65	-35.071	674.55	-25.813	914.65
-41.141	982.65	-40.148	620.35	-25.873	728.25
-49.950	698.45	-41.301	619.35	-29.977	708.25
-57.506	697.45	-42.000	604.35	-36.181	707.25
-114.000	<BIAS>	-113.000	<BIAS>	-39.632	706.25

B

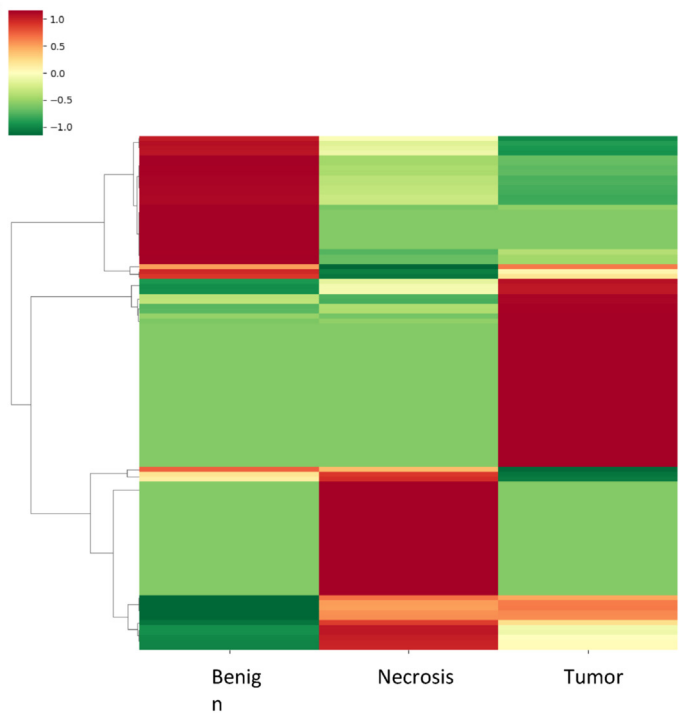


Figure 4. Example of the supervised and unsupervised marker discovery result

(A) Display of the top 120 *m/z* features contributing positively (in green) or negatively (in red) to the classification of each class in the model.

(B) Heatmap representing picked peaks for three distinct classes. The heatmap illustrates the intensity of each ion, with red denoting overexpression and green indicating under-expression.

12. Save the contributions of all *m/z* features for predicting each class, rather than only the top 40 *m/z* variables, as a CSV file ('LIME_mz_features.csv').

```
> from PRISM_Lib.Supervised import save_contributions

> save_contributions("LIME_mz_features.csv", best_model_pipeline, data)
```

Unsupervised markers discovery

Timing: 30 min

13. Generate the heatmap corresponding to the dataset.
    a. Perform peak picking, with a signal-to-noise ratio (S/N) of 10, on the entire dataset to isolate *m/z* features corresponding to real peaks, filtering out instrument noise.

```
> from PRISM_Lib.Supervised import peak_picking

> data_peak_picked = peak_picking(data, min_sn=10) #Peak picking with S/N > 10 and return the peak in Dataframe

> data_peak_picked
```

Note: The S/N threshold can be specified as desired.

    b. Use the 'create_heatmap' function to visualize the heatmap, depicting the under or over-expressed *m/z* features identified through peak picking for each (Figure 4B).

```
> from PRISM_Lib.Supervised import create_heatmap

> create_heatmap(data_peak_picked,cmap='RdYlGn_r',distance_metric='cosine', z_score=0) #Plot the clustering_heatmap based on the peak picking data
```

Note: The 'cmap' dictionary can be customized according to specific preferences, based on the desired color scheme, depending of the seaborn library (https://seaborn.pydata.org/tutorial/color_palettes.html) (here, the RdYlGn_r is used).

Note: The clustering heatmap can also be obtain on the original data by replacing 'data_peak_picked' by 'data'.

14. Identify the significant *m/z* features thanks to an automated Kruskal-Wallis statistic test for identifying *m/z* features that are significantly present or absent in each class.
    a. Use the 'significant_features' function to retrieve not only the count of significant variables but also the list of significant *m/z* features.

```
> from PRISM_Lib.Unsupervised import significant_features

> significant_mz_values = significant_features(data_peak_picked, alpha=0.05)

> print("There are", len(significant_mz_values), "significants features") #Print the number of significant m/z values found

> print("list of Significant_mzs :", significant_mz_values)significant_mz_values #Display the list of significant m/z values.
```

Note: The Kruskal-Wallis test is conducted on the ions identified through peak picking ('data_peak_picked'). It can of course be replaced by "data" to allow the automated Kruskal_Wallis test to be performed on the entire data set.

Note: This function will allow identification of only *m/z* features with a p-value lower than or equal to 0.05. Of course, this threshold can be specified as desired through change of the alpha value in the 'significant_features' argument. In addition, the Bonferroni correction is employed to maintain the statistical integrity of analyses across a multitude of tests, thereby ensuring the reliability of conclusions and reducing the likelihood of false discoveries.

b. Save the list of significant *m/z* variables in a CSV file.

```
> pd.DataFrame(significant_mz_values, columns=['Significant_mzs']).to_csv('Significant_mzs.csv', index=False)
```

Robust biomarkers discovery

Timing: 1 h

This part is carried out manually since, as of our current knowledge, automation is not feasible.

15. Refine the significant *m/z* features list by filtering out isotopes to isolate real peaks. This step involves manual inspection across all obtained MS spectra. Indeed, in the context of significant m/z features, a "real peak" means that the peak corresponding to the *m/z* ions is not an isotope of another ion and is not part of the instrument's noise; in conclusion, this peak really corresponds to a molecule.
16. Integrate results from both unsupervised and supervised approaches. Indeed, for a potential biomarker to become a robust biomarker, it must show a positive contribution in cases of over-expression and a negative contribution in case of under-expression, consistently across the same tissue type.
17. Modify and save all the real robust biomarkers in the previous created CSV file ('Significant_mzs.csv'), for generating their respective statistical plots.

Statistical plots display

Timing: 30 min

This step involves generating statistical plots corresponding to all robust biomarkers stored in a CSV file in the previous step ('Significant_mzs.csv').

18. Upload the CSV file containing the robust biomarkers as a 'robust_biomarkers' dataframe.

```
> robust_biomarkers = pd.read_csv('significant_mz_values.csv') #Upload the CSV file containing all the robust biomarkers

> robust_biomarkers
```

19. Transform this dataframe into a list named 'liste_mz_robust'.

```
> liste_mz_robust = []

> for i in robust_biomarkers["Column_name"]:

liste_mz_robust.append(i)

> liste_mz_robust
```

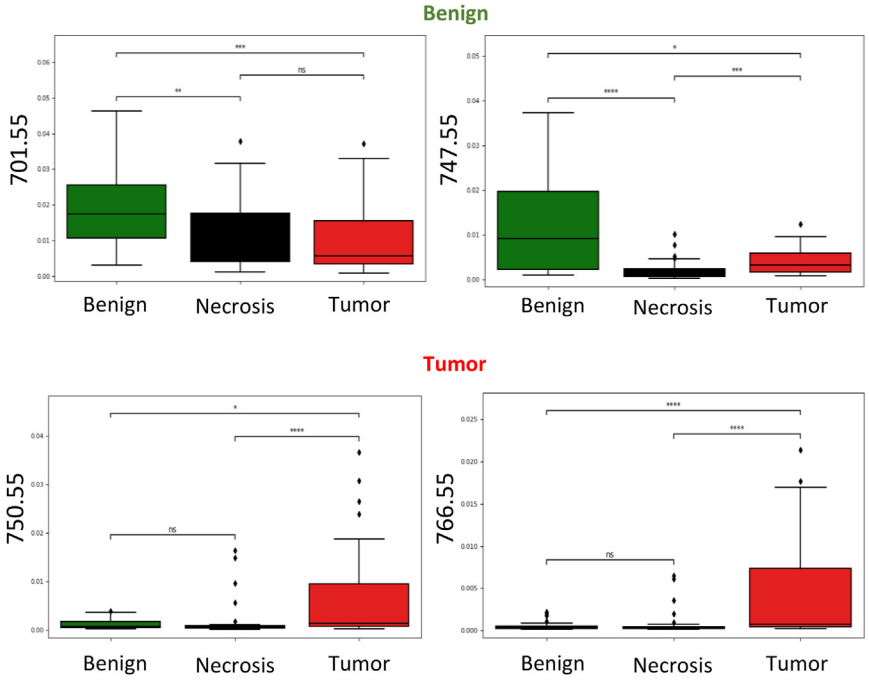


Figure 5. Specific boxplots of 4 confident lipid biomarkers obtained for benign and tumor tissue

* < 0.05; ** < 0.01; *** < 0.001; **** < 0.0001.

Note: The placeholder 'Column_name' in the code needs to be replaced with the actual name of the column in the CSV file.

20. Visualize the corresponding boxplots by using the 'boxplot_significant_features' function.

```
> from PRISM_Lib.Unsupervised import boxplot_significant_features

> boxplot_significant_features(data, liste_mz_robust, class_colors=custom_colors, show_
scatter=False)
```

Note: The colors assigned to each class match those selected for the spectra visualization, using the 'custom_colors' dictionary.

Note: The p-value annotation legend is as follow:

*$p < 0.05$.

**$p < 0.01$.

***$p < 0.001$.

****$p < 0.0001$.

21. Visualize the corresponding violin plots by using the 'violinplot_significant_features' function (Figure 5).

```
> from PRISM_Lib.Unsupervised import violinplot_significant_features

>violinplot_significant_features(data, liste_mz_robust, class_colors=custom_colors, show_
scatter=False)
```

Note: The 'violinplot_significant_features' function mirrors the 'boxplot_significant_features' function, differing solely in the substitution of 'sns.boxplot' with 'sns.violinplot'.

22. Moreover, if desired, it is possible to display boxplot and violin plot one by one for each significant *m/z* feature.

```
> from PRISM_Lib.Unsupervised import one_box_plot

> mz = '701.55' #specify m/z feature (ion)

> one_box_plot(data, mz, class_colors = custom_colors, show_scatter=False) #Plot boxplot of
a specific feature
```

Note: The mass of the significant ions is specified in the 'mz' variable (e.g., 701.55) and can be modified as needed.

Note: It is also possible to visualize the violin plot one by one by simply switching from the previous function 'sns.boxplot' to 'sns.violinplot'.

23. It is also possible to display these boxplots or violin plots as scatter plots.
    To display boxplots or violin plots as scatter plots, simply change False to True in the parameters 'show_scatter'.

Two-dimensional analysis pipeline

Timing: 1 h

This step involves using a pre-existing model to determine the relative presence of each class in a tissue image obtained through mass spectrometry imaging (MSI). In our scenario, the model was employed to generate immunoscores, using a classification model built on MS fingerprints of immune cells such as M1- and M2-like macrophages, lymphocytes, and glioblastoma cancerous cell line (NCH82). Note that this code can be used for all classification models based on various classes of tissues/cells to ascertain the distribution or localization of each within an MSI tissue.

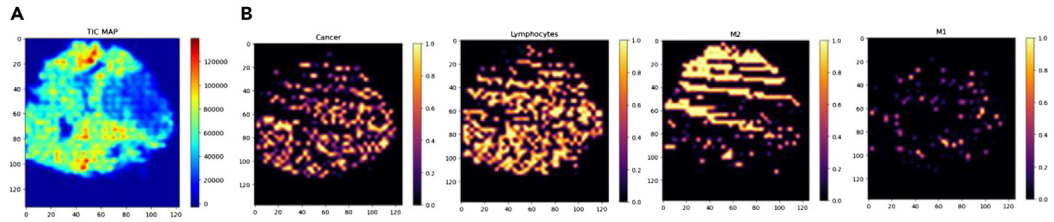
24. Use the 'generate_tic_map' function to upload the .imzML file (MSI dataset) and to display the corresponding total ion count (TIC) map.

```
> from PRISM_Lib.MSI_immunoScoring import generate_tic_map

> imzml_file = "yourfile.imzML" #the .ibd file should be located in the same folder as imzML file

> generate_tic_map(imzml_file, mzs_range=(600, 1000), sigma=0.5, new_resolution=3)
```

Note: The modifiable parameters correspond to the enhance of resolution for 'new_resolution' and to a smoothing for 'sigma', by changing, as needed, 'mass range', 'sigma' and 'new_resolution' parameters (Figure 6A).

Note: The corresponding .ibd file need to be in the same folder than the .imzML file. In addition, the colorbar used can be changed (here, jet was used) by replacing 'jet' in the 'cmap' parameter in MSI_immunoScoring.py module.

25. Upload a pre-existing classification model 'name_model.pkl' and generate label maps for each class contained in the classification model (Figure 6B) by using the 'generate_label_maps' function.



C

	Ratio
Cancer	0.203849
Lymphocytes	0.333697
M1	0.035546
M2	0.426908

Figure 6. Determining the relative presence of each class within a tissue image acquired through mass spectrometry imaging

(A) Total Ion Count (TIC) map generated for one glioblastoma tissue to visualize the intensity of detected peaks across the tissue.

(B) Representative label maps generated for each class within the classification model (including here cancer cells, lymphocytes, and M1- and M2-like macrophages).

(C) Percentage ratios depicting the distribution of each cell type across the entire tissue sample.

```
> from PRISM_Lib.MSI_immunoScoring import generate_label_maps

> model_file = "name_model.pkl"

> generate_label_maps(imzml_file, model_file, mass_range=(600, 1000), max_intensity_size=4000,
sigma=0.5, new_resolution=3, real_pixel_threshold=25000)model = joblib.load('name_model.
pkl')
```

Note: At this stage, binning is performed by determining the desired number of peaks within the specified mass range. For example, a mass range of 600–1000 with 4000 features (max_ intensity_size) corresponds to a bin size of 0.1.

Note: Select identical parameters (sigma, new resolution, binning) as those used for generating the TIC map and model upload. In addition, at this step, an intensity threshold is used in order to obtain predictions of cell presence only in the tissue and not on the slide around it (this threshold is chosen based on the TIC map, here 25000 is used).

Note: The colorbar can be customized using the 'cmap' parameters in the 'generate_label_ maps' function, in this case, 'inferno' was chosen.

26. Calculate the ratio of each class in the tissue (Figure 6C) by using the 'calculate_label_ratios' function.

```
> from PRISM_Lib.MSI_immunoScoring import calculate_label_ratios

> df_ratios = calculate_label_ratios(imzml_file, model_file)

> df_ratios
```

EXPECTED OUTCOMES

The protocol for 1D MS data provides an effective method for producing accurate classification models to enhance cancer diagnosis using mass spectrometry technology. Additionally, this protocol facilitates the identification of robust biomarkers through both supervised and unsupervised methods, which can be further used for discovering potential therapies. The pipeline is designed to improve the interpretability and reliability of results in cancer research.

The protocol focused on 2D MS data demonstrates the potential of mass spectrometry to detect and localize specific cells in imaged tissue, including cancerous, healthy, and immune cells, as exemplified in our paper.[1]

Both 1D and 2D protocols offer a straightforward and user-friendly approach, using Python scripts to generate comprehensive visualizations of large amounts of mass spectrometry data, focusing on lipids, metabolites, or proteins.

LIMITATIONS

Variability in MS data quality can affect the reliability of the protocol. Careful inspection of manual steps in the biomarker discovery process is mandatory. Furthermore, adapting the protocol to different types of MS data may require additional optimization to ensure consistent and accurate results.

TROUBLESHOOTING

It is essential to consider the important statements and notes that accompany each step described in the method details.

Problem 1

The attempt to evaluate 26 machine learning algorithms to develop the optimal classification model was unsuccessful because the script failed to recognize the input CSV file (related to step 2–1).

Potential solution

Ensure the CSV file structure matches the required format.

Problem 2

Obtainment of a classification model with poor performance (related to step 2–4).

Potential solution

Experiment with additional machine learning algorithms beyond the 26 available in the scikit-learn library, and adjust the parameter settings.

Problem 3

Training a classification model that achieves high accuracy but produces inconsistent results in blind predictions (related to step 3).

Potential solution

Verify MS data quality and consider the input of additional preprocessing steps.

Problem 4

Inconsistency between biomarkers obtained in a supervised way with LIME and those obtained in an unsupervised manner with Kruskall-Wallis (related to step 6).

Potential solution

It is then possible to use other techniques such as Shapley value or the permutation algorithm.

Furthermore, it is important to remember that the effectiveness of the LIME algorithm relies on the accuracy of the underlying classification model. If the model's accuracy is low, the reliability of the biomarkers decreases. To improve this, consider fine-tuning the model to boost its accuracy or explore only unsupervised learning methods via Kruskall-Wallis method.

Problem 5

Variation between 1D data (used to create the classification model) and 2D data (used for localization).

Potential solution

1D and 2D data must undergo the same pre-processing steps no matter which.

Cell localization results can be validated by immunohistochemistry.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Isabelle Fournier (isabelle.fournier@univ-lille.fr).

Technical contact

Questions about the technical specifics of performing the protocol should be directed to and will be answered by the technical contact, Yanis Zirem (yanis.zirem@univ-lille.fr).

Materials availability

This study did not generate new unique reagents.

Data and code availability

All original code has been deposited at GitHub ((https://github.com/yanisZirem/Protocol-to-analyse-1D-and-2D-mass-spec-data-for-cancer-diagnosis-and-immunecell-identification) in addition to files examples (csv files and imzmL file). The DOIs are available in the key resources table. If you have any questions or feedback, please contact yanis.zirem2016@gmail.com.

Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

ACKNOWLEDGMENTS

This research was supported by grants from Ministère de l'Enseignement Supérieur et de la Recherche (MESR), Inserm specific funding for SpiderMass project (I.F.), Inserm and Institut Universitaire de France (I.F.), and the National Agency of Research (ANR) on Deadpool (ANR-22-CE29-0016) (I.F.) and Click & Detect (ANR-23-CE29-0018) (M.S) support. Y.Z., PhD, is funded by ANR Click & Detect (ANR-23-CE29-0018). L.L., PhD, was cofunded by the University of Lille Excellence Initiative, Région Haut de France (EU FEDER funds), and OCR. We thank OCR for its contribution to the project.

AUTHOR CONTRIBUTIONS

Conceptualization, Y.Z.; writing, Y.Z. and L.L.; review and editing, M.S. and I.F.; funding acquisition, M.S. and I.F.; supervision, M.S. and I.F.

DECLARATION OF INTERESTS

M.S. and I.F. are inventors on a patent (priority number WO2015IB57301 20150922) related to the mass spectrometry technology used to develop this AI pipeline.

REFERENCES

1. Zirem, Y., Ledoux, L., Roussel, L., Maurage, C.A., Tirilly, P., Le Rhun, É., Meresse, B., Yagnik, G., Lim, M.J., Rothschild, K.J., et al. (2024). Real-time glioblastoma tumor microenvironment assessment by SpiderMass for improved patient management. Cell Rep. Med. *5*, 101482. https://doi.org/10.1016/j.xcrm.2024.101482.

2. Lee, E.S., and Durant, T.J.S. (2022). Supervised machine learning in the mass spectrometry laboratory: A tutorial. J. Mass Spectrom. Adv. Clin. Lab *23*, 1–6. https://doi.org/10.1016/j.jmsacl.2021.12.001.

3. Ifa, D.R., and Eberlin, L.S. (2016). Ambient Ionization Mass Spectrometry for Cancer Diagnosis and Surgical Margin Evaluation. Clin. Chem. *62*, 111–123. https://doi.org/10.1373/clinchem.2014.237172.

4. Seddiki, K., Saudemont, P., Precioso, F., Ogrinc, N., Wisztorski, M., Salzet, M., Fournier, I., and Droit, A. (2020). Cumulative learning enables convolutional neural network representations for small mass spectrometry data classification. Nat. Commun. *11*, 5595. https://doi.org/10.1038/s41467-020-19354-z.

5. Linardatos, P., Papastefanopoulos, V., and Kotsiantis, S. (2020). Explainable AI: A Review of Machine Learning Interpretability Methods. Entropy *23*, 18. https://doi.org/10.3390/e23010018.

6. Jayasingam, S.D., Citartan, M., Thang, T.H., Mat Zin, A.A., Ang, K.C., and Ch'ng, E.S. (2019). Evaluating the Polarization of Tumor-Associated Macrophages Into M1 and M2 Phenotypes in Human Cancer Tissue: Technicalities and Challenges in Routine Clinical Practice. Front. Oncol. *9*, 1512. https://doi.org/10.3389/fonc.2019.01512.

7. Vidyarthi, A., Agnihotri, T., Khan, N., Singh, S., Tewari, M.K., Radotra, B.D., Chatterjee, D., and Agrewala, J.N. (2019). Predominance of M2 macrophages in gliomas leads to the suppression of local and systemic immunity. Cancer Immunol. Immunother. *68*, 1995–2004. https://doi.org/10.1007/s00262-019-02423-8.

8. Michiba, A., Shiogama, K., Tsukamoto, T., Hirayama, M., Yamada, S., and Abe, M. (2022). Morphologic Analysis of M2 Macrophage in Glioblastoma: Involvement of Macrophage Extracellular Traps (METs). Acta Histochem. Cytochem. *55*, 111–118. https://doi.org/10.1267/ahc.22-00018.

9. Zhang, H., Luo, Y.-B., Wu, W., Zhang, L., Wang, Z., Dai, Z., Feng, S., Cao, H., Cheng, Q., and Liu, Z. (2021). The molecular feature of macrophages in tumor immune microenvironment of glioma patients. Comput. Struct. Biotechnol. J. *19*, 4603–4618. https://doi.org/10.1016/j.csbj.2021.08.019.

10. Römpp, A., Schramm, T., Hester, A., Klinkert, I., Both, J.-P., Heeren, R.M.A., Stöckli, M., and Spengler, B. (2011). imzML: Imaging Mass Spectrometry Markup Language: A Common Data Format for Mass Spectrometry Imaging. In Data Mining in Proteomics Methods in Molecular Biology, M. Hamacher, M. Eisenacher, and C. Stephan, eds. (Humana Press), pp. 205–224. https://doi.org/10.1007/978-1-60761-987-1_12.

11. Ribeiro, M.T., Singh, S., and Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Preprint at arXiv. https://doi.org/10.48550/arXiv.1602.04938.