POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Machine-Learning Techniques for the Diagnosis of COVID-19 from Exhaled-Breath Mass Spectra

Supervisors

Prof. Giovanni SQUILLERO

Prof. Nicolò BELLARMINO

Candidate

Matteo SERRA

October 2023

Abstract

The emergence of COVID-19 caused by SARS-CoV-2 has created a global health crisis, necessitating rapid and non-invasive diagnostic methods. Traditional approaches like RT-PCR have limitations, so this study aims to use Machine Learning to detect COVID-19 from patients' breath mass spectra. The study began by creating a dataset of mass spectra stored in .ASC files. These files contain multiple acquisitions, each corresponding to a mass spectrum. The first phase aimed to identify the zones where the mass spectrometer is stable, we considered flat the zones with first derivative inside a tolerance guard, then standard deviations within the plateau were computed, and the acquisitions with the lowest values were selected and mass spectra were extracted. After this preliminary phase we got four datasets, one for each range. Data exploration revealed measurement bias, where acquisitions from the same day or close days were closer together. Normalization techniques such as TIC and Krypton normalization were applied to address this. High dimensionality issues were mitigated using feature selection methods like PCA and gradient boosting. The lack of data samples were solved using a brand new data augmentation technique that used the combination of different ranges acquisitions of the same patient. To augment the signal quality we applied signal pre-processing methods to the spectra. Those included baseline correction with an ALS algorithm, a Savitzky-Golay smoothing filter and a peak alignment procedure. To detect and discard outliers a z-score filter and a comparison with the NIST krypton isotopic ratios are applied. A Convolutional Autoencoder (CAE) was designed as a feature extractor, trained for 20 epochs with various layers and regularization techniques. In particular we used a padded noised version of the signal as training set and the aim of the net is to reconstruct the denoised version. The choice of the l2 regularization was a key point for the realization of the CAE. The net architecture involved a basic block made by a 1D convolutional layer, a max pooling or up sampling layer (depending if we are in the encoder or decoder part) and a batch normalization layer with different kernel and pooling size. CAE achieved satisfactory performance in terms of signal reconstruction and its encoder part will be used as feature extractor with a dimensionality reduction of a factor of 6. Several Machine Learning models, including KNN, RF, LR, XB, SVM, and an ensemble model, were employed in a 10-fold cross-validation protocol with stratification and outlier reduction. Variance thresholding, PCA or xgboost addressed features reduction and oversampling addressed class imbalance. Experiments revealed range 2 as the most discriminating for classification. With PCA as feature selection and TIC normalization, accuracy and F1-score improved from 82% and 68% to 93% and 87%, respectively. Expanding the dataset to the whole mass range led to 95% accuracy and 92% F1-score. CAE improved results, achieving 92% balanced accuracy and 90% F1-score by mitigating day bias. This study introduced a framework for COVID-19 detection from breath mass spectra using Machine Learning and a CAE to handle high dimensionality. Range 2 was found to be the most informative, and the proposed method achieved a 95% accuracy and 92% F1-score with a portable mass spectrometer, representing an improvement over invasive methods.

Acknowledgements

I would like to express my recognition and gratefulness to my thesis advisor Squillero Giovanni for his unwavering support and insights throughout this research journey. I am deeply appreciative of Nicolò Bellarmino for his significant contribution and help to this work. I also extends my sincere thanks to the NanoTech Analysis S.r.l for their support in providing access to resources, data and domain insights that added depth and relevance to this study. In particular I want to thank Raffaele Correale and Michele Castelluzzo, since their partnership has been pivotal to understand the deep field of mass spectrometry. This thesis would not have been possible without the encouragement, understanding, and patience of my family and friends. I am profoundly grateful for their unwavering support, which sustained me throughout the challenges of this endeavor.

Alla mia famiglia, perchè a loro devo tutto.

Table of Contents

List of Tables VII

List of Figures VIII

Acronyms XII

1 Introduction 1
- 1.1 Problem Description . . . 1
- 1.2 Motivation and Objectives . . . 2
- 1.3 Mass spectrometry . . . 3
- 1.4 COVID-19 . . . 4
  - 1.4.1 Test Methods . . . 4
  - 1.4.2 Long COVID . . . 5
- 1.5 Organization . . . 6

2 Preliminary Dataset Studies 8
- 2.1 ASC files . . . 8
- 2.2 Data Collection . . . 9
- 2.3 Unsupervised Learning . . . 11
  - 2.3.1 T-SNE . . . 11
- 2.4 Data Exploration . . . 11
  - 2.4.1 Class labels distribution . . . 16
  - 2.4.2 Data skew . . . 17
- 2.5 Curse of Dimensionality . . . 18

3 Background 20
- 3.1 Machine Learning and Statistical Learning . . . 20
  - 3.1.1 KNN Classifier . . . 21
  - 3.1.2 Random Forest Classifier . . . 21
  - 3.1.3 Logistic Regression . . . 22
  - 3.1.4 Gradient Boosting Classifier . . . 23

3.1.5 SVM 23
3.1.6 Voting Classifier 24
3.2 Deep Learning and Neural Network 24
3.2.1 Convolutional Autoencoder 25

4 Method 28
4.1 Signal Processing 28
4.1.1 Signal and Noise 28
4.1.2 Baseline correction 29
4.1.3 Peak Alignment 31
4.2 Normalization 31
4.2.1 Standard Scaler 32
4.2.2 Robust Scaler 32
4.2.3 TIC normalization 32
4.2.4 Krypton normalization 33
4.3 Feature Selection 34
4.3.1 Peaks Selection 34
4.3.2 PCA 35
4.3.3 Gradient Boosting 36
4.3.4 Variance Threshold 36
4.3.5 Convolutional Autoencoder 37
4.4 Data Augmentation 37
4.5 Classification Task 38
4.5.1 Metrics 38

5 Experimental Results 41
5.1 Creation of the dataset 41
5.2 Data Visualization 44
5.3 Outliers identification 46
5.4 Normalization 49
5.5 Feature Selection 53
5.5.1 PCA 53
5.5.2 XGBoost 54
5.5.3 Convolutional Autoencoder 55
5.6 Classification Results 60
5.6.1 Range 2 analysis with fractional AMUs 61
5.6.2 Range 2 Analysis: peaks as features 63
5.6.3 Full spectrum analysis 64
5.6.4 Classification with the CAE as feature extractor 65
5.7 Long COVID analysis 66

6 Conclusions 70

Bibliography 72

List of Tables

5.2 Average results with different mass ranges on the test set on 10 runs, no features selection or pre-processing. 61
5.3 Mean results of classification models on the test sets with mass range 2 considering any kind of normalization. 62
5.4 Mean results of classification models on the test sets with mass range 2 using the proposed normalization methods (4.2.3 and 4.2.4) and PCA trained on multiple acquisitions. 63
5.5 Mean results of classification models on the test sets with mass range 2 using the TIC normalization method and the peaks as features. . 64
5.6 Mean results (10 splits) for the whole mass range 10-351 (augmented dataset). 65
5.7 Mean results (10 iterations) for the whole mass range 10-351. 66
5.1 Architecture of the Convolutional Autoencoder - The CAE is made by the repetition of the basic block identified in the Section 5.5.3, the chosen latent dimension is 8 has we can see by the dimensionality of the last layer. The output sizes of each layer are coherent with the pooling layers formula specified in (5.1). 69

List of Figures

2.1 raw dataframe for range 1 - as we can see we have a column with the acquisition, a column for each AMU with its intensity and the index column that identifies the patient and the range. 9

2.2 overflow error - we can see that in correspondence of the peak 28 AMU we have negative values due to the floating point representation. 10

2.3 AMUs plot after overflow correction 11

2.4 Comparison between air sample and patient sample for an acquisition. 12

2.5 Box plots in log scale of Nitrogen, Oxygen, Carbon Dioxide, Krypton and Xenon. Each gas is divided by positive and negative samples in order to visually represent the quartiles of the 5 features by class label. 13

2.6 Scatter plots of some well known compounds. On the top of the figure are represented the AMUs of the x-axis, while on the left the AMUs of the y-axis. The 2 classes are again separated by color. . . 14

2.7 Comparison between air sample and patient sample for an acquisition. 15

2.8 class distribution inside our dataset 16

2.9 skew problem 17

2.10 patients' spectra colored by day - in this scatter plot, it is shown the distribution of the patients' spectra projected inside a 2-dimensional space using the T-SNE method, each of them coloured by the date of acquisition. 18

3.1 The basic structure of the encoder and decoder block. 26

3.2 The general structure of a standard AE, compressing input data and reconstructing it from the encoder output layer. 26

4.1 Application of the Savitzky-Golay filter on a mass spectrum. The blue signal is the raw data while the blue is the smoothed one. . . . 29

4.2 Baseline detection for a patient breath sample. 30

4.3 Effect of the peak alignment procedure on a patient breath spectrum. The dashed lines highlight the principal peaks of the signal before and after the alignment. As we can see each peak has been correctly aligned to the respective integer AMU. 31
4.4 Application of the peak detection algorithm on a slice of a second range mass spectrum normalized with the TIC normalization and smoothed with the Savitzky-Golay filter. 35

5.1 TIC plot for a breath sample. In this figure we can clearly see the 3 phases of the acquisition. In blue we can see the TIC plot and the red rectangle represents the plateau region identified by our algorithm. 42
5.2 The acquisition dataframe for the range 1. 43
5.3 The patient dataframe and its composition. 44
5.4 The comparison between the negative raw spectra and the positive one. 44
5.5 TSNE - distribution of positive and negative patients. 45
5.6 TSNE - distribution of positive and negative patients after the application of the data augmentation technique. 45
5.7 Krypton isotopic ratios from the NIST organization. The reference is peak the 84 m/z since it saturates to 100, the other peaks that we are taking in consideration are the Kr-82, Kr-83 and Kr-86. . . . 47
5.8 Krypton ratios inside air acquisitions. 48
5.9 Krypton ratios inside patients acquisitions. 48
5.10 Comparison between raw features and normalized and filtered spectra. 49
5.11 Distribution of the patients' spectra both colored by date and COVID-19 positivity. 50
5.12 Distribution of the TIC normalized patients' spectra both colored by date and COVID-19 positivity. 51
5.13 Distribution of the TIC and Krypton-84 normalized patients' spectra both colored by date and COVID-19 positivity. 52
5.14 The cumulative variance is represented in red while each bar represents the individual variance contribute to the whole variance. . . . 53
5.15 Feature selection with xgboost - only the 20 most important features are shown 54
5.16 The difference between a normal spectrum sample and the same signal with the addition of a normal Gaussian noise with $\sigma = 0.001$ and $\mu = 0$. 57
5.17 Training vs Validation loss of the CAE training. 57
5.18 Difference between a validation spectrum and the reconstructed version, we show only a slice of the data in order to appreciate the reconstruction error better. 58

5.19 Reconstruction metrics for 8 random validation patients' spectra and the means of each metric. The plot uses a double axes to represent the lower scale of the RMSE, while the plot bars refer to the right y-axis 59
5.20 Patient 20220721_5 spectrum considering only the peaks as features. 64
5.21 Comparison between a standard spectrum and the extracted features of the encoder part of the CAE. We plotted the decimals so to get the number of features we have to multiply the maximum value on the x-axis by 10. 65
5.22 Long Covid effect on test samples. On the x-axis are shown the days after which we still considered as positive the patients, while on the y-axis the metrics value. 67

Acronyms

AI

artificial intelligence

ML

Machine Learning

TIC

Total Ion Current

CAE

Convolutional AutoEncoder

MS

Mass Spectrometry

AMU

Atomic Mass Unit

Chapter 1

Introduction

1.1 Problem Description

The emergence of the novel coronavirus disease (COVID-19) caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) virus has led to an unprecedented global health crisis, challenging healthcare systems, economies, and societies worldwide. Rapid and accurate diagnosis of COVID-19 is crucial for effective disease management, infection control, and public health response. Traditional diagnostic methods such as reverse transcription-polymerase chain reaction (RT-PCR) have played a vital role in detecting the virus, but they often require specialized equipment, reagents, and can be time-consuming. As the pandemic continues to evolve, there is a pressing need for innovative diagnostic approaches that can provide quick and reliable results. These tests require medicals staff because are classified as invasive tests, an invasive test method is not applicable taking in consideration the high number of personnel required when a mass testing is performed, without taking in consideration that these methods can be stressful or in some cases even dangerous for fragile people.

Mass spectrometry, a versatile analytical technique widely used in proteomics and metabolomics research, has shown promise in disease detection and identification due to its ability to characterize molecular composition. Recent studies have highlighted the potential of mass spectrometry in detecting viral infections through the analysis of complex biological samples. The distinct molecular signatures associated with viral infections can be captured by mass spectrometry, offering a unique opportunity for rapid and sensitive diagnostics.

Machine learning and deep learning techniques have demonstrated remarkable success across various domains, including healthcare and medical diagnostics. Their ability to extract intricate patterns and information from large datasets makes them particularly well-suited for complex data analysis tasks. By harnessing the power of these techniques, it is possible to develop accurate and robust diagnostic models for detecting COVID-19 from mass spectra samples.

The primary objective of this study is to explore the feasibility of using machine learning and deep learning algorithms to detect COVID-19 from mass spectra samples obtained from patient specimens. By leveraging the inherent information contained within mass spectra, we aim to develop a predictive model capable of distinguishing COVID-19-positive samples from negative ones with high accuracy and efficiency. In particular, for the reasons specified above, the aim of this study is the creation of a non invasive test method that exploits both the advantages of the mass spectrometry data and the machine learning techniques.

NanoTech Analysis S.r.l. (NTA) was set up in Turin in 2013 as an Italian "Innovative European Startup" operating in the domain of MEMS and NEMS technologies. NTA owns patented technologies with the main purpose to develop and launch a technically innovative product family in the fluid and chemical measurements industry. The company owns the mass spectrometer to make the acquisitions of the patients breaths directly at molecular level. The instrument is able to identify the sample composition and with this not invasive technique we are able to distinguish positive or negative to COVID-19 samples.

1.2 Motivation and Objectives

The motivations of our study are mainly related to the realization of a rapid, non-invasive test protocol that is able to detect the presence or the absence of COVID-19 from a breath sample converted in a mass spectrum.

The first goal of this research is to build a relatable and ready to use dataset made by mass spectra of different patients starting from the acquisitions done by the NanoTech Analysis company from the 2021 to the 2022. A mass spectrum is a graphical representation or data output produced by a scientific instrument called a mass spectrometer. It provides information about the distribution of ions (charged atoms or molecules) based on their mass-to-charge ratio (m/z) measured in AMU (atomic mass unit). Since these are the raw results taken directly from the instrument, it is possible that they are affected to some noise, peak misalignment and background noise and so they will be subsequently processed to have cleaned data. Collecting these samples together allows us to build a primary version of our dataset where all the acquisitions are stacked together and so we will have as columns the AMUs (the mass to charge ratios), as values the intensity and as index the patient ID. Since this tabular version of our data contains valuable and exploitable knowledge we may proceed with some exploration and visualization of our data or proceed with the classification.

An other objective of our study is to actually extract some knowledge from the mass spectra, in particular we want to classify each sample as positive or negative to COVID-19. Our problem can be seen as a machine learning classification problem where the labels are provided and are discrete and categorical. To do this we will exploit on some classical machine learning techniques like SVM (Support Vector Machine) and Linear Regression but we will also try some deep learning approach. For this work we take inspirations from [1], [2] and [3].

1.3 Mass spectrometry

Mass spectrometry (MS) is an analytical technique that is used to measure the mass-to-charge ratio of ions, the results are presented as a mass spectrum, a plot of intensity as a function of the mass-to-charge ratio. In particular MS is a powerful scientific technique used to analyze the composition of molecules based on their mass.

- The process includes a first part called **ionization**, in this phase the compounds enter in a room in which the molecules are turned into ions (charged particles) so they can be manipulated by an electric and magnetic field. This is typically done by bombarding the sample with high-energy particles causing some of the molecules to gain or to lose one or more electrons. In particular, we can have two different situations:

  $$e^- + A \longrightarrow A^+ + 2e^-$$

  In this case the bombarded molecule lose an electron and it is positively charged, or

  $$e^- + A \longrightarrow A^{2+} + 3e^-$$

  In this case the compound is charged with one more electron, which will influence the result in the next step.

- Once the molecules are ionized, they are subject to an electric field that **accelerate** them, this step allows the particles to gain kinetic energy.

- The accelerated ions then pass through a magnetic field, since they are electrically charged, they experience a force due to the magnetic field, causing them to deflect from their straight path. The degree of deflection depends on their mass and charge: lighter ions deflect more than heavier ions. Exploiting this fact the particles can be separated basing on their mass to charge ratio (m/z): in particular ions with the same charge but heavier will deflect less respect to lighter ions.

- At this point the separated ions can be detected by a device that can measure their time of flight or their final position and determine the m/z ratios of the ions. Computing the abundance of each ions is possible to build the mass spectrum of the gas since each peak of the spectrum corresponds to a specific ion with a particular mass. A problem that we can already arise is that different compounds but with the same atomic mass are detected inside the same peak. For example a molecule of N_2 with AMU 28 has the same atomic mass of the carbon monoxide CO, this two compounds will be added together and will be in the same peak.

1.4 COVID-19

COVID-19 or 2019-nCoV is a pathogenic virus responsible of the outbreak of pneumonia that began at the beginning of December 2019 near in Wuhan City, Hubei Province, China. Coronaviruses mostly cause gastrointestinal and respiratory tract infection. In particular, SARS-CoV (Severe Acute Respiratory Syndrome coronavirus) are known to be extremely pathogenic [4]. Specifically, the principal mode by which people are infected with SARS-CoV-2 (the virus that causes COVID-19) is through exposure to respiratory fluids carrying infectious virus. Exposure occurs in three principal ways:

- Inhaling air that contains infectious virus within very small fine droplets and aerosol particles poses the highest risk of transmission. This risk is most pronounced when one is within a proximity of three to six feet from an infectious source, where the concentration of these fine droplets and particles is the highest.

- The deposition of virus, carried within exhaled droplets and particles, onto exposed mucous membranes (referred to as "splashes and sprays," such as those from coughing) presents a significant risk of transmission. This risk is especially elevated when one is in close proximity to an infectious source, where the concentration of these exhaled droplets and particles is most concentrated.

- Transmitting the virus can also occur by touching mucous membranes with hands contaminated by exhaled respiratory fluids containing the virus or by coming into contact with inanimate surfaces contaminated with the virus.[5]

1.4.1 Test Methods

During the pandemic different kinds of test methods have been developed in order to trace the virus proliferation. Currently, the following tests are available:

- **molecular tests**
- **rapid antigen tests**
- **serological tests**

Molecular tests on nasopharyngeal and oropharayngeal respiratory samples still remain the international gold standard for COVID-19 diagnosis in terms of sensitivity and specificity. The real-time RT-PCR (Reverse Transcription-Polymerase Chain Reaction) method allows for the detection of the viral genome's presence by amplifying the most expressed viral genes, not only in symptomatic individuals but also in the presence of low viral load, pre-symptomatic, or asymptomatic individuals.

Rapid antigen tests detect the presence of viral proteins (antigens). Different types of antigen tests are available, ranging from lateral flow immunochromatographic assays (first generation) to immunofluorescence reading tests (second generation), which exhibit better performance. The latest generation tests (immunofluorescence with microfluidic reading) seem to yield results comparable to RT-PCR assays. Antigen tests that can be performed in laboratories are now also available. The performance characteristics of these tests, based on chemiluminescence detection systems, are fundamentally similar to those of the so-called "third-generation" antigen tests (microfluidic tests with fluorescence reading). They appear to be particularly suitable, among other things, for managing screening within hospital facilities. If the patient's clinical conditions are inconsistent with the latest generation antigen test, RT-PCR remains the gold standard for confirming COVID-19.

Serological tests detect exposure to the virus by highlighting the presence of antibodies against the virus, but they cannot confirm an ongoing infection. For this reason, given the current state of technological evolution, serological tests cannot replace diagnostic tests (molecular or antigenic). Serological tests are useful for epidemiological assessment of viral circulation and estimating the spread of infection within a community.

1.4.2 Long COVID

Long COVID, also known as 'post-acute sequelae of COVID-19,' is a complex condition characterized by severe and diverse symptoms that develop following an infection with the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). It is estimated that at least 65 million individuals worldwide are affected by long COVID. This estimate is based on a conservative assumption that 10% of infected individuals experience these prolonged symptoms, given the documented 651 million COVID-19 cases globally. However, the actual number is likely much higher due to numerous unreported cases.

The incidence of long COVID varies, with estimates suggesting it affects 10–30% of individuals who did not require hospitalization, 50–70% of those who were hospitalized, and even 10–12% of those who have been vaccinated against COVID-19. Research has uncovered hundreds of biomedical findings, revealing that many patients endure a wide array of symptoms affecting multiple organ systems [6].

Individuals with long COVID can experience persistent symptoms that last for more than four weeks after the acute phase of their COVID-19 infection. Additionally, there is growing evidence suggesting a link between long COVID and metabolic dysfunction, particularly metabolic syndrome (MS). Laboratory analyses of long COVID patients have identified imbalances in various cardiometabolic parameters, including lipids, glycemic control, and markers related to obesity [7]. For these reasons we also tried to study the relationship between the mass spectra of healed from COVID-19 patients and the predictions of our classifiers. In particular patients that presented long COVID symptoms, since they may present some bio markers typical of positive patients should be detected as positive while the tampon results may have marked them as negative.

1.5 Organization

This work aims to organize a dataset for the collection of mass spectra data coming from the NanoTech S.r.l mass spectrometer instrument. The global goal of this study is to create a framework able to detect the COVID-19 presence or absence from the collected mass spectra with the auxiliary help of Machine Learning classification models and a deep convolutional autoencoder for the feature extraction.

- The Chapter 2 describes the organization and the content of the output file of the instrument, how the data have been collected and starts exploring the content of our dataset and the main problem that this kind of acquisitions can bring.
- In Chapter 3 a brief analysis of the used machine learning methods is done and also we start explaining how a convolutional autoencoder works.
- The Chapter 4 contains the main application of the signal pre-processing methods applied on the mass spectra to align the peaks, to correct the baseline and smooth the samples. Also, we start discussing some machine learning techniques to normalize the data and also some features selection methods are explained. Eventually, the metrics to quantify the performance of the machine learning models are cleared up.
- In Chapter 5 the experimental results are presented, we first give an explanation of how the outliers are detected and how each normalization method impact

on the distribution of our samples. Also in this chapter is presented the CAE architecture and the training phase of the net. During the last section the classification results are presented with a brief consideration for each of the outcome.

- The Chapter 6 contains the conclusion and an overall evaluation of the study has been done.

Chapter 2

Preliminary Dataset Studies

In this chapter we will analyze the main characteristics of our dataset and how it has been build, in particular we will focus on how mass spectrometer data are represented and how the measures are organized.

2.1 ASC files

All our data has been collected with the NanoTech Mass Spectrometer, this tool stored the acquisitions into an ASC file that follows this format: the first lines describe the measure environment and setting, while after the *IntervalOfScan* line we have the AMU value followed by the intensity value. As we can imagine from the previous statement our measure will be made by the couple AMU - intensity value. All these files are collected into a directory structure that follows these criteria:

1. the folder is named with the data of the acquisition with the format *yyyy-mm-dd, i.e all the acquisitions taken the 25th December, 2022 will be into the folder 2022-12-25.*

2. Inside each of these folders, will be held all the ASC files containing the acquisitions of each patient. In particular, the file name is made by the date of the acquisition followed by the information about the patient ID inside the day and the range of measure (for this field the zero value means that this is an air acquisition and not a breath acquisition). *i.e an ASC file named 'mar 10 2021 2_1.ASC' means that it is relative to the second patient measured in that day and contains the first mass range measures.*

3. inside this folder an additional file is stored: this is a CSV file that contains the information about each file like the name, the COVID status of each patient and the mass range of the measure.

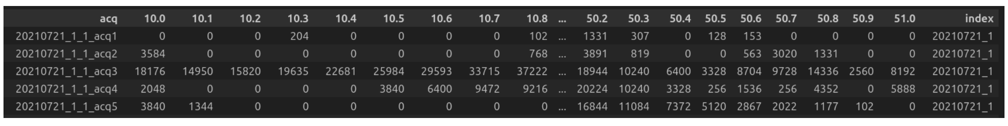
As we said before each file contains the information about the range of the measure, this field can contain values that goes from 0 to 4, in particular the 0 value means an air sample, while the others are specific for a range:

- the files with the *_1 notation contain the range that goes from 10 to 51 AMUs with an acquisition time of 10s and an EM voltage of 1000V.
- the files with the *_2 notation contain the range that goes from 49 to 151 AMUs with an acquisition time of 14s and an EM voltage of 1800V.
- the files with the *_3 notation contain the range that goes from 149 to 251 AMUs with an acquisition time of 14s and an EM voltage of 1800V.
- the files with the *_4 notation contain the range that goes from 249 to 351 AMUs with an acquisition time of 14s and an EM voltage of 1800V.

2.2 Data Collection

For a deeper understanding of how the dataset is built, we have to specify that the mass spectrometer takes its measures in different moments during the acquisition, so in each file we have the information about the Interval Of Scan that specifies when the various measures are taken and after that a set of couples of values that contain the information about the specific AMU and the relative intensity. So, inside a single file we will have different set of intensities for different interval of scan, in particular, when the machine is still pumping air in we will have growing TIC values, while when the machine is in full operation we will have a plateau and finally when the tool is pumping out the air we will see decreasing TIC values. When we talk about TIC (Total Ion Current), we mean the sum of all the intensity values that we have for a particular range, for each file we will build the TIC plot using the interval of acquisition as x values and the TIC (the sum of the intensities of all the AMUs) as y values.

From this plot we will select 4 TICs inside the plateau and the relative acquisition will be selected for the raw dataset. In this way will have a dataframe with primary key the date concatenate with the patient, the range and the acquisition selected from the plateau.



acq	10.0	10.1	10.2	10.3	10.4	10.5	10.6	10.7	10.8	...	50.2	50.3	50.4	50.5	50.6	50.7	50.8	50.9	51.0	index
20210721_1_1_acq1	0	0	0	204	0	0	0	0	102	...	1331	307	0	128	153	0	0	0	0	20210721_1
20210721_1_1_acq2	3584	0	0	0	0	0	0	0	768	...	3891	819	0	0	563	3020	1331	0	0	20210721_1
20210721_1_1_acq3	18176	14950	15820	19635	22681	25984	29593	33715	37222	...	18944	10240	6400	3328	8704	9728	14336	2560	8192	20210721_1
20210721_1_1_acq4	2048	0	0	0	0	3840	6400	9472	9216	...	20224	10240	3328	256	1536	256	4352	0	5888	20210721_1
20210721_1_1_acq5	3840	1344	0	0	0	0	0	0	0	...	16844	11084	7372	5120	2867	2022	1177	102	0	20210721_1

Figure 2.1: raw dataframe for range 1 - as we can see we have a column with the acquisition, a column for each AMU with its intensity and the index column that identifies the patient and the range.

The full dataset is built using the list of patients containing the date of the analysis, the relative file, the index and the COVID status and joining it with the one of the acquisitions that we talked before. From now we will consider our dataset as the set of columns made by the index that identifies the measure and the patient inside a date, the set of AMUs and their intensities and the COVID status.
A problem that we had to tackle during this preliminary phase was due the fact that in mass spectrometry when we talk about ion current values we deal with large numbers and in some cases they are too big to be represented with a finite decimal notation. We can graphically see such a fact plotting some spectra and we can notice that some current intensity values are negative. These numbers are physically impossible since that they can assume only positive values and so we know that there is a representation error.

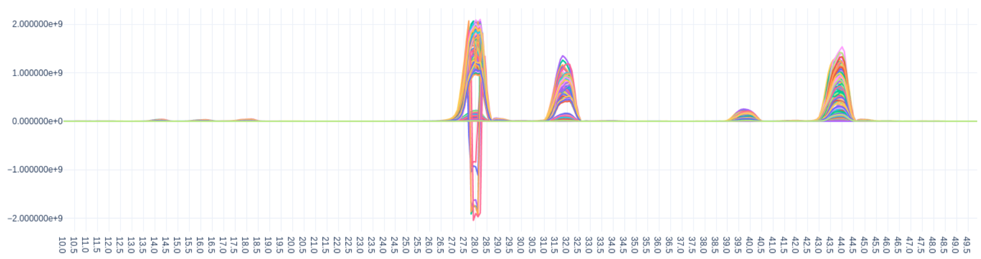


Figure 2.2: overflow error - we can see that in correspondence of the peak 28 AMU we have negative values due to the floating point representation.

The automatic `Python` conversion to `integer` is the cause of this error, in particular the programming language is able to represent values that are inside the range $[-2^{31}, 2^{31}-1]$ where the asymmetry is due the zero representation. In order to overcome this error, a manual conversation has to be done:

```
def UINT32_conversion(value):
    if value < 0:
        value = 2**32 - 1 + value
    return value
```

This solution uses the `unsigned integer` type provided by the programming language. In this way we can represent our values inside the range $[0, 2^{32}-1]$. After this transformation we can plot again our dataset to see the result. Proceeding with this method for each of the problematic value allows us to build a first version of our dataset with all the decimal AMUs as features.

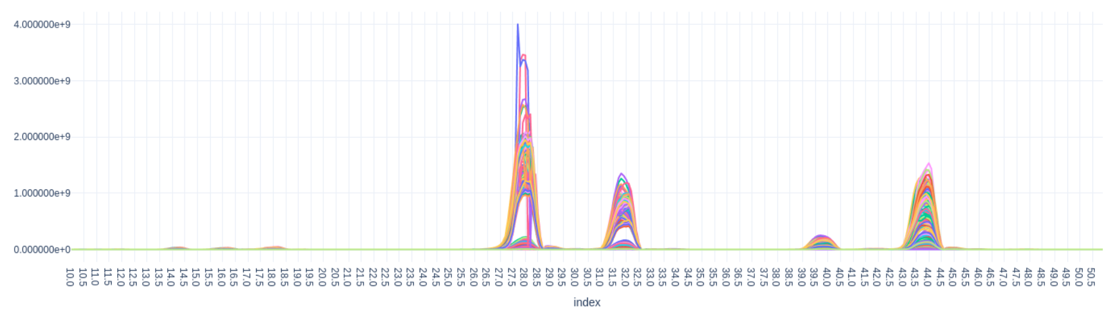


Figure 2.3: AMUs plot after overflow correction

2.3 Unsupervised Learning

2.3.1 T-SNE

In the following section and future chapters we make use of the t-SNE method. T-SNE stands for t-Distributed Stochastic Neighbor Embedding, is a dimensionality reduction technique used primarily for data visualization and exploration. It is particularly effective when dealing with high dimensional data by reducing the number of dominions while preserving the relationships between data points. In our study we mainly used it to plot the patients' spectra in a 2-dimensional space in order to be able to assess the distribution and the possible relationships between different samples.

T-SNE operates by first projecting data points into a lower-dimensional space. Then, it calculates pairwise similarities between these data points in both the original high-dimensional space and the lower-dimensional space. These similarities are quantified as conditional probabilities, reflecting how similar data points are in both spaces. The primary goal of t-SNE is to minimize the disparity between these two similarity metrics. In doing so, it determines optimal positions for data points in the lower-dimensional space, ensuring that similar points are positioned closely together, while dissimilar points are placed further apart. To do this, it exploits an iterative gradient descent technique [8].

2.4 Data Exploration

In this section we start exploring the mass spectra inside our dataset in order to identify some pattern, to assess the quality of our data, to make some studies on the distribution of the patients and how different compounds are related. A crucial point of this step is to identify outliers in our dataset, outliers are observations that are far from the samples distribution. We use the data coming from Section 5.1, but instead of using all the acquisitions we compute the mean for each patient, in this way we will have a single spectrum for each patient. The final dataset is stored in a `Pandas DataFrame` in this way the whole structure is more manageable and it's easier to build plots with the `plotly express` library. In this section we consider the whole spectrum as feature, the study of the set of features is important to try to understand the relationship between AMUs and extract valuable information from the dataset.

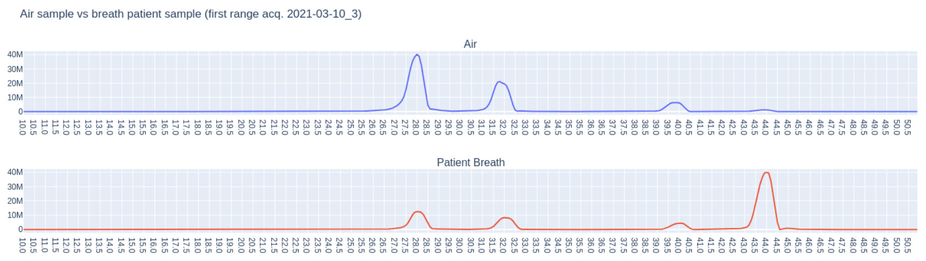


Figure 2.4: Comparison between air sample and patient sample for an acquisition.

To make this study coherent with the later classification phase we joined all the 4 ranges, in this way the patients that doesn't have the whole set of ranges acquisitions will be discarded. Since we are considering a very large set of features we focused on a reduced set of compounds to make the analysis more clear and significant. In particular the molecules that we investigated are:

- **AMU 28**, N_2 (Nitrogen). This compound is part of the first range of acquisitions, it's a colorless and odorless diatomic gas and it forms about 78% of Earth's atmosphere, making it the most abundant element in air.
- **AMU 32**, O_2 (Oxygen). Diatomic oxygen has constitute 20.95% of the Earth's atmosphere, for this reason it should represent the second most present compound in our air sample. Since this molecule it's transformed during the metabolism into carbon dioxide we should find it inside the breath sample but in minor quantity respect to the carbon dioxide.
- **AMU 44**, CO_2 (Carbon Dioxide). This molecule is present both in air samples and in breath samples, but in different quantities. Inside the air samples it should have a value about 0.04%, while inside the breath samples should be present in higher quantity due the metabolism transformation of the Oxygen.
- **AMU 84**, Kr (Krypton). It is a noble gas, since that it should not have any correlations with the other compounds. This gas belongs to the second range of acquisitions.

- **AMU 132**, Xe (Xenon). It is a noble gas, since that it should not have any correlations with the other compounds. This gas belongs to the third range of acquisitions.

As we can see from the Figure 2.4 in the air sample is easy to see the 4 peaks of first range relative to Nitrogen, the most abundant one, to Oxygen relative to the AMU 32 and the smaller one relative to the Carbon Dioxide. Instead, in the second subplot we can see the CO_2 peak more evident respect to the Nitrogen and Oxygen one, from the biological point of view this is a signal that our acquisitions are coherent with the literature.

A very common analysis to make in the data exploration phase is the construction of a box plot. A box plot is a graphical representation used to visually display key characteristics of numerical data, including its central tendency, variation, and distribution asymmetry. It achieves this by depicting data through quartiles. In addition to the central box, which covers the interquartile range, you'll often find lines (referred to as whiskers) extending from the box. These whiskers illustrate the range of variability beyond the upper and lower quartiles. Outliers, which are data points significantly different from the majority of the samples, are represented as individual points located beyond the whiskers on the box plot. As we can see from the Figure 2.5, the problem of the skewness is evident, in particular we can see it from the distance of the whiskers in each plot. For example, taking in consideration the Oxygen-32 we can see that for the negative class the min value is around 1.5M while the max value is over 1B, a difference of three order of magnitude. An assumption that we can make at this point is that the positive class suffers less this problem due the fact that the acquisitions are taken in a contracted lass of time and so they suffer less the data skewness. The choice of using the log scale for the y-axis is due the difference of intensity magnitude between a less present gas like the Xenon and the more abundant gas like CO_2 and O_2.

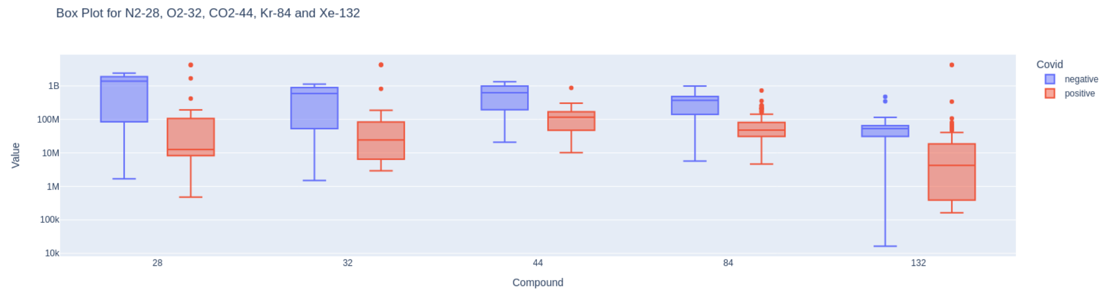


Figure 2.5: Box plots in log scale of Nitrogen, Oxygen, Carbon Dioxide, Krypton and Xenon. Each gas is divided by positive and negative samples in order to visually represent the quartiles of the 5 features by class label.

We can also notice a lower presence of outliers inside the first ranges, while it increases for higher ranges. This can be explained by the higher quality of the acquisitions inside the first 2 ranges due to the abundance of the compounds and the high quality of the TIC plot. On the other hand for higher ranges it is harder to detect compounds present in ppm quantities and also the plateau in those TIC plots are harder to detect or we had to increase the threshold parameter to have a sufficient number of samples scarifying a little the quality of the plateau.

In the next analysis will focus on the relation between the compounds, this is done with scatter plots. In particular this plot put in relation two compounds using on the x-axis and y-axis the intensities of these two compounds. The intersection of the points of the 2 axis identifies a patient, so compounds that are highly correlated should have an high number of points laid on the diagonal. This is the example of the CO_2 and the O_2 that seems to be highly correlated by their plot, this can be confirmed by the fact that they are also biologically related by the transformation inside the human respiration.

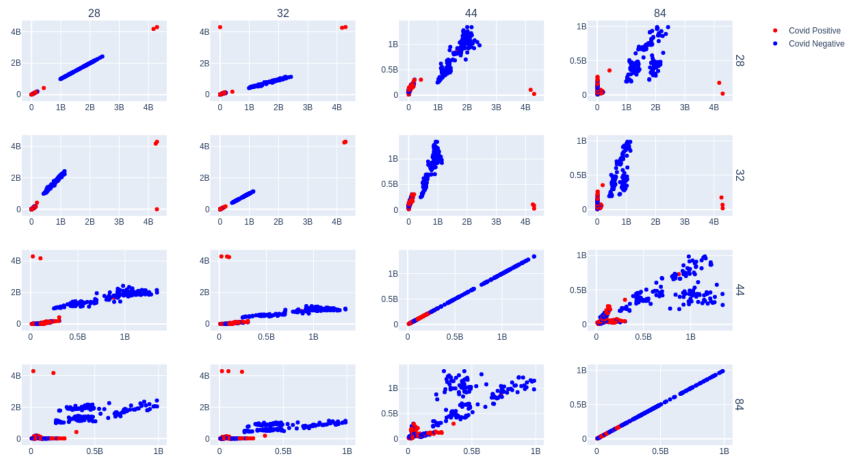


Figure 2.6: Scatter plots of some well known compounds. On the top of the figure are represented the AMUs of the x-axis, while on the left the AMUs of the y-axis. The 2 classes are again separated by color.

In this paragraph we will focus on the correlation between features, for this study we exploit the Pearson correlation coefficient and the relative heat map shown in Figure 2.7. Correlation is a statistical metric that quantifies the degree of association between two variables. It can take on either a positive or negative value. A positive correlation indicates that the variables tend to move in the same direction—when one increases, the other also tends to increase. Conversely, a negative correlation signifies that the variables move in opposite directions, meaning when one goes up, the other tends to go down.

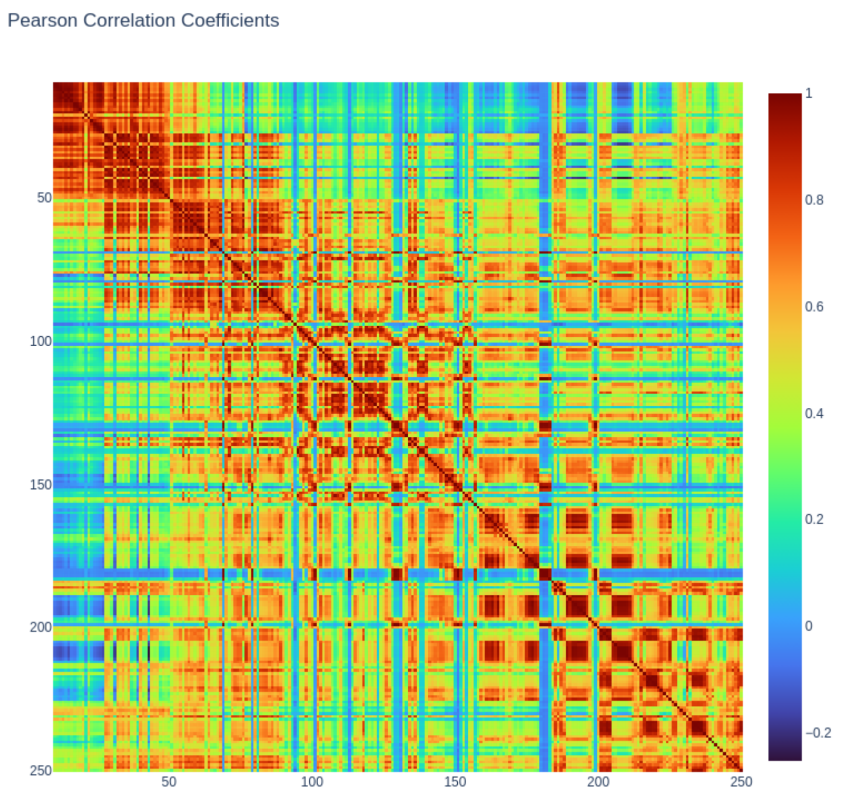


Figure 2.7: Comparison between air sample and patient sample for an acquisition.

In data analysis, the goal is to identify highly correlated features to select from a set of variables, as they exhibit similar behavior and often convey redundant information. To build this plot we use the Pearson Correlation Coefficient:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

where:

- $cov(X, Y)$ The covariance between two features (X and Y) represents a measure of how these two features vary together. In other words, it indicates whether, on average, they tend to increase or decrease together (positive covariance) or move in opposite directions (negative covariance).

- σ_X is the standard deviation of the X feature. The standard deviation for a single variable is a measure of how much individual data points in that variable deviate from the mean (average) of that variable. It quantifies the dispersion or spread of data points.
- σ_Y is the standard deviation of Y.

Since the high dimensionality of our dataset it's hard to build an understandable heat map, to make the task more easy we considered only the integer AMUs and their associated peak to reduce the number of variable to 250 for the three ranges. As we can see from Figure 2.7 the most correlated features are peaks near to each other and the correlation decrease while we are moving away from the considered peak.

2.4.1 Class labels distribution

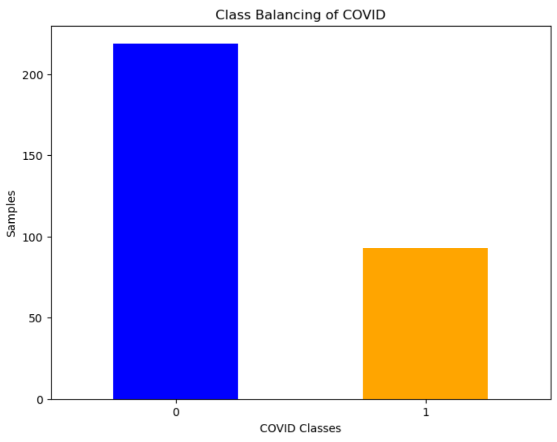


Figure 2.8: class distribution inside our dataset

As first analysis we will consider the class balancing: once we built the raw dataset with the class label we can count the number of positive and negative patients inside our dataset. Since the measures are taken from 2021 when the pandemic was at its peak, the positive samples are concentrate in that year, while in the 2022 there are only negative patients. Given that, our dataset results imbalanced, in particular the negative samples are more then twice the positive one.

This can lead to biased model performance. Many machine learning algorithms are designed to maximize overall accuracy. In the presence of class imbalance, the algorithm may become biased towards the majority class because it can achieve high accuracy by simply predicting the majority class most of the time. As a result, the minority class may be overlooked or misclassified at a higher rate. Since that, we will use some technique like under-sampling, oversampling or SMOTE (Synthetic Minority Over-sampling Technique).

2.4.2 Data skew

In time series analysis, the term "day bias" typically refers to the systematic patterns or variations that occur in data based on the day of the week. Day bias arises due to regular, recurring patterns associated with specific days of the week. This phenomenon can have a significant impact on the analysis and modeling of time series data, and it's important to understand and account for it appropriately. If we proceed our analysis on this first version of the dataset we can assess a skew problem inside our data. In particular, since the measures are taken during a large period of time (from March 2021 to July 2022), the machine under went to some maintenance and some deterioration, this caused a skew. More precisely, we can see this problem considering the spectra of the same person during different times of the study.

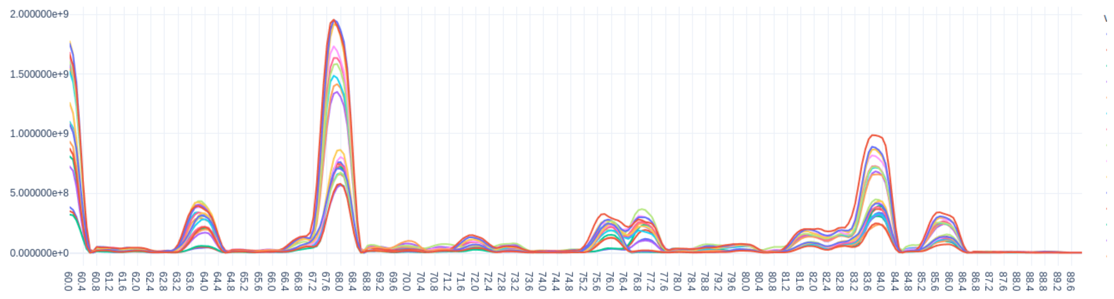


Figure 2.9: skew problem

As we can see from the plot above, the same person, whose wasn't positive to COVID and so we can assume that has the same breath composition except for some small changes, has a spectrum that is different from other spectra of some order of some order of magnitude. In the Figure 2.9 we can see a patient's breath and it is evident the difference between the peak on the AMU 68 of the acquisition of the 2022-04-12 (the highest one) and the lowest peak of the same AMU that is taken from the 2022-04-22.

This problem also led to a measurement bias, in particular if we run a dimensionality reduction technique like PCA or T-SNE, in order to be able to represent a single sample in a 2D space, we will see that acquisitions of the same day will be grouped together to form a cluster.

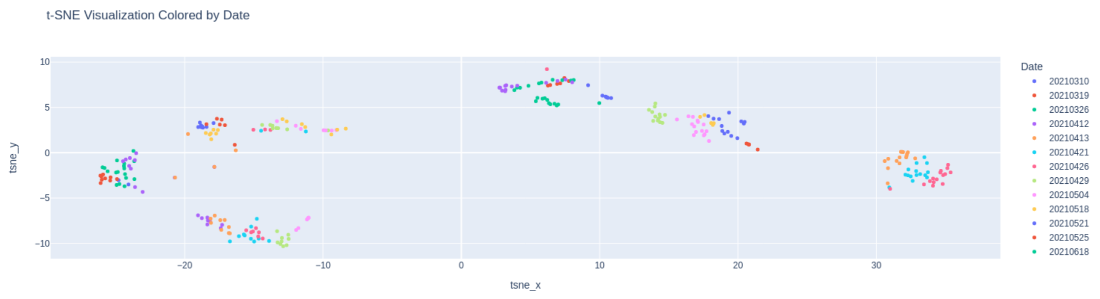


Figure 2.10: patients' spectra colored by day - in this scatter plot, it is shown the distribution of the patients' spectra projected inside a 2-dimensional space using the T-SNE method, each of them coloured by the date of acquisition.

A plot that makes more evident the measurement bias by day is shown in Figure 2.10, in this figure each point represents a patient spectrum coloured accordingly to its date: so acquisitions with the same colour belong to the same day. It's pretty straightforward to see that there is a pattern in the distribution of the dots: if we move from the left to the right we can see measures of the 2021, while moving from the right the date value increases to the 2022 acquisitions. A part from this, we can see that the measure of a day are concentrated in clusters, this is typical of day bias: points belonging to the same day are nearer respect to acquisitions taken in different dates.

This kind of problem can be mitigated using some normalization techniques or training our models using some stratified k-fold solution involving day separation.

2.5 Curse of Dimensionality

The curse of dimensionality arises when dealing with high-dimensional spaces, typically occurring when there is a large number of observed features. This phenomenon leads to an increase in data sparsity. To grasp this concept, let's consider a population of samples, denoted by n to represent the sample size, and p to indicate the number of observed variables (features). Some of these variables may even have missing values for certain samples.

As the dimensionality p increases, the "volume" of the space that the samples can occupy grows rapidly. You can visualize each of the p variables as an axis in a high-dimensional space. With a higher p, any specific neighborhood within this space is more likely to contain little or no data, making it sparse. In practical terms, this increase in sparsity poses significant challenges when collecting data that accurately represents the entire population.

Moreover, the curse of dimensionality can lead to computational challenges and hinder the effectiveness of various machine learning algorithms, as the amount of data required to adequately explore or model the high-dimensional space grows exponentially with the number of dimensions. Therefore, addressing the curse of dimensionality often involves techniques such as dimensionality reduction, feature selection, or feature engineering to reduce the adverse effects of high dimensionality on data analysis and modeling.[9]

Since we start from a first version of the dataset with 3021 features, it's highly probable that we occur in the curse of dimensionality and that our machine learning model may be penalized by that. For this reason we will recur to some dimensionality reduction techniques like PCA and also we will apply some of the most used methods in mass spectrometry like peak picking to try to reduce the number of features of our dataset.

Chapter 3

Background

3.1 Machine Learning and Statistical Learning

In today's rapidly evolving technological landscape, Machine Learning (ML) and Statistical Learning (SL) have emerged as pivotal disciplines that are reshaping a wide array of industries, including healthcare, finance, marketing, and autonomous transportation. These fields represent transformative approaches to harnessing the power of data, enabling more informed and efficient decision-making processes.

Machine Learning is both an art and a science, centered around the idea of training machines or computer systems to enhance their performance in specific tasks by leveraging knowledge acquired from data. It empowers computers to learn from experience, adapt to new information, and make predictions or decisions without explicit, task-specific programming. Machine Learning encompasses a broad spectrum of techniques, ranging from supervised learning (where models learn from labeled data) to unsupervised learning (which uncovers hidden patterns in unlabeled data) and reinforcement learning (where agents learn to interact with environments to maximize rewards). Machine Learning algorithms excel in tasks such as image and speech recognition, recommendation systems, and predictive analytics.

Conversely, Statistical Learning forms the bedrock upon which many Machine Learning methods are constructed. It is deeply rooted in statistical theory and focuses on comprehending the underlying relationships between variables within data. Statistical Learning is indispensable for tasks involving inference, prediction, and decision-making. It encompasses techniques like linear regression, classification, and clustering, providing the means to model and analyze data, extract valuable insights, and facilitate data-driven decision-making processes.

Together, Machine Learning and Statistical Learning offer a powerful toolbox for extracting knowledge and value from data, making them indispensable components of the ever-advancing field of data science [10].

In the next subsections we will explore some of the state-of-the-art models used in Machine Learning with a brief explanation of their behavior and functioning.

3.1.1 KNN Classifier

The K-nearest neighbors (KNN) algorithm stands as a non-parametric supervised learning classifier, leveraging the concept of proximity to make informed classifications or predictions regarding the categorization of individual data points. It frequently finds application in classification problems, where the assignment of a class label is determined through a majority vote mechanism. Nevertheless, before undertaking such classification tasks, a critical prerequisite is establishing a distance metric. In our work we chose the Minkowski distance with the parameter p equals to 2, that results as the classical Euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

KNN operates on the principle that data points with similar features tend to belong to the same class or category. To make a prediction for a new, unclassified data point, the algorithm calculates the distance between this point and its K-nearest neighbors within the dataset, where k represents a user-defined parameter indicating the number of neighbors to consider. The class label assigned to the new data point is then determined by the majority class among its K-nearest neighbors. By tuning the value of k, KNN allows for flexibility in the balance between model complexity and robustness. Smaller k values yield more flexible models that may be sensitive to noise in the data, whereas larger k values tend to provide smoother decision boundaries but might miss finer details in the data. The wrong choice of this value can lead to overfitting or underfitting.

In essence, the K-nearest neighbors algorithm is a versatile tool in the realm of machine learning, offering an intuitive approach to classification tasks by relying on the proximity of data points within a defined feature space.

3.1.2 Random Forest Classifier

The Random Forest algorithm is a versatile member of the ensemble learning family, serving dual roles in both classification and regression tasks. What sets it apart is its ability to harness the wisdom of multiple decision trees to improve predictive accuracy while mitigating the risk of overfitting. Random Forest operates by aggregating the predictions of individual decision trees. Instead of relying on a single tree's output, it takes a majority vote for classification tasks or averages predictions for regression tasks from a group of decision trees. This ensemble approach is what makes Random Forest robust and reliable.

To address overfitting, Random Forest introduces diversity into its decision trees. Each tree in the ensemble is trained on a distinct subset of the data and a random subset of features. This ensures that each tree focuses on capturing different aspects of the data, reducing the likelihood of overfitting to noise in the training data.

Furthermore, Random Forest often employs the bootstrap technique. This involves randomly sampling the dataset with replacement to create multiple training sets for each decision tree. By incorporating bootstrapping along with feature subsampling, Random Forest enhances the diversity of training data, further fortifying the ensemble's predictive power.

3.1.3 Logistic Regression

The logistic regression method is a machine learning algorithm employed firstly for binary classification tasks. The input of this algorithm is a set of features and a binary label. This classifier makes use of a sigmoid function (also called logistic function) to shape the probability that the input is a member of the positive class, in particular it transforms each input inside the range $[0, 1]$, making this algorithm appropriate to emulate a probability result. The sigmoid function is mathematically represented as:

$$\sigma(z) = \frac{1}{(1 + e^{(-z)})}$$

This model supposes that the log-odds of the probability of the class 1 is a linear combination of the output variables:

$$log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k$$

where p is the probability to belong to class 1 and each β is the coefficient associated to a feature. When training this model, the objective is to determine the optimal values for each β coefficient that effectively align with the training data. Achieving this entails utilizing optimization techniques such as gradient descent to minimize a specified loss function. Once these feature coefficients are established, they enable the transformation of log-odds into probabilities using the previously mentioned sigmoid function. This transformation facilitates the computation of the decision boundary, which acts as the threshold demarcating the two classes within the feature space.

3.1.4 Gradient Boosting Classifier

The gradient boosting classifier is a versatile machine learning algorithm categorized under ensemble learning. It serves both classification and regression purposes by harnessing the strengths of multiple weak learners, typically represented as decision trees, to construct a robust predictive model. This classifier initiates the process by creating one or more weak learners and utilizing them for preliminary classifications or predictions. At each stage of the process, a new learner is introduced, specifically optimized to rectify errors made by the preceding model, with a focus on addressing misclassified instances. The predictions from each decision tree are assigned varying weights based on their performance. These weighted predictions are then aggregated to form the final prediction for a given data point.

The algorithm further enhances its performance by adjusting the weights. It calculates the gradient of the loss function concerning predictions and updates the model's parameters in the opposite direction to minimize prediction errors.

This iterative process continues until a predefined number of new models have been generated or until the algorithm converges to an acceptable level of accuracy. Gradient boosting, with its iterative and ensemble-based approach, consistently delivers highly accurate and robust predictive models, making it a preferred choice in various machine learning applications.

3.1.5 SVM

Another widely employed state-of-the-art machine learning model is the Support Vector Machine (SVM). SVM is a powerful supervised learning model designed for both classification and regression tasks. Beyond its capability for linear classification, SVMs excel at efficiently conducting non-linear classification through a technique known as the kernel trick: instead of explicitly mapping data to a higher-dimensional space, which can be computationally expensive, the kernel computes the dot product in the higher-dimensional space directly in the original feature space. In our study we use the RBF kernel. The Radial Basis Function (RBF) kernel, also known as the Gaussian kernel, is a popular kernel function used in Support Vector Machines (SVMs) for classification and regression tasks. It is especially effective for non-linear classification problems, it allows the SVM to create non-linear decision boundaries that can flexibly adapt to complex patterns in the data.

The fundamental goal of SVM algorithms is to delineate a clear margin between samples belonging to different classes. These margins are drawn in such a way that the distance between the margin and the classes is maximized, thereby minimizing the classification error. This margin optimization process is a hallmark of SVM's discriminative power, enabling it to make robust and accurate predictions [11]

3.1.6 Voting Classifier

The concept behind a voting classifier is to leverage the collective wisdom of diverse machine learning classifiers and make predictions based on a majority vote among them. This approach proves valuable in harmonizing the strengths and compensating for the weaknesses of individual classifiers, enhancing the overall predictive performance. In our study, we employ a specific type of voting known as "hard majority class voting." With hard voting, the predicted class label for a particular instance in the dataset is determined by the majority vote among the class labels predicted by each individual classifier. In essence, the class that receives the most votes from the ensemble of classifiers is selected as the final prediction.

In contrast, there's another form of voting called "soft voting." Soft voting takes into account not only the majority class but also the associated class probabilities predicted by each classifier. The final prediction is made by aggregating these probabilities, often by calculating the weighted average or using other aggregation methods. Soft voting can provide more nuanced predictions, especially when the classifiers in the ensemble assign varying degrees of confidence to their predictions.

3.2 Deep Learning and Neural Network

Artificial intelligence (AI) has made remarkable strides in recent years, finding applications in various domains such as automation, speech and image recognition, medical diagnosis, and scientific research. However, one of the challenges in AI is extracting high-level, abstract features from raw data, especially when these features involve complex variations like accents in speech. This difficulty necessitates sophisticated understanding of the data, almost at a human-like level.

Deep learning, a powerful subset of AI, addresses this fundamental problem in representation learning by introducing hierarchical representations. These representations build complex concepts from simpler ones, allowing computers to understand data in a more human-like manner. At the core of deep learning is the feedforward deep network, often referred to as the multilayer perceptron (MLP). This mathematical function maps input data to output values through the composition of many simpler functions. Each mathematical function applied at different layers provides a new representation of the input, gradually forming a deep understanding.

One perspective on deep learning is that it empowers machines to learn the most suitable representation for the given data. Another perspective views deep learning as enabling computers to learn multi-step programs. In this view, each layer in the network can be seen as a state of the computer's memory after executing a set of instructions in parallel. Deeper networks can execute more instructions sequentially, leveraging the power of referring back to earlier results. However, it's important to note that not all information within a layer's activations directly encodes data-related variations. Some of it serves as state information, similar to counters or pointers in traditional computer programs, helping the model organize its processing. This approach helps to build deeper and deeper networks that are able to learn more complicated structure and improve task like image recognition and image segmentation. The improvements in this field also moved to create new layers or architectures with different goals. In particular in the next section we will focus on the Convolutional Autoencoders that take inspiration from the traditional autoencoder (Section 5.5.3) and the convolutional layers typically used in image-related projects.

3.2.1 Convolutional Autoencoder

A Convolutional Autoencoder (CAE) is an autoencoder that is commonly used for unsupervised learning tasks, such as image compression and denoising. It is an extension of the traditional autoencoder architecture that incorporates convolutional layers into both the encoder and the decoder part of the networks. Since our spectra are 1-dimensional inputs, we developed a 1D convolutional autoencoder that uses 1D convolutional layers, making it particularly suitable for sequential or time-series data. In CAEs, convolutional layers are used in the encoding part instead of the fully connected layers used in the classical autoencoder architectures and deconvolution layers are used in the decoding phase. A part from this layer, we also used standard layers of convolutional neural networks as well, as expanding layers such `MaxPooling`, `UpSampling` that performs inverse function in the decoding phase and `BatchNormalization`. The training process for a Convolutional Autoencoder is similar to that of a traditional autoencoder. The network is trained to minimize the difference between the input signal and the reconstructed spectrum using a loss function such as mean squared error (MSE). To build the architecture we took inspiration from [12] and [13]. The designed Convolutional Autoencoder (CAE) comprises 33 layers structured around a recurring block. This block consists of a 1D convolutional layer, followed by a batch normalization layer, and subsequently a max pooling layer in the encoder or an up-sampling layer in the decoder section, depending on its role (Figure 3.1).

In the decoder section, the input signal undergoes initial processing through a 1D convolution with a 128x3 kernel size. Subsequently, batch normalization is applied to the convolutional layer's output to maintain a mean close to 0 and introduce regularization during training. Following this, a max pooling operation reduces the sample size by half. These processes repeat in the encoder section with varying kernel sizes and pooling sizes.

The final layer yields a 63x8-dimensional feature space, representing the encoded version of the input spectrum. This encoded representation serves as input for the

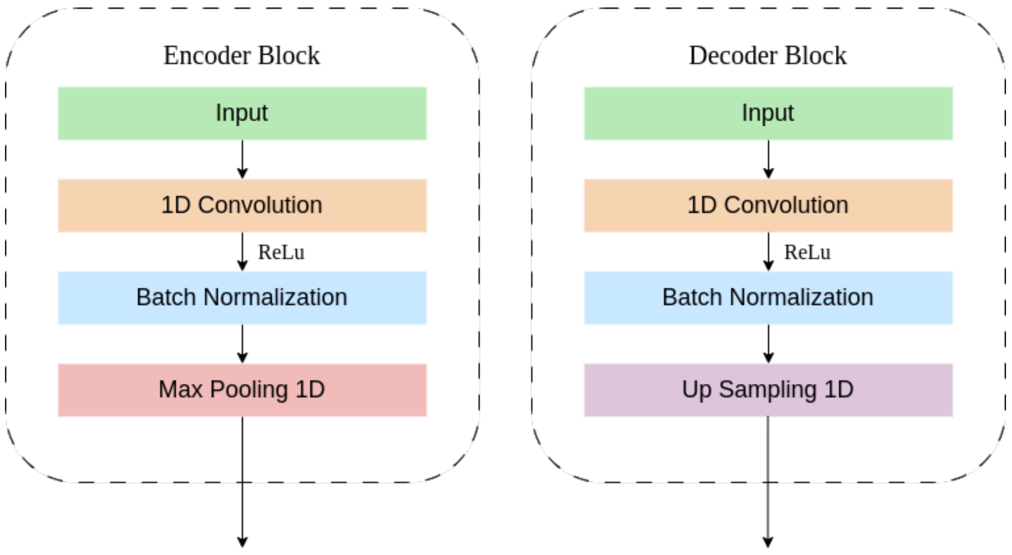


Figure 3.1: The basic structure of the encoder and decoder block.

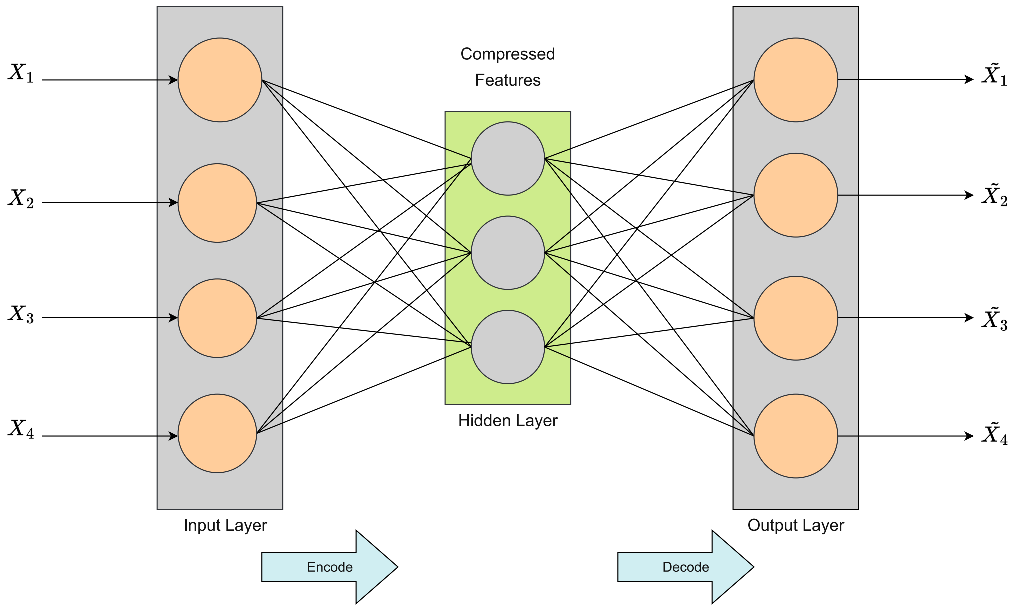


Figure 3.2: The general structure of a standard AE, compressing input data and reconstructing it from the encoder output layer.

decoder, where each layer executes operations that reverse those of the encoder. Specifically, up-sampling layers increase the input's dimensionality to reverse the max pooling operations. From this embedded space, the network aims to reconstruct the original signal with minimal Mean Squared Error (MSE).

To prevent data overfitting, L2 regularization is employed in each convolutional layer. This technique introduces a penalty term in the model's loss function, which is proportional to the sum of the squared weights. Consequently, it encourages smaller weights in the model, reducing complexity and mitigating the risk of overfitting. The terms follows the equation:

$$L2\_loss(w) = \lambda * \sum w^2$$

where

- w is the weight of a particular model parameter
- λ is the hyperparameter that controls the impact of the regularization

The architecture of the CAE, the actual code implementation, the training phase and the application of such convolutional autoencoder are explained in the Experimental Results chapter in Section 5.5.3.

Chapter 4

Method

4.1 Signal Processing

Signal processing in the context of mass spectrometry is the indispensable bridge that transforms raw, often noisy, and intricate data generated by mass spectrometers into meaningful insights. It involves a series of mathematical and computational techniques that extract valuable information from the signals produced during the ionization and detection processes. These signals provide a wealth of information about the mass-to-charge ratios of ions present in a sample, which in turn offers crucial insights into the molecular composition, isotopic distribution, and even three-dimensional structures of compounds.

The importance of signal processing in mass spectrometry cannot be overstated. It serves as the catalyst that enhances the accuracy, sensitivity, and reliability of mass spectrometric analyses. By efficiently filtering out noise, correcting for instrumental artifacts, and highlighting relevant features, signal processing techniques empower researchers to uncover hidden patterns, identify compounds with unparalleled precision, and unravel the complexities of biological, chemical, and physical systems.

In this introduction, we will explore the pivotal role of signal processing in the realm of mass spectrometry, delving into the methodologies that transform raw data into actionable knowledge.

4.1.1 Signal and Noise

Experimental measurements are never perfect, even with sophisticated modern instruments. Two main types of measurement errors are recognized: *systematic error*, in which every measurement is consistently less than or greater than the correct value by a certain percentage or amount and *random error*, in which there are unpredictable variations in the measured signal from moment to moment or from measurement to measurement. This latter type of error is often called *noise*. The main goal of denoising is to reduce the influence of random variations, several methods and transformations are available for this task, in our study we chose to adopt the Savitzky-Golay filter.

The Savitzky-Golay smooth is based on the least-squares fitting of polynomials to segments of the data. Compared to other technique like the sliding average smooths of the same width, this filter is less effective at reducing noise, but more effective at retaining the shape of the original signal.

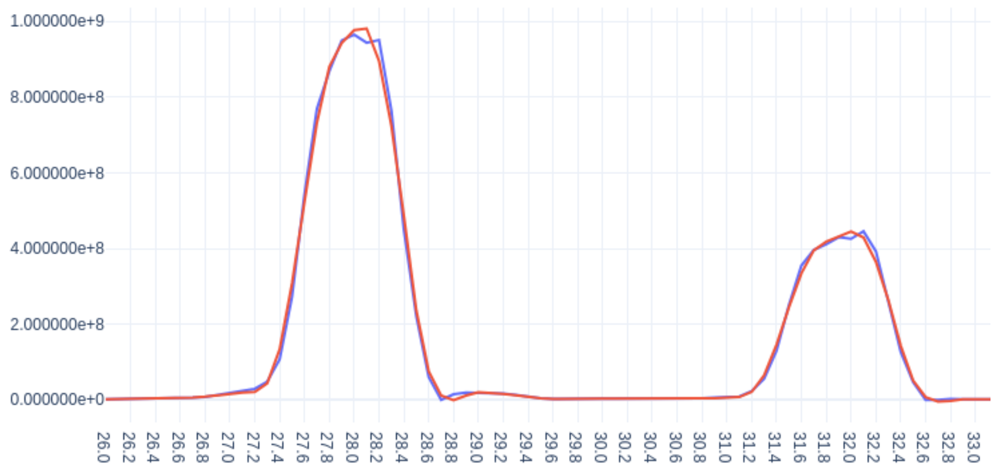


Figure 4.1: Application of the Savitzky-Golay filter on a mass spectrum. The blue signal is the raw data while the blue is the smoothed one.

4.1.2 Baseline correction

In mass spectrometry, the baseline problem refers to the challenge of separating the true signal representing the ions of interest from the background noise present in the acquired mass spectra. The baseline is essentially the flat or slightly undulating signal that runs along the zero intensity level on a mass spectrum plot. This baseline noise can arise from various sources, including electronic interference, fluctuations in the instrument, and other external factors, and it can obscure the accurate identification and quantification of analytes. The baseline problem is particularly significant in mass spectrometry because it can lead to inaccurate measurements, misidentification of compounds, and reduced sensitivity. If the baseline noise is not effectively removed or minimized, it can interfere with the detection and interpretation of the ions of interest, making it difficult to distinguish between true analyte signals and background noise. This is especially critical in applications where the target analytes are present in low concentrations, as the baseline noise can mask their presence or distort their peak shapes.

Addressing the baseline problem requires the application of sophisticated signal processing techniques. These techniques aim to distinguish between true signals and noise by employing mathematical algorithms to filter, smooth, and baseline-correct the mass spectra. Common approaches include baseline subtraction, where a fitting function is used to estimate and subtract the baseline noise, and various de-noising algorithms that aim to preserve the signal while reducing the impact of noise. The presence of a baseline or background signal, on which the peaks are superimposed, will greatly influence the measured peak if not corrected or compensated.

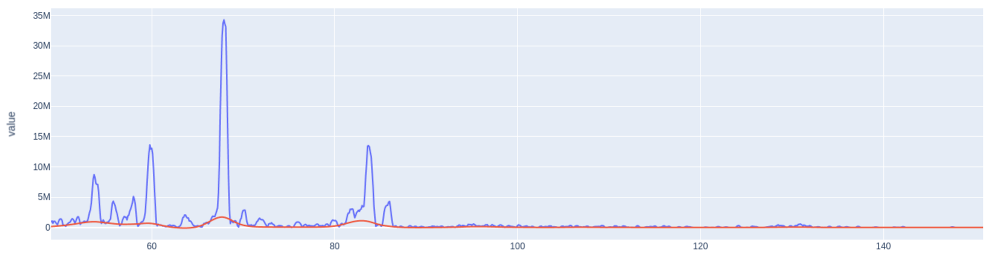


Figure 4.2: Baseline detection for a patient breath sample.

As we can see from the Figure 4.2, the mass spectrum contains a background signal that brings up the picks of the first part of the acquisition. This problem is due the fact that the machine cannot go immediately to zero after an higher pick is detected and so the next ion will start not from zero but from a higher value. This phenomenon can lead to overestimate the compounds near to a high peak and so it need to be tackle.
To solve this we leverage on the Asymmetric Least Squares (ALS) baseline correction method. ALS in an iterative mathematical algorithm used to estimate and correct the baseline of a signal, particularly when the baseline may exhibit both systematic variations and random noise. This is how it works:

- The ALS method assumes that the observed signal can be divided into two components: the baseline (slowly varying background) and the analyte peaks (rapidly varying signal components)
- The algorithm starts by making an initial estimation of the baseline, which can be a simple linear fit or a smooth curve that follows the general trend of the baseline.
- The ALS algorithm iteratively updates the baseline estimation and the analyte peaks' estimation. During each iteration, it applies a weighted least squares fit to the signal, where the weights are chosen asymmetrically to give more

importance to the data points that are likely to be part of the baseline rather than the analyte peaks. This asymmetric weighting helps to prevent the baseline from being pulled toward the analyte peaks.

It iterates the last step until the change in the estimated baseline between consecutive iterations becomes small.

4.1.3 Peak Alignment

Errors in calibration or limitations of the mass spectrometer can lead to variations between the observed m/z ratios values and the true ion intensity values [14]. Therefore, systematic shifts may appear in repeated acquisitions and the same compound that should belong to a particular AMU can have different m/z values in a different spectra. To correct this behavior different peak alignment algorithms have been proposed, in our case we adopted a simple approach moving the peak to the nearest integer m/z position, using them as anchors. The curves are adapted depending on the direction of the shift in order to avoid information loss. Stretching and compression are done with linear interpolation to fit the corresponding segments in the reference.

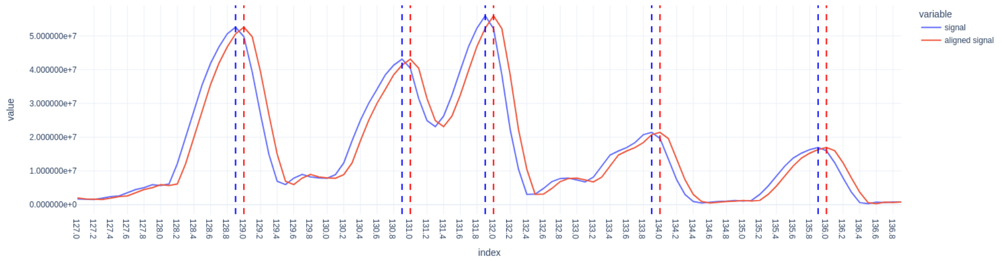


Figure 4.3: Effect of the peak alignment procedure on a patient breath spectrum. The dashed lines highlight the principal peaks of the signal before and after the alignment. As we can see each peak has been correctly aligned to the respective integer AMU.

4.2 Normalization

As we have described in the section 2.4 our dataset suffers of the problem of measurement bias in a way that acquisitions taken in the same day are nearer respect to others taken in different day. To overcome this problem we have to apply some normalization technique in order to scale our samples in a way that are comparable between them and not only with measures inside the same day. To overcome the problem the feature scaling is applied, taking in consideration both normalization and standardization solutions. In addition to classical approach we will use we try to build a Krypton normalization method to try to use an intrinsic value of the sample as reference.

4.2.1 Standard Scaler

The `StandardScaler` class from the `sklearn` library is used to standardize features by subtracting the mean and re-scaling to unit variance. In particular, the output will assume the value:

$$z = \frac{(x - u)}{s}$$

where u is the mean of the training samples and s is the standard deviation of the training samples. Standardization is a common prerequisite in many machine learning methods. This requirement arises from the observation that features with unscaled, high intrinsic values can dominate the importance of other features that typically possess lower values.

4.2.2 Robust Scaler

As we said before, normalization and standardization are useful to avoid that features wit a larger scaler dominates other features, similarly outliers can overshadow other data point for a given feature. For this reason in this section we will introduce the Robust Scaler, a scaling method resistant to outliers. The goal of this method is the same of the Standard Scaler: we want to re-scale our features in a way that are comparable with each other, the difference lies in how they scale the input values. Despite Standard Scaler, Robust Scaler uses median and interquantile range to scale the input features, in a more precise way it measures the distance in terms of IQR between the considered point and the input's median.

$$z = \frac{x - median_{inputs}}{IQR_{inputs}}$$

The IRQ is the range between the first quantile and the third quantile, the fact that this method uses the IRQ and median makes it resistant to outliers. That's because outliers, unlike the mean, doesn't affect the median.

4.2.3 TIC normalization

TIC normalization, which stands for Total Ion Current normalization, is a common data pre-processing technique used in mass spectrometry to correct for variations inside the spectrum intensities across different mass spectra. In mass spectrometry experiments, it's common to observe fluctuations in ion intensity due to various factors such as instrument settings and experimental conditions. TIC normalization aims to mitigate these variations and enhance the comparability of mass spectra acquired in different days or conditions. For TIC (Total Ion Current) we mean the sum of all the intensities inside a mass spectrum. So, the normalized sample z will assume the value:

$$z = \frac{x}{TIC}$$

where

$$TIC = \sum_{amu}^{N} intensity_{amu}$$

In practice we sum up all the m/z values inside the spectrum and then we will divide the sample by this sum. The result of such a method is a spectrum made by the relative compounds abundance respect to the TIC.

4.2.4 Krypton normalization

Normalizing mass spectra using a specific compound's intensity is an alternative approach to use the Total Ion Current (TIC) for normalization, this method employs the intensity of a chosen reference compound's ion peak. This technique is often referred to as **internal standard normalization** or **compound-specific normalization**. In our study we chose the Krypton-84 as reference compound, in this way the final result of this method is an histogram of values of relative intensities respect to the Kr-84. The reason for which our choice has fallen on this particular compound is that it is a noble gas and consequently it should not interfere with the metabolism and so it should remain constant between all the patients. In particular, the choice of the Kr-84 should reflect the property of constant concentration across all samples in our experiment, so it's important to pick a reference that is not affected by experimental condition and represents a stable concentration. This method involves the selection of the peak of the Kr-84 and the computation of the ratio between all the intensities inside the sample and the selected value. Mathematically:

$$z = \frac{x}{x_{Kr_{84}}}$$

This technique has the advantage of being compound specific, this method relies on the small variations of the selected compound between different samples rather then the whole TIC.

4.3 Feature Selection

In section 2.5 we talk about the curse of dimensionality and the relative problems of having an high number of features as input of a machine learning model. To tackle this problem we propose different features selection methods that leverage on classical machine learning approach like PCA and Gradient Boosting, on statistical methods like the variance threshold and on domain knowledge like using the peaks as feature. The feature selection phase should help us to have a more compact and lower dimensional dataset without the loss of information on the other side some of these methods will be later used to represent our samples in a 2-dimensional space in order to visualize the representation of our dataset.

4.3.1 Peaks Selection

Since now we have considered all the AMUs inside the spectrum as features, each acquisition has 3020 intensity values, but in this measures we are considering also the fractional AMUs. Physically talking, AMUs can assume only integer values, this value should be approximated by the peak intensity corresponding to that AMU. So, the idea behind this feature selection method is to use only the intensity associated with the peaks of the spectrum and the integer AMU as features. Doing this, since our acquisitions are made by mass ratios distanced by 0.1 of an AMU, we can reduce the dimensionality of our dataset by a factor of 10, bringing the features dimension to 302.

An important step in this method is the peak detection phase, in which we try to identify the peaks inside our spectrum and also we have to associate them with the integer AMU. A very common technique for this task is to use the differentiation, in particular the first derivative of a peak has a downward-going zero-crossing at the peak maximum, which can be used to locate the x-value of a peak [14]. This method assumes that there is no noise in the signal, because some noise in real experimental signals can cause many false zero-crossing. To avoid this problem we chose to apply a smooth algorithm before running this procedure, specifically we use the Savitzky-Golay filter as we have discussed in Section 4.1.1. For this step we used the `find_peaks` function of the `scipy.signal` module, this procedure also takes as input a `distance` parameter that specifies the minimum distance after which it can find another peak. Since in our case the peaks are often in the middle of the range $[AMU - 0.5, AMU + 0.5]$ where AMU is the integer m/z ratio, we set a `distance` of 6, in order to start the next identification after the 5-th decimal AMU. This function returns the position of the peaks inside our spectrum, so it's easy associate each maximum to its position. In some cases, especially when the spectrum is nearly flat or there is a particular noisy area of the spectrum the procedure is not able to find peaks. In this cases we associate the intensity value

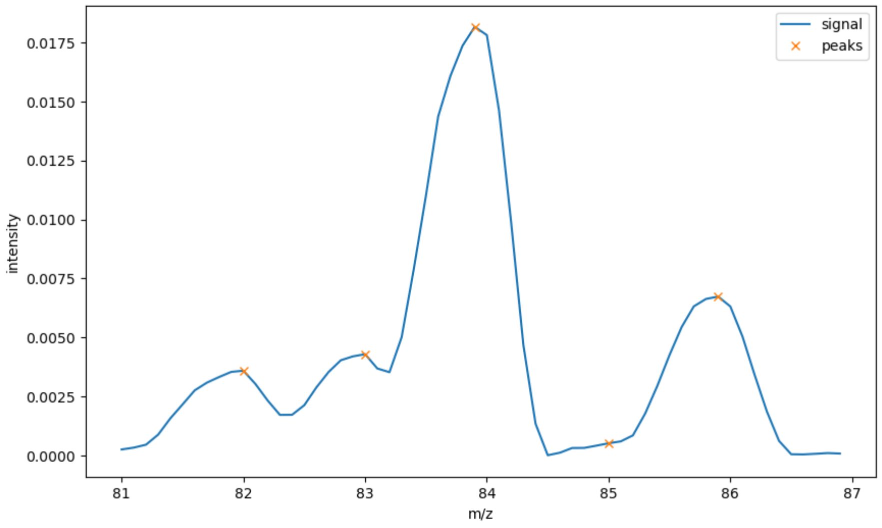


Figure 4.4: Application of the peak detection algorithm on a slice of a second range mass spectrum normalized with the TIC normalization and smoothed with the Savitzky-Golay filter.

to the integer AMU, this simple approximation works because it's highly probable that a high number of spectra has these flat zones and so these peaks will be filtered during the variance threshold phase (Section 4.3.4).

4.3.2 PCA

PCA (Principal Component Analysis) serves as a dimensionality reduction technique, employed to condense the number of features within input data instances. Typically, it transforms a large feature set into a smaller one while preserving the majority of the essential information from the original set. The objective of PCA is to discover a more meaningful basis for representing a given dataset. This newly derived basis is anticipated to unveil concealed patterns within the dataset while effectively reducing noise. PCA finds applications in various domains, including dimensionality reduction, data compression, feature extraction, and data visualization. [15]

The core objective of PCA is to diminish the feature count by constructing a covariance matrix, identifying highly correlated variables that can be safely removed due to their redundancy. In the quest to reduce features, a select few linearly uncorrelated principal components are chosen to account for the majority of data variation. The initial principal component explains the greatest variation within the data, followed by each subsequent component, which is selected to be the most orthogonal and varying. Principal components are essentially novel variables created as linear combinations of the original variables. These combinations are designed in such a way that the new components remain uncorrelated, and most of the information is condensed into the initial components. This makes PCA an invaluable tool for exploring and simplifying multidimensional data.

However, it's worth noting that PCA's effectiveness can be influenced by the scaling method applied to the data and the potential presence of outliers in the dataset.

4.3.3 Gradient Boosting

The concept behind feature selection using gradient boosting (detailed in Section 3.1.4) stems from the model's intrinsic use of decision trees. This characteristic makes it relatively straightforward to calculate importance scores for each attribute.

In the context of gradient boosting, *importance* refers to a score indicating how crucial a feature was in constructing the boosted decision trees within the model. Essentially, the more a feature is employed in the decision-making process within the trees, the higher its importance score becomes. This score is computed for every attribute in the dataset, enabling a comparison of feature importance rankings.

The determination of importance takes into consideration the impact of each attribute's split point on the model's performance, factoring in the number of observations handled by each node. Performance metrics, such as purity measured by the Gini index or other customized error functions, guide the selection of split points. Subsequently, these individual feature importances from each tree are aggregated by averaging them across all the decision trees in the model.

These importance scores serve as a basis for feature selection. By establishing a reasonable threshold, we can select only those features with relative importance scores surpassing that threshold value. This process enables us to isolate the most significant features, effectively reducing the number of attributes within our dataset.

4.3.4 Variance Threshold

Variance threshold feature selection is a technique used in machine learning to select or remove features from a dataset based on their variance, which quantifies the amount of variation or spread in the data for a particular feature. It's a simple method but effective for feature selection especially when dealing with high-dimensional datasets.

The basic idea behind this method is to identify and remove features with low variance because they are less informative and may not contribute much to the predictive power of a machine learning model. In particular, features with low variance have values that don't vary significantly or are nearly constant inside the dataset and may not carry useful information. Choosing a correct threshold is crucial, an higher threshold brings to a lower selection of features, while a lower threshold to an higher selection. In our study we run our machine learning models with different value for the threshold to compare their performance and chose the best.

4.3.5 Convolutional Autoencoder

An autoencoder represents a distinct type of neural network with the primary objective of replicating its input at its output. Its operation involves an initial encoding step, wherein it transforms input data into a lower-dimensional latent representation, followed by decoding, which reconstructs the original input using this latent representation. Through this process, an autoencoder learns to compress data while minimizing reconstruction errors. Typically, an autoencoder comprises two essential components: the encoder and the decoder. The encoder usually consists of multiple layers designed to extract feature representations from the input. It then forwards this representation to the decoder, which endeavors to reconstruct the initial input.

In our specific approach, we initially trained our model to replicate input spectra. Subsequently, we omitted the decoder component and retained the encoder part, which possesses the ability to compress information into a lower-dimensional space. In essence, this encoder segment functions as a feature extractor, enabling the projection of high-dimensional data into a lower-dimensional space[16].

4.4 Data Augmentation

Data augmentation is a technique commonly used in machine learning to artificially increase the size of the dataset by applying various transformations to the existing data. The primary goal of data augmentation is to improve the model's performance and generalization by exposing it to a wider range of variations and reducing the risk of overfitting. We use data augmentation to increase the number of samples at our disposal, in particular we start from the spectra of 312 patients and with this technique we are able to reach over 40,000 samples. The idea behind this method is to use all the acquisitions extracted from a single measures inside the plateau and use them in combination with each acquisition of the other ranges for that patient. In this way instead of extracting a single spectrum from each acquisition we can extract multiple spectra made by the combination of each measure of different ranges.

4.5 Classification Task

For the classification task we use some of the state of the art machine learning algorithms and some famous metrics. The objective of the classification is to discern COVID-19 positive instances from the negative one. This task is done on a dataset build from the `.ASC` files collected by NTA and filled with the mass spectrometer acquisitions, how the dataset has been built is explained in the Section 5.1.

For the sake of classification we follow the standard machine learning approach: we split our dataset into a training set and a test set and use we use part of the training split as a validation set to perform the hyperparameter tuning. Specifically, during the training phase we will use the training set to train our models, while during the testing part we will focus on the testing set to compute some metrics and evaluate each models in order to be able to compare each of them and select the best one. Since we use data augmentation as specified in the Section 4.4, the training set will be made by the combination of the acquisition of the same patient during that day, while the test set is composed by the mean of all the acquisitions taken from the plateau of the TIC plot for that patient.

In our research, we will focus on a specific validation approach known as stratified K-fold. This method combines the conventional k-fold cross-validation technique with the principles of stratified sampling. Stratified sampling is a sampling technique that maintains the same proportional representation of samples as seen in the original dataset. To achieve this, the population is divided into distinct groups, often referred to as *strata*, based on a specific feature. When applied within cross-validation, this technique ensures that both the training and test sets possess the same proportion of the feature of interest. In our case, this feature of interest pertains to the day of acquisition.

The stratified K-fold approach proves invaluable in cross-validation as it guarantees that the cross-validation results closely approximate the errors computed over the entire population. This method becomes particularly pertinent when dealing with time-dependent data, where variations in data characteristics may occur across different days. Since this is exactly our case, the application of this method allows us to capture day-specific characteristics and enhanced generalization.

4.5.1 Metrics

In our study the results are expressed in terms of F1-score (F1), precision (P), balanced-accuracy (A) and recall (R) to evaluate the performance of our classifiers and to be able to compare them. These metrics are computed using the concept of *True Positive (TP)*, *True Negative (TN)*, *False Positive (FP)* and *False Negative (FN)*. Specifically, they represents:

- **True Positive (TP)** represents the number of instances correctly predicted as positive by the model.

- **False Positive (FP)** represents the number of samples predicted as positive when they are actually negative.
- **True Negative (TN)** represents the number of instances correctly predicted as negative by the model.
- **False Negative (FN)** represents the number of samples predicted as negative when they are actually positive.

All of these four metrics are used as parameters for the computation of the evaluation metrics.

- **Precision** This metric measures how many of the positively predicted instances are actually positive, mathematically

  $$P = \frac{TP}{TP + FP}$$

  An high precision means that the model makes fewer false positive errors, meaning when it predicts a positive class, it's likely to be correct.

- **Recall**. Recall also known as sensitivity is a metric that measures how many of the actual positive samples were correctly predicted. It's also the ratio of true positives (TP) to the sum of true positives and false negatives (FN):

  $$R = \frac{TP}{TP + FN}$$

  An high value of recall indicates that the model captures a large proportion of actual positive instances, which is particularly important when the cost of missing a positive instance is high.

- **Balanced accuracy**. This metric takes in consideration both true positive and true negative rate, making it a good metric for imbalanced datasets. It's the average of recall and specificity (true negative rate).

  $$A = \frac{R + S}{2}$$

  $$S = \frac{TN}{TN + FP}$$

  and `R` is the recall as specified above. This metric can deal with imbalanced datasets.

- **F1-score**. The F1-score is a metric used to measure the balance between the precision and the recall. It takes into account both False Positives and False Negatives. The formula to compute the F1-score is:

  $$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

  It's particularly useful when you want to find a balance between precision and recall, especially, as in our case when the class distribution is imbalanced.

We have to specify that in case of unbalanced dataset, if the model tends to predict the majority class more frequently, it might achieve high accuracy due to the abundance of true negative despite the performance on the minority class might be poor. In this cases we may have high accuracy but recall could be very low, meaning that the model misses a large number of positive instances making the FN very big lowering the ratio.

Chapter 5

Experimental Results

In this section experiments are presented and some practical considerations are made, in particular we will concentrate on the implementation of each method presented in the previous chapter. The technologies used are mainly related to `Python` packages, in particular we used the `Scipy` library, `Pandas` for the presentation of the `DataFrame` and `NumPy` for the mathematical manipulation. All the plots and results representations are made with the `MatPlotLib` and the `Plotly Express` module. The convolutional autoencoder used as feature extractor has been developed with the `TensorFlow` library. All the classification models are imported from the `scikit learn` framework except for the Gradient Boosting Classifier that is taken from the `xgboost` Python module. Our study used the `Jupyter Notebook` technology, an open source web application to create and share documents containing texts and live code.

5.1 Creation of the dataset

In this section we discuss about the creation of the very first version of the database, the one that contains the raw acquisitions of each patient taken from its spectrum. We already talk in section 2 that each patient may have been tested multiple time (always in different days) and the organization of the ASC files with the acquisitions. The starting point is to specify that each patient acquisition is identified by the date, an incremental identifier relative to the day and the range in which the measures are taken. The goal of this analysis is to find the best way to represent the data collected for each patients in an accessible and easy to explore way.

Our mass spectrometer collects the acquisitions inside an ASC file, the machine collects the data at each interval of scan, so inside each of these files we can found multiple acquisitions for a single patient measure. We have to specify that the instrument goes through different phases during the measure: when we start the acquisition the machine room is presumably empty, when the breath or the air comes in, fills the room and the acquisition can start after the measurement has taken place the air chamber is emptied and the process ended. In each of these phases several acquisitions took place, but not each of them is a good measure. In particular, we are interested only in the acquisitions taken when the machine is operating at full capacity. To do that, we exploit the fact that when the instrument room is filling with air or with the patient breath, the quantity (intensities) of the compounds detected by the machine should grow since new substances are introduced into the instrument. Since the global quantity of compounds is growing, also the TIC should grow because more ions will be detected. Some seconds after the mass spectrometer should reach its regime and we should see a plateau, that region is where the flux reaches its stability. Eventually in the final phase the mass spectrometer valve is open and the flux will leave the instrument decreasing also the value of the TIC.

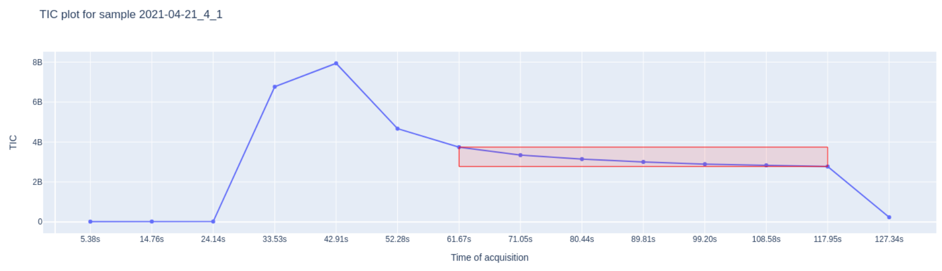


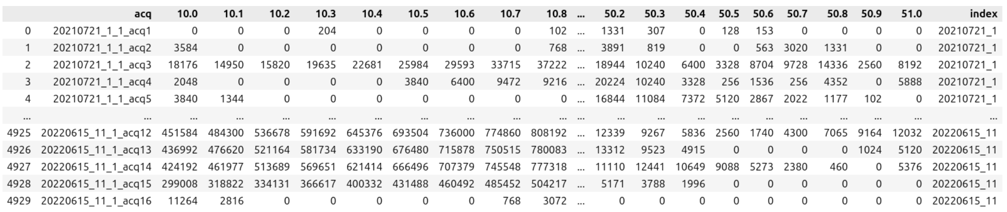
Figure 5.1: TIC plot for a breath sample. In this figure we can clearly see the 3 phases of the acquisition. In blue we can see the TIC plot and the red rectangle represents the plateau region identified by our algorithm.

The algorithm to identify the plateau works in this way, it takes 5 parameters:

- the signal
- the minimum allowed length of the plateau
- the maximum length of the plateau
- the tolerance that is a number between 0 and 1 indicating how strict are the requirements for the slope of the plateau to be considered constant.
- the size of the uniform 1D filter applied to the signal and its derivative

As first step the procedure applies a uniform 1D filter to the signal with the specified size taken as parameter. After that, the first derivative is computed on the smoothed version of the signal. The idea is to consider a plateau the region in which the first derivative of the function is almost flat. Then we apply the $q - th$-quantile function to the absolute value of the first derivative of the signal with q value equals to the specified tolerance, the returned value will be used as threshold under which other values will be considered as zero, this will constitute our tolerance guard-band ϵ (this can be seen from the Figure 5.1). Proceeding in this way allows us to specify different threshold depending on if we want to be more strict on the definition of the plateau or we can consider flat also the zone with bigger slope variations. After this step, all the zero values inside our array are considered as flat zone, so we have to take the largest one to find maximum plateau. As final step we computed the standard deviation of acquisition inside the detected plateau using a rolling window of size 4. We pick the acquisition for which this metric is minimum and in some cases we will return directly all the measures relative to each point of the TIC plot or one single spectrum made by the acquisitions inside the plateau.

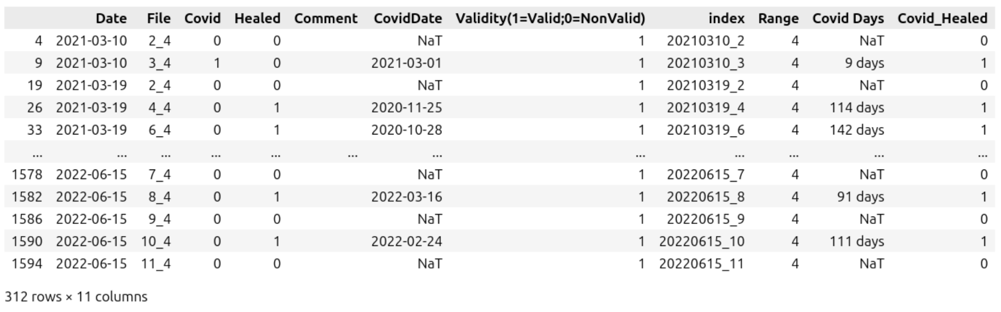
Specifically, all the acquisitions inside the plateau are considered during the training phase of our models, while the mean of the patient acquisitions is considered in the testing phase. Following this approach allows us to have a bigger dataset during the training phase and only a sample for each patient, or better acquisition, during the test and validation phase. Since sometimes was not possible to detect a plateau given the highly prototypical nature of the mass spectrometer or the non-optimal environment during the acquisition, some measures have been discarded. After this step we built 4 different datasets, one for each range, in particular the first goes from 10 to 51 AMUs, the second from 49 to 151 AMUs, the third from 149 to 251 AMUs and the latter from 249 to 351 AMUs. It's mandatory to specify that for the higher ranges less acquisitions are available and so it wasn't possible to build a full spectrum for each patient. The dataframes built at these step are made by the *acq* column that contains for each entry the information about the day, the patient and the id of the acquisition inside the plateau, by a column for each AMU with a single decimal point and the *index* that identifies the patient.



	acq	10.0	10.1	10.2	10.3	10.4	10.5	10.6	10.7	10.8	...	50.2	50.3	50.4	50.5	50.6	50.7	50.8	50.9	51.0	index
0	20210721_1_1_acq1	0	0	0	204	0	0	0	0	102	...	1331	307	0	128	153	0	0	0	0	20210721_1
1	20210721_1_1_acq2	3584	0	0	0	0	0	0	0	768	...	3891	819	0	0	563	3020	1331	0	0	20210721_1
2	20210721_1_1_acq3	18176	14950	15820	19635	22681	25984	29593	33715	37222	...	18944	10240	6400	3328	8704	9728	14336	2560	8192	20210721_1
3	20210721_1_1_acq4	2048	0	0	0	0	3840	6400	9472	9216	...	20224	10240	3328	256	1536	256	4352	0	5888	20210721_1
4	20210721_1_1_acq5	3840	1344	0	0	0	0	0	0	0	...	16844	11084	7372	5120	2867	2022	1177	102	0	20210721_1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4925	20220615_11_1_acq12	451584	484300	536678	591692	645376	693504	736000	774860	808192	...	12339	9267	5836	2560	1740	4300	7065	9164	12032	20220615_11
4926	20220615_11_1_acq13	436992	476620	521164	581734	633190	676480	715878	750515	780083	...	13312	9523	4915	0	0	0	0	1024	5120	20220615_11
4927	20220615_11_1_acq14	424192	461977	513689	569651	621414	666496	707379	745548	777318	...	11110	12441	10649	9088	5273	2380	460	0	5376	20220615_11
4928	20220615_11_1_acq15	299008	318822	334131	366617	400332	431488	460492	485452	504217	...	5171	3788	1996	0	0	0	0	0	0	20220615_11
4929	20220615_11_1_acq16	11264	2816	0	0	0	0	0	768	3072	...	0	0	0	0	0	0	0	0	0	20220615_11

Figure 5.2: The acquisition dataframe for the range 1.

A part for the acquisition datasets we also need the patient dataset to be able to identify the positive from the negative. This information can be taken from a csv file provided by the hospital, in this document we can find information about the date and the file that identifies the single ASC file, also for each entry it's specified the COVID status and if the patient is healed from COVID and if it is, in which date. It also keeps the information about if the particular ASC file is valid or if there was problems during the acquisitions. The raw dataset presents breath samples of 312 patients, divided into 93 positive records and 219 negative records.



	Date	File	Covid	Healed	Comment	CovidDate	Validity(1=Valid;0=NonValid)	index	Range	Covid Days	Covid_Healed
4	2021-03-10	2_4	0	0		NaT	1	20210310_2	4	NaT	0
9	2021-03-10	3_4	1	0		2021-03-01	1	20210310_3	4	9 days	1
19	2021-03-19	2_4	0	0		NaT	1	20210319_2	4	NaT	0
26	2021-03-19	4_4	0	1		2020-11-25	1	20210319_4	4	114 days	1
33	2021-03-19	6_4	0	1		2020-10-28	1	20210319_6	4	142 days	1
...	...	...	...	...	...	...	...	...	...	...	...
1578	2022-06-15	7_4	0	0		NaT	1	20220615_7	4	NaT	0
1582	2022-06-15	8_4	0	1		2022-03-16	1	20220615_8	4	91 days	1
1586	2022-06-15	9_4	0	0		NaT	1	20220615_9	4	NaT	0
1590	2022-06-15	10_4	0	1		2022-02-24	1	20220615_10	4	111 days	1
1594	2022-06-15	11_4	0	0		NaT	1	20220615_11	4	NaT	0

312 rows × 11 columns

Figure 5.3: The patient dataframe and its composition.

5.2 Data Visualization

In this section we examine the database composition visualizing the samples distribution with the help of the T-SNE (Section 2.3.1) and of the PCA (Section 4.3.2) algorithms. In order to have a visual comparison between the positive and the negative patients we also plot some samples belonging to the two label classes. The graphical impact of the normalization techniques is shown in Section 5.4.

The first analysis compares the positive and the negative classes, in Fig. 5.4 we can see the difference between 20 spectra of each class randomly taken.

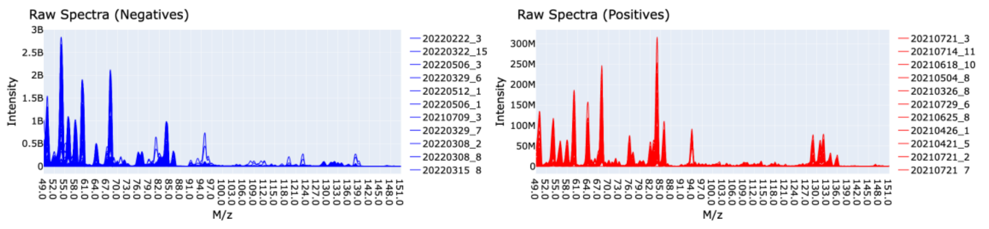


Figure 5.4: The comparison between the negative raw spectra and the positive one.

In order to highlight the distribution of each patient we can employee the T-SNE technique to plot in a 2D representation the positive and negative samples. Since we are dealing with not-normalized spectra the day bias is still present and the day clusters should be well-visible. To avoid misunderstanding we have to point out that in t-SNE plots, distance between points is not represented in a direct or absolute manner as it would be in, for example, a scatter plot of two continuous variables. Instead, t-SNE aims to preserve the pairwise similarity between data points and local structures and relationships between samples.

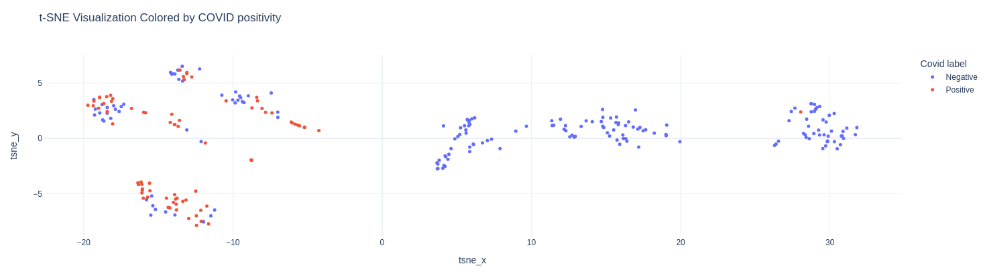


Figure 5.5: TSNE - distribution of positive and negative patients.

To highlight the benefit of the data augmentation technique shown in Section 4.4 we also plot the t-SNE representation of the augmented version of the dataset.

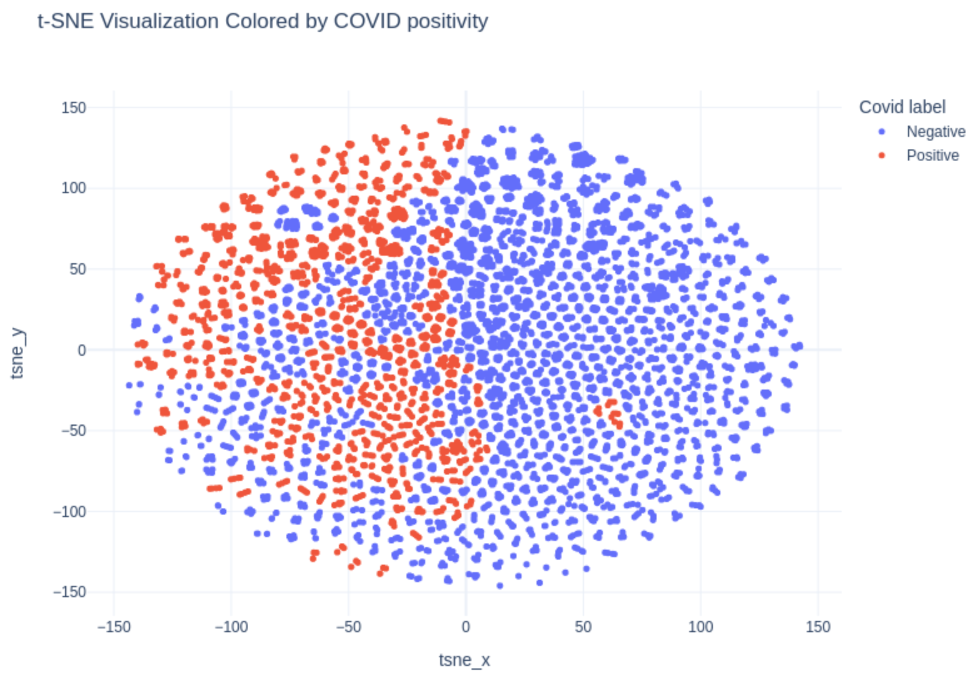


Figure 5.6: TSNE - distribution of positive and negative patients after the application of the data augmentation technique.

The Figure 5.6 shows the distribution of the artificial patient samples made by the combination of different ranges of the same patient taken in the same day. As we can see the classes are still well separated. The clusters are made by the derived versions of the original acquisition, this is a good sign since it means that are plausible mass spectra.

5.3 Outliers identification

In order to identify the outliers inside our data we exploit the air composition, in particular we know that some elements that are present in the air, are not involved inside the metabolism, so we should have a constant quantity (or in our case ratio) inside our breath. To understand better our method we should before talk about the isotopic ratio in mass spectrometry, it refers to the relative abundance of different isotopes of an element found in a sample. Isotopes are atoms of the same element that have the same number of protons in their nuclei but a different number of neutrons, leading to differences in their atomic masses. The ratios between each isotope should be nearly constant if we talk about pure elements like Krypton, but it should happen that some mass fragments coming from fragmentation of other masses could interfere with the measure adding some quantity to that isotope. This phenomenon is called isobaric interference, in mass spectrometry, the mass spectrometer measures the mass-to-charge ratio (m/z) of ions. If there are other ions with the same m/z value as the ions of interest, they can interfere with accurate measurement. In our analysis we chose the Kr for two reasons:

- It is an inert noble gas and is not typically involved in metabolic processes in living organisms. Unlike elements such as carbon, hydrogen, oxygen, and nitrogen, which are essential components of organic molecules and play crucial roles in biological processes, krypton does not have a recognized role in metabolism. Noble gases like krypton are chemically inert, meaning they do not readily react with other elements or molecules under normal physiological conditions. As a result, they are not typically incorporated into the biochemical reactions that drive metabolic processes.

- it's present in low concentration but big enough to overcome the presence of fragmentation of other isotopes.

To know about the relative ratio we will exploit the official documentation provided by the NIST organization.

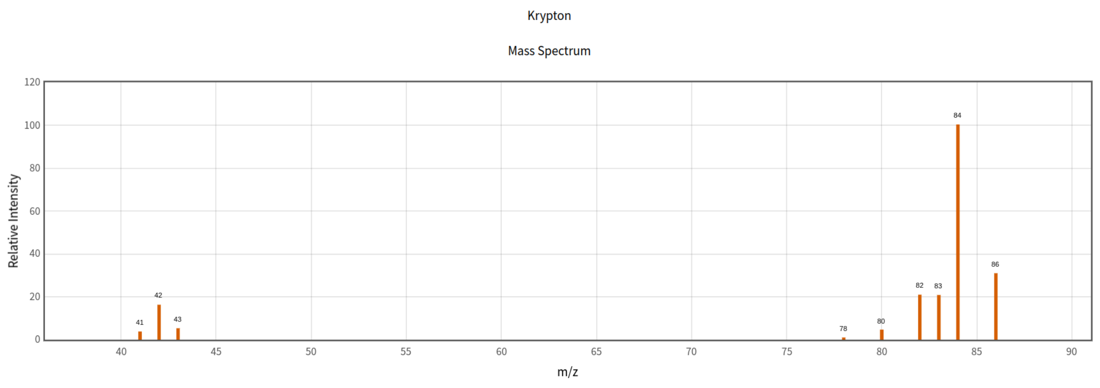


Figure 5.7: Krypton isotopic ratios from the NIST organization. The reference is peak the 84 m/z since it saturates to 100, the other peaks that we are taking in consideration are the Kr-82, Kr-83 and Kr-86.

As we can see from the figure above we take as reference the Kr-84 isotope and we will consider only the three mayor one: the Kr-82 and the Kr-83 that should both be 20% respect to the Kr-84 and the Kr-86 that should be the around the 30%. This analysis will be done on the acquisitions taken from the second range, the one that goes from 51 to 151 AMUs. To observe this ratios we will proceed in the following way:

- We first take the peaks in correspondence of the considered AMUs, to do this we exploit the ready made function find_peaks of the `scipy.signal` module. If the function cannot find a valid peak, we will consider the maximum values inside the range $(AMU - tollerance; AMU + tollerance)$ where $tollerance$ is a parameter, in our case we will use 0.5.

- After that we will assign the peak to the relative AMU, in particular if there are multiple peaks inside the previous range detected by the function, we will assign the peak nearer to the AMU and discard the other.

- Eventually we will compute the ratio between each isotope respect to the reference one (Kr-84).

The first run will be done on the air measures, in this way we can see if the machine had an unexpected behavior during a particular day and so all the measures have to be discarded.

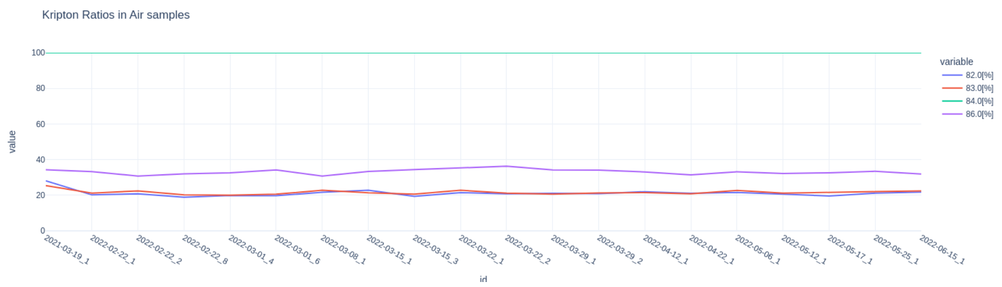


Figure 5.8: Krypton ratios inside air acquisitions.

As we can see, the ratios are seemingly constant so the acquisitions are well taken and we can consider this approach valid. One thing that we can notice from the plot above is that in the case of the isotopes Kr-82 and Kr-84 the relative abundance is a little higher respect to the 30% provided by the NIST, but this is acceptable since it's possible that different machine may detect more precisely an isotope respect to an other or it's possible that fragments of other isotope interfered during the acquisition. The important outcome of this preliminary analysis is that the three ratios are nearly constant during the days. Given that we can proceed with the outlier identifications and focus on the patients acquisitions. The procedure is the same as before and results are shown in the figure below.

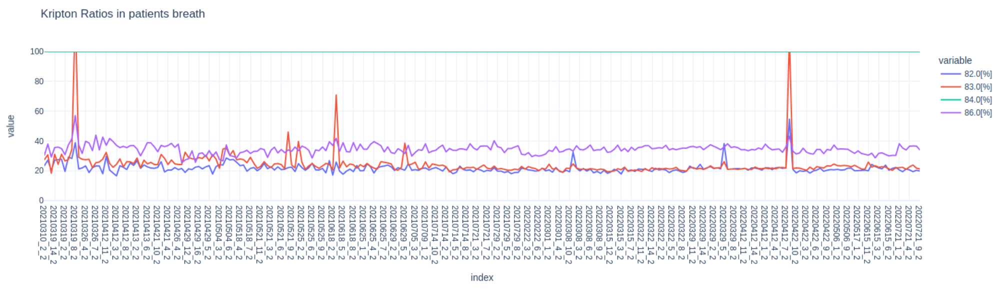


Figure 5.9: Krypton ratios inside patients acquisitions.

Unlike the air acquisitions, in this case we can see some anomalies, in particular we can see 5 red peaks that are outside their acceptable value, those are relative to 20210319_8, 20210521_8, 20210525_1, 20210618_3 and the 20220412_7 patients and will be classified as outliers. Since for our study we are using breath samples collected from patients and medical personnel from the hospital, we also have access to the file of the acquisitions. In this document are inserted comments and the validity of the acquisition, so beside the previous method we will consider only the measures without anomaly comment and the validity field checked inside the document. Discarding the problematic measures and the ones that don't respect the Kr-ratios method we will obtain a partially cleaned dataset.

Another method applied to identify and discard the outliers is the application of the z-score. The z-score (also called standard score) helps to understand if a data value is greater or smaller than the mean of a feature and how far away it is. More specifically, this score tells how many standard deviations away a data point is from the mean of the feature points.

$$zscore(x) = \frac{x - \mu_X}{\sigma_X}$$

where:

- x is the data sample on which we compute the z-score
- μ_X is the mean of the feature in consideration
- σ_X is the standard deviation of the feature taken in consideration

In our study we adopt a conservative approach discarding the spectra that have at least a feature with a z-score greater than 8. The results of such procedure can be seen in Figure 5.10.

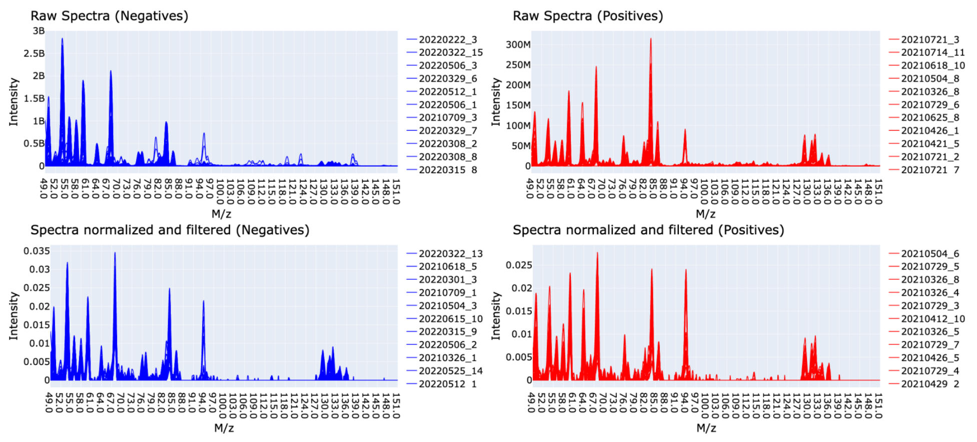


Figure 5.10: Comparison between raw features and normalized and filtered spectra.

5.4 Normalization

In this section we examine some normalization techniques in order to reduce the day bias present in the dataset. In machine and statistical learning literature a common approach is to subtract the mean and divide by the standard deviation of each feature, this is implemented by the `StandardScaler`, but we will also examine other methods more peculiar of the mass spectrometry field. One of this solution plans to use the mass spectrum intensity value (the TIC) as the divisor of the normalization process. In practice we divided each intensity value by the TIC to obtain the relative intensity of the compound respect the whole spectrum. Another technique that we will examine in this section is called compound normalization and it's explained in 4.2.4. In this method we used the Krypton-84 intensity value to normalize the whole spectrum, the obtained values represent the relative abundance of each AMU respect to the representative coefficient (Kr-84).

The effects of each method are visualized with the help of the TSNE algorithm (Section 2.3.1) in order to better understand the effects of each normalization technique. These representations show both the samples coloured by date and by COVID-19 positivity to better evaluate both the classification separation and the date bias effects. The first plot shows the distribution of the raw features without any kind of normalization in order to visualize the starting point of the analysis.

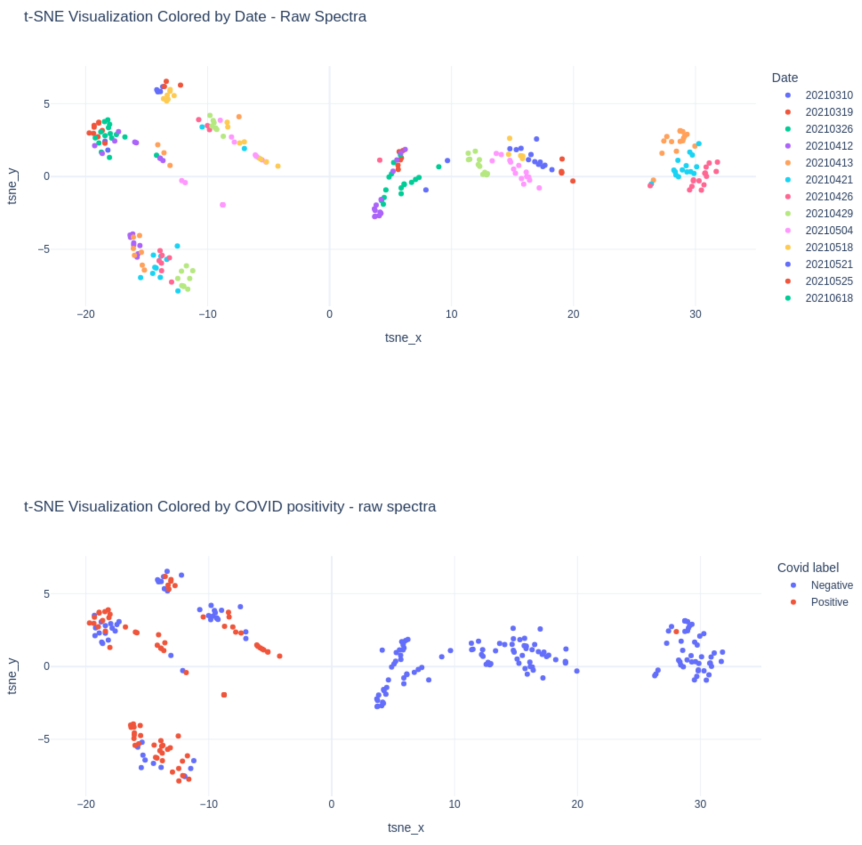


Figure 5.11: Distribution of the patients' spectra both colored by date and COVID-19 positivity.

As we can see from Figure 5.11 is evident the day bias problem in the first plot because the clusters are well defined and the distribution of the dates follows the calendar order: on the left we have the dates of 2021 growing until the last acquisition of the 2022 on the extreme right. This is the starting point of our analysis and from now on we try to evaluate the effects of each normalization technique.

The first normalization method that we developed was the TIC normalization also explained in Section 4.2.3. This method involves the use of the TIC as denominator to scale the samples in order to obtain the relative abundance of each compound respect the sum of the intensities of the whole spectrum. The results of this technique are shown in Figure 5.12.

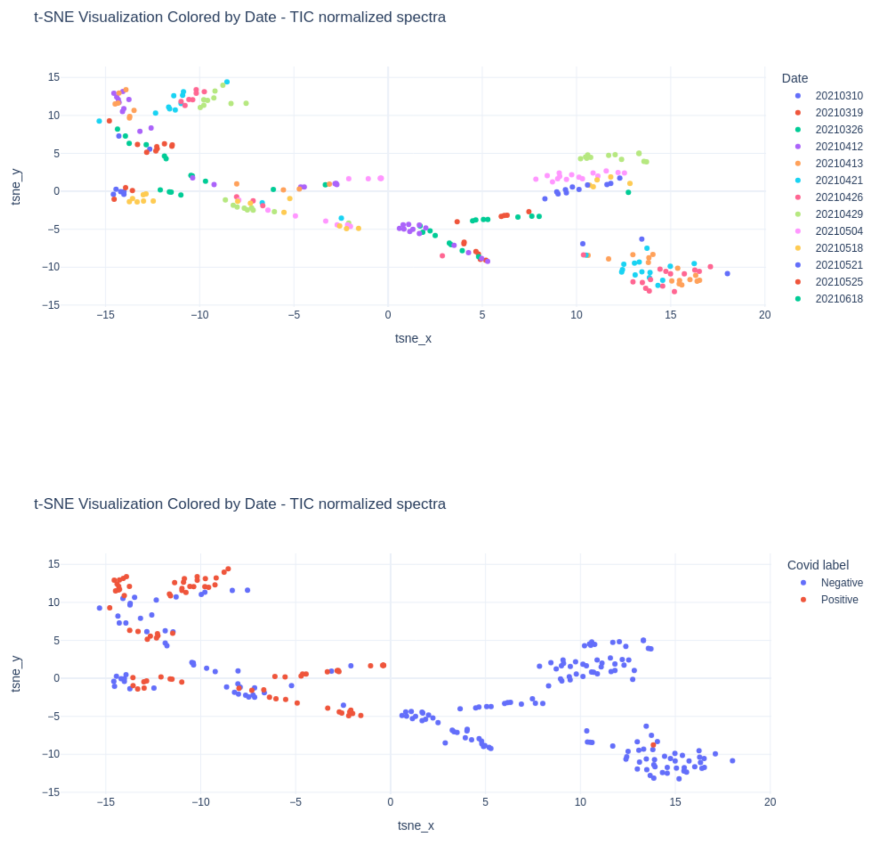


Figure 5.12: Distribution of the TIC normalized patients' spectra both colored by date and COVID-19 positivity.

The plots show that this normalization technique compacts the distribution of the patients' spectra, this can be seen by the smaller range of values in Figure

5.12 respect to the Figure 5.11. A part from that, we can also consider the fact that samples are starting to mix together and this is a sign of the reduction of the impact of the day bias.

The last analysis involves the usage of both the TIC normalization and Krypton normalization techniques. The Krypton normalization technique uses the Krypton 84 isotope as reference to normalize the whole spectrum, in this way the result of this operation is a spectrum made by relative intensities respect to the Kr-84. The results of such procedure can be seen in Figure 5.13.

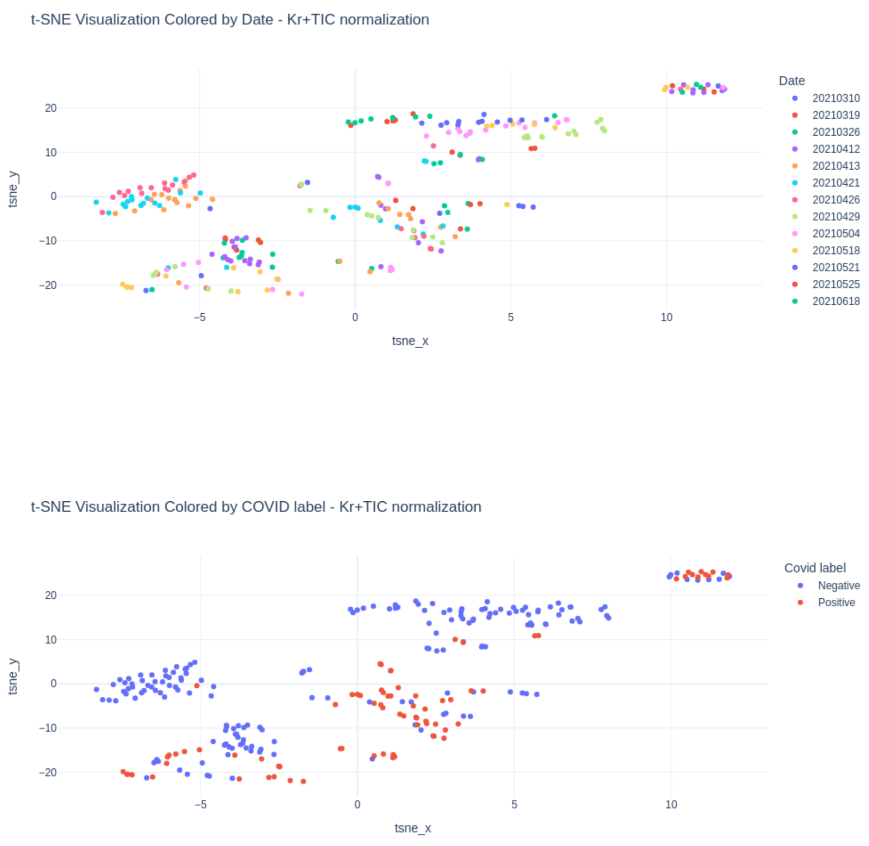


Figure 5.13: Distribution of the TIC and Krypton-84 normalized patients' spectra both colored by date and COVID-19 positivity.

This kind of normalization provided the best results both in terms of classification as we can later see in Section 5.6 and in terms of day bias reduction. As we can especially see in the Date visualization part of the plot the days are well-mixed, but the class are still well-separated. Also, the representation is more compact than the two before so the patients' mass spectra are more similar between each other.

5.5 Feature Selection

The feature selection part is one of the crucial step to improve the classification task, in particular this is the principal way to solve the problem of the course of dimensionality. In this study we mainly focused on two machine learning methods: PCA and Gradient Boosting and on a classical mass spectrometry method that is the peak selection. In the classification part we first choose the empirical method, so if we want to work with the whole spectra or only with the peaks and then the method to select the features: PCA or XGBoost. The method that is activated by default is the variance threshold, as we explained in Section 4.3.4, it is useful to discard those features that have a low impact on the overall variance of the correspondent feature.

5.5.1 PCA

The PCA is a dimensionality reduction method that project the dataset into a space of lower dimensionality. The output of this method can be the input of a classification task or of a clustering algorithm. The library used for this method is `Sklearn.decomposotion` module, since scaling the data is necessary before applying the dimensionality reduction we also used a scaler like `RobustScaler` or `StandardScaler` alway taken from the `Sklearn` library.

When we apply PCA to the original dataset with k features to get a transformed dataset with p number of variables (also called principal components) we have to specify as parameter of the `PCA` algorithm the `n_components` (the equivalent of p). Since this value is an hyperparameter, it is not learned from the data, so we have to manually tune it before running the `PCA()` function. The classical empirical method is based to the fact that each principal component explains a part of the variance, since we are interested to reduce the dataset dimensionality but also to keep most of the information, the classical choice is to select `n_components` that explain 95% (cut-off threshold) of the cumulative variance.

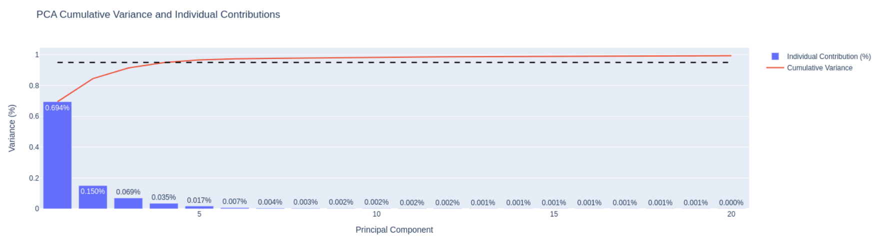


Figure 5.14: The cumulative variance is represented in red while each bar represents the individual variance contribute to the whole variance.

Exploiting the results of the plot shown in Figure 5.14 we should take in consideration 5 as `n_components`, but this could be a too aggressive dimensionality reduction. For this reason we chose as `n_components` 20 features in each experiment that deals with the whole spectrum as input.

5.5.2 XGBoost

The overall explanation of this method is explained in Section 4.3.3. A benefit of using gradient boosting is that it is relatively straightforward to retrieve the importance score of each feature after the decision trees are built. In this method, instead of the variance contribution of each attribute, it is considered the importance: the more an attribute is used to make key decisions inside the boosted trees, the higher is the relative importance.

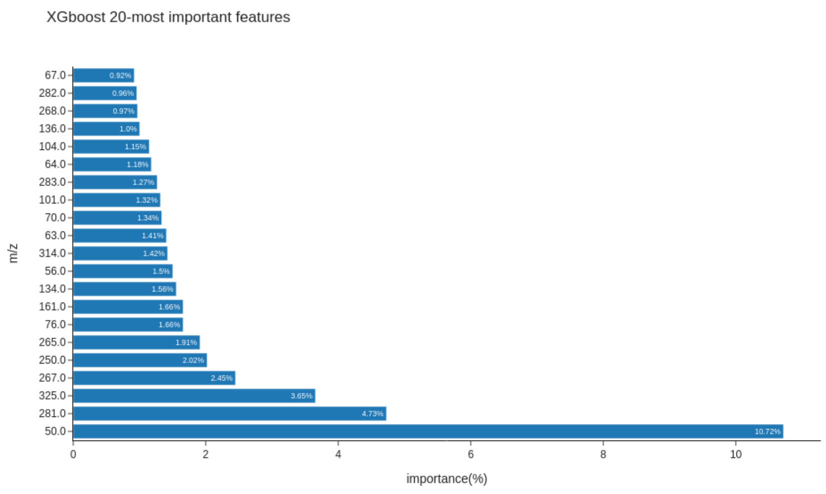


Figure 5.15: Feature selection with xgboost - only the 20 most important features are shown

For this step we used the `Python` package called `xgboost`. The procedure involves the usage of the `XGBClassifier` class to instantiate the classifier, after that we fitted a validation set of our data to extract the feature importances from the dataset attributes. To access to these values we used the `feature_importances_` attributes of the `XGBClassifier` fitted model. An important note has to be put in evidence: the results of this method may vary given the stochastic nature of the algorithm. Also, as in PCA (Section 5.5.1), the number of selected features is an hyperparameter, but differently respect to the other method, here we have to train our classification models with different number of features selected by importance and see the performance of the classification task. Nevertheless is possible to build a plot of the dataset features and their importances to access the most important features selected by this classifier. The results of this method are plotted in the Figure 5.15.

5.5.3 Convolutional Autoencoder

In this section we discuss the practical implementation of the Convolutional autoencoder and its architecture, we also focus on the train phase and the validation part of the neural network. The actual code is implemented using the `tensorflow` framework and its principal components such as `Conv1D`, `BatchNormalization`, `MaxPooling1D` and `UpSampling1D`.

- `Conv1D`: This layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. The configuration of this layer requires the specification of the `filters` parameter that indicates the dimensionality of the output space (the number of output filters in the convolution), the `kernel_size` parameter which specifies the length of the 1D convolution window and the `padding` that is set to `"same"` to pad with zeros such that the output has the same dimension of the input. The activation function, encoded with the parameter `activation`, is set to `"relu"`, in particular this corresponds to the *ReLu* activation function:
$$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$$
  In addition we use the `kernel_regularizer` parameter with the L2 regularizer which is taken from `tensorflow.keras.regulirizers` with penalty equals to 1^{-5}.

- `MaxPooling1D`: This layer is used to downsample the input representation by taking the maximum value over a spatial window of size `pool_size`. The resulted output has a shape of
$$output\_shape = \frac{input\_shape - pool\_size + 1}{strides} \tag{5.1}$$
  In our case the `stride` is always set to 1 and the `pool_size` to 2 or 3 depending on the input dimensionality. The `padding` parameter assumes always the value `"same"` as in the convolutional layers.

- The `UpSampling1D` is a layer that does the opposite operation respect to the `MaxPooling1D`, is used inside the decoder network to augment the dimensionality of the reconstructed data.

- The `BatchNormalization` layer applies a transformation that maintains the mean output close to 0 and the standard deviation close to 1. This layer is applied after each convolution.

The input of the Convolutional Autoencoder (CAE) is a 1D array of dimensionality 3024. The original dimension of the spectra is 3021, but we noticed that after the first downsampling we got 1007 as output dimension which is not divisible by 2 or 3. Since we didn't want to apply a bigger sampling due the risk of information loss we evenly padded our data to 3024 to be able to implement a deep neural network without the need of adding padding layer inside our architecture or large `pool_size` values. Following these criteria we can individuate the basic block of the CAE's architecture composed by a convolutional 1D layer, a batch normalization layer and according if we are in the encoder part or in the decoder part, respectively a max pooling or an up sampling layer. For compressing mass spectra, a CAE consisting of 33 layers has been designed. in the encoder section the signal is compressed, in the decoder part the starting signal is reconstructed and denoised. The overall architecture is described in the Table 5.1 where is possible to inspect the output size of each layer and the composition of the CAE.

To train the model we used as input the padded version of the dataset with features dimension equals to 3024, in addition to that we also add some Gaussian random noise to the training samples. The idea behind is to make the net being able to learn how to remove the random noise and reproduce the denoised spectrum. So, instead of copying the input spectrum this net will learn also to remove the artificial random noise. This practice can help improve the generalization and the global accuracy of the neural network for several reasons:

- it helps **regularization**, the noise acts as a form of regularization, which can prevent the autoencoder from overfitting the training data. By introducing noise, we forced the CAE to learn more robust and general features.
- It also helps **generalization**. By training on noisy data, the CAE learns a more generalized version of the data distribution. This can lead to better performance on unseen data during testing because the model learned how to extract features that are less specific to the training test.

To add the noise to the padded training samples we used the `numpy Python` package, in particular we used the `normal` procedure taken from `random` module. The added noise has a mean of 0 and a variance of 0.001. The low variance value is justified by the choice of normalize the training data with the TIC normalization (see Section 4.2.3). The normalized samples have an approximate variance of 0.003, so our choice is consistent with the training data.

For the training of the convolutional autoencoder 70% of the data were used for the training while the other 30% were used for the validation. For all the results of the experiment, the training phase of the model was performed by using 24 as batch size for 20 epochs. For the model optimization we pick the Adam optimizer with setting parameter $\beta_1 = 0.9$, $\beta_2 = 0.999$, the learning rate set to $1e^{-4}$ and $\epsilon = 1^{-7}$.

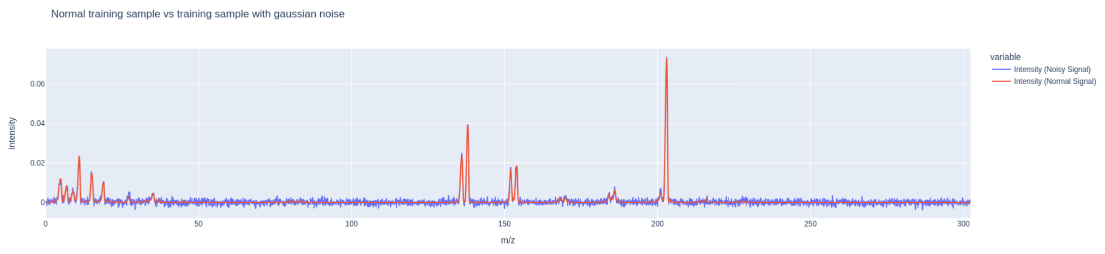


Figure 5.16: The difference between a normal spectrum sample and the same signal with the addition of a normal Gaussian noise with $\sigma = 0.001$ and $\mu = 0$.

A class balancing is applied during the training phase using the under-sample technique for the majority class. Using the augmented dataset with the combination of various spectra, as explained is Section 4.4, we had 12288 samples for the training dataset and 183 patients for the test part. In the validation phase we use a lower number of samples in order to use the mean of the acquisitions to maintain the procedure the most similar respect to the classification task. The losses of the training and validation phases are shown in Figure 5.17, it's visible that the two MSE are comparable and follow the same trend.

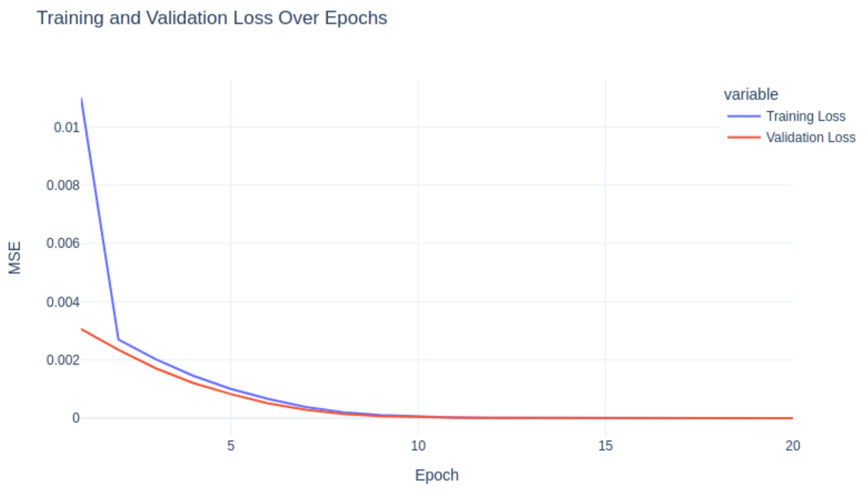


Figure 5.17: Training vs Validation loss of the CAE training.

To see the the goodness of the Convolutional autoencoder we can try to reconstruct a sample from the validation set and assess if the two spectra are similar. The results is shown in figure 5.18 where it's also present the reconstruction error. As can be seen from the plot, the original signal was successfully reconstructed with the proposed compression model. Since the reconstruction error is nearly visible, we can be satisfied of the results and use the convolutional autoencoder to extract features.

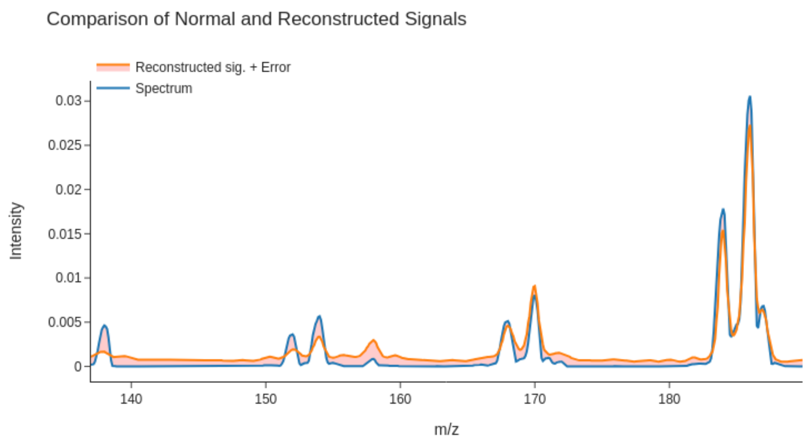


Figure 5.18: Difference between a validation spectrum and the reconstructed version, we show only a slice of the data in order to appreciate the reconstruction error better.

We also consider different metrics taken from [13]: the performance measures are as follows: root means squared error (RMS), percentage RMS difference (PRD) and PRD normalized (PRDN). The calculation of these criteria were performed on the reconstructed output signals compared to the original mass spectra of the designed CAE.

- Root Means Squared (RMS): It is a widely used method to determine the variance between the output predicted by the model and the original signal. In this study we used the Eq. 5.2:

$$RMS = \sqrt{\frac{\sum_{i=0}^{D-1} (S_o(i) - S_r(i))^2}{D-1}} \tag{5.2}$$

  where S_o represents the original signal and S_r the reconstructed one, this values should be as low as possible.

- Percentage RMS Difference (PRD): it is used to compute the quality of the reconstructed data in the compression. For a good quality compression it should be as low as possible.

$$PRD(\%) = 100 \cdot \sqrt{\frac{\sum_{i=0}^{D-1}(S_o(i) - S_r(i))^2}{\sum_{i=0}^{D-1}(S_o(i))^2}} \tag{5.3}$$

- PRD Normalized (PRDN): it's a performance metrics that is the normalized version of the PRD. It is calculated as the PRD but it subtract the average value of the original signal at denominator.

$$PRDN(\%) = 100 \cdot \sqrt{\frac{\sum_{i=0}^{D-1}(S_o(i) - S_r(i))^2}{\sum_{i=0}^{D-1}(S_o(i) - S_m)^2}} \tag{5.4}$$

- Signal to Noise Ratio (SNR): it is the measurement criteria used to compare the level of the background noise with the level of the desired signal.

$$SNR(dB) = 10 \cdot \log_{10} \frac{\sum_{i=0}^{D-1}(S_o(i) - S_m)^2}{\sum_{i=0}^{D-1}(S_o(i) - S_r(i))^2} \tag{5.5}$$

These metrics are applied on the validation set to test the compression process, Figure 5.19 shows the performance of the CAE for 8 patients plus the metrics means. The average parameters obtained for all the test samples are: PRD=26.67%, PRDN=27.86%, SNR=11.51dB and the the RMSE=9,1 $\times 10^{-9}$. This values are acceptable and show a good reconstruction quality especially the low level of the RMSE.

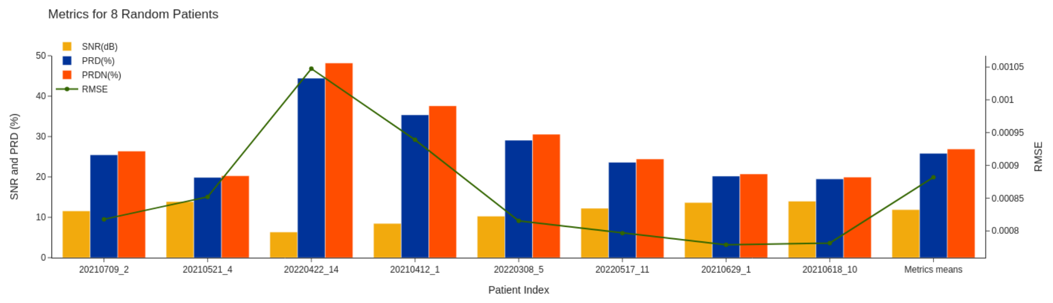


Figure 5.19: Reconstruction metrics for 8 random validation patients' spectra and the means of each metric. The plot uses a double axes to represent the lower scale of the RMSE, while the plot bars refer to the right y-axis

5.6 Classification Results

In this section we present the results of the classification task focusing on the metrics explained in Section 4.5.1. In each classification experiment we used a different kind of normalization, feature extraction or selection method, following this protocol we can compare each machine learning model and chose the best configuration to treat our data. For all the models and experiments the basic configuration used:

- the Savitzky-Golay smoothing algorithm with window size equals to 10 and the order of the fitting curve set to 2,
- the baseline correction with the ALS optimized algorithm with parameters $\lambda = 10^3$ and $p = 0.001$,
- the peak alignment algorithm enabled.

A part from that the chosen algorithms are taken from the Background chapter and their implementation comes from the `scikit Python` library. In particular we will use:

- the `KNeighborClassifier` for the KNN classification algorithm with `n_neighbors=5` and as distance metric we use the Minkowski estimator.
- for the Random Forest Classifier we chose the `RandomForestClassfier` classifier with `random_state=1` and the other parameters set with their default values.
- the `LogisticRegression` class implements the Logistic Regression algorithm, for this model we chose the default `scikit` configuration.
- for the implementation of the gradient boosting classifier we use the `GradientBoostingClassifier` class with parameter `min_samples_leaf=2`, `n_estimators=500`, `max_depth=3` and `learning_rate=0.1`.
- for the SVM classifier we used the `SVC` class with the RBF kernel with the automatic gamma selection configuration (`gamma="auto"`).
- the ensemble learning algorithm exploits the `VotingClassifier` with the `estimators` value set with the previous models instances and the `voting` parameter to `"hard"` to emulate the hard voting.

The first experiments are run separating each range in order to know the most discriminating ranges between the 4 available. The configuration for this classification used a stratified k-fold approach with 10 folds and the date value as stratified sampling criteria. This validation protocol uses a 70% of the data to train the model, 20% for the validation and the 10% for testing. Also, we use a class balancing with undersampling and the means of the patients' acquisitions in the day as spectra to build the datasets. In this configuration the dimensions of the dataset is different since the spectra for all the ranges are not available for all the patients. To know which is the most expressive range we run 10 runs of the the Random Forest (RF) and of the Ensamble learning (Ens) classification algorithms and then we took the average of the performance metrics.

Model	Mass Range N.	Accuracy	Precision	Recall	F1-score
RF	1	0.78	0.65	0.56	0.60
RF	2	0.82	0.69	0.69	0.68
RF	3	0.82	0.72	0.57	0.63
RF	4	0.79	0.65	0.53	0.57
Ens	1	0.82	0.70	0.67	0.68
Ens	2	0.84	0.75	0.69	0.71
Ens	3	0.80	0.68	0.57	0.61
Ens	4	0.77	0.62	0.60	0.59

Table 5.2: Average results with different mass ranges on the test set on 10 runs, no features selection or pre-processing.

As we can see from the Table 5.2 the most discriminating range is the second range, for this reason we first ran our experiments on this range and then we evaluate the performances on the whole spectrum to see the improvements.

5.6.1 Range 2 analysis with fractional AMUs

In this section we will consider the only the range 2 AMUs and their fractional intensity values. The next experiments are focused to highlights the impact of the normalization techniques and the feature selection methods on the classification task. In all these tests we applied the variance threshold algorithm to reduce the number of features in those cases that the impact of the (fractional or integer) AMUs are not considerable inside the overall variance. So, considering the range 2, after the variance threshold method has been applied the number of features drop from 1021 to 611. A part from this, we also show the impact of the employment of all the acquisitions available for each patients during the training part respect to the usage of the mean of the mass spectra inside the acquisition date. Inside the table the training phase with the mean of the acquisitions is annotated with `Single` while the usage of the whole acquisitions dataset with their combinations is signed as `Multiple`. In this first phase we considered the raw spectra without any kind of normalization except for the `StandardScaler` to improve the classification models internal performances.

Alg.	Filter	Feat. Sel.	Acq.	Accuracy	Precision	Recall	F1
xGB	No	-	Single	0.81 ± 0.05	0.73 ± 0.08	0.75 ± 0.11	0.73 ± 0.06
xGB	No	PCA	Single	0.83 ± 0.07	0.74 ± 0.01	0.78 ± 0.11	0.75 ± 0.10
xGB	**Yes**	**PCA**	**Multiple**	0.86 ± 0.05	0.78 ± 0.05	0.82 ± 0.11	0.80 ± 0.07
xGB	Yes	xGB(fs)	Multiple	0.86 ± 0.08	0.78 ± 0.10	0.83 ± 0.15	0.80 ± 0.11
KNN	No	-	Single	0.86 ± 0.06	0.68 ± 0.08	0.91 ± 0.1	0.77 ± 0.08
KNN	**No**	**PCA**	**Single**	0.87 ± 0.04	0.71 ± 0.06	0.91 ± 0.10	0.79 ± 0.06
KNN	Yes	PCA	Multiple	0.85 ± 0.08	0.73 ± 0.12	0.82 ± 0.13	0.77 ± 0.11
KNN	Yes	xGB(fs)	Multiple	0.81 ± 0.11	0.60 ± 0.13	0.87 ± 0.19	0.71 ± 0.14
LR	No	-	Single	0.81 ± 0.05	0.61 ± 0.08	0.85 ± 0.08	0.71 ± 0.07
LR	**No**	**PCA**	**Single**	0.83 ± 0.04	0.60 ± 0.05	0.92 ± 0.91	0.75 ± 0.06
LR	Yes	PCA	Multiple	0.81 ± 0.06	0.56 ± 0.08	0.93 ± 0.11	0.70 ± 0.09
LR	Yes	xGB(fs)	Multiple	0.83 ± 0.04	0.57 ± 0.06	0.98 ± 0.06	0.72 ± 0.06
RF	No	-	Single	0.85 ± 0.04	0.71 ± 0.06	0.84 ± 0.10	0.76 ± 0.05
RF	No	PCA	Single	0.81 ± 0.10	0.71 ± 0.14	0.75 ± 0.19	0.73 ± 0.15
RF	**Yes**	**PCA**	**Multiple**	0.88 ± 0.05	0.78 ± 0.09	0.87 ± 0.07	0.82 ± 0.07
RF	Yes	xGB(fs)	Multiple	0.88 ± 0.07	0.80 ± 0.11	0.87 ± 0.13	0.82 ± 0.09
SVC	No	-	Single	0.82 ± 0.04	0.55 ± 0.06	0.98 ± 0.04	0.71 ± 0.04
SVC	No	PCA	Single	0.85 ± 0.08	0.66 ± 0.10	0.89 ± 0.13	0.75 ± 0.10
SVC	**Yes**	**PCA**	**Multiple**	0.86 ± 0.06	0.70 ± 0.08	0.89 ± 0.08	0.78 ± 0.08
SVC	Yes	xGB(fs)	Multiple	0.83 ± 0.04	0.57 ± 0.07	0.98 ± 0.04	0.72 ± 0.06
Ens	No	-	Single	0.87 ± 0.05	0.68 ± 0.07	0.93 ± 0.08	0.78 ± 0.07
Ens	No	PCA	Single	0.86 ± 0.08	0.70 ± 0.12	0.88 ± 0.11	0.78 ± 0.11
Ens	**Yes**	**PCA**	**Multiple**	0.88 ± 0.06	0.75 ± 0.12	0.89 ± 0.09	0.81 ± 0.10
Ens	Yes	xGB(fs)	Multiple	0.85 ± 0.06	0.62 ± 0.08	0.97 ± 0.05	0.75 ± 0.08

Table 5.3: Mean results of classification models on the test sets with mass range 2 considering any kind of normalization.

As we can see from the Table 5.3, the performances generally improve with the application of PCA and the filtering. This is an hint that the dimensionality reduction techniques can improve the performance of our classifiers, but still the F1-score is quite poor. For this reason we applied some normalization techniques to try to make the performances better.

In order to assess the performances of the classification models with different kind of normalization we repeat the analyses with the TIC normalization (Section 4.2.3) and with the Krypton normalization (Section 4.2.4). Since it's clear from the table 5.3 the key role of the feature selection algorithms we avoided the classification considering all the range-2 AMUs as features, but directly adopting one of the methods between PCA and the XGboost. Also, since it's clear the big impact of a larger training set we avoided the training experiments with the Single acquisition. Following this protocol, the results are shown in Table 5.4, since we always adopted the multiple acquisitions strategy we substituted the acquisition column with the normalization technique used (Norm. column in 5.4).

Alg.	Feat. Sel	Norm.	Accuracy	Precision	Recall	F1-score
KNN	PCA	TIC	**0.91 ± 0.04**	0.75 ± 0.13	**0.93 ± 0.07**	**0.83 ± 0.08**
KNN	PCA	Kr	0.85 ± 0.07	0.75 ± 0.15	0.81 ± 0.08	0.77 ± 0.11
KNN	PCA	Kr+TIC	0.90 ± 0.06	**0.77 ± 0.14**	0.90 ± 0.08	0.82 ± 0.09
RF	PCA	TIC	0.89 ± 0.06	0.80 ± 0.14	0.86 ± 0.09	0.83 ± 0.10
RF	PCA	Kr	0.86 ± 0.05	0.76 ± 0.13	0.83 ± 0.09	0.79 ± 0.08
RF	PCA	Kr+TIC	**0.90 ± 0.04**	**0.81 ± 0.09**	**0.89 ± 0.06**	**0.85 ± 0.06**
LR	PCA	TIC	0.87 ± 0.04	**0.73 ± 0.12**	**0.92 ± 0.09**	**0.80 ± 0.08**
LR	PCA	Kr	0.81 ± 0.07	0.60 ± 0.13	0.91 ± 0.10	0.70 ± 0.11
LR	PCA	Kr+TIC	**0.88 ± 0.04**	0.72 ± 0.13	0.91 ± 0.08	0.79 ± 0.09
xGB	PCA	TIC	**0.92 ± 0.04**	0.81 ± 0.12	**0.94 ± 0.09**	**0.86 ± 0.08**
xGB	PCA	Kr	0.86 ± 0.05	0.76 ± 0.07	0.84 ± 0.01	0.80 ± 0.06
xGB	PCA	Kr+TIC	0.89 ± 0.09	**0.82 ± 0.15**	0.87 ± 0.15	$0.84 \pm 0.0.13$
SVC	PCA	TIC	**0.89 ± 0.05**	0.76 ± 0.17	**0.90 ± 0.10**	0.81 ± 0.10
SVC	PCA	Kr	0.86 ± 0.08	0.78 ± 0.16	0.82 ± 0.12	0.80 ± 0.12
SVC	PCA	Kr+TIC	0.89 ± 0.06	**0.78 ± 0.15**	0.89 ± 0.12	**0.82 ± 0.09**
Ens	PCA	TIC	0.91 ± 0.05	0.81 ± 0.13	0.91 ± 0.08	0.86 ± 0.08
Ens	PCA	Kr	0.87 ± 0.06	0.76 ± 0.11	0.86 ± 0.08	0.81 ± 0.08
Ens	PCA	Kr+TIC	**0.93 ± 0.04**	**0.83 ± 0.12**	**0.93 ± 0.07**	**0.87 ± 0.07**

Table 5.4: Mean results of classification models on the test sets with mass range 2 using the proposed normalization methods (4.2.3 and 4.2.4) and PCA trained on multiple acquisitions.

Table 5.4 shows the results of the classification models on normalized data. Comparing the Table 5.3 with this one is pretty straightforward to see the big impact of the normalization methods on the classifiers performance. In particular we can see that the TIC normalization method is the overall best improving the classification metrics of about 5% respect the best results obtained previously. The Krypton normalization brings an improvement only to the logistic regression classifier balanced accuracy. The combined application of the TIC and Krypton normalization on range 2 improves a lot the ensemble model bringing the F1-score of the Table 5.3 from 81% to 87% and it also reduces the uncertainty of the measure.

5.6.2 Range 2 Analysis: peaks as features

In this section we examine the impact of using the peaks of each spectrum as features, with this approach we was able to reduce the number of features from

1021 to 102. In the Table 5.5 are shown the results of the classification using such a technique considering the multiple acquisitions during the training phase and the PCA as features selector. We also use the TIC normalization as normalization method. The peak picking is useful to concentrate the information spread in 10 fractional AMUs inside a single feature.

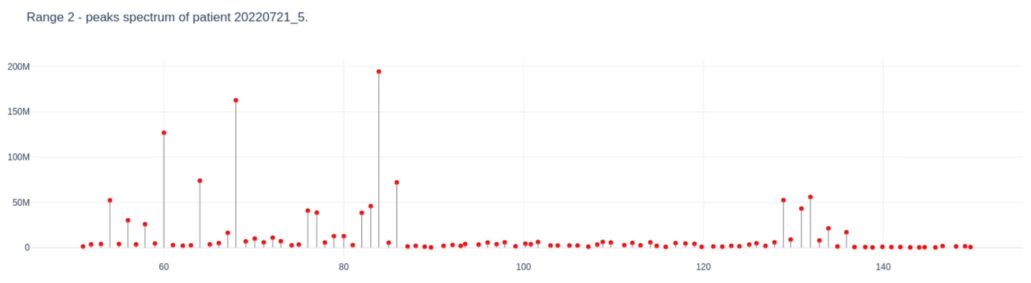


Figure 5.20: Patient 20220721_5 spectrum considering only the peaks as features.

To build the training dataset is used the same procedure explained in Section 4.3.1. As we can see from Figure 5.20 the number of features is much smaller and each peaks is related to its integer AMU, in this way each spectrum shares the same feature name set with the others.

Alg.	Feat. Sel	Accuracy	Precision	Recall	F1-score
KNN	PCA	0.89 ± 0.05	0.74 ± 0.14	0.92 ± 0.10	0.81 ± 0.09
RF	PCA	0.89 ± 0.09	0.80 ± 0.14	0.86 ± 0.17	0.82 ± 0.13
LR	PCA	0.85 ± 0.08	0.67 ± 0.13	0.89 ± 0.13	0.76 ± 0.11
xGB	PCA	0.91 ± 0.07	0.83 ± 0.12	0.90 ± 0.14	0.86 ± 0.10
SVC	PCA	0.88 ± 0.05	0.74 ± 0.12	0.89 ± 0.10	0.80 ± 0.07
Ens	PCA	0.90 ± 0.07	0.77 ± 0.12	0.90 ± 0.13	0.82 ± 0.10

Table 5.5: Mean results of classification models on the test sets with mass range 2 using the TIC normalization method and the peaks as features.

5.6.3 Full spectrum analysis

The last analysis consists of using the whole mass range 10-351 for each patients with multiple acquisitions inside the training set. Following this approach we have a much larger training set consisting of more than 20,000 samples and this should have an impact on the classification performances. To evaluate this protocol we consider the best configuration so far: the features selection with PCA, the normalization is done with the TIC normalization and the Krypton normalization.

Model	Accuracy	Precision	Recall	F1-score
KNN	0.93 ± 0.06	0.87 ± 0.09	0.92 ± 0.09	0.89 ± 0.08
RF	0.91 ± 0.06	0.88 ± 0.10	0.87 ± 0.12	0.87 ± 0.07
LR	0.94 ± 0.04	0.84 ± 0.12	0.96 ± 0.07	0.89 ± 0.07
xGB	0.94 ± 0.03	0.88 ± 0.08	0.93 ± 0.07	0.90 ± 0.03
SVC	0.93 ± 0.06	0.89 ± 0.09	0.90 ± 0.12	0.88 ± 0.06
Ens	$\mathbf{0.95 \pm 0.04}$	$\mathbf{0.90 \pm 0.08}$	$\mathbf{0.94 \pm 0.07}$	$\mathbf{0.92 \pm 0.05}$

Table 5.6: Mean results (10 splits) for the whole mass range 10-351 (augmented dataset).

As we can appreciate from the Table 5.6, the employment of the whole mass range further improves the classification performances boosting the F1-score to 0.92.

5.6.4 Classification with the CAE as feature extractor

In this last section we examine the effect of the feature reduction carried out by the encoder part of the trained CAE. Since we chose as latent dimension 8 the final shape of our data will be 63×8, considering that all the classification algorithms require 1D samples we flatten the output of the autoencoder into a single array of dimensions 504×1. The results of such a transformation are visible on Figure 5.21.

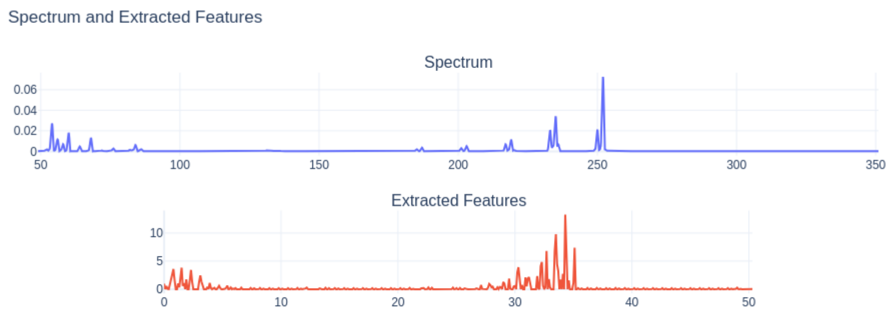


Figure 5.21: Comparison between a standard spectrum and the extracted features of the encoder part of the CAE. We plotted the decimals so to get the number of features we have to multiply the maximum value on the x-axis by 10.

The considered protocol for the classification with the CAE as feature extractor is a little different respect the previous one. In this case, since we trained the CAE with part of our dataset, we cannot again use those samples for the validation or testing parts for results bias reasons. For this motivation we used a part of our dataset to train the model and the net, a validation part to tune the hyper-parameters and a test partition to assess the quality of the classification. We executed this process 10 times and we computed the mean of each metrics during the different iterations and the standard deviations of each to evaluate the uncertainty. This is due the fact that the net made the optimization choices considering the validation test and computing the metrics on that partition may make an optimistic valuation.

Model	Accuracy	Precision	Recall	F1-score
KNN	0.91 ± 0.004	0.81 ± 0.016	0.91 ± 0.002	0.86 ± 0.009
RF	0.89 ± 0.008	0.85 ± 0.017	0.82 ± 0.020	0.84 ± 0.010
LR	0.87 ± 0.004	0.84 ± 0.012	0.90 ± 0.014	0.87 ± 0.012
xGB	0.92 ± 0.006	0.89 ± 0.018	0.91 ± 0.013	0.90 ± 0.011
SVC	0.90 ± 0.001	0.87 ± 0.009	0.91 ± 0.010	0.89 ± 0.018
Ens	$\mathbf{0.93 \pm 0.014}$	$\mathbf{0.90 \pm 0.013}$	$\mathbf{0.92 \pm 0.009}$	$\mathbf{0.91 \pm 0.05}$

Table 5.7: Mean results (10 iterations) for the whole mass range 10-351.

The results of the classification showed in Table 5.7 made use of the whole TIC normalized spectrum. The results are comparable with the ones shown in 5.6, in particular we can observe a lower uncertainty between the metrics. This method allowed us to use a simple K-fold cross validation since the CAE removed the day biases present in our acquisitions.

5.7 Long COVID analysis

In this final section we examine the impact of the COVID19 on healed patients' metabolism. Multiple studies [7], [6] and [17] have demonstrated that the impact of the COVID19 disease can affect the human metabolism also after the negativization. In order to assess the effect of that, the patients that are tested few days after the healing from the disease should still present some bio-markers symptoms that should be detected from the mass spectrometer. Those patients should also be considered positive by the classification models since they contain peaks that resemble positive patients.

The protocol to assess this behavior is straightforward, the idea is to train our model with the same configuration that we have examined in Section 5.6, after we have trained the model we tested it on the test partition, but this time we loop over a range that goes from zero to 100 days and we considered positive the patients that had been positive x days before being tested. Following this procedure we should see an increment of performance in the first iterations and a decrease when the amount of days is large enough to consider the patient completely healed. The first increase of performance is related to the fact that the patients are considered healed by the molecular tampon but still presented some bio-marker typical of positive patients detected by the mass spectrometer.

The experiment that we ran use a 10 stratified fold with stratified sampling on the `Date` column and also an oversampling on the minority (positive) class as before. To pursue our goal we added a specific column called `Days after healing` in order change the `Covid` label to those patients that were tested `days` after being positive. The simple formula that we used to transform the `Covid` label is:

$$f(days) = \begin{cases} 1 & \text{for } date_{COVID} - date_{HEALED} \geq days \\ covid & \text{otherwise} \end{cases} \tag{5.6}$$

where

- *days* is the value that grows inside the loop and it is used to iterate over the range $[0; 100]$ days.
- $date_{COVID}$ is the date when the patient has been tested with the molecular tampon
- $date_{HEALED}$ is when the patient has been tested negative after has been affected by the COVID-19.
- *covid* is the old value of the class label. It is the value returned is the patient cannot be considered affected by the long COVID since it had never contracted the disease or it doesn't fall back into the first case of Eq. 5.6.

The experiment employed only the Ensemble model for classification since it encloses all the other and at this point we it's the most performing. We also used the full augmented dataset and used the mean acquisitions as test set. The pre-processing techniques included the TIC normalization, the `StandardScaler` and the `PCA` with 20 features as features extractor.

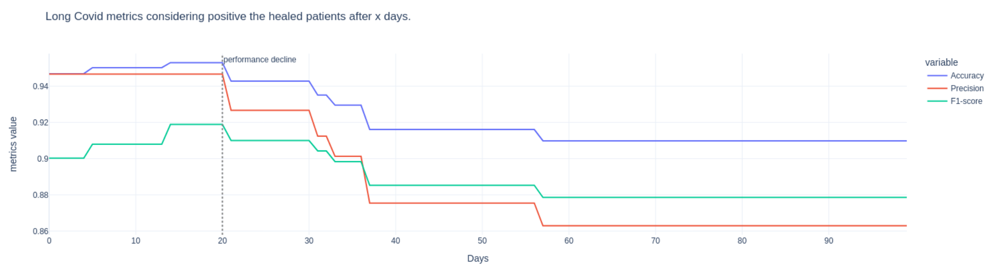


Figure 5.22: Long Covid effect on test samples. On the x-axis are shown the days after which we still considered as positive the patients, while on the y-axis the metrics value.

In Figure 5.22 we can see the results of our test, the behavior of the performances metrics is how we expected: the patients that have been healed in 20 days before the acquisition still presented some symptoms in their breath that affected the wrong classification by the models. After 20 days that the patients are healed (tested negative) the models worked well since changing the COVID19 label brought to a degradation of the performances. The precision metric is not shown since it obviously follows the opposite pattern of the other three metrics. From this analysis we can assess that some bio-markers are still presented in the patients that have been affected by COVID in the last 20 days.

Layer	No. of filter x kernel size	Unit size	Activation fun.	Output size
		Encoder Section		
Conv 1D	128 x 3	-	ReLu	3024 x 128
Batch Norm.	-	-	-	3024 x 128
Max Pool 1D	-	2	-	1512 x 128
Conv 1D	64 x 3	-	ReLu	1512 x 64
Batch Norm.	-	-	-	1512 x 64
Max Pool 1D	-	2	-	756 x 64
Conv 1D	32 x 3	-	ReLu	756 x 32
Batch Norm.	-	-	-	756 x 32
Max Pool 1D	-	2	-	378 x 32
Conv 1D	16 x 3	-	ReLu	378 x 16
Batch Norm.	-	-	-	378 x 16
Max Pool 1D	-	2	-	189 x 16
Conv 1D	8 x 3	-	ReLu	189 x 8
Batch Norm.	-	-	-	189 x 8
Max Pool 1D	-	3	-	63 x 8
Conv 1D	8 x 3	-	ReLu	63 x 8
		Decoder Section		
Conv 1D	8 x 3	-	ReLu	63 x 8
Batch Norm.	-	-	-	63 x 8
Up Sampl. 1D	-	3	-	189 x 8
Conv 1D	16 x 3	-	ReLu	189 x 16
Batch Norm.	-	-	-	189 x 16
Up Sampl. 1D	-	2	-	378 x 16
Conv 1D	32 x 3	-	ReLu	378 x 32
Batch Norm.	-	-	-	378 x 32
Up Sampl. 1D	-	2	-	756 x 32
Conv 1D	64 x 3	-	ReLu	756 x 64
Batch Norm.	-	-	-	756 x 64
Up Sampl. 1D	-	2	-	1512 x 64
Conv 1D	128 x 3	-	ReLu	1512 x 128
Batch Norm.	-	-	-	1512 x 128
Up Sampl. 1D	-	2	-	3024 x 128
Conv 1D	1 x 3	-	Sigmoid	3024 x 1

Table 5.1: Architecture of the Convolutional Autoencoder - The CAE is made by the repetition of the basic block identified in the Section 5.5.3, the chosen latent dimension is 8 has we can see by the dimensionality of the last layer. The output sizes of each layer are coherent with the pooling layers formula specified in (5.1).

Chapter 6

Conclusions

This study aimed to detect COVID-19 from breath mass spectra in order to deploy an accurate non-invasive model and to create a dataset from the `.ASC` files provided by the NanoTech Analysis S.r.l.

The creation of the dataset starting from the provided files allowed us to create a dataframe made by the intensity value of each m/z, the patient index, the COVID19 label and the date of the acquisition. The first version of this dataset is made by all the acquisitions for a given date of the patient. These acquisitions are taken directly from the particular mass range AMU file. In order to extract these measure a plateau detection algorithm has been developed (Section 5.1). One of the main improvement respect the previous works is the development of different signal pre-processing techniques able to improve the overall quality of the mass spectra. In particular we developed a smoothing procedure that employed the Savitzky-Golay filter, a peak alignment algorithm and a baseline correction method that make use of an ALS (Asymmetric Least Squares) algorithm [18] to detect and subtract the signal baseline.

The following step goal was to visualize the dataset and extract some knowledge from it. From this analysis we noticed a measurement bias connected to the day on which the acquisitions are taken and some outliers are detected via the z-score method and the Krypton-84 ratios analysis. To tackle the bias problem we proposed different normalization methods. The impact of these algorithms is evaluated plotting the projected results with the T-SNE method and the resulting best approach is the combination of the TIC normalization and the Krypton-84 normalization.

A part from this normalization methods we also developed a features extractor starting from a convolutional AutoEncoder (CAE). The neural network is trained as specified in Subsection 5.5.3 then the decoder part has been considered as feature extractor and it provided encouraging results reducing the starting 3021 dimensionality into a much lower embedding space of 504 features.

During the evaluation phase we noticed that range 2 is the most appropriate in predicting the presence of COVID-19. In this section we examined different feature extractor techniques trying PCA, xg-boost and the decoder of the CAE, also we evaluated the impact of the proposed normalization methods and the peak as feature procedure. The results of our study are positives since we were able to reach a 95% of accuracy and a 92% of F1-score using the whole spectra. The proposed method with the help of a portable mass spectrometer can be reputed an improvement respect to the current invasive method. Also, the convolutional autoencoder had very promising effect on reducing the higher dimensionality of the acquisitions typical of this field and can be further improved.

The last consideration concerns the long COVID analysis done in the last chapter, this experiment should be further evaluated by some field experts in order to assess the metabolic impact of the COVID-19 disease detected inside the breath samples. Our experiments demonstrates that patients that has been tested positive in the last 20 days still presented some bio-markers that are typical of positive patients since our models classified them as positive despite they have been tested negative with the molecular tampon.

Bibliography

[1] Aikaterini Liangou et al. «A method for the identification of COVID-19 biomarkers in human breath using Proton Transfer Reaction Time-of-Flight Mass Spectrometry». In: 42 (Nov. 2021) (cit. on p. 3).

[2] Nam K. Tran, Taylor Howard, Ryan Walsh, John Pepper, Julia Loegering, Brett Phinney, Michelle R. Salemi, and Hooman H. Rashidi. «Novel application of automated machine learning with MALDI-TOF-MS for rapid high-throughput screening of COVID-19: a proof of concept». In: (Apr. 2021) (cit. on p. 3).

[3] Yuan Zi-Cheng and Hu Bin. «Mass Spectrometry-Based Human Breath Analysis: Towards COVID-19 Diagnosis and Research». In: *Journal of Analysis and Testing* 5 (Dec. 2021), pp. 287–296 (cit. on p. 3).

[4] Jun She, Jinjun Jiang, Ling Ye, Lijuan Hu, Chunxue Bai, and Yuanlin Song. «2019 novel coronavirus of pneumonia in Wuhan, China: emerging attack and management strategies». en. In: *Clin. Transl. Med.* 9.1 (Feb. 2020), p. 19 (cit. on p. 4).

[5] URL: `https://www.cdc.gov/coronavirus/2019-ncov/science/science-briefs/sars-cov-2-transmission.html#:~:text=Inhalation%20of%20air%20carrying%20very,droplets%20and%20particles%20is%20greatest.` (cit. on p. 4).

[6] Hannah E. Davis, Lisa McCorkell, Julia Moore Vogel, and Eric J. Topol. «Long COVID: major findings, mechanisms and recommendations». In: *Nature Reviews Microbiology* 21.3 (Mar. 2023), pp. 133–146. ISSN: 1740-1534. DOI: `10.1038/s41579-022-00846-2`. URL: `https://doi.org/10.1038/s41579-022-00846-2` (cit. on pp. 6, 66).

[7] Daniel Carvalho de Menezes, Patrıcia Danielle Lima de Lima, Igor Costa de Lima, Juliana Hiromi Emin Uesugi, Pedro Fernando da Costa Vasconcelos, Juarez Antônio Simões Quaresma, and Luiz Fábio Magno Falcão. «Metabolic Profile of Patients with Long COVID: A Cross-Sectional Study». en. In: *Nutrients* 15.5 (Feb. 2023) (cit. on pp. 6, 66).

[8] Laurens Van der Maaten and Geoffrey Hinton. «Visualizing data using t-SNE.» In: *Journal of machine learning research* 9.11 (2008) (cit. on p. 11).

[9] Altman N. and Krzywinski M. «The curse(s) of dimensionality». In: *Nat Methods* (2018) (cit. on p. 19).

[10] Solveig Badillo, Balazs Banfai, Fabian Birzele, Iakov I Davydov, Lucy Hutchinson, Tony Kam-Thong, Juliane Siebourg-Polster, Bernhard Steiert, and Jitao David Zhang. *An introduction to machine learning.* Apr. 2020. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7189875/` (cit. on p. 21).

[11] Batta Mahesh. «Machine learning algorithms-a review». In: *International Journal of Science and Research (IJSR).[Internet]* 9.1 (2020), pp. 381–386 (cit. on p. 23).

[12] Nicolò Bellarmino, Riccardo Cantoro, Martin Huch, Tobias Kilian, Ulf Schlichtmann, and Giovanni Squillero. «Semi-Supervised Deep Learning for Microcontroller Performance Screening». In: *2023 IEEE European Test Symposium (ETS).* 2023, pp. 1–6. DOI: `10.1109/ETS56758.2023.10174083` (cit. on p. 25).

[13] Ozal Yildirim, Ru San Tan, and U. Rajendra Acharya. «An efficient compression of ECG signals using deep convolutional autoencoders». In: 52 (July 2018), pp. 198–211 (cit. on pp. 25, 58).

[14] Anestis Antoniadis, Jérémie Bigot, and Sophie Lambert-Lacroix. «Peaks detection and alignment for mass spectrometry data». en. In: *Journal de la société française de statistique* 151.1 (2010), pp. 17–37. URL: `http://www.numdam.org/item/JSFS_2010__151_1_17_0/` (cit. on pp. 31, 34).

[15] Takio Kurita. «Principal component analysis (PCA)». In: *Computer Vision: A Reference Guide* (2019), pp. 1–4 (cit. on p. 35).

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* `http://www.deeplearningbook.org`. MIT Press, 2016 (cit. on p. 37).

[17] D. C. Menezes, P. D. L. Lima, I. C. Lima, J. H. E. Uesugi, P. F. D. C. Vasconcelos, J. A. S. Quaresma, and L. F. M. o. «Metabolic Profile of Patients with Long COVID: A Cross-Sectional Study». In: *Nutrients* 15.5 (Feb. 2023) (cit. on p. 66).

[18] Eilers Hans F.M. Boelens Paul H. C. «Baseline Correction with Asymmetric Least Squares Smoothing». In: (2005) (cit. on p. 70).