Applying Design of Experiments and Machine Learning algorithms to define the consumption envelope of lactic acid

Marc Garcia Lopez

MU Bioinf. i Bioest.
Machine Learning

Tutor/a de TF
Daniel Fernández Martínez

Professor/a responsable de l'assignatura
Carles Ventura

15/01/2023

Universitat Oberta de Catalunya

UNIVERSITAT DE BARCELONA

Llicències alternatives (triar alguna de les següents i substituir la de la pàgina anterior)

A) Creative Commons:

B) GNU Free Documentation License (GNU FDL)

C) Copyright

Acknowledgments

I would like to acknowledge my supervisors Daniel Fernández Martínez and Sabrina Botton, for providing guidance and feedback throughout this project. Thanks also to my girlfriend Carlyn, for putting up with me being sat in the office for hours on end, and for providing guidance and a sounding board when required. Last but not least, I would like to thank my mother for supporting me and for always believing in me.

FITXA DEL TREBALL FINAL

Títol del treball:	*Applying Design of Experiments and Machine Learning algorithms to define the consumption envelope of lactic acid.*
Nom de l'autor:	*Marc Garcia Lopez*
Nom del consultor/a:	*Daniel Fernández Martínez*
Nom del PRA:	*Carles Ventura*
Data de lliurament (mm/aaaa):	*15/01/2023*
Titulació o programa:	*Màster Universitari en Bioinformàtica i Bioestadística UOC-UB*
Àrea del Treball Final:	*Machine Learning*
Idioma del treball:	*Anglès*
Paraules clau	• *Artificial Neural Networks* • *Biological pollutants* • *Consumption envelope* • *Design of Experiments* • *Microbial factories* • *Naive Bayes* • *Random Forests* • *Support Vector Machines*

Resum del Treball

Com a conseqüència de la crisis climàtica, la comunitat científica està dedicant cada vegada més esforços en el desenvolupament d'alternatives sostenibles als combustibles fòssils. Encara que no es tant conegut com l'energia eòlica o solar, l'ús de microorganismes i/o dels seus components cel·lulars per a la producció de biocombustibles i altres bioproductes està guanyant rellevància. No obstant, a llarg termini, aquests processos requereixin una important optimització per a que siguin econòmicament rentables. Històricament, aquesta optimització s'ha fet de forma un tant precària i a través de l'ús de mètodes subòptims (p.e. COST: *change one single variable at a time*). En qualsevol cas, si apliquem un disseny experimental i les eines computacionals adequades, podem obtenir resultats força robustos sense invertir ni massa temps ni massa esforç.

En aquest projecte, volem aplicar el disseny d'experiments (DoE) i alguns algoritmes de Machine Learning (Naive Bayes, *Support Vector Machines*, *Random Forests*, *Artificial Neural Networks*) per a optimitzar un procés bioindustrial per a l'empresa Photanol, l'objectiu de la qual es transformar CO_2 i llum solar en àcids orgànics a partir de soques modificades genèticament del cianobacteri *Synechocystis PCC 6803.* Sense entrar més en detall, la nostra finalitat és -mitjançant l'ús de classificadors- identificar aquelles combinacions de pH, temperatura i osmolaritat (π) que resulten en una taxa de consum d'àcid làctic -dels contaminants biològics que habiten els nostres reactors- segura (β < 0.1 mM/dia).

Abstract

Because of the climate crisis, the scientific community has been developing and exploring carbon neutral alternatives to fossil fuels. Although it is not as well-known as wind or solar energy, the use of microbial cell factories to produce biofuels and other bioproducts is gaining prominence lately. Nonetheless, on the long term, those processes require an important optimization to make them feasible and economically profitable. Historically, this optimization has been carried out by using suboptimal methods (e.g. COST: change one single variable at a time). Nevertheless, by applying the proper experimental design and by using the adequate computational tools, we can obtain solid results without investing too much time or effort. In this project, we aim to use design of experiments (DoE) and some Machine Learning algorithms (Random Forests, Support Vector machines, Artificial Neural Networks) to optimize a bioindustrial process for the company Photanol, whose goal is to turn CO_2 and sunlight into organic acids by cultivating genetically engineered strains of the photosynthetic cyanobacteria *Synechocystis PCC 6803.* Without going yet into further detail, our goal is to -through the use of classifiers- identify those combinations of pH, temperature and osmolarity (π) that result in a safe lactate consumption rate of the biological pollutants that inhabit our photobiorreactors (β < 0.1 mM/day).

Index

Figure list

Table list

1. Introduction

1.1. Background and justification of the MTP

1.1.1. General description

Recently, the use of fossil fuels has been recognized as unsustainable because of their non-renewable nature and influence on the accumulation of CO_2 and other harmful gases in the atmosphere. Since the oil crisis in the 1970s and due to the increasing demand for food, energy, and valuable chemicals, research and development on renewable, novel, and sustainable processes has been carried out. Nevertheless, to outcompete and replace fossil fuels, these processes must be reliable and economically profitable to build and operate[1]. The use of biological matter as a feedstock to generate fuels and valuable chemicals is a promising alternative that could make the economy more sustainable, being this sector declared to have potential for future growth, reindustrialization, and approaching societal challenges[2].

Currently, most bio-based industrial processes revolve around the conversion, via microorganisms, of carbohydrate compounds into diverse array of valuable chemicals, such as amino acids, vitamins, and organic acids. Unfortunately, for these production platforms, the cost of the carbohydrate feedstock is a significant fraction of the total expenses. In response to this problem, the use of photosynthetic organisms has been shown as an alternative approach, in which these costs could be eliminated. In addition, with the concern about global warming, the interest in processes that couple CO_2 capture to chemical synthesis is increasing[3]. Currently, two major technologies are employed with photosynthetic organisms. First, plant-based biofuel production via fermentation of its sugar content to ethanol and, secondly, algae derived biodiesel production through lipid extraction of biomass from large-scale cultures. Both technologies consist essentially of two phases. First, solar energy and CO_2 are converted into highly complex molecules and structures. In the second phase, a small fraction of these cellular components is converted to small molecules (as ethanol) in the case of plants, or are extracted and converted chemically to fatty acid-methyl esters in the case of algae. Hence, much of the light energy is lost in nonfermentable waste. In addition, both processes are limited by the capacity of the phototrophs to intracellularly store the substrates to be fermented or extracted[4]. Thus, although the use of photosynthetic organisms seems to yield some advantages compared to heterotrophic fermentative microorganisms, further research needs to be conducted to increase the overall efficiencies.

The use of engineered photosynthetic microorganisms as catalytic units, continuously producing bio-products extracellularly would minimize the number of steps from starting material to product and would not suffer from the disadvantages previously presented. Looking over the metabolic pathways which occur in photosynthetic organisms and in chemoheterotrophs (Fig. 1), we can observe that glyceraldehyde-3-phosphate (GAP) is in a central position for both.

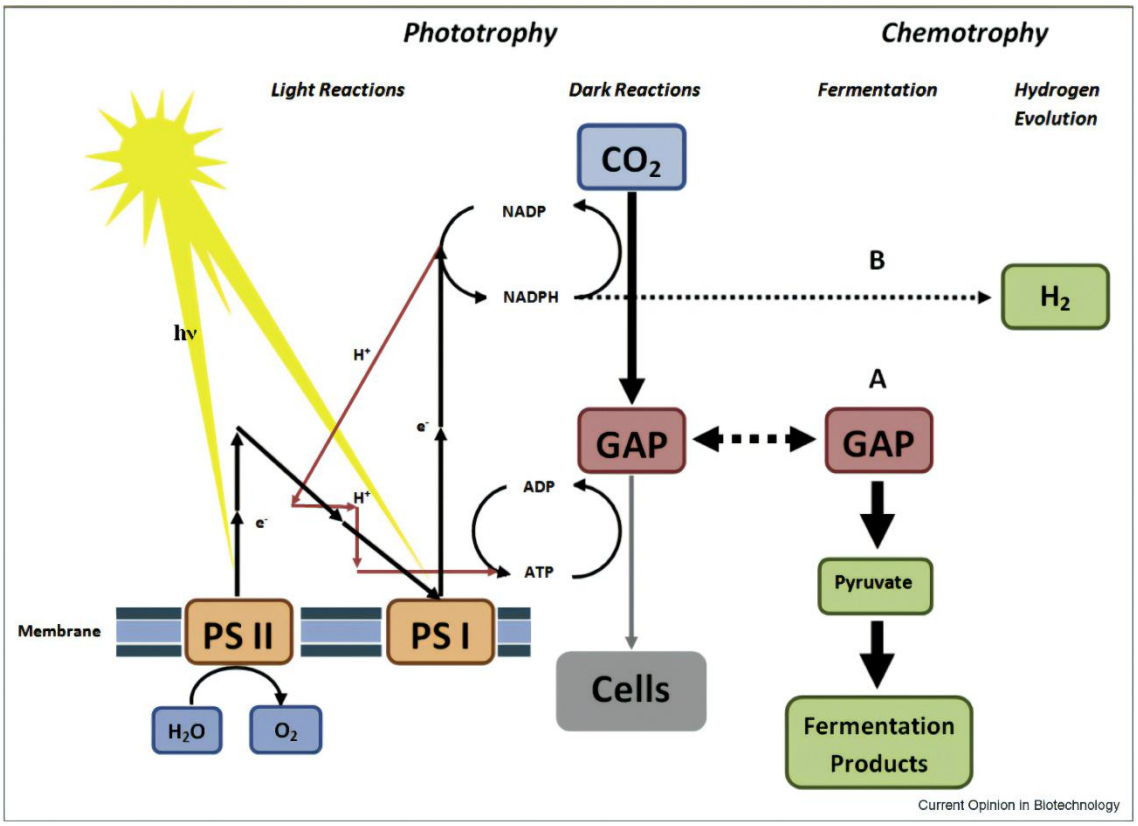


Figure 1: Schematic representation of the photofermentation concept. This idea is based on the introduction of a fermentation metabolic pathway from a chemotroph into a photosynthetic microorganism. Merging of both pathways occur through the central metabolite GAP (Angermayr et al., 2009).

For the former ones, GAP is a key metabolite in the Calvin cycle, whereas, for the latter ones, it plays the role as intermediate in numerous anaerobic catabolic pathways. The essence of the Photanol concept is then, using the methods of synthetic biology[5], to design and construct a metabolic network which combines phototrophic and fermentative metabolism into a new network that uses H_2O, CO_2, and solar energy as input and has a fermentation product as output. This type of metabolism, which we refer to as Photofermentation, involves a minimal number of steps in the conversion of CO_2 to biofuel, by bypassing the formation of the complex set of molecules of biomass. Therefore, the theoretical efficiency of biofuel production can be significantly increased as compared to first-generation and second-generation biofuel production processes. This approach is not limited to ethanol, but rather allows for a range of products as broad as

those from glycolysis-based fermentations found in nature. Thus, a collection of strains can be envisaged, obtained by introducing the proper fermentation cassette through the application of molecular genetic engineering. Diverting the carbon flow from GAP to some product, away from the natural flow toward cell synthesis, will usually lower the organisms' fitness. Therefore, the physiological effects of genetically introducing a metabolic pathway into a microbe are extremely difficult to predict and it should be realized that it is just the first step. Controllable and stable expression demands thorough understanding of the organism's genetic, regulatory, and metabolic makeup. Obviously then, the choice of the organism to use as a production platform in a Photanol approach should be guided by our level of knowledge of the organism[6].

Despite the potential of photosynthetic microorganisms (hereafter microalgae) for bio-based industrial processes, large scale culture technologies usually fail to produce high titers of biomass and bioproducts. One of the main causes, is the presence of biological pollutants, mainly heterotrophic bacteria and fungi, which use some nutrients present in the medium and the extracellular polymeric substances (EPS) and soluble microbial products (SMP) produced by cyanobacteria[7,8]. Therefore, these contaminants compete for nutrients and consume the bio-products, reducing the efficiency of the overall processes. Moreover, the presence of biological pollutants might cause sudden and massive death of microalgal cells. To overcome the challenges that these biological pollutants arise, substantial research has been conducted to find feasible strategies to control them. Traditionally, some attempts have been directed towards filtration[9] or the addition of chemicals to kill or inhibit the pollutants. Among them, the most commonly used options are pesticides, quinine sulphate[10], formaldehyde, ammonia or hydrogen peroxide[11,12]. Even though, the use of chemicals is a viable solution, these substances tend to also compromise the growth and productivity of the production strains. Therefore, as an alternative, it is quite common to adjust the environmental conditions to an optimum range at which the native microalgae have a favorable growth status while the biological pollutants have not. Adjusting the pH to acidic or alkaline values is the most common approach but there are other alternatives as using thermophilic and/or halophilic strains that can handle high temperature and salinities[13,14].

1.1.2. Justification of the MTP

One of the main issues that we must face when trying to optimize an industrial process, is the lack of time and resources[15]. As we will mention in the coming sections, neither the COST (change one factor at a time) nor the FFD (full factorial design) designs are feasible for our case. The former would lead to suboptimal results whereas the latter would involve an unrealizable number of experiments. Design of experiments (DoE) offers a promising solution to estimate proper

operating conditions. This approach has been successfully employed to develop and optimize processes in the pharmaceutical, chemical, and food industries[16]. Nevertheless, in a classical DoE approach, Linear Regression-like algorithms are used. These techniques have some limitations, including the shapes that they can use to model data, poor extrapolation properties, and sensitivity to outliers[17]. Modeling with some Machine Learning algorithms overcomes many of these shortcomings by implementing many smaller models to interpret sections of data. Therefore, in this project we intend to use few of these algorithms jointly with Oversampling and Cross Validation techniques to model our data and, in consequence, to optimize our bioindustrial process. In fact, this project is crucial for the subsistence and success of our company serving as a precedent for other optimization projects that can be carried out in the future. In general, R[18] will be our preferred coding language. Mainly, this decision was based on the fact that this language is open source, has excellent tools for data visualization and we are extensively used to its syntax. Besides, the R language has become very popular because it contains powerful libraries that easily combine different machine learning techniques. It also provides a simple way to share code between researchers.

1.2. Objectives

1.2.1. Main objectives

- Definition of those combinations of pH, temperature and osmolarity which lead to a safe lactate consumption rate ($\beta \leq 0.1$ mM/day).
- Construction of some classifiers based on Machine Learning algorithms that are capable of discerning between safe and not safe experimental conditions.

1.2.2. Specific objectives

Literature review

- Characterization of other studies that rely on DoE.
- Preselection of few Machine Learning classifiers that fit our data type.

Experimental design & data gathering

- Contextualization of the problem.

- Definition of an experimental design that implies a reasonable number of experiments that can be carried out in a realistic time frame.
- Guarantee the repeatability of the experiments.

Exploratory Data Analysis (EDA)

- Descriptive analysis of the data.

Modelling

- Application of oversampling techniques to overcome class imbalance.
- Construction of an array of classifiers based on the following Machine learning algorithms:
    - Naïve Bayes.
    - Random forests.
    - Artificial Neural Networks.
    - Support Vector Machines.
- Assessing the performance and hyperparameter optimization over the classifiers.
- Selection of the best algorithms.
- Making predictions and validating the classifiers.
- Identification of patterns and visualization of the predictions.
- Updating the classifiers.

1.3. Environmental, ethical, and social impact

During the redaction of this thesis we decided to make use of plural first-person pronouns. The rationale behind this decision was, first, to avoid using biased or sexist language and, secondly, to acknowledge other researchers and students that participated directly or indirectly in this project. Therefore, we assured that there are no negative connotations towards any gender in this report. Regarding the environmental impact of this MTP, the main milestone of this project was, after all, to optimize our bioindustrial processes and, as a consequence, make our platform an economically feasible alternative to fossil fuels. Although Photanol is still a significantly young company, we work hard to restore the balance in our planet and inspire other companies and investors to join the change.

Besides, it is also important to mention that during the execution of the cultivation experiments and all the work related to the data gathering we tried to minimize -as much as possible- the use of disposable material and to diminish the amount of generated waste. Finally, in terms of the ethical impact of this project, my only concern is that Photanol protects a significant part of its knowledge through the use of patents. Even though I am aware of how research works in the private sector, and I can understand that companies of the size of Photanol need to prevent others from copying or selling their concept, personally, I believe in a fully open and unrestricted scientific model. These thoughts are even stronger when talking about research that has to do with environmental issues and climate change.

1.4. Approach and methodology

This MTP has been divided into two main sections. The first part revolves around the experimental design and the gathering of the data whereas the second block relies on the modelling of this data. We will start by conducting an extensive literature review that will provide us with an idea of the methodology that other researchers used to tackle similar cases and that will guide us through the process of defining a solid experimental design. Amongst others, this will involve defining the range of values that we want to evaluate for each of the predictors, the number of experiments that will define each of the experimental subspaces, the duration of the experiments and the execution of some preliminary tests. Besides, this plan will be discussed with a team of professionals that will ensure its robustness. Once the experimental design is ready, we will start carrying out the cultivation experiments where we will monitor the variation of our predictors and our response variable. As opposed to most MTPs carried out in this program, the data gathering will constitute a big part of our project. In any case, once all the data has been collected, we will advance to the phase that is most relevant for this MTP: the data analysis. In general, R[18] is going to be our preferred coding language, but it might be that we use some Python[19] libraries for the graphical visualization of the results. After conducting a descriptive analysis of this data (EDA), a set of classifiers based on some of the most popular Machine Learning algorithms will be built. Then, the hyperparameters for each of these algorithms will be optimized and the performance of the classifiers will be compared. The best classifier will be selected and used to make predictions over unseen data. Afterwards, the outcome of these predictions will be analyzed and plotted to identify some underlying patterns. Some predictions will be validated by carrying out some extra cultivation experiments in the lab. Finally, the classifier that was used to make the predictions will be updated and a new and improved version will be released.

1.5. Planning

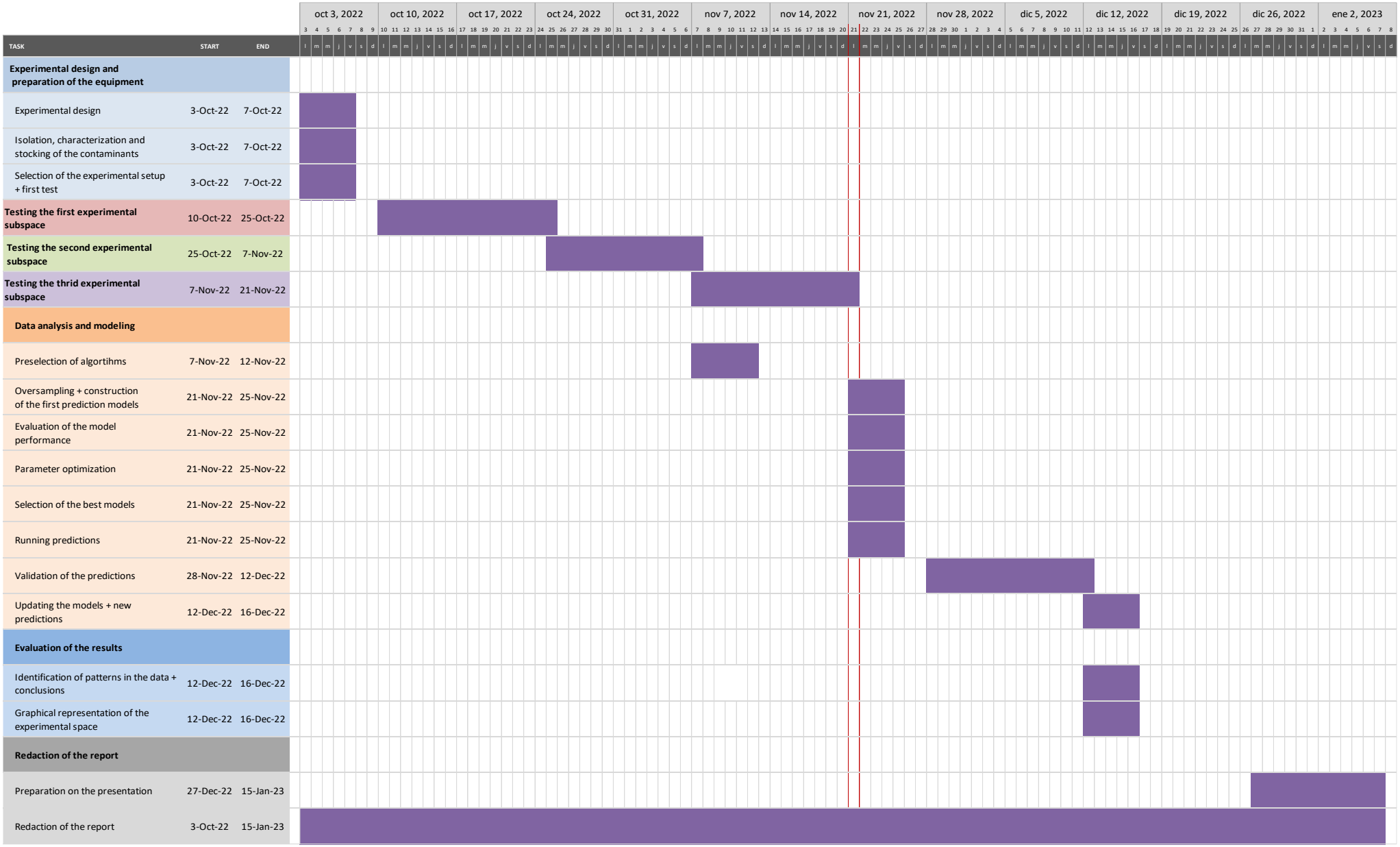
1.5.1. Main tasks and prioritization

Hereafter, a list containing the main tasks and their respective time frames is attached.

- Experimental design **(3-10-22, 7-10-22)**.
- Isolation, characterization, and stocking of the **(3-10-22, 7-10-22)**.
- Selection of the best experimental setup and preliminary tests **(3-10-22, 7-10-22)**.
- Evaluation of the first experimental subspace (**10-10-22, 25-10-22**).
- Evaluation of the second experimental subspace **(25-10-22, 7-11-22)**.
- Evaluation of the second experimental subspace **(7-11-22, 21-11-22)**.
- Algorithm preselection (**7-11-22, 12-11-22)**.
- Oversampling and construction of the first prediction models **(21-11-22, 25-11-22)**.
- Evaluation of the model performance and parameter optimization **(21-11-22, 25-11-22)**.
- Selection of the best model and running predictions (**21-11-22, 25-11-22)**.
- Validation of the predictions **(28-11-22, 12-12-22)**.
- Updating the models and new predictions **(12-12-22, 16-12-22)**.
- Identification of patterns in the data and conclusions (**12-12-22, 16-12-22**).
- Redaction of the project report **(3-10-22, 15-01-23)**.
- Preparation of the presentation **(27-12-22, 15-01-23)**.

1.5.2. Calendar

Regarding the plan of the Project, we thought it would be very beneficial to inform about the phases of this project through a Gannt diagram.


oct 3, 2022
oct 10, 2022
oct 17, 2022
oct 24, 2022
oct 31, 2022
nov 7, 2022
nov 14, 2022
nov 21, 2022
nov 28, 2022
dic 5, 2022
dic 12, 2022
dic 19, 2022
dic 26, 2022
ene 2, 2023
TASK
START
END
Experimental design and preparation of the equipment
Experimental design 3-Oct-22 7-Oct-22
Isolation, characterization and stocking of the contaminants 3-Oct-22 7-Oct-22
Selection of the experimental setup + first test 3-Oct-22 7-Oct-22
Testing the first experimental subspace 10-Oct-22 25-Oct-22
Testing the second experimental subspace 25-Oct-22 7-Nov-22
Testing the thrid experimental subspace 7-Nov-22 21-Nov-22
Data analysis and modeling
Preselection of algortihms 7-Nov-22 12-Nov-22
Oversampling + construction of the first prediction models 21-Nov-22 25-Nov-22
Evaluation of the model performance 21-Nov-22 25-Nov-22
Parameter optimization 21-Nov-22 25-Nov-22
Selection of the best models 21-Nov-22 25-Nov-22
Running predictions 21-Nov-22 25-Nov-22
Validation of the predictions 28-Nov-22 12-Dec-22
Updating the models + new predictions 12-Dec-22 16-Dec-22
Evaluation of the results
Identification of patterns in the data + conclusions 12-Dec-22 16-Dec-22
Graphical representation of the experimental space 12-Dec-22 16-Dec-22
Redaction of the report
Preparation on the presentation 27-Dec-22 15-Jan-23
Redaction of the report 3-Oct-22 15-Jan-23

1.5.3. Milestones

To ensure the fulfillment of all the project stages and, therefore, submit the final report before the deadline (15/01/2023), we decided to stablish several milestones with their respective deadlines. This way, we will be able to strictly monitor the project and we will avoid facing organization issues.

- **Milestone 1:** finish the experimental design **(7/10/2022).**
- **Milestone 2:** characterization and preparation of the contaminant stocks **(7/10/2022).**
- **Milestone 3:** submit the work plan **(17/10/2022).**
- **Milestone 4:** bibliographic research**,** algorithm preselection **(12/11/2022).**
- **Milestone 5:** finish the data gathering for all the experimental subspaces **(21/11/2022).**
- **Milestone 6:** descriptive analysis of the data **(28/11/2022).**
- **Milestone 7:** submit PEC2 **(21/11/2022).**
- **Milestone 8:** data modeling and evaluation of the performance of the preliminary models (**25/11/2022**).
- **Milestone 9:** model optimization and selection of the best model **(25/11/2022).**
- **Milestone 10:** making predictions with the models **(25/11/2022).**
- **Milestone 11:** model validation **(12/12/2022).**
- **Milestone 12:** updating the model and making new predictions**.**
- **Milestone 13:** identification of patterns in the predictions **(16/12/2022).**
- **Milestone 14:** graphical representation of the experimental space **(16/12/2022).**
- **Milestone 15:** submit PEC3 **(24/12/2022).**
- **Milestone 16:** submit final report and the presentation **(15/01/2022).**
- **Milestone 17:** thesis public defense **(03/02/2023).**

1.6. Risk analysis

1.6.1. Data gathering

- **Limited amount of time for the data gathering:** considering that each of the experiments will take approximately 2 weeks and that the amount of photobioreactors that we have available is limited (≈ 8-9), the number of experimental subspaces that we will be able to test is limited. In consequence, we think that 3 subspaces are a realistic number.

- **Reproducibility:** during the project, 2 types of photobioreactors will be used (PSI, Applikon). Although the experimental conditions will be the same, some important features as the morphology, the volume or the agitation and aeration systems are slightly different. Therefore, it is mandatory that we prove that both setups generate comparable outcomes by carrying out some validation experiments.

1.6.2. Data analysis

- **Reliability of the predictions:** previously, in this section, we anticipated that we will probably manage to test 3 experimental subspaces (27 experiments). This implies that our models will be built with a limited amount of data. Although there are some algorithms that perform quite well with little data or we can use some resampling (Bootstrapping) or some validation techniques (k-Fold Cross validation) to improve the prediction performance, we must understand that the reliability of our models will be a bit limited and, therefore, we will have to be very critical with the interpretation of the predictions.

- **Type of data:** we can already anticipate that for several experiments, we will obtain null lactate consumption rates ($\beta = 0$ mM/day). This will signify that we will have to deal with a zero-inflated distribution and, in consequence, class imbalance. It is crucial to realize that this phenomenon will directly affect the type of algorithm that we will use and will imply using some oversampling techniques to balance the data.

1.7. Expected results

Through the application of the procedure explained in the coming sections, we expect to obtain a classifier that can predict -with high accuracy- whether a specific combination of pH, T, π leads to a safe β ($\beta \leq 0.1$ mM/day) or not ($\beta > 0.1$ mM/day).

Considering the knowledge that Photanol has gathered over the years, we can anticipate that, regardless of the temperature and π, all those experimental conditions with pH ≥ 11.5 will result in a null or very low β and, therefore, will be safe. On the other hand, if we focus our attention on the temperature, based on information found in the literature, we can predict that those experiments with ≥ 50˚C will also be safe. In the case of π we ignore what to expect. Honestly, we foresee that those conditions with π ≥ 1 M NaCl will be safe, but we cannot affirm it. Additionally, we suspect about the existence of a correlation between these factors, but we ignore how strong is the interaction. In fact, we know that if we increase the value of one of these factors, we can lower the other ones without making our process unsafe. In consequence, it is crucial to understand how these factors behave and which impact they have on the definition of our experimental space. As soon as we obtain this information, we can overlap the safe regions of our experimental space (safe consumption envelope) with the space defined by the boundaries of our bioindustrial process (industrial envelope) and the limits of the candidate strains (strain envelope). As a result, we will obtain a fourth space that will contain those conditions that are safe and feasible for our bioindustrial process (operational envelope) (Fig. 2).

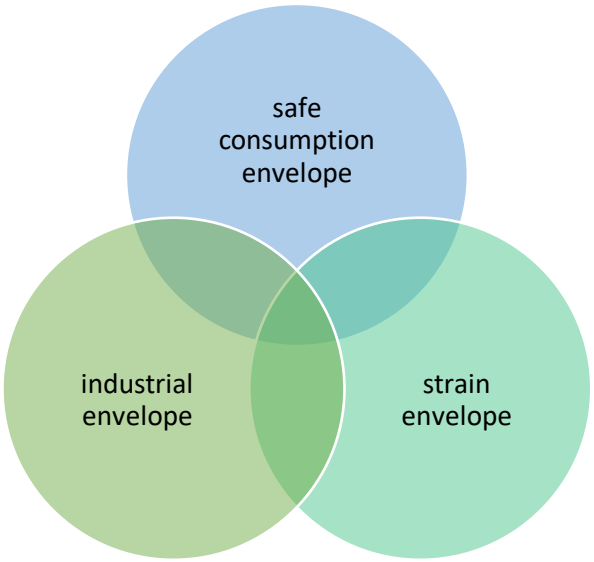


Figure 2: Venn diagram showing the convergence (known as operational envelope) between the safe consumption envelope (blue), industrial envelope (green), and strain envelope (turquoise).

1.8. Structure of the MTP

1.8.1. CAA1

Definition and work plan.

1.8.2. CAA2

Work development (phase 1). Since before the end of this phase we will not be able to finish the data gathering, we will keep focus our efforts on the first two specific objectives. Besides, we will start redacting the Materials & Methods and State of the art sections.

1.8.3. CAA3

Work development (phase 2). This will include the third and fourth specific objectives and is going to be the busiest part of the MTP. On the one hand, it will include the EDA, the modelling of the data, the optimization of the classifiers and the execution of the predictions. Moreover, we will focus the rest of our attention on finishing the redaction and submitting the first version of our memory.

1.8.4. CAA4

Final report and presentation. During this phase, we will align with our supervisor in order to

1.8.5. CAA5

Public defense in front of the board of examiners.

2. State of the art

2.1. The COST approach

Since the foundation of Photanol -but especially since last year- we have been diverting a large part of our efforts towards the development of contamination control strategies for our large-scale *Synechocystis PCC6803* cultures. Apart from growing our strains at an alkaline pH (≈ pH 11) we also explored other options such as increasing the osmotic strength (π) or the temperature. Nevertheless, to stablish the effectiveness of those factors we tried always to find their optimums independently and sequentially, overlooking the effects of their interactions. This approach is usually referred as COST (change one separate factor at a time) or OFAT (one factor at a time) and it can lead to suboptimal results[20,21]. For example, let's suppose that we want to optimize an enzymatic process (maximize the yield) by modifying two factors: volume and pH of the bioreactor. To do so, we start by defining a range of values (hereafter levels) for the factor 1 (volume) and we run a set of identical tests where we only change this factor. Once the optimal is defined (V = 550 mL), we fix this value, and we find the optimum for the second factor (pH = 4.5). As mentioned, the main issue of using this approach is that, since it does not consider the possible interaction between factors, it can lead to inaccurate conclusions (Fig. 3). Therefore, to overcome these drawbacks, we propose to apply a factorial design.

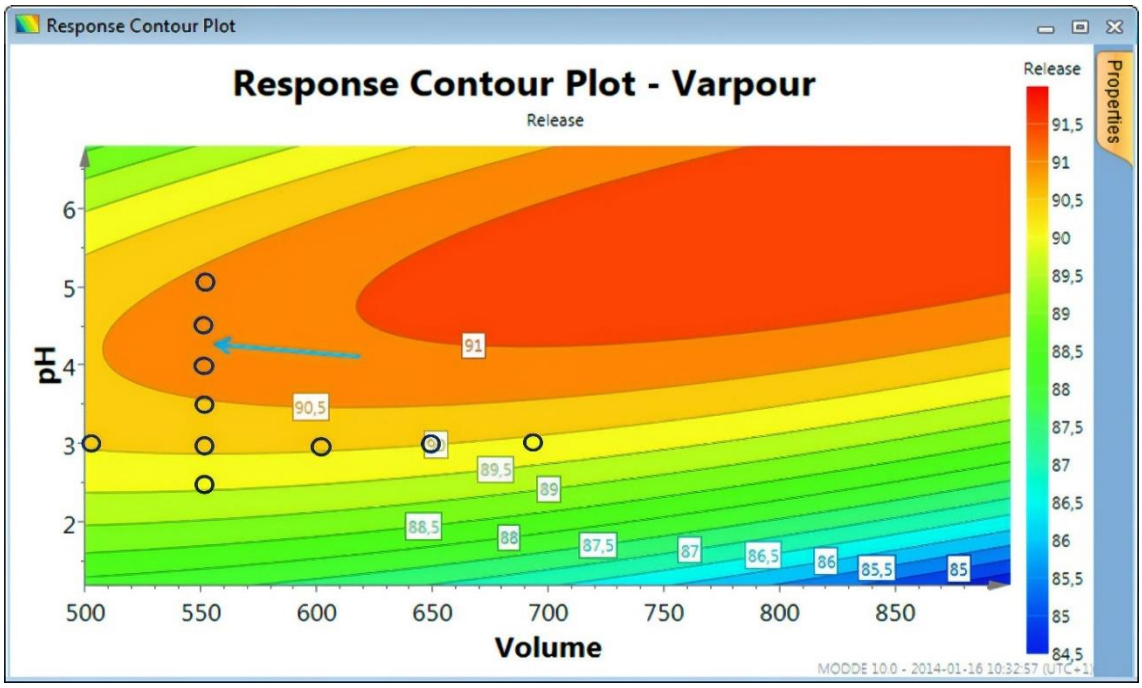


Figure 3: Definition of the optimal experimental conditions by COST (empty circles) as opposed to the real values (*contour plot*).

In a full factorial design (FFD), we would test the experimental conditions that result from all the possible combinations of the levels of our factors. This might be an option when we are working with few factors and/or factors that have few levels but, since the number of experiments increases exponentially with the number of factors/levels (m^n; m = levels, n = factors), it soon becomes overwhelming and unfeasible (Table 1).

Factors (n)	Levels (m)	Experiments
2	2	4
2	3	9
3	2	8
3	3	27
4	2	16
4	3	81
5	2	32
5	3	243
6	2	64
6	3	729
7	2	128
7	3	2187
8	2	256
8	3	6561

Table 1: Number of necessary experiments in a full factorial design in respect of the number of factors (*n*) and the number of levels per factor (*m*).

2.2. Design of experiments as an alternative strategy

Given that, in our project, we are planning to test the effect of 3 factors (pH, temperature and π) over β, and we intend to evaluate more than 5 levels per factor, a full factorial design is not an option (> 125 experiments). For those situations we can use a fractional factorial design or FrFD (sometimes also referred as design of experiments or DoE)[22–24]. The goal of this strategy is to extract the maximum amount of information regarding the factors from as few observations as possible[16]. Although there are many different designs like the Central Composite Design (CCD)[20] (Fig. 4B) or the Box-Behnken design (BBD)[25] (Fig. 4C), we decided to use a much simpler design. This consists in defining 2 levels per factor (-1 and +1), joint crossing them and testing all the possible combinations (Fig. 4A). Therefore, we will obtain a cubical space that will be delimited by 8 vertices or experimental conditions (hereafter experimental subspace). In order to increase the amount of information provided by our experimental subspace, we agreed to include an extra experimental condition in the centre of the cube that will act as a compass. Depending on the complexity of our optimization process and our experimental design of choice, it might be that one experimental subspace is enough to model the results and identify patterns.

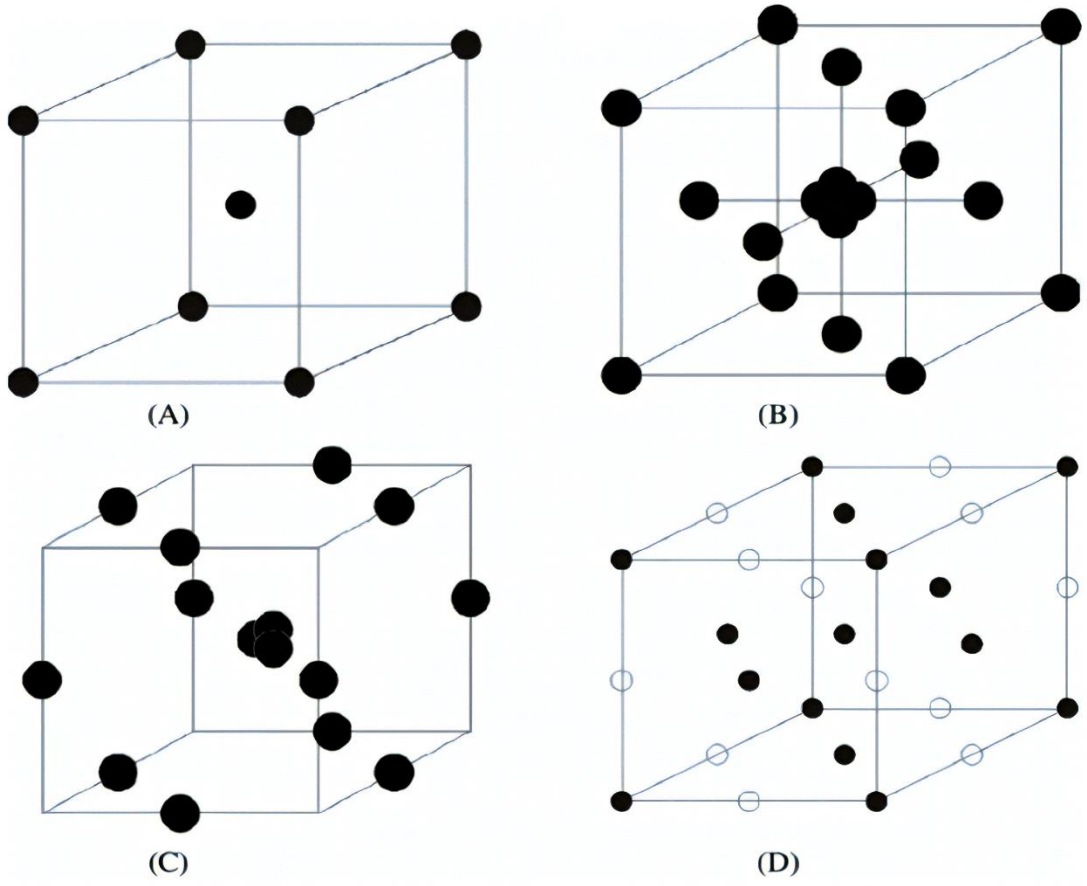


Figure 4: Schematic representation of experimental designs for three factors: (A) our design (m = 3, n = 2); (B) central composite design (CCD); (C) Box-Behnken design (BBD); and (D) three factor and level full factorial design.

Nevertheless, in our case, it is very unlikely that 9 experiments will suffice. In consequence, we will use a slightly different approach. Our idea is to use the optimal outcome of the first experimental subspace to define a second and -afterwards- a third subspace (Fig. 5). In other words, the experimental design shown in Fig. 4A, will be applied -most likely- three times, being every new experimental subspace dependent on the previous ones. It is very important to clarify that, when defining the optimal vertex of an experimental subspace, this does not have to be the one with a lowest β. Other factors as the industrial applicability or the physiological boundaries of our cyanobacterial strains are crucial and, therefore, will play an important role in our decision. For example, at pH ≥ 12 and/or temperatures ≥ 60˚C, β will be probably 0 mM/day but no cyanobacterial strain can handle those conditions. Moreover, the energy necessary to maintain a bioreactor running at 60˚C is unfeasible. After stablishing few experimental subspaces, we could assume that the β for the combination of pH, T, π that we obtained is optimal or -at least- way more optimal than the combination that we would have obtained by using the COST approach.

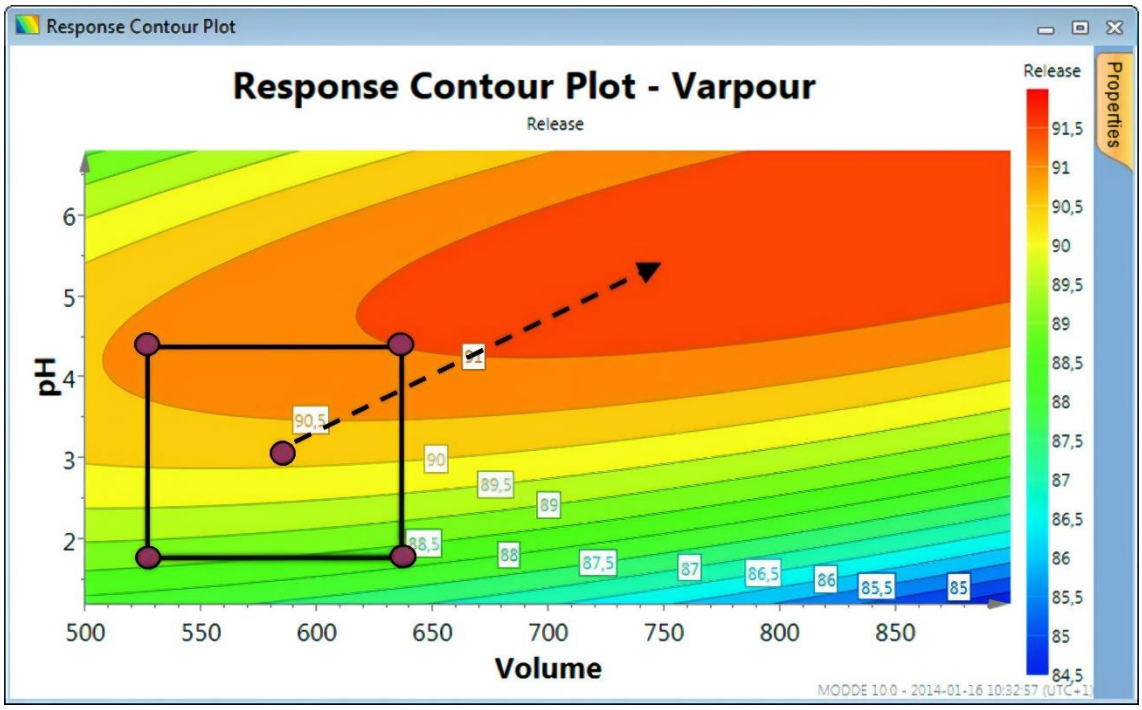


Figure 5: Definition of the optimal experimental conditions by applying the DoE approach.

Reached this point, we can put all the collected data together and model it. Basically, the idea behind the modelling is to generate an array of prediction models that will allow us to explain how β varies in function to our predictors (pH, T, π). In other words, we want to obtain some algorithms that -given some values for our predictors- predict whether that experimental condition is safe or not safe. In the section Approach and Methodology, we will dive a bit more into the methods that we will use for the construction of the prediction models. We will get back to that in the next section.

2.3. Classical DoE vs our approach

Classically, in DoE, linear regression and ANOVA models are combined to model the relationship between a quantitative variable (response) and the predictors and to study the interaction within predictors[26–28]. Nevertheless, in some situations, linear models (LM) might not be the best solution. Previously, Rodriguez et al., successfully applied artificial neural networks as an alternative to LM[29]. Getting back to our case, since we ignore if the distribution of our data is linear, it might be more adequate to use non-linear algorithms. Moreover, since we expect a lot of our experiments to result in no lactate consumption whatsoever (β = 0 mM/day), our data will suffer from what we know as zero-inflated distribution[30].

As its name suggests, zero inflation indicates that a dataset contains an excessive number of zeros. If not properly modeled, the presence of excess zeros can invalidate the distributional assumptions of the analysis, jeopardizing the integrity of the scientific inferences[31]. The zeros can also arise several computational difficulties[32–34]. There are many options to deal with this situation, being common to use zero-inflated Poisson (ZIP) regression models. Nevertheless, since our goal is ultimately to discern whether a combination of pH, T and π is safe or not for accumulating lactate (safe: $\beta \leq 0.1$ mM/day; not safe: $\beta > 0.1$ mM/day) the use of classification algorithms instead of regression models might simplify things and improve the performance. If we choose to move towards classification and to generate a new response variable by converting β into a binary categorical variable, we will encounter a different problem: class imbalance. Class imbalance occurs when the distribution of the classes is biased or skewed, being a common issue in the Machine Learning and Data mining community.

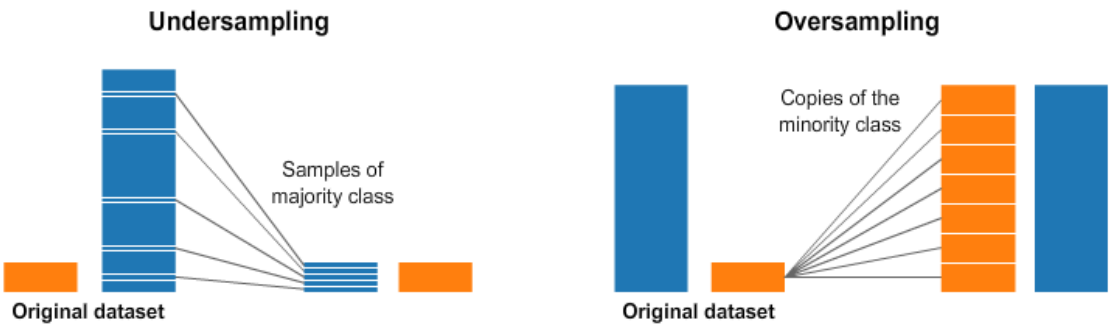


Figure 6: Schematic representation of how the undersampling and oversampling techniques work.

The class-imbalance distribution can make most classical classification algorithms neglect the significance of the minority class and tend toward the majority class[35,36]. Although some algorithms perform quite well over class-imbalanced data (e.g. Random forests, Decision Trees), there are other solutions as Random Undersampling or Oversampling. The former is based on reducing the observations from the predominant class whereas for the latter we try to generate more observations from the minority class usually by replicating the samples from the minority class (Fig. 6). Finally, the last issue that we will encounter when trying to model our data is the lack of observations. As mentioned previously, if we test 3 experimental subspaces, we will end with a dataset that will contain only 27 observations. The main consequence of this feature is underfitting and, therefore, low performance in our predictions (Fig. 7). Underfitting occurs when the prediction model cannot capture the underlying trend of the data.

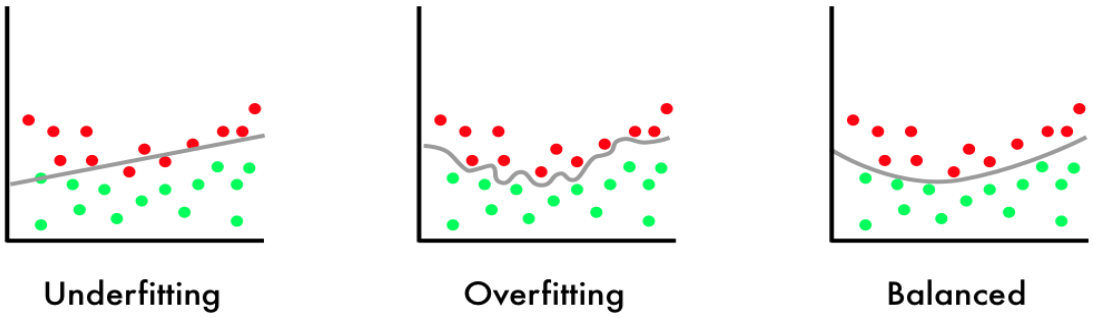


Figure 7: Schematic representation of how an underfitted, overfitted or balanced prediction model work.

In line to what we mentioned before, something we can consider implementing is data augmentation through the generation of synthetic random samples[37]. In other words, we can use the Bootstrap method[38] or other data augmentation techniques to increase the size of our data set and overcome both the lack of data and class imbalance (Fig. 8).

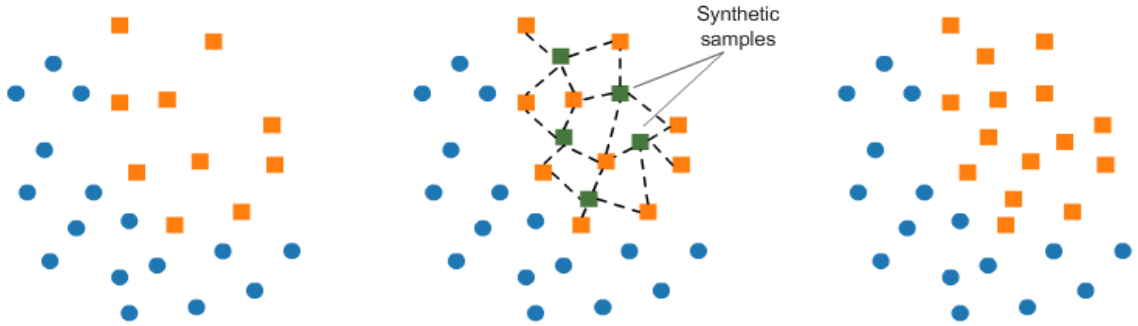


Figure 8: Schematic representation of the process of data augmentation through the creation of random synthetic samples.

In the next section we will review four Machine Learning algorithms that were selected to build our classifiers, explaining briefly how they work and reviewing their strengths and weaknesses.

3. Materials & Methods

3.1. Preparation of the inoculum

2.1.1. *Synechocystis PCC 6803*

Synechocystis PCC 6803[39] (hereafter *wt*) cells were pre-cultured in 500 mL shake flasks containing 2x BG11 medium (Table 24) + 20 mM $NaHCO_3$. The flasks were placed in one of our red-light incubators (35 °C, 1% CO_2, 150 rpm, 300 $\mu E \cdot m^{-2} \cdot s^{-1}$) and, once they reached an OD_{730} of 20, the cultures were spined down, the supernatant was discarded, and cells were concentrated 10x in the same medium. These inoculums were kept at 4 °C for a maximum of 2 days and plated for a biological contamination assessment before their use.

2.1.2. Contaminants

Raw samples

A 10 L sample was collected from a contaminated reactor located in the Pilot plant and concentrated 10 times (cell count: 10^8-10^9 cfu/mL). The cultivation conditions at sampling time were T = 30°C, pH ≈ 10.75, π = 0 M NaCl. After concentrating the sample, it was mixed with glycerol (12-15%) and 10 mL aliquots were prepared and stored at -80 °C.

Top 10

For each of the top 10 contaminants (see Table 2), cells were precultured in the dark (30 °C, 120 rpm) until they reached an OD_{600} of 1 (≈ $8 \cdot 10^8$ cfu/mL). Afterwards, we mixed -in 2x BG11 + 20 mM $NaHCO_3$ + 12% glycerol- the appropriate volumes to have the same concentration of each of the contaminants and a final cell count of 10^7 cfu/mL. The resulting mixture was plated to assess the actual cfu/mL. 0.5 mL aliquots were prepared and stored at -80 °C.

Israelian soil

In order to obtain a new “zoo” of contaminants that was better adapted to high temperatures, a soil sample from the Dead Sea was collected. Regarding the sample treatment, the soil was resuspended in 1x BG11 + 5 mM TES + 20 mM $NaHCO_3$ + wt (OD_{730} ≈ 1) + 20 mM lactate. Contaminants were grown in the dark (30 °C, 120 rpm) for approximately 24 hours and, afterwards, we plated the culture to assess the cfu/mL. Finally, 10 mL stocks were prepared (12% glycerol) and kept at -80 °C.

Top 10 lactate	
Erythrobacter neustonensis[40]	*Roseococcus suduntuyensis*[41]
Mongoliitalea lutea[42]	*Mucilaginibacter kameinonensis*[43]
Roseinatronobacter monicus[44]	*Solitalea canadensis*[45]
Paracoccus marinus[46]	*Pseudomonas stutzeri*[47]
Flavobacterium haoranii[48]	*Malikia spinosa*[49]

Table 2: Top 10 biological contaminants found in our photobioreactors containing lactate producing strains.

Common master mix

For the validations we decided that it was more representative to broaden the diversity and to increase the presence of the most abundant contaminants by combining the raw sample, the top 10 contaminants and the sample from the Dead Sea. Equivalent cell counts of each of the samples were mixed in 2x BG11 + 20 mM $NaHCO_3$ + 12% glycerol (cell count: 10^7 cfu/mL). 0.5 mL stocks were prepared and kept at -80 °C.

3.2. Data gathering

3.2.1. Normal experiments

Co-cultures of *wt* (OD_{730} ≈ 2) and contaminants (raw samples, initial cell count: 10^3-10^4 cfu/mL) were grown in either 400 mL or 1.5 L photobiorreactors (PSI or Applikon respectively) containing 1x BG11 + 20 mM TIC (different ratios $NaHCO_3$/Na_2CO_3 depending on the desired pH) + 5 mM TES + 50 mM lactate. Cells were bubbled with air (PSI: 200 mL/min, Applikon: 300 mL/min), subjected to a full dark regime and exposed to a specific combination of pH, temperature, and osmolarity. pH was kept constant by dosing NaOH through a peristaltic pump connected to either a Biocontroller or the PSI software (PSI: 1-1.5 M, Applikon: 2 M).

3.2.2. Validations

Co-cultures of *wt* (OD_{730} ≈ 2) and contaminants (common master mix, initial cell count: 10^3-10^4 cfu/mL) were grown in either 400 mL or 1.5 L PBRs containing 1x BG11 + 20 mM TIC ($NaHCO_3$/Na_2CO_3) + 5 mM TES + 50 mM lactate. Cells were bubbled with air (PSI: 200 mL/min, Applikon: 300 mL/min), subjected to a full dark regime and exposed to a specific and constant combination of pH, temperature, and osmolarity. pH was kept constant by dosing NaOH (PSI: 1-1.5 M, Applikon: 2 M).

3.3. Analytical Procedures

2 mL samples were taken from the photobiorreactors -at least once per day- and Lactate and TES concentrations were determined by High-performance liquid chromatography (HPLC). Moreover, we kept track of the inline and offline pH and temperature, and of the dO_2 and salinity. In some cases, contaminants were plated in BG11-J and/or LB plates (Table 25) to relate the OD_{600} to cell counts.

3.4. Evaluating lactate consumption

The parameters used to evaluate the contaminants' lactate consumption were the corrected lactate concentration (corrected by the evaporation factor), the OD_{600} and -most importantly- the β (average lactate consumption rate in mM/day). Additionally, before the classifiers were built, a binary variable (Class) was generated by fixing a threshold on β (safe: $\beta \leq 0.1$ mM/day, not safe: $\beta > 0.1$ mM/day).

3.5. Data modelling

3.5.1. Exploratory Data Analysis (EDA)

Before we started modelling the data, an exploratory data analysis was conducted. The goal during EDA is to visualize and understand our data. Although EDA is quite systematic, since it depends on the nature of our data and our criterium and experience as data analysts, it is considered to be a very open and creative process. For our particular case, we started using the functions `str` (`utils`), `datatable` (`DT`[50]) and `plot_intro` (`DataExplorer`[51]) to assess the structure of our dataset and whether we had some anomalies as -for example- missing values. Afterwards, making use of the library `ggplot2`[52], we studied the distribution of our response (β) through the construction of a histogram and a density plot. The next step was to evaluate the joint covariance of our variables; also known as covariation. Since the values of our predictors were fixed by us (control variables), we skipped the covariance study within predictors. Nevertheless, the covariation between β and each of the predictors was estimated. Once again, we used `ggplot2` to construct the scatterplots. Since the ultimate goal of this MTP was to end up with a classifier capable of discerning between safe and unsafe conditions, we thought it would be interesting to evaluate the joint effect of our predictors over the distribution of our binary variable Class. Therefore, we used `ggplot2` to build a second array of scatterplots where we crossed all the possible combinations within predictors and where we used a colorscale to differentiate both classes. Besides, a 4D plot (x = pH, y = temp, z = π, colorscale = class) was built with the library `plotly`[53] in order to improve the visualization.

The final part of this process was repeated after balancing the classes of our dataset. For this purpose, the function `upSample` was used (`caret`[54]). All the code related to the EDA is attached in the script EDA.R.

3.5.2. Construction & optimization of the classifiers

Once the EDA was finished, we utilized some Machine Learning R packages (`caret`, `e1071`[55], `klaR`[56], `randomForest`[57], `nnet`[58]) to model the data and to obtain some high-performance classifiers. As mentioned before, the predictors used during this process were the pH, temperature and π whereas the response variable was Class (Fig. 9). The employed algorithms were Naïve-Bayes (NB), Support Vector Machine (SVM), Random Forest (RF), and Artificial Neural Network (ANN). A detailed explanation of each of these algorithms will be provided later in this section.

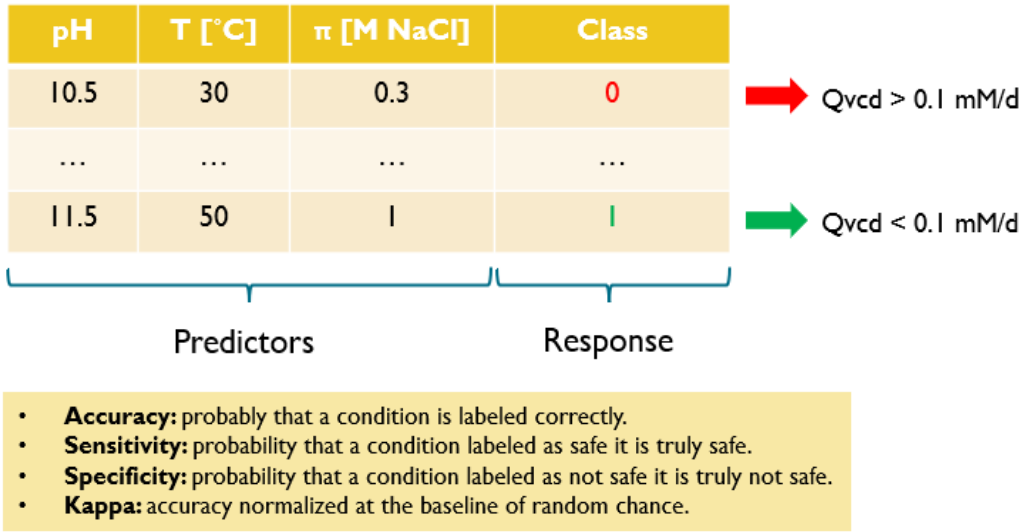


Figure 9: Predictors and response used to model our data and performance metrics used to select the best classifiers.

Regardless of the algorithm, the implemented pipeline was the same. To start, we made use of the functions `trainControl` and `train` of the package `caret` to build a set of classifiers, carry out a random search and, as a result, find the most optimal hyperparameters.

The function `trainControl` controls some computational nuances of the `train` function as the resampling method (`method`), the number of folds or resampling iterations (`number`) or the number of repeats (`repeats`). In our case, since we decided to implement a 5-fold cross validation with 10 repeats, the arguments used were `method = repeatedcv; number = 5 and repeats = 10`.

On the other hand, `train` fits the algorithm into the data using the rules defined by the `trainControl` function. Besides, it sets up a random grid for each of the hyperparameters and evaluates the performance for all the constructed models. The best classifiers were selected after the evaluation of the following performance metrics: accuracy, kappa, ROC, sensitivity, and specificity. Then, once we had a rough idea of which were the best hyperparameters, we carried out a grid search. To do so, we used the same functions but, this time, the values tested for the hyperparameters were defined manually in the `tuneGrid` argument of the function `train`. Same performance metrics were used to assess the reliability of the models. Finally, the performances of the best classifiers for each of the algorithms were contrasted and the best classifier was selected. All the code used along this process can be found in any of the scripts used for the modelling (e.g. SVM (classification.R).

Support Vector Machines (SVM)

Support Vector Machines (SVMs) are a type of supervised Machine Learning algorithms that can be employed both for classification and regression problems. They were initially proposed by Cortes and Vapnik[59] in the late 1990s and have gained popularity since then within the scientific community[60–62]. They can solve linear and non-linear problems and they proved to be successfully applied in all kinds of domains, including text classification[63], handwritten digit recognition[64], face recognition[65], bioinformatics[66], among many others. The idea of SVM for classification is simple: the algorithm creates a boundary called hyperplane that splits the data. The Maximum Margin Hyperplane (MMH) is the hyperplane that generates the greatest separation between the classes. The support vectors are the points from each class that are the closest to the MMH; each class must have at least one support vector, but it is possible to have more than one (Fig. 10). In many real-world cases, the relationships between variables are nonlinear. A key feature of SVMs is their ability to map the problem into a higher dimension space using a process known as the kernel trick. In doing so, a nonlinear relationship may suddenly appear to be quite linear. SVMs with nonlinear kernels are extremely powerful classifiers. Some of the most popular kernels are the linear kernel (no transformation), polynomial kernel, sigmoid kernel, and Gaussian RBF kernel (Fig. 11). There are no rules on how to assign a kernel to a certain learning job. This will depend mainly on th amount of training data, the relationships between the predictors and the nature of the learning task. Therefore, it is strongly recommended to apply a bit of trial and error and to evaluate several kernels. Moreover, it is important to evaluate different values for some of the hyperparameters.

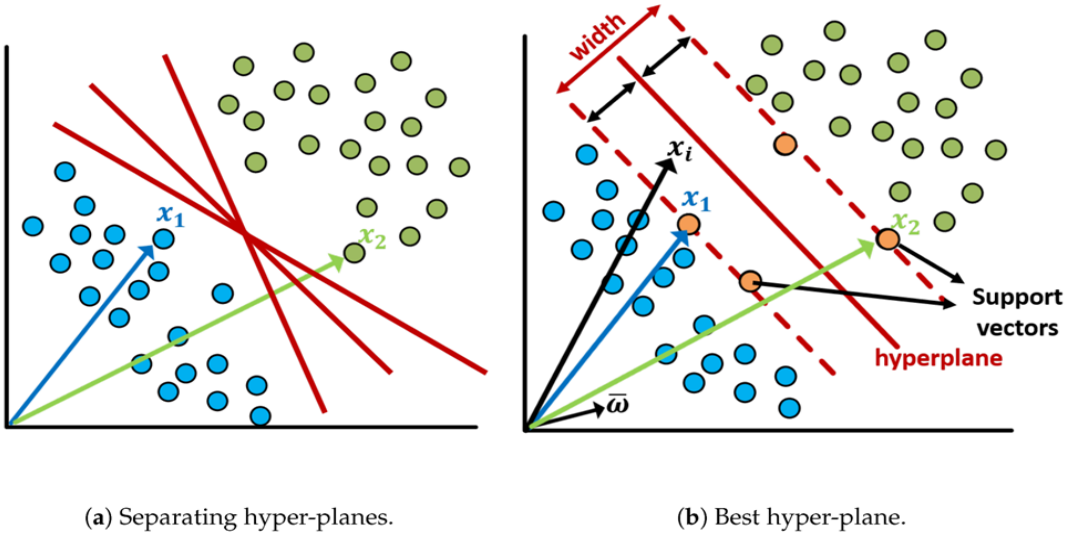


Figure 10: Schematic representation of how SVM-classifiers work.

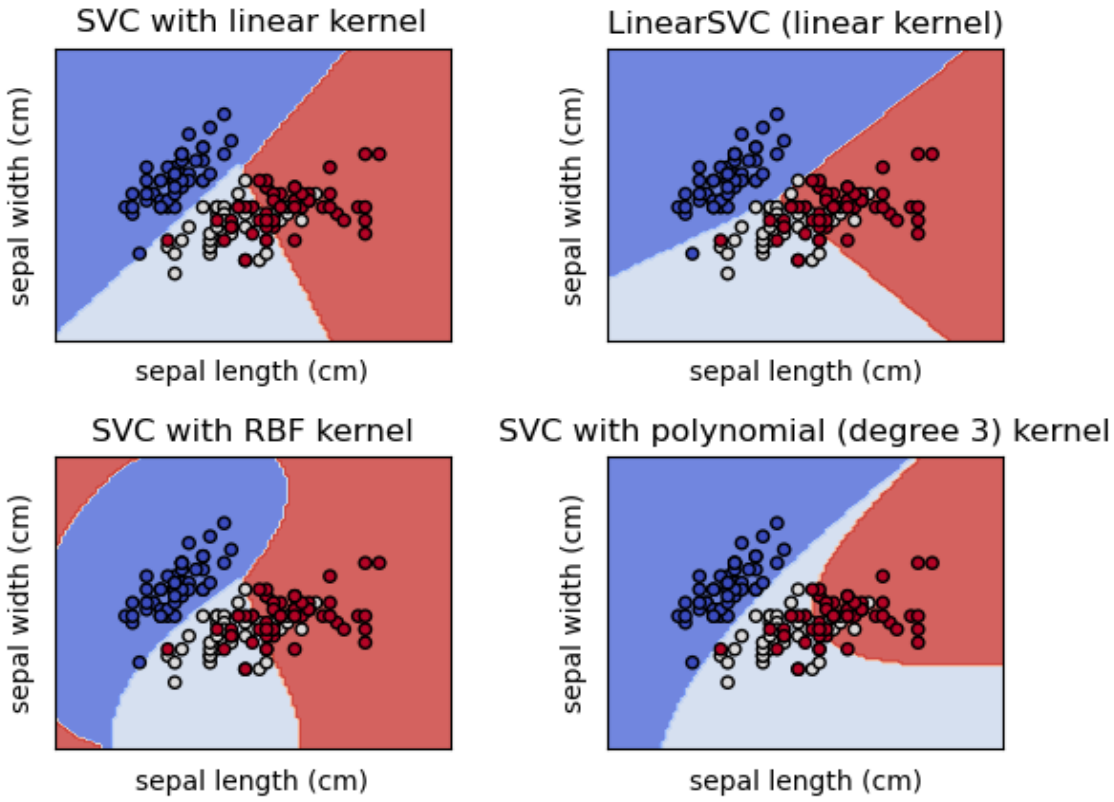


Figure 11: Graphical representation of how different SVM kernels split the data for the famous dataset iris.

Strengths:

- Can be used both for regression and classification.
- They perform well when there is a clear margin of separation.

- Effective in high dimensional spaces (high number of predictors).
- Effective when the number of dimensions is greater than the number of samples.
- Not very prone to overfitting.
- Easier to implement than other algorithms like ANNs.

Weaknesses:

- Performs poorly when the margin between classes is not well defined.
- Choosing the proper kernel is not always easy.
- Difficult to fine tune some of the hyperparameters (like C or γ).
- Could be slow to train.

Naïve Bayes (NB)

Naïve Bayes (NB) is one of the most popular, simplest, and fastest supervised Machine Learning classifiers that is based in the Bayes Theorem. It is quite popular because it can outperform highly advanced classification techniques without requiring much computational resources[67]. Naïve Bayes has been applied successfully for text classification[68,69], diagnosing medical conditions[70,71] or for the detection of anomalies in computer networks. This algorithm is named as such because it makes some "naive" assumptions about the data. Naïve Bayes assumes that all the predictors in the dataset are equally important and independent which is rarely true in most real-world applications.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Figure 12: Mathematical formula of the Bayes Theorem.

The mathematical formula of this theorem is shown in Fig. 12 where P(A) and P(B) are the probabilities of the events A and B without regarding each other. P(A|B) is the probability of A conditional on B and P(B|A) is the probability of B conditional on A. In Naïve Bayes classification, A is categorical outcome events or the response whereas B is the array of predictors.

Strengths:

- Simple, fast, and very effective.
- Suitable for multiclass datasets.

- Works well with small datasets and numerical predictors.
- You can obtain the probability of the predictions.

Weaknesses:

- Assumes that all predictors are independent from each other.
- Normally it gives worse results than the other ML algorithms presented in this MTP.

Artificial Neural Networks (ANN)

Artificial neural networks (ANNs) mimic the behavior of the human brain in the processing of input signals that are transformed into output signals[72]. Our brains are composed of billions of neurons which process and transfer information through electrochemical signals. External information (input) is collected by the dendrites, processed in the neural cell body and, eventually, converted into an output and transferred to the next neuron. Likewise, ANNs are composed of artificial neurons (also known as nodes) that receive inputs from the outside (predictors) or from other neurons that are allocated in more external layers. These inputs are processed, and an output is generated and either transmitted to the next layer of neurons or used as the final decision of the classifier.

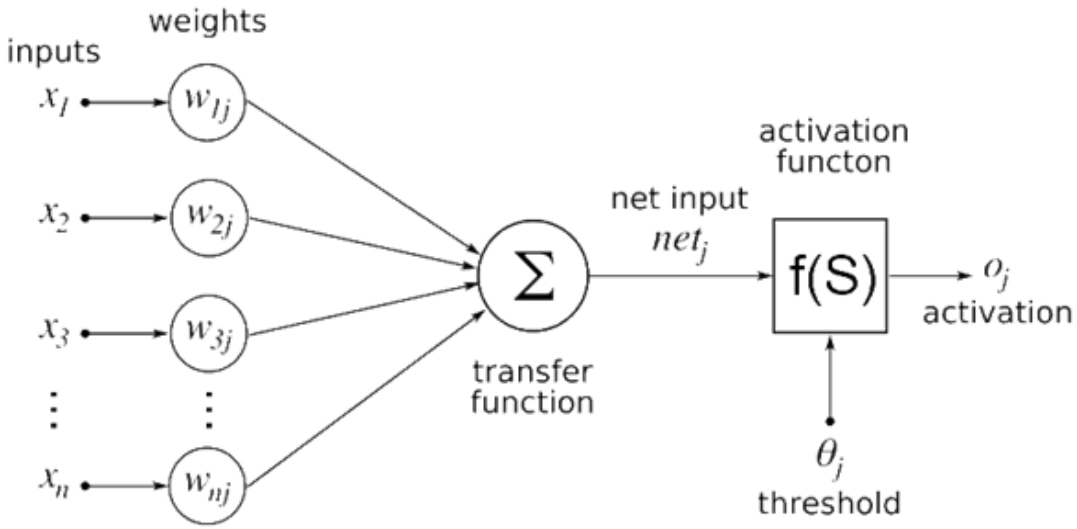


Figure 13: Schematic representation of the simplest ANN: the perceptron.

In order to understand how this algorithm functions let's have a look at Fig. 13. In this figure the simplest possible version of an ANN is presented (perceptron). In the first step, the inputs from the predictors (x_i) are transferred to the only neuron that composes the network. These inputs form the so-called input layer and are linked to weights (w_i) that make these signals more or less relevant. On the other

hand, the neuron is part of what we know as the hidden layer. Once the inputs reach the neuron, the information is processed, and an output is generated. In this case, since there are no more neurons in the hidden layer, the output would be the predicted class. The case of the multiperceptron is very similar. In fact, the only difference is that between the output and the input layer we have several nodes that are interconnected and that rely on each other (Fig. 14) .

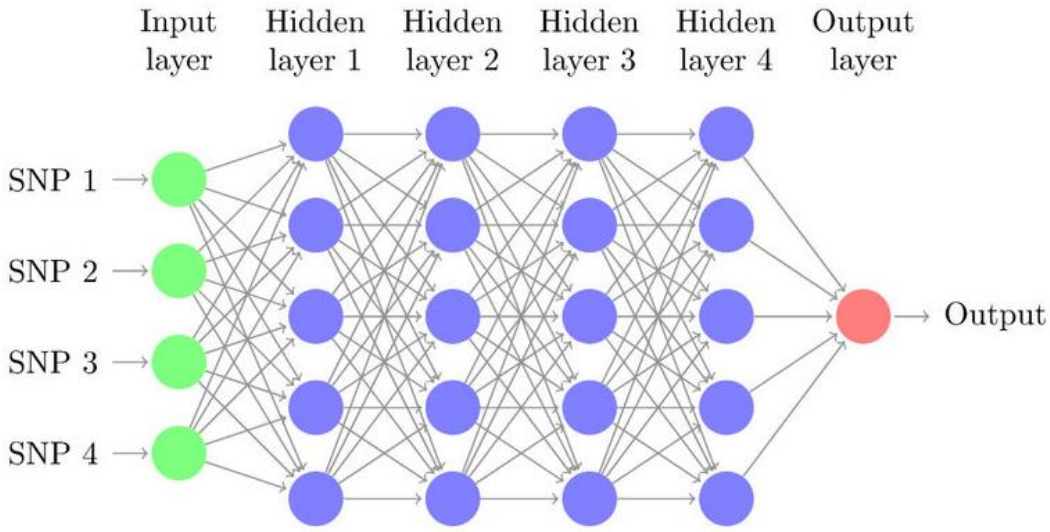


Figure 14: Graphical representation of a multiperceptron.

Although it is recommended to define the optimal number of neurons in the hidden layers by applying a bit of trial and error, the following formula might help (Fig. 15).

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

N_i = number of input neurons.
N_o = number of output neurons.
N_s = number of samples in training data set.
α = an arbitrary scaling factor usually 2-10.

Figure 15: Rule of thumb that might be used to define the optimal number of neurons in the hidden layer.

Similarly to what we did for the SVM and NB, we attach a list of the strengths and weaknesses of this algorithm:

Strengths:

- Can be used both for regression and classification.

- Can be used to model non-linear and complex relationships.
- Does not impose restriction on the predictors.
- Outstanding performance over large datasets.
- Small sensibility to outliers.

Weaknesses:

- Require huge amounts of data.
- Computational expensive.
- Optimization process can be challenging.
- Prediction models produced are almost incomprehensible.

Random Forests (RF)

Random forest (RF) is a Supervised Machine Learning Algorithm that came into the spotlight in 2001[73] and that has been widely used to solve Classification and Regression problems. It successfully faced some challenges as predicting drug response in cancer cell lines[74], identifying DNA-binding proteins[75], and localizing cancer to particular tissues from a liquid biopsy[76]. The rationale behind this algorithm is simple; it builds decision trees on different samples and takes their majority vote for classification and average in case of regression. To better understand how this algorithm works, let's have a look at how a decision tree works in a classification problem. For that purpose, we will use the example shared in Fig 16. Imagine that our data set consists of the numbers that are presented at the top of the figure. The features or predictors are the color and whether the number is underlined or not. To classify the observation, the decision tree goes through several decision steps (also known as decision nodes). For instance, the first decision node refers to the question: is it red? If that observation is not red, it will be classified as blue. In contrast, if the answer is yes, we will advance to the next decision node. In the next node the question is: is it underlined? If the answer is yes, the observation will be classified as red underlined numbers whereas, if the answer is no, it will be classified as a red not underlined number. Therefore, the idea of Random forests is very simple: the wisdom of the crowds. They consist of a large number of individual decision trees that work as a committee or ensemble. Each individual tree makes an independent prediction and, afterwards, the class with the most votes is selected.

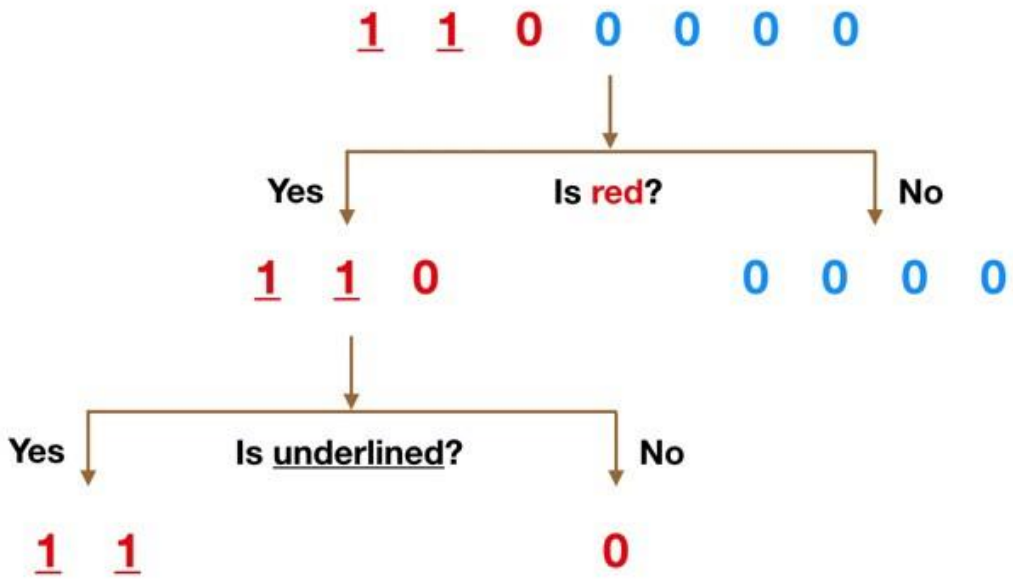


Figure 16: Schematic representation of a decision tree classifier.

Strengths:

- Can be used both for regression and classification.
- Can be used to model non-linear and complex relationships.
- Can handle both small and big datasets.
- Small sensibility to outliers.
- Small sensibility to class imbalance.

Weaknesses:

- Computational expensive.
- More prone to overfitting.

3.5.3. Making & visualizing predictions

Once the best classifiers were selected, we proceeded to make some predictions. With that purpose in mind, two parallel approaches were used. The former consisted in making predictions over few specific conditions, whereas the latter was based on making hundreds of thousands of predictions and plotting the outcome in order to visualize the consumption envelope.

Predictions over specific conditions

A tibble (`tidyverse`[77]) or data.frame object was created by defining a sequence of values for each of our predictors (pH, temp, osm). Afterwards, the function `predict` from the package `stats` was used to predict the expected outcome for each of these conditions. The arguments passed to this function were the classifier (`object`) and the dataset containing the experimental conditions (`newdata`). The code used to make these predictions is shown in any of the scripts used for the modelling.

Extensive predictions & visualization of the consumption envelope

The second main application of the classifiers was to run hundreds of thousands of predictions and to build some graphs in order to visualize the lactate consumption envelope. Apart from the functions mentioned in the previous section, we made use of `seq` (`base`), `crossing` (`tidyr`[78]), and `plot_ly` (`plotly`). The first function was used to create long sequences of values for each of the predictors whereas the second was used to cross join these predictors. Once we had the dataset with all the experimental conditions, `predict` was used to make the predictions. Finally, for the graphical representation of these predictions we used either `ggplot2` (3D plots) or `plot_ly` (4D plots).

3.5.4. Validating the predictions & updating the model

Last but not least, we had to validate and update the models. In order to do so, the Chassis strain team chose some interesting conditions that they wanted to evaluate, and we carried out the experiments. Afterwards, the models were updated. This process can be repeated as many times as desired in order to increase the models' performance.

4. Results

4.1. Exploratory Data Analysis (EDA)

The first step in any data analysis process is the exploration of the data. The dataset that was used for our study consists of 31 observations which correspond to cultivation experiments where the pH, temperature, and osmotic strength (predictors) were controlled, and where the average daily lactate consumption rate (β) was monitored (response). A short description of the variables, their corresponding data type, and a basic statistical summary are presented in Table 3A and Table 3B.

A)

Variable	Description	Data type
pH	pH fixed during the experiment	Numeric continuous
temp	Temperature fixed during the experiment [˚C]	Numeric continuous
osm	Osmolarity fixed during the experiment [M NaCl]	Numeric continuous
qvcd or β	Average daily lactate consumption rate [mM lactate/day]	Numeric continuous

B)

	pH	temp	osm	β
Min	9.00	30.00	0.30	0.00
1st Quantile	10.00	35.00	0.30	0.00
Median	10.50	45.00	0.70	0.00
Mean	10.37	44.19	0.74	4.60
3rd Quantile	10.88	50.00	1.00	1.40
Max	11.50	55.00	1.40	36.48

Table 3: Table including some general information about the variables of our dataset (A). Summary table including the minimum, 1st quantile, median, mean, 3rd quantile and max of each of the variables of our dataset (B).

Even though we were dealing with a very small dataset (31 x 4), it was interesting to carry out a preliminary exploration. As expected, our 4 variables were continuous, all rows were complete (no missing values) and there were no missing columns (Fig. 17). Without doubt, one of the most relevant features that we could investigate was how our variables are distributed. Although our predictors are quantitative variables and they were treated as such during the modelling, since only few values were tested and they were frequently repeated, it was convenient to treat them as categorical variables to study their distribution. Therefore, instead of generating an array of histograms we built three barplots that showed the most common values tested for each of the predictors.

As shown in Fig. 18, regarding the pH, 9, 10.5, and 10.75 were the most common values. As for the temperature and π the most frequent values were 45 and 55 ˚C and 0.3 and 1 M NaCl respectively.

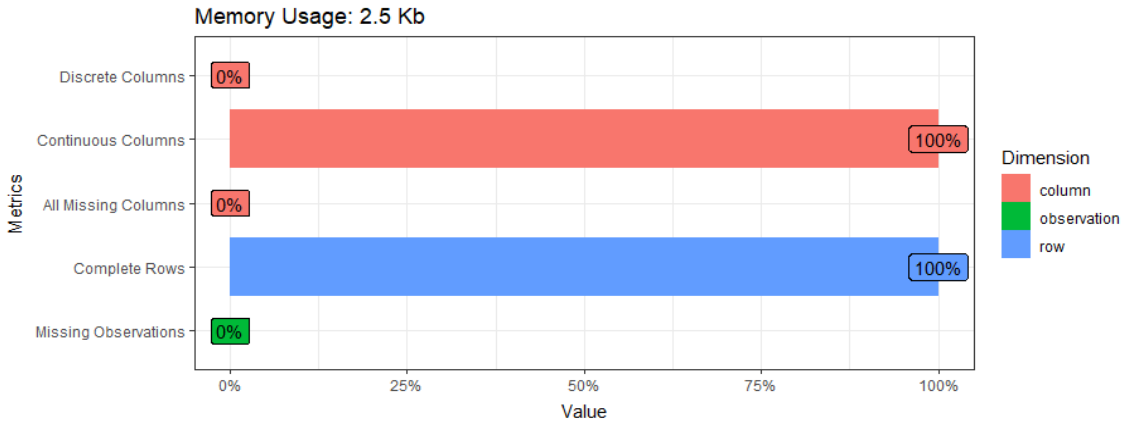


Figure 17: Barplot displaying the data type of the variables and the percentage of complete rows and missing observations.

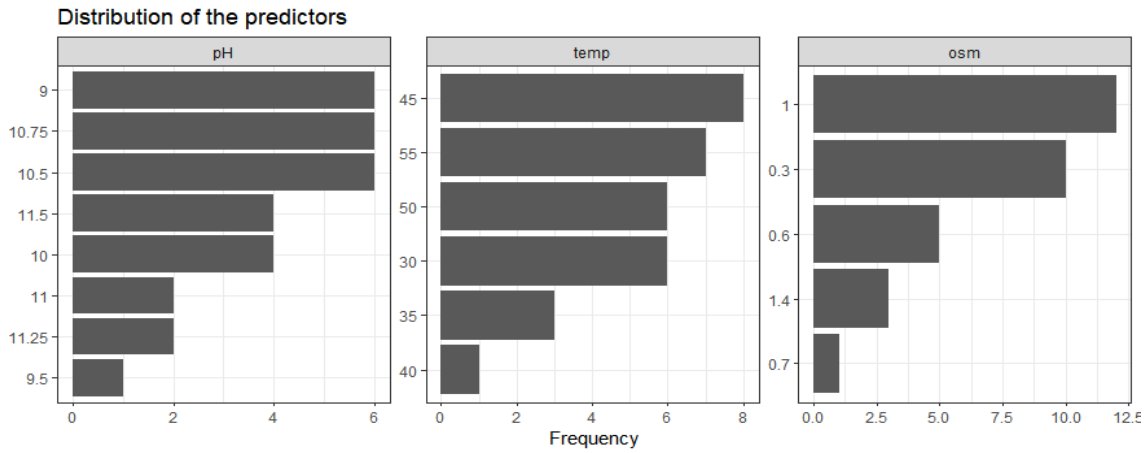


Figure 18: barplots showing the distribution of the predictors (pH, temperature, and osmolarity).

A)

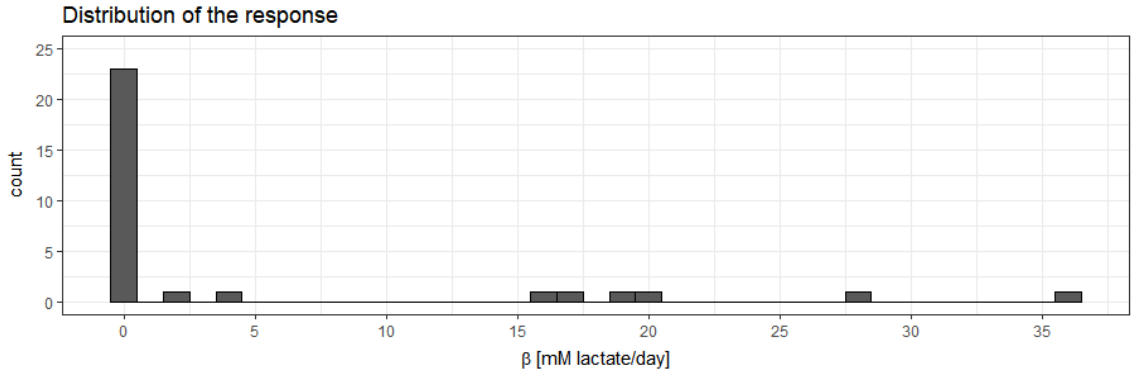


Figure 19: histogram showing the distribution of β.

Nevertheless, it was significantly more interesting to study how β is distributed. To do so, we decided to build a histogram (Fig. 19). The main conclusion that we could draw from this plot was that our response is left skewed and suffers from what is referred to as zero-inflated distribution. In other words, our data is not normally distributed. This hypothesis was confirmed after building a Q-Q plot (Figure 20) and applying some statistical tests (Shapiro Wilk: 1.5e-08, Anderson-Darling: 2.6e-16, Kolmogorov-Smirnov: 5.3e-05).

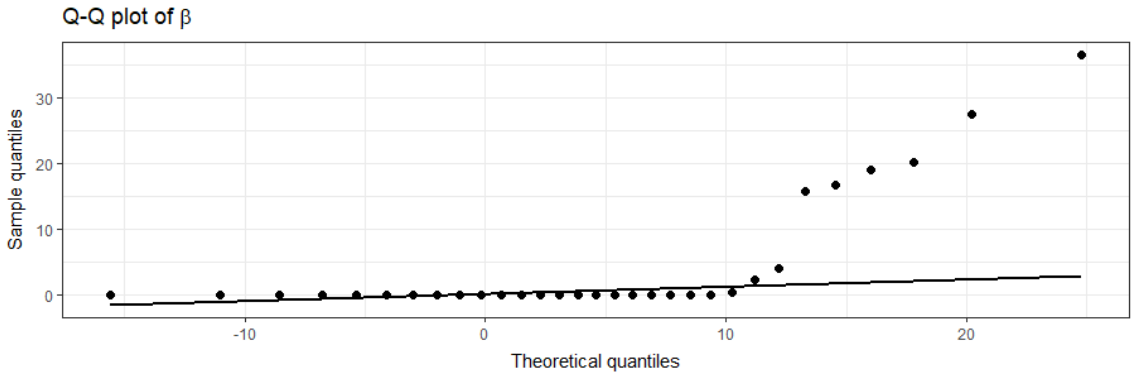


Figure 20: Q-Q plot of β showing the distribution of the sample quantiles versus the distribution of the theoretical quantiles.

This had several implications but the main one was that, since our data is not normally distributed, it was not recommended to use OLS Regression methods to model the data. There were several options that could help us overcome this issue. On the one hand, we could resort to Zero-inflated Poisson (ZIP) or Zero-inflated Negative Binomial Regression. Another interesting solution was to build a mixed prediction model that would combine a classifier with a regressor. The former would filter out those observations that are expected to be zeros and the latter would predict the β. Finally, it could also be wise to just build a classifier instead of a regressor. After all, as previously mentioned, the goal of this analysis was to identify which experimental conditions are safe for our industrial process. In other words, we wanted to make sure that a given experimental condition results into a lactate consumption rate that is lower than a certain value. Thus, by defining this threshold, we could create a new binary variable that could be used to build a classifier instead of a regressor. This way we could simplify the resolution of our problem and -most surely- improve the reliability of our predictions. We will get back to that in the next section. Even though we decided to move towards the construction of a classifier, something that was interesting to study was the covariation within β and each of our predictors (Fig. 21). The covariation between predictors was not assessed because the values of these variables were defined by us and, therefore, they are completely independent from each other.

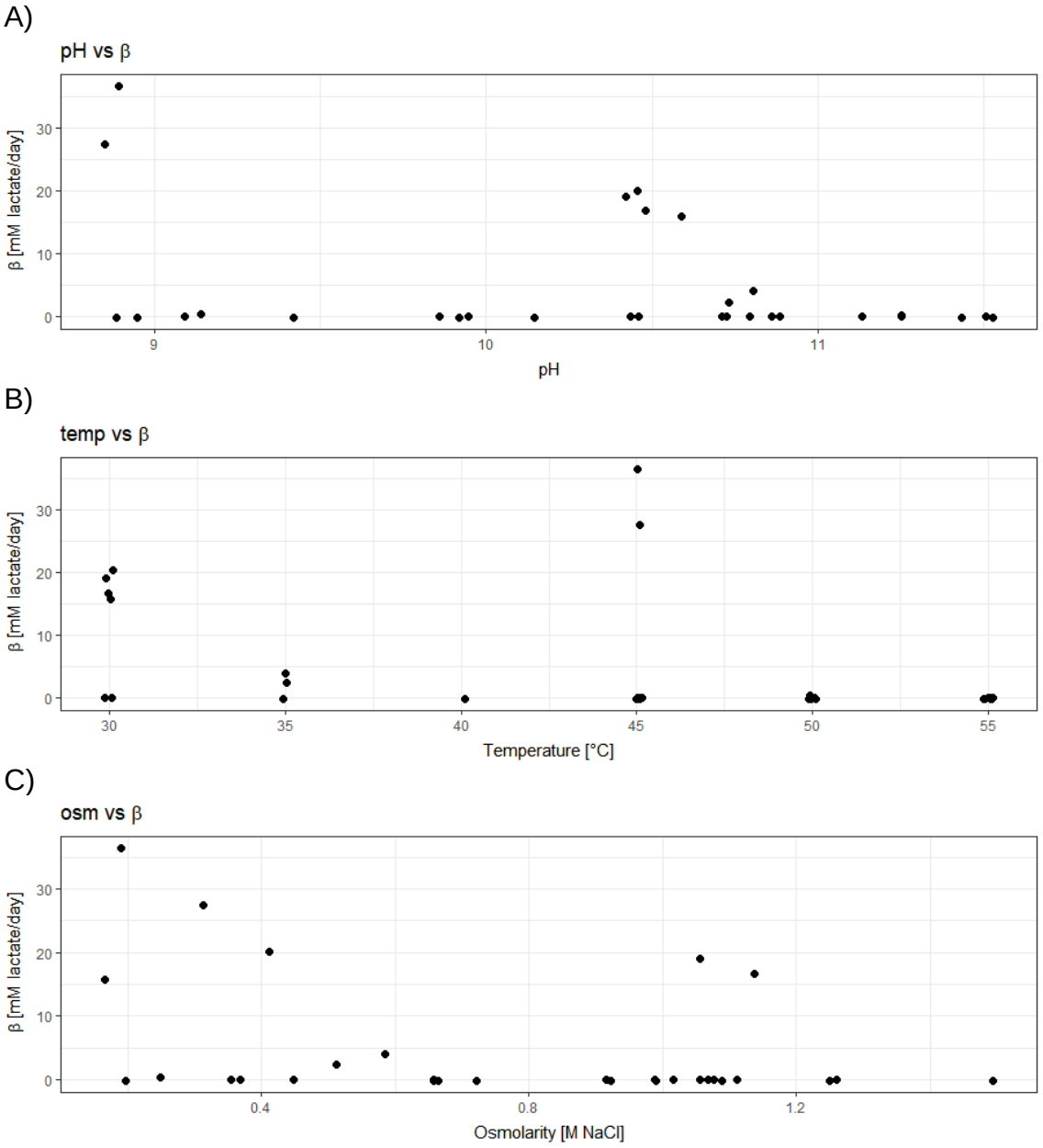
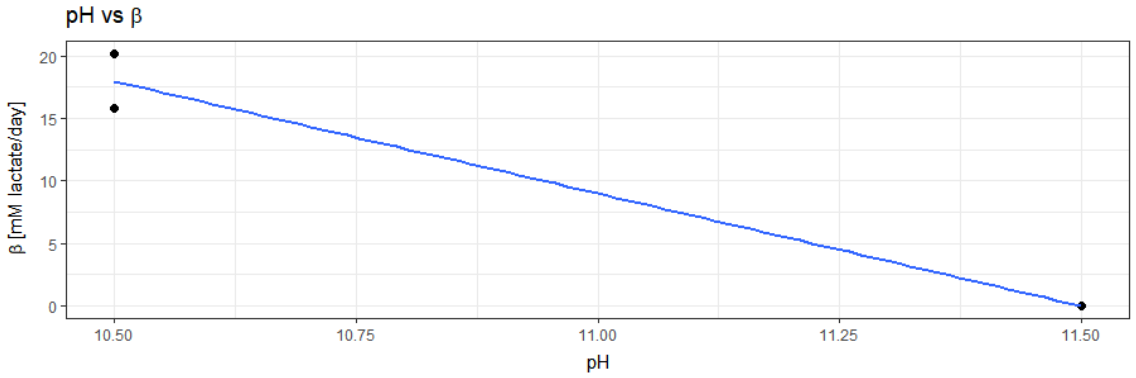


Figure 21: Scatter plots showing the covariation within β and each of the predictors (pH, temp, osm).

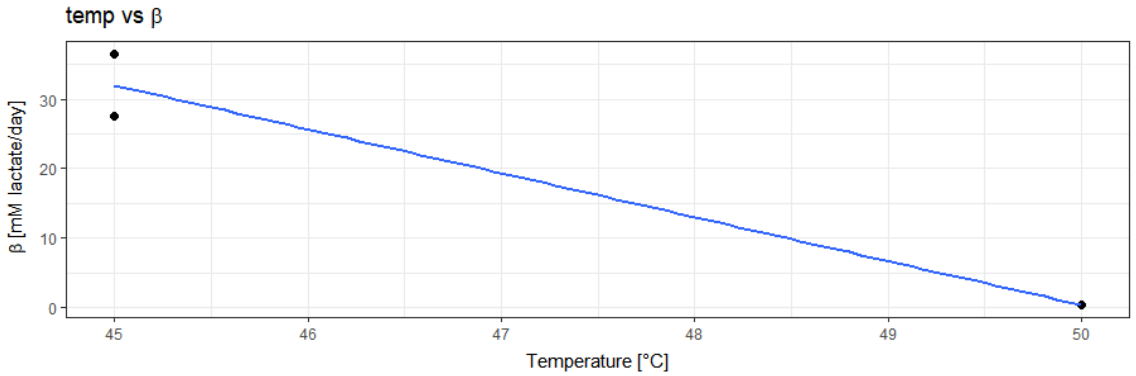
Let's start with the pH (Fig. 21A). After plotting all the observations of our dataset, we observed that, even though there seems to be a negative correlation with β, a lot of observations at low pH have a β = 0. This could be explained by the fact that these observations correspond to experiments that had high temperatures and/or high osmolarities. Therefore, it was more interesting to plot only those observations that belong to experiments carried out at low temperature (30 ˚C) and low π (0.3 M NaCl). As shown in Fig. 22A, by filtering out those data points, the existence of a negative correlation became more obvious. As for the covariation between the temperature and β we observed a very similar pattern.

Once again, although we seem to observe a negative correlation in Fig. 21B, we identified some outliers (e.g., the two data points at 45 that have a β ≈ 30 mM/day). Nevertheless, as soon as we removed those observations pertaining to experiments at high pH (> 9) and high osmolarity (> 0.3 M NaCl), our hypothesis was corroborated (Fig. 22B). Finally, regarding the covariation between π and β, even though the plot containing the whole dataset suggested that there might be a negative correlation within these variables, we also identified some outliers (Fig. 21C).

A)



B)



C)

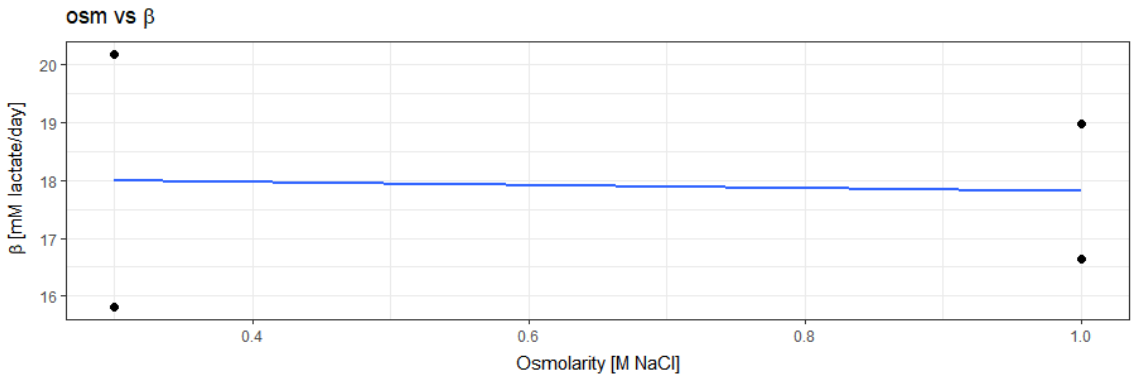


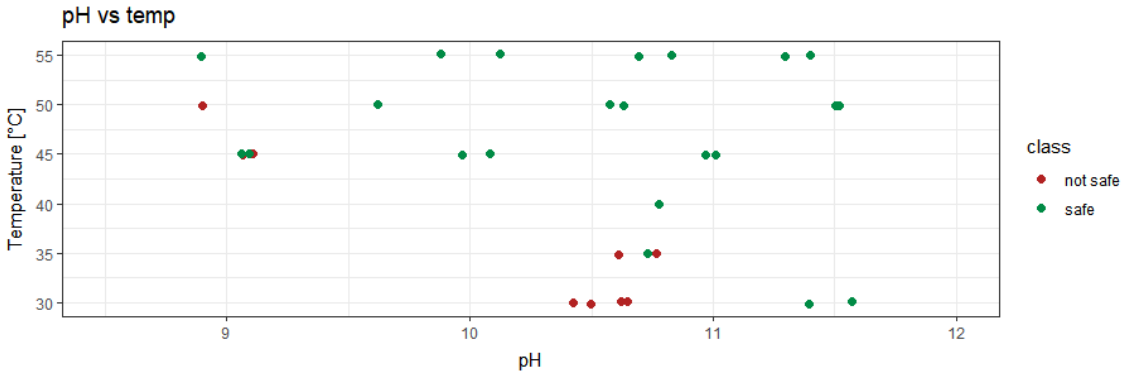
Figure 22: Simplified scatter plots showing the covariation within β and each of the predictors (pH, temp, osm).

Nonetheless, as opposed to the other two predictors, when we filtered out those datapoints with pH ≤ 10.5 and temperature ≤ 30 ˚C, the negative correlation between these variables seemed to be very weak (Fig. 22C).

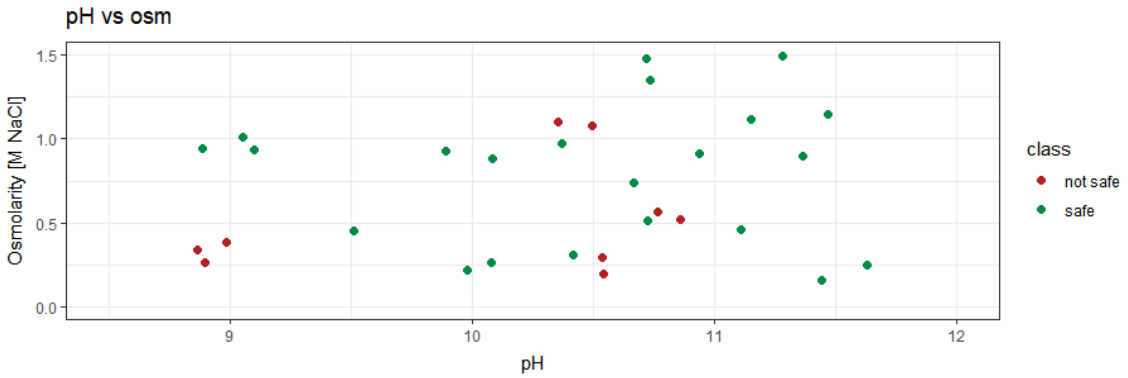
4.2. Creating a new variable

As previously stated in section 4.1, after the EDA, we decided that it would be more adequate to convert β into a binary categorical variable by defining a threshold (0.1 mM/day) that would split our experiments into safe (β ≤ 0.1 mM/day) and not safe (β > 0.1 mM/day) conditions. Hereunder we present a follow-up of the EDA where we studied how our class is distributed for each of the possible combinations between predictors. Although we could observe some clusters, we could not stablish clear boundaries between the two classes (Fig. 23). This was mainly a consequence of the fact that we have 3 predictors that interact with each other and that we were visualizing our observations in a three-dimensional space (x = predictor 1, y = predictor 2, colorscale = class). In consequence, we decided that it would be more representative to plot all these observations in a 4D plot were x = pH, y = temperature, z = π and colorscale = class.

A)



B)



C)

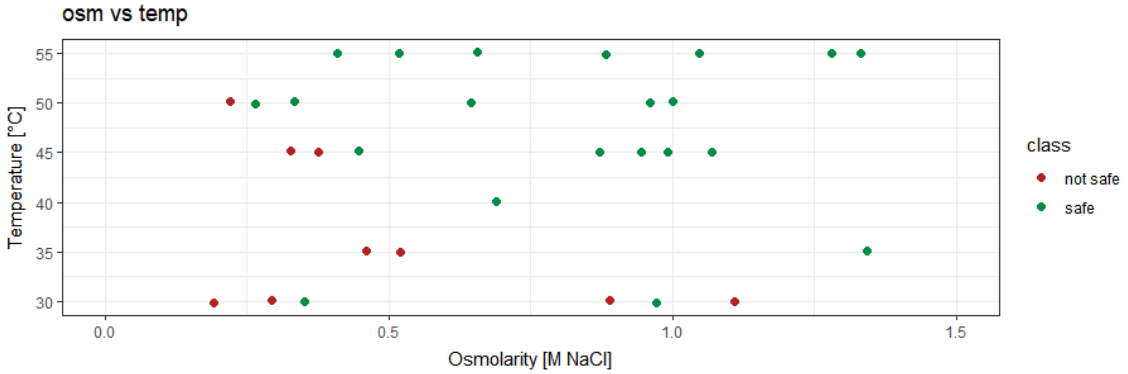


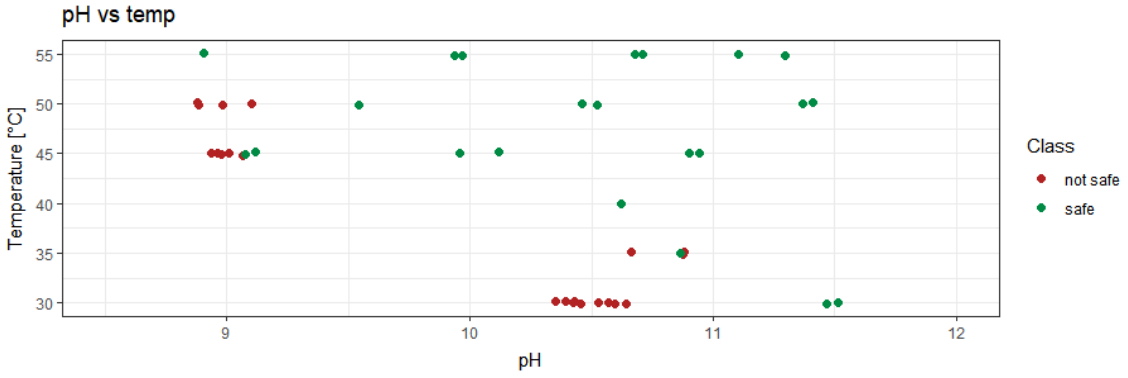
Figure 23: Scatter plots displaying how the variable class is distributed for each of the possible combinations within the predictors (imbalanced dataset).

This plot displayed clearer boundaries. In any case, most importantly, what we could conclude from these plots is that our dataset suffers from class imbalance (29%: safe, 71%: not safe).

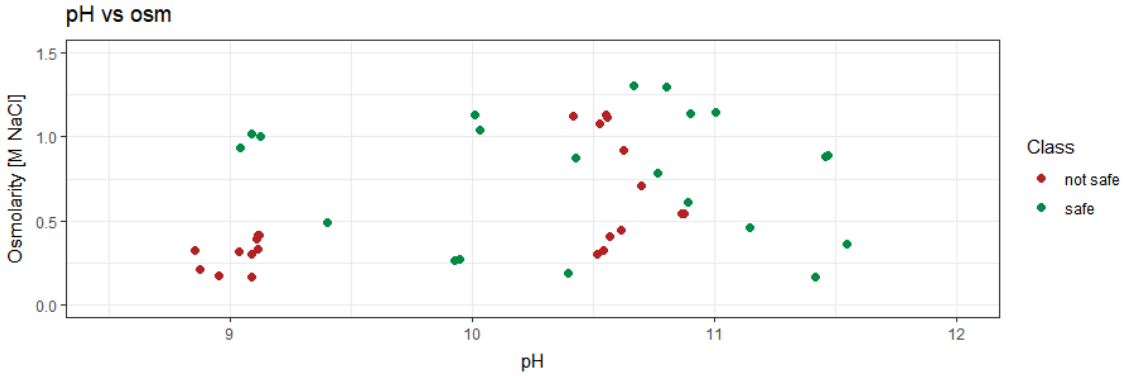
4.3. Balancing the data

Although upsampling methods were only applied over the training sets that were used to build the classifiers, we thought it would be interesting to use it over the whole dataset and to remake the plots presented in Fig. 24.

A)



B)



C)

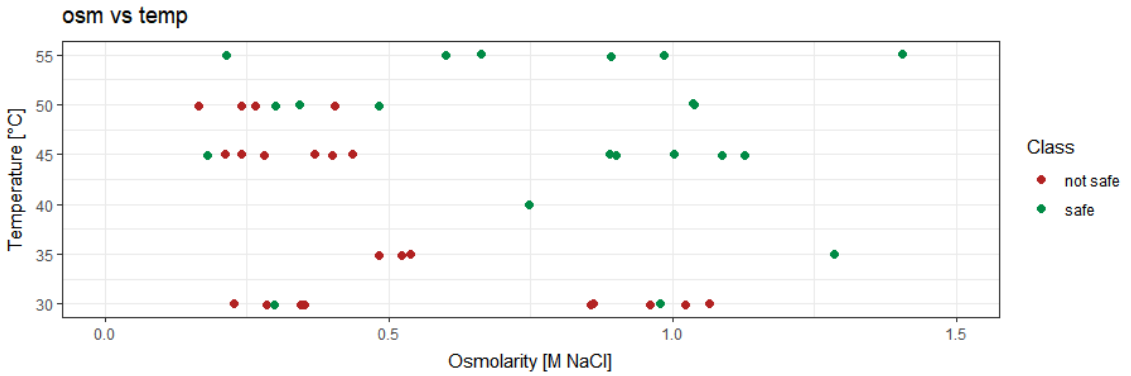


Figure 24: Scatter plots displaying how the variable class is distributed for each of the possible combinations within the predictors (balanced dataset).

This way we could observe the effect that upsampling has over a dataset with class imbalance (Fig. 24). Moreover, similarly to what was done in section 4.2, a 4D plot was generated with the balanced data.

4.4. Construction & optimization of the classifiers

Once the EDA and the modifications over the dataset were finalized, we proceeded to model the data. Regardless of the used algorithm, the core of the protocol was the same. As mentioned in the Materials & Methods section, this process revolved around the construction of an array of classifiers and the adjustment of some of their features. Essentially, these adjustments consisted in balancing the class imbalance of the training set and of tweaking some hyperparameters that are specific for each of the algorithms. Employing a 5-fold Cross Validation with 10 repeats, we evaluated the performance and selected the best classifiers. The parameters used for this evaluation were the accuracy, kappa, sensitivity, sensibility, and ROC. It is important to highlight that we aimed for the models with the highest values for these parameters and with a ratio sensitivity/specificity as close as possible to 1. Therefore, our goal was to obtain a classifier that is as reliable as possible and that is equally good at predicting both classes or -at least- a bit better at predicting which experiments are safe. After all, it is worse to predict that a condition is not safe when is safe (false negative or error type II) than to predict that a condition is safe, test it, and realize that is not safe (false positive or error type I). Hereunder we present the obtained results.

4.4.1. Naïve Bayes (NB)

To begin with, we will talk about the simplest of the selected algorithms: Naïve Bayes. As previously discussed, despite its simplicity and the assumption that all

predictors are independent, for some datasets, this algorithm can achieve performances that are beyond one's expectations. Therefore, we considered that it would be interesting to include it in the analysis. The hyperparameters that were tweaked were the factor of the Laplace correction formula (fL), the kernel density estimate (usekernel) and the bandwidth of the kernel (adjust). Initially, to optimize these hyperparameters, a random search was applied and, once we had a decently optimized classifier, a grid search was carried out. The results obtained from the random search over NB-classifiers built with imbalanced data are presented in Table 4. Note that both fL and adjust were held constant at a value of 1.

usekernel	fL	adjust	acc	kappa	ROC	sens	spec	sens/spec
FALSE	0	1	0.78	0.39	0.87	0.90	0.48	1.87
TRUE	**0**	**1**	**0.79**	**0.46**	**0.91**	**0.88**	**0.57**	**1.54**

Table 4: Results obtained from the hyperparameter random search over NB-classifiers built with an imbalanced training set. Accuracy, kappa, ROC, sensibility, and specificity were used as performance metrics.

The use of a kernel had a positive impact over the accuracy, kappa, ROC, and specificity. Additionally, as a result of the increase over the specificity, the sensitivity experienced a slight decrease, resulting -however- in a better ratio sensitivity/specificity (1.54 instead of 1.87). The improvement was particularly significant for kappa (from 0.39 to 0.46) and the specificity (from 0.48 to 0.57). Same hyperparameters were tested over a set of NB-classifiers constructed with balanced data (Table 5).

usekernel	fL	adjust	acc	kappa	ROC	sens	spec	sens/spec
FALSE	0	1	0.77	0.45	0.85	0.79	0.70	0.89
TRUE	**0**	**1**	**0.8**	**0.56**	**0.9**	**0.77**	**0.87**	**1.13**

Table 5: Results obtained from the hyperparameter random search over NB-classifiers built with a balanced training set. Accuracy, kappa, ROC, sensibility, and specificity were used as performance metrics.

Similarly to what we observed for the imbalanced data, the use of the kernel density estimation led to a better classifier. Although the accuracy was very similar to what we observed for the imbalanced data, the kappa increased from 0.46 to 0.56. Most importantly, the specificity increased from 0.57 to 0.87 and overtook the sensitivity (0.77), leading to a ratio sensitivity/specificity slightly higher than 1 (1.13). Finally, once it was proved that applying the kernel density estimation and using a balanced training set were beneficial, a grid search was applied to screen 11 equidistant values ranging from 0 to 1 for fL and adjust. The highest accuracy and kappa were reached for fL = 0.5 and adjust = 0.7 (Figure 25).

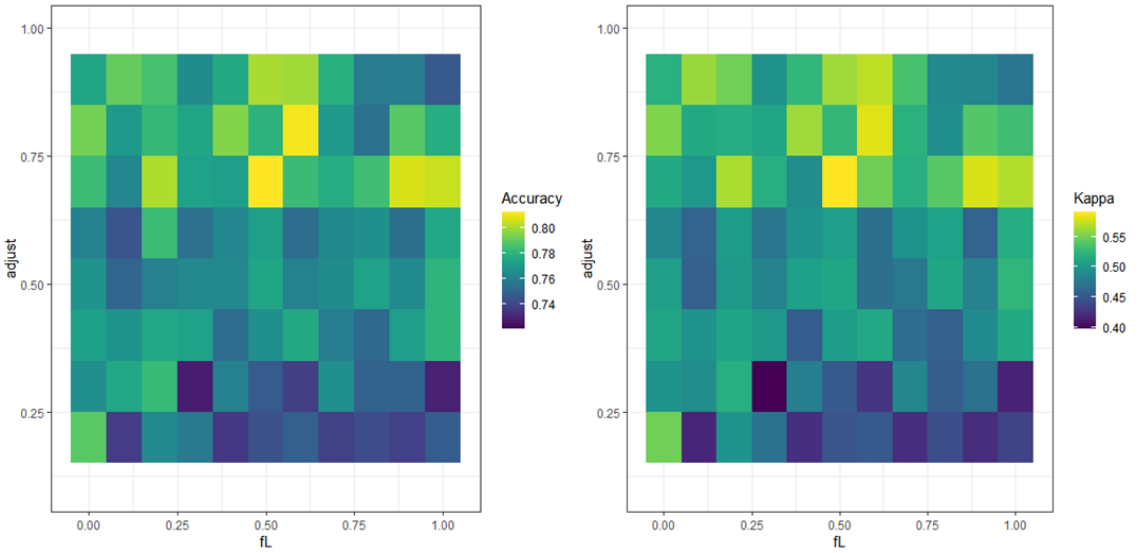


Figure 25: Heatmap showing the results obtained from the grid search over NB-classifiers built with a balanced training set. The combination of different values of adjust and fL was tested over the accuracy and kappa.

Therefore, our classifier could be improved a bit further (Table 6).

usekernel	fL	adjust	acc	kappa	ROC	sens	spec	sens/spec
TRUE	0.5	0.7	0.81	0.59	0.90	0.80	0.85	0.94

Table 6: Hyperparameters and performance metrics of the selected NB-classifier.

4.4.2. Support Vector Machine (SVM)

The next algorithm to be evaluated was Support Vector Machines. Three kernels (linear, polynomial, radial) and their corresponding hyperparameters were tested.

Linear kernel

The results obtained from the hyperparameter random search over linear SVM-classifiers built with imbalanced data are shown in Table 7.

kernel	C	weight	acc	kappa	ROC	sens	spec	sens/spec
Linear	0.25	1	0.80	0.42	0.91	0.91	0.5	1.82
Linear	0.25	2	0.80	0.57	0.94	0.76	0.9	0.85
Linear	0.25	3	0.81	0.62	0.95	0.74	0.99	0.74
Linear	0.5	1	0.78	0.41	0.93	0.87	0.55	1.57
Linear	0.5	2	0.84	0.67	0.95	0.8	0.95	0.85
Linear	0.5	3	0.87	0.74	0.96	0.82	1	0.82
Linear	1	1	0.80	0.49	0.91	0.85	0.68	1.24
Linear	1	2	0.86	0.71	0.94	0.82	0.96	0.86
Linear	**1**	**3**	**0.88**	**0.75**	**0.95**	**0.85**	**0.97**	**0.87**

Table 7: Results obtained from the hyperparameter random search over linear SVM-classifiers built with an imbalanced training set. Accuracy, kappa, ROC, sensibility, and specificity were used as performance metrics.

Regardless of the cost function (C), those classifiers with weight = 3, showed the best overall performance whereas, regardless of the weight, the optimal value for C was 1. As a consequence of class imbalance, most classifiers had a ratio sensitivity/specificity slightly lower than 1, being the selected classifier the one with the closest ratio to 1 (0.87). The use of a balanced training set slightly improved the specificity and overall performance of the classifiers (Table 8). Once again, the optimal value for C was 1 while, this time, the optimal weight was 2. In comparison to the optimal classifier obtained with the imbalanced data, the accuracy, the kappa, the ROC, and the sensitivity experienced a small increase. Although the specificity stayed constant, due to the increase over the sensitivity, the ratio sensitivity/specificity got closer to 1 (0.91).

kernel	C	weight	acc	kappa	ROC	sens	spec	sens/spec
Linear	0.25	1	0.81	0.6	0.95	0.77	0.92	0.84
Linear	0.25	2	0.79	0.59	0.95	0.7	1	0.7
Linear	0.25	3	0.8	0.61	0.96	0.72	1	0.72
Linear	0.5	1	0.86	0.71	0.95	0.82	0.96	0.86
Linear	0.5	2	0.88	0.77	0.95	0.84	1	0.84
Linear	0.5	3	0.88	0.76	0.96	0.83	1	0.83
Linear	1	1	0.86	0.7	0.95	0.85	0.91	0.93
Linear	**1**	**2**	**0.91**	**0.8**	**0.96**	**0.88**	**0.97**	**0.91**
Linear	1	3	0.9	0.79	0.95	0.87	0.97	0.90

Table 8: Results obtained from the hyperparameter random search over linear SVM-classifiers built with a balanced training set. Accuracy, kappa, ROC, sensibility, and specificity were used as performance metrics.

Polynomial kernel

The next kernel to be tested was the polynomial kernel. The tuned hyperparameters were the degree of the polynomial (degree), the scale, and -once again- the cost function (C).

kernel	d	s	C	acc	kappa	ROC	sens	spec	sens/spec
Polynomial	1	0.001	0.25	0.71	0	0.9	0.97	0.05	19.34
Polynomial	1	0.001	0.5	0.71	0	0.93	0.93	0.28	3.34
Polynomial	1	0.001	1	0.71	0	0.93	0.90	0.45	1.99
Polynomial	1	0.01	0.25	0.71	0	0.92	0.91	0.51	1.79
Polynomial	1	0.01	0.5	0.71	0	0.91	0.91	0.51	1.77
Polynomial	1	0.01	1	0.71	0	0.91	0.90	0.56	1.6
Polynomial	1	0.1	0.25	0.71	0	0.91	0.90	0.54	1.66
Polynomial	1	0.1	0.5	0.71	0	0.89	0.88	0.57	1.55
Polynomial	1	0.1	1	0.71	0.07	0.91	0.90	0.53	1.69
Polynomial	2	0.001	0.25	0.71	0	0.93	0.94	0.23	4.07
Polynomial	2	0.001	0.5	0.71	0	0.92	0.92	0.44	2.08
Polynomial	2	0.001	1	0.71	0	0.92	0.89	0.55	1.62
Polynomial	2	0.01	0.25	0.71	0	0.91	0.89	0.49	1.82
Polynomial	2	0.01	0.5	0.71	0	0.91	0.89	0.5	1.78
Polynomial	2	0.01	1	0.71	0	0.91	0.89	0.59	1.51
Polynomial	2	0.1	0.25	0.71	-0.01	0.90	0.90	0.44	2.03
Polynomial	2	0.1	0.5	0.71	0.08	0.90	0.89	0.5	1.78
Polynomial	2	0.1	1	0.78	0.38	0.91	0.88	0.46	1.9

Polynomial	3	0.001	0.25	0.71	0	0.92	0.91	0.47	1.93
Polynomial	3	0.001	0.5	0.71	0	0.92	0.90	0.5	1.8
Polynomial	3	0.001	1	0.71	0	0.92	0.90	0.46	1.96
Polynomial	3	0.01	0.25	0.71	0	0.92	0.91	0.5	1.81
Polynomial	3	0.01	0.5	0.71	0	0.92	0.90	0.54	1.67
Polynomial	3	0.01	1	0.71	0	0.92	0.89	0.58	1.53
Polynomial	3	0.1	0.25	0.70	0.03	0.90	0.88	0.51	1.73
Polynomial	3	0.1	0.5	0.77	0.36	0.91	0.89	0.55	1.61
Polynomial	**3**	**0.1**	**1**	**0.81**	**0.48**	**0.89**	**0.89**	**0.57**	**1.56**

Table 9: Results obtained from the hyperparameter random search over polynomial SVM-classifiers built with an imbalanced training set. Accuracy, kappa, ROC, sensibility, and specificity were used as performance metrics.

The results obtained from the hyperparameter random search over polynomial SVM-classifiers built with imbalanced data are shared in Table 9. Most classifiers performed very poorly, having kappas of 0 and very low specificities. In contrast, the observed sensitivities were very high (≈ 0.90). The hyperparameters that led to the best classifier were degree = 3, scale = 0.1, and C = 1. Nevertheless, the overall performance and, especially, the ratio sensitivity/specificity was worse than what we observed for the linear kernel (1.56 vs 0.9).

kernel	d	s	C	acc	kappa	ROC	sens	spec	sens/spec
Polynomial	1	0.001	0.25	0.66	0.41	0.60	0.52	0.59	0.87
Polynomial	1	0.001	0.5	0.66	0.40	0.60	0.54	0.60	0.9
Polynomial	1	0.001	1	0.65	0.39	0.60	0.55	0.61	0.91
Polynomial	1	0.01	0.25	0.66	0.40	0.60	0.55	0.60	0.92
Polynomial	1	0.01	0.5	0.66	0.41	0.80	0.61	0.83	0.74
Polynomial	1	0.01	1	0.66	0.40	0.90	0.65	0.97	0.67
Polynomial	1	0.1	0.25	0.69	0.45	0.90	0.76	0.87	0.87
Polynomial	1	0.1	0.5	0.76	0.51	0.90	0.81	0.72	1.12
Polynomial	1	0.1	1	0.8	0.60	0.90	0.81	0.88	0.92
Polynomial	2	0.001	0.25	0.66	0.41	0.60	0.56	0.60	0.93
Polynomial	2	0.001	0.5	0.67	0.42	0.60	0.55	0.56	0.97
Polynomial	2	0.001	1	0.65	0.39	0.60	0.53	0.58	0.91
Polynomial	2	0.01	0.25	0.67	0.42	0.80	0.60	0.84	0.71
Polynomial	2	0.01	0.5	0.65	0.39	0.90	0.66	0.97	0.68
Polynomial	2	0.01	1	0.66	0.40	0.90	0.67	0.93	0.72
Polynomial	2	0.1	0.25	0.78	0.55	0.90	0.85	0.67	1.27
Polynomial	2	0.1	0.5	0.8	0.60	0.90	0.83	0.89	0.93
Polynomial	2	0.1	1	0.83	0.66	0.90	0.82	0.89	0.93
Polynomial	3	0.001	0.25	0.67	0.42	0.60	0.53	0.57	0.93
Polynomial	3	0.001	0.5	0.66	0.40	0.60	0.55	0.59	0.93
Polynomial	3	0.001	1	0.67	0.43	0.60	0.56	0.61	0.91
Polynomial	3	0.01	0.25	0.66	0.41	0.90	0.68	0.95	0.71
Polynomial	3	0.01	0.5	0.66	0.41	0.90	0.71	0.95	0.74
Polynomial	3	0.01	1	0.73	0.48	0.90	0.84	0.77	1.09
Polynomial	3	0.1	0.25	0.83	0.68	0.90	0.83	0.84	0.98
Polynomial	**3**	**0.1**	**0.5**	**0.84**	**0.69**	**0.90**	**0.85**	**0.87**	**0.97**
Polynomial	3	0.1	1	0.84	0.67	0.90	0.82	0.87	0.95

Table 10: Results obtained from the hyperparameter random search over polynomial SVM-classifiers built with a balanced training set. Accuracy, kappa, ROC, sensibility, and specificity were used as performance metrics.

Balancing the data helped to improve the performance but it was still worse than what we obtained with the linear kernel (Table 10). This time, the selected hyperparameters were degree = 3, scale = 0.1 and C = 0.5.

Radial kernel

Finally the last kernel to be assessed was the radial kernel. First, the results obtained from the random search over the classifiers constructed with imbalanced data are shown (Table 11). In this case, only the value of the cost function (C) was modified. The bandwidth of the kernel function (σ) was held constant at a value of 0.39. The achieved performances were better than what was obtained for the polynomial kernel but worse than what was observed for the linear kernel. Therefore, these results suggested that amongst the tested kernels, the linear kernel was the best option to model our data.

kernel	C	acc	kappa	ROC	sens	spec	sens/spec
Radial	0.25	0.71	0	0.94	0.83	0.77	1.08
Radial	0.5	0.75	0.23	0.94	0.83	0.78	1.06
Radial	**1**	**0.86**	**0.67**	**0.97**	**0.85**	**0.82**	**1.04**

Table 11: Results obtained from the hyperparameter random search over radial SVM-classifiers built with an imbalanced training set. Accuracy, kappa, ROC, sensibility, and specificity were used as performance metrics.

This hypothesis was confirmed when we used the balanced training set (Table 12). Although all the performance metrics improved significantly and the overall performance was comparable to what we obtained for the best linear classifier, making use the Occam's razor principle, we decided to stick to the linear kernel.

kernel	C	acc	kappa	ROC	sens	spec	sens/spec
Radial	0.25	0.81	0.6	0.96	0.83	0.86	0.97
Radial	0.5	0.84	0.65	0.99	0.89	0.88	1.01
Radial	**1**	**0.88**	**0.74**	**0.99**	**0.93**	**0.96**	**0.97**

Table 12: Results obtained from the hyperparameter random search over radial SVM-classifiers built with a balanced training set. Accuracy, kappa, ROC, sensibility, and specificity were used as performance metrics.

Grid search (Linear kernel)

Now that we knew which was the best kernel and that we already had a very promising classifier, we decided to conduct a grid search by testing a sequence of values for C and weight (from 0 to 5, by 0.25). The results of this screening are shown in Figure 26 in the form of a heatmap. The optimal hyperparameters for this classifier were weight = 3.5 and C = 2.25. The performance metrics obtained for this classifier are presented in Table 13.

The accuracy, ROC, and specificity were perfect or -at least- very close to 1 (0.92, 0.96, and 0.95 respectively). The kappa was also very high (0.81). The only drawback was that the sensitivity was slightly lower than the specificity (0.90 vs 0.95) and, therefore, the ratio sensitivity/specificity was a bit lower than 1 (0.95). In any case, we could conclude that this was the best classifier so far.

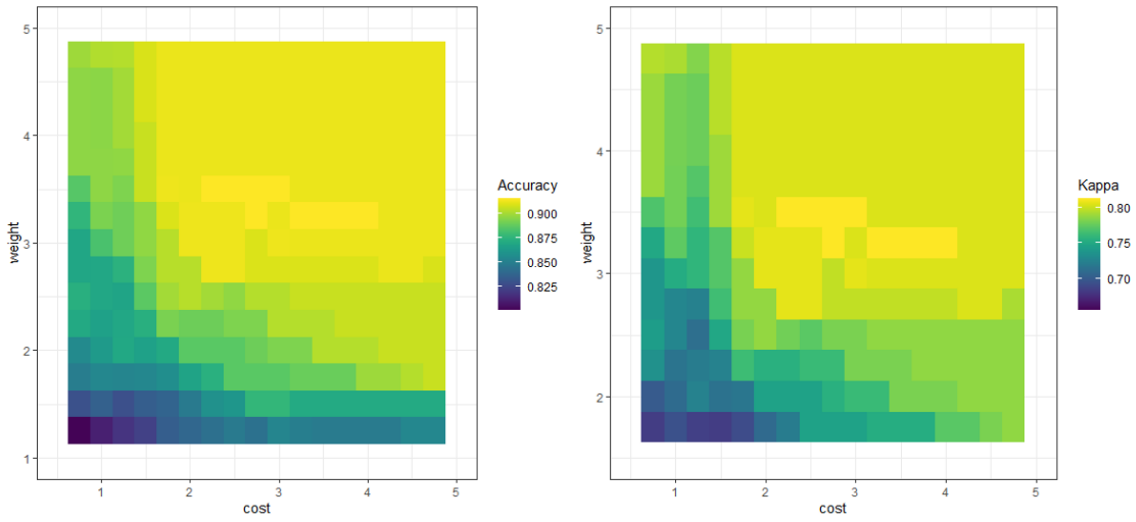


Figure 26: Heatmap showing the results obtained from the grid search over linear SVM-classifiers built with an imbalanced training set. The combination of different values of C and weights were tested over the accuracy and kappa.

kernel	C	weight	acc	kappa	ROC	sens	spec	sens/spec
Linear	2.25	3.5	0.92	0.81	0.96	0.90	0.95	0.95

Table 13: hyperparameters and performance metrics of the selected SVM-classifier.

4.4.3. Random Forest (RF)

Then, the next algorithm to be assessed was Random Forest. As previously done for NB and SVM we started with the imbalanced data, and -afterwards- we moved to the balanced data. Three hyperparameters were evaluated: the regularization value (coefReg), the importance coefficient (coefImp) and the number of variables randomly sampled as candidates at each split (mtry). Hereunder the results obtained from the hyperparameter random search over RF-classifiers built with imbalanced data are presented (Table 14). Regardless of the values of the hyperparameters, all classifiers displayed very similar performances (acc: 0.82-0.83; kappa: 0.63-0.66, ROC: 0.91-0.93, sensitivity: 0.78-0.8, specificity: 0.91-0.93). Moreover, the sensitivity/specificity ratio was slightly lower than 1 (0.85-0.86) suggesting that our classifiers have a bit of a struggle with predicting the positive class (safe). In contrast to what we observed for NB and SVM, the use of a balanced training set did not have a significant impact over the performance metrics (Table 15).

mtry	coefReg	coefImp	acc	kappa	ROC	sens	spec	sens/spec
2	0.01	0	0.82	0.63	0.91	0.79	0.92	0.85
2	0.01	0.5	0.82	0.63	0.92	0.79	0.92	0.85
2	0.01	1	0.83	0.66	0.92	0.8	0.93	0.86
2	0.505	0	0.82	0.63	0.92	0.78	0.92	0.85
2	0.505	0.5	0.82	0.64	0.93	0.79	0.92	0.86
2	0.505	1	0.82	0.64	0.93	0.8	0.91	0.88
2	1	0	0.82	0.64	0.92	0.79	0.93	0.85
2	1	0.5	0.82	0.64	0.93	0.79	0.92	0.86
2	1	1	0.83	0.65	0.93	0.79	0.93	0.85
3	0.01	0	0.82	0.63	0.91	0.78	0.92	0.85
3	0.01	0.5	0.83	0.64	0.93	0.8	0.92	0.86
3	0.01	1	0.83	0.66	0.92	0.8	0.93	0.86
3	0.505	0	0.83	0.65	0.92	0.79	0.93	0.85
3	**0.505**	**0.5**	**0.83**	**0.66**	**0.93**	**0.8**	**0.93**	**0.86**
3	0.505	1	0.83	0.65	0.92	0.79	0.93	0.85
3	**1**	**0**	**0.83**	**0.66**	**0.92**	**0.8**	**0.93**	**0.86**
3	1	0.5	0.83	0.66	0.91	0.8	0.93	0.86
3	1	1	0.82	0.64	0.92	0.79	0.92	0.86

Table 14: Results obtained from the hyperparameter random search over RF-classifiers built with an imbalanced training set. Accuracy, kappa, ROC, sensibility, and specificity were used as performance metrics.

Nevertheless, the best model so far was built with a balanced training set (mtry = 3, coefReg = 1, and coefImp = 0).

mtry	coefReg	coefImp	acc	kappa	ROC	sens	spec	sens/spec
2	0.01	0	0.81	0.62	0.93	0.77	0.93	0.83
2	0.01	0.5	0.81	0.6	0.93	0.77	0.91	0.84
2	0.01	1	0.82	0.62	0.93	0.8	0.9	0.88
2	0.505	0	0.81	0.6	0.93	0.78	0.9	0.86
2	0.505	0.5	0.83	0.65	0.93	0.79	0.93	0.85
2	0.505	1	0.82	0.64	0.95	0.78	0.94	0.83
2	1	0	0.82	0.63	0.93	0.78	0.93	0.84
2	1	0.5	0.81	0.63	0.92	0.77	0.93	0.83
2	1	1	0.83	0.66	0.95	0.8	0.94	0.85
3	0.01	0	0.82	0.63	0.93	0.78	0.93	0.84
3	0.01	0.5	0.81	0.62	0.93	0.77	0.92	0.84
3	0.01	1	0.82	0.64	0.93	0.79	0.92	0.86
3	0.505	0	0.81	0.61	0.94	0.77	0.92	0.83
3	0.505	0.5	0.8	0.6	0.94	0.77	0.91	0.84
3	0.505	1	0.81	0.6	0.93	0.78	0.89	0.88
3	**1**	**0**	**0.84**	**0.67**	**0.94**	**0.81**	**0.94**	**0.86**
3	1	0.5	0.83	0.65	0.96	0.79	0.93	0.85
3	1	1	0.83	0.65	0.96	0.8	0.93	0.86

Table 15: Results obtained from the hyperparameter random search over RF-classifiers built with a balanced training set. Accuracy, kappa, ROC, sensibility, and specificity were used as performance metrics.

Lastly, we decided to carry out a grid search to fine tune the hyperparameters. The mtry was fixed at 3 whereas the same sequence was defined for coefReg

and coefImp (from 0 to 1, by 0.01). The outcome of this search is presented in Figure 27 in the form of a heatmap.

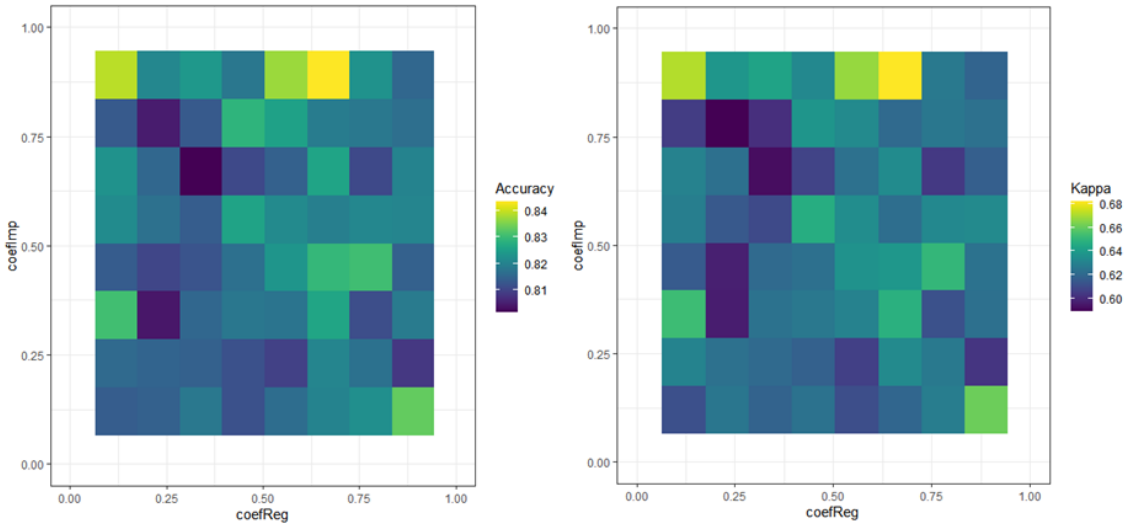


Figure 27: Heatmap showing the results obtained from the grid search over RF-classifiers built with an imbalanced training set. The combination of different values of coefReg and coefImp were tested over the accuracy and kappa.

The optimal hyperparameters amongst the tested combinations were coeReg = 0.67 and coefImp = 0.89. The obtained performance metrics were acc: 0.84, kappa: 0.68, ROC: 0.94, sens: 0.81, spec: 0.95 (Table 16).

mtry	coefReg	coefImp	acc	kappa	ROC	sens	spec	sens/spec
3	0.67	0.89	0.84	0.68	0.94	0.81	0.95	0.85

Table 16: hyperparameters and performance metrics of the selected RF-classifier.

4.4.4. Artificial Neural Network (ANN)

Finally, it was the turn for the Artificial Neural Network. Two hyperparameters were evaluated: the number of units in the hidden layer (size) and the regularization parameter (decay). The results obtained from the random search over classifiers built with imbalanced data are presented in Table 17. Although most classifiers had a decent performance, those with decay = 0 were slightly better. On the other hand, for this decay, the optimal size was 5. Balancing the training set had a very modest effect on improving the performance (Table 18). The observed accuracy for all the classifiers was above 0.86 whereas the kappa was higher than 0.70. Regarding the best classifiers, whereas the accuracy and the sensitivity did not change (0.89 and 0.87 respectively), the kappa went slightly up (from 0.74 to 0.76). Moreover, the ROC and the specificity increased (from 0.89 to 0.90 and from 0.91 to 0.94).

Nevertheless, in order to avoid the risks linked to oversampling, we decided to select the classifier built with an imbalanced training set (size = 5, decay = 0).

size	decay	acc	kappa	ROC	sens	spec	sens/spec
1	0.00E+00	0.86	0.63	0.93	0.9	0.75	1.2
1	1.00E-04	0.86	0.62	0.94	0.92	0.72	1.28
1	1.00E-01	0.75	0.21	0.95	0.95	0.25	3.8
3	0.00E+00	0.88	0.72	0.9	0.88	0.88	1
3	1.00E-04	0.87	0.71	0.91	0.86	0.89	0.97
3	1.00E-01	0.78	0.3	0.95	0.94	0.35	2.69
5	**0.00E+00**	**0.89**	**0.74**	**0.89**	**0.87**	**0.91**	**0.96**
5	1.00E-04	0.87	0.69	0.93	0.85	0.89	0.96
5	1.00E-01	0.78	0.32	0.95	0.94	0.37	2.54

Table 17: Results obtained from the hyperparameter random search over ANN-classifiers built with an imbalanced training set. Accuracy, kappa, ROC, sensibility, and specificity were used as performance metrics.

size	decay	acc	kappa	ROC	sens	spec	sens/spec
1	0.00E+00	0.88	0.71	0.94	0.89	0.87	1.02
1	1.00E-04	0.89	0.75	0.94	0.9	0.89	1.01
1	1.00E-01	0.87	0.73	0.95	0.85	0.92	0.92
3	0.00E+00	0.89	0.74	0.89	0.89	0.89	1.00
3	1.00E-04	0.88	0.72	0.92	0.87	0.89	0.98
3	1.00E-01	0.89	0.76	0.95	0.86	0.94	0.91
5	**0.00E+00**	**0.89**	**0.76**	**0.90**	**0.87**	**0.94**	**0.93**
5	1.00E-04	0.86	0.70	0.92	0.85	0.90	0.94
5	1.00E-01	0.87	0.72	0.95	0.83	0.95	0.87

Table 18: Results obtained from the hyperparameter random search over ANN-classifiers built with a balanced training set. Accuracy, kappa, ROC, sensibility, and specificity were used as performance metrics.

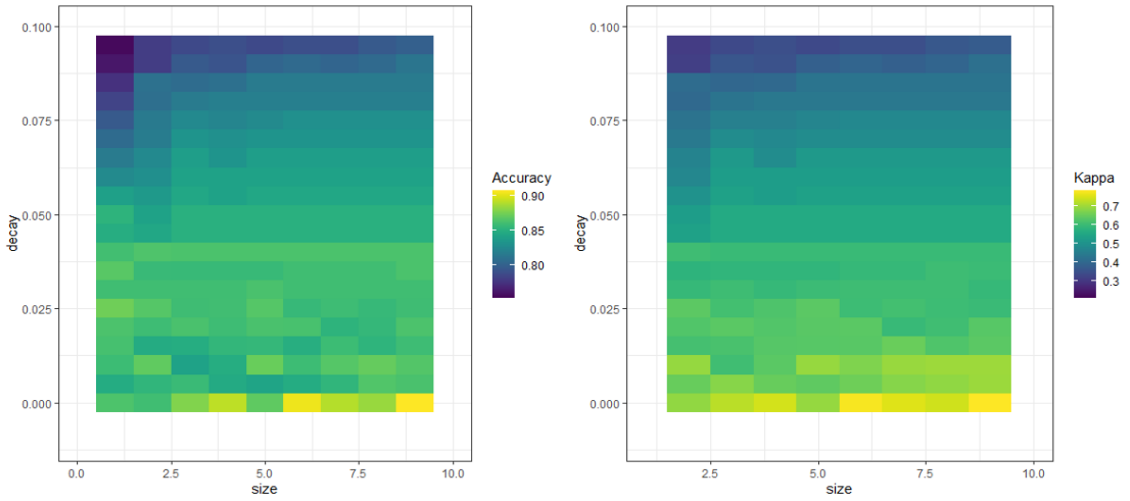


Figure 28: Heatmap showing the results obtained from the grid search over ANN-classifiers built with an imbalanced training set. The combination of different values of decay and size were tested over the accuracy and kappa.

As done for all the other algorithms, a grid search was carried out for the hyperparameters decay and size. For the former, we tested a sequence within 0 and 0.1, and for the latter we used a sequence from 1 to 10. As seen in Figure 28, regarding the accuracy, the optimal hyperparameters were size = 9 and decay = 0. The performance metrics of the classifier constructed with these values are shown in Table 19.

size	decay	acc	kappa	ROC	sens	spec	sens/spec
9	0.00E+00	0.91	0.78	0.91	0.91	0.90	1.01

Table 19: Hyperparameters and performance metrics of the selected ANN-classifier.

4.5. Selecting the best algorithm

Once the optimal classifier for each of the algorithms was selected, we decided to contrast their performances. As seen in Table 20 and Fig. 29, the best classifier was the SVM, followed very closely by the ANN. Although they had similar accuracies (0.92 vs 0.91), the kappa was slightly higher for the SVM (0.81 vs 0.78). As for the sensitivity and specificity, whereas the former was a bit higher for the ANN (0.91 vs 0.90), the latter was a bit higher for the SVM (0.95 vs 0.90).

algorithm	hyperparameters	acc	kappa	ROC	sens	spec	sens/spec
NB	Usekernel = T, fL = 0, adjust = 1, balanced train	0.81	0.59	0.90	0.8	0.85	0.94
SVM	C = 2.25, weights = 3.5, imbalanced train	0.92	0.81	0.96	0.90	0.95	0.95
RF	mtry = 3, coefReg = 0.67, coefImp = 0.89, balanced train	0.84	0.68	0.94	0.81	0.95	0.85
ANN	size = 9, decay = 0, imbalanced train	0.91	0.78	0.91	0.91	0.90	1.01

Table 20: Hyperparameters, accuracy, kappa, ROC, sensibility, specificity, and ratio sensibility/specificity of the best classifiers obtained for each of the algorithms.

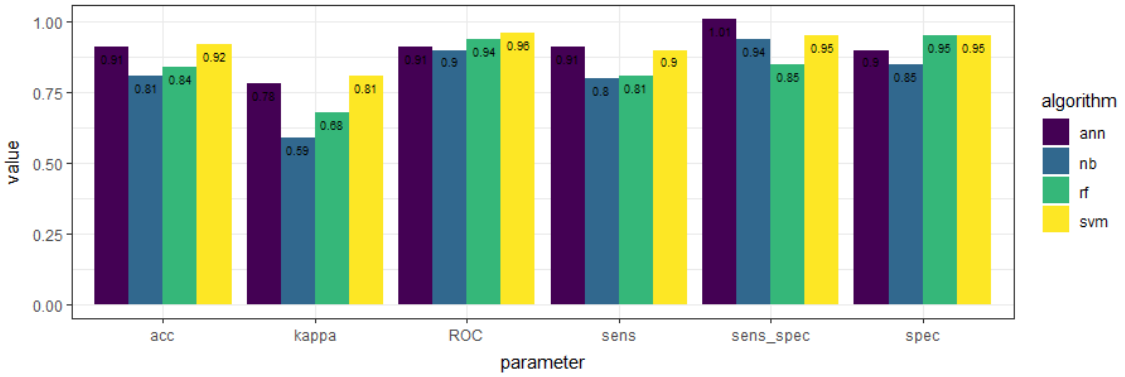


Figure 29: Barplot containing the accuracy, kappa, ROC, sensibility, specificity, and ratio sensibility/specificity of the best classifiers obtained for each of the algorithms.

Therefore, since the reliability of these two classifiers was comparable, we decided to use them both to make predictions. In the next section, we will analyze and contrast the outcome of these predictions. Apart from the evaluation of the performance metrics previously presented, we thought that it could be interesting to compare the training time and prediction time (over 1100000 experimental conditions) of each of the algorithms (Table 21). Except for the NB algorithm, the rest of the algorithms had very low training (≈ 0 sec) and prediction times (≈ 1 sec).

algorithm	training time [s]	prediction time [s]
NB	0.02	257.62
SVM	0	0.81
RF	0	1.19
ANN	0.01	1.21

Table 21: Training time and prediction time (over 1100000 experimental conditions) in seconds of the best classifiers for each of the implemented algorithms.

4.6. Making & visualizing predictions

Now that we had the two best classifiers, the next step was to use them to make predictions over unseen data. With that objective, two parallel strategies were used. Whereas the first approach was meant to be a tool to carry out a quick screening over potentially interesting conditions, the second strategy was meant to be a more exploratory and visual alternative that would help us to navigate through the lactate consumption envelope.

4.6.1. Predictions over specific conditions

As stated in the Materials & Methods section, the first approach consisted of generating a short dataset containing few combinations of pH, T & π and, afterwards, using the classifiers to predict whether these conditions were safe or not. Although the ultimate goal of this project was to provide the Chassis team with this tool, we considered that it would be interesting to run some predictions as an example and share them in this report. Therefore, predictions were carried out over 11 experimental conditions (including 3 conditions that we already tested). For each of the conditions, the prediction was repeated 1100000 times and the percentage of the principal outcome was calculated (Table 22).

pH	T [˚C]	π [mM NaCl]	Prediction (SVM)	Prediction (ANN)	Actual value
9	55	0.3	Safe (100%)	Safe (100%)	-
11.5	30	1	Safe (100%)	Safe (100%)	Safe
9.5	38	1	Safe (100%)	Safe (100%)	-
9.75	47	0.3	Safe (100%)	Safe (100%)	-
11	35	0.4	Not safe (100%)	Safe (100%)	-
10.2	46	0.3	Safe (100%)	Safe (100%)	-
9	30	0.3	Not safe (100%)	Not safe (100%)	-

9	45	0.3	Not safe (100%)	Not safe (100%)	Not safe
9.5	45	0.3	Not safe (100%)	Not safe (100%)	-
10	45	0.3	Safe (100%)	Safe (100%)	Safe
9	42	0.6	Not safe (100%)	Not safe (100%)	-

Table 22: Predictions carried out with the SVM and ANN-classifiers over 11 experimental conditions (1100000 repetitions).

4.6.2. Visualizing the consumption envelope

The idea behind the second strategy was to make predictions for hundreds of thousands of combinations of pH, T and π and on plotting them to visualize the decision regions (safe and not safe) defined by the algorithms. The advantage of this approach is that it is more visual and provides a better overall picture of how our classifiers stablish the boundaries within classes. Since in our optimization project we were working with 4 dimensions (pH, T, π and Class), some questions arise on how to deal with the visual representation of the consumption envelope. Mainly, two approaches were considered. On the one hand we decided to fix a specific level for one of the predictors (e.g. π = 0.3 M NaCl) and plot the other two predictors. A colorscale was used to discern between safe and not safe conditions. Additionally, to get a better perception of the overall consumption envelope we built up some interactive 4D plots.

3D plots

Hereunder, three 3D plots containing the predictions carried out by the best SVM-classifier are presented. Three slices were made. The first slice was taken at π = 0.3 M (Fig. 30A), the second slice at T = 30 ˚C (Fig. 30B) and, finally, the last slice was taken at pH = 9 (Fig. 30C). Since the used kernel was linear, the boundaries observed between classes were defined by perfectly straight lines. Moreover, as hypothesized, a negative correlation was observed between the predictors, meaning that the increase of one of these predictors allows us to lower the ones without making the process unsafe. Temperature proved to be the most relevant factor, followed by the pH and, lastly the osmolarity. Some other conclusions that we could draw from these plots are:

- When π = 0.3 M NaCl:
    - Approximately, half of the predictions were safe.
    - And pH ≈ 9, T needs to be ≥ 48 ˚C.
    - And pH ≈ 9.5, T has to be ≥ 46 ˚C.
    - And pH ≈ 10, T needs to be almost 45 ˚C.
    - And pH ≈ 11, T needs to be ≥ 40 ˚C.

- And pH ≈ 11.5, T needs to be ≥ 37 ˚C.

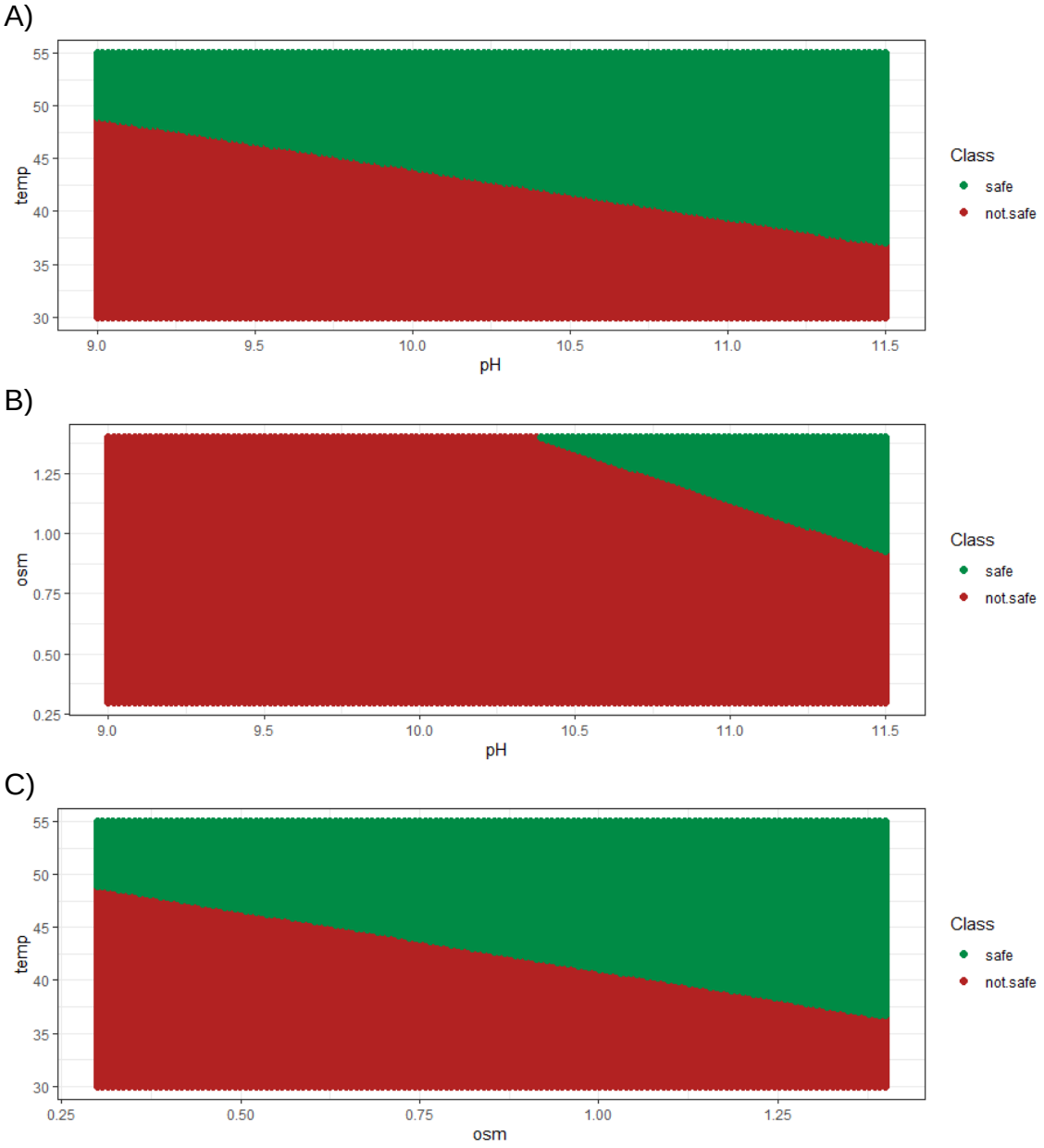


Figure 30: Consumption envelope obtained with the best SVM-classifier. The safe conditions are shown in green whereas the not safe conditions are shown in red. A) pH vs temperature at low osmolarity (0.3 M NaCl). B) pH vs osmolarity at low temperature (30 ˚C). C) Temperature vs osmolarity at low pH (9).

- When T = 30 ˚C:
  - Most predictions were unsafe.
  - And pH = 10.5, π has to be ≥ 1.3 M NaCl.
  - And pH = 11, π needs to be ≥ 1.1 M NaCl.
  - And pH = 11.5, π needs to be ≥ 0.9 M NaCl.

- When pH = 9:
    - Again, approximately half of the predictions are safe.
    - As mentioned before, when π = 0.3 M NaCl, T has to be ≥ 48 ˚C.
    - When π = 0.6 M NaCl, T needs to be ≥ 45 ˚C.
    - When π = 1 M NaCl, T needs to be ≥ 42 ˚C.
    - When π = 1.25 M NaCl, T has to be ≥ 38 ˚C.

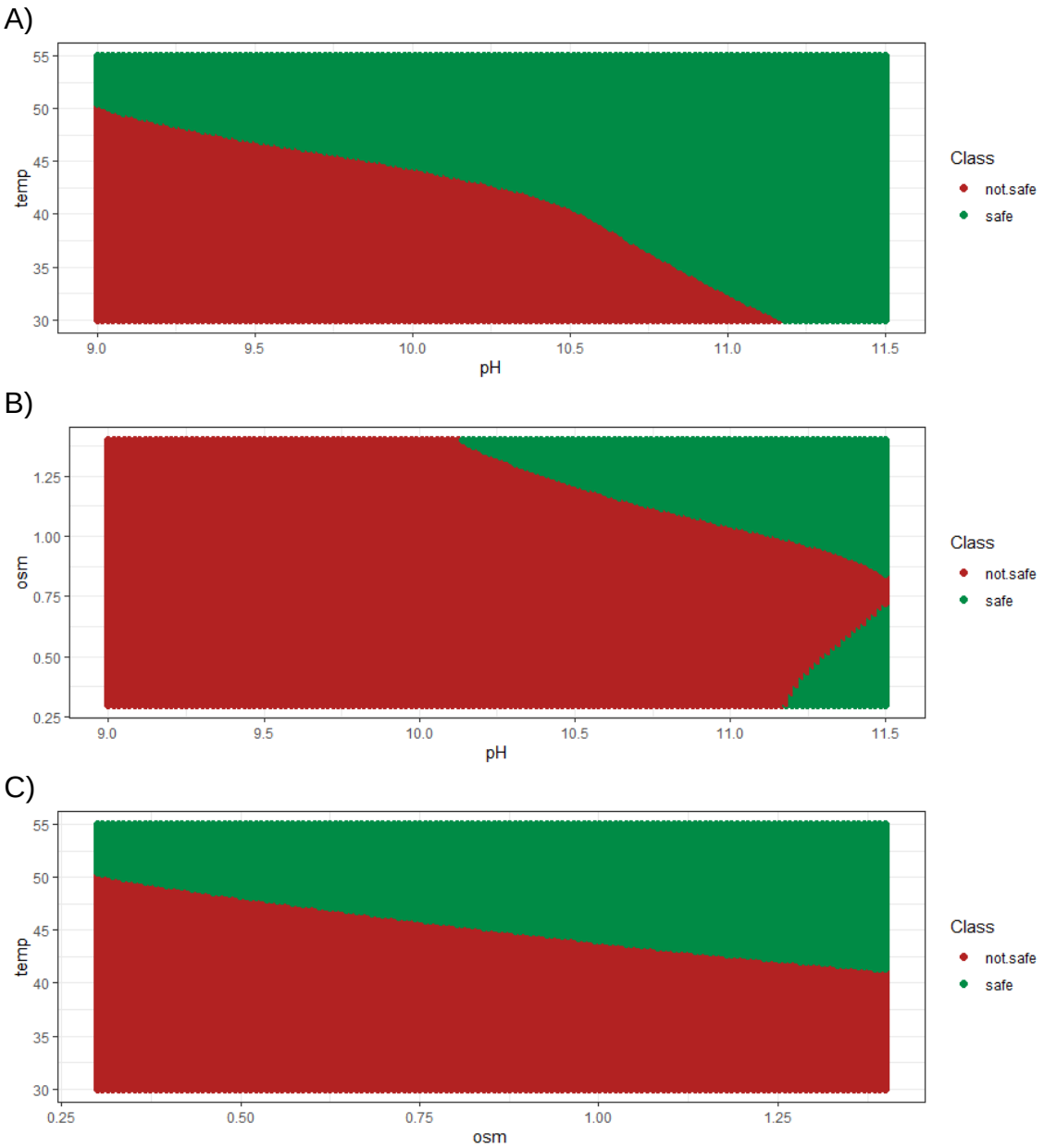


Figure 31: Consumption envelope obtained with the best ANN-classifier. The safe conditions are shown in green whereas the not safe conditions are shown in red. A) pH vs temperature at low osmolarity (0.3 M NaCl). B) pH vs osmolarity at low temperature (30 ˚C). C) Temperature vs osmolarity at low pH (9).

Similarly to what we did for the SVM, we used the ANN-classifier to make predictions over the same experimental conditions. The same slices were taken and contrasted to what was obtained before. Although the conclusions that we could drew from these plots were very similar to what we concluded previously, the consumption envelope predicted by the ANN was slightly different to what the SVM defined (Fig. 31). Apart from the fact that the boundaries between classes were slightly curvy, the algorithm predicted that at π = 0.3 M NaCl, regardless of the temperature, all the conditions with a pH ≥ 11.25 were safe. Moreover, unexpectedly, the classifier predicted that at, T = 30 ˚C, some conditions with pH ≥ 11.25 and low π were safe. Finally, although these results will not be commented in this MTP, in the annex, you can find the 3D plots obtained with the NB and RF-classifiers (Fig. 32 & Fig. 33).

4D plots

Even though we already got a broad idea of how our consumption envelope looks, it is suboptimal to visualize it in a 3D space. In order to avoid taking slices of the consumption envelope and be able to observe the whole space in a unique plot, we constructed three 4D plots for each of the classifiers (all predictions, safe predictions, and not safe predictions). Unfortunately, since this document does not allow the visualization of interactive 4D plots, hereunder we attach few links that bring the reader to a website were a proper visualization can be carried out. The 4D plots obtained with the NB and RF-classifiers are also presented.

- SVM (4D plots): all predictions; safe predictions; not safe predictions.
- ANN (4D plots): all predictions; safe predictions; not safe predictions.
- NB (4D plots): all predictions; safe predictions; not safe predictions.
- RF (4D): all predictions; safe predictions; not safe predictions.

4.7. Validating predictions & updating the model

Even though 5-fold CV with 10 repeats was applied to evaluate the performance of the classifiers, any modelling project needs to culminate with a validation process. Thus, in order to prove the reliability of our selected classifiers, some predictions had to be verified. Due to the limited amount of time, only 4 experimental conditions were selected. The predictions obtained from the best SVM, and ANN-classifiers are presented and contrasted with the actual outcomes in Table 23. Similarly to what was done in section 4.6.1, for each of the experimental conditions, the prediction was repeated 1100000 times. Both the ANN and the SVM classified all the validations correctly, being all the predictions consistent. Therefore, the two algorithms showed a perfect accuracy. Moreover, the outcome of these experiments strengthened the previous hypothesis that -at

low pH- temperature is the most effective factor whereas osmolarity has very moderated effect. In fact, that is why, even when we increase the osmolarity to 0.6 M NaCl, 42 ˚C is not enough to keep the contaminants away from consuming our lactate.

pH	T [˚C]	π [mM NaCl]	Prediction (SVM)	Prediction (ANN)	Actual value
9	45	0.3	Not safe (100%)	Not safe (100%)	Not safe
9.5	45	0.3	Not safe (100%)	Not safe (100%)	Not safe
10	45	0.3	Safe (100%)	Safe (100%)	Safe
9	42	0.6	Not safe (100%)	Not safe (100%)	Not safe

Table 23: Experimental conditions tested for the validations jointly with the predictions made by the selected SVM and ANN-classifiers and the experimental outcomes.

To finalize this project the classifiers were updated with the outcome from the validations and a newer and better version of the classifiers was obtained (v 2.0). Although making predictions with the new version of the classifiers is out of the scope of this project, it is important to understand that the more validations and, in consequence, the more updates that we carry out, the higher robustness our classifiers will have.

5. Conclusions

As previously stated, the goal of this project was to improve the lactate production yield of our bioindustrial process by limiting the activity of the biological pollutants that inhabit our industrial scale photobiorreactors. To do so, three factors were studied (pH, temperature and osmolarity) and the average daily lactate consumption rate of the contaminants in mM/day (β) was monitored. In terms of the experimental design and the modelling, an uncommon but promising procedure was employed. Most articles on process optimization rely either on the COST approach or on the combination of DoE and OLR regression methods. The first option was never considered due to the fact that this method leads to an unnecessarily large number of experiments, and, most importantly, that it does not study the possible interactions among the factors. On the other hand, since our data was not normally distributed, the second option was also discarded. As previously commented, although there exist several options to deal with this problematic (e.g. Zero-Inflated-Poisson or hybrid classifier-regressor models), in this project we decided to combine the DoE approach with Machine Learning algorithms to build an array of classifiers that can predict if certain combinations of the mentioned factors prevent the loss of the lactate that we produce ($\beta < 0.1$ mM lactate/day). Four of the most broadly used ML algorithms were evaluated (Naïve-Bayes, Support vector machines, Random forests, and Artificial neural networks) and some of their main hyperparameters were tweaked. The best classifiers for every of the algorithms were selected and contrasted with each other. The SVM and ANN showed the best performances (accuracy: ≈ 0.9, kappa: ≈ 0.8, sensitivity: ≈ 0.9, specificity: 0.9-0.95) followed by the RF (accuracy: ≈ 0.85, kappa: ≈ 0.7, sensitivity: ≈ 0.8, specificity ≈ 0.95) and lastly by the NB algorithm (accuracy: ≈ 0.8, kappa: ≈ 0.6, sensitivity: ≈ 0.8, specificity ≈ 0.85). Therefore, the ANN and SVM-classifiers were selected and used to run predictions over unseen data and map the lactate consumption envelope. In broad terms, these predictions brought us to the conclusion that temperature is the most important factor to keep the lactate consumption under control, followed by the pH and, finally, the osmolarity. Moreover, the boundaries between those conditions with either an acceptable or unacceptable lactate consumption were defined. This information was shared with the chassis team and their team members were instructed in order to be able -if necessary- to make further predictions.

Although the reliability of the classifiers was higher than expected, these prediction models have some limitations. First, since our dataset had only 31 observations, they might suffer from some issues related to the use of small datasets. This includes a higher sensitivity to outliers and overfitting and/the incapacity to reflect reality. During the course of this project, in order to fix these problems, some powerful tools such as k-fold cross validation and oversampling were used. Whereas the former helped to prevent overfitting and allowed us to

get a better understanding of our models' reliability, the latter did not show a significant positive effect. On the other hand, on the long term it is recommended to carry out more experiments and use additional data to update the classifiers and make them more robust and accurate. Secondly, the dichotomization of β has some serious implications. As previously mentioned, the reasons why we decided to do it were to be able to deal with our non-normally distributed data and to simplify the modelling. Although this process is rarely advised, if a continuous response variable falls into two well-defined categories, it is generally accepted. Nevertheless, it is impossible to fully bypass all the issues related to this transformation. By dichotomizing, you are asserting that there is a straight line between the two classes of your response. For example, in our particular case, by splitting β into Safe and Not safe experiments, you assert that those are the only two values that matter. There is a risk of lactate consumption in Safe, and there is one in Not safe. But what if the consumption rises steadily for a while, then flattens out, then rises again before finally spiking at high values? All of that is lost. Luckily for us, all those experiments with $\beta \leq 0.1$ displayed a β exactly equal to 0 mM lactate/day. In other words, we did not observe any Safe experimental conditions with a $\beta \in (0, 0.1]$ making the problematic substantially less relevant.

Regarding the further steps that this project should take, it is strongly advised to carry out few more validations in order to evaluate and increase the reliability of our classifiers and of the lactate consumption envelope. Besides, we might want to cross the safe lactate consumption envelope with the strain and the industrial envelope in order to obtain the lactate operational envelope. Finally, it could be also interesting to define the product and operational envelope of glycolate, the other major organic acid that is produced in our facilities.

6. Glossary

- **ANN:** Artificial Neural Network.
- **ANOVA:** Analysis of Variance.
- **COST:** Change one factor at a time.
- **CV:** Cross validation.
- **DoE:** Design of Experiments.
- **EDA:** Exploratory Data Analysis.
- **FFD:** Full Factorial Design.
- **FrFD:** Fractional Factorial Design.
- **NB:** Naïve-Bayes.
- **OFAT:** One factor at a time.
- **OLS:** Ordinary least squares.
- **RF:** Random Forest.
- **ROC:** Receiver Operating Characteristic.
- **SVM:** Support Vector Machine.
- **T:** Temperature.
- **ZIP:** Zero Inflated Poisson.
- **β:** Average lactate consumption rate in mM per day.
- **π:** Osmotic strength.

7. Bibliography

1. Kim HW, Vannela R, Zhou C, Rittmann BE. Nutrient Acquisition and Limitation for the Photoautotrophic Growth of Synechocystis sp . PCC6803 as a Renewable Biomass Source. 2011;108(2):277-285. doi:10.1002/bit.22928

2. Council NR. *Biobased Industrial Products: Research and Commercialization Priorities*. The National Academies Press; 2000. doi:10.17226/5295

3. Ducat DC, Way JC, Silver PA. Engineering cyanobacteria to generate high-value products. *Trends Biotechnol*. 2011;29(2):95-103. doi:10.1016/j.tibtech.2010.12.003

4. Touloupakis E, Cicchi B, Benavides AMS, Torzillo G. Effect of high pH on growth of Synechocystis sp. PCC 6803 cultures and their contamination by golden algae (Poterioochromonas sp.). *Appl Microbiol Biotechnol*. Published online 2016. doi:10.1007/s00253-015-7024-0

5. Hasty J, McMillen D, Collins JJ. Engineered gene circuits. *Nature*. 2002;420:224. http://dx.doi.org/10.1038/nature01257

6. Angermayr SA, Hellingwerf KJ, Lindblad P, Mattos MJT De. Energy biotechnology with cyanobacteria. Published online 2008. doi:10.1016/j.copbio.2009.05.011

7. Nguyen BT, Rittmann BE. Electron partitioning in soluble organic products by wild-type and modified Synechocystis sp. PCC 6803. *Biomass and Bioenergy*. 2016;90:237-242. doi:10.1016/j.biombioe.2016.04.016

8. Laspidou CS, Rittmann BE. A unified theory for extracellular polymeric substances, soluble microbial products, and active and inert biomass. *Water Res*. 2002;36(11):2711-2720. doi:10.1016/S0043-1354(01)00413-4

9. Borowitzka MA. Culturing Microalgae in Outdoor Ponds. In: *Algal Culturing Techniques*. ; 2005. doi:10.1016/b978-012088426-1/50015-9

10. Moreno-Garrido I, Cañavate JP. Assessing chemical compounds for controlling predator ciliates in outdoor mass cultures of the green algae Dunaliella salina. *Aquac Eng*. Published online 2001. doi:10.1016/S0144-8609(00)00067-4

11. Rach JJ, Howe GE, Schreier TM. Safety of formalin treatments on warm- and coolwater fish eggs. *Aquaculture*. Published online 1997. doi:10.1016/S0044-8486(96)01447-0

12. Schreier TM, Rach JJ, Howe GE. Efficacy of formalin, hydrogen peroxide, and sodium chloride on fungal-infected rainbow trout eggs. *Aquaculture*. Published online 1996. doi:10.1016/0044-8486(95)01182-X

13. Day JG, Slocombe SP, Stanley MS. Overcoming biological constraints to enable the exploitation of microalgae for biofuels. *Bioresour Technol*. Published online 2012. doi:10.1016/j.biortech.2011.05.033

14. Wang H, Zhang W, Chen L, Wang J, Liu T. The contamination and control of biological pollutants in mass cultivation of microalgae. *Bioresour Technol*. 2013;128:745-750. doi:10.1016/j.biortech.2012.10.158

15. Ruiz Espejo M. Design of Experiments for Engineers and Scientists. *Technometrics*. Published online 2006. doi:10.1198/tech.2006.s381

16. Ebrahimi-Najafabadi H, Leardi R, Jalali-Heravi M. Experimental design in analytical chemistry -Part II: Applications. *J AOAC Int*. Published online 2014. doi:10.5740/jaoacint.SGEEbrahimi2

17. Ignova M, Glassey J, Ward AC, Montague GA. Multivariate statistical methods in bioprocess fault detection and performance forecasting. *Trans Inst Meas Control*. Published online 1997. doi:10.1177/014233129701900507

18. Team RC. R: A Language and Environment for Statistical Computing. *R Found Stat Comput*. Published online 2021.

19. Van Rossum G, Drake Jr FL. *Python Reference Manual*. Centrum voor Wiskunde en Informatica Amsterdam; 1995.

20. Box GEP, Draper NR. *Response Surfaces, Mixtures, and Ridge Analyses: Second Edition*.; 2007. doi:10.1002/0470072768

21. Fukuda IM, Pinto CFF, Moreira CDS, Saviano AM, Lourenço FR. Design of experiments (DoE) applied to pharmaceutical and analytical quality by design (QbD). *Brazilian J Pharm Sci*. 2018;54(Special Issue). doi:10.1590/s2175-97902018000001006

22. Mongomery DC. Montgomery: Design and Analysis of Experiments. *John Willy Sons*. Published online 2017.

23. Mandenius CF, Brundin A. Bioprocess optimization using design-of-experiments methodology. *Biotechnol Prog*. Published online 2008. doi:10.1002/btpr.67

24. Kumar V, Bhalla A, Rathore AS. Design of experiments applications in bioprocessing: Concepts and approach. *Biotechnol Prog*. Published online 2014. doi:10.1002/btpr.1821

25. Jeff Wu CF, Hamada M, Joseph VR. A Review of: “Experiments: Planning, Analysis, and Parameter Design Optimization.” *IIE Trans*. Published online 2006. doi:10.1080/07408170500488964

26. Politis SN, Colombo P, Colombo G, Rekkas DM. Design of experiments (DoE) in pharmaceutical development. *Drug Dev Ind Pharm*. Published online 2017. doi:10.1080/03639045.2017.1291672

27. Tavares Luiz M, Santos Rosa Viegas J, Palma Abriata J, et al. Design of experiments (DoE) to develop and to optimize nanoparticles as drug delivery systems. *Eur J Pharm Biopharm*. Published online 2021. doi:10.1016/j.ejpb.2021.05.011

28. Minatovicz B, Bogner R, Chaudhuri B. Use of a Design of Experiments (DoE) Approach to Optimize Large-Scale Freeze-Thaw Process of Biologics. *AAPS PharmSciTech*. Published online 2021. doi:10.1208/s12249-021-02034-6

29. Rodriguez-Granrose D, Jones A, Loftus H, et al. Design of experiment (DOE) applied to artificial neural network architecture enables rapid bioprocess improvement. *Bioprocess Biosyst Eng*. 2021;44(6):1301-1308. doi:10.1007/s00449-021-02529-3

30. Tu W. Zero-Inflated Data. In: *Encyclopedia of Environmetrics*. ; 2012. doi:10.1002/9780470057339.vaz000g

31. Lambert D. Zero-inflated poisson regression, with an application to defects in manufacturing. *Technometrics*. Published online 1992. doi:10.1080/00401706.1992.10485228

32. Zhou XH, Tu W. Comparison of several independent population means when their samples contain log-normal and possibly zero observations. *Biometrics*. Published online 1999. doi:10.1111/j.0006-341x.1999.00645.x

33. Zhou XH, Tu W. Confidence intervals for the mean of diagnostic test charge data containing zeros. *Biometrics*. Published online 2000. doi:10.1111/j.0006-341X.2000.01118.x

34. Böhning D, Dietz E, Schlattmann P, Mendonça L, Kirchner U. The zero-inflated Poisson model and the decayed, missing and filled teeth index in dental epidemiology. *J R Stat Soc Ser A Stat Soc*. Published online 1999. doi:10.1111/1467-985X.00130

35. Du G, Zhang J, Jiang M, et al. Graph-Based Class-Imbalance Learning With Label Enhancement. *IEEE Trans Neural Networks Learn Syst*. Published online 2021. doi:10.1109/TNNLS.2021.3133262

36. Zhang ML, Li YK, Yang H, Liu XY. Towards Class-Imbalance Aware Multi-Label Learning. *IEEE Trans Cybern*. Published online 2022. doi:10.1109/TCYB.2020.3027509

37. Li DC, Hu SC, Lin LS, Yeh CW. Detecting representative data and generating synthetic samples to improve learning accuracy with imbalanced data sets. *PLoS One*. Published online 2017. doi:10.1371/journal.pone.0181853

38. Henderson AR. The bootstrap: A technique for data-driven statistics. Using computer-intensive analyses to explore experimental data. *Clin Chim Acta*. Published online 2005. doi:10.1016/j.cccn.2005.04.002

39. Kaneko T, Sato S, Kotani H, et al. Sequence analysis of the genome of the unicellular cyanobacterium synechocystis sp. strain PCC6803. II. Sequence determination of the entire genome and assignment of potential protein-coding regions. *DNA Res*. Published online 1996. doi:10.1093/dnares/3.3.109

40. Fuerst JA, Hawkins JA, Holmes A, Sly LI, Moore CJ, Stackebrandt E. Porphyrobacter neustonensis gen. nov., sp. nov., an aerobic bacteriochlorophyll-synthesizing budding bacterium from fresh water. *Int J Syst Bacteriol*. Published online 1993. doi:10.1099/00207713-43-1-125

41. Boldareva EN, Tourova TP, Kolganova T V., Moskalenko AA, Makhneva ZK, Gorlenko VM. Roseococcus suduntuyensis sp. nov., a new aerobic bacteriochlorophyll a-containing bacterium isolated from a low-mineralized soda lake of Eastern Siberia. *Microbiology*. Published online 2009. doi:10.1134/S0026261709010123

42. Yang CX, Liu YP, Bao QH, et al. Mongoliitalea lutea gen. nov., sp. nov., an alkaliphilic, halotolerant bacterium isolated from a haloalkaline lake. *Int J Syst Evol Microbiol*. Published online 2012. doi:10.1099/ijs.0.031286-0

43. Urai M, Aizawa T, Nakagawa Y, Nakajima M, Sunairi M. Mucilaginibacter kameinonensis sp., nov., isolated from garden soil. *Int J Syst Evol Microbiol*. Published online 2008. doi:10.1099/ijs.0.65777-0

44. Boldareva EN, Bryantseva IA, Tsapin A, et al. The new alkaliphilic bacteriochlorophyll a-containing bacterium Roseinatronobacter monicus sp. nov. from the hypersaline Soda Mono Lake (California, United States). *Microbiology*. Published online 2007. doi:10.1134/S0026261707010122

45. Weon HY, Kim BY, Lee CM, et al. Solitalea koreensis gen. nov., sp. nov. and the reclassification of [Flexibacter] canadensis as Solitalea canadensis comb. nov. *Int J Syst Evol Microbiol*. Published online 2009. doi:10.1099/ijs.0.007278-0

46. Khan ST, Takaichi S, Harayama S. Paracoccus marinus sp. nov., an adonixanthin diglucoside-producing bacterium isolated from coastal seawater in Tokyo Bay. *Int J Syst Evol Microbiol*. Published online 2008. doi:10.1099/ijs.0.65103-0

47. Lalucat J, Bennasar A, Bosch R, García-Valdés E, Palleroni NJ. Biology of Pseudomonas stutzeri . *Microbiol Mol Biol Rev*. Published online 2006. doi:10.1128/mmbr.00047-05

48. Zhang J, Jiang RB, Zhang XX, Hang BJ, He J, Li SP. Flavobacterium haoranii sp. nov., a cypermethrin-degrading bacterium isolated from a wastewater treatment system. *Int J Syst Evol Microbiol*. Published online 2010. doi:10.1099/ijs.0.020776-0

49. Spring S, Wagner M, Schumann P, Kämpfer P. Malikia granosa gen. nov., sp. nov., a novel polyhydroxyalkanoate- and polyphosphate- accumulating bacterium isolated from activated sludge, and reclassification of Pseudomonas spinosa as Malikia spinosa comb. nov. *Int J Syst Evol Microbiol*. Published online 2005. doi:10.1099/ijs.0.63356-0

50. Xie Y, Cheng J, Tan X. DT: A Wrapper of the JavaScript Library "DataTables." Published online 2022. https://cran.r-project.org/package=DT

51. Cui B. DataExplorer: Automate Data Exploration and Treatment. Published online 2020. https://cran.r-project.org/package=DataExplorer

52. Wickham H. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York; 2016. https://ggplot2.tidyverse.org

53. Sievert C. *Interactive Web-Based Data Visualization with R, Plotly, and Shiny*. Chapman and Hall/CRC; 2020. https://plotly-r.com

54. Kuhn M. caret: Classification and Regression Training. Published online 2022. https://cran.r-project.org/package=caret

55. Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F. e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. Published online 2022. https://cran.r-project.org/package=e1071

56. Weihs C, Ligges U, Luebke K, Raabe N. klaR Analyzing German Business Cycles. In: Baier D, Decker R, Schmidt-Thieme L, eds. *Data Analysis and Decision Support*. Springer-Verlag; 2005:335-343.

57. Liaw A, Wiener M. Classification and Regression by randomForest. *R News*. 2002;2(3):18-22. https://cran.r-project.org/doc/Rnews/

58. Venables WN, Ripley BD. *Modern Applied Statistics with S*. Fourth. Springer; 2002. https://www.stats.ox.ac.uk/pub/MASS4/

59. C. C, V V. Support Vector networks. *Mach Learn*. Published online 1995.

60. Sain SR, Vapnik VN. The Nature of Statistical Learning Theory. *Technometrics*. Published online 1996. doi:10.2307/1271324

61. Burges CJC. A tutorial on support vector machines for pattern recognition. *Data Min Knowl Discov*. Published online 1998. doi:10.1023/A:1009715923555

62. Cristianini N, Shawe-Taylor J. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*.; 2000. doi:10.1017/cbo9780511801389

63. Joachims T. *Learning to Classify Text Using Support Vector Machines*.; 2002. doi:10.1007/978-1-4615-0907-3

64. Decoste D, Schölkopf B. Training invariant support vector machines. *Mach Learn*. Published online 2002. doi:10.1023/A:1012454411458

65. Moghaddam B, Yang MH. Learning gender with support faces. *IEEE Trans Pattern Anal Mach Intell*. Published online 2002. doi:10.1109/34.1000244

66. Hua S, Sun Z. Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*. Published online 2001. doi:10.1093/bioinformatics/17.8.721

67. Zhang Z. Naïve bayes classification in R. *Ann Transl Med*. Published online 2016. doi:10.21037/atm.2016.03.38

68. Lowres N, Duckworth A, Redfern J, Thiagalingam A, Chow CK. Use of a machine learning program to correctly triage incoming text messaging replies from a cardiovascular Text-based secondary prevention program: Feasibility study. *JMIR mHealth uHealth*. Published online 2020. doi:10.2196/19200

69. Zhu J, Wang H, Zhang X. Discrimination-based feature selection for multinomial naïve Bayes text classification. In: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. ; 2006. doi:10.1007/11940098_15

70. Shen Y, Li Y, Zheng HT, Tang B, Yang M. Enhancing ontology-driven diagnostic reasoning with a symptom-dependency-aware Naïve Bayes classifier. *BMC Bioinformatics*. Published online 2019. doi:10.1186/s12859-019-2924-0

71. Shaban WM, Rabie AH, Saleh AI, Abo-Elsoud MA. Accurate detection of COVID-19 patients based on distance biased Naïve Bayes (DBNB) classification strategy. *Pattern Recognit*. Published online 2021. doi:10.1016/j.patcog.2021.108110

72. Wesolowski M, Suchacz B. Artificial neural networks: Theoretical background and pharmaceutical applications: A review. *J AOAC Int*. Published online 2012. doi:10.5740/jaoacint.SGE_Wesolowski_ANN

73. Breiman L. *Random Forests*. Springer Nature; 2001. doi:10.1023/A:1010950718922

74. Riddick G, Song H, Ahn S, et al. Predicting in vitro drug sensitivity using random forests. *Bioinformatics*. 2011;27(2). doi:10.1093/bioinformatics/btq628

75. Nimrod G, Szilágyi A, Leslie C, Ben-Tal N. Identification of DNA-binding Proteins Using Structural, Electrostatic and Evolutionary Features. *J Mol Biol*. 2009;387(4). doi:10.1016/j.jmb.2009.02.023

76. Cohen JD, Li L, Wang Y, et al. Detection and localization of surgically resectable cancers with a multi-analyte blood test. *Science (80-)*. 2018;359(6378). doi:10.1126/science.aar3247

77. Wickham H, Averick M, Bryan J, et al. Welcome to the {tidyverse}. *J Open Source Softw*. 2019;4(43):1686. doi:10.21105/joss.01686

78. Wickham H, Girlich M. tidyr: Tidy Messy Data. Published online 2022. https://cran.r-project.org/package=tidyr

8. Annex

8.1. Code

The code used during the course of the MTP can be found in this Git repository.

8.2. Tables

	Ingredient	Formula	mg/L	µmol/L
Trace metal mix	Boric acid	H3BO3	2.86	46.26
	Manganese chloride tetrahydrate	Cl2Mn	1.81	9.15
	Zinc sulfate heptahydrate	ZnSO4	0.22	0.77
	Sodium molybdate	Na2MoO4	0.39	1.61
	Copper sulfate pentahydrate	CuSO4	0.08	0.32
	Cobalt Nitrate hexahydrate	CoN2O6	0.049	0.17
S1a	Magnesium disodium EDTA	MgNa2C10H16N2O8	0	0.00
	Titriplex	Na2C10H16N2O8	5	13.43
	Citric acid	C6H8O7	0	0.00
	Ammonium Iron citrate	C6H8O7 F3+ + NH3	0	0.00
	Ferric Chloride hexahydrate	FeCl3	4	14.80
	Magnesium sulfate heptahydrate	MgSO4	147.88	599.97
S1b	Extra titriplex (S1b)	Na2C10H16N2O8	13.6	36.60
S2 (a&b later)	Sodium nitrate	NaNO3	1000	11763.32
	Calcium chloride dihydrate	CaCl2	36	244.88
S3	Potassium Phosphate dibasic	K2HPO4	40	229.66

LB medium (plates)			
Chemical	**Formula**	**g/L**	**Labcode**
Difco bacto tryptone	$C_{19}H_{14}O_2$	10	T125
Difco yeast extract	-	5	Y5
Sodium chloride	NaCl	5	N59
Agar	$C_{14}H_{24}O_9$	15	A22
BG11-J medium (plates)			
Chemical	**Formula**	**mL/L**	**Labcode**
S1a	See Table A1	2.5	-
S2a	See Table A1	2.5	-
S2b	See Table A1	2.5	-
S3	See Table A1	2.5	-
TES-KOH 1M (pH 9)	$C_6H_{15}NO_6S$	10	T1/T6
Thiosulphate 30%	$S_2O_3^{-2}$	10	T2
Sodium bicarbonate 1M	$NaHCO_3$	10	N5
Glucose 1M	$C_6H_{12}O_6$	10	G1
LB 9x	See Table 23	5.55	-

Table 24: Composition of the LB and BG11-J plates.

8.3. Figures

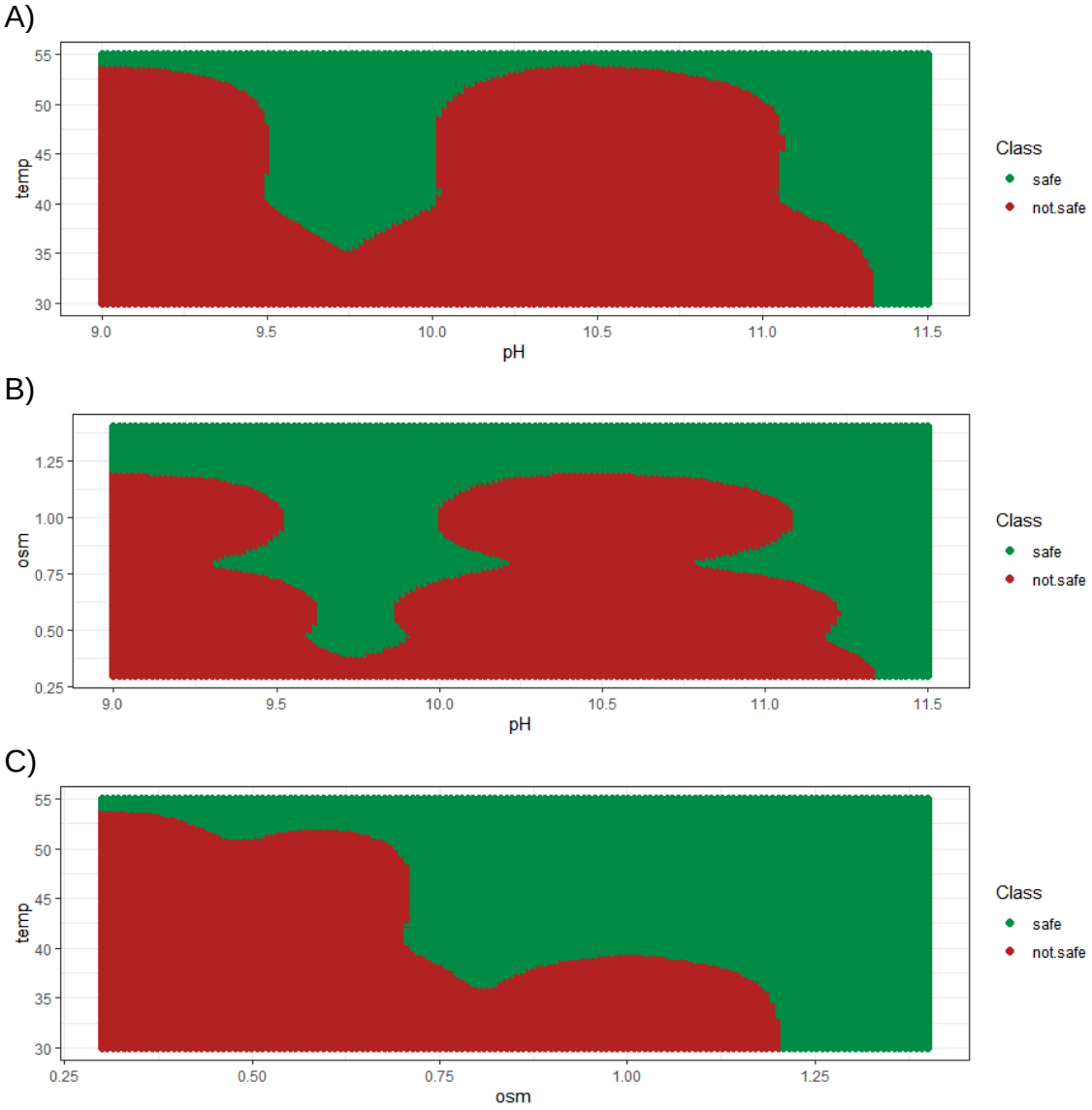
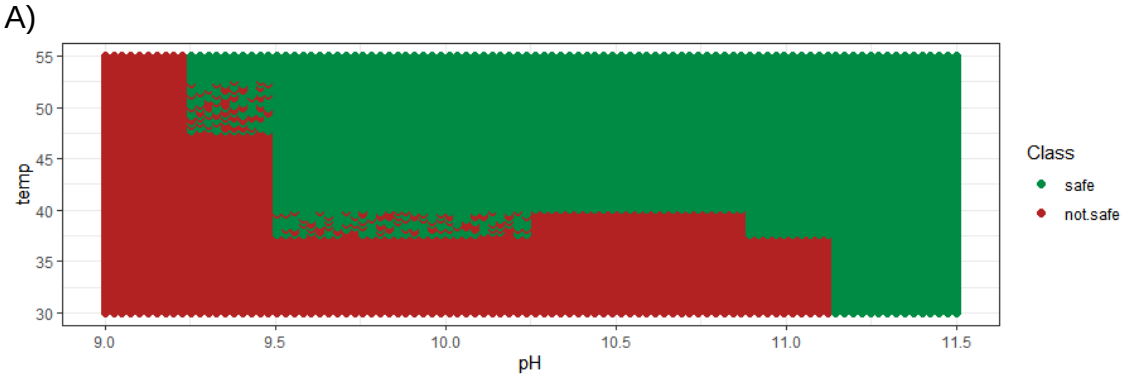


Figure 32: Consumption envelope obtained with the best NB-classifier. The safe conditions are shown in green whereas the not safe conditions are shown in red. pH vs temperature at low osmolarity (0.3 M NaCl). B) pH vs osmolarity at low temperature (30 ˚C). C) Temperature vs osmolarity at low pH (9).



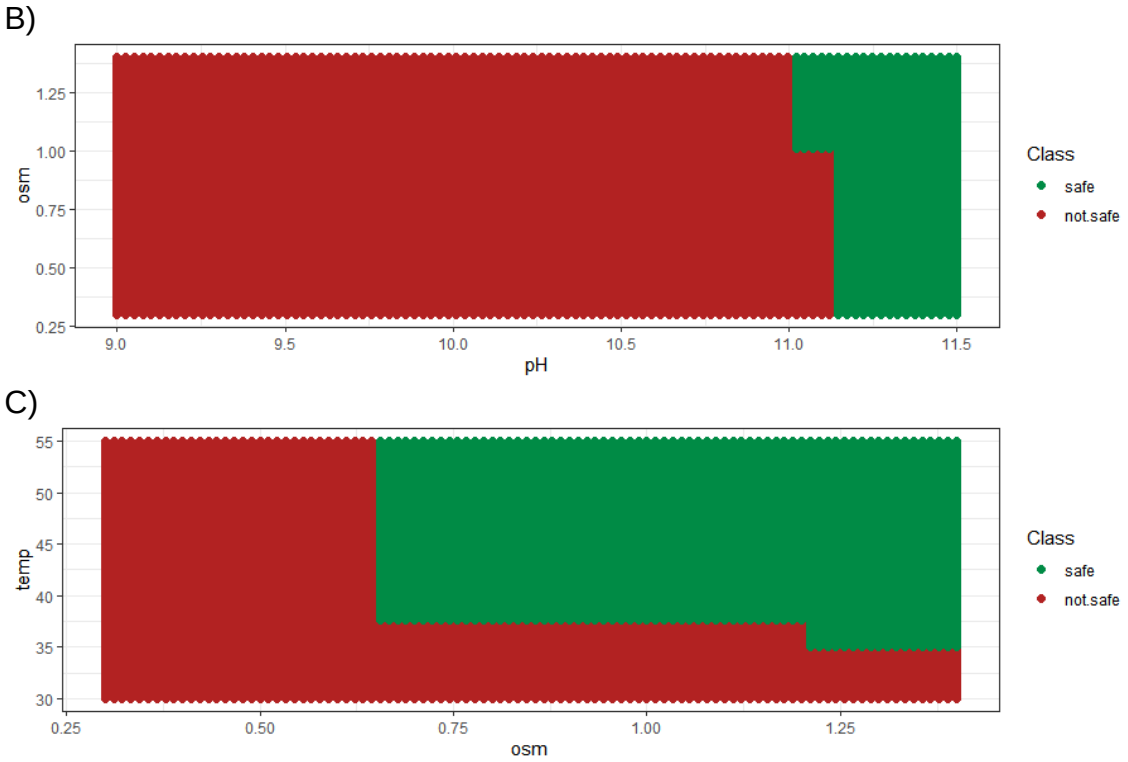


Figure 33: Consumption envelope obtained with the best RF-classifier. The safe conditions are shown in green whereas the not safe conditions are shown in red. pH vs temperature at low osmolarity (0.3 M NaCl). B) pH vs osmolarity at low temperature (30 ˚C). C) Temperature vs osmolarity at low pH (9).