Comparing linear classifiers and nonlinear classifiers on the normalised single-cell Raman spectra of a strain of Chlorella and a strain of Rhodobacter.

Federico Concas

26 August, 2019

Robert's Notes:
- have newlines between paragraphs; it's hard to read.
- References should be ordered. Your first reference is 15

Abstract

Raman spectra are a physical technique that ~~could~~ be used to identify the metabolic state of bacteria in real-time and, therefore, be incorporated in industrial processes. For this reason, it is crucial to find an adequate analysis that allows for fast and accurate predictions. In this dissertation, we compared linear to non-linear algorithms for this task, while establishing a protocol to train the best-performing classifier. Upon comparing Regularized Logistic regression with and without a PCA pre-processing step, Support vector machines with different kernels, and Random forest, on two different data sets, we were able to show that the linear strategy outperformed the non-linear one on both ~~our~~ data sets, amongst the considered algorithms. After repeating the analysis on a subset of the variables, selected for their importance, we showed that the best-performing algorithm is Regularized Multinomial Logistic Regression with a PCA pre-processing step. Our pipeline achieved 96.6% out-of-sample Kappa for Chlorella Raman spectroscopy data and 90.2% out-of-sample Kappa for Rhodobacter data.

Contents

Abstract

Chapter 1 - Introduction to the problem **1**
Discussion of the context . . . 1
What is Raman Spectroscopy . . . 1
Literature Review . . . 2
Description of the study . . . 2

Chapter 2 - Materials and Methods **4**
Data . . . 4
Algorithms and transformations . . . 6
Support Vector Machines (nonlinear) . . . 6
Logistic Regression . . . 8
PCA . . . 9
Random Forest . . . 9
Cross-validation . . . 10
Nested Cross-Validation . . . 11
Kappa . . . 11
Statistical Hypothesis Testing . . . 12

Chapter 3 - Standard analysis of the data **13**

Discussion **17**
Linear algorithms had better performance . . . 17
Why did SVM with the Radial kernel perform so poorly? . . . 17
How did feature selection improve the global performance? . . . 18
Why did PCA work and what are its shortcomings? . . . 18
Are the results generalizable? . . . 19
Further improvements . . . 19

Appendix **20**
Data . . . 20
Tuning iterations . . . 21
Comparison with the full data sets . . . 21
Bayesian hierarchical tests for the full data sets . . . 24
Bayesian hierarchical tests for the selected data sets . . . 25
Methods . . . 25
Logistic Regression . . . 25
PCA . . . 29

Chapter 1 - Introduction to the problem

Discussion of the context

Sludge and wastewater are usually listed under the costs of an engineered plant. The rise of biotechnology is progressively allowing us to convert such cost into a resource, thanks to the advances in industrial microbiology, but also the introduction of novel techniques, such as Raman spectroscopy and Machine Learning.

In wastewater plants, microalgae are essential for the oxygenation of wastewater, which allows the neutralization of pathogenic microorganisms and bad smells coming from the plant. They are also fundamental for the removal of N and P, elements contained in organic compounds that would cause the eutrophication of water sources if contaminated water were dumped directly into the environment. The use of biomass in wastewater plants is estimated to cause a 40% reduction in their lifecycle assessment (LCA). [15] Once the biomass has consumed most of the nutrients, it is extracted from the liquid, which then faces additional treatments. The removed biomass is, de facto, a waste product itself; nevertheless, it is a high-calorie byproduct that can be converted into energy. Aerobic treatments require space for desiccation and can only be done in sunlit and well-ventilated environments. The use of machinery for such purpose is considered even more inefficient, so alternative processes have been suggested. Microalgae Chlorella sp. has been proposed as biomass in the anaerobic production of biogas, to reduce the costs associated with harvesting and drying this high-energy source. [15]

The use in digestors is not exclusive to Chlorella. Microorganisms from the genus Rhodobacter have been used for the production of hydrogen gas in a two-step anaerobic process. [11]

Moreover, both the previously mentioned microorganisms are currently employed in the production of a herbicide using effluents from anaerobic digestors. [21]

It is clear that the widespread adoption of these processes is conditioned on improving their efficiency. Succeeding in such a task will require a system that allows for cheap and real-time diagnostics, and one of these is the determination of the metabolic state of microorganisms.

What is Raman Spectroscopy

Raman spectroscopy is a molecular spectroscopic technique that uses molecular vibrations to determine the molecular structure and chemical composition of a sample. The simplest form of Raman spectroscopy uses the photons in monochromatic light to hit a target and therefore interact with the electron clouds of the sample molecules. The absorbed energy is released in the form of scattered photons, most of which have the same energy as the incident photons. However, some molecules stay locked in a higher or lower vibrational state; hence, 10^{-8} of incident photons have higher or lower energy compared to the ones from the laser. [17] The difference between the energy of the laser photons (i.e. the frequency) and these photons is called Raman shift and it was discovered by Sir C.V. Raman, who won the 1930 Nobel Prize in Physics. Every molecule produces a specific Raman shift; therefore, Raman shifts can be used to infer the chemical composition of a sample, while the intensity at the shifted frequencies provides quantitative information. Taken together, these pieces of information can be used to identify a sample effectively.

~~When it comes to biology~~, Raman spectroscopy has various advantages over other chemical, biological and biochemical techniques. Firstly, every molecule in an organism has a distinct Raman spectrum, so the technique is precise; secondly, it is non-invasive and real-time. Lastly, it is possible in the presence of water so that it can be used directly and even in-vivo. Researchers have started to apply this technique to microbiology, i.e. Yuan et al. [22] used it to identify

bacterial strains without the use of sequencing or biochemical kits; however, most applications have not been explored.

Literature Review

In the past, Raman spectroscopy has been applied to many fields. When it comes to biological samples, it is especially challenging to interpret Raman spectra, due to the wide range of biomolecules that they contain. A solution to this problem is to use multivariate methods, which can handle multidimensional data sets and process all the information simultaneously. To correctly process Raman data, Einax [7] recommends a tidy format in which Raman spectra appear as observations and each Raman shift frequency as a variable.
Raman spectra are noisy as they contain background noise arising from the intrinsic fluorescence of biological molecules, ambient light, thermal noise of the Raman spectrometer, contributions of the culture medium and slight variation of experimental conditions; preprocessing is therefore fundamental and it includes normalization, smoothing and outlier removal. After preprocessing, dimensionality reduction with Principal Component Analysis can be applied to visualize the patterns that are present in the data [8]. Many techniques can be then used, encompassing both supervised and unsupervised algorithms. Specifically for classification, the following models have been trained:

- Linear discriminant analysis, coupled with PCA [@Maquelin2000];
- Hierarchical cluster analysis, coupled with PCA [@Maquelin2000];
- Support vector machines. One of the most popular applications to microbiology was made by @Rosch2005, who used them to identify bacterial contamination;
- Neural networks, e.g. used by @Gniadecka2004 on cancer tissue. An interesting aspect of their work is that the neural network was only fed selected regions of the spectra, based on previous publications in which researchers had identified what regions had the highest difference in Raman intensity between the cancer class and the healthy class.

All these studies use a decision-based metric and agree on using cross-validation to obtain a good estimate, although they disagree on how many folds to use and how many times CV should be repeated.

Description of the study

The aim of this dissertation is to compare the predictive power of linear classification methods to non-linear methods on discretized Raman spectra. The best-performing method for each task is then used to create a classifier which can predict bacterial metabolic states from Raman spectra with great performance.
To do so, we compare regularized Logistic Regression (LR), ~~a Generalized Linear Model~~, to Support Vector Machines, a popular method chosen for its similarity to LR.
We show, empirically, that linear methods seem to outperform non-linear methods. However, a non-parametric hypothesis test of this hypothesis fails to show a statistically significant result, probably due to the lack of power of the test. Therefore, by using a Bayesian hierarchical test, we show that GLMs give better predictions on both data sets.
To ensure a fair comparison between the two method types, we couple Logistic Regression with a pre-processing step with Principal Component Analysis (PCA) of the discretized Raman spectra; this is to check if there is a noticeable improvement in its effectiveness. For the same reason, we compare the Polynomial kernel to the Radial one for Support Vector Machines. Finally, we train Random Forest, a non-linear algorithm that usually outperforms SVMs in the literature, and SVM with a linear kernel. The former ensures that our results do not depend on Logistic

gression being probabilistic and SVM being non-probabilistic, while the latter rules out that SVMs are always better or worse than GLMs. In one data set, SVM with the Linear kernel is the best-performing algorithm, closely followed by Regularized Logistic Regression; on the other data set, the PCA pre-processing step provides an out-of-sample performance increase of 14.7% compared to SVMs with the Linear kernel, which is the second best-performing algorithm, therefore showing that linear methods are better on both data sets. Random forest was the worst-performing algorithm, possibly due to overfitting.

We perform the comparison by using nested cross-validation, a resampling technique which is used to assess the predictive performance of a model, whose (hyper)parameters also need to be tuned. We chose this approach due to the size of our data set, as we must always use unseen observations to obtain an unbiased estimation of the chosen metrics. The Methods section features a more thorough explanation.

From the results of Random Forest, it is possible to extract the importance of the features. By selecting a band of the spectrum in both data sets, where most of the important variables are localized, and performing the comparison again, it is evident that the results improve considerably. Even in this case, linear methods outperform non-linear methods on both tasks.

Chapter 2 - Materials and Methods

The goal of our analysis is to compare the performance of linear to non-linear classifiers of metabolic state for Raman spectroscopy data of algae from the genus Chlorella and bacteria from the genus Rhodobacter.

Data

In this dissertation, we used 171 single cell spectra of one strain from the genus Chlorella and 139 single cell spectra of one strain from the Rhodobacter genus. These were ~~centred~~ around $1300cm^{-1}$. For each of the 498 frequencies of the Raman shift, there is an associated standardized scatter intensity. The data were collected by McIlvenna et al. [14] and obtained using the same setup described in their study.
Figure 1 shows the discretized Raman spectra of the Chlorella and Rhodobacter strains and provides a visual indication of what pattern might be present in the data: in specific bands of the Chlorella spectrum, certain metabolic states seem more separable, whereas the Rhodobacter data set generally seems harder to classify, as the metabolic states are stacked upon each other. Figure {fig:raman2} in the Appendix presents the same plots, faceted by metabolic state Figure {fig:raman2} in the Appendix presents the same plots, faceted by metabolic state.

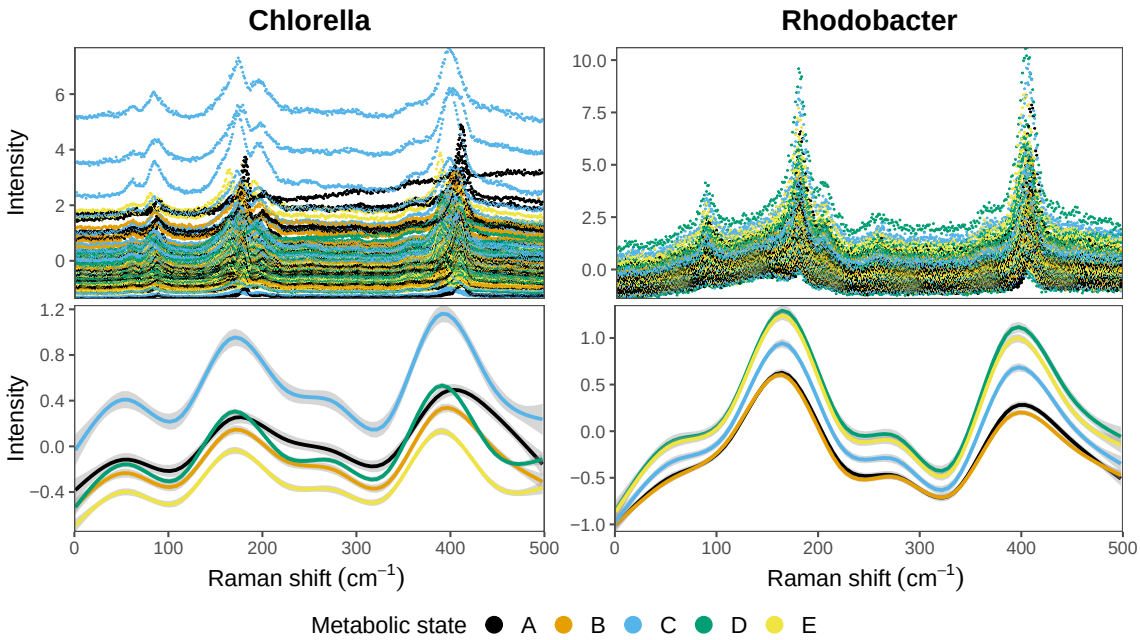


Figure 1: *On top, the discretized Raman spectra of the Chlorella and the Rhodobacter data sets. On the bottom, the same spectra have been smoothed with GAM to visualize the trends better. Each colour in the plots represents a metabolic state.*

Figure 2 shows an extremely high correlation between the covariates; this correlation is present even when we compare variables that are far from each other on the spectrum. This warns us against drawing conclusions from the parameter estimates of the models. The predictive performance is unaffected by multicollinearity.

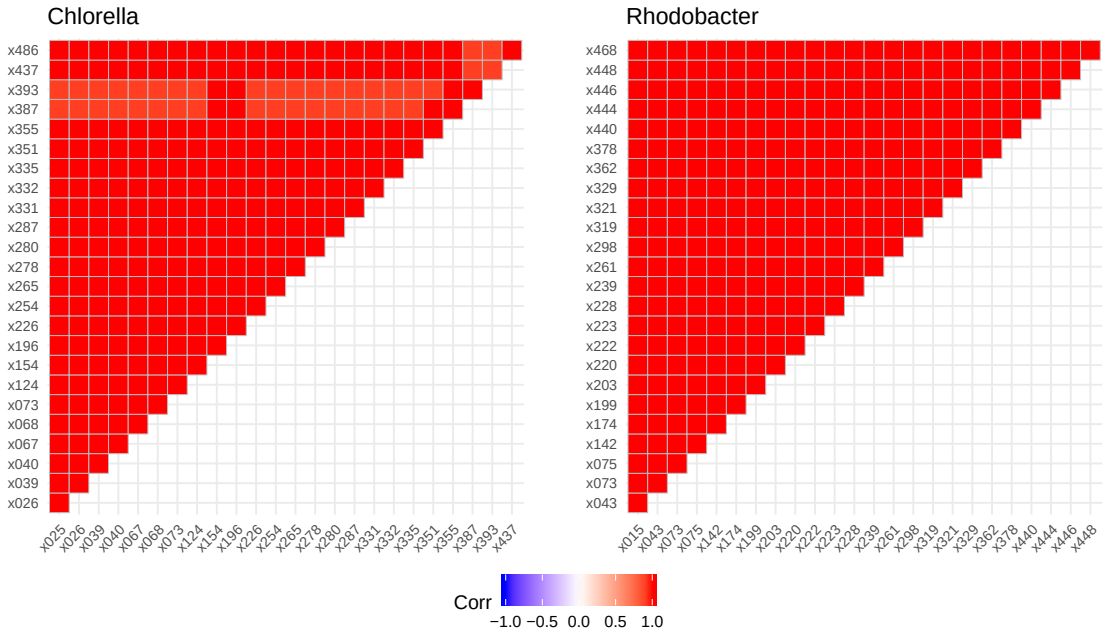


Figure 2: *Covariate correlation plot of a random sample of variables and observations from the Chlorella and the Rhodobacter data sets.*

Performing PCA confirms the strong correlation amongst the variables, as one single component summarises more than 97% of the variability in both the Chlorella and the Rhodobacter data sets (Table 7 in the Appendix).

It also shows visual separation between the classes (Figure 3). This suggests that PCA could be later employed as a pre-processing step before linear algorithms. Interestingly, the second principal component seems to be the dimension on which the data is best separable.

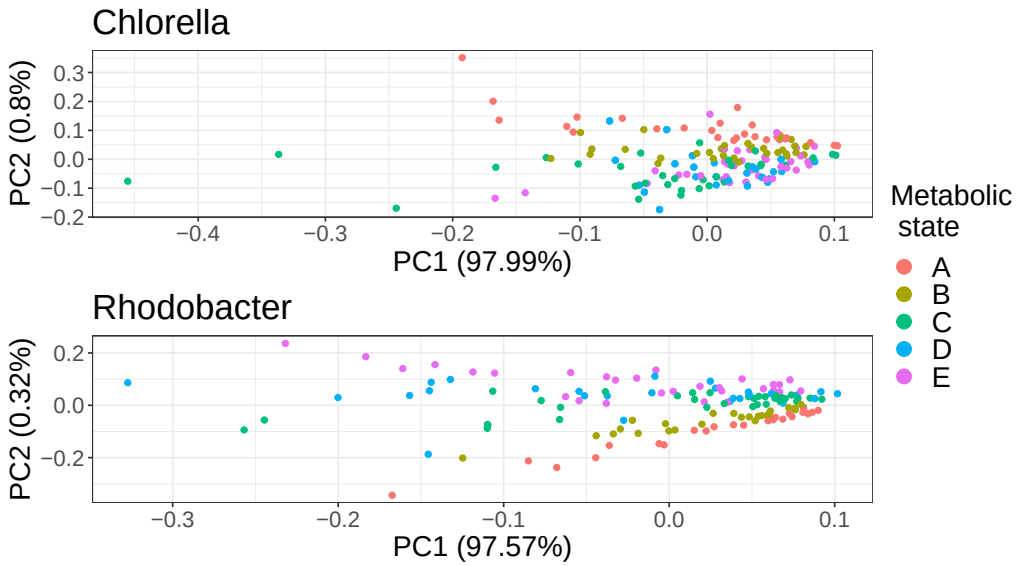


Figure 3: *PCA plot of the Chlorella and Rhodobacter data sets.*

Algorithms and transformations

As our linear classifier, we chose Logistic Regression, due to its simplicity. As our non-linear algorithm, we chose Support Vector Machines (SVM), due to its widespread use in the literature and its similarity to Logistic Regression. SVM can be both linear and nonlinear, so it allows us to exclude an effect of the chosen method in our results. Both these algorithms tend to overfit on the training data, i.e. they are unable to generalize; for this reason, we used the Elastic Net penalty and a soft margin for Logistic Regression and SVM, respectively.
To ensure a fair comparison between the two classes of methods, we used a PCA pre-processing step before Logistic Regression, two more kernels for SVM (one linear and one non-linear), and Random Forest, a non-linear ensemble model.

Support Vector Machines (nonlinear)

Support Vector Machines, a non-probabilistic classifier, aims at finding a decision boundary that maximizes the *margin*. Let us assume that our data points have only two possible classes so $\mathbb{R}^2$ as seen in Figure 4a, i.e. $y \in \{-1, 1\}$. These values are chosen so that later expressions can be simplified. Let us define a decision boundary (a hyperplane)

$$f(x) = \mathbf{x}^{\mathrm{T}}\theta + b = 0. \tag{1}$$

Then

$$\begin{aligned} \theta^{\mathrm{T}}\mathbf{x}_{\text{ new}} + b > 0 \quad &\rightarrow \quad \widehat{y} = 1 \\ \theta^{\mathrm{T}}\mathbf{x}_{\text{ new}} + b < 0 \quad &\rightarrow \quad \widehat{y} = -1 \end{aligned},$$

which is the signed distance from a point x to Equation 1. It can be shown that, after scaling, the margin is equal to $\frac{1}{||\theta||}$ There are infinite linear hyperplanes that separate the two classes, however there is only one that maximises the margin; maximising the margin is equivalent to computing

$$\begin{aligned} &\underset{\theta}{\operatorname{argmin}} \tfrac{1}{2}\theta^{\mathrm{T}}\theta, \\ &\text{with constraint } y^{(i)}\left(\theta^{\mathrm{T}}\mathbf{x} + b\right) \geq 1. \end{aligned}$$

The data points that rest on the outer boundaries of the margin are called "Support points" and they are indispensable for defining the margin. Removing any other data point would leave the margin intact, however, removing even a single support point would change the margin.
After using Lagrange Multipliers on the previous equation, the cost becomes:

$$\begin{aligned} &\underset{\alpha}{\operatorname{argmax}} \textstyle\sum_n \alpha_n - \frac{1}{2}\sum_{n,m} \alpha_n\alpha_m t_n t_m \mathbf{x}_n^{\mathrm{T}}\mathbf{x}_m \\ &\text{with constraint: } \textstyle\sum_{n=1}^{N} \alpha_n y_n = 0, \quad \alpha_n \geq 0 \end{aligned} \tag{2}$$

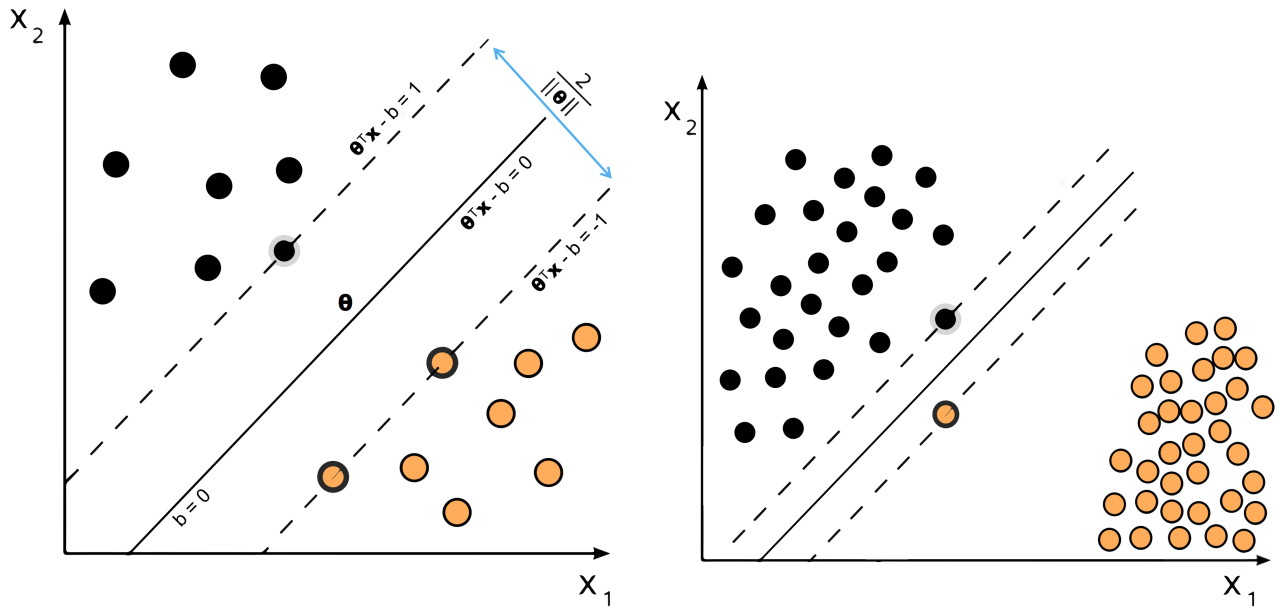
and the prediction formula becomes:

$$y_{\text{ new}} = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_n y_n \mathbf{x}_n^{\mathrm{T}}\mathbf{x}_{\text{new}} + b\right). \tag{3}$$

Soft margin Having a solution that only relies on a few points is not necessarily good as we can see from Figure 4b.
We can turn the hard margin into a soft one by changing the constraint from Eq. 2 into

$$\sum_{n=1}^{N} \alpha_n y_n = 0, \quad 0 \leq \alpha_n \leq C.$$

A high C leaves the situation unchanged, however a small C will cause us to ignore the important variability in the data and lead to equally bad predictions (underfitting).



(a) (b)

Figure 4: *a) Graphical explanation of Support Vector Machines. The data belongs to two classes and it is linearly separable. The distance presented in blue represents the margin, while the three highlighted data points are the support points. b) Graphical explanation of why a hard margin adds bias, although it helps with overfitting. Due to the constraint in the cost function, all the data points need to be on the correct side of the boundary.*

Kernel trick What happens, though, when the data are not linearly separable in the given dimensions? Since the data only appears as a dot product in Equations 2 and 3, we cab swap this product with another one, so that a non-linearly separable problem becomes linearly separable in the original dimensions. This is called the "Kernel Trick" and it substitutes the dot product with "kernel functions", functions that compute what the product would be if we projected the data onto a higher-dimensional space. Re-writing the previous equation returns the following formula:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_{n} \alpha_n - \frac{1}{2} \sum_{n,m} \alpha_n \alpha_m y_n y_m K\left(\mathbf{x}_n, \mathbf{x}_m\right), \tag{4}$$

in which $K(\mathbf{x}_n, \mathbf{x}_m)$ is the dot product of the data in different dimensions, computed by the kernel function K. The most common kernel functions are:

- Polynomial: $K(\mathbf{x}_n, \mathbf{x}_m) = (1 + \mathbf{x}_n^\top \mathbf{x})^d$, where d is the degree of the polinomial;
- Radial: $K\left(\mathbf{x}_n, \mathbf{x}_m\right) = \exp\left\{-\gamma \left(\mathbf{x}_n - \mathbf{x}_m\right)^{\mathrm{T}} \left(\mathbf{x}_n - \mathbf{x}_m\right)\right\}$, where γ is a hyperparameter which controls how complex the decision boundary is.

The kernel trick allows us to solve complex non-linear classifications problems by finding one global minimum, therefore keeping the otherwise computationally-expensive requirements to a minimum. A visual example of a nonlinear classification problem can be found in the Appendix (Fig. 19)

Specifically in the case of the Radial kernel, the hyperparameters C (regularization strength) and γ (inverse kernel width) are related and it is important to come up with a good combination. For both parameters, too high of a value leads to overfitting and too low of a value leads to underfitting. We tuned both C and γ via cross-validation.

Logistic Regression

Logistic Regression (LR), performed by using Generalized Linear Models (GLMs), is a linear classifier; it is composed of a linear combination of explanatory variables and parameters, which is linked to a Binomial probability distribution via the logit function. For this reason, it a probabilistic classifier.

Let us define the design matrix $X = \begin{bmatrix} 1 & \boldsymbol{x}_1 \\ \vdots & \vdots \\ 1 & \boldsymbol{x}_{N_d} \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1N_p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N_d1} & \dots & x_{N_dN_p} \end{bmatrix}$, where N_d is the number of observations and N_p is the number of variables. Then LR models the *probability* of a categorical outcome $y \in \{1, \dots, K\}$, which ranges from 0 to 1 by definition, given a vector of variables $\mathbf{x} = (x_1, \dots, x_{N_p})^{\mathrm{T}}$.
In our case, it models

$$h_\theta(x) = P(y_i = k|\mathbf{x}; \theta) = \frac{e^{\theta_k^T \mathbf{x}}}{\sum_{k=1}^{K} e^{\theta_k^T \mathbf{x}}}, K = 5 \tag{5}$$

where K is the number of classes in our data set, $\boldsymbol{\theta}_i = (\theta_{i1}, \dots, \theta_{N_p})$. This is the so-called "softmax approach", which is incorporated in the *glmnet* package, used in our analysis.
To obtain the parameter estimates, we need to minimize the Cost function of in Eq. 7 with respect to θ, via numerical techniques.

$$J(\theta) = -\sum_{i=1}^{N_d} I \cdot \log h_\theta(\mathbf{x}_i), I = \begin{cases} 1 & \text{for } y_i = k \\ 0 & \text{for } y_i \neq k \end{cases} \tag{6}$$

where $\mathbf{x}_i$ is the i_{th} row of the design matrix.
The output of the model is a vector of the estimated probabilities that $y_i|\mathbf{x}_i = k$, i.e. the probability that one observation belongs to each one of the classes (the metabolic states). In our case, $N_p > N_d$: in minimizing the cost function, the parameters are sent to infinity. Moreover, our goal is to obtain good performance on unseen observations; therefore, the model cannot "overfit" on the training data, due to the bias-variance trade-off. A possible solution is regularization via the L_1 norm or/and the L_2 norm, penalties which are added to the cost function:

- The L_1 norm (LASSO) is the absolute value of the parameter estimates, which are shrunk equally and can be zeroed out due to a discontinuity in the first derivative of the penalty. In our case, it can only select up to N_d parameters; moreover, it can only pick one variable from a group of correlated features.
- The L_2 norm (Ridge) is the square of the parameter estimates and it places a heavier penalty on the larger values. No value is shrunk to exactly zero, therefore Ridge can help when there are more parameters than observations; in the case of highly correlated features, they all tend to enter the model.

As we prefer a sparse solution, but would also like the benefits of Ridge regression, we used the **Elastic net penalty**, which combines the above-mentioned penalties [24]. After incorporating Elastic net in Eq. 7, the problem becomes:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} J(\boldsymbol{\theta}) + \lambda \sum_{k=1}^{K} \left[\alpha |\boldsymbol{\theta}_k| + (1-\alpha)\boldsymbol{\theta}_k^2 \right], \tag{7}$$

where $\lambda > 0$ is the regularization parameter, while $0 \leq \alpha \leq 1$ is the mixing coefficient that controls how the penalties are mixed ; $\alpha = 0$ is equal to using only the LASSO penalty, while

$\alpha = 1$ is equal to performing Ridge regression [9]. We tuned both λ and α via the inner loop of cross-validation. A more detailed description of this method can be found in the Appendix.
We chose Logistic Regression over Linear Discriminant Analysis (LDA), due to its higher flexibility and robustness. LDA focuses on all the observations and it requires the classes to have a similar number of observations, while Logistic Regression is less sensitive to outliers and allows for a varied class distribution. Although some papers put forward the idea that LDA gives better results when its assumptions are met [6], other sources have shown that these performance gains are considerable only in asymptotic cases [20].

PCA

Principal component analysis (PCA) is an unsupervised technique for dimensionality reduction. PCA can be thought as a rotation of the coordinate system, which minimises the sum of the squares between the data (plotted on the original axes) and the first rotated axis, the first principal component (PC1). This new dimension is the one that summarises most of the variance in the data. PCA then identifies other axes, perpendicular the preceding PCs, that summarise most of the remaining variance [4]. The process ends when the number of PCs is equal to the number of variables; however, only the first PCs are kept since they account for most of the variability in the data. It can be used for:

- Data visualization, to display the variance of many features into a 2D plot;
- Data compression, to reduce the computational cost and the storage cost of a data set. For the second reason, PCA can be used as a pre-processing step for Logistic Regression.

A more detailed description of this technique can be found in the Appendix. In our analysis, the number of principal components is tuned in the inner loop of nested cross-validation.

Random Forest

Since GLMs are probabilistic and SVMs are not, the comparison between linear and non-linear algorithms could easily turn into a one between probabilistic and non-probabilistic classifier. Hence, we should also introduce a probabilistic classifier that usually performs well when we are dealing with highly correlated features: Random Forest. The latter has the benefits of returning the importance of every feature and an interpretable solution to the classification problem.
Random Forest(RF) is a trademark for which more information is available at the website in the Bibliography [3]. The basic unit of RF is a Decision tree, an algorithm that finds the feature that best partitions the dataset into homogenous classes. In order to limit overfitting, we introduce hyperparameters (Figure 5): deep trees have indeed low bias but great variance. These parameters should be tuned by using cross-validation.

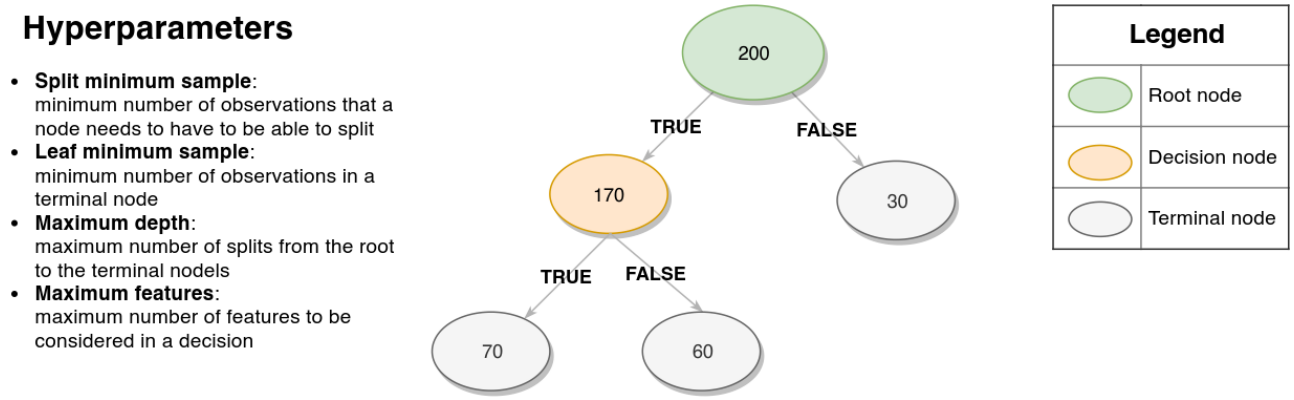


Figure 5: *Graphical explanation of a Decision Tree. By acting on the listed hyperparameters, one can control the properties of the tree and therefore limit overfitting.*

To improve the predictions on unseen data, Leo Breiman and Adele Cutler created a so-called *ensemble* method by using Bootstrap aggregation. The training data is bootstrapped or bagged, i.e. sampled with replacement, and a Decision tree is trained on each sample. This is how a "random forest" of Decision trees is grown. The assignments on the new data are made by making predictions with all the trees in the Random forest (voting): the class with most votes is assigned to the target.
In this dissertation, Random forest was not tuned due to its tendency to overfit, to its mechanics (bootstrapping) and to our need to save computing resources. For confirmation, we conducted an experimental test, not shown in this dissertation due to its secondary importance, which determined that the performance of the tuned model and that of the untuned model were similar. We used 300 trees and set mtry to its default value from the package RandomForest and recommended by the literature, $\sqrt{N_p}$, where N_p is the number of variables in the data set. These parameters were chosen as a large enough number for mtry returns a reliable estimate of feature importance [18] and a large number of trees gives stability to the model ensemble.

Cross-validation

The prediction metric of any model tends to perfection as we increase model complexity. This is because we are predicting the same data on which we trained the model, so the discrepancy between the ground truth and the predicted values tends to zero. Hence, to obtain a better estimate of the prediction metrics, we should consider the *out-of-sample error*,by making the predictions on unseen data. A common strategy is to split the available data into a training and a test set; however, holding out a portion of the data is only possible when a large number of observations is available.
A novel solution is k-fold cross-validation, an iterative resampling algorithm that works by splitting the data into k parts or *folds*; the models are fit to $k-1$ folds, tested on the remaining fold and a metric based on the prediction error (e.g. accuracy) is calculated. The process is repeated for every fold and the metrics are averaged. k is itself a hyperparameter; however, Kohavi [12] showed, in a well-cited paper, that $k = 10$ provides a good balance between bias (lower k and cheaper) and variance (higher k, expensive and with high variability), therefore this value is usually taken as default. Zhang and Yang [23] later demonstrated empirically that the variance keeps decreasing beyond $k = 10$ and reaches a plateau, nevertheless $k = 10$ remains

the preferred option due to the computational resources required to carry out CV.

Nested Cross-Validation

It is important that we choose the right hyperparameter for the regularization of Logistic regression and Support Vector Machines, since they act upon the bias-variance trade-off. Traditionally, for probabilistic algorithms, model selection has been carried out with AIC and BIC, two analytic approximations. For SVM, the standard procedure was to hold out an additional test set, called *validation set*, that could be used for tuning the hyperparameters and then be discarded. Reusing the test set to improve the models would prove inconvenient, as it would lead to optimistically biased out-of-sample estimates; the observations in the test set would cease to be independent. A more modern approach in both cases is to use cross-validation, at the end of which the model with the best average performance is chosen.
Our goal is to compare multiple learning algorithms and it is important that we do not reuse the test data in the comparison. To ensure a fair comparison, we should train and tune two algorithms on the same data partitions; however, once the unknown data has been used for the tuning, it cannot be reused to estimate the out-of-sample performance, as previously stated. It could be arguable that it is possible to compare *raw models*, models with the default (hyper)parameters; however, we deem this inappropriate as we would be comparing a combination of defaults that were subjectively chosen and that would lead to highly variable results. Although usually considered an incorrect stance, some argue that the sole variability in the new data could cause a difference in the performance of the algorithms to be significant. Therefore we will also show the training performance in our analysis; incidentally, this will allow us to diagnose further problems. The approach we will take to isolate the tuning procedure from the comparison is called Nested Cross-validation. As illustrated in Figure 6, Nested CV consists of two rounds on CV: the inner loop performs the tuning of model parameters, whereas the outer loop retrains the tuned models and obtains an averaged metric for the prediction error.

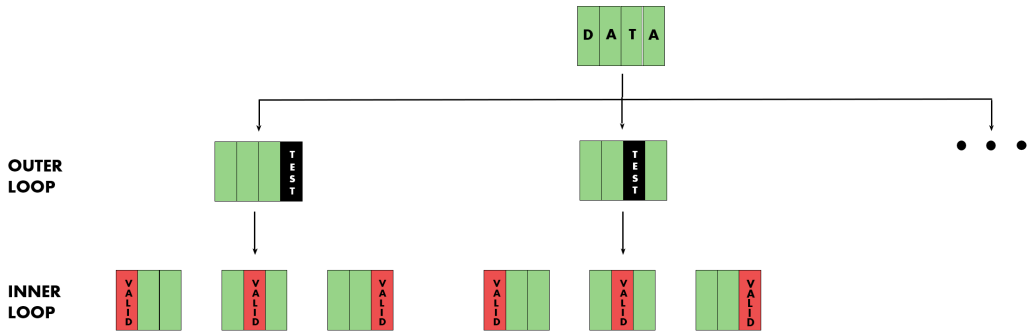


Figure 6: *Graphical explanation of Nested Cross-Validation. The original data set is split into k-folds, one of which is used as a test set and the rest as a training set; this is called the 'outer' loop and it is used to estimate the predictive accuracy. Each training set in the outer loop is split again into k-folds, one of which is the validation set. This is called the 'inner' loop and it is used for tuning.*

Kappa

We chose Kappa as a metric to estimate the predictive performance of the algorithms and compare them to each other. Accuracy, defined as the ratio between the true predictions and the total predictions of the confusion matrix, can be problematic when the classes are unbalanced, which is the case in our data set as we can see from Table 6 in the Appendix. Kappa improves such metric by accounting for the marginal distribution of the target variable; this is done by using both the *Expected accuracy* and the *Calculated accuracy* [13]. Kappa allows us to directly

compare the performance of two algorithms on the same classification task. There are also indicative tables to rate the values.
We should note that many models output a probability, so a problem with these metrics is that they require a threshold to assign a prediction to the positive class; therefore, they have higher variance compared to threshold-independent metrics.
A better choice of metric would have been Brier score, which uses the output model probabilities, but it requires extra storage and additional calculations; this is inconvenient, as it increases the computational cost and it causes problems with parallelization.
A possible fix to the unbalanced classes problem would have been to stratify the resampling so that the proportion of observation in each class was maintained during the training and the testing phases; however, we wanted to avoid making assumptions about the real distribution of metabolic states.

Statistical Hypothesis Testing

After comparing and sorting the algorithms according to their performance, we need to confirm the findings by performing a statistical test. Demsar [5] warns against using t-tests or ANOVA: as the tests requires the assumption of normality, the number of data sets needs to be greater than 30 for the central limit theorem to kick in. When this number is smaller than 30, we can only *assume* that the data is normally distributed; however, we cannot verify this assumption because tests like the Kolmogorov-Smirnov test lack power on small samples.
For this reason, we used Friedman's non-parametric test, which does not require heavy assumptions; it is however less powerful than parametric tests. Friedman's test is rank-based and it ranks the algorithms on the different datasets, independently. Its null hypothesis states that, if the algorithms had equal performance, all the ranks would be equal and the Friedman's statistic would be Chi-squared-distributed when N and k are big enough (as a rule of a thumb, $N > 10$ and $k > 5$). For a smaller number of data sets, exact critical values have been computed [10].
An alternative to the previous tests is a Bayesian hierarchical test, which does not require the assumption of normality and gives more informative results. Taking a Bayesian approach also eliminates the need to account for multiple comparisons, a problem of the frequentist approach. This test uses a vector of differences between the test Kappa of the models, selected by the inner loop of cross-validation, and assumes that such differences come from a Multivariate Normal distribution with the same mean and the same variance-covariance matrix. It samples from the posterior distribution of the differences by using MCMC and then calculates the probability that the differences are positive, irrelevant or negative (by comparing them to a range of practical equivalence). The calculated probabilities mean that one algorithm performed better than another one with respect to one data set. Due to the choice of priors, the hierarchical test allows for "borrowing strength" between the data sets. More details on this test are available from the research conducted by Benavoli et al. [2]. The same priors defined in their article were used in this dissertation. The repository for the coded test can be found in the Bibliography [1].

Chapter 3 - Standard analysis of the data

We established approximately how many iterations of tuning would be necessary to explore the parameter space using a Random Search strategy. Figure 12 in the Appendix shows that we can obtain the same Mean out-of-sample Kappa by reducing the number of tuning iterations and using a 10-fold cross-validation (CV) scheme. CV is what we used subsequently, as it reduces the optimistic bias of the out-of-sample Kappa deriving from using the same data for tuning and metric estimation. For tuning Regularized Logistic Regression and Support Vector Machines with Radial kernel, we chose a conservative number of iterations, 15 and 30, respectively. Reducing the number of Random Search iterations per cross-validation step allowed us to reduce the computational resources required and speed up the training. Arguably, SVMs require more iterations for the tuning due to the presence of more parameters.
A nested 10-fold CV comparison showed very little difference between the Mean Test Kappa of the two tuned models, with respect to each other, on each data set; this can be seen from the aggregated performance shown in Figure 7 and Table 8 in the Appendix. A boxplot of the non-aggregated performance of both algorithms achieved during the outer CV iterations can be found in the Appendix (Figure 13). Although the training Kappa is seemingly equal on all data sets, the test Kappa on the Rhodobacter data set seemed consistently lower. According to Landis and Koch [13], we can consider the obtained Kappa on the Chlorella data set as almost perfect and on the Rhodobacter data set as substantial, albeit we should note that the scale is subjective.

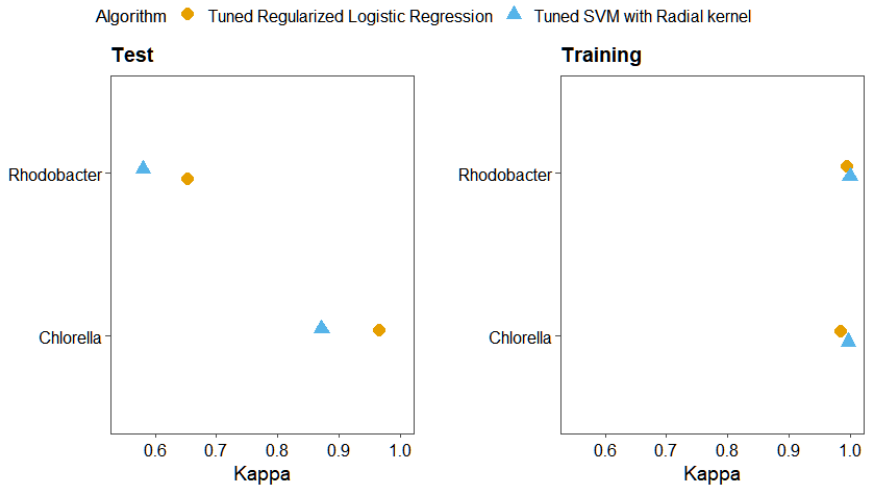


Figure 7: *Mean Test Kappa and mean Training Kappa of Logistic Regression and Radial kernel SVM on the full data sets.*

Friedman's test did not allow us to reject the null hypothesis of a difference between the algorithms.*
By using the Bayesian approach (Table 1), we infered that Regularized Logistic Regression is likely to be better on both data sets.
Diagnostics for this comparison allowed us to verify that we could trust the obtained results. Having already checked for the the difference between the training performance and the test performance of the algorithms in Figure 13 in the Appendix, we noticed some overfitting for the Rhodobacter data set. Comparing the performance of untuned models with the tuned ones resulting from the inner rounds of nested cross-validation showed a clear improvement in the out-of-sample predictive Kappa (Figure 14 in the Appendix).

Table 1: *Results of the Bayesian hierarchical test for the difference between Logistic Regression and Radial kernel SVM. The values express probabilities.*

	Radial SVM is better	Equivalence	GLM is better
Chlorella	0.00375	0.00725	0.98900
Rhodobacter	0.07300	0.05275	0.87425

Figures 15 and 16 in the Appendix are heatmaps that highlights the relation between the hyperparameters that we tuned with the inner CV loop. GLMs performed better with the Ridge penalty and little regularization. SVMs performed better with a high C, whereas the choice of sigma seemed irrelevant.
Expanding the comparison to more algorithms showed improvements on the linear side. Figure 8 is a *summary* of the aggregated test mean Kappa: regularized Logistic regression with PCA and without PCA seem to be better on the the Rhodobacter and the Chlorella data sets, respectively.

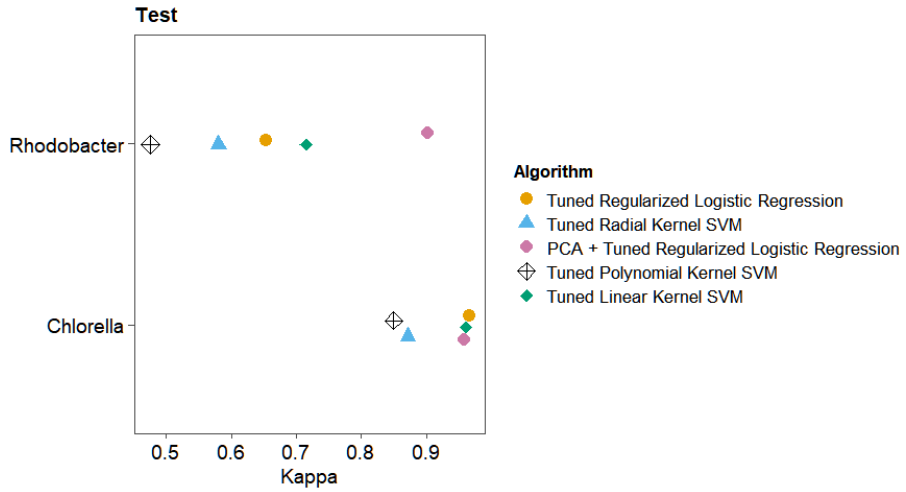


Figure 8: *Comparison between the aggregated out-of-sample Kappa of the tuned SVMs with different kernels and Tuned Regularized Logistic Regression with and without a PCA pre-processing step.*

Tables 10 and 11 in the Appendix confirm that SVM with the Linear kernel is extremely likely to perform better than SVM with the Radial kernel or the Polynomial kernel, on both data sets. Table 2 shows that there is a slightly higher probability that SVM with the Linear kernel performs better on the Chlorella data set compared to regularized Logistic regression. On the Rhodobacter data set, it is clear that using PCA as a pre-processing step before Logistic Regression improves the out-of-sample Kappa considerably.

Table 2: *Results of the Bayesian hierarchical test for the difference between Linear kernel SVM and Logistic Regression with PCA. The values express probabilities.*

	Logistic Regression with PCA is better	Equivalence	Linear SVM is better
Chlorella	0.19975	0.50050	0.29975
Rhodobacter	0.98875	0.00475	0.00650

Finally, Table 3 confirms that Linear kernel SVM is the best-performing classifier on the Chlorella data set.
Logistic regression is a probabilistic classifier, whereas Support Vector Machines is non-probabilistic. Adding Random Forest, a probabilistic non-linear classifier, to the comparison

Table 3: *Results of the Bayesian hierarchical test for the difference between Linear kernel SVM and Logistic Regression. The values express probabilities.*

Logistic Regression is better	Equivalence	Linear SVM is better
0.344	0.34	0.316

helped us dismiss the concern that probabilistic algorithms performed better on this task. Figure 9 shows that Random Forest did not provide an improvement over Support Vector Machines. This is because it overfits the training data (Figure 18 in the Appendix).

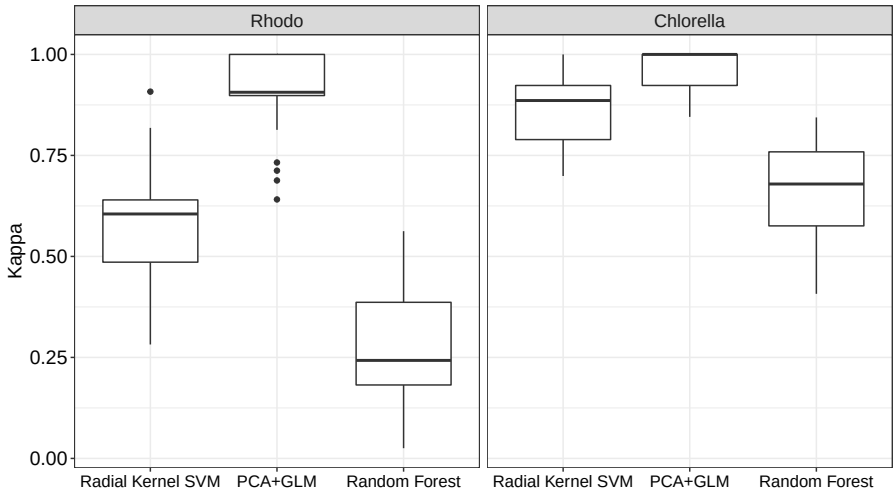


Figure 9: *Boxplot of the mean test Kappa of Random Forest compared to the performance of the worst-performing non-linear algorithm. One of the best-performing linear algorithms is also shown for reference.*

Thanks to Random forest, we were able to plot feature importance. Figure 10 shows that the most important features were those localized in the intensity peaks.

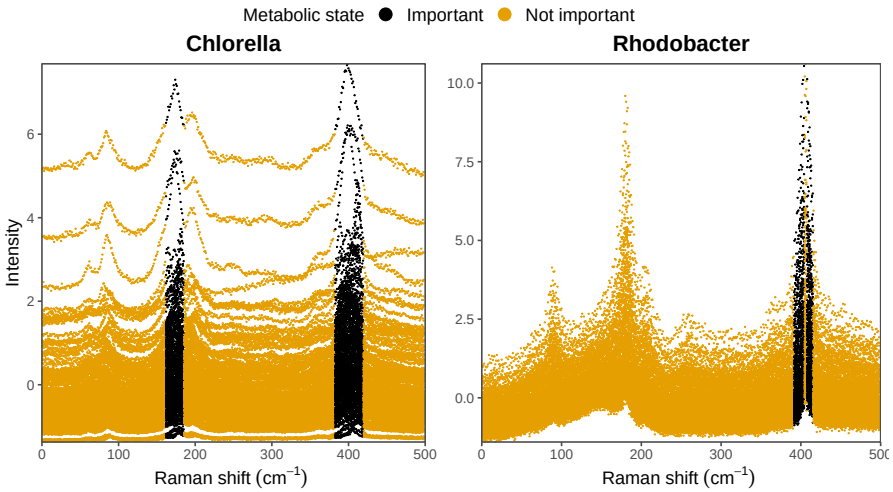


Figure 10: *Raman spectra of the Chlorella and the Rhodobacter data sets. The most important features (for which importance is greater than 0.5) are highlighted in black.*

Based on the importance returned by Random Forest, we selected the band of important features (for which importance was greater than 0.5) that the spectra have in common. Specifically, we

kept the variables from x383 to x419 for Chlorella and from x392 to x415 for Rhodobacter. After repeating the comparison on the new data sets, we obtained the results in Table 4, which show the aggregated Kappa for the training and the testing of Tuned Regularized Logistic Regression and SVMs with Radial kernel before and after variable selection. We noticed an 18% increase of the out-of-sample Kappa for the Rhodobacter data set.

Table 4: *Mean Training Kappa and mean Test Kappa of Logistic Regression and Radial kernel SVM on the Chlorella and Rhodobacter data sets. The metrics were obtained by using both the original and the modified data sets.*

	Mean Train Kappa		Mean Test Kappa	
Algorithm	Full data set	Selected data set	Full data set	Selected data set
Chlorella				
Tuned Radial SVM	0.997	0.996	0.871	0.928
Tuned Regularized GLM	0.984	0.990	0.966	0.967
Rhodobacter				
Tuned Radial SVM	0.999	0.982	0.580	0.828
Tuned Regularized GLM	0.995	0.984	0.653	0.838

Table 5: *Mean test Kappa of all algorithms for both comparisons. The algorithms highlighted in red have the best test performance on the data sets with selected variables*

	Mean Test Kappa	
Algorithm	498 variables	Selected variables
Chlorella		
Regularized Logistic Regression (tuned)	**0.966**	**0.967**
PCA + Regularized Logistic Regression (tuned)	0.959	0.961
PCA + Linear Kernel SVM (tuned)	NA	0.951
Linear Kernel SVM (tuned)	0.962	0.948
Radial Kernel SVM (tuned)	0.871	0.928
Polynomial Kernel SVM (tuned)	0.850	0.892
Random Forest	0.655	0.767
Rhodobacter		
PCA + Regularized Logistic Regression (tuned)	**0.902**	**0.893**
Linear Kernel SVM (tuned)	0.715	0.880
PCA + Linear Kernel SVM (tuned)	NA	0.858
Regularized Logistic Regression (tuned)	0.653	0.838
Radial Kernel SVM (tuned)	0.580	0.828
Polynomial Kernel SVM (tuned)	0.476	0.816
Random Forest	0.271	0.438

Table 5 presents the out-of-sample performance of all the algorithms. The test performance improved considerably on the data sets with the selected variables.
On the Rhodobacter data set, Regularized Logistic Regression coupled with the PCA pre-processing step was the best-performing algorithm in both comparisons. On the Chlorella data set, Regularized Logistic Regression was the best-performing algorithm. Applying PCA to Logistic Regression seemed detrimental, while applying it to Linear kernel SVM lead to a minimal improvement, that could even be nonsignificant.
Bayesian tests between the first rank and the second rank of the comparison with the selected variables are in Figures 12 and 13 in the Appendix, while Friedman's test again leads to non-significant results

Discussion

Linear algorithms had better performance

In all the comparisons we performed, Friedman's non-parametric hypothesis test consistently failed to show a statistically significant difference between the algorithms, possibly due to its low power. It is clear that a Bayesian approach is more powerful, while also giving practically-significant result; this is thanks to the definition of a range of practically insignificant differences, which embedded in this hierarchical test.
From the results on the full data set, it is clear that SVM with a non-linear kernel and Random Forest were not able to achieve the same predictive Kappa as their linear counterparts. Linear algorithms always returned better results, even when we do not take into account the use of PCA, and we found the same pattern after restricting the data to specific bands of the spectra. Random Forest proved to be the worst-performing algorithm; this suggests that the results are independent of Logistic Regression being probabilistic and SVM being non-probabilistic. We can, therefore, assume that linear classifiers give better predictions for the Raman spectra of these two strains. Broadening the comparison to more algorithms, both linear and non-linear, will provide more evidence towards this hypotheses.

Why did SVM with the Radial kernel perform so poorly?

The following insights are based on Figures 14 and 16 in the Appendix. We should remember that the latter was created by using the maximum achieved out-of-sample performance as a reference. When we look at the untuned performances of Radial kernel SVM, which is a Gaussian transformation, we notice that they are very poor. By default, sigma, the width parameter of the Radial kernel, is estimated based on the data and C, the regularization parameter that controls the margin, is set to 1. If we let these hyperparameters vary freely, we notice that SVM with the Radial kernel requires a very small sigma on both the Chlorella and the Rhodobacter data sets: this implies that the classifier is local and it is probably the cause of some overfitting. Nevertheless, larger values of sigma result in even poorer performance, which is the reason why they were not selected by the cross-validation. To counteract the overfitting, the tuning increases the value of C, which in turns softens the margin. Taken together, these considerations suggest that the Radial kernel is a poor choice for this type of data. We can extend this conclusion to the Polynomial kernel, in the light of its poor predictive performance, and to other non-linear algorithms, although this is a hypothesis that we will need to properly test in future works.
For Logistic regression, the defaults are set to be $\alpha = 0$, which is the Lasso penalty, and λ is chosen automatically. When we let the hyperparameters change freely, we can see that the penalty type makes no difference and λ is very small. This and the automatic tuning by default are probably the reasons why the boxplots for tuned and untuned Logistic regression overlap on both the Chlorella and the Rhodo data sets. The same figures show that, on the Rhodobacter data set, Logistic regression could achieve the maximum recorded performance with the same set of hyperparameters that were chosen for the Chlorella data set. In light of the better results with PCA, this suggests that the algorithm was overfitting.
We can, therefore, assume that the lower performance of all algorithms on Rhodobacter data, with respect to the Chlorella results, might be due to overfitting and not due to non-linearity, as we initially thought. This is further confirmed by the gap between the training Kappa and the test Kappa of Logistic Regression and SVM with the Radial kernel, shown in Figure 13 in the Appendix. The same gap was present in the diagnostic plot of Random Forest (Figure 18 in the Appendix), although we purposefully kept the recommended model parameters instead of

subjecting the model to tuning: this is likely to be the reason why the test metric of Random Forest is the worst of the comparison. It is important to note that the Rhodobacter data set has about 20% fewer observations than the Chlorella data set. The out-of-sample metric could be improved by collecting more data, as the variance in the predictions would decrease; we believe that, in such instance, Logistic regression would still be a better classifier, as the Radial SVM algorithm is biased.
There is yet another cause of overfitting, which does not affect the comparison since it is a methodological choice. After the inner loop of cross-validation, we chose the model that returned the best average Kappa; indeed, one of the assumptions in Nested CV is that all the models selected by the inner loop of CV are equally good. Ng [16] proposed that a better way to proceed would be to select a model with a higher cross-validation error by using a computationally-efficient flavour of leave-one-out cross-validation. We considered this approach but found that we could not afford the additional computational effort due to our limited resources. An alternative way to reduce overfitting in the outer loop of CV would be to decrease the number of tuning iterations so that the exploration of the parameter space does not return the best values and therefore we do not obtain the best models during tuning. However, this procedure would expose us to some criticism, since the amount of variance left in the models of the inner loop of CV is a subjective choice. Luckily, the PCA pre-processing step allowed us to keep computation to a minimum while significantly improving the out-of-sample Kappa.

How did feature selection improve the global performance?

The only important features that the Rhodobacter data set had in common with the Chlorella data set are the ones centred around $400cm^{-1}$. Although Figure 10 suggests that the most important variables(for which importance is greater than 0.5) are the ones that correspond to the highest peaks in the Raman spectra, we shall provide a different interpretation. We propose, based on 1, that the most important variables are the ones associated with a specific category of molecules, whose production is increased or decreased by the cell under particular environmental conditions and, therefore, metabolic states. This translates into spectral regions where there is a clear separation between the metabolic states. What corroborates our hypothesis is that the classes are always visually separable on the Chlorella data set, but they become so only in the second intensity peak on the Rhodobacter data set. For this reason, the predictions on the selected Chlorella data set stayed constant or degraded for linear algorithms but improved for non-linear models; they always improved when we trained the algorithms on the selected variables of the Rhodobacter spectrum. An exception to the global trend was the performance of the top-ranking algorithms, which slightly degraded for the Rhodobacter data set and stayed constant for the Chlorella data set. This suggests that, while variable selection could help tackle overfitting, it is detrimental when a model is not overfitting, as suggested in 9 in the Appendix.

Why did PCA work and what are its shortcomings?

PCA uses the linear relationships between the variables to construct new features. As it aims at summarizing most of the variability in the first few components, features that give little contribution to the variability but are key elements in the classification problems are not summarised in the first components and often discarded. This statement is likely related to the fact that in Figure 3, clustering seems more feasible on PC2 (0.35% of the variance) than on PC1. For this reason, training the algorithms on PCA data after removing Principal Component 1 still allowed us to make predictions with good performance, although we do not show this result as it is secondary. There is one more fact that validates our statement: when we consider that retaining 98% of the variability was not enough to obtain good predictions, as shown in Figure

17 in the Appendix, it becomes clear that relatively smaller differences in the data carry a lot of information. In our data sets, this information is likely to be located in the region which contains the selected variables; indeed, even though any band could allow for some separation of the metabolic states, it is evident that some bands contain more discriminatory elements than others. On the Rhodobacter data set, PCA improved the performance of Logistic Regression by 35% and it proved to be the best-performing algorithm in both the original comparison and the one with selected variables. As the difference between these instances is minimal and the performance degraded considerably after removing the PCA step, we can conclude that PCA is necessary for improving the predictive performance of Logistic Regression on the Rhodobacter data set. However, we cannot extend this statement to all the linear algorithms, since PCA degrades the Kappa of SVM with the Linear kernel, the other linear algorithm, which was the second best-performing classifier for Rhodobacter data. On the Chlorella data set, Regularized Logistic Regression was the best-performing algorithm. Applying PCA to Logistic Regression seemed detrimental while applying it to Linear kernel SVM lead to a minimal improvement, that could even be nonsignificant. We can, therefore, conclude that, in the absence of computational constraints, retaining the full spectrum would provide slightly better performance.

Are the results generalizable?

As the data only come from a single strain and it is, therefore, non-representative of the whole population, we cannot extend the results of our comparison to the respective genera. The results only become important when we consider the industrial applications of these microorganisms since engineered processes use specific strains, which are selected for their properties and maintained in culture collections. Thanks to the large sample, we can extend our results to the specific strains of Chlorella and Rhodobacter that were used to obtain the spectra and conclude that our pipeline could be used to effectively classify the metabolic state of these microorganisms.

Further improvements

In the future, we could try using other transformations for the data to improve the test Kappa on the Rhodobacter data set and, more generally, for hardly separable Raman spectra. We could also extend the comparison to other algorithms; this would allow us to identify the algorithm which is most robust to noise. As previously mentioned, any band of the spectrum allows for some separation amongst the classes. Therefore, if we removed the selected variables from the full spectrum, we could determine how many discriminatory elements are left in the data. Moreover, performing the analysis by considering only two metabolic states at a time would allow us to identify separable regions and therefore compounds, whose production gets boosted or turned off during the switch from one state to another. Finally, as pointed out by Rosch et al. [19], we should also investigate whether the Raman spectra vary when the laser focus shifts locally while capturing the Raman scatter of a single cell. We should also investigate if there are variations in the recorded data when there are other cells behind a specimen.

Appendix

Data

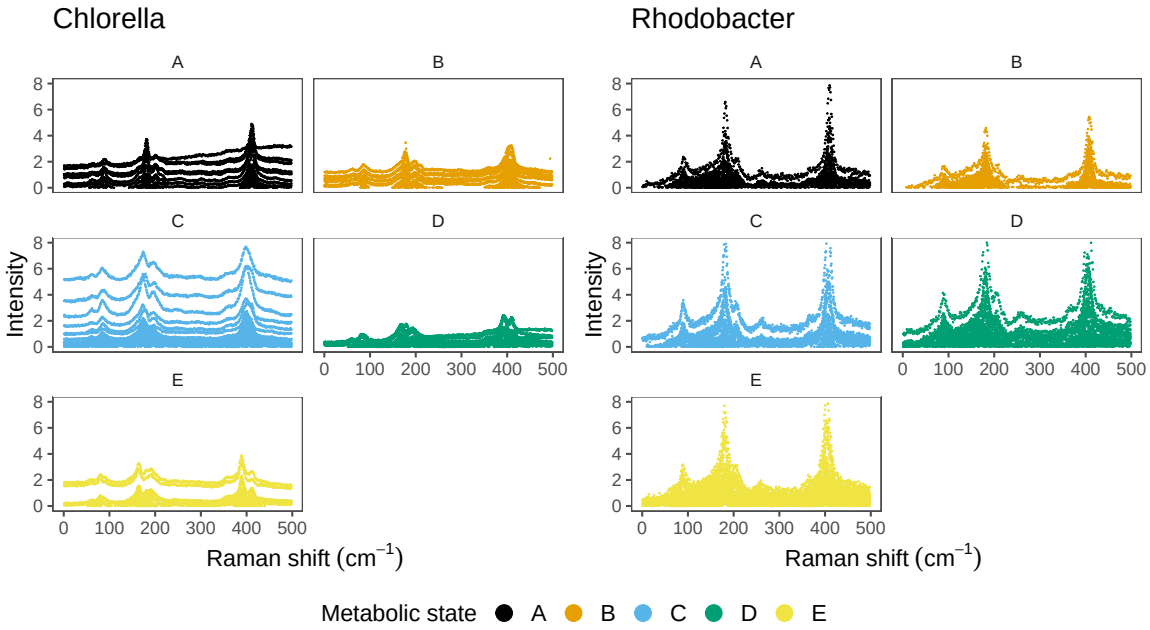


Figure 11: *Discretized Raman spectra of the Chlorella and the Rhodobacter strains plotted by metabolic state.*

Table 6: *Frequency of the observations in both data sets with respect to each metabolic state.*

	Data Sets	
Class	Chlorella	Rhodobacter
A	0.164	0.165
B	0.175	0.187
C	0.187	0.245
D	0.246	0.194
E	0.228	0.209

Table 7: *PCA summary of the Chlorella and Rhodobacter data sets.*

Estimate	PC1	PC2
Chlorella		
Standard deviation	22.09	2.001
Proportion of Variance	0.98	0.008
Cumulative Proportion	0.98	0.988
Rhodobacter		
Standard deviation	22.043	1.257
Proportion of Variance	0.976	0.003
Cumulative Proportion	0.976	0.979

Tuning iterations

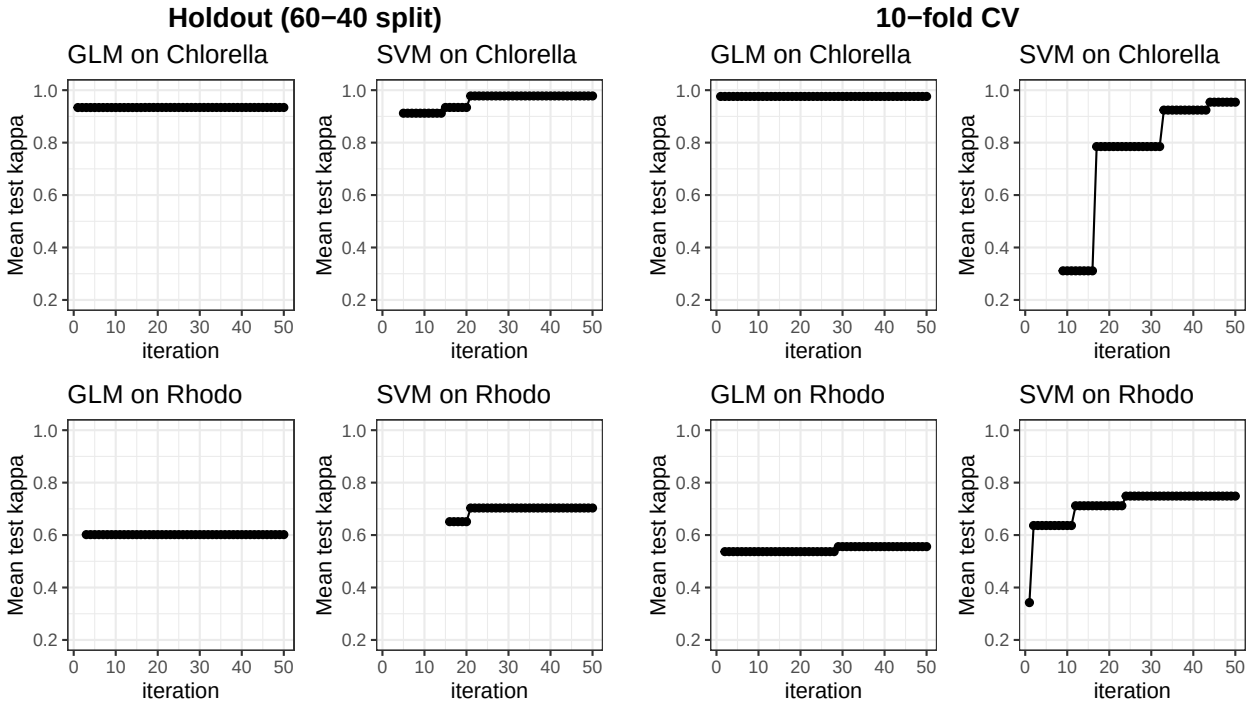


Figure 12: *Scatterplot of the Mean Test Kappa and the iterations of tuning. A comparison was made between different resampling strategies.*

Comparison with the full data sets

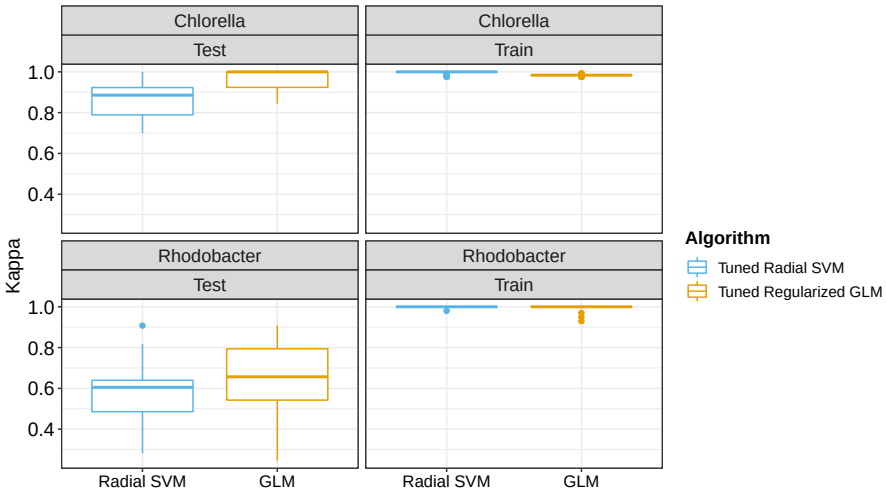


Figure 13: *Boxplot of the non-aggregated performance achieved by Tuned Regularized Logistic regression and Tuned Support Vector Machines with Radial Kernel on both the Test set and the Training set during 10 cross-validation iteration.*

Table 8: *Mean Training Kappa and mean Test kappa of Logistic Regression and Radial kernel SVM on the full data sets.*

Algorithm	Mean Test Kappa	Mean Training Kappa
Chlorella		
Regularized Logistic Regression (tuned)	0.966	0.984
Radial Kernel SVM (tuned)	0.871	0.997
Rhodobacter		
Regularized Logistic Regression (tuned)	0.653	0.995
Radial Kernel SVM (tuned)	0.580	0.999

Table 9: *Mean Training Kappa and mean Test Kappa of the best-performing algorithms.*

Dataset	Algorithm	Train Kappa	Test Kappa
Chlorella	Logistic Regression (tuned)	0.984	0.966
Rhodobacter	PCA + Logistic Regression (tuned)	0.974	0.902

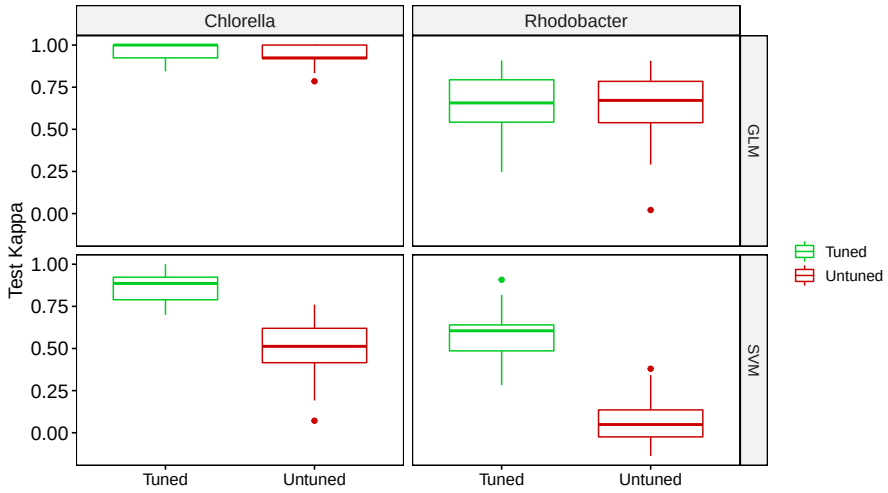


Figure 14: *Boxplot comparing the non-aggregated test performance achieved by tuned Regularized Logistic regression models and Support Vector Machines with Radial Kernel and their untuned counterparts. The plot is faceted by the data set. The testing is done with three repetitions of ten-fold cross-validation and the tuned models are chosen by estimating their out-of-sample performance through an inner loop of ten-fold cross-validation.*

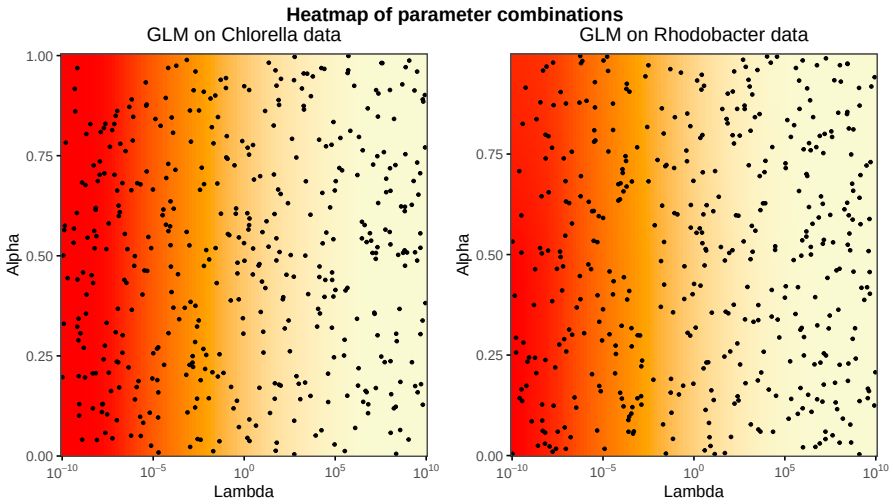


Figure 15: *Heatmap of the out-of-sample Kappa of Logistic Regression on the full data sets. The darker red represents the maximum performance achieved during cross-validation. The dots represent the combinations of hyperparameters tried during the Random Search.*

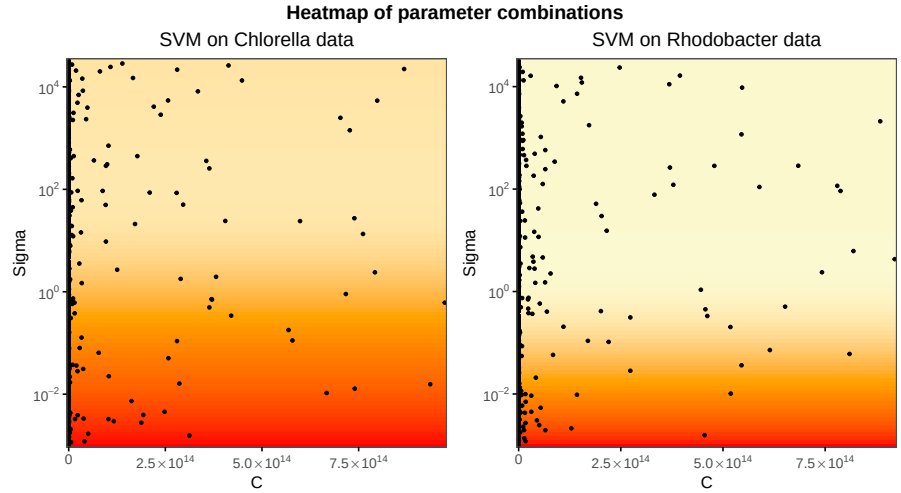


Figure 16: *Heatmap of the out-of-sample Kappa of Radial Kernel SVM on the full data sets. The darker red represents the maximum performance achieved during cross-validation. The dots represent the combinations of hyperparameters tried during the Random Search.*

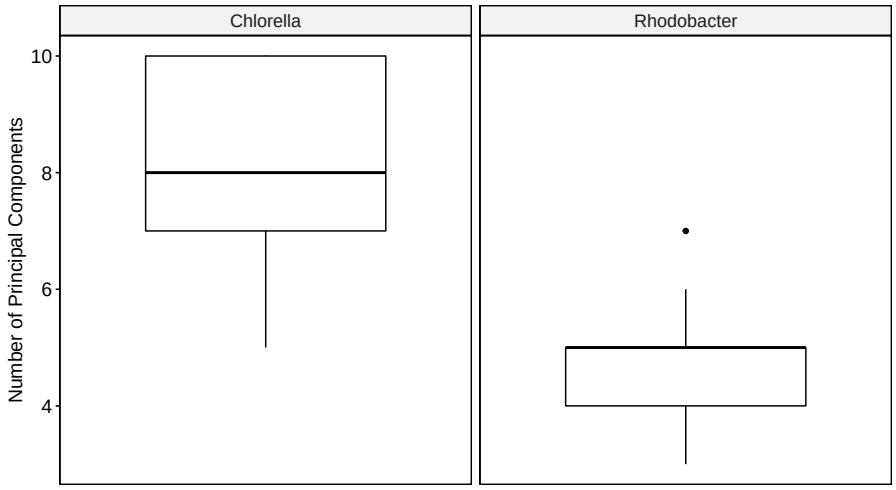


Figure 17: *Boxplots of the number of Principal components in the best models of PCA + Regularized Logistic Regression from the inner loop of cross-validation. The selection criterion for these models is the best out-of-sample performance in the inner loop.*

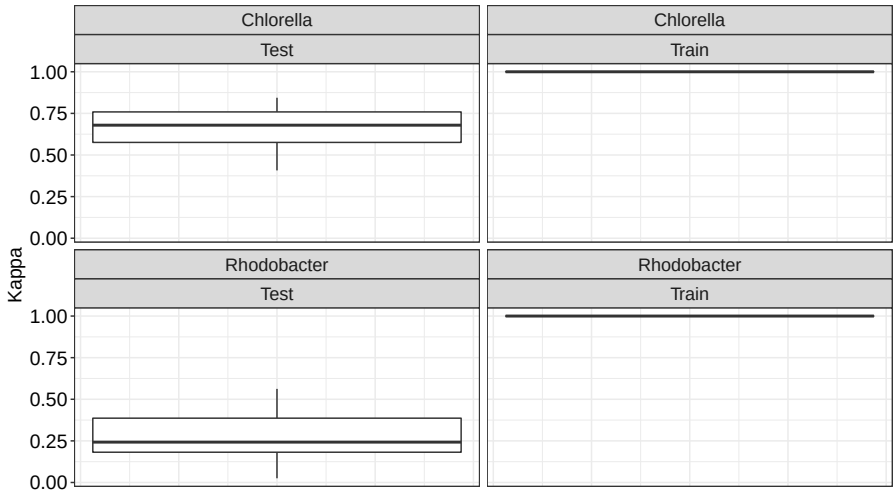


Figure 18: *Boxplot showing the comparison between the mean training performance and the mean test performance of Random Forest, for both data sets. Random Forest overfits on the training data.*

Bayesian hierarchical tests for the full data sets

Table 10: *Results of the Bayesian hierarchical test for the difference between Linear SVM and Radial SVM. The values express probabilities.*

	Radial.SVM.is.better	Equivalence	Linear.SVM.is.better
Chlorella	0.00900	0.01550	0.9755
Rhodobacter	0.00275	0.00325	0.9940

Table 11: *Results of the Bayesian hierarchical test for the difference between Linear SVM and Polynomial SVM. The values express probabilities.*

	Polynomial.SVM.is.better	Equivalence	Linear.SVM.is.better
Chlorella	0.00375	0.0075	0.98875
Rhodobacter	0.00025	0.0000	0.99975

Table 12: *Posterior probabilities that the difference between Kappa (Regularized GLM with PCA - Regularized GLM) is positive or negative for the Chlorella data set. This is equivalent to saying that one algorithm performed better than another one with respect to one data set. The data set used only contains data from a selected band of the spectrum.*

Regularized GLM is better	Equivalence	Regularized GLM with PCA
0.36575	0.35925	0.275

Table 13: *Posterior probabilities that the difference between Kappa (Regularized GLM with PCA - Linear Kernel SVM) is positive or negative for the Rhodobacter data set. This is equivalent to saying that one algorithm performed better than another one with respect to one data set. The data set used only contains data from a selected band of the spectrum.*

Linear Kernel SVM is better	Equivalence	Regularized GLM with PCA
0.2585	0.2325	0.509

Bayesian hierarchical tests for the selected data sets

Methods

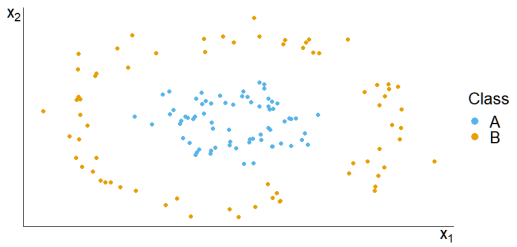


Figure 19: *Non-linear classification problem, solvable by using SVMs with a Radial kernel. The different colors represent the classes.*

Logistic Regression

Similarly to SVMs, Logistic regression estimates a linear decision boundary; however, this is done by using probabilities. To describe how it works, let us define X as the design matrix of our data set, with dimensions $N_d \times N_p$, which are the number of observations and the number of predictors, respectively. y is our target categorical variable, to which we wish to assign labels. Let us also assume that our factor is binary, that is

$$y \in \begin{pmatrix} 0, \text{for the positive class,} \\ 1, \text{for the negative class} \end{pmatrix}.$$

A GLM is made of three parts:

- a systematic component, the linear combination of predictors and parameters $\theta^{\mathrm{T}}x$, where x is a vector representing a single training example and θ is a vector of parameters;
- a random component, which defines the distributions of the outcomes, which has the general formula $f(\mathbf{y};\theta) = \exp[(a(y)b(\theta) + c(\theta) + d(y)]$;
- a link function, so that $\mathbb{E}(Y) = g^{-1}(\theta^{\mathrm{T}}x)$.

GLMs allow to model outcomes that come from a distribution that belongs to the exponential family of distributions, which is a subset of probability distributions that are related to the Normal. They are linear classifiers because the variables and the parameters are combined linearly; however, the output is transformed by the link function, which is the sigmoid function in the case of logistic regression (Figure 20). The formula therefore becomes: $\mathbb{E}(Y) = \frac{1}{1+e^{-\theta^{\mathrm{T}}x}}$.

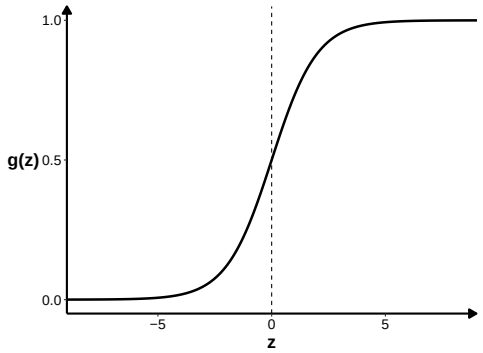


Figure 20: *Logistic function.*

In order to make predictions, logistic regression requires the additional definition of a threshold between 0 and 1: the output of the model is indeed a probability.
The following is an example:

$$\mathbb{E}(Y) = g^{-1}\left(\theta^T x\right) = g^{-1}\left(\theta_0 + \theta_2 x + ...\right) \geq 0.5$$

Therefore,

$$\begin{aligned} &\text{for } \theta^{\mathrm{T}}x \geqslant 0 \longrightarrow \hat{y} = 1 \\ &\text{for } \theta^{\mathrm{T}}x < 0 \longrightarrow \hat{y} = 0 \end{aligned}$$

and

$$\theta_1 x_1 + \theta_2 x_2 \geq -\theta_0.$$

As we can see, the decision boundary is a property of the systematic component and the input of the link function could even be a non-linear function that fits the data.
In order to choose the optimal values for the parameters, the cost is defined as the negative log-likelihood function:

$$J(\theta) = -\frac{1}{N_d}\sum_{i=1}^{N_d}\left[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right] \tag{8}$$

where

$$\mathrm{h}_\theta(x) = g^{-1}\left(\theta^T\mathbf{x}\right) = \frac{1}{1+e^{-\theta^T\mathbf{x}}} = P(Y=1|\mathbf{x};\theta).$$

Figure 21 shows that the cost is zero when the prediction is correct and it approaches infinity when the prediction is wrong.

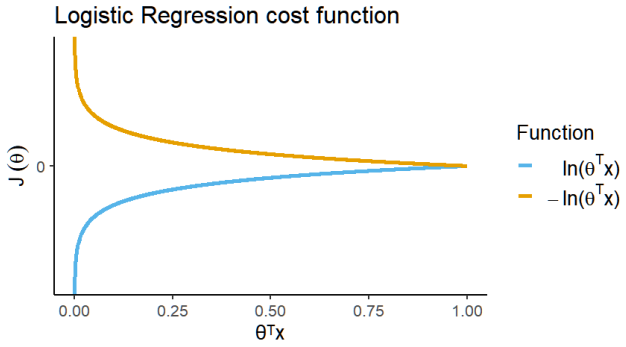


Figure 21: *Logistic regression cost function.*

To extend these results to multiple classes, we only need to use a proportion (used in the package "glmnet"):

$$P(Y = k|\mathbf{x}; \theta) = \frac{e^{-\theta_k^T \mathbf{x}}}{\sum_{k=1}^{K} e^{-\theta^T \mathbf{x}}},$$

where K is the number of classes.
Minimizing the cost with respect to θ by using a numerical method leads to an estimate of the parameters. It is important to remember that we want to predict $\mathbf{y}$ for unforeseen $\mathbf{x}$ and if the model overfits the training data set, it will result in predictions with high variance. We are indeed interested in minimizing the cost function of the *test* dataset. There are different possibilities to address overfitting and obtain the best-predictions according to the bias-variance trade-off. The one we will be focusing on is regularizing the function, by retaining all the covariates but reducing (or zeroing out) the magnitude of the associated parameters θ_i.

Regularization Regularization consists of putting a penalty on the model parameters, that increases the more they grow large. Minimizing a regularized cost function means that some parameters will approach zero during the procedure; this is done so that only the most important variance is modelled, whereas the parameters that approximate the randomness in the data are shrunk and do not contribute to the predictions. In order to choose which parameter to shrink, we can use different penalty functions. LASSO regularization uses the L1 norm as a penalty function. The cost function needs to be minimized given $\sum_{j=1}^{n_P} |\theta_j| \leq$ Quota. Ridge regularization uses the L2 norm as a penalty function. The cost function needs to be minimized given $\sum_{j=1}^{n_P} \theta_j^2 \leq$ Quota. To better explain how the minimization works, we shall examine the following image. It was produced assuming there are only two parameters.

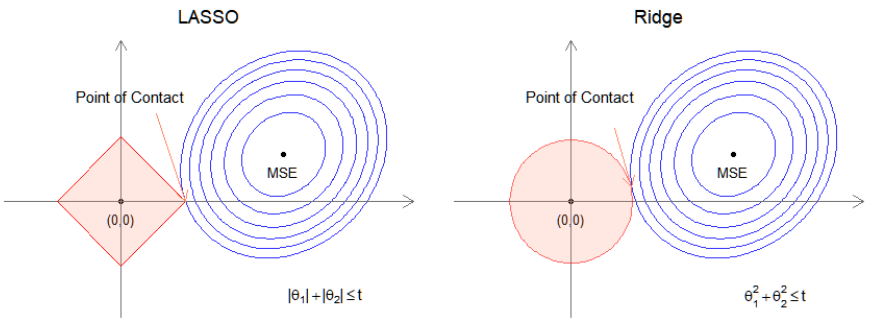


Figure 22: *Contour plot of the Cost function, in blue. The shapes in red contain the area of the L1 and L2 norms, respectively.*

The "MSE" in Figure 22 is the loss function, which is represented by an ellipse due to its formula, while the shapes in red delimit the area defined by the penalties (e.g. $a^2 + b^2 \leq c$ is a circle). The shape of the ellipse is determined by the observed data; however, its size is not fixed since the parameter values can vary. The image presents a contour plot, a way to represent a 3D function on a 2D plane; it is a series of ellipses, in this case. Changing the parameter values results in a smaller area delimited by the ellipses and, as we see in Figure 22, there are multiple combinations of parameters that lead to the same area. Without penalties, finding the parameters' values that minimize the cost function consists of searching the parameter space for those values that produce the smallest ellipsis, i.e. the smallest cost function. The introduction of a constraint, the area delimited by the red shape, forces the sum of the parameters to stay under a certain quota, therefore producing a biased estimate. However, it can be proven that the prediction error is $\text{MSE}(\widehat{\theta}) = \text{Var}(\widehat{\theta}) + \text{Bias}(\widehat{\theta}, \theta)^2$; therefore, accepting some bias will allow us to reduce the variance in our predictions. The smallest area of the ellipses achievable under the red constraint is given by the combination of parameters at the intersection between the contour lines and the constraint. In other words, the cost function is minimized when the optimization algorithm finds a tangent point (we do not use the closed solution, when it exists, due to the computational resources required). The main difference between LASSO and Ridge regression is that decreasing the quota, hence the sum of the parameters, will force some parameters in LASSO to be equal to zero and some parameters in Ridge to be close to zero. Mathematically, the optimization happens by finding the parameter values for which the first derivative with respect to the parameters is zero for both the penalty and the cost function. The reason why LASSO regularization can zero out parameters lies in the fact that the absolute value in the L_1 norm has a discontinuity in which the first derivative is undefined.
Plotting the L_1 and L_2 norms in the same graphs (Figure 23) makes it clear that Ridge regression will penalize large coefficients in multiparameter regression and leave the others relatively similar.

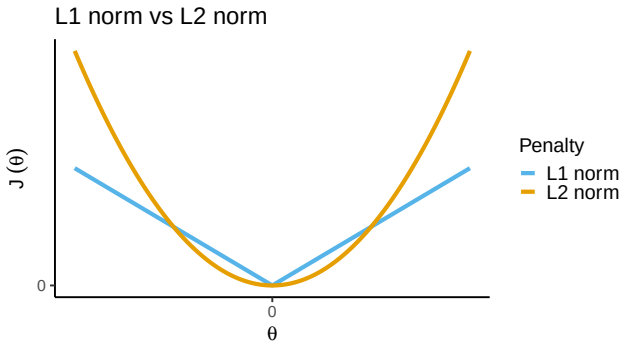


Figure 23: *Comparison between the L1 penalty function, in blue, and the L2 penalty function, in red.*

LASSO has the following characteristics: it performs continuous shrinkage and it is able to feature selection by shrinking parameters to zero, however it is not able to perform grouped selection, therefore only one amongst many correlated variables is selected; when $N_p > N_d$, where the former is the number of data points and the latter is the number of parameters, only a maximum of N_d variables are selected. Ridge regression, instead, is able to handle collinearity and linear dependencies much better: if one variable amongst a group of highly correlated ones is selected, the whole group enters the model. One additional benefit is that it works well when $N_p > N_d$; however, the model is not parsimonious, since the parameters are shrunk but never to zero.
The type of regularization we will be using for logistic regression is Elastic Net. It is primarily used when $N_p > N_d$ and all the variables are highly correlated. The formula for the elastic net cost is:

$$J_{elastic}(\theta) = \sum_{i=1}^{N_d} \left(y_i - \mathbf{x}_i\theta^T\right) + \lambda_1 \sum_{j=1}^{n_P} |\theta_j| + \lambda_2 \sum_{j=1}^{n_P} \theta_j^2,$$

where λ is a hyperparameter inversely proportional to the quota. As it merges both the L_1 and L_2 penalties, it can perform simultaneous parameter shrinkage and variable selection; like Ridge regression, it can perform grouped selection and similarly to LASSO regression, it can zero out parameters. As there is no closed solution, numerical techniques are needed. It is important to mention that it can suffer from double shrinkage.
Applying the Elastic Net penalty to Equation 8 results in the following regularized cost function:

$$J(\theta) = -\frac{1}{N_d} \sum_{i=1}^{N_d} \left[y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] + \lambda \sum_{k=1}^{K} \sum_{j=1}^{N_p} \alpha \left\|\theta_{kj}\right\| + (1-\alpha)\theta_{kj}^2, \tag{9}$$

in which α is the mixing coefficient and it controls how the norms mix.

PCA

Performing PCA involves five steps:

1. Standardizing the covariates by doing $x_{\text{scaled}} = \dfrac{\text{value} - \text{mean}}{\text{standard deviation}}$; this is because large differences in the ranges of the variables would cause the ones with higher ranges to dominate the results;
2. Defining a strategy so that we can perform PCA on the training sets and test sets independently;
3. Computing the Variance-Covariance matrix, to define how the means of the variables vary with respect to each other. We compute the covariance specifically because the co-

variates are sometimes highly correlated. Let x_1 and x_2 be the covariates, then $A = \begin{bmatrix} \text{Var}(x_1) & \text{Cov}(x_1, x_2) \\ \text{Cov}(x_1, x_2) & \text{Var}(x_2) \end{bmatrix}$ A is orthonormal, so $A^{\text{T}}A = I$

4. Computing the eigenvectors and the eigenvalues of A to find the principal components. This is equivalent to finding $Z = XA$, where X is the design matrix. Principal components are new variables created by combining the original features linearly, so that they are all uncorrelated with each other and that the first few components summarize most of the variability in the data. For N_p dimensions, there are N_p principal components; however, PCA allows us to discard most of them. Now for a geometrical interpretation. We centred the data with respect to the origin. For N_p dimensions, we have N_p eigenvalues and eigenvectors. The principal component is a line that goes through the origin onto which we project the data, the eigenvalues are the sum of the squared distances of the projections from the origin (the amount of variance summarized by the PC), and the eigenvector is a scaled one-unit vector that indicates the direction of the line. Once we have the first PC, we find the one-unit vector that is perpendicular to the first one and calculate the sum of squared distances of the projections onto the new line. Rotating the principal components so that the first is horizontal and using the projections as coordinates in an x, y graph gives us a PCA plot. Note that the axes are not equivalent and that the first PC accounts for most of the total variability around the PCs. To find the third PC, we find the best fitting line that goes through the origin and is perpendicular to both the first and the second PC. The process is usually stopped by using a threshold of summarized variability. This is done using a simple proportion between the eigenvalues of the principal component and the sum of all the eigenvalues.

Bibliography

[1] Alessio Benavoli. *BayesianTestsML*. 2017. URL: https://github.com/BayesianTestsML/tutorial (visited on 08/24/2019).

[2] Alessio Benavoli et al. "Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis". In: *Journal of Machine Learning Research* 18 (2017), pp. 1–36. ISSN: 15337928.

[3] Leo Breiman. *Random Forest*. URL: https://www.stat.berkeley.edu/%7B~%7Dbreiman/RandomForests/cc%7B%5C_%7Dhome.htm (visited on 08/24/2019).

[4] N A Campbell and William R Atchley. "The Geometry of Canonical Variate Analysis". In: *Systematic Zoology* 30.3 (1981), pp. 268–280. ISSN: 00397989. URL: http://www.jstor.org/stable/2413249.

[5] Janez Demsar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *Journal of Machine Learning Research* 7 (2006), pp. 1–30. URL: http://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf.

[6] Bradley Efron. "The Efficiency of Logistic Regression Compared to Normal Discriminant Analysis". In: *Journal of the American Statistical Association* 70.352 (1975), pp. 892–898. ISSN: 01621459. URL: http://www.jstor.org/stable/2285453.

[7] Jürgen W. Einax. *M.J. Adams: Chemometrics in Analytical Spectroscopy, 2nd edition*. Vol. 382. 4. 2005, pp. 861–862. DOI: 10.1007/s00216-005-3181-9.

[8] Rekha Gautam et al. “Review of multidimensional data processing approaches for Raman and infrared spectroscopy”. In: *EPJ Techniques and Instrumentation* 2.1 (2015). ISSN: 2195-7045. DOI: 10.1140/epjti/s40485-015-0018-6. URL: http://dx.doi.org/10.1140/epjti/s40485-015-0018-6%20https://epjtechniquesandinstrumentation.springeropen.com/track/pdf/10.1140/epjti/s40485-015-0018-6.

[9] Heikki Huttunen, Tapio Manninen, and Jussi Tohka Tohka, Jukka-pekka Kauppi. “Mind reading with regularized multinomial logistic regression”. In: *Machine Vision and Applications* 24.6 (2013), pp. 1311–1325. DOI: https://doi.org/10.1007/s00138-012-0464-y.

[10] Karen Kafadar and David J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Vol. 51. 4. Chapman and Hall/CRC, 1997, p. 374. DOI: 10.2307/2685909.

[11] Gökhan Kars et al. “Improved hydrogen production by uptake hydrogenase deficient mutant strain of Rhodobacter sphaeroides O.U.001”. In: *International Journal of Hydrogen Energy* 33.12 (2008), pp. 3056–3060. ISSN: 03603199. DOI: 10.1016/j.ijhydene.2008.01.037.

[12] Ron Kohavi. “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection”. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2* 5 (1995), p. 7. ISSN: 1-55860-363-8. URL: http://robotics.stanford.edu/%7B~%7Dronnyk.

[13] J Richard Landis and Gary G Koch. “The Measurement of Observer Agreement for Categorical Data Published by : International Biometric Society Stable URL : http://www.jstor.org/stable/2529310”. In: *Biometrics* 33.1 (1977), pp. 159–174. ISSN: 0006341X. DOI: 10.2307/2529310.

[14] David McIlvenna et al. “Continuous cell sorting in a flow based on single cell resonance Raman spectra”. In: *Lab on a Chip* 16.8 (2016), pp. 1420–1429. ISSN: 14730189. DOI: 10.1039/c6lc00251j. URL: http://dx.doi.org/10.1039/C6LC00251J.

[15] John Milledge et al. “A Brief Review of Anaerobic Digestion of Algae for Bioenergy”. In: *Energies* 12.6 (2019), p. 1166. DOI: 10.3390/en12061166.

[16] Andrew Y Ng. “Preventing "Overfitting" of Cross-Validation Data”. In: *Proceedings of the Fourteenth International Conference on Machine Learning*. ICML ’97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 245–253. ISBN: 1-55860-486-3. URL: http://dl.acm.org/citation.cfm?id=645526.657119.

[17] Renate Petry, Michael Schmitt, and Jürgen Popp. “Raman Spectroscopy—A Prospective Tool in the Life Sciences”. In: *ChemPhysChem* 4.1 (2003), pp. 14–30. DOI: 10.1002/cphc.200390004. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cphc.200390004.

[18] Philipp Probst, Marvin N. Wright, and Anne Laure Boulesteix. “Hyperparameters and tuning strategies for random forest”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9.3 (2019), pp. 1–19. ISSN: 19424795. DOI: 10.1002/widm.1301. arXiv: arXiv:1804.03515v2.

[19] Petra Rosch et al. “Chemotaxonomic identification of single bacteria by micro-Raman spectroscopy: Application to clean-room-relevant biological contaminations”. In: *Applied and Environmental Microbiology* 71.3 (2005), pp. 1626–1637. ISSN: 00992240. DOI: 10.1128/AEM.71.3.1626-1637.2005.

[20] David Ruppert. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction". In: *Journal of the American Statistical Association* 99.466 (2004), pp. 567–567. ISSN: 0162-1459. DOI: 10.1198/jasa.2004.s339. URL: http://www.tandfonline.com/doi/abs/10.1198/jasa.2004.s339.

[21] Emilia Noor Sharifah and Mitsuru Eguchi. "Benefits of live phytoplankton, Chlorella vulgaris, as a biocontrol agent against fish pathogen Vibrio anguillarum". In: *Fisheries Science* 78.2 (2012), pp. 367–373. ISSN: 09199268. DOI: 10.1007/s12562-011-0465-1.

[22] Xiaofei Yuan et al. "Effect of Laser Irradiation on Cell Function and Its Implications in Raman Spectroscopy Downloaded from". In: *Applied and Environmental Microbiology* 84 (2018), pp. 2508–2525. DOI: 10.1128/AEM. URL: http://aem.asm.org/.

[23] Yongli Zhang and Yuhong Yang. "Cross-validation for selecting a model selection procedure". In: *Journal of Econometrics* 187.1 (2015), pp. 95–112. ISSN: 18726895. DOI: 10.1016/j.jeconom.2015.02.006.

[24] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net". In: (2005), pp. 301–320.