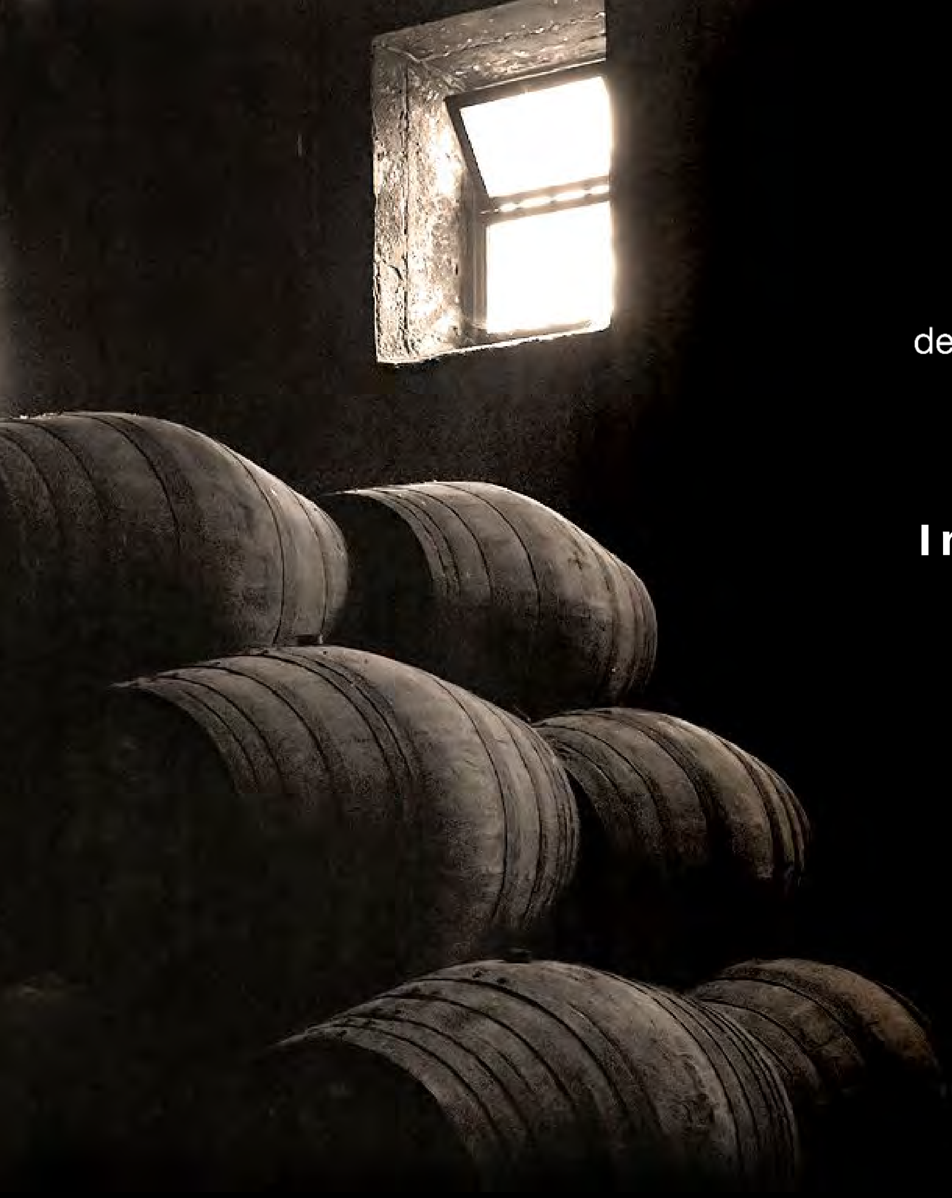


eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Imanol Bilbao Quintana

Doctoral Thesis

2023

Application of Machine Learning techniques to the personalized recommendation of Rioja red wines

Wine classification according to hedonistic tastes and score prediction

DOCTORAL THESIS

Application of Machine Learning techniques to the personalized recommendation of Rioja red wines.

Wine classification according to hedonistic tastes and score prediction.

Presentada por:

Imanol Bilbao Quintana

Para la obtención del título de Doctor por la Universidad del País Vasco / Euskal Herriko Unibertstitatea

Dirigida por el

Doctor Javier Bilbao Landatxe

Bilbao, 2023

Agradecimientos

La elaboración de una Tesis doctoral es un intenso y largo camino, plagado de dificultades que no puede ser elaborado en solitario. Si bien no me es posible incluir en este apartado a todas y cada una de las personas que han contribuido al desarrollo de la misma, tanto de forma activa como con su apoyo en los momentos más duros, me gustaría expresar mi agradecimiento aquí a gran parte de ellas.

En primer lugar, me gustaría agradecer al director de esta Tesis, el Doctor Javier Bilbao Landatxe por su apoyo, su saber hacer, sus sugerencias, su criterio siempre en positivo y su buen ánimo. De no haber sido por él creo jamás habría terminado un trabajo como éste. Una vez más gracias.

En segundo lugar, me gustaría agradecer el trabajo y la generosidad de los miembros del grupo de investigación QAProdNat (Química Analítica de Productos Naturales), en especial a la Doctora Noelia Prieto Perea y al Doctor Luis Ángel Berrueta Simal, cuyo trabajo previo sirvió de base inestimable para esta investigación, y a la Doctora Oihane Elena Albóniga Díez, catalizadora de esta relación y breve compañera de viaje.

También me gustaría expresar mi gratitud a los miembros del KEDRI (Knowledge Engineering & Discovery Research Institute) perteneciente a la AUT (Auckland University of Technology), por permitirme realizar una estancia de investigación de 6 meses en el centro y en concreto a la Doctora Elisa Capecci por hacerme sentir como en casa aun estando en las antípodas. Mi gratitud a Komsan Kaewkool por sus enseñanzas llenas de generosidad y comprensión que contribuyeron a una agradable estancia.

Debo dar también agradecimiento a la Doctora Marta Inés Saloña Bordas, también amiga, por escucharme disertar sobre la Tesis, ayudarme y aconsejarme sobre la redacción de la misma.

En último lugar me gustaría dar las gracias a mi familia y amigos. Gracias por soportarme, por apoyarme, por empujarme y por comprenderme. Gracias a todos vosotros: Aita, Ama, Elvis, Ekaitz, Beatriz, Sergio, Ibon, Zarrabeitia, Quintanilla y más y más y más. Sin vosotros este trabajo nunca hubiese sido posible.

La portada y contraportada se realizaron utilizando los recursos facilitados por la web PoweredTemplate.com.

Resumen

Este trabajo presenta una metodología para realizar recomendaciones individualizadas de vinos tintos de Denominación de Origen Rioja en tres etapas. En la primera, caracteriza cada uno de los vinos tintos mediante su contenido en antocianinas, derivados antociánicos y taninos a través de Cromatografía Líquida de Alta Resolución acoplada a detector tándem de Espectrómetro de Masas (HPLC-MS/MS). En la segunda, recoge la valoración guiada que de los distintos vinos realiza cada catador, haciendo uso de la ficha de cata estandarizada por la Organización Internacional de la Viña y el Vino (OIV), y agrupa de forma personalizada las puntuaciones en cuatro macro-categorías: recomendable positivamente, recomendable negativamente y dos categorías sin recomendación. En la tercera, aplica las técnicas de aprendizaje automático Análisis de Componentes Principales (PCA), Análisis de Discriminantes lineales (LDA), Análisis de Discriminantes Cuadráticos (QDA), algoritmo de *k* vecinos más cercanos (kNN), Árboles de Clasificación y Regresión (CART), Perceptrón Multicapa (MLP) y Redes Neuronales Probabilísticas (PNN) para construir clasificadores que produzcan recomendaciones significativas de los vinos.

El trabajo comprueba la viabilidad de la metodología aplicada sobre las valoraciones que dos catadores distintos realizan sobre los mismos vinos. Los clasificadores validados mediante validación cruzada dejando una muestra fuera (LOO) demuestran su capacidad para realizar recomendaciones negativas de vinos con un porcentaje de acierto del 100% y recomendaciones positivas de los vinos con un porcentaje de acierto superior al 90%.

Laburpena

Lan honek Errioxako Jatorri Deiturako ardo beltzen banakako gomendioak hiru etapatan egiteko metodologia aurkeztu egiten du. Lehenengo etapan, ardo beltz bakoitza bereizi egiten du bere antozianina, deribatu antozianiko eta taninoen edukiaren arabera. Kontzentrazioak neurtzeko Masen Espektrometro tandem Detektagailuari lotutako Bereizmen Handiko Kromatografia Likidoa (HPLC-MS/MS) erabiltzen du. Bigarren etapan, dastatzaile bakoitzak ardoei buruz egiten duen balorazio gidatua jasotzen da. Horretarako *Organización Internacional de la Viña y el Vino* (OIV) erakundeak estandarizatutako dastatze-fitxa erabiltzen da. Lortutako puntuazioak lau makro-kategoriatan multzokatzen ditu: positiboki gomendagarria, negatiboki gomendagarria eta gomendiorik gabeko bi kategoria. Multzokatzeak dastatzailearen arabera sortu egiten dira. Hirugarrenen etapan, Osagai Nagusien Analisia (PCA), Diskriminatzaile Linealen Analisia (LDA), Diskriminatzaile Koadratikoen Analisia (QDA), hurbilen dauden *k* bizilagunen algoritmoa (kNN), Sailkapen eta Erregresio zuhaitzak (CART), Geruza Anitzeko Pertzeptroia (MLP) eta Probabilitate Sare Neuronalak (PNN) aplikatzen ditu ardoen gomendio esanguratsuak sortzen dituzten sailkatzaileak eraikitzeko.

Lanak aplikatutako metodologiaren bideragarritasuna egiaztatzen du bi dastatzaile ezberdinek ardo berei buruz egiten dituzten balorazioen datuetatik abiatuz. Lagin bat kanpoan utzita (LOO) baliozkotze gurutzatuaren bidez sailkatzaileak balioztatu ondoren emaitzek demostratu egiten dute beren gaitasuna gomendio negatiboen %100eko asmatzea eta gomendio positiboen %90etik gorako asmatzea lortzeko.

Abstract

This work presents a methodology to make individualized recommendations of red wines of the Rioja Denomination of Origin (DO) in three stages. In the first one, it characterizes each of the red wines by their content of anthocyanins, anthocyanin derivatives and tannins using High Performance Liquid Chromatography coupled to a tandem Mass Spectrometer detector (HPLC-MS/MS). In the second one, it collects the guided evaluation of the different wines made by each taster using the tasting card standardized by the International Organisation of Vine and Wine (OIV) and, in a way adapted to each taster, groups the scores in four macro-categories: positively recommendable, negatively recommendable and two categories without recommendation. In the third stage, it applies the machine learning techniques Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), k nearest neighbor (kNN), Classification and Regression Trees (CART), Multi-Layer Perceptron (MLP) and Probabilistic Neural Networks (PNN) to build classifiers that produce meaningful wine recommendations.

The work tests the feasibility of the methodology applied on the evaluations made by two different tasters on the same wines. The classifiers validated by Leave One Out (LOO) cross-validation method demonstrate their ability to make negative recommendations of wines with a 100% correct percentage and positive recommendations of wines with a correct percentage of more than 90%.

Index

Chapter I. Introduction

1 Introduction

Wine is currently a product of remarkable consumption, with a very important direct and indirect economy generated around it. Its long historical tradition has led it to blend in, influence and even constitute part of our culture, but despite having generated such passion for centuries, measures of its quality and appreciation are still assessments based on the collegiate opinions of experts. Given that each person's experience with the same bottle is a highly personal process, is it possible to make some kind of personalized and automated predictive assessment of wine?

The range of techniques that comprise the so-called Machine Learning (ML) are postulated as suitable tools for, among other things, generating personalized predictive models automatically. They make it possible to start from accumulated data and evaluations on, in the case of this research, different red wines and to extract a verifiable prediction that summarizes the assessment that a specific consumer makes of a specific wine. The generated system, the generated machine, can offer an almost instantaneous prediction about any wine as long as it is provided with the distinctive information of the wine through the features defined in its design.

The techniques available in ML fall into two main groups according to their type of learning: supervised and unsupervised. Supervised learning requires that the data used to design the machine include the result of already known assessments. The major exponents of this type of learning are Classification and Regression Trees (CART) [1, 2], Multi-Layer Perceptron (MLP) [3, 4], Support Vector Machines (SVM) [5, 6], k nearest neighbor (kNN) algorithm [7], Naive Bayes Classifiers (NBC) [8, 9], logistic regression [10] and the simpler linear regression (LR). Each of the techniques offers different advantages and disadvantages in terms of final design accuracy, interpretability, design time, etc. Unsupervised learning works only with the features that identify the wines, without the score. Although its effectiveness in these cases may be inferior to supervised learning techniques, its contribution can be very useful to simplify the number of features that the machine will require. Among these techniques are Principal Component Analysis (PCA) and Independent Component Analysis (ICA). As it will be shown throughout the research, the combination of different techniques offers a better result in the prediction of an individual's personal assessment of any red wine.

1.1 Wine as an object of study: Economic and cultural importance of wine

Wine is certainly an interesting product. Widely consumed globally and locally, with a very significant volume of sales, its production is almost exclusive to a few countries. Wine has influenced the history of nearby regions such as La Rioja, which have evolved to modern production techniques. Its cultural ties and economic importance have led to transformations that have left their mark on many levels: agriculture, infrastructure, the social sphere, politics and even the scientific sphere.

According to the report of the Spanish Ministry of Agriculture [11] during 2018, wine represents 5.6% of the total volume of beverage consumption, being the fifth most consumed beverage in the State.

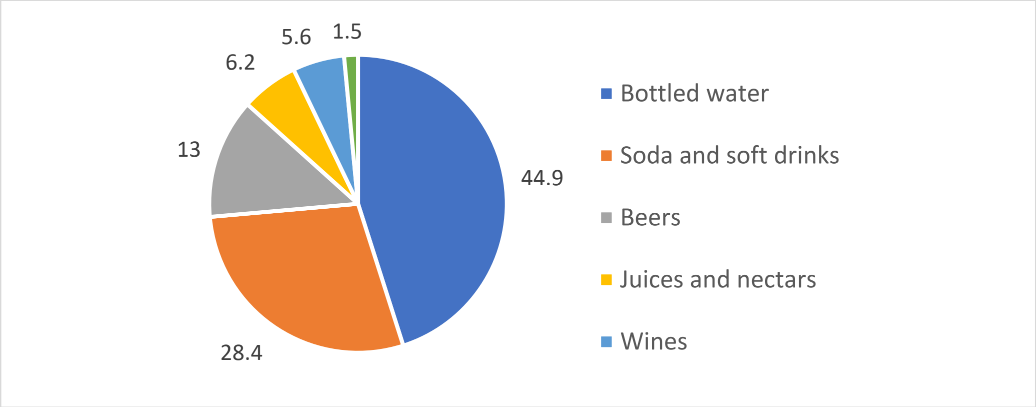


Figure 1. Percentage of Volume consumed with respect to beverages marketed in Spain during 2018. [11]

From a more global perspective and focusing only on alcoholic beverages, the data compiled by Anderson et al. [12] show that during the years 2010-2014 wine was among the most consumed alcoholic beverages worldwide becoming the majority (with respect to total alcoholic beverages) in many countries such as France, Italy or Portugal with percentages above 60%.

Year	France	Italy	Portugal	Spain	Germany	USA	Australia	Canada	Argentina	China	World Average
2010	57.4	64.0	63.5	20.5	27.8	16.8	34.7	23.2	50.9	22.3	12.80
2011	59.5	63.8	64.9	22.5	28.6	17.5	34.7	23.7	54.4	23.1	12.74
2012	59.3	66.7	68.3	23.2	28.6	17.5	35.6	24.7	53.9	23.8	12.81
2013	60.2	66.1	62.5	22.6	28.2	17.9	35.5	25.0	47.5	23.0	12.58
2014	57.1	63.7	63.3	23.2	28.1	17.9	35.5	25.6	45.1	22.4	12.50

Table 1. Volume of wine consumed as a percentage of total alcoholic beverages by country and year. [12]

The data provided by three different sources [12, 13, 14][1] show a very stable world consumption of wine in the last decade, around 245 million hectoliters of wine, a number that shows its importance.

An analysis of its production and direct sales helps to understand its economic relevance in Spain and even its capital importance in the regions of La Rioja-Rioja Alavesa. In the state as a whole, the area planted with vineyards for grape processing during 2018, according to data from the Ministry of Agriculture [15], amounted to 960,758 hectares which indicates 1.2% of the territory. In the nearby region of La Rioja this area came to mean 10.4% of the total area.

If we take the reading in the context of tilled land[2] , the percentage of cultivated land dedicated to wine in Spain as a whole corresponds to 5.2%. [16] and in regions such as La Rioja it reaches 27.78%, being, together with cereals and grain, the greatest use given to agricultural land.

All these crops produce annually, depending on the variability of each season, around 260-290 million hectoliters worldwide. [14] and 30-40 million hectoliters nationally. It should be noted that the Basque Country and La Rioja are responsible for around 20% of national production[3] .

[1] The International Organisation of Wine and Vine (OIV), which replaces the International Vine and Wine Office, was created in 2001. It is an intergovernmental scientific and technical organization with recognized competence in its field.

[2] Data are for 2016 but are consistent with the data from the Ministry of Agriculture, Fisheries and Food report [15].

[3] The Ministry of Agriculture, Fisheries and Food, through the INFOVI market information system [202], calculated an average for the 4 seasons from 2013/2014 to 2016/2017 of 39.21 million hectoliters.

This makes Spain, together with France, Italy and the United States, the main wine producers in the world, Spain being the world's leading producer in terms of volume of wine produced per capita.

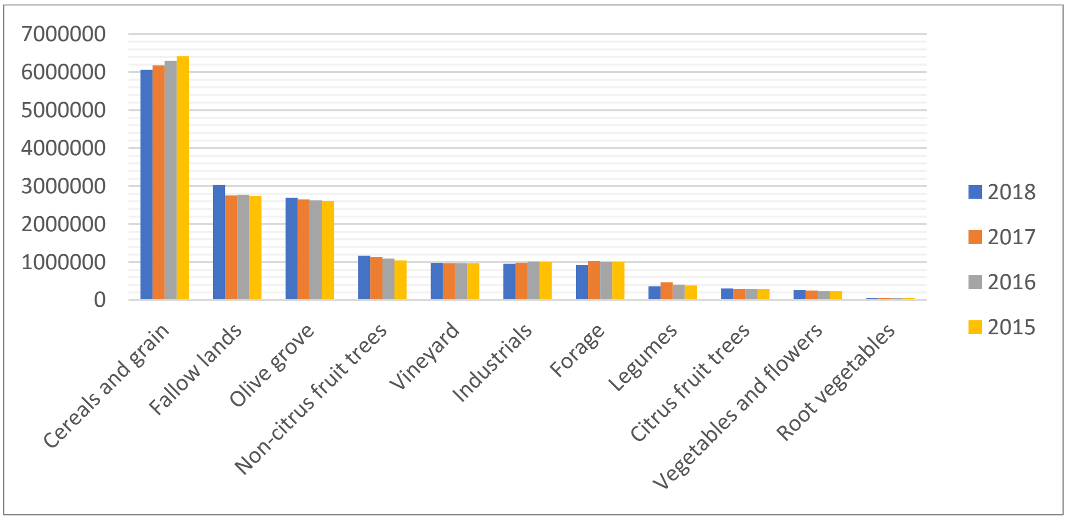


Figure 2. Number of occupied hectares in Spain by crop type. [15]

All this production translates into a considerable total volume of business. Although the most recent data do not provide a follow-up of the costs and results of Spanish wineries, attending to the Spanish Wine Market Observatory [13] and data from the Industrial Survey of Companies [17], in 2014 the total revenue was about to exceed €6.4 billion, with total profits between €125-375 million. The same data indicate that the workforce generated by this industry is estimated at 22,000 paid people. In La Rioja alone [18] the total income of the companies ranges between 700-1,000 million euros depending on the season, with profits between 50 and 150 million euros.

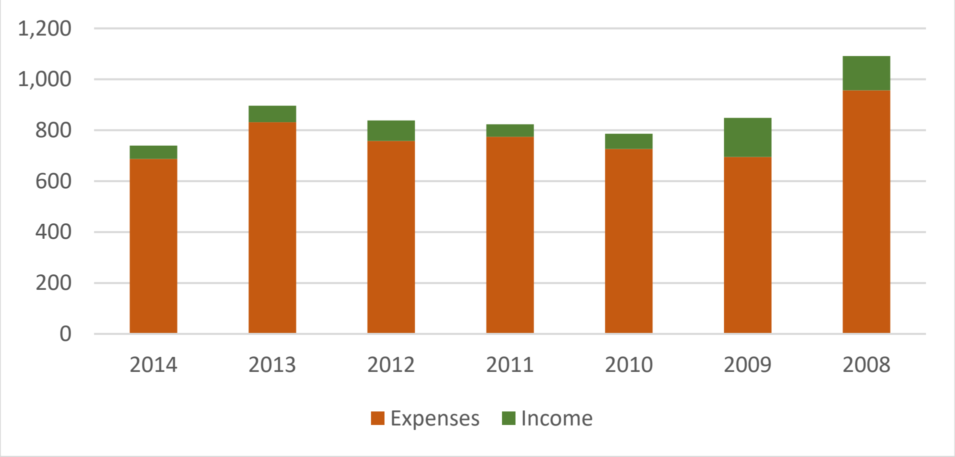


Figure 3. Total revenues and expenses in millions of euros in the community of La Rioja. [19]

The quantities for world wine exports and imports reflect the importance of the Spanish market. The globalization process has turned wine into a worldwide consumer product where many countries with a considerable volume of consumption do not produce wine and, therefore, must import it.

Both [12] and [13] show that global wine exports have seen spectacular growth between 2007 and 2017. The global volume of exports has seen an increase of around 20%, from 80-90 million hectoliters to well over 100 million. If in 2007 exports were already significant, accounting for 35% of global consumption of 255 million, in 2017 they already represent 44% of the same. The increase in sales value has been even more important, being close to 55%, with a minimum value of €18 billion in 2009 and an increase to over €32 billion in 2017.

Five major exporters from old Europe and six from the new world export 87% of the total sales value, almost 27 billion euros, and approximately 91% of the liters of wine. In terms of volume, Spain has become the world's largest wine exporter in the last 10 years, accounting for around 20% of the world's total volume and around 10% of sales value.

	Sales 2007 (Million €)	Sales 2017 (Million €)	Volume 2007 (Million hl)	Volume 2017 (Million hl)
France	6,824.0	9,101.2	15.33	14.98
Italy	3,541.8	5,988.8	18.82	21.41
Spain	1,819.3	2,854.1	15.43	22.95
Chile	915.0	1,782.8	6.11	9.45
Australia	1,816.9	1,775.0	7.81	8.17
USA	660.9	1,308.9	4.23	3.45
New Zealand	407.4	1,060.3	0.84	2.55
Germany	723.9	1,004.3	3.45	3.82
Portugal	624.1	778.4	3.70	2.98
Argentina	365.4	712.7	3.64	2.24
South Africa	490.2	534.1	5.02	5.16
World Total	20.001.0	31.010.5	90.84	100.70

Table 2. Total wine exports, by volume and sales value, by country and by year. [13]

In addition to the direct sale of wine, in recent years a new type of indirect economic activity related to wine and the cultural link it has generated over the centuries has taken off: wine tourism [20]. Wine tourism feeds on traditions, history, landscapes, landscapes, gastronomy, architecture, wine production methods and lifestyle. Thus, a new sector emerges, a new type of travel with the aim of living experiences closely related to these elements, always connected through wine as a common nexus. Such a way, a product is formed that goes beyond the sale of itself to become a historical-cultural element.

Figure 4. The Marqués de Riscal City of Wine located in Elciego (Álava).

Wine has sunk its roots in the history of different cultures, creating a strong link over the centuries, especially in the circum-Mediterranean context. In the peninsula there is evidence [21, 22] that already in the 6th century BC, after its introduction by the Greeks and Phoenicians, mainly through the areas of Huelva, Cádiz and Levante, Iberians and Celtiberians already began to produce wine locally. This is attested by the sites of Benimaquía [23], La Quéjola [24] and Cancho Roano, and it can be affirmed that, although preferably consuming beer, the Iberians also consumed wine at banquets in multiple contexts.

Throughout the following centuries, the use of wine would spread until it reached a wider diffusion in Roman times [25, 26]. In imperial Hispania, the southern and Levantine strip, with Tarraco as the nucleus, produced a respectable quantity of quality wine. When the Pax Romana ended, the agricultural decline was enormous. The prominence of wine in the ceremony of the Eucharist led the Catholic Church to assume the maintenance of the vines and their cultivation, especially for the supply of monasteries and congregations [27].

The first evidence of wine production in La Rioja dates from the eleventh and twelfth centuries [28]. The fame of its wines and the increase in production went hand in hand, especially in the sixteenth, seventeenth and eighteenth centuries. A growth model was chosen based on the increase in production but not in the quality of the wine [29], which continued to be produced with careless practices and a traditional system that did not favor its conservation and transport [28].

The 19th century saw how the influence of the Bordeaux production system finally boost the quality of Rioja wines and their entry into the international market. Powdery mildew, a vine disease characterized by small white spots, had a very negative effect on the production of Bordeaux wines, which, together with the potential of Rioja wines, caused some French wineries to move to the Haro area to try to save their situation. The changes with respect to the previous system were numerous (hygiene, care of the grape harvest, control of fermentation, racking[4], maturation in barrels, etc.) and involved adapting the wineries to their current structure, transforming the architecture of the whole area. In addition, trade was favored thanks to new infrastructures such as the inauguration of the train tracks that connected Haro with Bilbao.

From the political point of view, the *Real Sociedad Económica de La Rioja Castellana* was a key element. Although its initial objective as an association was to improve the activities of winegrowers, it ended up favoring autonomous political practices for the entire region in addition to promoting the construction of numerous infrastructures.

During the transition to the 20th century, a new pest appeared coming from America: phylloxera. A tiny aphid that forms clusters on both roots and leaves and kills the plant in approximately three years. The solution was to repopulate the area with new vines grafted on American rootstock, resistant to the parasite. It took almost two decades for the recovery, but it meant a scientific, procedural and mentality modernization in the vineyard that still continues today.

In summary, wine is a product that deserves to be studied both for its broad historical background and for its consumption and production levels, which make it an economically relevant product globally and locally.

[4] Transfer and decanting the wine.

1.2 Objectives

The objective of the current thesis is to show the applicability of different Machine Learning (ML) techniques for the personalized significant recommendation of red wines from the Rioja Designation of Origin (DO), characterizing each of the wines through their content in anthocyanins, anthocyanin derivatives and tannins by means of High Performance Liquid Chromatography coupled to a tandem detector Mass Spectrometer (HPLC-MS/MS).

What do we understand by a personalized significant wine recommendation? If an individual were to score wines according to his or her personal tastes and impressions through a guided tasting process, thus obtaining a final score for each wine, a recommendation would be a prediction of the approximate score of a wine that the individual has not yet tasted, trying to predict whether the wine will be one of those with a higher score (a positive recommendation so that in a choice between other wines the individual will opt for it), whether the wine will be one of those with a lower score (a negative recommendation so that the individual will discard it in his choice), or whether the wine will have an intermediate score (neutral recommendation or lack of recommendation).

Therefore, the objective of the thesis is specified by showing four aspects and their global results. Firstly, to show a methodology for evaluating red Rioja wines. Secondly, to define a way of grouping the scores into the three categories mentioned (positively recommended, negatively recommended and no recommendation). Thirdly, to establish a methodology to characterize each of the wines. Fourth, the choice of the supervised classifier that generalizes the data from known wines to new wines and generates new recommendations. Finally, to show the results of the integration of all these elements.

The choice of the classifier will be made based on the ML techniques most commonly used in similar research such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Multilayer Perceptrons (MLP), Probabilistic Neural Networks (PNN), Naïve Bayes Classifiers, k-Nearest Neighbor (kNN) algorithm, Support Vector Machines (SVM), Classification and Regression Trees (CART) or newer algorithms such as Impulse Neural Networks (SNN).

1.3 Contents of the document

This report is organized in 8 chapters detailing the different aspects covered in this work.

After this introductory chapter, *Chapter 2*: *Some techniques included in ML* describes the fundamentals and practical uses of different ML techniques, with special emphasis on their way of appliance, advantages and potential errors.

Chapter *3: The production of Rioja red wines, the Bordeaux model* briefly describes the vinification process of the grapes and wines discussed in the rest of the document, with the aim of showing the different phases that can affect the product.

Chapter 4: ML applied to wines, background describes the current state of the art of ML techniques applied to wine, focusing on the last two decades. It describes the different analytical techniques used to characterize wines, the number of samples used in each study and their typology, and examples of the different ML algorithms used, sorted according to the objective sought in their classification.

On the other hand, *Chapter 5: Chemical compounds of interest in red wines: Polyphenols* presents the compounds, ordered by family, that will be quantified in the research to characterize the wines, their relationship with the winemaking processes in *Chapter 3* and the methodology that will be followed to quantify them in this thesis. The last part of this chapter presents a summary of the data collected on the samples in this aspect.

Chapter 6: Organoleptic assessment of the wine, personal scoring explains the procedure that will be followed to subjectively evaluate and give a final score to each of the red wines in the study. With this aim, it first introduces the tasting sheet and then details each of its phases. In each phase, the corresponding descriptors of the tasting sheet and their possible gradations are explained, as well as the relationships that may exist between certain evaluations and the concentrations of substances or processes described in *Chapters 3* and *5*, and a brief explanation of the most common values in the Rioja red wines under study. After completing the explanation of the phases, the collected data of assessments are presented together with the groupings that will be used in the classifiers and how to measure their results.

On the other hand, *Chapter 7: Classification of red wines and Results* presents the application of the techniques described in *Chapter 2* to the data collected in *Chapters 5* and 6*.* It details the parameters chosen for each classifier, any modifications made to the algorithms to adapt them to the data, the decisions and comparisons made when creating each of the classifiers, and the overall validated results of their application for each personalized model.

Finally, *Chapter 8: Conclusions and Future lines of research* gathers the conclusions of the work carried out, detailing in a summarized way the main contributions and exposing the future work that opens after the completion of this Thesis.

Chapter II. ML Techniques

2 Some techniques included in ML

In this section we will try to explain some of the most widely used ML techniques. Although a detailed explanation of all the techniques would be far beyond the scope and objective of this thesis, we will try to describe the fundamentals, the way of use and the applications of a selection of them. The presence or not of a particular technique in this selection is based, on the one hand, on its importance in previous ML research applied to red wines (see *Chapter 4*) and, on the other hand, on the original application of this technique in the research carried out in this thesis on data from Rioja red wines (see *Chapter 7*).

Even at the risk of offering a biased view, it can be stated that ML techniques seek the generalization of conclusions drawn by the machine itself through training data to different data not previously presented to the system, real data that the machine will have to face. The final objective of these techniques can be described in one of the following two mutually exclusive ways: either the prediction of the numerical value of a characteristic or parameter from the proposed data (known as regression problem), or the assignment of a data to a category or class (known as classification problem). In the first case the final result, the output of the system, is usually a real number whose error depends on the difference with respect to the measurement of the parameter represented in reality. In the second case the result is an arbitrary number, symbol, or some kind of trigger signal representing one of the classes in the set of possible classification classes. The error in the latter case will depend arbitrarily on how many cases are correctly classified in their actual category and how many cases are misinterpreted. During the training phase, by adjusting machine parameters and weights or through some transformation or grouping, those features that are of interest to represent the data will be created and selected, and in this new space that they create, the decisions that limit each classification zone will be established, partitioning the space by means of decision boundaries. During the classification phase, the patterns will be assigned to one of these partitions, without modifying them later on.

In addition, machine learning is often categorized as either supervised or unsupervised learning [30, 31, 32]. Supervised learning uses the data together with the final result, the correct answer, associated with each data. In the case of a regression problem, each of the data used to train the machine will have the actual value of the desired output associated with it. In the case of a classification problem, the training data include a label that identifies to which category they belong, the examples with which the system learns are, in some way, marked and the classes to which the data may belong are predefined. In unsupervised techniques, the category of the examples and even how many categories can be differentiated in the set are unknown. It is up to the machine itself to look for similarities in the data and draw some conclusions, which is why these techniques are used to analyze the data and discover intrinsic relationships between their characteristics.

It is worth noting that ML techniques have certain typical problems. Sometimes optimizing a classifier to maximize its performance on a training set does not always lead to a good result on new data, it does not generalize well despite showing apparently good results on the training set. This according to Jain et al. [33] can be due to three factors: *the curse of dimensionality* (too many features in the data relative to the number of samples), classifier complexity (the classifier has too many design parameters with respect to the number of samples), or overtraining or overfitting [34] (it has been optimized too much over the training set).

The selection of which data features to use is crucial in any classification problem. Eliminating irrelevant or redundant features that do not contribute any information and discarding those features that in classification only introduce noise and only contribute to worsen machine results offers significant advantages. This dimensionality reduction leads to better classifiers in terms of accuracy, shorter information processing times, reduced storage requirements and lower costs when acquiring the data. Fukunaga and Hayes [35] go so far as to show that the relationship between the number of samples and the dimensionality of the data (the number of features) is an important factor in improving the error of classifiers, it is what other authors will call *the curse of* dimensionality [33]. As can be seen in Table 3, in the case of linear classifiers a ratio of 10:1 or even 5:1 may be sufficient to perform competent estimations and obtain accuracies close to optimal classifiers.

Number of samples (r)	Data dimensionality (n)				
	4	8	16	32	64
$r = 3 \cdot n$	0.144	0.143	0.134	0.130	0.132
$r = 5 \cdot n$	0.117	0.122	0.121	0.120	0.121
$r = 10 \cdot n$	0.105	0.109	0.109	0.109	0.109
$r = 15 \cdot n$	0.105	0.108	0.106	0.106	0.106
$r = 20 \cdot n$	0.104	0.104	0.106	0.104	0.105
$r = 30 \cdot n$	0.104	0.103	0.103	0.103	0.103
$r = 40 \cdot n$	0.103	0.103	0.102	0.102	0.102
$r = 50 \cdot n$	0.102	0.102	0.102	0.102	0.102

Table 3. Error degradation of a linear classifier of optimal error 0.1 depending on the number of samples used and the size of the data. [35]

In quadratic and nonlinear classifiers (see Table 4) the ratio may not be constant and may need a much larger number of samples as the dimensionality increases, in fact, in the case of the quadratic classifier Fukunaga and Hayes [35] show that what we have to try to keep constant is the rate r/n^2, where r is the number of training examples and n is the dimensionality of the data.

Number of samples (r)	Data dimensionality (n)				
	4	8	16	32	64
$r = 3 \cdot n$	0.167	0.204	0.220	0.267	0.313
$r = 5 \cdot n$	0.140	0.164	0.173	0.208	0.255
$r = 10 \cdot n$	0.115	0.124	0.136	0.157	0.193
$r = 15 \cdot n$	0.109	0.118	0.123	0.141	0.166
$r = 20 \cdot n$	0.108	0.116	0.119	0.139	0.151
$r = 30 \cdot n$	0.105	0.107	0.111	0.130	0.136
$r = 40 \cdot n$	0.104	0.106	0.109	0.115	0.127
$r = 50 \cdot n$	0.123	0.104	0.107	0.113	0.122

Table 4. Error degradation of a quadratic classifier of optimal error 0.1 depending on the number of samples used and the size of the data. [35]

Ultimately, given that the number of samples available to us in many investigations is limited due to the scarcity of them and the difficulty of collecting more data, reducing the dimensionality of the samples to acceptable ratios plays a very important role. Measures such as Vapnik-Chervonenkis dimensionality [36] can provide guidance in this regard.

Along with *the curse of dimensionality* itself, and partly due to it, overtraining with the aim of improving classification results with the few samples that are usually available is another bad habit in the ML world. Techniques such as regularization or appropriate classifier design, terminating the training phase when the improvement curve on the results seems to stabilize, can help to mitigate this [32, 34].

2.1 Bayesian Classifiers (BC)

A big family of classifiers are based on or have their equivalent in some kind of Bayesian classifier. This interpretation stems from a purely statistical/probabilistic treatment of the data that has been extensively explained in references such as [1, 8, 9, 33] and which we will try to explain briefly here.

A classifier seeks to correctly assign a datum represented by a vector of d features, to one of c predefined different categories. We will symbolize from now on the data as $\boldsymbol{x} = (x_1, x_2, \dots, x_d)$ and the categories or classes as $C_1, C_2, \dots, C_c$.

A good classifier looks for and assigns that category to which a lower derived risk is associated in case of error. If we must find a category for our data $\boldsymbol{x}$, we will first calculate the risk or consequences of making a wrong decision by wrongly selecting the category C_i through the expression

$$R(C_i|\boldsymbol{x}) = \sum_{j=1}^{c} L(C_i, C_j) P(C_j|\boldsymbol{x}) \tag{1}$$

where $L(C_i, C_j)$ represents the losses of deciding to classify an element as C_i when in fact it belongs to class C_j. The losses in case of getting the category right will be null, and the most usual, with some exceptions, is to take the losses of any wrong decision as 1; that is

$$L(C_i, C_j) = \begin{cases} 0 \; si \; j = i \\ 1 \; si \; j \neq i \end{cases} \tag{2}$$

In that particular case the equation (1) can be rewritten as $R(C_i|\boldsymbol{x}) = 1 - P(C_i|\boldsymbol{x})$.

When selecting the final category we will use the most conservative criterion possible, we will select the one whose risk is the minimum possible. If (2) is satisfied, this will occur for the category where $P(C_i|\boldsymbol{x})$ is maximum, a tremendously intuitive criterion also called maximum a posteriori probability (MAP) rule.

BCs rely on Bayes' theorem to calculate the different $P(C_i|\boldsymbol{x})$ and thus compare them. From the probabilistic point of view the already known examples of supervised learning used to train the classification machine so that the machine learns to classify, represent observations that provide us with $P(\boldsymbol{x}|C_i)$. Depending on whether the data features $x_1, x_2, \dots, x_d$ are discrete or continuous the probability function representing $P(\boldsymbol{x}|C_i)$ will be a mass function or it will be a density function[5]. Using Bayes' theorem one can calculate each $P(C_i|\boldsymbol{x}) = P(C_i)P(\boldsymbol{x}|C_i)/P(\boldsymbol{x})$ and among all of them look for the maximum one. When comparing all the *a posteriori*

[5] Strictly speaking in the case of continuous variables the $P(\boldsymbol{x}|C_i)$ is not directly represented by the density function but by the integral defined over the density function, that is $P(\boldsymbol{x_A}|C_i) = \int_{\boldsymbol{A}} f(\boldsymbol{x}|C_i) d\boldsymbol{x}$. Throughout the text, a more intuitive notation has been chosen.

probabilities, the denominator $P(\boldsymbol{x})$ will be common to all of them, so we can eliminate it from the expression, leaving

$$\max_i P(C_i|\boldsymbol{x}) = \max_i P(C_i)P(\boldsymbol{x}|C_i) \tag{3}$$

For the more general case where the loss function is not the one indicated in (2), it is necessary to evaluate $R(C_i|\boldsymbol{x})$ for each class and choose that C_i with a lower value. In this case the simplest way to evaluate it is to calculate, as in the previous case, $P(C_i|\boldsymbol{x}) = P(C_i)P(\boldsymbol{x}|C_i)/P(\boldsymbol{x})$ without counting the term $P(\boldsymbol{x})$ that is present equally in each and every one of the $R(C_i|\boldsymbol{x})$ corresponding to the different categories. Thus, we would evaluate for each i, one by one, the expression

$$R(C_i|\boldsymbol{x}) = \sum_{j=1}^{c} L(C_i, C_j)P(C_j)P(\boldsymbol{x}|C_j) \tag{4}$$

and we would choose the category offering the lowest risk.

The different $P(C_i)$ should reflect the expected frequency of each category in the future real cases that the machine has to classify. The best choice is the one that is closest to the future reality. The values can be provided by experts using previous knowledge of the problem, they can be the result of a hypothesis such as the assumption of a normal or uniform distribution, or they can be estimated through the proportion of samples of each class i in the training dataset[6].

Knowing the value of $P(\boldsymbol{x}|C_i)$ is somewhat more complicated. If, fortunately, all probability functions $P(\boldsymbol{x}|C_i)$ are specified, we are faced with the optimal BC. In case we can calculate it, usually cause we are dealing with predesigned experiments where the data have been generated according to a known probability function, the optimal BC serves as a comparative ceiling against the performance of our own designs. Unfortunately, most of the time we do not know these probability functions in practice, so we must estimate them from the training data. At this point there are two main alternatives. If we assume that we know the form of the probability functions (e.g. multivariate Gaussian), the problem is reduced to estimating the parameters that define these functions (e.g. mean and covariances) in order to substitute them and make the decision. If, as is usually the case, we do not know also the form of the probability functions, we will have to estimate directly the function in a more generic form.

As an example, the Naïve Bayes Classifier (NBC) considers that the features forming $\boldsymbol{x}$ are independent of each other and that the estimation of $P(\boldsymbol{x}|C_j)$ can be done by estimating each of the features separately and then multiplying them $x_1, x_2, \dots, x_d$ separately and then multiplying them with each other. In other words, it first estimates separately on the training data each of the $P(x_l|C_j)$ (with $l = 1 \dots d$) and then, thanks to the assumed independence between the features, it estimates the $P(\boldsymbol{x}|C_j) = \prod_l P(x_l|C_j)$.

2.2 Principal Component Analysis (PCA)

PCA is an unsupervised learning technique that allows us to create a new universe of features from the existing ones through a linear transformation and then select a smaller number of them losing as little information as possible. As stated by Goodfellow et al. [31] PCA allows us to move from an m-dimensional data space to a lower dimensional n-dimensional data space by

[6] A reverse path is also possible. Some authors such as Briemann [2] when they have a sufficiently large number of samples, choose the training cases so that their proportion matches the previously known proportion of future cases.

minimizing the Euclidean distance between the real point represented by $\boldsymbol{x} \in \mathbb{R}^m$ and the approximation represented by $\boldsymbol{y} \in \mathbb{R}^n$. That is, the method helps to find a vector space of dimension $n < m$ on which to project the data so that the overall projection error is minimized. It thus performs a lossy compression.

From the mathematical point of view, the final objective of PCA is to find an orthogonal matrix $U_{m\times m}$ that allows a change of features for new ones uncorrelated with each other and in decreasing order of variance, that is, new features whose covariance matrix is diagonal and with the elements of the main diagonal arranged from largest to smallest. Let $[\boldsymbol{x_1} \quad \boldsymbol{x_2} \quad \cdots \quad \boldsymbol{x_r}]$ be the matrix of original data represented by each data of dimension arranged in a column, let m arranged in a column, let S_x be its covariance matrix, if $[\boldsymbol{y_1} \quad \boldsymbol{y_2} \quad \cdots \quad \boldsymbol{y_r}]$ the matrix of the data in the new features exists, with each data of dimension arranged in a column, the change n arranged in a column, the change of variable matrix U that obtains the r original data from the modified ones is

$$[\boldsymbol{x_1} \quad \boldsymbol{x_2} \quad \cdots \quad \boldsymbol{x_r}] = U\,[\boldsymbol{y_1} \quad \boldsymbol{y_2} \quad \cdots \quad \boldsymbol{y_r}] \tag{5}$$

where thanks to its orthogonality

$$U^{-1} = U^T \tag{6}$$

If the average of data is

$$\boldsymbol{\mu_x} = \frac{1}{r}(\boldsymbol{x_1} + \cdots + \boldsymbol{x_r}) \tag{7}$$

then, the covariance matrix of the original data is computable using

$$\mathrm{S_x} = \frac{1}{r-1}[\boldsymbol{x_1} \quad \boldsymbol{x_2} \quad \cdots \quad \boldsymbol{x_r}][\boldsymbol{x_1} \quad \boldsymbol{x_2} \quad \cdots \quad \boldsymbol{x_r}]^T + \boldsymbol{\mu_x}\boldsymbol{\mu_x^T} \tag{8}$$

If we assume that $\boldsymbol{\mu_x} = \boldsymbol{0}$, what we can achieve simply normalizing the data, the new covariance matrix of the data may be written as $\mathrm{S}_y = U^T\mathrm{S}_xU$. This is the matrix we will try to make diagonal. Let the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots > \lambda_m$ of S_x and $U = [\boldsymbol{u_1} \quad \boldsymbol{u_2} \quad \cdots \quad \boldsymbol{u_m}]$ be the matrix whose columns are the corresponding unit modulus eigenvectors, we will have achieved the goal [37]. An accurate representation of the data requires the m eigenvectors, but if we dispense with the last components, those associated with the lowest eigenvalues, the approximation made will have the advantage of having lost the smallest orthogonal projections.

The variance captured by each component is represented by the elements λ_i of the diagonal of S_y. In other words, the mean error made when approximating the data

$$error = \frac{1}{r}\left(X - X_{aprox}\right)^T\left(X - X_{aprox}\right) \tag{9}$$

will coincide with the sum of the eigenvalues of the unselected components to form $U_{reducida}$, that is, $\lambda_{n+1} + \cdots + \lambda_m$. Thus, the coefficient $\lambda_i/tr(\mathrm{S}_y) = \lambda_i/tr(\mathrm{S}_x)$ measures how much of the total variance is captured by the ith component. The first principal component captures the maximum variance of the data that a single vector can capture. The second component will capture the maximum possible variance among the orthogonal, uncorrelated features with the first component. The third will do so with the orthogonal, uncorrelated features with both of the previous components, and so on. Data that have a higher correlation between their features will therefore need fewer principal components to represent the information while maintaining a small error. The more linearly related some of the original features that make up the data are, the smaller the error made in the simplification.

Figure 5 shows two linearly related features. The lengths of each of the principal components symbolize the value of the associated eigenvalue. The first component, much larger than the second, captures almost 97% of the total variance and as can be seen the approximations thus achieved incur a small error.

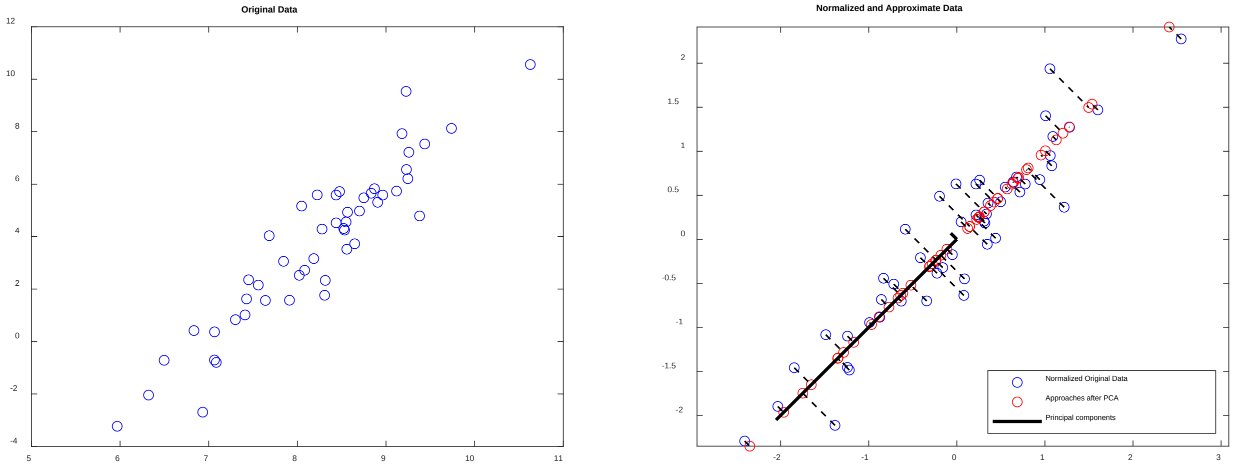


Figure 5. PCA on some random data with linearly related features. The original two-dimensional data are plotted on the left. The normalized data and its approximation in the first PCA component are plotted on the right.

Figure 6 shows what happens when the features are not linearly related to each other. The first component is barely able to capture 52% of the variance, thus incurring a considerable error in approximating the data.

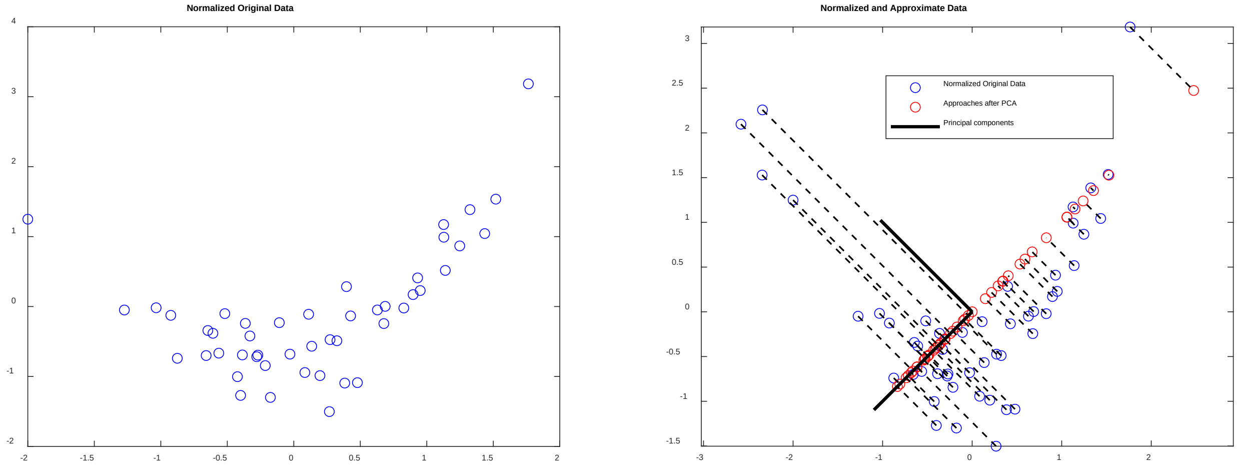


Figure 6. PCA on some random data with non-linear features. The original two-dimensional data are plotted on the left. The normalized data and its approximation in the first PCA component are plotted on the right.

To apply PCA we will take all the training data $\boldsymbol{x} \in \mathbb{R}^m$ without its associated label or corresponding outcome, as this is an unsupervised technique. We will start by correcting the mean, skewing it, so that the new mean is zero and thus simplify the subsequent calculation of the covariance matrix. Simultaneously, we will normalize each characteristic to avoid the greater influence of those with a higher value. After it, we will calculate the transformation matrix U formed by the n most important eigenvectors of the covariance matrix. From this point on, any new data entering our machine will first have to be mean-adjusted, scaled and transformed by means of U. The matrix U must be the same as the one originally calculated only with the training cases, it is not updated unless the whole machine is trained again. It is important to emphasize that new data will not influence the U calculation.

Starting from a data matrix, as usually presented, consisting of samples arranged in rows as r samples arranged in rows as

$$X = \begin{bmatrix} \boldsymbol{x_1}^T \\ \boldsymbol{x_2}^T \\ \vdots \\ \boldsymbol{x_r}^T \end{bmatrix}$$

the steps to apply the algorithm are as follows:

Step 1. Center the data around 0 subtracting the mean, and normalize the features with some form of scaling (FS). It is important that the new variable has mean 0 to agree with the theory. For each feature of the data $i \in [1, m]$ we will calculate separately its mean μ_i and a measure of its variance, such as the range or more commonly the standard deviation, which we will denote by s_i. With these parameters, a pair for each feature x_i, we will fit each of the data entries:

$$x_i = \frac{x_i^{unnormalized} - \mu_i}{s_i} \tag{10}$$

By dividing by the standard deviation we get normalized data with new average 0 and variance 1.

Step 2. We will now calculate the covariance matrix on the normalized data. If the new matrix of the data consists of r normalized samples arranged in rows, as usual, the covariance matrix $\mathrm{S}_{m\times m}$ can be calculated directly as:

$$\mathrm{S} = \frac{1}{r} X^T \cdot X \tag{11}$$

which due to its definition will always be positive definite.

Step 3. It's time to obtain the eigenvalues and associated eigenvectors. The most widely used algorithm for this is the singular value decomposition and singular vector decomposition of Sthen finding the eigenvalues as their corresponding squares and selecting their eigenvectors from the appropriate singular vectors. This is because the iterative computation of the singular value decomposition is usually faster and more accurate.

Step 4. Take the n eigenvectors of S associated to the largest eigenvalues. With these n eigenvectors we will form the new basis of smaller dimensions. The rest of the information associated to the smaller eigenvalues will be lost. Having obtained the eigenvalues in descending order we will directly take the first n columns of the matrix U:

$$U_{reduced} = [\boldsymbol{u_1} \quad \boldsymbol{u_2} \quad \cdots \quad \boldsymbol{u_n}] \tag{12}$$

Although there are different criteria for selecting the number of principal components, it is usual, as explained by Goodfellow et al. [31] that n is chosen so that the mean error committed does not exceed a certain percentage of the variance of the data. Typically, n is chosen to retain at least 99%, 95% or 90% of the variance. This criterion can always be altered with the aim of achieving a further reduction in the number of components if subsequent classification or regression proves effective.

The covariance matrix S calculated above has along its main diagonal the variance of each of the ith features, i.e. the element S_{ii} represents the variance of the feature i throughout the data. We can therefore calculate the total variance of the data as the trace of the covariance matrix. Having normalized the data matrix with the standard deviation of each feature, all the S_{ii} elements of the diagonal will be 1and the total variance obtained should be m, the initial number of features.

With this in mind, we will select the number n of components so that the percentage of $1-\frac{error}{variance}=\frac{\lambda_1+\lambda_2+\cdots\lambda_n}{m}$ is ≥ 0.99 or 0.95 or 0.9.

Step 5. Finally, we obtain the transformed data in the new vector space. For the training dataset we will have $X_{reduced}=X\cdot U_{reduced}$. For any new data we should enter $\boldsymbol{x_{reduced}}^T=\boldsymbol{x_{normalized}}^T\cdot U_{reduced}$ with $\boldsymbol{x_{reduced}}^T$ being of dimension $1\times n$, $\boldsymbol{x_{normalizada}}^T$ of $1\times m$ and $U_{reduced}$ of $m\times n$.

Step 6. Obtain the transformed data in the new vector space. For the training dataset we will have $X_{reducida}=X\cdot U_{reducida}$. For any new data we have to introduce $\boldsymbol{x_{reducida}}^T=\boldsymbol{x_{normalizada}}^T\cdot U_{reducida}$ being $\boldsymbol{x_{reducida}}^T$ of dimensions $1\times n$, $\boldsymbol{x_{normalizada}}^T$ of $1\times m$ and $U_{reducida}$ of $m\times n$.

Exceptionally, if we would like to calculate the projection of the data into the new vector space, the approximation of the reduced data, we could do following the expression $X_{aprox}=X_{reduced}\cdot U_{reduced}{}^T$.

The usefulness of this technique is twofold. On the one hand, as we have stated above, it serves to compress the input data, thus reducing the space needed to store the data and speeding up the training time of supervised learning algorithms. On the other hand, it makes it possible to visualize multidimensional data in 2D or 3D, facilitating their understanding. In addition, some authors such as Tharwat et al. [38] indicate that by compressing PCA data it can also remove the low intensity noise that accompanies the data, and can be used as a noise filter. We will focus just on its first utility.

As highlighted by Jain et al. [33] as PCA is a technique that creates new features from linear combinations of the real features, it can obscure and hinder the interpretation of decision making in the real problem to the point of making it impossible. However, and especially if the correlation between different features in the data is high, applying PCA can lead to significant reductions in the number of features needed and significant improvements in the rate of the final classifier [35].

It is important to emphasize that whenever the technique is used, information is lost and its use is not always recommended. From the point of view of ML training sets its use will always eliminate some part of the data that has been collected, simplifying it. Different ML techniques, such as MLP and CART, are intended to search and find the relationships between variables themselves. In the case of CART, Breiman et al. [1] show that the inclusion of variables that have nothing to do with the output (even in greater number than those that do have a relationship) does not alter the result of the algorithm at all, being simply ignored in the result. Therefore, authors such as Andrew NG [32] do not recommend applying PCA before applying the corresponding ML algorithm, because if the ratio of training samples to the number of variables is more than sufficient to avoid the curse of dimensionality, it is preferable that the ML technique itself has as much information as possible, without simplifications, since part of its power lies in being able to discern the importance of the variables. In such cases it is not recommended to use PCA, not even to prevent overtraining [34] because even if it could achieve it, equal or better results can be obtained with regularization, a technique that does not remove information from the data that could be valuable.

If, for example, one of the features that provides little information in the input dataset has a greater relationship with the desired output variable, PCA can minimize its effect and it is impossible for the ML technique to find its influence on the output variable. PCA identifies the variables with more information in the input set but not those that have a greater relationship with the output variable. In case they are different the PCA technique will only cause the ML algorithm to fail. Unfortunately, as we discussed in the beginning of section 2, there are not always as many training cases available as we would like. In those circumstances applying PCA before the classifier algorithm can lead to significant improvements on the final result. In such cases applying PCA makes sense, but the final results should be reviewed in the future with a larger number of data and without applying PCA.

It is a mistake, therefore, to use PCA out of habit in the design process of a learning system. It is advisable to always test the learning algorithm without applying PCA. Only if we see that the results of the system are not good, either because it becomes computationally unmanageable (because it requires too much disk/memory space or because it becomes too slow), or because the ratio of features to number of samples is low, we will try PCA. By not using PCA we will use the potential of all the data we have, which, if possible, is always better.

It is also important to emphasize that when using PCA to help in graphic visualizations, we must choose well the features that are combined because it is best that they are somehow related to each other so that the new feature that is generated makes some sense when interpreting it in the visualization.

2.3 Fisher's Discriminant, Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA)

Fischer's discriminant [39] is a supervised learning technique that, without assuming any kind of distribution in the data, applies a linear transformation on the feature space trying to keep the different samples associated to different categories as far apart as possible in the new space. If we use the same notation as in section 2.1, like PCA, the Fisher discriminant helps to find a vector space of dimension $n < m$ but uses the knowledge we have about each sample, the category associated with each sample, to find a space where the separability of the classes is simpler. Also, as in the case of PCA, the features created from the new space are linear combinations of the features of the original space which can make their interpretation more difficult to compare or the original features.

The criterion introduced by Fisher [39], on the basis of the data with their respective associated categories, tries to obtain a new space where the distance between categories is as large as possible while minimizing the dispersion of the data in each category. In order to apply it, therefore, we must know in advance the number of classes that exist and the training samples must be labeled with their respective classes. It will try to maximize the ratio of the interclass variance, the distance between the mean representative of each class, with respect to the internal variance of each class.

Once the data has been compressed into new features we can use any classifier to make decisions in the new space. The technique known as LDA, assumes that the distributions of each of the categories are of Gaussian type and with the same covariance matrix common to all of them, and leads to a linear classifier. Thus, we would classify the samples in the new space according to their closeness to the centroid of each class in the new space with a fit that depends on the prior probabilities of each category in the total samples. The decision boundaries generated in this way by LDA are linear separation hyperplanes. The fact that starting from the Gaussian assumption leads, when diagonalizing the covariance matrix, to the same result causes both techniques to be confused, and the term LDA is often used not only for the classifier but also for the methodology used to reduce the number of features.

Let us develop how Fischer's method works and how it is applied. Suppose then, that we start from a training data formed by r examples of dimension m arranged as columns in the matrix $X = [\boldsymbol{x_1} \quad \boldsymbol{x_2} \quad \cdots \quad \boldsymbol{x_r}]$, where each $\boldsymbol{x_i}$ is a column belonging **to** $\mathbb{R}^m$, and their corresponding class labels are $[l_1 \quad l_2 \quad \cdots \quad l_r]$. The labels will indicate to which class of all possible classes each sample belongs, i.e., $l_i \in \{C_1, C_2, \dots, C_c\}$ with c being the number of possible classes.

Fischer will try to find a new representation for the data, $[\boldsymbol{y_1} \quad \boldsymbol{y_2} \quad \cdots \quad \boldsymbol{y_r}]$, in a new space of features where $n \leq m$. Thus, each data $\boldsymbol{y_i}$ will be a column vector belonging to $\mathbb{R}^n$. Following the same notation as in section 2.1, the variable change matrix U that obtains the original r data from the modified ones is

$$[\boldsymbol{x_1} \quad \boldsymbol{x_2} \quad \cdots \quad \boldsymbol{x_r}] = U\,[\boldsymbol{y_1} \quad \boldsymbol{y_2} \quad \cdots \quad \boldsymbol{y_r}] \tag{13}$$

and thanks to its orthogonality

$$U^{-1} = U^T \tag{14}$$

The first step to obtain U is to calculate the measure of separability between distinct classes: the variance matrix between distinct classes, S_{inter} from the separation between each class i and the rest, $S_{inter,i}$. The second step is to calculate the measure of diversity of the samples of each class (the intra-class variance matrix, $S_{intra,i}$) and from it that of all classes S_{intra}. The third step is to construct the new dimensional space that maximizes separability and minimizes diversity.

The matrix S_{inter} of class i represents the distance between the class i and the other classes, mathematically the distance between the mean of class i and the mean of all the data taken as a whole. In the original data the mean μ_i of each class i, and the total mean of the data can be calculated by means of

$$\boldsymbol{\mu_i} = \frac{1}{n_i}\sum \boldsymbol{x_i} \tag{15}$$

$$\boldsymbol{\mu} = \frac{1}{r}\sum_{j=1}^{r} \boldsymbol{x_j} = \sum_{i=1}^{c} \frac{n_i}{N}\boldsymbol{\mu_i} \tag{16}$$

where n_i is the number of training samples associated with class i and x_i are the training samples associated with class i.

By applying the transformation, in the new vector space the averages will be converted to

$$\boldsymbol{m_i} = U^T\boldsymbol{\mu_i} \tag{17}$$

$$\boldsymbol{m} = U^T\boldsymbol{\mu} \tag{18}$$

The squared norm, a good measure of their difference, can be found by the equations

$$\|\boldsymbol{m_i} - \boldsymbol{m}\|^2 = Tr[(\boldsymbol{m_i} - \boldsymbol{m})(\boldsymbol{m_i} - \boldsymbol{m})^T] \tag{19}$$

$$\|\boldsymbol{m_i} - \boldsymbol{m}\|^2 = Tr[(U^T\boldsymbol{\mu_i} - U^T\boldsymbol{\mu})(U^T\boldsymbol{\mu_i} - U^T\boldsymbol{\mu})^T] \tag{20}$$

$$\|\boldsymbol{m_i} - \boldsymbol{m}\|^2 = Tr[U^T(\boldsymbol{\mu_i} - \boldsymbol{\mu})(\boldsymbol{\mu_i} - \boldsymbol{\mu})^T U] \tag{21}$$

The term $(\boldsymbol{\mu_i} - \boldsymbol{\mu})(\boldsymbol{\mu_i} - \boldsymbol{\mu})^T$ represents the distance between each class and the overall mean, equivalent to the distance between classes, and is what we will call the individual interclass distance

$$\mathrm{S}_{inter,i} = (\boldsymbol{\mu_i} - \boldsymbol{\mu})(\boldsymbol{\mu_i} - \boldsymbol{\mu})^T \tag{22}$$

Thus, we can express $\|\boldsymbol{m_i} - \boldsymbol{m}\|^2$ as

$$\|\boldsymbol{m_i} - \boldsymbol{m}\|^2 = Tr\big(U^T \mathrm{S}_{inter,i} U\big) \tag{23}$$

The average sum of the differences of the different classes, taking into account that the classes with more samples should contribute more to this sum is as follows

$$\sum_i \frac{n_i}{n} \|\boldsymbol{m_i} - \boldsymbol{m}\|^2 = \frac{n_i}{n} \sum_i Tr\big(U^T \mathrm{S}_{inter,i} U\big) = Tr\left(U^T \sum_i \frac{n_i}{n} \mathrm{S}_{inter,i}\, U\right) \tag{24}$$

$$\sum_i \frac{n_i}{n} \|\boldsymbol{m_i} - \boldsymbol{m}\|^2 = Tr(U^T \mathrm{S}_{inter} U) \tag{25}$$

where

$$\mathrm{S}_{inter} = \sum_i \frac{n_i}{n} \mathrm{S}_{inter,i} = \sum_i \frac{n_i}{n} (\boldsymbol{\mu_i} - \boldsymbol{\mu})(\boldsymbol{\mu_i} - \boldsymbol{\mu})^T \tag{26}$$

Let us now focus on the next step, the matrix of class S_{intra} of class i which represents the dispersion between the elements of the same class, mathematically the distance between the mean of class i and the rest of the samples of the same class.

For a sample j of the class i its squared norm would be

$$\left\|\boldsymbol{y_j} - \boldsymbol{m_i}\right\|^2 = Tr\left[\left(\boldsymbol{y_j} - \boldsymbol{m_i}\right)\left(\boldsymbol{y_j} - \boldsymbol{m_i}\right)^T\right] \tag{27}$$

$$\left\|\boldsymbol{y_j} - \boldsymbol{m_i}\right\|^2 = Tr\left[\left(U^T\boldsymbol{x_j} - U^T\boldsymbol{\mu_i}\right)\left(U^T\boldsymbol{x_j} - U^T\boldsymbol{\mu_i}\right)^T\right] \tag{28}$$

$$\left\|\boldsymbol{y_j} - \boldsymbol{m_i}\right\|^2 = Tr\left[U^T\left(\boldsymbol{x_j} - \boldsymbol{\mu_i}\right)\left(\boldsymbol{x_j} - \boldsymbol{\mu_i}\right)^T U\right] \tag{29}$$

If we accumulate the distances between all the samples of a class we will have

$$\sum_{j\ from\ class\ i} \left\|\boldsymbol{y_j} - \boldsymbol{m_i}\right\|^2 = \sum_{j\ from\ class\ i} Tr\left[U^T\left(\boldsymbol{x_j} - \boldsymbol{\mu_i}\right)\left(\boldsymbol{x_j} - \boldsymbol{\mu_i}\right)^T U\right] \tag{30}$$

$$\sum_{j\ from\ class\ i} \left\|\boldsymbol{y_j} - \boldsymbol{m_i}\right\|^2 = Tr\left[U^T \sum_{j\ from\ class\ i} \left(\boldsymbol{x_j} - \boldsymbol{\mu_i}\right)\left(\boldsymbol{x_j} - \boldsymbol{\mu_i}\right)^T U\right] \tag{31}$$

The term $\sum_{j\ from\ class\ i}(\boldsymbol{x_j}-\boldsymbol{\mu_i})(\boldsymbol{x_j}-\boldsymbol{\mu_i})^T$ represents the dispersion of the samples of each class. In order that the classes with more elements do not contribute a much greater weight than those with less we will look for the average of the distances rather than their accumulation by dividing the whole expression by the number of samples of each class n_i, leaving definitively the expression

$$\sum_{j\ from\ class\ i}\frac{1}{n_i}\left\|\boldsymbol{y_j}-\boldsymbol{m_i}\right\|^2 = Tr\left[U^T\frac{1}{n_i}\sum_{j\ from\ class\ i}(\boldsymbol{x_j}-\boldsymbol{\mu_i})(\boldsymbol{x_j}-\boldsymbol{\mu_i})^T\,U\right] \tag{32}$$

$$\sum_{j\ from\ class\ i}\frac{1}{n_i}\left\|\boldsymbol{y_j}-\boldsymbol{m_i}\right\|^2 = Tr\left(U^T\mathrm{S}_{intra,i}U\right) \tag{33}$$

with

$$\mathrm{S}_{intra,i} = \frac{1}{n_i}\sum_{j\ from\ class\ i}(\boldsymbol{x_j}-\boldsymbol{\mu_i})(\boldsymbol{x_j}-\boldsymbol{\mu_i})^T \tag{34}$$

If we sum averaging the dispersions of all classes i we will obtain the global dispersion

$$\sum_i\frac{n_i}{n}\sum_{j\ from\ class\ i}\frac{1}{n_i}\left\|\boldsymbol{y_j}-\boldsymbol{m_i}\right\|^2 = \sum_i\frac{n_i}{n}Tr\left(U^T\mathrm{S}_{intra,i}U\right) \tag{35}$$

$$\sum_i\frac{n_i}{n}\sum_{j\ from\ class\ i}\frac{1}{n_i}\left\|\boldsymbol{y_j}-\boldsymbol{m_i}\right\|^2 = Tr\left(U^T\sum_i\frac{n_i}{n}\mathrm{S}_{intra,i}\,U\right) \tag{36}$$

$$\sum_i\frac{n_i}{n}\sum_{j\ from\ class\ i}\frac{1}{n_i}\left\|\boldsymbol{y_j}-\boldsymbol{m_i}\right\|^2 = Tr(U^T\mathrm{S}_{intra}U) \tag{37}$$

where

$$\mathrm{S}_{intra} = \sum_i\frac{n_i}{n}\mathrm{S}_{intra,i} = \frac{1}{n}\sum_i\sum_{j\ from\ class\ i}(\boldsymbol{x_j}-\boldsymbol{\mu_i})(\boldsymbol{x_j}-\boldsymbol{\mu_i})^T \tag{38}$$

As stated by Fukunaga [40] both matrices are symmetric and are related through the covariance matrix of the data

$$\mathrm{S}_{inter} + \mathrm{S}_{intra} = \mathrm{S} \tag{39}$$

so, if we already know the total covariance, it could be directly calculated from equation (39).

Once both matrices have been calculated we can follow Fisher's criterion [39] which will try to separate the classes as much as possible keeping each of them as compact as possible. In short, we seek to find the conversion matrix U such that it maximizes $U^T\mathrm{S}_{inter}U$ and minimizes $U^T\mathrm{S}_{intra}U$ in other words

$$\arg\max_U\frac{Tr(U^T\mathrm{S}_{inter}U)}{Tr(U^T\mathrm{S}_{intra}U)} \tag{40}$$

Note that in equation (38) the n_i terms cancel, so some authors such as [38] eliminate this term directly from expression (34). Moreover, the division introduced in equation (40) causes the n elements of the summations presented so far to cancel out, so most authors [38, 40] do not introduce the $1/n$ term in equations (26) and (38).

Fukunaga [40] and Wilks [41] provide an expression for the derivative of with respect to a matrix

$$\frac{d}{dA}Tr(A^T RA) = (R + R^T)A \qquad (41)$$

that when applied to the expression to be optimized leads to the following condition

$$Tr(U^T \mathrm{S}_{intra} U) \cdot (\mathrm{S}_{inter} + \mathrm{S}_{inter}^T)U - Tr(U^T \mathrm{S}_{inter} U) \cdot (\mathrm{S}_{intra} + \mathrm{S}_{intra}^T)U = 0 \qquad (42)$$

Considering that S_{inter} and S_{intra} are symmetrical, we can rewrite it as

$$Tr(U^T S_{intra} U) \cdot 2S_{inter} U - Tr(U^T S_{inter} U) \cdot 2S_{intra} U = 0 \qquad (43)$$

which can be reformulated [38] as

$$S_{intra} U = kS_{inter} U \qquad (44)$$

The solution to (44) can be obtained by calculating the eigenvectors associated with the eigenvalues of the matrix $W = \mathrm{S}_{intra}^{-1} \cdot \mathrm{S}_{inter}$ whenever S_{intra} is not a singular matrix, or alternatively, the eigenvectors of $W = \mathrm{S}_{inter}^{-1} \cdot \mathrm{S}_{intra}$ whenever S_{inter} is not singular [38, 42].

Since equation (16) establishes a dependence relationship between $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}$ only $c - 1$ of the $\boldsymbol{\mu}_i$ are independent at most, so the maximum rank of the matrix S_{inter} will be $c - 1$. Since multiplication between matrices is an operation that preserves linear dependencies, the rank of $\mathrm{S}_{intra}^{-1}\mathrm{S}_{inter}$ matrix will also consequently be at most $c - 1$. This means that at most $c - 1$ eigenvalues of $\mathrm{S}_{intra}^{-1}\mathrm{S}_{inter}$ will be nonzero and the rest will be zero [40].

Those eigenvectors associated with a larger eigenvalue will be the projections in which the distances between classes are larger and the dispersion of each class smaller. If the total separability is represented by the sum of all $\lambda_t = \lambda_1 + \cdots + \lambda_m$, the coefficient λ_i/λ_t measures how much of the total separability is captured by the ith component. As we did with the PCA technique we will choose to approximate the projections to the number of vectors that retain a higher separability (usually higher than 90%, 95% or even 99% of total separability).

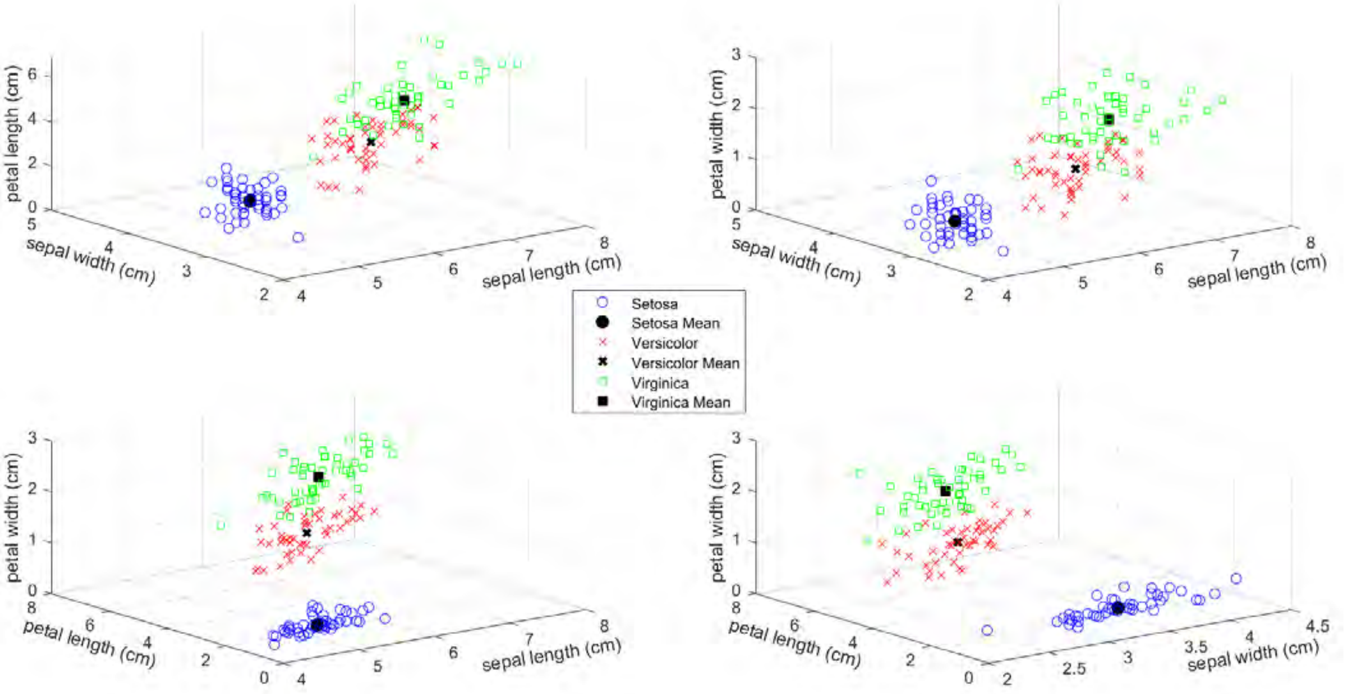


Figure 7. Distribution of the original Fisher data by categories according to the original four features. [39]

As an example of the technique, Fischer's discriminant has been applied to the original Fisher data [39] shown in Figure 7. Since the original data are a function of four different features, the Figure 7 shows the data with respect to the different possible triplets on the axes. The first component represented by the vector

$$\begin{bmatrix} \text{sepal length} \\ \text{sepal width} \\ \text{petal length} \\ \text{petal width} \end{bmatrix} = \begin{bmatrix} -0.21 \\ -0.39 \\ 0.55 \\ 0.72 \end{bmatrix}$$

already retains 99.24% of the total separability. This allows a clear differentiation of the classes on a single feature according to their closeness to the centroids, as shown in Figure 8.

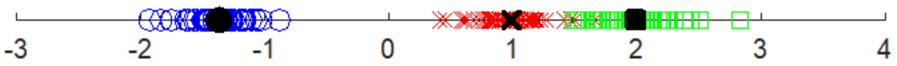


Figure 8. Fisher's data [39] representation on the first main feature after applying LDA.

Starting from a data matrix X as usually presented, consisting of r samples arranged in rows, and their respective labels L identifying the category to which they belong (one of the possible $\{C_1, C_2, \dots, C_c\}$ classes)

$$X = \begin{bmatrix} \boldsymbol{x_1}^T \\ \boldsymbol{x_2}^T \\ \vdots \\ \boldsymbol{x_r}^T \end{bmatrix} \qquad L = \begin{bmatrix} l_1 \\ l_2 \\ \vdots \\ l_r \end{bmatrix}$$

the steps to apply the algorithm are as follows:

Step 1. For each class C_i with $i = \{1, \dots, c\}$, taking only the samples belonging to that class, we will calculate its mean $\boldsymbol{\mu_i}$, a $1 \times m$ vector.

Step 2. We will also calculate the mean of all data $\boldsymbol{\mu}$, a $1 \times m$ vector.

Step 3. We will calculate the distance matrix between classes as:

$$\mathrm{S}_{inter} = \sum_i \frac{n_i}{r} (\boldsymbol{\mu_i} - \boldsymbol{\mu})^T (\boldsymbol{\mu_i} - \boldsymbol{\mu}) \tag{45}$$

where n_i is the number of samples of each class i.

Step 4. We will calculate the internal dispersion matrix of each class i:

$$\mathrm{S}_{intra,i} = \sum_{j\ from\ class\ i} (\boldsymbol{x_j} - \boldsymbol{\mu_i})^T (\boldsymbol{x_j} - \boldsymbol{\mu_i}) \tag{46}$$

Step 5. We will calculate the total dispersion matrix as the sum of the individual ones:

$$\mathrm{S}_{intra} = \frac{1}{r} \sum_i \mathrm{S}_{intra,i} \tag{47}$$

Step 6. We will calculate the matrix to optimize as:

$$W = S_{intra}^{-1} S_{inter} \tag{48}$$

Step 7. We will obtain the eigenvalues and associated eigenvectors of W.

Step 8. We take the n eigenvectors of W associated to the largest eigenvalues. With these n eigenvectors we will form the new basis of smaller dimensions. The rest of the information associated to the smaller eigenvalues will be lost. Having obtained the eigenvalues in descending order, the first n columns of the matrix U will be taken directly to get $U_{reduced}$. The number of eigenvalues will not be very large, since having few categories, the number of non-zero eigenvalues will be limited to $c-1$.

Step 9. We will obtain the transformed data in the new vector space. For the training dataset we will have $X_{reduced} = X \cdot U_{reduced}$. For any new data we should enter $\boldsymbol{x_{reduced}}^T = \boldsymbol{x}^T \cdot U_{reduced}$ with $\boldsymbol{x_{reduced}}^T$ being of dimension $1 \times n$, $\boldsymbol{x}^T$ of $1 \times m$ and $U_{reduced}$ of $m \times n$.

If we wished, exceptionally, to calculate the projection of the data into the new vector space, the approximation of the reduced data, we could do so by applying the equation $X_{aprox} = X_{reduced} \cdot {U_{reduced}}^T$.

Once the features have been reduced, LDA tries to classify each of the samples assuming that the probability distribution $P(\boldsymbol{y}|C_j)$ from equation (4) is multivariate Gaussian with mean $\boldsymbol{m_j}$ and that the distributions of the different classes all have the same covariance matrix, which is $\mathrm{S'}_{intra} = U^T \mathrm{S}_{intra} U$. In that case,

$$P(C_j|\boldsymbol{y}) = \frac{e^{-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{m_j})\mathrm{S}'_{intra}{}^{-1}(\boldsymbol{y}-\boldsymbol{m_j})^T}}{2\pi^{n/2}|\mathrm{S}'_{intra}|^{1/2}} \mathrm{P}(C_j) \tag{49}$$

and taking logarithms we obtain expressions that are easier to evaluate, which are the so-called Fischer discriminant functions:

$$\log P(C_j|\boldsymbol{y}) = -\frac{1}{2}(\boldsymbol{y}-\boldsymbol{m_j})\mathrm{S}'_{intra}{}^{-1}(\boldsymbol{y}-\boldsymbol{m_j})^T + \log P(C_j) - \frac{1}{2}\log|\mathrm{S}'_{intra}| - \frac{n}{2}\log 2\pi \tag{50}$$

The last step is to calculate these functions for the different categories and by means of equation (1) compare the different values taken by $R(C_i|\boldsymbol{y})$ for each i, choosing that class that makes it minimum.

Remember that if the loss function is that of equation (2), the most direct route is to find that category that maximizes $P(C_j|\boldsymbol{y})$, i.e., that maximizes $\log P(C_j|\boldsymbol{y})$. Since S'_{intra} and $2\pi^{n/2}$ have the same values for all classes C_j, their associated terms will be constant in each and every category and it will not even be necessary to calculate them since in the comparison between the different classes they will appear equally in all of them, contributing the same. Therefore, in the case where the loss function is

$$L(C_i, C_j) = \begin{cases} 0 \; si \; j = i \\ 1 \; si \; j \neq i \end{cases} \tag{51}$$

discriminant functions in the LDA classifier can be reduced to

$$\log P(C_j|\boldsymbol{y}) = -\frac{1}{2}(\boldsymbol{y}-\boldsymbol{m_j})\mathrm{S}'_{intra}{}^{-1}(\boldsymbol{y}-\boldsymbol{m_j})^T + \log P(C_j) \tag{52}$$

and predict the category that maximizes the expression.

The decision boundaries between two categories can be found by equating the discriminant functions of both classes. By making $\log P(C_j|\boldsymbol{y}) = \log P(C_i|\boldsymbol{y})$, developing the term $-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{m_j})\mathrm{S}'_{intra}{}^{-1}(\boldsymbol{y}-\boldsymbol{m_j})^T = -\frac{1}{2}\boldsymbol{y}\mathrm{S}'_{intra}{}^{-1}\boldsymbol{y}^T + \boldsymbol{y}\mathrm{S}'_{intra}{}^{-1}\boldsymbol{m_j}^T - \frac{1}{2}\boldsymbol{m_j}\mathrm{S}'_{intra}{}^{-1}\boldsymbol{m_j}^T$ and eliminating from both parts of the equation the common elements, we obtain

$$\boldsymbol{y}\,\mathrm{S}'_{intra}{}^{-1}\boldsymbol{m_j}^T - \frac{1}{2}\boldsymbol{m_j}\,\mathrm{S}'_{intra}{}^{-1}\boldsymbol{m_j}^T + \log P(C_j) = \boldsymbol{y}\,\mathrm{S}'_{intra}{}^{-1}\boldsymbol{m_i}^T - \frac{1}{2}\boldsymbol{m_i}\,\mathrm{S}'_{intra}{}^{-1}\boldsymbol{m_i}^T + \log P(C_i) \tag{53}$$

Putting the terms in order

$$\boldsymbol{y}\,\mathrm{S}'_{intra}{}^{-1}\left(\boldsymbol{m_j}-\boldsymbol{m_i}\right)^T=\frac{1}{2}\boldsymbol{m_j}S'_{intra}{}^{-1}\boldsymbol{m_j}{}^T-\frac{1}{2}\boldsymbol{m_i}S'_{intra}{}^{-1}\boldsymbol{m_i^T}+\log\frac{P(C_i)}{P(C_j)}\qquad(54)$$

Having diagonalized S'_{intra} into these new features, we can apply the simplification $\frac{1}{2}\boldsymbol{m_j}\,\mathrm{S}'_{intra}{}^{-1}\boldsymbol{m_j}{}^T-\frac{1}{2}\boldsymbol{m_i}\,\mathrm{S}'_{intra}{}^{-1}\boldsymbol{m_i^T}=\frac{1}{2}\left(\boldsymbol{m_j}+\boldsymbol{m_i}\right)\mathrm{S}'_{intra}{}^{-1}\left(\boldsymbol{m_j}-\boldsymbol{m_i}\right)^T$ leaving the equation as

$$\boldsymbol{y}\,S'_{intra}{}^{-1}\left(\boldsymbol{m_j}-\boldsymbol{m_i}\right)^T=\frac{1}{2}\left(\boldsymbol{m_j}+\boldsymbol{m_i}\right)S'_{intra}{}^{-1}\left(\boldsymbol{m_j}-\boldsymbol{m_i}\right)^T+\log\frac{P(C_i)}{P(C_j)}\qquad(55)$$

Finally, grouping with respect to the common

$$\left(\boldsymbol{y}-\frac{\boldsymbol{m_j}+\boldsymbol{m_i}}{2}\right)\mathrm{S}'_{intra}{}^{-1}\left(\boldsymbol{m_j}-\boldsymbol{m_i}\right)^T=\log\frac{P(C_i)}{P(C_j)}\qquad(56)$$

The equation represents those points where the orthogonal projection of $\left(\boldsymbol{y}-\frac{\boldsymbol{m_j}+\boldsymbol{m_i}}{2}\right)$ onto $\left(\boldsymbol{m_j}-\boldsymbol{m_i}\right)$ averaged with the values of the diagonal $\mathrm{S}'_{intra}{}^{-1}$ is constant, i.e., it represents a hyperplane perpendicular to the segment joining both centroids. The projection, measured from the midpoint of the centroids, defines the location of the hyperplane where this takes the value $\log P(C_i)/P(C_j)$. In the particular case where $P(C_i)=P(C_j)$ and consequently the plane will be equidistant to both centroids cause $\log P(C_i)/P(C_j)=0$.

Fischer's model shows problems if the classes are not linearly separable, that is, when the information is more contained in the dispersion of the data than in their mean representatives. If the means of the different classes are similar S_{inter} will be almost zero, as will W. As a solution, transformations using nonlinear functions that map the original data to a higher dimensional space where the distance between mean representatives is relevant can be used [43, 44]. Gaussian and radial basis functions are the most commonly used in these cases.

Another potential problem arises when the $m\times m$ matrix S_{intra} is singular. The rank, number of independent columns, of the S_{intra} matrix is limited by the dependencies introduced by equations (15), (16) and (34). Each category introduces at least one dependency relationship between its n_i equations. For c categories together, among r equations there will be at least c of them dependent. Thus, we can take $r-c$ as an upper bound of the rank of S_{intra}. If the number of original features is much larger than the number of samples, $m\gg r-c$ the matrix will certainly be singular [45, 46]. One potential solution that we have discussed previously is to find the solution to equation (44) through the eigenvectors of the $\mathrm{S}_{inter}^{-1}\mathrm{S}_{intra}$ matrix if the S_{inter} matrix is invertible. Another possibility is to apply a regularization factor [47] to the matrix by making $\mathrm{S}_{intrareg}=\mathrm{S}_{intra}+\eta I$ but this method has no clear mathematical explanation and the proper choice of η can be costly. Other authors propose to remove the null space from the matrix to make it invertible [48] or to apply PCA previously LDA to reduce the number of features before applying LDA. m of features before applying LDA.

As it happened with the PCA technique, since LDA is a technique that creates new features from linear combinations of the real features, it can obscure and hinder the interpretation of the decisions made in the real problem when we want to interpret them on the basis of the original features.

Depending on the classification technique used, its use is not always recommended, especially with MLP where it is preferable that the ML technique itself has all the possible information, without simplifications, and that it is its training that discovers the optimal variables for classification. In other cases applying LDA may offer improvements in subsequent classification. Again, it is advisable to always test the learning algorithm without applying LDA and then apply LDA to compare the results.

To conclude, QDA is a technique similar to LDA but instead of assuming the same covariance matrix for all classes, it uses a different covariance matrix specific to each class. That is, it assumes that the probability distribution $P(\boldsymbol{x}|C_j)$ of equation (4) is multivariate Gaussian with mean $\boldsymbol{\mu_j}$ and that the distributions of the different classes have different covariance matrices $\mathrm{S}_{intra,j}$.

The corresponding equations are similar to equations (49) and (50) but taking the class-specific covariance matrix instead of the common one. This time some simplifications cannot be made and the full expressions should be used, except for the factor $2\pi^{n/2}$, which can be eliminated from the calculation for standard loss functions. This leaves the equations as

$$P(C_j|\boldsymbol{x}) = \frac{e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_j})\mathrm{S}_{intra,j}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_j})^T}}{2\pi^{n/2}|\mathrm{S}_{intra,j}|^{1/2}}\mathrm{P}(C_j) \tag{57}$$

$$\log P(C_j|\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)\mathrm{S}_{intra,j}{}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_j})^T + \log P(C_j) - \frac{1}{2}\log|\mathrm{S}_{intra,j}| - \frac{n}{2}\log 2\pi \tag{58}$$

In this case, the orthogonalization of each class separately will produce a different set of eigenvalues and eigenvectors, U_j for each class j, achieving a non-common transformation space that can provide us with new features with a simpler classification for class j over the rest. The adaptation of Fischer's method that instead of optimizing S_{intra} seeks to optimize $\mathrm{S}_{intra,j}$ from equation (34) is made through

$$\arg\max_{U_j} \frac{U_j^T \mathrm{S}_{inter} U_j}{U_j^T \mathrm{S}_{intra,j} U_j} \tag{59}$$

It is known as the class-dependent Fischer method and similarly to the class-independent Fischer method it is achieved by solving the equation

$$\mathrm{S}_{intra,j} U_j = k\mathrm{S}_{inter} U_j \tag{60}$$

which can be obtained following the same idea of Fischer's method, that is, calculating the eigenvectors associated to the eigenvalues of the matrix $W_j = \mathrm{S}_{intra,j}^{-1} \cdot \mathrm{S}_{inter}$.

This time, the steps to be followed starting from the data matrix X made up of r samples arranged in rows with their respective labels identifying the categories $\{C_1, C_2, \dots, C_c\}$ will be:

Step 1. For each class C_i with $i = \{1, \dots, c\}$, we will calculate its mean $\boldsymbol{\mu_i}$ taking only the samples belonging to that class. $\boldsymbol{\mu_i}$ is a $1 \times m$ vector.

Step 2. We will also calculate the mean of all data $\boldsymbol{\mu}$, a $1 \times m$ vector.

Step 3. We will calculate the distance matrix between classes

$$\mathrm{S}_{inter} = \sum_i \frac{n_i}{r}(\boldsymbol{\mu_i} - \boldsymbol{\mu})^T(\boldsymbol{\mu_i} - \boldsymbol{\mu}) \tag{61}$$

where n_i is the number of samples in each class i.

Step 4. We will calculate the internal dispersion matrix of each class i

$$\mathrm{S}_{intra,i} = \sum_{j\ from\ class\ i} \left(\boldsymbol{x_j} - \boldsymbol{\mu_i}\right)^T \left(\boldsymbol{x_j} - \boldsymbol{\mu_i}\right) \tag{62}$$

Step 5. We will calculate the different matrices

$$W_i = \mathrm{S}_{intra,i}^{-1} \mathrm{S}_{inter} \tag{63}$$

Step 6. We will obtain the eigenvalues and eigenvectors associated with W_i.

Step 7. Finally, we can sort the eigenvalues and take the n eigenvectors of W_i associated to the largest eigenvalues to form the new basis U_i. The number of non-zero eigenvalues will be less than or equal to $c - 1$.

If we wished, exceptionally, to calculate the projection of the data in the new vector space, the approximation of the reduced data, we could do so through the expression $X_{aprox} = X_{reduced} \cdot U_{i,reduced}$.

2.4 kNN

The kNN algorithm is a simple supervised learning algorithm currently widely used for classification [33], although there is a variant applicable for regression [49]. The algorithm directly constructs the decision regions separating each class without first estimating the density function. Under certain assumptions both ways of obtaining a classifier are equivalent. Estimating first the probability functions and then establishing the boundaries or directly establishing the boundaries (which will correspond to a type of probability function) are two sides of the same coin, two interpretations that under certain assumptions tell us about the same technique sometimes from a statistical point of view and sometimes from a more algorithmic point of view.

To build the classification regions, the algorithm stores all the training cases and compares the distances between the new sample to be classified and each and every one of the training cases. It thus obtains an ordering with the k nearest neighbors to the data to be classified and chooses for the sample the most probable classification category C_i among those neighbors (in case it wants to be used for regression, it will assign to the sample the mean of the values of its neighbors).

The most important parameter in classifier design is the integer k. A good starting approximation can be $k \approx \sqrt{n}$, where n is the number of training cases. Large values of k will reduce the effect of noise on classification but at the cost of increasing bias. Values that are too small will produce overly specific decision regions that are very sensitive to noise from examples that are out of place and with reduced generalizability.

The metric used to calculate the distances must be such that two elements that are more similar to each other will have smaller distances from each other than those elements that are more different. If this condition is fulfilled, and although the shape of the decision regions created is dependent on the metric used, the results are usually equally valid regardless of the metric. Two of the most commonly used metric distances are the Euclidean distance between two points and the Manhattan distance.

Metrics must somehow use all the features of the data, all the attributes, which entails a risk. If the attributes are not normalized or if there are many irrelevant attributes, they can dominate the ranking. In the first case the features with larger values would have more weight with respect to those measured with larger scales (and therefore smaller values). In the second case, two relevant features would lose weight among many irrelevant ones, because if in principle they all weigh the same, each of them contributes the same. To correct the possible bias, a weight can be assigned to the distance component of each attribute, thus giving greater importance to the most relevant attributes. Unfortunately, this implies that the weights become new design parameters, increasing the complexity of the classifier. The Mahalanobis distance metric assigns as weights the values of the covariance matrix to do this. A good compromise for assigning the weights is to normalize the data and rely on the eigenvalue relationships obtained through the PCA technique.

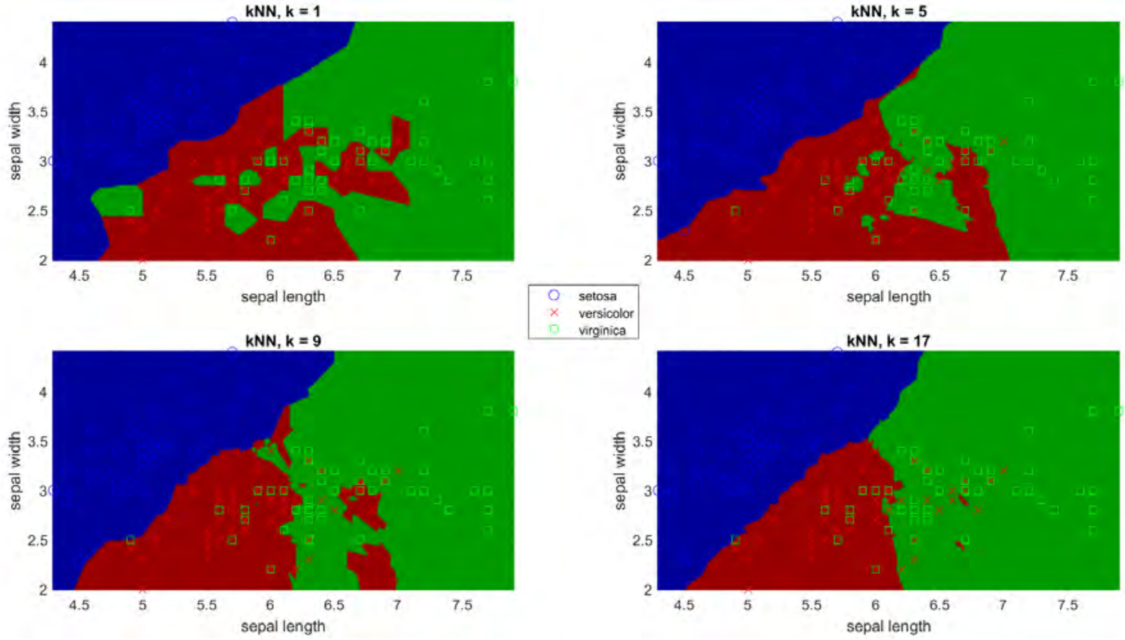


Figure 9. Decision regions applying the kNN algorithm with Euclidean distances on Fisher's original data [39].

Zhenxing et al. [50] propose a variant where the contribution of each neighbor is weighted according to the distance between it and the sample to be ranked, giving more weight to the closest neighbors. For example, we can weight the vote of each neighbor according to the inverse square of their distances. With high values of k this can serve to regularize the influence of the more distant values and can be especially indicated in data samples with different numbers of samples for each category.

In addition, taking one of the ideas proposed by these same authors we can built a kNN classifier that admits a non-standard loss matrix. After calculating the distances with respect to the sample to be classified, ordering them from smallest to largest and taking the k nearest neighbors, we can understand the number of nearest neighbors of each category as an estimate of the probability $P\left(C_j\middle|\boldsymbol{x}\right)$ to subsequently apply equation (1) with the corresponding loss matrix and take that classification category that represents a lower risk.

2.5 Classification and Regression Trees (CART)

Classification and regression trees are an algorithm belonging to the BC family that shows the decisions made on the values of the different features in its goal of classifying the data in an easily interpretable way.

CART understand classification as a process in which consecutive decisions are made. The decisions to be made in the process are always binary and are represented as nodes with two possible output paths. Each division of the nodes will depend on the value of a single characteristic. For numerical features we will form an ordering with their possible values. Binary questions will be of the type "is $x < c$?", with c being each of those possible values to be set during machine training. For non-numeric features, labeled $\{b_1, b_2, \dots, b_L\}$ we will form all possible sets with the categories b_l to produce binary questions of the type "$x \in S$?", with S being each of the possible sets[7].

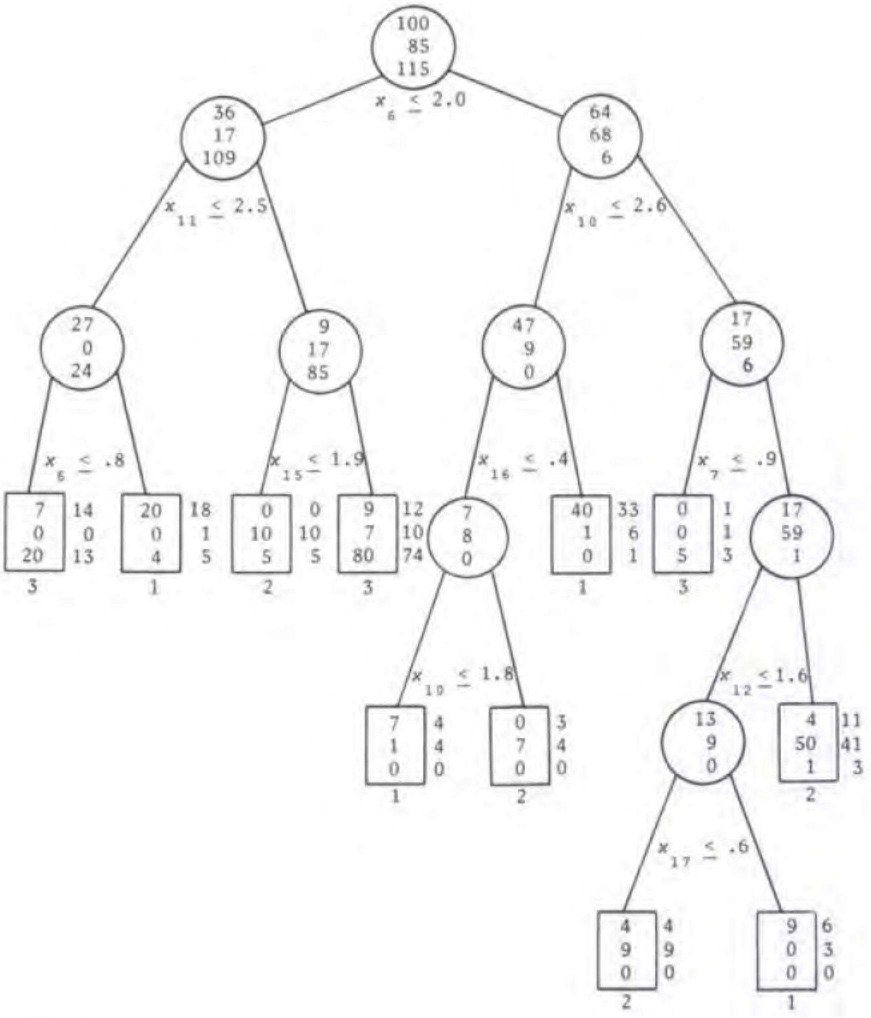


Figure 10. Example of CART visualization showing the splitting questions, training and test populations by category in each node and class assigned to each terminal node. [1]

The classification process as a whole can be represented by a tree graph, with a first and unique arrival of all the data to a first decision node that branches more and more as we make more decisions. Therefore, the more decisions we have in the process, the more branches that we have in the tree. The values of the features of the data presented at the input to the classifier will mark a path through the branches of the tree that will end in one of its final branches (also called leaves).

[7] For L options the number of subsets to be searched grows in the order of 2^L, so it is computationally interesting to keep the possible number of options of any such feature at a moderate value. However, in case there are only two final categories of classification of the data it is possible to reduce the search to a smaller number of subsets of the order of L. The procedure that facilitates handling as many categories as desired in two-class problems is given in Annex II: Optimized search on categorical features.

A particularly useful way to visualize the data and the tree structure is the one in Figure 10. Inside each node (including terminal nodes) is shown the training population for each category (in this case with 3 classes and 300 training cases in total). Below each node is the question used to create the division by taking the left branch in case of affirmative answers, and the right branch in case of negative answers. The terminal nodes are marked by rectangles, below them the assigned category is shown, and to their right the test population for each category that ended at that node is shown for comparison with the training population.

To each leaf or end node t of the tree we can associate a vector of probabilities $\boldsymbol{p}$ where each $p_j = P(C_j|t)$ represents the probability that an element in that node is classified in the class $C_j \in \{C_1, \dots, C_c\}$. Thus, $p_j \geq 0$ and $\sum_{j=1}^{c} p_j = 1$. Once we arrive at that final node t we will assign it a classification category C_{j^*} based on the different probabilities p_j. The assigned class C_{j^*} at a terminal node t is the one that minimizes the misclassification risk[8] at the node, i.e., the one that minimizes $r(t) = \sum_{j=1}^{c} L(C_i, C_j) P(C_j|t)$. In the case of the standard loss tree, this is achieved by accumulating the lowest $P(C_i|t)$ in the rate, i.e., assigning to the node the class $P(C_{j^*}|t)$ is the largest: $r^*(t) = \min_{C_{j^*}^{\ *}} r(t) = \min_{C_{j^*}} \sum_{i \neq j^*} P(C_i|t) = 1 - \max_{C_{j^*}} P(C_{j^*}|t)$. If there is more than one maximum any of the classes that produce it can be taken.

If all the data features are numerical, we can interpret that each time the tree performs a division at a node, it performs a rectangular partition[9] in the data space. Thus, the whole tree would divide the data space into rectangles of different sizes, assigning each space a class.

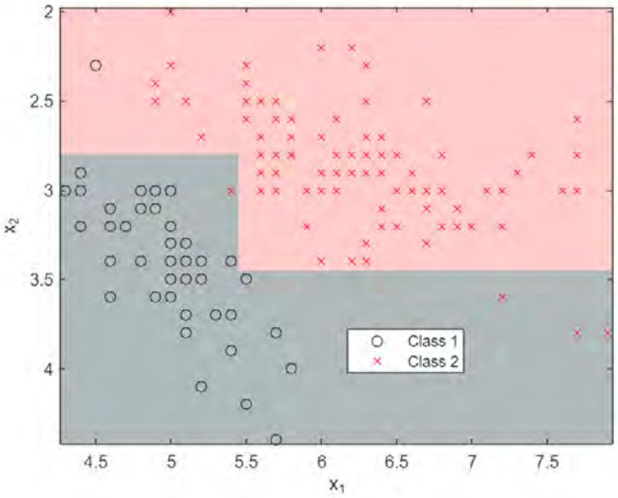


Figure 11. Numerical feature space partitioning using CART. [1]

To among all possible decisions determine which is the best decision we can make Breiman et al. [1] propose to take a measure of node indecision, an impurity function $\phi(\boldsymbol{p})$ that satisfies the following 4 properties:

i. ϕ is maximum only at the point $\left(\frac{1}{c}, \frac{1}{c}, \dots, \frac{1}{c}\right)$.
ii. ϕ is minimum only at the points $(1,0, \dots, 0), (0, 1, \dots, 0), \dots, (0, 0, \dots, 1)$.
iii. ϕ is symmetrical with respect to the p_j.

[8] Remember the equation (1) in the BC.

[9] By rectangular we mean that the divisions occur on hyperplanes orthogonal to the features defining the boundary.

iv. ϕ is strictly concave.[10]

If a node t is a terminal node, with a probability of being reached $P(t)$, the indecision measure can be computed as

$$i(t) = \phi\big(P(C_1|t), P(C_2|t), \dots, P(C_c|t)\big) \tag{64}$$

If instead of using that node t as the final node, we chose to replace it by an intermediate decision node (also t) by means of a decision s, it would send a proportion $p_R = P(t_R)/P(t)$ of the data to the right and a proportion $p_L = P(t_L)/P(t)$ to the left, so that both $p_R + p_L = 1$. The change in the total tree impurity introduced by this split would be defined by

$$\Delta i(s,t) = i(t) - p_R i(t_R) - p_L i(t_L) \tag{65}$$

where $i(t_R), i(t_L)$ would be the indecisions of the new final nodes created under the new t. Emphasize that as a consequence of choosing ϕ with the indicated properties, it will always occur that $\Delta i(s,t) > 0$.

The latter $\Delta i(s,t)$ will be taken as a measure of how good a splitting criterion is, since selecting those splits that contribute to maximize it is equivalent to taking those splits that decrease the total indecision of the tree, calculated as $i_{Total} = \sum_{t \in terminal} i(t)P(t)$.

Experimental data collected by Breiman et al. [1, 2] seem to indicate that the trees created are not very sensitive to changes in the impurity function. Three possible functions used by them are:

a) The entropy function: $i(t) = -\sum_j P\big(C_j\big|t\big)\log P\big(C_j\big|t\big)$.

b) The Gini diversity index: $i(t) = \sum_{i \neq j} P(C_i|t)P\big(C_j\big|t\big) = 1 - \sum_j P^2\big(C_j\big|t\big)$. It is simple and easily computable. Moreover, it can be interpreted as the misclassification rate if instead of assigning each node always always its most likely category, we assign the category randomly over the probabilities $P\big(C_j\big|t\big)$.

c) The rule of classification into two superclasses or *twoing rule*:

$$\Delta i\big(s,t,C_1(s)\big) = \frac{p_L \cdot p_R}{4}\left[\sum_{C_j}\big|P\big(C_j\big|t_L\big) - P\big(C_j\big|t_R\big)\big|\right]^2 \text{ with } C_1(s) = \big\{C_j : P\big(C_j\big|t_L\big) \geq P\big(C_j\big|t_R\big)\big\} \tag{66}$$

It separates the classes of a node into two superclasses, and computes Δi as if it were a binary problem[11] . It provides information on the similarities between classes, especially in the groupings performed in the upper nodes, since in the first divisions it will try to group in a superclass a large number of classes that are similar in some characteristics, and in the lower nodes it will try to isolate classes individually. The more classes the problem has, the more information the *twoing rule* will provide.

The choice of one function or another depends on the problem and what information is to be interpreted in the divisions. Gini is often preferred to entropy because of its greater ease of computation. In general, both Gini and the *twoing rule* produce similar splits with similar classification rate trees. In those splits where they differ, Gini tends to favor splits that produce a smaller, purer node and another, more impure node, while the *twoing rule* tends to prefer splits that equalize populations at their descendant nodes. Where they differ, the Gini criterion splits tend to be somewhat better so that the Gini criterion is usually preferred over twoing [1].

[10] This condition is introduced by Breiman et al. [1] as a solution to avoid the case $\Delta i = 0$ for all possible splits that occurred when $P\big(C_j\big|t_D\big) = P\big(C_j\big|t_I\big) = P\big(C_j\big|t\big)$.

[11] For further details see Annex I.

As a measure of the classification goodness of the tree as a whole we will use a variant of the misclassification rate that also takes into account the complexity of the tree, rewarding smaller trees (and thus offering a solution to the overtraining of the tree). If we take as a measure of complexity the number of terminal nodes of the tree, $|\tilde{T}|$ we can define

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}| \tag{67}$$

where $\alpha \geq 0$ is the complexity parameter and

$$R(T) = \sum_{j=1}^{c} L(C_{j^*}, C_j)Q(C_{j^*}, C_j)P(C_j) = \sum_{t \in terminal} P(t)r^*(t) \tag{68}$$

where, in turn, $L(C_{j^*}, C_j)$ represents the losses when deciding to classify an element as C_{j^*} when it actually belongs to class C_j, and $Q(C_{j^*}, C_j)$ is the proportion of cases of class C_j classified as C_{j^*}. An important property of $R(T)$ is that the more the tree is split the lower is its total rate, or in other words, $R(t) \geq R(t_L) + R(t_R)$.

A tree with too many levels minimizes the rate $R(T)$ but will have a higher rate $R_\alpha(T)$ than a tree with the appropriate number of levels. A tree with fewer levels than adequate would not use some of the available information, also resulting in a higher misclassification rate. To determine the depth of the tree Breiman et al. [1] finally choose to let the tree grow to high levels and then prune it, following a sequence of nested trees, looking for the best misclassification rate $R_\alpha(T)$ on the training data.

Thanks to the properties of $R_\alpha(T)$, if we search for the tree that for a given α minimizes $R_\alpha(T)$ we will have found the smallest tree that minimizes $R_\alpha(T)$, since its various trees are nested (see demonstration in Annex 10.3). Thus, if for a given α the tree obtained has, for example, 7 terminal nodes, there will be no other tree with 7 terminal nodes with a lower $R(T)$. The trees thus obtained are guaranteed to have the lowest possible $R(T)$ fixed by their number of nodes, among all possible trees with that number of nodes.

How far to let the tree grow is not critical, as long as it is allowed to grow long enough. Because pruned trees are nested, trees that have grown beyond the proper size will, when pruned, produce the same final tree. The growth phase will usually continue until one of the following conditions is met:

i. Until all terminal nodes are pure, with all cases of the same class inside every terminal node.
ii. Until all terminal nodes have less than N_{min} cases. A typical value for the parameter [1] is usually $N_{min} = 5$. If N_{min} is too small (the extreme case is $N_{min} = 1$) several of the $P(C_j|t)$ will be 0 or 1, and $R(T)$ will probably be 0. This can lead to very poor estimates when applying the test data, with $R^{test}(T) \gg R(T)$, cause the trees would have more levels than the training data can warrant.
iii. Define a threshold $\beta > 0$ and stop splitting a node (and take it as a terminal node) if none of its splits improves that threshold, i.e., if in all splits $\max_{s \in S} \Delta i(s,t) < \beta$. This means that the space is so homogeneous that any partition will contribute nothing.
iv. Up to a maximum number of subdivisions k_{max}.

For each value of α we will try to find the tree pruning $T(\alpha)$ that minimizes $R_\alpha(T)$. As we have already said, in this process, thanks to the properties of $R_\alpha(T)$, for each value of α there is only a single pruning that minimizes $R_\alpha(T)$ and moreover the successive prunings of the ordering are nested. We start from the tree $T_{max} = T_0$ which corresponds to an $\alpha = 0$. Each of the k steps to be taken during the process consists of:

1. For each non-terminal node t of the tree we calculate
$$g(t) = \frac{r(t) - R(T_t)}{|\tilde{T}| - 1} \tag{69}$$
$R(T_t)$ being the sum of $r(t)$ of the terminal nodes of the branches below the node t, and $|\tilde{T}|$ the number of terminal nodes below it.

2. We increase α to a value $\alpha_k = \min_i g(t_i)$ of all possible t_i non-terminal nodes and prune the tree by node t_i getting the new tree T_k.
3. Iterate over the new tree until only the root node remains.

The sequence of trees thus obtained is guaranteed to be a sequence where, for each number of stablished terminal nodes we will have the tree with the lowest possible $R(T)$. Moreover, as Breiman et al. [1] point out, the algorithm tends initially to prune longer branches with many terminal nodes and finally shorter branches.

Once we have this sequence of trees, we will calculate the error $R_{cv}(T)$ on the cross-validation data (using equation (68) on the validation data) for each of the trees in the sequence, finally choosing a tree with the lowest possible $R_{cv}(T)$.

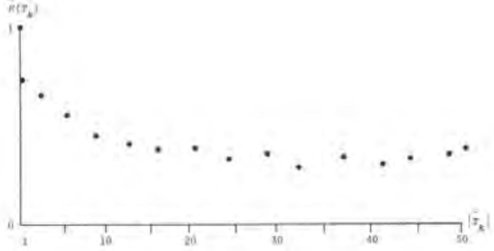


Figure 12. Example of graph $R_{cv}(T_k(\alpha))$. [1]

Since the auxiliary trees that have been created in the cross-validation process are created with fewer training examples, they will tend to be less accurate. Estimates through cross-validation therefore tend to be conservative, overestimating the misclassification cost. Experience with classification problems usually shows plots of $R_{cv}(T_k(\alpha))$ as a function $|\widetilde{T_k}|$ like that in Figure 12, with a fairly rapid initial decrease followed by a flat valley and then a gradual increase for large $|\tilde{T}_k|$. The minimum occurs somewhere in the valley, where $R_{cv}(T_k(\alpha))$ is nearly constant except for changes in a range[12] limited by the number of samples used in validation [1].

The position of the minimum within the valley can be unstable in case of small changes in the parameter values or in the distribution of the examples. Looking for the simplest solution within this instability, we choose to search among those that are within the valley and have a smaller number of terminal nodes.

[12] Breiman et al [1] state that the probability that a single case among L validation data is misclassified has $p = R_{cv}(T(\alpha))$ (measured value). Under this view we have a binomial situation of L independent attempts (test cases), with probability of success p in each attempt. The proportion of successes over the total would be our estimated variable. $R_{cv}(T(\alpha))$. The binomial distribution provides us with the standard error, $\sqrt{p(1-p)/L}$. Therefore, the error range would be $SE(R_{cv}(T)) = \sqrt{R_{cv}(T)(1 - R_{cv}(T))/L}$.

It is therefore very useful to display during training and tree choice the following aspects: the iteration number k, the tree size $|\widetilde{T_k}|$, the cumulative error in the training cases $R_\alpha(T)$ and the cumulative error in the validation cases $R_{cv}\big(T_k(\alpha)\big)$.

k	$\lvert\widetilde{T_k}\rvert$	$R_\alpha(T)$	$R_{cv}\big(T(\alpha)\big)$
1	31	0.17	0.30±0.03
2**	23	0.19	0.27±0.03
3	17	0.22	0.30±0.03
4	15	0.23	0.30±0.03
5	14	0.24	0.31±0.03
6*	10	0.29	0.30±0.03
7	9	0.32	0.41±0.04
8	7	0.41	0.51±0.04
9	6	0.46	0.53±0.04
10	5	0.53	0.61±0.04
11	2	0.75	0.75±0.03
12	1	0.86	0.86±0.03

Table 5. Example of visualization of data collected during tree training and pruning . [1]

In Table 5 the minimum $R_{cv}(T)$ is marked with ** and the tree finally selected according to the rule of $\min_k R_{cv}(T_k) + SE$ is marked with *.

In summary, the steps to follow to apply CART are:

Step 1. Group the data by final category and separate 10% of the cases for validation and 90% of the cases for training (maintaining the proportions of the data in each category).

Step 2. Define the indecision function $i(t)$ to be used, usually Gini. Choose the stop-growth parameters β, N_{min} and k_{max}. Their value is usually not influential on the result. Define the loss matrix $L\big(C_i, C_j\big)$.

Step 3. Grow the tree iteratively starting from the first node of which we already know its $P(t)$ its $i(t)$, the number of cases that reach it, both training (those to be used in the iterations) and validation cases.

a. In each final node, the following divisions are proposed:

i. For all numerical variables we propose the divisions that cover the whole range of the variables. That is, for the first variable x_1 we will order its values by obtaining a list of c_i ascending from the minimum value of x_1 to the maximum. The proposed questions will be "$x_1 < c_i$?"

ii. For all non-numeric variables labeled $\{b_1, b_2, \dots, b_L\}$we will form all possible sets S_i with the categories b_l. The binary questions will be of the type "$x \in S_i$?"

b. For each proposed division its $p_L, p_R, i(p_L), i(p_R), p_{Lcv}, p_{Rcv}$ and finally $\Delta i(t)$ will be calculated. The one with the highest $\Delta i(t)$ will be chosen among all the splits. The proposed node t will be split with that splitting and two new final nodes will be created, saving for the newly created nodes the values of $p_L, p_R, i(p_L), i(p_R), p_{Lcv}, p_{Rcv}$ and the number of cases per category reached by each node.

c. The iterations will end when any of the following conditions are met:

i. All terminal nodes contain cases of a single category. These nodes are excluded from the proposed splits.

ii. All terminal nodes contain less than N_{min} cases. These nodes are excluded from the proposed splits.

iii. Nodes where all proposed splits make $\max_{s\in S} \Delta i(s,t) < \beta$. Those nodes are excluded from the proposed splits.

iv. Until a maximum number of iterations k_{max} is reached.

Step 4. Prune the tree into a sequence of nested trees by proceeding in a new iterative process:

a. With the number of cases for each category we assign to each node of the tree the category t of the tree the category C_{j^*} that minimizes the risk $r^*(t)$ calculating:

$$\begin{aligned} r^*(t) &= \sum_{j=1}^{c} L(C_{j^*}, C_j) Q(C_{j^*}, C_j) P(C_j|t) \\ r_{cv}^*(t) &= \sum_{j=1}^{c} L(C_{j^*}, C_j) Q(C_{j^*}, C_j) P_{cv}(C_j|t) \end{aligned} \tag{70}$$

We also calculate the error rates of the whole tree:

$$\begin{aligned} R(T) &= \sum_{t\in terminal} P(t) r^*(t) \\ R_{cv}(T) &= \sum_{t\in terminal} P_{cv}(t) r_{cv}^*(t) \\ SE_{cv} &= \sqrt{R_{cv}(T)\left(1 - R_{cv}(T)\right)/L_{cv}} \end{aligned} \tag{71}$$

b. For each non-terminal node t of the tree we calculate:

$$g(t) = \frac{r^*(t) - R(T_t)}{|\tilde{T}| - 1} \tag{72}$$

where $|\tilde{T}|$ is the number of terminal nodes below it, and

$$R(T_t) = \sum_{u\in terminal\ node\ below\ t} P(u) r^*(u) \tag{73}$$

c. We prune the tree by the node with a lower value $g(t)$.

d. Iterate over the new tree until only the root node remains.

Step 5. Select the tree with the smallest value $\min_k R_{cv}(T_k) + SE$.

Classification trees offer numerous advantages. As we have already seen, the final classification has a simple form that can be stored in a compact form, in addition to providing easily understandable and interpretable information. It offers a natural and enlightening way to understand the structure of the problem and an estimate of the probability of error in the structure.

CART makes use of the information we know through conditional probabilities, a use that manages to handle the non-homogeneous relationships of the data, since it searches each node individually for the most significant division. At each step it selects the features automatically, being able to extract the most relevant information from the portion of the space it is working on.

It is invariant to individual monotonic transformations of the continuous variables [1] with a simple modification of the boundary c values. In terms of the influence of noise and discordant values it is robust since a single example only has the weight of 1 compared to all other N total cases.

2.6 Multilayer Perceptron (MLP)

Ever since in the 1940s McCulloch and Pitts [51, 52] presented their neuron model and Hebb's [53] work on learning provided a theoretical foundation for how real neuronal structures interact, and in the late 1950s Rosenblatt [54, 55] published his learning model for the Perceptron, the models of the basic units of neural networks, the artificial neurons, have followed a similar idea. The fundamental idea of the model is that the neuron receives signals from the axon terminals of other neurons through certain synaptic connections that can amplify or attenuate them. These signals captured in the dendrites are combined and accumulate in the body of the neuron until, upon reaching a certain threshold level, the neuron passes from an inactive to an active state, triggering a signal that travels along the axon and is transmitted through the axon terminals and the corresponding synaptic connections to the next neuron.

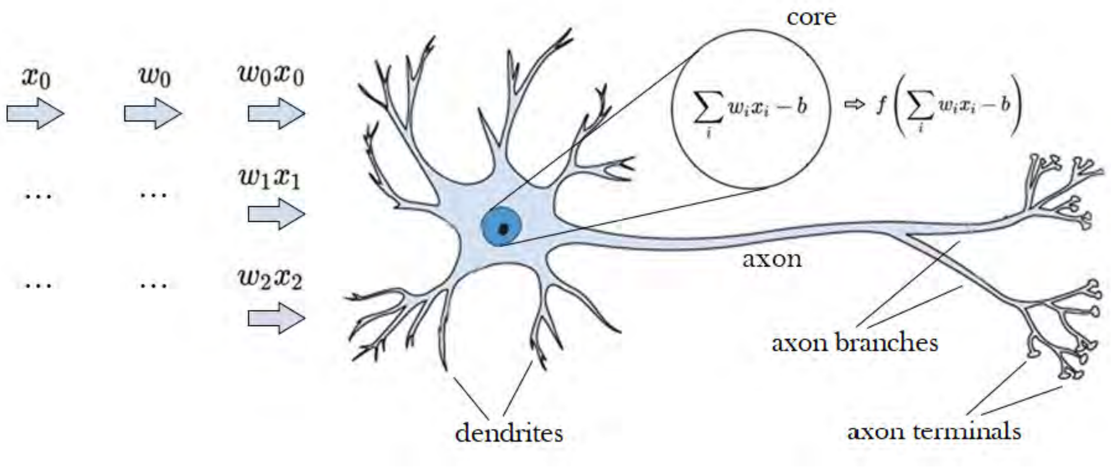


Figure 13. Artificial Neuron Model. [56]

Following these ideas, the artificial neuron based on the biological models shown in Figure 13 is first composed of a series of inputs x_i weighted by different weights w_i whose sum (or linear combination) comes to define the internal level of the neuron, $a = \sum_i w_i x_i$. Each input feature is thus multiplied by a corresponding adjustable weight and subsequently their contributions are summed to determine the neuron's internal level. Moreover, to calculate this internal level, the contribution of an independent term or bias, b, that modifies the base threshold level is also included. This term can be represented directly in the linear combination, without being associated to any input, or in many occasions and for simplicity when implementing the algorithm, it is represented as the weight $w_0 = -b$ (note the negative sign) associated to a unitary input $x_0 = 1$. After calculating the internal level of the neuron, it will be decided whether or not it triggers the signal by applying a certain activation function $z = f(a)$. In the original model this activation function was the step function of Figure 14 that achieves an activated output, $z = 1$, for all those internal levels $a > 0$ and an idle output, $z = 0$, for $a < 0$.

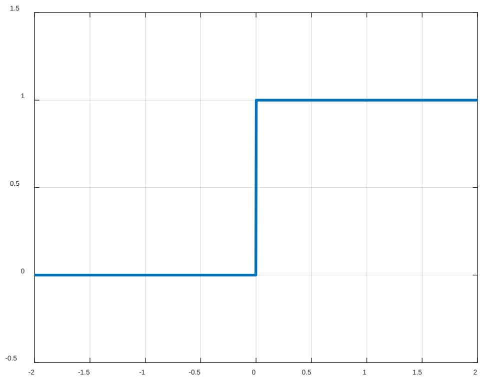


Figure 14. Step activation function.

In these neurons, learning is performed by modifying the values of the weights w_i. The boundary value in the internal level that achieves a difference in the output of the neuron is, thanks to the very definition of the activation function, $a = 0$. Those values that make the internal level zero, $\sum_i w_i x_i - b = 0$ geometrically constitute the decision boundary between the zone where the neuron will be activated or deactivated. The equation $\sum_i w_i x_i - b = 0$ defines in the case of two input variables a given line, in the case of three input variables a given plane, and in the more general case of d input variables a given hyperplane. Thus, by varying the values of the weights w_i the artificial neuron is able to vary the position and orientation of the boundary and the classification regions.

In order to reach the optimal values of the weights starting from any initial values, Widrow and Hoff [57] stated in 1960 that what we should do was to pose a cost function, if possible convex, and iteratively, using the gradient descent algorithm, reach its global minimum. The gradient descent algorithm is a local search algorithm that allows to reach with some certainty the local minimum closest to a given point in a function. This is why if the cost function, the function to be optimized, consists of a single global minimum, the algorithm will converge to this single minimum. The cost function that Widrow and Hoff proposed was a function proportional to the mean square error, $J = \frac{1}{2}(y_{real} - z)^2$ and is still one of the most widely used. At each iteration the gradient method would seek to advance, within the space defined by the weights $\mathbb{R}^d$, in the local direction of greatest descent, i.e., to advance in the direction of the vector $-\overline{\nabla}J$. If at a given time we find ourselves, within the $\mathbb{R}^d$ space, at the point $(w_0, w_1, \dots, w_d)$, our next proposal of weights will be $(w_0, w_1, \dots, w_d) - \alpha\left(\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}, \dots, \frac{\partial J}{\partial w_d}\right)$, where $\alpha > 0$ is a hyperparameter known as learning rate that determines how far we will move in that direction and that we will have to specify before training the network.

The learning rate determines the stability and speed of convergence of the vector of weights to the minimum error value. It controls how large we make the steps as we move through the cost function. Very large values for the learning rate will cause the displacement to be aggressive, with large steps. The number of iterations needed to reach the minimum will be smaller, that is, the neuron will converge faster, but too high values will cause the cost to oscillate continuously between iteration and iteration, oscillating around a point without converging, as the local minimum may be a shorter distance from the current point than the rate allows to reach. If α is small the steps will be smaller and it will take many steps to converge, so it may be very slow.

The landmark that was the publication in 1958 of Rosenblatt's Perceptron [54] raised expectations of what artificial neurons could do. The publication ten years later of Minsky and Papert's book [58] demonstrating the limitation of the Perceptron to classify only linearly separable patterns dashed many of those expectations. The problem, also known as the XOR problem, is due to the inability of an artificial neuron to satisfactorily separate some data patterns, such as those in Figure 15. We have already outlined that the decision boundaries that a single neuron can produce are defined by the hyperplane equation $\sum_i w_i x_i - b = 0$, and this in the case of Figure 15 or other so-called linearly non-separable patterns is not sufficient, because even if we change the orientation and position the separation boundary will never manage to separate both classes.

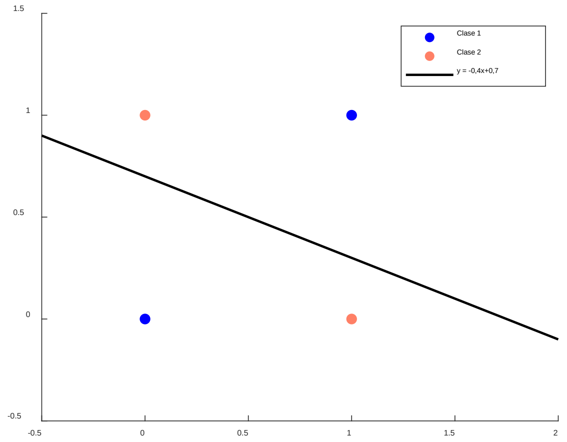


Figure 15. The XOR classification problem using an artificial neuron.

The solution to the problem of linear inseparability of classes was intuited to come from the sequential connection of artificial neurons, connecting the outputs of a first artificial neuron to the inputs of a second. This is where again it makes sense to have introduced a nonlinear activation function in the artificial neuron model, because otherwise, if the activation function were a linear function, the output of the neurons would always be a linear operation of their inputs (as a result of the linear combination and some linear operation) and the concatenation of linear operations that would be achieved by connecting them sequentially would be a linear operation that could be expressed through some linear combination, that is, if the activation function were linear, the sequential connection would be equivalent to a single neuron and would not contribute anything else. In contrast, by using nonlinear activation functions that distort the results of linear combinations of inputs (the internal level a), the sequential connection of neurons becomes meaningful. The nonlinear manipulations and deformations introduced by the activation function are necessary to be able to concatenate several neurons in an effective and useful way.

This type of sequential connections allows to organize the artificial neurons as in Figure 16, forming layered forward networks[13]. In this type of network, a layer of artificial neurons is formed by a set of neurons that all receive the same information from the previous layer in their inputs, and transmit their outputs to all the neurons of the next layer. The input layer of these

[13] The topology of a neural network is related to how the outputs of neurons are channeled to become inputs of other neurons. In addition to forward connections there is also the possibility of self-recurrent (the output of a neuron reconnects to its input), lateral (the output connects to the input of a neuron of the same layer) and backward connections. However, when referring to MLP, it is understood that its topology is constructed exclusively with forward connections.

networks is a first layer that without performing any computation transfers the values of the input data or variables to all the inputs of the next layer, the output layer provides the visible result of the network and the other layers are called hidden layers.

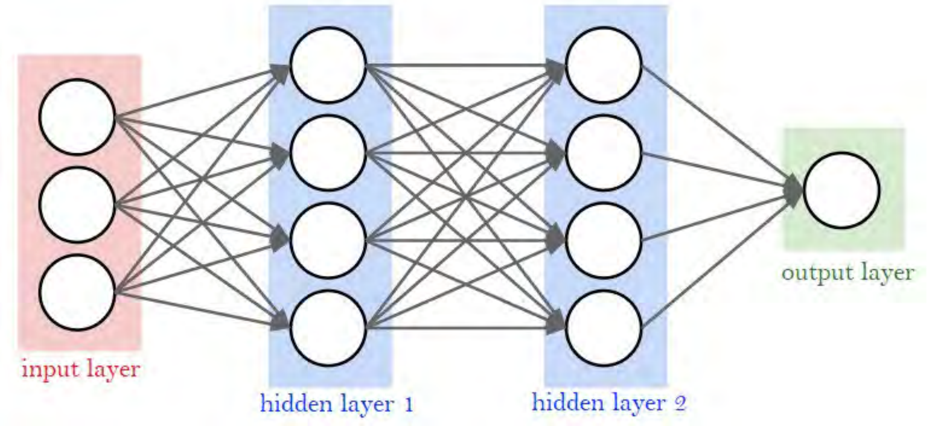


Figure 16. Layered structure of MLP networks. [59]

Sequential connections allow hierarchical knowledge, where each neuron specializes in one type of discrimination. Neurons in the initial layers acquire more basic knowledge, they are able to establish simple decision boundaries with simple hyperplanes, and then their activation function distorts the decision surface causing regions near the boundary to decay continuously. Figure 17 shows the geometric interpretation of the output surface for three neurons in the first layer when we work with only two features.

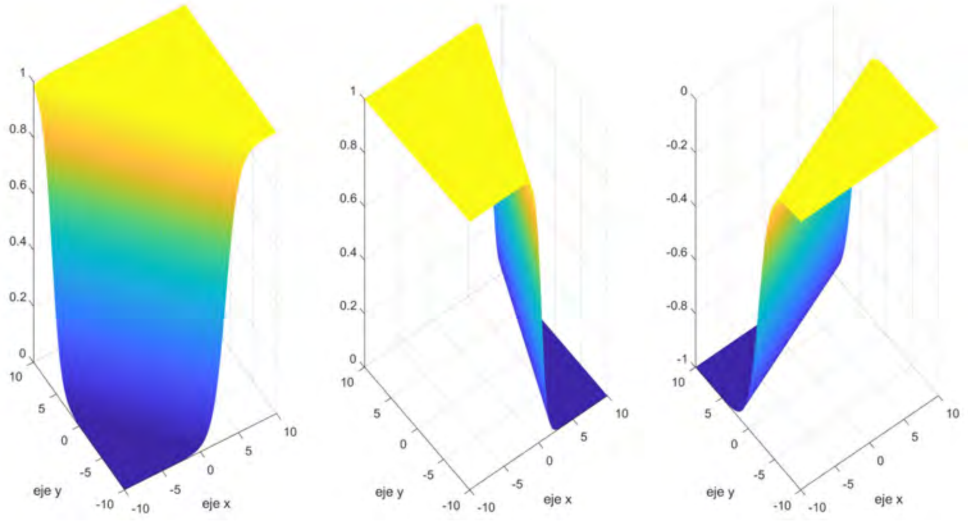


Figure 17. Output surface of already trained neurons of a first hidden layer.

The neurons of the next layer will be able, by taking the decision hyperplanes (distorted by the activation function) of the previous layer as edges or faces, to create closed areas or volumes. The subsequent layers, by means of the composition of these areas or volumes, will generate more complex spaces. Thus, an increasingly abstract, more complex knowledge is created as we advance through the layers of the network from its input. As an example, Figure 18 shows the geometric interpretation of the internal level of a neuron in the next hidden layer. In it, thanks to the linear combination of the decision surfaces of the previous layer, Figure 17, a new surface is obtained, which by establishing a constant height as a cut of the curve offers more complex decision regions delimited by the combination of the linear boundaries of the neurons of the previous layer. The application of an activation function to this decision surface would distort this surface while keeping the boundaries of constant value.

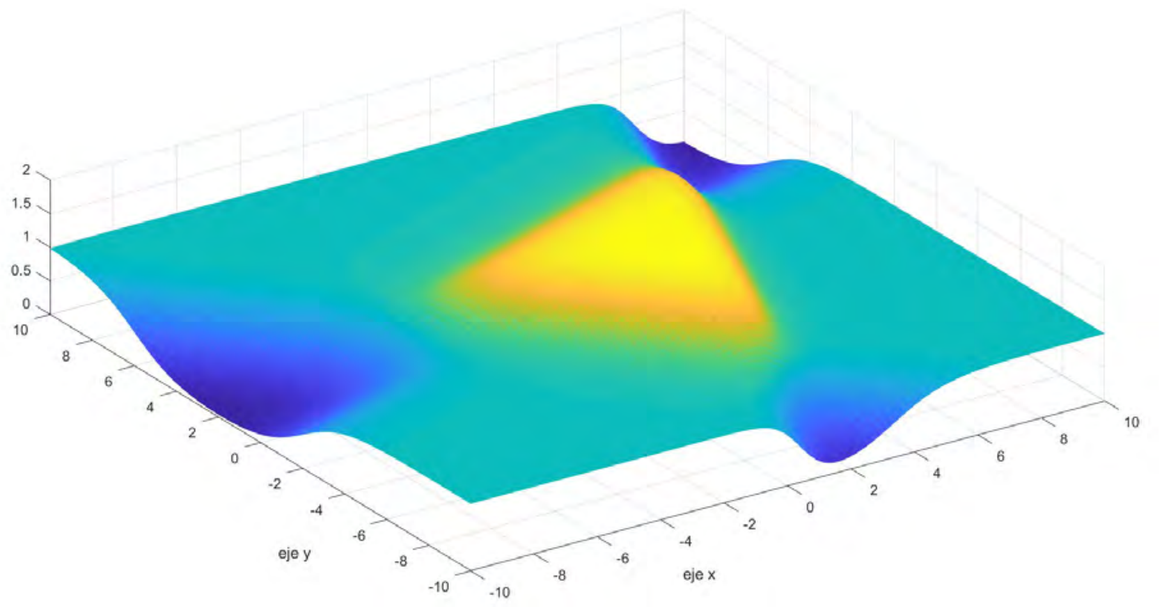


Figure 18. Internal level of a neuron of the second hidden layer result of a combination of the surfaces in Figure 17.

The distrust generated by Minsky and Papert [58] was overcome when, in 1985-1986, Rumelhart et al. [60, 61, 3] presented the error backpropagation learning algorithm (better known as Backpropagation), which offered an automatic learning method for these hierarchical networks based on knowledge of the desired outputs of the network as a whole for the training cases. The final accolade of overcoming an important part of the problems that had appeared in neural networks coincided, moreover, with the fact that in the 1980s various studies [62, 63, 64, 65] crystallized a range of neural networks with completely different structures and functions. It was the resurgence of neural networks.

In an MLP network the error of the previous layers depends directly on the error of the subsequent layers. A higher error assigned to a neuron in the later layers corresponds to a higher error share of the previous layers. Backpropagation allows to calculate the derivative of the error with respect to each parameter with a single backward pass. The algorithm applies the idea of the descending gradient from the final layers to the initial layers[14] to get a change in the value of the parameters w_{ji}^{l} that minimizes the cost function in the output of the whole network J. It distributes the error to each particular neuron of the previous layer proportionally to the participation of each neuron (through the weights that join each individual neuron of the previous layer with the neurons in the next layer and the error produced in that next layer). In this case each neuron j belonging to layer l will have its own weights w_{ji}^{l} starting from the weight w_{j0}^{l} (associated to a unitary input and interpretable as $-b$) up to the weight associated to its connection with the last neuron of the previous layer w_{jn}^{l}. Thus, defined a learning rate α, each of the weights will be updated to a new value $w_{ji}^{l} = w_{ji}^{l} - \alpha \frac{\partial J}{\partial w_{ji}^{l}}$. The key achievement of Rumelhart et al. [3] was, on the basis of the fact that each neuron performs a composition of

[14] w_{ji}^{l} indicates here the weight linking the output of neuron i of layer $l-1$ to neuron j of layer l. In case of using the term $i = 0$ we are referring to the negative of the bias introduced for neuron j of layer l.

functions $z_j^l = f_j^l\left(a^l\left(w_{ji}^l\right)\right)$ and applying the chain rule in the derivation, to get a reduced set of equations that could be solved from the last layer of the network to the first in a recurrent manner.

Let us explain it in more detail. If we have a forward neural network with L layers, and each of the l layers formed by $N(l)$ number of neurons, all of them with the same activation function $z = f(a)$, we will have:

- z_j^l: output of the neuron j of layer l.
- a_j^l: internal level of the neuron j of layer l.
- w_{ji}^l: weight of the connection from the neuron i of layer $l-1$ to the neuron j of layer l.
- w_{jo}^l: bias of the neuron j of layer l. It is equal to $-b_j^l$.
- In these elements the index $i \in [1, N(l)]$ and $j \in [1, N(l-1)]$.

Let us now analyze the error in the last layer formed by $N(L)$ output neurons. If we take the error function proposed by Widrow and Hoff, $J = \sum_{j=1}^{N(L)} \frac{1}{2}\left(y_{realj} - z_j\right)^2$the total error will be the sum of the accumulated error in all outputs.

Considering that each of the outputs is nothing more than a simple composition of functions we can rewrite J as

$$J\left(z_1^L, \dots, z_j^L, \dots z_{N(L)}^L\right) = \sum_{j=1}^{N(L)} J\left(z_j^L\right) = \sum_{j=1}^{N(L)} J\left(z_j^L\left(a_j^L\left(w_{j0}^L, \dots, w_{ji}^L, w_{jN(L-1)}^L\right)\right)\right) \tag{74}$$

Applying the chain rule in derivation and noting that w_{ji}^L is only related to the chain originating from z_j^L (the rest of $z_{x\neq j}^L$ do not participate in the term w_{ji}^L), we obtain that

$$\frac{\partial J}{\partial w_{ji}^L} = \frac{\partial J}{\partial z_j^L}\frac{\partial z_j^L}{\partial a_j^L}\frac{\partial a_j^L}{\partial w_{ji}^L} \tag{75}$$

Analyzing the terms of the equation we can see that:

- With the squared error $\frac{\partial J}{\partial z_j^L} = \left(z_j^L - y_{realj}\right)$.
- $\frac{\partial z_j^L}{\partial a_j^L}$ is the derivative of the activation function evaluated at a_j^L, $f'\left(a_j^L\right)$.
- $\frac{\partial a_j^L}{\partial w_{j0}^L} = 1$ and $\frac{\partial a_j^L}{\partial w_{ji}^L} = z_i^{L-1}\ \forall i \neq 0$.

As we continue in the demonstration we will see that in trying to get the derivative of the error in the weights of the following layers there are terms that repeat, namely $\frac{\partial J}{\partial z_j^L}\frac{\partial z_j^L}{\partial a_j^L} = \frac{\partial J}{\partial a_j^L} = \delta_j^L$. This term is interpreted as how the error changes as a function of the internal level of the neuron. That is, how the error changes in the face of a change in the neuron's internal level. If δ_j^L is large, a small change in the internal value of that neuron will be quite reflected in the final result. If it is small, changes in that neuron will have hardly any influence. It indicates the responsibility of each neuron in the error. For this reason, and from now on we will refer to δ_j^l as the error imputed to neuron j of layer l.

With this new formulation we can rewrite the results of the last layer as

$$\frac{\partial J}{\partial w_{ji}^{L}} = \delta_j^L \frac{\partial a_j^L}{\partial w_{ji}^L} = \begin{cases} \delta_j^L z_i^{L-1} \; \forall i \neq 0 \\ \quad \delta_j^l \text{ if } i = 0 \end{cases} \tag{76}$$

and knowing the expression of the error function and the activation function of each neuron

$$\delta_j^L = \frac{\partial J}{\partial z_j^L} f'\left(a_j^L\right) = \left(z_j^L - y_{realj}\right) f'\left(a_j^L\right) \tag{77}$$

Let's continue, now, to see what happens with the input weights to the previous layer $L-1$. If, again, we continue the simple composition chain[15] for $J = \sum_{k=1}^{N(L)} J\left(z_k^L\right)$ we will realize that, this time, the weights w_{ji}^{L-1} do influence each z_k^L. This is because each neuron in the layer $L-1$ is connected to all the neurons in the output layer, and in particular all the output neurons will be connected to the neuron z_j^{L-1}neuron, which is the one influenced by the weight w_{ji}^{L-1}. That is to say, $\forall k \in [1, N(L)]$, $z_k^L \rightarrow a_k^L \rightarrow z_j^{L-1} \rightarrow a_j^{L-1} \rightarrow w_{ji}^{L-1}$. In this case, $\frac{\partial J}{\partial w_{ji}^{L-1}} = \sum_{k=1}^{N(L)} \frac{\partial J\left(z_k^L\right)}{\partial w_{ji}^{L-1}}$, the error variances by the different output paths will be accumulated to form the total error variance.

For each pathway, for each output neuron k the dependence relationships are

$$J\left(z_k^L\right) = J\left(z_k^L \left(a_k^L \left(w_{k0}^L, \dots, w_{kN(L-1)}^L, \begin{array}{l} z_1^{L-1}\left(a_1^{L-1}\left(w_{10}^{L-1}, \dots, w_{1N(L-2)}^{L-1}\right)\right) \\ \vdots \\ z_j^{L-1}\left(a_j^{L-1}\left(w_{j0}^{L-1}, \dots, w_{ji}^{L-1}, \dots, w_{jN(L-2)}^{L-1}\right)\right) \\ \vdots \\ z_{N(L-1)}^{L-1}\left(a_{N(L-1)}^{L-1}\left(w_{N(L-1)0}^{L-1}, \dots, w_{N(L-1)N(L-2)}^{L-1}\right)\right) \end{array} \right)\right)\right) \tag{78}$$

As we have said, w_{ji}^{L-1} only appears in the chain of z_j^{L-1} and therefore

$$\frac{\partial J\left(z_k^L\right)}{\partial w_{ji}^{L-1}} = \frac{\partial J\left(z_k^L\right)}{\partial z_k^L} \frac{\partial z_k^L}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^{L-1}} \frac{\partial z_j^{L-1}}{\partial a_j^{L-1}} \frac{\partial a_j^{L-1}}{\partial w_{ji}^{L-1}} \tag{79}$$

but there are terms that we have already calculated as:

- $\frac{\partial J\left(z_k^L\right)}{\partial z_k^L} \frac{\partial z_k^L}{\partial a_k^L} = \delta_k^L$.
- $\frac{\partial z_j^{L-1}}{\partial a_j^{L-1}} = f'\left(a_j^{L-1}\right)$ is the derivative of the activation function.
- $\frac{\partial a_j^{L-1}}{\partial w_{j0}^{L-1}} = 1$ and $\frac{\partial a_j^{L-1}}{\partial w_{ji}^{L-1}} = z_i^{L-2} \; \forall i \neq 0$.

The remaining term $\frac{\partial a_k^L}{\partial z_j^{L-1}}$ is just w_{kj}^L, thanks to the fact that $a_k^L = \sum_j w_{kj}^L z_j^{L-1}$. We can reformulate the equation (79) as

$$\frac{\partial J\left(z_k^L\right)}{\partial w_{ji}^{L-1}} = \delta_k^L w_{kj}^L f'\left(a_j^{L-1}\right) \frac{\partial a_j^{L-1}}{\partial w_{ji}^{L-1}} \tag{80}$$

[15] We have changed the indexes of the original expression, since it is now j is an index that goes through the neurons of the layer $L-1$ and k will traverse the output neurons of the layer. L.

As it happened before, we will see that when we try to get the derivative of the error in the weights of the previous layers there are terms that repeat and we can define

$$\frac{\partial J\left(z_k^L\right)}{\partial a_j^{L-1}} = \delta_{kj}^{L-1} = \delta_k^L w_{kj}^L f'\left(a_j^{L-1}\right) \tag{81}$$

leaving the derivative of the error

$$\frac{\partial J}{\partial w_{ji}^{L-1}} = \sum_{k=1}^{N(L)} \frac{\partial J\left(z_k^L\right)}{\partial w_{ji}^{L-1}} = \sum_{k=1}^{N(L)} \delta_{kj}^{L-1} \frac{\partial a_j^{L-1}}{\partial w_{ji}^{L-1}} = \begin{cases} \displaystyle\sum_{k=1}^{N(L)} \delta_{kj}^{L-1} z_i^{L-1} \ \forall i \neq 0 \\ \displaystyle\sum_{k=1}^{N(L)} \delta_{kj}^{L-1} \text{ if } i = 0 \end{cases} \tag{82}$$

If we continue for the l previous layers we can extrapolate the result that we will obtain for the layer $L-2$. In this case the variation of the total error will be again the sum of the variations of the error by the different paths, $\frac{\partial J}{\partial w_{ji}^{L-2}} = \sum_{k=1}^{N(L)} \frac{\partial J\left(z_k^L\right)}{\partial w_{ji}^{L-2}}$ or in the more general case

$$\frac{\partial J}{\partial w_{ji}^{l}} = \sum_{k=1}^{N(L)} \frac{\partial J\left(z_k^L\right)}{\partial w_{ji}^{l}} \tag{83}$$

For each neuron in the layer $l+1$ there will always be a connection with the neuron of the previous layer whose z_j^l will be affected by w_{ji}^l. As a consequence all neurons in the layer $l+1$, all outputs z_t^{l+1}, will be affected by w_{ji}^l. Consequently, all the neurons in the layers $l+2, \dots, L$ will also be affected by w_{ji}^l. In the case of $l = L-2$ we will have that the different possible routes to reach z_k^L produce

$$J\left(z_k^L\right) = J\left(z_k^L \left(a_k^L \left(w_{k0}^L, \dots, w_{kN(L-1)}^L, \begin{array}{l} z_1^{L-1}\left(a_1^{L-1}\left(w_{10}^{L-1}, \dots, w_{1N(L-2)}^{L-1}, \begin{array}{c} z_1^{L-2}\left(a_1^{L-2}\left(w_{10}^{L-2}, \dots, w_{1N(L-3)}^{L-2}\right)\right) \\ \vdots \\ z_j^{L-2}\left(a_j^{L-2}\left(w_{j0}^{L-2}, \dots, w_{ji}^{L-2}, \dots, w_{jN(L-3)}^{L-2}\right)\right) \\ \vdots \\ z_{N(L-2)}^{L-2}\left(a_{N(L-2)}^{L-2}\left(w_{N(L-2)0}^{L-2}, \dots, w_{N(L-2)N(L-3)}^{L-2}\right)\right) \end{array} \right)\right) \\ \vdots \\ z_{N(L-1)}^{L-1}\left(a_{N(L-1)}^{L-1}\left(w_{N(L-1)0}^{L-1}, \dots, w_{N(L-1)N(L-2)}^{L-1}, \begin{array}{c} z_1^{L-2}\left(a_1^{L-2}\left(w_{10}^{L-2}, \dots, w_{1N(L-3)}^{L-2}\right)\right) \\ \vdots \\ z_j^{L-2}\left(a_j^{L-2}\left(w_{j0}^{L-2}, \dots, w_{ji}^{L-2}, \dots, w_{jN(L-3)}^{L-2}\right)\right) \\ \vdots \\ z_{N(L-2)}^{L-2}\left(a_{N(L-2)}^{L-2}\left(w_{N(L-2)0}^{L-2}, \dots, w_{N(L-2)N(L-3)}^{L-2}\right)\right) \end{array} \right)\right) \end{array} \right)\right)\right) \tag{84}$$

The basic literature of differential calculus of several variables, take as an example [66], reminds us that if a function $f\left(u(x,y,z), v(x,y,z), w(x,y,z)\right)$ then its partial derivatives $\begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial u} & \frac{\partial f}{\partial v} & \frac{\partial f}{\partial w} \end{pmatrix} \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial z} \\ \frac{\partial u}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial z} \\ \frac{\partial u}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial z} \end{pmatrix}$ which generalized for n independent variables entails that $\frac{\partial f}{\partial x_n} = \sum_t \frac{\partial f}{\partial u_t}\frac{\partial u_t}{\partial x_n}$. In our case $u_t = a_t^{L-1}$ and $x_n = w_{ji}^{L-2}$ or in the more general case $u_t = a_t^{l+1}$ and $x_n = w_{ji}^{l}$. As a consequence we can write

$$\frac{\partial J(z_k^L)}{\partial w_{ji}^{L-2}} = \sum_{t=1}^{N(L-1)} \frac{\partial J(z_k^L)}{\partial a_t^{L-1}}\frac{\partial a_t^{L-1}}{\partial w_{ji}^{L-2}} = \sum_{t=1}^{N(L-1)} \delta_{kt}^{L-1}\frac{\partial a_t^{L-1}}{\partial w_{ji}^{L-2}} = \sum_{t=1}^{N(L-1)} \delta_{kt}^{L-1}\frac{\partial a_t^{L-1}}{\partial z_j^{L-2}}\frac{\partial z_j^{L-2}}{\partial a_j^{L-2}}\frac{\partial a_j^{L-2}}{\partial w_{ji}^{L-2}} \quad (85)$$

and in the more general case

$$\frac{\partial J(z_k^L)}{\partial w_{ji}^{l}} = \sum_{t=1}^{N(l+1)} \frac{\partial J(z_k^L)}{\partial a_t^{l+1}}\frac{\partial a_t^{l+1}}{\partial w_{ji}^{l}} = \sum_{t=1}^{N(l+1)} \delta_{kt}^{l+1}\frac{\partial a_t^{l+1}}{\partial w_{ji}^{l}} = \sum_{t=1}^{N(l+1)} \delta_{kt}^{l+1}\frac{\partial a_t^{l+1}}{\partial z_j^{l}}\frac{\partial z_j^{l}}{\partial a_j^{l}}\frac{\partial a_j^{l}}{\partial w_{ji}^{l}} \quad (86)$$

As we have previously seen

- $\frac{\partial z_j^l}{\partial a_j^l}$ is the derivative of the activation function.
- $\frac{\partial a_j^l}{\partial w_{j0}^l} = 1$ and $\frac{\partial a_j^l}{\partial w_{ji}^l} = z_i^{l-1}\ \forall i \neq 0$.

and again we can redefine

$$\frac{\partial J(z_k^L)}{\partial a_j^l} = \delta_{kj}^l = \sum_{t=1}^{N(l+1)} \delta_{kt}^{l+1}\frac{\partial a_t^{l+1}}{\partial z_j^l}\frac{\partial z_j^l}{\partial a_j^l} = \sum_{t=1}^{N(l+1)} \delta_{kt}^{l+1} w_{tj}^{l+1} f'(a_j^l) \quad (87)$$

which substituting in equation (83) leads us to

$$\frac{\partial J}{\partial w_{ji}^l} = \sum_{k=1}^{N(L)} \delta_{kj}^l \frac{\partial a_j^l}{\partial w_{ji}^l} = \begin{cases} \sum_{k=1}^{N(L)} \delta_{kj}^l z_i^{l-1} \ \forall i \neq 0 \\ \sum_{k=1}^{N(L)} \delta_{kj}^l \text{ if } i = 0 \end{cases} \quad (88)$$

In order to be able to apply this algorithm, the activation function used in the neurons must be derivable. The most typical functions are the sigmoid, the hyperbolic tangent and the relu function, all of which are derivable.

Name	Hyperparameter	Function	Derivative
Sigmoid	$\beta \geq 0$ usually $\beta = 1$	$z(a) = \frac{1}{1+e^{-\beta a}}$	$z'(a) = \beta\frac{e^{-\beta a}}{(1+e^{-\beta a})^2} = \beta z(1-z)$
Tangent Hyperbolic	$\beta \geq 0$ usually $\beta = 1$	$z(a) = \frac{e^{\beta a}-e^{-\beta a}}{e^{\beta a}+e^{-\beta a}}$	$z'(a) = \beta(1-z^2)$
Linear Rectified (Relu)		$z(a) = \max(0, a)$	$z'(a) = escalón(a)$
Softmax		$z_j(a_1, a_2, \dots, a_j, \dots, a_{N(l)}) = \frac{e^{a_j}}{\sum e^{a_i}}$	$\frac{\partial z_j}{\partial a_j} = z_j(1-z_j)$ $\frac{\partial z_j}{\partial a_{i\neq j}} = -z_i z_j$
Gaussian	$\beta \geq 0$ usually $\beta = 1$	$z(a) = e^{-\beta a^2}$	$z'(a) = -2\beta a e^{-\beta a^2}$

Table 6. Most common activation functions in artificial neurons.

The sigmoid function is continuous, non-decreasing and derivable, and its derivative can be expressed in simple form with respect to its outputs[16]. It is the closest relative of the step function that was originally used, but being derivable. Taking high values for the parameter β it allows to simulate the step function because $\lim_{\beta\to\infty} sigmoid_{\beta}(a) = step(a)$. Its bounding in the range $[0,1]$ makes it especially attractive when one desires to interpret the outputs of a neuron as probabilities. Moreover, its derivative is 0 if the value of the output is close to 0 or 1, and its derivative is maximum and with value $\frac{\beta}{4}$ for values of a values close to 0 (if the internal level is close to 0). This makes it more likely during training to modify the weights of the inner levels closer to 0, i.e. to modify the weights of neurons that are more prone to change state, the neurons that are closer to a state change. This may prove to be a problem in certain applications where neurons close to the extreme values will rarely see their weights modified. The convergence of networks based on sigmoid function is usually slow.

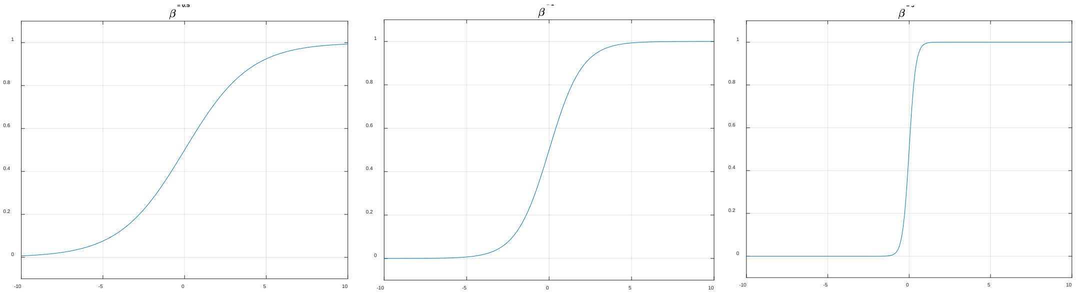


Figure 19. Representation of the sigmod activation function for $\beta = 0.5, \beta = 1$ *and* $\beta = 5$.

The hyperbolic tangent is a function very similar to the sigmoid, but with an output range of $[-1, 1]$. It can be used when the problem presents a certain symmetry, when one option and its opposite are presented.

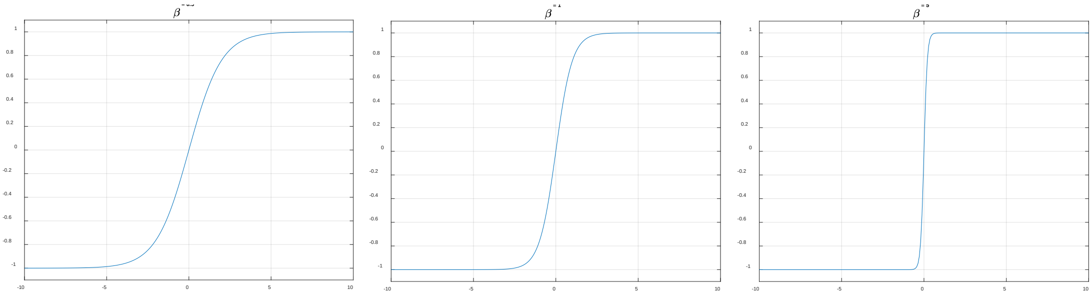


Figure 20. Representation of hyperbolic activation function for $\beta = 0.5, \beta = 1$ *and* $\beta = 5$.

Relu is an easy to train function, faster because its value is easily calculated. It is not bounded and behaves specially well with images and convolutional networks. It is only activated if the internal level of the neuron is positive, so that a phenomenon can occur where neurons with negative initial levels never come into operation, is the so-called dead neuron problem. In an attempt to solve such problems, a variant of the Relu function called Leaky Relu is sometimes used, which instead of using a value of zero for all values $a \leq 0$ assigns a line of small slope so that $z(a) = \begin{cases} \beta a & a \leq 0 \\ a & a > 0 \end{cases}$. In this way, negative internal levels are penalized with a rectifying coefficient that prevents the neurons from not modifying their weights.

[16] $z'(a) = \beta \frac{e^{-\beta a}}{\left(1+e^{-\beta a}\right)^2} = \beta \frac{1}{1+e^{-\beta a}} \frac{1-1+e^{-\beta a}}{1+e^{-\beta a}} = \beta z \left(\frac{1+e^{-\beta a}}{1+e^{-\beta a}} - \frac{1}{1+e^{-\beta a}}\right) = \beta z \left(1 - \frac{1}{1+e^{-\beta a}}\right) = \beta z(1-z).$

The Softmax function is a function that depends on the internal states of the neurons of the same layer. It has the advantage of being bounded in the range $(0,1]$ and it also has the property that $\sum_{j=1}^{N(l)} z_j = 1$. It is used when we want to have a representation in the form of coherent probabilities among the neurons of that layer, where each neuron represents a different option. In other words, Softmax transforms the outputs to a representation in the form of probabilities in such a way that the sum of all the probabilities of the outputs of that Softmax layer is 1. Its derivative is easily calculated[17] with an expression similar to that of the sigmoid and in order to apply the backpropagation algorithm it is usually accompanied by the cross-entropy error function $J = -\sum_{j=1}^{N(L)} y_{realj} \ln z_j$. They perform very well in the last layers when dealing with classification problems.

Finally, we have Gaussian activation functions. Mappings that usually require two hidden levels using neurons with sigmoid transfer functions can sometimes be performed with only one level in networks with Gaussian function neurons.

The error backpropagation algorithm, like other local search algorithms, is carried out iteratively on the training dataset. These cases can be taken one at a time in each iteration and update the value of weights w_{ji}^{l} at the end of each iteration or they can be taken in batches (several cases at a time accumulating the gradient) and update the weights at the end of each batch to avoid erratic behavior in the updates.

To measure the number of iterations to be performed, authors who have dealt with the problems of Overffiting and Underffitng [32, 34] usually, in each iteration, compare the values of the error function with the training cases with the value of the error function on some validation cases (different from the test cases) reserved for this purpose. If the error in the validation cases tends to increase while decreasing in the training cases, it is time to stop the training.

When working on the implementation of MLPs, one of the practical problems that arises is the choice of the network architecture. Although the input and output layers are defined by the application itself, how many layers should we use to obtain better performance, and how many neurons should we use in each one? Some authors such as Hilera et al. [30] argue that no more than four layers are necessary, because although a higher number of layers can speed up training, four are sufficient to generate arbitrarily complex decision regions. Regarding the number of neurons per layer Hilera et al. recall that the number of nodes in the 3rd layer, $N(3)$ must be greater than one when the decision regions are disconnected or *endentate* and cannot be formed with a convex region. This number, in the worst case, is equal to the number of disconnected regions in the input distributions. The number of neurons in the 2nd layer, $N(2)$ should normally be sufficient to provide three or more angles for each convex area generated by each neuron in the 3rd layer. Thus, there should be more than three times the number of neurons in the 3rd layer, $N(2) > 3N(3)$. In practice, an excessive number of neurons in any layer can generate noise but, on the other hand, if there are a number of redundant neurons, greater fault tolerance is obtained. The last layer will be very small because its neurons will only have to perform the OR of its inputs, that is, the union of all the disconnected regions produced

[17] $\frac{\partial z_j}{\partial a_j} = \frac{e^{a_j} \sum_{i=1}^{N(l)} e^{a_i} - e^{a_j} e^{a_j}}{\left(\sum_{i=1}^{N(l)} e^{a_i}\right)^2} = \frac{e^{a_j}}{\sum_{i=1}^{N(l)} e^{a_i}} \left(1 - \frac{e^{a_j}}{\sum_{i=1}^{N(l)} e^{a_i}}\right) = z_j(1 - z_j)$ and in the case of the partial derivative with respect to every term with $i \neq j$, $\frac{\partial z_j}{\partial a_{i \neq j}} = \frac{-e^{a_j} e^{a_i}}{\left(\sum_{i=1}^{N(l)} e^{a_i}\right)^2} = -\frac{e^{a_j}}{\sum_{i=1}^{N(l)} e^{a_i}} \frac{e^{a_i}}{\sum_{i=1}^{N(l)} e^{a_i}} = -z_j z_i$.

by the 3rd layer. Thus, a 4th layer with one neuron per output would be sufficient. The answer on the number of neurons depends therefore on the distribution of points to be classified in the feature space and is not clear, depending on the application for which the network is intended.

The underlying gradient descent algorithm is also not without problems in its application. The error function with respect to the different weights w_{ji}^{l} may have multiple local minima, so the algorithm may converge to local minima instead of the global one depending on the learning rate used. As discussed above, higher values could avoid falling into local minima but at the cost of achieving convergence in a larger number of iterations. Both Rumelhart et al. and Hilera et al. [3, 30] propose as a possible solution the addition of a moment to the gradient such that

$$\Delta w_{ji}^{l}(t+1) = -\alpha \frac{\partial J}{\partial w_{ji}^{l}} + \gamma\left[w_{ji}^{l}(t) - w_{ji}^{l}(t-1)\right] \tag{89}$$

With this moment, the convergence of the network is achieved in fewer iterations, since if in t the increase of a weight was positive and in $t+1$ also, then the decrease by the error surface in $t+1$ is greater. However, if in t the increment was positive and in $t+1$ is negative, the step in $t+1$ is smaller, which is appropriate, since that means that a minimum has been passed and that the steps must be smaller to reach it. This technique allows us to start from higher values of α that will be reduced in subsequent iterations.

Repeating the training of the network starting from different initial values will also help to avoid the problem of local minima, but does not ensure that we get the global minimum in any of these repetitions. Trying to alleviate the problems related to local optima, the algorithm is run several times starting from different initial states each time, and the best solution is taken from all the iterations. By running several times, we can observe that the result is much more consistent between runs (hopefully the same) [67]. Even so, nothing assures us that we have reached the global maximum. Authors such as Garcia [67] also indicate that even if the algorithm stabilizes in local solutions, if the result is acceptable, it should not be an element to worry about. That is, they judge the result not with respect to the best possibility but with respect to the general behavior of the network when it comes into operation.

In summary, starting from some training data[18] $X_{training} = \left\{\boldsymbol{x}^{(\mathbf{1})}, \boldsymbol{x}^{(\mathbf{2})}, \dots, \boldsymbol{x}^{(\boldsymbol{n}_{training})}\right\}$ and other different validation data $X_{validation} = \left\{\boldsymbol{x}^{(\mathbf{1})}, \boldsymbol{x}^{(\mathbf{2})}, \dots, \boldsymbol{x}^{(\boldsymbol{n}_{validation})}\right\}$ that have a similar proportion of cases of each class, the steps to follow to apply the algorithm are:

Step 1. Define the network architecture. Define the number of layers, L, and the number of neurons per layer to be used, $N(l)\ l \in [1, \dots, L]$. We will also choose the activation function, $z = f(a)$, that will be used in these neurons and thus we will know the expression of the derivative of this function at each point, $f'(a)$. We will set a value for the learning rate, α, large enough to achieve progress in learning and small enough not to have large oscillations in the errors per batch.

Step 2. Define the error function to be used, by default, the quadratic error function proposed by Widrow and Hoff

$$J(\boldsymbol{z}) = \sum_{j=1}^{N(L)} J(z_j) = \sum_{j=1}^{N(L)} \frac{1}{2}\left(y_{realj} - z_j\right)^2 \tag{90}$$

[18] Due to the use of subscripts in the neural network and for the sake of clarity, we use a different format for the vectors than that used in other sections.

Consequently, we can use the following expression from now on

$$\frac{\partial J}{\partial z_j^L} = \left(z_j^L - y_{realj}\right) \tag{91}$$

Step 3. Define the size of the batch we are going to work with, M. If we want to work without batches $M = 1$.

Step 4. Initialize the weights with unproblematic random values according to the type of activation function.

Step 5. In each iteration t we will take the following M corresponding training cases, $r \in [1, M]$. For each of these training cases, one by one:

a. Calculate the forward propagation of the signals. The bias terms will always be 1 and the outputs of the first layer will be directly the inputs applied to the system

$$z_i^1 = x_i^{(r)}, z_0^l = 1 \tag{92}$$

From the lowest layers up to the last one, from $l = 2, \dots, L$ we can advance by calculating

$$a_j^l = \sum_{i=0}^{N(l-1)} w_{ji}^l z_i^{l-1} \tag{93}$$

$$z_j^l = f\left(a_j^l\right) \tag{94}$$

b. We now calculate the backward error propagation. For the last layer $l = L$,

$$\delta_j^L = \frac{\partial J}{\partial z_j^L} f'\left(a_j^L\right) \tag{95}$$

$$\frac{\partial J}{\partial w_{ji}^L} = \begin{cases} \delta_j^L z_i^{L-1} \text{ if } i \in [1, N(L)] \\ \delta_j^L \text{ if } i = 0 \end{cases} \tag{96}$$

For the previous layer, $l = L - 1$:

$$\delta_{kj}^{L-1} = \delta_k^L w_{kj}^L f'\left(a_j^{L-1}\right) \tag{97}$$

$$\frac{\partial J}{\partial w_{ji}^{L-1}} = \begin{cases} \sum_{k=1}^{N(L)} \delta_{kj}^{L-1} z_i^{L-1} \text{ if } i \in [1, N(L-1)] \\ \sum_{k=1}^{N(L)} \delta_{kj}^{L-1} \text{ if } i = 0 \end{cases} \tag{98}$$

For the rest of the layers $l \in [2, L-2]$:

$$\delta_{kj}^l = \sum_{t=1}^{N(l+1)} \delta_{kt}^{l+1} w_{tj}^{l+1} f'\left(a_j^l\right) \tag{99}$$

$$\frac{\partial J}{\partial w_{ji}^{l}} = \begin{cases} \sum_{k=1}^{N(L)} \delta_{kj}^{l} z_{i}^{l-1} \text{ if } i \in [1, N(l)] \\ \sum_{k=1}^{N(L)} \delta_{kj}^{l} \text{ if } i = 0 \end{cases} \tag{100}$$

Step 6. After having calculated all the differentials, we are able to update the weights of the network:

$$\Delta w_{ji}^{l}(t+1) = -\alpha \sum_{r=1}^{M} \frac{\partial J^{(r)}}{\partial w_{ji}^{l}} + \gamma\left[w_{ji}^{l}(t) - w_{ji}^{l}(t-1)\right] \tag{101}$$

If we do not wish to apply the correction proposed in (89) we can take $\gamma = 0$.

Step 7. At the end of each complete iteration of the training data we will evaluate the error per batch

$$J_{training} = \sum_{r=1}^{M} J\left(\boldsymbol{z}_{\boldsymbol{training}}^{(r)}\right) = \sum_{r=1}^{M} \sum_{j=1}^{N(L)} \frac{1}{2}\left(y_{realj}^{(r)} - z_{j}^{(r)}\right)^{2} \tag{102}$$

and we will also evaluate the error in the validation data for comparison. We will proceed in a similar way to Step 5.a but with the validation data, equations (92),(93) and (94) applied on the validation data, to then calculate the accumulated error in these data

$$J_{validation} = \sum_{r=1}^{nvalidation} J\left(\boldsymbol{z}_{\boldsymbol{validaiton}}^{(r)}\right) = \sum_{r=1}^{nvalidaiton} \sum_{j=1}^{N(L)} \frac{1}{2}\left(y_{realj}^{(r)} - z_{j}^{(r)}\right)^{2} \tag{103}$$

The comparison between the evolution of the two errors will help us to know when to stop the algorithm: if we detect that the error after a minimum number of iterations in the validation data tends to rise, moving away from the curve made by the evolution of the error in the training data (which will tend to go down until converging to the minimum), we will end the training, otherwise we will repeat from the Step 5 with the next training batch.

MLPs have proven to be a very powerful technology. Each element of the network is able to learn by itself during the learning iterations, dynamically and adaptively, without the need to define previous models of the global output process to be modeled or represented and, thanks to the introduced nonlinear activation functions, able to learn nonlinear relationships between input and output vectors. The neurons of the intermediate layers are able to create their own abstract representation of the information, abstracting increasingly complex features and self-organizing the information. The network as a whole shows a distributed representation of information, capable of generalizing results to previously unexposed cases. They are distributed networks where information is encoded by activating distributed elements of the network. Thanks to this distributed organization, the network as a whole shows tolerance to some failures in the neurons and isolated connections, tolerance to some failures in the defined architecture (different architectures can obtain acceptable results) and adaptation to the noise of the inputs. In addition, they are capable of operating in real time and at high speed because, being massive parallel networks, made up of processing elements, the neurons, arranged in parallel in the different layers, the computation of the inputs is performed in parallel by each of the neurons.

It is due to these capabilities that they have been applied to countless fields such as pattern recognition, signal filtering, classification, signal prediction, prototyping, associative (content-addressable) memories, data segmentation, data compression and encoding, optimization problems, adaptive control, etc. [30].

2.7 Probabilistic Neural Networks (PNNs)

For the case of density function estimation of continuous variables, as early as 1962, Parzen [68] proposed a single-variate kernel series, later extended to multivariate versions by Cacoullos [69] which combined according to the training data lead asymptotically to the density functions we want to estimate and thus lead asymptotically to the optimal Bayesian classifier. Their approximation is guaranteed regardless of the kernel used, as Parzen himself demonstrates [68], provided that the distribution to be estimated is continuous (Murthy demonstrated later [70] that it is still applicable where the cumulative distribution is continuous).

Taking this idea Specht [71] offers an implementation, for the particular case with multivariate Gaussian kernel, in a type of neural network that he calls Probabilistic Neural Network (PNN). PNNs are three-layer forward neural networks that can be trained on a single pass of the data. They produce outputs with Bayes a posteriori probabilities in very short training times.

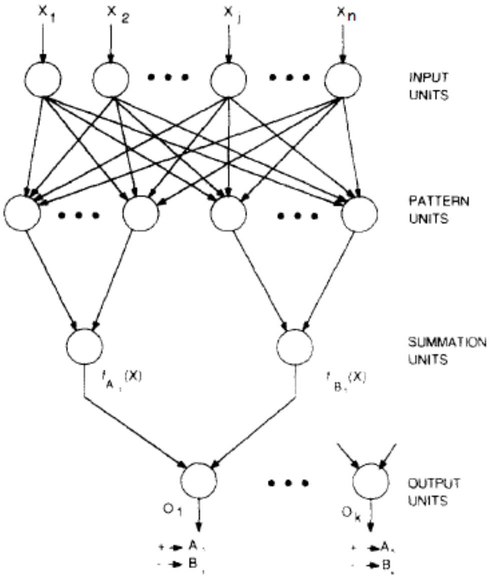


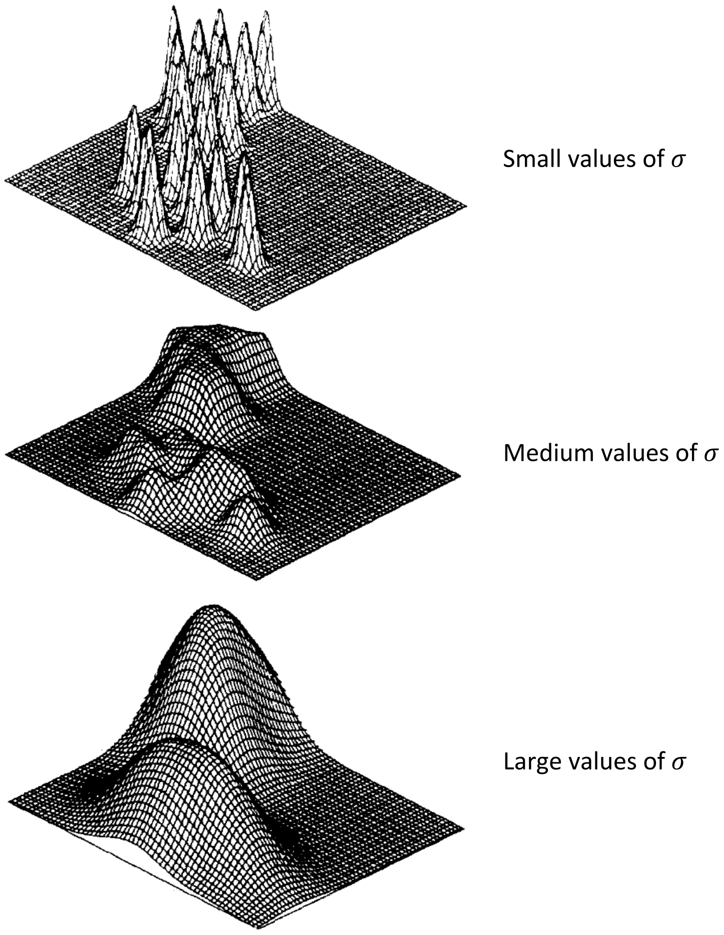
Figure 21. PNN network structure. [71]

Specht starts from a two-category problem and approximates the functions by means of the expression

$$f(\boldsymbol{x}|C_i) = \frac{1}{(2\pi)^{n/2}\sigma^n}\frac{1}{r_i}\sum_{j=1}^{r_i} e^{-\frac{\left\|x-\boldsymbol{s}_j^{(i)}\right\|^2}{2\sigma^2}} \tag{104}$$

where n is the dimension of the data, σ is the so-called "smoothing parameter", r_i is the total number of training examples of the category C_i, $\boldsymbol{s}_j^{(i)}$ is the example number j of the class i.

$f(\boldsymbol{x}|C_i)$ is simply the sum of small multivariate Gaussian distributions centered on the coordinates of each training example. σ is a design parameter in the range $[0,\infty)$ that regulates the degree of interpolation between the small Gaussians as shown in Figure 22. As shown by Spetch in a previous paper [72], for values close to 0 the classifier produces boundaries similar to the ones produced by a kNN classifier with $k = 1$, that is, taking the decision based on the nearest known neighbor. As the values of σ values increase the classifier tends to behave in a kNN-like manner taking into account an increasing number of neighbors in its decisions. Finally, for the extreme case $\sigma \rightarrow \infty$ it will take hyperplanes as classification boundaries.



Small values of σ

Medium values of σ

Large values of σ

Figure 22. Effect of the variation of σ at $f(\boldsymbol{x}|C_i)$. [71]

To implement the equation (104) Specht chooses to calculate through a scalar product the term $\left\|\boldsymbol{x}-\boldsymbol{s}_j^{(i)}\right\|^{\mathbf{2}} = \left(\boldsymbol{x}-\boldsymbol{s}_j^{(i)}\right)\cdot\left(\boldsymbol{x}-\boldsymbol{s}_j^{(i)}\right) = \|\boldsymbol{x}\|^2 + \left\|\boldsymbol{s}_j^{(i)}\right\|^2 - 2\boldsymbol{x}\cdot\boldsymbol{s}_j^{(i)}\boldsymbol{.}$ To simplify the result of this product, it will force to normalize the data before being used in the network (both training and real-use data), thus achieving $\|\boldsymbol{x}\|^2 = \left\|\boldsymbol{s}_j^{(i)}\right\|^2 = 1$ and therefore simplifying the previous expression to $\left\|\boldsymbol{x}-\boldsymbol{s}_j^{(i)}\right\|^{\mathbf{2}} = \left(\boldsymbol{x}-\boldsymbol{s}_j^{(i)}\right)\cdot\left(\boldsymbol{x}-\boldsymbol{s}_j^{(i)}\right) = 2 - 2\boldsymbol{x}\cdot\boldsymbol{s}_j^{(i)}$.

The scalar product is carried out through the inputs of the pattern neurons. If the input of the neural network is going to be the pattern to be classified $\boldsymbol{x}$, placing $\boldsymbol{s}_j^{(i)}$ in the weights of the pattern neuron j which is identified with the class i the value we will get before the activation function will be $z_j^{(i)} = \boldsymbol{x}\cdot\boldsymbol{s}_j^{(i)}$. That is to say, there will be one pattern neuron for each example $i = [1, r]$ where each weight of its input corresponds to each feature of a particular example.

To adapt it to the equation (104) Specht seeks to realize the equivalent of $e^{-\frac{\left\|x-s_j^{(i)}\right\|^2}{2\sigma^2}}$ under his assumptions, that is, $e^{\frac{2z_j^{(i)}-2}{2\sigma^2}} = e^{\frac{z_j^{(i)}-1}{\sigma^2}}$, which he achieves by means of the activation function $g(z) = e^{\frac{z-1}{\sigma^2}}$.

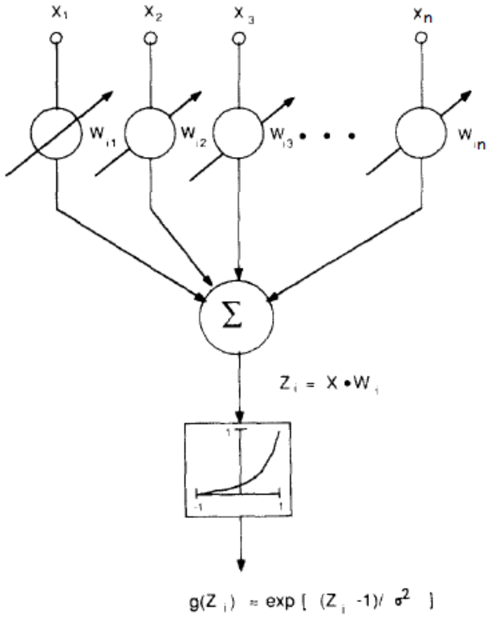


Figure 23. Diagram of the pattern units of the PNN. [71]

The summation $\sum_j g\left(z_j^{(i)}\right)$ will be achieved with an additional layer of adder units (neurons with weight 1 and linear activation function) that will be connected to those pattern units that represent examples of the same class C_i.

Finally, we must multiply the result $\frac{1}{(2\pi)^{m/2}\sigma^m}\frac{1}{r_i}$ to estimate the probability density $P(\boldsymbol{x}|C_i)$ and compare it through equation (3) to see which category is the most likely. Recall that [71] is facilitating the design of a binary classifier: a classifier that determines whether it $\boldsymbol{x}$ belongs to a category C_i or belongs to its complementary C_i' (does not belong to C_i). The same design reasoning followed to obtain $P(\boldsymbol{x}|C_i)$ has been followed to obtain the one of the complementary class $P(\boldsymbol{x}|C_i')$ by connecting in another adder those pattern units that **do not** represent examples of the class C_i. The equation (3) is reduced to know if $P(C_i)\frac{1}{(2\pi)^{m/2}\sigma^m}\frac{1}{r_i}\sum_j g\left(z_j^{(i)}\right) > P(C_i')\frac{1}{(2\pi)^{m/2}\sigma^m}\frac{1}{r_i'}\sum_j g\left(z_j^{(i')}\right)$. Since all the data have the same dimensionality, if we take the same value σ in all the pattern neurons of the system (which also reduces the number of design parameters considerably) we are left with a comparison where some terms are simplified to

$$\sum_j g\left(z_j^{(i)}\right) - \frac{P(C_i')r_i}{P(C_i)r_i'}\sum_j g\left(z_j^{(i')}\right) > 0 \qquad (105)$$

It is possible to implement this by multiplying by the constant $C = -\frac{P(C_i')r_i}{P(C_i)r_i'}$ which in most cases will be -1 because $P(C_i) = 1/r_i$ y $P(C_i') = 1/r_i'$ and a step activation function centered on the ordinate axis.

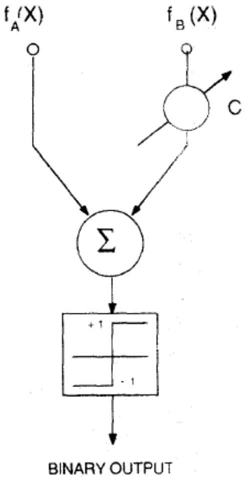


Figure24. Diagram of the NNP output units. Source: [71].

In short, the network is trained in a single pass by creating as many pattern units as training examples we have, by introducing successive examples in successive pattern unit weights. A second layer of summation units, with two summation units for each category. The first of the summation units will connect all the master units reflecting examples of the same particular category. The second summative unit will connect all the other pattern units that have not been connected to the first one (those that were related to the other classes). Finally, these two summative units (per class) will be connected to a new output unit through the constant

$$C = -\frac{P(C_i')r_i}{P(C_i)r_i'} \tag{106}$$

where $P(C_i)$ is the probability that a future data will be of the type i, $P(C_i')$ is the probability that a future data item is *not of the* class i, r_i the number of training cases of the class i, r_i' the number of training cases that are *not of the class* i.

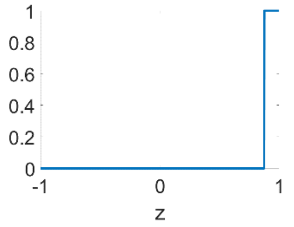
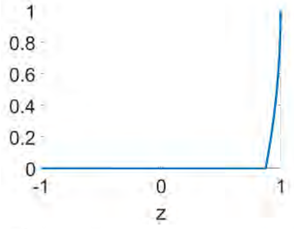
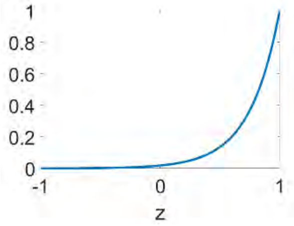
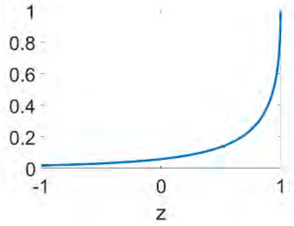
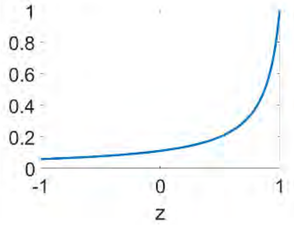
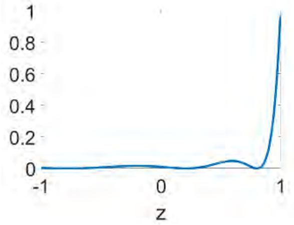
Activation function	Representation graphic	Activation function	Representation graphic
$\begin{matrix} \mathbf{1} & \boldsymbol{y \geq 1} \\ \mathbf{0} & \boldsymbol{y \leq 1} \end{matrix}$ $\boldsymbol{\sigma = 0.5}$		$\begin{matrix} \mathbf{1 - y} & \boldsymbol{y \leq 1} \\ \mathbf{0} & \boldsymbol{y \geq 1} \end{matrix}$ $\boldsymbol{\sigma = 0.5}$	
$\boldsymbol{e^{-\frac{1}{2}y^2}}$ $\boldsymbol{\sigma = 0.5}$		$\boldsymbol{e^{-\lvert y \rvert}}$ $\boldsymbol{\sigma = 0.5}$	
$\boldsymbol{\frac{1}{1 + y^2}}$ $\boldsymbol{\sigma = 0.5}$		$\boldsymbol{\left(\frac{\sin(y/2)}{y/2}\right)^2}$ $\boldsymbol{\sigma = 0.1}$	

Table 7. Activation functions in PNN pattern neurons.

With other cores proposed by [68, 69] one arrives at other equally valid activation functions that also lead asymptotically to the optimal BC. All of them lead to activation functions based on the term $y=\frac{\left\|\boldsymbol{x}-\boldsymbol{s}_j^{(i)}\right\|}{\sigma}=\frac{\sqrt{2-2z_j^{(i)}}}{\sigma}$ which is the one obtained by multiplying the weights of the pattern neurons. Thus, any one of the activation functions listed in Table 7 is valid and will lead to the optimal Bayesian classifier.

The problem is a two-category problem so that a positive output will only be provided if any of the categories offers a probability greater than 50% with respect to all the others considered as a single counter category. If we want to take advantage of the calculation of the different density function estimates to compute $P(C_i|\boldsymbol{x})$ we can take advantage of the results of the cumulative neurons and use Bayes' theorem directly by means of

$$P(C_i|\boldsymbol{x})=\frac{\frac{P(C_i)}{m_i}\sum_j g\left(z_j^{(i)}\right)}{\frac{P(C_i)}{r_i}\sum_j g\left(z_j^{(i)}\right)+\frac{P\left(C_i'\right)}{r_i'}\sum_j g\left(z_j^{(i')}\right)}=\frac{\frac{P(C_i)}{r_i}a_i}{\frac{P(C_i)}{r_i}a_i+\frac{P\left(C_i'\right)}{r_i{}'}a_i'} \tag{107}$$

where a_i is the output of the summative neuron related to class C_i and a_i' is the output of the summative neuron related to complementary class C_i'.

The advantage of this type of network over other neural networks is that training is simple and instantaneous. As soon as an example pattern of each class is stored, the network can begin to classify, and as more patterns accumulate in the network design, its generalization capacity will improve and its decision boundaries will become more complex. Furthermore, the complexity of the decision regions can be adjusted thanks to the parameter σ, for example, making it smaller as more examples are available, and all this without altering any other parameter of the network. Its main drawback is that the amount of computation needed to classify a new sample of data is proportional to the size of the training set, a fact that must be taken into account for the real-time application of this type of networks. Once the training is finished, PNNs have higher memory requirements and need a longer execution time to provide classification on unknown patterns compared to other conventional neural networks.

Chapter III. Red Wines Production

3 The production of Rioja red wines, the Bordeaux model

The production of any wine begins with the careful cultivation of the vine and the growth of the grapes. Rioja Denomination of Origin red wines are authorized to use specific varieties. The most common varieties are Tempranillo, Garnacha, Mazuelo and Graciano, with lower percentages of white grapes of Viura, Garnacha blanca and Malvasía de Rioja. The care of the vine, irrigation conditions, canopy management [73], and the health of the plant are essential to obtain a good raw material.

When the grapes are at optimum ripeness, harvesting is carried out, ideally by hand to ensure the selection and integrity of the grapes [74], although the aid of mechanized harvesting can reduce the time required for harvesting and can be more advantageous in large volumes. During the ripening process of the grapes, acids have given way to sugars synthesized through leaf photosynthesis. The trunks of the vine also contributed to the sweetness of the grapes by acting as sugar accumulators. The time to harvest has traditionally been estimated through sugar and acidity (pH) levels, and nowadays also through tannin and anthocyanin analysis. Authors such as Falcó [74] talk about polyphenolic maturity. The stage of ripening will vary in just a few days, but usually occurs sometime between August and October. During the harvest itself, the grapes are first selected, discarding damaged or immature clusters, etc.

The grapes have to be transferred to the winery in the shortest possible time and with the least possible damage, avoiding uncontrolled early fermentation. A breakage of the grapes results in a loss of must and possible premature and undesirable alcoholic fermentation [75]. To this end, crates of no more than 15-25 kg are used to prevent the grapes at the top from crushing those at the bottom, and attempts are made to harvest at low temperatures, sometimes at night [74]. If the distance between the winery and the vineyard is long, some type of refrigerated transport is used. The grapes enter the winery through the reception hopper, without violent unloading, re-sorting them and subjecting them to sanitary control analyses and the aforementioned parameters. It is also here where it is decided how much of the stem, the woody part of the bunch, will be included, eliminating the largest part through a process called destemming. Destemming is carried out in perforated metal drums that are rotated at high speed, so that the grapes escape whole, respecting their integrity, through the perforations of the drum and the herbaceous parts remain inside.

Next, the grapes are usually crushed in a press by gently and progressively increasing pressure, thus controlling that it does not break the pips or destroy the remaining stems, as this would release certain fatty acids that would contribute herbaceous flavors to the must, flavors that can become unpleasant [76]. The objective is to tear the skins to a greater length by rapid but not violent crushing of the grains. The press used is usually a hermetically sealed pneumatic press that avoids the introduction of oxygen, which could spoil the wine. This process facilitates subsequent maceration, by increasing the contact surface and activating the exchange of substances between the solid and liquid phases [75]. During maceration most of the substances transferred are beneficial to the wine (anthocyanins, tannins, nitrogenous compounds, polysaccharides, etc.) but other undesirable substances can also be transferred [77]. The resulting must is transferred by gravity or pumping to stainless steel tanks.

Alcoholic fermentation, the transformation of the sugars in the must into ethyl alcohol by *Saccharomyces* yeasts, is then carried out. The yeasts can be natural, coming from the grapes themselves, or added (selected yeasts). This process releases a significant amount of heat and carbon dioxide gas, so the main purpose of stainless steel tanks is to maintain the contents, by external or internal refrigeration, at a temperature of 25°C to 30°C in order to achieve greater extraction of flavors and aromas. Red wines made from quality ripe grapes and subjected to a long extraction from the skins by prolonged fermentations also contain a much higher percentage of flavonoids. The presence of lees (dead yeasts, vegetal remains, non-soluble acids and other sediments) helps to enrich the aromatic expression of the wine.

During fermentation it is important to control the amount of sugar remaining in the must, as this characterizes sweeter or drier wines. Once the alcoholic fermentation process is finished, the wine must have a content of less than 4 grams of sugar per liter to be considered dry. Taking all this into account, the vatting time usually ranges from 4 days to 4 weeks. In those wines that are to be aged in barrels and that come from fully ripe grapes, the process can be extended up to four or more weeks, resulting in wines of an overcast color, with a high tannin content, ideal for barrel aging.

Due to the carbon dioxide gas produced, fermentation pushes the skins and other solid parts to the top, giving rise to the *cap* that floats on the surface of the tank. To achieve greater contact between skins and must, favoring maceration, daily pumping over and punching down operations are applied [75]. The pumping-over consists of extracting the must or wine from the lower part of the tank and pumping it to the upper part to irrigate the *cap*. On the other hand, the bazuqueo technique consists of breaking the cap manually, pushing its mass from above with a pole so that it is submerged.

Once fermentation is complete, the tank is *devatted* or emptied to separate the must from the solid parts and transfer the liquid to another tank. Approximately 85% of its content comes out by gravity and natural pressure and is called *yolk wine (vino de yema)*. To take advantage of the remaining 15%, the pressure level is intensified to obtain what is known as first, second, etc. must. Finally, the leftover paste left in the press, the pomace (orujo), is used to make pomace brandies or to transform it into fertilizers or animal feed. If the press is used sparingly (without trying to extract the last drop), the resulting press wine will complement the *yolk wine*, but is not usually mixed with it, if at all, until bottling [74].

The next step to stabilize it is to get it to undergo malolactic fermentation. This softens the wine, making it more unctuous and reducing its acidity. This is a process in which the lactic acid bacteria naturally present in the grapes transform malic acid into lactic acid, which is somewhat less aggressive, releasing carbon dioxide gas [75]. The effects of this fermentation are especially beneficial for wines intended for aging, providing biological stability, which guarantees their conservation. Other young wines with high malic acidity should also necessarily undergo this process. However, it also has some trade-offs, as it causes a decrease in color intensity and a loss of aromatic potential [77]. On the one hand, the concentration of anthocyanins is altered due to the modification of the pH during malolactic fermentation and, in addition, the bacteria destroy anthocyanins in search of the glycoside part. On the other hand, there is a degradation of aromatic compounds and the diacetyl produced can give a, maybe excessive, buttery aroma.

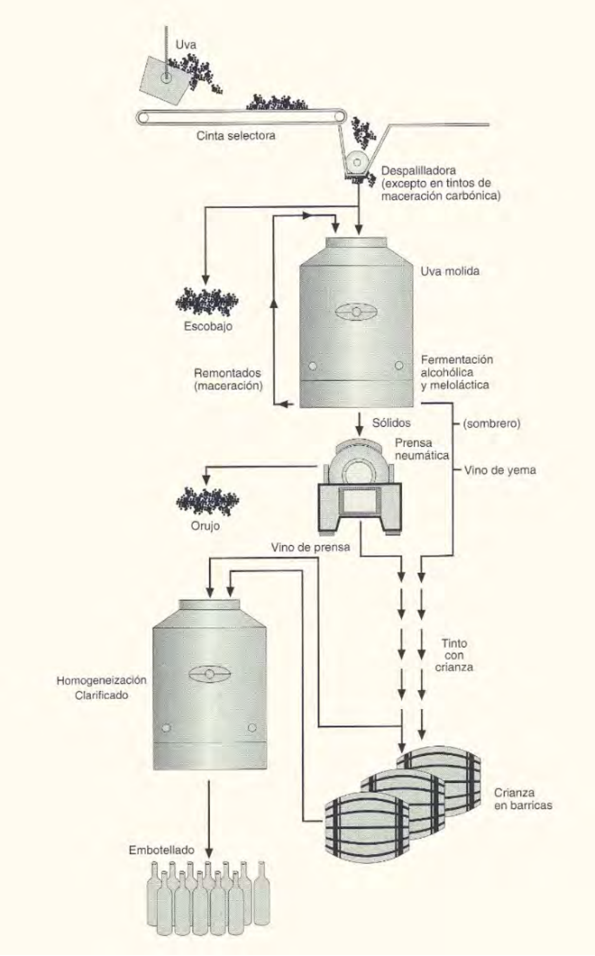


Figure 25. Diagram of red wine vinification. [74]

Finally, before fining and bottling, the wines can undergo a phase in barrels that lasts from a few months to several years and is called *crianza*. The barrel is an airtight container that allows the wine to be decanted and insolubilize various substances, eliminating its impurities or lees by gravity. This not only makes it possible to eliminate the filtering of the wine, avoiding its loss of flavors and aromas, but also, by using new or semi-new barrels (up to three years old) that also provide substances (such as ellagitannins or aldehydes from the wood [78]), adds aromatic and gustatory complexity. The most recognizable aromas and flavors contributed by aging include vanilla and spices.

The porosity of the barrels allows controlled oxygenation[19] within a mild and constant temperature range of 12-15°C and a relative humidity of 70-80% [77]. These conditions soften the tannins and stabilize the color, resulting in rounder, more elegant wines with better bottle ageability. Although barrels can be of different sizes, an adequate contact surface with the barrel is of interest. The 225 liter Bordeaux model is the one that has prevailed, using two types of oak with very different characteristics: French oak and American oak. French oak contains more phenols per unit of non-volatile extractable material than American oak and provides more solid extracts and phenolic compounds; however, it is believed that American oak contributes more to flavor per unit of tannins [77].

[19] Oxygen permeation through wood favors the formation of anthocyanin and flavanol derivatives [211].

The sediments that accumulate during aging can have a negative effect on the quality of the wine, and to eliminate them, the wine is periodically transferred from one barrel to another. This operation, known as *racking (trasiego)*, aims to separate the wine from the sediments accumulated at the bottom. Racking reduces the possibility of reactivation of microorganisms, eliminates excess carbon dioxide from the wine, homogenizes the aged wine and corrects the sulfur dioxide level [77].

Short aging, from 3 to 9 months, will provide a slight touch of wood, but will result in wines that should be consumed earlier. Long aging, with 12 or more months in barrels (usually up to 24 months with *rackings* every 3 or 4 months), produces wines with sufficient tannic structure for a long life in bottle. The latter generally require between 1 and 5 years in bottle to develop their full potential.

The most favorable situation before the wine ages in barrel is to have an anthocyanin to flavanol concentration of $1/4$. If this is not achieved, a situation of higher flavanol concentration is preferable so that all available anthocyanins can participate in polymerization reactions, eliminating the excess flavanols after aging to avoid condensation between them, which would give a taste with excess astringency in the wine. A wine poor in flavanols, in which the concentration of anthocyanins is higher than that of flavanols, is the least suitable for aging, because the excess of anthocyanins reacts by oxidation originating colorless phenolic acids, causing a significant destruction of color [77].

After aging, only two more steps remain. The clarification of the wine, which has the effect of precipitating the impurities still in suspension at the bottom, can be carried out by adding egg white or bentonite, a clay with a very fine structure. It is an essential operation in wines without aging [74], as they have not benefited from decanting in barrels, but the current trend is to reduce it to the minimum necessary for the preservation of aromas and flavors. It is so much so that a small deposit in the bottle should not be considered as a defect, but as a guarantee of quality and authenticity.

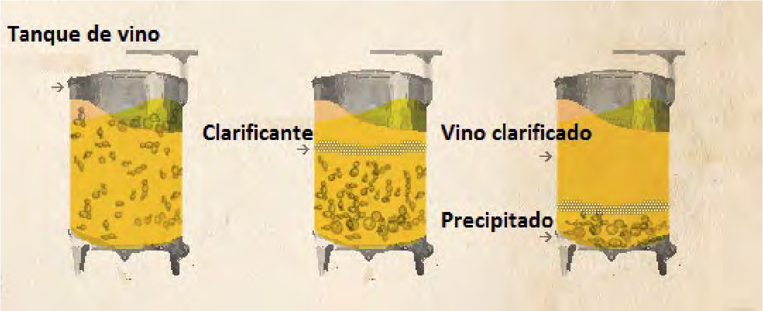


Figure 26. Clarification process. [74]

Finally, bottling should be done immediately after clarification. The key here is the quality of the cork. The advantages of cork are many. It is elastic, which allows it to adapt to the shape of the neck of the bottle, does not contribute flavors and keeps oxygen out as long as it is embedded. The increasing scarcity of quality cork has progressively increased its cost, giving rise to an alternative consisting of stoppers made from synthetic materials, screw caps or agglomerate stoppers.

During several of these steps there is the controversial possibility of adding sulfur dioxide to the wine. It is a well-known preservative that can have very positive effects as an antioxidant and antimicrobial. In addition, its intense degrading action on the skins allows greater maceration, and can result in the improvement and maintenance of wine aromas [77]. However, if used in high doses, it can lead to defective odors or risks for human consumption [76]. In summary, sulfur dioxide can be added during harvest and pre-fermentation macerations to prevent oxidation and undesirable fermentation, at the start of fermentation to prevent the growth of undesirable yeasts, and for aging and storage in tanks, barrels or bottles as a preservative.

Chapter IV. Background.

ML applied to Wines

4 ML applied to wines. Background

In the last two decades, the ML field has made numerous incursions into the world of wine. The studies have had different objectives, and can be grouped into three broad categories: those that predict or classify the hedonistic score that a wine will obtain, those that try to predict the sensations (mainly olfactory) that wines produce from their components, and those that classify wines according to their origin, grape variety or age.

4.1 Oriented to score prediction

In 2009 Cortez et al. [79, 80] established the most direct precedent of the current research, applying ML techniques to analytical data that are obtained in a standardized way to ensure the quality of products before selling in the food market. Their main advantage is that by taking data with high availability they were able to analyze almost 6500 samples of Portuguese green wines (1600 of red wines and 4900 of white wines) collected between 2004 and 2007 and analyzed by the official certifying entity CVRVV[20]. The chemical parameters they analyzed are summarized in Table 8. About the assessments, each sample was evaluated by a minimum of three sensory evaluators in blind tastings, giving a score between 0 and 10. The median of these evaluations makes up the final sensory score, all in the $[3, 9]$ range. This system means that most of the wines are within the $[4, 6]$ range, with very few high scores, below $0{,}1\%$.

Chemical features and units	Relative importance in SVM classifier (%)
Fixed acidity: Tartaric acid $(\boldsymbol{g/dm^3})$	7.4
Volatile acidity: Acetic acid $(\boldsymbol{g/dm^3})$	9.3
Citric acid $(\boldsymbol{g/dm^3})$	4.3
Residual sugar $(\boldsymbol{g/dm^3})$	6.8
Chlorides: Sodium Chloride	5.2
Free sulfur dioxide $(\boldsymbol{mg/dm^3})$	8.3
Total sulfur dioxide $(\boldsymbol{mg/dm^3})$	11.5
Density $(\boldsymbol{g/cm^3})$	4.9
pH	15.1
Sulfates: Potassium sulfate $(\boldsymbol{g/dm^3})$	16.9
Alcohol (% of volume)	10.3

Table 8. Analytical chemistry features used by Cortez et al. [79, 80].

Based on these data and using the R software [81], they apply three different ML techniques: LR, MLP with a single layer of hidden neurons and SVM with Gaussian kernel. The data are trained and evaluated through the k-fold cross validation procedure with $k = 5$, achieving the

[20] CVRVV: Comissão De Viticultura Da Região Dos Vinhos Verdes.

best results when SVM is used. The accuracy of the models generated as a result is summarized in Table 9.

	Red wines			White wines		
	LR	MLP	SVM	LR	MLP	SVM
Accuracy error tolerances of up to 0.5%	59.1%	59.1%	62.4%	51.7%	52.6%	64.6%
Accuracy error tolerances of up to 1%	88.6%	88.8%	89%	84.3%	84.7%	86.8%

Table 9. Accuracies of the models generated using SVM in [79, 80].

The models generated for red wines are consistent with oenological theory. They indicate that an alcohol content between 9% and 13% is one of the most influential factors in wine scoring, the higher the alcohol content, the higher the score. A pH around 3.4 in line with the alcohol content also contributes to a higher score. On the other hand, sulfate concentration seems to be another important factor, where values between 0.4% and 0.7% also improve the score, and may be a key element in the generation of aromas. In addition, the presence of acetic acid seems to have a negative influence on the score.

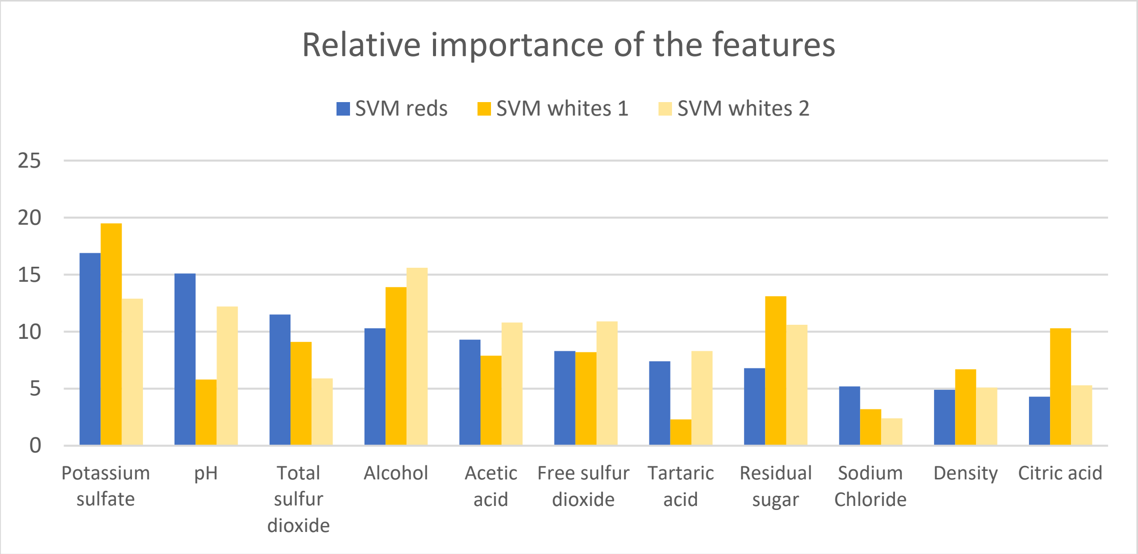


Figure 27. Relative importance of each data feature in the obtained SVM models of Cortez et al. [79, 80].

His work has been complemented in 2018 by the work of Gupta [82], which starting from the same dataset again applies LR, MLP and SVM. In this case, he first focuses on identifying the most important features of the data that help to predict their final score through the values of the regression coefficients obtained in LR, to subsequently apply MLP and SVM. A reduction to the most relevant features achieves small improvements in the overall accuracy of the models. The selected features (in order of importance) were: alcohol, acetic acid, sulfates, total sulfur dioxides, chlorides, pH and free sulfur dioxides. In addition, more recently, Kumar et al. [83] show that new variations of MLP and SVM can be used for this same task.

On the same dataset Lee et al. [84] apply a CART algorithm, the so-called C4.5, to predict wine scores using integers. The predictors reach accuracies around 60% for white and red wines, but with much lower results for the top scores of red wines, only around 35%. The same study shows that SVM and MLP achieve similar accuracies. The decision tree shows that alcohol content is again one of the most important features to make predictions.

A completely different approach is that of Chen et al. [85] in 2014, that takes not the chemical components as the basis for obtaining a personalized rating, but the various descriptive ratings and reviews gathered from specialized journals. They first create a system that automatically obtains the flavors and features of the wines from the various tasting notes and reviews written in common language. Following the example of the wine aroma wheel [86, 87] shown in Figure 44, they analyze the reviews of the 100 highest rated wines of the year in the 2011 Wine Spectator journal and extract their keywords/attributes by assigning them to the necessary subcategories and categories. After a normalization of these, in short, a convergence of terms whose content is similar, they manage to go from 547 recognizable features in the texts to a total of 376 binary characteristics that may or may not be present in a representation of a specific sample, grouped into different categories and subcategories. Subsequently, in 2016, the same research group generated an improved version of these standardized features [88], starting from the 100 most highly rated wines each year published in the same journal from 2003 to 2013 (a total of 1000 wines). The new feature dictionary consists of 1881 recognizable features that converge into 985 normalized binary features. All the data can be consulted in the archives provided by Chen [89].

Once the dictionary of normalized features has been created, they proceed to the prediction of its score. They take the texts of 1000 Wine Spectator journal reviews equally distributed between 80 and 100 points and using AR extract a series of inference rules with varying degrees of confidence. Applying these rules to particular samples in order of confidence, higher confidence will take precedence in case of discrepancy, they proceed to create an algorithm to predict whether a wine's score will be very high (greater than 90) or not. In short, the work is able, from a critical review of a particular wine, to predict whether its score is very high. If there are no rules applicable to the sample, the sample score will not be predictable and the algorithm will not provide any result. This is more likely the higher the degree of confidence required for the inferences. Taking 20% of the samples for training and 80% for testing, they evaluate them by cross-validations with $k = 5$. Their results reach accuracies of up to 82%, which allow them to predict up to 61.9% of the samples correctly.

Minimum confidence level (%)	Accuracy (%)	Applicability to data (%)
60	72.9	97.3
70	74.5	91.9
80	76.1	80.3
90	82.3	61.9

Table10. Results for a minimum support of 1% and different confidence levels using RA.

In a subsequent investigation by the same group of researchers [90], they apply other different ML techniques on the same dataset, these with full applicability. Using CART their results barely exceed an accuracy of 50%, but using kNN with Jaccard distances and adjustment of attribute weights they manage to reach 76.5%. The results are even better using NB with a slight modification to handle the 0 frequency elements, achieving 85.7% accuracy. Finally, the best result is by SVM with 88%.

In addition, using Jaccard coefficients as a measure of similarity between samples and adjusting the different weights of the characteristics, they achieve very interesting groupings by means of HCE. With at least 60% similarity they achieve useful groupings to identify wine regions of origin or, among other things, wines that may be attractive as a recommendation for a specific user.

4.2 Oriented to the prediction of sensations

The first steps in this direction that we can find date back to 2001 when Vlassides et al. [91] in a research mainly aimed at modeling the kinetics of the fermentation process and the resulting chemical features (final levels of ethanol, phenols, acidity, pH, etc.), used data from a hundred samples of Californian Sauvignon Blanc between 1996 and 1998 to obtain, by applying MLP, an assessment of the sweetness, acidity and mouthfeel (viscosity) perceived by 15 judges. To do so, they start from certain significant features of the samples, such as the level of ripeness of the grapes, the contact time of the skin with the must and the fermentation temperature.

Another important group of researches start from the so-called Volatile Organic Compounds (VOCs) and try to predict with them the standardized olfactory sensations they produce. Vigneau et al. [92] try to predict the olfactory characteristics of Cabernet Franc wines from their VOC patterns using Random Forest (RF) classifiers and focusing mainly on providing a list according to the order of importance of the variables involved in the classification. The data used were obtained by means of Gas Chromatography coupled to Mass Spectrometry (GC-MS) on 20 wines produced in 2011 and 10 wines in 2012 from the Loire Valley, thus searching for the 38 VOCs indicated in the Table 10. The 15 olfactory sensations selected are those that a group of experts found most significant in these wines within the usual descriptions in olfactory tastings[21]. The work tries to predict whether an olfactory sensation will be present or not depending on the VOC pattern of each wine. To do so, they train an RF classifier through the R software with 1000 trees, one third of samples checked at each node for decision making, a standard measure of squared error, and a minimum tree size of 5 nodes. The experiment is repeated until 1000 distinct forests are reached. Their way of achieving an order of importance of the variables in each tree is to measure the error (with the samples not participating in the training decision) by randomly permuting the values of the variables under consideration. By averaging over all the trees in the forest, it is expected that variables that are very important in the prediction will obtain worse scores for predicting correctly, the more influential variables when randomly modified will give worse results. The number of variables selected is based on the variability of the estimates in the forests obtained. With a variability greater than 2, they are discarded, although finally the 10 most significant variables are selected. These ordinations select variables considerably different from those selected by other regression methods.

Volatile Compound	No. Characteristic	Volatile Compound	No. Characteristic
2-methyl-1-butyric acetate	1	1-hexanol	20
amyl propionate	2	1-octanol	21
butyl acetate	3	1-phenoxypropan-2-ol	22
ethyl 2-hydroxy-propanoate	4	2-methyl-butan-1-ol	23
ethyl 2-methyl-butanoate	5	3-hexen-1-ol	24
ethyl 2-methyl-propanoate	6	3-methyl-butanol	25
ethyl 3-hydroxy-butanoate	7	methionol	26
ethyl 3-methyl-butanoate	8	Phenylethyl alcohol	27
ethyl butanoate	9	α-ionone	28
ethyl hexanoate	10	β-damascenone	29
ethyl octanoate	11	β-ionone	30
ethyl propanoate	12	2-methoxy-4-ethylphenol	31
ethyl-6-hydroxyhexanoate	13	4-ethylphenol	32

[21] The aromas analyzed were: leather, musk, black pepper, cherry stone, blackcurrant, fresh cherry, cooked cherry, fresh strawberry, blackberry, dried plum, cut grass, bell pepper, wood, hay and ethanol.

hexyl acetate	14	2,3-butanedione	33
isoamyl acetate	15	benzeneacetaldehyde	34
isoamyl propanoate	16	metional	35
3-methyl-butanoic acid	17	furaneol	36
benzeneacetic acid	18	homofuraneol	37
butanoic acid	19	3-isobutyl-2-methoxypyrazine	38

Table 11. Volatile compounds used in Vigneau et al. research [92].

A final improvement proposed by Vigneau et al. [92] is, with all possible subsets of only the most significant variables to analyze which of RF gives better results and how many of those variables should be taken. The results are tested by cross-validation leaving one sample out to test (LOO) [93] equivalent to overlapping with $k = number\ of\ samples$. Their results show some alternative ways of identification of olfactory sensations with respect to those of previous literature, where compounds directly related to certain odors are not present in the first decisions of the trees and yet combinations of VOCs, so far considered less important, achieve a better result.

Already in 2020, and with a similar approach, Sigfredo et al. [94] try to predict aroma, flavor and color traits of 8 wines from vintages from 2008 to 2016 made from the same Pinot Noir vines from a vineyard in Victoria, Australia. The wines obtained all followed the same winemaking process, fermented with wild yeasts and aged in barrel for 20 to 22 months. The research collects three types of data that are treated independently by MLP classifiers. On the one hand, there are the data collected from the bottles by Near Infrared Spectroscopy (NIR) (1596 nm to 2396 nm every 8 nm) and colorimetry. On the other hand, there are meteorological data such as the integration of temperatures throughout the day from September to harvest, maximum temperatures in January, average maximum and minimum temperatures from veraison to harvest and the water balance as the difference between rainfall and evaporation rate. Finally, organoleptic evaluations were obtained from 12 tasters according to ISO 8586-1:1993, which evaluated the aspects of Table 11.

Description	Values	Description	Values
Color intensity	Clear-Dark	**Aroma of red fruits**	Absent-Intense
Body	Light-Heavy	**Aroma of black fruits**	Absent-Intense
Sweet flavor	Absent-Intense	**Yeast aroma**	Absent-Intense
Bitter taste	Absent-Intense	**Floral aroma**	Absent-Intense
Herbal flavor	Absent-Intense	**Sweet aroma**	Absent-Intense
Black fruit flavor	Absent-Intense	**Spicy aroma**	Absent-Intense
Sour taste	Absent-Intense	**Wood aroma**	Absent-Intense
Wood flavor	Absent-Intense	**Astringency**	Absent-Intense
Red fruit flavor	Absent-Intense	**Warm mouthfeel**	Absent-Intense
Spicy flavor	Absent-Intense		

Table 12. Organoleptic evaluations on Fuentes et al. research [94].

Three different classifiers are built. The first one, a MLP with 10 hidden neurons using as input the 100 NIR wavelengths and as supervised output the organoleptic evaluations. The second one, a MLP with 5 hidden neurons for the same outputs as the previous one but using as inputs the 5 meteorological features. Finally, a MLP with 10 hidden neurons using the meteorological data to predict color in CIELab, RGB and CMYK coordinates (10 output neurons). The first classifier offers worse results and indications of overtraining as the accuracy drops from 96% in the training set to 82% in the validation set. The second and the third classifiers have very good results with and global accuracy of 96% and 97% respectively. Higher water availability leads to higher foliage on the vine, lower sun influx on the grapes and higher relative plant effort to support the leaves with respect to the grapes. This leads to lower color, aromatic and taste intensities consistent with what has been shown in several studies [74, 95, 96]. On the other hand, lower water balances and higher sun exposures produce wines with higher intensities.

Research [97] developed by the same group in 2019, this time with the aim of predicting the VOC content of wines measured through GC-MS, shows similar results.

4.3 Oriented to the prediction of origin, variety or age

The mineral and metallic components of wines can reflect different factors. They are a reflection of the variety of grapes used, the soils where the vines take root and the climatic conditions during the growth of the grapes. For example, some elements such as Ba, Mg, Mn, Cs or Sr are lithophilic elements and are primarily related to the terroir [98, 99]. The winemaking process also leaves its own imprint on these types of components of the final product. For example, the contributions of lead, copper or arsenic are mainly of human origin, introduced through winery equipment [100, 101], wine treatments or pesticides [102], fungicides or fertilizers [103, 104, 105] and may be correlated with decreases in elements such as potassium [99].

A first group of research focuses on measuring the mineral content of wines, demonstrating a direct connection between the composition of wines and the soils of the vines from which they originate. It is so much so that they achieve a high degree of success identifying the origin of grapes, especially among very different regions.

Year	Authors	Samples	Origin	Analytical methods	Features	Varieties	ML Techniques	Reference
2018	Moreno et al.	140	Canary Islands: Tacoronte-Acentejo, Orotava Valley, Ycoden-Daute-Isora, Abona, Valle de Güímar, La Gomera, La Palma	Mixed ICP-OES with FTIR	B, Ca, Cd, Co, Cr, Cu, Fe, K, Li, Mg, Mn, Mo, Ni, Na, Pb, Zn, B, Ca, Cd, Co, Cr, Fe, K, Li, Mg, Mn, Mo, Ni, Na, Pb, Zn		LDA, MLP	[106]
2015	Azcárate et al.	57	Argentina: Mendoza, Río Negro, San Juan, Salta	ICP-MS	Li, Be, V, Mn, Co, Co, Ni, Cu, Ge, As, Rb, Sr, Mo, Cd, Ba, Tl, Pb, Bi	Torrontés, Chardonnay, Sauvignon Blanc	PCA, LDA	[100]
2014	Coetzee et al.	120	South Africa: 23 farms in Stellenbosch	ICP-MS	Li, B, Mg, Al, Ca, Ca, V, Mn, Co, Ni, Cu, Zn, Rb, Sr, Cd, Cs, Ba, Tl, U	Cabernet Sauvignon, Cabernet Franc, Malbec, Merlot, Petit Verdot, Pinot Noir, Pinotage, Shiraz	PCA, CA, DA	[107]
2012	Martin et al.	1397	Australia: 51 regions	ICP-MS, ICP-AES	Li, Be, Al, Ti, V, Cr, Mn, Co, Ni, Cu, Zn, Ga, Ge, As, Se, Rb, Sr, Y, Z, Nb, Mo, Cd, In, Sn, Sb, Ta, Cs, Ba, La, Ce, Pr, Nd, Sm, Eu, Gd, Tb, Dy, Ho, Er, Lu, Hf, Ta, W, Tl, Pb, Bi, Th, U, Na, Mg, Si, P, S, S, K, Ca, Fe	39 varieties	LDA	[108]
2009	Gonzálvez et al.	67	Levant: Utiel-Requena, Jumilla, Yecla, Valencia	ICP-OES	Al, Ba, Be, Ca, Cd, Ce, Ce, Co, Cr, Cu, Dy, Er, Eu, Fe, Gd, Ho, K, La, Li, Lu, Mg, Mn, Mo, Na, Nd, Ni, Pb, Pr, Sc, Se, Sm, Sr, Tb, Ti, Tm, V, Y, Yb, Zn		HCE, PCA, CART, LDA	[109]
2007	Moreno et al.	54	Canary Islands: Tacoronte-Acentejo, Orotava Valley	ICP-OES, GFAAS	Al, Ba, Cu, Fe, Mn, Sr, Zn, Ca, K, Na, M, Ni, Pb, Al, Ba, Cu, Fe, Mn, Sr, Zn, Ca, K, Na, M, Ni, Pb	Listán Negro, Negramoll	LDA, PNN	[110]

2007	Álvarez et al.	150	Andalusia: Jerez and San Lucar, Huelva	ICP-OES	Al, Ba, Cu, Fe, Mn, Sr, Zn, Ca, K, Na, M, Ni, Pb, Al, Ba, Cu, Fe, Mn, Sr, Zn, Ca, K, Na, M, Ni, Pb	Palomino	LDA, MLP	[111]
2006	Iglesias et al.	43	Spain: Catalonia, La Rioja, Somontano, Penedés	ICP-AES, ICP-MS	K, P, Ca, Mg, Na, Fe, Mn, Zn, Sr, Ba, Ni, Pb, V, Co, Cd, Sb, Li, Cu, Al		PCA, LDA	[105]
2005	Kment et al.	31	Czech Republic: Prague, Karlštejn, Mělník, Roudnice, Žernoseky, Most	ICP-MS, AAS	Na, K, Ca, Mg, Mn, Fe, Ag, Al, As, Ba, Be, Cd, Co, Cr, Cs, Cu, Li, Ni, Pb, Rb, Sr, Sb, Tl, U, V, Zn	Irsay Oliver, Müller-Thurgau, Red Tramin, Pinot blanc, Pinot noir, Saint Laurent, Sylvan green, Pinot gris, Riesling, Muscatel	PCA, CA, FA	[99]
2005	Šperková and Suchánecк	53	Czech Republic: 4 regions of Bohemia	ICP-MS, ICP-OES	Al, As, Ba, Ca, Ce, Co, Cr, Cr, Cs, Cu, Fe, K, Li, Mg, Mn, Mo, Na, Ni, Pb, Rb, Sb, Sn, Sr, Th, U, V, Y, Zn	13 Varieties including whites and reds	PCA, LDA	[112]
2003	Pérez-Trujillo et al.	153		IPC-MS	Te, Re, Be, Cd, Sn, Sb, Co, As, V, Ni, Ti, Ti, Cu, Pt, U, Cs, La, Zr, As, Zn, Rb, Tl, Tb, Ce, Pr, Nd, W, Pb, Sr, Ba, Ho, Tm, Sm, Eu, Gd, Lu, Dy, Er, Yb, Th		LDA, MLP	[113]
2001	Frías et al.	45	Canary Islands: El Hierro, La Palma, Lanzarote	FAAS, FAES	K, Na, Ca, Mg, Fe, Cu, Zn, Mn, Sr, Li, Rb		CA, ANOVA, PCA, LDA, SIMCA, SIMCA	[114]
2000	Nuñez et al.	25	Galicia: Riberia Sacra, Ribeiro, Valdeorras	CE	Na, K, Ca, Mg, Mn, Li, Li		AC, PCA, LDA, kNN, SIMCA	[115]
1997	Sun et al.	17	Germany: 6 regions	ICP-OES	B, V, Mn, Zn, Fe, AJ, Cu, Sr, Ba, Rb, Na, P, Ca, Mg, K	Silvaner, Riesling, Müller Thurgau, Pinot Blanc, Gewürztraminer, Kerner	PCA, MLP, CA, Bayes DA, Fisher DA	[116]

Table 13. Summary: Recent researches on the classification of wines according to their origin, based on metal concentration.

Among these investigations is the one carried out by Azcárate et al. [100] who in 2015 used Inductively Coupled Plasma coupled Mass Spectrometry (ICP-MS) with a previous dilution of the samples to avoid possible interferences in the plasma to analyze the composition in 17 metals[22] of 57 duplicate samples (a total of 114 samples) of white wines from 4 different areas of Argentina. The selected samples were 33 wines of Torrontés, Chardonnay and Sauvignon Blanc varieties from the Mendoza region, 9 samples of Torrontés and Sauvignon Blanc varieties from Río Negro, 9 samples of Torrontés variety from the San Juan region, and finally, 6 samples of Chardonnay and Sauvignon Blanc from the Salta region. The application of PCA to the 114 samples showed that only two principal components accounted for 95.95% of the variance of the data. The metals Ba, As, Pb, Mo and Co are specially involved in these principal components, in consistency with other researchs [98], [101], [117], [118] and [119]. Finally, all the principal components are used as inputs for an LDA classifier. Taking 63 samples to train LDA and using the remaining 51 for testing, three principal components are obtained that classify with an accuracy of 96%.

Analytes $(\mu g \cdot L^{-1})$	Sauvignon blanc			Chardonnay		Torrontés		
	Mendoza	Río Negro	San Juan	Mendoza	San Juan	Mendoza	Río Negro	Salta
Li	941±855	815±272	1448±2	778±354	1102±370	396±303	146±3	192±175
Be	17±15	11±4	15±1	5±2	19±15	4±3	5±1	7±3
V	358±286	128±11	454±5	192±73	205±65	93±14	73±2	120±28

[22] The metals quantified were: Li, Be, V, Mn, Co, Ni, Cu, Ge, As, Rb, Sr, Mo, Cd, Ba, Tl, Pb and Bi.

Mn	1778±454	1370±660	1367±2	1739±762	1689±220	1345±182	1133±1	1544±817
Co	24±6	23±1	28±1	20±4	20±3	14±2	15±1	12±3
Ni	276±42	222±16	343±1	216±46	240±6	154±24	142±1	132±39
Cu	186±164	487±22	243±3	171±32	219±166	82±44	84±1	175±151
Ge	6.3±0.7	6.3±1	6.4±0.3	3.2±.09	2.7±0.6	1.1±0.2	1.7±0.2	1.7±0.2
As	20±5	20±1	36±3	26±10	29±6	18±3	11±1	13±1
Rb	531±253	804±43	800±1	719±95	679±132	592±295	891±1	977±217
Mr.	556±217	849±25	923±1	1007±241	1040±46	854±150	855±1	740±107
Mo	10±5	6±2	16±1	9±5	9±1	10±8	3±1	7±3
Cd	0.7±0.21	0.54±0.11	0.64±0.02	0.91±0.73	0.05±0.09	0.55±0.12	0.06±0.06	2.4±1.3
Ba	29±3	62±1	55±1	44±22	48±7	58±23	56±1	120±9
Tl	0.65±0.43	0.88±24	1.7±0.1	1.2±0.6	1.1±0.5	2.7±0.8	1.8±0.1	3±0.4
Pb	3±2	4±1	7±1	5±3	6±2	13±4	8±1	13±1
Bi	0.3±0.21	0.15±0.02	0.14±0.04	0.26±0.23	0.10±0.04	0.19±0.09	0.13±0.01	0.30±0.07

Table14. Element concentrations in different Argentine white wines. Source: [100].

In 2014, Coetzee et al. [107] analyzed by ICP-MS trace elements in 93 red and 27 white wines from a region of 1000 km^2 comparable to the area under the Rioja DO, in the Stellenbosch district of South Africa. They selected 23 estates and wineries with different soil topology working red varieties Cabernet Sauvignon, Cabernet Franc, Malbec, Merlot, Petit Verdot, Pinot Noir, Pinotage and Shiraz. The wines are from different vintages, between 2001 and 2010. To characterize the samples, they select those minerals that seem to be less affected by the winemaking process and, after a previous selection with PCA, they classify their origin by applying Cluster Analysis (CA) and, on the latter's groupings, DA. From a first set of 18 characteristics (Li, B, Mg, Al, Ca, V, Mn, Co, Ni, Cu, Zn, Rb, Sr, Cd, Cs, Ba, Tl, U), those 10 with the highest participation in the three main components (B, Mg, Sr, Ba, Rb, Cs, Tl, Cu, Ni and Zn) were selected. Subsequently, CA was applied using Ward's method, finding four different groupings, 3 of them for red wines. As in these groupings the wines from the different estates were not yet identifiable, cause several wineries were mixed in each group, DA was applied to each of them on the logarithm of the concentrations. The results are good, 80% success, discriminating in the first two groups and poor, only 30%, in the third. These same researchers had already applied similar techniques in 2005 [120] to differentiate between wines from different provinces of South Africa with success.

As early as 2012, Martin et al. [108] used numerous samples of Australian red and white wines. They took 1397 samples from up to 51 different regions of Australia. Thirty-nine different grape varieties from each of the vintages between 1992 and 2010 were represented in the samples. To characterize the samples, 56 different elements were quantified[23] by ICP-MS and ICP coupled to Atomic Emission Spectroscopy (ICP-AES). With the same technique, LDA, they create different classifiers. The first is to identify the age (2005-2008) within wines of the same variety (Syraz) and region (McLaren Vale). The results, where LDA was not able to create a good classifier, indicated that the composition is independent of age over almost two decades. The second classifier is to differentiate between whites and red wines, and subsequently through a new classifier, the variety within each group. The system is able to correctly classify the first part thanks to the higher Mg, P and K concentrations of the reds and thanks to the fact that the whites show higher concentrations of Be, Al, Y, Zr, Nb and rare earths. These differences seem

[23] ICP-MS to quantify: Li, Be, Al, Ti, V, Cr, Mn, Co, Ni, Cu, Zn, Ga, Ge, As, Se, Rb, Sr, Y, Z, Nb, Mo, Cd, In, Sn, Sb, Ta, Cs, Ba, La, Ce, Pr, Nd, Sm, Eu, Gd, Tb, Dy, Ho, Er, Lu, Hf, Ta, W, Tl, Pb, Bi, Th and U. ICP-AES to quantify: Na, Mg, Si, P, S, S, K, Ca and Fe.

to come from the very different contact times with the skins in the vinification processes. The second part did not achieve a classifier that improved a rate of 58%. The aim of third classifier is to differentiate wines according to their origin. The authors showed an 80%-90% success rate on the validation data (according to variety) with errors between subgroups of the same region. The concentrations of Li, Na, Mg, Si, P, K, Ca, Mn, Fe, Ni, Zn, Rb, Sr, Cs and Ba appear to be the most influential in these classifications.

In 2009, Gonzálvez et al. [109] attempt to classify red wines from eastern Spain by their metal content. The wines analyzed can come from 4 different Protected Designations of Origin (PDO): Utiel-Requena, Jumilla, Yecla or Valencia. For this purpose, the concentrations present in each wine of 38 different metals[24] are quantified by Inductively Coupled Optical Emission Spectroscopy (ICP-OES) at levels of mg/l. The origin of each of the 67 samples used from the different vintages from 1999 to 2006 can be consulted in Table 13. Once the data are collected they are analyzed using four different techniques: HCE, PCA, CART and LDA. The results show that the wines from Utiel-Requena and Jumilla can be discriminated from the rest by means of HCE and CART, the most important metals in this differentiation being Li and Mg. The samples from Valencia and Yecla could not be differentiated due, according to their authors, to the heterogeneity of the Valencia PDO, since it comprises a wide region with a variety of climates, soils and grape varieties, and it is not easy to group them together. In fact, Yecla wines seem to be more similar to those of the Clariano sub-region than to those of Alto Turia, Valentino or Muscatel. Through LDA all the wines could be correctly classified.

Region	Subregion	Samples
Utiel-Requena	Utiel-Requena	6
Yecla	Yecla	2
Jumilla	Jumilla	3
Valencia	Alto Turia	3
	Valentino	2
	Muscatel	6
	Clariano	45

Table 15. Distribution of samples in Gonzálvez et al. research [109].

As early as 2007, Moreno et al. [110] classified according to their origin 54 samples of 2004 red wines grown in volcanic soils: 30 samples from the Designation of Origin (DO) Tacoronte-Acentejo (Listán Negro and Negramoll varieties) and 24 samples from the Orotava Valley (Listán Negro). In the first case the soils are described as reddish in color, rich in organic soils, deficient in lime and rich in nitrogen, phosphorus and potassium. In the second case the soils are permeable, rich in nutrients and with a higher acidity due to the proximity of the volcano. Based on the concentrations of Al, Ba, Cu, Fe, Mn, Sr, Zn, Ca, K, Na and Mg detected using ICP-OES and Ni and Pb measured by Graphite Furnace Atomic Absorption Spectrometry (GFAAS) they apply LDA and PNN to classify the samples. LOO is used to test the models.

Analyte	Tacoronte-Acentejo	Orotova	Jerez	Chamomile	Huelva County
Al (mg/l)	7.32±1.91	8.79±2.29	1.71±0.61	1.57±0.46	2.50±0.67
Ba (mg/l)	0.12±0.03	0.14±0.06	0.03±0.02	0.06±0.12	0.06±0.03
Ca (mg/l)	111.55±26.79	106.99±17.99	68.95±11.96	68.38±11.22	85.34±27.75
Cu (mg/l)	0.31±0.37	0.47±0.76	0.30±0.46	0.22±0.15	0.33±0.35

[24] The metals quantified were: Al, Ba, Be, Ca, Cd, Ce, Co, Cr, Cu, Dy, Er, Eu, Fe, Gd, Ho, K, La, Li, Lu, Mg, Mn, Mo, Na, Nd, Ni, Pb, Pr, Sc, Se, Sm, Sr, Tb, Ti, Tm, V, Y, Yb and Zn.

Fe (mg/l)	3.01±0.96	3.1±0.88	1.73±0.70	1.84±0.50	3.83±1.80
K (mg/l)	1363.47±170.62	1307.16±290.02	474.86±152.03	484.85±85.55	888.31±196.12
Mg (mg/l)	124.94±25.6	110.66±15.52	60.06±15.47	58.58±7.08	68.03±11.92
Mn (mg/l)	2.36±3.29	1.82±0.62	0.55±0.29	0.59±0.18	0.77±0.35
Na (mg/l)	106.58±39.42	83.99±20.29	33.25±7.96	29.74±4.11	31.77±10.51
Sr (mg/l)	0.88±0.18	0.97±0.36	0.83±0.31	0.98±0.17	0.53±0.23
Zn (mg/l)	0.74±0.17	0.63±0.33	0.53±0.51	0.47±0.16	0.63±0.37
Ni (µg/l)	37.11±16.21	31.55±13.62	Not analyzed	Not analyzed	Not analyzed
Pb (µg/l)	1.16±1.46	Undetectable	Not analyzed	Not analyzed	Not analyzed
P (mg/l)	Not analyzed	Not analyzed	66.85±19.00	60.66±7.25	71.71±13.14

Table 16. Metal concentrations in the samples of Canary Island and Andalusian red wines. [110, 111]

LDA offers a 90.7% predictive ability, while PNN even exceeds it, reaching levels of sensitivity and specificity above 95%. According to LDA, Zn, Ba, Pb, Na and Mg are the optimal features to differentiate the two classes of wines.

In the same year, Alvarez et al. [111] follow a similar procedure to the previous one, but with fine wines from two DOs both using Palomino grapes mainly: on the one hand, Jerez-Xérès-Sherry and Manzanilla San Lucar de Barrameda; on the other, Condado de Huelva. They use 100 samples of the first DO (as they initially try to differentiate between Jerez and Manzanilla, which they later discard as they are very similar) and 50 samples of the second, which they submit to ICP-OES. They reserve 50% of the samples of each DO to test the models and take the remaining 50% to train them. By applying LDA they achieve an accuracy of 97.4% with the most discriminant elements, in order: K, Sr, P, Na, Al, Fe, Mg, Mn. On these pre-selected elements they create an artificial MLP neural network with two hidden neurons. By setting the learning rate to 0.2 and the momentum to 0.5, they achieve 100% accuracy.

In 2006, Iglesias et al. [105] determined by ICP-AES and ICP-MS the concentrations of 19 elements[25] to differentiate those DO Empordà-Costa Brava from the rest (DO Rioja, Somontano and Penedés). For this purpose, 43 wines from 2004 were analyzed and applying LDA they were able to distinguish the DO Empordà-Costa Brava wines from all the others in 100% of the cases only with the concentrations of Ba and Sr. In addition, for samples subjected to certain pretreatments, extremely low concentrations were found for certain elements. In the case of Zn this loss effect was found by Castiñeira et al. [119] who explained it through the oxidation process. The researchers emphasized that direct measurement of the wine, decomposition of the sample or digestion in an open container should therefore be avoided when taking the samples.

In 2005, Kment et al. [99] went a step further and analyzed not only the metals present in 31 wines, but also the 31 different soils in the Czech Republic where the grapes came from. Using ICP-MS, they analyzed the concentrations of 20 elements[26], and using Atomic Absorption Spectroscopy (AAS) the concentrations of Na, K, Ca, Mg, Mn and Fe. In addition, phosphorus was quantified by spectrophotometry at 820 nm. The concentrations can be found in Table 15. The data were analyzed by PCA, CA (with Euclidean distance and Ward's minimum variance) and Factor Analysis (FA) on the correlation matrix separately, i.e. soils on the one hand and wines on the other. Their results show a statistically significant dependence only between vineyard soil

[25] ICP-AES: K, P, Ca, Mg, Na, Fe, Mn, Zn and Sr. ICP-MS: Pb, Ba, V, Ni, Co, Cd, Sb, Li, Cu, Sr and Al.

[26] The elements were: Ag, Al, As, Ba, Be, Cd, Co, Cr, Cs, Cu, Li, Ni, Pb, Rb, Sr, Sb, Tl, U, V and Zn.

and wine contents for Mg. It was also observed that K and Mg contents could be affected by wine pH.

Analyte	Concentration in wines	Concentration in soil	Analyte	Concentration in wines	Concentration in soil
Na	14.7±12.7 mg/l	95.7±139 mg/kg	**Ni**	26.2±4.12 µg/l	2.52±1.32 mg/kg
Mg	75.4±16.8 mg/l	366±188 mg/kg	**As**	7.08±3.4 µg/l	-
K	1126±601 mg/l	510±487 mg/kg	**Mr.**	434±153 µg/l	35.9±32.8 mg/kg
Ca	108±45 mg/l	0,844±0,882 %	**Ag**	0.614±0.741 µg/l	-
Mn	0.925±0.712 mg/l	104±73 mg/kg	**Cd**	0.78±0.848 µg/l	0.093±0.086 mg/kg
Faith	2.64±1.06 mg/l	165±112 mg/kg	**Sb**	2.26±5.54 µg/l	-
P	15.3±13 mg/l	31.7±61.2 mg/kg	**Ba**	86.1±35.9 µg/l	26.2±17 mg/kg
Cu	448±1237 µg/l	11.7±17.6 mg/kg	**Tl**	0.221±0.154 µg/l	-
Zn	401±252 µg/l	9.22±12.4 mg/kg	**Pb**	67.1±217 µg/l	8.67±7.16 mg/kg
Be	0.695±0.813 µg/l	0.284±0.177 mg/kg	**U**	0.712±1.05 µg/l	-
To	560±387 µg/l	317±136 mg/kg	**Li**	38.2±74.9 µg/l	0.239±0.237 mg/kg
V	12.9±18.8 µg/l	-	**Rb**	702±558 µg/l	1.23±1.18 mg/kg
Cr	58.9±15.9 µg/l	-	**Cs**	6.3±12.2 µg/l	0.069±0.092 mg/kg
Co	2.07±1.02 µg/l	1.72±1.11 mg/kg			

Table 17. Metal concentrations in wines and soils from Czech Republic. [99]

Also in 2005, and also with wines from the Czech Republic, Šperková and Sucháneck [112] applied LDA to classify the origin of 53 wine samples from 4 different regions of Bohemia. The wines, from the 2000 and 2001 vintages, were collected directly from the manufacturers and subjected to ICP-MS (and ICP-OES) to quantify their content of 27 metals[27]. In the case of red wines, 100% of the wines could be correctly classified.

Four years earlier, in 2001, Frías et al. [114] classified 45 wine samples according to three possible origins (El Hierro, La Palma and Lazarote). For this purpose, they quantified the concentrations of 11 metals[28] by Flame Atomic Absorption Atomic Spectrophotometry (FAAS) and Emission Spectrophotometry (EES).) and Flame Atomic Emission Spectrophotometry (FAES). After normalizing the data they applied CA, ANOVA, PCA, Smooth Independent Modeling by Class Analogy (SIMCA) and LDA and validated them by LOO with different results. In the case of CA, with Euclidean distances and Ward's method, two main groups were differentiated grouping them according to the grape overripeness factor: one for the samples from Lanzarote and another for the rest of the samples from El Hierro and La Palma. ANOVA showed Rb, Na and Li as the most discriminating characteristics. In the case of PCA, four principal components explained 78.3% of the variance, and showed that the samples of the different groups were separated from each other. SIMCA presented a high degree of sensitivity (97.8%) and specificity (100%). Finally, LDA correctly classified 95.6% of the samples using only Rb, Na and Mn.

Along the same lines, the research of Nuñez et al. [115] in 2000, classified by means of CA, PCA, LDA, kNN with $k = 3$ and SIMCA 25 samples of red wines according to two origins, DO Ribeira Sacra or other regions. They used 15 samples from Ribeira Sacra, 5 from DO Ribeiro and 5 from DO Valdeorras. They measured the Na, K, Ca, Mg, Mn and Li contents of all the samples by Capillary Electrophoresis (CE) obtaining the results shown on Table 16. The classifiers were trained with 75% of the samples and 25% were reserved for testing. By means of CA, three groups were differentiated; in the first one the Valdeorras wines were isolated, the second one

[27] The metals quantified were: Al, As, Ba, Ca*, Ce, Co, Cr, Cs, Cu, Fe*, K*, Li, Mg*, Mn, Mo, Na*, Ni, Pb, Rb, Sb, Sn, Sr, Th, U, V, Y and Zn. Those marked with * are those analyzed by ICP-OES.

[28] The metals quantified were: K, Na, Ca, Mg, Fe, Cu, Zn, Mn, Sr, Li and Rb.

isolated part of the Ribeira Sacra wines and in the third one some Ribeira Sacra and Ribeiro wines were mixed.

	Ribeira Sacra	Ribeiro	Valdeorras
Li (µg/l)	46±20	51±10	46±25
Ca (mg/l)	68.1±9.3	78.2±8.4	27.7±3.0
Na (mg/l)	10.8±7.2	24.2±24.6	8.0±3.6
Mg (mg/l)	70.1±4.2	75.5±6.4	21.0±3.5
Mn (mg/l)	4.4±3.3	0.5±0.1	0.8±0.3
K (mg/l)	1116±145	1055±136	1702±118

Table 18. Metal concentrations in Nuñez et al. research [115].

By applying PCA three principal components were able to account for 86% of the variability in the data and visually could isolate the same groups as revealed by CA. LDA and kNN were able to correctly classify 92% of the Ribeira Sacra wines, but only 75% of those that were not. SIMCA gave a better result classifying those that were not (87%) than those that were (75%).

In 1997, Sun et al. [116] used PCA and MLP to differentiate the origin of 17 German wines. They took samples of all the wines and by IC-OES and 4 different preprocessing methods they analyze each wine 10 times at 17 different wavelengths. The readings at these 17 wavelengths are those that will then be used as features of the total 170 samples (10 for each of the 17 wines) for the ML techniques applied. The application of PCA is shown to be inadequate for the data, as it fails to separate from the different origins. CA (using Ward's method) gives dubious results even for its best grouping. Bayes and Fisher's discriminant analysis give near-optimal results. The most remarkable result, reaching 100% accuracy, is by means of a MLP fed by 17 input variables and 11 hidden neurons.

Finally, in this first group, it is worth mentioning the research carried out in 2003 by Pérez-Trujillo et al. [113] which, although it does not classify wines according to origin but according to whether they are white, red or rosé, does quantify the content of 39 trace elements and rare earths in 153 wines by means of ICP-MS. With an LDA classifier they were able to correctly classify 96.7% of the samples and with MLP 100%.

A second group of researches, instead of starting from mineral concentrations, start from the UV-visible or NIR spectra of the samples. Some of these investigations carry out a previous separation of the compounds in the samples, mostly by HPLC, and then use the spectra directly, as in the case of the investigations carried out by Acevedo et al. [121] or Beltrán et al. [122, 123, 124], or to quantify the concentrations of different polyphenols. With the concentrations or spectra as input features, wines are classified to one or more of multiple factors, either their origin, variety, age or even their identification.

Authors	No of Samples	Analytical methods	Features	Classification according to	ML Techniques	Reference	Year
Moreno et al.	140	Mixed ICP-OES with FTIR	16 metals, pH, volatile acidity, total acidity, reduced sugars, alcohol, free sulfur dioxide, total sulfur dioxide, malic acid, acetic acid, total polyphenols	Origin: Tacoronte-Acentejo, Orotava Valley, Ycoden-Daute-Isora, Abona, Valle de Güímar, La Gomera, La Palma	LDA, MLP	[106]	2018
Acevedo et al.	135	UV-visible spectrum	Full spectrum 200-800 nm every 10nm	Origin: La Mancha, Madrid Penedés, Rioja, Valdepeñas, Ribera del Duero, Toro, Somontano, Media	SVM, MLP, kNN, PLS-DA	[121]	2007
Beltrán et al.	172	HPLC-DAD	Full spectrum reduced by WT	Variety: Cabernet Sauvignon, Merlot, Carménère	LDA, kNN, PNN	[123] [122]	2005, 2006
Beltrán et al.	100	GC-SAW	Full spectrum reduced by WT	Variety: Cabernet Sauvignon, Merlot, Carménère	PCA, LDA, RBFNN, SVM	[124]	2008

Pisano et al.	27	HPLCA-MS/MS	Complete spectrogram	8 Varieties: Aspiran, Bonarda, Cabernet Sauvignon, Malbec, Merlot, Sangiovese, Syrah, Tempranillo. Origin: 13 areas of Argentina	MCR-ALS, D-UPLS	[125] [126]	2014 and 2015
Nunes-Miranda et al.	14	MALDI-TOF-MS	Full spectrum		CART, RF, NB, Bayesian Networking	[127]	2012
Gaeta et al.	7	UV-visible and NIR spectra	NIR: $1450\ \mu m$, $1600\ \mu m$, $1778\ \mu m$, $1800\ \mu m$, $2180\ \mu m$, $2230\ \mu m$, $2270\ \mu m$. UV-visible: 100 wavelengths reduced by PCA.	Identification	MLP, FNN	[128]	1998
Yu et al.	147	NIR	Full spectrum reduced by PCA	Aging: 1, 3, 5 years	LDA, SVM	[129]	2008
Costa et al.	37	HPLC-MS, Colorimetry	Color, total polyphenols, free radicals, antioxidant activity. Anthocyanins and derivatives: cy-3-glc, dp-3-(6-Ac)-glc, dp-3-glc, mv-3-(6-p-coum)-glc, mv-3-(6-Ac)-glc, mv-3-glc, pn-3-(6-p-coum)-glc, pn-3-(6-Ac)-glc, pn-3-glc, pt-3- (6-p-coum)-glc, pt-3- (6-Ac)-glc, pt-3-glc, vitisin A.	Origin: Mendoza (Argentina), Central Valley (Chile)	Chi^2, RF, RF, SMV, MLP, ELM	[130]	2018
Sen and Tokatli	63	UV-visible spectrum HPLC Colorimetry	pH Color Full spectrum 380-680 nm, first and second derivative spectra Anthocyanins and derivatives: mv-3-glc, dp-3-glc, pt-3-glc, pn-3-glc, dp-3-(6-Ac)-glc, pt-3-(6-Ac)-glc, pn-3-(6-Ac)-glc, mv-3-(6-Ac)-glc, mv-3-(6-p-coum)-glc, dp-3-(6-p-coum)-glc	4 vintages: 2006, 2007, 2008, 2009 9 varieties: Boğazkere, Cabernet Sauvignon, Kalecik Karası, Merlot, Öküzgözu, Syrah, Emir, Chardonnay, Moscato	PCA, OPLS-DA	[131]	2016
Gómez-Meire et al.	42	GC-FID, GC-FPD, GC-MS	Monoterpenes, alcohols, acetates, sulfides, esters, fatty acids 41 VOCs: linalool, a-terpineol, citronellol, nNerol, geraniol, , total monoterpenes, methanol, 1-propanol, 2-methyl-1-propanol, 2-methyl-1-butanol, 3-methyl-1-butanol, hexanol, 2-phenylethanol, total alcohols, ethyl acetate, isoamyl acetate, hexyl acetate, phenylethyl acetate, total acetates, ethyl butyrate, ethyl butyrate, ethyl hexanoate, ethyl octanoate, ethyl decanoate, ethyl laurate, ethyl myristate, ethyl lactate, diethyl succinate, total ethyl esters, butyric acid, isobutyric acid, hexanoic acid, octanoic acid, decanoic acid, lauric acid, total fatty acids, guaiacol, 4-vinyl-guaiacol, 4-vinyl-phenol, methyl-guaiacol, 4-ethyl-phenol, 4-ethyl-vanillate, vanillin, o-Cresol, total volatile phenols, Thiophenone, Methionol, methionyl acetate, total sulfides	Origin: 3 sub-zones of DO Rías Baixas of Galicia (O Rosal, Condado do Tea and Val do Salnés) Variety: Albariño, Treixadura, Loureira and Dona Branca.	PCA, LDA, SVM	[132]	2014
Cabrita et al.	81	HPLC-DAD	Gallic acid, protocatechuic acid, protocatechuic aldehyde, vanillic acid, caftaric	7 Varieties: Trincadeira, Aragonez, Cabernet Sauvignon, Alfrocheiro,	PCA, CA, SOM. CART	[133]	2012

			acid, syringic acid, p-coumaric acid and ferulic acid	Castelão, Touriga Nacional, Alicante Bouschet Malolactic Fermentation: Yes/No			
Gutierrez et al.	399	HPLC-DAD	Anthocyanins in logarithmic scale: dp-3-glc, cy-3-glc, pt-3-glc, pn-3-glc, mv-3-glc, pn-3-(6-Ac)-glc, mMv-3-(6-Ac)-glc, pn-3-(6-p-coum)-glc and mv-3-(6-p-coum)-glc.	Variety: Merlot, Carménère, Cabernet Sauvignon	BC	[134]	2011
Kruzlikova et al.	87	GC-MS	20 VOCs: Ethylhexanoate, (E)-3-Hexen-1-ol, (Z)-3-Hexen-1-ol, (E)-2-Hexen-1-ol, Ethyl 2-hydroxy-3-methylbutanoate, (E)-Furan linalool oxide, neroloxide, (Z)-Furan linalool oxide, Ethyl 3-hydroxy-butanoate, Linalool, α-Terpineol, Citronellol, 2-Ethylphenylacetate, Geraniol, 3,7-Dimethyl-1,5-octadien-3,7-diol, 1-Hexanol, Diethyl succinate, Hexanoic acid, Benzyl alcohol, 2-Phenylethanol.	Variety: Riesling, Veltliner, Mueller Vintage: 1996, 1997, 1998 5 Slovakian producers: Nitra, Krtíš, Pezinok, Dvory, Modra	MLP, LDA, kNN	[135]	2009

Table 19. Summary: Recent researches on wine classification based on UV-visible spectrometry, NIR and polyphenol quantification.

The research conducted by Moreno et al. [106] bridges the gap between these two major groups, as it uses both techniques. In 2018 they successfully classify red wines from 7 regions of the Canary Islands according to their origin. The regions were Tacoronte-Acentejo, Orotava Valley, Ycoden-Daute-Isora, Abona, Güímar Valley, La Gomera and La Palma whose wines come from volcanic soils with marked differences between them. To do so, they took 20 samples of red wines from the 2010 and 2011 vintages (a total of 140 samples) from which they obtained 29 different characteristics between metallic compounds and physicochemical parameters. Among the physicochemical parameters pH, volatile acidity, total acidity, reduced sugars, alcohol content, free sulfur dioxide, total sulfur dioxide, malic acid, acetic acid and total polyphenols were measured through Fourier Transform Infrared Spectroscopy (FTIR) in the Ultra Violet (UV) and visible spectra. To obtain the concentrations of metals, the samples were incinerated and the ashes were dissolved in nitric acid to apply ICP-OES.

Using LDA for classification, they were able to correctly classify more than 80% of the wines in each of the categories[29]. The results are even better when applying MLP, with 100% of the samples being correctly classified. They have used an architecture with 14 hidden neurons and sigmoid transfer functions in the intermediate and output layer. They reserved 10% of the training samples to avoid overtraining the classifier.

In 2007, Acevedo et al. [121] classified 82 white wines and 153 red wines, mostly young wines, according to their DO starting exclusively from the full signal of the UV-visible spectrum of each sample from 200 nm to 800 nm by 10 at 10 nm[30]. The wines and regions for red wines can be consulted in Table 18.

DO	Brand	Year	No. of samples	Predominant variety
La Mancha	Gold Vintages	2004	6	Tempranillo

[29] 90% for those from Tacoronte, 100% for those from Ycoden, 90% for those from Abona, 85% for those from Güimar, 95% for those from Orotava, 100% for those from La Palma, and 80% for those from La Gomera.

[30] Improving the resolution to 1 nm gave similar results.

	Alambrada Vineyard	2005	3	Tempranillo and Garnacha
Madrid	Castizo	2004	4	Tempranillo and Garnacha
	Puerta de Alcalá	2004	3	Tempranillo
	Puerta de Hierro	2005	4	Tempranillo and Garnacha
	Puerta de Hierro	2004	9	Tempranillo and Garnacha
Penedés	Sant Llach	2004	8	Tempranillo and Merlot
	Puig de Montlor	2004	3	Tempranillo and Merlot
	Val de Juy	2005	3	Tempranillo
Rioja	Espolón Vineyard	2004	10	Tempranillo and Garnacha
	Baron of Urzande	2005	6	80% Tempranillo and Garnacha
	Amate Vineyard	2005	4	70% Tempranillo and Garnacha
Valdepeñas	Viña Albali	2004	9	90% Tempranillo and Garnacha
	Señorío de Ojailén	2004	7	Tempranillo
	Señorío de Ojailén Reserva	2001	7	Tempranillo
Ribera del Duero	Dehesa de la Jara	2004	15	Tempranillo
	Vega de Nava	2004	9	Tempranillo
	Baron de Santuy	2004	7	Tempranillo
Bull	Gran Cermeño Crianza	2002	15	Red de Toro
Somontano	Montesierra	2005	9	Tempranillo, Moristel and Cabernet Sauvignon
	Viñas del Vero	2005	12	Merlot and Cabernet Sauvignon

Table 20. Description of red samples in Acevedo et al. research [121].

To classify the wines they tried different techniques: SVM with linear kernels; kNN with $k = 3$; MLP with 40 hidden sigmoidal neurons and linear neurons in the output layer; Partial Least Squares Discriminant Analysis (PLS-DA). After using 80% of the data for training and 20% for testing the classifiers, their results on red wines can be seen in Table 19. It shows the accuracy in percentage measured through cross-validation on the training set itself and, separately, on the test set. The best result is achieved by the SVM classifier with an average accuracy of 78.62%. The researchers warn that, as in other works, nearby regions, with similar climates and characteristics, or those wines with the same variety present more problems in order to be classified. Finally, they use the final SVM classifier obtained to select the number of wavelengths offering the best classification result (on 3 sets validated by overlapping cross-validation with $k = 5$). For red wines, spectra at wavelengths 290, 330, 360, 490, 540, 610, 650, 760 and 800 nm are sufficient. As the researchers themselves point out, this set includes the lengths that reflect the presence of anthocyanins, their derivatives and other phenolic compounds.

DO	SVM		MLP		kNN		PLS-DA	
	validation	test	validation	test	validation	test	validation	test
La Mancha	100.00	66.67	100.00	66.67	93.33	0.00	0.00	0.00
Madrid	96.00	66.67	96.00	66.67	75.00	33.33	87.00	66.67
Penedés	100.00	100.00	100.00	100.00	91.67	100.00	85.00	100.00
Rioja	100.00	75.00	100.00	75.00	80.00	25.00	80.00	50.00
Valdepeñas	93.33	71.43	93.33	57.14	86.67	28.57	86.67	42.86
Ribera del Duero	97.14	71.43	96.67	71.43	89.29	57.14	97.14	0.00
Bull	100.00	100.00	100.00	100.00	95.00	100.00	100.00	100.00
Somontano	96.67	77.78	93.33	66.67	100.00	33.33	100.00	33.33
Media	**97.89**	**78.62**	97.42	75.45	88.87	47.17	79.48	49.11

Table 21. Results grouped by region of classifiers with full spectrum in Acevedo et al. research [121].

Another research prior to this one by Acevedo et al. that used the full spectrum is that of Beltrán et al. [122, 123]. In this case, they classified a set of 172 Chilean wines according to their variety. The grape varieties used are Cabernet Sauvignon, Merlot and Carménère (80, 25 and 57 wines, respectively) grown in 5 valleys of central Chile during 2000 and 2001. Each sample was analyzed by High Perfomance Liquid Chromatography coupled Diode Array Detector (HPLC-DAD) obtaining chromatograms of 6751 points. Instead of identifying the phenolic compounds in each chromatogram and representing the samples by the concentration of these compounds, they chose to use the chromatogram points as input features of the system. By dealing with a much larger number of features with respect to the number of samples, they try to avoid the problems reported by Fukunaga and Hayes [35] by drastically reducing the number of features before using different classifiers. They first normalize the values of each chromatogram and discard the points corresponding to the first 5 minutes, discard the first 375 points, whose information is related to the effluents used in the HPLC technique. Then, they use the Nyquist frequency of the data to resample the data with a period of 4 seconds without losing information, which means reducing the number of points to one-fifth. The data are then subjected to a transformation stage and a classifier. The proposed transforms are the Discrete Fourier Transform (DFT) or the Wavelet Transform[31] (WT) which allow the signal to be represented in time by a much smaller number of coefficients. The DFT reduces the total number of coefficients to 480 and the WT is even more efficient by reducing the chromatogram to 43 points. Together with these two transforms, the authors decided to include residual and correlation coefficients with respect to the 3 class profiles to be identified in the data. These coefficients will be calculated in the time, frequency and wavelet domains and will be available for any classifier. The final classifiers proposed are LDA, kNN and PNN with spherical Gaussian radial basis functions centered on each eigenvector. Validating the data using the LOO procedure the best results with an accuracy of 94.77% are achieved for the WT together with the correlation coefficients over time and the PNN classifier. kNN and LDA classifiers also have results above 91%.

In a subsequent 2008 investigation [124] the same research group used samples of 100 wines of the same varieties from 1997 to 2003 from the same regions of Chile. Each sample was analyzed 10 times by Gas Chromatography (GC) coupled to a Surface Acoustic Wave (SAW) detector with valuable information only in the initial 12-second period which was sampled every 0.02 seconds (getting 600 points per profile). The data were normalized in the range $[-1, 1]$ being very similar among the different varieties. As they did previously [122, 123] they applied WT and PCA to the entire dataset to reduce the number of points/feature input to the classifiers. Using WT they achieved n levels of reduction getting $600/2^n$ points at each level. It should be noted that for the $n = 5$ the detector signal is represented by only 19 points. By means of PCA they manage to represent 99.87% of the variance with 20 principal components and 99.46% with only 10. Finally, they applied the data to train three different classifiers to obtain which of the three varieties the sample corresponded to. The classifiers were LDA, SVM and Radial Basis Function Neural Networks (RBFNN). The 1,000 samples were divided according to the Table 20 for training and testing of the algorithms.

No. of samples	Training	Test	Total
Cabernet Sauvingnon	330	30	360
Merlot	390	50	440
Carménère	180	20	200
Total	900	100	1000

Table 22. Distribution of training, validation and test cases in Beltran et al. research [124].

[31] They use the transform with the Haar wavelet function.

LDA provided good results with the training cases but poorer in the case of the test sets where the best result was 80% correct classifications with WT and $n = 5$.

During the training process of the RBFNN classifiers, their selectivity s (inverse of the extension σ) had to be specified. The best choice was obtained using part of the training set for validation of this parameter. Their results in the different ensembles can be consulted in Table 21. The best result, 88% of correctly classified cases, was obtained with the WT reduction, $n = 5$ and a selectivity of 0.1.

	Selectivity = 0.1		Selectivity = 0.02		Selectivity = 0.01	
WT decomposition	Success (%) in training	Success (%) in test	Success (%) in training	Success (%) in test	Success (%) in training	Success (%) in test
$n = 5$	75.6	**88**	76.8	82	76.5	83
$n = 4$	75.6	77	76.5	77	72.7	79
$n = 3$	74.8	71	76.6	71	75.4	78
$n = 2$	72.3	63	75.4	63	74.1	69

Table 23. Results of the RBFNN classifiers in Beltran et al. research [124].

The SVM classifier required two design parameters that were obtained by reserving part of the training cases for validation: penalty factor C and the extension σ. The best results of all the proposed classifiers were obtained by applying SVM with radial kernel ($\sigma = 0{,}1$) to the data reduced with WT, as shown in Table 22. To highlight that component reduction using PCA produced poorer results in all classifiers.

C	Success (%) in training	Success (%) in test
128	76.6	80
256	77.3	83
512	79.5	84
1024	81.3	88
2048	82.2	**90**
4096	83.3	89
8192	84.0	89
16384	84.8	88
32768	84.8	84

Table 24. Results of the SVM classifiers in Beltran et al. research [124].

Research by Pisano et al. in 2014 [125] and in 2015 [126] show how full 3D chromatograms can be used in wine classifiers according to variety or region successfully. They obtained red wine samples of 8 different varietal types (Aspiran, Bonarda, Cabernet Sauvignon, Malbec, Merlot, Sangiovese, Syrah and Tempranillo) collected in 2012 from 13 areas of Argentina. Samples were obtained directly from the tanks at the end of malolactic fermentation and prior to bottling to avoid condensation and polymerization of anthocyanins during aging. Their research employs HPLC-DAD and HPLC-MS/MS with columns and phases prepared for anthocyanin detection. After this, they used Multivariate Curve Resolution-Alternating Least Squares (MCR-ALS) [136] and Discriminant-Unfolded Partial Least Squares (D-UPLS) [137] to construct classifiers according to variety and origin region. With MCR-ALS, all varieties were correctly classified, while D-UPLS was able to differentiate Malbec, Merlot and Cabernet Sauvignon varieties from the rest. D-UPLS was also able to discriminate between the regions of eastern Mendoza, southern Mendoza, San Juan and the remaining 10 regions. Analysis of which variables participated with the greatest weight in the D-UPLS models identified mv-3-glc, mv-3-(6-Ac)-glc, mv-3-glc-4-vinylguaiacol and mv-3-(6-p-coum)-glc as some of the most important 4 variables.

Also among the researches working with the full spectrum we have the one carried out by Nunes-Miranda et al. [127] in 2012, where they managed to classify 14 Spanish and Italian white wines according to their variety and according to their winery. The wines are characterized with three full spectra when subjected to Matrix-Assisted Laser Desorption/Ionization Time of Flight coupled Mass Spectrometry (MALDI-TOF-MS) with three different matrices. The classifiers were constructed with 11 different techniques including CART, RF, NB, and Bayesian networks. The best results were obtained with Bayesian networks and CART, with accuracies of 95.20% and 94.40%, respectively.

In 1998, Gaeta et al. [128] used measurements of UV-visible and NIR spectra to characterize and subsequently identify wines with MLP classifiers and Fuzzy Neural Networks (FNN). Three red wines, three whites and one rosé were used in the study. In NIR, a total of 120 measurements were taken using the 7 wavelengths supplied by the manufacturer, namely, $1450\ \mu m$, $1600\ \mu m$, $1778\ \mu m$, $1800\ \mu m$, $2180\ \mu m$, $2230\ \mu m$ y $2270\ \mu m$. In the case of the UV-visible spectrum, a total of 100 measurements were taken. The original frequency data were subjected to PCA and screening by experienced technicians finally reducing to 5 wavelengths: $216\ \mu m$, $246\ \mu m$, $278\ \mu m$ (colloid protectants), $420\ \mu m$ (anthocyanins) and $520\ \mu m$ (anthocyanins). The wavelength data are exclusively those that would be used to characterize each of the samples. Of the 100 UV-visible samples collected, 60 were used to train a MLP classifier with a mean square error cost function to identify each wine and 40 samples to test it. The best classifier with two layers of hidden neurons and a 5-12-24-7 structure (5 input neurons, 12 neurons in the first hidden layer, 24 neurons in the second hidden layer and 7 neurons in the output layer) obtained 100% correct results when tested. When the same samples were used to construct a 5-15-36-7 structure FNN classifier, it also obtained a perfect classification. Of the samples obtained by NIR, 70 would be used for training and the rest for testing, also obtaining 100% of correctly classified cases with MLP 7-24-24-7 and FNN 7-21-45-7 networks. When comparing both networks they observed that FNN generalized better as the samples deteriorated over time or aged.

Other researches have shown that they can identify wines according to their aging time. In 2008, Yu et al. [129] succeeded in classifying 147 bottles of rice wine into three different categories: aged 1, 3 or 5 years. They measured by NIR at different frequencies and reduced by PCA these data taken directly from the spectrum to 10 principal components (although the first 3 components already managed to explain 92.82% of the variation in the data). They then applied comparatively LDA and SVM using LOO to make design parameter decisions. Using one third of the samples for training and the rest to test the model, they achieved 98% accuracy for the SVM models and a lower percentage, namely 93.5%, for the LDA models.

As we have already highlighted above, an important group of investigations do not use the complete spectra directly as input features, but choose to first quantify a series of compounds and then use the value of these concentrations in the classifiers. Most of the research seeks to quantify anthocyanins, a group of polyphenols present in wine. Its justification as a wine discriminant has been shown by different studies [77, 138, 139, 140, 141, 142], because the grape variety, climate, soil conditions and winemaking process all influence the anthocyanin composition of the final wine.

The research of Costa et al. [130] shows the feasibility of classifying different wines of at least 75% Syrah variety from the 2009 and 2010 vintages according to one of their two possible origins. For this purpose, they collected 26 samples of wines from Mendoza (in Argentina) and 11 samples from the Central Valley (in Chile), and represented them through 20 features that contemplate color, total polyphenols and the concentration of several anthocyanins[32] through HPLC-ESI-DAD-MS and colorimetry as indicated in Llobodanin et al. [143]. After normalizing the data between 0 and 1, they are subjected to a process of extraction of the most relevant features through two different techniques: on the one hand Chi-squared [144] and, on the other hand, RF (sets of 500 CART) using $2/3$ of the data for training and $1/3$ for the final classification. With them, they separate each sample into their respective classes and obtain scores of the importance of each feature. These scores are used to select different sets of features in three classifiers that are validated by overlapping cross validation with $k = 10$: SVM, MLP and a variation of MLP called Extreme Learning Machine (ELM) [145]. The first 6 features of both lists match, although in different order: mv-3-glc, pn-3-glc, pt-3-(6-Ac)-glc, mv-3-(6-Ac)-glc, total anthocyanins, pn-3-(6-Ac)-glc. The learning algorithms hover around 93% accuracy for SVM and MLP using only 7 features in SVM and 10 in MLP with 33 neurons in a hidden layer. The result is even better, close to 98%, when using ELM with only 3 features: pn-3-glc, mv-3-glc and pt-3-(6-Ac)-glc.

Analyte (mg/l)	Argentina	Chile
cy-3-glc	0.25±0.06	0.28±0.09
dp-3-(6-Ac)-glc	0.52±0.27	0.72±0.21
dp-3-glc	1.53±1.08	2.98±1.23
mv-3-(6-p-coum)-glc	3.19±2.14	5.14±2.22
mv-3-(6-Ac)-glc	6.62±5.45	12.48±5.59
mv-3-glc	27.49±19.41	46.53±17.78
pn-3-(6-p-coum)-glc	1.23±1.21	2.58±2.64
pn-3-(6-Ac)-glc	1.34±3.93	1.91±0.82
pn-3-glc	2.5±3.35	4.28±2.44
pt-3- (6-p-coum)-glc	0.24±0.18	0.27±0.16
pt-3- (6-Ac)-glc	0.04±0.47	0.63±0.33
pt-3-glc	2.83±2.26	4.14±0.98
vitisin A	6.11±3.36	7.84±3.84

Table 25. Concentrations of anthocyanins and derivatives in Costa et al. research [130].

In 2016, Sen and Tokatli [131] choose to characterize single-varietal red and white wines by measuring their color, pH, anthocyanin content, and UV to visible spectrum ($380\ nm$ to $680\ nm$ wavelengths), after classifying them by PCA and Orthogonal Partial Least Squares Regression Discriminant Analysis (OPLS-DA) according to two different classifications: the variety of grapes used and their age. The 63 red wine samples and 28 white wine samples are summarized in Table 24, all Turkish wines.

Variety	Type	Harvest year	Number of samples	Variety	Type	Harvest year	Number of samples
Boğazkere	Red	2006	1	Syrah	Red	2006	3
		2007	2			2007	5
		2008	3			2008	3
		2009	2			2009	2
Cabernet Sauvignon	Red	2006	2	Emir	White	2007	3
		2007	5			2008	3
		2008	1			2009	3
Kalecik Karası	Red	2006	5	Chardonnay	White	2006	2

[32] The anthocyanins are: cy-3-glc, dp-3-(6-Ac)-glc, dp-3-glc, mv-3-(6-p-coum)-glc, mv-3-(6-Ac)-glc, mv-3-glc, pn-3-(6-p-coum)-glc, pn-3-(6-Ac)-glc, pn-3-glc, pt-3- (6-p-coum)-glc, pt-3- (6-Ac)-glc, pt-3-glc and vitisin A.

		2007	6			2007	3
		2008	3			2008	2
Merlot	Red	2006	2			2009	3
		2007	4	**Moscato**	White	2006	2
		2008	1			2007	2
		2009	2			2008	3
Öküzgözu **Öküzgözu**	Red	2006	3			2009	2
		2007	3				
		2008	2				
		2009	3				

Table 26. Number of samples by variety and vintage in Sen and Tokatli's research [131].

The samples were characterized, on the one hand, by spectrometry in the UV to visible range, thus obtaining the general color parameters, and on the other hand, the different values of the spectrum at each wavelength analyzed, its first derivative and its second derivative. In addition, the quantification of anthocyanins and their derivatives was obtained by HPLC: mv-3-glc (used for calibration), dp-3-glc, pt-3-glc, pn-3-glc, dp-3-(6-Ac)-glc, pt-3-(6-Ac)-glc, pn-3-(6-Ac)-glc, mv-3-(6-Ac)-glc, mv-3-(6-p-coum)-glc, dp-3-(6-p-coum)-glc.

The classifiers were validated by LOO, obtaining by PCA a score $R^2 = 0{,}76$ with only two main components for red wines. When they applied OPLS-DA to the anthocyanin contents, it was possible to differentiate the red wines of 2006-2007 from those of later years, and by applying the same technique with the complete spectra (and their derivatives) it was possible to differentiate the native Turkish varieties from those that were not.

In 2014, Gómez-Meire et al. [132] took a total of 42 samples of white wines from three areas of Galicia and 4 grape varieties (Albariño, Treixadura, Loureira and Dona Branca). All of them had undergone the same vinification process. Trhough GC coupled respectively to Flame Ionization Detector (FID), Flame Photometric Detector (FPD) and MS they identified and quantified 41 VOCs and 7 more added characteristics (monoterpenes, alcohols, acetates, sulfides, esters and fatty acids). After normalizing the data, they applied different ML techniques to classify the wines according to their combination of variety and origin. In this case, classifiers were built with PCA, LDA, polynomial kernel SVM, sets of 10 CART (RF), MLP, kNN and NB. The data were checked by overlapping cross-validation with $k = 10$. PCA succeeded in reducing the number of components to 15 principal components that explained 95.6% of the variance of the data but did not achieve good rankings. LDA managed to separate 100% of the data and RF manages to perfectly classify all samples. However, if instead of using all the data only some of the feature families were used, MLP seemed to be the most robust classifier.

Going back another couple of years, in 2012, Cabrita et al. [133] used reverse-phase HPLC-DAD to characterize non-flavonoid compounds[33] from 81 samples of Trincadeira, Aragonez, Cabernet Sauvignon, Alfrocheiro, Castelão, Touriga Nacional and Alicante Bouschet varietals before and after malolactic fermentation. The application of PCA showed that two principal components accounted for 86.53% of the variance of the data. In search of a classification according to variety and whether or not malolactic fermentation has taken place, they apply CA to the data reduced to two principal components. Thus, they analyze the variability measured by each group of variables and try to quantify their contribution to the groupings. With the same purpose, they create two Kohonen's Self-Organizing Maps (SOM) of 9x9 neurons, one with the original features

[33] Gallic acid, protocatechic acid, protocatechic aldehyde, vanillic acid, caftaric acid, syringic acid, p-coumaric acid and ferulic acid.

without normalization and the other one after been z-normalized. The maps revealed a significant clustering of samples according to variety. Among the varieties studied, only Cabernet Sauvignon, Trincadeira and Alfrocheiro showed significant differences according to whether fermentation was performed or not. In these cases, gallic acid seems to be a clear indicator of fermentation. Finally they try to apply CART to the data to classify them according to fermentation. The tree is able to correctly classify 76% of the samples through two decisions involving protocatechuic and caftaric acids.

In 2011 Gutiérrez et al. [134] used samples of red wines from the same Chilean valley from the 2001 to 2004 vintages, and after characterizing each sample by the concentration of 9 anthocyanins, they classified them according to their variety. They took data from a total of 399 samples, of which 76 were Merlot, 95 Carménère and 228 Cabernet Sauvignon. Anthocyanins were quantified by HPLC, taking the samples immediately after the end of malolactic fermentation, and a logarithmic scale was preferred to reflect them. The anthocyanins measured were: dp-3-glc, cy-3-glc, pt-3-glc, pn-3-glc, mv-3-glc, pn-3-(6-Ac)-glc, mMv-3-(6-Ac)-glc, pn-3-(6-p-coum)-glc and mv-3-(6-p-coum)-glc. Three Bayesian classifiers were constructed to guess the variety of each sample. The second proposed model showed the lowest error of the three, misclassifying only 4% of the cases when validated by LOO.

In 2009 Kruzlikova et al. [135] used VOCs measured through GC-MS to characterize Slovakian single-varietal white wines. They applied different techniques but especially MLP to classify them according to their variety, according to their vintage and according to their producer. Their study used wines of three different varietal types (Riesling, Veltliner and Mueller) and three vintages (1996, 1997 and 1998) from 5 different producers (Nitra, Krtíš, Pezinok, Dvory and Modra) from southwestern Slovakia. From the grape varieties used they chose 20 VOCs[34] to analyze and formed two datasets. A first smaller set of 36 samples using only 4 producers (Modra was excluded) and one sample from each vintage and variety; and a second larger set of 87 samples, taking into account all 5 producers.

The first dataset did not use all the VOCs analyzed but only 19 of them, and was used to create three different classifiers according to variety, vintage and producer. The classifiers were MLP networks with sigmoidal neurons and mean square error cost function that were validated by LOO. A summary of the results can be found in Table 25. After the poor results they decided to perform a previous selection of the features to be used through the analysis of the variability that each of them offered in the error, the so-called forward and backward feature selection. This way, they obtained a scored ranking of the VOCs, which they confirmed by means of analysis of variance (ANOVA). Terpenes seemed to be especially important when sorting wines according to variety and very little according to vintage and producer, which, however, showed a greater dependence on alcohols and esters respectively. Reducing the input VOCs to the 7 most important ones improved the classification results.

The second dataset, due to being larger, was divided into 72 training and 15 test samples. The results of applying analogous steps to the first set to create MLP classifiers and applying other reference classifiers can be seen in Table 25.

[34] VOCs analyzed: Ethylhexanoate, (E)-3-Hexen-1-ol, (Z)-3-Hexen-1-ol, (E)-2-Hexen-1-ol, Ethyl 2-hydroxy-3-methylbutanoate, (E)-Furan linalool oxide, neroloxide, (Z)-Furan linalool oxide, Ethyl 3-hydroxy-butanoate, Linalool, α-Terpineol, Citronellol, 2-Ethylphenylacetate, Geraniol, 3,7-Dimethyl-1,5-octadien-3,7-diol, 1-Hexanol, Diethyl succinate, Hexanoic acid, Benzyl alcohol, 2-Phenylethanol.

Dataset: No. of training samples	No. of samples for validation	Classifier output	Classifier Type	Accuracy %
First: 35	LOO	Variety	MLP 19-2-3	69.4
		Producer	MLP 19-2-4	69.4
		Add	MLP 19-2-3	80.6
		Variety	MLP 7-3-3	100
		Producer	MLP 7-4-4	97.2
		Add	MLP 7-3-3	100
Second: 72	15	Variety	MLP 15-5-3	100
		Producer	MLP 17-11-5	100
		Add	MLP 14-8-3	93.3
		Variety	LDA	53.3
		Producer	LDA	93.3
		Add	LDA	86.7
		Variety	kNN	66.7
		Producer	kNN	86.7
		Add	kNN	86.7

Table 27. Results of the different classifiers in Kruzlicova et al. research [135].

Somewhat later is the work of Astray et al. [146] in 2010, which attempts to conclude which winemaking process have followed each wine. They analyzed red *Vinhos Verdes* from a winery in northern Portugal from 2000, 2001, 2003 and 2004. The wines were characterized by year, type of fining, pH, three absortances, color, total anthocyanins, ionization index, Folin-Coicalteau index and catechin concentration. Some color, aroma and flavor assesments in the range of $[0,8]$ are also obtained thanks to the scores given by 8 tasters according to the OIV tasting sheet. Samples are classified using MLP with sigmoidal neurons and a hidden layer consisting of 9 neurons. After reserving 10% of the samples for validation, it was found that the network correctly classified all samples by differentiating between 3 types of fermentation. The pH and color patterns appeared to be the most significant elements.

A final and completely different line of work is the use of so-called artificial noses to expose samples to them. This type of small, portable devices are usually made by doping polymeric materials in a controlled manner. A recent example of this type of research can be found in the work of Lozano et al. [147] from 2018. The authors use an electronic nose based on 16 sensors, as described by Horrillo et al. [148], and PCA and PNN techniques to classify 17 Tempranillo varietal wines from Madrid Region according to barrel aging time and barrel type. Each sample is kept at 30 °C for 30 minutes to generate some vapor from the liquid, and then this vapor is redirected to the electronic nose through Nitrogen gas flow of 200 ml/min for 20 minutes. During that time the value of the resistances at each sensor is measured. PCA shows a clear separation between the data, where with only two principal components the groupings are highly differentiated. A first classifier serves to differentiate whether the wines were taken before aging, after 3, 6 or 12 months of barrel aging. A second classifier tried to classify the wines depending on whether they were aged in French or American oak barrels. Applying PNN with three principal components they could classified correctly 97% and 84% of the samples respectively (validated with LOO).

An older work, is the paper by Yamazaki et al. [149] in 2001. In this research, an artificial nose is used to expose it to samples of the same wine (Almaden, Brazil) and capture its aroma signal and then classify those signals according to age (1995, 1996 or 1997) using MLP. The artificial nose prototype is the one described by Santos [150] which consists of six different conductive polymer sensors constructed with different types of dopants. For each vintage, 3 shots are taken, and at each shot the resistive data offered by the sensors are recorded with a sampling period of 0.5 s for 5 min. This makes a total of 5400 samples with 1800 for each vintage. 50% of the samples are used for training, 25% for validation and the remaining 25% for testing the classifiers, trying to ensure that each of the vintages is present in the division of the samples in a proportional way, especially in the test one. After normalizing the data with respect to the smallest and largest value of all samples, 5 MLP classifiers were constructed with sigmoidal neurons and 4, 8, 12, 16 and 20 hidden neurons respectively. After performing 10 rounds of testing with each topology starting from random initial weights and seeking to minimize the quadratic error, the topology that gave the best results was the one with 16 hidden neurons with 9% classification error and two rounds with perfect classification. Furthermore, their research shows that if one switches to using the entire signal as input and modifies the neural network to use time-delayed neurons, the 9 time series with their respective 6 inputs for each sensor can be used directly. Using 3 signals for training, 3 for validation and 3 for test the result for any classifier, regardless of the number of hidden neurons, was 0% failure.

Los capítulos 5 y 6 están sujetos a confidencialidad por el autor

Chapter VII.

Red Wines Classifiers and Results

7 Classification of red wines and Results

7.1 PCA

In search of a simple representation of the data and of a possible reduction of the number of features to be used in subsequent techniques, we proceed to first apply the PCA technique following exactly the steps indicated in section 2.2 in the Matlab R2019b environment. By applying PCA to the data collected from the 96 samples (recall that PCA does not require using the scores) we get the eigenvalues, ordered from highest to lowest, shown in the Table 39.

No. of eigenvalue	Eigenvalue value	Cumulative variance	No. of eigenvalue	Eigenvalue value	Cumulative variance
1	19.797	31.93%	**32**	0.125	98.21%
2	10.542	48.93%	**33**	0.120	98.40%
3	4.913	56.86%	**34**	0.116	98.59%
4	4.116	63.49%	**35**	0.107	98.76%
5	2.738	67.91%	**36**	0.094	98.91%
6	2.437	71.84%	**37**	0.080	99.04%
7	1.900	74.91%	**38**	0.072	99.16%
8	1.835	77.86%	**39**	0.062	99.26%
9	1.453	80.21%	**40**	0.056	99.35%
10	1.392	82.45%	**41**	0.052	99.43%
11	1.235	84.45%	**42**	0.048	99.51%
12	0.931	85.95%	**43**	0.044	99.58%
13	0.913	87.42%	**44**	0.038	99.64%
14	0.742	88.62%	**45**	0.029	99.69%
15	0.646	89.66%	**46**	0.028	99.73%
16	0.602	90.63%	**47**	0.024	99.77%
17	0.568	91.55%	**48**	0.023	99.81%
18	0.526	92.39%	**49**	0.021	99.84%
19	0.485	93.18%	**50**	0.017	99.87%
20	0.382	93.79%	**51**	0.016	99.89%
21	0.355	94.36%	**52**	0.013	99.91%
22	0.324	94.89%	**53**	0.012	99.93%
23	0.302	95.37%	**54**	0.010	99.95%
24	0.279	95.82%	**55**	0.009	99.96%
25	0.258	96.24%	**56**	0.006	99.97%
26	0.221	96.60%	**57**	0.005	99.98%
27	0.213	96.94%	**58**	0.004	99.99%
28	0.193	97.25%	**59**	0.003	99.99%
29	0.165	97.52%	**60**	0.002	100.00%
30	0.158	97.77%	**61**	0.002	100.00%
31	0.144	98.01%	**62**	0.001	100.00%

Table 39. Eigenvalues associated with the PCA principal components and their contribution to the total variance.

The cumulative variance (Figure 47) is represented by the ratio of the cumulative eigenvalues of the components to be used with respect to the total sum of the eigenvalues as indicated in section 2.1. The classifiers that we then build will be trained with several compressions of the data to compare their results. The first option will be not to compress the data and use the original features, without applying PCA. The second is to work with the principal components (new features) that accumulate 99% of the variance, i.e. 37 principal components. This already represents a reduction of 25 features (40.3% reduction). The second option is to take the principal components that account for 95% of the variance, i.e. 23 principal components, which represents a reduction of 39 features (62.9% reduction). Finally, the third option is to use only the components that represent 90% of the variance, i.e. 16 principal components, obtaining a reduction of 46 characteristics (74.2% reduction). From now on, when these compressions are used we will refer to them as PCA compressions at 99%, 95% and 90%.

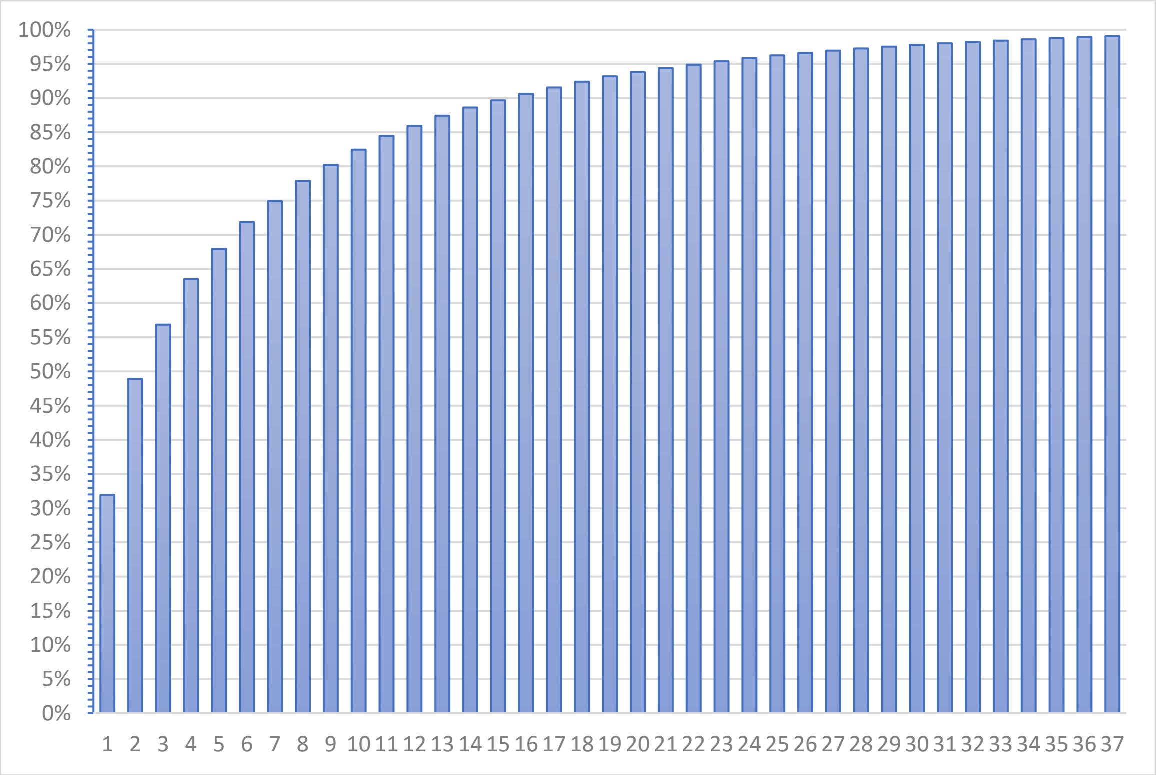


Figure 47. Total cumulative variance of the first principal components.

We now seek to get a visual representation of the first dataset (first taster's data) with the first two and three principal components. To do this, we obtain the coordinates of each sample with only two components and symbolize them according to the ratings issued by the first taster (according to the categories in the Table 37). The result is shown in Figure 48. It is obvious that it is not possible to separate the different categories.

Representing it with 3 principal components we obtain the results shown in Figure 49, where we can see once again that it is not possible to separate the different categories. This lack of separability into a few components was somewhat to be expected, since these do not even account for 60% of the total variance of the data and the judgments made seem to be much more complex and involve many more dimensions that are independent of each other. The representation of the second taster's dataset gives similar results, and the information from the three components is insufficient to represent the samples in a simple way due to the high error committed.

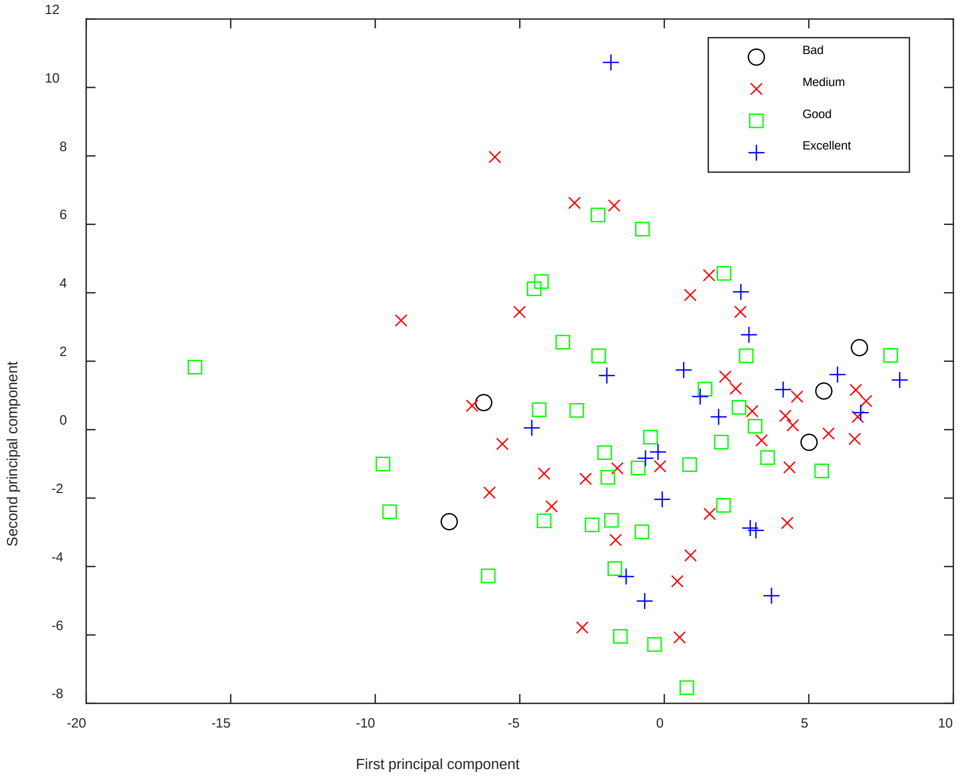


Figure 48. Representation of the first taster's scores on the first two PCA principal components.

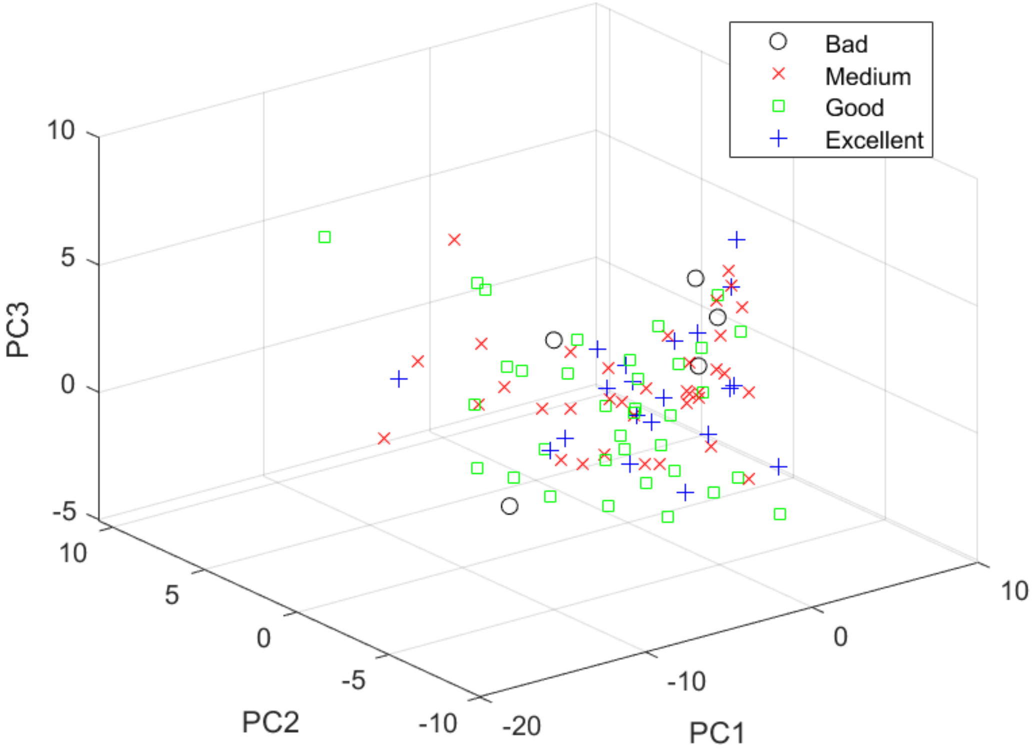


Figure 49. Representation of the first taster's scores on the first three PCA principal components.

In addition to compressing the data and obtaining a simple representation, we can use PCA to select some features among the original ones. By analyzing the eigenvectors of the principal components, we can see which features are most taken into account in these eigenvectors and select them as the most important ones as Jauregui does [196]. The 16 eigenvectors related to the first 16 principal components indicate which original features will be used and how to combine them to compose the new components. Their absolute value can be taken as an indicator of the importance of each feature in the new component (see Table 40 and Table 41).

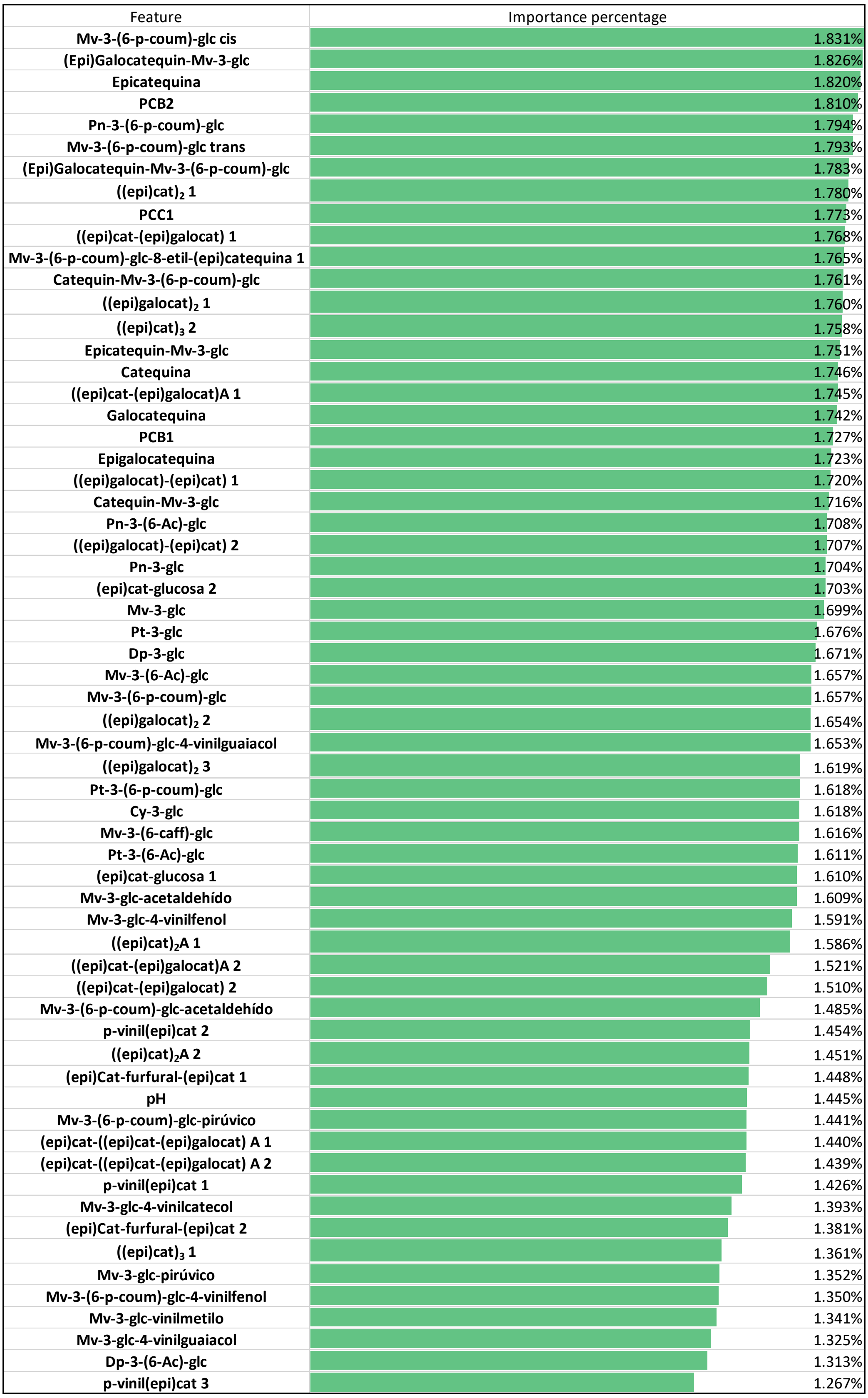
	Eigenvalues (importance of each component)	19.797	10.542	4.913	4.116	2.738	2.437	1.900	1.835
		PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Original Feature Weight 1	**pH**	0.075	0.051	0.084	0.168	0.280	0.089	0.122	0.079
Original Feature Weight 2	**Dp-3-glc**	0.196	0.091	0.097	0.033	0.014	0.102	0.023	0.010
Original Feature Weight 3	**Cy-3-glc**	0.172	0.059	0.157	0.007	0.021	0.165	0.022	0.004
Original Feature Weight 4	**Pt-3-glc**	0.197	0.108	0.073	0.071	0.004	0.079	0.010	0.016
Original Feature Weight 5	**Pn-3-glc**	0.180	0.079	0.123	0.067	0.003	0.166	0.062	0.068
Original Feature Weight 6	**Mv-3-glc**	0.191	0.126	0.064	0.083	0.025	0.053	0.053	0.053
Original Feature Weight 7	**Dp-3-(6-Ac)-glc**	0.019	0.045	0.141	0.311	0.225	0.145	0.073	0.171
Original Feature Weight 8	**Pt-3-(6-Ac)-glc**	0.136	0.122	0.060	0.080	0.282	0.099	0.013	0.002
Original Feature Weight 9	**Pn-3-(6-Ac)-glc**	0.141	0.110	0.111	0.122	0.038	0.081	0.122	0.155
Original Feature Weight 10	**Mv-3-(6-Ac)-glc**	0.185	0.143	0.011	0.078	0.044	0.039	0.057	0.048
Original Feature Weight 11	**Mv-3-(6-caff)-glc**	0.153	0.147	0.016	0.057	0.131	0.055	0.063	0.023
Original Feature Weight 12	**Pt-3-(6-p-coum)-glc**	0.184	0.145	0.006	0.063	0.127	0.006	0.048	0.001
Original Feature Weight 13	**Pn-3-(6-p-coum)-glc**	0.161	0.160	0.066	0.104	0.084	0.062	0.093	0.126
Original Feature Weight 14	**Mv-3-(6-p-coum)-glc**	0.185	0.141	0.014	0.102	0.080	0.005	0.074	0.031
Original Feature Weight 15	**Mv-3-glc-acetaldehído**	0.061	0.116	0.063	0.141	0.076	0.225	0.064	0.377
Original Feature Weight 16	**Mv-3-glc-vinilmetilo**	0.031	0.042	0.133	0.282	0.116	0.176	0.040	0.173
Original Feature Weight 17	**Mv-3-glc-pirúvico**	0.025	0.058	0.179	0.308	0.029	0.246	0.136	0.103
Original Feature Weight 18	**Mv-3-glc-4-vinilfenol**	0.046	0.169	0.102	0.054	0.081	0.266	0.112	0.189
Original Feature Weight 19	**Mv-3-glc-4-vinilcatecol**	0.078	0.066	0.060	0.002	0.233	0.344	0.047	0.145
Original Feature Weight 20	**Mv-3-(6-p-coum)-glc cis**	0.138	0.118	0.104	0.142	0.170	0.186	0.032	0.028
Original Feature Weight 21	**Mv-3-(6-p-coum)-glc trans**	0.134	0.125	0.132	0.152	0.129	0.160	0.004	0.051
Original Feature Weight 22	**Mv-3-glc-4-vinilguaiacol**	0.012	0.150	0.264	0.001	0.187	0.210	0.043	0.171
Original Feature Weight 23	**Mv-3-(6-p-coum)-glc-acetaldehído**	0.043	0.081	0.075	0.108	0.205	0.265	0.089	0.323
Original Feature Weight 24	**Mv-3-(6-p-coum)-glc-pirúvico**	0.064	0.022	0.267	0.203	0.121	0.114	0.119	0.063
Original Feature Weight 25	**Mv-3-(6-p-coum)-glc-4-vinilfenol**	0.018	0.136	0.271	0.026	0.266	0.152	0.025	0.059
Original Feature Weight 26	**Catequin-Mv-3-glc**	0.104	0.106	0.239	0.142	0.110	0.004	0.056	0.150
Original Feature Weight 27	**Epicatequin-Mv-3-glc**	0.124	0.093	0.239	0.118	0.100	0.051	0.032	0.198
Original Feature Weight 28	**Mv-3-(6-p-coum)-glc-4-vinilguaiacol**	0.075	0.168	0.124	0.094	0.002	0.243	0.115	0.045
Original Feature Weight 29	**(Epi)Galocatequin-Mv-3-glc**	0.148	0.143	0.146	0.074	0.079	0.141	0.134	0.076
Original Feature Weight 30	**Catequin-Mv-3-(6-p-coum)-glc**	0.158	0.082	0.172	0.029	0.109	0.054	0.092	0.181
Original Feature Weight 31	**(Epi)Galocatequin-Mv-3-(6-p-coum)-glc**	0.146	0.080	0.142	0.062	0.111	0.133	0.059	0.191
Original Feature Weight 32	**Mv-3-(6-p-coum)-glc-8-etil-(epi)catequina 1**	0.161	0.086	0.120	0.095	0.085	0.066	0.046	0.097
Original Feature Weight 33	**Catequina**	0.146	0.124	0.047	0.094	0.218	0.009	0.097	0.128
Original Feature Weight 34	**Epicatequina**	0.133	0.103	0.093	0.149	0.227	0.038	0.222	0.099
Original Feature Weight 35	**PCB1**	0.174	0.113	0.065	0.074	0.095	0.006	0.078	0.179
Original Feature Weight 36	**PCB2**	0.129	0.120	0.096	0.128	0.231	0.043	0.190	0.122
Original Feature Weight 37	**((epi)cat)$_2$ 1**	0.167	0.155	0.075	0.075	0.126	0.021	0.027	0.074
Original Feature Weight 38	**PCC1**	0.159	0.134	0.129	0.047	0.097	0.047	0.164	0.041
Original Feature Weight 39	**((epi)cat)$_3$ 1**	0.012	0.176	0.114	0.264	0.028	0.081	0.110	0.182
Original Feature Weight 40	**((epi)cat)$_3$ 2**	0.169	0.148	0.080	0.062	0.062	0.080	0.008	0.098
Original Feature Weight 41	**Galocatequina**	0.157	0.140	0.056	0.063	0.173	0.088	0.142	0.014
Original Feature Weight 42	**Epigalocatequina**	0.164	0.083	0.025	0.126	0.106	0.140	0.133	0.134
Original Feature Weight 43	**((epi)galocat)$_2$ 1**	0.126	0.164	0.133	0.094	0.063	0.049	0.254	0.025
Original Feature Weight 44	**((epi)galocat)$_2$ 2**	0.089	0.177	0.097	0.143	0.007	0.048	0.232	0.096
Original Feature Weight 45	**((epi)galocat)$_2$ 3**	0.156	0.160	0.045	0.010	0.133	0.041	0.172	0.037
Original Feature Weight 46	**((epi)cat-(epi)galocat) 1**	0.146	0.164	0.067	0.061	0.053	0.056	0.214	0.091
Original Feature Weight 47	**((epi)cat-(epi)galocat) 2**	0.118	0.127	0.010	0.088	0.006	0.128	0.231	0.007
Original Feature Weight 48	**((epi)galocat)-(epi)cat) 1**	0.131	0.172	0.099	0.075	0.004	0.084	0.086	0.077
Original Feature Weight 49	**((epi)galocat)-(epi)cat) 2**	0.121	0.206	0.093	0.071	0.010	0.053	0.136	0.104
Original Feature Weight 50	**(epi)cat-glucosa 1**	0.176	0.087	0.002	0.155	0.047	0.053	0.006	0.051
Original Feature Weight 51	**(epi)cat-glucosa 2**	0.105	0.129	0.056	0.265	0.064	0.079	0.128	0.048
Original Feature Weight 52	**((epi)cat)$_2$A 1**	0.107	0.134	0.023	0.167	0.042	0.013	0.197	0.114
Original Feature Weight 53	**((epi)cat)$_2$A 2**	0.076	0.049	0.179	0.170	0.012	0.006	0.067	0.199
Original Feature Weight 54	**((epi)cat-(epi)galocat)A 1**	0.146	0.115	0.026	0.129	0.205	0.127	0.115	0.049
Original Feature Weight 55	**((epi)cat-(epi)galocat)A 2**	0.116	0.053	0.196	0.089	0.004	0.234	0.047	0.014
Original Feature Weight 56	**(epi)cat-((epi)cat-(epi)galocat) A 1**	0.014	0.201	0.221	0.143	0.079	0.080	0.190	0.126
Original Feature Weight 57	**(epi)cat-((epi)cat-(epi)galocat) A 2**	0.029	0.161	0.129	0.073	0.088	0.099	0.298	0.089
Original Feature Weight 58	**p-vinil(epi)cat 1**	0.023	0.115	0.176	0.033	0.191	0.119	0.083	0.249
Original Feature Weight 59	**p-vinil(epi)cat 2**	0.106	0.007	0.129	0.042	0.033	0.040	0.291	0.113
Original Feature Weight 60	**p-vinil(epi)cat 3**	0.064	0.008	0.035	0.089	0.169	0.132	0.026	0.169
Original Feature Weight 61	**(epi)Cat-furfural-(epi)cat 1**	0.007	0.202	0.166	0.151	0.024	0.097	0.169	0.045
Original Feature Weight 62	**(epi)Cat-furfural-(epi)cat 2**	0.004	0.201	0.154	0.071	0.052	0.049	0.215	0.119

Table 40. Contribution of each original feature in the first 16 PCA principal components. Part I.

	Eigenvalues (importance of each component)	1.453	1.392	1.235	0.931	0.913	0.742	0.646	0.602
		PC9	PC10	PC11	PC12	PC13	PC14	PC15	PC16
Original Feature Weight 1	**pH**	0.070	0.064	0.037	0.066	0.144	0.645	0.112	0.019
Original Feature Weight 2	**Dp-3-glc**	0.059	0.121	0.012	0.134	0.050	0.065	0.042	0.027
Original Feature Weight 3	**Cy-3-glc**	0.096	0.176	0.024	0.228	0.076	0.019	0.125	0.023
Original Feature Weight 4	**Pt-3-glc**	0.076	0.088	0.020	0.067	0.015	0.076	0.030	0.022
Original Feature Weight 5	**Pn-3-glc**	0.104	0.111	0.007	0.093	0.000	0.022	0.049	0.112
Original Feature Weight 6	**Mv-3-glc**	0.066	0.062	0.005	0.004	0.019	0.061	0.009	0.034
Original Feature Weight 7	**Dp-3-(6-Ac)-glc**	0.054	0.190	0.052	0.011	0.199	0.088	0.035	0.085
Original Feature Weight 8	**Pt-3-(6-Ac)-glc**	0.046	0.033	0.109	0.006	0.148	0.002	0.011	0.104
Original Feature Weight 9	**Pn-3-(6-Ac)-glc**	0.026	0.017	0.004	0.208	0.133	0.127	0.210	0.066
Original Feature Weight 10	**Mv-3-(6-Ac)-glc**	0.019	0.029	0.038	0.100	0.075	0.000	0.070	0.018
Original Feature Weight 11	**Mv-3-(6-caff)-glc**	0.051	0.133	0.039	0.036	0.033	0.085	0.213	0.018
Original Feature Weight 12	**Pt-3-(6-p-coum)-glc**	0.029	0.001	0.011	0.064	0.024	0.055	0.016	0.041
Original Feature Weight 13	**Pn-3-(6-p-coum)-glc**	0.084	0.069	0.018	0.082	0.017	0.045	0.006	0.041
Original Feature Weight 14	**Mv-3-(6-p-coum)-glc**	0.023	0.007	0.014	0.075	0.043	0.052	0.018	0.002
Original Feature Weight 15	**Mv-3-glc-acetaldehído**	0.233	0.195	0.174	0.100	0.180	0.042	0.055	0.100
Original Feature Weight 16	**Mv-3-glc-vinilmetilo**	0.209	0.061	0.200	0.067	0.116	0.190	0.066	0.176
Original Feature Weight 17	**Mv-3-glc-pirúvico**	0.012	0.254	0.004	0.050	0.097	0.182	0.109	0.036
Original Feature Weight 18	**Mv-3-glc-4-vinilfenol**	0.051	0.212	0.108	0.068	0.324	0.146	0.170	0.175
Original Feature Weight 19	**Mv-3-glc-4-vinilcatecol**	0.137	0.145	0.153	0.012	0.265	0.041	0.026	0.033
Original Feature Weight 20	**Mv-3-(6-p-coum)-glc cis**	0.061	0.064	0.211	0.071	0.065	0.127	0.180	0.113
Original Feature Weight 21	**Mv-3-(6-p-coum)-glc trans**	0.053	0.027	0.248	0.039	0.060	0.033	0.223	0.040
Original Feature Weight 22	**Mv-3-glc-4-vinilguaiacol**	0.068	0.091	0.021	0.001	0.030	0.041	0.164	0.132
Original Feature Weight 23	**Mv-3-(6-p-coum)-glc-acetaldehído**	0.245	0.294	0.010	0.046	0.144	0.032	0.089	0.165
Original Feature Weight 24	**Mv-3-(6-p-coum)-glc-pirúvico**	0.096	0.160	0.095	0.004	0.193	0.125	0.034	0.165
Original Feature Weight 25	**Mv-3-(6-p-coum)-glc-4-vinilfenol**	0.010	0.147	0.029	0.088	0.026	0.184	0.159	0.062
Original Feature Weight 26	**Catequin-Mv-3-glc**	0.128	0.031	0.097	0.129	0.179	0.108	0.220	0.020
Original Feature Weight 27	**Epicatequin-Mv-3-glc**	0.094	0.057	0.040	0.112	0.150	0.101	0.146	0.061
Original Feature Weight 28	**Mv-3-(6-p-coum)-glc-4-vinilguaiacol**	0.262	0.068	0.183	0.159	0.105	0.001	0.195	0.100
Original Feature Weight 29	**(Epi)Galocatequin-Mv-3-glc**	0.056	0.105	0.019	0.108	0.063	0.006	0.139	0.019
Original Feature Weight 30	**Catequin-Mv-3-(6-p-coum)-glc**	0.086	0.161	0.017	0.142	0.073	0.129	0.175	0.019
Original Feature Weight 31	**(Epi)Galocatequin-Mv-3-(6-p-coum)-glc**	0.123	0.211	0.044	0.169	0.077	0.107	0.074	0.179
Original Feature Weight 32	**Mv-3-(6-p-coum)-glc-8-etil-(epi)catequina 1**	0.052	0.036	0.222	0.172	0.184	0.124	0.125	0.093
Original Feature Weight 33	**Catequina**	0.221	0.049	0.010	0.032	0.120	0.058	0.208	0.076
Original Feature Weight 34	**Epicatequina**	0.052	0.114	0.120	0.097	0.149	0.025	0.149	0.119
Original Feature Weight 35	**PCB1**	0.082	0.071	0.031	0.002	0.043	0.072	0.068	0.248
Original Feature Weight 36	**PCB2**	0.107	0.055	0.067	0.106	0.192	0.021	0.080	0.137
Original Feature Weight 37	**((epi)cat)$_2$ 1**	0.146	0.071	0.060	0.021	0.055	0.049	0.056	0.037
Original Feature Weight 38	**PCC1**	0.162	0.040	0.032	0.033	0.076	0.017	0.030	0.169
Original Feature Weight 39	**((epi)cat)$_3$ 1**	0.016	0.049	0.112	0.152	0.047	0.135	0.104	0.042
Original Feature Weight 40	**((epi)cat)$_3$ 2**	0.171	0.040	0.017	0.022	0.108	0.023	0.063	0.094
Original Feature Weight 41	**Galocatequina**	0.057	0.092	0.043	0.000	0.096	0.001	0.221	0.004
Original Feature Weight 42	**Epigalocatequina**	0.088	0.018	0.064	0.076	0.083	0.153	0.149	0.105
Original Feature Weight 43	**((epi)galocat)$_2$ 1**	0.106	0.139	0.054	0.053	0.009	0.056	0.068	0.059
Original Feature Weight 44	**((epi)galocat)$_2$ 2**	0.144	0.168	0.042	0.059	0.039	0.115	0.085	0.182
Original Feature Weight 45	**((epi)galocat)$_2$ 3**	0.003	0.004	0.006	0.029	0.093	0.095	0.018	0.061
Original Feature Weight 46	**((epi)cat-(epi)galocat) 1**	0.079	0.171	0.045	0.097	0.105	0.002	0.016	0.067
Original Feature Weight 47	**((epi)cat-(epi)galocat) 2**	0.049	0.049	0.025	0.017	0.376	0.157	0.031	0.325
Original Feature Weight 48	**((epi)galocat)-(epi)cat) 1**	0.221	0.098	0.105	0.000	0.059	0.018	0.122	0.121
Original Feature Weight 49	**((epi)galocat)-(epi)cat) 2**	0.081	0.126	0.012	0.044	0.045	0.024	0.051	0.174
Original Feature Weight 50	**(epi)cat-glucosa 1**	0.192	0.135	0.035	0.006	0.027	0.007	0.136	0.126
Original Feature Weight 51	**(epi)cat-glucosa 2**	0.117	0.135	0.078	0.252	0.048	0.030	0.250	0.074
Original Feature Weight 52	**((epi)cat)$_2$A 1**	0.126	0.097	0.187	0.156	0.027	0.035	0.269	0.062
Original Feature Weight 53	**((epi)cat)$_2$A 2**	0.050	0.382	0.236	0.149	0.059	0.240	0.046	0.107
Original Feature Weight 54	**((epi)cat-(epi)galocat)A 1**	0.142	0.020	0.041	0.012	0.198	0.023	0.203	0.076
Original Feature Weight 55	**((epi)cat-(epi)galocat)A 2**	0.053	0.032	0.277	0.145	0.072	0.137	0.055	0.147
Original Feature Weight 56	**(epi)cat-((epi)cat-(epi)galocat) A 1**	0.137	0.010	0.001	0.053	0.096	0.015	0.021	0.016
Original Feature Weight 57	**(epi)cat-((epi)cat-(epi)galocat) A 2**	0.123	0.084	0.063	0.426	0.037	0.114	0.193	0.055
Original Feature Weight 58	**p-vinil(epi)cat 1**	0.118	0.095	0.250	0.319	0.163	0.192	0.180	0.003
Original Feature Weight 59	**p-vinil(epi)cat 2**	0.286	0.239	0.220	0.089	0.170	0.030	0.061	0.468
Original Feature Weight 60	**p-vinil(epi)cat 3**	0.287	0.126	0.549	0.046	0.177	0.099	0.095	0.277
Original Feature Weight 61	**(epi)Cat-furfural-(epi)cat 1**	0.113	0.132	0.084	0.250	0.144	0.215	0.106	0.067
Original Feature Weight 62	**(epi)Cat-furfural-(epi)cat 2**	0.228	0.087	0.008	0.311	0.133	0.107	0.090	0.153

Table 41. Contribution of each original feature in the first 16 PCA principal components. Part II.

If we weight each eigenvector with the value of its eigenvalue, we obtain a score that allows us to rank the original features according to their contribution in the first n components. We have reflected this score as a percentage with respect to the total points (the total of the sum of all the original features). The ranking by taking the first 16 components (see Table 42) reflects a fairly high participation of almost all the features, where the contribution of the feature that contributes the most is 1.83% (mv-3-(6-p-coum)-glc cis) and the least is 1.267%. In this ordering it does show that the p-vinyl(epi)cat 3, dp-3-(6-Ac)-glc, mv-3-glc-4-vinylguaiacol, mv-3-glc-vinylmethyl, and mv-3-(6-p-coum)-glc-4-vinylphenol features are the least important, and could be the first candidates to be discarded, but as mentioned above, its comparative contribution is still too high to do so.



Feature	Importance percentage
Mv-3-(6-p-coum)-glc cis	1.831%
(Epi)Galocatequin-Mv-3-glc	1.826%
Epicatequina	1.820%
PCB2	1.810%
Pn-3-(6-p-coum)-glc	1.794%
Mv-3-(6-p-coum)-glc trans	1.793%
(Epi)Galocatequin-Mv-3-(6-p-coum)-glc	1.783%
((epi)cat)$_2$ 1	1.780%
PCC1	1.773%
((epi)cat-(epi)galocat) 1	1.768%
Mv-3-(6-p-coum)-glc-8-etil-(epi)catequina 1	1.765%
Catequin-Mv-3-(6-p-coum)-glc	1.761%
((epi)galocat)$_2$ 1	1.760%
((epi)cat)$_3$ 2	1.758%
Epicatequin-Mv-3-glc	1.751%
Catequina	1.746%
((epi)cat-(epi)galocat)A 1	1.745%
Galocatequina	1.742%
PCB1	1.727%
Epigalocatequina	1.723%
((epi)galocat)-(epi)cat) 1	1.720%
Catequin-Mv-3-glc	1.716%
Pn-3-(6-Ac)-glc	1.708%
((epi)galocat)-(epi)cat) 2	1.707%
Pn-3-glc	1.704%
(epi)cat-glucosa 2	1.703%
Mv-3-glc	1.699%
Pt-3-glc	1.676%
Dp-3-glc	1.671%
Mv-3-(6-Ac)-glc	1.657%
Mv-3-(6-p-coum)-glc	1.657%
((epi)galocat)$_2$ 2	1.654%
Mv-3-(6-p-coum)-glc-4-vinilguaiacol	1.653%
((epi)galocat)$_2$ 3	1.619%
Pt-3-(6-p-coum)-glc	1.618%
Cy-3-glc	1.618%
Mv-3-(6-caff)-glc	1.616%
Pt-3-(6-Ac)-glc	1.611%
(epi)cat-glucosa 1	1.610%
Mv-3-glc-acetaldehído	1.609%
Mv-3-glc-4-vinilfenol	1.591%
((epi)cat)$_2$A 1	1.586%
((epi)cat-(epi)galocat)A 2	1.521%
((epi)cat-(epi)galocat) 2	1.510%
Mv-3-(6-p-coum)-glc-acetaldehído	1.485%
p-vinil(epi)cat 2	1.454%
((epi)cat)$_2$A 2	1.451%
(epi)Cat-furfural-(epi)cat 1	1.448%
pH	1.445%
Mv-3-(6-p-coum)-glc-pirúvico	1.441%
(epi)cat-((epi)cat-(epi)galocat) A 1	1.440%
(epi)cat-((epi)cat-(epi)galocat) A 2	1.439%
p-vinil(epi)cat 1	1.426%
Mv-3-glc-4-vinilcatecol	1.393%
(epi)Cat-furfural-(epi)cat 2	1.381%
((epi)cat)$_3$ 1	1.361%
Mv-3-glc-pirúvico	1.352%
Mv-3-(6-p-coum)-glc-4-vinilfenol	1.350%
Mv-3-glc-vinilmetilo	1.341%
Mv-3-glc-4-vinilguaiacol	1.325%
Dp-3-(6-Ac)-glc	1.313%
p-vinil(epi)cat 3	1.267%

Table 42. Order of importance of the original features in the first 16 principal PCA components.

If we look at what happens with a smaller number of components (Table 43 and Table 44), it is noticeable that the number of original features that contribute is still very high, since the contribution of the lowest ones is always higher or close to 50% of that of the highest contribution. It can also be seen that the same compounds are always present in the last places: p-vinyl(epi)cat 3, dp-3-(6-Ac)-glc, mv-3-glc-4-vinylguaiacol, mv-3-glc-vinylmethyl, and mv-3-(6-p-coum)-glc-4-vinylphenol, pH, mv-3-glc-pyruvic, p-vinyl(epi)cat 2, p-vinyl(epi)cat 1, (epi)Cat-furfural-(epi)cat 2.

	Importance (%) up to PC1:	Importance (%) up to PC2:	Importance (%) up to PC3:	Importance (%) up to PC4:	Importance (%) up to PC5:	Importance (%) up to PC6:	Importance (%) up to PC7:	Importance (%) up to PC8:
pH	1.07%	0.94%	0.98%	1.14%	1.33%	1.34%	1.36%	1.35%
Dp-3-glc	2.80%	2.25%	2.14%	1.98%	1.88%	1.87%	1.81%	1.75%
Cy-3-glc	2.45%	1.86%	1.93%	1.75%	1.67%	1.72%	1.66%	1.61%
Pt-3-glc	2.80%	2.33%	2.17%	2.06%	1.95%	1.91%	1.84%	1.79%
Pn-3-glc	2.57%	2.04%	2.02%	1.92%	1.81%	1.85%	1.82%	1.79%
Mv-3-glc	2.73%	2.37%	2.18%	2.09%	1.99%	1.94%	1.89%	1.86%
Dp-3-(6-Ac)-glc	0.27%	0.39%	0.62%	1.02%	1.17%	1.23%	1.23%	1.28%
Pt-3-(6-Ac)-glc	1.95%	1.85%	1.72%	1.67%	1.84%	1.83%	1.76%	1.70%
Pn-3-(6-Ac)-glc	2.02%	1.84%	1.81%	1.82%	1.75%	1.72%	1.73%	1.76%
Mv-3-(6-Ac)-glc	2.64%	2.40%	2.10%	2.01%	1.94%	1.87%	1.83%	1.79%
Mv-3-(6-caff)-glc	2.18%	2.12%	1.88%	1.78%	1.80%	1.75%	1.72%	1.67%
Pt-3-(6-p-coum)-glc	2.63%	2.40%	2.10%	1.98%	1.99%	1.89%	1.85%	1.79%
Pn-3-(6-p-coum)-glc	2.29%	2.26%	2.09%	2.04%	2.00%	1.95%	1.93%	1.93%
Mv-3-(6-p-coum)-glc	2.64%	2.39%	2.10%	2.05%	2.00%	1.90%	1.88%	1.83%
Mv-3-glc-acetaldehído	0.88%	1.13%	1.10%	1.21%	1.21%	1.33%	1.31%	1.48%
Mv-3-glc-vinilmetilo	0.44%	0.48%	0.68%	1.04%	1.08%	1.17%	1.15%	1.20%
Mv-3-glc-pirúvico	0.36%	0.52%	0.80%	1.18%	1.14%	1.28%	1.31%	1.32%
Mv-3-glc-4-vinilfenol	0.66%	1.25%	1.29%	1.24%	1.24%	1.39%	1.41%	1.46%
Mv-3-glc-4-vinilcatecol	1.11%	1.04%	1.02%	0.92%	1.09%	1.30%	1.28%	1.32%
Mv-3-(6-p-coum)-glc cis	1.97%	1.84%	1.81%	1.84%	1.89%	1.95%	1.89%	1.84%
Mv-3-(6-p-coum)-glc trans	1.92%	1.85%	1.86%	1.91%	1.92%	1.95%	1.88%	1.84%
Mv-3-glc-4-vinilguaiacol	0.16%	0.84%	1.25%	1.13%	1.24%	1.34%	1.32%	1.37%
Mv-3-(6-p-coum)-glc-acetaldehído	0.62%	0.80%	0.84%	0.92%	1.06%	1.21%	1.22%	1.36%
Mv-3-(6-p-coum)-glc-pirúvico	0.91%	0.69%	1.13%	1.32%	1.36%	1.38%	1.40%	1.38%
Mv-3-(6-p-coum)-glc-4-vinilfenol	0.25%	0.83%	1.25%	1.17%	1.35%	1.40%	1.37%	1.35%
Catequin-Mv-3-glc	1.49%	1.48%	1.75%	1.79%	1.79%	1.71%	1.68%	1.70%
Epicatequin-Mv-3-glc	1.78%	1.60%	1.86%	1.85%	1.84%	1.79%	1.74%	1.79%
Mv-3-(6-p-coum)-glc-4-vinilguaiacol	1.06%	1.50%	1.55%	1.54%	1.45%	1.57%	1.58%	1.55%
(Epi)Galocatequin-Mv-3-glc	2.11%	2.06%	2.07%	1.98%	1.94%	1.95%	1.96%	1.93%
Catequin-Mv-3-(6-p-coum)-glc	2.26%	1.85%	1.95%	1.80%	1.80%	1.75%	1.74%	1.78%
(Epi)Galocatequin-Mv-3-(6-p-coum)-glc	2.08%	1.73%	1.78%	1.70%	1.70%	1.72%	1.69%	1.74%
Iv-3-(6-p-coum)-glc-8-etil-(epi)catequina	2.30%	1.90%	1.89%	1.84%	1.81%	1.78%	1.74%	1.73%
Catequina	2.08%	1.95%	1.78%	1.75%	1.85%	1.77%	1.76%	1.77%
Epicatequina	1.89%	1.72%	1.68%	1.74%	1.85%	1.79%	1.85%	1.84%
PCB1	2.48%	2.15%	1.99%	1.91%	1.88%	1.80%	1.77%	1.81%
PCB2	1.83%	1.76%	1.72%	1.74%	1.86%	1.80%	1.84%	1.85%
((epi)cat)$_2$ 1	2.39%	2.29%	2.14%	2.04%	2.04%	1.95%	1.89%	1.87%
PCC1	2.27%	2.12%	2.09%	1.96%	1.93%	1.87%	1.90%	1.86%
((epi)cat)$_3$ 1	0.17%	0.97%	1.07%	1.36%	1.30%	1.30%	1.32%	1.37%
((epi)cat)$_3$ 2	2.41%	2.27%	2.13%	2.01%	1.95%	1.92%	1.85%	1.84%
Galocatequina	2.25%	2.13%	1.96%	1.86%	1.91%	1.89%	1.90%	1.84%
Epigalocatequina	2.33%	1.91%	1.71%	1.73%	1.72%	1.75%	1.76%	1.77%
((epi)galocat)$_2$ 1	1.79%	1.96%	1.96%	1.91%	1.86%	1.80%	1.89%	1.83%
((epi)galocat)$_2$ 2	1.27%	1.68%	1.65%	1.70%	1.61%	1.57%	1.64%	1.64%
((epi)galocat)$_2$ 3	2.22%	2.21%	2.01%	1.83%	1.85%	1.79%	1.82%	1.78%
((epi)cat-(epi)galocat) 1	2.09%	2.15%	1.99%	1.89%	1.83%	1.78%	1.84%	1.83%
((epi)cat-(epi)galocat) 2	1.68%	1.70%	1.50%	1.48%	1.40%	1.43%	1.52%	1.47%
((epi)galocat)-(epi)cat) 1	1.86%	2.04%	1.97%	1.88%	1.78%	1.76%	1.74%	1.72%
((epi)galocat)-(epi)cat) 2	1.73%	2.12%	2.02%	1.93%	1.83%	1.78%	1.79%	1.79%
(epi)cat-glucosa 1	2.50%	2.04%	1.77%	1.83%	1.77%	1.72%	1.66%	1.63%
(epi)cat-glucosa 2	1.49%	1.59%	1.49%	1.74%	1.70%	1.68%	1.69%	1.66%
((epi)cat)$_2$A 1	1.53%	1.64%	1.47%	1.57%	1.52%	1.45%	1.51%	1.52%
((epi)cat)$_2$A 2	1.09%	0.94%	1.17%	1.31%	1.24%	1.18%	1.18%	1.25%
((epi)cat-(epi)galocat)A 1	2.09%	1.91%	1.71%	1.73%	1.82%	1.83%	1.83%	1.79%
((epi)cat-(epi)galocat)A 2	1.65%	1.32%	1.54%	1.52%	1.43%	1.55%	1.52%	1.47%
(epi)cat-((epi)cat-(epi)galocat) A 1	0.20%	1.11%	1.40%	1.48%	1.47%	1.46%	1.51%	1.53%
(epi)cat-((epi)cat-(epi)galocat) A 2	0.42%	1.05%	1.17%	1.16%	1.18%	1.20%	1.33%	1.33%
p-vinil(epi)cat 1	0.32%	0.77%	1.02%	0.97%	1.09%	1.13%	1.13%	1.23%
p-vinil(epi)cat 2	1.51%	1.00%	1.12%	1.08%	1.04%	1.02%	1.16%	1.18%
p-vinil(epi)cat 3	0.91%	0.62%	0.61%	0.69%	0.80%	0.87%	0.85%	0.91%
(epi)Cat-furfural-(epi)cat 1	0.10%	1.05%	1.24%	1.34%	1.29%	1.30%	1.35%	1.33%
(epi)Cat-furfural-(epi)cat 2	0.06%	1.02%	1.19%	1.18%	1.16%	1.14%	1.22%	1.25%

Table 43. Importance in % of each original feature in the first n principal PCA components. Part I.

	Importance (%) up to PC9:	Importance (%) up to PC10:	Importance (%) up to PC11:	Importance (%) up to PC12:	Importance (%) up to PC13:	Importance (%) up to PC14:	Importance (%) up to PC15:	Importance (%) up to PC16:
pH	1.35%	1.34%	1.33%	1.32%	1.34%	1.45%	1.46%	1.44%
Dp-3-glc	1.73%	1.73%	1.70%	1.71%	1.70%	1.69%	1.68%	1.67%
Cy-3-glc	1.60%	1.63%	1.61%	1.65%	1.64%	1.63%	1.63%	1.62%
Pt-3-glc	1.77%	1.76%	1.74%	1.73%	1.71%	1.70%	1.69%	1.68%
Pn-3-glc	1.79%	1.79%	1.76%	1.75%	1.73%	1.71%	1.70%	1.70%
Mv-3-glc	1.83%	1.81%	1.78%	1.76%	1.73%	1.73%	1.71%	1.70%
Dp-3-(6-Ac)-glc	1.27%	1.31%	1.30%	1.29%	1.32%	1.32%	1.31%	1.31%
Pt-3-(6-Ac)-glc	1.68%	1.65%	1.66%	1.63%	1.64%	1.63%	1.61%	1.61%
Pn-3-(6-Ac)-glc	1.72%	1.68%	1.65%	1.68%	1.69%	1.70%	1.71%	1.71%
Mv-3-(6-Ac)-glc	1.75%	1.72%	1.70%	1.70%	1.69%	1.68%	1.67%	1.66%
Mv-3-(6-caff)-glc	1.65%	1.66%	1.64%	1.63%	1.61%	1.61%	1.63%	1.62%
Pt-3-(6-p-coum)-glc	1.75%	1.70%	1.68%	1.67%	1.65%	1.64%	1.63%	1.62%
Pn-3-(6-p-coum)-glc	1.91%	1.89%	1.87%	1.86%	1.83%	1.82%	1.80%	1.79%
Mv-3-(6-p-coum)-glc	1.79%	1.74%	1.72%	1.71%	1.70%	1.69%	1.67%	1.66%
Mv-3-glc-acetaldehído	1.53%	1.57%	1.61%	1.61%	1.63%	1.62%	1.61%	1.61%
Mv-3-glc-vinilmetilo	1.26%	1.25%	1.30%	1.30%	1.31%	1.33%	1.33%	1.34%
Mv-3-glc-pirúvico	1.29%	1.36%	1.34%	1.33%	1.33%	1.36%	1.36%	1.35%
Mv-3-glc-4-vinilfenol	1.44%	1.49%	1.50%	1.50%	1.55%	1.57%	1.58%	1.59%
Mv-3-glc-4-vinilcatecol	1.34%	1.36%	1.39%	1.38%	1.42%	1.41%	1.40%	1.39%
Mv-3-(6-p-coum)-glc cis	1.81%	1.79%	1.84%	1.83%	1.81%	1.82%	1.83%	1.83%
Mv-3-(6-p-coum)-glc trans	1.81%	1.77%	1.83%	1.81%	1.80%	1.79%	1.80%	1.79%
Mv-3-glc-4-vinilguaiacol	1.36%	1.36%	1.34%	1.32%	1.31%	1.30%	1.32%	1.32%
Mv-3-(6-p-coum)-glc-acetaldehído	1.42%	1.50%	1.48%	1.47%	1.48%	1.47%	1.47%	1.49%
Mv-3-(6-p-coum)-glc-pirúvico	1.39%	1.41%	1.42%	1.40%	1.43%	1.44%	1.43%	1.44%
Mv-3-(6-p-coum)-glc-4-vinilfenol	1.32%	1.34%	1.33%	1.33%	1.32%	1.34%	1.35%	1.35%
Catequin-Mv-3-glc	1.71%	1.68%	1.68%	1.69%	1.71%	1.71%	1.73%	1.72%
Epicatequin-Mv-3-glc	1.78%	1.75%	1.74%	1.74%	1.75%	1.75%	1.76%	1.75%
Mv-3-(6-p-coum)-glc-4-vinilguaiacol	1.62%	1.60%	1.64%	1.66%	1.66%	1.64%	1.65%	1.65%
(Epi)Galocatequin-Mv-3-glc	1.90%	1.90%	1.87%	1.87%	1.86%	1.84%	1.84%	1.83%
Catequin-Mv-3-(6-p-coum)-glc	1.76%	1.78%	1.76%	1.77%	1.76%	1.76%	1.77%	1.76%
(Epi)Galocatequin-Mv-3-(6-p-coum)-glc	1.74%	1.78%	1.77%	1.78%	1.78%	1.78%	1.77%	1.78%
Mv-3-(6-p-coum)-glc-8-etil-(epi)catequina	1.70%	1.67%	1.72%	1.74%	1.76%	1.76%	1.77%	1.76%
Catequina	1.81%	1.78%	1.76%	1.74%	1.74%	1.73%	1.75%	1.75%
Epicatequina	1.81%	1.81%	1.82%	1.82%	1.83%	1.81%	1.82%	1.82%
PCB1	1.80%	1.78%	1.76%	1.73%	1.71%	1.71%	1.70%	1.73%
PCB2	1.84%	1.82%	1.81%	1.81%	1.83%	1.81%	1.81%	1.81%
((epi)cat)$_2$ 1	1.88%	1.86%	1.85%	1.83%	1.81%	1.80%	1.79%	1.78%
PCC1	1.87%	1.84%	1.82%	1.80%	1.79%	1.78%	1.76%	1.77%
((epi)cat)$_3$ 1	1.34%	1.33%	1.34%	1.36%	1.35%	1.36%	1.37%	1.36%
((epi)cat)$_3$ 2	1.86%	1.83%	1.81%	1.78%	1.78%	1.77%	1.76%	1.76%
Galocatequina	1.82%	1.81%	1.79%	1.76%	1.76%	1.74%	1.76%	1.74%
Epigalocatequina	1.76%	1.73%	1.72%	1.71%	1.70%	1.72%	1.72%	1.72%
((epi)galocat)$_2$ 1	1.83%	1.84%	1.82%	1.81%	1.78%	1.78%	1.77%	1.76%
((epi)galocat)$_2$ 2	1.66%	1.68%	1.67%	1.66%	1.64%	1.64%	1.64%	1.65%
((epi)galocat)$_2$ 3	1.73%	1.69%	1.66%	1.64%	1.64%	1.64%	1.62%	1.62%
((epi)cat-(epi)galocat) 1	1.81%	1.83%	1.81%	1.81%	1.81%	1.79%	1.77%	1.77%
((epi)cat-(epi)galocat) 2	1.45%	1.43%	1.41%	1.40%	1.47%	1.48%	1.47%	1.51%
((epi)galocat)-(epi)cat) 1	1.77%	1.76%	1.77%	1.74%	1.73%	1.71%	1.72%	1.72%
((epi)galocat)-(epi)cat) 2	1.77%	1.78%	1.75%	1.74%	1.72%	1.71%	1.70%	1.71%
(epi)cat-glucosa 1	1.66%	1.68%	1.66%	1.63%	1.62%	1.60%	1.61%	1.61%
(epi)cat-glucosa 2	1.66%	1.67%	1.67%	1.71%	1.70%	1.68%	1.71%	1.70%
((epi)cat)$_2$A 1	1.54%	1.54%	1.57%	1.59%	1.57%	1.56%	1.59%	1.59%
((epi)cat)$_2$A 2	1.24%	1.36%	1.41%	1.43%	1.42%	1.46%	1.45%	1.45%
((epi)cat-(epi)galocat)A 1	1.80%	1.77%	1.75%	1.73%	1.75%	1.73%	1.75%	1.75%
((epi)cat-(epi)galocat)A 2	1.45%	1.43%	1.50%	1.51%	1.51%	1.52%	1.51%	1.52%
(epi)cat-((epi)cat-(epi)galocat) A 1	1.55%	1.51%	1.48%	1.48%	1.48%	1.46%	1.45%	1.44%
(epi)cat-((epi)cat-(epi)galocat) A 2	1.35%	1.35%	1.34%	1.43%	1.42%	1.43%	1.44%	1.44%
p-vinil(epi)cat 1	1.25%	1.26%	1.32%	1.38%	1.40%	1.42%	1.44%	1.43%
p-vinil(epi)cat 2	1.27%	1.33%	1.38%	1.39%	1.41%	1.40%	1.39%	1.45%
p-vinil(epi)cat 3	1.01%	1.04%	1.21%	1.20%	1.23%	1.23%	1.24%	1.27%
(epi)Cat-furfural-(epi)cat 1	1.34%	1.36%	1.36%	1.41%	1.42%	1.45%	1.45%	1.45%
(epi)Cat-furfural-(epi)cat 2	1.31%	1.31%	1.29%	1.35%	1.36%	1.37%	1.37%	1.38%

Table 44. Importance in % of each original feature in the first n principal PCA components. Part II.

7.2 Class independent Fischer's discriminant and LDA

As was done with PCA, we apply Fischer's discriminant in search of new features that may be useful in subsequent classifications. To do so, we proceed to apply Fischer's model by following exactly the steps indicated in section 2.3 in the Matlab R2019b environment on the datasets of each taster without normalization.

Reducing the two datasets separately yields the eigenvalues of the Table 45.

No. Eigenvalue	Eigenvalue value (1st dataset)	Cumulative variance (1st dataset)	Eigenvalue value (2nd dataset)	Cumulative variance (2nd dataset)
1	4.821	63.82%	9.811	54.73%
2	1.862	88.47%	5.509	86.12%
3	0.871	100.00%	2.465	100.00%

Table 45. Eigenvalues associated with Fischer's principal components and their contribution to the total variance.

Representing the first dataset visually with 3 components shows a clear improvement in the new dimensions that allow a better classification. In the figure Figure 50 you can see two different views of the three-dimensional representation where the centroids for each category have been included in bold. The samples of the *Bad* category are easily separable from the rest, and most of the samples of the other three groups also appear to be separable, although in the central area they are mixed together.

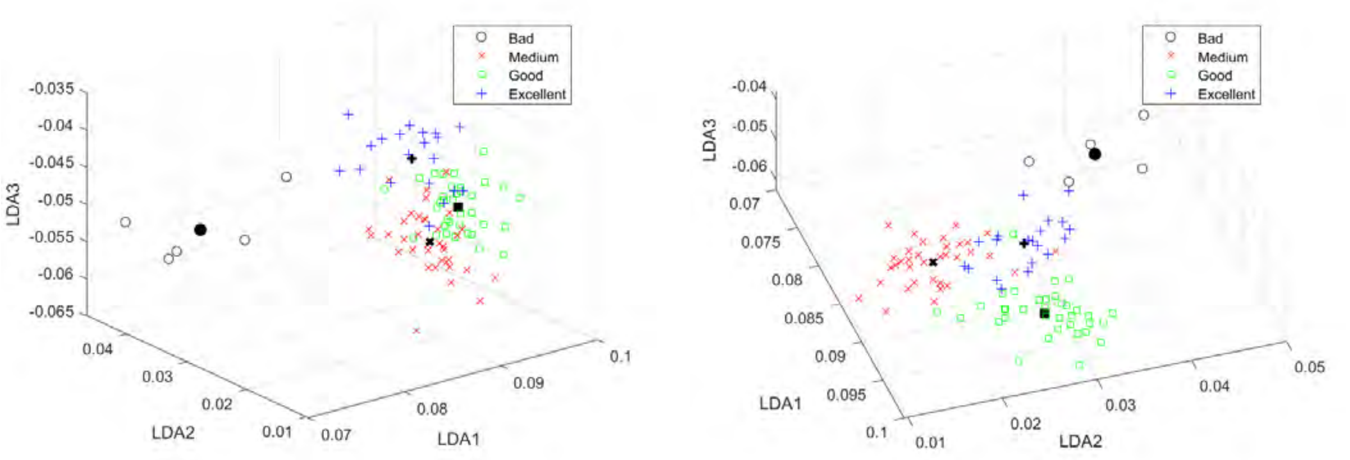


Figure 50. Representation of the first taster's scores on the first three Fischer's principal components.

In any case, the new features defined by the three new eigenvectors can be very useful in future classifiers, so they will be included hereafter as LDA features.

If we analyze the participation of the original features in the new vectors to draw conclusions about their importance in the same way as we did with the PCA components, we will see that this time there are features with an outstanding contribution. Table 46 shows the absolute values of the weights that each original feature has on each principal component, according to the three different eigenvectors.

	Eigenvalues (importance of each component)	4.82100421	1.86234045	0.8710116
		LDA1	LDA2	LDA3
Weight of original feature No 1	**pH**	0.022	0.011	0.012
Weight of original feature No 2	**Dp-3-glc**	0.001	0.001	0.002
Weight of original feature No 3	**Cy-3-glc**	0.001	0.012	0.001
Weight of original feature No 4	**Pt-3-glc**	0.000	0.001	0.001
Weight of original feature No 5	**Pn-3-glc**	0.002	0.007	0.004
Weight of original feature No 6	**Mv-3-glc**	0.000	0.001	0.000
Weight of original feature No 7	**Dp-3-(6-Ac)-glc**	0.007	0.007	0.004
Weight of original feature No 8	**Pt-3-(6-Ac)-glc**	0.002	0.003	0.015
Weight of original feature No 9	**Pn-3-(6-Ac)-glc**	0.015	0.010	0.004
Weight of original feature No 10	**Mv-3-(6-Ac)-glc**	0.001	0.002	0.002
Weight of original feature No 11	**Mv-3-(6-caff)-glc**	0.005	0.013	0.000
Weight of original feature No 12	**Pt-3-(6-p-coum)-glc**	0.014	0.018	0.012
Weight of original feature No 13	**Pn-3-(6-p-coum)-glc**	0.007	0.013	0.013
Weight of original feature No 14	**Mv-3-(6-p-coum)-glc**	0.006	0.003	0.005
Weight of original feature No 15	**Mv-3-glc-acetaldehído**	0.020	0.019	0.012
Weight of original feature No 16	**Mv-3-glc-vinilmetilo**	0.014	0.261	0.052
Weight of original feature No 17	**Mv-3-glc-pirúvico**	0.020	0.013	0.015
Weight of original feature No 18	**Mv-3-glc-4-vinilfenol**	0.012	0.020	0.004
Weight of original feature No 19	**Mv-3-glc-4-vinilcatecol**	0.003	0.079	0.012
Weight of original feature No 20	**Mv-3-(6-p-coum)-glc cis**	0.117	0.378	0.040
Weight of original feature No 21	**Mv-3-(6-p-coum)-glc trans**	0.003	0.008	0.001
Weight of original feature No 22	**Mv-3-glc-4-vinilguaiacol**	0.028	0.170	0.047
Weight of original feature No 23	**Mv-3-(6-p-coum)-glc-acetaldehído**	0.091	0.117	0.002
Weight of original feature No 24	**Mv-3-(6-p-coum)-glc-pirúvico**	0.031	0.007	0.041
Weight of original feature No 25	**Mv-3-(6-p-coum)-glc-4-vinilfenol**	0.039	0.064	0.085
Weight of original feature No 26	**Catequin-Mv-3-glc**	0.025	0.099	0.043
Weight of original feature No 27	**Epicatequin-Mv-3-glc**	0.077	0.170	0.260
Weight of original feature No 28	**Mv-3-(6-p-coum)-glc-4-vinilguaiacol**	0.318	0.123	0.539
Weight of original feature No 29	**(Epi)Galocatequin-Mv-3-glc**	0.224	0.303	0.102
Weight of original feature No 30	**Catequin-Mv-3-(6-p-coum)-glc**	0.846	0.318	0.128
Weight of original feature No 31	**(Epi)Galocatequin-Mv-3-(6-p-coum)-glc**	0.250	0.682	0.056
Weight of original feature No 32	**Mv-3-(6-p-coum)-glc-8-etil-(epi)catequina 1**	0.177	0.090	0.765
Weight of original feature No 33	**Catequina**	0.004	0.006	0.001
Weight of original feature No 34	**Epicatequina**	0.003	0.012	0.001
Weight of original feature No 35	**PCB1**	0.000	0.000	0.000
Weight of original feature No 36	**PCB2**	0.001	0.000	0.000
Weight of original feature No 37	**((epi)cat)$_2$ 1**	0.004	0.004	0.003
Weight of original feature No 38	**PCC1**	0.044	0.068	0.005
Weight of original feature No 39	**((epi)cat)$_3$ 1**	0.046	0.004	0.015
Weight of original feature No 40	**((epi)cat)$_3$ 2**	0.000	0.010	0.011
Weight of original feature No 41	**Galocatequina**	0.004	0.006	0.003
Weight of original feature No 42	**Epigalocatequina**	0.000	0.034	0.009
Weight of original feature No 43	**((epi)galocat)$_2$ 1**	0.002	0.014	0.010
Weight of original feature No 44	**((epi)galocat)$_2$ 2**	0.004	0.039	0.015
Weight of original feature No 45	**((epi)galocat)$_2$ 3**	0.021	0.004	0.013
Weight of original feature No 46	**((epi)cat-(epi)galocat) 1**	0.001	0.004	0.008
Weight of original feature No 47	**((epi)cat-(epi)galocat) 2**	0.001	0.000	0.000
Weight of original feature No 48	**((epi)galocat)-(epi)cat) 1**	0.000	0.003	0.004
Weight of original feature No 49	**((epi)galocat)-(epi)cat) 2**	0.011	0.005	0.006
Weight of original feature No 50	**(epi)cat-glucosa 1**	0.001	0.001	0.000
Weight of original feature No 51	**(epi)cat-glucosa 2**	0.005	0.002	0.006
Weight of original feature No 52	**((epi)cat)$_2$A 1**	0.021	0.031	0.005
Weight of original feature No 53	**((epi)cat)$_2$A 2**	0.006	0.024	0.010
Weight of original feature No 54	**((epi)cat-(epi)galocat)A 1**	0.005	0.046	0.048
Weight of original feature No 55	**((epi)cat-(epi)galocat)A 2**	0.010	0.004	0.013
Weight of original feature No 56	**(epi)cat-((epi)cat-(epi)galocat) A 1**	0.002	0.002	0.000
Weight of original feature No 57	**(epi)cat-((epi)cat-(epi)galocat) A 2**	0.016	0.013	0.014
Weight of original feature No 58	**p-vinil(epi)cat 1**	0.005	0.042	0.064
Weight of original feature No 59	**p-vinil(epi)cat 2**	0.001	0.001	0.001
Weight of original feature No 60	**p-vinil(epi)cat 3**	0.003	0.001	0.004
Weight of original feature No 61	**(epi)Cat-furfural-(epi)cat 1**	0.001	0.004	0.002
Weight of original feature No 62	**(epi)Cat-furfural-(epi)cat 2**	0.001	0.003	0.001

Table 46. Contribution of each original feature in each Fischer’s principal component of the first dataset.

Feature	Importance percentage
Catequin-Mv-3-(6-p-coum)-glc	22.66%
(Epi)Galocatequin-Mv-3-(6-p-coum)-glc	11.97%
Mv-3-(6-p-coum)-glc-4-vinilguaiacol	10.57%
(Epi)Galocatequin-Mv-3-glc	8.21%
Mv-3-(6-p-coum)-glc-8-etil-(epi)catequina 1	8.01%
Mv-3-(6-p-coum)-glc cis	6.17%
Epicatequin-Mv-3-glc	4.33%
Mv-3-(6-p-coum)-glc-acetaldehído	3.12%
Mv-3-glc-vinilmetilo	2.84%
Mv-3-glc-4-vinilguaiacol	2.34%
Mv-3-(6-p-coum)-glc-4-vinilfenol	1.81%
Catequin-Mv-3-glc	1.63%
PCC1	1.63%
((epi)cat)$_3$ 1	1.14%
Mv-3-(6-p-coum)-glc-pirúvico	0.93%
Mv-3-glc-4-vinilcatecol	0.83%
((epi)cat)$_2$A 1	0.78%
p-vinil(epi)cat 1	0.75%
((epi)cat-(epi)galocat)A 1	0.71%
Mv-3-glc-acetaldehído	0.67%
pH	0.65%
Mv-3-glc-pirúvico	0.63%
((epi)galocat)$_2$ 3	0.56%
(epi)cat-((epi)cat-(epi)galocat) A 2	0.53%
Pt-3-(6-p-coum)-glc	0.52%
((epi)galocat)$_2$ 2	0.50%
Mv-3-glc-4-vinilfenol	0.47%
Pn-3-(6-Ac)-glc	0.45%
((epi)cat)$_2$A 2	0.40%
Epigalocatequina	0.35%
Pn-3-(6-p-coum)-glc	0.32%
((epi)galocat)-(epi)cat) 2	0.31%
((epi)cat-(epi)galocat)A 2	0.31%
Dp-3-(6-Ac)-glc	0.24%
Mv-3-(6-caff)-glc	0.22%
((epi)galocat)$_2$ 1	0.20%
Epicatequina	0.18%
Mv-3-(6-p-coum)-glc	0.18%
Galocatequina	0.16%
(epi)cat-glucosa 2	0.15%
((epi)cat)$_2$ 1	0.15%
Pt-3-(6-Ac)-glc	0.14%
Catequina	0.14%
Cy-3-glc	0.14%
((epi)cat)$_3$ 2	0.14%
Mv-3-(6-p-coum)-glc trans	0.13%
Pn-3-glc	0.12%
p-vinil(epi)cat 3	0.09%
((epi)cat-(epi)galocat) 1	0.09%
(epi)Cat-furfural-(epi)cat 1	0.07%
(epi)cat-((epi)cat-(epi)galocat) A 1	0.07%
Mv-3-(6-Ac)-glc	0.06%
((epi)galocat)-(epi)cat) 1	0.05%
(epi)Cat-furfural-(epi)cat 2	0.04%
Dp-3-glc	0.03%
p-vinil(epi)cat 2	0.03%
PCB2	0.03%
(epi)cat-glucosa 1	0.02%
Pt-3-glc	0.02%
((epi)cat-(epi)galocat) 2	0.02%
Mv-3-glc	0.02%
PCB1	0.00%

Table 47. Order of importance of the features in the first 3 LDA principal components for the first dataset.

If, again, we weight each eigenvector with the value of its eigenvalue for the first 3 components combining the later results we obtain the ordered score of the Table 47. It is more than remarkable that the features contributing by far the most to the components are Catequin-Mv-3-(6-p-coum)-glc, (Epi)Galocatequin-Mv-3-(6-p-coum)-glc, Mv-3-(6-p-coum)-glc-4-vinylguaiacol, (Epi)Galocatequin-Mv-3-glc, Mv-3-(6-p-coum)-glc-8-ethyl-(epi)catechin 1, Mv-3-(6-p-coum)-glc cis, Epicatequin-Mv-3-glc, Mv-3-(6-p-coum)-glc-acetaldehyde, Mv-3-glc-vinylmethyl and Mv-3-glc-4-vinylguaiacol. The first 10 compounds, all of them, are anthocyanin derivatives. If we wish to accumulate 90% of the scores calculated in this way other components should be included in the list as well, such as Mv-3-(6-p-coum)-glc-4-vinylphenol, Catequin-Mv-3-glc, PCC1, ((epi)cat)$_3$ 1, Mv-3-(6-p-coum)-glc-pyruvic, Mv-3-glc-4-vinylcatechol, ((epi)cat)$_2$A 1, p-vinyl(epi)cat 1 and ((epi)cat-(epi)galocat)A 1. Of the 19 compounds required 14 are anthocyanin derivatives (only 4 anthocyanin derivatives have been left out) and 5 tannins. The anthocyanin derivatives seem to have a special discriminatory value in determining the scores of the first taster. The listed characteristics will be selected to experiment only with them later. They will be referred to as LDA Selection$_1$.

We will now apply the LDA classifier with the standard loss function and calculate the various discriminant functions. To this end we will use a version of equation (50) without the terms common to all classes

$$\log P(C_j|\boldsymbol{y}) = -\frac{1}{2}(\boldsymbol{y}-\boldsymbol{m_j})S_{intra}'^{\ -1}(\boldsymbol{y}-\boldsymbol{m_j})^T + \log P(C_j) \qquad (108)$$

and classify the sample in the class with the highest value. In Figure 51 you can see the data and the decision regions for the LDA classifier applied on only the first two principal components.

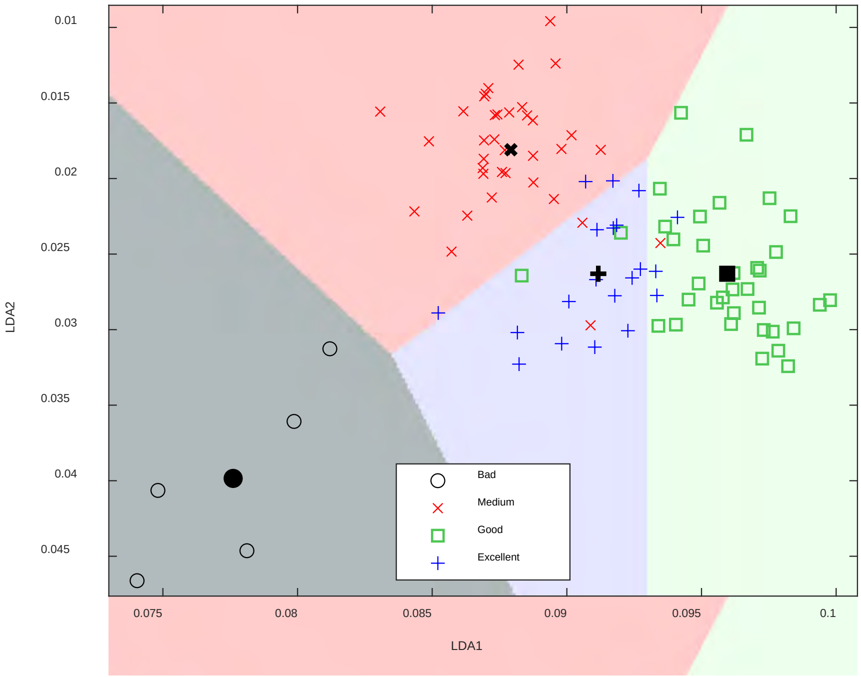
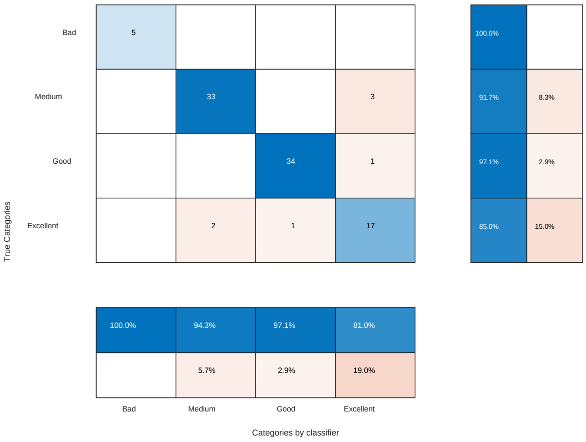


Figure 51. Unvalidated LDA classifier decision regions on first two principal components for the first taster data.

The classification using the first three principal components offers a slight improvement in the elements of the central zone, but they are very similar. The confusion matrix of this classifier can be seen in Figure 52.



True Categories

	Bad	Medium	Good	Excellent		
Bad	5				100.0%	
Medium		33		3	91.7%	8.3%
Good			34	1	97.1%	2.9%
Excellent		2	1	17	85.0%	15.0%
	100.0%	94.3%	97.1%	81.0%		
		5.7%	2.9%	19.0%		

Categories by classifier

Figure 52. Confusion matrix of the 3 principal components unvalidated LDA classifier for the first taster's data.

The classifier global accuracy is 92.71%. As can be seen in Figure 52 all samples in the true *Bad* category are classified correctly. The classifier has some problems with the samples that the taster rated as *Excellent*. Of the 20 samples with the highest score, 1 was classified as *Good* and two of them as *Medium*, so 15% of the samples that should be recommended to the user in the first place would be lost. The biggest problem is that of the samples that should be classified as *Medium*, there are 3 of them that the classifier would recommend as excellent. In addition, one of the *Good* category samples would also be advised as *Excellent*. This can cause significant disappointment to the end user, as 19% of the samples that the classifier issues as positive recommendations would not be such.

Using the loss matrix proposed in section 6.4 the results of the 3 principal component classifier do not vary, although there is a variation in the decision regions that tend to classify fewer elements as *Excellent*. The difference between the regions can be seen in Figure 53, which is obtained on the linear classifier on two principal components.

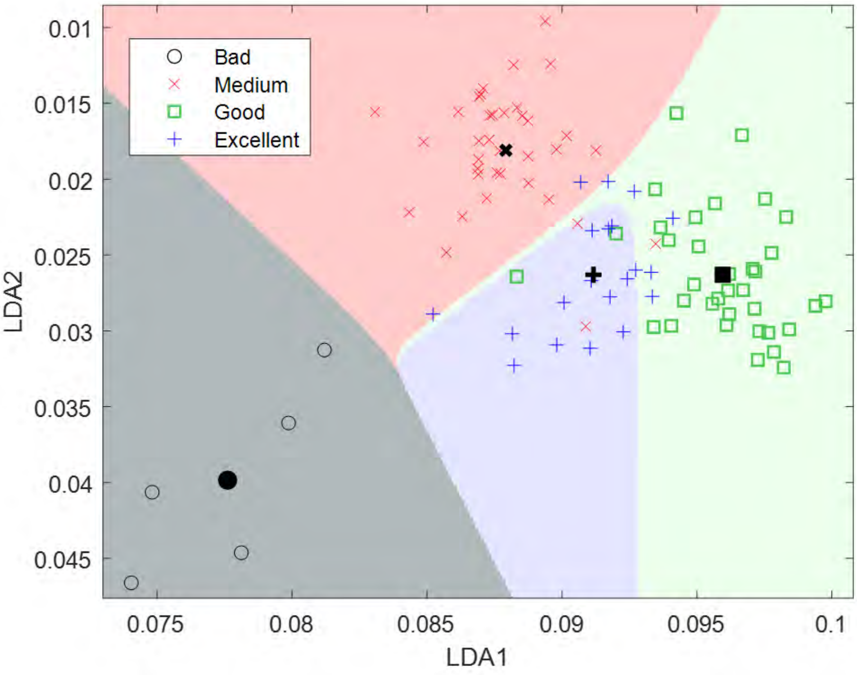


Figure 53. Unvalidated LDA classifier decision regions on first two principal components for the first taster data with non-standard loss matrix.

Using the second dataset the algorithm failed in its original version due to the singularity of S_{intra}. After trying to orthogonalize the alternative matrix $S_{inter}^{-1}S_{intra}$ the problem persists, as the matrix S_{inter} is also singular. The problem was solved by introducing a regularization of the matrix $S_{intra\,reg} = S_{intra} + \eta I$ as indicated in section 2.3 with a regularization factor 10^{-5} times the average value of the elements of the main diagonal of S_{intra}, i.e. $\eta = 3{,}142 \cdot 10^{-4}$.

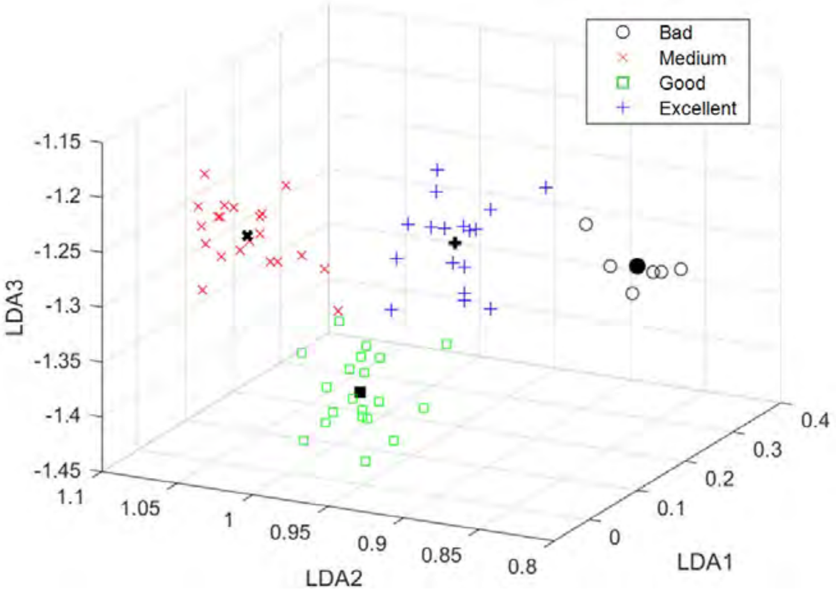


Figure 54. Representation of the second taster's scores on the first three Fischer's principal components.

Plotting the data in the three (Figure 54) and even in the first two principal components (Figure 55) a clear separation between the different classes is observed, which is reflected in a correct classification through LDA of 98.43% of the samples (only one case in the *Medium* category wrongly classified as *Good*). Furthermore, the introduction of the loss matrix in the classifier, although tending to slightly modify the decision regions, gives similar results (an overall accuracy of 96.86% with only one more case of the *Excellent* category misclassified as *Good*).

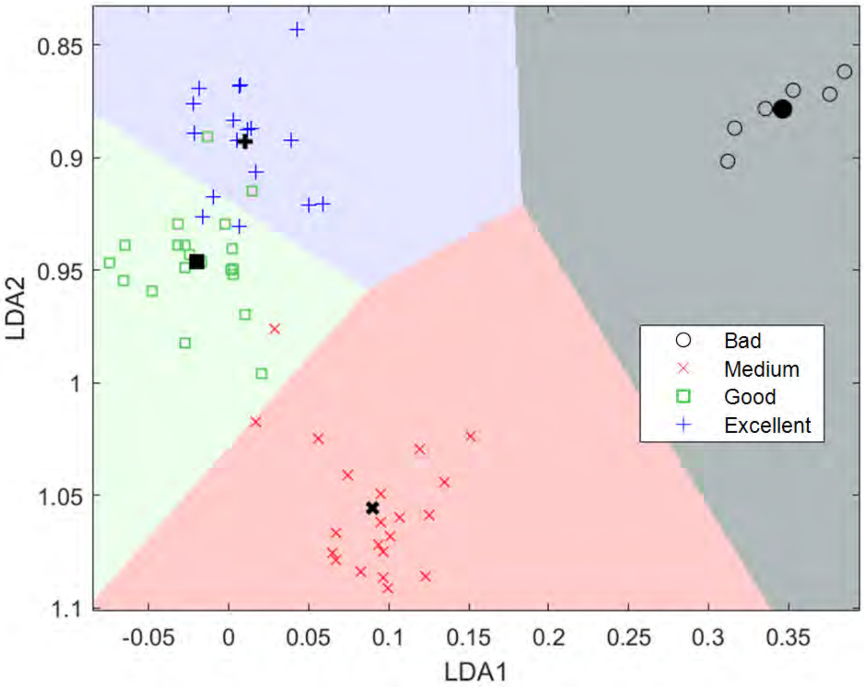


Figure 55. Unvalidated LDA classifier decision regions on first two principal components for the second taster data.

	Eigenvalues (importance of each component)	9.811	5.509	2.465
		LDA1	LDA2	LDA3
Weight of original feature No 1	**pH**	0.017	0.276	0.334
Weight of original feature No 2	**Dp-3-glc**	0.005	0.007	0.002
Weight of original feature No 3	**Cy-3-glc**	0.153	0.057	0.057
Weight of original feature No 4	**Pt-3-glc**	0.010	0.007	0.004
Weight of original feature No 5	**Pn-3-glc**	0.049	0.004	0.017
Weight of original feature No 6	**Mv-3-glc**	0.000	0.001	0.001
Weight of original feature No 7	**Dp-3-(6-Ac)-glc**	0.152	0.070	0.020
Weight of original feature No 8	**Pt-3-(6-Ac)-glc**	0.075	0.015	0.054
Weight of original feature No 9	**Pn-3-(6-Ac)-glc**	0.117	0.135	0.126
Weight of original feature No 10	**Mv-3-(6-Ac)-glc**	0.012	0.010	0.050
Weight of original feature No 11	**Mv-3-(6-caff)-glc**	0.070	0.002	0.052
Weight of original feature No 12	**Pt-3-(6-p-coum)-glc**	0.065	0.068	0.007
Weight of original feature No 13	**Pn-3-(6-p-coum)-glc**	0.036	0.171	0.137
Weight of original feature No 14	**Mv-3-(6-p-coum)-glc**	0.009	0.032	0.016
Weight of original feature No 15	**Mv-3-glc-acetaldehído**	0.276	0.150	0.440
Weight of original feature No 16	**Mv-3-glc-vinilmetilo**	0.108	0.060	0.196
Weight of original feature No 17	**Mv-3-glc-pirúvico**	0.496	0.186	0.025
Weight of original feature No 18	**Mv-3-glc-4-vinilfenol**	0.208	0.036	0.036
Weight of original feature No 19	**Mv-3-glc-4-vinilcatecol**	0.077	0.319	0.409
Weight of original feature No 20	**Mv-3-(6-p-coum)-glc cis**	0.062	0.046	0.190
Weight of original feature No 21	**Mv-3-(6-p-coum)-glc trans**	0.030	0.019	0.007
Weight of original feature No 22	**Mv-3-glc-4-vinilguaiacol**	0.025	0.024	0.053
Weight of original feature No 23	**Mv-3-(6-p-coum)-glc-acetaldehído**	0.061	0.061	0.003
Weight of original feature No 24	**Mv-3-(6-p-coum)-glc-pirúvico**	0.203	0.044	0.190
Weight of original feature No 25	**Mv-3-(6-p-coum)-glc-4-vinilfenol**	0.025	0.067	0.218
Weight of original feature No 26	**Catequin-Mv-3-glc**	0.296	0.141	0.105
Weight of original feature No 27	**Epicatequin-Mv-3-glc**	0.078	0.027	0.057
Weight of original feature No 28	**Mv-3-(6-p-coum)-glc-4-vinilguaiacol**	0.009	0.029	0.002
Weight of original feature No 29	**(Epi)Galocatequin-Mv-3-glc**	0.027	0.002	0.016
Weight of original feature No 30	**Catequin-Mv-3-(6-p-coum)-glc**	0.003	0.009	0.033
Weight of original feature No 31	**(Epi)Galocatequin-Mv-3-(6-p-coum)-glc**	0.010	0.004	0.003
Weight of original feature No 32	**Mv-3-(6-p-coum)-glc-8-etil-(epi)catequina 1**	0.008	0.012	0.005
Weight of original feature No 33	**Catequina**	0.007	0.001	0.016
Weight of original feature No 34	**Epicatequina**	0.084	0.051	0.043
Weight of original feature No 35	**PCB1**	0.000	0.001	0.001
Weight of original feature No 36	**PCB2**	0.013	0.009	0.008
Weight of original feature No 37	**((epi)cat)$_2$ 1**	0.030	0.048	0.053
Weight of original feature No 38	**PCC1**	0.376	0.068	0.038
Weight of original feature No 39	**((epi)cat)$_3$ 1**	0.065	0.276	0.011
Weight of original feature No 40	**((epi)cat)$_3$ 2**	0.056	0.093	0.058
Weight of original feature No 41	**Galocatequina**	0.067	0.185	0.107
Weight of original feature No 42	**Epigalocatequina**	0.101	0.279	0.029
Weight of original feature No 43	**((epi)galocat)$_2$ 1**	0.036	0.032	0.069
Weight of original feature No 44	**((epi)galocat)$_2$ 2**	0.169	0.137	0.166
Weight of original feature No 45	**((epi)galocat)$_2$ 3**	0.208	0.115	0.020
Weight of original feature No 46	**((epi)cat-(epi)galocat) 1**	0.025	0.059	0.016
Weight of original feature No 47	**((epi)cat-(epi)galocat) 2**	0.017	0.021	0.004
Weight of original feature No 48	**((epi)galocat)-(epi)cat) 1**	0.108	0.060	0.035
Weight of original feature No 49	**((epi)galocat)-(epi)cat) 2**	0.027	0.011	0.003
Weight of original feature No 50	**(epi)cat-glucosa 1**	0.052	0.015	0.049
Weight of original feature No 51	**(epi)cat-glucosa 2**	0.007	0.035	0.018
Weight of original feature No 52	**((epi)cat)$_2$A 1**	0.070	0.089	0.271
Weight of original feature No 53	**((epi)cat)$_2$A 2**	0.054	0.121	0.289
Weight of original feature No 54	**((epi)cat-(epi)galocat)A 1**	0.251	0.607	0.214
Weight of original feature No 55	**((epi)cat-(epi)galocat)A 2**	0.032	0.125	0.029
Weight of original feature No 56	**(epi)cat-((epi)cat-(epi)galocat) A 1**	0.027	0.029	0.081
Weight of original feature No 57	**(epi)cat-((epi)cat-(epi)galocat) A 2**	0.022	0.075	0.153
Weight of original feature No 58	**p-vinil(epi)cat 1**	0.239	0.034	0.083
Weight of original feature No 59	**p-vinil(epi)cat 2**	0.018	0.014	0.009
Weight of original feature No 60	**p-vinil(epi)cat 3**	0.017	0.016	0.033
Weight of original feature No 61	**(epi)Cat-furfural-(epi)cat 1**	0.007	0.007	0.026
Weight of original feature No 62	**(epi)Cat-furfural-(epi)cat 2**	0.012	0.006	0.023

Table 48. Contribution of each original feature in each Fischer's principal component of the second dataset.

Feature	Importance percentage
((epi)cat-(epi)galocat)A 1	7.32%
Mv-3-glc-pirúvico	6.87%
Mv-3-glc-acetaldehído	5.34%
PCC1	4.80%
Catequin-Mv-3-glc	4.55%
Mv-3-glc-4-vinilcatecol	4.07%
((epi)galocat)2 2	3.25%
p-vinil(epi)cat 1	3.16%
((epi)galocat)2 3	3.14%
Mv-3-(6-p-coum)-glc-pirúvico	3.12%
Epigalocatequina	3.00%
pH	2.90%
Mv-3-glc-4-vinilfenol	2.70%
Pn-3-(6-Ac)-glc	2.54%
((epi)cat)3 1	2.53%
Cy-3-glc	2.26%
Galocatequina	2.24%
Dp-3-(6-Ac)-glc	2.23%
((epi)cat)2A 2	2.21%
Mv-3-glc-vinilmetilo	2.17%
((epi)cat)2A 1	2.13%
Pn-3-(6-p-coum)-glc	1.89%
((epi)galocat)-(epi)cat) 1	1.71%
Mv-3-(6-p-coum)-glc cis	1.54%
Epicatequina	1.40%
((epi)cat)3 2	1.39%
Mv-3-(6-p-coum)-glc-4-vinilfenol	1.33%
((epi)cat-(epi)galocat)A 2	1.24%
Epicatequin-Mv-3-glc	1.22%
Pt-3-(6-p-coum)-glc	1.19%
(epi)cat-((epi)cat-(epi)galocat) A 2	1.17%
Pt-3-(6-Ac)-glc	1.10%
Mv-3-(6-p-coum)-glc-acetaldehído	1.09%
Mv-3-(6-caff)-glc	0.96%
(epi)cat-glucosa 1	0.83%
((epi)galocat)2 1	0.80%
((epi)cat)2 1	0.79%
(epi)cat-((epi)cat-(epi)galocat) A 1	0.71%
((epi)cat-(epi)galocat) 1	0.70%
Pn-3-glc	0.63%
Mv-3-glc-4-vinilguaiacol	0.59%
Mv-3-(6-p-coum)-glc trans	0.48%
p-vinil(epi)cat 3	0.39%
((epi)galocat)-(epi)cat) 2	0.38%
Mv-3-(6-p-coum)-glc	0.36%
(Epi)Galocatequin-Mv-3-glc	0.36%
(epi)cat-glucosa 2	0.35%
Mv-3-(6-Ac)-glc	0.34%
((epi)cat-(epi)galocat) 2	0.34%
p-vinil(epi)cat 2	0.32%
Mv-3-(6-p-coum)-glc-4-vinilguaiacol	0.30%
(epi)Cat-furfural-(epi)cat 2	0.24%
PCB2	0.23%
(epi)Cat-furfural-(epi)cat 1	0.20%
Catequin-Mv-3-(6-p-coum)-glc	0.19%
Mv-3-(6-p-coum)-glc-8-etil-(epi)catequina 1	0.18%
Pt-3-glc	0.17%
(Epi)Galocatequin-Mv-3-(6-p-coum)-glc	0.14%
Catequina	0.13%
Dp-3-glc	0.11%
Mv-3-glc	0.01%
PCB1	0.01%

Table 49. Order of importance of the features in the first 3 LDA principal components for the second dataset.

Following exactly the same procedure as for the data from the first taster, we analyzed the contribution of each original feature in the first three principal components. The results can be seen in Table 48 and Table 49. As it happened with the first set of data, there are features with a more outstanding contribution than others. In the case of the second taster, more original features are necessary to accumulate 90% of the calculated scores (the percentages of the feature that contribute the most are lower), and, although anthocyanin derivatives continue having an important weight compared to the rest of the families, some specific tannins and anthocyanins seem to be taken into account.

Of the 36 compounds required, 11 are anthocyanin derivatives (only 4 of them in the top 10) and 16 tannins (5 of them in the top 10). If we compare their situation with their ranking in the first dataset on the one hand, and with their situation in the first 16 PCA components on the other hand, we will see that few of them occupy the first places in more than one ranking and no general conclusion can be drawn that would be valid for all tasters.

Feature	LDA Ordination (1st dataset)	LDA Ordination (2nd dataset)	PCA Ordination
Catechin-Mv-3-(6-p-coum)-glc	1	55	12
(Epi)Galocatechin-Mv-3-(6-p-coum)-glc	2	58	7
Mv-3-(6-p-coum)-glc-4-vinylguaiacol	3	51	33
(Epi)Galocatechin-Mv-3-glc	4	46	2
Mv-3-(6-p-coum)-glc-8-ethyl-(epi)catechin 1	5	56	11
Mv-3-(6-p-coum)-glc cis	6	24	1
Epicatequin-Mv-3-glc	7	29	15
Mv-3-(6-p-coum)-glc-acetaldehyde	8	33	45
Mv-3-glc-vinyl methyl	9	20	59
Mv-3-glc-4-vinilguaiacol	10	41	60
((epi)cat-(epi)galocat)A 1	51	1	17
Mv-3-glc-pyruvic	22	2	57
Mv-3-glc-acetaldehyde	20	3	40
PCC1	13	4	9
Catequin-Mv-3-glc	12	5	22
Mv-3-glc-4-vinylcatechol	16	6	54
((epi)galocat)$_2$ 2	36	7	32
p-vinyl(epi)cat 1	18	8	53
((epi)galocat)$_2$ 3	23	9	34
Mv-3-(6-p-coum)-glc-pyruvic acid	15	10	50

Table 50. Comparison of feature importance in PCA and LDA.

Validating the LDA classifiers obtained in this section using LOO, the results (see Figure 56 and Figure 57) are slightly lower with overall accuracies for the classifiers of 90.62% for the first dataset and 95.31% for the second one. When using validation on the lossy function the results are similar. The confusion matrices of the first set (Figure 56) reflect classification values above 80% if only the *Bad* and *Excellent* categories are considered.

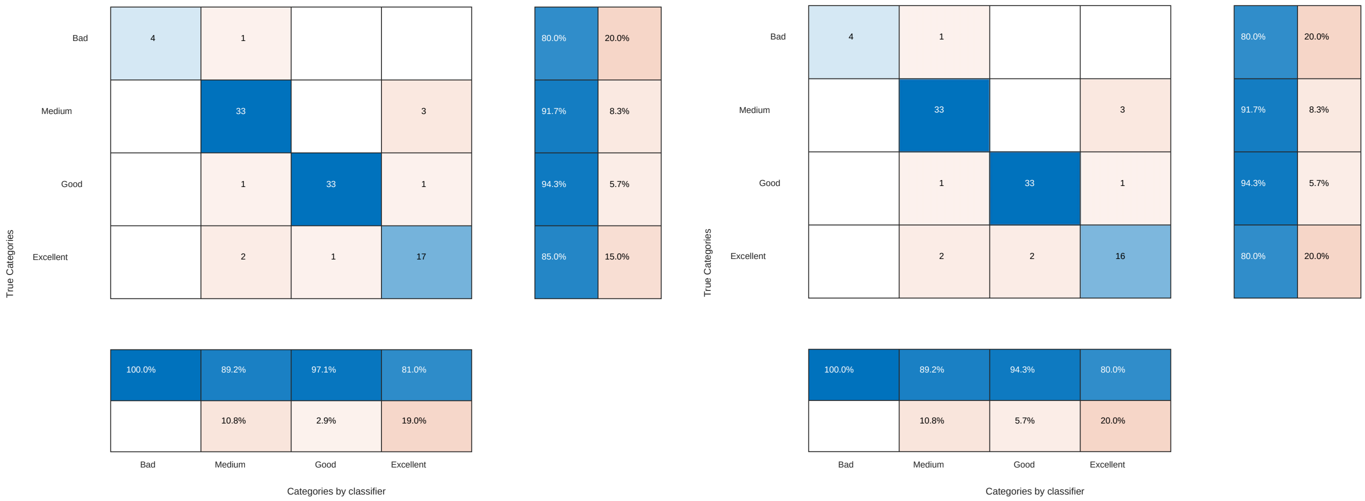


Figure 56. Confusion matrix of the 3 principal components LOO validated LDA classifier for the first taster's data with the standard loss matrix (left) and the proposed loss matrix (right).

The good results of these classifiers are due to the high separability of the classes. As already indicated, the anthocyanin derivatives seemed to have a special discriminatory value in determining the scores of both tasters. However, when we use only the LDA_1 features in an LDA classifier, even if it gets acceptable discriminations of the *Medium* and *Good* categories with accuracies of 60-75%, it classifies poorly those classes of most interest to us (*Bad* and *Excellent*), with accuracies dropping to 10%. This is due to the fact that the separation between the samples is not complete in the new spaces.

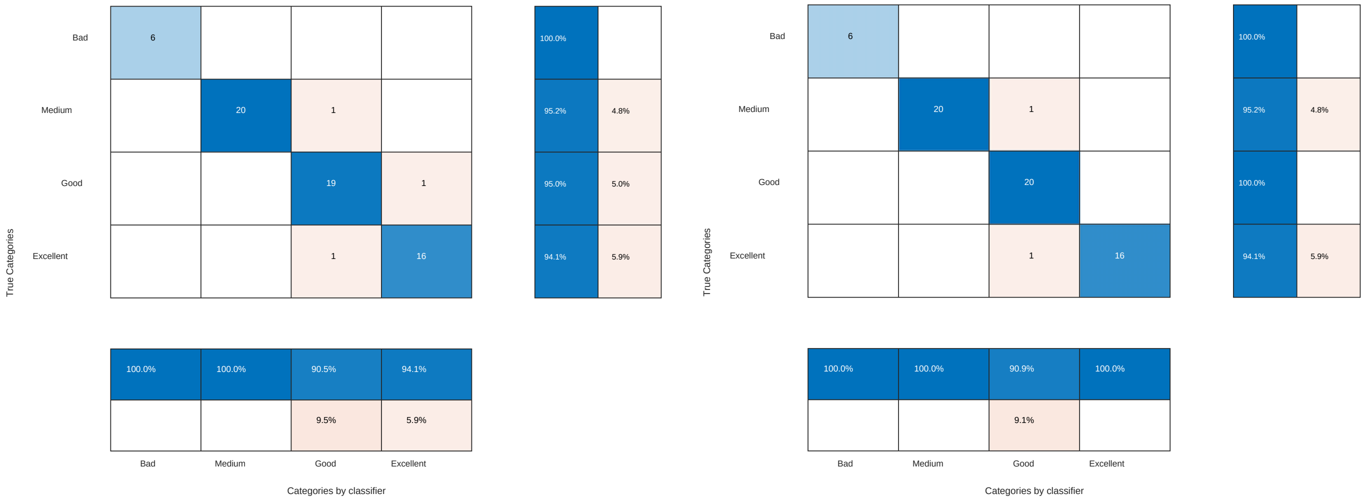


Figure 57. Confusion matrix of the 3 principal components LOO validated LDA classifier for the second taster's data with the standard loss matrix (left) and the proposed loss matrix (right).

Applying PCA to remove noise in the data prior to Fischer's method and the classifier is not advantageous. In the first dataset the categories are not separable in the new spaces. Although the *Bad* category is distinguishable from the rest by taking the 37 principal components in the first taster data (Figure 58), the rest of the categories are not separable.

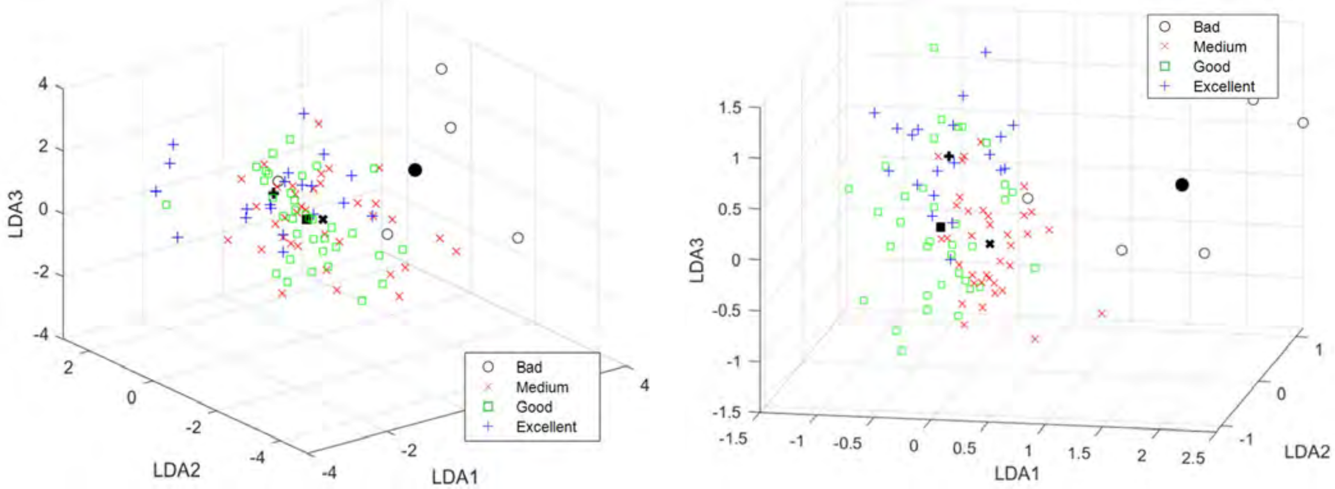


Figure 58 . Representation of the first taster's scores on the first three Fischer's principal components after applying PCA with 16 principal components (left) and 37 principal components (right).

In the second set of data the categories are separable from each other but less than without applying PCA previously, which is translated into a worse result of the subsequent LDA classifiers whose overall accuracy drops to 93.75% with 37 components and to 65.62% with 16. The results show that a large part of the samples in the *Medium*, *Good* and *Excellent* categories are again mixed with each other.

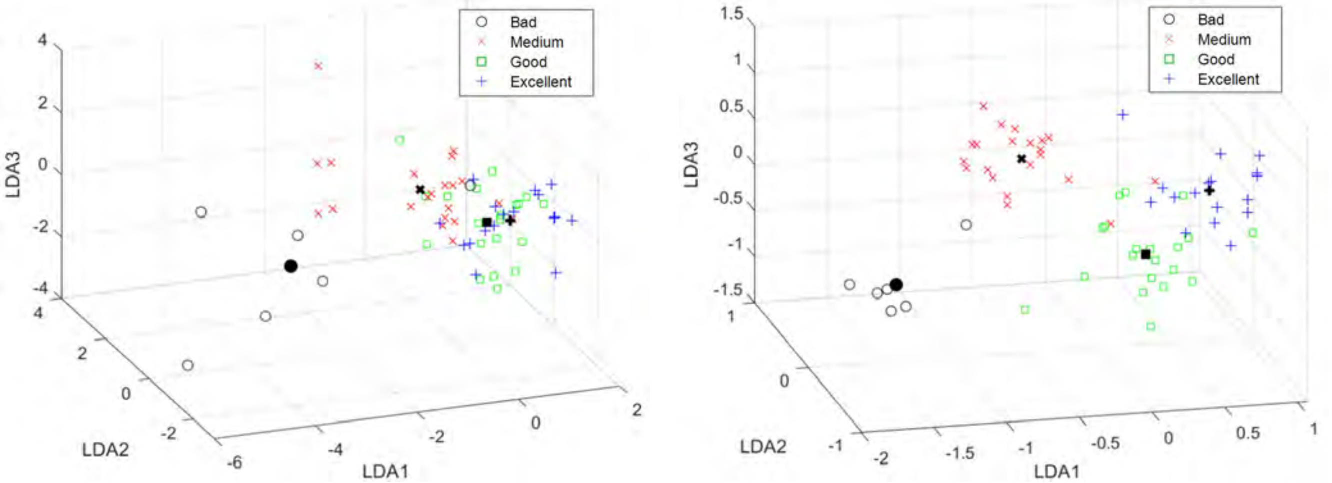


Figure59. Representation of the second taster's scores on the first three Fischer principal components after applying PCA with 16 principal components (left) and 37 principal components (right).

This may be due to the high variability of the data across the entire feature set, so that a large number of its principal components become necessary to correctly describe the samples and trimming the number of components may remove important information.

7.3 QDA

Following the corresponding steps explained in section 2.3 we applied QDA algorithm in the Matlab R2019b environment on the first taster dataset. Each of the internal covariance matrices of each class were orthogonalized producing 3 eigenvectors for each of them. Due to the singularity of the matrices $S_{intra,i}$ a regularization of the matrices was applied. To find the optimal regularization factors, we started from a value similar to the reference used in LDA (with a regularization factor of 10^{-5} times the average value of the elements of the main diagonal) and, on observing substantial differences between the results when validating the classifiers, it was decided to optimize its value as another design parameter. For this purpose, 1 sample from each class (a total of 4 samples) was extracted from the training dataset and used to validate the results by varying η_i incrementally and seeking an overall accuracy for the classifier as high as possible. In the first dataset the regularization factors obtained were. $\eta_1 = 1{,}45, \eta_2 = 7{,}6 \cdot 10^{-3}, \eta_3 = 8{,}6 \cdot 10^{-3}, \eta_4 = 4{,}37 \cdot 10^{-2}$. For the second dataset the factors were: $\eta_1 = 2{,}74, \eta_2 = 6{,}8 \cdot 10^{-2}, \eta_3 = 0{,}54, \eta_4 = 0{,}12$.

The percentages shown in Table 51 correspond to the importance taken by each eigenvalue in the orthogonalization of each covariance matrix, without it being possible to establish any relationship between the percentages corresponding to different matrices.

Using $S_{intra,1}$			Using $S_{intra,2}$		
No. Eigenvalue	Eigenvalue	Cumulative variance	No. Eigenvalue	Eigenvalue	Cumulative variance
1	0.962	50.98%	1	3.125	48.89%
2	0.803	93.57%	2	2.238	83.89%
3	0.121	100.00%	3	1.030	100.00%
Using $S_{intra,3}$			Using $S_{intra,4}$		
No. Eigenvalue	Eigenvalue	Cumulative variance	No. Eigenvalue	Eigenvalue	Cumulative variance
1	6.288	73.02%	1	4.281	53.53%
2	1.686	92.60%	2	2.543	85.33%
3	0.638	100.00%	3	1.173	100.00%

Table 51. Eigenvalues associated with the class-dependent Fischer's principal components and their contribution to the total variance on the first dataset.

Following exactly the same procedure with the second taster's data and with the regularization factors, $\eta_1 = 2{,}74, \eta_2 = 6{,}87 \cdot 10^{-2}, \eta_3 = 0{,}542, \eta_4 = 1{,}19 \cdot 10^{-1}$ we obtained the results shown in Table 52.

Using $S_{intra,1}$			Using $S_{intra,2}$		
No. Eigenvalue	Eigenvalue	Cumulative variance	No. Eigenvalue	Eigenvalue	Cumulative variance
1	0.827	63.05%	1	7.385	62.61%
2	0.340	88.95%	2	3.356	91.07%
3	0.145	100.00%	3	1.053	100.00%
Using $S_{intra,3}$			Using $S_{intra,4}$		
No. Eigenvalue	Eigenvalue	Cumulative variance	No. Eigenvalue	Eigenvalue	Cumulative variance
1	7.525	86.57%	1	27.752	90.75%
2	0.612	93.62%	2	1.7812	96.58%
3	0.555	100.00%	3	1.0469	100.00%

Table 52. Eigenvalues associated with the class-dependent Fischer's principal components and their contribution to the total variance on the second dataset.

Since there is no common representation space in this case it has no sense to provide a reduced representation of the data in the principal components.

When analyzing the contribution of the original features in the 3 principal components created for each matrix to be optimized, we obtained the results in Table 53 (first dataset) and Table 54 (second dataset).

	Importance (%)	Importance (%)	Importance (%)	Importance (%)
	until LDA3 Covariance class 1:	until LDA3 Covariance class 2:	until LDA3 Covariance class 3:	until LDA3 Covariance class 4:
pH	0.88%	2.49%	1.34%	1.27%
Dp-3-glc	1.76%	0.90%	1.11%	2.26%
Cy-3-glc	1.29%	5.05%	2.80%	2.30%
Pt-3-glc	3.49%	0.39%	0.87%	2.30%
Pn-3-glc	6.18%	0.46%	1.43%	4.57%
Mv-3-glc	2.07%	0.21%	0.75%	0.91%
Dp-3-(6-Ac)-glc	4.52%	1.52%	5.98%	2.63%
Pt-3-(6-Ac)-glc	0.30%	2.23%	2.72%	0.97%
Pn-3-(6-Ac)-glc	0.57%	0.45%	0.51%	0.87%
Mv-3-(6-Ac)-glc	2.90%	1.10%	2.76%	4.36%
Mv-3-(6-caff)-glc	0.53%	1.47%	1.32%	0.34%
Pt-3-(6-p-coum)-glc	0.89%	1.88%	4.12%	1.30%
Pn-3-(6-p-coum)-glc	1.36%	5.78%	3.65%	2.20%
Mv-3-(6-p-coum)-glc	1.54%	1.88%	1.15%	3.07%
Mv-3-glc-acetaldehído	0.43%	4.49%	3.24%	2.32%
Mv-3-glc-vinilmetilo	0.13%	0.74%	0.30%	0.10%
Mv-3-glc-pirúvico	1.63%	2.93%	2.87%	1.49%
Mv-3-glc-4-vinilfenol	0.94%	2.04%	2.04%	1.53%
Mv-3-glc-4-vinilcatecol	0.06%	1.51%	0.98%	1.21%
Mv-3-(6-p-coum)-glc cis	0.12%	0.68%	0.40%	0.21%
Mv-3-(6-p-coum)-glc trans	3.07%	1.11%	0.45%	4.03%
Mv-3-glc-4-vinilguaiacol	0.19%	0.53%	0.52%	0.41%
Mv-3-(6-p-coum)-glc-acetaldehído	0.06%	0.55%	0.43%	0.20%
Mv-3-(6-p-coum)-glc-pirúvico	0.27%	0.73%	1.11%	0.43%
Mv-3-(6-p-coum)-glc-4-vinilfenol	0.23%	0.47%	0.25%	0.24%
Catequin-Mv-3-glc	0.16%	0.34%	0.85%	0.31%
Epicatequin-Mv-3-glc	0.05%	0.14%	0.53%	0.09%
Mv-3-(6-p-coum)-glc-4-vinilguaiacol	0.02%	0.12%	0.20%	0.07%
(Epi)Galocatequin-Mv-3-glc	0.02%	0.07%	0.10%	0.08%
Catequin-Mv-3-(6-p-coum)-glc	0.02%	0.07%	0.22%	0.09%
(Epi)Galocatequin-Mv-3-(6-p-coum)-glc	0.00%	0.02%	0.06%	0.02%
Mv-3-(6-p-coum)-glc-8-etil-(epi)catequina 1	0.00%	0.08%	0.02%	0.01%
Catequina	4.92%	0.86%	0.89%	3.36%
Epicatequina	1.11%	2.53%	5.41%	1.85%
PCB1	3.09%	0.11%	0.14%	0.27%
PCB2	13.48%	0.43%	1.89%	2.56%
((epi)cat)2 1	0.57%	4.63%	4.06%	2.06%
PCC1	0.29%	2.65%	0.58%	0.67%
((epi)cat)3 1	0.11%	0.37%	0.68%	0.96%
((epi)cat)3 2	1.11%	4.52%	0.42%	2.99%
Galocatequina	1.10%	2.60%	1.10%	1.50%
Epigalocatequina	0.80%	3.64%	4.48%	0.94%
((epi)galocat)2 1	0.81%	4.18%	3.65%	1.15%
((epi)galocat)2 2	0.14%	1.14%	0.37%	0.46%
((epi)galocat)2 3	0.25%	1.45%	0.39%	0.12%
((epi)cat-(epi)galocat) 1	1.80%	2.75%	3.59%	5.08%
((epi)cat-(epi)galocat) 2	1.61%	0.96%	2.65%	3.03%
((epi)galocat)-(epi)cat) 1	1.24%	1.61%	3.50%	4.41%
((epi)galocat)-(epi)cat) 2	0.56%	1.05%	3.07%	1.94%
(epi)cat-glucosa 1	3.18%	2.37%	0.26%	2.06%
(epi)cat-glucosa 2	0.44%	1.48%	2.03%	1.06%
((epi)cat)2A 1	0.12%	0.40%	0.35%	0.22%
((epi)cat)2A 2	0.26%	4.43%	1.96%	2.03%
((epi)cat-(epi)galocat)A 1	0.45%	1.23%	1.03%	1.59%
((epi)cat-(epi)galocat)A 2	0.88%	3.21%	1.70%	1.93%
(epi)cat-((epi)cat-(epi)galocat) A 1	3.82%	2.34%	2.12%	3.73%
(epi)cat-((epi)cat-(epi)galocat) A 2	0.46%	1.31%	1.83%	1.73%
p-vinil(epi)cat 1	0.17%	0.86%	0.61%	0.23%
p-vinil(epi)cat 2	7.24%	0.73%	2.17%	2.61%
p-vinil(epi)cat 3	0.68%	1.64%	1.53%	3.41%
(epi)Cat-furfural-(epi)cat 1	4.82%	0.55%	1.05%	2.25%
(epi)Cat-furfural-(epi)cat 2	8.78%	1.55%	1.40%	1.62%

Table 53. Importance in % of each feature in the first 3 class-dependent Fischer's principal components in each of the optimized matrices (first dataset).

	Importance (%)	Importance (%)	Importance (%)	Importance (%)
	until LDA3 Covariance class 1:	until LDA3 Covariance class 2:	until LDA3 Covariance class 3:	until LDA3 Covariance class 4:
pH	0.14%	0.91%	0.08%	0.86%
Dp-3-glc	3.82%	1.87%	1.20%	4.16%
Cy-3-glc	1.77%	2.98%	2.81%	1.17%
Pt-3-glc	1.50%	4.31%	2.58%	1.57%
Pn-3-glc	4.39%	4.07%	14.57%	7.75%
Mv-3-glc	2.76%	0.18%	3.27%	1.20%
Dp-3-(6-Ac)-glc	3.82%	0.73%	0.70%	5.23%
Pt-3-(6-Ac)-glc	0.49%	1.57%	0.20%	0.52%
Pn-3-(6-Ac)-glc	0.66%	1.30%	0.72%	0.75%
Mv-3-(6-Ac)-glc	4.67%	3.93%	5.18%	5.05%
Mv-3-(6-caff)-glc	0.43%	4.02%	0.12%	0.12%
Pt-3-(6-p-coum)-glc	0.93%	1.79%	3.36%	1.65%
Pn-3-(6-p-coum)-glc	0.15%	2.62%	0.76%	0.31%
Mv-3-(6-p-coum)-glc	3.16%	4.90%	9.76%	2.56%
Mv-3-glc-acetaldehído	0.98%	0.43%	0.07%	0.27%
Mv-3-glc-vinilmetilo	0.05%	0.23%	0.04%	0.12%
Mv-3-glc-pirúvico	0.27%	0.85%	0.25%	0.67%
Mv-3-glc-4-vinilfenol	0.57%	1.41%	0.28%	1.01%
Mv-3-glc-4-vinilcatecol	0.13%	0.71%	0.30%	0.12%
Mv-3-(6-p-coum)-glc cis	0.13%	0.40%	0.24%	0.16%
Mv-3-(6-p-coum)-glc trans	5.87%	5.97%	14.46%	12.41%
Mv-3-glc-4-vinilguaiacol	0.10%	0.22%	0.39%	0.37%
Mv-3-(6-p-coum)-glc-acetaldehído	0.12%	0.11%	0.14%	0.06%
Mv-3-(6-p-coum)-glc-pirúvico	0.12%	0.29%	0.08%	0.44%
Mv-3-(6-p-coum)-glc-4-vinilfenol	0.11%	0.17%	0.20%	0.53%
Catequin-Mv-3-glc	0.17%	0.13%	0.50%	0.32%
Epicatequin-Mv-3-glc	0.04%	0.04%	0.12%	0.04%
Mv-3-(6-p-coum)-glc-4-vinilguaiacol	0.02%	0.04%	0.06%	0.07%
(Epi)Galocatequin-Mv-3-glc	0.01%	0.02%	0.08%	0.04%
Catequin-Mv-3-(6-p-coum)-glc	0.01%	0.01%	0.07%	0.03%
(Epi)Galocatequin-Mv-3-(6-p-coum)-glc	0.00%	0.01%	0.02%	0.01%
Mv-3-(6-p-coum)-glc-8-etil-(epi)catequina 1	0.01%	0.09%	0.02%	0.01%
Catequina	2.20%	4.82%	1.35%	5.66%
Epicatequina	1.16%	1.43%	1.40%	2.05%
PCB1	2.76%	0.17%	0.19%	0.27%
PCB2	12.58%	1.90%	2.83%	0.70%
((epi)cat)2 1	0.47%	3.43%	0.38%	0.72%
PCC1	0.54%	0.25%	0.27%	0.75%
((epi)cat)3 1	0.12%	0.16%	0.20%	0.21%
((epi)cat)3 2	1.11%	0.97%	1.24%	2.14%
Galocatequina	0.79%	1.22%	0.59%	0.30%
Epigalocatequina	1.41%	1.89%	0.50%	0.78%
((epi)galocat)2 1	0.75%	0.87%	0.12%	0.32%
((epi)galocat)2 2	0.09%	0.83%	0.08%	0.16%
((epi)galocat)2 3	0.13%	0.52%	0.14%	0.38%
((epi)cat-(epi)galocat) 1	1.59%	2.18%	1.05%	4.24%
((epi)cat-(epi)galocat) 2	3.14%	4.10%	1.81%	4.78%
((epi)galocat)-(epi)cat) 1	0.28%	5.96%	1.38%	1.90%
((epi)galocat)-(epi)cat) 2	0.62%	1.84%	1.19%	2.09%
(epi)cat-glucosa 1	2.95%	2.76%	4.64%	7.44%
(epi)cat-glucosa 2	1.24%	2.66%	3.11%	2.02%
((epi)cat)2A 1	0.08%	0.22%	0.02%	0.18%
((epi)cat)2A 2	0.34%	1.16%	0.89%	1.00%
((epi)cat-(epi)galocat)A 1	0.43%	0.68%	0.30%	0.36%
((epi)cat-(epi)galocat)A 2	0.93%	2.54%	0.74%	0.57%
(epi)cat-((epi)cat-(epi)galocat) A 1	3.52%	3.18%	3.61%	4.79%
(epi)cat-((epi)cat-(epi)galocat) A 2	0.35%	0.64%	0.28%	0.22%
p-vinil(epi)cat 1	0.11%	0.22%	0.15%	0.06%
p-vinil(epi)cat 2	6.03%	2.28%	4.13%	0.83%
p-vinil(epi)cat 3	1.35%	1.31%	1.91%	1.44%
(epi)Cat-furfural-(epi)cat 1	8.68%	2.32%	1.32%	0.34%
(epi)Cat-furfural-(epi)cat 2	6.84%	1.17%	1.57%	3.74%

Table 54. Importance in % of each feature in the first 3 class-dependent Fischer's principal components in each of the optimized matrices (second dataset).

In the absence of a relationship of importance between each of the components of the different classes, we have chosen to analyze which of the original features were present among the top 10 most important of the proposed orthogonalization for each class. A summary of this information can be seen in Table 55. Those features that were not among the top 10 have been eliminated to get a more compact representation.

Order in ...	First dataset				Second dataset			
	$S_{intra,1}$	$S_{intra,2}$	$S_{intra,3}$	$S_{intra,4}$	$S_{intra,1}$	$S_{intra,2}$	$S_{intra,3}$	$S_{intra,4}$
Dp-3-glc					9			10
Cy-3-glc		2						
Pt-3-glc	9					5		
Pn-3-glc	4			2	7	7	1	2
Mv-3-glc							9	
Dp-3-(6-Ac)-glc	7		1		8			5
Mv-3-(6-Ac)-glc				4	6	9	4	6
Mv-3-(6-caff)-glc						8		
Pt-3-(6-p-coum)-glc			4				8	
Pn-3-(6-p-coum)-glc		1	6					
Mv-3-(6-p-coum)-glc				9		3	3	
Mv-3-glc-acetaldehyde		5	10					
Mv-3-glc-pyruvic		10						
Mv-3-(6-p-coum)-glc cis								
Mv-3-(6-p-coum)-glc trans				5	5	1	2	1
Catechin	5			8		4		4
Epicatechin			2					
PCB2	1				1			
((epi)cat)$_2$ 1		3	5			10		
((epi)cat)$_3$ 2		4						
Epigallocatechin		8	3					
((epi)cat-(epi)galocat) 1								9
((epi)cat-(epi)galocat) 2						6		8
((epi)galocat)-(epi)cat) 1						2		
(epi)cat-glucose 1							5	3
(epi)cat-glucose 2							10	
(epi)cat-((epi)cat-(epi)cat-(epi)galocat) A 1					10		7	7
p-vinyl(epi)cat 2					4		6	
(epi)Cat-furfural-(epi)cat 1					2			
(epi)Cat-furfural-(epi)cat 2					3			

Table 55. Importance of the first 10 features in the different optimized matrices.

The original features that are most present in the various ordinations are 21: Cy-3-glc, Pn-3-glc, Dp-3-(6-Ac)-glc, Mv-3-(6-Ac)-glc, Pt-3-(6-p-coum)-glc, Pn-3-(6-p-coum)-glc, Mv-3-(6-p-coum)-glc, Mv-3-(6-p-coum)-glc cis, Mv-3-(6-p-coum)-glc trans, Catechin, Epicatechin, PCB2, ((epi)cat)$_2$ 1, ((epi)cat)$_3$ 2, Epigallocatechin, ((epi)galocat)-(epi)cat) 1, (epi)cat-glucose 1, (epi)cat-((epi)cat-(epi)cat-(epi)galocat)A 1, p-vinyl(epi)cat 2, (epi)Cat-furfural-(epi)cat 1, (epi)Cat-furfural-(epi)cat 2. This group of features named *QDA features* will be selected in future classifiers to measure their performance.

We now apply QDA for classification. Using equation (58) we calculate the probabilities that the samples belong to each class. In the first dataset, the classifier with standard loss function (see Figure 60) gives an overall accuracy of 88.54%. The model validated by LOO shows similar results. All its positive recommendations are right, all the wines recommended as *Excellent* were truly *Excellent*, but it loses a 30% of the possible positive recommendations, it doesn't recommend 6 truly *Excellent* wines. Looking at the negative recommendations, again, all are correct but losing 40% of the truly *Bad* wines. Fortunately none of them would be recommended positively.

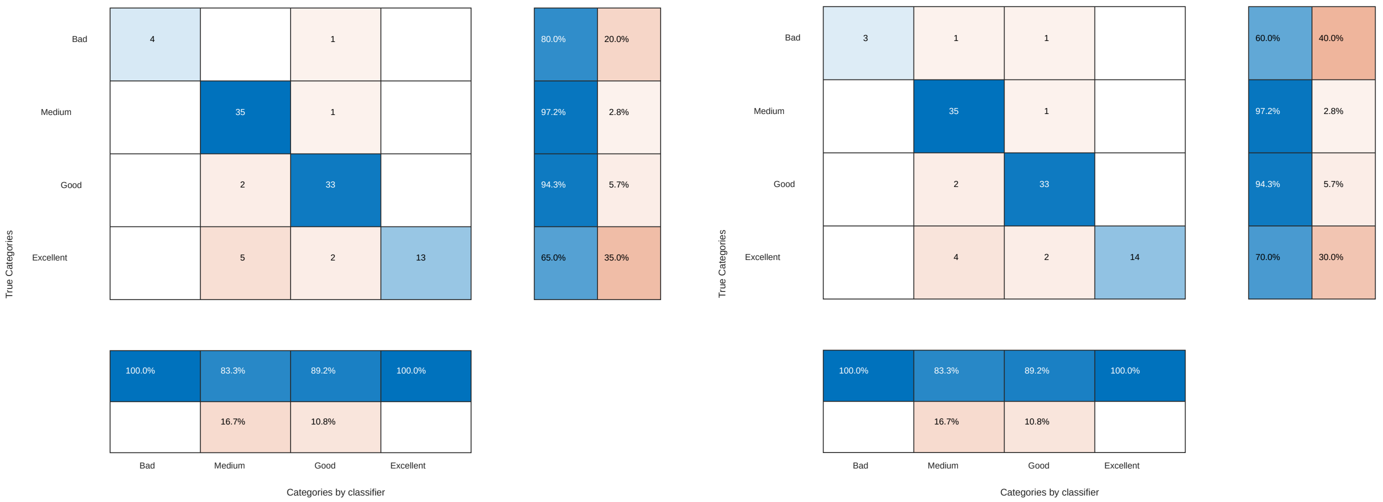


Figure 60. Confusion matrix of the standard losses QDA classifier on the first dataset using all samples (left) and LOO validated (right).

When we apply to the classifier the loss matrix proposed in Table 38 it will try to raise the level of certainty of both positive and negative recommendations at the cost of losing wine recommendations that would be recommended. The results of the the classifiers based on all samples and the LOO validated ones can be seen in Figure 50. Figure 43. Since the hit rate among positive recommendations was already high, it offers no improvement over the classifier with the standard loss function.

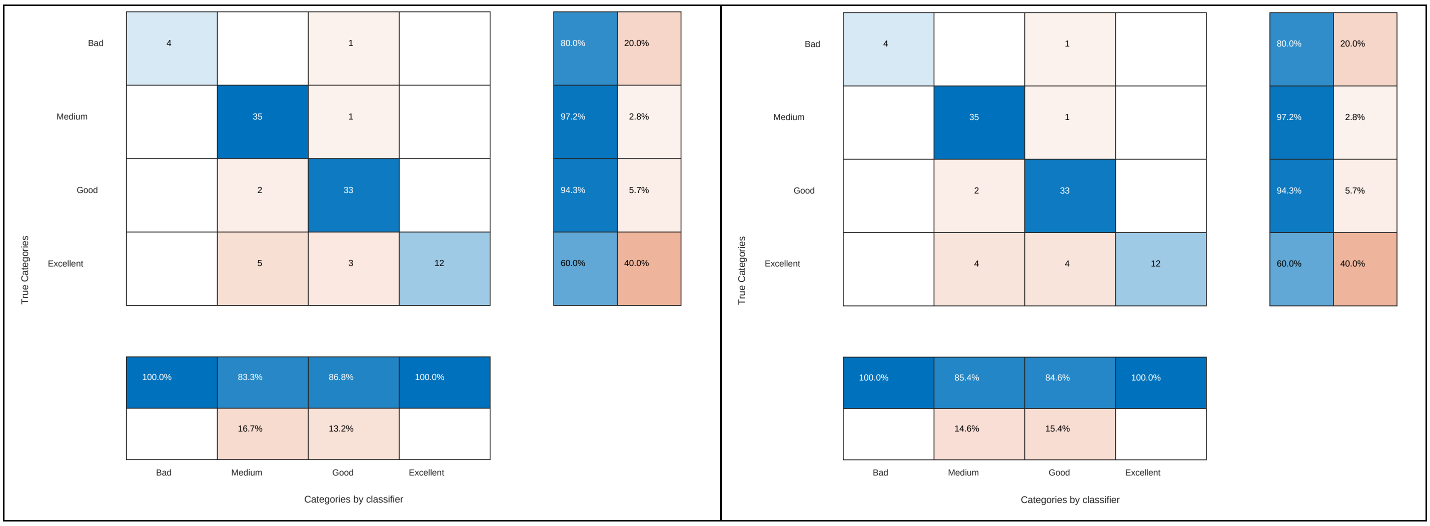


Figure 61. Confusion matrix of the QDA classifier with proposed losses on the first dataset using all samples (left) and LOO validated (right).

When referring to the second dataset, the classifier with standard loss function (see Figure 62) build with all the samples and the validated model show similar behavior (just with differences in a specific sample of the *Excellent* category that would not be well classified). The validated classifier gives an overall accuracy of 82.81% with a missclassification of 33.3% of the wines in the real *Bad* category (and all the *Bad* recommendations correct), with only two *Excellent* wines lost in the positive recommendations (11.8% of *Excellent* wines lost), but with 16.7% of the wines wrongly proposed as *Excellent*.

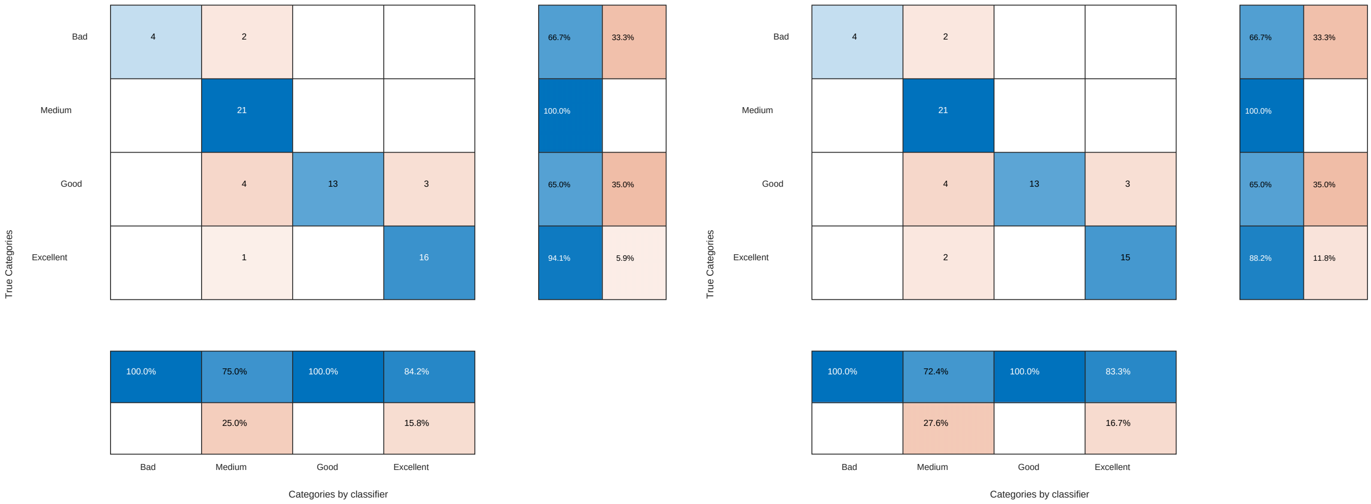


Figure 62. Confusion matrix of the standard losses QDA classifier on the second dataset using all samples (left) and LOO validated (right).

By applying the loss matrix (Figure 63) the overall results are very similar with a small degradation of the overall accuracy to 81.25%.

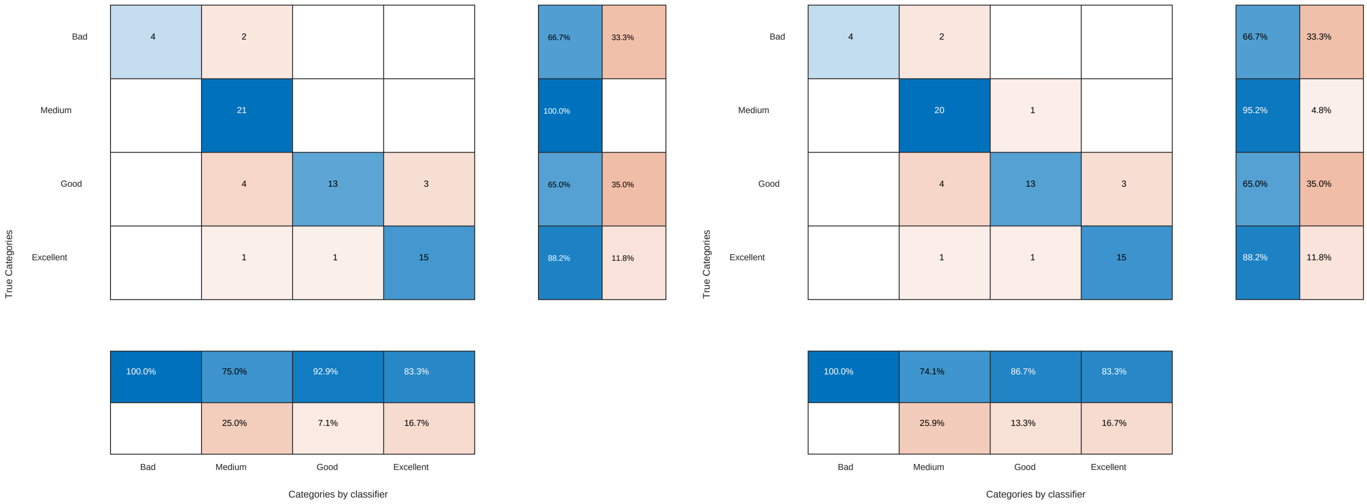


Figure 63. Confusion matrix of the QDA classifier with proposed losses on the second dataset using all samples (left) and LOO validated (right).

7.4 Naïve Bayes

As mentioned in section 2.1 the Näive Bayes classifier assumes that the different features that make up the data are independent of each other, thus estimating $P(\boldsymbol{x}|C_j)$ as a product of the distributions of each feature $P(x_i|C_j)$. The correlation matrix of the original features (see section 6.4) shows strong indications of dependence between them, so it is preferable to apply NB on uncorrelated features such as those obtained through PCA or Fischer discriminants, whose orthogonalization process achieves diagonal correlation matrices.

If we start from the first taster's data and its three principal components in the class-independent Fischer's discriminants, the first step we will need to perform is to find some sort of probability distributions that adequately model each $P(x_i|C_j)$. The sparse representatives of the first class make it very difficult to predict the type of distribution that each of the individual features in this class follows. Given that the number of samples in the *Bad* class is only 5 samples, and that they are insufficient to be able to identify the type of distribution, we will start with the classes with a larger number of samples and extrapolate the findings to this first class. In the Figure 46 different histograms made with the samples previously grouped by each of the classes have been represented and fitted according to two probability distributions: Gaussian distribution and an approximation by superposition of Gaussian kernels [68, 197].

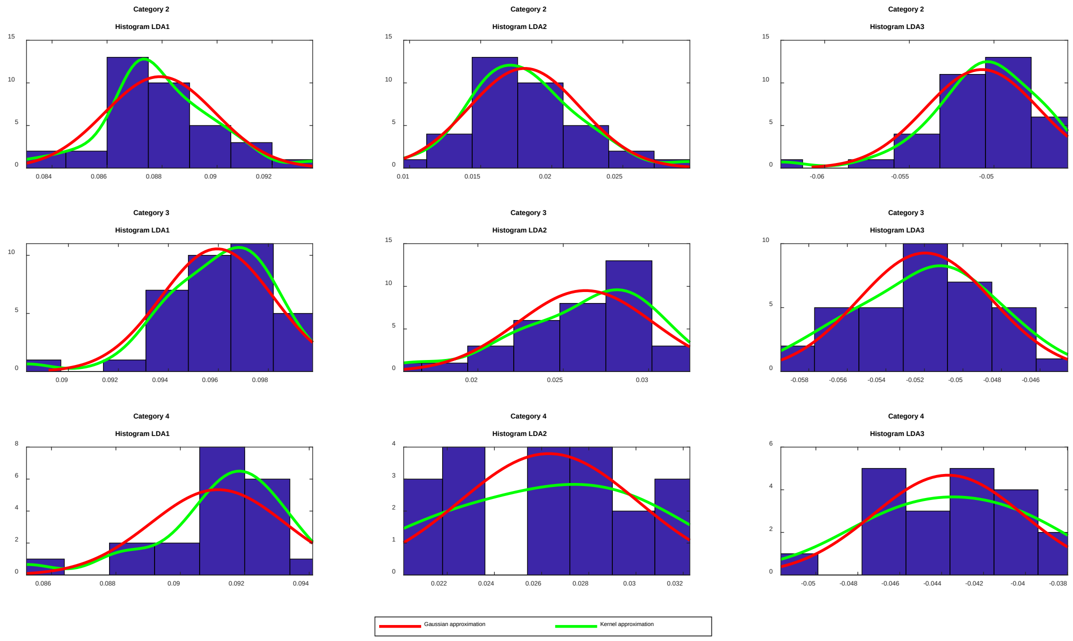


Figure 64. Histograms and approximations of different probability distributions on the first taster's data expressed in the LDA components.

The goodness-of-fit of the distributions is measured by the Kolmogorov-Smirnov test [198]. The distributions seem to be able to fit correctly to the *Medium* and *Good* categories, but the lower number of samples in the *Excellent category*, especially in the case of the second component, does not allow final conclusions to be drawn about the optimal distribution in this case. After verifying that any of them could be a good candidate, 4 Naïve Bayes classifiers were constructed and validated following the steps indicated in section 2.1, two of them with standard losses and two of them with the losses of Table 38. In these classifiers the same type of distribution was used for all features (one pair assuming a Gaussian probability distribution for all features, another pair with kernel distribution).

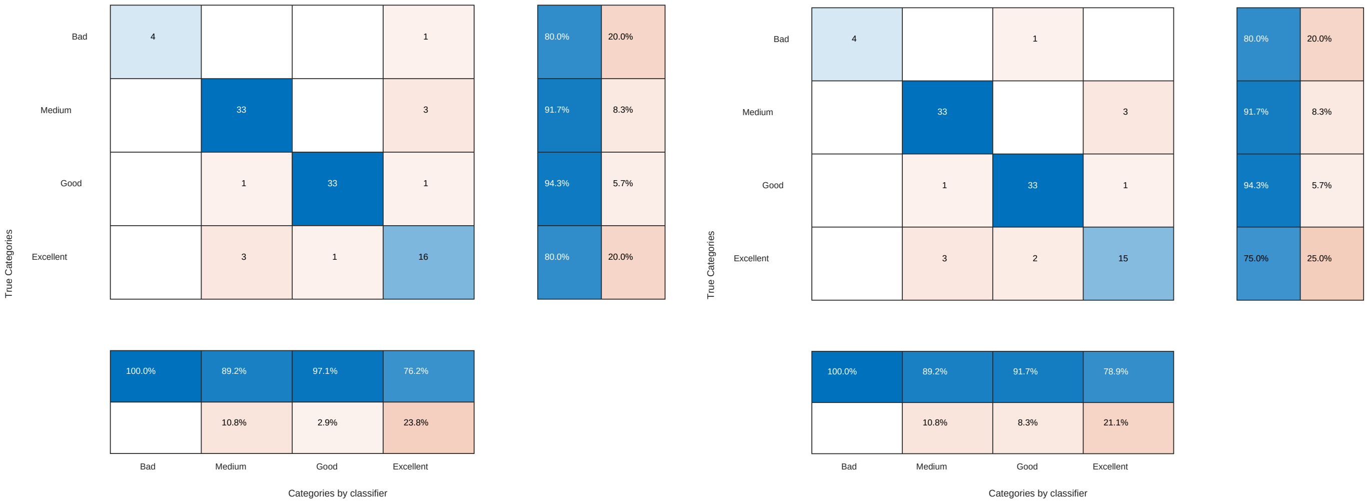


Figure 65. Confusion matrix of the LOO validated NB classifier with Gaussian distributions with standard losses (left) and with proposed losses (right).

The classifiers constructed using this methodology were validated using LOO by obtaining the confusion matrices from the Figure 65 for the Gaussian distributions and those Figure 66 for the kernel distributions. It can be seen that, while obtaining similar overall results of around 88% accuracy, the kernel type distributions produce more reliable positive recommendations with 80% of the recommended samples being correct.

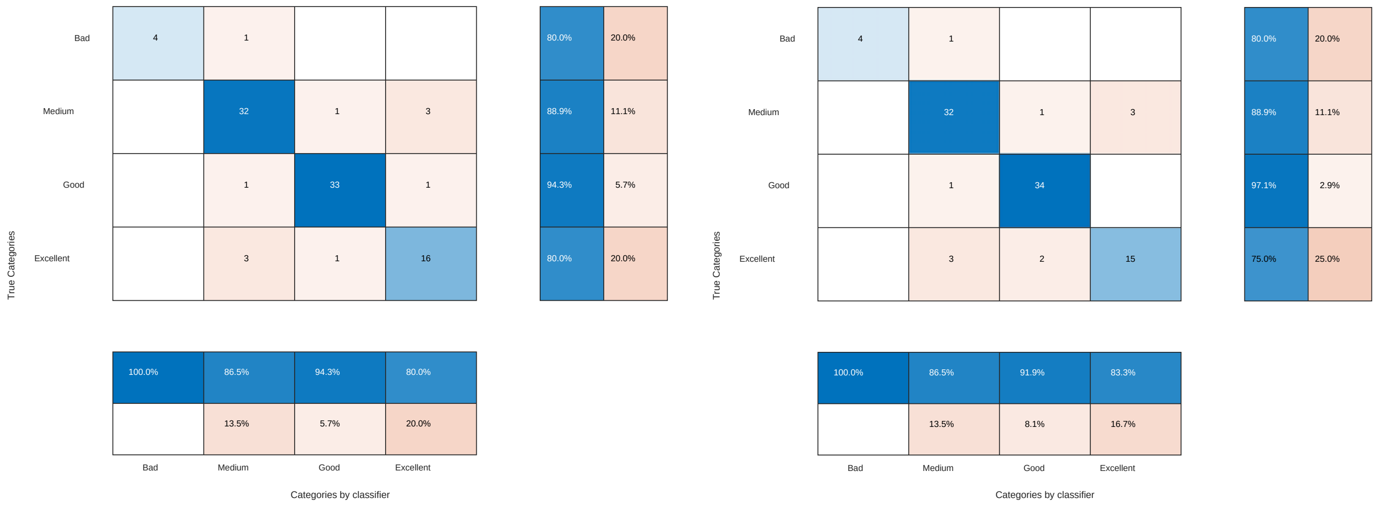


Figure 66. Confusion matrix of the LOO validated NB classifier with kernel distributions with standard losses (left) and with proposed losses (right).

Following a similar procedure on the different groups of features that can represent the first taster's data and validating the classifiers through LOO the results in Table 56 were obtained. Only the class-dependent and class-independent Fischer's components achieve good results. The interdependencies between the original features and the anthocyanin derivatives partly justify the poor results for their classifiers, as NB assumes that the features are linearly independent and the correlation matrix showed dependencies.

		No Losses		At a loss	
Features		Normal Distribution	Kernel Distribution	Normal Distribution	Kernel Distribution
Originals		29.17%	23.96%	29.17%	25%
Anthocyanin derivatives		30.21%	31.25%	34.38%	30.21%
LDA Selection$_1$		26.04%	26.04%	26.04%	32.29%
QDA Selection		31.25%	19.79%	31.25%	28.13%
PCA	100	33.33%	22.92%	34.38%	23,96%
	99	28.13%	28.13%	25%	27,08%
	95	31.25%	33.33%	30.21%	34,38%
	90	28.13%	29.17%	34.38%	35,42%
LDA		89.58%	88.54%	88.54%	88.54%
QDA		89.58%	85.42%	89.58%	85.42%

Table 56. Overall results of LOO validated Naïve Bayes classifiers on the first dataset.

An analogous procedure was followed for the data from the second taster. The 6 samples of the *Bad* category are insufficient to test any probability distribution, so the goodness-of-fit was first measured on the features of the other categories. You can see the distributions calculated on the histograms of the class-independent Fischer's component features (LDA) In Figure 67.

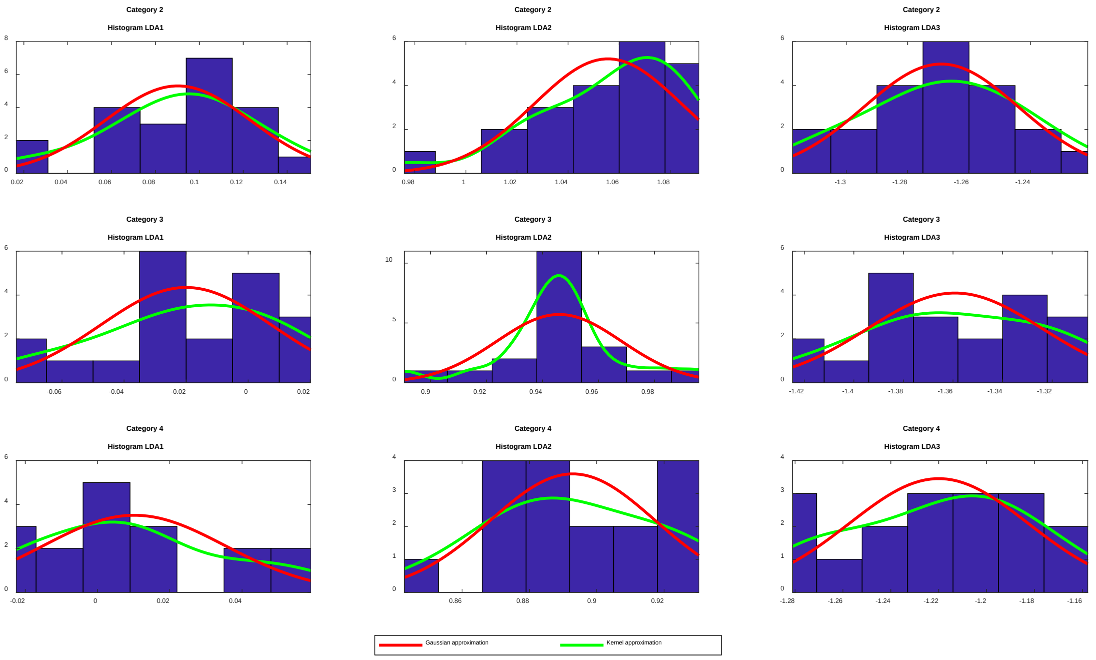
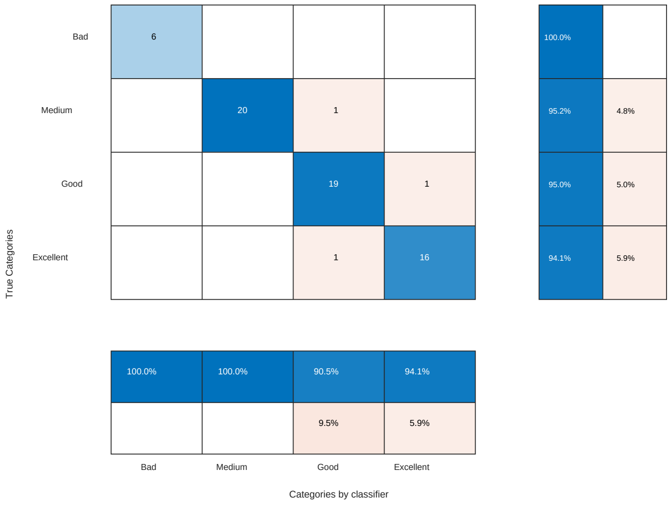


Figure 67. Histograms and approximations of different probability distributions on the second taster's data expressed in the LDA components.

The smaller number of samples in this case makes many of the distributions of doubtful application, but when constructing the classifiers assuming these distributions are true, the results are similar, and even better than the classifiers on the first dataset, with overall accuracies exceeding 95% once validated by LOO. You can see their confusion matrix in Figure 68, which in this case is coincident for the classifiers with normal distributions (with standard and proposed losses) and the classifier with kernel-type distributions with proposed losses.



	Bad	Medium	Good	Excellent		
Bad	6				100.0%	
Medium		20	1		95.2%	4.8%
Good			19	1	95.0%	5.0%
Excellent			1	16	94.1%	5.9%
	100.0%	100.0%	90.5%	94.1%		
			9.5%	5.9%		

Figure 68. Confusion matrix of the LOO validated NB classifier with both losses and Gaussian distributions, and with proposed loss matrix and kernel distribution for the second taster's data.

The Figure 69 shows the confusion matrix achieved by validating the NB classifier with lossless kernel-type distributions, whose behavior is the best so far in all classifiers.

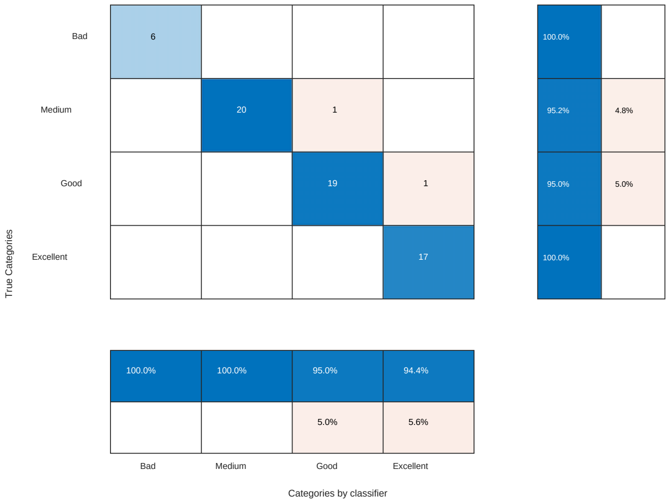


Figure 69. Confusion matrix of the LOO validated NB classifier with kernel-type distributions and standard losses.

Applying the same methodology on the other sets of features yields similar results (see Table 57) compared with those of the first dataset.

		No Losses		At a loss	
Features		**Normal Distribution**	**Kernel Distribution**	**Normal Distribution**	**Kernel Distribution**
Originals		34,38%	32,81%	34,38%	32,81%
Anthocyanin derivatives		39,06%	35,94%	32,81%	34,38%
LDA Selection$_1$		37,5%	35,94%	35,94%	32,81%
QDA Selection		35,94%	34,38%	29,69%	37,5%
PCA	**100**	17,19%	25%	18,75%	23,44%
	99	26,56%	32,81%	31,25%	32,81%
	95	32,81%	39,06%	32,81%	34,38%
	90	37,5%	32,81%	31,25%	32,81%
LDA		95,31%	96,88%	95,31%	95,31%
QDA		81,25%	84,38%	81,25%	84,38%

Table 57. Overall results of LOO validated Naïve Bayes classifiers on the second dataset.

The major difference lies in the results on the class-dependent Fischer's components whose NB classifiers behave worse. An analysis of their behavior through the confusion matrices (see Figure 70) shows, however, that all the recommendations it issues are correct, with a greater tendency to discard both negative and positive recommendations, but perhaps working with greater certainty.

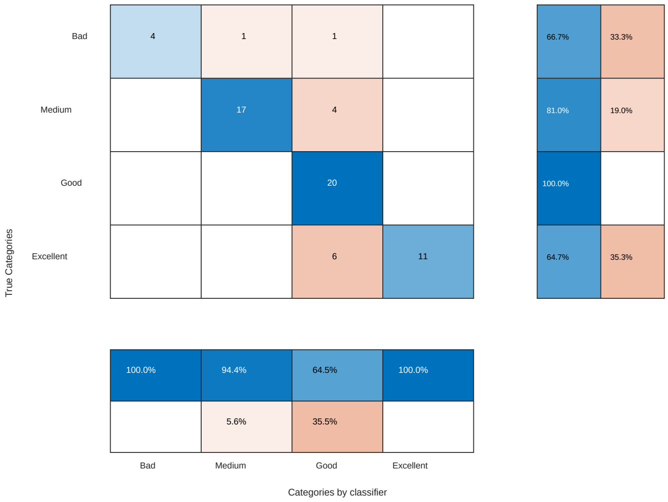


Figure 70. Confusion matrix of the LOO validated NB classifier with both loss matrices and kernel-type distributions over QDA features.

7.5 kNN

When we apply the kNN algorithm on the datasets as described in section 2.4 we must treat the number of neighbors to be chosen and the metric used to measure the distances as design parameters. In order to choose the most favorable option we pre-select the distance metric (Euclidean or Manhattan) and reserve one sample to test the classifiers using LOO. From the remaining samples we reserve one to validate the classifier and select the number of neighbors and use the remaining ones for training. Thus, in the case of the first data set we obtain 96 different classifiers with 95 validated variations for each one according to the number of neighbors. With each of the variations we form its confusion matrix and calculate the total losses by applying on the confusion matrix the loss matrix. Finally, we averaged the costs among the 95 experiments performed, choosing the optimal number of neighbors. It should be noted that the samples reserved for testing the final classifier were never used to obtain this value.

The first dataset shows very different results depending on the features selected to discriminate the data. The results in classifiers with the standard loss matrix and with the loss matrix suggested in section 6.4 taking the 62 original features show overall accuracies around 45%. If only the anthocyanin derivatives are taken to represent the data or the so-called *QDA features* the results are even worse, with accuracies of 36% to 42%. Similar results are obtained working with the so-called LDA_1 features where the overall accuracy is again around 45%.

When proceeding with the feature space obtained by applying PCA on the data the results are not good either. Using all PCA features the accuracies are in the range 45% to 48%. With the 37 PCA features associated with the highest eigenvalues the results drop to 43%-47%, with 23 features 42%-46% and with 16 features 43%-47%.

All these classifiers show a clear tendency to classify in the majority classes with almost no cases classified as *Excellent* or *Bad*, which prevents any positive or negative predictions. This is because the kNN algorithm is not able to find a good representation of the data, remembering us that it needs to work with a separable representation of the categories in its input.

Thanks to the separation that was achieved in the LDA principal components, the application of kNN on this new dimensional space is remarkably better with accuracies somewhat below 90%. If we represent the samples by the three LDA principal components the optimal number of neighbors among all the variants with a Euclidean metric was 4 and 8 neighbors (Figure 71).

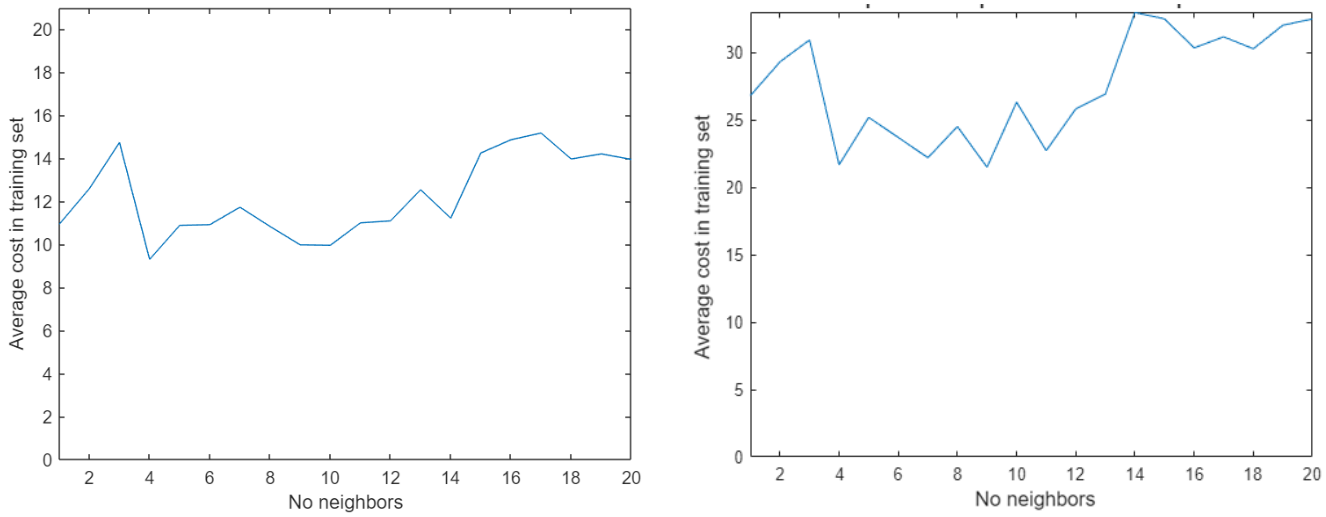


Figure 71. Averaged cost over the validation data as a function of the number of number of neighbors for kNN classifiers with standard loss matrix (left) with proposed loss marix (right) on the first dataset, Euclidean metric.

You can see the confusion matrices for the 4-neighbor kNN classifier with Euclidean distance validated by LOO in Figure 72. The overall accuracy of the classifier with standard losses is 89.58%, while for the proposed losses classifier it is 87.5%.

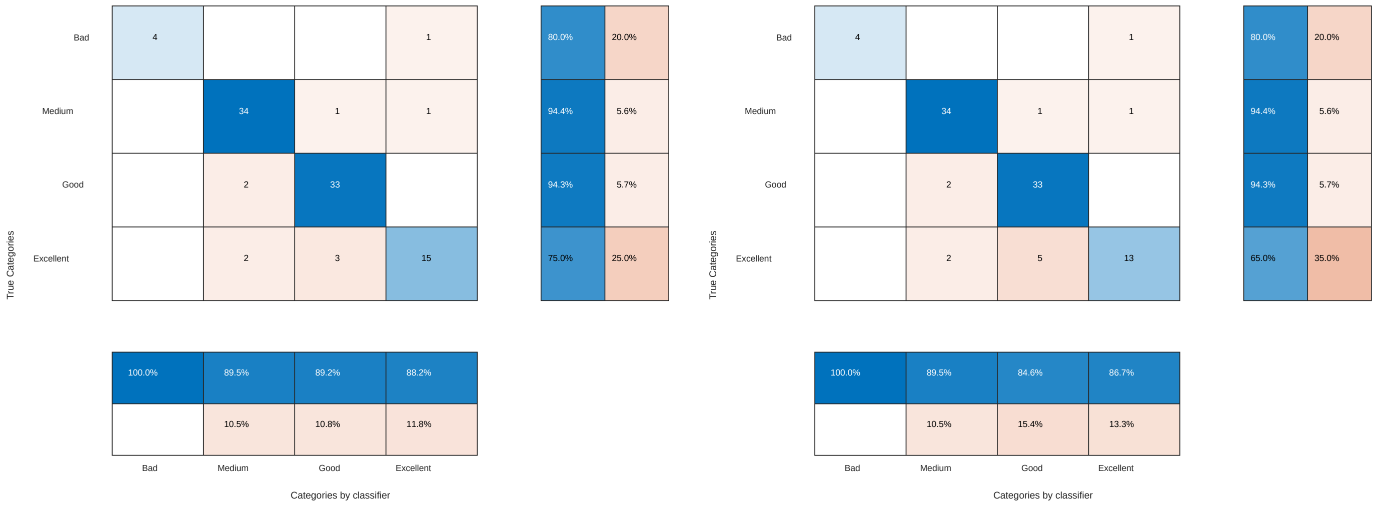
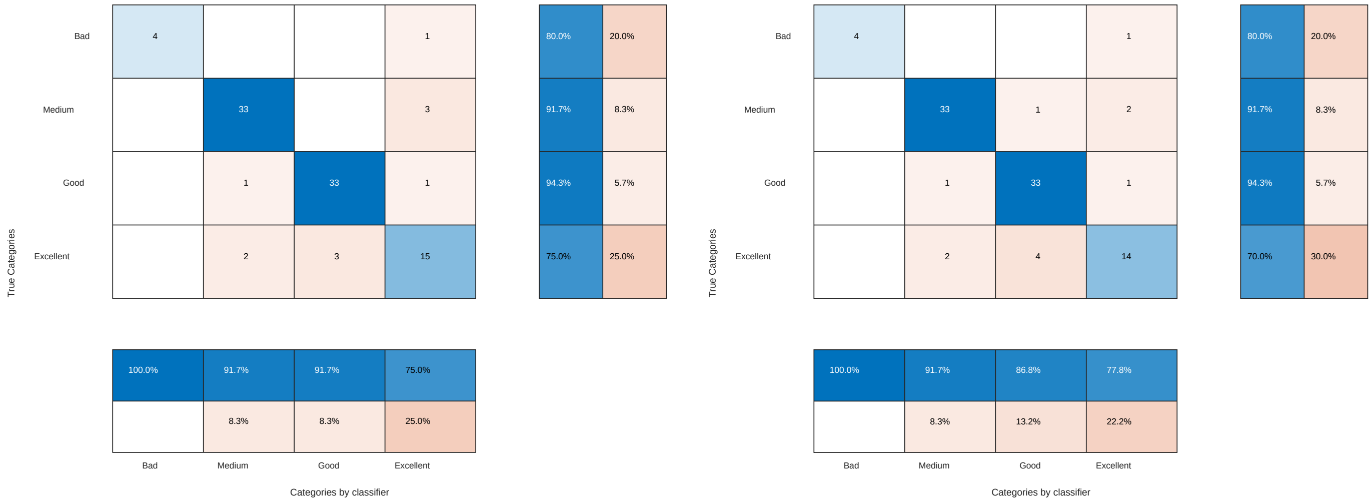


Figure 72. Confusion matrix of the LOO validated kNN classifier with 4 neighbors and Euclidean distance with standard losses (left) and proposed losses (right) on the first dataset.

If the number of neighbors is increased to 8 the overall results for the proposed loss classifier are maintained at the cost of a higher error in the positive predictions, but losing fewer truly *Excellent* wines in those predictions. The standard loss classifier achieves better results with 4 neighbors.



True Categories	Bad	Medium	Good	Excellent		
Bad	4			1	80.0%	20.0%
Medium		33		3	91.7%	8.3%
Good		1	33	1	94.3%	5.7%
Excellent		2	3	15	75.0%	25.0%
	100.0%	91.7%	91.7%	75.0%		
		8.3%	8.3%	25.0%		

Categories by classifier

True Categories	Bad	Medium	Good	Excellent		
Bad	4			1	80.0%	20.0%
Medium		33	1	2	91.7%	8.3%
Good		1	33	1	94.3%	5.7%
Excellent		2	4	14	70.0%	30.0%
	100.0%	91.7%	86.8%	77.8%		
		8.3%	13.2%	22.2%		

Categories by classifier

Figure 73. Confusion matrix of the LOO validated kNN classifier with 8 neighbors and Euclidean distance with standard losses (left) and proposed losses (right) on the first dataset.

Analyzing the behavior of the classification regions in the first two Fischer's components (Figure 74) it can be seen that the classifier with the proposed losses tries to ensure the positive and negative predictions, ensuring the cases predicted as *Excellent* and *Bad* by decreasing the regions assigned to these classes. This can be interpreted as an increase of some safety margin at the boundary of these regions.

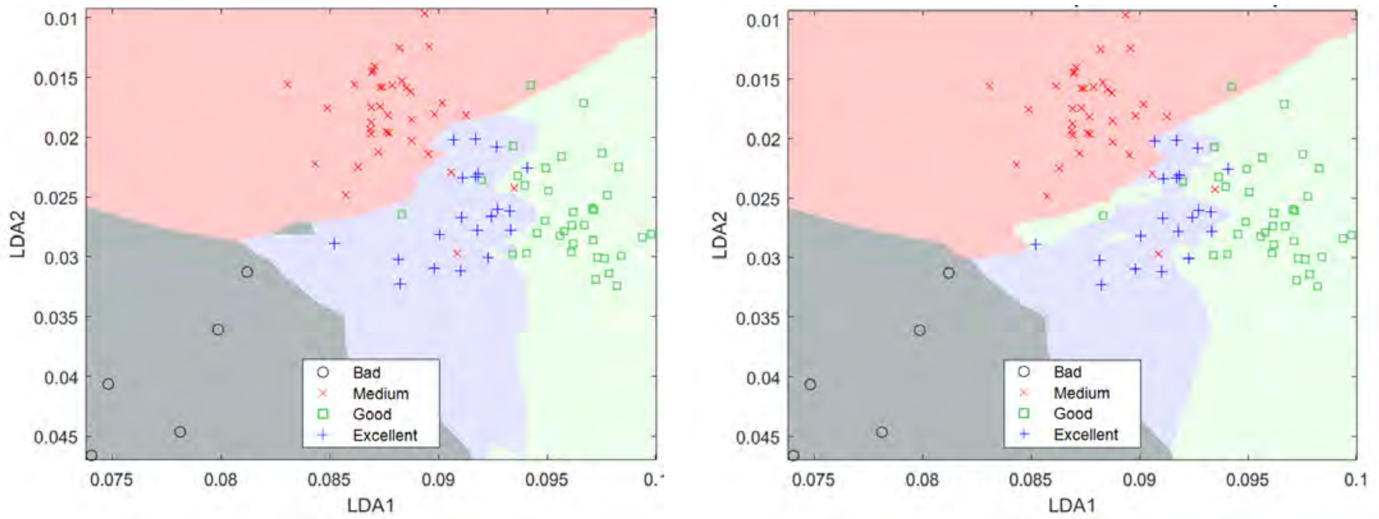


Figure 74. Classification regions using the first two Fischer's principal components for the 4-neighbor kNN and Euclidean distance. Standard losses classifier is on the left and proposed losses one on the right.

If we opt for the Manhattan distance metric, where the distance between two vectors is calculated by accumulating the absolute value of the differences in each dimension, ie, $d(\boldsymbol{a}, \boldsymbol{b}) = \sum|a_i - b_i|$ the results are similar, being somewhat better for the proposed losses matrix classifier but with a completely wrong positive recommendation.

When selecting the number of neighbors for this type of distance according to the results on the validation groups (Figure 75) the optimal values are 4 neighbors for the standard loss matrix classifier and 7 neighbors for the proposed loss matrix.

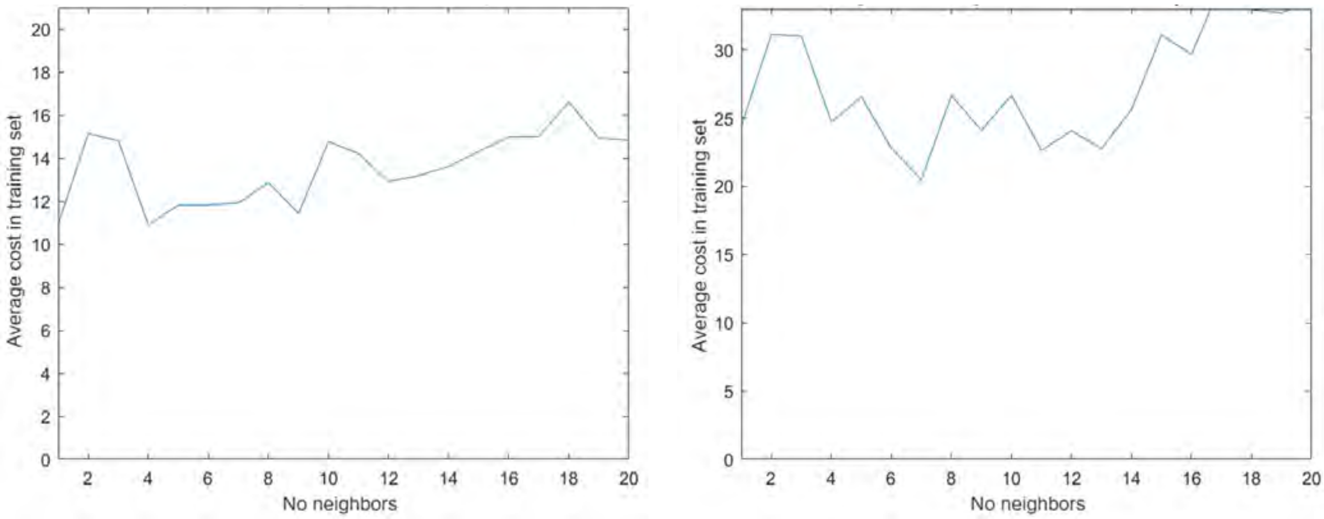


Figure 75. Averaged cost over the validation data as a function of the number of number of neighbors for kNN classifiers with standard loss matrix (left) with proposed loss marix (right) on the first dataset, Manhattan metric.

Manhattan distance classifiers have classification regions (Figure 76) different from those with Euclidean distance, with a tendency to form regions of separation on the axes on which the distances are measured. In this case the classifiers with the proposed losses also tend to increase the regions of the *Medium* and *Good* categories to the detriment of the *Bad* and *Excellent ones*.

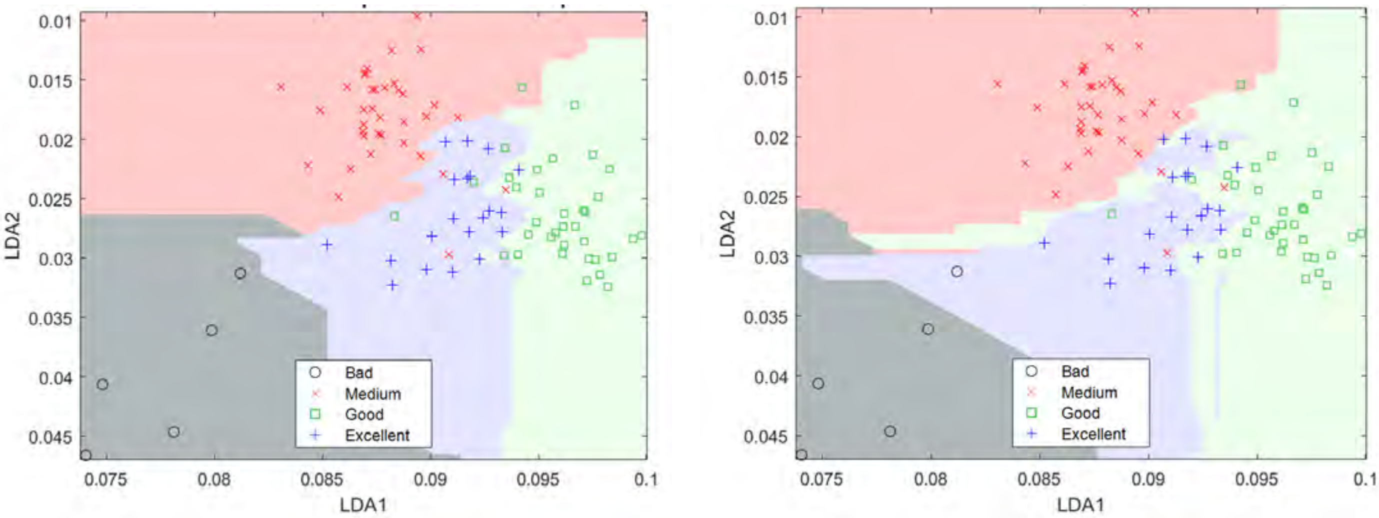


Figure 76. Classification regions using the first two Fischer's principal components for the 4-neighbor kNN and Manhattan distance. Standard losses classifier is on the left and proposed losses one on the right.

Once the type of distance and the number of neighbors have been selected when validating the classifiers by LOO on the samples reserved for testing, the overall accuracies are 88.54% for the standard losses case and 87.5% for the proposed losses case. Their confusion matrices can be seen in Figure 77.

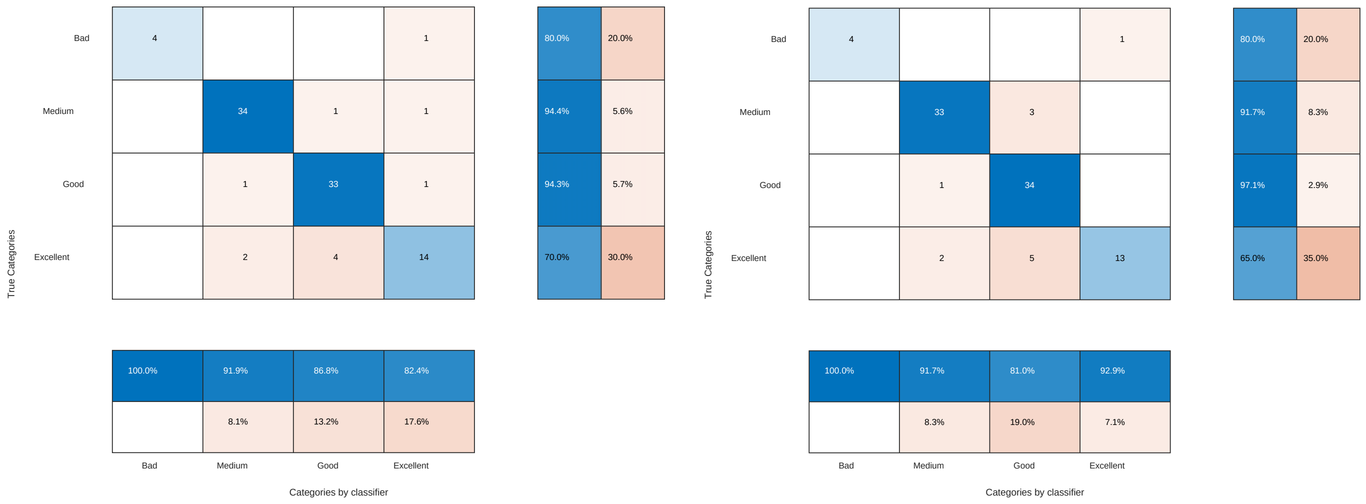


Figure 77. Confusion matrix of the LOO validated kNN classifier with 4 neighbors and Mahattan distance with standard losses (left) and proposed losses (right) on the first dataset.

When we express the first dataset in the space defined by the Fischer's method dependent on each class (the space of principal components previously used with the QDA classifier) and apply the same methodology, optimal results are obtained with 2 neighbors for the standard losses classifiers of Euclidean distance and 9 and 6 neighbors for the classifier with proposed losses and Manhattan metric. When validating these classifiers by LOO, we obtain the results of Figure 78 and Figure 79 where it is observed that the kNN classifier presents more problems in these components than in the LDA components (independent of each class). The overall accuracies drop to around 40% and the poor classification of the *Excellent* class samples is particularly relevant. This is because the QDA components do not form a single space in which to represent the data, but rather we have unified four different spaces into one and the classifier is not able to discern which one offers advantages for which category. That is, there is a space among the four where the samples of a certain class are much better separated from the remaining classes but the contribution to the classifier kNN distances of each of the spaces will be similar, since the classifier will not know a priori in which of the four spaces the sample is better classified.

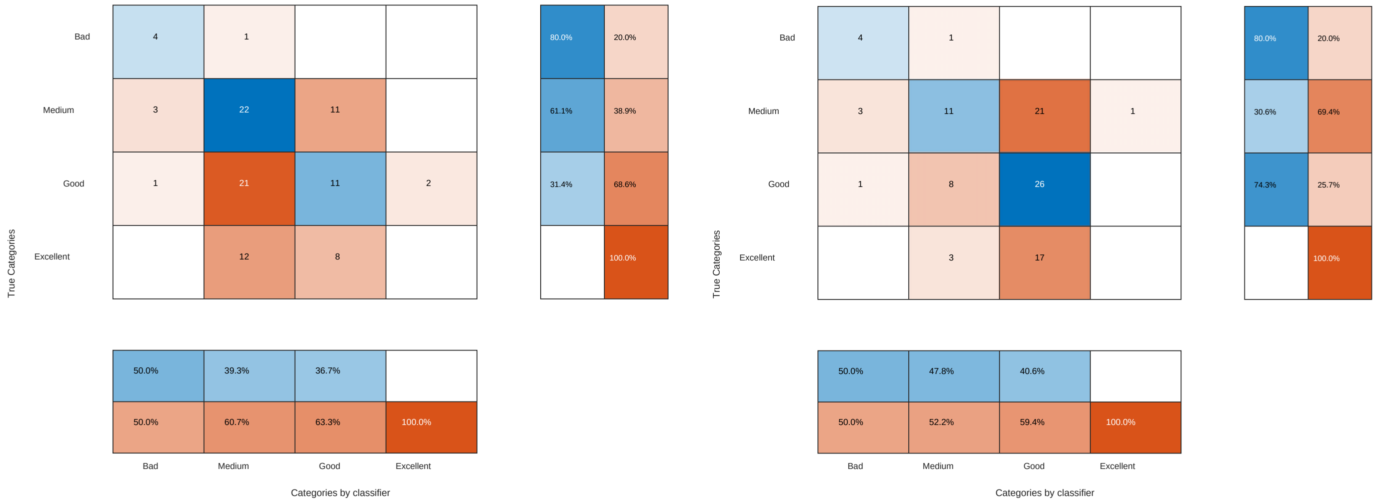


Figure 78. Confusion matrix of LOO validated kNN classifier with Euclidean distance on the first dataset expressed by QDA features. 2 neighbors and standard losses on the left. 9 neighbors and proposed losses on the right.

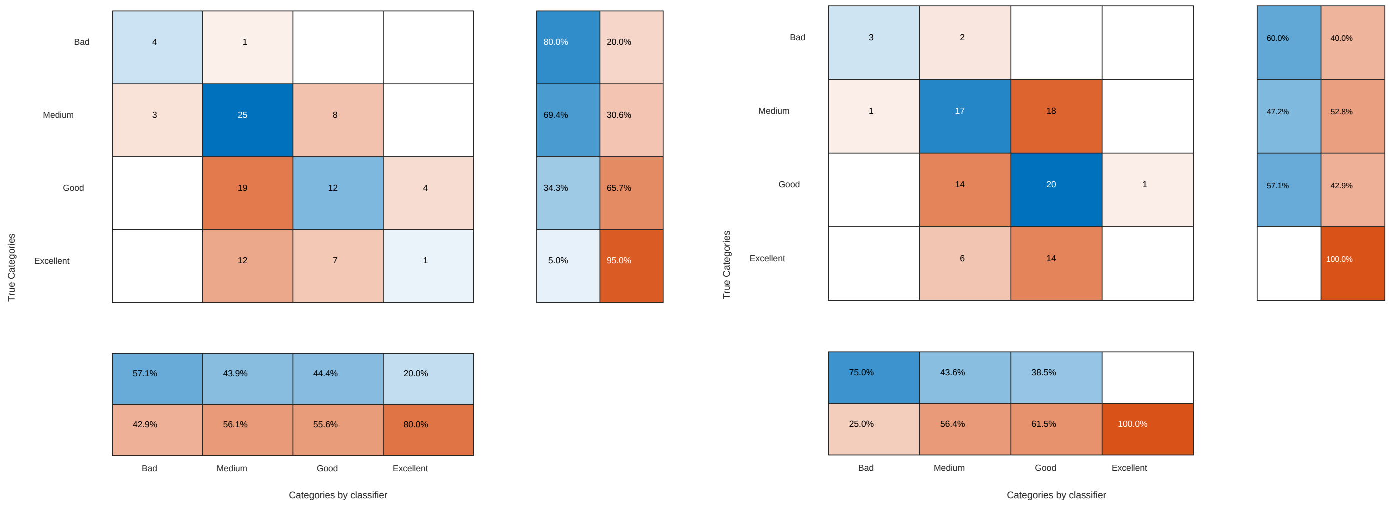


Figure 79. Confusion matrix of LOO validated kNN classifier with Manhattan distance on the first dataset expressed by QDA features. 2 neighbors and standard losses on the left. 6 neighbors and proposed losses on the right.

The results of applying kNN to the second dataset are shown in Table 58 . The table reflects the features in which the data were expressed, the optimal number of neighbors, the metric used and the global accuracies for the standard losses and proposed losses. The results are similar to those of the previous taster. When expressing the data in the full original features, using only the anthocyanin derivatives, those selected as LDA$_2$, those *selected QDA* or PCA with different amounts of principal components the results are poor.

			Standard Losses		Proposed Losses	
Features		Distance	Optimum number of neighbors	Accuracy	Optimum number of neighbors	Accuracy
Originals		Euclidean	18	35.40	13	40.63
		Manhattan	13	37.50	13	37.50
Anthocyanin derivatives		Euclidean	4	43.75	15	39.06
		Manhattan	15	45.31	15	40.63
LDA$_2$ Selection		Euclidean	3	43.75	6	37.50
		Manhattan	2	48.44	2	48.44
QDA Selection		Euclidean	11	43.75	18	37.50
		Manhattan	11	42.19	11	40.63
PCA	99%	Euclidean	2	43.75	16	37,50
		Manhattan	1	45.31	1	45,31
	95%	Euclidean	2	43.75	2	43,75
		Manhattan	2	45.31	2	45,31
	90%	Euclidean	2	45.31	2	45,31
		Manhattan	2	46.88	2	46,88
LDA		Euclidean	5	96.88	9	96.88
		Manhattan	9	96.88	11	96.88
QDA		Euclidean	2	46.88	9	43.75
		Manhattan	9	60.94	8	59.38

Table 58. Design parameters and overall results of kNN classifiers on the second dataset.

The results when representing the features in the independent components of each Fischer's class are excellent with over 95%. The use of different metrics, although providing slight improvements in the classifiers, shows very similar results with differences lower than the margin of error that arises when validating the data. As can be seen from the confusion matrices in Figure 80, which coincide for both types of metrics with the optimal number of neighbors, the proposed losses classifier also provides correct recommendations 100% of the time, losing only one originally *Excellent* case in the process.

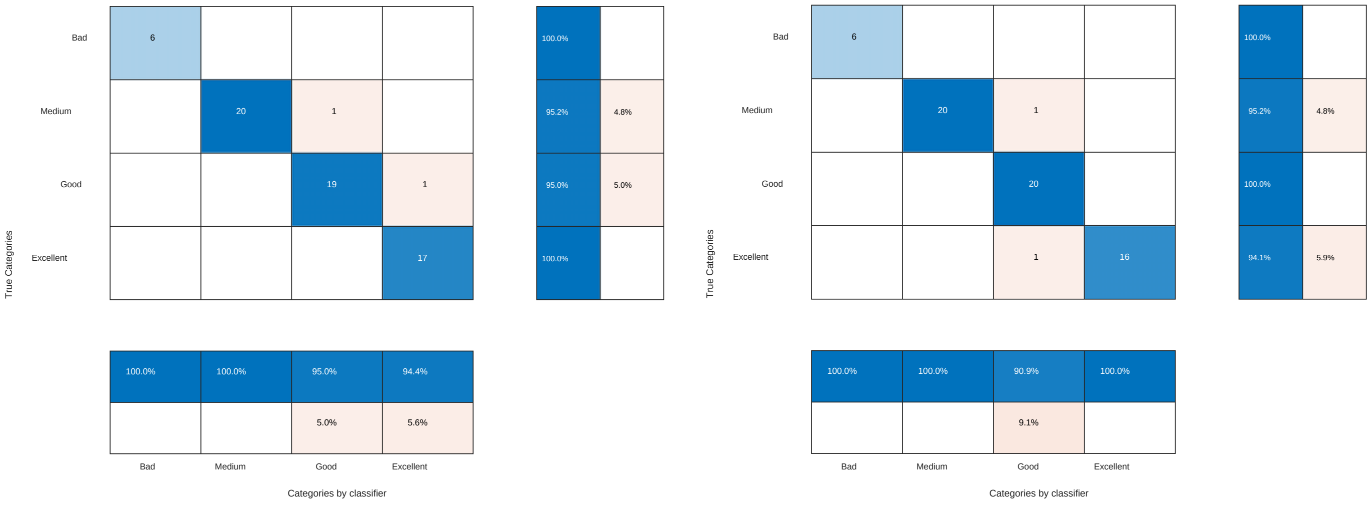


Figure 80. Confusion matrix of the LOO validated kNN classifier with optimum number of with standard losses (left) and proposed losses (right) on the second dataset.

When using the dependent components of each class the results improve on those of the counterpart classifiers of the first dataset but with accuracies of no more than 60%. The confusion matrices of the classifiers with Manhattan metrics can be seen in Figure 81.

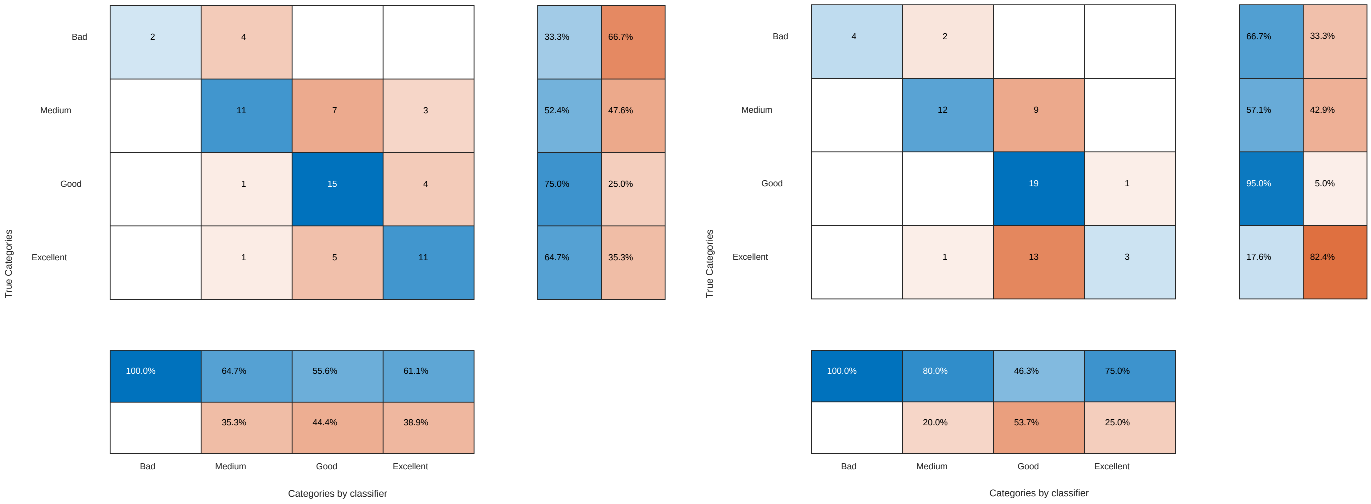


Figure 81. Confusion matrix of LOO validated kNN classifier with Manhattan distance on the second dataset expressed by QDA features. 9 neighbors and standard losses (left). 8 neighbors and proposed losses (right).

7.6 CART

We will apply the CART algorithm following the methodology described in section 2.5 for the datasets of each taster separately, first with the standard loss matrix and then with an adaptation of the loss matrix of Table 38. In order to apply the proposed matrix we must make its maximum value 1 or the estimated error of R_{cv}, the value of $SE(R_{cv})$, will result in a complex number. As a solution we will adapt the loss matrix by dividing it by 4 (the maximum value present in the matrix) and thus keeping the same proportions.

The first step to apply the algorithm is the separation of the dataset into three subsets: testing (for final validation of the classifier using LOO), validation of design parameters, and training.

LOO requires that at each iteration we create an independent classifier by excluding a different sample for testing that classifier. Thus, in the case of the first taster we will perform 96 iterations selecting a different sample each time for testing and leaving the remaining 95 for validation and training. In the case of the second taster, 64 iterations will be performed, also selecting a different sample each time for testing and leaving the remaining 63 for validation and training.

To choose the validation set and to be able to choose the tree, we have proceeded in two different ways. The first one was to take 2 samples for category (a total of 8 samples) and reserving them for validation. The second one, more in accordance with the specifications of Breiman et al. [1] was to take a number of cases close to the proportionality of the categories present in the data. Table 59 shows the original proportions in the data and the number of cases extracted in each case for validation that approximate these same proportions. The cases chosen for validation were taken at random from those remaining after extracting the test case, i.e. from the 95 examples (in the case of the first taster) and from the 63 examples respectively (in the case of the second taster).

		No. of *Bad* category cases	No. of *Medium* category cases	No. of *Good* category cases	No. of *Excellent* category cases
First taster	Total cases	5	36	35	20
	Validation cases	1	7	7	4
Second taster	Total cases	6	21	20	17
	Validation cases	1	4	4	3

Table 59. Number of cases per category for training and validation set.

For the growth phase of the trees it is necessary to select which impurity function $i(t)$ to use. We have tested both the Gini diversity index and the *towing rule*.

For all the proposed variants, we have preferred to let the tree grow to the last necessary levels because the cost of computing the CART training is not high (the training takes less than a minute) and thus we ensure that in the pruning we will obtain the optimal tree in each case. With this objective in mind we chose $\beta = 1000$, $N_{min} = 1$ and $k_{max} = 200$.

Once the tree has grown we will proceed pruning as indicated in section 2.5 selecting the pruning that produces a tree with the lowest value of $R_{cv}(T_k) + SE$. In case of equal values, the smaller tree will be chosen.

k (No. of iteration)	$\lvert T_k \rvert$	R_{cv}	SE
1	19	0.11	0.08
2	13	0.11	0.08
3**	9	0.11	0.08
4	7	0.16	0.08
5	5	0.21	0.07
6	3	0.32	0.07
7	1	0.63	0.07

Table 60. Data collected during the pruning of a tree (iteration 25) on the data from the first taster.

For example, for the specific case of the first taster data expressed in the independent components of each Fischer's class with the binarism rule, standard loss matrix, proportional validation subset, and iteration 25 (excluding sample 25 for LOO), the nested prunings yielded the values of the Table 60 for $k, |T_k|, R_{cv}$ y SE. Some of these values have also been plotted in the Figure 83.

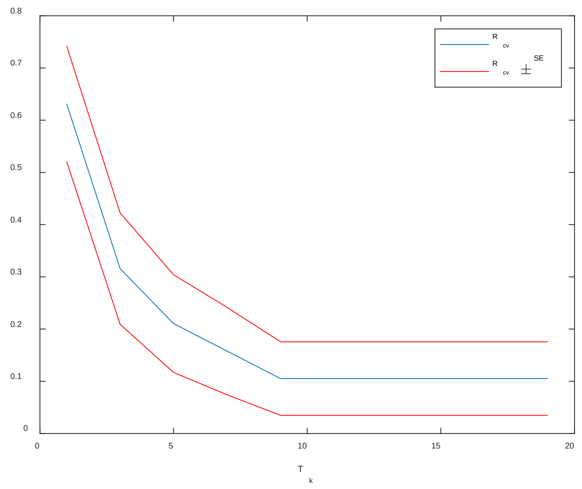


Figure 82. R_{cv} and SE values during pruning of a tree (iteration 25) on the first taster data.

Consequently, the tree chosen is the one with 9 nodes, the one generated in the 3rd iteration. Of the 9 nodes present, 4 correspond to decision nodes and 5 to final nodes. The structure of this decision tree is shown in Figure 83.

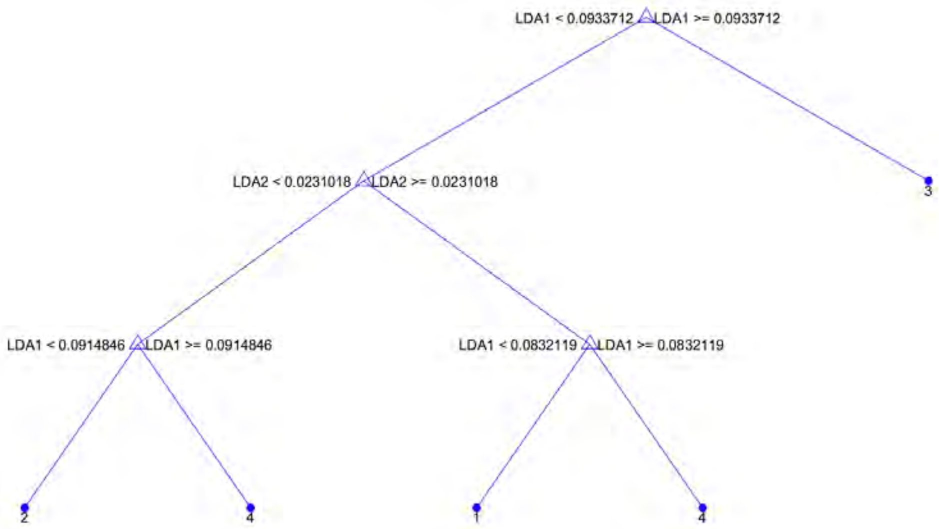


Figure 83. Decision tree generated during the pruning of iteration 25 on the data of the first taster. At the final nodes the categories represented are: $Bad = 1, Medium = 2, Good = 3, Excellent = 4$.

The classifiers obtained for the first taster's dataset are firstly measured using the overall accuracy parameter, the results of which are shown in percentages in Table 61.

Features	Standard losses				Proposed losses			
	Gini		Binarism Rule		Gini		Binarism Rule	
	Constant validation	Proportional validation	Constant validation	Proportional validation	Constant validation	Proportional validation	Constant validation	Proportional validation
Originals	8.33	23.96	9.38	33.33	47.92	36.46	32.29	36.46
Anthocyanin derivatives	5.21	37.50	5.21	37.50	28.13	36.46	33.33	36.46
LDA_1 Selection	17.71	32.29	0.00	33.33	32.29	34.38	28.13	36.46
QDA Selection	27.08	25.00	34.38	43.75	23.96	36.46	41.67	33.33

PCA	100	41.67	33.33	41.67	31.25	21.88	35.42	33.33	35,42
	99	27.08	38.54	26.04	45.83	30.21	34.38	28.13	39,58
	95	23.96	37.50	17.71	41.67	37.50	22.92	31.25	20,83
	90	1.042	44.79	0.00	32.29	38.54	31.25	39.58	40,63
LDA		84.38	83.33	84.38	87.50	84.38	84.38	81.25	87.50
QDA		71.88	66.67	68.75	69.79	69.79	65.63	72.92	66.67

Table 61. Accuracy of CARTs in % for first taster data validated by LOO according to impurity function, choice of samples for cross validation in the optimization process, features to describe the samples and loss matrix used.

As can be seen most of the classifiers working with the original features or with the selected feature sets (over the originals) produce very poor results with accuracies below 40%. Although working in the PCA feature space involves some improvement (especially if the binarism rule and proportional validation are used), their results are still of little practical value, with accuracies always below 45%.

If we compare the results by varying the impurity function we see that both solutions tend to achieve similar percentages (usually with differences of less than 5%) and sometimes even produce the same trees. In cases where the difference is greater, there is no clear advantage in favor of either of the possibilities.

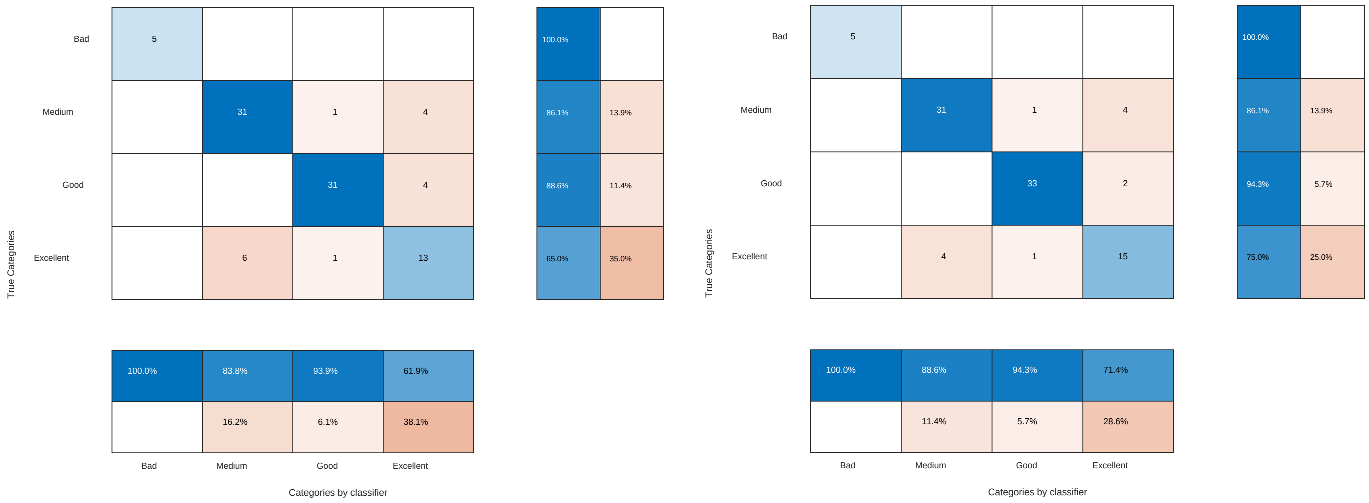


Figure 84. Confusion matrix of the LOO validated CART on the first dataset's LDA components, standard losses, with proportional validation [1 7 7 4], using the Gini index (left) and using the binarism rule (right).

The inclusion of the non-standard loss matrix, although it can cause variations of up to 4 percentage points, does not seem to have much influence on the overall results. However, introducing it does force fewer errors to occur in key positive or negative predictions.

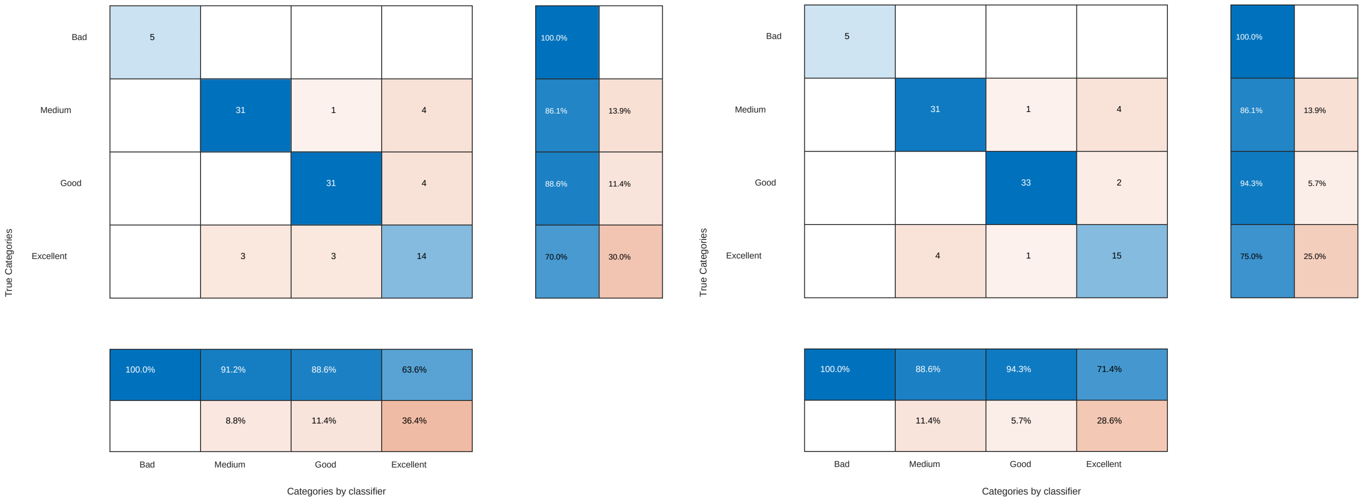


Figure 85. Confusion matrix of the LOO validated CART, on the first dataset in LDA components, with proposed losses, proportional validation [1 7 7 4] and Gini index (left) and binarism rule (right).

The proportions in the validation samples may show significant differences in some cases but both are consistent within a range of about 10 percentage points difference. Its effect is similar to the introduction of the loss matrix, since by taking the proportions constant the error rate of the classifier is measured on a population where the *Bad* and *Excellent* categories are overrepresented, giving a higher importance to their best classification.

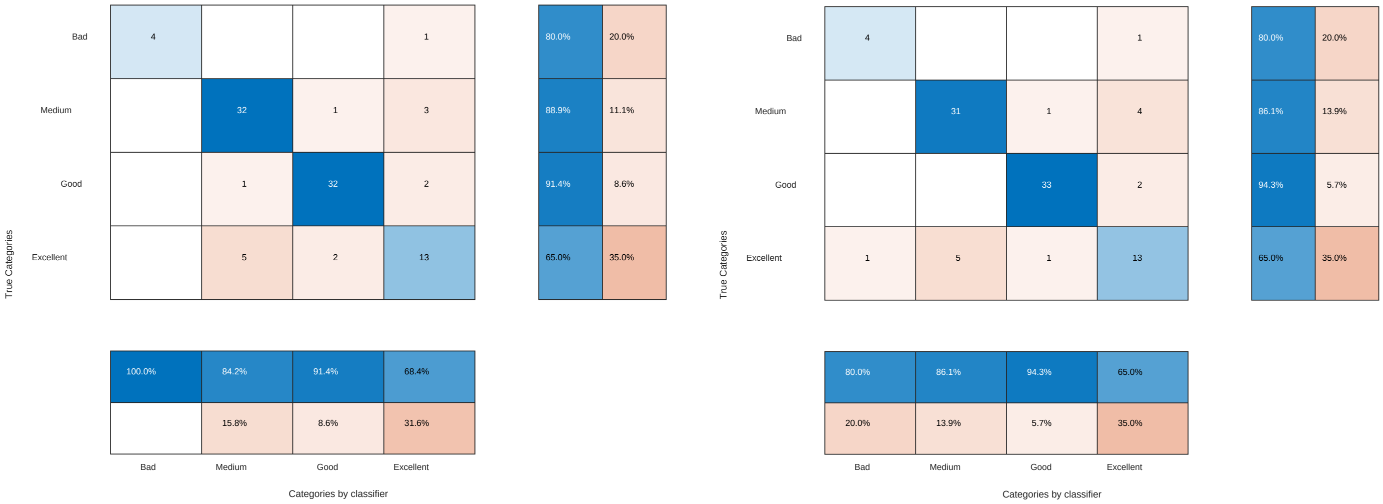


Figure 86. Confusion matrix of LOO validated CART, on the first dataset in LDA components, constant validation [2 2 2 2] with standard losses and Gini index (left) and with proposed losses and binarism rule (right).

When working in the space defined by the class-dependent Fischer's method (the component space labeled QDA), the classifier is able to correctly classify 65-70% of the results. Finally, the best CART classifiers on the ensemble of the first taster are achieved by representing the data in the class-independent Fischer features (LDA space) where the classifiers achieve accuracies of 83-87%.

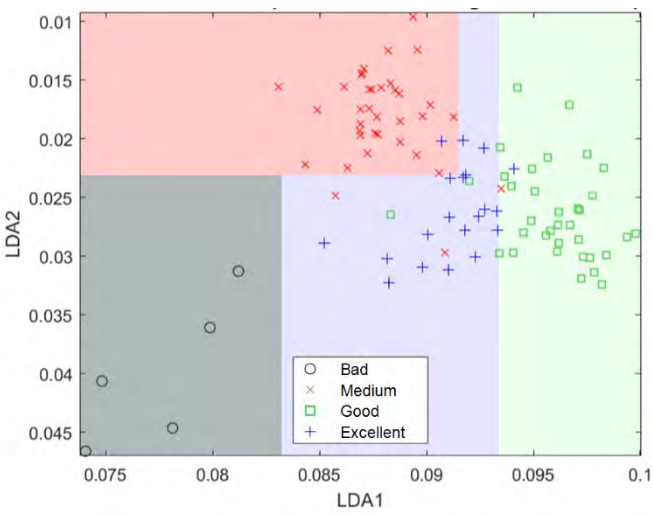


Figure 87. Decision regions generated during pruning of a tree (iteration 25) on the first taster data.

Moreover, such classifiers produce trees with few splits (no more than 7) and can be translated to simple decisions on the original features. Continuing with the previous example, the decision tree generated in iteration 25 is the one depicted in Figure 83. In this tree, only 4 decisions are made based on only the first two Fischer components that generate the splitting regions represented in the Figure 87.

Classified category	Condition
Excellent	$LDA_1 \geq 0{,}0832119$ and $LDA_2 \geq 0{,}0231018$ or $LDA_1 \geq 0{,}0914846$ and $LDA_2 < 0{,}0231018$
Good	$LDA_1 \geq 0{,}0933712$
Medium	$LDA_1 < 0{,}0914846$ and $LDA_2 < 0{,}0231018$
Bad	$LDA_1 < 0{,}0832119$ and $LDA_2 \geq 0{,}0231018$

Table 62. Deciding conditions for each region as a function of Fischer's components for the decision tree of iteration 25 of the first taster.

These decisions are translatable into equations about the original features that are easily interpretable. Thus, for example, the four divisions above would provide the conditions summarized in Table 62, and using the weights with which we constructed each component in section 7.2 where

$$\begin{aligned} LDA_1 \approx\ & 0{,}846 \cdot \text{Catechin-Mv-3-(6-p-coum)-glc} - 0{,}318 \cdot \text{Mv-3-(6-p-coum)-glc-4-vinylguaiacol} \\ & - 0{,}25 \cdot \text{(Epi)Galocatequin-Mv-3-(6-p-coum)-glc} - 0{,}224 \cdot \text{(Epi)Galocatequin-Mv-3-glc} \\ & + 0{,}177 \cdot \text{Mv-3-(6-p-coum)-glc-8-ethyl-(epi)catechin 1} - 0{,}117 \cdot \text{Mv-3-(6-p-coum)-glc cis} \end{aligned} \tag{109}$$

$$\begin{aligned} LDA_2 \approx\ & 0{,}318 \cdot \text{Catechin-Mv-3-(6-p-coum)-glc} + 0{,}682 \cdot \text{(Epi)Galocatequin-Mv-3-(6-p-coum)-glc} \\ & - 0{,}123 \cdot \text{Mv-3-(6-p-coum)-glc-4-vinylguaiacol} + 0{,}303 \cdot \text{(Epi)Gallocatechin-Mv-3-glc} \\ & - 0{,}378 \cdot \text{Mv-3-(6-p-coum)-glc cis} - 0{,}09 \cdot \text{Mv-3-(6-p-coum)-glc-8-ethyl-(epi)catechin 1} \end{aligned} \tag{110}$$

would give us the simplified conditions based on the original features for simpler models.

To explain why the selection of the original anthocyanin derivatives does not produce an efficient classifier when they are used to train the CART algorithm (its best accuracy is 37.5%) while the above equations evidence the possibility of establishing a simple classifier using these anthocyanin derivatives, we must take into account that the CART algorithm splits are always performed on a single variable. In situations where the drawing of class boundaries fits better with linear combinations of the variables the trees may give poor results [1].

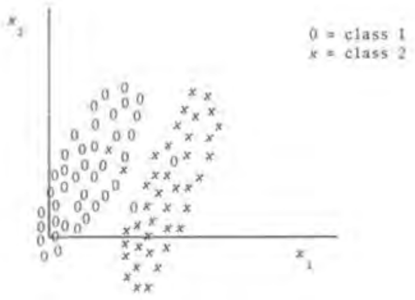


Figure 88. Optimal data structure for linear combinations of variables rather than for divisions over one variable. [1]

In problems where such a structure can be suspected, the possible questions can be extended to include linear combinations of the original variables. Thus, we would also admit questions of the type “is it $\sum_m a_m x_m < c$?” at the cost of complicating the interpretability of the final tree. The decomposition in Fischer’s space fulfills this role of proposing optimal linear combinations for subsequent classification, achieving better results with the same algorithm.

The data from the second taster produce similar results. Again, the classifiers based on the original features produce very poor results (not exceeding 36% accuracy) and the classifiers based on the LDA and QDA features stand out.

Features		Estándar losses				Proposed losses			
		Gini		Binarism Rule		Gini		Binarism Rule	
		Constant validation	Proportional validation	Constant validation	Proportional validation	Constant validation	Proportional validation	Constant validation	Proportional validation
Originals		34.38	26.56	32.81	28.13	28.13	46.79	32.81	31.25
Anthocyanin derivatives		31.25	18.75	32.81	25.00	32.81	28.13	32.81	31.25
LDA_2 Selection		29.69	28.13	32.81	25.00	31.25	31.25	32.81	35.94
QDA Selection		31.25	23.44	31.25	21.88	31.25	29.13	31.25	31.25
PCA	100	10.94	21.88	31.25	23.44	20.31	25.00	32.81	26,56
	99	23.44	28.13	31.25	6.25	14.06	29.69	25.00	28,13
	95	21.83	21.88	26.56	4.688	20.31	29.69	26.56	28,13
	90	31.25	32.81	35.94	26.56	21.88	34.38	15.62	29,69
LDA		93.75	85.94	82.81	79.69	90.63	84.38	81.2	87.50
QDA		50.00	45.31	51.56	48.44	59.38	65.63	53.12	68.75

Table 63. Accuracy of CARTs in % for second taster data validated by LOO according to impurity function, choice of samples for cross-validation in the optimization process, features to describe the samples and loss matrix used.

In this second case all LDA-based CARTs produce very similar classification regions. As shown in Figure 89 the decision regions for a CART based only on two principal components with the standard losses Gini index and with the proposed losses binarism rule (both with proportional validations).

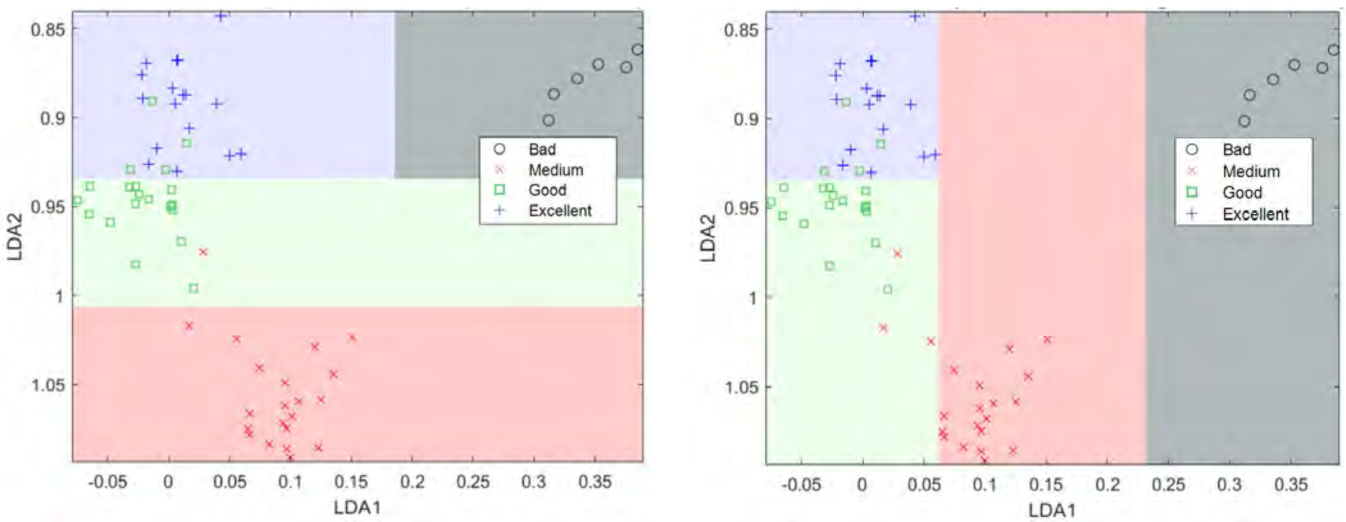
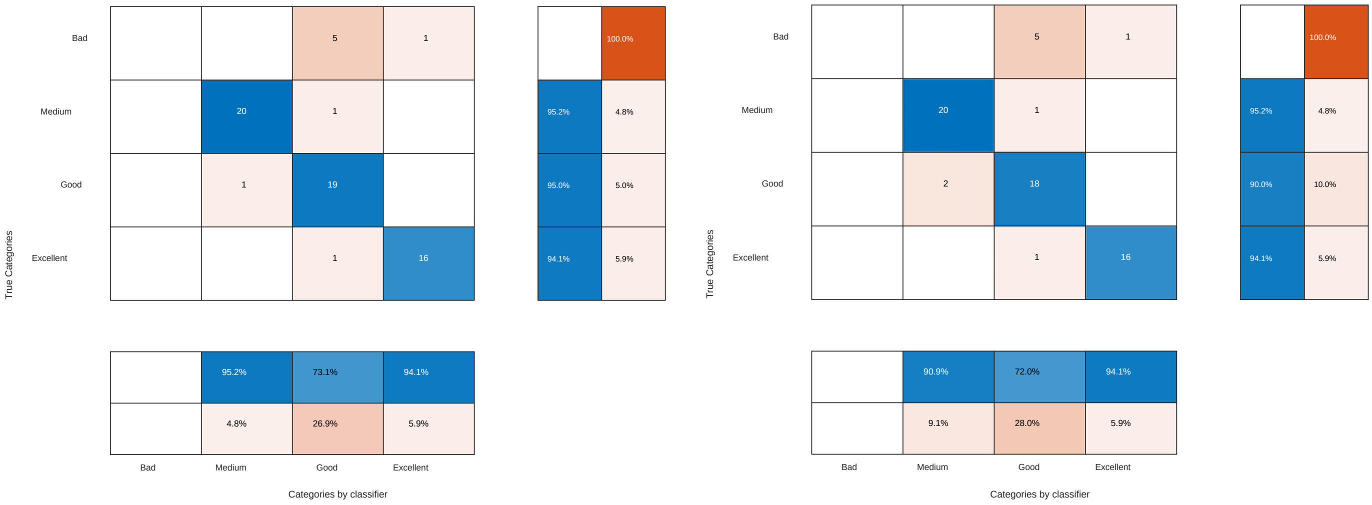


Figure 89. Classification regions for CART based on two independent features of Fischer space: with the Gini index and the standard loss matrix (on the left) and with the binarism rule and the proposed loss matrix (on the right).

In this case the introduction of non-proportional validation acts by giving much more weight to the *Bad* and *Excellent* category examples, helping to better positive and negative predictions. If the samples for validation are proportional, more cases of the intermediate categories are used and this causes the classifier to have a tendency to classify more samples in these categories, not classifying any as *Bad*. On the other hand, in the number of constant samples per category, the samples that were previously in the minority acquire a greater value, notably improving the negative predictions.



True Categories	Bad	Medium	Good	Excellent		
Bad			5	1		100.0%
Medium		20	1		95.2%	4.8%
Good		1	19		95.0%	5.0%
Excellent			1	16	94.1%	5.9%
		95.2%	73.1%	94.1%		
		4.8%	26.9%	5.9%		

True Categories	Bad	Medium	Good	Excellent		
Bad			5	1		100.0%
Medium		20	1		95.2%	4.8%
Good		2	18		90.0%	10.0%
Excellent			1	16	94.1%	5.9%
		90.9%	72.0%	94.1%		
		9.1%	28.0%	5.9%		

Figure 90. Confusion matrices for CART based on two LDA features with proportional validation [1 4 4 3]: with the Gini index and the standard loss matrix (on the left) and with the modified loss matrix (on the right).

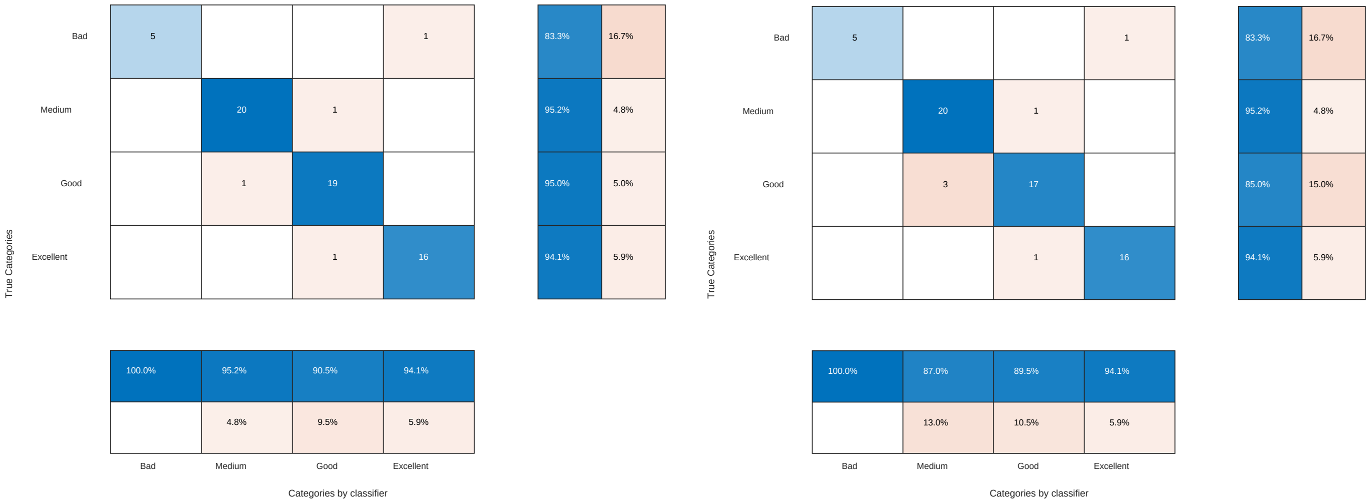


Figure 91. Confusion matrices for CART based on two LDA features with constant validation [2 2 2 2]: with the Gini index and the standard loss matrix (on the left) and with the modified loss matrix (on the right).

7.7 MLP

Next, we will apply the algorithm described in section 2.6 to the datasets of each taster separately to obtain MLP classifiers. The first decision we must make concerns the choice of network architecture. We could define a network with a final layer based on sigmoidal activation functions and interpret the outputs of each neuron as a certain probability that the example presented to the neural network should be so classified, but as this is a classification problem using a Softmax layer as the output layer offers advantages when training the network, since the network will try to increase the probability of one of the outputs by decreasing the others and the interpretation of the outputs as classification probabilities will be mathematically more faithful by summing all 1. As already indicated in section 2.6 the gradient of the neurons of this layer will depend on the internal states of the other neurons of the same layer, so the algorithm needs a small adaptation which consists in modifying the cost function by the cross-entropy. Thus, we will substitute the equation (90) by the following equation:

$$J = -\sum_{k=1}^{N(L)} y_{realk} \ln z_k \qquad (111)$$

Here the component y_{realk} is the desired value of the network for each of the outputs. Since these are networks that classify the samples into 4 different categories, the network will consist of a last layer of 4 neurons each one related to the probability that the sample is of a category. When an example is presented, the desired output will consist of a probability of 1 in that neuron related to the true class of the example and a probability of 0 in the others. That is to say, the desired output for a data of the class *Bad* (class associated with the output neuron 1) would be $\boldsymbol{y_{real}} = (1, 0, 0, 0)$, the output for a data of the class *Medium* would be $\boldsymbol{y_{real}} = (0, 1, 0, 0)$ and so on. Thus, the desired output vectors would have $y_{realk} = 1$ for the value k coinciding with its real class and the rest of components $y_{realk} = 0$. Thanks to this presence of null elements in the vector $\boldsymbol{y_{real}}$ the cost value can be simplified to $J = -\ln z_{k=real\ class}$ which, as it is evident, depends only on the output neuron $z_{k=real\ class}$. This situation offers advantages since we can follow a reasoning analogous to the one followed in section 2.6 with a single modification for the last layer L. In this case the chain of derivations for the weights of the last layer will be $J \rightarrow z^L_{k=real\ class} \rightarrow a^L_j \rightarrow w^L_{ji}$ and the equation (65) will be

$$\frac{\partial J}{\partial w_{ji}^L}=\frac{\partial J}{\partial z_{k=real\ class}^L}\frac{\partial z_{k=real\ class}^L}{\partial a_j^L}\frac{\partial a_j^L}{\partial w_{ji}^L}=\delta_j^L\frac{\partial a_j^L}{\partial w_{ji}^L} \tag{112}$$

but in this case to calculate δ_j^L we will have two possibilities for $\frac{\partial z_k^L}{\partial a_j^L}$ (using the derivatives of Table 1). If a_j is of the neuron associated with the real class k then

$$\delta_{j=k=real\ class}^L=\frac{\partial J}{\partial z_k^L}\frac{\partial z_k^L}{\partial a_{j=k}^L}=-\frac{1}{z_k}z_k(1-z_k)=z_k-1 \tag{113}$$

and if it is not

$$\delta_{j\neq k=real\ class}^L=\frac{\partial J}{\partial z_k^L}\frac{\partial z_k^L}{\partial a_{j\neq k}^L}=-\frac{1}{z_k}\left(-z_j z_k\right)=z_j \tag{114}$$

For the rest of the terms and layers, the reasoning followed in the backpropagation algorithm is still fully valid, being able to use the equations from (69) to (88)[46].

The final modification of the algorithm would affect the Step 2, whose equation (90) would be replaced by the new error function[47]

$$J=-\sum_{k=1}^{N(L)}y_{realk}\ln z_k \tag{115}$$

affecting the last part of equations (102) and (103) of Step 7 which would be

$$J_{training}=\sum_{r=1}^{M}J\left(\boldsymbol{z}_{\boldsymbol{training}}^{(r)}\right)=-\sum_{r=1}^{M}\sum_{k=1}^{N(L)}y_{realk}\ln z_k \tag{116}$$

$$J_{validation}=\sum_{r=1}^{nvalidation}J\left(\boldsymbol{z}_{\boldsymbol{validation}}^{(r)}\right)=-\sum_{r=1}^{nvalidation}\sum_{k=1}^{N(L)}y_{realk}\ln z_k \tag{117}$$

and modifying the way of calculating δ_j^L in equation (95) from Step 5.b by[48]

$$\delta_j^L=z_j-y_{realj} \tag{118}$$

The algorithm requires, just as it happened with CARTs, to reserve a part of the samples for training and others for validation, in addition to the ones we need for the final testing of the classifiers by LOO. So we will act in exactly the same way as in section 7.6. That is, we will perform as many iterations as number of samples we have, extracting a different sample for the final test (LOO) of the classifier in each iteration. On the remaining samples we will separate a subset of samples for training and a subset for validation during training, always randomly, but respecting certain proportions for each category. The validation samples during training will be to decide when to end the training used to by comparing the evolution of the values of (115) and (116). The number of samples per category taken for these validation subsets varies according to the

[46] To be strict the terms a_k^L in the equations (68) and (74) should be replaced by the list of all the a_j^L but this does not influence the development of the subsequent chains which again depend on the same variables.

[47] The equation (90) would also become worthless.

[48] Since $y_{realj}=1$ when j is that of the true class of the sample, and $y_{realj}=0$ when j is not that of the true class, the equations (112) and (113) can be summarized in a single equation.

taster. As we wish to maintain proportions similar to those of the total dataset we will use the samples per class in Table 59 already used with the CARTs.

On the other hand, before training each of the MLP networks we must pre-establish their architecture, understood as the number of hidden layers (the first and last layers are defined by the data), the number of neurons in each layer and their activation functions. Two families of classifiers will be trained, in the first one the activation functions of all the hidden neurons will be sigmoidal functions and in the second family all the functions will be rectified linear (relu). When training the classifiers of the second family, some iterations produced a very slow progress in the gradient so it was necessary to normalize the data following the procedure explained in the Step 1 of section 2.2 to facilitate a better evolution of the algorithm. The number of hidden layers was increased starting from a single layer and not continuing beyond 3 layers as no significant improvements in the results were observed. In order to pre-select the range of values in which to search for the optimal value of the number of neurons per layer, some random training datasets were used, and by training them for a maximum number of 300 iterations it was observed whether they were able to converge to high classification rates of the training examples themselves or not. With the quantities where high rates were achieved, approximate ranges of values to search for were estimated.

The initial values of the weights have been randomly initialized in a range of $[-1{,}1]$ through a uniform distribution and we have worked in training without batches, $M = 1$, since it has been possible to train each of the networks in relatively short times (less than one minute).

The values for the learning rate have been modified in each experiment depending on the results in the first steps of the algorithm, always starting from the same initial value of the weights to adjust α. Starting from a rate of $\alpha = 0{,}1$ if, on one hand, the algorithm made little progress in reducing the error of the training set between iteration and iteration, the rate was increased by 100%. If, on the other hand, there were many oscillations between iterations (instead of a continuous decrease), a reduction of 50% was proposed until a suitable value was found.

In order to decide when to discontinue the training algorithm, the error in the validation data, $J_{validation}$, was taken into account. If after 10 iterations the error was higher than a previous minimum, the training was terminated.

As an example, the procedure followed to build the MLP classifier for the last LOO iteration on the first taster's data expressed in the first two Fischer's components, for the specific case of sigmodial activation functions and an architecture of 2 neurons in the first input layer, 60 neurons in the second, 40 neurons in the third and 4 neurons in the last output layer. In this last LOO iteration, of the 96 samples the last one was reserved for final validation. Of the remaining samples, 1 from the *Bad category*, 7 from the *Medium category*, 7 from the *Good category* and 4 from the *Excellent category* were extracted for validation during training, leaving the remaining 76 for training. We started from a value of $\alpha = 0{,}1$ and when training the network it was observed that the training cost was decreasing rapidly in the first iterations, so it was taken as a good value. In addition, after about 60 iterations, it was observed that the cost of $J_{validation}$ instead of continuing to decrease, it started to increase (probably a symptom of overtraining [34]) so the training was stopped and the weights of the minimum cost iteration, in this case iteration number 61, were chosen as the final values.

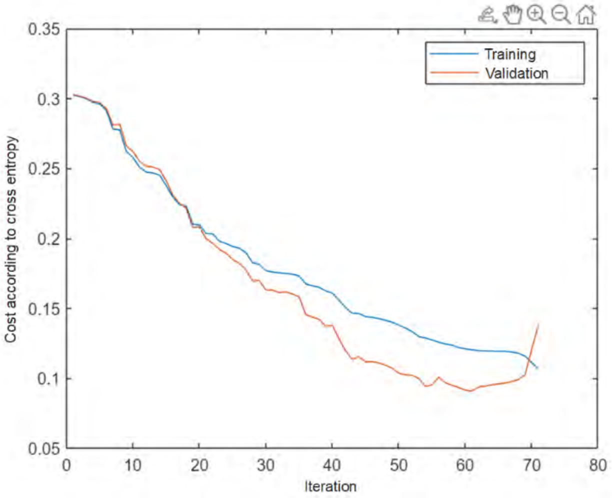


Figure 92. Evolution of $J_{training}$ y $J_{validation}$ when constructing the MLP classifier on the first two Fischer's components taken as an example.

The decision regions of the resulting classifier can be seen in Figure 93 (left), where thanks to the 60 neurons of the first hidden layer, the soft edges of the different regions can be constructed and joined in a second layer through the remaining 40 neurons. When comparing the regions with similarly constructed classifiers of a smaller number of neurons per layer (right part of Figure 93), it is observed that in the latter the classification regions are simpler, less smooth, less complex.

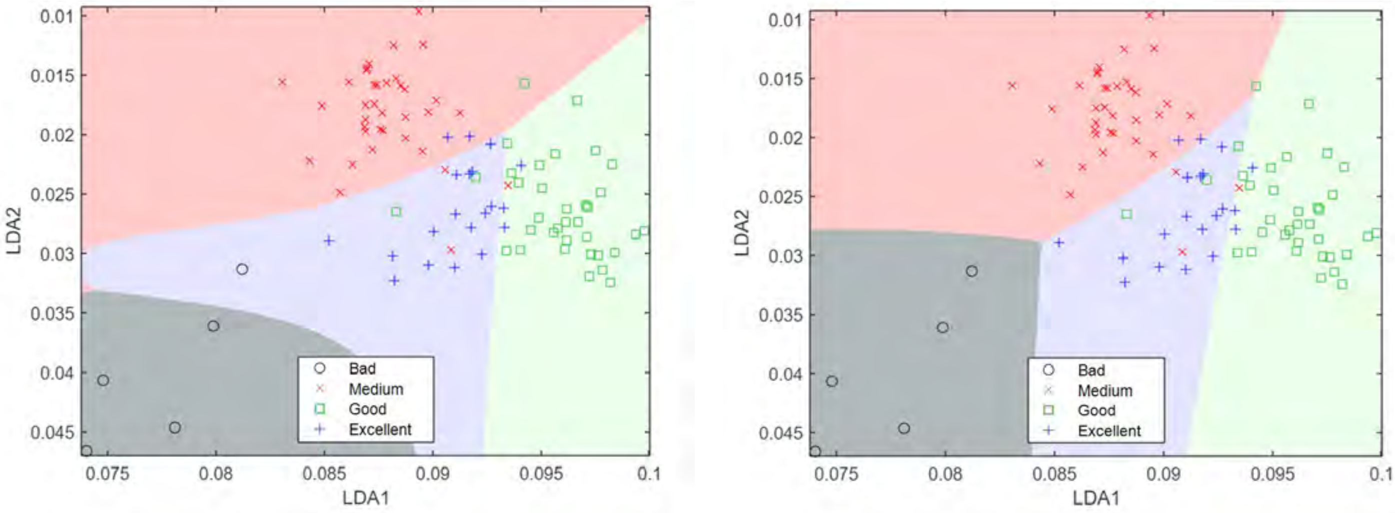


Figure 93. Classification regions for the MLP classifier on the first two Fischer's components taken as an example (left) and the classification regions for a similar MLP classifier with 6 neurons in the first hidden layer and 6 in the second (right).

The results of the MLP classifiers based on sigmoidal functions obtained for the first taster's dataset, measured by the overall accuracy (in percentage), can be found in Table 64. Only the results of the classifiers applied in the space of the independent Fischer's components of each class stand out as acceptable with accuracies around 88% in the best case. It can also be seen from the data collected in the table that with a single hidden layer and a relatively high number of neurons, good classifier results are already achieved. By increasing the number of layers to 2 hidden layers the results of these classifiers can be slightly improved, but not significantly and increasing the number of hidden layers to 3 achieves worse results than with the previous architectures, as well as entailing a longer training time.

	1 hidden layer	2 hidden layers	3 hidden layers

		Neurons in layer...	Accuracy (%)	Neurons in layer...		Accuracy (%)	Neurons in layer...			Accuracy (%)
		1		1	2		1	2	3	
Originals		68	42.71	32	4	39.58	5	10	5	34.38
Anthocyanin derivatives		66	35.46	24	8	30.21	8	13	15	30.21
LDA$_1$ Selection		13	43.75	35	7	44.79	6	10	10	44.79
QDA Selection		34	42.71	20	6	38.54	5	15	5	38.54
PCA	100	82	43.75	33	8	40.63	13	9	20	35,41
	99	27	44.79	35	26	40.63	7	20	13	36,46
	95	90	45.83	37	12	39.58	20	8	19	34,38
	90	45	45.84	35	42	41.67	13	5	10	36,46
LDA		49	87.50	54	51	87.50	12	8	20	83.33
QDA		6	45.83	15	16	47.92	7	9	20	38.54

Table 64. Best overall results and architecture for LOO validated MLP classifiers with sigmoidal activation functions on the first taster data.

If we compare the results with the family of MLP classifiers with relu activation functions on the same datasets (see Table 65), we can observe that the relu classifiers show better results (over 90%) with two hidden layers than with one, and that is taking into account that with a single layer they already improve the results of the first family of classifiers. Unlike the sigmoidal activation functions, it is observed that relu classifiers may require some additional layer of neurons, since sometimes their results with 3 hidden layers may be better than with two layers. In any case, all the results obtained with 4-layer architectures are worse than those of simpler architectures.

		1 hidden layer		2 hidden layers			3 hidden layers			
		Neurons in layer...	Accuracy (%)	Neurons in layer...		Accuracy (%)	Neurons in layer...			Accuracy (%)
		1		1	2		1	2	3	
Originals		97	44.79	2	19	48.96	12	8	16	54.17
Anthocyanin derivatives		84	41.67	51	38	48.96	14	20	10	47.92
LDA$_1$ Selection		6	43.75	30	42	48.96	13	12	7	50
QDA Selection		18	43.75	23	18	47.92	15	17	20	47.92
PCA	100	41	44.79	38	11	48.95	13	19	19	51,04
	99	96	45.83	6	47	46.88	6	6	13	48,96
	95	12	42.71	2	51	46.88	13	9	6	51,04
	90	12	43.75	14	52	51.04	12	6	15	47,92
LDA		18	89.58	46	12	91.66	9	5	16	91.66
QDA		13	56.25	44	41	61.46	12	11	11	62.5

Table 65. Best overall results and architecture for LOO validated MLP classifiers with relu activation functions on the first taster data.

The confusion matrix for the classifier with the best overall results can be seen in Figure 94. In the case of the 2- and 3-layer classifiers, 100% of the predictions it makes about the *Bad* category are correct, but only 77.3% of the positive recommendations are correct, losing 15% of the possible wines to be recommended as *Excellent*. The results of the single-layer classifier, although somewhat lower in overall accuracy, are very similar in terms of their validity for issuing negative and positive recommendations, where 100% of the predictions for the *Bad* category are correct and 80% of the predictions for the *Excellent* category are correct.

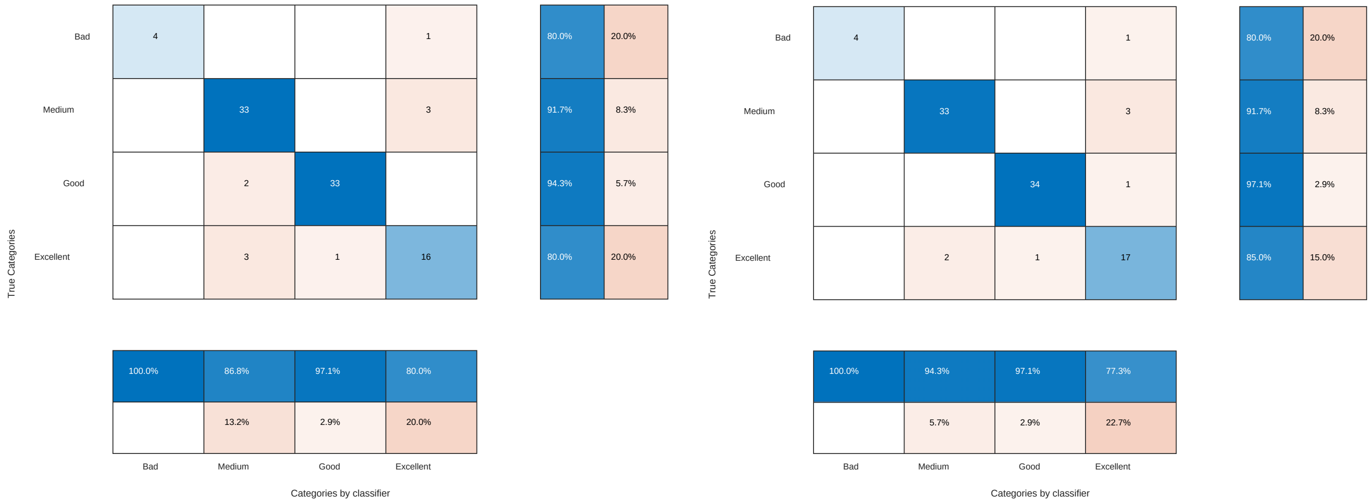


Figure 94. Confusion matrix for LOO validated MLP classifier with relu activation functions and the optimal number of neurons, with one layer architecture (left) and two and three layers architecture (right).

By applying the same technique to the data provided by the second taster, two families of classifiers were again obtained. The results of the first family, that of sigmoidal activation functions, can be found in Table 66 and those corresponding to the second family, that of relu activation functions, are shown in Table 67.

		1 hidden layer		2 hidden layers			3 hidden layers			
		Neurons in layer...	Accuracy (%)	Neurons in layer...		Accuracy (%)	Neurons in layer...			Accuracy (%)
		1		1	2		1	2	3	
Originals		56	40.63	59	7	39.06	7	12	11	28.13
Anthocyanin derivatives		74	40.63	51	20	42.19	7	9	12	40.63
LDA_2 Selection		52	45.31	39	4	45.31	15	5	11	45.31
QDA Selection		54	50	41	4	45.31	10	17	12	42.19
PCA	100	32	42.19	54	53	46.88	8	18	8	45,31
	99	94	42.19	50	43	48.44	11	10	18	48,44
	95	12	43.75	43	39	48.44	11	8	7	48,44
	90	39	48.44	6	7	45.31	12	12	20	45,31
LDA		39	93.75	9	47	93.75	18	14	20	90.63
QDA		69	67.19	60	40	59.38	19	18	13	40.63

Table 66. Best overall results and architecture for LOO validated MLP classifiers with sigmoidal activation functions on the second taster data.

Both results are consistent with those achieved with the first taster's data. Again, classifiers starting from data expressed in the independent Fischer's components of each class achieve overall accuracies above 90% in contrast to data expressed in any of the other options.

	1 hidden layer		2 hidden layers			3 hidden layers			
	Neurons in layer...	Accuracy (%)	Neurons in layer...		Accuracy (%)	Neurons in layer...			Accuracy (%)
	1		1	2		1	2	3	
Originals	7	43.75	9	43	46.88	7	5	5	46.88

Anthocyanin derivatives		42	50.00	28	24	53.13	7	16	12	51.56
LDA$_2$ Selection		40	45.31	50	32	53.13	8	14	11	48.44
QDA Selection		30	35.94	18	6	46.88	7	9	13	43.75
PCA	**100**	70	50.00	24	9	51.56	14	14	11	48,44
	99	76	53.13	35	54	53.13	18	11	15	54,69
	95	83	51.56	21	56	60.94	13	9	15	54,69
	90	33	45.31	29	35	54.69	19	17	13	48,44
LDA	**3 components**	28	98.44	38	17	98.44	9	6	8	98,44
	2 components	6	92.19	2	18	92.19	6	10	11	93,75
QDA		88	71.88	15	42	75.00	19	9	14	76.56

Table 67. Best overall results and architecture for LOO validated MLP classifiers with relu activation functions on the second taster data.

Again, the classifiers of the second family show somewhat higher accuracies than those based on sigmoidal functions. This time even classifying correctly all the samples except one and with unbeatable results in the *Bad* and *Excellent* categories, since when the classifier predicts a sample within those two categories it is never wrong and, in addition, no possible recommendation on wines in those categories is missed, i.e., no wine that in reality belonged to those categories has not been classified as such.

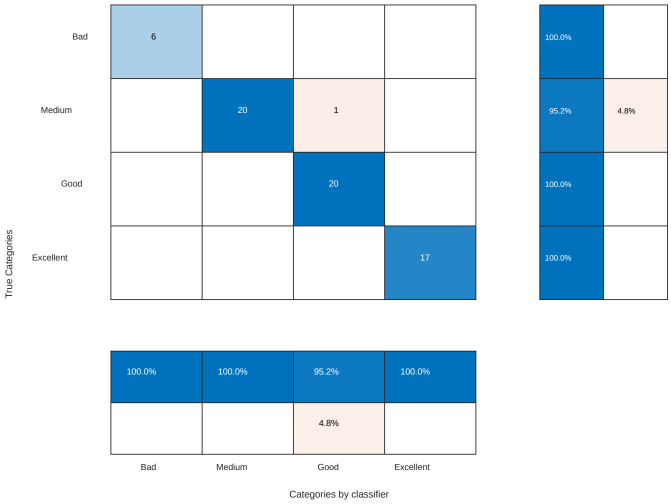


Figure 95. Confusion matrix for LOO validated MLP classifier with relu activation functions and the optimal number of neurons, with one layer architecture on the second taster data expressed in Fischer's three independent principal components (left) and in two components (right).

Classifiers based on only two independent Fischer's principal components also achieve results above 90%, even with a single layer of hidden neurons. In Figure 96 you can see the confusion matrices for the simplest single hidden layer architecture and for the best performing architecture in this case, the 3 hidden layer architecture.

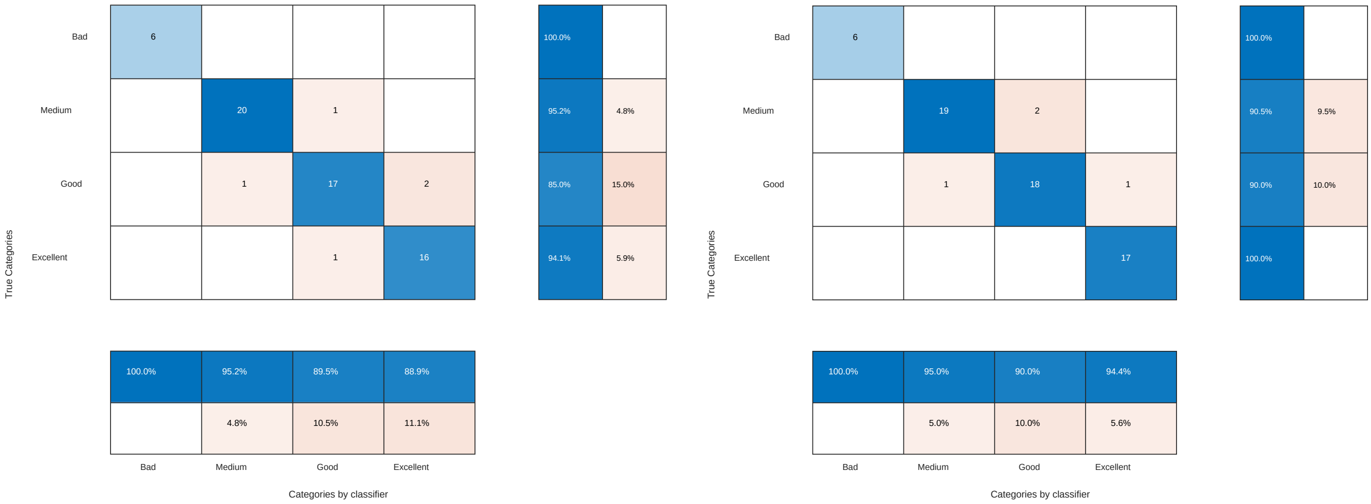


Figure 96. Confusion matrix for LOO validated MLP classifier based on two independent Fischer's components with relu activation functions and the optimal number of neurons on the second taster data. One layer architecture with 6 neurons (left) and three-layer architecture [6 10 11] (right).

In the case of the three-layer classifier, with only two components it is able to correctly classify all cases of the *Bad* category and only fails in one case of the *Excellent* category. The classification regions created have been visualized in Figure 97.

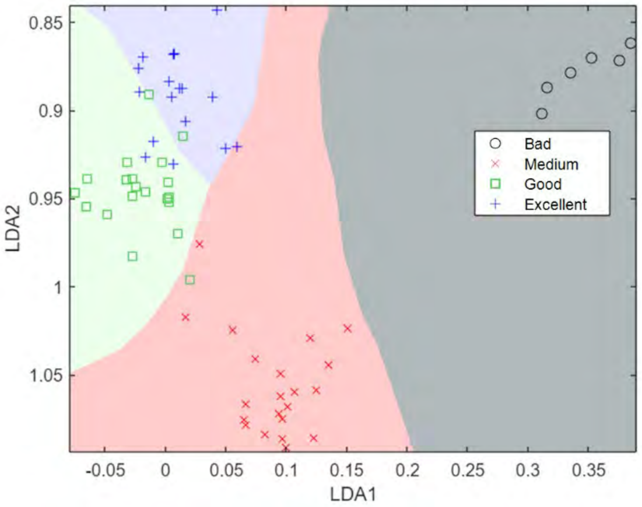


Figure 97. Classification regions for LOO validated MLP classifier based on two independent Fischer's components with relu activation functions, optimal number of neurons and three-layer architecture [6 10 11] on the second tasting dataset.

In addition, it should be noted that classifiers based on the class-dependent Fischer's components achieve intermediate results, with accuracies around 70%-75%, but when analyzing the confusion matrices their hit rates in the *Bad* and *Excellent* categories are still much lower, with erroneous predictions of 70%.

7.8 PNN

When applying the PNN algorithm on the different datasets, we proceed as described in section 2.7. In this case the only design parameter to work with is the parameter σ. First we will estimate the approximate range of values in which σ can be valid by training a classifier with some training data and testing the classifier with the training data itself. The range of values that we will then use for σ will be around the values that produce a change in the results from classifying all data correctly to classifying all data into a single category, typically between $\sigma = 0{,}1$ and $\sigma = 50$. In order to choose the most favorable option, we will act in a manner analogous to that used with the kNN classifiers. We reserve one sample to test the classifiers using LOO, of the remaining samples we reserve one to validate the classifier and select the optimal σ parameter, and we use the remaining samples for training. Thus, in the case of the first set we again obtain 96 different classifiers with 95 validated predictions for each according to σ. With the predictions we form their confusion matrix and calculate the total losses by summing the costs. Finally, we average the costs among the 96 experiments performed by choosing the optimal σ parameter. Note again that in order to obtain this value, the samples reserved for testing the final classifier were not used in any case.

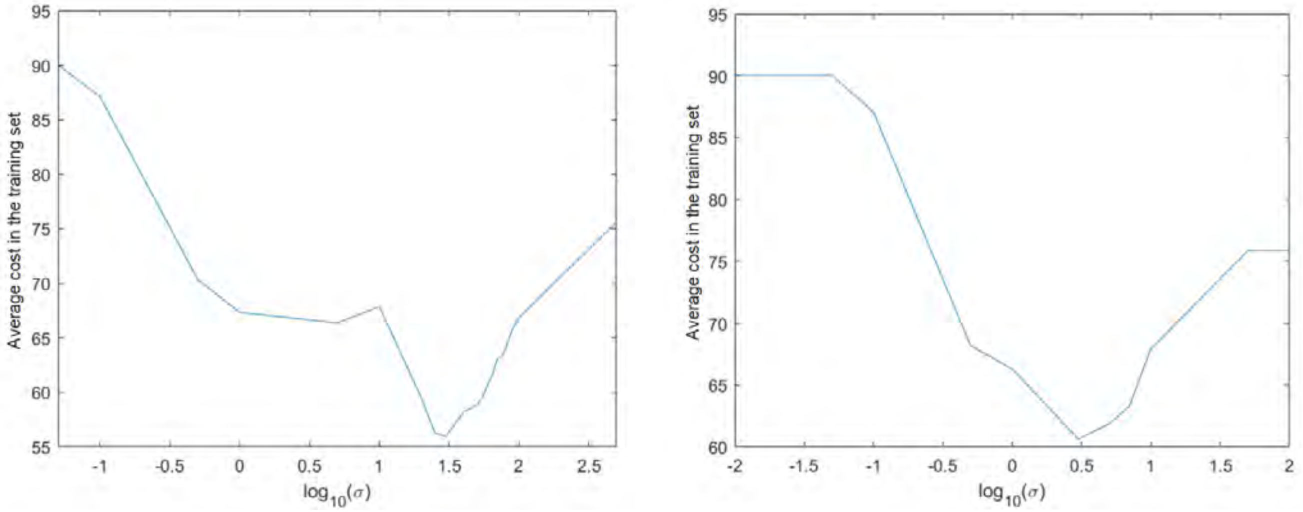


Figure 98. Average training cost by varying σ over the validation sets for the first taster's original features (left) and PCA features$_{95\%}$ (right).

As an example using the first dataset, the procedures and graphs (Figure 99) obtained by using the original features and the first 23 features of the PCA space, the so-called PCA$_{95\%}$. In the first case, the initial estimate of the σ values is around 10, in the second case a quick inspection on the training set reveals that the changes in accuracy occur at the σ values close to 1.

The final results when validating the classifiers by LOO are similar to those presented in kNN. The first dataset shows very poor results for almost all feature sets, with the exception of the LDA principal components. Taking both the original 62 features, as well as the various feature selections, and proceeding in the feature space obtained by applying PCA, as well as applying the QDA space, the classifiers show overall accuracies below or close to 40%. The results and the values of σ with which they were obtained are detailed in Table 68. This is due, once again, to the fact that the PNN algorithm must work with a separable representation of the categories in its input, and in these spaces this does not happen.

Features		σ	Accuracy (%)
Originals		30	41.67
Anthocyanin derivatives		1	40.60
QDA Selection		5	35.00
LDA$_1$ Selection		0.5	34.38
PCA	99%	5	34,38
	95%	3	36,46
	90%	5	35,42
LDA		0.005	89.58
QDA		10	34.38

Table 68. Optimum σ parameter and final accuracy of the LOO validated PNN classifier with different feature selections on the first taster's data.

Except for the PCA classifier$_{95\%}$ all these classifiers show, as with the kNN technique, a clear tendency to classify in the majority classes (*Medium* and *Good*) with almost no cases classified as *Excellent* or *Bad*, which prevents any positive or negative predictions from being made. In the case of the PCA classifier$_{95\%}$, up to 11 of the 96 cases are classified as *Excellent*, but only one of them is correct, which is equally useless for making positive or negative predictions.

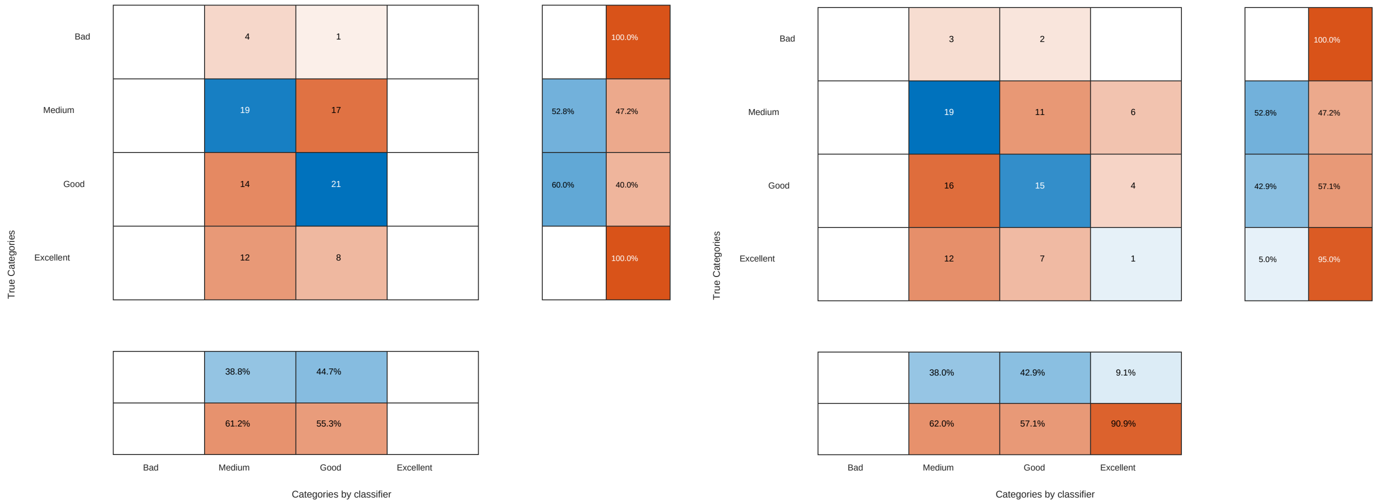


Figure 99. Confusion matrix of the LOO validated PNN classifier for the optimal σ with standard losses on the first dataset: original features (left) and PCA$_{95\%}$ space (right).

Thanks, again, to the separation achieved in the main LDA components, the application of PNN on this dimensional space is remarkably better with an accuracy of 89.58% for $\sigma = 0{,}005$. The confusion matrix for this classifier can be found in Figure 100.

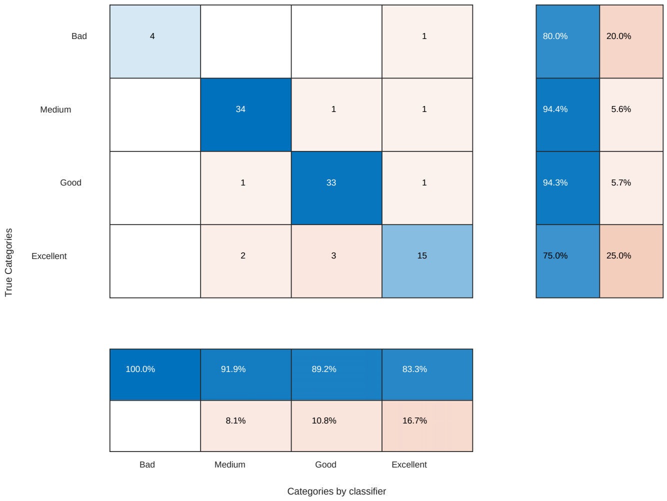


Figure 100. Confusion matrix of the LOO validated PNN classifier for σ optimal on the LDA space of the first dataset.

Analyzing the behavior of the classification regions in the first two Fischer's components for different values of σ (Figure 101) it can be seen that, the lower σ value, the classifier takes results and classification regions more and more similar to those of kNN classifiers with few neighbors, as pointed out by Spetch [71].

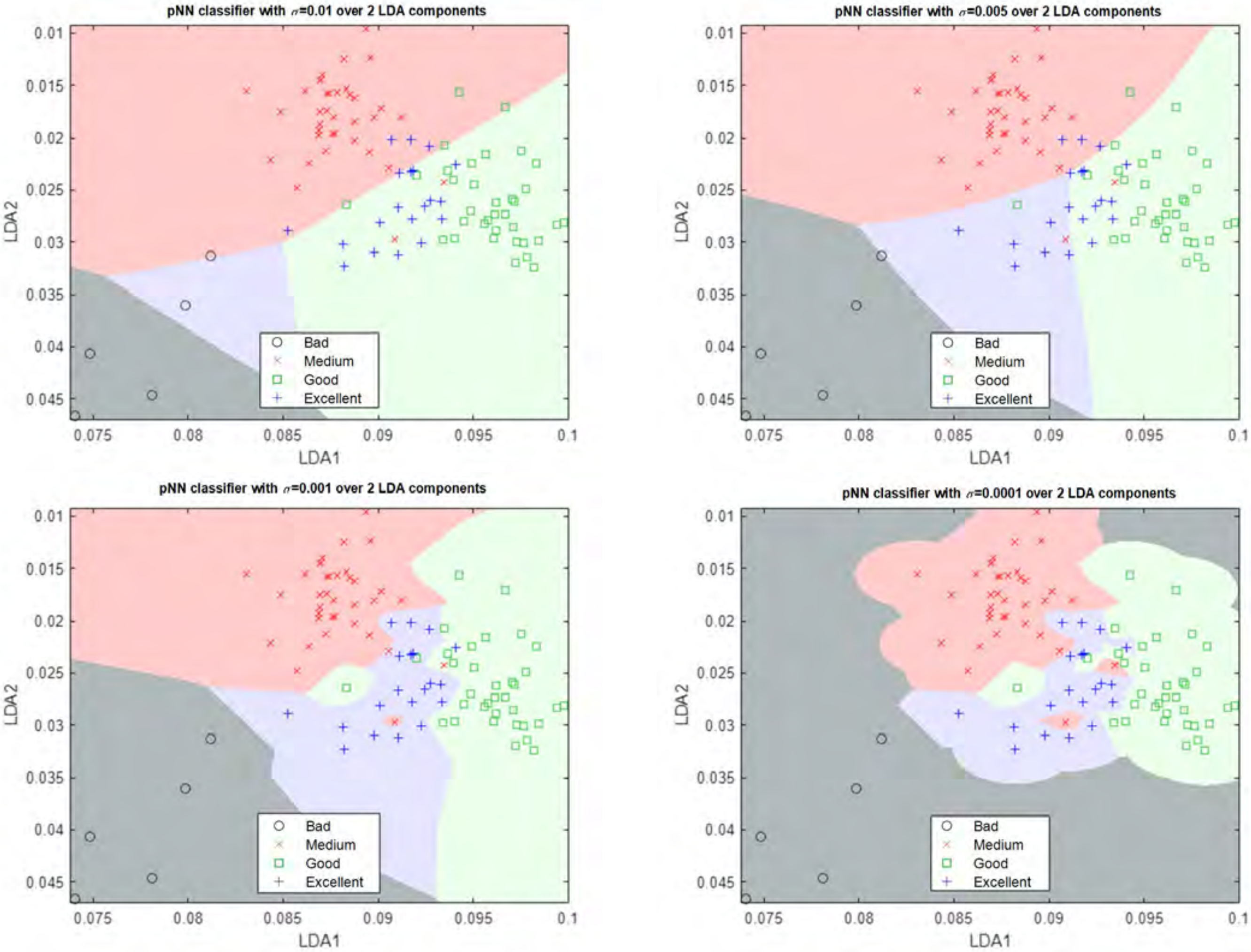


Figure 101. Classification regions of PNN classifiers using the first two Fischer's principal components on the first dataset for different values of σ.

The results of applying PNN to the second are shown in Table 69, together with the optimal parameter *σ* for each set of features used. Applying the algorithms to the second taster's dataset gives similar results. When expressing the data in the full original features, with the anthocyanin derivatives, the *selected QDA,* those of the PCA space with different amounts of principal components and those of the QDA space, the results are poor, although somewhat better than for the first taster.

Features		σ	Accuracy (%)
Originals		50	32.81
Anthocyanin derivatives		0.1	34.38
QDA Selection		10	35.94
PCA	99%	3	40,63
	95%	3	40,63
	90%	1	43,75
LDA		0.07	96.88
QDA		1	48.44

Table 69. Optimum σ parameter and final accuracy of the LOO validated PNN classifier with different feature selections on the second taster's data.

The results when presenting the features in the Fischer's independent components are excellent with accuracies of 96.8%, misclassifying only two samples in categories close to the real ones.

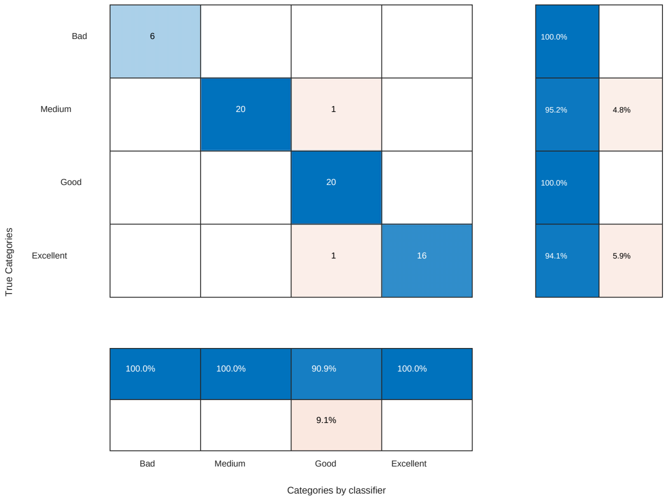


Figure 102. Confusion matrix of the LOO validated PNN classifier for σ optimal over the LDA space of the second dataset.

7.9 Summary of results

In the following tables we will summarize the results for the best classifiers for each of the techniques applied. The Table 70 summarizes the results for the first taster and Table 71 those corresponding to the second taster. The results are divided into the classifiers that used the standard loss matrix and those that used the matrix proposed in Table 38. Together with the overall accuracy of the classifier, the accuracies only of the categories referring to the significant predictions extracted from the confusion matrices have been reflected. The first columns express the percentage of correct samples out of the samples proposed by the classifier in the *Bad* and *Excellent* categories. These percentages are a measure of the good quality of the significant predictions of each classifier. The last two columns express the number of samples, as a percentage, that tasters have rated as *Bad* and *Excellent* that the classifiers would not recommend, by predicting them in intermediate categories. These percentages give a measure of the number of possible predictions that each classifier losses.

		Standard Losses					Proposed Losses				
Classifier	**Design parameters**	Overall accuracy (%)	Samples correct proposals as *Bad* (%)	Correct samples proposed *Excellent* (%)	Actual cases cat. *Bad* lost (%)	Real cases cat. *Excellent* lost (%)	Overall accuracy (%)	Correct samples proposed as *Bad* (%)	Correct samples proposed *Excellent* (%)	Actual cases cat. *Bad* lost (%)	Real cases cat. Excellent lost (%)
LDA	Three components	90.62	100	81.0	20	15	89.58	100	80,0	20	20
QDA		88.54	100	100	40	30	87.50	100	100	20	40
NB about LDA feat.	Gaussian distribution	89.58	100	76.2	20	20	88.54	100	78,9	20	25
	Kernel distribution	88.54	100	80.0	20	20	88.54	100	83,3	20	25
NB about QDA feat.	Gaussian distribution	89.58	80	81.8	20	10	89.58	80	81,8	20	10
	Kernel distribution	85.42	66.7	78.3	60	10	85.42	66,7	81,8	60	10
kNN about LDA feat.	Dist. Euclidean 4 neighbors	89.58	100	88.2	20	25	87.50	100	86,7	20	35
	Euclidean Dist. 8 neighbors	88.54	100	75.0	20	25	87.50	100	77,8	20	30
	Dist. Manhattan 4/8 neighbors[49]	88.54	100	82.4	20	30	87.50	100	92,9	20	35
CART about LDA feat.	Gini, const. validation	84.38	100	68.4	20	35	84.38	100	76,5	20	35
	Gini prop. validation	83.33	100	61.9	0	35	84.38	100	63,6	0	30
	Proportional twoing	87.50	100	71.4	0	25	87.50	100	71,4	0	25

[49] 4 neighbors for classifier with standard losses and 8 neighbors for classifier with proposed losses.

MLP about LDA feat.	Relu [46 12]	91.66	100	80.0	20	20					
	Sigmoidal [49]	87.50	100	73.7	40	30					
PNN about LDA feat.	$\sigma = 0,005$	89.58	100	83.3	20	25					

Table 70. Summary of results for LOO validated classifiers on first taster's data.

Among the classifiers of the first taster we can observe that the introduction of the modified loss matrix improves in almost all cases the positive significant predictions made, since the percentage of correct samples among those proposed as *Excellent* is always higher. This increase in most cases comes at the cost of not recommending some of the cases originally rated as such, that is, at the cost of losing certain recommendations in favor of safer recommendations. The classifiers of the second taster show the same behavior.

		Standard Losses					**Proposed Losses**				
Classifier	**Design parameters**	Overall accuracy (%)	Samples correct proposals as *Bad* (%)	Correct samples proposed *Excellent* (%)	Actual cases cat. *Bad* lost (%)	Real cases cat. *Excellent* lost (%)	Overall accuracy (%)	Correct samples proposed as *Bad* (%)	Correct samples proposed *Excellent* (%)	Actual cases cat. *Bad* lost (%)	Real cases cat. Excellent lost (%)
LDA	Three components	95.31	100	94.1	0	5.9	96.87	100	100	0	5,9
QDA		82.81	100	83.3	33.3	11.8	81.25	100	83,3	33,3	16,7
NB about LDA feat.	Gaussian distribution	95.31	100	94.1	0	5.9	95.31	100	94,1	0	5,9
	Kernel distribution	96.88	100	94.4	0	0	95.31	100	94,1	0	5,9
NB about QDA feat.	Gaussian distribution	81.25	100	100	33.3	35.3	81.25	100	100	33,3	35,3
	Kernel distribution	84.38	100	93.3	33.3	17.6	84.38	100	100	33,3	23,5
kNN about LDA feat.	Dist. Euclidean 5/9 neighbors[50]	97.91	100	94.4	0	0	97.91	100	100	0	5,9
	Dist. Manhattan 9/11 neighbors[51]	97.91	100	94.4	0	0	97.91	100	100	0	5,9
CART about LDA feat.	Gini, cte. validation	93.75	100	94.1	16.7	5.9	90.63	100	94,1	16,7	5,9
	Constant twoing.	82.81	83.3	83.3	16.7	11.8	81.25	55,6	83,3	16,7	11,8
	Proportional twoing	79.69	0	83.3	100	11.8	87.50	66,7	88,2	0	11,8
MLP about 3 feat. LDA	Relu [38 17]	98.44	100	100	0	0					
	Sigmoidal [39]	93.75	100	93.8	0	11.8					
MLP about 2 feat. LDA	Relu [6]	92.19	100	88.9	0	5.9					
	Relu [6 10 11]	93.75	100	94.4	0	0					
PNN about LDA feat.	$\sigma = 0,07$	96.88	100	100	0	5.9					

Table 71. Summary of results for LOO validated classifiers on the second taster's data.

Among the classifiers created on the first taster's data, the lossless QDA classifier obtains more reliable predictions than the other classifiers but in return, it loses a higher number of samples of the samples originally rated as *Bad* and *Excellent* in the recommendations (40% and 30% respectively). On the other hand, the kNN classifier without losses on the LDA components with 4 neighbors and Euclidean distance offers the second best classification rate for wines categorized as *Excellent* (88.2%) losing 20% and 25% of the original samples. The LDA classifier with standard losses offers an intermediate level between the two with hit rates of 81% in the *Excellent* category and losing 20% and 15% of the original samples. Among the classifiers created on the data of the second taster, the MLP on 3 independent Fischer components and two hidden layers with relu activation functions is able to provide reliable predictions without missing any samples. Losing only 5.9% of the samples originally rated as *Excellent* but making reliable predictions are a second group of classifiers: PNN, LDA with proposed losses, kNN over the LDA components with proposed losses (regardless of the distance metric used) and MLP on only two LDA components. As can be seen in both tasters the MLP classifiers with relu activation functions perform better than those based on sigmoidal activation functions.

[50] 5 neighbors for classifier with standard losses and 9 neighbors for classifier with proposed losses.

[51] 9 neighbors for classifier with standard losses and 11 neighbors for classifier with proposed losses.

In view of the results, all LDA, QDA, NB, kNN, CART, MLP and PNN classifiers are useful when applied to the prediction of hedonic scores, depending on the taster to be predicted. This indicates that mixed predictions based on several classifiers in parallel could be useful to improve their reliability without losing samples.

Chapter VIII.

Conclusions and Future Lines

8 Conclusions and Future lines of research

8.1 Conclusions

The current research proposes and tests the applicability of different methodologies to make personalized recommendations for red wines from the Rioja DO, following the following steps:

1. Characterization of each of the wines by anthocyanin, anthocyanin derivatives and tannin content through High Performance Liquid Chromatography coupled to a tandem detector Mass Spectrometer (HPLC-MS/MS), detailed in Chapter 5.
2. Guided assessment of red wines from the Rioja DO using a tasting sheet standardized by the OIV, detailed in Chapter 6.
3. Grouping of the personalized scores in four macro-categories: *Excellent* or positively recommended, *Bad* or negatively recommended and two categories without recommendation (*Medium* and *Good*). The grouping for each taster is carried out independently as detailed in section 6.4.
4. Application of PCA techniques, class-independent and class-dependent Fischer's discriminants to obtain representations of wines in new feature spaces. Proposal of potential representations and compressions of the data of each wine in the new spaces: PCA_{100}, PCA_{99}, PCA_{95}, PCA_{90}, LDA and QDA. Selection of feature subsets (the so-called LDA_1 selection, LDA_2 selection and QDA selection) from the original set that provide further information in the new feature spaces. The algorithms and their implementation are explained in sections 2.2 y 2.3. The application of the algorithms, their results and feature selections are detailed in Sections 7.1, 7.2 and 7.3.
5. Training of supervised ML classifier algorithms (LDA, QDA, NB, NN, CART, MLP and PNN) on the different representations of wines and macrocategories defined previously. Validation of classifiers using LOO. Review of confusion matrices to confirm the ability to make meaningful recommendations. The algorithms and their implementation are explained in the Chapter 2. The application of the algorithms and their results are detailed in the Chapter 7.

The current research is an innovation in the application of different ML techniques to a red wine recommendation system. In particular, it is a novelty in showing the applicability and results of PCA, LDA, QDA, NB, kNN, CART, MLP and PNN techniques in a comparative way.

The main conclusion of this research is that it is possible to make significant positive and negative recommendations for the individualized recommendation of DO Rioja red wines using the proposed methodology, as shown by the results of the classifiers applied to each taster separately.

Regarding wine characterization and data acquisition, the methodology has proven that the alcohol content, anthocyanins, anthocyanin derivatives and tannins are sufficient to characterize DO Rioja red wines in order to make recommendations. The quantification of the concentrations in DO Rioja wines of certain anthocyanins, anthocyanin derivatives and tannins by reversed-phase HPLC with tandem MS/MS detector coupling is possible and sufficient to characterize DO Rioja red wine samples for future recommendations.

Regarding the organoleptic evaluation and scoring of the wines, the guided methodology for evaluating the wines through a standardized tasting sheet, evaluating a visual, an olfactory and a gustatory section, with a percentage of 59% of the points open to the evaluation of personal sensations, offers assessments with Gaussian distributions centered on average scores. The guided methodology for evaluating wines is a valid process that produces consistent and quality data to differentiate the samples according to the opinion of each of the tasters. The grouping of scores into four broad categories facilitates and makes possible a subsequent classification of the samples according to each taster's opinion.

Regarding the representation spaces of the samples, the application of PCA on the original data does not seem to obtain significant improvements in the classifiers. The expression of the data in class-independent Fischer’s principal components allows to significantly improve the classification of any classifier applied afterwards, being an essential step. Expressing the data in class-dependent Fischer principal components improves the classification of some classifiers but only to certain levels. Compared to the decomposition into class-independent Fischer principal components its results are inferior. The application of the PCA technique prior to the Fischer decomposition also fails to improve the classification results.

The reduction of features to those most influential in the class-independent Fischer’s principal components (referred to as LDA_1 and LDA_2 selections) does not provide significant results in any classifier, which seems to indicate that the information contained in the other features is also relevant. The same is true for the proposed feature reduction based on those features most present in the class-dependent Fischer’s principal components (termed QDA selection).

The proposed loss matrix helps to produce classifiers that make more reliable significant predictions, with a higher accuracy per significant category, but at the cost of reducing the total number of wines recommended, that is, losing more possible recommendations. The use of a loss matrix other than the standard one in the classifications does not have a decisive influence.

All classifiers LDA, QDA, NB, kNN, CART, MLP and PNN are useful when applied to the prediction of hedonic scores, depending on the taster to be predicted. The decision on which classifiers to base the decision on should be made depending on the taster to be modeled.

MLP classifiers with relu activation functions perform comparatively better than those based on sigmoidal activation functions.

It is feasible to classify and make significant positive and negative predictions on a minimum number of 60 samples assessed by tasters. It is feasible to create a customized model that is capable of delivering significant predictions with rates higher than 80% on at least 60 samples. The essential use of Fischer’s component decomposition to achieve such results indicates the need to use a small number of features, no more than the 12 resulting from applying QDA.

8.2 Future work

The present research opens a future developable path mainly in three areas. On the one hand, the application of new ML techniques or variants of the described techniques. On the other hand, the adaptation of the way in which wines are assessed for marketable recommendation systems. Finally, the description of wines by means of features other than or in addition to those already used.

The number of techniques available in the field of ML is tremendously wide. In the current research some of them have been applied, but techniques such as SVM [5], Fuzzy Neural Networks (FNN) [199] or the Spikinge Neural Networks (SNN) [200] may also be applicable as classifiers.

Variations can also be made on the ideas of the algorithms used, such as varying the downward gradient mechanism when training MLPs in order to escape from local minima in the search for a better solution. The alternatives in this sense are manifold: simulated annealing, tabu search, genetic algorithms [67]. These nondeterministic techniques have in common that each iteration will not always choose a neighbor that improves the solution of the previous iteration, but may select a solution with a higher error cost. The simulated tempering technique is an analogy of the physical tempering process of steel, glass or ceramics that starts from a high temperature level to cool it in a controlled manner to achieve a material with the desired properties. The acceptance of a worse solution depends on a probabilistic function which in turn is a function proportional to temperature. The technique will start with a high temperature value that will decrease in each iteration in a controlled manner, so that in the first iterations it will be more likely to accept worse solutions and as the iterations progress it will discard them, finally converging to a stable minimum. The tabu search performs a random search among the possible nearby solutions, always comparing it with the optimal one, but remembering the paths visited in the most recent iterations. In each new iteration it will avoid exploring the most recent paths (marked as tabu) in favor of new paths. Genetic algorithms are inspired by the mechanisms of natural evolution. They start from a bank of solutions and randomly combine the best solutions found to generate new ones that will replace the previous ones in the bank until a final set of solutions is achieved where any of them can be used. Combination is not the only mechanism that allows improvement, because from time to time (with a very low probability) accidental mutations arise in the genes. Although these mutations are mostly discarded due to their worse outcome, sometimes they result in improvements that help a better generation of solutions. All these techniques can also be used to optimize the design parameters of any of the classifiers presented in section 2.

In addition, while many features sets have provided results with accuracies of less than 40%, this may be due to a low number of samples in the data [31], so it cannot be ruled out that a similar procedure with a larger number of samples could provide better accuracies. Conducting a study with 2000 samples of red wines per taster could show how the original features relate to the score, but such numbers are beyond our scope.

On the other hand, commercial recommendation systems tend to be oriented towards greater customer satisfaction and, as a consequence, higher product sales, but without requiring them to invest excessive time or answer a number of uncomfortable questions. Guided assessment, following a tasting sheet, has provided more detailed and orderly information on the perception of the different aspects of the product at the cost of requiring more time and answering a greater number of questions, making it more uncomfortable for commercial systems. This option also requires some prior education of the consumer to be able to taste and evaluate the different aspects of the wine, especially the aromatic aspect, experience that not all users of the recommendation system may have. In any case, it is inevitable to assume that a very important weight in the final score will correspond to the purely subjective aspects that are hidden behind the objective descriptions. Commercial systems tend to focus on the assessment of pleasure and originality perceived by the user. Its presence to the eye, its smell, its taste and many other factors end up being summarized in a variable that simplistically represents consumer satisfaction. The system proposed in section 6 should rethink how to obtain ratings in a simpler way and by inexperienced tasters, while maintaining the reliability of data collection.

The environment, the conditions and the way in which wine is usually consumed are not homogeneous, even for the same taster. Wine is hardly consumed in tasting conditions, in isolation and without interference. On the contrary, it is usually drunk in society, sharing, paired with food, smoking, in the street, etc. Tasting under these conditions can have a very different impression from the one obtained in the section 6. We should not forget that it is this opinion that we are seeking to assess. We should therefore look for intermediate conditions between the tasting in an isolated space and the environment of consumption and personal enjoyment of each consumer. It is a task for future research to demonstrate that the system is equally valid if the evaluations are taken in the usual places of consumption.

In addition, the same person's assessment of the same wine can be different at different times. The perception of flavors is highly variable, and even in the case of the same individual it can vary according to his or health or eating habits. This can lead to different wine scores based on external features that should be considered in the design of improved classifiers. Another possibility without including new features is some kind of probabilistic treatment of the ratings or a possible use of FNN, since fuzzy algorithms can better model this space of differences between ratings. A final possibility in the same vein could be the use of evolutionary networks such as the Evolving SNNs used by Kasabov et al. [201] to generate adaptive personalized models over time.

Finally, although the characterization of red wines through anthocyanins, anthocyanin derivatives and tannins proposed in section 5 represents an intense work, the characterization of the aromatic nuances produced by the wines and appreciated especially by the professional tasters, seems more directly related to the aromatic compounds, which are much more volatile and cannot be analyzed using the techniques used in this research. Moreover, in order to extend a similar study to white wines, it remains to be discussed which compounds to choose to characterize them, since the winemaking process followed and the type of grapes used rule out those measured here.

References

9 References

[1] L. Breiman, J. Friedman, C. J. Stone and R. A. Olshen, Classification and Regression Trees, Taylor & Francis, 1984.

[2] L. Breiman, "Random forests," *Machine learning,* vol. 45, pp. 5-32, 2001.

[3] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation," 1985.

[4] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation," *Proceedings of the IEEE,* vol. 78, pp. 1415-1442, 1990.

[5] B. E. Boser, I. M. Guyon and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory - COLT 92*, 1992.

[6] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning,* vol. 20, pp. 273-297, 9 1995.

[7] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory,* vol. 13, pp. 21-27, 1 1967.

[8] R. O. Duda, P. E. Hart and D. G. Stork, Pattern Classification, Wiley John & Sons, 2000.

[9] P. Langley, W. Iba, K. Thompson and others, "An analysis of Bayesian classifiers," in *Aaai*, 1992.

[10] D. W. Hosmer Jr, S. Lemeshow and R. X. Sturdivant, Applied logistic regression, 3rd ed., vol. 398, John Wiley & Sons, 2013.

[11] P. y. A. Ministerio de Agricultura, "Informe del Consumo Alimentario en España 2018," 2018.

[12] K. Anderson, S. Nelgen and V. Pinilla, Global wine markets, 1860 to 2016: a statistical compendium, University of Adelaide Press, 2017.

[13] Observatorio Español del Mercado del Vino, "Dossier estadístico del vino," C/ Atenas 2, 1º F - Pozuelo de Alarcón, Madrid 28224, 2019.

[14] Organización Internacional de la Viña y el Vino, "Organización Internacional de la Viña y el Vino: Estadísticas," [Online]. Available: http://www.oiv.int/es/statistiques/recherche. [Accessed 20 Febrero 2020].

[15] P. y. A. Ministerio de Agricultura, "ESYRCE. Encuesta sobre Superficies y Rendimientos 2018," 2019.

[16] I. N. Estadística, "España en cifras 2018," *España en cifras 2018,* 2019.

[17] Instituto Nacional de Estadística, "Encuesta Industrial de Empresas. Serie 2008-2014. CNAE-2009.," [Online]. Available: https://www.ine.es/dynt3/inebase/index.htm?padre=420. [Accessed 2 Febrero 2020].

[18] Ministerio de Agricultura, Pesca y Alimentación, "Cuentas Económicas de la Agricultura (Renta Agraria: Macromagnitudes Agrarias)," [Online]. Available: https://www.mapa.gob.es/es/estadistica/temas/estadisticas-agrarias/economia/cuentas-economicas-agricultura/. [Accessed 2 Febrero 2020].

[19] Gobierno de La Rioja, "Agregados contables según agrupaciones de actividad (CNAE 09) y años.," [Online]. Available: https://www.larioja.org/estadistica/es/area-tematica-economia/industria-energia/encuesta-industrial-empresas-rioja/eie_contenido/agregados-contables-segun-agrupaciones-actividad-cnae-09-an. [Accessed 2 Febrero 2020].

[20] G. M. V. d. l. Torre, "Las denominaciones de origen y las rutas del vino en España: un estudio de caso," *Rotur: revista de ocio y turismo,* pp. 41-66, 2012.

[21] F. Q. Sanz, "Vino, aristócratas, tumbas y guerreros en la cultura ibérica (ss. V-II a.C.)," *Verdolay: Revista del Museo Arqueológico de Murcia,* pp. 99-124, 1994.

[22] C. Riva, "Wine production and exchange and the value of wine consumption in sixth-century BC Etruria," *Journal of Mediterranean Archaeology,* vol. 30, pp. 237-261, 2017.

[23] E. D. Cusí, C. G. Bellard and P. G. Fockedey, "El vino en los inicios de la cultura ibérica: Nuevas excavaciones en L´Alt de Benimaquia, Denia," *Revista de arqueología,* pp. 16-27, 1993.

[24] J. B. Pérez and R. O. Romera, "Arqueología en Albacete : jornadas de arqueología albacetense en la Universidad Autónoma de Madrid," J. B. Pérez, R. S. Gamo and M. T. M. Hervás, Eds., 1993, pp. 85-110.

[25] A. Pecci, P. Reynolds, S. Mileto, J. M. Vargas Giron and D. Bernal-Casasola, "Production and transport of goods in the Roman period: Residue analysis and wine derivatives in Late Republican Baetican ovoid amphorae," *Environmental Archaeology,* pp. 1-13, 2021.

[26] V. Martínez Ferreras, C. Capelli, M.-p. Jézégou, M. Salvat, G. Castellvi and R. Cabella, "The Port-Vendres 4 Shipwreck Cargo: evidence of the Roman wine trade in the western Mediterranean," *International Journal of Nautical Archaeology,* vol. 44, pp. 277-299, 2015.

[27] D. S. Sáez, *Historia Económica, Social y Cultural del Vino en la Ribera del Duero Vallisoletana,* Valladolid: Diputación de Valladolid, 2008.

[28] M. Á. A. Mondragón, "El vino de Rioja: historia y cultura," 2012.

[29] J. Peñín, Historia del vino, Espasa, 2008.

[30] J. R. Hilera González and V. J. Martínez Hernando, Redes Neuronales Artificiales Fundamentos, modelos y aplicaciones, Ra-Ma, 1995.

[31] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.

[32] A. Ng, *Machine Learning Course,* Standford University, 2020.

[33] A. K. Jain, P. W. Duin and J. Mao, "Statistical pattern recognition: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, pp. 4-37, 2000.

[34] I. Bilbao and J. Bilbao, "Overfitting problem and the over-training in the era of data: Particularly for Artificial Neural Networks," in *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2017.

[35] K. Fukunaga and R. R. Hayes, "Effects of sample size in classifier design," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 11, pp. 873-885, 1989.

[36] Vapnik, Statistical Learning Theory, John Wiley & Sons, 1998.

[37] D. C. Lay, J. J. McDonald and S. R. Lay, Linear Algebra and Its Applications, 6 ed., Pearson Education, 2020.

[38] A. Tharwat, T. Gaber, A. Ibrahim and A. E. Hassanien, "Linear discriminant analysis: A detailed tutorial," *AI Communications,* vol. 30, pp. 169-190, 5 2017.

[39] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics,* vol. 7, pp. 179-188, 9 1936.

[40] K. Fukunaga, Introduction to Statistical Pattern Recognition, Elsevier Science Publishing Co Inc, 1990.

[41] S. S. Wilks, "Mathematical statistics," 1964.

[42] J. Ye, R. Janardan and Q. Li, "Two-dimensional linear discriminant analysis," in *Advances in neural information processing systems*, 2005.

[43] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural computation,* vol. 12, pp. 2385-2404, 2000.

[44] S. Mika, G. Ratsch, J. Weston, B. Scholkopf and K.-R. Mullers, "Fisher discriminant analysis with kernels," in *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*, 1999.

[45] X.-S. Zhuang and D.-Q. Dai, "Inverse Fisher discriminate criteria for small sample size problem and its application to face recognition," *Pattern Recognition,* vol. 38, pp. 2192-2194, 2005.

[46] A. Sharma and K. K. Paliwal, "Linear discriminant analysis for the small sample size problem: an overview," *International Journal of Machine Learning and Cybernetics,* vol. 6, pp. 443-454, 2015.

[47] J. Lu, K. N. Plataniotis and A. N. Venetsanopoulos, "Regularization studies of linear discriminant analysis in small sample size scenarios with application to face recognition," *Pattern recognition letters,* vol. 26, pp. 181-191, 2005.

[48] H. Yu and J. Yang, "A direct LDA algorithm for high-dimensional data—with application to face recognition," *Pattern recognition,* vol. 34, pp. 2067-2070, 2001.

[49] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician,* vol. 46, pp. 175-185, 1992.

[50] Z. Qin, A. T. Wang, C. Zhang and S. Zhang, "Cost-Sensitive Classification with k-Nearest Neighbors," in *Knowledge Science, Engineering and Management*, Springer Berlin Heidelberg, 2013, pp. 112-131.

[51] W. S. McCulloch and W. Pitts, "How nervous structures have ideas," *Transactions of the American Neurological Association,* pp. 10-16, 1949.

[52] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics,* vol. 5, pp. 115-133, 12 1943.

[53] D. O. Hebb, The organization of behavior: A neurophysiological theory, John Wiley & Sons, 1949.

[54] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review,* vol. 65, pp. 386-408, 1958.

[55] F. Rosenblatt, "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms," 1961.

[56] K. Bolton, "krisbolton.com," 09 04 2018. [Online]. Available: https://krisbolton.com/a-quick-introduction-to-artificial-neural-networks-part-1. [Accessed 15 04 2022].

[57] B. Widrow and M. E. Hoff, "Adaptive switching circuits," 1960.

[58] M. Minsky and S. Papert, Perceptrons: An Introduction to Computational Geometry, MIT Press, 1969.

[59] M. Musiol, "Speeding up Deep Learning," 2016.

[60] D. E. Rumelhart and J. McClelland, Parallel Distributed Processing. Explorations in the Microstructure of Cognition: Foundations, vol. 1, MIT Press, 1986.

[61] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, pp. 533-536, 10 1986.

[62] J. A. Anderson, J. W. Silverstein, S. A. Ritz and R. S. Jones, "Distinctive features, categorical perception, and probability learning: Some applications of a neural model.," *Psychological Review,* vol. 84, pp. 413-451, 9 1977.

[63] K. Fukushima, S. Miyake and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics,* Vols. SMC-13, pp. 826-834, 9 1983.

[64] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics,* vol. 43, pp. 59-69, 1982.

[65] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities.," *Proceedings of the National Academy of Sciences,* vol. 79, pp. 2554-2558, 4 1982.

[66] J. E. Marsden and A. J. Tromba, Cálculo Vectorial, 3 ed., Addison-Wesley Iberoamericana, 1991.

[67] A. García, Inteligencia Artificial. Fundamentos, práctica y aplicaciones, Rc Libros, 2012.

[68] E. Parzen, "On Estimation of a Probability Density Function and Mode," *The Annals of Mathematical Statistics,* vol. 33, pp. 1065-1076, 9 1962.

[69] T. Cacoullos, "Estimation of a multivariate density," *Annals of the Institute of Statistical Mathematics,* vol. 18, pp. 179-189, 12 1966.

[70] V. K. Murthy, "Estimation of Probability Density," *Ann. Math. Statist.,* vol. 36, pp. 1027-1031, 6 1965.

[71] D. F. Specht, "Probabilistic neural networks," *Neural Networks,* vol. 3, pp. 109-118, 1 1990.

[72] D. F. Specht, "Generation of Polynomial Discriminant Functions for Pattern Recognition," *IEEE Transactions on Electronic Computers,* Vols. EC-16, pp. 308-319, 6 1967.

[73] W. Zheng, "Viticultural techniques of canopy management to mitigate the effects of global warming," 2017.

[74] C. Falcó, Entender de vino, 10 ed., Madrid: Ediciones Martínez Roca, 2004, pp. 219-240.

[75] J. H. Togores, Tratado de enología, vol. 1, Mundi-Prensa Libros, 2010.

[76] C. Flanzy, Enología: fundamentos científicos y tecnológicos, 2 ed., Mundi Prensa Libros: A. Madrid Vicente, Ediciones, 2003.

[77] N. Prieto Perea, "Anthocyanin derivatives and tannins of red wine by hplc-dad-ms/ms. Evolution along winemaking and comparision between rioja and bordeaux wines," 2018.

[78] M. Cano-López, J. M. López-Roca, F. Pardo-Minguez and E. G. Plaza, "Oak barrel maturation vs. micro-oxygenation: Effect on the formation of anthocyanin-derived pigments and wine colour," *Food chemistry,* vol. 119, pp. 191-195, 2010.

[79] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems,* vol. 47, pp. 547-553, 2009.

[80] P. Cortez, J. Teixeira, A. Cerdeira, F. Almeida, T. Matos and J. Reis, "Using Data Mining for Wine Quality Assessment," in *Discovery Science*, Berlin, 2009.

[81] R. C. Team, "R: A Language and Environment for Statistical Computing," Vienna, 2020.

[82] Y. Gupta, "Selection of important features and predicting wine quality using machine learning techniques," *Procedia Computer Science,* vol. 125, pp. 305-312, 2018.

[83] S. Kumar, Y. Kraeva, R. Kraleva and M. Zymbler, "A Deep Neural Network Approach to Predict the Wine Taste Preferences," in *Intelligent Computing in Engineering*, Springer Singapore, 2020, pp. 1165-1173.

[84] S. Lee, J. Park and K. Kang, "Assessing wine quality using a decision tree," in *2015 IEEE International Symposium on Systems Engineering (ISSE)*, 2015.

[85] B. Chen, C. Rhodes, A. Crawford and L. Hambuchen, "Wineinformatics: Applying Data Mining on Wine Sensory Reviews Processed by the Computational Wine Wheel," in *2014 IEEE International Conference on Data Mining Workshop*, 2014.

[86] A. C. Noble, R. A. Arnold, B. M. Masuda, S. D. Pecore, J. O. Schmidt and P. M. Stern, "Progress Towards a Standardized System of Wine Aroma Terminology," *American Journal of Enology and Viticulture,* vol. 35, pp. 107-109, 1984.

[87] A. Noble, R. Arnold, J. Buechestein, E. Leach, J. Schmidt and P. Stern, "Modification of Standardized System of Wine Aroma Terminology," *American Journal of Enology and Viticulture,* vol. 38, no. 2, pp. 143-146, 1987.

[88] B. Chen, C. Rhodes, A. Yu and V. Velchev, "The Computational Wine Wheel 2.0 and the TriMax Triclustering in Wineinformatics," in *Advances in Data Mining. Applications and Theoretical Aspects*, Cham, 2016.

[89] B. Chen, *21st Century Bordeaux Wines Dataset,* IEEE DataPort, 2020.

[90] B. Chen, H. Le, C. Rhodes and D. Che, "Understanding the Wine Judges and Evaluating the Consistency Through White-Box Classification Algorithms," in *Advances in Data Mining. Applications and Theoretical Aspects*, Cham, 2016.

[91] S. Vlassides, J. G. Ferrier and D. E. Block, "Using historical data for bioprocess optimization: Modeling wine characteristics using artificial neural networks and archived process information," *Biotechnology and Bioengineering,* vol. 73, pp. 55-68, 2001.

[92] E. Vigneau, P. Courcoux, R. Symoneaux, L. Guérin and A. Villière, "Random forests: A machine learning methodology to highlight the volatile organic compounds involved in olfactory perception," *Food Quality and Preference,* vol. 68, pp. 135-145, 2018.

[93] S. Theodoridis and K. Koutroumbas, Pattern Recognition, Elsevier LTD, Oxford, 2008.

[94] S. Fuentes, D. D. Torrico, E. Tongson and C. G. Viejo, "Machine Learning Modeling of Wine Sensory Profiles and Color of Vertical Vintages of Pinot Noir Based on Chemical Fingerprinting, Weather and Management Data," *Sensors,* vol. 20, p. 3618, 6 2020.

[95] C. G. Viejo, D. D. Torrico, F. R. Dunshea and S. Fuentes, "Development of Artificial Neural Network Models to Assess Beer Acceptability Based on Sensory Properties Using a Robotic Pourer: A Comparative Model Approach to Achieve an Artificial Intelligence System," *Beverages,* vol. 5, p. 33, 5 2019.

[96] J. Song, R. Smart, H. Wang, B. Dambergs, A. Sparrow and M. C. Qian, "Effect of grape bunch sunlight exposure and UV radiation on phenolics and volatile composition of Vitis vinifera L. cv. Pinot noir wine," *Food chemistry,* vol. 173, pp. 424-431, 2015.

[97] S. Fuentes, C. Gonzalez Viejo, X. Wang and D. Torrico, "Aroma and quality assessment for vertical vintages using machine learning modelling based on weather and management information," 2019.

[98] G. Diana, C. Beni and S. Marconi, "Organic and Mineral Fertilization: Effects on Physical Characteristics and Boron Dynamic in an Agricultural Soil," *Communications in Soil Science and Plant Analysis,* vol. 39, pp. 1332-1351, 5 2008.

[99] P. Kment, M. Mihaljevič, V. Ettler, O. Šebek, L. Strnad and L. Rohlová, "Differentiation of Czech wines using multielement composition – A comparison with vineyard soil," *Food Chemistry,* vol. 91, pp. 157-165, 6 2005.

[100] S. M. Azcarate, L. D. Martinez, M. Savio, J. M. Camiña and R. A. Gil, "Classification of monovarietal Argentinean white wines by their elemental profile," *Food Control,* vol. 57, pp. 268-274, 11 2015.

[101] M. Inácio, V. Pereira and M. Pinto, "The soil geochemical atlas of Portugal: overview and applications," *Journal of Geochemical Exploration,* vol. 98, pp. 22-33, 2008.

[102] V. Ramachandran and T. J. D'souza, "Plant uptake of cadmium, zinc, and manganese in soils amended with sewage sludge and city compost," *Bulletin of environmental contamination and toxicology,* vol. 61, pp. 347-354, 1998.

[103] S. Frías, J. P. P. Trujillo, E. Peña and J. E. Conde, "Classification and differentiation of bottled sweet wines of Canary Islands (Spain) by their metallic content," *European Food Research and Technology,* vol. 213, pp. 145-149, 2001.

[104] N. Jakubowski, R. Brandt, D. Stuewer, H. R. Eschnauer and S. Görtges, "Analysis of wines by ICP-MS: Is the pattern of the rare earth elements a reliable fingerprint for the provenance?," *Fresenius Journal of Analytical Chemistry,* vol. 364, pp. 424-428, 7 1999.

[105] M. Iglesias, E. Besalú and E. Anticó, "Internal Standardization-Atomic Spectrometry and Geographical Pattern Recognition Techniques for the Multielement Analysis and Classification of Catalonian Red Wines," *Journal of Agricultural and Food Chemistry,* vol. 55, pp. 219-225, 12 2006.

[106] I. M. Moreno, A. J. Gutiérrez, C. Rubio, A. G. González, D. Gonzalez-Weller, N. Bencharki, A. Hardisson and C. Revert, "Classification of Spanish Red Wines Using Artificial Neural

Networks with Enological Parameters and Mineral Content," *American Journal of Enology and Viticulture,* vol. 69, pp. 167-175, 1 2018.

[107] P. P. Coetzee, F. P. Jaarsveld and F. Vanhaecke, "Intraregional classification of wine via ICP-MS elemental fingerprinting," *Food Chemistry,* vol. 164, pp. 485-492, 12 2014.

[108] A. E. Martin, R. J. Watling and G. S. Lee, "The multi-element determination and regional discrimination of Australian wines," *Food Chemistry,* vol. 133, pp. 1081-1089, 8 2012.

[109] A. Gonzálvez, A. Llorens, M. L. Cervera, S. Armenta and M. Guardia, "Elemental fingerprint of wines from the protected designation of origin Valencia," *Food Chemistry,* vol. 112, pp. 26-34, 1 2009.

[110] I. M. Moreno, D. González-Weller, V. Gutierrez, M. Marino, A. M. Cameán, A. G. González and A. Hardisson, "Differentiation of two Canary DO red wines according to their metal content from inductively coupled plasma optical emission spectrometry and graphite furnace atomic absorption spectrometry by using Probabilistic Neural Networks," *Talanta,* vol. 72, pp. 263-268, 2007.

[111] M. Álvarez, I. M. Moreno, Á. Jos, A. M. Cameán and A. G. González, "Differentiation of `two Andalusian DO `fino' wines according to their metal content from ICP-OES by using supervised pattern recognition methods," *Microchemical Journal,* vol. 87, pp. 72-76, 10 2007.

[112] J. Sperkova and M. Suchanek, "Multivariate classification of wines from different Bohemian regions (Czech Republic)," *Food Chemistry,* vol. 93, pp. 659-663, 12 2005.

[113] J.-P. Pérez-Trujillo, M. Barbaste and B. Medina, "Chemometric Study of Bottled Wines with Denomination of Origin from the Canary Islands (Spain) Based on Ultra-Trace Elemental Content Determined by ICP-MS," *Analytical Letters,* vol. 36, pp. 679-697, 2003.

[114] S. Frías, J. E. Conde, J. J. Rodríguez-Bencomo, F. García-Montelongo and J. P. Pérez-Trujillo, "Classification of commercial wines from the Canary Islands (Spain) by chemometric techniques using metallic contents," *Talanta,* vol. 59, pp. 335-344, 2003.

[115] M. Núñez, R. M. Peña, C. Herrero and S. García-Martín, "Analysis of some metals in wine by means of capillary electrophoresis. Application to the differentiation of Ribeira Sacra Spanish red wines," *Analusis,* vol. 28, pp. 432-437, 6 2000.

[116] L.-X. Sun, K. Danzer and G. Thiel, "Classification of wine samples by means of artificial neural networks and discrimination analytical methods," *Fresenius Journal of Analytical Chemistry,* vol. 359, pp. 143-149, 9 1997.

[117] G. Grindlay, J. Mora, L. Gras and M. T. C. Loos-Vollebregt, "Atomic spectrometry methods for wine analysis: A critical evaluation and discussion of recent applications," *Analytica Chimica Acta,* vol. 691, pp. 18-32, 4 2011.

[118] J. O'Reilly, M. J. Watts, R. A. Shaw, A. L. Marcilla and N. I. Ward, "Arsenic contamination of natural waters in San Juan and La Pampa, Argentina," *Environmental Geochemistry and Health,* vol. 32, pp. 491-515, 5 2010.

[119] M. d. M. Castiñeira, I. Feldmann, N. Jakubowski and J. T. Andersson, "Classification of German White Wines with Certified Brand of Origin by Multielement Quantitation and Pattern Recognition Techniques," *Journal of Agricultural and Food Chemistry,* vol. 52, pp. 2962-2974, 5 2004.

[120] P. P. Coetzee, F. E. Steffens, R. J. Eiselen, O. P. Augustyn, L. Balcaen and F. Vanhaecke, "Multi-element Analysis of South African Wines by ICP-MS and Their Classification According to Geographical Origin," *Journal of Agricultural and Food Chemistry,* vol. 53, pp. 5060-5066, 6 2005.

[121] F. J. Acevedo, J. Jiménez, S. Maldonado, E. Domínguez and A. Narváez, "Classification of Wines Produced in Specific Regions by UV-Visible Spectroscopy Combined with Support Vector Machines," *Journal of Agricultural and Food Chemistry,* vol. 55, pp. 6842-6849, 8 2007.

[122] N. H. Beltrán, M. A. Duarte-Mermoud, S. A. Salah, M. A. Bustos, A. I. Peña-Neira, E. A. Loyola and J. W. Jalocha, "Feature selection algorithms using Chilean wine chromatograms as examples," *Journal of Food Engineering,* vol. 67, pp. 483-490, 4 2005.

[123] N. H. Beltrán, M. A. Duarte-Mermoud, M. A. Bustos, S. A. Salah, E. A. Loyola, A. I. Peña-Neira and J. W. Jalocha, "Feature extraction and classification of Chilean wines," *Journal of Food Engineering,* vol. 75, pp. 1-10, 7 2006.

[124] N. H. Beltrán, M. A. Duarte-Mermoud, V. A. S. Vicencio, S. A. Salah and M. A. Bustos, "Chilean Wine Classification Using Volatile Organic Compounds Data Obtained With a Fast GC Analyzer," *IEEE Transactions on Instrumentation and Measurement,* vol. 57, pp. 2421-2436, 2008.

[125] P. L. Pisano, M. F. Silva and A. C. Olivieri, "Exploration of liquid chromatographic-diode array data for Argentinean wines by extended multivariate curve resolution," *Chemometrics and Intelligent Laboratory Systems,* vol. 132, pp. 1-7, 3 2014.

[126] P. L. Pisano, M. F. Silva and A. C. Olivieri, "Anthocyanins as markers for the classification of Argentinean wines according to botanical and geographical origin. Chemometric modeling of liquid chromatography–mass spectrometry data," *Food Chemistry,* vol. 175, pp. 174-180, 5 2015.

[127] J. D. Nunes-Miranda, H. M. Santos, M. Reboiro-Jato, F. Fdez-Riverola, G. Igrejas, C. Lodeiro and J. L. Capelo, "Direct matrix assisted laser desorption ionization mass spectrometry-based analysis of wine as a powerful tool for classification purposes," *Talanta,* vol. 91, pp. 72-76, 2012.

[128] M. Gaeta, M. Marsella, S. Miranda and S. Salerno, "Using neural networks for wine identification," in *Proceedings. IEEE International Joint Symposia on Intelligence and Systems (Cat. No.98EX174)*, 1998.

[129] H. Yu, H. Lin, H. Xu, Y. Ying, B. Li and X. Pan, "Prediction of Enological Parameters and Discrimination of Rice Wine Age Using Least-Squares Support Vector Machines and Near

Infrared Spectroscopy," *Journal of Agricultural and Food Chemistry,* vol. 56, pp. 307-313, 1 2008.

[130] N. L. Costa, L. A. G. Llobodanin, M. D. Lima, I. A. Castro and R. Barbosa, "Geographical recognition of Syrah wines by combining feature selection with Extreme Learning Machine," *Measurement,* vol. 120, pp. 92-99, 2018.

[131] I. Sen and F. Tokatli, "Differentiation of wines with the use of combined data of UV–visible spectra and color characteristics," *Journal of Food Composition and Analysis,* vol. 45, pp. 101-107, 2 2016.

[132] S. Gómez-Meire, C. Campos, E. Falqué, F. Díaz and F. Fdez-Riverola, "Assuring the authenticity of northwest Spain white wine varieties using machine learning techniques," *Food Research International,* vol. 60, pp. 230-240, 2014.

[133] M. J. Cabrita, J. A. De-Sousa, M. D. R. G. D. Silva, F. Rei and A. M. C. Freitas, "Multivariate statistical approaches for wine classification based on low molecular weight phenolic compounds," *Australian Journal of Grape and Wine Research,* vol. 18, pp. 138-146, 4 2012.

[134] L. Gutiérrez, F. A. Quintana, D. Baer and C. Mardones, "Multivariate Bayesian discrimination for varietal authentication of Chilean red wine," *Journal of Applied Statistics,* vol. 38, pp. 2099-2109, 10 2011.

[135] D. Kruzlicova, J. Mocak, B. Balla, J. Petka, M. Farkova and J. Havel, "Classification of Slovak white wines using artificial neural networks and discriminant techniques," *Food Chemistry,* vol. 112, pp. 1046-1052, 2 2009.

[136] R. Tauler, "Multivariate curve resolution applied to second order data," *Chemometrics and Intelligent Laboratory Systems,* vol. 30, pp. 133-146, 11 1995.

[137] J. A. Arancibia, C. E. Boschetti, A. C. Olivieri and G. M. Escandar, "Screening of Oil Samples on the Basis of Excitation-Emission Room-Temperature Phosphorescence Data and Multiway Chemometric Techniques. Introducing the Second-Order Advantage in a Classification Study," *Analytical Chemistry,* vol. 80, pp. 2789-2798, 3 2008.

[138] I. Arozarena, A. Casp, R. Marín and M. Navarro, "Differentiation of some Spanish wines according to variety and region based on their anthocyanin composition," *European Food Research and Technology,* vol. 212, pp. 108-112, 12 2000.

[139] G. González-Neves, J. Franco, L. Barreiro, G. Gil, M. Moutounet and A. Carbonneau, "Varietal differentiation of Tannat, Cabernet-Sauvignon and Merlot grapes and wines according to their anthocyanic composition," *European Food Research and Technology,* vol. 225, pp. 111-117, 6 2006.

[140] D. P. Makris, S. Kallithraka and A. Mamalos, "Differentiation of young red wines based on cultivar and geographical origin with application of chemometrics of principal polyphenolic constituents," *Talanta,* vol. 70, pp. 1143-1152, 12 2006.

[141] M. L. Gonzalez-San Jose, G. Santa-Maria and C. Diez, "Anthocyanins as parameters for differentiating wines by grape variety, wine-growing region, and wine-making methods," *Journal of Food Composition and Analysis,* vol. 3, pp. 54-66, 3 1990.

[142] J. Saurina, "Characterization of wines using compositional profiles and chemometrics," *TrAC Trends in Analytical Chemistry,* vol. 29, pp. 234-245, 3 2010.

[143] L. G. Llobodanin, L. P. Barroso and I. A. Castro, "Prediction of the functionality of young S outh A merican red wines based on chemical parameters," *Australian Journal of Grape and Wine Research,* vol. 20, pp. 15-24, 12 2013.

[144] J. N. K. Rao and A. J. Scott, "The analysis of categorical data from complex sample surveys: chi-squared tests for goodness of fit and independence in two-way tables," *Journal of the American statistical association,* vol. 76, pp. 221-230, 1981.

[145] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing,* vol. 70, pp. 489-501, 12 2006.

[146] G. Astray, J. X. Castillo, J. A. Ferreiro-Lage, J. F. Gálvez and J. C. Mejuto, "Artificial neural networks: a promising tool to evaluate the authenticity of wine Redes neuronales: una herramienta prometedora para evaluar la autenticidad del vino," *CyTA - Journal of Food,* vol. 8, pp. 79-86, 5 2010.

[147] J. Lozano, T. Arroyo, J. P. Santos, J. M. Cabellos and M. C. Horrillo, "Electronic nose for wine ageing detection," *Sensors and Actuators B: Chemical,* vol. 133, pp. 180-186, 7 2008.

[148] M. C. Horrillo, J. Getino, J. Gutiérrez, L. Arés, J. I. Robla, C. Garcia and I. Sayago, "Measurements of VOCs in soils through a tin oxide multisensor system," *Sensors and Actuators B: Chemical,* vol. 43, pp. 193-199, 9 1997.

[149] A. Yamazaki, T. B. Ludermir and M. C. P. Souto, "Classification of vintages of wine by artificial nose using time delay neural networks," *Electronics Letters,* vol. 37, pp. 1466-1467, 2001.

[150] M. S. Santos, "Construction of an artificial nose using neural networks," 2000.

[151] J. Aires-De-Sousa, "Verifying Wine Origin: A Neural Network Approach," *American Journal of Enology and Viticulture,* vol. 47, pp. 410-414, 1996.

[152] P. Etiévant, P. Schlich, J.-C. Bouvier, P. Symonds and A. Bertrand, "Varietal and geographic classification of French red wines in terms of elements, amino acids and aromatic alcohols," *Journal of the Science of Food and Agriculture,* vol. 45, pp. 25-41, 1988.

[153] F. Zamora Marín, Elaboración y crianza del vino tinto. Aspectos científicos y prácticos, Mundi Prensa Libros, 2003.

[154] J.-M. Souquet, B. Labarbe, C. Le Guernevé, V. Cheynier and M. Moutounet, "Phenolic composition of grape stems," *Journal of Agricultural and Food Chemistry,* vol. 48, pp. 1076-1080, 2000.

[155] J.-J. Macheix, A. Fleuriet and J. Billot, Fruit Phenolics, CRC Press, 2018.

[156] A. Castañeda-Ovando, M. Lourdes Pacheco-Hernández, M. E. Páez-Hernández, J. A. Rodríguez and C. A. Galán-Vidal, "Chemical studies of anthocyanins: A review," *Food chemistry,* vol. 113, pp. 859-871, 2009.

[157] I. Fernandes, A. Faria, C. Calhau, V. Freitas and N. Mateus, "Bioavailability of anthocyanins and derivatives," *Journal of functional foods,* vol. 7, pp. 54-66, 2014.

[158] R. L. Jackman, R. Y. Yada, T. U. N. G. MARVIN A and R. A. L. E. X. SPEERS, "Anthocyanins as food colorants—a review," *Journal of food biochemistry,* vol. 11, pp. 201-247, 1987.

[159] F. Chinnici, F. Sonni, N. Natali, S. Galassi and C. Riponi, "Colour features and pigment composition of Italian carbonic macerated red wines," *Food Chemistry,* vol. 113, pp. 651-657, 2009.

[160] I. Revilla, S. Pérez-Magariño, M. L. González-SanJosé and S. Beltrán, "Identification of anthocyanin derivatives in grape skin extracts and red wines by liquid chromatography with diode array and mass spectrometric detection," *Journal of Chromatography A,* vol. 847, pp. 83-90, 1999.

[161] A. Miele, L. A. Rizzon and M. C. Zanus, "Discrimination of Brazilian red wines according to the viticultural region, varietal, and winery origin," *Ciência e Tecnologia de Alimentos,* vol. 30, pp. 268-275, 3 2010.

[162] S. R. Segade, I. Orriols, V. Gerbi and L. Rolle, "Phenolic characterization of thirteen red grape cultivars from Galicia by anthocyanin profile and flavanol composition," *J. Int. Sci. Vigne Vin,* vol. 43, pp. 189-198, 2009.

[163] S. Remy, H. Fulcrand, B. Labarbe, V. Cheynier and M. Moutounet, "First confirmation in red wine of products resulting from direct anthocyanin--tannin reactions," *Journal of the Science of Food and Agriculture,* vol. 80, pp. 745-751, 2000.

[164] P. Markakis, Anthocyanins as food colors, Elsevier, 2012.

[165] C. M. Willemse, M. A. Stander, J. Vestner, A. G. J. Tredoux and A. Villiers, "Comprehensive two-dimensional hydrophilic interaction chromatography (HILIC)× reversed-phase liquid chromatography coupled to high-resolution mass spectrometry (RP-LC-UV-MS) analysis of anthocyanins and derived pigments in red wine," *Analytical chemistry,* vol. 87, pp. 12006-12015, 2015.

[166] A. Marquez, M. P. Serratosa and J. Merida, "Influence of bottle storage time on colour, phenolic composition and sensory properties of sweet red wines," *Food chemistry,* vol. 146, pp. 507-514, 2014.

[167] A. Delcambre and C. Saucier, "Identification of new flavan-3-ol monoglycosides by UHPLC-ESI-Q-TOF in grapes and wine," *Journal of Mass Spectrometry,* vol. 47, pp. 727-736, 2012.

[168] C. Saucier, M. Jourdes, Y. Glories and S. Quideau, "Extraction, detection, and quantification of flavano-ellagitannins and ethylvescalagin in a Bordeaux red wine aged in oak barrels," *Journal of Agricultural and Food Chemistry,* vol. 54, pp. 7349-7354, 2006.

[169] I. Jarauta, J. Cacho and V. Ferreira, "Concurrent phenomena contributing to the formation of the aroma of wine during aging in oak wood: An analytical study," *Journal of agricultural and food chemistry,* vol. 53, pp. 4166-4177, 2005.

[170] I. García-Estévez, C. Alcalde-Eon, V. Puente and M. T. Escribano-Bailón, "Enological tannin effect on red wine color and pigment composition and relevance of the yeast fermentation products," *Molecules,* vol. 22, p. 2046, 2017.

[171] I. García-Estévez, M. T. Escribano-Bailón, J. C. Rivas-Gonzalo and C. Alcalde-Eon, "Development of a fractionation method for the detection and identification of oak ellagitannins in red wines," *Analytica chimica acta,* vol. 660, pp. 171-176, 2010.

[172] A. Versari, R. B. Boulton and G. P. Parpinello, "A comparison of analytical methods for measuring the color components of red wines," *Food Chemistry,* vol. 106, pp. 397-402, 2008.

[173] V. Sáez, C. Gayoso, S. Riquelme, J. Pérez, C. Vergara, C. Mardones and D. Baer, "C18 core-shell column with in-series absorbance and fluorescence detection for simultaneous monitoring of changes in stilbenoid and proanthocyanidin concentrations during grape cane storage," *Journal of Chromatography B,* vol. 1074, pp. 70-78, 2018.

[174] L. F. Ribeiro, R. H. Ribani, T. M. G. Francisco, A. A. Soares, R. Pontarolo and C. W. I. Haminiuk, "Profile of bioactive compounds from grape pomace (Vitis vinifera and Vitis labrusca) by spectrophotometric, chromatographic and spectral analyses," *Journal of chromatography B,* vol. 1007, pp. 72-80, 2015.

[175] HPLC Química, "HPLC Química," [Online]. Available: https://www.cromatografia.com.mx/. [Accessed 30 Agosto 2021].

[176] V. Di Stefano, G. Avellone, D. Bongiorno, V. Cunsolo, V. Muccilli, S. Sforza, A. Dossena, L. Drahos and K. Vékey, "Applications of liquid chromatography--mass spectrometry for food analysis," *Journal of Chromatography A,* vol. 1259, pp. 74-85, 2012.

[177] R. Flamini, "Mass spectrometry in grape and wine chemistry. Part I: Polyphenols," *Mass spectrometry reviews,* vol. 22, pp. 218-250, 2003.

[178] M. Ji, C. Li and Q. Li, "Rapid separation and identification of phenolics in crude red grape skin extracts by high performance liquid chromatography coupled to diode array detection and tandem mass spectrometry," *Journal of Chromatography A,* vol. 1414, pp. 138-146, 2015.

[179] S.-Y. Li, F. He, B.-Q. Zhu, R.-R. Xing, M. J. Reeves and C.-Q. Duan, "A systematic analysis strategy for accurate detection of anthocyanin pigments in red wines," *Rapid Communications in Mass Spectrometry,* vol. 30, pp. 1619-1626, 2016.

[180] M. B. S. M. Ilarduya, "Pigmentos derivados antociánicos de los vinos tintos de Rioja. Estudio analítico, influencia en el color y evolución durante la crianza," 2010.

[181] C. Sánchez-Fernández, "Búsqueda de marcadores de tipo tanino en vinos tintos de rioja: estudio cualitativo y cuantitativo por hplc-ms/ms," 2012.

[182] Z. Rasines-Perea, N. Prieto-Perea, M. Romera-Fernández, L. A. Berrueta and B. Gallo, "Fast determination of anthocyanins in red grape musts by Fourier transform mid-infrared spectroscopy and partial least squares regression," *European Food Research and Technology,* vol. 240, pp. 897-908, 1 2015.

[183] C. S. Fernandez, "Búsqueda de marcadores de tipo tanino en vinos tintos de rioja: estudio cualitativo y cuantitativo por hplc-ms/ms," 2012.

[184] L. Barreiro, D. Charamelo and G. Gonzalez-Neves, "Perfil antociánico y composición fenólica de vinos Tannat elaborados con adición de enzimas pectolíticas.," *Revista de la Facultad de Ciencias Agrarias,* vol. 37, pp. 9-18, 1 2006.

[185] J. P. Roggero, J. L. Larice, C. Rocheville-Divorne, P. Archier and S. Coen, "Anthocyanin composition of grapevine varieties. 1. Attempt of classification by principal components analysis and by factorial discriminant analysis," *Revue Francaise d'Oenologie (France),* 1988.

[186] J. Bakker and C. F. Timberlake, "Isolation, identification, and characterization of new color-stable anthocyanins occurring in some red wines," *Journal of agricultural and food chemistry,* vol. 45, pp. 35-43, 1997.

[187] C. Dallas, J. M. Ricardo-da-Silva and O. Laureano, "Degradation of oligomeric procyanidins and anthocyanins in a Tinta Roriz red wine during maturation," *Vitis,* vol. 34, pp. 51-56, 1995.

[188] H. Fulcrand, C. Benabdeljalil, J. Rigaud, V. Cheynier and M. Moutounet, "A new class of wine pigments generated by reaction between pyruvic acid and grape anthocyanins," *Phytochemistry,* vol. 47, pp. 1401-1407, 4 1998.

[189] C. Romero and J. Bakker, "Effect of Storage Temperature and Pyruvate on Kinetics of Anthocyanin Degradation, Vitisin A Derivative Formation, and Color Characteristics of Model Solutions," *Journal of Agricultural and Food Chemistry,* vol. 48, pp. 2135-2141, 5 2000.

[190] M. Monagas, C. Gómez-Cordovés, B. Bartolomé, O. Laureano and J. M. Ricardo da Silva, "Monomeric, Oligomeric, and Polymeric Flavan-3-ol Composition of Wines and Grapes from Vitis vinifera L. Cv. Graciano, Tempranillo, and Cabernet Sauvignon," *Journal of Agricultural and Food Chemistry,* vol. 51, pp. 6475-6481, 2003.

[191] A. g. OIV, "RESOLUCIÓN OIV/CONCOURS 332A/2009. NORMA OIV DE LOS CONCURSOS INTERNACIONALES DE VINOS Y BEBIDAS," Zagreb, 2009.

[192] F. Pérez-Elortondo, I. Etaio-Alonso, M. Albisu-Aguado, J. Salmerón-Egea, M. Ojeda-Atxiaga and E. Gastón-Estanga, Guía para la evaluación sensorial de la calidad de los vinos

tintos de Rioja Alavesa, 1 ed., Vitoria-Gasteiz: Servicio Central de Publicaciones del Gobierno Vasco, 2007.

[193] vinetur, "vinetur.com," 2016. [Online]. Available: https://www.vinetur.com/2016071424716/claves-para-descubrir-la-edad-de-un-vino-por-su-color.html. [Accessed 22 Diciembre 2016].

[194] P. Benito-Sáez, "urbinavinos.blogspot.com," 2013. [Online]. Available: http://urbinavinos.blogspot.com.es/2013/01/los-colores-del-vino.html. [Accessed 22 Diciembre 2016].

[195] C. Marangoni, Sull'espansione delle goccie d'un liquido galleggianti sulla superfice di altro liquido, Fratelli Fusi, 1865.

[196] J. Jauregui, "Principal component analysis with linear algebra," *Philadelphia: Penn Arts & Sciences,* 2012.

[197] M. Rosenblatt, "Remarks on Some Nonparametric Estimates of a Density Function," *The Annals of Mathematical Statistics,* vol. 27, pp. 832-837, 9 1956.

[198] F. J. Massey Jr, "The Kolmogorov-Smirnov test for goodness of fit," *Journal of the American statistical Association,* vol. 46, pp. 68-78, 1951.

[199] J. J. Buckley and Y. Hayashi, "Fuzzy neural networks: A survey," *Fuzzy sets and systems,* vol. 66, pp. 1-13, 1994.

[200] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier and A. Maida, "Deep learning in spiking neural networks," *Neural networks,* vol. 111, pp. 47-63, 2019.

[201] N. Kasabov, V. Feigin, Z.-G. Hou, Y. Chen, L. Liang, R. Krishnamurthi, M. Othman and P. Parmar, "Evolving spiking neural networks for personalised modelling, classification and prediction of spatio-temporal patterns with a case study on stroke," *Neurocomputing,* vol. 134, pp. 269-279, 6 2014.

[202] Ministerio de Agricultura, Pesca y Alimentación, "Vitivinicultura: Datos INFOVI," [Online]. Available: https://www.mapa.gob.es/es/agricultura/temas/producciones-agricolas/vitivinicultura/default.aspx. [Accessed 2 Febrero 2020].

[203] A. C. Noble, R. A. Arnold, J. Buechsenstein, E. J. Leach, J. O. Schmidt and P. M. Stern, "Modification of a Standardized System of Wine Aroma Terminology," *American Journal of Enology and Viticulture,* vol. 38, pp. 143-146, 1987.

[204] S. Frías, J. E. Conde, M. A. Rodríguez, V. Dohnal and J. P. Pérez-Trujillo, "Metallic content of wines from the Canary Islands (Spain). Application of artificial neural networks to the data analysis," *Food / Nahrung,* vol. 46, pp. 370-375, 2002.

[205] H. Fulcrand, C. Benabdeljalil, J. Rigaud, V. Cheynier and M. Moutounet, "A new class of wine pigments generated by reaction between pyruvic acid and grape anthocyanins," *Phytochemistry,* vol. 47, pp. 1401-1407, 1998.

[206] A. Morata, F. Calderón, M. C. González, M. C. Gómez-Cordovés and J. A. Suárez, "Formation of the highly stable pyranoanthocyanins (vitisins A and B) in red wines by the addition of pyruvic acid and acetaldehyde," *Food Chemistry,* vol. 100, pp. 1144-1152, 2007.

[207] C. Escott, A. Morata, I. Loira, W. Tesfaye and J. A. Suarez-Lepe, "Characterization of polymeric pigments and pyranoanthocyanins formed in microfermentations of non-Saccharomyces yeasts," *Journal of applied microbiology,* vol. 121, pp. 1346-1356, 2016.

[208] P. X. Etievant, "Volatile phenol determination in wine," *Journal of Agricultural and Food Chemistry,* vol. 29, pp. 65-67, 1981.

[209] M. F. N. Bourden, N. Vivas, C. Absalon, C. Vitry, E. Fouquet and N. V. De Gaulejac, "Structural diversity of nucleophilic adducts from flavanols and oak wood aldehydes," *Food chemistry,* vol. 107, pp. 1494-1505, 2008.

[210] N.-E. Es-Safi, V. Cheynier and M. Moutounet, "Interactions between cyanidin 3-O-glucoside and furfural derivatives and their impact on food color changes," *Journal of agricultural and food chemistry,* vol. 50, pp. 5586-5595, 2002.

[211] M. De Rosso, A. Panighel, A. Dalla Vedova, L. Stella and R. Flamini, "Changes in chemical composition of a red wine aged in acacia, cherry, chestnut, mulberry, and oak wood barrels," *Journal of Agricultural and Food Chemistry,* vol. 57, pp. 1915-1920, 2009.

[212] J. Drinkine, P. Lopes, J. A. Kennedy, P.-L. Teissedre and C. Saucier, "Ethylidene-bridged flavan-3-ols in red wine and correlation with wine age," *Journal of Agricultural and Food Chemistry,* vol. 55, pp. 6292-6299, 2007.

[213] M. Yeo, T. Fletcher and J. Shawe-Taylor, "Machine Learning in Fine Wine Price Prediction," *Journal of Wine Economics,* vol. 10, p. 151–172, 2015.

Annexes

10 Annexes

Details of demonstrations on specific aspects of CART have been included in this section so as not to divert the reader's attention from the main ideas of this technique.

10.1 Annex I: Towing Rule

Once a certain division has been fixed, p_L, p_R will be fixed. We are interested in finding that grouping in superclasses that maximizes Δi as if it was a problem of separation into two superclasses. The impurity function used in the new binary problem by Breiman et al. [1] is a proportional variation of the Gini index. Thus, we would have that

$$\Delta\mathrm{i}\big(s,t,C_1(s)\big) = P(C_1|t)P(C_2|t) - p_L P(C_1|t_L)P(C_2|t_L) - p_R P(C_1|t_R)P(C_2|t_R) \tag{119}$$

Substituting $P(C_2|k) = 1 - P(C_1|k)$ we have

$$\Delta i\big(s,t,C_1(s)\big) = P(C_1|t) - P^2(C_1|t) - p_L\, P(C_1|t_L) + p_L P^2(C_1|t_L) - p_R P(C_1|t_R) + p_I P^2(C_1|t_R) \tag{120}$$

As $p_L P(C_1|t_L) + p_R P(C_1|t_R) = p(C_1|t)$,

$$\Delta i\big(s,t,C_1(s)\big) = p_L P^2(C_1|t_L) + p_R P^2(C_1|t_R) - \big(p_L P(C_1|t_L) + p_R P(C_1|t_R)\big)^2 \tag{121}$$

$$\Delta i\big(s,t,C_1(s)\big) = p_L P^2(C_1|t_L) - p_L^2 P^2(C_1|t_L) + p_R P^2(C_1|t_R) - p_L^2 P^2(C_1|t_R) - 2p_L p_R P(C_1|t_L)P(C_1|t_R) \tag{122}$$

$$\Delta i\big(s,t,C_1(s)\big) = p_L \underbrace{(1-p_L)}_{p_R} P^2(C_1|t_L) + p_R \underbrace{(1-p_R)}_{p_L} P^2(C_1|t_R) - 2p_L p_R P(C_1|t_L)P(C_1|t_R) \tag{123}$$

$$\Delta i\big(s,t,C_1(s)\big) = p_L p_R\big(P(C_1|t_L) - P(C_1|t_R)\big)^2 \tag{124}$$

By grouping the classes in C_1, C_2 we have that $P(C_1|k) = \sum_{C_j \in C_1} P\big(C_j\big|k\big)$ and with the rest of the classes that have not been grouped into C_1, $P(C_2|k) = \sum_{C_j \in C_2} P\big(C_j\big|k\big)$.

Each class C_j we include in C_1 will contribute something to Δi. If we call each one of these possible contributions $z_j = P\big(C_j\big|t_L\big) - P\big(C_j\big|t_R\big)$. We seek to maximize or minimize $\sum_{C_j \in C_1} z_j$ to maximize the later power-of-two operation.

The maximum value will be reached by the following procedure: If z_j is positive, because $P\big(C_j\big|t_L\big) > P\big(C_j\big|t_R\big)$, we include C_j in C_1 and it will always contribute something positive. If that z_j is negative, we will not include it (and consequently it will be part of C_2). Let's call z_j^+ these contributions that will take the value 0 if z_j is negative.

The minimum value will be reached by an analogous procedure: If z_j is negative, because $P\big(C_j\big|t_L\big) < P\big(C_j\big|t_R\big)$, we include C_j in C_1 and will always contribute something negative. If that z_j is positive, we will not include it (and consequently it will be part of C_2). Let us call z_j^- these contributions (counted in absolute value, counted as positive numbers) which will also take the value 0 if z_j is positive.

Those $z_j = 0$ can be indistinctly included in C_1 or C_2.

One of the properties of the set of z_j is that $\sum_j z_j = \sum_j P(C_j|t_L) - P(C_j|t_R) = 1 - 1 = 0$. On the other hand, each z_j will always be either positive or negative, and will be counted either in z_j^+ or in z_j^-. The sum of its values $\sum_j z_j$ will be the sum of all the positive contributions minus the sum of all the negative contributions (the null ones do not matter), that is, $\sum_j z_j = \sum_j z_j^+ - \sum_j z_j^-$.

Putting both properties together we have that $\sum_j z_j^+ = \sum_j z_j^-$. That is, the maximum and minimum values coincide. Therefore, the maximum value can be expressed as half of its sum, as absolute values. That is to say,

$$\max_{C_j \in C_1} \sum_j z_j = \sum_j z_j^+ = \frac{\sum_j z_j^+ + \sum_j z_j^+}{2} = \frac{\sum_j z_j^+ + \sum_j z_j^-}{2} \tag{125}$$

As we have stated, if z_j is positive $|z_j| = z_j^+$ and $z_j^- = 0$. Similarly, if z_j is negative $|z_j| = z_j^-$ and $z_j^+ = 0$. This allows us to write the equation (125) as

$$\max_{j \in C_1} \sum_j z_j = \frac{\sum_j z_j^+ + \sum_j z_j^-}{2} = \frac{\sum_j |z_j|}{2} \tag{126}$$

Substituting in $\Delta i(s, t, C_1(s)) = p_L p_R \left(p(C_1|t_L) - P(C_1|t_R)\right)^2$ we have

$$\max_{C_1} \Delta i(s, t, C_1(s)) = p_I p_D \left(\frac{\sum_{C_j} |p(C_j|t_I) - p(C_j|t_D)|}{2} \right)^2$$

where we have ended up including in the class C_1 those classes where $P(C_j|t_L) > P(C_j|t_R)$.

The binarism rule can be perfectly well applied to categorical features by first looking for all possible subsets of questions that are created around a categorical feature, and then calculating for each question the best superclass (consisting of those classes where $P(C_j|t_L) > P(C_j|t_R)$) and its value $\Delta \mathrm{i} = p_L p_R \left(P(C_1|t_L) - P(C_1|t_R)\right)^2$. The one that obtains the highest value among all the Δi will be the best division.

To take advantage of the inherent binarism of twoing rule and to be able to search linearly on categorical features (instead of trying all possible subsets), we can modify the way to proceed and instead of setting the split first and then searching for the best superclass, we can do it in opposite order. The search is performed by first creating all possible C_1, then looking for each C_1 the sorting $P(C_1|x = b_l)$ that creates, and finally searching linearly in the sorting. For example, let's have a categorical feature $x_1 = \{red, green, blue, yellow, grey, purple, black, white\}$ and 4 possible classes in which classify the problem. The superclasses that can be formed, are $2^4 = 16$. Let's take one, C_1 formed with classes 1, 2 and 4. Now let's calculate the probability of $p(C_1|x = red)$, $p(C_1|x = green)$,..., $p(C_1|x = white)$ and let's order them from smallest to largest. Finally, let's search which of the divisions in that ordering offers a greater $\Delta i = p_L p_R \left(P(C_1|t_L) - P(C_1|t_R)\right)^2$. For each of the 2^{n_c} possible superclasses we have L probability calculations $p(C_1|x = x_i)$. We must order them to then form the $2^{n_c}L$ divisions where to evaluate Δi and obtain the largest. If we did not do it this way, we would have to calculate all the possible 2^L subsets in which to make the division of the categorical feature and then for each one of them to evaluate n_c times $P(C_j|t_L) > P(C_j|t_R)$ to obtain the superclass C_1 and its corresponding Δi, and then compare them with each other and obtain the largest Δi.

In the first way the operations grow with $L \cdot 2^{n_c}$, exponentially to the number of classes. In the second, they grow with $n_c \cdot 2^L$, exponentially to the number of categories. Which of the two forms offers advantages when computed in one way or the other? The answer depends on whether the number of classes is greater than the number of categories or vice versa.

10.2 Annex II: Optimized search on categorical features

For a categorical feature which can take the set of values $\{b_1, b_2, \dots, b_L\}$ the possible questions will be of the form "$¿x \in S$?" where S are the subsets formed with those categories of the form $\bigcup b_i$. Thus, a subset to be analyzed can be $b_1 \cup b_2 \cup b_6$. Another could be $b_2 \cup b_4$. If we have L possible categories we can form $2^L - 2$ subsets to test. The 2 that we subtract is due to the fact that we will neither analyze the divisions $b_1 \cup b_2 \cup \dots \cup b_L$ nor $\emptyset$ because they are meaningless.

Among these proposed divisions, we will be interested in looking for the one that maximizes $\Delta i(s,t)$. If we have a division into two classes and a single categorical feature, we can estimate the probability that, at a node t, an example belongs to a class knowing the category of that feature. That is, we can estimate

$$P(C_j|x = b_i) = \frac{P(x = b_l|C_j)P(C_j)}{\sum_j P(x = b_l|C_j)P(C_j)} \tag{127}$$

where in turn an estimate of $P(C_j)$ can be obtained directly from the data itself, and $P(x = b_l|C_j) = N_{j,l}(t)/N_j(t)$ (is the number of cases in a node t that are of the class C_j with the category b_l divided by the number of cases of class C_j at that same node).

If we order these estimated probabilities for the first class we will have

$$P(C_1|b_{l_1}) \leq P(C_1|b_{l_2}) \leq \cdots \leq P(C_1|x = b_{l_L}) \tag{128}$$

The management of the second class will be complementary, that is,

$$P(C_2|b_{l_L}) \leq P(C_2|b_{l_{L-1}}) \leq \cdots \leq P(C_2|x = b_{l_1}) \tag{129}$$

Since the best division tries to obtain the best probability, the most efficient questions are then reduced to those that divide the ordination in two, cause the highest values will be obtained as a combination of the categories that provide a higher probability and leaving aside those that provide less. That is to say, $S_1 = b_{l_L}$, $S_2 = b_{l_L} \cup b_{l_{L-1}}$,…, $S_{L-1} = b_{l_L} \cup b_{l_{L-1}} \cup \dots \cup b_{l_2}$. With this we will have reduced the search for $2^L - 2$ possible subsets to $L - 1$ possible subsets.

What do we do if there are two different categorical features and the ordering of each of them is different? It is not a problem, since the divisions are not raised on combination of categorical features, but on each of them in isolation and independently. In case of divisions on the feature x_1 we search in its ordering, in case of divisions over x_2 we search in the other ordering, and so on.

What do we do if we have more than two classes and the sorting in the different classes is not the same or complementary? We would necessarily have to look at all possible 2^{L-1} subsets one by one and the method would not be applicable.

10.3 Annex III: Nesting of trees produced by pruning based on $R_\alpha(T)$ measure

In any division t there is always an improvement in the error rate, that is, any division produces $R(t) \geq R(t_L) + R(t_R)$. Trying to minimize $R_\alpha(T)$ we will first prune the divisions that do not introduce $\Delta R(t)$. We thus eliminate the equality in $R(t) \geq R(t_L) + R(t_R)$ and this guarantees that for any non-terminal t, $R(t) > R(t_L) + R(t_R) = R(T_t)$.

For each value of α there is a tree $T(\alpha)$ that minimizes $R_\alpha\big(T(\alpha)\big)$. Suppose a multilevel branch with multiple levels and $\left|\tilde{T}\right|$ terminal nodes deriving from a node t.

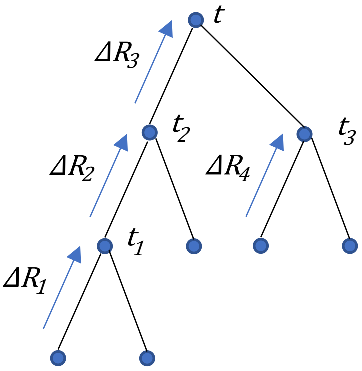


Figure 103. Example branch for pruning.

Its $R_\alpha(T_t) = R(T_t) + \alpha\left|\tilde{T}\right|$. If we prune leaving a single node $R_\alpha(t) = R(t) + \alpha \cdot 1$, then it would have only one terminal node, the t node itself.

Since we know in advance that $R(t) > R(T_t)$ we will have that for lower α values $R_\alpha(t) > R_\alpha(T_t)$. As α increases, $R_\alpha(T_t)$ will go up and approach $R_\alpha(t)$ which will also go up, but less. This is due to the factor $\alpha\left|\tilde{T}\right|$. There will come an α value when both rates will equalize, that will happen at $\alpha_{limit} = \frac{R(t)-R(T_t)}{|\tilde{T}|-1}$. For values greater than α_{limit} we will have that the inequality is inverted, obtaining $R_\alpha(T_t) > R_\alpha(t)$.

For each non-terminal node in the tree, we can calculate its

$$g(t) = \frac{R(t) - R(T_t)}{\left|\tilde{T}\right| - 1}$$

where $R(T_t)$ is the sum of the different $R(t)$-s of the terminal nodes of the branches below it, $\left|\tilde{T}\right|$ is the number of terminal nodes below it. This value depends only on the ΔR and the number of terminal nodes at each level. By gradually increasing α, the tree will remain the same (unpruned) until α reaches the next $\min_i g(t_i)$ at which point the tree will be pruned by the next node, t_i. If this situation were to occur at two nodes at the same time, with the same value of $g(t_i)$, both nodes would be pruned.

Clarify that in this process it is possible to prune several levels at once, in a non-terminal level of the hierarchy, because if the node of $\min_i g(t_i)$ node is not at the last level, it will be necessary to prune several levels.

Is it possible that by increasing α the $T(\alpha)$ corresponding to $R_\alpha\big(T(\alpha)\big)$ is not nested? Is it possible that by incrementing the corresponding α the $T(\alpha)$ corresponding to $R_\alpha\big(T(\alpha)\big)$ is a tree with a branch not present in previous trees? No, because when going up α beyond the $g(t_i)$ that branch will be kept pruned, because if it is not pruned it would provide greater R_α. In other words, in order for a branch not to be pruned the $g(t_i)$ of the upper nodes must be higher than the current α.

That's equivalent to say that:

1. If $\alpha \geq g(t_i)$ all the lower branches from the node t_i are pruned.

2. If $\alpha < g(t_i)$ will not be pruned.

Using the previous example tree in Figure 103, let us first assume that $g(t_1) < g(t_2) < g(t_3)$. As α increases, we will go through the following phases:

1. $\alpha < g(t_1)$: No branches are pruned.
2. $g(t_1) < \alpha < g(t_2)$: One level is pruned. t_1 and its lower branches are pruned.
3. $g(t_2) < \alpha < g(t_3)$: Two levels are pruned on the left side (this includes the pruning of one level). t_2 and its lower branches are pruned.
4. $g(t_3) < \alpha$: Two levels are pruned on the left side and one on the right side. t_2 and their lower branches are pruned and t_3 and their lower branches.

Let's see how we would proceed if we assume this time that $g(t_2) < g(t_1) < g(t_3)$. This time, as the α increases, we will go through the following phases:

1. $\alpha < g(t_2)$: No branches are pruned.
2. $g(t_2) < \alpha < g(t_1)$: Two levels are pruned on the left side (this includes the pruning of one level). t_2 and its lower branches are pruned.
3. $g(t_1) < \alpha < g(t_3)$: Two levels are pruned on the left side (this includes the pruning of one level). t_2 and its lower branches are pruned. That is, there are no changes compared to the previous phase.
4. $g(t_3) < \alpha$: Two levels are pruned on the left side and one on the right side. t_2 and their lower branches are pruned and t_3 and their lower branches.

If you have already pruned up to one node because $\alpha \geq g(t_i)$ for a $\alpha' > \alpha$ will continue to be pruned, and therefore successive trees will be nested.

In addition, the solution thus obtained must be unique, because being nested, if for a given value of α there were two possible solutions, one should be nested in the other, necessarily implying that the latter is unique.

11 Glossary

(epi)cat. *Epicatequina.*

(epi)galocat. *Epigalocatequina.*

AAS. *Atomic Absorption Spectroscopy. Espectroscopia de Absorción Atómica.*

AES. *Atomic Emission Spectroscopy.*

ANN. *Artifcial Neural Networks. Redes Neuronales Artificiales.*

ANOVA. *Análisis de Varianza.*

API. *Atmospheric Pressure Ionization. Ionización a Presión Atmosférica.*

BC. *Bayesian Classifiers.*

CA. *Cluster Analysis.*

CART. *Classification and Regression Tree.*

Cat. *Catequina., Catequina.*

CE. *Electroforesis Capilar.*

CID. *Collision Induced Dissociation. Disociación por Colisión Inducida., Collision induced dissociation. Disociación inducida por colisión.*

Cy-3-glc. *Cyanidin-3-glucoside. Cianidina-3-glucósido., Cyanidin-3-glucoside. Cianidina-3-glucósido.*

DAD. *Diode Array Detector. Detector de Array de Diodos., Diode Array Detector. Detector de Array de Diodos.*

DI. *Diámetro Interno.*

DO. *Denominación de Origen.*

DOP. *Denominación de Origen Protegida.*

Dp-3-(6-Ac)-glc. *Delphinidin-3-(6-acetyl)-glucoside. Delfinidina-3-(6-acetil)-glucósido., Delphinidin-3-(6-acetyl)-glucoside. Delfinidina-3-(6-acetil)-glucósido.*

Dp-3-glc. *Delphinidin-3-glucoside. Delfinidina-3-glucósido., Delphinidin-3-glucoside. Delfinidina-3-glucósido.*

ELM. *Extreme Learning Machine. Máquina de Aprendizaje Extremo.*

ESI. *Electrospray Ionization. Ionización por Electrospray.*

FA. *Análisis de Factores.*

FAAS. *Flame Atomic Absorption Spectrophotometry. Espectrofotometría de Absorción Atómica de Llama.*

FAES. *Flame Atomic Emission Spectrophotometry. Espectrofotometría de Emisión Atómica de Llama.*

FID. *Flame Ionization Detector. Detector de Ionización de Llama.*

FNN. *Fuzzy Neural Networks. Redes Neuronales Difusas., Fuzzy Neural Networks. Redes Neuronales Difusas.*

FPD. *Flame Photometric Detector. Detector Fotométrico de Llama.*

FS. *Feature Scaling.*

FTIR. *Fourier Transform Infrared Spectroscopy. Espectroscopía de Infrarrojos por Transformada de Fourier.*

galocat. *Galocatequina.*

GC-MS. *Gas Chromatography coupled with Mass Spectroscopy.*

GFAAS. *Graphite furnace atomic absorption spectroscopy. Espectrometría de Absorción Atómica de Horno de Grafito*

HPLC. *High Perfomance Liquid Chromatography. Cromatografía Líquida de Alta Resolución.*

HPLC-MS. *High Pressure Liquid Chromatography coupled Mass Spectrometry. Cromatografía Líquida de Alta Resolución con acoplamiento de Espectrometro de Masas.*

ICA. *Independent Component Analysis.*

ICP-MS. *Inductively Coupled Plasma coupled Mass Spectrometry.*

ICP-OES. *Inductively Coupled Plasma Optical Emission Spectroscopy. Espectrometría de Emisión Atómica de Plasma Acoplado Inductivamente., Inductively Coupled Plasma Optical Emission Spectroscopy. Espectrometría de Emisión Atómica de Plasma Acoplado Inductivamente.*

kNN. *k Nearest Neighbor.*

LDA. Linear Discriminant Analysis.

LOO. *Leave-One-Out Cross Validation.*

LR. *Linear Regression.*

m/z. *Mass-charge ratio. Relación masa-carga.*

MALDI. *Matrix Assisted Laser Desorption/Ionization. Desorción/Ionización Láser Asistida por Matriz.*

MAP. *Maximum a Posteriori.*

ML. *Machine Learning.*

MLP. *Multilayer Perceptron., Multilayer Perceptron. Perceptrón Multicapa.*

MS. *Mass Spectrometry. Espectrometría de Masas.*

MS/MS. *Tandem Mass Spectrometry. Experimentos de Masas en Tándem.*

Mv-3-(6-Ac)-glc. *Malvidin-3-(6-acethyl)-glucoside. Malvidina -3-(6-acetil)-glucósido., Malvidin-3-(6-acethyl)-glucoside. Malvidina -3-(6-acetil)-glucósido.*

Mv-3-(6-caff)-glc. *Malvidin-3-(6-caffeoyl)-glucoside. Malvidina-3-(6-cafeoil)-glucósido., Malvidin-3-(6-caffeoyl)-glucoside. Malvidina-3-(6-cafeoil)-glucósido.*

Mv-3-(6-p-coum)-glc. *Malvidin-3-(6-p-coumaroyl)-glucoside. MAlvidina-3-(6-p-cumaroil)-glucósido., Malvidin-3-(6-p-coumaroyl)-glucoside. MAlvidina-3-(6-p-cumaroil)-glucósido.*

Mv-3-glc. *Malvidin-3-glucoside. Malvidina-3-glucósido., Malvidin-3-glucoside. Malvidina-3-glucósido.*

NB. *Naïbe Bayes Classifier.*

NIR. *Near-Infrared Spectroscopy.*

OIV. *La Organización Internacional del Vino y la Viña.*

OPLS-DA. *Orthogonal Partial Least Squares Discriminant Analysis. Regresión de Mínimos Cuadrados Parciales Ortogonales.*

PCA. *Principal Component Analysis.*

PCB1. *Prodelphinidin B1. Prodelfinidina B1.*

PCB2. *Prodelphinidin B2. Prodelfinidina B2., Prodelphinidin B2. Prodelfinidina B2.*

PCC1. *Procyanidin C1. Procianidina C1.*

PLS-DA. *Partial Least Squares Discriminant Analysis. Análisis Discriminante Mínimos Cuadrados Parciales.*

Pn-3-(6-Ac)-glc. *Peonidin-3-(6-acethyl)-glucoside. Peonidina-3-(6-acetil)-glucósido.*

Pn-3-(6-p-coum)-glc. *Peonidin-3-(6-p-coumaroyl)-glucoside. Peonidina-3-(6-p-cumaroil)-glucósido., Peonidin-3-(6-p-coumaroyl)-glucoside. Peonidina-3-(6-p-cumaroil)-glucósido.*

Pn-3-glc. *Peonidin-3-glucoside. Peonidina-3-glucósido., Peonidin-3-glucoside. Peonidina-3-glucósido.*

PNN. Probabilistic Neural Networks., *Probabilistic Neural Networs. Redes Neuronales Probabilísticas.*

Pt-3-(6-Ac)-glc. *Petunidin-3-(6-acethyl)-glucoside. Petunidina-3-(6-acetil)-glucósido.*

Pt-3-(6-p-coum)-glc. *Petunidin-3-(6-p-coumaroyl)-glucoside. Petunidina-3-(6-p-cumaroil)-glucósido., Petunidin-3-(6-p-coumaroyl)-glucoside. Petunidina-3-(6-p-cumaroil)-glucósido.*

Pt-3-glc. *Petunidin-3-glucoside. Petunidina-3-glucósido., Petunidin-3-glucoside. Petunidina-3-glucósido.*

QDA. *Quadratic Discriminant Analysis.*

QqQ. *Analizador Triple Quadripolo.*

RBFNN. *Radial Basis Function Neural Network. Redes Neuronales de Función de Base Radial.*

Relu. *Rectified Linear Unit. Unidad Lineal Rectificada.*

RF. *Random Forest.*

SAW. *Suface Acoustic Wave. Onda Acústica Superficial.*

SIMCA. *Soft Independent Modelling of Class Analogy. Modelado Suave Independiente por Analogía de Clases.*

SNN. *Spiking Neural Network. Redes Neuronales de Impulsos.*

SOM. *Self Organizing Maps. Mapas Autoorganizados.*

SPE. *Solid Phase Extraction. Extracción de Fase Sólida.*

SVM. *Support Vector Machine.*

TOF. *Time Of Flight. Tiempo de Vuelo.*

UV. *Ultra Violeta., Ultra Violeta.*

Abstract

This work presents and tests the feasibility of a methodology to make individualized recommendations for red wines of the Rioja Denomination of Origin in three stages. In the first one, it characterizes red wines by their content of anthocyanins, anthocyanin derivatives and tannins. In the second, it includes the guided evaluation of the different wines made by each taster using a standardized tasting sheet. In the third, it applies different machine learning techniques to build classifiers that produce meaningful wine recommendations. The validation of the classifiers demonstrates their ability to make negative recommendations of wines with a 100% success rate and positive recommendations of wines with a success rate of over 90%.