NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

MATILDE ALEXANDRA DAMAS LUCAS

Bachelor in Computer Science Engineering

SERS SPECTRA CLASSIFICATION USING DEEP LEARNING

PORTUGUESE WHITE WINE CLASSIFICATION USING SERS AND DEEP LEARNING

NOVA University Lisbon
July, 2023

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

SERS SPECTRA CLASSIFICATION USING DEEP LEARNING

PORTUGUESE WHITE WINE CLASSIFICATION USING SERS AND DEEP LEARNING

MATILDE ALEXANDRA DAMAS LUCAS

Bachelor in Computer Science Engineering

Adviser: Doutor Carlos Augusto Isaac Piló Viegas Damásio
Associate Professor, NOVA University Lisbon

Examination Committee:

Chair: Doutor João Alexandre Carvalho Pinheiro Leite
Full Professor, NOVA University Lisbon

Rapporteur: Doutora Teresa Cristina de Freitas Gonçalves
Associate Professor, Évora University

Adviser: Doutor Carlos Augusto Isaac Piló Viegas Damásio
Associate Professor, NOVA University Lisbon

NOVA University Lisbon
July, 2023

SERS Spectra Classification using Deep Learning

Acknowledgements

Firstly, I would like to thank my adviser Professor Ludwig Krippahl for his guidance and support throughout the development of this thesis. I would also like to thank Professor Carlos Damásio for accepting to take on the role of my thesis adviser. To Professor Ricardo Franco and Maria João Oliveira, thank you for allowing me to be a part in this study and for helping me understand some of the more complex concepts of SERS.

A big thanks to the NOVA School of Science and Technologies, to the Department of Computer Science and its faculty members. The last five years were very challenging but your guidance made this achievement possible.

Last but not least, I would like to express my eternal gratitude to my family and friends. To my parents and my sister, thank you for always believing in me and supporting me throughout this journey. To my friends and colleagues, your help and companionship has been essential these last five years. To my boyfriend, thank you for all the love, encouragement, patience and for keeping me motivated. Thank you all, without you this achievement would not have been possible.

Abstract

Recently, wines became a carefully engineered mixture of components. The particularities of each mixture are what make wines unique and, in order to make this analysis of mixtures and combinations, techniques of thorough identification of components are required. Such techniques, like SERS, produce very detailed information, however, that information can be hard to interpret. Given the large number of possible samples and the endless variations in the wines' compositions, the interpretation becomes more complicated.

The main goal of this study is to find relations within the characteristics and compositions of a variety of Portuguese white wines. We propose the use of Deep Learning for the task of feature extraction and dimensionality reduction of the SERS spectra of the Portuguese white wines, in order to find a representation that is suitable for analysis by simple predictive algorithms. Deep Neural Networks have been proven to be very useful in finding simpler representations of complex data, while improving the relevance of the extracted features and preserving their structure, whilst simultaneously giving an insight on the characteristics the network considered most relevant for that representation. Autoencoders are Deep Neural Networks especially designed for learning these representations from learned patterns in the data, with the sole purpose of making the representations sufficiently informative to be reconstructed back to its original state.

In the presented thesis, we attempted to use traditional Autoencoders (AE) for feature extraction and dimensionality reduction prior to classification and clustering. With the autoencoders, it was not possible to generate representations that showed a clear distinction between wine regions, types or grape variety, but were able to successfully generate representations that demonstrated the distinction between the different wines, meaning that this technique can be useful to provide understanding on the characteristics that distinguish each wine.

Keywords: Deep Learning, Deep Neural Networks, Autoencoder, Dimensionality Reduction, Feature Extraction, Surface Enhanced Raman Spectroscopy (SERS), Silver Nanostars, White Wine Classification ...

Resumo

Recentemente, os vinhos tornaram-se uma mistura de componentes cuidadosamente concebida. As particularidades de cada mistura são o que torna os vinhos únicos e, de modo a que esta análise de misturas e combinações seja possível, é necessária a utilização de ferramentas de identificação rigorosa de componentes. Tais técnicas, como SERS, produzem informação muito detalhada, no entanto, essa informação é muito difícil de interpretar. Dado o grande número de amostras possíveis e as infinitas variações nas composições dos vinhos, a interpretação torna-se ainda mais complicada.

O principal objetivo deste estudo é encontrar relações nas características e composições de uma variedade de vinhos brancos portugueses. Propomos a utilização de Aprendizagem Profunda para extracção de features e redução da dimensionalidade dos espectros SERS de vinhos brancos Portugueses, a fim de encontrar uma representação que seja adequada para análise com algoritmos preditivos simples. Redes Neurais Profundas provaram ser muito úteis na procura de representações mais simples de dados complexos que, ao mesmo tempo, melhoram a relevância das features extraídas e preservam a estrutura dos dados, enquanto permitem também dar uma visão das características que a rede considera mais relevantes para gerar essa representação. Os Autoencoders são Redes Neurais Profundas especialmente concebidas para aprender estas representações a partir de padrões aprendidos dos dados, com o único objetivo de tornar as representações suficientemente informativas para serem reconstruídas de volta ao seu estado original.

Nesta tese utilizamos os Autoencoders (AE) tradicionais para extracção de características e redução da dimensionalidade antes da classificação e clustering. Com os autoencoders não foi possível gerar representações que mostrassem uma clara distinção entre regiões, tipos ou castas de vinho, mas foi possível gerar com sucesso representações que demonstram a distinção entre os diferentes vinhos, o que significa que esta técnica pode ser útil para a compreensão das características que distinguem cada vinho.

Palavras-chave: Aprendizagem Profunda, Redes Neuronais Profundas, Autoencoder, Redução de Dimensionalidade, Extração de Features, Espectroscopia Raman Amplificada por Superfície (SERS), Nanoestrelas de Prata, Classificação de Vinhos Brancos ...

Contents

List of Figures

List of Tables

Glossary

Feature Engineering Analysis, selection and transformation of the variables in raw data, also known as features, prior to their use. Feature engineering allows to select the most relevant features and/or transform existing features in representations more suited for use in all sorts of applications, like Machine Learning and Deep Learning algorithms. 2

Geodesic Distance Geodesic distance is the shortest path in between two points measured along the surface of the plane 20

Gradient Vector of the partial derivatives of the loss function with respect to the parameters being adjusted. It represents the slope of the loss function. In Machine Learning, the gradient can be used as a measure of the changes in the function weights with respect to the changes in the error. 9

Gradient Descent Algorithm used to minimise the loss function. At each iteration, the algorithm calculates the gradient of the loss function of the current iteration, with respect to the current parameters/weights, and calculates the direction and step in which the function weights most be updated in the next iteration in order to minimise the error. It is called gradient descent because the algorithm follows the surface of the loss function and slides down, in the opposite direction of the gradient, until the gradient is close to 0, which indicates that the error is close to its minimum value. 13

Imbalanced Datasets Datasets with over or under represented classes. In these datasets, some classes have a considerably different amount of examples than others 2

Jacobian Jacobian is a matrix of partial derivatives that describes the rate of change of a function with respect to each of its input variables. In the field of Deep Learning, this function can be a deep learning model and the jacobian of that model is a matrix that represents how the output is influenced by each variable in the input (features). 15, 41, 52, 53

Outliers Data examples with abnormal values that deviate from an acceptable range of values for that dataset 2, 7, 59

Preprocessing Treatment of the data prior to its use in the primary processing algorithms 2

wavenumber Number of complete wavelengths per unit distance. The wavenumber is inversely proportional to the wavelength. It can be defined as $\frac{1}{\lambda}$, with λ being the wavelength. 33, 52, 53, 57, 66, 67, 74

Acronyms

AE Autoencoders 10

ANN Artificial Neural Networks 8

BIRCH Balanced Iterative Reducing and Clustering using Hierarchies 7, 22, 23, 39, 63

CAE Convolutional Autoencoder 27, 28

CNN Convolutional Neural Network 29, 30

DBSCAN Density-based Spatial Clustering of Applications with Noise 7, 22, 23, 39, 54, 58, 63, 66, 73

DNN Deep Neural Networks 2, 3, 8, 10, 15, 16, 31, 35, 40, 41, 51, 53, 54, 57, 62, 63, 66, 71, 72

FCN Fully Convolutional Neural Network 29, 30

GMM Gaussian Mixture Model 7, 22, 23, 39, 63

ICA Independent Component Analysis 19, 20

ISOMAP Isometric Mapping 20, 23, 40, 43, 44, 52, 55, 56, 63, 66, 72, 73, 83

KL Divergence Kullback–Leibler Divergence 15, 27

KNN K-Nearest Neighbours 21, 30, 39, 43, 44, 47, 48, 50, 51, 61, 62

LReLU Leaky Rectified Linear Unit 9, 36, 37, 40

MSE Mean Squared Error 14, 38

PCA Principal Component Analysis 2, 6, 18, 19, 20, 23, 24, 27, 29, 39, 40, 43, 44, 49, 50, 52, 55, 56, 63, 66, 72, 73, 83

PCANet Principal Component Analysis Network 29, 30

ReLU Rectified Linear Unit 9, 30

RF Random Forest 21, 39, 43, 44, 47, 48, 50, 51, 61, 62

SAE Stacked Autoencoder 30, 31

SERS Surface-enhanced Raman Spectroscopy 1, 2, 3, 4, 17, 18, 29, 30, 32, 33, 34, 70, 76

SVM Support Vector Machine 30, 31

t-SNE t-Distributed Stochastic Neighbour Embedding 20, 39, 40, 43, 44, 45, 49, 50, 52, 55, 63, 66, 73, 83

VAE Variational Autoencoder 25, 26, 27

1

Introduction

Wine has been part of many cultures for thousands of years. Previously, wine production was a process of trial and error and, maybe, some luck, but recently, winemaking and enology, the science behind the winemaking process, have taken on an unprecedented popularity. The best wines are a carefully engineered mixture of components derived from the different grape strains, barrels, soils, among others. This means that wines are very complex mixtures of multiple components. The particularities in the mixtures are what make wines good and unique, hence the motivation for this area of study.

Nowadays, the advances in technology are allowing us to create very sensitive data gathering tools that are leading to a progressively higher complexity of information. Traditional data processing techniques are not able to keep up with the rising demand, therefore there is a need to study the application of more powerful processing methods when dealing with complex data.

Surface-enhanced Raman Spectroscopy (SERS) is a highly sensitive technique for detecting molecules adsorbed on metal nanostructures. By mixing the substances with colloidal solutions containing specific nanoparticles, it is possible to identify the presence of molecules by the particular spectrum dispersed when these are adsorbed on to the nanoparticles' surfaces and illuminated with laser lines. Due to its highly sensitive nature, SERS is capable of detecting individual molecules, making it the indicated technique for studies where a thorough identification of the components is needed.

However, in SERS spectra of complex mixtures, such as wines, the contributions of many different molecules present in the mixtures overlap, making interpretation of the results very difficult. Besides, the spectra of each sample is converted into a vectorial representation of all the characteristics SERS can collect, resulting in high dimensional vectorial spaces. As a consequence, SERS spectra requires advanced and sophisticated data processing to extract meaningful information from the data.

Recently, the advances in Artificial Intelligence and Machine Learning techniques are making these very complex computations of data processing solvable with Neural Networks. Deep Learning Neural Networks allow to accurately analyse big sets of highly complex data by extracting its most relevant features and identifying relevant relations

and patterns between them. These methods have been shown superior to the traditional machine learning techniques where feature extraction and noise reduction relies on tedious analyses by experienced personnel or application of computationally expensive preprocessing algorithms. Furthermore, some of these algorithms apply dimensionality reductions to the feature space that, in many cases, end up causing loss of important information, leading to potentially tainted results. Therefore, in the case of complex data sets as the ones obtained from SERS spectra, even after the preprocessing of features, machine learning algorithms may still struggle to find meaningful relations in the given features, making these methods lack in performance and accuracy.

Deep Neural Networks (DNN) combine multiple learning layers to overcome the obstacles of traditional techniques. These layers are specially trained to output meaningful representations of the data, reducing the dimensionality without losing important information while finding relations and patterns hardly detected by human analysis. The obtained representations are then used for tasks like classification or clustering. Since the obtained data is overall more informative, the algorithm's prediction performance improves as the results are not nearly as compromised by uninformative features as with machine learning methods.

1.1 Motivation

Despite all the advantages of machine learning in dealing with complex problems, to achieve the full potential of these techniques, data Preprocessing and Feature Engineering steps must be included. Whilst not mandatory, the application of Preprocessing and Feature Engineering can improve the performance of the algorithms by detecting Outliers, by balancing Imbalanced Datasets [21] and performing feature extraction, selection and transformation [19]. However, the misuse of these techniques can distort the data and negatively affect performance.

Deep Learning techniques have been shown useful in solving these types of problems by applying self learned dimensionality reductions to complex datasets without losing important information. This is where traditional Machine Learning methods are flawed, hence needing the additional feature engineering prior to their usage.

In the particular case of SERS spectra of white wines, a study was conducted in the past applying Principal Component Analysis (PCA) to a portion of the Portuguese white wines dataset [2] and it was possible to relate the wines to its regions and types using the simple dimensionality reduction algorithm (study further analysed in Related Work - Applications in spectra analysis). However, the data is very complex and in this thesis the dataset is composed of more samples, therefore we believe that by applying more sophisticated techniques it will be possible to obtain better results and make more detailed analyses.

In short, the motivation for this study is to solve the current problem of the high

dimensionality and complexity of the feature space obtained in SERS spectra by manipulating the data using more powerful techniques.

1.2 Objectives

The goal of this study is to use machine learning techniques for the processing and categorisation of the complex data obtained from the SERS spectra of Portuguese white wines. The main focus is identifying relations between the wines' grape strains, region and other attributes, such as minor components responsible for the unique characteristics of each wine.

With the proposed solution described in 1.2.1, the aim is to apply the combination of the spectroscopic detection technique and the trained model to obtain a fast and efficient tool for accurate fingerprinting and discrimination of Portuguese white wines. Next are some of the aspects this thesis is expected to accomplish.

- Analysis of the distribution of the spectra samples in low dimensional manifolds through clustering algorithms
  - Goal 1: Gain understanding how wines relate and differ from each other from the formed clusters
  - Goal 2: Quantify the similarities and disparities between wines from their distances when projected on to a low dimensional manifold
- Classification of the spectra samples with different labels (ex. region, type of wine and type of grape strains)
  - Goal 1: Analysis of the capability of prediction of different labels given the information in the spectra
  - Goal 2: Gain understanding the influence of the features (areas of the spectra) in the classification

The combined solution is represented in figure 1.1.

1.2.1 Proposed solution

The proposed solution is to build a DNN model trained for the analysis of the SERS spectra data in a universe of related samples, meaning that the aim is to build a model whose performance is optimised for the discrimination of spectra of white wines.

The model will be composed of an unsupervised part, responsible for manifold learning and dimensionality reduction of the data. As the data is very complex in its normal state, the goal is to make a model able to find some structure in the data through a dimensionality reduction of the feature space. In theory, the unsupervised part will force

the model to extract new, more relevant features by finding relations between them and discard uninformative or redundant characteristics collected by the spectroscopy process.

In this thesis, the solution will focus on exploring algorithms to obtain multiple latent representations that emphasise the particularities of each wine. After the unsupervised feature extraction, this simpler representation of wine's SERS spectra, represented by the OUTPUT in figure 1.1, can be used for multiple purposes. Since the goal is to go beyond the relations that define if a wine belongs to a class or another and evaluate capability of relating wines and wine's characteristics to its components through the compressed representations, not only will the application of classification be studied, but also multiple clustering methods will be experimented.

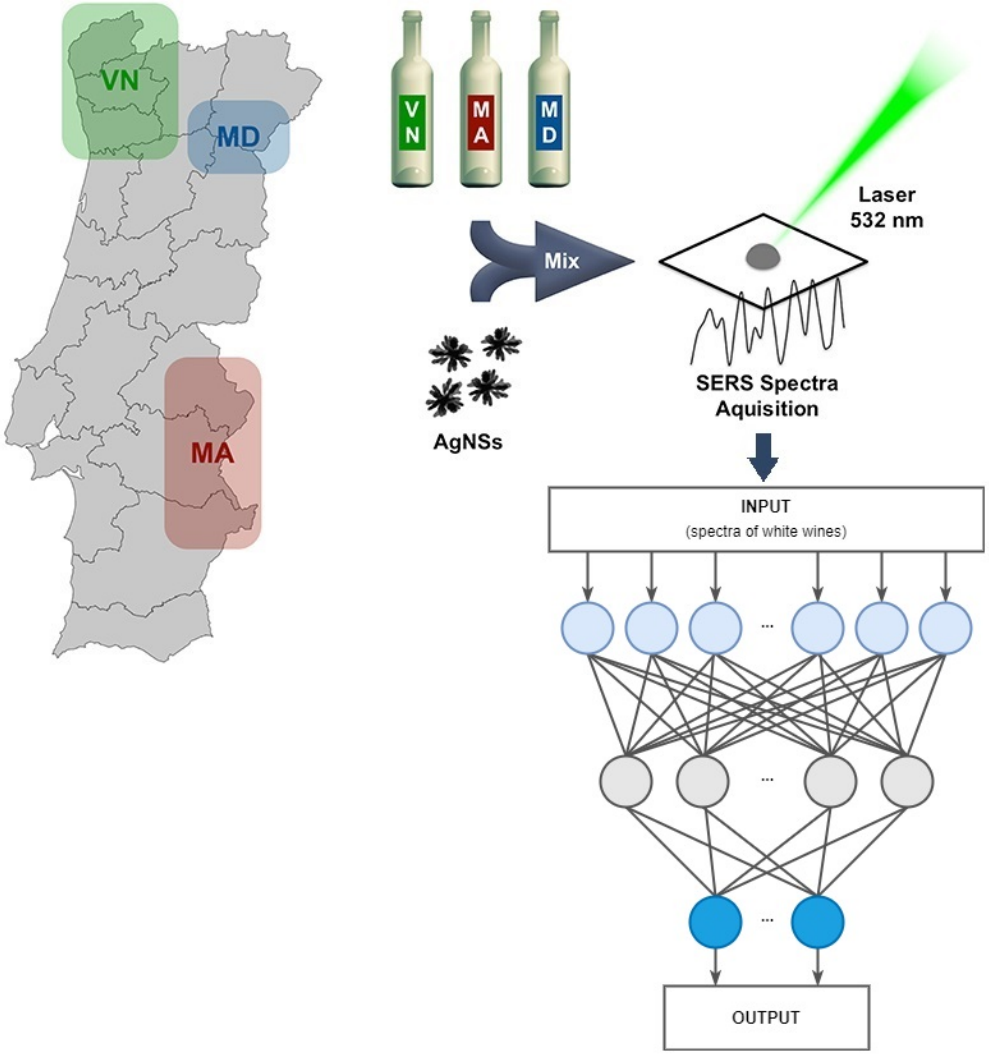

Figure 1.1: Spectra of white wine samples from three different wine regions in Portugal are collected using SERS and are later fed into a deep neural network trained for sample discrimination. Adapted from [2]

2

Concepts

2.1 Machine Learning

Machine Learning is a field of computer science that focuses on developing algorithms capable of learning from given data and performing some task with that acquired knowledge. As the data examples are fed into the algorithm, it adapts and improves its performance by fine tuning its parameters, achieving impressive results that in many cases can not be replicated by human hand.

Essentially, machine learning algorithms get a set of examples as input. Each example is represented by a group of different features which are not all equally significant. Therefore, since not all features take on the same importance, different weights are employed to reflect the significance each feature has in the universe of related examples. Training the algorithm is the process of adjusting and fine tuning the weights with the objective of minimising the difference between the reality and the calculated result.

The learning algorithms can be divided into several categories, some of which are Supervised Learning and Unsupervised Learning, depending on the existence of labelled data or lack thereof.

2.1.1 Supervised Learning

Supervised Learning is the subset of Machine Learning that requires all data to be labelled. This means that, when training, the data used in that process must already be categorised into the classes the algorithm is meant to predict. This is because these methods approach learning by comparing their predictions to the actual labels and adjusting themselves in order to improve prediction capacity and minimise that difference. Supervised techniques are mostly used for classification and regression. In classification, algorithms are trained to predict the class an example belongs to based on their features, this is, classification is used to split data into concrete, predefined values. On the other hand, regression models try predict continuous values, this means that, instead of predicting the quality of a data example, they will predict some quantity.

Multiple algorithms exist for the task of supervised classification, however, not all

algorithms are capable of predicting the exact same thing. There are multiple types of classification problems, the principal being: binary classification, where the examples belong to one of two classes; multi-class classification, where the examples belong to one of many possible classes; and multi-label classification, where examples can belong to more than one class at the same time. This means that classifiers need to be chosen carefully, depending on the problem and expected prediction.

2.1.2 Unsupervised Learning

Unsupervised Learning is the subset of Machine Learning that makes no requirements in terms of labels since it does not use them for training. This process is done by finding patterns in the features. Unsupervised methods are useful for dimensionality reduction and clustering, where the algorithms are able to learn how to obtain low-dimensional representations of the data, which can later be used for visualisation, as input for supervised prediction algorithms, among others. These methods are especially useful when there is little knowledge about common and specific features in the data and their relation in the data examples, as manually selecting relevant features is nearly impossible even for experts. The practice of learning and finding the representations of high dimensional spaces in abstract, low dimensional spaces is called Manifold Learning.

Manifold Learning bases itself on the notion of manifold. Manifolds are a mathematical term for surfaces of n dimensions that form a generalisation of Euclidean spaces. This subset of unsupervised approaches operate on the assumption that high dimensional representations follow a lower dimensional manifold and, consequently, it is possible to obtain a low dimensional representation, which lies in that manifold, without losing much information [23]. Unlike other widely used dimensionality reduction methods like PCA, Manifold Learning dimensionality reduction does not apply linear transformations to the data. Instead, the manifolds have no requirement of dimension or shape [36] giving more freedom in the possible transformations. In short, manifold learning admits non linearity in the data it is trying to reduce and is capable of maintaining its structure without losing information.

Clustering is the task of grouping examples of unlabelled datasets based on some similarity metric. Clustering algorithms are unsupervised approaches, meaning that they blindly form groups of examples with no prior knowledge of the classes. This can be quite beneficial because categories are defined by the algorithm, which can show relations and patterns between data points beyond the classes they are part of. Additionally, these methods can be used for reduction of data to its relevant examples by creating abstract properties from the clusters and using them to replace all data points in the group [22].

The criteria used for forming the clusters is very important in these techniques as it can lead to the same datasets being divided in different ways depending on the chosen

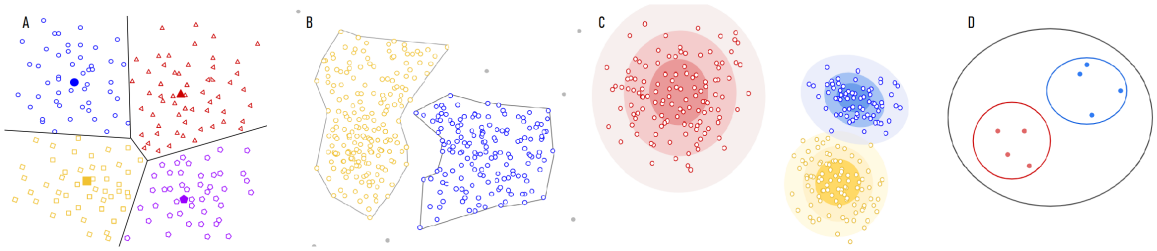

Figure 2.1: Visual representation of types of clustering. A) Centroid-based; B) Density-based; C) Distribution-based; D) Hierarchical. Adapted from [9].

rule. The principal categories are: Centroid-based clustering, Density-based clustering, Distribution-based clustering and Hierarchical clustering.

Centroid-based algorithms (represented in Figure 2.1.A), like K-means, define clusters by the points' proximity to the centre of the clusters, called centroids. This is an iterative method that starts by choosing random points and gather their neighbouring data points to form initial clusters. Afterwards, the centre of the cluster is recalculated and the neighbours are added to the groups. This process is repeated until there is little to no change in the positioning of the centroids in between iterations. In the end we end up with several clusters that can be represented by the centroid, which can be seen as an overview of all the points in that group. These algorithms are quite simple and efficient, however their results can be affected by the presence of Outliers.

On the other hand, Density-based clustering algorithms (represented in Figure 2.1.B), like Density-based Spatial Clustering of Applications with Noise (DBSCAN), group areas very populated with examples into clusters. This leads to clusters that vary in size and shape. In spite of working well with datasets that contain Outliers, it struggles with data of varying densities and high dimensions.

Distribution-based clustering (represented in Figure 2.1.C), like Gaussian Mixture Model (GMM), forms data clusters by assuming they follow a certain distribution and therefore, based on the distance to the centre of the cluster, some points have a higher probability of belonging to the clusters than others. This method, however, does not work well when distributions are unknown.

Finally, Hierarchical clustering (represented in Figure 2.1.D), like Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH), creates trees of clusters where the parent clusters are formed by smaller ones. This means that the number and size of clusters can be defined by choosing to separate child clusters or maintain the higher level representations. In Figure 2.1.D the black cluster is composed of both red an blue clusters, one can choose to represent the data by presenting the black cluster or its child components.

As mentioned, clustering algorithms can be quite useful in finding patterns between examples. In real world problems, data does not only relate in one way, this is, there are more relations between the examples then the ones that define the category the example

is inserted into. Regardless of data being labelled or not, with the right clustering algorithm, it is possible to find other relations and patters that cannot be detected by expert analysis and that group the data. However, real world problem also end up being very complex problems with highly dimensional datasets that clustering algorithms struggle to analyse [45]. This is when dimensionality reduction and manifold learning are of most importance. In order to analyse datasets through clustering techniques, the data has to be compressed into lower dimensional spaces where features are more representative.

2.2 Deep Learning

Deep Learning is a sub-field of Machine Learning that focuses on models capable of learning to find patterns and extract new, more representative features from the data. These models are able to generate new representations of the data by passing the examples through multiple layers of neurons and applying non linear transformations.

Each of these mentioned neurons is a computational representation of human neurons. These nodes are a combination of inputs, weight and bias and produce an output. Individually, machine learning neurons are able to classify linear sets by applying a non linear activation function to a linear combination of inputs [23].

Artificial Neural Networks (ANN) try to mimic the interactions between neurons and the human brain's capability to process information. This is done by having an architecture of connected groups of nodes that work together in layers that, like in human brains, pass on the learned information onto the next set of neurons until it reaches a state where there is sufficient knowledge to make a conclusion or decide the best course of action.

In ANNs, the basic idea is to, for a fixed size input, obtain a fixed size representation as output. Depending on the intended function of the network, this can be a variety of things, for example, in the case of a classification network, the output represents all the possible categorisations (classes) of the examples, or, in the case of an autoencoder, the output is a reconstructed representation of the original input data.

The starting point of using a DNN is to feed the input to the first layer of the network, often referred to as the input layer. Usually, this layer has the same size as the input data feature space, meaning that, for each feature in a data example, there is a corresponding neuron. Once the data goes through this layer, the outputted representation is fed into the following layer as its input. This process is repeated until the data reaches the output layer. Since this is a deep artificial neural network, between input and output layers there are multiple hidden layers, which is where the major non linear transformations of the data occur.

Like in machine learning models, features are not equally important. Since each layer is trying to optimise its own representation based on the inherited transformations from previous layers, each one of them employs its own set of weights. The model's objective is to minimise some loss function between the output and the input, however, given that the

output results of consecutive transformations on the multiple layers, this calculated error needs to be propagated through all layers so that all weights can be adjusted accordingly. This technique is called backpropagation.

The non linearity is inserted in the algorithm by means of the activation functions. Despite non linearity being optional, it if often preferred since it allows the neurons to learn more complex patterns in the data. These are applied in each layer by each neuron and are functions of the weighted sum of the inputs of that neuron. One of the most common used activation functions is the Rectified Linear Unit (ReLU) function (graphical representation and equation are presented in 2.2). Unlike other activation functions, ReLU is not affected by the vanishing Gradients problem, where, during backpropagation, the Gradient values start to tend to 0, forcing the training to become very slow and, eventually, stop.

$$ReLU(x) = \begin{cases} x & ,x \geq 0 \\ 0 & ,x < 0 \end{cases}$$

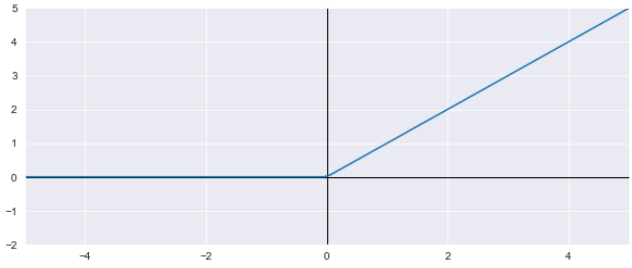

Figure 2.2: Rectified Linear Unit (ReLU) function

Leaky Rectified Linear Unit (LReLU) or Leaky ReLU is an improvement of the ReLU activation where a small slope is applied in the negative side of the function, unlike the constant 0 of standard ReLU. The 0 slope part of the ReLU activation function could cause the Dying ReLU problem where, for all values under 0, the gradient will be 0 and result in the weight of that specific neuron not getting adjusted and essentially, "killing"that neuron. The slope in Leaky ReLU, which is defined before training with a coefficient (represented by the variable m in Figure 2.3) that determines its inclination, prevents that from happening.

$$LReLU(x) = \begin{cases} x & ,x > 0 \\ mx & ,x \leq 0 \end{cases}$$

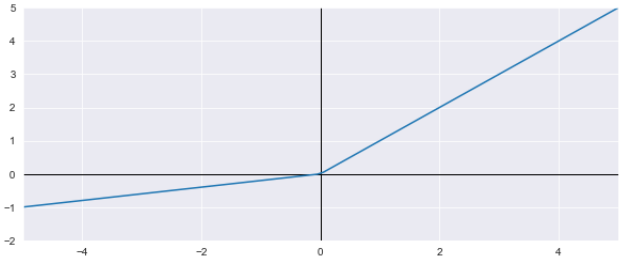

Figure 2.3: Leaky ReLU (LReLU) function. The variable m represents the inclination of the slope

The motivation behind the use of non linearity is, if performed correctly, by the time the data reaches the final layers of the network, the successive application on non linear

transformations will have created a representation in which its classes are linearly separable and are, consequently, more fit for linear classification. The non linearity applied in DNN contrasts with the method of shallow classifiers simply because these transformations are not a function of hyperparameters that are fixed during training, but instead are self learned and adapted by the algorithm during training [23].

Given all of the presented characteristics, it is possible to understand the application of deep neural networks in manifold learning. There are many possible deep learning approaches one can use to obtain the low dimensional manifolds of a high dimensional dataset. One method is through undercomplete autoencoders.

2.2.1 Autoencoders

Autoencoders (AE) [1] are neural networks trained to create an abstract representation of their input in a way that the data remains sufficiently informative so that it can be reconstructed to the original form from that representation. These methods are forms of unsupervised learning, even though the data does not need to be labelled, the models try to minimise the difference between the output and the input, similar to supervised approaches.

These architectures can be divided into an encoder part and a decoder part, as seen in figure 2.4. The encoder's function is to transform the data and obtain, in the middle hidden layer, an abstract representation made of new and hopefully more representative features. On the other hand, the decoder's job is, from that abstract representation of the middle layer, to recreate the original data.

Depending on the size of the hidden layers, autoencoders can be classified as overcomplete or undercomplete. The first is when the hidden layers have more neurons than the input and output layers. This way the features are mapped into high dimensional spaces that, in theory, will expand the data providing more markers for the models to analyse. On the other hand, undercomplete autoencoders have an architecture that employs progressively smaller layers, from input to the middle layer, which force the feature space to be compressed onto a lower dimensional space. This means that the output of the middle layer can be optimised to obtain a low dimensional representation of the data with the most relevant and informative features, making it ideal for manifold learning [23].

2.2.2 Classifiers

Deep Neural Networks can be built and trained to perform classification tasks. As explained, classification is usually a supervised type of learning. In the case of classifier neural networks, the input is the set of features that characterise the dataset and the output is the model's prediction of the class the examples belong to. These predictions are then compared to the real labels/classes of the examples so that the neural network can learn to make more accurate predictions.

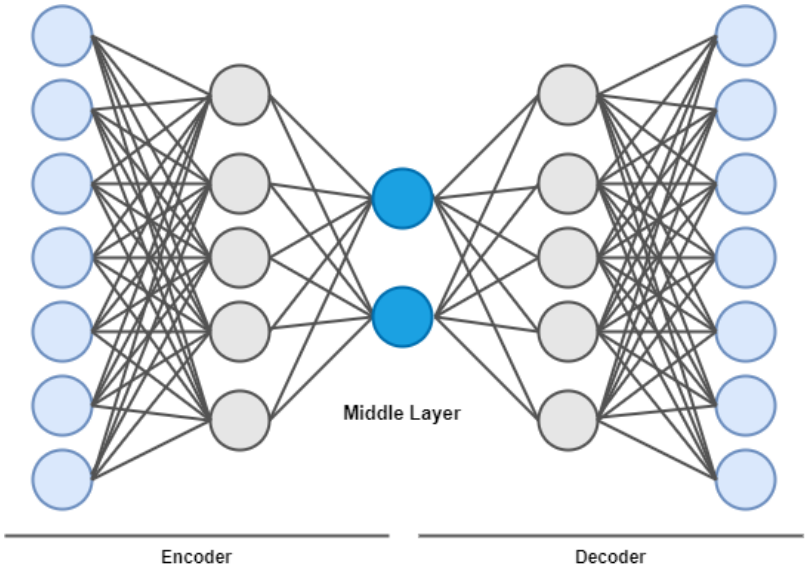

Figure 2.4: Undercomplete Autoencoder. The middle layer works as a bottleneck to force the encoder to find a lower dimensional representation of the input data that is informative enough so that the decoder can reconstruct it.

In order to provide predictions about the datapoints most probable classes, these networks have a particularity in the output layer, which is the activation function and the number of neurons. In the case of binary classification, where the objective is to get the probability of the datapoint belonging to a class or another, most common tactic is to use the Sigmoid function (Figure 2.5) as activation function of an output layer of one neuron, that will output 0 or 1 depending on the predicted class, or two neurons, where each neuron will output the probability of that datapoint belonging to that class. The same logic can be applied to multilabel classification problems, where each datapoint can be attributed one or more labels at the same time. In this case, the Sigmoid activation (Figure 2.5) can be used as activation function of the last layer, that would have as many neurons as the number of labels. This way, each output neuron will represent the probability of the labels being attributed to that datapoint. The combination of the higher probabilities is the classes the neural network predicts that the example belongs to.

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}$$

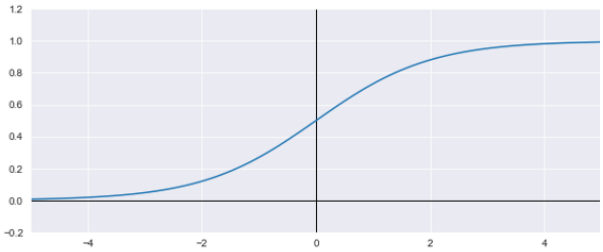

Figure 2.5: Sigmoid function

Lastly, in the case of multiclass classification problems, where there are more than two classes but the datapoints can only belong to one of them at a time, the most common

approach is using the SoftMax fuction as activation function on the last layer, that would have as many neurons as the number of classes. The combination of Softmax function and k neurons (being k the number of classes) results an output that sums up to 1, where the value of each neuron can be interpreted as the probability of the datapoint belonging to the class that neuron represents. This means that the neuron with the highest probability is the most probable class of the datapoint. The SoftMax equation is the following:

$$SoftMax(\vec{x})_i = \frac{e^{x_i}}{\sum_{k=1}^{k} e^{x_k}}$$

2.2.3 Deep Learning Training

As briefly explained throughout this chapter, both machine learning and deep learning algorithms get as input a set of features that characterise a datapoint and output a prediction. This is because these algorithms study the data and its features and learn the characteristics that are significant or not. The networks use different weights to penalise the features to reflect their significance in the predictions they are trying to make. Training is the process of adjusting and fine tuning the weights to perfect the predictions.

Normally, the training process for deep learning networks starts by setting the weights to random values [23]. Then, the datapoints start being fed into the network and the network can start making predictions (the first predictions will be random due to the random weights) that will be compared with the expected results. For a classifying neural network that will be the model's prediction of what class the datapoint belongs compared with the class it actually belongs to, and for an autoencoder it will be a comparison between the reconstructed and original datapoint. This comparison is done using a loss function that calculates the loss, which is a measure of the error between prediction and true value. The way the network learns how to improve itself is by adjusting its weights in a way that minimises the difference between the original data and these predicted results.

In Deep Neural Networks, the error is calculated at output, however, all weights from all neurons in the network must be adjusted in order to improve predictions, this means that this error needs to be propagated backwards through all the hidden layers until the input layer. This is called backpropagation. Because all layers of the network are dependant on the computed representations inherited from their previous layers, this technique of backpropagating the error allows to update the neurons' weights on every layer taking into account that all other neurons are also updated in order to minimise the loss function. This is, since the sequence of actions and transformations is what is responsible for the output, then the whole sequence needs to be adjusted to optimise the network. At some point during training, the network reaches a state where the backpropagated error is generating derivatives that tend to zero and the weight fine-tuning is so minimal that it minimally affects results, so training is just becoming slower while the solution is not really improving. This stage is when training ends.

Nevertheless, this process of calculating loss and readjusting the weights does not usually happen for every datapoint that passes through the network. Instead it is done every x datapoints using a Gradient Descent methodology.

The training can be divided into epochs and batches, where epochs are a passing of all the training data through the network, exactly once, and the batches are the subsets of the training data after which the loss is calculated and the weights are adjusted. The definition of the number of epochs and the batch size can greatly influence the training. Using too little epochs and the network will not be able to learn properly and go into underfitting, but use to many epochs and the model will run the same data so many times that it will overfit to the training data. On the other hand, the batch size determines both the number of times the weights are adjusted and the size of the adjustment step. In order to optimise training, this number must be determined taking into account that bigger batch sizes allow for more parallelization during training, which leads to faster training, however, larger batch sizes can also lead to less randomness in the stochastic gradient descent.

2.2.3.1 Data Preprocessing

Data preprocessing is the treatment of the data prior to its use in the primary processing algorithms. These techniques are commonly employed in Machine Learning and Deep Learning because they can help to improve the performance of algorithms, by making the data more suited to the learning process.

Standardisation and Normalisation are two of the most used techniques to rescale the values of numeric variables.

Standardisation or Z-score normalisation transforms the numerical values so that they have a mean of 0 and a standard deviation of 1. The equation is the following:

$$X = \frac{x - \mu}{\sigma}$$

Where x is the datapoint prior standardisation, μ is the mean of the feature and σ is the standard deviation of the feature. This technique preserves the scale of the features, which can be beneficial for comparison purposes, however it assumes the data follows a Gaussian distribution.

Normalisation is the scaling of numeric values in the dataset to a fixed range which usually is [0, 1]. There are multiple normalisation methods, but one of the more common is the min-max scaling, which rescales the data to values between [0, 1]. The equation is the following:

$$X = \frac{x - min}{max - min}$$

Where x is the datapoint prior normalisation, min is the minimum of the feature and max is the maximum value of the feature.

Other way of normalise the data is by using mean normalisation, where the values are divided by the mean of the feature. With this normalisation, the range in which the data is transformed depends on the range of the original data. The equation is the following:

$$X = \frac{x}{\mu}$$

Where x is the datapoint prior normalisation and μ is the mean of the feature.

Since all values are within the same interval, normalisation can be beneficial for visualisation and for the performance and speed of algorithms as the data is more easily converged. However, forcing the values to be in that range can cause loss of information as the real scales of the features are lost and can no longer be comparable in the same way.

The most common way to perform standardisation and normalisation is by feature, meaning the rescaling is done based of the values of each feature in all datapoints in the dataset, so these transformations are applied column by column. Another way to apply these techniques is to do them row by row, meaning that instead of using the mean and standard deviation of the feature/column for standardisation, the minimal and maximum value of the feature for the min-max normalisation and the mean of the column for the mean normalisation, the values used are of the row. This way helps preserve the relations within the data, whereas doing it by columns can actually distort those relations.

2.2.3.2 Loss Functions

Mean Squared Error (MSE) is one of the simplest and most commonly used loss function in machine learning. As the name indicates, it measures the average of the square difference, or squared error, between the predicted values and the actual values.

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y_i})^2$$

Where N is the number of samples, y is the true value and $\widehat{y}$ is the prediction

Binary Cross-Entropy is another common loss function in machine learning and deep learning, especially in classification problems. Cross-Entropy is the total divergence between probabilistic distributions. As the name indicates, this loss function measures how much 2 probabilistic distributions differ from one another.

$$-\frac{1}{N}\sum_{i=1}^{N}(\widehat{y_i} \times log(y_i) + (1 - y_i) \times log(1 - y_i))$$

Where N is the number of samples, y is the true value and $\widehat{y}$ is the prediction (probability of it belonging to that class)

Categorical Cross-Entropy is derivation of the Binary Cross-Entropy loss function but applies to more than 2 probabilistic distributions, making this loss function suitable for multi-class classification problems.

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \times log(\widehat{y}_{ij})$$

Where N is the number of samples, C is the number of classes, y is the true value and $\widehat{y}$ is the prediction (probability of ti belonging to that class). Note that this loss function is to be used when the labels are one-hot encoded. For integer labels there exists a derivation of this loss function called Sparse Categorical Cross-Entropy, that works in the same way as Categorical Cross-Entropy but with labels not one-hot encoded

Kullback–Leibler Divergence (KL Divergence) [24] is very similar to Binary Cross-Entropy with the difference that it calculates the divergence of one probabilistic distribution in relation to another distribution $D_{KL} = (P||Q)$, rather than the total divergence between two distributions.

$$-\sum_{i=1}^{N} \widehat{y_i} \times \log(\frac{\widehat{y_i}}{y_i})$$

Where N is the number of samples, y is the true value and $\widehat{y}$ is the prediction

2.2.4 Sensitivity Analysis

Sensitivity analysis studies the sensitivity of the outputs of Deep Neural Networks when there are changes in the inputs. This analysis is used to understand which features of the input data have the most influence on the output of the DNN. Having the knowledge of what features have an impact on the output and the relative measure of that impact can be very beneficial for interpretation of results, especially when the meaning of the features is known, and even to choose and improve the models in order to have better results.

There are multiple techniques for sensitive analysis. The ones which have demonstrated to have a better performance are the Partial Derivative algorithm and the Input Perturbation algorithm [5].

The Partial Derivative algorithm consists in calculating the partial derivatives, or gradients, of the outputs of the neural network with respect to the inputs, which mathematically corresponds to calculating the Jacobian of the output with respect to the input features. From these values, it is possible to evaluate, quantitatively, how each variable in the generated output is influenced by each feature in the input, or how sensitive each output feature is to changes in each of the input features. The most important features will be the ones which cause the biggest changes in the output and the least important features will be the one which lead to little or no changes in the output.

The Input Perturbation algorithm consists in repeatedly feeding the same input to DNN model with a small perturbation in one of the features while maintaining the value of the remaining features and registering the output. The idea is that, by repeating the process but applying the same perturbation for all features, one at a time, and comparing the changes in the output, it is possible to get an estimation on the influence of each feature on the output, which is the sensitivity of the output to each input feature.

2.2.5 Transfer Learning

Transfer Learning in Deep Learning is the combination of a previously trained deep neural network to another deep neural network. The first pre-trained network is thoroughly trained to perform transformations on a specific universe of data in order to learn the features and perform feature extraction. These networks are then combined with other network that will reuse or adapt these transformations to their advantage, as the pre-trained model will feed their output to this following network. Although the first network is not usually trained to perform the same task as the final model, nor is it trained using the exact same dataset, as long as it is used in a similar context as the training, the transformations of the pre-trained model can reduce significantly the needed training. This technique is especially useful in cases where the dataset is small and it does not have enough examples for a proper training of a deep neural network.

2.3 Surface-enhanced Raman Spectroscopy (SERS)

Raman Spectroscopy is a technique used to provide structural information about chemical components. In order to understand how Raman Spectroscopy works, we must remember that everything is composed of atoms, most of them forming molecules by establishing chemical bonds with other atoms. Molecular Vibration is a phenomenon described by vibration of the atoms along those chemical bonds. When a molecule absorbs energy, the molecular vibration is excited. This means that the chemical bonds in the molecule vibrate with a higher frequency. Molecules thus transition to a higher vibrational energy state, when excited, relaxing back to the original lower vibrational energy state, releasing energy. Raman takes advantage of molecular vibration for the identification of components. This technique uses intense light to interact with molecules and increase their vibrational energy. When light is pointed at a molecule, the molecules energy state increases and the molecules transition to another vibrational state. Instantaneous relaxation to the fundamental state with light scattering, results in two types of scattered radiation: Rayleigh or elastic scattering, at the same wavelength of the impinging laser, and Raman or inelastic scattering, at different wavelengths relative to the impinging laser. These wavelength variations between incident and scattered light are unique to each component, and this is what allows Raman to identify molecules. The problem with Raman Spectroscopy is that the scattered light results in a small portion of inelastic scattering,

or Raman scattering, and mostly (by six orders of magnitude) elastic or Rayleigh scattering. This extremely weak intensity of Raman signals leads to low sensitivity and poor component discrimination of Raman Spectroscopy.

Surface-enhanced Raman Spectroscopy (SERS) is an improvement of the Raman spectroscopy technique. Unlike Raman, SERS is selective to molecules adsorbed to metal nanostructures. When some molecular solution is mixed with metal nanoparticles with rough surfaces, the different compounds start being adsorbed onto those surfaces. When intense light (laser) is pointed at them, it produces a stronger scattering effect of different wavelengths, called SERS, as it can be seen in the spectra in 2.6. In the graph there is a noticeably bigger variation in the intensity of the scattering effect using SERS versus when using Raman. This allows SERS to have molecular fingerprint specificity and high sensitivity. Additionally, under carefully controlled conditions, with high laser stability and proper calibration, this technique is also capable of quantification, as the intensity of the obtained spectrum is directly proportional to concentration of the component [43].

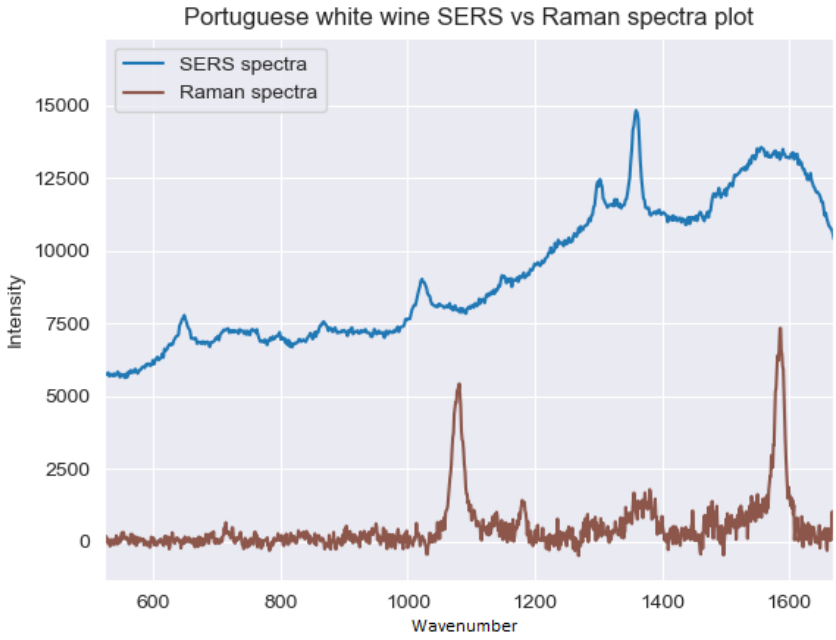

Figure 2.6: Comparison of spectra of Portuguese white wines using Raman method and SERS.

3

State of the Art

3.1 Machine Learning

Machine Learning is a continuously growing field. The ability to take any kind of data and learn to predict outcomes made machine learning algorithms very powerful tools that have been shown to be beneficial in everyday problems such as image and speech recognition, information retrieval and sentiment analysis [6], and complex problems such as detection of cancer [27], forecasting stock markets [38] and spectroscopy [2, 30].

In the following sub chapters, some studies will be presented, studies that resorted to traditional machine learning techniques to solve the dimensionality issues in the data, a common problem when dealing with Surface-enhanced Raman Spectroscopy (SERS) spectra, and some supervised and unsupervised learning algorithms that represent the more common approaches used for classification and clustering.

3.1.1 Dimensionality Reduction with Machine Learning

In real world problems, we are more then often confronted with information which is represented by many characteristics. The treatment of the data must be specialised, depending on the intended purpose of the datasets and the problem being studied. For instance, when dealing with a visualisation problem, the goal is to project the data into a space human brains can comprehend, this is normally 2 or 3 dimensions. With classification problems, there isn't a defined number of dimensions where we aim to project the data on to, instead the goal is to transform the data into a feature space where data examples from each class can be clearly separated [14]. Regardless of what one intends to accomplish with the data, it is necessary that, in the compressed representations, the data structure remains true to its origin.

Principal Component Analysis (PCA) [33] is an statistical method that transforms data into fewer dimensions while maintaining patterns in the features. The reduction is done by linearly projecting the data on to lower dimensions, the Principal Components, and tries to achieve the best low dimensional representation of the features while

preserving the variance of the original datapoints. The principal components (PC) are each representative of a portion of the total variance of the data and are then ordered by their associated variance, meaning that the most significant representation, which is responsible for most of the variance of the dataset, is in PC-1, the second most significant representation is in PC-2 and so on. This means that the last PCs represent little of the data's variance and have, therefore, less influence in the low-dimensional data representation.

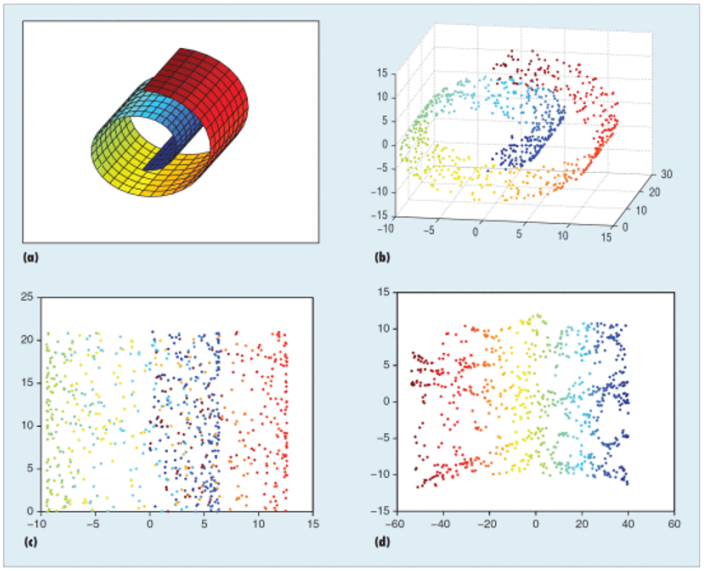

Figure 3.1: Unfolding 800 3D Swiss-roll data points to a 2D space with PCA and Isomap algorithms. (a) Swiss roll, (b) Swiss roll data, (c) PCA and (d) Isomap [47]

In spite of its popularity, PCA is not capable of accurately preserving the data structure when it lies on non linear manifolds [29]. In figure 3.1 is represented a 3-dimensional plot of a Swiss-roll object. When observing the 3D representation, it is possible to infer that its 2D representation must, at least, respect the sequence of colours. However, when PCA is applied to that set of data points, that is not the obtained result. This technique misidentifies the 2-dimensional positioning of points that are in close proximity of each other in the 3D representation but do not have any close relation, such as red, blue and orange data points that are closely projected in 3.1(c).

The struggle to capture the intrinsic structure of a dataset is not a problem limited only to PCA. Other linear methods such as Independent Component Analysis (ICA) also struggle from this.

Independent Component Analysis (ICA) [10] is a generalisation of PCA that aims to separate multivariate data into independent Principal Components. It works by assuming that each sample is composed of statistically independent components, meaning that by changing a value other values will not suffer from that change, and it aims to find them

and separate them into Principal Components. This method can be used for dimensionality reduction in a similar way to PCA, by selecting certain PCs to represent the data. The problem with ICA, other that it not being suitable for transformations of non-linearly separable datasets, is that it does not differentiate between important or unimportant PCs nor does it reveal how many of them are relevant to the representation. This means that it requires extra analysis power and is more prone to error by misusing the PCs.

These methods, compared to nonlinear alternatives, lack the necessary capacity in dealing with real world problems that frequently consist of highly dimensional, non-linear data [4].

By analysing the Figure 3.1, it is possible to see a clear difference when using linear and non-linear transformations. By using non-linear manifold learning techniques, such as Isomap, the dimensionality reduction was able to accurately maintain the the structure and correctly represent the close relations between the data points, as seen in 3.1(d).

Isometric Mapping (ISOMAP) [37] is a distance preserving algorithm that, when mapping data into lower dimensions, seeks to preserve the shortest path between all points along the surface of the manifold. This method builds a neighbourhood network of the shortest paths between pairs of points and computes an eigenvalue decomposition in which the first K eigenvectors with K highest eigenvalues represent the best representation of the original, high-dimensional data. Because the calculations are based on an approximation of the Geodesic Distance of the data, the structure is more easily preserved and the data is not distorted as with linear algorithms. However, ISOMAP also has its weaknesses, for instance, its sensitivity to complex and noisy datasets. In the first step of the algorithm, when the local neighbouring network is being mapped, ISOMAP cannot distinguish between outliers and other noise from the rest of the samples in the collection and this leads to the distortion of the neighbouring graph that is the base of the entire algorithm.

t-Distributed Stochastic Neighbour Embedding (t-SNE) [39] is a statistical algorithm for non-linear dimensionality reduction commonly used for visualisation on high dimensional data. It is based on Stochastic Neighbour Embedding and it calculates the similarity between pairs of points in the high dimensional embedding and, using a cost function, attempts to optimise these similarities in the low dimensional embedding. Since t-SNE tries to optimise the similarities, this results in representations where pairwise similarities in the high dimensional representations are translated into small distances between similar points on the low dimensional representation. However, when reducing the dimensionality, global distances are lost, making these representations not ideal for distance based clustering algorithms.

3.1.2 Classification Techniques

Given the objective of this thesis, a supervised learning classification algorithm is needed to evaluate if the data is sufficiently representative in order to discriminate samples from different classes. Since the dataset can be separated in multiple classes and, in some cases, examples can belong to multiple classes at the same time, we will analyse multi-class and multi-label classifiers. K-Nearest Neighbours and Random Forest are widely used classification algorithms that can be adapted for these types of classification.

K-Nearest Neighbours (KNN) [11] is a classification and clustering algorithm that classifies datapoints based on their proximity. During training, the algorithm learns, iteratively, which points from the training set belong to each class. As the learning progresses, the algorithm bases classification on the assumption that the K nearest points belong to same class of the point used as reference. This type of algorithm performs well given any structure of the data, however it has the downside of having to define the K a priori. KNN is a type of multi-class classifier, however it can be adapted to perform multi-label classification.

Random Forest (RF) [16] is a supervised algorithm used for classification and regression. The RF algorithm employs multiple decision trees, hence it being called forest, and combines their results to make its own single prediction. Decision trees are also supervised algorithms that create feature-based rules to support decisions that lead up to a final prediction. The creation of these Decision Trees, in this algorithm, is based on subsets of the data and features, so that in the end, the final decisions of all trees serve as result of the algorithm. When used in classification, the result is the decision made by the majority of the trees, while for regression it is the average. This algorithm can be used for either multi-class and multi-label classification.

3.1.3 Clustering Techniques

K-means [31] is an unsupervised centroid-based clustering algorithm. With this technique, the clusters are formed by grouping every datapoint near a centroid, reducing the in-cluster sum of squares. The basic implementation of the K-means algorithm requires that the number of clusters, the value k, be provided a priori to the clustering process. This is because the algorithm bases its clusters on initial random centroids that are iteratively updated based on the datapoints that join the cluster in that iteration. This process is repeated until the clusters are fixed. This means that the centroids are a mean of the data from its clusters and are good representations of the points in that group. In spite of being a simple algorithm with good performance, it has its drawbacks, for instance, variations in data can lead to increased variance. Additionally, clusters are assumed to have a spherical shape which can jeopardise clustering accuracy. The fact that the number of clusters has to be defined in advanced can be disadvantageous if one is

using K-means to analyse relations and patterns in data because the algorithm is forced to divide the examples into that number of groups even if the data does not express that sort of division.

Density-based Spatial Clustering of Applications with Noise (DBSCAN) [13] is a density-based clustering algorithm that uses the distance between nearest points to form the clusters. The basic idea is, for each point in a cluster, there must be a neighbouring area with defined radius that contains a minimum number of points. Both neighbourhood radius and minimum number of points must be defined a priori. The algorithm starts of by randomly visiting all datapoints, identifying the ones that comply with the constraints and group them in the same cluster. The clusters are then joined by recursively repeating the neighbourhood calculation for all neighbouring points. A consequence of this process is that the points that are in low-density regions are not included in the formed clusters, however, if points are alone with little neighbourhood then it is probable that they are outliers. This makes DBSCAN effective in the detection of outliers, unlike centroid-based clustering algorithm that are affected by them. Additionally, this algorithm also allows arbitrary shaped clusters and makes no assumptions as to the shape of the different clusters.

Gaussian Mixture Model (GMM) is as distribution-based clustering algorithm that clusters data points that belong to a single distribution of unknown parameters. This method assumes that all the data points are generated from a mixture of a finite number of independent Gaussian distributions, which it aims to identify in order to make the clusters. To this end, expectation-maximization (EM) algorithm employed for finding the parameters of the Gaussian mixture models that best represent the latent space the sets of points are inserted into, this is, the mean, covariance and density of the distributions. After getting these values, the probabilities of the datapoints belonging to a cluster are calculated. This process is repeated until the likelihood loss function is minimised. Because this algorithm minimises the likelihood, it does not force clusters to have certain sizes or structures, unlike other clustering algorithms. However, when a mixture does not have a sufficient number of points, the algorithm struggles to find the covariance values which can impact results.

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [48] is a Hierarchical clustering algorithm that firstly summarises the data into compact, smaller sets of data that are later used for clustering. The algorithm can be divided into two separate phases. The first focuses on building the summaries of the data and map them into a tree of triples, called a Clustering Feature tree. In each of the leafs in the tree lies the entry with the triplets composed of *(number of points in the cluster, linear sum of the points in the cluster, squared sum of points in the cluster)*, and the pointer to the child nodes, if they exist. This means that each leaf has a sub cluster that can be decomposed in its

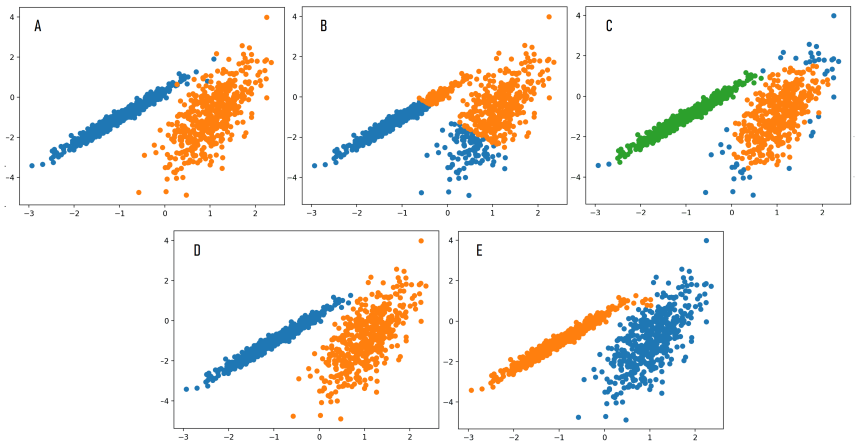

Figure 3.2: Demonstration of different results obtained by different algorithms. A) Real plot of data; B) K-means; C) DBSCAN; D) GMM; E) BIRCH. Adapted from [3].

child leafs. The second phase of the algorithm consists of applying an existing clustering algorithm to the Clustering Feature tree leaves.

These clustering methods are common and widely used, however, K-means, DBSCAN and GMM suffer from the curse of dimensionality [18, 44, 7] which means that performance drops as data dimensionality increases. Even though BIRCH scales well with dimensionality due to the initial compression of the data, is can also be beneficial to apply some dimensionality reduction prior to that analysis.

J. Shi and Z. Luo proved this by comparing the performance of K-means and hierarchical clustering of five datasets of cancer tissue samples with and without prior dimensionality reduction [35]. By analysing the resulting rand index of the clusters obtained when using the raw data and after dimensionality reduction it is visible that, for the majority of the datasets, both clustering algorithms perform better when the dimensionality has been reduced. However, these results depend on the algorithm used to perform the reduction. In addition to evaluating the influence of the dimensionality in the clustering process, this study also evaluated the influence of the dimensionality reduction algorithm. Two method were studied, linear and nonlinear, using PCA and ISOMAP, respectively. In all datasets, the combination ISOMAP + clustering was superior. In some cases, the combination PCA + clustering did not show any benefits since the clustering of the raw data achieved very similar results. This may be due to PCA's difficulty in preserving the data structure through its linear transformations. This can alter the representation of datapoints, therefore damaging the clustering capacity. On the contrary, ISOMAP is capable of reducing dimensionality while maintaining the data structure. The clustering after applying this algorithm can get better results because the features are compressed in a way that the most relevant characteristics remain while irrelevant features that might confuse the clustering process are ignored, making clustering more straightforward. It is important to mention that the hierarchical algorithm was also superior to the k-means.

3.2 Deep Learning

3.2.1 Dimensionality Reduction with Deep Learning

The goal of this study is to find new structures based on the information learned with self-learning models and, as it was previously established, this can be accomplished with algorithms that perform dimensionality reductions, feature extraction and feature compression. In section 3.1.1 we discussed some of these more common and popular algorithms using traditional machine learning techniques. In this section we will discuss the application of deep learning for this task.

As mentioned earlier, deep manifold learning uses autoencoders to find the low dimensional manifolds where high dimensional data lies. These manifolds are able to represent the initial, complex data in less dimensions without loss of information. The more straightforward technique is to use a simple undercomplete autoencoder, as shown in Figure 2.4, in which the data travels through layers that are progressively smaller in terms of dimensionality. This forces the data to be compressed. The way autoencoders ensure that the representations remain informative enough is by trying to reconstruct the original input from those representations. For it to be a learning process, the reconstructed data is then compared to the original data and the discrepancies between representations are evaluated through a loss function that computes a measure of the error. The optimal solution would be to have zero error, meaning that the encoder is able to compute a low dimension representation that contains all the information necessary for the decoder to accurately recuperate it to its original state, leading to no difference between original and reconstructed data. However, that pretty much does not happen for the majority of the cases, especially when the model is not familiar with the data. Before training, the layers are initialised with random weights, this means that, initially, the generated representations are obtained by randomly favouring some features and penalising others. This is the normal behaviour when training Deep Neural Networks. What really differs the training of Autoencoders from the training of Deep Neural Networks explained in Section 2.2.3 is that it compares its output to the input because the goal is for the output to be as similar to the input as possible. After training, the representation obtained from the middle layer of the autoencoder, this is, the last layer of the encoder part, is a new way to represent the data that has less dimensions and nearly the same information [23] (See figure 2.4).

Y. Wang, H. Yao, and S. Zha studied the autoencoder's dimensionality reduction performance and the influence of the dimensions the latent space is forced to take [41]. This study demonstrated that, for the purpose of visualisation, which requires a reduction to 2-dimensional and 3-dimensional spaces, the autoencoders performance was superior compared to a PCA reduction. When applied to the MNIST dataset, the autoencoder projected the examples of each of the 10 classes mainly to different corners and edges, while PCA projected them around the same area. This demonstrates that PCA was not

capable of clearly separating examples representing different numbers. Additionally, this study evaluated the performance of a SoftMax classifier after different dimensionality reductions. In the case of the MNIST dataset, all reductions $\mathbb{R}^{784} \rightarrow \mathrm{R}^{S}$, where 784 is the dimensionality of the samples and s varies between 784 and 1, were computed and fed to a classifier. This evaluation found that there is a relation between the intrinsic dimensionality of the datasets, the number of variables needed for a minimal but informative representation of the data, and the classifier's ability to predict the classes. This demonstrates how misleading high dimensional feature spaces can be. The existence of redundant and uninformative features can affect the performance of classifiers whilst a lower dimensional representation where relevant features are compressed and uninformative ones are ignored can help the algorithms to learn the patterns to discriminate examples.

Even though traditional autoencoders are a great option for finding the low-dimensional representations of data, they are solely trained to do their job, that is encoding and decoding data while minimising the loss of information. Without some control, the networks will perfect their performance by all means and that can cause overfitting of the model. Despite the benefits of this adaptability in handling complex data, when dealing with small datasets, reaching a state of model overfitting is very easy.

Overfitting happens when the network over adjusts to the set of data being used in training. This results in the model learning aspects of the training set that do not generalise to data outside this set [23], meaning that the model is optimised for that specific set of examples instead of learning the general aspects and patters of the universe the data is part of.

In order to counteract this problem, regularisation can be inserted into the model [46]. Regularisation is a method that constrains the training of the model in order to decrease variance, a side effect of overfitted models. In deep neural networks, this can be accomplished by adding a regularisation parameter to the loss function, by using L1 or L2 regression to add penalties the loss function, weights or bias of the neurons, by using dropout layers or by adding noise to the dataset. Additionally, when the dataset is small, one can try to enlarge the dataset by generating new examples. In these cases, a subset of autoencoders known as generative come in handy as they are able to generate artificial examples from the learned information while respecting given constraints.

This types of autoencoders have been shown quite useful in problems were data is highly dimensional. M. S. Mahmud and X. Fu compared the performance of a Variational Autoencoder and a standard Autoencoder when predicting classes of a small set of high-dimensional data [32].

Variational Autoencoder (VAE) [20] are generative autoencoders that learn to encode the inputs as probabilistic distributions in regularised latent spaces. This means that they can learn latent space representations of the input data based on the input's probability distribution. Unlike more traditional autoencoders that consider deterministic encoders

and decoders that encode and decode inputs as single points, VAEs consider probabilistic versions where the encoder computes parameters for a probabilistic distribution of the encoded example given its original version, $p(z|x)$, and the decoder tries to approximate in the other way around, $p(x|z)$, where x represents the data and z the encoded representation. The autoencoders get their name from the technique used to infer the distribution of $p(z|x)$, called Variational Inference (VI).

Assuming that $p(z)$ is of a given distribution and that $p(x|z)$ is approximated from the latent representation obtained in the encoder, we can use the decoder to sample new examples from the joint probabilistic distribution of the data and the encoded representations because $p(x,z) = p(x|z).p(z)$ [23]. Seeing that the encoder learns the distribution's parameters given each input, by sampling representations from that calculated distribution it should obtain similar examples in the output since the decoder is trained to generate data examples randomly removed from that same distribution (See figure 3.3).

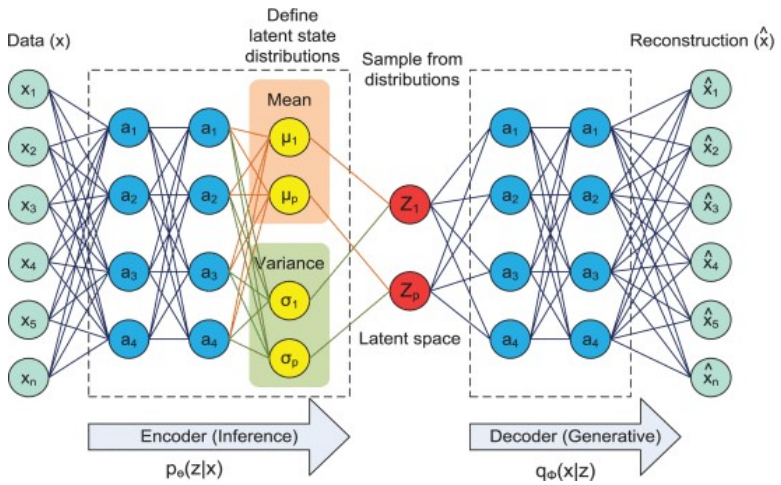

Figure 3.3: Variational Autoencoder. Encoder part learns parameters of the latent state distribution and decoder reconstructs low-dimensional data examples randomly sampled from the latent state distribution [32].

One problematic that arises from this architecture is that training is not as straightforward as other autoencoders since the decoder´s input is a vector randomly sampled from the probability distribution given by the output of the encoder. This means that backpropagation cannot be applied. A workaround this issue is to use a reparametrization trick, which consists of adding the mean to the standard deviation, multiplied by a random number drawn from a Gaussian distribution of mean 0 and variance of 1.

In order to prevent punctual distributions caused by a parameters taking on values of a wide range, a regularisation term can be added. To sum up, the loss function will include a reconstruction error and a regularisation term that penalises the model if the encoder's outputs deviate from the standard normal distribution. This is measured by the Kullback-Leibler divergence between the approximation and the target, $p(z|x)$.

In this mentioned study, the performance of K-means clustering was evaluated when using all the original features and with dimensionality reduction prior to the clustering.

Additionally, the performance was also compared with PCA dimensionality reductions and VAE dimensionality reductions. The results showed that the dimensionality reduction of an average of 10 dimensions improves the the obtained clusters. This fact goes according to theory and other presented studies. In the classification with latent representations, the results show the Variational Autoencoder was the best algorithm overall.

It is true that VAEs are good at extracting features from highly dimensional, complex data. One of the reasons that explained this is that these networks usually learn using on the Kullback-Leibler divergence as a regularisation term. This helps reducing overfitting and helps the representations be evenly distributed by making the learned latent distributions follow a standard normal distribution by penalising the deviations. However, achieving an appropriate balance between the Kullback–Leibler Divergence (KL Divergence) term and the reconstruction error can be tricky. If the reconstruction error term is overly valued in relation to the KL Divergence term, the VAE will focus to much on reconstructing the data and not learning the latent space distribution, whereas if the KL Divergence term is overly valued in relation to the reconstruction error term, the VAE will learn good latent space distribution but the reconstruction of the input from that distribution will not be as accurate as it should.

One other type of networks in Deep Learning are Convolutional Networks that, due to their implementation of sliding kernels, have way less parameters to train.

X. Guo et al. proposed a dimensionality reduction algorithm optimised for clustering. This solution consists of a Convolutional Autoencoder that aims to minimise the reconstruction loss between original and decoded data and the clustering loss [15].

Convolutional Autoencoder (CAE) [25] is the designation for autoencoders that take advantage of the convolution operation to learn locally connected features, through multiple convolutional layers or by a combination of convolutional layers and dense layers. Convolution is a linear operation that, in combination with smaller sized kernel filters, is applied to the data, mapping the result into an output representation, the feature maps, that are then taken through an activation function. The kernel filters are a set of weights that is used across all neurons of the layer it is being applied on. This means that the weights are shared among the neurons of the layers, leading to less weights (parameters) to be adjusted in training, compared to other types of deep neural networks.

By stacking multiple convolutional layers after the input, each of them smaller than the previous, the model can learn hierarchical features and map them into a lower dimensional representation. Additionally, pooling layers can be used after the convolutional layers to reduce the size of the feature maps. This reduces the dimensionality of the representation and, consequently, reduces the number of parameters used to represent the input. In the middle layer, the data is flattened in order to obtain a vectorial representation. As we know, this is the latent representation of the input.

The study by X. Guo et al. [15] proposes a model composed of stacked convolutional layers, followed by a flatten layers that feeds into a fully connected layer with 10 neurons,

which is the embedded layer. The authors theorise that by forcing the autoencoder to learn such under complete representations that it would be able to capture the most relevant features, thus forcing the embedded shape's dimensionality to equal the number of clusters.

The results were evaluated by applying this solution to the MNIST dataset and compare the performance with other techniques. In combination with the K-means clustering algorithm, the convolutional autoencoder optimised for clustering outperformed techniques like dimensionality reduction by a standard Stacked Autoencoder followed by K-means and dimensionality reduction by a standard Convolutional-Autoencoder also followed by K-means.

This method has proven to be efficient for the task it was designed for, however, it also forces the model to find a certain number of groups in the data instead of giving it the freedom to group the data as it finds more correct given the learned features. To this end, a standard CAE followed by a clustering algorithm would be more appropriate and as we can see by the results from this study, this option also gets great results.

Nonetheless, these types of architectures are not ideal to deal with inputs in which the positioning of the patters matters for what they represent. These networks perform feature extraction by means of the mentioned kernel filters, that slide through the data matrix to find patters and structure based on position. This is part of the reason why CNNs are so popular with feature extraction of data like images or sound. Because the patterns can occur in any position and still have the same meaning, whilst in data like spectra, the positioning defines the component. This means that the filter sliding through the data does not learn such robust representations of these inputs.

There are several ways one can make these types of networks more robust. One way is to deliberately feed the network with noisy data. This types of autoencoders are referred as Denoising Autoencoders.

Denoising Autoencoders are networks trained to reconstruct the data from a somewhat corrupted version of the input. This technique, often referred to as noise injection, forces the network to become more robust by relying less on the input and more on regular patterns in the data.

As mentioned, these techniques are often combined. For example, B. Du et al proposed a Stacked Convolutional Denoising Autoencoder for feature extraction prior to classification of images [12]. This model stacks convolutional and denoising layers that learn to ignore noise and more easily generalise the feature space without being influenced by those irrelevant characteristics. This way, the obtained representations are reduced and more robust and were proved better for classification compared to representations obtained with a normal convolutional autoencoder or a denoising autoencoder.

3.3 Related Work - Applications in spectra analysis

Machine learning methods have been studied in the context of SERS spectra analysis.

M. P. de Almeida et al. successfully used PCA to differentiate between *Verde* and *Maduro* wines, as well as the different regions of production of *Maduro* wines [2]. In this study, 10 spectra samples were collected from 3 different wines, one *Verde* and the two *Maduro*. The PCA was applied and the results were plotted after the data was preprocessed using Baseline Linear Correction.

In the PC1 vs PC2 score plot, the distinct clusters representing the different wines were clearly visible. Furthermore, by analysing the negative and positive sections, there could be identified the distinction between the different regions of *Maduro* wine. Moreover, PCA attributed a 91% weight to the PC-1 representation, however the score points were too dispersed over the correspondent axis to be able to give any useful information on its own. It is believed that the abundant components, common in all wines, biased the representation obtained in PC1 by being its biggest contributors. This is supported by the defined clusters obtained when analysing both PC1 and PC2 simultaneously. PC2 is responsible for only 2% of the total variance, meaning that it probably represents some of the minor components present in wines, and these minor components are what differentiate the wines, as expected. This is one known weakness of PCA. When a subset of the features consistently takes on higher values relative to other features, PCA tends to heavily weigh those features while others are undervalued [26]. Taking this into account, even though the combination of PC1 and PC2, in this study, was sufficient to obtain decent results, in the scope of the analysis of minor components, this method is not adequate since it lacks the necessary rigour in maintaining patterns for both abundant and rare components. Although this study is not directly comparable with the one being presented, given the big size differences in the datasets and the different algorithms used for the preprocessing of the spectra, this is a good indication of what is possible to accomplish with these types of analysis.

Deep Learning techniques have been applied to deal with the very complex data collected using Raman and SERS techniques. These algorithms have been shown useful in cancer detection [28], drug identification [42], drug resistant bacteria classification [8, 40] and even SARS-CoV-2 antigen detection [17].

S. Weng et al. studied the use of neural networks to classify the high dimensional datasets of SERS of drugs [42]. More specifically, the classification was made using a Convolutional Neural Network (CNN), a Fully Convolutional Neural Network (FCN) and a Principal Component Analysis Network (PCANet). The difference between these networks and autoencoders is that autoencoders try to minimise the reconstruction loss, meaning that they are optimised to learn other representations of the data that generalise the examples by learning relevant patters that are true to what the examples represent overall. On the other hand, these classifying networks are connected to a classifier and are consequently trained to minimise the classification error. This means that the networks'

hidden layers are optimised to learn representations of the data that reward discriminate features in order to differentiate examples from different classes. Nonetheless, both techniques are able to extract features and obtain latent space representations that are more representative.

As for the CNN, a combination of convolutional and pooling layers was used for the feature extraction and dimensionality reduction. Afterwards, a flatten layer is added to convert the representation into one dimension. This is done for the transition to the following fully connected layers that serve as a way to relate the abstract features to their meaning, in this case is the types of substances. This layer's output is then fed into the classification layer that uses a SoftMax activation function. The convolutional, pooling and fully connected layers all used Rectified Linear Unit (ReLU) activation for the nonlinearity. The model used root mean square prop as an optimiser and the chosen loss function was the cross entropy loss.

The FCN differs from the CNN only in the pooling and fully connected layers that were in this case replaced by more convolutional layers with larger step size. The motivation behind this is the fact that pooling layers cause loss of information by reducing the size of the feature maps calculated by the convolutional layers. This change increases the number of parameters that travel through the network, therefore the fully connected layer is substituted by a 1x1 convolution to decrease the parameters and improve performance. Since this model consist only of learning layers, by the end of the network, the learned features are very complex and more abstract than in CNNs.

PCANet, unlike the other proposed non linear networks, is a cascaded linear network consisting of an input layer, two convolutional layers and an output layer.

The results of the classification when the dataset was used in vectorial shape were compared with classification performed by traditional machine learning algorithms. In this case, classification by both Support Vector Machine (SVM) and KNN outperformed all other techniques, including the neural networks presented, achieving 97.76% and 95.75% accuracy versus the 94.58% obtained with the CNN. This may be because convolution-based networks perform better with data with positional relations with each other, as found in images. When the position itself has meaning, the network can struggle to find relations. For the case of the of classification the dataset in the form of a matrix, the CNN was superior to the other proposed solutions, even better then the machine learning algorithms when fed 1D data. This improvement cam be because, has mentioned, CNNs have less parameters to train and that can compensate the difficulty in finding the relations in the features, compared to the other tested networks.

F. U. Ciloglu et al. proposed the use of a Stacked Autoencoder (SAE)-based Deep Neural Network for the classification of SERS spectra of drug resistant bacteria [8]. The model built for this study is composed by an unsupervised part, a stack of multiple autoencoders, used for dimensionality reduction and a supervised part used for classification of the representation outputted from the previous unsupervised part.

Prior to being fed into the model, some pre-processing algorithms were applied to

the dataset, mainly outlier detection by Isolation Forest algorithm and standardisation by Standard Normal Variate. The pre-processing applied is not mandatory for the model in question, however it can improve performance.

In the particular case of this study, two simple autoencoders where stacked instead of building a more complex autoencoder with multiple layers. The reasoning for this is to train the autoencoders separately and then join their trained encoders to form the so called stacked autoencoder. This method makes sure that each encoder is optimised for its individual representation of the input instead of fine-tuning simultaneously multiple layers that depend on each other in order to optimise the representation that results of the sequence of the previous representations. The used model employed two encoder layers that were individually trained. This was done by inputting the pre-processed spectra into the first autoencoder, which is composed of three layers, the input, with the dimensionality of the dataset, a significantly smaller hidden/middle layer, and the output, with the same dimensionality of the input. After the loss function is minimised, the weights are fixed and the decoder part is removed. To train the second autoencoder, which has the same initial architecture as the first autoencoder, the outputs of the encoder/middle layer of the first autoencoder are used as input and the training is done by trying to encode and reconstruct that input. Finally, when this loss function is minimised, only the encoder/middle layer are maintained. The result is a stacked autoencoder composed of the input layer that passes data to middle/encoding layer of the first autoencoder, that in its turn passes its data to the middle/encoding layer of the second autoencoder. This obtained representation is then used for classification using SoftMax activation. Since the goal is classify the spectra in either containing drug resistant bacteria or not containing, this classification layer only needs to have two outputs. After connecting the unsupervised and the supervised parts the model is trained, this time as a whole, in order to optimise the classification task by backpropagating the classification error and adjust the parameters. It is important to mention that regularisation parameters were used during training to prevent overfitting.

To measure performance of the model, 10-fold cross-validation was used, and the results were compared to classification by other machine learning methods. The SAE-based Deep Neural Networks (DNN) proved to be superior to the other algorithms with an accuracy of approximately $97.66\% \pm 0.26$ versus the $95.87\% \pm 0.01$ accuracy of the next best algorithm which was SVM. Although SVM gets more stable results than SAE, the latter gets on average the best results.

4

Solution and Implementation

The problem we are trying to solve, as a whole, can be divided into three main phases. The first is the collection and gathering of the samples of SERS spectra of different wines to build the dataset. Next is the processing of the data to obtain useful latent representation and its subsequent analysis by application of supervised and unsupervised machine learning algorithms. This study focuses on the latter two phases, as the dataset was already built and made available.

In this chapter we will go over the details of the implemented solutions. Firstly, we will go over the treatment of the data, mainly how the data was prepared to be used in the Deep Learning techniques. Secondly, we will describe some of the architectures and details of the Deep Neural Networks experimented with. The technique chosen in this study was the normal autoencoder, mainly because it has been proven successful in performing the needed tasks and because, to the best of our knowledge, no other deep learning technique has been applied to this dataset, nor to the context of spectra of wines. Lastly, we will describe the specifications of the machine learning algorithms and other techniques used to evaluate the quality of the representations found by the encoders.

All solutions were implemented using Python 3.8.5. TensorFlow 2.3.0 was used for the implementation and training of the Deep Learning algorithms. Mainly, we used the Keras module of TensorFlow since it is an opensource library dedicated to Deep Learning and Neural Networks.

4.1 Dataset

The dataset used in this thesis was obtained by a team of investigators at NOVA School of Science and Technology. It consists of a collection of spectra of Portuguese white wine samples from two different varieties, *Verde* and *Maduro*. In addition, multiple samples were collected from different Portuguese wine making regions, Northwest (Verde), Douro and Alentejo, represented in the map in Figure 1.1 as VN, MD and MA, respectively. All the *Verde* wines are from the Northwest region of Portugal, whereas the *Maduro* wines can be from either Douro or Alentejo.

The spectroscopy was obtained by firstly mixing the wine samples to a colloidal solution containing star-shaped silver nanoparticles. After an incubation period, in which the wine's components are adsorbed on to the nanoparticles' surfaces, a laser line of defined wavelength pointed at the mixture. The dispersed light is then captured and the different resulting wavenumbers and corresponding intensities are registered, this corresponds to the so called spectra.

Spectra were collected from 10 *Verde* wines, 10 *Maduro* wines from Douro and 9 *Maduro* wines from Alentejo, adding up to a total of 29 different wines. From each of these wines, multiple samples were collected and, from each of these samples, SERS spectra were obtained from multiple positions, making that each wine sample has, on average, 867 spectra associated. Since for each wine there are multiple samples and for each sample there are multiple spectra, we considered each of these spectra as a datapoint that represents its wine. A summary of the dataset can be seen in the bellow Table 4.1.

Table 4.1: Summary of the dataset of SERS spectra of Portuguese white wines

Type of wine	**Number of Wines**	**Samples**	**Spectra**	**Dimensionality**
Verde	10	35	10115	1050
Douro	10	38	10982	1050
Alentejo	9	31	8959	1050
Total	29	104	30056	-

The information present in the spectra is wavenumber and intensity. More specifically, 1015 pairs of wavenumber-intensity that represent the intensity of the waves composing the dispersed light after the laser with known wavelength is pointed at the prepared solution of wine and silver nanoparticles. The graph pictured in Figure 4.1 is a plot of SERS spectra, i.e. pairs of wavenumber-intensity, from two different wines in the dataset. To make sure the data are consistent, each spectrum was compared to confirm that the intensities were measured for the same wavenumbers across all spectra and that these were presented in the same order. Since the wavenumber differences were in the order of the thousandths, the gap was considered negligible and all spectra were treated as having the same wavenumbers, which are all in the interval $[61, 1836]$.

The graphics in Figure 4.1 are the intensities of the waves with different wavenumbers refracted when collecting the spectra of two different wines present in the dataset. The differences in the intensities of the same wavenumbers are what differentiates the wines. From the graphics it is clear that there is a difference between the wines from their spectra. This exemplifies what we will be exploring, which is the differentiation of wines based on the waves' intensities captured in SERS spectra. Considering that the spectra capture the intensities for the same wavenumbers across all wines and its samples, the only variable factor is the intensity of the waves, this means that the data characterising the spectra is the intensity, therefore, those are the values that we will use as data features in the study.

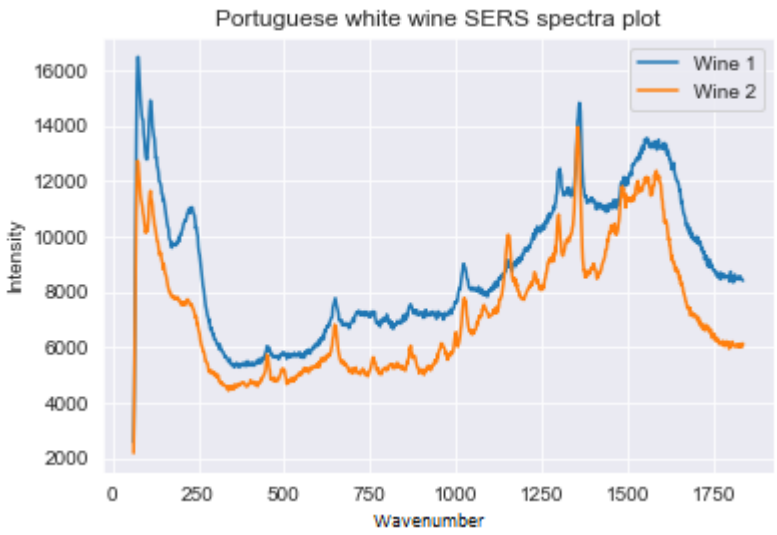

Figure 4.1: Plot of SERS spectra collected from two different Portuguese white wines

To this end, the created dataset was composed of the intensities of the waves, grouped by individual spectrum, which resulted in a set of 30056 one dimensional arrays of 1015 intensities. The arrays of intensities were then associated to the known labels of their wines, which were the type, region, alcohol level and, in some cases, the grape variety. Additionally, each different wine was attributed with a letter that represents that wine, so besides the mentioned labels, the arrays of intensities were also associated to the letter of the wine the example belongs to.

Prior to feeding the data to the deep learning models, the data were separated into training set and test set. Out of the 29 wines, the spectra from 9 different wines were selected to be part of the test set - for the sake of representation, 3 wines from each region were chosen - and the spectra of the remaining 20 wines formed the training set. This results in a division of approximately 70% of data for training and 30% of data for test, which translates to 22253 training spectra and 7803 testing spectra. The reasoning behind forming the test set using spectra from selected wines instead of using random spectra from the dataset is to prevent the algorithms from being trained and validated using very similar data. As mentioned, the spectra are from multiple positions of the same wine sample, and from the same wine there were multiple samples taken. In a perfect sample, all spectra from its multiple positions should result in the same intensities for the same waves since it is constituted by the same components in the same quantities, however that is not the case. It is expected that the spectra for the same sample are similar but not equal since SERS measures can be influenced by a number of factors, being one of them the fact that the components of the wine will not be uniformly distributed over the sample, nor will the nanoparticles adsorb them in the same way. Nevertheless, if the training and test sets have spectra from the same samples and wines, the similarity between them will be quite high and the model will not be able to generalise. This results in deep learning

Table 4.2: Average correlation between pairs of features in the dataset

Average of Correlations between column i and column $i+n$	
$avgCorrelation(i, i+\mathbf{1})$	0.99366
$avgCorrelation(i, i+\mathbf{2})$	0.99079
$avgCorrelation(i, i+\mathbf{3})$	0.98756
$avgCorrelation(i, i+\mathbf{4})$	0.98381
$avgCorrelation(i, i+\mathbf{5})$	0.97983
$avgCorrelation(i, i+\mathbf{6})$	0.97611
$avgCorrelation(i, i+\mathbf{7})$	0.97237
$avgCorrelation(i, i+\mathbf{8})$	0.96861
$avgCorrelation(i, i+\mathbf{9})$	0.96501
$avgCorrelation(i, i+\mathbf{10})$	0.96146
.	.
$avgCorrelation(i, i+\mathbf{50})$	0.83607
.	.
$avgCorrelation(i, i+\mathbf{100})$	0.69082
...	...

models that specialised in learning representations specific to the wines in the dataset. Furthermore, this is an ongoing study, so it is important to study and develop a model that can be used with other wines without needing to be adapted.

To further understand the relations between the features in the dataset, we calculated the correlation between the pairs of features. Some results are shown in Table 4.2. We measured the average correlation between all rows in a column i and all rows in the following n column, this is, $averageCorrelation(i, i+n)$.

As seen in the Table 4.2, the correlation decreases as the interval between columns increases, however, the correlation between a column and its next three columns is 99% and with its next five columns is 98%, which indicates a very close positive relation between these features. For this reason, in addition to using the full dataset to find low dimensional representations of the data, we will also use datasets where the features will be the average of every 3 columns of the original dataset or the average of every 5 columns of the original dataset.

Depending on the experiments being conducted, mean normalisation by row or column (as explained in the last paragraph of the subsection Data Preprocessing) was applied to the training and testing data to prevent big variations in feature numerical ranges that could have a negative effect on the training of the deep learning algorithms. Other types of preprocessing where experimented with, such as standardisation, min-max normalisation and l2 normalisation, also applied to row and column, however the mean normalisation was the preprocessing technique that showed more consistent results. Furthermore, some experiments were done using the data in its original form, with no preprocessing, and the results demonstrated that the big variations in the numerical intervals of the features negatively impacted the training of the DNNs, therefore the use of mean normalisation as preprocessing was chosen for the study.

4.1.1 Noisy Dataset

In order to implement and train the Denoising Autoencoders, a dataset consisting of noisy data was created. The noise was added by setting roughly 10% of the features to 0, randomly, for all spectra in the dataset, so that the model learns to infer the missing features from the surrounding values and from what it knows from the data. Using this method, no specific feature should be more impacted that others, since the features to be set to 0 are chosen randomly per spectra in the dataset, and the remaining data should be enough to compensate for the missing information. Ideally, the noise should not be constant across all training. It should in every batch affect different features randomly in every spectrum so that the model could build the robust representations by inferring patters in the information given. Due to the time restrictions associated with the thesis, that implementation was not done, however it should the contemplated in future studies.

4.2 Autoencoder

The implemented autoencoder networks consisted of an encoding part and a decoding part. The encoding part receives as input the data examples from the dataset, which are one dimensional arrays of 1015 features, 339 features or 203 features, depending on the used dataset (detailed in Section Dataset), and it outputs a representation of that input in the latent dimension defined. The encoder's output is then fed to the decoder part as its input. From there, the representation is reconstructed to match the original data.

From input to output, the autoencoders implemented varied in depth and width, but regardless of the number of hidden layers and the number of neurons in each layer, all models were constituted by an input layer followed by x fully connected, or dense, hidden Leaky Rectified Linear Unit (LReLU) layers that were then followed by the output layer.

Since the main goal is to reduce the dimensionality of the dataset, the autoencoders implemented were undercomplete autoencoders with an hourglass shape, where, from the input, the middle layers progressively got smaller until a desired dimension was achieved. Afterwards, in the decoder, the hidden layers got progressively bigger until it reached the original dimension in the output layer. In the implemented architectures, the decoder always mirrored the encoder's layers, meaning that the decoder had the same number of layers with the same number of neurons as the encoder, only in the opposite order. This type of architecture is pictured in Figure 2.4.

A variety of deep autoencoder architectures were experimented with. The motivation behind this is to find the architecture which best fits the data. The increase in depth and width of deep networks will lead to an increase in the number of trainable parameters and this is exactly what makes complex models more capable of learning complex features. However, more parameters to train means that the models need more training to learn. Another consequence of this growth is that the models start to learn very specific and complex features about the data used in training that do not generalise for data of

the same kind. This is called overfitting. On the other hand, when the model does not have enough trainable parameters, the model can underfit and not be capable of learning relevant and representative new features. So, following this line of thought, we experimented simple architectures and gradually increased their complexity and evaluated how the training and the predictions progressed with the increase in depth and width of the models. The Table 4.3 shows a summary of the experimented architectures.

Table 4.3: Summary of Implemented Autoencoder Architectures. Input and output layers are defined by the number of features in the dataset being used, which was 1015, 339 or 203.

Input	Hidden Encoder Layers	Latent Dimension	Hidden Decoder Layers	Output
1015 or 339 or 203	128x32x8x4	2	4x8x32x128	1015 or 339 or 203
	128x32x8	4	8x32x128	
	128x32	8	32x128	
	64x32x8x4	2	4x8x32x64	
	64x32x8	4	8x32x64	
	64x32	8	32x64	
	64x32x16x8x4	2	4x8x16x32x64	
	64x32x16x8	4	8x16x32x64	
	64x32x16	8	16x32x64	
	128x64x32x16x8x4	2	4x8x16x32x64x128	
	128x64x32x16x8	4	8x16x32x64x128	
	128x64x32x16	8	16x32x64x128	

As described earlier in the chapter, all layers are fully connected layers and, with the exception of the input and output layers, all layers have LReLU activation, with a negative slope coefficient set to 0.05. In the output layer, no activation was used because it is expected that the decoder reconstructs the data as closely to the input as possible. If there is an activation restricting the values of the reconstruction, like for example a *sigmoid* activation that restricts the output to a value between 0 and 1, which is not the case since the data is reshaped using mean normalisation, that does not guarantee the [0, 1] range of value, this restriction means that the learning process would be affected and the model would not be able to find good representations.

Denoising Autoencoders are standard autoencoders with a twist during training since the loss is not measured between the input data and the reconstructed data, instead this measure is done by comparing the reconstruction to the original data without noise,

The implemented Denoising Autoencoders were undercomplete autoencoders as well. No special implementation was made for the these types of autoencoders besides the adjustments necessary in training, covered in the following Section Autoencoder Training. Given all the architectures presented in the Table 4.3, a Denoising version was developed only for the ones using the full dataset, meaning the 1015 features as input and output layers.

4.2.1 Autoencoder Training

All models were trained using Mean Squared Error (MSE) as a loss function. This function compares the true value of the datapoint to the predicted value and measures the error between them. As explained in the section about Autoencoders, these models' predictions are a reconstruction of the input, so the loss is measured between the reconstruction and the original input. The evaluation of the training loss is done after each batch of 64 datapoints.

The optimiser used in all implementations of the standard autoencoders was the Adam optimiser. The optimiser itself has several hyperparameters that can modified and influence the training in their own way and, in these implementations, all parameters were set to the library's default values except for the learning rate, which controls the speed of the leaning process by managing the size of the optimisation step. All models struggled with the default learning rate, but when using the normalised data, the models struggled a bit more, so the value used was 0.0005 when using row normalised data and 0.0001 when using column normalisation, in order to achieve a more smooth training.

As for the Denoising Autoencoders, the training was very similar to the normal autoencoders with the exception of the input dataset.

In these experiments, techniques like Early Stopping were not utilised. Early stopping is used mainly to avoid neural networks from reaching a state where they learn too much and go into overfitting. It does so by stopping the training when it determines that the model is not learning. In the case of the autoencoder, if we were to use Early stopping, what would be considered as the model not improving is when the loss stops decreasing. However, since our goal is to train the autoencoder in a way that the encoder learns very detailed representations, even if in lower dimensions. This phase where the model is learning the small details that differentiate datapoints normally does not reflect in major changes in loss, hence this changes can be overlooked by the Early Stopping evaluation. Because of this, the models were trained past the point were loss decrease in not very noticeable so that the model can really specialise. The chosen number of epochs to experiment with and obtain results was 500, 1000 for all autoencoders.

4.3 Latent Representation Analysis

The last phase of the study is to evaluate the quality of the representations obtained using the methods described above. To this end, we used a variety of simple machine learning algorithms for both classification and clustering of the compressed data, and a more complex deep learning network, fed by the encoders, for classification of the data. The used techniques were already described in Classification Techniques, Clustering Techniques and Classifiers.

4.3.1 Machine Learning algorithms

For this part of the study, no real implementation was made since these algorithms are present in the Python libraries *scikit-learn* [34]. Given that the goal of this part is to make simple evaluations of the encoders results and compare the performance with other simple techniques, we opted to use these already implemented algorithms using the library's default parameters with slight adjustments in some cases. The chosen techniques are representative of the most popular algorithms used for these types of problems and are simple enough to really evaluate the compressed representations' quality by means of performance evaluations.

Classification algorithms used:

- **KNN** using the Euclidean distance and a K of 5
- **RF** with a default number of 100 trees

Clustering algorithms used:

- **K-Means** with a defined K of 2 and 3
- **BIRCH** with a set number of final clusters of 2 and 3
- **GMM** defined to identify 2 and 3 mixtures
- **DBSCAN** using $2 \times dimensionality\ of\ dataset$ as the minimal number of datapoints in the neighbourhood for it to be considered as cluster and determining the optimal ϵ (neighbourhood radius) using the Elbow Method

For comparison purposes, besides using these algorithms on the representations obtained from the trained encoders, these algorithms were also used on the dataset without dimensionality reduction and on dimensionaly reduced representation of the dataset obtained from PCA and t-SNE.

For the classification, the labels used were the wine's region and the wine's type, so the classification algorithms were trying to predict the region of the wine based on the representation given or the type of wine in question.

For the original dataset classification, the algorithms were trained with the training set and then the trained algorithms would try to predict the regions and types of the test set and we evaluated the accuracy of these predictions by comparing them to the real regions and types of the spectra in the test set.

In the classification using the dataset with dimensionality reduction via PCA or t-SNE, firstly, the dimensionality reduction algorithm was applied to the training and testing set. Then, the algorithms were trained with the reduced datasets and then the trained algorithms would predict the regions and wines of the test set.

Lastly, for the classification of the representation obtained from the trained encoders, firstly both train set and test set were fed to the trained encoders to obtain the representation, then the algorithms were trained with representation of the training set and the the trained algorithms would try to predict the regions and types of the representation of the test set and its accuracy was measured.

The clustering algorithms were only applied to the test set, for both test set with dimension reduction via PCA/t-SNE or the representation of the test set obtained from the trained encoders, if the datasets had dimensionality of 2, the algorithm would be applied directly to the data and the predicted clusters were plotted with different colours representing each cluster. If the dimensionality was over 2, in the case of the original dataset and some of the encoder representations, the algorithms would be applied directly to the data and, in order to get a visualisation of the results, the predicted clusters were plotted in 2-dimensions by applying PCA, ISOMAP and t-SNE and preserving the clusters each datapoint was attributed to.

4.3.2 Deep Learning Classifier

The classification via DNN was done by connecting the trained encoders to a combination of dense LReLU layers, which gradually decreased in size until the final layer that applied a SoftMax activation and outputted the probabilities of the spectra belonging to each class. The first layer of the classifier network had the same number of neurons as the output of the encoder and the last layer of this network had the same number of neurons as the number of classes. This classification was only applied to the encoders that generated 8-dimensional representations, and only for the classification of spectra by wine region, so the input and output sizes are 8 and 3, respectively.

Before performing classification or any training with this model, the encoder was fully trained and the neurons were frozen so that the values of the weights and other hyperparameters learnt during the autoencoder training remained unaltered. This way the training of the classifier was solely of the other layers and was based on the encoder representations as they were generated for the previous machine learning techniques. The objective is to see if the DNN classifier can benefit from the encoders' transformations better than the more simple classification algorithms and using this technique in combination with sensitive analysis allows for the study of which features are considered more relevant for each region prediction.

The architecture and training was consistent. As mentioned, the input layer consisted of 8 neurons, that fed into a 6 neuron layer. Since the networks showed a lot of overfitting, after this layer was a dropout layer that randomly set some of the neurons to zero. Next was a 4 neuron layer, followed by another dropout layer which in turn connects to the Softmax output layer of 3 neurons. This classification was performed with the normal encoders and the denoising encoders, for both normalisation by column and normalisation by row. For all models, except the ones using the denoising encoder with data normalised

by column, the dropout rate was 0.2 in both dropout layers. On the denoising model with column normalised data, the dropout rate was 0.1.

All models were trained using Adam optimiser with a learning rate of $5 \times 10^{-}5$. Since it was a multi-class classification problem, the chosen loss function was Categorical Crossentropy. For the training of these models, no fixed number of epochs was defined. Instead, the technique of Early Stopping was used. For the training of all models, the Early Stopping was monitoring the validation accuracy during training and had a patience of 20 epochs with no improvement before terminating the training.

4.4 Sensitivity Analysis

In order to understand how the encoders generate the representations, for each model it was performed a sensitivity analysis using Partial Derivative algorithm.

After the neural networks were trained, examples from the test set were fed to the models in order to output the prediction and obtain the gradients of the output with respect to the input. With all gradients, the Jacobian of the output with respect to the input features is obtained. This matrix has the shape *[number of output features, number of input features]* and it demonstrates for each output feature how it is influenced by each input feature.

For the case of the encoders, since we want to understand how the input features influence the output as a whole, we calculated a new array formed of the magnitudes of each column in the Jacobian, meaning that for each feature/column, the sensibility value would be the highest absolute value out of the sensitivities of the outputs with respect to that feature. With these values we can perform an quantitative analysis of how sensitive the representation is to changes in each of the input features.

For the case of the DNN classifier, the objective is to understand which input features most influence each of the outputs features, that are the probabilities of the example belonging to that certain class. To this end, each Jacobian was divided by row and the feature sensitivity values were grouped by output/class. With these values we can perform an quantitative analysis of how sensitive the probability of belonging to that class is to changes in each of the input features.

Additionally, to know clearly which features are the most important and which are less important or ignored, the features were ranked based of the sensitivity values obtained.

The sensibility analysis was performed for all examples in the test set. In the case of the encoders, these results where then grouped by wine the samples belong to by calculating the mean of the sensibilities of all samples belonging to that wine. Additionally, the results were grouped by wine region using the same method. Finally, the average of sensibilities of all samples of the test set was measured to have a general description of the encoder on the influence of the inputs on the outputs.

For the case of the DNN classifier, the sensibility analysis was also performed for all examples in the test set, however, it was only measured the average of sensibilities of all

samples of the test to have a general description of the encoder+classifier network on the influence of the inputs on the prediction of each class.

5

Results

In the Chapter Solution and Implementation, we went over the main experiments done in order to find a solution or solutions for the problem. In this chapter, we will present the results obtained.

All the results presented in this chapter were obtained using Python 3.8.5, TensorFlow 2.3.0 and Keras 2.3.0 in a laptop equipped with a NVIDIA® GeForce® MX250 GPU with 2GB of dedicated memory supporting CUDA 10.1.

As explained, the representations found by the encoders were evaluated using classification and clustering methods. This means that the representations are computed and then are used in these machine learning algorithms. Firstly, in order to establish a baseline for the results, we applied the same algorithms to the dataset but prior to dimensionality reduction and also to the dataset after dimensionality reduction using PCA, ISOMAP and t-SNE. The classification accuracy for the KNN classifier and RF classifier are in Table 5.1 and Table 5.2.

From the results we can conclude that the classifiers do not perform well with the original data. Since we are attempting to predict the regions from where the wine is from, which are 3, we can say from the accuracy that the prediction is close to random when using the dataset with all dimensions for region classification. The same goes for the wine type classification, where the wines can be either *Verde* or *Maduro*. This dataset is unbalanced in terms of wine type, with 33.7% of the spectra being from Verde wines and 66.3% being from *Maduro* wines, which leads to random prediction accuracy deviating a bit from 50%.

Table 5.1: KNN and RF classification accuracy for the classification of Wine Type and Wine Region using the original dataset of 1015 dimensions (features). The accuracies were measured after applying different preprocessing techniques to the dataset, namely normalisation per column and normalisation per row.

	Wine Type Prediction		Wine Region Prediction	
	KNN accuracy	RF accuracy	KNN accuracy	RF accuracy
Row Normalised Dataset	56.1%	69.2%	32.3%	41.3%
Column Normalised Dataset	51.8%	66.7%	30.8%	38.9%

Table 5.2: KNN and RF classification accuracy for the classification of Wine Type (A) and Wine Region (B) using PCA and t-SNE for dimensionality reduction of the original dataset. The accuracies were measured after applying different preprocessing techniques to the dataset, namely normalisation per column and normalisation per row.

A)

	KNN Accuracy for Wine Type Prediction				RF Accuracy for Wine Type Prediction			
	PCA			t-SNE	PCA			t-SNE
	2D	4D	8D	2D	2D	4D	8D	2D
Row Normalised Dataset	53.1%	48.3%	37.8%	56.0%	65.8%	62.2%	67.1%	59.8%
Column Normalised Dataset	55.5%	54.1%	53.3%	57.2%	62.8%	60.3%	61.3%	66.7%

B)

	KNN Accuracy for Region Prediction				RF Accuracy for Region Prediction			
	PCA			t-SNE	PCA			t-SNE
	2D	4D	8D	2D	2D	4D	8D	2D
Row Normalised Dataset	28.5%	28.1%	24.7%	40.1%	30.0%	20.0%	27.1%	32.6%
Column Normalised Dataset	38.4%	33.9%	32.1%	30.6%	36.4%	35.6%	29.4%	38.3%

Looking at the accuracy results obtained with PCA reduced datasets, we see that there is no major improvements in the accuracy and in some cases there is actually a decrease in accuracy. These results prove the expected, that the high dimensionality and complexity of the dataset does not allow standard machine learning techniques to perform well and that PCA is not a powerful enough tool to extract meaningful features for it to be beneficial for the study of relations among wines. The same goes for the t-SNE reduced datasets, since this is mainly a visualisation technique and not a dimensionality reduction technique, it does not perform for reductions other than 2 or 3 dimensions, for this reason we only show the accuracies for the dataset reduced to 2D with this technique and it was no better than PCA when it comes to classification of wine type and region.

In Figure A.1 and Figure A.2 are the confusion matrices for the classification by wine type and region using the full dataset with its original dimensions. The Figure A.1 and Figure A.2 show the same classification but using the dataset after dimensionality reduction by t-SNE. As it is visible in the matrices, in many cases the classification appears to be random while in others there is a tendency for the predictions to be of a specific class. This indicates that both classification algorithms cannot correctly discriminate by either type or region, and this does not change when using the full dataset or the dataset with reduced dimensions.

The graphs pictured in Figure A.5 show the plot of the test set datapoints after dimensionality reduction via application of PCA, ISOMAP and t-SNE, after the application of normalisation by columns and after normalisation by row. It is visible that, from the representations obtained from PCA or ISOMAP, regardless of the preprocessing, there is no clear separation between the datapoints, whether it be separation by the three wine regions, by the two wine types or any other identifiable separation. However, when analysing the representation obtained with t-SNE, after the application of normalisation by row (Figure A.5 - plot c.2)), it is noticeable that some datapoints are grouped mostly with points representing samples of the same wine. This means that, using only t-SNE to reduce dimensions and visualise the dataset we can already understand that, in some way, the spectra from the same wine are similar. As for the datapoints from different wines

that are close together, this can mean that either the wines are similar and thus produce similar spectra and similar representations, or the datapoints are actually distant but in a plane not captured by the t-SNE projection. Nevertheless, knowing that the samples represented with letters *b*, *e* and *o* are samples from three different *Verde* wines, the samples represented with letters *d*, *f* and *s* are samples from three different *Alentejo* wines and, lastly, the samples represented with letters *g*, *i* and *q* are samples from three different *Douro* wines, there is no visible connection amongst the representations and their wine type, region or any other known possible relation other than the wine itself. This fact is supported by the clusters formed when the clustering algorithms are applied to the data, before or after the application of the t-SNE. Some examples of these results are presented in the plots in Figure A.6.

From this Figure A.6 it is also visible how the dimensionality of the dataset when applying the algorithm has influence on the results. In the first row, the clustering algorithms were applied to the datasets with 1015 dimensions, and when the datapoints were attributed to the different clusters, the dimensionality was reduced to 2D and the datapoints were plotted in that reduced dimension while preserving the cluster attributed to it previously using different colours. On the other hand, on the last row, the clustering algorithms were applied to the dataset after the dimensionality reduction. When comparing the two rows you can see the formed clusters are quite different and contain different datapoints. This difference in results in the latter case can be because in the new representation, the new distance between the datapoints signifies that there are similarities between that datapoint, or can be that the dimensionality reduction technique reshaped the data in a way that caused information loss and, when reduced, the data does not relate in a way true to its original form.

5.1 Autoencoder Results

As mentioned in Solution and Implementation, all autoencoder architectures were experimented using 2, 4 or 8 neurons as middle layers, so that the encoder was trained to produce representations with 2, 4 or 8 dimensions. The autoencoders were trained for 500 and 1000 epochs and the datasets fed into these autoencoders varied in the preprocessing applied and features used, using normalisation by row or column for preprocessing and datasets composed of all original features, the mean of every three features or the mean of every five features of this original dataset.

In the case of the models trained with the dataset normalised by column, the autoencoders initially showed a lot of unstable training with big variations in the loss and difficulty converging. To achieve a more smoother training, the learning rate was adjusted to a smaller values so that the learning steps are smaller. This phenomenon was also present in autoencoders trained with the dataset normalised by row, but the instability was less severe and it was more common in the complex architectures. With a smaller learning rate, the training of the models using column normalised data converges on

average at around epochs 100 to 300, depending on the complexity of the dataset and architecture of the model, and the loss values stabilise without getting better or worse for the remaining epochs. In Figure 5.1 is a side by side of the loss plots of training using 0.001 learning rate and 0.0001 learning rate, left to right. The latter plot shows how the training converges very early on, suffering no major changes for the rest of the epochs.

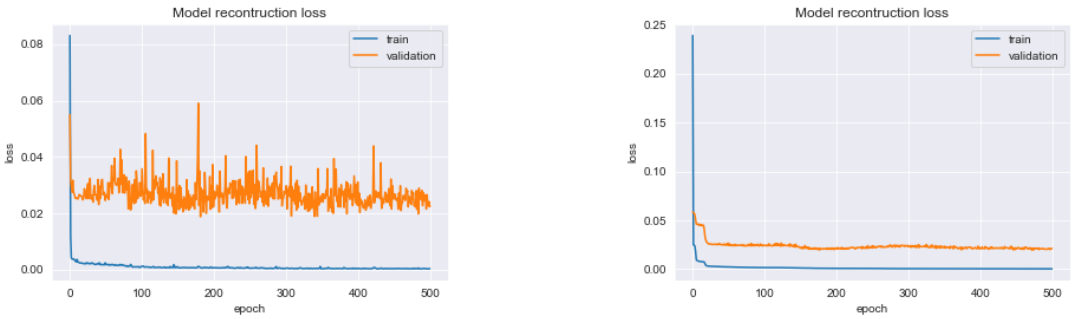

Figure 5.1: Loss plots of training of autoencoder using the column normalised dataset of means of every 5 features of the original dataset. On the left is the loss plot of training with 0.001 learning rate showing the instability in training caused by the big learning steps. On the right is the loss plot of training with 0.0001 learning rate that shows a smoother training that converges very early on.

When trained with row normalised data, in most models there is a slight worsening of the results at around epochs 400 to 600, depending on the complexity of the architectures. The validation loss is overall higher when more training is applied and the classifications tend to move towards the random predictions. This is because the model starts overfitting and adjusting too much to the training data, losing the general knowledge that should be also valid when applied to the test data. The overfitting can be confirmed by analysing the loss plots. To give an example of this behaviour, Figure 5.2 shows the loss plot of an autoencoder with architecture 203x128x64x32x16x**8**x16x32x64x128x203, trained with row normalised dataset of mean of every 5 features and, at around the mentioned interval of 400-600 epochs, the validation loss starts to increase whilst the training loss tends to decrease. In the cases that the model did not overfit, the loss tended to converge, similar to the models trained with datasets normalised by column.

Tables with the results for autoencoders trained for 1000 epochs using the dataset with the mean of every five features of the original dataset and the models trained for 500 and 1000 epochs with the full dataset and the dataset composed of the means of every three features of the dataset can be found in the Appendices at A.3. For the case of column normalised data, since the results of training over 500 or 1000 epochs are very similar, the results shown are only regarding the training over 500 epochs. Presented in Table 5.3 are the results relative to the experiments over 500 epochs with the models using data composed of the mean of every five features of the original dataset after application of row normalisation (Table 5.3 on the left) and column normalisation(Table 5.3 on the right). These were the models that overall produced better results.

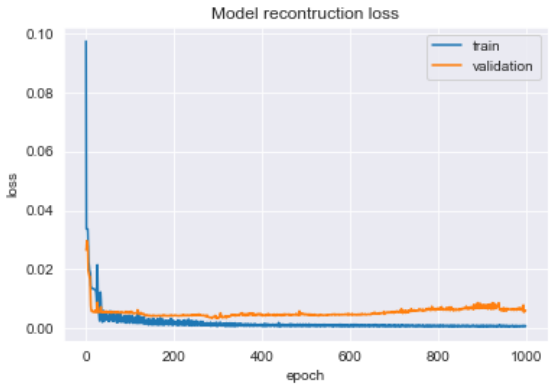

Figure 5.2: Loss plot of training of autoencoder with architecture 203x128x64x32x16x**8**x16x32x64x128x203 and using the row normalised dataset of means of every 5 features of the original dataset. Shows an increase in validation loss in the middle of training at around epoch 500, which indicates the model is overfitting

Table 5.3: Results of autoencoder training and encoder prediction for all architectures after training of 500 epochs with dataset composed of the mean of every five features in the original dataset. Validation loss is obtained from the comparison of the reconstructed data and the input data after the final epoch of autoencoder training. KNN and RF classification accuracy for the predictions of wine region (KNN acc Regions and RF acc Regions) and for the predictions of wine type (KNN acc Type and RF acc Type) given the representation of the spectra obtained from the trained encoder. On the left are presented the results for when the data was normalised by row, prior to being fed into the models, and on the right are presented the results for when was applied normalisation by column

Training with Dataset of Mean of Every 5 Features

	Normalised by Row - 500 epochs					Normalised by Column - 500 epochs				
	Val Loss	KNN acc Regions	RF acc Regions	KNN acc Type	RF acc Type	Val Loss	KNN acc Regions	RF acc Regions	KNN acc Type	RF acc Type
203x128x32x8	0.0028	28.5%	35.4%	52.2%	65.6%	0.0254	26.5%	36.4%	53.0%	66.1%
203x128x32x8x4	0.0040	31.6%	43.7%	60.3%	63.1%	0.0482	32.4%	44.6%	52.7%	66.2%
203x128x32x8x4x2	0.0050	27.2%	23.1%	54.4%	66.4%	0.0479	40.1%	38.9%	63.7%	66.7%
203x64x32x8	0.0031	32.4%	44.3%	57.7%	64.6%	0.0199	31.9%	41.3%	55.9%	67.3%
203x64x32x8x4	0.0033	33.3%	31.1%	56.0%	67.2%	0.0230	30.7%	42.3%	57.3%	67.6%
203x64x32x8x4x2	0.0056	27.1%	29.1%	48.3%	65.2%	0.0395	42.4%	37.7%	61.8%	66.7%
203x64x32x16x8	0.0032	32.7%	35.5%	61.5%	64.9%	0.0188	31.6%	46.7%	56.5%	66.2%
203x64x32x16x8x4	0.0051	36.6%	33.6%	57.1%	65.8%	0.0273	36.2%	41.0%	57.2%	66.1%
203x64x32x16x8x4x2	0.0050	25.9%	31.8%	55.5%	65.6%	0.0445	42.7%	39.2%	61.6%	66.7%
203x128x64x32x16x8	0.0030	35.7%	35.3%	58.7%	63.0%	0.0159	32.7%	35.2%	52.9%	66.7%
203x128x64x32x16x8x4	0.0039	24.6%	32.0%	55.5%	62.4%	0.0236	30.5%	36.8%	49.0 %	66.5%
203x128x64x32x16x8x4x2	0.0052	27.7%	28.2%	51.5%	56.3.3%	0.0512	39.8%	38.8%	61.5%	64.7%

By looking only at the results in Table 5.3, choosing the best model is difficult since there is no model that is clearly the best on all measures. While some models' representations perform best with the KNN classifier and lack on the RF classifier, other representations are the opposite. This can be due to the nature of the classifier algorithm. The KNN uses the Euclidean distance and data proximity as a basis for its predictions and RF uses multiple decision trees that together make predictions, so it is possible that, in some autoencoder architectures, the learnt representation is more suited for one algorithm than the other, maybe because the representation is better in terms of the distribution of the data, whether by preserving the proximity amongst related datapoints or even bringing

them even closer, or maybe the new features in the new representation allow for better distinction.

As mentioned, the Table 5.3 shows the results for the best models when using row normalised data and when using column normalised data. However, these models were the best in different ways, given their preprocessing. When trained with row normalised data, the loss between the reconstructed representation and the original representation is smaller, this can indicate that the representation is more faithful to its original state and it preserves the important information in a way that is able to be reconstructed with less loss. When the autoencoders were trained with data with row normalisation, in spite the classification accuracies not being the highest, the representations found were much better in terms of data visualisation.

Other than the validation loss, another difference between the results of row normalised data models and column normalised data models is that the latter obtained slightly better accuracies when classifying the representations by region or type. Although quite close to random accuracies, seen when the algorithms do not learn how to differentiate the classes from the data and, therefore make random decisions, there is a slight increase in the values. In this case of the column normalisation, the accuracies are worse when the architectures middle layer have more dimensions, which means the representations have more dimensions. This was not expected since the information does not have to be so compressed, so it is possible that the representation with more dimensions have more information. This is corroborated by the loss values being lower in these cases. But, in reality, that did not translate in the classification maybe because the extra information in the representations did not relate the spectrum to the region or type of wine it belonged to, thus having a negative effect on those classifications.

Figure 5.3 shows the confusion matrices for the classification of the representations obtained from the encoder with 203x128x64x32x16x**8** architecture, present in Table 5.3, trained with the column normalised dataset of means of every 5 features. From matrices a) and b) we can conclude that when classifying by type there is a big tendency towards *Maduro* wines, especially in the RF classification of wine type, predicting over 99% of examples as *Maduro* wine. Since the data is unbalanced, with 2/3 of the data belonging to the *Maduro* wine class, it is possible that the examples from Verde wines were not enough for the autoencoder to learn how to differentiate the Verde wines from the *Maduro* wines. Another hypothesis is that the representation is not good for wine type classification because it does not have any information that differentiates the spectra. Regarding the wine region classification, both algorithms struggled as well. Knowing that all *Maduro* wines are either wines from the *Douro* region or from *Alentejo*, you can see that the tendency for *Maduro* wines seen in the type classification did not continue in wine region classification, probably because the data is more balanced. However, from the confusion matrices we see that there are many *Douro* wines being mislabelled as *Verde* wines and *Alentejo* wines in the KNN classification presented in matrix c) and many *Douro* and *Alentejo* wines being mislabelled as *Verde* wines in the RF classification in matrix d). This

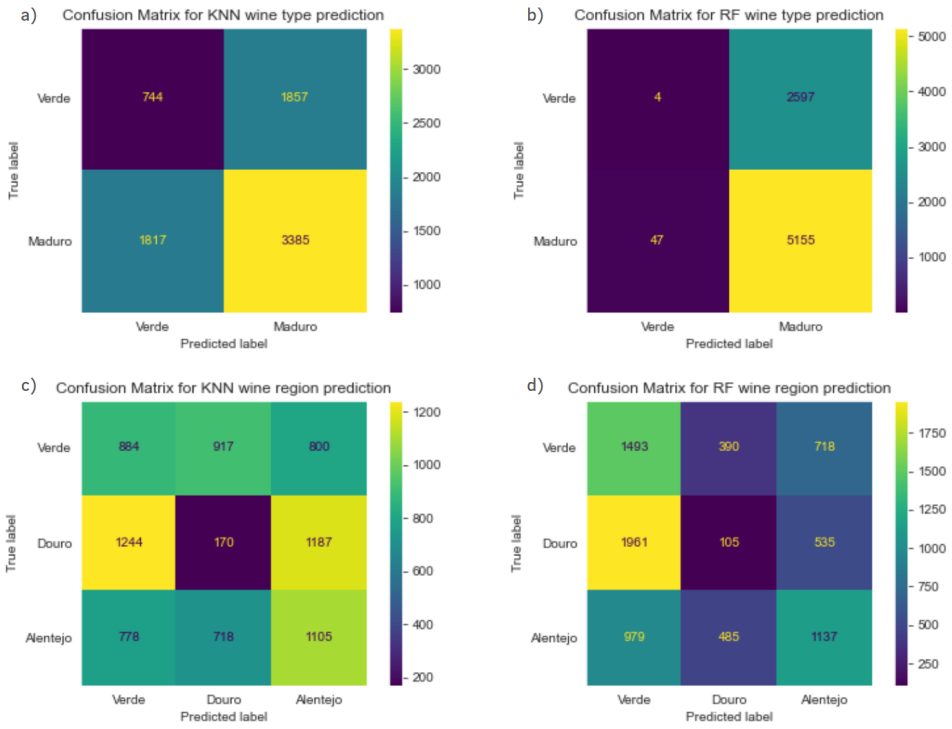

Figure 5.3: Confusion Matrices for type and region classification of representations obtained from the encoding by the trained 203x128x64x32x16x**8** encoder. The encoder was trained with the column normalised dataset of means of every 5 features and the confusion matrices presented are of: a) KNN wine type classification; b) RF wine type classification; c) KNN wine region classification and d) RF wine region classification.

raises the question if there are any similarities between these wines, captured by the encoder, that would lead to datapoints from these wines being closer together than with the datapoints of other wines of its own region. This can also explain why the wine type classification is so biased to Maduro wines. If, in addition to the dataset being unbalanced in favour of Maduro type, there is a similar component between the *Douro* wines in the Maduro class and the Verde wines in the Verde class that could lead the classifier to consider that combination as a defining factor for classification and see all wines as similar, thus classifying them all or a major part as the same type.

When comparing these classification results with the corresponding classification results of the original dataset in Table 5.1 and the classification results of the classification results of the original dataset after dimensionality reduction by PCA and t-SNE in Table 5.2 we see that there is an increase on accuracies from the encoder representations, however it is not a significant increase. This means that the spectra of wines do not allow for a straightforward classification on the type of wine they represent and that the autoencoder was not capable of learning a representation of the spectra that facilitated and improved that classification. When comparing the confusion matrices in Figure 5.3 with the confusion matrices of the original dataset classification in Figure A.1 and Figure A.1, the same conclusion is drawn.

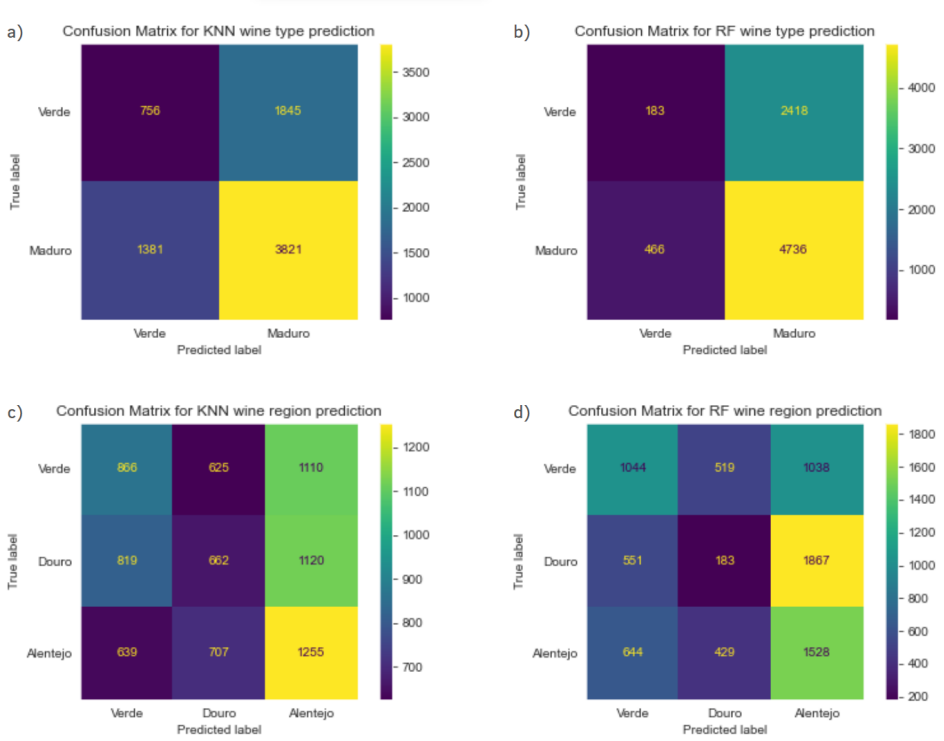

Figure 5.4: Confusion Matrices for type and region classification of representations obtained from the encoding by the trained 203x128x64x32x16x**8** encoder. The encoder was trained with the row normalised dataset of means of every 5 features and the confusion matrices presented are of: a) KNN wine type classification; b) RF wine type classification; c) KNN wine region classification and d) RF wine region classification.

In Figure 5.4 are presented the confusion matrices relative to the classification results of the 203x128x64x32x16x**8** encoder after being trained with the row normalised dataset of means of every 5 features. Similarly to the representations from the column normalised data, there is a tendency for *Maduro* wine when classifying the wine type, although in the case of the RF type classification it is not as unbalanced. Unlike the classification of representations from the column normalised data, the wine region classification with the row normalised dataset shows a tendency for *Alentejo* wines. This can mean that the classifiers found in the encoder representation a characteristic or a combination of that is shared amongst wines from all regions but more common in *Alentejo* wines, leading to most wines being classified as *Alentejo*.

When comparing these classification results with the corresponding classification results of the original dataset in Table 5.1 and the classification results of the classification results of the original dataset after dimensionality reduction by PCA and t-SNE in Table 5.2 we see that there is a small increase on accuracies of the KNN classification of the encoder representations, but no improvement on the RF classification. By further analysing the confusion matrices of the original dataset classification in Figure A.2 and Figure A.2 and comparing to Figure 5.4 its visible that the improvement is accuracy is only because there is less of a tendency for predicting Alentejo wines and the predictions

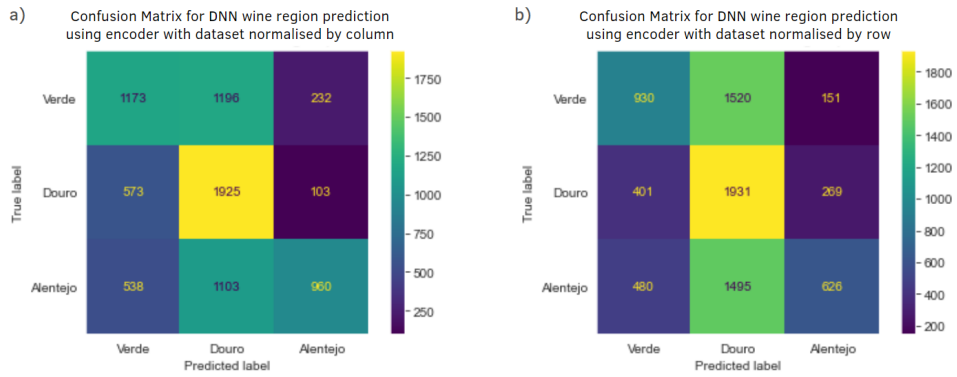

Figure 5.5: Confusion Matrices for region classification obtained by the DNN being fed by the trained 203x128x64x32x16x8 encoders. The encoders and encoder+classifying network combination models were trained with: (a) the column normalised dataset of means of every 5 features and (b) the row normalised dataset of means of every 5 features.

are more random.

Both encoders from Figure 5.3 and Figure 5.4 were connected to classifier networks and trained to predict the regions of the wines the spectra belong to. The classifier using the encoder trained with the dataset normalised by column finished training after 79 epochs and was able to predict the regions of the test set with 52% accuracy. The classifier using the encoder trained with the dataset normalised by row finished training after 60 epochs and was able to predict the regions of the test set with 45% accuracy. Considering that the region classification of the encoder's representations using the KNN and RF algorithms only obtained accuracies of 35.7%, 35.3% and 32.7%, 35.2%, respectively, the accuracies obtained from these models are a big improvement. The confusion Matrices for these classifications are presented in Figure 5.5.

From analysing the confusion matrices, it is visible that there is a slight tendency for Douro wines, which is not consistent with the classification of the representations by the machine learning encoders. Despite this tendency, the number of correct predictions of each class is bigger that the number of incorrect predictions, which means that the combination of the encoder's transformations and the classifier network was able to find relations among the features that allowed to distinguish to a certain extent the samples of different wine regions, something that the simple machine learning algorithms were unable to do with the full dataset, the dimensional reduced dataset, nor with the representations found by the encoders.

The plots in Figure 5.6 show the representation of the row normalised test data obtained from the 203x128x64x32x16x**8** encoder. Since the encoding is an 8-dimensional representation and cannot be directly plotted for visualisation, this representation was submitted to dimensionality reduction by multiple algorithms in order to fully analyse the representation. In these plots, the datapoints are coloured depending on the wine that sample belongs to, and that shows that the representations found relations between the

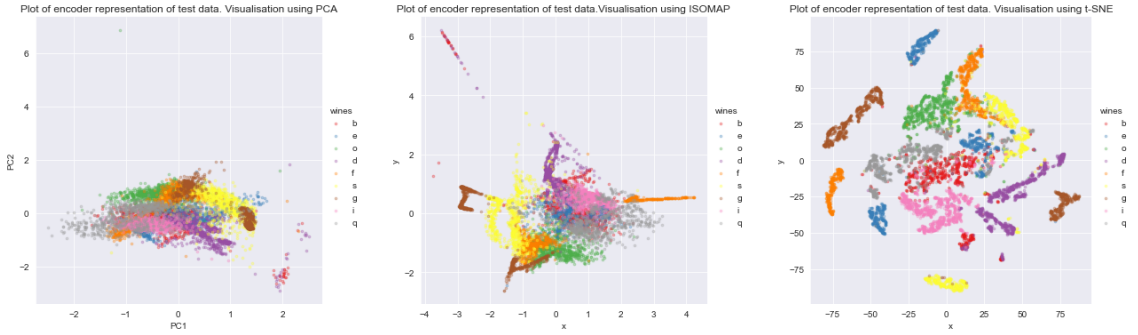

Figure 5.6: Plot of encoder 8-dimensional representation of test data after dimensionality reduction by PCA, ISOMAP and t-SNE, respectively from left to right. The 8D representation was obtained from the 203x128x64x32x16x**8** encoder after training with the row normalised dataset of mean of every 5 features. Points are coloured by wine that sample belongs to.

spectra belonging to the same wines. This is clearly visible in the plot using t-SNE (Figure 5.6.(c)), where the points clearly form groups mostly made of spectra from the same wines. Comparing with the corresponding plot of the original dataset in Figure A.5.c.2) we see that the datapoints have a similar distribution in both plots, however, in the encoder representation, the agglomerates of points from the same wines appear slightly more disperse and less concentrated then in the original dataset. That also applies to the plots using PCA and ISOMAP, suggesting that in the representation by the encoder the points are really more dispersed, and is not only the t-SNE algorithm separating them. Since the representations from the encoders and the original dataset are very alike, its safe to say that the encoder was able to find a latent representation of the spectra that preserved the information, however the encoder was unable to learn new features and relations hidden in the complexity of the spectra that would allow to, from a visual analysis, identify new relations among wines besides the relation between spectra and the wine the sample belonged to.

Using the Partial Derivative algorithm, a sensitivity analysis on this trained encoder was performed. The Jacobians containing the gradients of the outputs with respect to the inputs were obtained for all spectra in the test set and the results were grouped and the average was calculated to know on average which are the features that most influence the representations obtained by the encoder. Since this model was trained with the dataset whose features are the mean of every 5 consecutive features of the original dataset, each feature is equivalent a 5 original features, meaning that each feature is the average of intensities of 5 wavenumbers. Out of the 203 input features, the 10 most influential features and the 10 least influential features are in Table 5.4. The table contains the ranking attributed to that feature based on the sensitivity value, which is the average of the magnitudes of the jacobians of the output w.r.t. the input of all examples in the test set. Finally, the column wavenumbers shows the wavenumbers that correspond to the feature in question.

Table 5.4: Sensitivity Analysis of 203x128x64x32x16x8 encoder trained with row normalised dataset composed of the means of every 5 features of the original dataset. Results obtained from the average of the sensibilities of all spectra in the test set.

Ranking	Sensibility	Wavenumbers (cm^{-1})	Ranking	Sensibility	Wavenumbers (cm^{-1})
1	0.4981	1361, 1359, 1357, 1356 1354	**194**	0.0500	1460, 1458, 1456, 1455, 1453
2	0.3911	1352, 1351, 1349, 1347, 1346	**195**	0.0497	995, 993, 991, 989, 988
3	0.3272	737, 736, 734, 732, 730	**196**	0.0488	610, 608, 606, 605, 603
4	0.3160	880, 878, 876, 875, 873	**197**	0.0487	1159, 1157, 1155, 1153, 1152
5	0.2627	89, 87, 85, 83, 81	**198**	0.0465	1176, 1174, 1172, 1170, 1169
6	0.2626	889, 887, 886, 884, 882	**199**	0.0443	977, 975, 974, 972, 970
7	0.2448	728, 727, 725, 723, 721	**200**	0.0434	1193, 1191, 1189, 1188, 1186
8	0.2439	108, 106, 104, 102, 100	**201**	0.0432	1209, 1207, 1206, 1204, 1202
9	0.2418	1327, 1326, 1326, 1322, 1321	**202**	0.0417	1210, 1208, 1206, 1205, 1203
10	0.2398	98, 96, 95, 93, 91	**203**	0.0388	674, 672, 670, 669, 667

The wavenumbers, presented in Table 5.4, with the best rankings are the wavenumbers which intensities most influence the representations from the encoder, meaning that the resulting representations suffer big changes with little changes in these intensities. As mentioned, the plots in Figure 5.6 show that the encoder generated representations where the relations between spectra of the same wine are visible in the agglomerations of the datapoints. This means that these wavenumbers represent something which allows the encoder to create representations where different wines are differentiated and the same wines are matched. On the other hand, the wavenumbers with the worst rankings are not a big influence on the resulting representations. These wavenumbers are possibly ignored because their values are constant across the different wine spectra, and therefore do not add any information to the encoder. On the contrary, the values which are of most importance are the ones that vary, and if there is a pattern in the way those features change than that is an identifiable marker that can be used to distinguish spectra and find relations.

The sensitivity analysis on the DNN classifier model using this encoder was also performed using the Partial Derivative algorithm. Similarly to the encoder analysis, the Jacobians containing the gradients of the outputs with respect to the inputs were obtained for all spectra in the test set and the results were grouped by output, meaning that they were grouped by predicted class. The average was then calculated to know on average which are the features that most influence the prediction of each class. Figure 5.7 shows the plot of the average sensitivity of each of the outputs, which are the prediction of probability of the spectra belonging to that class, to each of the inputs. It is visible from the graph that the Verde, Alentejo and Douro predictions follow a similar pattern of feature importance, however, it appears that the less important features are equally unimportant, but the most important features have different degrees of influence on the output. What this means is that, for example, the feature 115, despite being the most influential feature for all outputs, changes in these features would affect more the Verde prediction, than the Alentejo prediction or the Douro prediction, that are less sensitive to its changes.

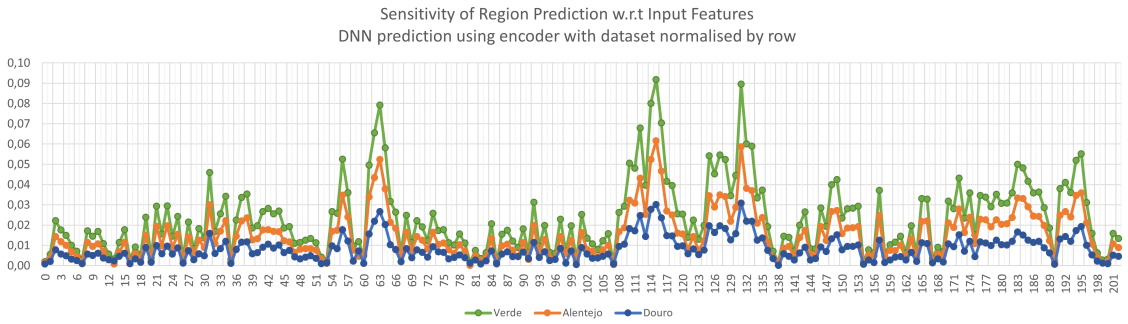

Figure 5.7: Sensitivity values for classifier DNN fed by encoder trained and using row normalised data. Each feature in the horizontal axis represents the mean of 5 consecutive features of the original dataset

Table 5.5: Top Sensitivity values for outputs of classifier DNN fed by encoder trained and using row normalised data.

Ranking	Sensibility Verde Pred.	Sensibility Alentejo Pred.	Sensibility Douro Pred.	Wavenumbers (cm^{-1})
1	0.0917	0.0616	0.0308	880, 879, 877, 875, 873
2	0.0895	0.0586	0.0278	737, 736, 734, 732, 730
3	0.0801	0.0524	0.0278	889, 887, 886, 884, 882
4	0.0791	0.0523	0.0267	1327, 1326, 1324, 1322, 1321
5	0.0703	0.0467	0.0236	872, 870, 868, 866, 864

The most important features in this model are represented in Table 5.5. Out of these top 5 features, only the top 2 are present in the top of the important features of the encoder, and are represented in third and fifth place of the ranking in Table 5.4. Interestingly, with the exception of the third and fifth most important features for the encoder's representation, the remaining features considered as most influential are not even ranked as top half of influential features in none of the outputs of the DNN classifier. This indicates that the network in the classification training adapted, and that the features the encoder considered most important in its representations are not considered important in the prediction of classes.

The plots in Figure 5.8 and in Figure 5.9 show the clusters identified by the clustering algorithms. All the algorithms were applied to the same representations obtained by the encoders, shown in Figure 5.6, and the dimensionality reduction algorithms were applied after that, preserving the clusters the datapoints were attributed to. A detail important to mention is that all clustering algorithms presented in Figure 5.8 are configured to find 3 clusters prior to being applied to the data in question. This meas that, no matter the representation presented, the algorithms will find three and only tree clusters. However, the DBSCAN algorithm, whose results are in Figure 5.9, has no such restriction. This algorithm, for the case of the representation from this specific encoder, had predefined 16 as a minimal number of neighbours each datapoint must have in order to define a cluster and 0.75 as ϵ, a variable defined using the help of the elbow method. Without the restriction, this algorithm identified 3 different clusters as well, a big cluster consisting of the majority of the data, and two more clusters consisting of points which are mostly

deviated from the big agglomerate of datapoints. Those points, that belong to wines *b* and *d* are a *Verde* wine and a *Alentejo* wine, respectively, so they do not share neither the region nor the type of wine. It is possible that the algorithm only identified and separated the outliers from the rest of the data, but within the outliers detected from wines *b* and *d*, it appears to differentiate correctly between both wines in the most cases, with green representing wine *b* and blue representing wine *d*. None of the clustering algorithms found any clusters that demonstrated any relation among the wine samples. There is no identifiable pattern in the datapoints within each cluster and no identifiable differentiation pattern between datapoints from different clusters

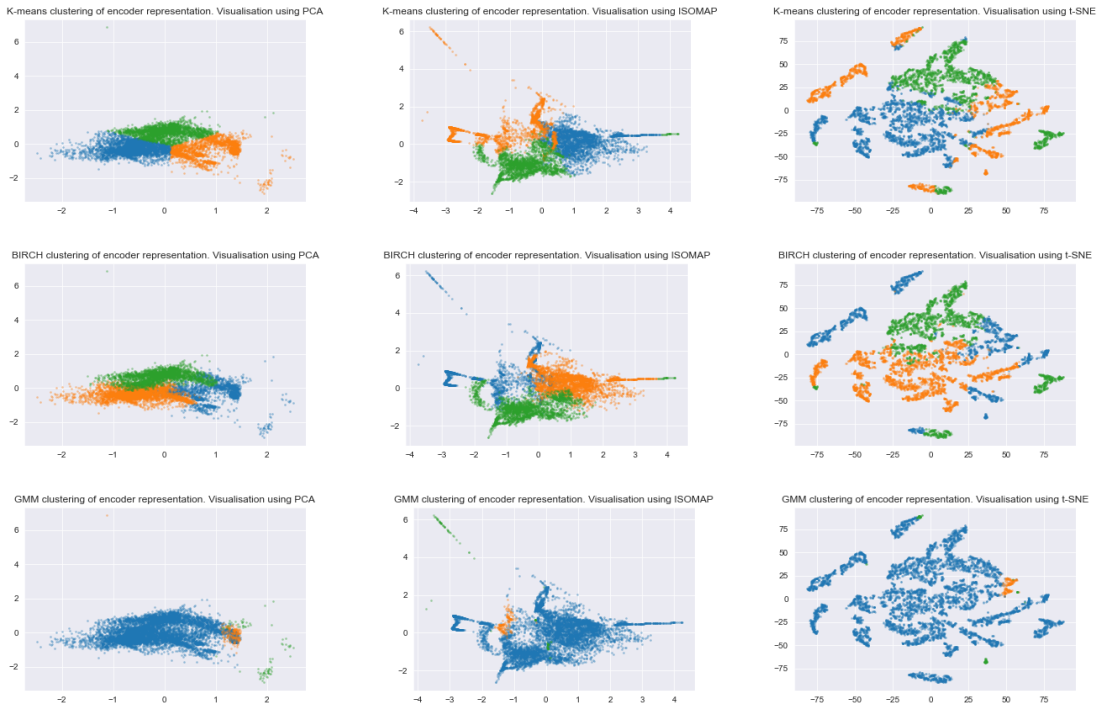

Figure 5.8: Plots of identified clusters on encoder 8-dimensional representation of test data. The 8D representation was obtained from the 203x128x64x32x16x**8** encoder after training with the row normalised dataset of mean of every 5 features. On the top row are the clusters formed by K-means, in the middle row are the clusters from BIRCH and on the bottom row are the clusters from GMM.

Comparing the results from models trained and fed with row normalised data in Figures 5.6, 5.8 and 5.9 to the results obtained from the models trained and fed with column normalised data in Figures 5.10, 5.13 and 5.14 it is clear the difference the type preprocessing of the data has in the results.

In Figure 5.10 are the plots of the representation of the column normalised test data obtained from the 203x128x64x32x16x**8** encoder that were dimensionaly reduced by PCA, ISOMAP and t-SNE in order to be visualised. Unlike the representations from encoding of row normalised data, from the t-SNE plot there is no visible relation between samples from the same wine as the points are scattered all over. However, looking at the plots

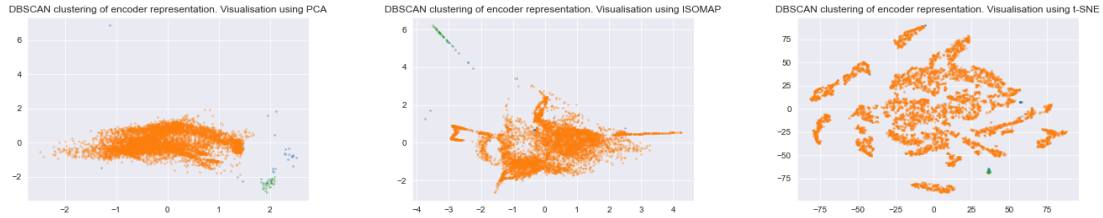

Figure 5.9: Plots identified clusters of encoder 8-dimensional representation of test data using DBSCAN with 16 as the minimal number of neighbours to form a cluster and 0.75 as ϵ. The DBSCAN algorithm was applied on the 8D representation from the encoder and the dimensionality reduction was applied after for visualisation purposes.

obtained with PCA and ISOMAP we can speculate that the datapoints group by wine or show some other kind of grouping in a plan not captured by the PC1 and PC2 plot. To confirm this, it was plotted the PCA's PC2 and PC3 plot, shown in Figure 5.11, however it remains unclear if any relation is found as the points appear very close and mixed.

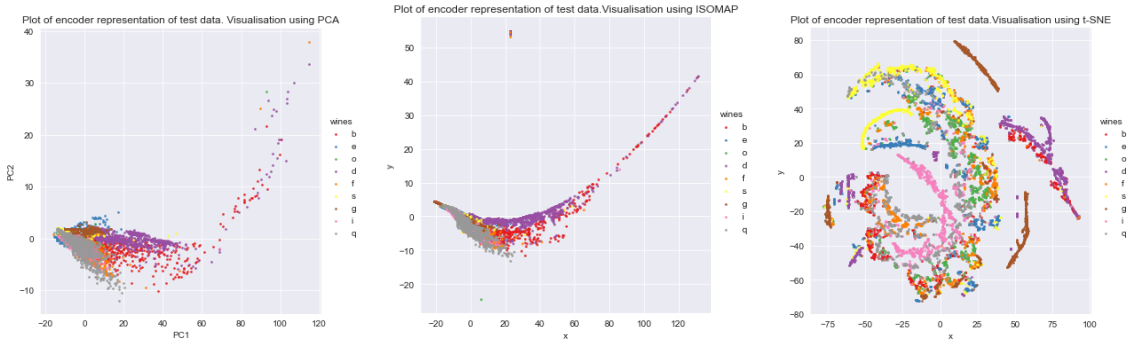

Figure 5.10: Plot of encoder 8-dimensional representation of test data after dimensionality reduction by PCA, ISOMAP and t-SNE, respectively from left to right. The 8D representation was obtained from the 203x128x64x32x16x**8** encoder after training with the column normalised dataset of mean of every 5 features. Points are coloured by wine that sample belongs to.

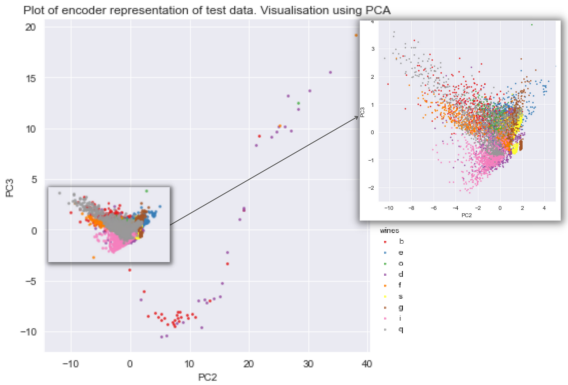

Figure 5.11: Plot of Principal Components 2 and 3 of encoder 8-dimensional representation of column normalised test data

In the case of this encoder trained with the column normalised dataset, the Partial Derivatives algorithm was also applied in order to do a sensitivity analysis. Similarly to the analysis mentioned earlier, the Jacobians were calculated for all examples in the test set and the average was calculated. The results for the 10 most important features and the 10 least important features are in Table 5.6.

Table 5.6: Sensitivity Analysis of 203x128x64x32x16x8 encoder trained with column normalised dataset composed of the means of every 5 features of the original dataset. Results obtained from the average of the sensibilities of all spectra in the test set.

Ranking	Sensibility	Wavenumbers (cm^{-1})	Ranking	Sensibility	Wavenumbers (cm^{-1})
1	0.7993	1558, 1556, 1554, 1553, 1551	**194**	0.1178	1686, 1684, 1683, 1681, 1680
2	0.7740	1525, 1523, 1522, 1520, 1519	**195**	0.1171	960, 958, 956, 954, 953
3	0.7638	1590, 1588, 1587, 1585, 1583	**196**	0.1147	1443, 1442, 1440, 1438, 1437
4	0.7336	764, 763, 761, 759, 757	**197**	0.1146	147, 145, 143, 141, 139
5	0.6507	1598, 1596, 1595, 1593, 1591	**198**	0.1135	1184, 1182, 1181, 1179, 1177
6	0.6134	1820, 1818, 1817, 1815, 1814	**199**	0.1110	339, 338, 336, 334, 332
7	0.6026	1038, 1036, 1035, 1033, 1031	**200**	0.1088	1193, 1191, 1189, 1187, 1186
8	0.5914	1344, 1342, 1341, 1339, 1337	**201**	0.0986	1460, 1458, 1456, 1455, 1453
9	0.5901	1549, 1548, 1546, 1545, 1543	**202**	0.0970	1003, 1002, 1000, 998, 996
10	0.5798	773, 772, 770, 768, 766	**203**	0.0919	1073, 1071, 1069, 1067, 1066

The first apparent thing about this table is that many of the features considered as most important correspond to the intensities for wavenumbers within the range [1519, 1598] and others vary close to this range, which indicates that there is a big influence of that portion of the wavenumbers. Additionally, none of the most important features in this encoder are considered as important in Table 5.4. Two less important features in this table are present in Table 5.4, which are the features ranked as 194 in this table and 202 in Table 5.4 and 199 both tables. Since the representations are so different, it was expected that most influential features in the the different encoders were nor the same.

The sensitivity analysis on the DNN classifier model using this encoder was also performed using the Partial Derivative algorithm. Figure 5.12 shows the plot of the average sensitivity of each of the output features to each of the input features. It is visible from the graph that the Alentejo and Douro predictions are very similarly influenced by the features. As for the Verde wine prediction, the importance of the features follows the same pattern as the other classes, however, it appears that the most important features have more influence on the output. Given that this classifier was able to classify the test set with a 52% accuracy, it is possible that the network captured the similarities between wines of Alentejo and Douro region, since they are same type of wine, and Verde wine is another wine type with a possible different composition. If these similarities are captured in the spectra then they are possibly captured in the network and allow for an accurate prediction

The most important features in this model are the features in Table 5.7. Out of these top 5 features, all are present in the top of the important features of the encoder, and are represented in rank 3, 1, 6, 4 and 9, respectively, in Table 5.6. This indicates that, either the network in the classification training agreed that those features are in fact good

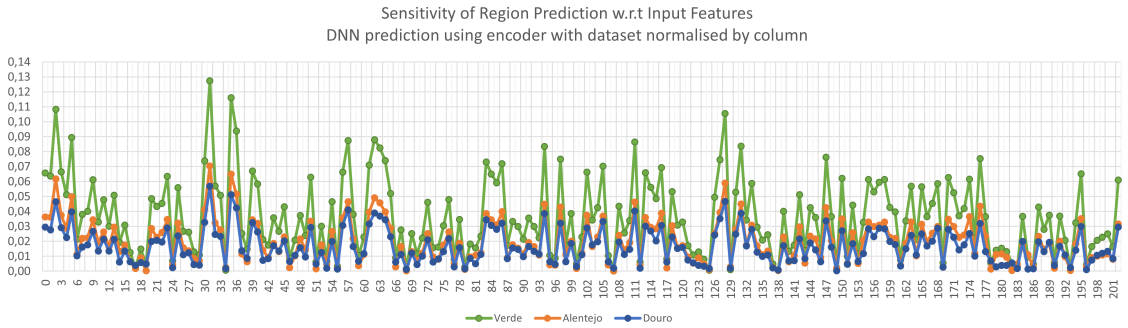

Figure 5.12: Sensitivity values for classifier DNN fed by encoder trained and using column normalised data. Each feature in the horizontal axis represents the mean of 5 consecutive features of the original dataset

Table 5.7: Top Sensitivity values for outputs of classifier DNN fed by encoder trained and using colum normalised data.

Ranking	Sensibility Verde Pred.	Sensibility Alentejo Pred.	Sensibility Douro Pred.	Wavenumbers (cm^{-1})
1	0.1273	0.0704	0.0568	1590, 1588, 1587, 1585, 1583
2	0.1161	0.0649	0.0512	1558, 1556, 1554, 1553, 1551
3	0.1083	0.0618	0.0467	1820, 1818, 1817, 1815, 1814
4	0.1055	0.0588	0.0465	764, 763, 761, 759, 757
5	0.0937	0.0515	0.0423	1549, 1548, 1546, 1545, 1543

indicator of the classes the examples belong to, or the network did not adapt a lot, and simply used the transferred knowledge of the encoder to perform a classification. Since the accuracy improved in this classification, it is most likely that those features are good indicators of wine region.

The plots in Figure 5.13 and in Figure 5.14 show the clusters identified by the clustering algorithms for the representation from the column normalised data after encoding. The same procedure as the clustering of the representation of row normalised data was followed. Figure 5.13 shows the clustering algorithms restricted to finding 3 clusters. The different algorithms give different results, which is normal since they have different basis for their decisions however, no similarity, difference nor relation can be found from the datapoints in each identified clusters of any of the algorithm.

The Figure 5.14 shows the result of DBSCAN, the only algorithms not bound to find a certain number of algorithms. For the column normalised data representation, the DBSCAN parameters used were 16 as had the minimal number of neighbours each datapoint must have in order to define a cluster and 2 as ϵ, variable defined using the help of the elbow method. Without being restricted to find a defined number of cluster, when applied to this representation, the algorithm found only two clusters, one containing the majority of the datapoints and another containing the more dispersed datapoints. This result is similar to the DBSCAN result of the row normalised data encoders in Figure 5.9. Since DBSCAN is a density based clustering algorithm, that indicates that within the distribution of the datapoints in the 8-dimensional manifold, the algorithm can distinguish

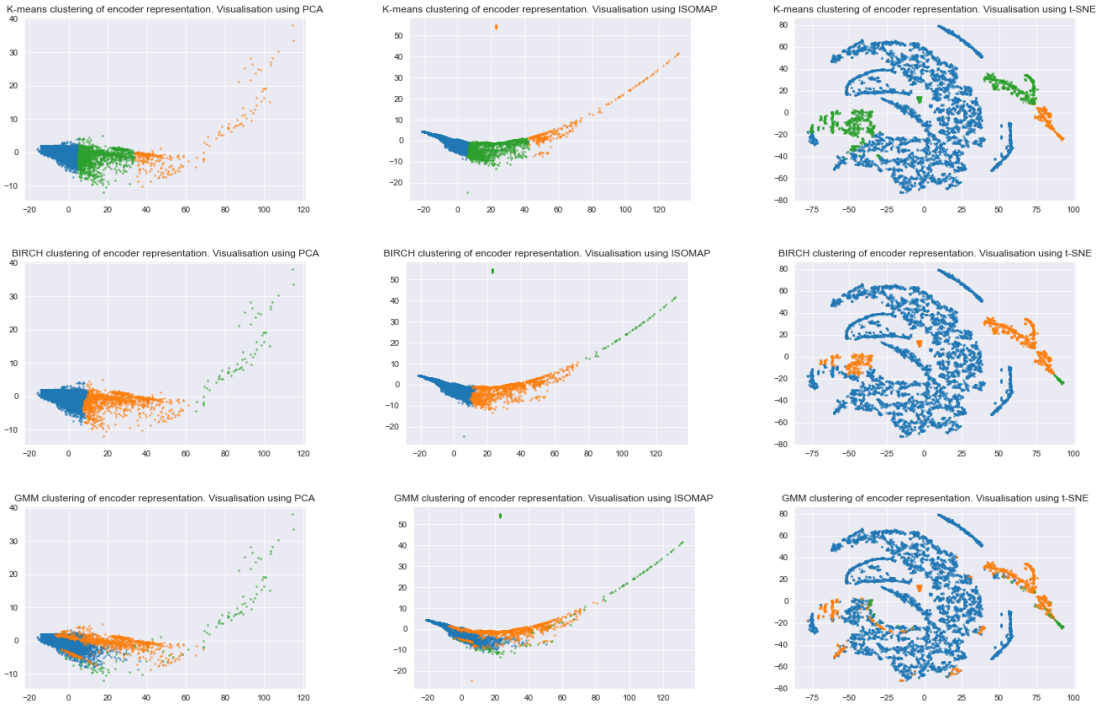

Figure 5.13: Plots of identified clusters on encoder 8-dimensional representation of test data. The 8D representation was obtained from the 203x128x64x32x16x**8** encoder after training with the column normalised dataset of mean of every 5 features. On the top row are the clusters formed by K-means, in the middle row are the clusters from BIRCH and on the bottom row are the clusters from GMM.

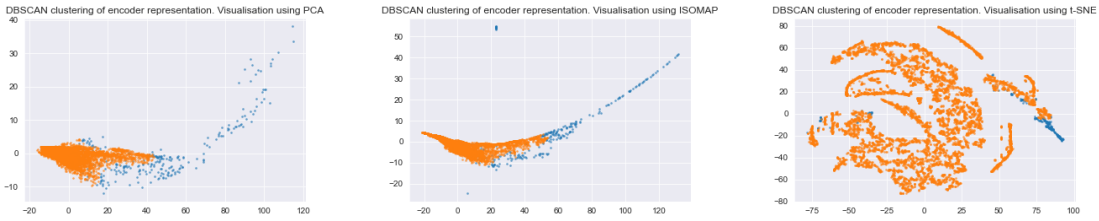

Figure 5.14: Plots identified clusters of encoder 8-dimensional representation of test data using DBSCAN with 16 as the minimal number of neighbours to form a cluster and 0.06 as ϵ. The DBSCAN algorithm was applied on the 8D representation from the encoder and the dimensionality reduction was applied after for visualisation purposes.

an area more densely populated with datapoints, and the other area, more scarcely populated with datapoints which are mostly deviated from the big agglomerate of datapoint, meaning that the algorithm could be identifying Outliers.

5.1.1 Denoising Autoencoder Results

After running the experiments with the mentioned encoders using all the different preprocessing techniques, in another attempt to obtain more robust representations, the models

which obtained the best results when trained with the full 1015 dimension dataset (results presented in Table A.2 and Table A.3) where trained using a noisy dataset with 10% noise as described in the section Noisy Dataset. The criteria for determining the best models and those which had potential for these experiment where a combination of the overall results of the reconstruction loss, classification accuracies for the classification of wine type and region and quality of the representation when visualised. Additionally, since the denoising is a regularisation technique, the chosen architectures and preprocessing used where combination that resulted models that overfitted the data. For this reason, the dataset that was mainly used for the denoising autoencoders was the original dataset containing the 1015 features. This is the dataset that contains the most details and information. With the regularisation introduced by the adding noise and denoising component of training, the hope is that the autoencoder can take advantage of that extra detail in the 1015 features without overfitting, creating more robust representations that contain new features that could potentially reveal some relations among wines.

The models and their results are in Table 5.8.

Table 5.8: Results of denoising autoencoder training and encoder prediction for all architectures after training of 500 epochs with full dataset with 10% noise. Training and validation loss are of the final epoch of autoencoder training. KNN and RF classification accuracy for the predictions of wine region given the representation of the spectra obtained from the trained encoder.

Training with Full Dataset with 10% noise

	Normalised by Row					Normalised by Column				
	Val Loss	**KNN acc Regions**	**RF acc Regions**	**KNN acc Type**	**RF acc Type**	**Val Loss**	**KNN acc Regions**	**RF acc Regions**	**KNN acc Type**	**RF acc Type**
1015x128x32x8	0.00068	34.5%	33.1%	55.3%	66.6%	0.00290	35.4%	40.4%	51.8%	66.7%
1015x64x32x8	0.00088	45.3%	44.4%	58.1%	65.9%	0.00359	39.5%	34.7%	58.4%	65.9%
1015x64x32x16x8	0.00101	41.0%	40.6%	66.5%	66.7%	0.00441	33.8%	34.2%	53.8%	66.7%
1015x128x64x32x16x8	0.00093	43.9%	29.5%	63.8%	66.5%	0.00349	34.8%	41.6%	51.9%	66.7%

Looking at the results in Table 5.8 and comparing to the results of the other models, the most apparent difference is in the validation loss, which is lower in the denoising autoencoders. This means that the encoders, after training with this technique, created representations that could be reconstructed to a state closer to the original, with less loss of information than the standard autoencoders. This means that the denoising encoder representations are potentially more informative. However, when looking at the classification accuracies, there isn't a big improvement expected from the decreased loss. Unlike the models presented in the section before, the classification results of denoising models trained with column normalised data are worse than the row normalised dataset denoising models. On the other hand, when trained with row normalised data, the denoising autoencoders obtain better accuracies than the models in Table 5.3.

Out of all the denoising autoencoders, the architecture which appears to obtain the best overall results in reconstruction loss, classification accuracy and visualisation is the 1015x64x32x8 encoder, trained with a symmetrical decoder. The confusion matrices of the classification of the representation obtained from this encoder when trained with the

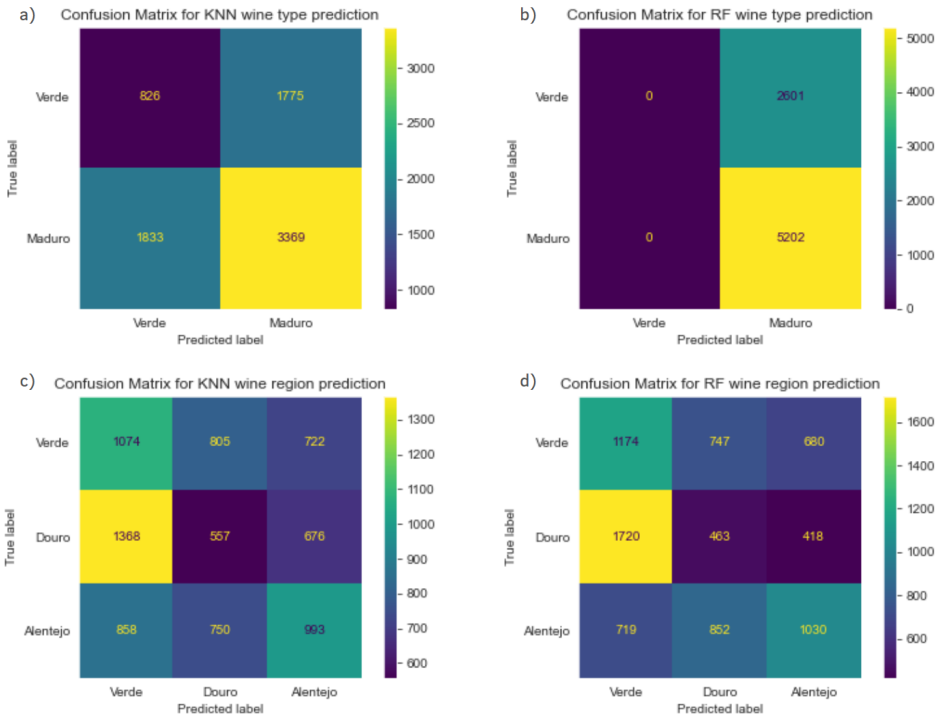

Figure 5.15: Confusion Matrices for type and region classification of representations obtained from the encoding by the trained 1015x64x32x**8** denoising encoder. The encoder was trained with the column normalised dataset with 10% noise and the confusion matrices presented are of: a) KNN wine type classification; b) RF wine type classification; c) KNN wine region classification and d) RF wine region classification.

column normalised dataset and of the representation obtained from this encoder when trained with the row normalised dataset are in Figure 5.15 and Figure 5.16, respectively.

As for the wine type classification of the representation from column normalised dataset, the confusion matrices show that the classifiers could not distinguish between classes. Both classification algorithms show a tendency for *Maduro* wine predictions, with RF algorithm resulting in 100% *Maduro* predictions, however, this can be explained by the unbalance in class distribution in the data. The confusion matrix of the KNN classification of representations of row normalised dataset shows a tendency for Verde wine prediction. This is not consistent with the confusion matrix of the RF classification algorithm. Knowing that the dataset is unbalanced with 67% of spectra being Maduro wines, this emphasis that the encoder did find some information in the spectra that lead to this result and contradicted the imbalance. It can mean that the denoising autoencoder found a representation that highlights some similarity between all wines that are more common in *Verde* wines, bringing these points closer and leading to mislabelling by the KNN classifier. As for the region classification, neither algorithm was able to analyse and correctly classify representations. The column normalised dataset representations' region classification resulted in similar results from both algorithm. There is a tendency for Verde predictions and a big misclassification of the Douro wines mainly for Verde

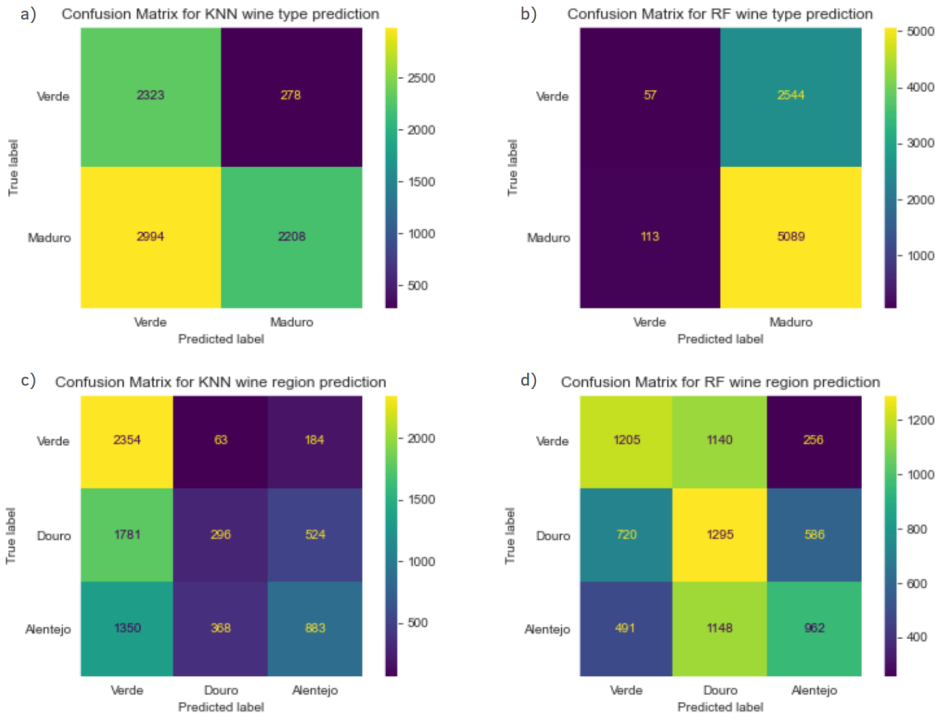

Figure 5.16: Confusion Matrices for type and region classification of representations obtained from the encoding by the trained 1015x64x32x**8** denoising encoder. The encoder was trained with the row normalised dataset with 10% noise and the confusion matrices presented are of: a) KNN wine type classification; b) RF wine type classification; c) KNN wine region classification and d) RF wine region classification.

wines. In the case of the row normalised representations' region classification, the KNN classifier also has a tendency for Verde predictions. In contrast, the RF shows any tendency for Douro wines, unlike any other encoder. It can mean that this encoder found a representation that allows the decision trees of the RF algorithm correctly differentiate, to a certain extent, wines from different regions.

Both denoising encoders from Figure 5.15 and Figure 5.16 were connected to classifier networks and trained to predict the regions of the wines the spectra belong to. The classifier using the encoder trained with the dataset normalised by column was able to predict the regions of the test set with 53% accuracy after training for 70 epochs. The classifier using the encoder trained with the dataset normalised by row was able to predict the regions of the test set with 46% accuracy after training for 97 epochs. Considering that the region classification of the encoder's representations using the KNN and RF algorithms only obtained accuracies of 39.5.7%, 34.7% and 45.3%, 44.4%, respectively, the accuracies obtained from these models are an improvement from the machine learning algorithms. The confusion Matrices for these classifications are presented in Figure 5.17.

Unlike the classification using machine learning algorithms, the DNN classifier being fed by the encoders achieved better results when using the dataset normalised by column. Looking at the confusion matrices, the one obtained from this denoising classifier

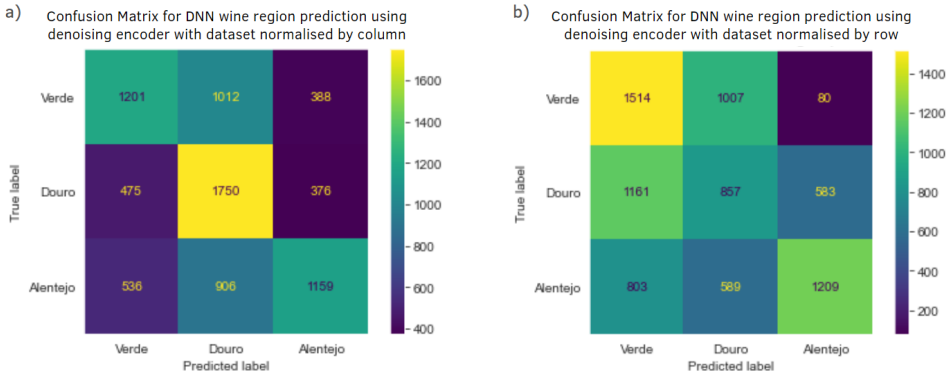

Figure 5.17: Confusion Matrices for region classification obtained by the DNN being fed by the trained 1015x64x32x8 denoising encoders. The encoders and encoder+classifying network combination models were trained with: (a) the column normalised dataset of means of every 5 features and (b) the row normalised dataset of means of every 5 features, both with 10% noise.

network using the dataset normalised by column (Figure 5.17.a)) is very similar to the normal encoder+DNN classifier network when using the same data, shown in Figure 5.5.a), indicating that maybe both networks found similar relations in the features.

The Figure 5.18 and Figure 5.19 show the 2-dimensional plots of the representation obtained by the denoising 1015x64x32x**8** encoder using column normalised data, labelled by wine the datapoint represents (Figure 5.18), and labelled by the clusters found by the clustering algorithms when applied to the 8-dimensional representation from the denoising encoder (Figure 5.19). Comparing these plots to the representations from the normal encoders using column normalised data (Figure 5.18 and Figure 5.19), the representation from the denoising encoder appears to have a similar distribution of the datapoints but more compacted. From the PCA and ISOMAP representations, it does not appear to exist agglomerates of datapoints with any type of relations or patterns, however from this plane it is not clear. In the t-SNE representation, it is visible that some datapoints appear to form separate groups, however that can be from the dimensionality reduction algorithm. Only from the plots in Figure 5.18 and the known labels, it is not possible to understand if relations are in fact being demonstrated by these representations. The same applies for the clusters found. From the visualisations, there are no clear and separate clusters, and for the ones identified by the clustering algorithms, we cannot deduce a relation since the clusters seem to consist of samples from multiple wines of both types and from all regions, following no pattern. Even in the case of DBSCAN, which is not bound to find a certain number of clusters, in the case of the denoising encoder representation of column normalised data, the algorithm appears to determine most datapoints as similar, and separate a few datapoints that can be seen as outliers. A similar cluster of outliers is also identified as a separate cluster by algorithms like BIRCH and GMM. Since these points are from multiple wines that have the majority of its datapoints in the main cluster, it is safe to say that these points are, in fact, outliers, meaning that

Table 5.9: Sensitivity Analysis of 1015x64x32x8 denoising encoder trained with column normalised dataset with 10% noise. Results obtained from the average of the sensibilities of all spectra in the test set.

Ranking	Sensibility	Wavenumber (cm^{-1})	Ranking	Sensibility	Wavenumber (cm^{-1})
1	0.2325	85	**1006**	0.0281	956
2	0.2234	91	**1007**	0.0279	389
3	0.2234	89	**1008**	0.0276	937
4	0.2063	112	**1009**	0.0273	745
5	0.2062	83	**1010**	0.0270	1194
6	0.2048	73	**1011**	0.0269	820
7	0.2015	75	**1012**	0.0264	406
8	0.1979	108	**1013**	0.0262	1590
9	0.1940	61	**1014**	0.0254	1129
10	0.1896	87	**1015**	0.0253	804

are spectra with values that deviate a lot from the normal range of values.

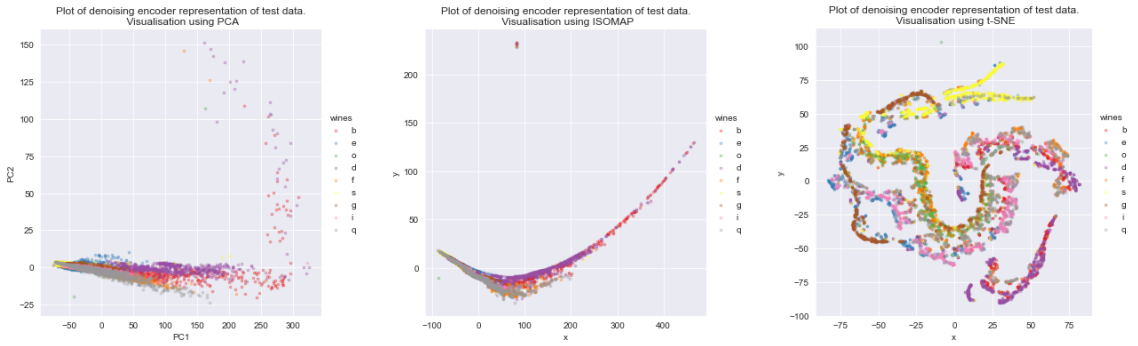

Figure 5.18: Plot of denoising encoder 8-dimensional representation of test data after dimensionality reduction by PCA, ISOMAP and t-SNE, respectively from left to right. The 8D representation was obtained from the 1015x64x32x**8** encoder after training with the column normalised dataset with 10% noise. Points are coloured by wine that sample belongs to.

The sensibility analysis of this denoising encoder was performed. The method used was the same as the other models. The sensibilities of the outputs with respect to the input features were calculated using Partial Derivatives algorithm for all the spectra in the test set and the results were grouped per spectrum to get the general sensibility of the representation calculated to the input features. The difference in these denoising encoders is that they use all the features of the dataset (1015), so the sensibility is being calculated regarding the original features and not the mean of the original features like with the normal encoders.

The results for the 10 most important features and the 10 least important features of the denoising encoder trained with the column normalised dataset with 10% noise are in Table 5.9. None of the top 10 features from this model is also present in top 10 features of the normal encoder trained with the column normalised dataset in Table 5.6.

When it comes to the classifier DNN fed by this encoder, the sensibility analysis demonstrated, once again, that all outputs follow the same pattern of sensibility to the different features, with the same features being roughly in the same rank of influence in

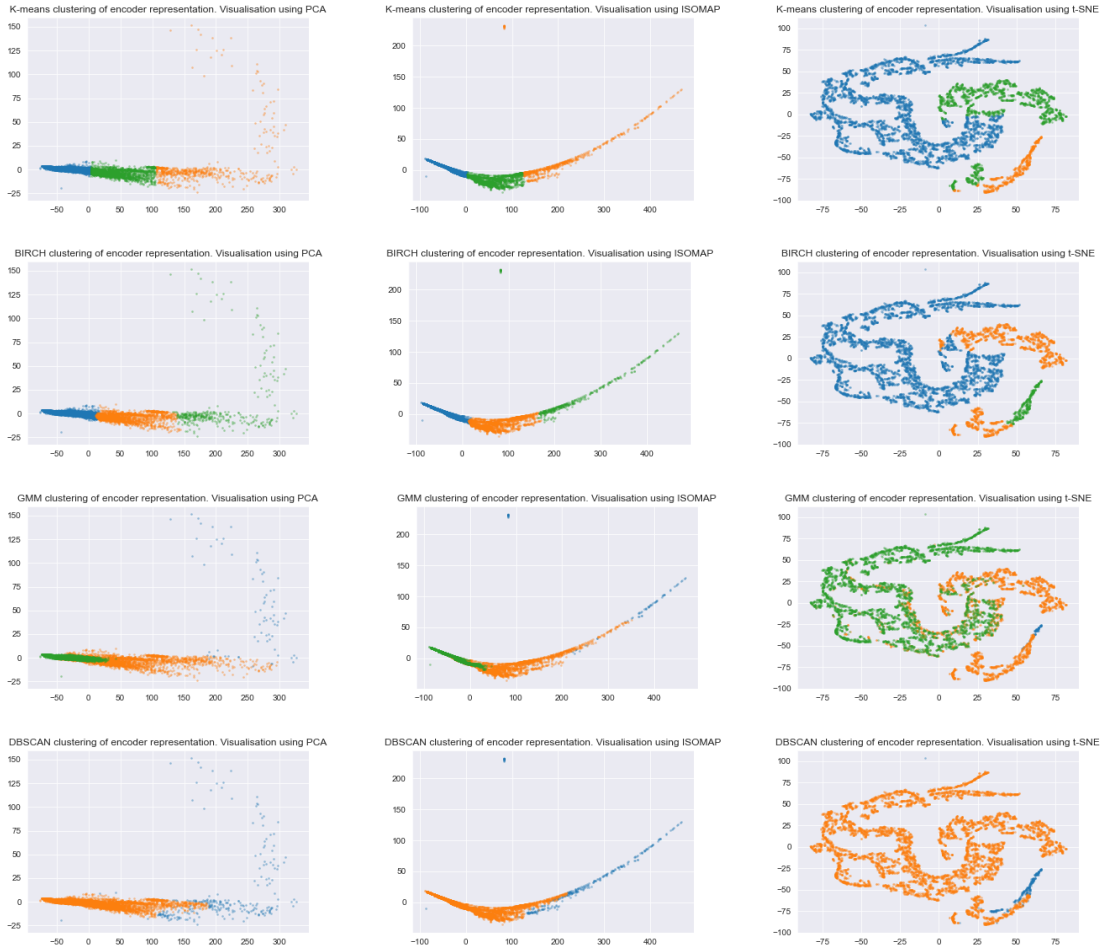

Figure 5.19: Plots of identified clusters on denoising encoder 8-dimensional representation of test data. The 8D representation was obtained from the 1015x64x32x**8** encoder after training with the column normalised dataset with 10% noise.From top to bottom rows are the clusters formed by K-means, BIRCH, GMM and DBSCAN.

Table 5.10: Top Sensitivity values for outputs of classifier DNN fed by denoising encoder trained and using column normalised data.

Ranking	**Sensibility Verde Pred.**	**Sensibility Alentejo Pred.**	**Sensibility Douro Pred.**	**Wavenumber (cm^{-1})**
1	0.0179	0.0124	0.0056	1787
2	0.0171	0.0116	0.0055	1825
3	0.0166	0.0113	0.0053	1811
4	0.0156	0.0107	0.0049	1814
5	0.0156	0.0107	0.0048	1809

the correspondent output. Additionally, the features with big influences on the outputs show sensitivity values more elevated in the Verde output relatively to the sensitivities in the Alentejo output which are also higher than the sensibilities in the Douro output. Meaning that these features have a bigger influence in the Verde prediction that they have in the Alentejo or Douro predictions.

The most important features in this model are the features in Table 5.10. Out of these

top 5 features, none are present in the top of the important features of the encoder in Table 5.6. Additionally, it appears that the classifier was more influenced by the side of the spectra with bigger wavenumbers, whilst the encoder was more influenced by smaller wavenumbers on the opposite side of the spectra. Moreover, not all of the most influential features on the encoders representations are considered as relevant in the predictions made by the classifier. This indicates that, possibly, the network in the classification training adapted to give more attention to the features in Table 5.10 in order to improve the predictions. Considering that the encoder+DNN combined model improved the accuracy of region predictions by 13%, this indicates that its most influential features are more informative on the wines region that the features considered as important for the encoder representation. Considering that the predictions by this classifier were similar to the predictions by the DNN classifier without the noisy data in Figure 5.5.a, it was expected to find some similarities in the features considered as most influential in the classifications. Comparing the Tables 5.10 and 5.7 we see there are no shared wavenumbers, but despite not sharing the top 5 wavenumbers, the top 5 wavenumbers in one network are very influential in the other network and are ranked high. This shows some consistency in the most important features of classifiers that were able to correctly classify 52% and 53% of the data, demonstrating that those wavenumbers may represent something that allows for better discrimination of regions.

The Figure 5.20 shows the 2-dimensional plots of the 8-dimensional representations obtained from the denoising encoder trained with row normalised data. These plots, when labelled by wine, demonstrate that once again the encoder was able to produce representations where datapoints representing the same wine are close to one another. In the representations obtained from PCA and ISOMAP, many datapoints overlap, with no clear clusters, however there are some areas where there is an increased density of points from the same wine. In the representation from t-SNE, those agglomerations appear more separated from the rest of the datapoints. These representations obtained from the denoising encoder are similar to the representations obtained with the normal encoder, plotted in Figure 5.6 in the sense that a relation between samples of the same wines can be observed, however, that relation is more clear in the latter case, meaning that possibly the denoising encoder found a representation where other features and relations are brought out and, as it was mentioned previously in the former section, the normal encoder was able to find a representation that highlighted the relation between wines.

The Figure 5.21 shows the plots of the clusters found by the different clustering algorithms when applied to the 8-dimensional representation of the denoising encoder after training with the row normalised dataset. Once again, all algorithms were configured to find 3 clusters, except for the DBSCAN, however, this last algorithm also found 3 different clusters in the data. The found clusters do not appear to represent any relations between wines, at least relations that can be identified from the knowledge we have on the data. The datapoints in the clusters do not appear to be from wines of the same region or type and do not appear to be consistent with the wines the datapoints represent, nor with the

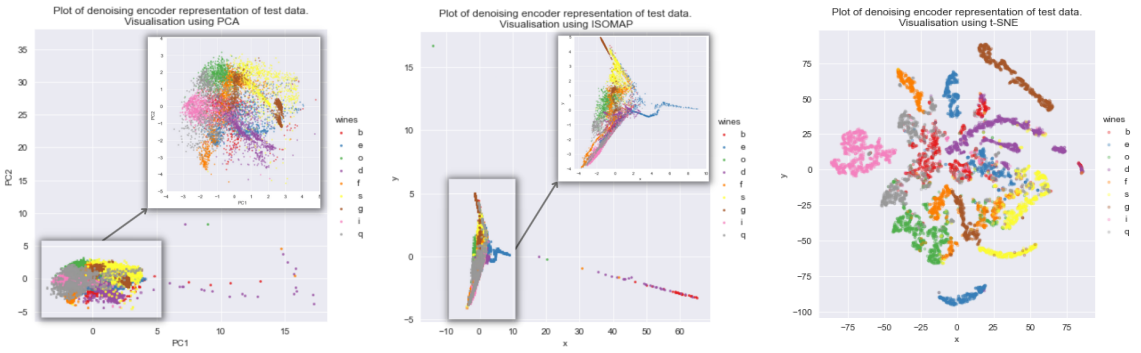

Figure 5.20: Plot of denoising encoder 8-dimensional representation of test data after dimensionality reduction by PCA, ISOMAP and t-SNE, respectively from left to right. The 8D representation was obtained from the 1015x64x32x**8** encoder after training with the row normalised dataset with 10% noise. Points are coloured by wine that sample belongs to.

Table 5.11: Sensitivity Analysis of 1015x64x32x8 denoising encoder trained with row normalised dataset with 10% noise. Results obtained from the average of the sensibilities of all spectra in the test set.

Ranking	**Sensibility**	**Wavenumber** (cm^{-1})	**Ranking**	**Sensibility**	**Wavenumber** (cm^{-1})
1	0.1993	61	**1006**	0.0281	1476
2	0.1643	63	**1007**	0.0272	694
3	0.1093	65	**1008**	0.0271	921
4	0.0885	223	**1009**	0.0264	628
5	0.0855	67	**1010**	0.0262	1419
6	0.0836	252	**1011**	0.0254	1496
7	0.0822	246	**1012**	0.0253	1194
8	0.0818	372	**1013**	0.0248	834
9	0.0788	227	**1014**	0.0245	1243
10	0.0782	221	**1015**	0.0228	1456

grape variety used in the wines

Since this encoder was also able to generate representations that show a relation between samples of the same wines, the sensibility analysis of this encoder should provide some information on what features/wavenumbers the model mostly bases its representations from, and which it ignores. Those results are in Table 5.11.

In this encoder, none of the important wavenumbers coincide with the ones of the normal encoders in either Table 5.4 however since the datasets are build different, the fact that the representations are similar but the important wavelengths are different can be explained by that. Additionally,, even thought the encoders do not share the same most influential features, the majority of them are highly ranked in the other encoder, meaning that they are quite influential in the resulting representations. The biggest difference in this is that the features ranked as 1, 2, 3 and 5 in this encoder all represent the same feature in the dataset used in the regular encoder, and this feature in the regular encoder is ranked at 139 out of 203, which means the representations are not very influenced by this feature. Since the representations have some differences, where the representations from the denoising encoders show less definition in the groups formed by spectra of the same wines, it could possibly mean that those wavenumbers represent some extra information

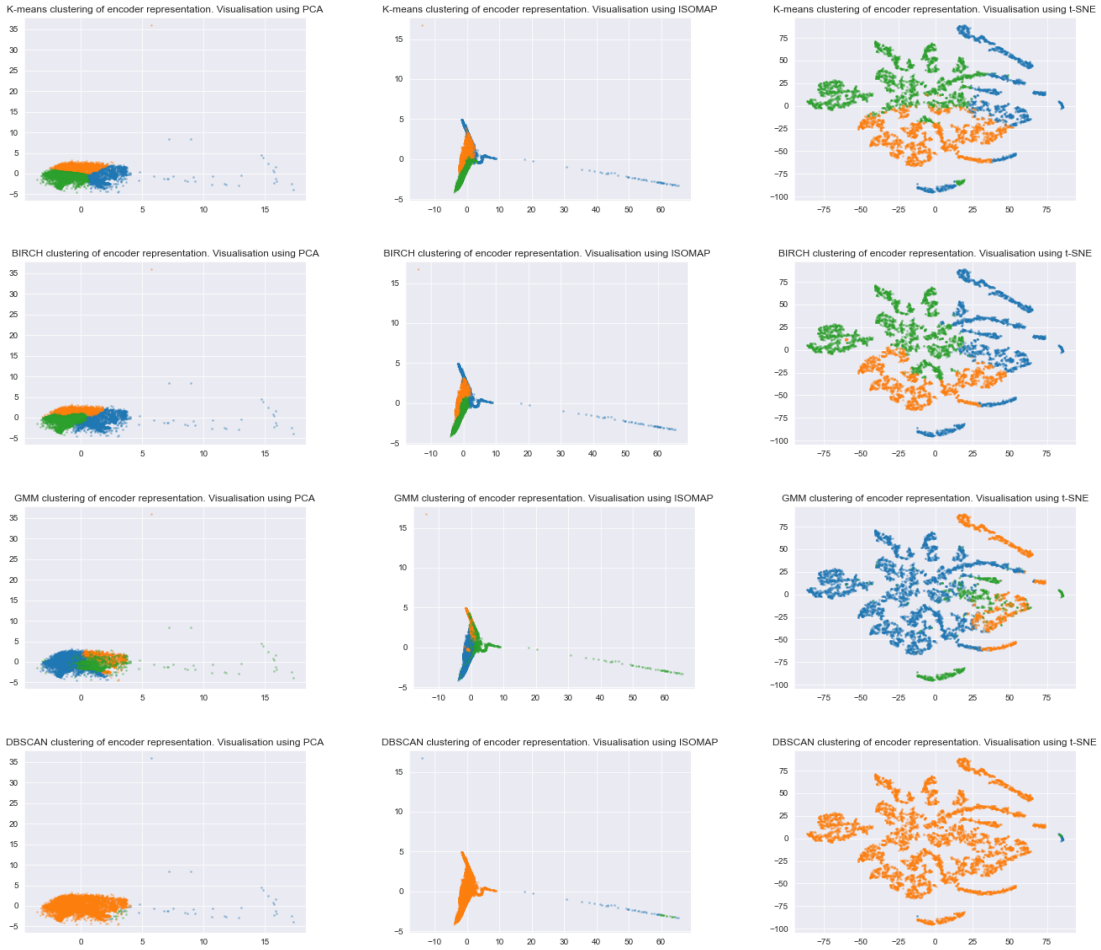

Figure 5.21: Plots of identified clusters on denoising encoder 8-dimensional representation of test data. The 8D representation was obtained from the 1015x64x32x**8** encoder after training with the row normalised dataset 10% noise. From top to bottom rows are the clusters formed by K-means, BIRCH, GMM and DBSCAN.

that is not directly related to each individual wine and deviates the datapoints from those of the same wine.

Table 5.12: Top Sensitivity values for outputs of classifier DNN fed by denoising encoder trained and using row normalised data.

Ranking	**Sensibility Verde Pred.**	**Sensibility Alentejo Pred.**	**Sensibility Douro Pred.**	**Wavenumber** (cm^{-1})
1	0.0162	0.0096	0.0066	61
2	0.0117	0.0073	0.0044	63
3	0.0098	0.0063	0.0035	880
4	0.0088	0.0058	0.0031	902
5	0.0085	0.0058	0.0027	714

When this encoder was connected to a classifier network and was trained to predict regions based on the wine's spectra, the accuracy of the classification of the dataset was 45%, which is the same accuracy as the classification of the representations of the encoder

by the machine learning algorithms. The sensitivity analysis of this classifier model demonstrated that, similar to all other models, all the outputs are influenced by the same features in a similar patters, but with the most important features being more influential in the Verde prediction, than in the Alentejo prediction and, lastly, in the Douro prediction. Looking at the features and the sensitivity measure the outputs have on changes in those features, we can see in Table 5.12 that the top 2 most influential features in all outputs are the same as the top 2 most influential features in the encoder in Table 5.11. On the other hand, the remaining features considered most important by the encoder are, in terms of rank, mostly in the lower half of influential features of the classifier results. This change however did not improve the results, indicating that maybe none of these features are particularly informative about the wine's region.

6

Discussion

The goal of this study is to use Machine and Deep Learning techniques for the processing and categorisation of the complex data obtained from the SERS spectra of Portuguese white wines. The results of the most relevant experiments performed to achieve this goal were presented in the previous chapter. In this chapter well go over the results and discuss some of the drawn conclusions.

6.1 Autoencoder Architecture, Finetuning and Training

In architectures with many layers, it was expected to encounter many problems in training related to model overfitting since more layers mean there are more neurons capable of learning. The models struggled from unstable training with big variations in the loss where in some cases it appeared that the model struggled to learn, hence the need to use a smaller learning rate. These models also struggled from very early on in training with a bit of overfitting. This was not as noticeable in the models using the datasets of the means of every 3 or five features of the original dataset, that only suffered with some overfitting that was resolved by only applying less training, as demonstrated in the loss plot in Figure 5.2. The overfitting can be explained by the type of data being used. When the models use a dataset composed of a mean of every five features of the original dataset, it means that the dataset being used is probably less complex and has less information that the model can use to learn and specify its knowledge, so, in a way, using these datasets can be seen as applying some regularisation to the training, and from analysing the representations obtained and results from the models trained with these different datasets, we can conclude that that transformation on the dataset did not affect the results since they are better when not using the full dataset.

When using the smaller learning rates in training, in the case of the models trained with normalisation by column, the autoencoder training converges on average at around epochs 100 to 300, depending on the complexity of the dataset and architecture of the model, and the loss values stabilise without getting better or worse for the remaining epochs. This indicates that the autoencoders reach a state where they cannot learn more

information from the given data. The fact that the dataset with normalisation by column led to models converging (shown in Figure 5.1 on the right), while the dataset with normalisation by row led to overfitting in most cases (shown in Figure 5.2) suggests that the type of preprocessing affects the performance of be model. The normalisation by column, or normalisation by feature, allows for better convergence since all features are rescaled to be within the same ranges, meaning that the values do not have huge variations that lead to big oscillations in the gradients from example to example that is fed into the autoencoder. In contrast, when normalising by row, or by datapoint, the features are not necessarily following the same scale, and their context can actually be distorted, which can confuse the training as the models may focus on features of the training data that are not actually relevant in the universe of the dataset.

As demonstrated by the results of the validation loss and accuracies in Tables 5.3, 5.8 and all remaining Tables in A, the depth and width of the autoencoders influenced the results, as expected, for all cases. The models which performed better in terms of reconstruction loss are the ones using 8 neurons as middle layer, followed by models using 4 neurons as middle layer. The more neurons are present in the middle layer, the more dimensions are in the lower dimension representation and, therefore, the information does not have to be so compressed, leading to less information being lost in this transformation. However, that did not translate into improved classification accuracies of the representations from those encoders.

6.2 Classification of Spectra Representations

Given the poor classification results and the fact that the different machine learning algorithms resulted in such different predictions, it shows that the representations from the regular encoders did not have the necessary information for the classification algorithms to make accurate predictions. The fact that the accuracies are slightly better when using the dataset normalised by column can also be explained by type of preprocessing and the implications in training mentioned in the previous paragraph.

The training with the noisy data did not improve the results as they also failed to find representations that improve the distinction between wines from different types and regions. From the loss values, we know that the found representations allow for a better reconstruction with less loss of information, however, even though the representations are more informative, the classification is still not successful. This indicates the possibility that the spectra simply do not have information necessary for the classification of wine regions and types.

When the knowledge obtained by the autoencoders was applied to classify the dataset using transfer learning, the models ability to predict the region of the wine based on the spectra improved for 3 out of the 4 encoder+DNN combinations presented. This improvement indicated that even with the simplified representation obtained by the encoders, the machine learning algorithms had a lower performance compared to the

DNN classifier, probably because the representation is still way to complex for simple machine learning algorithms to obtain good results. On the other hand, although this type of learning did get better results, these are still not satisfactory, since in the best case there was still a 47% misclassification rate. This high error rate reinforces the question of if the spectra do have the necessary information for the discrimination of samples per region of the wine they belong to.

6.3 Visualisation of Spectra Representations

Looking at the plots generated by the representations obtained from all the encoders, regardless of using the whole dataset or a dataset composed of the mean of every 5 features of the original data to train the autoencoders into finding lower dimensional representations, the normalisation by row allowed the encoders to compress the data into representations that resulted in reasonable visualisations where relations among wines could be analysed, yet, from the visualisation of representations obtained when using the datasets normalised by column and from the known information on the dataset, it is not possible to identify relations or patterns among the datapoints. The normalisation by row, meaning that each example in the dataset is normalised independently, allows for a better preservation of the relations within the data, whereas normalising by columns can actually distort those relations when normalising the features (columns), independently, along all datapoints. This can explain the big differences seen in the plots in Figure 5.6 with representation from encoders with row normalised data and Figure 5.10 with representation from encoders with column normalised data. On the other hand, what allows row normalisation to preserve the relations within the data can also have a negative effect on the data. This method makes it that the datapoints may no longer be directly comparable with one another because they are treated independently, so they may preserve their proximity or distance to other points but impact the way the individual features in each point relate to the features in another point. The consequence of all this is that it might be more difficult to find new relations based on some features of the data and not on the whole data and, as the datapoint cannot be compared in the same way, that can affect the learning process of an autoencoder and the classification process.

A common factor in the representations of the test set obtained by the encoders and from the representations of the full dataset obtained with dimensionality reduction algorithms, there is always a tail of datapoints that spread out from the big agglomerate of points where the majority samples are represented. This tail is mostly visible in the representations obtained using PCA and ISOMAP and it consists mainly of some samples from wines b and d, plus one or two samples from wine f. This consistency indicates that there is something different between the samples in the tail and the samples in the big cluster of datapoints, and since this tail only has a small subset of the samples from the mentioned wines, this supports the hypothesis that these particular samples are not really a good representative of the wine and can be considered outliers, as the expected

was that the samples from the same wine are consistent (based on the assumption that the components of wines are evenly distributed in the samples). Examples like this also exist in the training set and this might have had implications in training. The presence of outliers in the training set could have affected the representations. Depending on the amount of outliers, the autoencoders might have struggled to learn from the inconsistencies in the features of good datapoints and outliers, or even fit to the features of the outliers and not generalised for the type of data being handled.

All of the plots presented were of representations of the test set, which is a subset of 30% of the spectra in the dataset. After obtaining the representations from the encoders, these were projected onto a 2-dimensional space using PCA, ISOMAP and t-SNE. Each of these algorithms transforms the data in different ways leading to different visualisations of the same set of datapoints. In all plots of all representations from the encoders, PCA and ISOMAP transform the data in a way that there was a lot of overlapping, even when visualising only the test subset, however the t-SNE algorithm is able to project the data in a way that there was less overlapping of the test subset. Especially when using the datasets normalised by row, as seen for example in Figure 5.6, the overlapping is mostly of samples belonging to the same wine, indicating a high degree of similarity between those points. When this algorithm is applied to the encoder's representations of the whole dataset (training set + test set) there is no longer a clear separation of datapoints, and there is more overlapping, from either samples from the same wines and samples from different wines.

6.4 Clustering of Spectra Representations

Given that the representations obtained from the encoders were not clear in the separation of spectra by their wine regions or types, it was expected that the clustering algorithms would not have a good performance in the identification of these groups. Additionally, by analysing the plots, it was visible that there was no clear agglomerations of datapoints based on those factors. That is to say that from the clustering algorithms restricted to finding 2 and 3 different clusters, it was expected that these clusters were not directly related to this differentiation of spectra by their region and type. From what is known about the dataset, which is the different wines the spectra belong to, the wine type, wine region, grape varieties and alcohol level, no identifiable relation or distinction was found in the clusters besides the fact that multiple algorithms identify the mentioned "tail"as its own cluster, reinforcing the hypothesis of those points being outliers. The DBSCAN algorithm was the only clustering technique utilised that was not bound to find a specific number of clusters. The results from this cluster consisted mainly of 2 or 3 clusters, but where one cluster consisted of the majority of the datapoints and the other one or two clusters were, once again, the datapoints from the "tail".

6.5 Sensitivity Analysis of the Deep Neural Networks

As mentioned, the knowledge on the dataset is limited. The features being used in this study are the intensities of the different waves captured after the the laser light is refracted in the wine and silver nanoparticles solution. It is know that the intensities of the different waves represent different components present in the wine and they can even represent the concentration of these components in the wine samples, however, at the moment of this study, we do not know what components are being identified in the wines. When calculating the sensitivity of the encoders' representations and classifier predictions to the different input features, what we can obtain from that analysis is what wavenumbers the deep networks consider most and less influential for their outputs.

From the sensitivity analysis of the encoders, we know the features that are more and less important in the representations. Since the representations from the encoders trained with datasets normalised by column did not have very good classification results nor their representations demonstrated identifiable relations when plotted and visualised, it is difficult to infer their meaning in the context of the dataset. On the other hand, the encoders trained with datasets normalised by row did produce representations which related to each other by the wine they belong to, so from the sensitivity analysis we know that the features which had the biggest influence on those representations are the wavenumbers represented in Table 5.4.

The same logic applies to the sensitivity analysis of the denoising encoders. The denoising encoder trained with the dataset normalised by row generated representations that demonstrated relations based on the wine the samples belong to, so we know that the features which had the biggest influence on those representations are the wavenumbers represented in Table 5.12. Given that both this denoising encoder and the normal encoder obtained similar representations, the fact that the most important features in each encoder are different can be because in the case of the denoising encoder the dataset used was the full dataset of 1015 features, whilst in the normal encoder was used the dataset composed of the mean of each consecutive 5 features of the original dataset. This difference means that the features end up having a different significance, which could lead each encoder having to focus on different features to reach the results. Additionally, what determines the results is the entirety of the features having their different input in the output, whether big or small, the most influential features only influence the outputs to a certain extent.

When combining the encoders to classifier networks, the accuracies improved. The sensitivity analysis on these models could reveal which features had a greater influence on the prediction of each class. All the encoders were trained and the neurons were frozen so that their hyperparameters remained the same when being used for the classification task, meaning that only the added layers trained to classify the spectra by region. By comparing the features that most influence the encoders' representations, and the features that most influence the classifiers predictions it is possible to analyse how the network differs from the encoder. In most cases, the majority of the most important features of the classifier

were different from the top features of the encoder, in some cases, these top features of the encoder were not even very influential in the classification result. This can indicate that in the generation of representations, the encoders focused on features that are not important in the differentiation of classes, and this can explain why the classification results are so different. Another hypothesis is that the encoder's representations are still way to complex for machine learning techniques to analyse, however, a more powerful tool like a neural network is more capable of extracting meaningful information from the data that allows for a better distinction between regions.

7

Conclusion and Future Work

The goal of this study is to use Machine and Deep Learning techniques for the processing and categorisation of the complex data obtained from the SERS spectra of Portuguese white wines.

As established from the beginning, the complexity of the data was too high for the spectra to be directly analysed with simple machine learning algorithms and the option of performing dimensionality reduction in an attempt to decrease the complexity of the dataset resulted in overall worse results due to possible information loss. From past studies, we know that these techniques allow for a limited study of the dataset, but, to fully analyse and understand the very intricate reality of wines and how they relate, we need a more powerful tool.

The chosen tool in this study was Deep Learning, more specifically deep autoencoders, that were used with different variations of the dataset. The expected was to be able to obtain less complex representations of the wine's spectra, where the relations between the different wines could be studied. To this end, we conducted a series of experiments with autoencoders, varying the type of the dataset, grouping the features based on the high correlation values to analyse if the simplified dataset benefited the results, varying the type of preprocessing of the dataset to analyse if the transformations influenced the results, and varying in width and depth of the autoencoder model to analyse which architecture was better suited for the data. To verify the quality of the representations found by the encoder part of the autoencoders, we used multiple classification, clustering and visualisation techniques, plus sensitivity analysis of the encoder's outputs with respect to each of the input features.

The results of the experiments demonstrated that there was improvement in the representations, however not as much as expected. When using machine learning algorithms to perform classification of the representations, the wine region classifications increased up to 4%, comparing with the same classification with the raw dataset or after dimensionality reduction. On the other hand, when using a deep network to classify the representations by their regions, the maximum accuracy obtained was a 12% improvement from the same task performed on the raw dataset or after dimensionality reduction. The application

of clustering algorithms did not provide any new insight into the data as no clusters with clear relations were identified in any of the cases. Some combinations of encoders and dataset preprocessing were able to generate representations that show the spectra scattered all over the latent plane, in a distribution that follows no specific order or organisation. In other cases, the obtained representations demonstrated a relation between each spectrum and the wine it belongs to. Despite similar representations may be obtained by reducing the dimensionality of the original dataset with traditional techniques, the proposed solution can be used to bring a new insight into this relation among wines and can be used to explore it further by understanding which wavenumbers have more influence in that separation of spectra of different wines by means of sensitivity analyses and this can be useful in understanding how the wines relate to each other.

The main struggle of this study was that knowledge on the dataset was very limited and this made it hard to interpret the results when they deviated from the known labels. The meaning of the features is also quite abstract, so we were able to analyse which wavelengths most influenced the results, but we were not able to interpret their meaning in the context of the different wines and therefore could not infer from those features what information could be in the encoders representations. Additionally, this lack of understanding limited our ability to perfect and finetune the models to highlight some other features that could have generated more interesting results.

Future work on this subject should start by gaining a better understanding of the dataset, in order to be fully capable of finding and analysing the possible relations between different wines from their spectra. The dataset is very complex and on the course of this study it has been demonstrated that traditional autoencoders are not the appropriate tool for feature extraction of these types of datasets. Variational Autoencoders are generative autoencoders have been shown effective at feature extraction of complex, highly dimensional datasets, and have the additional advantage of giving more control over how the model represents the latent distribution of data, which is something that can be very helpful in the study of this dataset.

Bibliography

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. "A learning algorithm for Boltzmann machines". In: *Cognitive science* 9.1 (1985), pp. 147–169 (cit. on p. 10).

[2] M. P. de Almeida et al. "Expedite SERS Fingerprinting of Portuguese White Wines Using Plasmonic Silver Nanostars". In: *Frontiers in Chemistry* 7 (2019). ISSN: 2296-2646. DOI: 10.3389/fchem.2019.00368. URL: https://www.frontiersin.org/article/10.3389/fchem.2019.00368 (cit. on pp. 2, 4, 18, 29).

[3] J. Brownlee. *10 Clustering Algorithms With Python*. Machine Learning Mastery, 2020. URL: https://machinelearningmastery.com/clustering-algorithms-with-python/ (visited on 02/12/2022) (cit. on p. 23).

[4] L. Cao et al. "A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine". In: *Neurocomputing* 55.1 (2003). Support Vector Machines, pp. 321–336. ISSN: 0925-2312. DOI: https://doi.org/10.1016/S0925-2312(03)00433-8. URL: https://www.sciencedirect.com/science/article/pii/S0925231203004338 (cit. on p. 20).

[5] M. Cao et al. "Advanced Methods in Neural Networks-Based Sensitivity Analysis with their Applications in Civil Engineering". In: *Artificial Neural Networks*. Ed. by J. L. G. Rosa. Rijeka: IntechOpen, 2016. Chap. 13. DOI: 10.5772/64026. URL: https://doi.org/10.5772/64026 (cit. on p. 15).

[6] S. Casale et al. "Speech Emotion Classification Using Machine Learning Algorithms". In: *2008 IEEE International Conference on Semantic Computing*. 2008, pp. 158–165. DOI: 10.1109/ICSC.2008.43 (cit. on p. 18).

[7] N. K. Chandra, A. Canale, and D. B. Dunson. "Escaping the curse of dimensionality in Bayesian model based clustering". In: *arXiv preprint arXiv:2006.02700* (2020) (cit. on p. 23).

[8] F. U. Ciloglu et al. "Drug-resistant Staphylococcus aureus bacteria detection by combining surface-enhanced Raman spectroscopy (SERS) and deep learning techniques". In: *Scientific Reports* 11 (2021). ISSN: 2045-2322. DOI: 10.1038/s41598-021-97882-4 (cit. on pp. 29, 30).

[9] *Clustering Algorithms | Clustering in Machine Learning*. Google Developers. URL: https://developers.google.com/machine-learning/clustering/clustering-algorithms (visited on 02/08/2022) (cit. on p. 7).

[10] P. Comon. "Independent component analysis, a new concept?" In: *Signal processing* 36.3 (1994), pp. 287–314 (cit. on p. 19).

[11] T. Cover and P. Hart. "Nearest neighbor pattern classification". In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27. DOI: 10.1109/TIT.1967.1053964 (cit. on p. 21).

[12] B. Du et al. "Stacked convolutional denoising auto-encoders for feature representation". In: *IEEE transactions on cybernetics* 47.4 (2016), pp. 1017–1027 (cit. on p. 28).

[13] M. Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *kdd*. Vol. 96. 34. 1996, pp. 226–231 (cit. on p. 22).

[14] X. Geng, D.-C. Zhan, and Z.-H. Zhou. "Supervised nonlinear dimensionality reduction for visualization and classification". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 35.6 (2005), pp. 1098–1107. DOI: 10.1109/TSMCB.2005.850151 (cit. on p. 18).

[15] X. Guo et al. "Deep clustering with convolutional autoencoders". In: *International conference on neural information processing*. Springer. 2017, pp. 373–382 (cit. on p. 27).

[16] T. K. Ho. "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1. 1995, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994 (cit. on p. 21).

[17] J. Huang et al. "On-site detection of sars-cov-2 antigen by deep learning-based surface-enhanced raman spectroscopy and its biochemical foundations". In: *Analytical Chemistry* 93.26 (2021), pp. 9174–9182 (cit. on p. 29).

[18] *K-Means Advantages and Disadvantages | Clustering in Machine Learning*. Google Developers. URL: https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages (visited on 02/11/2022) (cit. on p. 23).

[19] S. Khalid, T. Khalil, and S. Nasreen. "A survey of feature selection and feature extraction techniques in machine learning". In: *2014 science and information conference*. IEEE. 2014, pp. 372–378 (cit. on p. 2).

[20] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013) (cit. on p. 25).

[21] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas. "Data preprocessing for supervised leaning". In: *International journal of computer science* 1.2 (2006), pp. 111–117 (cit. on p. 2).

[22] L. Krippahl. "Aprendizagem Automática (Machine Learning) - Lecture Notes". In: (2020) (cit. on p. 6).

[23] L. Krippahl. "Aprendizagem Profunda - Lecture Notes". In: (2021) (cit. on pp. 6, 8, 10, 12, 24–26).

[24] S. Kullback and R. A. Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86 (cit. on p. 15).

[25] Y. LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 27).

[26] J. Lever, M. Krzywinski, and N. Altman. "Principal component analysis". In: *Nature Methods* 14.7 (2017). ISSN: 1548-7105. DOI: 10.1038/nmeth.4346 (cit. on p. 29).

[27] X. Li et al. "Noninvasive liver diseases detection based on serum surface enhanced Raman spectroscopy and statistical analysis". In: *Opt. Express* 23.14 (July 2015), pp. 18361–18372. DOI: 10.1364/OE.23.018361. URL: http://www.osapublishing.org/oe/abstract.cfm?URI=oe-23-14-18361 (cit. on p. 18).

[28] X. Lin et al. "High Throughput Blood Analysis Based on Deep Learning Algorithm and Self-Positioning Super-Hydrophobic SERS Platform for Non-Invasive Multi-Disease Screening". In: *Advanced Functional Materials* 31.51 (2021), p. 2103382 (cit. on p. 29).

[29] D. Lunga et al. "Manifold-Learning-Based Feature Extraction for Classification of Hyperspectral Data: A Review of Advances in Manifold Learning". In: *IEEE Signal Processing Magazine* 31.1 (2014), pp. 55–66. DOI: 10.1109/MSP.2013.2279894 (cit. on p. 19).

[30] F. Lussier et al. "Deep learning and artificial intelligence methods for Raman and surface-enhanced Raman scattering". In: *TrAC Trends in Analytical Chemistry* 124 (2020), p. 115796. ISSN: 0165-9936. DOI: https://doi.org/10.1016/j.trac.2019.115796. URL: https://www.sciencedirect.com/science/article/pii/S0165993619305783 (cit. on p. 18).

[31] J. MacQueen. "Classification and analysis of multivariate observations". In: *5th Berkeley Symp. Math. Statist. Probability*. 1967, pp. 281–297 (cit. on p. 21).

[32] M. S. Mahmud and X. Fu. "Unsupervised classification of high-dimension and low-sample data with variational autoencoder based dimensionality reduction". In: *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*. 2019, pp. 498–503. DOI: 10.1109/ICARM.2019.8834333 (cit. on pp. 25, 26).

[33] K. Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11 (1901), pp. 559–572 (cit. on p. 18).

[34] *scikit-learn: machine learning in Python — scikit-learn 1.0.2 documentation*. URL: https://scikit-learn.org/stable/ (visited on 02/17/2022) (cit. on p. 39).

[35] J. Shi and Z. Luo. "Nonlinear dimensionality reduction of gene expression data for visualization and clustering analysis of cancer tissue samples". In: *Computers in biology and medicine* 40 (2010), pp. 723–732 (cit. on p. 23).

[36] A. Talwalkar, S. Kumar, and H. Rowley. "Large-scale manifold learning". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587670 (cit. on p. 6).

[37] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. "A global geometric framework for nonlinear dimensionality reduction". In: *science* 290.5500 (2000), pp. 2319–2323 (cit. on p. 20).

[38] M. Usmani et al. "Stock market prediction using machine learning techniques". In: *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*. 2016, pp. 322–327. DOI: 10.1109/ICCOINS.2016.7783235 (cit. on p. 18).

[39] L. Van der Maaten and G. Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008) (cit. on p. 20).

[40] S. Wang et al. "Rapid SERS identification of methicillin-susceptible and methicillin-resistant Staphylococcus aureus via aptamer recognition and deep learning". In: *RSC Advances* 11.55 (2021), pp. 34425–34431 (cit. on p. 29).

[41] Y. Wang, H. Yao, and S. Zhao. "Auto-encoder based dimensionality reduction". In: *Neurocomputing* 184 (2016), pp. 232–242 (cit. on p. 24).

[42] S. Weng et al. "Deep learning networks for the recognition and quantitation of surface-enhanced Raman spectroscopy". In: *Analyst* 145.14 (2020), pp. 4827–4835 (cit. on p. 29).

[43] "What Is Raman Spectroscopy?" In: HORIBA. URL: https://www.horiba.com/usa/raman-imaging-and-spectroscopy/ (visited on 02/01/2022) (cit. on p. 17).

[44] Q. Xianting and W. Pan. "A Density-Based Clustering Algorithm for High-Dimensional Data with Feature Selection". In: *2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*. 2016, pp. 114–118. DOI: 10.1109/ICIICII.2016.0038 (cit. on p. 23).

[45] R. Xu and D. Wunsch. "Survey of clustering algorithms". In: *IEEE Transactions on Neural Networks* 16.3 (2005), pp. 645–678. DOI: 10.1109/TNN.2005.845141 (cit. on p. 8).

[46] X. Ying. "An overview of overfitting and its solutions". In: *Journal of Physics: Conference Series*. Vol. 1168. 2. IOP Publishing. 2019, p. 022022 (cit. on p. 25).

[47] J. Zhang, H. Huang, and J. Wang. "Manifold learning for visualizing and analyzing high-dimensional data". In: *IEEE Intelligent Systems* 25.04 (2010), pp. 54–61 (cit. on p. 19).

[48] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: an efficient data clustering method for very large databases". In: *ACM sigmod record* 25.2 (1996), pp. 103–114 (cit. on p. 22).

A

Appendix

A.1 Dataset Details

Table A.1: Mapping between different wines and their type and region

Type	Region	Wines - Training set	Wines - Testing set
Verde	Verde/Northwest	a, h, k, l, t, u, v	b, e, o
Maduro	Alentejo	m, n, w, x, z, ac	d, f, s
	Douro	c, j, p, r, y, ab, ad	g, i, q

A.2 Results of Analysis of Dataset using Full Dimensions and Dimensionality Reduction

The following results (Figures A.1, A.2, A.3, A.4, A.5, A.6) were obtained by using the dataset with its original 1015 features and by using that dataset after dimensionality reduction using machine learning algorithms like PCA, ISOMAP and t-SNE.

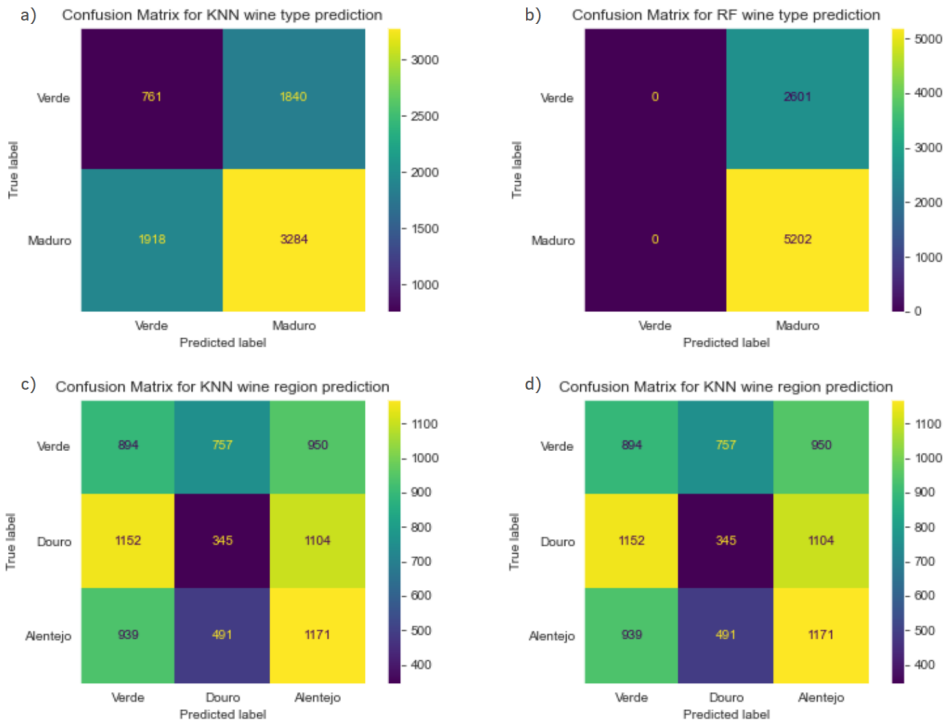

Figure A.1: Confusion Matrices for type and region classification of original dataset normalised by column. The confusion matrices presented are of: a) KNN wine type classification; b) RF wine type classification; c) KNN wine region classification and d) RF wine region classification.

A.3 Autoencoder results

The following tables contain the results obtained from the models after being trained with different combinations of dataset and preprocessing of the data. For all tables and results, the validation loss was obtained from the comparison of the reconstructed data and the input data after the final epoch of autoencoder training. KNN and RF classification accuracy for the predictions of wine region KNN acc Regions and RF acc Regions and for the predictions of wine type KNN acc Type and RF acc Type given the representation of the spectra obtained from the trained encoder.

Table A.2: Results of autoencoder training and encoder prediction for all architectures after training of 500 and 1000 epochs with dataset composed of original dataset with all 1015 features, and normalised by row.

Training with Row Normalised Data with all Original Features

	500 epochs					1000 epochs				
	Val Loss	KNN acc Regions	RF acc Regions	KNN acc Type	RF acc Type	Val Loss	KNN acc Regions	RF acc Regions	KNN acc Type	RF acc Type
1015x128x32x8	0.0037	28.5%	34.7%	59.2%	66.2%	0.0028	30.7%	35.2%	60.0%	66.1%
1015x128x32x8x4	0.0055	28.8%	34.5%	58.7.%	62.5%	0.0043	31.9%	33.9%	55.7%	54.3%
1015x128x32x8x4x2	0.0056	25.9%	25.6%	47.7%	52.7%	0.0055	29.5%	26.1%	51.4%	65.6%
1015x64x32x8	0.0033	31.4%	27.5%	58.3%	65.6%	0.0054	35.0%	28.6%	60.3%	66.5%
1015x64x32x8x4	0.0037	32.7%	39.8%	59.8%	64.7%	0.0061	29.9%	38.5%	56.7%	67.8%
1015x64x32x8x4x2	0.0054	25.2%	23.3%	54.5%	65.1%	0.0059	29.1%	28.0%	50.4%	59.9%
1015x64x32x16x8	0.0032	35.2%	32.0%	60.5%	65.1%	0.0031	27.5%	30.1%	54.9%	66.6%
1015x64x32x16x8x4	0.0047	28.9%	43.1%	55.4%	68.8%					
1015x64x32x16x8x4x2	0.0058	29.7%	28.9%	49.3%	63.8%	0.0060	29.9%	36.6%	50.8%	60.9%
1015x128x64x32x16x8	0.0036	33.2%	32.7%	52.5%	67.2%	0.0052	28.1%	42.1%	59.3%	66.3%
1015x128x64x32x16x8x4	0.0032	33.2%	32.8%	52.5%	67.2%	0.0040	38.4%	31.5%	56.7%	68.4%
1015x128x64x32x16x8x4x2	0.0060	30.1%	25.8%	52.9%	64.1%	0.0057	27.7%	34.8%	50.1%	66.3%

Table A.3: Results of autoencoder training and encoder prediction for all architectures after training of 500 epochs with dataset composed of original **dataset with all 1015 features**, and **normalised by column**.

	Training with Column Normalised Data - 500 epochs				
	Val Loss	KNN acc Regions	RF acc Regions	KNN acc Type	RF acc Type
1015x128x32x8	0.0169	29.4%	43.3%	55.6%	65.5%
1015x128x32x8x4	0.0223	32.1%	34.6%	51.9%	66.6%
1015x128x32x8x4x2	0.0437	39.9%	40.7%	60.4%	64.8%
1015x64x32x8	0.0140	32.6%	42.0%	57.3%	66.3%
1015x64x32x8x4	0.0238	31.0%	38.8%	54.3%	66.5%
1015x64x32x8x4x2	0.0519	39.5%	44.8%	61.4%	68.9%
1015x64x32x16x8	0.0166	30.7%	40.0%	54.3%	66.7%
1015x64x32x16x8x4	0.0290	33.6%	37.2%	54.1%	66.7%
1015x64x32x16x8x4x2	0.0414	40.8%	41.2%	60.4%	66.7%
1015x128x64x32x16x8	0.0206	32.7%	39.0%	49.1%	66.6%
1015x128x64x32x16x8x4	0.0208	37.3%	35.0%	56.3%	68.2%
1015x128x64x32x16x8x4x2	0.0414	39.8%	38.5%	59.8%	66.7%

Table A.4: Results of autoencoder training and encoder prediction for all architectures after training of 500 and 1000 epochs with **dataset composed of the mean of every three features** in the original dataset, and **normalised by row**.

Training with Row Normalised Data of Mean of Every 3 Features

	500 epochs					1000 epochs				
	Val Loss	KNN acc Regions	RF acc Regions	KNN acc Type	RF acc Type	Val Loss	KNN acc Regions	RF acc Regions	KNN acc Type	RF acc Type
339x128x32x8	0.0028	29.3%	32.8%	58.1%	65.1%	0.0065	28.3%	38.5%	58.8%	64.8%
339x128x32x8x4	0.0042	31.1%	35.2%	60.4%	66.3%	0.0066	27.4%	35.1%	57.2%	64.9%
339x128x32x8x4x2	0.0055	26.0%	36.5%	48.1%	65.9%	0.0061	29.7%	35.6%	52.3%	66.4%
339x64x32x8	0.0027	37.1%	37.9%	62.6%	64.7%	0.0066	33.4%	38.8%	55.1%	64.8%
339x64x32x8x4	0.0041	27.7%	32.2%	58.7%	65.8%	0.0039	27.9%	37.8%	57.3%	65.6%
339x64x32x8x4x2	0.0055	28.0%	31.4%	52.7%	61.7%	0.0054	30.2%	30.6%	53.6%	65.7%
339x64x32x16x8	0.0032	34.6%	39.1%	62.6%	68.8%	0.0032	29.4%	32.1%	56.8%	60.8%
339x64x32x16x8x4	0.0036	28.4%	41.3%	59.5%	63.3%	0.0044	27.9%	37.9%	57.3%	65.6%
339x64x32x16x8x4x2	0.0056	30.0%	29.3%	50.4%	55.6%	0.0056	30.2%	30.5%	53.6%	65.7%
339x128x64x32x16x8	0.0028	35.1%	35.6%	59.1%	65.9%	0.0037	31.3%	46.8%	60.1%	62.6%
339x128x64x32x16x8x4	0.0045	36.2%	25.4%	61.3%	65.7%	0.0039	31.5%	34.3%	62.7%	66.2%
339x128x64x32x16x8x4x2	0.0064	30.0%	32.3%	51.4%	59.7%	0.0064	26.4%	32.5%	56.4%	61.3%

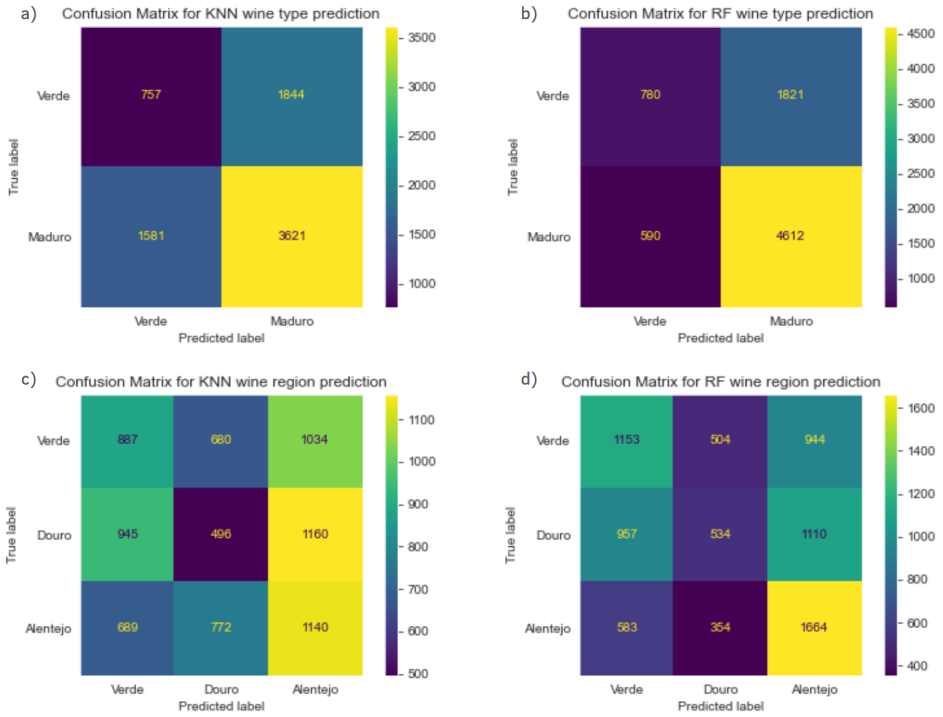

Figure A.2: Confusion Matrices for type and region classification of original dataset normalised by row. The confusion matrices presented are of: a) KNN wine type classification; b) RF wine type classification; c) KNN wine region classification and d) RF wine region classification.

Table A.5: Results of autoencoder training and encoder prediction for all architectures after training of 500 epochs with **dataset composed of the mean of every three features** in the original dataset, and **normalised by column**.

	Training with Column Normalised Data - 500 epochs				
	Val Loss	**KNN acc Regions**	**RF acc Regions**	**KNN acc Type**	**RF acc Type**
339x128x32x8	0.0174	30.8%	42.8%	57.0%	66.3%
339x128x32x8x4	0.0260	32.2%	36.1%	52.2%	66.3%
339x128x32x8x4x2	0.0436	40.1%	44.1%	61.0%	66.7%
339x64x32x8	0.0404	41.4%	42.7%	62.9%	64.9%
339x64x32x8x4	0.0245	33.7%	35.0%	50.7%	66.5%
339x64x32x8x4x2	0.0413	39.2%	37.7%	58.8%	66.7%
339x64x32x16x8	0.0208	27.0%	37.9%	53.7%	65.8%
339x64x32x16x8x4	0.0224	32.3%	36.1%	52.7%	66.7%
339x64x32x16x8x4x2	0.0454	40.5%	36.9%	61.2%	66.7%
339x128x64x32x16x8	0.0184	31.9%	44.0%	57.1%	68.0%
339x128x64x32x16x8x4	0.0238	30.1%	39.2%	56.2%	65.9%
339x128x64x32x16x8x4x2	0.0404	41.4%	42.7%	62.9%	64.9%

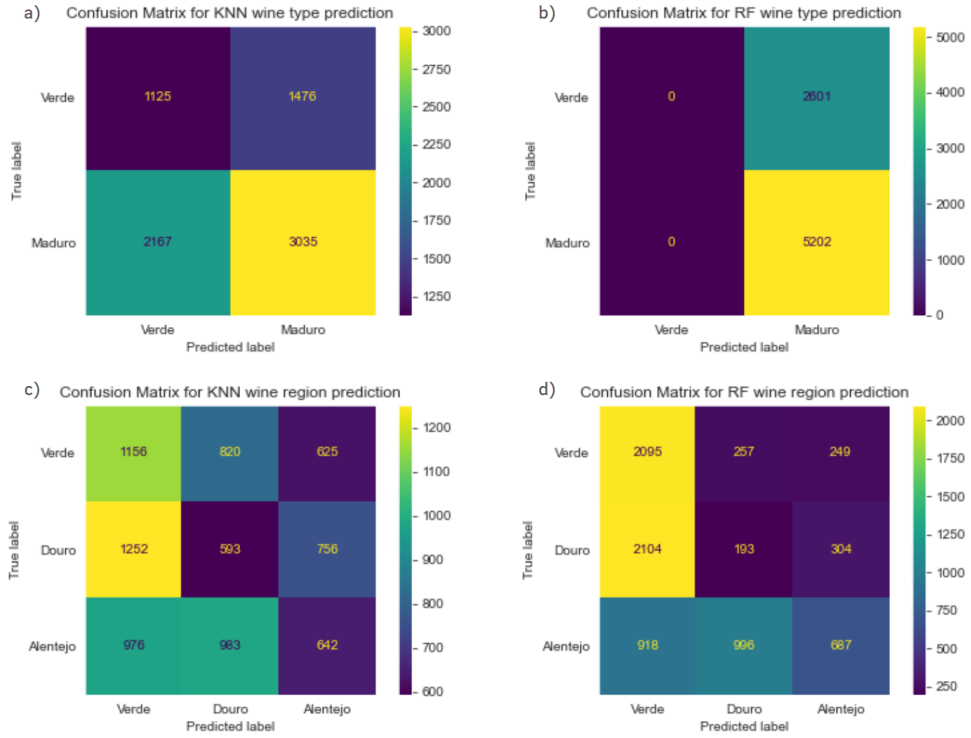

Figure A.3: Confusion Matrices for type and region classification of original dataset normalised by column and after dimensional reduction to 2-dimensions using the t-SNE algorithm. The confusion matrices presented are of: a) KNN wine type classification; b) RF wine type classification; c) KNN wine region classification and d) RF wine region classification.

Table A.6: Results of autoencoder training and encoder prediction for all architectures after training of 1000 epochs with **dataset composed of the mean of every five features** in the original dataset, and **normalised by row**.

	Training with Row Normalised Data - 1000 epochs				
	Validation Loss	**KNN acc Regions**	**RF acc Regions**	**KNN acc Type**	**RF acc Type**
203x128x32x8	0.0028	34.4%	48.7%	58.6%	60.0%
203x128x32x8x4	0.0061	29.9%	38.5%	56.7%	67.8%
203x128x32x8x4x2	0.0064	29.8%	26.3%	58.1%	65.4%
203x64x32x8	0.0043	30.4%	34.0%	58.2%	66.4%
203x64x32x8x4	0.0039	26.6%	31.1%	55.1%	64.4%
203x64x32x8x4x2	0.0056	28.8%	25.3%	50.5%	63.6%
203x64x32x16x8	0.0056	29.5%	38.5%	58.6%	64.3%
203x64x32x16x8x4	0.0043	30.8%	25.3%	58.6%	66.8%
203x64x32x16x8x4x2	0.0051	29.8%	21.6%	56.2%	65.8%
203x128x64x32x16x8	0.0038	34.3%	36.0%	60.1%	65.0%
203x128x64x32x16x8x4	0.0049	28.9%	36.9%	60.6%	65.7%
203x128x64x32x16x8x4x2	0.0060	34.9%	26.7%	51.9%	60.3%

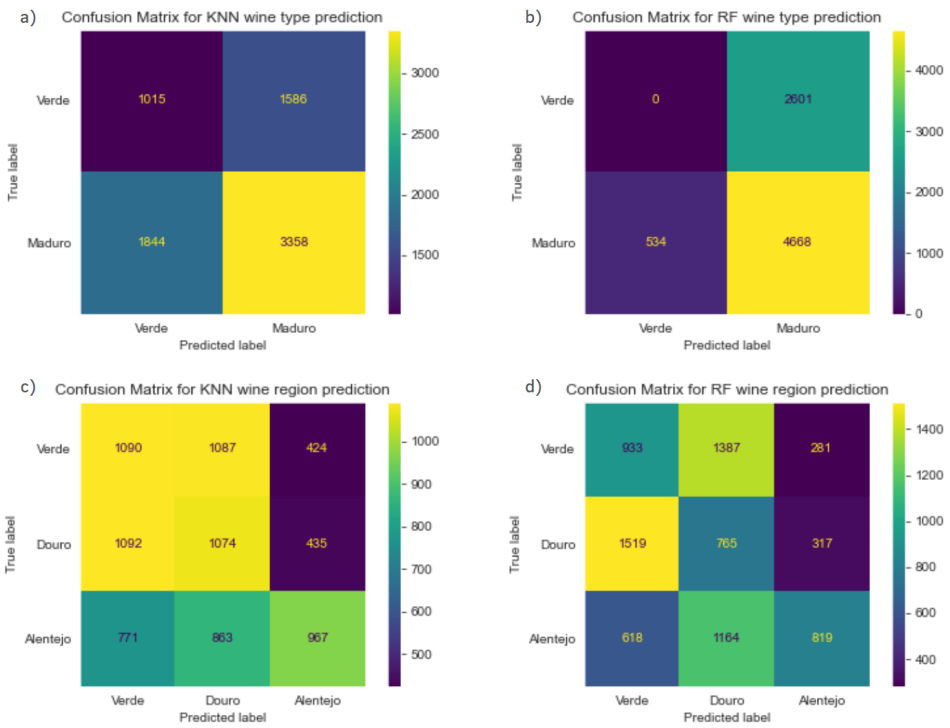

Figure A.4: Confusion Matrices for type and region classification of original dataset normalised by row and after dimensional reduction to 2-dimensions using the t-SNE algorithm. The confusion matrices presented are of: a) KNN wine type classification; b) RF wine type classification; c) KNN wine region classification and d) RF wine region classification.

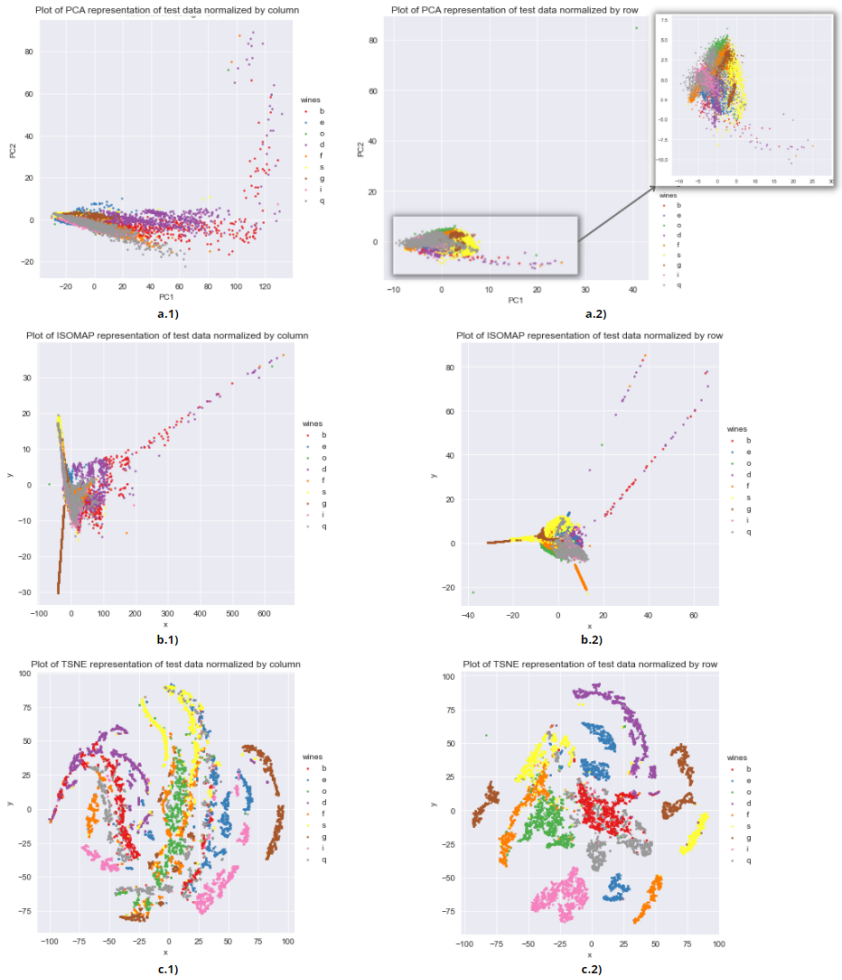

Figure A.5: Plot of test dataset after dimensionality reduction with different preprocessing techniques. (a.1) PCA representation of test data normalised by column; (a.2) PCA representation of test data normalised by row; (b.1) ISOMAP representation of test data normalised by column; (b.2) ISOMAP representation of test data normalised by row; (c.1) t-SNE representation of test data normalised by column; (b.2) t-SNE representation of test data normalised by row.

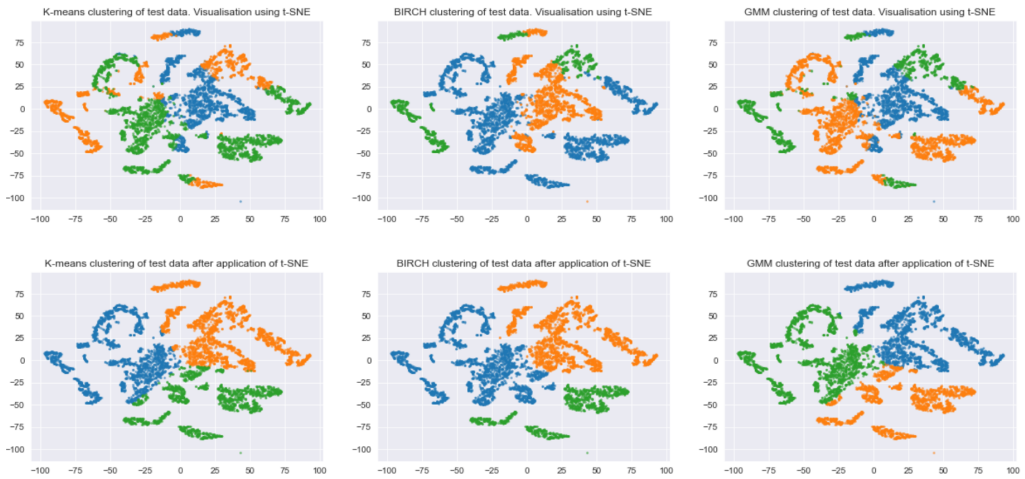

Figure A.6: Plots of clusters formed when applying K-Means, BIRCH and GMM clustering algorithms to test set with original dimensions (3 plots of first row) and plots of clusters formed when applying K-Means, BIRCH and GMM clustering algorithms to test set after dimensionality reduction to 2-dimensions using t-SNE