Homework 1: Nearest Neighbors, Linear SVM, SVM with RBF Kernel

Sanguineti, Francisco Javier (s279265)

Professor: Barbara Caputo

Teaching Assistant: Antonio D'Innocente

01TXFSM Machine Learning and Deep Learning - A.A 2019/20 - Data Science and Engineering Master's Degree - Politecnico di Torino

Contents

Contents **1**

1 Introduction **2**

2 Overview of classification algorithms **3**

2.1 Nearest Neighbors 3

2.2 Support Vector Machine 3

2.2.1 Linearly separable classes 4

2.2.2 Non linear extension 4

2.2.3 The multiclass extension 5

3 The data **6**

3.1 An overview of the data 6

3.2 Data projection 6

4 Methodology **8**

4.1 Training procedure 8

4.2 Evaluating a model 8

4.2.1 Accuracy 8

4.2.2 Cross Validation 8

4.3 Grid Search 9

5 Analysis and Results **10**

5.1 Nearest Neighbors 10

5.2 Linear SVM 11

5.3 SVM with RBF Kernel 14

5.3.1 Tuning the C parameter 14

5.3.2 Tuning gamma and C 16

5.3.2.1 Without cross validation 16

5.3.2.2 With 5-fold cross validation 17

6 Conclusion **20**

References **21**

1 Introduction

The main purpose of this report is to provide a brief explanation of the data used, the logic of the methods, and the outcomes of the first homework of the Machine Learning and Deep Learning course. This assignment is about the usage of different types of classification algorithms, and how their results are influenced by their hyper-parameters and the data used.
To perform this homework, the *Python* programming language has been used; more specifically the $Scikit - Learn$ library, also known as sklearn.

2 Overview of classification algorithms

The algorithms described below belong to the group of supervised learning algorithms. A supervised machine learning algorithm is one that relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data. This type of algorithms is commonly used to solve classification problems.
Classification problems are generally characterized by having a dataset $S = (x_1, y_1), ..., (x_m, y_m)$, where $x_i \in \mathbb{R}_N$ is the i-th N-dimensional data point (or feature vector) from M samples and $y_i \in \{1, ..., K\}$ is the categorical outcome (or class label) corresponding to each data point.
The goal of these algorithms is then, to learn a mapping function $y = f(x) : \mathbb{R}_N \rightarrow \mathbb{Z}$, such that, given an unseen sample $x_0 \in \mathbb{R}_N$ we can predict its label.

2.1 Nearest Neighbors

The nearest-neighbors algorithm, or also called k-nearest neighbor (KNN), is a supervised lazy learning algorithm based on learning by analogy. Its basic assumption is that data points of similar classes are closer to each other, and this "closeness" is defined in terms of Euclidean distance.
It makes use of all the training data to predict the class of a new sample according to the majority of the k-nearest neighbor's category. This is why this classifier does not train any model. It instead creates a new one when it needs to predict a new, unseen sample, basing itself only on memory (lazy learner).
The K in this algorithm is a positive integer hyper-parameter that defines the number of nearest neighbors to include in the majority of the voting process.

2.2 Support Vector Machine

This algorithm, introduced by Vapnik [1], involves drawing a boundary between groups of samples that fall into different classes. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the neighboring data points of the classes.
The basic SVM supports only binary classification, it will construct the most favorable separating hyperplane in the data space: one which maximizes the margin between the two data sets.

2.2.1 Linearly separable classes

There are some variations of SVM, first we have the Hard Margin SVM that finds the solution with maximum margin, but when the training samples are not linearly separable, it has no solution.
The optimization problem that it tries to solve is the following:

$$\begin{gathered}\underset{w,b}{\text{minimize}}\ \frac{1}{2}\|w\|^2 \\ \text{subject to } y_i[\langle x_i\,, w\rangle + b] \geq 1\end{gathered} \tag{1}$$

where w is the normal vector to the hyperplane, and $\frac{b}{w}$ determines the offset of the hyperplane from the origin along the normal vector w.
Therefore, an extension of Hard Margin SVM is the Soft Margin SVM that allows a controlled classification error: it tries to maximize a margin while trying to minimize the distance between misclassified points and their correct margin plane. It can be written as:

$$\begin{gathered}\underset{w,b}{\text{minimize}}\ \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \\ \text{subject to } y_i[\langle x_i\,, w\rangle + b] \geq 1 - \xi_i \text{ and } \xi_i \geq 0\end{gathered} \tag{2}$$

where ξ_i represents the slack for each data point i, which allows misclassification of data points while the data is not linearly separable.
The C hyper-parameter in this algorithm represents the cost of misclassification, also it is known as the penalty error.

2.2.2 Non linear extension

Support Vector Machines can be used to perform non-linear classification with a kernel trick. A kernel transforms (maps) feature vectors to a higher dimensional space, making it possible for a linear classifier to separate them. The back-projection of the optimal separating hyperplane from this new feature space to the original input data space will then result in a non-linear boundary of given complexity that better suits the distribution, providing the feature space is correctly defined.
The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function.
It make use of the dual soft margin problem described as follows

$$\begin{gathered}\underset{\alpha}{\text{maximize}}\ -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle x_i\,, x_j\rangle + \sum_i \alpha_i \\ \text{subject to } \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, C]\end{gathered} \tag{3}$$

where α_i are defined such that $w = \sum_i y_i \alpha_i x_i$.
SVM has the flexibility to handle many types of kernels, the one used in this homework is the Radial Basis Function (RBF), also called Gaussian kernel, and is defined as

$$K(x, x') = exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \tag{4}$$

where x and x' are samples represented as feature vectors in the input space. The $\|x - x'\|^2$ is known as the squared Euclidean distance between the two feature vectors, and σ is a free parameter.
Letting be $\gamma = \frac{1}{2\sigma^2}$, an equivalent definition is

$$K(x, x') = exp\left(-\gamma\|x - x'\|^2\right) \tag{5}$$

The hyper-parameter γ defines the sharpness of the RBF distribution. A small γ widens the distribution, making the influence of each sample diffuse giving a more regularized classification. A large γ sharpens the influence of each data point, giving a less regularized (tighter fitting) classification.

2.2.3 The multiclass extension

The multiclass classification problem can be decomposed into several binary classification tasks that can be solved using binary classifiers.
In this assignment we will use the **One-versus-all**, whose approach is to reduce the problem of classifying among K classes into K binary problems, where each problem discriminates between a given class and the other $K1$ classes. It will require K binary classifiers, where the $k-th$ classifier is trained with positive samples belonging to class k and negative samples belonging to the other $K1$ classes.
When predicting an unknown sample, the classifier that produces the maximum output is considered the winner, and this class label is assigned to that sample.

3 The data

The dataset used in this experience is the *WineDataSet* provided by the Scikit-Learn library. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.
In a classification context, this is a well-posed problem with "well behaved" class structures.

3.1 An overview of the data

The wine data has three classes, thirteen features and 178 instances or samples, and the distribution of this classes are:

class 0: 59 samples
class 1: 71 samples
class 2: 48 samples

We could say that it is a little unbalanced.

3.2 Data projection

One of the assignment points establishes us to work only with the first two features (**alcohol** and **malic acid**), therefore, we have to project our dataset into a 2D subset.

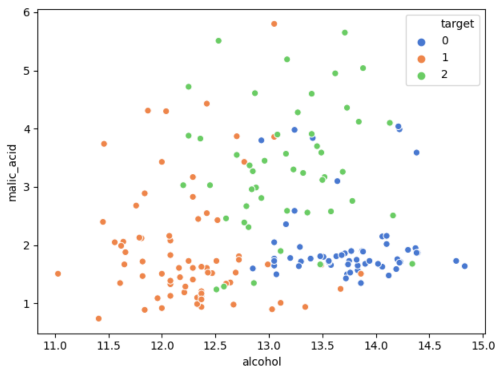

Figure 1: 2D space illustration of the data

Figure 1 shows that despite a little overlap, we could separate the data into 3 regions or clusters. This will be the challenge for our classifiers to overcome.

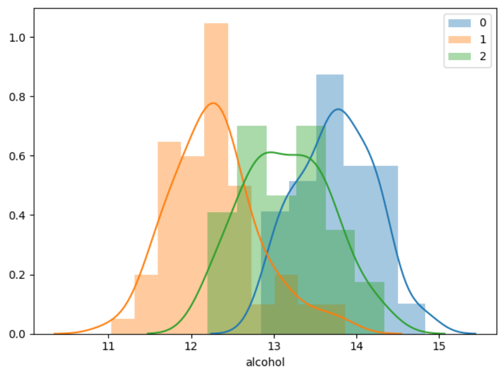

Figure 2: Histogram fitted with kernel density estimate of the alcohol between the classes

In Figure 2 is illustrated that there is a distinction between the mean of alcohol in each class.
There can also be noted that although there is a clear separation, a bit of overlapping is seen between differently labeled points. Classifiers might be expected to behave differently in these regions.

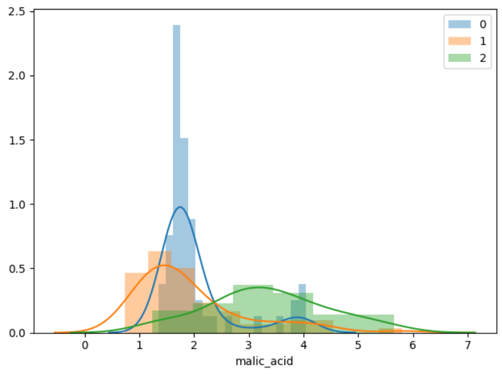

Figure 3: Histogram fitted with kernel density estimate of the malic acid between the classes

In terms of the features taken into consideration, it can be seen that the "alcohol" distributions are more meaningful for an accurate classification because they are less sparsely distributed. However, these clusters can't be separated using only this attribute.
From Figure 3 it can be said that there is no clear conclusion to extract yet at this point in the analysis.

4 Methodology

As we know, this homework aims to find how to choose the hyperparameters for the classification algorithms. How the methodology of this experience is stipulated is given by the assignment, and it is described below.

4.1 Training procedure

Given the discussed dataset, the one that contains two features, to analyze the performance and evaluate a model we will need to divide this dataset into 3 splits: **training**, **validation** and **test**, and being their proportions 50%, 20%, and 30% respectively of the dataset. The training procedure will consist of

1. Training the model using only the training set with the parameters chosen.
2. Evaluating the performance of the model on the validation set.
3. Choosing other parameters and repeating 1. and 2. until the parameters that work best on the validation set are found.

In the end we will evaluate the model on the test set without changing the parameters in function of the performance on the testing set.

4.2 Evaluating a model

Evaluation metrics explain the performance of a model. An important aspect of evaluation metrics is their capability to discriminate among model results.

4.2.1 Accuracy

In the majority of this experience we are going to use the **accuracy** as a metric to evaluate the performance of a model. It is defined as the ratio of the number of correct predictions to the total number of input samples.

4.2.2 Cross Validation

Cross validation (CV) is a resampling procedure used to evaluate machine learning algorithms. The basic approach that we will use is called **k-fold cross validation**, it consists in split the training set into k smaller sets and applying the following procedure for each of the k "folds":

- A model is trained using $k - 1$ of the folds as training data

- The resulting model is validated on the remaining part of the data.

The performance measure reported by k-fold CV is then the average of the values computed in the loop.
A test set should still be held out for final evaluation, but the validation set is no longer needed when doing CV.

4.3 Grid Search

For those classification algorithms that require to optimize more than one hyper-parameter, as the SVM with RBF Kernel, we will use the Grid Search algorithm which is simply an exhaustive searching through a specified subset of the hyperparameters of a learning algorithm.
This algorithm must be guided by some performance metric, in our case, the metric chosen is the accuracy. It is typically measured by cross-validation on the training set or evaluation on a held-out validation set.

5 Analysis and Results

5.1 Nearest Neighbors

The K in the Nearest Neighbors algorithm is a parameter that we have to choose in order to obtain the best possible fit on the data set. Intuitively, we can think of K as the control of the shape of the algorithm's decision boundary.
These are the 2-dimensional graphs of the KNN's decision boundaries with four different K values: 1, 3, 5, and 7.

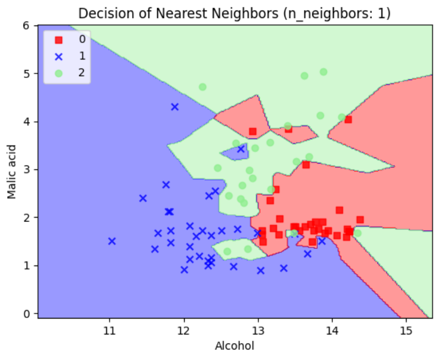

Figure 4: Decision boundaries of the KNN with k = 1

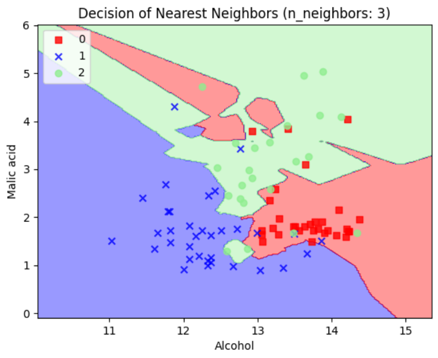

Figure 5: Decision boundaries of the KNN with k = 3

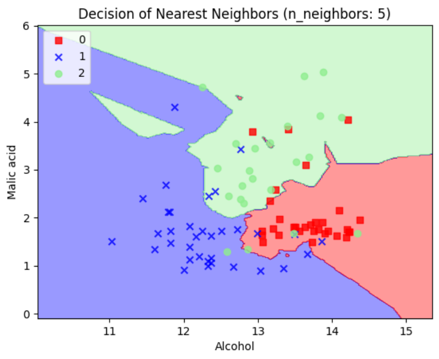

Figure 6: Decision boundaries of the KNN with k = 5

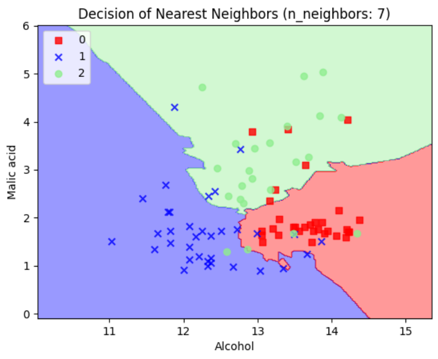

Figure 7: Decision boundaries of the KNN with k = 7

As we can see, it follows the expected behavior, for small values of K as 1 or 3, the algorithm is more flexible with the outliers, that is why in Figure 4

and 5 there are small islands of likely incorrect predictions. Differently, the classifier smoothens these anomalies with higher values of K as 5 and 7, which will possibly result in a greater generalization. Despite the irregularities of the classifier on small values of K, the separation of the clusters is not so bad, but it is better when K rises.
The accuracy's scores on the validation set of the classifier for each value of K are shown in Figure 8.

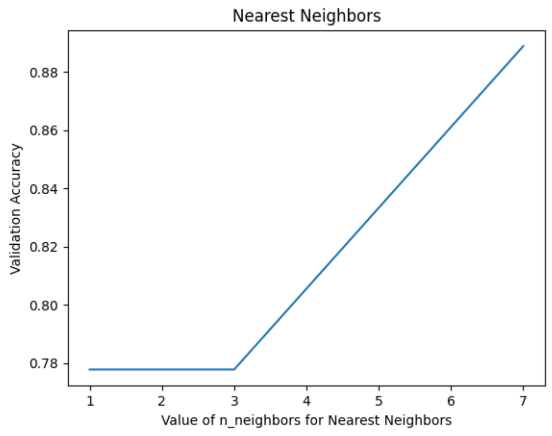

Figure 8: Accuracy on validation set of KNN for K = 1, 3, 5, 7

The highest accuracy is with a value of K equal to 7, with a score of 0.8889. This model was also evaluated on the test set and it got an accuracy score of 0.7778. This difference can tell us that the model is slightly overfitted on the training dataset, also these results depend more on the distribution of classes in each dataset.

5.2 Linear SVM

The following analysis is to determine the C penalty factor of a Linear SVM model. To recap briefly, what this parameter represents is how to control the trade-off between the hyperplane complexity and training errors. In this homework it is set the possible range of C to 0.001, 0.01, 0.1, 1, 10, 100, 1000.

High C values force the error to be smaller, approximating the behavior of hard margin SVM. The figures 9 to 15 shows the effect of C on the decision boundary. A too-small C (0.01) or less, allows as many mistakes that the model cannot even distinguish between the different classes. Small C (C = 0.1) however allows be more strict with the errors, that is why it can separate

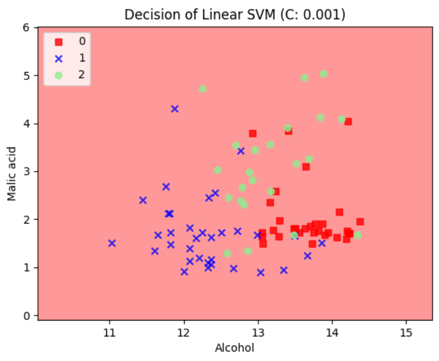

Figure 9: Decision boundaries of the Linear SVM with C = 0.001

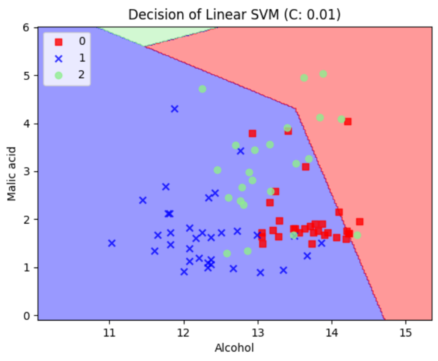

Figure 10: Decision boundaries of the Linear SVM with C = 0.01

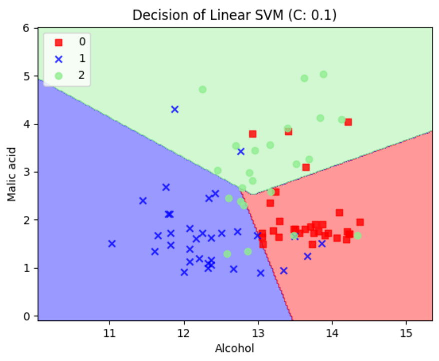

Figure 11: Decision boundaries of the Linear SVM with C = 0.1

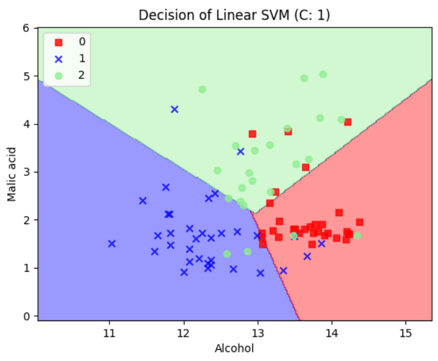

Figure 12: Decision boundaries of the Linear SVM with C = 1

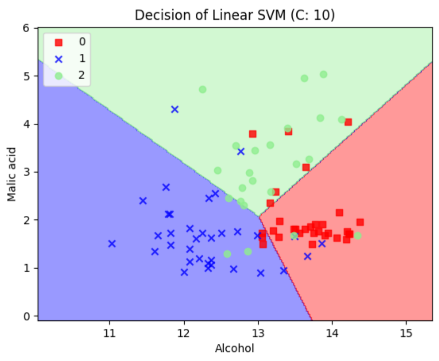

Figure 13: Decision boundaries of the Linear SVM with C = 10

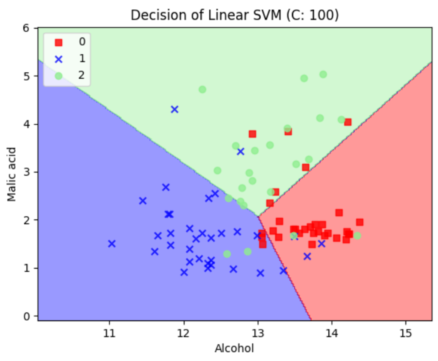

Figure 14: Decision boundaries of the Linear SVM with C = 100

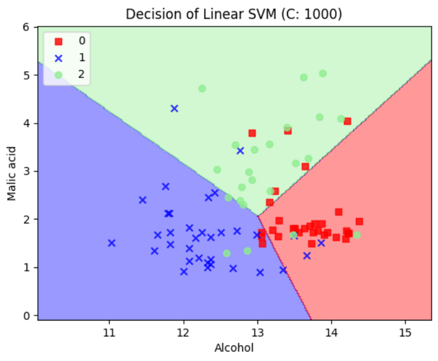

Figure 15: Decision boundaries of the Linear SVM with C = 1000

the big clusters. The shape of Figure 12, C=1 is typically preferred because it represents a good trade-off between acceptable classifier performance and generalization. Larger values of C does not allow much training error, so they tend to overfit.
It can be seen in the Figure 16 the accuracy in the validation set for each model of the Linear SVM with its parameter C. The best validation score was found with a value of C=1, this score is 0.8611. Also this model was evaluated in the test and it reached a score of 0.7593.

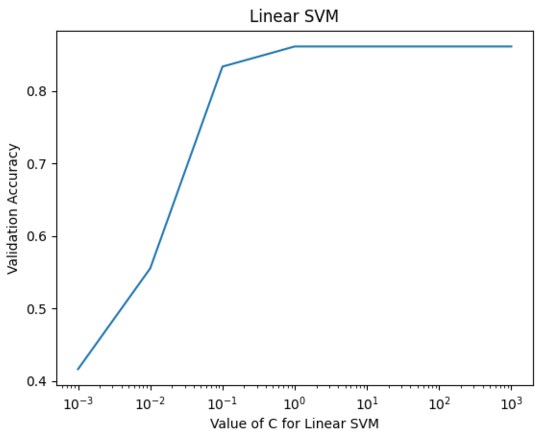

Figure 16: Accuracy on validation set of Linear SVM on the different values of C

The difference between the validation score and the test score is probably due to overfitting. We are selecting the model with the best scores in the validation set that has a very different class distribution from the test set.

5.3 SVM with RBF Kernel

This section is composed of two parts. One of them is about finding the best C parameter of an SVM with RBF Kernel maintaining the gamma parameter constant, and the other part is about tuning the classifier over both parameters.

5.3.1 Tuning the C parameter

This analysis is similar to the one of the Linear SVM, it consists of defining the C parameter of an SVM with RBF Kernel. As in the previous experience, the range of C is settled as 0.001, 0.01, 0.1, 1, 10, 100, 1000.
When combined with an RBF (or Gaussian) kernel, large values for parameter C can dramatically overfit the data. This caused the isolated region of classification at the decision boundary shown in Figure 23.
Comparing the accuracy on the validation set of each model (Figure 24), it is found that the best score is 0.8889 where C is equal to 100. This model had an accuracy score of 0.7778 in the test set.

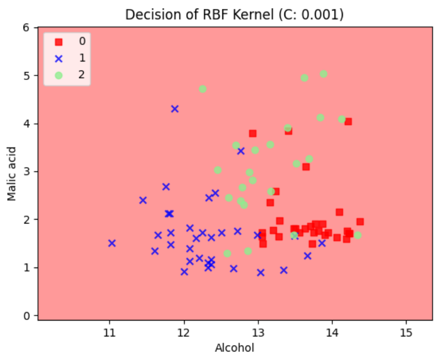

Figure 17: Decision boundaries of the RBF SVM with C = 0.001

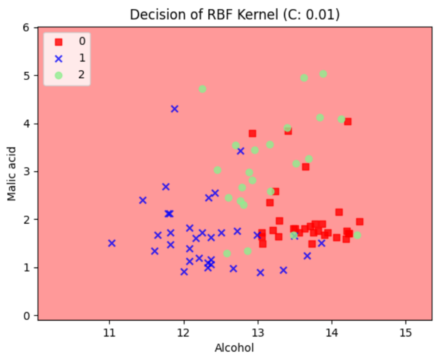

Figure 18: Decision boundaries of the RBF SVM with C = 0.01

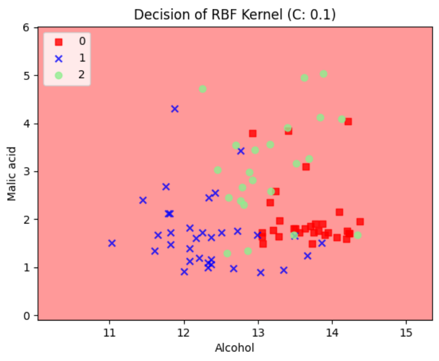

Figure 19: Decision boundaries of the RBF SVM with C = 0.1

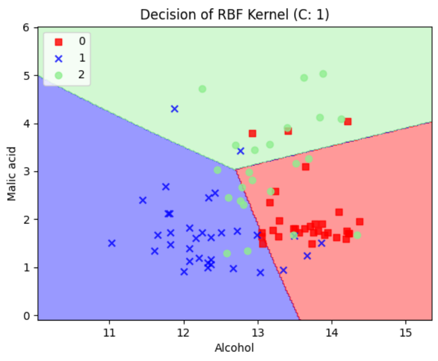

Figure 20: Decision boundaries of the RBF SVM with C = 1

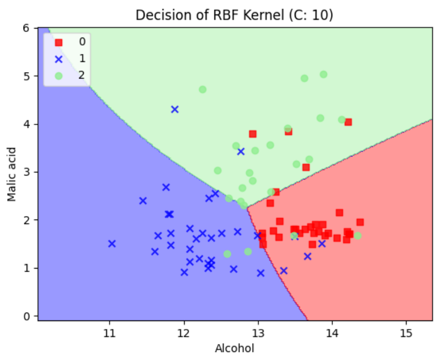

Figure 21: Decision boundaries of the RBF SVM with C = 10

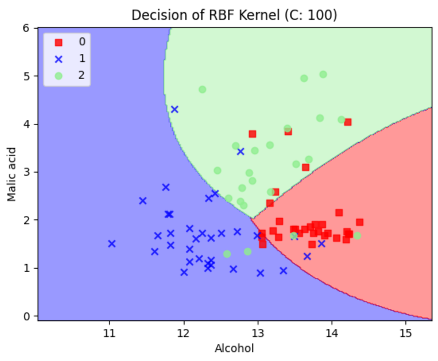

Figure 22: Decision boundaries of the RBF SVM with C = 100

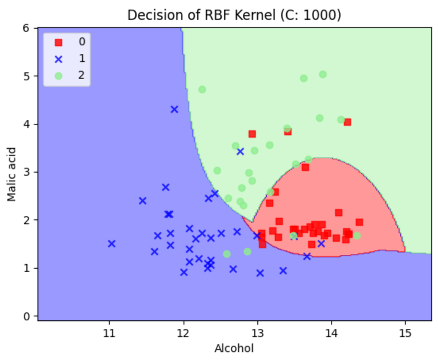

Figure 23: Decision boundaries of the RBF SVM with C = 1000

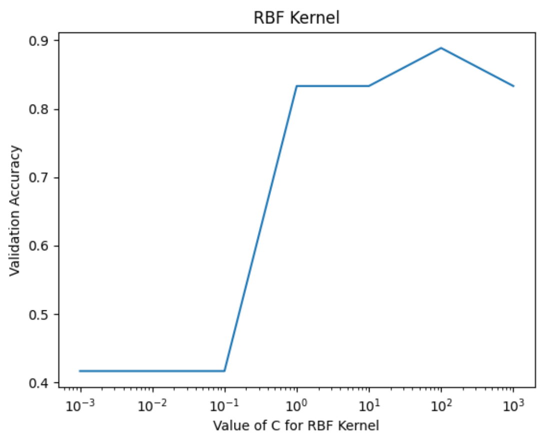

Figure 24: Accuracy on validation set of Linear SVM on the different values of C

5.3.2 Tuning gamma and C

Tuning over both parameters is a good option to find a model that fits better the data.
Intuitively, the gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.
For this analysis of the parameters, we performed a grid search over them, which range of each parameter are:

C: 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000
gamma: 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100

5.3.2.1 Without cross validation

In this part of the analysis we simply use the same procedure as in the previous ones, we have our training set and validation set. Each combination of C and gamma is evaluated on the validation set, Figure 25 shows the comparison between the accuracy of the models.
We found that the higher accuracy is from the combination of C = 1000 and gamma = 100. As it is shown in Figure 26, this model is severally overfitted, which explains the apparition of small islands due to the sensibility to the gamma parameter. When gamma is too large, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C can prevent overfitting. As expected, this model obtains a low score on the test set, the accuracy was 0.5556.

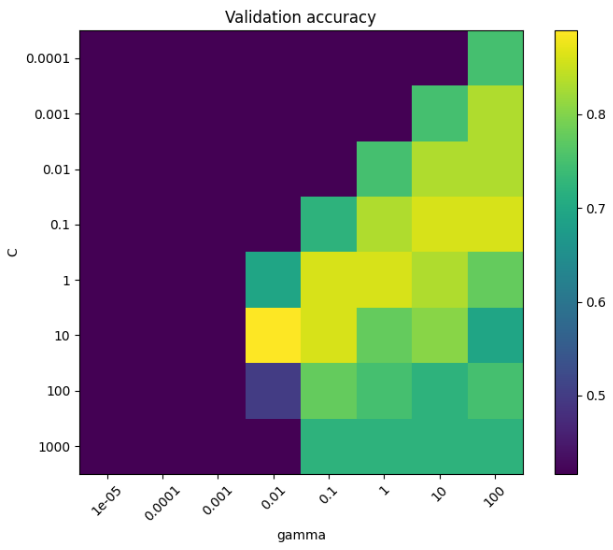

Figure 25: Grid search of the RBF SVM without cross validation

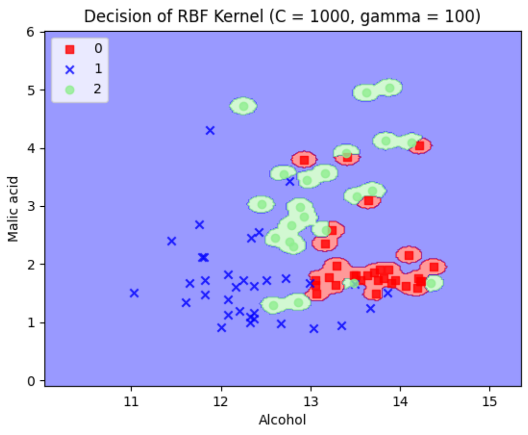

Figure 26: Decision boundaries of the RBF SVM with C = 1000 and gamma = 100

5.3.2.2 With 5-fold cross validation

To achieve better generalization of our RBF SVM classifier, we have to make use of a cross-validation method, consequently, a higher accuracy on the test set will be obtained.
The cross-validation method used in this analysis is the 5-fold CV, it consists of splitting our training data randomly into five subsets, making the validation accuracy an average over these subsets.

In Figure 27 we can observe this cross-validation accuracy obtained by the grid search procedure, comparing it with the one in the Figure 25 it is appreciated how conditioned are the grid search algorithms to the evaluating metric, the results are very different. In Figure 27 we found that the highest score is given by the combination of parameters C = 10 and gamma = 1, which is 0.7817. As we can see, it does not seem to be much higher than models of previous analyzes, but we must remember that this value is not the one used to compare the models.

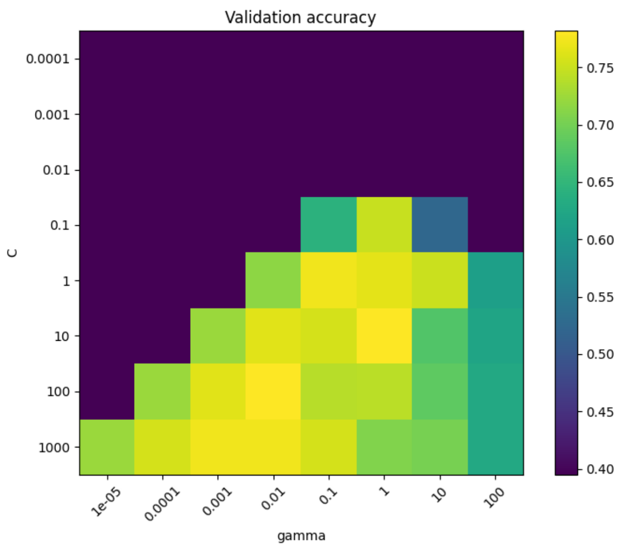

Figure 27: Grid search of the RBF SVM with cross validation

The decision boundaries generated by the classifier with the best score is shown in Figure 28. This model reaches an accuracy of 0.7963 on the test set, it is the highest value of test accuracy obtained in this report.

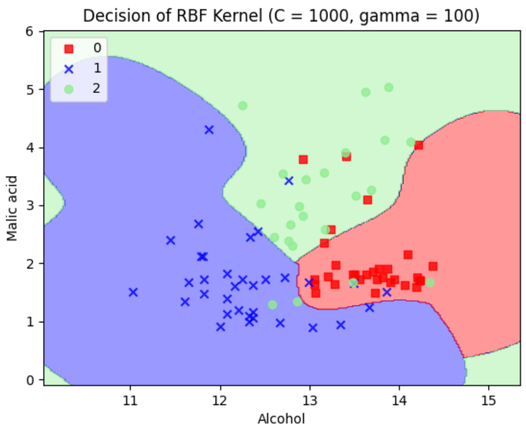

Figure 28: Decision boundaries of the RBF SVM with C = 10 and gamma = 1

6 Conclusion

First of all, we can conclude that the accuracy metric that judges our models is not a clear indicator of the performance, worse it is when classes are imbalanced.
Also, it is found in the results that there is a large gap between the validation scores and the test scores of the analyzed classifiers, it could be said that to obtain better and more reliable results, we must use a cross validation method such as k-fold, and to overcome the unbalanced distribution of classes in the different data sets, we can probably use the stratified division method.
The KNN classifier is a lazy algorithm that is based on the assumption is that data points of similar classes are closer to each other. This provides good properties: it is automatically non-linear, it can detect linear or non-linear distributed data, it is easily extended to multi classes. The main disadvantage for a KNN is that the distance metric is calculated exhaustively to all the training data each time we come across a set of new sample data, it is slow in real-time. It is less computationally at training time but in case of a large dataset, this algorithm could be not a good choice because it scales badly.
On the other hand SVM is an algorithm that guarantees separating optimally the data. For a multi-class classifier, we have to train as many SVM algorithms as classes on the data. SVM is more computationally expensive but that model can be used to predict classes even when we come across new unlabelled data.
Both algorithms have worked well, at the time of choosing one of them, what could conclude is that we should do an analysis of the data and test both to know which one to choose.

References

[1] Corinna Cortes and Vladimir Vapnik, *Support-vector networks*, Machine Learning, 1995, pp. 273–297.