Machine Learning and Artificial Intelligence Homework1

Lorenzo Vaiani

15 November 2019

1 Abstract

In this study two machine learning techniques, KNN and SVM, are used to solve a classification problem. The dataset that contains the samples to classify is called "wine dataset" and at the beginning only the first two features are used. To understand in a better way how the models created by these techniques can classify the elements, the decision boundaries are plotted. For KNN different values of K and two types of weight are tuned. For SVM the linear kernel is used tuning the C hyperparameter, next also the RBF kernel is used tuning at first the same hyperparameter, then tuning C and γ hyperparametrs at the same time. After this, K-fold Cross Validation is applied to evaluate again the RBF with the same hyperparameters range of values. Eventually, always using SVM, all the features are considered to find out the couple that best classifies the dataset.

2 Introduction

In this study we will use different machine learning algorithms in order to solve a classification problem. The entire implementation is made with Python 3.7 and using functions from Scikit-learn, an open source library for machine learning. The dataset used is called "wine dataset" and is provided by Scikit. This dataset has 178 records and each one has 13 attributes and belong to one of three different classes.

At first we will consider only two attributes: alcohol value and malic acid value. In this way we will work in a two-dimensional space and it will be easier to observe the decision boundaries created by learning algorithms.
The dataset elements are sorted according to their class, so after selecting their first two features and their class, they must be mixed randomly. This is done using the "shuffle" function of Python with a seed equal to zero (random_state=0). Repeating the whole experiment changing the data order each time, changing the seed value or removing it, could be useful to have a better evaluation of the generated models.

After this, the dataset is divided in three parts:

- The first one includes 50% of the samples and it is called train set, used to fit the mathematical model of the learning algorithms.
- The second one includes 20% of samples and it is called validation set, used to evaluate the model and to adjust the hyperparameter values based on the evaluation result. It can be considered part of the train set, so it is possible to assume that the data in this set are known to perform future operation like scaling.
- The third one includes the remaining 30% of samples and it is called test set, used to get the final unbiased score of the model. It is assumed that nothing is known about this data during the training phase.

The last thing to do before applying the learning algorithms is to rescale the data in order to prevent that one feature can dominate on the other one. In this case alcohol feature has one order of magnitude more than malic acid feature and this can affect the computation (for example during distance calculation) as the algorithm could consider alcohol values more relevant than the other ones. Rescale process is done using a "StandardScaler" object provided by Scikit, that perform, separately for each feature, the following operation on each value V in order to obtain the scaled version V_s:

$$V_s = \frac{(V - u)}{s} \tag{1}$$

where u is the mean and s is the standard deviation, both calculated on training and validation samples because it is assumed that there are no information about test elements, but also the test set is scalded using these values.

Scaled data of train set can be represent used to show a 2D representation of the samples (figure 1) and to fit the models of machine learning algorithms such as K-Nearest Neighbour (KNN) and Support Vector Machine (SVM).

3 K-Nearest Neighbour

The KNN is a technique used for classification and regression. It is non-parametric, it means that it does not make any assumptions about data distribution. It is based on the Nearest Neighbour Classifier idea which says that a new sample is classified by calculating the distance to the nearest element in the train set, then the class label of that point is used to classify the sample. This idea is extended by taking the K nearest train samples and assigning the class label of the majority of theme. The votes of the nearest K samples can all be considered uniformly or can be weighed based on the distance of the corresponding element from the sample to be classified. It does not construct a general model during the training phase, all the samples in the train set will be kept and will be used to compute the distance. The proximity of the K neighbors

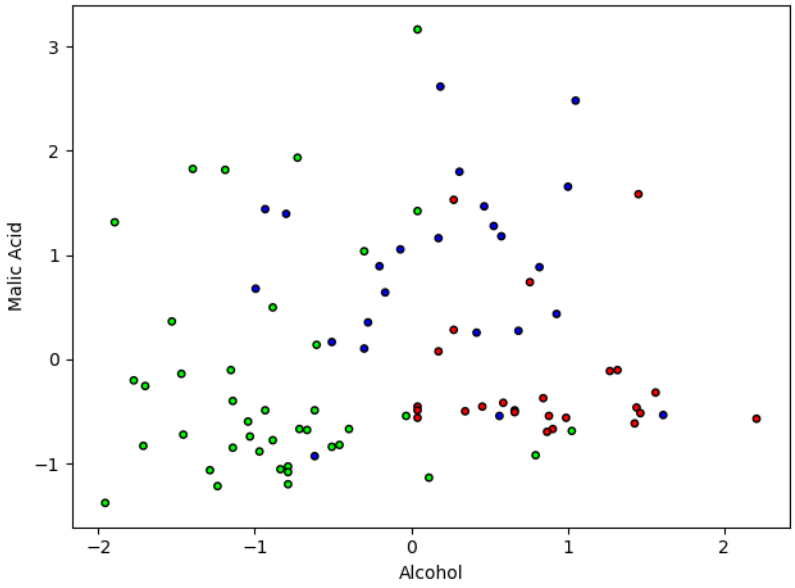

Figure 1: a 2D representation of training set elements, considering only the first two features. The class which an element belongs to is expressed by the color.

to be considered can be defined using different types of distance (Minkowski, Manhattan, Euclidean, Chebyshev, ..). The value of K is usually small and it is also odd when the number of classes is equal to two, in order to avoid the same quantity of votes for both classes. The best value of K depends on the dataset itself and is not known a priori. A greater value of K helps to reduce the effect of noisy points in the train set and prevents overfitting.

In this study KNN technique is performed by a Scikit object called "KNeighborsClassifier". In the computation it uses the Minkowski distance

$$L_p(x, y) = (\sum_{i=1}^{d} |x_i - y_i|^p)^{\frac{1}{p}}$$

with $p = 2$, therefore it corresponds to Euclidean distance

$$L_2(x, y) = (\sum_{i=1}^{d} |x_i - y_i|^2)^{\frac{1}{2}}$$

A model is fit with train data for each combination between both types of weight and every odd value for K, from one to seven, both included. Then, the decision boundaries of each model are plotted and the points of validation set are drawn too (figure 2 and 3). In this way it is possible to observe if a validation sample is situated in a region whose color matches its class label. In

table 1 it is possible to see the accuracies calculated evaluating the models on the validation set. These accuracies are plotted on a graph (figure 4 and 5) to show how they vary when K value changes.

	K = 1	K = 3	K = 5	K = 7
weight = uniform	77.14%	77.14%	80.00%	77.14%
weight = distance	77.14%	77.14%	80.00%	74.28%

Table 1: accuracies for all the KNN models evaluated on the validation set. The best value for each weight is highlighted in red.

There are only small changes in the various decision boundaries. Worthy of note is the fact that, independent of the weight used, for lower K values it is common to see small colored areas completely surrounded or almost (like peninsulas) by a different class label color. This can be caused by the effect of outliers or by an overfitting situation. On the other hand, as K value increases, the regions of the same color have less articulated borders and are more compact. Accuracy values are all quite good, without particular differences when the type of weight used varies, except for the value of $K = 7$ where the distance weight it's slightly worse. The best result is an accuracy of 80.00% and it was obtained with a value of $K = 5$, that will therefore be used to classify the test set sample. This accuracy is the same for both weights, so since it is not possible to know which of the two models is the most adequate to evaluate the test set, the program chooses the one relative to the first best result identified, that in this case is the one built with uniform weight. Using that model to evaluate the test set an accuracy of 81.48% is obtained and this confirms that the model used can be considered good for classify this kinds of element.

Eventually, all the features are considered. The research of the best couple of feature is performed repeating the previous experiment (without plotting the boundaries) for every single pair separately. It is possible to do this because the dataset is very small, the features are not too many and the research is limited to pairs and not to larger groups of features. In fact, test all the couples takes only a few seconds. The same values for K and for the weight are used in this research. The best result is obtained using the seventh and tenth features, with a $K = 7$ and the distance weight. The model trained with these features and parameters reach an accuracy of 94.44%, that is much better than the one obtained with the first two features.
Changing the code, using the number of these features as index in the parameters of the function "shuffleSplitScale()" (line 48, use the number of the feature minus one, because the function starts to count from zero) it is possible to rerun the program so that it plots the boundaries for this pair of feature.

If the dataset had been larger or if combinations consisting of more than two features had been checked, the brute force approach used here might not have been feasible due to the computation times. In that case it would have been necessary to implement a feature selection mechanism to identify in a smart

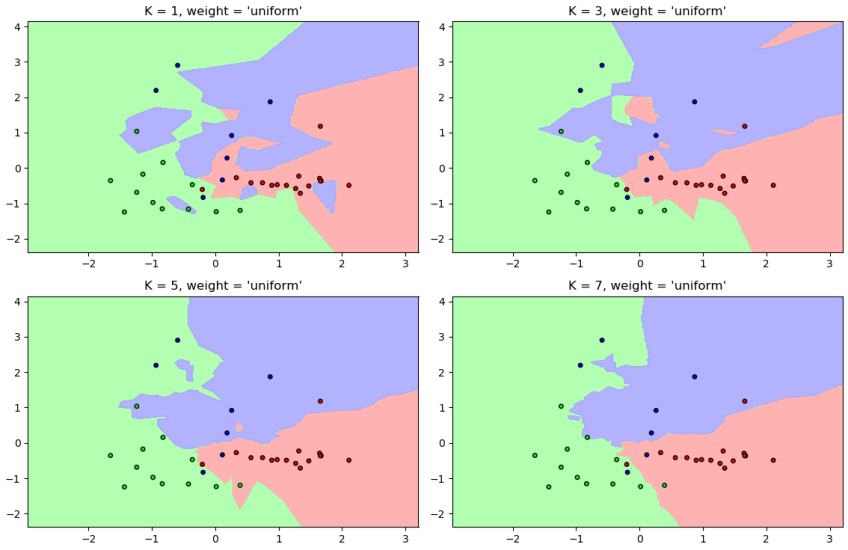

Figure 2: KNN decision boundaries for uniform weight and for all the tested values of K. The represented points belong to the validation set.

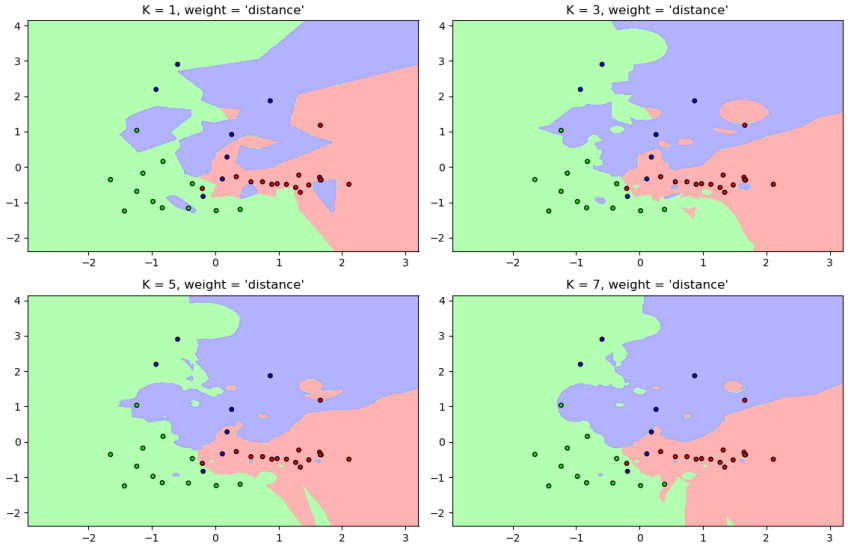

Figure 3: KNN decision boundaries for distance weight and for all the tested values of K. The represented points belong to the validation set.

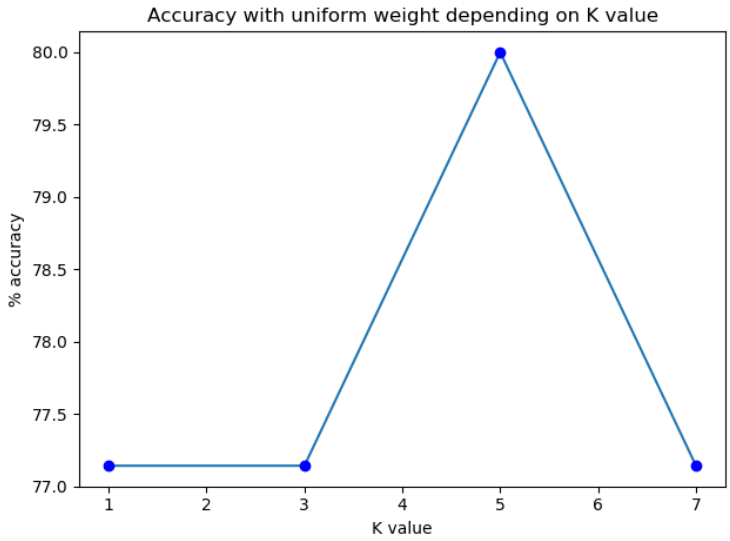

Figure 4: how the accuracy calculated with uniform weight on the validation set varies depending on K value.

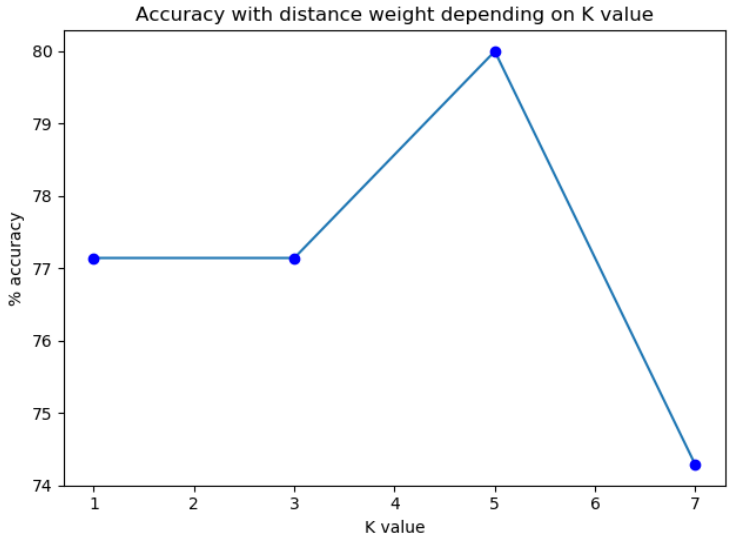

Figure 5: how thw accuracy calculated with distance weight on the validation set varies depending on K value.

way the best combination of features to train the model.

4 Support Vector Machine

The SVM is a supervised learning techniques for classification and regression. Given a training set of samples where each element belongs to one of two different classes, called positive class and negative class, it is possible to build an SVM model which acts as a binary linear classifier, assigning to the new elements one of the two class label. To do this, if the elements are linearly separable, SVMs construct a hyperplane that divides the space in two halfspaces. All the points of the positive class belong to a halfspace and all the points of the negative class belong to the other halfspace. There are many hyperplanes that can divide the elements but the best is the one that represents the largest margin between the two classes. In other words, the best choice is the hyperplane that maximize the distance from it to the nearest element on each side.

In a real case study it is very unlikely that the data are linearly separable but still the two classes are pretty much separated except for a few training samples where the two categories overlap. It is possible to decide to accept some level of error, having some training data on the wrong side of the chosen hyperplane. To do this, positive slack variables are added to the SVM mathematical formulation in order to relax the margin constraint. These variables are weighted by a constant, called C hyperparameter. It says how much to avoid error in the classification of each training samples. If C is set to a high value, SVM will choose a hyperplane with a smaller margin but that will be able to classify all the training samples in a better way. Contrariwise if C is set to a low value SVM will look for a larger margin hyperplane, even if it will make more classification errors. The best value for C hyperparameter depends on the dataset and it must be found during the training phase.

A way to create a non linear classifier able to divide data that are not linearly separable is to apply the "kernel trick": the dot product in the linear classifier is replaced by a non linear kernel function that maps the features in a higer-dimensional space. In this way the separator is a hyperplane in the new feature space and it may not be linear in the original space.
There are several kernels that can be used as an alternative to the linear one. In this study the Radial Basis Function (RBF) kernel will be used. It depends on a parameter, called γ hyperparameter, that can be considered as the spread of the kernel. For high values of γ the curvature of the decision boundaries is greater and SVM is able to capture more of the complexity and shape of the data but there is the risk of incurring overfitting. On the other hand, for low values the boundaries curvature is smaller so the model is less prone to overfitting by avoiding to create "boundaries islands" around training samples, but there is the risk that the boundaries are unable to capture all the complexity of the data.

In this study SVM technique is performed by a Scikit object called "svm". At first, a model is fit using train data with the linear kernel for different values of

the hyperparameter C. Then, the decision boundaries of each model are plotted and the points of validation set are drawn too (figure 6). It is observable that for very low values of C (such as 0.001 and 0.01) the classification is very badly performed since all the elements are classified with the same label.
This happens because a low value of C means that misclassification do not matter a lot and SVM only cares about finding the hyperplane with the larger margin as possible. For the others value of C the boundaries obviously have a linear shape, look pretty good and most of the elements of the validation set belong to the area with the same color as their label. Increasing the value of C from 0.1 to 1000 does not make significant changes to the boundaries.

This procedure is repeated also for the RBF kernel (figure 7). For for very low values of C the classification is not good, like for the linear kernel, and it is not possible to see the shape of the boundaries. With a sufficiently high C value it is possible to notice that the boundaries are no longer straight but have a rounded shape, they are able to classify quite well the validation samples and increasing the value of C they take on an increasingly articulated shape.

In table 2 it is possible to see all the accuracies calculated evaluating the models on the validation set for both kernels. These accuracies are plotted on a graph (figure 8 and 9) to show how they vary when C value changes.

	kernel = linear	kernel = RBF
C = 0.001	34.28%	34.28%
C = 0.01	37.14%	34.28%
C = 0.1	85.14%	80.00%
C = 1	82.86%	82.86%
C = 10	82.86%	82.86%
C = 100	82.86%	82.86%
C = 1000	82.86%	77.14%

Table 2: accuracies for all the tested values of C for both tested kernels. The best value for each kernel is highlighted in red.

For the linear kernel the best accuracy on the validation set is obtained with $C = 0.1$. Fitting the model with this value and using it to evaluate the test set an accuracy of 79.63% is obtained. For the RBF kernel instead the best result is given by more than one value of C. The lower one, that is $C = 1$, is selected and it produces an accuracy of 87.04% on the test set. This kernel obtains a better accuracy on the test set even if the validation one was lower compared to the linear kernel, but both models can be considered good to classify the samples of this dataset.

Then the entire procedure is repeated again for the RBF kernel but this time also tuning the γ hyperparameter. In the previous experiments it was set to its auto default value, that is $1/\#features$ (in this case 0.5). It is tuned on a range of values from 0.0001 to 100 which also includes the default value. The range of C hyperparameter values has also changed: since the accuracies

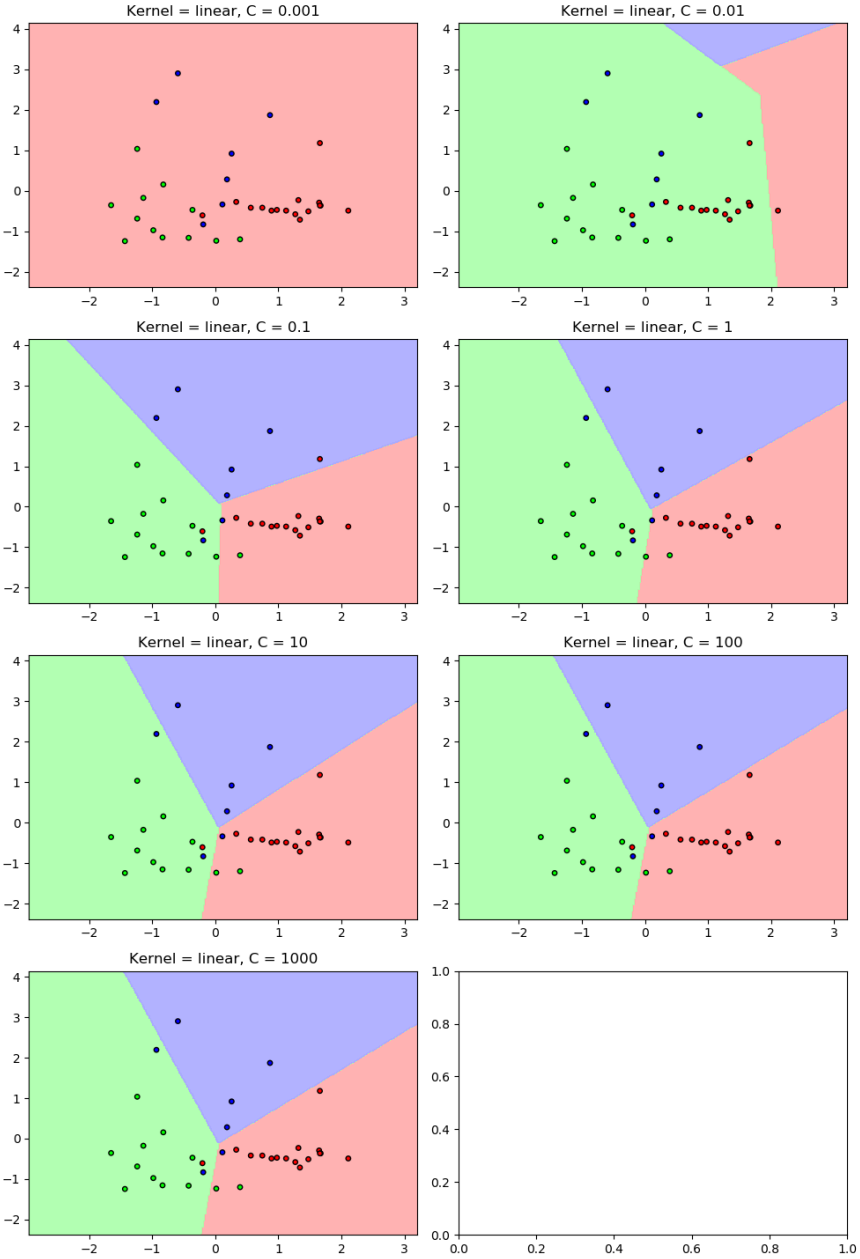

Figure 6: SVM decision boundaries for linear kernel and for all the tested values of C. The represented points belong to the validation set.

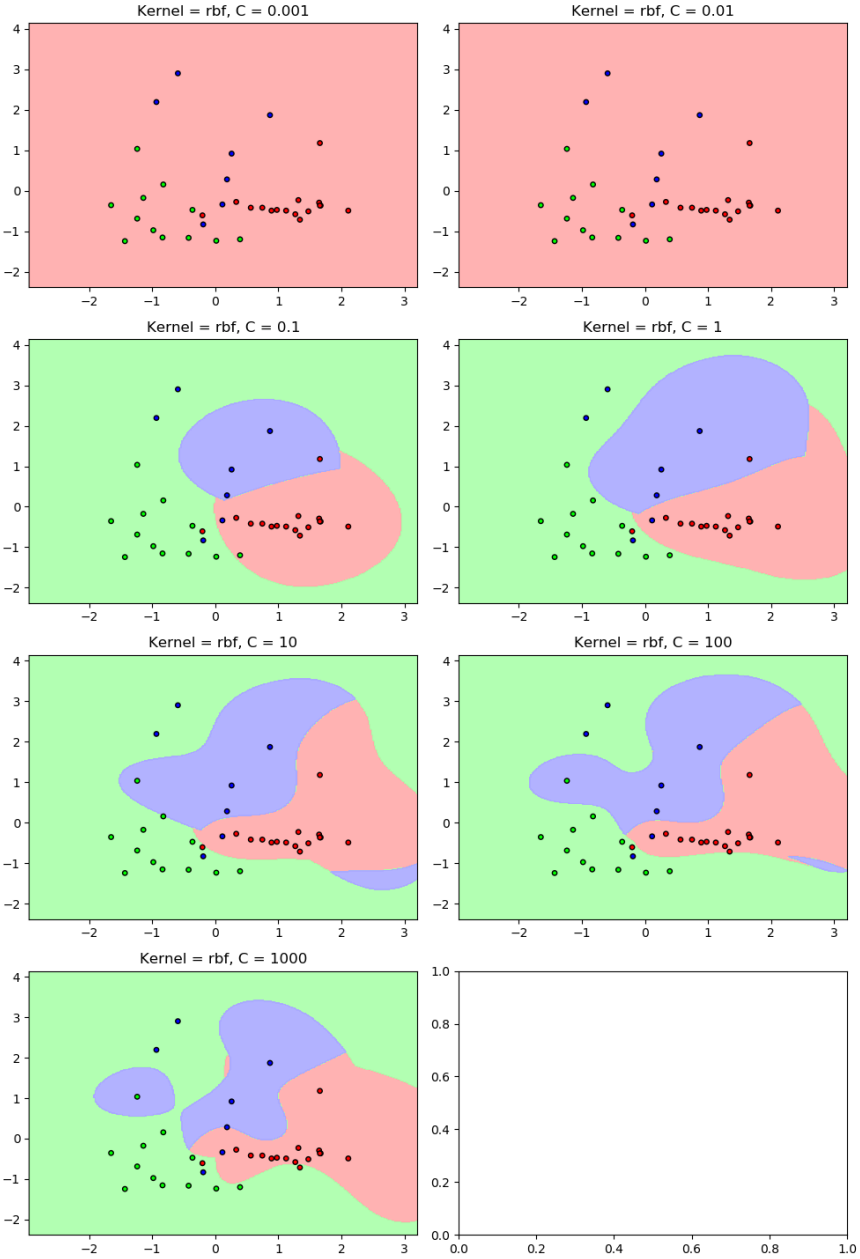

Figure 7: SVM decision boundaries for RBF kernel and for all the tested values of C. The represented points belong to the validation set.

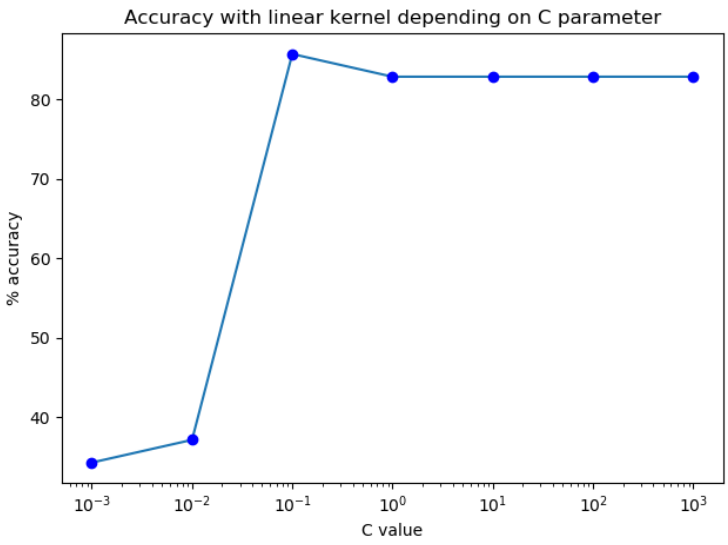

Figure 8: how the accuracy calculated with linear kernel on the validation set varies depending on C value.

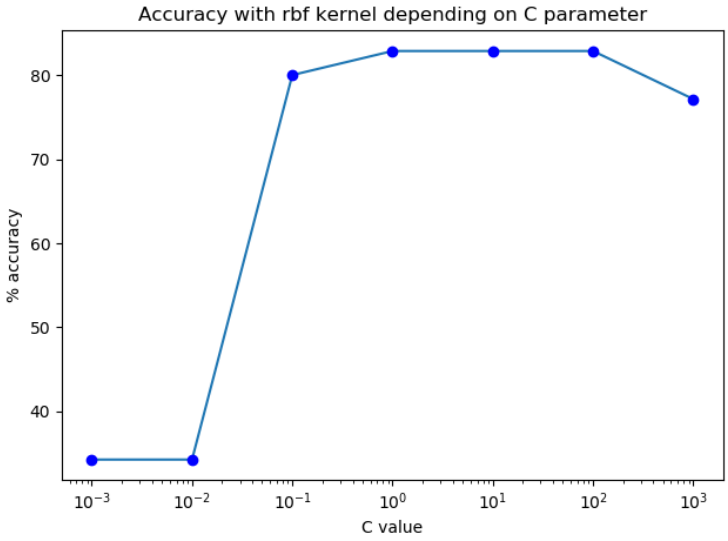

Figure 9: how the accuracy calculated with RBF kernel on the validation set varies depending on C value.

obtained with the lower values of this parameter were not good, those values have been removed from tested range. In table 3 it is possible to see all the accuracies calculated evaluating on the validation set all the models generated for each combination of γ and C values.

	C = 0.1	C = 1	C = 10	C = 100	C = 1000
$\gamma = 0.0001$	34.28%	34.28%	34.28%	77.14%	85.71%
$\gamma = 0.001$	34.28%	34.28%	77.14%	85.71%	82.86%
$\gamma = 0.01$	34.28%	80.00%	85.71%	82.86%	82.86%
$\gamma = 0.1$	65.71%	80.00%	82.86%	82.86%	80.00%
$\gamma = 0.5$	80.00%	82.86%	82.86%	82.86%	77.14%
$\gamma = 1$	82.86%	82.86%	82.86%	77.14%	65.71%
$\gamma = 10$	34.28%	77.14%	62.86%	62.86%	62.86%
$\gamma = 100$	34.28%	51.43%	54.28%	54.28%	54.28%

Table 3: accuracies for all combinations of γ and C values. Best accuracies are highlighted in red.

The program plots the decision boundaries for all of these combinations, but only some examples are reported here (figure 10, 11 and 12). For $C = 0.1$ it is noticeable that only γ values close to the default value avoid to classify all the samples with the same label. This C is probably still to small to be taken as the best value, even if with $\gamma = 1$ it can reach a very good accuracy. For higher values of C it is possible to see how as γ increases the boundaries go from straight to round shape, as expected. Furthermore, for high values of γ (10 or higher) there is no difference in the boundaries between $C = 10$ and $C = 1000$. In both cases the boundaries are made up of many small circular regions in correspondence with the points of the train set. As mentioned before, this is due to the overfitting.

Different combinations produce the best accuracy evaluating the validation set. The one that corresponds to $C = 10$ and $\gamma = 0.01$ is selected to evaluate the test set, whose boundaries are very similar to those produced using the linear kernel (the curvature is very low). The model fitted with this combination reaches a 79.63% accuracy result, that is the same as the one reached using the liner kernel.

Next, train and validation sets are merged together to perform a K-fold Cross Validation on them. The trianing data are divided in K different folds of the same dimension: K-1 of theme are used to fit a model while the remainder fold is used as validation set. This process is repeated K times, each time changing the fold used as validation set, for all the combinations of hyperparameters. The average of the K results is the final score of the model returned by this procedure. Testing more models built with the same combination of hyperparameters on different validation sets makes the performance assessment of the final model built with that combination more stable.

This technique is performed by a Scikit object called "GridSearchCV" using

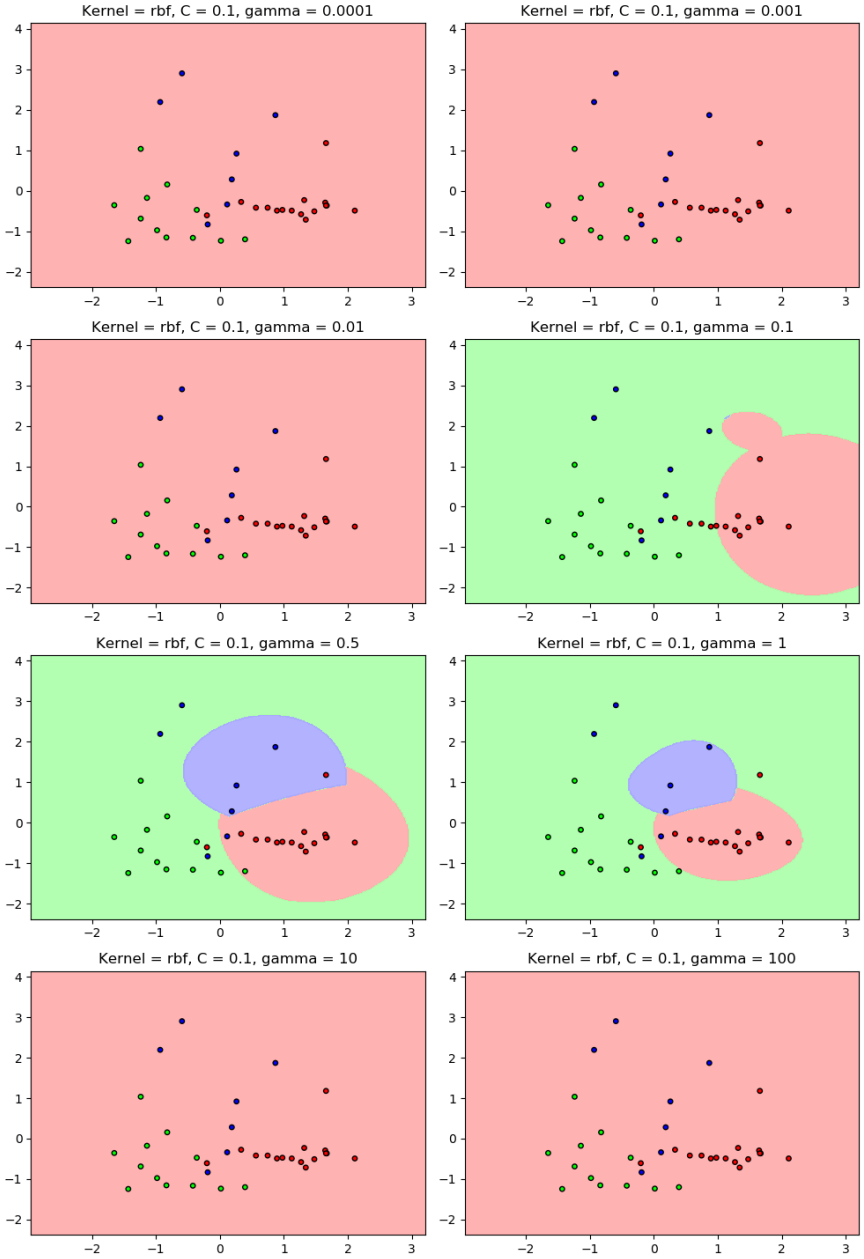

Figure 10: SVM decision boundaries for RBF kernel with C = 0.1 and for all the tested values of γ. The represented points belong to the validation set.

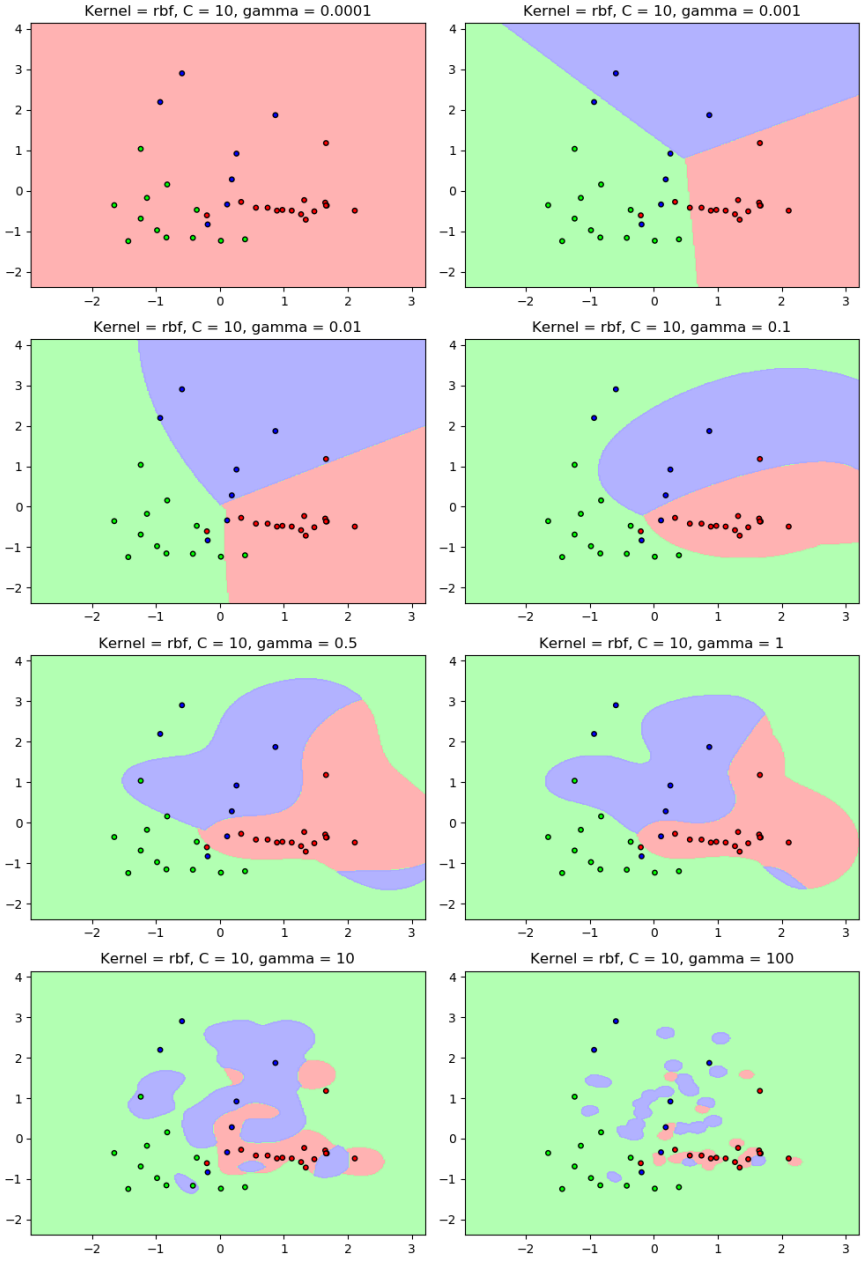

Figure 11: SVM decision boundaries for RBF kernel with C = 10 and for all the tested values of γ. The represented points belong to the validation set.

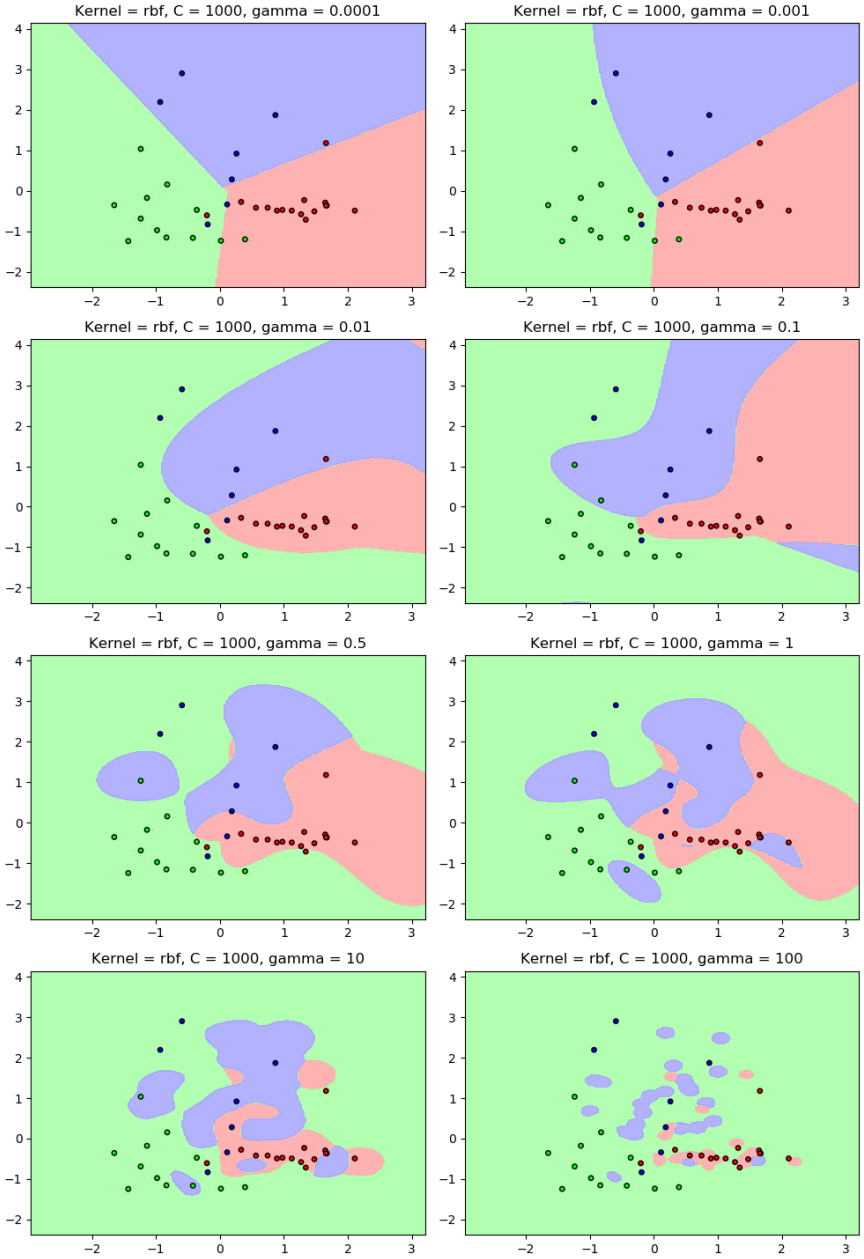

Figure 12: SVM decision boundaries for RBF kernel with C = 1000 and for all the tested values of γ. The represented points belong to the validation set.

$K = 5$. This object tune both C and γ hyperparameters using the same values of the previous experiment and return the best combination after applying the 5-fold Cross Validation. It is also possible to see the accuracies for all the combinations.
The values $C = 1000$ and $\gamma = 0.01$ are returned as best hyperparameters combination, with an accuracy on the validation around 80%, and are used to evaluate the test set. The result is an accuracy of 87.04% which equals the highest found so far.

Eventually, all the features are again considered. Considering that Cross Validation obtained the best result, the research of the best couple of feature is performed repeating only this technique and not the whole experiment. It is possible to try all the pairs of features for the same reasons given in the KNN section. The same values for C and γ tuned in the previous Cross Validation are now uesed in this research. The highest accuracy is obtained using the first and the seventh features, with a $C = 100$ and $\gamma = 0.1$. The model trained with these features and parameters reach an accuracy of 98.15%, which is the highest obtained in this study. It means that almost all samples in the test set are correctly classified. Again, using the number of these features as index in the parameters of the function "shuffleSplitScale()" (line 48, use the number of the feature minus one, because the function starts to count from zero) it is possible to rerun the program so that it repeats all the tests described in this section as well as plots the decision boundaries.

Noteworthy is the fact that, as with the KNN, the best couple of features includes the sixth one. Therefore it may be legitimate to think that this feature is very relevant to classify the samples of this dataset and a feature selection algorithm will most likely recommend the use of this feature.