Machine Learning and Artificial Intelligence
Homework 1

Ilio Di Pietro, s266393

Prof. Tatiana Tommasi, A.Y. 2019/2020

1. Introduction

K-Nearest Neighbors (KNN) and Support Vector Machines (SVMs) are a set of learning methods used for regression and classification (also density estimation in the case of KNN). The aim of this experience is to apply both KNN and SVM to the same dataset, analysing a set of parameters (k in the case of KNN and C, Gamma for SVM) in order to achieve the best accuracy on the validation set and then trying to predict correctly the labels on the test set. At the end, K-fold Cross-Validation will be applied to improve the quality of the classification.

1.1. Homework Sub-tasks

- Train the KNN on the training set show how the accuracy on the validation test varies when changing the regularization parameter K and evaluate the model on the test set. Plot decision boundaries;
- Train a linear SVM on the training set and plot a graph showing how the accuracy on the validation set varies when changing parameter C. Evaluate the best model on the test set;
- Repeat using a RBF kernel. Perform a grid search of the best parameters for a RBF kernel and show how they score on the validation test. Evaluate the best parameters on the test set;
- Repeat the grid search for gamma and C performing 5-fold validation to analyse difference whit the previous case.
- Comparison between KNN and SVM

2. Environment

To perform this tasks we used the Wine dataset. It contains in total 178 samples, divided into 3 classes: in particular 59 of this 178 belong to class0, 71 to class1 and 48 to class2. Each samples is described by 13 features but initially we selected only the first 2 dimension, which are Alcohol and Malic Acid. We randomly split data into train, validation and test sets in proportion 5:2:3. Using the validation set can help to reduce overfitting on the test set. On the other hand, dividing the dataset when the number of samples available for training the model is reduced and the accuracy of the prediction might become really dependant on the random choice of the training and validation sets, so to avoid this problem we used a random split but with a fixed seed(1 in our case). The aim of this choice is also that in this way we will have a random but equal split for KNN and SVM and so we will be able to compare the results obtained. The programming language used to solve this exercises is Python 3.7 and in particular the most important libraries used are:

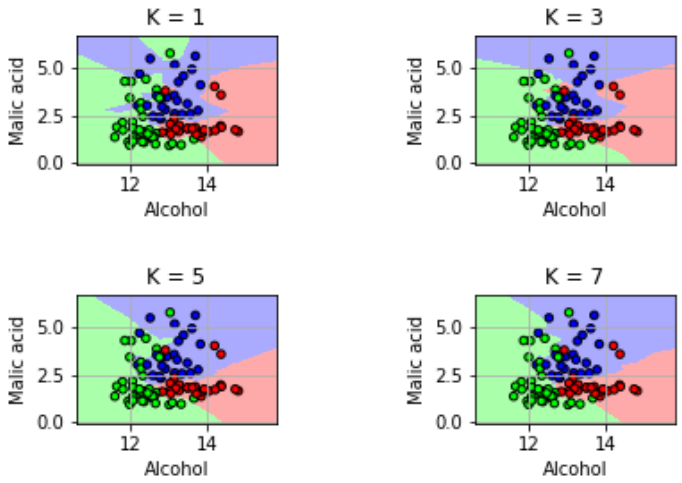
- scikit-learn, a free software machine learning library for Python;
- matplotlib.pyplot: matplotlib is a Python 2D plotting library; pyplot provides a MATLAB- like interface;
- numpy, a package for scientific computing with Python.

3. K-Nearest Neighbors

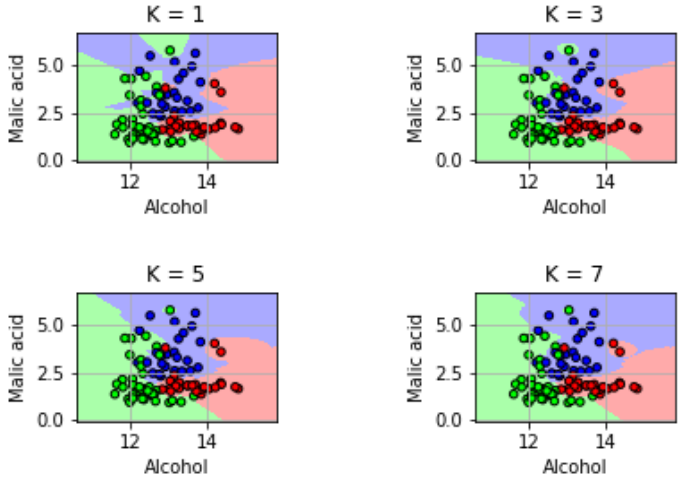
KNN is a multiclass classifier whose purpose is to find a predefined number K of training samples closest in distance to the new point, and predict the label from these. It does not have a learning process to construct a general internal model, but simply stores all the training samples. The strong assumption that has to be done is that closest samples belongs to the same class, so they have to have the same lables. The tuning of K parameter is important because if K is too small, our classifier will be sensible to noise; too high values of K will introduce samples belonging to other class.
Another key factor is the choice of the metric, that influence the computation of the distances between points in the given space(the KNN algorithm is based on the comparison of the distances). In our case the range of K is given: 1, 3, 5, 7. Those values are used to train a KNN classifier on the training set, using the function **KNeighborsClassifier** from **sklearn.neighbors** library. By default it uses as metric for distance computation the **Minkowski** distance, defined as:

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

where d is our dimensionality and p is an integer number. This classifier also use a weight function for the prediction and its default value is **uniform**, which means that all points in each neighborhood are weighted equally. We choose to use also another kind of weight that is **distance**, which weight points by the inverse of their distance. In this case, closer neighbors of a point will have a greater influence than neighbors which are further away. The obtained decision boundaries are shown in Figure[1].We can see that for small values of K the boundaries are more defined. Otherwise, with higher value of K the class assigned is the most probable ones(more robust to noise). The method is evaluated on the validation set and Figure[2] shows how the accuracy varies depending on the value of K, both in the case of 'uniform' and 'distance' weights. The best accuracy on the validation set (0.74) is obtained with K = 5 and weight = 'uniform' and with this parameters we have trained the best KNN to evaluate the test set. Figure[3] shows the decision boundaries on the test set. We obtained an accuracy equal to 0.81.

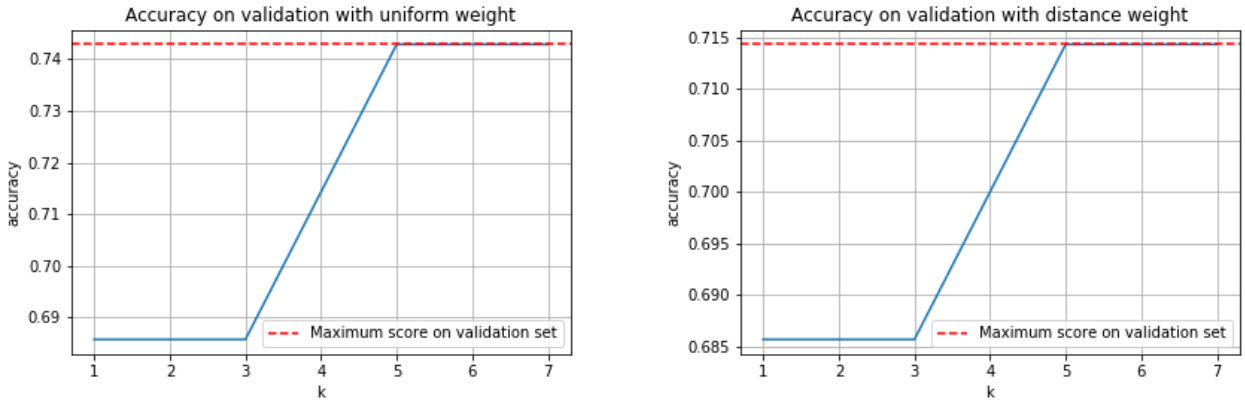


((a)) *Weight = 'uniform'*



((b)) *Weight = 'distance'*

Figure 1: Decision boundaries of KNN on training set for different values of K



((a)) *Weight = 'uniform'*

((b)) *Weight = 'distance'*

Figure 2: Accuracy on validation set as a function of K

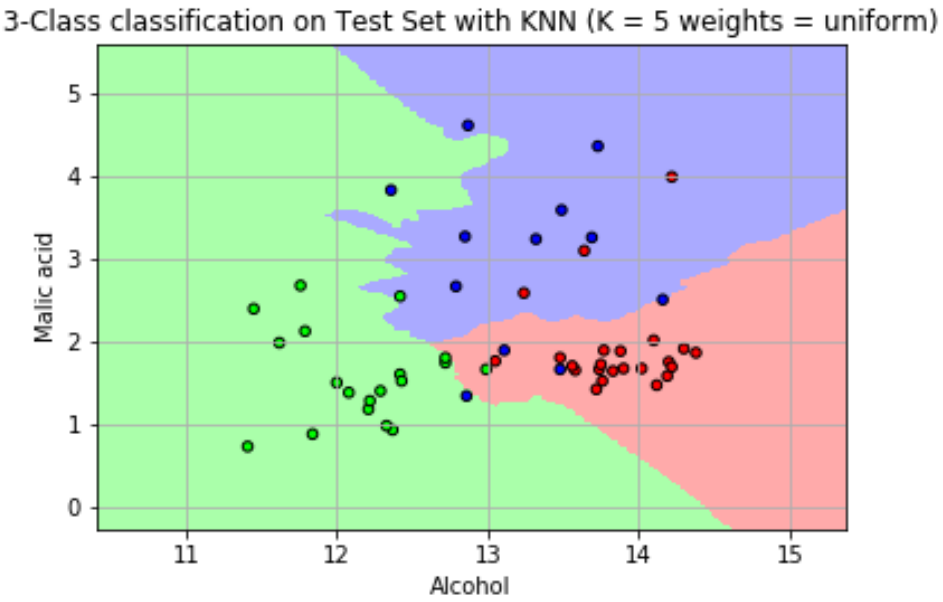


Figure 3: Decision boundaries on test set

4. Linear SVM

As we said, Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. Given two data points x and x' , the linear kernel function k is defined as:

$$k(x, x') =< x, x' >$$

Given labeled training data, the algorithm outputs an optimal hyperplane (defined as the hyperplane that has the largest distance to the nearest training data points of any class) which categorizes new examples. We are interested in the largest margin because it give us maximum robustness relative to uncertainty and independence from correctly classified instances. We tuned the parameter C to to regularize the amount of misclassified training points allowed in the model. For large values of C, a smaller-margin hyperplane will be chosen if that hyperplane gets all the training points classified correctly. Too high values of C might cause overfitting. Conversely, for a very small value of C, the optimizer will look for a larger-margin separating hyperplane, therefore for a simpler decision function, even if that choice brings to heavy misclassification. In our case we have $C = [10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3]$.
Those values are used to train a linear SVM on the training set. The obtained decision boundaries are shown in Figure[4]. We can see how the smallest values of C bring to more fleeting boundaries (i.e., for C = 0.001, all the training points belong to the same decision surface); the more C grows the more accurate the boundaries and the classification get, but it could present an overfitting behaviour. The method is then evaluated on the validation set and Figure[5] shows how the accuracy depends on C. The best accuracy on validation set (0.77) is obtained whit C = 0.1 (also with C = 10, 100, 1000 we obtained an equal accuracy but at this point our algorithm choose the first value). The best value of C is now used to evaluate the model on the test set. The computed decision boundaries are reported in Figure[6]. The obtained accuracy is equal to 0.83

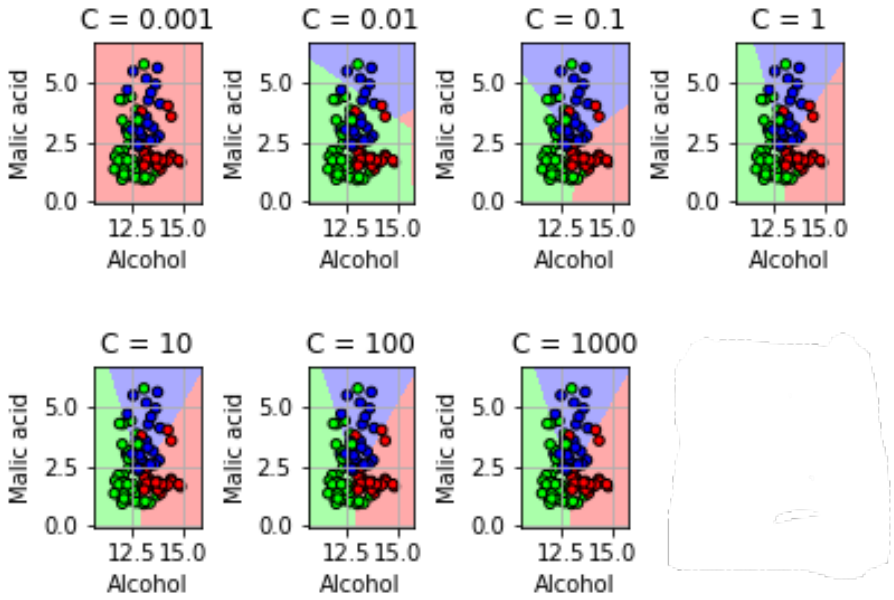


Figure 4: Decision boundaries on training set for different values of C (linear SVM)

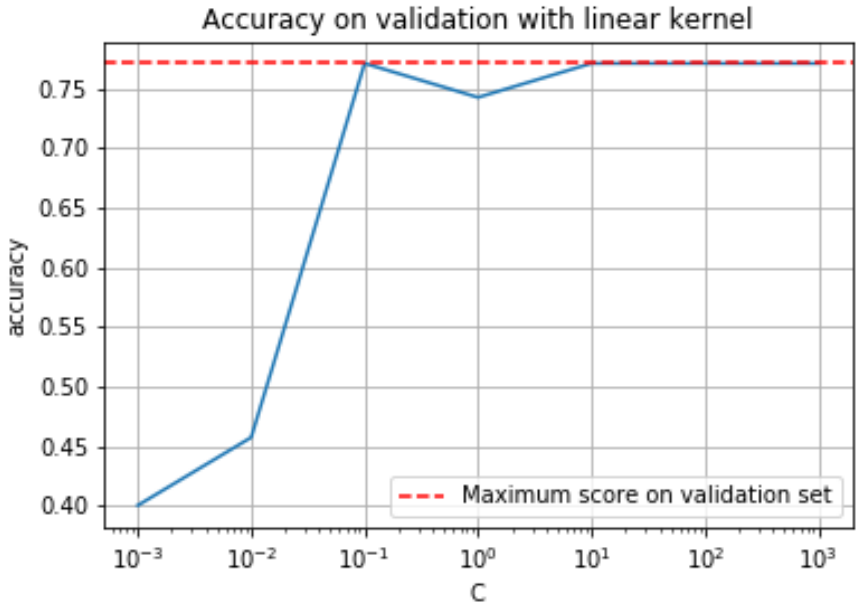


Figure 5: Accuracy on validation set in function of C (linear SVM)

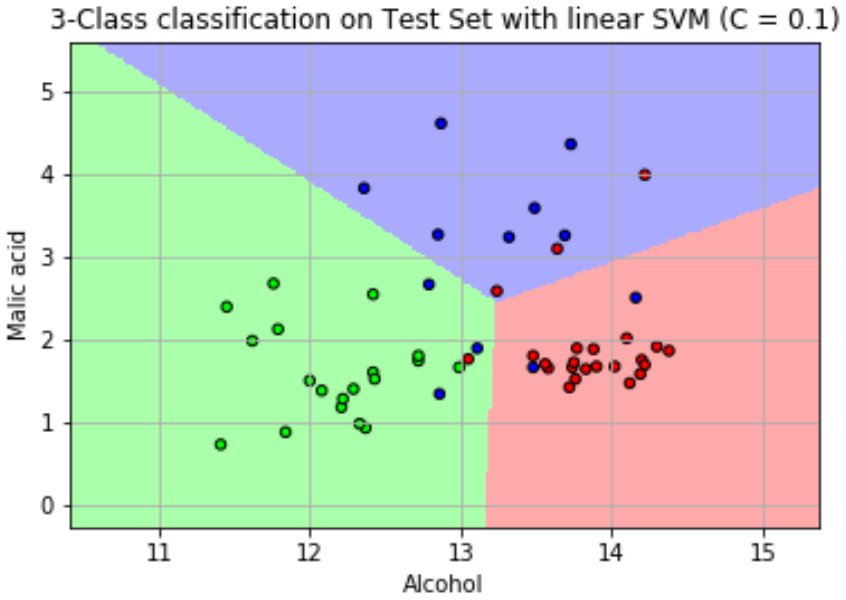


Figure 6: Decision boundaries on test set (linear SVM)

5. RBF Kernel

Given two data points x and x' , the Radial Basis Function (RBF) kernel SVM k is defined as:

$$k(x, x') = \exp\left(-\gamma \|x - x'\|^2\right)$$

where γ must be greater than 0 and represents the inverse of the radius of the area of influence of samples selected by the model as support vectors, and $\|x - x'\|^2$ is the squared Euclidean distance between two data points x and x. Kernels in SVM classification refer to the function that is responsible for defining the decision boundaries between the classes. Apart from the classic linear kernel which assumes that the different classes are separated by a straight line, a RBF kernel is used when the boundaries are hypothesized to be curve-shaped. Also in this case the values of the regularization parameter C remains the same ($C = [10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3]$) and we set γ to 'auto' for the first computation (the algorithm choose the best value). Figure[7] shows the decision boundaries on training set and we can immediately see the difference compared to the linear case, and it is the shape: with RBF kernel, the decision regions tend to cover the spread of the data. Figure[8] represent how accuracy change in function of C on validation set and its best value is 0.77 and it is reached for C = 0.1 . The accuracy on the test set improves compared to the one resulted using a linear kernel: its value is 0.85 and Figure[9] shows the decision boundaries.

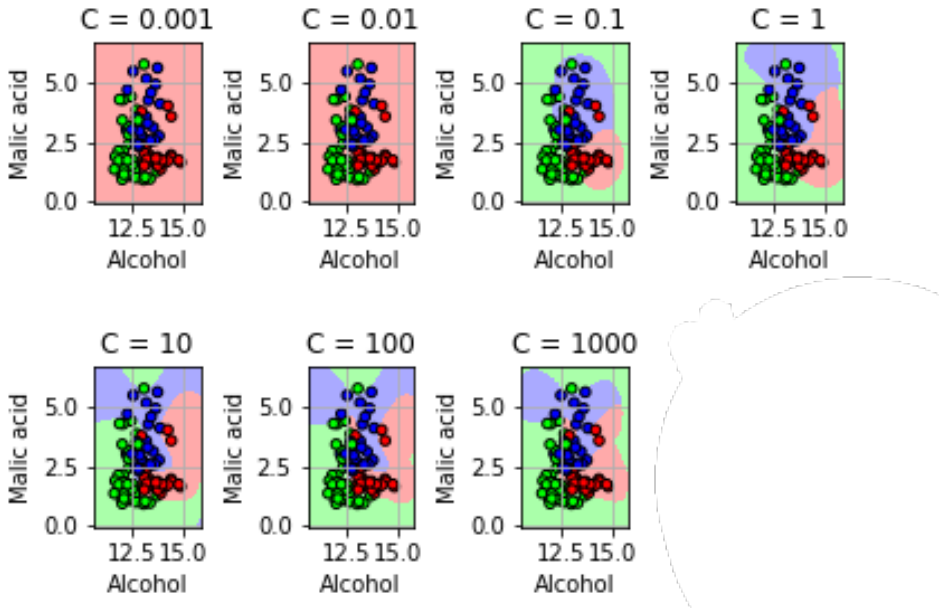


Figure 7: Decision boundaries on training set for different values of C (RBF kernel SVM)

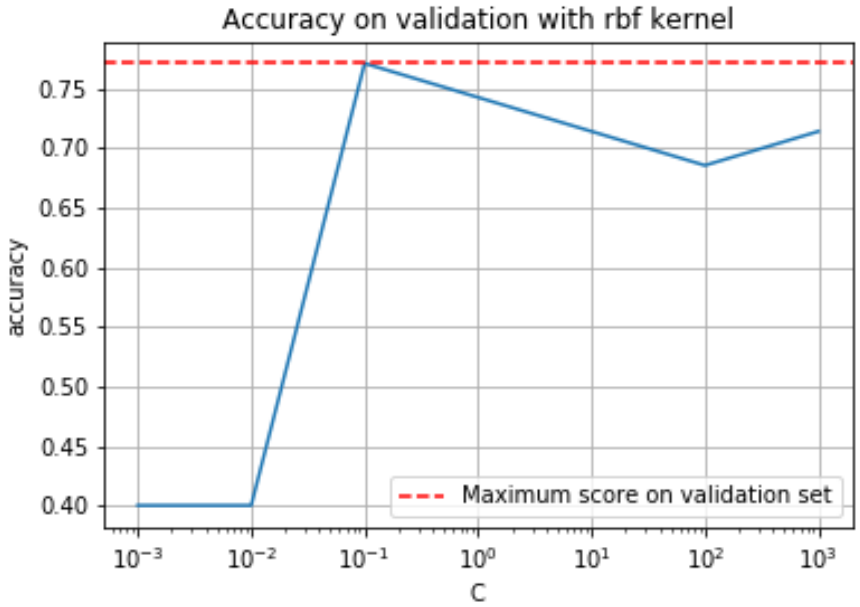


Figure 8: Accuracy on validation set in function of C (RBF kernel SVM)

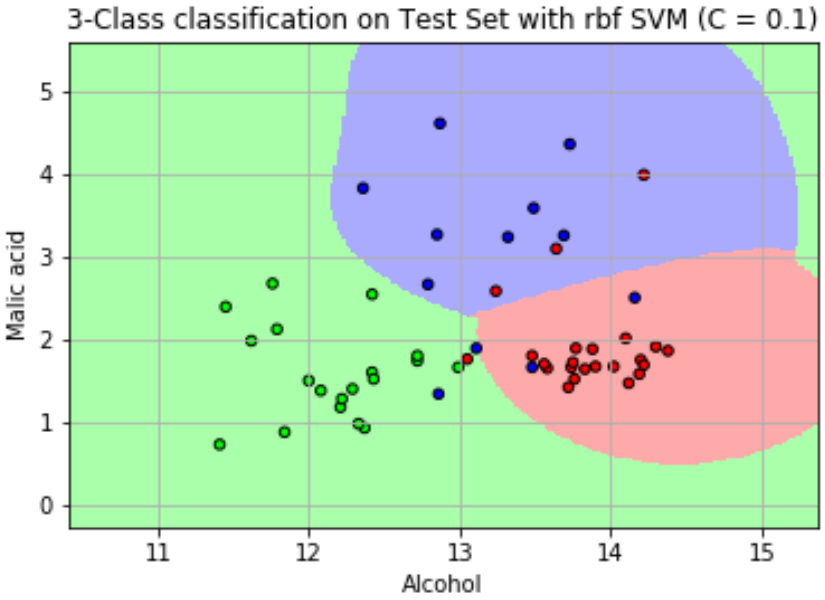


Figure 9: Decision boundaries on test set (RBF kernel SVM)

5.1. Grid Search

We are interested now in the research of the best value of C and γ so we perform a grid search in order to tune both parameters. the parameters are chosen in this way:
$C = [10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3]$
$\gamma = [10^{-7}, 10^{-5}, 10^{-3}, 10^{-1}, 0.5, 1, 10, 10^2]$.
These values are used to train our model. The value of γ affects a lot the model. In fact, if γ is too large, the region of influence will only include the support vector itself and we will have overfitting; if we choose a value that is too small the area of influence of each support vector will include the whole training set and so the model will be too generalized. Figure[10] reports a heat map to highlight the distribution of the scores obtained and the best value is 0.77, reached by the parameters C = 0.1 and $\gamma = 0.5$ .Figure[11] shows the model applied on the training set whit this best parameters. We used these values to apply the model on the test set(see Figure[12]) and the accuracy obtained is 0.85.

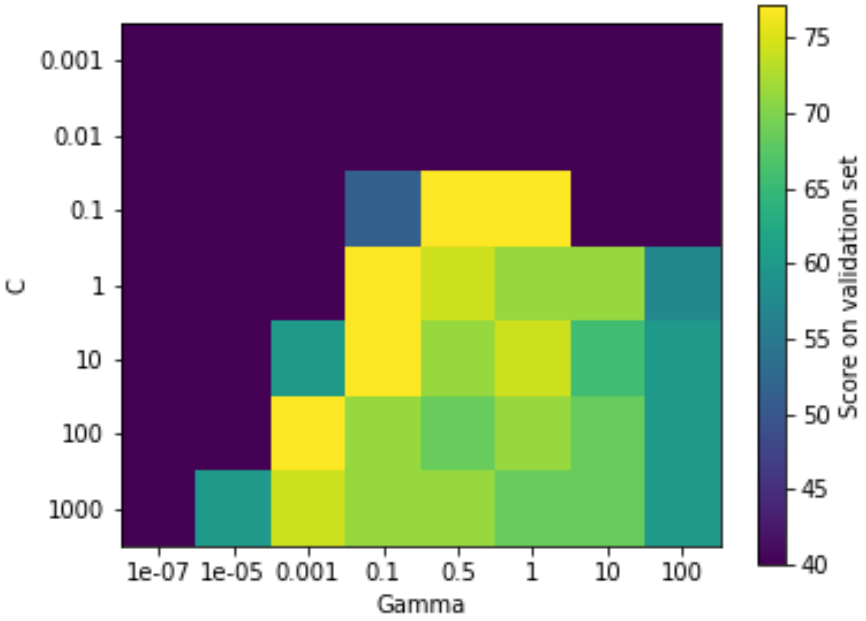


Figure 10: distribution of scores on validation set. (C, γ grid search RBF kernel SVM)

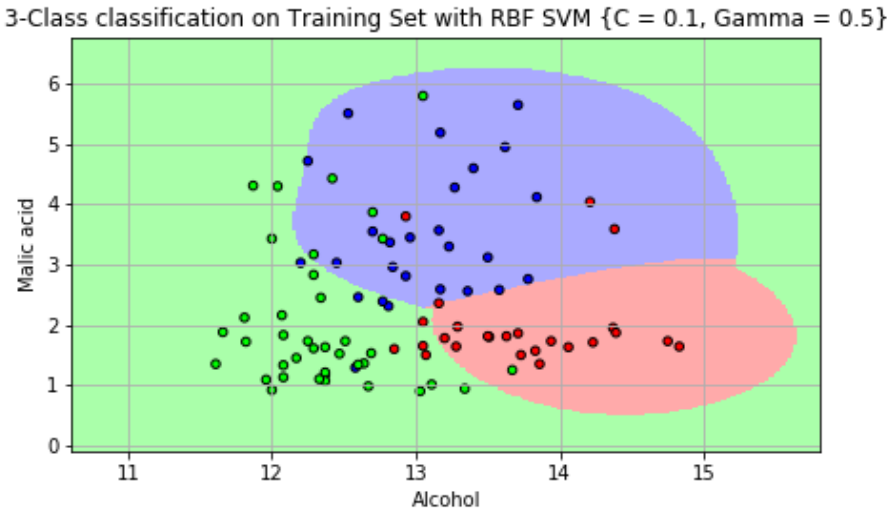


Figure 11: Decision boundaries on training set whit best C and γ (C, γ grid search RBF kernel SVM)

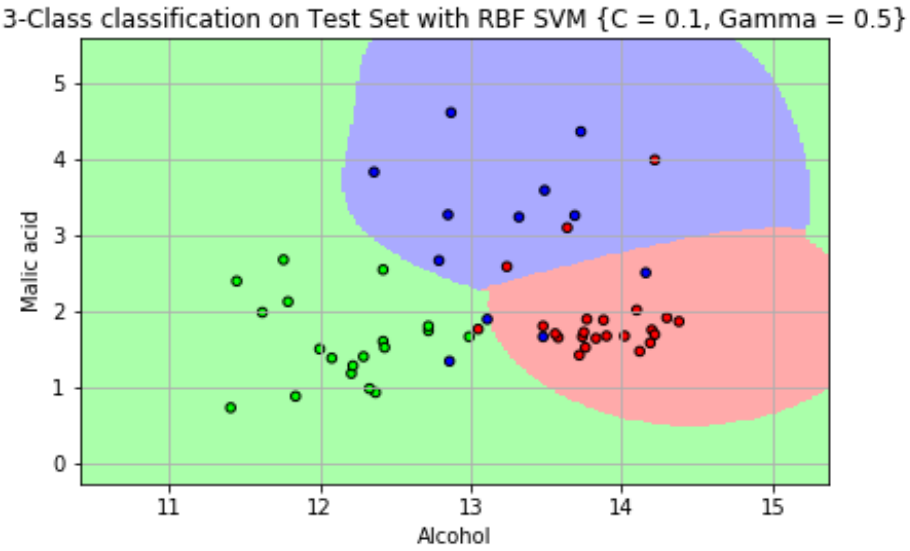


Figure 12: Decision boundaries on test set (C, γ grid search RBF kernel SVM)

6. K-Fold Cross-Validation

The aim of K-Fold Cross-Validation (CV) is to improve the precision of our model solving the problems related to the split of the initial data into training and validation set. Removing a part of training data for validation poses a problem of underfitting. By reducing the training data, we risk to lose important patterns in data set, which in turn increases error induced by bias. Furthermore, the use of the validation set basically improves the generalization performance of the model, but the results depend on the random chosen partition, especially if a small dataset like Wine ones that contains only 178 samples. In K Fold cross validation, all the data that we have are divided into k equal sized subsets. The training part now consist in train our model for k times, each time considering one of the k subset as validation set and the remaining k-1 as training set and computing the accuracy.The final estimation is obtained computing the average of the k accuracies. In our case we merged training and validation set to obtain 70% training and 30% test data and we performed the same grid search computed before (tuning of C and γ) but now with a 5-fold CV. In this way for every combination of C and γ we computed the accuracy as the average of the scores obtained by the 5 iteration of the 5-fold CV (see Figure[13]). For this task we used the **GridSearchCV** function that find the optimal parameters performing the grid search and the cross validation (the default behavior of this function is to apply a stratified cross validation so we used a specific argument to specify the k-fold cross validation). In this case the best parameters are C = 1000 and $\gamma = 0.1$ with an accuracy equal to 0.86, the best value reached until now. Also in this case we applied the best model to the test set and Figure[14] shows it: the accuracy is 0.81.These results are even more accurate than the previous case because both the values of accuracy(validation and test) are much more consistent.

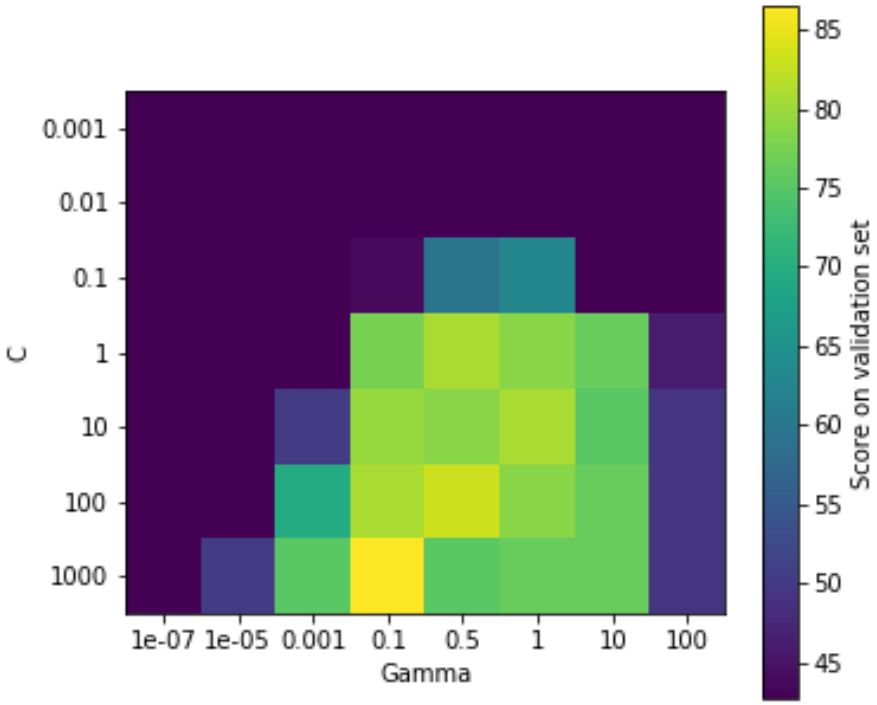


Figure 13: distribution of scores on validation set (C, γ grid search 5 fold cross validation RBF kernel SVM)

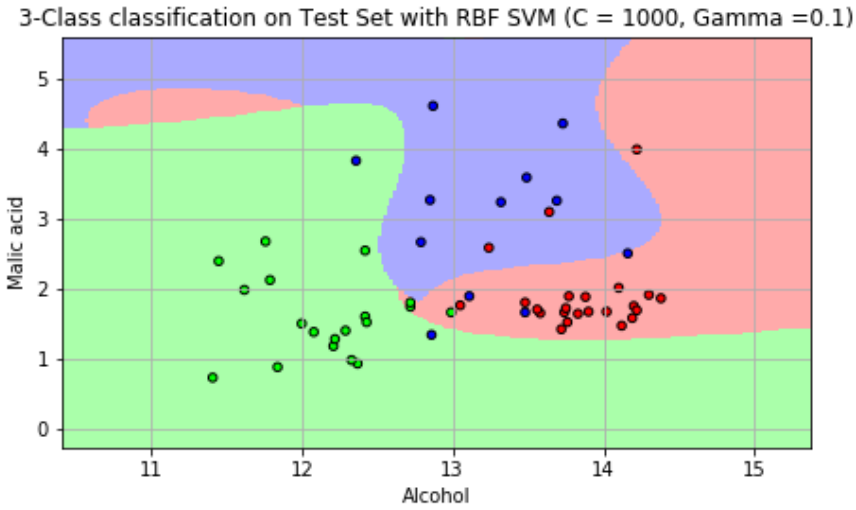


Figure 14: Decision boundaries on test set (C, γ grid search RBF 5 fold cross validation kernel SVM)

7. Extra

7.1. Differences between KNN and SVM

KNN and SVM are two of the most used algorithms for supervised learning, which means that it is possible the classification of new data having access to a set of labeled training data.
Main differences:

K-Nearest Neighbors

- Assumption: close sample belongs to the same class, so they have the same labels;
- No learning process: stores all the training samples;
- Memory-consuming algorithm: compute all the distances and save the results;
- Good for multiclass classification;
- Robust: it depends only on K and the metric used for computing distances;
- Scales badly with the number of samples: not the best choice for high dimensionality problem or when the number of samples became huge;
- generalization guarantee: when the number of samples goes to ∞, the theoretical error remains lower than twice the bias error (the optimal ones);

Support Vector Machines

- Assumption: data linearly separable;
- Learning process: find all the support vectors;
- Memory-saving algorithm: we have to memorize only the support vectors;
- Not good for multiclass classification: other approach are used in this case (One-vs-one, One-vs-All);
- SVM depend on C for linear problem and on kernel parameters for non linear problems;
- Possibility to use kernel function for non linear problems;