POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Data Science and Engineering

Machine Learning and Deep Learning Report
Exam session: Winter 2021

KNN versus SVM: application and comparison on the wine dataset

Emanuele Fasce

1. INTRODUCTION

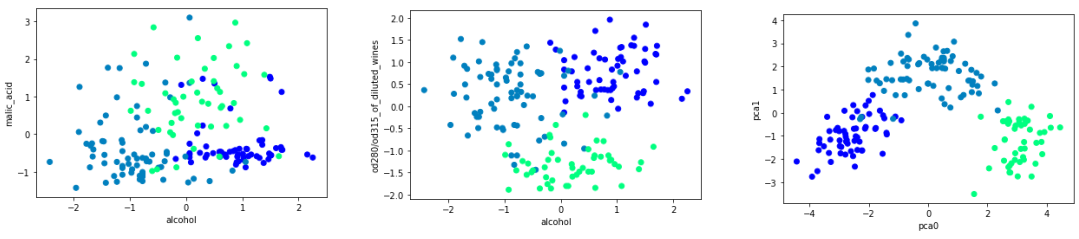
In this project I have applied a K-Nearest-Neighbors (KNN), a linear Support Vector Machine (SVM) and a Radius Basis Function kernel SVM (RBF) algorithms on the popular wine dataset available on the Scikit-learn library in Python. The task is to classify three wines considering only two features out of thirteen.
The decision boundaries are visualized to better understand the behavior of these algorithms as hyperparameters change, and the best combination of attributes and algorithms are evaluated according to the accuracy score.

2. DATASET ANALYSIS AND PREPROCESSING

This dataset contains 13 continuous attributes related to wine characteristics and 3 wine labels. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars.

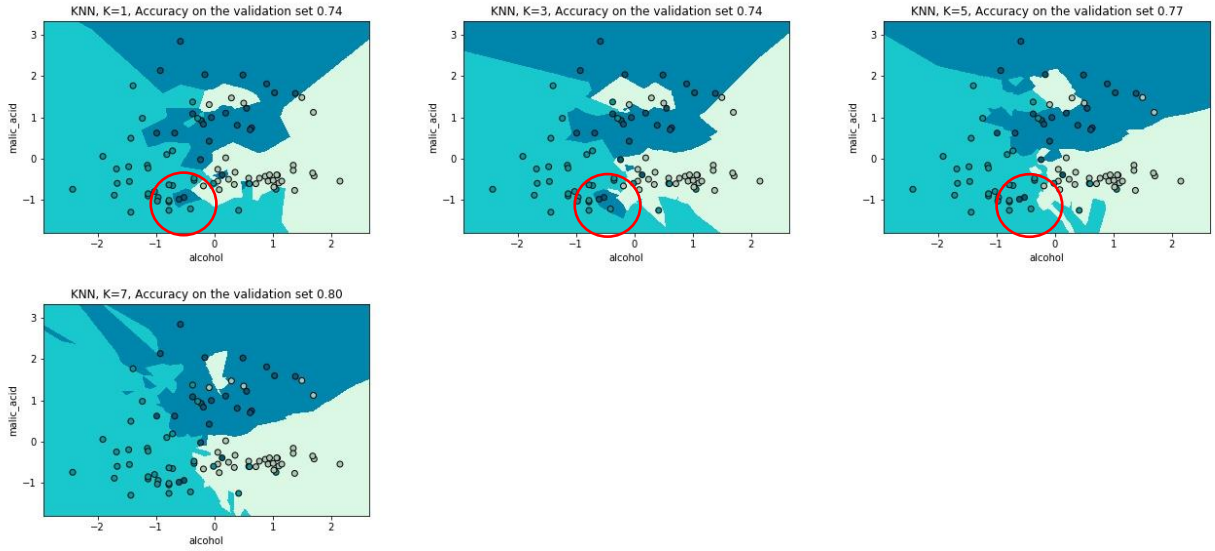
alcohol	malic_acid	ash	alcalinity	magnesium	total_phenols	flavanoids	nonflavanoids	proanthocyanins	intensity	hue	od280/od315	proline
14.23	1.71	2.43	15.6	127.0	2.8	3.06	0.28	2.29	5.64	1.04	3.92	1065.0
13.2	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.4	1050.0
13.16	2.36	2.67	18.6	101.0	2.8	3.24	0.3	2.81	5.68	1.03	3.17	1185.0
14.37	1.95	2.5	16.8	113.0	3.85	3.49	0.24	2.18	7.8	0.86	3.45	1480.0
13.24	2.59	2.87	21.0	118.0	2.8	2.69	0.39	1.82	4.32	1.04	2.93	735.0
14.2	1.76	2.45	15.2	112.0	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450.0
14.39	1.87	2.45	14.6	96.0	2.5	2.52	0.3	1.98	5.25	1.02	3.58	1290.0
14.06	2.15	2.61	17.6	121.0	2.6	2.51	0.31	1.25	5.05	1.06	3.58	1295.0
14.83	1.64	2.17	14.0	97.0	2.8	2.98	0.29	1.98	5.2	1.08	2.85	1045.0
13.86	1.35	2.27	16.0	98.0	2.98	3.15	0.22	1.85	7.22	1.01	3.55	1045.0
14.1	2.16	2.3	18.0	105.0	2.95	3.32	0.22	2.38	5.75	1.25	3.17	1510.0
14.12	1.48	2.32	16.8	95.0	2.2	2.43	0.26	1.57	5.0	1.17	2.82	1280.0
13.75	1.73	2.41	16.0	89.0	2.6	2.76	0.29	1.81	5.6	1.15	2.9	1320.0
14.75	1.73	2.39	11.4	91.0	3.1	3.69	0.43	2.81	5.4	1.25	2.73	1150.0
14.38	1.87	2.38	12.0	102.0	3.3	3.64	0.29	2.96	7.5	1.2	3.0	1547.0
13.63	1.81	2.7	17.2	112.0	2.85	2.91	0.3	1.46	7.3	1.28	2.88	1310.0
14.3	1.92	2.72	20.0	120.0	2.8	3.14	0.33	1.97	6.2	1.07	2.65	1280.0

The features have been normalized using the Z-score. When using Euclidean distance as in this case, normalization allows to give the same weight to the features even if they have different ranges of values. Since only 2 attributes out of 13 are used to classify the wines, a prior analysis could be helpful in order to choose the best pair of attributes to be used.
When considering which attributes could separate data more effectively to help the analysis, building two new dimensions with LDA could be a reasonable choice, however in the dataset the normality assumption does not hold for all the features.
Consequently, three couples of features that will be considered in the analysis are plotted below after applying normalization: from left to right, two random features, two features with a low correlation coefficient, two features built with PCA. To choose the best hyperparameters for the algorithms and evaluate them, the dataset is splitted into training, validation and test set in proportion 5:2:3.

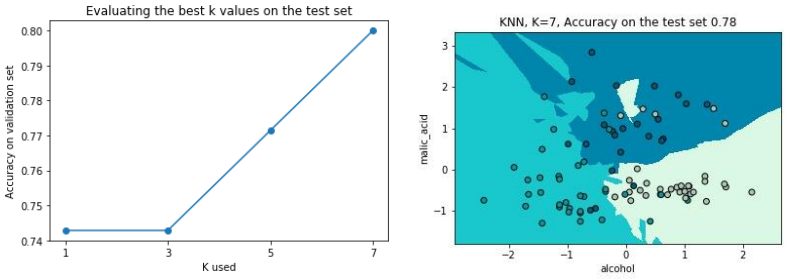


3. K-NEAREST NEIGHBORS CLASSIFICATION

The K-Nearest Neighbors algorithm is a non-parametric method used for classification and regression. The main idea of this algorithm is that similar things exist in close proximity.
In this algorithm, for each sample in the test set, the training samples are ordered according to the distance to it, and the closest k samples are considered.
In the classification case, the mode of the k labels is returned as output, in the regression case the mean.
The accuracies for the different values for k are evaluated on the validation set (in this report k could assume the values 1,3,5,7), and the value for k which gives the best accuracy is used to test the algorithm on the test set. The decision boundaries are plotted below.



From these plots it is interesting to notice how decision boundaries change with K.
As K increases, decision regions become less and less isolated and tend to form more homogeneously separated regions. This behavior could be noticed by looking at the highlighted blue region which disappears as K=5. Three near samples indeed are no more enough to form a decision region.
Since K=7 reaches the best accuracy on the validation set, it will be the hyperparameter used also on the test set. The graph on the left shows the accuracy for each value of K, while the graph on the right represents the points belonging to the test set and the decision boundaries built on the training set with K=7.



The KNN method is very intuitive, requires only a single hyperparameter, can learn non-linear decision boundaries and does not have an explicit training phase.
On the other hand, the prediction complexity increases dramatically as the numbers of features and training samples increase.

4. LINEAR SVM CLASSIFICATION

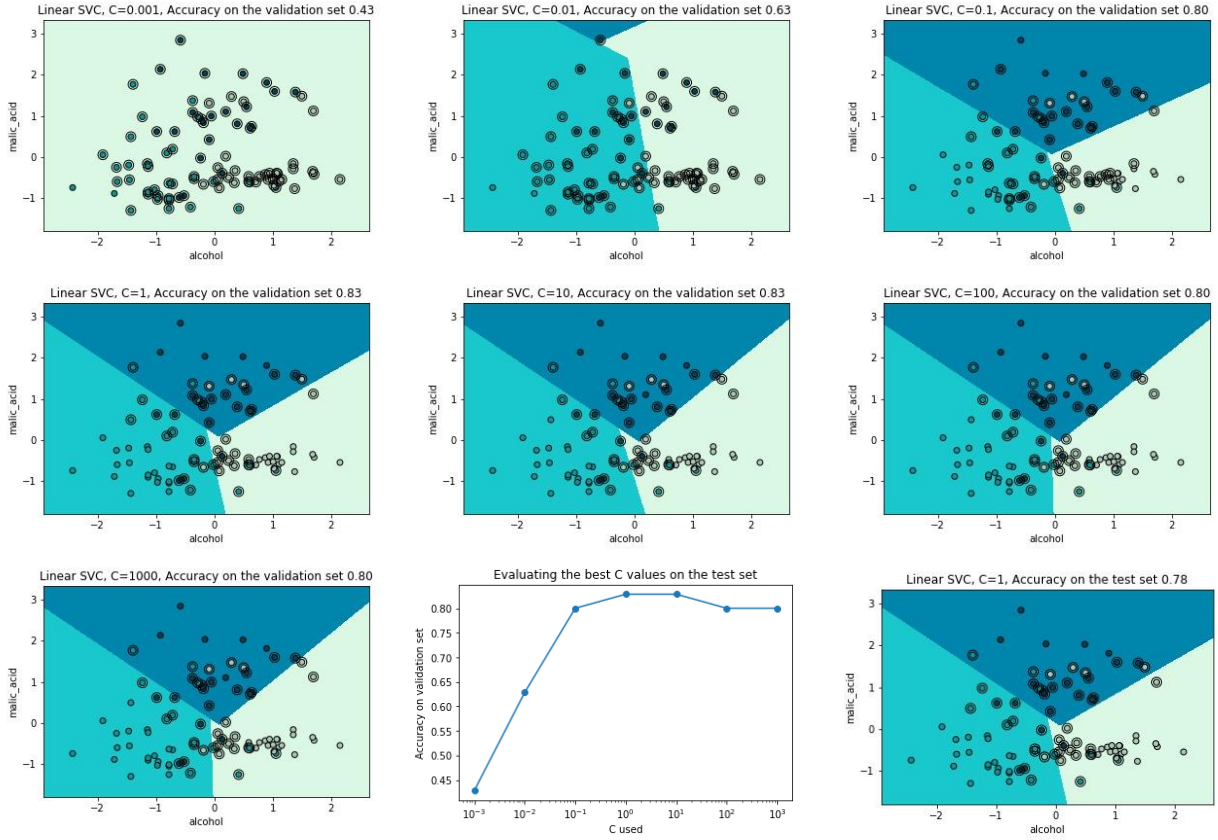
The Support Vector Machines (SVMs) algorithms are supervised models used for classification and regression. The objective of the support vector machine algorithm is to find the hyperplane that best separates the data. It does so by maximizing the margin, the minimum distance between points of different classes, so that future data can be predicted with more confidence.

In formulas, an hyperplane can be written as $< w, x > + b = 0$
and our goal can be written as:
$argmax(min_{i \in M}(< w, x_i > + b)) \ \mid \ y_i(< w, x_i > + b)) > 0$

This is the hard-margin formulation that can be applied when data are linearly separable.
However, data are not always linearly separable, and for this reason a parameter C that accounts for the slack variables is introduced. In the formula it is indicated with the Greek letter ξ.

$$argmin(\lambda||w||^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i) \ \mid \ y_i(< w, x_i > + b)) >= 1 - \xi_i$$

The hyperparameter C is fundamental when training a SVM: it trades off correct classification of training examples against maximization of the decision function's margin. For larger values of C, a smaller margin will be accepted if the decision function is better at classifying all training points correctly. A lower C will encourage a larger margin, therefore a simpler decision function, at the cost of training accuracy. However, if C is too low the classification algorithm sacrifices too much the accuracy, recognizing only one class (C=0.001). The decision regions for different values of C are visualized and the value that performs better on the validation set is used to fit the final model on the test set.

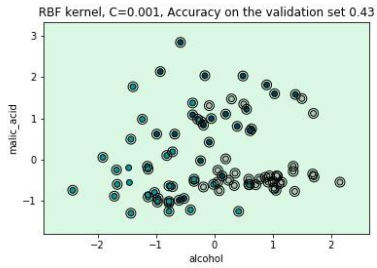


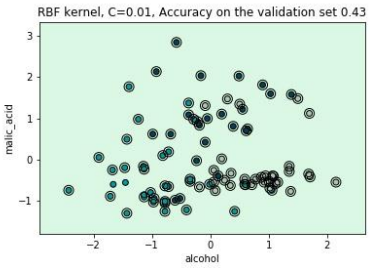
5.RBF KERNEL CLASSIFICATION

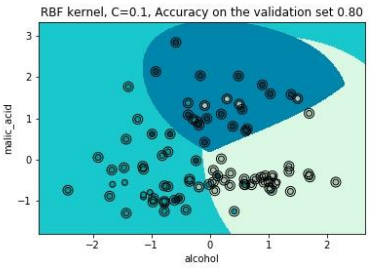
In the non-linearly separable case, the goal is finding a mapping $\psi : X \to F$ that maps data onto a higher dimensional space where data are linearly separable. The problem is that finding and using a proper mapping is a complex and computationally expensive operation. In this context the kernel trick is exploited. Kernels are cheap function that define the dot product in the transformed space, so that the explicit mapping can be avoided.
There are a lot of available kernels, but in this report only the Radial Basis Function kernel is used.

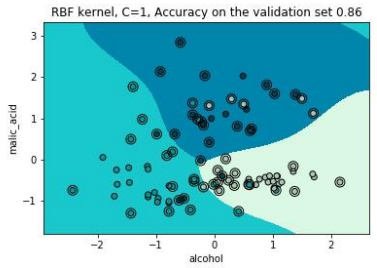
$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

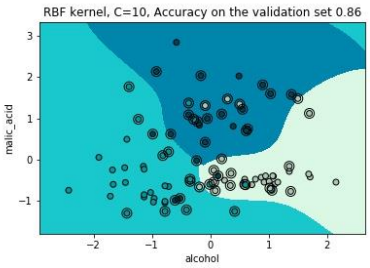
Since this exploitation of the Kernel trick needs also the parameter γ, the validation procedure will be executed to find the best γ and C values.
These are the decision boundaries fixing γ as default and allowing C taking different values. However, the effect of C will be more clear later when keeping γ fixed.

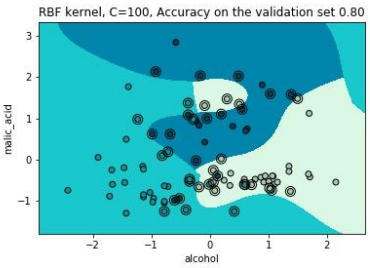


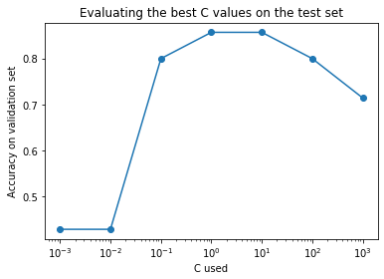


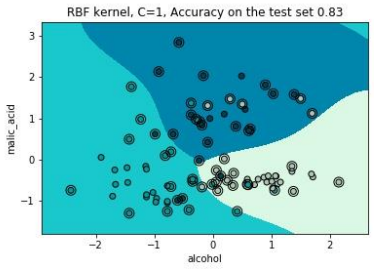




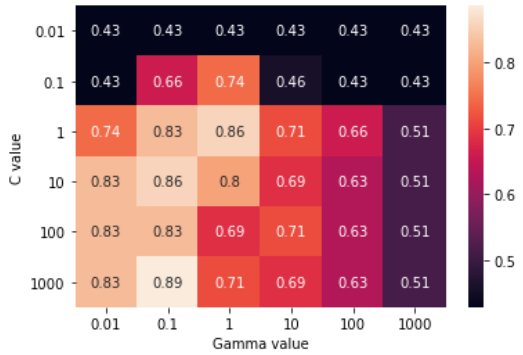


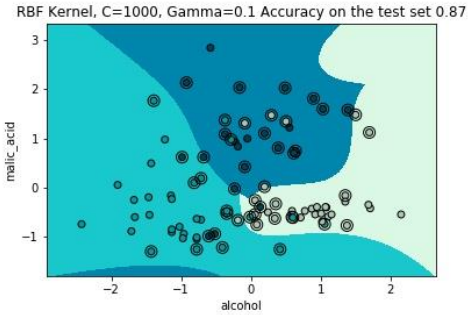




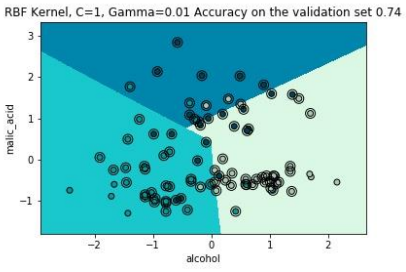


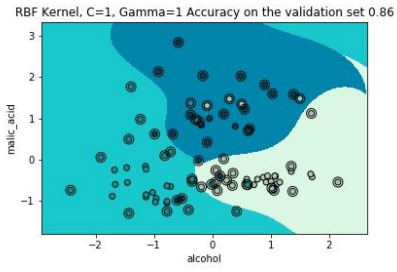
A grid-search is now performed to find the best values of both the parameters C and γ on the validation set.
It is a trial and error approach: the algorithm is evaluated on the validation set with every combination of sets of reasonable values for γ and C.
The hyperparameters that perform better are chosen to train the final model on the test set.
In this case, the values that work better on the validation set are C=1000 and γ=0.1, reaching an accuracy of 0.87 on the test set.

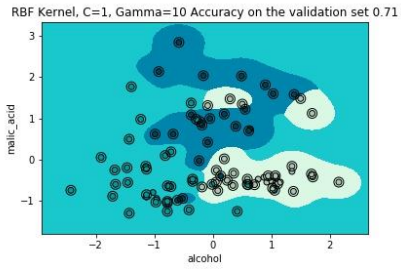


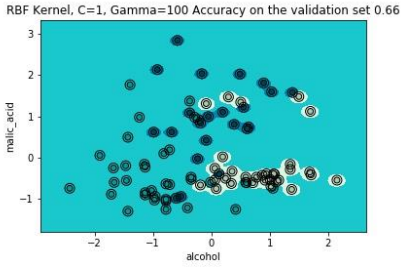


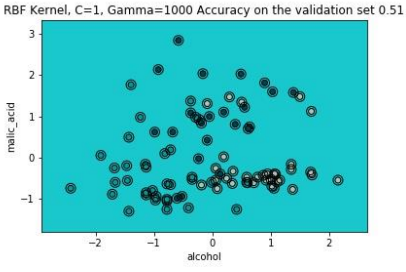
In order to see how the boundary regions change when γ changes and C is kept fixed, here are the plots for C=1. Intuitively, the γ parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. From these plots this explanation seems to be clear, since as γ increases the influence of training samples for two classes (the darker and the brighter class) gets so low that the decision boundaries tend to surround only the training samples, like in the plot for γ=100.



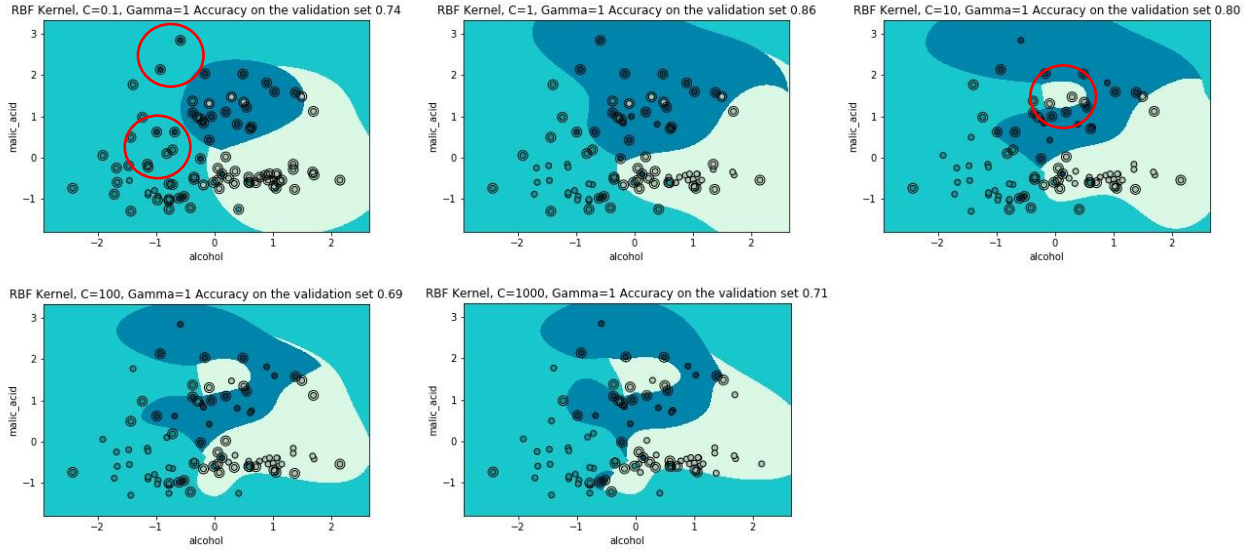




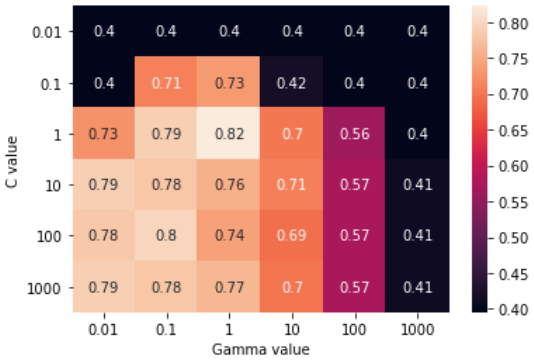


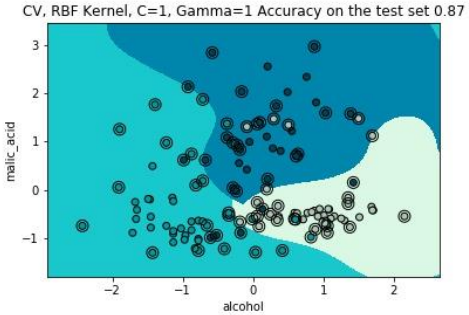


In order to see how the boundary regions change when C changes and γ is fixed, here are the plots for γ=1. As stated before, for larger values of C a smaller margin will be accepted if the decision function is better at classifying all training points correctly. This is evident from the following graphs, for example let's focus on C=0.1 and for C=10. For C=0.1, the regions have regular shapes and some training data are not classified correctly (some of these data are highlighted by red circles). For C=10 instead they are classified correctly, the boundaries are less regular and a bright region appears inside the dark one.



A k-fold cross validation with k=5 is now executed for the evaluation of the best hyperparameters. In this procedure, the original sample (not the one used for testing) is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k − 1 subsamples are used as training data. The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation data. The k results can then be averaged to produce a single estimation.
Even though the ranges of values that work better are very similar to the ones found with the simple grid-search, the exact hyperparameters that work best on the validation set are different. The accuracy reached with these hyperparameters is 0.87, the same as with the best hyperparameters found with the grid-search.



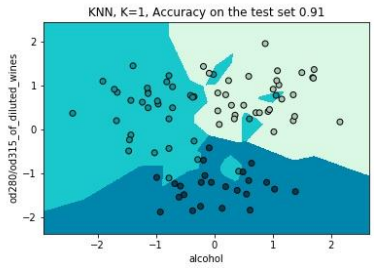


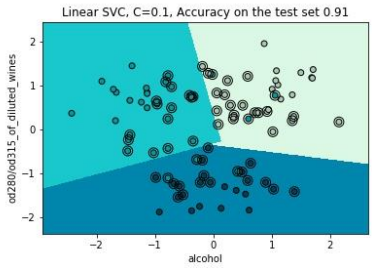
6. DIFFERENCES BETWEEN SVM AND KNN

First, both these algorithms depend on the distance among the samples, so the normalization phase could be crucial in order to get the most out of these methods. Both can be used for regression and classification. While, as said before, KNN does not have a training phase but just stores all the samples, the SVM in the training phase stores only the support vectors and divides the space into regions separated by the support vectors. Therefore, the first big difference that has to be considered when choosing between them is the training time and evaluation time. The KNN takes no time for training but a lot of time for the evaluation, since training samples have to be ordered according to the distance between the evaluated sample, the SVM instead takes more time for the training phase than in the evaluation phase.
Secondly, the KNN algorithm requires only the choice of K, while in SVM the hyperparameters optimization is much more complex. The KNN learns a non-linear decision function, while SVM can learn a linear or non linear function, depending on the kernel.

7. BEST ATTRIBUTES TO USE

As discussed in the introduction, the analysis is now repeated using different couples of attributes.
The first couple is “alcohol” and “od280/od315_of_diluted_wines”, between which the correlation coefficient is very low.
As it can be seen below, KNN with K=1 and Linear SVC with C=0.1 reach an accuracy of 0.91 on the test set.





Next, the analysis is repeated by using two new dimensions built with PCA. In this case several models reach an accuracy of 0.98 on the test set.