MACHINE LEARNING AND DEEP LEARNING

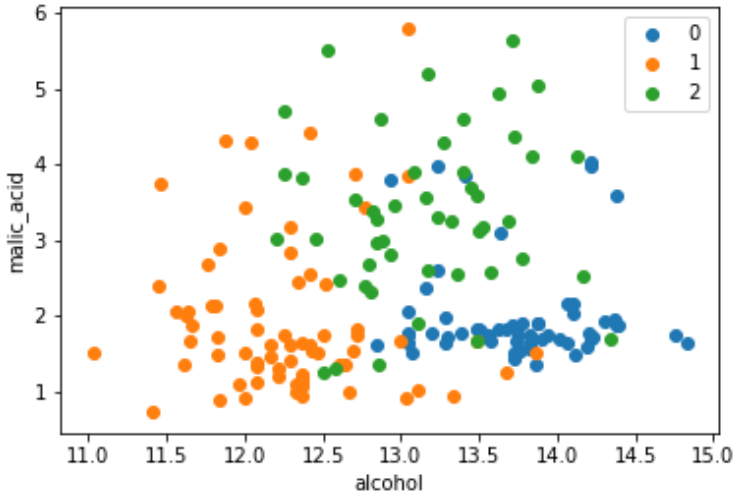
HOMEWORK 1

Sofia Perosin S269748

Description of the data

The starting dataframe contains 178 samples, each of them is described by 13 attributes and has the relative target.

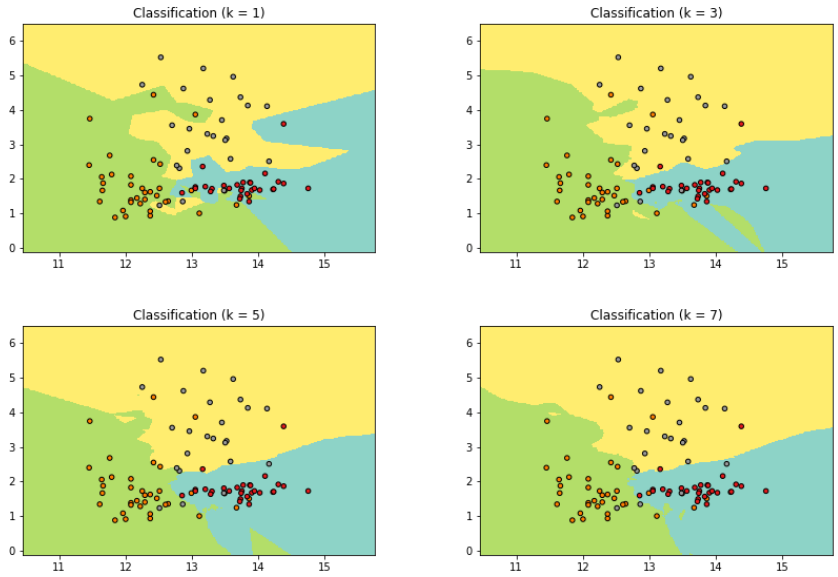
In order to be able to represent data in a 2D dimensions only the attributes *"alcohol"* and *"malic_acid"* will be considered for the analysis. The representation of the resulting dataset is the following one.



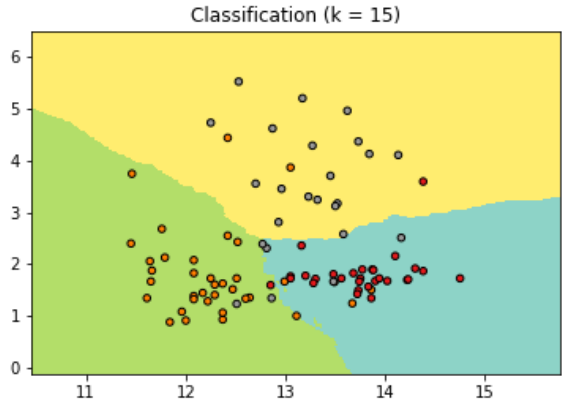
The goal of the homework is to classify each entry wine (identified by attribute *"alcohol"* and *"malic_acid"*) in the corresponding target, by different algorithms: KNN and SVM.

KNN

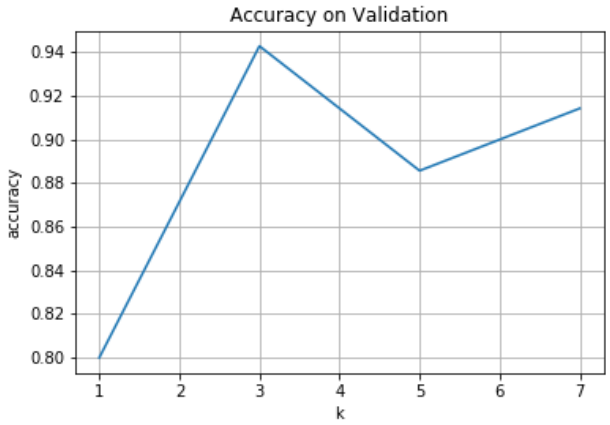
The first algorithm used is KNN and the number of nearest neighbours to take in consideration varies in order to find the best configuration, evaluated on the validation set.



It's possible to see the boundaries change a lot according to the number of nearest neighbours chosen.
When k is small the overall distribution is not taken into account, and the classifier is more flexible. When the number of neighbours increase, the algorithm is more robust in front of outliers because more neighbours are taken into account.
It's possible to see also that when k becomes bigger, the decision boundaries are smoother, so the variance is lower. This is more evident if the value of k increase significantly, for example with k=15 the decision boundaries become:



The accuracies on the validation set obtaining by varying k are the following:

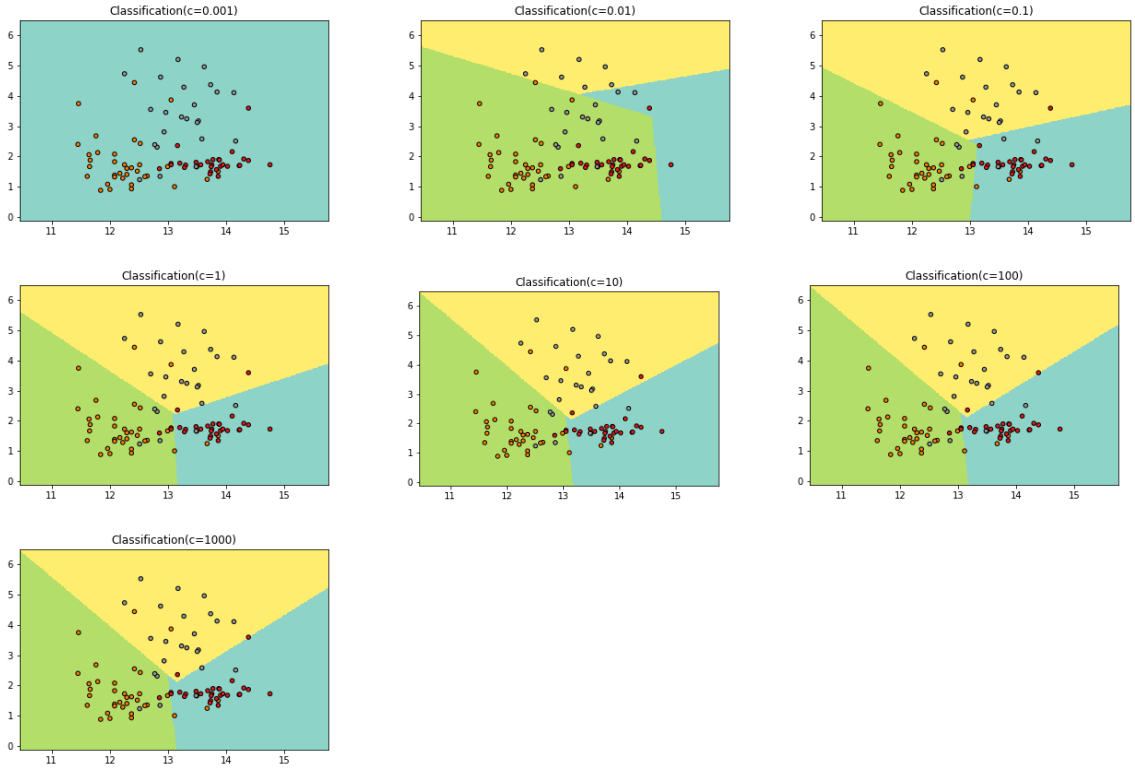


K	Accuracy
1	0.8
3	0.9428571428571428
5	0.8857142857142857
7	0.9142857142857143

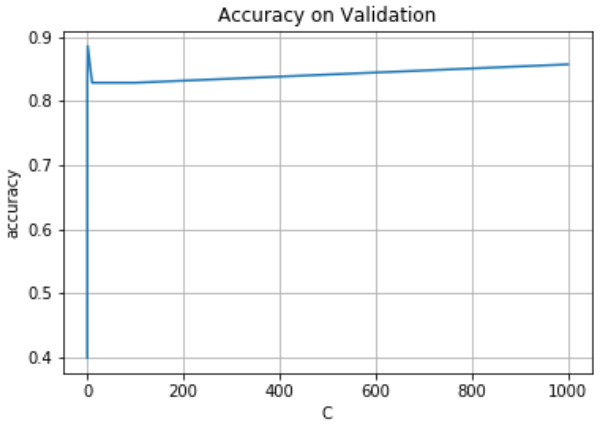
The best k value seems to be 3, having an accuracy of 0.94.
So this will be used to do the final classification on the test set and the accuracy obtained is equal to 0.722.

LINEAR SVM

The second algorithm used is the Linear SVM, implemented by SVC with linear kernel.
In this case the hyper parameter to set is C, which corresponds to the cost of misclassification.
So the larger the C, the more similar the model will be to a small margin classifier and the boundaries will depend only on few samples; instead if C is small ($C \in (0,1]$) the algorithm will look for a large margin separating hyper plane. So if works better a bigger C, the smaller will be the probability that the sample is linearly separable.
So by varying the value of C the boundaries obtained are:



As it's possible to see when C is very small all the points are classified in the same class because errors are not enough penalized, then as C starts to increase the boundaries define the three classes.
The accuracies on the validation set obtaining by varying C are the following:

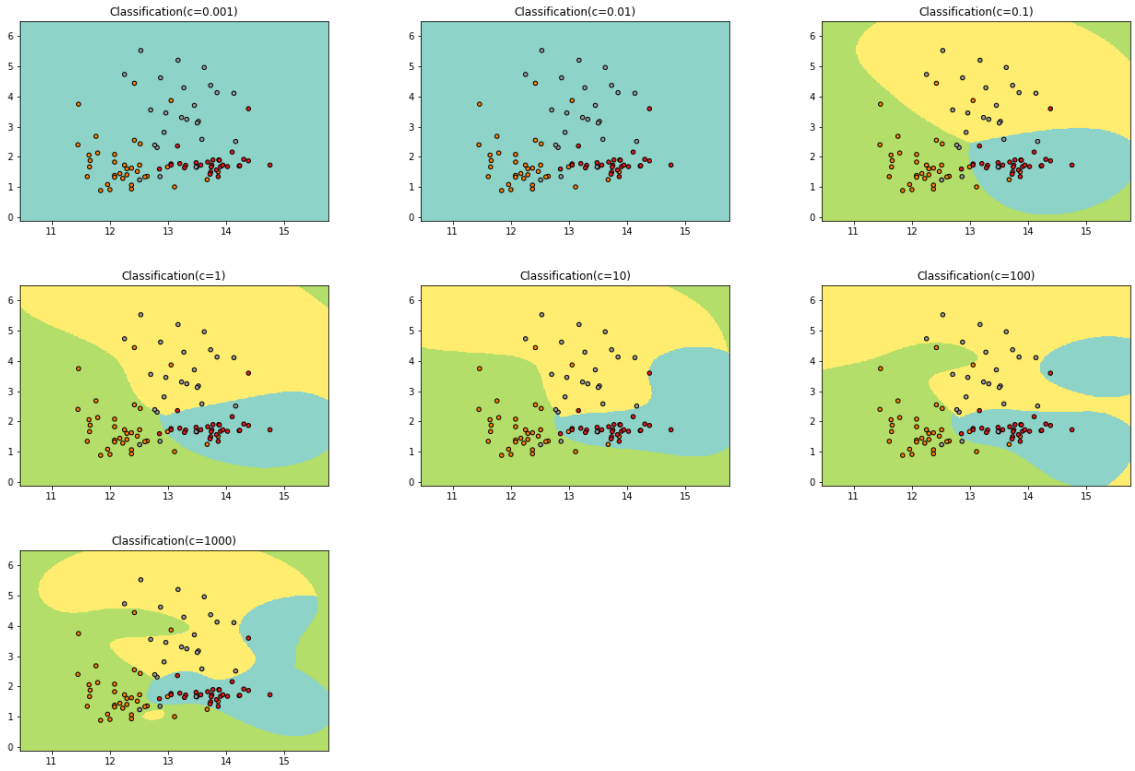


C	Accuracy
0.001	0.4
0.01	0.4857142857142857
0.1	0.8285714285714286
1	0.8857142857142857
10	0.8285714285714286
100	0.8285714285714286
1000	0.8571428571428571

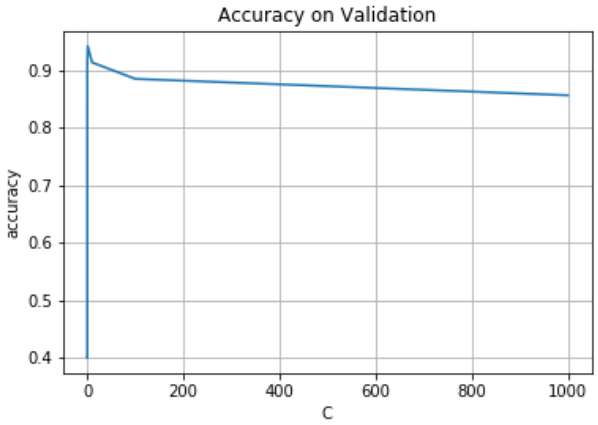
The best C value is 1, having an accuracy of 0.88.
Having C equal to 1 is positive, in fact big C could lead to overfitting: by a theoretical perspective, C has a bound influence on the number of support vectors, bigger C more support vector there will be but it will be also higher the risk of overfitting. Moreover large margin (small C) has the positive aspect to have maximum robustness relative to uncertainty, so it will have good performances with new data.
Then, using the algorithm tuning with C=1 on the test set, the accuracy is 0.67.

SVM RBF KERNEL

The third algorithm is SVM with RBF Kernel, and the boundaries by varying C are:



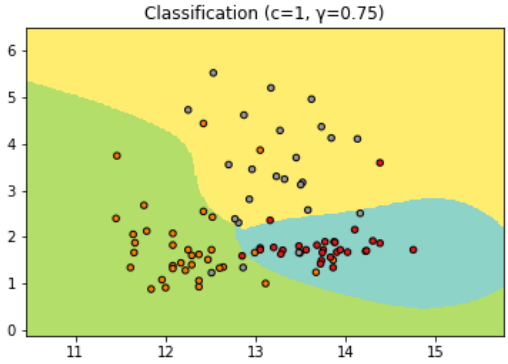
Also in this case small C's are not enough to force the algorithm to find all the three classes.
The main differences with the previous boundaries are due to the fact that are used different kernels: in the first case the algorithm looks for a linear separator and as it is possible to see the boundaries are straight line; instead in this second case, data are separated by a non linear function, this is due to the radial basis function used as kernel.



C	Accuracy
0.001	0.4
0.01	0.4
0.1	0.9142857142857143
1	0.9428571428571428
10	0.9142857142857143
100	0.8857142857142857
1000	0.8571428571428571

The best value is reached with C=1 (with an accuracy of 0.94), then the accuracy decreases, and this could be done to the fact that higher C causes an higher risk of overfitting.
Then evaluating the best C on the test set the accuracy reached is equal to 0.74.

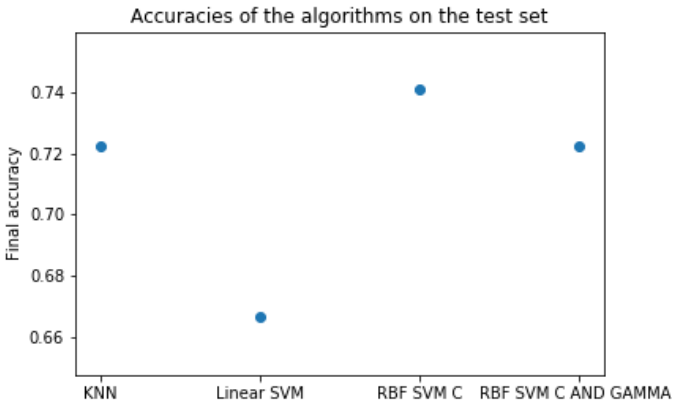
Using RBF kernel it's possible to tune also another hyper parameter: γ. In fact C interacts quite strongly with the kernel parameters, so it could be a good idea to tune both C and γ together.
γ is inversely proportional to the variance of the Gaussian, so it defines how powerful in terms of distance is the influence of a single training data.
When γ is very small the model can't get the shape of the data because it's constrained, instead when γ is too large the radius of the area of influence of the support vectors only includes the support vector itself.
So, in order to obtain a more suitable model it's possible to tune together C and γ.
In order to force the algorithm to commit as less error as possible the lowest value of C used is 0.1 and the biggest 10. Instead the value of γ is tuned for different values between 0 (slightly bigger than 0 because γ can't be equal to 0) and 2, according to the range of variation of the attributes.
The best combination of γ and C is given by γ =0.75 and C=1, that reaches an accuracy of 0.97 on the validation set and 0.722 on the test set. The corresponding decision boundaries are:



As it is possible to see from the preceding evaluation, the accuracy obtained in the test set is always lower than the one obtained in the validation set.
This difference can be due to the fact that the dataset used to tune the hyper parameters is not big enough, and the tuning depends on the data shuffling.
To overcome this issue a good approach could be the one to merge the training and validation set and perform on the new set a k-fold validation.

With a 5-fold validation the best score on the training is 0.87, instead the accuracy on the test set is 0.722 as before.
So there is not the improvement expected, this could be due to the fact that using a stratified sampling in the train, validation and test sets creation, adding more data does not change the final result; or the shuffling used for the creation of the sets has create a test set where data have particular attributes' values which make more difficult the classification.
It could be interesting to note that, instead of merging the train and validation sets, if the train and test sets are merged, and the 5-fold validation is performed on this new set, then the final accuracy obtained evaluating on the validation set greatly improves: the best score obtained during the 5-fold validation is 0.80, instead the accuracy on the evaluation set is 0.91. (There is not overfitting because the model is trained on train+test set and it is evaluated on the validation set).

The overall accuracies obtained are:



Algorithms	Accuracy
KNN	0.7222222222222222
Linear SVM	0.6666666666666666
RBF SVM C	0.7407407407407407
RBF SVM C AND GAMMA	0.7222222222222222

KNN and SVM

- KNN (K-nearest neighbors) is an algorithm that is based on the assumption according to which similar things exist in close proximity.
    - A positive integer k has to be specified, it indicates the number of neighbors that have to be taken into account.
    - After have found the k nearest neighbours of the record that has to be classified, the label is assigned according to the most common label among the neighbors.
    - Usually to compute the distance between each record is used the Euclidean distance, but also other metrics can be used, as Manhattan, Mahalanobis, Minkowski...
- SVM (Support Vector Machine) is an algorithm that aims to find the hyperplane that separates in the best way the classes in the feature space. There are different variants of SVM:
    - SVM with hard margin: no errors are allowed, it's suitable only in the case in which data are linearly separable and there is no noise.
        - $maximize_{\alpha} -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle x_i x_j\rangle + \sum \alpha_i$
          $subjects\ to \sum \alpha_i y_i = 0, \alpha_i \geq 0$
    - SVM with soft margin: some errors are allowed (the percentage allowed depends on C), works well when data are linearly separable.
        - $maximize_{\alpha} -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle x_i x_j\rangle + \sum \alpha_i$
          $subjects\ to \sum \alpha_i y_i = 0, \alpha_i \epsilon\ [0, C]$
    - SVM with kernel: when data are not linearly separable in the current feature space, they might be separable in an higher dimensional space. Since the classifier can be expressed as a sum of inner products it's possible to use a kernel in order to overcome the heavy computation of all these inner products. The decision boundaries found are non linear in the original space.
        - $maximize_{\alpha} -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K(x_i x_j) + \sum \alpha_i$
          $subjects\ to \sum \alpha_i y_i = 0, \alpha_i \epsilon\ [0, C]$

In more practical terms the main difference between SVM and KNN could be seen in the shape of the boundaries.
SVM in fact look for an hyper plane, the boundaries are well defined, they are not jagged, and according to the kernel used are straight or curved lines.
KNN instead classify one point at time according to a majority voting between its k-nearest neighbours, then the resulting boundaries are very irregular.

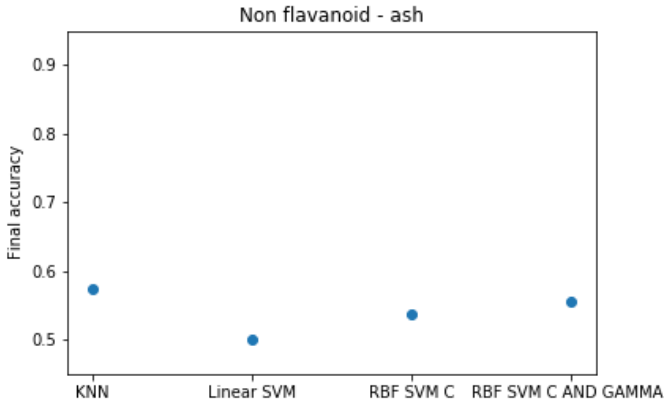
OTHER ATTRIBUTES

Then the same process can be redone with different attributes.

Could be interesting to identify which attributes have the strongest relationship with the output thanks to an Univariate selection, the so called ANOVA, the analysis of variance.

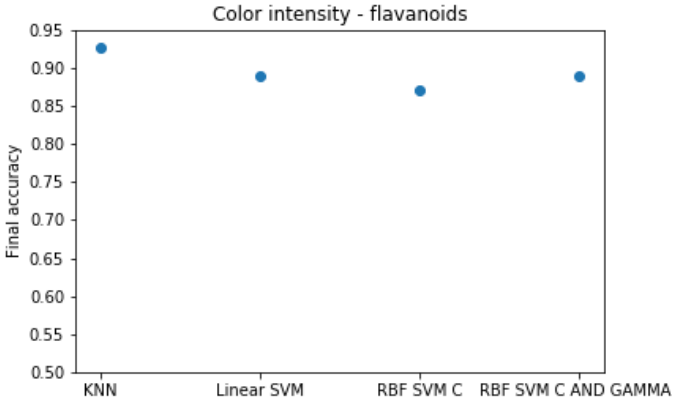
Attribute	Score
proline	16540.067145
color_intensity	109.016647
flavanoids	63.334308
magnesium	45.026381
alcalinity_of_ash	29.383695
malic_acid	28.068605
od280/od315_of_diluted_wines	23.389883
total_phenols	15.623076
proanthocyanins	9.368283
alcohol	5.445499
hue	5.182540
nonflavanoid_phenols	1.815485
ash	0.743381

So it's possible to see that attribute *"ash"* and *"non flavanoid_phenol"* are not good enough to be used, the performances by using these two attributes in fact are not satisfactory.



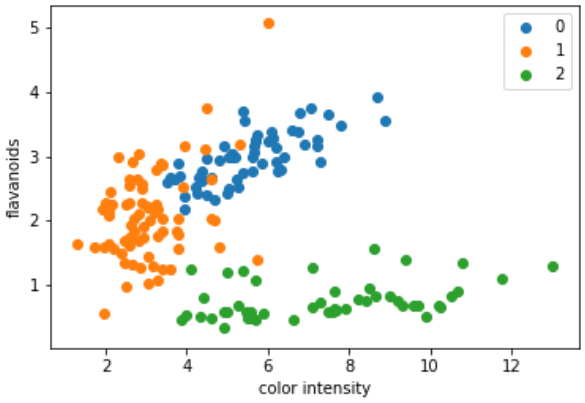
Algorithms	Accuracy
KNN	0.5740740740740741
Linear SVM	0.5
RBF SVM C	0.5370370370370371
RBF SVM C AND GAMMA	0.5555555555555556

Instead by using *"color_intensity"* and *"flavanoids"* the performances increase a lot.



Algorithms	Accuracy
KNN	0.9259259259259259
Linear SVM	0.8888888888888888
RBF SVM C	0.8703703703703703
RBF SVM C AND GAMMA	0.8888888888888888

In fact looking the 2D representation of this new dataset it's possible to see that now, with respect to the two attributes used at the begging, the classes are more clearly separate.



So this is an ulterior proof about which attributes should be used in order to have a stronger, more efficient algorithm.

Looking in the table it's possible to see that the highest attributes are *"color_intensity"* and *"proline"*, but the second one has an enormous variance which affects the performances, so this attribute has to be mandatory standardized, but the combination *"color_intensity"* and *"flavanoids"* still remains the best one.

POSSIBLE IMPROVEMENT

In order to obtain better performances could be done two operations:

- Merge the training and validation sets not only in the case of RBF kernel to tune γ and C (as suggest at point 16) but for all the algorithms applied.
- Standardize data.