POLITECNICO DI TORINO

Master’s Degree in Data Science and Engineering

Machine Learning and Deep Learning

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

Giovanni Cioffi

S277046

Homework 1 - Report

Wine dataset

Wine dataset is a collection of 173 records that are the results of 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. All the quantities are expressed as floating-point numbers, while the three target cultivars are encoded with integers 0,1,2. The distribution of data depending on class labels is the following: 59, 71, 48 records belonging respectively to class 0,1,2.

Data exploration and pre-processing

The entire dataset was loaded using a Pandas Dataframe of size 178 rows (records) x 13 columns (constituents). Another column representing the class label was added. First two columns corresponding to the quantities of 'alcohol' and 'malic_acid' where extracted from the Dataframe in order to represent data on a plane. In this way, we are excluding the other 11 constituents' quantities, so let us suppose that these two features can sufficiently represent the whole variability of the dataset.

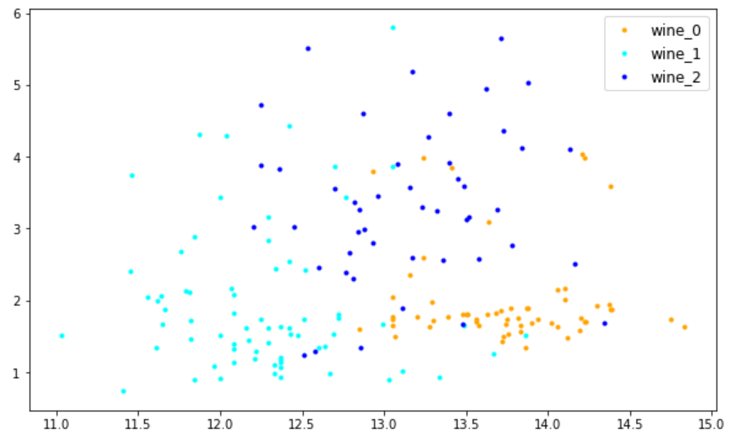


Figure 1 - 2D plot of data

The scatter plot (in Figure 1), made with *matplotlib* library functions, shows all the 178 scatter points with different colours depending on their target class. Scatter points belonging to the same class tend to be in the same area of the plot (wine_0 at bottom right, wine_1 at bottom left, wine_2 at the center), even if the boundaries between those three areas is not so well defined, since there seem to be a lot of noisy points. In order words, data are not perfectly linearly separable.

Dataset was first randomly split to training set and testing set with a 7:3 proportion. Then, the only training set was further split with the same proportion in order to set aside a validation set. Both random splits where made using *train_test_split* Scikit-Learn function, setting the shuffle parameter to *True* in order to shuffle data before the split.

K-Nearest-Neighbour

K-NN is a supervised machine learning algorithm used mainly for solving classification tasks. It exploits similarity measures between points (such as Euclidean or Minkowski distance) for classifying unlabelled incoming data. A new case is classified by a majority vote of its neighbors, with the case being assigned to the class most common among its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor.

Tuning parameter K

For different values of k, which is the number of neighbours used for majority voting, a classification model was build using K-NN *Scikit-learn* classifier. Each model was trained on the training set and evaluated on the validation set in order to find the best k parameter between K = [1,3,5,7]. Moreover, after each training phase decision boundaries were plotted. In each decision boundary plot, scatter points represent training set points coloured with their own labels, while the blue, orange and light-blue areas represent the way in which the model will classify future incoming data depending on where the points will fall.

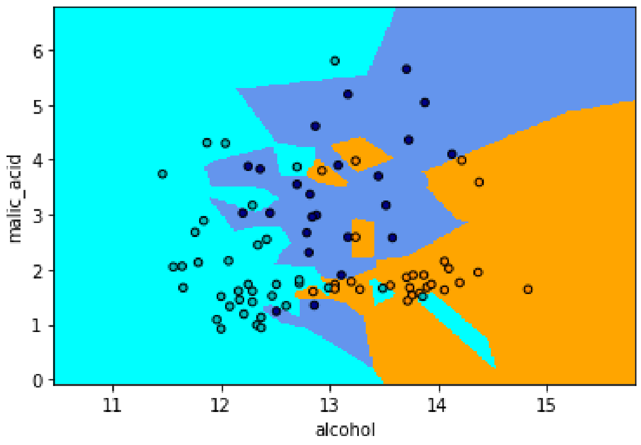


Figure 2 - Decision boundary with k=1

Decision boundaries change evidently with an increasing k. With a small k, the model tends to overfit being very sensitive to noisy points. In Figure 2, decision boundaries with model trained with k=1 are showed; it is quite evident, especially in the middle of the scatter plot (see the orange zones which fall on the blue zone and generate noisy orange decision boundaries) how the model is overfitting on the training data: each new incoming point would be classified as his most close training point even if this could be noise. In this case the model will not be able to generalize on new incoming data. When the model is trained with higher k, decision boundaries appear in a clearer way dividing plane in a number closer to three distinguishable zones, as showed in Figure 3.

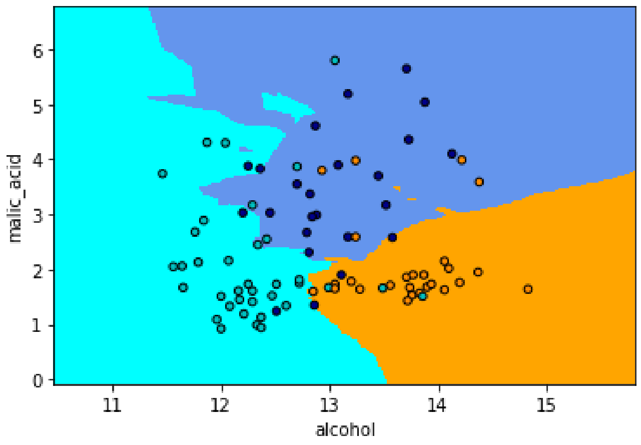


Figure 3 - Decision boundary with k=7

In this way the model will likely be more robust and able to generalize better because of the greater number of training points considered for the majority voting, thus noisy points tend to less influence the classification when they're surrounded by a high number of points belonging to the actual class. Moreover, as shown in figure 4, the accuracy score measured on the evaluation set improves as the value of k increases. Until k=5, there is a consistent and continuous improvement (from less than 75% to 92.5%), then for k=7 the accuracy score does not change. Models with a k greater than 7 reached worst performances. Finally, since k=5 and k=7 parameters gave same results in terms of accuracy score on the evaluation set, k=7 whose chosen as best parameter because it generates the most coherent decision boundaries, which seem to be less influences by noise.

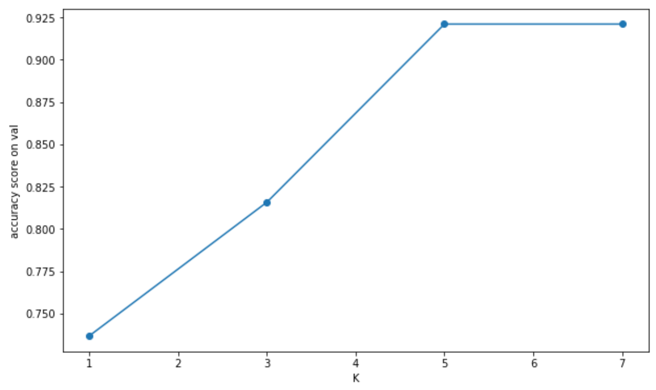


Figure 4 - Accuracy score of model trained with different k

Evaluation on test set

A classification model with k-NN was then trained on the whole initial training set X (training set + validation test) with k=7 and evaluated on the test set reaching an accuracy score of 79.6%. Classification report and confusion matrix (with actual class on the x axis and predicted class on the y axis) are showed below, it is clear from all the metrics that the model finds difficulties on properly classifying points belonging to class 2 (which is the class with less training points, as mentioned in the data exploration section).

```
              precision    recall  f1-score   support

           0       0.83      0.79      0.81        19
           1       0.94      0.81      0.87        21
           2       0.61      0.79      0.69        14

    accuracy                           0.80        54
   macro avg       0.80      0.79      0.79        54
weighted avg       0.82      0.80      0.80        54
```

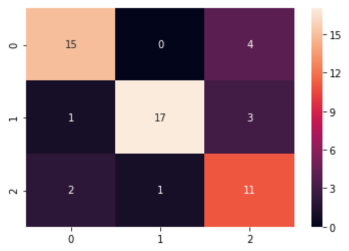


Figure 5 – Classification report and confusion matrix with KNN with k=7

Linear SVM

Linear Support Vector Machine is a supervised learning algorithm that finds a hyperplane in an N-dimensional space (where N is the number of features) that distinctly classifies the data points. Its objective is to find a plane that has the maximum margin, the maximum distance between data points of both classes. Support vectors are the data points that are closer to the hyperplane and influence the position and orientation of the hyperplane.

The same procedure described above for k-NN was applied using Linear SVM classification algorithm varying the parameter C. C is called regularization parameter (or cost of misclassification) in a soft margin classifier, it multiplies the slack variables in the minimization objective function introduced to soften the problem. In the margin maximization problem solved by this algorithm, α_i, which are the coefficients that determine which points will be in the solution (the support vectors) are bounded by C. C defines how much the margin is soft/hard: when C is high the classifier is 'allowed' to make more errors, more support vectors will be part of the solution. Several models where trained on the training set with several values of C = [0.001, 0.01, 0.1, 1, 10, 100,1000] and the decision boundaries were plotted. The following plot shows how the accuracy score on the validation set varies with different value of C.

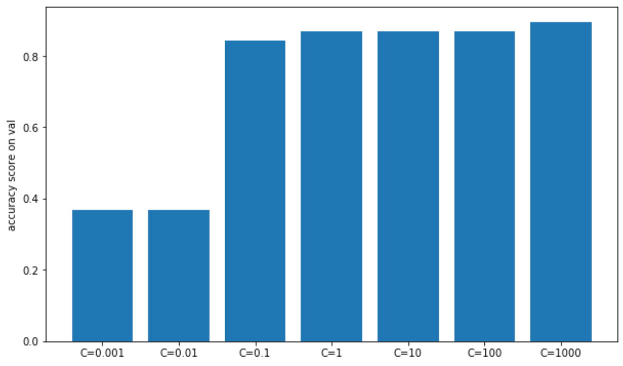


Figure 6 - Accuracy score on validation set with SVC(kernel='linear') with several C

Support Vector Machine linear classification with linear kernel performs better with C=1000 with respect to the other values of C. Performances in terms of accuracy score tend to increase with higher values of C, mostly when passing from C=0.01 to C=0.1, then there are only slight improvements. The 2D representation of data mentioned above clearly shows that data are not perfectly linearly separable because there are a lot of noisy points; for this reason, it is necessary to soften the classification model in order to allow it to make some mistake. Looking at the decision boundaries plot with small values of C such as 0.001 and 0.01: in the former case every new incoming point would be classified at the same way because the algorithm is not able to find any hyperplane that maximizes the margin among the 3 classes, in the SVM finds a way to separate classes, but right separation between the three classes (figure X), resulting in both cases with an accuracy less than 40%.

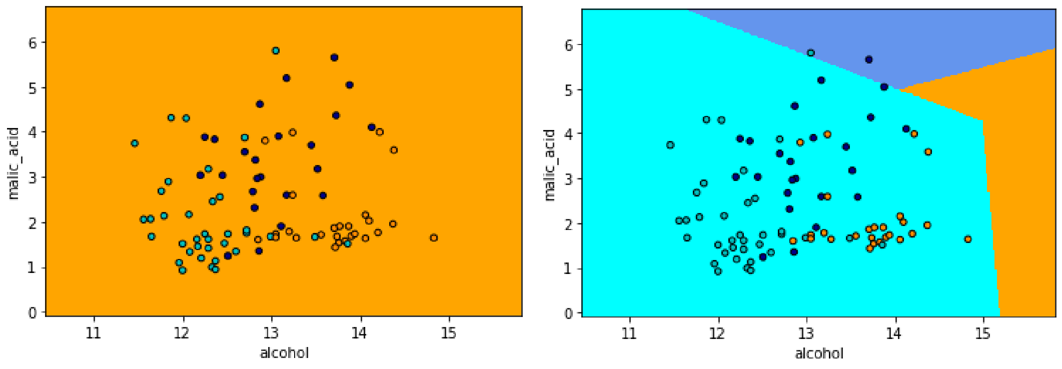


Figure 7 – Decision boundaries with C = 0.001 (on the left) and C=0.01 (on the right)

Then, starting from C=0.1 on, a significant improvement is observable: the decision boundaries start to divide the area in a reasonable way, the three classification areas are not strongly influenced by outliers, the accuracy doubles overcoming 80% with C=0.1 and arrives at 89% with C=1000.

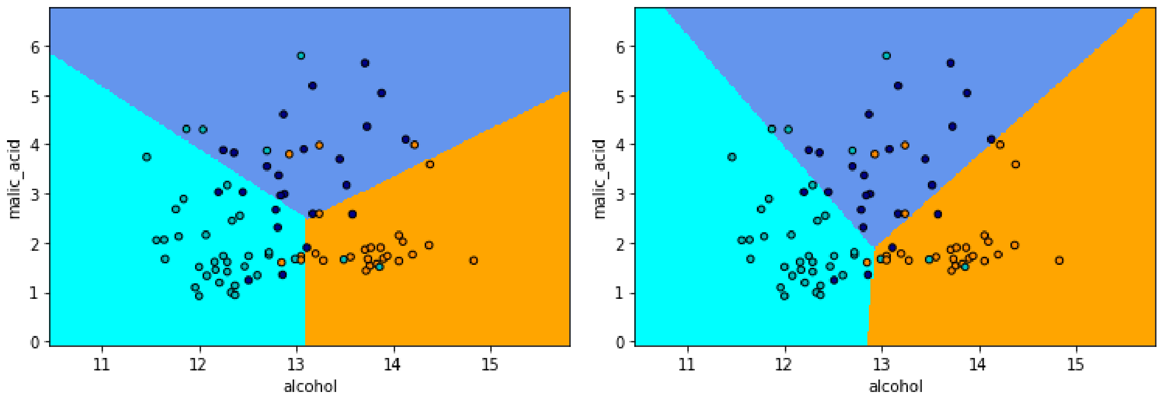
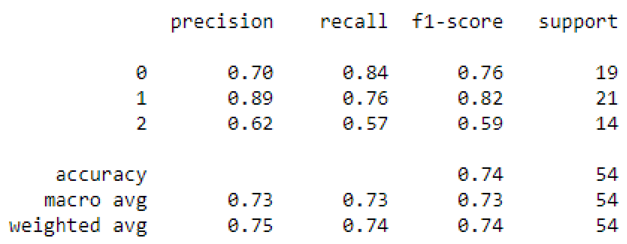


Figure 8 - Decision boundaries with C=0.1 (on the left) and C=1000 (on the right)

A model was trained with C=1000 on both training and validation sets and then evaluated on the test set. Classification report and confusion matrix are showed below. Performances are worst with respect to the model trained with k-NN, accuracy score is 74% and again performances for class 2 are not good.



```
              precision    recall  f1-score   support

           0       0.70      0.84      0.76        19
           1       0.89      0.76      0.82        21
           2       0.62      0.57      0.59        14

    accuracy                           0.74        54
   macro avg       0.73      0.73      0.73        54
weighted avg       0.75      0.74      0.74        54
```

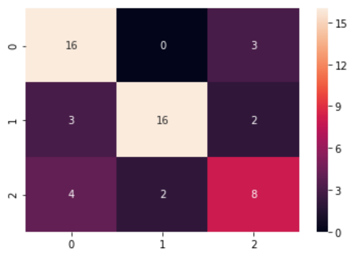
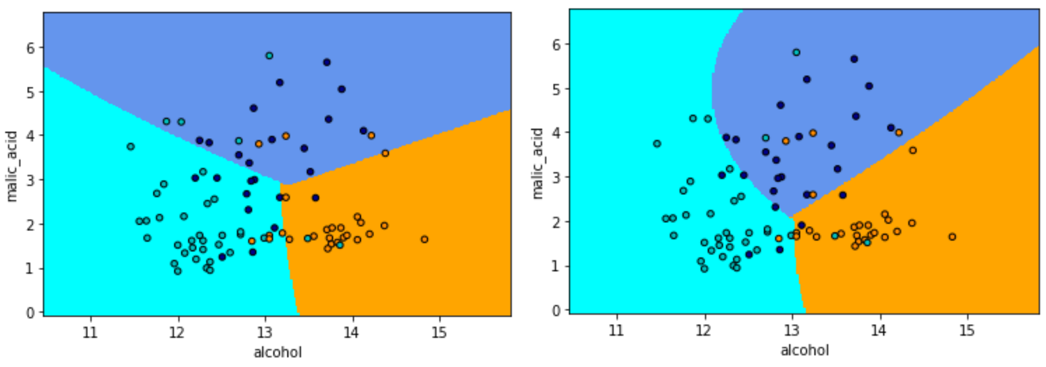


Figure 9 - Classification report and confusion matrix with linear SVM and C=1000

SVM classification with RBF kernel

The training phase was then conducted with the Radial Basis Function kernel, using the same classifier. Same values of C were tuned and again decision boundaries were plotted. RBF decision boundaries have a curvy shape which is different from the linear one of linear SVM decision boundaries. This is due to the different kernel function used to compute the dot product of two vectors in some higher dimensional feature space in order to make data separable. The model which gave the best score on the validation set was the one with C=10; evaluating it on the test set we obtain accuracy close to 78%, which was better that the model trained with a linear kernel. Without specifying the gamma parameter, by default sklearn utilizes 1 / (n_features * X.var()) as value of gamma.



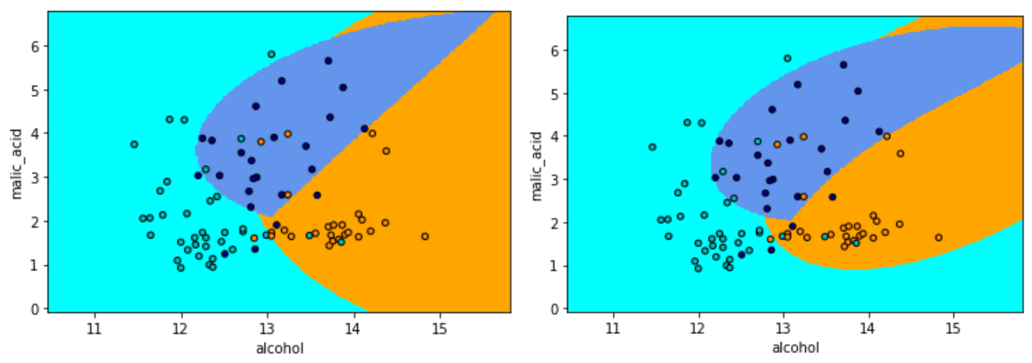


Figure 10 – Decision boundaries with rbf kernel. C=1 (upper-left), C=10 (upper-right), C=100 (bottom-left), C=1000 (bottom-right)

Using RBF kernel, a grid search was applied in order to find the best combination of C and gamma parameter both manually (Table 1) and using GridSearchCV. The gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’. It can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.

Gamma/C	**0.01**	**0.1**	**1**	**10**	**100**
1e-7	0.3684	0.3684	0.3684	0.3684	0.3684
1e-5	0.3684	0.3684	0.3684	0.3684	0.3684
1e-3	0.3684	0.3684	0.3684	0.6315	0.8421
1e-1	0.3684	0.52631	0.8947	0.8947	0.8947
1e-0	0.3684	0.8684	0.8684	0.8684	0.7894

Table 1 – Accuracies on the validation set with ‘rgf’ kernel after manual grid-search

One of the three best combination found is {C=1, gamma=1e-1, kernel = 'rbf'},. The model was trained with these parameters on the training set and evaluated on the test set obtaining scores 77% of accuracy and 78% weighted average f1-score. His decision boundaries are showed below in figure.

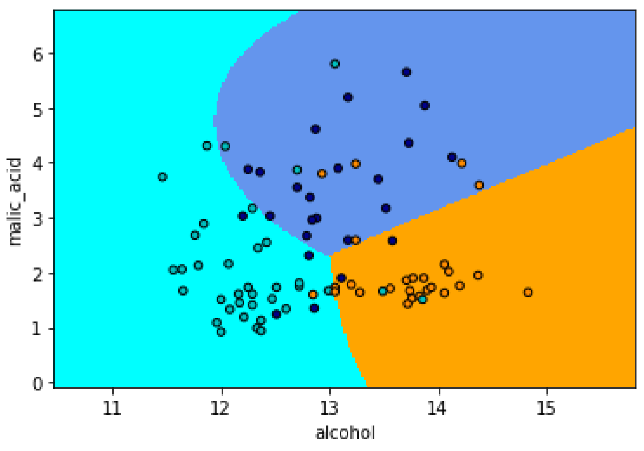


Figure 11 – Decision boundaries with C=1, gamma=1e-1, kernel = 'rbf'

KFold and grid search on the entire training set

Then, training and validation set were merged, 70% of training and 30% test data. The grid search was repeated on the entire training set with the same configurations of C and gamma, this time with 5-fold validation. By making the grid search on the entire training set, the best configuration found is {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'} and it performs better on the test set with an accuracy of 81.48%. The more training data are fed to a classification model, the better it performs, thus by feeding the model with both training and validation set the model is more robust and more able to generalize.

Difference between KNN and SVM

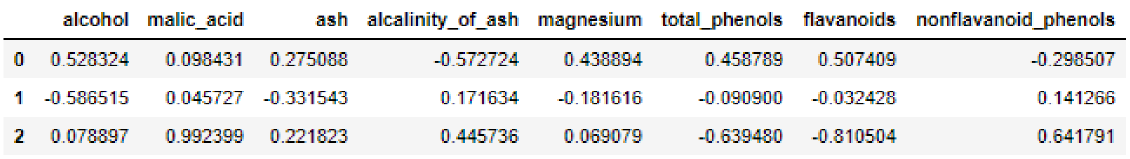
Both K-Nearest-Neighbour and Support Vector Machine are widely used supervised learning algorithms for classification tasks. When unlabelled new data come, those algorithms make the prediction in different ways.

KNN algorithm's basic logic is to explore test data neighbourhood for making the prediction, assuming them to be similar to training ones. It computes the distances between the new unlabelled point and all the training data, then the lowest k distances are taken into consideration and the point is classified with majority voting. Thus, especially when the sample size is very large (which is not the case of this homework), KNN has a high computational cost, because it needs compute of all the distances with all the training data for each new incoming point to classify. Nevertheless, it is very easy to implement since it has few parameters to manage: K and the measure of distance.

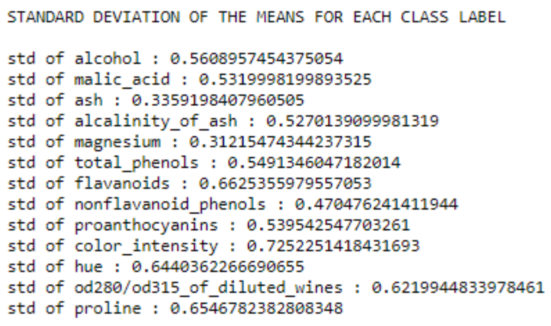
SVM, instead, tries to find a hyperplane (a separation which can be different depending on the kernel function used) between the training data that maximizes the margin between the classes. Thus, differently from KNN, it does not need all training data to compute distances, but the solution is made of only the so-called support vectors, which are a small part of the training data. In most of real cases such as the one solved in this homework, data are not perfectly linearly separable, so SVM has to soften his margin and allow some points to be out of the boundary depending on how large is the regularization parameter. SVM is generally less influenced by outliers with respect to KNN and it performs better especially when the number of features exceed the training size, in other words in very high dimensional spaces.

Different pairs of attributes

In order the choose the best pairs of attributes for training the model, data were firstly normalized with RobustScaler from *Scikit-learn* in order to properly compare values of different scales, depending on the attribute. For each attribute and class label, the mean of the values of that label is computed; then, for each attribute, the standard deviation of the means of the classes was computed. Means and standard deviations of each features are showed below.



	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols
0	0.528324	0.098431	0.275088	-0.572724	0.438894	0.458789	0.507409	-0.298507
1	-0.586515	0.045727	-0.331543	0.171634	-0.181616	-0.090900	-0.032428	0.141266
2	0.078897	0.992399	0.221823	0.445736	0.069079	-0.639480	-0.810504	0.641791



```
STANDARD DEVIATION OF THE MEANS FOR EACH CLASS LABEL

std of alcohol : 0.5608957454375054
std of malic_acid : 0.5319998199893525
std of ash : 0.3359198407960505
std of alcalinity_of_ash : 0.5270139099981319
std of magnesium : 0.31215474344237315
std of total_phenols : 0.5491346047182014
std of flavanoids : 0.6625355979557053
std of nonflavanoid_phenols : 0.470476241411944
std of proanthocyanins : 0.539542547703261
std of color_intensity : 0.7252251418431693
std of hue : 0.6440362266690655
std of od280/od315_of_diluted_wines : 0.6219944833978461
std of proline : 0.6546782382808348
```

Figure 12 – Means of values for each attribute and class label (up) and standard deviations of the means (down)

'color_intensity' and 'flavanoids' and 'proline' are the attributes with the highest standard deviation between the means of the class labels values. They should be the features that allow to better discriminate among different classes; indeed, the plot of figure 13 showing the 2D representation of 'flavonoids' and 'proline' data, states that data are better linearly separable with respect to 'alcohol' and 'malic_acid' ones.

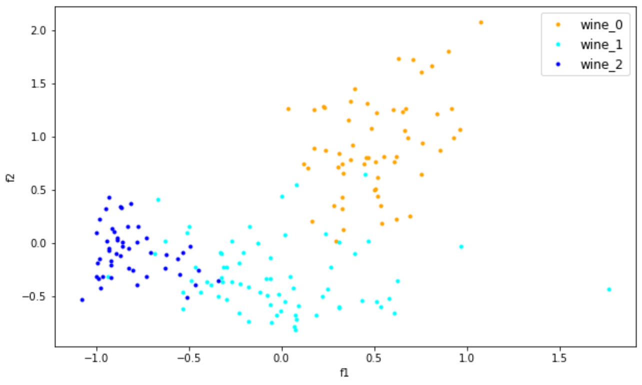


Figure 13 – 2D representation of 'flavonoids' and 'proline'

The classification task with k-NN, linear kernel and rbf kernel SVM was repeated with those features (parameters were tuned with grid-search and K-fold cross validation) the results are summarized in the table below.

ATTRIBUTES	**CLASSIFICATION ALGORITHM**	**ACCURACY ON TEST SET**
color_intensity and flavanoids	k-NN	90.7%
	SVM with linear kernel	87%
	SVM with rbf kernel	87%
color_intensity and proline	k-NN	87%
	SVM with linear kernel	88%
	SVM with rbf kernel	87%
flavonoids and proline	k-NN	88.9%
	SVM with linear kernel	92.6%
	SVM with rbf kernel	90.7%

Table 2 – Accuracy scores with several couples of attributes using k-NN and SVM with linear and rbf kernel

The best accuracy score on the test set is achieved with linear SVM on the couple plotted in figure 13; the algorithm finds a proper hyperplane to maximize the margin between classes because data are well linearly separable.