MACHINE LEARNING & DEEP LEARNING

POLYTECHNIC OF TURIN

HOMEWORK 1

MAURIZIO VASSALLO, S276961

Contents

INTRODUCTION

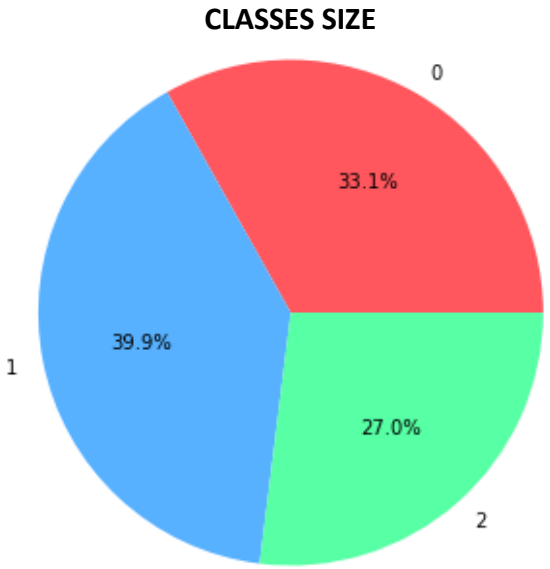
This is the homework 1 report of the course of Machine Learning & Deep Learning at polytechnic of Turin.

The objective is to analyze the wine dataset available in the scikit library, use this dataset to learn a model in order to classify the data in the right category.

DATA EXPLORATION

This dataset contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. It contains 13 features representing the quantities of 13 constituents found in each of the three types of wines.

The dataset size is 178 rows and 13 columns, labelled as **0** (59), **1** (71), **2** (48).



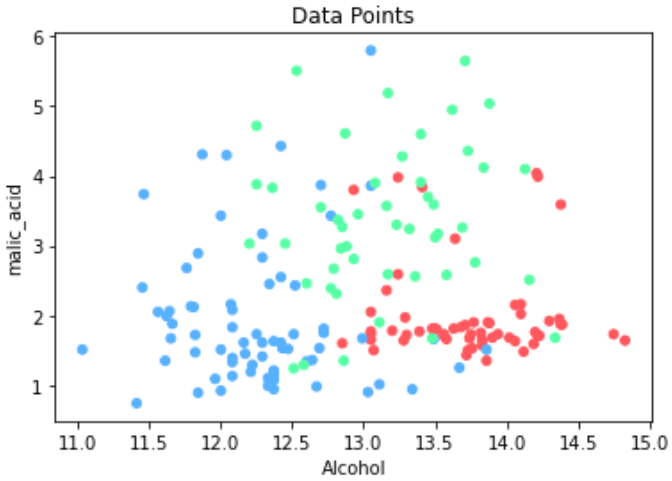
As shown in the pie chart the classes are balanced and this is good to have a fair model.

As requested, the main features of interest are the first 2: **alcohol** volume and the concentration of **malic acid**.

In the table below there are some statistics of the 2 features:

	count	mean	std	min	25%	50%	75%	max
alcohol	178	13.00	0.81	11.03	12.36	13.05	13.67	14.83
malic acid	178	2.34	1.12	0.74	1.60	1.86	3.08	5.80

Scattering all the point in a 2D plane



It is possible to see that the classes are almost well defined, since the points within the same class are near, but there are some points belonging to a specific class in proximity of a 'group' of another class points.

PREPROCESSING

For the pre-processing part there are some steps.

For the cleaning process there is not so much to do since the data is well formatted and there are not null values inside the data.

The dataset is split in train, validation and test set with the following proportions: 5, 3, 2. This is made using the *train_test_split* function avaible in the *sklearn.model_selection* library, two times: one time for splitting between train and test set and then split the train set in the actual train set and validation set. There is also merged train set made by the concatenation of the train and validation set used to evaluate the performance of the model with the best set of parameters found; since it merges the two datasets the proportions are 7 for the train test and 3 for the test set.

I also tried to normalize data with *normalize* function in *sklearn.preprocessing*, but the accuracy has not improved.

MODELS TRAINING

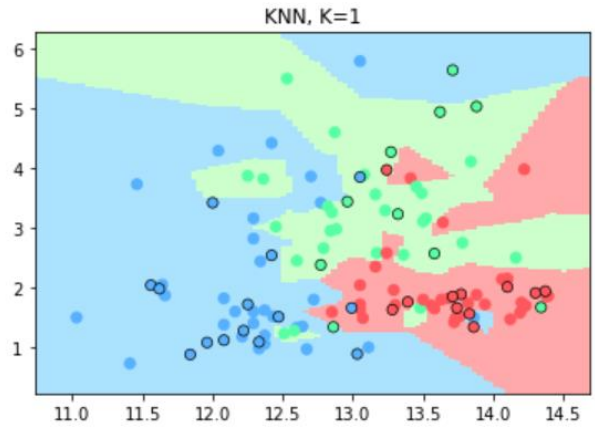
As requested, there are 3 models to train: K-Nearest Neighbor, SVM with linear kernel and SVM with rbf kernel.

KNN

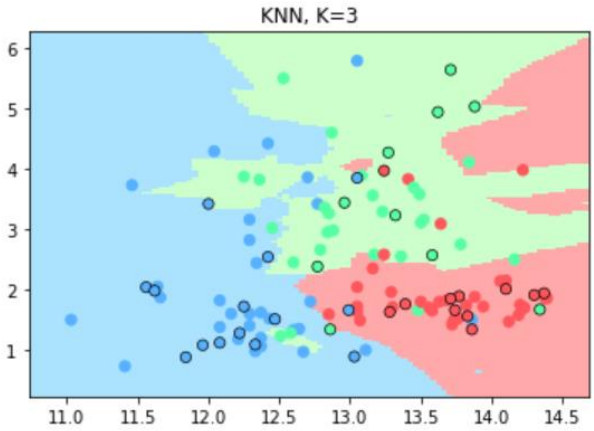
K-Nearest Neighbor must be trained used different values of k (the number of points to be considered to classify a given data). These values are:

K = [1,3,5,7]:

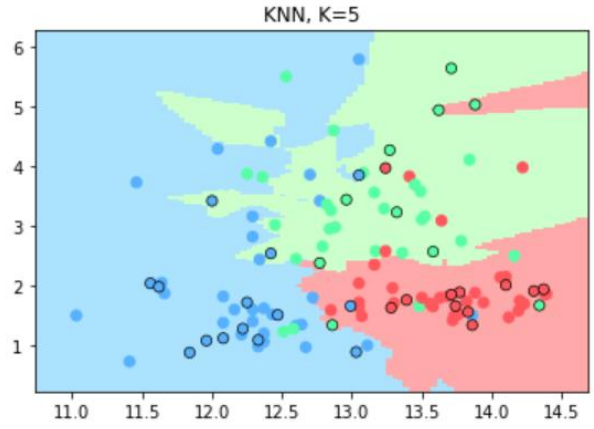
Legend:



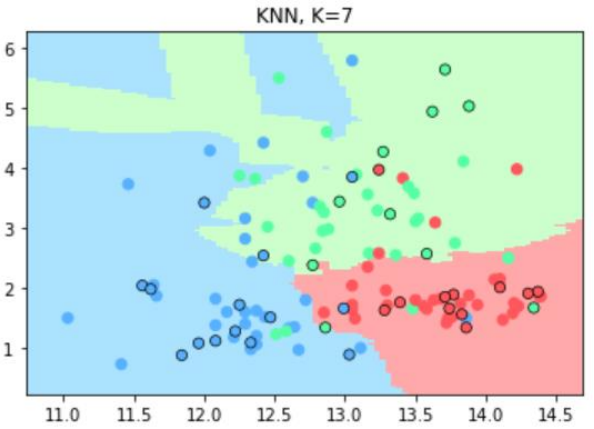
k=1 accuracy=0.77778



k=3 accuracy=0.83333

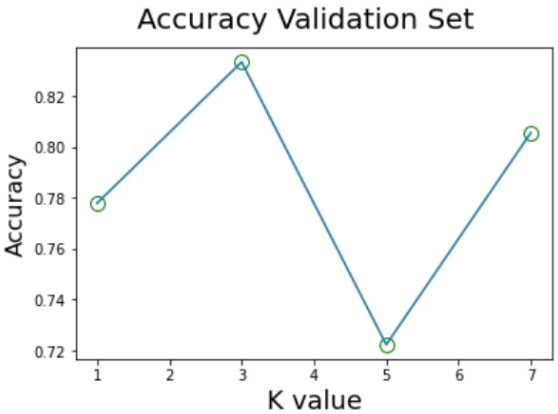


k=5 accuracy=0.72222

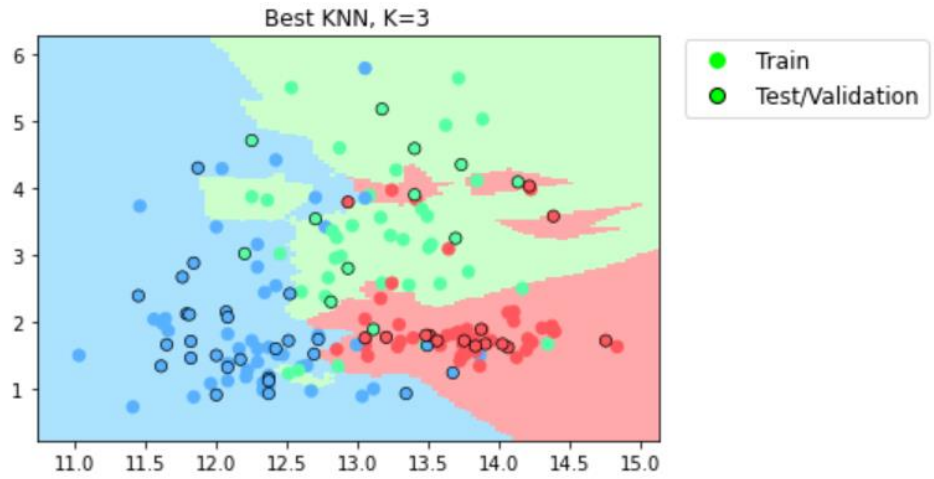


k=7 accuracy=0.80556

The boundaries changes according to the number k: increasing k we accept to mis-classify some points in order to improve the performance, since we look for a higher number of points in the neighbourhood of the data.



As we can see the accuracy increases and decreases but in general it gets better with higher values of k.




```
Test with k=3 accuracy=0.85185 (Best Validation=0.83333)
```

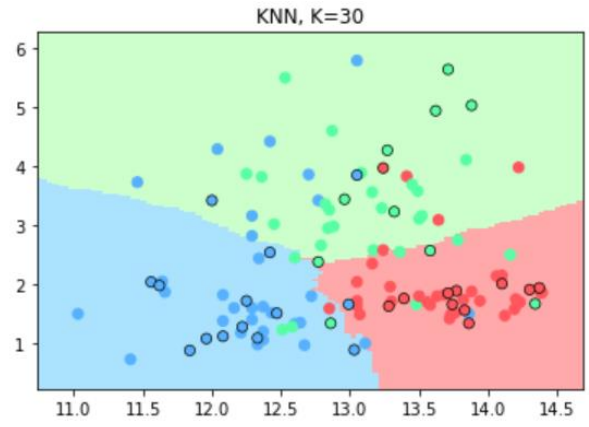
The k value with the highest accuracy is used to evaluate the model in the test set. The model works fine and the accuracy on the test set is slightly higher than the validation set.

Even if the k value with highest accuracy is 3 it may not generalize well so it would be better to choose a higher value of k. Low value of key may overfit the model.

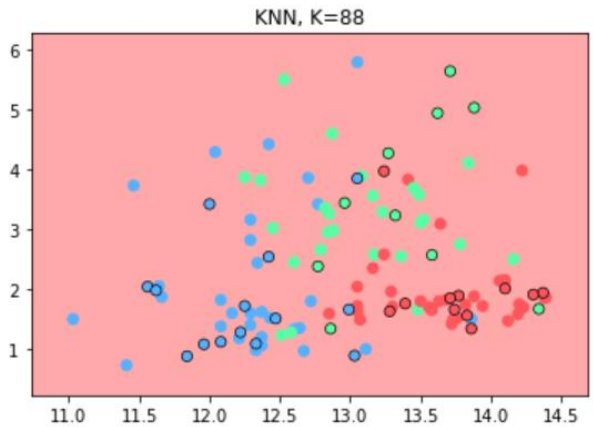
I tried different values for the KNN, in particular:

K_ext = [1, 2, 3, 4, 5, 6, 7, 9, 15, 30, 60, 88]

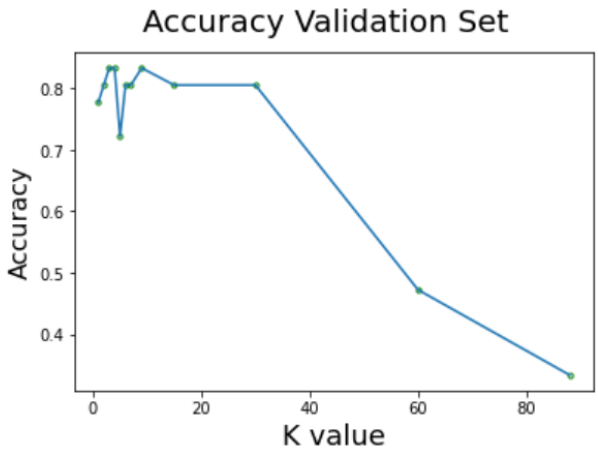
An interesting thing is that for K large enough (K=30) KNN boundaries are similar to SVM with linear kernel, while for values of K even higher it starts to perform very badly how it can be expected (for K=88, the training set size, the boundaries are like SVM with a very low C)




```
k=30 accuracy=0.80556
```




```
k=88 accuracy=0.33333
```

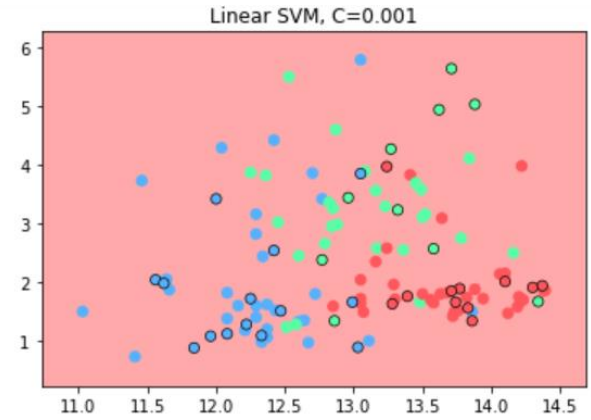


The graph above shows the K must be lower than half of the train set size (88 in this case)

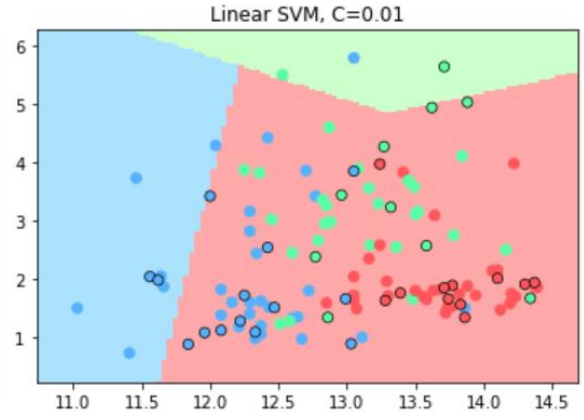
LINEAR SVM

SVM with linear must be trained used different values of C (degree of importance that is given to miss-classifications). These values are:

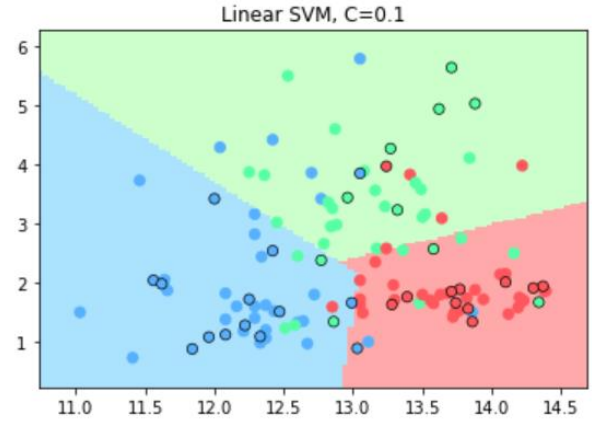
C = [0.001, 0.01, 0.1, 1, 10, 100, 1000]:



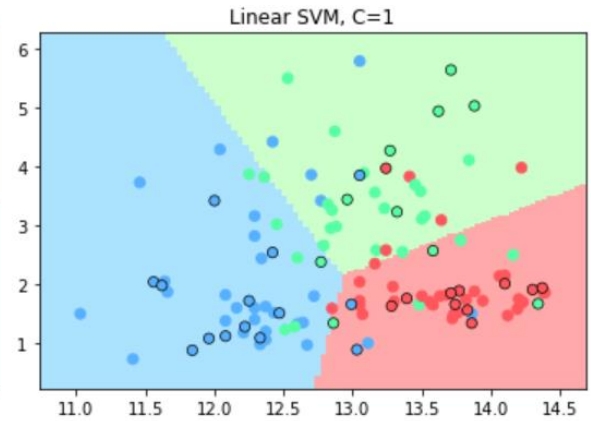
C=0.001 accuracy=0.33333



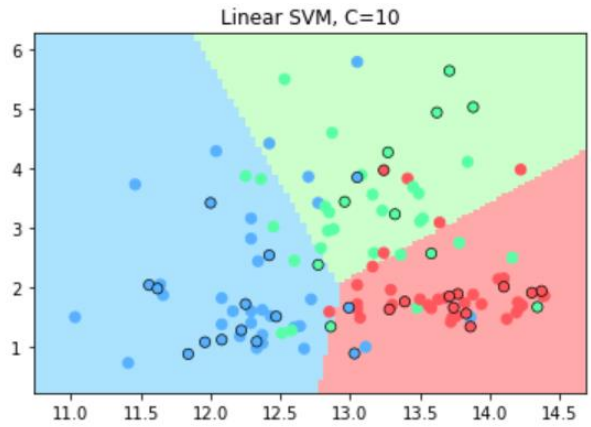
C=0.01 accuracy=0.41667



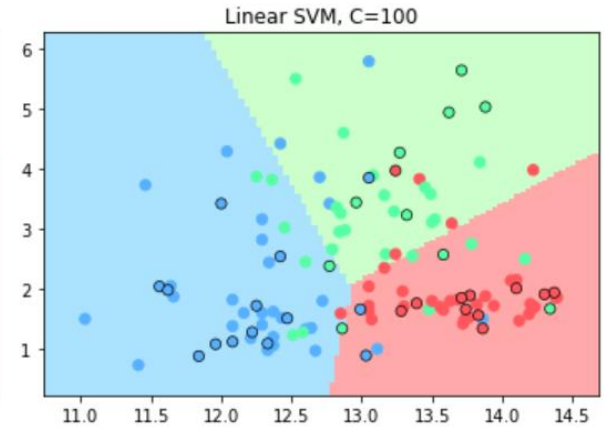
C=0.1 accuracy=0.80556



C=1 accuracy=0.77778

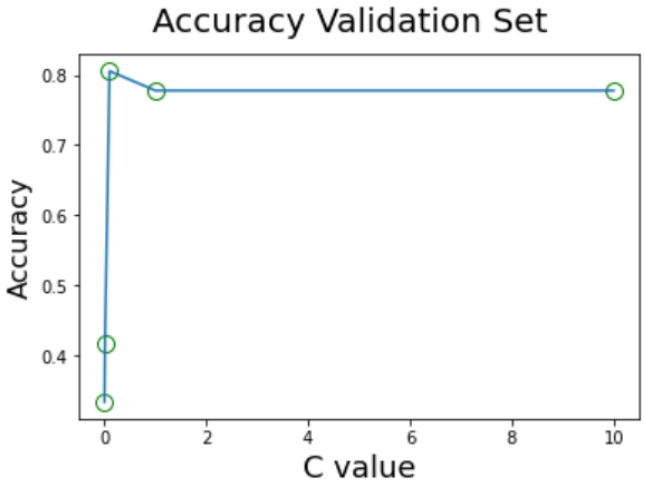


C=10 accuracy=0.77778



C=100 accuracy=0.77778

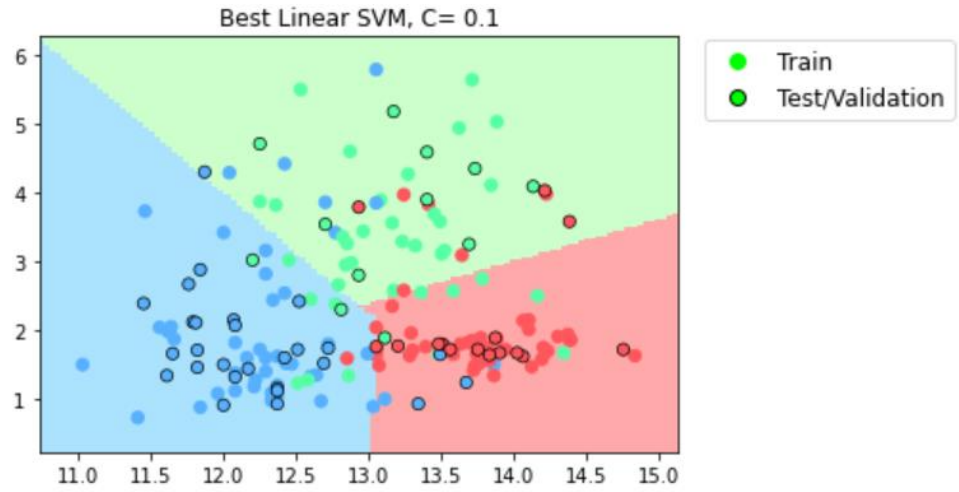
(Graph with C=1000 is omitted since the boundaries is very similar to the graph with c=100)



(Values of C higher than 10 are omitted for visualize better the interesting parts)

An interesting fact that the graphs show is that when c is very small (the penalization for mis-classified data is very small), the model does not care about the points and it classifies right only the points of one class, the boundaries in the zone of interest is all the same red color (same class).

With the increasing of c the model start to right classify the data. With a value of c equal to 1 the model seems to reach it max ability and for a value of c greater than 10 the boundaries do not change anymore.




```
Test with k=0.1 accuracy=0.81481 (Best Validation: 0.80556)
```

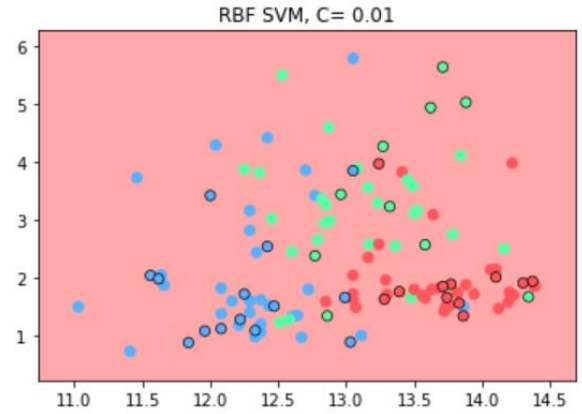
The c value with the highest accuracy is used to evaluate the model in the test set. It gets a high score, very similar to the validation set, it means that the model may generalize well.

RBF SVM

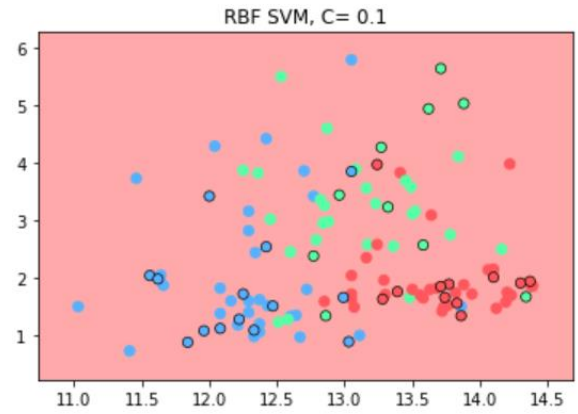
Next model to train is the SVM with rbf kernel, in this case the model tries to map the data in a higher dimensional space and then find an appropriate hyperplane in order to separate data and maximize the margin.

Also, in this case the value of c must be chosen among these values:

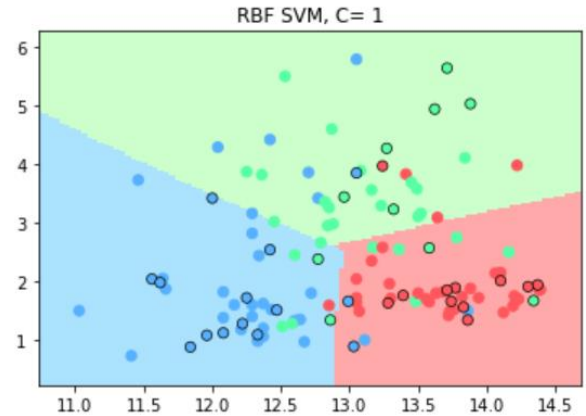
C = [0.001, 0.01, 0.1, 1, 10, 100, 1000]:



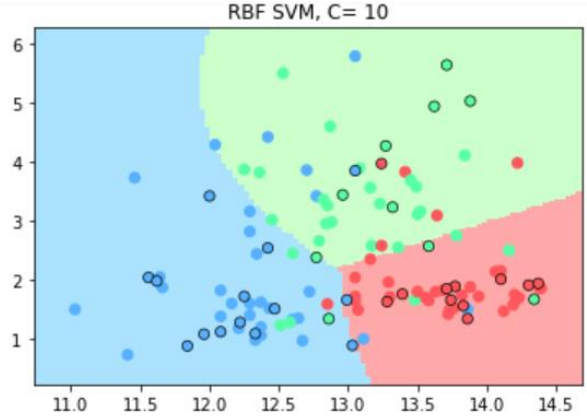
C=0.01 accuracy=0.33333



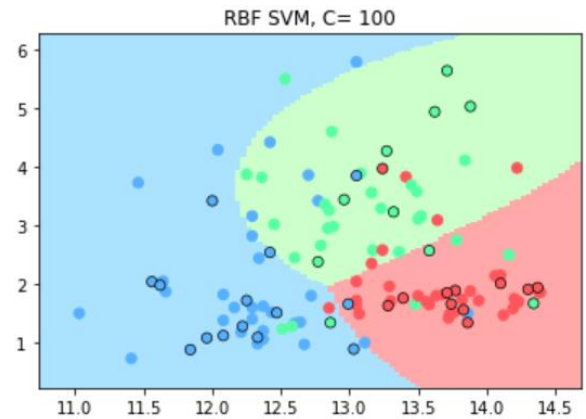
C=0.1 accuracy=0.33333



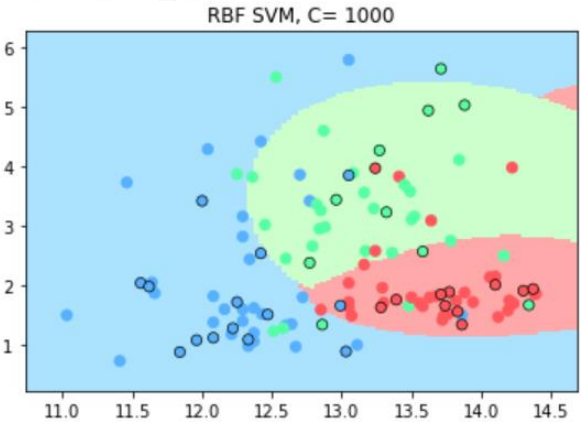
C=1 accuracy=0.77778



C=10 accuracy=0.80556

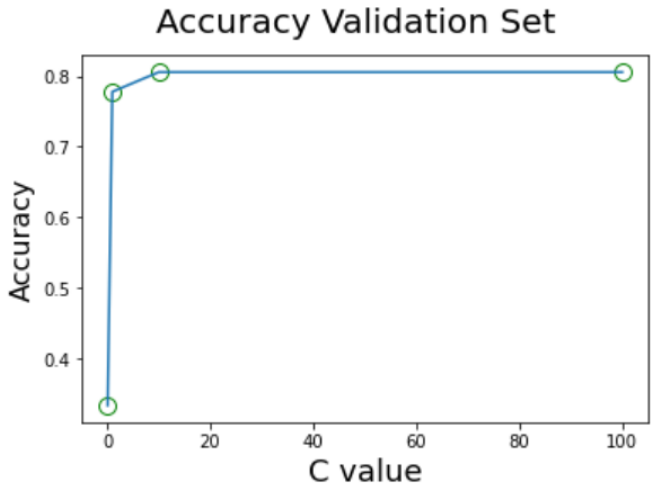


C=100 accuracy=0.80556



C=1000 accuracy=0.80556

(Graph with C=0.001 is omitted since the boundaries is very similar to the graph with c=0.01)



(Values of C lower than 0.1 or higher than 10 are omitted for visualize better the interesting parts)

Also, for SVM with rbf kernel, when c is very small, the model does not care about the points and a lot of points are mis-classified.

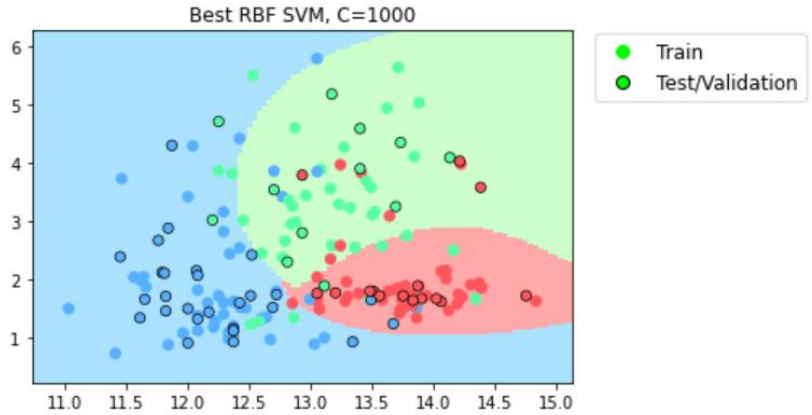
With the increasing of c the model start to right classify the data. With a value of c equal to 1 the model classifies well the points and for a value of c greater than 1 the boundaries start to be non-linear but the score does not increase.

There are 3 different values of C with the highest score, C = [10, 100, 1000]

So using this values to train a model to predict the test, the higher accuracy is for C = 1000

Test with C=10 accuracy=0.83333 (Best Validation: 0.80556) *(graph omitted)*

Test with C=100 accuracy=0.83333 (Best Validation: 0.80556) *(graph omitted)*




```
Test with C=1000 accuracy=0.85185 (Best Validation: 0.80556)
```

GRIDSEARCH

For SVM with rbf kernel classifier must be performed also a GridSearch on both C and Gamma to find the best couple.

Gamma decides that how much curvature we want in a decision boundary.

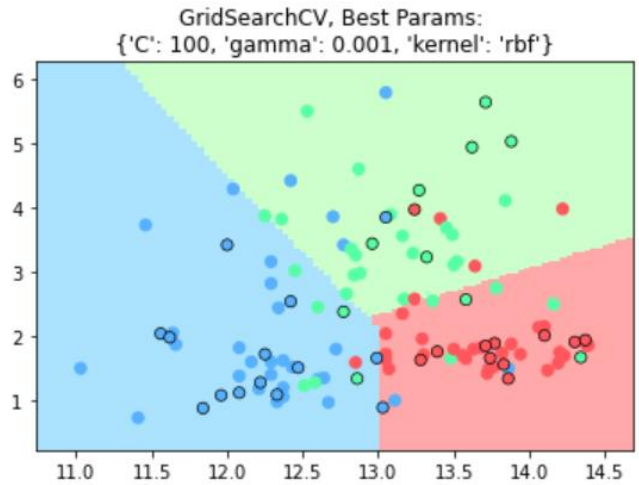
Gamma high means more curvature.

Gamma low means less curvature.

The chosen values are:

C = [0.0001, 0.001, 0.1, 1, 10, 100, 1000, 10000] and

Gamma = [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]



With an accuracy of 0.83333

The graph above is the graph of the model with the best parameters found during the GridSearch: C=100, Gamma=0.001.

The graph is like the linear one since gamma is very small.

The final step is to apply K Fold validation with k equal to 5 and perform GridSearch on both C and Gamma.

The results are:

First Fold: accuracy=0.611

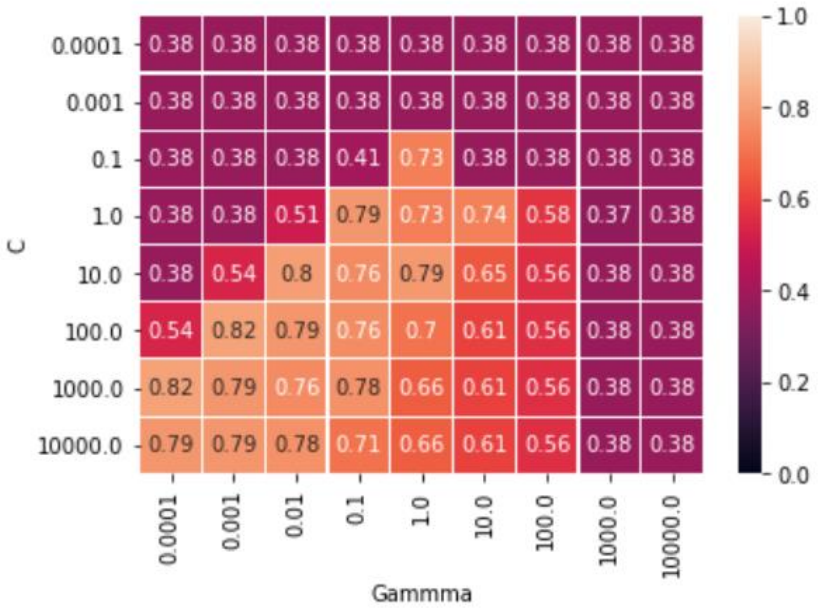
Second Fold: accuracy=0.667

Third Fold: accuracy=0.833

Fourth Fold: accuracy=0.941

Fifth Fold: accuracy=0.647

Mean=0.740



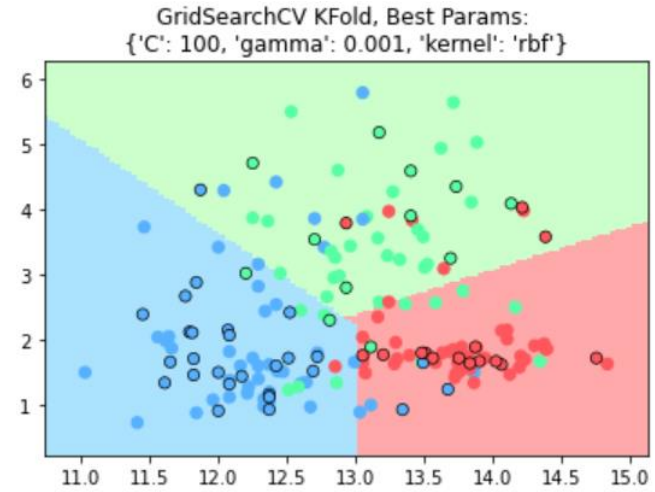
A heat map of the result obtained shows that:

When gamma is very small, the model is too constrained and cannot capture the complexity or “shape” of the data. The resulting model will behave similarly to a linear model with a set of hyperplanes that separate the centers of high density of any pair of two classes.

For intermediate values, we can see on the second plot that good models can be found on a diagonal of C and gamma.

In fact, the best score is obtained with C = [100, 1000] and Gamma = [0.0001, 0.001].

While the worst scores are obtained with C too low (similar to what happens on linear SVM) and Gamma too high.



The graph above is the graph of the model with the best parameters found during the GridSearch and K-Fold: C=100, Gamma=0.001.

The graph is like the linear one and similar to the graph of the model not using K-Fold validation.

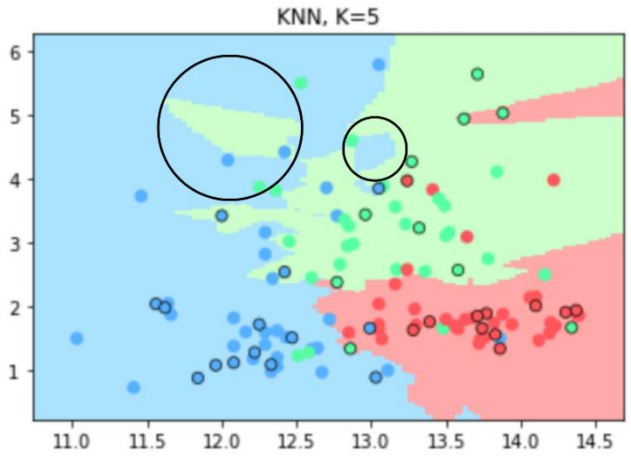
EXTRA

Point 19) Discuss the difference between KNN and SVM

Both KNN and SVM perform well on the test set, the differences are on how they work and how the build the boundaries.

In KNN, it determines the nearest k training instances to the test data. Figuring out which k are the nearest involves calculating some sort of distance function (usually Euclidian Distance). So, to the target is given a classification based only on the nearby instances, and anything farther away is ignored. This brings to a boundary representation which contains 'holes' in it, the boundaries are not continuous, because of the different neighbourhood that can be created.

So, in knn, you should consider every point in training set, and select the nearest points to compute the score.



k=5 accuracy=0.72222

Highlighted with black circles the 'holes'

While SVM, attempts to find a hyper-plane separating the different classes of the training instances, with the maximum error margin. Since it tries to find a hyper-plane this means that the space is continuous, without 'holes'. The most important training instances are the ones that are making up the boundary, called 'support' points. SVM allows to have some mis-classified data in order to obtain a higher margin, this is made with the regulation parameter C, higher is C, the more we penalize mis-classified data. So, it is a trade-off between how many points we allow to be mis-classified and how large the margin should be.

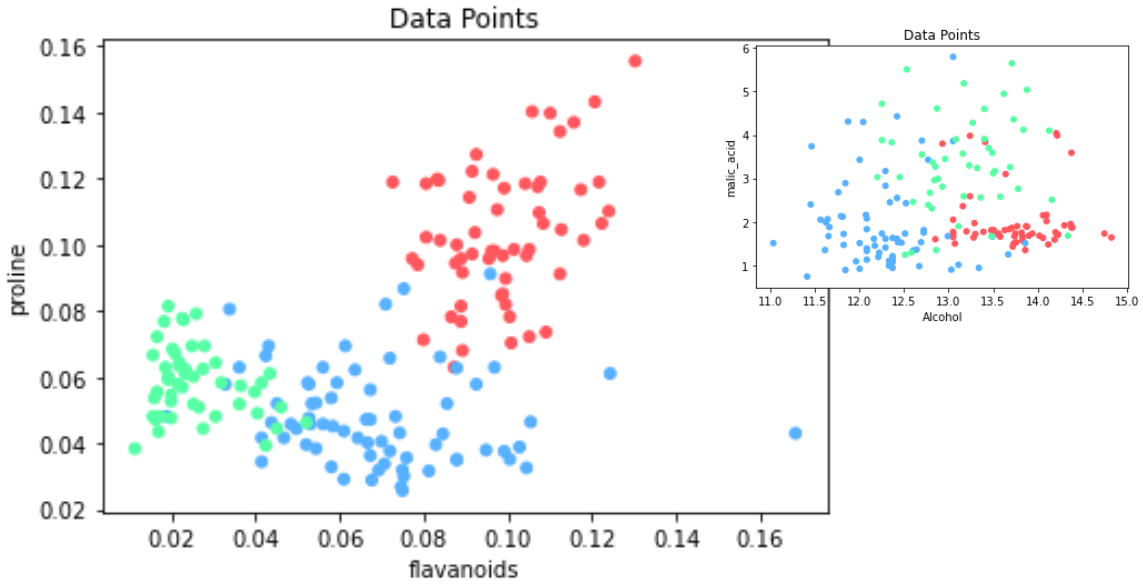
Point 20) Try also with different pairs of attributes

To select a different pair of attribute I used the method *SelectKBest()* from the *sklearn.feature_selection* library using k as 2, since the features of interest are 2, and as score function *f_classis* function based on ANOVA F-value method, since is appropriate for numerical inputs and categorical output, as in the wine dataset.

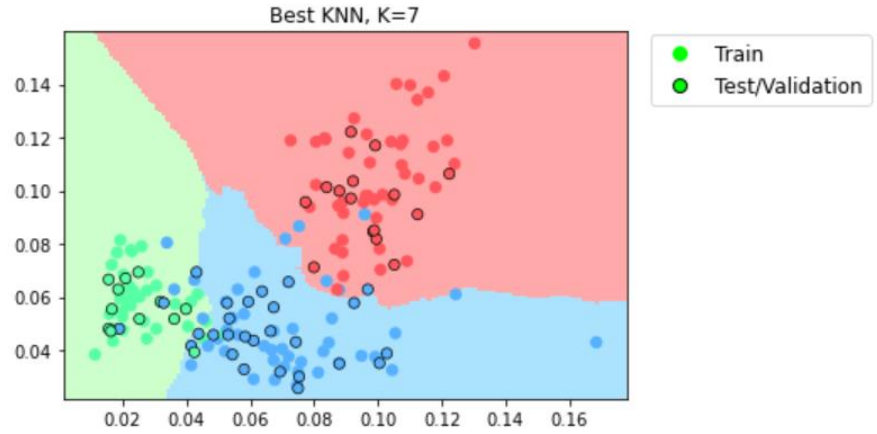
Here a table with the 4 highest score of the feature with respect to the output label

ColumnsIndex	ColumnsName	Score
6	flavanoids	233.925873
12	proline	207.920374
11	od280/od315_of_diluted_wines	189.972321
0	alcohol	135.077624

So, the two chosen features are **flavanoids** and **proline**. Since proline value are too large (min=278, max=1680), I normalized the data using the *normalize* method from the *sklearn.preprocessing* library.



As we can see, the classes are more distinguishable (less points of a class are near a group of other points), so we aspect to have a higher score.




```
Test with k=7 accuracy=0.88889 (Best Validation=0.88889)
```

This is the plot with the best k score found during the training of the KNN classifier. As expected, the score is higher than score with be previous features (previous best k score=0.85185).

The same happens with SVM, both with linear and rbf kernel.

With a score of: Linear: k=100 accuracy=0.88889 (previous: 0.81481)

RBF: C=1000 accuracy=0.87037 (previous: 0.85185)

Some interesting facts are:

- The KFold validation (k=5) obtain a mean score of 0.921;
- The GridSearchCV with KFold validation the classes boundaries are really ‘close’ (especially the green class), and this may bring bad performance even on a not too different data.