Homework 1

Marco Di Nepi 277959

The wine datased is composed by 178 samples and 13 attributes. For the purpose of this homework, only the first two attributes are selected, which are *Alcohol* and *Malic_Acid.*
The target variable includes three possible classes, each corresponding to a different cultivation.

Once loaded, data is randomly split into three parts:

- Training set (50% of samples)
- Validation set (20%)
- Test Set (30%)

This is made exploiting the train_test_split function from sklearn.

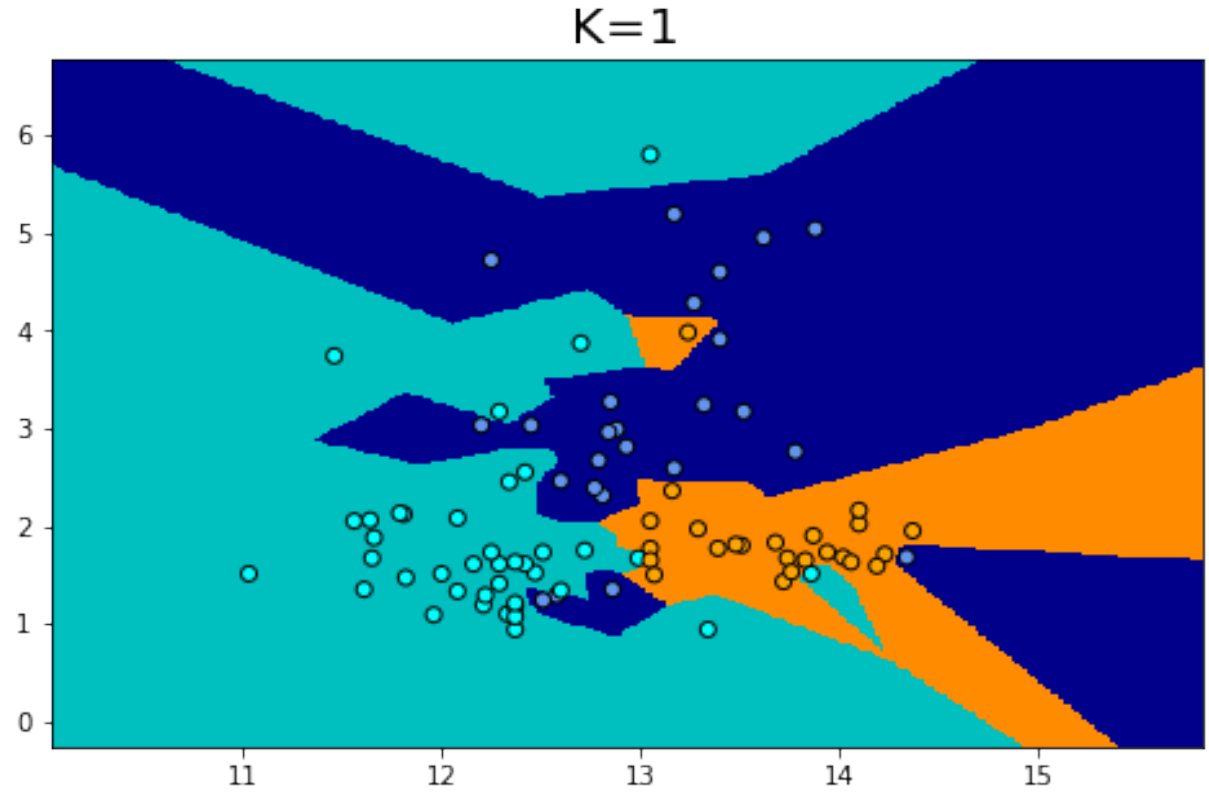
To avoid different results at each run, a good solution may be to fix a random_state for the split and the svm models. This is needed because the dataset is small and the accuracy of the models is highly dependent on the initial split. For this report, it has been used random_state = 12.

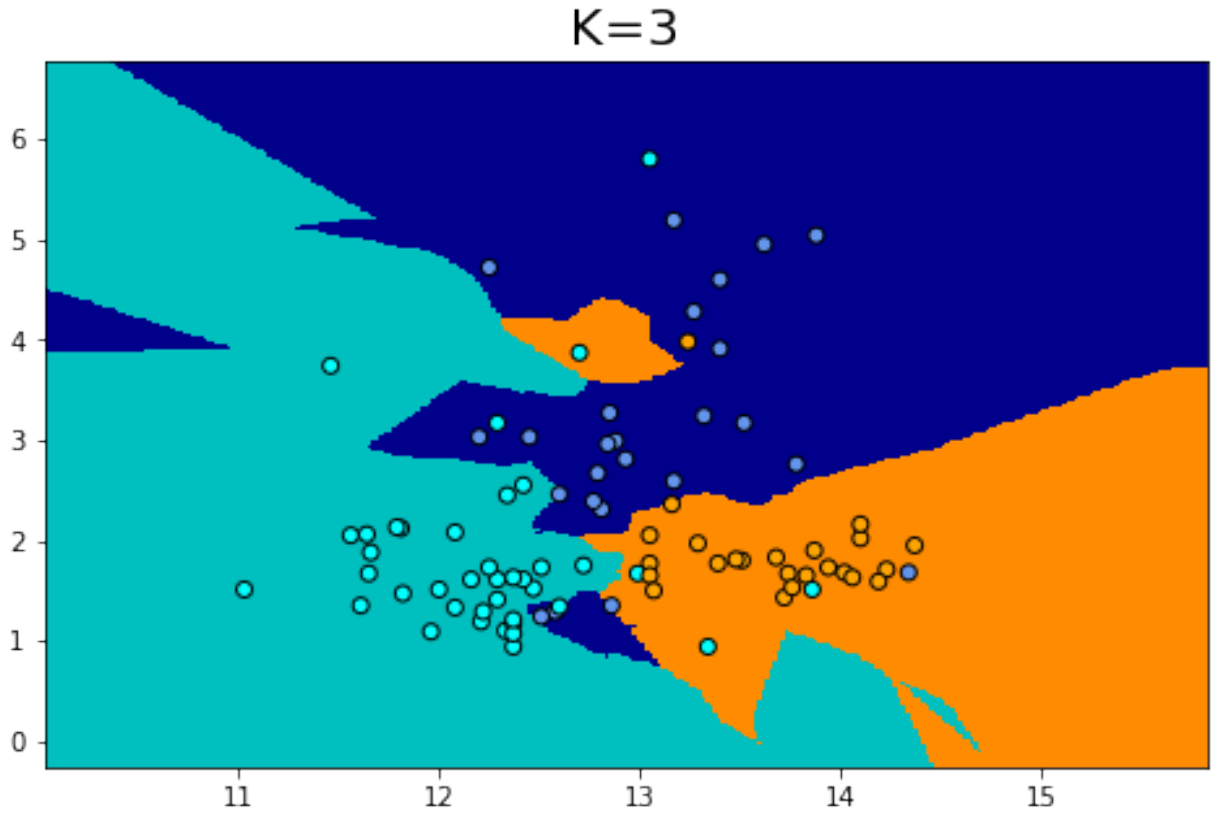
K-Nearest Neighbors

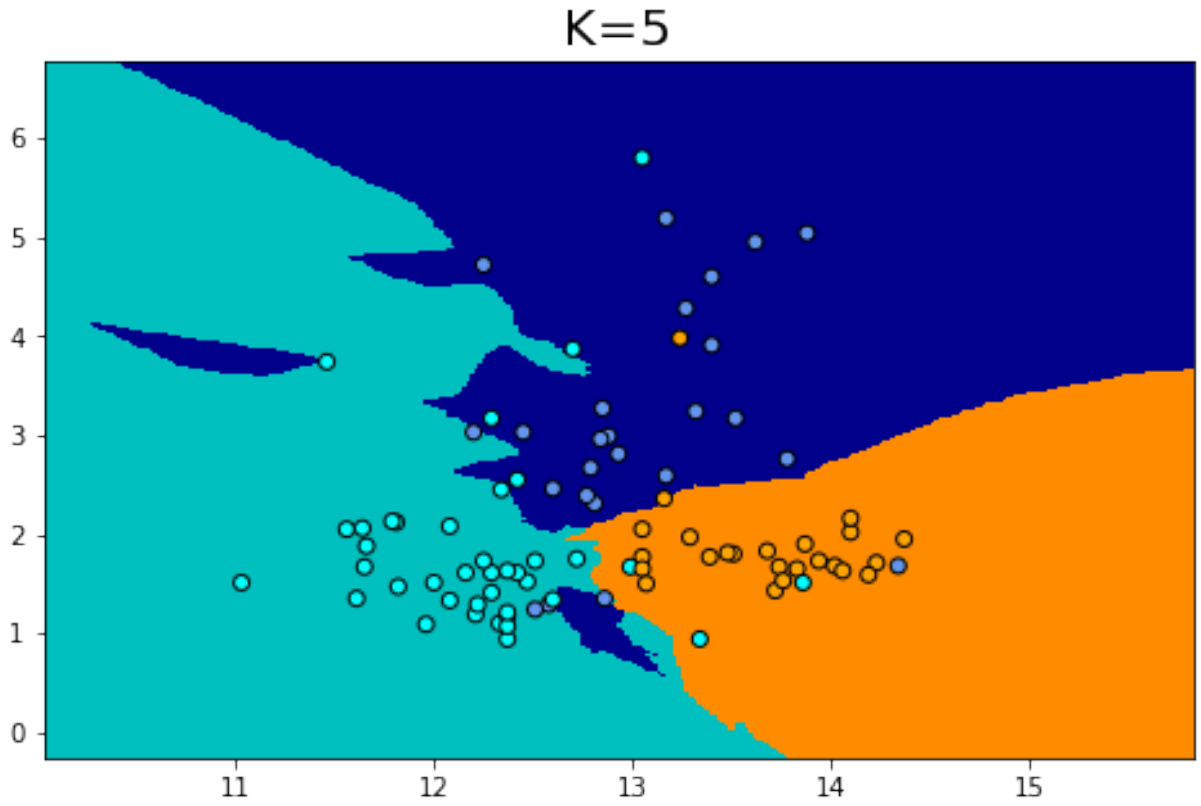
The first classification algorithm used is k-NN. The main idea behind this algorithm is not building and internal general model but storing all the training data with their label and for each query point compute the distances from all the other points. Then the k nearest neighbors are considered and the label is assigned by majority voting.
The strong point of K-Nearest Neighbors is that is very easy to implement and to use. On the other hand, it can be very expensive to compute all the distances for large dataset and there may be memory issues.

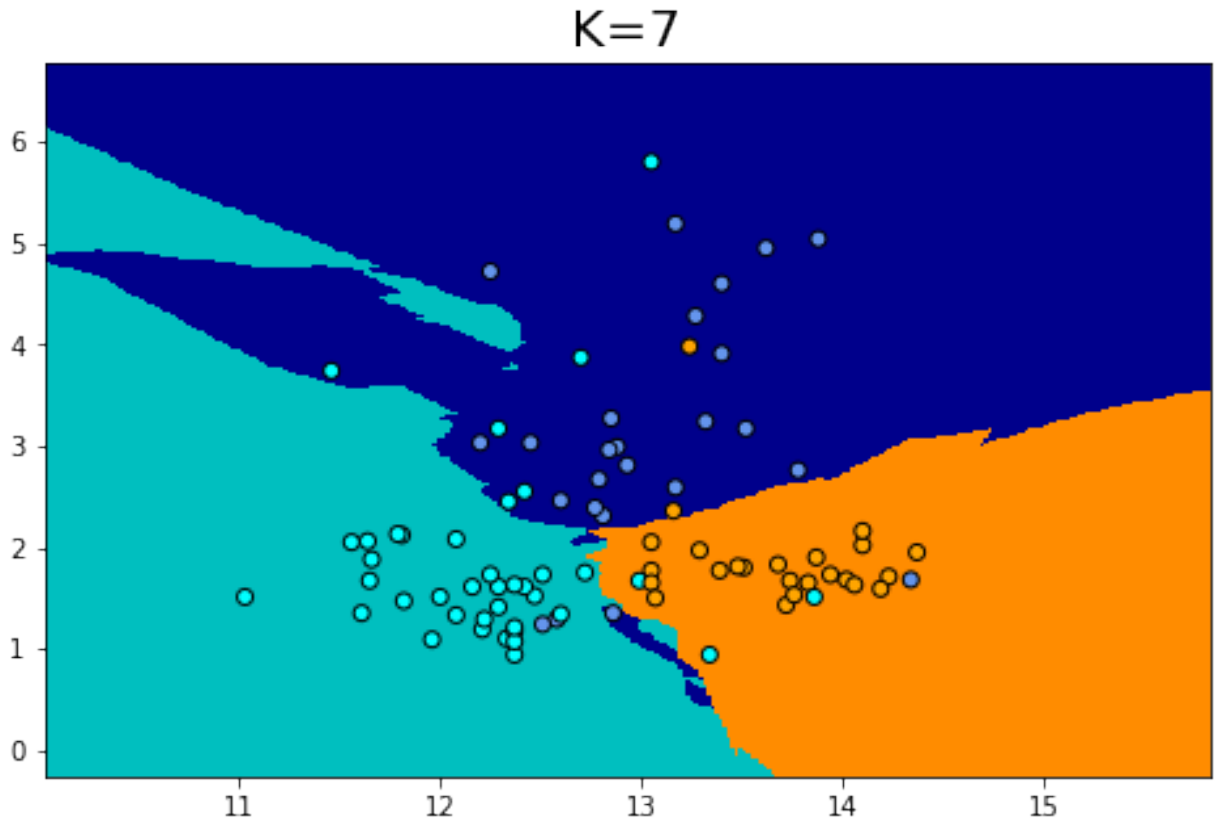
Given K = [1,3,5,7] the possible values for the hyperparameter k, the following steps for each k have been performed:

1. Create the model using the sklearn's KneighborsClassifier
2. train the model using the training set
3. Plot the points and the decision bondaries
4. predict the labels for the validation set
5. compute the accuracy score between y_train and the predicted values, where accuracy is defined as $1/n_{samples} \sum 1(y_i = \hat{y}_i)$


K=1
6
5
4
3
2
1
0
11
12
13
14
15


K=3
6
5
4
3
2
1
0
11
12
13
14
15

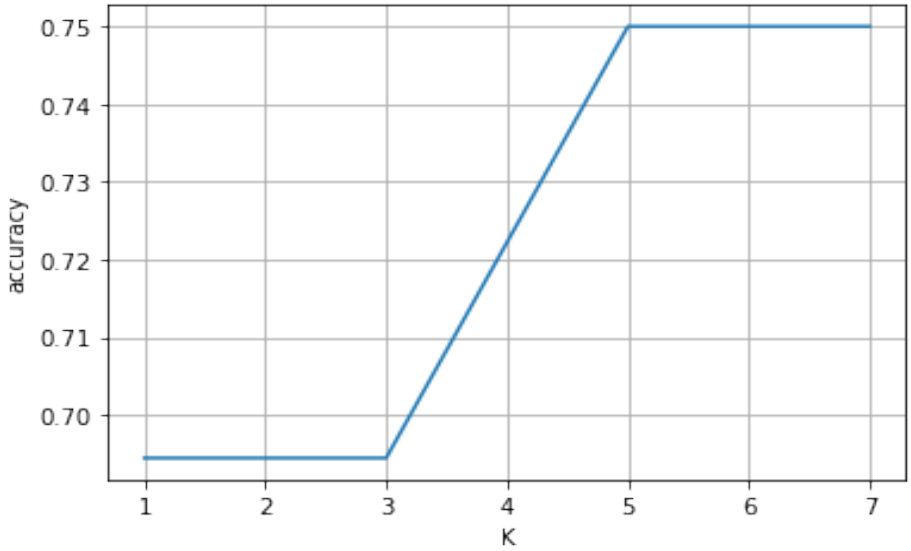

K=5
6
5
4
3
2
1
0
11
12
13
14
15


K=7
6
5
4
3
2
1
0
11
12
13
14
15

All the graphs are made using the Matplotlib library. Useful functions to plot the decision boundaries are pcolormesh and meshgrid from numpy.

In this case the results are:

K	Accuracy
1	0,72
3	0,72
5	0,77
7	0,77



Using the best value of K we can now train a new model with both training and evaluation set and evaluate it on the test set. The final accuracy is **0.796,** slightly greater than the one with k = 5 using only training data.
It can be also noticed that the boundaries are different for a different k. Choosing a small k means that the classification may be more sensitive to outliers, in fact there are well isolated points with their own area. With a bigger k there will be smoother decision boundaries, but it's not a good idea making it too big otherwise neighbors that are far apart (maybe irrelevant) will be taken into account. It may happen that the neighborhood is composed by two classes with the same number of points. When there is a tie like this, the classifier chooses the first class following the order in the dataset.

Linear SVM

A support vector machine (svm) tries to build one (or more) hyper-plane to separate the distributions of data. In the case of a linear svm we look for a linear separator that put the largest margin among the points. To do this, an optimization problem is solved. When linear separation is impossible, slack variables are introduced. This means that we are allowing for some mistakes.

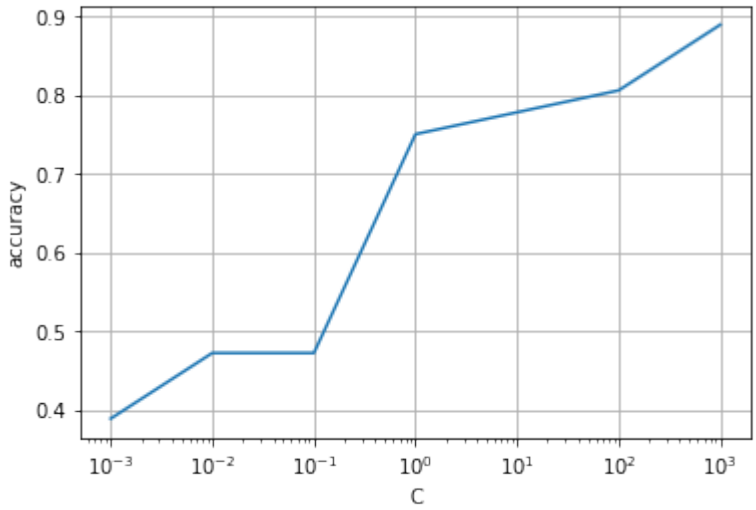
$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$
$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, \ldots, n$$

The hyperparameter C is the cost of misclassification. When C is high the algorithm aims to maximize the number of points correctly classified but the margin will be smaller, while a smaller C will encourage a larger margin, therefore a simpler decision function, at the cost of training accuracy.
It can happen that when C is very small, the margins are so large that one of the classes is not taken into account.
The advantages of support vector machines are:

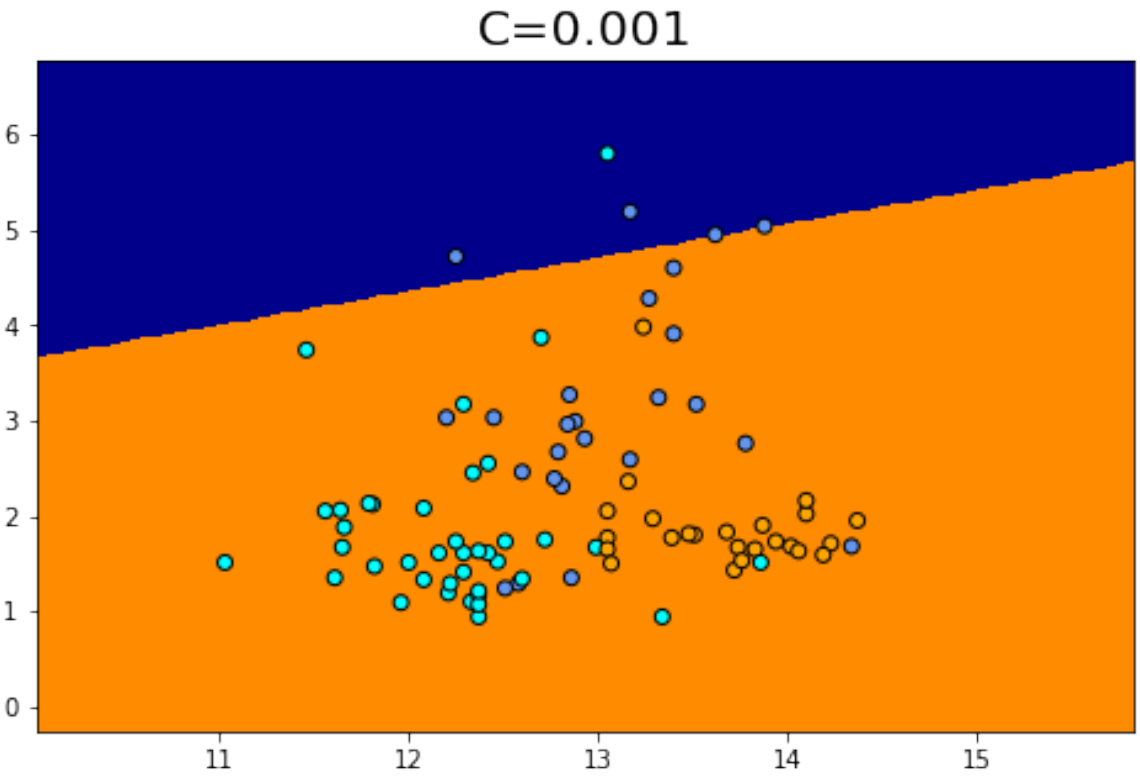
- Effective even in high dimensional spaces
- Memory efficient since the solution is made of a subset of training points (the support vectors)
- Finds the optimum in the separation problem

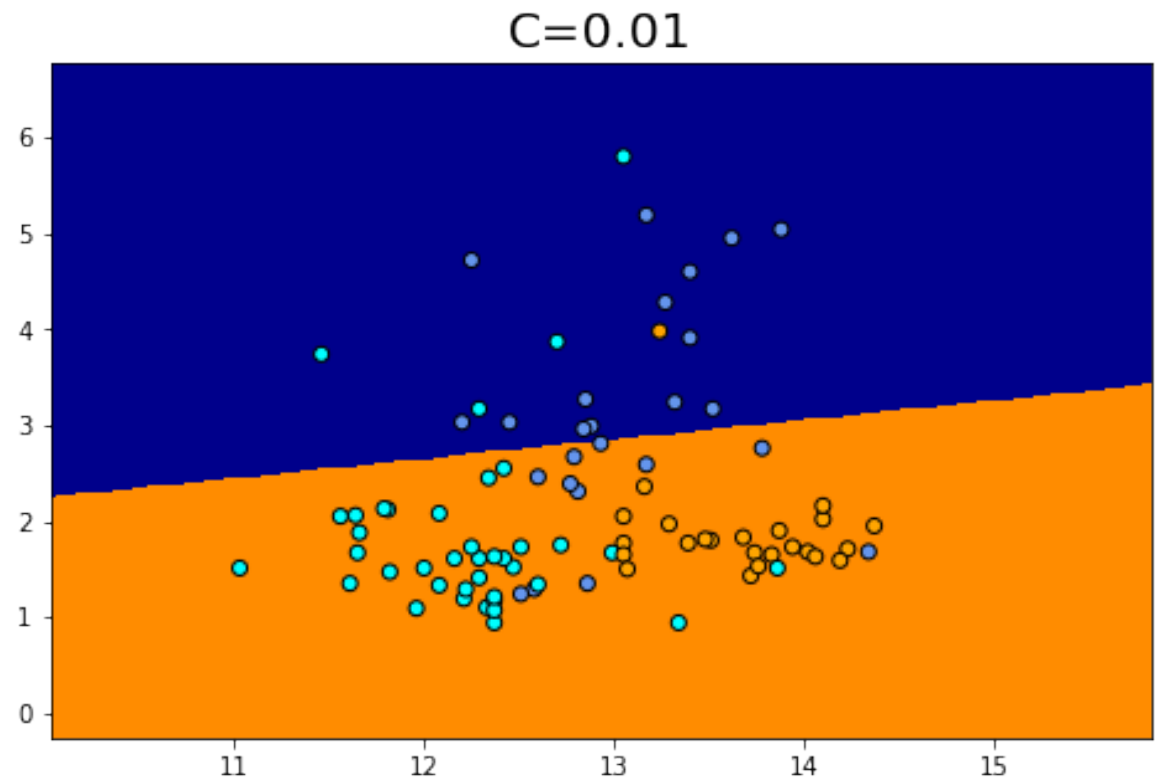
To find the best value for C, the same steps as before are performed. In this case we use the LinearSVC classifier insted of SVC with linear kernel. LinearSVC implements a one-vs-rest strategy for multi-class classification.

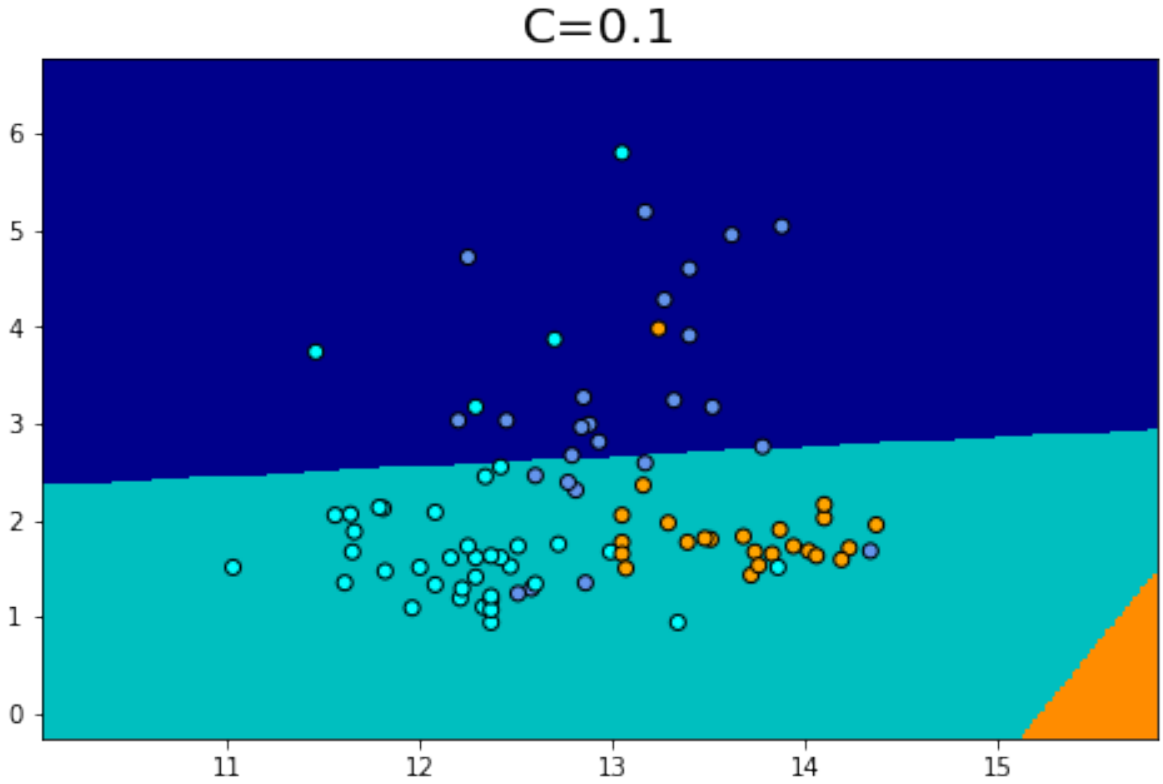
C	Accuracy
0,001	0,38
0,01	0,47
0,1	0,47
1	0,75
10	0,77
100	0,8
1000	0,88

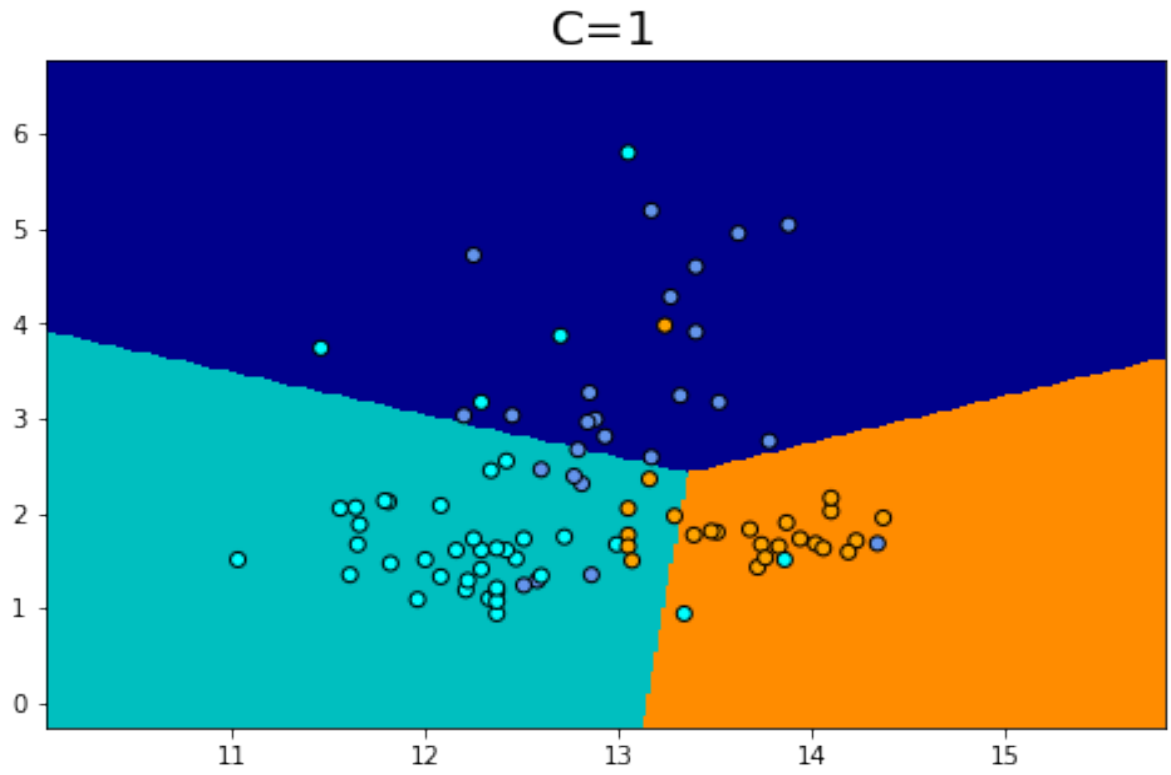


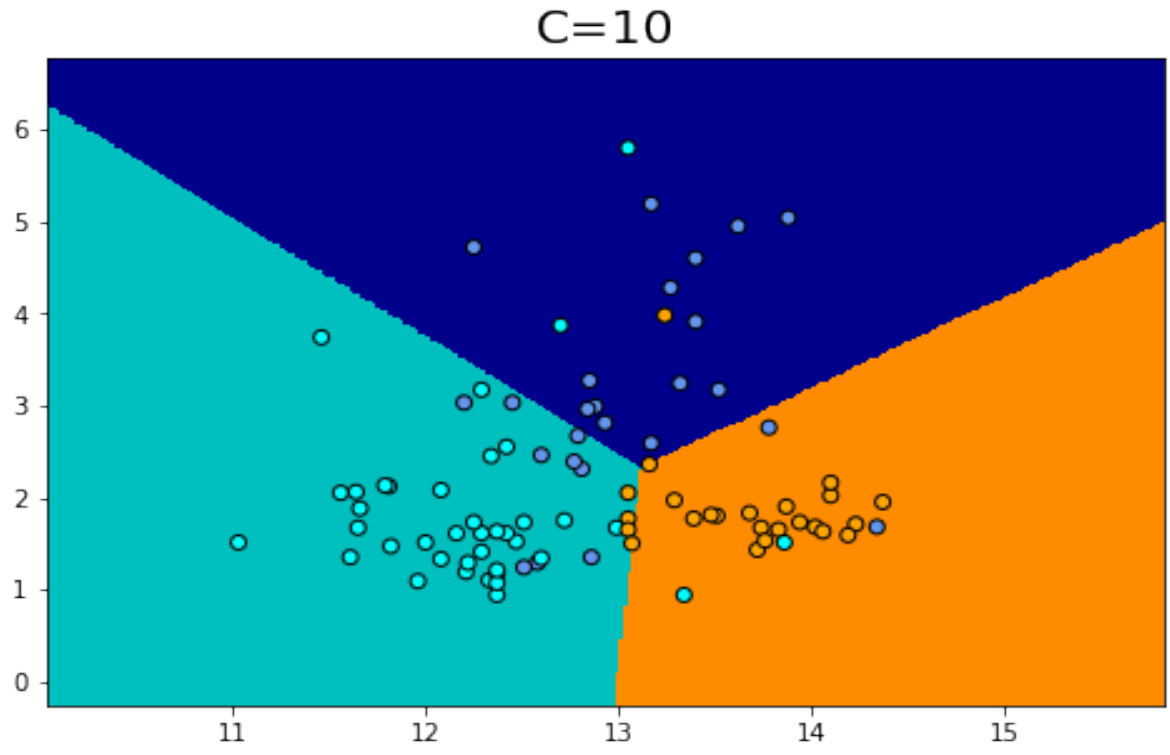
The best value for this split is C = 1000.
Evaluating the accuracy on the test test, it gives 0,759.
A linear svm seems to work worse than knn. Probably the problem is not easily separable and the algorithm makes some mistakes.

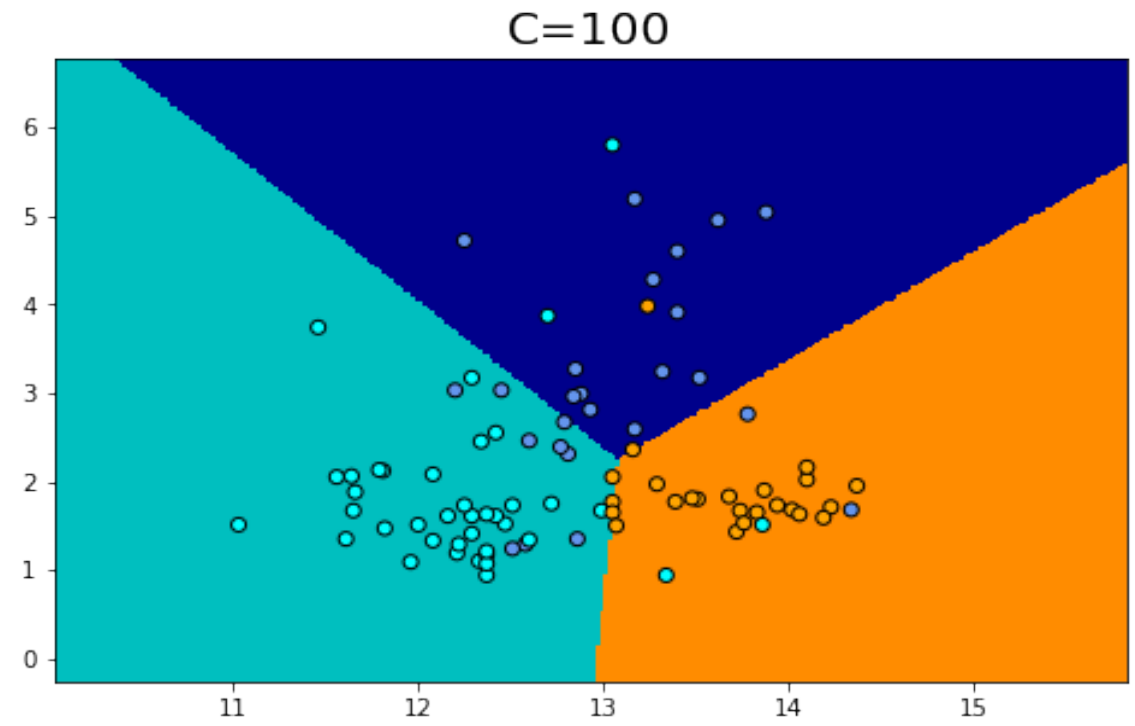


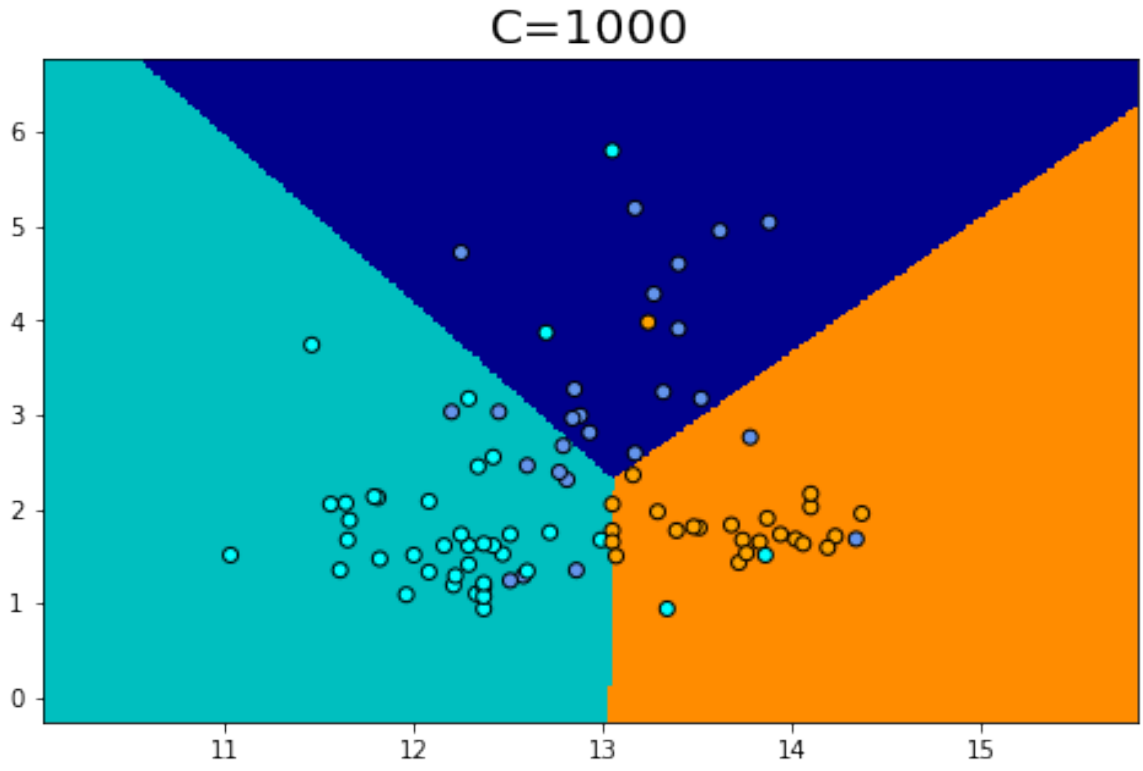

C=0.01
6
5
4
3
2
1
0
11
12
13
14
15


C=0.1
6
5
4
3
2
1
0
11
12
13
14
15


C=1
6
5
4
3
2
1
0
11
12
13
14
15


C=10
6
5
4
3
2
1
0
11
12
13
14
15


C=100
6
5
4
3
2
1
0
11
12
13
14
15


C=1000
6
5
4
3
2
1
0
11
12
13
14
15

RBF KERNEL

Support Vector Machines are capable of operating with kernels. Through kernel functions a linear model can be turned into a non-linear model. It's not necessary to compute the coordinates in the new higher dimensional feature space since the strongest point of kernels is that they are based on computing the inner products.

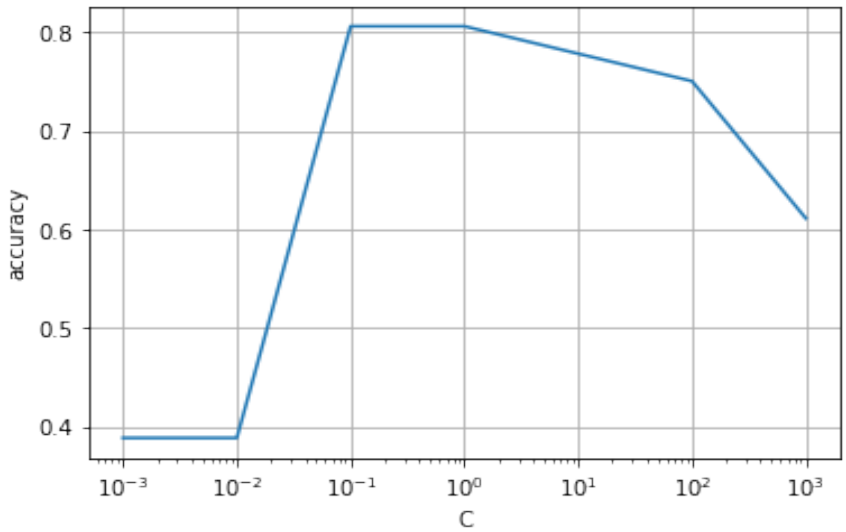
For the rbf kernel the expression is $\exp(\gamma \|x - x'\|^2)$

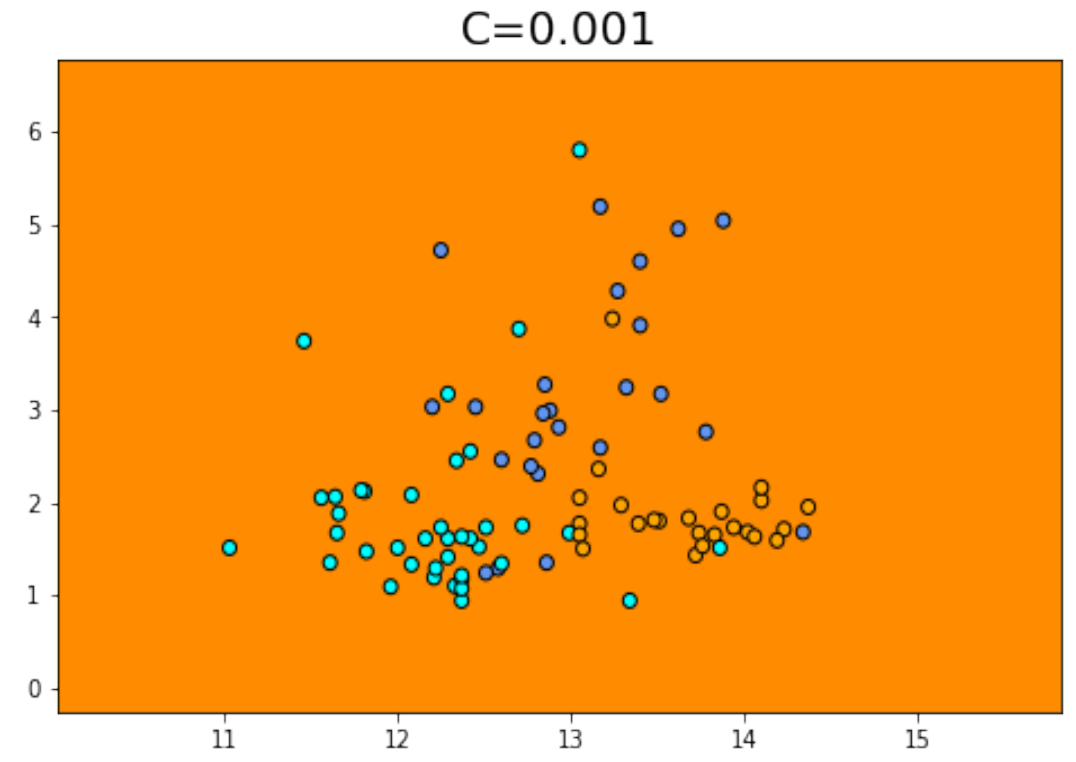
The gamma parameter is inversely proportional to the variance of the Gaussian. It defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’.

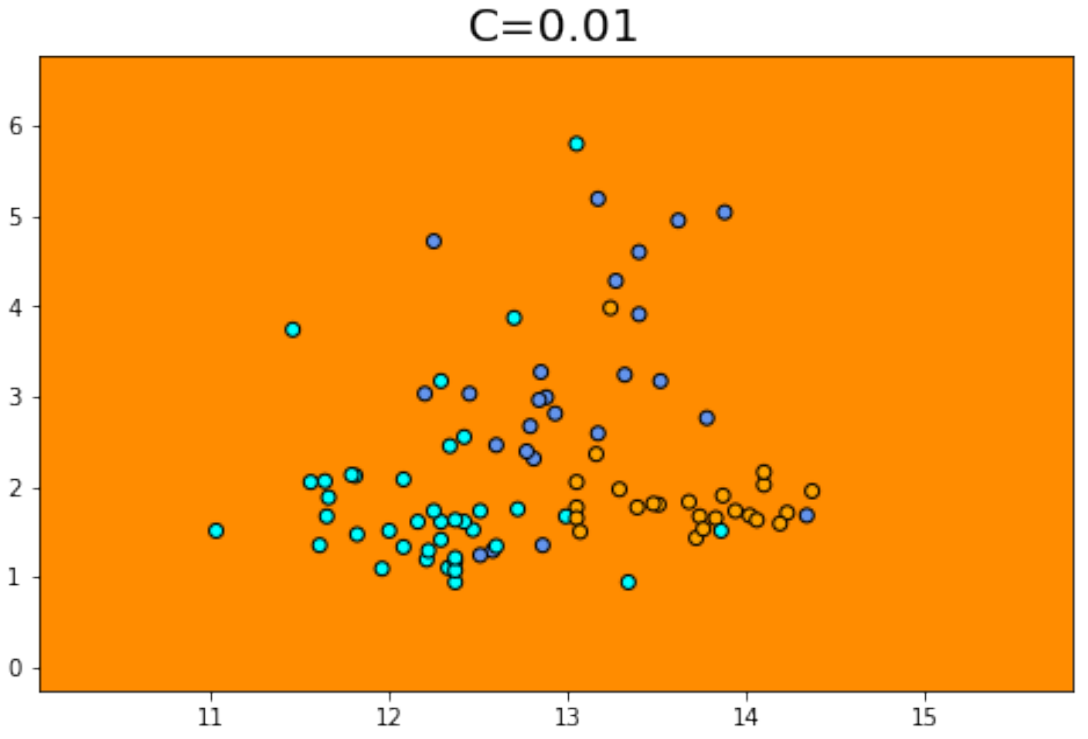
For this first phase we will use the SVC with kernel = 'rbf' and gamma = 'scale' that is the default value.
Repeating again the procedure we get

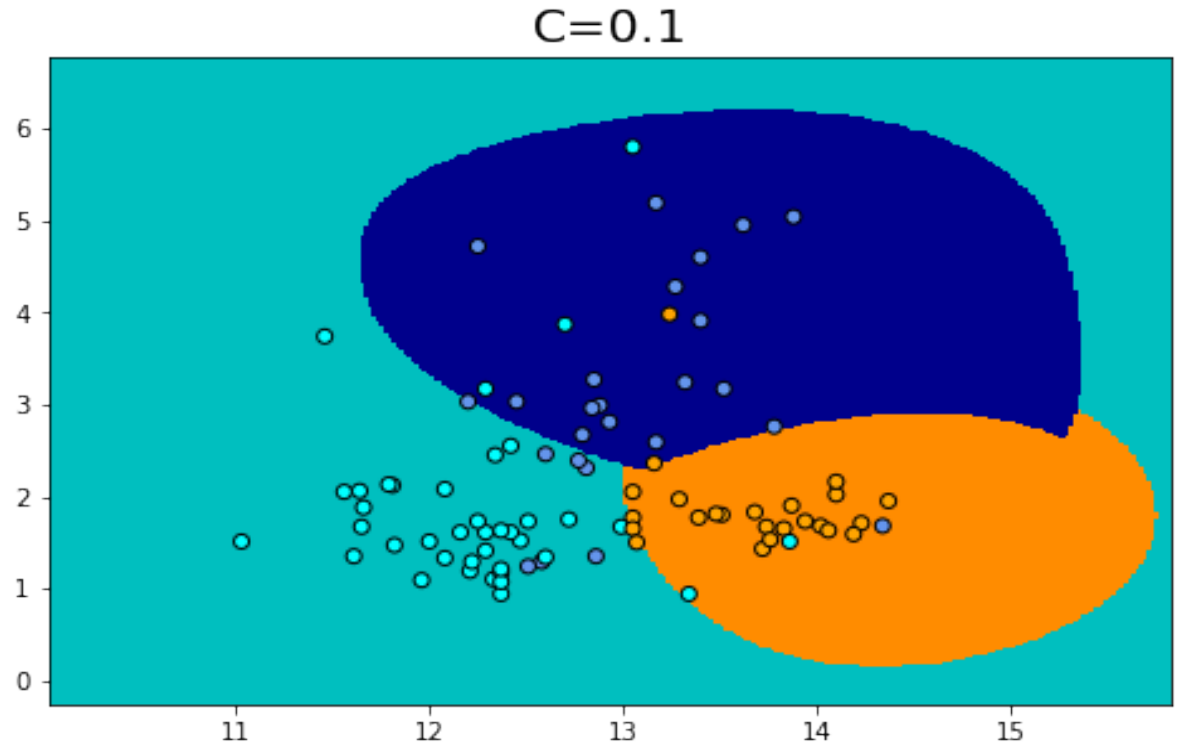
C	Accuracy
0,001	0,38
0,01	0,38
0,1	0,805
1,00	0,81
10	0,77
100	0,75
1000	0,61

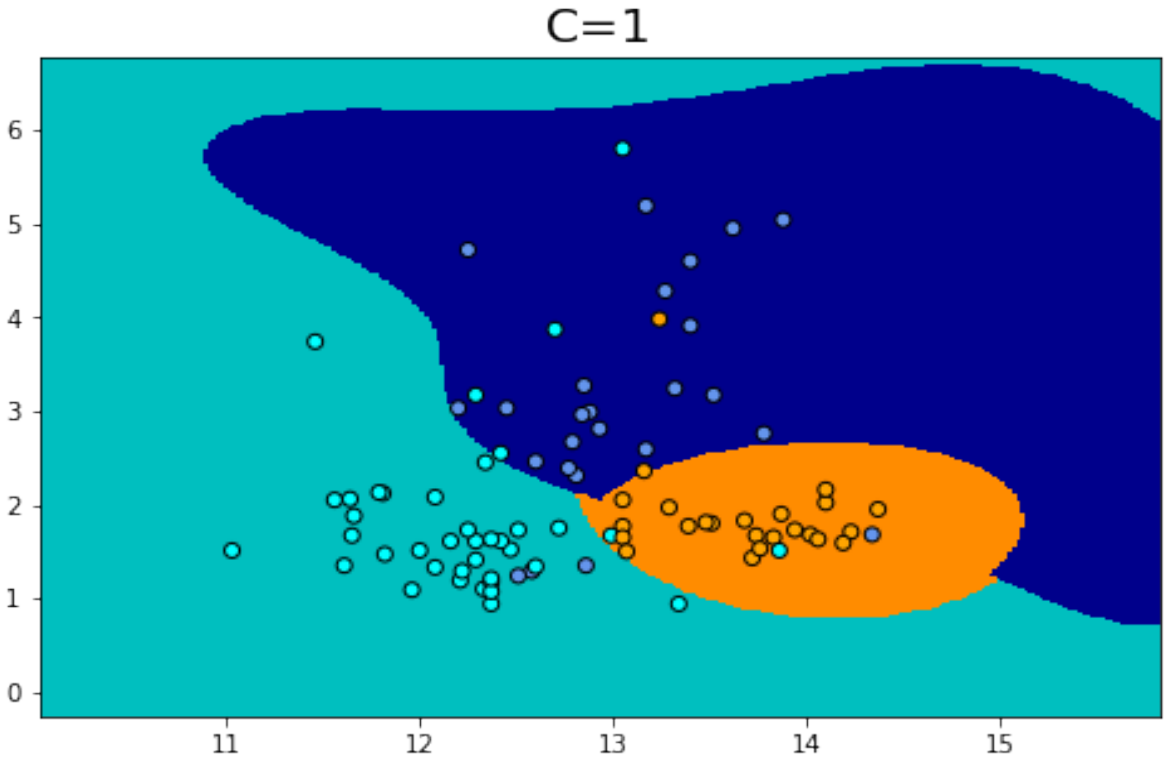
Evaluating the accuracy on the test test witch C=0,1, it gives 0.796.
With the rbf kernel the results are much better. Since the separation is no more linear, the decision boundaries have very different shapes and the larger is C the more these will change to better classify all the points. Instead, with a very small value of C, the accuracy is very low because every point is assigned to the same class.

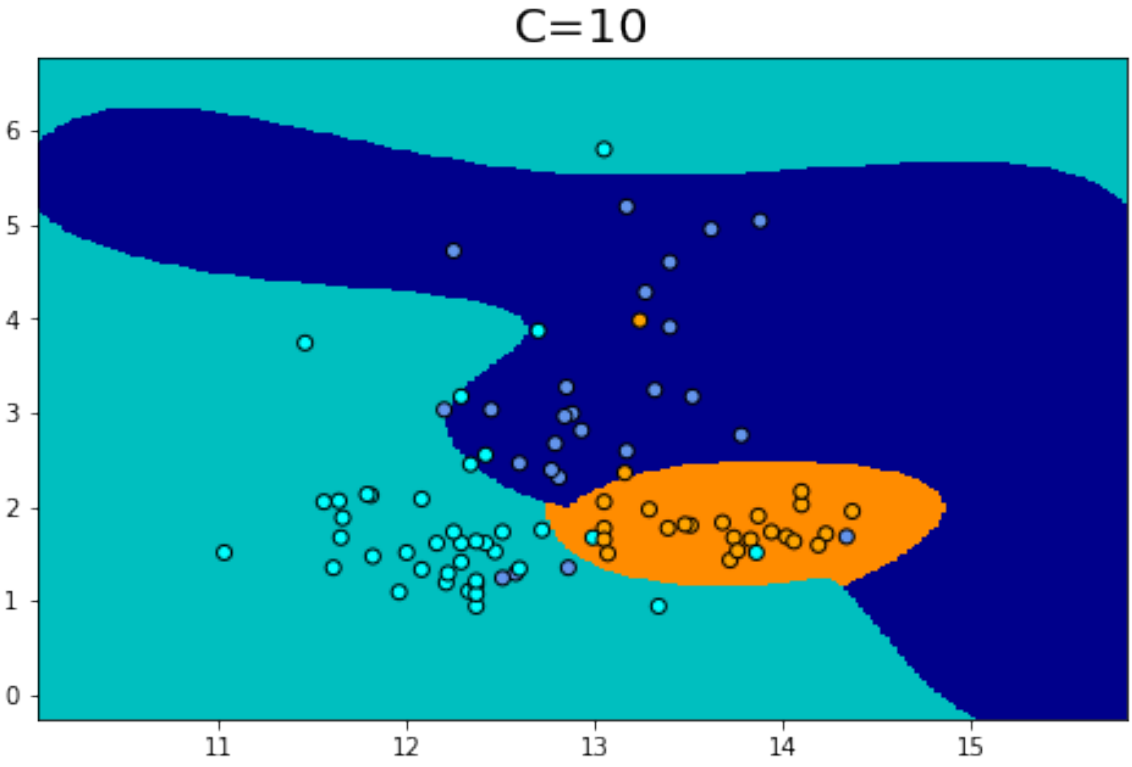


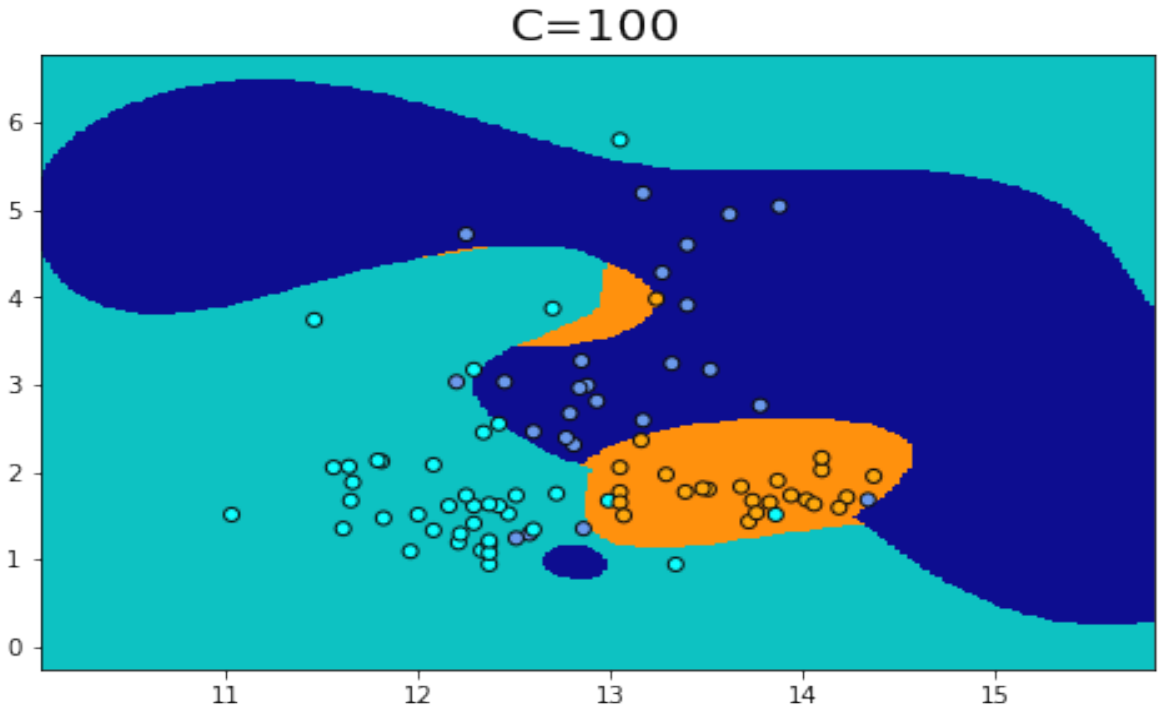

C=0.001
6
5
4
3
2
1
0
11
12
13
14
15

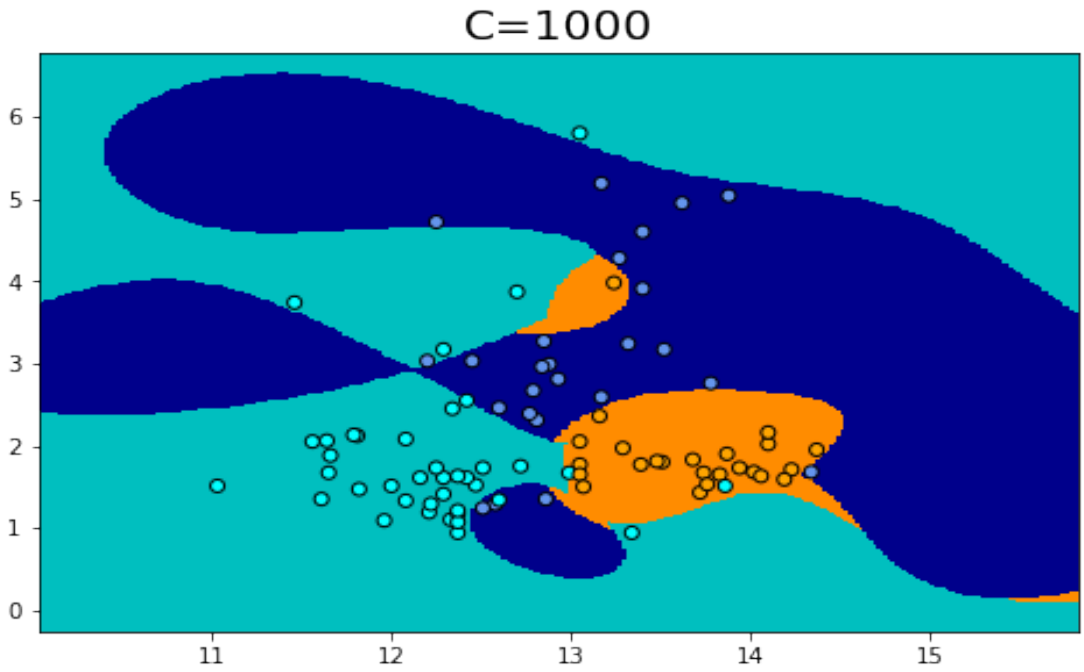

C=0.01
6
5
4
3
2
1
0
11
12
13
14
15


C=0.1
6
5
4
3
2
1
0
11
12
13
14
15

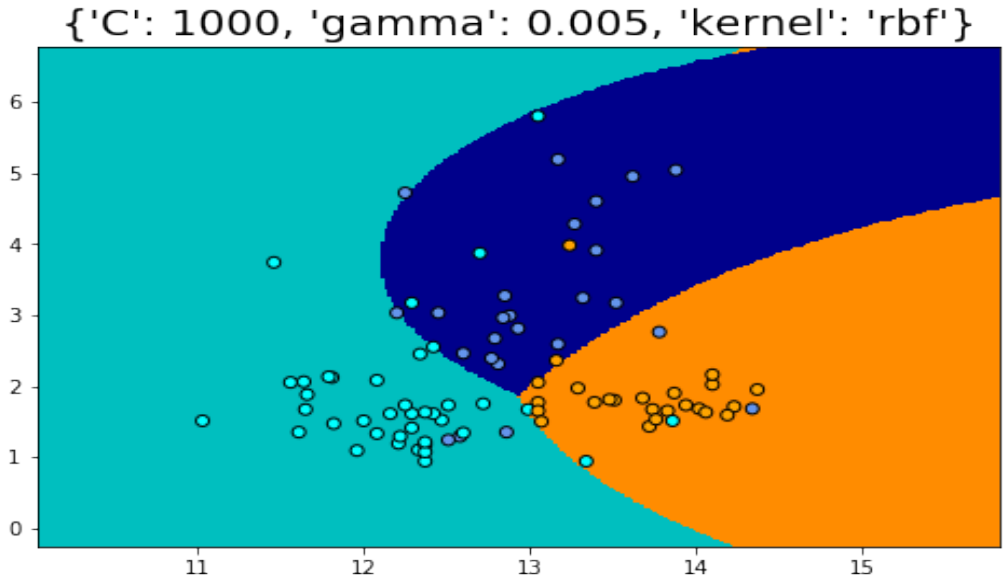

C=1
6
5
4
3
2
1
0
11
12
13
14
15


C=10
6
5
4
3
2
1
0
11
12
13
14
15


C=100
6
5
4
3
2
1
0
11
12
13
14
15



To find the best values for C and gamma at the same time a grid search can be exploited. We try for each combination of the values and we choose the pair wich gives the best accuracy on the evaluation set. Hence, the final model is evaluated on the test test. In order to achieve this task it's enough to iterate for each configuration returned by the function ParameterGrid. The ranges used for the hyperparameters are:
C = [0.001, 0.01, 0.1, 1, 10, 100,1000]
gamma = [0.0001,0.001,0.005,0.01,0.05,0.1,0.5,1].
The best configuration obtained is {'C': 1000, 'gamma': 0.005,} and the accuracy score on the test set using this model is 0,759. It can be noticed that this is not the highest achievable score and that is lower than the previous one.



K-Fold

Another different approach that can be used to find a more precise model is k-fold.
Instead of splitting in evaluation set and training set, we divide the data in k=5 parts. Next, we train on 4 parts and then we evaluate the accuracy on the remaining part. This process is repeated 5 times, each with a different "evaluation set". The final accuracy is obtained computing the average between the five results. This method should return a less biased and more accurate model since it ensures that every point in the original dataset (excluding test points) appears in both training and test set.
Keeping the same ranges for gamma and C we obtain that the best configuration is {'C': 10, 'gamma': 0.05 }. The final score on the test set is 0,77, better than the one without k-fold but still lower than the one obtained with the default value of gamma.

SVM vs KNN

In conclusion, KNN works well, it can detect linear or non-linear data and it's a good choice in a low dimensional space. On the other hand, it is very sensitive to outliers. SVM has kernels and tends to be good also with outliers and in a high dimensional space but the tuning of hyperparameters is a critical point.

Experiments with different features

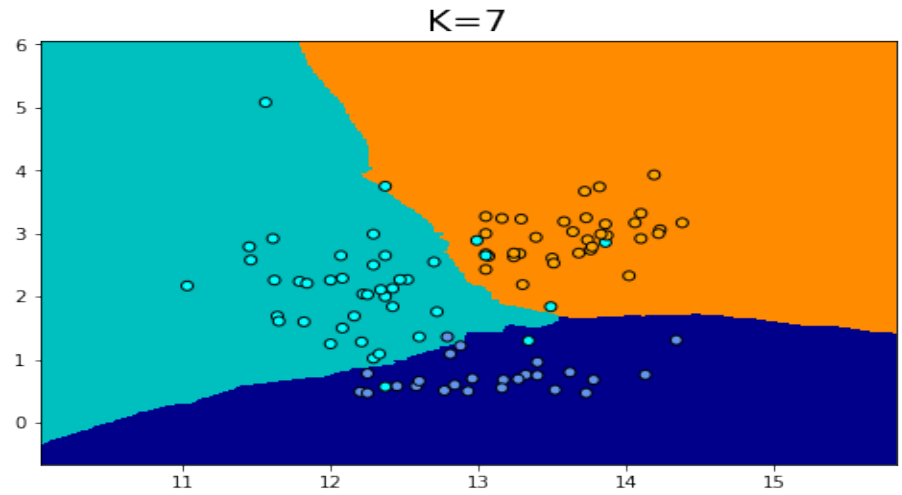
Finally, I report here the results of k-fold for svm with rbf kernel using different pairs of attributes.

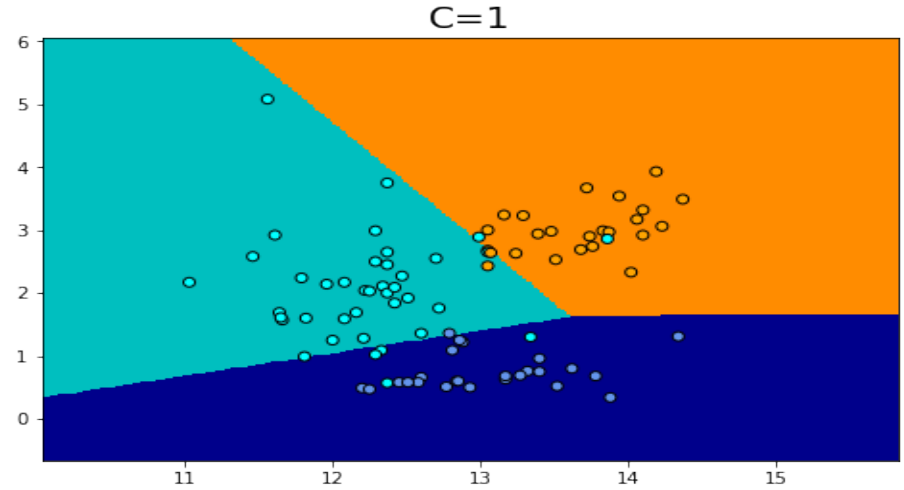
- Alcohol and Hue: 0.88
- Flavanoids Proanthocyanins: 0.722
- Alcohol Flavanoids 0.94

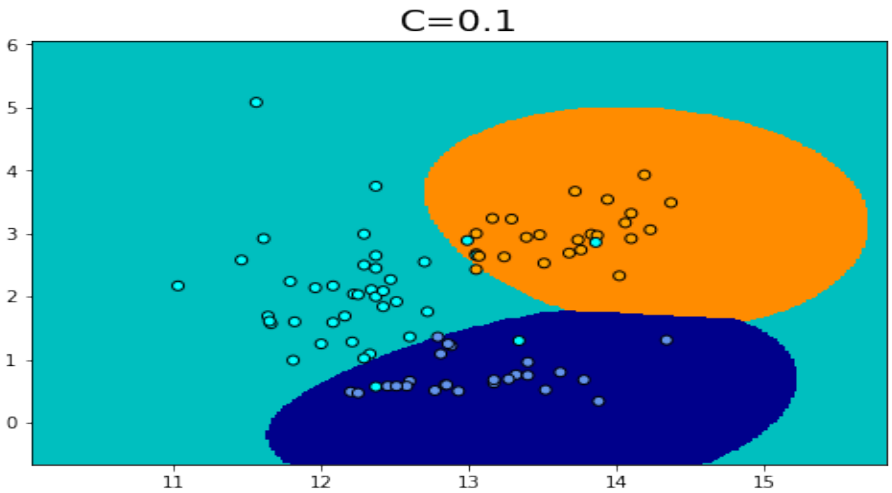
I found Alcohol and Flavanoids to be the best pair for classification.
The accuracies obtained with the different methods are:

- Linear Svc: 0.92
- Svc with rbf kernel: 0.925
- knn : 0.87

As we can see, the classification is better if the features are relevant and better characterize the data. The figures below show the decision boundaries for this last case and as we can see, they are able to separate points in a very good way.


K=7
6
5
4
3
2
1
0
11
12
13
14
15


C=1
6
5
4
3
2
1
0
11
12
13
14
15


C=0.1
6
5
4
3
2
1
0
11
12
13
14
15