Machine Learning and Deep Learning

Politecnico di Torino

Homework 1 Report
Student ID: s275090

Topic: Nearest Neighbors, Linear SVM, SVM with RBF Kernel

In this homework I tried to use two different supervised machine learning algorithms on the Wine dataset of scikit library to evaluate their performances on this data.

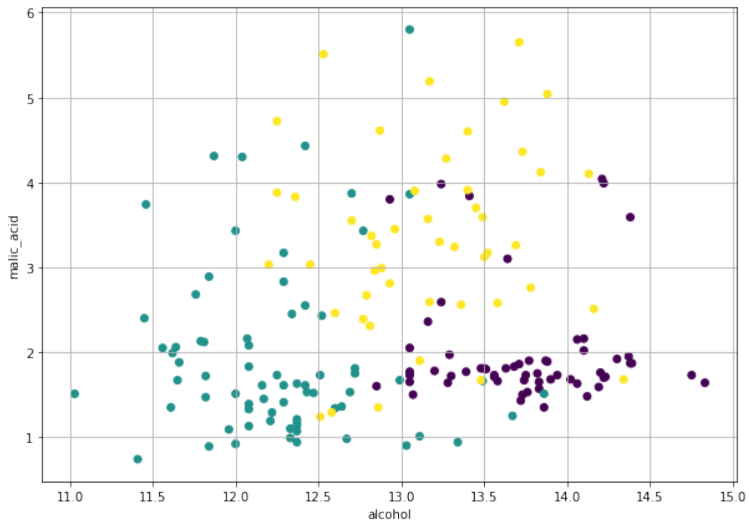
The k-nearest neighbors algorithm is a simple supervised machine learning algorithm that can be used to solve classification and regression problems. The principle behind it is to find a predefined number of training samples (K) closest in distance to the new point and predict the label from these.

The Linear SVM is a supervised learning model for linear classification and regression problems. The idea is to create a line which separates the data into classes. With RBF Kernel the algorithm finds a hyperplane for a non-linear classification, where implicitly mapping inputs into high-dimensional feature spaces.

The Wine Dataset

The wine dataset is a classic multi-class classification dataset. It is the result of a chemical analyse of wines grown in the same region in Italy but derived from three different cultivars. The analyse determined the quantities of 13 constituents.

So, we have 3 classes, the different cultivars, with a total of 178 samples (59, 71, 48). Every sample has 13 numeric variables for each constituent of wine. For this work we select only the first two attributes, alcohol and malic acid, and plot a 2D representation.



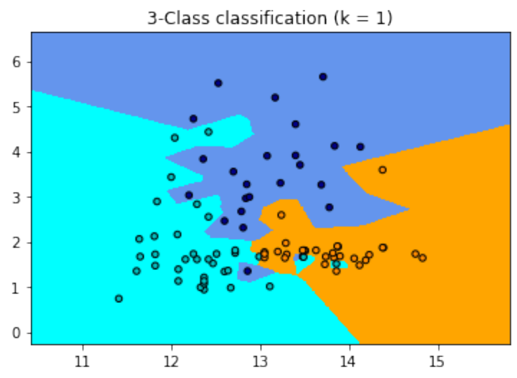
We already can see a first separation of the three classes with only these two attributes.

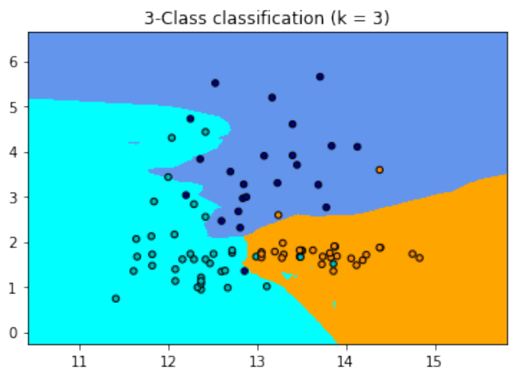
To build and test a good model we randomly split data into train, validation and test sets in proportion 5:2:3.

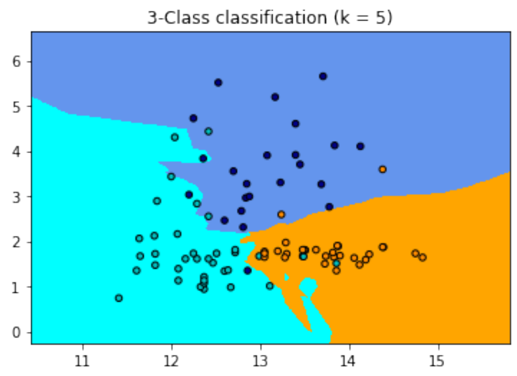
In this part I chosen to ensure that the sets have approximately the same of percentage of samples of each target class as the complete set because the dataset is very small and there is the risk to exclude one of three classes during the train or test part.

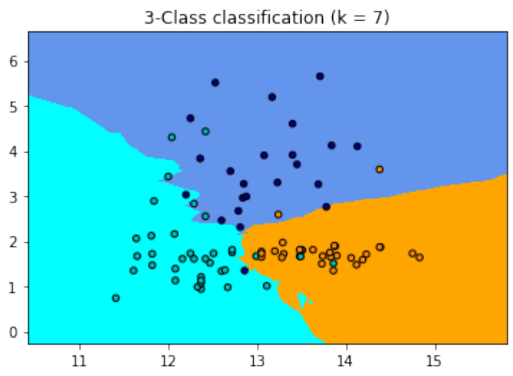
K-Nearest Neighbors Algorithm

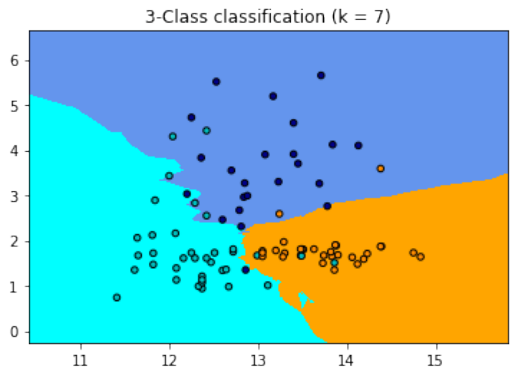
The first supervised machine learning algorithm that I want to apply is K-nearest Neighbors Alg. with different value of K: [1,3,5,7] (the number of closest data points). For each K I train the model on the train set, plot the decision boundary and test the model on valuation set.



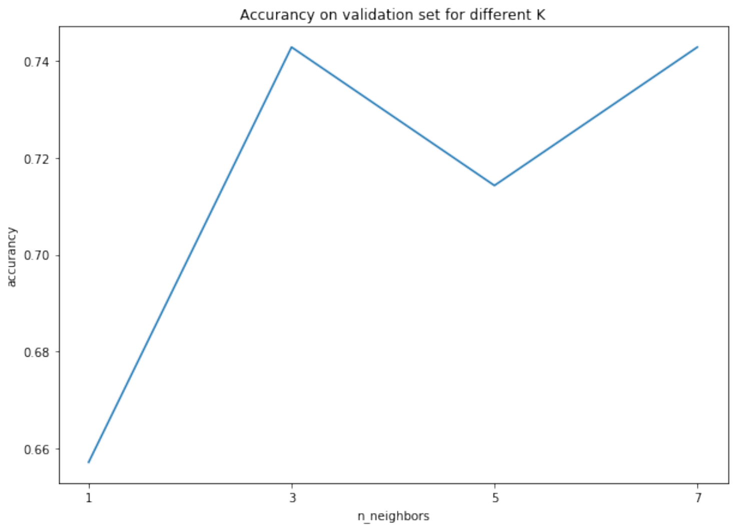








We can see from the graphs above the decision boundary about 3 classes. For a k too small the regions are fragmented while for k = 7 is clearer the division. To select the best parameter K, I calculated the accuracy of prediction on the valuation set when changing K.



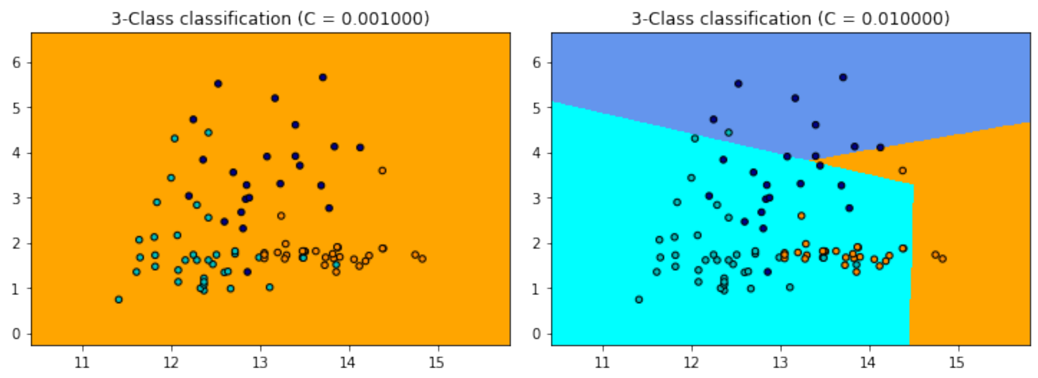
From the graph, for K = 3,7 we have an accuracy of .74 on the val set. We use the best value of K and evaluate the model on the test set. We obtain an accuracy of .78, in line with the value obtained on the valuation set. So, the model predicts correctly.

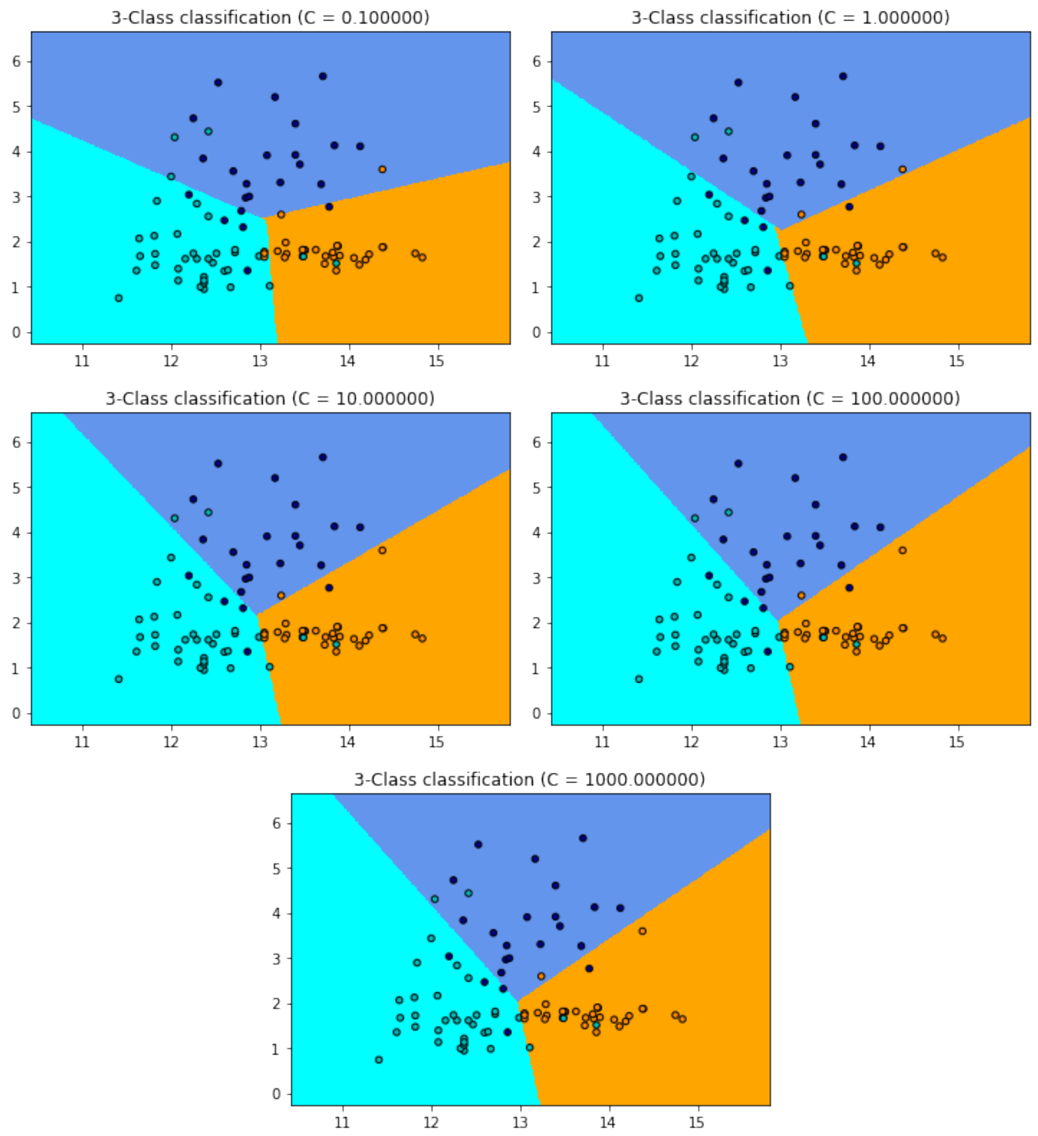
```
The method with k = 3 on the test set
              precision    recall  f1-score   support

           0       0.74      0.78      0.76        18
           1       0.89      0.81      0.85        21
           2       0.69      0.73      0.71        15

    accuracy                           0.78        54
   macro avg       0.77      0.77      0.77        54
weighted avg       0.78      0.78      0.78        54
```

Linear SVM Algorithm

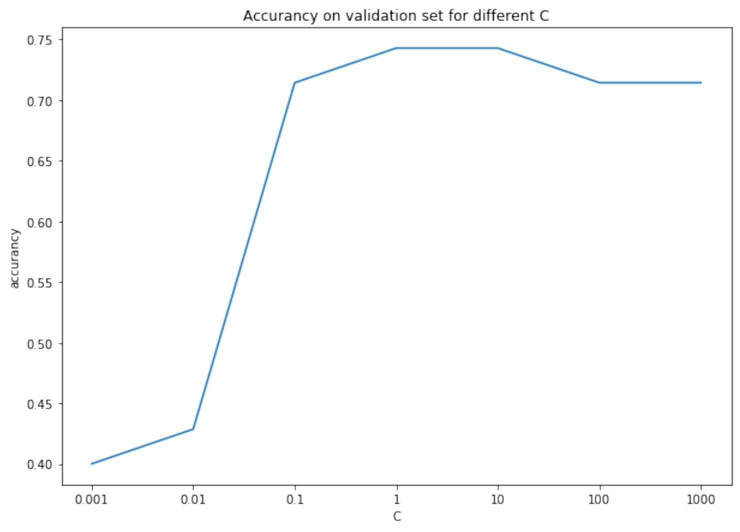
The second supervised ML Alg. is the Linear SVM, a faster implementation of support vector classification for the case of a linear kernel.

We train our model with different C: [0.001, 0.01, 0.1, 1, 10, 100, 1000], a parameter that controls the trade off between smooth decision boundary and classifying training points correctly, finally evaluate the method on the validation set.





For a low C the model is not able to find a good boundary, however for a big value of C the decision boundaries are clearer. In fact, in the graph about the accuracy on validation set shows a result of .74 for value of C = 1, 10.



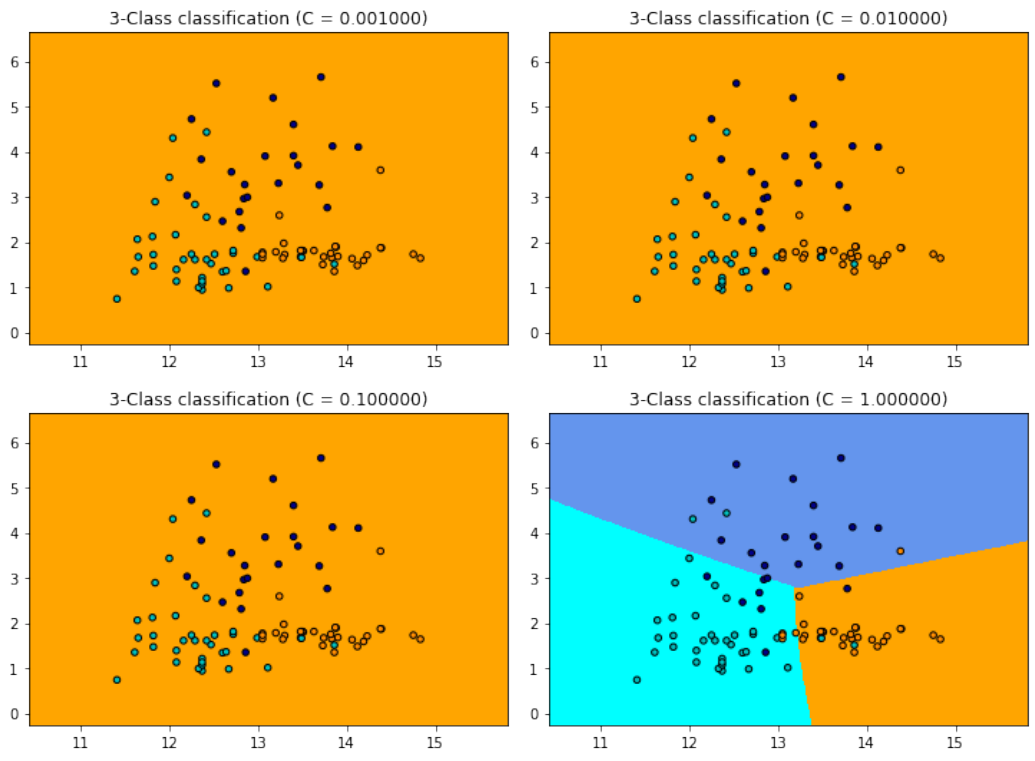
We use the best value of C and evaluate the model on the test set where we obtain an accuracy of 0.76.

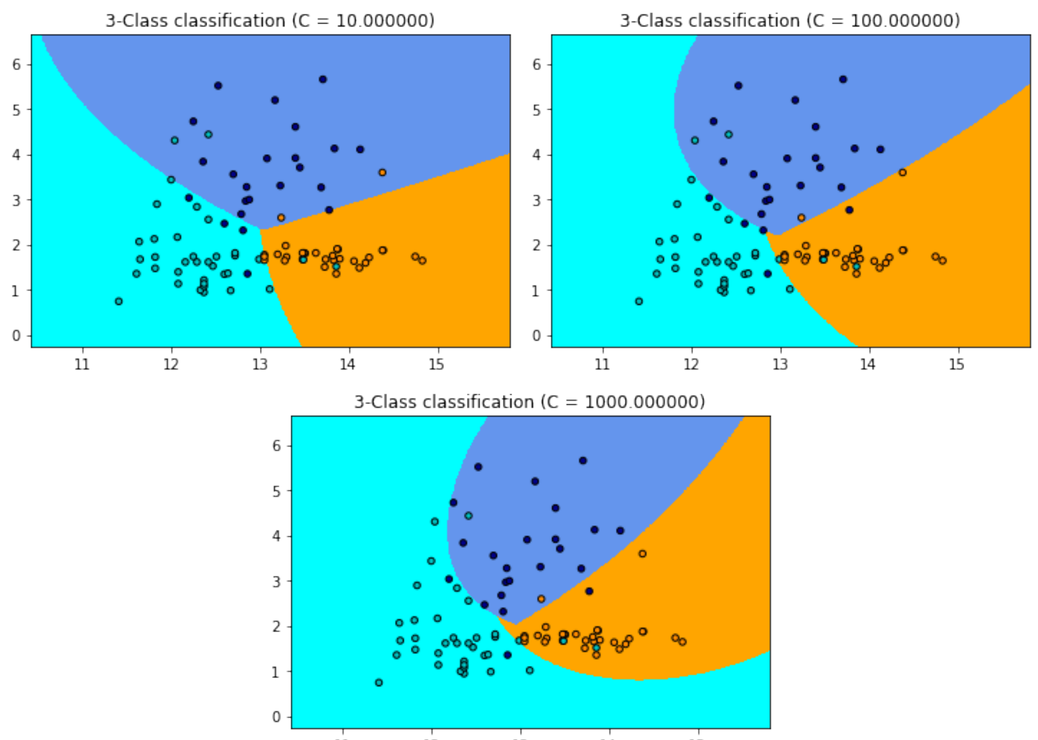
```
The method with C = 1.000000 on the test set
              precision    recall  f1-score   support

           0       0.74      0.78      0.76        18
           1       0.85      0.81      0.83        21
           2       0.67      0.67      0.67        15

    accuracy                           0.76        54
   macro avg       0.75      0.75      0.75        54
weighted avg       0.76      0.76      0.76        54
```

SVM with RBF Kernel

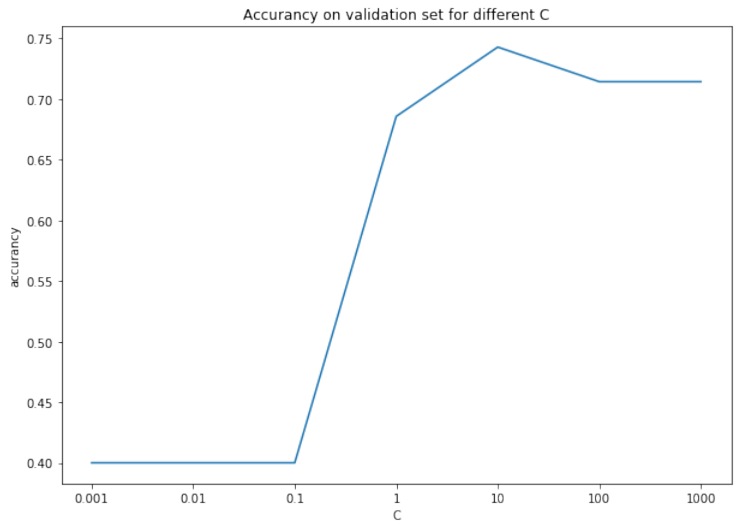
The most popular kernel is the Radial Basis Function (RBF) kernel. It is used when the boundaries are hypothesized to be curve shaped. RBF kernel uses two main parameters: gamma and C, that are related to the decision region and the penalty for misclassifying a data point.

First, we train the model with the same parameters used for the linear SVM **C = [0.001, 0.01, 0.1, 1, 10, 100,1000]** and we leave the parameter gamma by default ('scale').





For C less equal than 0.1 the algorithm is not able to find a boundary, it considers all data point in only one class. For C grater than 1 the graphs are different. We can see, respect to the linear svm, the boundaries are smooth/curved. So, we evaluate every model with different C on the validation set to find the best C.



With C = 10 we obtain an accuracy of 0.74 on validation set, like in linear svm. Now we evaluate the best C on the test set.

```
The method with C = 10.000000 on the test set
              precision    recall  f1-score   support

           0       0.72      0.72      0.72        18
           1       0.81      0.81      0.81        21
           2       0.67      0.67      0.67        15

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

In this case the accuracy obtained is slightly lower than the linear svm. In all of the algorithms tested, we have some problem with the prediction of class 2. Perhaps one of reasons is the number of samples, only 48 on 178 samples.

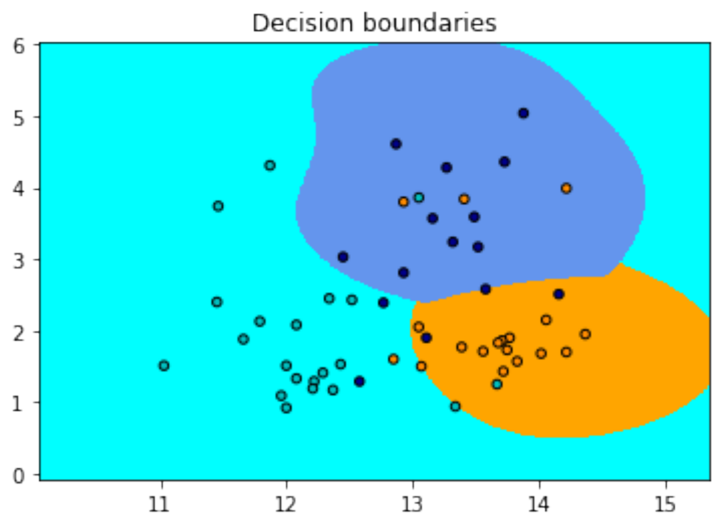
To improve the result, we perform a grid search of the best parameters for an RBF kernel: we will now tune both gamma and C at the same time, then we train the model and score it on the validation set.

```
parameters = {'C':[0.001, 0.01, 0.1, 1, 10, 100,1000],

            'gamma':['scale','auto', 1.e-02, 1.e-01,
            1.e+00, 1.e+01, 1.e+02]}
```

```
The method with {'C': 0.1, 'gamma': 1.0} on the val set
              precision    recall  f1-score   support

           0       0.90      0.75      0.82        12
           1       0.91      0.71      0.80        14
           2       0.50      0.78      0.61         9

    accuracy                           0.74        35
   macro avg       0.77      0.75      0.74        35
weighted avg       0.80      0.74      0.76        35
```

Then we evaluate the best parameters on the test set and plot the decision boundaries. We obtain a better result on the test set with {'C': 0.1, 'gamma': 1.0} and accuracy of 0.78.

```
The method with {'C': 0.1, 'gamma': 1.0} on the test set
              precision    recall  f1-score   support

           0       0.76      0.72      0.74        18
           1       0.83      0.90      0.86        21
           2       0.71      0.67      0.69        15

    accuracy                           0.78        54
   macro avg       0.77      0.76      0.77        54
weighted avg       0.77      0.78      0.78        54
```



K-Fold cross validation

To avoid overfitting, we use the K-fold cross validation. The idea is to use the initial training data to generate multiple mini train-test splits. The procedure has a single parameter called K that refers to the number of groups that a given data sample is to be split into. This is one among the best approach if we have a limited input data.

First, we merge the training and validation split and now, we have 70% training and 30% test data. After that, we repeat the previous grid search for gamma and C with 5-fold validation to find the best configuration for the model.

```
The method with {'C': 0.001, 'gamma': 'scale'} on the test
set
              precision    recall  f1-score   support

           0       0.81      0.72      0.76        18
           1       0.83      0.90      0.86        21
           2       0.73      0.73      0.73        15

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

Finally, we evaluate the best parameter on the test set and we obtain an accuracy of 0.80, the best result so far, especially for class 2. One of the reasons is given by the union of the two sets which allowed the model to train on more samples and another reason is the no randomness of using some observations for training and validation set like in validation-set method as each observation is considered for both training and validation.

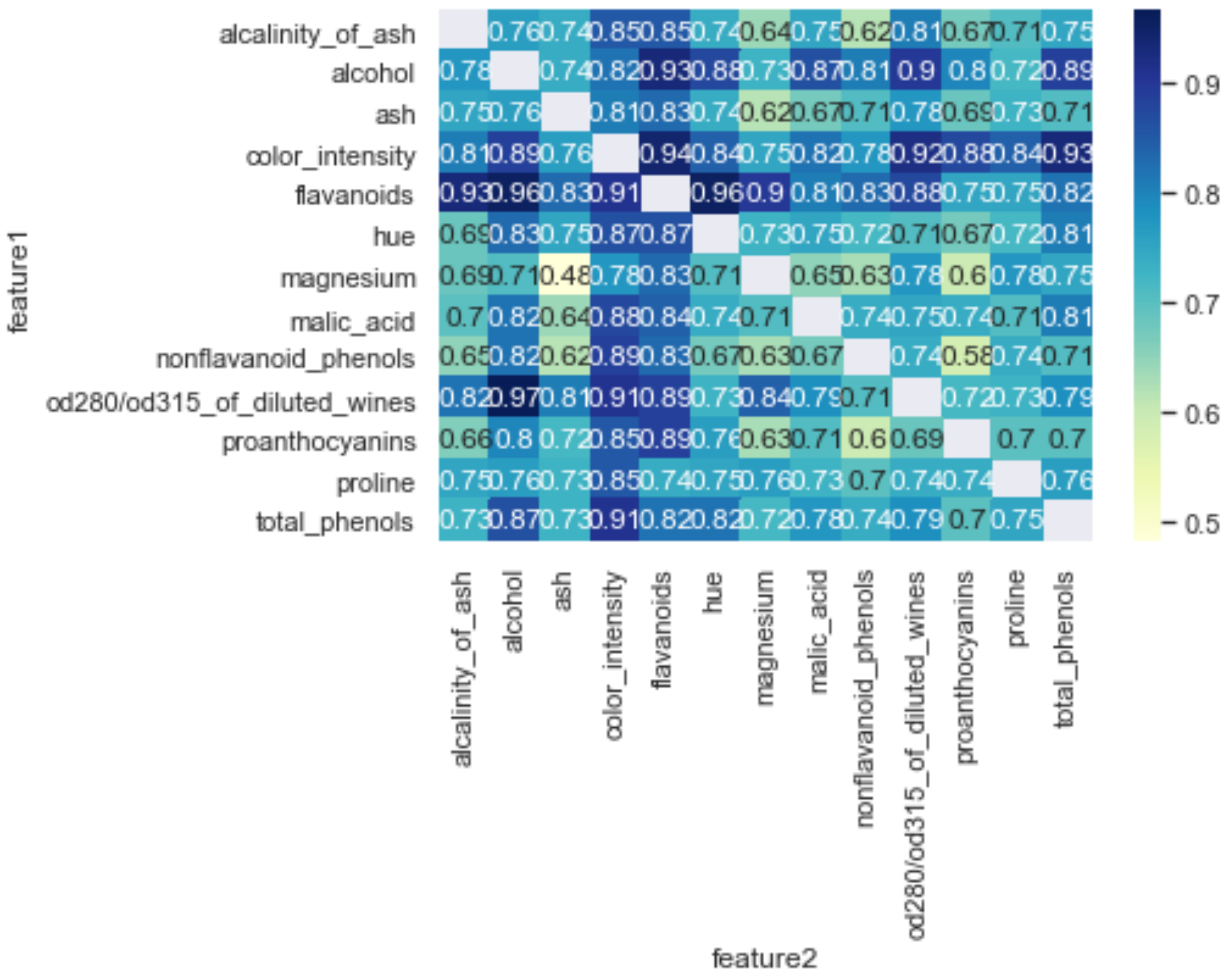
Difference between KNN and SVM

In this dataset with only two features, we have seen that KNN and SVM have obtained similar results, a slightly better result with SVM but after some adjustments. So, SVM take cares of outliers better than KNN, but if training data is much lager than number of features, KNN is better than SVM. SVM outperforms KNN when there are large features and lesser training data.

Different pairs of attributes

To test different pairs of attributes we use the SVM with RBF Kernel and a grid search to find best parameters. The dataset is randomly split into train and test sets in proportion 5:3.

The two attributes that could build a good model for this dataset are od280/od315_of_diluted_wines and alcohol with accuracy on test set of 0.97.



	alcalinity_of_ash	alcohol	ash	color_intensity	flavanoids	hue	magnesium	malic_acid	nonflavanoid_phenols	od280/od315_of_diluted_wines	proanthocyanins	proline	total_phenols
alcalinity_of_ash		0.76	0.74	0.85	0.85	0.74	0.64	0.75	0.62	0.81	0.67	0.71	0.75
alcohol	0.78		0.74	0.82	0.93	0.88	0.73	0.87	0.81	0.9	0.8	0.72	0.89
ash	0.75	0.76		0.81	0.83	0.74	0.62	0.67	0.71	0.78	0.69	0.73	0.71
color_intensity	0.81	0.89	0.76		0.94	0.84	0.75	0.82	0.78	0.92	0.88	0.84	0.93
flavanoids	0.93	0.96	0.83	0.91		0.96	0.9	0.81	0.83	0.88	0.75	0.75	0.82
hue	0.69	0.83	0.75	0.87	0.87		0.73	0.75	0.72	0.71	0.67	0.72	0.81
magnesium	0.69	0.71	0.48	0.78	0.83	0.71		0.65	0.63	0.78	0.6	0.78	0.75
malic_acid	0.7	0.82	0.64	0.88	0.84	0.74	0.71		0.74	0.75	0.74	0.71	0.81
nonflavanoid_phenols	0.65	0.82	0.62	0.89	0.83	0.67	0.63	0.67		0.74	0.58	0.74	0.71
od280/od315_of_diluted_wines	0.82	0.97	0.81	0.91	0.89	0.73	0.84	0.79	0.71		0.72	0.73	0.79
proanthocyanins	0.66	0.8	0.72	0.85	0.89	0.76	0.63	0.71	0.6	0.69		0.7	0.7
proline	0.75	0.76	0.73	0.85	0.74	0.75	0.76	0.73	0.7	0.74	0.74		0.76
total_phenols	0.73	0.87	0.73	0.91	0.82	0.82	0.72	0.78	0.74	0.79	0.7	0.75	