Pattern Recognition Methods
and
Introduction to Machine Learning

Homework 7 - Report
Margins and Kernels

Prepared by: Mehmet Kapson

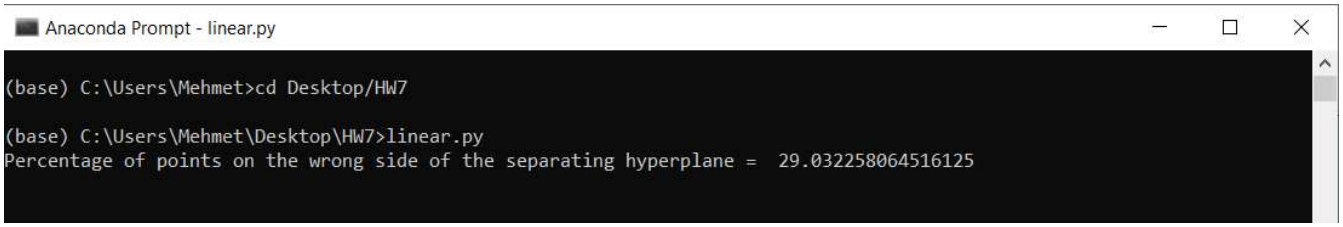
22.05.2019

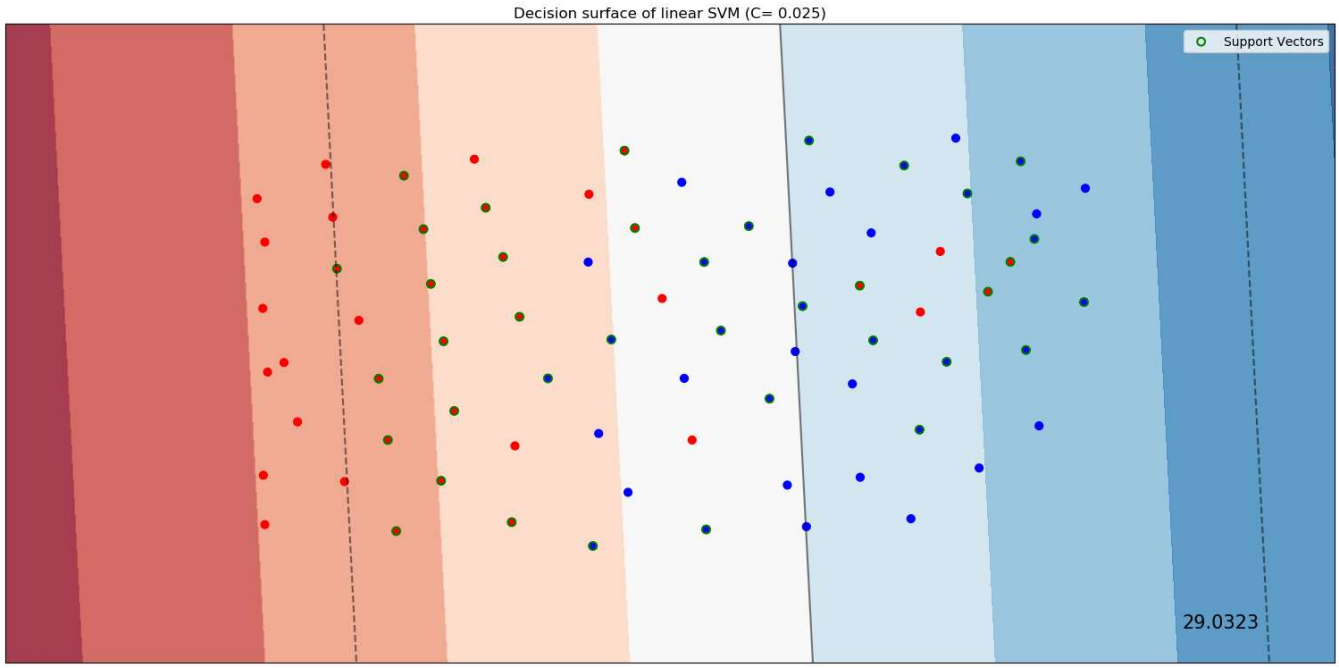
MATLAB R2018b is used to find the given points' coordinates. Besides, all data points of both classes are manually written to two separate text files, points_class_0 and points_class_1, respectively. Data is standardized and it is splitted to train and test on 60-40 ratio, respectively. Linear SVM, 3rd degree polynomial SVM and RBF SVM are repeated several times by using different C and gamma values to see the effects of these coefficients. Some functions are defined in a different python file named as utils.py for easier understanding of the code, also less complexity. For all graphs, 'percentage of points on the wrong side of the separating hyperplane' is shown on the right bottom side of the graph. Furthermore, regions which are created by class points are colored brighter red-blue or darker red-blue, depending on their classes and their distances to the separating hyperplane. Moreover, support vectors are marked by green circles.

Linear SVM

linear.py file is run several times by manually changing C value after every run.

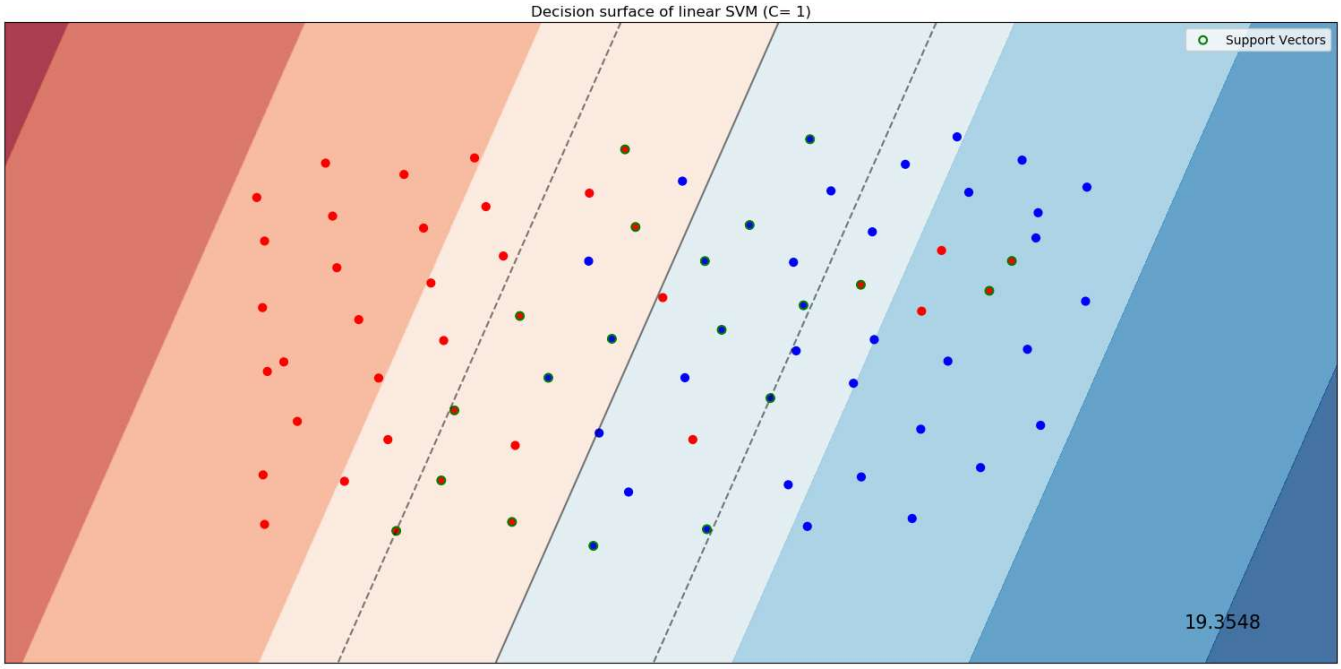
The results of using linear SVM with C = 0.025 are shown as below:





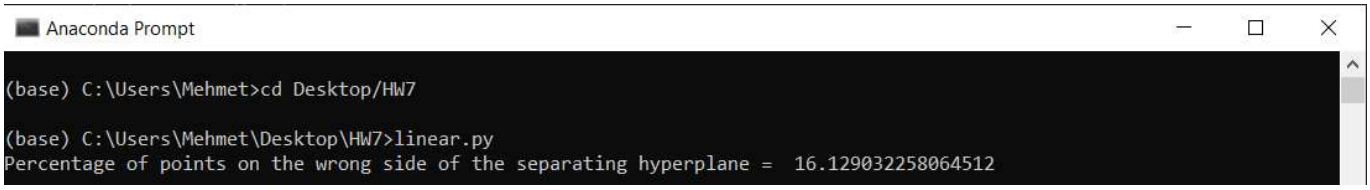
As you can see, margin is large; however, there are lots of misclassified points. Percentage of misclassified points is 29.0323%.

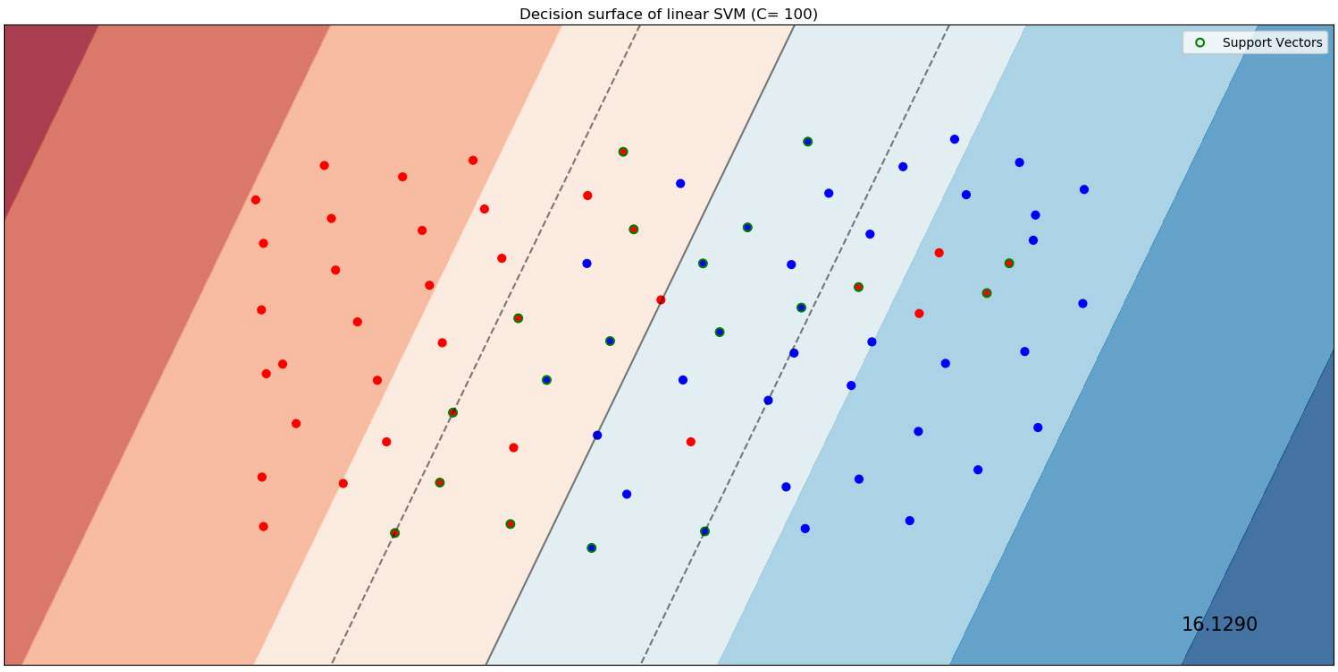
The results of using linear SVM with C = 1 are shown as below:



This time C is increased from 0.025 to 1. Margin is decreased but now, there are less misclassified points than previous run. Percentage of misclassified points is 19.3548%.

The results of using linear SVM with C = 100 are shown as below:





C is increased again. Now, it's value is 100. Margin is slightly decreased; besides, percentage of misclassified points becomes 16.1290%.

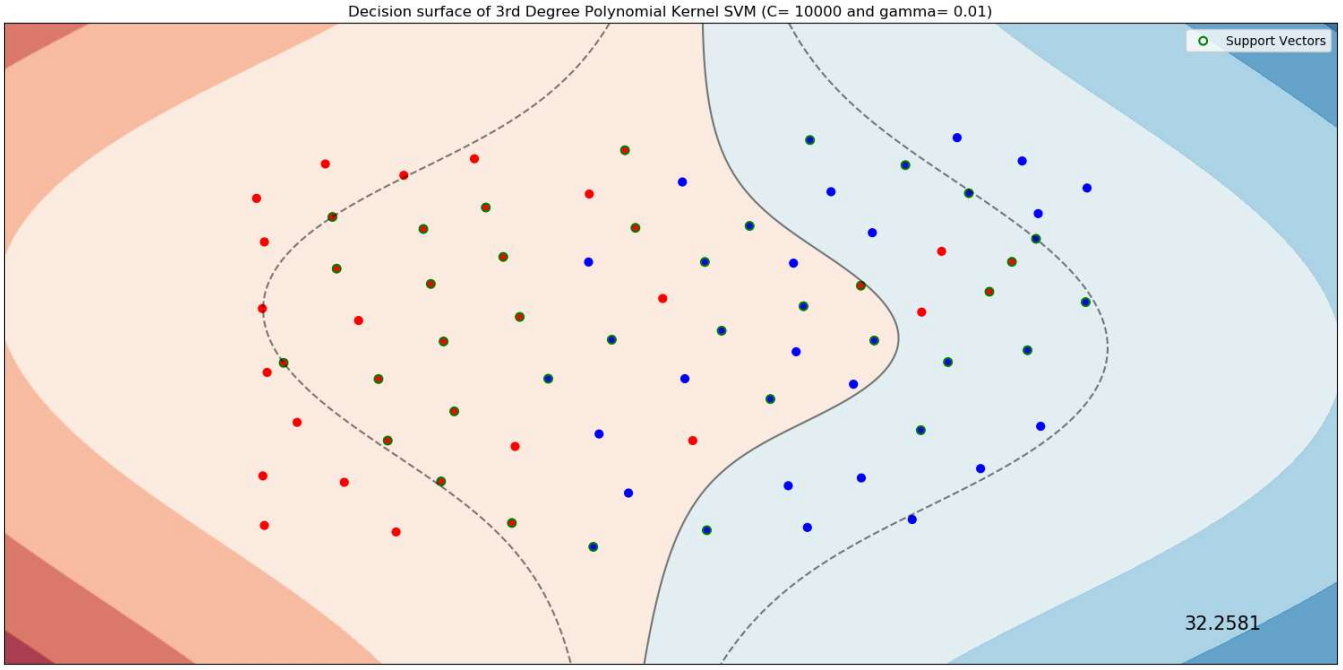
As a result of these runs for linear SVM, one can easily say that when C increases, margin and misclassification error decrease.

3rd Degree Polynomial Kernel SVM

poly3.py file is run several times by manually changing C value and gamma value after every run.

The results of using 3rd Degree Polynomial Kernel SVM with C = 10000 and gamma = 0.01 are shown as below:

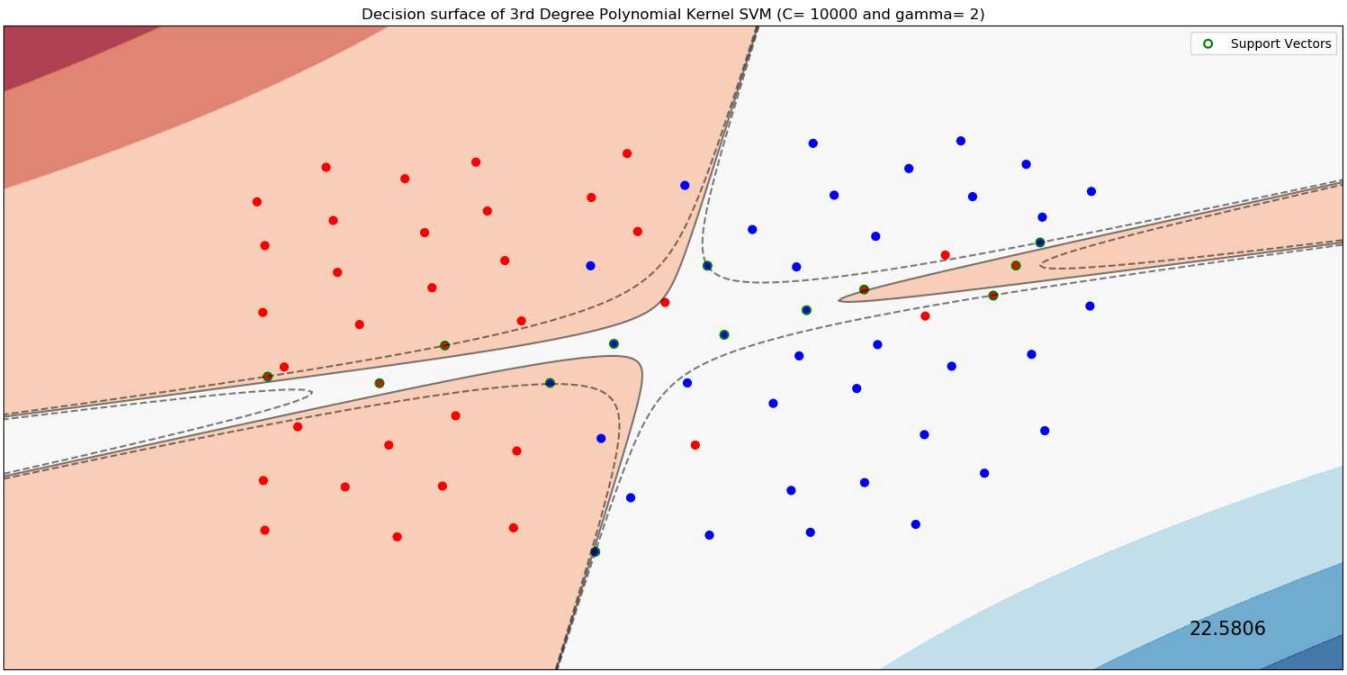
```
(base) C:\Users\Mehmet\Desktop\HW7>poly3.py
Percentage of points on the wrong side of the separating hyperplane =  32.25806451612904
```



When gamma is too small, SVM cannot understand the dataset's shape well. Thus, it causes lot of misclassified points. Percentage of misclassified points is 32.2581%.

The results of using 3rd Degree Polynomial Kernel SVM with C = 10000 and gamma = 2 are shown as below:

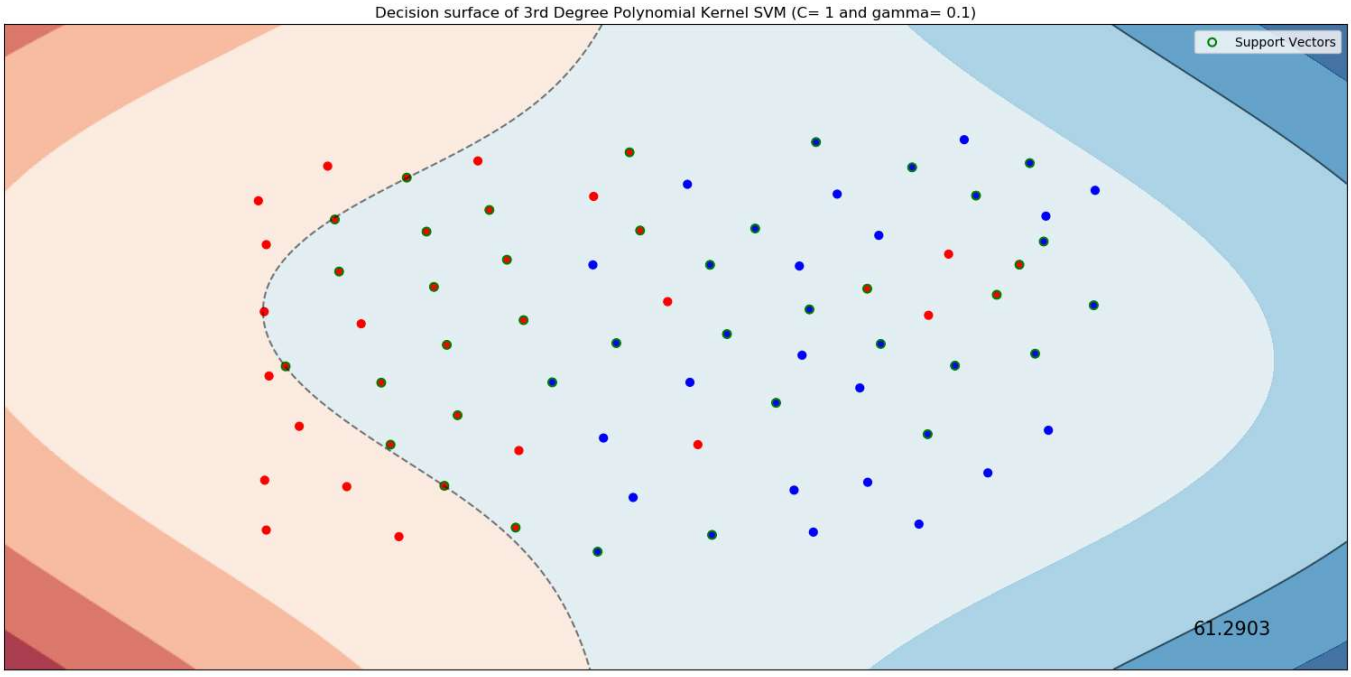
```
(base) C:\Users\Mehmet\Desktop\HW7>poly3.py
Percentage of points on the wrong side of the separating hyperplane =  22.580645161290324
```



C is still 10000. However, gamma is equal to 2 now. As you can see, when gamma is very big, points which are closer to hyperplane get more weight than the distant points, thus the hyperplane becomes curled. Percentage of misclassified points is 22.5806%.

The results of using 3rd Degree Polynomial Kernel SVM with C = 1 and gamma = 0.1 are shown as below:

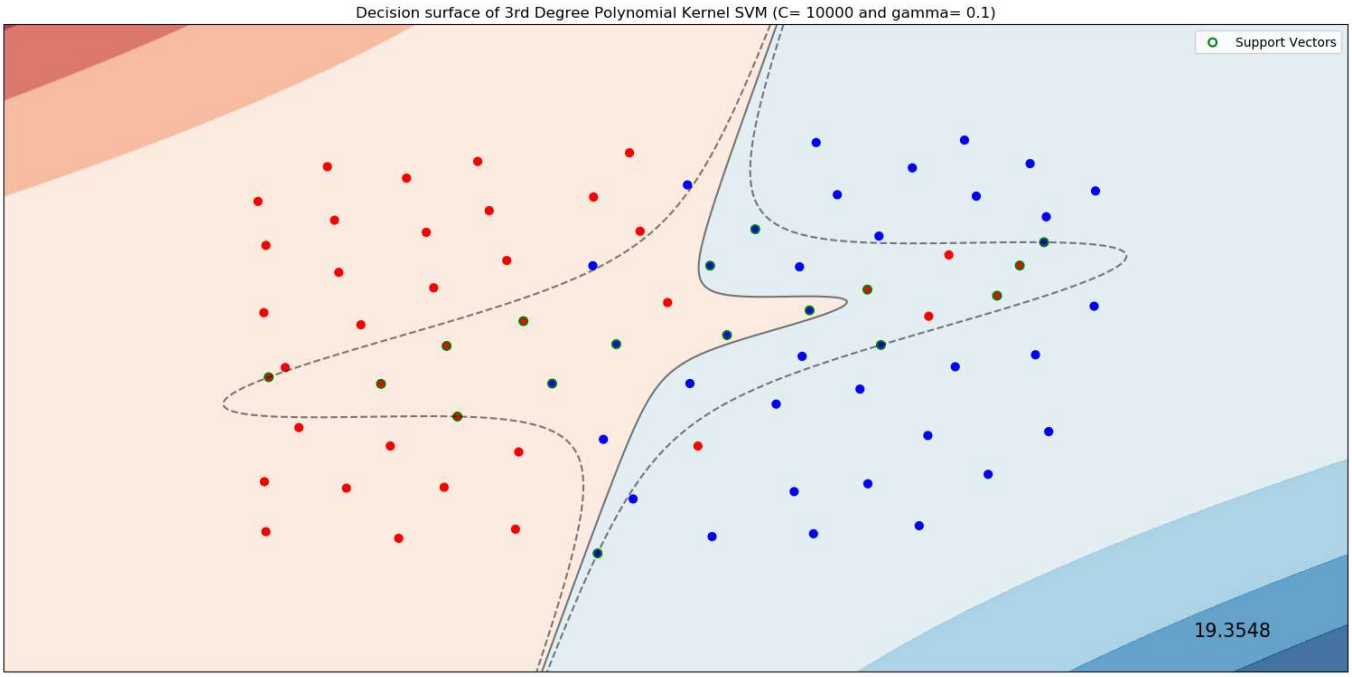
```
(base) C:\Users\Mehmet\Desktop\HW7>poly3.py
Percentage of points on the wrong side of the separating hyperplane =  61.29032258064516
```



As you can see, C is too small that SVM cannot work properly and many points are classified wrong. Percentage of misclassified points is 61.2903%.

The results of using 3rd Degree Polynomial Kernel SVM with C = 10000 and gamma = 0.1 are shown as below:

```
(base) C:\Users\Mehmet\Desktop\HW7>poly3.py
Percentage of points on the wrong side of the separating hyperplane =  19.354838709677423
```



C is increased from 1 to 1000, while gamma is the same as previous one. Hyperplane got pretty good curves and there are less misclassified points. Percentage of misclassified points is 19.3548%.

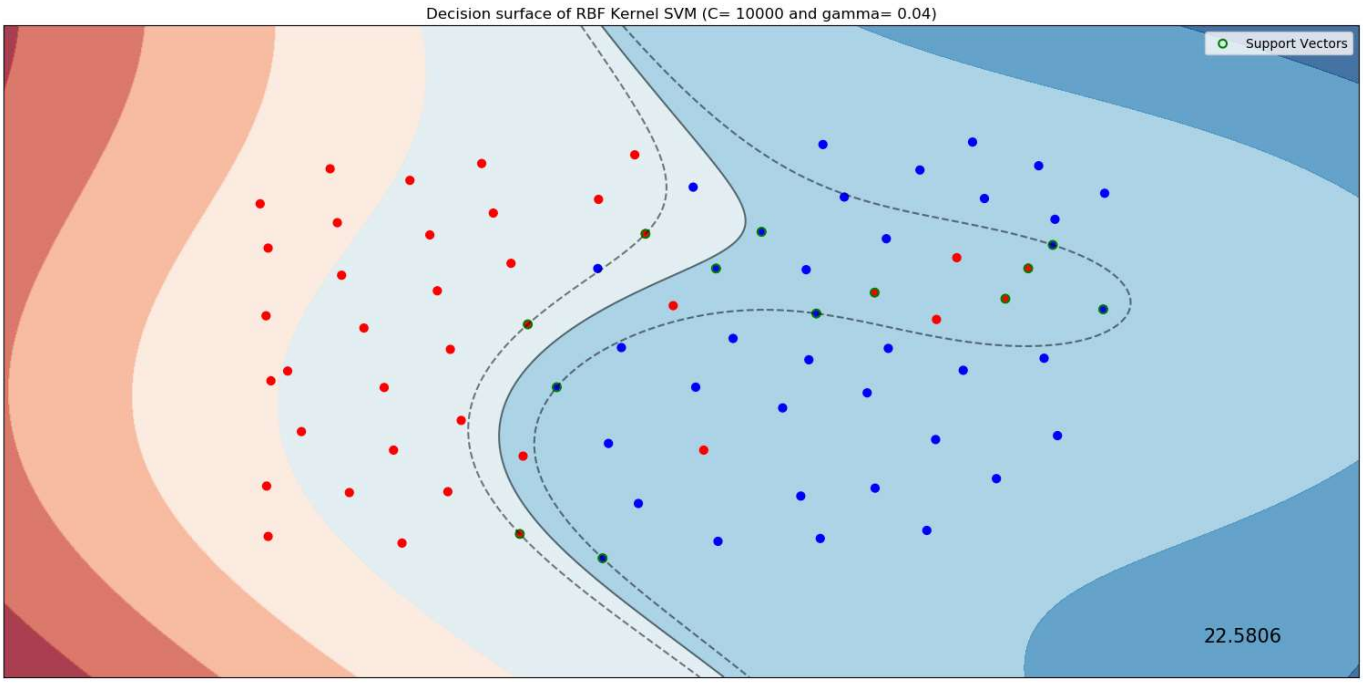
To conclusion, to be able to run 3rd Degree Polynomial Kernel SVM properly, C and gamma values must be chosen correctly. Otherwise, it gives wrong results.

RBF Kernel SVM

rbf.py file is run several times by manually changing C value and gamma value after every run.

The results of using RBF Kernel SVM with C = 10000 and gamma = 0.04 are shown as below:

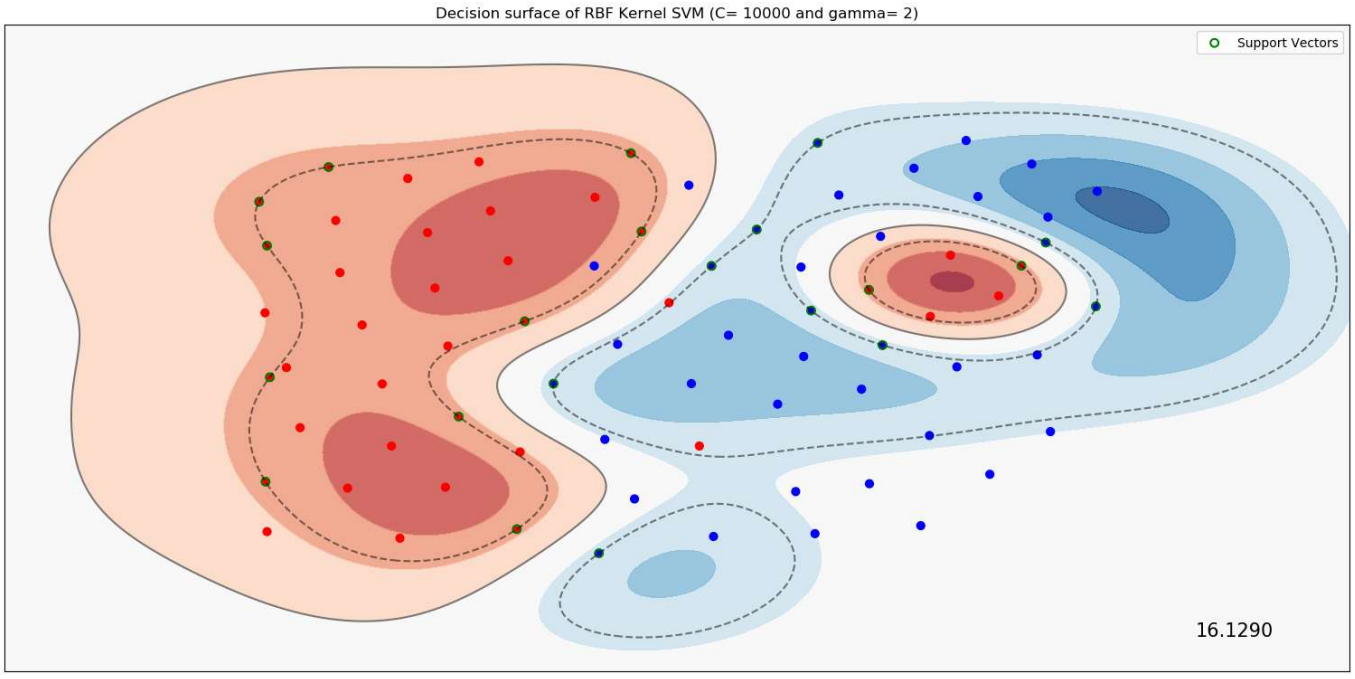
```
(base) C:\Users\Mehmet\Desktop\HW7>rbf.py
Percentage of points on the wrong side of the separating hyperplane =  22.580645161290324
```



Results are better than not all of them but most of the previous attempts with 3rd degree polynomial kernel SVM. Percentage of misclassified points is 22.5806%.

The results of using RBF Kernel SVM with C = 10000 and gamma = 2 are shown as below:

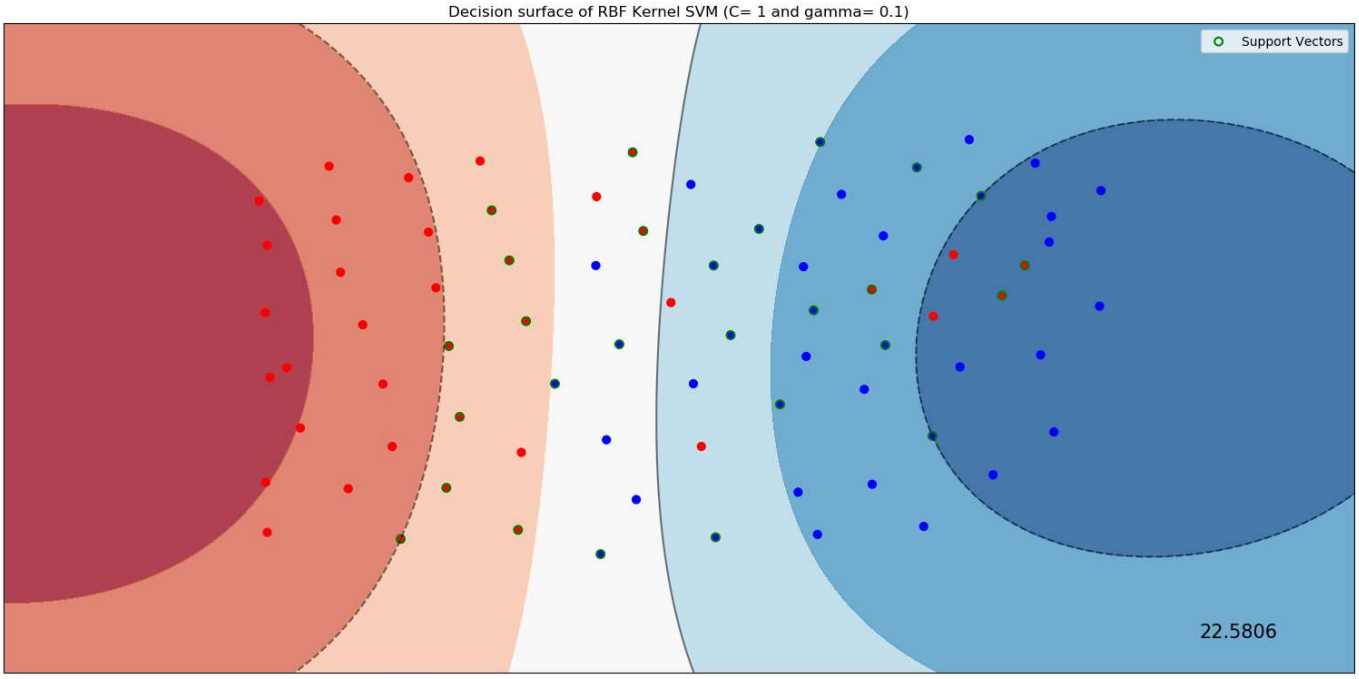
```
(base) C:\Users\Mehmet\Desktop\HW7>rbf.py
Percentage of points on the wrong side of the separating hyperplane =  16.129032258064512
```



When C = 10000 and gamma = 2, result looks like mountains with different heights. Classification error is lower than the previous one. It is 16.1290%.

The results of using RBF Kernel SVM with C = 1 and gamma = 0.1 are shown as below:

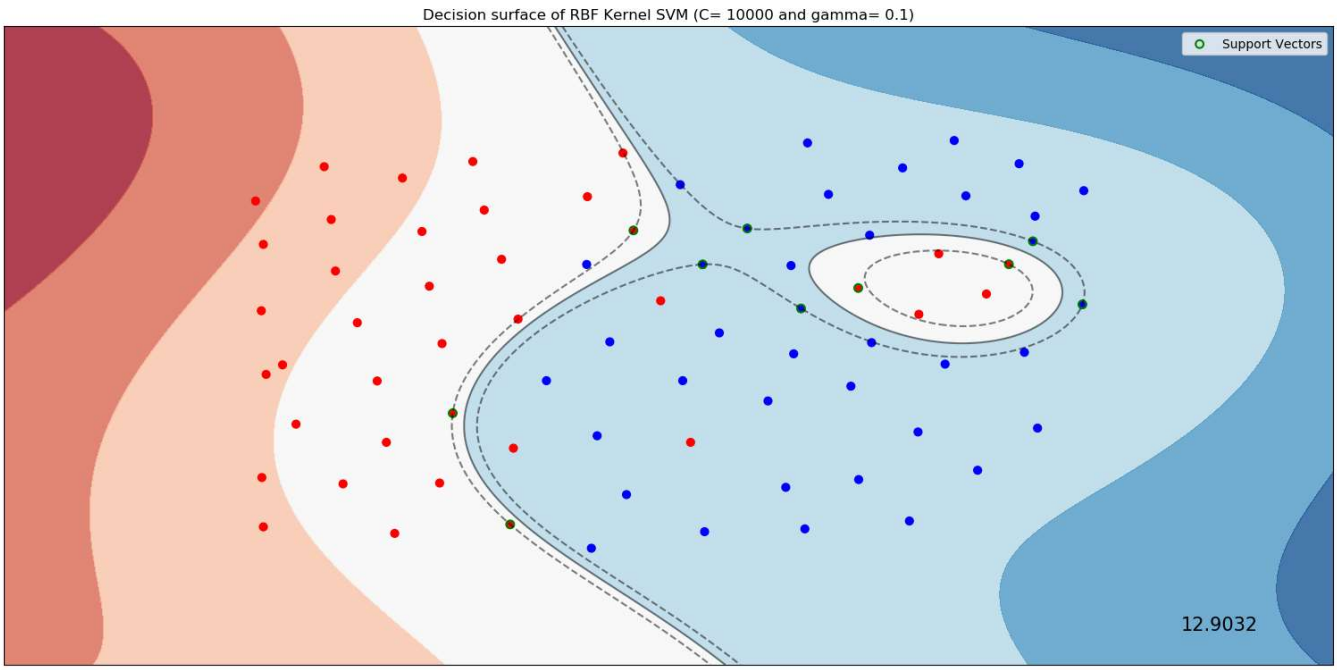
```
(base) C:\Users\Mehmet\Desktop\HW7>rbf.py
Percentage of points on the wrong side of the separating hyperplane =  22.580645161290324
```



Again, percentage of misclassified points is 22.5806%. It is same as first RBF run.

The results of using RBF Kernel SVM with C = 10000 and gamma = 0.1 are shown as below:

```
(base) C:\Users\Mehmet\Desktop\HW7>rbf.py
Percentage of points on the wrong side of the separating hyperplane =  12.903225806451612
```



These results are the best between all of runs which are reported in this document. The misclassification ratio is only 12.9032%. As one can see, most of the points are classified well. There are 2 hyperplanes which separates most of two classes' points.

Conclusion

To sum up, 3 different methods of SVM is run in this report. C values and gamma values affect the SVM's performance and accuracy. It is really sensitive to C values and gamma values. Thus, these parameters must be chosen correctly by considering the given dataset.