SVM

Support Vectors are simply the coordinates of individual observation. Support vector machine is a supervised machine learning that can be used both SVms are based on idea of finding a hyperplane that best divides a dataset into classes.

Pros

It works really well with a clear margin of separation

It is effective in high dimensional spaces

It is effective in cases where the number of dimensions is greater than the number of samples

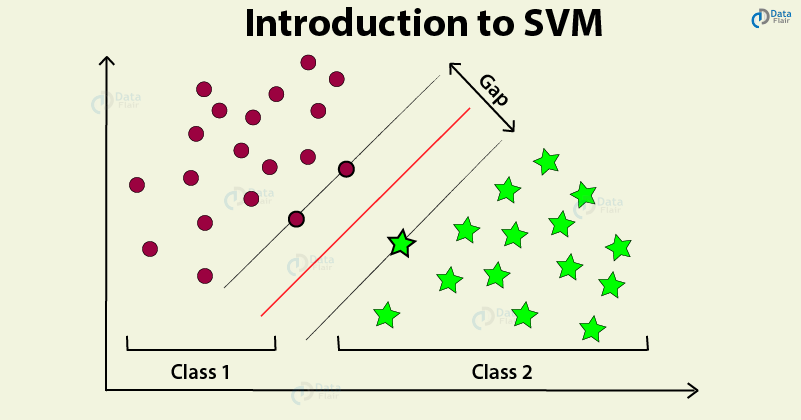
It is also memory efficient

Cons

It doesn’t perform well when we have large data set because the required time is higher

It also doesn’t perform very well, when the data set has more noise i.e target classes are overlapping.

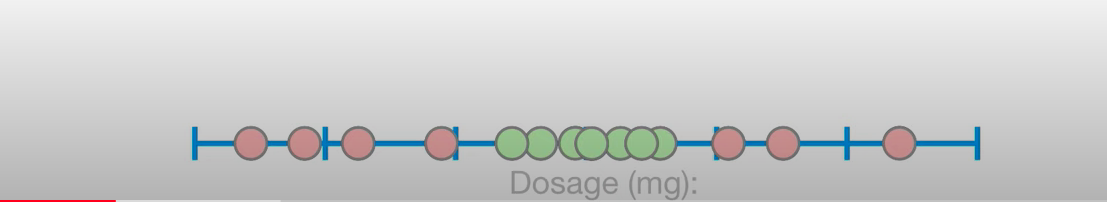
How does it work?



SVM uses either polynomial kernel or radial kernel to classify by moving into higher dimension.

SVM uses soft margin and allows classification to prevent high variance. Hard margin is a boundary that is determined based on a single point which makes it too sensitive to noisy data.

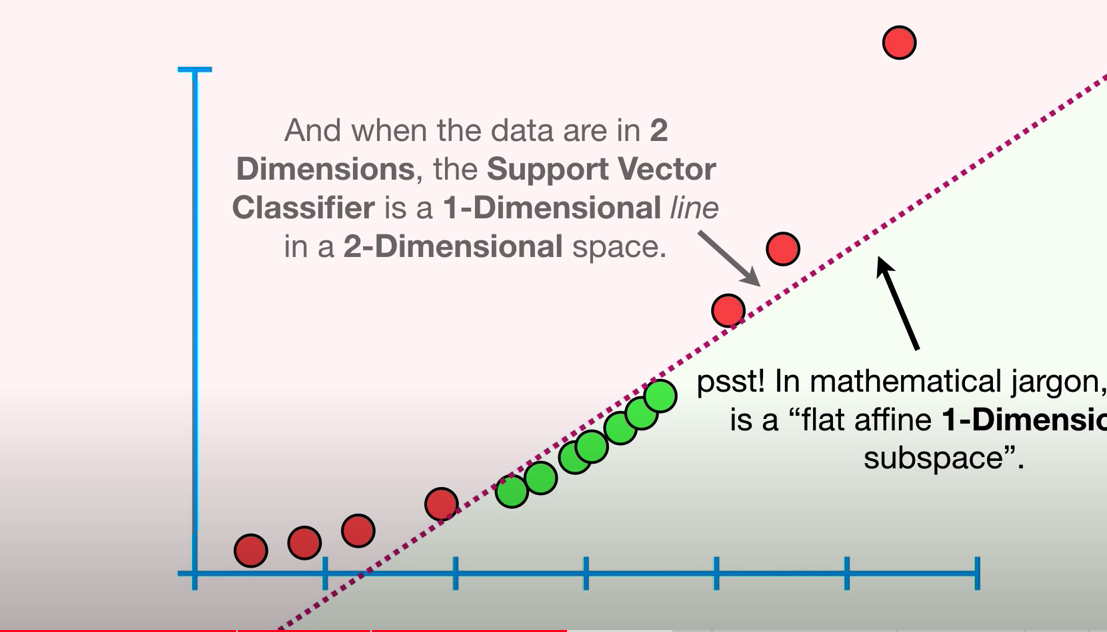
We can classify the above dataset well enough by using SVM and allowing some misclassifications. However there is some cases no matter what we won’t be able to have a good soft margin.



Let ‘s above dataset represent dosage and treatment result. Green dots are treated red and are not treated by a given dose. If dosage is too low or high it doesn't treat the patients. And there is no way to have a proper support vector classifier to separate these points.

This is where the support vector machine comes into play.

If we use polynomial kernel . We can simply use y=x^2 (where dosage input) and take square of each input, then we will have following graph.



Now we can use line and separate classes. If there is 3 attributes we would be in tree dimensions and use planes to separate classes. Ultimately, n attributes will require n dimensions and n-1 figures to classify.

Also we can use a radial kernel that finds weighted distance in infinite dimensions and assign value based on nearest neighbor.

Python Sklearn Implementation

Import libraries

Load data

Separate futures and target

(Split data into train and test for some cases to find accuracy, Also we may need to normalize input for some ML algorithm)

Define the model

Fit the Model

Predict