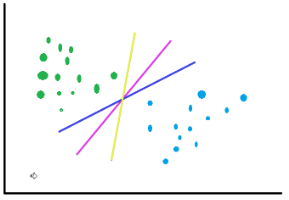
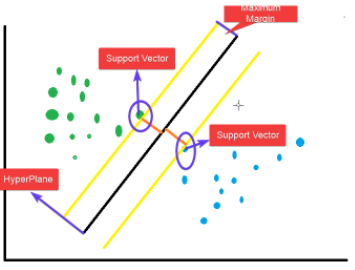
**Support Vector Machines**

**Introduction**

Support Vector Machine is a supervised Machine Learning algorithm widely used for solving different machine learning problems. Given a dataset, the algorithm tries to divide the data using hyperplanes and then makes the predictions. SVM is a non-probabilistic linear classifier. While other classifiers, when classifying, predict the probability of a data point to belong to one group or the another, SVM directly says to which group the datapoint belongs to without using any probability calculation.

**Understanding the Mathematics involved**

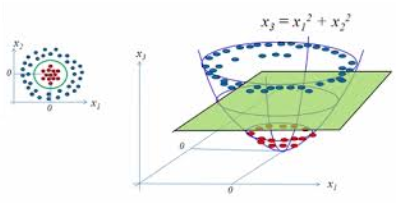
Let’s take the example of the following dataset and see how can we divide the data into appropriate groups.



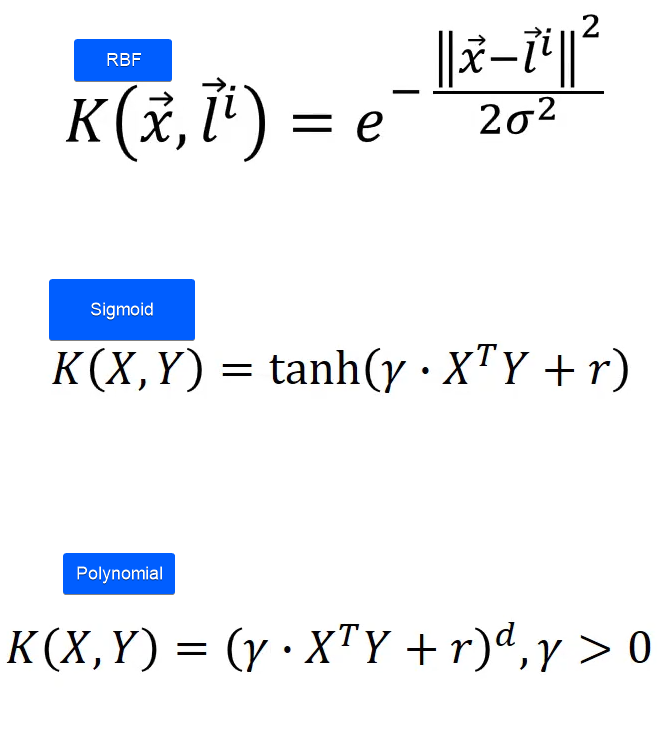
We can see that there are two groups of data. The question is how to divide these points into two groups. It can be done using any of the three lines. Or, for that purpose, there can be an infinite number of straight lines that can divide these points into two classes. Now, which line to choose? SVM solves this problem using the maximum margin as shown

The black line in the middle is the optimum classifier. This line is drawn to maximise the distance of the classifier line from the nearest points in the two classes. It is also called a **hyperplane** in terms of SVM. A Hyperplane is an n-1 dimensional plane which optimally divides the data of n dimensions. Here, as we have only a 2-D data, so the hyperplane can be represented using one dimension only. Hence, the hyperplane is a line here. The two points (highlighted with circles) which are on the yellow lines, they are called the **support vectors**. As it is a 2-D figure, they are points. In a multi-dimensional space, they will be vectors, and hence, the name- support vector machine as the algorithm creates the optimum classification line by maximising its distance from the two support vectors.

When the data is not linearly separable, then to create a hyperplane to separate data into different groups, the SVM algorithm needs to perform computations in a higher-dimensional space. But the introduction of new dimensions makes the computations for the SVMs more intensive, which impacts the algorithm performance. To rectify this, mathematicians came up with the approach of Kernel methods. Kernel methods use kernel functions available in mathematics. The unique feature of a kernel function is to compute in a higher-dimensional space without calculating the new coordinates in that higher dimension. It implicitly uses predefined mathematical functions to do operations on the existing points which mimic the computation in a higher-dimensional space without adding to the computation cost as they are not actually calculating the coordinates in the higher dimension thereby avoiding the computation of calculating distances from the newly computed points. This is called the kernel trick.

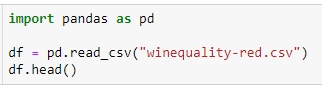
In the left diagram above, we have a non-linear distribution of data as we can not classify a data using a linear equation. To solve this problem, we can project the points in a 3-dimensional space and then derive a plane which divides the data into two parts. In theory, that’s what a kernel function does without computing the additional coordinates for the higher dimension.

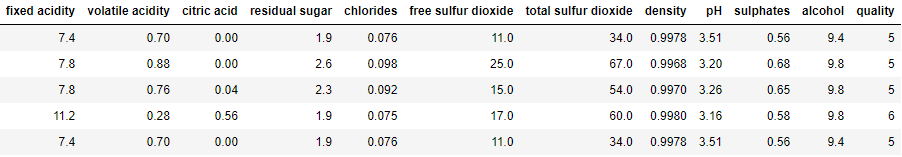
**The equation for some of the kernels are:**



**for more:**[**https://en.wikipedia.org/wiki/Support-vector\_machine**](https://en.wikipedia.org/wiki/Support-vector_machine)

**1.import the data**



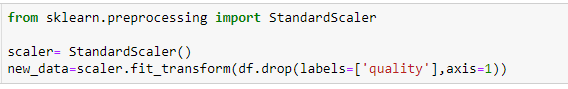


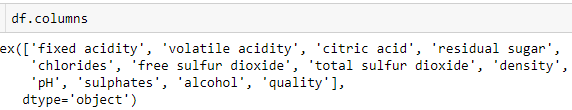
df.info()

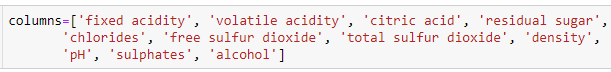
df.isnull().sum()

**2.Preprocessing of data**

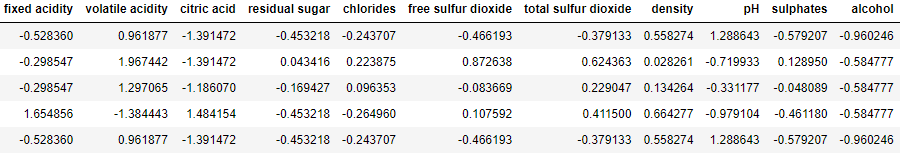
As the data consists of values which differ a lot in magnitude, they need to be brought to the same scale. It's done using the standard scalar.





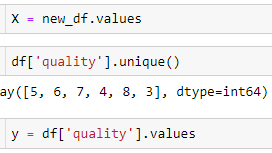




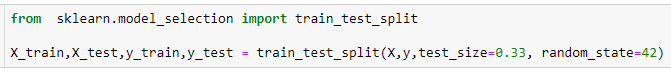


As there are no missing values, we don't need to do data imputation.

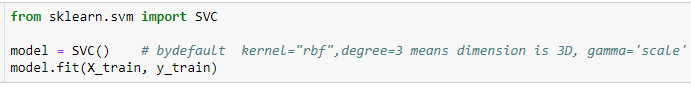
**3.Identify X(independent variables), y(dependent variables)**



**4.Split the data into Train and Test**



**5.Fit / Train the model using train data**

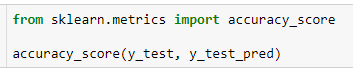


SVC()

**6.predict on test data**



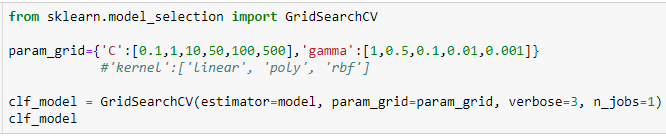
**7.Metrics(Accuracy)**



0.5984848484848485

* As observed, the accuracy of the model is quite low.
* it means 59% of data can be classified and remaining 41% of data cannot be classified.so, the hyperplane which we drawn is wrong.
* We need to implement the grid search approach to optimize the parameters to give the best accuracy.

**8. if we are not happy with accuracy then do hyper parameter tunning and rebuild the model Implementing Grid Search**



GridSearchCV(estimator=SVC(), n\_jobs=1,

param\_grid={'C': [0.1, 1, 10, 50, 100, 500],

'gamma': [1, 0.5, 0.1, 0.01, 0.001]},

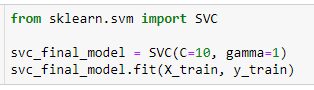
verbose=3)



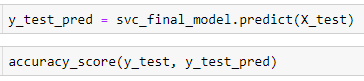


{'C': 10, 'gamma': 1}

**Final accuracy**

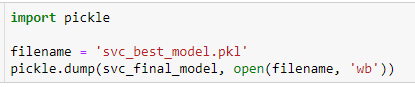


SVC(C=10, gamma=1)



0.6268939393939394

**9. Save the model**



**10. load the model**



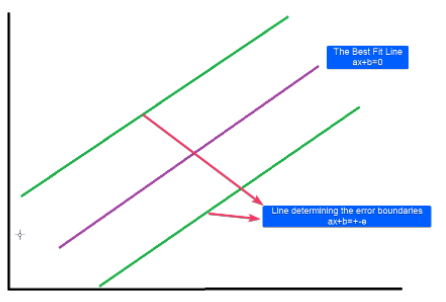


array([5], dtype=int64)

**for More:**[**https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote09.html**](https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote09.html)

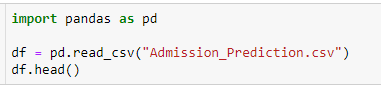
**Support Vector Regression**

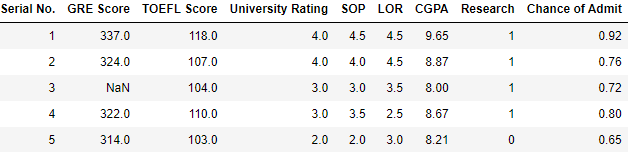
Let’s talk about Linear Regression first. How to determine the best fit line? The idea is to create a line which minimises the total residual error. The SVR approach is a bit different. Instead of trying to minimise the error, SVR focuses on keeping the error in a fixed range. This approach can be explained using three lines. The first line is the best fit regressor line, and the other two lines are the bordering ones which denote the range of error.



What does this mean? It means that we are going to consider the points inside this ± error boundary only for preparing our model. In other words, the best fit line(or the hyperplane) will be the line which goes through the maximum number of data points and the error boundaries are chosen to ensure maximum inclusion. This error term can be customized using the 'epsilon' parameter defined for the scikit-learn SVR implementation.

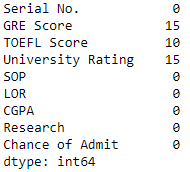
**1.import the data**



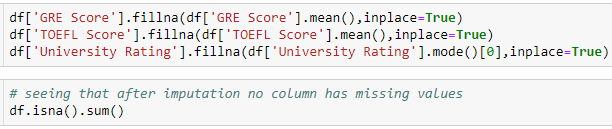


df.info()

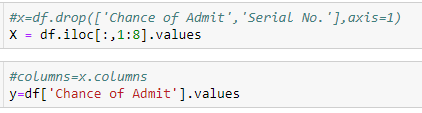
df.isna().sum()



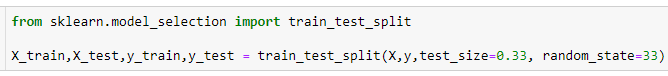
**2.Preprocessing of data**



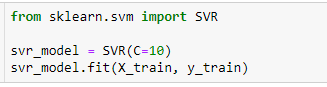
**3.Identify X(independent variables), y(dependent variables)**



**4.Split the data into Train and Test**



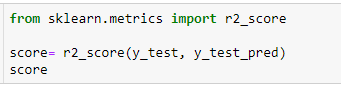
**5.Fit / Train the model using train data**



**6.predict on test data**

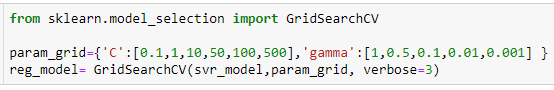


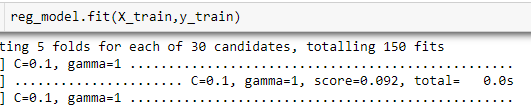
**7.Metrics(Accuracy)**



0.7340987165302327

**8. if we are not happy with accuracy then do hyper parameter tunning and rebuild the model Implementing Grid Search**

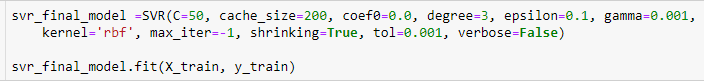




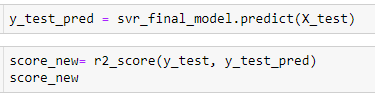


SVR(C=50, gamma=0.001)

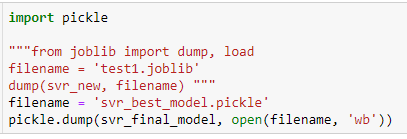
**Final Accuracy**



SVR(C=50, gamma=0.001)

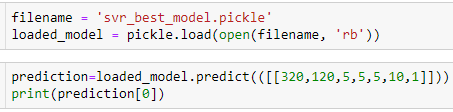


**9. Save the model**



Now this saved model file will be used for prediction. We'll create a Flask app for the same and deploy it to AWS.

**10. load the model**



0.9164032690651018