Support Vector Machines (SVM)

- Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. It is mostly used in classification problems.
- The main objective of this algorithm is to find a hyperplane which separates the classes well.
- There are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.
- The margin is calculated as the perpendicular distance from the line to only the closest points. Only these points are relevant in defining the line and in the construction of the classifier. These points are called the support vectors. They support or define the hyperplane.

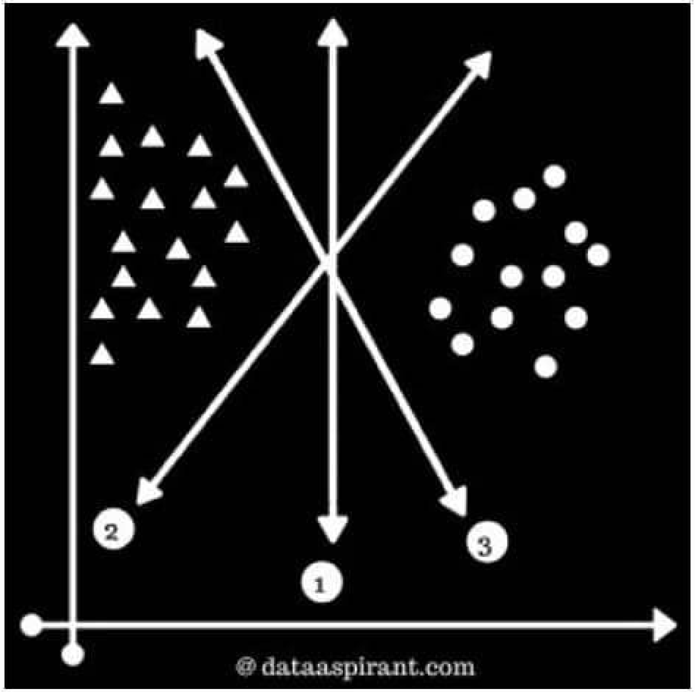
@learn.machinelearning

Support Vector Machines (SVM)

- There are 2 kinds of SVM classifiers:
  - A. Linear SVM Classifier
  - B. Non-Linear SVM Classifier
- Linear SVM Classifier: In the linear classifier model, we assumed that training examples plotted in space. These data points are expected to be separated by an apparent gap. It predicts a straight hyperplane dividing 2 classes. The primary focus while drawing the hyperplane is on maximizing the distance from hyperplane to the nearest data point of either class. The drawn hyperplane called as a maximum-margin hyperplane.
- Non-Linear SVM Classifier: In the real world, our dataset is generally dispersed up to some extent. To solve this problem separation of data into different classes on the basis of a straight linear hyperplane can't be considered a good choice. For this Vapnik suggested creating Non-Linear Classifiers by applying the kernel trick to maximum-margin hyperplanes. In Non-Linear SVM Classification, data points plotted in a higher-dimensional space.

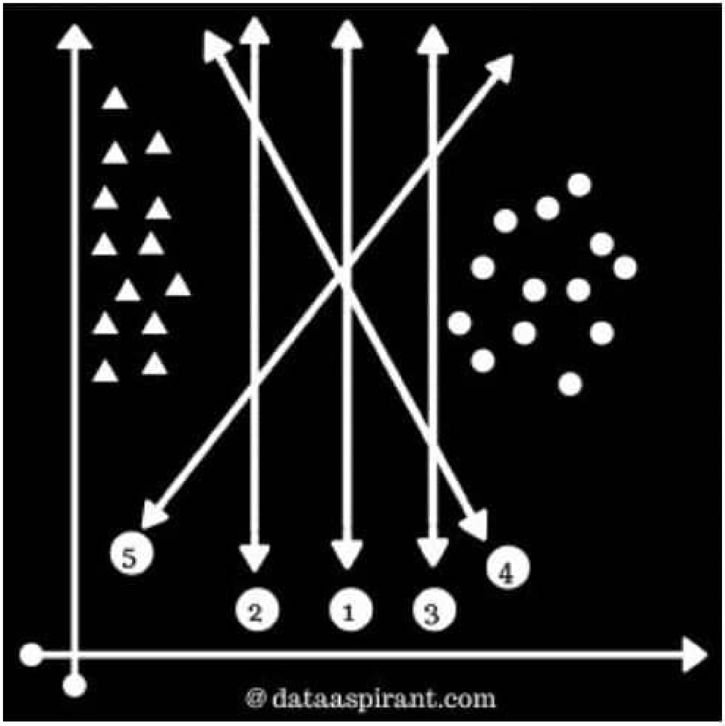
Support Vector Machines (SVM)

- Let's see a few examples.
- Now, we wish to find the best hyperplane which can separate the two classes.
- on the right to find which hyperplane best suit this use case.In SVM, we try to maximize the distance between hyperplane & nearest data point. This is known as margin.



- Since the 1st decision boundary is maximizing the distance between classes on left and right. So, our maximum margin hyperplane will be “1st “.

Support Vector Machines (SVM)

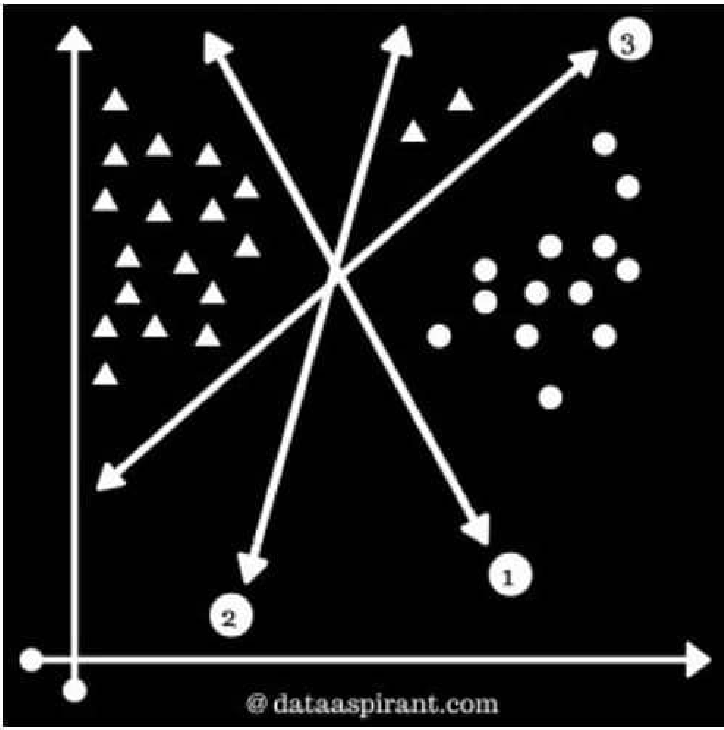


- Now, we wish to find the best hyperplane which can separate the two classes.
- As data of each class is distributed either on left or right. Our motive is to select hyperplane which can separate the classes with maximum margin.

- In this case, all the decision boundaries are separating classes but only 1st decision boundary is showing maximum margin between triangle & circle

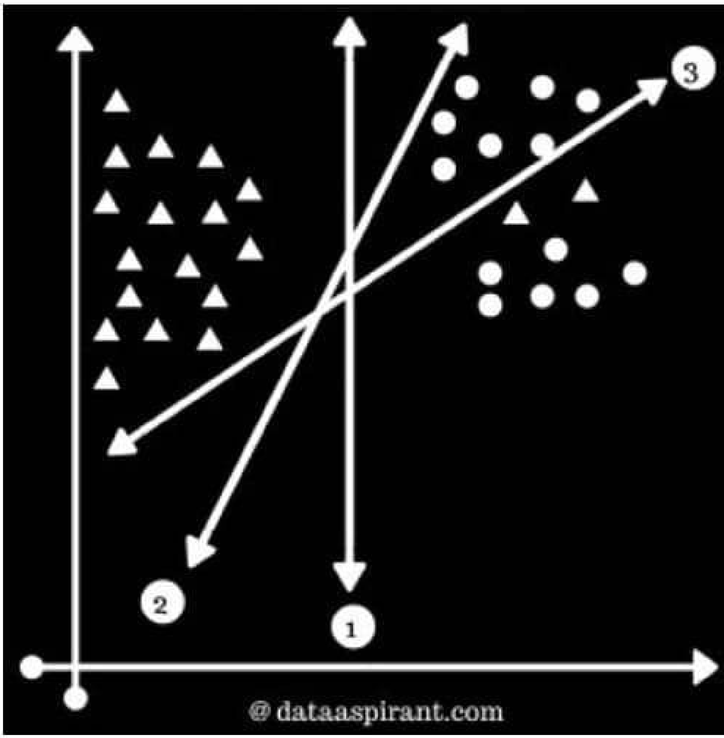
Support Vector Machines (SVM)

- Now, we wish to find the best hyperplane which can separate the two classes.
- Data is not evenly distributed on left and right. Some of the triangle are on right too. You may feel we can ignore the two data points above 3rd hyperplane but that would be incorrect.



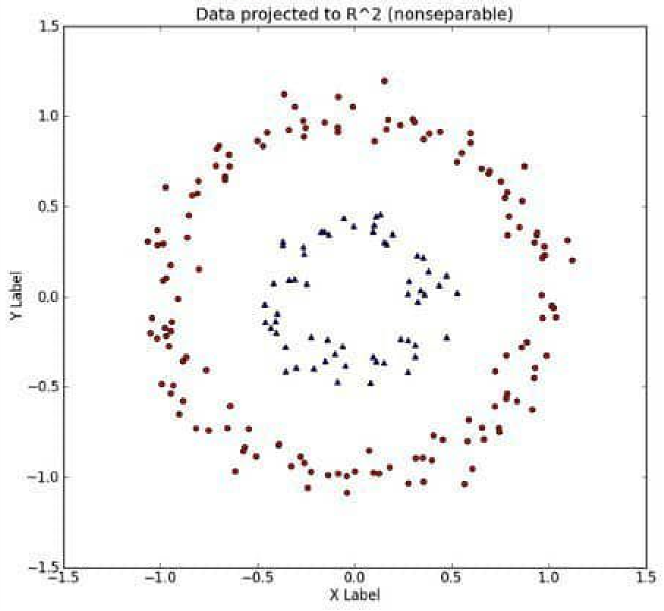
- SVM tries to find out maximum margin hyperplane but gives first priority to correct classification. 1st decision boundary is separating some triangle from the circle but not all. It's not even showing good margin.
- 2nd decision boundary is separating the data points similar to 1st boundary but here margin between boundary and data points is larger than the previous case. So SVM selects 3rd line as best separating line.

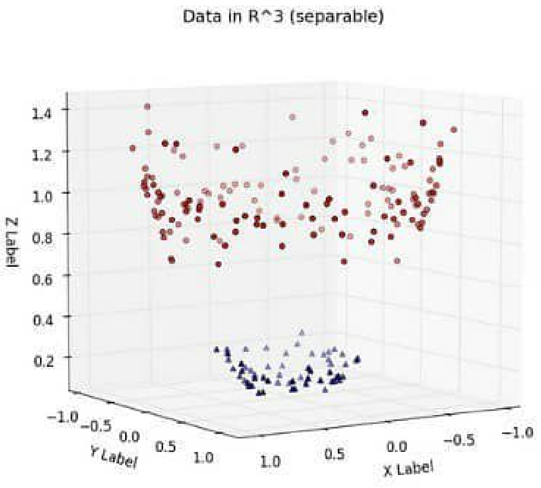
Support Vector Machines (SVM)



- We wish to find the best hyperplane which can separate the two classes.
- Data is not evenly distributed on left and right. Some of the triangles are on right too.
- In the real world, you may find few values that correspond to extreme cases i.e, exceptions. These exceptions are known as Outliers.
- SVM has the capability to detect and ignore outliers. In the image, 2 triangles are in between the group of circles. These triangles are outliers.
- While selecting hyperplane, SVM will automatically ignore these triangles and select best-performing hyperplane. 1st & 2nd decision boundaries are separating classes but 1st decision boundary shows maximum margin in between boundary and support vectors.

Support Vector Machines (SVM)





- In the scenario below, we can’t have linear hyperplane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.
- SVM can solve this problem. Easily! It solves this problem by introducing an additional feature. Here, we will add a new feature z=x^2+y^2. Now, let’s plot the data points on axis x and z.
- In the original plot, red circles appear close to the origin of x and y axes, leading to a lower value of z and star relatively away from the original result to the higher value of z.

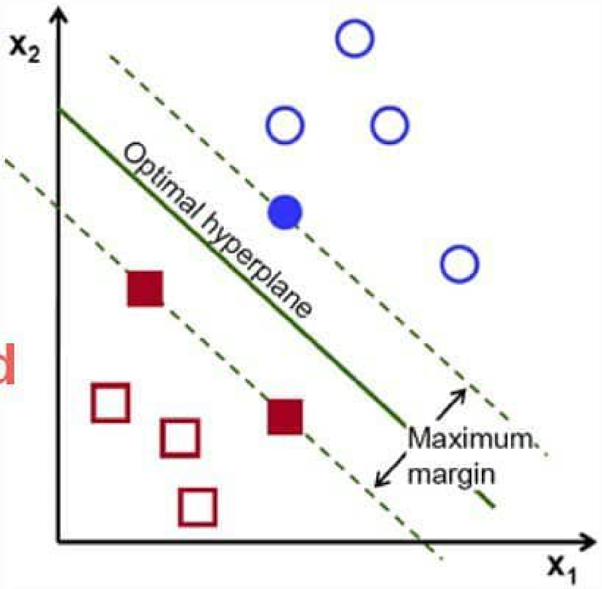
Support Vector Machines (SVM)

- The data points which line on the parallel lines drawn to the margin line are called support vectors which play a major role in SVM.
- Let's see SVM mathematically.
- We can prove that the distance between the parallel lines and the margin live will be same on both sides. The equation of margin(q0) = W^T*X + B = 0 and parallel lines equation are

  Q1 = W^T*X + B = 1 and

  Q2 = W^T*X + B = -1

  W = weight vector

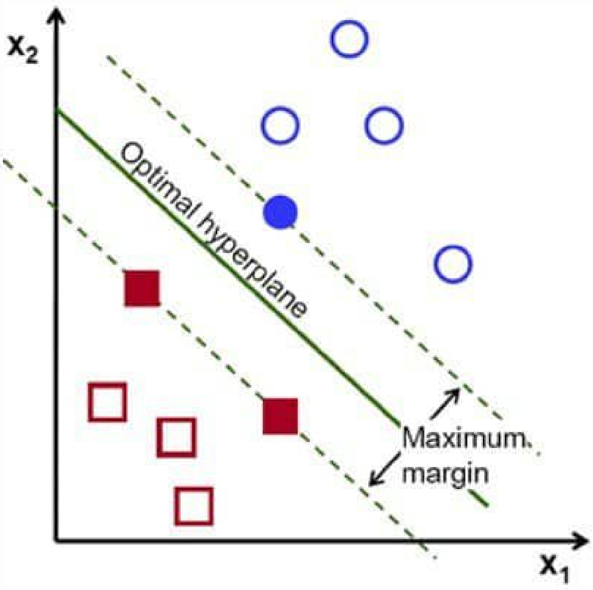


- i.e margin = 2 / ||W|| so finally we want to find the (W*, B*) = aargmax (2 / ||w||) such that all squares are below the lower parallel line and circles are always aove the upper parallel line.

@learn.machinelearning

Support Vector Machines (SVM)

- The equation we considered are margin(q0) = W^T*X + B = 0 and parallel lines equation are Q1 = W^T*X + B = 1 and, Q2 = W^T*X + B = -1 , W = weight vector , we considered 1 and -1 but it can be any value.
- let us consider binary classification problem and let classes be 1 and -1.
- If we consider the points which lie on the parallel lines are called support vectors and they are always equal to 1 and -1 i.e let we take the blue circle which lies on the parallel line the value will be y*(W^T*X + B) = 1 here y = 1 and the red squares on the parallel line the value will be y*(W^T*X + B) = 1 here y = -1.



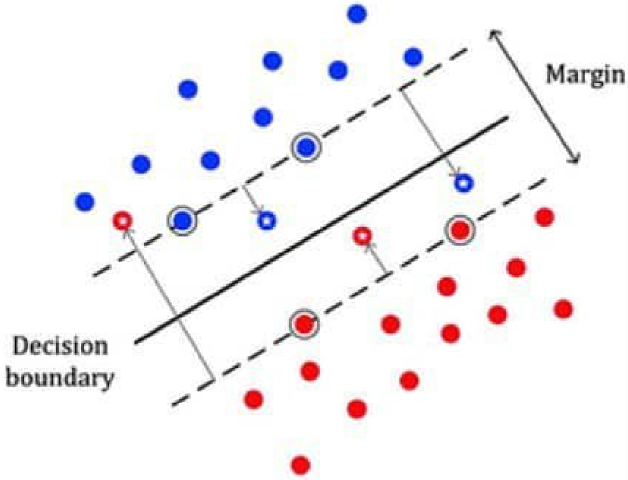
@learn.machinelearning

Support Vector Machines (SVM)

- And the circles which lie far from the parallel line or the squares which lie far from the parallel line(non-support vectors) are always greater than 1 i.e y*(W^T*X + B) > 1
- So finally the optimization problem we are solving is (W*, B*) = argmax 2 / ||W|| such that for all points y*(W^T*X + B) >= 1. W is the weight vector we need to find.
- Until now we see the problem where the 2 classes are well separated and the margin we drew can separate them easily. And the above equation we try to solve always assumes data are well separated.
- But in the real world, we don't get such kind of data at least few points are mixed in other classes. And the above equation which we are trying is called Hard margin SVM.
- Now let's solve the problem using Soft margin SVM

Support Vector Machines (SVM)

- What if data is not linearly separable as shown in the fib below. y for red points is -1 and blue = +1
- Then we use the SOFT margin SVM.
- Here we can see that a few points lie at different locations.
- let's consider the blue circle between the first hyperplane and the margin.



- The equation of that point will be y*(W^T*X + B) <1 and >0 let it be 0.5
- We can also write it as y*(W^T*X + B) = 1 - (0.5)
- The equation of another blue circle which in between the margin and second hyperplane is y*(W^T*X + B) < 0 and > -1 = -0.5 can also be written as y*(W^T*X + B) = 1 - (1.5)
- for the red point in blue region y*(W^T*X + B) < -1 = -1.5 can be y*(W^T*X + B) = 1 - (2.5)

@learn.machinelearning

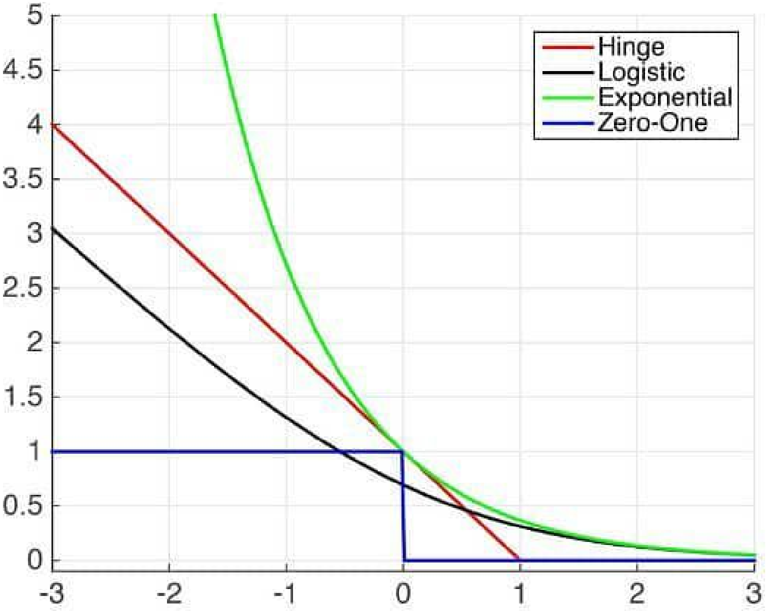
Support Vector Machines (SVM)

- The values inside the brackets are called slack variable (ξi).
- For all other points which are correctly classified the slack variable value is ZERO.
- In HARD margin SVM we are solving the optimization problem as argmax 2 / ||W||.
- Which converted as argmin ||W|| / 2 as mas f(x) = min 1/f(x).
- So the optimization problem in SOFT margin SVM will be argmin ||W|| / 2 + C * 1\n *sum(ξi). such that y*(W^T*X + B) >= 1 - ξi and ξi = 0 for correctly classified points and ξi>0 for miss classified points.
- Here we are trying to minimize the weight vector and the sum of the slack variable. Here C is the same as lambda in logistic regression By maintaining the balance between bias and variance tradeoff.

Support Vector Machines (SVM)

- final equation will be argmin ||W|| / 2 + C * 1\n *sum(ξi) .
- you can see here ||W|| will be working as L2 regularizer and 1\n *sum(ξi)(avg sum of distances between the hyperplane and the points) is like loss function where we need to minimize this loss. And C is a hyperparameter to maintain the balance between bias and variance.
- If C increases you are giving more importance to making mistakes and you overfit.If C decreases you are giving more importance to regularizer and you underfit.
- So why we got ||W||/2 (2) here. That is because of the equations we had taken are Q0 = y*(W^T*X + B) = 0,
 Q1 = y*(W^T*X + B) = 1 and Q2 = y*(W^T*X + B) = -1.
- So the distance between the margin and hyperplane will be 1 on both sides as we took 1 and -1 values for hyperplanes. so we had taken 2. if you use any value in the equations then we get 2*(that value).

Support Vector Machines (SVM)



- Let's solve SVM with Hinge loss. You can see that the values which are greater than 1 are zero.
- If y*(W^T*X + B) >=1 then hinge loss = 0
- If y*(W^T*X + B) > 1 then hinge loss = 1-y*(W^T*X + B)
- we can also write this as max(0, 1- y*(W^T*X + B)).
- so the final equation will be

  argmin sum(max(0, 1- y*(W^T*X + B))) + regularizer
- If you see closly both soft SVM and Hinge Loss works similar.

@learn.machinelearning

Support Vector Machines (SVM)

- Now let's discuss the Non-Linear Support Vector Machine.
- As per the previous example(No 5) you can see that data is not linear and cannot be solved using soft SVM.
- So we added new features to map the data from low dimension to high dimension and now the data can be easily separable.
- As dimension increases the density in data deceases and sparsity increases.
- And this technique which maps the low dimension data to high dimension data is called kernel trick.
- There are many kernels that have been developed. Some standard kernels are
  1. Polynomial kernel
  2. RBF kernel (Radial Basis Function)
  3. Domain-specific kernels.

Support Vector Machines (SVM)

- **Polynomial Kernel SVM:** Instead of the dot-product, we can use a polynomial kernel, for example. K(x,xi) = 1 + sum(x * xi)^d. The polynomial kernel function can be represented by the above expression. Where k(xi, xj) is a kernel function, xi & xj are vectors of feature space and d is the degree of a polynomial function.
- **Radial Basis Function Kernel:** It is also known as the RBF kernel. It is one of the most popular kernels. For distance metric squared euclidean distance is used here. It is used to draw completely non-linear hyperplanes. K(x,xi) = exp(-gamma * sum((x – xi^2)). Where gamma is a parameter that must be specified to the learning algorithm. A good default value for gamma is 0.1, where gamma is often 0 < gamma < 1.
- If you don't know which kernel to use then use RBF kernel.

Support Vector Machines (SVM)

- **Advantages of SVM**
- **SVMs are effective when the number of features is quite large.**
- **It works effectively even if the number of features is greater than the number of samples.**
- **Non-Linear data can also be classified using customized hyperplanes built by using kernel trick.**
- **It is a robust model to solve prediction problems since it maximizes margin.**
- **It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.**

Support Vector Machines (SVM)

- **Disadvantages of SVM**
- The biggest limitation of Support Vector Machine is the choice of the kernel. The wrong choice of the kernel can lead to an increase in error percentage.
- With a greater number of samples, it starts giving poor performances.
- SVMs have good generalization performance but they can be extremely slow in the test phase.
- SVMs have high algorithmic complexity and extensive memory requirements due to the use of quadratic programming.
- It also doesn’t perform very well, when the data set has more noise i.e. target classes are overlapping
- SVM doesn’t directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

Support Vector Regression (SVR)

- We will do the same thing finding the hyperplane.
- The final question will be argmin ||W||/2 such that
  Y - Yhat <= epsilon or Yhat - y <=epsilon and epsilon >= 0. Y is orginal value and yhat is predicted value and epsilon is threshold value.
- Here epsilon is hyperparameter and we need to find that.
- If epsilon value increases then you are thinking that distance between the original value and the predicted value will be more and you are not learning anything, which means you are underfitting the model.
- If epsilon value decreases then you are thinking that distance between the original value and the predicted value will be less, which means you are overfitting the model.
- We also have Kernel SVR to find the non-linear line which fits the data.