Andhra Pradesh State Skill Development Corporation

Skill AP
APSSDC

Skill AP
Learn Anytime Anywhere
Andhra Pradesh State Skill Development Corporation

Machine Learning

Support Vector Machines

CHAPTER 10

Support Vector Machines

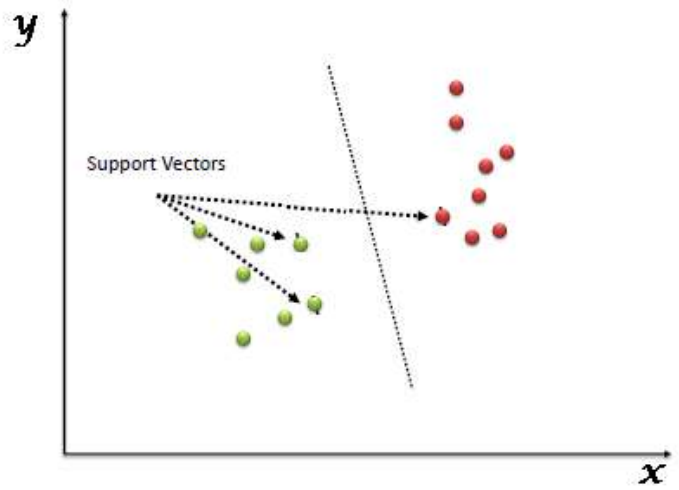
Introduction to Support Vector Machines

SVM is a model, which can do linear classification as well as regression. SVM is based on the concept of a surface, called a hyperplane, which draws a boundary between data instances plotted in the multi-dimensional feature space. The output prediction of an SVM is one of two conceivable classes which are already defined in the training data. In summary, the SVM algorithm builds an N-dimensional hyperplane model that assigns future instances into one of the two possible output classes

Understanding SVM Model

What is a Support Vector Machine?

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).



Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier which best segregates the two classes (hyper-plane/ line).

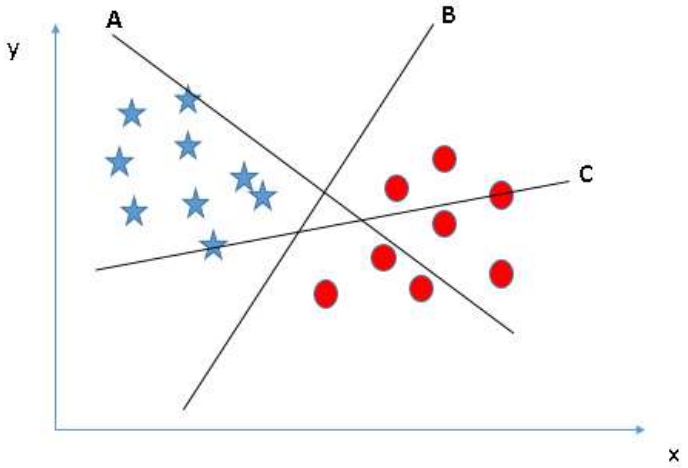
You can look at support vector machines and a few examples of its working here.

How does it work?

Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is "How can we identify the right hyper-plane?". Don't worry, it's not as hard as you think!

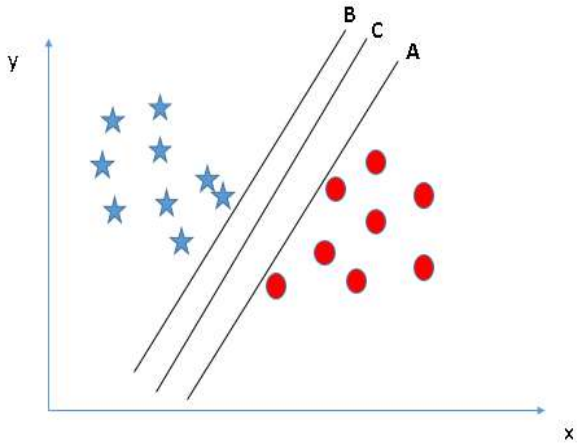
Let's understand:

- **Identify the right hyper-plane (Scenario-1):** Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

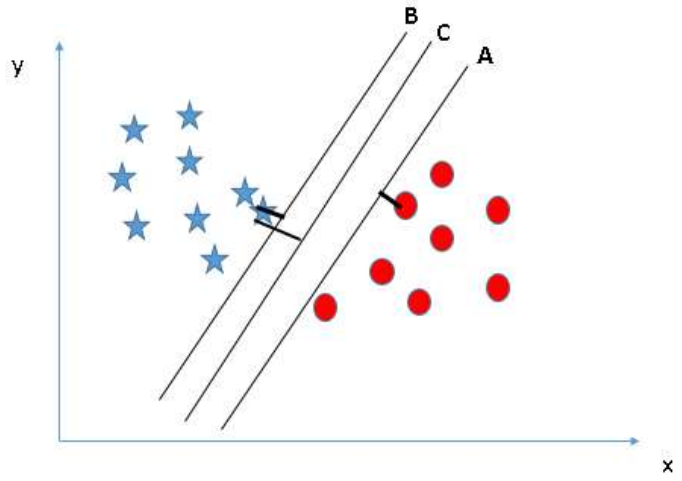


- You need to remember a thumb rule to identify the right hyper-plane: "Select the hyper-plane which segregates the two classes better". In this scenario, hyper-plane "B" has excellently performed this job.

- **Identify the right hyper-plane (Scenario-2):** Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

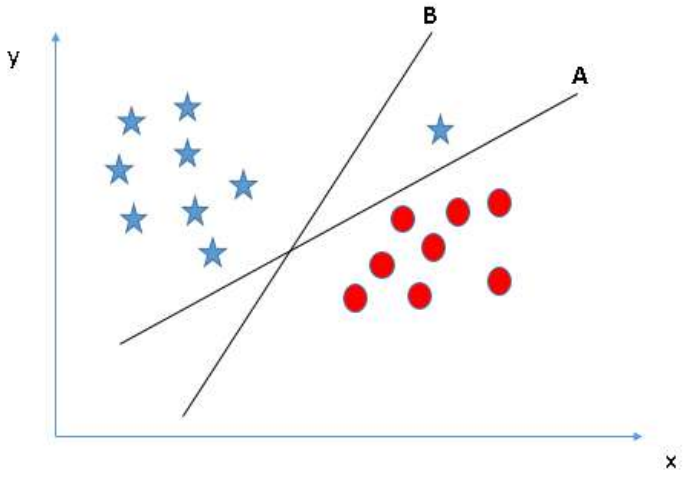


- Here, maximizing the distances between the nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called Margin. Let' look at the below snapshot:



Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is a high chance of miss-classification.

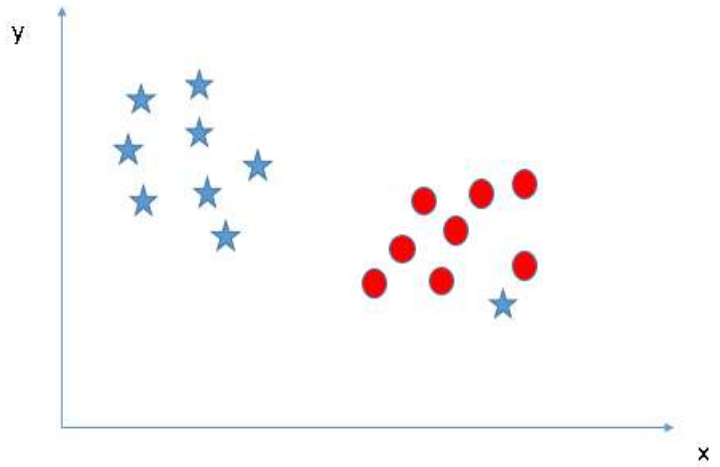
- **Identify the right hyper-plane (Scenario-3):**Hint: Use the rules as discussed in previous section to identify the right hyper-plane



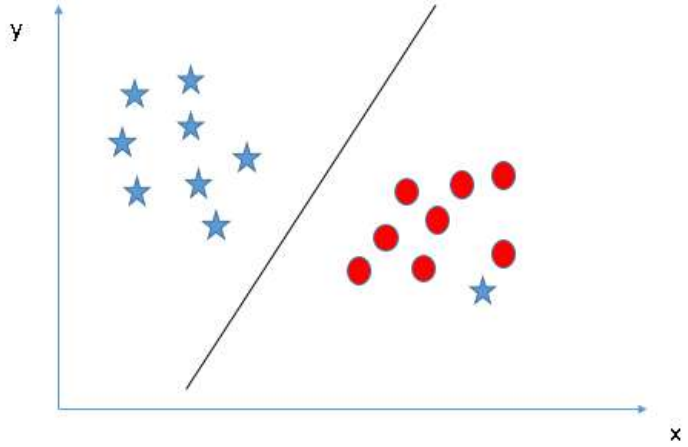
Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A.** But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

- **Can we classify two classes (Scenario-4)?:**Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of another(circle) class

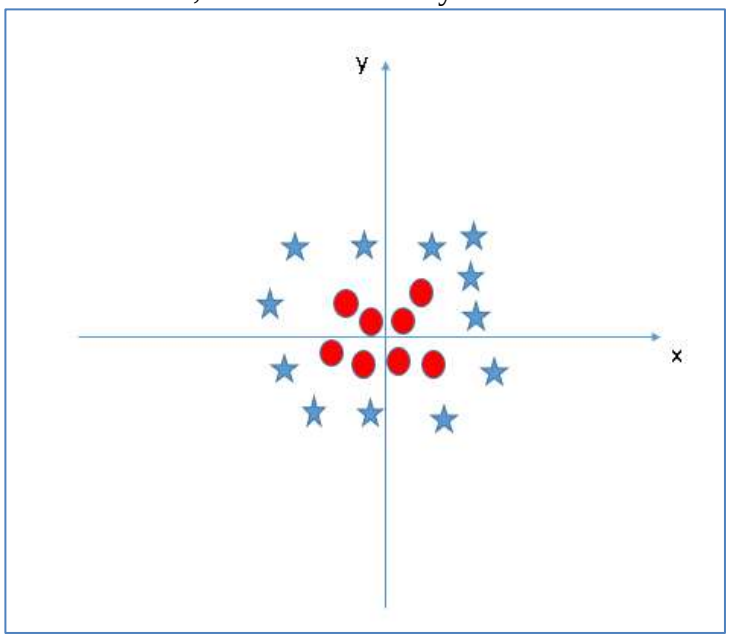
as an outlier.



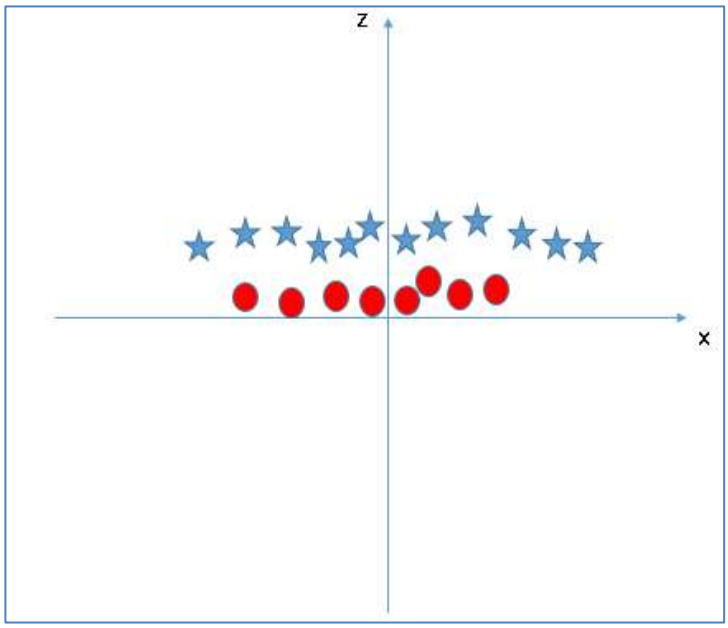
- As I have already mentioned, one star at the other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.



- **Find the hyper-plane to segregate to classes (Scenario-5):** In the scenario below, we can't have a linear hyper-plane 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 additional features. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:

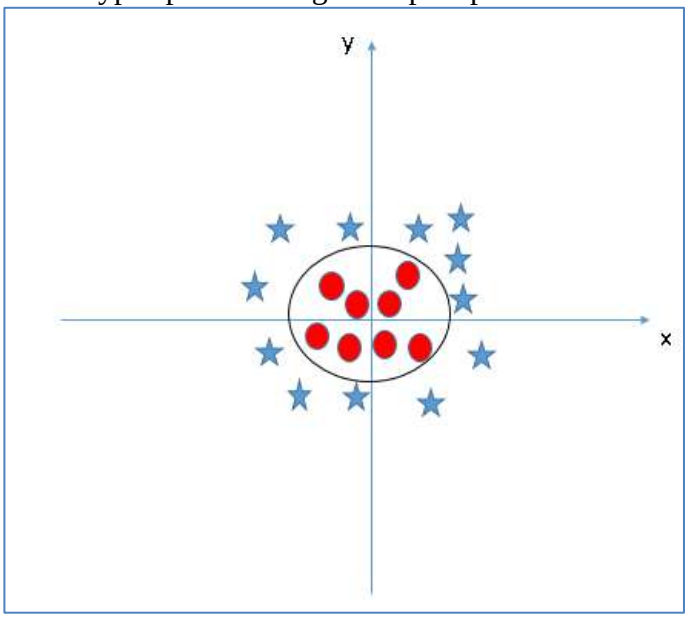


In above plot, points to consider are:

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

- In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM algorithm has a technique called the **kernel trick**. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problems. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you've defined.

When we look at the hyper-plane in original input space it looks like a circle:

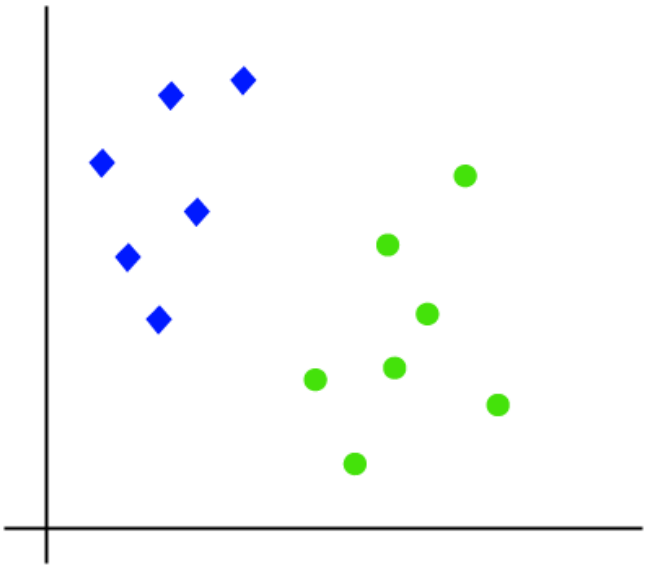


Now, let's look at the methods to apply the SVM classifier algorithm in a data science challenge.

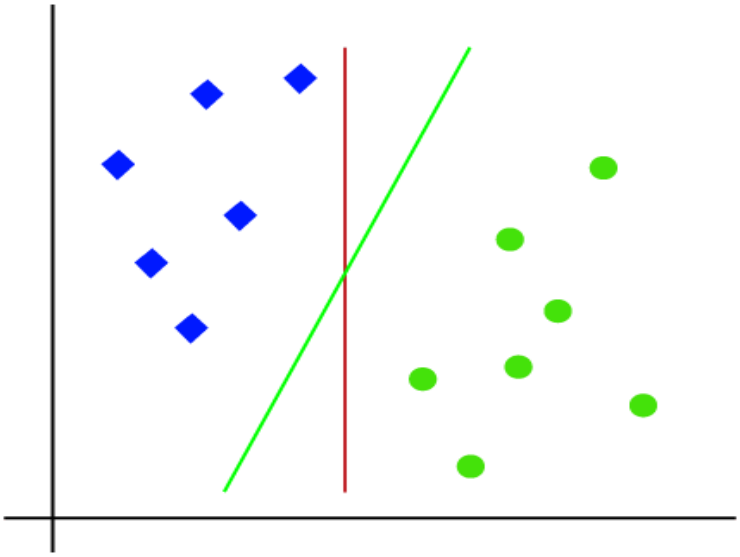
Understanding Linear SVM Classification

Linear SVM:

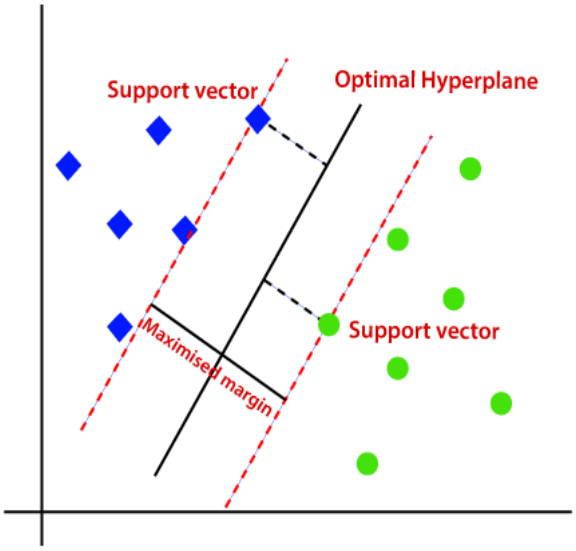
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



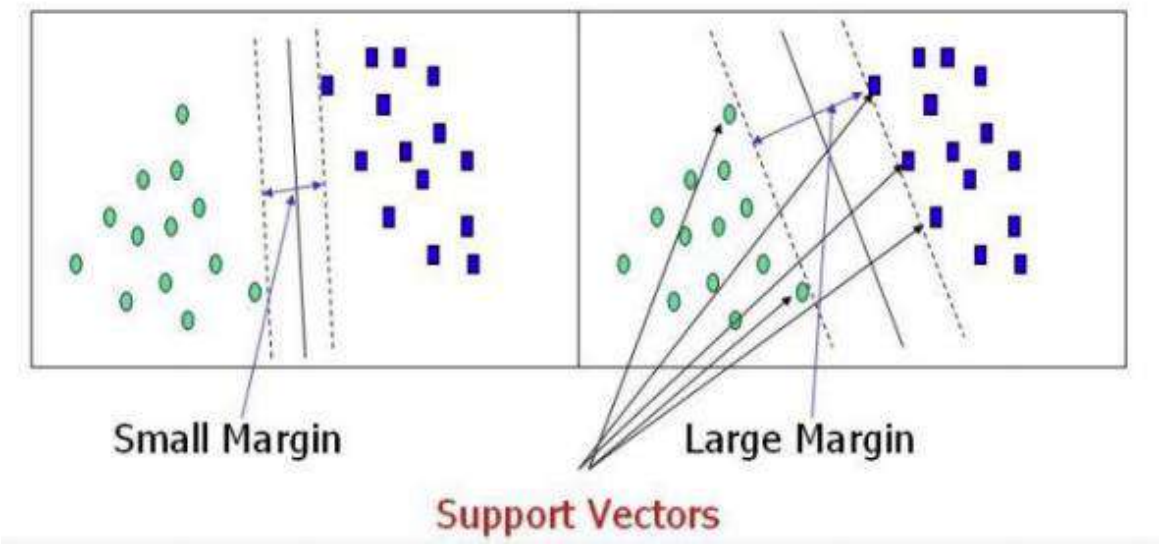
Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



So we take scikit-learn builtin dataset make_blobs,generate 50 number of points equally divided among clusters and understand the implementation of Support Vector Machine.

Building Linear SVM Classification

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.



So we plot decision function using Support Vector Classifier and also generate support vectors having maximum margin from the Optimal Hyperplane.We create a Contour plot is a graphical technique for representing a 3-dimensional surface by plotting constant z slices, called contours, on a 2-dimensional format. That is, given a value for z, lines are drawn for connecting the (x,y) coordinates where that z value occurs,where Z is the labels now we have 3 different points which indicates that the Support vector to the left or the hyper plane in middle or Support vector to the right.30 points in each direction 30 times 30 which gives 900 points of data and we separate it between the three lines and we reshaped it to fit those three lines

Non Linear SVM Classification - Data Collection and Preprocessing

SVM can be extended to solve nonlinear classification tasks when the set of samples cannot be separated linearly. By applying kernel functions, the samples are mapped onto a high-dimensional feature space, in which the linear classification is possible.So we take wine quality dataset and understand implementation process.The variable quality rating
is considered as dependent variable and other 11 variables are assumed as predictors or independent variables

Data collection is the process of gathering and measuring information from countless different sources. In order to use the data we collect to develop practical artificial intelligence (AI) and machine learning solutions, it must be collected and stored in a way that makes sense for the business problem at hand.

PREPROCESSING Preprocessing can be done by several methods. Real world data are generally inconsistent, noisy and incomplete data. In this system filtering method is used for data reduction (removing the repeated data), mean method is used for incomplete data (i.e., missing values) and also removes the inconsistent data (impossible data combination).

Modelling

In almost any Machine Learning project, we train different models on the dataset and selecting the one with the best performance. However, there is almost a room for improvement as we cannot say for sure that this particular model is best for the problem at hand, hence our aim is to improve the model in any way possible. One important factor in the performances of these models are their hyperparameters, once we set appropriate values for these hyperparameters, the performance of a model can improve significantly. We evaluate the model performance and we will also find out how we can find optimal values for the hyperparameters of a model by using **GridSearchCV.**

It is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. As mentioned above, the performance of a model significantly depends on the value of hyperparameters. Note that there is no way to know in advance the best values for hyperparameters so ideally, we need to try all possible values to know the optimal values.

Doing this manually could take a considerable amount of time and resources and thus we use GridSearchCV to automate the tuning of hyperparameters.GridSearchCV is a function that comes in Scikit-learn's(or SK-learn) model_selection package.As we already have scikit-learn package installed, this function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

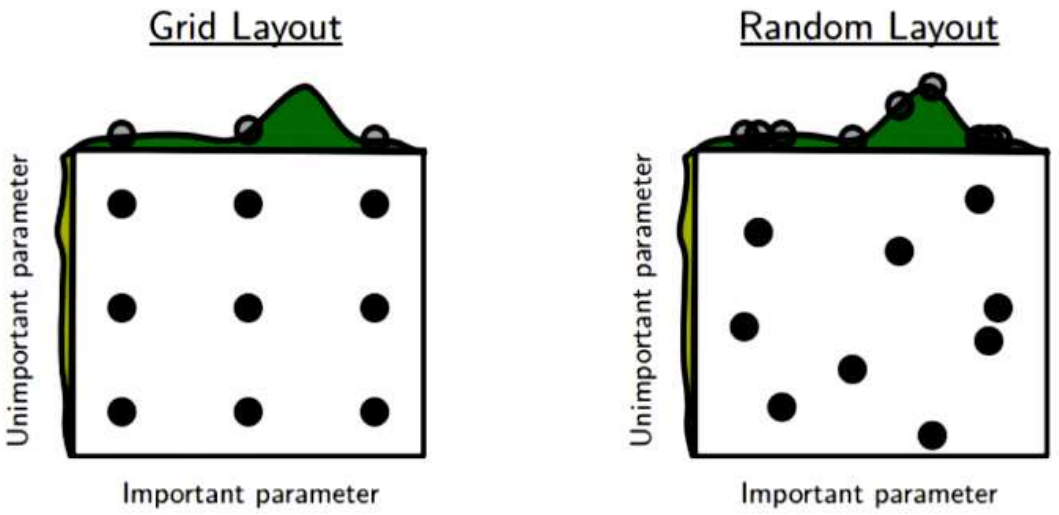
We do this by defining a dictionary in which we mention a particular hyperparameter along with the values it can take. Here is an example of it

```
{ 'C': [0.1, 1, 10, 100, 1000],
 'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
 'kernel': ['rbf','linear','sigmoid'] }
```

Here C, gamma and kernels are some of the hyperparameters of an SVM model. Note that the rest of the hyperparameters will be set to their default values

GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method. Hence after using this function we get accuracy/loss for every combination of hyperparameters and we can choose the one with the best performance.

While it's possible that RandomizedSearchCV will not find as accurate of a result as GridSearchCV, it surprisingly picks the best result more often than not and in a *fraction* of the time it takes GridSearchCV would have taken. Given the same resources, Randomized Search can even outperform Grid Search. This can be visualized in the graphic below when continuous parameters are used.

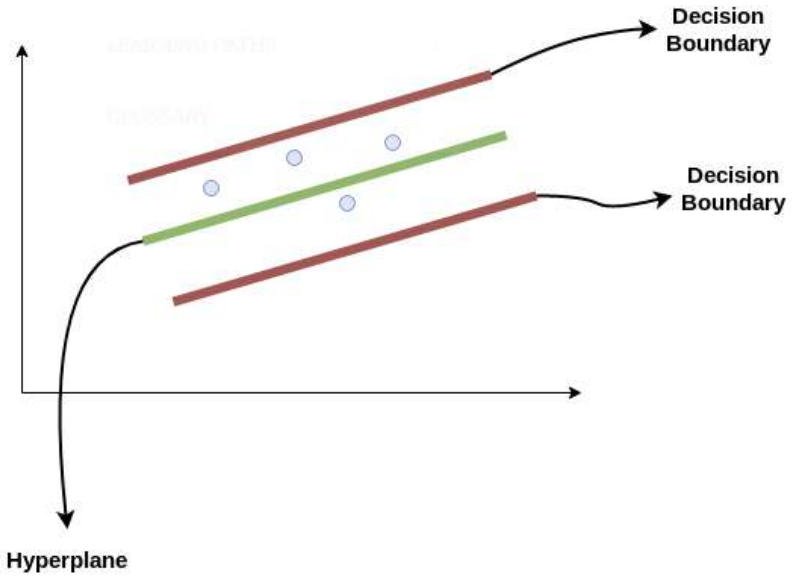


With grid search, nine trials only test three distinct places. With random search, all nine trails explore distinct values.

SVM Regression - Introduction and Data Collection

Support Vector Machine (SVM) is a very popular Machine Learning algorithm that is used in both Regression and Classification. Support Vector Regression is similar to Linear Regression in that the equation of the line is y= wx+b In SVR, this straight line is referred to as ***hyperplane***. The data points on either side of the hyperplane that are closest to the hyperplane are called ***Support Vectors*** which is used to plot the boundary line.

Unlike other Regression models that try to minimize the error between the real and predicted value, the SVR tries to fit the best line within a threshold value (Distance between hyperplane and boundary line), **a**. Thus, we can say that the SVR model tries to satisfy the condition -a < y-wx+b < a. It used the points with this boundary to predict the value.The problem of regression is to find a function that approximates mapping from an input domain to real numbers on the basis of a training sample.



Consider these two red lines as the decision boundary and the green line as the hyperplane. **Our objective, when we are moving on with SVR, is to basically consider the points that are within the decision boundary line.** Our best fit line is the hyperplane that has a maximum number of points.

The first thing that we'll understand is what is the decision boundary (the danger red line above!). Consider these lines as being at any distance, say 'a', from the hyperplane. So, these are the lines that we draw at distance '+a' and '-a' from the hyperplane. This 'a' in the text is basically referred to as epsilon.

Assuming that the equation of the hyperplane is as follows:

Y = wx+b (equation of hyperplane)

Then the equations of decision boundary become:

wx+b= +a
wx+b= -a

Thus, any hyperplane that satisfies our SVR should satisfy:

-a < Y- wx+b < +a

Our main aim here is to decide a decision boundary at 'a' distance from the original hyperplane such that data points closest to the hyperplane or the support vectors are within that boundary line.Hence, we are going to take only those points that are within the decision boundary and have the least error rate, or are within the Margin of Tolerance. This gives us a better fitting model.Thus we work on abalone dataset to implement it.

Removing Outliers and Modelling

As Outliers are unusual values in your dataset, and they can distort statistical analyses and violate their assumptions. Unfortunately, all analysts will confront outliers and be forced to make decisions about what to do with them. Given the problems they can cause, you might think that it's best to remove them from your data. But, that's not always the case. Removing outliers is legitimate only for specific reasons.

It's essential to understand how outliers occur and whether they might happen again as a normal part of the process or study area. Unfortunately, resisting the temptation to remove outliers inappropriately can be difficult. Outliers increase the variability in your data, which decreases statistical power. Consequently, excluding outliers can cause your results to become statistically significant.

When considering whether to remove an outlier, you'll need to evaluate if it appropriately reflects your target population, subject-area, research question, and research methodology. Did anything unusual happen while measuring these observations, such as power failures, abnormal experimental conditions, or anything else out of the norm? Is there anything substantially different about an observation, whether it's a person, item, or transaction? Did measurement or data entry errors occur?

If the outlier in question is:

- A measurement error or data entry error, correct the error if possible. If you can't fix it, remove that observation because you know it's incorrect.
- Not a part of the population you are studying (i.e., unusual properties or conditions), you can legitimately remove the outlier.]
- A natural part of the population you are studying, you should not remove it.

When you decide to remove outliers, document the excluded data points and explain your reasoning. You must be able to attribute a specific cause for removing outliers. Another approach is to perform the analysis with and without these observations and discuss the differences. Comparing results in this manner is particularly useful when you're unsure about removing an outlier and when there is substantial disagreement within a group over this question.

Then we go for FeatureSelection as it is a technique where we choose those features in our data that contribute most to the target variable. In other words we choose the best predictors for the target variable.The classes in the **sklearn.feature_selection** module can be used for feature selection/dimensionality reduction on sample sets, either to improve estimators' accuracy scores or to boost their performance on very high-dimensional datasets.Thus we use SelectKBest method as it select features according to the k highest scores. Finally we compare RootMeanSquareError for different models.