[**https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/**](https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/)

**What is Support Vector Machine?**

“Support Vector Machine” (SVM) is a supervised [machine learning algorithm](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2?utm_source=blog&utm_medium=understandingsupportvectormachinearticle) 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).

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_1.png)

Support Vectors are simply the co-ordinates 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](https://www.analyticsvidhya.com/blog/2014/10/support-vector-machine-simplified/?utm_source=blog&utm_medium=understandingsupportvectormachinearticle) 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 nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**. Let’s look at the below snapshot:[[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_4.png)](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_4.png)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 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

**[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_5.png)**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 other(circle) class as an outlier.  **[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_61.png)**As I have already mentioned, one star at 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.  
  **[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_71.png)**
* **Find the hyper-plane to segregate to classes (Scenario-5):**In the scenario below, we can’t have 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.**[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_8.png)**SVM can solve this problem. Easily! It solves this problem by introducing 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:  
  [[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_9.png)](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_9.png)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.

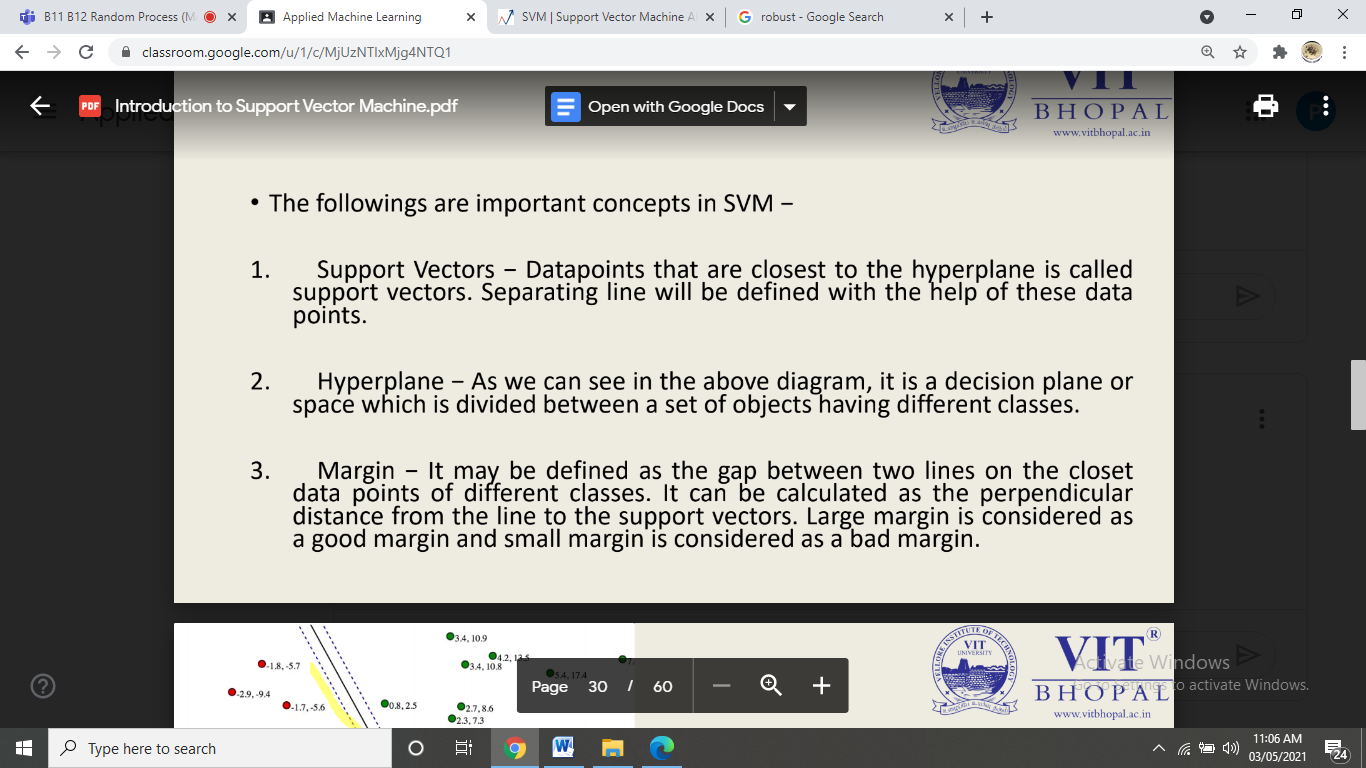
In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM  algorithm has a technique called the [**kernel**](https://en.wikipedia.org/wiki/Kernel_method)**trick**. 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 problem. 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.

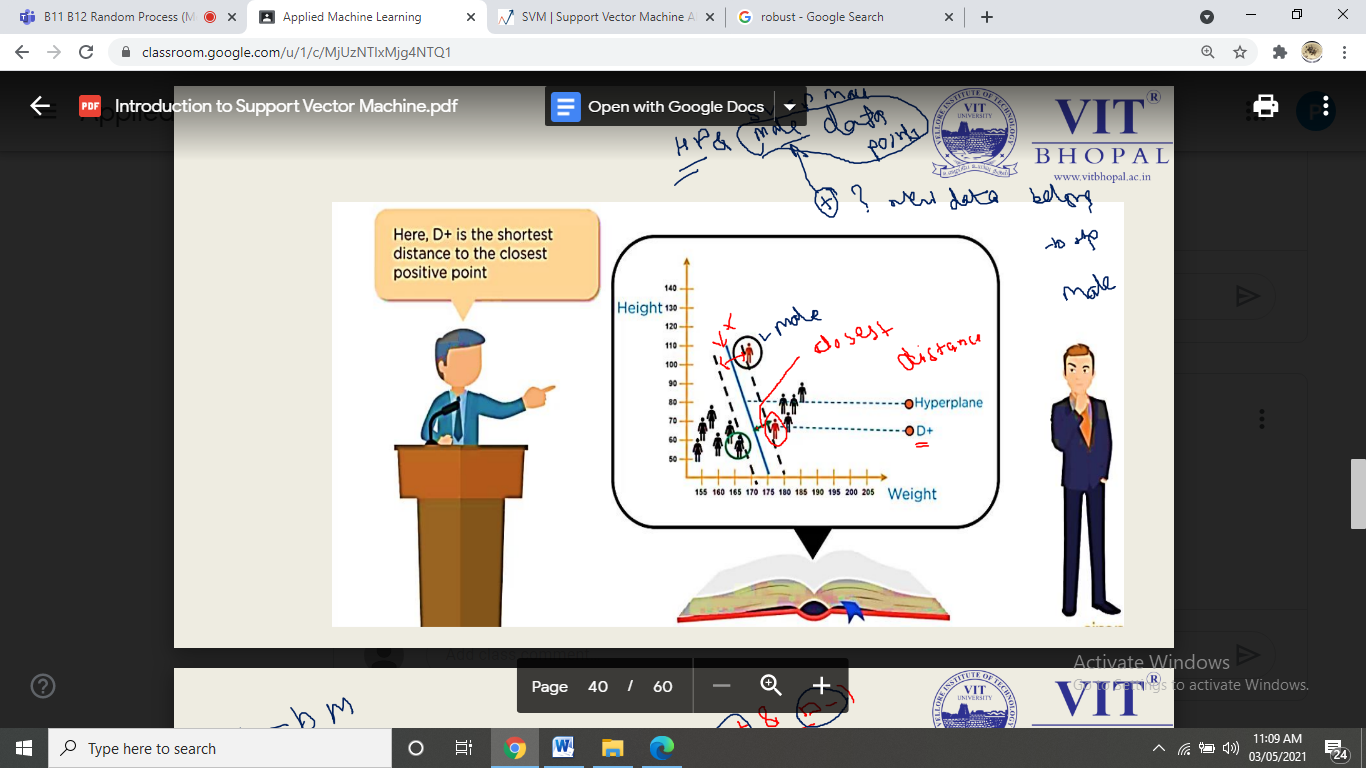
When we look at the hyper-plane in original input space it looks like a circle:  
[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_10.png)

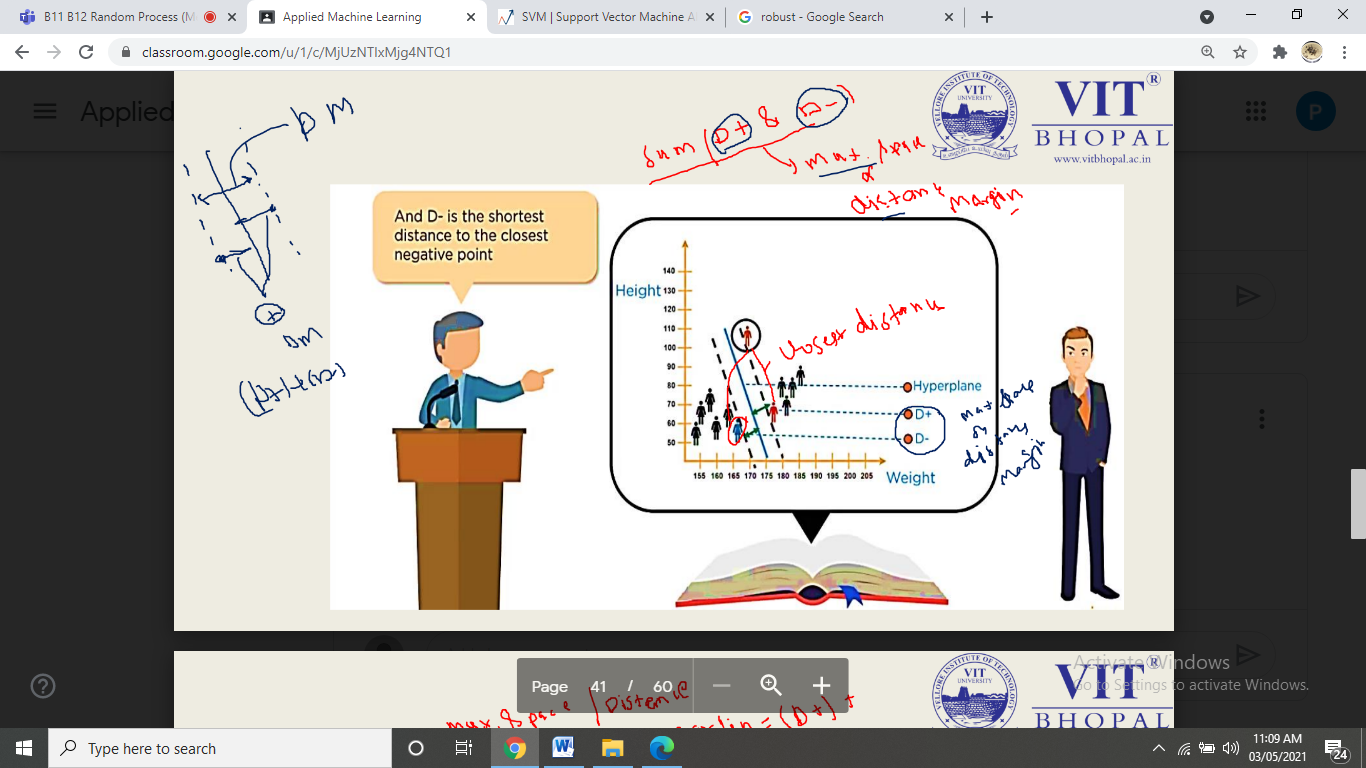
Now, let’s look at the methods to apply SVM classifier algorithm in a data science challenge.

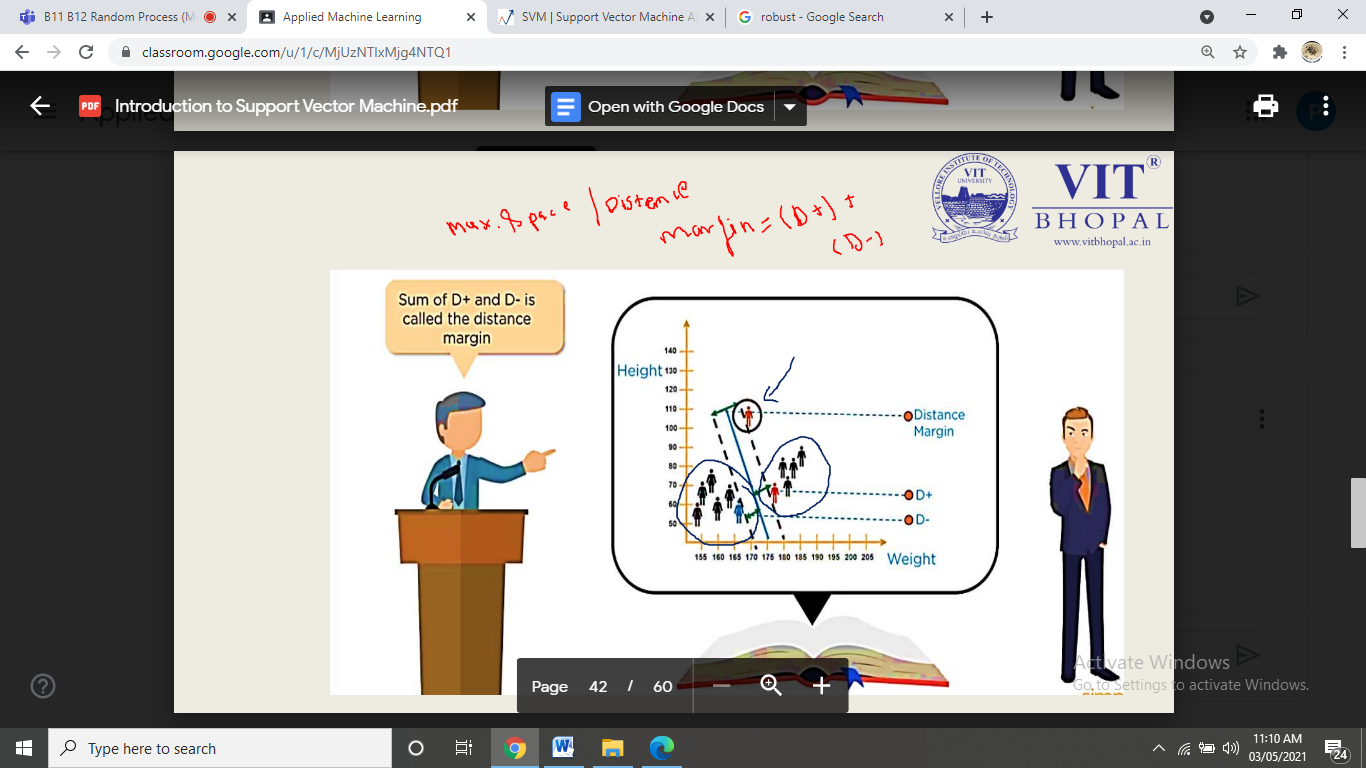
**Pros and Cons associated with SVM**

* **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 uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
* **Cons:**
  + It doesn’t perform well when we have large data set because the required training time is higher
  + 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. It is included in the related SVC method of Python scikit-learn library.

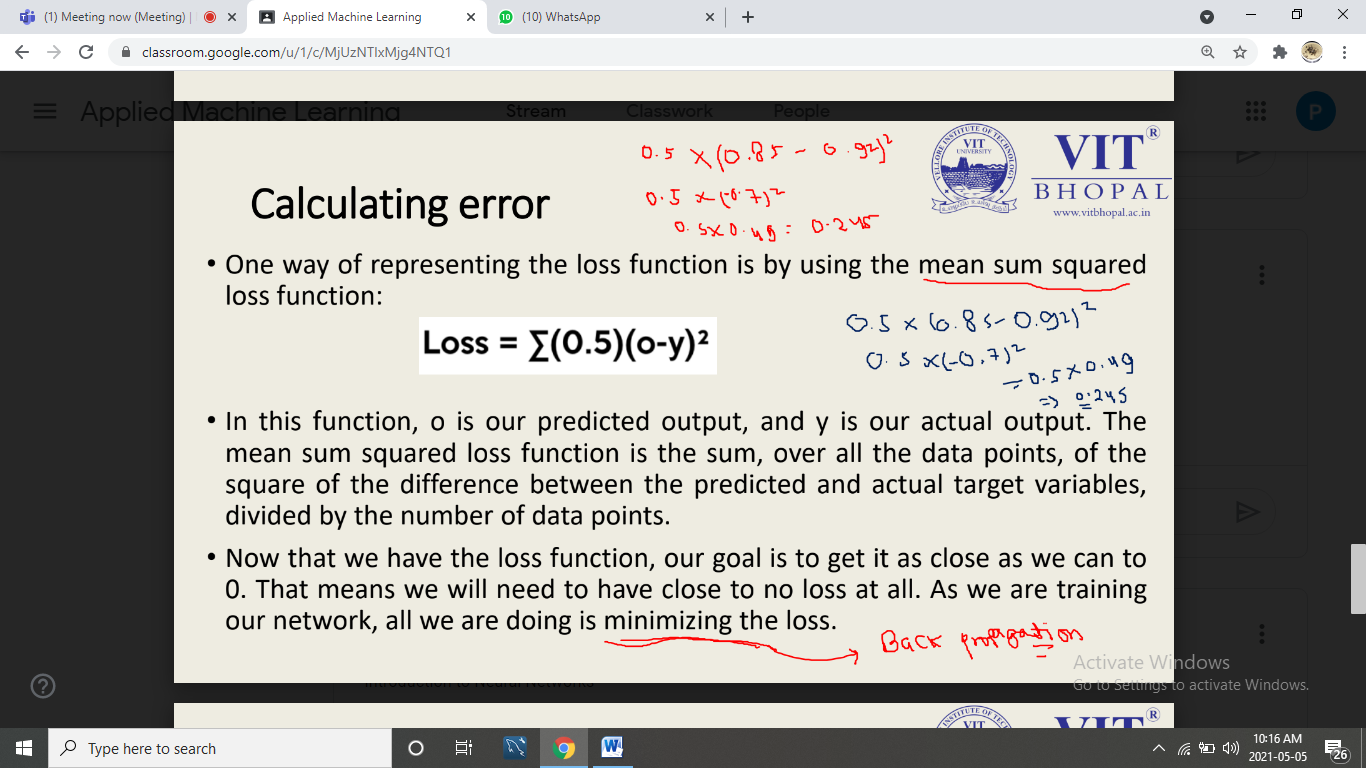








ANN:



Decision tree:

The primary **difference** between classification and **regression decision** trees is that, the classification **decision** trees are built with unordered values with dependent variables. ... In case of **regression decision tree** algorithm, the variable Cl takes the ordered values instead of unordered values.

Regression and classification: <https://www.javatpoint.com/regression-vs-classification-in-machine-learning>

The **main difference between Regression** and **Classification** algorithms that **Regression** algorithms are used to predict the continuous values such as price, salary, age, etc. and **Classification** algorithms are used to predict/**Classify** the discrete values such as Male or Female, True or False, Spam or Not Spam, etc.

**A general algorithm for a decision tree can be described as follows:**

Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

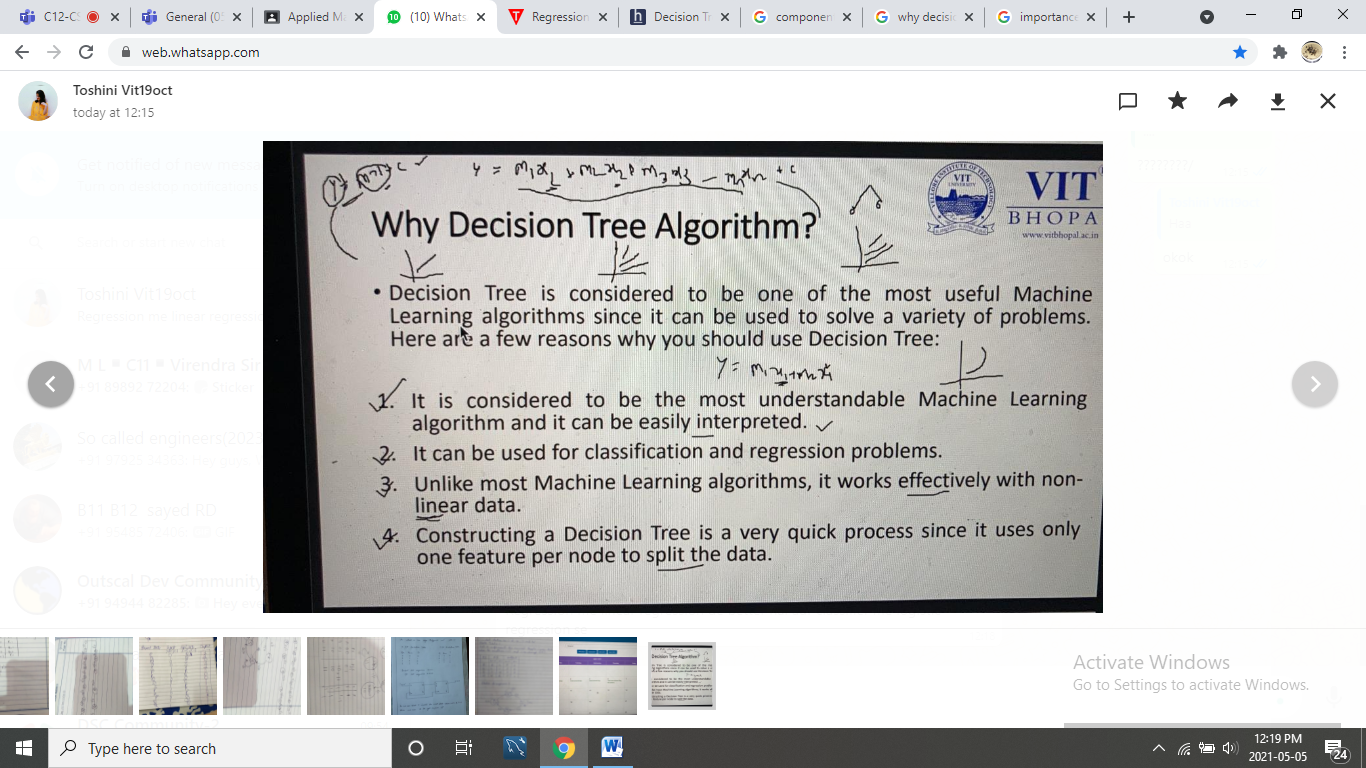
The decision rules are generally in form of if-then-else statements. The deeper the tree, the more complex the rules and fitter the model.

1. Pick the best attribute/feature. The best attribute is one which best splits or separates the data.
2. Ask the relevant question.
3. Follow the answer path.
4. Go to step 1 until you arrive to the answer.

The best split is one which separates two different labels into two sets.

**Hidden layers**, simply put, are **layers** of mathematical functions each designed to produce an output specific to an intended result. ... **Hidden layers** allow for the function of a **neural** network to be broken down into specific transformations of the data. Each **hidden** layer function is specialized to produce a defined output. see ppt also

An inordinately large **number** of **neurons** in the **hidden layers** can **increase** the time it takes to train the **network**. The **amount** of training time can **increase** to the point that it is impossible to adequately train the **neural network**.



DT:

Regression me linear regression de better hai aur classifier me logistic regression se

**Over-fitting** is the phenomenon in which the learning system tightly fits the given training data so much that it would be inaccurate in predicting the outcomes of the untrained data. In **decision trees**, **over-fitting** occurs when the **tree** is designed so as to perfectly fit all samples in the training data set.

**Underfitting** occurs when a model is too simple — informed by too few features or regularized too much — which makes it inflexible in learning from the dataset. Simple learners tend to have less variance in their predictions but more bias towards wrong outcomes.

If "Accuracy" (measured against the training set) is very good and "Validation Accuracy" (measured against a validation set) is not as good, then your model is **overfitting**. **Underfitting** is the opposite counterpart of **overfitting** wherein your model exhibits high

**Hyper parameter of svm: Gamma decides that how much curvature we want in a decision boundary. Gamma high means more curvature. ... C is a hypermeter which is set before the training model and used to control error and Gamma is also a hypermeter which is set before the training model and used to give curvature weight of the decision boundary**.

# How can we prune the above tree?

Decision trees in general will continue to form branches till every node becomes homogeneous. As a result of this, the tree works well with the training data but fails to produce quality output for the test data. Hence the tree should be pruned to prevent overfitting.

Pruning or post-pruning

As the name implies, pruning involves cutting back the tree. After a tree has been built (and in the absence of early stopping discussed below) it may be overfitted. The CART algorithm will repeatedly partition data into smaller and smaller subsets until those final subsets are homogeneous in terms of the outcome variable. In practice this often means that the final subsets (known as the *leaves* of the tree) each consist of only one or a few data points. The tree has learned the data exactly, but a new data point that differs very slightly might not be predicted well.

I will consider 3 pruning strategies,

* *Minimum error*. The tree is pruned back to the point where the cross-validated error is a minimum. *Cross-validation* is the process of building a tree with most of the data and then using the remaining part of the data to test the accuracy of the decision tree.
* *Smallest tree*. The tree is pruned back slightly further than the minimum error. Technically the pruning creates a decision tree with cross-validation error within 1 standard error of the minimum error. The smaller tree is more intelligible at the cost of a small increase in error.
* None.