

# MACHINE LEARNING ALGORITHM

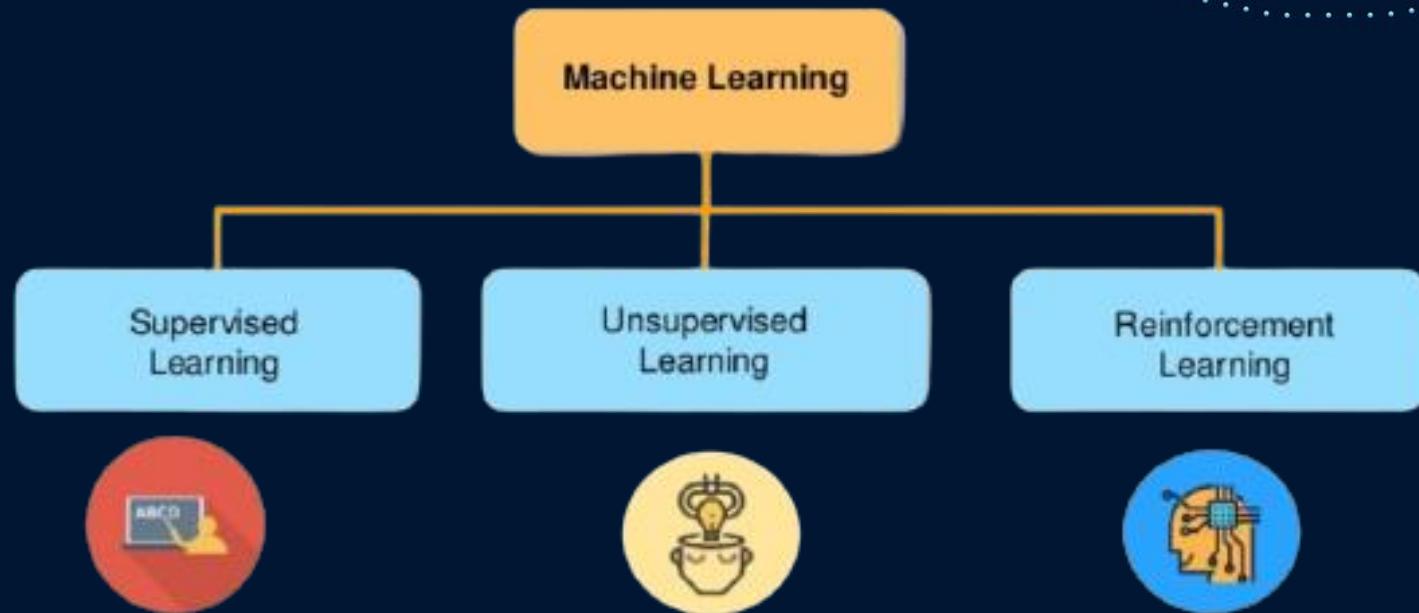
---

Unit-III

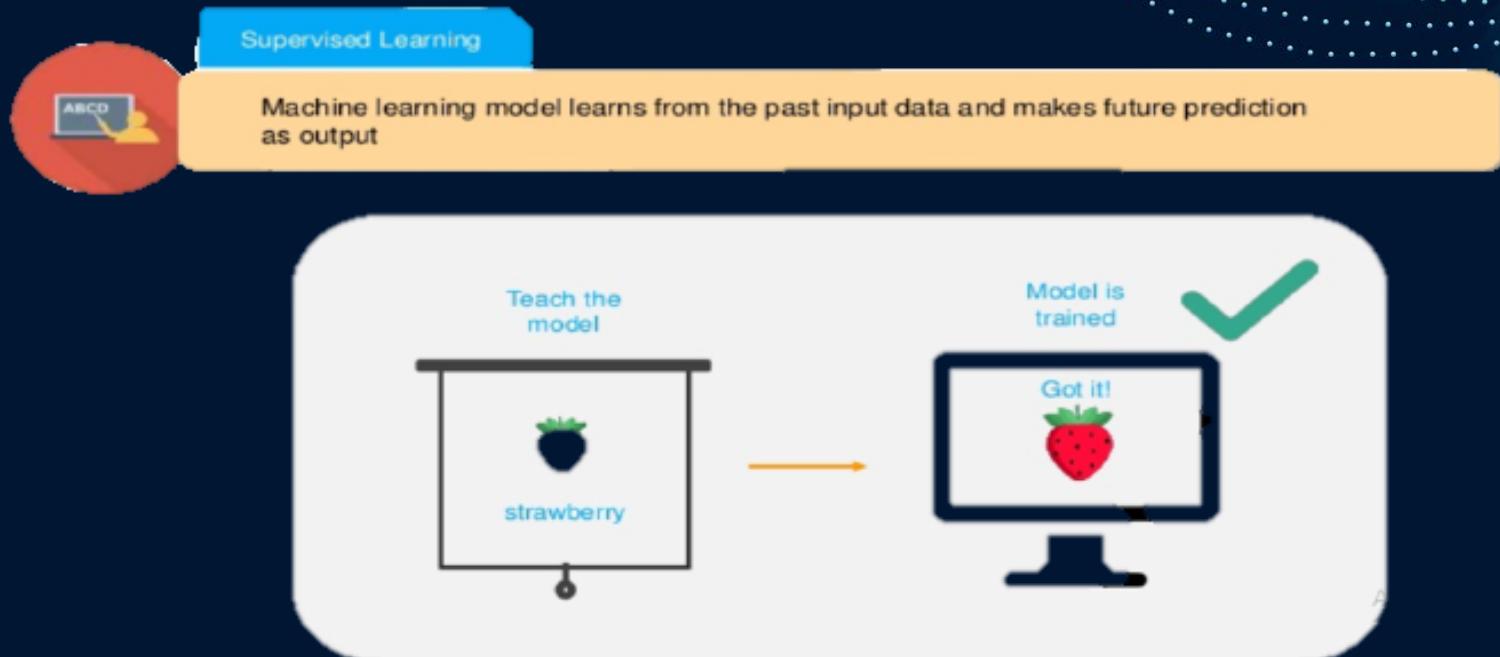
Dr. Gopal Sakarkar  
Department of AI, GHRCE, Nagpur

YouTube Channel on Machine Learning Algorithms  
<https://tinyurl.com/GopalMachineLearningAlgorithms>

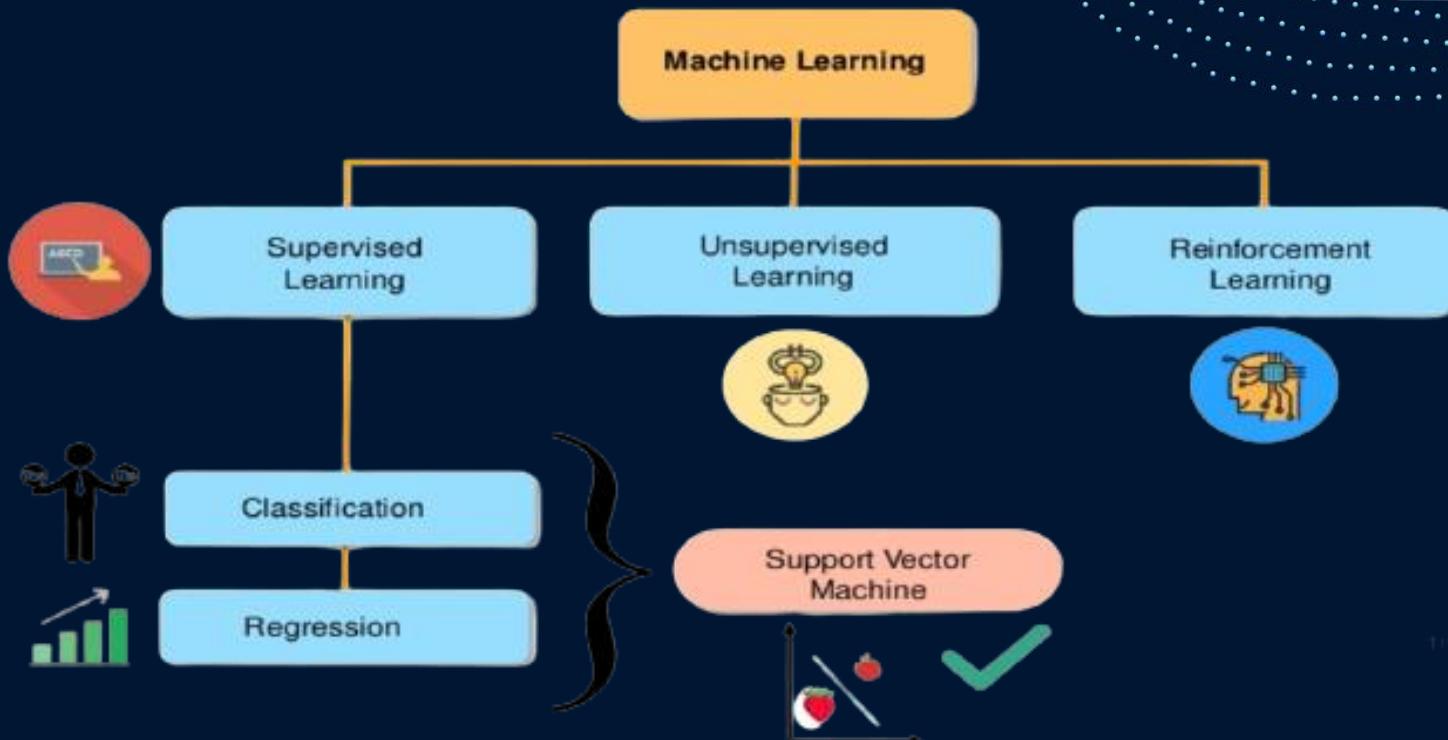
# Support Vector Machine



# Support Vector Machine



# Support Vector Machine



Private W  
Setting

# Support Vector Machine

Last week, my son and I visited a fruit shop



# Support Vector Machine



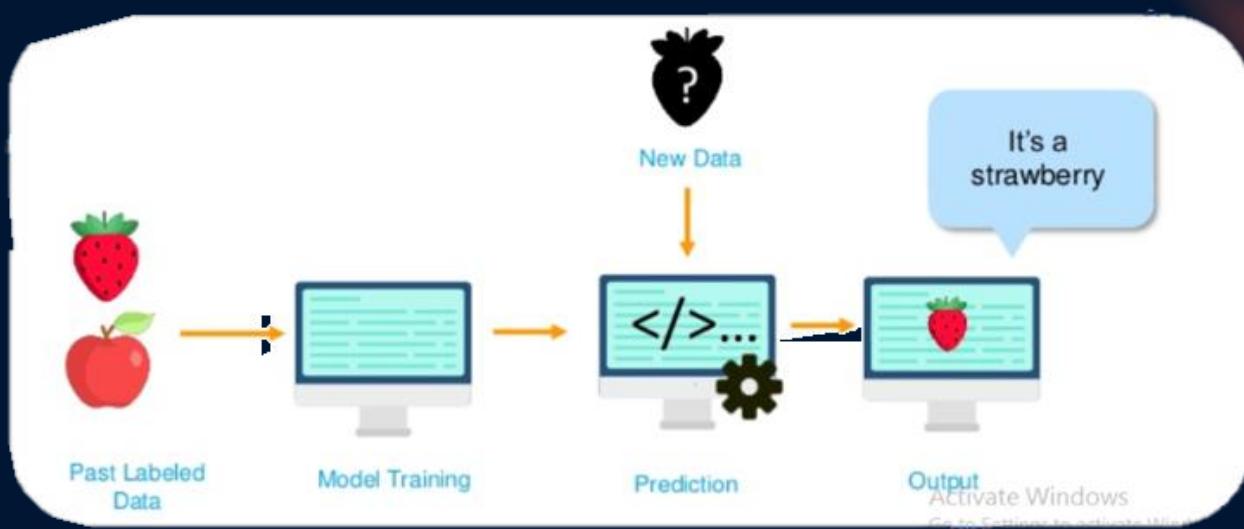
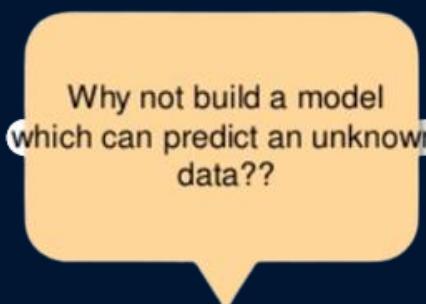
# Support Vector Machine

After a couple of seconds,  
he could figure out that it  
was a strawberry

It is a strawberry!

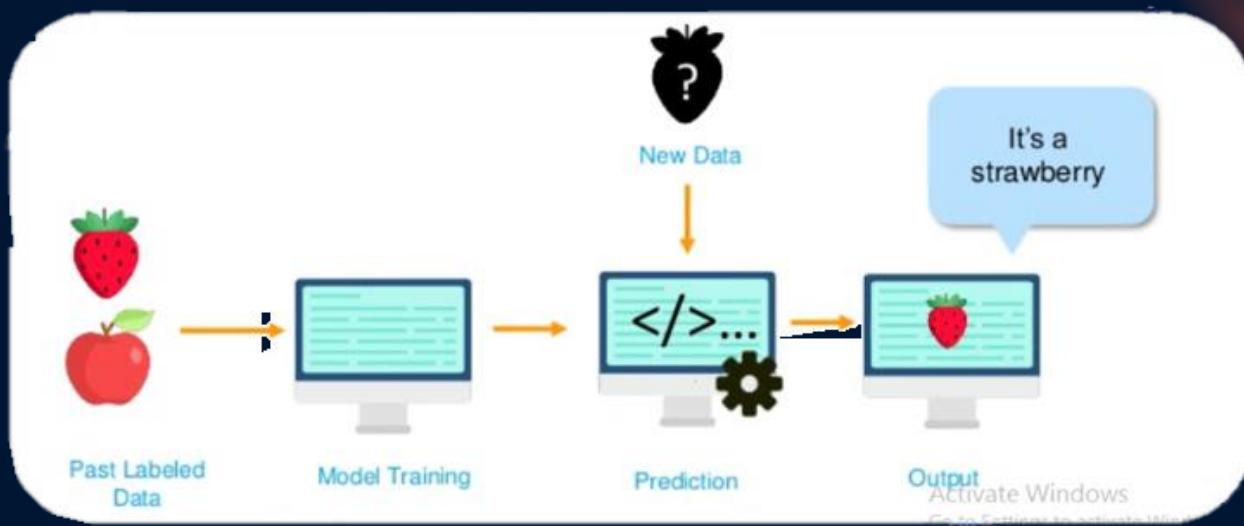


# Support Vector Machine



# Support Vector Machine

This is Support Vector Machine



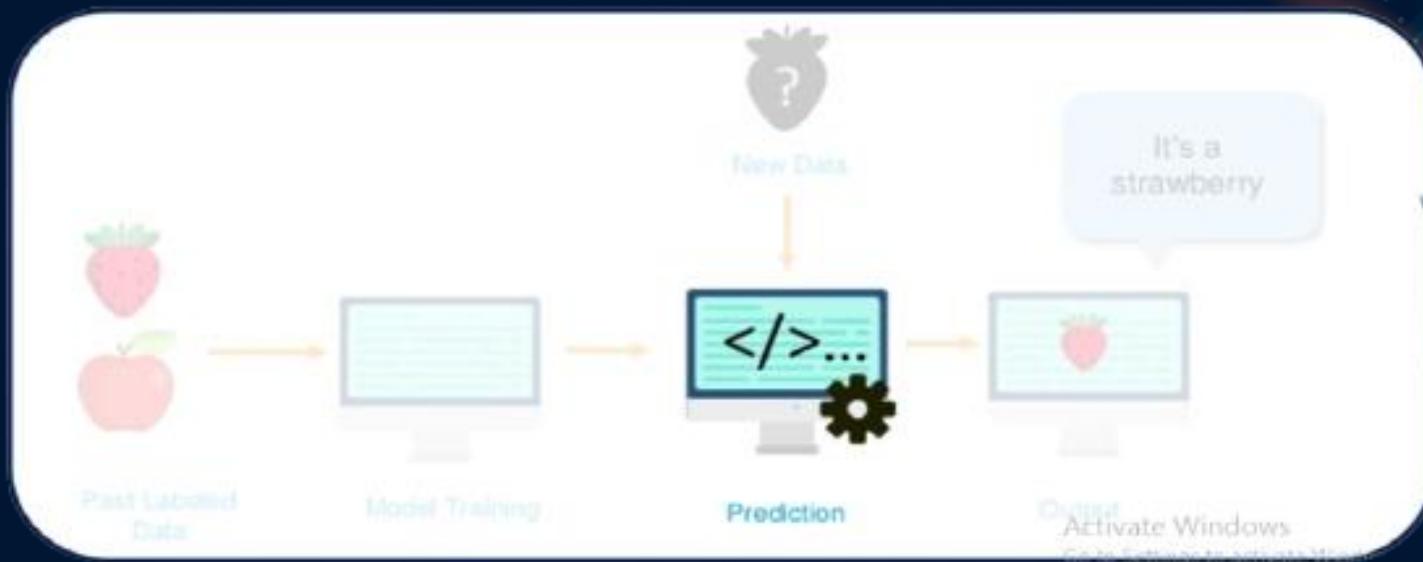
# Support Vector Machine

SVM is a supervised learning method that looks at data and sorts it into one of the two categories

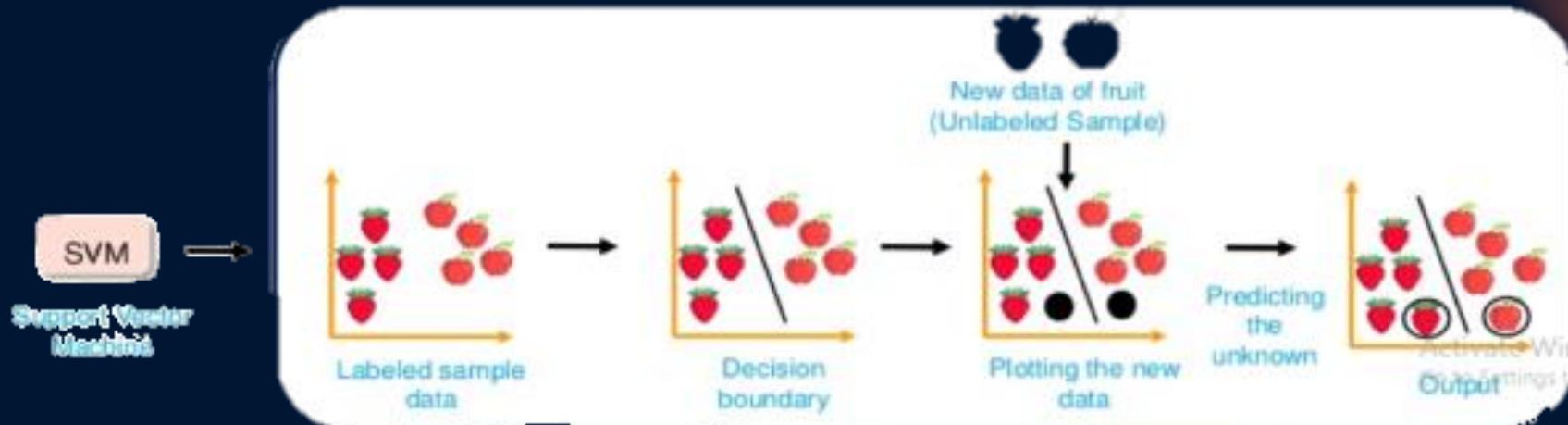


# Support Vector Machine

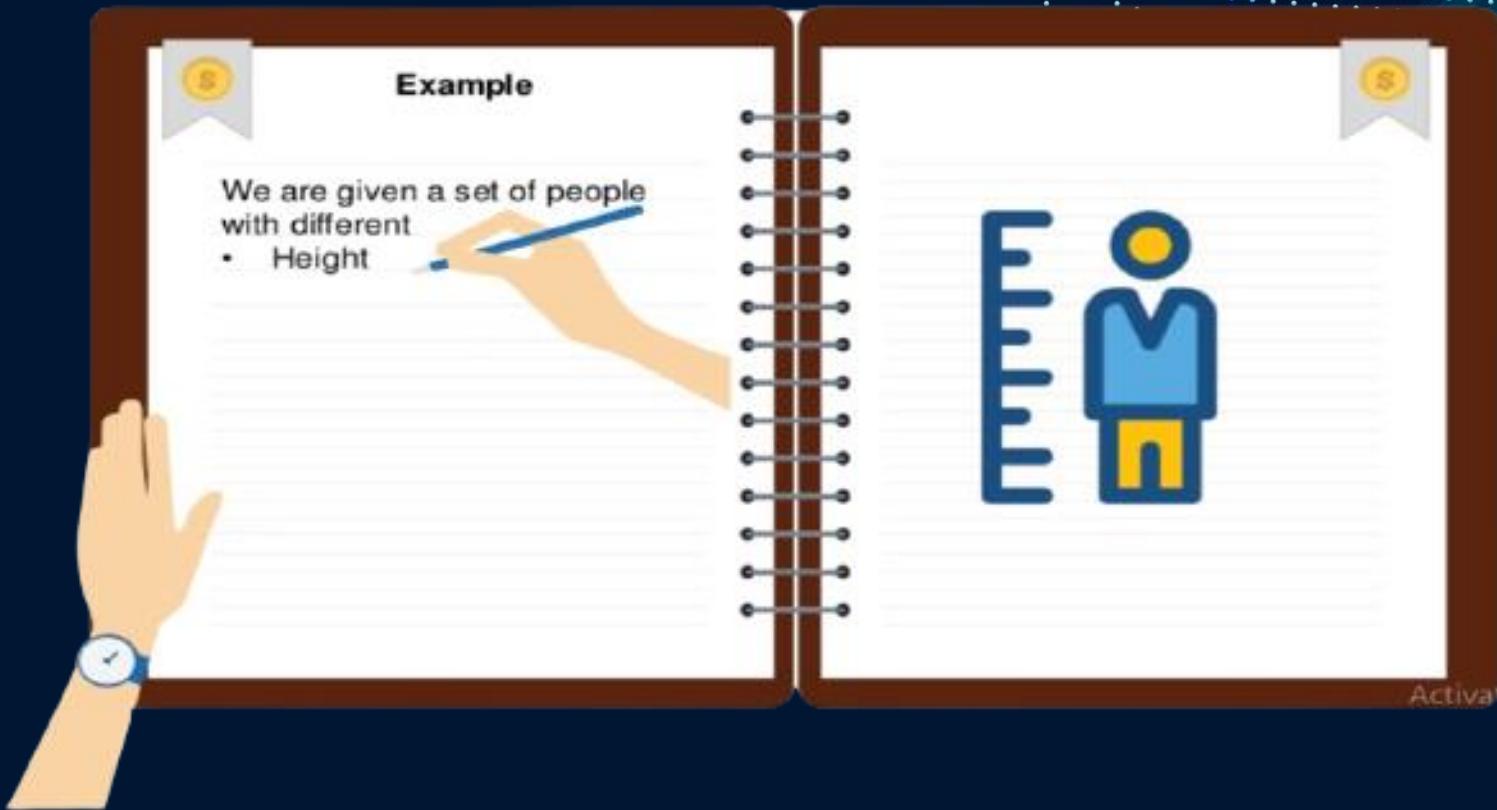
But how does prediction work?



# Support Vector Machine



# Support Vector Machine



# Support Vector Machine

**Example**

We are given a set of people with different

- Height and
- Weight

A hand is shown writing on the page.

Activat

# Support Vector Machine

The image shows an open notebook with two pages. The left page, titled 'Example', contains text and a hand-drawn illustration. The right page, titled 'Sample data set', contains a table of data.

**Example**

We are given a set of people with different

- Height and
- Weight

**Sample data set**

**Female**

| Height | Weight |
|--------|--------|
| 174    | 65     |
| 174    | 88     |
| 175    | 75     |
| 180    | 65     |
| 185    | 80     |

# Support Vector Machine

The image shows an open notebook with a dark brown cover. The left page is titled "Example" and contains the following text:

We are given a set of people with different

- Height and
- Weight

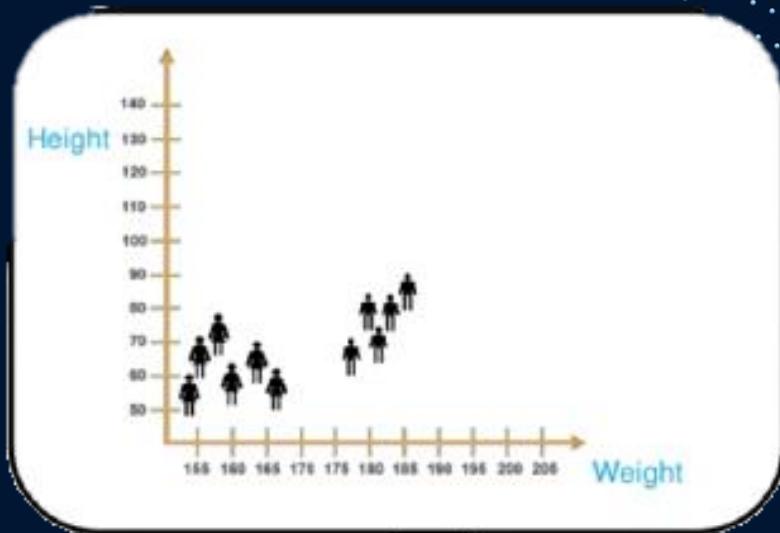
A hand is shown writing on the page with a blue pen. The right page is titled "Sample data set" and contains the following table:

**Male**

| Height | Weight |
|--------|--------|
| 179    | 90     |
| 180    | 80     |
| 183    | 80     |
| 187    | 85     |
| 182    | 72     |

At the bottom right of the right page, the word "Activai" is partially visible.

# Support Vector Machine



# Support Vector Machine

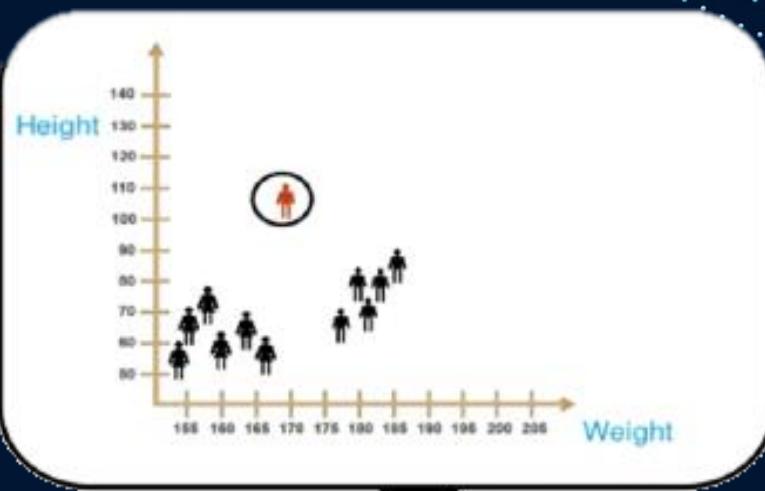


Let's add a new data point and figure out if it's a male or a female?



# Support Vector Machine

Sure.. For this task, we need to split our data first



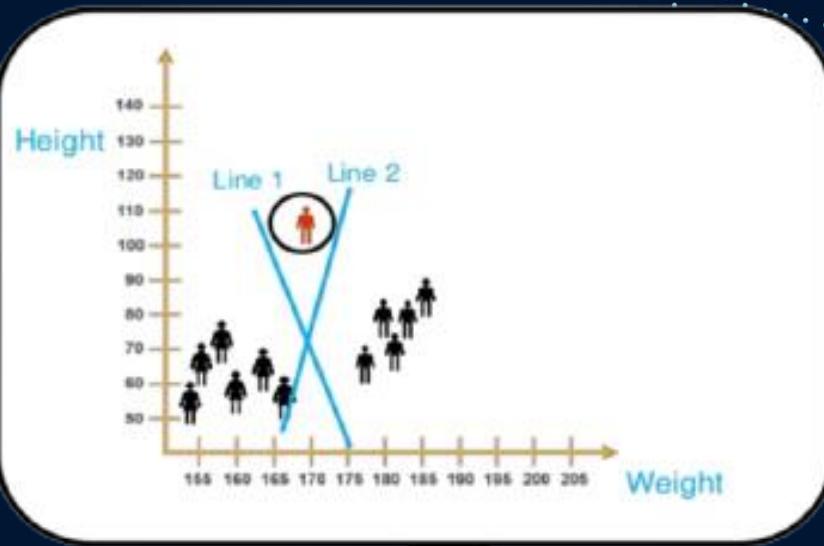
# Support Vector Machine

We can split our data by choosing any of these lines



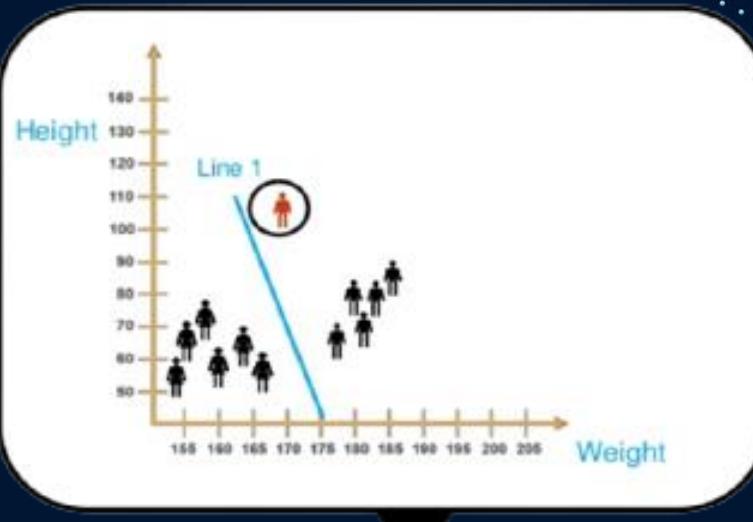
# Support Vector Machine

But to predict the gender of a new data point we should split the data in the best possible way



# Support Vector Machine

Then I would say, this line best splits the data



Why do you say  
it's the best  
split??



# Support Vector Machine

This line has the maximum space that separates the two classes



Why do you say  
it's the best  
split??



# Support Vector Machine

While the other line doesn't have the maximum space that separates the two classes

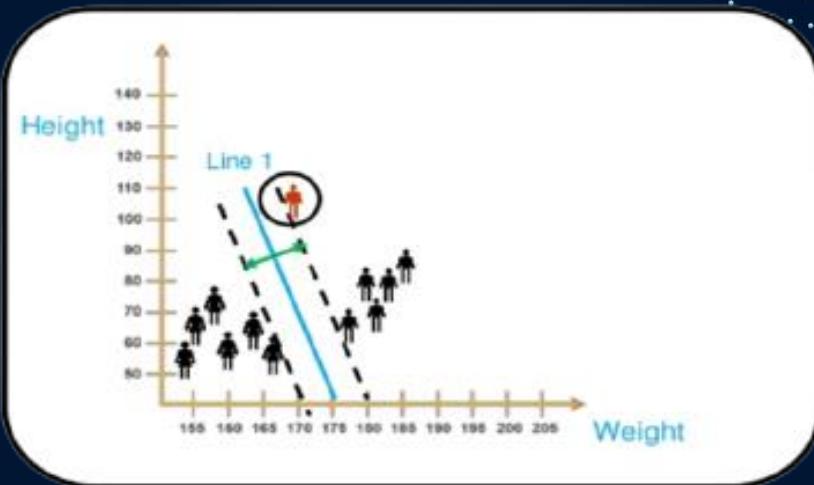


Why do you say it's the best split??



# Support Vector Machine

That is why this line best splits the data

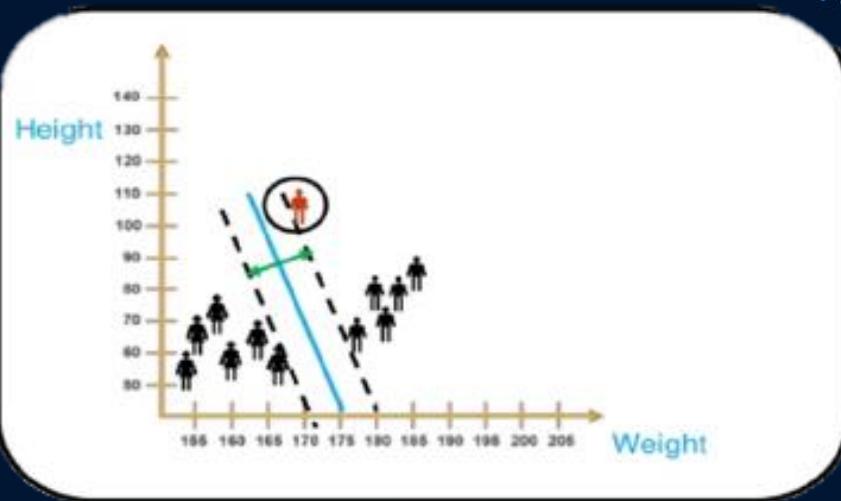


Well yes.. This is the best split!



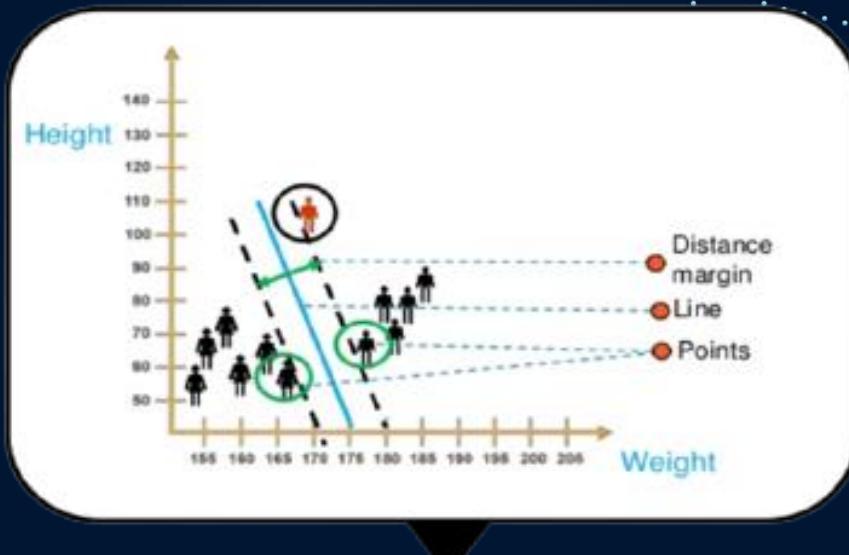
# Support Vector Machine

Now, Let me add some technical terms to this



# Support Vector Machine

We can also say that the distance between the points and the line should be far as possible



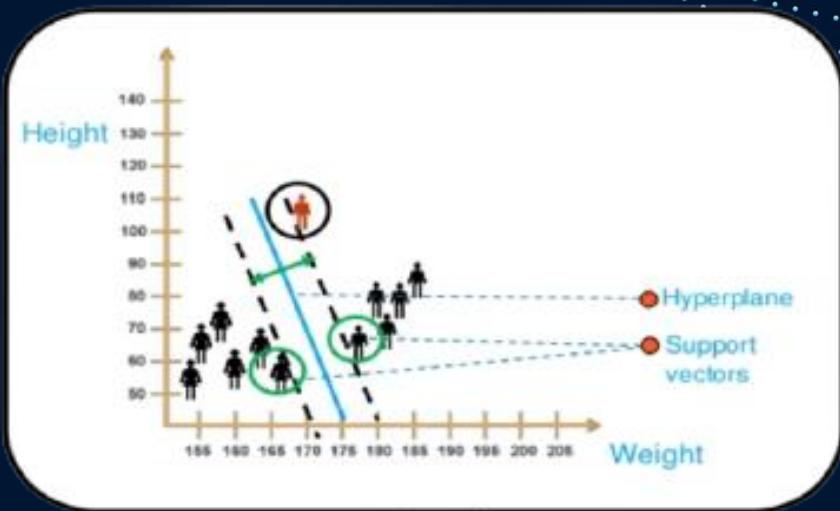
# Support Vector Machine

## Margin:-

1. Margin is the perpendicular distance between the closest data points and the Hyperplane ( on both sides )
2. The best optimised line ( hyperplane ) with maximum margin is termed as Margin Maximal Hyperplane.
3. The closest points where the margin distance is calculated are considered as the support vectors.
4. Here the hyperplane is a 2d plane drawn parallel to x-axis that is the separator.

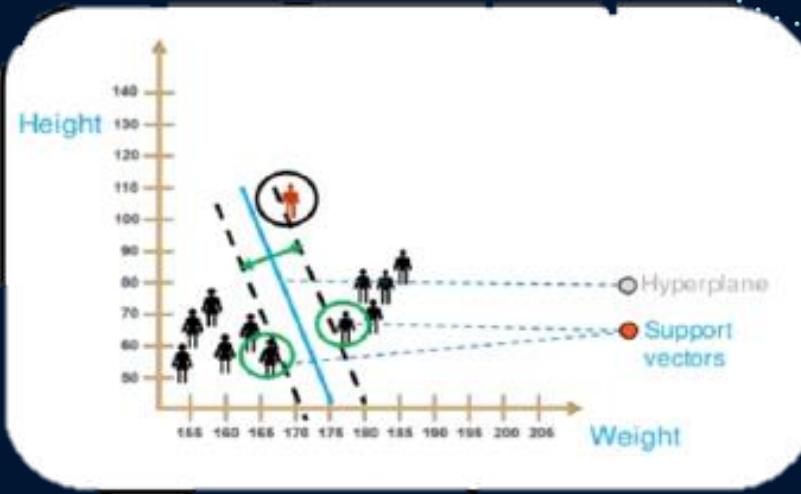
# Support Vector Machine

In technical terms we can say,  
the distance between the  
support vector and the  
hyperplane should be far as  
possible



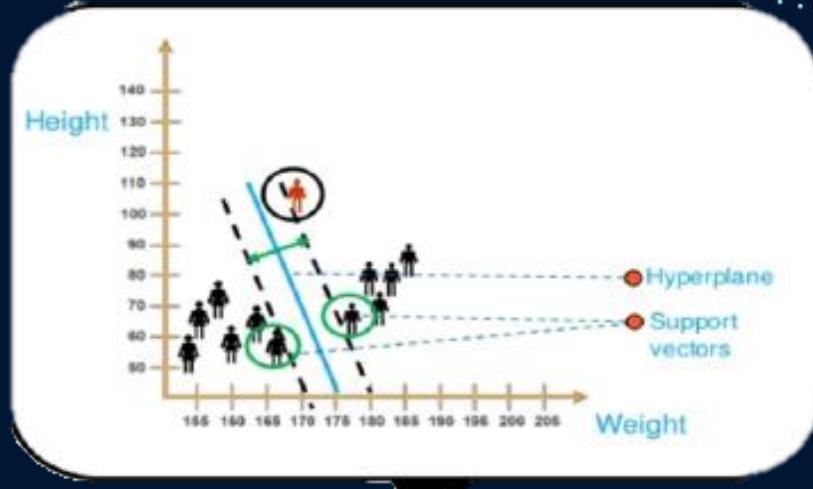
# Support Vector Machine

Where support vectors are the extreme points in the datasets



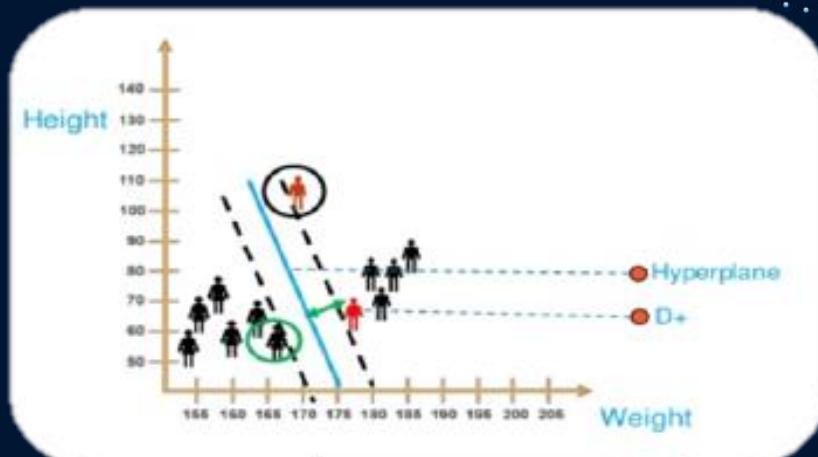
# Support Vector Machine

And a hyperplane has the maximum distance to the support vectors of any class



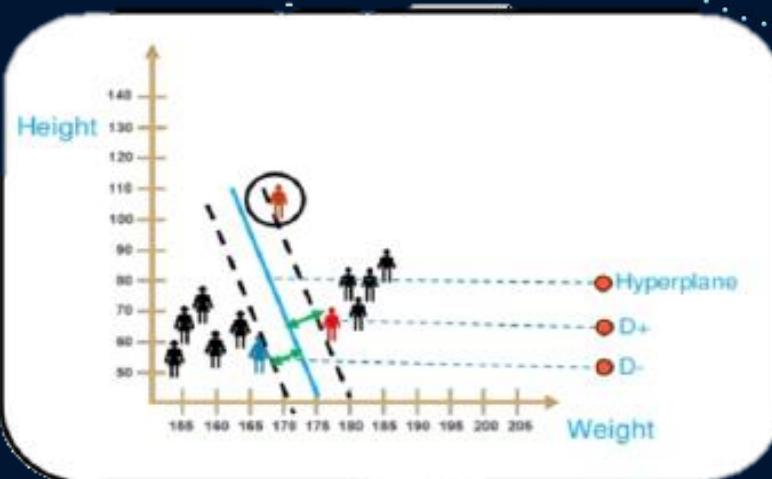
# Support Vector Machine

Here,  $D_+$  is the shortest distance to the closest positive point.



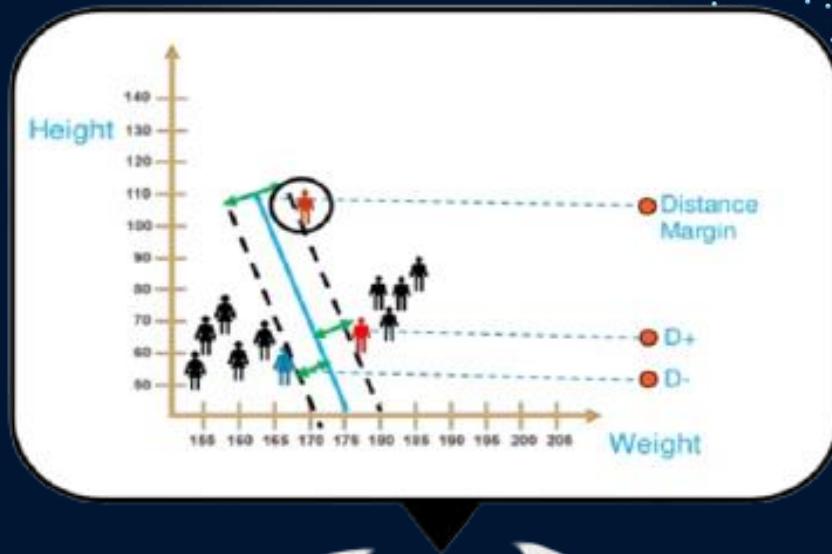
# Support Vector Machine

And  $D_-$  is the shortest distance to the closest negative point



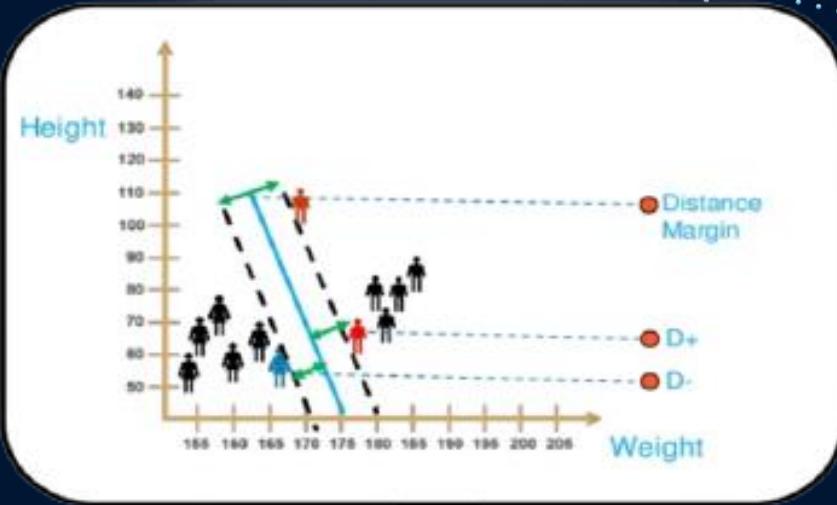
# Support Vector Machine

Sum of D+ and D- is called the distance margin



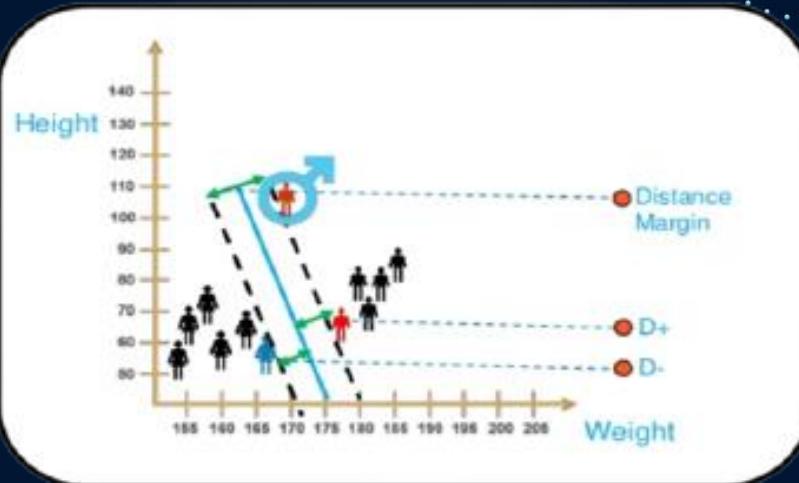
# Support Vector Machine

From the distance margin, we get an optimal hyperplane



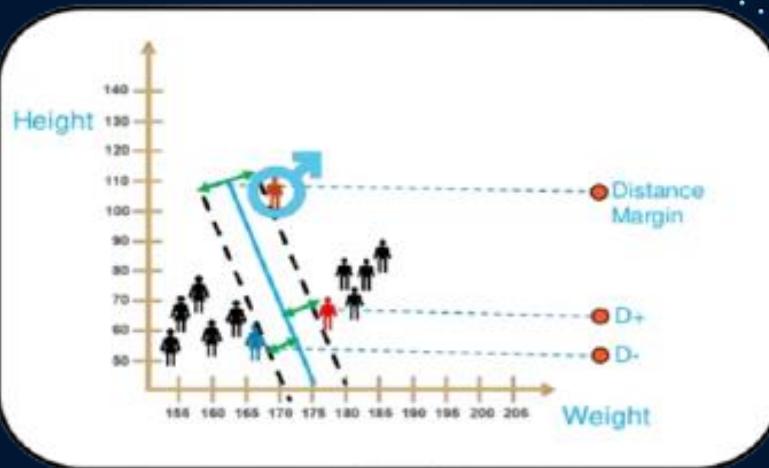
# Support Vector Machine

Based on the hyperplane,  
we can say the new data point  
belongs to male gender



# Support Vector Machine

Based on the distance margin,  
we can say the new data point  
belongs to male gender

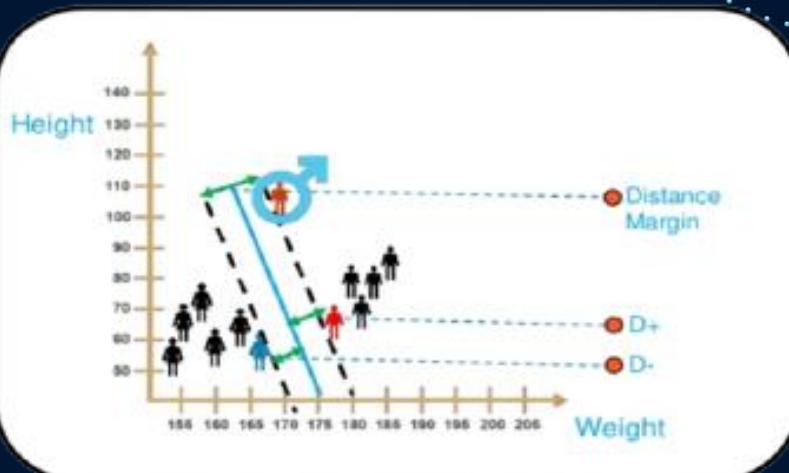


That was so clear!



# What is Support Vector Machine

Based on the distance margin,  
we can say the new data point  
belongs to male gender



That was so clear!



# What is Support Vector Machine

If we select a hyperplane having low margin then there is high chance of misclassification

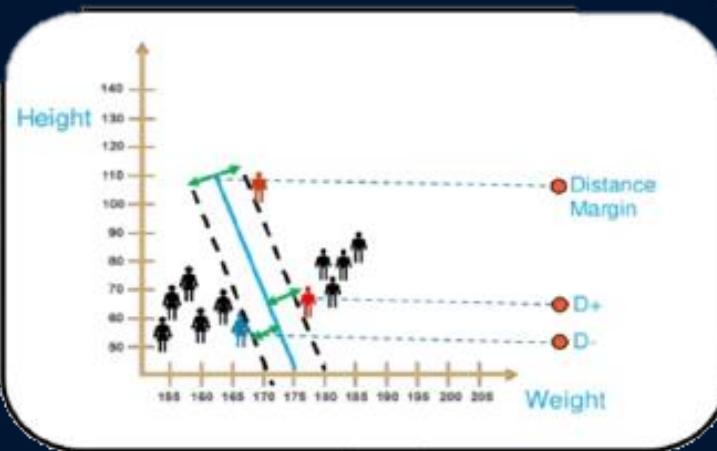


But what happens if a hyperplane is not optimal?



# What is Support Vector Machine

What we discussed so far, is also called as LSVM



But what happens if a hyperplane is not optimal?



# Understanding Support Vector Machine



# Understanding Support Vector Machine

But like this?



Sample Dataset



# Understanding Support Vector Machine

Here, we cannot use a hyperplane



Transformation



# Understanding Support Vector Machine

So, it's necessary to move away from a 1-D view of the data to a 2-D view



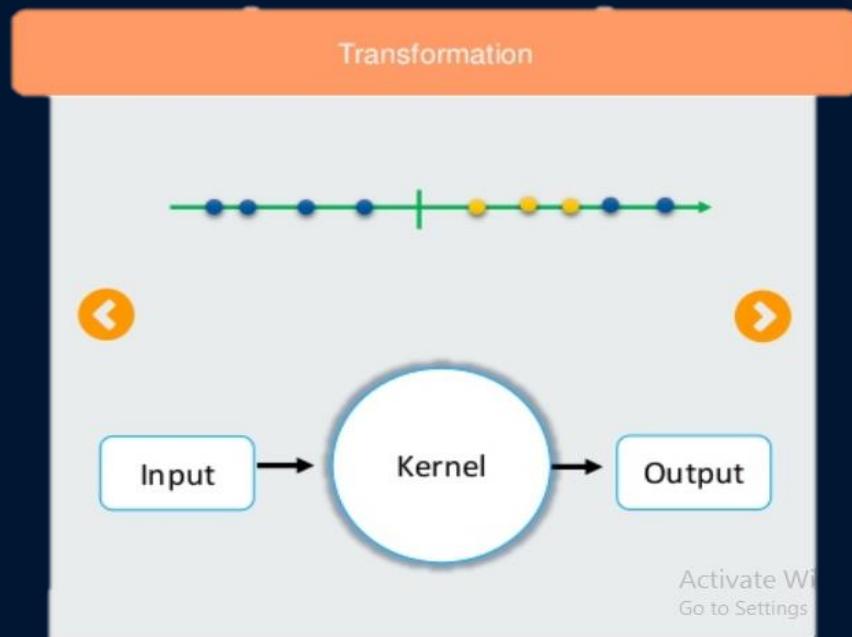
Transformation



Activate W  
Go to Settings

# Understanding Support Vector Machine

For the transformation, we use a Kernel Function



---

# Kernel Function

- Mathematical functions for transforming data
- using some linear algebra
- Different SVM algorithms use different types of kernel functions



---

## Kernel Function: Example

$$K(x, y) = \langle f(x), f(y) \rangle$$

Kernel function dot product of n- dimensional inputs



# Understanding Support Vector Machine

## Mathematical representation

$$x = (x_1, x_2, x_3); y = (y_1, y_2, y_3)$$

$$f(x) = (x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3)$$

$$f(y) = (y_1y_1, y_1y_2, y_1y_3, y_2y_1, y_2y_2, y_2y_3, y_3y_1, y_3y_2, y_3y_3)$$

$$K(x, y) = \langle x, y \rangle^2$$

$$x = (1, 2, 3)$$

$$y = (4, 5, 6)$$

$$f(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)$$

$$f(y) = (16, 20, 24, 20, 25, 30, 24, 30, 36)$$

$$\langle f(x), f(y) \rangle = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$$

$$K(x, y) = (4 + 10 + 18)^2 = 1024 \longrightarrow \text{Kernel function}$$

---

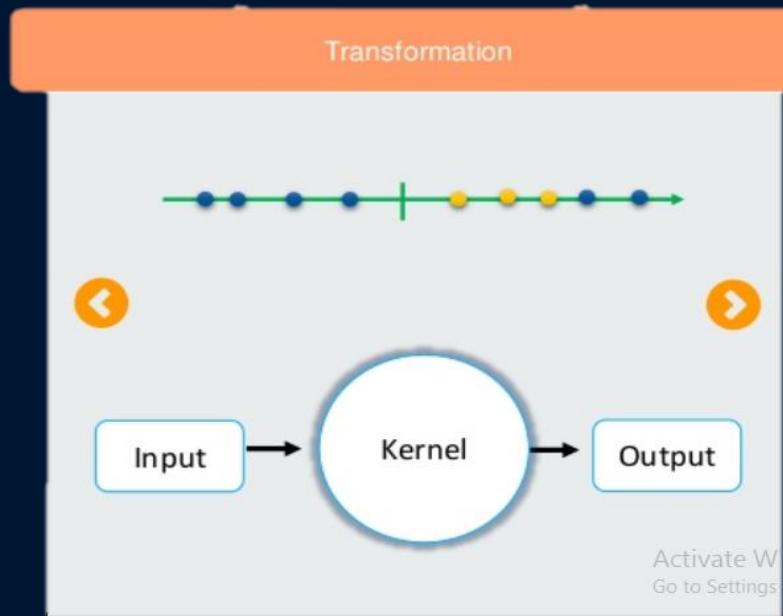
## Various kernels available

1. Linear kernel
2. Non - linear kernel
3. Radial basis function ( RBF )
4. Sigmoid
5. Polynomial
6. Exponential



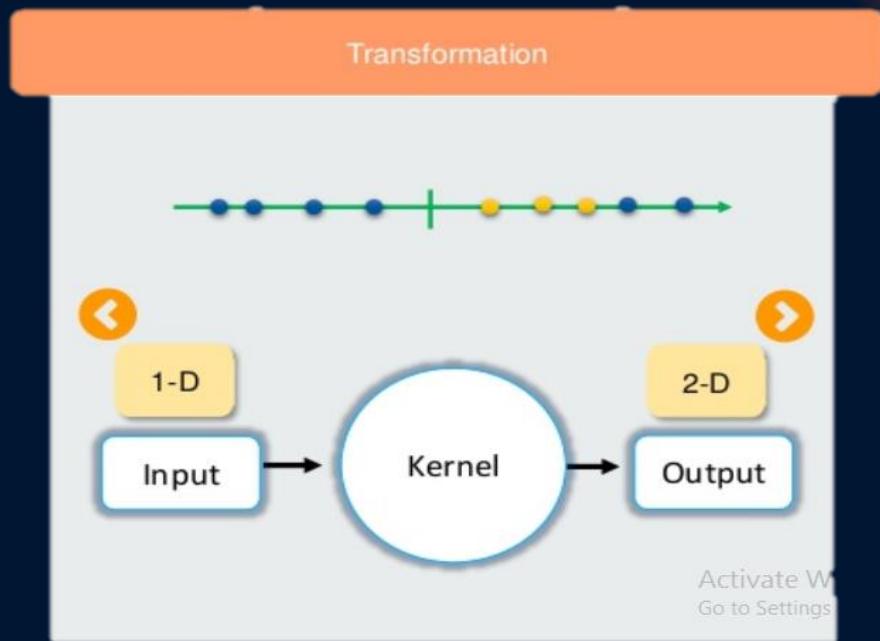
# Understanding Support Vector Machine

For the transformation, we use a Kernel Function



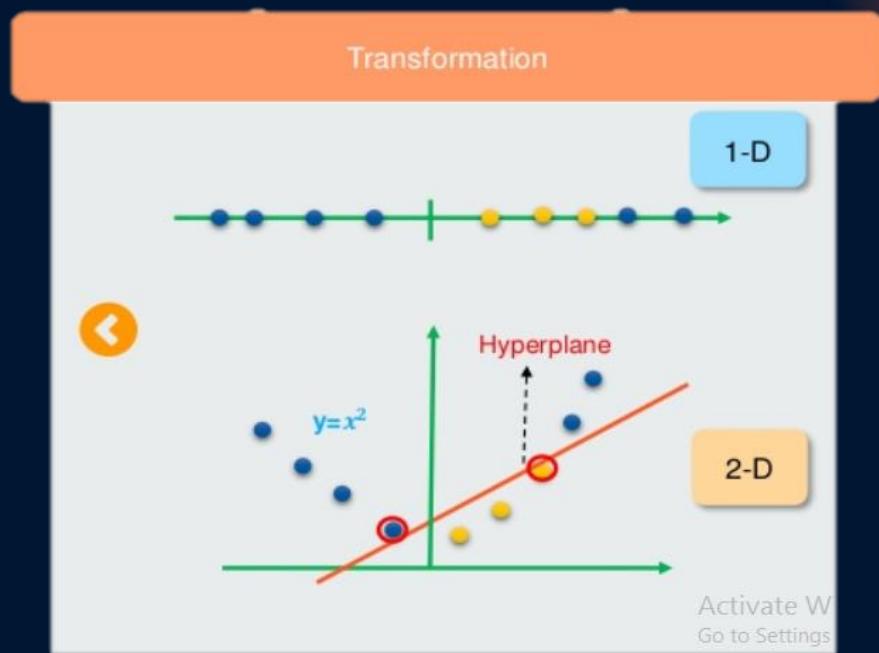
# Understanding Support Vector Machine

Which will take the 1-D input and transfer it to 2-D Output



# Understanding Support Vector Machine

Now, we got the result !!

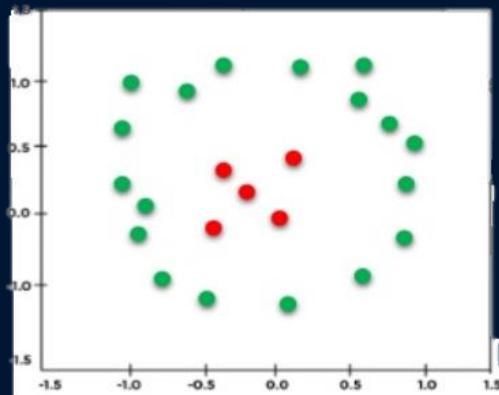


# Understanding Support Vector Machine

How to perform SVM  
for this type of dataset?



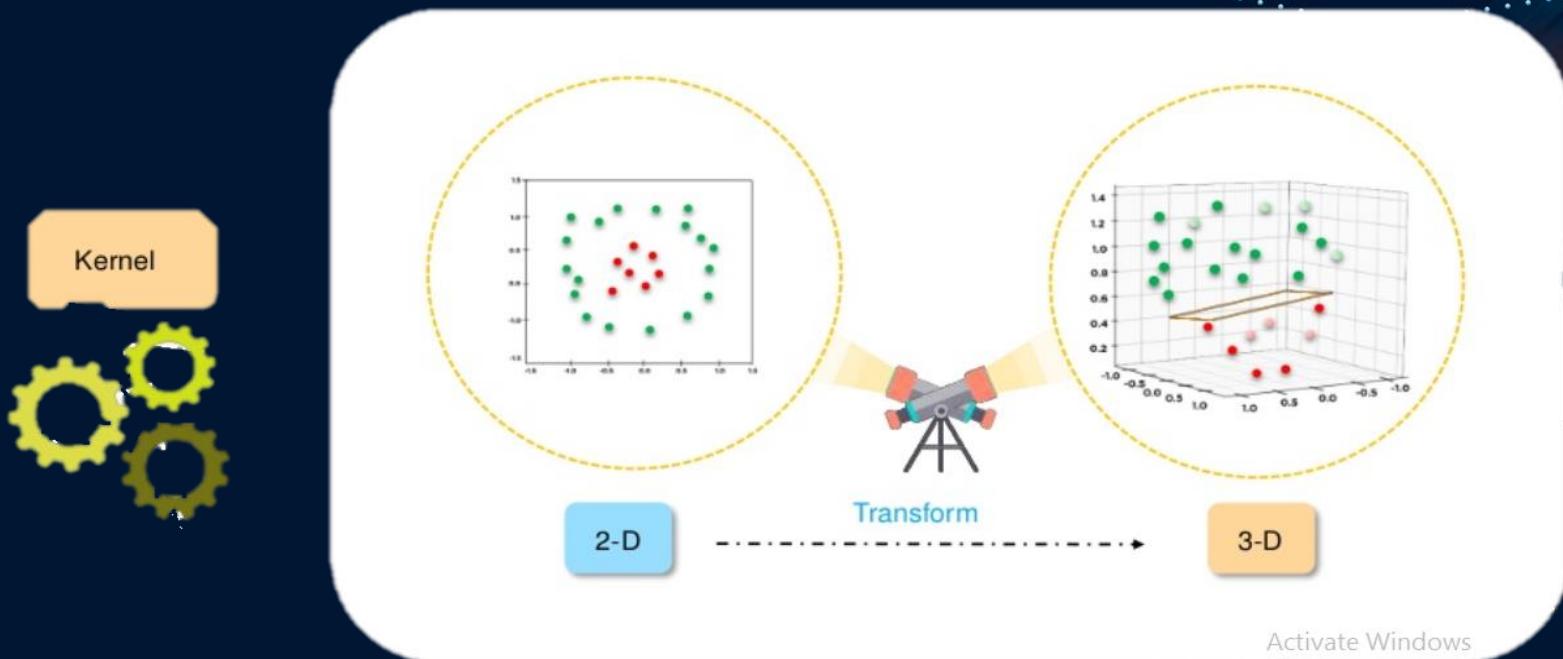
Sample Dataset



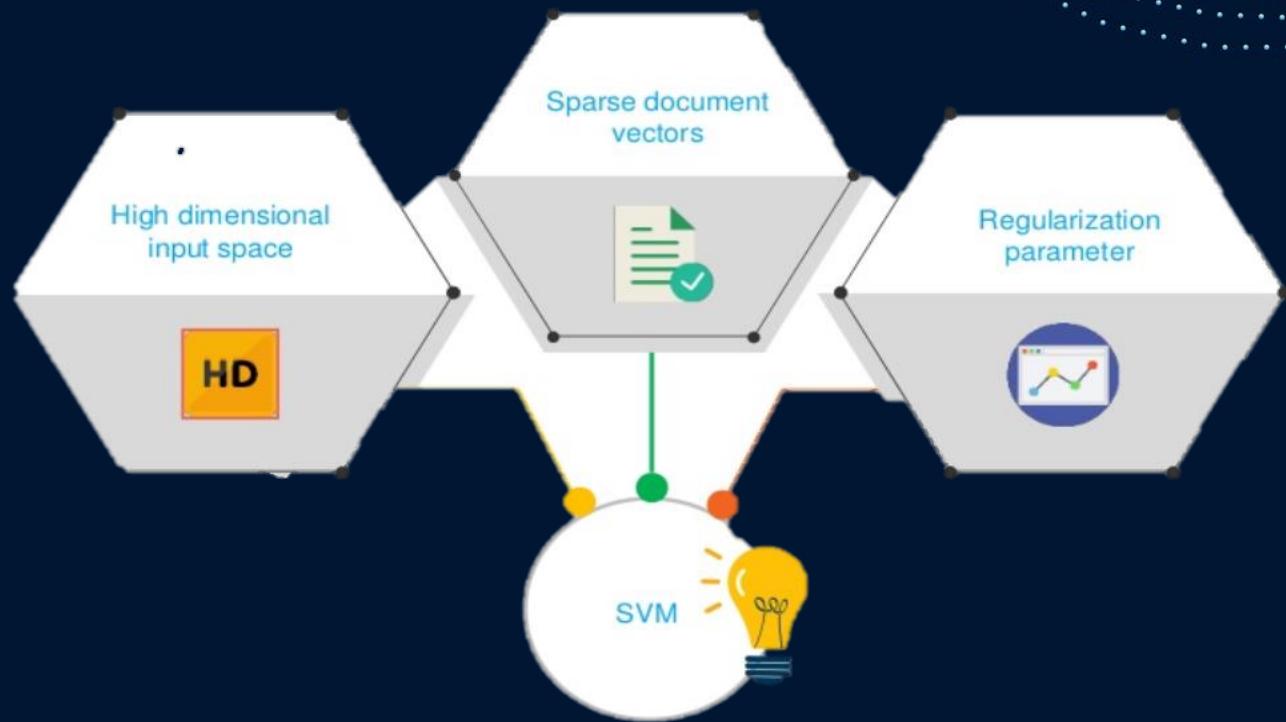
# Understanding Support Vector Machine



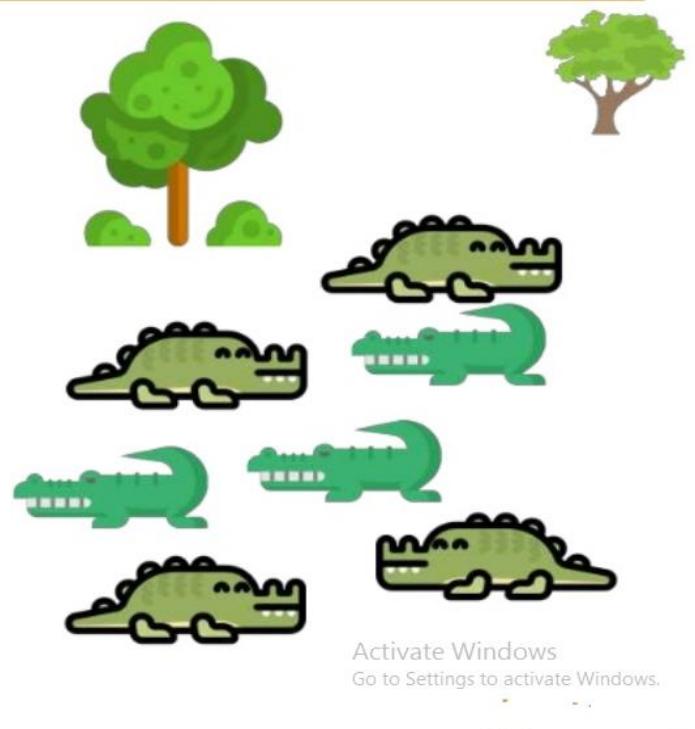
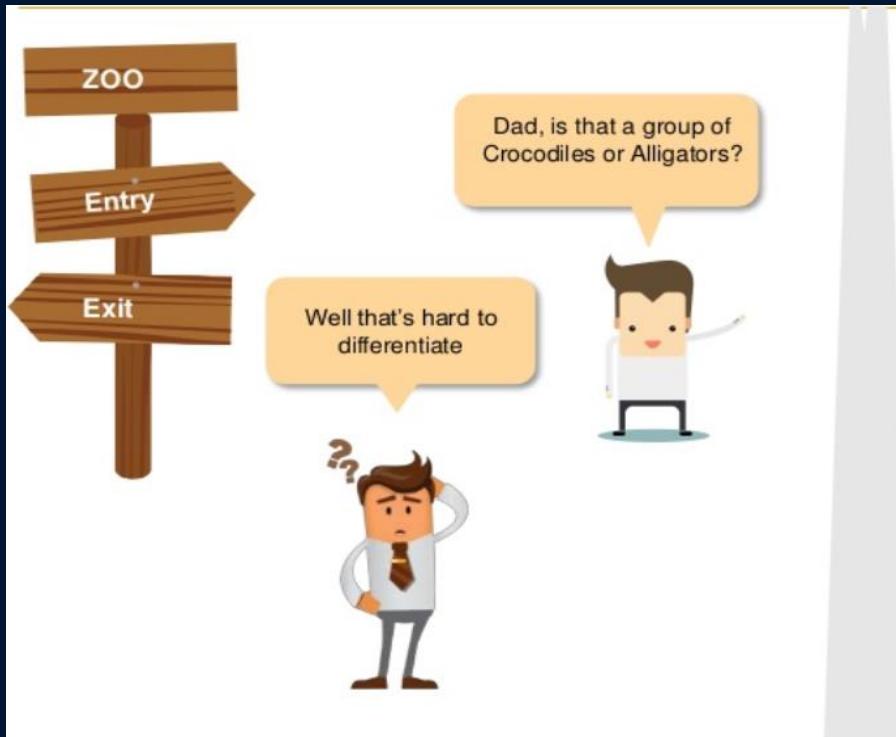
# Understanding Support Vector Machine



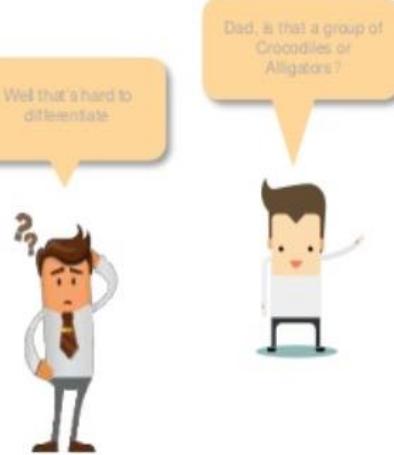
# Advantages of Support Vector Machine



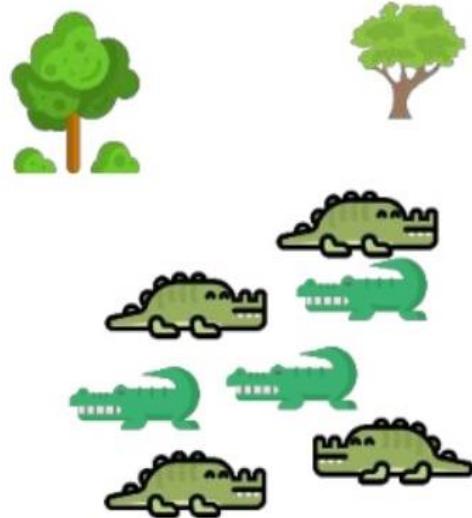
# Use Case – Problem Statement



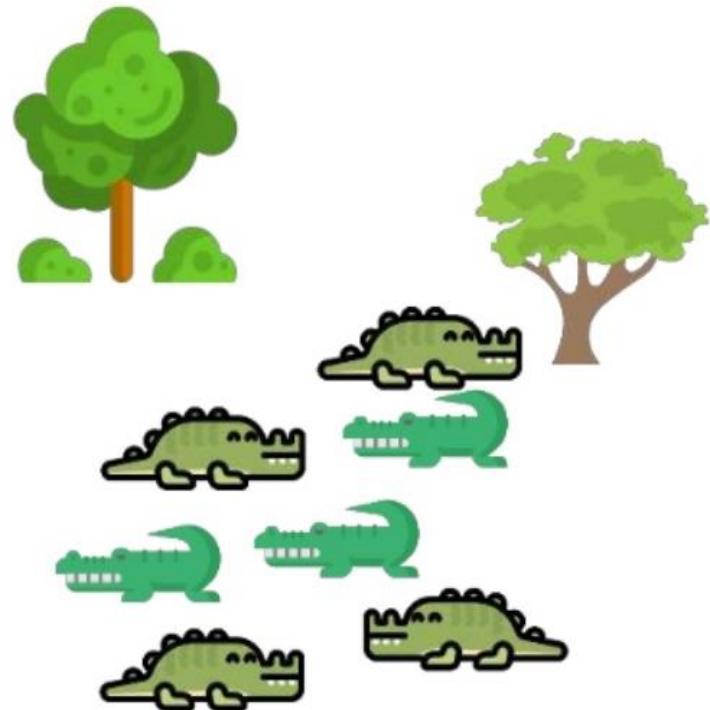
# Use Case – Problem Statement



| Difference  |   |  |  |
|-------------|---|--|--|
|             |   |  |  |
| Size        | <ul style="list-style-type: none"><li>Crocodiles are larger in size</li></ul> | <ul style="list-style-type: none"><li>Alligators are smaller in size</li></ul> |  |
| Snout Width | <ul style="list-style-type: none"><li>Crocodiles have narrow snout</li></ul>  | <ul style="list-style-type: none"><li>Alligators have wider snout</li></ul>    |  |



# Use Case – Problem Statement



---

## Support Vector Machine: Part 2

- The Support Vector Machine is a supervised learning algorithm mostly used for classification but it can be used also for regression.
- The main idea is that based on the labeled data (training data) the algorithm tries to find the optimal hyperplane which can be used to classify new data points.
- In two dimensions the hyperplane is a simple line.

---

# Support Vector Machine: Part 2

- Usually a learning algorithm tries to learn the most common characteristics (what differentiates one class from another) of a class and the classification is based on those representative characteristics learnt (so classification is based on differences between classes).
- The SVM works in the other way around. It finds the most similar examples between classes. Those will be the support vectors.

---

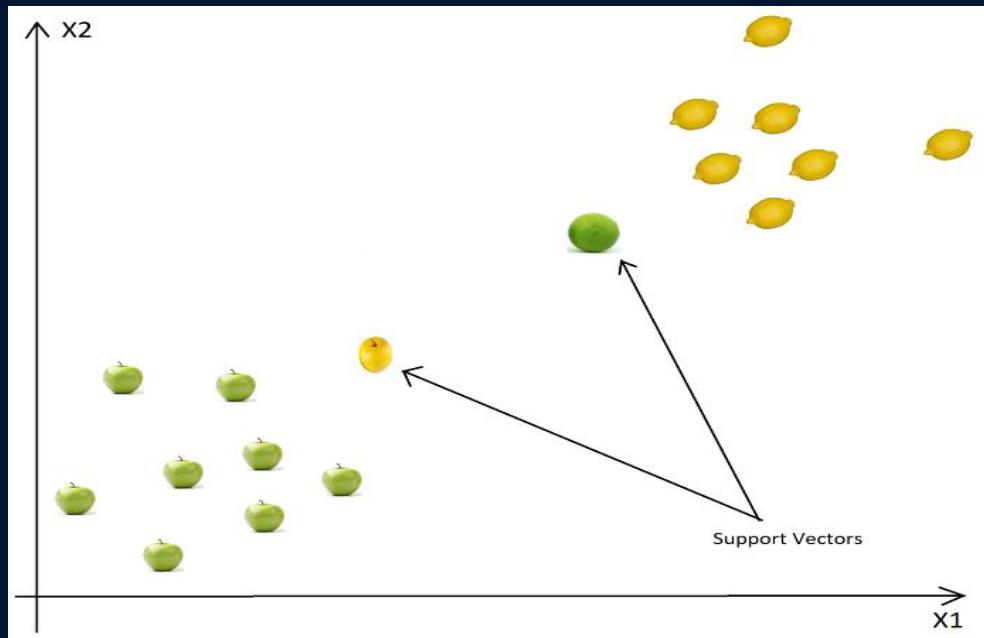
# Support Vector Machine: Part 2

## An example

- lets consider two classes, apples and lemons.
- Other algorithms will learn the most evident, most representative characteristics of apples and lemons, like apples are green and rounded while lemons are yellow and have elliptic form.
- In contrast, SVM will search for apples that are very similar to lemons, for example apples which are yellow and have elliptic form. This will be a support vector.
- The other support vector will be a lemon similar to an apple (green and rounded).
- So other algorithms learns the differences while SVM learns similarities.

# Support Vector Machine: Part 2

Visualize the example above in 2D, we will have something like this:



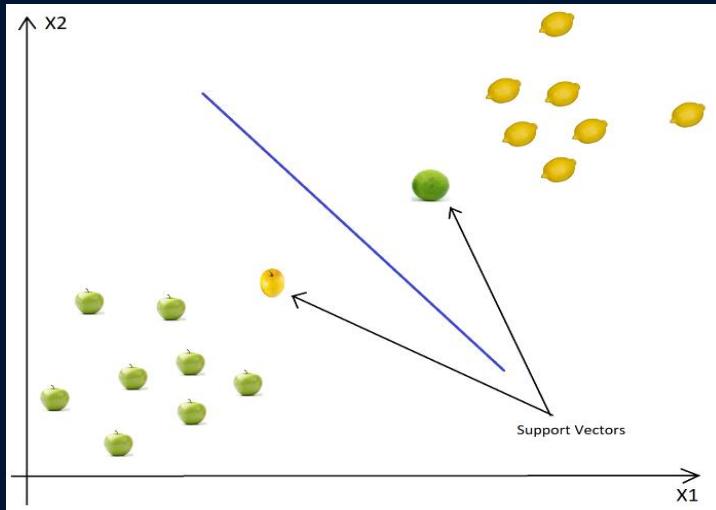
---

## Support Vector Machine: Part 2

- As we go from left to right, all the examples will be classified as apples until we reach the yellow apple.
- From this point, the confidence that a new example is an apple drops while the lemon class confidence increases.
- When the lemon class confidence becomes greater than the apple class confidence, the new examples will be classified as lemons (somewhere between the yellow apple and the green lemon).

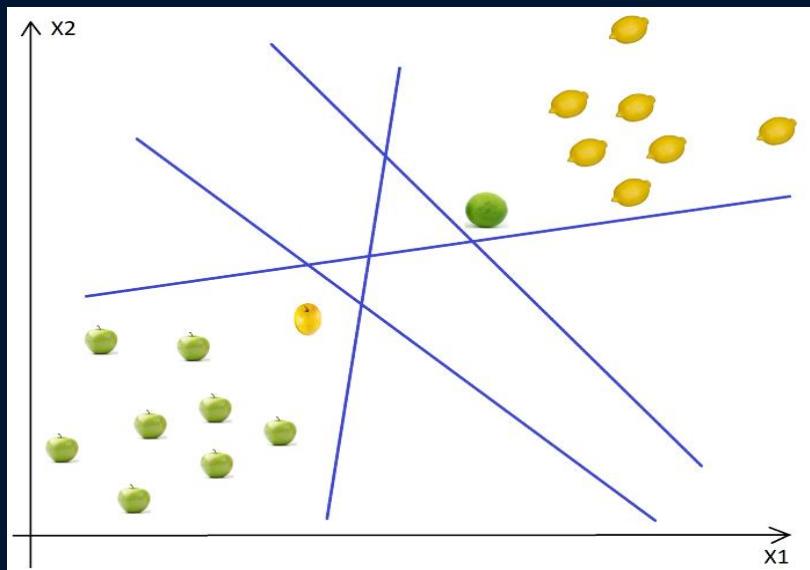
# Support Vector Machine: Part 2

- Based on these support vectors, the algorithm tries to find the best hyperplane that separates the classes.
- In 2D the hyperplane is a line, so it would look like this:



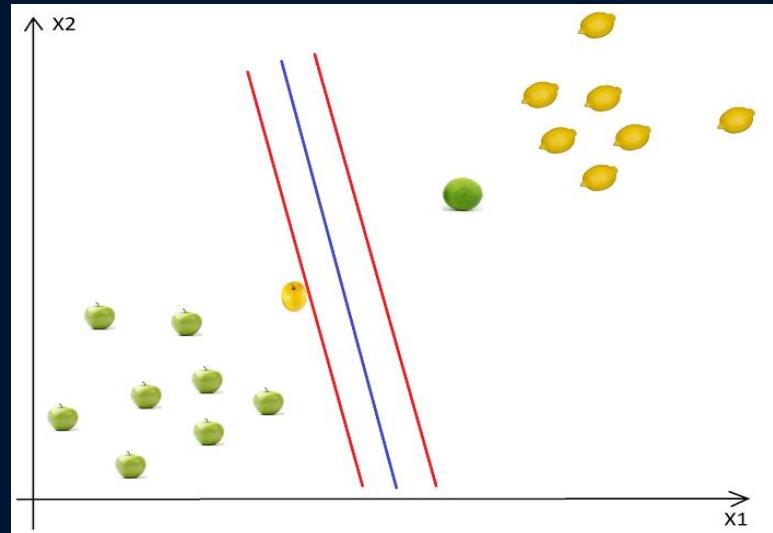
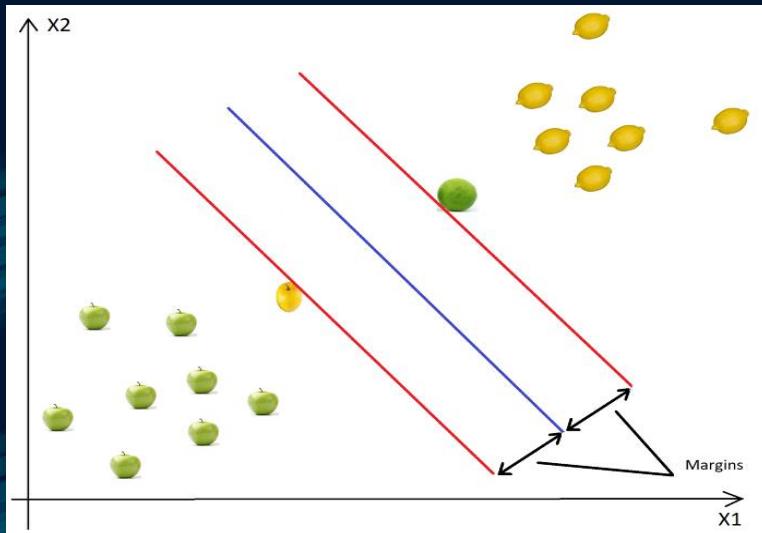
## Support Vector Machine: Part 2

- why did I draw the blue boundary like in the picture above? I could also draw boundaries like this:



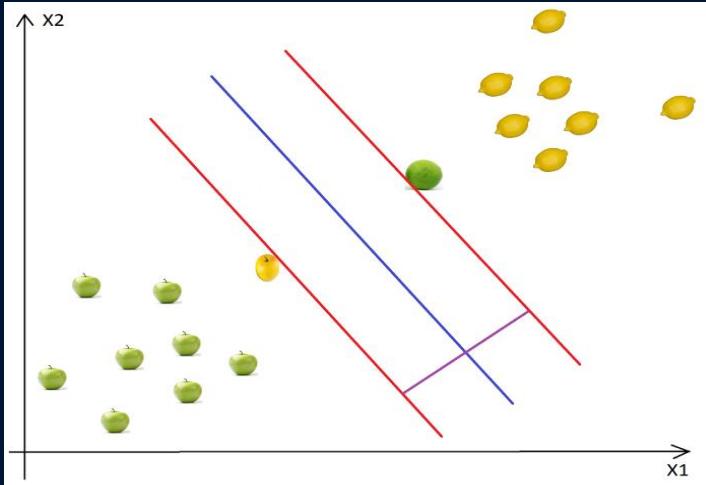
## Support Vector Machine: Part 2

- if we compare the picture Left with the picture Right , we can easily observe, that the first is the optimal hyperplane (line) and the second is a sub-optimal solution, because the margin is far shorter.



## Support Vector Machine: Part 2

- Because we want to maximize the margins taking in consideration all the classes, instead of using one margin for each class.
- we use a “global” margin, which takes in consideration all the classes. This margin would look like the purple line in the following picture:



---

## Support Vector Machine: Part 2

- support vectors are data points that defines the position and the margin of the hyperplane.
- We call them “support” vectors, because these are the representative data points of the classes, if we move one of them, the position and/or the margin will change.
- Moving other data points won’t have effect over the margin or the position of the hyperplane.

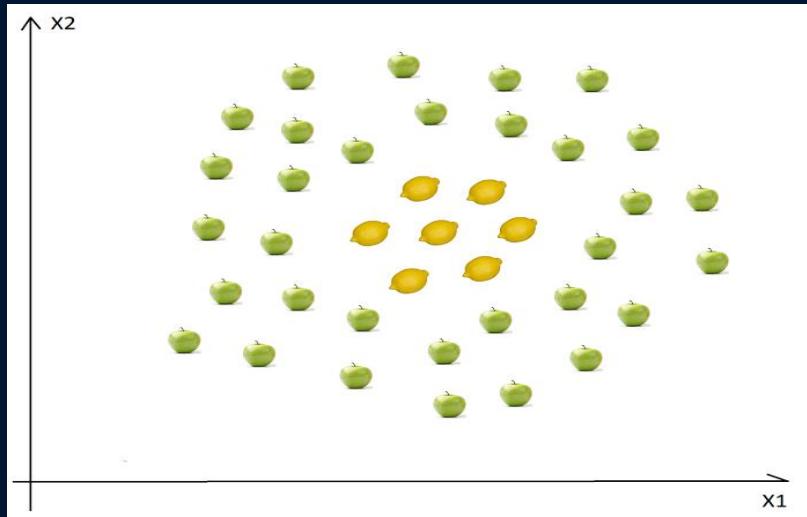
---

# Support Vector Machine: Part 2

- The basic steps of the SVM are:
- select two hyperplanes (in 2D) which separates the data with no points between them (red lines)
- maximize their distance (the margin)
- the average line (here the line half way between the two red lines) will be the decision boundary

# Support Vector Machine: Part 3

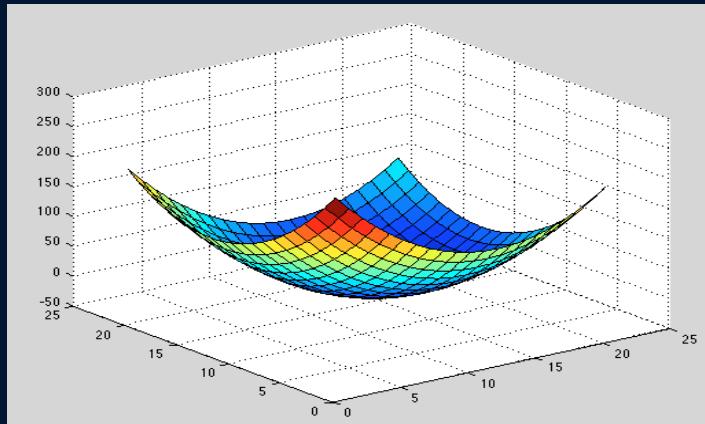
## SVM for Non-Linear Data Sets



**In this case we cannot find a straight line to separate apples from lemons. So how can we solve this problem. We will use the Kernel Trick!**

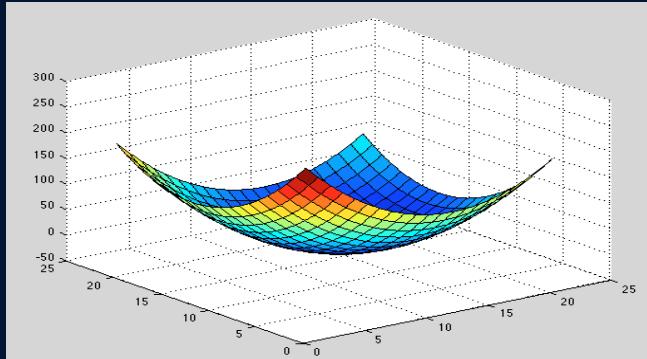
# Support Vector Machine: Part 3

- Mapping from 2D to 3D
- let's assume that we add another dimension called X3. Another important transformation is that in the new dimension the points are organized using this formula  $x_1^2 + x_2^2$ .
- If we plot the plane defined by the  $x^2 + y^2$  formula, we will get something like this:



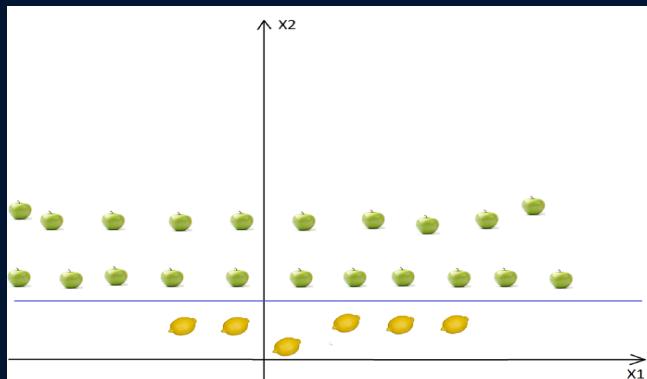
# Support Vector Machine: Part 3

- Now we have to map the apples and lemons (which are just simple points) to this new space. Think about it carefully, what did we do? We just used a transformation in which we added levels based on distance.
- If you are in the origin, then the points will be on the lowest level. As we move away from the origin, it means that we are climbing the hill (moving from the center of the plane towards the margins) so the level of the points will be higher.



## Support Vector Machine: Part 3

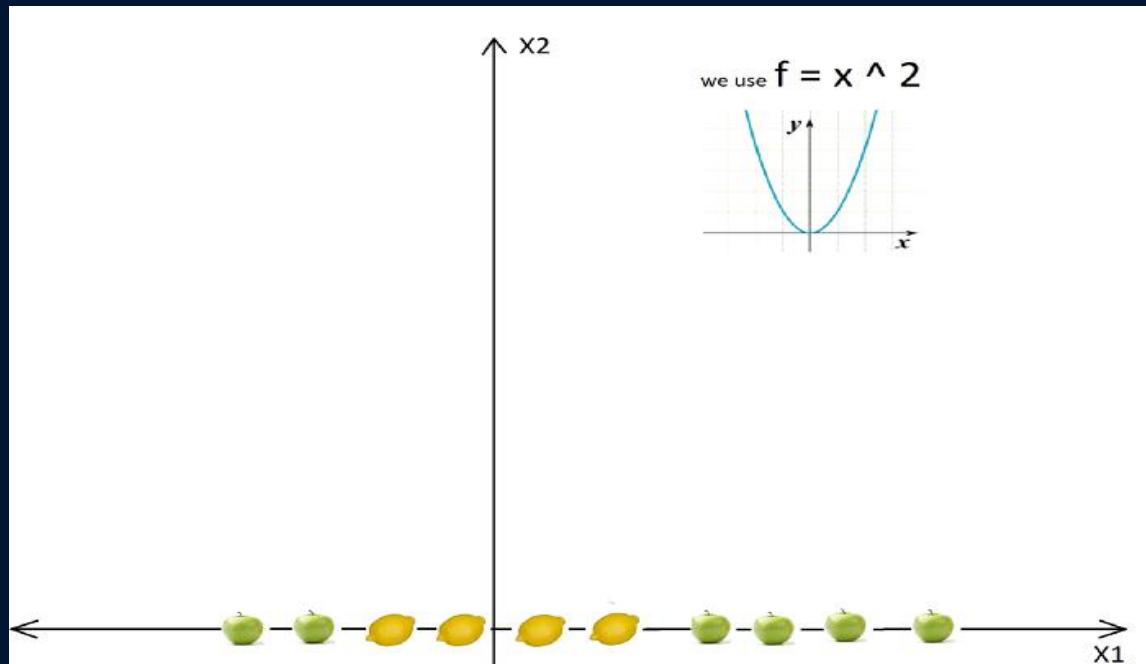
- Now if we consider that the origin is the lemon from the center, we will have something like this:



- Now we can easily separate the two classes. These transformations are called **kernels**. Popular kernels are: **Polynomial Kernel**, **Gaussian Kernel**, **Radial Basis Function (RBF)**, **Laplace RBF Kernel**, **Sigmoid Kernel**, **Anova RBF Kernel**, etc.

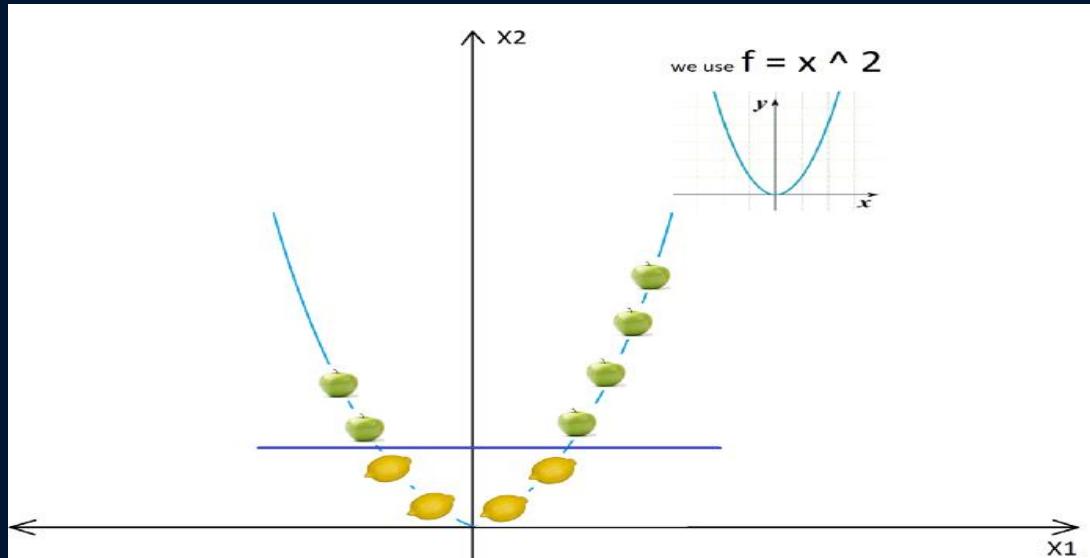
# Support Vector Machine: Part 3

- Mapping from 1D to 2D
- Another, easier example in 2D would be:



# Support Vector Machine: Part 3

After using the kernel and after all the transformations we will get:



**So after the transformation, we can easily delimit the two classes using just a single line**

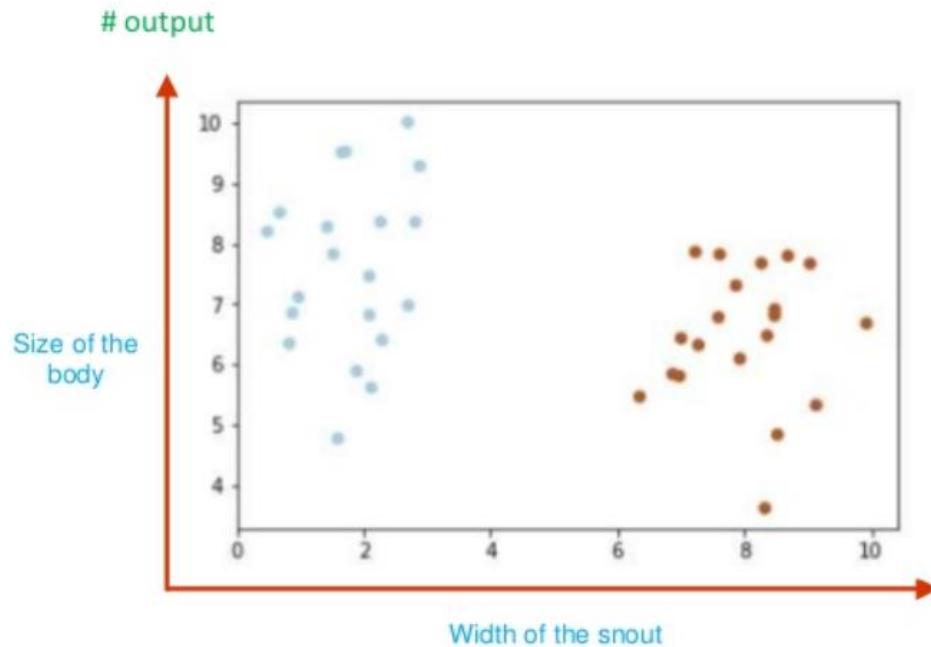


```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.datasets.samples_generator import make_blobs
# we create 40 separable points
X, y = make_blobs(n_samples=40, centers=2, random_state=20)

# fit the model, don't regularize for illustration purposes
clf = svm.SVC(kernel='linear', C=1000)
clf.fit(X, y)

plt.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=plt.cm.Paired)
```

Activate Web  
Go to Sett



Activate W  
Go to Sett



```
# fit the model, don't regularize for illustration purposes
clf = svm.SVC(kernel='linear', C=1000)
clf.fit(X, y)

plt.scatter(X[:, 0], X[:, 1], c=y, s=30,cmap=plt.cm.Paired)
plt.show

# plot the decision function
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()

# create grid to evaluate model
xx = np.linspace(xlim[0], xlim[1], 30)
yy = np.linspace(ylim[0], ylim[1], 30)
YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T
Z = clf.decision_function(xy).reshape(XX.shape)
```

---

# Thank You!

