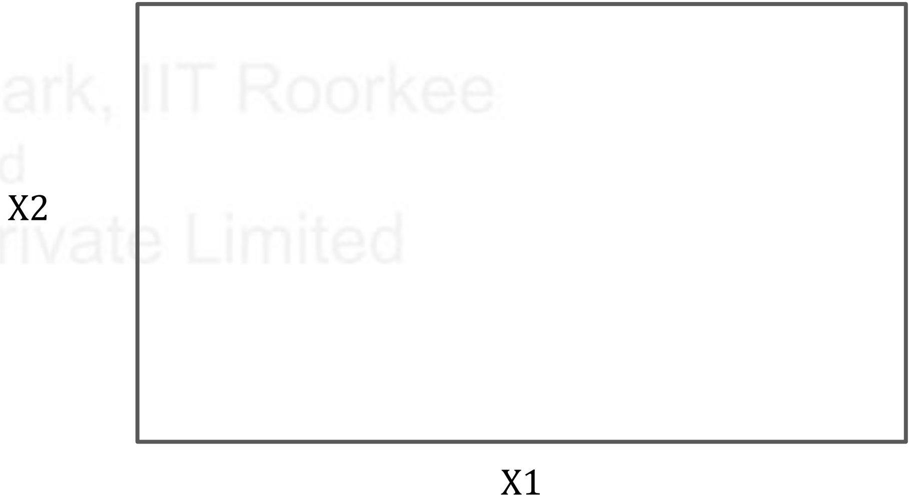
Support Vector Machines

Theory and Intuition - Kernel Trick and Math

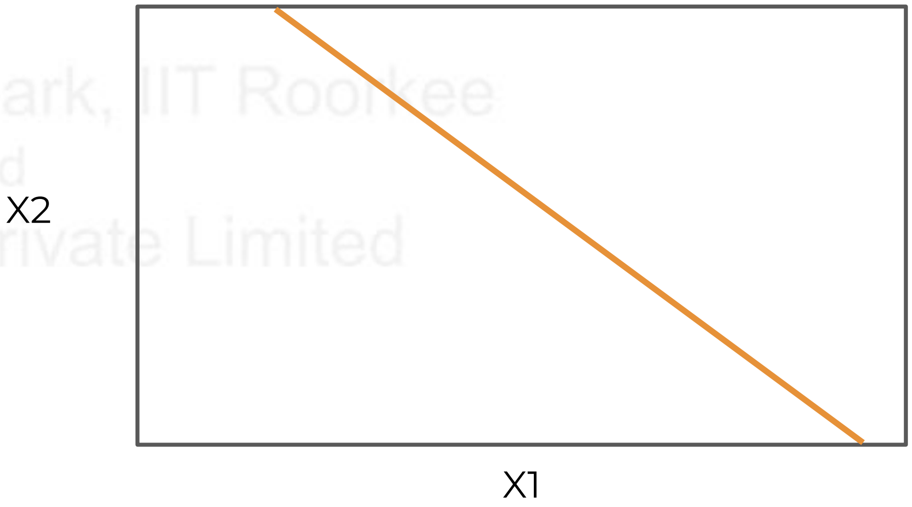
Let’s briefly talk about general mathematics of SVM and how it is related to the Scikit-Learn class calls.

Feel free to consider this an “optional” lecture.

Hyperplanes Defined



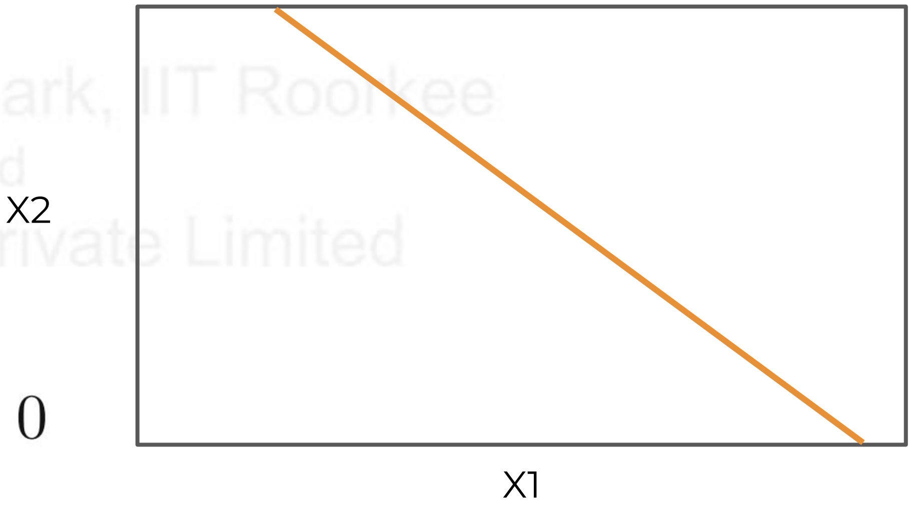
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$



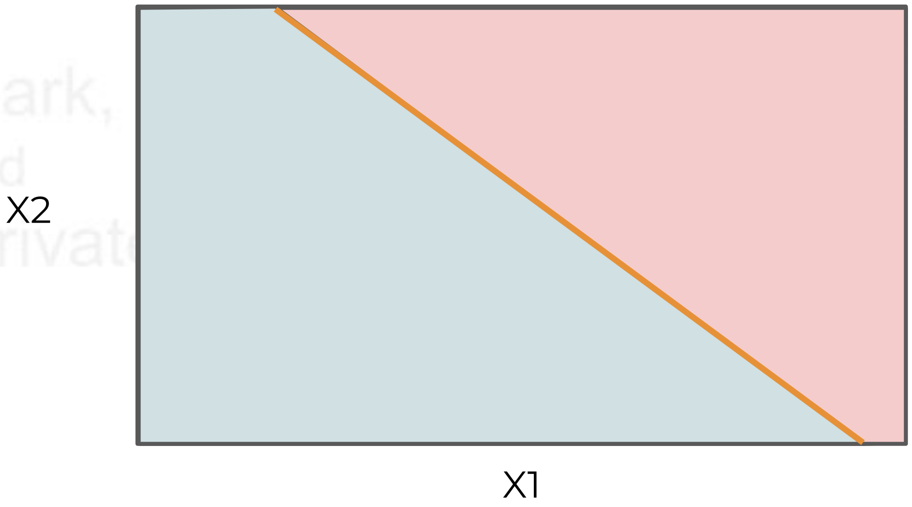
Hyperplanes Defined

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

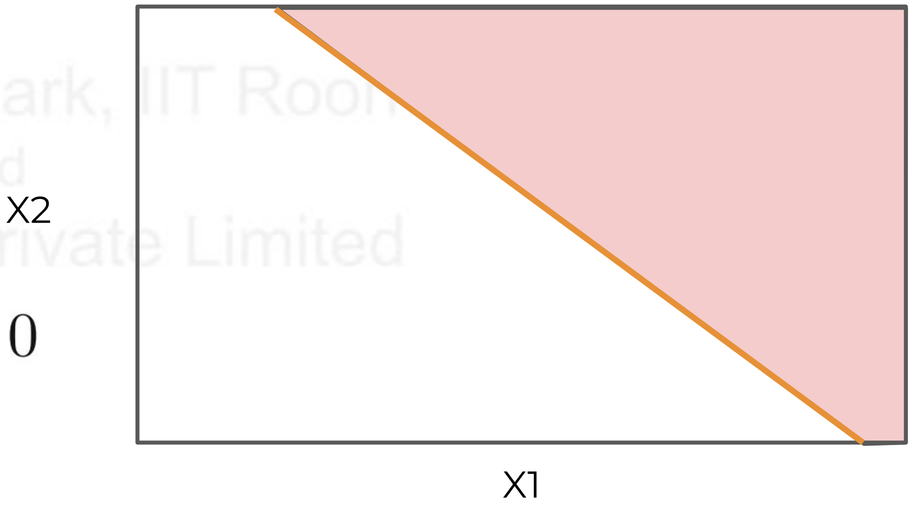
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$



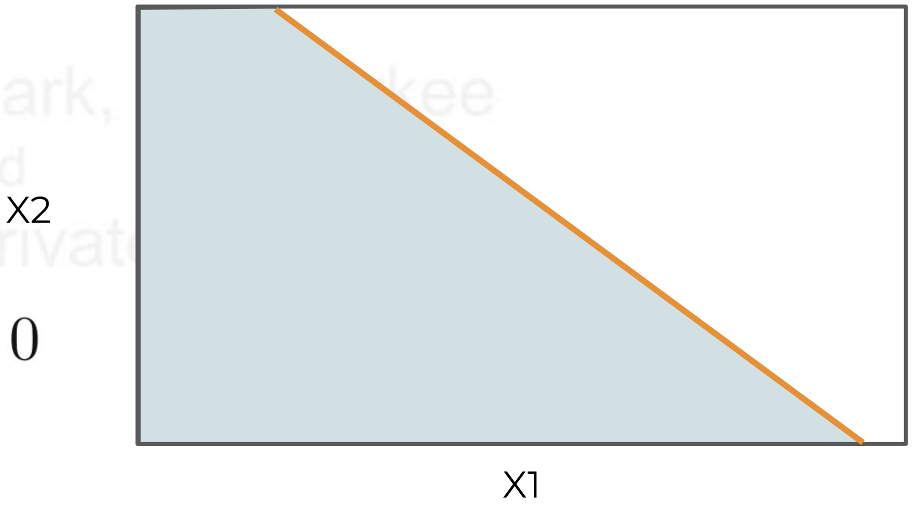
Separating Hyperplanes



$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p > 0$$



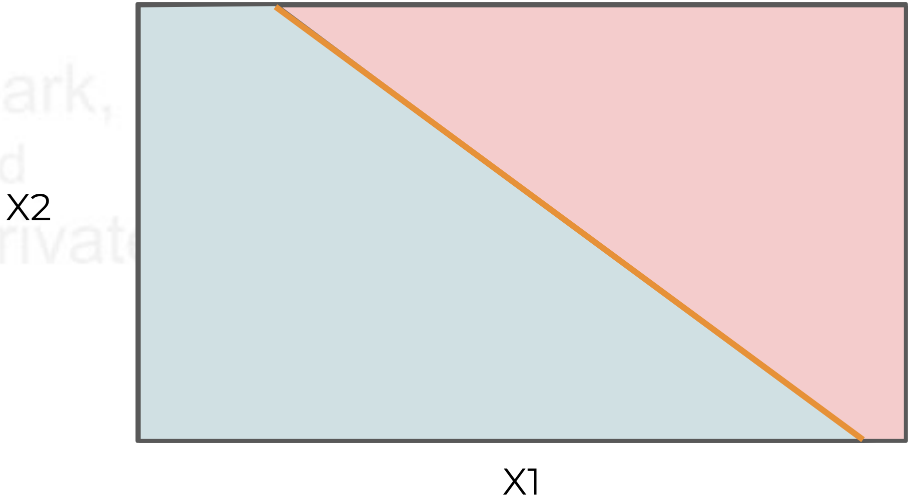
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p < 0$$



Separating Hyperplanes

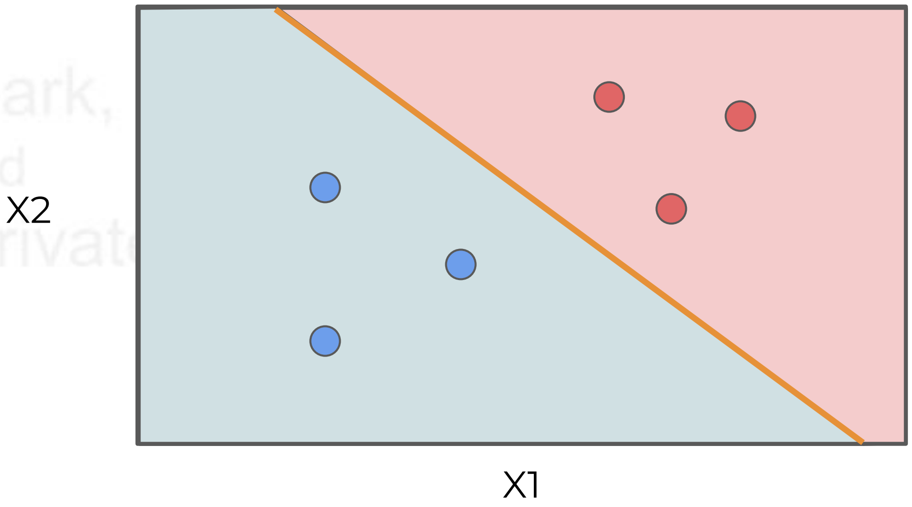
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p > 0$$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p < 0$$



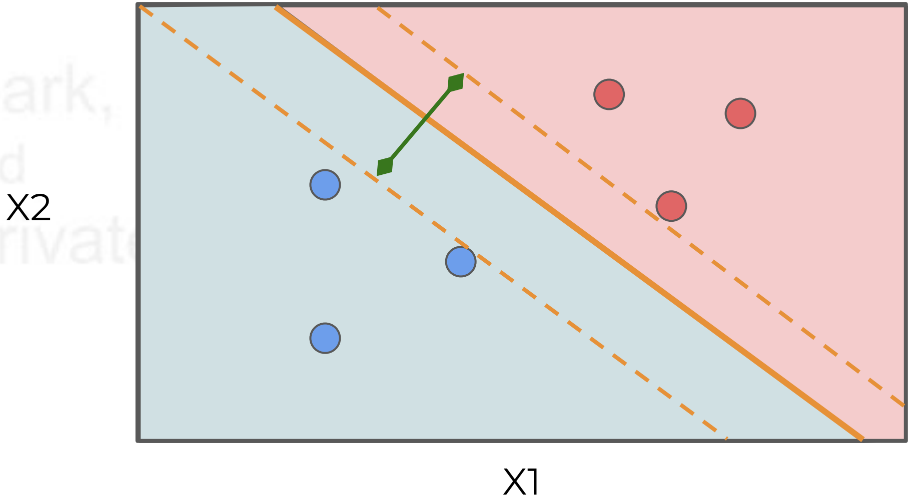
Data Points

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$



Max Margin Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \ \boxed{M}$$



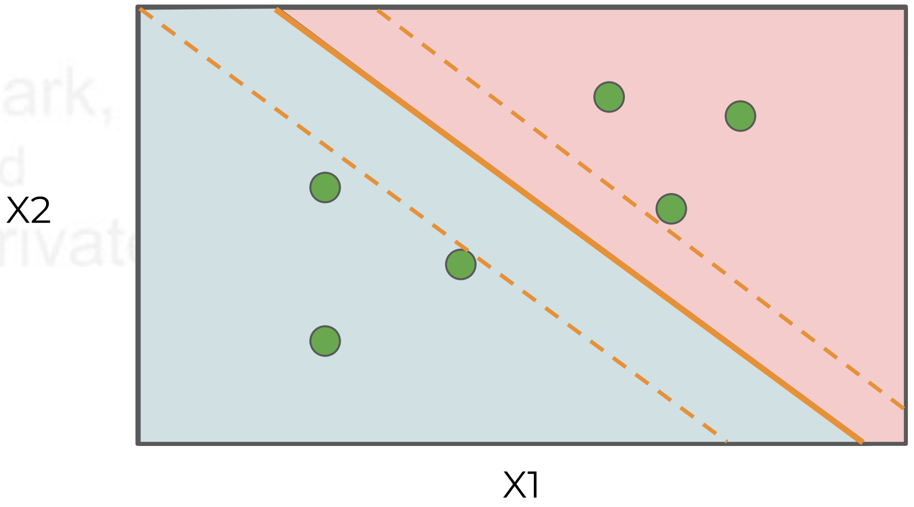
Max Margin Classifier

$$\boxed{x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}}$$

$$\underset{\beta_0, \beta_1, \ldots, \beta_p, M}{\text{maximize}} \quad M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \; \boxed{\forall\, i = 1, \ldots, n.}$$

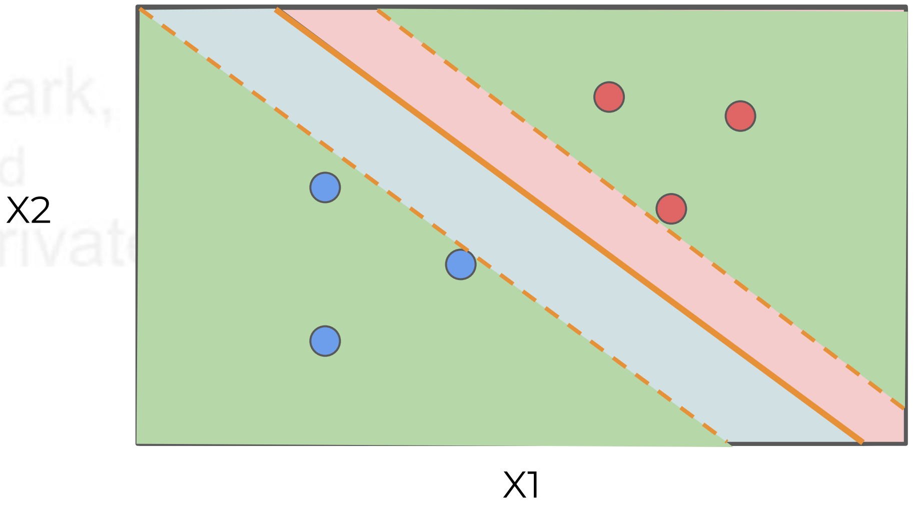


$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

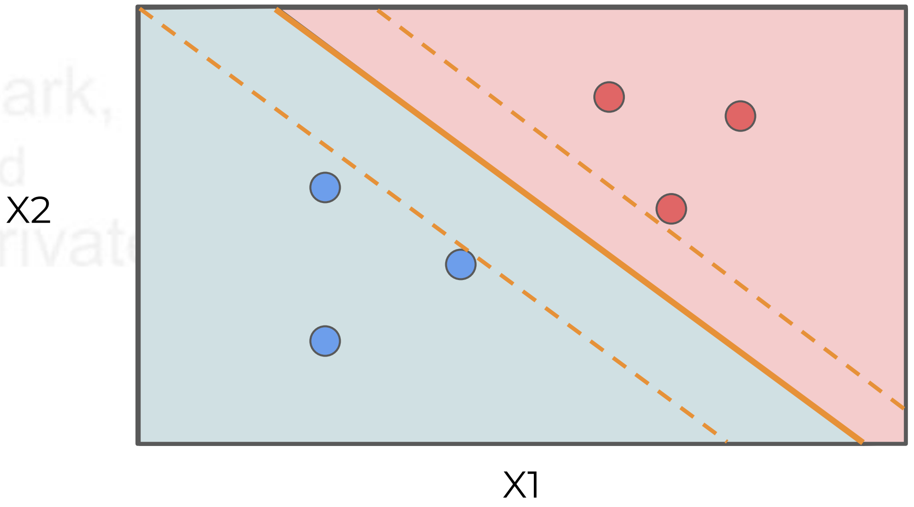
$$\underset{\beta_0, \beta_1, \ldots, \beta_p, M}{\text{maximize}} \quad M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

$$\boxed{y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M} \quad \forall\, i = 1, \ldots, n.$$



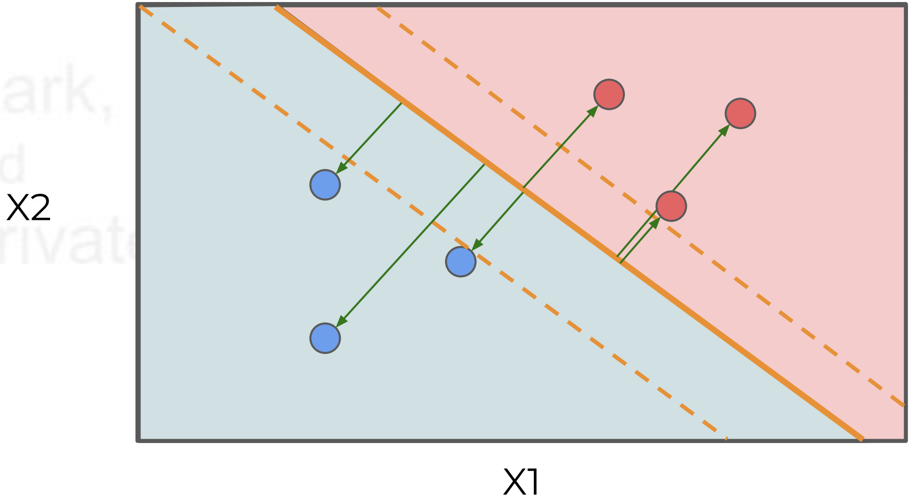
$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$



$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \boxed{\sum_{j=1}^{p} \beta_j^2 = 1}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \quad \forall\, i = 1, \ldots, n.$$



$$\text{subject to } \boxed{\sum_{j=1}^{p} \beta_j^2 = 1}$$

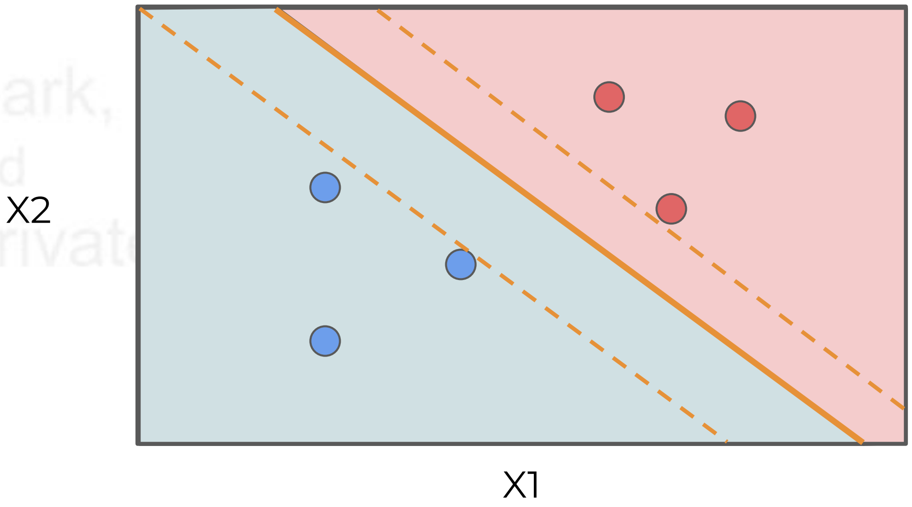
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip})$$

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

$$\underset{\beta_0, \beta_1, \ldots, \beta_p, M}{\text{maximize}} \quad M$$

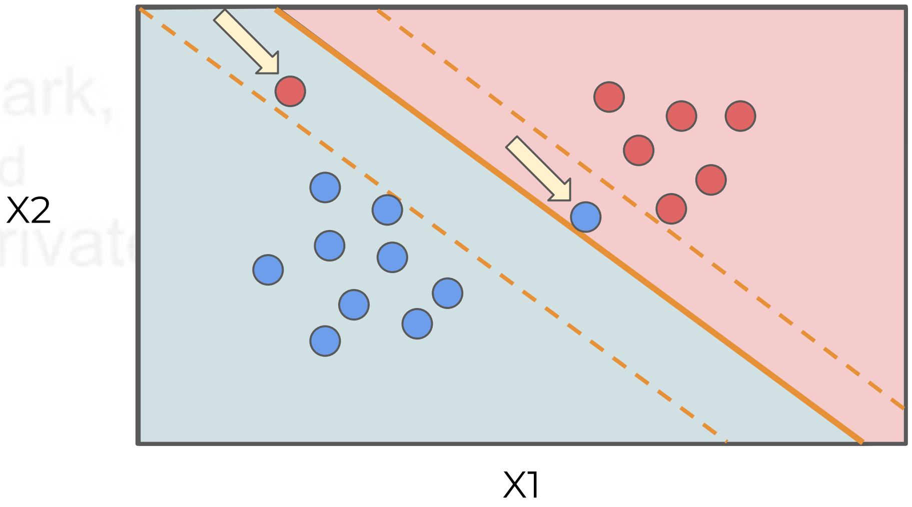
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \ \ \forall\, i = 1, \ldots, n.$$



Support Vector Classifier

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$



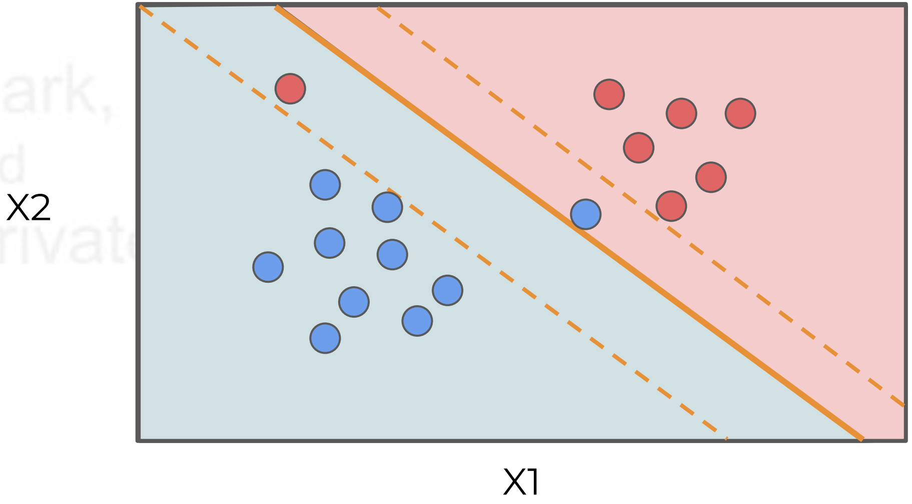
Support Vector Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$



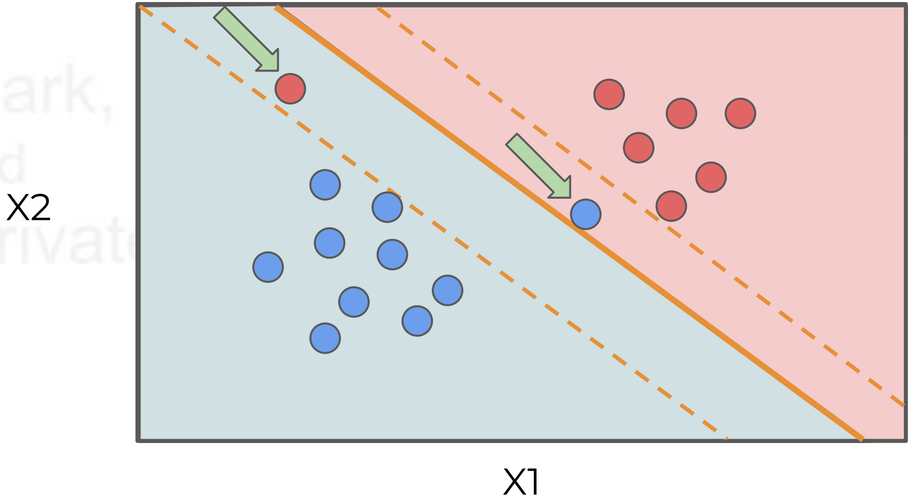
Support Vector Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1$$

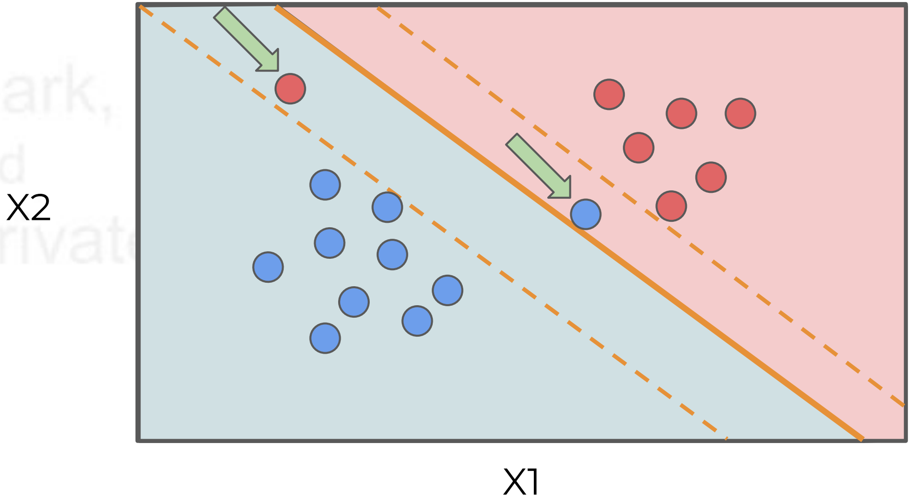
$$\boxed{\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M\boxed{(1-\epsilon_i)}$$



C : *float, default=1.0*

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

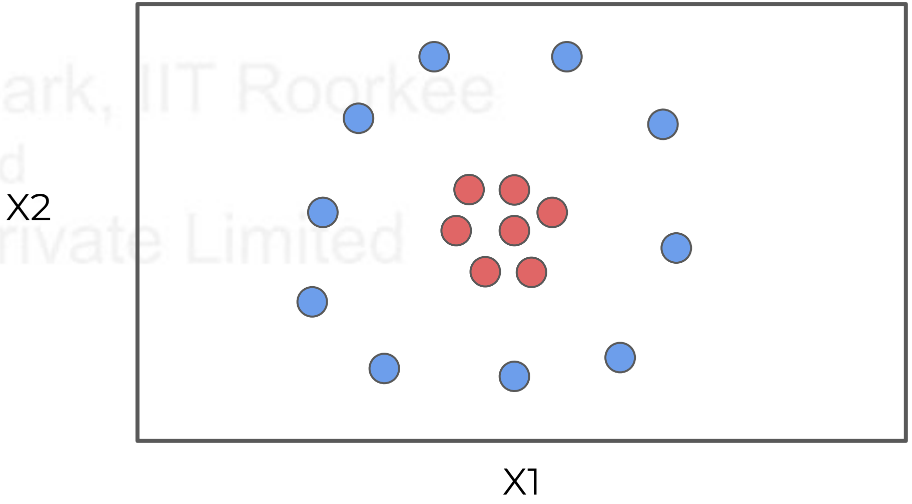


$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

Support Vector Machines

$$X_1, X_2, \ldots, X_p,$$

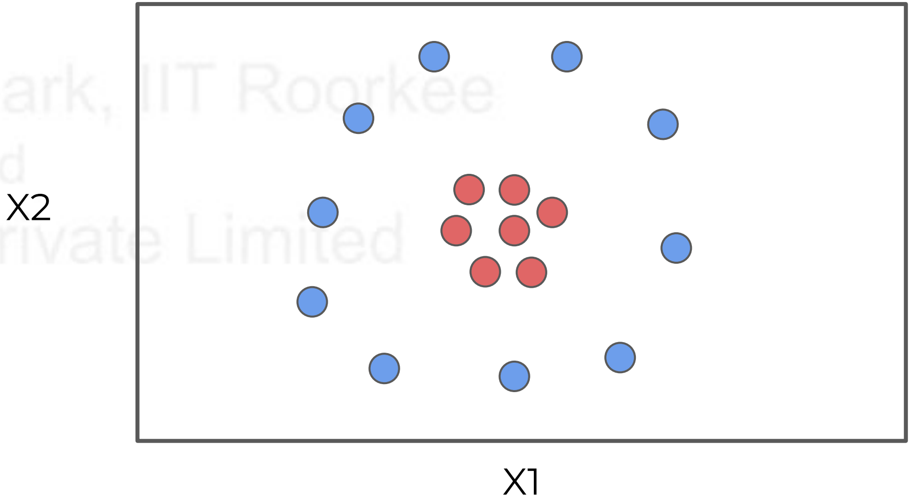


Support Vector Machines

$$X_1, X_2, \ldots, X_p,$$

$$\Downarrow$$

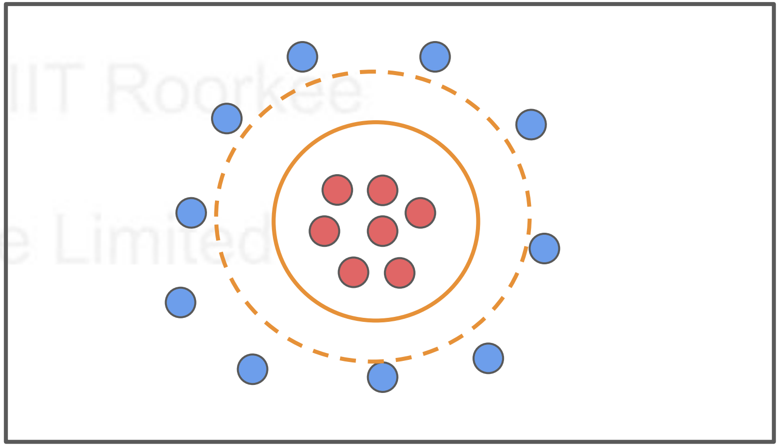
$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$$



Support Vector Machines

$$\underset{\beta_0,\beta_{11},\beta_{12}\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

X2



X1

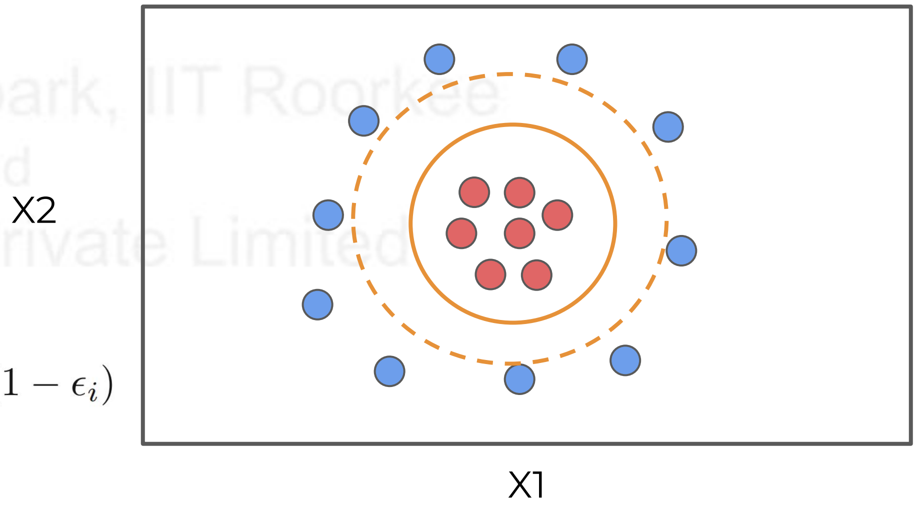
Support Vector Machines

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$$

$$\underset{\beta_0,\beta_{11},\beta_{12}\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to } y_i\left(\beta_0 + \sum_{j=1}^{p}\beta_{j1}x_{ij} + \sum_{j=1}^{p}\beta_{j2}x_{ij}^2\right) \geq M(1-\epsilon_i)$$

$$\sum_{i=1}^{n}\epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2 = 1.$$



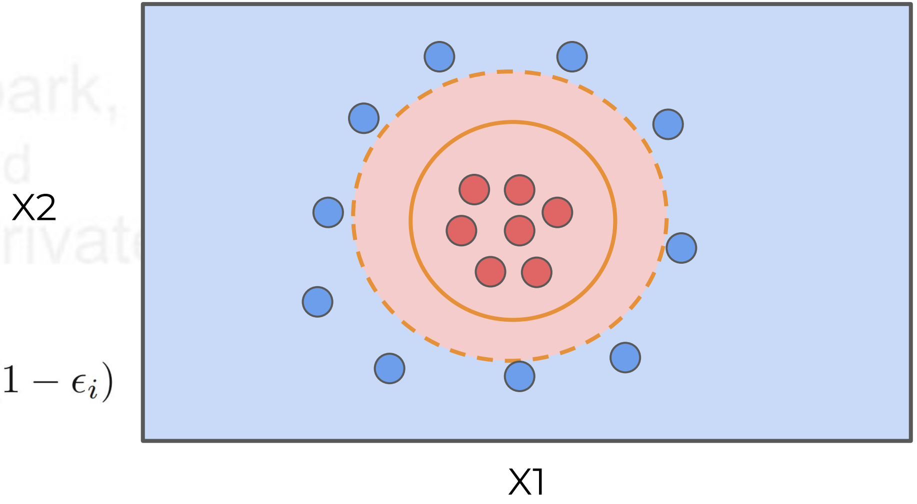
Support Vector Machines

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$$

$$\underset{\beta_0, \beta_{11}, \beta_{12} \ldots, \beta_{p1}, \beta_{p2}, \epsilon_1, \ldots, \epsilon_n, M}{\text{maximize}} \quad M$$

$$\text{subject to } y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i)$$

$$\sum_{i=1}^{n} \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1.$$

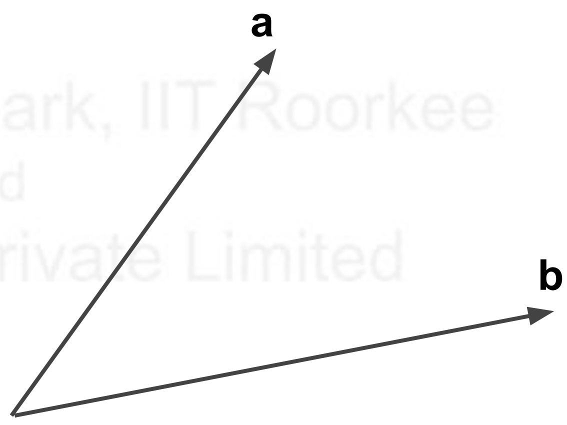


How to deal with very large feature space? As polynomial order growS, the number of computations necessary to solve for margins also grows!

We use **Kernel trick** which makes use of the **inner product** of vectors, also known as the **dot product**.

Dot Product

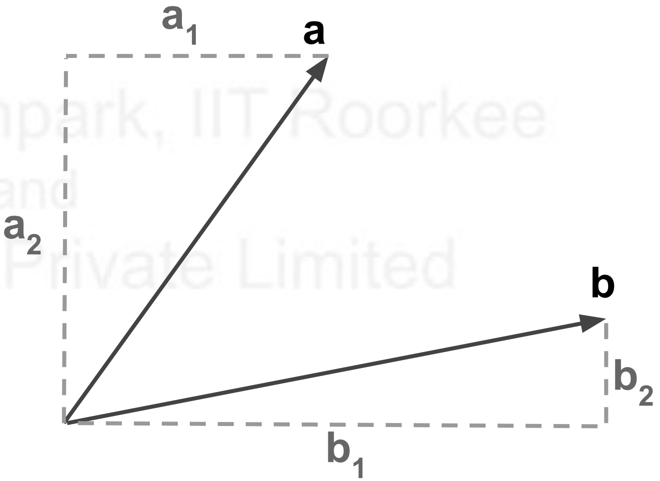
$$\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$$



Dot Product

$$\langle a, b\rangle = \sum_{i=1}^{r} a_i b_i$$

$$a \cdot b = a_1 b_1 + a_2 b_2$$

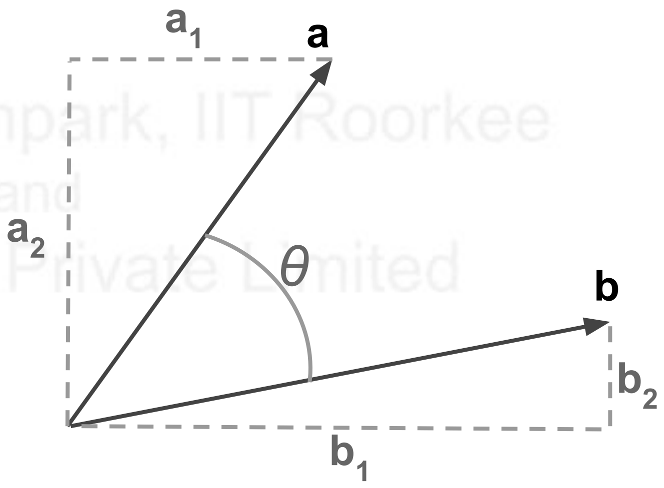


Dot Product

$$\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i$$

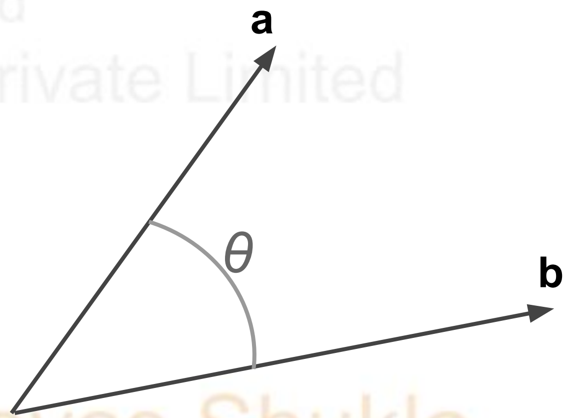
$$a \cdot b = a_1 b_1 + a_2 b_2$$

$$a \cdot b = |a||b|cos(\theta)$$



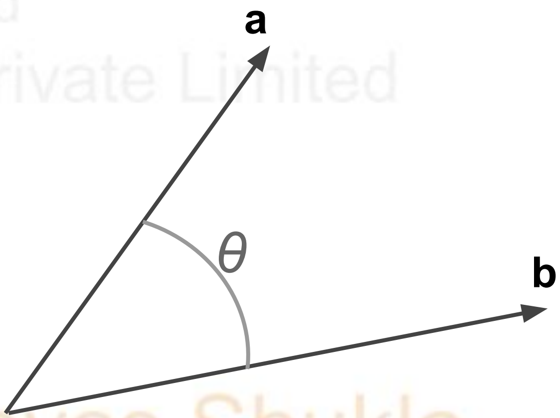
Notice how the dot product can be thought of as a similarity between the vectors.

$$a \cdot b = |a||b|cos(\theta)$$



- $\cos(0^\circ) = 1$
- $\cos(90^\circ) = 0$
- $\cos(180^\circ) = -1$

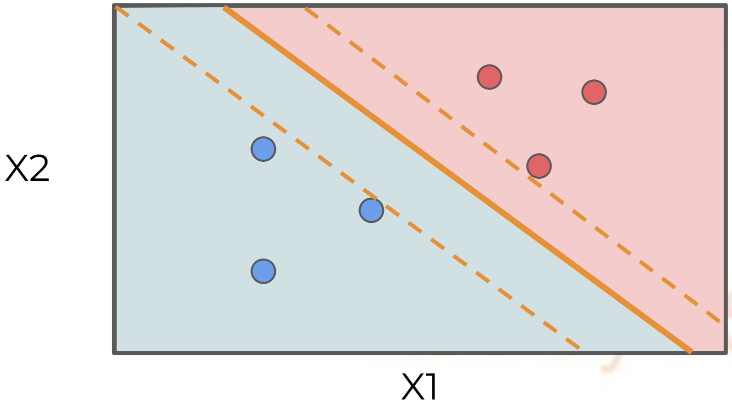
$$a \cdot b = |a||b|cos(\theta)$$



Let's discuss Kernel Trick

Linear Support Vector Classifier rewritten:

$$f(x) = \beta_0 + \boxed{\sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle}$$

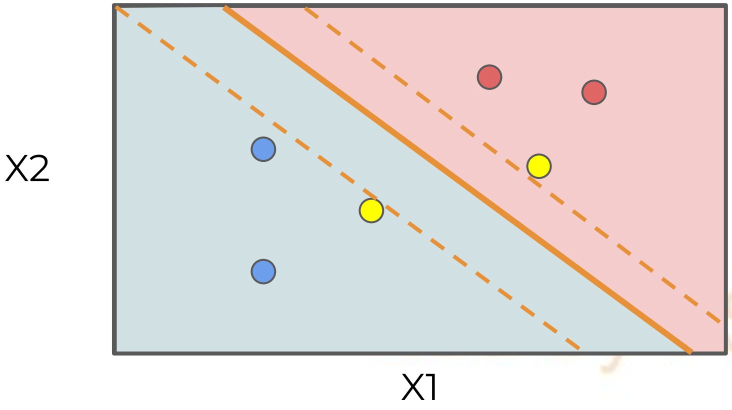


Calculating the inner products of all pairs of training observations

Linear Support Vector Classifier rewritten:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \boxed{\alpha_i} \langle x, x_i \rangle$$

Only non-zero for the support vectors.



Linear Support Vector Classifier rewritten:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

$$\Downarrow$$

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

Linear Support Vector Classifier rewritten:

$$f(x) = \beta_0 + \sum_{\boxed{i \in \mathcal{S}}} \alpha_i \langle x, x_i \rangle$$

S collection of indices of these support points

Kernel Function

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

A kernel is a function that quantifies the similarity of two observations.

Kernel Function

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

Kernel Function

$$K(x_i, x_{i'}) = \boxed{\sum_{j=1}^{p} x_{ij} x_{i'j}}$$

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \boxed{\langle x, x_i \rangle}$$

$$\boxed{\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i}$$

Kernel Function

$$\boxed{K(x_i, x_{i'})} = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \boxed{K(x, x_i)}$$

$$\boxed{\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i}$$

Polynomial Kernel

$$\boxed{K(x_i, x_{i'})} = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \boxed{\langle x, x_i \rangle}$$

$$\boxed{\langle a, b \rangle = \sum_{i=1}^{r} a_i b_i}$$

Radial Basis Kernel

$$\boxed{K(x_i, x_{i'})} = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \boxed{\langle x, x_i \rangle}$$

$$\boxed{\langle a, b \rangle = \textstyle\sum_{i=1}^{r} a_i b_i}$$

The use of **kernels** as a replacement is known as the **kernel trick**.

Kernels allow us to avoid computations in the enlarged feature space, by only needing to perform computations for each distinct pair of training points

Intuitively we’ve already seen inner products act as a measurement of similarity between vectors.

The use of kernels can be thought of as a measure of similarity between the original feature space and the enlarged feature space.

Tree Based Methods

Tree Based Methods

Three main methods:

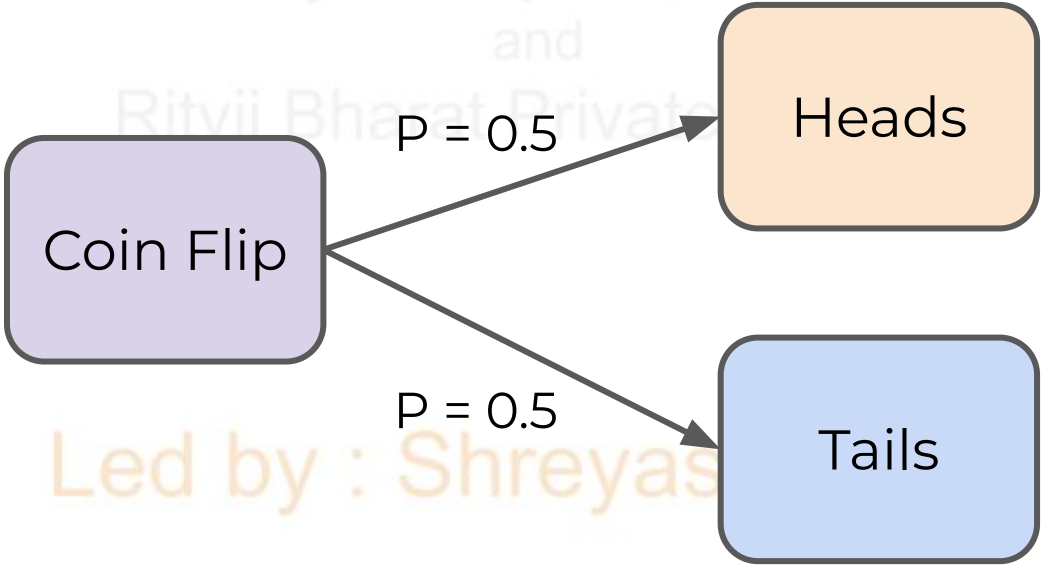
- Decision Trees
- Random Forests
- Boosted Trees

Decision Trees

Theory and Intuition

- While the use of basic decision trees for modeling choices and outcomes have been around for a very long time, statistical decision trees are a more recent development.
- **Note the difference here**!

The general term “decision tree” can refer to a flowchart mapping out outcomes.



Decision Tree Learning refers to the statistical modeling that uses a form of decision trees, where node splits are decided based on an information metric.

Decision trees methods is basically the ability to split data based on information from features.

We need a mathematical definition of information and the ability to measure it.

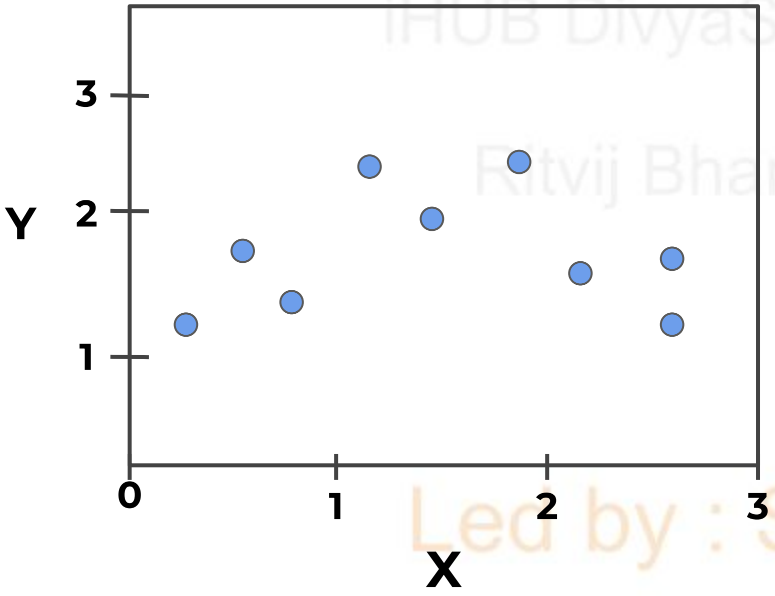
The ability to measure and define information will become more important as we learn the mathematics of tree based methods.

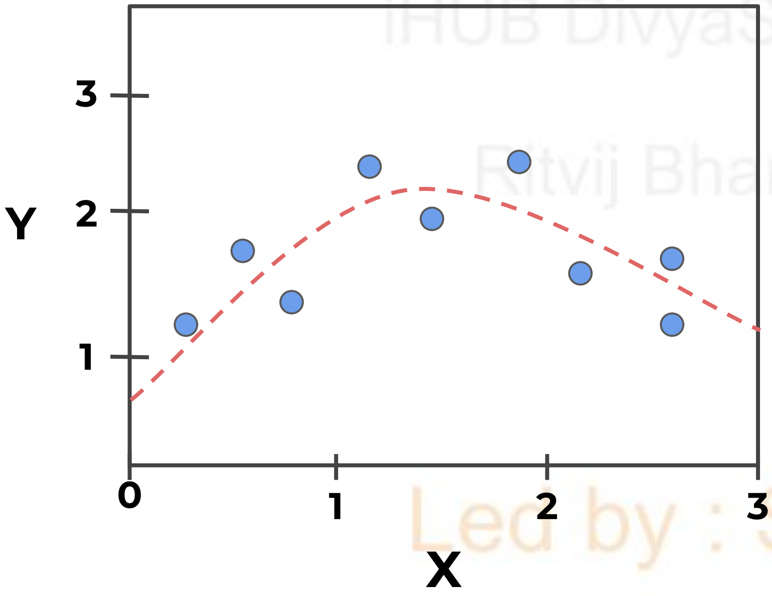
Let’s talk about the development of decision trees.

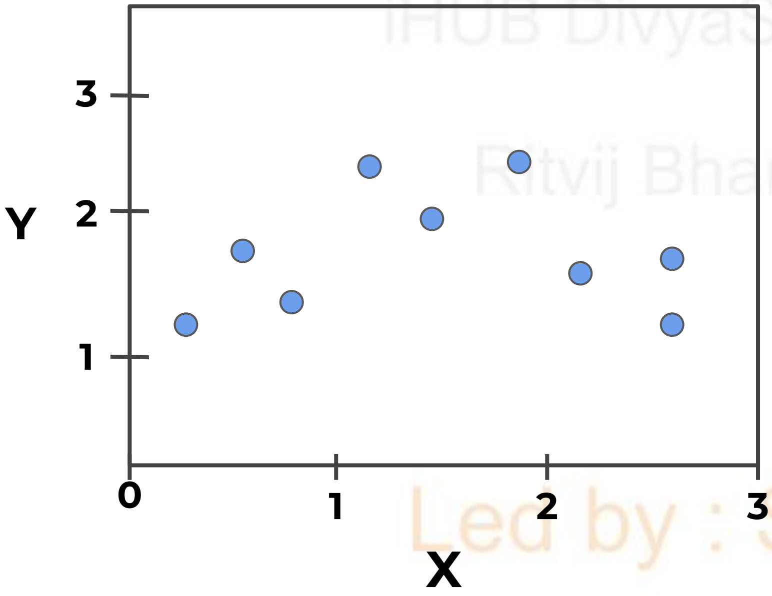
1963: First publication of regression tree algorithm by Morgan and Sonquist

1963: Morgan and Sonquist created piecewise-constant model with splits.

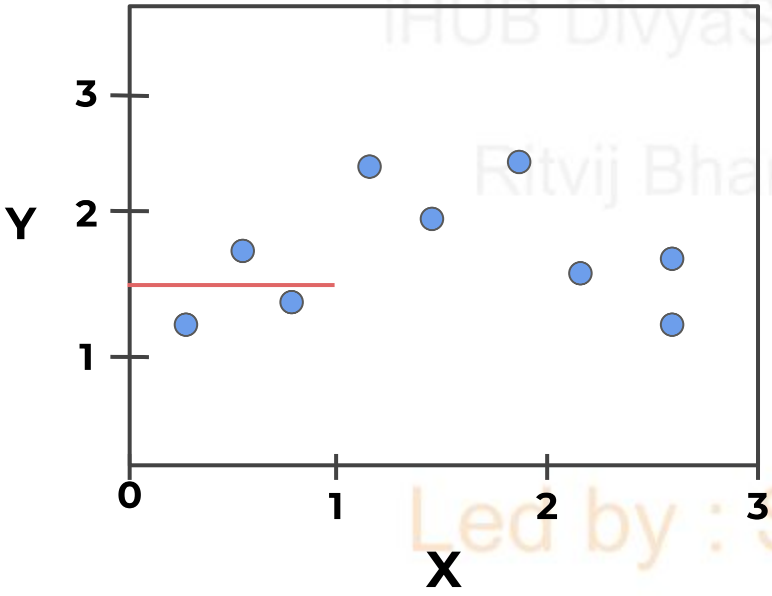
1963: Piecewise-constant regression tree

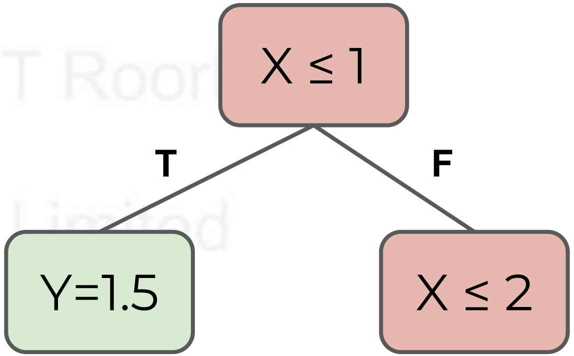

Y
3
2
1
0
1
2
3
X

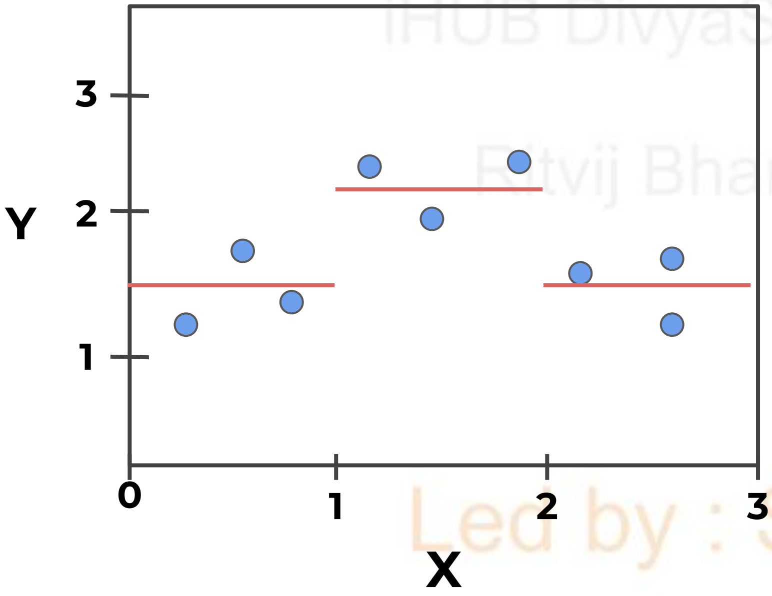

Y
3
2
1
0
1
2
3
X

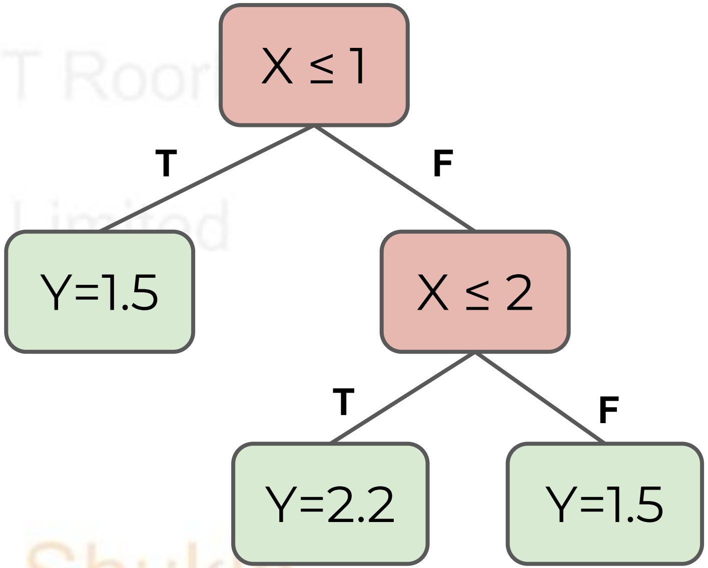


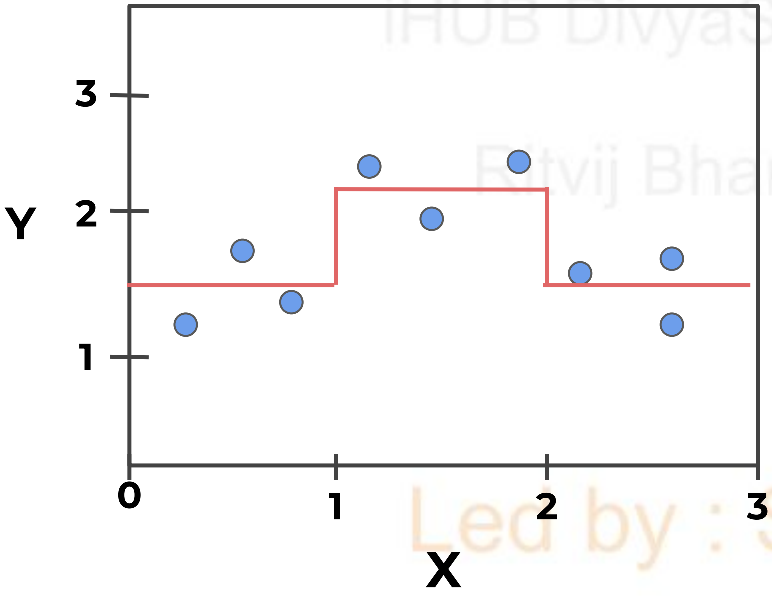
X < 1

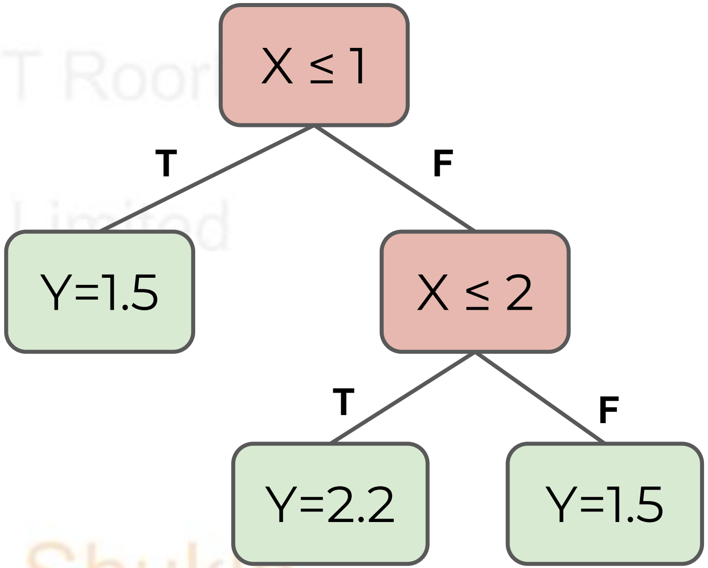

Y
3
2
1
0
1
2
3
X


X ≤ 1
T
F
Y=1.5
X ≤ 2


Y
X
0
1
2
3
1
2
3


X ≤ 1
T
F
Y=1.5
X ≤ 2
T
F
Y=2.2
Y=1.5


3
2
1
Y
0
1
2
3
X


X ≤ 1
T
F
Y=1.5
X ≤ 2
T
F
Y=2.2
Y=1.5

Node impurity

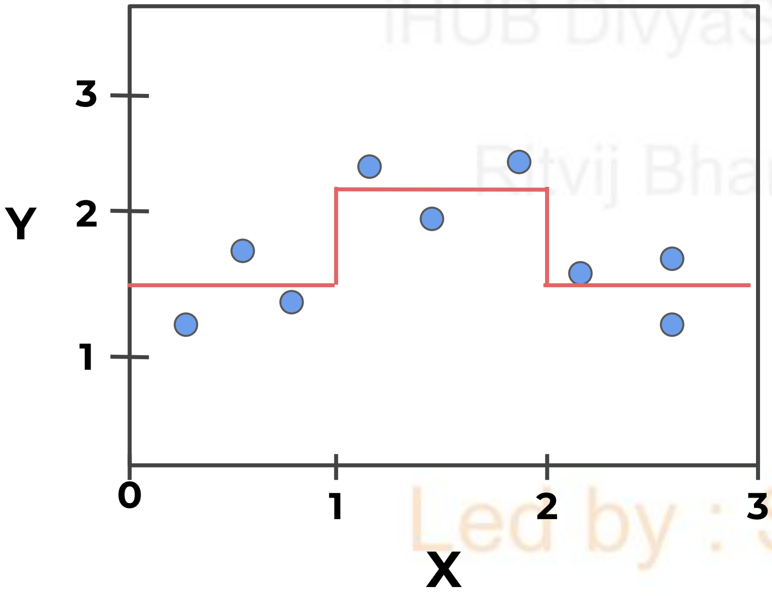
$$\phi(t) = \sum_{i \in t} (y_i - \bar{y})^2$$

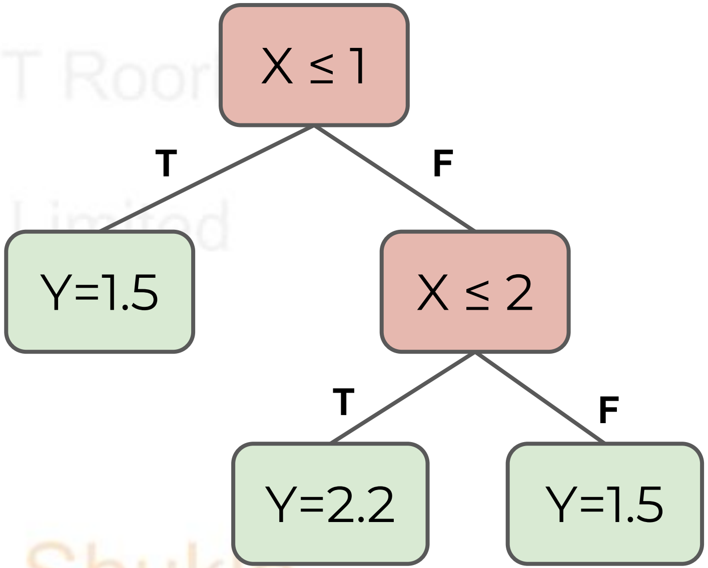
Decision Trees

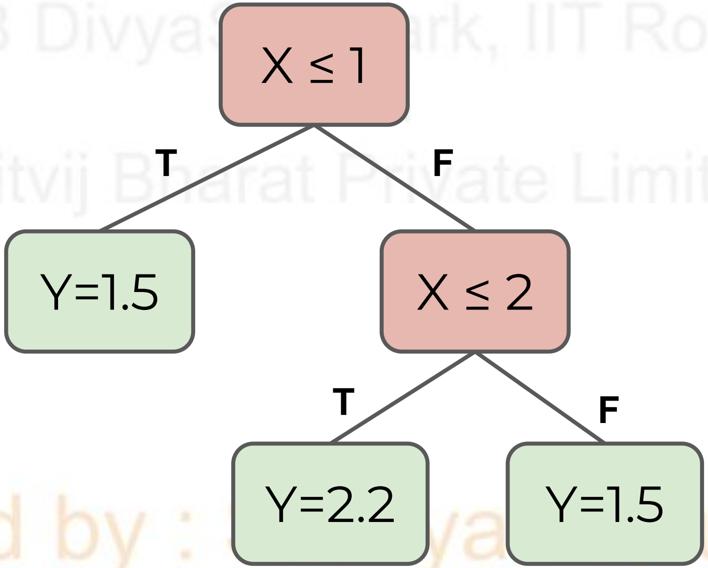
Decision Tree Basics

Let us understand some terminology about the decision tree components.

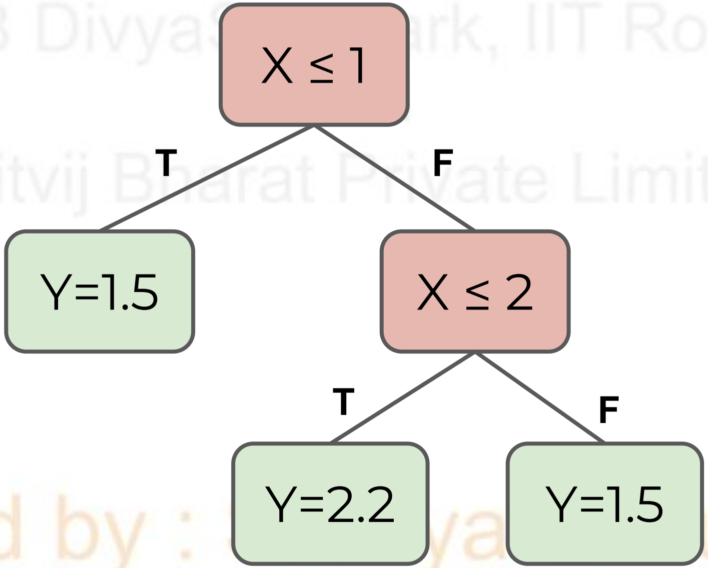
Recall our simple regression tree:



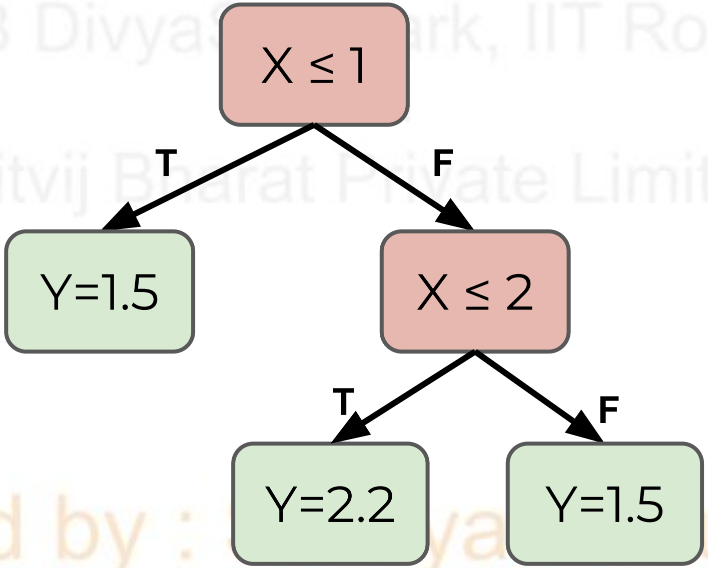



X ≤ 1
T
F
Y=1.5
X ≤ 2
T
F
Y=2.2
Y=1.5

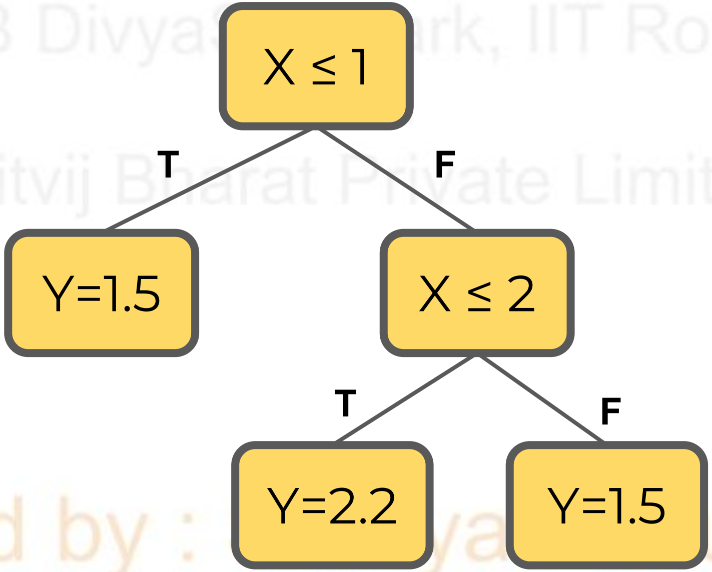
Splitting



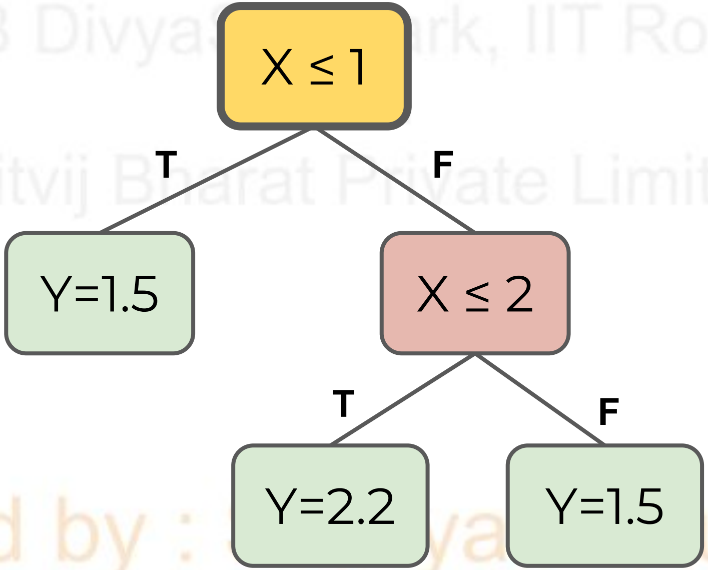
Splitting



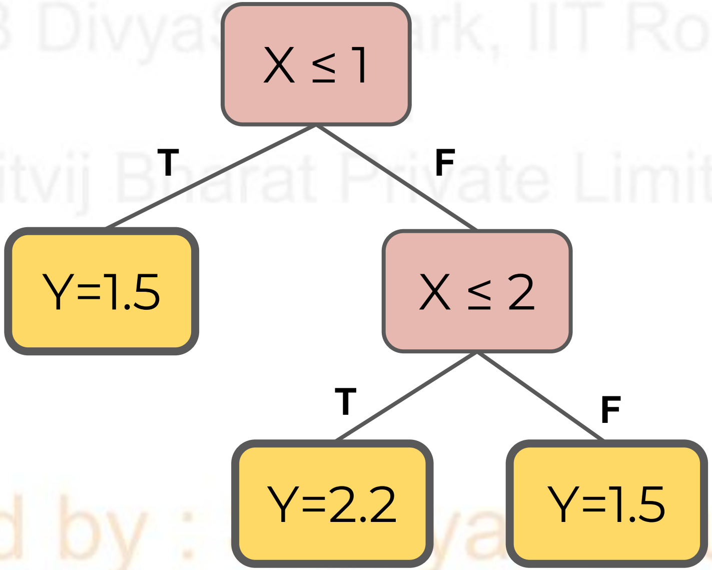
Nodes:



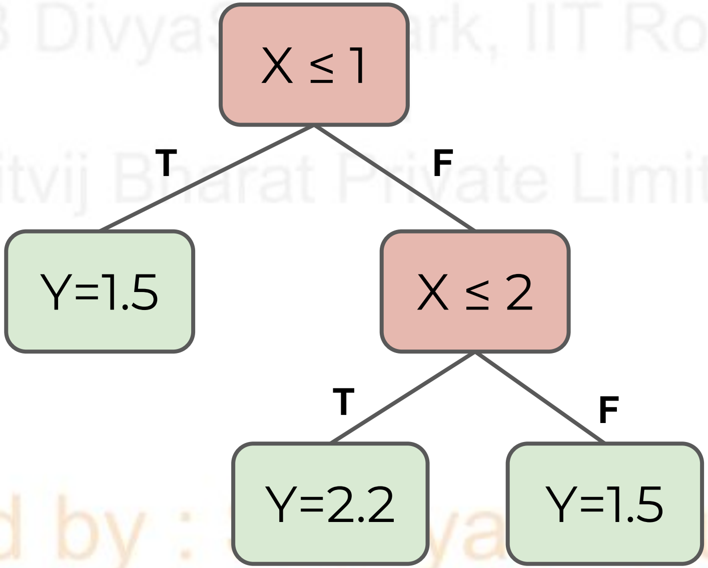
Root Node:



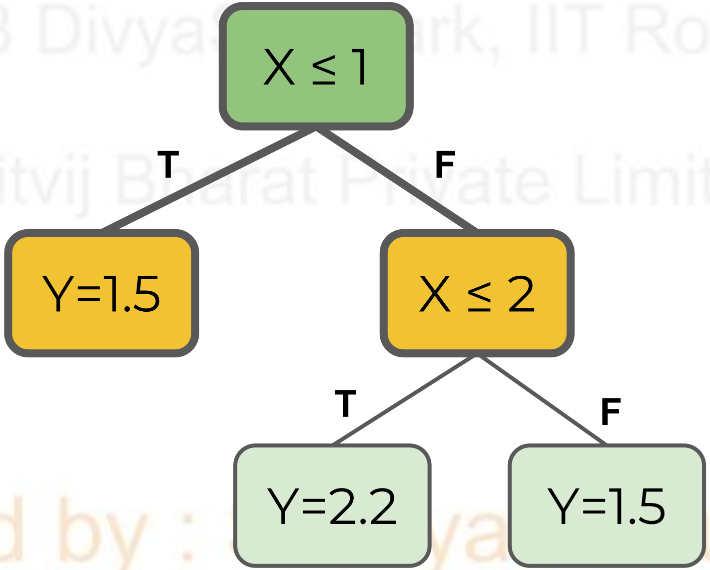
Leaf (Terminal) Nodes:



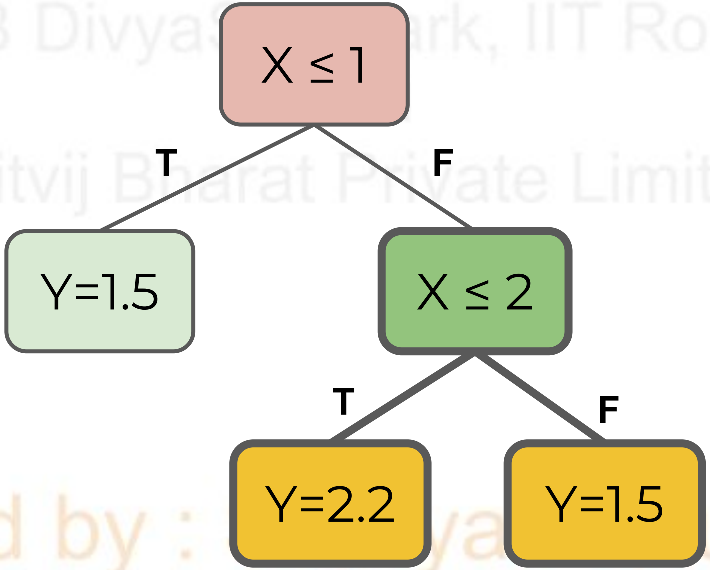
Parent and Children Nodes:



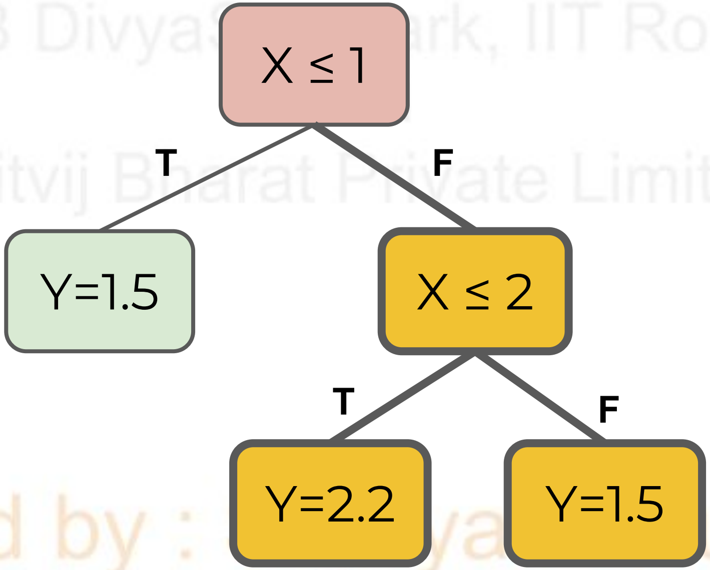
Parent and Children Nodes:



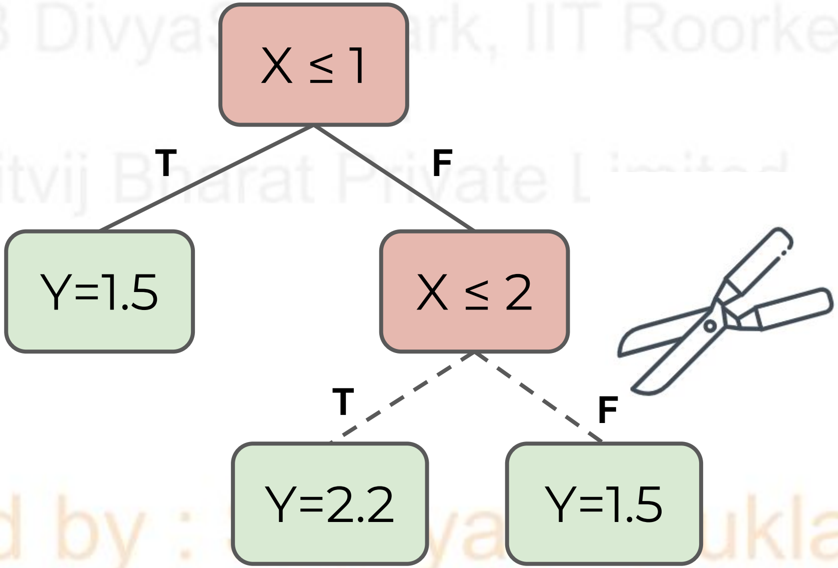
Parent and Children Nodes:



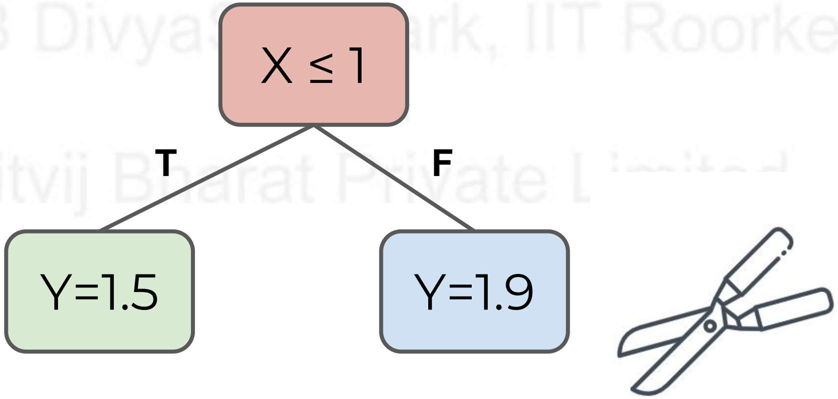
Tree Branches (Sub Trees):



Pruning:



Pruning:



Let’s begin constructing a tree!