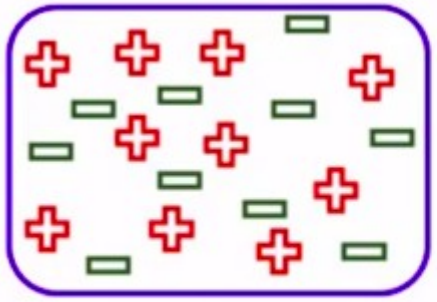
Decision Tree

Example

Lets assume that we have a data of 20 students those who play cricket and those who don’t play cricket.



Total number of students = 20

Play cricket = 10

Do not play cricket = 10

We have the following features of the students:

1) Height

2) Performance in the class

3) Class (which defines the current class of the student)

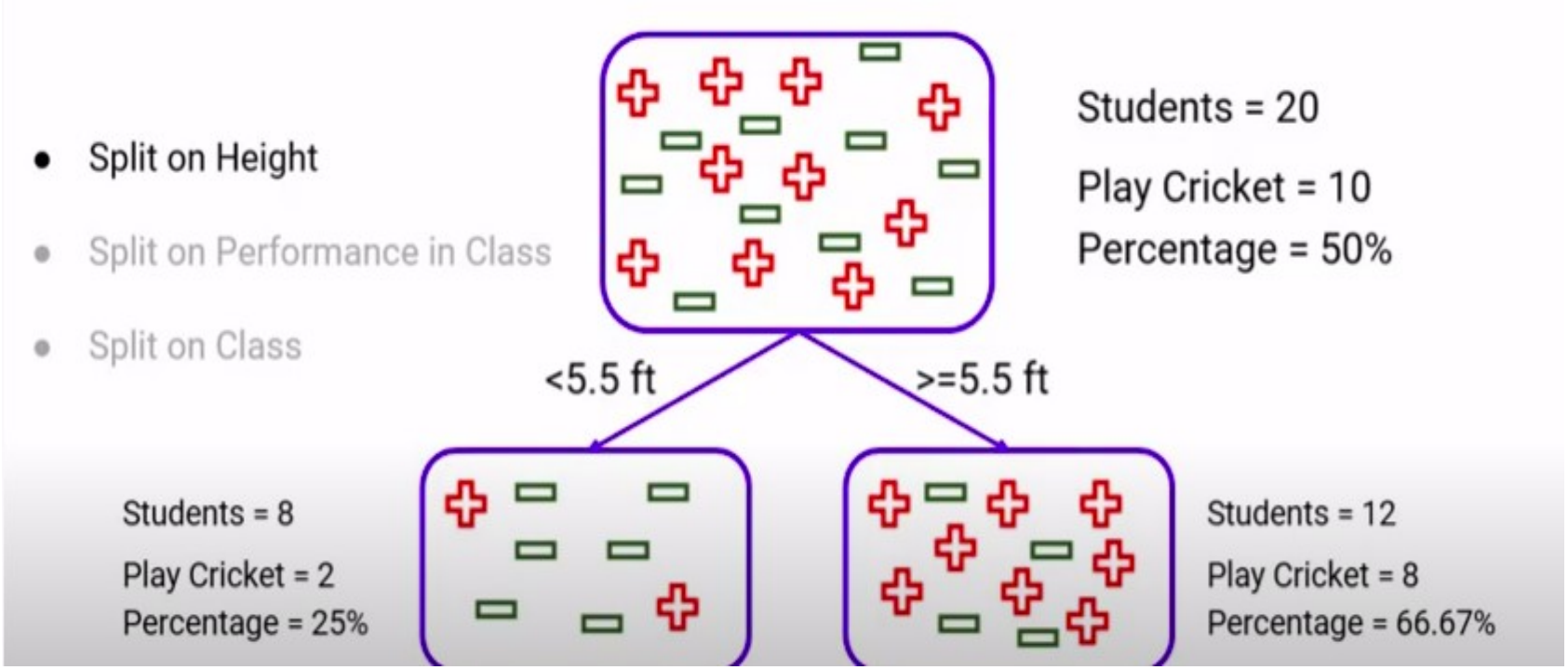
Using this data we want to train a model whether the student will play cricket or not.

Students = 20

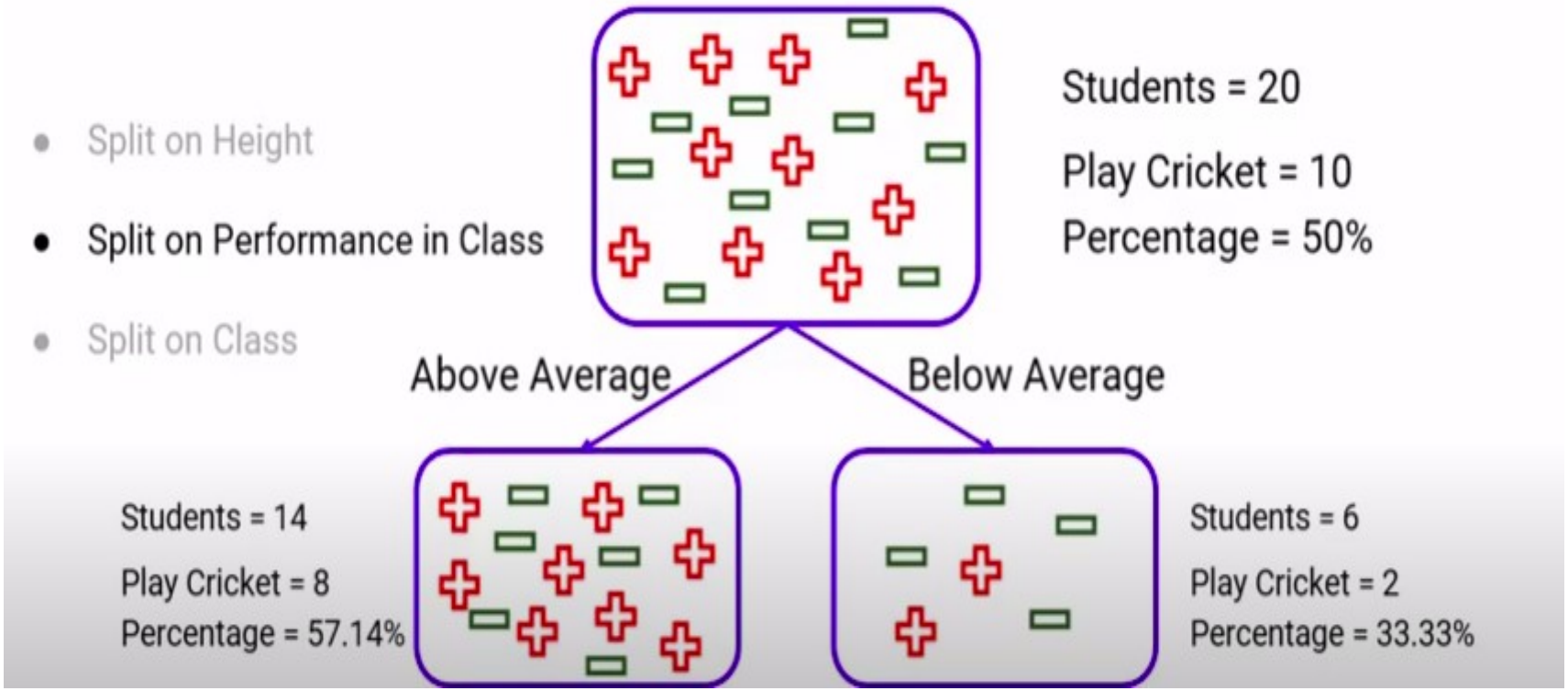
Play Cricket = 10

Percentage = 50%

1) Split Data on the basis of height
2) Split Data on the basis of performance in class
3) Split on class.

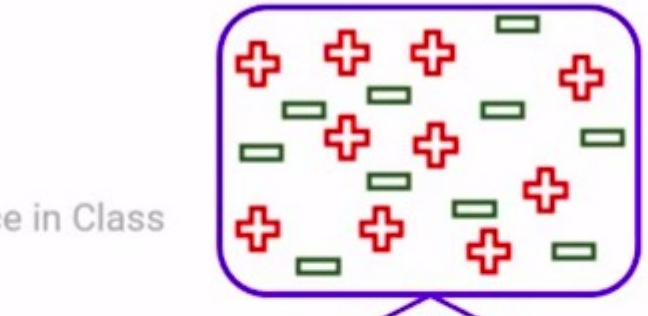


Split on Performance



Split on Class

- Split on Height
- Split on Performance in Class
- Split on Class



Students = 20

Play Cricket = 10

Percentage = 50%

Class IX

Class X

Students = 10

Play Cricket = 8

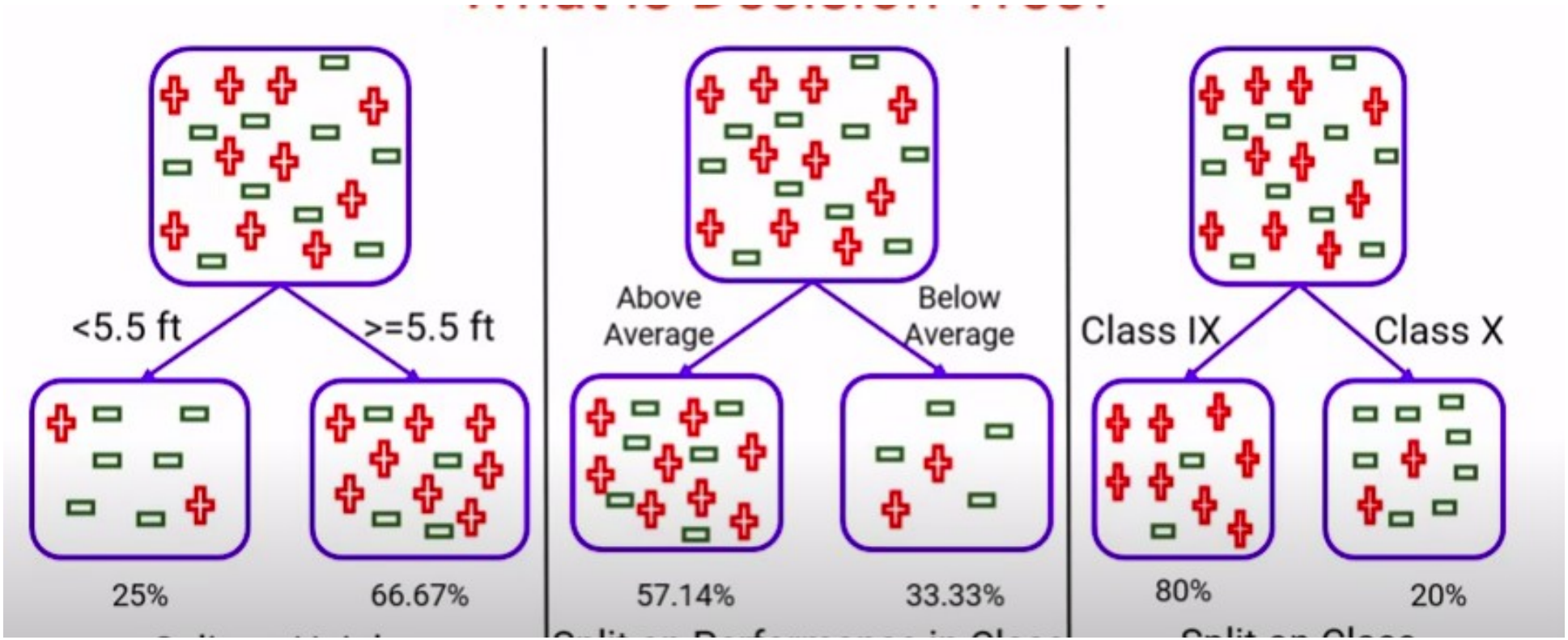
Percentage = 80%

Students = 10

Play Cricket = 2

Percentage = 20%

Which 1 do you think is the best scenario for split?

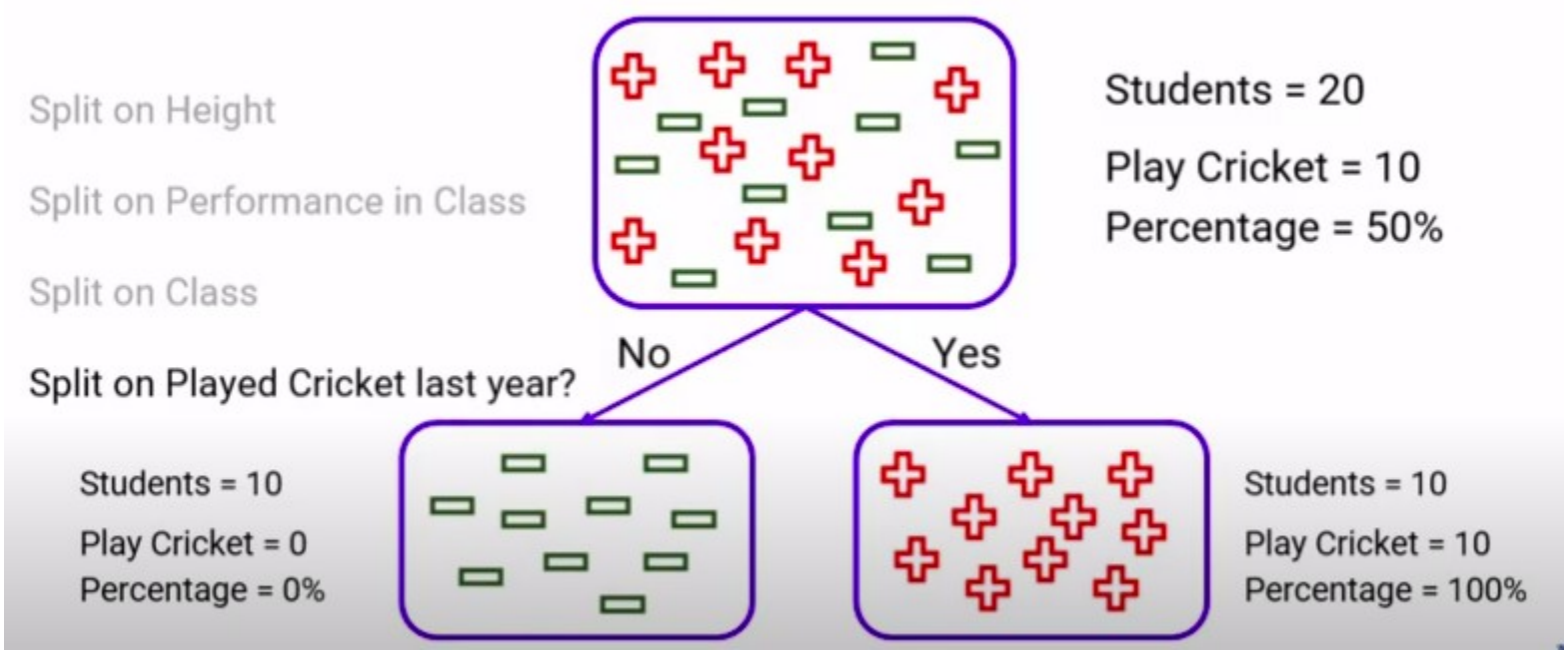


What is an ideal split ?

An ideal split is the one which can segregate the positives completly from the negetives.

Let us understand in the below scenario:

For eg : we can consider one more feature (Played cricket last year?)

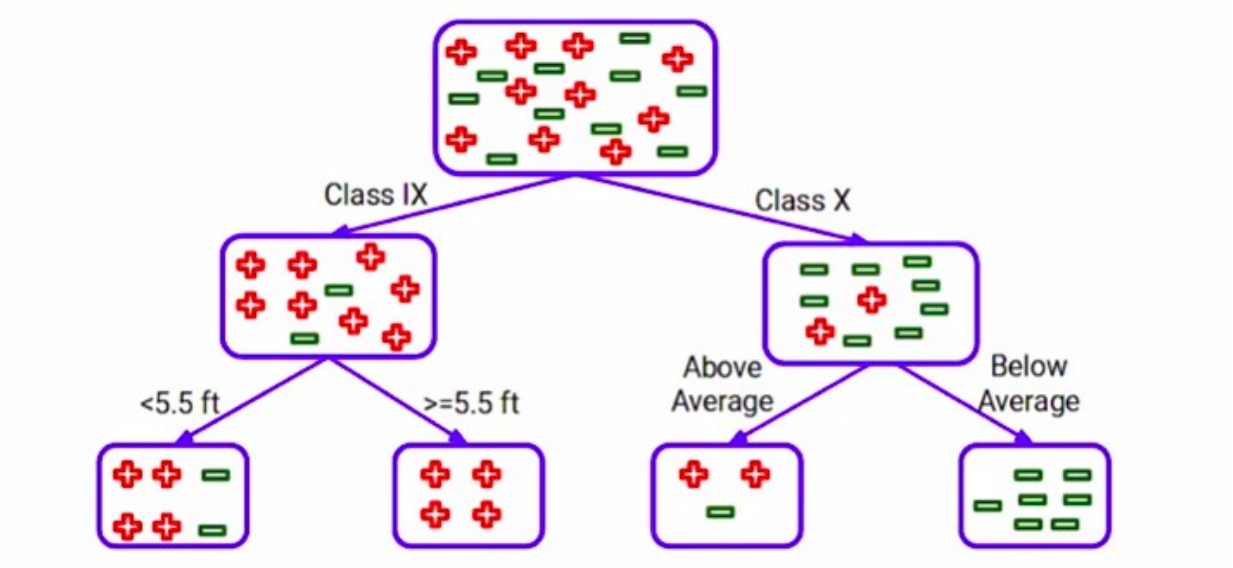


What is the objective of decision tree ?

It is essentially to have pure nodes

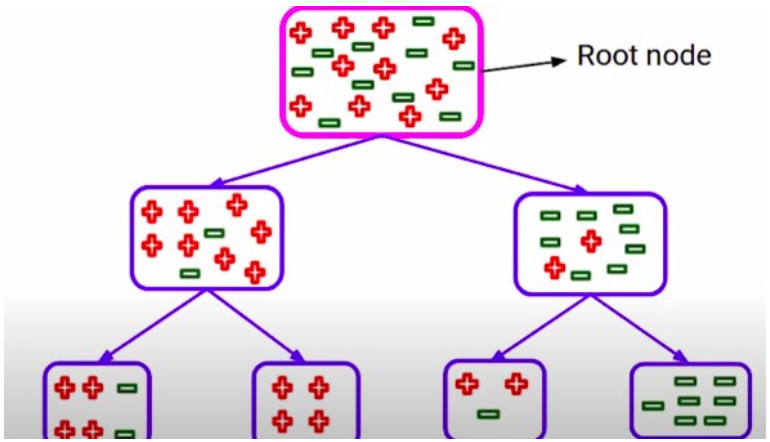
We saw that split on class shows us the pure nodes.hence we chose as first split

But the decision tree does not stop there, infact we can have more splits.

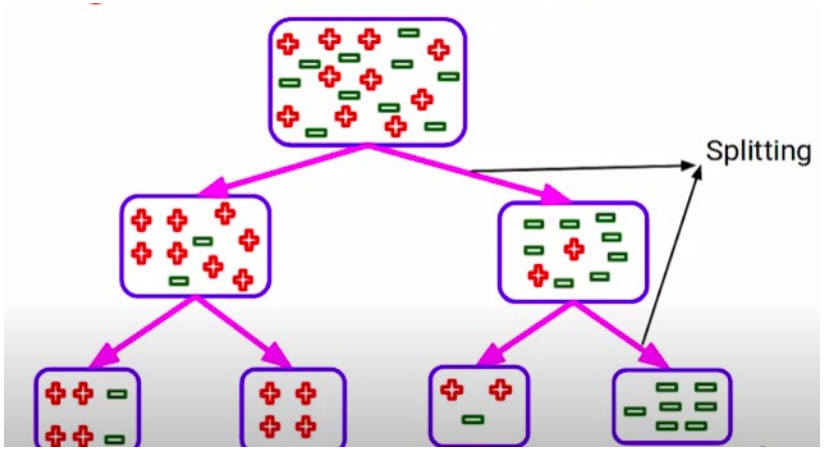


Terminologies related to Decision Tree.

1) Root node : A root node represents the entire population for the data we have

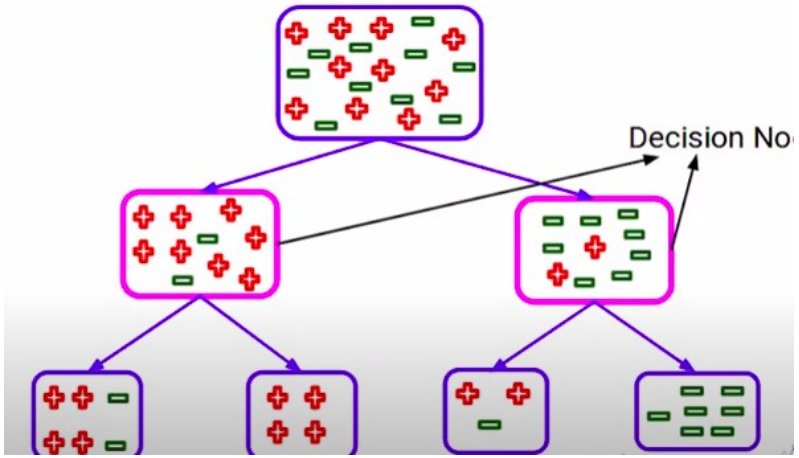


2) **Splitting**: Splitting is the process of dividing the nodes into 2 or more sub nodes.



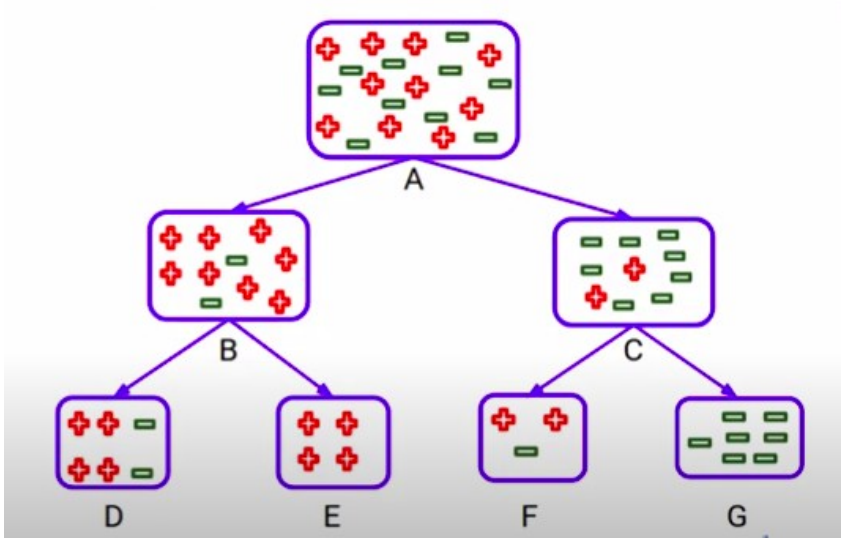
Terminologies related to Decision Tree.

Decision Node: When a subnode is further divided into further even more subnodes then the initial subnode will be called as a decision tree.



Leaf Node/Terminal Node: The nodes which do not split further are called as the leaf or the terminal nodes.

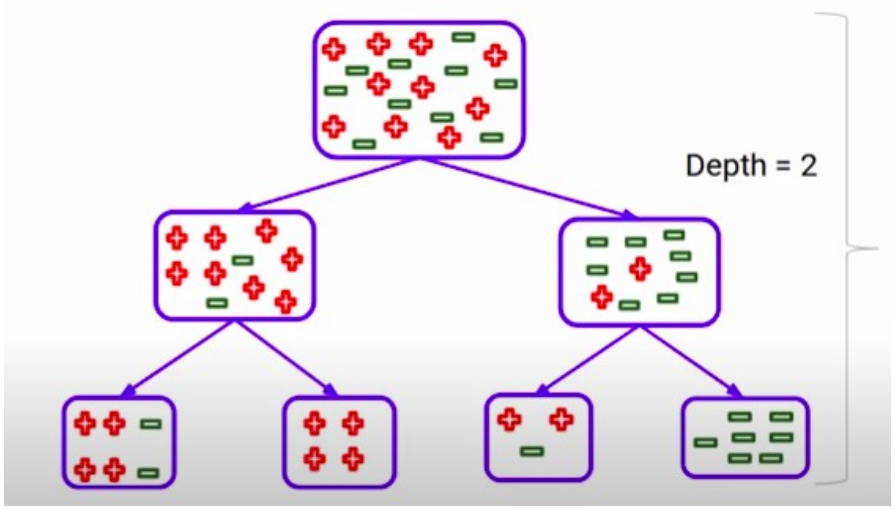
Parent Child Node

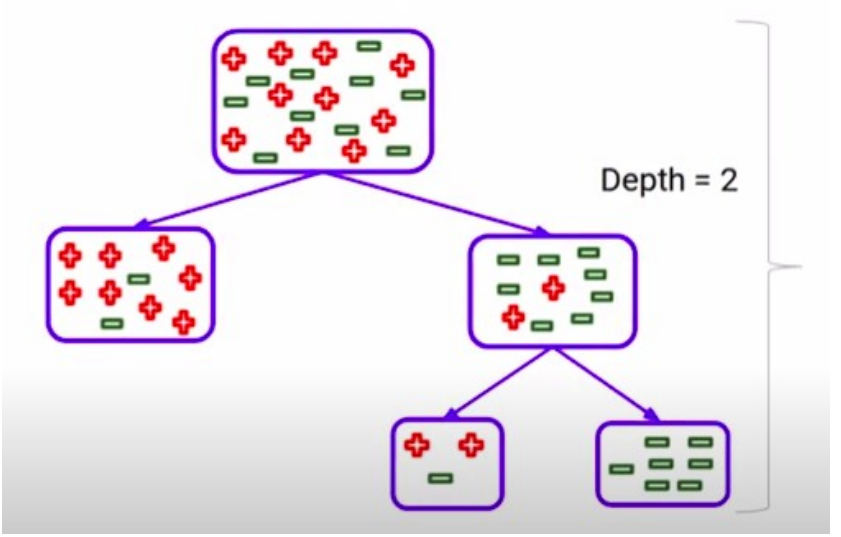


A is the parent node of B & C (here B & C are child node for Parent A)

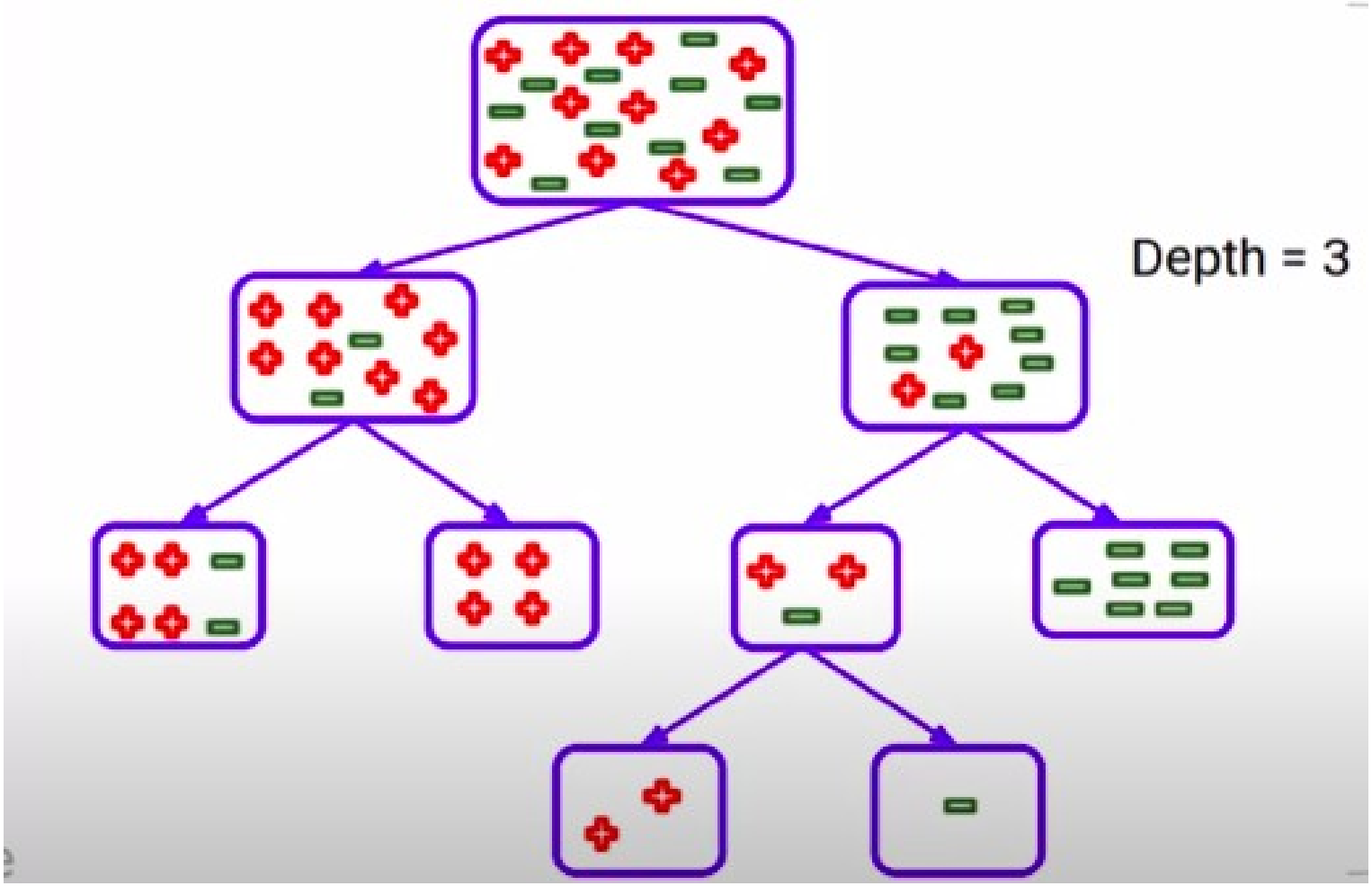
Similarly if we see left section B is the parent node and (D&E) are the child node to parent B and so on....

Depth of the Tree: It is the length of the longest path from the root node till the leaf node.





Qs



How to select best split points

1) Decision Tree splits the nodes on all available variable

2) Select the split which results in the most homogeneous sub nodes.

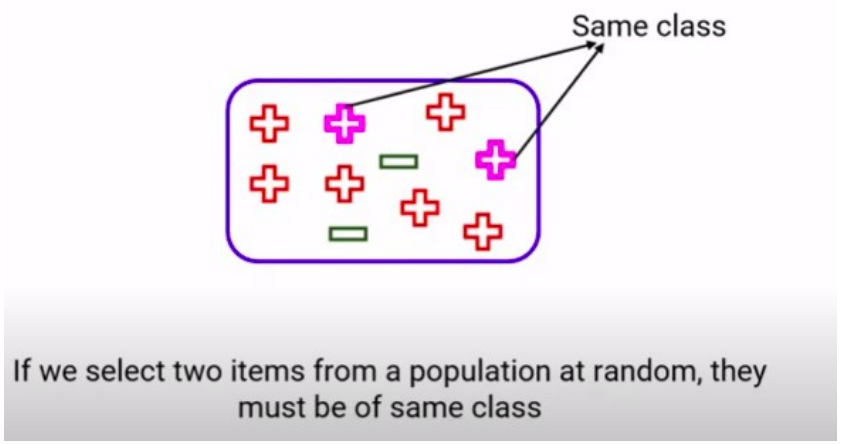
Homogeneous means – having similar behaviour for the problem we have

There are multiple algorithm available to decide the best split for the problem

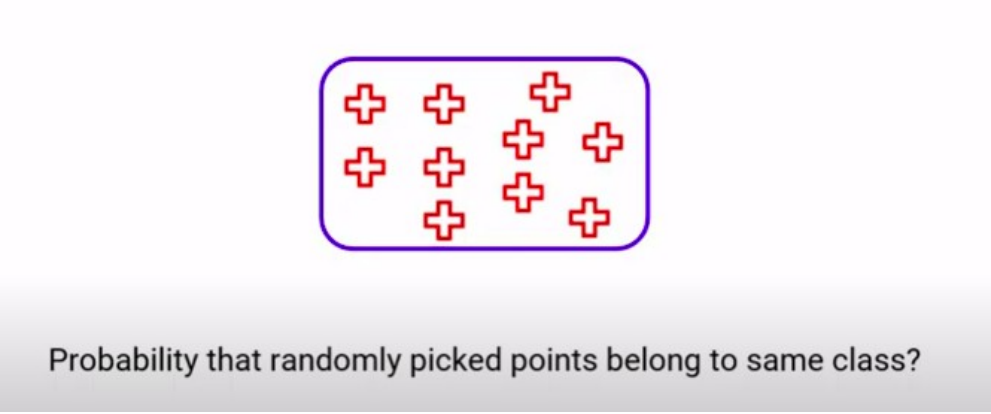
1) Gini Impurity.: It measures the impurity of the nodes and is calculated as

    1-Gini

Eg: Lets assume we have a node



How to select best split points



The probablity will be 1 since all the points here belongs to the same class.

No matter which 2 points you pick they will always point to that 1 class hence the probablity will always be 1.

This is what we want to achieve using Gini.

Gini value will be between 0 to 1 , lower the gini impurity higher the homogeneity of the nodes.

Works only with categorical targets (values) , does not work on continous values. Work for binary splits (YES NO)

Steps

- Calculate the gini impurity for sub-nodes :

$$Gini\ Impurity = 1 - Gini$$

- Gini = Sum of square of probabilities for each class/category

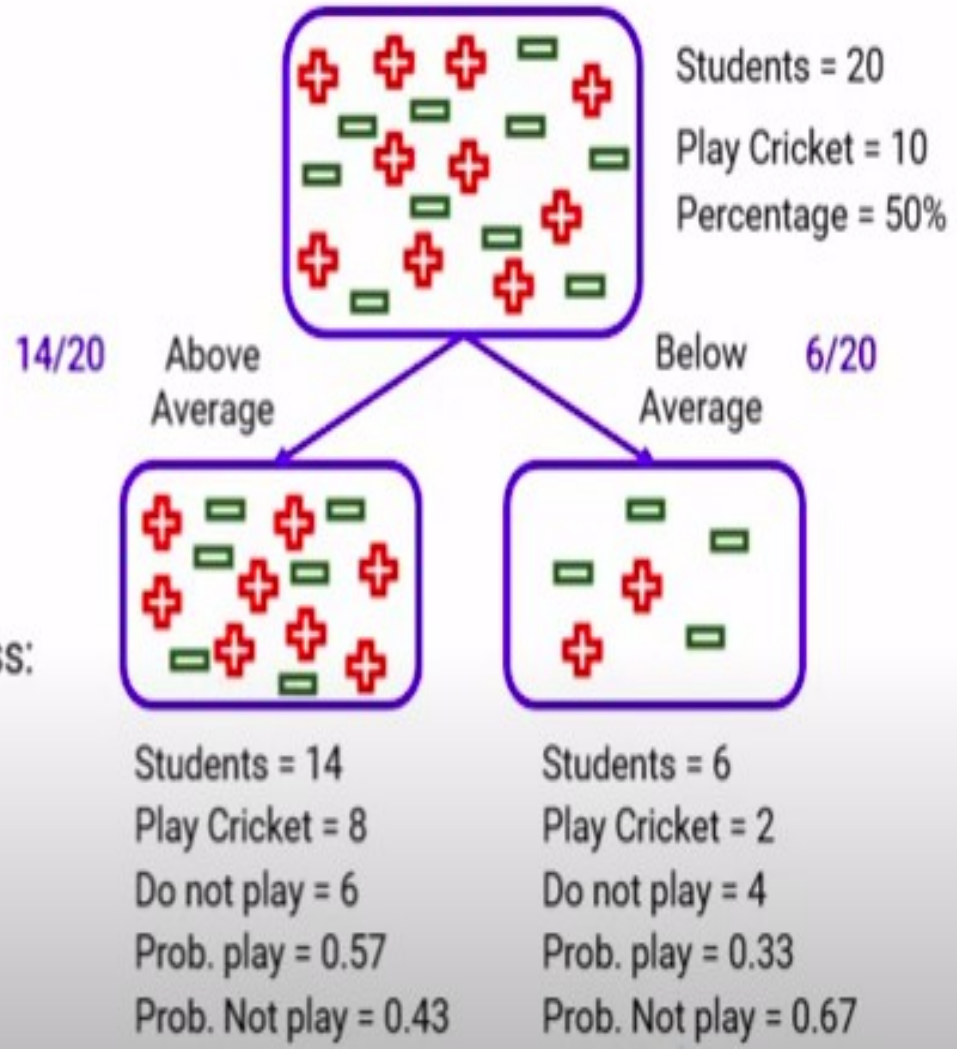
$$Gini = (p_1^2 + p_2^2 + p_3^2 + \ldots + p_n^2)$$

- To calculate the gini impurity for split, take weighted gini impurity of both sub-nodes of that split

Example

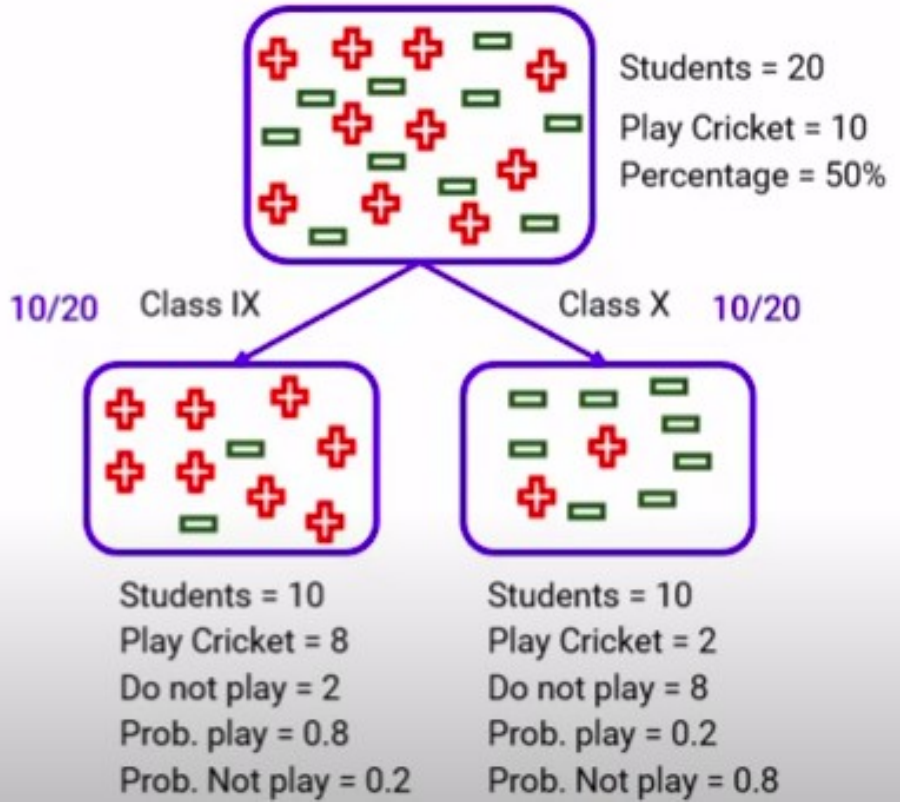
Split on Performance in Class

- Gini Impurity: sub-node Above Average:
  1 - [(0.57)*(0.57) + (0.43)*(0.43)] = 0.49
- Gini Impurity: sub-node Below Average:
  1 - [(0.33)*(0.33) + (0.67)*(0.67)] = 0.44
- Weighted Gini Impurity: Performance in Class:
  (14/20)*0.49 + (6/20)*0.44 = 0.475



Split on Class

- Gini Impurity: sub-node Class IX:
  1 - [(0.8)*(0.8) + (0.2)*(0.2)] = 0.32
- Gini Impurity: sub-node Class X:
  1 - [(0.2)*(0.2) + (0.8)*(0.8)] = 0.32
- Weighted Gini Impurity: Class:
  (10/20)*0.32 + (10/20)*0.32 = 0.32



Split	Weighted Gini Impurity
Performance in Class	0.475
Class	0.32

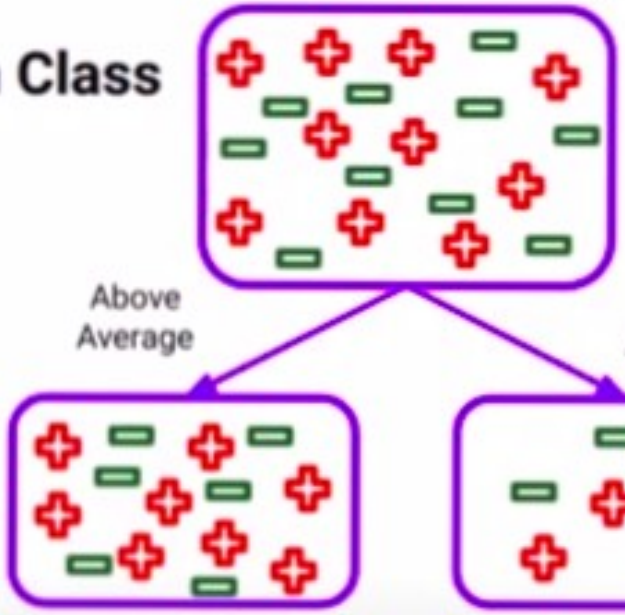
Chi-Square

It calculates the statistical significance of difference between child and parent node.

It is measured as sum of squared standardized difference between actual and expected frequencies of target variable of each node.

$$Chi\text{-}Square = \sqrt{[(Actual - Expected)^2 / Expected]}$$

Split on Performance in Class



Students = 20
Play Cricket = 10
Do not play = 10
Percentage play = 50%
Percentage not play = 50%

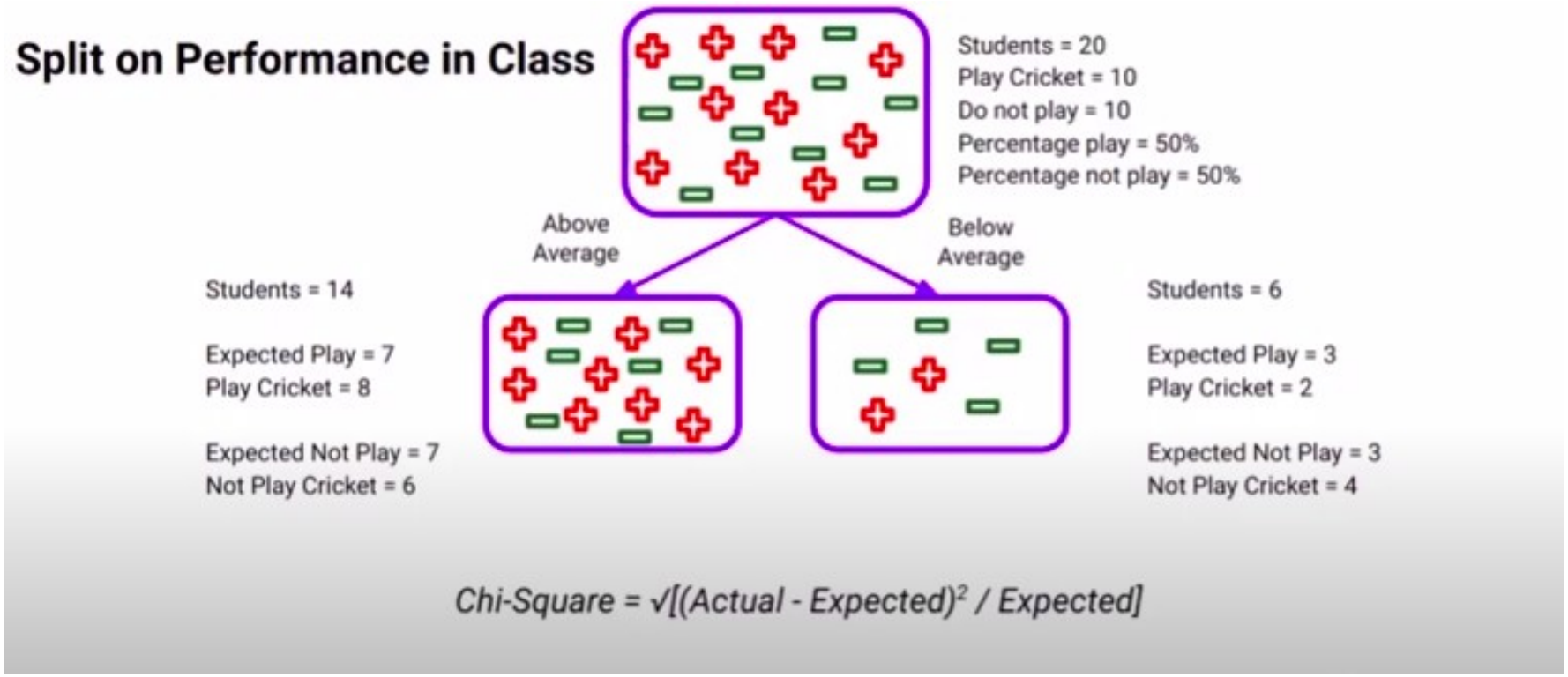
Above Average

Below Average

Students = 14

Expected Play = 7
Play Cricket = 8

Expected Not Play = 7
Not Play Cricket = 6



Higher the chi square value more will be the purity of the node after the split.

Works only with categorical values

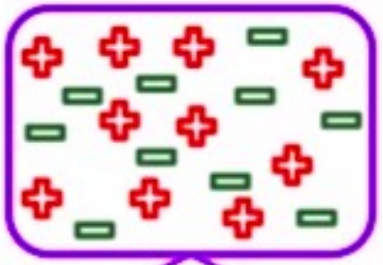
Steps for Chi square

- Calculate the expected values for each class for every child nodes
- Calculate the Chi-Square for every child node

$$\textit{Chi-Square} = \sqrt{[(\textit{Actual} - \textit{Expected})^2 / \textit{Expected}]}$$

- Calculate Chi-Square for split using sum of Chi-Square of each child node of that split

Split on Performance in Class



Students = 20

Play Cricket = 10

Percentage = 50%

Above Average

Below Average

Students = 14

Expected Play = 7
Play Cricket = 8

Expected Not Play = 7
Not Play Cricket = 6

Students = 6

Expected Play = 3
Play Cricket = 2

Expected Not Play = 3
Not Play Cricket = 4

Node	Actual Play	Actual Not Play	Expected Play	Expected Not Play	Deviation Play	Deviation Not Play	Chi-Square (Play)	Chi-Square (Not Play)
Above Average	8	6	7	7	1	-1		
Below Average	2	4	3	3	-1	1		

Split on Performance in Class

Students = 20

Play Cricket = 10

Percentage = 50%

Above Average

Below Average

Students = 14

Expected Play = 7
Play Cricket = 8

Expected Not Play = 7
Not Play Cricket = 6

Students = 6

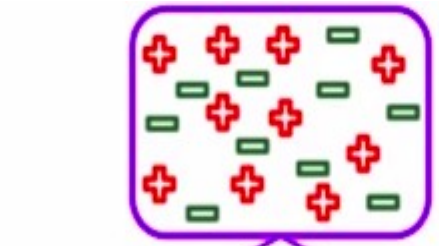
Expected Play = 3
Play Cricket = 2

Expected Not Play = 3
Not Play Cricket = 4

Node	Actual Play	Actual Not Play	Expected Play	Expected Not Play	Deviation Play	Deviation Not Play	Chi-Square (Play)	Chi-Square (Not Play)
Above Average	8	6	7	7	1	-1	0.38	0.38
Below Average	2	4	3	3	-1	1	0.58	0.58

Total = 0.38 + 0.38 + 0.58 + 0.58 = 1.92

Split on Class



Students = 20

Play Cricket = 10

Percentage = 50%

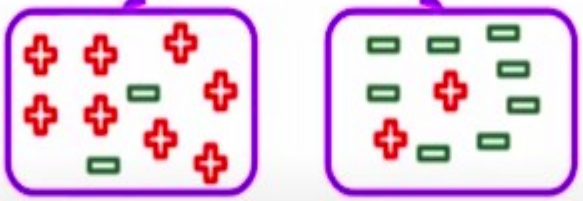
Total = 1.34 + 1.34 + 1.34 + 1.34 = 5.36

Class IX

Students = 10

Expected Play = 5
Play Cricket = 8

Expected Not Play = 5
Not Play Cricket = 2



Class X

Students = 10

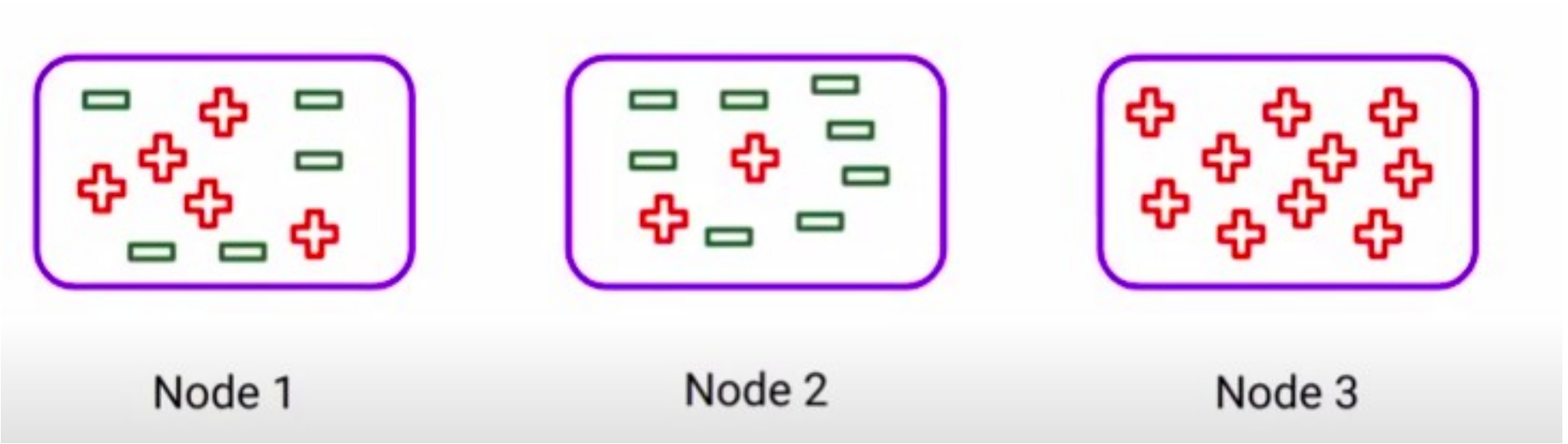
Expected Play = 5
Play Cricket = 2

Expected Not Play = 5
Not Play Cricket = 8

Node	Actual Play	Actual Not Play	Expected Play	Expected Not Play	Deviation Play	Deviation Not Play	Chi-Square (Play)	Chi-Square (Not Play)
IX	8	2	5	5	3	-3	1.34	1.34
X	2	8	5	5	-3	3	1.34	1.34

Split	Chi-Square
Performance in Class	1.92
Class	5.36

Information Gain

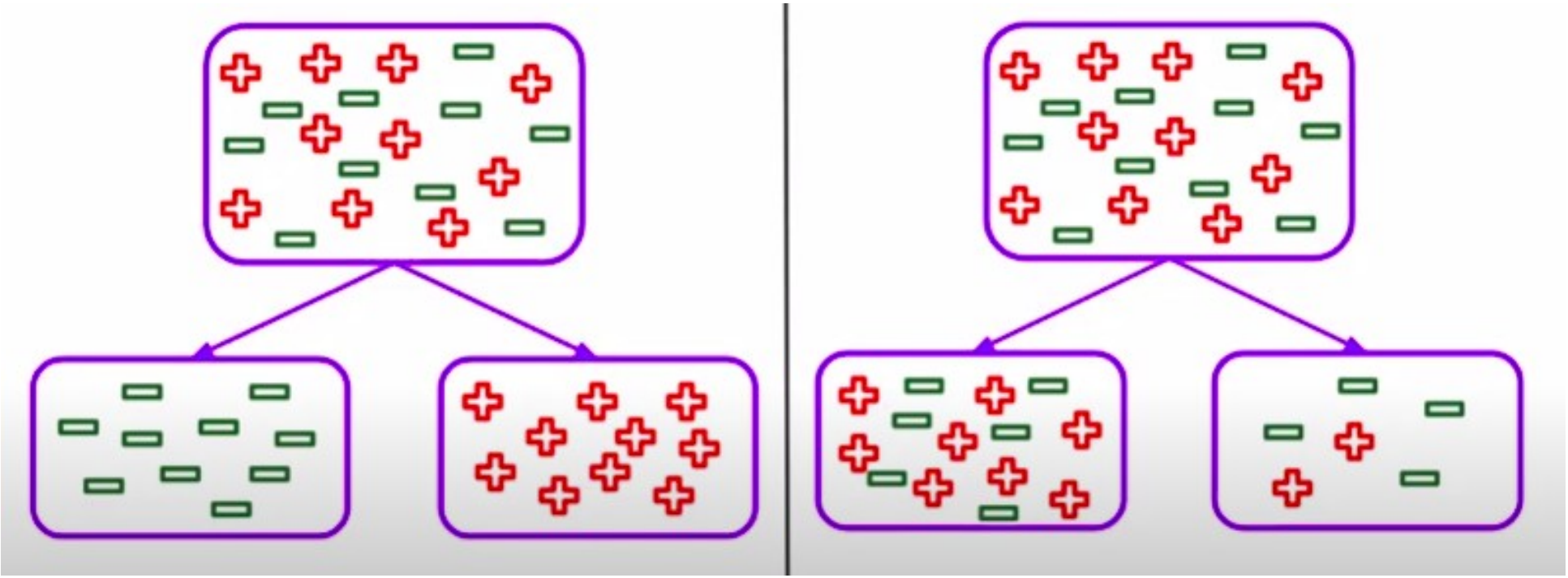


Consider the above nodes.. You can see Node3 is the pure node as it only contains 1 class.

Node2 is less pure & Node 1 is the most impure node

As impurity of node increases we need more info.

More impure nodes requires more info.



Split on right is giving less information gain.
Higher info gain leads to more homogenous or pure node.

Information Gain = 1- Entropy

Entropy

$$- p_1 * \log_2 p_1 - p_2 * \log_2 p_2 - p_3 * \log_2 p_3 - \ldots - p_n * \log_2 p_n$$

% Play = 0.50
% Not play = 0.50

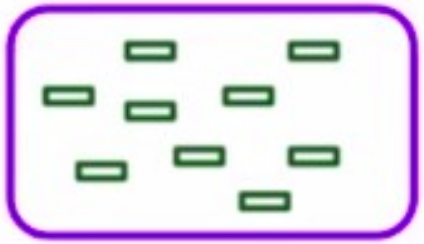
Entropy = - (0.5) * log2(0.5) - (0.5) * log2(0.5)

E = 1

Where p is the percentage of each class in the node.

Entropy

$$- p_1 * \log_2 p_1 - p_2 * \log_2 p_2 - p_3 * \log_2 p_3 - ..... - p_n * \log_2 p_n$$



% Play = 0

% Not play = 1

Entropy = - (0) * log2(0) - (1) * log2(1)

Here E will be 0

Lower entropy ---> more pure nodes
Higher entropy ---> impure nodes.
Works with categorical data (value)

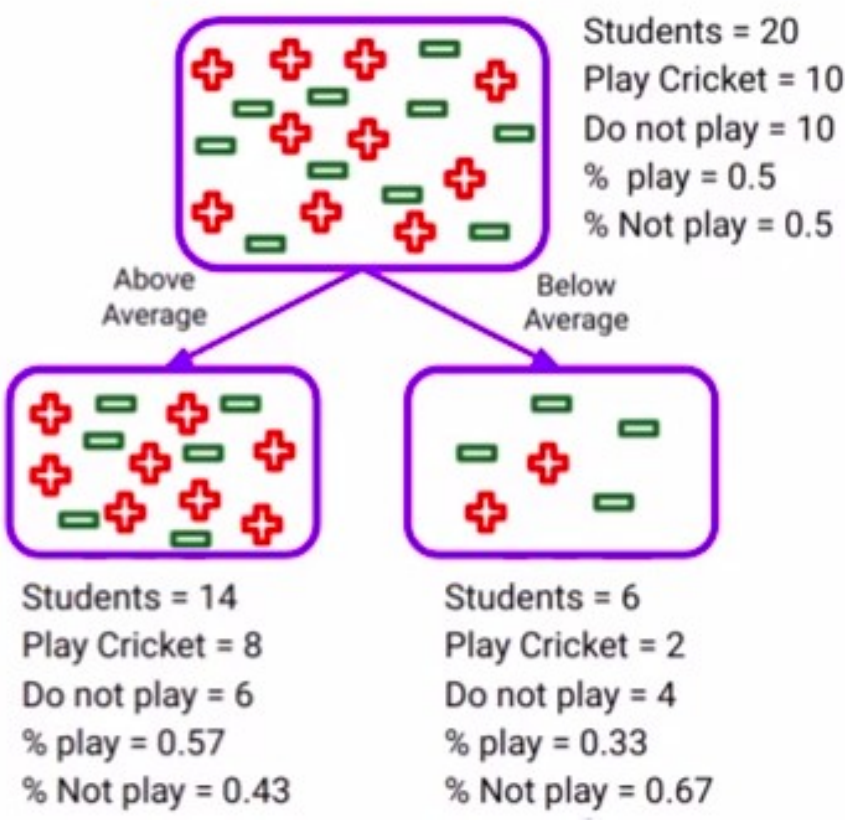
Steps to calculate Entropy for a split

- Calculate the entropy of the parent node
- Calculate the entropy of each child node
- Calculate the weighted average entropy of the split

Weight of node = no of sample in node / total sample in parent node.

Split on Performance in Class

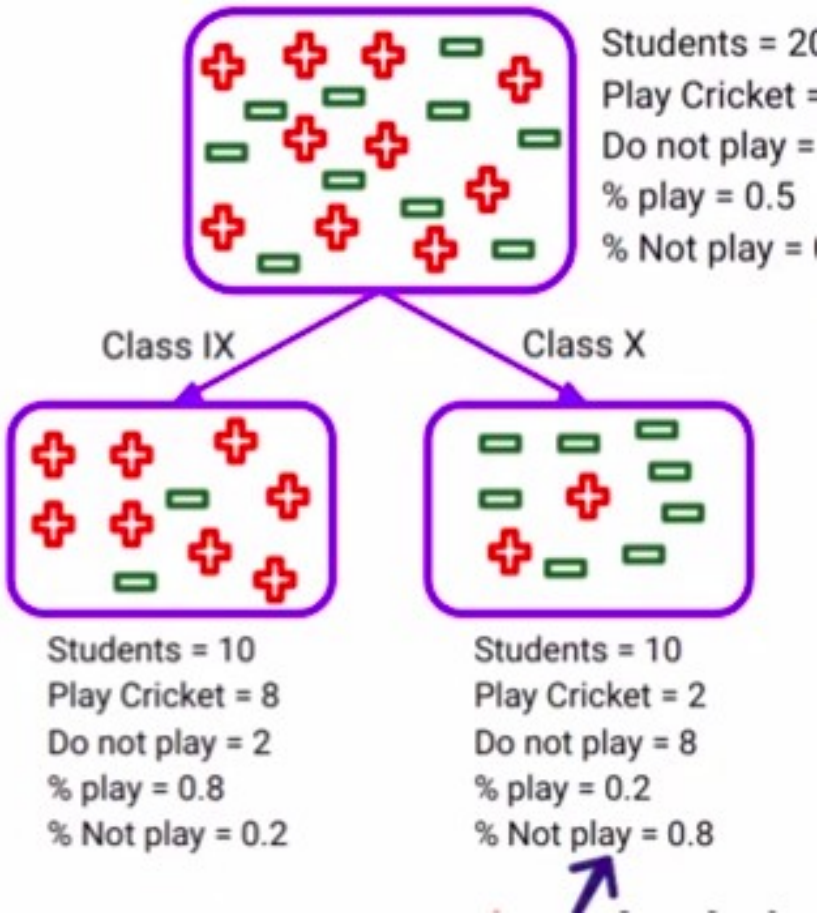
- Entropy for Parent node:
  $-(0.5)*\log_2(0.5) -(0.5)*\log_2(0.5) = 1$
- Entropy for sub-node Above Average:
  $-(0.57)*\log_2(0.57) -(0.43)*\log_2(0.43) = 0.98$
- Entropy for sub-node Below Average:
  $-(0.33)*\log_2(0.33) -(0.67)*\log_2(0.67) = 0.91$



Weighted Entropy: Performance in Class:
$(14/20)*0.98 + (6/20)*0.91 = 0.959$

Split on Class

- Entropy for Parent node:
  $-(0.5)*\log_2(0,5) -(0.5)*\log_2(0.5) = 1$
- Entropy for sub-node Class IX:
  $-(0.8)*\log_2(0.8) -(0.2)*\log_2(0.2) = 0.722$
- Entropy for sub-node Class X:
  $-(0.2)*\log_2(0.2) -(0.8)*\log_2(0.8) = 0.722$
- Weighted Entropy: Class:
  $(10/20)*0.722 + (10/20)*0.722 = 0.722$



Split	Entropy	Information Gain
Performance in Class	0.959	0.041
Class	0.722	0.278

Reduction in Variance

$$\text{Variance} = \Sigma [(X - \mu)^2] / n$$

X --> sample
U --> mean
N ---> number of sample.

Lower value --> pure nodes

Higher value --> impure nodes

Consider example below:

- Plays Cricket = 1
- Do not play Cricket = 0

Above Average node:

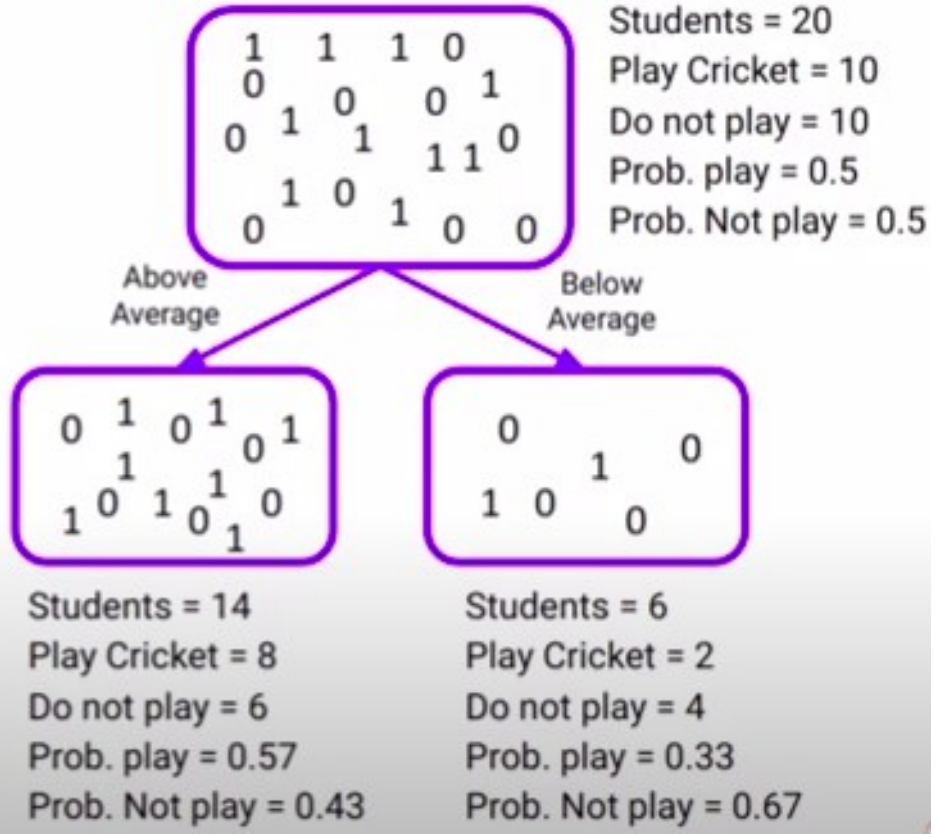
- Mean = (8*1 + 6*0) / 14 = 0.57
- Variance =
  $[8*(1-0.57)^2 + 6*(0-0.57)^2] / 14 = 0.245$

Below Average node:

- Mean = (2*1 + 4*0) / 6 = 0.33
- Variance =
  $[2*(1-0.33)^2 + 4*(0-0.33)^2] / 6 = 0.222$

Variance: Performance in Class:

(14/20)*0.245 + (6/20)*0.222 = 0.238



Class IX node:

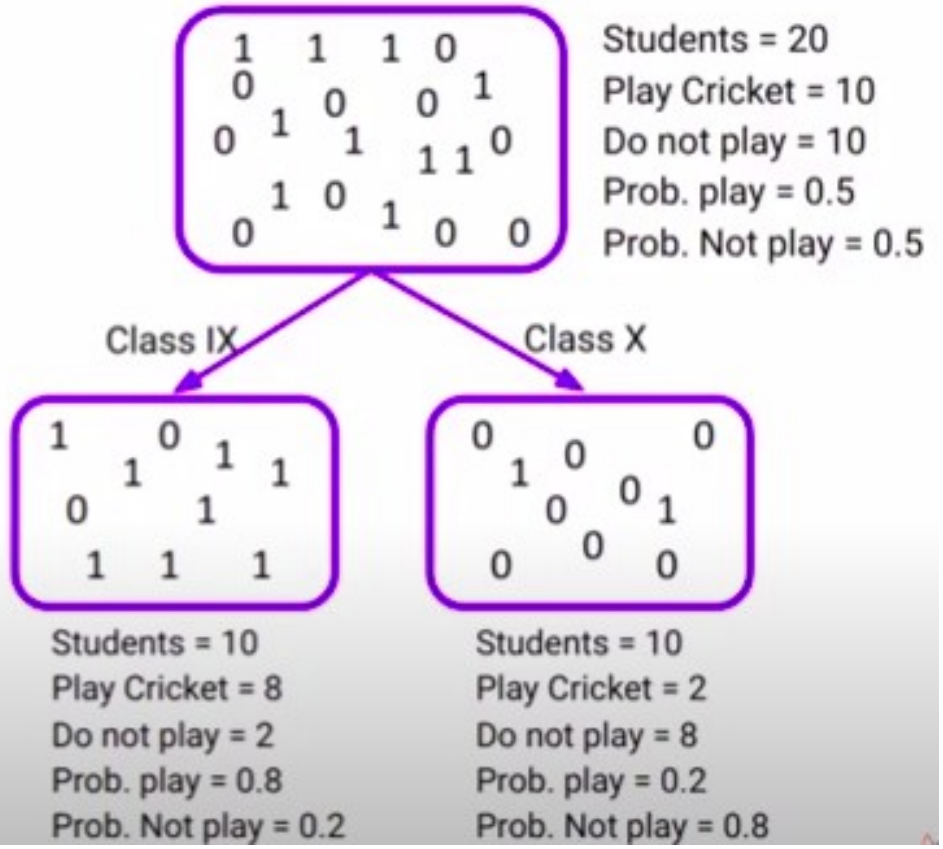
- Mean = (8*1 + 2*0) / 10 = 0.8
- Variance = $[8*(1-0.8)^2 + 2*(0-0.8)^2] / 10 = 0.16$

Class X node:

- Mean = (2*1 + 8*0) / 10 = 0.2
- Variance = $[2*(1-0.2)^2 + 8*(0-0.2)^2] / 10 = 0.16$

Variance: Class:

(10/20)*0.16 + (10/20)*0.16 = 0.16



Split	Variance
Performance in Class	0.238
Class	0.16