

Decision Trees

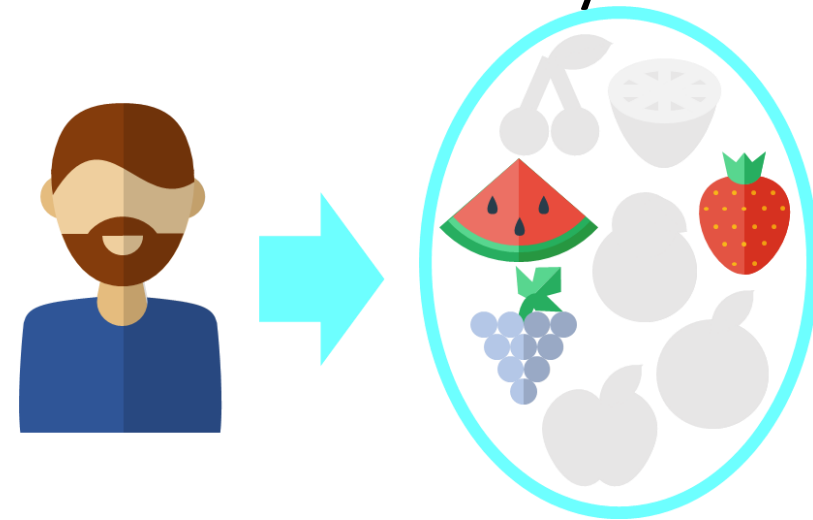
CS771: Introduction to Machine Learning

Purushottam Kar

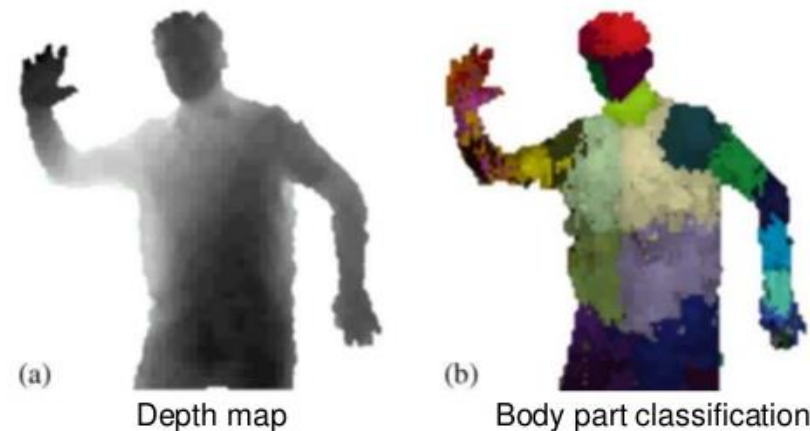
Decision Trees

- Can be seen as a *faster* way to do kNN
- Can handle non-numeric features too!
- Very popular in ML – classification, recsys
- Extremely fast at making predictions
- Easy to interpret by humans too
- Model size can be large
- Can give good train perf. but bad test perf. (known as *overfitting*)

Recommendation Systems



Gaming – Kinect for Xbox 360



Decision Trees for Classification

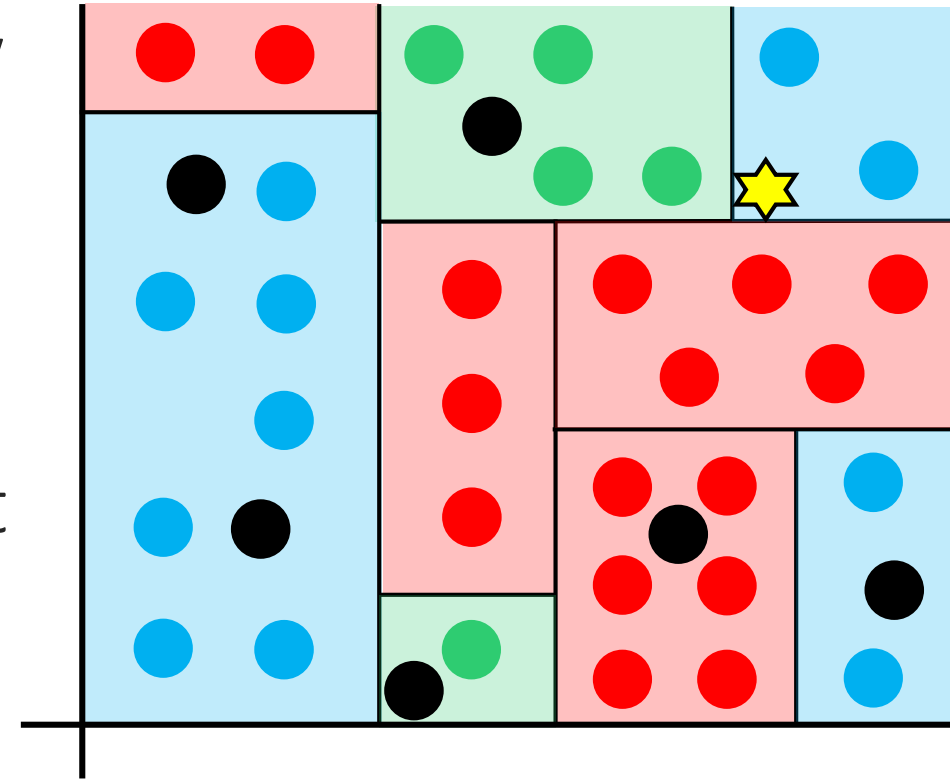
3

Finding the nearest neighbours can be slow

So repeatedly partition feature space \mathbb{R}^d

For test data point, much faster to find out which partition is it in which it lies

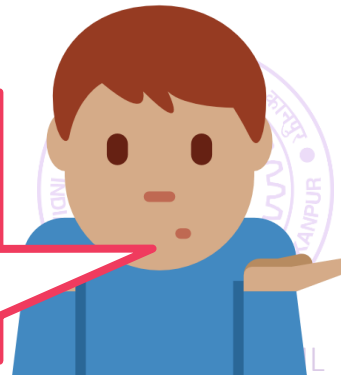
Can consider the other (train) points in that partition as approximate neighbours



However, if your partitions are fine enough, this will happen very rarely and not hurt performance too much!

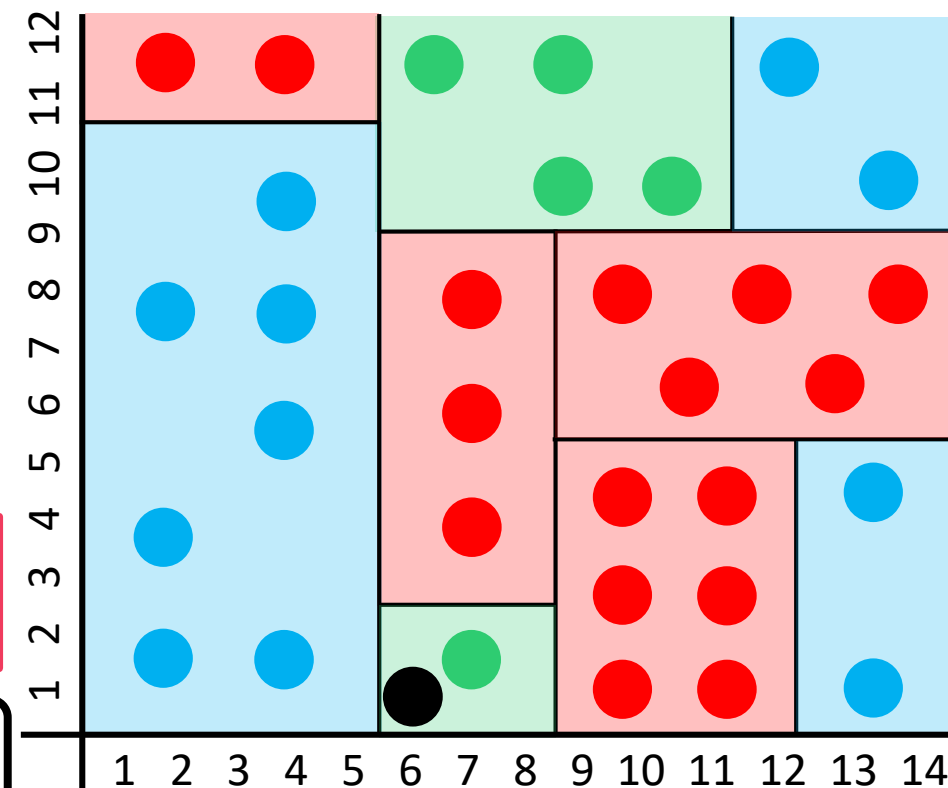
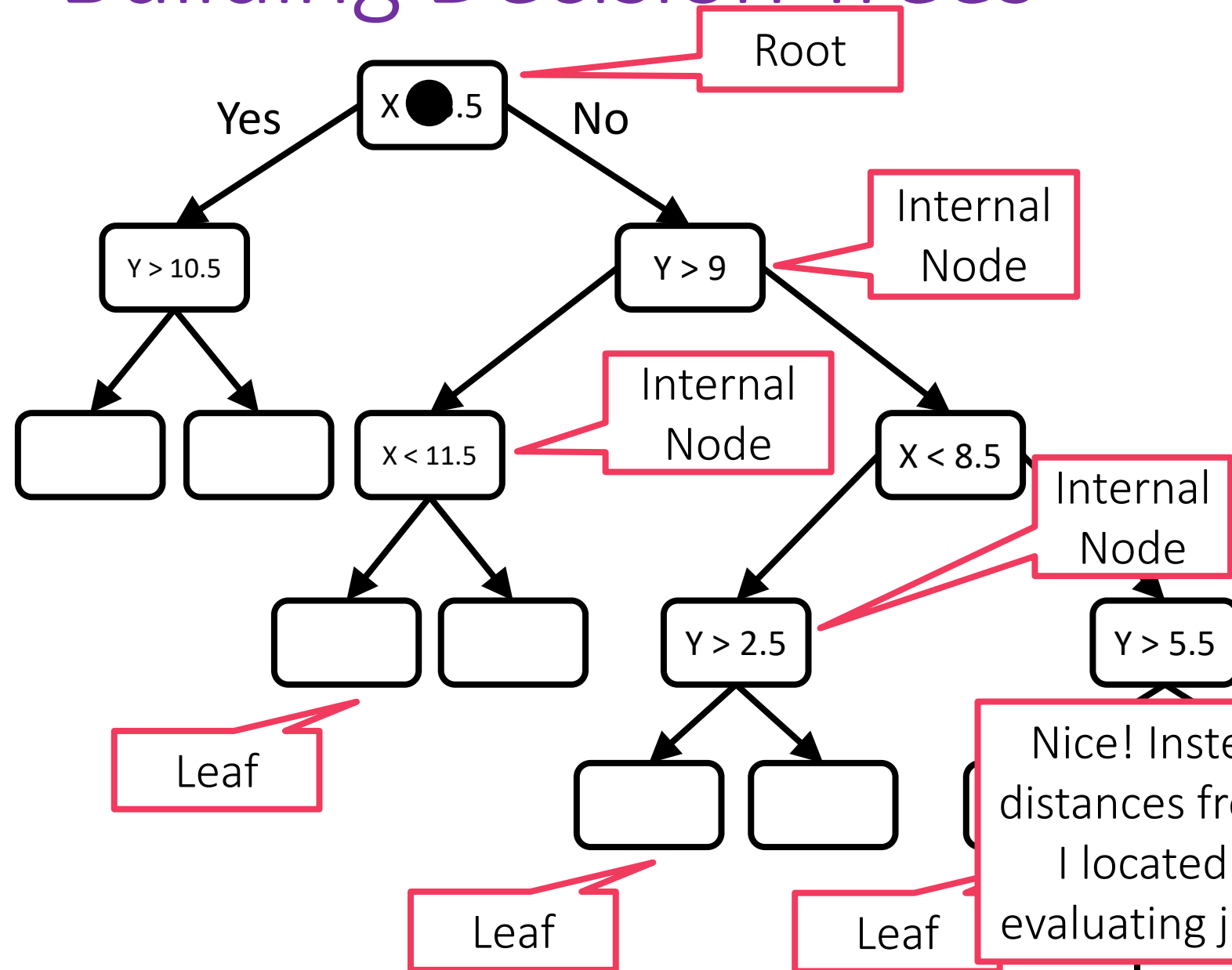
Yes, decision trees may not always give you the exact neighbors for points lying at boundary of a partition

What about points like this?
The nearest neighbours are lying in another partition ☹️

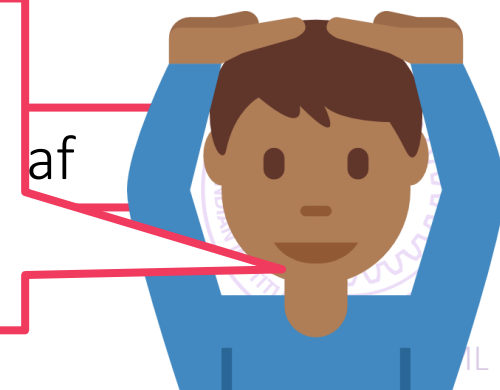


Building Decision Trees

4

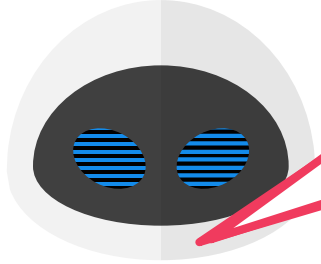


Nice! Instead of calculating distances from 32 data points, I located my partition by evaluating just 4 inequalities!!



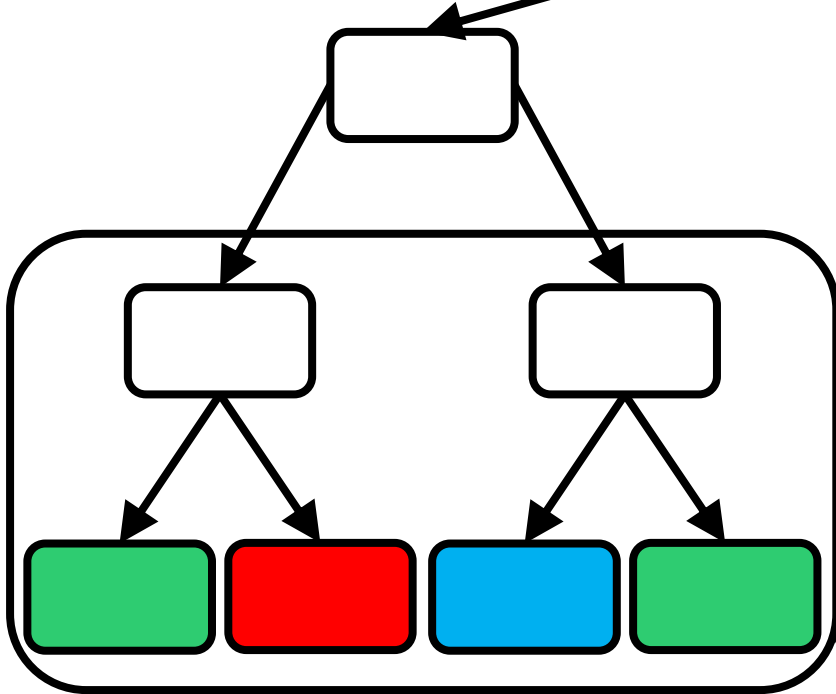
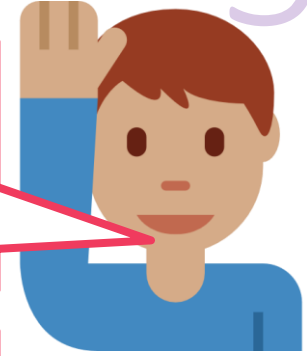
Decision Trees – all shapes and sizes

5

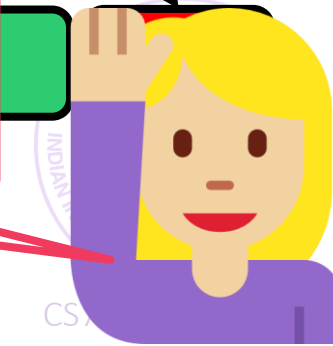
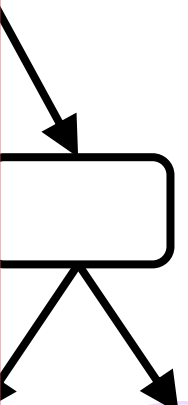
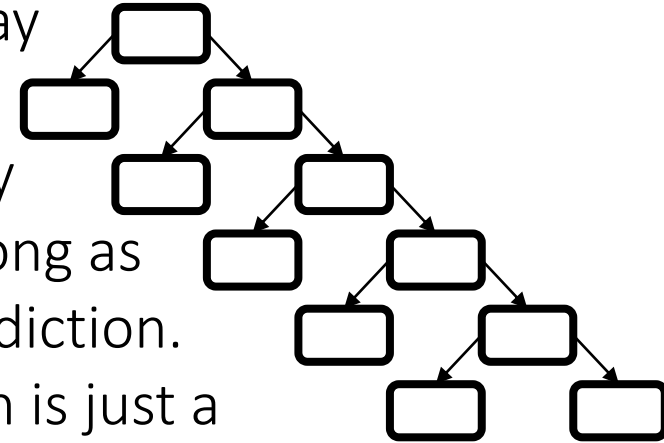


May prune the tree to make it more shallow as well. Also possible to have DT with more than 2 children per internal node

The previous DT was very imbalanced and considered bad in general

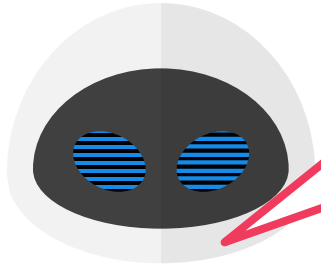


Imbalanced DTs may offer very poor Prediction accuracy as well as take as long as kNN to make a prediction. Imagine a DT which is just a chain of nodes. With n data points, there would be $\mathcal{O}(n)$ chain: some predictions will take $\mathcal{O}(n)$ time ☹️. With a balanced DT, every prediction takes at most $\mathcal{O}(\log n)$ time 😊😊

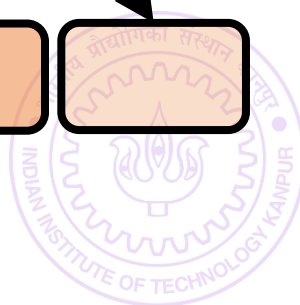
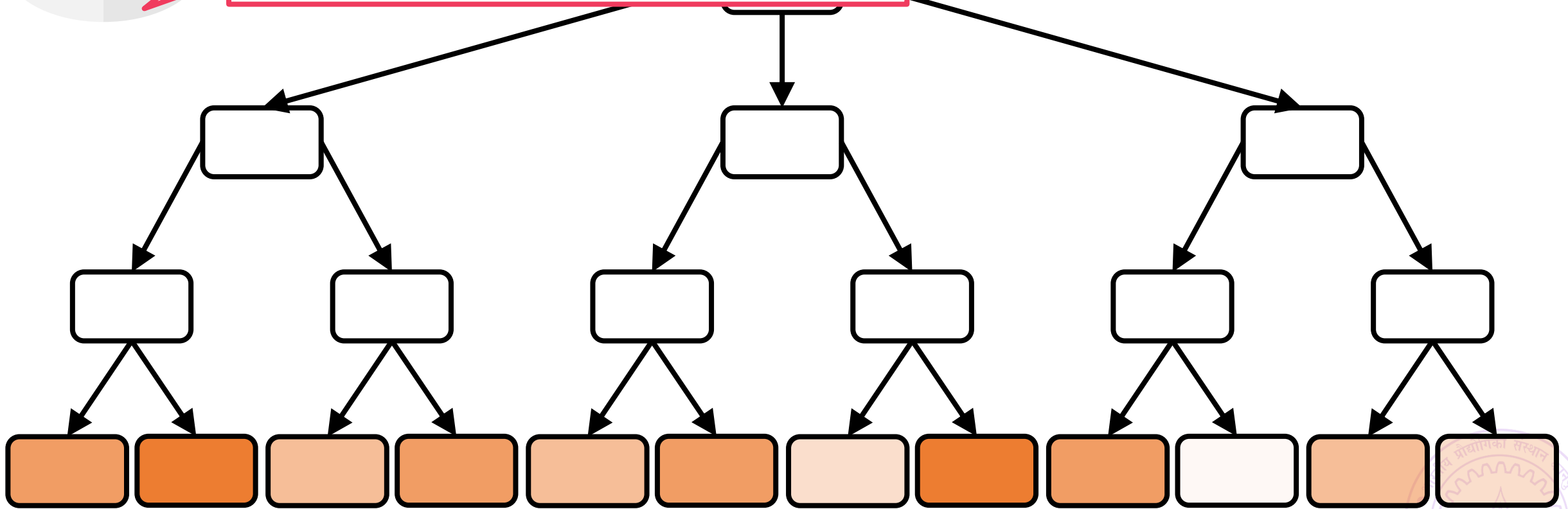


Regression with Decision Trees

6



To perform real valued regression, may simply use average score at a leaf node to predict scores for test data points



How to learn a DT?

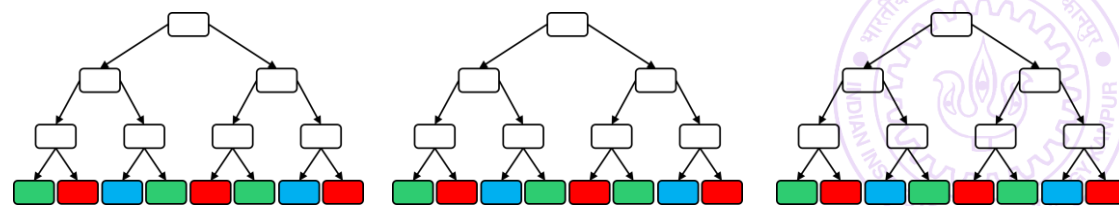
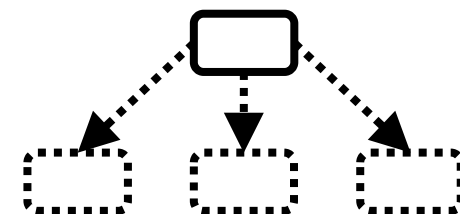
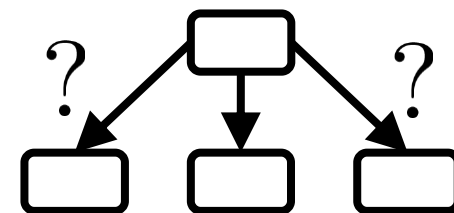
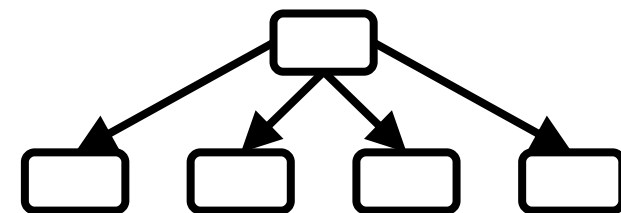
How many children should a node have?

How to send data points to children?

When to stop splitting and make the node a leaf?

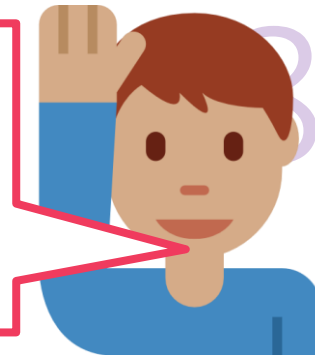
What to do at a leaf?

How many trees to train?



What to do a

Cheapest thing to do would be to store the majority color (label) at a leaf. A slightly more informative (more expensive as well) thing can be to store how many training points of each color (label) reached that leaf



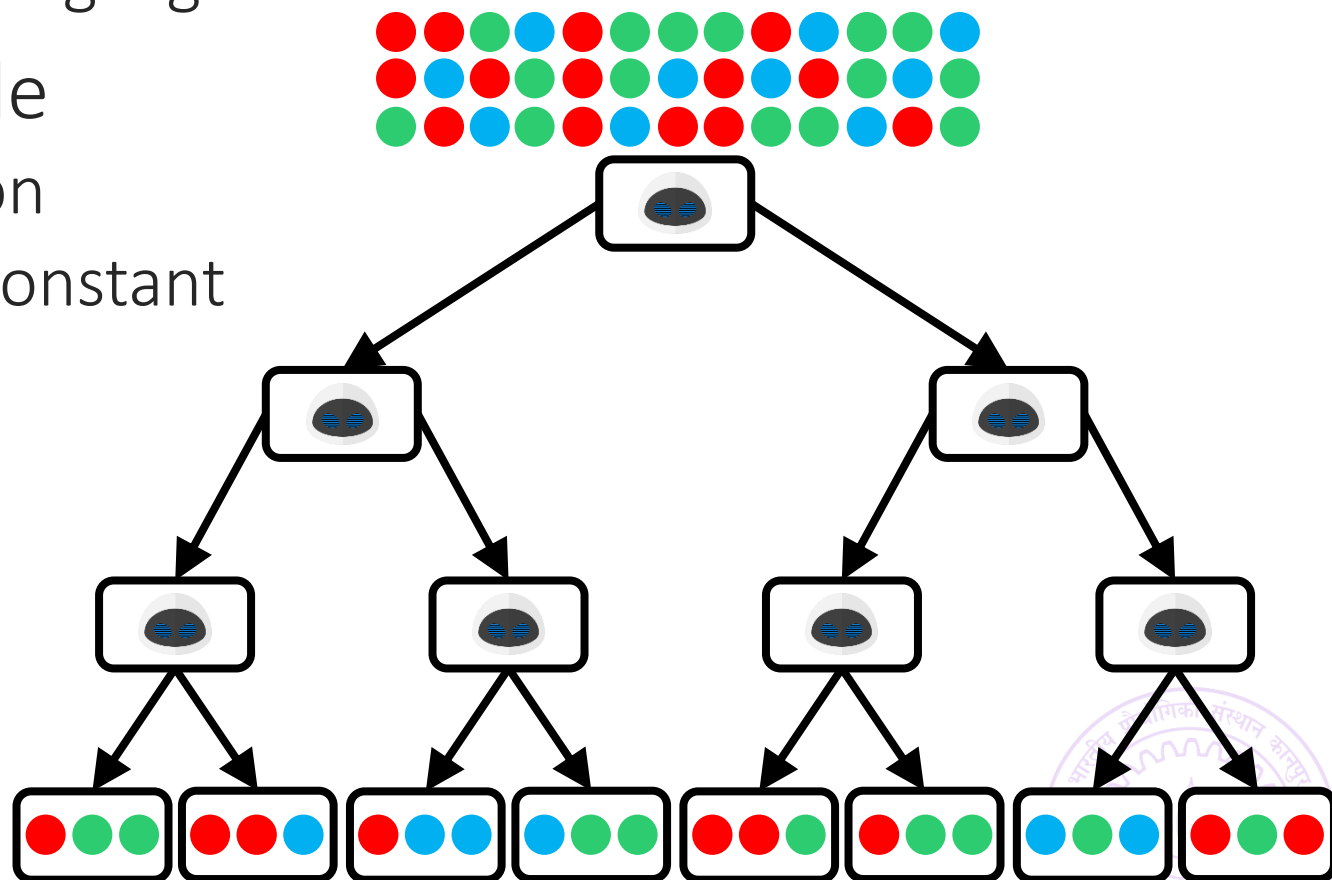
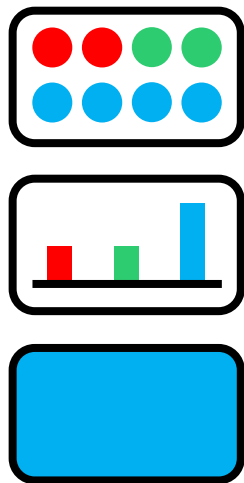
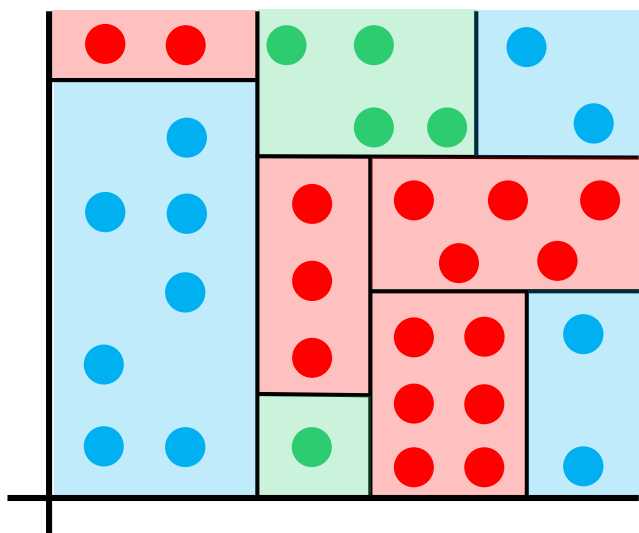
Can take any (complex)

Why not call another machine learning algorithm?

For speed, keep leaf action simple

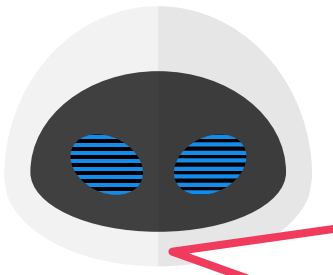
Simplest action – constant prediction

Such a DT will encode a piecewise constant prediction function

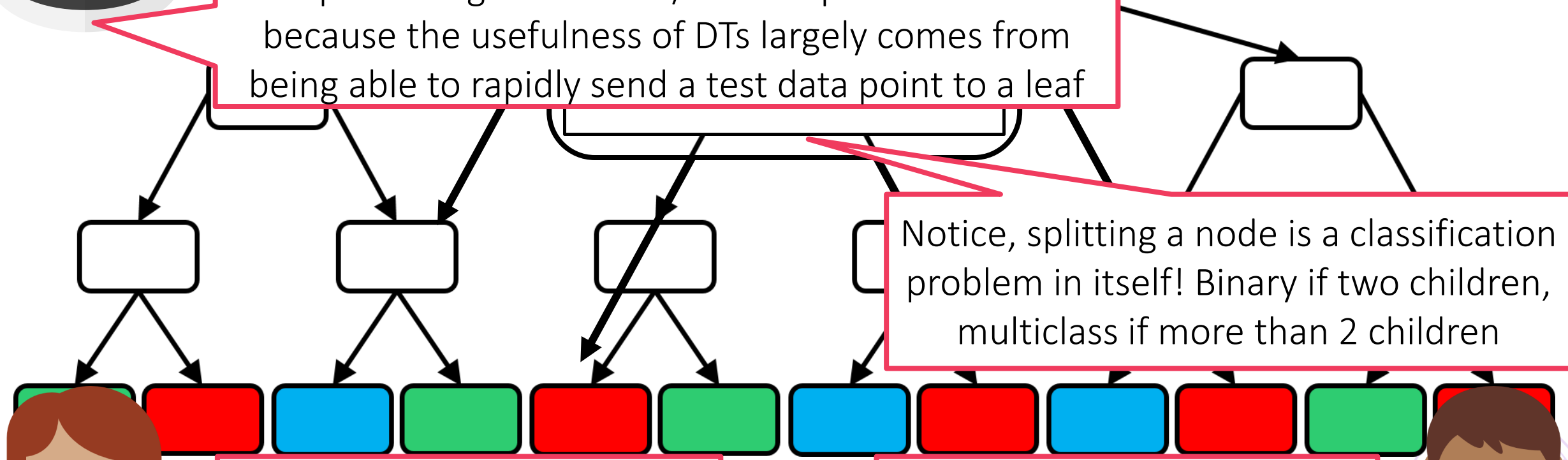


How to split a node into children nodes?

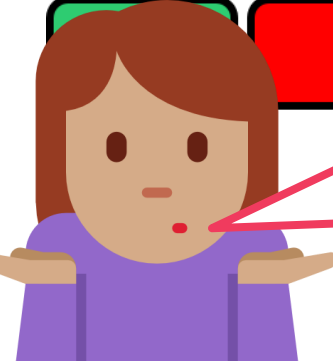
9



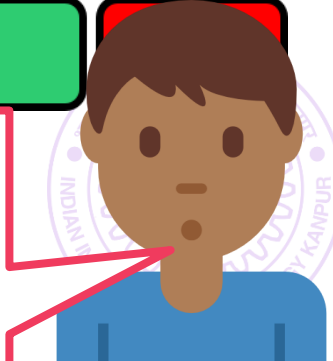
In principle there is no restriction (e.g. can even use a deep net to split a node). However, in practice we use a simple ML algo like linear/LwP to split nodes. This is because the usefulness of DTs largely comes from being able to rapidly send a test data point to a leaf



Notice, splitting a node is a classification problem in itself! Binary if two children, multiclass if more than 2 children



Can we use any classification technique to split a node or are there some restrictions?



Oh! So we are using a simple ML technique such as binary classification to learn a DT!

Splitting a Node – s

Recall, one of the goals of DT

Thus, the splitting algorithm must be super fast otherwise no

ben
Often, making sure that the split is balanced (e.g. roughly half the data points go left and right) is also important to ensure that the tree is balanced. However, Such ensuring balance is often tricky

Various notions of purity exist – entropy and Gini index for classification problems, variance for regression problems

Pure nodes are very convenient. We can make them leaves right away and not have to worry about splitting them 😊

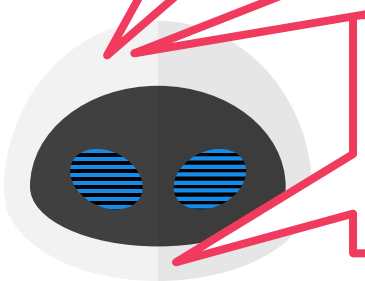
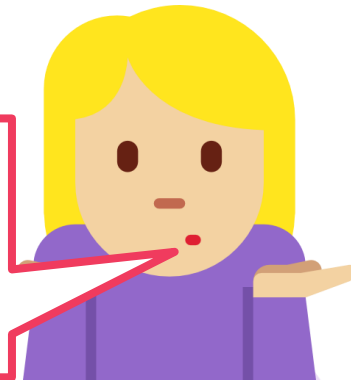
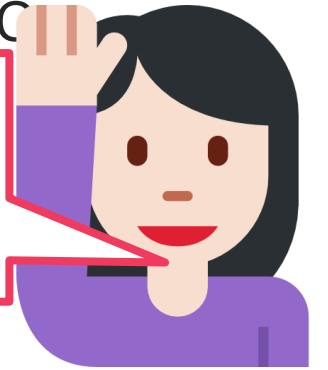
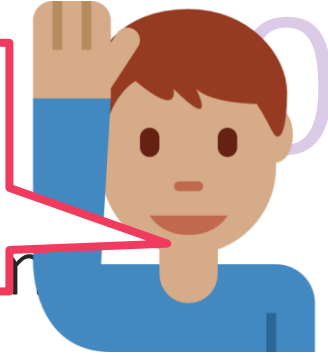
go left, age ≤ 25 go right)

often called *decision stumps*

A child node is completely pure if it contains training data of only one class.

How do I decide whether to use age or gender? Even if using age, how do I decide whether to threshold at 25 or 65?

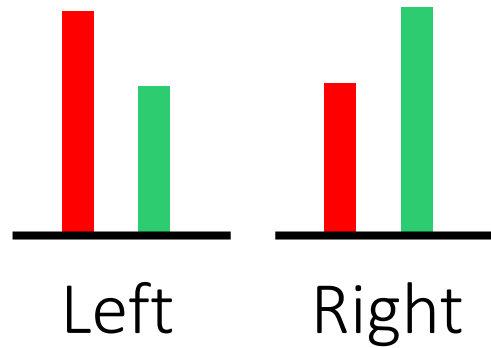
Usually, people go over all available features and all possible thresholds (can be slow if not done cleverly) and choose a feature and a threshold for that feature so that the child nodes that are created are as *pure* as possible



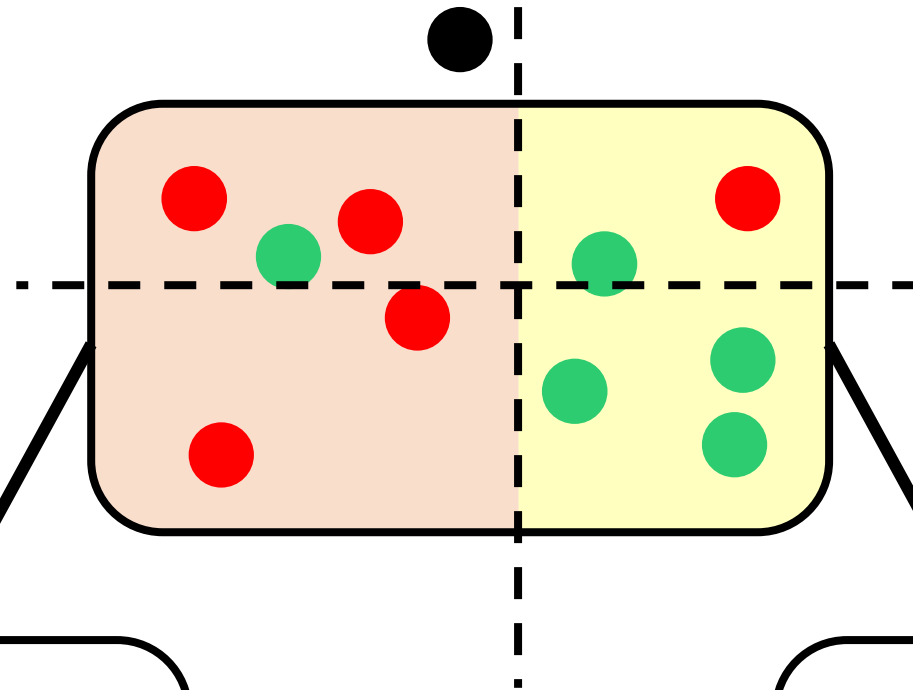
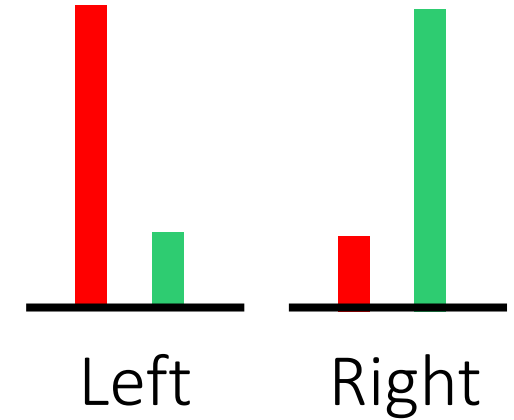
Purifying Decision Stumps

11

Purest Horizontal Split



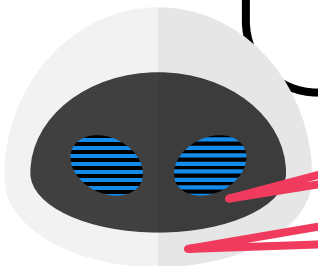
Purest Vertical Split



Search purest horizontal split by going over all possible thresholds and checking

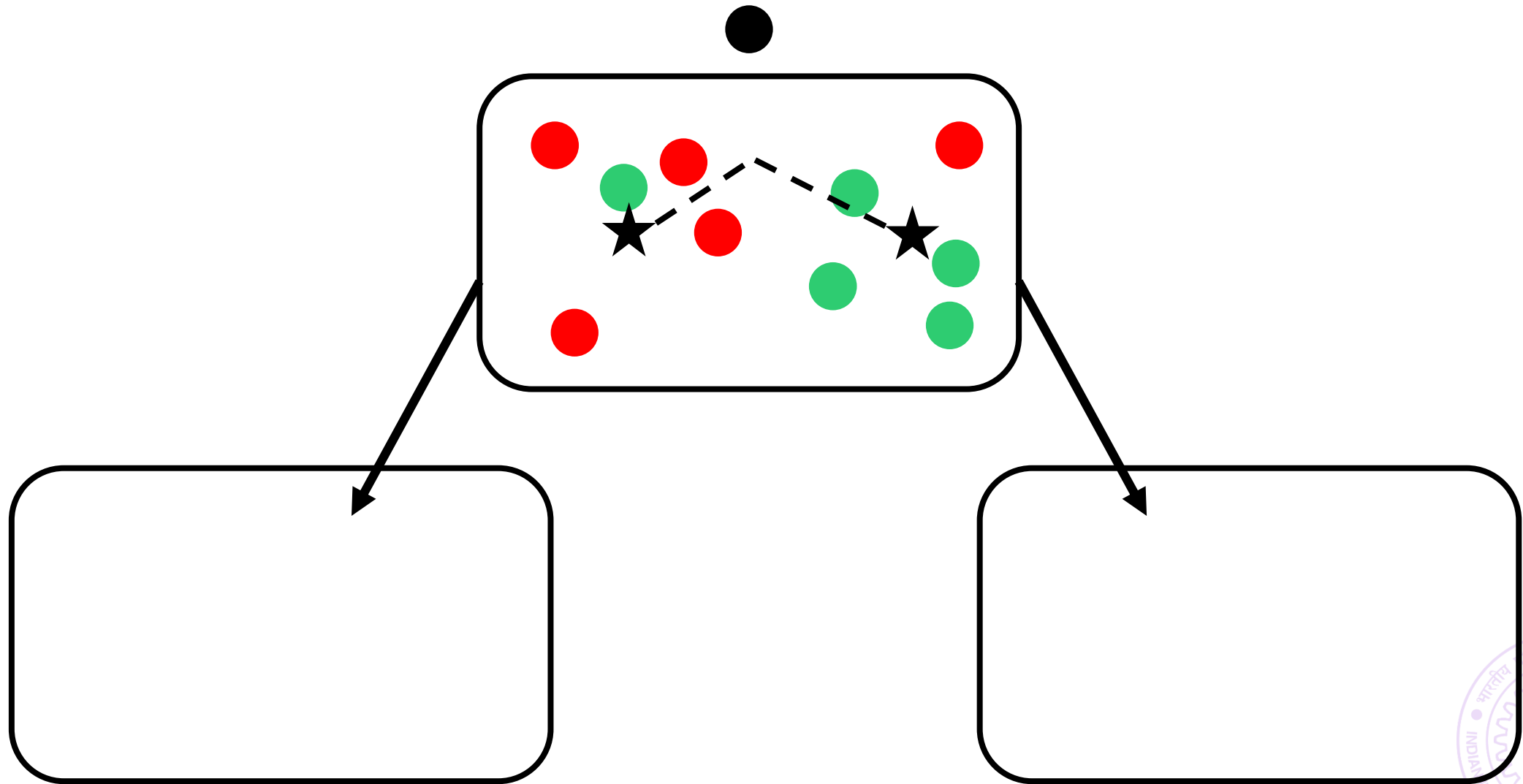
Two possible splitting directions. Let us choose the one that gives us purer children

Purest vertical split is more pure – use it!



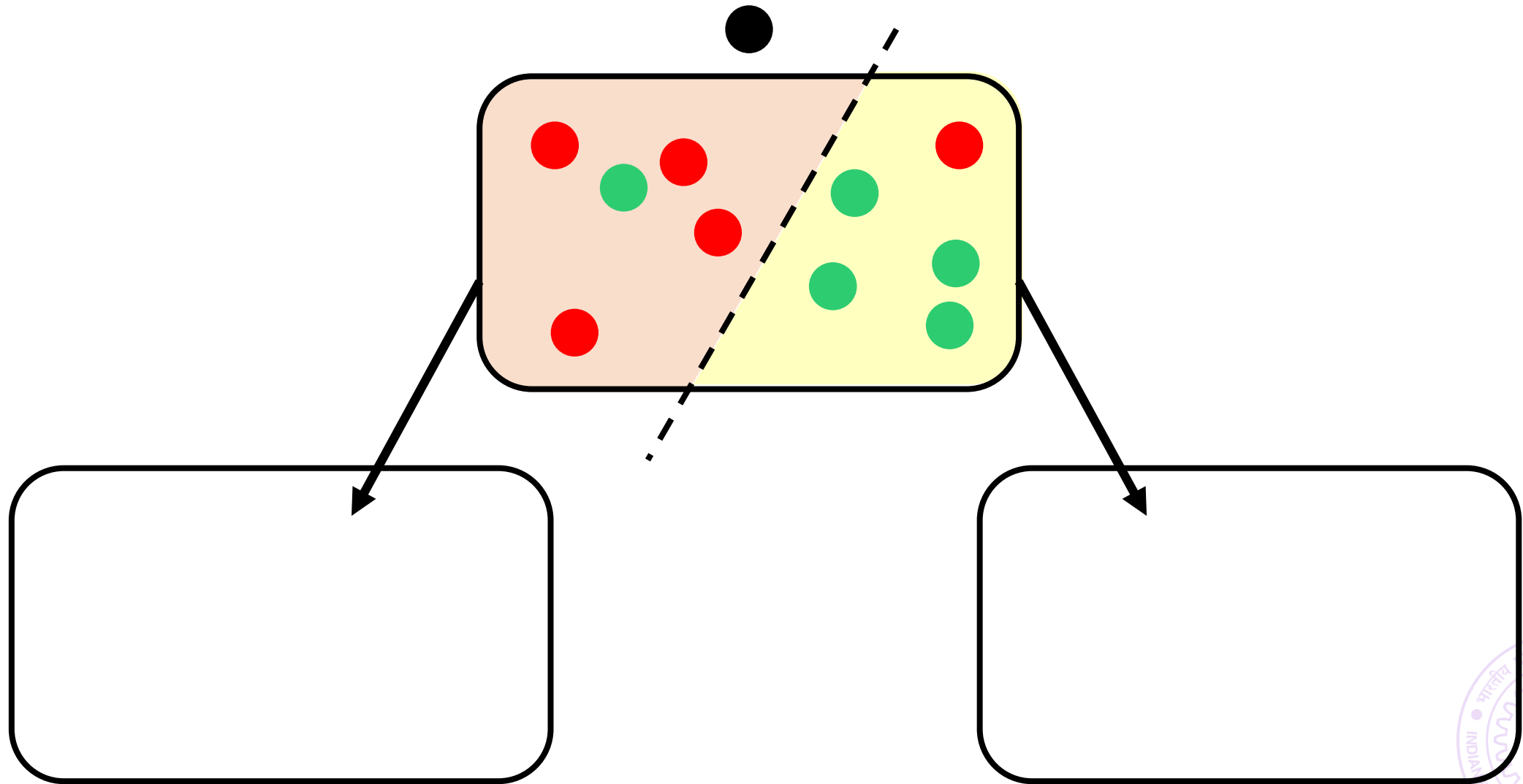
Node splitting via LwP/linear classifiers

12



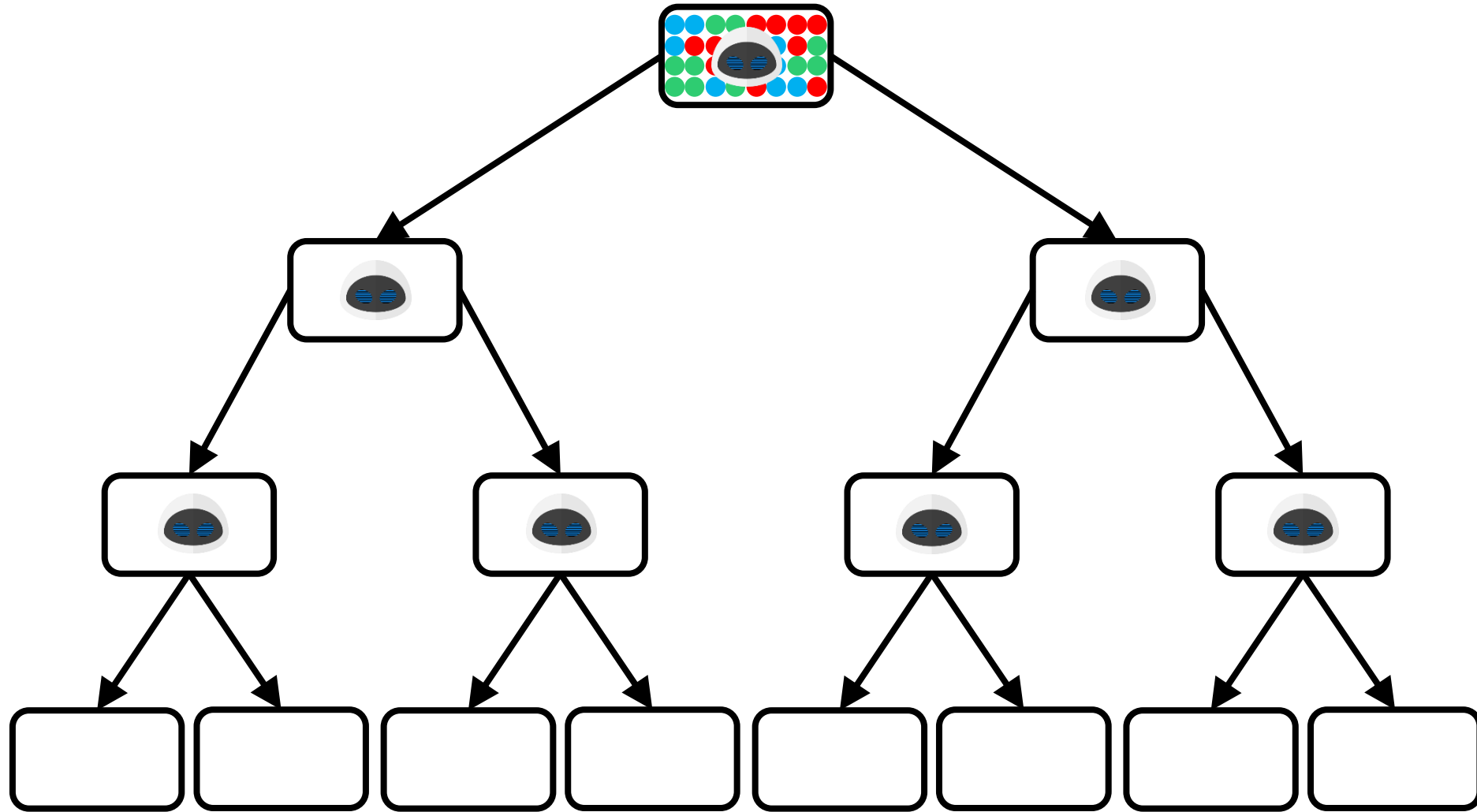
Node splitting via LwP/linear classifiers

13



One Final Recap

14



Pruning Strategies

15

Stop if node is pure or almost pure

Stop if all features exhausted – avoid using a feature twice on a path

Limits depth of tree to d (num of dimensions)

Can stop if a node is ill-populated i.e. has few training points

Can also (over) grow a tree and then merge nodes to shrink it

Merge two leaves and see if it worsens performance on the validation set or not – rinse and repeat

Use a validation set to make these decisions (never touch test set)



Decision Trees - Lessons

16

Very fast at making predictions (if tree is reasonably balanced)

Can handle discrete data (even non numeric data) as well – e.g. can have a stump as: blood group AB or O go left, else go right

LwP, NN have difficulty with such non-numeric and discrete data since difficult to define distance and averages with them (however, there are workarounds to do NN/LwP with discrete data as well)

Tons of DT algorithms – both classical (ID3, C4.5) as well as very recent (GBDT, LPSR, Parabel) – DTs are versatile and very useful

Reason: DT learning is an NP hard problem – no single algo ☹️

If you think you have a better way of splitting nodes or handling leaf nodes, it might be the next big thing in DT learning 😊

