Decision Trees

CS771: Introduction to Machine Learning

Purushottam Kar

Announcements

Please form groups of 5 by Wednesday – will give you an online form to specify your group details – please do not mail group details to me

Please activate your Piazza account if not done so

Heads-up: Quiz 1 is coming up next week August 14 – do not miss

Recap of Last Lecture

Notion of decision boundary – points where the classifiers are confused about what to predict - all classifiers have a decision boundary whether LwP, NN, trees, deep nets

Hyperplane classifiers – model vector, bias and their role

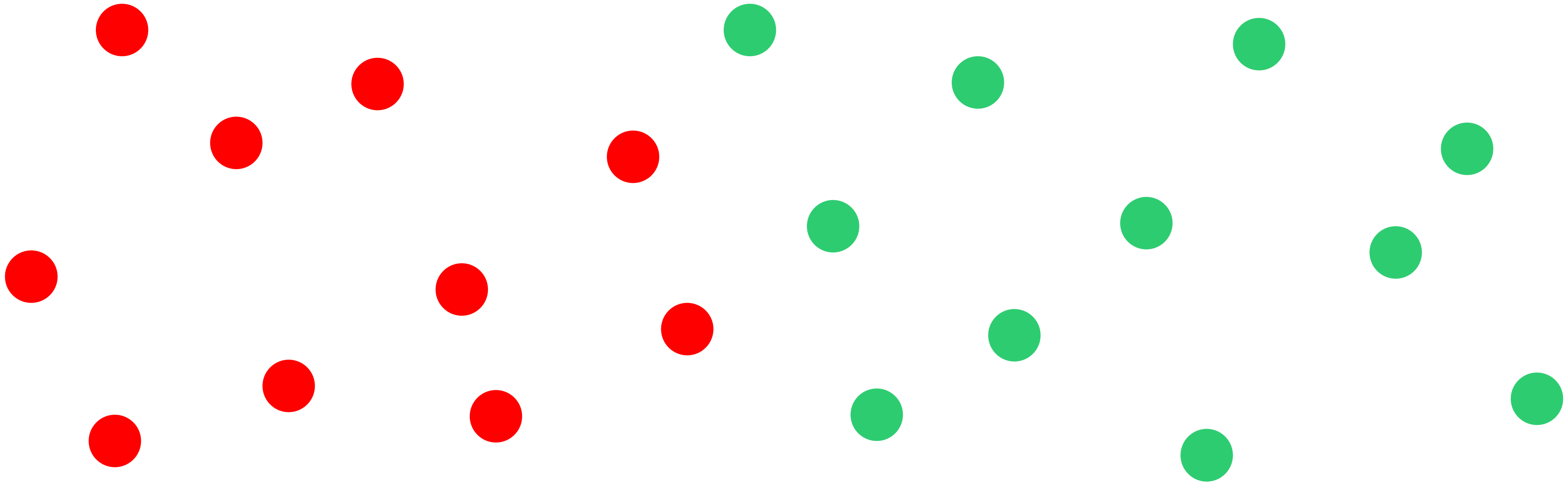
Metric learning and how it can help LwP and NN improve

NN variants – kNN, rNN (not to be confused with RNN)

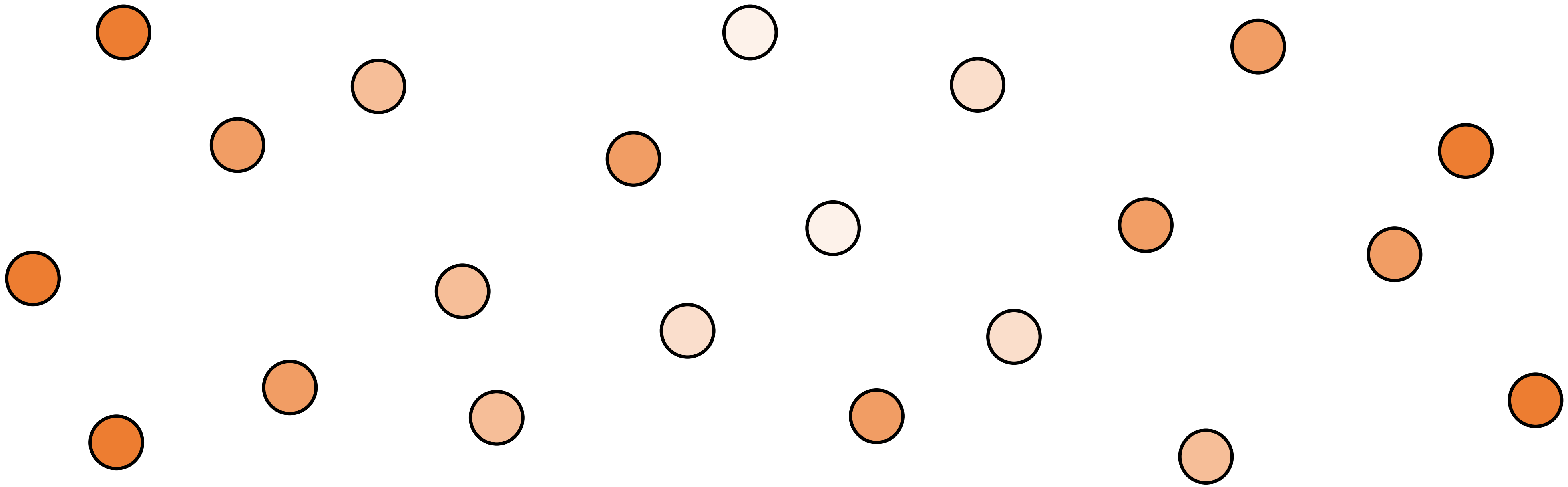
Hyperparameter tuning via held out validation/cross validation

Regression with rNN

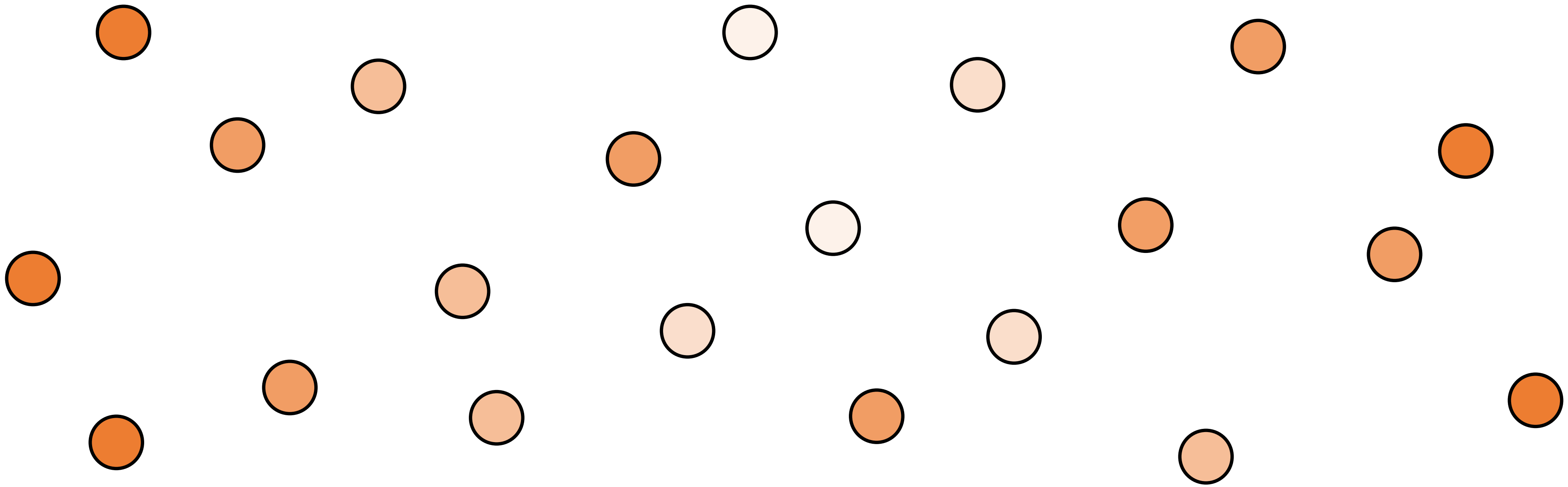
Regression with rNN



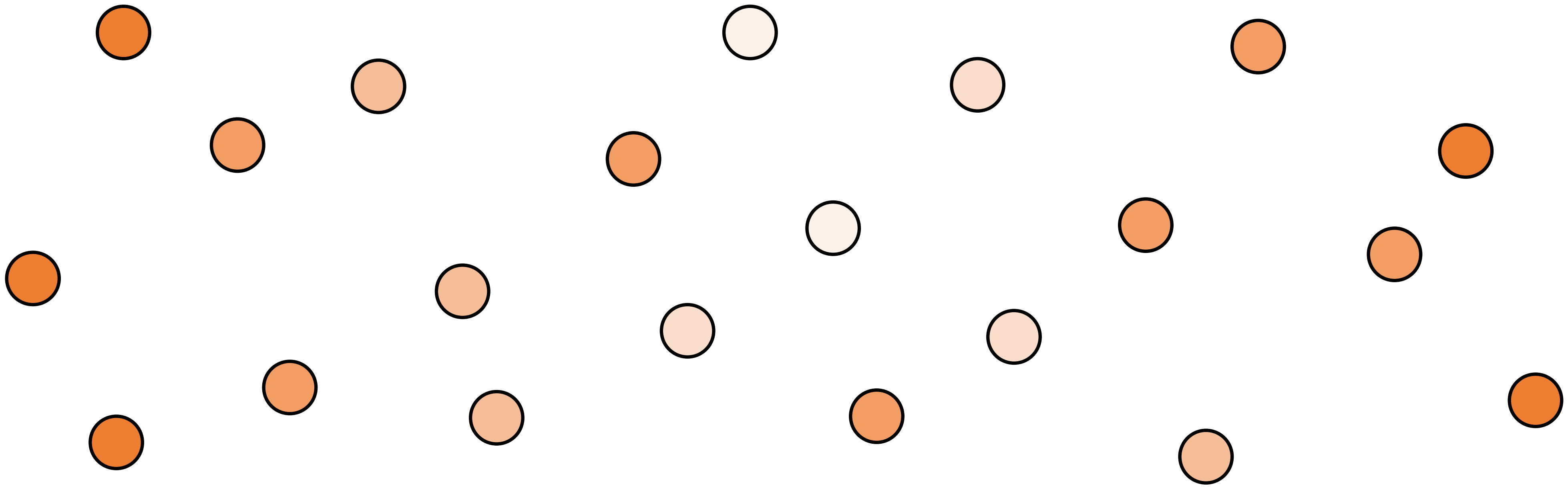
Regression with rNN

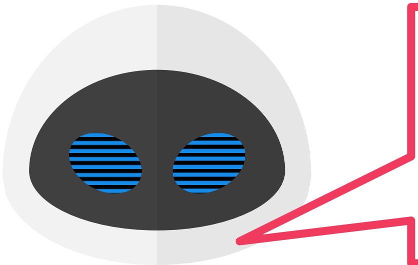


Regression with rNN



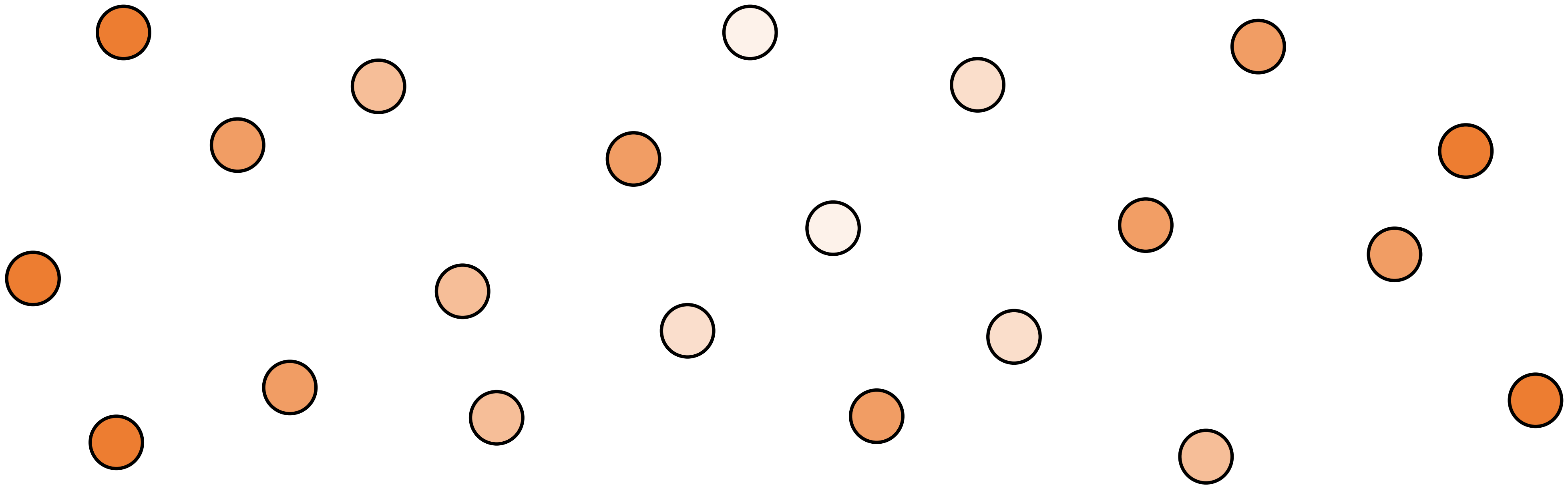
Regression with rNN

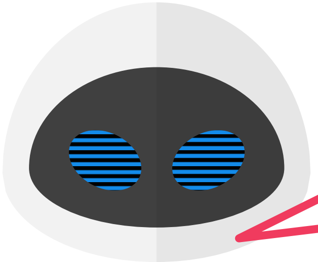




In regression, each point needs to be given a real-valued score instead of a label like spam/non-spam

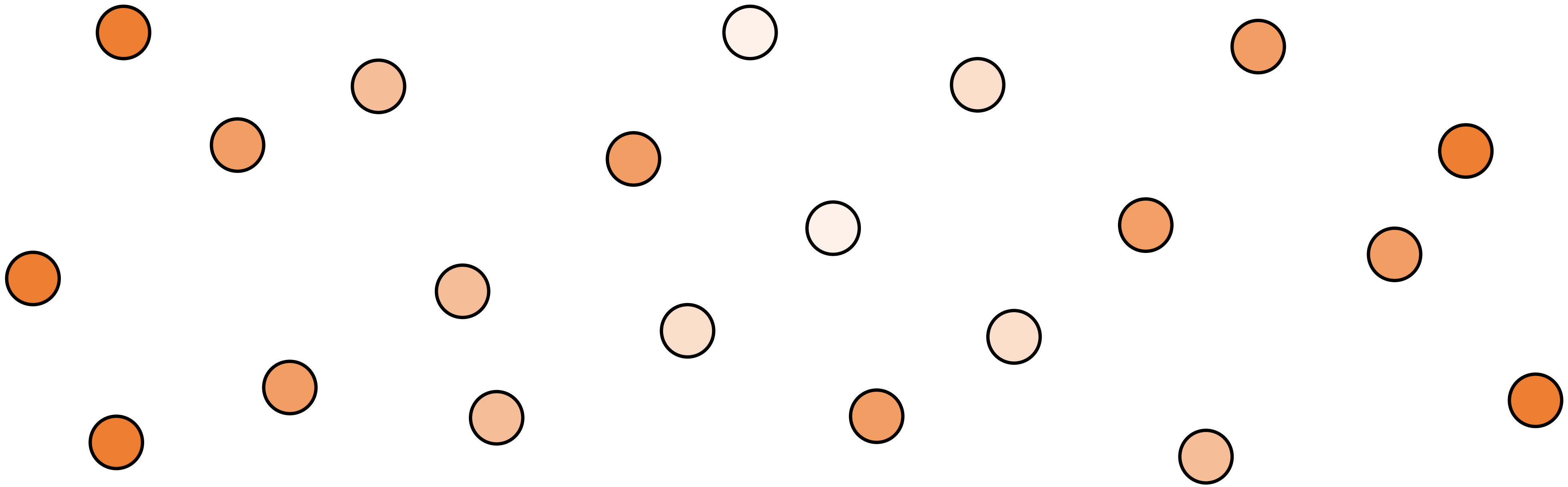
Regression with rNN

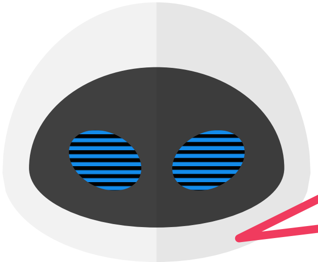




In regression, each point needs to be given a real-valued score instead of a label like spam/non-spam

Regression with rNN

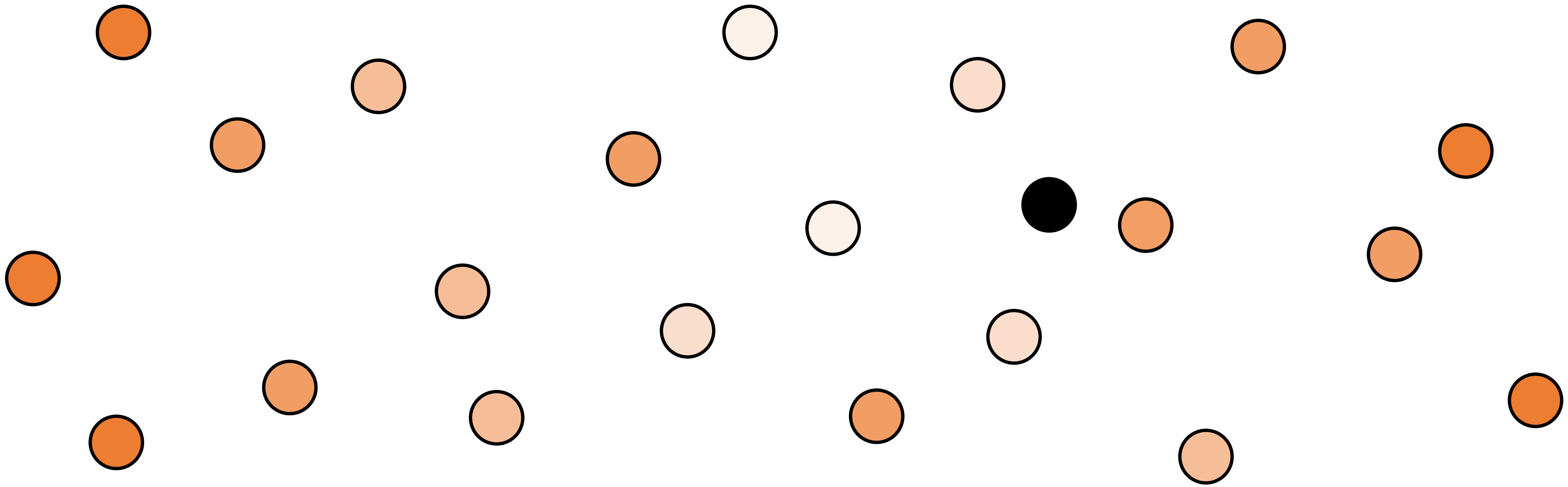


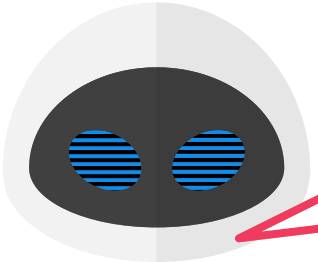


In regression, each point needs to be given a real-valued score instead of a label like spam/non-spam

E.g. when data point is a student and we wish to predict their marks

Regression with rNN

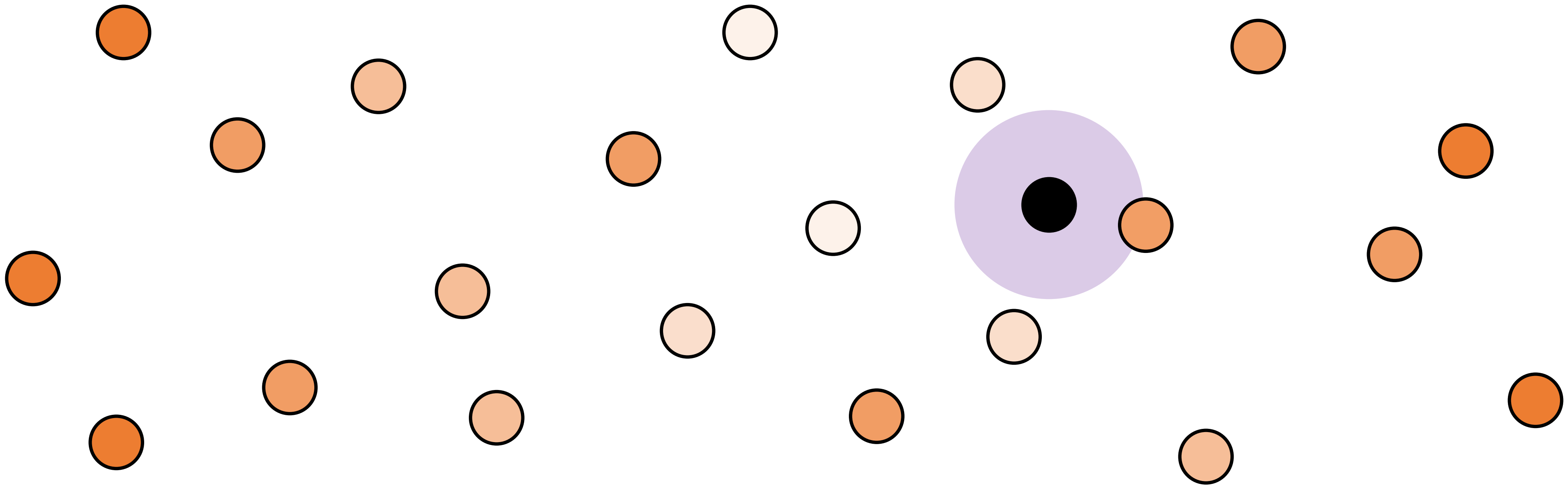


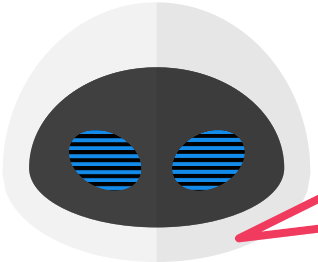


In regression, each point needs to be given a real-valued score instead of a label like spam/non-spam

E.g. when data point is a student and we wish to predict their marks

Regression with rNN

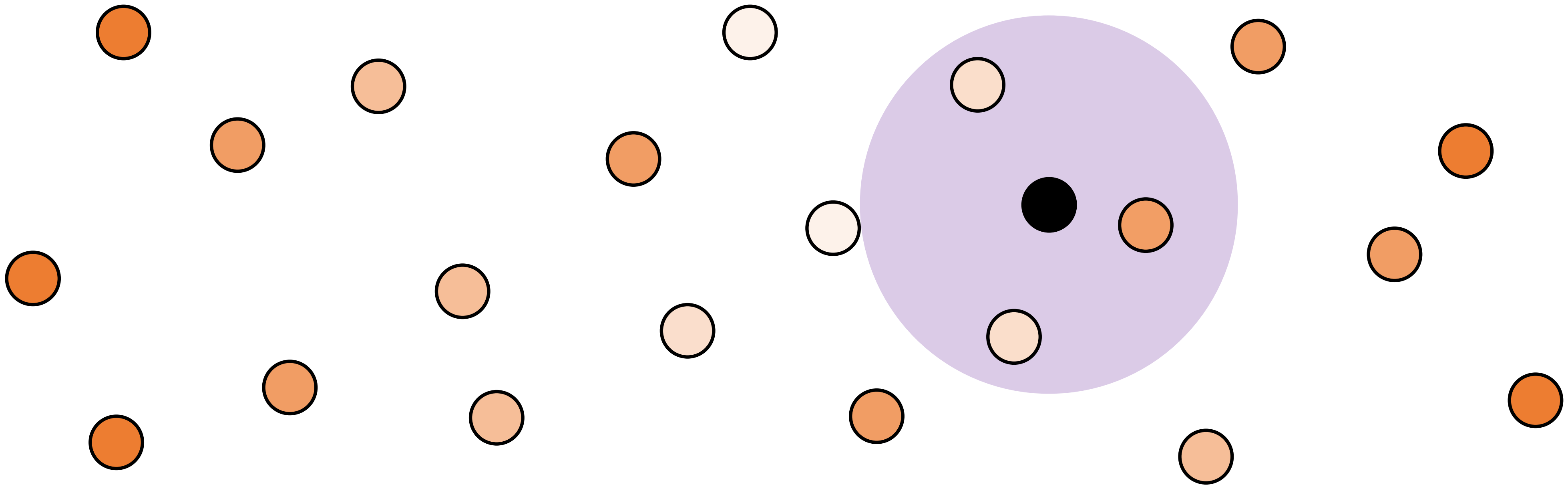


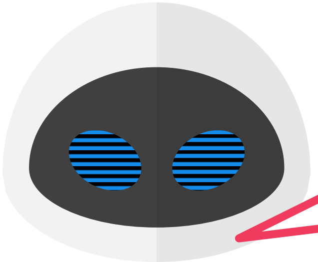


In regression, each point needs to be given a real-valued score instead of a label like spam/non-spam

E.g. when data point is a student and we wish to predict their marks

Regression with rNN

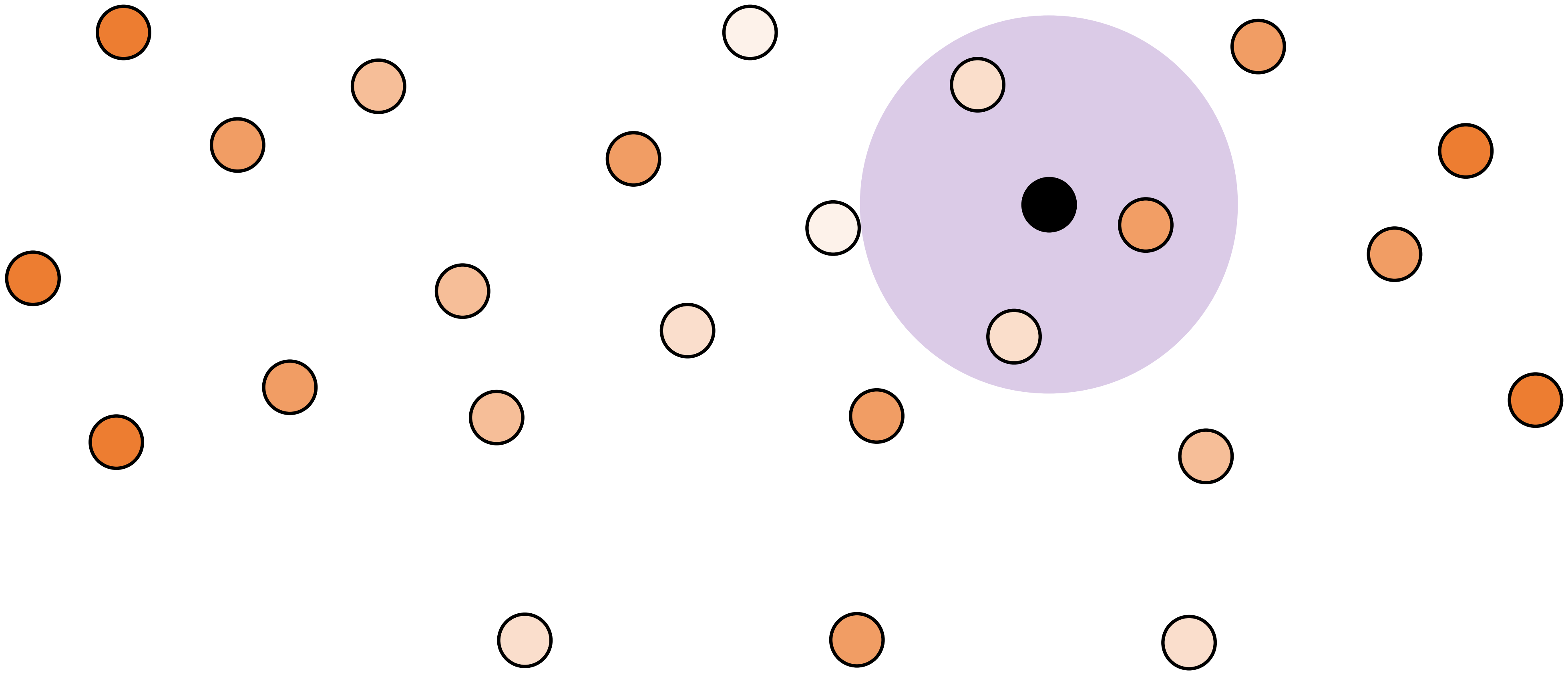


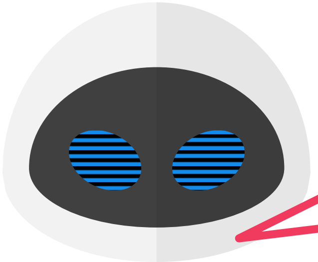


In regression, each point needs to be given a real-valued score instead of a label like spam/non-spam

E.g. when data point is a student and we wish to predict their marks

Regression with rNN

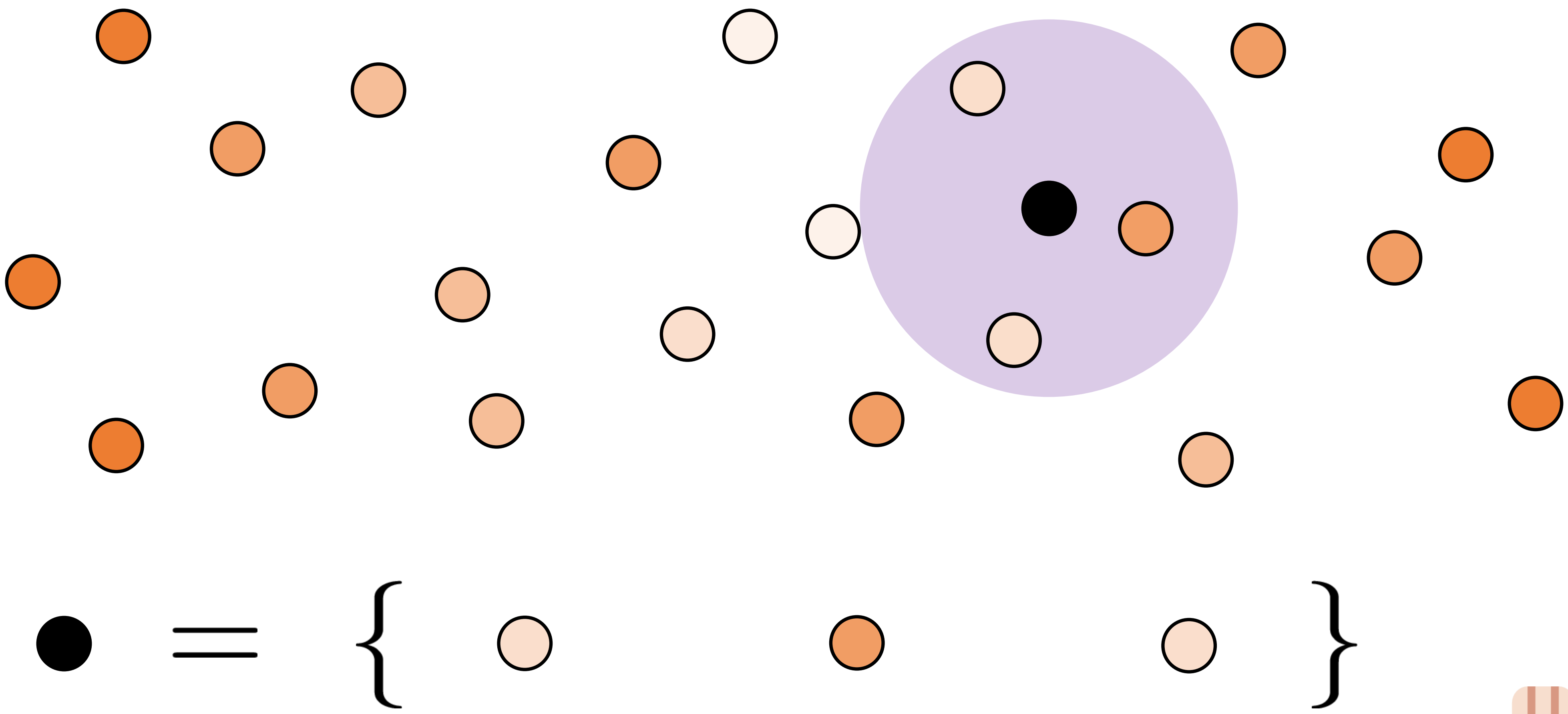


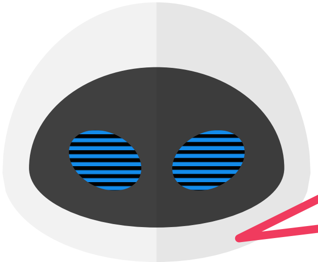


In regression, each point needs to be given a real-valued score instead of a label like spam/non-spam

E.g. when data point is a student and we wish to predict their marks

Regression with rNN

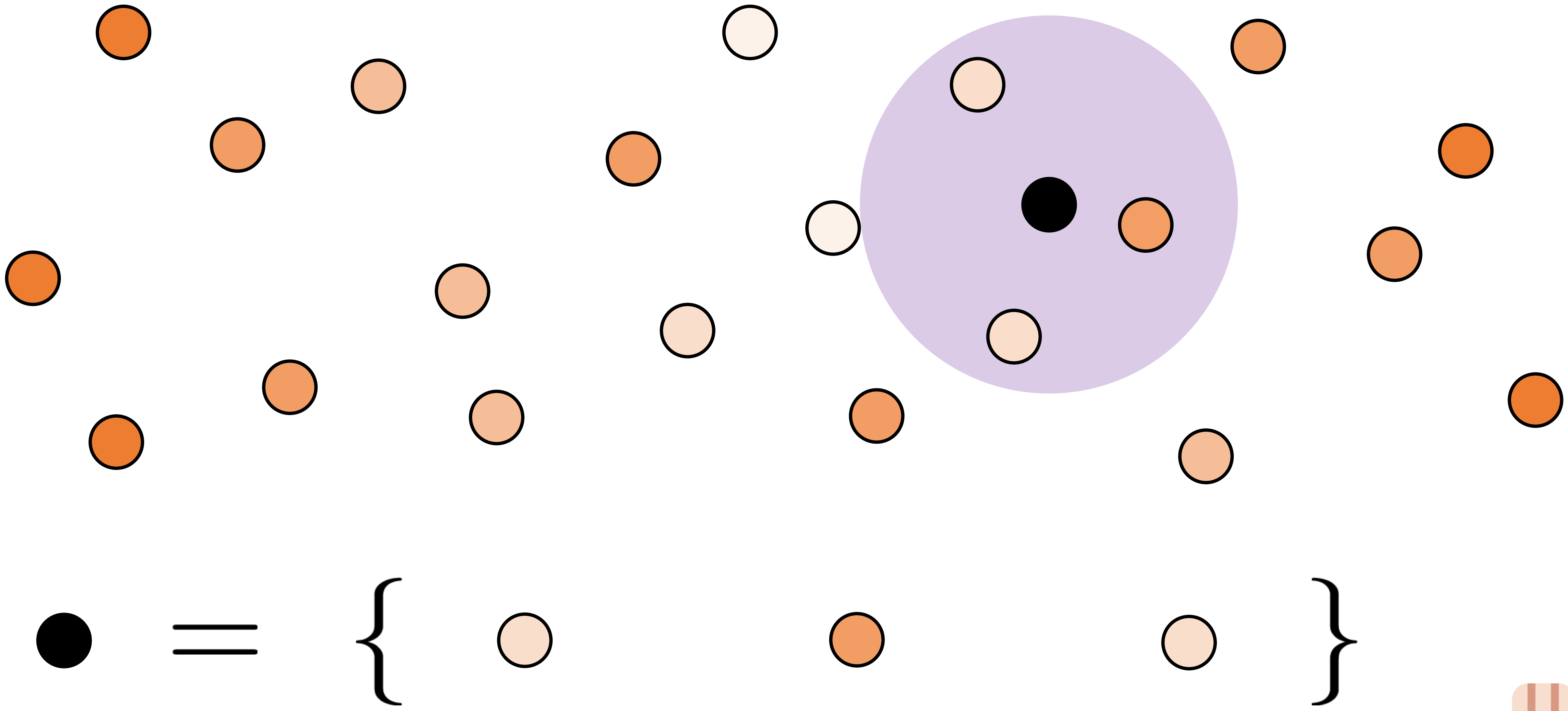


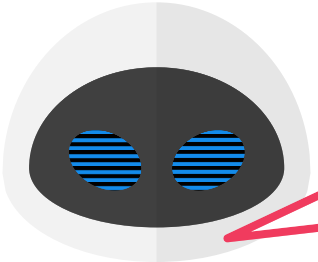


In regression, each point needs to be given a real-valued score instead of a label like spam/non-spam

E.g. when data point is a student and we wish to predict their marks

Regression with rNN

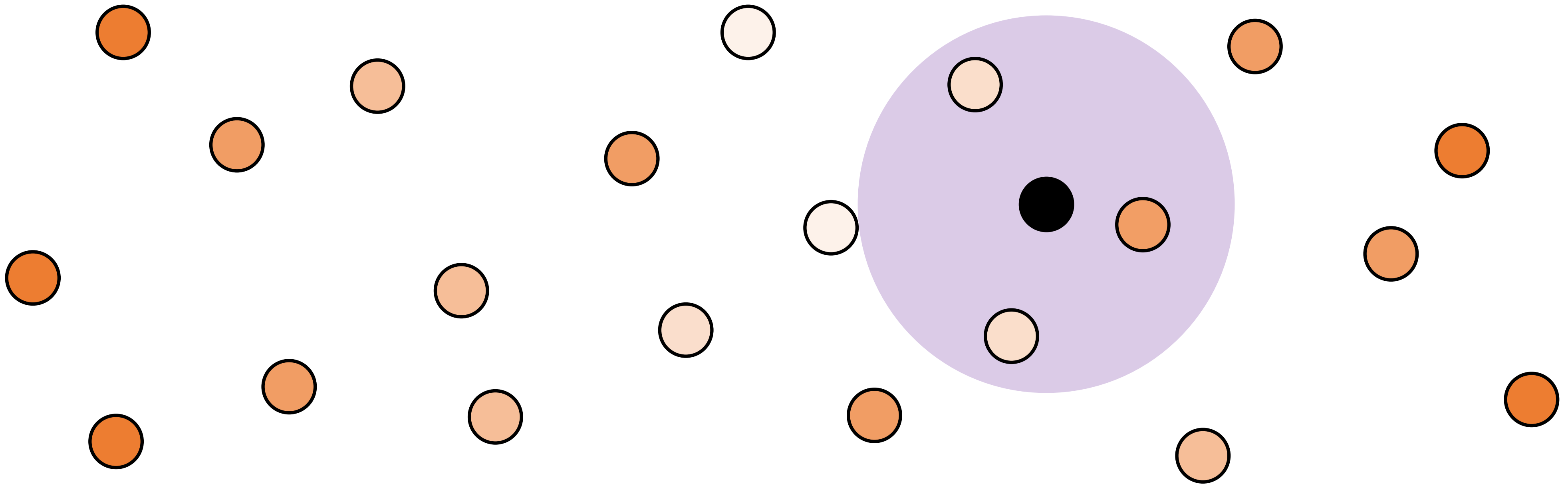




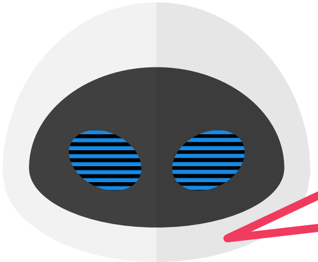
Once you have found the neighbours, you can average their scores to predict a score for the test data point

E.g. when data point is a student and we wish to predict their marks

Regression with rNN



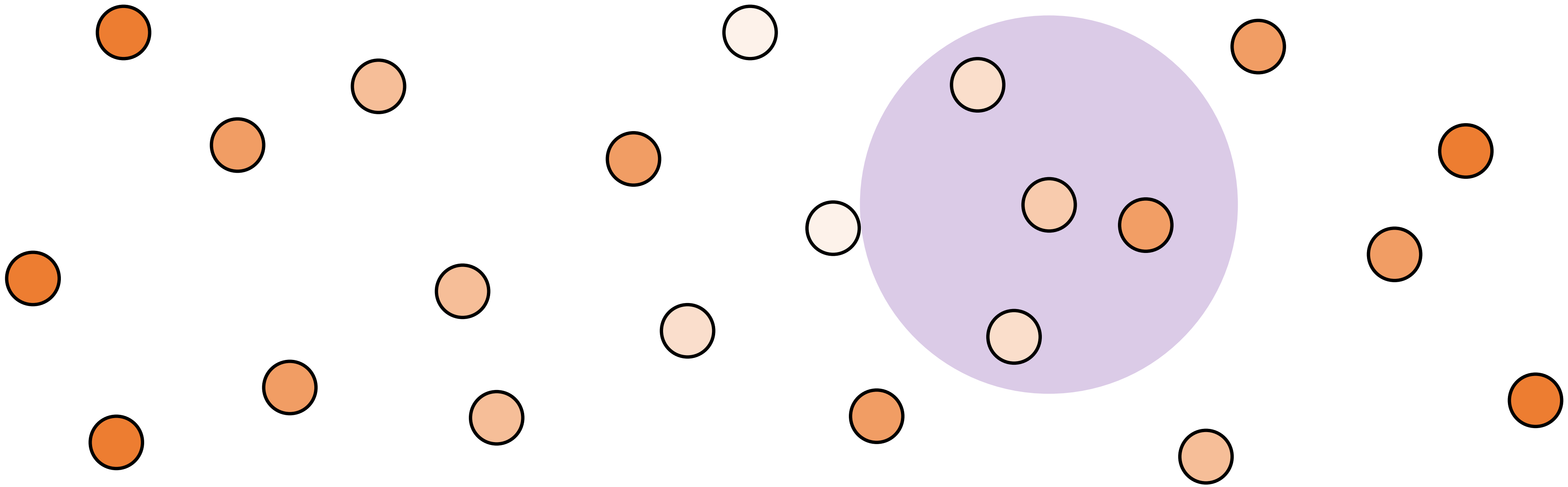
$$\bullet = \left\{ \frac{1}{3} \circ + \frac{1}{3} \circ + \frac{1}{3} \circ \right\}$$

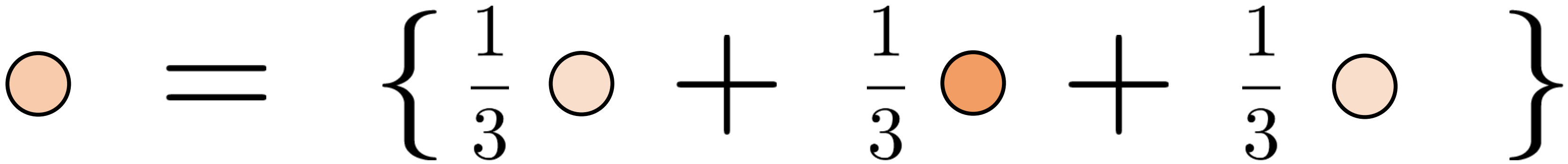


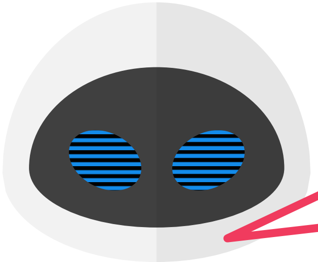
Once you have found the neighbours, you can average their scores to predict a score for the test data point

E.g. when data point is a student and we wish to predict their marks

Regression with rNN



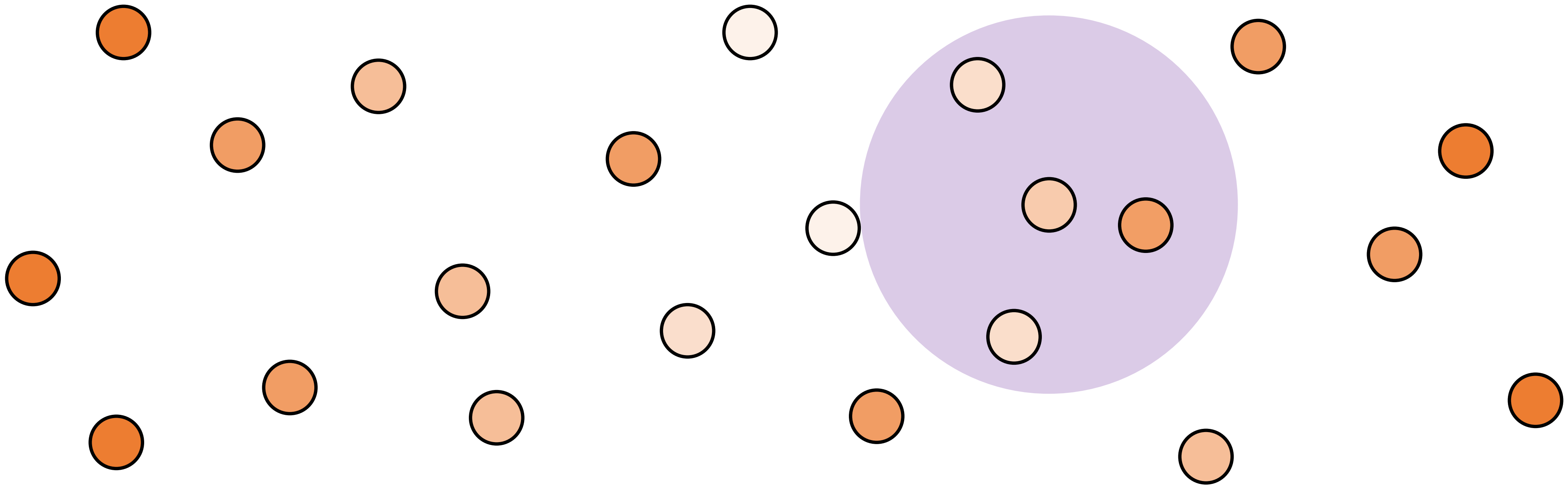


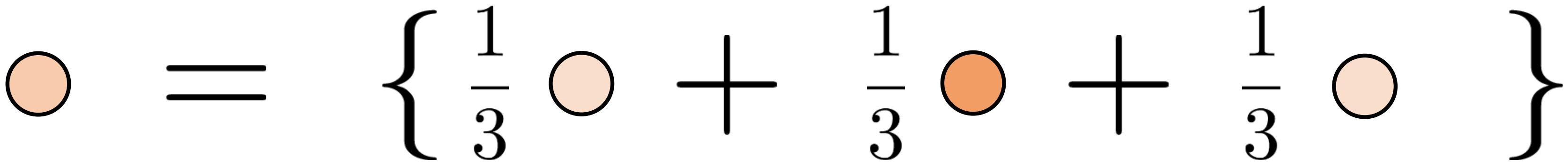


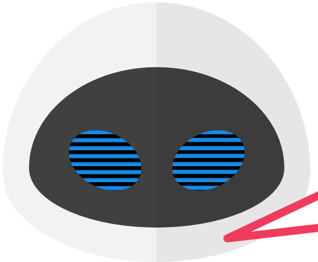
Once you have found the neighbours, you can average their scores to predict a score for the test data point

E.g. when data point is a student and we wish to predict their marks

Regression with Weighted rNN



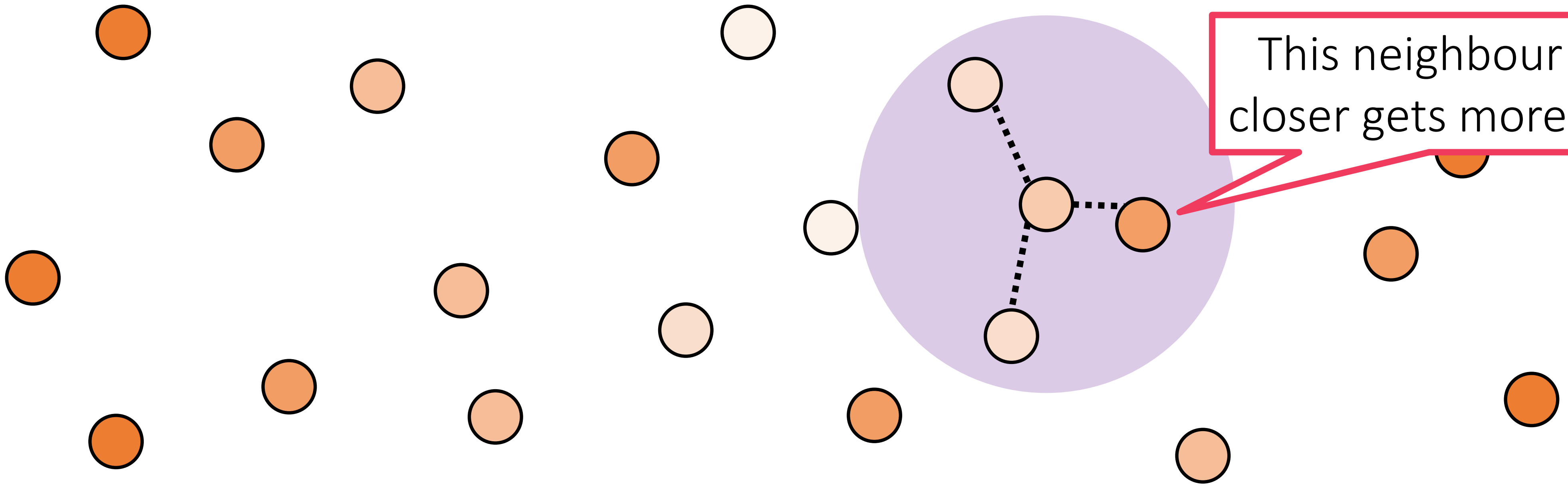




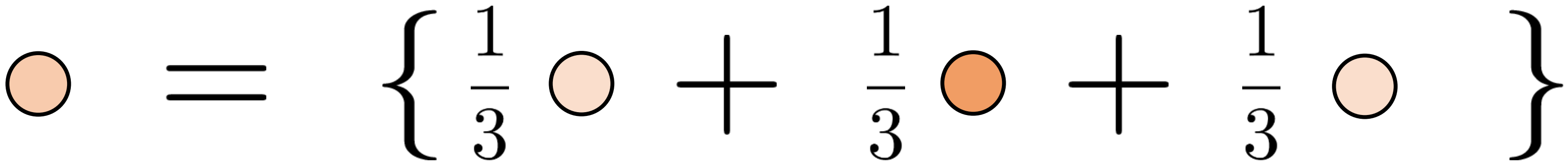
Once you have found the neighbours, you can average their scores to predict a score for the test data point

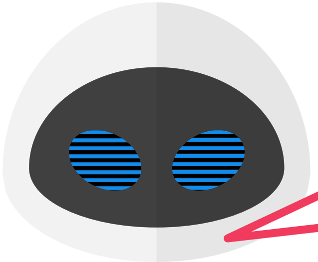
E.g. when data point is a student and we wish to predict their marks

Regression with Weighted rNN



This neighbour being closer gets more weight

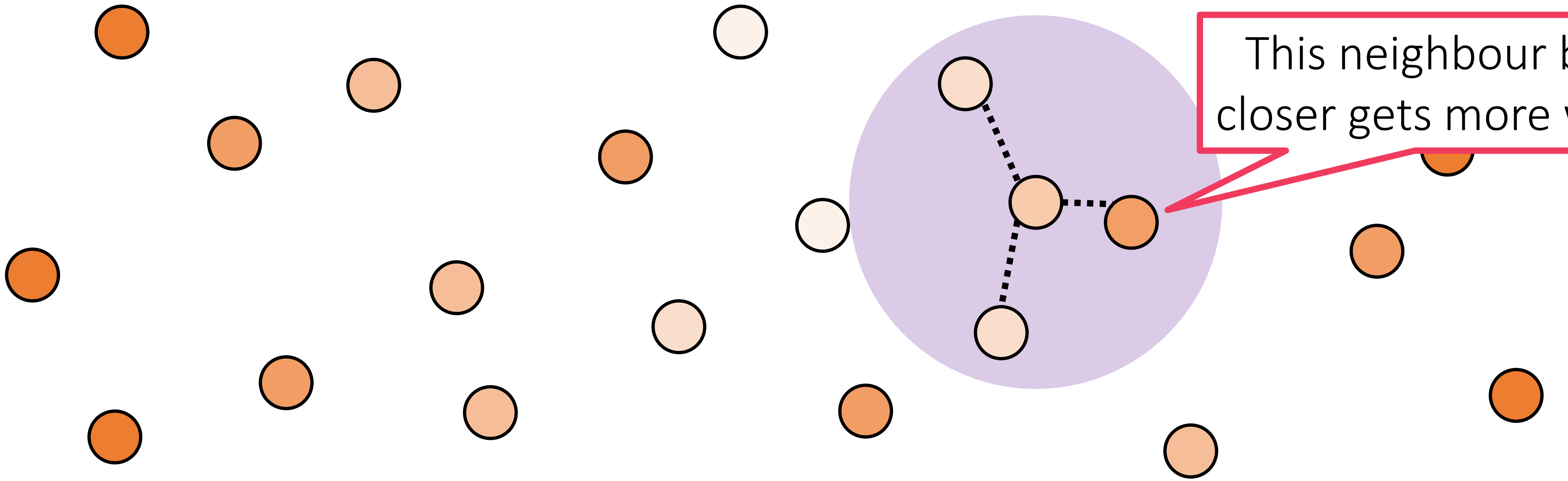




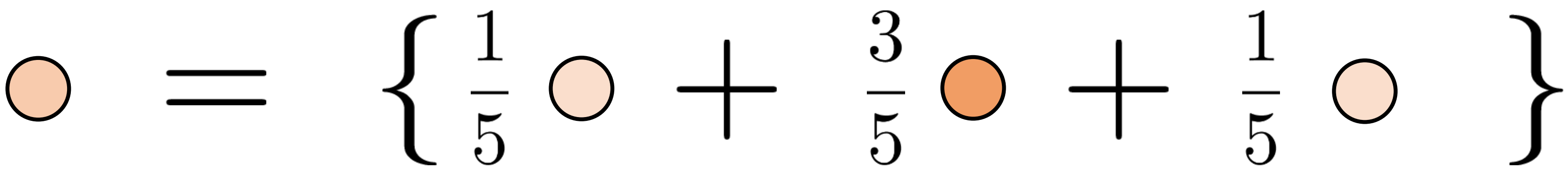
Or else, you may give the score of closer neighbours more weight and those of far neighbours less weight

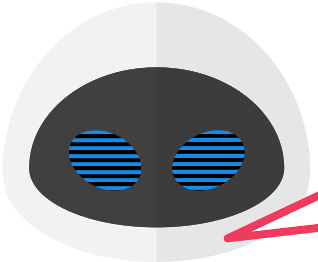
E.g. when data point is a student and we wish to predict their marks

Regression with Weighted rNN



This neighbour being closer gets more weight

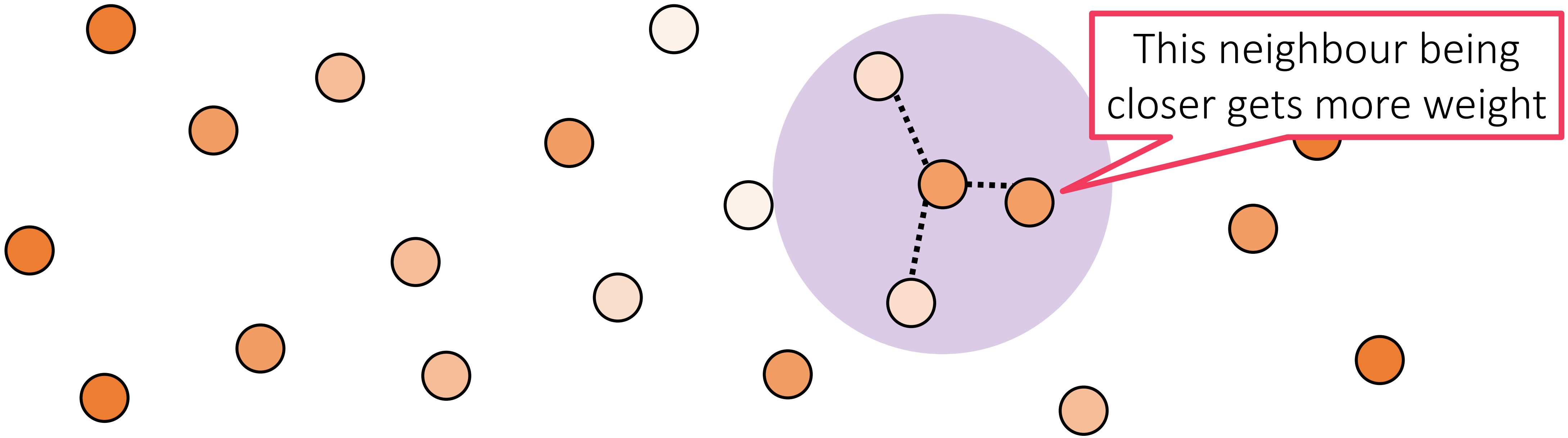


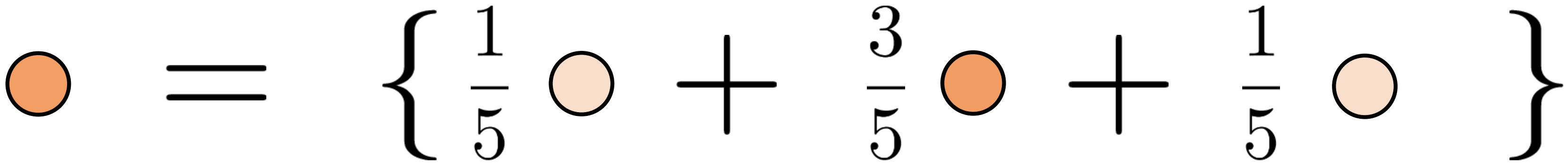


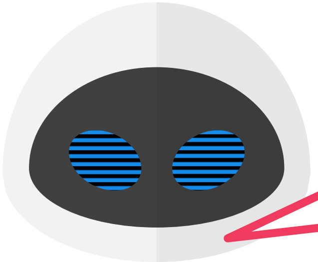
Or else, you may give the score of closer neighbours more weight and those of far neighbours less weight

E.g. when data point is a student and we wish to predict their marks

Regression with Weighted rNN



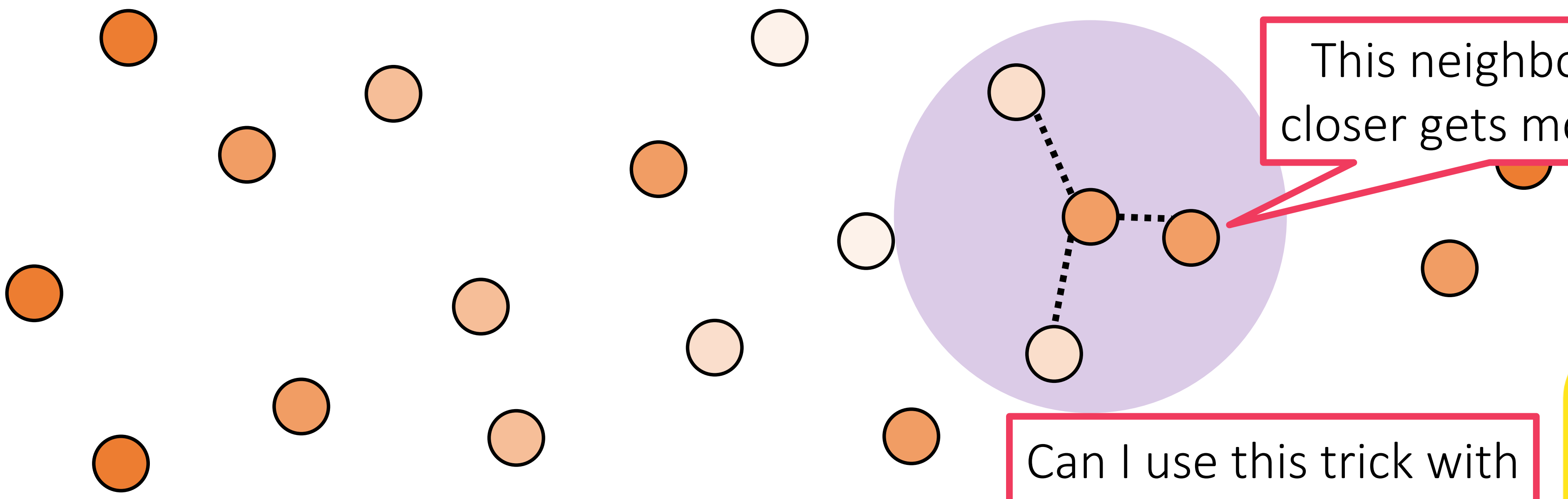




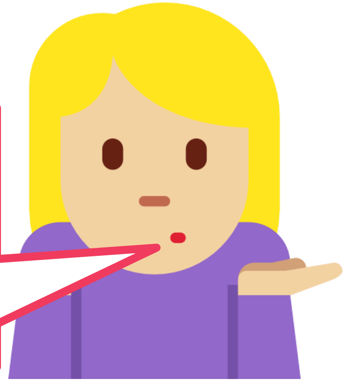
Or else, you may give the score of closer neighbours more weight and those of far neighbours less weight

E.g. when data point is a student and we wish to predict their marks

Regression with Weighted rNN

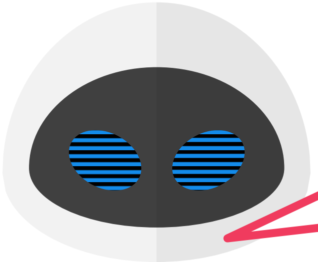


This neighbour being closer gets more weight



Can I use this trick with kNN? Can I use it while doing classification?

$$\bigcirc = \left\{ \frac{1}{5}\bigcirc + \frac{3}{5}\bigcirc + \frac{1}{5}\bigcirc \right\}$$

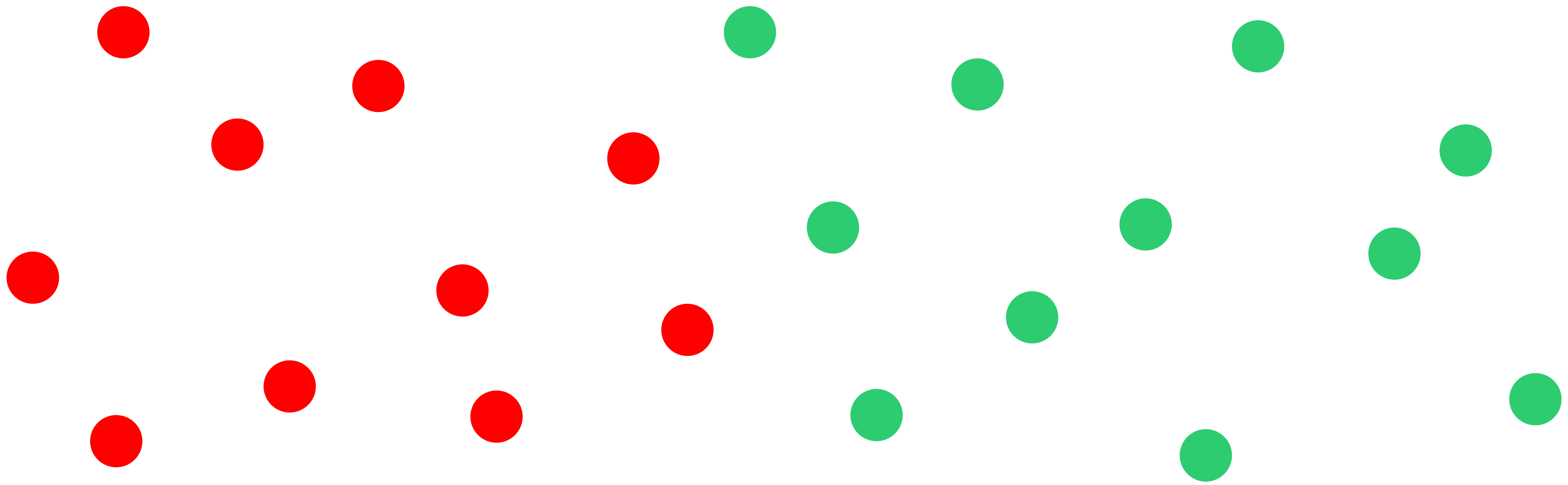


Yes, and yes. Need to be a bit careful while doing weighted classification since adding labels makes no sense

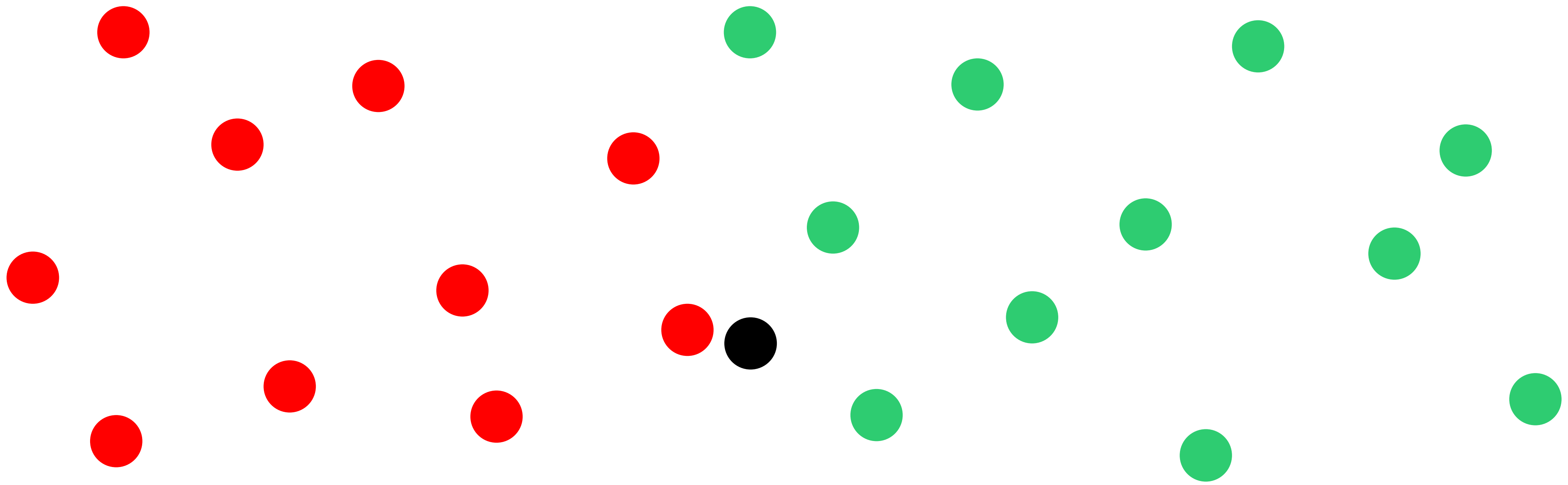
E.g. when data point is a student and we wish to predict their marks

Classification with Weighted rNN

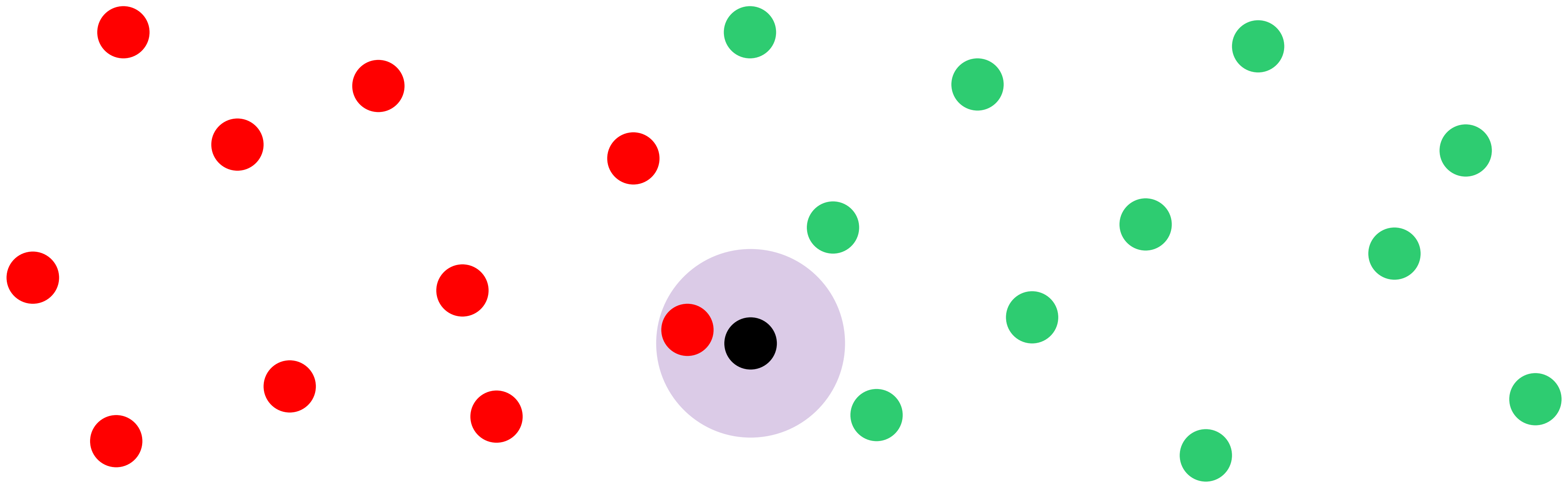
Classification with Weighted rNN



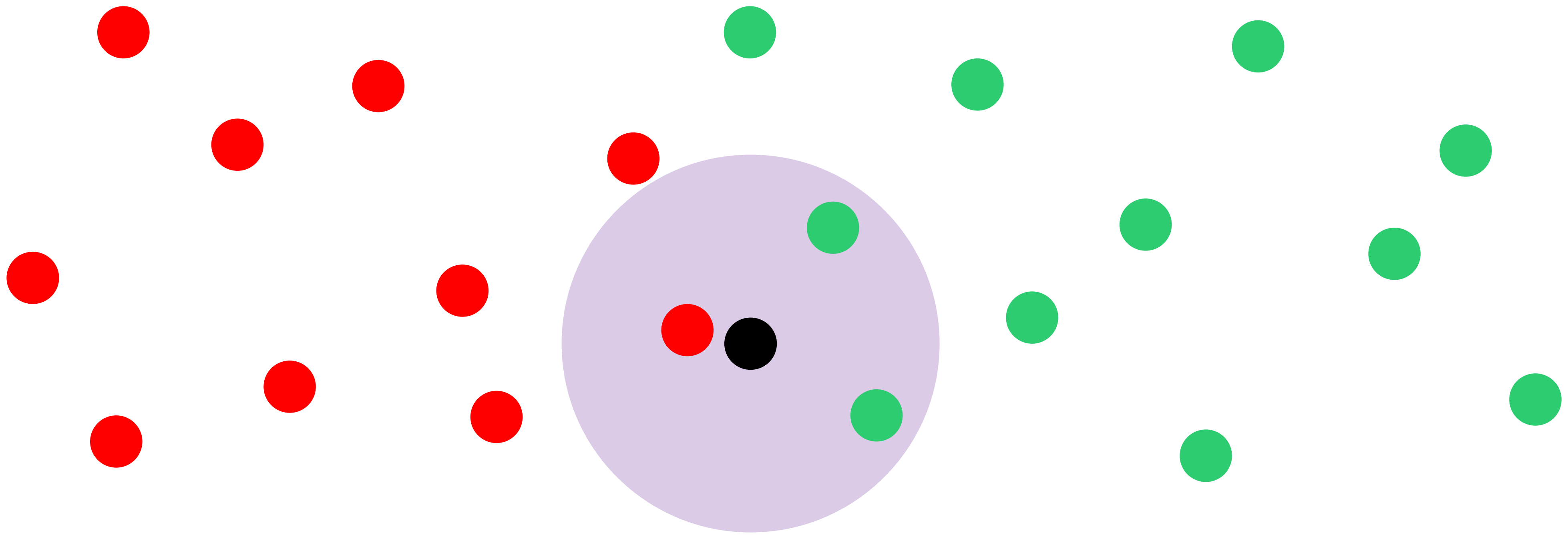
Classification with Weighted rNN



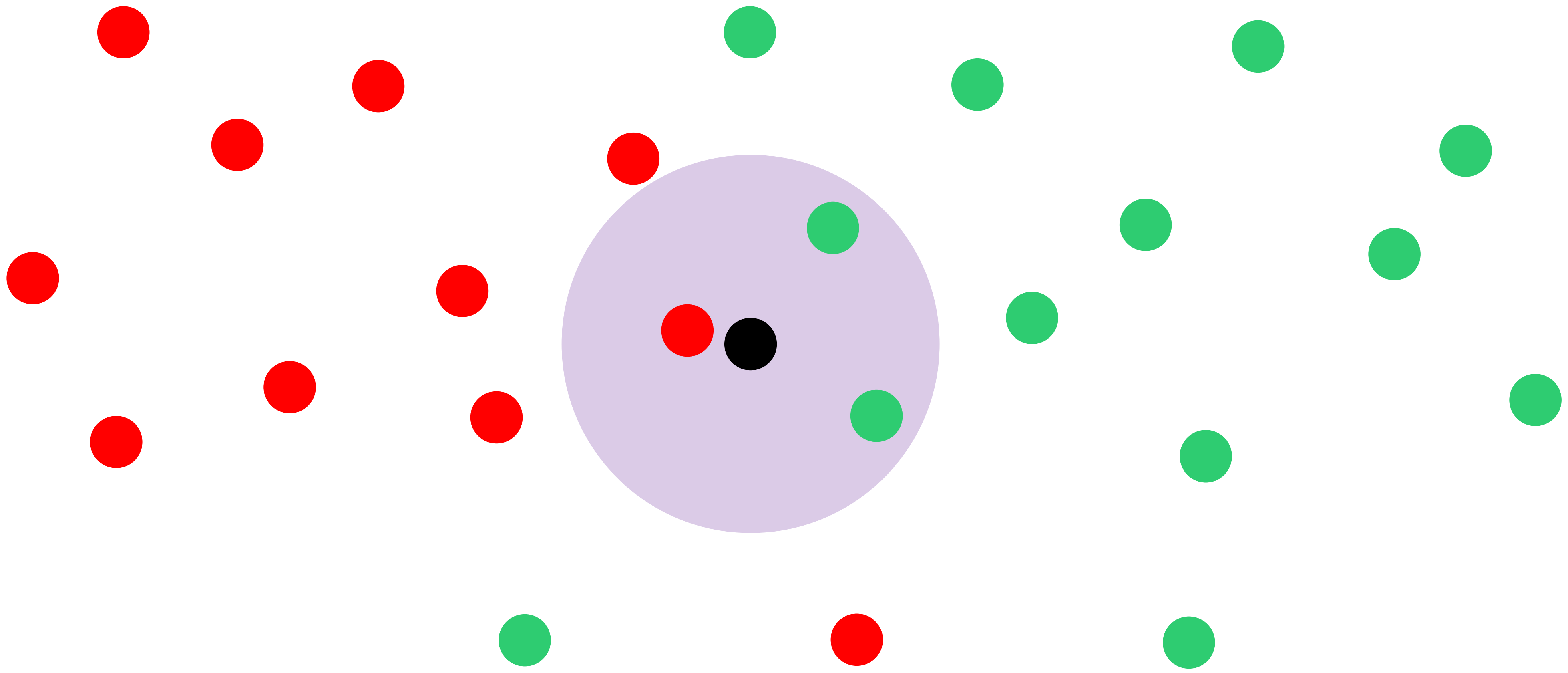
Classification with Weighted rNN



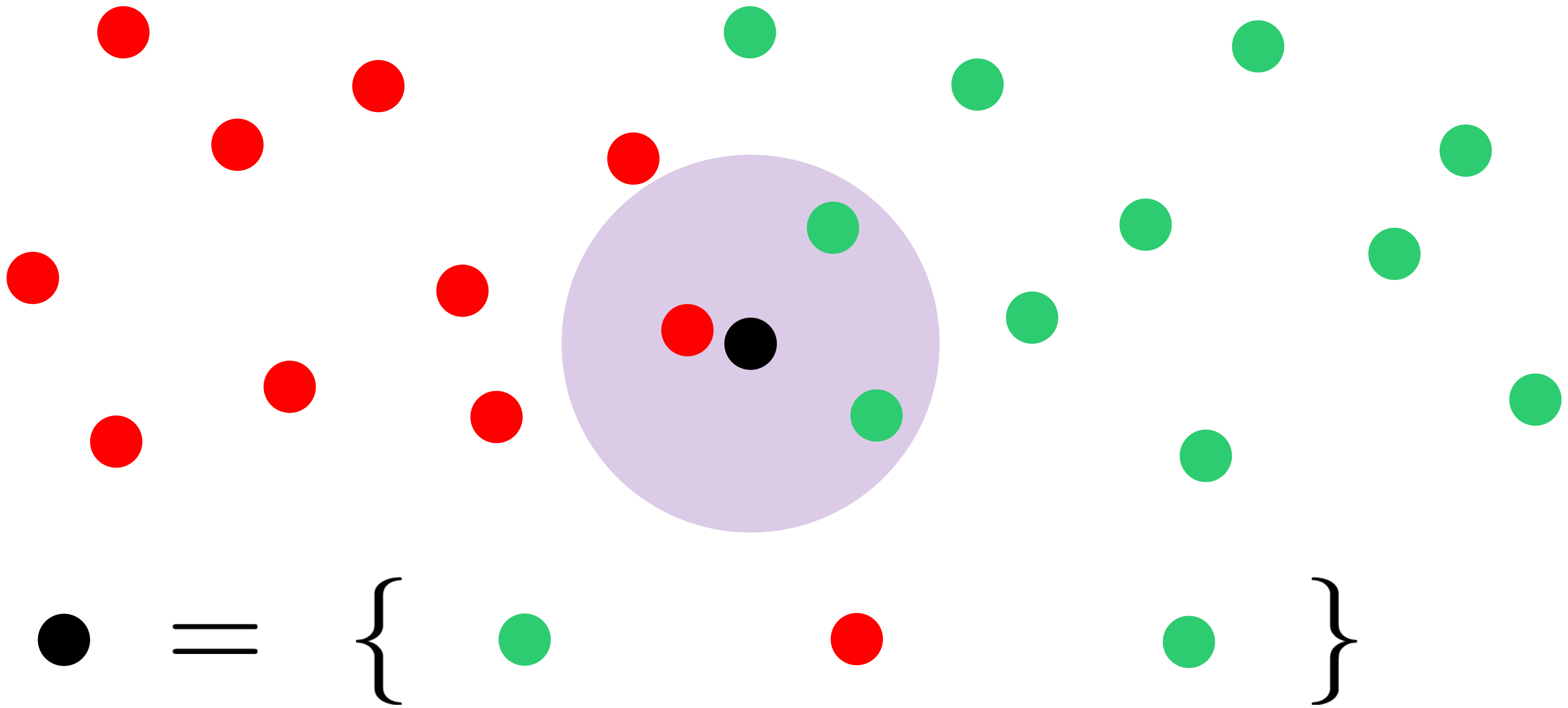
Classification with Weighted rNN



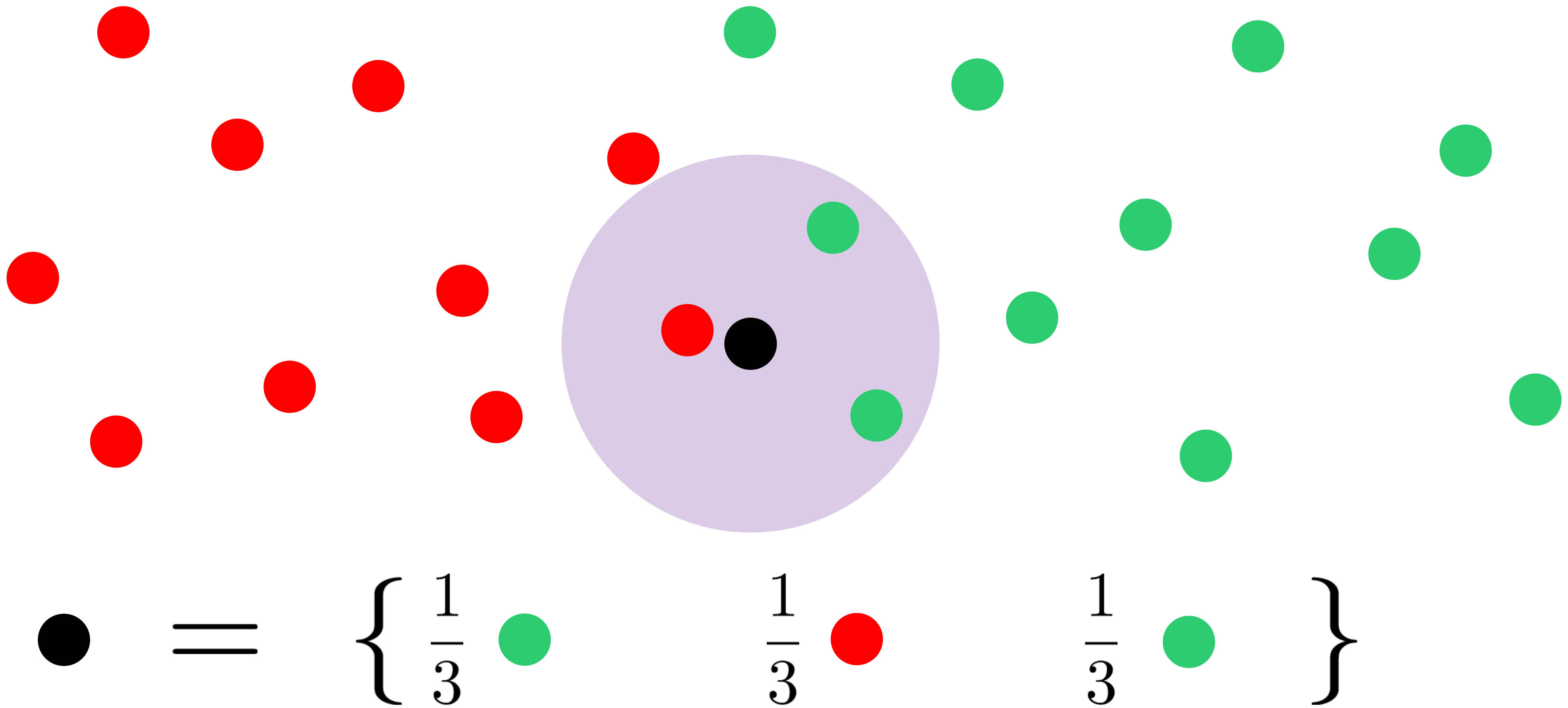
Classification with Weighted rNN



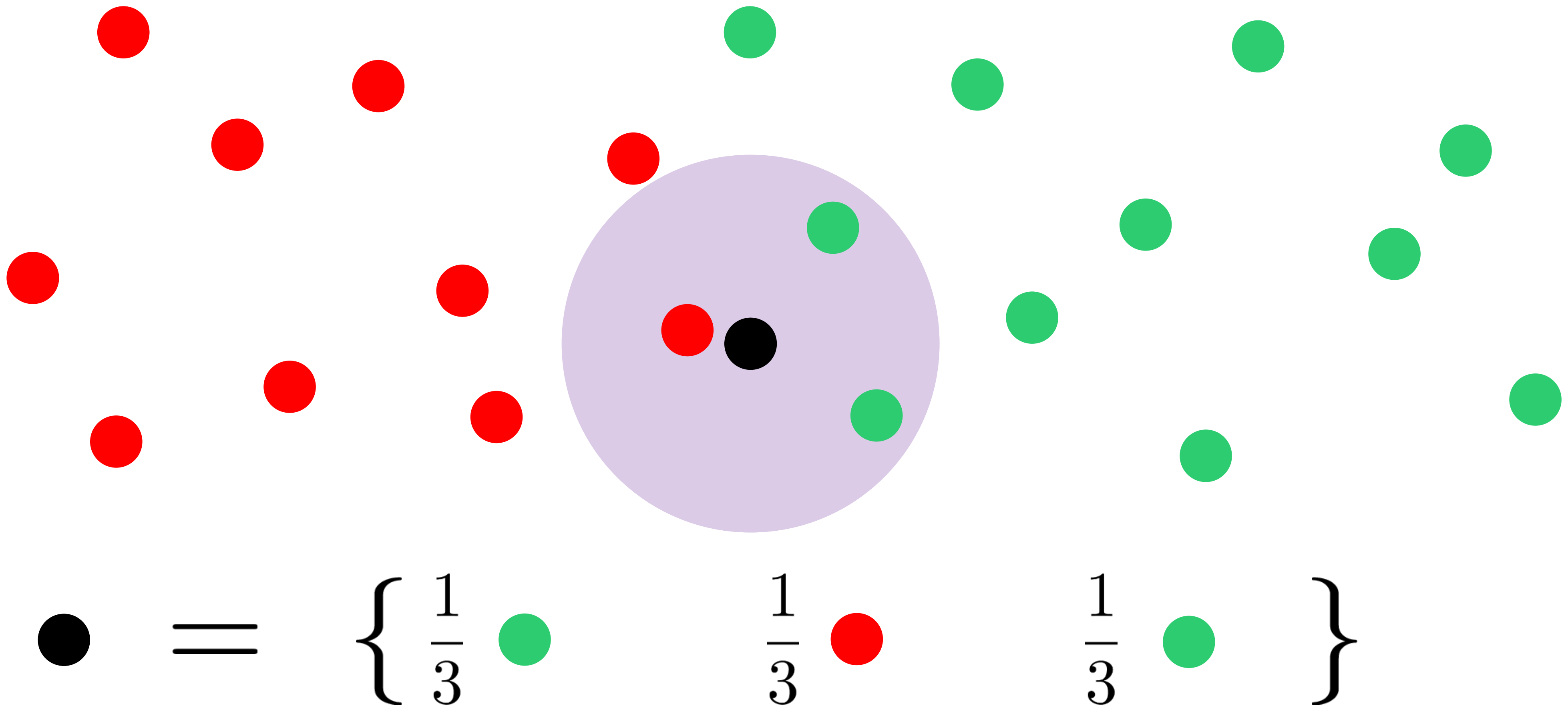
Classification with Weighted rNN



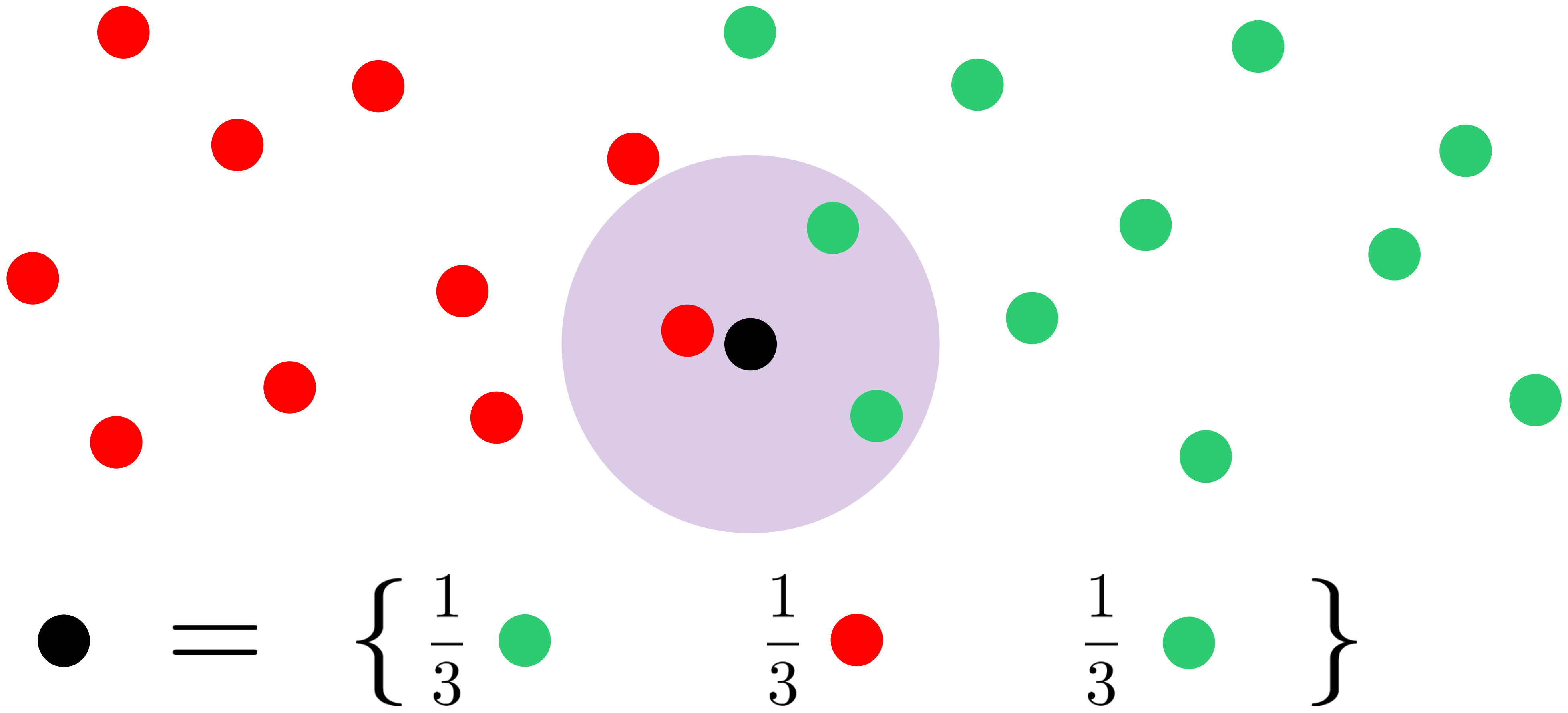
Classification with Weighted rNN

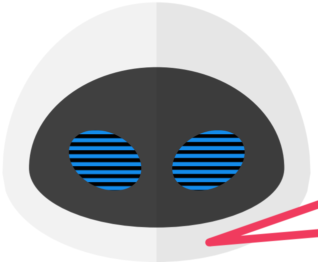


Classification with Weighted rNN



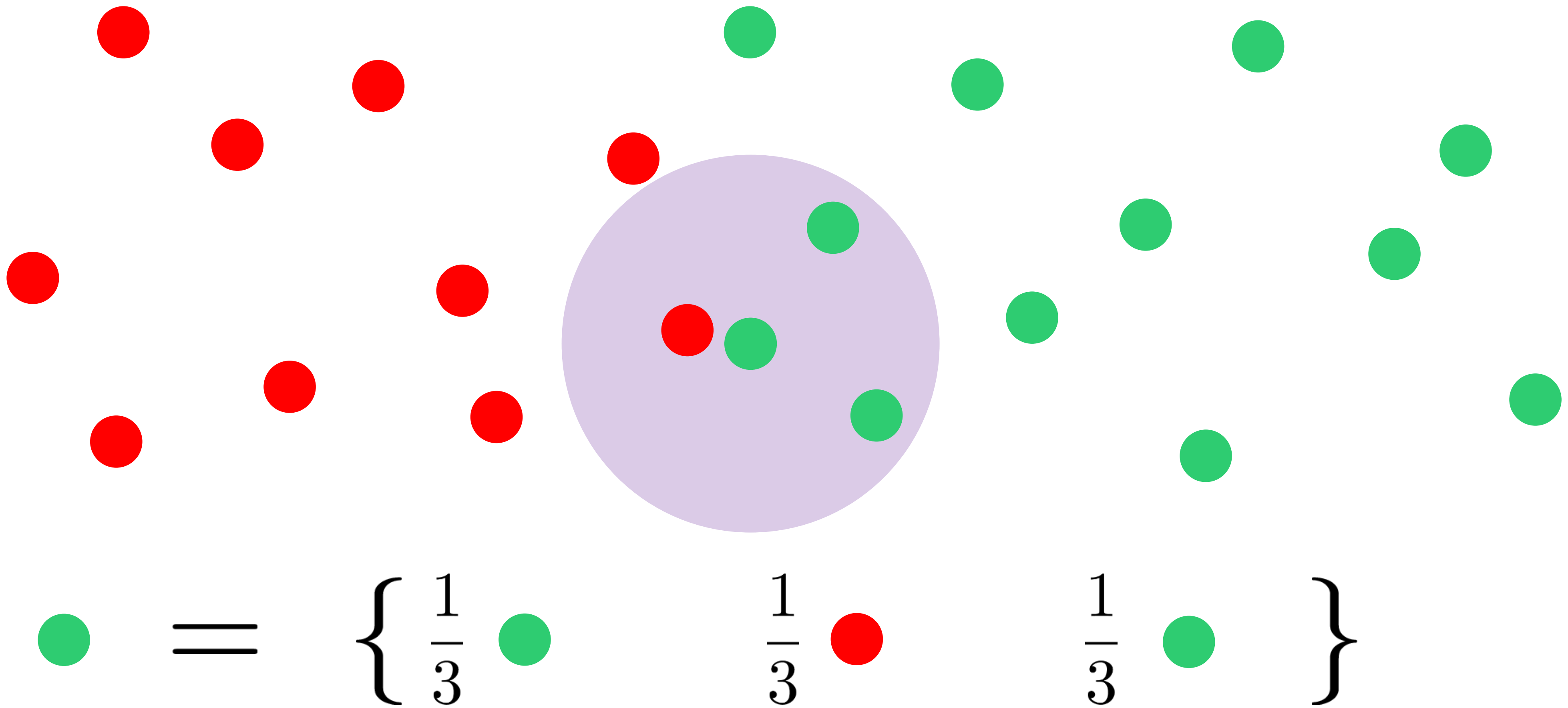
Classification with Weighted rNN

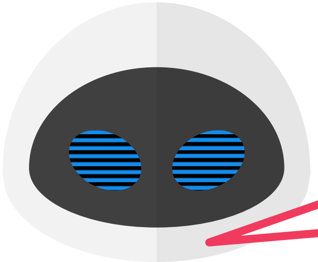




Green gets 1/3 + 1/3 = 2/3 votes whereas Red gets 1/3 votes – Green wins!!

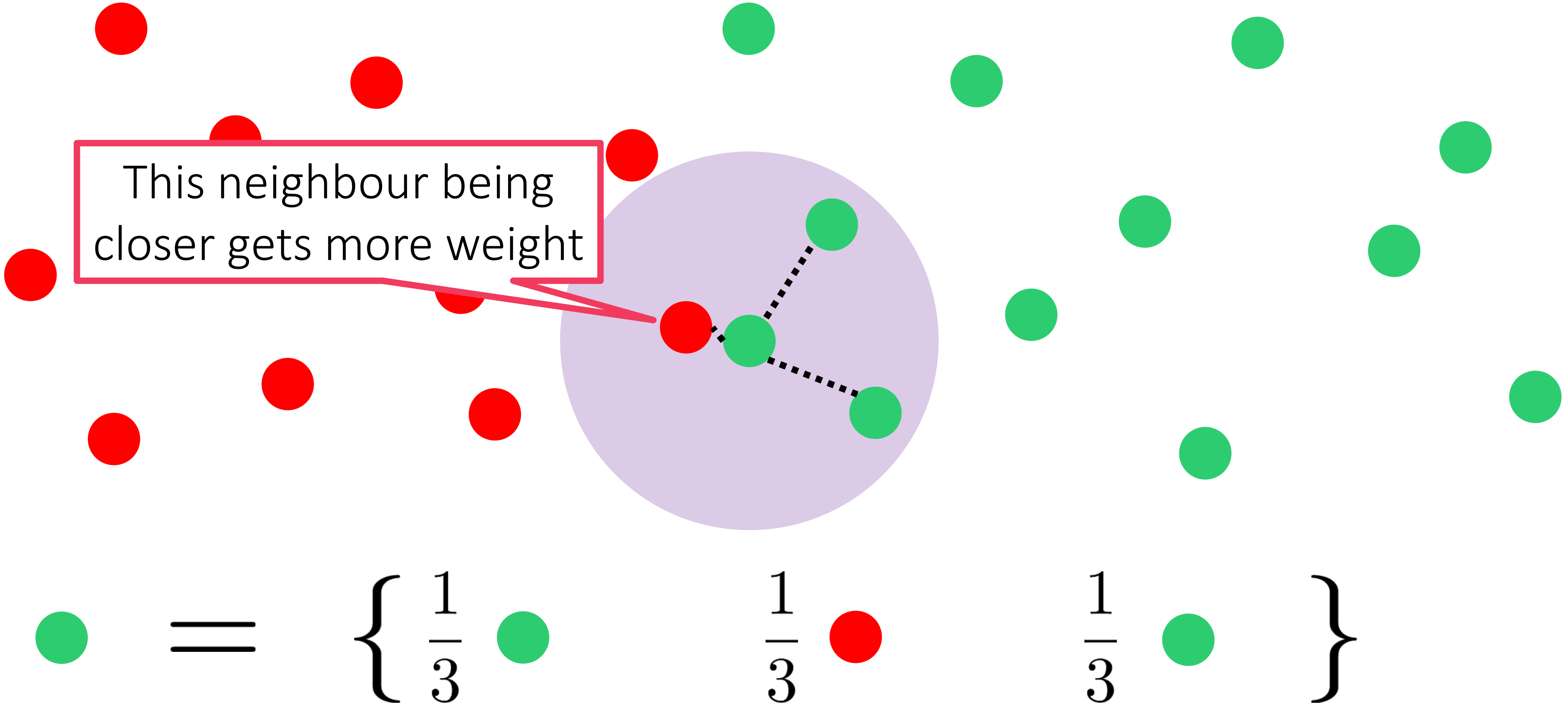
Classification with Weighted rNN

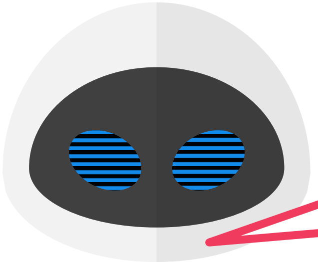




Green gets 1/3 + 1/3 = 2/3 votes whereas
Red gets 1/3 votes – Green wins!!

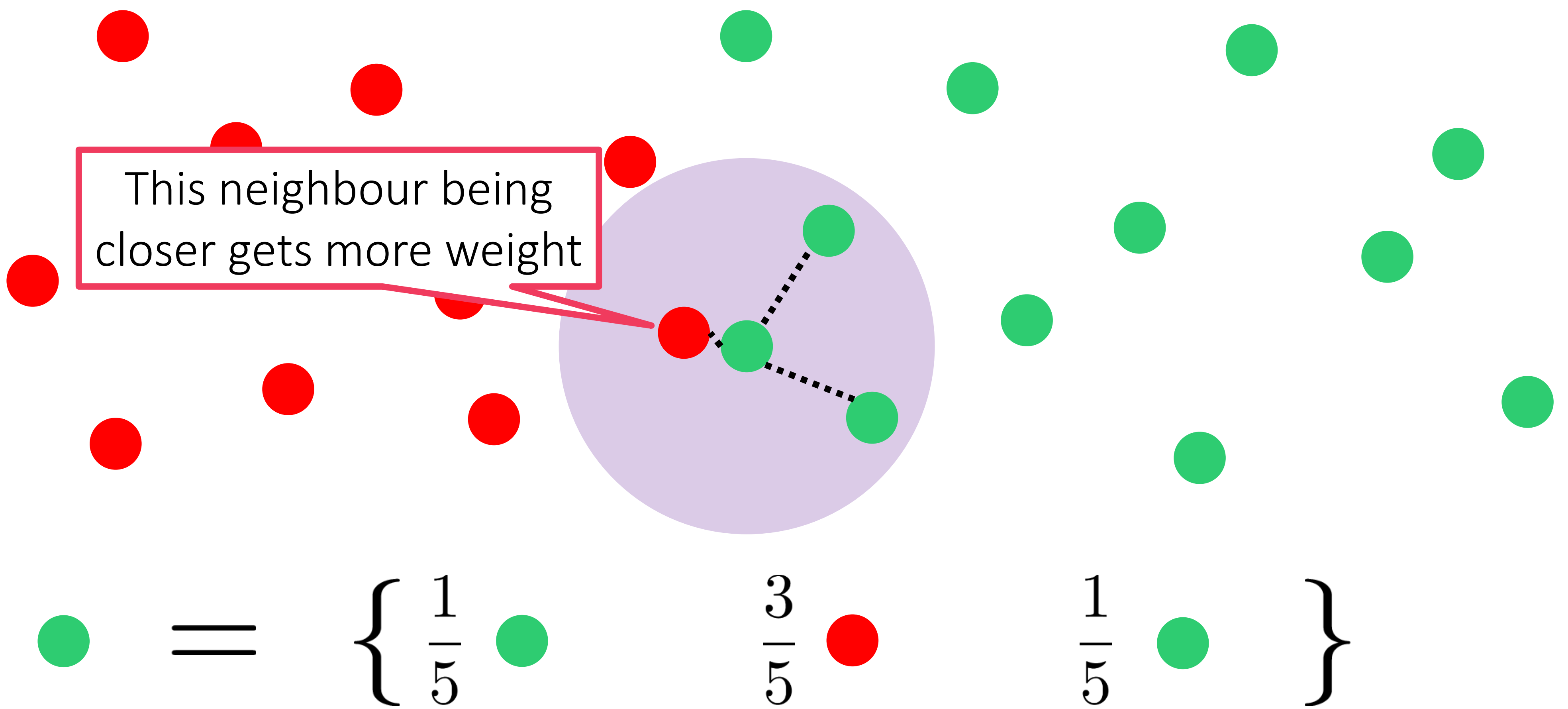
Classification with Weighted rNN

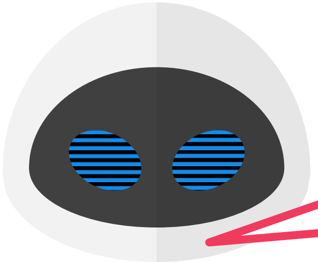




Green gets 1/3 + 1/3 = 2/3 votes whereas Red gets 1/3 votes – Green wins!!

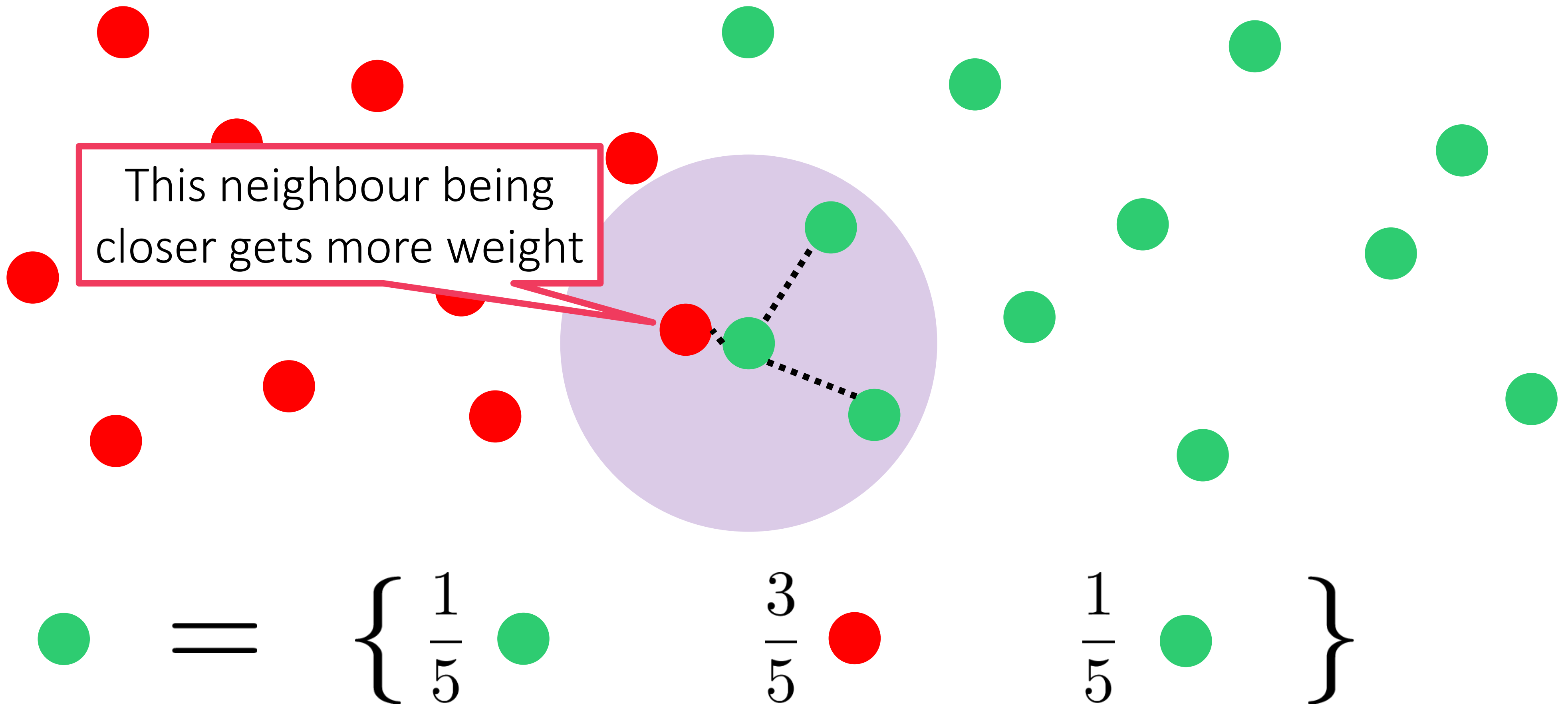
Classification with Weighted rNN

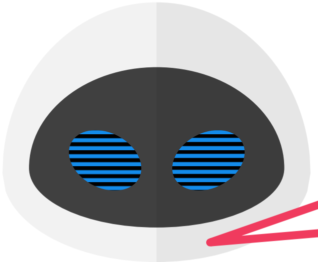




Green gets 1/3 + 1/3 = 2/3 votes whereas Red gets 1/3 votes – Green wins!!

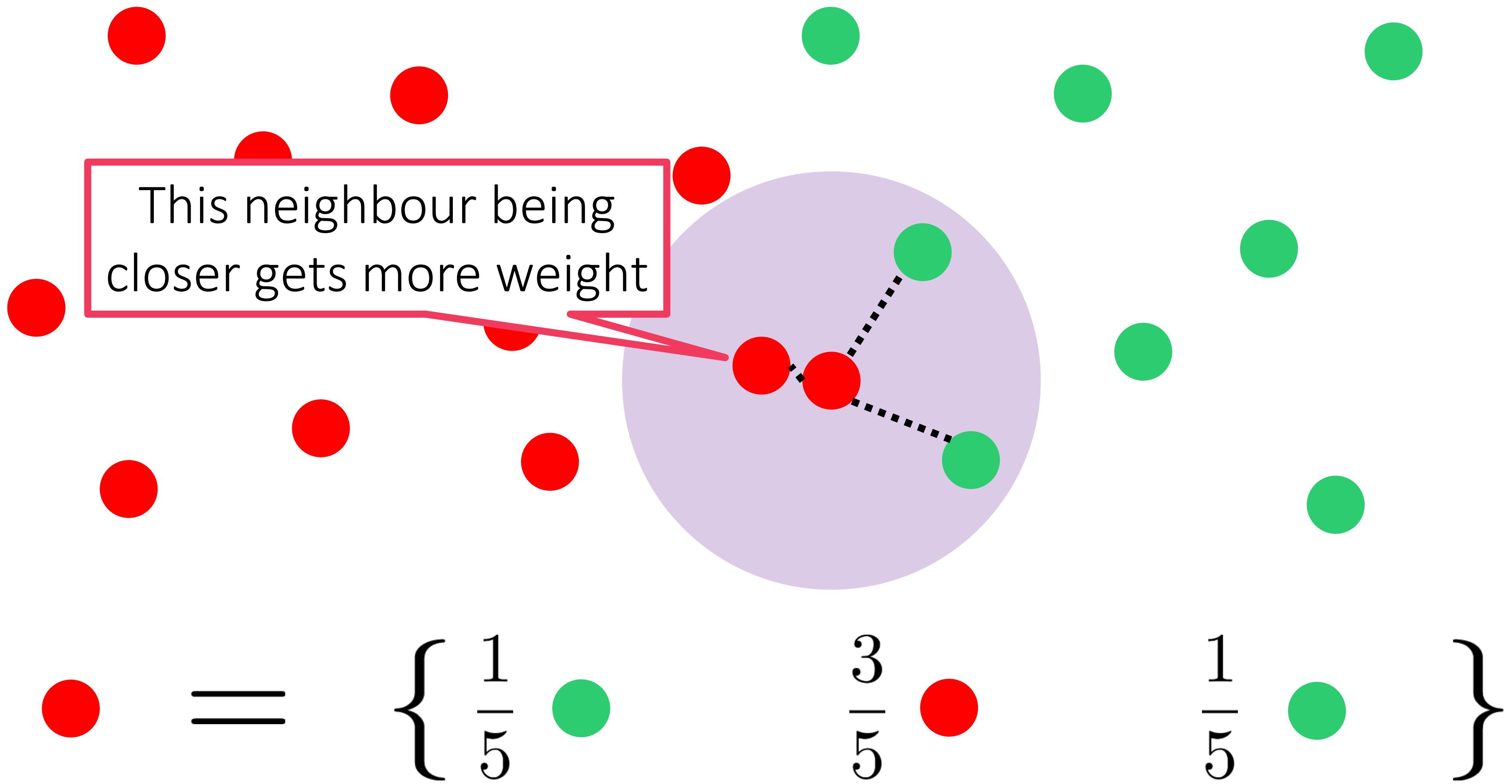
Classification with Weighted rNN

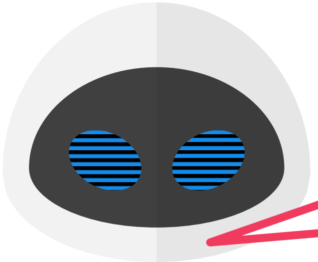




Green gets 1/5 + 1/5 = 2/5 votes whereas Red gets 3/5 votes – Red wins!!

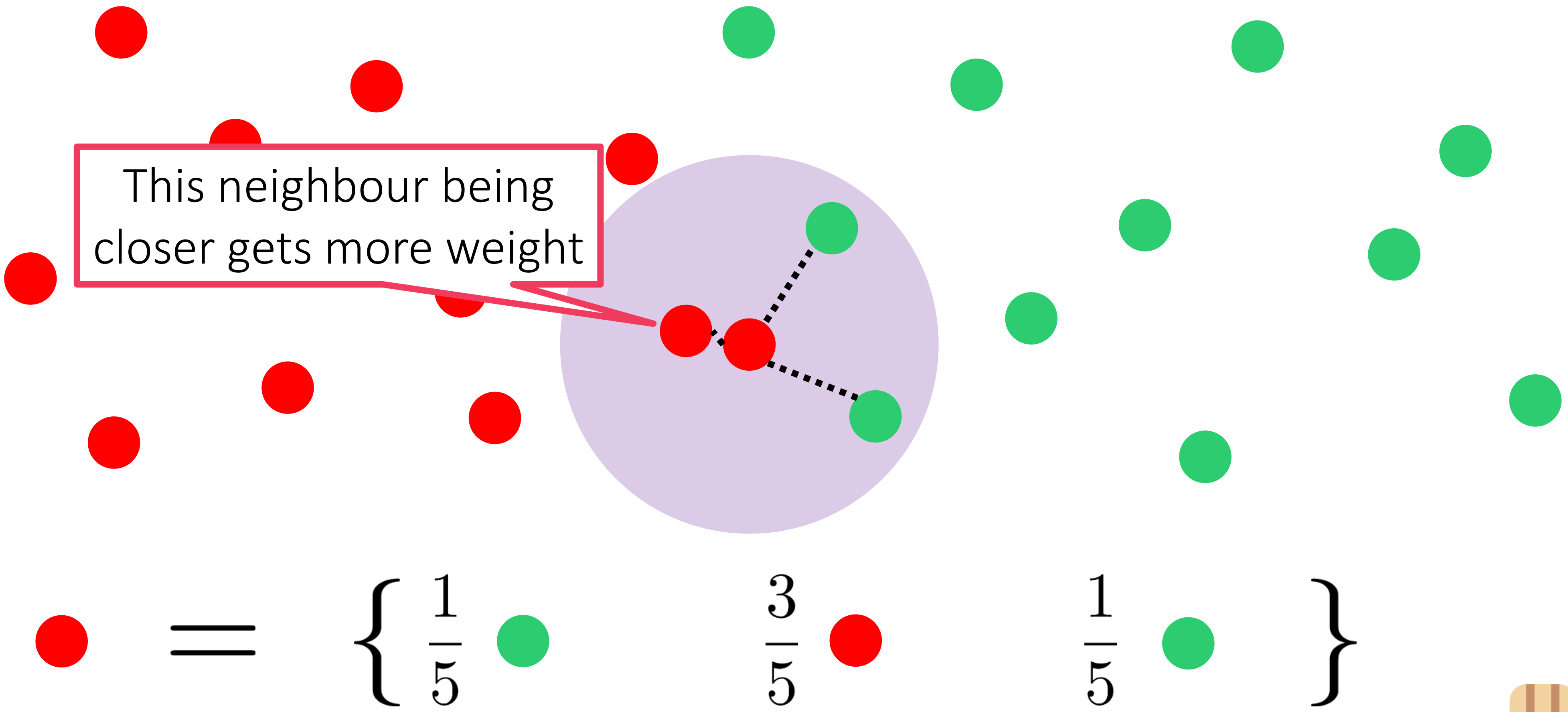
Classification with Weighted rNN

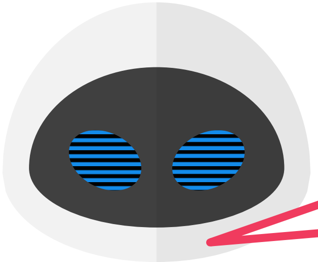




Green gets 1/5 + 1/5 = 2/5 votes whereas Red gets 3/5 votes – Red wins!!

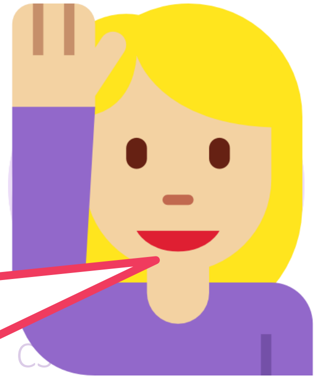
Classification with Weighted rNN





Green gets 1/5 + 1/5 = 2/5 votes whereas Red gets 3/5 votes – Red wins!!

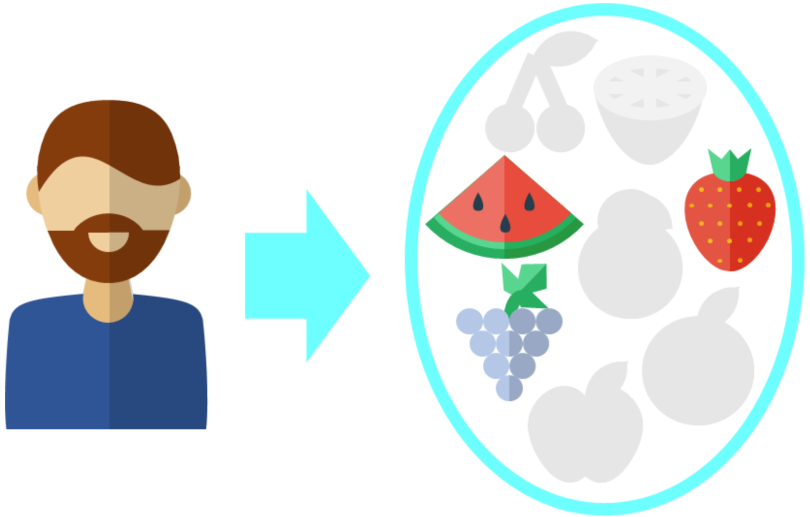
This method elegantly works even if there are more than 2 classes!



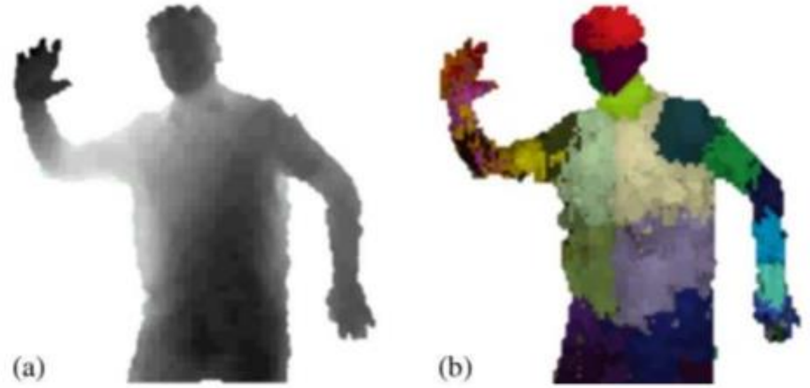
Decision Trees

- Give a *faster* way to perform NN search
- 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 very large ☹
- Can give good train perf. but bad test perf. (known as *overfitting*)

Recommendation Systems



Gaming – Kinect for Xbox 360



SHOTTON ET AL. REAL-TIME HUMAN POSE RECOGNITION IN PARTS FROM A SINGLE DEPTH IMAGE, CVPR 2011

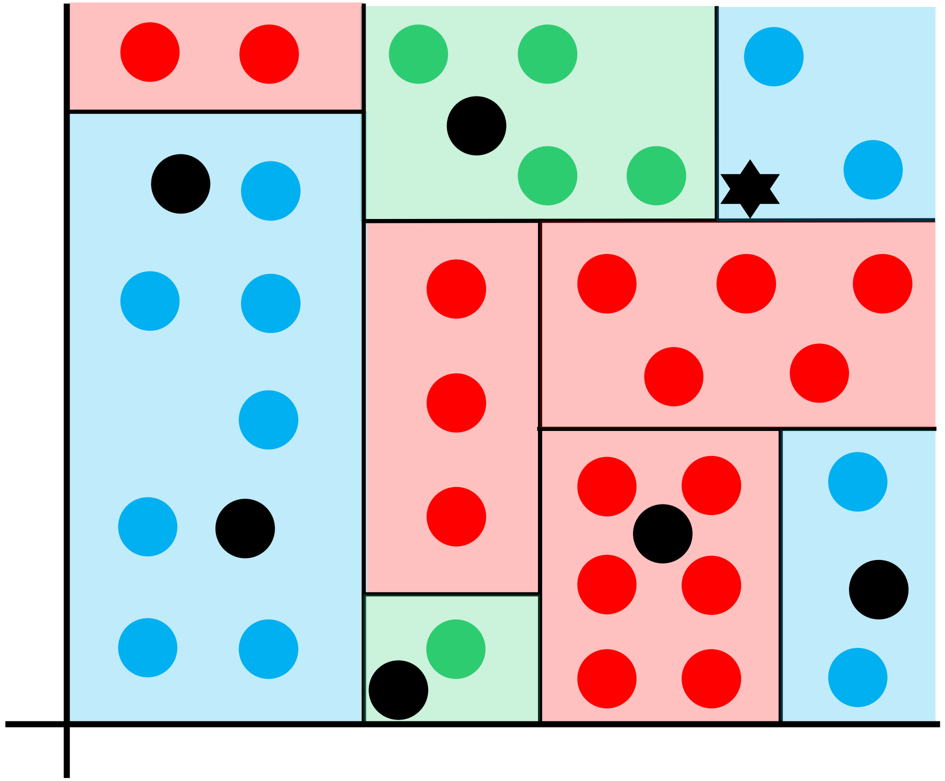
Decision Trees for Classification

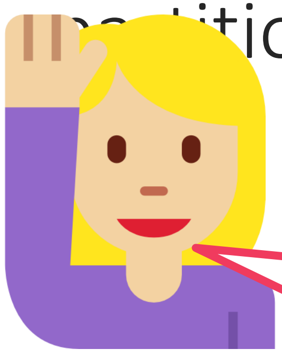
Finding the nearest neighbours can be slow

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

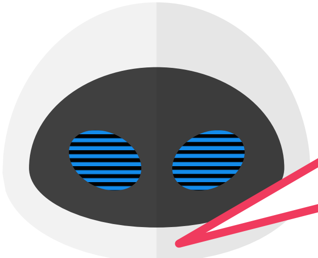
For test data point, much faster to find out which partition is it in which it lies

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

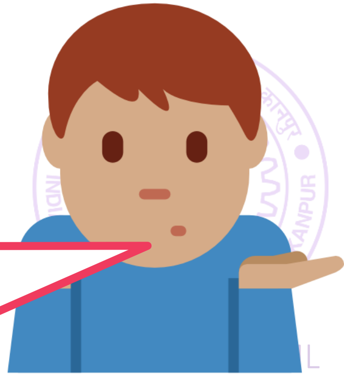




However, if your partitions are fine enough, this will happen very rarely and not hurt performance too much!



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 ☹

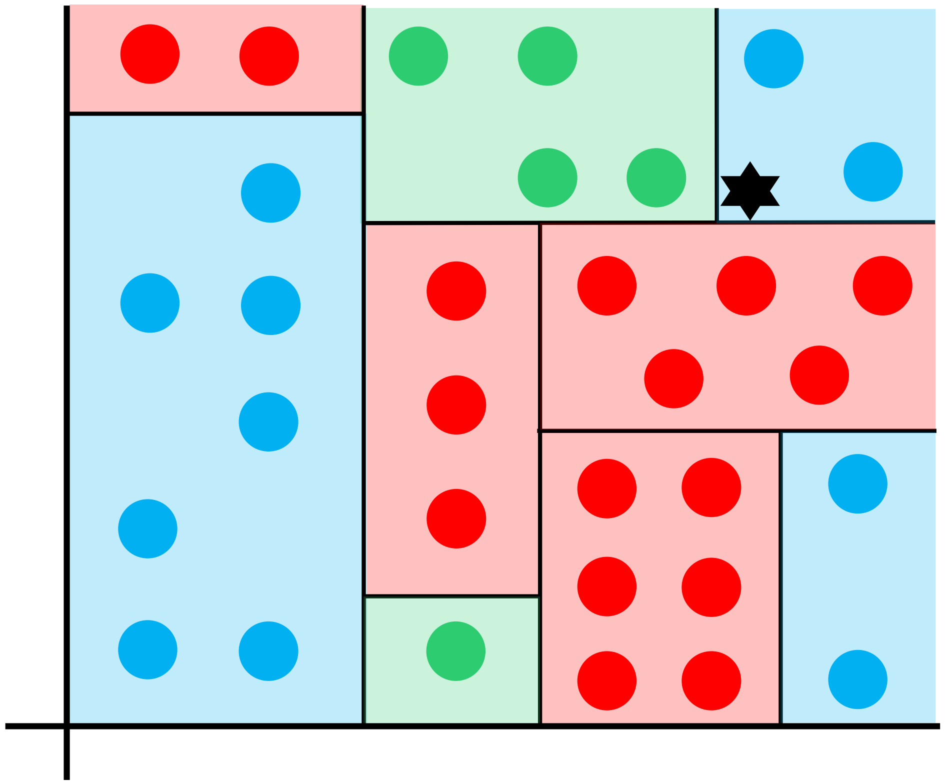
Decision Trees for Classification

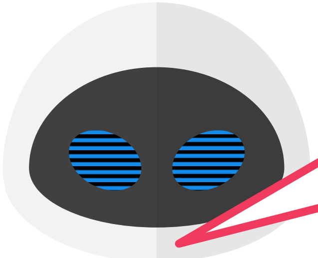
Finding the nearest neighbours can be slow

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

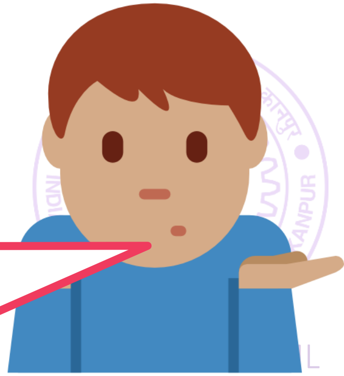
Can consider the other (train) points in that partition as approximate neighbours





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 ☹



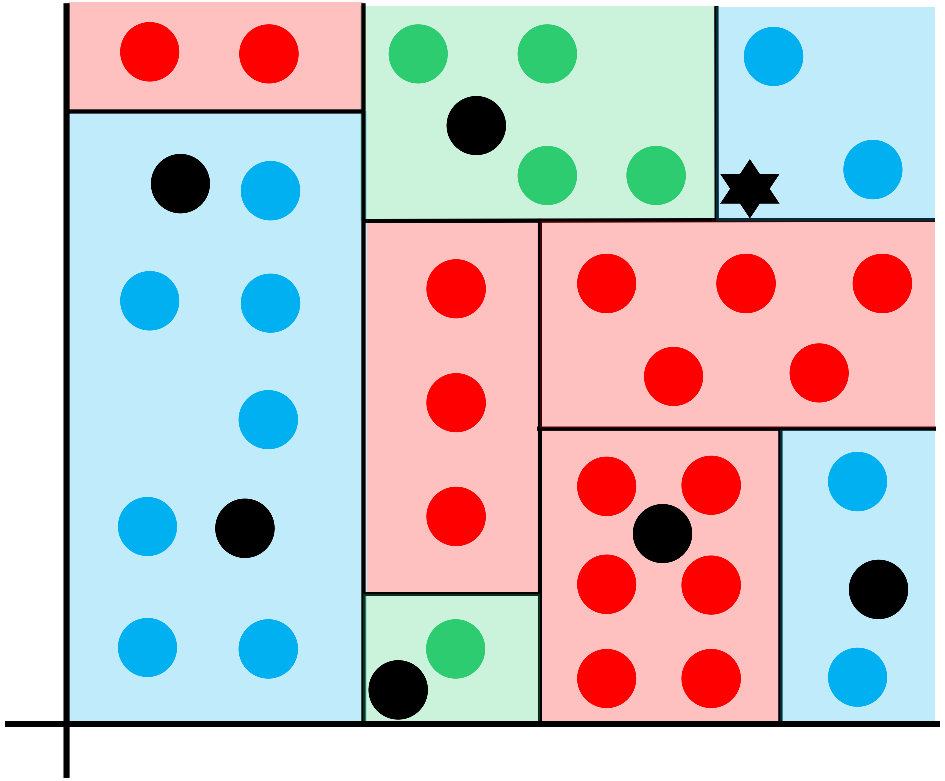
Decision Trees for Classification

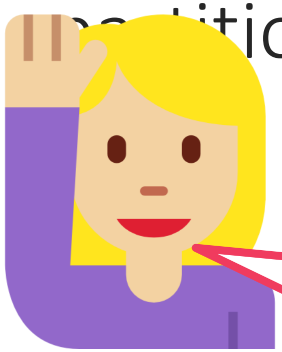
Finding the nearest neighbours can be slow

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

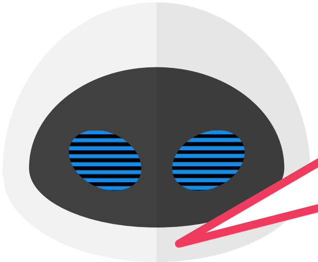
For test data point, much faster to find out which partition is it in which it lies

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



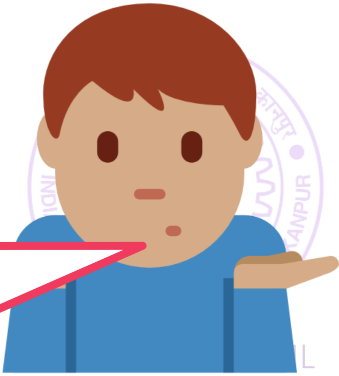


However, if your partitions are fine enough, this will happen very rarely and not hurt performance too much!



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

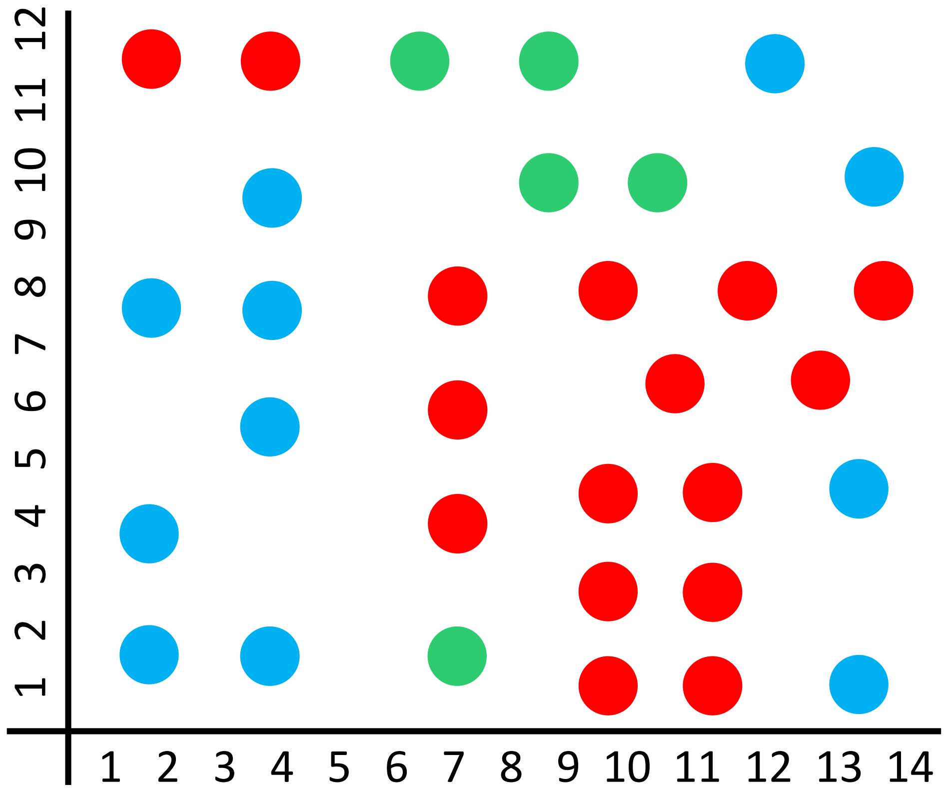
What about points like this? The nearest neighbours are lying in another partition ☹



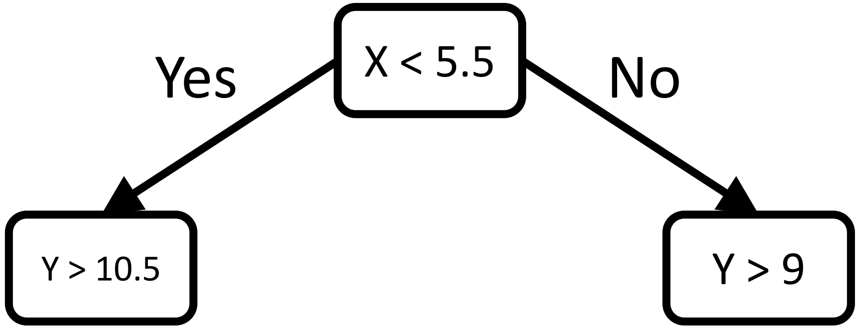
Building Decision Trees

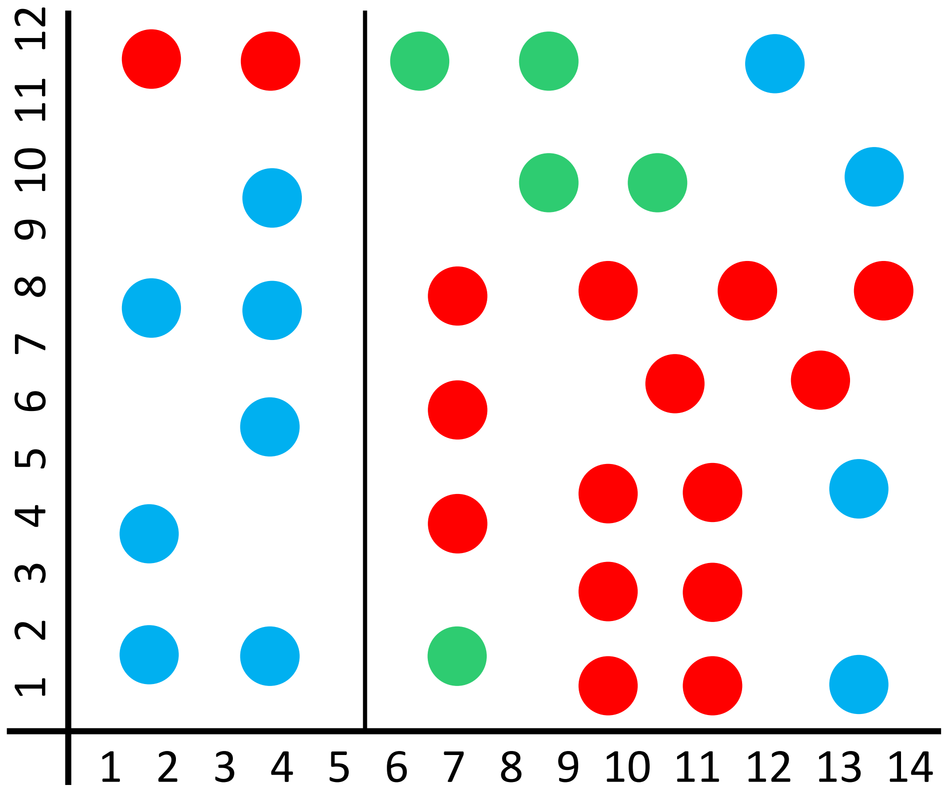
Building Decision Trees

X < 5.5

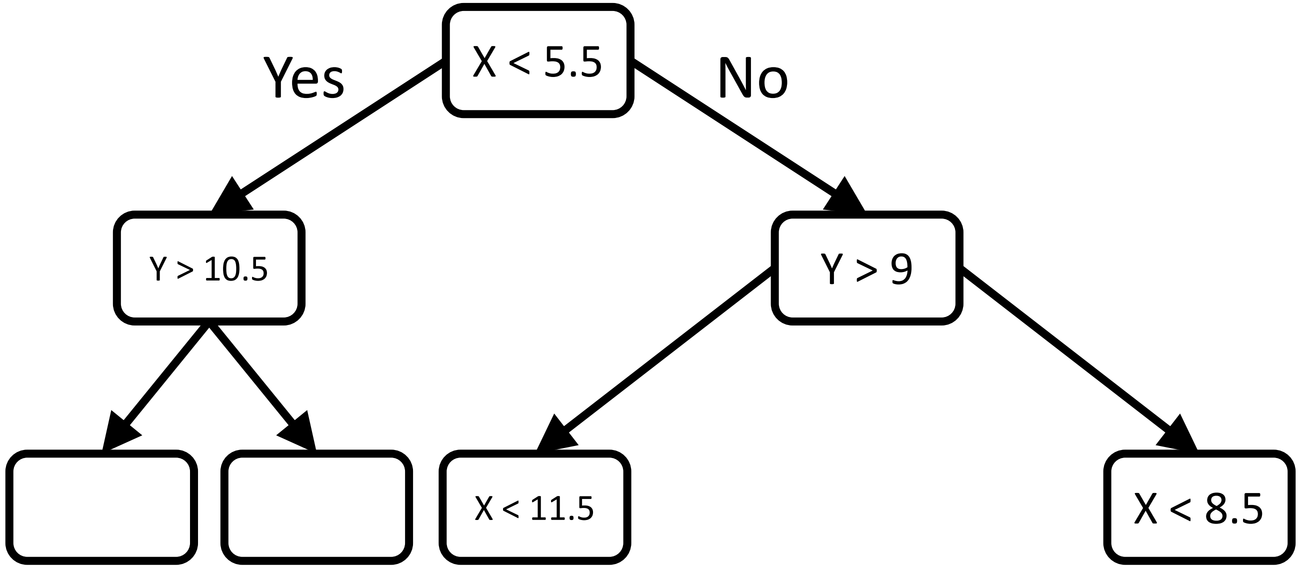


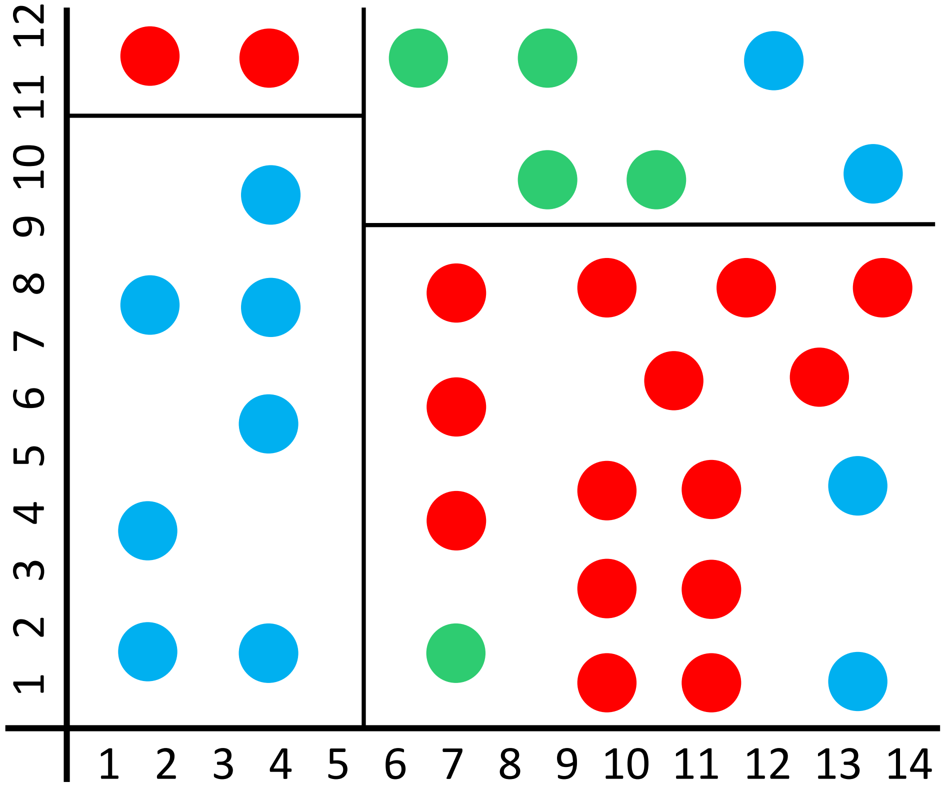
Building Decision Trees



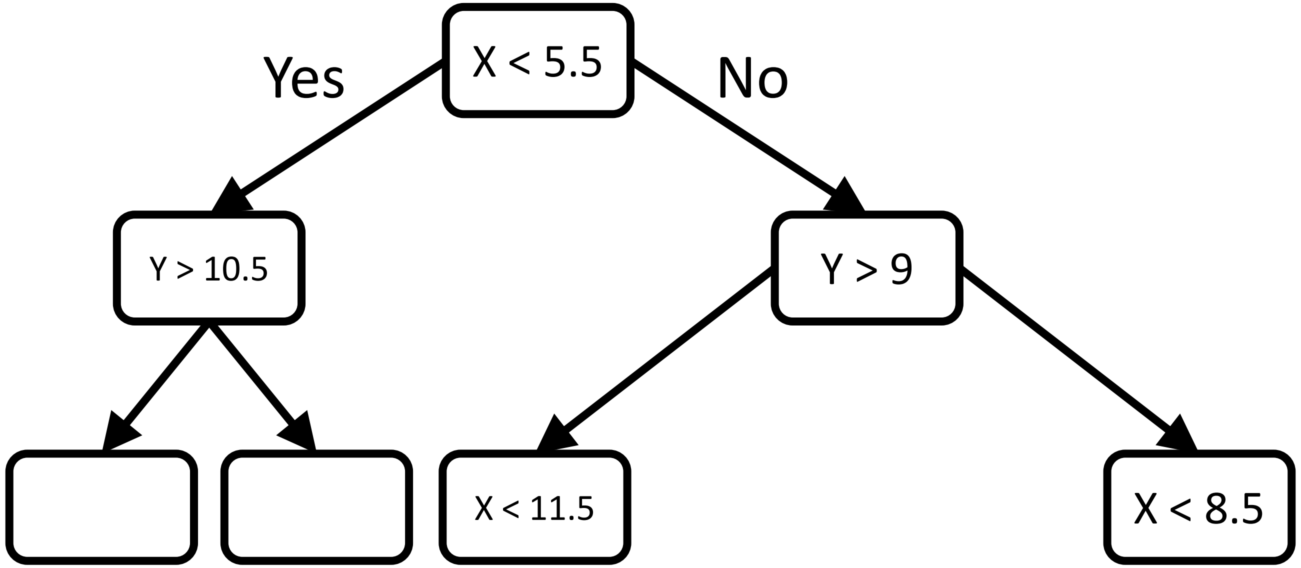


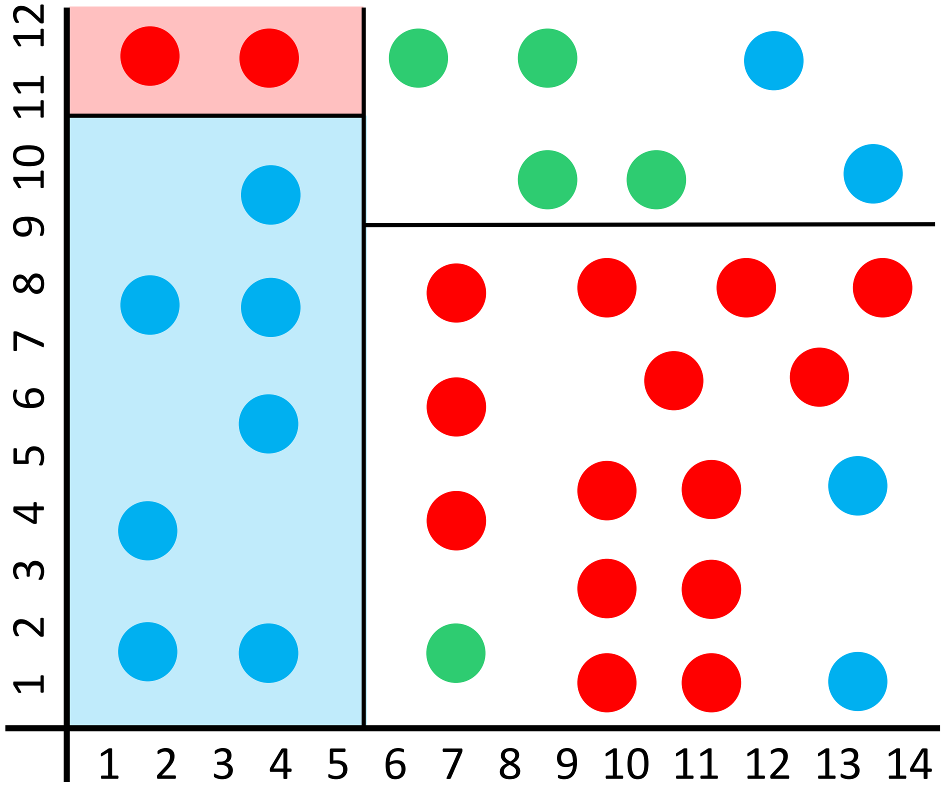
Building Decision Trees



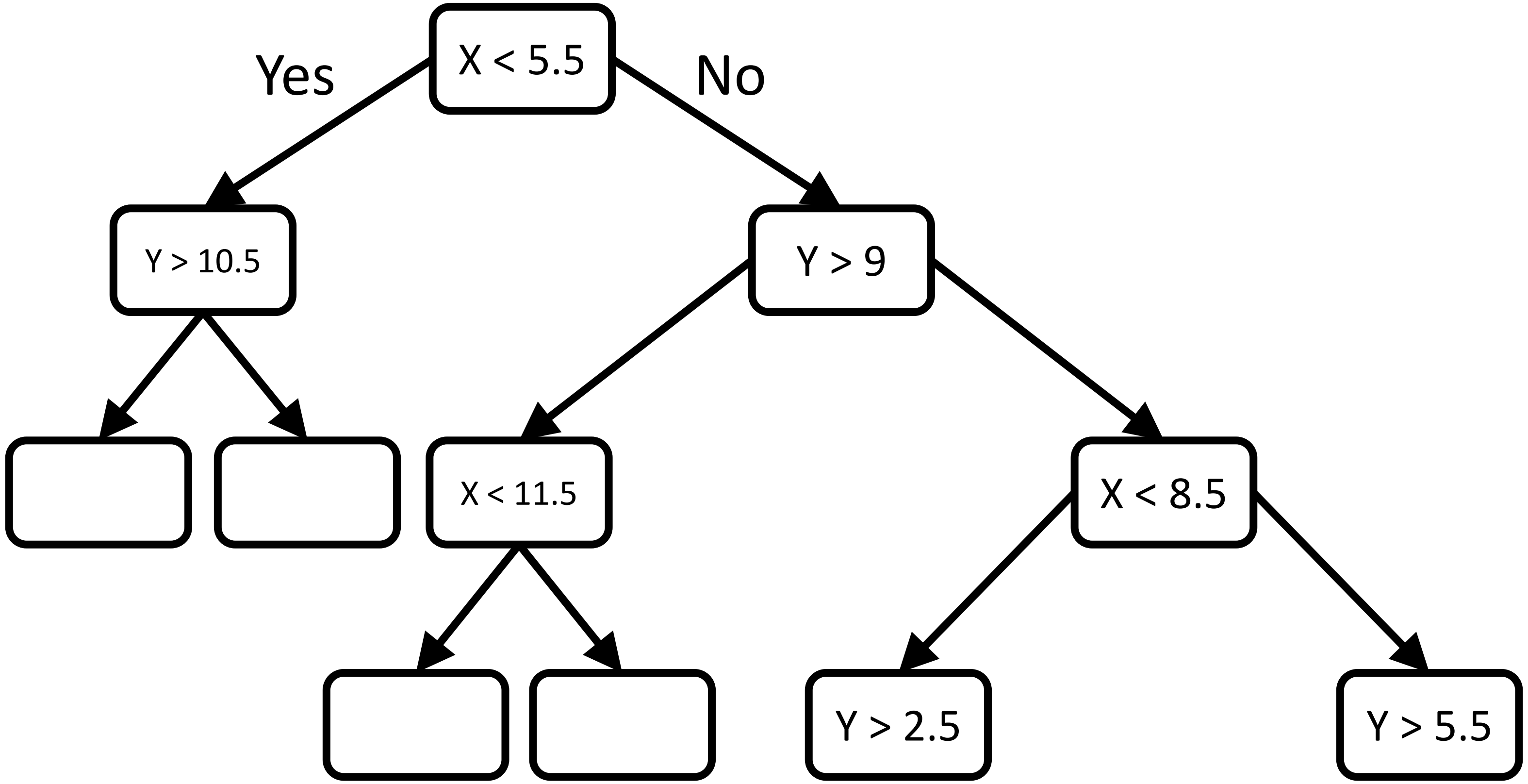


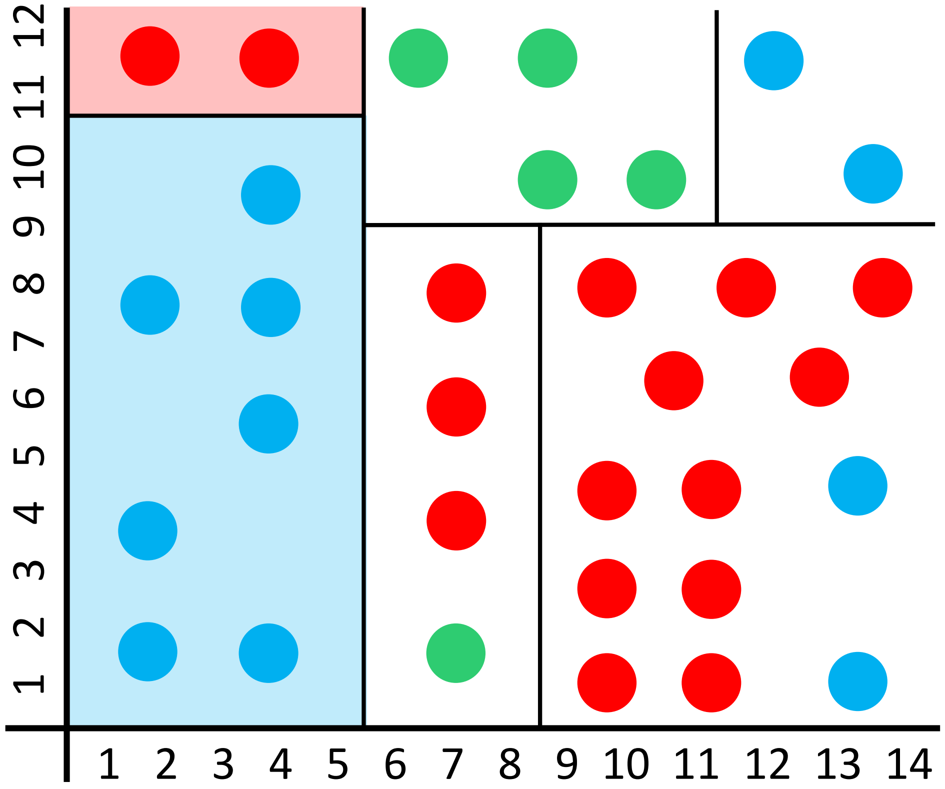
Building Decision Trees



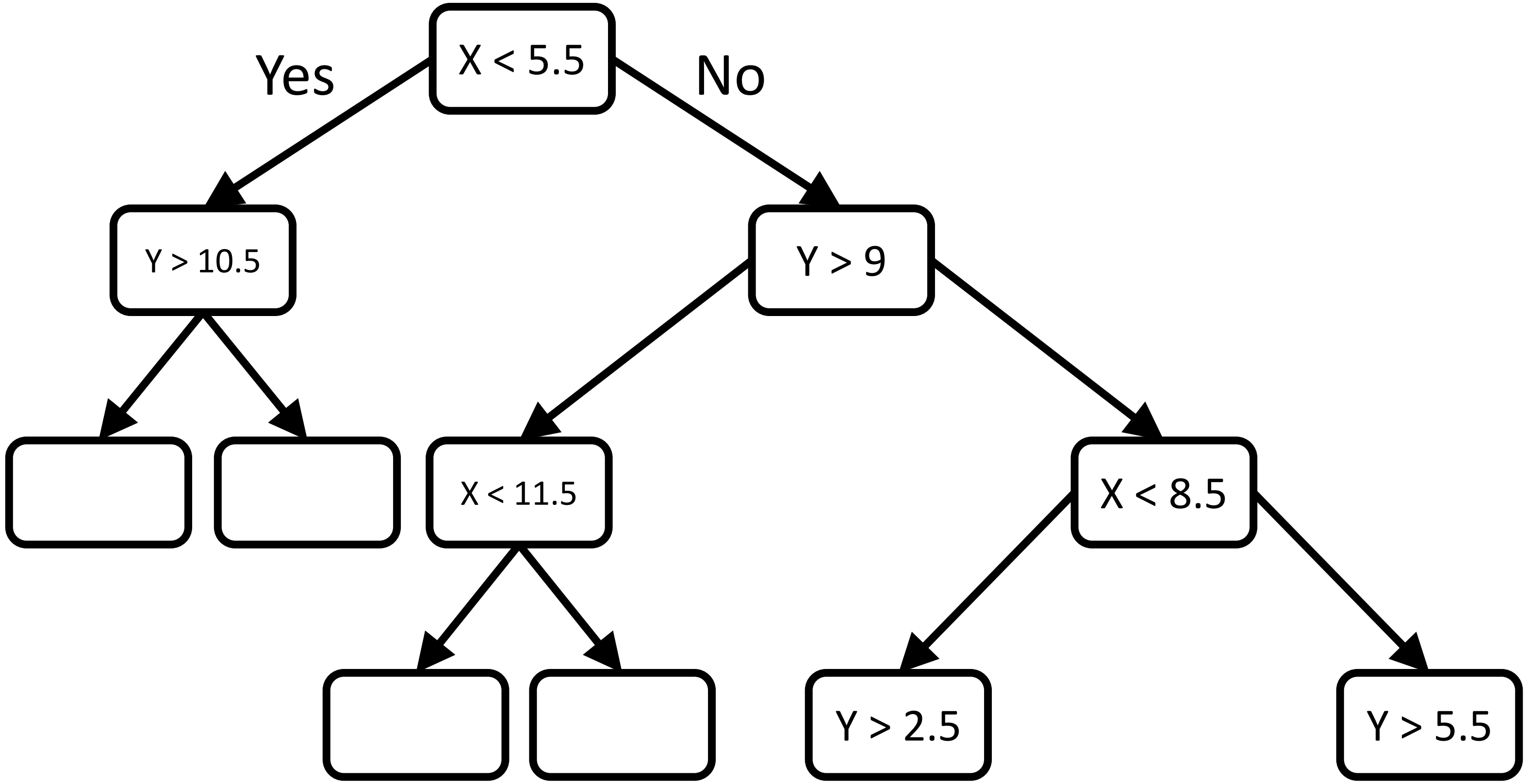


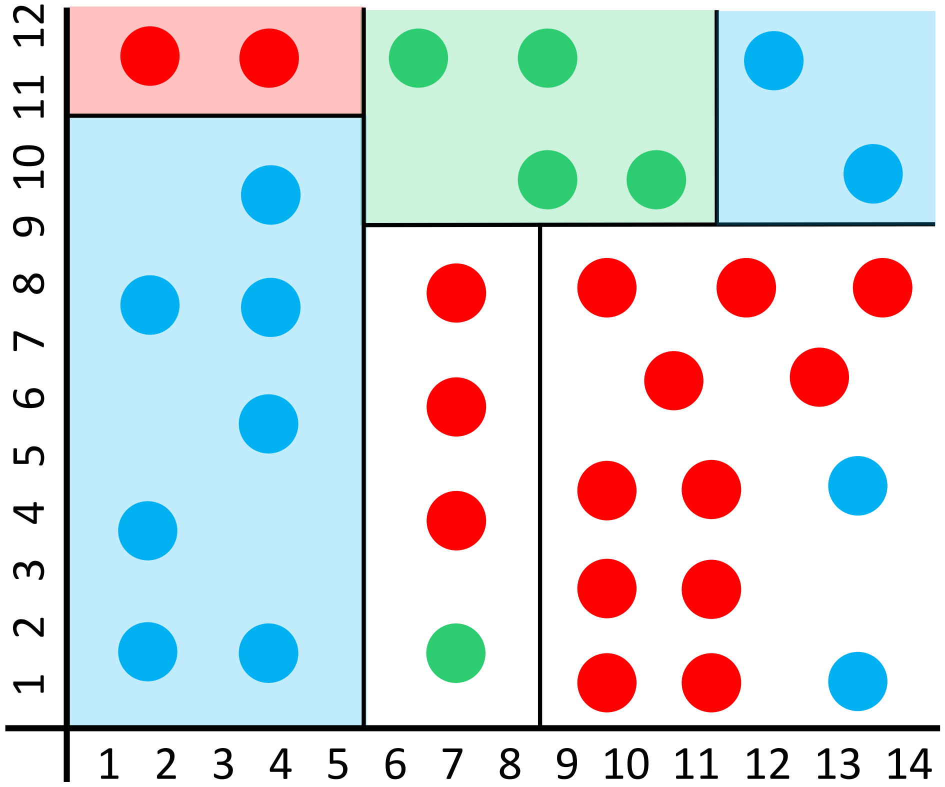
Building Decision Trees



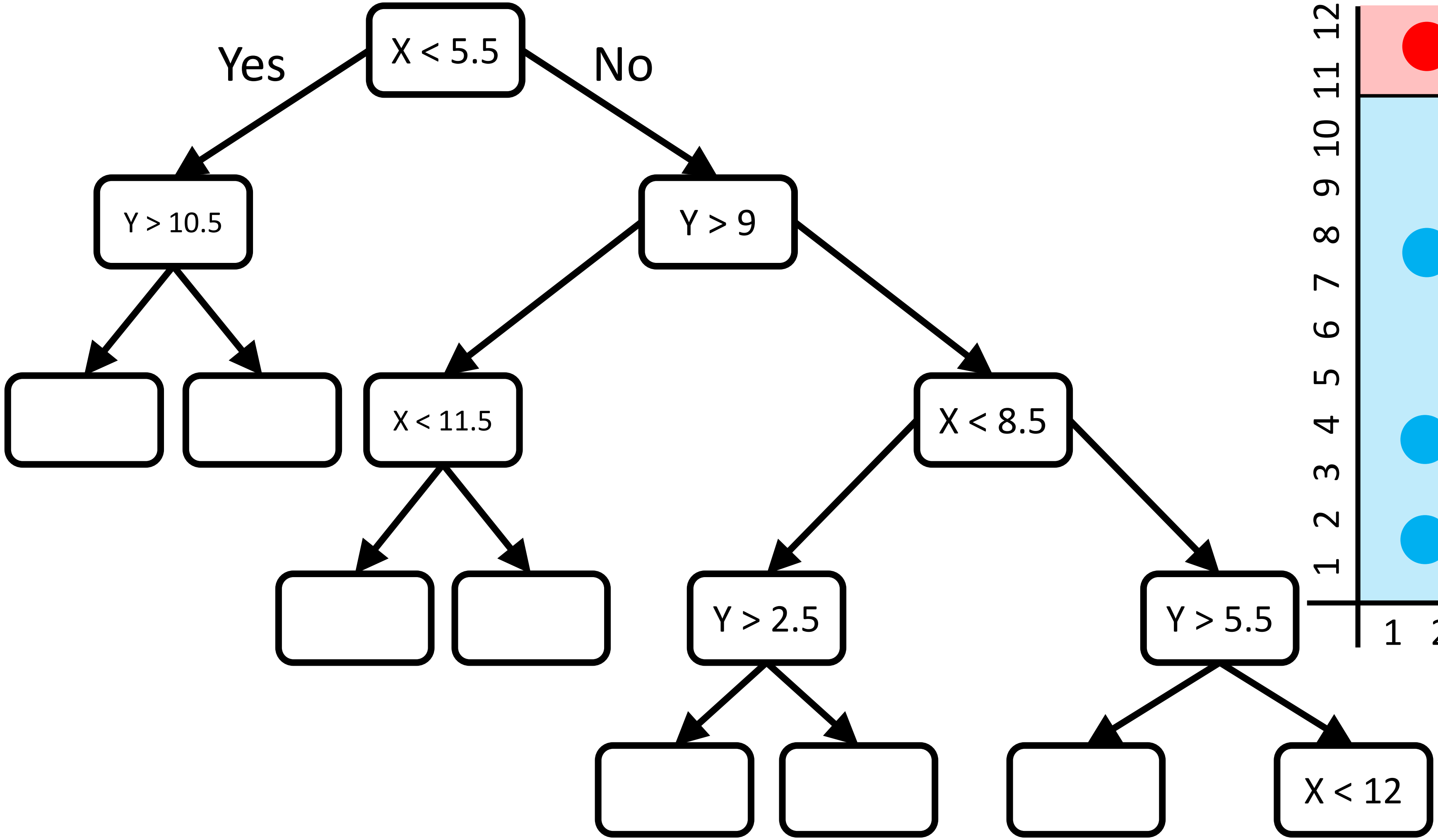


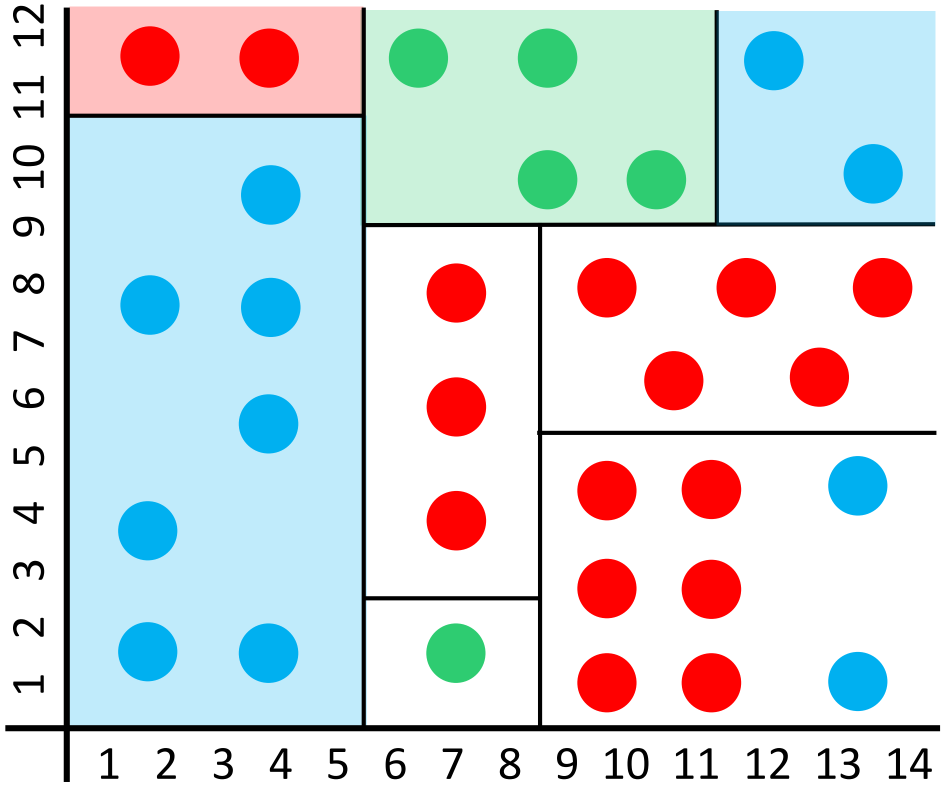
Building Decision Trees



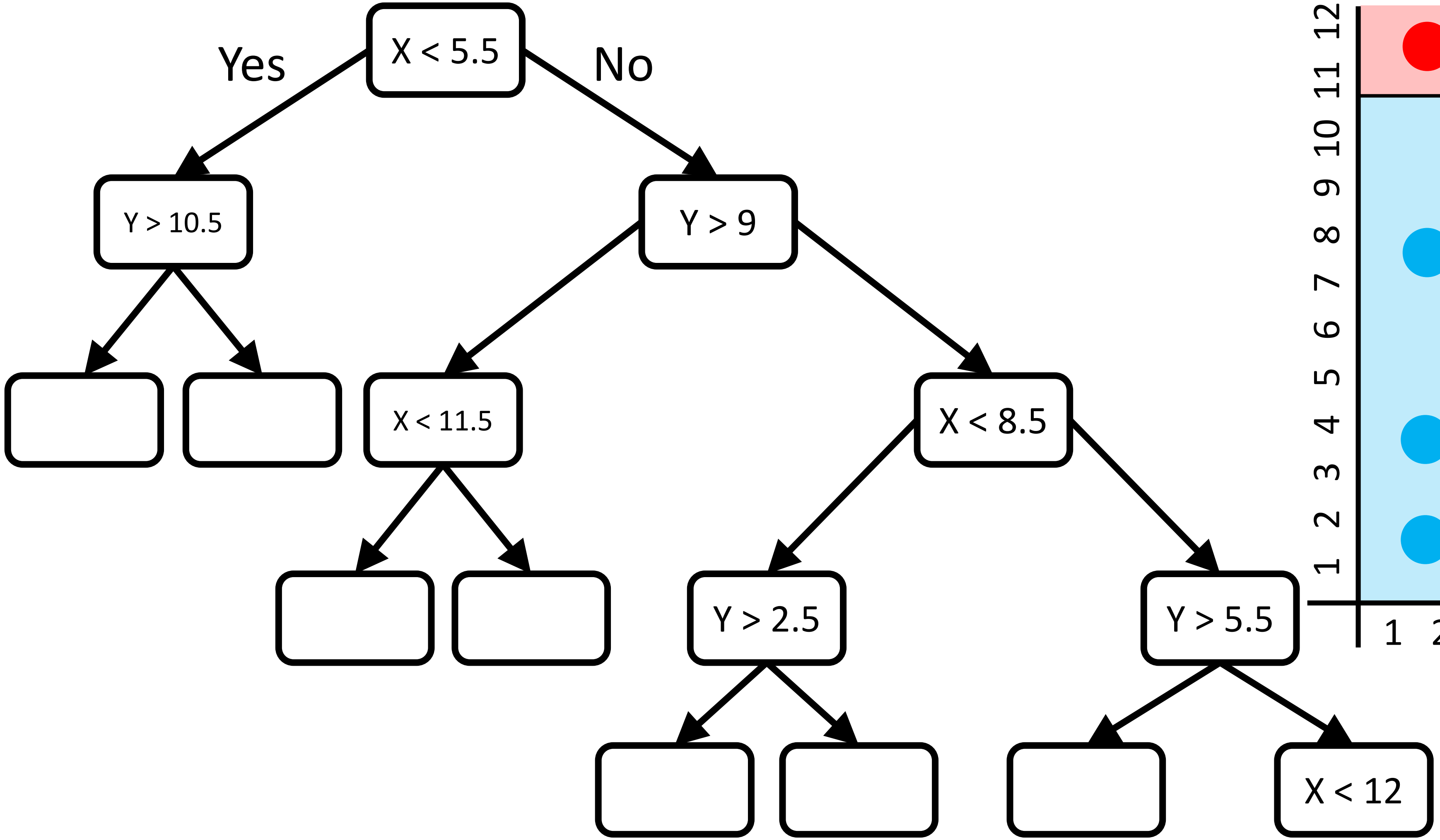


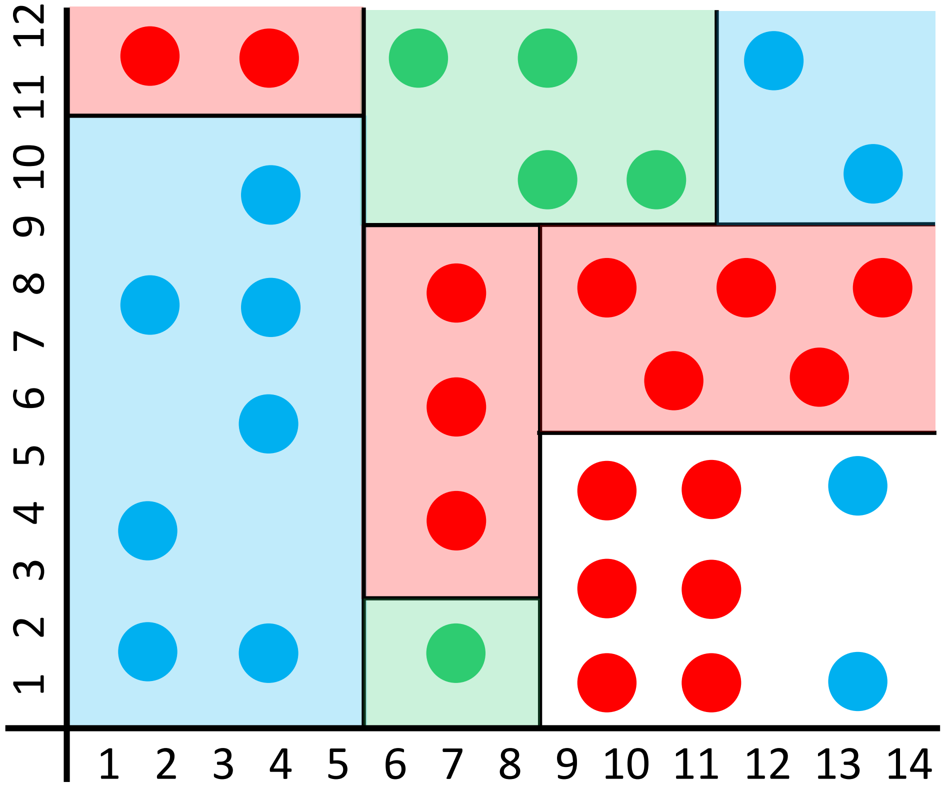
Building Decision Trees



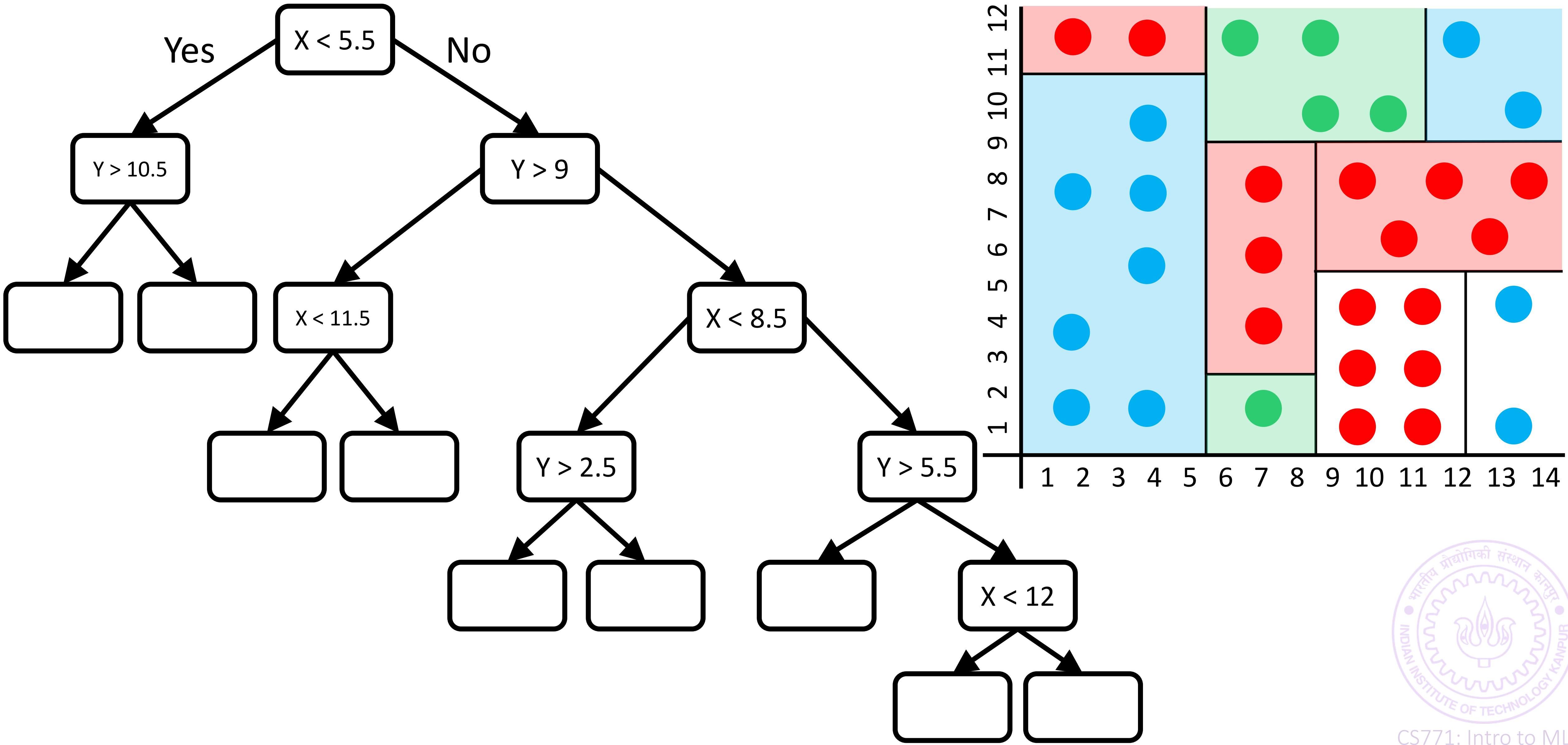


Building Decision Trees

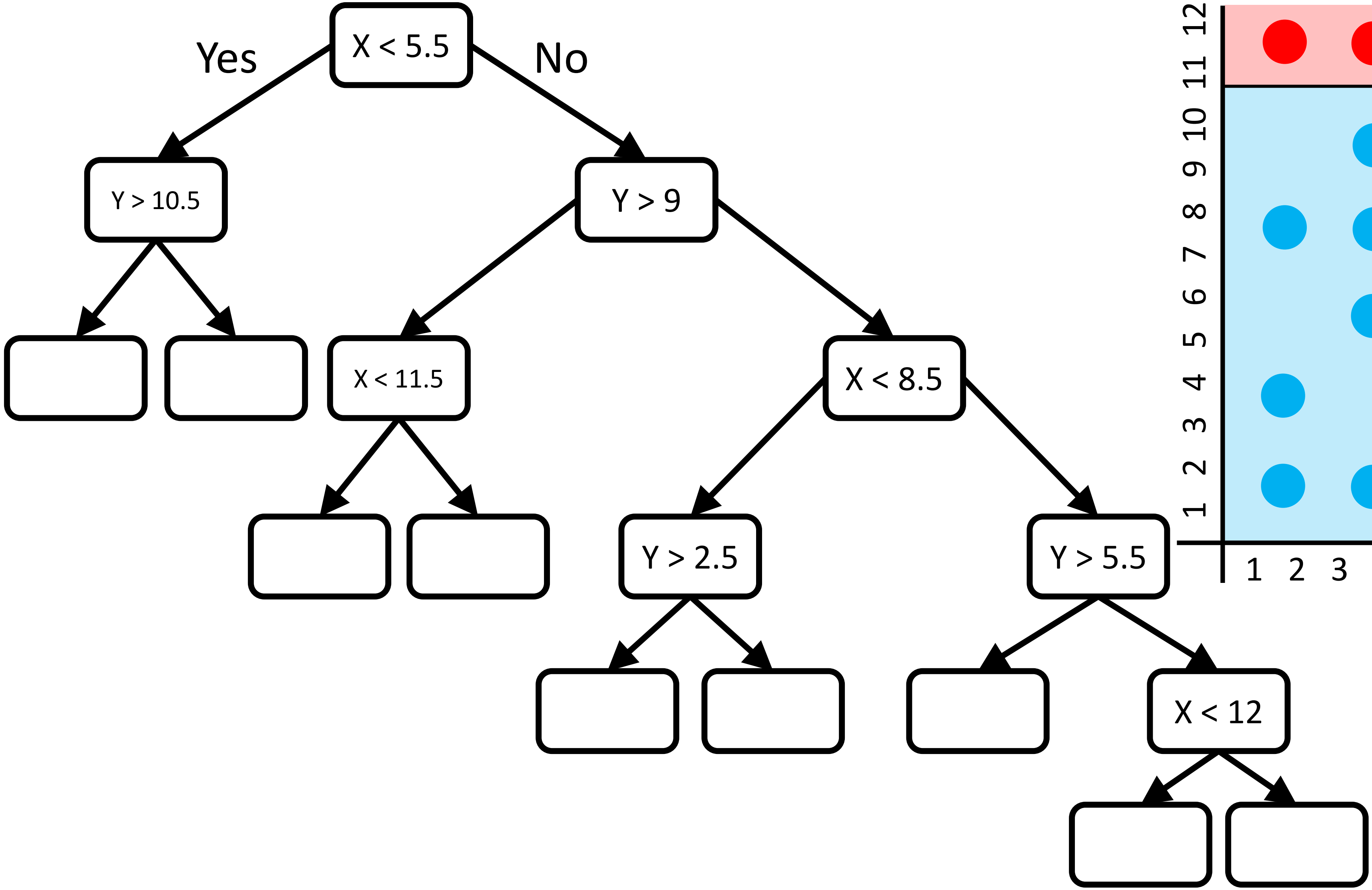


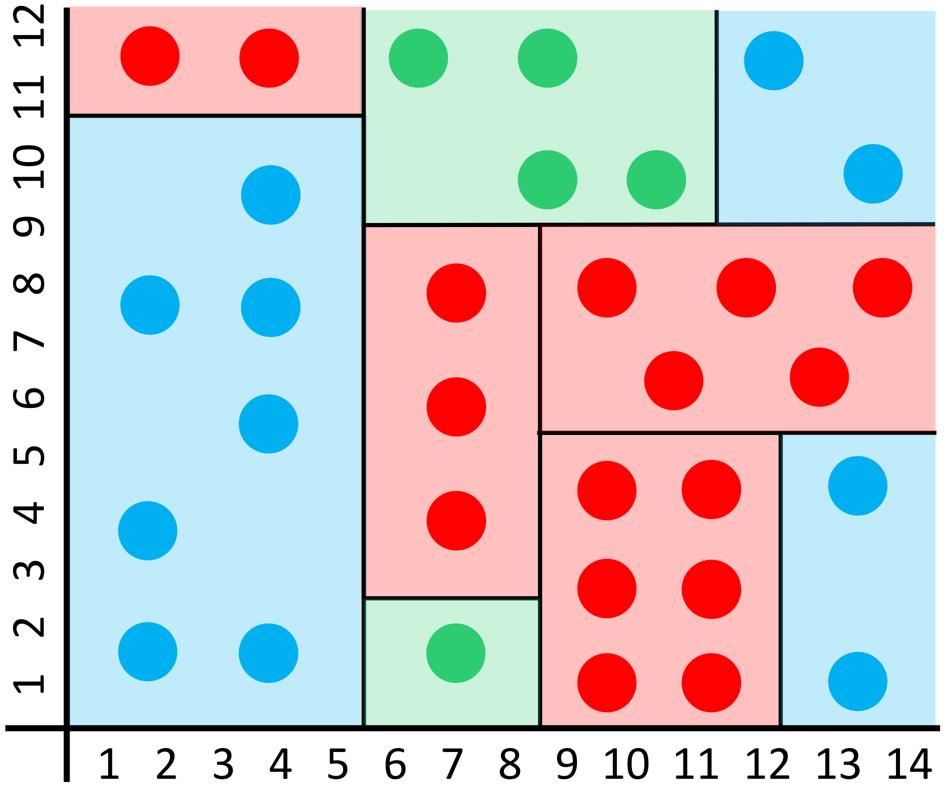


Building Decision Trees

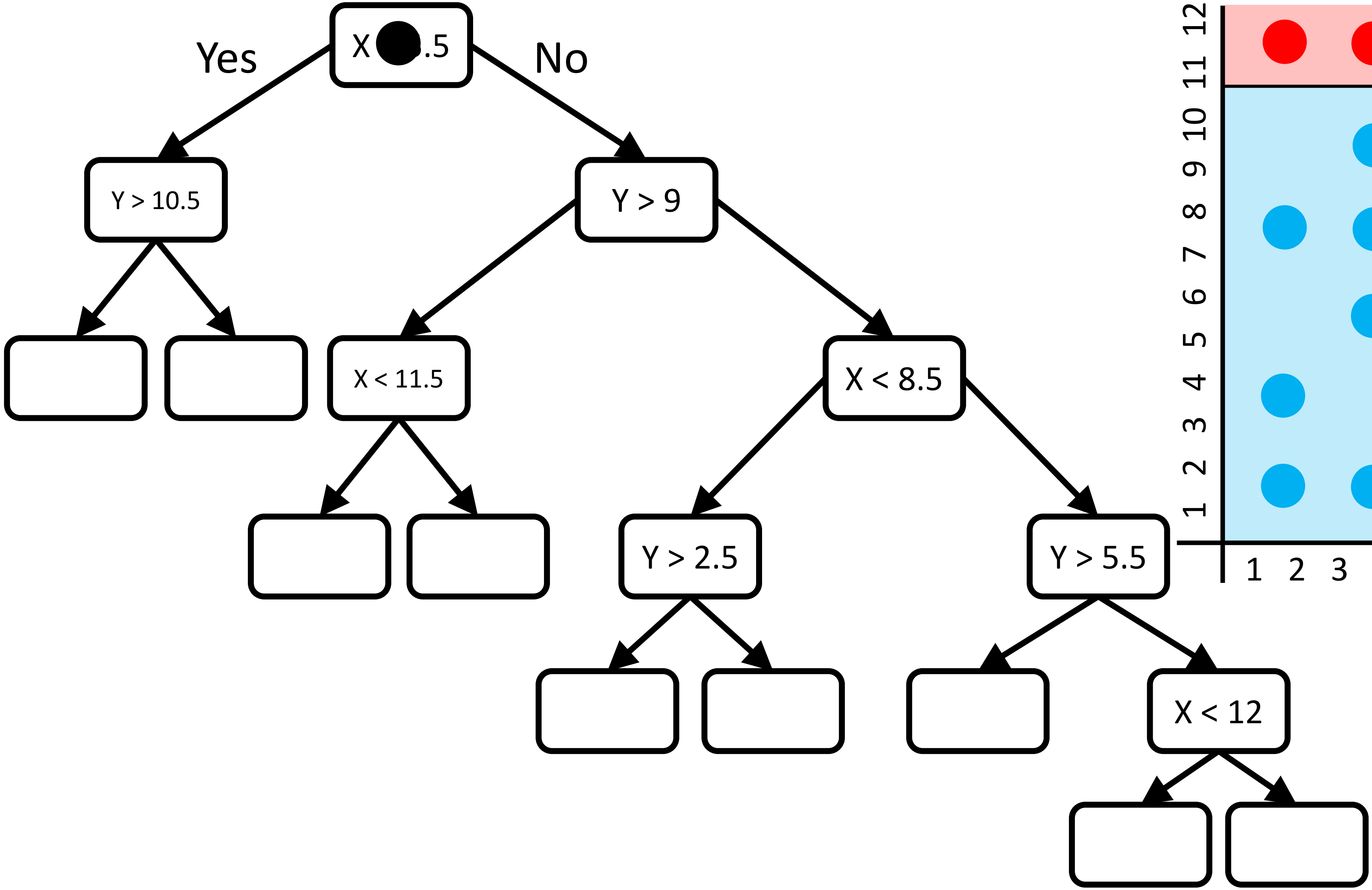


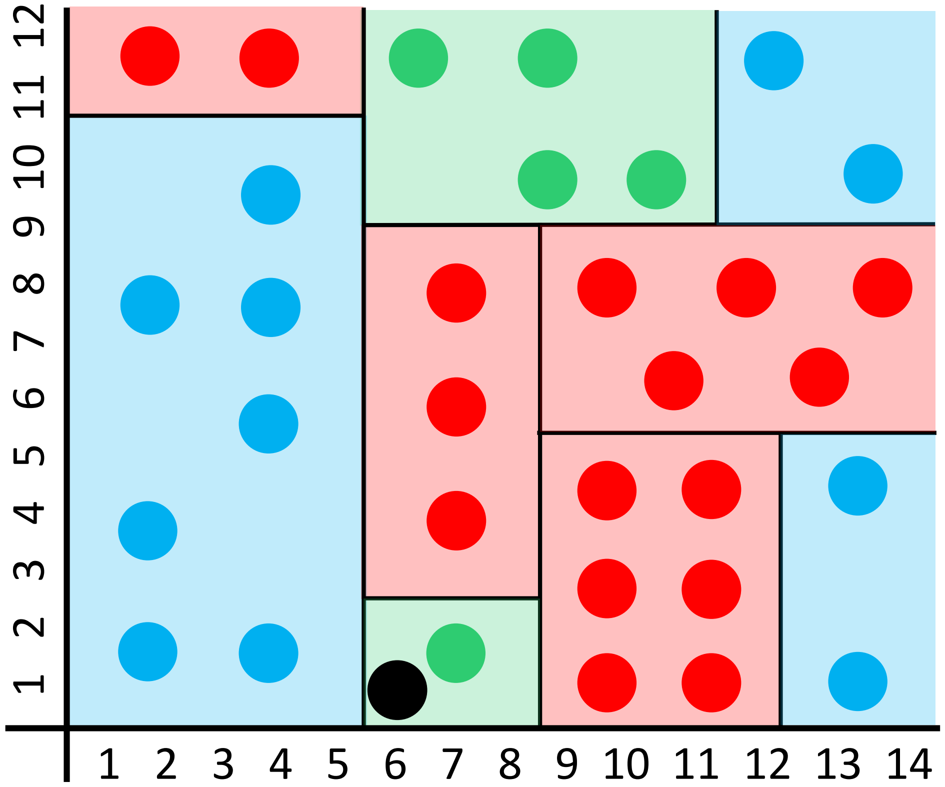
Building Decision Trees



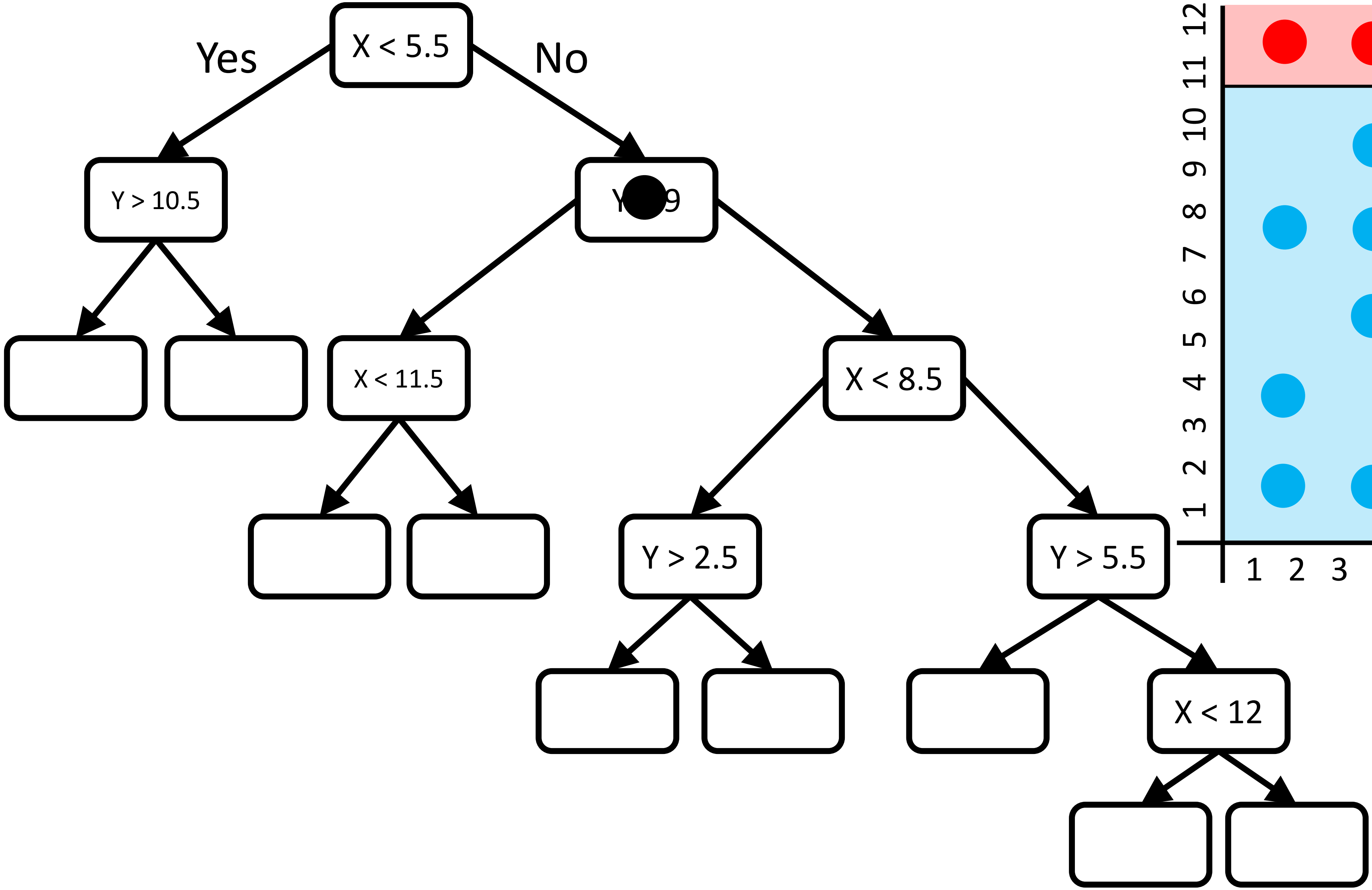


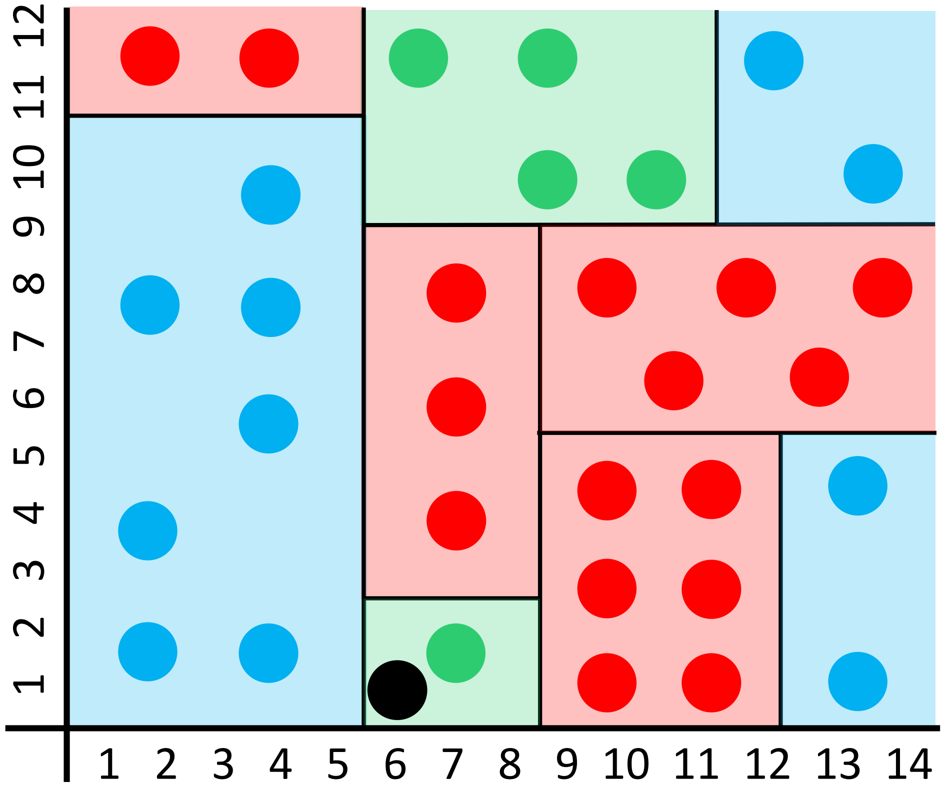
Building Decision Trees



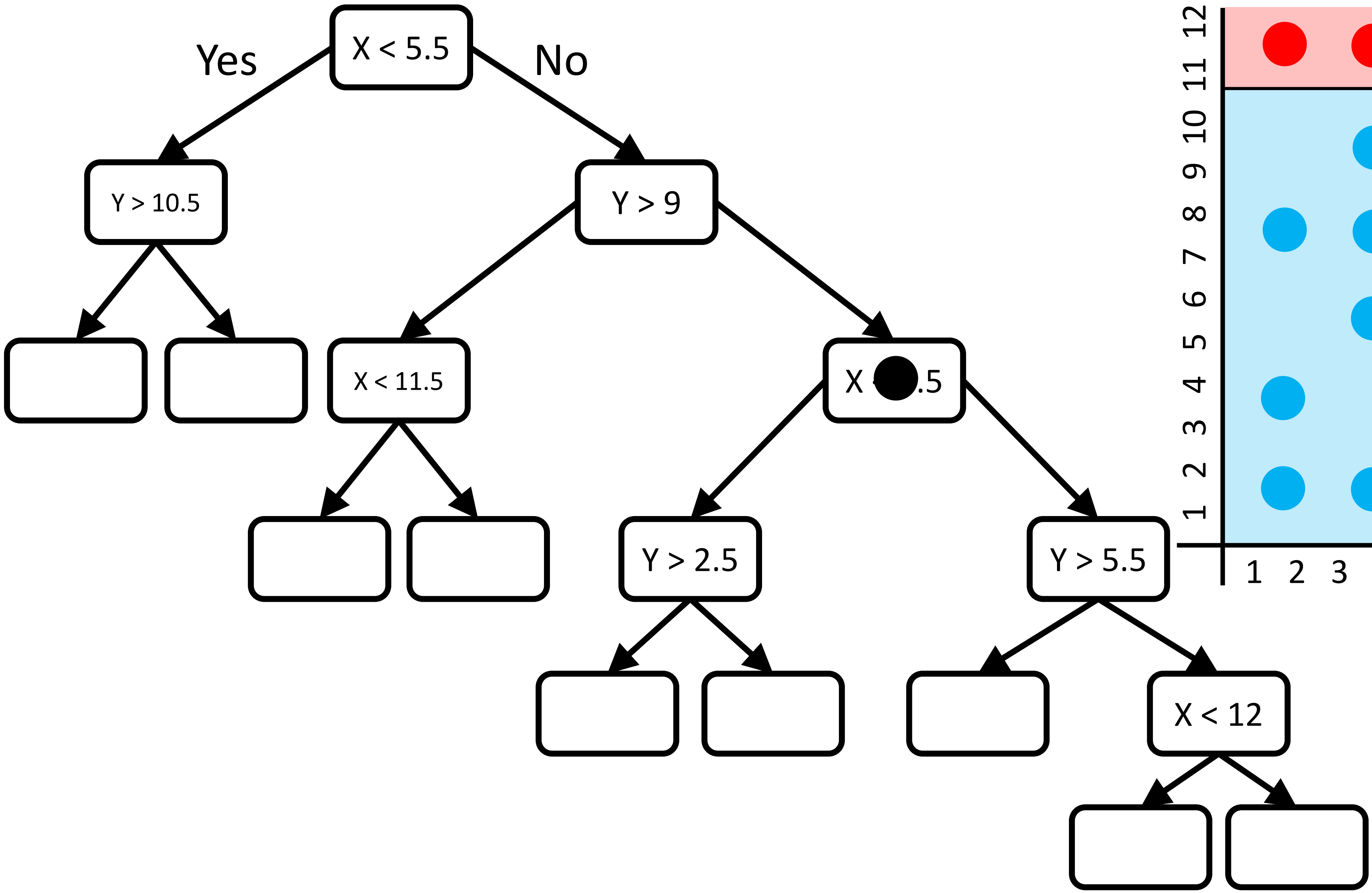


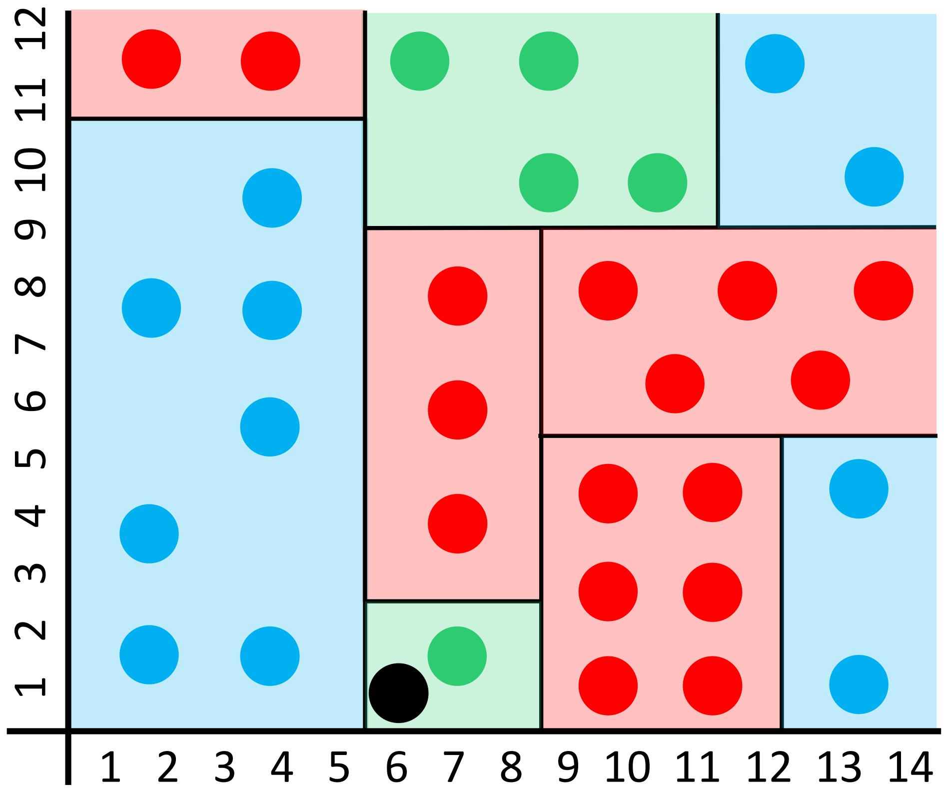
Building Decision Trees



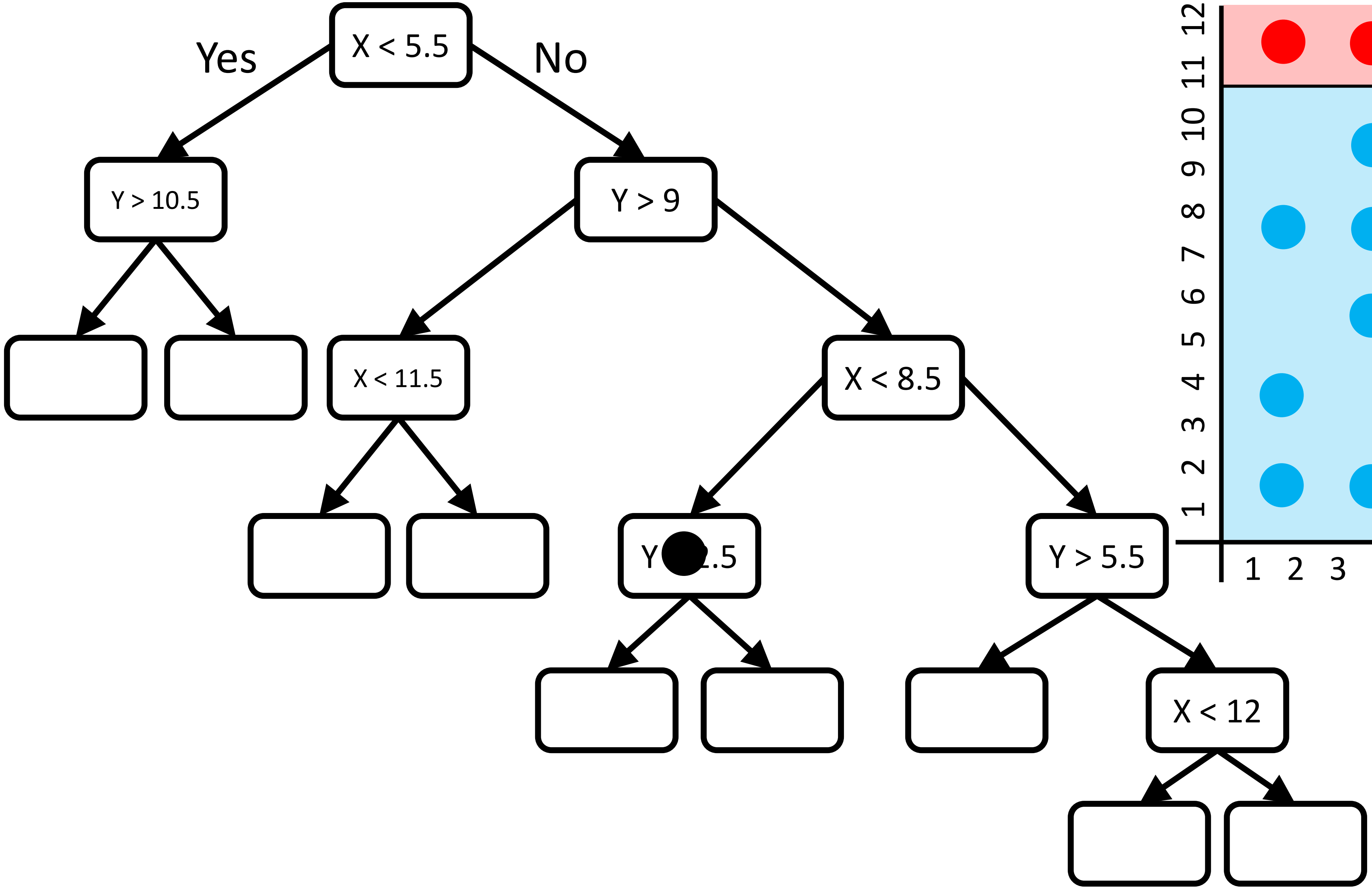


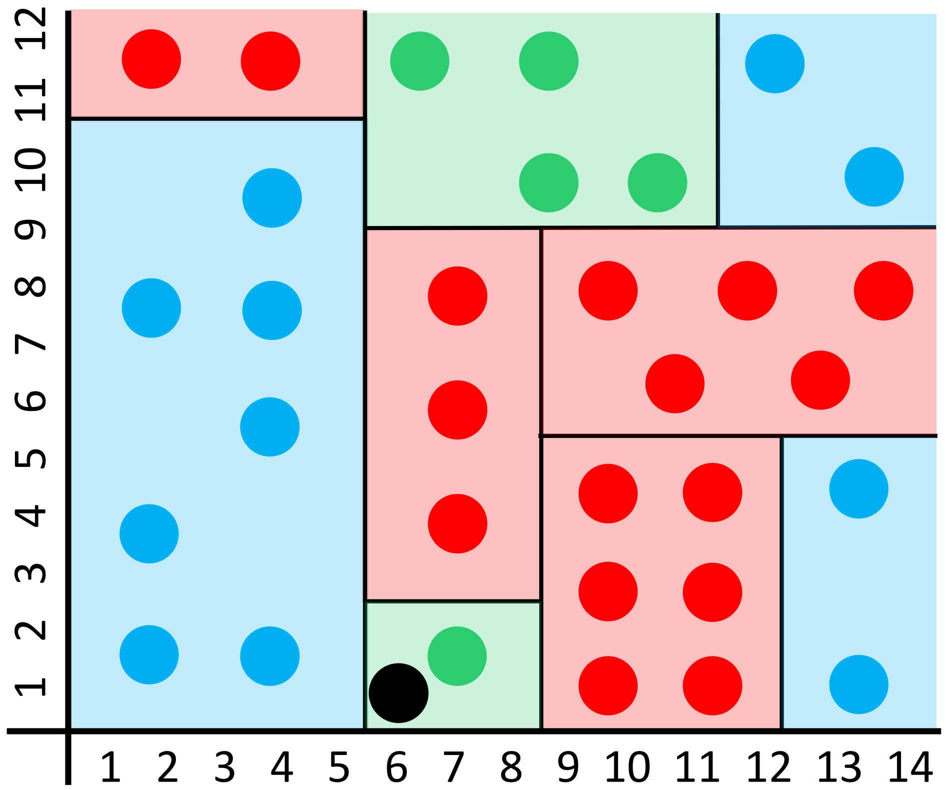
Building Decision Trees



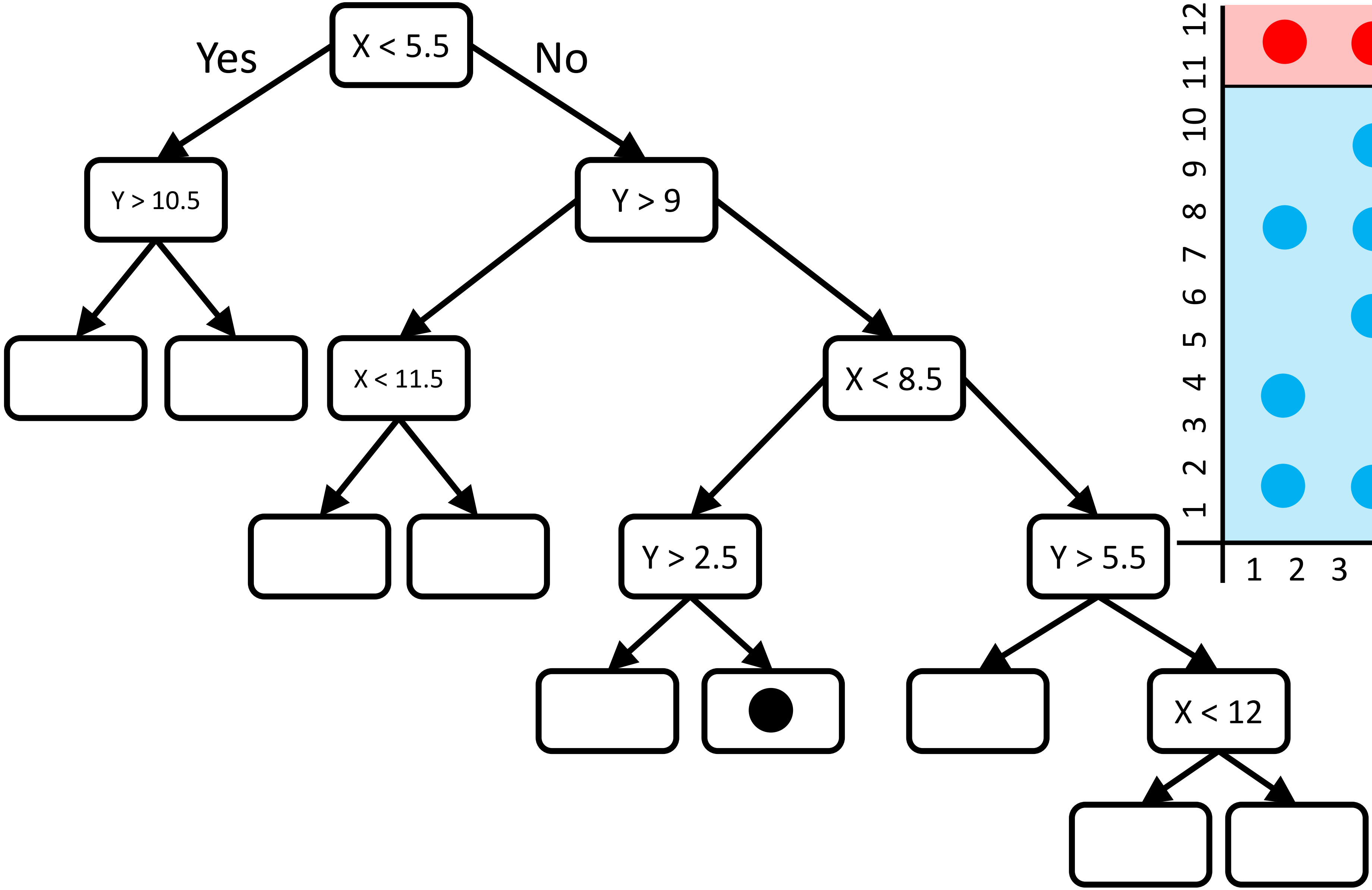


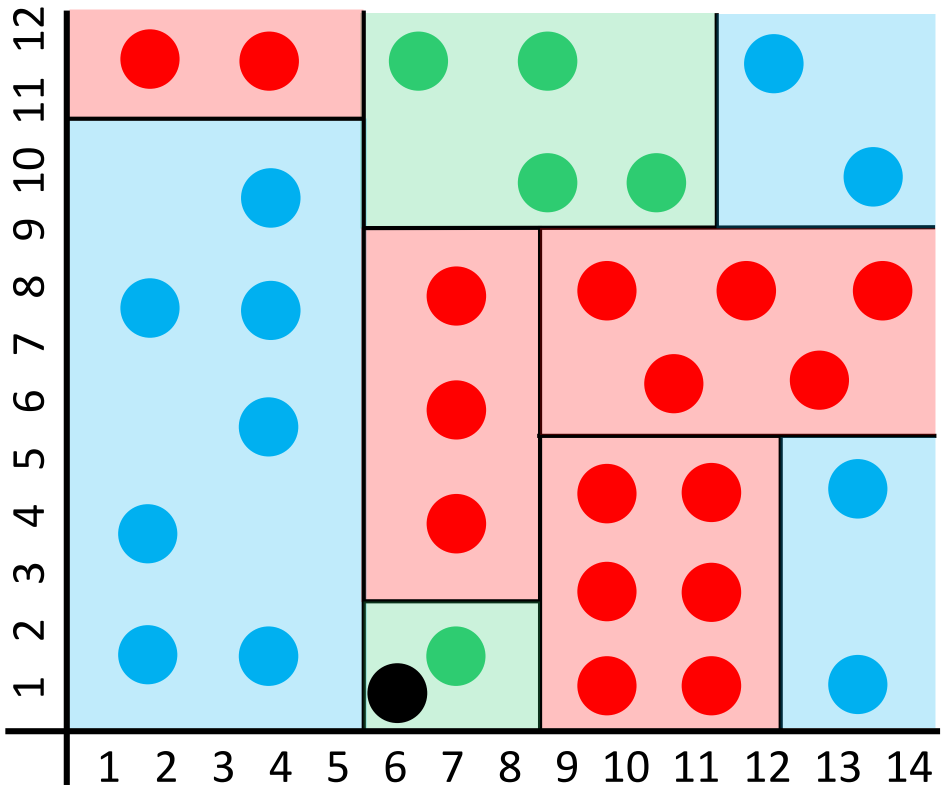
Building Decision Trees



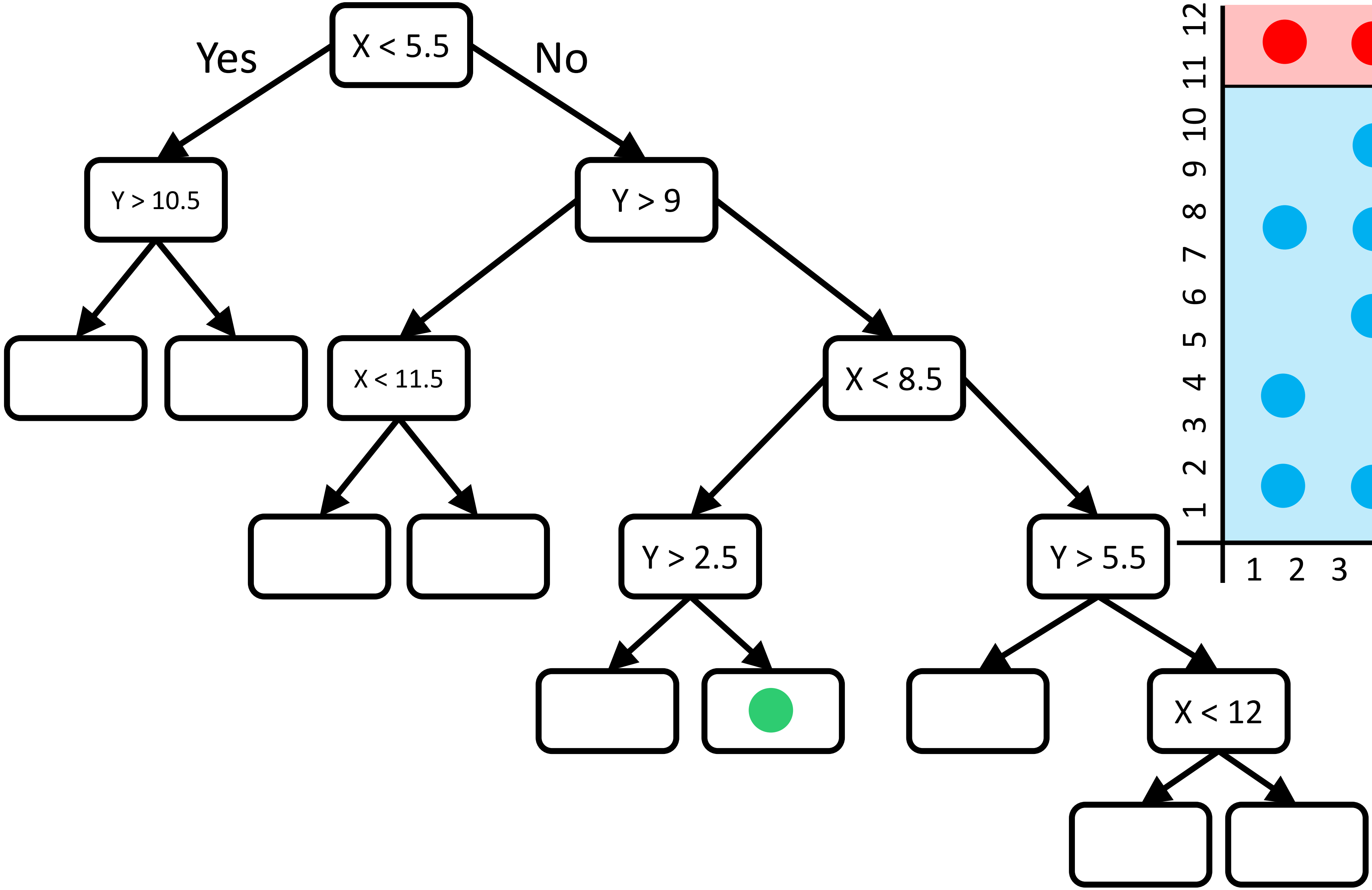


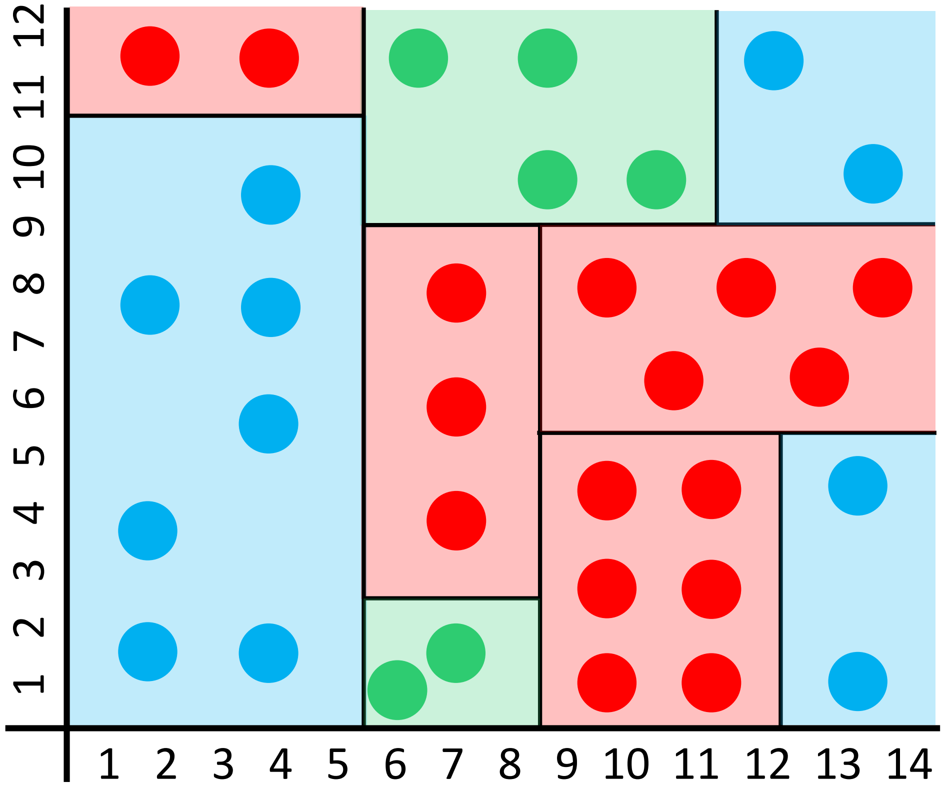
Building Decision Trees



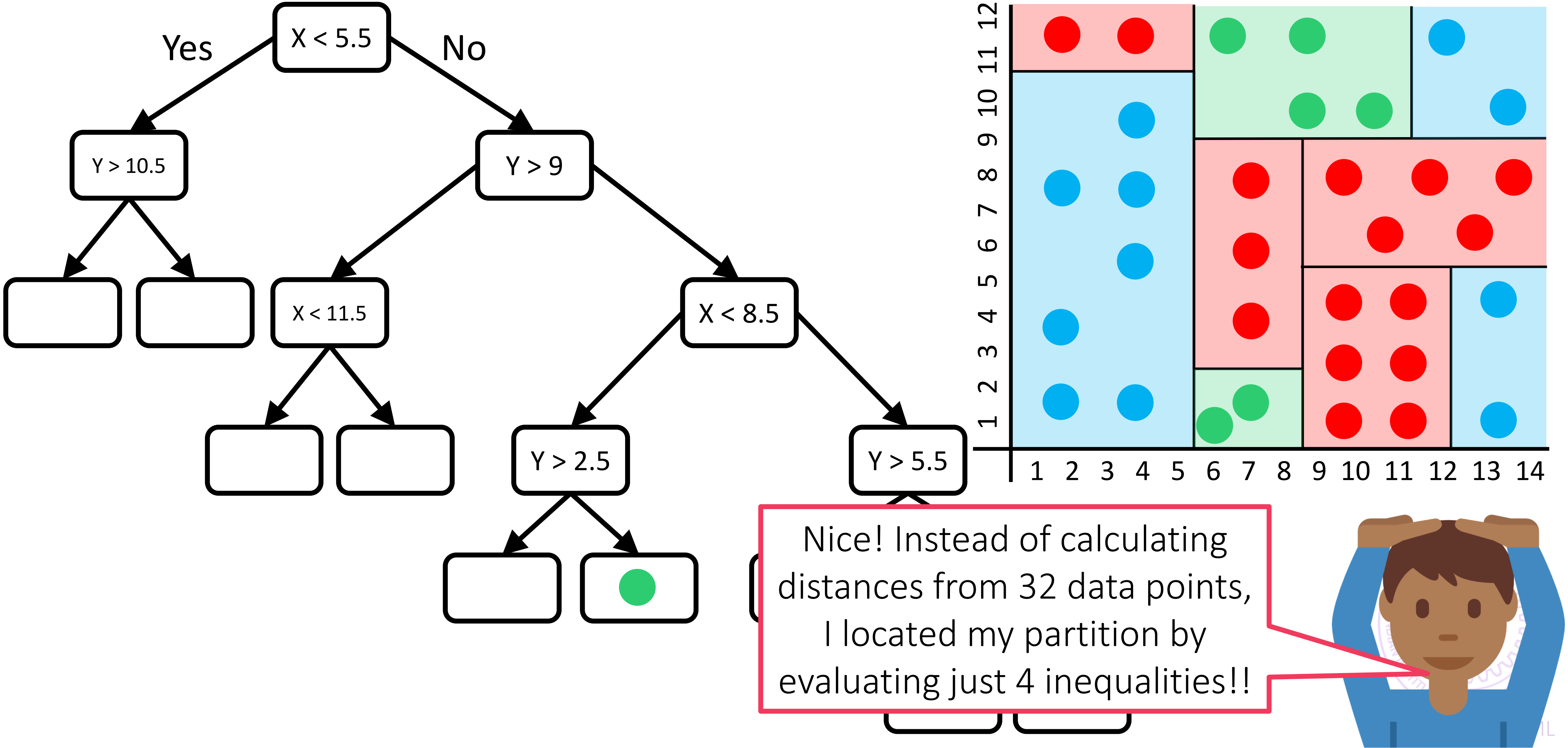


Building Decision Trees

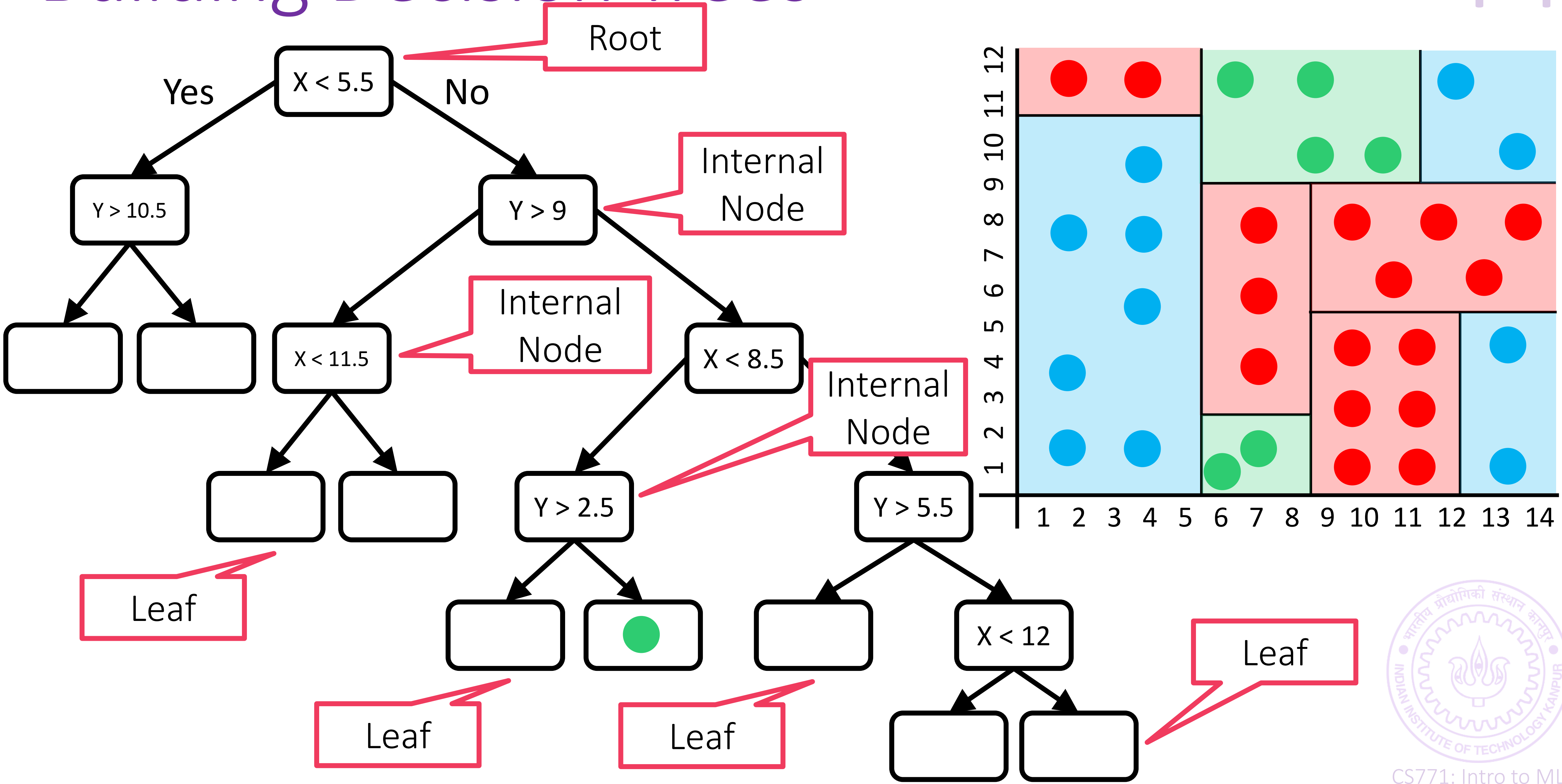




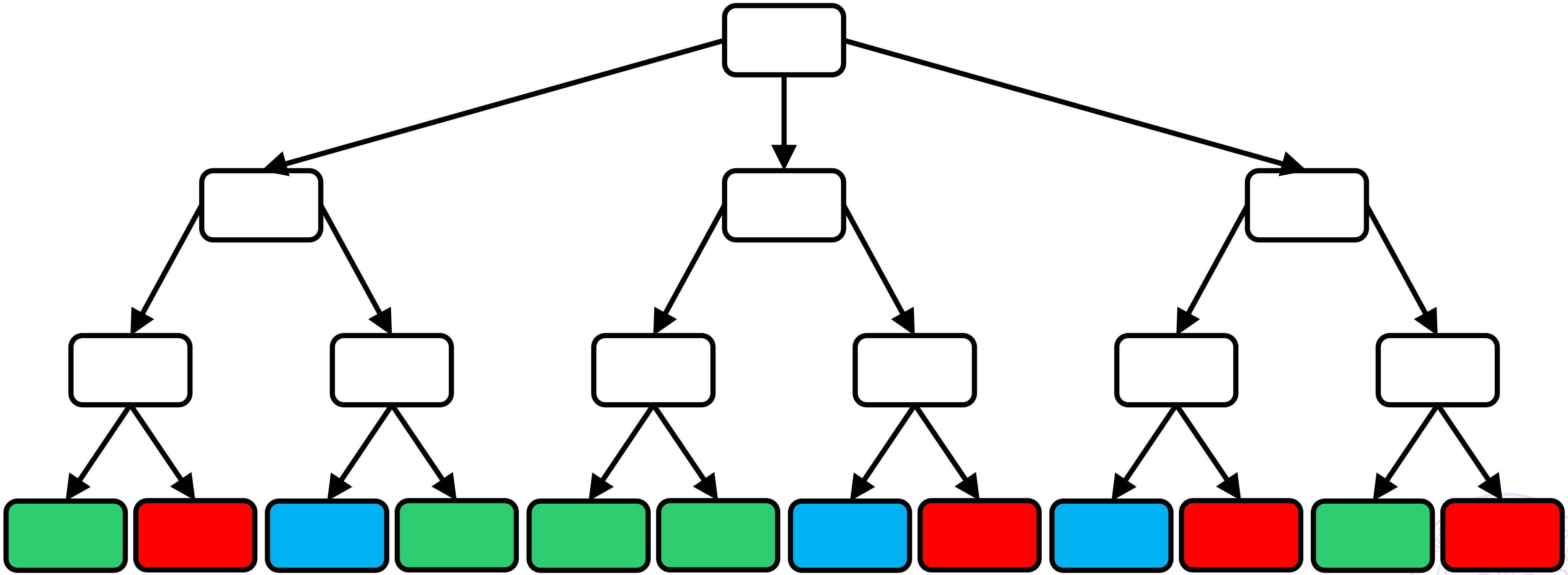
Building Decision Trees



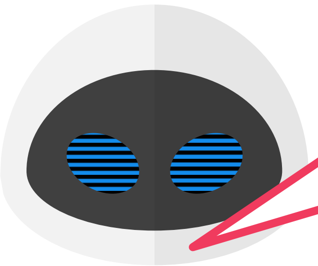
Building Decision Trees



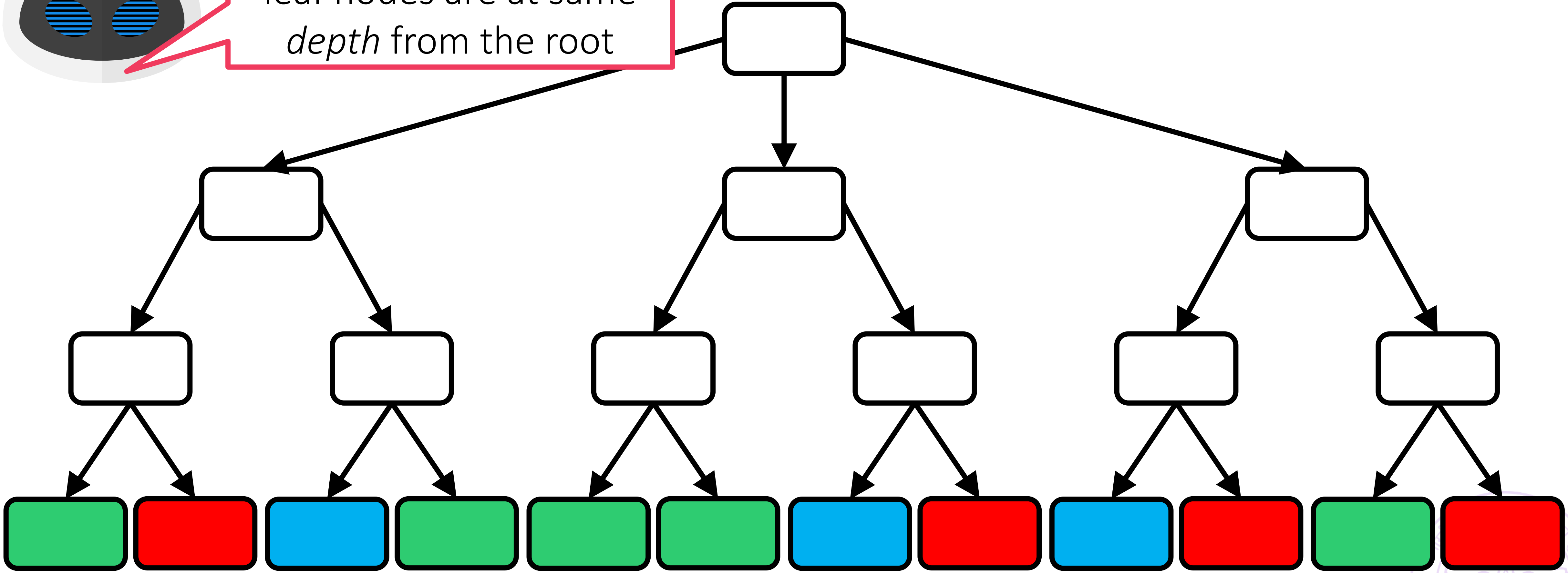
Decision Trees – all shapes and sizes



Decision Trees – all shapes and sizes



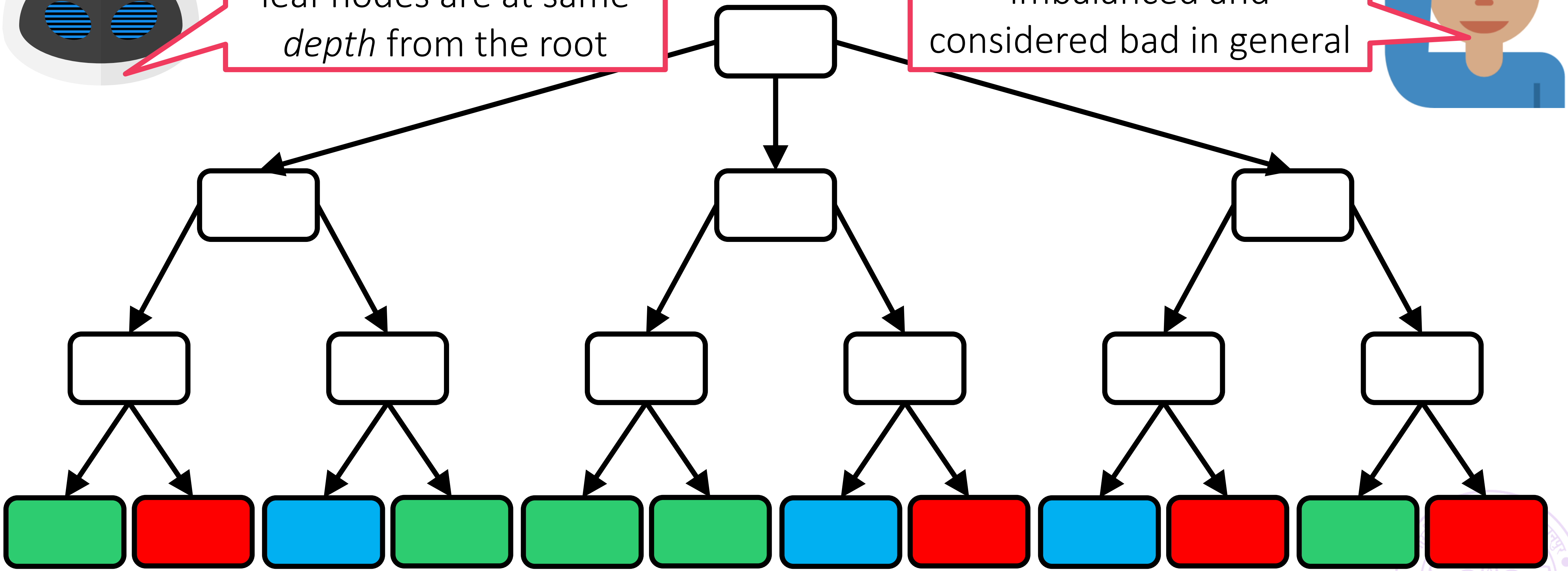
This DT is *balanced* – all leaf nodes are at same *depth* from the root



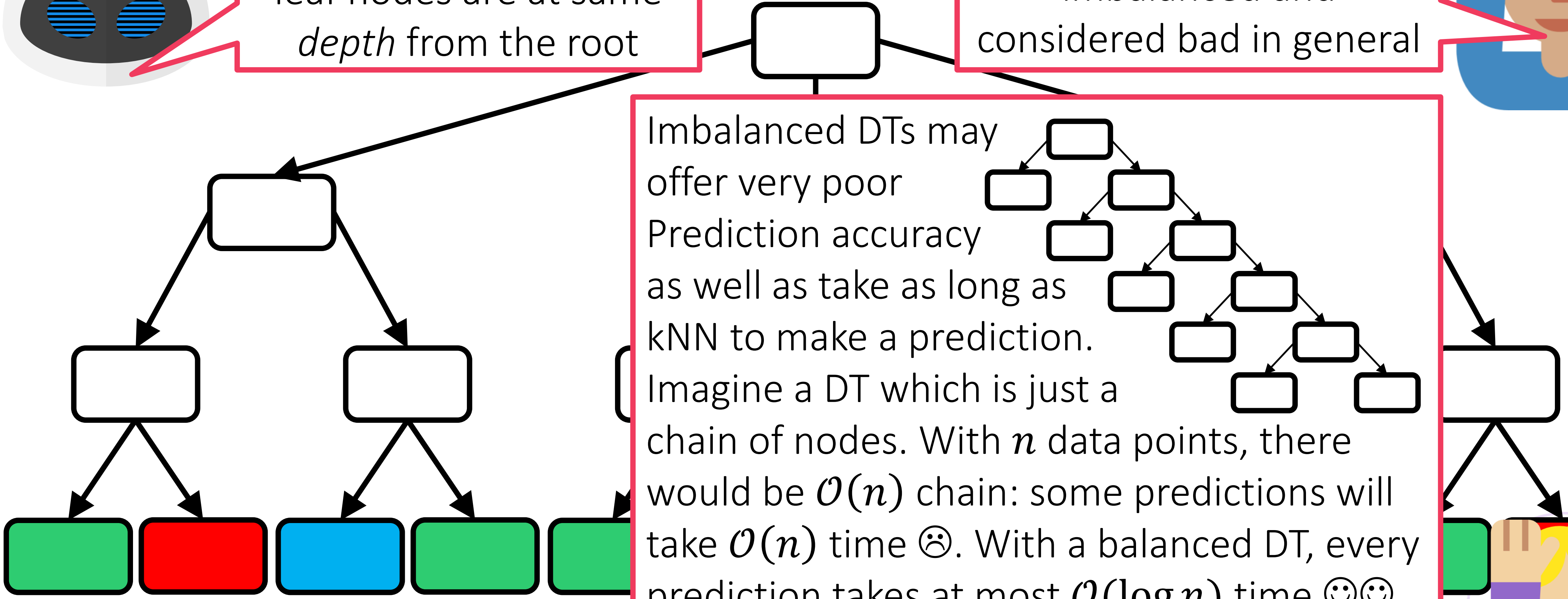
Decision Trees – all shapes and sizes

This DT is *balanced* – all leaf nodes are at same *depth* from the root

The previous DT was very imbalanced and considered bad in general



Decision Trees – all shapes and sizes

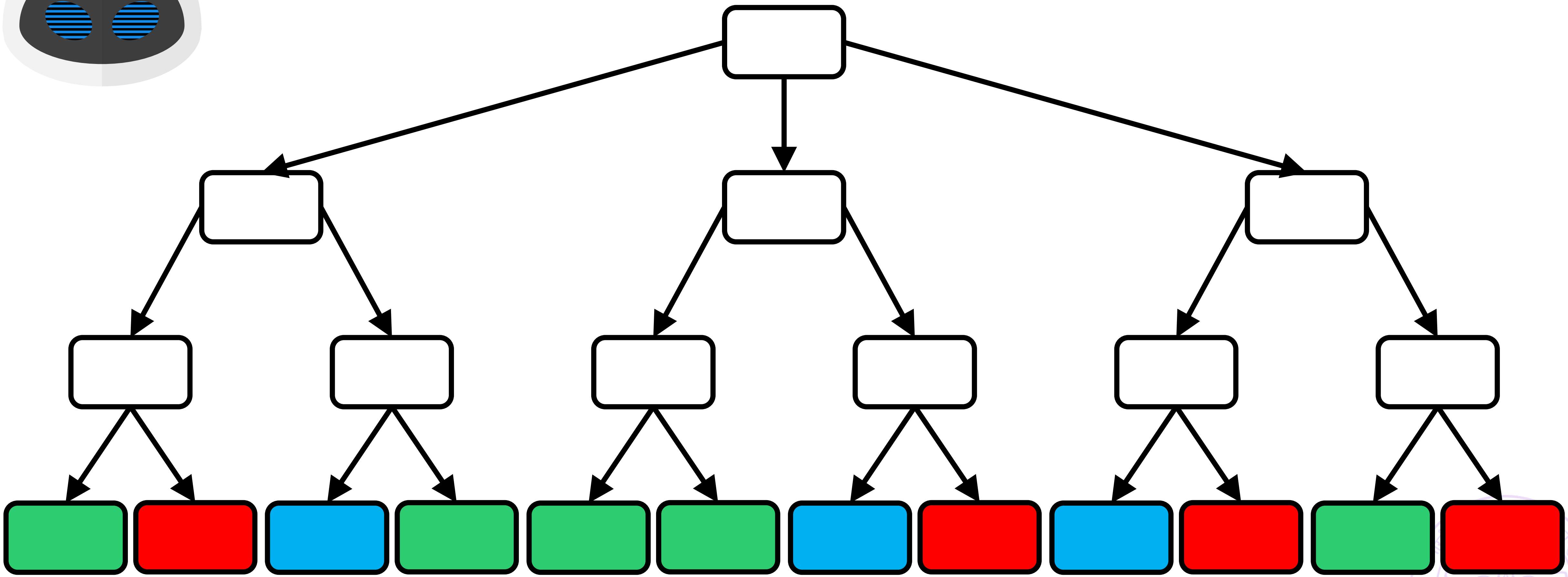


This DT is *balanced* – all leaf nodes are at same *depth* from the root

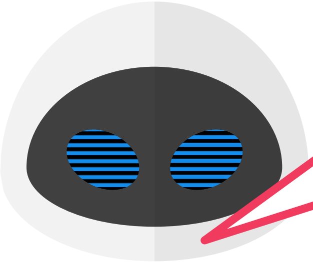
The previous DT was very imbalanced and considered bad in general

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 ☺☺

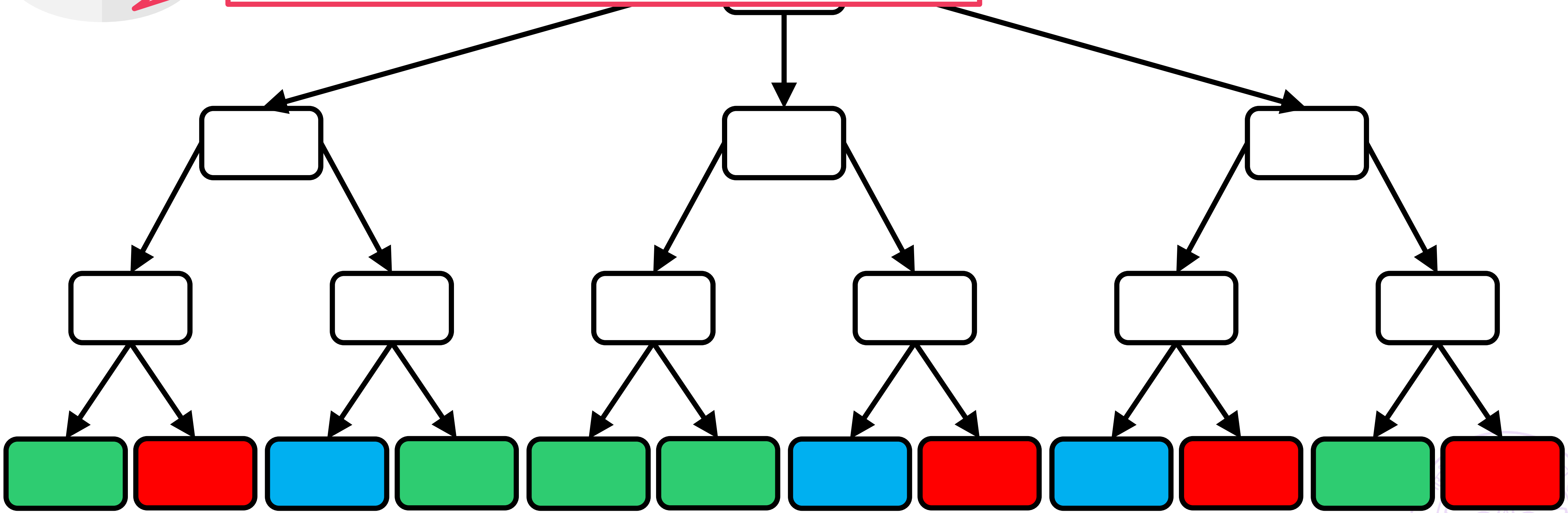
Decision Trees – all shapes and sizes



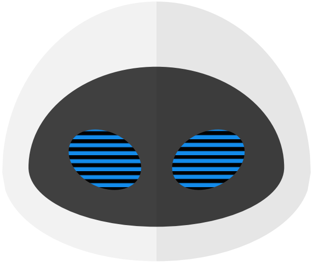
Decision Trees – all shapes and sizes

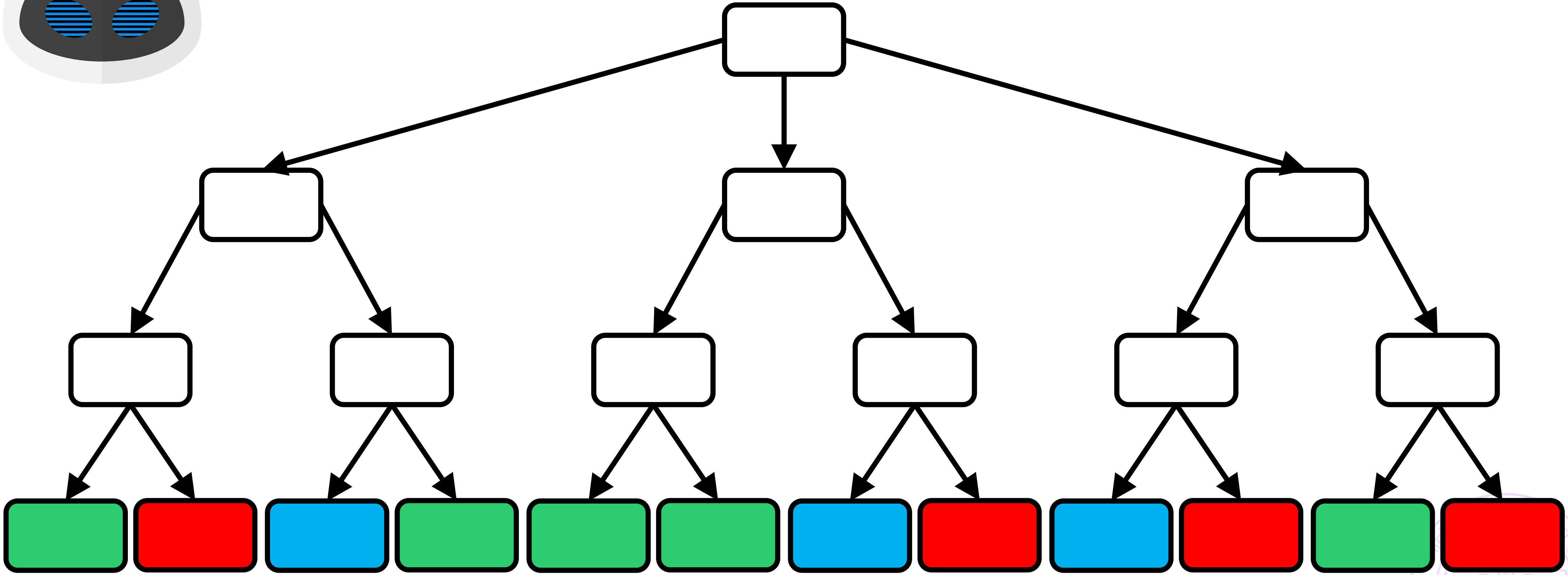


May prune the tree to make it more shallow as well. Also possible to have DT with more than 2 children per internal node

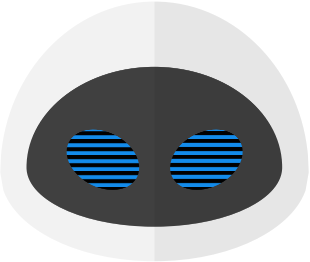


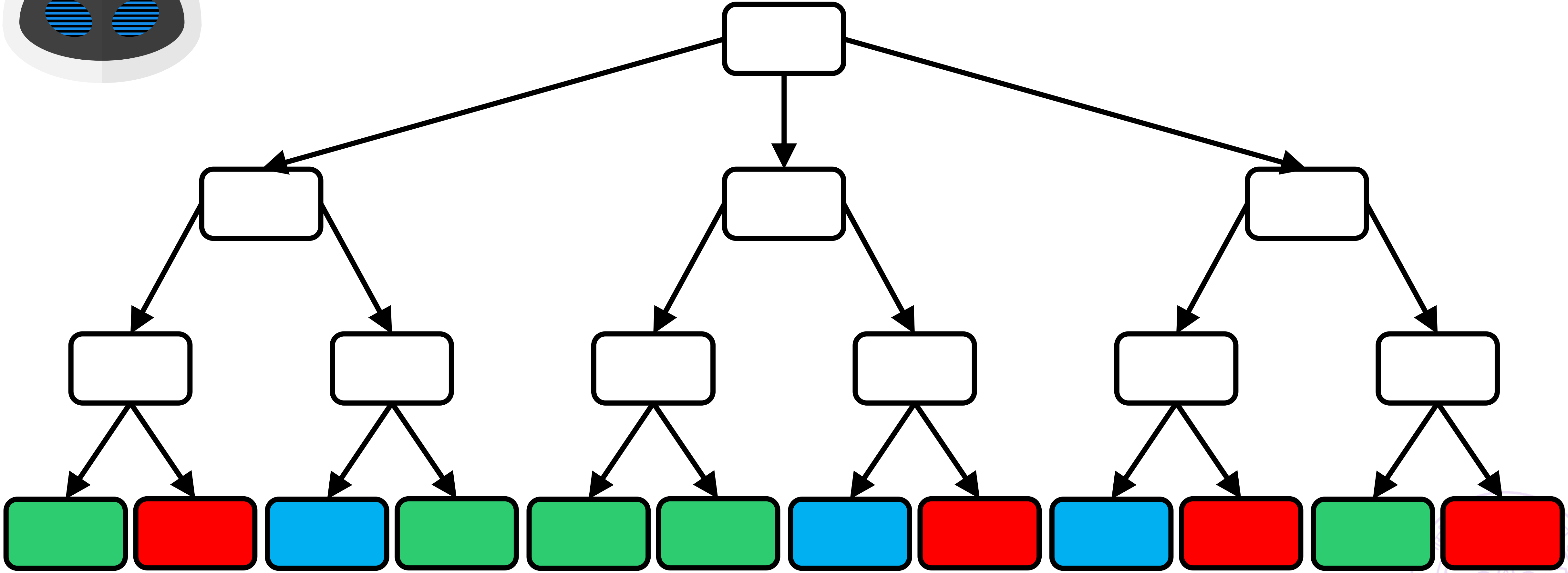
Decision Trees – all shapes and sizes



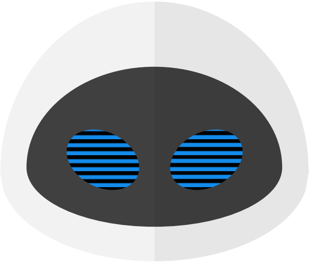


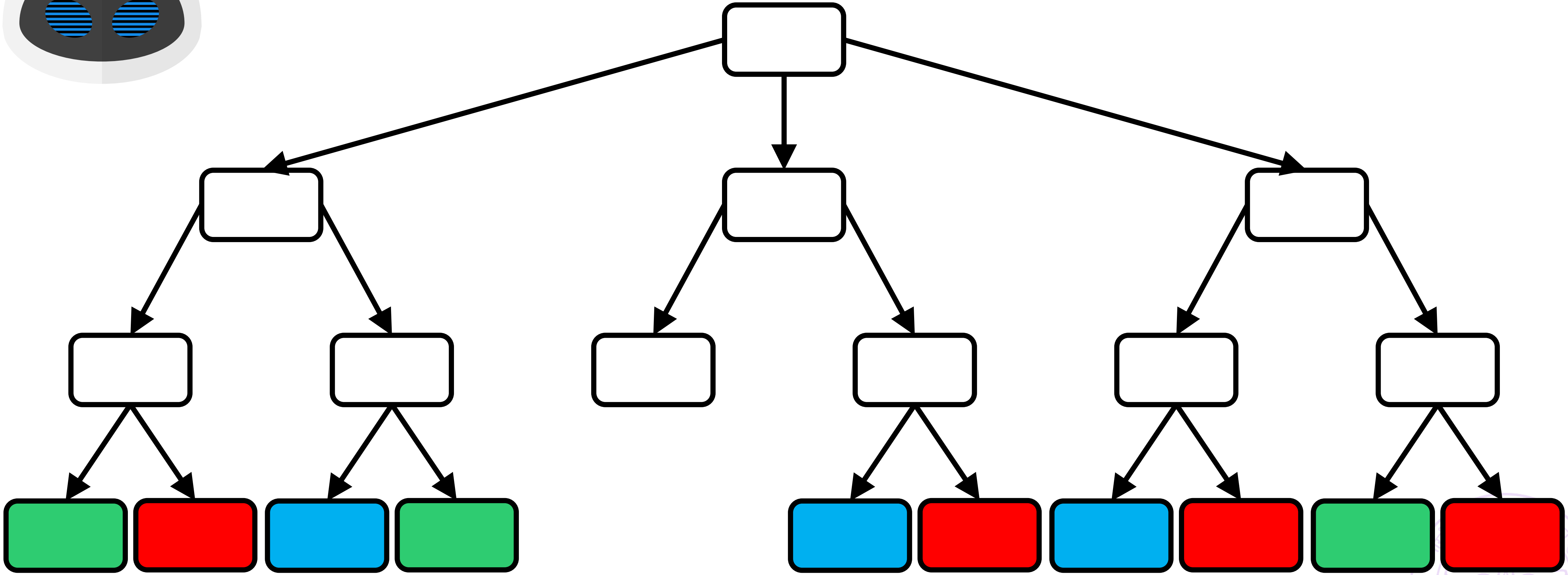
Decision Trees – all shapes and sizes



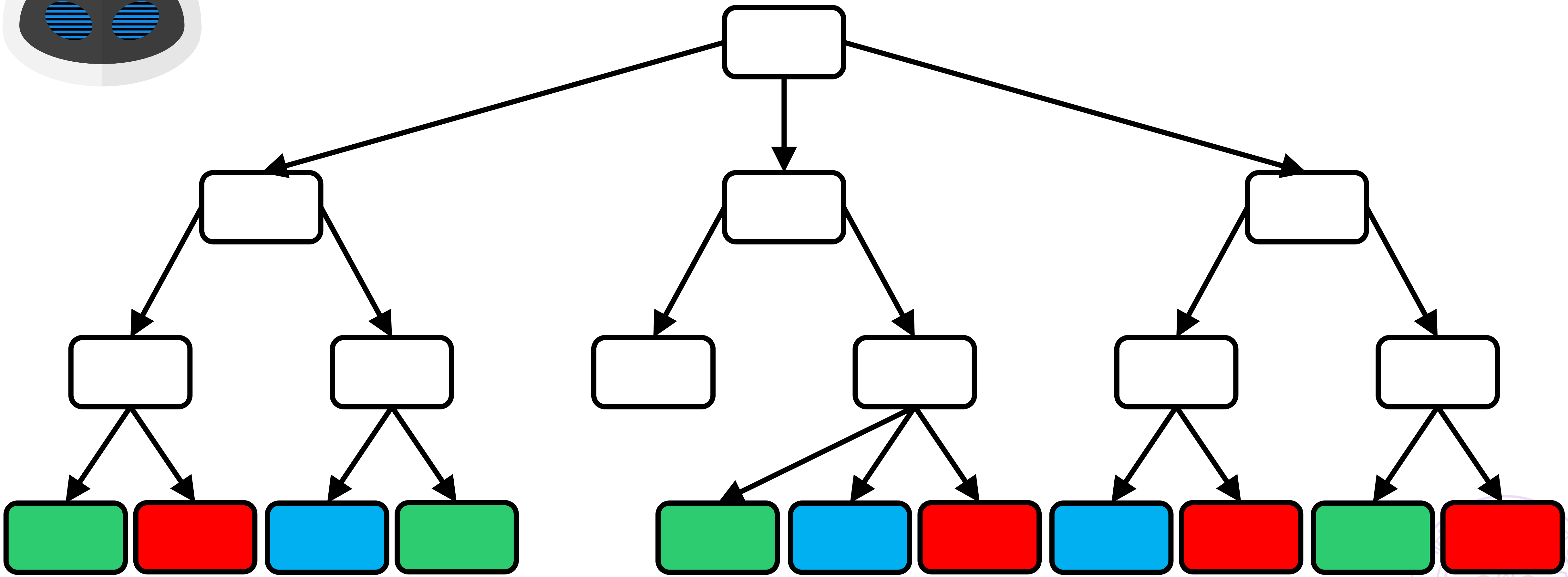


Decision Trees – all shapes and sizes

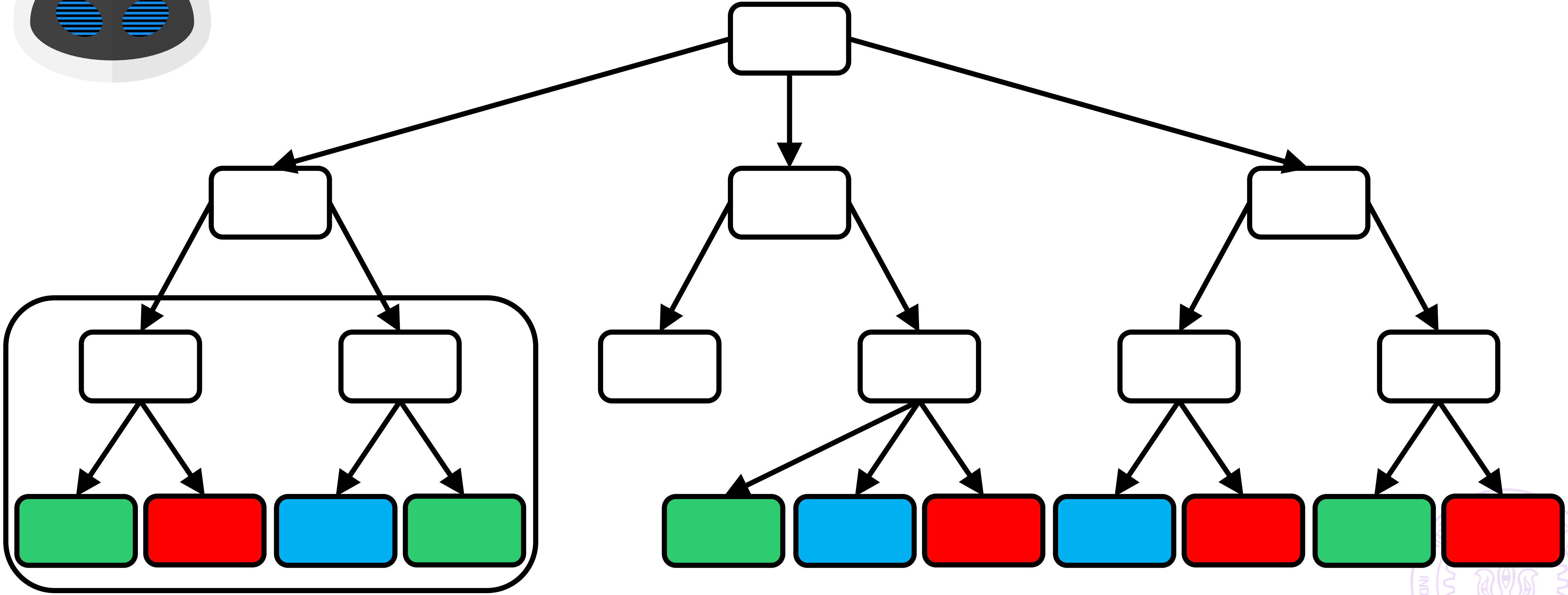




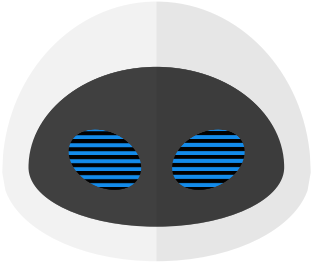
Decision Trees – all shapes and sizes

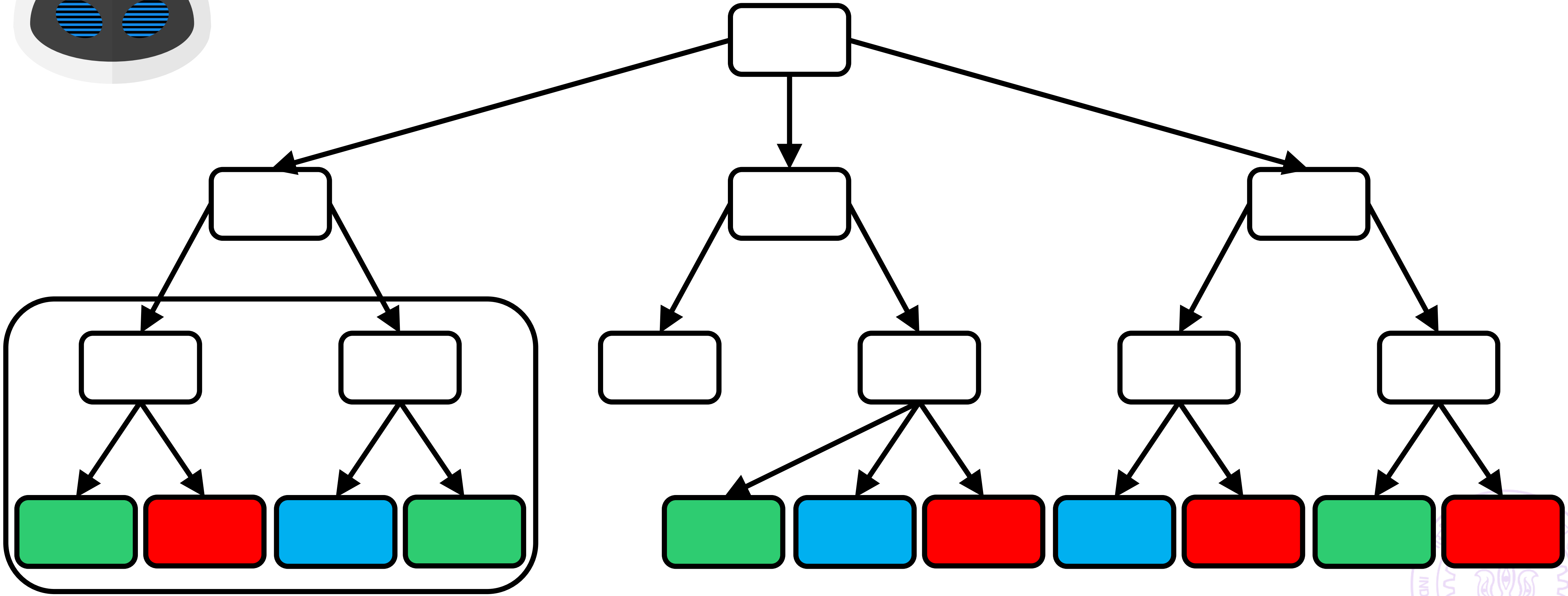


Decision Trees – all shapes and sizes

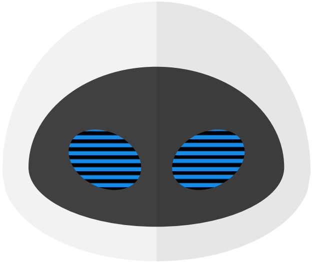


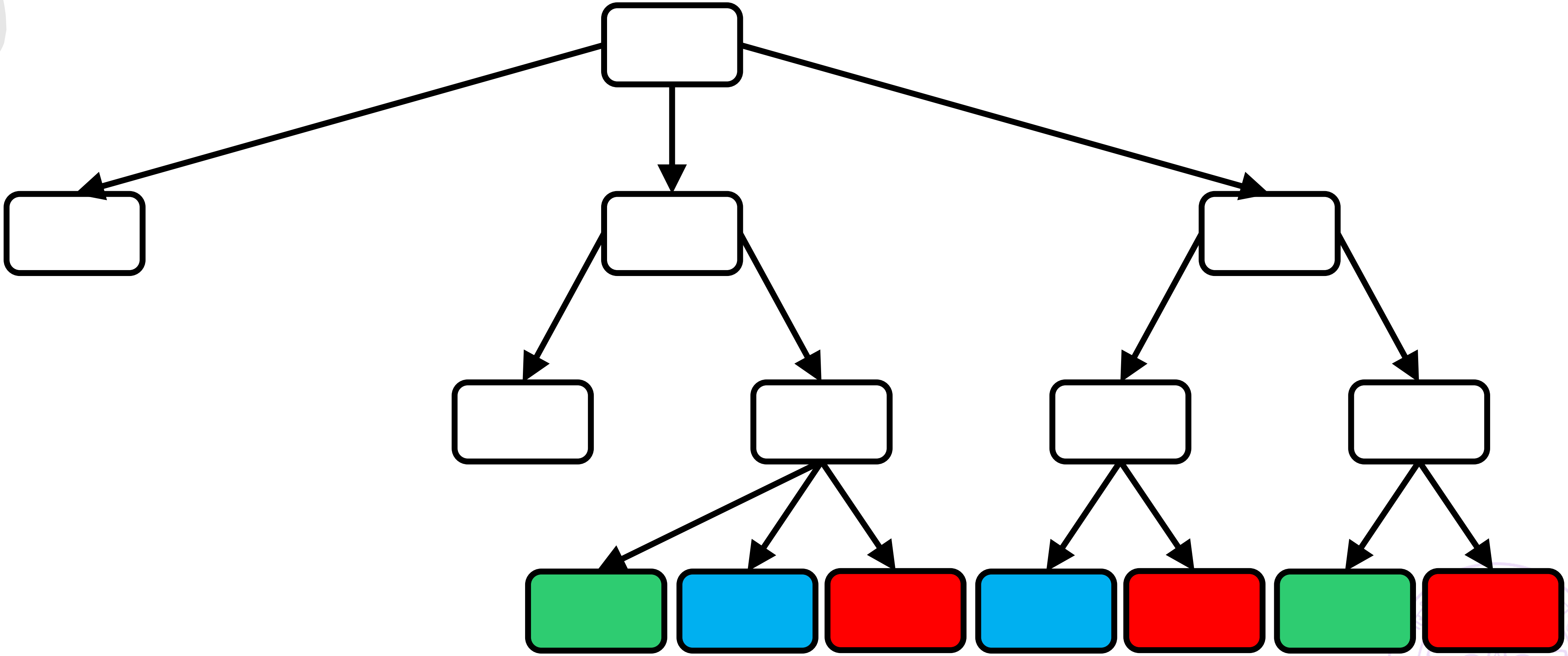
Decision Trees – all shapes and sizes



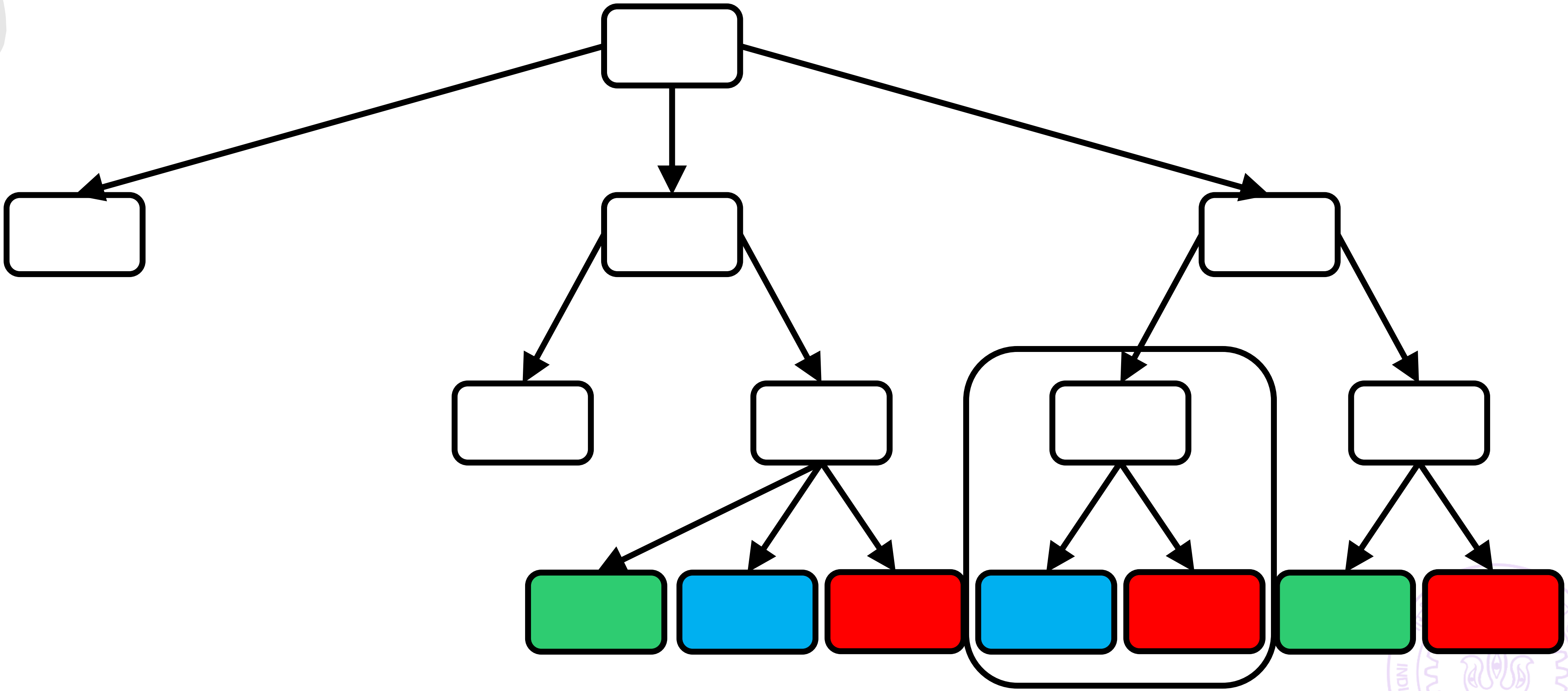


Decision Trees – all shapes and sizes

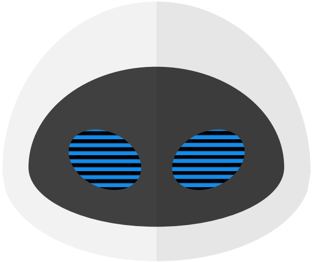


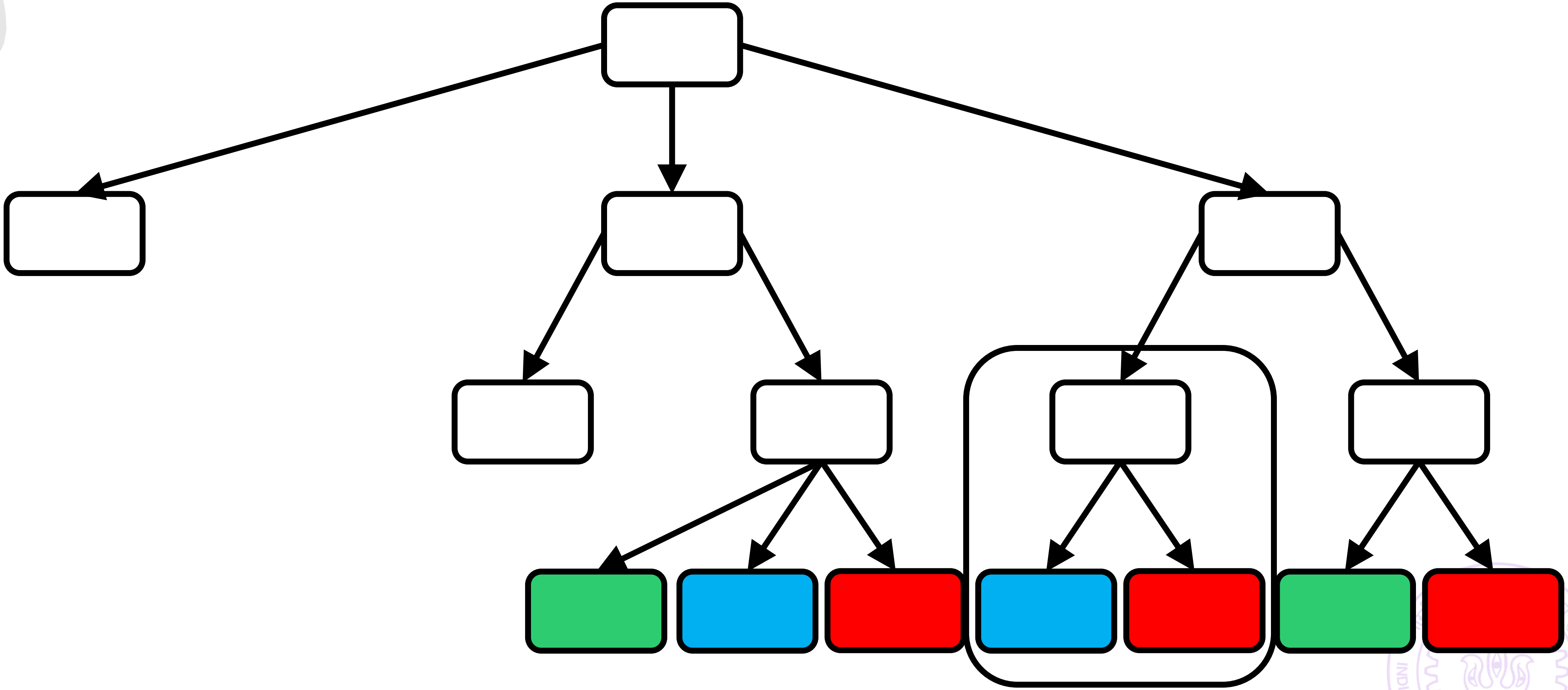


Decision Trees – all shapes and sizes

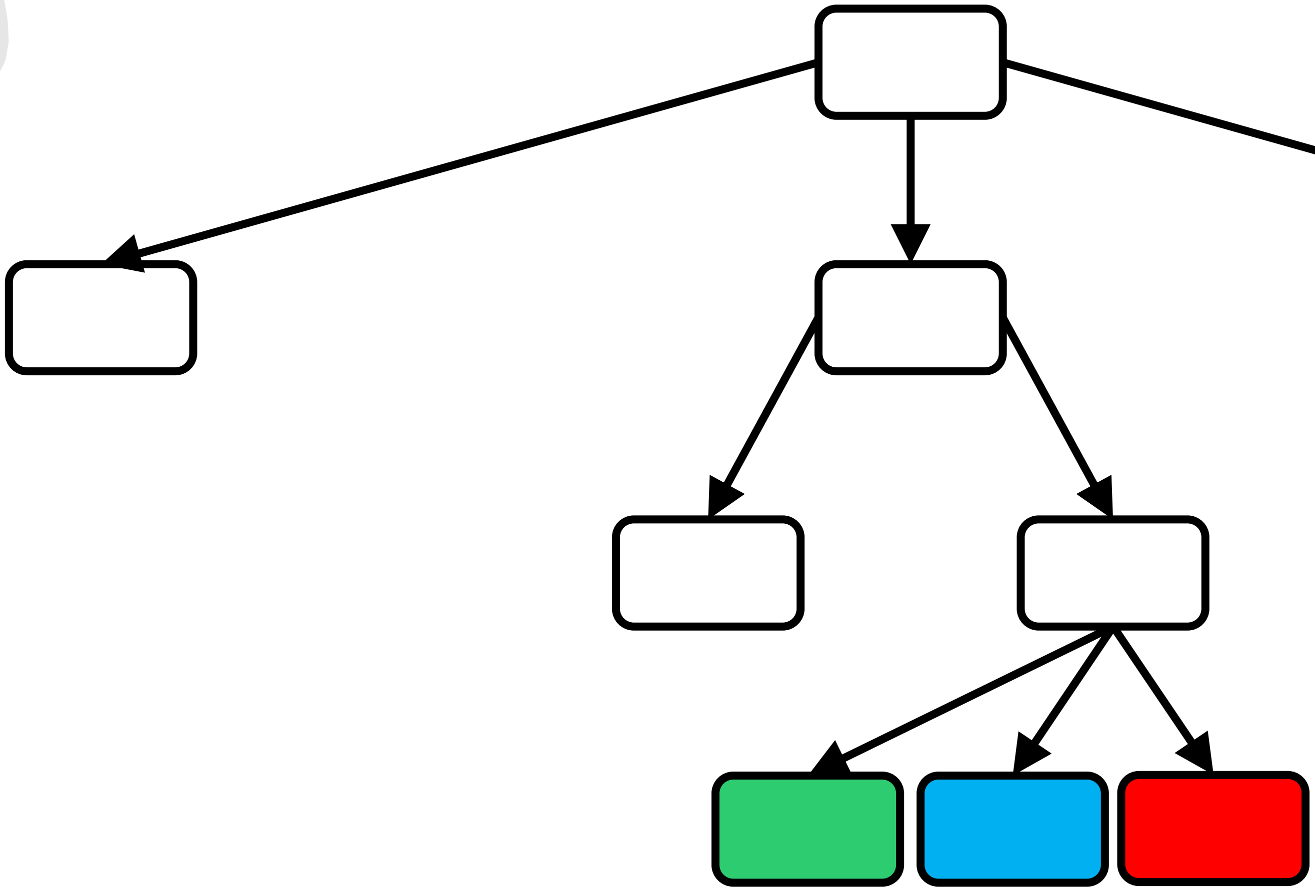


Decision Trees – all shapes and sizes

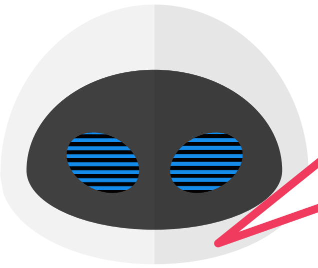




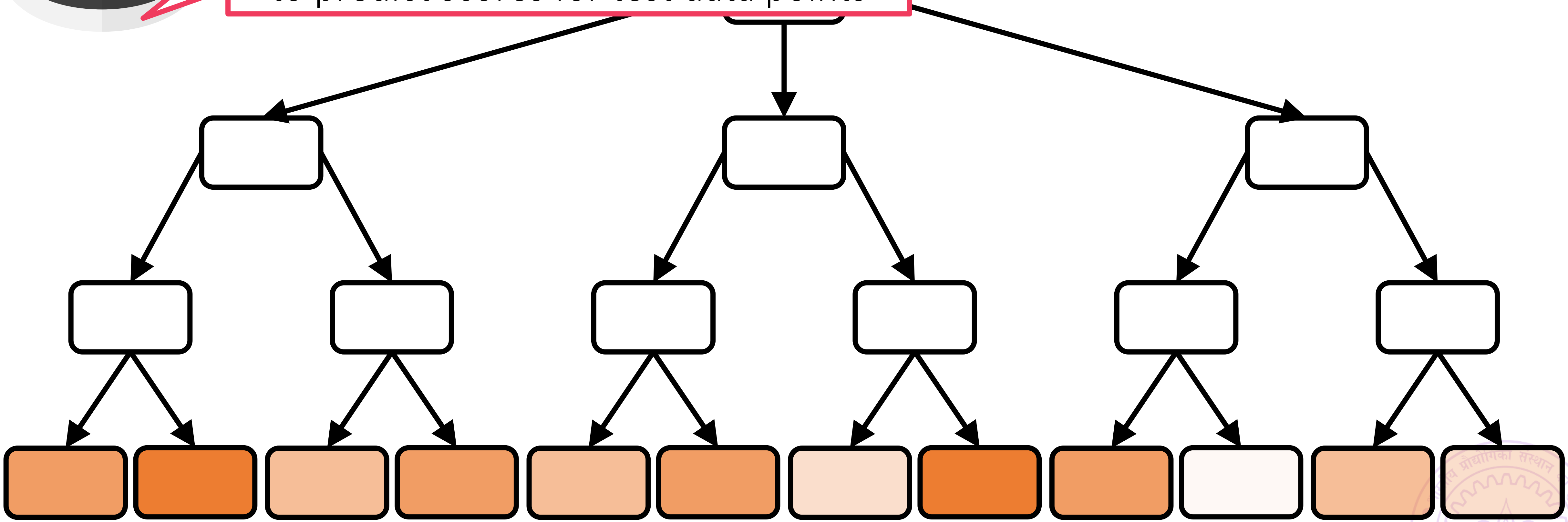
Decision Trees – all shapes and sizes



Regression with Decision Trees

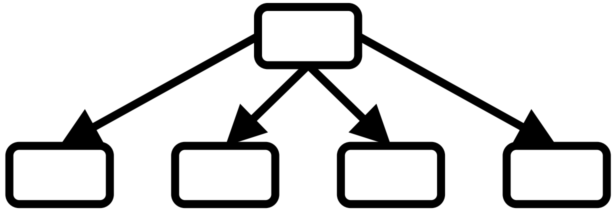


To perform real valued regression, may simply use average score at a leaf node to predict scores for test data points

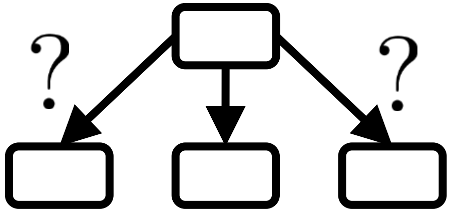


How to learn a DT?

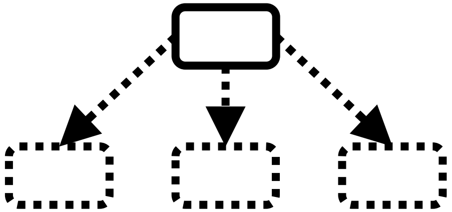
How many children should a node have?



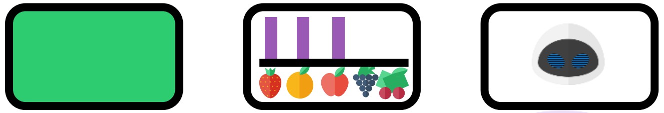
How to send data points to children?



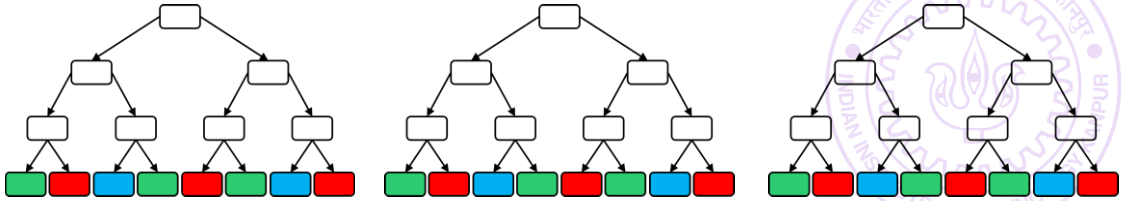
When to stop splitting and make the node a leaf?



What to do at a leaf?



How many trees to train?



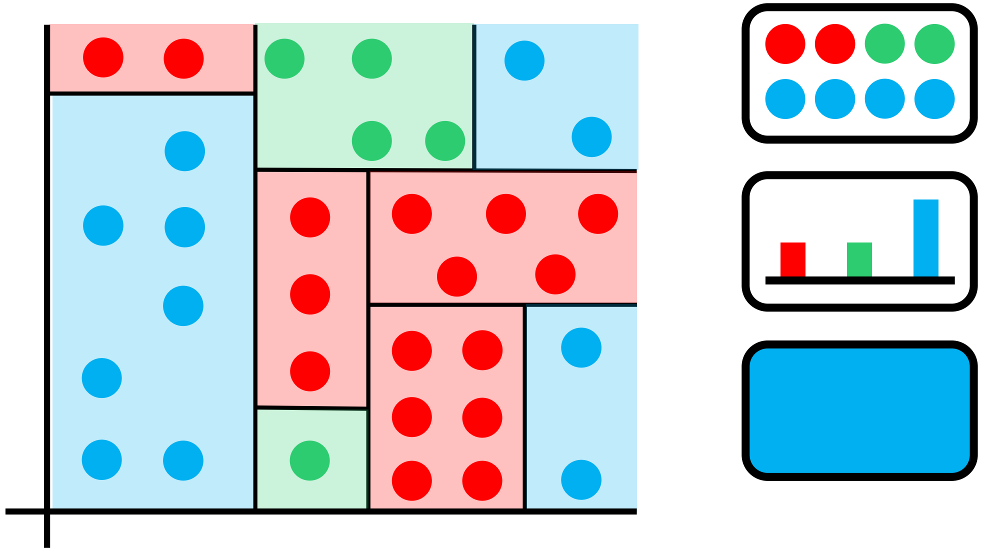
What to do at a leaf?

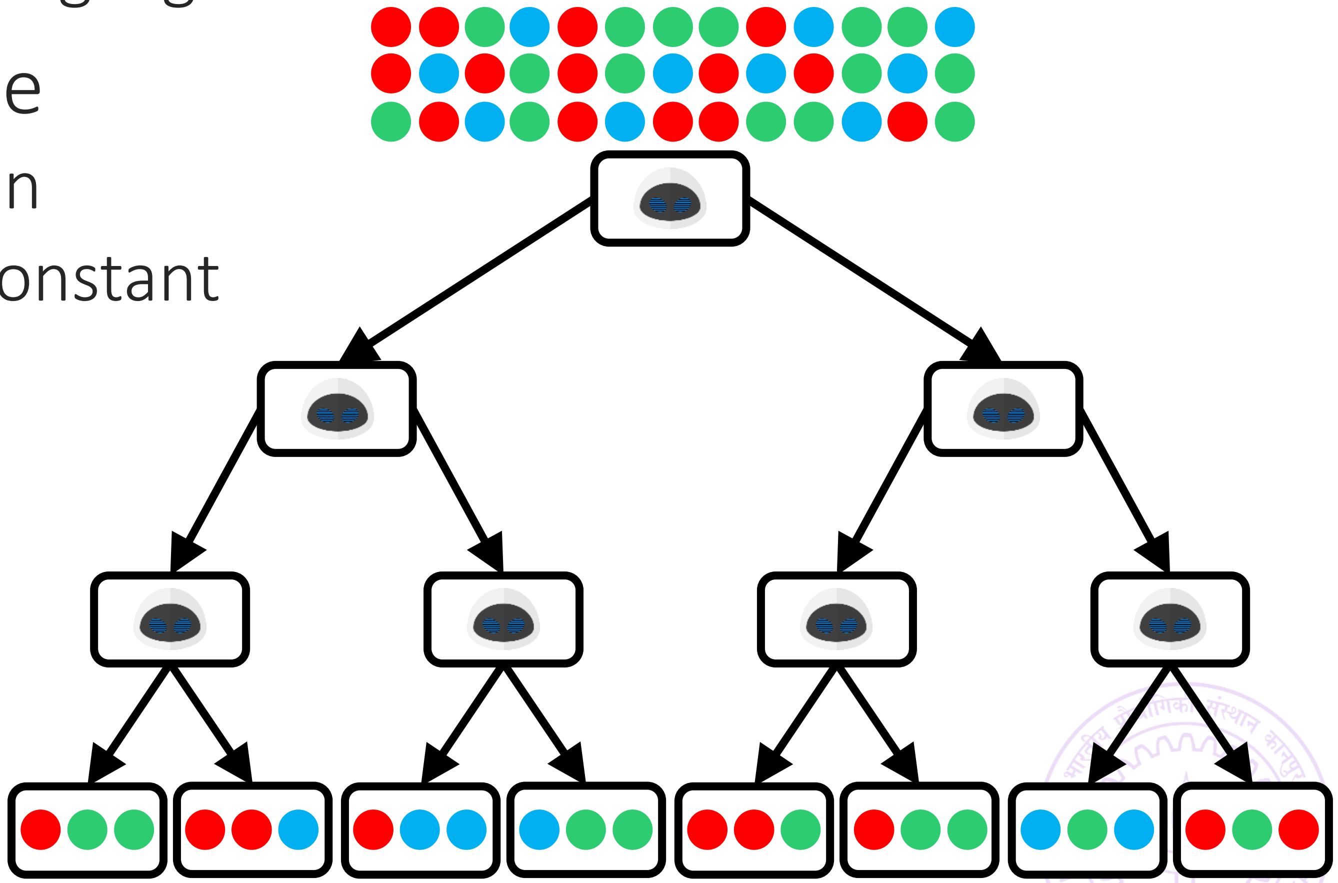
Can take any (complicated) action at a leaf

- Why not call another machine learning algorithm?

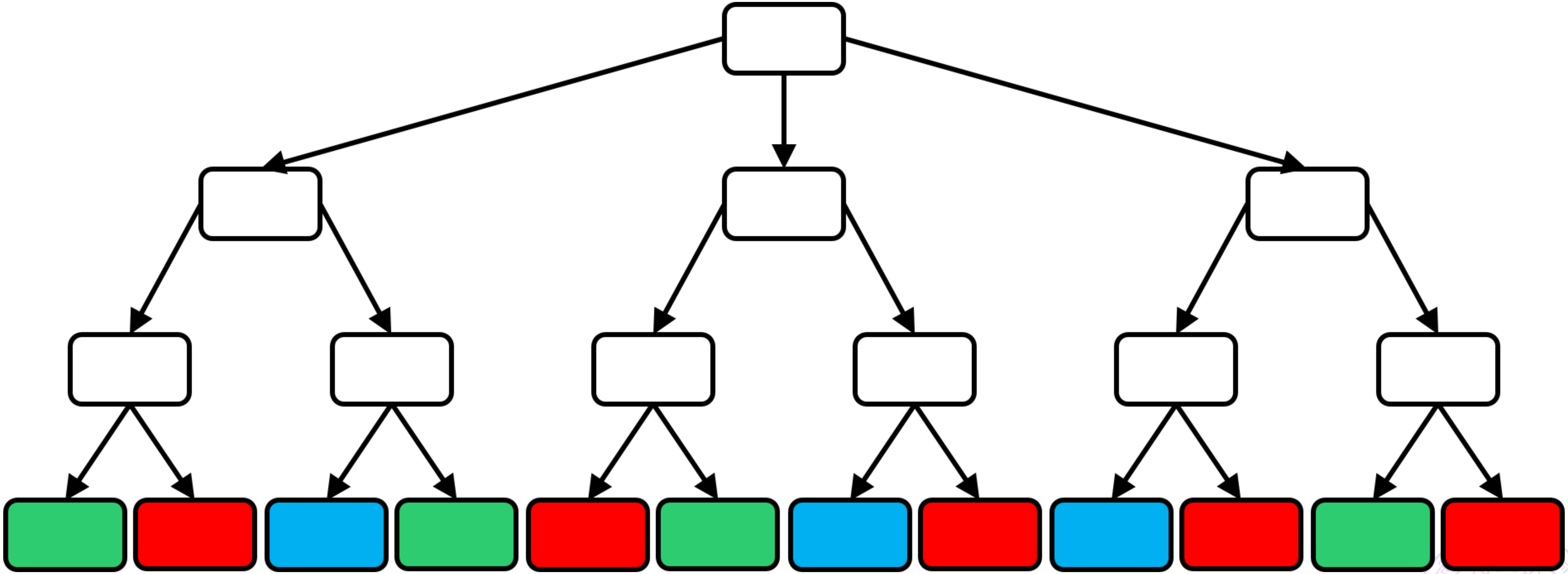
For speed, keep leaf action simple

- Simplest action – constant prediction
- Such a DT will encode a piecewise constant prediction function

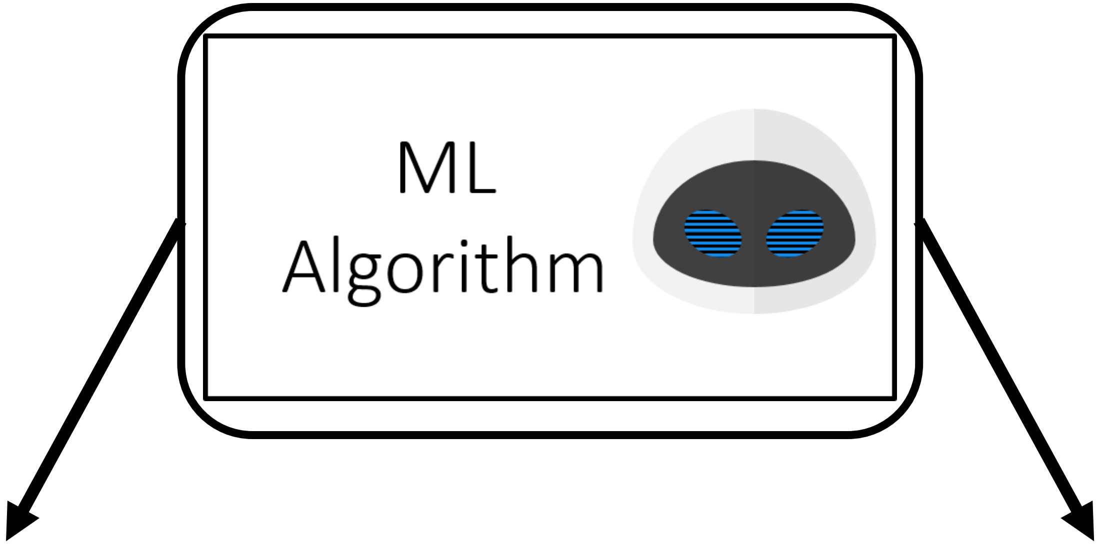




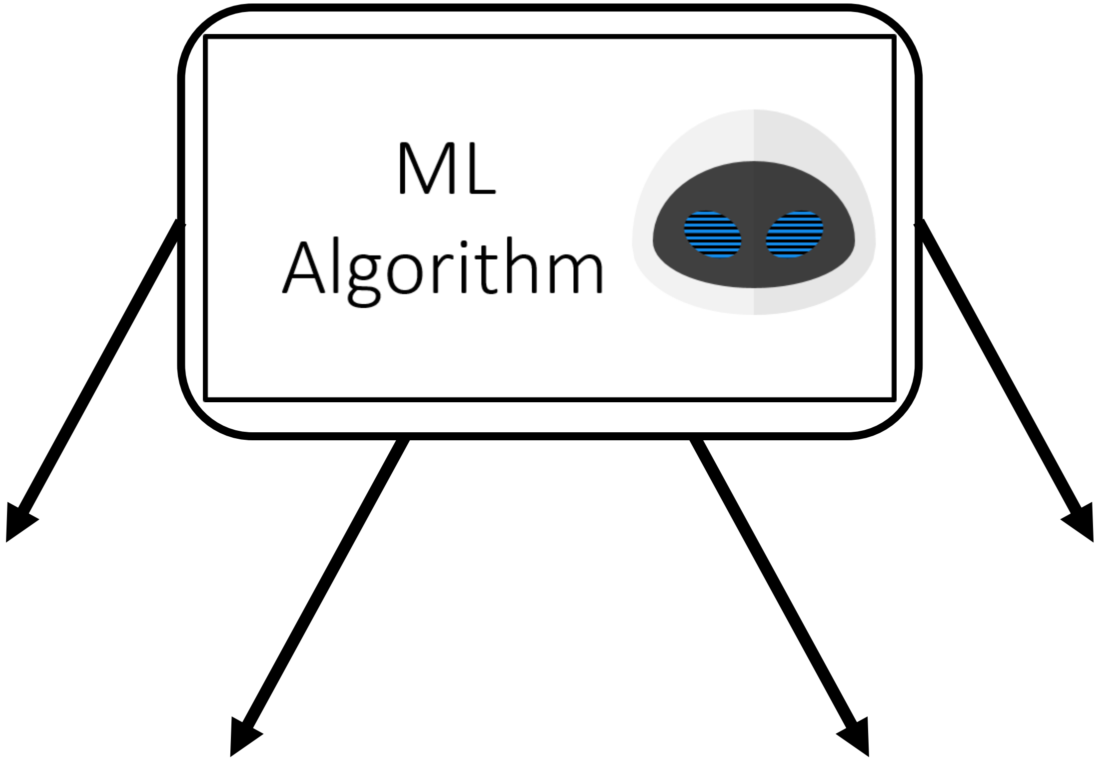
How to split a node into children nodes?



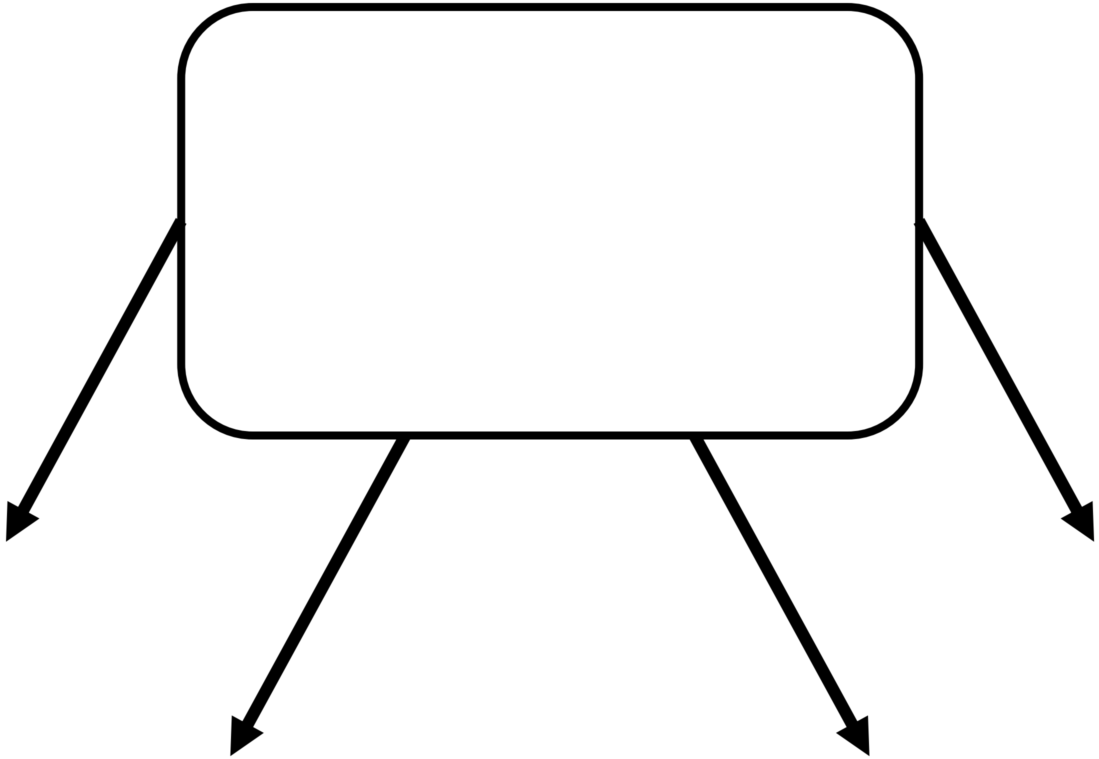
How to split a node into children nodes?



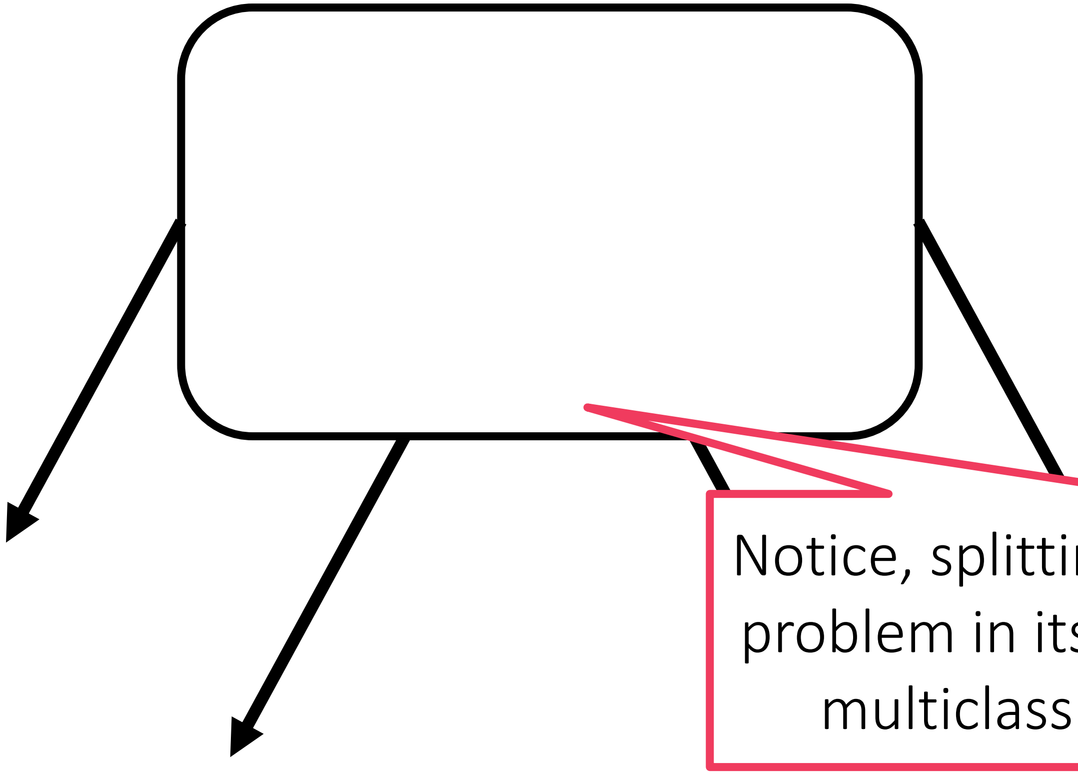
How to split a node into children nodes?



How to split a node into children nodes?

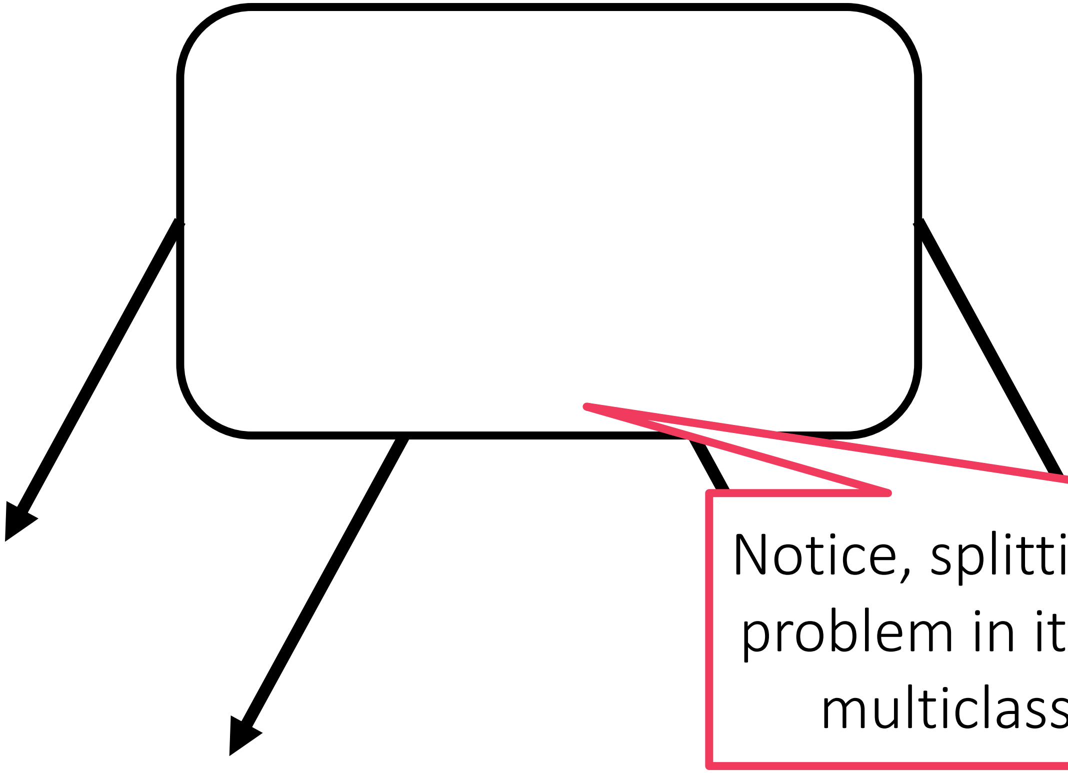


How to split a node into children nodes?



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

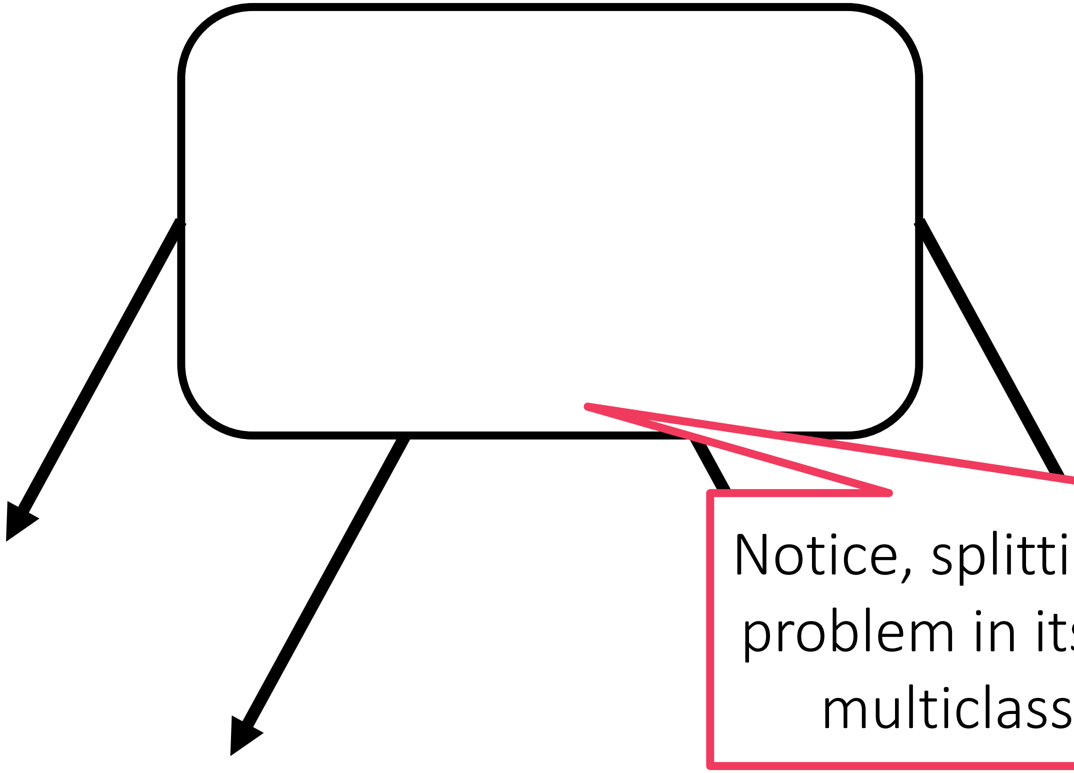
How to split a node into children nodes?



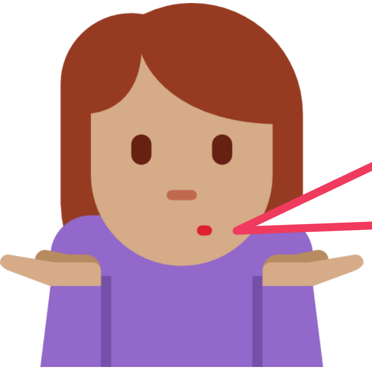
Notice, splitting a node is a classification problem in itself! Binary if two children, multiclass if more than 2 children

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

How to split a node into children nodes?



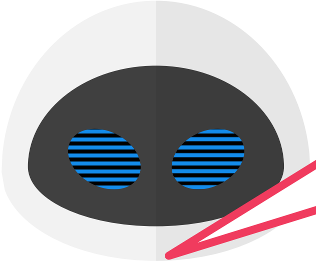
Notice, splitting a node is a classification problem in itself! Binary if two children, multiclass if more than 2 children



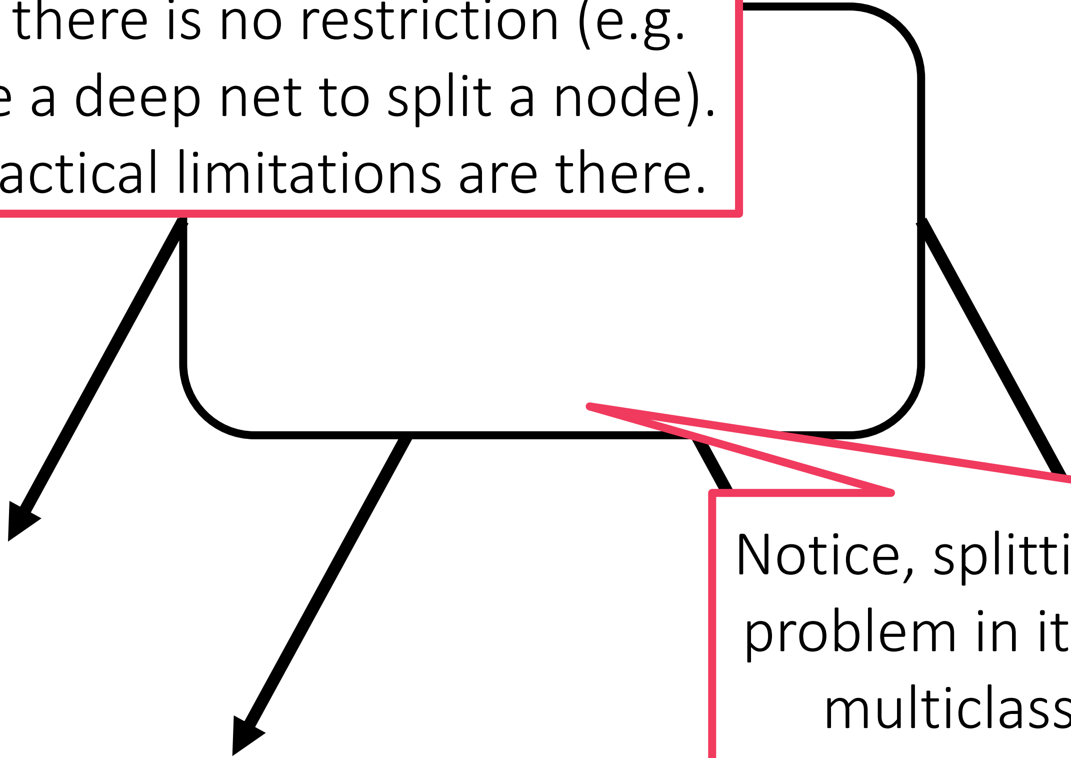
Can we use any classification technique to split a node or are there some restrictions?

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

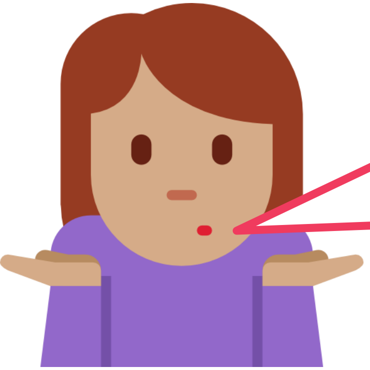
How to split a node into children nodes?



In principle there is no restriction (e.g. can even use a deep net to split a node). However, practical limitations are there.



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 – some lessons

Recall, one of the goals of DT is to speed up kNN prediction time

Thus, node splitting algorithm must be super fast otherwise no benefit of using DT – may just as well do NN directly

Often people carefully choose just a single feature and split a node based on that e.g. (age < 25 go left, age ≥ 25 go right)

Such "simple classifiers" are often called *decision stumps*

Splitting a Node – s

Recall, one of the goals of DT

Thu m must be super fast otherwise no
ben st

Ofte e
base go left, age ≥ 25 go right)

Suc often called *decision stumps*

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 ☺

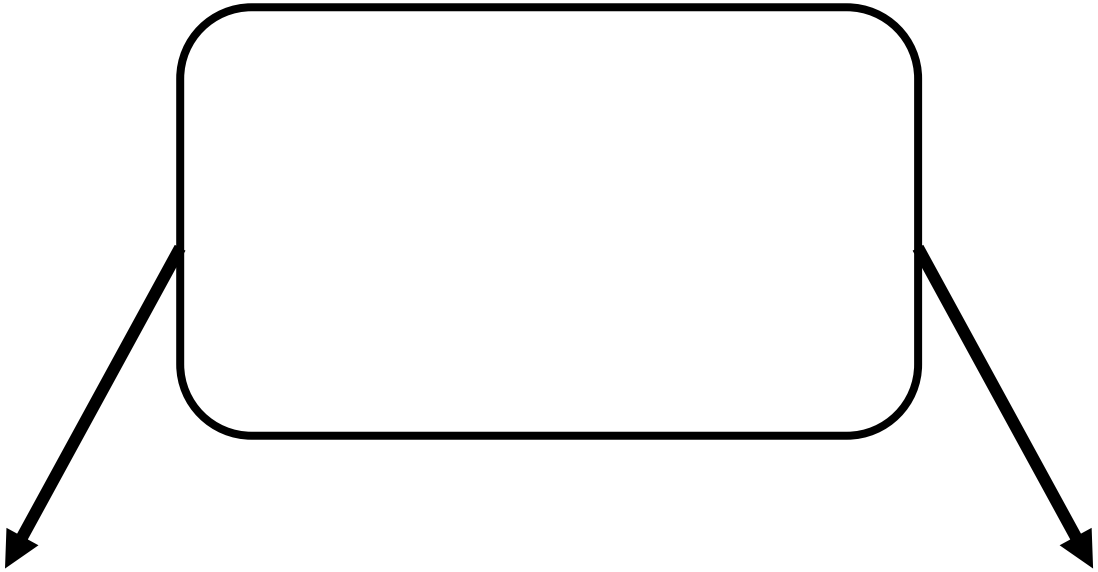
Making sure that the split is balanced (e.g. roughly half the data points go left and right) is also important to ensure that the tree is balanced. However, ensuring balance is often tricky

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

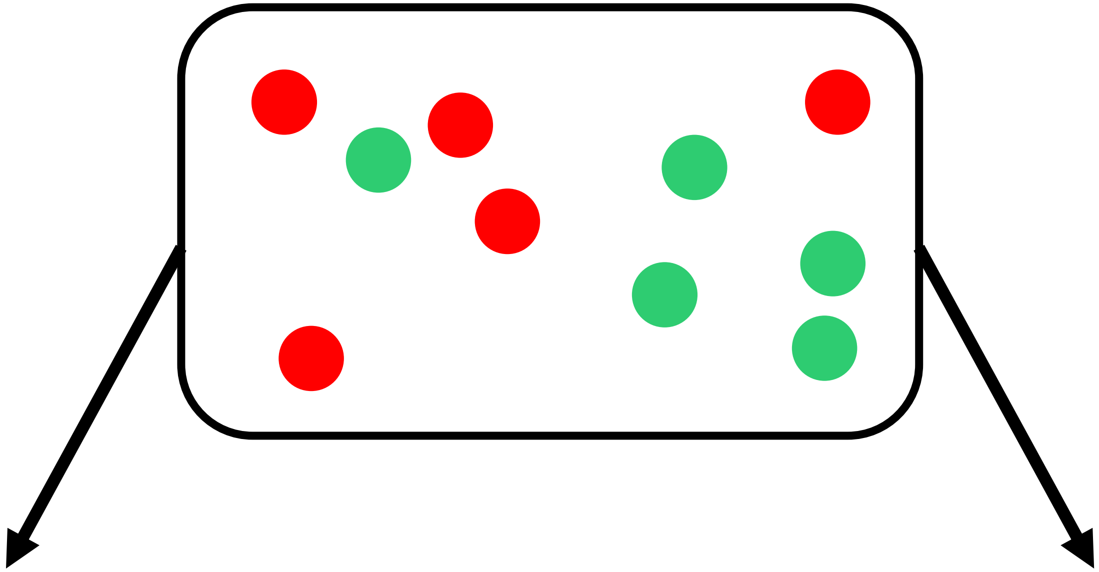
How do I decide whether to use age or gender? Even if using age, how do I decide whether to threshold at 25 or 65?

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

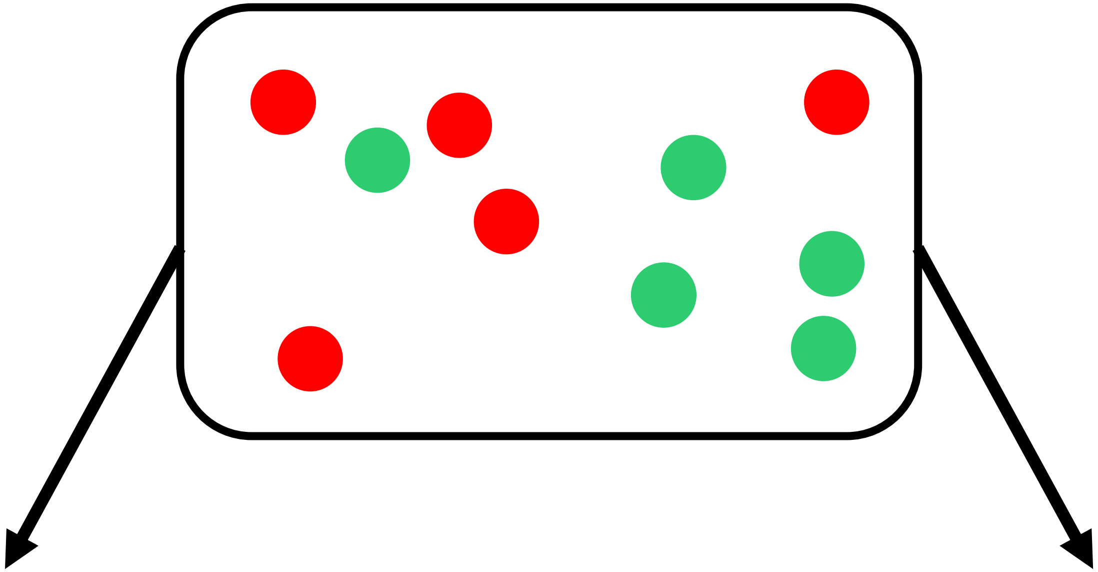
Purifying Decision Stumps

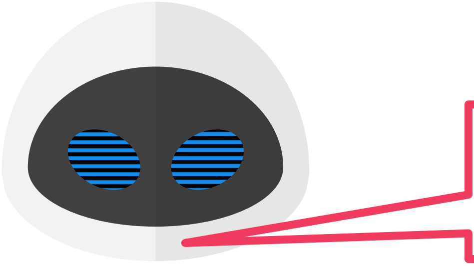


Purifying Decision Stumps



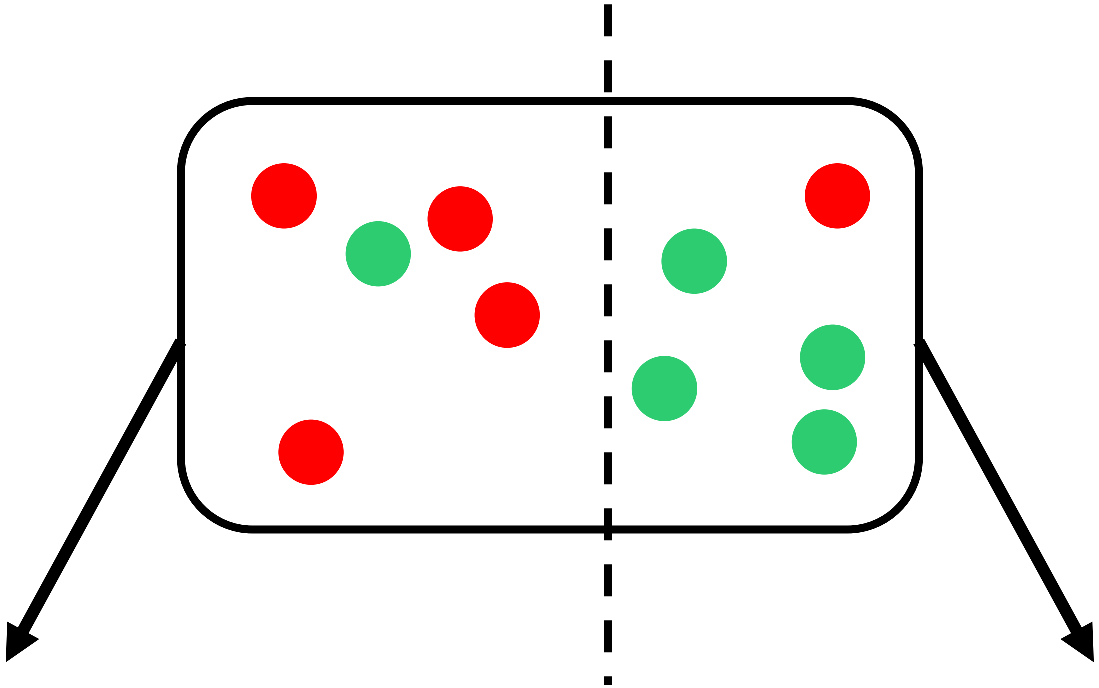
Purifying Decision Stumps

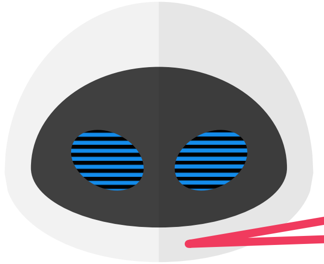




Two possible splitting directions. Let us choose the one that gives us purer children

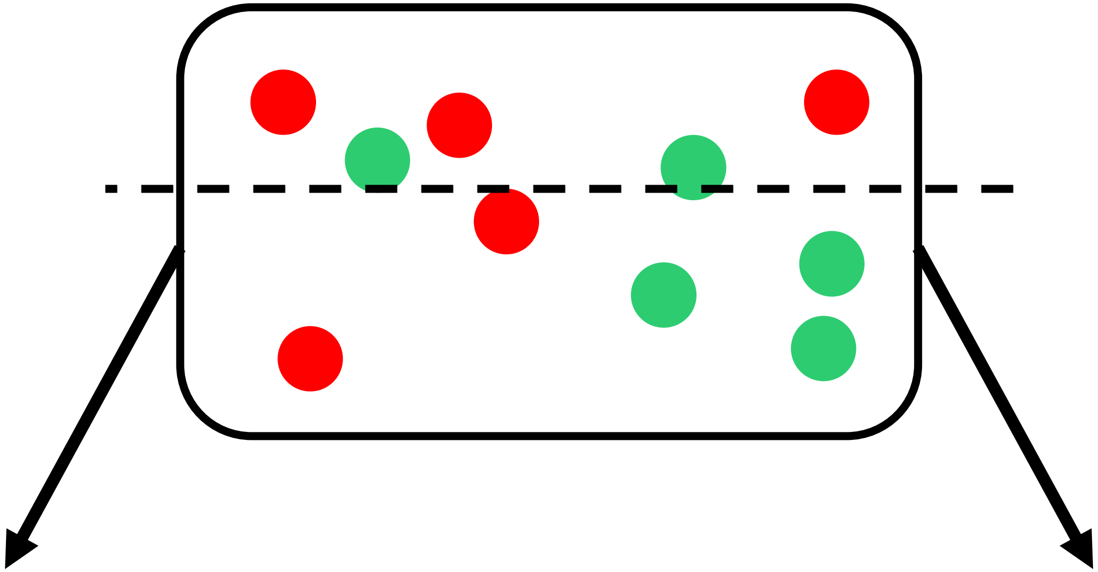
Purifying Decision Stumps

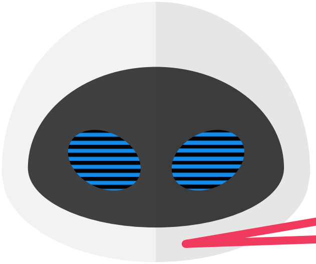




Two possible splitting directions. Let us choose the one that gives us purer children

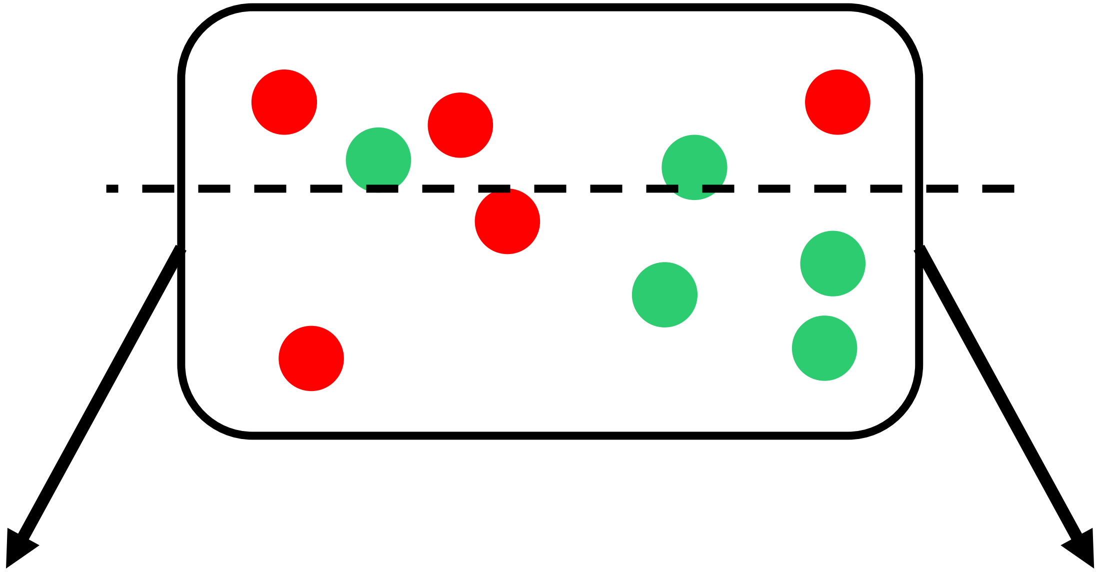
Purifying Decision Stumps

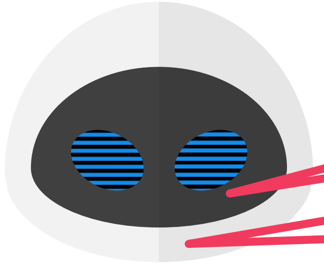




Two possible splitting directions. Let us choose the one that gives us purer children

Purifying Decision Stumps

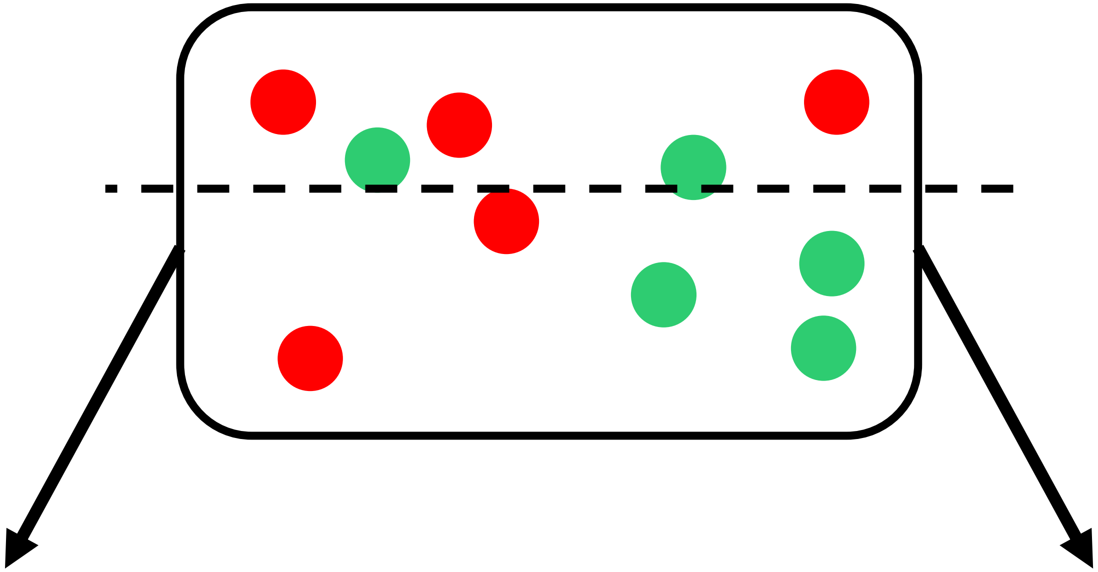


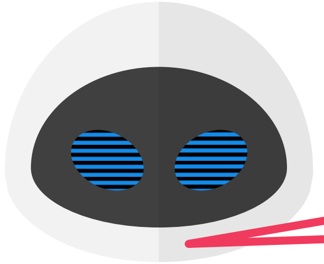


Search purest horizontal split by going over all possible thresholds and checking

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

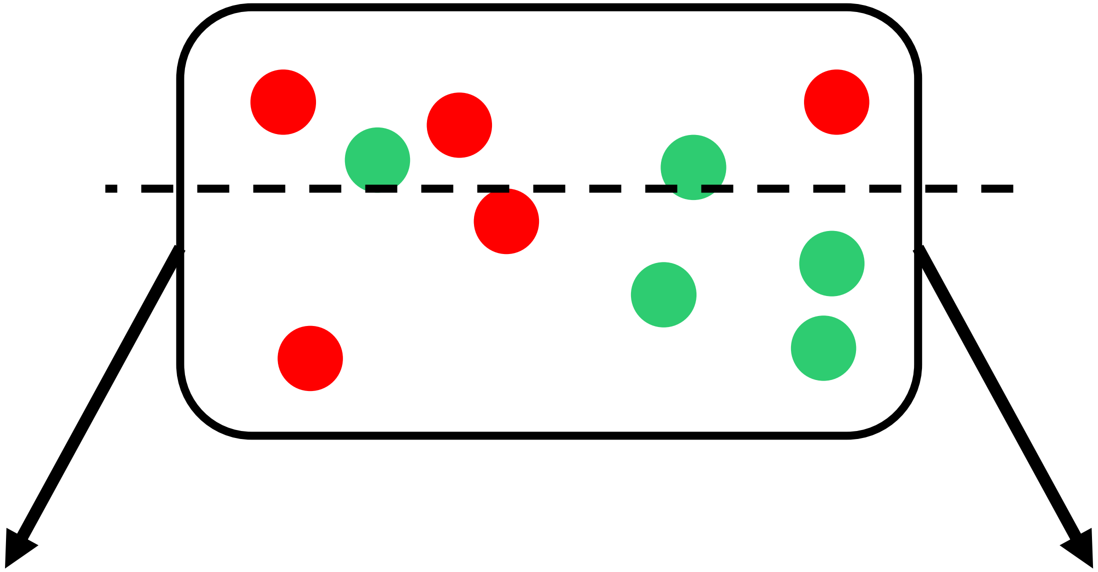
Purifying Decision Stumps

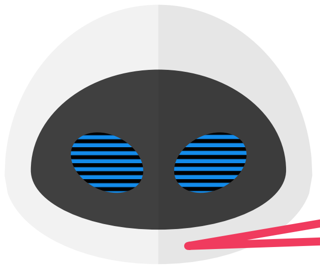




Two possible splitting directions. Let us choose the one that gives us purer children

Purifying Decision Stumps

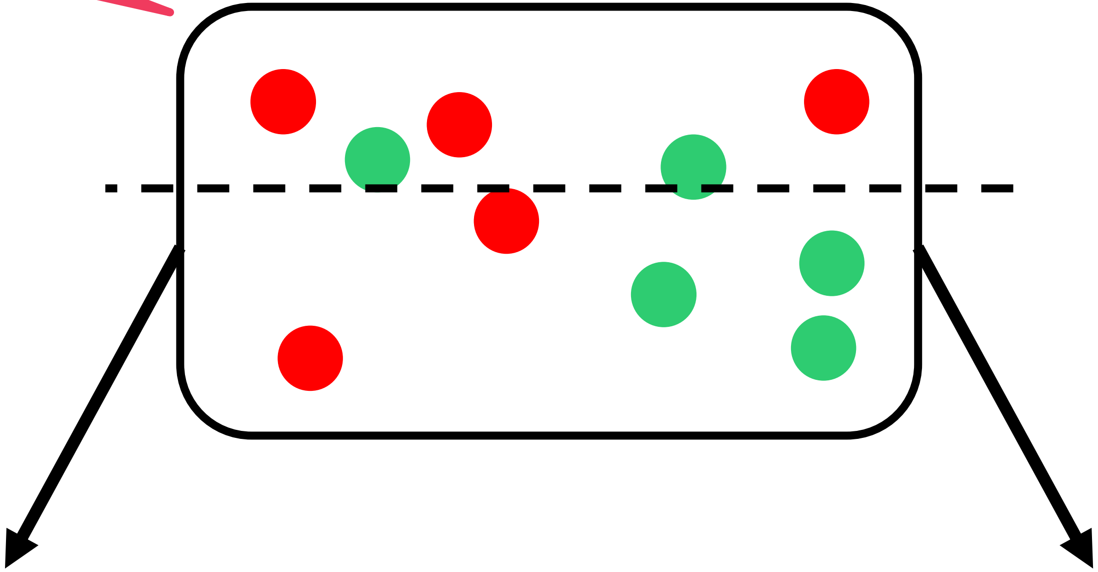


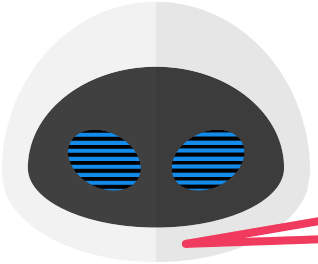


Two possible splitting directions. Let us choose the one that gives us purer children

Purifying Decision Stumps

Purest horizontal split



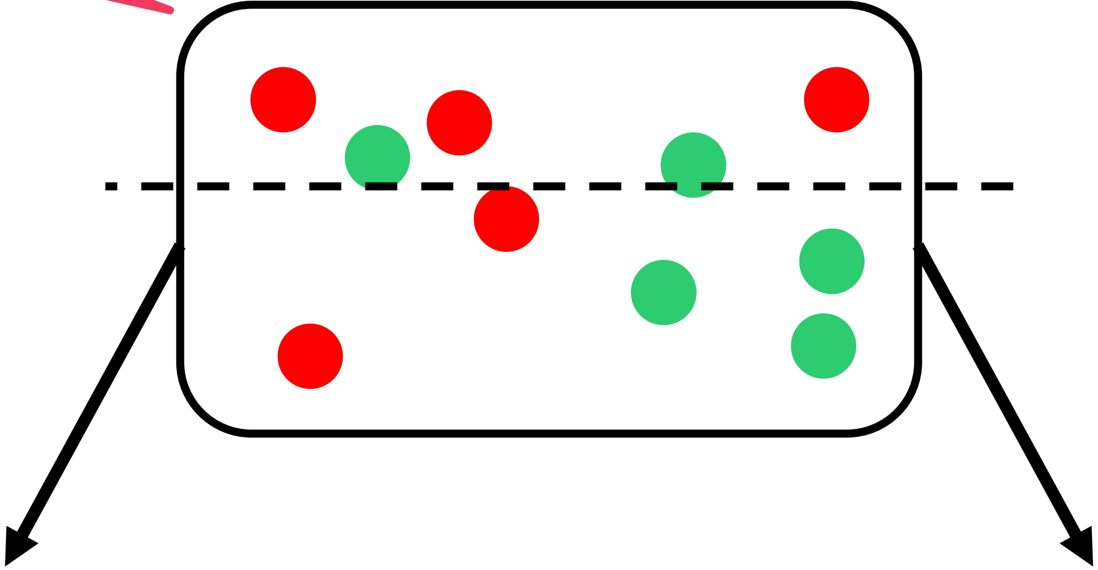


Two possible splitting directions. Let us choose the one that gives us purer children

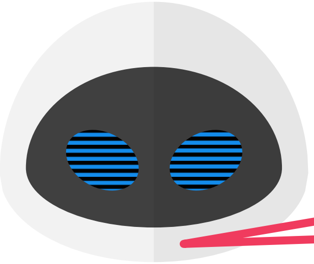
Purifying Decision Stumps

Purest horizontal split

Horizontal Split



Left Right

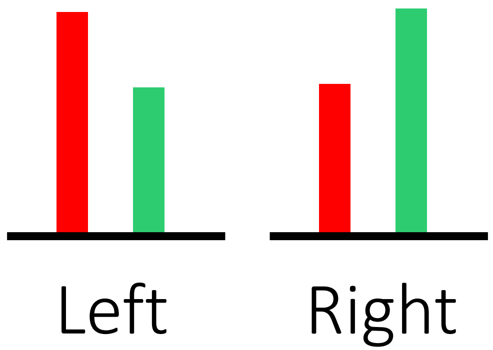


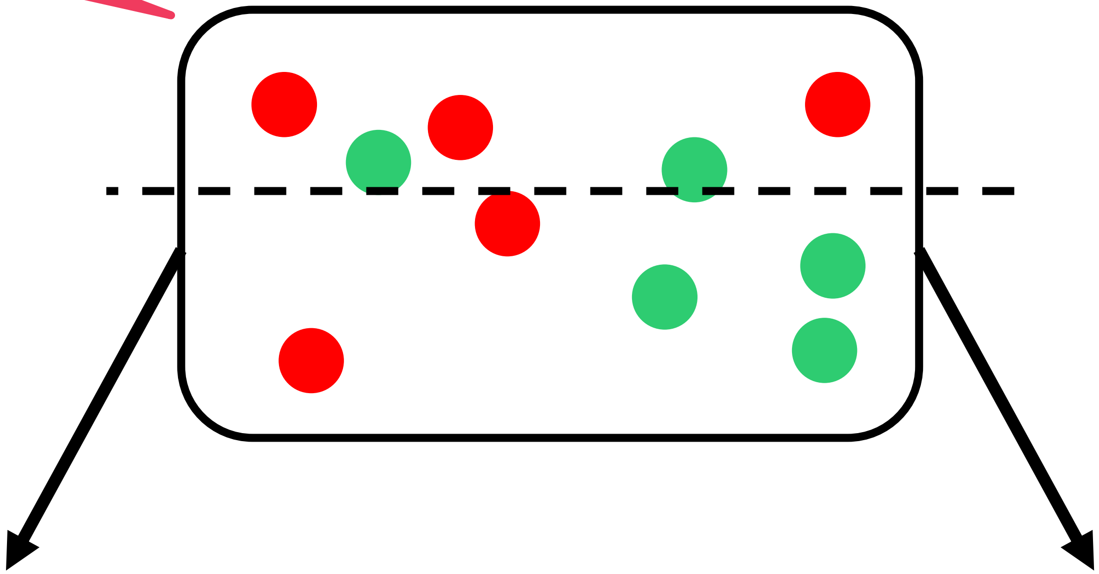
Two possible splitting directions. Let us choose the one that gives us purer children

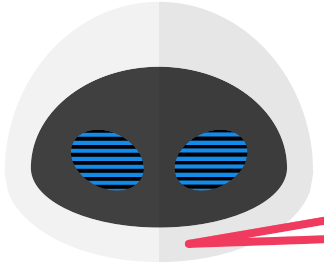
Purifying Decision Stumps

Purest horizontal split

Horizontal Split



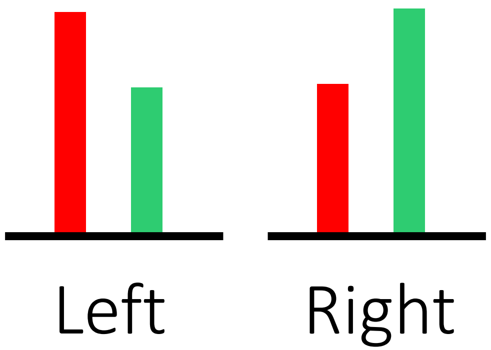


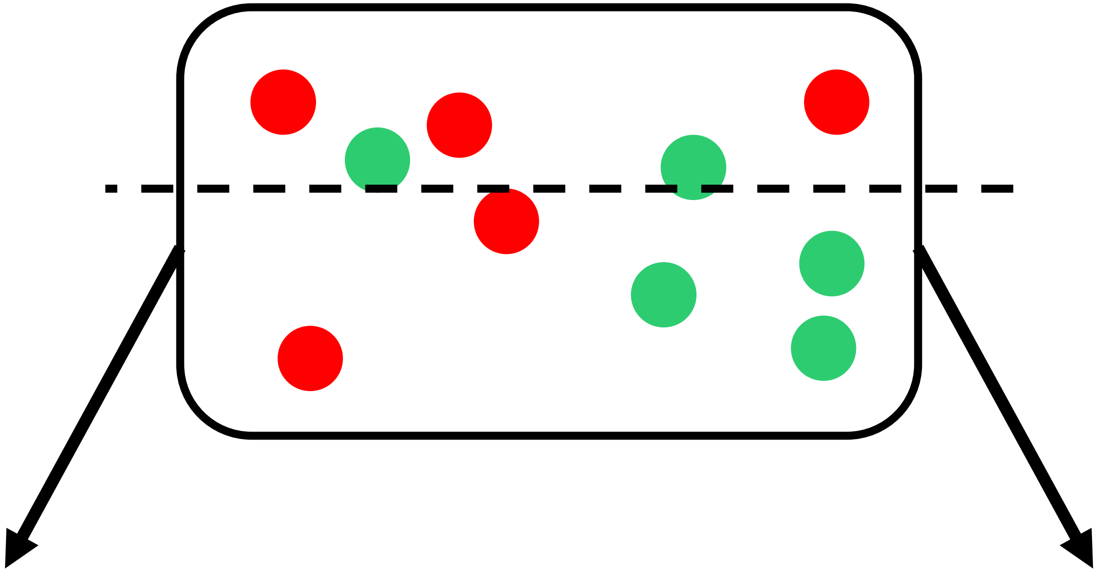


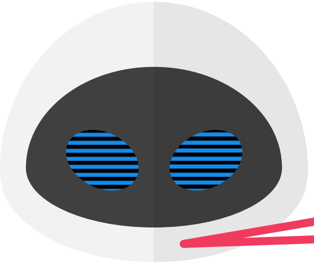
Two possible splitting directions. Let us choose the one that gives us purer children

Purifying Decision Stumps

Horizontal Split



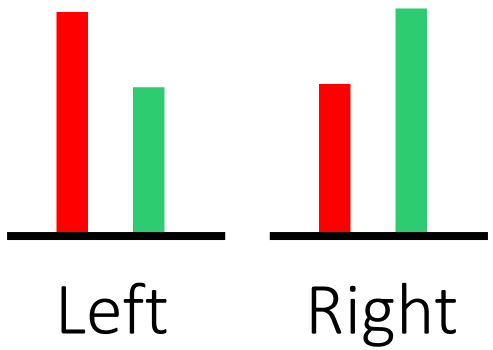


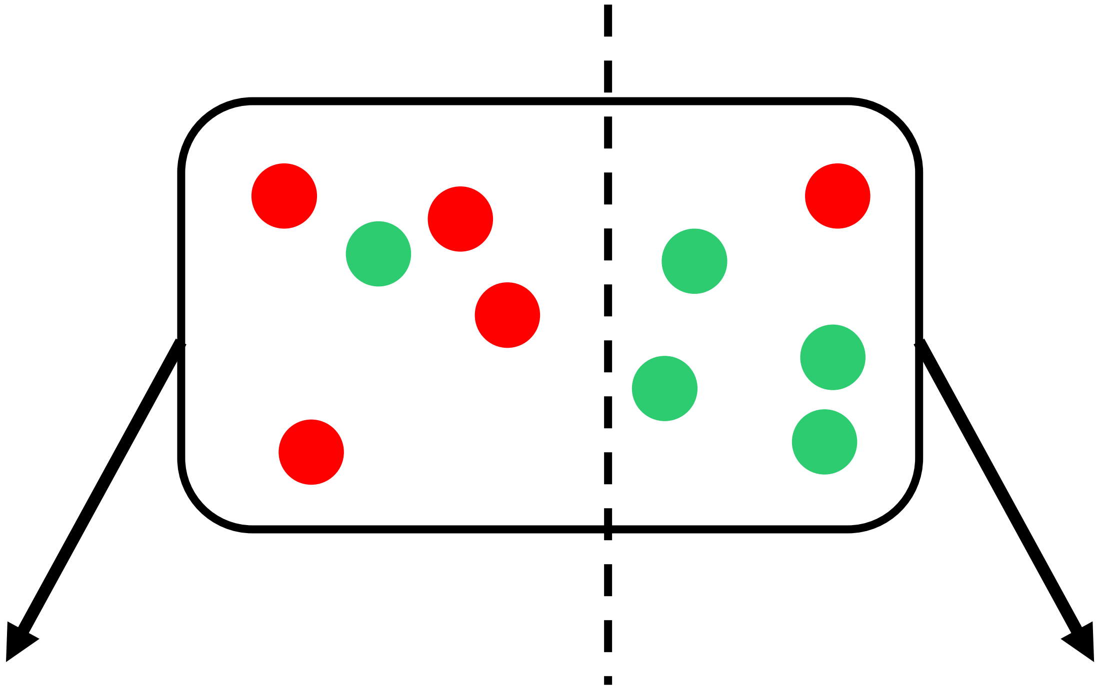


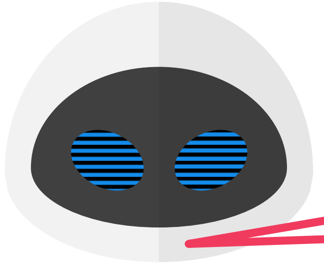
Two possible splitting directions. Let us choose the one that gives us purer children

Purifying Decision Stumps

Horizontal Split



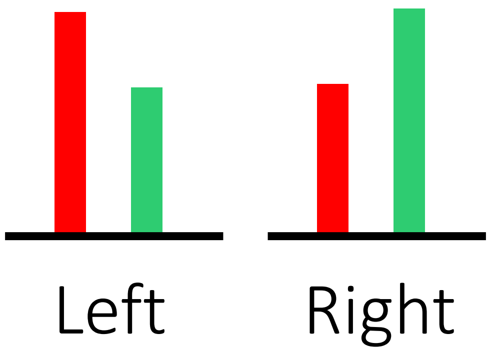


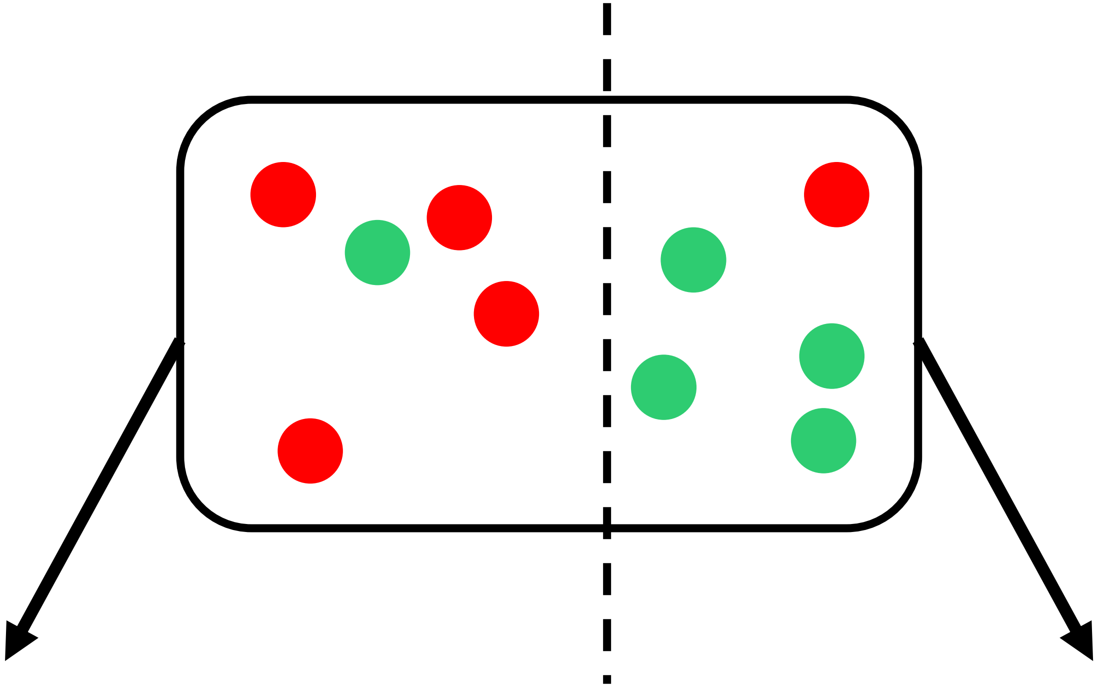


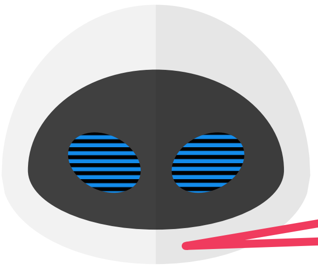
Two possible splitting directions. Let us choose the one that gives us purer children

Purifying Decision Stumps

Horizontal Split





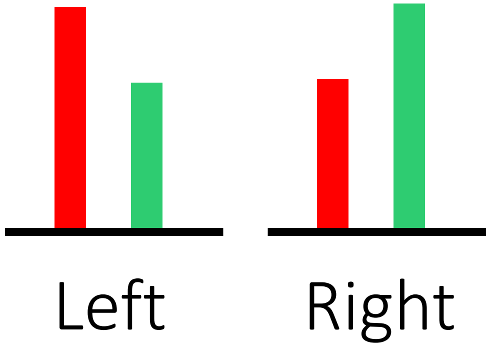


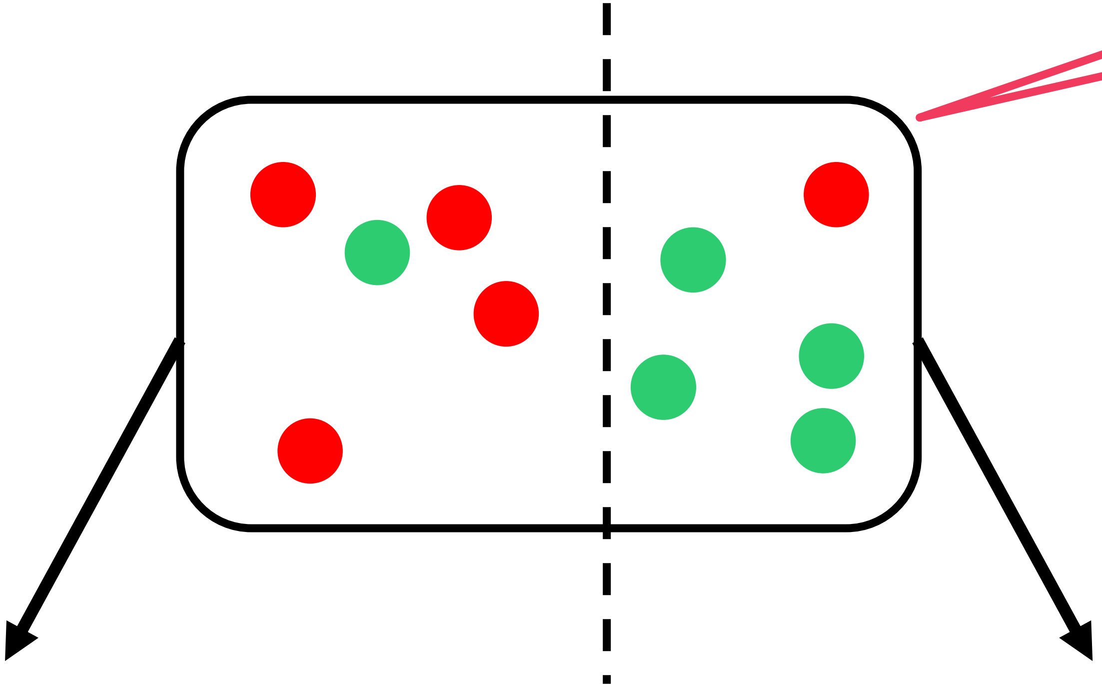
Two possible splitting directions. Let us choose the one that gives us purer children

Purifying Decision Stumps

Purest vertical split

Horizontal Split

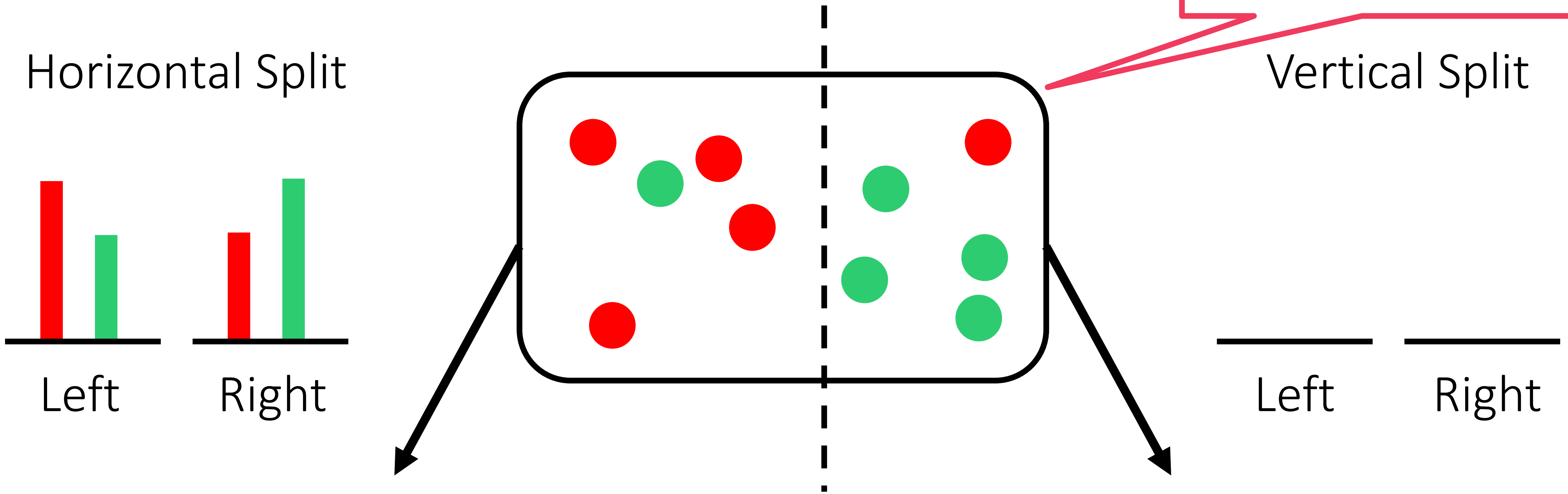


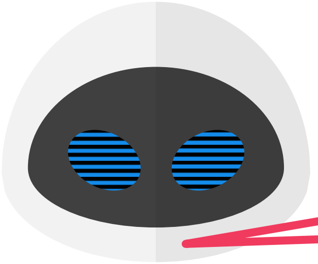


Two possible splitting directions. Let us choose the one that gives us purer children

Purifying Decision Stumps

Purest vertical split

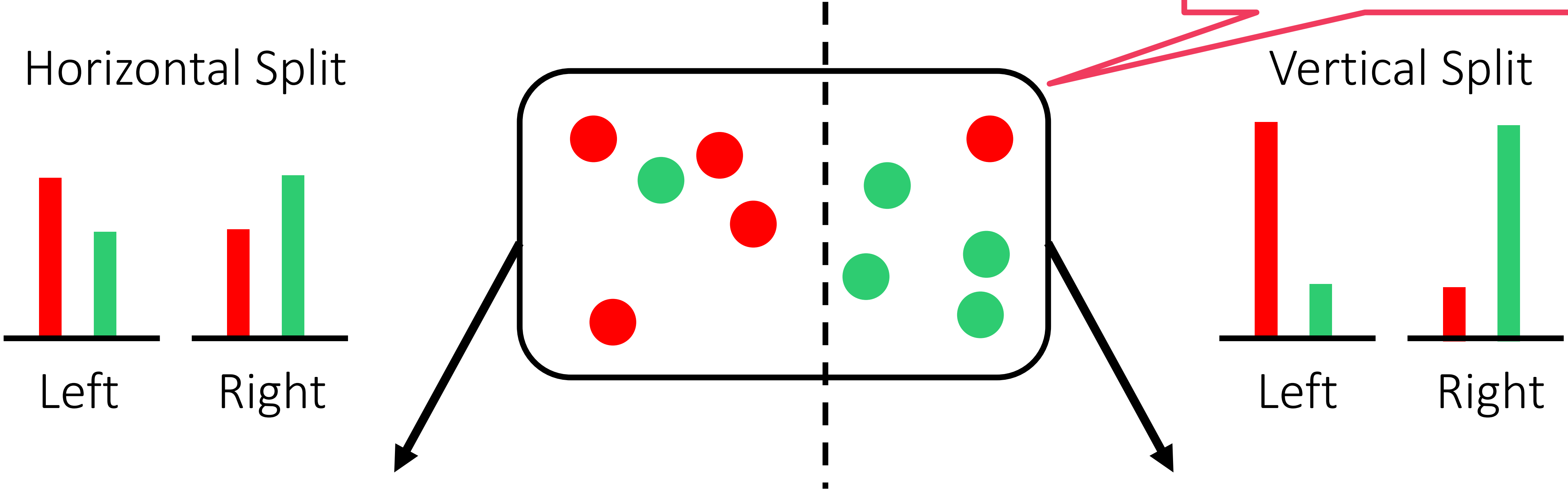


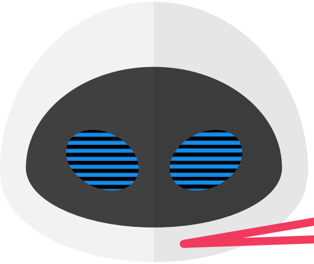


Two possible splitting directions. Let us choose the one that gives us purer children

Purifying Decision Stumps

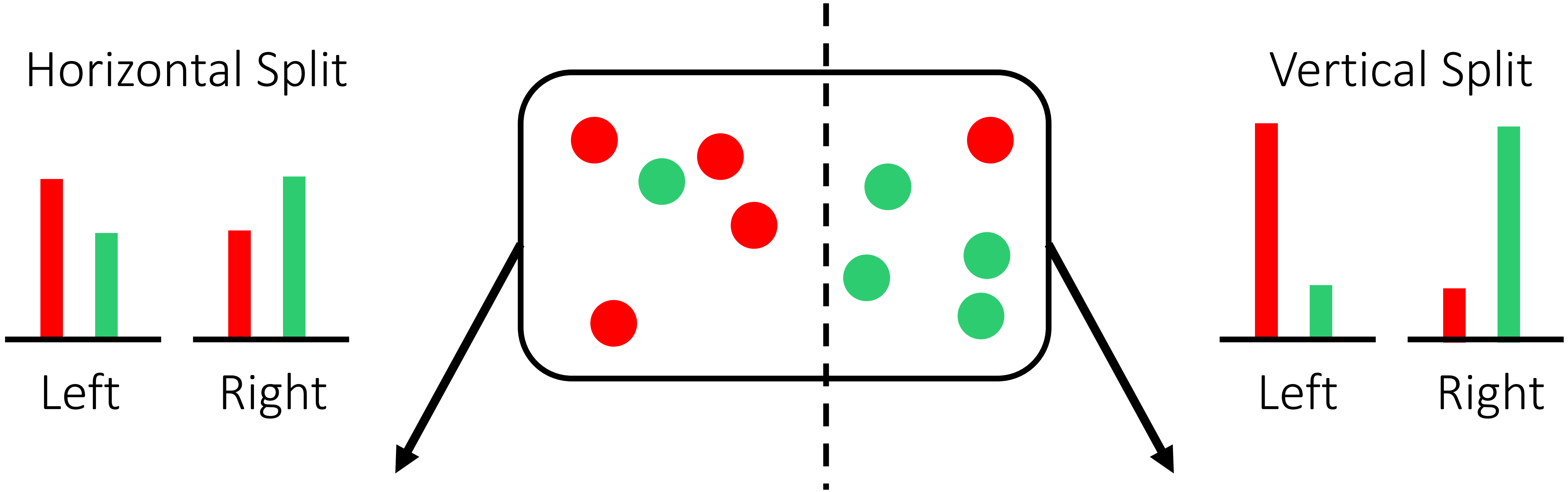
Purest vertical split

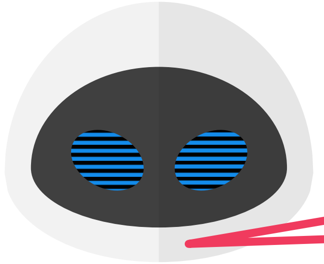




Two possible splitting directions. Let us choose the one that gives us purer children

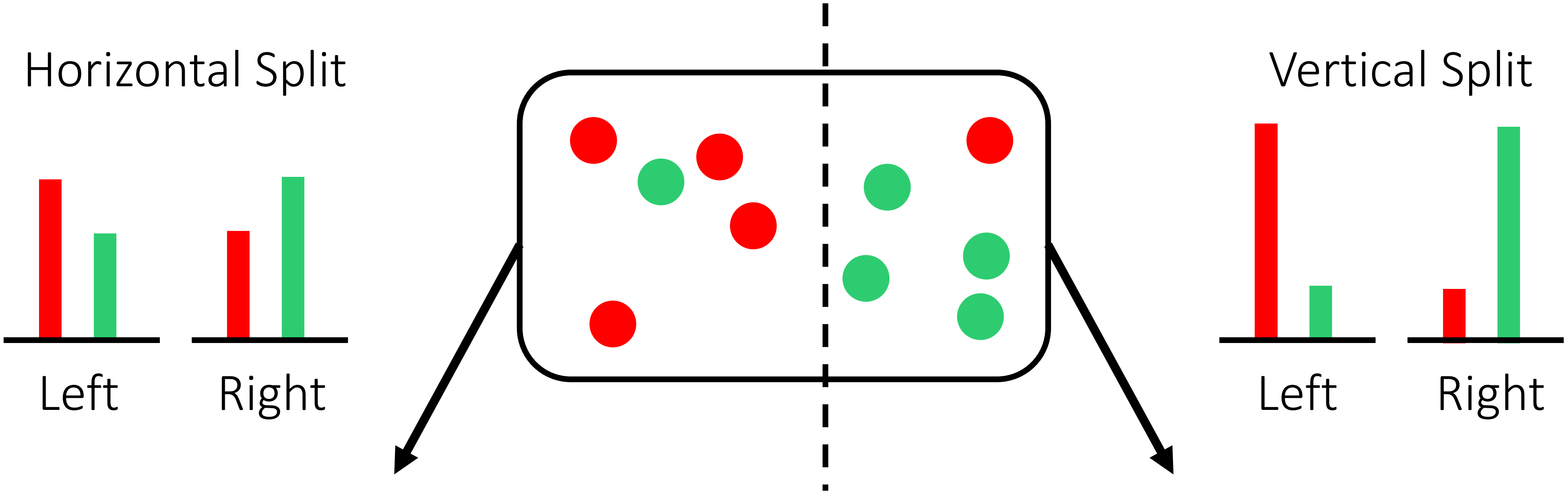
Purifying Decision Stumps

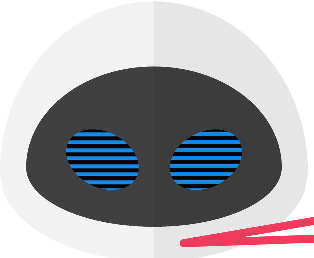




Two possible splitting directions. Let us choose the one that gives us purer children

Purifying Decision Stumps

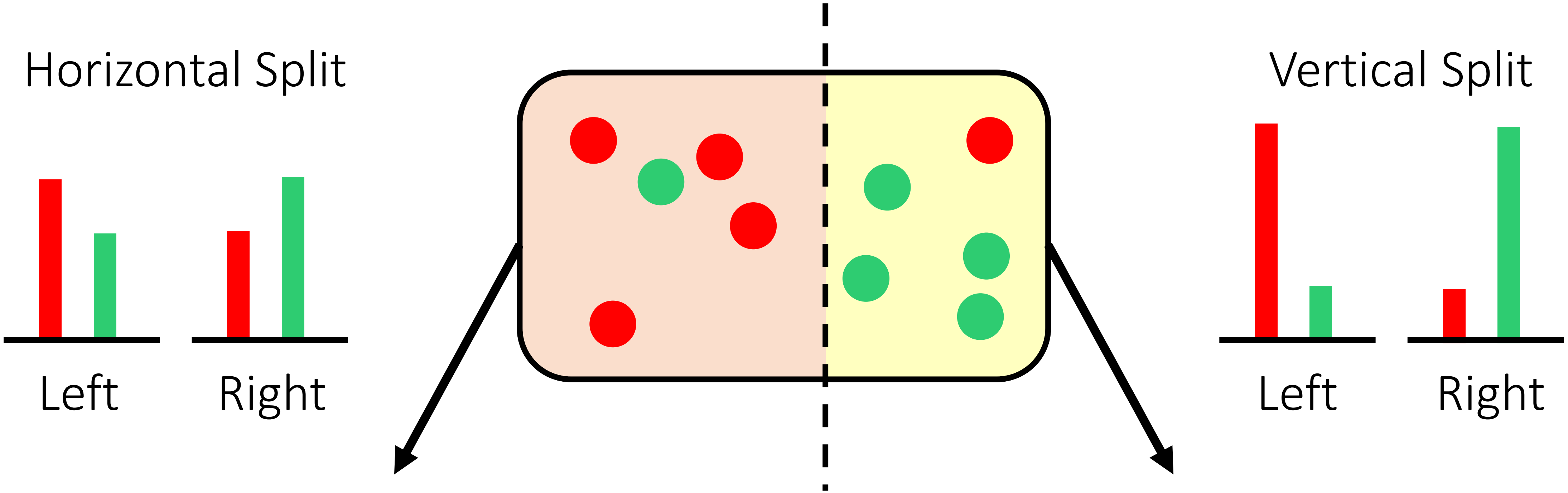


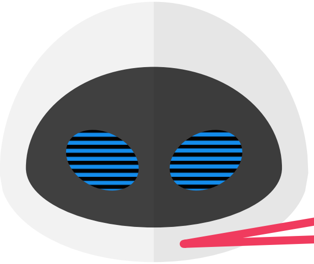


Two possible splitting directions. Let us choose the one that gives us purer children

Vertical split is more pure – lets go with it!

Purifying Decision Stumps

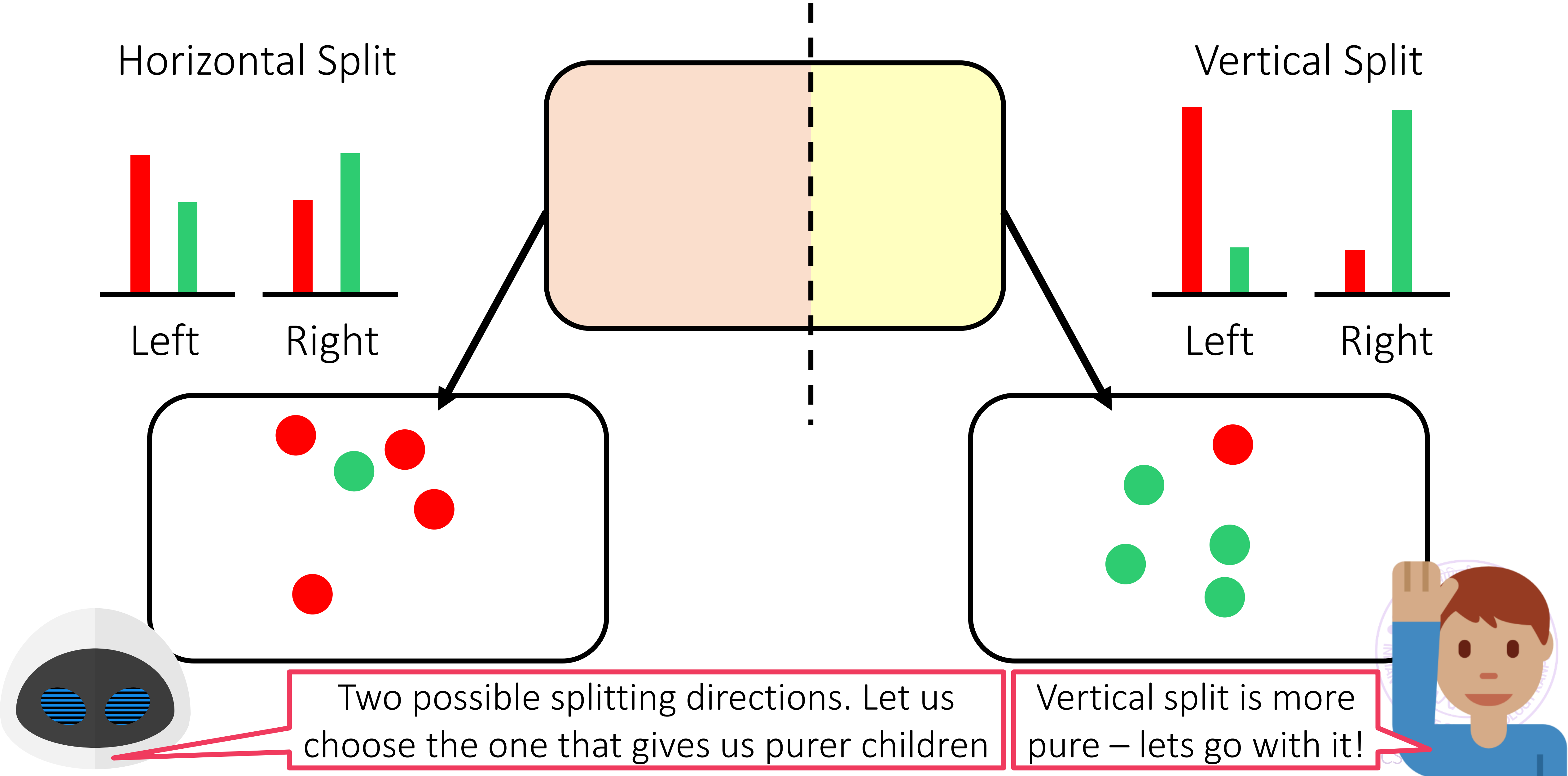




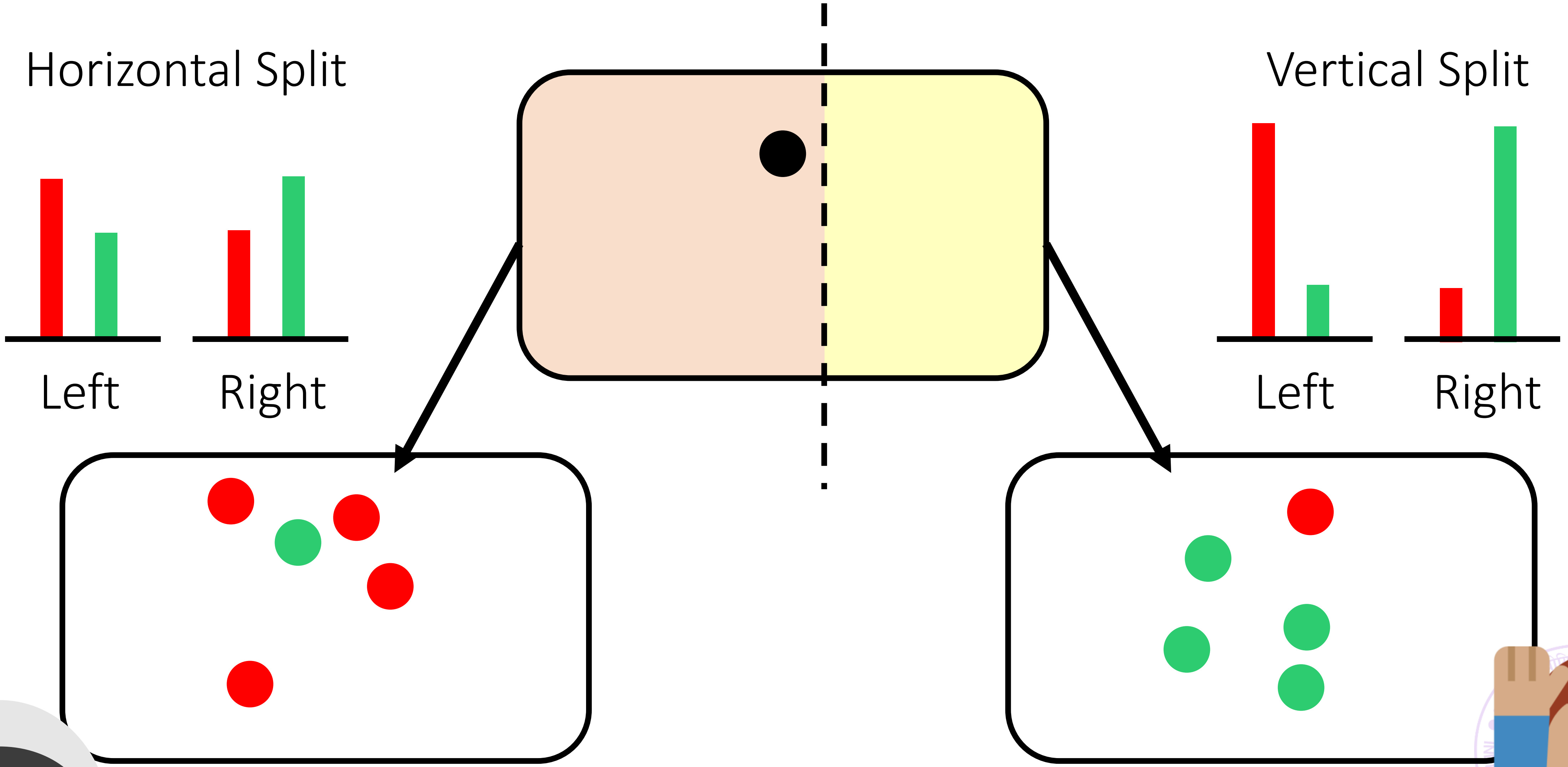
Two possible splitting directions. Let us choose the one that gives us purer children

Vertical split is more pure – lets go with it!

Purifying Decision Stumps



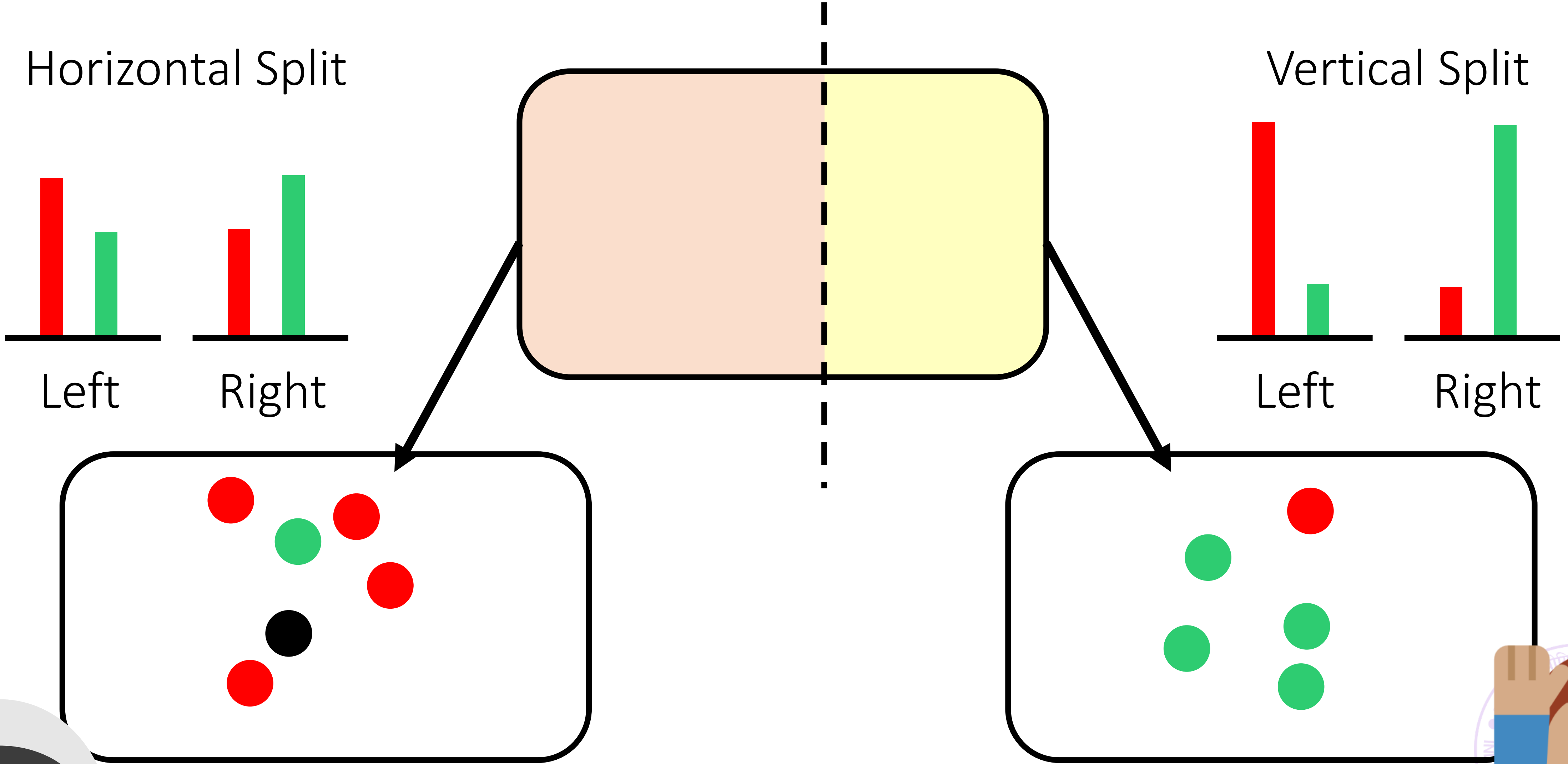
Purifying Decision Stumps

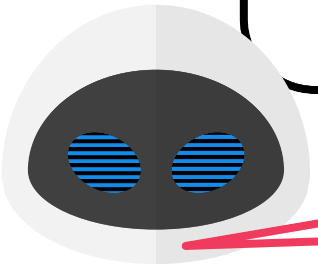


Two possible splitting directions. Let us choose the one that gives us purer children

Vertical split is more pure – lets go with it!

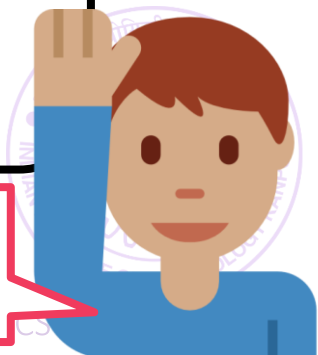
Purifying Decision Stumps





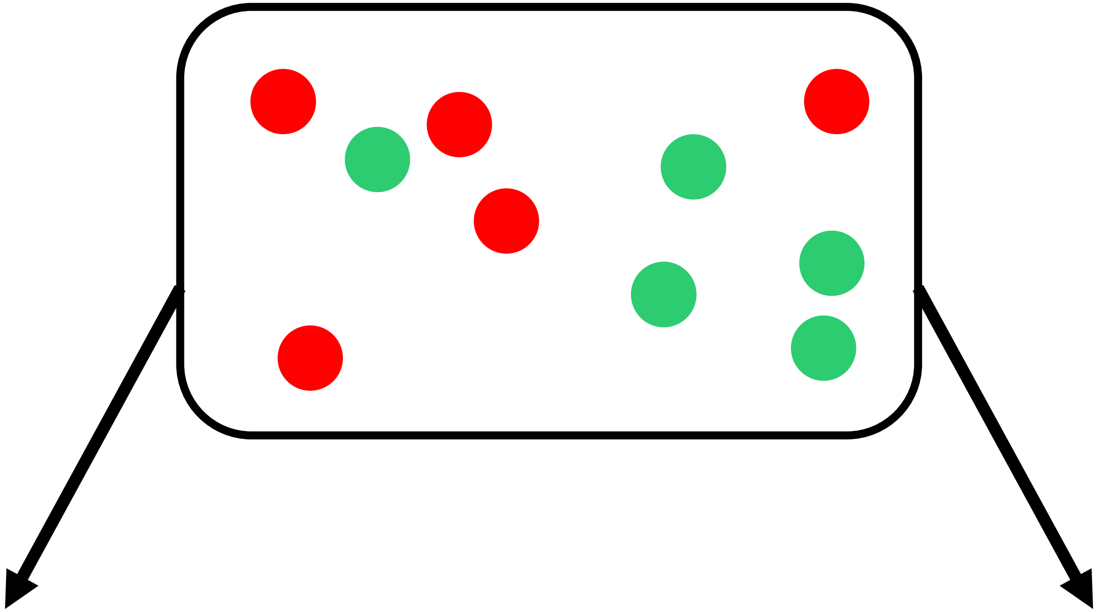
Two possible splitting directions. Let us choose the one that gives us purer children

Vertical split is more pure – lets go with it!

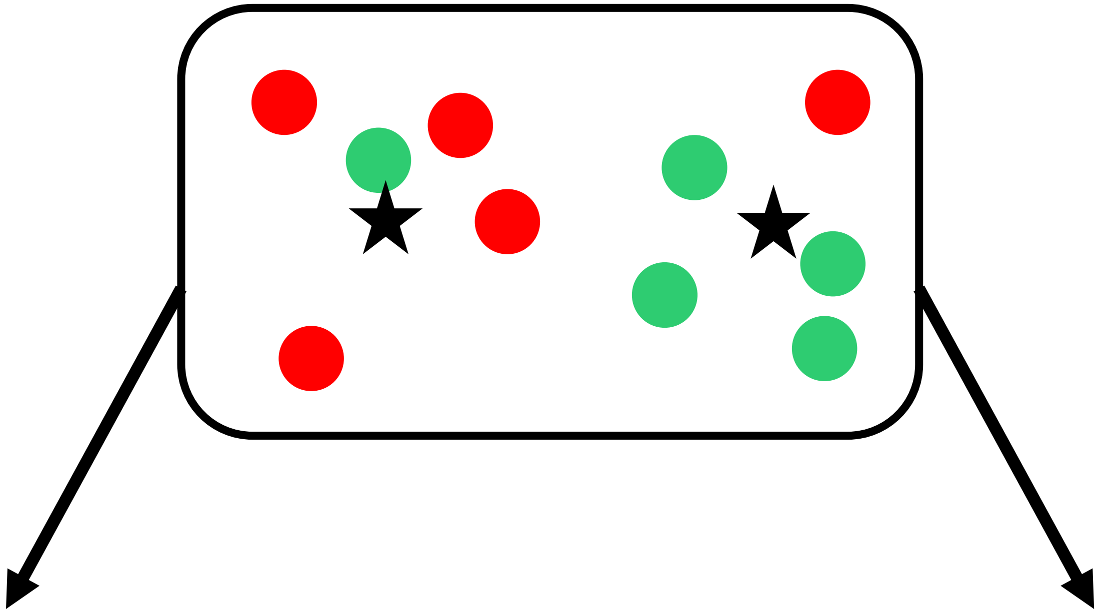


Node splitting via LwP/linear classifiers

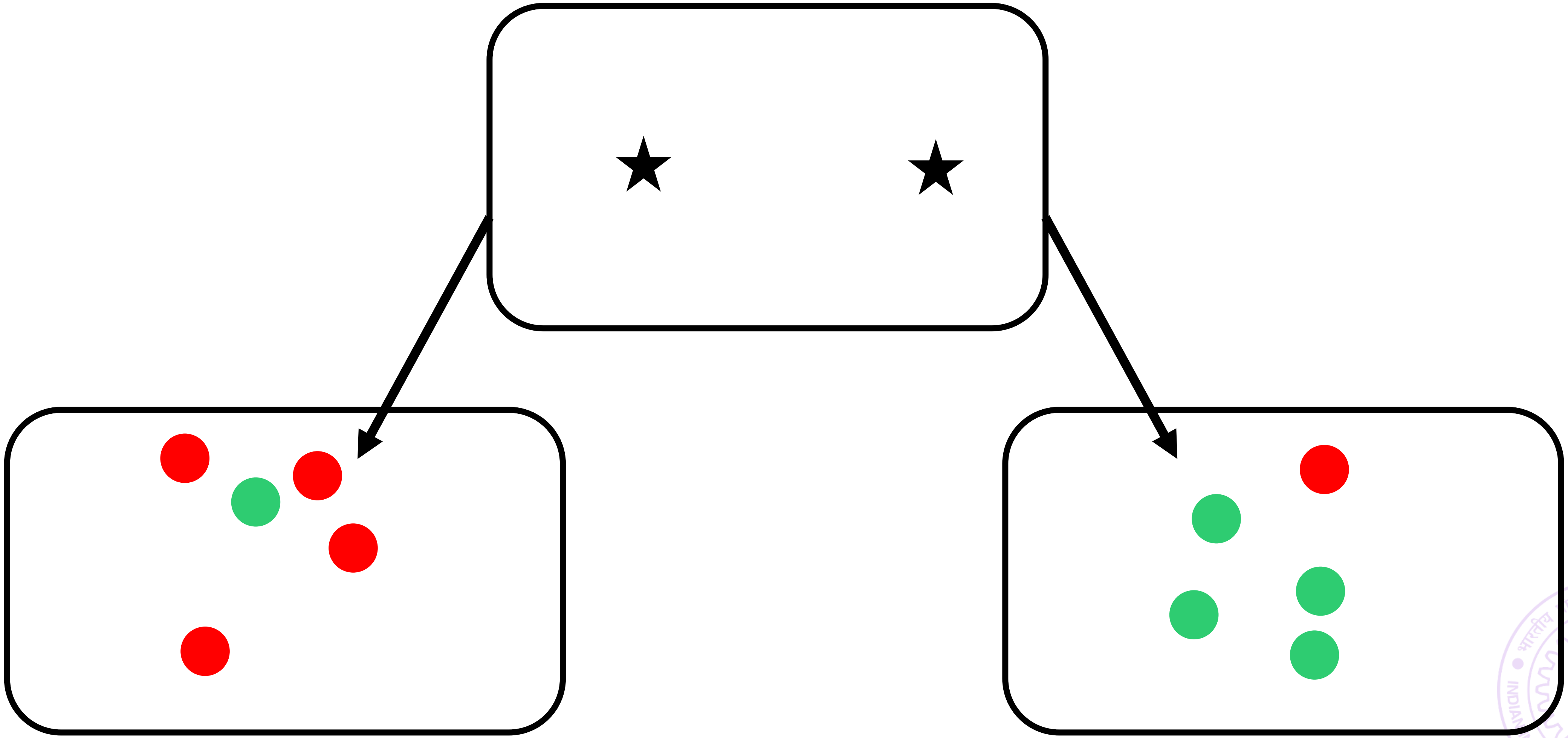
Node splitting via LwP/linear classifiers



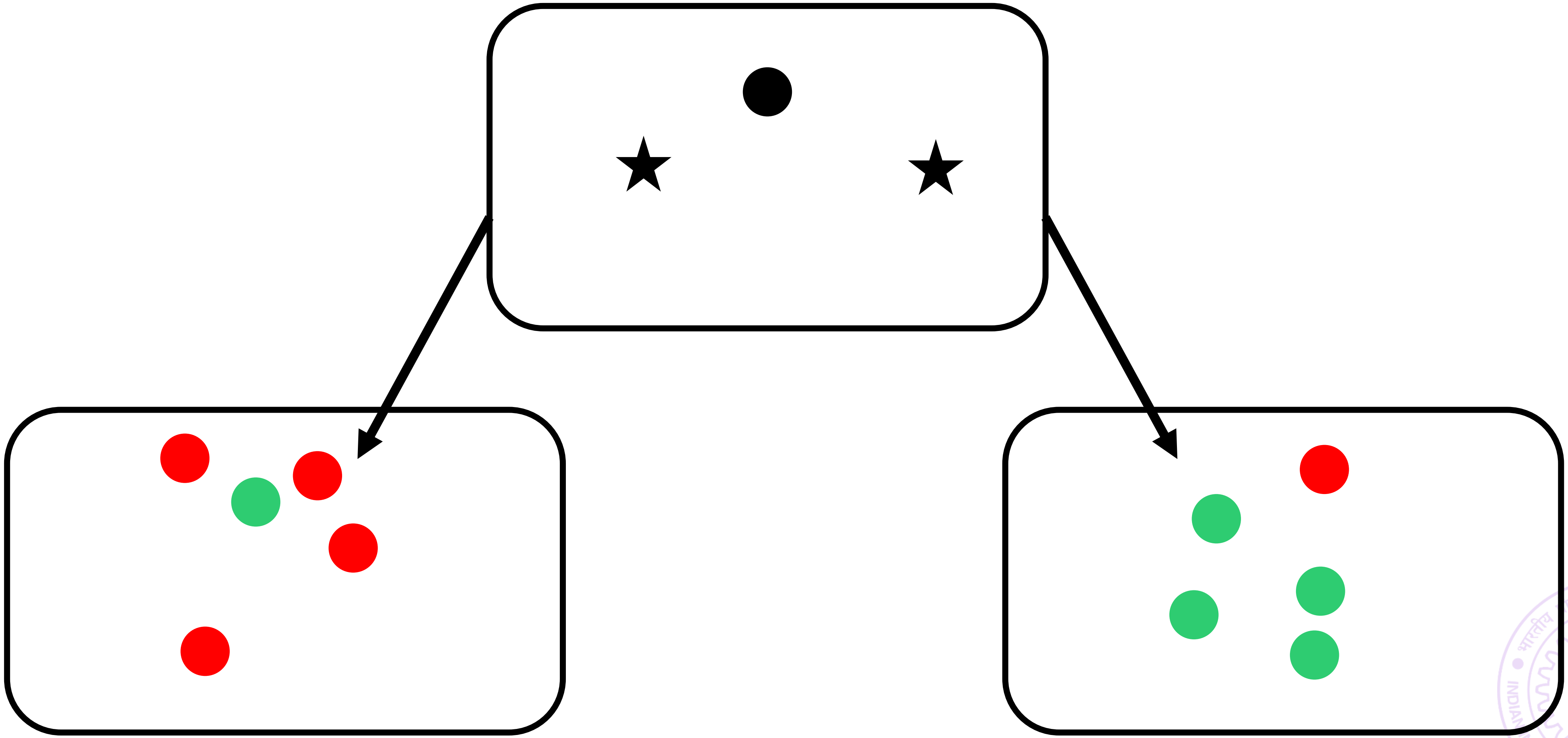
Node splitting via LwP/linear classifiers



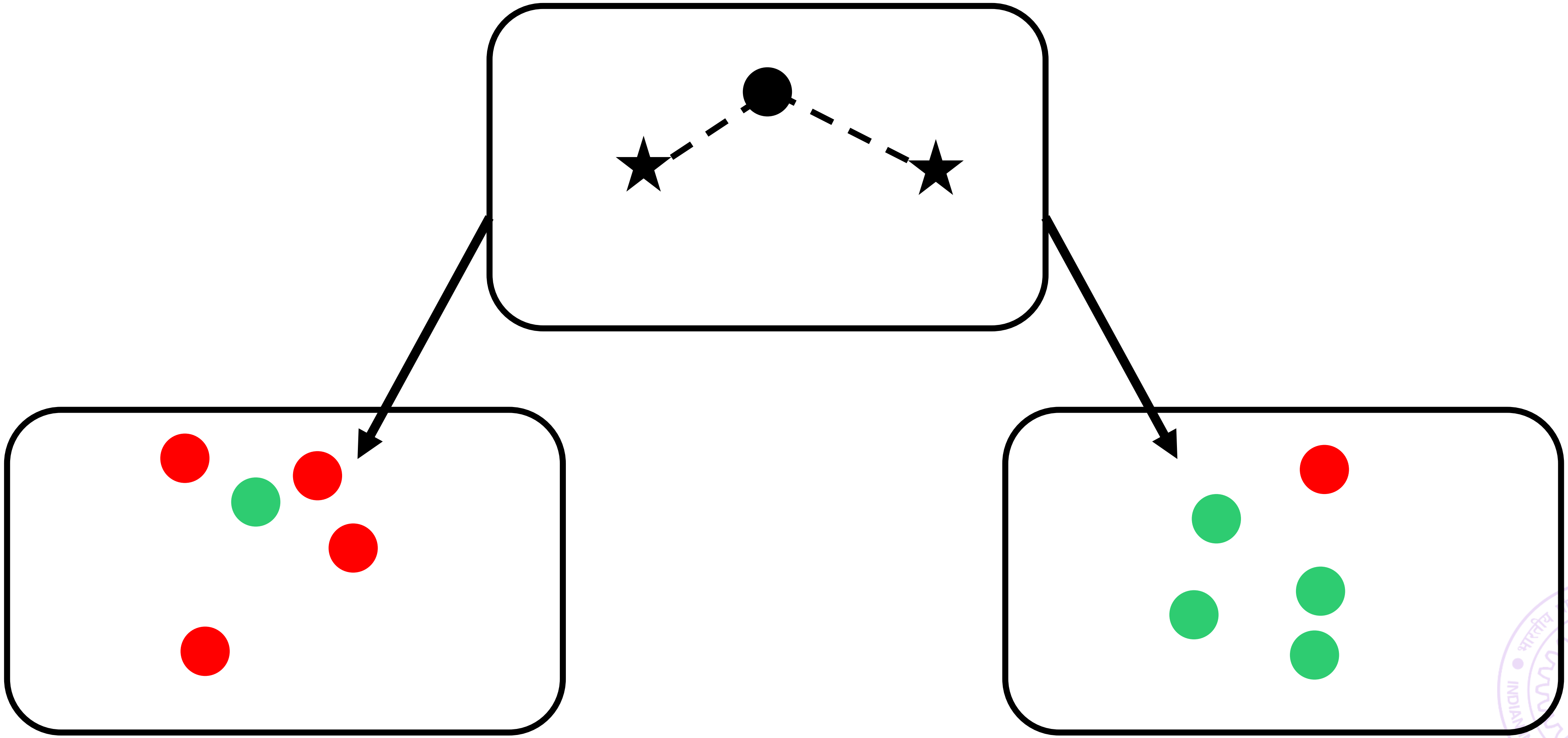
Node splitting via LwP/linear classifiers



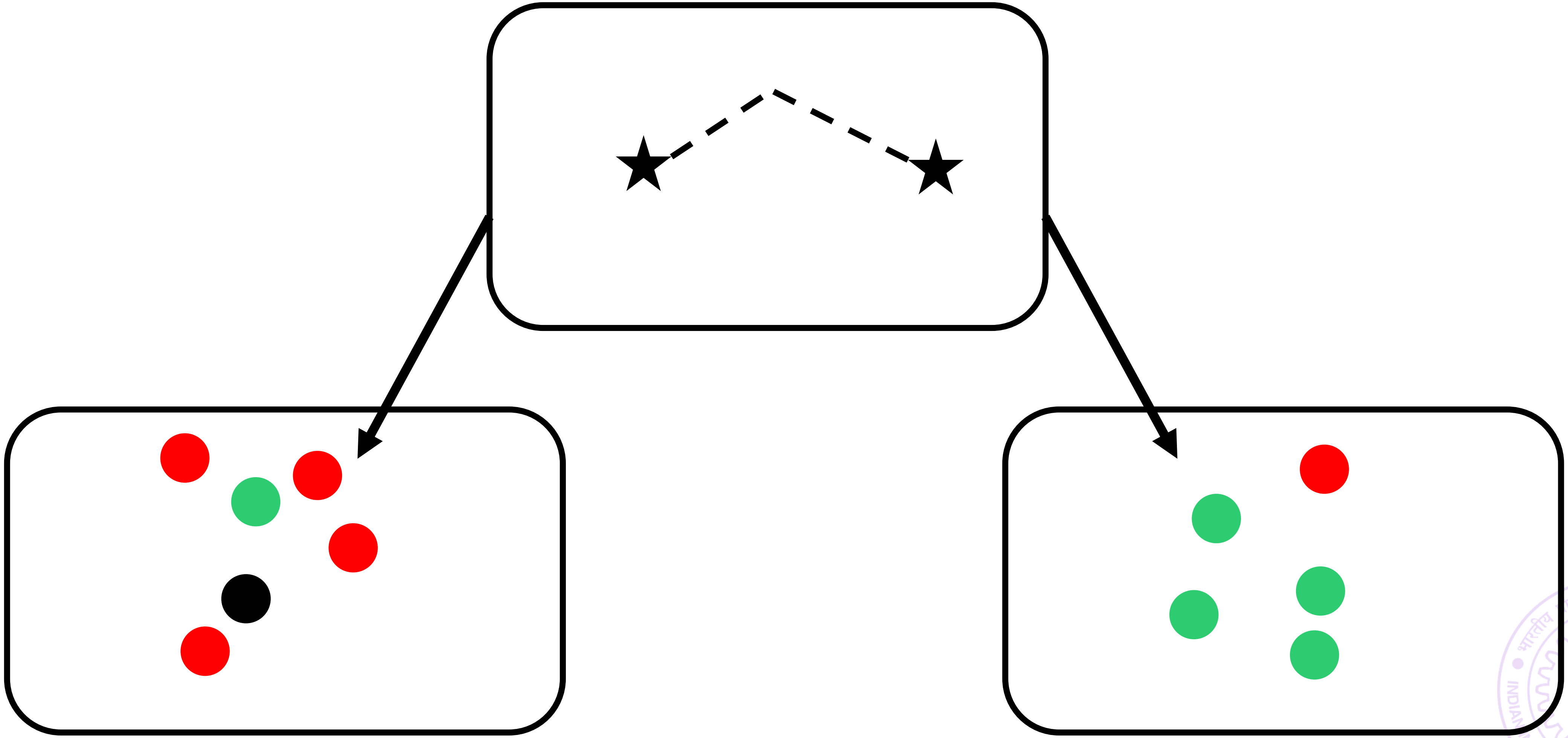
Node splitting via LwP/linear classifiers



Node splitting via LwP/linear classifiers

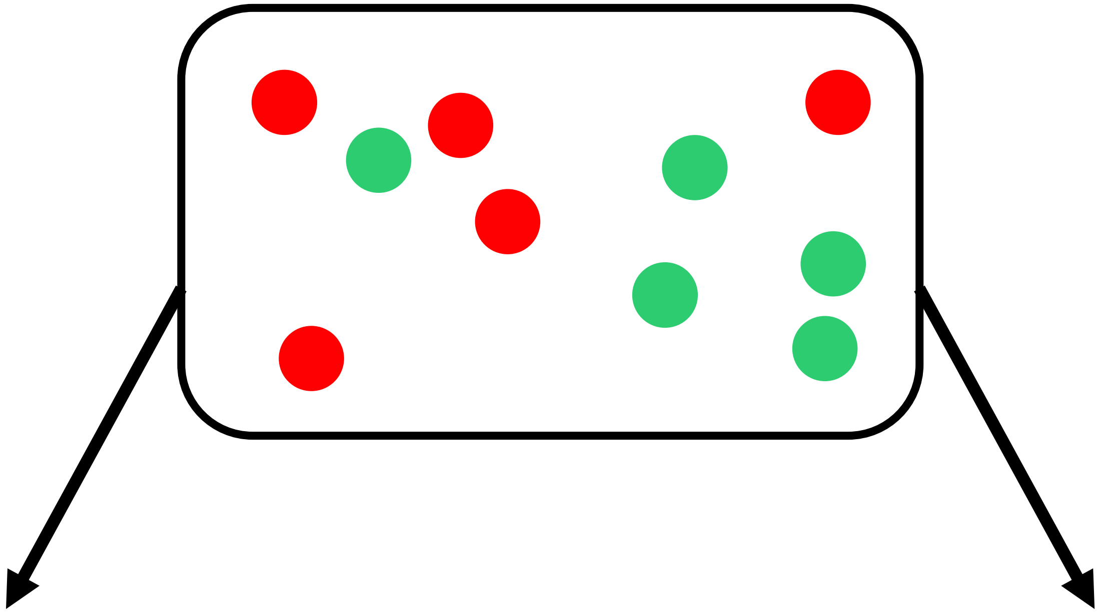


Node splitting via LwP/linear classifiers

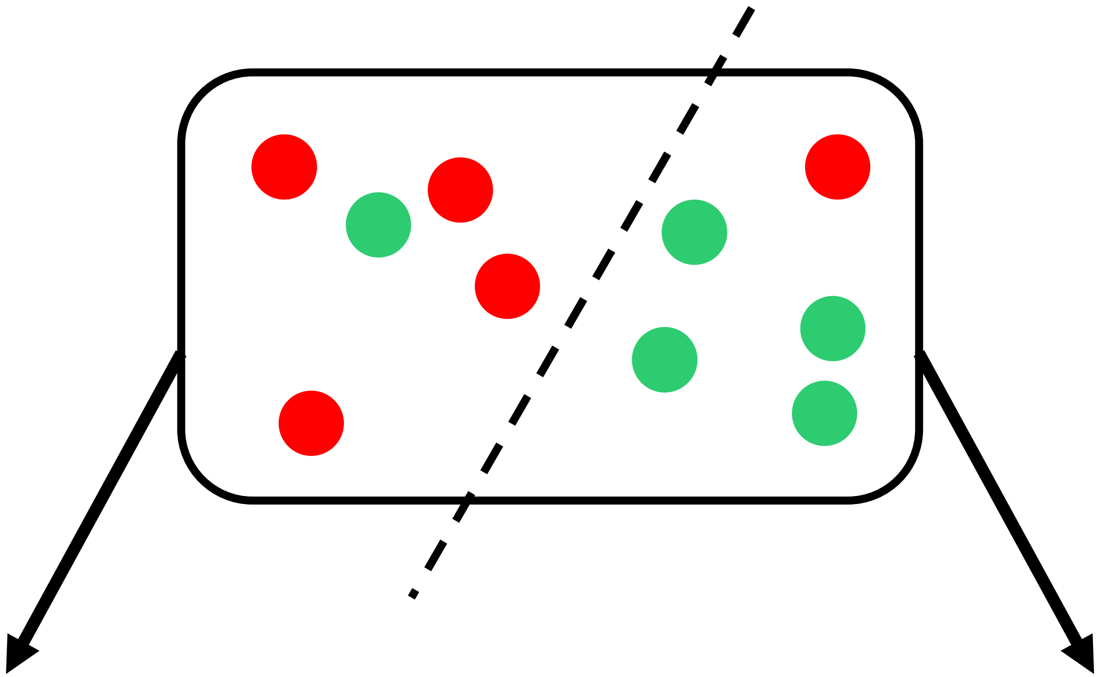


Node splitting via LwP/linear classifiers

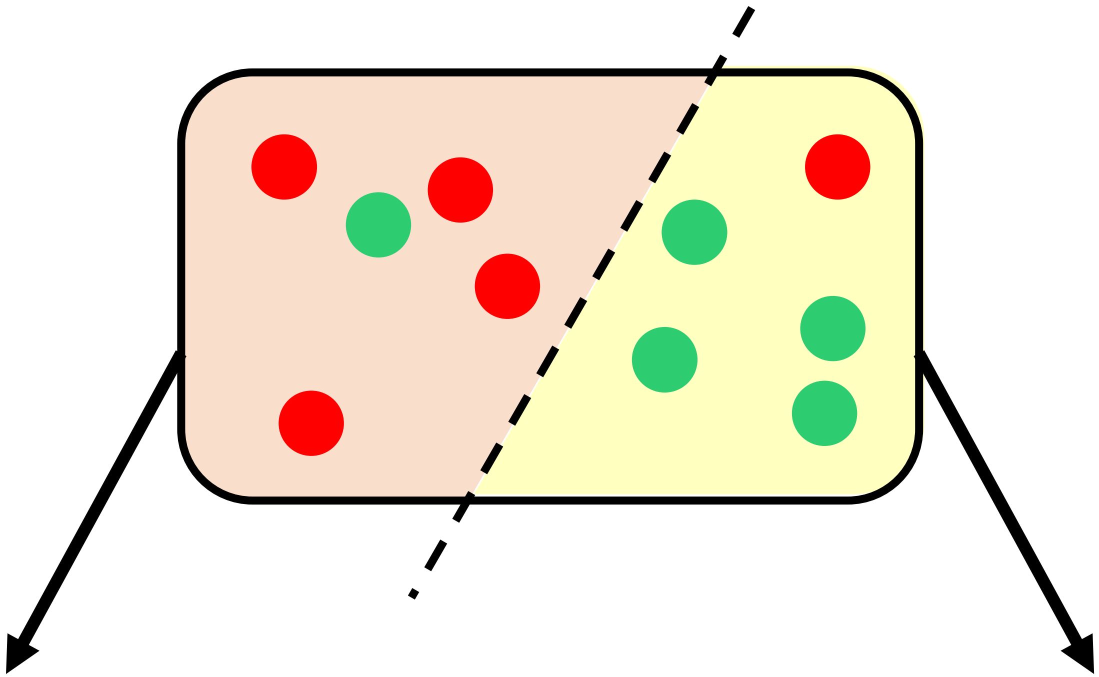
Node splitting via LwP/linear classifiers



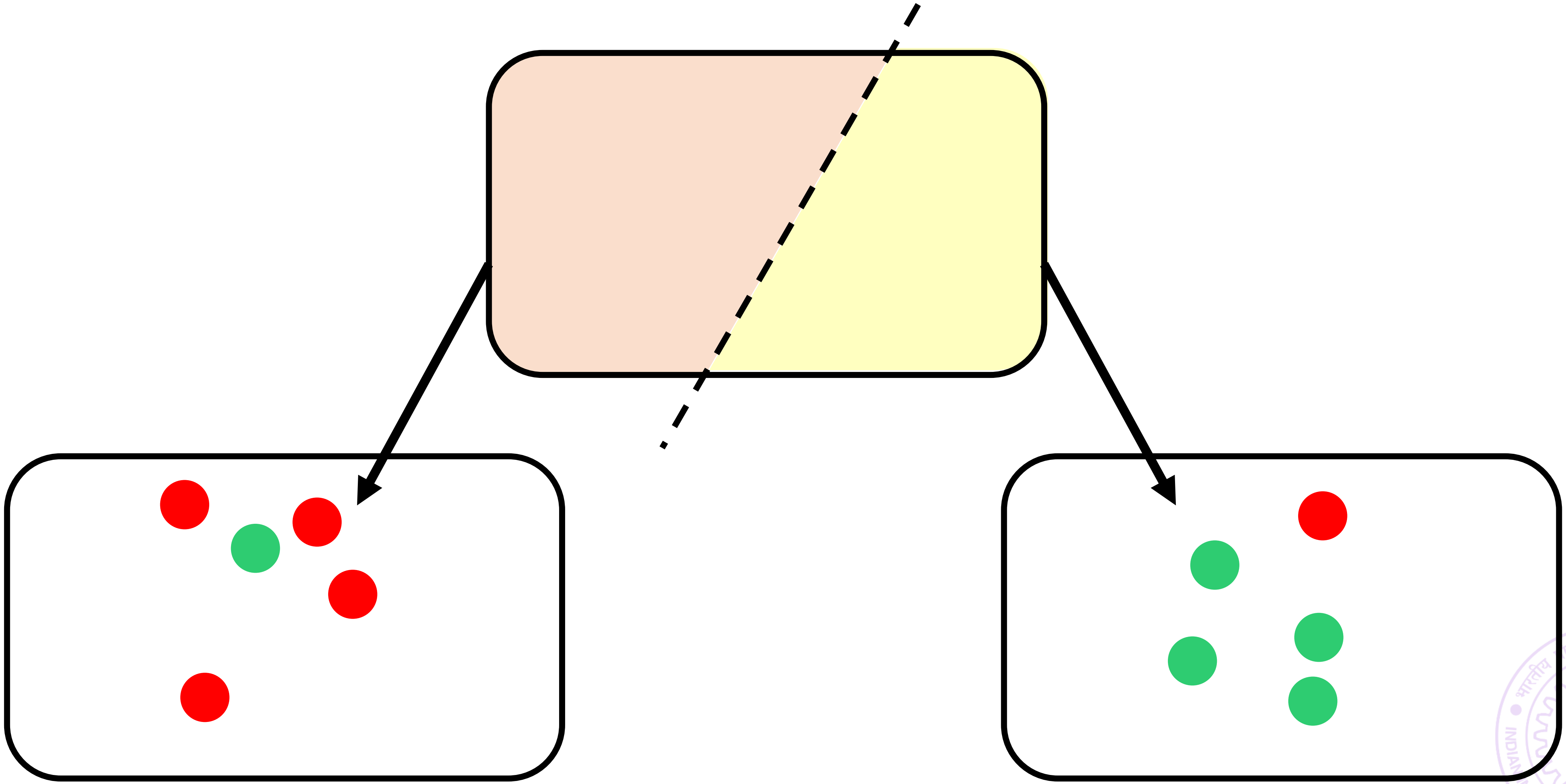
Node splitting via LwP/linear classifiers



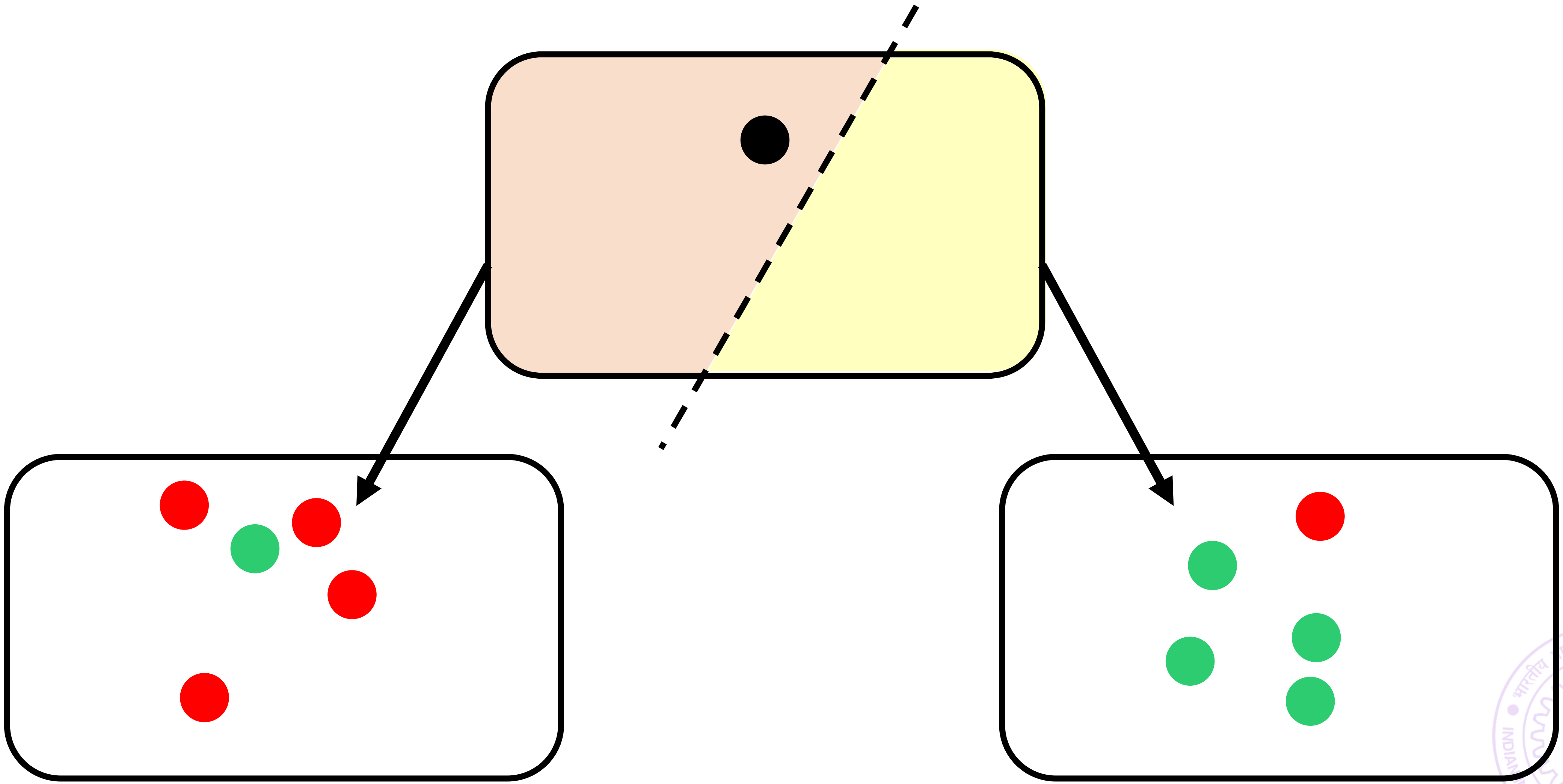
Node splitting via LwP/linear classifiers



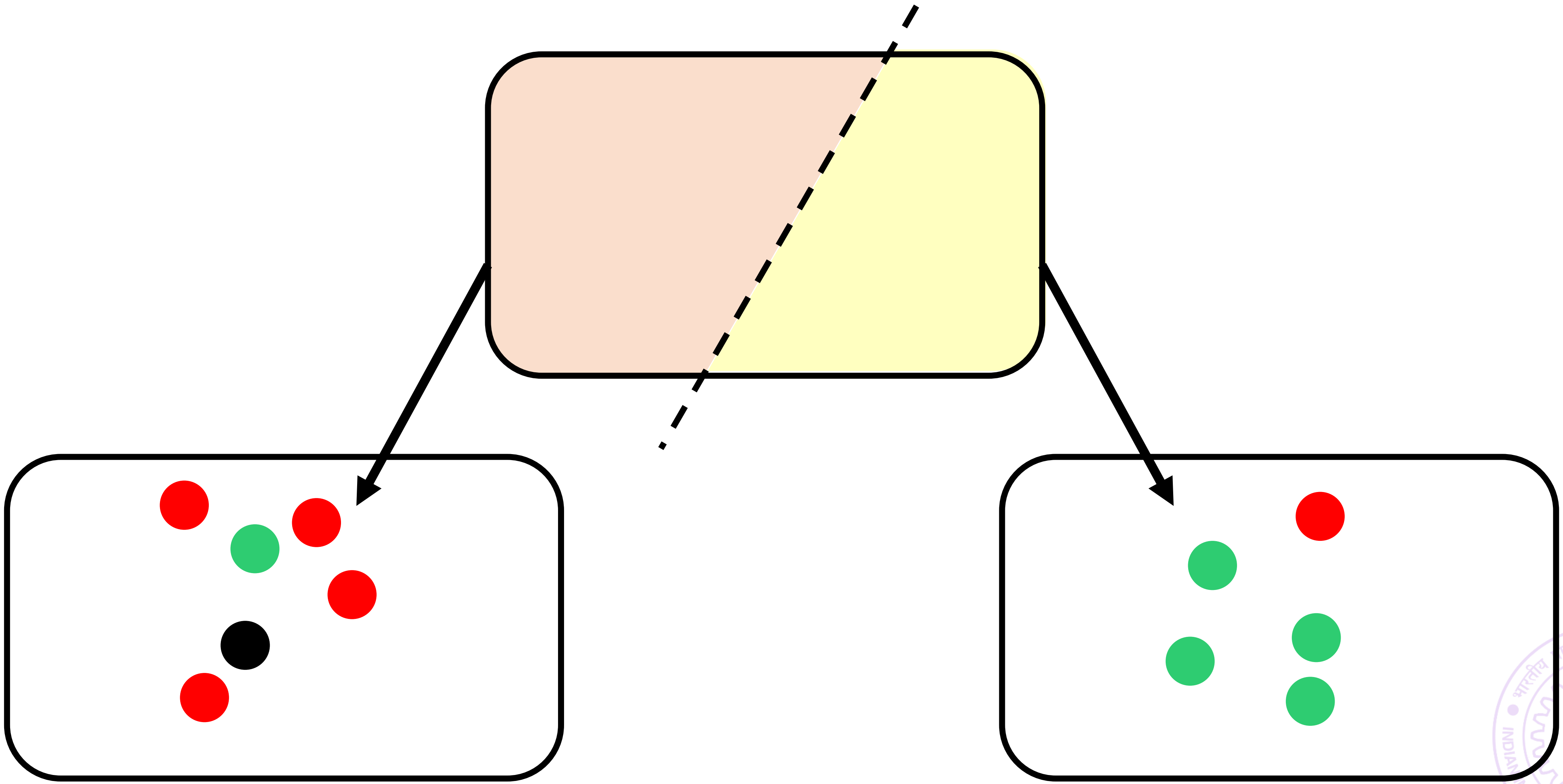
Node splitting via LwP/linear classifiers



Node splitting via LwP/linear classifiers



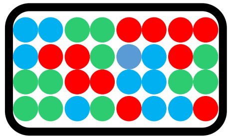
Node splitting via LwP/linear classifiers



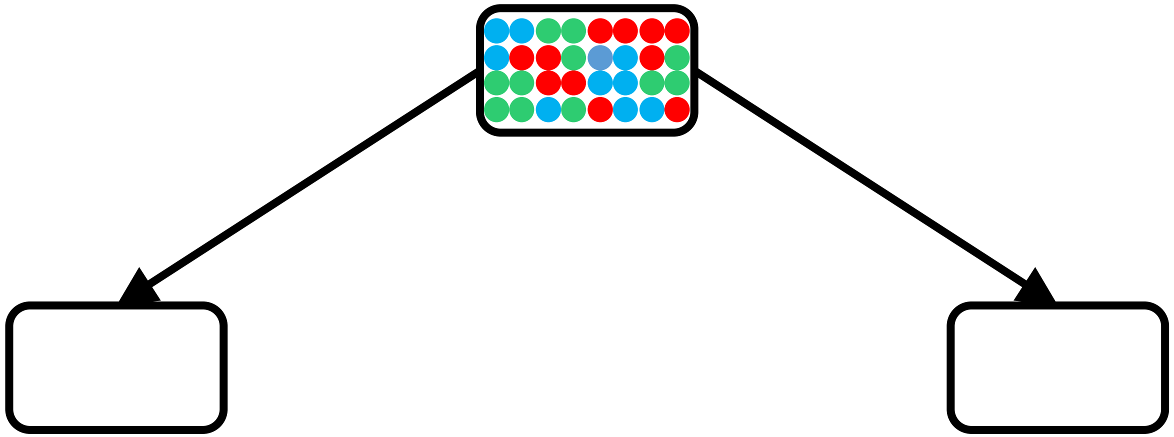
One Final Recap

One Final Recap

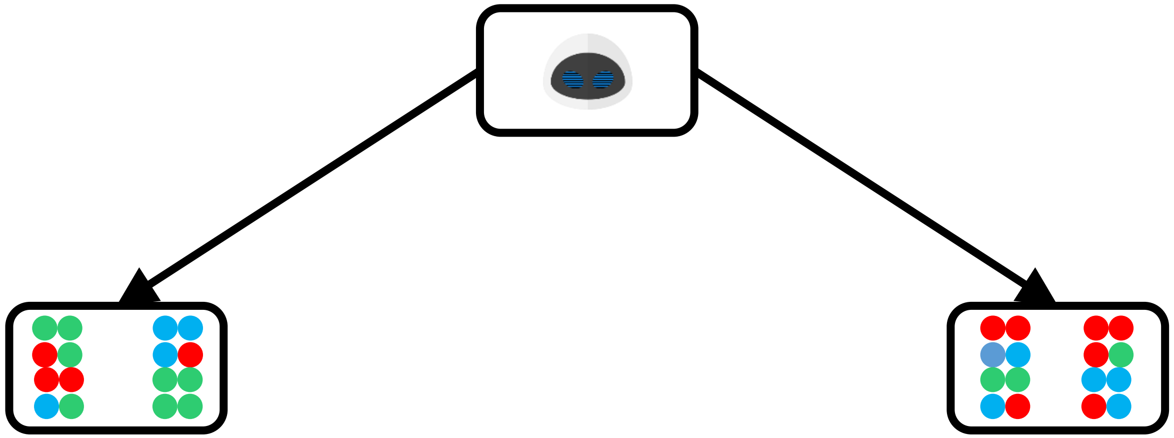
One Final Recap



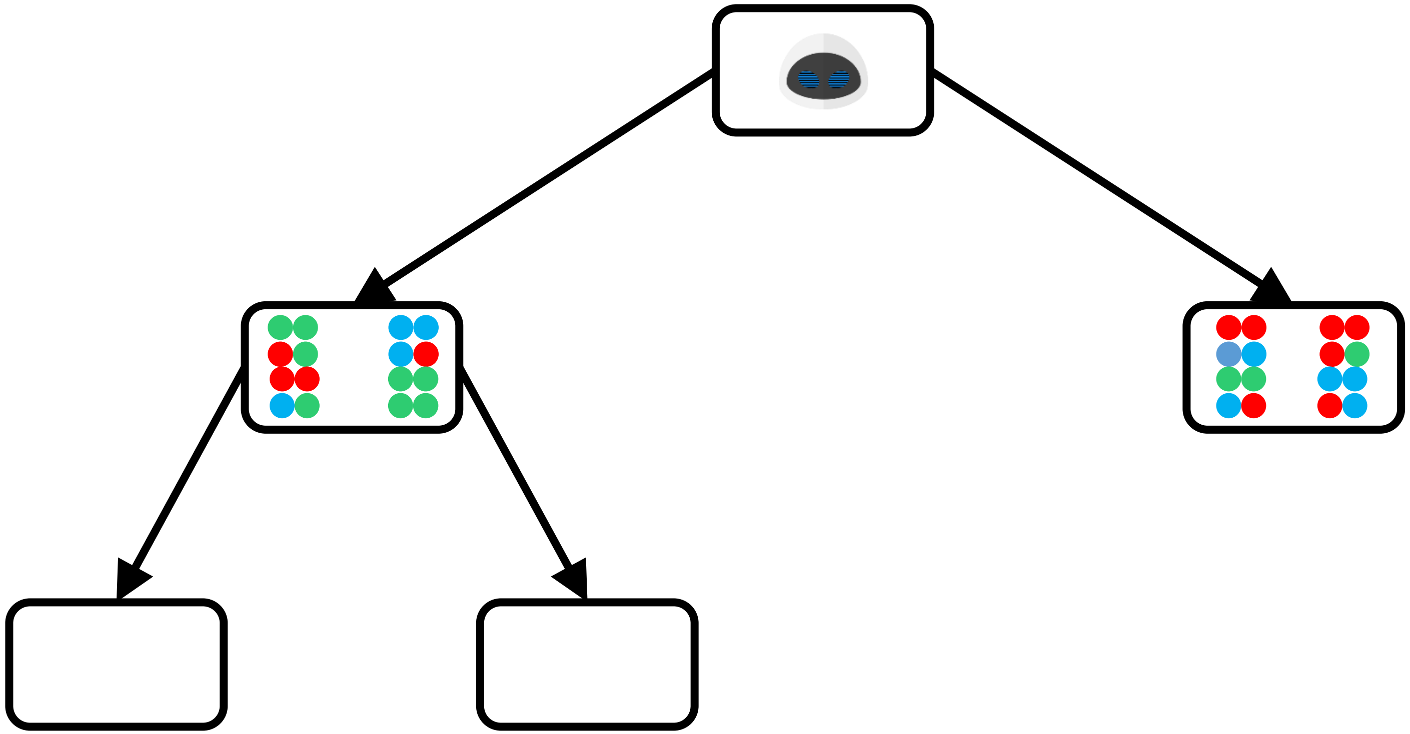
One Final Recap



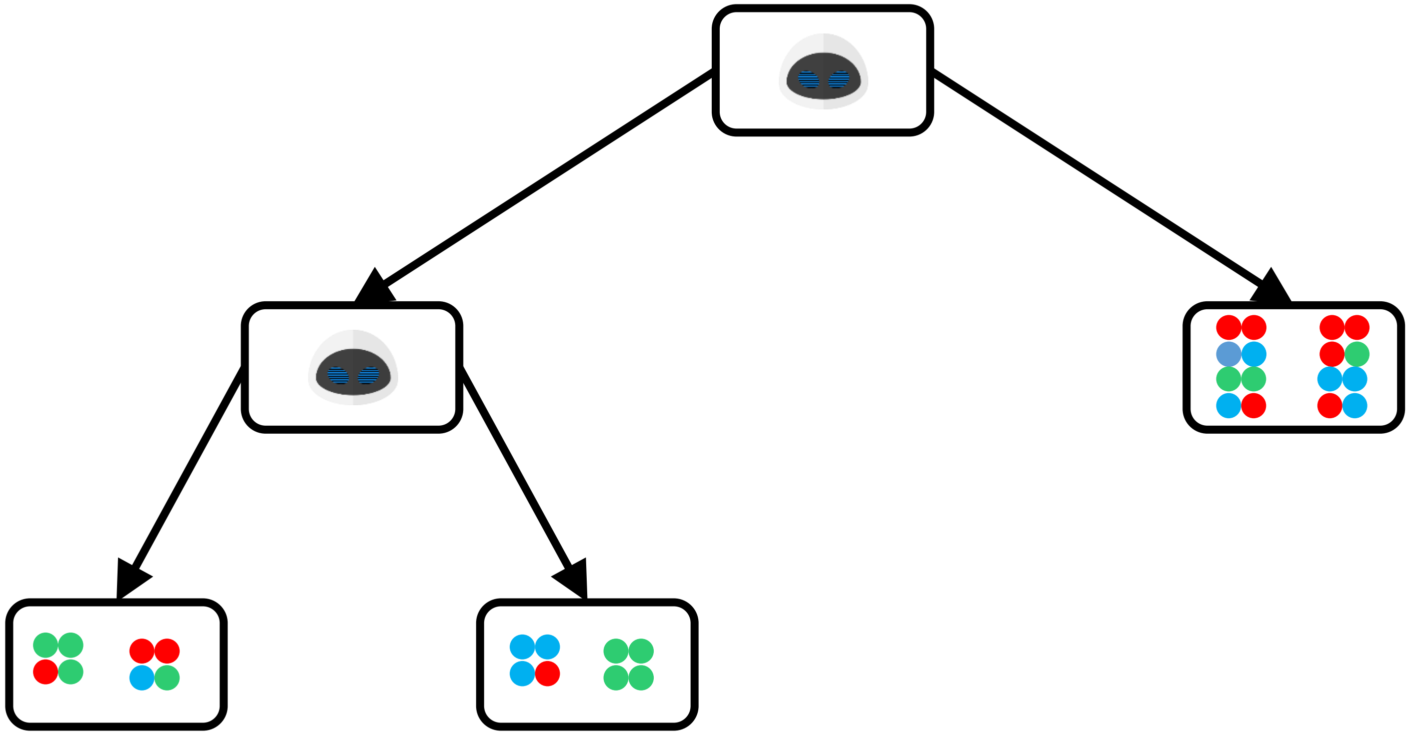
One Final Recap



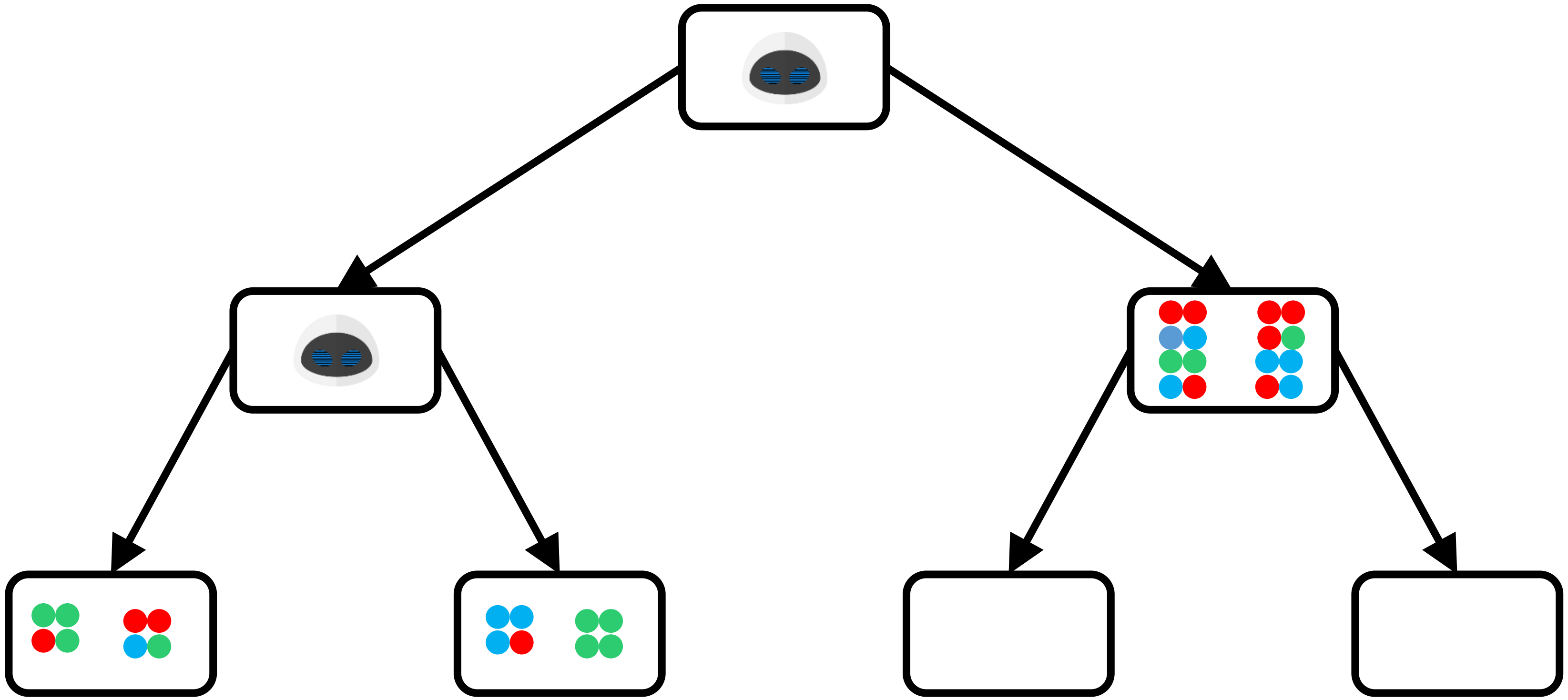
One Final Recap



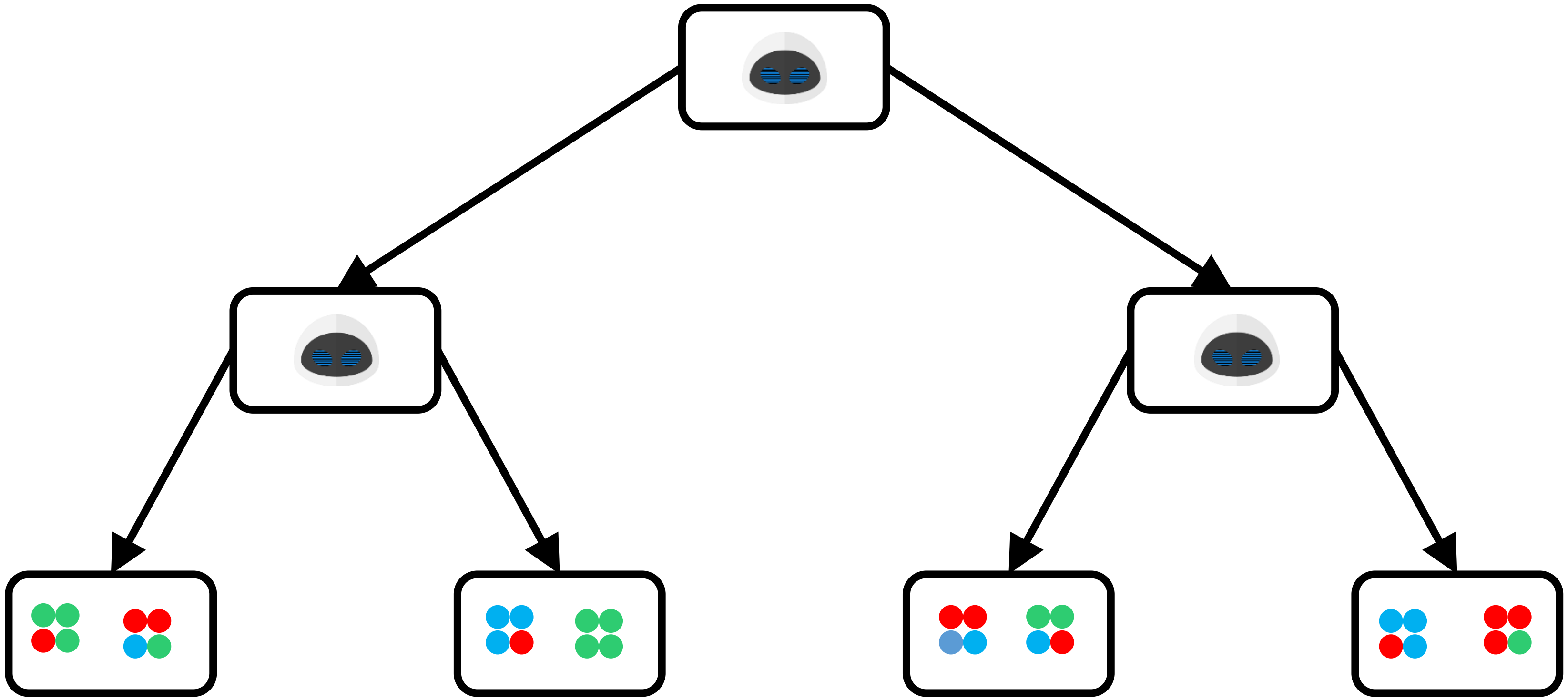
One Final Recap



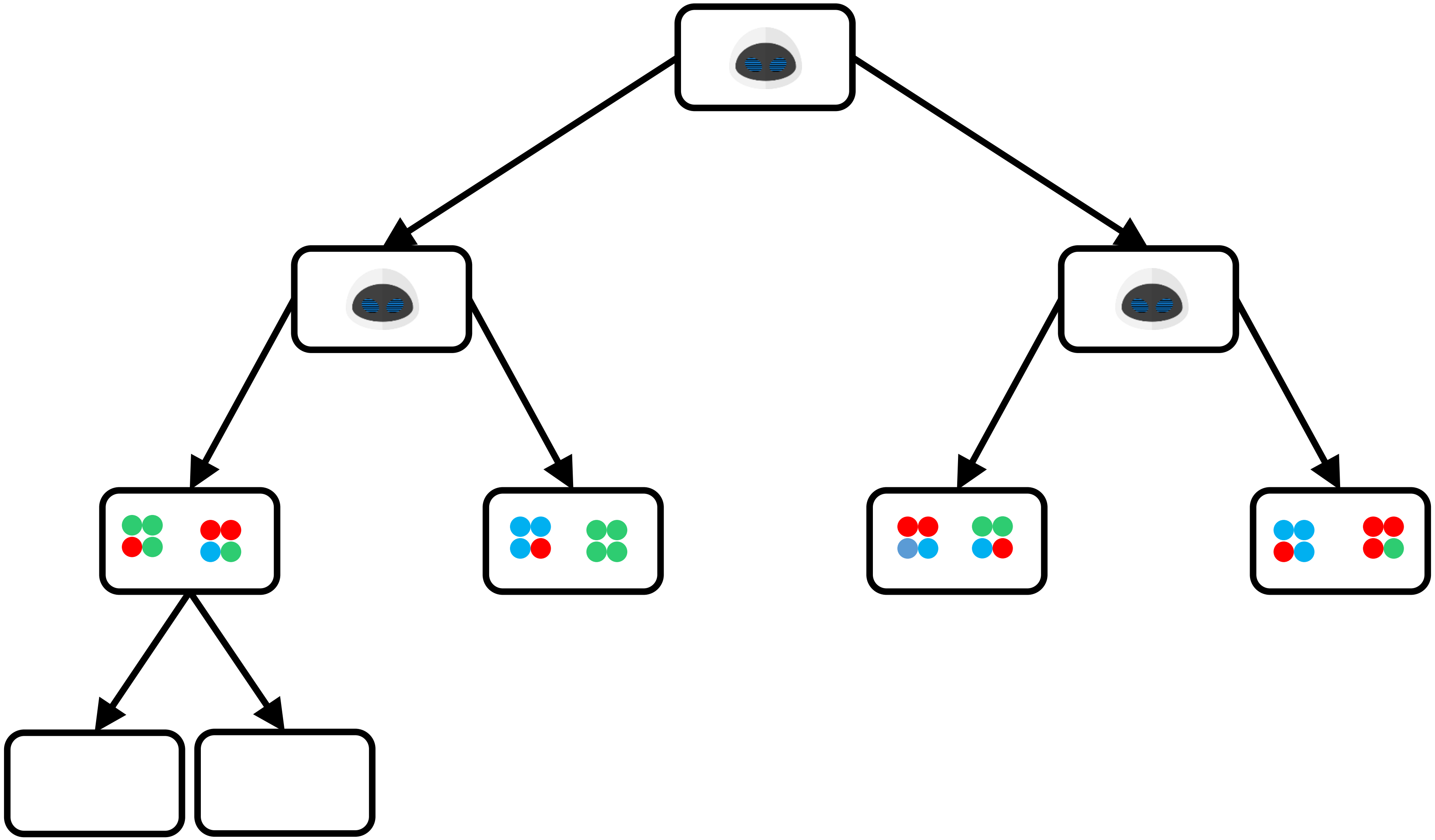
One Final Recap



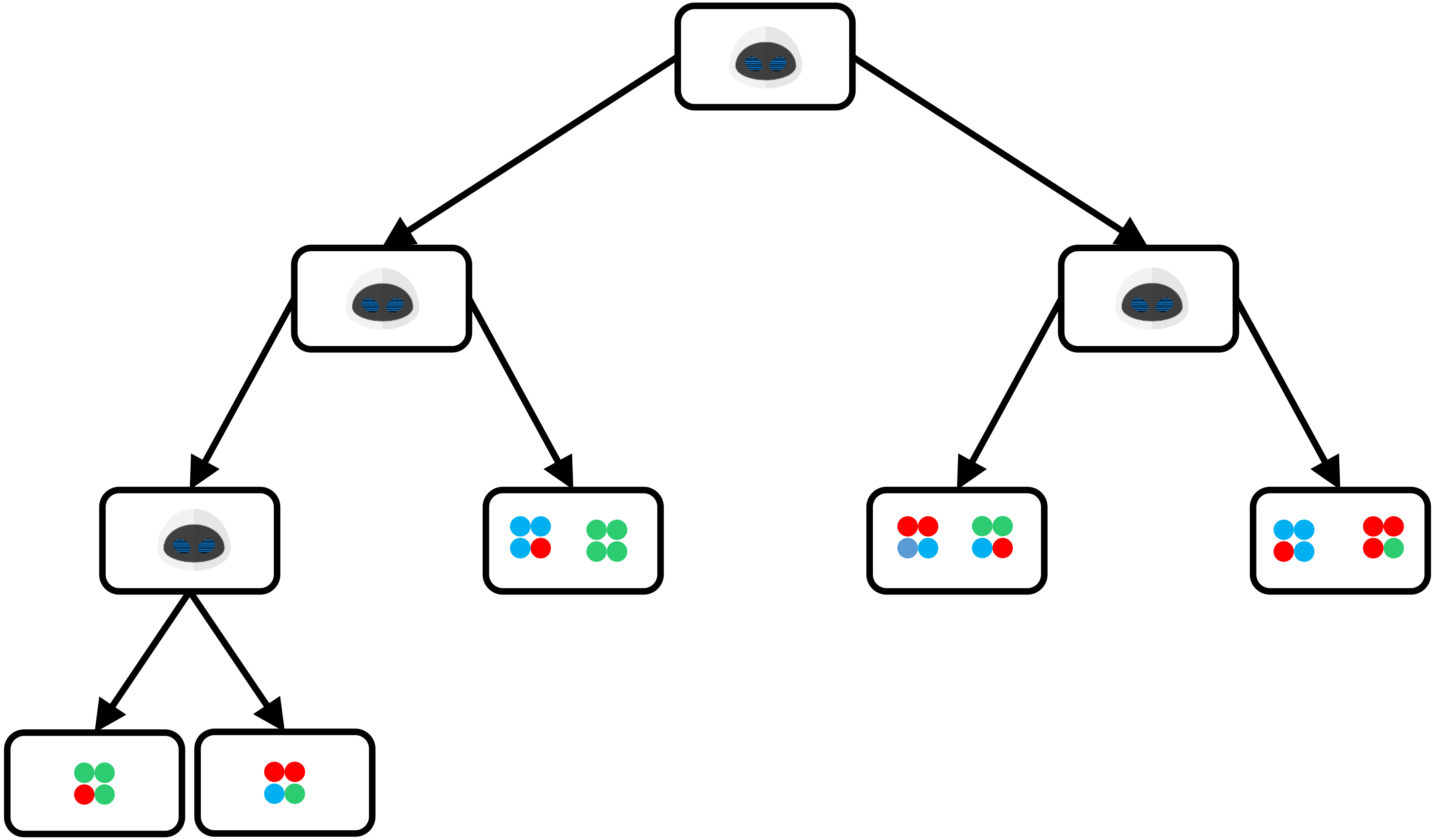
One Final Recap



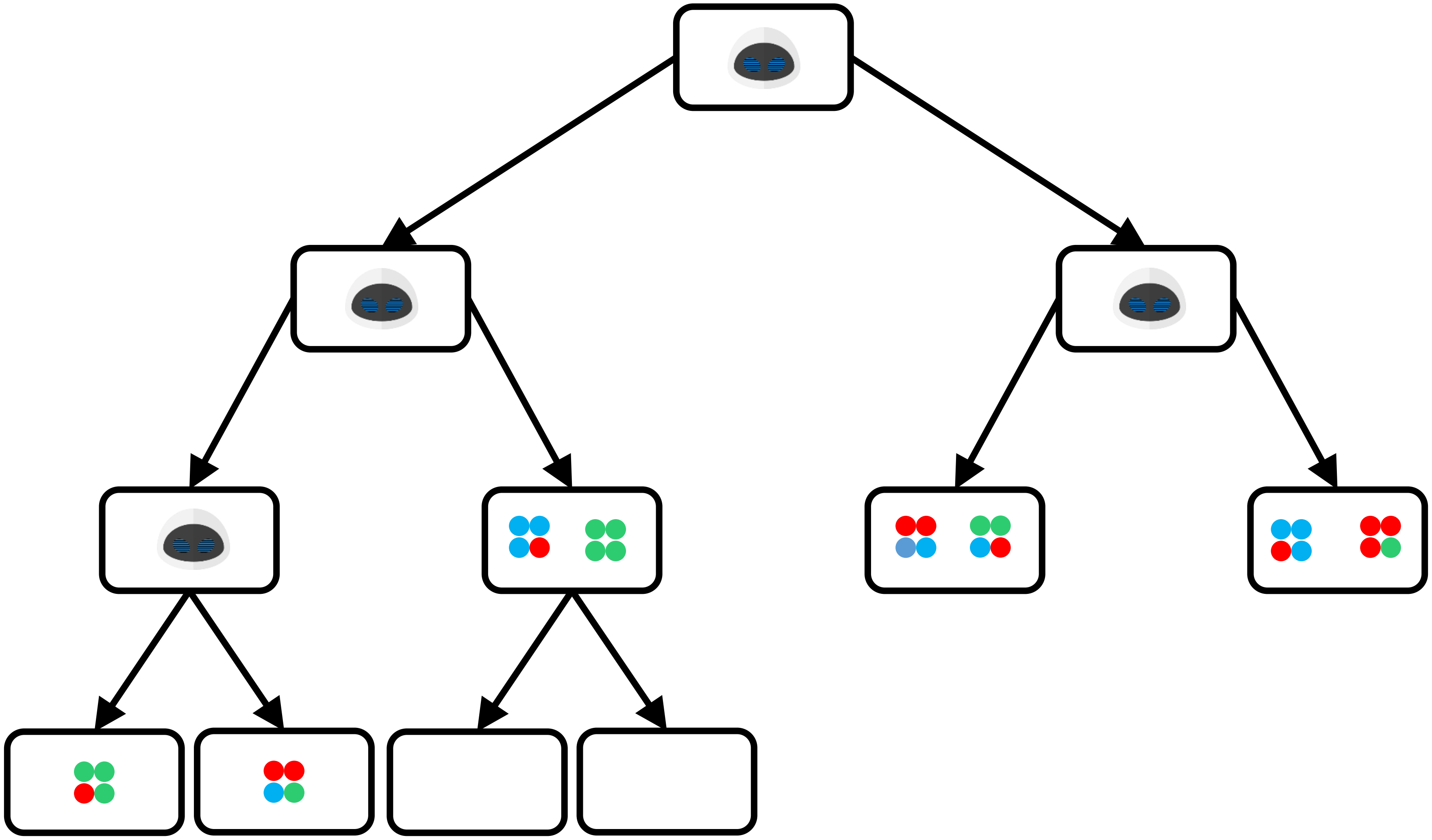
One Final Recap



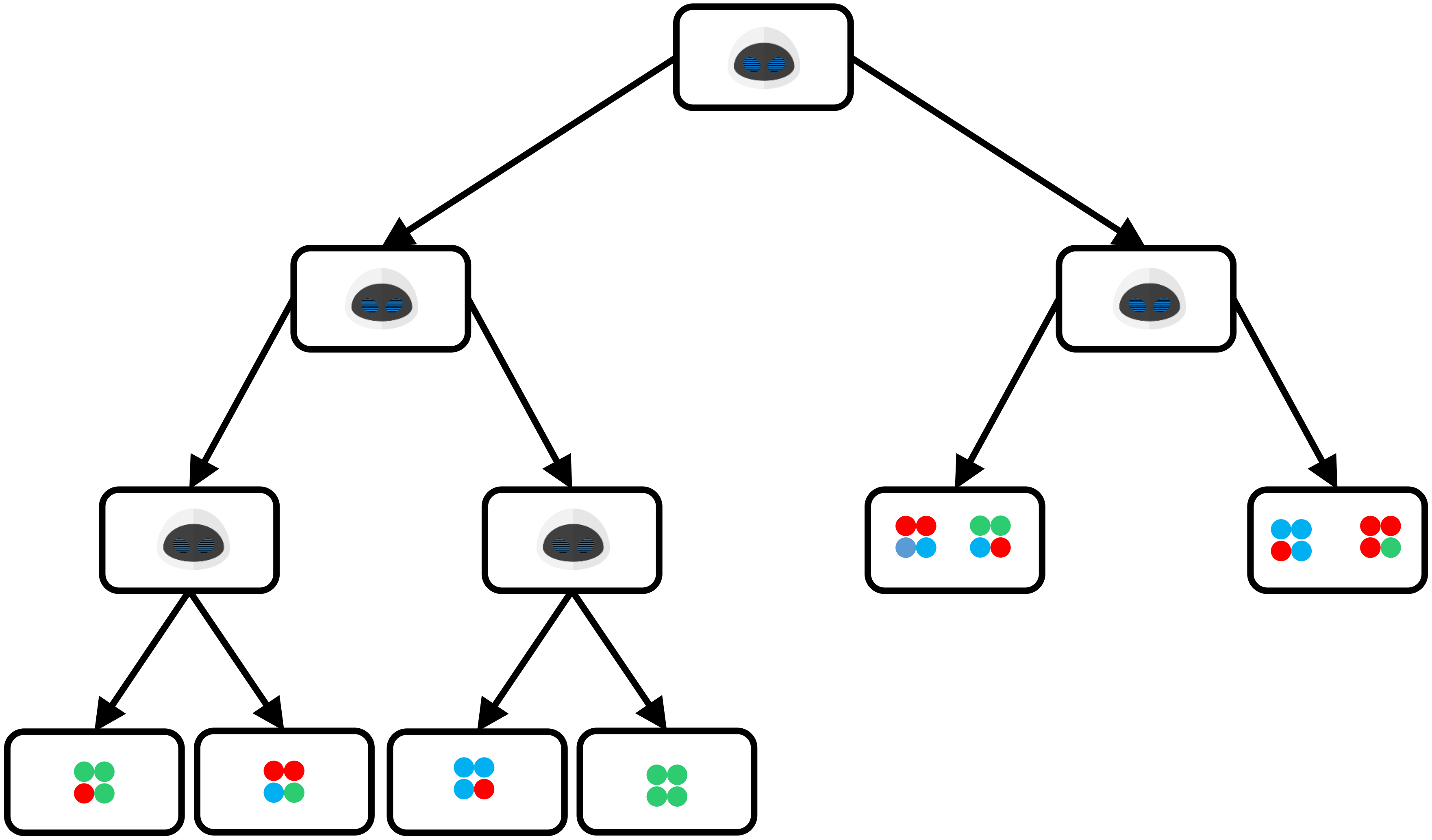
One Final Recap



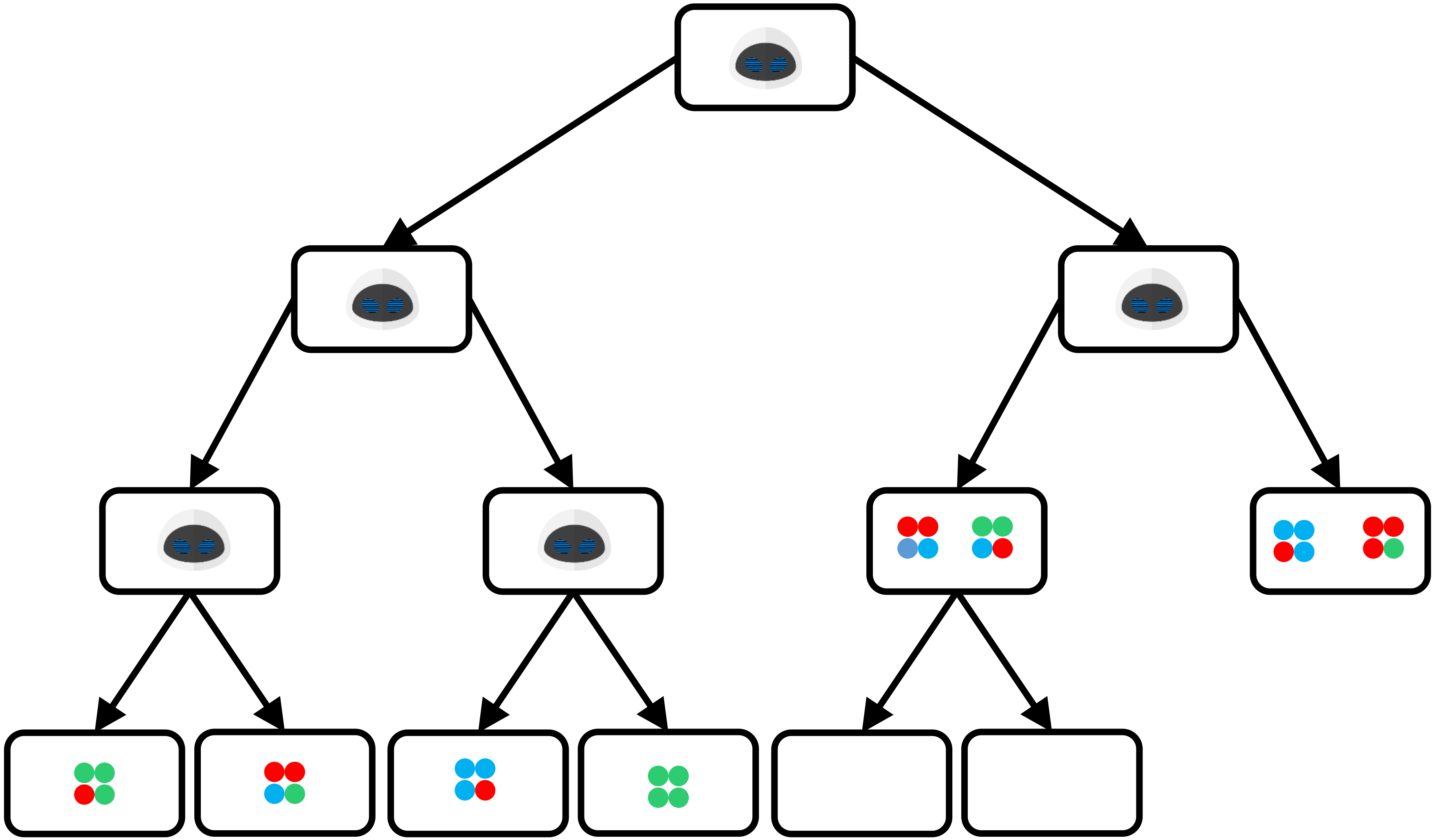
One Final Recap



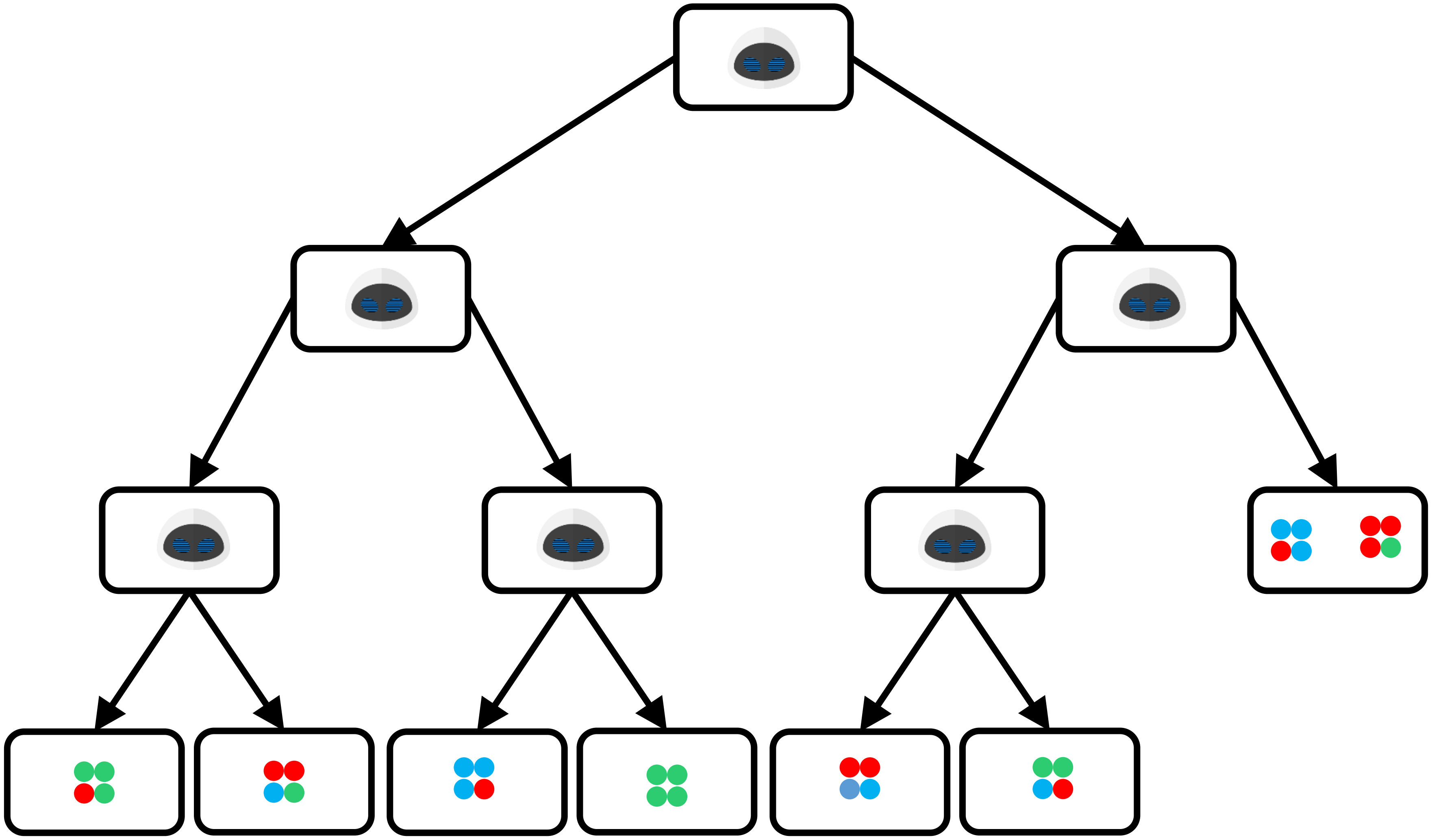
One Final Recap



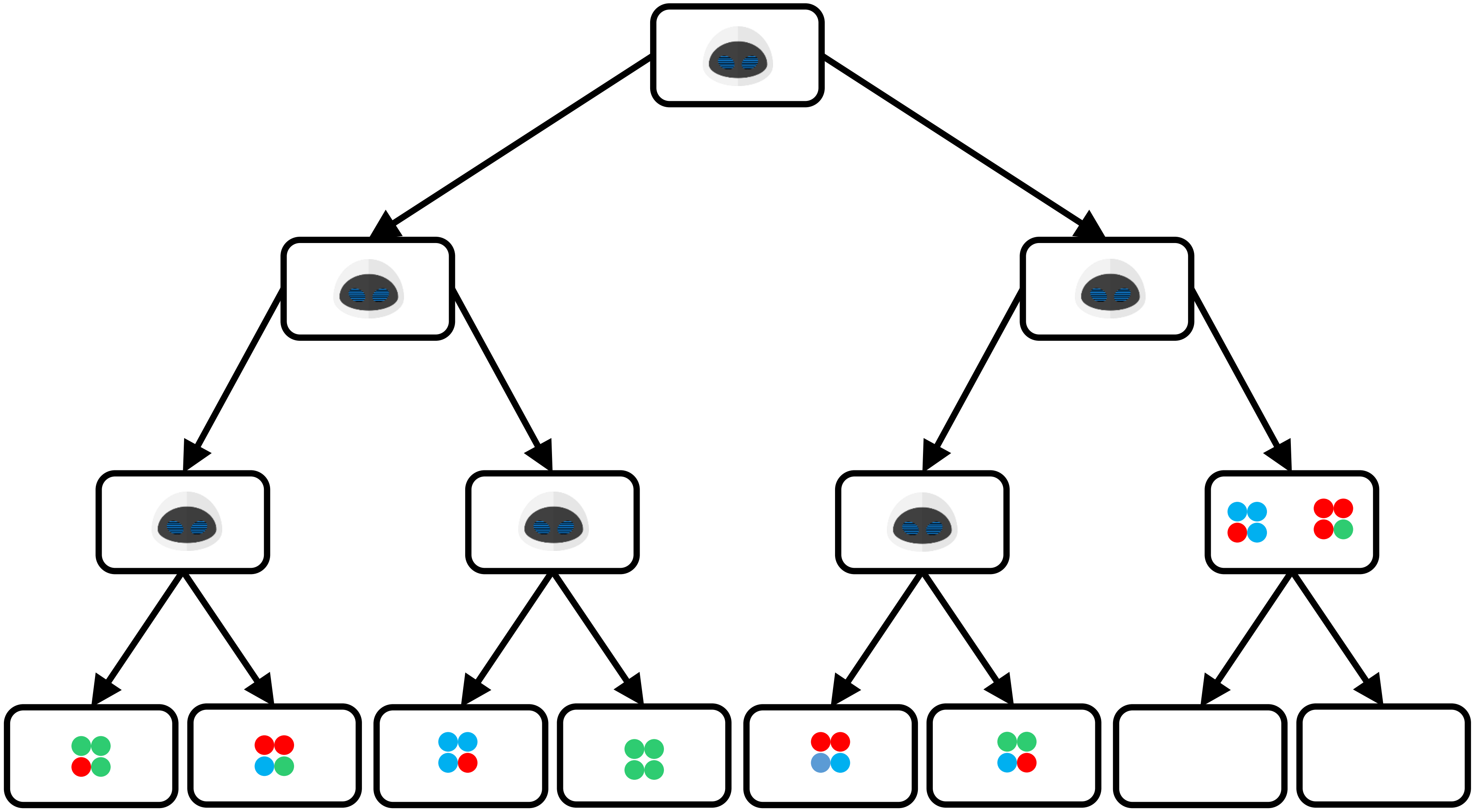
One Final Recap



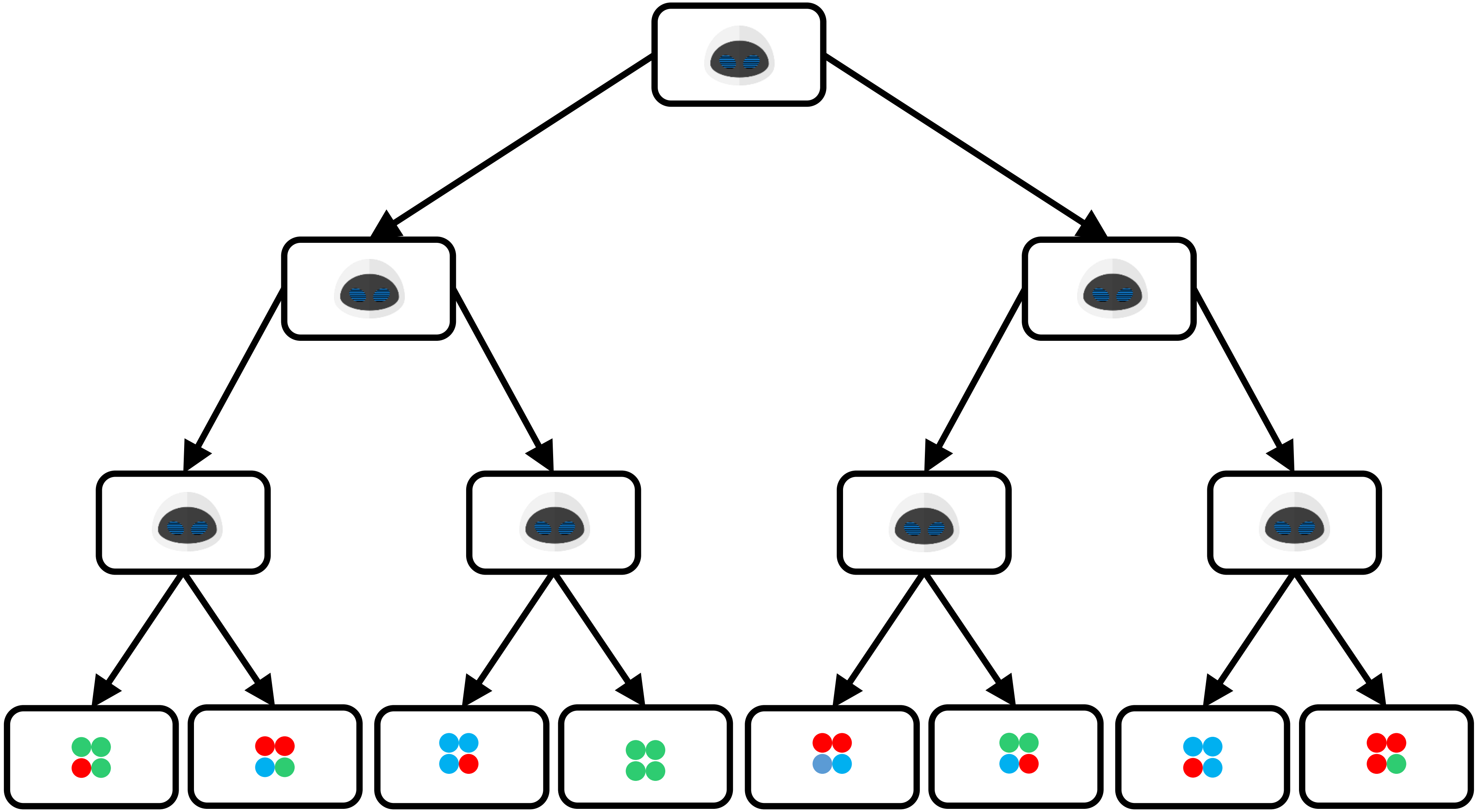
One Final Recap



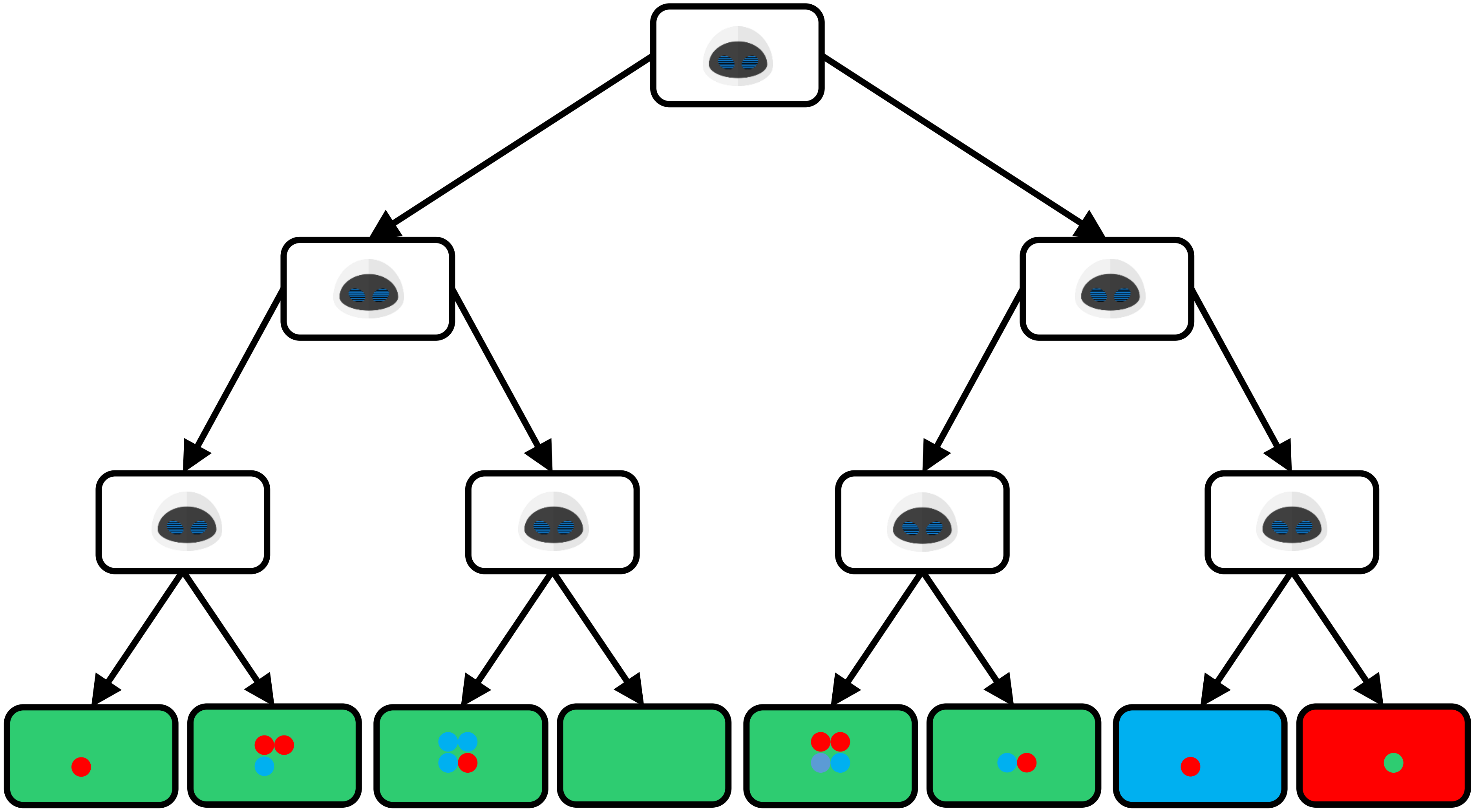
One Final Recap



One Final Recap



One Final Recap



Pruning Strategies

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

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 ☺