Learning using Decision Trees

CS771: Introduction to Machine Learning

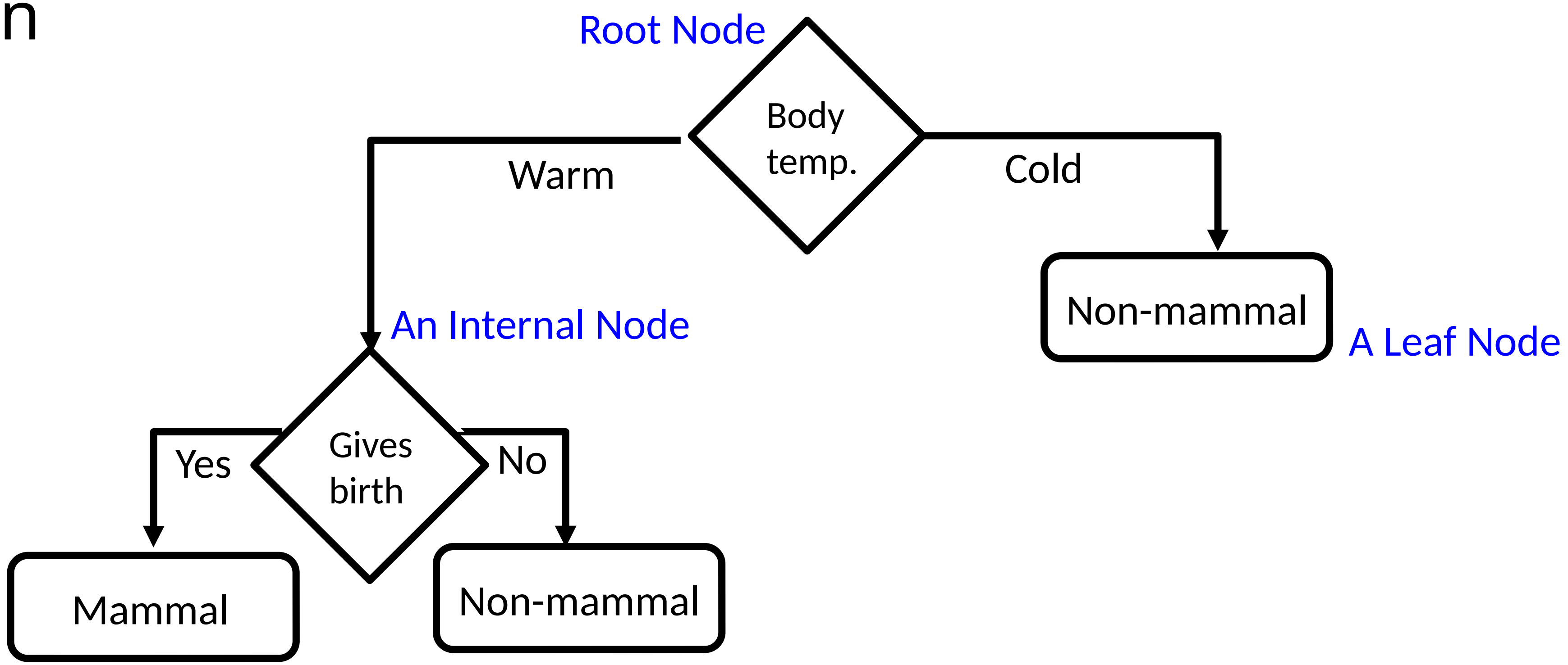
Nisheeth

Announcements

- Today: learning with Decision Trees
- Quiz 1: Next Friday (27th Aug, in-class hours)
  - Syllabus: everything we will have covered by the end of today
- No class this Friday (Muharram holiday)

Decision Trees

- A Decision Tree (DT) defines a hierarchy of rules to make a prediction



- Root and internal nodes test rules. Leaf nodes make predictions
- Decision Tree (DT) learning is about learning such a tree from

A decision tree friendly problem

Loan approval prediction

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

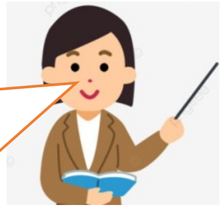
Learning Decision Trees with Supervision

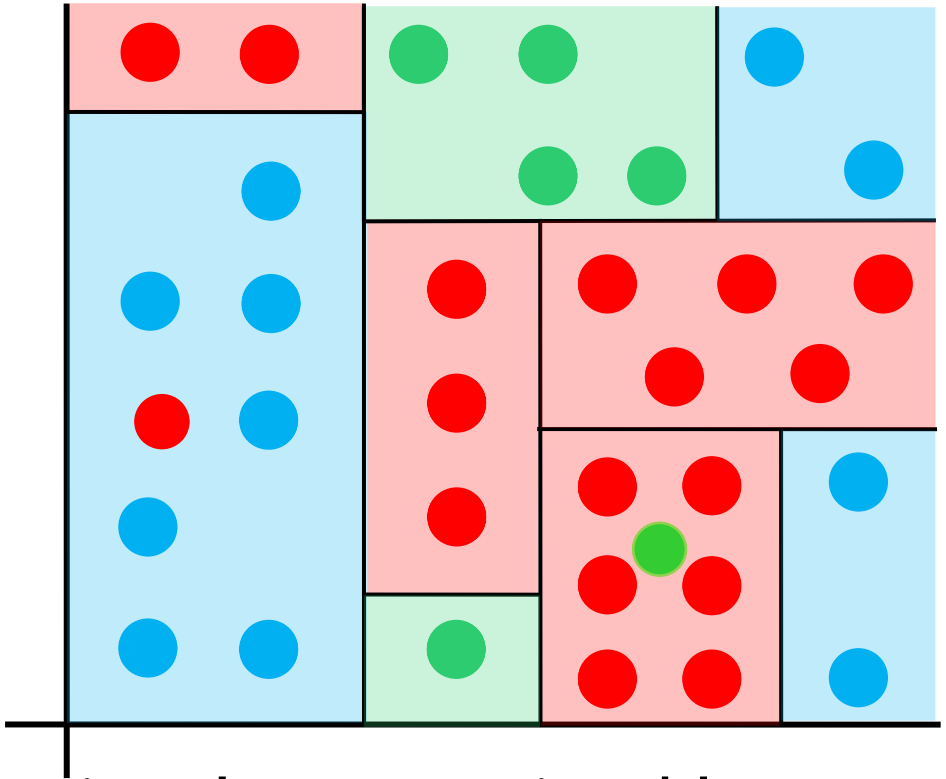
- The basic idea is very simple

- Recursively partition the training data into homogeneous regio

What do you mean by “homogeneous ” regions?

A homogeneous region will have all (or a majority of) training inputs with the same/similar outputs

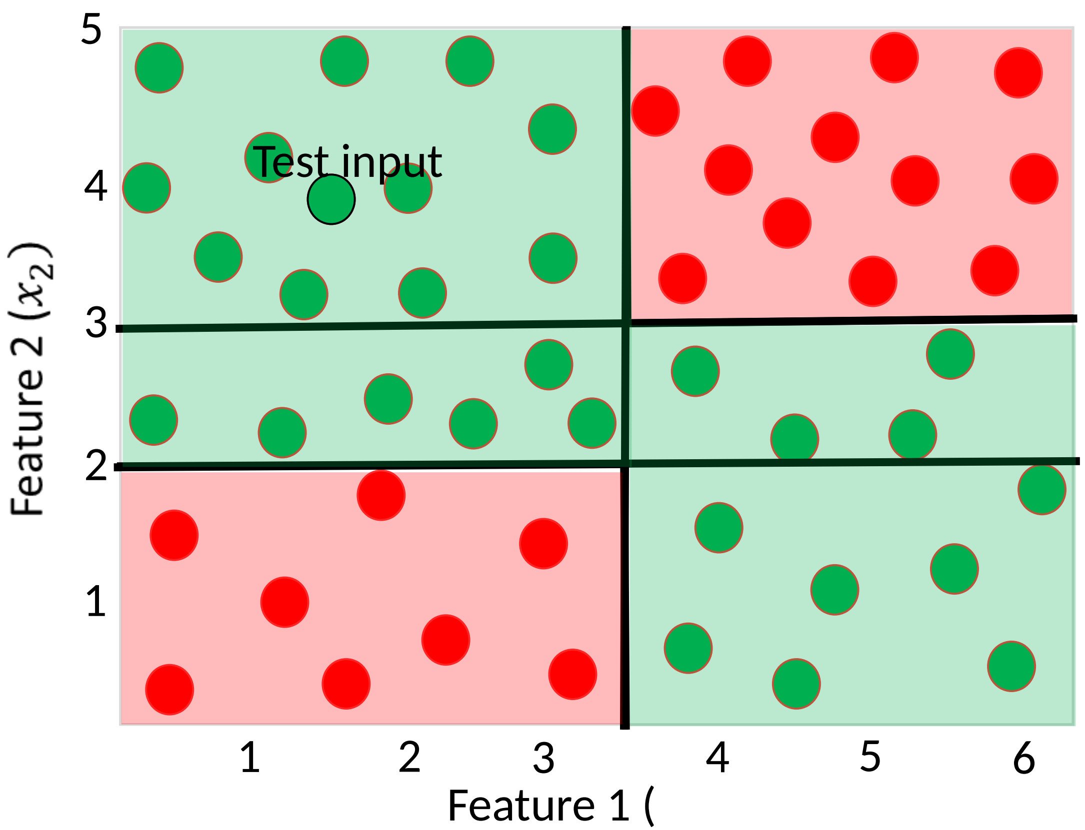


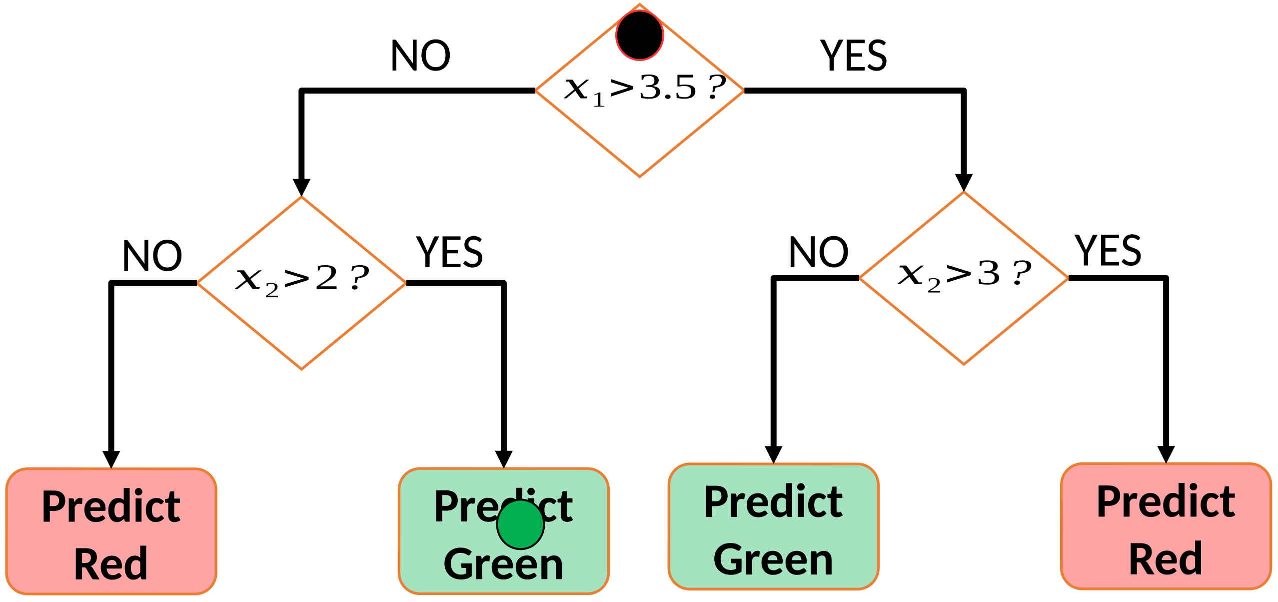


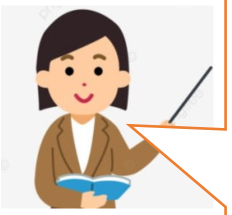
Even though the rule within each group is simple, we are able to learn a fairly sophisticated model overall (note in this example, each rule is a simple horizontal/vertical classifier but the overall decision boundary is rather sophisticated)

- Within each group, fit a simple supervised learner (e.g., predict the majority label)

Decision Trees for Classification







DT is very efficient at test time: To predict the label of a test point, nearest neighbors will require computing distances from 48 training inputs. DT predicts the label by doing just 2 feature-value comparisons! Way faster!

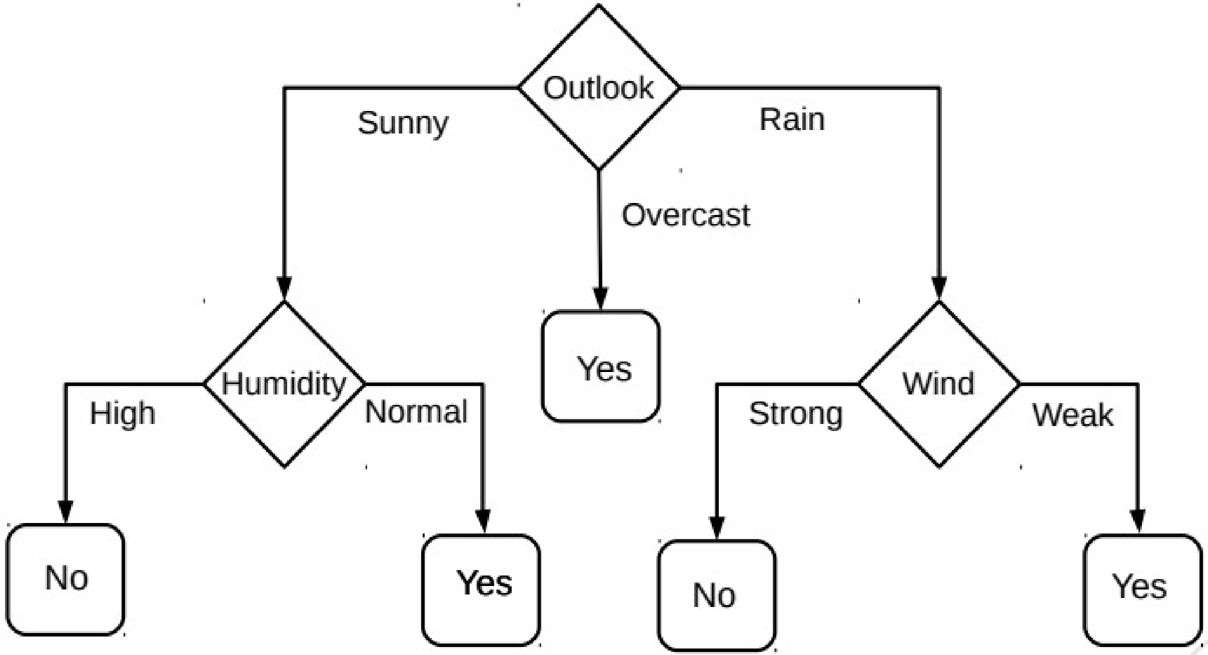
Remember: Root node contains all training inputs

Each leaf node receives a subset of training inputs

Decision Trees for Classification: Another Example

- Deciding whether to play or not to play Tennis on a Saturday
- Each input (Saturday) has 4 categorical features: Outlook, Temp., Humidity, Wind
- A binary classification problem (play vs no-play)
- Bel[illegible] elo[illegible] usi[illegible]

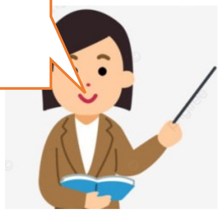
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



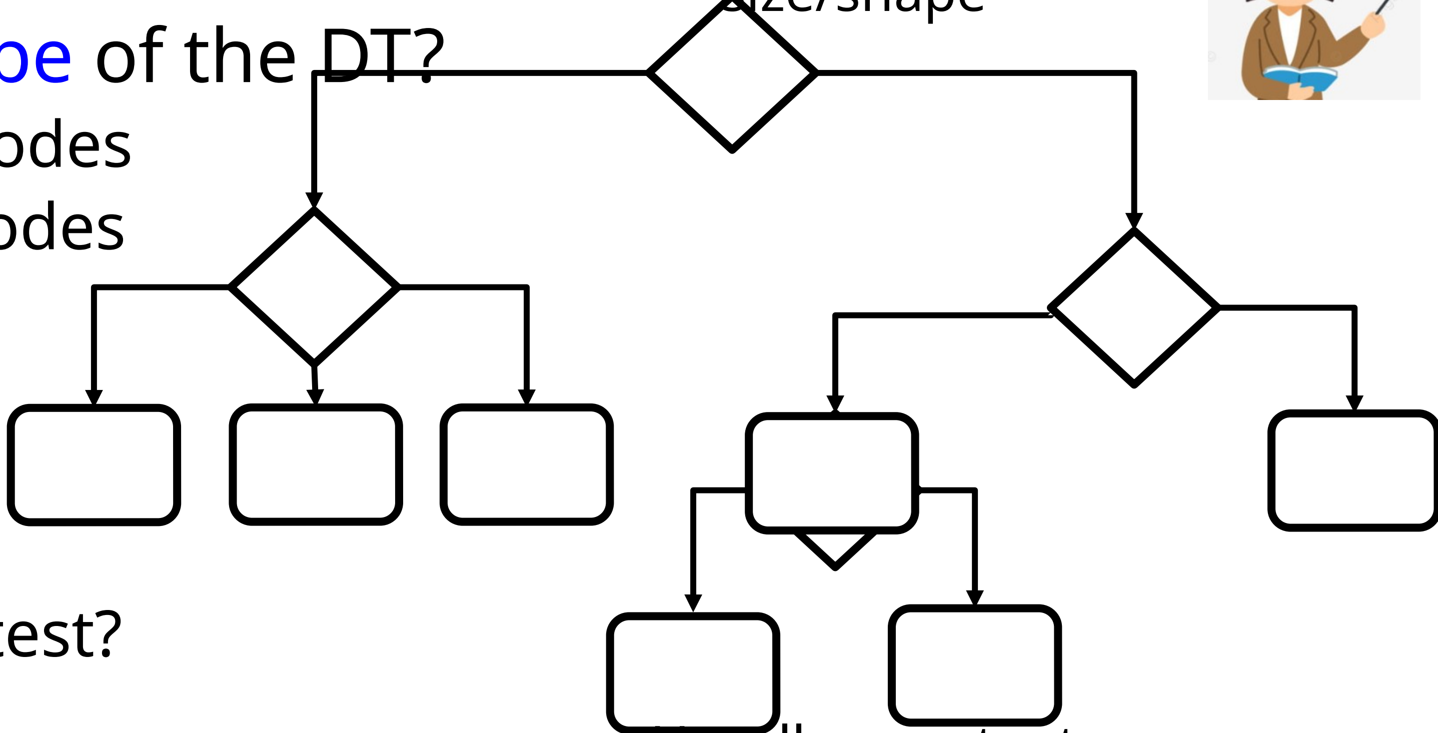
Example credit: Tom Mitchell

Decision Trees: Some Considera

Usually, cross-validation can be used to decide size/shape



- What should be the size/shape of the DT?
  - Number of internal and leaf nodes
  - Branching factor of internal nodes
  - Depth of the tree



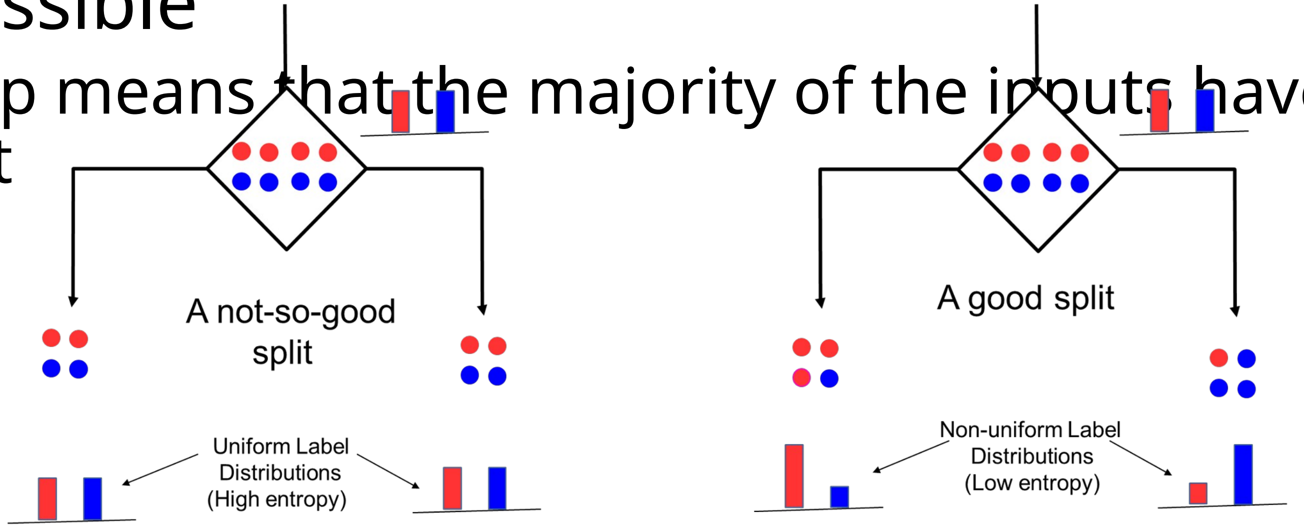
- Split criterion at internal nod
  - Use another classifier?
  - Or maybe by doing a simpler test?

Usually, constant prediction at leaf nodes used since it will be very fast

- What to do at the leaf node? Some options:
  - Make a constant prediction for each test input reac
  - Use a nearest neighbor based prediction using training inputs at that le node
  - Train and predict using some other sophisticated supervised learner on that node

How to Split at Internal Nodes?

- Recall that each internal node receives a subset of all the training inputs
- Regardless of the criterion, the split should result in as “pure” groups as possible
  - A pure group means that the majority of the inputs have the same label/output

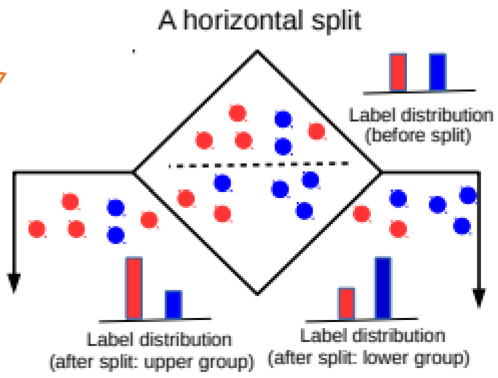


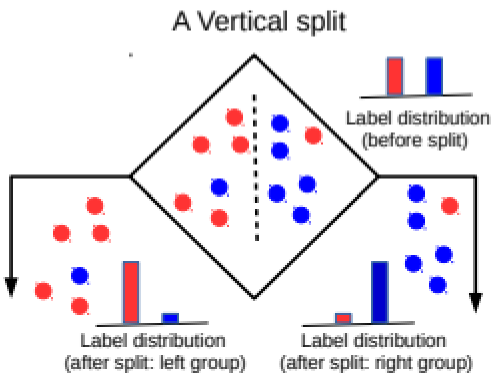
- For classification problems (discrete outputs), entropy is a measure of purity

Techniques to Split at Internal Nodes

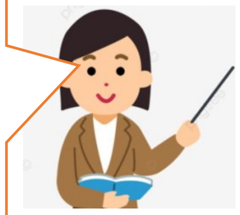
- Each internal node decides which outgoing branch an input should be sent to
- This decision/split can be done using various ways, e.g.,
  - Testing the value of a single feature at a time (such internal node called

With this approach, all features and all possible values of each feature need to be evaluated in selecting the feature to be tested at each internal node
(can be slow but can be made faster using some tricks)

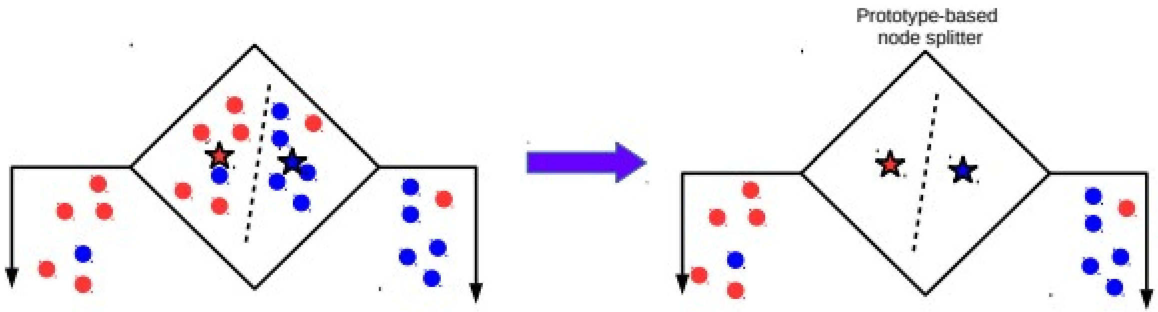




DT methods based on testing a single feature at each internal node are faster and more popular (e.g., ID3, C4.5 algos)



- Learr



so

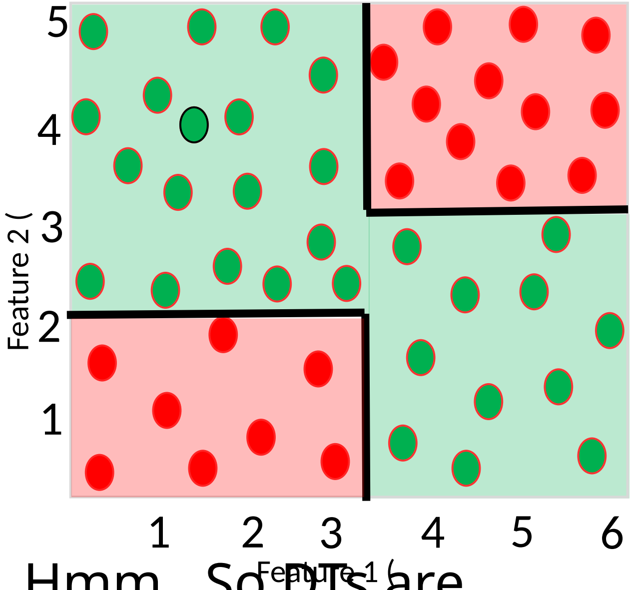
DT methods based on learning and using a separate classifier at each internal node are less common. But this approach can be very powerful and are sometimes used in some advanced DT methods

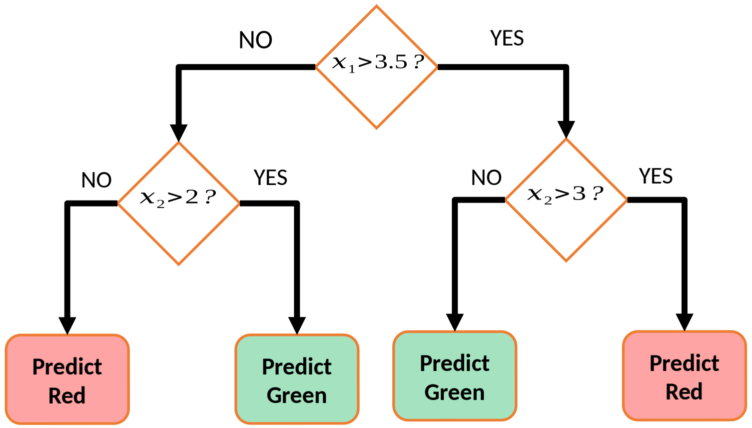
Constructing Decision Trees

Given some training data, what's the "optimal" DT?

How to decide which rules to test for and in what order?

How to assess informativeness of a rule?





Hmm.. So DTs are like the "20 questions" game (ask the most useful questions first)

The rules are organized in the DT such that most informative rules are tested first

Informativeness of a rule is of related to the extent of the purity of the split arising due to that rule. More informative rules yield more pure splits

In general, constructing DT is an intractable problem (NP-hard)

Often we can use some "greedy" heuristics to construct a "good" DT

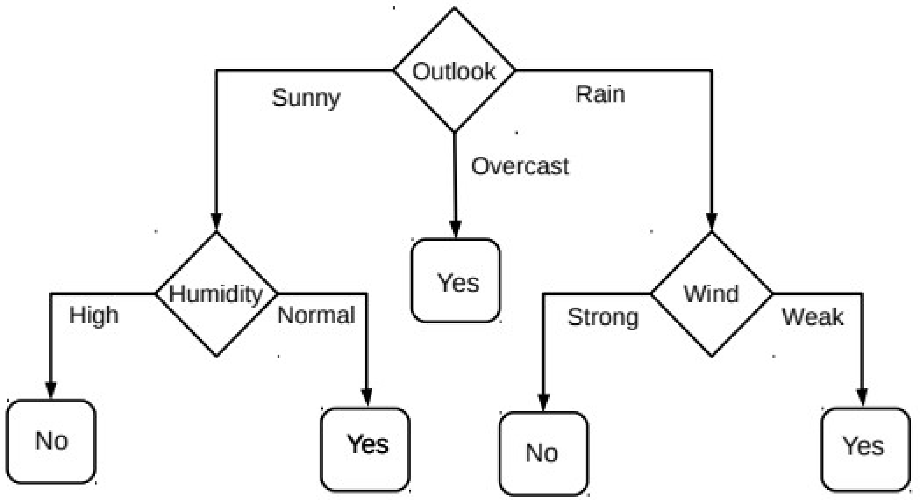
To do so, we use the training data to figure out which rules should be tested at each node

The same rules will be applied on the test inputs to route them along the tree until they reach some leaf node where the prediction is made

Decision Tree Construction: An Example

- Let's consider the playing Tennis example
- Assume each internal node will test the value of one of the features

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



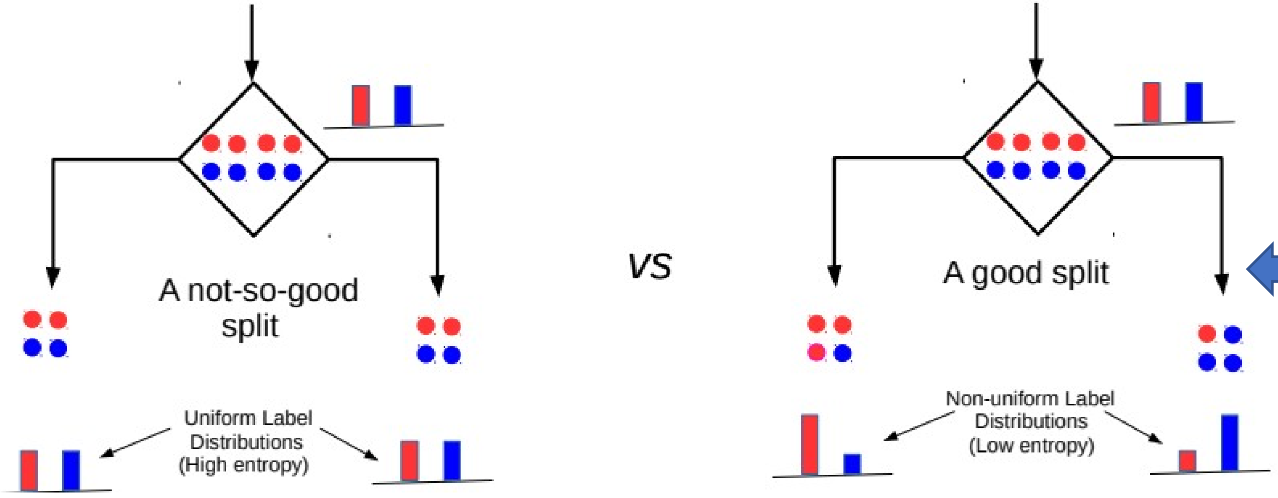
- Question: Why does it make more sense to test the feature "outlook" first?
- Answer: Of all the 4 features, it's the most informative
  - It has the highest information gain as the root node

Entropy and Information Gain

- Assume a set of labelled inputs **from *C*** classes, as fraction of class c inputs
- **Entropy** of the set is defined as
- Suppose a rule splits into two smaller disjoint sets and
- Reduction in entropy after the split is called information gain

Uniform sets (all classes roughly equally present) have high entropy; skewed sets low

This split has a low IG (in fact zero IG)



This split has higher IG

Entropy and Information Gain

- Let's use IG based criterion to construct a DT for the Tennis example
- At root node, let's compute IG of each of the 4 features
- Consider feature "wind". Root contains <u>all</u> examples $\boldsymbol{S}$ =

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

$$H(S) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.94$$

$$\boldsymbol{S}_{\text{weak}} = [6+, 2-] \Rightarrow H(\boldsymbol{S}_{\text{weak}}) = 0.811$$

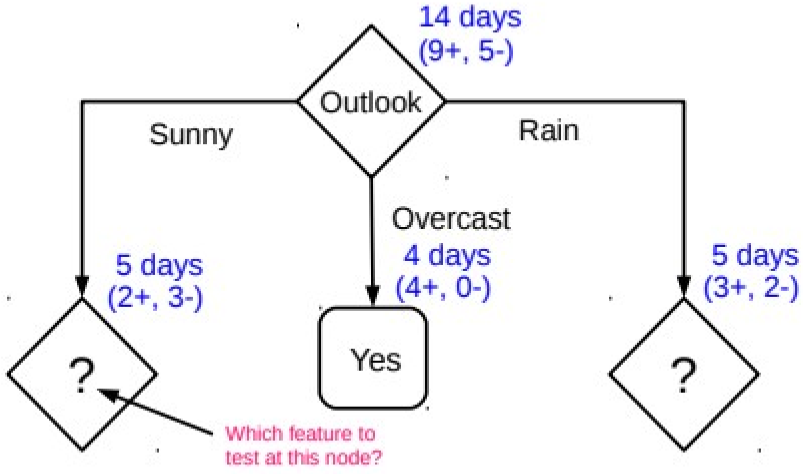
$$\boldsymbol{S}_{\text{strong}} = [3+, 3-] \Rightarrow H(\boldsymbol{S}_{\text{strong}}) = 1$$

$$= 0.94 - 8/14 * 0.811 - 6/14 * 1 = 0.048$$

- Likewise, at root: IG(S, outlook) = 0.246, IG(S, humidity) = 0.151, IG(S,temp) = 0.029
- Thus we choose "outlook" feature to be tested at the root node
- Now how to grow the DT, i.e., what to do at the next level? Which feature to test next?

Growing the tree

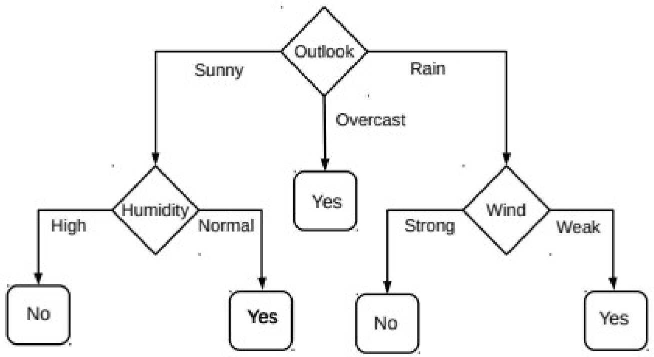
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



- Proceeding as before, for level 2, left node, we can verify that
  - IG(S,temp) = 0.570, IG(S, humidity) = 0.970, IG(S, wind) = 0.019
- Thus humidity chosen as the feature to be tested at level 2, left node
- No need to expand the middle node (already “pure” - all “yes” training examples)
- Can also verify that wind has the largest IG for the right node
- Note: If a feature has already been tested along a path earlier, we don’t

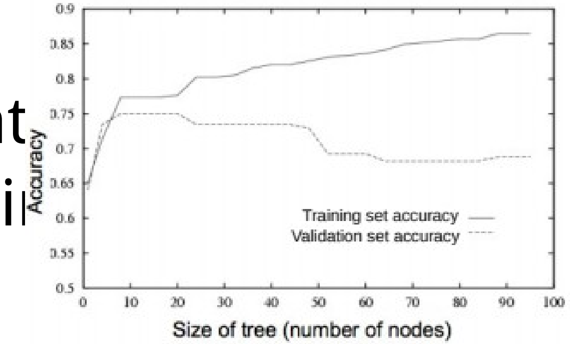
When to stop growing the tree?

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

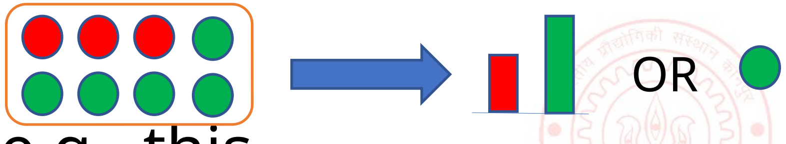


- Stop expanding a node further (i.e., make it a leaf node) when
  - It consist of all training examples having the same label (the node becomes "pure")
  - We run out of features to test along the path to that
  - The DT starts to overfit (can b ... itori ... the validation set accuracy)

To help prevent the tree from growing too much!



- **Important:** No need to obsess too much for purity
  - It is okay to have a leaf node that is not fully pure, e.g., this
  - At test inputs that reach an impure leaf, can predict probability of belonging to each class (in above example, p(red) = 3/8, p(green) = 5/8), or simply



Avoiding Overfitting in DTs

- Desired: a DT that is not too big in size, yet fits the training data reasonably
- Note: An example of a very simple DT is "decision-stump"
  - A decision-stump only tests the value of a single feature (or a simple rule)
  - Not very powerful in itself but often used in large ensembles of decision stumps
- Mainly two approaches to prune a complex DT
  - Prune while building the tree (stopping early)
  - Prune after building the tree (post-pruning)

Either can be done using a validation set

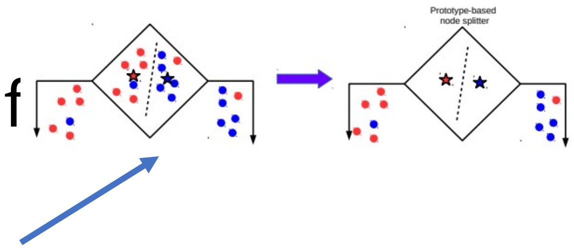
- Criteria for judging which nodes could potentially be pruned
  - Use a validation set (separate from the training set)
    - Prune each possible node that doesn't hurt the accuracy on the validation set
    - Greedily remove the node that improves the validation accuracy the most
    - Stop when the validation set accuracy starts worsening

Decision Trees: Some Comments

- Gini-index defined as can be an alternative to IG

- For DT regression[1], variance in the outputs can be used to assess purity

- When features are real-valued (no finite possible values to try), things are a bit more tricky
  - Can use tests based on thresholding feature values (recall our synthetic data examples)
  - Need to be careful w.r.t. number of threshold points, how f etc.

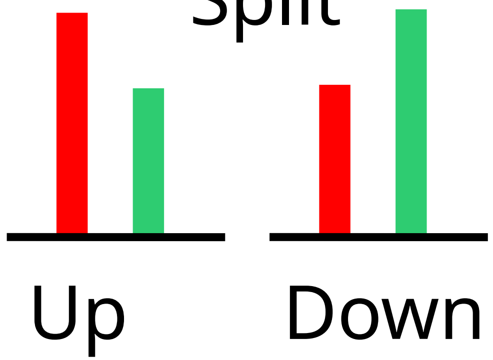


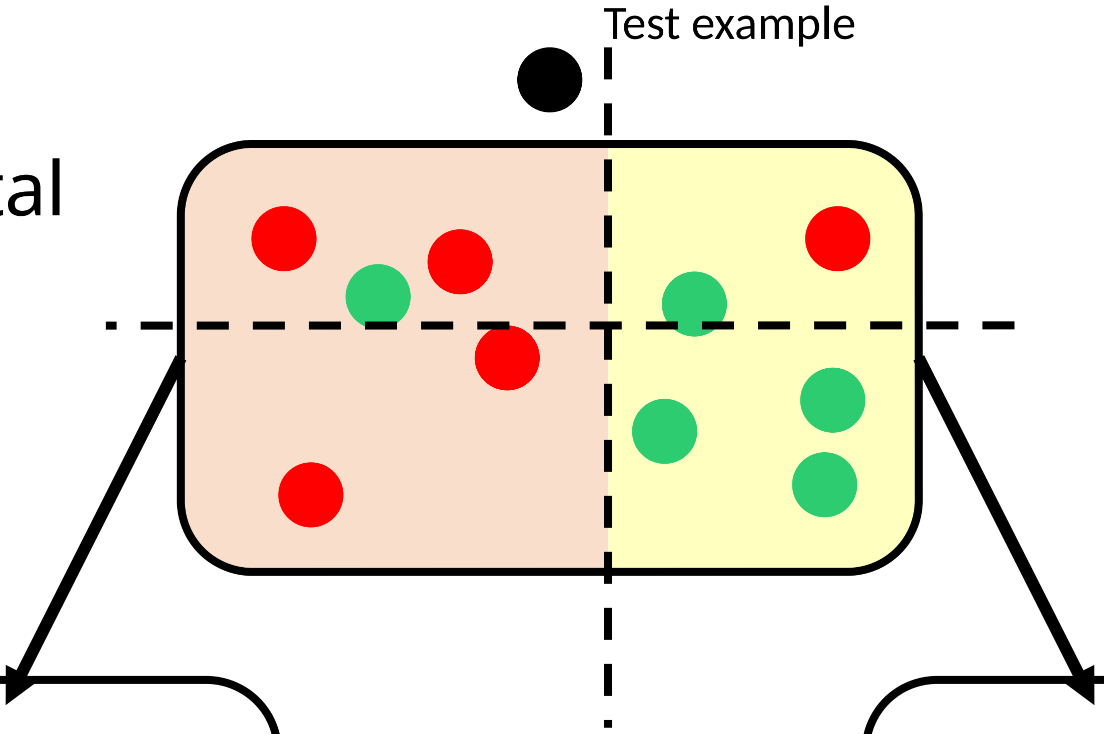
- More sophisticated decision rules at the internal nodes can also be used
  - Basically, need some rule that splits inputs at an internal node into homogeneous groups
  - The rule can even be a machine learning classification algo (e.g., LwP or a deep learner)

[1] man, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees

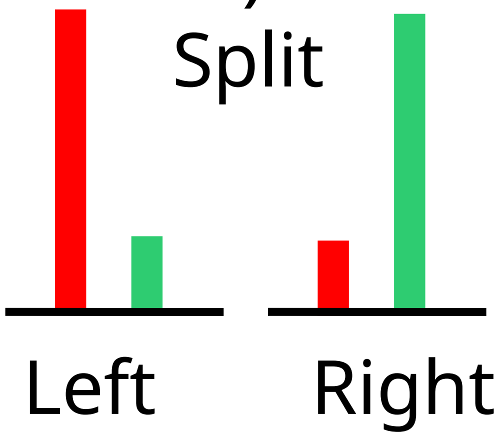
An Illustration: DT with Real-Valued Features

“Best” (purest possible) Horizontal Split

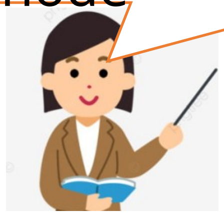




“Best”(purest possible) Vertical Split



Between the best horizontal vs best vertical split, the vertical split is better (purer), hence we use this rule for the internal node

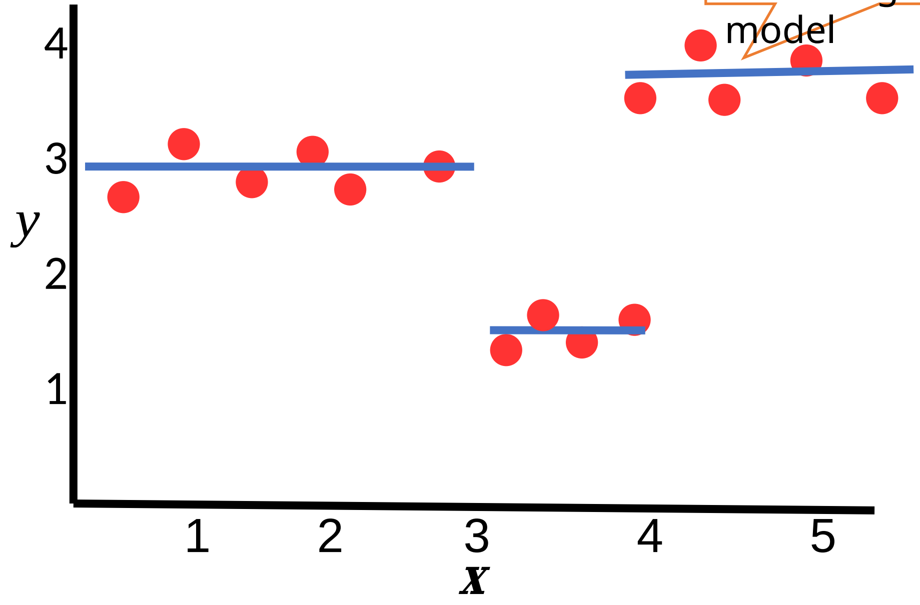


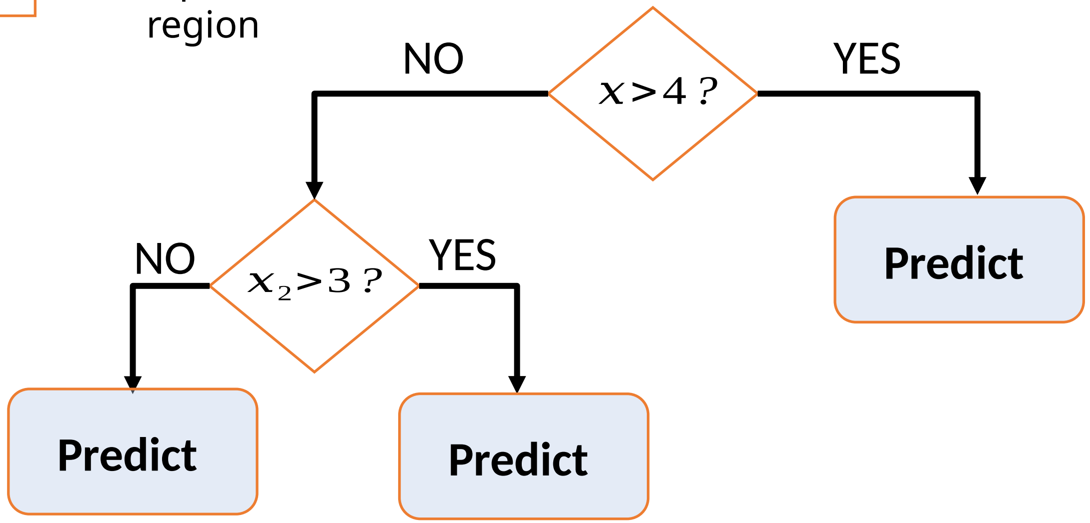
This illustration’s credit: Purushottam Kar

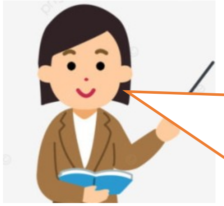
Decision Trees for Regression

Can use any regression model but would like a simple one, so let’s use a constant prediction based regression model

Another simple option can be to predict the average output of the training inputs in this region







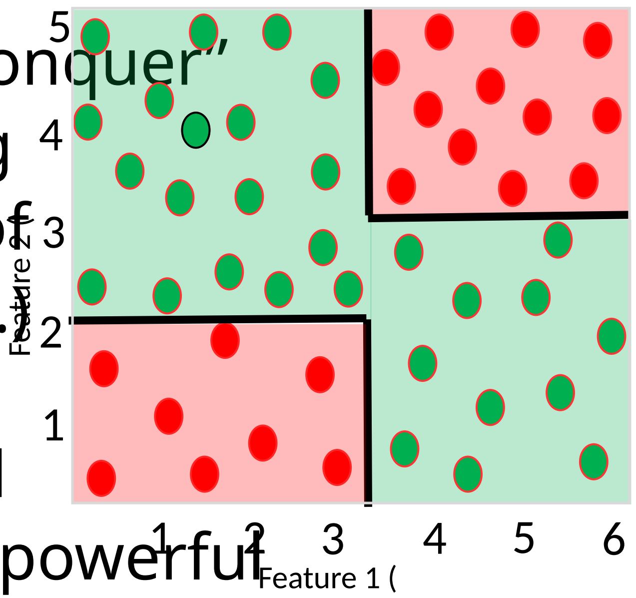
To predict the output for a test point, nearest neighbors will require computing distances from 15 training inputs. DT predicts the label by doing just at most feature-value comparisons! Way faster!

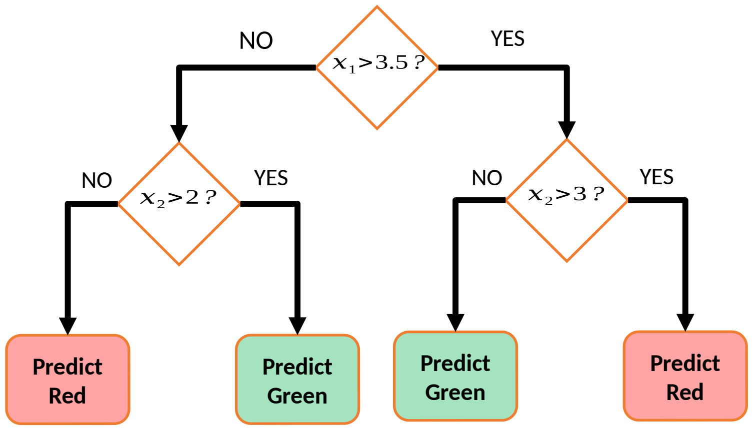
Decision Trees: A Summary

.. thus helping us learn complex rule as a combination of several simpler rules

Some key strengths:

- Simple and easy to interpret
- Nice example of "divide and conquer" paradigm in machine learning
- Easily handle different types of features (real, categorical, etc.)
- Very fast at test time
- Multiple DTs can be combined via ensemble methods: more powerful (e.g., Decision Forests; will see later)
- Used in several real-world ML applications, e.g., recommender systems, gaming (Kinect)





Human-body pose estimation

Some key weaknesses:

- Learning optimal DT is (NP-hard) intractable. Existing algos mostly greedy heuristics