Learning using Decision Trees

CS771: Introduction to Machine Learning

Piyush Rai

Plan for today

- Wrap-up the discussion of nearest neighbors
  - How to speed-up nearest neighbors at test time?
- Model/hyperparameter Selection
- Learning with Decision Trees

Remember: There is no "training" stage in the standard nearest neighbors (unless we are learning the feature of distance function from the training data)

The training data itself is the model that we need to keep at test time. Recall that NN is a memory-based or nonparametric approach

Speeding-up Nearest Neighbors

- Can use techniques to reduce the training set size
  - Several data summarization techniques exist that discard redundant training inputs
  - Now we will require fewer number of distance computations for each test input
- Can use techniques to reduce the data dimensionality (no. of features)
  - Won't reduce no. of distance computations but each distance computation will be faster
- Compressing each input into a small binary vector (a type of dim-red)
  - Distance/similarity computation between bin. vecs is very fast (can even be done in H/W)
- Various other techniques as well, e.g.,
  - Locality Sensitive Hashing (group training inputs into buckets)
  - Clever data structures (e.g., k-D trees) to organize training inputs
  - Use a divide-and-conquer type approach to narrow down the search region

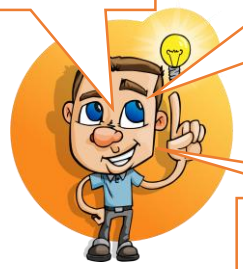
We will look at Decision Trees which is also like a divide-and-conquer approach

Hyperparameter Selection

- Every ML model has some hyperparameters that need to be tuned, e.g.,
  - K in KNN or ϵ in ϵ-NN
  - Choice of distance to use in LwP or nearest neighbors

- Would like to choose h.p. values that would give best performance on test data

Oops, sorry! What to do then?



Okay. So I can try multiple hyperparam values and choose the one that gives the best accuracy on the test data. Simple, isn't it?

Is validation set a good proxy to test set?

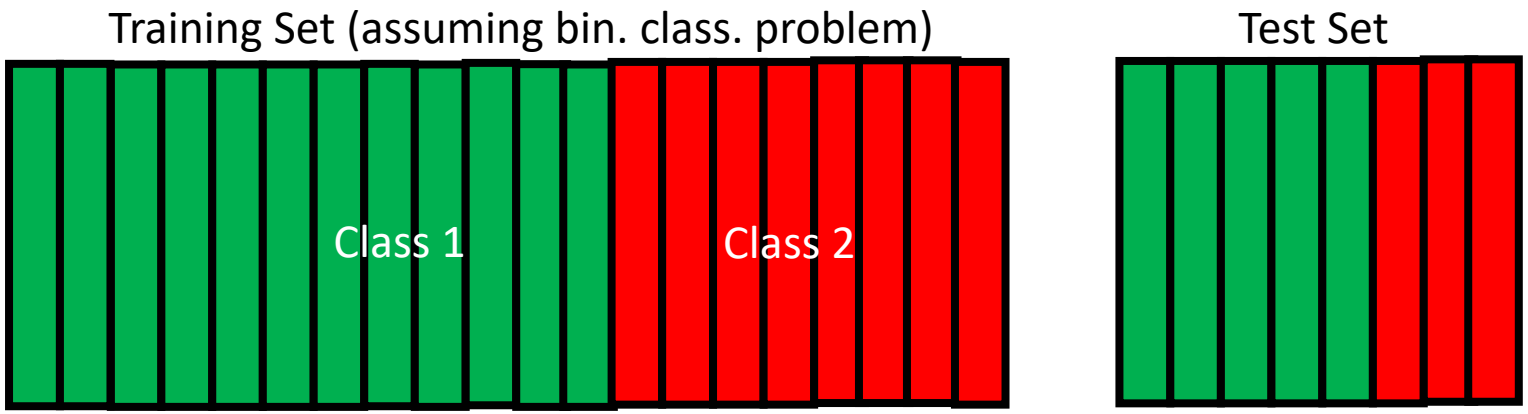
Beware. You are committing a crime. Never Ever touch your test data while building the model

Use cross-validation - usea part of your training data (we will call it "validation/held-out set") as test data. That's not a crime. ☺

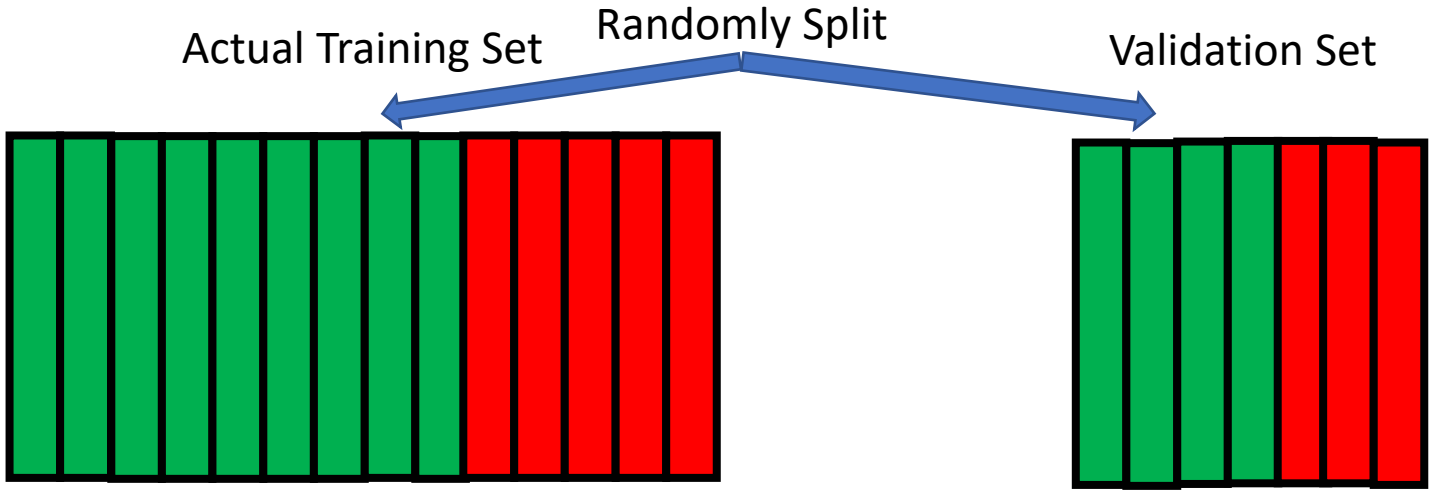
Usually yes since training set and test sets are assumed to be similar (plus, you are careful in choosing your validation set)

Cross-Validation

No peeking while building the model



Note: Not just h.p. selection; we can also use CV to pick the best ML model from a set of different ML models (e.g., say we have to pick between two models we may have trained - LwP and nearest neighbors. Can use CV to choose the better one.



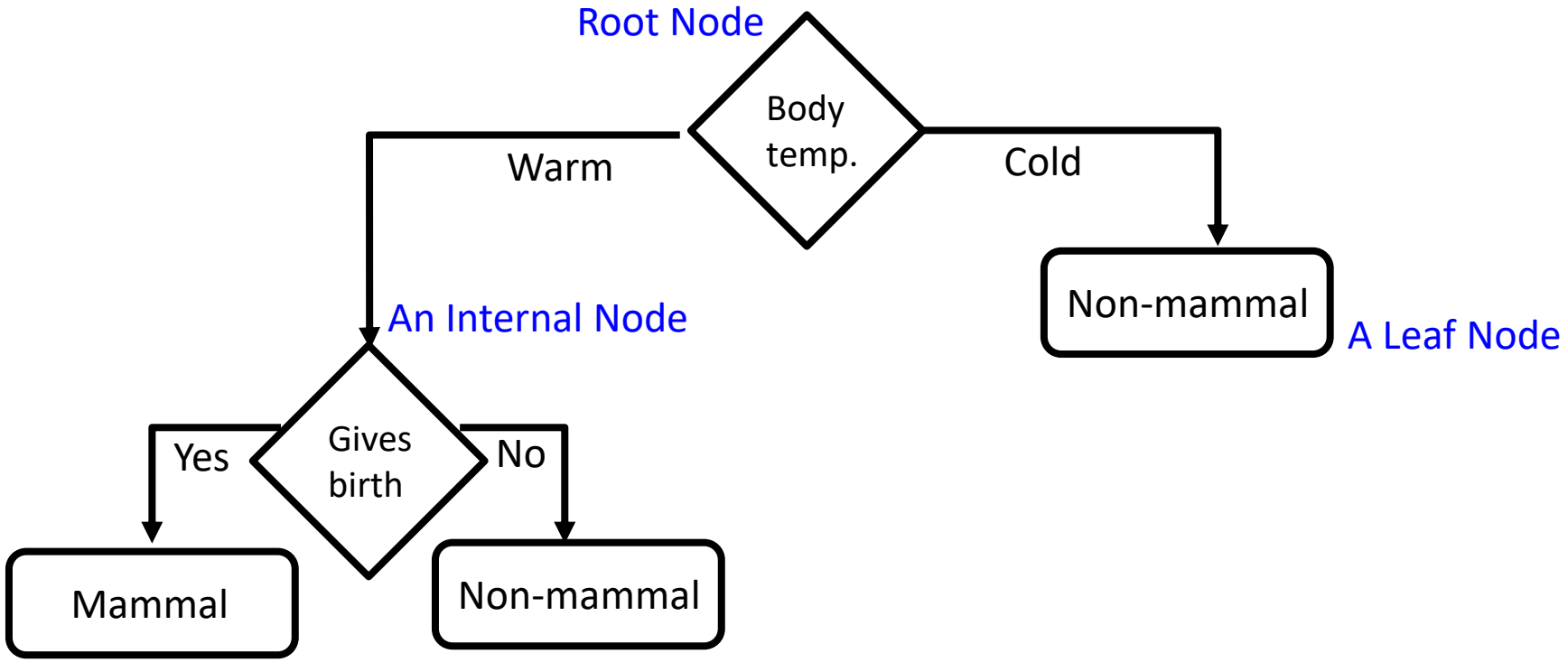
Randomly split the original training data into actual training set and validation set. Using the actual training set, train several times, each time using a different value of the hyperparam. Pick the hyperparam value that gives best accuracy on the validation set

What if the random split is unlucky (i.e., validation data is not like test data)?

If you fear an unlucky split, try multiple splits. Pick the hyperparam value that gives the best average CV accuracy across all such splits. If you are using N splits, this is called N–fold cross validation

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

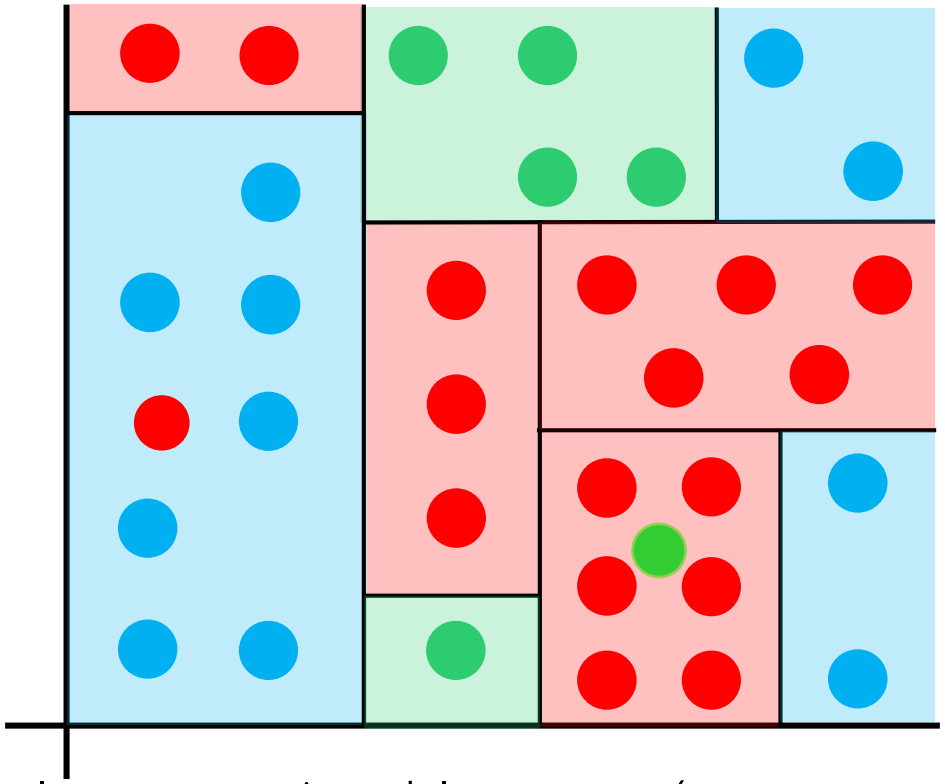
- Decision Tree (DT) learning is about learning such a tree from labeled data

Decision Trees for Supervised Learning

- The basic idea is very simple
- Recursively partition the training data into homogeneous regions

What do you mean by "homogeneous" regions?

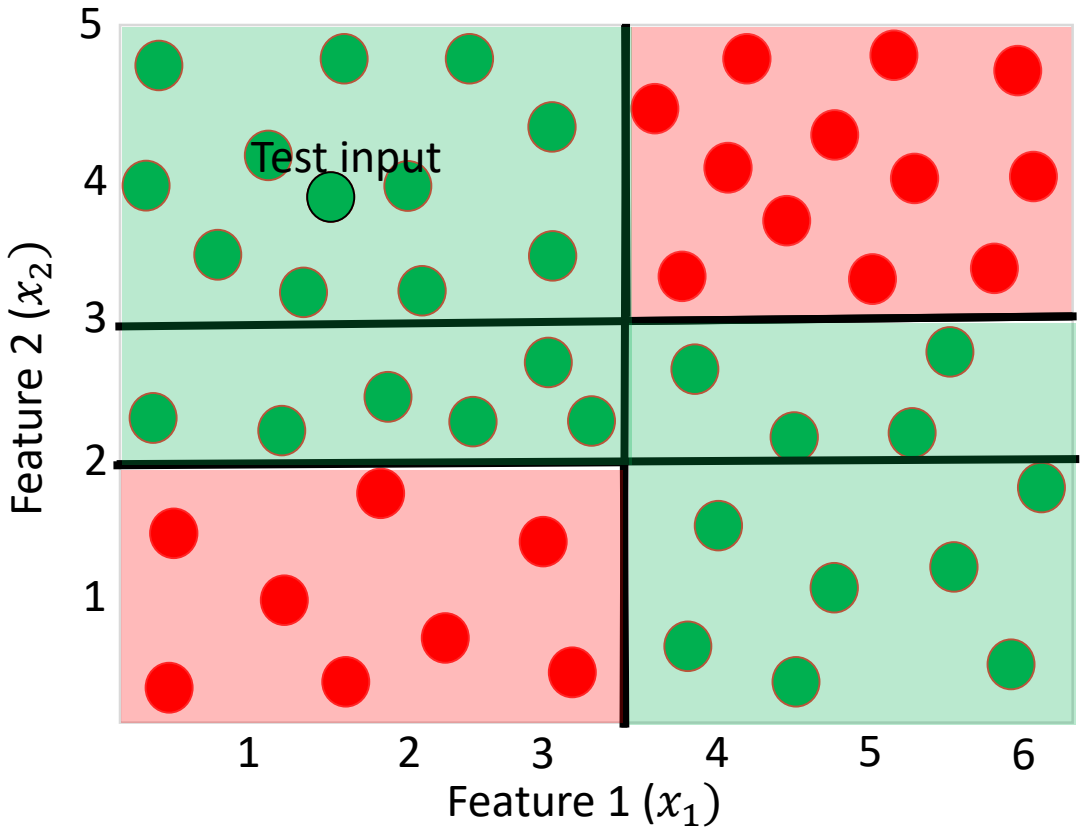
A homogeneous region will have all (or a majority of) training inputs with the same/similar outputs

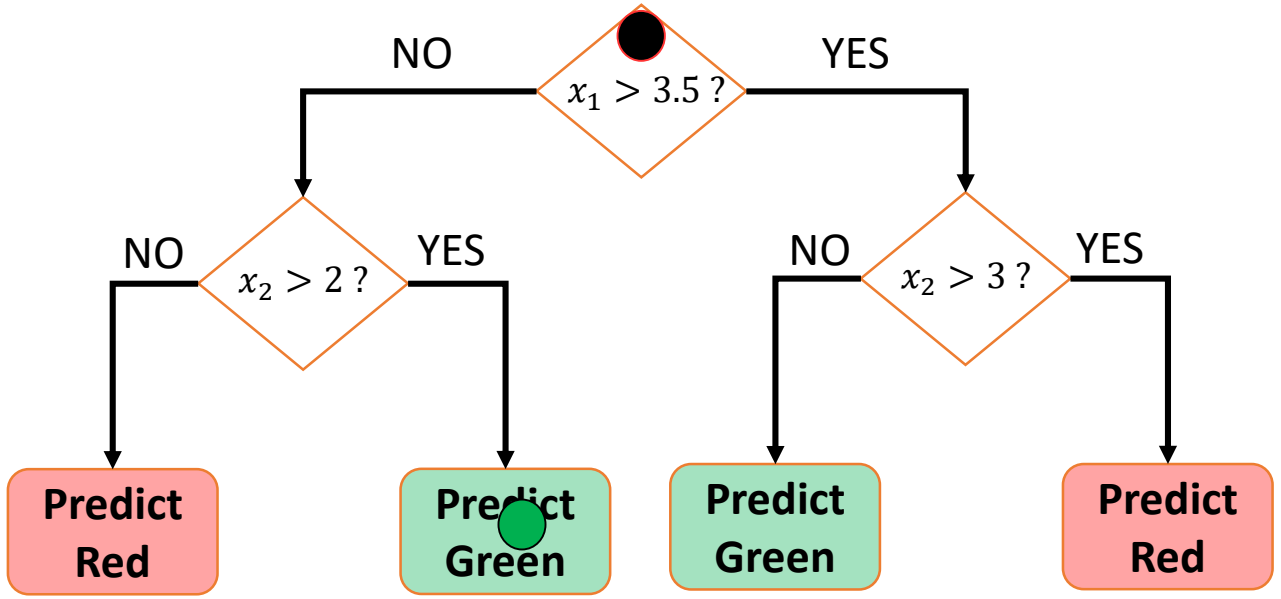


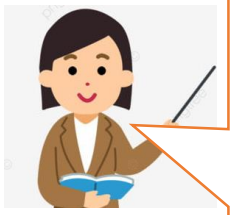
Even though the rule within each group is simple, we are able to learn a fairly sophisticated model overall (note in this example, each rule is a simple horizontal/vertical classifier but the overall decision boundary is rather sophisticated)

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

Decision Trees for Classification







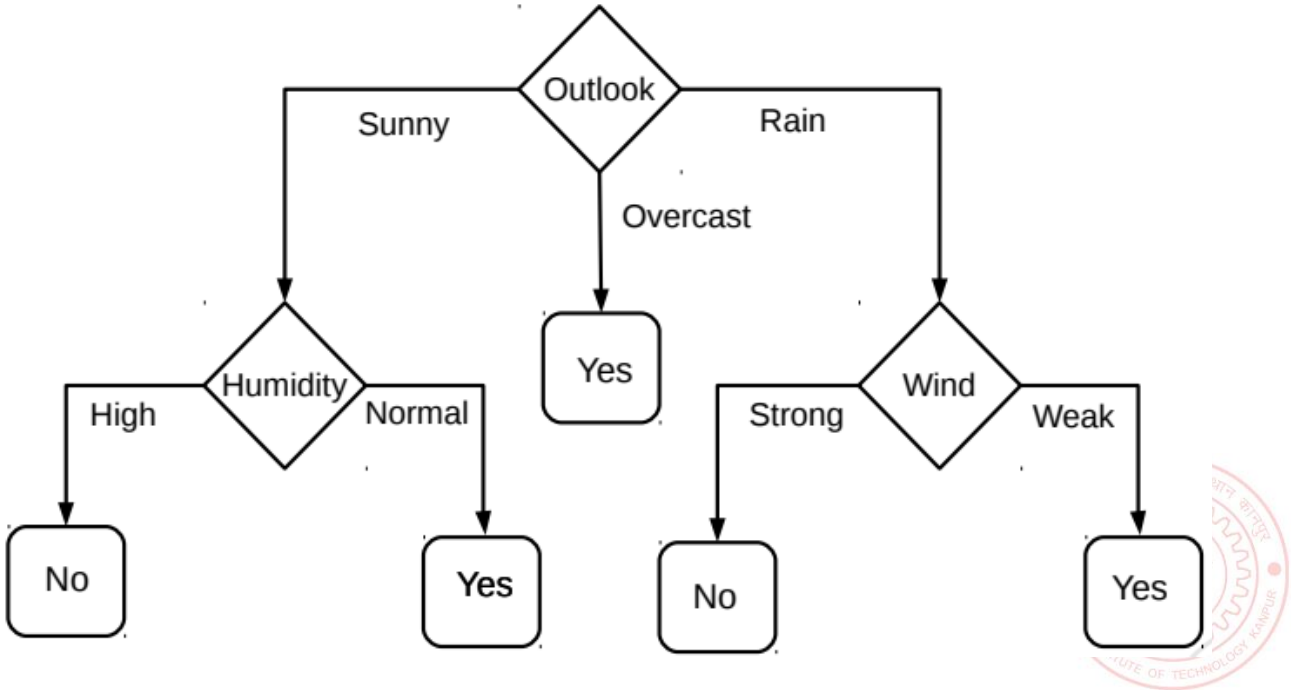
DT is very efficient at test time: To predict the label of a test point, nearest neighbors will require computing distances from 48 training inputs. DT predicts the label by doing just 2 feature-value comparisons! Way more fast!!!

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)
- Below Left: Training data, Below Right: A decision tree constructed using this data

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

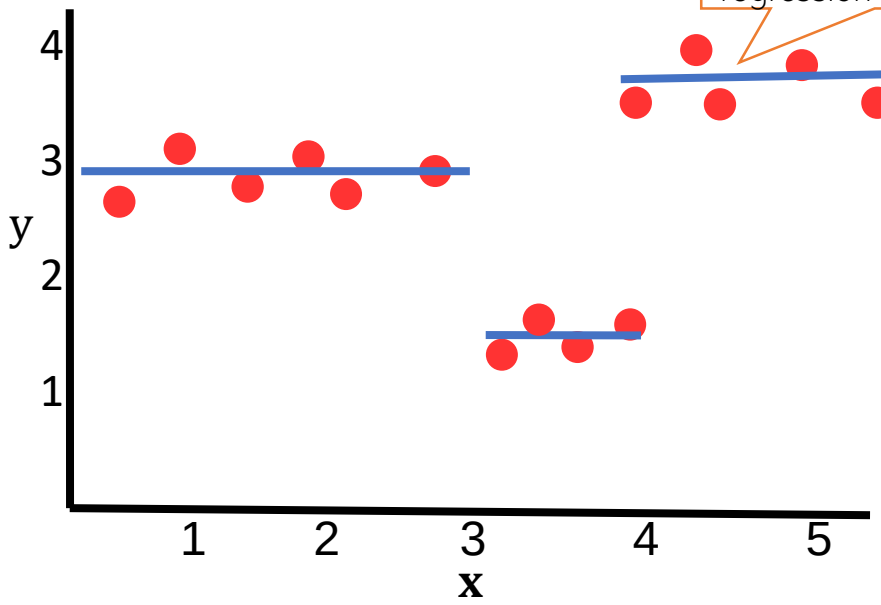


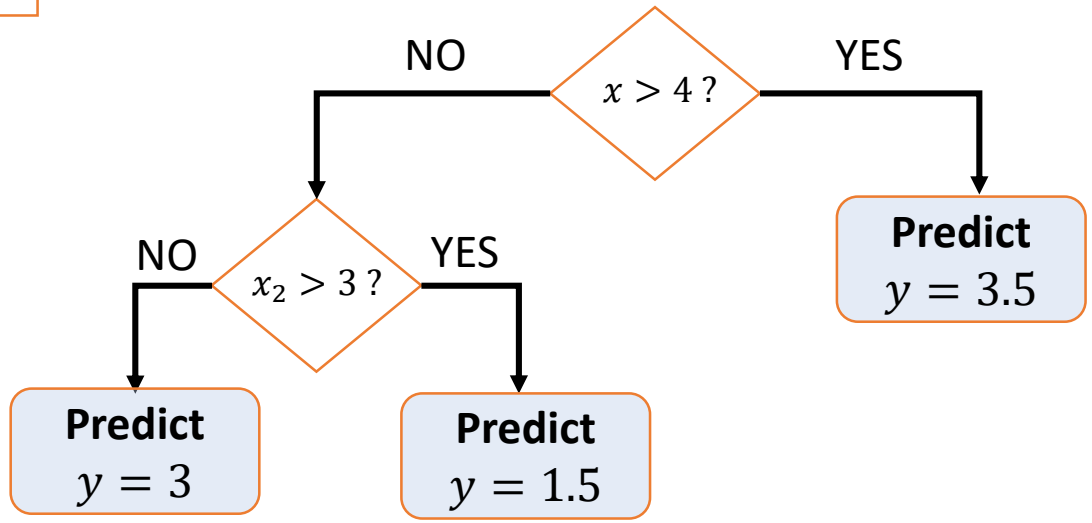
Example credit: Tom Mitchell

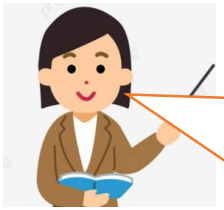
Decision Trees for Regression

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



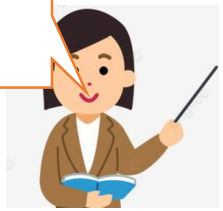




To predict the output for a test point, nearest neighbors will require computing distances from 15 training inputs. DT predicts the label by doing just at most feature-value comparisons! Way more fast!!!

Decision Trees: Some Considerations

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

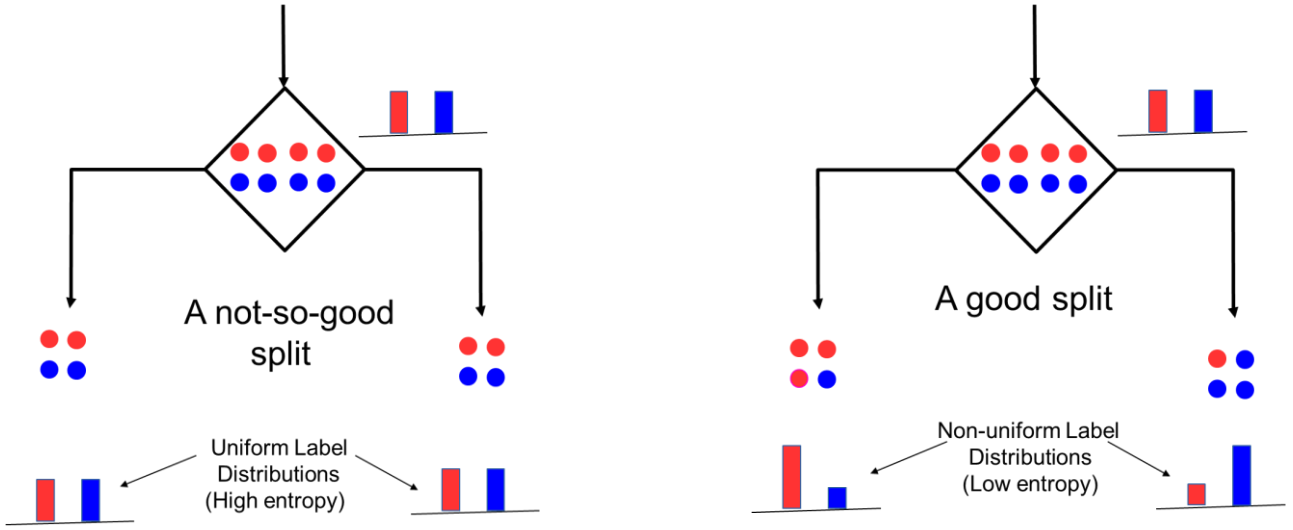
- Split criterion at internal nodes
  - Use another classifier?
  - Or maybe by doing a simpler test?

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

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

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

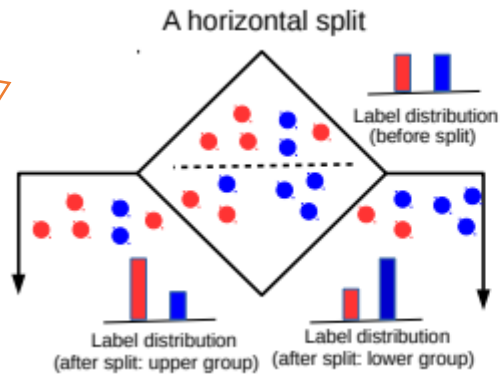


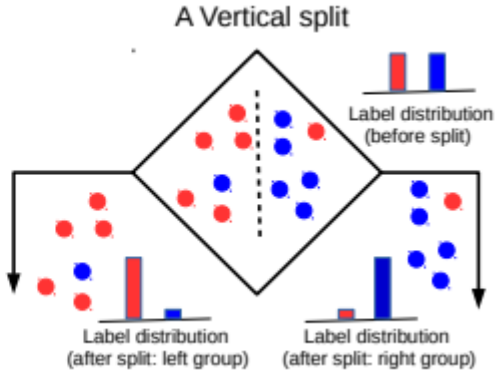
- For classification problems (discrete outputs), entropy is a measure of purity
  - Low entropy ⇒ high purity (less uniform label distribution)
  - Splits that give the largest reduction (before split vs after split) in entropy are preferred (this reduction is also known as "information gain")

Techniques to Split at Internal Nodes?

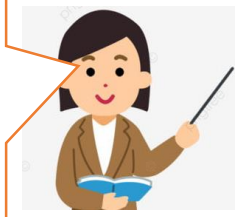
- Each internal node decides which outgoing branch an input should be sent to
- This decision/split can be done using various ways, e.g.,
  - Testing the value of a single feature at a time (such internal node called "Decision Stump")

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)

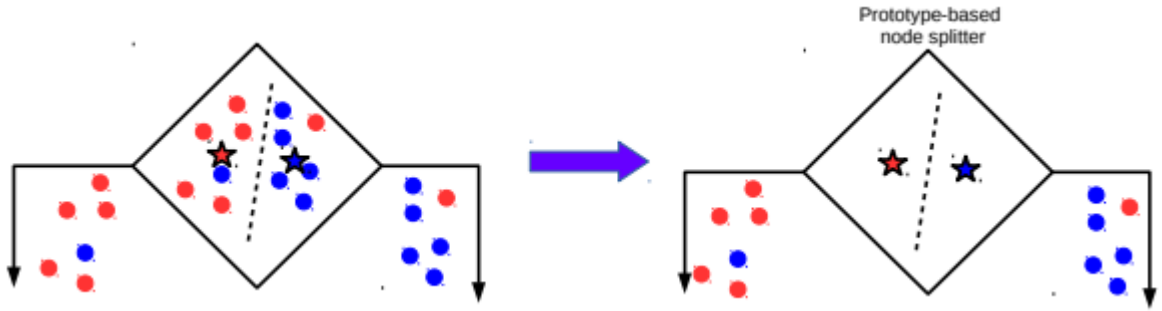




DT methods based on testing a single feature at each internal node are faster and more popular (e.g., ID3, C4.5 algos)



  - Learning a classifier (e.g., LwP or some more sophisticated classifier)

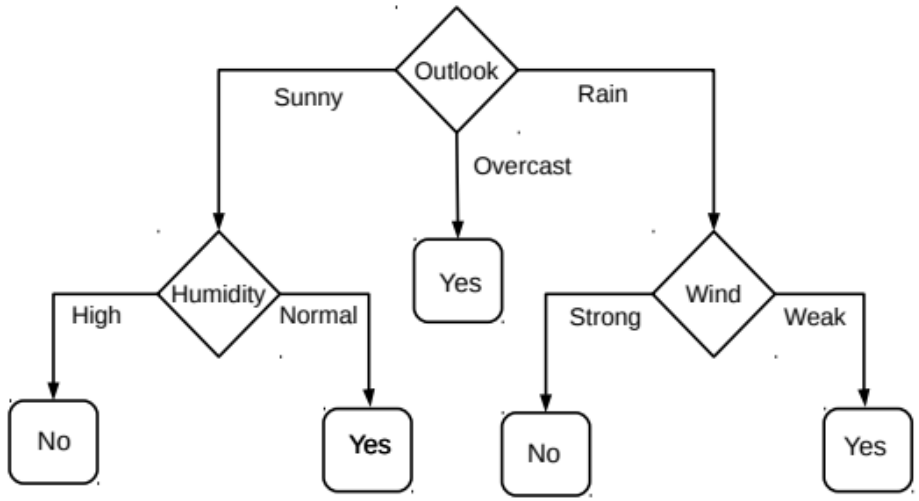


DT methods based on learning and using a separate classifier at each internal node are less common. But this approach can be very powerful and are sometimes used in some advanced DT methods

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

Next Class

- Wrap-up Decision Trees
- Linear Models