CS 307: Random Forests

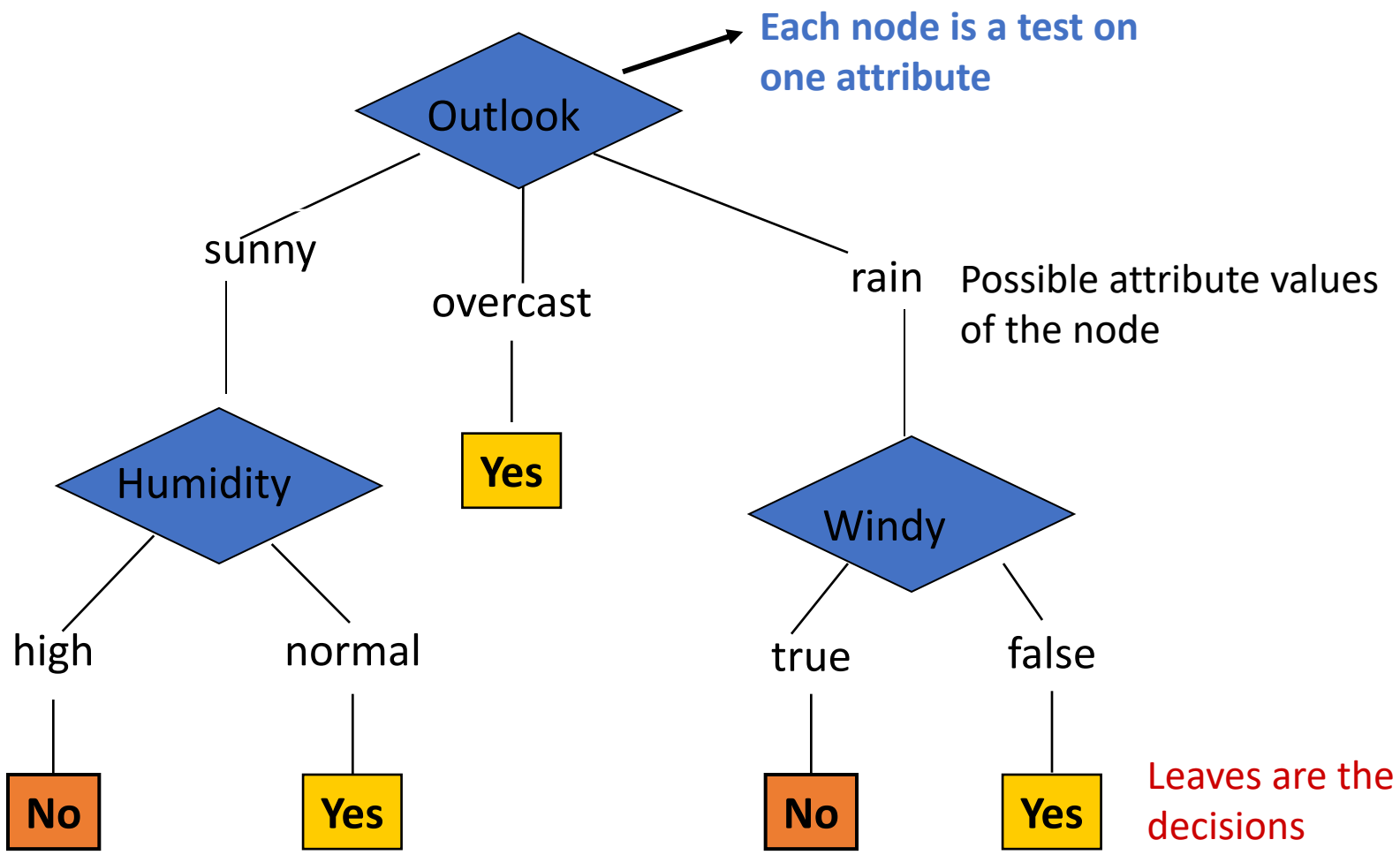
Today’s goal

- Decision trees
- Random forest (RF)
- Tuning the hyperparameters of a RF
- Feature interpretation in a RF

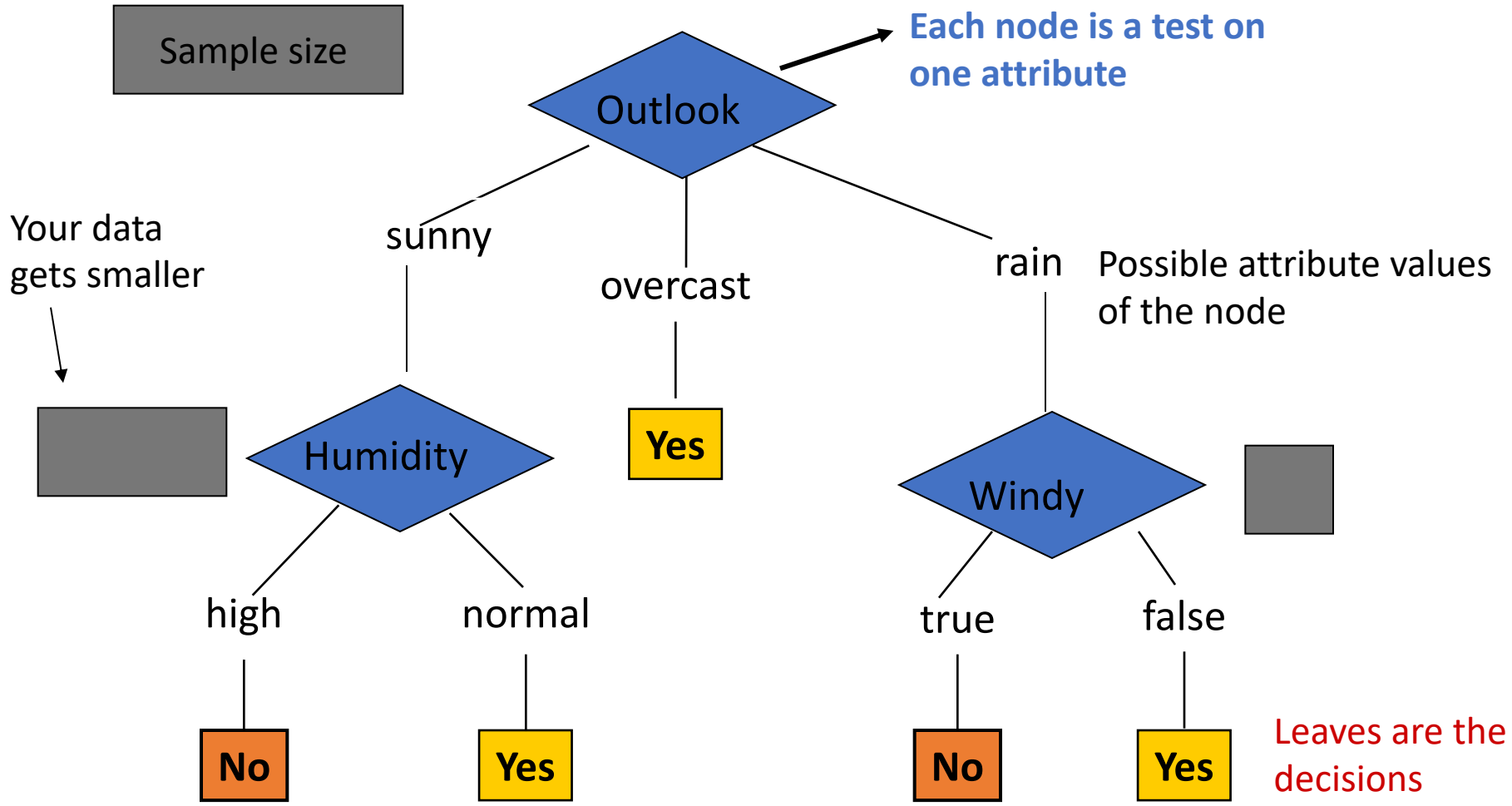
Decision Trees

- Non-linear classifier
- Easy to use
- Easy to interpret
- Susceptible to overfitting but can be avoided

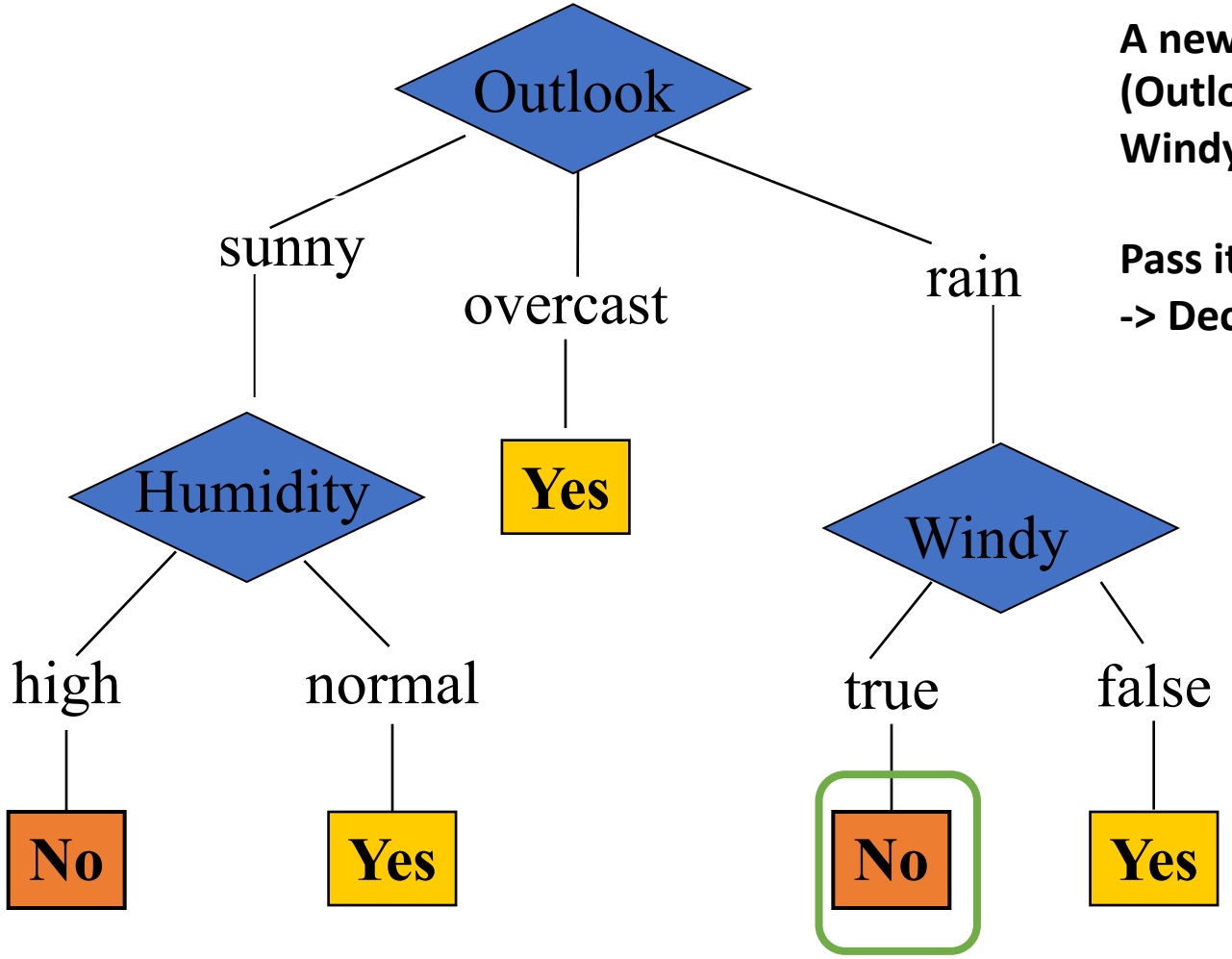
Anatomy of a decision tree



Anatomy of a decision tree



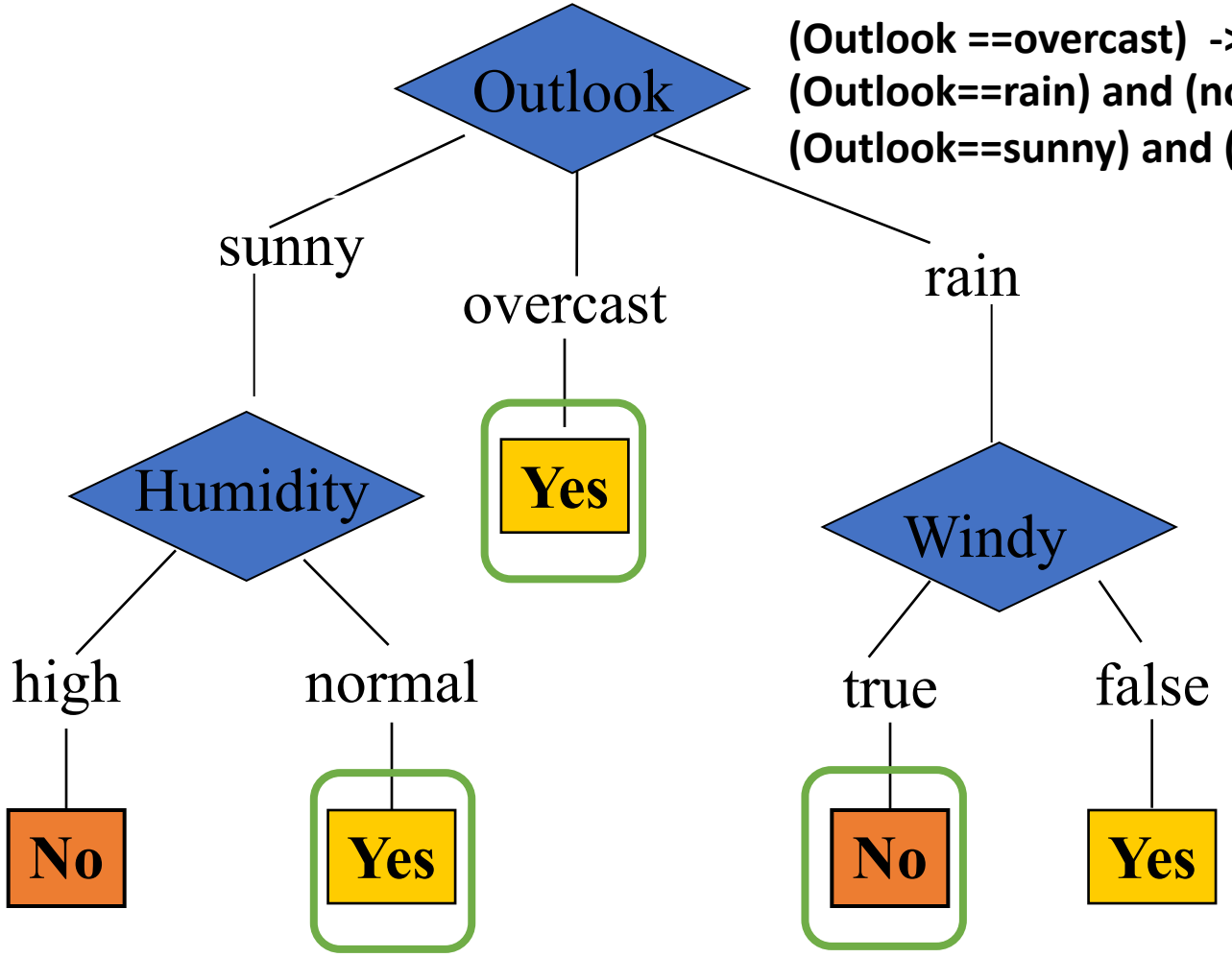
Case Study: To 'play tennis' or not.



A new test example:
(Outlook==rain) and (not Windy==false)

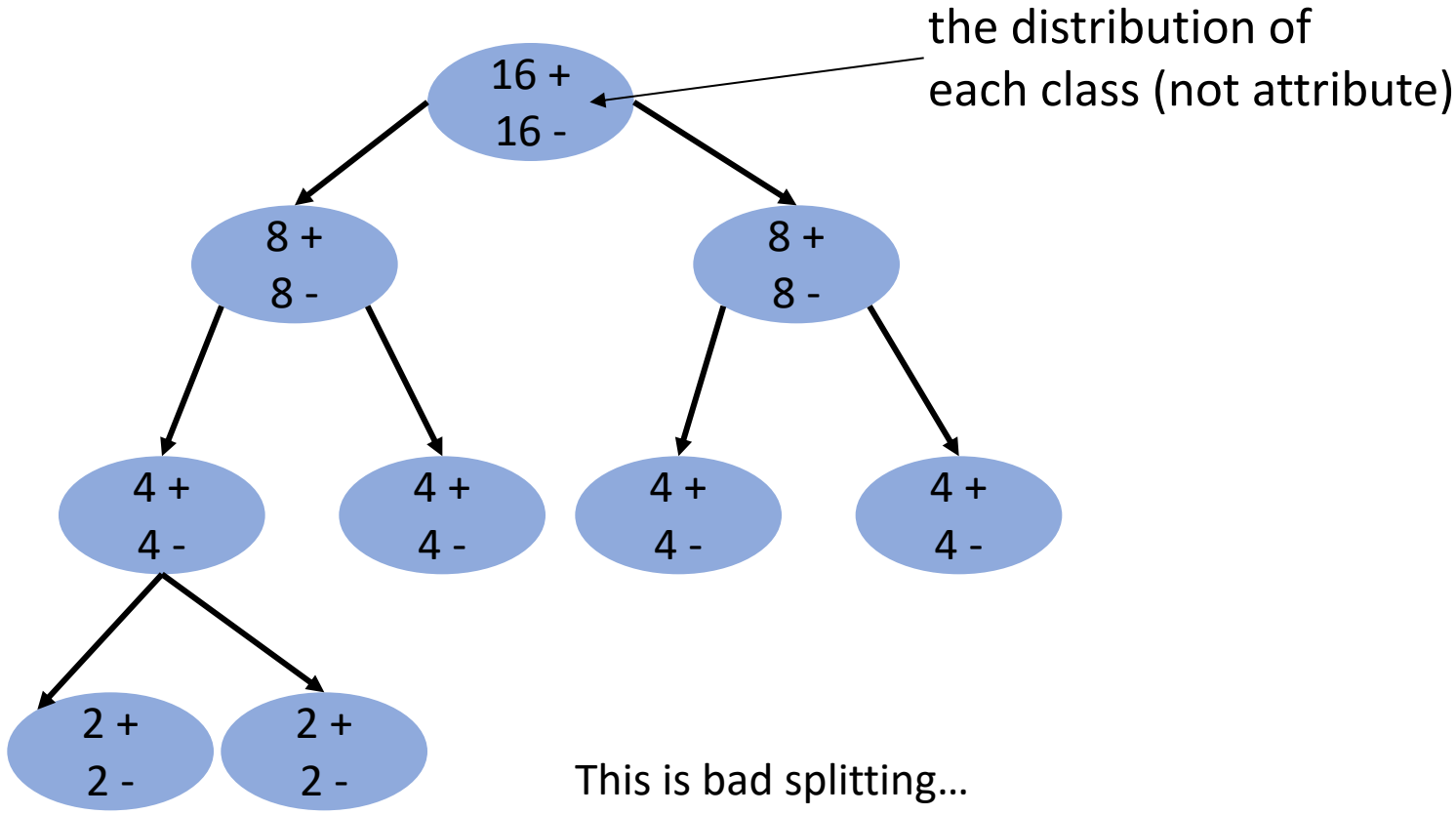
Pass it on the tree
-> Decision is ?

Case Study: To ‘play tennis’ or not.



(Outlook ==overcast) -> ?
(Outlook==rain) and (not Windy==false) ->?
(Outlook==sunny) and (Humidity=normal) ->?

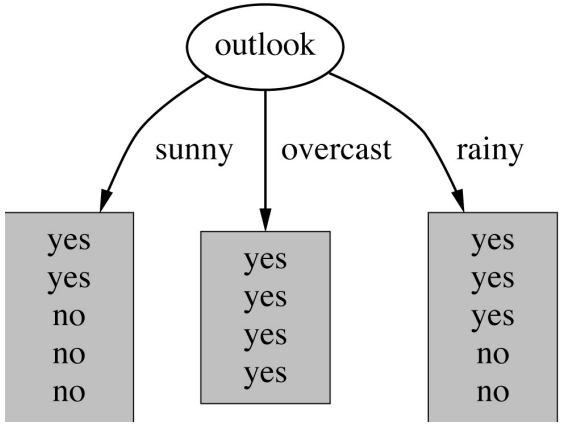
Which attribute to select for splitting?

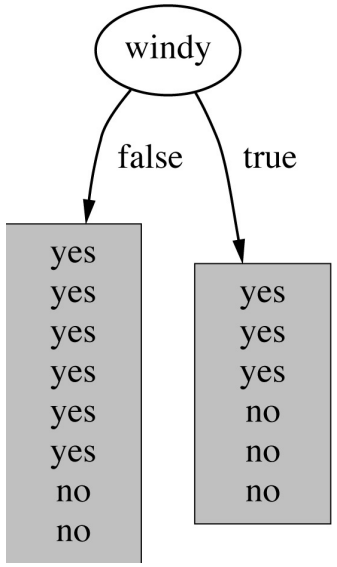


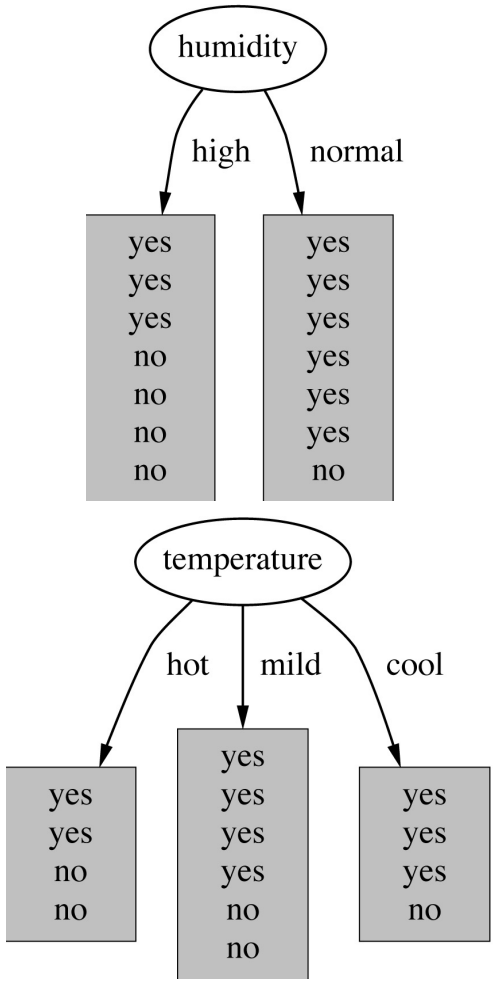
How do we choose the Attributes?

Which attribute should be used as the test?

Intuitively, you would prefer the one that *separates* the training examples as much as possible.







Information Gain

- Information gain is one criteria to decide on the attribute.
- Sklearn function:
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Information

- Imagine:
- 1. Someone is about to tell you your own name
- 2. You are about to observe the outcome of a dice roll
- 2. You are about to observe the outcome of a coin flip
- 3. You are about to observe the outcome of a biased coin flip

- Each situation have a different *amount of uncertainty* as to what outcome you will observe.

Information

- Information:
- reduction in uncertainty (amount of surprise in the outcome)

$$I(E) = \log_2 \frac{1}{p(x)} = -\log_2 p(x)$$

If the probability of this event happening is small and it happens the information is large.

- Observing the outcome of a coin flip is head $\longrightarrow$ $I = -\log_2 1/2 = 1$
- Observe the outcome of a dice is 6 $\longrightarrow$ $I = -\log_2 1/6 = 2.58$

Entropy

- The *expected amount of information* when observing the output of a random variable X

$$H(X) = E(I(X)) = \sum_i p(x_i) I(x_i) = -\sum_i p(x_i) \log_2 p(x_i)$$

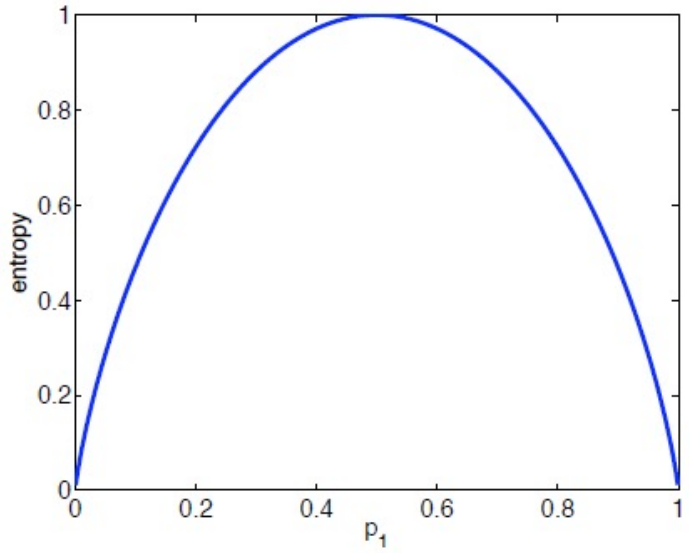
If there X can have 8 outcomes and all are equally likely

$$H(X) == -\sum_i 1/8 \log_2 1/8 = 3 \quad \textit{bits}$$

Entropy

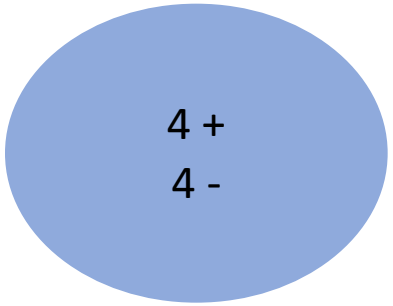
- Equality holds when all outcomes are equally likely

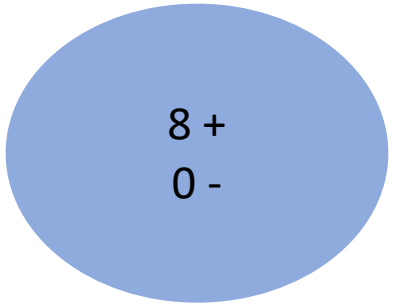
- The more the probability distribution deviates from uniformity the lower the entropy



Entropy, purity

- Entropy measures the purity





The distribution is less uniform
-> Entropy is lower
-> The node is purer

Conditional entropy

$$H(X) = -\sum_i p(x_i) \log_2 p(x_i)$$

$$H(X \mid Y) = -\sum_j p(y_j) H(X \mid Y = y_j)$$

$$= -\sum_j p(y_j) \sum_i p(x_i \mid y_j) \log_2 p(x_i \mid y_j)$$

Information gain

- $IG(X,Y)=H(X)-H(X|Y) =H(Y)-H(Y|X)$

Reduction in uncertainty by knowing Y

Information gain:
(information before split) – (information after split)

Example

Attributes Labels

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

Which one do we choose X1 or X2?

$$IG(X1,Y) = H(Y) - H(Y|X1)$$

$$H(Y) = -(5/10)\log(5/10) - 5/10\log(5/10) = 1$$

$$H(Y|X1) = P(X1=T)H(Y|X1=T) + P(X1=F)H(Y|X1=F)$$

$$= 4/10\,(1\log 1 + 0\log 0) + 6/10\,(5/6\log 5/6 + 1/6\log 1/6)$$

$$= 0.39$$

Information gain (X1,Y)= 1-0.39=0.61

Which one do we choose?

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

Information gain (X1,Y)= 0.61

Information gain (X2,Y)= 0.12

Pick the variable which provides the **most** information gain about Y

Pick X1

Recurse on branches

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

One branch

The other branch

Caveats

- The number of possible values influences the information gain.

- The more possible values, the higher the gain (the more likely it is to form small, but pure partitions)

Purity (diversity) measures

- Purity (Diversity) Measures:
- – Gini (population diversity) Gini index = $1 - \sum (P(x = k))^2$
- – **Information Gain**
- – Chi-square Test

Overfitting

- You can perfectly fit to any training data
- Zero bias, high variance

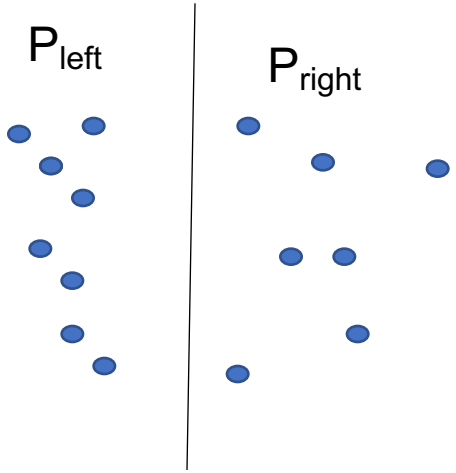
- Two approaches:
- 1. Stop growing the tree when further splitting the data does not yield an improvement
- 2. Grow a full tree, then prune the tree, by eliminating nodes.

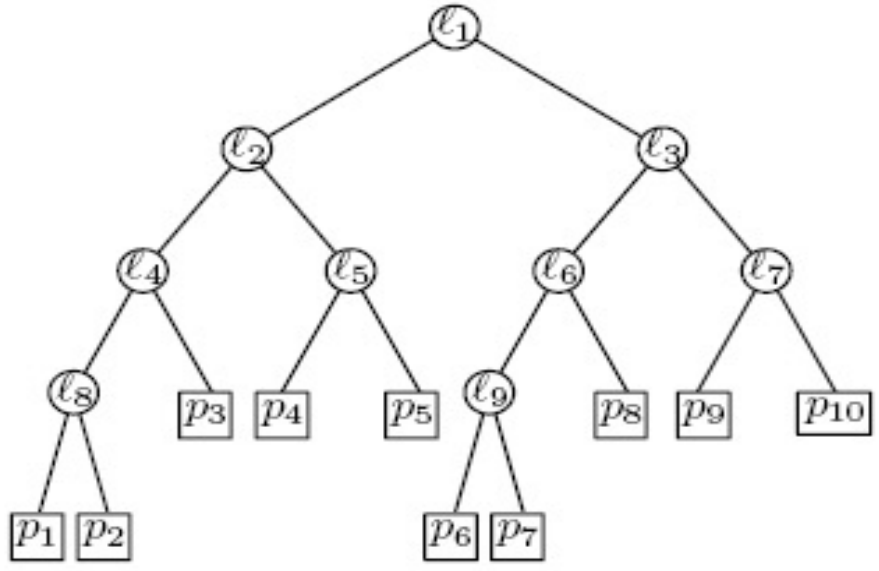
Decision Trees

- Recall:
  - Node splitting criteria – information gain
  - Binary tree
  - K-d tree
  - Can we search for the best tree?

KD Tree

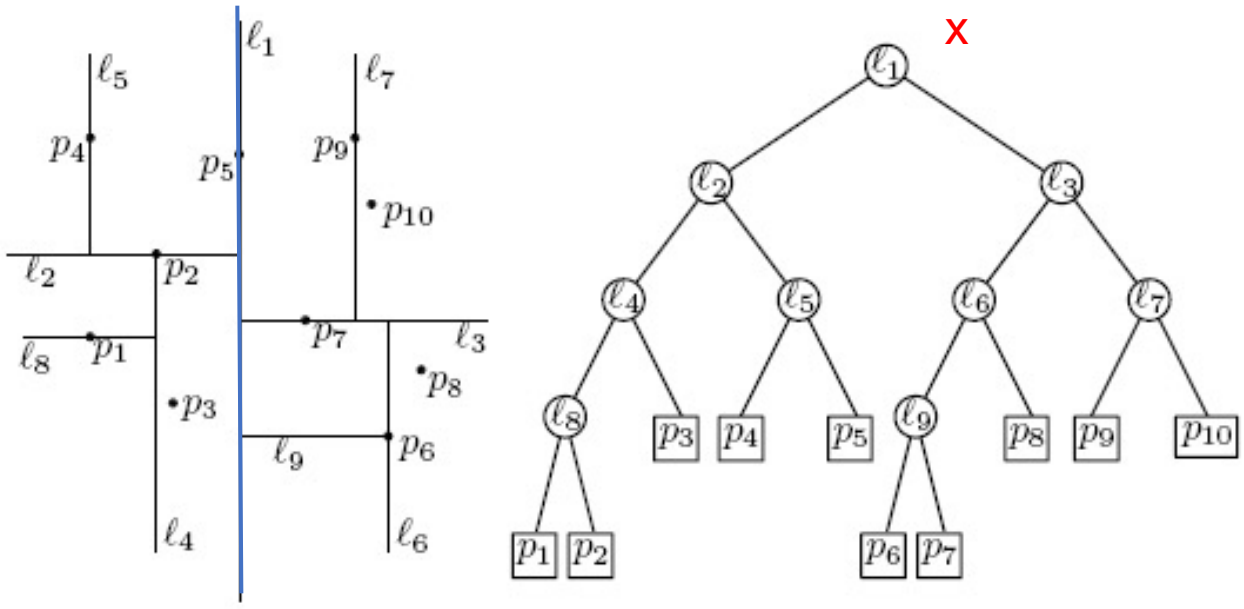
- Every node (except leaves) represents a hyperplane that divides the space into two parts.
- Points to the left (right) of this hyperplane represent the left (right) sub-tree of that node.





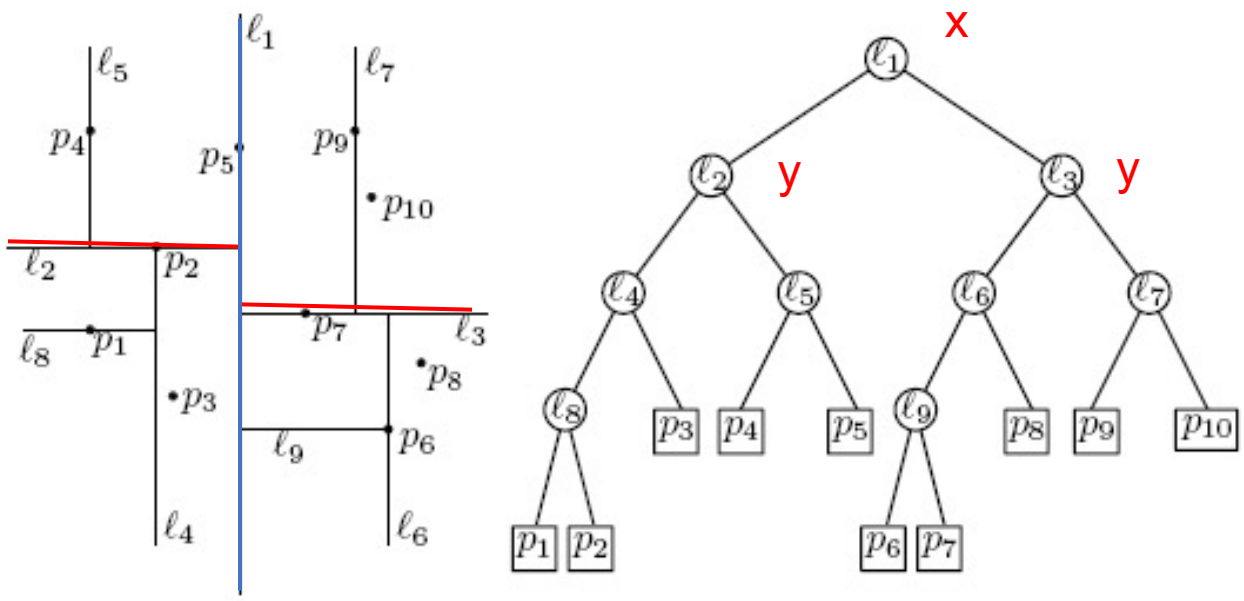
KD Tree - Example

Split by x-coordinate: split by a vertical line that has approximately half the points left or on, and half right.



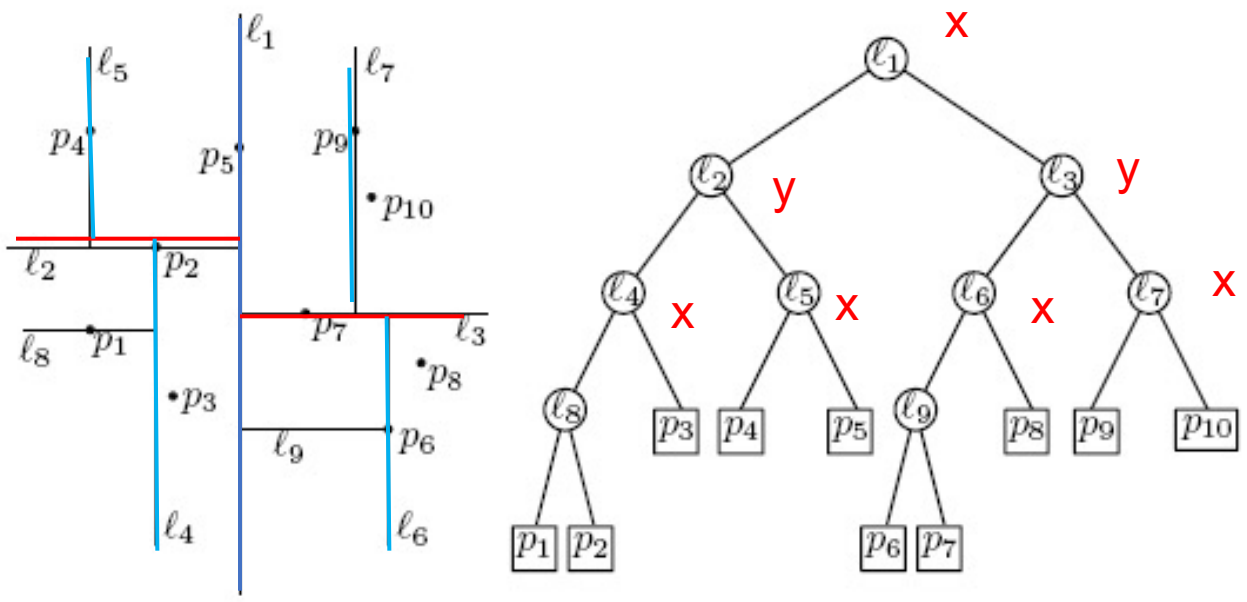
KD Tree - Example

Split by y-coordinate: split by a horizontal line that has half the points below or on and half above.



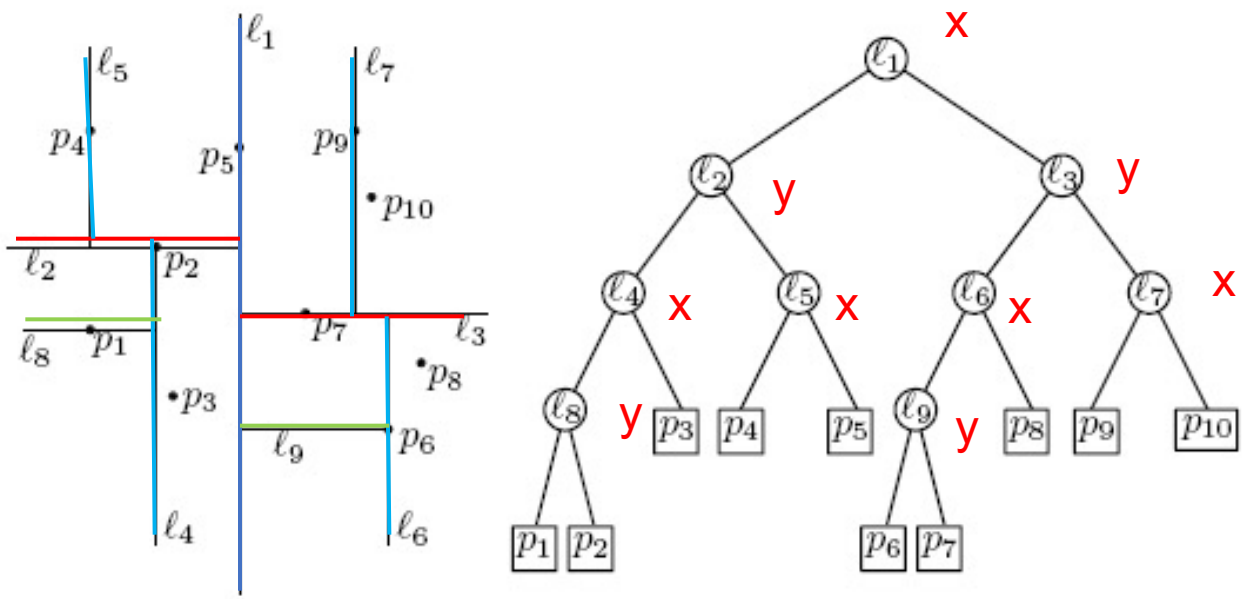
KD Tree - Example

Split by x-coordinate: split by a vertical line that has half the points left or on, and half right.

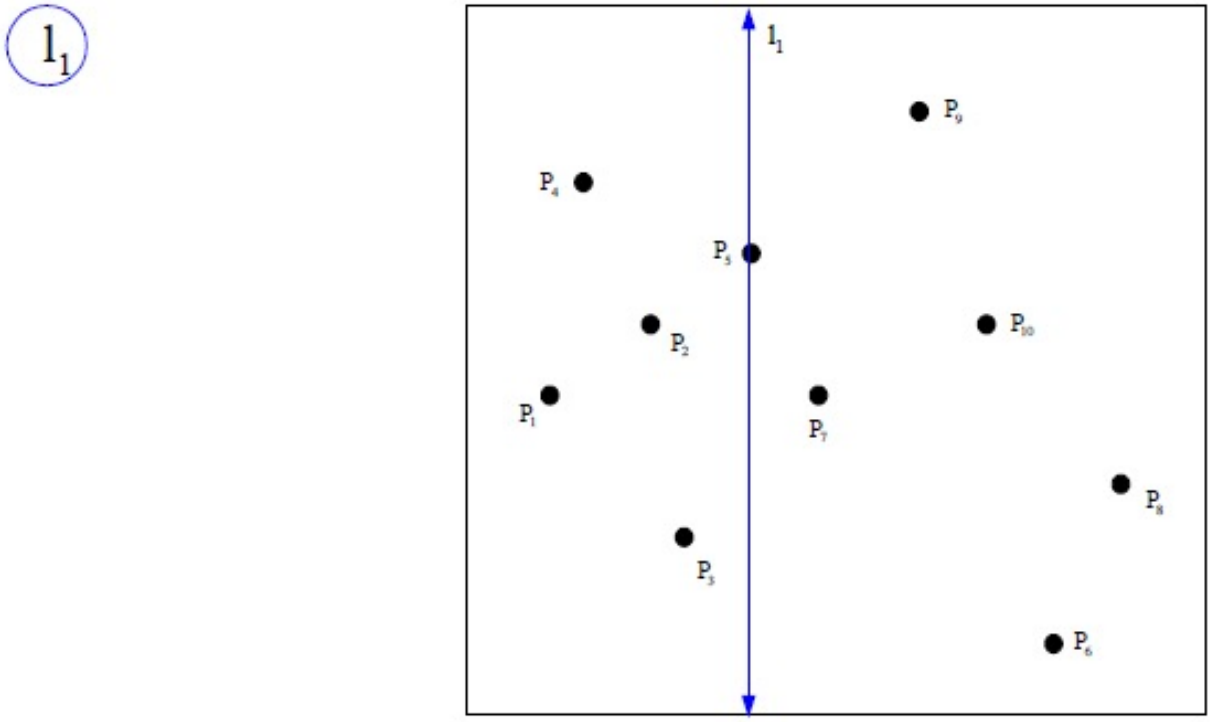


KD Tree - Example

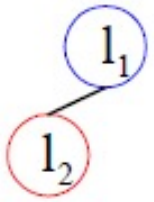
Split by y-coordinate: split by a horizontal line that has half the points below or on and half above.

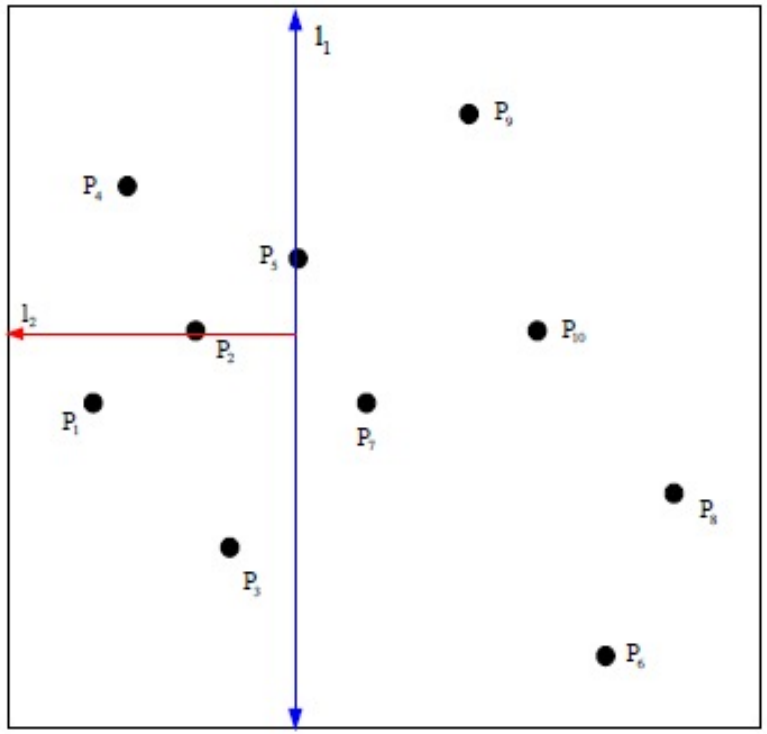


Example – using median (data stored at the leaves)

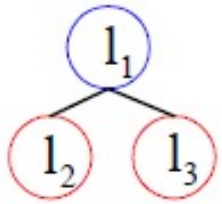


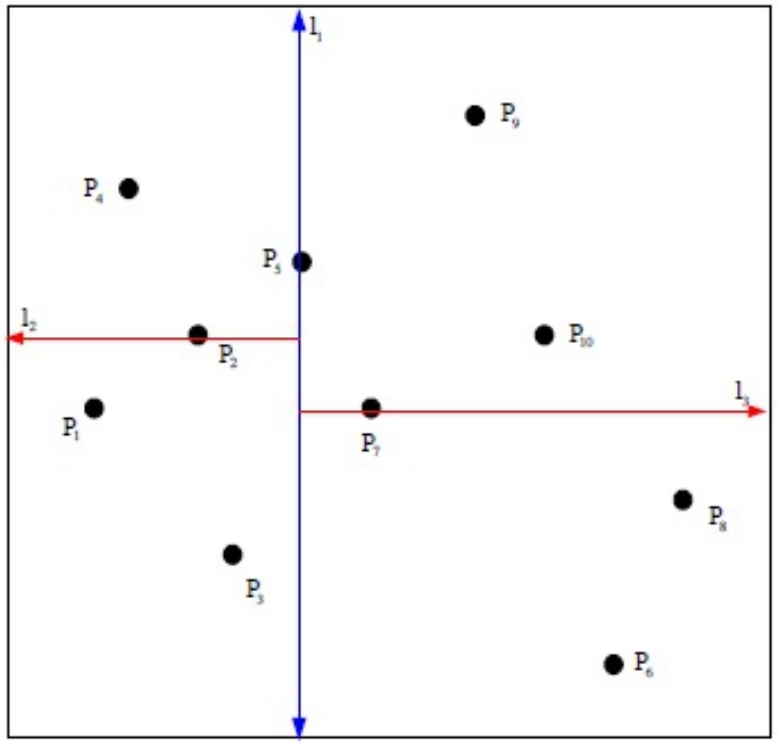
Example – using median (data stored at the leaves)



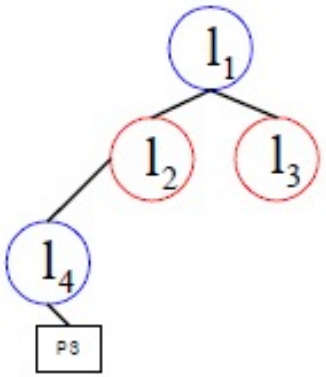


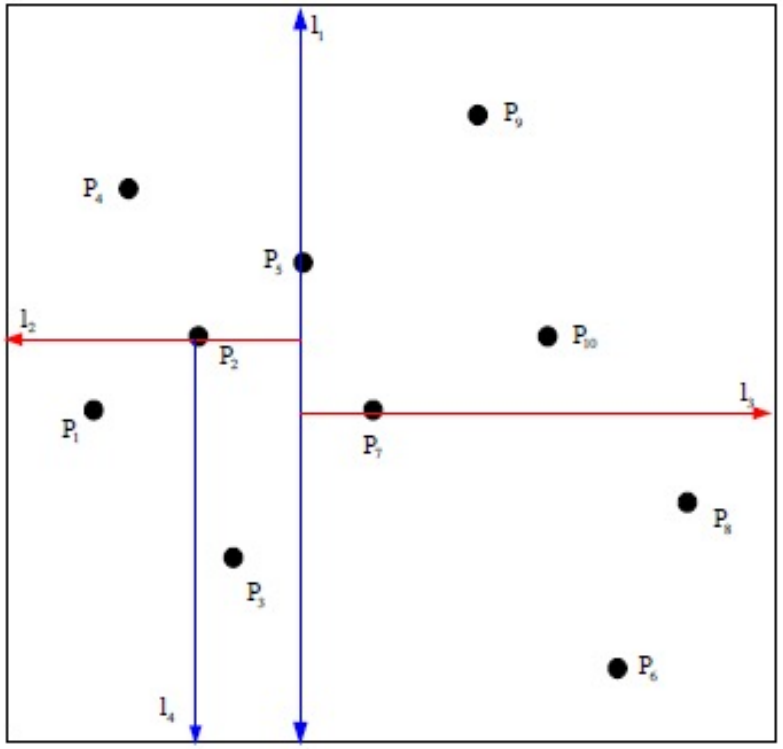
Example – using median (data stored at the leaves)



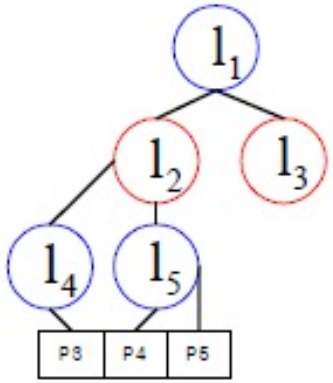


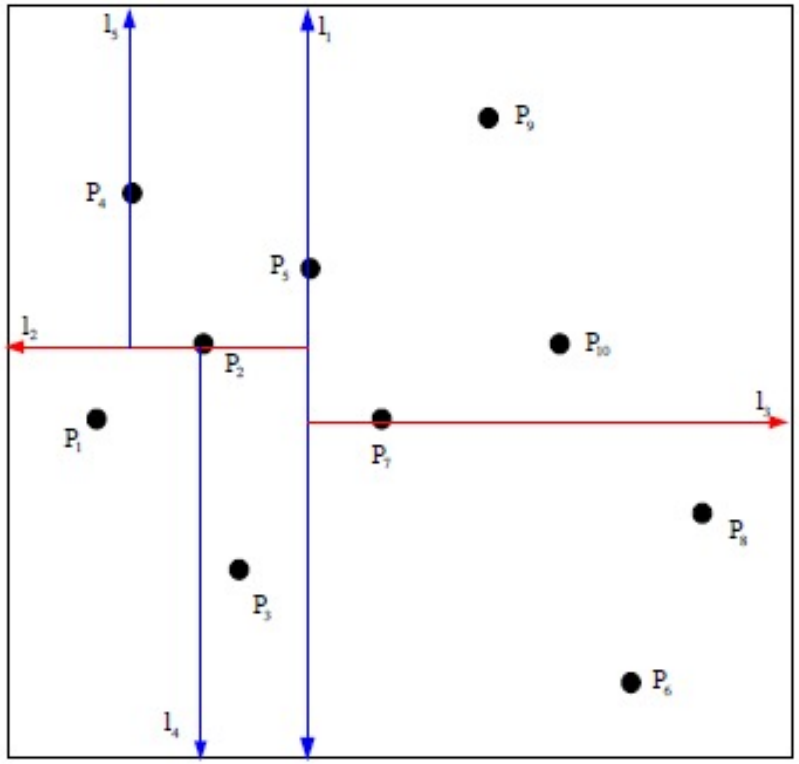
Example – using median (data stored at the leaves)



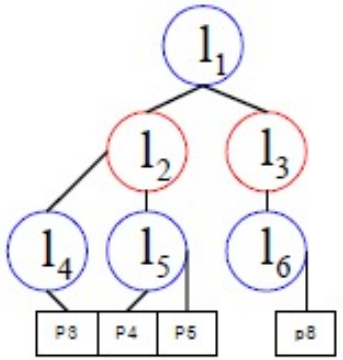


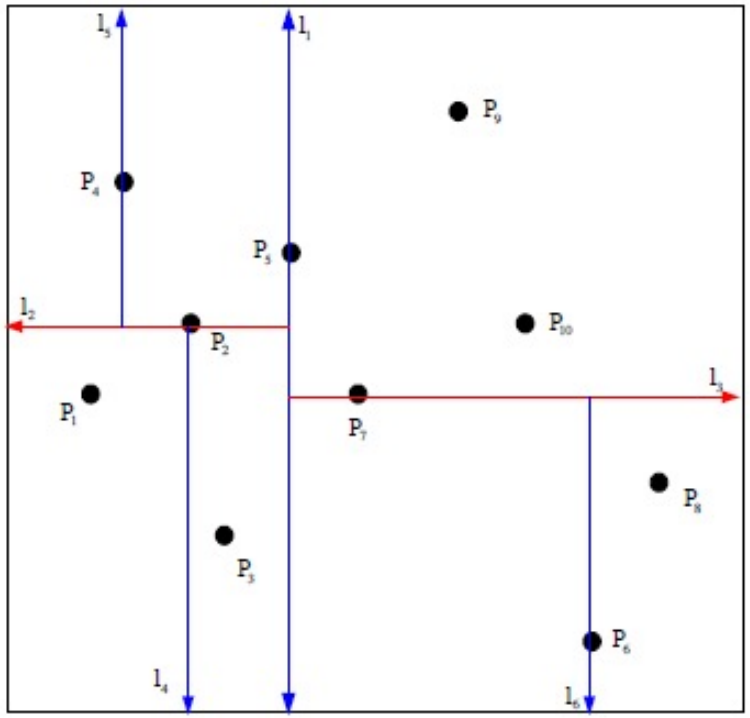
Example – using median (data stored at the leaves)



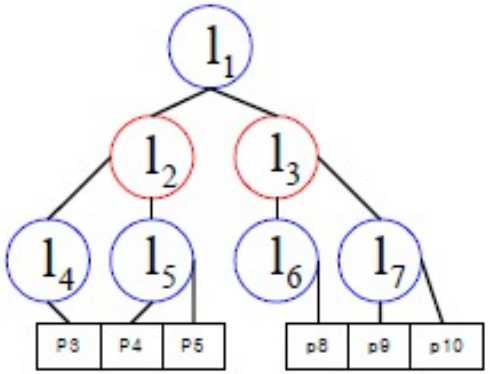


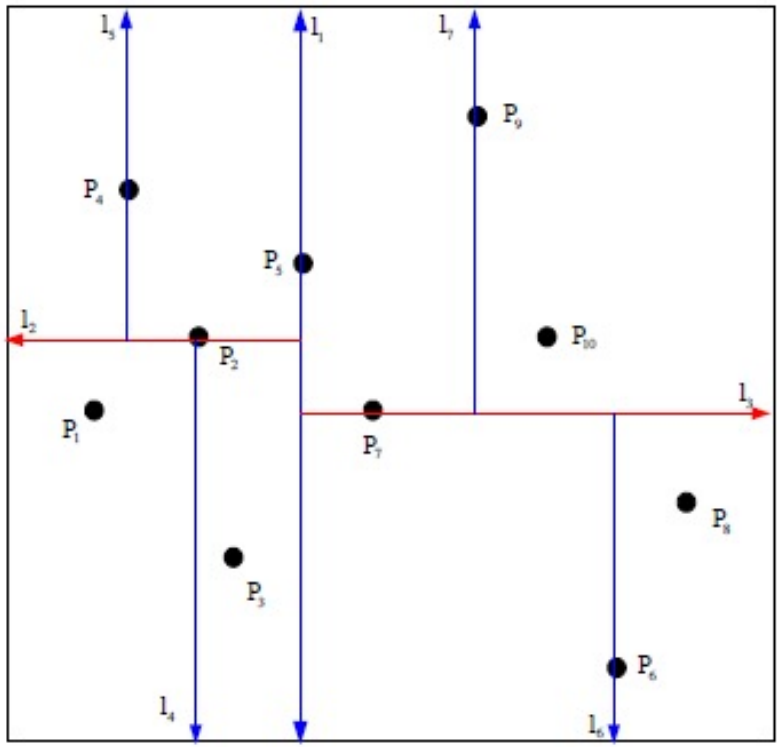
Example – using median (data stored at the leaves)



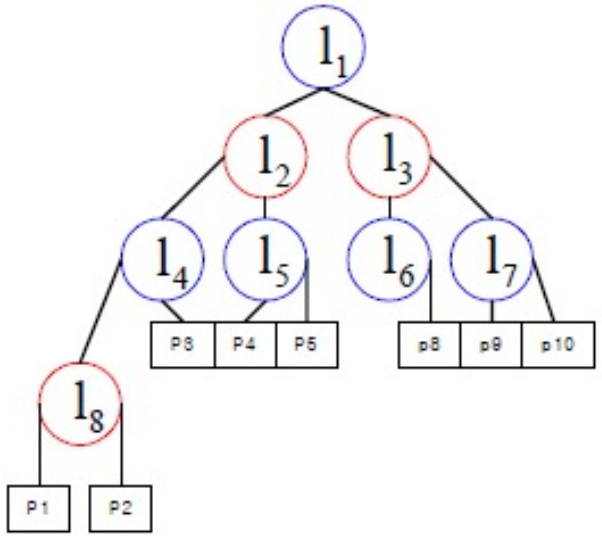


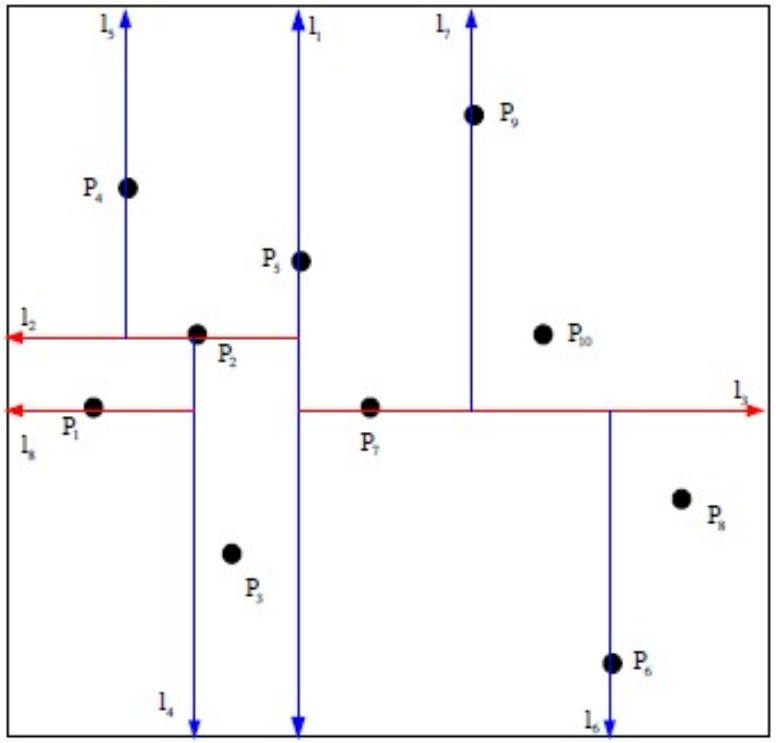
Example – using median (data stored at the leaves)



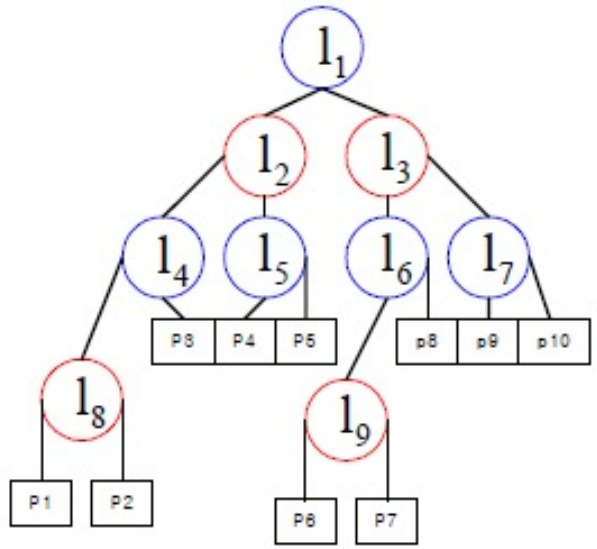


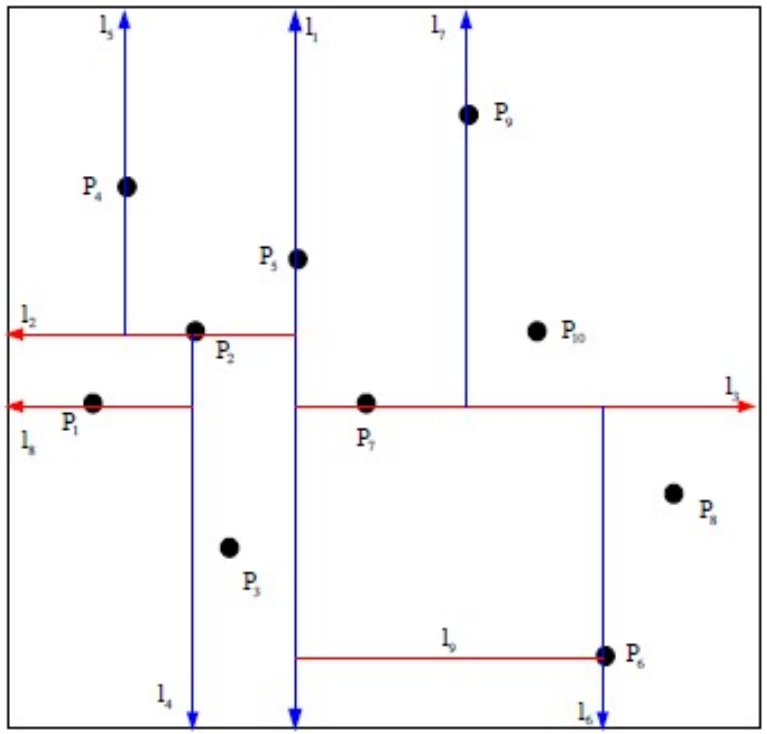
Example – using median (data stored at the leaves)



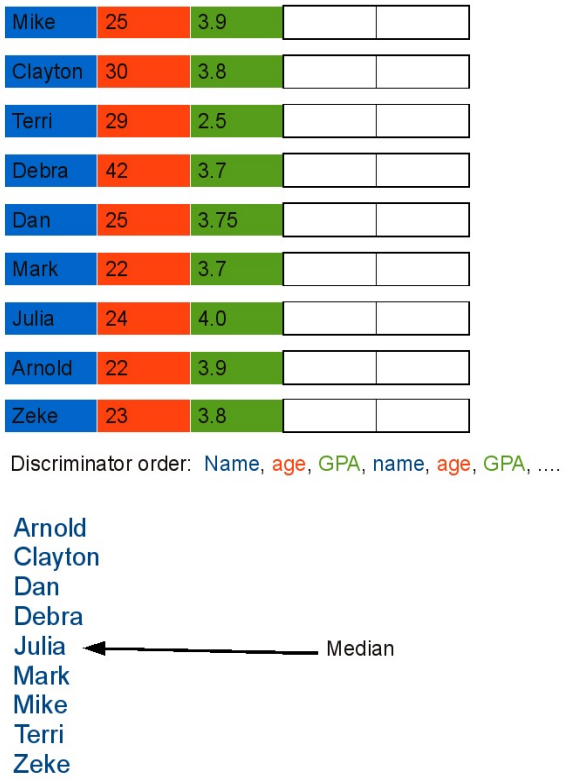


Example – using median (data stored at the leaves)





Application - using median

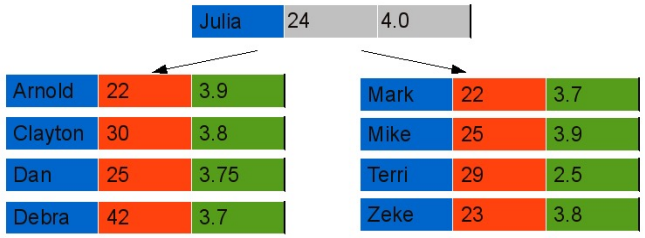


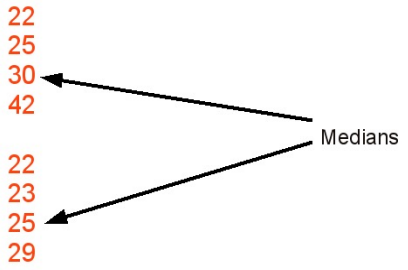
Application - using median

Arnold
Clayton
Dan
Debra
Julia ← Median
Mark
Mike
Terri
Zeke

Discriminator order: Name, age, GPA, name, age, GPA, ...

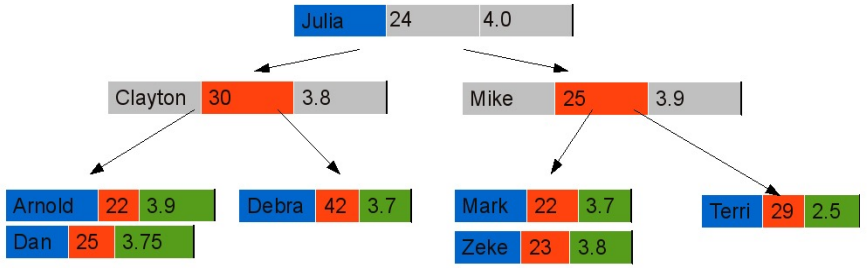
Application - using median

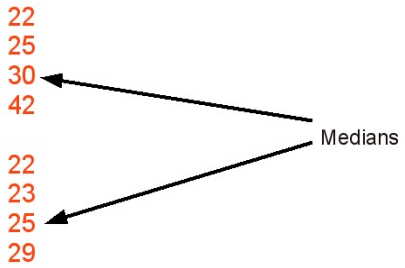




Discriminator order: Name, age, GPA, name, age, GPA, ...

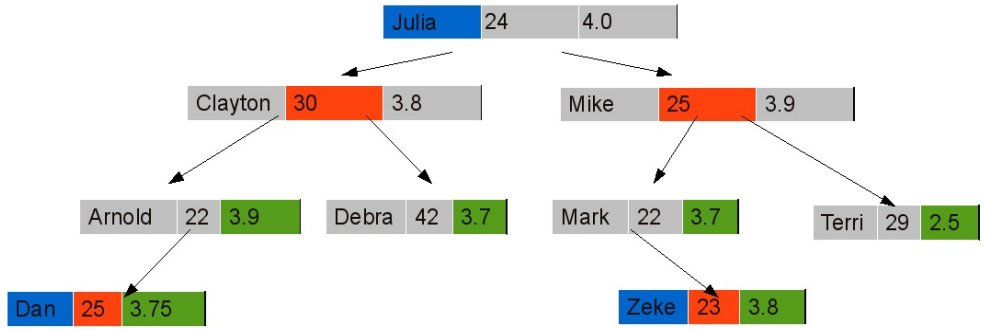
Application - using median





Discriminator order: Name, age, GPA, name, age, GPA, ...

Application - using median



Discriminator order: Name, age, GPA, name, age, GPA, ...

Bagging

- Bagging or *bootstrap aggregation* a technique for reducing the variance of an estimated prediction function.

- For classification, a *committee* of trees each
- cast a vote for the predicted class.

Bootstrap

The basic idea:

Randomly draw datasets *with replacement* from the
training data, each sample *the same size as the original training set*

Bagging

$\mathbf{Z} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$

Z^{*b} where = 1,.., B..

The prediction at input x when bootstrap sample *b* is used for training

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$

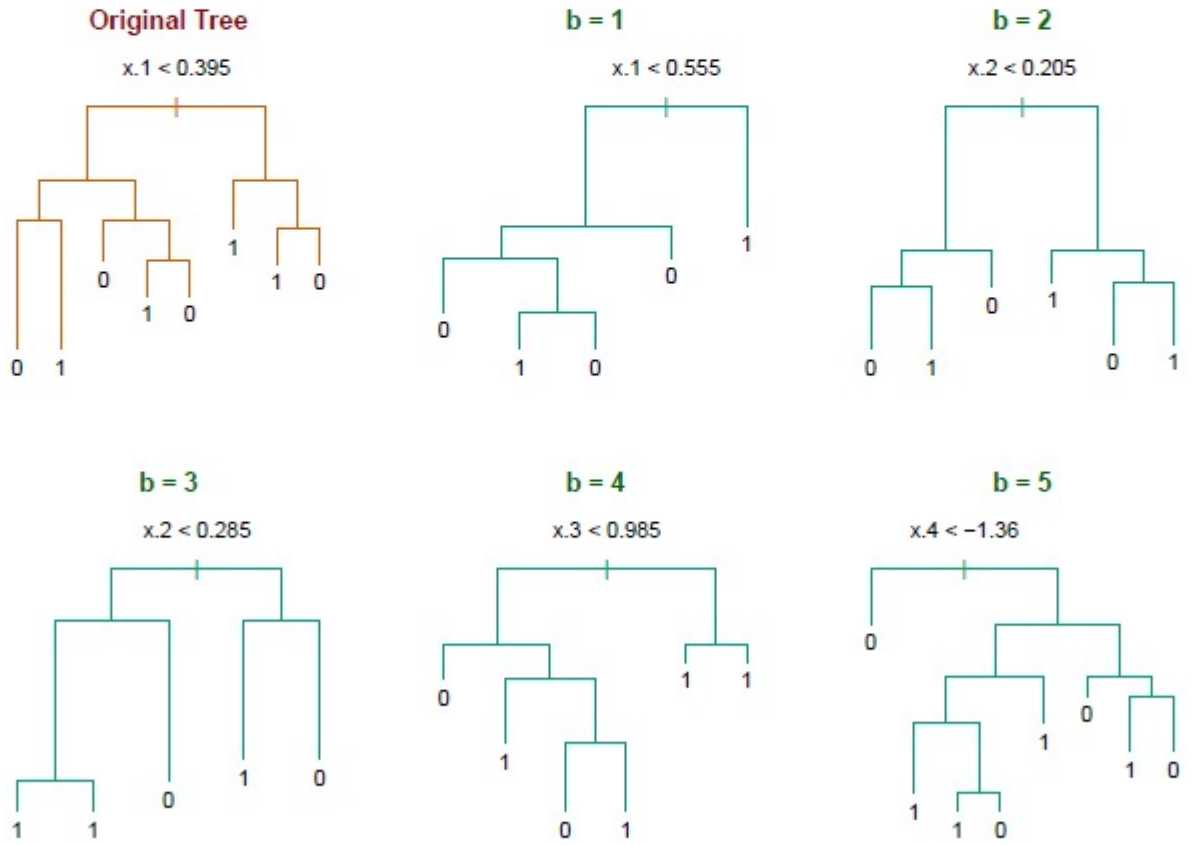
Bagging: an simulated example

- Generated a sample of size $N = 30$, with two classes and $p = 5$ features, each having a standard Gaussian distribution with pairwise Correlation 0.95.

- The response Y was generated according to
- $\Pr(Y = 1 / x1 \leq 0.5) = 0.2$,
- $\Pr(Y = 0 / x1 > 0.5) = 0.8$.

Bagging

Notice the bootstrap trees are different than the original tree

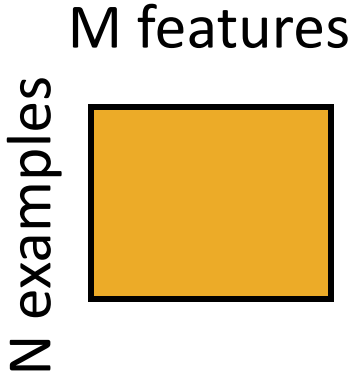


Random forest classifier

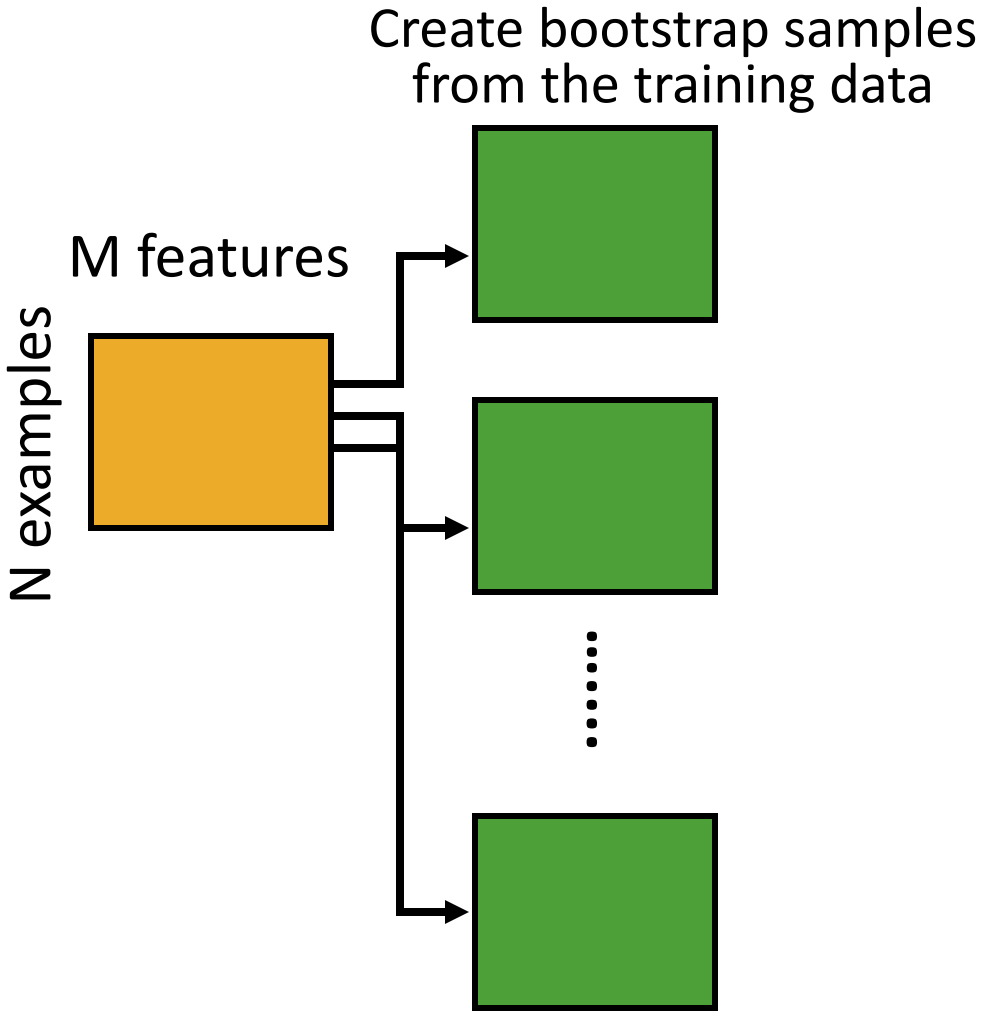
- Random forest classifier, an extension to bagging which uses *de-correlated* trees.

Random Forest Classifier

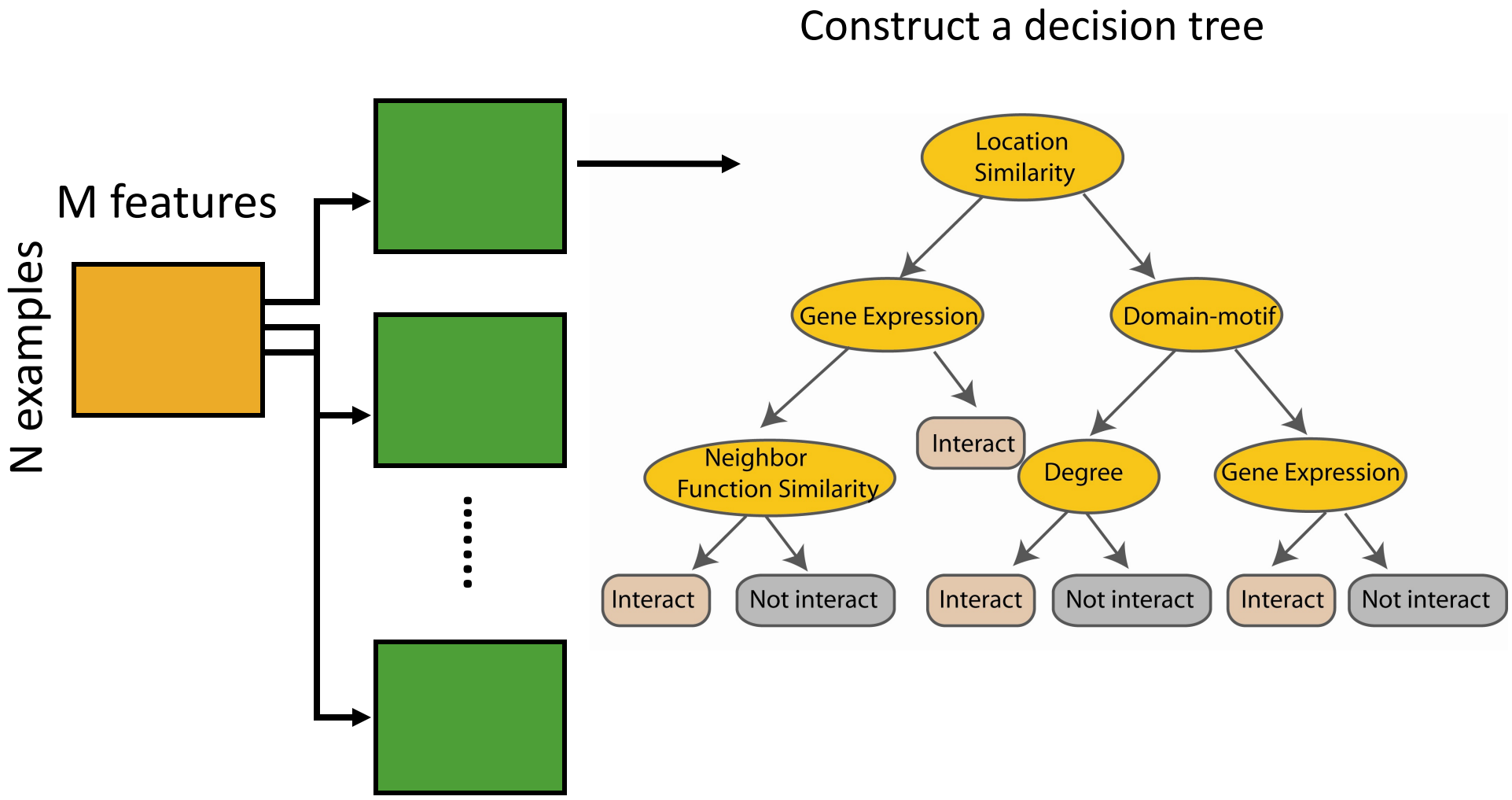
Training Data



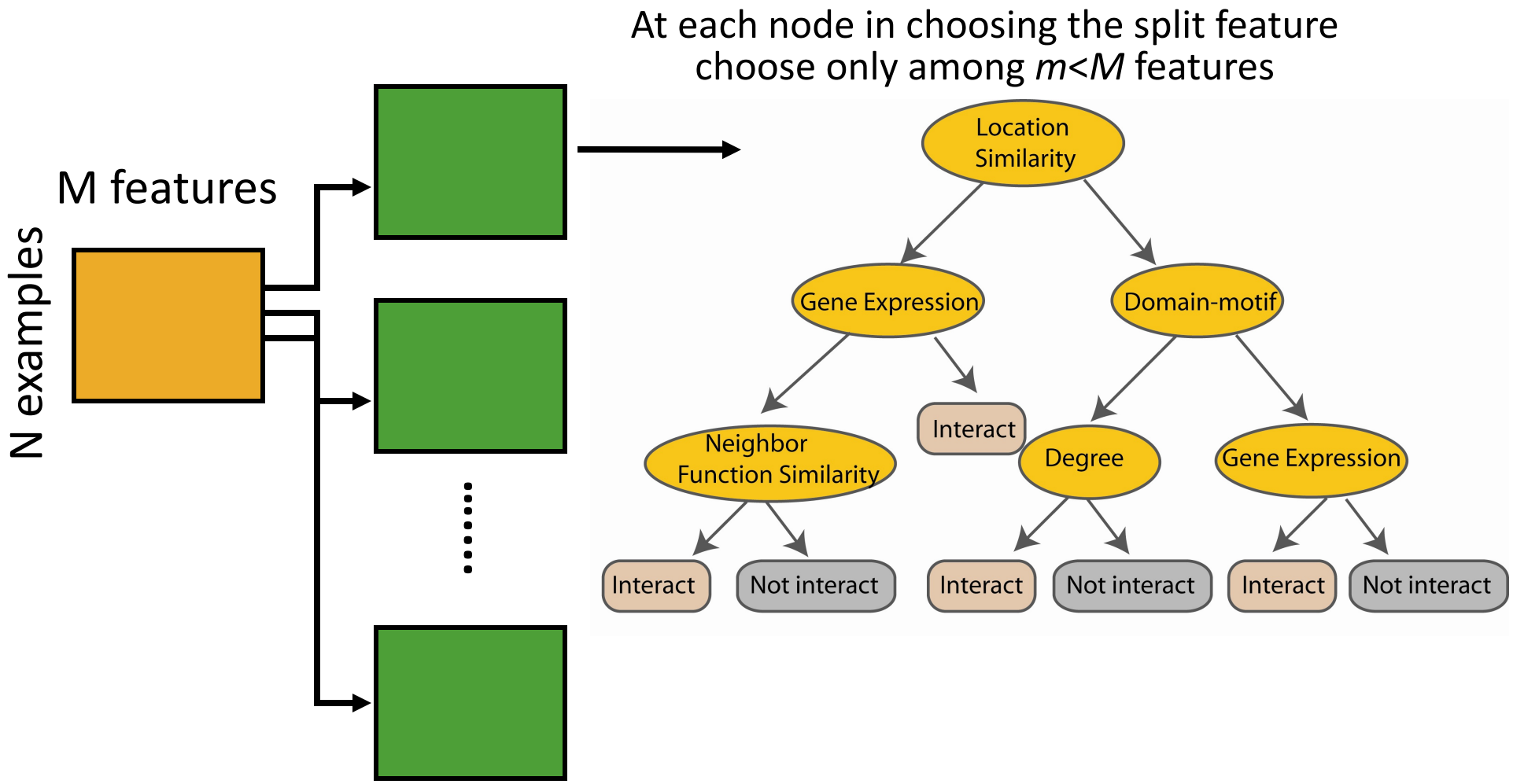
Random Forest Classifier



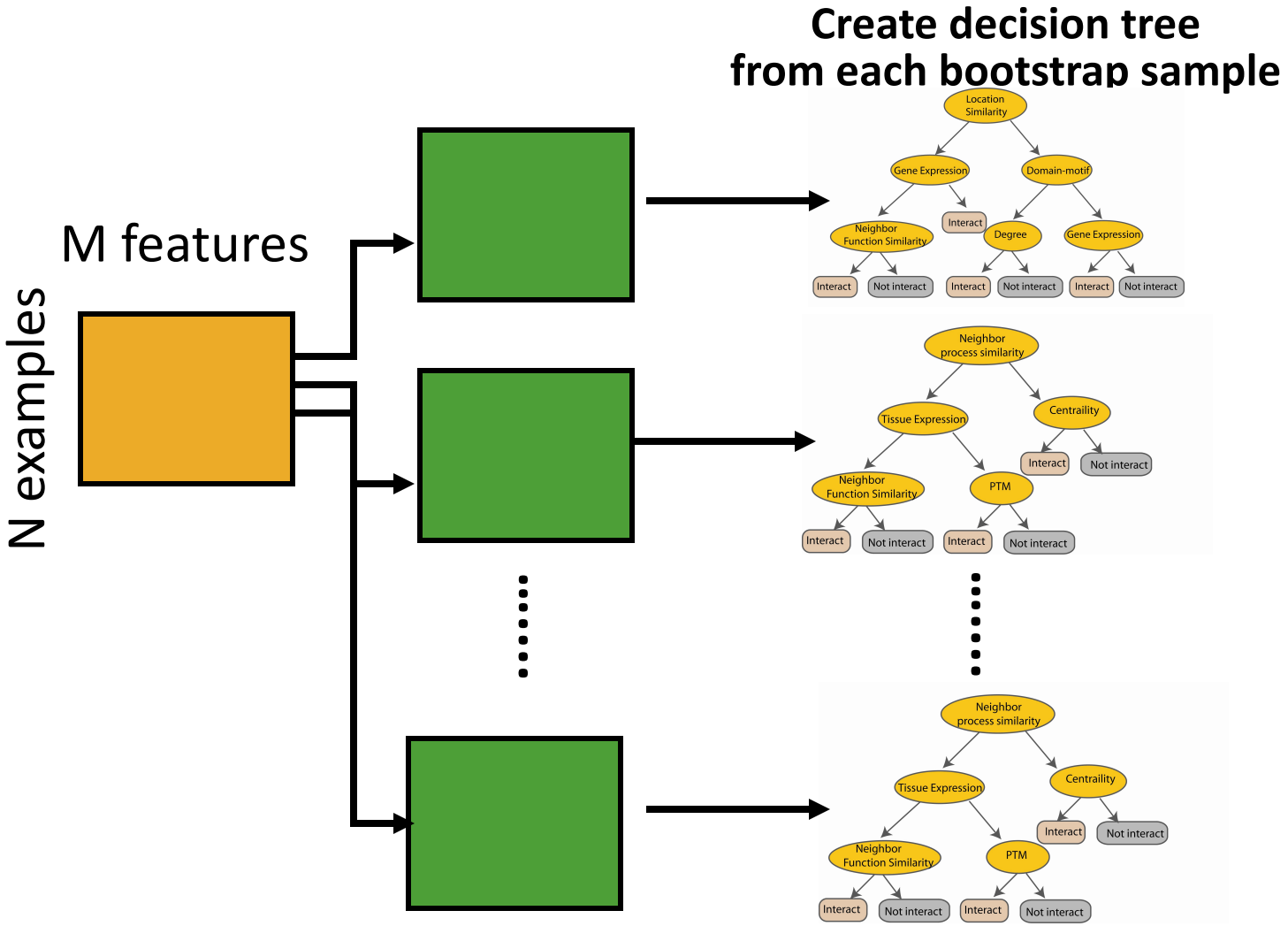
Random Forest Classifier



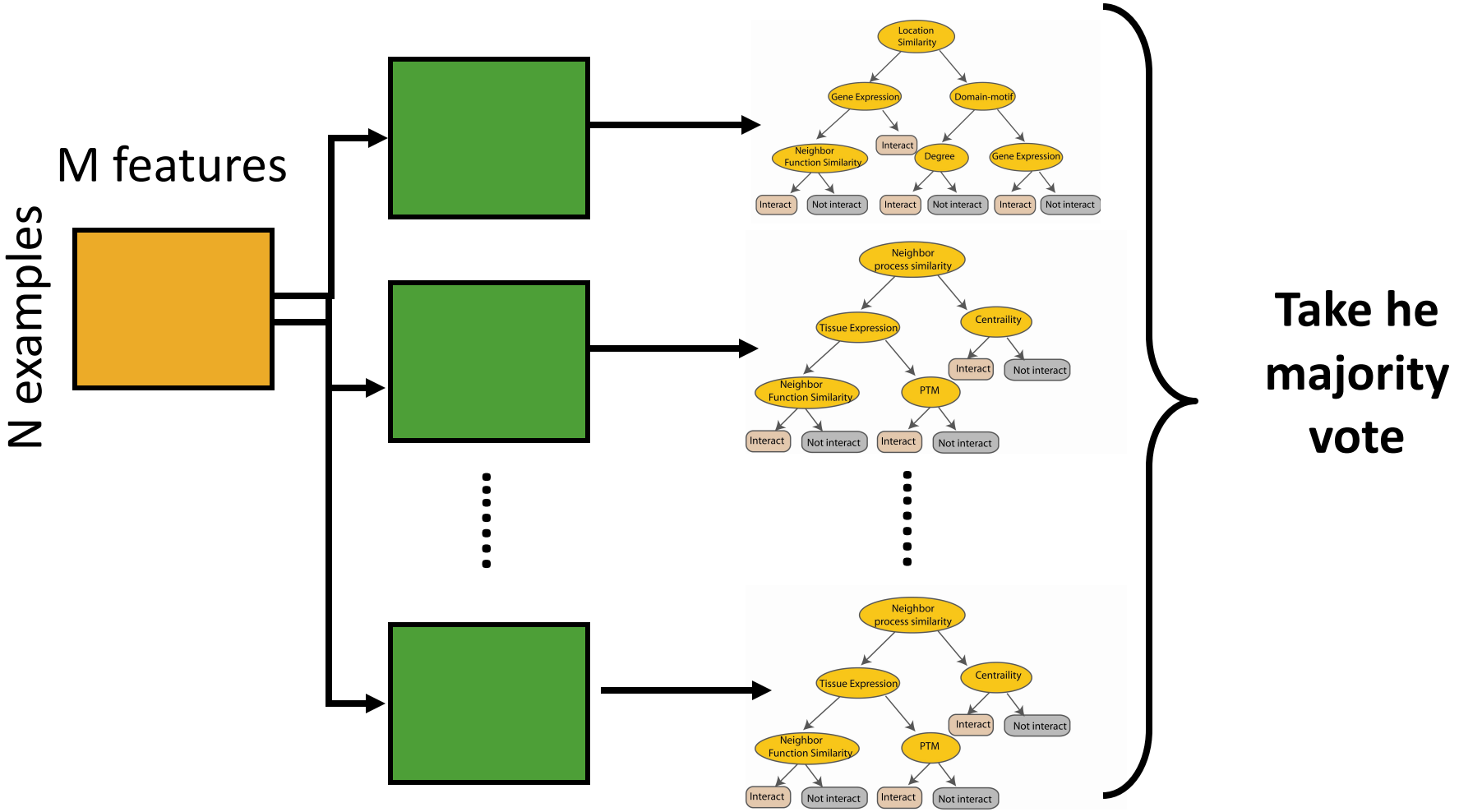
Random Forest Classifier



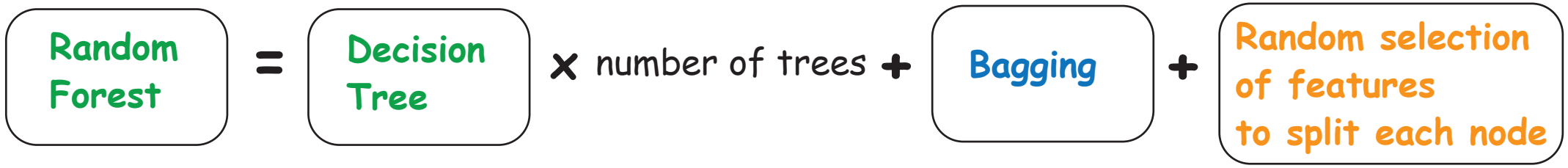
Random Forest Classifier



Random Forest Classifier



Random Forest Classifier



Random Forest Classifier

- A Random Forest is a modified form of bagging that creates ensembles of independent decision tree stumbs.

- To decorrelate the trees, we:

    1. train each tree on a separate bootstrap sample of the full training set (same as in bagging).

    2. for each tree, at each split, we ***randomly*** select a set of J predictors from the full set of predictors.

    3. From amongst the J predictors, we select the optimal predictor and the optimal corresponding threshold for the split.

Random Forest Classifier

- Why called “random” forest?
- Bagging introduces randomness into the rows of the data
- Random forest introduces randomness into the rows and columns of the data
- Combined, this provides a more diverse set of trees that almost always lowers our prediction error

Decision Trees

- Recall --
- To learn a decision tree model, we take a greedy approach:
    1. Start with an empty decision tree (undivided feature space)
    2. Choose the ‘optimal’ predictor on which to split and choose the ‘optimal’ threshold value for splitting by applying a **splitting criterion,** purity of the regions for classification.
    3. Recurse on each new node until **stopping condition** is met
    4. For classification, we label each region in the model with the label of the class to which the plurality of the points within the region belong
    5. (For regression, we predict with the average of the output values of the training points contained in the region.)

Pros and Cons of Decision Trees

Strengths

- Small trees are easy to interpret
- Trees scale well to large *N* (fast!!)
- Can handle data of all types (i.e., requires little, if any, preprocessing)
- Automatic variable selection
- Can handle missing data
- Completely nonparametric

Weaknesses

- Large trees can be difficult to interpret
- All splits depend on previous splits (i.e. capturing interactions 👍; additive models 👎)
- Trees are step functions (i.e., binary splits)
- Single trees typically have poor predictive accuracy
- Single trees have high variance (easy to overfit to training data)

Random Forest Classifier

- Recall --
- A Random Forest is a modified form of bagging that creates ensembles of independent decision tree stumps.
- To decorrelate the trees, we:

    1. train each tree on a separate bootstrap sample of the full training set (same as in bagging).
    2. for each tree, at each split, we ***randomly*** select a set of J predictors from the full set of predictors.
    3. From amongst the J predictors, we select the optimal predictor and the optimal corresponding threshold for the split.

Pros and Cons of Random Forests

Strengths

- Competitive performance.
- Remarkably good "out-of-the box" (very little tuning required).
- Built-in validation set (don't need to sacrifice data for extra validation).
- Typically does not overfit.
- Robust to outliers.
- Handles missing data (imputation not required).
- Provide automatic feature selection.
- Minimal preprocessing required.

Weaknesses

- Although accurate, often cannot compete with the accuracy of advanced boosting algorithms.
- Can become slow on large data sets.
- Less interpretable (although this is easily addressed with various tools such as variable importance, partial dependence plots, LIME, etc.).