W3. Decision Tree and Random Forest

김선
서울대학교 컴퓨터 공학부
생물정보 연구소

Contents

- Decision Trees
  - What is a Decision Tree
  - Sample Decision Trees
  - Constructing a Decision Tree
  - Splitting a Decision Tree
    - GINI index, Entropy (Information Gain)
  - Pruning a Decision Trees

- Random Forests
  - Bagging/Bootstrap
  - Constructing a Random Forest
  - Variable importance

What is a Decision Tree?

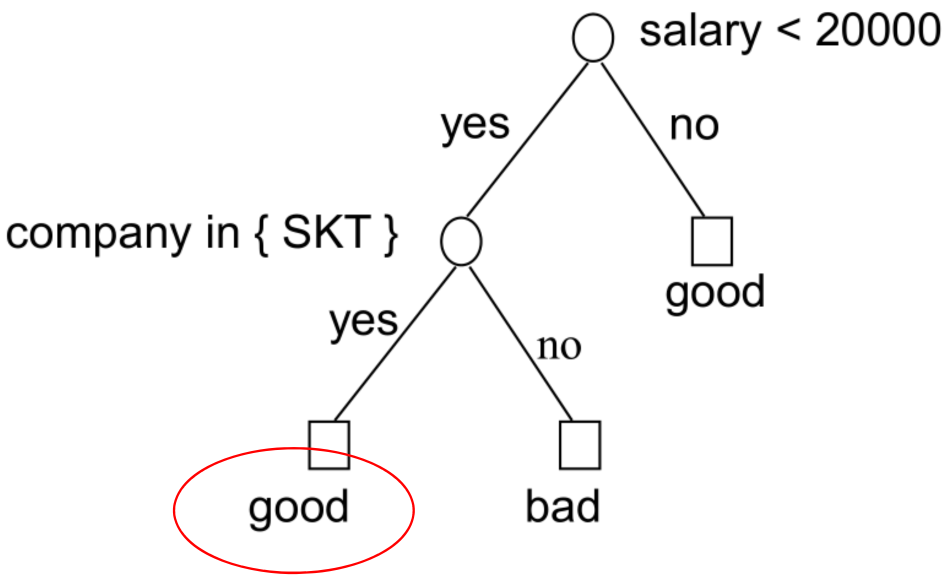
- A branch of artificial intelligence, concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data.

- As intelligence requires knowledge, it is necessary for the computers to acquire knowledge.

Decision Tree Algorithms

- The basic idea behind any decision tree algorithm is as follows:
  - Choose the *best* attribute(s) to split the remaining instances and make that attribute a decision node
  - Repeat this process for recursively for each child
  - Stop when:
    - All the instances have the same target attribute value
    - There are no more attributes
    - There are no more instances

Predicting Credit Approval Status



If Salary < 20000 and customer works for SKT what is the credit approval status of the customer?

Credit: Professor Kyuseok Shim

Inductive Learning

- In this decision tree, we made a series of Boolean decisions and followed the corresponding branch
  - Is the salary above 20K?
  - Does the customer work for SKT?

- By answering each of these yes/no questions, we then came to a conclusion on how long our commute might take

Decision Trees as Rules

- We did not have represent this tree graphically

- We could have represented as a set of rules. However, this may be much harder to read…

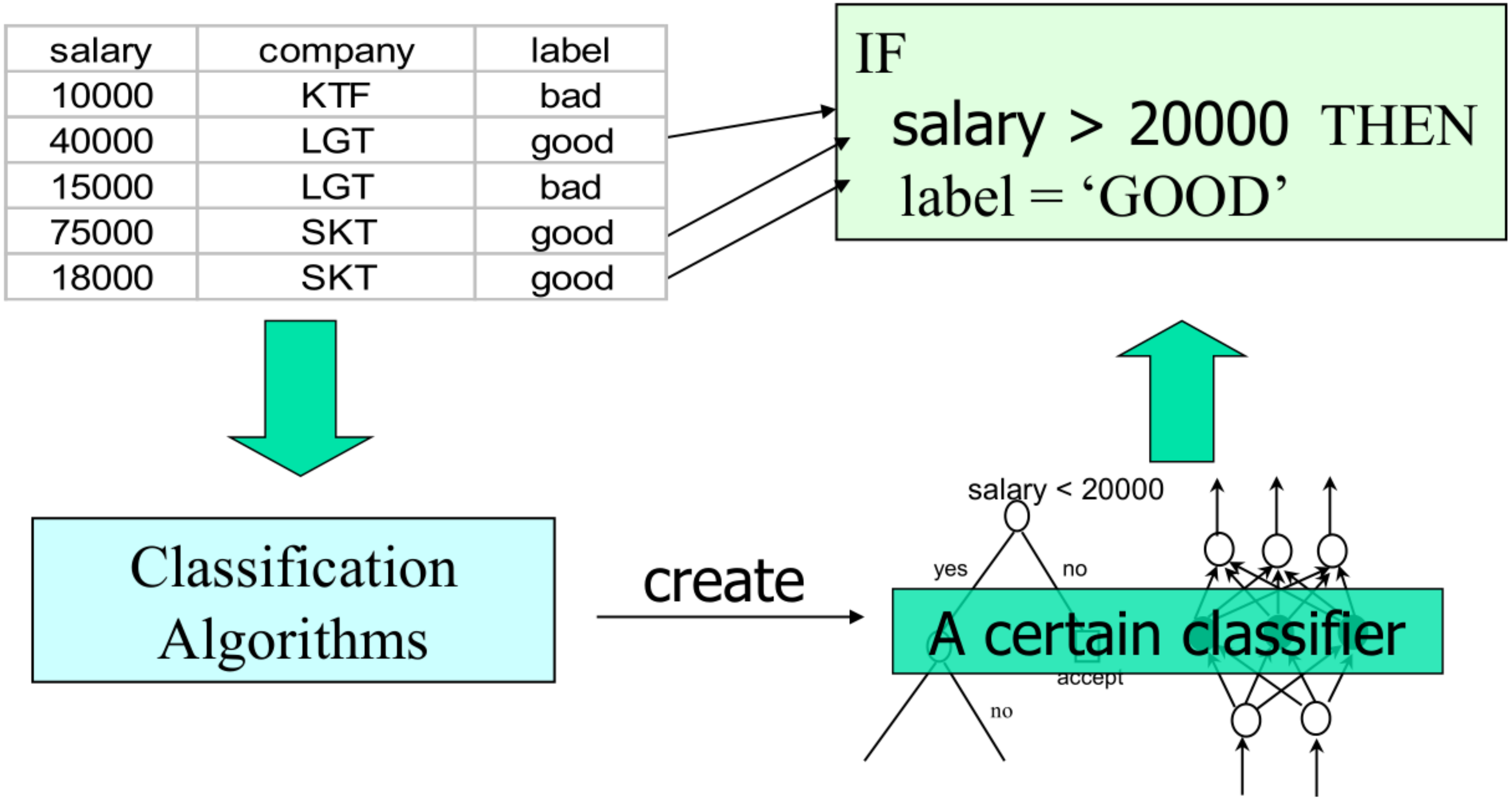
How to Create a Decision Tree

- We first make a list of attributes that we can measure
  - These attributes (for now) must be discrete
- We then choose a *target attribute* that we want to predict
- Then create an *experience table* that lists what we have seen in the past

Constructing Classifier

Database (training set)

salary	company	label
10000	KTF	bad
40000	LGT	good
15000	LGT	bad
75000	SKT	good
18000	SKT	good



Credit: Professor Kyuseok Shim

Prediction

Input Data

salary	company	label
40000	SKT	?
10000	KTF	?

good

bad

Classifier (rules)

IF
salary > 20000 THEN
label = 'GOOD'

Satisfy?

Credit: Professor Kyuseok Shim

Pros And Cons of Decision Tree

- Pros
  - Fast execution time
  - Generated rules are easy to interpret by humans
    - c.f. neural networks
  - Scale well for large data sets
  - Can handle high dimensional data (i.e. many columns)
- Cons
  - Cannot capture correlations among attributes
  - Consider only axis-parallel cuts

Decision Tree Algorithm

- A decision tree is created in two phases:
  - Building Phase
    - Recursively split nodes using best splitting attribute for node until all the examples in each node belong to one class
  - Pruning Phase
    - Prune leaf nodes recursively to prevent overfitting
    - Smaller imperfect decision tree generally achieves better accuracy

Building Phase

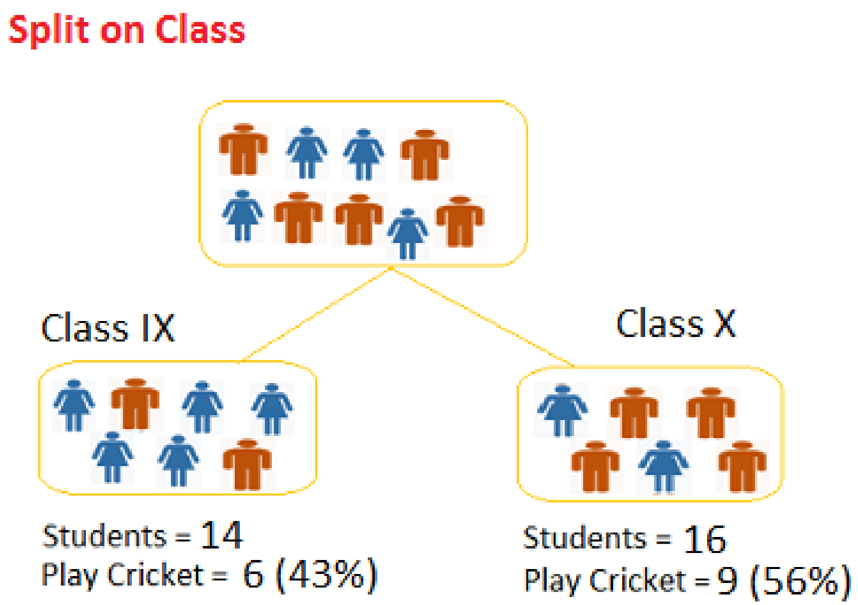
- General tree-growth algorithm (binary tree) Partition(Data S)
  - If (all points in S are of the same class) then return;
  - for each attribute A do
    - evaluate splits on attribute A;
  - Use best split to partition S into S1 and S2;
  - Partition(S1);
  - Partition(S2);

사용 사례

- Credit risk를 measure 함 – 신용카드 회사에서 신용카드 발급을 위하여 새로운 신청자에 대한 카드 발급 결정
  - 4000만 명의 신용카드 고객이 있는 미국의 어느 은행에서는 새로운 신용 카드 고객의 과거 사실로 부터 가장 이익을 많이 내 줄 수 있는 고객과 또 손 해를 끼칠 수 있는 고객을 찾아내 회사의 이익을 증대시켰다고 함.

Example Decision Tree

- Segregate the students based on target variable (playing cricket or not) with attributes "Gender" and "Class"
- The decision tree below has been split by Class.



How can we get best split?

- Select the attribute that is most useful for classifying training set
- GINI index and entropy
  - Statistical properties
  - Measure how well an attribute separates the training set
  - $Entropy(T) = -\sum p_j \times log_2(p_j)$
  - $GINI\ Index(T) = 1 - \sum p_j^2$

Performing the split

- Each attribute list will be partitioned into two lists, one for each child

- Splitting attribute
  - Scan the attribute list, apply the split test, and move records to one of the two new lists

- Non-splitting attribute
  - Cannot apply the split test on non-splitting attributes
  - Use rid to split attribute lists

GINI index (1)

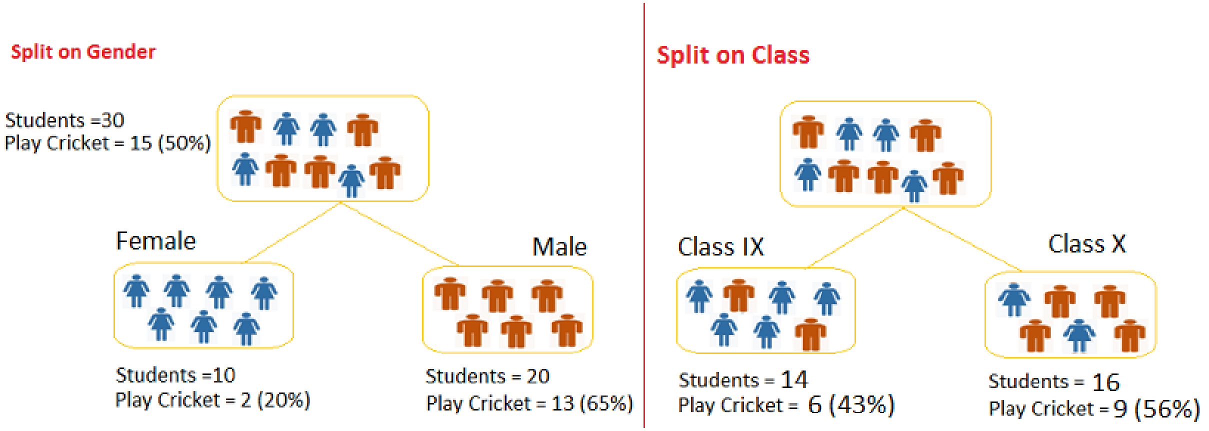
- GINI Index for a given node T:

$$GINI\ Index(T) = 1 - \sum p_j^2$$

  - where j is the class label
  - Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

Computing GINI index

$$GINI\ Index(T) = 1 - \sum p_j^2$$



Split on Gender		
GINI(Female)	$1 - 0.2^2 - 0.8^2$	=0.35
GINI(Male)	$1 - 0.65^2 - 0.35^2$	=0.455

Split on Class		
GINI(Class IX)	$1 - 0.43^2 - 0.57^2$	=0.49
GINI(Class X)	$1 - 0.56^2 - 0.44^2$	=0.5264

Splitting based on GINI

- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

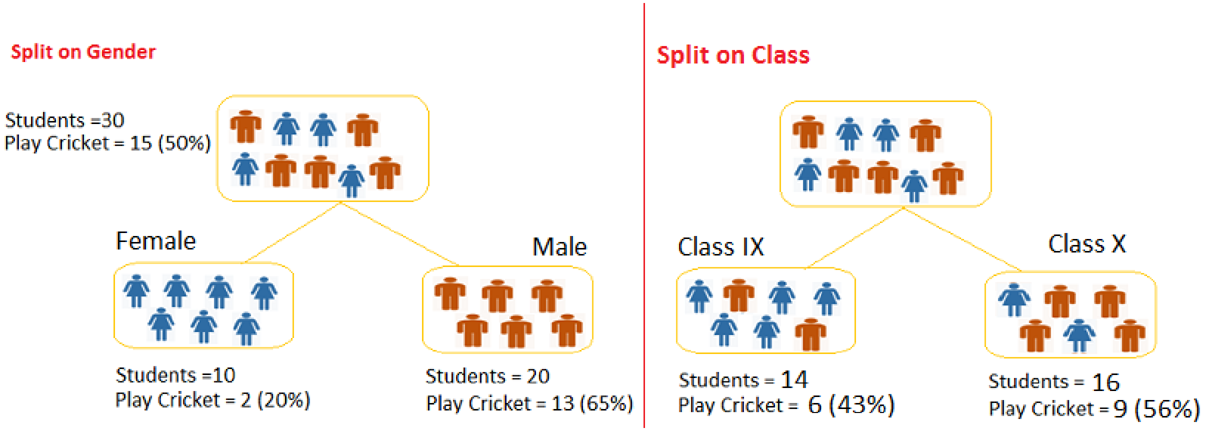
where, n_i = number of records at child i,

n = number of records at node p.

- The lower the $GINI_{split}$ the better the split
  - Minimum $GINI_{split}$=0

Computing GINI index

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$



Split on Gender		
GINI(Female)	$1 - 0.2^2 - 0.8^2$	=0.32
GINI(Male)	$1 - 0.65^2 - 0.35^2$	=0.455
GINISplit	**(10/30)*0.32+(20/30)*0.455**	**=0.41**

Split on Class		
GINI(Class IX)	$1 - 0.43^2 - 0.57^2$	=0.49
GINI(Class X)	$1 - 0.56^2 - 0.44^2$	=0.5264
GINISplit	**(14/30)*0.49+(16/30)*0.5264**	**=0.514333**

Gender wins the split so it should be placed on the root

Entropy (INFO)

- Entropy at a given node T:

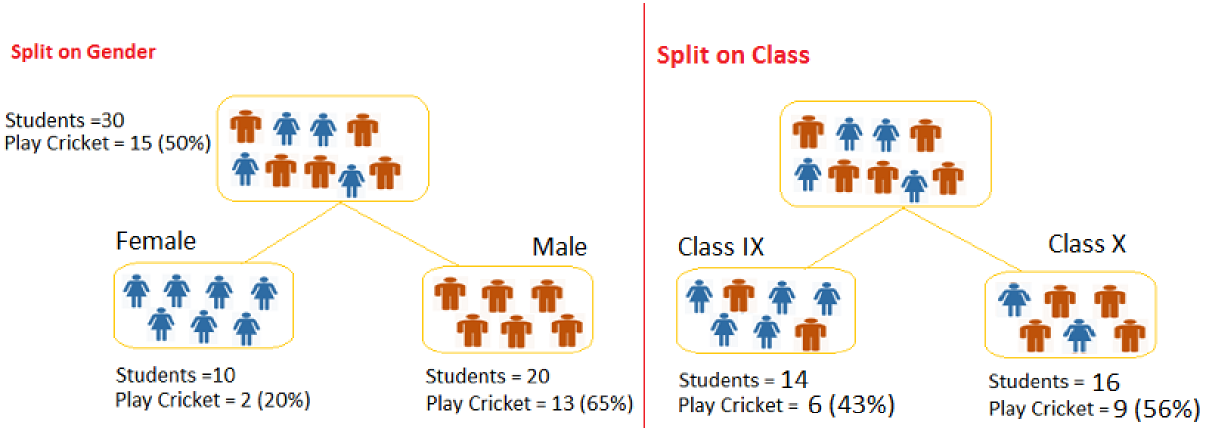
$$Entropy(T) = -\sum p_j \times log_2(p_j)$$

(NOTE: j is the relative frequency of class j at node T.

- Measures homogeneity of a node.
  - Maximum (log n_c) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information
- Entropy based computations are similar to the GINI index computations

Computing GINI index

$$Entropy(T) = -\sum p_j \times log_2(p_j)$$



Split on Gender		
Entropy(Parent)	-(15/30) log2 (15/30) – (15/30) log2 (15/30)	=1
Entropy(Female)	-(2/10) log2 (2/10) – (8/10) log2 (8/10)	=0.72
Entropy(Male)	-(13/20) log2 (13/20) – (7/20) log2 (7/20)	=0.93

Split on Class		
Entropy(Parent)	-(15/30) log2 (15/30) – (15/30) log2 (15/30)	=1
Entropy(Class IX)	-(6/14) log2 (6/14) – (8/14) log2 (8/14)	=0.99
Entropy(Class X)	-(9/16) log2 (9/16) – (7/16) log2 (7/16)	=0.99

Splitting based on Entropy

- Information Gain

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;
n_i is number of records in partition i

- Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3 and C4.5
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Splitting based on Entropy

- Gain ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \qquad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions

n_i is the number of records in partition i

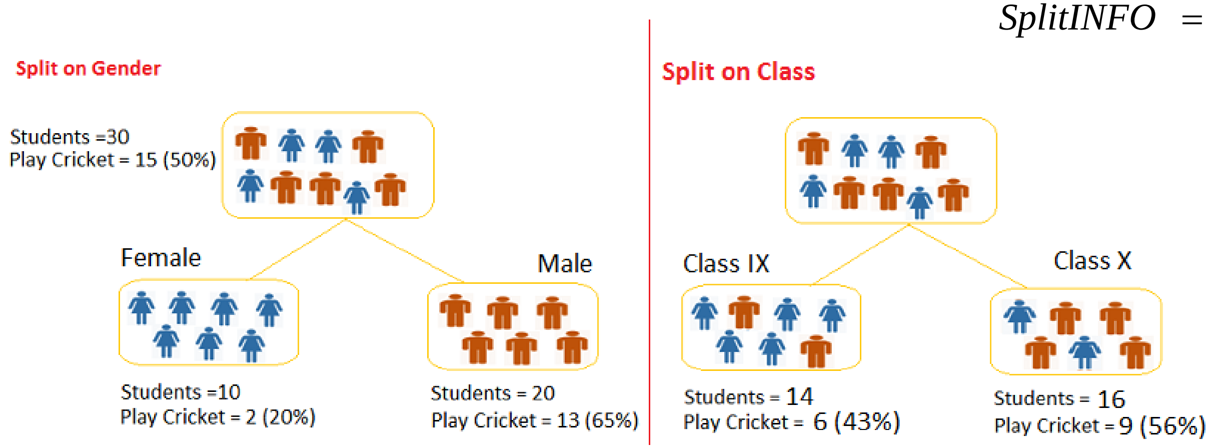
- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain
- The higher the GainRATIO the better the split

Computing GINI index

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$



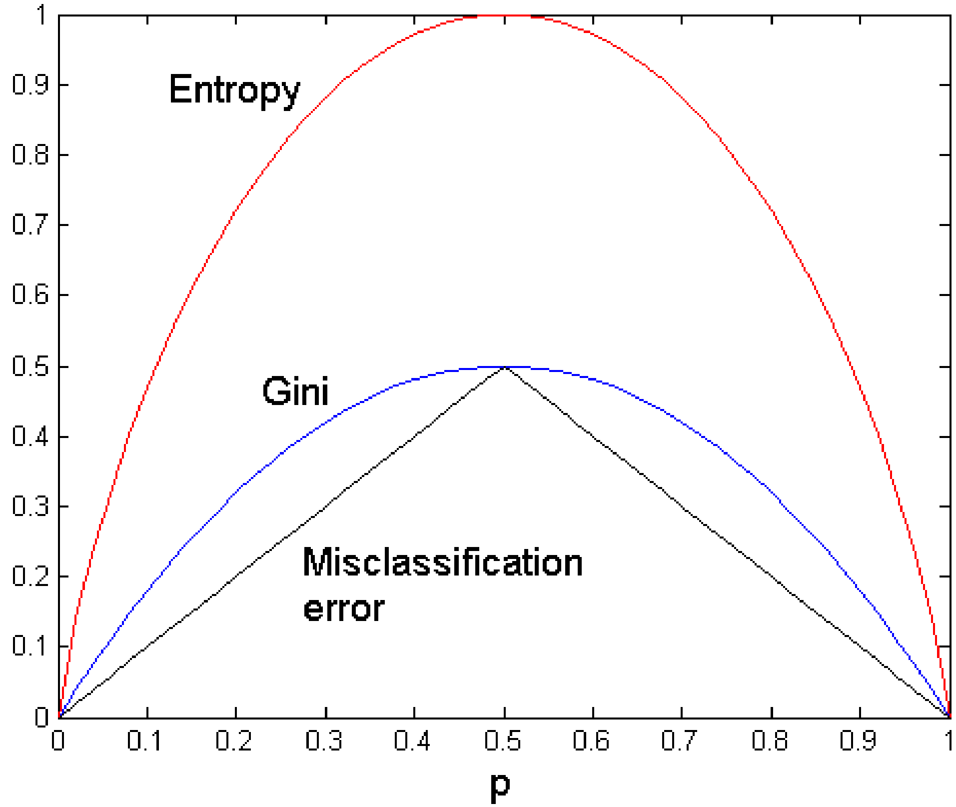
Split on Gender		
Entropy(Parent)	-(15/30) log2 (15/30) – (15/30) log2 (15/30)	=1
Entropy(Female)	-(2/10) log2 (2/10) – (8/10) log2 (8/10)	=0.72
Entropy(Male)	-(13/20) log2 (13/20) – (7/20) log2 (7/20)	=0.93
Gain(Gender)	1-((10/30*0.72)+(20/30)*0.93)	=0.14
SplitINFO(Gender)	-(10/30*log(10/30)+20/30*log(20/30))	=0.9183
GainRATIO(Gender)	0.14/0.9183	=**0.1525**

Split on Class		
Entropy(Parent)	-(15/30) log2 (15/30) – (15/30) log2 (15/30)	=1
Entropy(Class IX)	-(6/14) log2 (6/14) – (8/14) log2 (8/14)	=0.99
Entropy(Class X)	-(9/16) log2 (9/16) – (7/16) log2 (7/16)	=0.99
Gain(Class)	1-((14/30)*0.99+(16/30)*0.99))	=0.01
SplitINFO(Class)	-(14/30*log(14/30)+16/30*log(16/30))	=0.9968
GainRATIO(Class)	0.01/0.9968	=0.01

Gender wins the split so it should be placed on the root

Comparison among Splitting Criteria

- For a 2-class problem

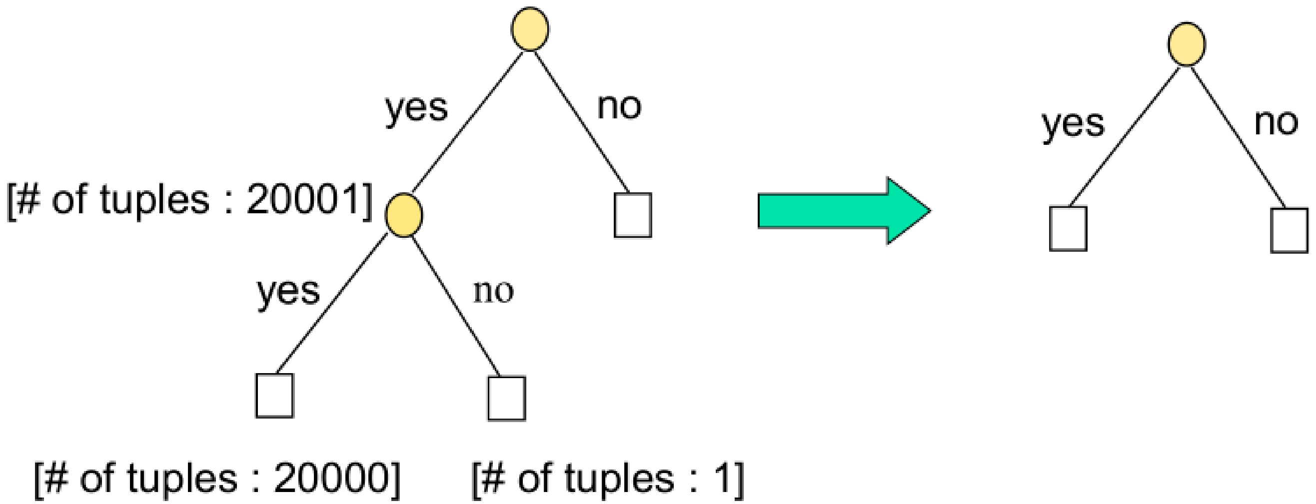


Avoiding overfitting by Pruning

- Problem: Overfitting
  - Overfitting results in decision trees that are more complex than necessary
  - Many branches of the decision tree will reflect anomalies in the training data due to noise or outliers
  - Poor accuracy for unseen samples
- Smaller imperfect decision tree generally achieves better accuracy
- Prune leaf nodes recursively to prevent overfitting
- Two types of pruning:
  - Pre-pruning (forward pruning)
  - Post-pruning (backward pruning)

Pruning example

- Smaller imperfect decision tree generally achieves better accuracy
- Prune leaf nodes recursively to prevent overfitting



Credit: Professor Kyuseok Shim

Pre-pruning

- In pre-pruning, we decide during the building process when to stop adding attributes (possibly based on their information gain)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same

- However, this may be problematic – Why?
  - Sometimes attributes individually do not contribute much to a decision, but combined, they may have a significant impact

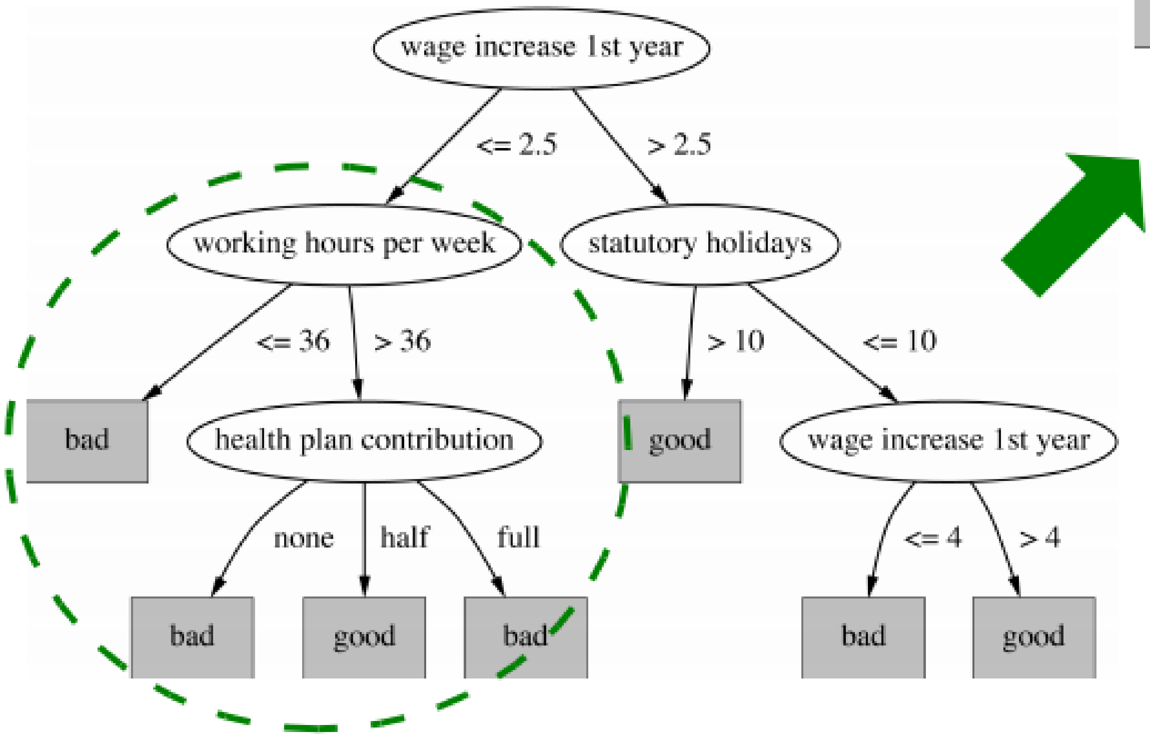
Post-pruning

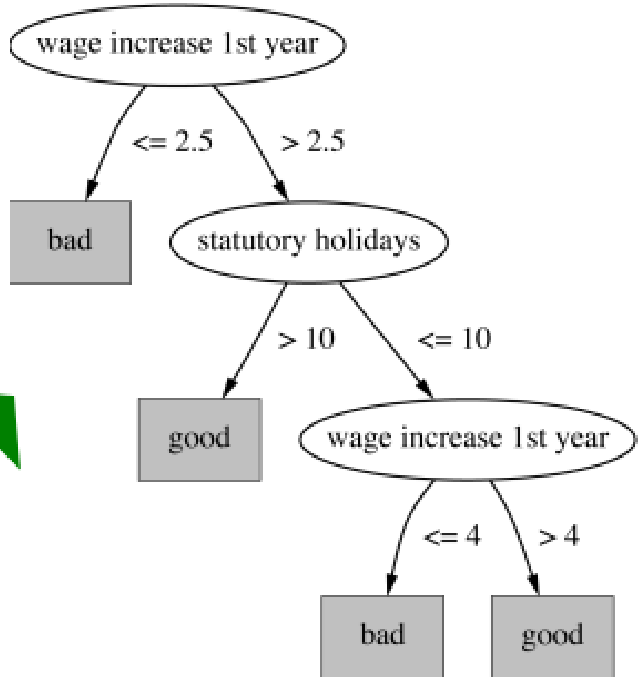
- Post-pruning waits until the full decision tree has built and then prunes the attributes
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree
  - Can use MDL for post-pruning

- Two techniques:
  - Subtree Replacement
  - Subtree Raising

Subtree replacement

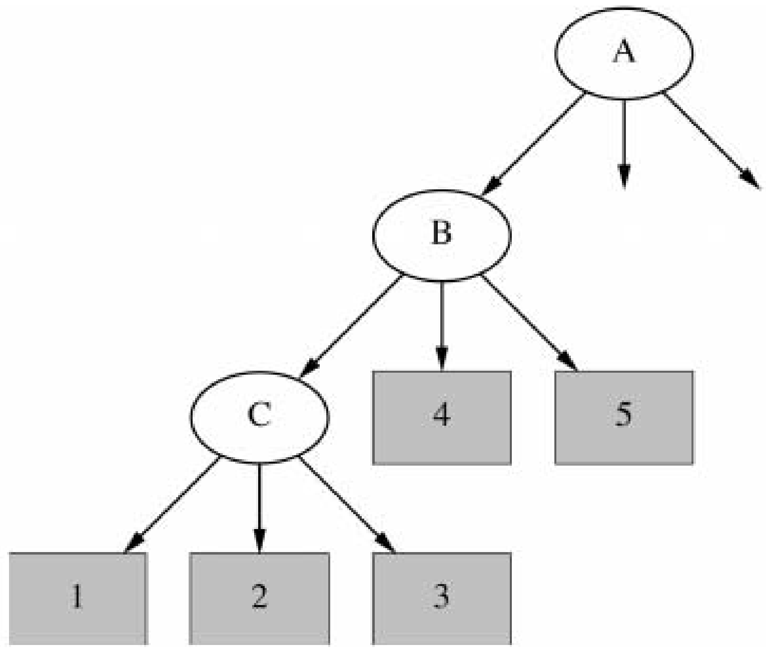
- Bottom-up
- Consider replacing a tree only after considering all its subtrees



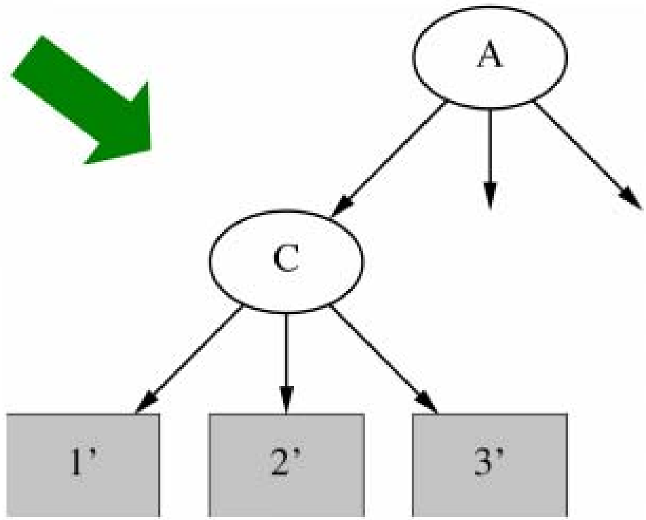


Credit: J. Furnkranz

Subtree replacement



- Delete node B
- Redistribute instances of leaves 4 and 5 into C



Credit: J. Furnkranz

Problems with Decision Trees

- While decision trees classify quickly, the time for building a tree may be higher than another type of classifier

- Decision trees suffer from a problem of errors propagating throughout a tree
  - A very serious problem as the number of classes increases

Error Propagation

- Since decision trees work by a series of local decisions, what happens when one of these local decisions is wrong?
  - Every decision from that point on may be wrong
  - We may never return to the correct path of the tree

Bagging

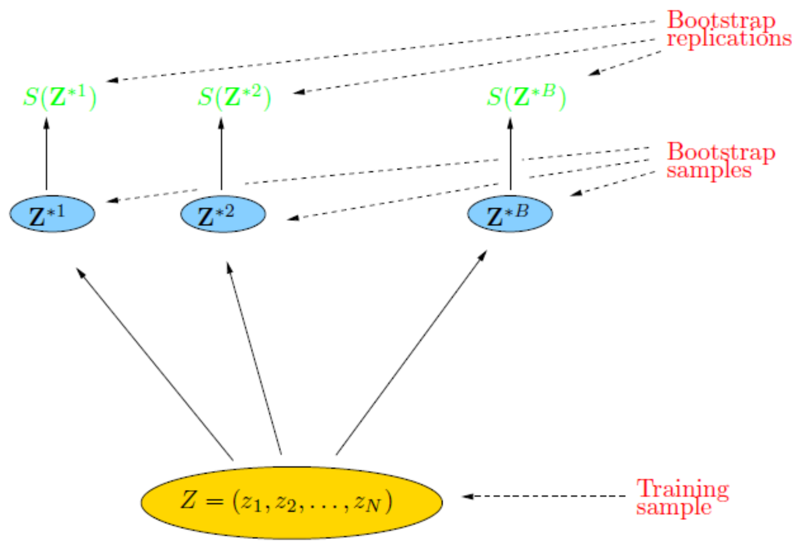
- Bagging or *bootstrap aggregation* a technique for reducing the variance of an estimated prediction function.

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

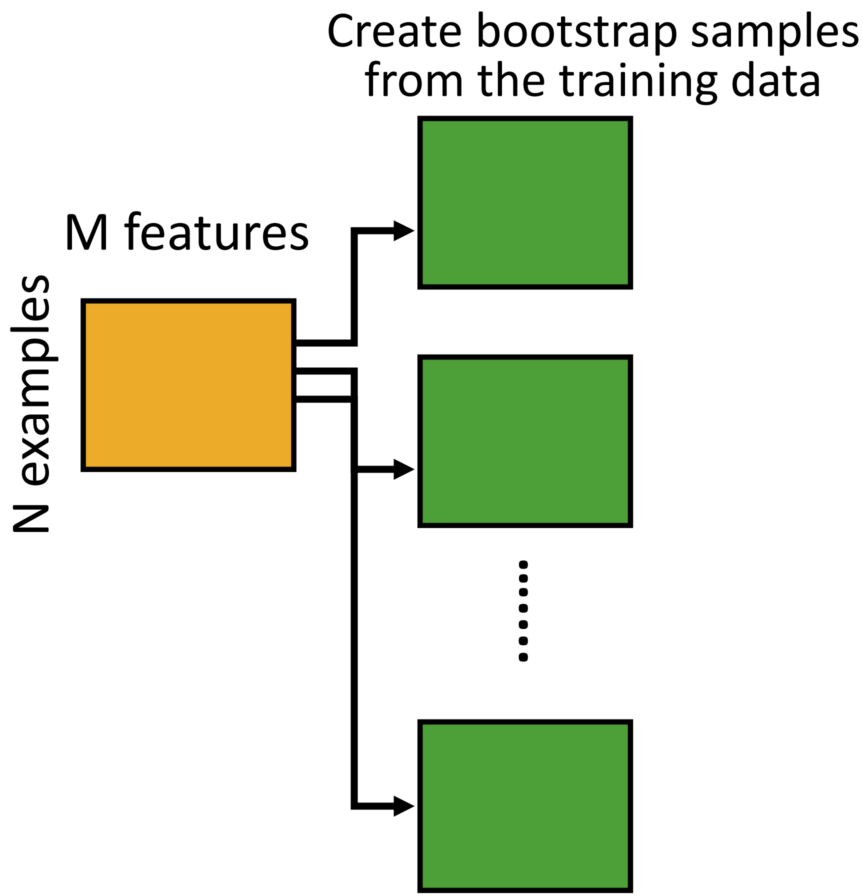
Credit: Oznur Tastan

Bootstrap

- The basic idea:
  - randomly draw datasets *with replacement* from the
  - training data, each sample *the same size as the original training set*

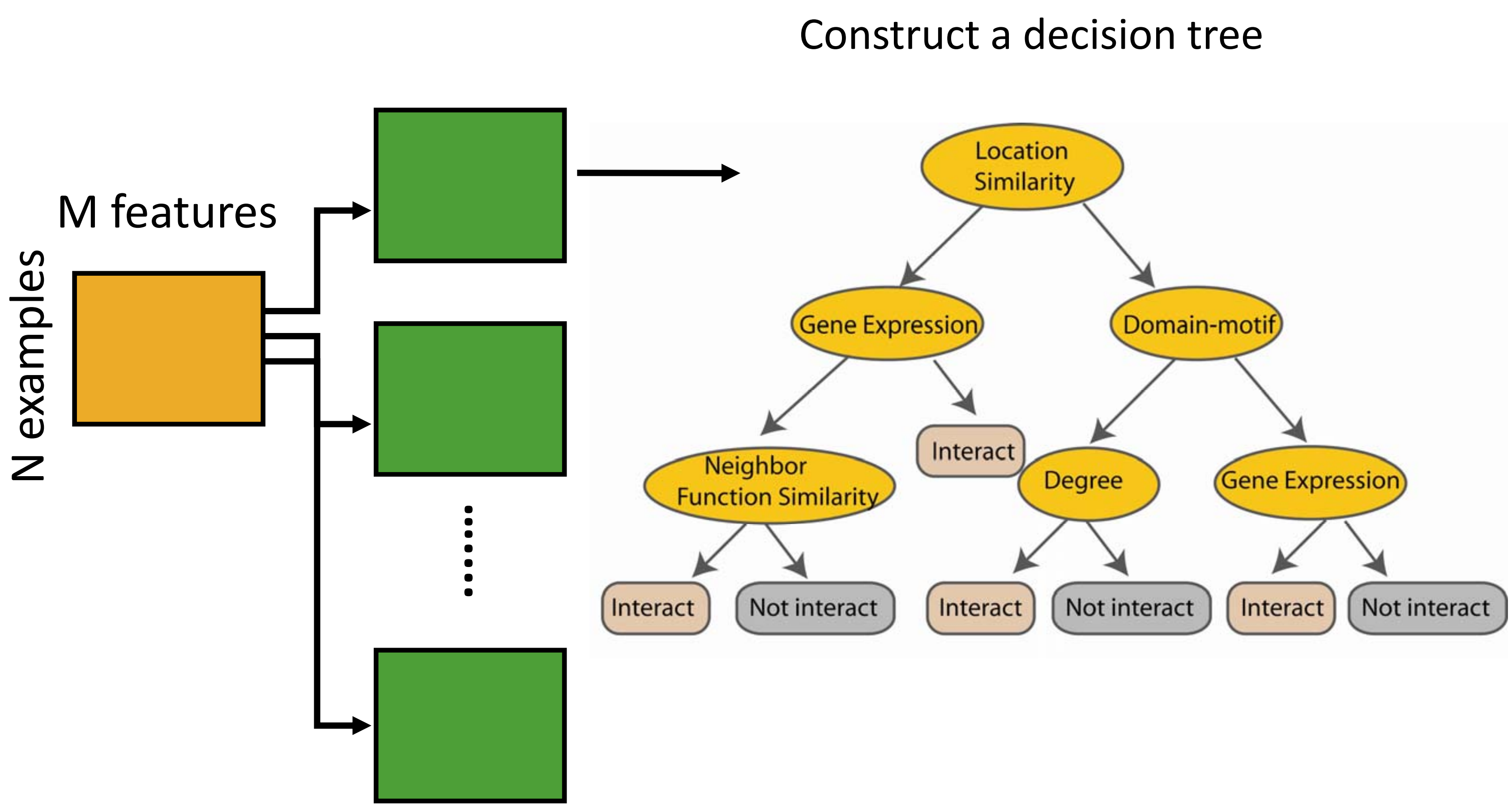


Bagging

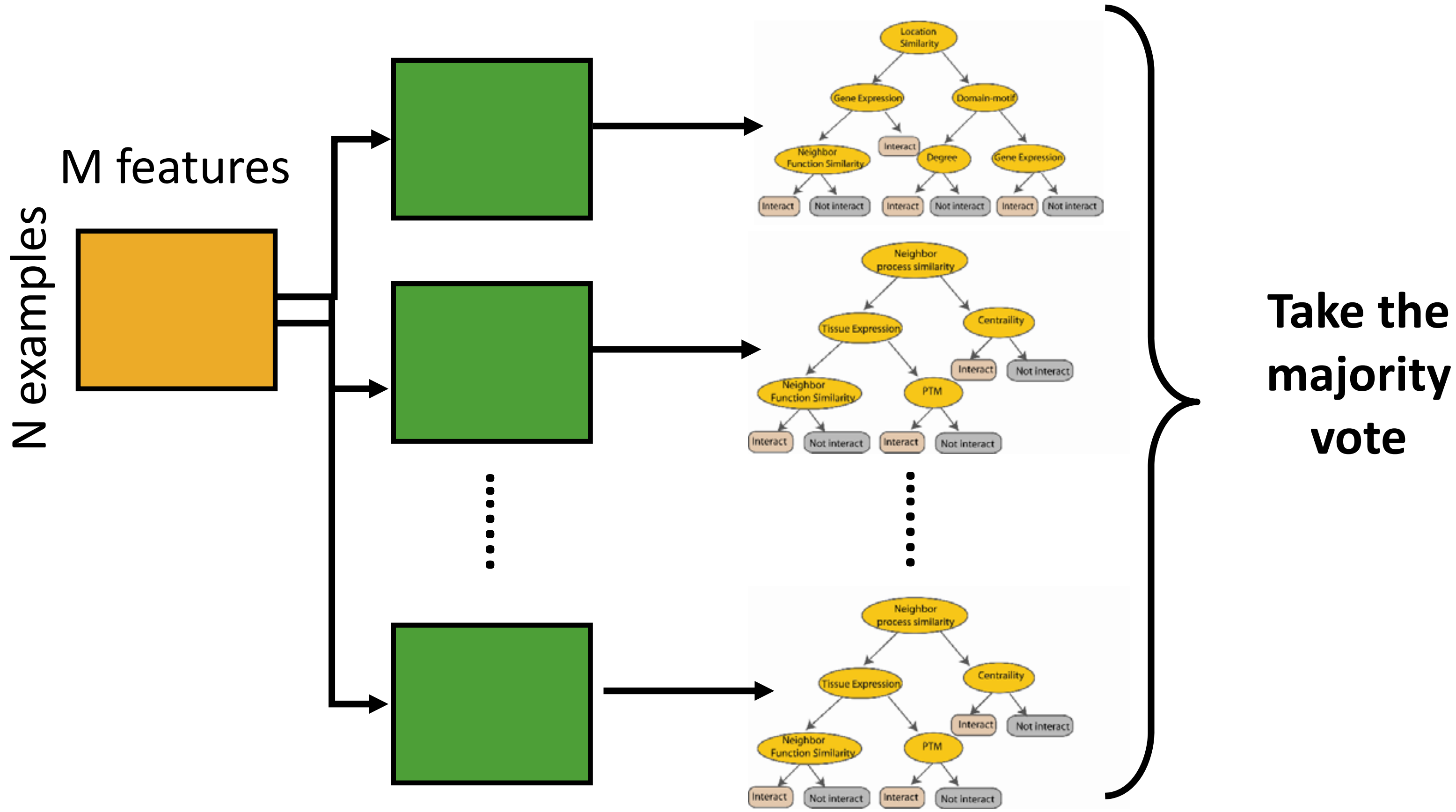


Create bootstrap samples
from the training data
M features
N examples

Random Forest Classifier



Random Forest Classifier



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

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

http://www-stat.stanford.edu/~hastie/Papers/ESLII.pdf (Chapter 8.7)

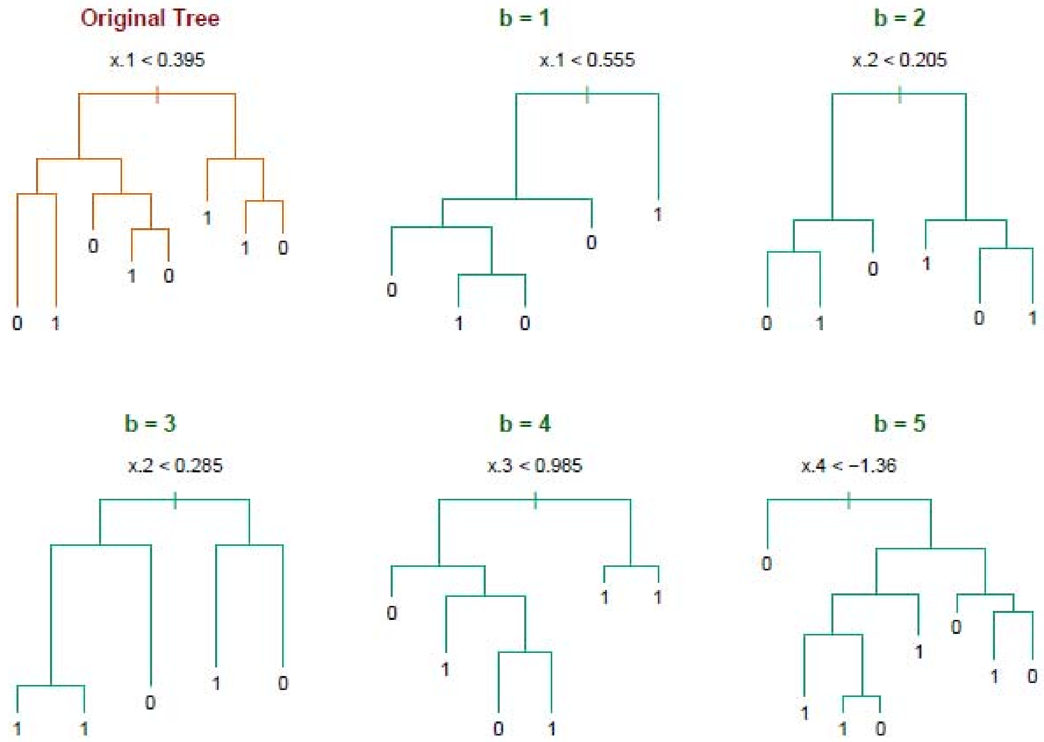
Bagging : an simulated example

- 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

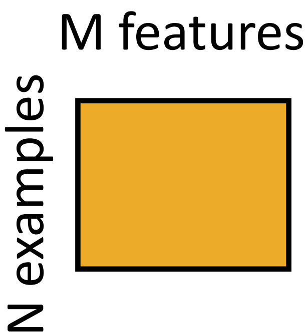


Random forest classifier

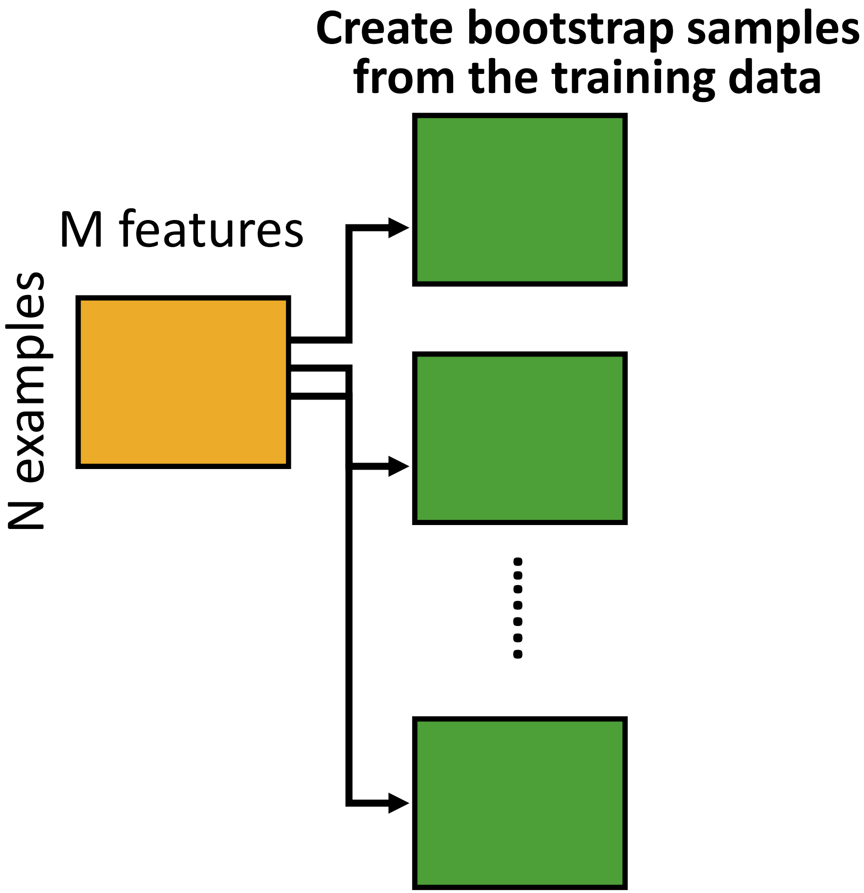
- Random forest classifier, an extension to bagging which uses *de-correlated* trees.

Random Forest Classifier

Training Data

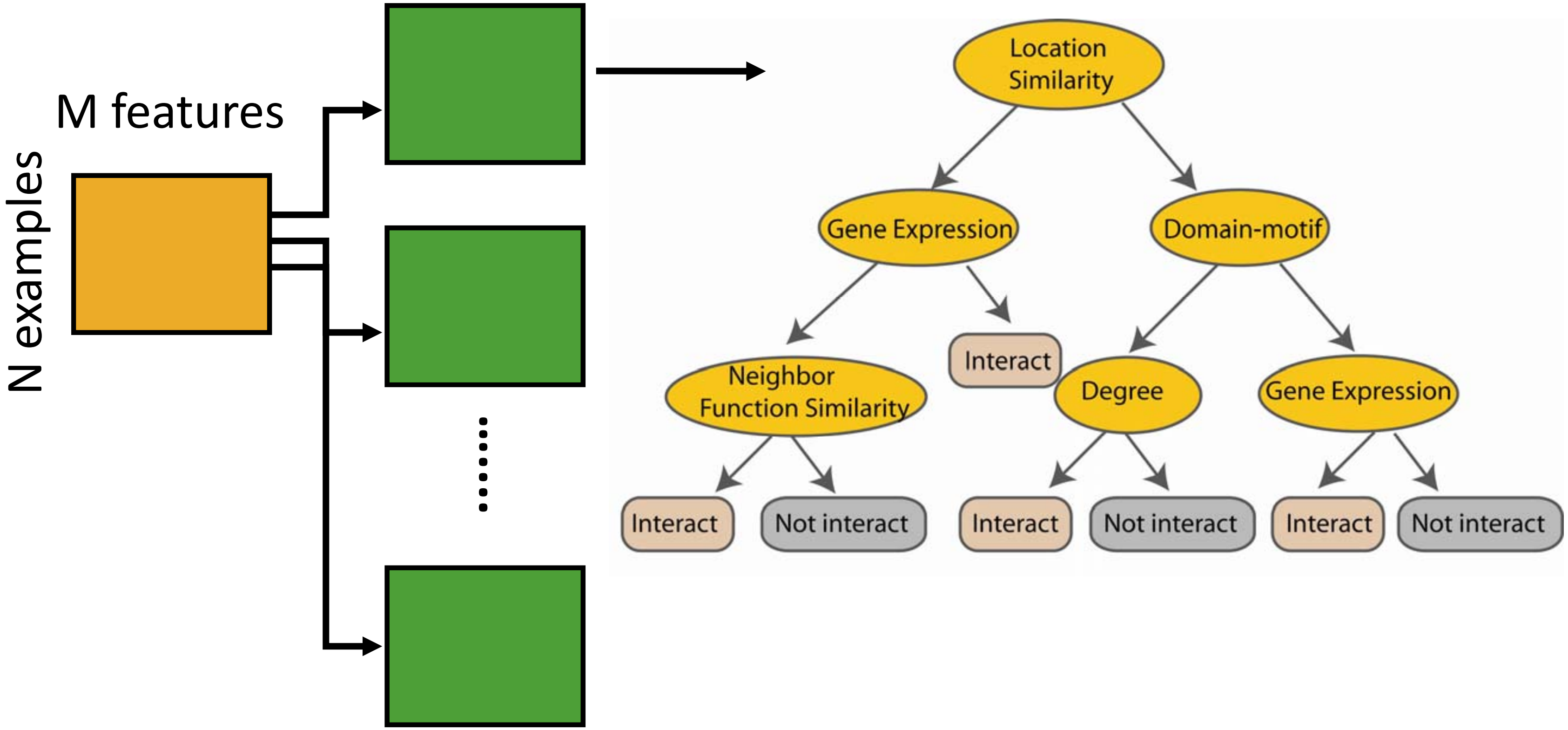


Random Forest Classifier



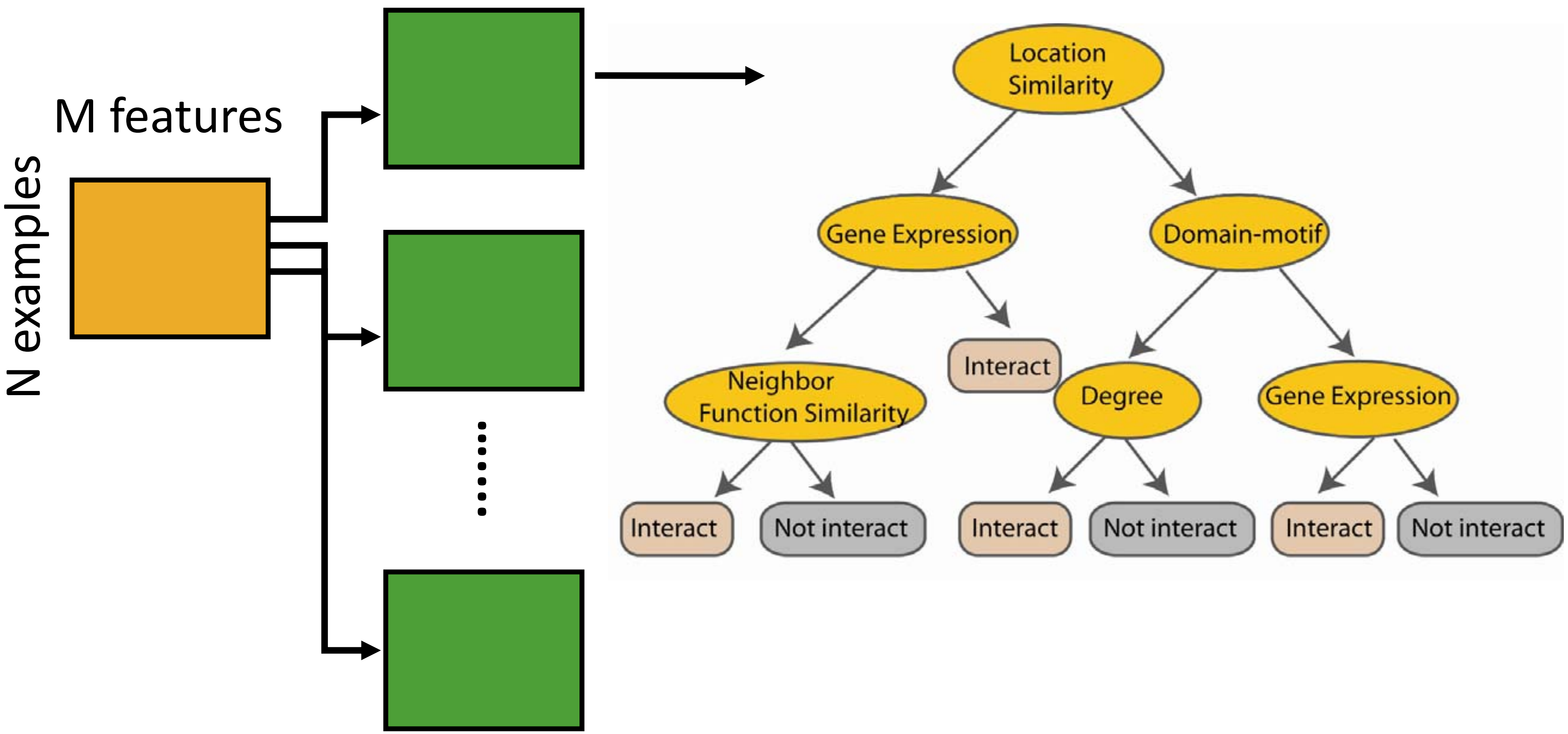
Random Forest Classifier

Construct a decision tree

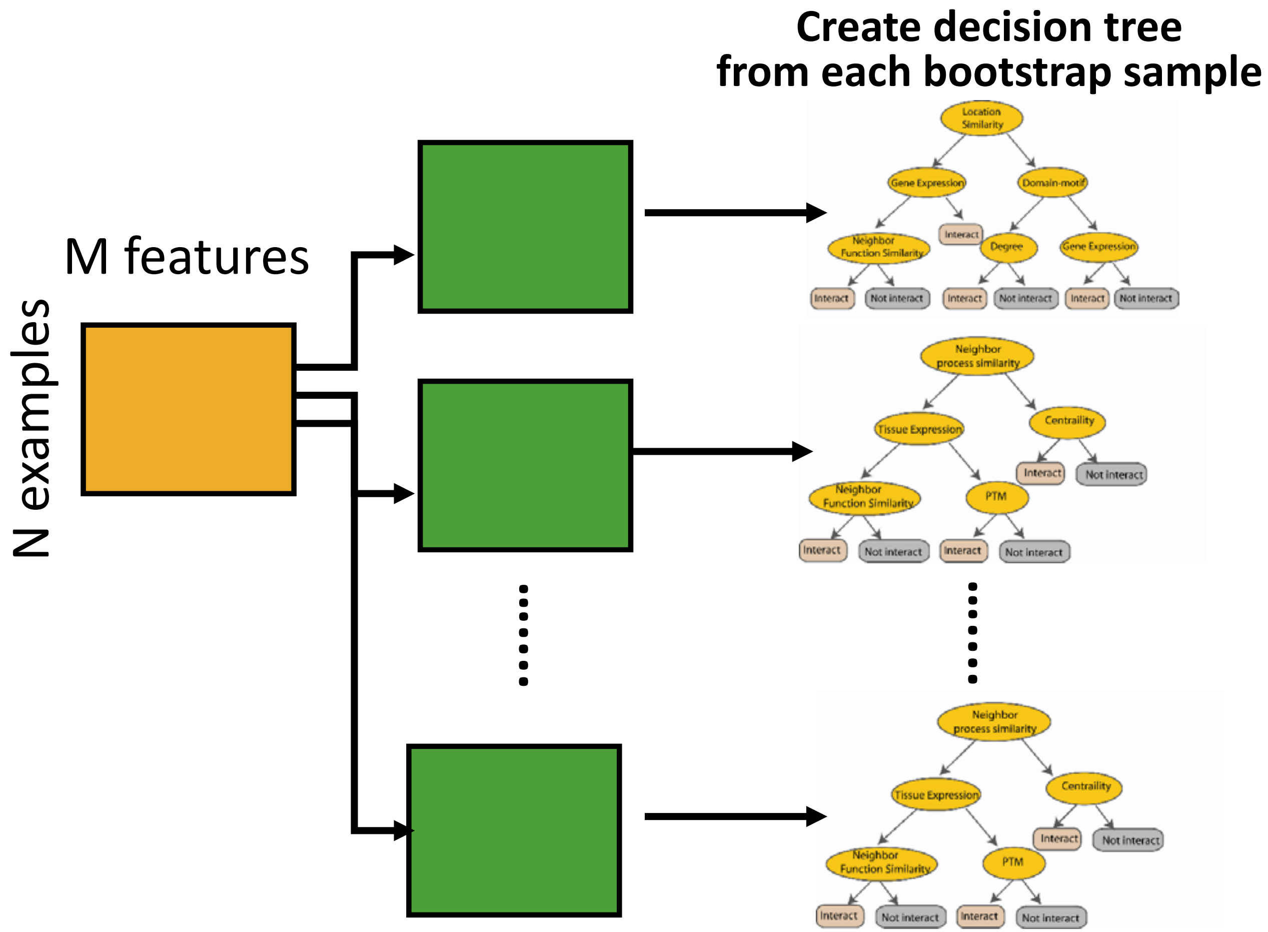


Random Forest Classifier

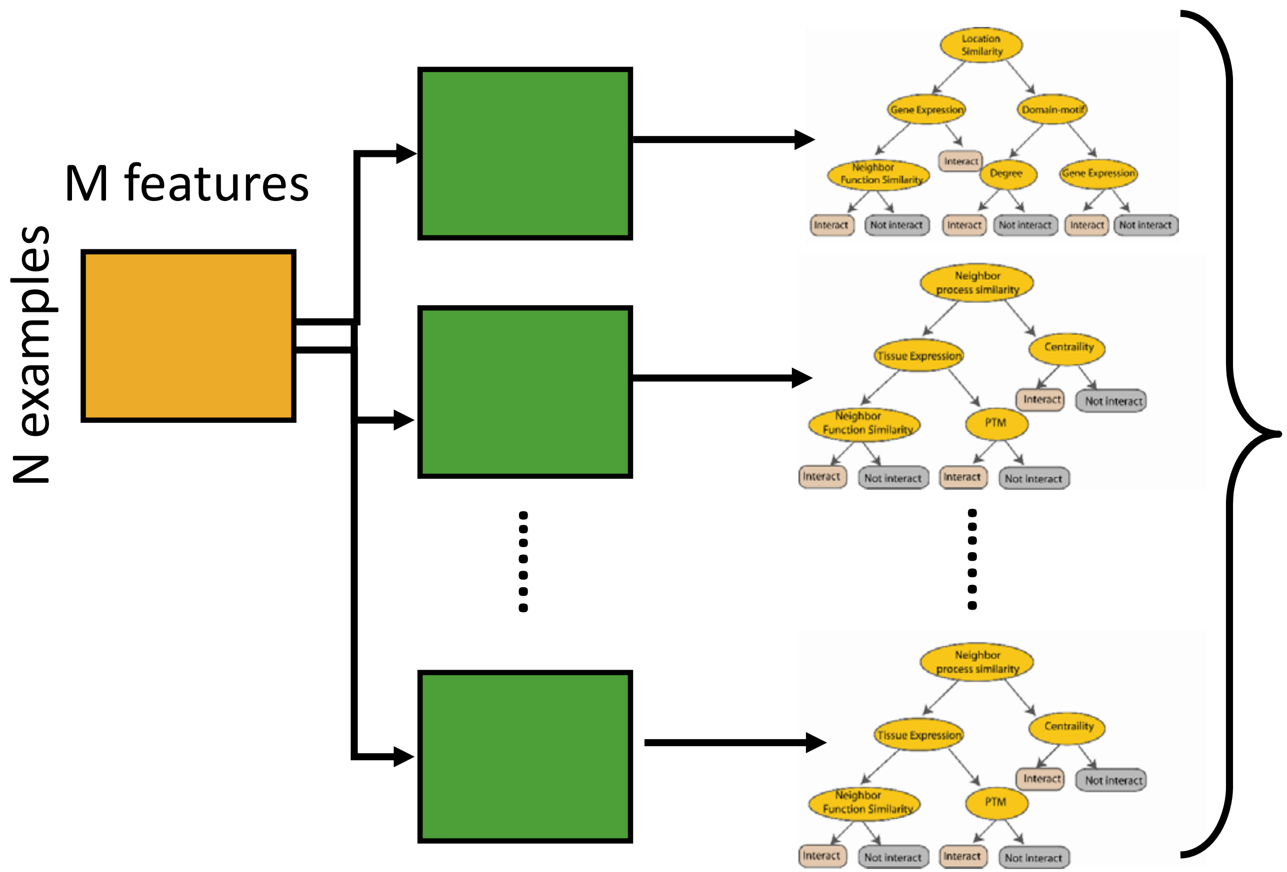
At each node in choosing the split feature
choose only among *m<M* features



Random Forest Classifier



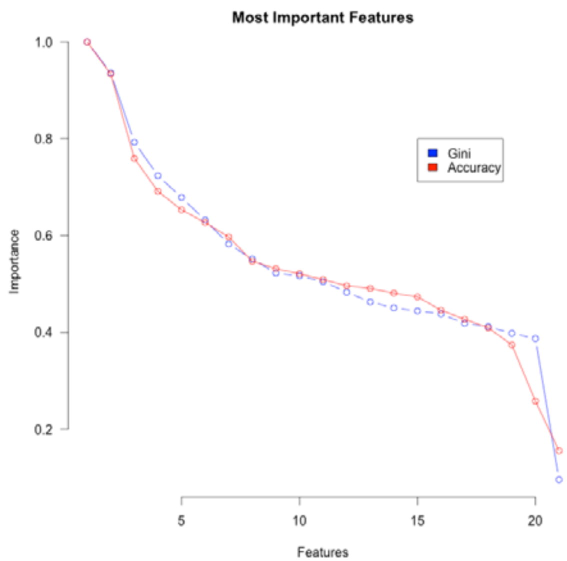
Random Forest Classifier



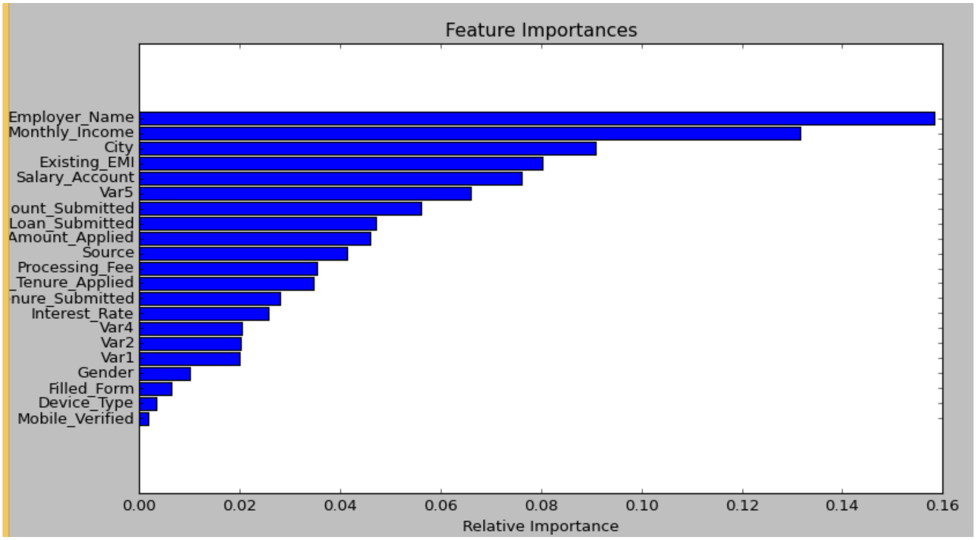
Take he majority vote

Variable importance

- Random Forests output the list of predictor variables and their importance (i.e., importance variable)
- The importance variable can be measured by:
  - GINI index based
    - total decrease in node impurities from splitting on the variable, averaged over all trees
  - permutation test of variables (usually used)
    - If a variable is not important then rearranging the values of that variable will not degrade prediction accuracy



GINI index based



Permutation based