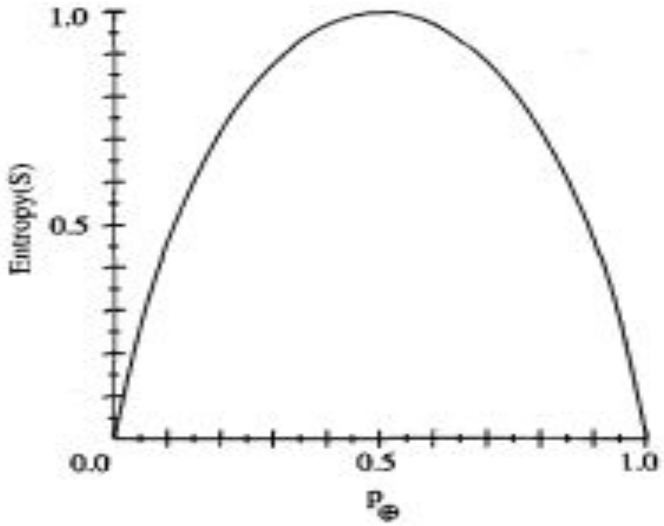
Decision Tree Classifier



1.0
0.5
Entropy(S)
0.0
0.5
1.0
$p_\oplus$

ID3(*Examples*, *Target_attribute*, *Attributes*)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
    - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
    - The decision attribute for *Root* $\leftarrow A$
    - For each possible value, v_i, of A,
        - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
        - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
        - If $Examples_{v_i}$ is empty
            - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
            - Else below this new branch add the subtree
              ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* $- \{A\}$))
- End
- Return *Root*

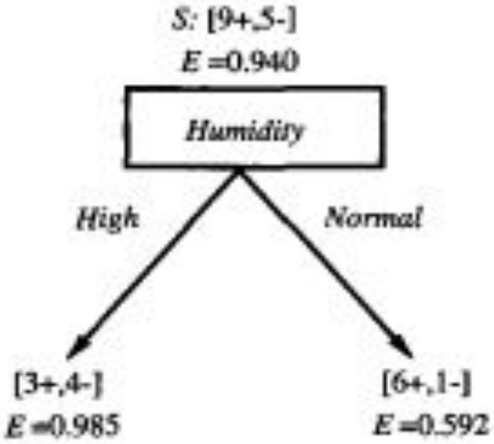
Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$$
\begin{aligned}
Entropy([9+, 5-]) &= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) \\
&= 0.940
\end{aligned}
$$

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad ($$

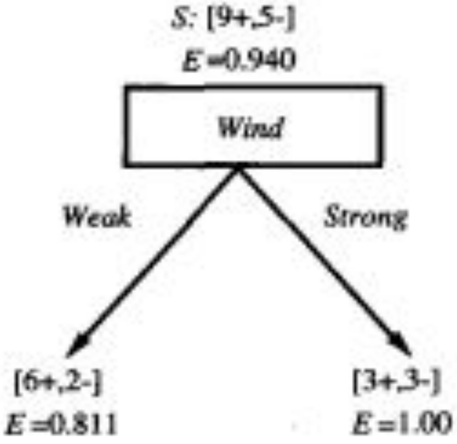
$$
\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}
$$

Which attribute is the best classifier?



Gain (S, Humidity)

$= .940 - (7/14).985 - (7/14).592$
$= .151$



Gain (S, Wind)

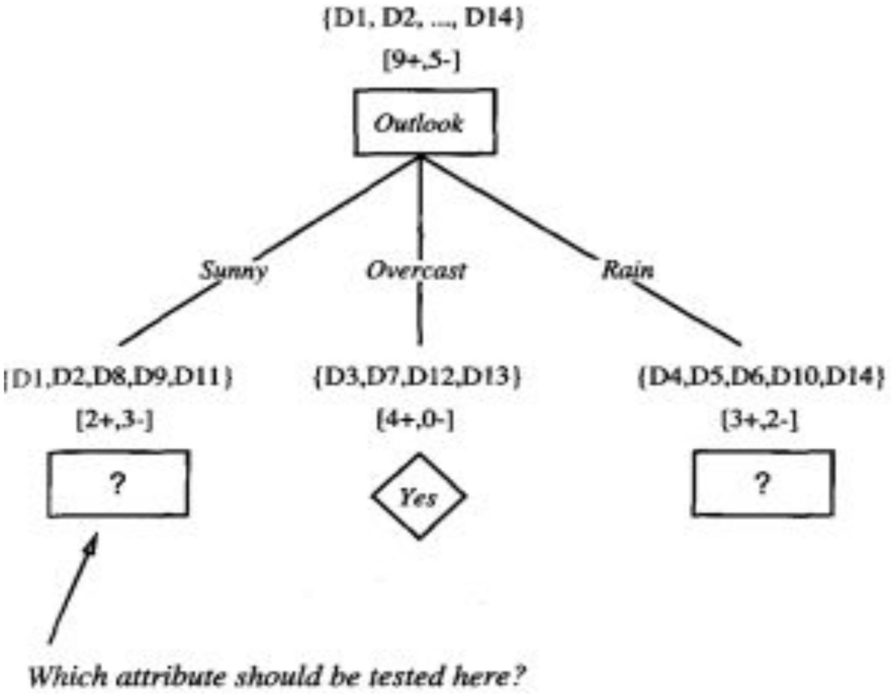
$= .940 - (8/14).811 - (6/14)1.0$
$= .048$

$$Gain(S, Outlook) = 0.246$$

$$Gain(S, Humidity) = 0.151$$

$$Gain(S, Wind) = 0.048$$

$$Gain(S, Temperature) = 0.029$$



S_{sunny} = {D1,D2,D8,D9,D11}

$Gain\ (S_{sunny}, Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

$Gain\ (S_{sunny}, Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$

$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

$$E(S) = -p_{(+)}\log p_{(+)} - p_{(-)}\log p_{(-)}$$

Decision Tree

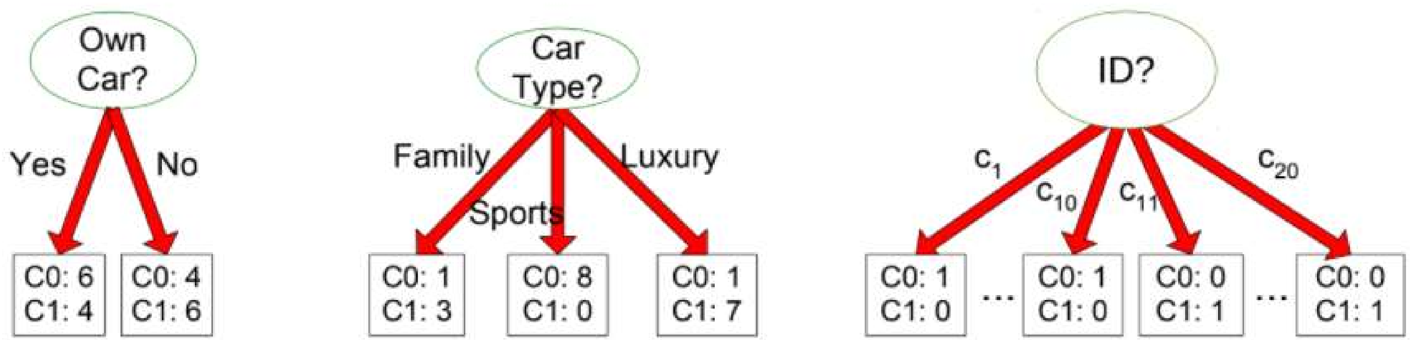
RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Methods

- Ex. information gain, Gain Ratio, Gini index.
- Whether the tree is strictly binary is generally driven by the attribute selection measure.
- Attribute selection measures, such as the Gini index, enforce the resulting tree to be binary.
- Others, like information gain, do not,
  - Therein allowing multiway splits
  - (i.e., two or more branches to be grown from a node)

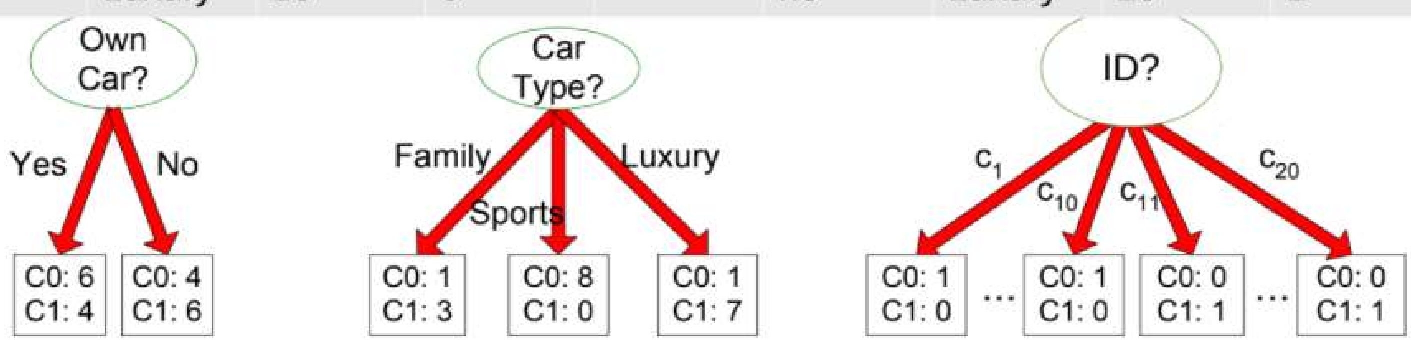
Attribute Selection Methods

- These methods are defined in terms of the class distribution of the records before and after splitting.
- Let p(i|t) denote the fraction of records belonging to class i at a given node t. We sometimes omit the reference to node t and express the fraction as p i .
- In a two-class problem, the class distribution at any node can be written as (p 0 ,p 1), where p 1 =1−p 0
- **Before Splitting: 10 records of class 0,**
  10 records of class 1



Which test condition is the best?

OC	CT	ID	Class		OC	CT	ID	Class
yes	Family	1	0		no	Family	11	1
yes	Sports	2	0		no	Family	12	1
yes	Sports	3	0		no	Family	13	1
yes	Sports	4	0		no	Luxury	14	1
yes	Sports	5	0		yes	Luxury	15	1
yes	Sports	6	0		yes	Luxury	16	1
no	Sports	7	0		yes	Luxury	17	1
no	Sports	8	0		yes	Luxury	18	1
no	Sports	9	0		no	Luxury	19	1
no	Luxury	10	0		no	Luxury	20	1



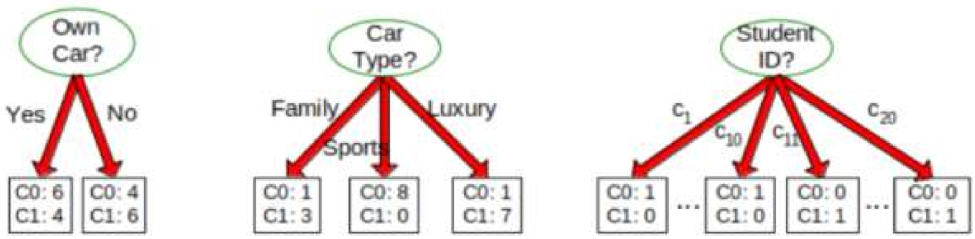
Which test condition is the best?

Attribute Selection Methods

- The class distribution before splitting is (0.5, 0.5) because there are an equal number of records from each class.
- If we split the data using the ***Own Car?*** attribute, then the class distributions of the child nodes are (0.6, 0.4) and (0.4, 0.6), respectively.
- Although the classes are no longer evenly distributed, the child nodes still contain records from both classes.
- Splitting on the second attribute, Car Type, will result in purer partitions.
- The measures developed for selecting the best split are often based on the degree of impurity of the child nodes.
  - The smaller the degree of impurity, the more skewed the class distribution.
  - For example, a node with class distribution (0, 1) has zero impurity, whereas a node with uniform class distribution (0.5, 0.5) has the highest impurity.
- Some of the popular impurity measures are Entropy, Gini and Classification Error.

Gain Ratio

- The information gain measure is biased toward tests with many outcomes.
- It prefers to select attributes having a large number of values.



- Comparing the first test condition, *OwnCar*, with the second,*Car Type:*
- *Car Type* provide a better way of splitting the data since it produces purer descendent nodes.
- However, if we compare both conditions with Customer ID, the latter appears to produce purer partitions.
- Yet Customer ID is not a predictive attribute because its value is unique for each record.

Gain Ratio

- For each outcome, it considers the number of tuples having that outcome with respect to the total number of tuples in D.
- It differs from information gain, which measures the information with respect to classification that is acquired based on the same partitioning

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

- The attribute with the **maximum gain ratio** is selected as the splitting attribute.
- Note, however, that as the split information approaches 0, the ratio becomes unstable.
- A constraint is added to avoid this, whereby the information gain of the test selected must be large—at least as great as the average gain over all tests examined

Gain Ratio : Example

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Outlook	
Info	0.693
Gain: 0.940-0.693	0.247
Split info: info ([5,4,5])	1.577
Gain ratio: 0.247/1.577	0.156

Humidity	
Info	?
Gain: 0.940-0.788	?
Split info: info ([7,7])	?
Gain ratio: 0.152/1	?

Temperature	
Info	?
Gain: 0.940-0.911	?
Split info: info ([4,6,4])	?
Gain ratio: 0.029/1.362	?

Windy	
Info	?
Gain: 0.940-0.892	?
Split info: info ([8,6])	?
Gain ratio: 0.048/0.985	?

Gain Ratio : Example

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Outlook	
Info	0.693
Gain: 0.940-0.693	0.247
Split info: info ([5,4,5])	1.577
Gain ratio: 0.247/1.577	0.156

Class : P-9 N-5

Outlook:

Sunny : P - 2 N - 3
Overcast : P - 4 N - 0
rain : P - 3 N - 2

Gain Ratio : Example

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Outlook	
$Info_{Outlook}$	**0.693**
Gain: 0.940-0.693	0.247
Split info: info ([5,4,5])	1.577
Gain ratio: 0.247/1.577	0.156

Class : P-9 N-5

Outlook:

Sunny : P - 2 N - 3
Overcast : P - 4 N - 0
rain : P - 3 N - 2

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$$= (5/14)[- 2/5 \log_2(2/5) – 3/5 \log_2(3/5)] + (4/14)[- 4/4 \log_2(4/4)] + (5/14)[- 3/5 \log_2(3/5) – 2/5 \log_2(2/5)]$$

= 0.693

Gain Ratio : Example

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Outlook	
Info	0.693
Gain: 0.940-0.693	0.247
Split info: info ([5,4,5])	1.577
Gain ratio: 0.247/1.577	0.156

Class : P-9 N-5

Outlook:

Sunny : P - 2 N - 3
Overcast : P - 4 N - 0
rain : P - 3 N - 2

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

$$= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14)$$

$$= \mathbf{0.940}$$

Gain Ratio : Example

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Outlook	
Info	0.693
Gain: 0.940-0.693	0.247
Split info: info ([5,4,5])	**1.577**
Gain ratio: 0.247/1.577	0.156

Class : P-9 N-5

Outlook:

Sunny : P - 2 N - 3
Overcast : P - 4 N - 0
rain : P - 3 N - 2

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

$= -2^* [(5/14)\log_2(5/14)]\ -(4/14)\log_2(4/14)$

$=$ **1.577**

Gain Ratio : Example

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Outlook	
Info	0.693
Gain: 0.940-0.693	0.247
Split info: info ([5,4,5])	1.577
Gain ratio: 0.247/1.577	**0.156**

Class : P-9 N-5

Outlook:

Sunny : P - 2 N - 3
Overcast : P - 4 N - 0
rain : P - 3 N - 2

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

= 0.247 / 1.577
= **0.156**

Gain Ratio : Example

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Outlook	
Info	0.693
Gain: 0.940-0.693	0.247
Split info: info ([5,4,5])	1.577
Gain ratio: 0.247/1.577	**0.156**

Humidity	
Info	0.788
Gain: 0.940-0.788	0.152
Split info: info ([7,7])	1
Gain ratio: 0.152/1	**0.152**

Temperature	
Info	0.911
Gain: 0.940-0.911	0.029
Split info: info ([4,6,4])	1.362
Gain ratio: 0.029/1.362	**0.021**

Windy	
Info	0.892
Gain: 0.940-0.892	0.048
Split info: info ([8,6])	0.985
Gain ratio: 0.048/0.985	**0.049**

Gini Index

- The coefficient ranges from 0 (or 0%) to 1 (or 100%), with 0 representing perfect equality and 1 representing perfect inequality.
- The Gini index is used in CART.
- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified
- Impurity of D, a data partition or set of training tuples:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

- p_i is the probability that a tuple in D belongs to class C_i and is estimated by $|C_{i,D}| / |D|$.
- The sum is computed over m classes

Gini Index

- The Gini index considers a binary split for each attribute.
- Let's first consider the case where A is a discrete-valued attribute having v distinct values, {a 1 , a 2 ,..., a v }, occurring in D.
- To determine the best binary split on A, we examine all the possible subsets that can be formed using known values of A.
- If A has v possible values, then there are 2^v possible subsets.
- For example, if income has three possible values, namely {low, medium, high}, then the possible subsets are {low, medium, high}, {low, medium}, {low, high}, {medium, high}, {low}, {medium}, {high}, and {}.
- We exclude the power set, {low, medium, high}, and the empty set from consideration since, conceptually, they do not represent a split.
- Therefore, there are 2 ^ v − 2 possible ways to form two partitions of the data, D, based on a binary split on A

Gini Index

- When considering a binary split, we compute a weighted sum of the impurity of each resulting partition.
- For example, if a binary split on A, partitions D into D 1 and D 2 , the Gini index of D given that partitioning is:

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

- For each attribute, each of the possible binary splits is considered.
- For a discrete-valued attribute, the subset that gives the minimum Gini index for that attribute is selected as its splitting subset.
- For continuous-valued attributes, each possible split-point must be considered
- The strategy is similar to that described earlier for information gain, where the midpoint between each pair of (sorted) adjacent values is taken as a possible split-point.
- The point giving the minimum Gini index for a given (continuous-valued) attribute is taken as the split-point of that attribute

Gini Index

- The reduction in impurity that would be incurred by a binary split on a discrete-or continuous-valued attribute A is :

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

- The attribute that maximizes the reduction in impurity (or, equivalently, has the minimum Gini index) is selected as the splitting attribute.
- This attribute and either its splitting subset (for a discrete-valued splitting attribute) or split-point (for a continuous-valued splitting attribute) together form the splitting criterion

Gini Index: Example

Class	Var1	Var2
A	0	33
A	0	54
A	0	56
A	0	42
A	1	50
B	1	55
B	1	31
B	0	-4
B	1	77
B	0	49

For Var1

Var1 has 4 instances (4/10) where it's equal to 1 and
6 instances (6/10) when it's equal to 0.

For Var1 == 1 & Class == A: 1 / 4 instances

For Var1 == 1 & Class == B: 3 / 4 instances

Gini Index here is 1-((1/4)^2 + (3/4)^2) = 0.375

For Var1 == 0 & Class== A: 4 / 6 instances

For Var1 == 0 & Class == B: 2 / 6 instances

Gini Index here is 1-((4/6)^2 + (2/6)^2) = 0.4444

We then weight and sum each of the splits based on the proportion of the data each split takes up.

4/10 * 0.375 + 6/10 * 0.444 = **0.41667**

Gini Index: Example

Class	Var1	Var2
A	0	33
A	0	54
A	0	56
A	0	42
A	1	50
B	1	55
B	1	31
B	0	-4
B	1	77
B	0	49

For Var2 (Let's Threshold T>=32)

Var2 has 8 instances (8/10) where it's >= 32 and

2 instances (2/10) when it's < 32.

For Var2 >=32 & Class == A: 5 / 8 instances

For Var2 >=32 & Class == B: 3 / 8 instances

Gini Index here is 1-((5/8)^2 + (3/8)^2) = 0.46875

For Var2 < 32 & Class== A: 0 / 2 instances

For Var1 < 32 & Class == B: 2 / 2 instances

Gini Index here is 1-((0/2)^2 + (2/2)^2) = 0

We then weight and sum each of the splits based on the proportion of the data each split takes up.

8/10 * 0.46875 + 2/10 * 0 = **0.375**

Gini Index: Example

Class	Var1	Var2
A	0	33
A	0	54
A	0	56
A	0	42
A	1	50
B	1	55
B	1	31
B	0	-4
B	1	77
B	0	49

For Var1 = 0.41667

For Var2 (T>=32) = **0.375**

Based on these results, Var2>=32 is selected as the split.(since its weighted Gini Index is smallest)

The next step would be to take the results from the split and further partition.

Overfitting Due to Noise: An Example

An example training set for classifying mammals. Asterisks denote mislabelings.

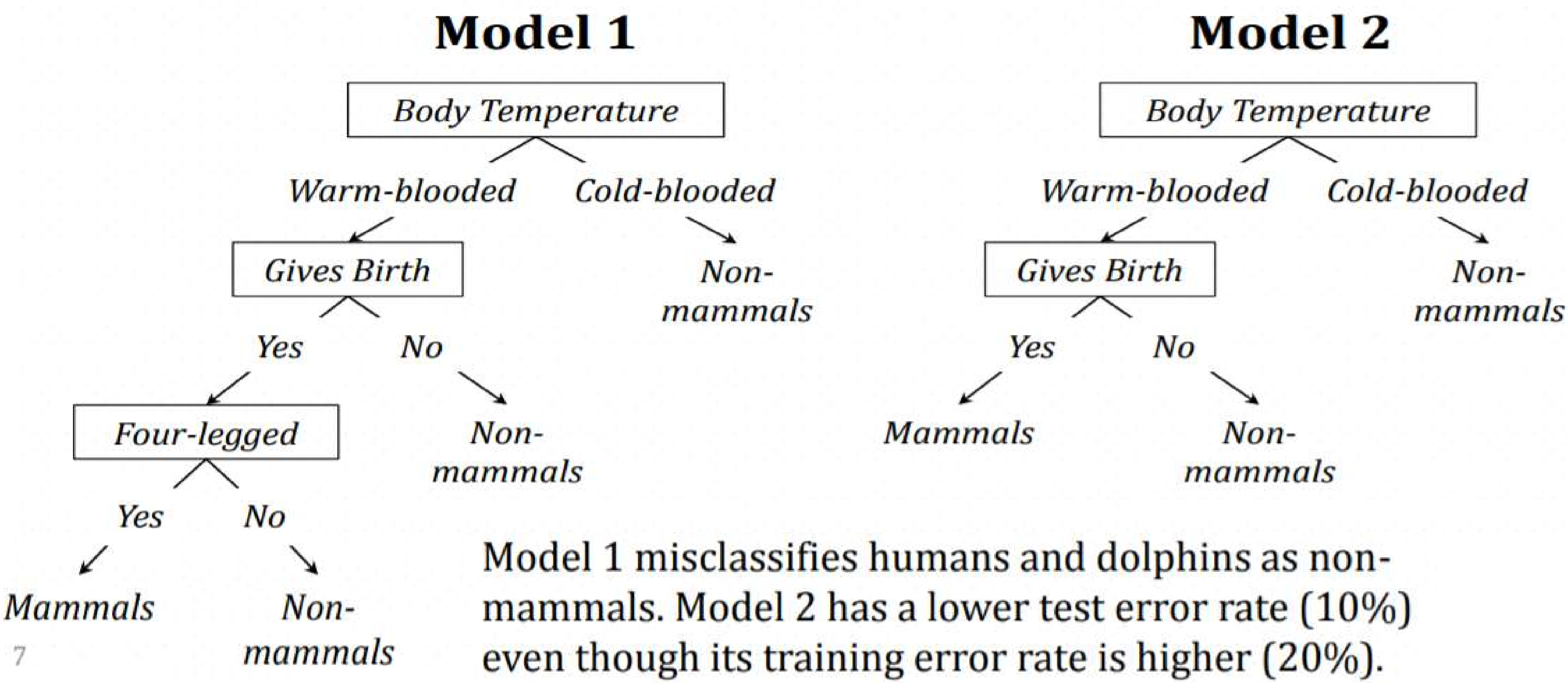
Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
Porcupine	Warm-blooded	Yes	Yes	Yes	***Yes***
Cat	Warm-blooded	Yes	Yes	No	***Yes***
Bat	Warm-blooded	Yes	No	Yes	***No****
Whale	Warm-blooded	Yes	No	No	***No****
Salamander	Cold-blooded	No	Yes	Yes	***No***
Komodo dragon	Cold-blooded	No	Yes	No	***No***
Python	Cold-blooded	No	No	Yes	***No***
Salmon	Cold-blooded	No	No	No	***No***
Eagle	Warm-blooded	No	No	No	***No***
Guppy	Cold-blooded	Yes	No	No	***No***

Overfitting Due to Noise

An example testing set for classifying mammals.

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
Human	Warm-blooded	Yes	No	No	***Yes***
Pigeon	Warm-blooded	No	No	No	***No***
Elephant	Warm-blooded	Yes	Yes	No	***Yes***
Leopard shark	Cold-blooded	Yes	No	No	***No***
Turtle	Cold-blooded	No	Yes	No	***No***
Penguin	Cold-blooded	No	No	No	***No***
Eel	Cold-blooded	No	No	No	***No***
Dolphin	Warm-blooded	Yes	No	No	***Yes***
Spiny anteater	Warm-blooded	No	Yes	Yes	***Yes***
Gila monster	Cold-blooded	No	Yes	Yes	***No***

Overfitting Due to Noise



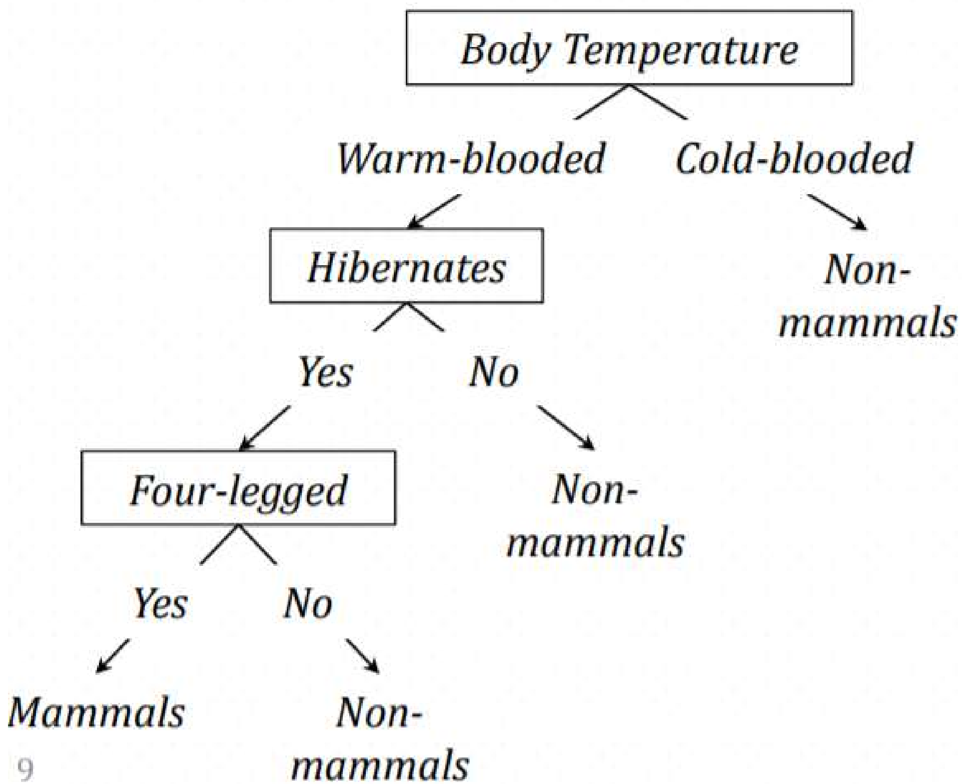
Model 1 misclassifies humans and dolphins as non-mammals. Model 2 has a lower test error rate (10%) even though its training error rate is higher (20%).

Overfitting Due to Lack of Samples

An example training set for classifying mammals.

Name	**Body Temperature**	**Gives Birth**	**Four-legged**	**Hibernates**	**Class Label**
Salamander	Cold-blooded	No	Yes	Yes	***No***
Guppy	Cold-blooded	Yes	No	No	***No***
Eagle	Warm-blooded	No	No	No	***No***
Poorwill	Warm-blooded	No	No	Yes	***No***
Platypus	Warm-blooded	No	Yes	Yes	***Yes***

Overfitting Due to Lack of Samples



Although the model's training error is zero, its error rate on the test set if 30%.

Humans, elephants, and dolphins are misclassified because the decision tree classifies all warm-blooded vertebrates that do not hibernate as non-mammals. The tree arrives at this classification decision because there is only one training records, which is an eagle, with such characteristics.

Overfitting Solution: Prepruning

- By halting its construction early (e.g., by deciding not to further split or partition the subset of training tuples at a given node).
- Upon halting, the node becomes a leaf.
- The leaf may hold the most frequent class among the subset tuples
- If partitioning the tuples at a node would result in a split that falls below a pre-specified threshold, then further partitioning of the given subset is halted.
- There are difficulties, however, in choosing an appropriate threshold.
  - High thresholds : oversimplified trees
  - low thresholds : very little simplification.

- Max_leaf_nodes
- Min_samples_leaf(Data in each node 30,100,300)
- Max_depth

Overfitting Solution: Postpruning

- Removes subtrees from a “fully grown” tree and replace it with leaf
- The leaf is labeled with the most frequent class among the subtree being replaced

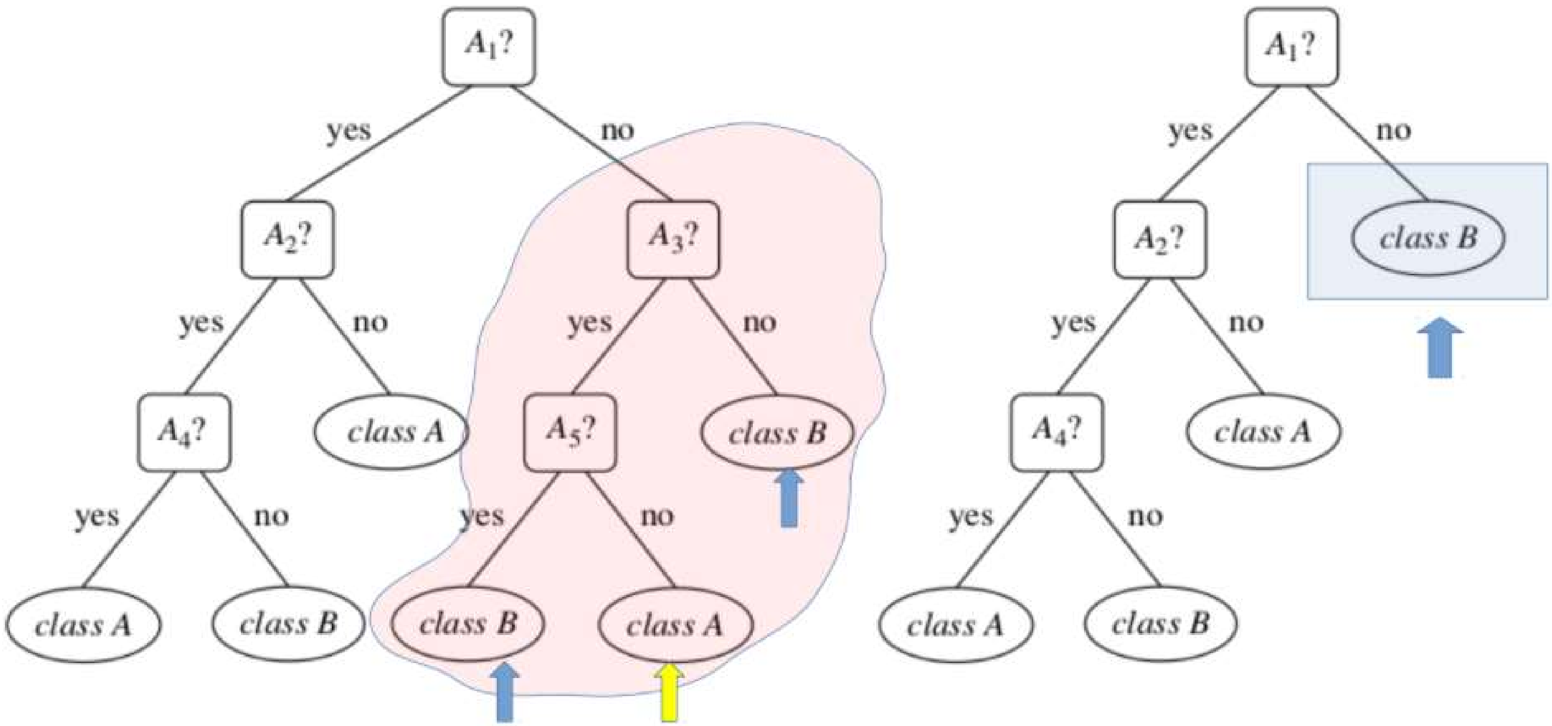
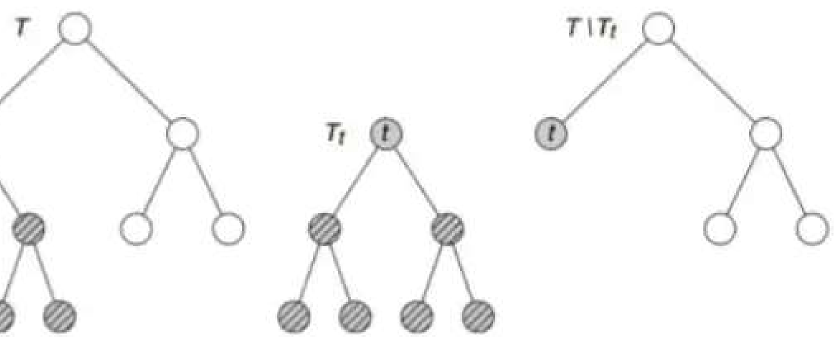
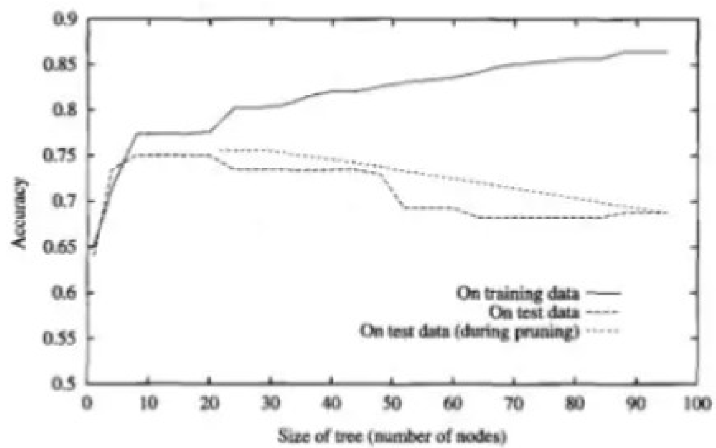


Illustration:





Accuracy
Size of tree (number of nodes)
On training data
On test data
On test data (during pruning)

Postpruning: Cost Complexity Pruning

- Used in CART as a postpruning approach.
- It Considers the cost complexity of a tree to be a function of the number of leaves in the tree and the error rate of the tree (where the error rate is the percentage of tuples misclassified by the tree).
- It starts from the bottom of the tree.
- For each internal node, N, it computes:
  - 1 cost complexity of the subtree at N,
  - 2 cost complexity of the subtree at N if it were to be pruned

    (i.e., replaced by a leaf node).
- The two values are compared. If pruning the subtree at node N would result in a smaller cost complexity, then the subtree is pruned. Otherwise, it is kept.
- A **pruning set** of class-labeled tuples is used to estimate cost complexity.
- This set is independent of the training set used to build the unpruned tree and of any test set used for accuracy estimation.
- In general, the smallest decision tree that minimizes the cost complexity is preferred.

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up

- **Calculating the Gini Index for Past Trend**
- P(Past Trend=Positive): 6/10
- P(Past Trend=Negative): 4/10
- If (Past Trend = Positive & Return = Up), probability = 4/6
- If (Past Trend = Positive & Return = Down), probability = 2/6
- Gini index = 1 - ((4/6)^2 + (2/6)^2) = 0.45
- If (Past Trend = Negative & Return = Up), probability = 0
- If (Past Trend = Negative & Return = Down), probability = 4/4
- Gini index = 1 - ((0)^2 + (4/4)^2) = 0
- Weighted sum of the Gini Indices can be calculated as follows:
- Gini Index for Past Trend = (6/10)0.45 + (4/10)0 = 0.27

- Calculation of Gini Index for Open Interest
- P(Open Interest=High): 4/10
- P(Open Interest=Low): 6/10

- If (Open Interest = High & Return = Up), probability = 2/4
- If (Open Interest = High & Return = Down), probability = 2/4
- Gini index = 1 - ((2/4)^2 + (2/4)^2) = 0.5

- If (Open Interest = Low & Return = Up), probability = 2/6
- If (Open Interest = Low & Return = Down), probability = 4/6
- Gini index = 1 - ((2/6)^2 + (4/6)^2) = 0.45

- Weighted sum of the Gini Indices can be calculated as follows:
- Gini Index for Open Interest = (4/10)0.5 + (6/10)0.45 = 0.47

- Calculation of Gini Index for Trading Volume
- P(Trading Volume=High): 7/10
-
- P(Trading Volume=Low): 3/10
-
- If (Trading Volume = High & Return = Up), probability = 4/7
- If (Trading Volume = High & Return = Down), probability = 3/7
- Gini index = 1 - ((4/7)^2 + (3/7)^2) = 0.49
-
- If (Trading Volume = Low & Return = Up), probability = 0
- If (Trading Volume = Low & Return = Down), probability = 3/3
- Gini index = 1 - ((0)^2 + (1)^2) = 0
-
- Weighted sum of the Gini Indices can be calculated as follows:
- Gini Index for Trading Volume = (7/10)0.49 + (3/10)0 = 0.34

Attributes/Features	**Gini Index**
Past Trend	0.27
Open Interest	0.47
Trading Volume	0.34

- From the above table, we observe that ‘Past Trend’ has the lowest Gini Index and hence it will be chosen as the root node for how decision tree works.
-
- We will repeat the same procedure to determine the sub-nodes or branches of the decision tree.

We will calculate the Gini Index for the 'Positive' branch of Past Trend as follows:

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Positive	Low	High	Up
Positive	High	High	Up
Positive	Low	Low	Down
Positive	Low	Low	Down
Positive	High	High	Up

- Calculation of Gini Index of Open Interest for Positive Past Trend
- P(Open Interest=High): 2/6
-
- P(Open Interest=Low): 4/6
-
- If (Open Interest = High & Return = Up), probability = 2/2
- If (Open Interest = High & Return = Down), probability = 0
- Gini index = 1 - (sq(2/2) + sq(0)) = 0
-
- If (Open Interest = Low & Return = Up), probability = 2/4
- If (Open Interest = Low & Return = Down), probability = 2/4
- Gini index = 1 - (sq(0) + sq(2/4)) = 0.50
-
- Weighted sum of the Gini Indices can be calculated as follows:
- Gini Index for Open Interest = (2/6)0 + (4/6)0.50 = 0.33

- Calculation of Gini Index for Trading Volume
- P(Trading Volume=High): 4/6
-
- P(Trading Volume=Low): 2/6
-
- If (Trading Volume = High & Return = Up), probability = 4/4
- If (Trading Volume = High & Return = Down), probability = 0
- Gini index = 1 - (sq(4/4) + sq(0)) = 0
-
- If (Trading Volume = Low & Return = Up), probability = 0
- If (Trading Volume = Low & Return = Down), probability = 2/2
- Gini index = 1 - (sq(0) + sq(2/2)) = 0
-
- Weighted sum of the Gini Indices can be calculated as follows:
- Gini Index for Trading Volume = (4/6)0 + (2/6)0 = 0

Attributes/Features	Gini Index
Open Interest	0.33
Trading Volume	0

We will split the node further using the ‘Trading Volume’ feature, as it has the minimum Gini index.

References

https://www.youtube.com/watch?v=0JVmcCZLL7g

https://www.youtube.com/watch?v=FuJVLsZYkuE

Machine Learning by Tom Mitchell

https://www.youtube.com/watch?v=IZnno-dKgVQ

http://cv.znu.ac.ir/afsharchim/AI/lectures/Decision%20Trees%203.pdf