Lecture 5
Classification by Decision Tree

Dr. Mourad YKHLEF

The slides content is derived and adopted from many references

- The Course

DS = Data source
DW = Data warehouse
DM = Data Mining
DP = Staging Database

OLAP

DM

Association

Classification

Clustering
What is Classification?

- The goal of data classification is to organize and categorize data in distinct classes.
  - A model is first created based on the data distribution.
  - The model is then used to classify new data.
  - Given the model, a class can be predicted for new data.

- Classification = prediction for discrete and nominal values

What is Prediction?

- The goal of prediction is to forecast or deduce the value of an attribute based on values of other attributes.
  - A model is first created based on the data distribution.
  - The model is then used to predict future or unknown values

- **In Data Mining**
  - If forecasting discrete value → **Classification**
  - If forecasting continuous value → **Prediction**
Supervised and Unsupervised

- **Supervised Classification = Classification**
  - We know the class labels and the number of classes

- **Unsupervised Classification = Clustering**
  - We do not know the class labels and may not know the number of classes

Preparing Data Before Classification

- **Data transformation:**
  - Discretization of continuous data
  - Normalization to [-1..1] or [0..1]

- **Data Cleaning:**
  - Smoothing to reduce noise

- **Relevance Analysis:**
  - Feature selection to eliminate irrelevant attributes
Application

- Credit approval
- Target marketing
- Medical diagnosis
- Defective parts identification in manufacturing
- Crime zoning
- Treatment effectiveness analysis
- Etc

Supervised learning process: 3 steps

1. **Training Data** → **Classification Method** → **Classification Model**
2. **Test Data** → **Classification Model** → **Accuracy**
3. **New Data** → **Classification Model** → **Class**
Classification is a 3-step process

1. Model construction (Learning):
   - Each tuple is assumed to belong to a predefined class, as determined by one of the attributes, called the class label.
   - The set of all tuples used for construction of the model is called training set.

   - The model is represented in the following forms:
     - Classification rules, (IF-THEN statements),
     - Decision tree
     - Mathematical formulae

1. Classification Process (Learning)

<table>
<thead>
<tr>
<th>Name</th>
<th>Income</th>
<th>Age</th>
<th>Leasing rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samir</td>
<td>Low</td>
<td>&lt;30</td>
<td>bad</td>
</tr>
<tr>
<td>Ahmed</td>
<td>Medium</td>
<td>[30...40]</td>
<td>good</td>
</tr>
<tr>
<td>Salah</td>
<td>High</td>
<td>&lt;30</td>
<td>good</td>
</tr>
<tr>
<td>Ali</td>
<td>Medium</td>
<td>&gt;40</td>
<td>good</td>
</tr>
<tr>
<td>Sami</td>
<td>Low</td>
<td>[30..40]</td>
<td>good</td>
</tr>
<tr>
<td>Emad</td>
<td>Medium</td>
<td>&lt;30</td>
<td>bad</td>
</tr>
</tbody>
</table>

Classification Method

IF Income = 'High'
OR Age > 30
THEN Class = 'Good'
OR
Decision Tree
OR
Mathematical Formulae
Classification is a 3-step process

2. Model Evaluation (Accuracy):
- Estimate accuracy rate of the model based on a test set.
- The known label of test sample is compared with the classified result from the model.
- Accuracy rate is the percentage of test set samples that are correctly classified by the model.
- Test set is independent of training set otherwise over-fitting will occur.

### 2. Classification Process (Accuracy Evaluation)

<table>
<thead>
<tr>
<th>Name</th>
<th>Income</th>
<th>Age</th>
<th>Leasing rating</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naser</td>
<td>Low</td>
<td>&lt;30</td>
<td>Bad</td>
<td>Bad</td>
</tr>
<tr>
<td>Lutfi</td>
<td>Medium</td>
<td>&lt;30</td>
<td>Bad</td>
<td>good</td>
</tr>
<tr>
<td>Adel</td>
<td>High</td>
<td>&gt;40</td>
<td>good</td>
<td>good</td>
</tr>
<tr>
<td>Fahd</td>
<td>Medium</td>
<td>[30..40]</td>
<td>good</td>
<td>good</td>
</tr>
</tbody>
</table>

Accuracy 75%
Classification is a three-step process

3. Model Use (Classification):
   - The model is used to classify unseen objects.
     - Give a class label to a new tuple
     - Predict the value of an actual attribute

3. Classification Process (Use)
Classification Methods

- Decision Tree Induction
- Neural Networks
- Bayesian Classification
- Association-Based Classification
- K-Nearest Neighbour
- Case-Based Reasoning
- Genetic Algorithms
- Rough Set Theory
- Fuzzy Sets
- Etc.

Evaluating Classification Methods

- **Predictive accuracy**
  - Ability of the model to correctly predict the class label

- **Speed and scalability**
  - Time to construct the model
  - Time to use the model

- **Robustness**
  - Handling noise and missing values

- **Scalability**
  - Efficiency in large databases (not memory resident data)

- **Interpretability:**
  - The level of understanding and insight provided by the model
What is a Decision Tree?

- A decision tree is a flow-chart-like tree structure.
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
    - All tuples in branch have the same value for the tested attribute.

- Leaf node represents class label or class label distribution
Sample Decision Tree

Income

Age

2000 6000 10000

< 6K

YES

>= 6K

No

Excellent customers

Fair customers

Sample Decision Tree

Income

Age

2000 6000 10000

< 6K

NO

<=50

Yes

>50

NO
### Decision-Tree Classification Methods

- The basic top-down decision tree generation approach usually consists of two phases:

  1. **Tree construction**
     - At the start, all the training examples are at the root.
     - Partition examples are recursively based on selected attributes.

  2. **Tree pruning**
     - Aiming at removing tree branches that may reflect noise in the training data and lead to errors when classifying test data → improve classification accuracy
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

- Depends on number of ways to split
  - 2-way split
  - Multi-way split
Splitting Based on Nominal Attributes

- **Multi-way split:** Use as many partitions as distinct values.

- **Binary split:** Divides values into two subsets. Need to find optimal partitioning.

Splitting Based on Ordinal Attributes

- **Multi-way split:** Use as many partitions as distinct values.

- **Binary split:** Divides values into two subsets. Need to find optimal partitioning.

What about this split?
Splitting Based on Continuous Attributes

- Different ways of handling
  - **Discretization** to form an ordinal categorical attribute
    - *Static* – discretize once at the beginning
    - *Dynamic* – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - **Binary Decision**: \((A < v)\) or \((A \geq v)\)
    - consider all possible splits and finds the best cut
    - can be more compute intensive

(i) Binary split

(ii) Multi-way split
Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to determine the Best Split

- **Income**
  - <10k
  - >=10k

- **Age**
  - young
  - old

- Customers
  - Good customers
  - fair customers
How to determine the Best Split

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred

- Need a measure of node impurity:

  ![Image showing node impurity levels]

  - High degree of impurity
  - Low degree of impurity
  - Pure

Measures of Node Impurity

- Information gain
  - Uses Entropy

- Gain Ratio
  - Uses Information Gain and Splitinfo

- Gini Index
  - Used only for binary splits
Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Classification Algorithms

- ID3
  - Uses information gain

- C4.5
  - Uses Gain Ratio

- CART
  - Uses Gini
Information theory (1/5)

- Intuition: more an event is probable, less it brings an information
  - E.g., You are in desert, someone told you: “tomorrow it will be sunny (more probable event)”, this message brings no information, the less you know the more information is provided
- The information quantity $H$ associated to an event is a decreased function with its probability

$$ h(X) = f \left( \frac{1}{\text{Proba}(X)} \right) $$

- The information quantity of two independent variables $X$ and $Y$ is

$$ h(X, Y) = f \left( \frac{1}{\text{Proba}(X, Y)} \right) = f \left( \frac{1}{\text{Proba}(X) \times \text{Proba}(Y)} \right) = h(X) + h(Y) $$

- Choice

$$ h(X) = - \log_2(\text{Proba}(X)) $$

Information theory (2/5)

- Why logarithm?
  - Want the information, when there is one on/off relay (2 choices), to be 1 bit of information. We get this with $\log_2 2$.
    - One relay: 0 or 1 are the choices (on or off)
  - Want the information, when there are 3 relays ($2^3 = 8$ choices) to be 3 times as much (or 3 bits of) information. We get this with $\log_2 8 = 3$.
    - Three relays: 000, 001, 011, 010, 100, 110, or 111 give possible values for all three relays.
Information theory (3/5)

- $S$ sample of training data, $s$ element from $S$
- $p$ elements of the class $P$ of positive examples
- $n$ elements of the class $N$ of negative examples
- $\text{Proba}(s \text{ belongs to } P) = \frac{p}{p+n}$
- The information quantity needed to decide whether $s$ is belonging to $P$ or to $N$

\[
I(p,n) = -\frac{p}{p+n} \log_2 \left( \frac{p}{p+n} \right) - \frac{n}{p+n} \log_2 \left( \frac{n}{p+n} \right)
\]

Information theory -Coding- (4/5)

**Intuitive example (Biology):**

- Suppose that we have four symbols $A C G T$ with probabilities
  
  \begin{align*}
  P_A &= 1/2, & P_C &= 1/4, \\
  \end{align*}

- Information quantity
  \begin{align*}
  H(A) &= -\log_2(P_A) = 1 \text{bits}, & H(C) &= 2 \text{bits} \\
  H(G) &= 3 \text{bits}, & H(T) &= 3 \text{bits}
  \end{align*}

  \[
  I = \frac{1}{2}(1) + \frac{1}{4}(2) + \frac{1}{8}(3) + \frac{1}{8}(3) = 1.75 \text{ (bits per symbol)}
  \]

- If code(A)=1, code(C)=01, code(G)=000, code(T)=001 so the string of 8 symbols ACATGAAC is coded as 10110010001101 (14 bits)

  for 8 symbols we need 14 bits, so the average is 14/8=1.75 bits per symbol
Information theory-Entropy (5/5)

- **Information theory**: optimal length code assigns \(-\log_2(p)\) bits to a message with probability \(p\)
- **Entropy(S)** expected number of bits needed to encode the class of a randomly chosen member of \(S\)

\[
Entropy(S) = I(p, n) = \frac{p}{p+n}(-\log_2(p)) + \frac{n}{p+n}(-\log_2(n))
\]

- Test on a single attribute \(A\) will give us some of information.
  - \(A\) divides \(S\) into subsets \(S_1, \ldots, S_v\) (i.e., \(A\) can have \(v\) distinct values)
  - Each \(S_i\) has positive and negative examples \(p_i\) and \(n_i\)
  - After testing \(A\) we need \(Entropy(A)\) bits of information to classify the sample

\[
Entropy(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} I(p_i, n_i)
\]

- Information gain from \(A\) attribute test is the difference between the original information requirement (before splitting) and the new requirement (information after splitting)

\[
Gain(A) = Entropy(S) - Entropy(A)
\]

### Heuristic
- Choose attribute with the largest gain

---

**Example (1/2)**

Class \(P\): buys_computer = “yes”
Class \(N\): buys_computer = “no”

\(I(p, n) = I(9, 5) =\)

\(-9/14 \times \log(9/14)/\log2 - 5/14 \times \log(5/14)/\log2 = 0.940\)

Compute the entropy for age:

<table>
<thead>
<tr>
<th>age</th>
<th>(p_i)</th>
<th>(n_i)</th>
<th>(I(p_i, n_i))</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>2</td>
<td>3</td>
<td>0.971</td>
</tr>
<tr>
<td>31...40</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&gt;40</td>
<td>3</td>
<td>2</td>
<td>0.971</td>
</tr>
</tbody>
</table>

\(E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694\)

\(\frac{5}{14} I(2,3)\) means “age <=30” has 5 out of 14 samples, with 2 yes and 3 no.

Hence

\[Gain(age) = I(p, n) - E(age) = 0.246\]

Similarly,

\[Gain(income) = 0.029\]
\[Gain(student) = 0.151\]
\[Gain(leasing_rating) = 0.048\]
Example (2/2)

Training set

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Target Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>C1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>C1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>C2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>C2</td>
</tr>
</tbody>
</table>
**Entropy (general case)**

- Select the attribute with the highest information gain
- S sample of training data contains $s_i$ tuples of class $C_i$ for $i = 1, \ldots, m$
  - for boolean classification $m=2$ (positive, negative) (see last example)

- **information** measures info required to classify any arbitrary tuple
  \[
  I(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} p_i \log_2(p_i) \quad \text{where} \quad p_i = \frac{s_i}{s}
  \]

- **entropy** of attribute $A$ with values $\{a_1, a_2, \ldots, a_v\}$
  \[
  E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \ldots + s_{mj}}{s} I(s_{1j}, \ldots, s_{mj})
  \]

- **information gained** by branching on attribute $A$
  \[
  Gain(A) = I(s_1, s_2, \ldots, s_m) - E(A)
  \]

---

**The ID3 Algorithm**

**Generate_decision_tree(samples, attrib-list)**

1. Create a Node $N$;
2. If samples are all of the same class, $c_i$ then
   - return $N$ as a leaf node labeled with the class $c_i$;
3. If the $attrib_list$ is empty then
   - return $N$ as a leaf node labeled with the most common class in $samples$;
4. Select $test_attribute$, the attribute among $attrib_list$ with the highest information gain
5. Label node $N$ with $test_attribute$;
6. For each known value of $a_i$ of $test_attribute$
   - grow a branch from Node $N$ for the condition $test_attribute = a_i$;
7. let $s_i$ be the set of samples in $samples$ for which $test_attribute = a_i$;
8. if $s_i$ is empty then (null values in all attributes)
   - attach a leaf labeled with the most common class in $samples$;
9. else attach the node returned by
   \[
   \text{// recompute the attributes gains and reorder with the highest info gain}
   \]
10. $Generate\_decision\_tree(s_i, attrib\_list \text{ minus test\_attrib})$;
The ID3 Algorithm

- Conditions for stopping partitioning
  
  - All samples for a given node belong to the same class (steps 2 and 3)
  
  - There are no remaining attributes for further partitioning (step 4). In this case majority voting is employed (step 5) for classifying the leaf
  
  - There are no samples left (step 11)

```sql
Generate_decision_tree(sample, {age, student, leasing, income}) L9-L12
```

```sql
select income, student, leasing, buys
from sample
where age = 31..40
```
Generate_decision_tree(s, {student,leasing,income}) L9-L12

Generate_decision_tree(s, {student,leasing,income}) L2-L3

Generate_decision_tree(s, {leasing,income}) L2-L3
\[
\text{Generate\_decision\_tree}(s, \{\text{student,leasing,income}\}) \quad 1.2-1.3
\]

\[
\begin{array}{|c|c|c|c|}
\hline
\text{income} & \text{student} & \text{leasing} & \text{buys} \\
\hline
\text{high} & \text{no} & \text{fair} & \text{yes} \\
\text{low} & \text{yes} & \text{excellent} & \text{yes} \\
\text{medium} & \text{no} & \text{excellent} & \text{yes} \\
\text{high} & \text{yes} & \text{fair} & \text{yes} \\
\hline
\end{array}
\]

\[
\text{Generate\_decision\_tree}(s, \{\text{student,leasing,income}\}) \quad 1.9-1.12
\]

\[
\begin{array}{|c|c|c|c|}
\hline
\text{income} & \text{leasing} & \text{buys} \\
\hline
\text{medium} & \text{no} & \text{fair} & \text{yes} \\
\text{low} & \text{yes} & \text{fair} & \text{yes} \\
\text{low} & \text{yes} & \text{excellent} & \text{no} \\
\text{medium} & \text{yes} & \text{fair} & \text{yes} \\
\text{medium} & \text{no} & \text{excellent} & \text{no} \\
\text{medium} & \text{fair} & \text{yes} \\
\text{medium} & \text{excellent} & \text{no} \\
\hline
\end{array}
\]
51

age?

<=30

31..40

>40

student?

yes

student?

no

yes

no

yes

leasing rating?

excellent

fair

excellent

fair

no

yes

no

yes

52
Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are easier for humans to understand

Example

IF age = “<=30” AND student = “no”  THEN buys_computer = “no”
IF age = “<=30” AND student = “yes” THEN buys_computer = “yes”
IF age = “31…40” THEN buys_computer = “yes”
IF age = “>40” AND leasing_rating = “excellent” THEN buys_computer = “no”
IF age = “>40” AND leasing_rating = “fair” THEN buys_computer = “yes”

Entropy: Used by ID3

- Entropy measures the impurity of S
- S is a set of examples
- p is the proportion of positive examples
- q is the proportion of negative examples

\[
\text{Entropy}(S) = - p \log_2 p - q \log_2 q
\]
Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)
  \[
  \text{SplitInfo}_A(D) = -\sum_{j=1}^{n} \frac{|D_j|}{|D|} \times \log_2 \left( \frac{|D_j|}{|D|} \right)
  \]
  \[
  \text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}(A)}
  \]
- Ex.
  \[
  \text{SplitInfo}_{\text{income}}(D) = -\frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) - \frac{6}{14} \times \log_2 \left( \frac{6}{14} \right) = 1.557
  \]
  \[
  \text{gain\_ratio}(\text{income}) = \frac{0.029}{1.557} = 0.019
  \]
- The attribute with the maximum gain ratio is selected as the splitting attribute

CART (More details in Tutorial)

- If a data set \( D \) contains examples from \( n \) classes, gini index, \( gini(D) \) is defined as
  \[
  gini (D) = 1 - \sum_{j=1}^{n} p_j^2
  \]
  where \( p_j \) is the relative frequency of class \( j \) in \( D \)
- If a data set \( D \) is split on \( A \) into two subsets \( D_1 \) and \( D_2 \), the gini index \( gini(D) \) is defined as
  \[
  gini_A(D) = \frac{|D_1|}{|D|} \times gini(D_1) + \frac{|D_2|}{|D|} \times gini(D_2)
  \]
- Reduction in Impurity:
  \[
  \Delta gini(A) = gini(D) - gini_A(D)
  \]
- The attribute provides the smallest \( gini_{\text{split}}(D) \) (or the largest reduction in impurity) is chosen to split the node (need to enumerate all the possible splitting points for each attribute)
### Example: Various Partition Numbers

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute 1</td>
<td>Yes 4</td>
<td>Yes 4</td>
<td>Yes 2</td>
<td>Yes 2</td>
</tr>
<tr>
<td></td>
<td>No 8</td>
<td>No 8</td>
<td>No 2</td>
<td>No 2</td>
</tr>
<tr>
<td>Total</td>
<td>12</td>
<td>12</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Attribute 2</td>
<td>Yes 2</td>
<td>Yes 2</td>
<td>Yes 2</td>
<td>Yes 2</td>
</tr>
<tr>
<td></td>
<td>No 4</td>
<td>No 4</td>
<td>No 4</td>
<td>No 4</td>
</tr>
<tr>
<td>Total</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Gain</th>
<th>SplitInfo</th>
<th>Gain Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute 1</td>
<td>0.082</td>
<td>1.000</td>
<td>0.082</td>
</tr>
<tr>
<td>Attribute 2</td>
<td>0.082</td>
<td>2.000</td>
<td>0.041</td>
</tr>
</tbody>
</table>

### Example: Unbalanced Partitions

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute 1</td>
<td>Yes 2</td>
<td>Yes 6</td>
</tr>
<tr>
<td></td>
<td>No 4</td>
<td>No 12</td>
</tr>
<tr>
<td>Total</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>Attribute 2</td>
<td>Yes 4</td>
<td>Yes 4</td>
</tr>
<tr>
<td></td>
<td>No 8</td>
<td>No 8</td>
</tr>
<tr>
<td>Total</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>

\[
\text{Gain Info}_1(D) = -\frac{6}{24} \times \log_2 \left(\frac{6}{24}\right) - \frac{18}{24} \times \log_2 \left(\frac{18}{24}\right) = 0.811
\]

\[
\text{Gain Info}_2(D) = -\frac{12}{24} \times \log_2 \left(\frac{12}{24}\right) - \frac{12}{24} \times \log_2 \left(\frac{12}{24}\right) = 1
\]

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Gain</th>
<th>SplitInfo</th>
<th>Gain Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute 1</td>
<td>0.082</td>
<td>0.811</td>
<td>0.101</td>
</tr>
<tr>
<td>Attribute 2</td>
<td>0.082</td>
<td>1</td>
<td>0.082</td>
</tr>
</tbody>
</table>
Ex. D has 9 tuples in buys_computer = “yes” and 5 in “no”

\[ gini(D) = 1 - \left( \frac{9}{14} \right)^2 - \left( \frac{5}{14} \right)^2 = 0.459 \]

Suppose the attribute income partitions D into 10 in D₁: {low, medium} and 4 in D₂

\[ gini_{\text{income}}[\text{low, medium}] (D) = \left( \frac{10}{14} \right) \text{Gini}(D₁) + \left( \frac{4}{14} \right) \text{Gini}(D₂) \]

\[ = \frac{10}{14} \left[ 1 - \left( \frac{7}{10} \right)^2 - \left( \frac{3}{10} \right)^2 \right] + \frac{4}{14} \left[ 1 - \left( \frac{2}{4} \right)^2 - \left( \frac{2}{4} \right)^2 \right] \]

\[ = 0.443 \]

\[ = Gini_{\text{income in } \{\text{low}\}}(D) \]

<table>
<thead>
<tr>
<th>income</th>
<th>yes</th>
<th>no</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Medium</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>high</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Comparing Attribute Selection Measures

The three measures, in general, return good results but

- Information gain:
  - biased towards multivalued attributes

- Gain ratio:
  - tends to prefer unbalanced splits in which one partition is much smaller than the others

- Gini index:
  - biased to multivalued attributes
  - has difficulty when # of classes is large
  - tends to favor tests that result in equal-sized partitions and purity in both partitions
Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on $\chi^2$ test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistics: has a close approximation to $\chi^2$ distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

Underfitting and Overfitting (Homework)

Explain the phenomena of overfitting and underfitting and how to solve them

500 circular and 500 triangular data points.

Circular points:
$0.5 \leq \sqrt{x_1^2 + x_2^2} \leq 1$

Triangular points:
$\sqrt{x_1^2 + x_2^2} > 0.5$ or $\sqrt{x_1^2 + x_2^2} < 1$
Underfitting and Overfitting

**Underfitting**: when model is too simple, both training and test errors are large.

**Overfitting due to Noise**: Decision boundary is distorted by noise point.
Underfitting due to Insufficient Examples

Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region.

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task.

Two approaches to avoid Overfitting

- **Prepruning:**
  - Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
  - Difficult to choose an appropriate threshold

- **Postpruning:**
  - Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
  - Use a set of data different from the training data to decide which is the “best pruned tree”
Scalable Decision Tree Induction Methods

- **ID3**, **C4.5**, and **CART** are not efficient when the training set doesn’t fit the available memory. Instead the following algorithms are used
  - **SLIQ**
    - Builds an index for each attribute and only class list and the current attribute list reside in memory
  - **SPRINT**
    - Constructs an attribute list data structure
  - **RainForest**
    - Builds an AVC-list (attribute, value, class label)
  - **BOAT**
    - Uses bootstrapping to create several small samples

BOAT (for reading)

- **BOAT** (Bootstrapped Optimistic Algorithm for Tree Construction)
  - Use a statistical technique called *bootstrapping* to create several smaller samples (subsets), each fits in memory
  - Each subset is used to create a tree, resulting in several trees
  - These trees are examined and used to construct a new tree $T'$
    - It turns out that $T'$ is very close to the tree that would be generated using the whole data set together
  - Adv: requires only two scans of DB, an incremental alg.
Why decision tree induction in data mining?

- Relatively faster learning speed (than other classification methods)
- Convertible to simple and easy to understand classification rules
- Comparable classification accuracy with other methods