Master of Technology in Knowledge Engineering

# KE4102: Intelligent Systems & Techniques for Business Analytics: Discovering Knowledge from Data using Decision Trees

Charles Pang
Institute of Systems Science
National University of Singapore
E-mail: charlespang@nus.edu.sg

## Course Outline

- Knowledge Acquisition Problem
- Decision Tree Technique
- Building Decision Trees
- Extracting Rules from Decision Trees
- Pros and Cons of Decision Trees

## Knowledge problem

- Banks receive many loan applications that has to be processed for approval
- Each application consists of many inputs such as Age, Gender, Housing, Salary, Credit history, etc.
- Some applications are approved, others are not; some debtors default, others don't
- Banks do not like defaulters – they want to approve only applicants who are not likely to default
- Their task is to **classify applications** into **Approve** or **Reject**

## Knowledge Acquisition

- If you want to build an automatic loan approval system, you need to acquire the knowledge of an experienced Loan Manager
- One option is to **Interview** the loan manager, extract that knowledge, and build a rule-based system
  - What are some of the problems with this approach?
  - Are there other factors involved in the decision making?
- Another option is to make use of all available cases and try to "**learn**" from it
  - This learning process is called **Machine Learning**
  - **Decision Tree** is a machine learning technique

## What are Decision Trees?

- Decision Tree is a simple, but powerful form of **multiple variable analysis** – i.e. problems whose solution (outcome) is the result of the combined effects of multiple input variables or factors.
- Decision Tree analysis provides Data Scientists with the ability to **predict, explain and describe** an outcome
- Decision Tree is the most widely used technique for classification and prediction problems because:
  - Good accuracy can be achieved
  - Tree building is fast and efficient
  - Trees can be converted into Business logic which can be understood by humans

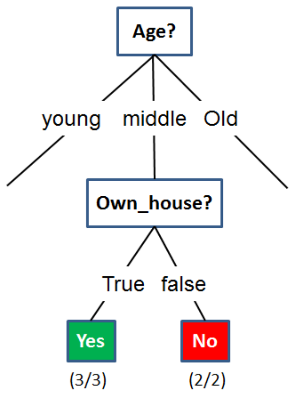
## A Bank Loan Example

| ID | Age | Has_job | Own_house | Credit_rating | Outcome |
|---|---|---|---|---|---|
| 1 | young | False | False | fair | No |
| 2 | young | False | False | good | No |
| 3 | young | True | False | good | Yes |
| 4 | young | True | True | fair | Yes |
| 5 | young | False | False | fair | No |
| 6 | middle | False | False | fair | No |
| 7 | middle | False | False | good | No |
| 8 | middle | True | True | good | Yes |
| 9 | middle | False | True | excellent | Yes |
| 10 | middle | False | True | excellent | Yes |
| 11 | old | False | True | excellent | Yes |
| 12 | old | False | True | good | Yes |
| 13 | old | True | False | good | Yes |
| 14 | old | True | False | excellent | Yes |
| 15 | old | False | False | fair | No |

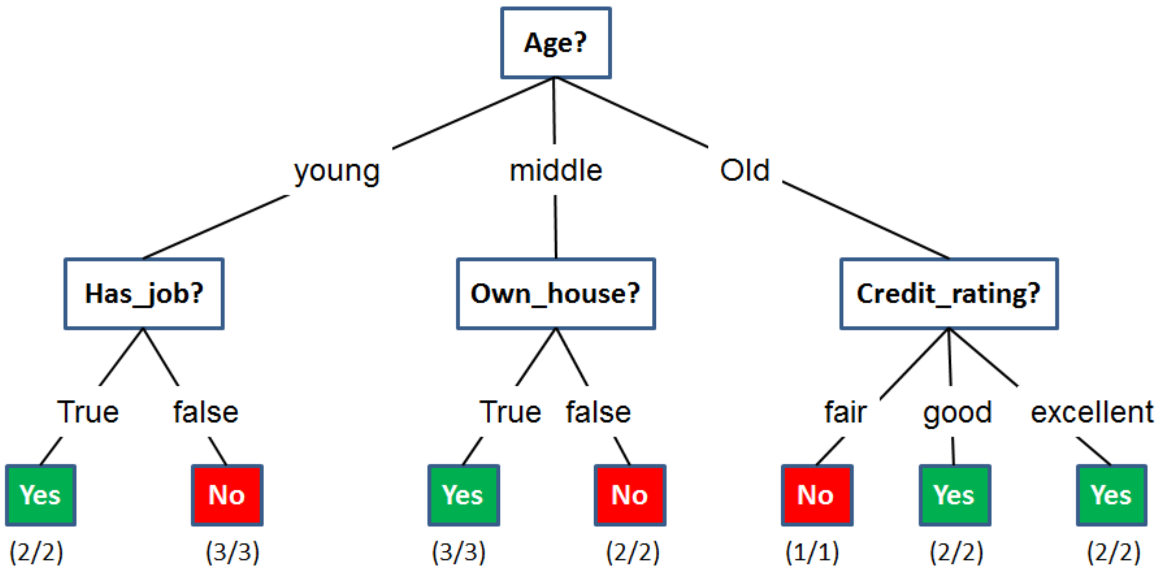
Defaulters => Outcome = No (Reject)

## Finding patterns in data

| ID | Age | Has_job | Own_house | Credit_rating | Outcome |
|---|---|---|---|---|---|
| 1 | young | False | False | fair | No |
| 2 | young | False | False | good | No |
| 3 | young | True | False | good | Yes |
| 4 | young | True | True | fair | Yes |
| 5 | young | False | False | fair | No |
| 6 | middle | False | False | fair | No |
| 7 | middle | False | False | good | No |
| 8 | middle | True | True | good | Yes |
| 9 | middle | False | True | excellent | Yes |
| 10 | middle | False | True | excellent | Yes |
| 11 | old | False | True | excellent | Yes |
| 12 | old | False | True | good | Yes |
| 13 | old | True | False | good | Yes |
| 14 | old | True | False | excellent | Yes |
| 15 | old | False | False | fair | No |

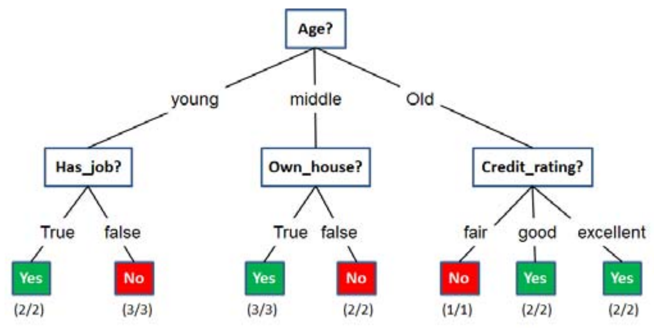


## One possible Decision Tree



## Decision Tree Concepts

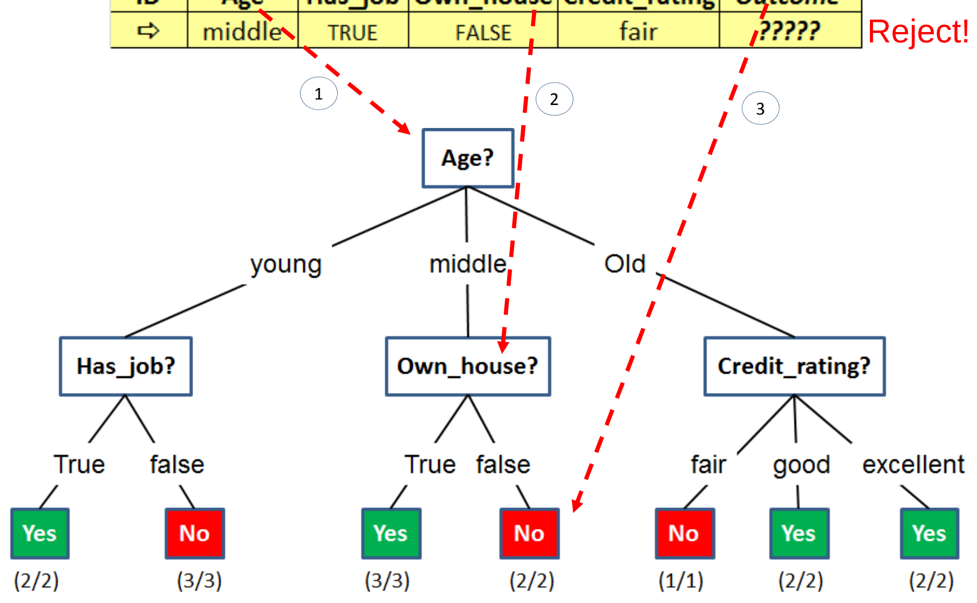
- A decision tree consists of:
  - An **internal node** which performs a test on an attribute
  - A **branch** that represents the outcome of the test
  - A **leaf** node that represents a class label
- A path from root to a leaf node is a **conjunction** (“AND”) of attribute tests
  - If Age=middle AND Own_house=True THEN Approve=Yes



## Decision Trees are used to predict outcome

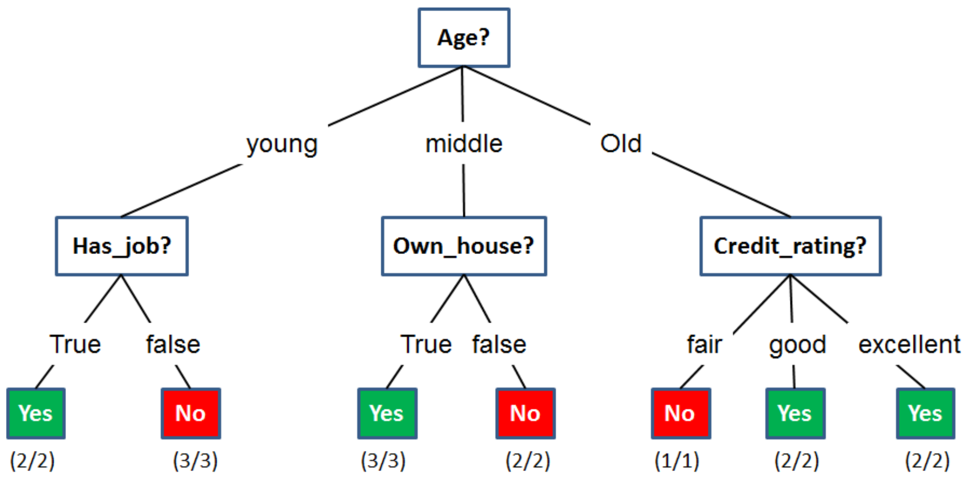
| ID | Age | Has_job | Own_house | Credit_rating | *Outcome* |
|---|---|---|---|---|---|
| ⇨ | middle | TRUE | FALSE | fair | *?????* |

Reject!



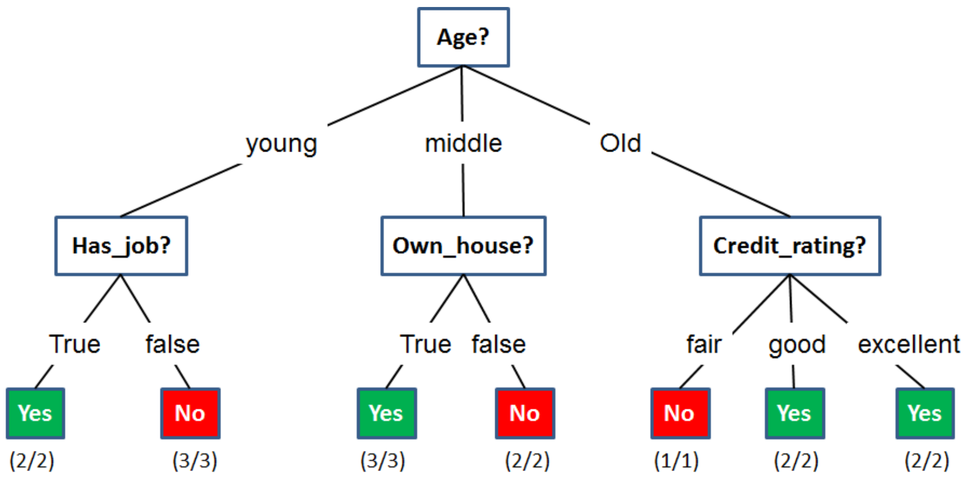
## Patterns uncovered- describe customers

- ✓ Defaulters are young and un-employed
- ✓ Defaulters are also middle-age who don’t own houses
- ✓ Defaulters are also old and don’t have good/excellent credit_ratings



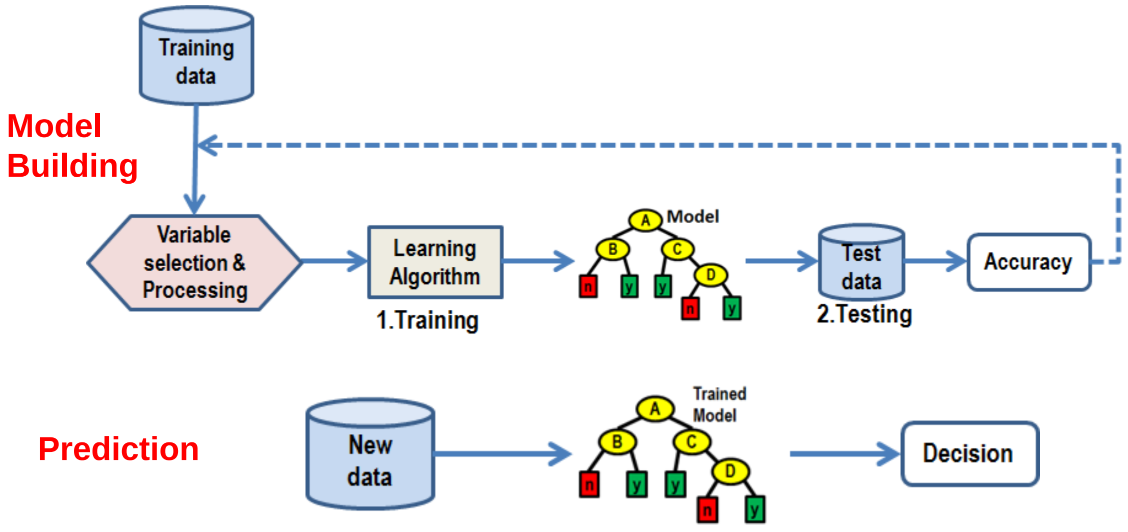
## Patterns uncovered- explain critical factors

- ✓ Age is the key determinant in decision making
- ✓ For middle age customers, house ownership is critical
- ✓ For young customers, employment is critical
- ✓ For old customers, credit-rating is critical



## Decision Tree Modeling

- Decision Tree modeling involves a **model-building** process that produces a tree-like classification model, then using that model for **prediction** of new data



## Decision Tree Modeling (cont'd)

- Given a collection of records (***training set***)
  – Each record contains a set of *attributes*
  – One of the attributes is the ***class*** *(target)*
- Find a ***model*** for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  – A ***test set*** is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

## Decision Tree Learning Algorithms

- Decision Tree models are built (learnt!) using special algorithms:
  - ID3, C4.5, C5.0
  - CART (Classification And Regression Tree)
  - CHAID (CHi-squared Automatic Interaction Detector)
- Each algorithm differs from another in terms of:
  - Best-Split Methods
  - Handling of Numerical/Nominal/ordinal variable
  - Performing Binary or Mulitway splits
  - Handling of missing data
  - Pre-pruning or post-pruning

## Heuristic Search

- Search bias: Search the space of decision trees from simplest to increasingly complex (greedy search, no backtracking, prefer small trees)
- Search heuristics: At a node, select the attribute that is most useful for classifying examples, split the node accordingly
- Stopping criteria: A node becomes a leaf
  - if all examples belong to same class Cj, label the leaf with Cj
  - if all attributes were used, label the leaf with the most common value Ck of examples in the node

## Example Algorithm

Let $\mathbf{D_t}$ be the set of training records at a node ***t***

1. If $\mathbf{D_t}$ contains records that belong to the same class $\mathbf{y_t}$, then ***t*** is the leaf node labeled as $\mathbf{y_t}$
2. If $\mathbf{D_t}$ contains records that belong to more than one class, use an **attribute test** to split the data into smaller subsets. A child node is created for each outcome of the test condition and the records in $\mathbf{D_t}$ are distributed to the children based on the outcomes.
   Recursively apply the procedure to each subset.

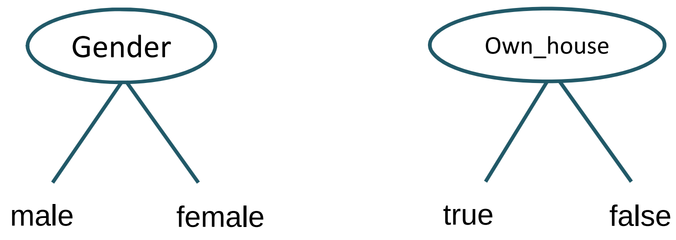
## Attribute test

- Two things are considered during the attribute test for splitting data

1. Split depends on **attribute types**
   – Nominal, Ordinal, Numerical (continuous)
   – Binary or Multi-way splits
2. Split depends on **best way** to split attributes
   – Which attribute do we choose that yields a best-split
   – Aim is to build an "optimal" tree

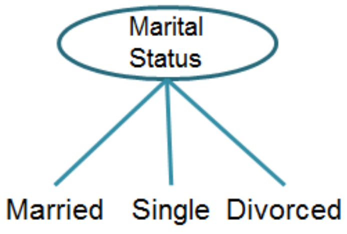
## Splitting Binary Attributes

- There are only two possible outcomes

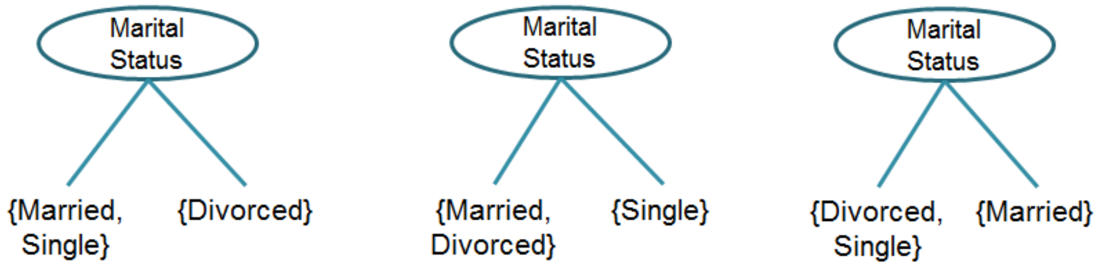


## Splitting Nominal Attributes

- **Multi-way split**: use all distinct attribute values:

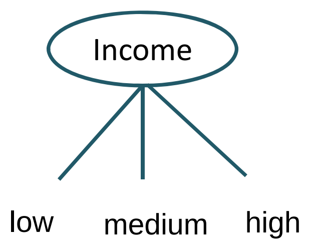


- **Binary split**: divide values into 2 subsets. 3 possible splits:

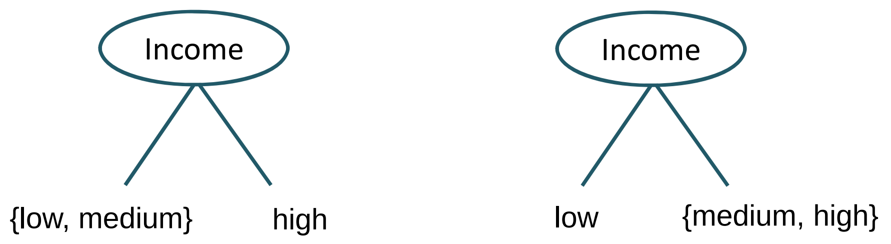


## Splitting Ordinal Attributes

- **Multi-way split**: use all distinct attribute values:

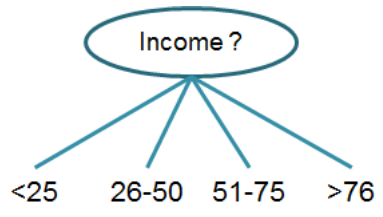


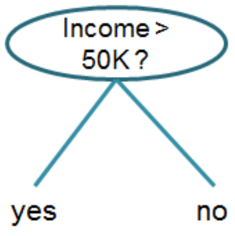
- **Binary split**: divide values into 2 subsets. ONLY 2 possible splits:



## Splitting Numerical Attributes

- Different ways of handling numerical (continuous) attributes
- Discretization
  - Transform into Ordinal attributes
  - Equal Interval Binning: 0-25, 26-50, 51-75, 76-100
  - Equal Frequency Binning: 2-8, 10-44, 68-90
    - Every bin has equal # of records
- Binarization
  - Transform into Binary attributes
  - Split by Mean or Median
    - {X <= Median, X > Median}
    - {X <= Mean, X > Mean}

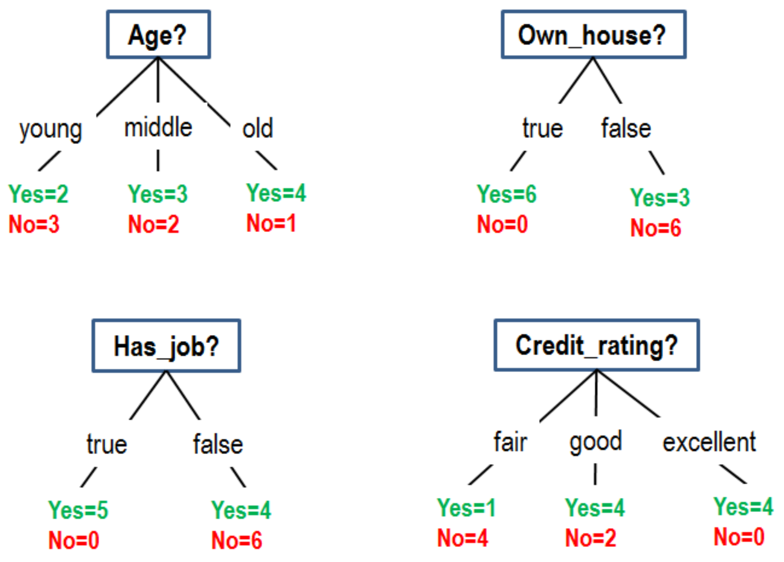




## Determining the best split

- **Which attribute** (age? own_house? has_job?) should be split
- The best attribute is one that best separates the data into groups, where a **single class predominates** (homogeneity= purity) within each group
- Measures of node purity:
  - Information gain (ID3/C4.5/C5.0)
  - Gini index (CART)
  - Chi-square (CHAID)

## Which Attribute do we choose?



- Choose an attribute that gives the best class purity

## Entropy

- Entropy is a measure of ***impurity*** or disorder of data. e.g. Data set *D* with *c number of classes:*

$$\boxed{entropy(D) = -\sum_{i=1}^{C} \Pr(C_i)\cdot\log_2 \Pr(C_i)} \qquad \sum_{i=1}^{C} pr(C_i) = 1$$

- Entropy in binary classification problems

$$\boxed{entropy(D) = -Pr(+)*log_2 Pr(+) - Pr(-)*log_2 Pr(-)}$$

- The entropy = **0** if the data is **pure**
- The entropy = **1** if the data is **impure**

## Properties of Entropy

- The data set D has 50/50: Yes/No

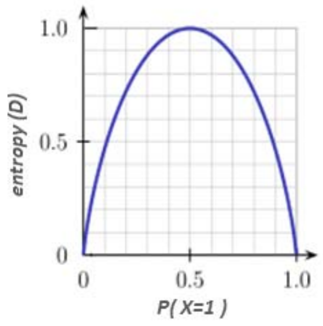
  **entropy(D) = –0.5 x log$_2$0.5 –0.5 x log$_2$0.5 = 1.0**

- The data set D has 100/0: Yes/No

  **entropy(D) = –1 x log$_2$1 –0 x log$_2$0 = 0**

- The data set D has 20/80: Yes/No

  **entropy(D) = –0.2 x log$_2$0.2 –0.8 x log$_2$0.8 = 0.722**



- As data gets **purer**, entropy gets **smaller**

- Entropy is used in the calculation of ***Information Gain***

## Information Gain

- **Information gain** measures the expected reduction in entropy, or impurity
  – Used to minimize the number of tests needed for the classification of a new data
- ***gain*(*D*, *$A_i$*)** = expected reduction in entropy of ***D*** when attribute ***$A_i$*** is selected to branch (i.e. split) the data:

$$gain(D, A_i) = entropy(D) - entropy_{A_i}(D)$$

- We choose the attribute ***$A_i$*** with the ***highest gain*** to branch/split the current tree

## Using Information Gain

- Given a set of examples ***D***, we first compute its entropy:

$$entropy(D) = -\sum_{i=1}^{C} \Pr(C_i) \cdot \log_2 \Pr(C_i)$$

- If we choose attribute ***$A_i$*** with ***v*** values, as the root of the current tree, this will partition ***D*** into ***v*** subsets ***$D_1$*, *$D_2$*** … ***$D_v$*** and … the expected entropy is :

$$entropy_{A_i}(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times entropy(D_j)$$

## Calculating Entropy

$e(D) = -(9/15)*\log_2(9/15) - (6/15)*\log_2(6/15) = 0.971$

$e_{age}(D) = (5/15)*e(young) + (5/15)*e(middle) + (5/15)*e(old) = 0.888$

$e(young) = -(2/5)*\log_2(2/5) - (3/5)*\log_2(3/5)$

$e(middle) = -(3/5)*\log_2(3/5) - (2/5)*\log_2(2/5)$

$e(old) = -(4/5)*\log_2(4/5) - (1/5)*\log_2(1/5)$

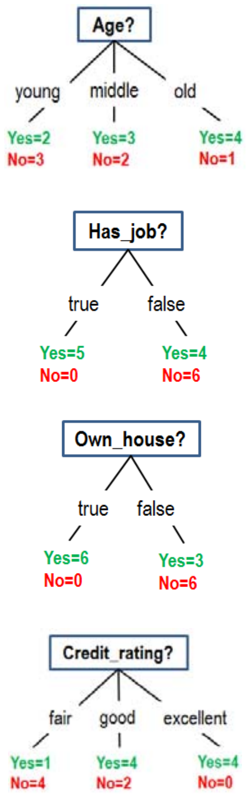
$e_{has\_job}(D) = (5/15)*e(true) + (10/15)*e(false) = 0.647$

$e(true) = -(5/5)*\log_2(5/5) - (0/5)*\log_2(0/5)$

$e(false) = -(4/10)*\log_2(4/10) - (6/10)*\log_2(6/10)$

$e_{own\_house}(D) = 0.551$

$e_{credit\_rating}(D) = 0.607$



## Calculating Gain

***gain(D, age)*** = 0.971 – 0.888 = 0.083

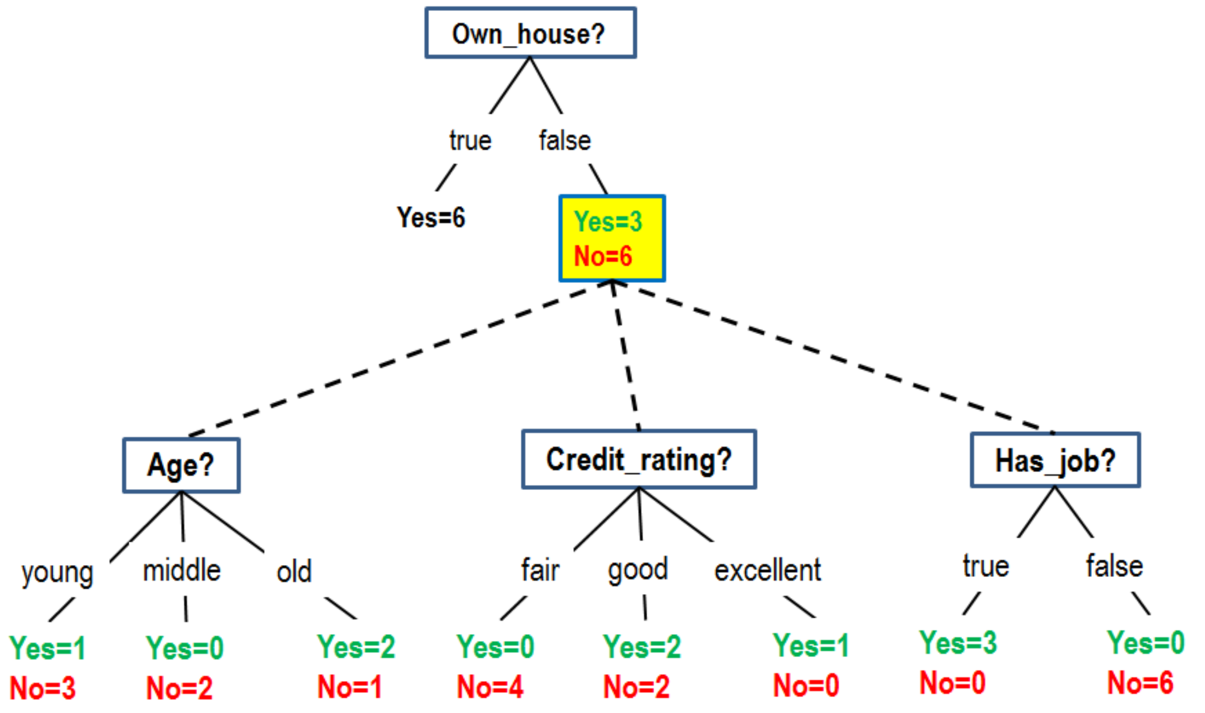
***gain(D, has_job)*** = 0.971 – 0.647 = 0.324

***gain(D, own_house)*** = 0.971 – 0.551 = 0.420

***gain(D, credit_rating)*** = 0.971 – 0.607 = 0.364

- ***Own_house*** yields the most information gain and therefore it will be chosen as the root node

## Deciding on the next node



## Calculate Entropy & Gain

$e(D1) = -(3/9)*\log_2(3/9) - (6/9)*\log_2(6/9) = 0.918$

$e_{age}(D1) = (4/9)*e(young) + (2/9)*e(middle) + (3/9)*e(old) = 0.666$

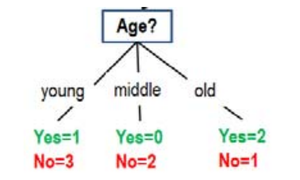
$e(young) = -(1/4)*\log_2(1/4) - (3/4)*\log_2(3/4) = 0.811$

$e(middle) = -(0/2)*\log_2(0/2) - (2/2)*\log_2(2/2) = 0$

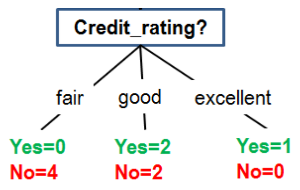
$e(old) = -(2/3)*\log_2(2/3) - (1/3)*\log_2(1/3) = 0.918$

$e_{credit\_rating}(D1) = (4/9)*e(fair) + (4/9)*e(good) + (1/9)*e(excellent) = 0.444$

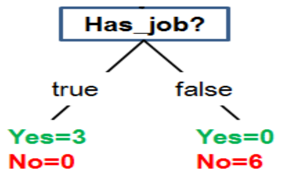
$e_{Has\_job}(D1) = 0$



gain=0.252

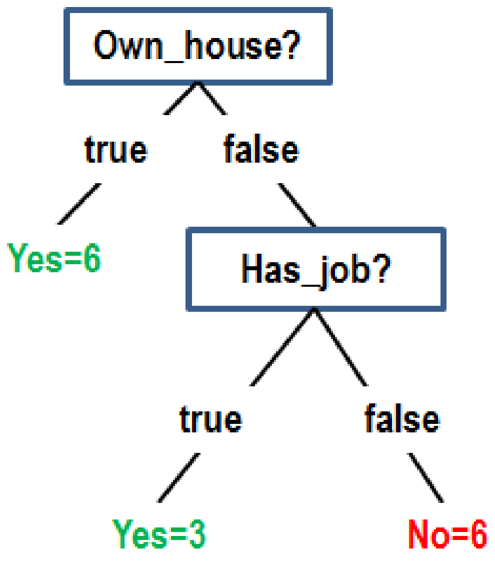


gain=0.474



MAX ! $\rightarrow$ gain=0.918

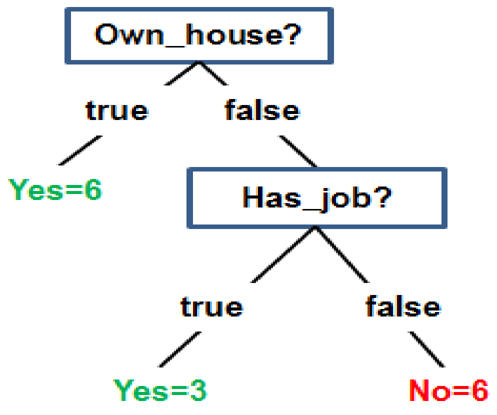
## Final Decision Tree



## Extracting Rules from a Decision Tree

- Decision Trees are essentially decision-making knowledge represented in a tree structure
  - Each branch corresponds to a rule that has its leaf class as the consequent and the conjunction of attributes along the path as the antecedent
- To convert a DT into rules, you need to examine the whole tree
- The rule set can be overly complex in a straightforward conversion
  - Can contain many unnecessary rules
  - With some pre-processing, these rules can be used to build a RBS for decision making

## Rules from DT



- If own_house=true then Yes
- If own_house=false and has_job=true then Yes
- If own_house=false has_job=false then No

## Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the ***best split point*** for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of $a_i$ and $a_{i+1}$
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - S1 is the set of tuples in S satisfying A ≤ split-point, and S2 is the set of tuples in S satisfying A > split-point

## Gini Index: Splitting Criteria in CART

- CART (Classification And Regression Trees)
- If a data set T contains examples from n classes, gini index -- gini(T) is defined as

$$gini\ (T) = 1 - \sum_{j=1}^{n} p_j^2$$

where $p_j$ is the relative frequency of class j in T

gini(T) is minimized if the classes in T are skewed

- After splitting T into two subsets T1 and T2 with sizes N1 and N2, the gini index of the split data is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute providing smallest $gini_{split}$(T) is chosen to split the node

## Computation of Gini Index

- Ex. T has 9 “yes” and 6 “no”

$$gini(T) = 1 - \left(\frac{9}{15}\right)^2 - \left(\frac{6}{15}\right)^2 = 0.48$$

- The attribute “has_job” splits T into 5 in $T_1$: [5 yes, 0 no] and 10 in $T_2$: [4 yes, 6 no]

$$gini_{split}(T) = \left(\frac{5}{15}\right) Gini\ (T_1) + \left(\frac{10}{15}\right) Gini\ (T_2)$$

$$= \frac{5}{15}\left(1 - (\frac{5}{5})^2 - (\frac{0}{5})^2\right) + \frac{10}{15}\left(1 - (\frac{4}{10})^2 - (\frac{6}{10})^2\right) = 0.32$$

## Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Pre-pruning (forward pruning): stop growing the tree e.g.
    - When data split not statistically significant
    - Too few examples are in a split
  - 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"
- Forward pruning considered inferior

## Points to consider for using Decision Trees

- Variable Preparation
  - Must have a target variable
  - Select important variables
  - Transform variables to facilitate tree growing
- Bin continuous attributes
  - Define new discrete values (0-10,10-20,..) for numerical attributes to reduce noise and improve prediction accuracy
- Examine the missing values
  - Impute missing values before using decision trees analysis
  - Know how the algorithm handles missing values
- Examine the quality of trees
  - A good tree generalizes well
  - Study the confusion matrix for accuracy rates

## Advantage of DT

- Requires simple data preparation
- Is a multi-variable analysis technique
- Able to handle both numerical and categorical data
- Trees can be visualized - simple to understand
- “white box” model - Situation that are observed in the model can be easily explained by boolean logic
- Complexity of using the tree to predict/classify = O(log n)
- Model can be validated using statistical tests to show its reliability and accuracy

## Problems with DT

- Over-specialization (“overfitting”)
  – One branch for each value of the attribute
- Trees can be very large and complex
  – Confusing and hard to understand
  – Attributes may be fragmented across the tree structure
- Difficult to understand a part of the tree without reference to the whole
- Learning an Optimal DT is an NP-Complete problem
  – Local optimum achieved using heuristic greedy algorithm

## Decision Tree Summary

- Decision Trees
  - splits – binary, multi-way
  - splitting criteria – info gain, gini, …
  - pruning
  - rule extraction from trees
- There are many other attribute selection criteria!

  (But almost no difference in accuracy of result.)

## References

Last

- C5.0/C4.5
  - http://www.rulequest.com/
- CART
  - www.SalfordSystems.com
- Machine learning / Tom Mitchell, McGraw Hill, 1997
- *Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey*, Sreerama K. Murthy, Data Mining and Knowledge Discovery, 2. 345-389 (1998) Kluwer Academic Publishers.
- *Classification and Regression Trees*, Leo Breiman, Jerome Friedman, Richard Olshen and Charles Stone, 1984, Wadsworth Int. Group.