Dr. Azadeh Mohammadi

Lecturer in Data Science

Classification

Dr. Azadeh Mohammadi

School of Science, Engineering & Environment
University of Salford

Learning outcome

- Describing the basic principles of classification
- Explaining KNN algorithm
- Explaining decision tree algorithm
- Analysing the classification performance and comparing the results

Classification

- Classification is a supervised method
  - The training data (observations) are accompanied by **labels** indicating the class of the observations

- It creates a model based on the training data

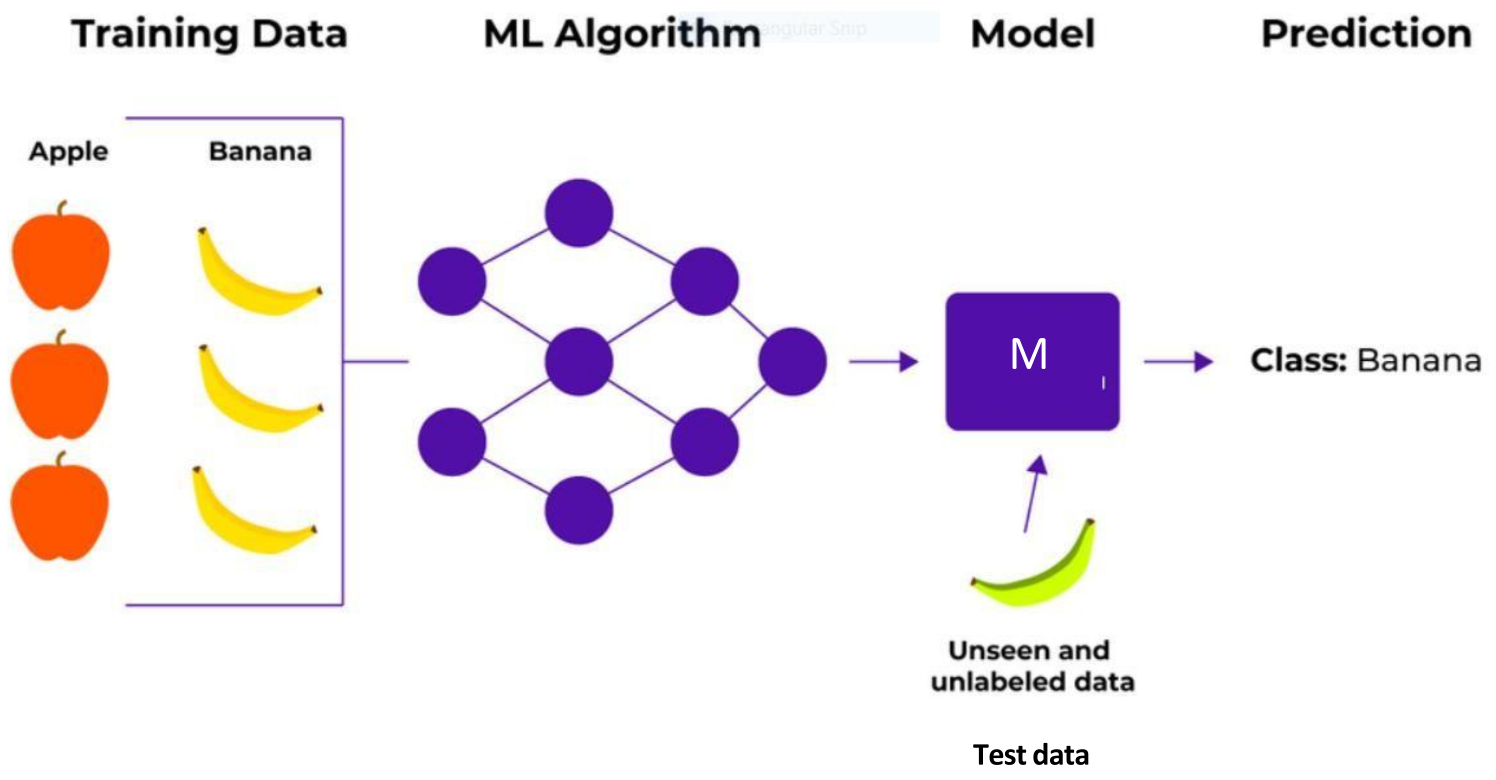
- Classification model (classifier) predicts categorical class labels for new data (based on the model which is trained on the training set)

Classification

- Human can learn through examples
  - It is difficult for a child to differentiate between apple and orange. When you constantly show them pictures and the real fruits, they will be able to identify them correctly.

Classification

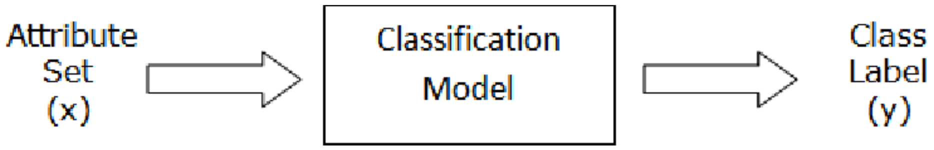
color	shape	texture	Has core	label
Red	Round	smooth	Yes	Apple
Yellow	Oval	Smooth	No	Banana
Green	Round	smooth	Yes	Apple
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…



Source: https://neurospace.io/blog/2020/08/what-is-supervised-learning/

Classification

- Classification: Given a collection of records (training set), find a model for class attribute as a function of the values of other attributes (learn a model for discriminating between records of different classes)



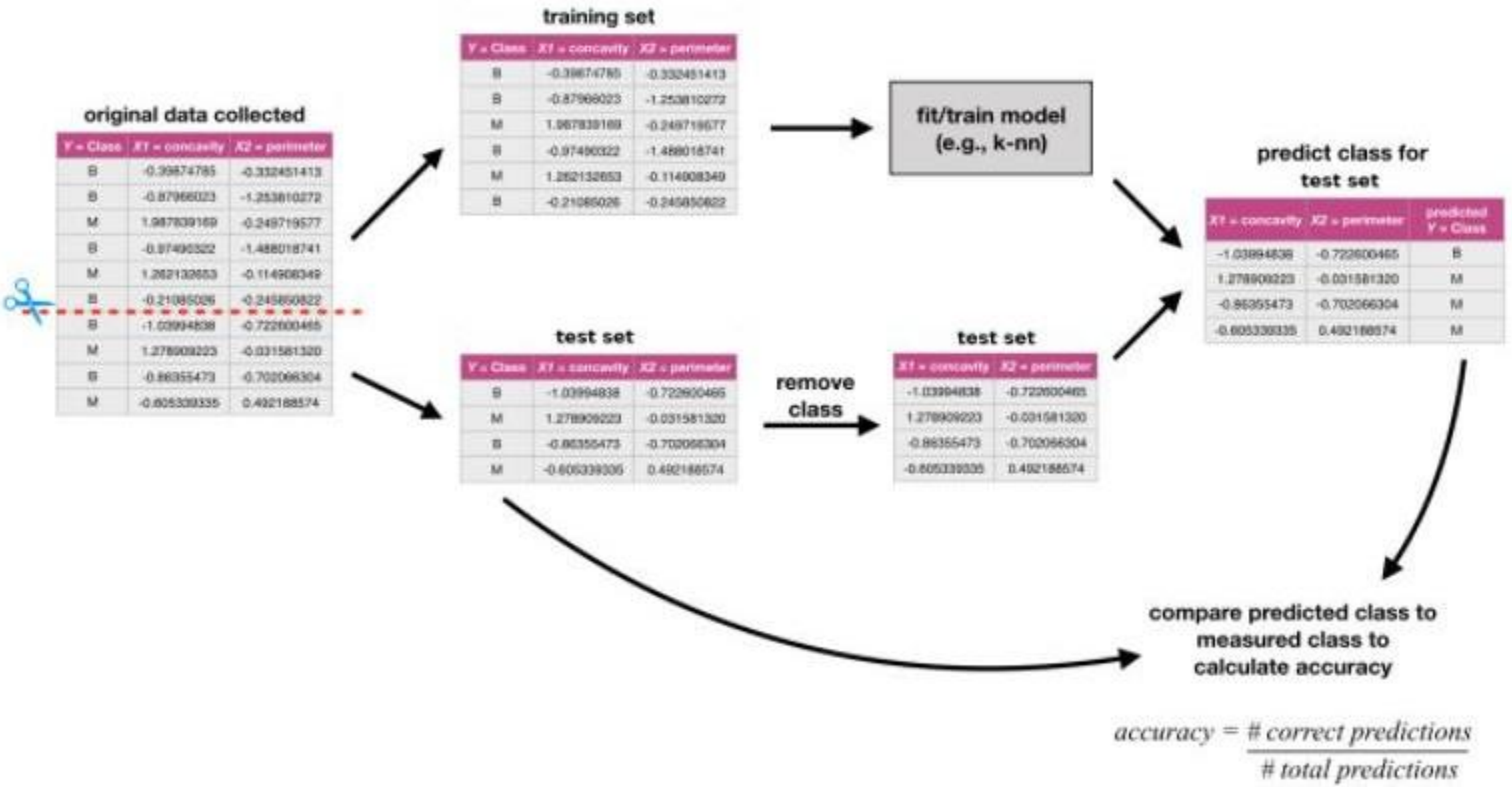
- Goal: previously unseen records should be assigned to a class as accurately as possible.

- A test set is used to determine the accuracy of the model.

Classification

- We usually split our dataset into two partitions:
  - Training set:
    - Each tuple/sample in training set has a set of features (attributes) and a class label attribute (Records with known class labels)
    - Training set is used for Model construction
    - The model is represented as classification rules, decision trees, mathematical formulas, …
  - Test set:
    - Test set is used to evaluate the model
    - We apply our model on the test set to predict their class label and compare it with main labels
    - Test set is independent of training set
- The classification model is applied to new records with unknown class labels

Classification



Applications

- Credit/loan approval: if a loan application is safe or risky
- Medical diagnosis: if a tumor is cancerous or benign
- Fraud detection: if a transaction is fraudulent
- Web page classification: which topic web page belongs (finance, weather, entertainment, sports, etc)

Classification algorithms

- There are different methods for classification:
  - Decision tree
  - KNN
  - Support Vector Machines
  - Neural Networks
  - Deep Learning
  - …

K-Nearest Neighbor (KNN)

- The KNN algorithm algorithm assumes that similar things exist in close proximity.

- It is a **Lazy** classification method
  - Lazy Learning: The model is not learned using training data in advance; instead, the learning process is deferred until a prediction is requested for a new instance

K-Nearest Neighbor (KNN)

- Nearest neighbors are those data points that have minimum distance in feature space from our new data point (test data)

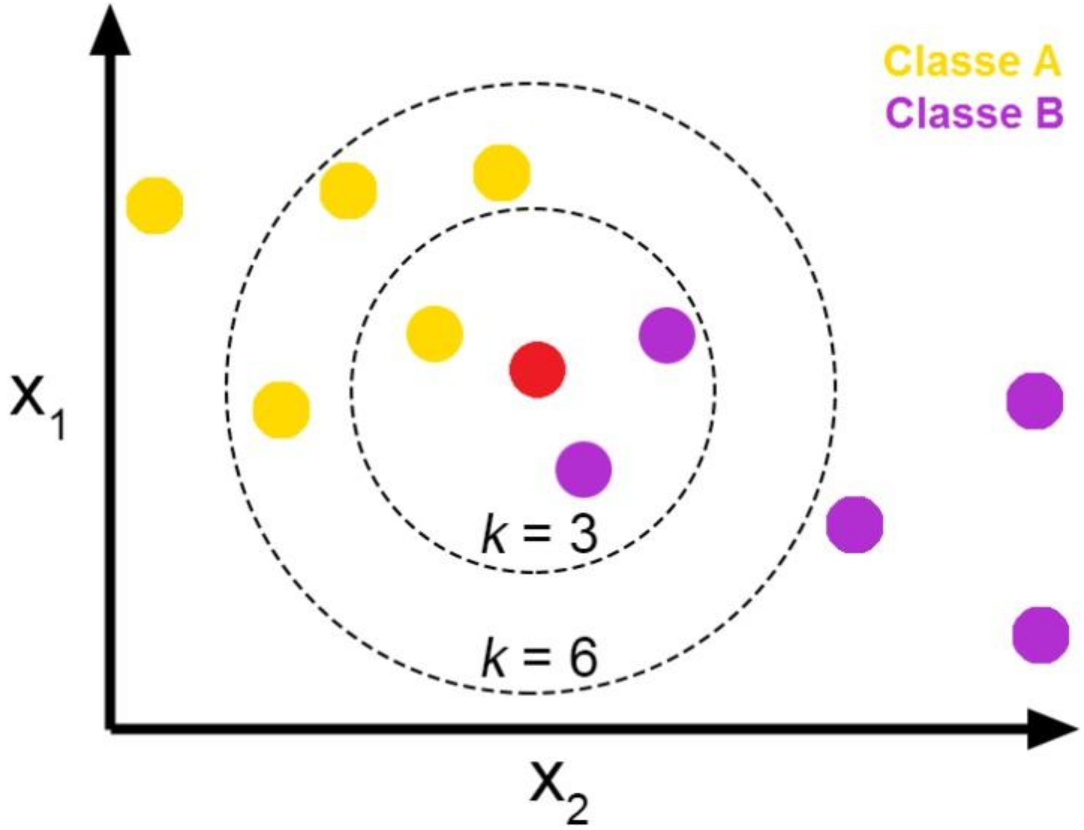
- **K** is **the number of data points we consider for neighborhood**

- Therefore, **distance metric** and **K value** are two important considerations while using the KNN algorithm

K-Nearest Neighbor (KNN)

- KNN algorithm:
  - Load the training data
  - Select the number K of the neighbors
  - Predict a class value for new data X:
    - Calculate distance(X, Xi) based on chosen distance metric, where X= new data point, Xi= training data samples, i=1,2,3,….,n.
    - Sort these distances in increasing order.
    - From this sorted list, select the top ‘K’ rows (K neighbors with least distance)
    - Find the most frequent class from these chosen ‘K’ rows (major voting). This will be your predicted class

K-Nearest Neighbor (KNN)



K-Nearest Neighbor (KNN)

- Choosing K:
  - If the problem is a binary classification, we usually make K an odd number to have a tiebreaker
  - If we keep the value of k low, we risk ourselves of overfitting (the model can't generalize well), while if we keep the value of k high, we risk ourselves of underfitting
    - **Overfitting** occurs when a model fits closely to the peculiarities of the training set but is not able to generalize on new data
    - **Underfitting** occurs when a model is too simple and fails even on the training set

K-Nearest Neighbor (KNN)

- Advantages of KNN Algorithm:
  - It is simple to implement
  - It is robust to the noisy training data
  - It can be more effective if the training data is large
  - There's no need to build a model, tune several parameters, or make additional assumptions
  - The algorithm is versatile. It can be used for classification, regression, and missing value estimation

K-Nearest Neighbor (KNN)

- Disadvantages of KNN Algorithm:
  - Always needs to determine the value of K which may be complex some time
  - The computation cost is high because of calculating the distance between the data points for all the training samples
  - The algorithm becomes significantly slower as the number of examples and/or variables increase

Decision tree

- Many Decision tree Algorithms:
  - ID3
  - C4.5
  - CART
  - SLIQ
  - SPRINT
  - …

Decision tree

- Decision tree: Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive 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
  - Attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

Decision tree

- Conditions for stopping partitioning

  - All samples for a given node belong to the same class

  or

  - There are no remaining attributes for further partitioning (majority voting is employed for classifying the leaf)

  or

  - There are no samples left

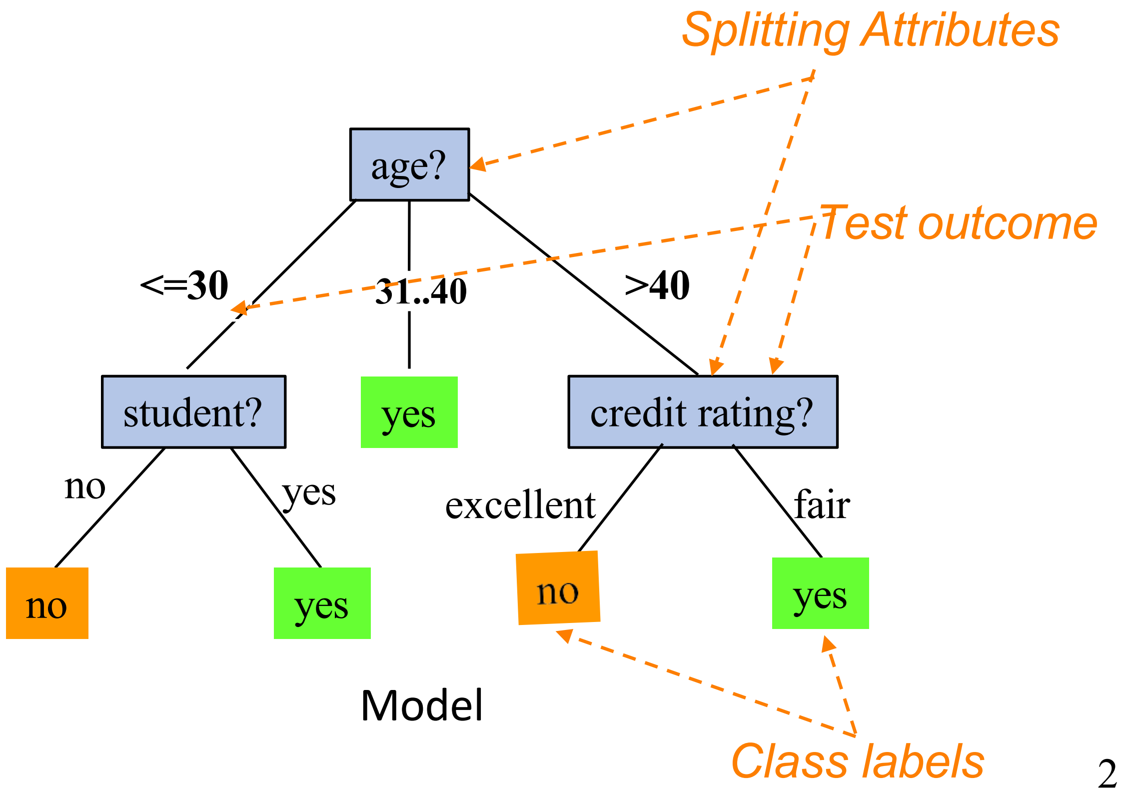
Decision tree

- Internal node denotes a decision node (splitting attributes)
- Branch shows the values of the attribute
- Leaf nodes represent class labels or class distribution

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

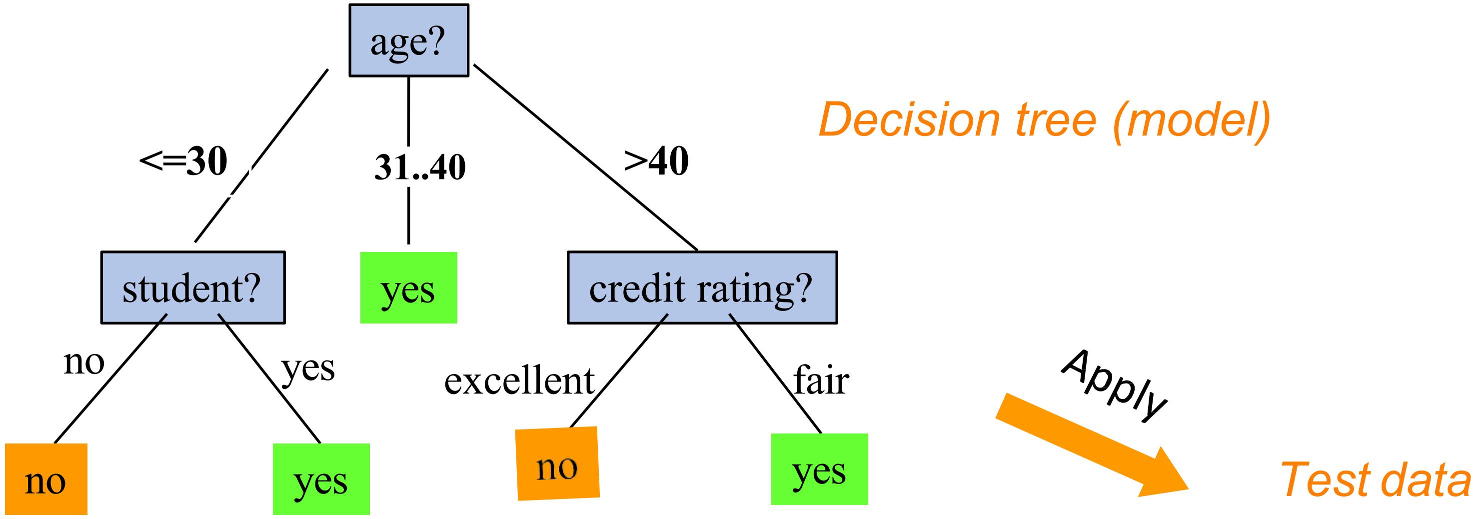
Training dataset

train



Model

Decision tree



age	Income	Student	credit_rating	buys_compter
<=30	No	Yes	fair	?

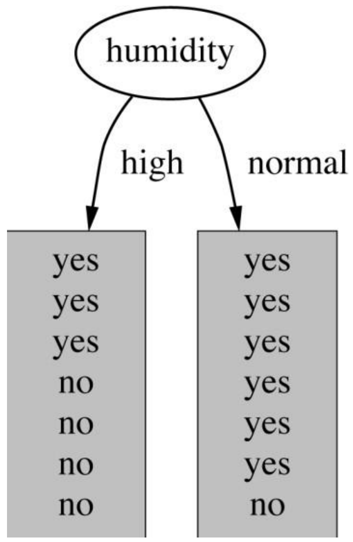
Decision tree

- Important aspects:

  - Determine how to split the records (best split)
  - Determine when to stop splitting
  - How to Classify a leaf node

Decision tree

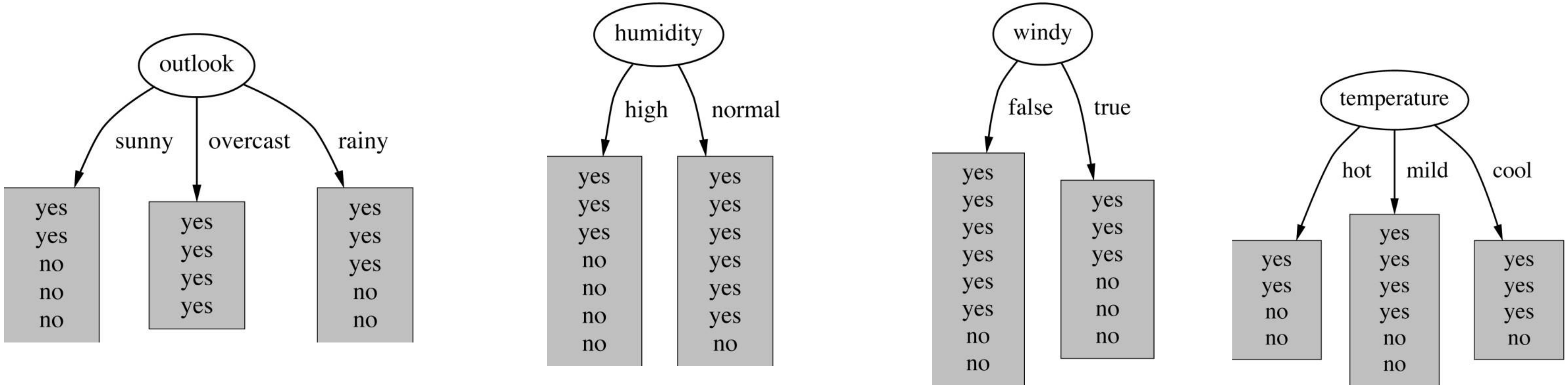
- How to split datasets?
  - Select an attribute as the <u>decision node</u> and partition dataset based on different values of that node



Day	Outlook	Temp.	Humidity	Wind	Play Tennis
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	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	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

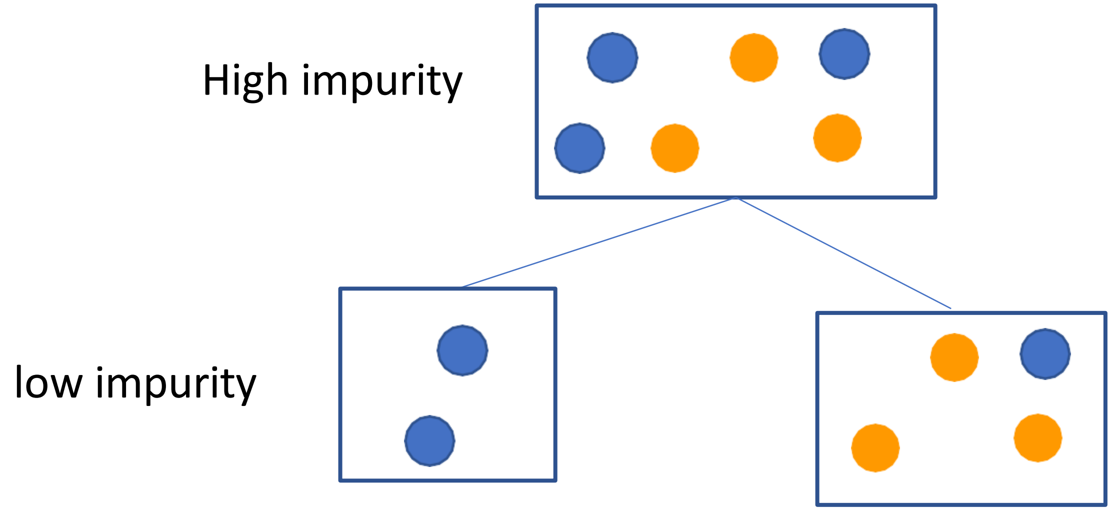
Decision tree

- How to select the decision node?



Decision tree

- How to select the decision node?
  - Choose the attribute that decrease impurity more (information gain)

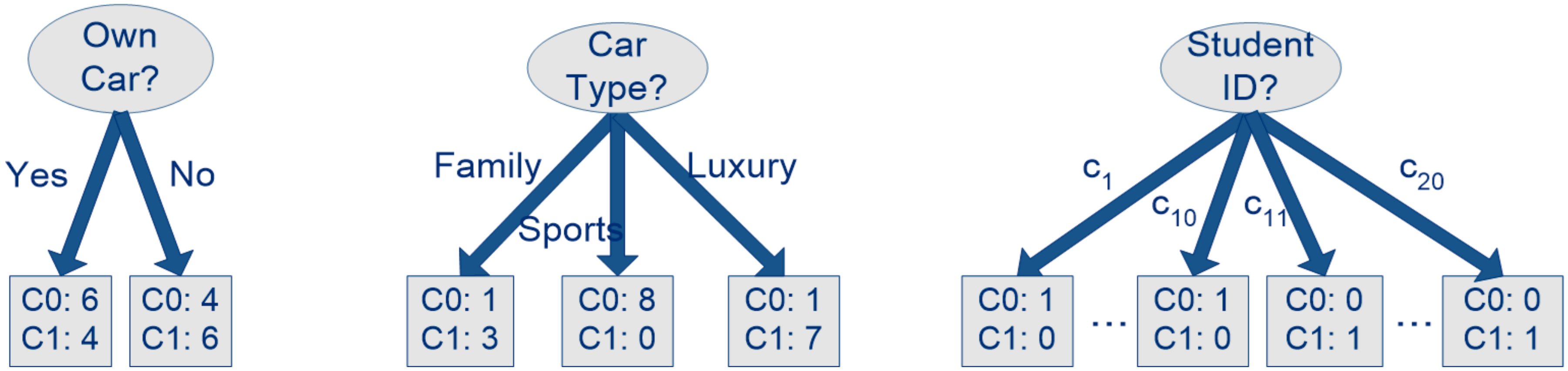


  - **Entropy** shows the **impurity** in a dataset. When set of object is pure, entropy is zero. When we have maximum impurity entropy is one.

Decision tree

- How to determine the Best Split?

Before Splitting: 10 records of class 0,
10 records of class 1



Which test condition is the best?

Decision tree

- Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

C0: 5
C1: 5

Non-homogeneous,

High degree of impurity

C0: 9
C1: 1

Homogeneous,

Low degree of impurity

Decision tree

- Entropy at a given node t: $\boxed{Entropy(t) = -\sum_j p(j\,|\,t)\log_2 p(j\,|\,t)}$
  - $p(j \mid t)$ is the relative frequency of class j at node t
  - Measures homogeneity of a node
    - Minimum (0) when all records belong to one class
    - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information (n_c is the number of class)

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Decision tree

- 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)

Decision tree

- ID3 algorithm:
  - Figure out the best feature to split by using information gain
  - Add this node to the tree
  - Partition the dataset using this attribute
  - For each partition, grow branches from this node
  - Recursively repeat the process for each of these branches using the remaining partition of the dataset

Decision tree

- Stop the recursion and construct a leaf node when:

  - All of the instances in the remaining dataset have the same classification class label
    - Create a leaf node with that classification as its label

  or
  - The set of features left to check is empty
    - Create a leaf node with the majority class of the dataset as its classification

  or
  - The remaining dataset is empty
    - Create a leaf note one level up (parent node), with the majority class

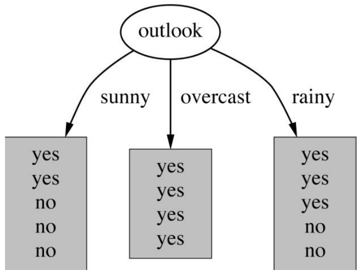
Decision tree

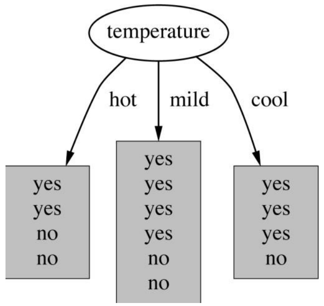
- Example:

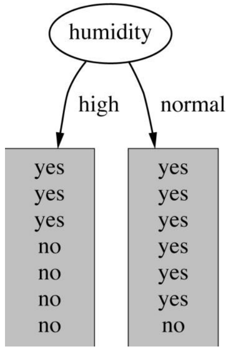
Day	Outlook	Temp	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	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

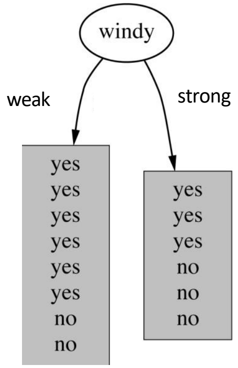
Which attribute should be selected as root?

- The one with the greatest information gain
- We should calculate the information gain for each case







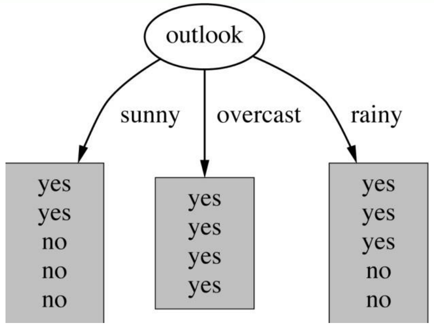


Decision tree

Day	Outlook	Temp	Humidity	Wind	Play Tennis
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

- Entropy of entire dataset

$S = [9+, 5-]$ $\quad Entropy(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$



- Entropy of sunny partition

$S_{Sunny} \leftarrow [2+, 3-]$ $\quad Entropy(S_{Sunny}) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$

- Entropy of overcast partition

$S_{Overcast} \leftarrow [4+, 0-]$ $\quad Entropy(S_{Overcast}) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0$

- Entropy of rainy partition

$Entropy(S_{Rain}) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$

Decision tree

Day	Outlook	Temp	Humidity	Wind	Play Tennis
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

outlook:

$S = [9+, 5-]$ $\quad Entropy(S) = -\frac{9}{14}log_2\frac{9}{14} - \frac{5}{14}log_2\frac{5}{14} = 0.94$

$S_{Sunny} \leftarrow [2+, 3-]$ $\quad Entropy(S_{Sunny}) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.971$

$S_{Overcast} \leftarrow [4+, 0-]$ $\quad Entropy(S_{Overcast}) = -\frac{4}{4}log_2\frac{4}{4} - \frac{0}{4}log_2\frac{0}{4} = 0$

$S_{Rain} \leftarrow [3+, 2-]$ $\quad Entropy(S_{Rain}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.971$

$$Gain\ (S, Outlook) = Entropy(S) - \sum_{v \in \{Sunny, Overcast, Rain\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Outlook)$$

$$= Entropy(S) - \frac{5}{14}Entropy(S_{Sunny}) - \frac{4}{14}Entropy(S_{Overcast})$$

$$-\frac{5}{14}Entropy(S_{Rain})$$

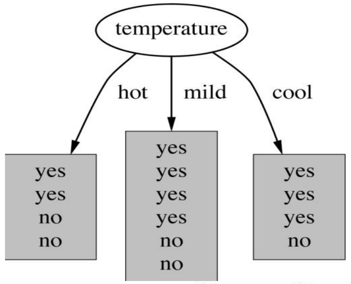
$$Gain(S, Outlook) = 0.94 - \frac{5}{14}0.971 - \frac{4}{14}0 - \frac{5}{14}0.971 = 0.2464$$

Decision tree

- We calculate the information gain of spitting based on other attributes

Day	Outlook	Temp	Humidity	Wind	Play Tennis
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

Temp:



$S = [9+, 5-]$ $\quad Entropy(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$

$S_{Hot} \leftarrow [2+, 2-]$ $\quad Entropy(S_{Hot}) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1.0$

$S_{Mild} \leftarrow [4+, 2-]$ $\quad Entropy(S_{Mild}) = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.9183$

$S_{Cool} \leftarrow [3+, 1-]$ $\quad Entropy(S_{Cool}) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.8113$

$$Gain\,(S, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Temp)$$

$$= Entropy(S) - \frac{4}{14} Entropy(S_{Hot}) - \frac{6}{14} Entropy(S_{Mild})$$

$$- \frac{4}{14} Entropy(S_{Cool}) \quad = 0.0289$$

Decision tree

Day	Outlook	Temp	Humidity	Wind	Play Tennis
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

Humidity:

$$S = [9+, 5-] \qquad Entropy(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

$$S_{High} \leftarrow [3+, 4-] \qquad Entropy(S_{High}) = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.9852$$

$$S_{Normal} \leftarrow [6+, 1-] \qquad Entropy(S_{Normal}) = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.5916$$

$$Gain(S, Humidity) = Entropy(S) - \sum_{v \in \{High, Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Humidity)$$

$$= Entropy(S) - \frac{7}{14} Entropy(S_{High}) - \frac{7}{14} Entropy(S_{Normal})$$

$$Gain(S, Humidity) = 0.94 - \frac{7}{14} 0.9852 - \frac{7}{14} 0.5916 = 0.1516$$

Decision tree

Day	Outlook	Temp	Humidity	Wind	Play Tennis
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

Wind:

$S = [9+, 5-]$ $\quad Entropy(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$

$S_{Strong} \leftarrow [3+, 3-]$ $\quad Entropy(S_{Strong}) = 1.0$

$S_{Weak} \leftarrow [6+, 2-]$ $\quad Entropy(S_{Weak}) = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} = 0.8113$

$$Gain\ (S, Wind) = Entropy(S) - \sum_{v \in \{Strong, Weak\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S, Wind) = Entropy(S) - \frac{6}{14} Entropy(S_{Strong}) - \frac{8}{14} Entropy(S_{Weak})$$

$$= 0.94 - \frac{6}{14} 1.0 - \frac{8}{14} 0.8113 = 0.0478$$

Decision tree

Day	Outlook	Temp	Humidity	Wind	Play Tennis
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

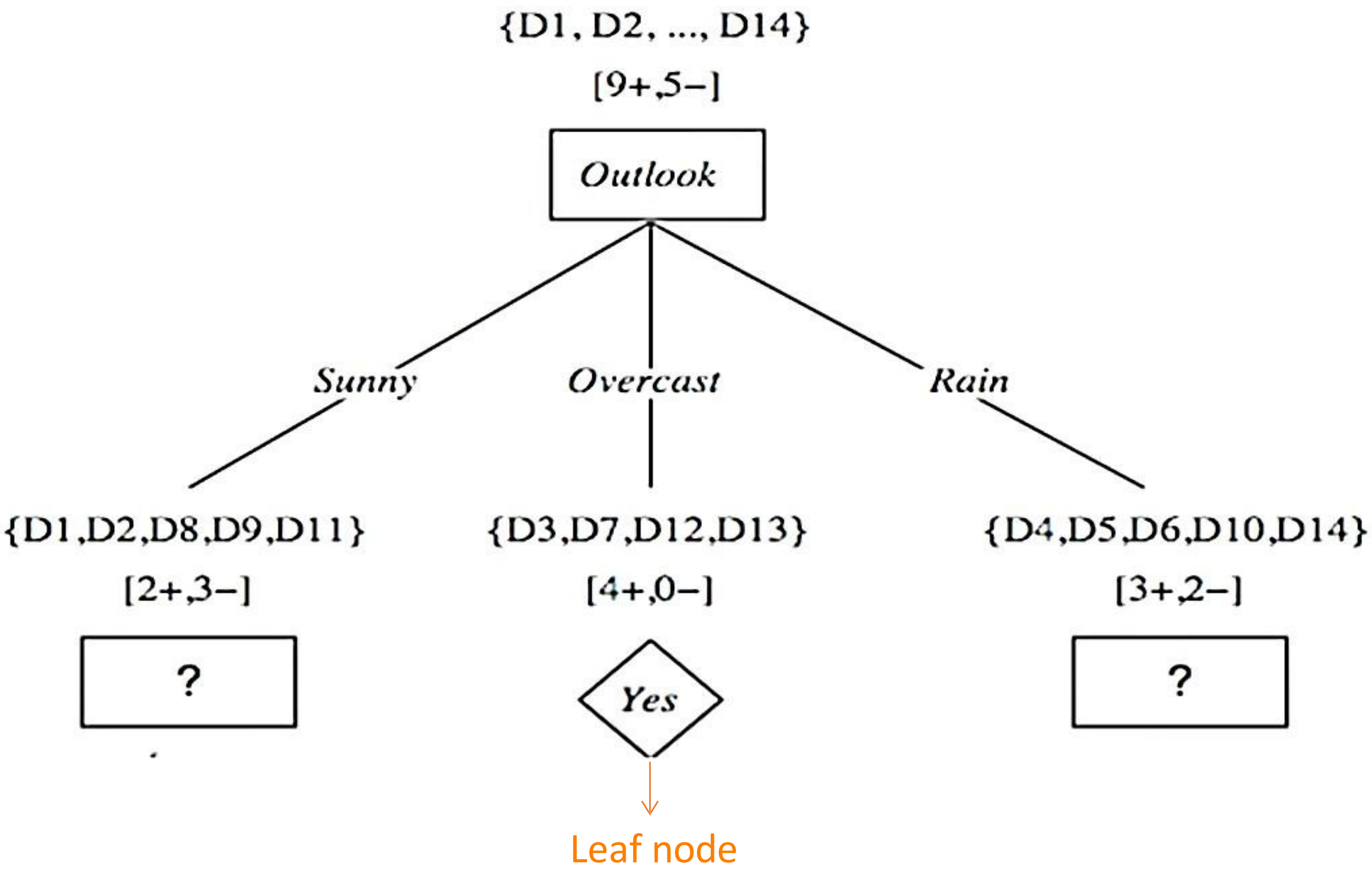
$$Gain(S, Outlook) = 0.2464$$ → Max info gain

$$Gain(S, Temp) = 0.0289$$

$$Gain(S, Humidity) = 0.1516$$

$$Gain(S, Wind) = 0.0478$$

Decision tree



Decision tree

- Continuing left hand side branch:

Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

Temp:

$S_{Sunny} = [2+, 3-]$ $\quad Entropy(S_{Sunny}) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.97$

$S_{Hot} \leftarrow [0+, 2-]$ $\quad Entropy(S_{Hot}) = 0.0$

$S_{Mild} \leftarrow [1+, 1-]$ $\quad Entropy(S_{Mild}) = 1.0$

$S_{Cool} \leftarrow [1+, 0-]$ $\quad Entropy(S_{Cool}) = 0.0$

$$Gain\ (S_{Sunny}, Temp) = Entropy(S) - \sum_{v \in \{Hot, Mild, Cool\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, Temp)$$

$$= Entropy(S) - \frac{2}{5}Entropy(S_{Hot}) - \frac{2}{5}Entropy(S_{Mild})$$

$$-\frac{1}{5}Entropy(S_{Cool})$$

$$Gain(S_{sunny}, Temp) = 0.97 - \frac{2}{5}0.0 - \frac{2}{5}1 - \frac{1}{5}0.0 = 0.570$$

Decision tree

- Continuing left hand side branch:

Humidity:

Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

$S_{Sunny} = [2+, 3-]$ $\quad Entropy(S) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.97$

$S_{high} \leftarrow [0+, 3-]$ $\quad Entropy(S_{High}) = 0.0$

$S_{Normal} \leftarrow [2+, 0-]$ $\quad Entropy(S_{Normal}) = 0.0$

$$Gain\,(S_{Sunny}, Humidity) = Entropy(S) - \sum_{v\in\{High,Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(S_{Sunny}, Humidity) = Entropy(S) - \frac{3}{5}Entropy(S_{High}) - \frac{2}{5}Entropy(S_{Normal})$$

Decision tree

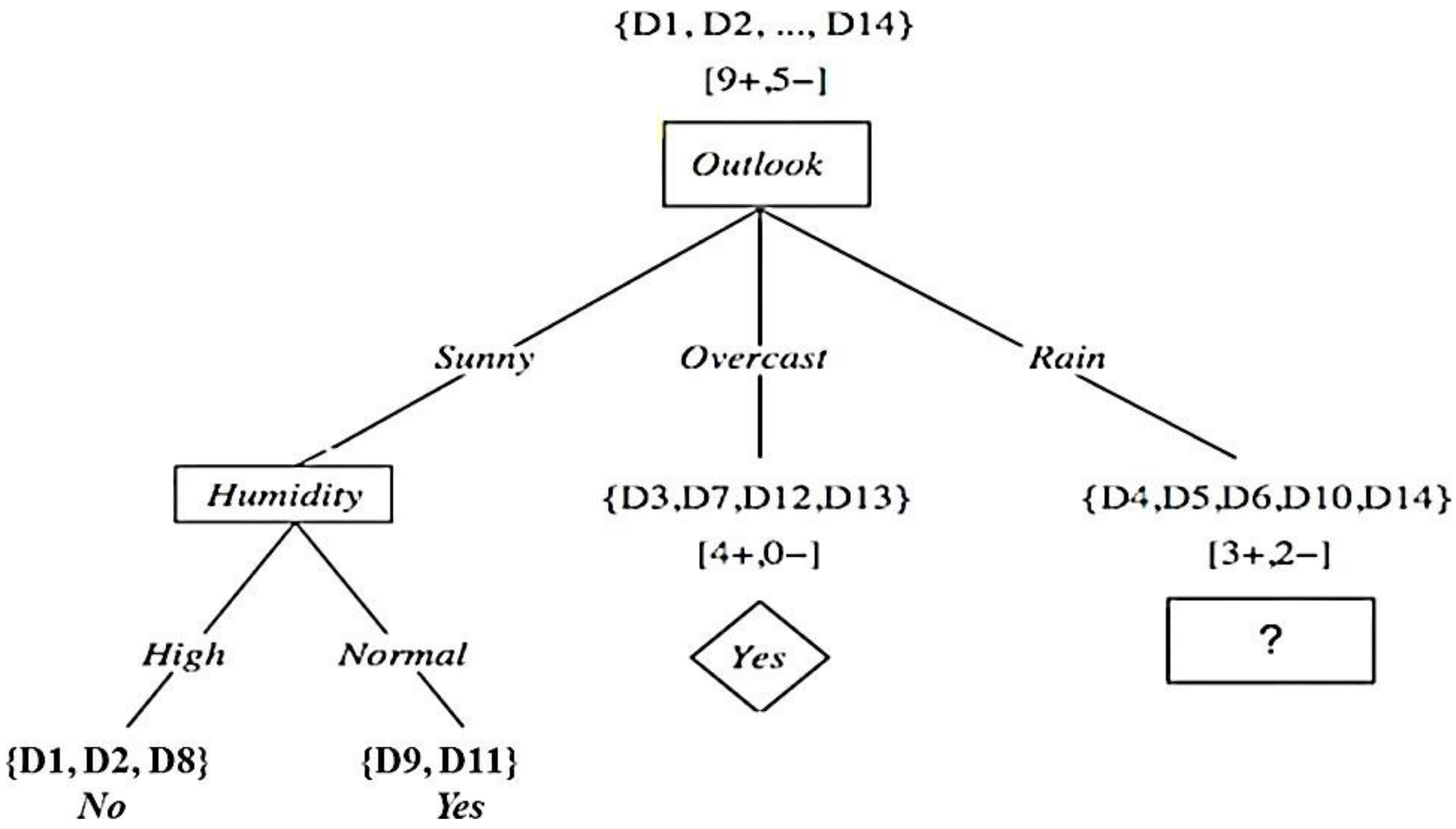
Day	Temp	Humidity	Wind	Play Tennis
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

$$Gain(S_{sunny}, Temp) = 0.570$$

$$Gain(S_{sunny}, Humidity) = 0.97$$ → Max info gain

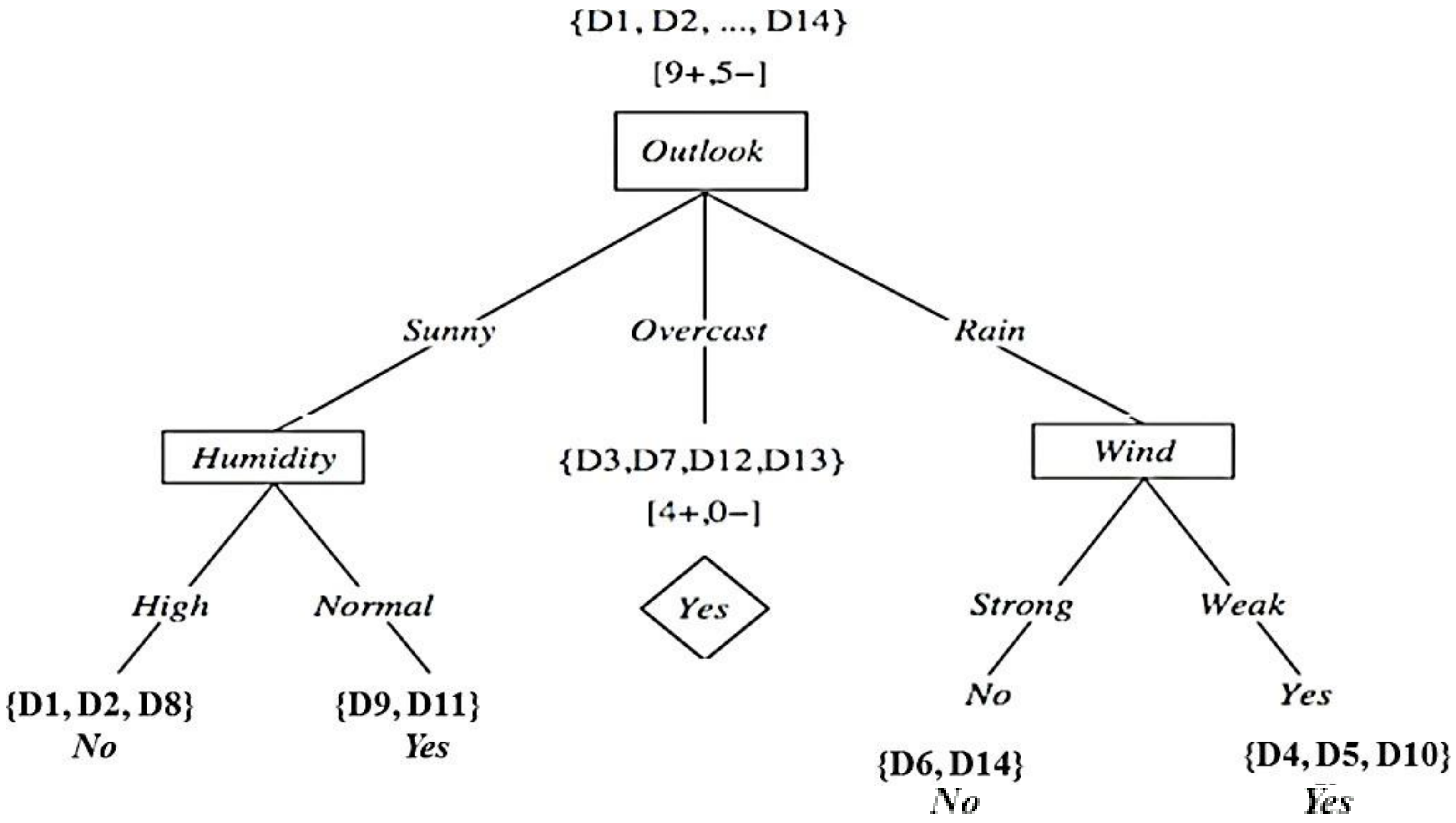
$$Gain(S_{sunny}, Wind) = 0.0192$$

Decision tree



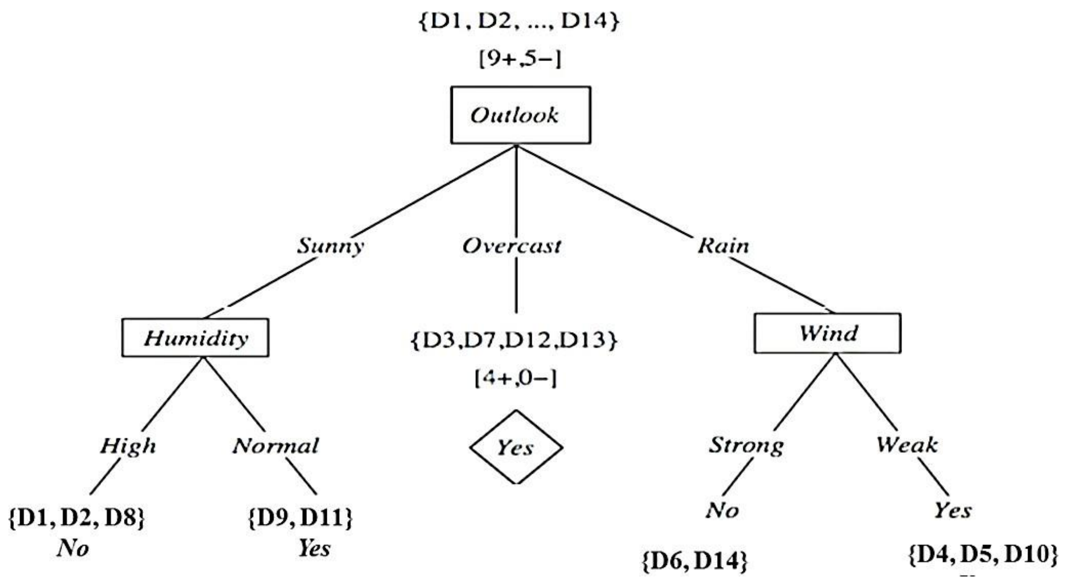
Decision tree

- Final decision tree:



Decision tree

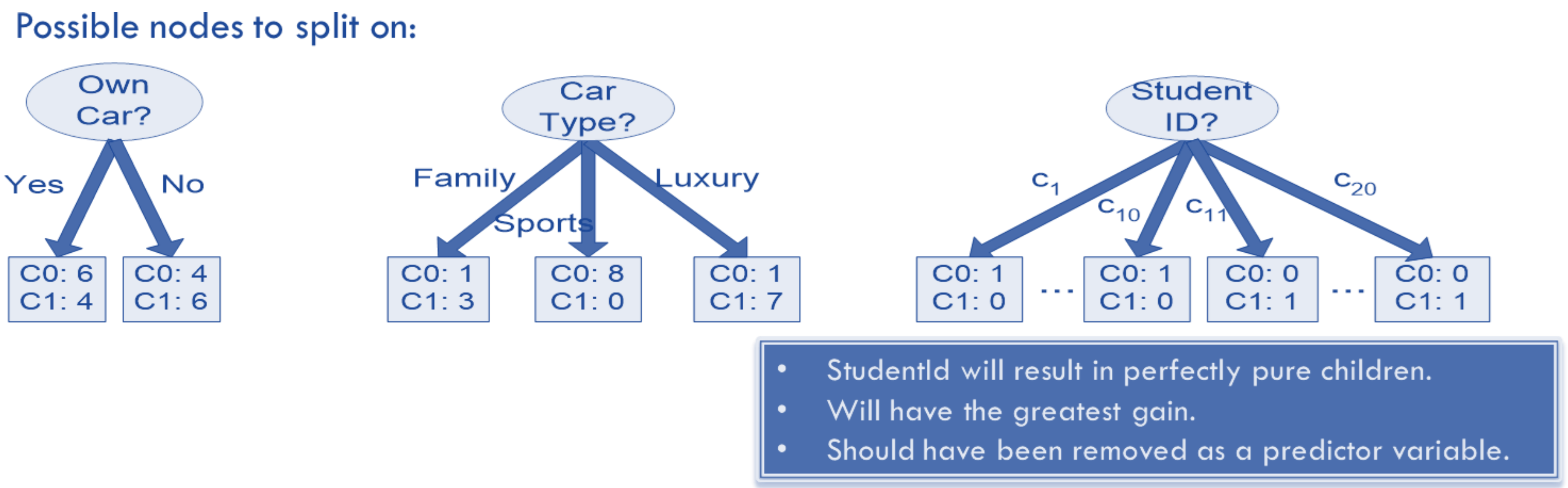
- Converting decision tree to rules:
  - Each branch shows a rule



R_1: If (Outlook=Sunny) $\wedge$ (Humidity=High) Then PlayTennis=No
R_2: If (Outlook=Sunny) $\wedge$ (Humidity=Normal) Then PlayTennis=Yes
R_3: If (Outlook=Overcast) Then PlayTennis=Yes
R_4: If (Outlook=Rain) $\wedge$ (Wind=Strong) Then PlayTennis=No
R_5: If (Outlook=Rain) $\wedge$ (Wind=Weak) Then PlayTennis=Yes

Decision tree

- 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)



Decision Tree

Gain ratio:

GainRatio(A) = Gain(A)/SplitInfo(A)

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

- Example:

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$$

gain_ratio(income) = 0.029/1.557 = 0.019

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Decision Tree

- C4.5 Algorithm:

  - is similar to ID3, but use gain ratio instead of information gain for selecting features
    - The attribute with the maximum gain ratio is selected as the splitting attribute

Decision Tree

- Gini index:
  - If a data set D contains examples from C classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^{C} p_j^2$$

    - where p_j is the relative frequency of class j in D

  - In Gini calculation we perform only binary split

  - If a data set D is split on attribute A into **two** subsets D_1 and D_2, the *gini* index $gini_A(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

Decision Tree

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity, $\Delta gini(A)$) is chosen to split the node

Decision Tree

- Example:

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- D has 9 tuples in buys_computer = "yes" and 5 in "no"

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

Decision Tree

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

$$gini_{income\in\{low,medium\}}(D) = \left(\frac{10}{14}\right)Gini(D_1) + \left(\frac{4}{14}\right)Gini(D_2)$$

$$= \frac{10}{14}\left(1-\left(\frac{7}{10}\right)^2-\left(\frac{3}{10}\right)^2\right)+\frac{4}{14}\left(1-\left(\frac{2}{4}\right)^2-\left(\frac{2}{4}\right)^2\right)$$
$$= 0.443$$
$$= Gini_{income\in\{high\}}(D).$$

- Gini$_{\{low,high\}}$ is 0.458; Gini$_{\{medium,high\}}$ is 0.450.
- {low,medium} (and {high}) has the lowest Gini index

Decision Tree

- CART Algorithm:

    - Use gini index instead of information gain or gain ratio for selecting features
        - The attribute with the minimum gini index is selected as the splitting attribute

Decision Tree

- Advantages of Decision tree:
  - Easy to understand, interpret, visualize
  - A decision tree does not require normalization or scaling of data (scale-invariant)
  - Missing values in the data also do not affect the process of building a decision tree to any considerable extent

Decision Tree

- Disadvantages of Decision tree:

  - They are unstable (a small change in the data can lead to a large change in the structure of the decision tree)
  - The space and time complexity of decision tree model is relatively high
  - Decision Tree is prone to overfit

Model Evaluation

- How to evaluate the performance of a model?

- What are the performance measure?

- Methods of performance estimation

  - Holdout
    - Keep part of the data set aside for testing purposes and use the rest to train the classifier (separating training and test dataset)

  - Cross validation
    - Partition data into k disjoint subsets
    - k-fold: train on k-1 partitions, test on the remaining one
    - Leave-one-out: k=n
    - Guarantees that each record is used the same number of times for training and testing

  - Bootstrap
    - Sampling with replacement

Model Evaluation

- Confusion matrix:
  - Given *m* classes, an entry, $CM_{i,j}$ in a **confusion matrix** indicates # of tuples in class *I* that were labeled by the classifier as class *j*
  - May have extra rows/columns to provide totals

Actual class\Predicted class	C_1	$\neg C_1$
C_1	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

Model Evaluation

- Example:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

Model evaluation

- Classifier Evaluation Metrics:

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P’	**N’**	**All**

  - Accuracy: **Accuracy** is the fraction of predictions our model got right

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

  - Error rate:

$$Error - rate = 1 - Accuracy$$

Model evaluation

- Problem with accuracy:

  - Consider a 2-class problem
    - Number of Class NO examples = 990
    - Number of Class YES examples = 10

  - If a model predicts everything to be class NO, accuracy is 990/1000 = 99 %
    - This is misleading because the model does not detect any class YES example
    - Detecting the rare class is usually more interesting (e.g., frauds, intrusions, defects, etc)

Model evaluation

Precision: Precision attempts to answer what proportion of positive identifications was actually correct?

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall: Recall (sensitivity) attempts to answer what proportion of actual positives was identified correctly?

$$\text{Recall} = \frac{TP}{TP + FN}$$

Model evaluation

- Specificity: Specificity is known as the True Negative Rate. It informs us about the proportion of actual negative cases that have gotten predicted as negative by our model.

$$Specificity = \frac{TN}{TN + FP}$$

Model evaluation

- F1-Score: The F1 score is the harmonic mean of precision and recall, taking both metrics into account in the following equation:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Model evaluation

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	10	0
	Class=No	10	980

$$\text{Precision (p)} = \frac{10}{10+10} = 0.5$$

$$\text{Recall (r)} = \frac{10}{10+0} = 1$$

$$\text{F−measure (F)} = \frac{2 * 1 * 0.5}{1+0.5} = 0.66$$

$$\text{Accuracy} = \frac{990}{1000} = 0.99$$

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	1	9
	Class=No	0	990

$$\text{Precision (p)} = \frac{1}{1+0} = 1$$

$$\text{Recall (r)} = \frac{1}{1+9} = 0.1$$

$$\text{F−measure (F)} = \frac{2 * 0.1 * 1}{1+0.1} = 0.18$$

$$\text{Accuracy} = \frac{991}{1000} = 0.991$$