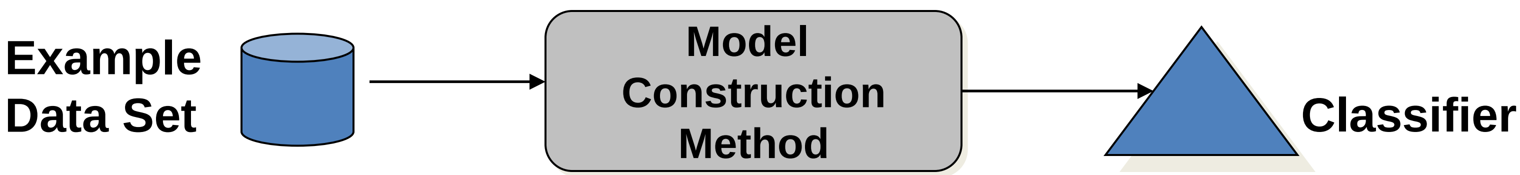
Classification

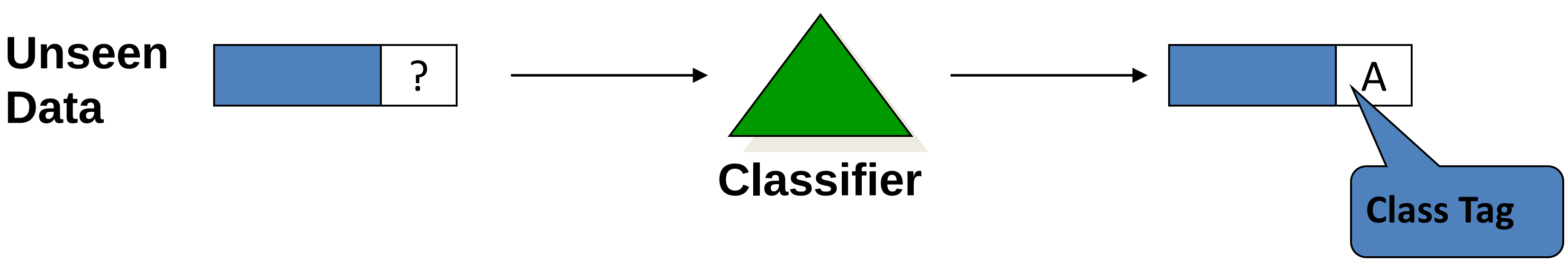
Classification Models

- Assign labels to objects
- Two-Stage Process

  - Given a data set of **labeled** examples, use a classification method to train a classification model, known as **classifier**



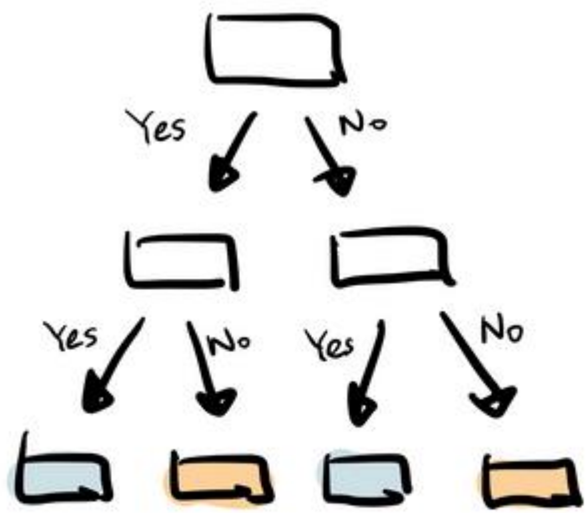
  - Given a trained classifier, classify a data record with unknown class to one of the pre-defined classes

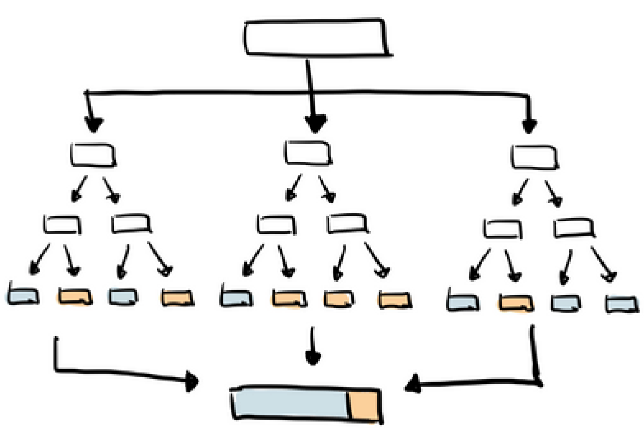


Classification Examples

- **Spam Email Filter (binary classifier)**: classify emails labeled as spam or not spam by learning patterns in the content, sender information, and other features
- **Sentiment Analysis (multi-class classifier)**: classify media posts or product reviews as positive, negative, or neutral sentiments expressed by the author
- **Medical Diagnosis (binary)**: a model trained on patient symptoms and medical history can classify whether a patient is likely to have a certain disease
- **Credit Risk Assessment (multi-class)**: classify loan applicants as low, medium, or high risk based on factors such as credit score, income, and debt-to-income ratio
- **Image Recognition (multi-class)**: a model can classify images of animals into different categories such as cats, dogs, or birds

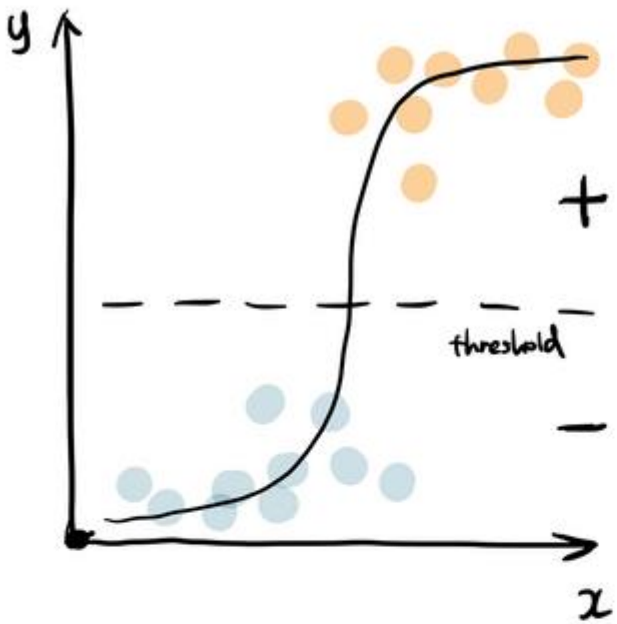
Decision Trees & Random Forest





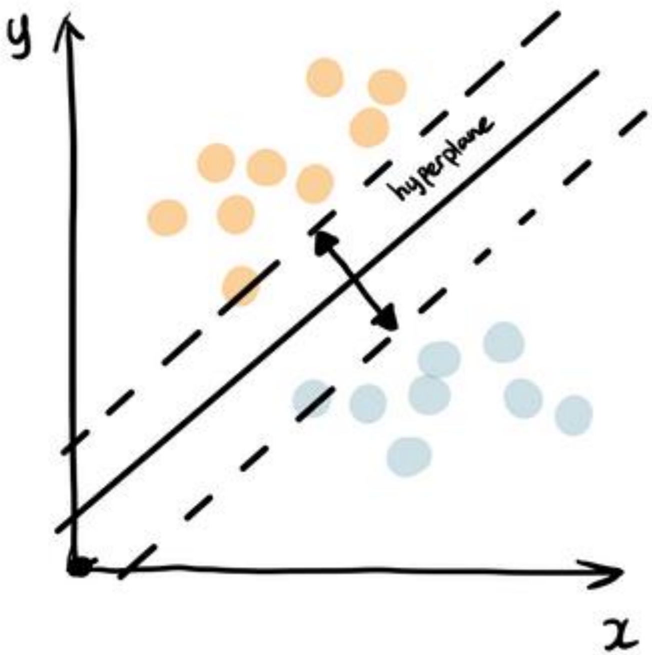
- **Decision Trees**:
  - A tree-like model where each internal node represents a "test" on an attribute
  - It splits the data into different branches based on the attribute values
  - Decision trees are interpretable and can handle both numerical and categorical data
- **Random Forest**:
  - A collection of decision trees where each tree is built using a random subset of features and a random subset of the training data
  - It reduces overfitting and improves generalization compared to individual decision trees
  - Random forests are robust and perform well on a variety of datasets

Logistic Regression



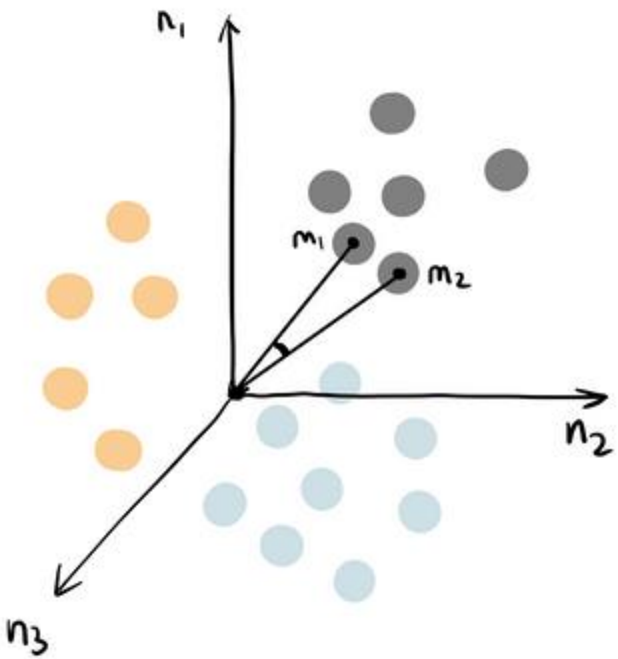
- A linear model used for **binary** classification problems
- It models the probability that a given input belongs to a certain class using the logistic function
- It's simple, interpretable, and efficient for linearly separable data

Support Vector Machine (SVM)



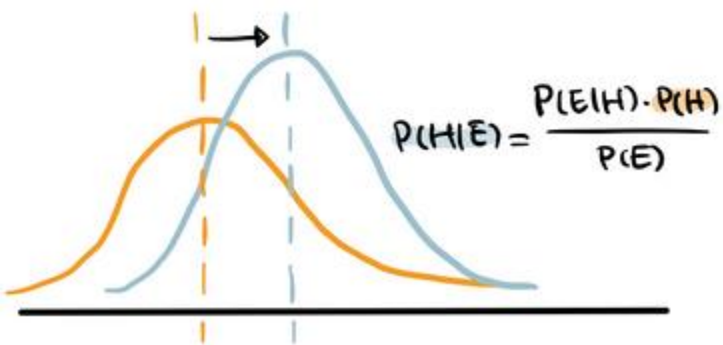
- A model that finds the optimal hyperplane separating different classes in the feature space
  - Classify the data based on the position in relation to the hyperplane between positive class and negative class
- SVM aims to maximize the margin between classes, thus enhancing generalization
- It can handle both linear and non-linear classification tasks using different kernel functions

K-Nearest Neighbors (KNN)



- Each data point is represented in a n dimensional space, which is defined by n features
  - And it calculates the distance between one point to another, then assign the label of unobserved data based on the labels of nearest observed data points
  - The classification of a data point is determined by the majority class among its k nearest neighbors in the feature space
- KNN is simple to understand and implement, especially for small datasets
- It does not learn explicit models and can be sensitive to the choice of k

Naive Bayes



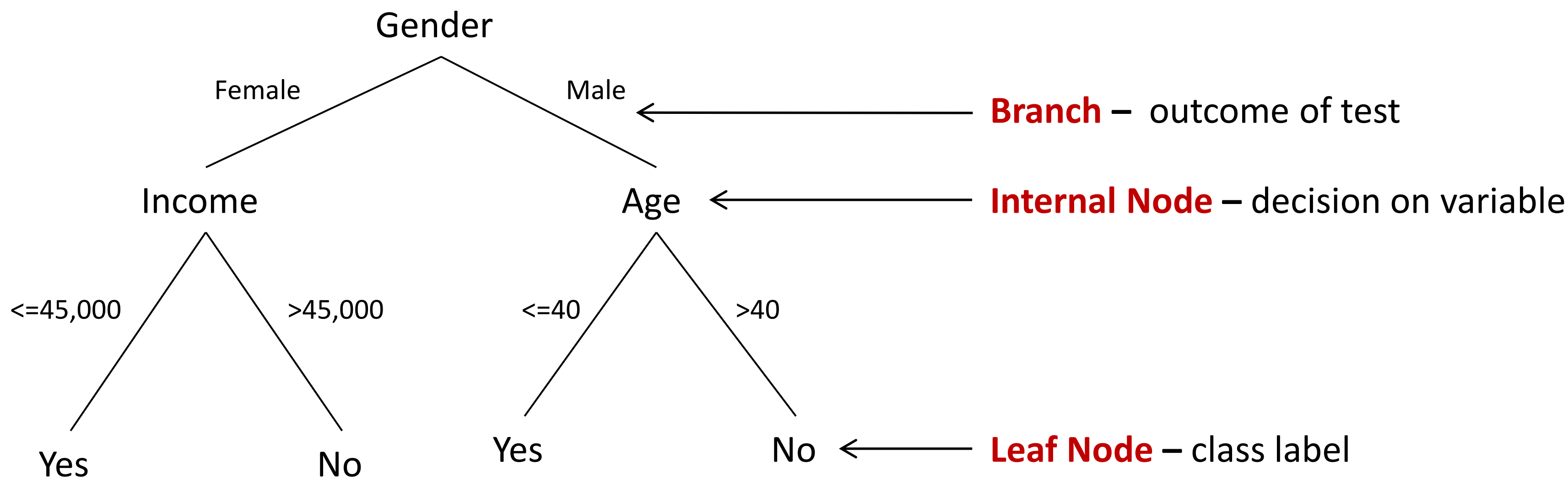
- A probabilistic classifier based on Bayes' theorem with an assumption of independence between features
- It calculates the probability of each class given a set of features and selects the class with the highest probability
- Naive Bayes is efficient, especially for text classification and other high-dimensional datasets

Influential Factors for a Good Model

- Accuracy
  - Estimated accuracy during development stage vs. actual accuracy during practical use
- Performance
  - Time taken for model construction (training time)
  - Time taken for the model to infer
- Interpretability
  - Ease of interpreting decisions by the model
  - Understanding and insight provided by the model
- Robustness:
  - Handling noise and missing values
- Scalability:
  - Ability to handle large datasets
- Other measures, e.g., decision tree size or compactness of rules

Decision Tree

Example of Visual Structure



Decision Tree - Use Cases

- When a series of questions (yes/no) are answered to arrive at a classification decision
  - Ex.:
    - Checklist of symptoms during a doctor’s evaluation of a patient

- When “if-then” conditions are preferred to mathematical models
  - Ex.:
    - Financial decisions such as loan approval or fraud detection

Example of a Decision Tree

Training Data

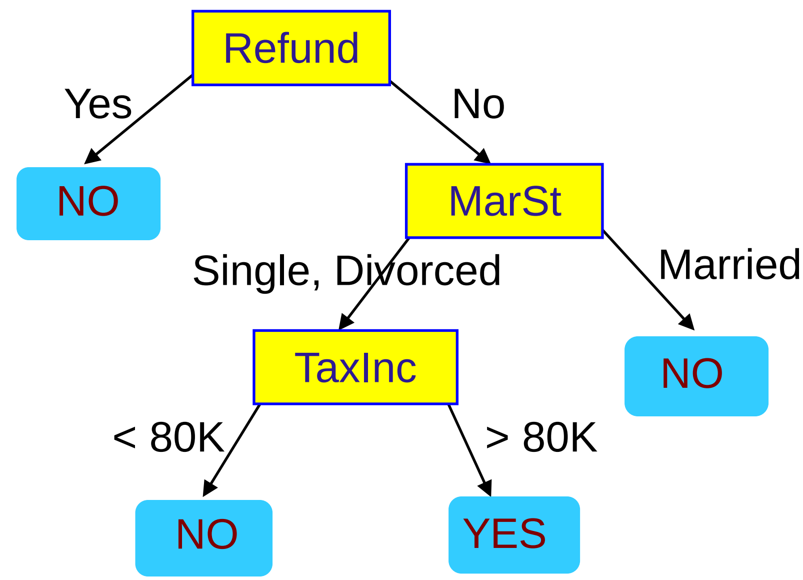
categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Descriptive Attributes

Class attribute

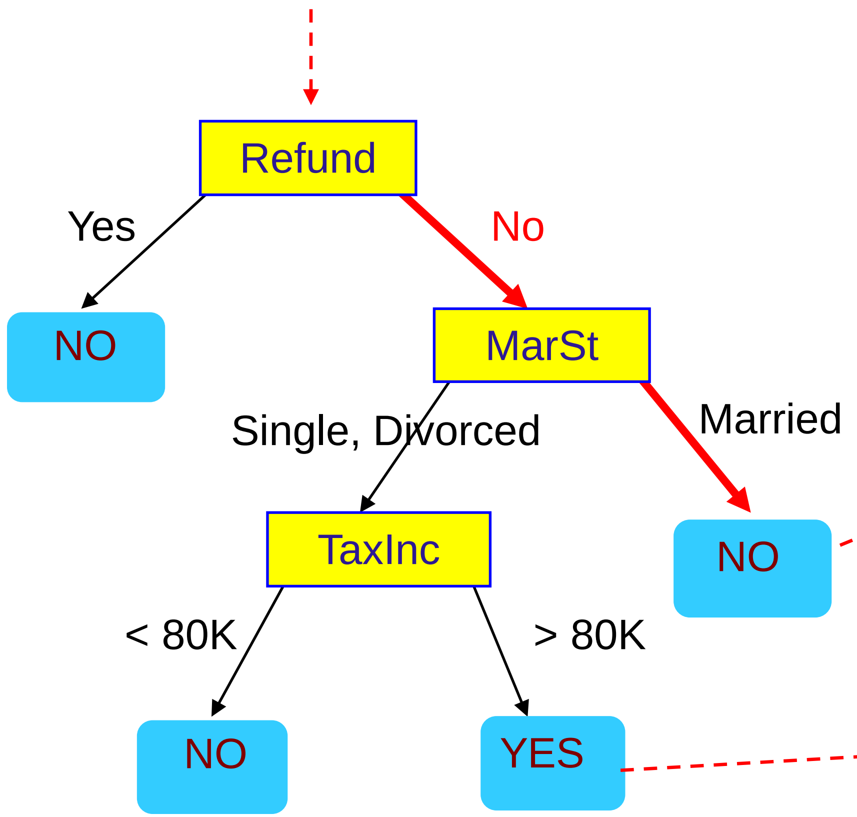
Model: Decision Tree



Internal Nodes: Splitting Attributes
Links: Attribute values
Leaf Nodes: Class labels

Apply Model to Test Data

Start from the root of tree.



Test Data

Assign Cheat to “No”

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

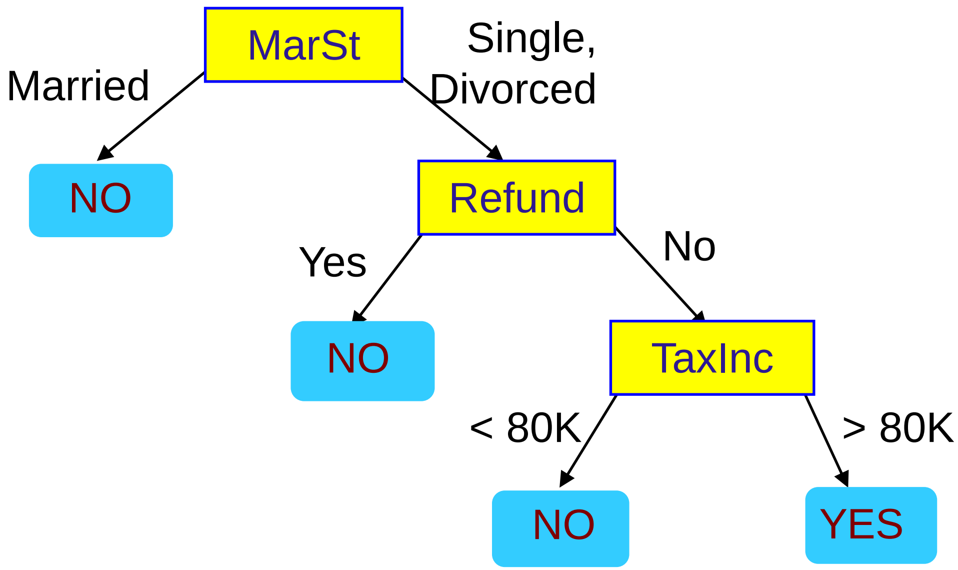
Assign Cheat to “Yes”

Refund	Marital Status	Taxable Income	Cheat
No	Single	90K	?

Another Example of a Decision Tree

categorical categorical continuous class

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**



There could be more than one tree that fits the same data!

Decision Trees Induction

- Input variables can be continuous or discrete
- Output:
  - A tree that describes the decision flow.
  - Leaf nodes return return class labels and, in some implementations, they also return the probability scores.
  - Trees can be converted to a set of "decision rules“
    - "IF Refund=Yes AND Marital_Status=No THEN Cheat=No with 75% probability“

Decision Tree Induction

- Algorithms:
  - CART
  - ID3, C4.5, C5
  - CHAID
  - C-SEP
  - G-statistics
  - SLIQ
  - SPRINT
  - …others

Decision Tree Induction

- **Principle of Tree Construction**

1. If all examples are of the same class, create a leaf node labelled by the class

2. If examples in the training set are of different classes, determine which attribute should be *selected* as the root of the current tree

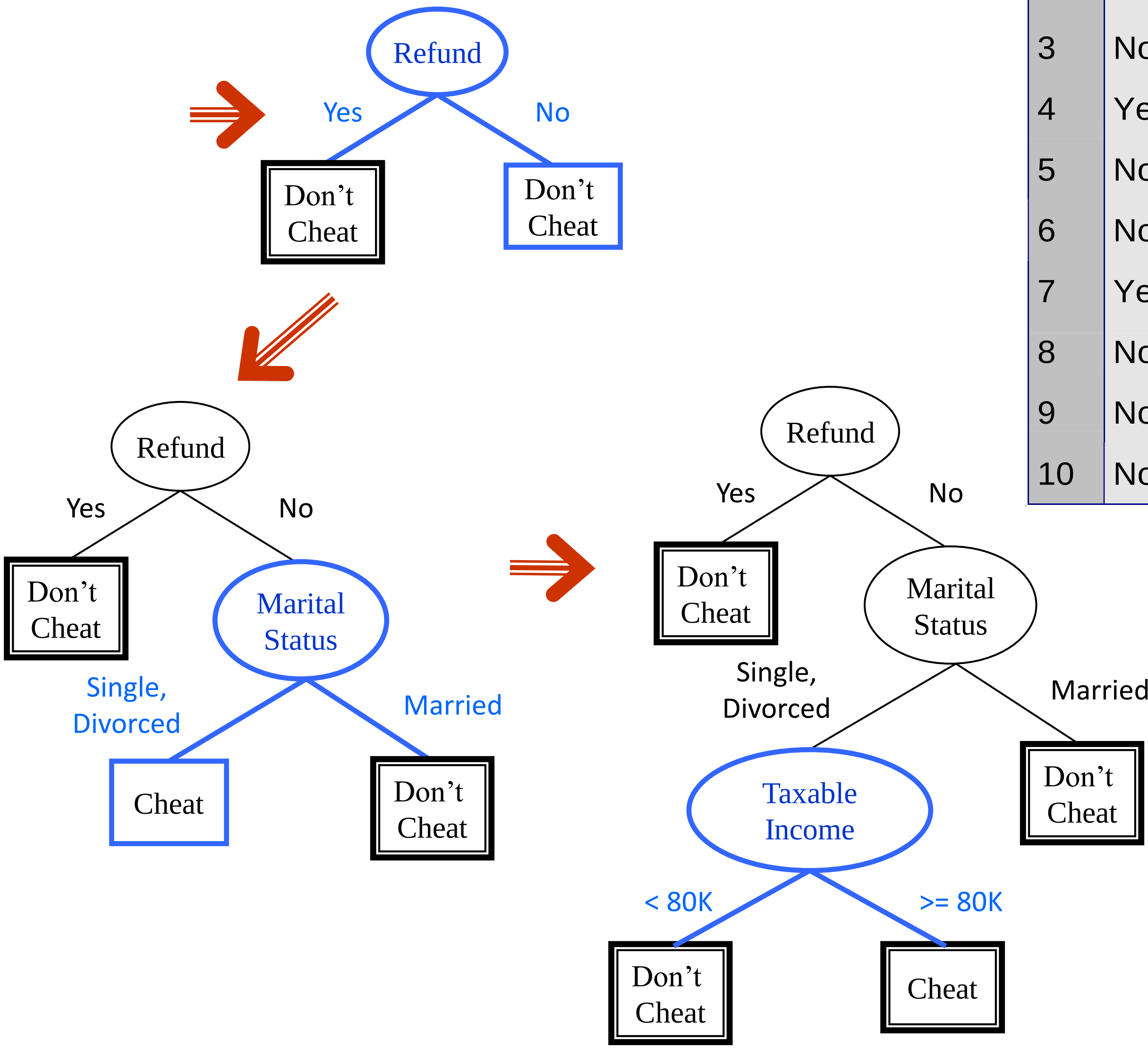
3. Partition the input examples into subsets according to the values of the selected root attribute

4. Construct a decision tree recursively for each subset

5. Connect the roots for the subtrees to the root of the whole tree via labelled links

Example

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



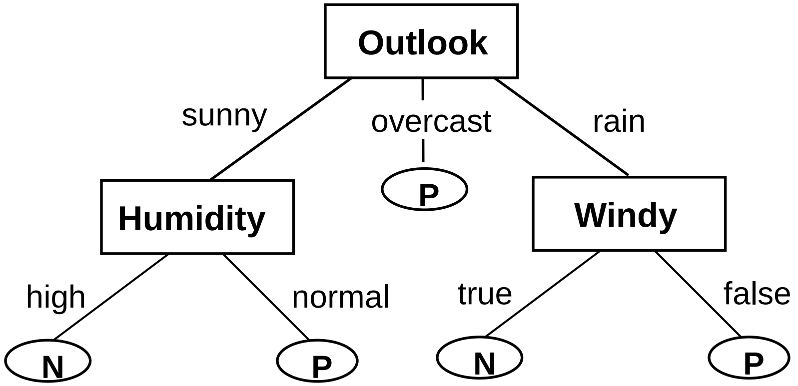
Example

Input Training Examples

Outlook	Temperature	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

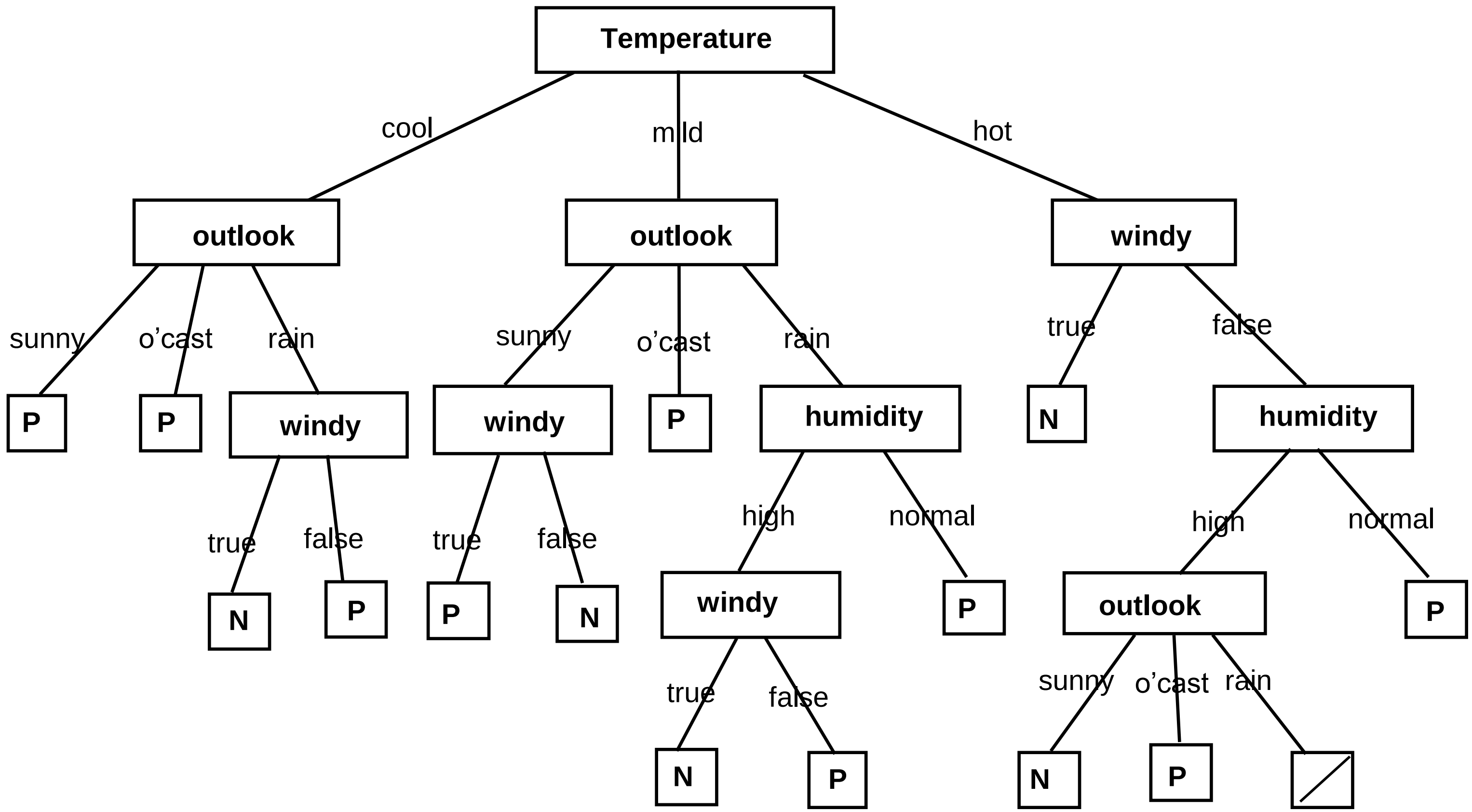
Descriptive Attributes

Class attribute



Decision Tree Induction

- What could random selection of attributes produce?



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

Tree Induction

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

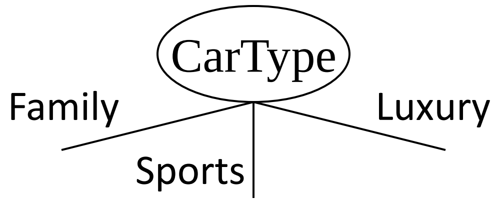
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to Specify Test Condition?

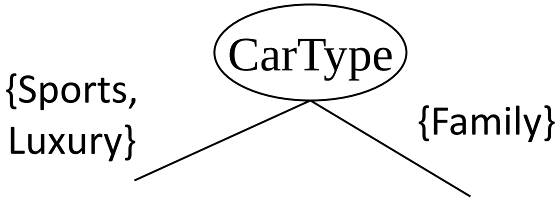
- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

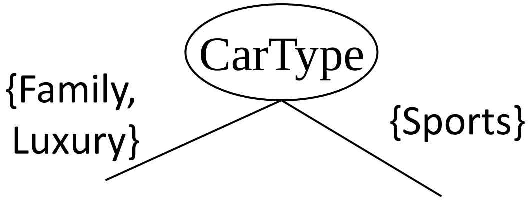
- Multi-way split: Use as many partitions as distinct values.



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

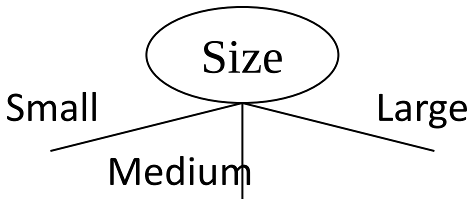


OR

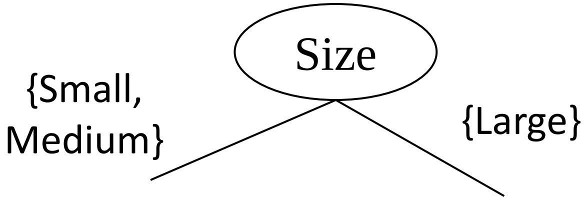


Splitting Based on Ordinal Attributes

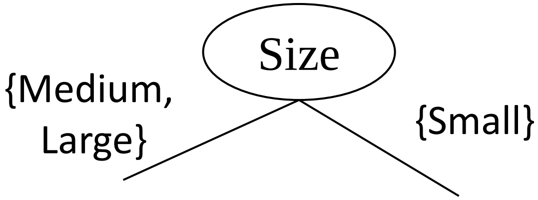
- Multi-way split: Use as many partitions as distinct values.



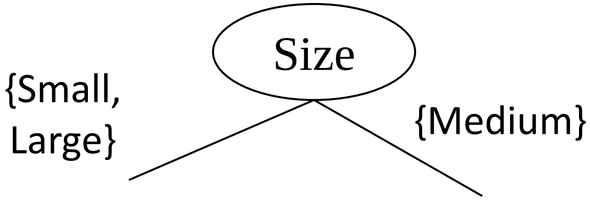
- Binary split: Divides values into two subsets. Need to find optimal partitioning.



OR



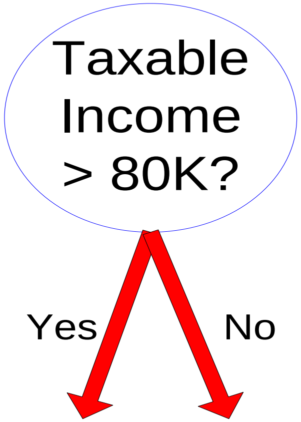
- What about this split?
  – No! the grouping should not violate the order property of the attribute values.



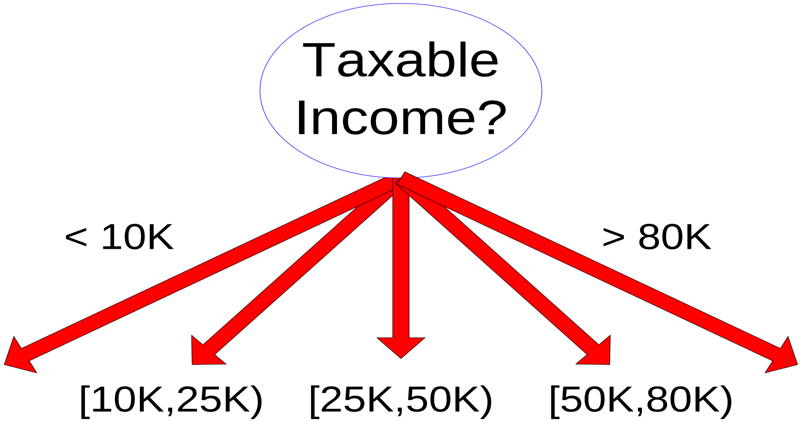
Splitting Based on Continuous Attributes

- Different ways of handling
  - Let attribute *A* be a continuous-valued attribute
  - Discretization to form an ordinal categorical attribute
    - ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - 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}
  - Binary split: (A < v) or ($A \geq$ v), OR A in a range
    - Consider all possible splits and finds the best cut
  - Multi-way split: A in one of the ranges
- Can be compute intensive

Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

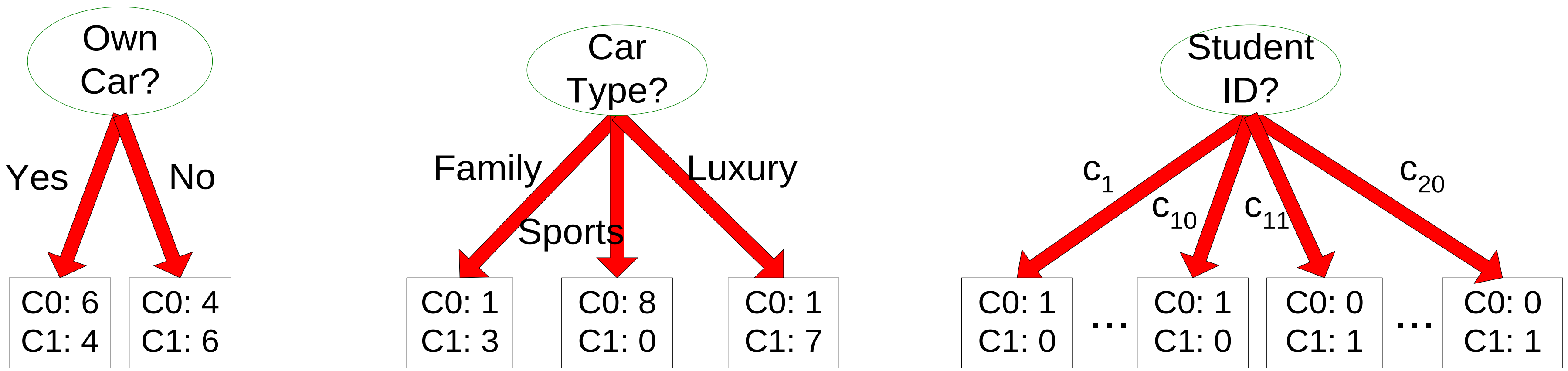
Tree Induction

- 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

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



Which test condition is the best?

How to determine the Best Split

- Greedy approach:
  – 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

Measures of Node Impurity

- Misclassification Error
- Gini Index (CART, SLIQ, SPRINT)
- Information Gain (ID3), Information Gain Ratio (C4.5 and C5)
- Chi-square Statistic (CHAID)

Splitting Criteria based on Classification Error

- Classification error at a node *t* :

$$\boxed{Error(t) = 1 - \max_{i} P(i \mid t)}$$

- Measures misclassification error made by a node.
  - Maximum ($1 - 1/n_c$) when records are equally distributed among all classes
  - Minimum (0.0) when all records belong to one class

Examples for Computing Error

$$Error(t) = 1 - \max_i P(i \mid t)$$

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

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

Error = 1 – max (0, 1) = 1 – 1 = 0

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

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

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

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

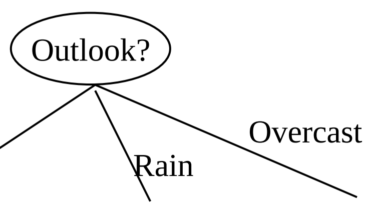
P(C1) = 2/6   P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Outlook	Temperature	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

	Parent
P	9
N	5
Error= 5/14	

Outlook	Temperature	Humidity	Windy	Class
sunny	Hot	high	FALSE	N
sunny	Hot	high	TRUE	N
Sunny	Mild	high	FALSE	N
Sunny	Cool	normal	FALSE	P
Sunny	Mild	normal	TRUE	P



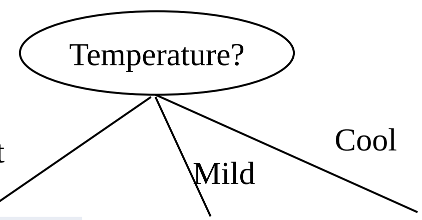
P	2
N	3
Error= 2/5	

P	3
N	2
Error = 2/5	

P	4
N	0
Error = 0	

Weighted Average Error for the Outlook Split

$$Error(Outlook) = \frac{5}{14} \times \frac{2}{5} + \frac{5}{14} \times \frac{2}{5} + \frac{4}{14} \times 0 = \frac{2}{14} + \frac{2}{14} = \frac{4}{14}$$



P	2
N	2
Error= 2/4	

P	4
N	2
Error = 2/6	

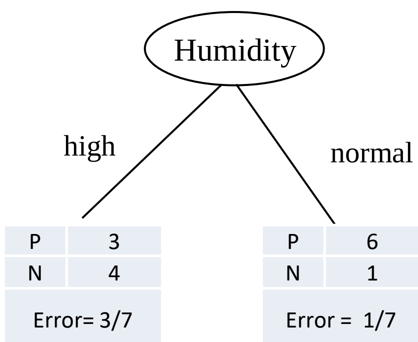
P	3
N	1
Error = 1/4	

Weighted Average Error for the Temperature Split

$$Error(Temperature) = \frac{4}{14} \times \frac{2}{4} + \frac{6}{14} \times \frac{2}{6} + \frac{4}{14} \times \frac{1}{4} = \frac{2}{14} + \frac{2}{14} + \frac{1}{14} = \frac{5}{14}$$

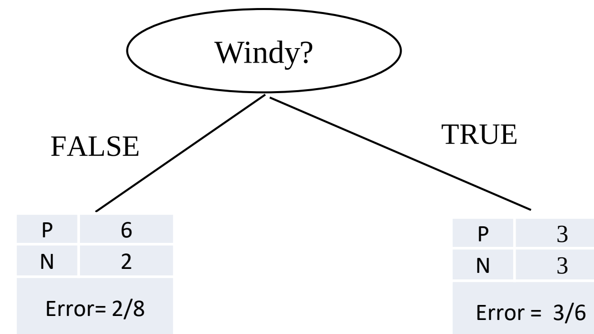
Outlook	Temperature	Humidity	Windy	Class
sunny	Hot	high	FALSE	N
sunny	Hot	High	TRUE	N
Overcast	Hot	high	FALSE	P
Rain	Mild	high	FALSE	P
Rain	Cool	Normal	FALSE	P
Rain	Cool	Normal	TRUE	N
Overcast	Cool	Normal	TRUE	P
Sunny	Mild	High	FALSE	N
Sunny	Cool	normal	FALSE	P
Rain	Mild	Normal	FALSE	P
Sunny	Mild	Normal	TRUE	P
Overcast	Mild	High	TRUE	P
Overcast	Hot	Normal	FALSE	P
Rain	mild	high	TRUE	N

	Parent
P	9
N	5
Error= 5/14	



Weighted Average Error for the Humidity Split

$$Error(Outlook) = \frac{7}{14} \times \frac{3}{7} + \frac{7}{14} \times \frac{1}{7} = \frac{3}{14} + \frac{1}{14} = \frac{4}{14}$$

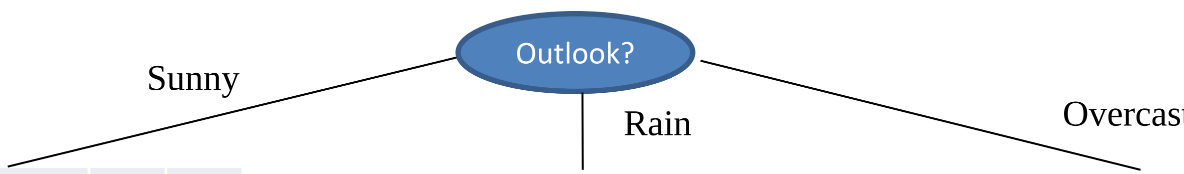


Weighted Average Error for the Windy Split

$$Error(Temperature) = \frac{8}{14} \times \frac{2}{8} + \frac{6}{14} \times \frac{3}{6} = \frac{2}{14} + \frac{3}{14} = \frac{5}{14}$$

The best two splits are Outlook and Humidity
We can use Outlook or Humidity since both have the same error.

Outlook	Temperature	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	Temperature	Humidity	Windy	Class
sunny	Hot	high	FALSE	N
sunny	Hot	high	TRUE	N
Sunny	Mild	high	FALSE	N
Sunny	Cool	normal	FALSE	P
Sunny	Mild	normal	TRUE	P

Outlook	Temperature	Humidity	Windy	Class
Rain	Mild	high	FALSE	P
Rain	Cool	normal	FALSE	P
Rain	Cool	normal	TRUE	N
Rain	Mild	normal	FALSE	P
Rain	mild	high	TRUE	N

Outlook	Temperature	Humidity	Windy	Class
Overcast	Hot	high	FALSE	P
Overcast	Cool	normal	TRUE	P
Overcast	Mild	high	TRUE	P
Overcast	Hot	normal	FALSE	P

Next: Split each smaller data

Decision Tree Induction Algorithms

- **Overview**
  - CART Algorithm
    - Produce binary decision tree
    - Use Gini Index of Impurity as attribute selection measure
  - ID3 Family: C4.5 and C5 (See5)
    - Similar to ID3
    - Uses Information Gain Ratio
  - CHAID Algorithm
    - Use Chi-square test (χ^2) as attribute selection measure
  - C-SEP: performs better than info. gain and gini index in certain cases
  - G-statistics: has a close approximation to χ2 distribution
  - ... others
- The algorithms mainly differ in adopted attribute selection measures
- Studies show that there are only marginal differences among the attribute selection measures w.r.t. model accuracy

Tree Construction Algorithm

```
Algorithm TreeConstruct (C: Training Set) : Decision Tree
begin
    Tree := ∅;
    if C is not empty then
        if all examples in C are of one class then
            Tree := a leaf node labelled by the class tag
        else begin
            select attribute T according to an attribute selection measure as the root;
            partition C into C1, C2, ..., Cw by values of T;
            for each Ci (1≤ i ≤ w) do
                ti ← TreeConstruct (Ci);
            Tree := a tree T as root and t1, t2, ..., tw as subtrees
            label the links from T to the subtrees with values of T
        end;
    return(Tree);
end;
```

Attribute Selection Measures

Measure of Impurity: Gini (CHAD Algorithm)

- Gini Index for a given node *t* :

$$GINI(t) = 1 - \sum_{j} [p(j \mid t)]^2$$

(NOTE: $p(j \mid t)$ is the relative frequency of class *j* at node *t*).

- – Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying most impure set
- – Minimum (0.0) when all records belong to one class, implying least impure set

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Examples for computing Gini

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

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

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

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = 0

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

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

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

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

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

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

Splitting Based on Gini

- When a node p is split into k partitions (children), the quality of split is computed as,

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

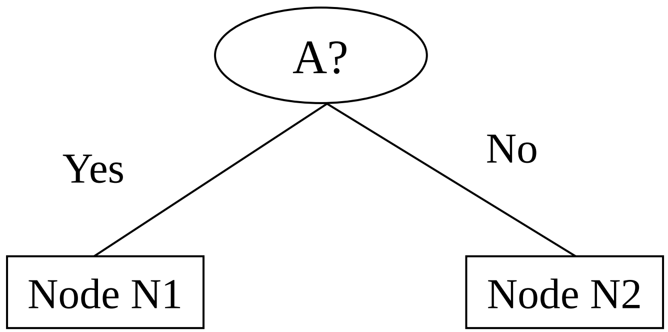
where, n_i = number of records at child i,

n = number of records at node p.

Binary Attributes: Computing Gini Index

- Splits into two partitions
- Effect of weighing partitions:
  - Larger and purer partitions are sought for.

	Parent
C1	**6**
C2	**6**
Gini = 0.500	



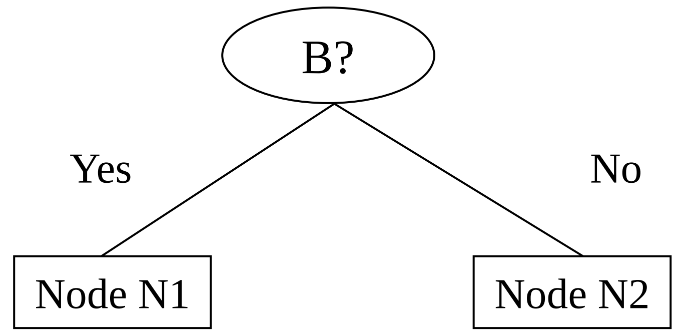
	N1
C1	6
C2	1

	N2
C1	0
C2	5

Gini(N1) = $1 - (6/7)^2 - (1/7)^2 = 0.245$

Gini(N2) = $1 - (0/5)^2 - (5/5)^2 = 0$

Gini(Children) = $7/12 * 0.245 + 5/12 * 0 = 0.1429$



	N1
C1	4
C2	2

	N2
C1	2
C2	4

Gini(N1) = $1 - (4/6)^2 - (2/6)^2 = 0.2222$

Gini(N2) = $1 - (2/6)^2 - (4/6)^2 = 0.2222$

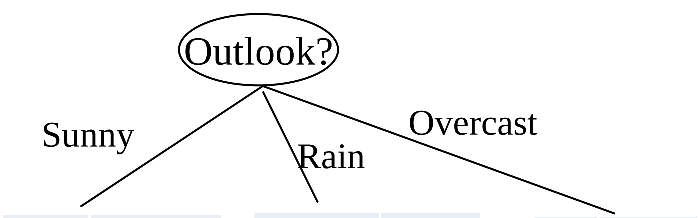
Gini(Children) = $6/12 * 0.2222 + 6/12 * 0.2222 = 0.2222$

Choose split A; it has less Gini Index than split B

$$GINI(t) = 1 - \sum_j [p(j|t)]^2$$

Outlook	Temperature	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

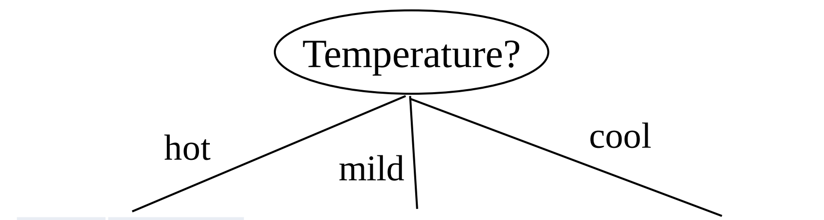
	Parent
P	9
N	5
Gini Index = 1 –(25/196 + 81/196) = 90/196 = 0.4592	



P	2
N	3
Gini Index = 1-(9/25+4/25) =1-(13/25) = 12/25	

P	3
N	2
Gini Index = 1-(13/25) = 12/25	

P	4
N	0
Gini Index = 0	



P	2
N	2
Gini Index = 1-(8/16) = 8/16	

P	4
N	2
Gini Index = 1-(20/36) = 16/36	

P	3
N	1
Gini Index = 1-(10/16) = 6/16	

Weighted Average Gini Index for the Outlook Split

$$Gini(Outlook) = \frac{5}{14} \times \frac{12}{25} + \frac{5}{14} \times \frac{12}{25} + \frac{4}{14} \times 0 = \frac{6}{35} + \frac{6}{35} = \frac{12}{35} = 0.3429$$

Weighted Average Gini Index for the Temperature Split

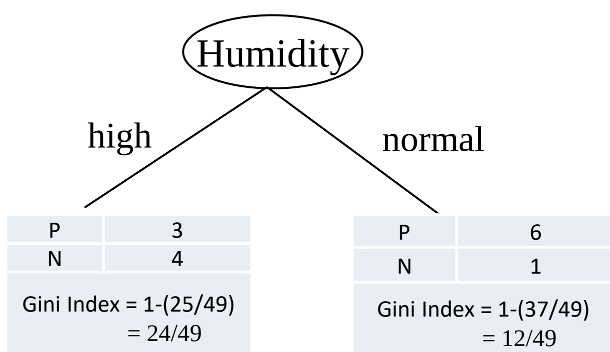
$$Gini(Temperature) = \frac{4}{14} \times \frac{8}{16} + \frac{6}{14} \times \frac{16}{36} + \frac{4}{14} \times \frac{6}{16}$$

$$Gini(Temperature) = \frac{2}{14} + \frac{4}{21} + \frac{3}{28} = \frac{4}{21} + \frac{7}{28} = 0.4405$$

$$GINI(t) = 1 - \sum_{j} [p(j \mid t)]^2$$

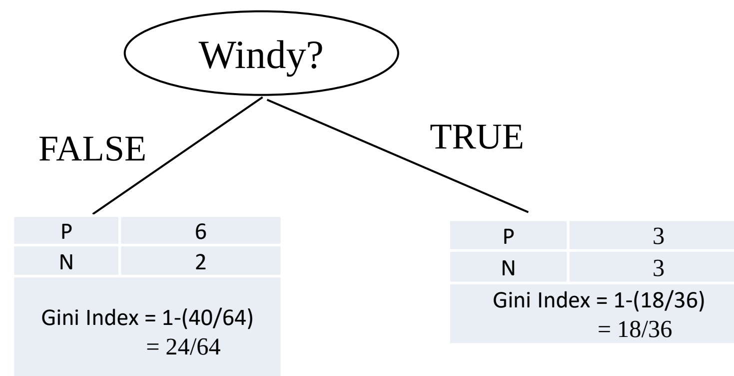
Outlook	Temperature	Humidity	Windy	Class
sunny	Hot	high	FALSE	N
sunny	Hot	high	TRUE	N
Overcast	Hot	high	FALSE	P
Rain	Mild	high	FALSE	P
Rain	Cool	normal	FALSE	P
Rain	Cool	normal	TRUE	N
Overcast	Cool	normal	TRUE	P
Sunny	Mild	high	FALSE	N
Sunny	Cool	normal	FALSE	P
Rain	Mild	normal	FALSE	P
Sunny	Mild	normal	TRUE	P
Overcast	Mild	high	TRUE	P
Overcast	Hot	normal	FALSE	P
Rain	mild	high	TRUE	N

	Parent
P	9
N	5
Gini Index = 1 –(25/196 + 81/196) = 90/196 = 0.4592	



Weighted Average Gini Index for the Humidity Split

$$Error(Outlook) = \frac{7}{14} \times \frac{24}{49} + \frac{7}{14} \times \frac{12}{49} = \frac{12}{49} + \frac{6}{49} = \frac{18}{49} = 0.3673$$

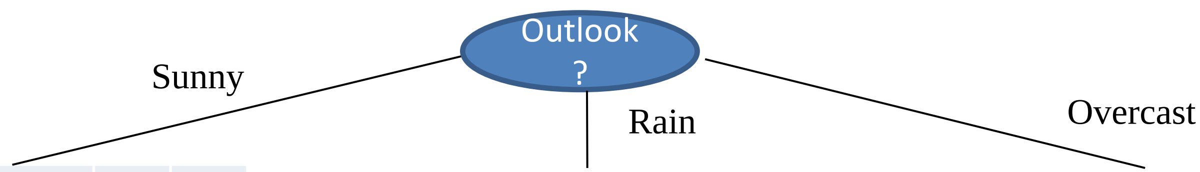


Weighted Average Gini Index for the Windy Split

$$Error(Temperature) = \frac{8}{14} \times \frac{24}{64} + \frac{6}{14} \times \frac{18}{36} = \frac{3}{14} + \frac{3}{14} = \frac{6}{14} = 0.4286$$

The best split is Outlook (0.3429)

Outlook	Temperature	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	Temperature	Humidity	Windy	Class
sunny	Hot	high	FALSE	N
sunny	Hot	high	TRUE	N
Sunny	Mild	high	FALSE	N
Sunny	Cool	normal	FALSE	P
Sunny	Mild	normal	TRUE	P

Outlook	Temperature	Humidity	Windy	Class
Rain	Mild	high	FALSE	P
Rain	Cool	normal	FALSE	P
Rain	Cool	normal	TRUE	N
Rain	Mild	normal	FALSE	P
Rain	mild	high	TRUE	N

Outlook	Temperature	Humidity	Windy	Class
Overcast	Hot	high	FALSE	P
Overcast	Cool	normal	TRUE	P
Overcast	Mild	high	TRUE	P
Overcast	Hot	normal	FALSE	P

Next: Split each smaller data

Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

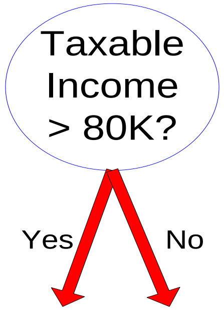
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A < v and A ≥ v
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Taxable Income									
Sorted Values	60	70	75	85	90	95	100	120	125	220

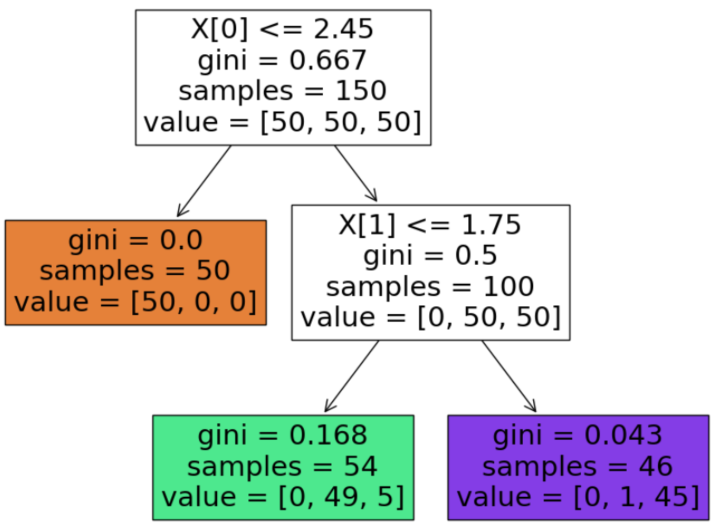
Split Positions	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

Decision Tree for the Iris dataset: using two attributes

In [15]:
```
iris = load_iris(as_frame=True)
columns_to_use = ["petal length (cm)", "petal width (cm)"]
X_iris = iris.data[columns_to_use].values
y_iris = iris.target
tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf = tree_clf.fit(X_iris, y_iris)
```

In [18]:
```
plt.figure(figsize=(10,8))
plot_tree(tree_clf, filled=True)
plt.title("Decision tree trained on two attributes")
plt.show()
```

Decision tree trained on two attributes



In [19]:
```
tree_clf.predict([[3, 2.5]])
```
Out[19]:
```
array([2])
```

In [16]:
```
tree_clf.predict([[5, 1.5]])
```
Out[16]:
```
array([1])
```

In [17]:
```
tree_clf.predict([[.5, 1.5]])
```
Out[17]:
```
array([0])
```

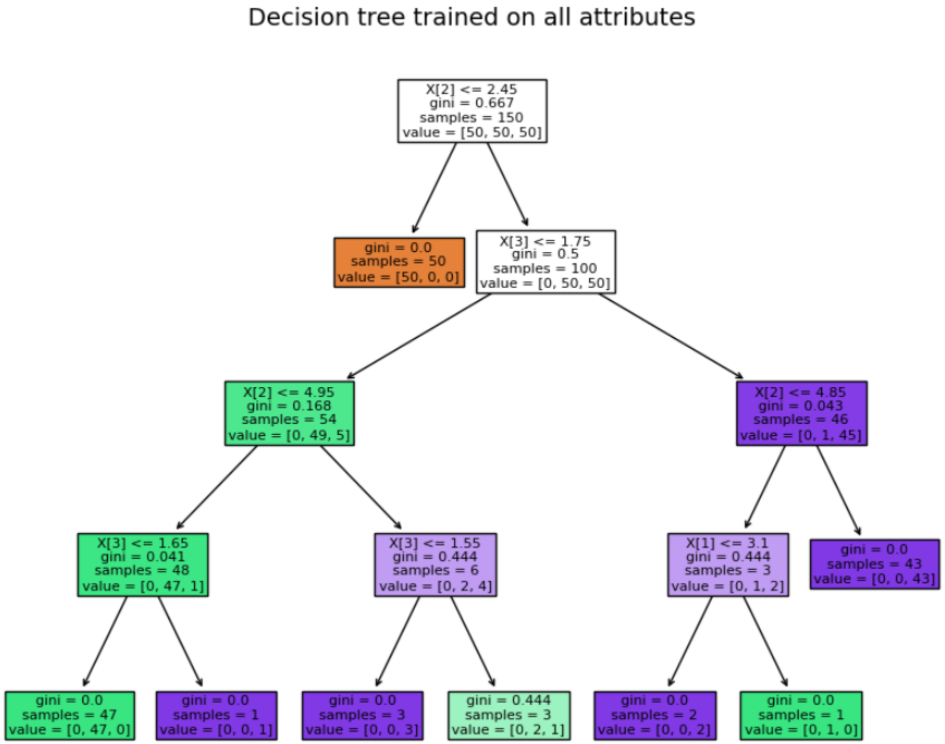
In [20]:
```
r = export_text(tree_clf, feature_names=columns_to_use)
print(r)
```

```
|--- petal length (cm) <= 2.45
|   |--- class: 0
|--- petal length (cm) >  2.45
|   |--- petal width (cm) <= 1.75
|   |   |--- class: 1
|   |--- petal width (cm) >  1.75
|   |   |--- class: 2
```

Decision Tree for the Iris dataset: using all the four attributes

```python
X_iris = iris.data.values
y_iris = iris.target
tree_clf = DecisionTreeClassifier(max_depth=4, random_state=42)
tree_clf = tree_clf.fit(X_iris, y_iris)
```

```python
tree_clf.fit(X_iris, y_iris)
plt.figure(figsize=(10,8))
plot_tree(tree_clf, filled=True)
plt.title("Decision tree trained on all attributes")
plt.show()
```



In [31]:
```python
print(tree_clf.predict([[.5, 1.5,5,7.5]]))
print(tree_clf.predict([[.5, 1.5,2.5,1.2]]))
print(tree_clf.predict([[5, 1.5,2,3]]))
```

In [31]:
```python
print(tree_clf.predict([[.5, 1.5,5,7.5]]))
print(tree_clf.predict([[.5, 1.5,2.5,1.2]]))
print(tree_clf.predict([[5, 1.5,2,3]]))
```

```
[2]
[1]
[0]
```

Attribute Selection Measures

- **Information Gain (ID3 Algorithm)**
  - An information system:
    - A set of n possible events $E_1, E_2, \ldots, E_n$;
    - Each event may occur with a probability $p(E_k)$ $(1 \leq k \leq n)$
    - $p(E_1) + p(E_2) + \ldots + p(E_n) = 1$.
  - Given a training set, each attribute can be considered as such an information system (e.g. Outlook).
  - Classification involves two information systems:
    - **Attribute A (e.g. Outlook)**: events: sunny, overcast and rain;

    event probabilities: p(sunny) = 5/14, p(overcast) = 4/14, p(rain) = 5/14

    - **Class attribute**: events: P and N;

    event probabilities: p(P) = 9/14, p(N) = 5/14

Attribute Selection Measures

- **Information Gain**
  - Self-information of event E: the amount of information being conveyed when E occurs, defined as:

$$\boxed{I(E) = \log\frac{1}{p(E)} = -\log p\,(E)}$$

If someone says “I have something” This does not provide any information. I(E) = 0
If we know that someone has a coin, and they say “we have a coin” again, no information
“I was born in a day” No information
“I was born in January” Now this provided us with some information. I(E) = log(1/(1/12)), 1 out of twelve months
“I have a number of two digits, and it is 15 .. I(E) = log (1/(1/100))

Attribute Selection Measures

- **Entropy**

– The average of the self-information of all events in an information system S, known as *entropy*, is defined as: The expected amount of information (entropy)

For any random variable S that follows a probability distribution P

$$H(S) = \sum_{k=1}^{N} p(E_k) \cdot I(E_k) = -\sum_{k=1}^{N} p(E_k) \cdot \log p\,(E_k)$$

H(S) represents degree of uncertainty. When one event always occurs, i.e. the least uncertain, H(S) = 0. When all events have equal chance to occur, i.e. the most uncertain, H(S) = log N.

C1	0
C2	16
Enropy H(S)=0.000	

C1	4
C2	12
Enropy H(S) = 0.811	

C1	6
C2	10
Enropy H(S) = 0.9644	

C1	8
C2	8
Enropy H(S) = 1	

- Entropy will be low if a message is highly predictable.
- On the other hand, if the message is highly unpredictable, then the entropy will be high.

Attribute Selection Measures

- **Information Gain (cont'd)**
  - Conditional self-information of event E of system S_1 given that event F of system S_2 has occurred is defined as:

$$I(E\,|\,F) = \log\frac{1}{p(E\,|\,F)} = -\log\, p(E\,|\,F) = -\log\frac{p(E\cap F)}{p(F)}$$

  - The *expected information* of system S_1 in the presence of system S_2, is defined as the average of conditional self-information of all events in S_1:

$$H(S_1\,|\,S_2) = \sum_{i=1}^{n}\sum_{j=1}^{m} p(E_i \wedge F_j)\cdot I(E_i\,|\,F_j) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(E_i \wedge F_j)\cdot \log\frac{p(E_i \wedge F_j)}{p(F_j)}$$

Attribute Selection Measures

- **Information Gain (cont'd)**
  - The information gain, also known as mutual information between S_1 and S_2, is defined as

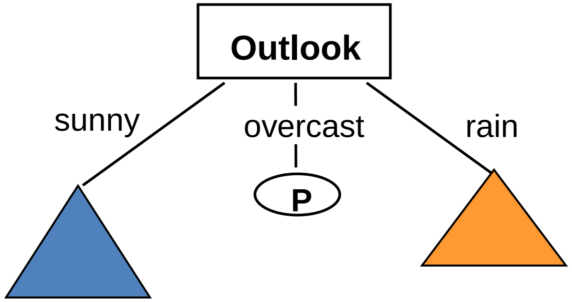
$$\boxed{Gain(S_1) = H(S_1) - H(S_1 \mid S_2)}$$

  - Rationale behind information gain:
    - H(S) represents degree of uncertainty. When one event always occurs, i.e. the least uncertain, H(S) = 0. When all events have equal chance to occur, i.e. the most uncertain, H(S) = log N.
    - Gain(S_1) signifies a reduction of uncertainty in system S_1 before and after system S_2 is present.
    - *In classification, Gain(A) represents a reduction of uncertainty over class attribute before and after attribute A is chosen as the root*

ID3 Algorithm

- **Algorithm Illustration**

Outlook	Temperature	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



$$H(S) = \sum_{k=1}^{n} p(E_k) \cdot I(E_k) = -\sum_{k=1}^{n} p(E_k) \cdot \mathbf{log}\, p(E_k)$$

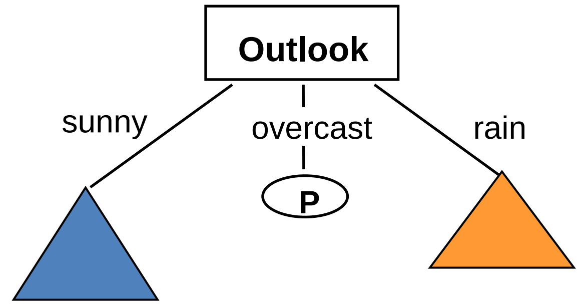
$$H(Class) = -\frac{9}{9+5} \cdot \log_2 \frac{9}{9+5} - \frac{5}{9+5} \cdot \log_2 \frac{5}{9+5} \approx 0.94$$

ID3 Algorithm

- **Algorithm Illustration**

$$H(Class) = -\frac{9}{9+5}\cdot\log_2\frac{9}{9+5} - \frac{5}{9+5}\cdot\log_2\frac{5}{9+5} \approx 0.94$$

Outlook	Temperature	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	Temperature	Humidity	Windy	Class
sunny	hot	high	FALSE	N
sunny	hot	high	TRUE	N
sunny	mild	high	FALSE	N
sunny	cool	normal	FALSE	P
sunny	mild	normal	TRUE	P

Outlook	Temperature	Humidity	Windy	Class
overcast	hot	high	FALSE	P
overcast	cool	normal	TRUE	P
overcast	mild	high	TRUE	P
overcast	hot	normal	FALSE	P

Outlook	Temperature	Humidity	Windy	Class
rain	mild	high	FALSE	P
rain	cool	normal	FALSE	P
rain	cool	normal	TRUE	N
rain	mild	normal	FALSE	P
rain	mild	high	TRUE	N

$$H(S) = \sum_{k=1}^{n} p(E_k)\cdot I(E_k) = -\sum_{k=1}^{n} p(E_k)\cdot\mathbf{log}\, p(E_k)$$

$$H(S) = \sum_{k=1}^{n} p(E_k)\cdot I(E_k) = -\sum_{k=1}^{n} p(E_k)\cdot\mathbf{log}\, p(E_k)$$

$$H(S) = \sum_{k=1}^{n} p(E_k)\cdot I(E_k) = -\sum_{k=1}^{n} p(E_k)\cdot\mathbf{log}\, p(E_k)$$

$$H(Class\,|\,Outlook = sunny) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} \approx 0.971$$

$$H(Class\,|\,Outlook = overcast) = -\frac{4}{4}\log_2\frac{4}{4} = 0$$

$$H(Class\,|\,Outlook = rain) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} \approx 0.971$$

$$H(Class\,|\,Outlook) = \frac{5}{14}H(Class\,|\,Outlook = sunny) + \frac{4}{14}H(Class\,|\,Outlook = overcast) + \frac{5}{14}H(Class\,|\,Outlook = rain)$$

$$\backslash\quad H(Class\,|\,Outlook) = \frac{5}{14}\times 0.971 + \frac{5}{14}\times 0 + \frac{5}{14}\times 0.971 = 0.694$$

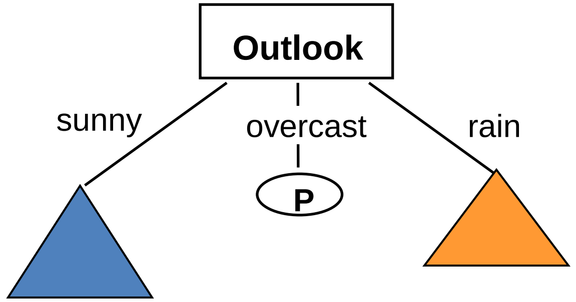
$$Gain(Outlook) = H(Class) - H(Class\,|\,Outlook) = 0.94 - 0.694 = 0.246$$

ID3 Algorithm

- **Algorithm Illustration**

Outlook	Temperature	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

Gain(Outlook) = 0.246 bits
Gain(Temperature) = 0.029 bits
Gain(Humidity) = 0.151 bits
Gain(Windy) = 0.048 bits
∴ Outlook is chosen as the root.



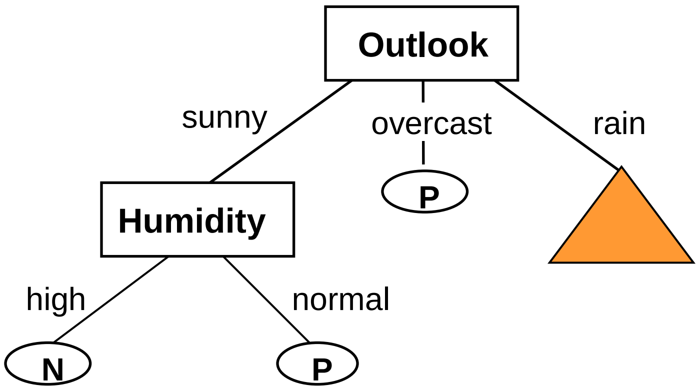
ID3 Algorithm

- **Algorithm Illustration**

Training Set(outlook=sunny)

Temperature	Humidity	Windy	Class
hot	high	FALSE	N
hot	high	TRUE	N
mild	high	FALSE	N
cool	normal	FALSE	P
mild	normal	TRUE	P

Gain(Temperature) = 0.571 bits
Gain(Humidity) = 0.971 bits
Gain(Windy) = 0.020 bits
∴ Humidity is chosen as the root.



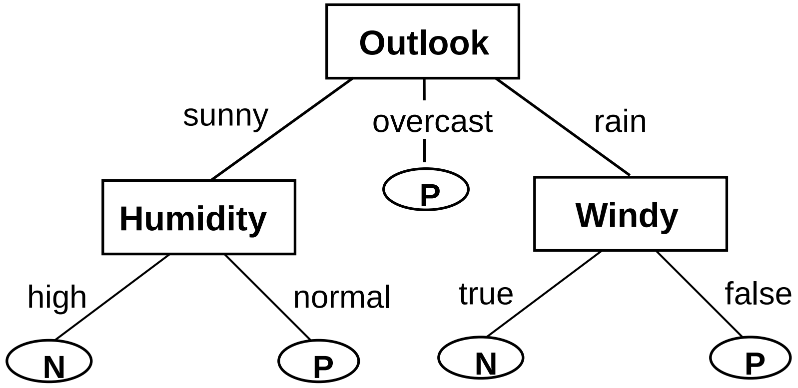
ID3 Algorithm

- **Algorithm Illustration**

Training Set(outlook=rain)

Temperature	Humidity	Windy	Class
mild	high	FALSE	P
cool	normal	FALSE	P
cool	normal	TRUE	N
mild	normal	FALSE	P
mild	high	TRUE	N

Gain(Temperature) = 0.020 bits
Gain(Humidity) = 0.020 bits
Gain(Windy) = 0.971 bits
∴ Windy is chosen as the root.



Attribute Selection Measures

- **Other Attribute Selection Measures**
  - **Information Gain Ratio**
    - GainRatio(A) = Gain(A)/H(A), e.g. GainRatio(Outlook) ≈ 0.156
      - Used to avoid (by normalization) bias towards attributes with many values
  - **Gini Index of Impurity**
    - Use Gini impurity function: Based on the concept of reducing impurity (mixture of different classes) of data set by selecting a right attribute
  - **Chi-square (χ^2) Statistic**
    - Based on the concept of measuring the degree of dependence of class to a chosen attribute
  - **and others..**

Chi-square (χ^2) statistic

- **χ^2**: measure for the degree of association/dependence between two categorical variables; a given attribute and the class variable.

- Given N examples of w classes, C_1, C_2, …, C_w and an attribute A of v values, a_1, a_2, …, a_v, the (χ^2) function is defined as follows:

$$\chi^2 = \sum_{j=1}^{v} \sum_{i=1}^{w} \frac{(x_{ij} - E_{ij})^2}{E_{ij}}$$

where x_{ij} is the actual frequency that examples have attribute value a_j and class C_i, and E_{ij} is the expected frequency.

with the assumption that the class and the attribute are not dependant on each other, $E_{ij} = \frac{n_i \times n_j}{N}$, where
n_i is the number of records with class Ci
n_j is the number of records with attribute A with value aj

Chi-square (χ^2) statistic

$$\chi^2 = \sum_{j=1}^{v} \sum_{i=1}^{w} \frac{(x_{ij} - E_{ij})^2}{E_{ij}}$$

$n_P = 9$ # of records with class P

$n_{sunny} = 5$ # of records with Attribute=sunny

$x_{P,sunny} = 2$

$$E_{P,sunny} = \frac{n_P \times n_{sunny}}{n} = \frac{9 \times 5}{14}$$

- $C = \{P, N\}$ .. $A = \{sunny, overcast, rain\}$
- $$\chi^2(Outlook) = \left(\frac{\left(\left(2 - \frac{9\times5}{14}\right)^2\right)}{\left(\frac{9\times5}{14}\right)} + \frac{\left(\left(3 - \frac{5\times5}{14}\right)^2\right)}{\left(\frac{5\times5}{14}\right)} \right) + \left(\frac{\left(\left(4 - \frac{9\times4}{14}\right)^2\right)}{\left(\frac{9\times4}{14}\right)} \right) + \left(\frac{\left(\left(3 - \frac{9\times5}{14}\right)^2\right)}{\left(\frac{9\times5}{14}\right)} + \frac{\left(\left(2 - \frac{5\times5}{14}\right)^2\right)}{\left(\frac{5\times5}{14}\right)} \right) \approx 29.653$$

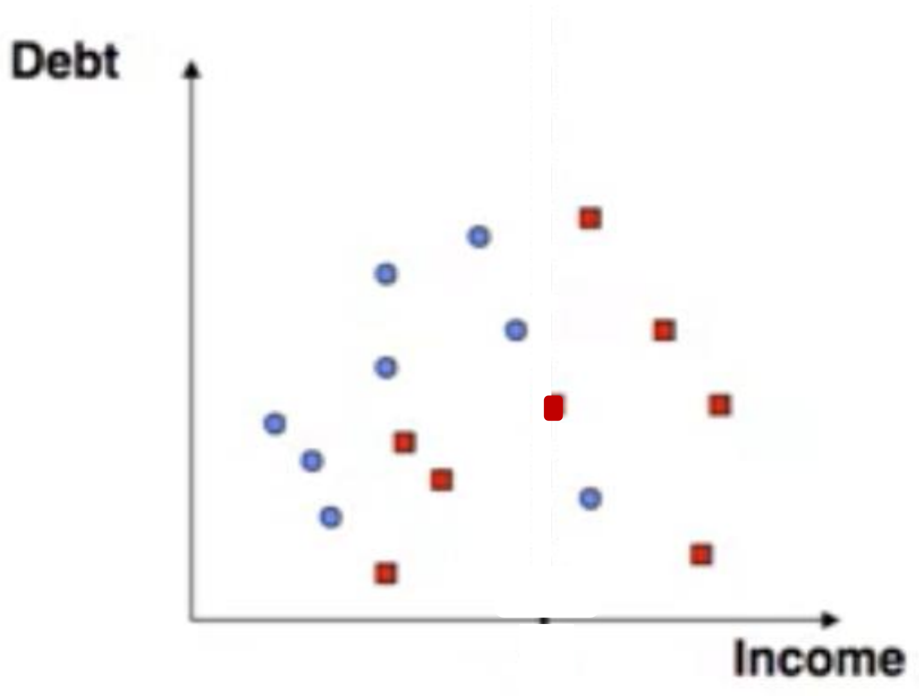
- Similarly,
  - $\chi^2(Temperature) \approx 7.985$,
  - $\chi^2(Humidity) \approx 39.2$
  - $\chi^2(Windy) \approx 13.067$

- Thus, Humidity is chosen as the root of the tree.

Outlook	Temperature	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

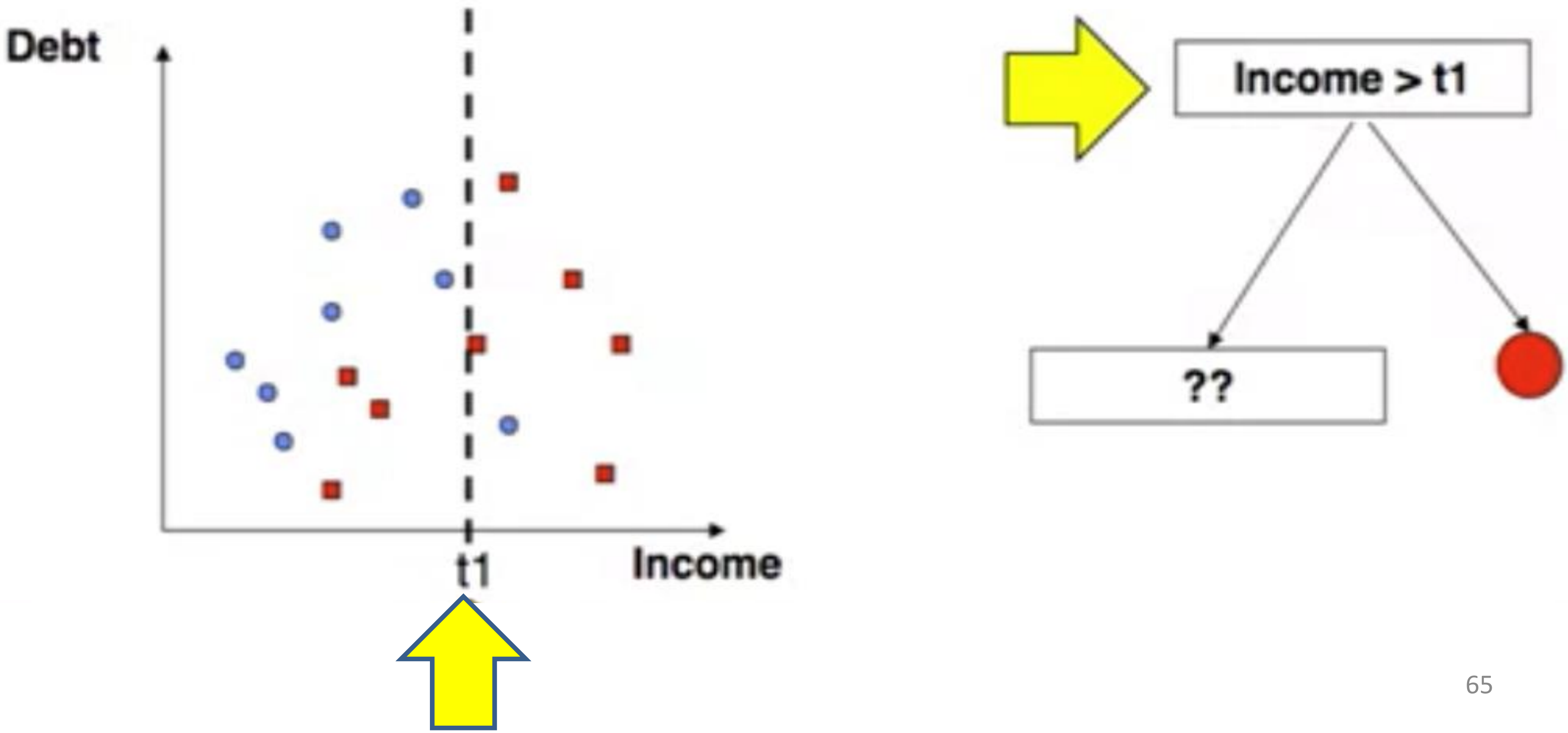
Decision Boundries (Univeriate)

- Ex.: classify loan applicants to:
  → Not-/likely to repay, based on income and existing debt



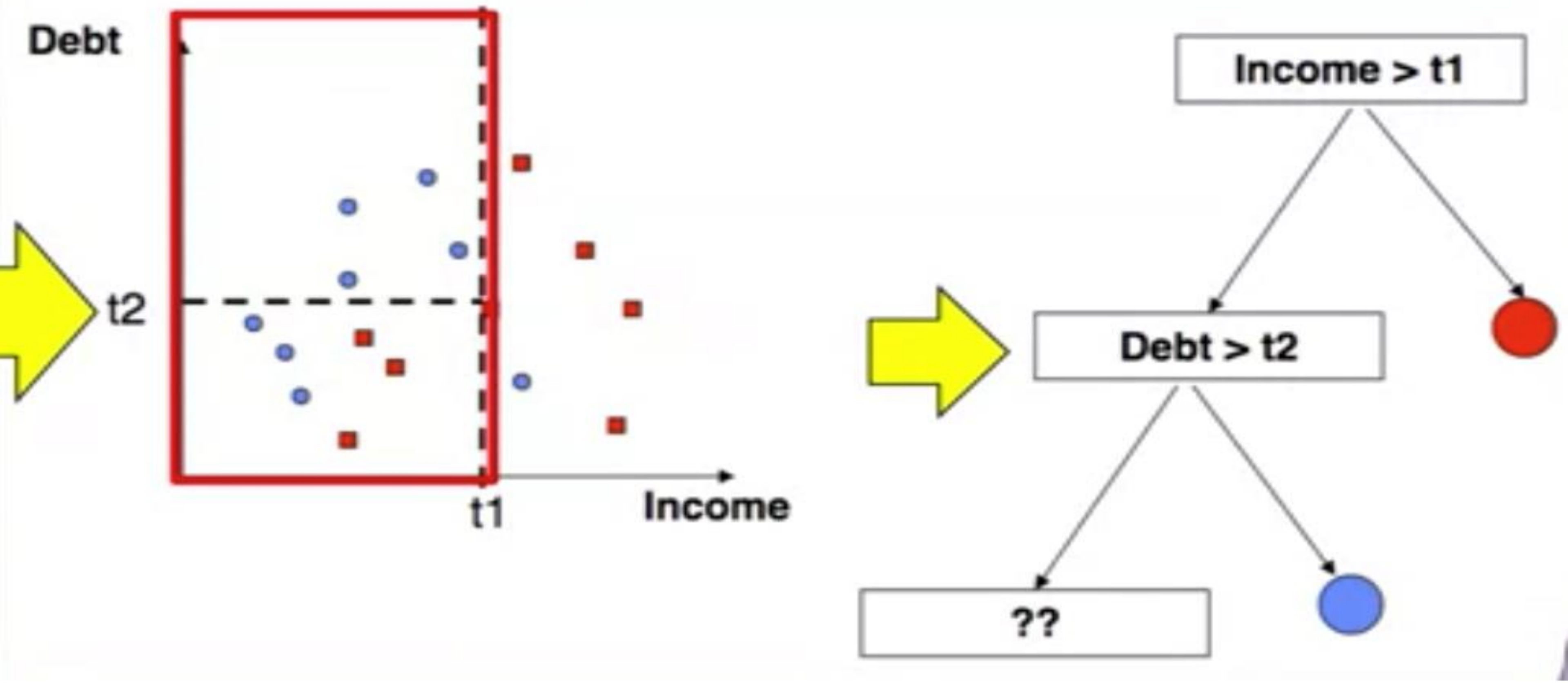
Decision Boundries (Univeriate)

- Split 1



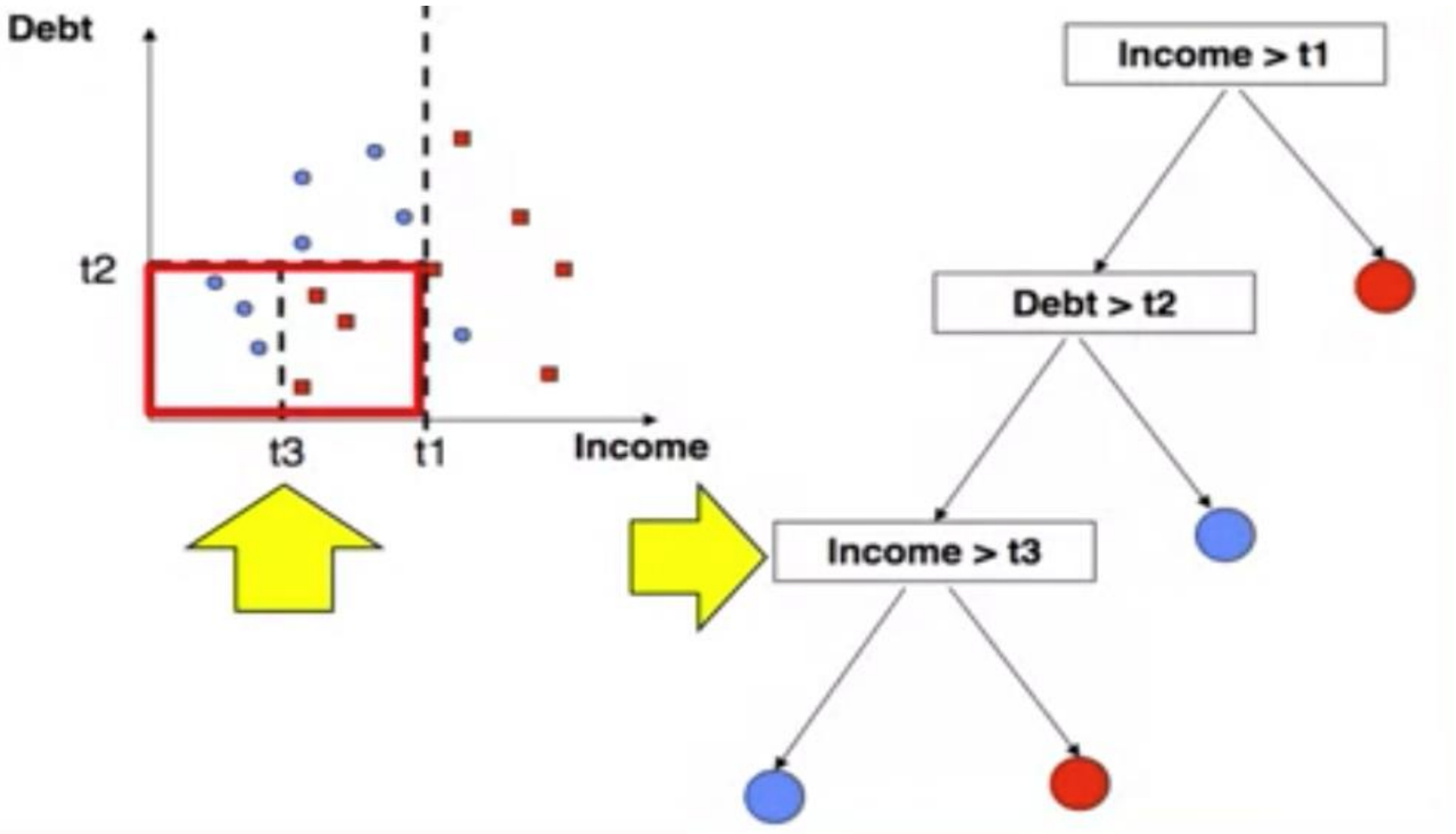
Decision Boundries (Univeriate)

- Split 2



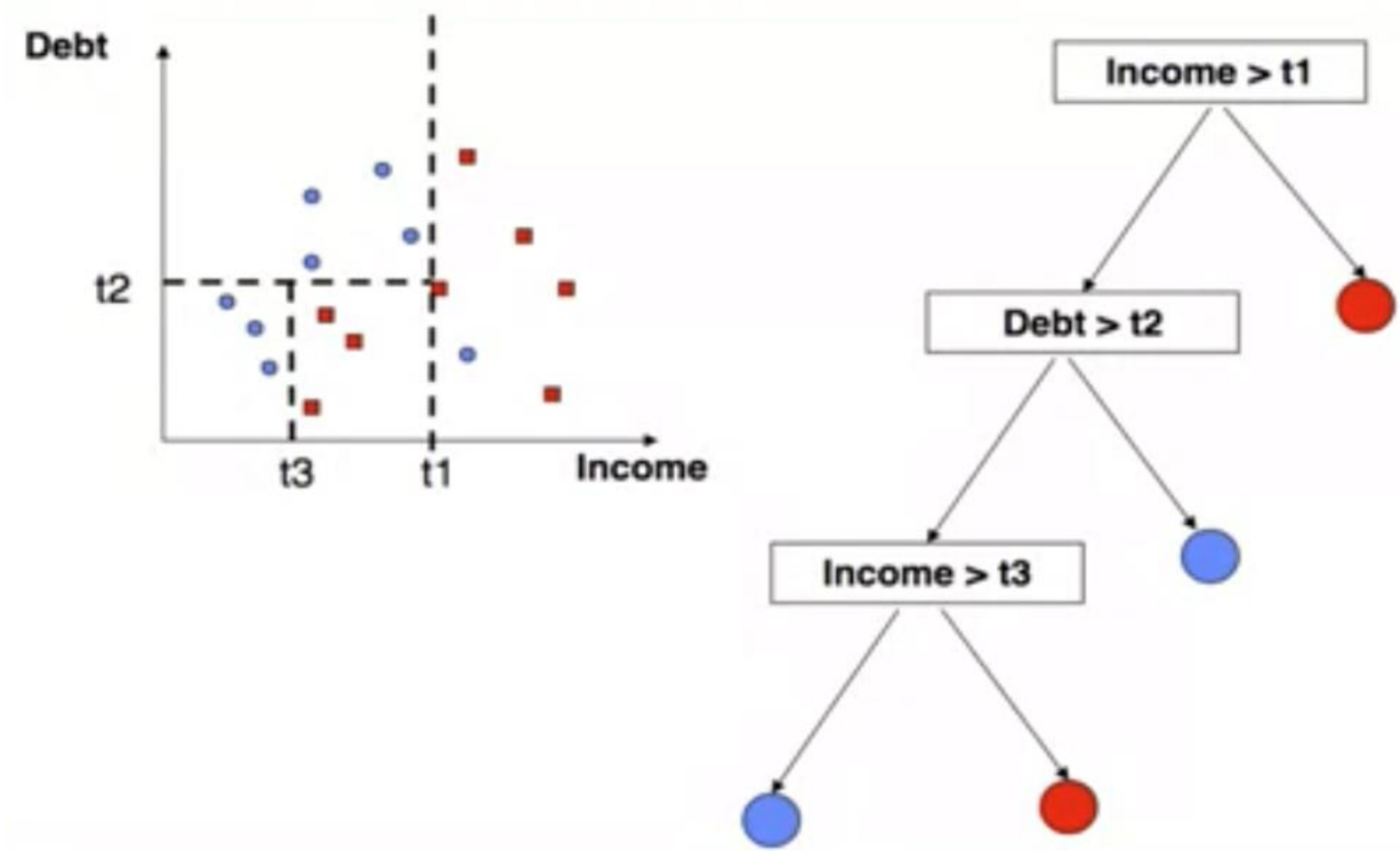
Decision Boundries (Univeriate)

- Split 3



Decision Boundries (Univeriate)

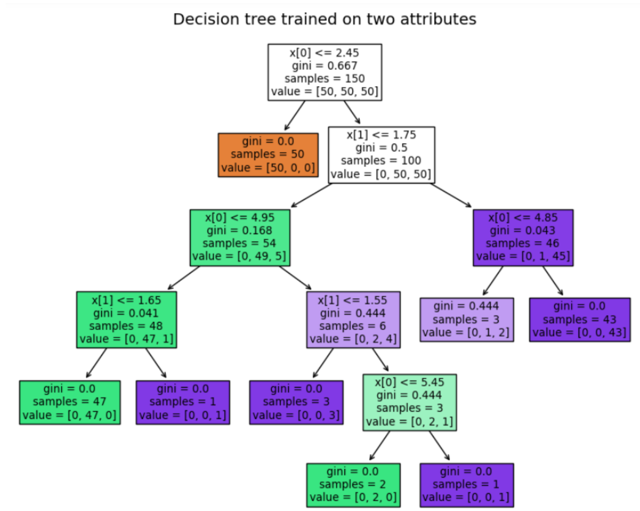
- Resulting model
  - Boundaries are "Rectilinear" = Parallel to the axes



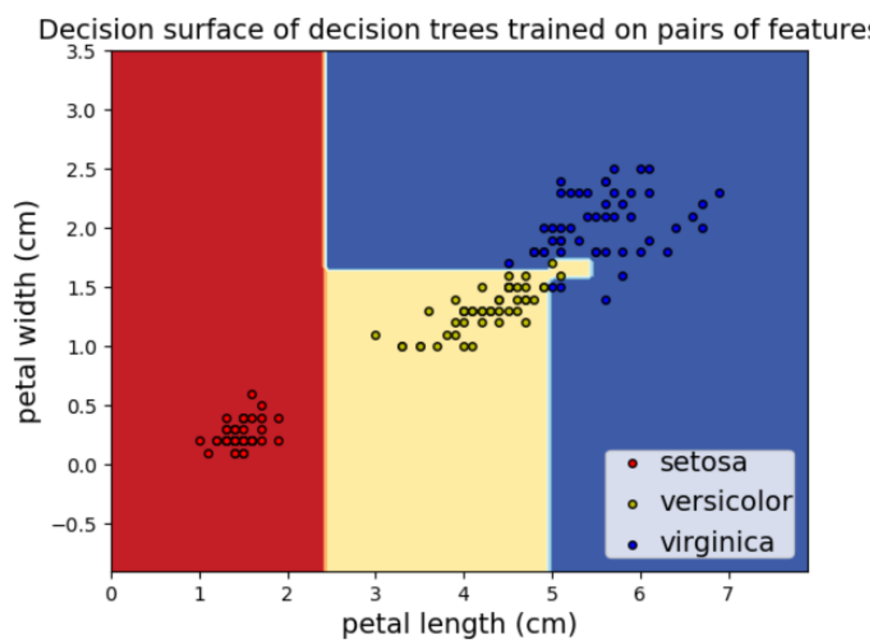
Decision Boundries (Univeriate)

In [27]:

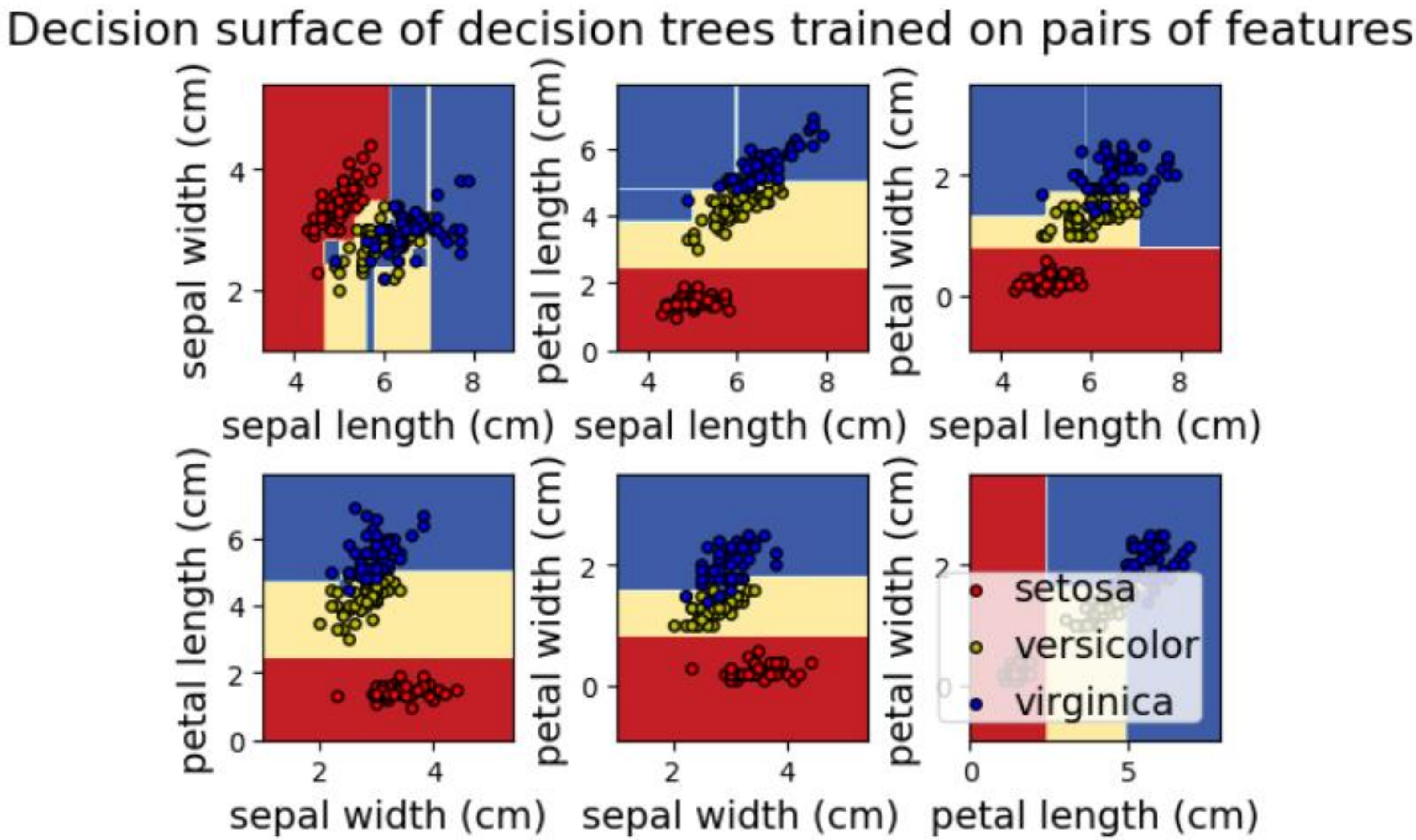
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.inspection import DecisionBoundaryDisplay

# Parameters
n_classes = 3
plot_colors = "ryb"
plot_step = 0.02

pair_of_columns = [2, 3]

# We only take the two corresponding features
X = iris.data.values[:, pair]
y = iris.target
# Train
tree_clf = DecisionTreeClassifier().fit(X, y)

plt.figure(figsize=(10,8))
plot_tree(tree_clf, filled=True)
plt.title("Decision tree trained on two attributes")
plt.show()
```




```python
ax = plt.subplot(1, 1, 1)
# Plot the decision boundary
DecisionBoundaryDisplay.from_estimator(tree_clf, X, cmap=plt.cm.RdYlBu, response_method="predict",
        ax=ax,
        xlabel=iris.feature_names[pair[0]],
        ylabel=iris.feature_names[pair[1]],
    )

# Plot the training points
for i, color in zip(range(n_classes), plot_colors):
    idx = np.where(y == i)
    plt.scatter(X[idx, 0], X[idx, 1], c=color, label=iris.target_names[i],
            cmap=plt.cm.RdYlBu, edgecolor="black", s=15,)

plt.title("Decision surface of decision trees trained on pairs of features")
plt.legend(loc="lower right", borderpad=0, handletextpad=0)
plt.show()
```



Decision Boundries (Univeriate)



Decision Boundries (Multivariate/Multi-variable)

- A multi-variable split, in the form of a linear equation* is more general
  - Takes the form:

    $$\boldsymbol{w}_m^T \boldsymbol{x} + w_0 > 0$$

    where $\boldsymbol{w}_m^T$ is the vector of weights
  - Requires all attributes to be numeric
  - CART was the original algorithm for generating multi-variable trees
  - Generating the linear equation is not cheap
- (*) Doesn't have to be linear!