

Lecture 4: 20 January, 2026

Madhavan Mukund

<https://www.cmi.ac.in/~madhavan>

Data Mining and Machine Learning
January–April 2026

Decision tree algorithm

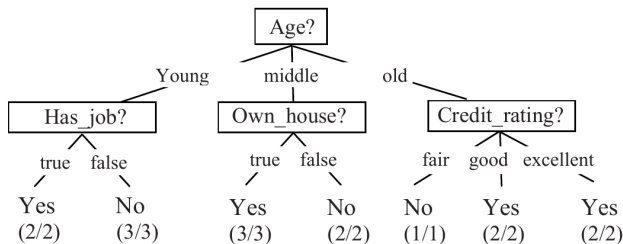
A : current set of attributes

Pick $a \in A$, create children corresponding to resulting partition with attributes $A \setminus \{a\}$

Stopping criterion:

- Current node has uniform class label
- A is empty — no more attributes to query

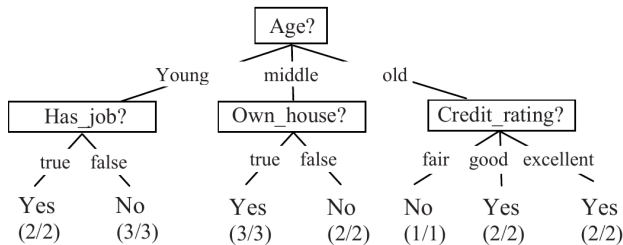
If a leaf node is not uniform, use majority class as prediction



- Non-uniform leaf node — identical combination of attributes, but different classes
- Attributes do not capture all criteria used for classification

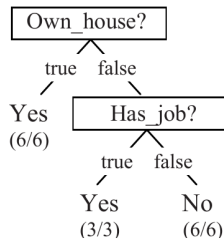
Decision trees

- Tree is not unique
- Which tree is better?
- Prefer small trees
 - Explainability
 - Generalize better (see later)



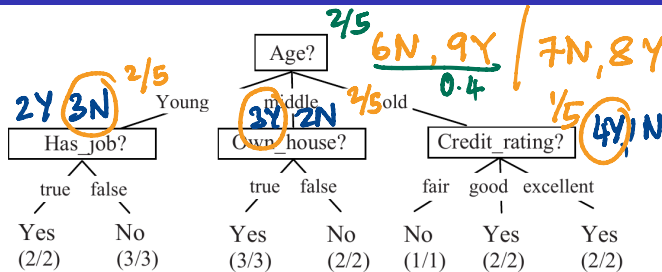
Unfortunately

- Finding smallest tree is NP-complete — for any definition of “smallest”
- Instead, greedy heuristic



Greedy heuristic

- Goal: partition with uniform category — **pure** leaf
- Impure node — best prediction is majority value
- Minority ratio is **impurity**
- Heuristic: reduce impurity as much as possible



$$\frac{1}{3} \times \frac{2}{5} + \frac{1}{3} \times \frac{2}{5} + \frac{1}{3} \times \frac{1}{5} = \frac{5}{15} = \frac{1}{3} = 0.33$$

Own_house?

$$0.4, \frac{2}{3}$$

Weighted average

$$0 + \frac{3}{15} = \frac{1}{5} = 0.2$$

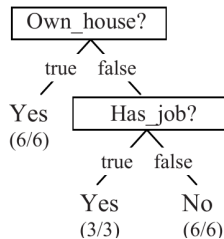
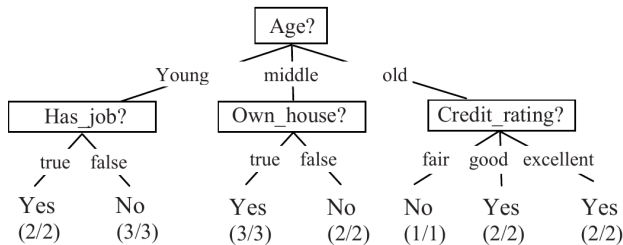
$$\frac{6}{15} \times 0$$

Has_job?

$$9 \times \frac{3}{9}$$

Greedy heuristic

- Goal: partition with uniform category — **pure** leaf
- Impure node — best prediction is majority value
- Minority ratio is **impurity**
- Heuristic: reduce impurity as much as possible
- For each attribute, compute weighted average impurity of children
- Choose the minimum

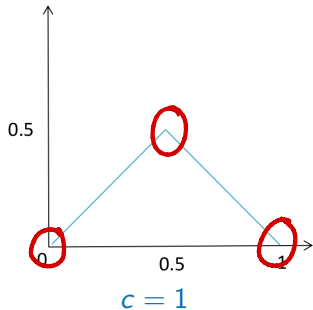


Greedy heuristic — misclassification rate

- Minority ratio is **misclassification rate**

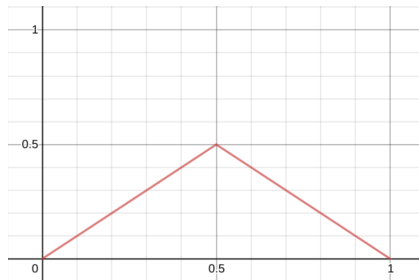
Greedy heuristic — misclassification rate

- Minority ratio is **misclassification rate**
- Misclassification rate is **linear**
 - $c \in \{0, 1\}$ **2 classes**
 - x-axis: fraction of inputs with $c = 1$
 - y-axis: impurity



A better impurity function

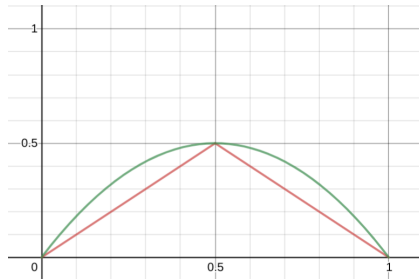
- Misclassification rate is linear



$$c = 1$$

A better impurity function

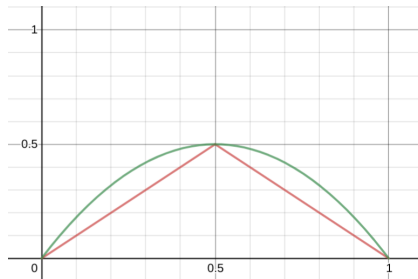
- Misclassification rate is linear
- Impurity measure that increases more sharply performs better, empirically



$$c = 1$$

A better impurity function

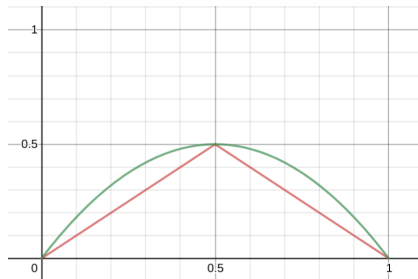
- Misclassification rate is linear
- Impurity measure that increases more sharply performs better, empirically
- Entropy — [Quinlan]



$c = 1$

A better impurity function

- Misclassification rate is linear
- Impurity measure that increases more sharply performs better, empirically
- Entropy — [Quinlan]
- Gini index — [Breiman]



$c = 1$

Entropy

- Information theoretic measure of randomness
- Minimum number of bits to transmit a message — [Shannon]

Entropy

- Information theoretic measure of randomness
- Minimum number of bits to transmit a message — [Shannon]
- n data items
 - n_0 with $c = 0$, $p_0 = n_0/n$
 - n_1 with $c = 1$, $p_1 = n_1/n$

Entropy

- Information theoretic measure of randomness
- Minimum number of bits to transmit a message — [Shannon]
- n data items
 - n_0 with $c = 0$, $p_0 = n_0/n$
 - n_1 with $c = 1$, $p_1 = n_1/n$
- Entropy
 $E = -(p_0 \log_2 p_0 + p_1 \log_2 p_1)$

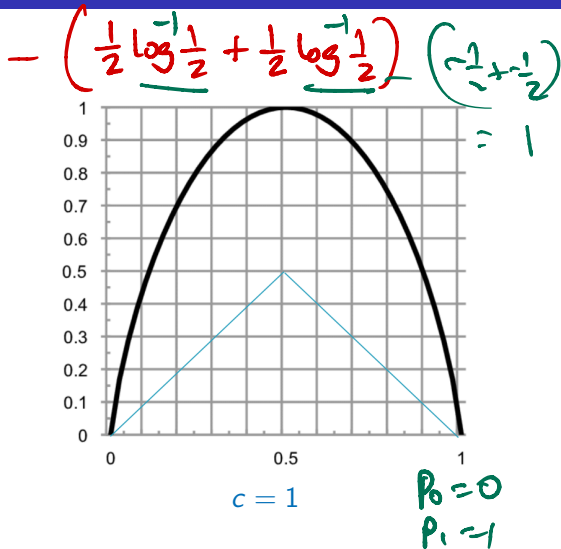
negative

$$0 \leq p_0, p_1 \leq 1$$

$$\log_2 p_0 \leq 0$$

Entropy

- Information theoretic measure of randomness
- Minimum number of bits to transmit a message — [Shannon]
- n data items
 - n_0 with $c = 0$, $p_0 = n_0/n$
 - n_1 with $c = 1$, $p_1 = n_1/n$
- Entropy
$$E = -(p_0 \log_2 p_0 + p_1 \log_2 p_1)$$
- Minimum when $p_0 = 1, p_1 = 0$ or vice versa — note, declare $0 \log_2 0$ to be 0
- Maximum when $p_0 = p_1 = 0.5$



Gini Index

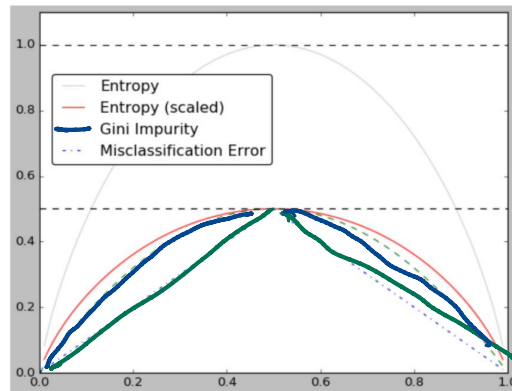
- Measure of unequal distribution of wealth
- Economics — [Corrado Gini]
- As before, n data items
 - n_0 with $c = 0$, $p_0 = n_0/n$
 - n_1 with $c = 1$, $p_1 = n_1/n$

Gini Index

- Measure of unequal distribution of wealth
- Economics — [Corrado Gini]
- As before, n data items
 - n_0 with $c = 0$, $p_0 = n_0/n$
 - n_1 with $c = 1$, $p_1 = n_1/n$
- Gini Index $G = 1 - (p_0^2 + p_1^2)$

Gini Index

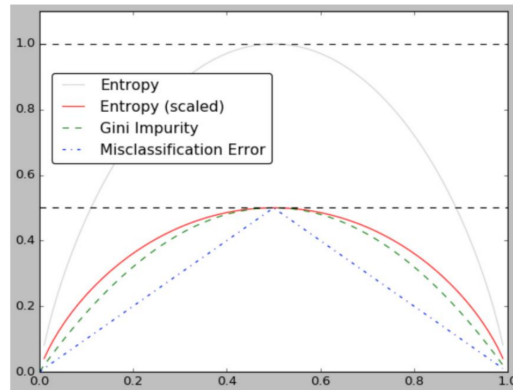
- Measure of unequal distribution of wealth
- Economics — [Corrado Gini]
- As before, n data items
 - n_0 with $c = 0$, $p_0 = n_0/n$
 - n_1 with $c = 1$, $p_1 = n_1/n$
- Gini Index $G = 1 - (p_0^2 + p_1^2)$
- $G = 0$ when $p_0 = 0$, $p_1 = 0$ or v.v.
 $G = 0.5$ when $p_0 = p_1 = 0.5$



$c = 1$

Gini Index

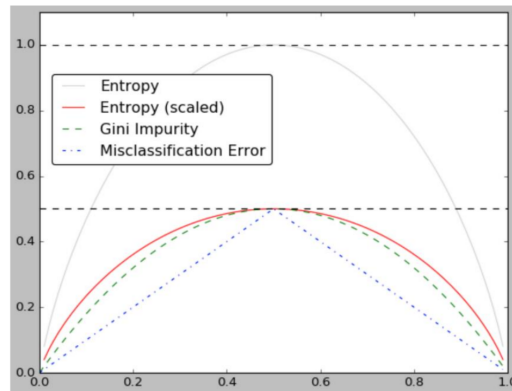
- Measure of unequal distribution of wealth
- Economics — [Corrado Gini]
- As before, n data items
 - n_0 with $c = 0$, $p_0 = n_0/n$
 - n_1 with $c = 1$, $p_1 = n_1/n$
- **Gini Index** $G = 1 - (p_0^2 + p_1^2)$
- $G = 0$ when $p_0 = 0$, $p_1 = 0$ or v.v.
 $G = 0.5$ when $p_0 = p_1 = 0.5$
- Entropy curve is slightly steeper, but Gini index is easier to compute



$c = 1$

Gini Index

- Measure of unequal distribution of wealth
- Economics — [Corrado Gini]
- As before, n data items
 - n_0 with $c = 0$, $p_0 = n_0/n$
 - n_1 with $c = 1$, $p_1 = n_1/n$
- **Gini Index** $G = 1 - (p_0^2 + p_1^2)$
- $G = 0$ when $p_0 = 0$, $p_1 = 0$ or v.v.
 $G = 0.5$ when $p_0 = p_1 = 0.5$
- Entropy curve is slightly steeper, but Gini index is easier to compute
- Decision tree libraries usually use Gini index



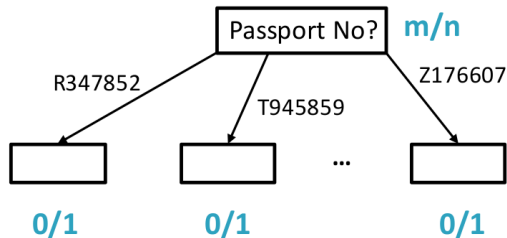
$c = 1$

Information gain

- Greedy strategy: choose attribute to maximize reduction in impurity — maximize **information gain** — from entropy

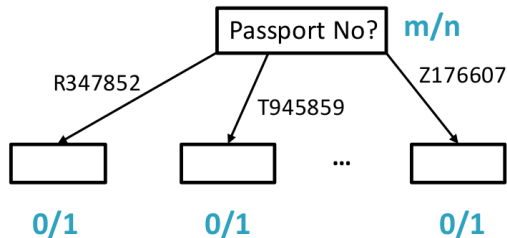
Information gain

- Greedy strategy: choose attribute to maximize reduction in impurity — maximize **information gain**
- Suppose an attribute is a unique identifier
 - Roll number, passport number, Aadhaar ...
- Querying this attribute produces partitions of size 1
 - Each partition guaranteed to be pure
 - New impurity is zero



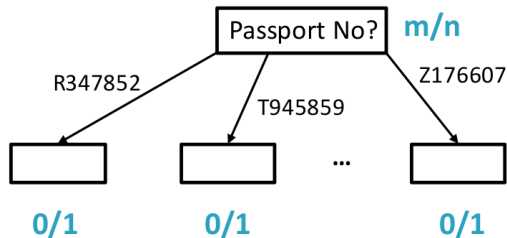
Information gain

- Greedy strategy: choose attribute to maximize reduction in impurity — maximize **information gain**
- Suppose an attribute is a unique identifier
 - Roll number, passport number, Aadhaar ...
- Querying this attribute produces partitions of size 1
 - Each partition guaranteed to be pure
 - New impurity is zero
- Maximum possible impurity reduction, but useless!



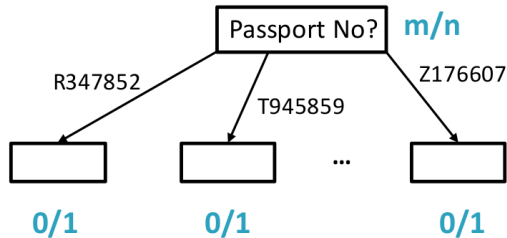
Information gain

- Tree building algorithm blindly picks attribute that maximizes information gain
- Need a correction to penalize attributes with highly scattered attributes
- Extend the notion of impurity to attributes



Attribute Impurity

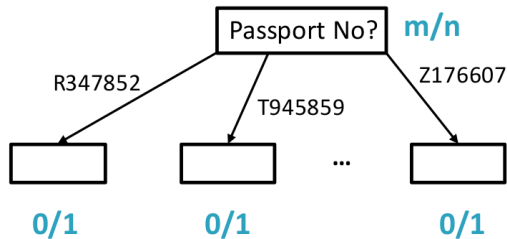
- Attribute takes values $\{v_1, v_2, \dots, v_k\}$
- v_i appears n_i times across n rows
- $p_i = n_i/n$



Attribute Impurity

- Attribute takes values $\{v_1, v_2, \dots, v_k\}$
- v_i appears n_i times across n rows
- $p_i = n_i/n$
- Entropy across k values

$$-\sum_{i=1}^k p_i \log_2 p_i$$



Attribute Impurity

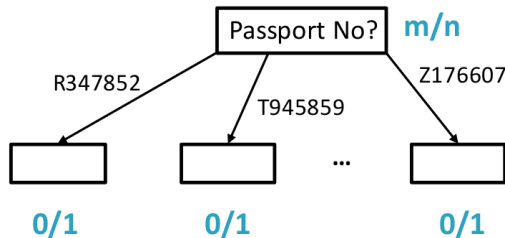
- Attribute takes values $\{v_1, v_2, \dots, v_k\}$
- v_i appears n_i times across n rows
- $p_i = n_i/n$

- Entropy across k values

$$-\sum_{i=1}^k p_i \log_2 p_i$$

- Gini index across k values

$$1 - \sum_{i=1}^k p_i^2$$



Attribute Impurity

- Extreme case, each $p_i = 1/n$

Attribute Impurity

- Extreme case, each $p_i = 1/n$
- Entropy

$$-\sum_{i=1}^n \frac{1}{n} \log_2 \frac{1}{n} = -n \cdot \frac{1}{n} (-\log_2 n) = \log_2 n$$

Attribute Impurity

- Extreme case, each $p_i = 1/n$

- Entropy

$$-\sum_{i=1}^n \frac{1}{n} \log_2 \frac{1}{n} = -n \cdot \frac{1}{n} (-\log_2 n) = \log_2 n$$

- Gini index

$$1 - \sum_{i=1}^n \left(\frac{1}{n}\right)^2 = 1 - \frac{n}{n^2} = \frac{n-1}{n}$$

$$\frac{5}{6} < \frac{6}{7} < \frac{7}{8} \dots$$

Attribute Impurity

- Extreme case, each $p_i = 1/n$

- Entropy

$$-\sum_{i=1}^n \frac{1}{n} \log_2 \frac{1}{n} = -n \cdot \frac{1}{n} (-\log_2 n) = \log_2 n$$

- Gini index

$$1 - \sum_{i=1}^n \left(\frac{1}{n}\right)^2 = 1 - \frac{n}{n^2} = \frac{n-1}{n}$$

- Both increase as n increases

Attribute Impurity

- Extreme case, each $p_i = 1/n$

- Entropy

$$-\sum_{i=1}^n \frac{1}{n} \log_2 \frac{1}{n} = -n \cdot \frac{1}{n} (-\log_2 n) = \log_2 n$$

- Gini index

$$1 - \sum_{i=1}^n \left(\frac{1}{n}\right)^2 = 1 - \frac{n}{n^2} = \frac{n-1}{n}$$

- Both increase as n increases

Penalizing scattered attributes

- Divide information gain by attribute impurity

- Information gain ratio(A)

$$\frac{\text{Information-Gain}(A)}{\text{Impurity}(A)}$$

- Scattered attributes have high denominator, counteracting high numerator

Categorical vs numeric attributes

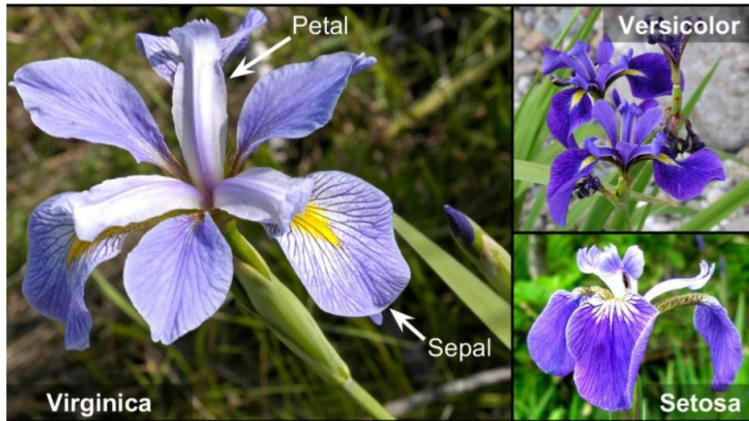
- So far, all attributes have been categorical
- What age groups make up young, middle, old?
- How are these boundaries defined?
- How do we query numerical attributes?
 - Height, weight, length, income,

ID	Age	Has_job	Own_house	Credit_rating	Class
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

Iris dataset

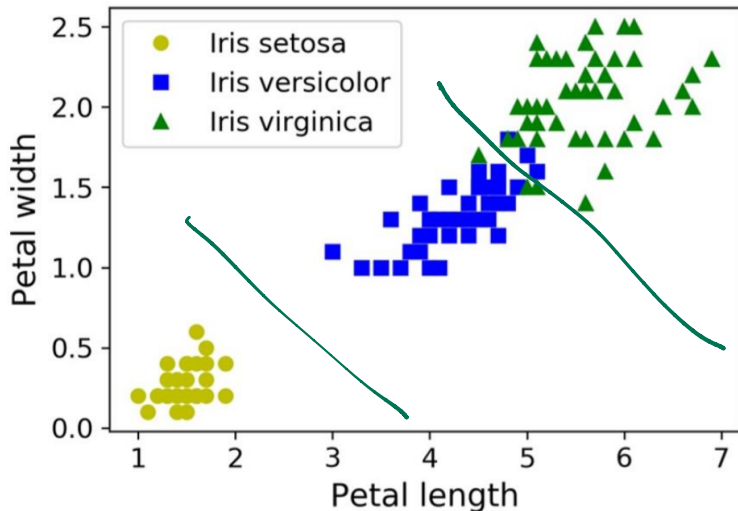
- Iris is a type of flower
- Three species: *iris setosa*, *iris versicolor*, *iris virginica*
- Dataset has sepal length and width and petal length and width for 150 flowers

Ronald Fisher



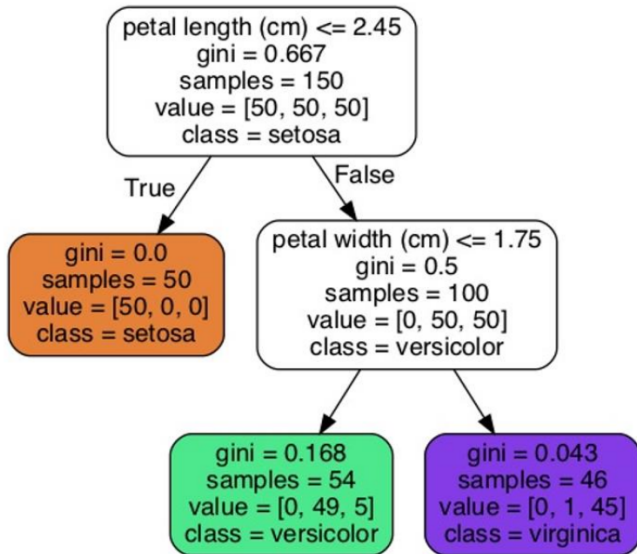
Iris dataset

- Iris is a type of flower
- Three species: *iris setosa*, *iris versicolor*, *iris virginica*
- Dataset has sepal length and width and petal length and width for 150 flowers
- Scatter plot for two attributes, petal length and petal width



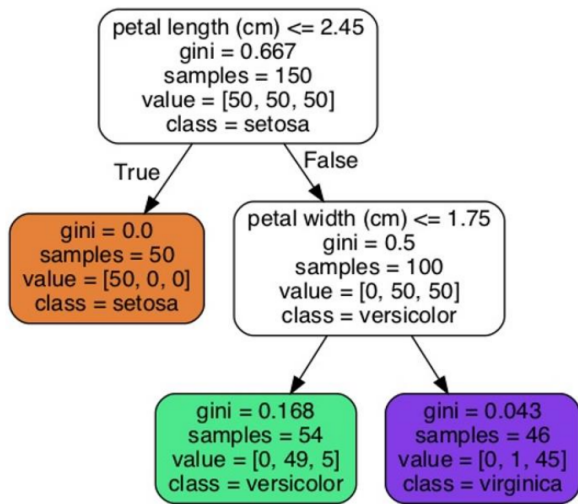
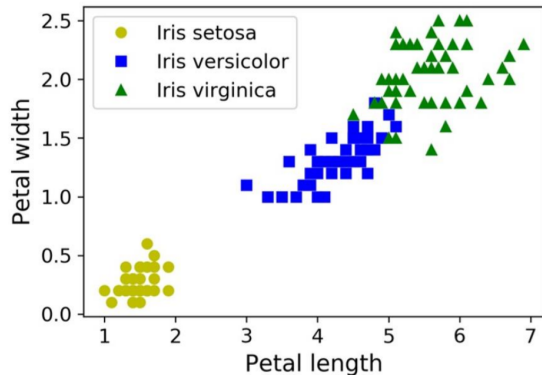
Iris dataset

- Iris is a type of flower
- Three species: *iris setosa*, *iris versicolor*, *iris virginica*
- Dataset has sepal length and width and petal length and width for 150 flowers
- Scatter plot for two attributes, petal length and petal width
- Decision tree for this data set



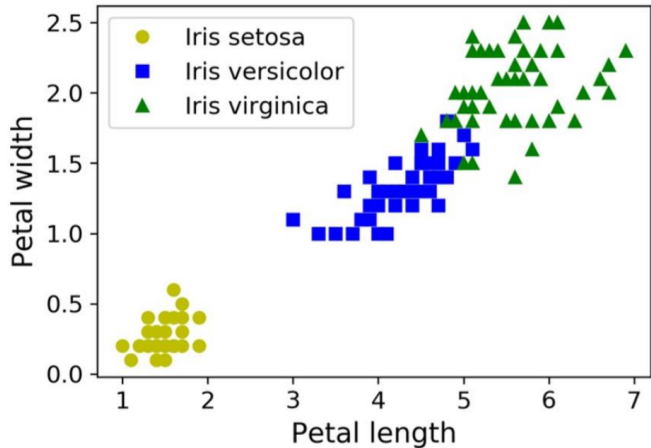
Decision tree for iris dataset

- Queries compare numerical attribute against a value
- How do we find these query values?



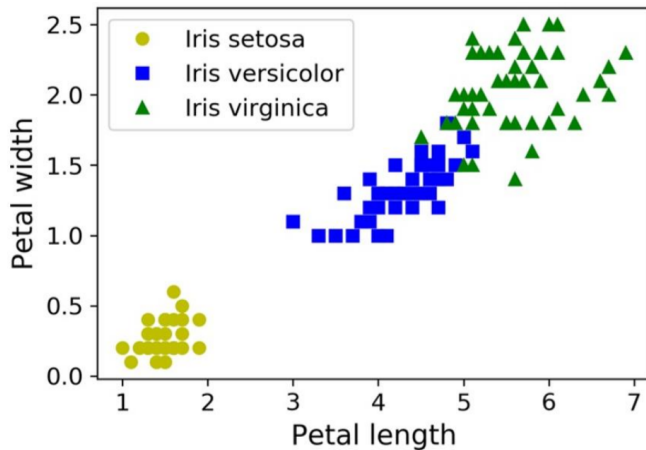
Querying numerical attributes

- Numerical attribute takes values in a range $[L, U]$
 - Petal length : $[1, 7]$
 - Petal width : $[0, 2.5]$



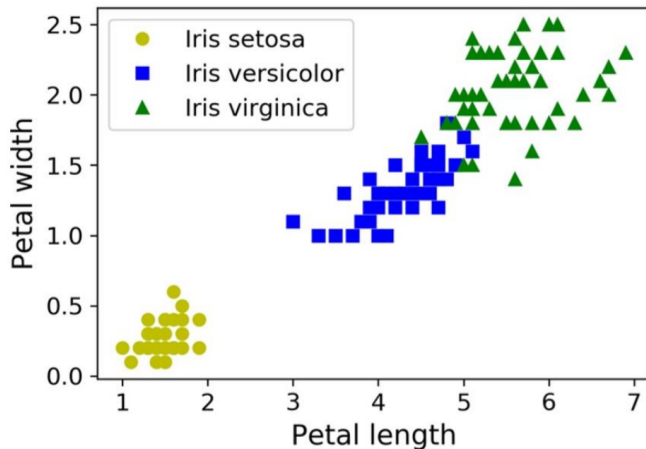
Querying numerical attributes

- Numerical attribute takes values in a range $[L, U]$
 - Petal length : $[1, 7]$
 - Petal width : $[0, 2.5]$
- Pick a value v in the range and check if $A \leq v$



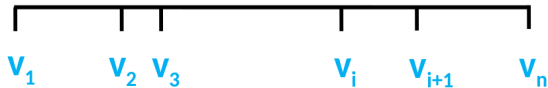
Querying numerical attributes

- Numerical attribute takes values in a range $[L, U]$
 - Petal length : $[1, 7]$
 - Petal width : $[0, 2.5]$
- Pick a value v in the range and check if $A \leq v$
- Infinitely many choices for v
- How do we pick a sensible one?



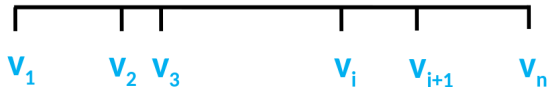
Querying numerical attributes

- Only n values for A in training data
 - Sort as $v_1 < v_2 < \dots < v_n$



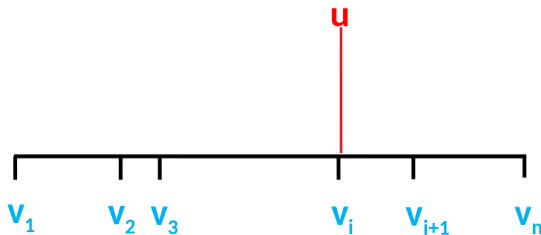
Querying numerical attributes

- Only n values for A in training data
 - Sort as $v_1 < v_2 < \dots < v_n$
- Consider interval $[v_i, v_{i+1}]$



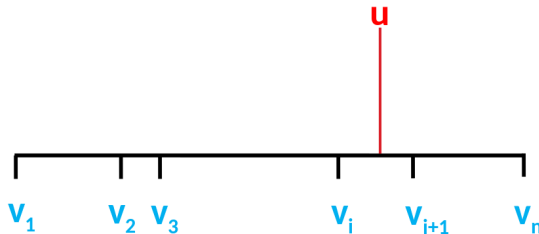
Querying numerical attributes

- Only n values for A in training data
 - Sort as $v_1 < v_2 < \dots < v_n$
- Consider interval $[v_i, v_{i+1}]$
- For each $v_i \leq u < v_{i+1}$, query $A \leq u$ gives the same answer



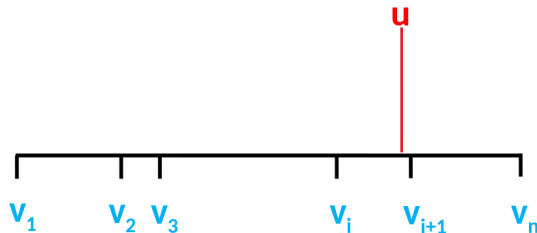
Querying numerical attributes

- Only n values for A in training data
 - Sort as $v_1 < v_2 < \dots < v_n$
- Consider interval $[v_i, v_{i+1}]$
- For each $v_i \leq u < v_{i+1}$, query $A \leq u$ gives the same answer



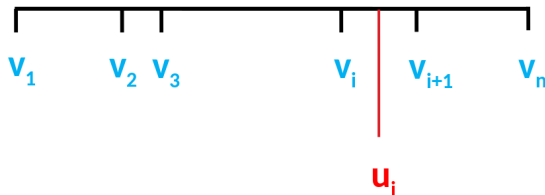
Querying numerical attributes

- Only n values for A in training data
 - Sort as $v_1 < v_2 < \dots < v_n$
- Consider interval $[v_i, v_{i+1}]$
- For each $v_i \leq u < v_{i+1}$, query $A \leq u$ gives the same answer
- Only $n-1$ useful intervals to check



Querying numerical attributes

- Only n values for A in training data
 - Sort as $v_1 < v_2 < \dots < v_n$
- Consider interval $[v_i, v_{i+1}]$
- For each $v_i \leq u < v_{i+1}$, query $A \leq u$ gives the same answer
- Only $n-1$ useful intervals to check
- Pick midpoint $u_i = (v_i + v_{i+1})/2$ as query value for each interval



Querying numerical attributes

- Pick midpoint $u_i = (v_i + v_{i+1})/2$ as query value for each interval
- Each query $A \leq u_i$ partitions training data



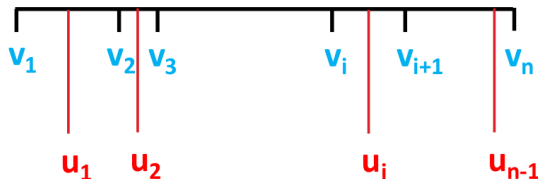
Querying numerical attributes

- Pick midpoint $u_i = (v_i + v_{i+1})/2$ as query value for each interval
- Each query $A \leq u_i$ partitions training data
- Choose the query $A \leq u_i$ with maximum information gain
- Assign this as the information gain for this attribute



Querying numerical attributes

- Pick midpoint $u_i = (v_i + v_{i+1})/2$ as query value for each interval
- Each query $A \leq u_i$ partitions training data
- Choose the query $A \leq u_i$ with maximum information gain
- Assign this as the information gain for this attribute
- Compare across all attributes and choose best one



Querying numerical attributes

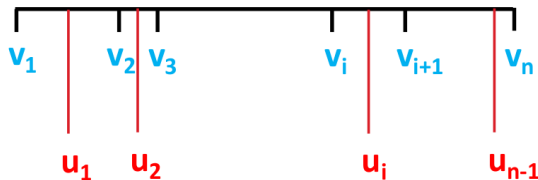
- Pick midpoint $u_i = (v_i + v_{i+1})/2$ as query value for each interval
- Each query $A \leq u_i$ partitions training data
- Choose the query $A \leq u_i$ with maximum information gain
- Assign this as the information gain for this attribute
- Compare across all attributes and choose best one



- Any point within an interval can be used

Querying numerical attributes

- Pick midpoint $u_i = (v_i + v_{i+1})/2$ as query value for each interval
- Each query $A \leq u_i$ partitions training data
- Choose the query $A \leq u_i$ with maximum information gain
- Assign this as the information gain for this attribute
- Compare across all attributes and choose best one



- Any point within an interval can be used
- May prefer endpoints — midpoints may not be meaningful values

Building a decision tree

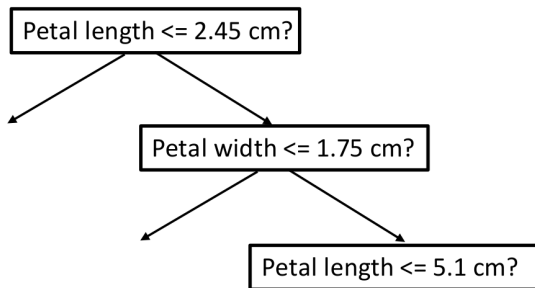
- For each numerical attribute, choose query $A \leq v$ with maximum information gain

Building a decision tree

- For each numerical attribute, choose query $A \leq v$ with maximum information gain
- Across all categorical and numerical attributes, choose the one with best information gain

Building a decision tree

- For each numerical attribute, choose query $A \leq v$ with maximum information gain
- Across all categorical and numerical attributes, choose the one with best information gain
- Categorical attributes can be queried only once on a path
- Numerical attributes can be queried repeatedly — interval to query keeps shrinking



Testing a supervised learning model

- How do we validate software?
 - Test suite of carefully selected inputs
 - Compare output with expected answers

Testing a supervised learning model

- How do we validate software?
 - Test suite of carefully selected inputs
 - Compare output with expected answers
- What about classification models?
 - By definition, deploy on data where the outcome is unknown
 - If expected answer available, have a deterministic solution, model not needed!

Testing a supervised learning model

- How do we validate software?
 - Test suite of carefully selected inputs
 - Compare output with expected answers
- What about classification models?
 - By definition, deploy on data where the outcome is unknown
 - If expected answer available, have a deterministic solution, model not needed!
- On what basis can we evaluate a supervised learning model?

Creating a test set

- Training data is labelled
 - No other source of inputs with expected answers

Creating a test set

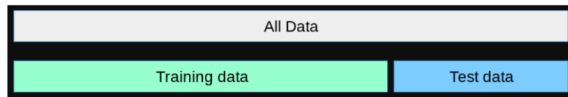
- Training data is labelled
 - No other source of inputs with expected answers
- Segregate some training data for testing
 - Terminology: **training set** and **test set**
 - Build model using training set, evaluate on test set

Creating a test set

- Training data is labelled
 - No other source of inputs with expected answers
- Segregate some training data for testing
 - Terminology: **training set** and **test set**
 - Build model using training set, evaluate on test set
- Creating the test set
 - Need to choose a random sample
 - Can further use **stratified sampling**, preserve relative ratios (e.g., age wise distribution)
 - ML libraries can do this automatically

Creating a test set

- How large should the test set be?
 - Typically 20-30% of labelled data
- Depends on labelled data available
 - Need enough training data to build the model

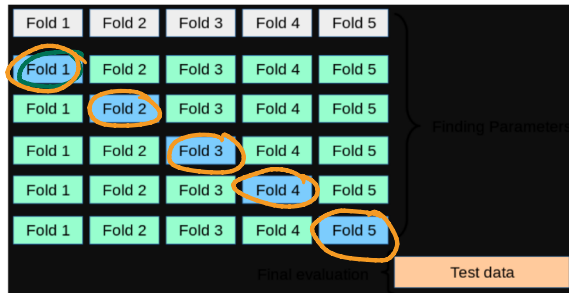
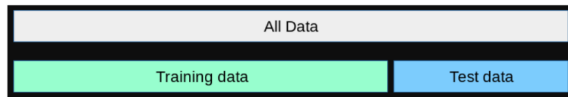


Creating a test set

- How large should the test set be?
 - Typically 20-30% of labelled data
- Depends on labelled data available
 - Need enough training data to build the model

Cross validation

- Partition labelled data into k chunks
- Hold out one chunk at a time
- Build k models, using $k-1$ chunks for training, 1 for testing
- Useful if labelled data is scarce

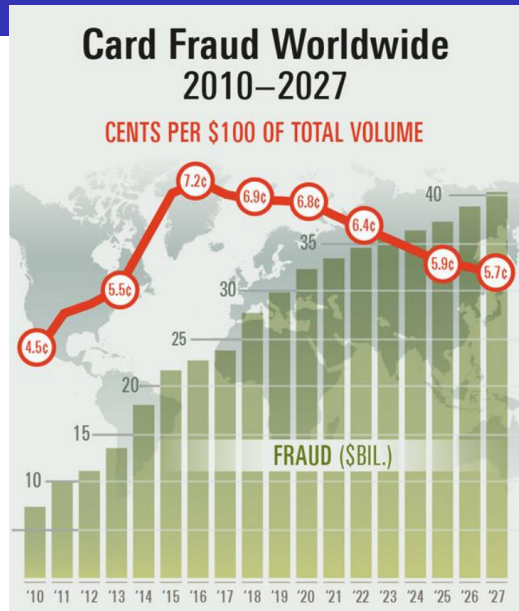


What are we measuring?

- Accuracy is an obvious measure
 - Fraction of inputs where classification is correct

What are we measuring?

- Accuracy is an obvious measure
 - Fraction of inputs where classification is correct
- Classifiers are often used in asymmetric situations
 - Less than 1% of credit card transactions are fraud



What are we measuring?

- Accuracy is an obvious measure
 - Fraction of inputs where classification is correct
- Classifiers are often used in asymmetric situations
 - Less than 1% of credit card transactions are fraud
- “Is this transaction a fraud?”
 - Trivial classifier — always answer “No”
 - More than 99% accurate, but useless!

