SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Established in collaboration with MIT

Classification

PROF. D. HERREMANS

50.038 Computational data science

Data science tasks

Task	Supervised Methods	Unsupervised Methods
*Classification	✓	
*Regression	✓	
Causal Modeling	✓	
Similarity Matching	✓	✓
Link Prediction	✓	✓
Data Reduction	✓	✓
*Clustering		✓
Co-occurrence Grouping		✓
Profiling		✓

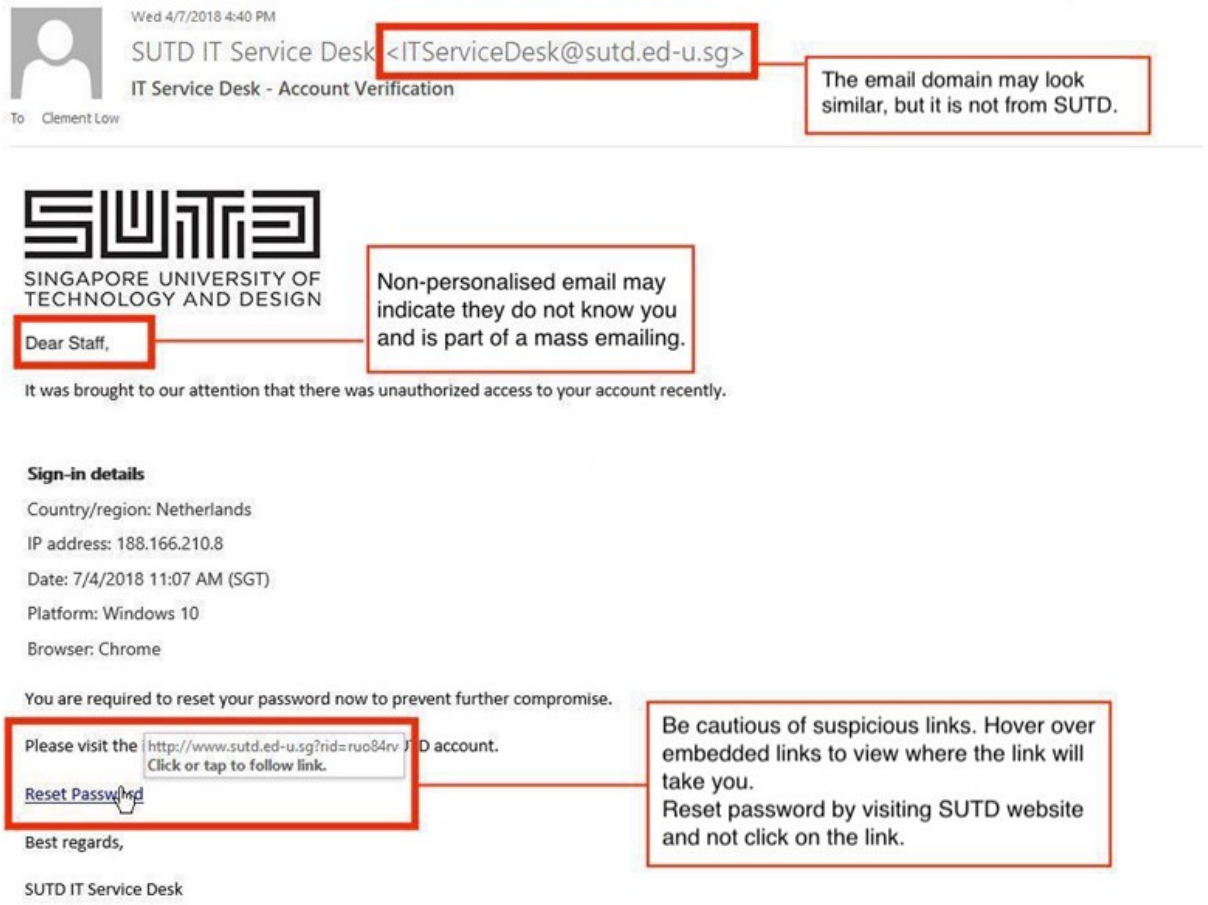
What is classification?

"***The act or process of dividing things into groups according to their type***"

Cambridge Dictionary

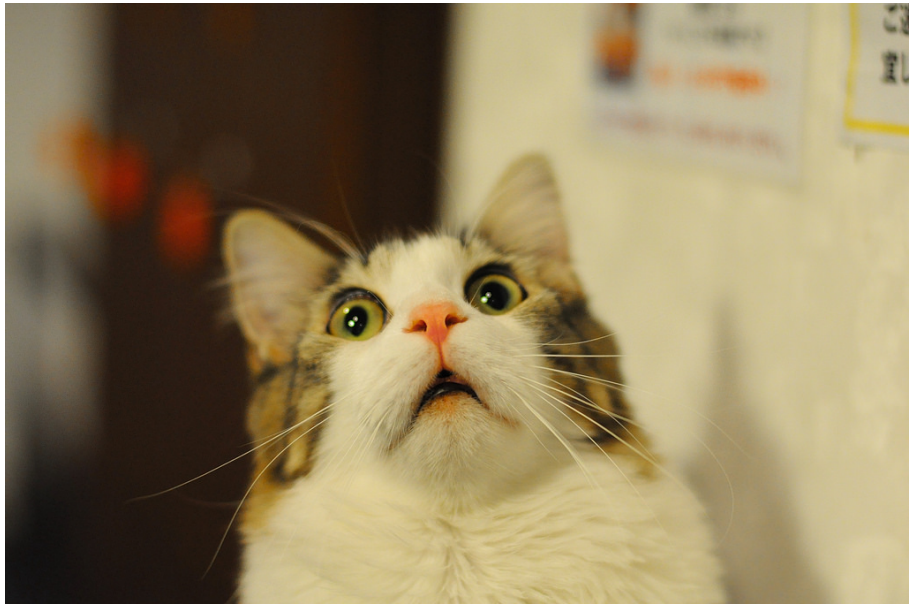
Examples of Classification

- Email, spam or scam?



Examples of Classification

- Cat or dog?



https://www.flickr.com/photos/dat-pics/4553277701

https://www.flickr.com/photos/mydoglikes/27249865650

Examples of Classification

- Traffic jam or not?

Examples of Classification

- Is a movie review positive or negative?
- What species of an animal, plant, etc?
- Where was a message/tweet/post sent from?
- What genre a song belongs to?
- And many more…

Types of dataset / sources

- Text
- Audio
- Images
- Videos
- …

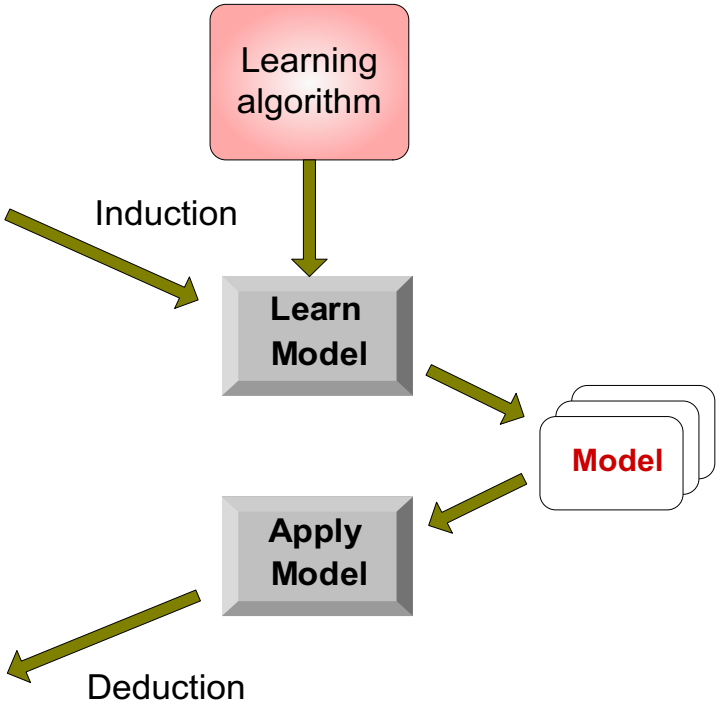
General Approach to Classification Tasks

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Source: Introduction to Data Mining, 2nd Edition

Classification: Definition

- Given a collection of records (training set)
  - Each record is by characterized by a tuple ($\boldsymbol{x}$,y), where $\boldsymbol{x}$ is the attribute set and y is the class label
    - $\boldsymbol{x}$: attribute, predictor, independent variable, input
    - y: class, response, dependent variable, output

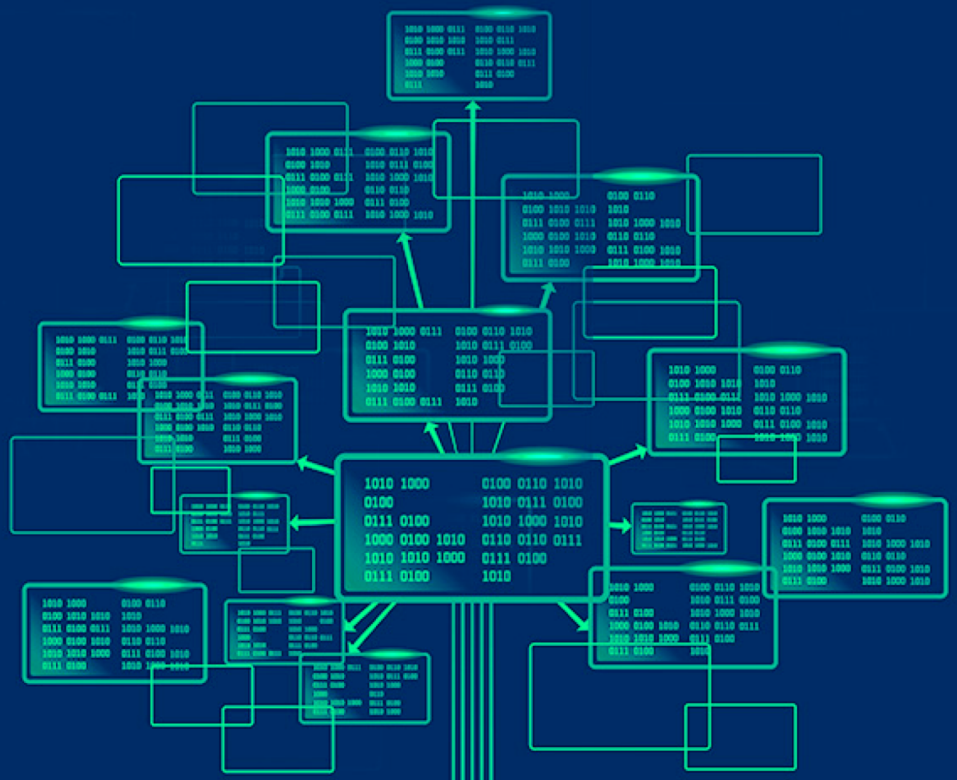
- Task:
  - Learn a model that maps each attribute set $\boldsymbol{x}$ into one of the predefined class labels y

Classification Techniques

- Base Classifiers
  - Decision Tree based Methods
  - Rule-based Methods
  - Nearest-neighbor
  - Logistic regression
  - Naïve Bayes and Bayesian Belief Networks
  - Support Vector Machines
  - Neural Networks
  - Deep Learning
- Ensemble Classifiers
  - Boosting, Bagging, Random Forests

Which classification method should we use?

- The one that gives the **best predictions**…
  - On the training data
  - On the (unseen) test data
  - On the (held-out) validation data
- The one that is **computationally** efficient…
  - during training
  - during classification
- The most **interpretable** one…
  - In terms of its parameters
  - as a model
- The one that is **easiest to implement**…
  - For learning
  - For classification



Decision trees

Decision Tree

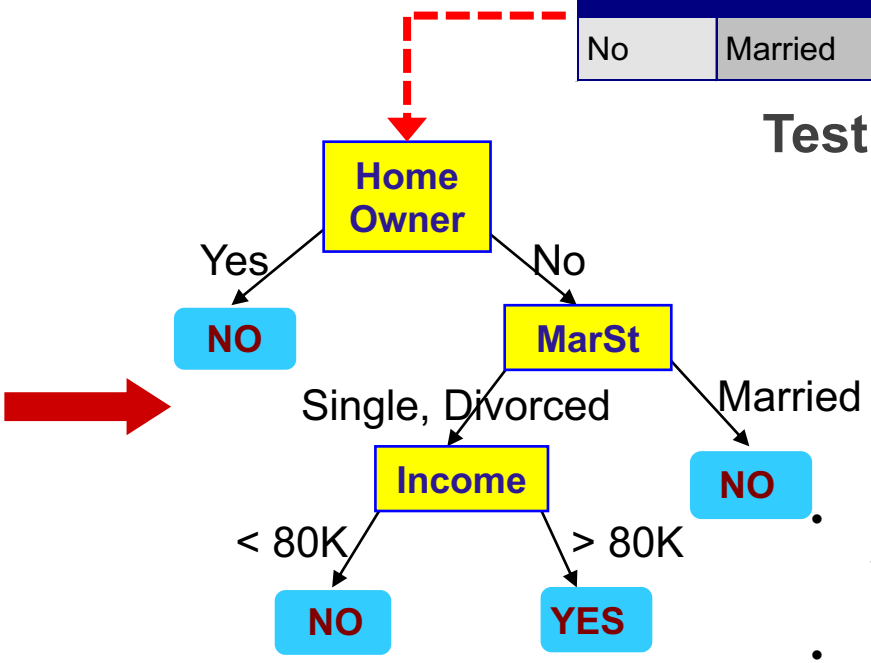
- Decision point at each feature, until reaching the leaf node (category)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

Training Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Test Data

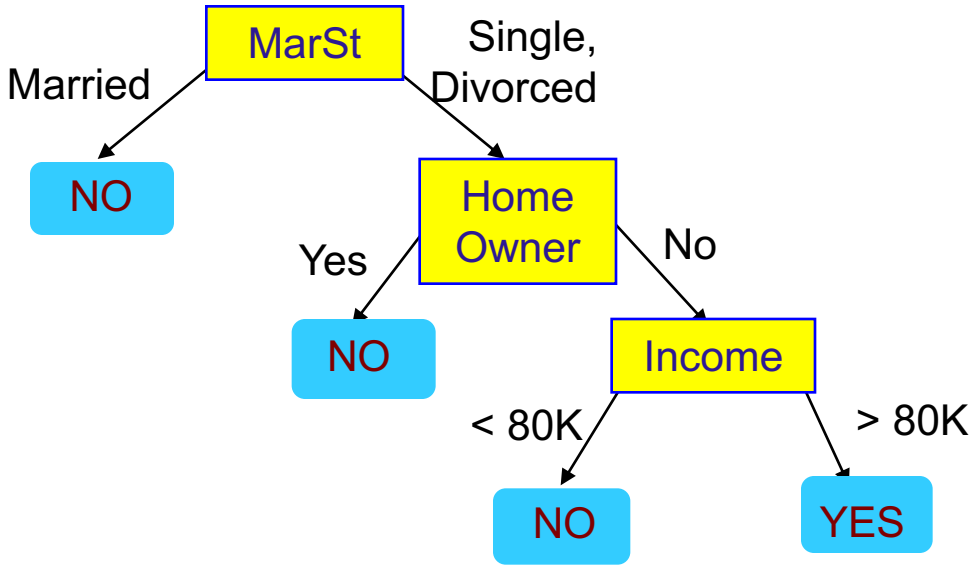


Model: Decision Tree

- **Internal node**: test of the value of a given attribute
- **Branch**: value of an attribute
- **Leaf**: predicted class

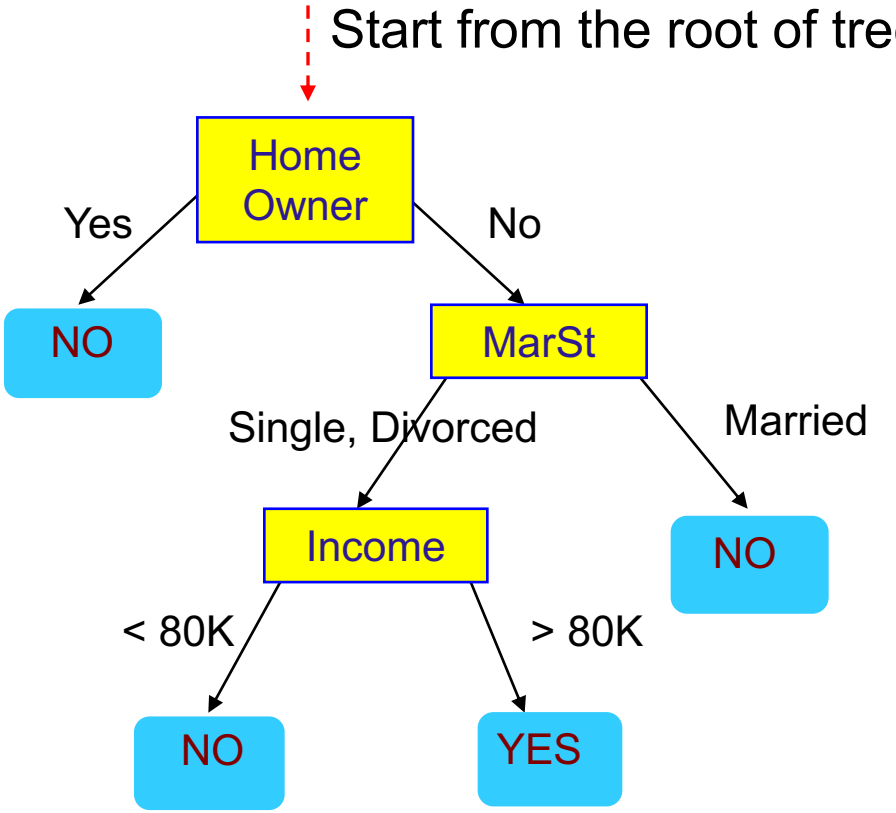
Another Example of Decision Tree

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
	categorical	categorical	continuous	class
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!

Apply Model to Test Data



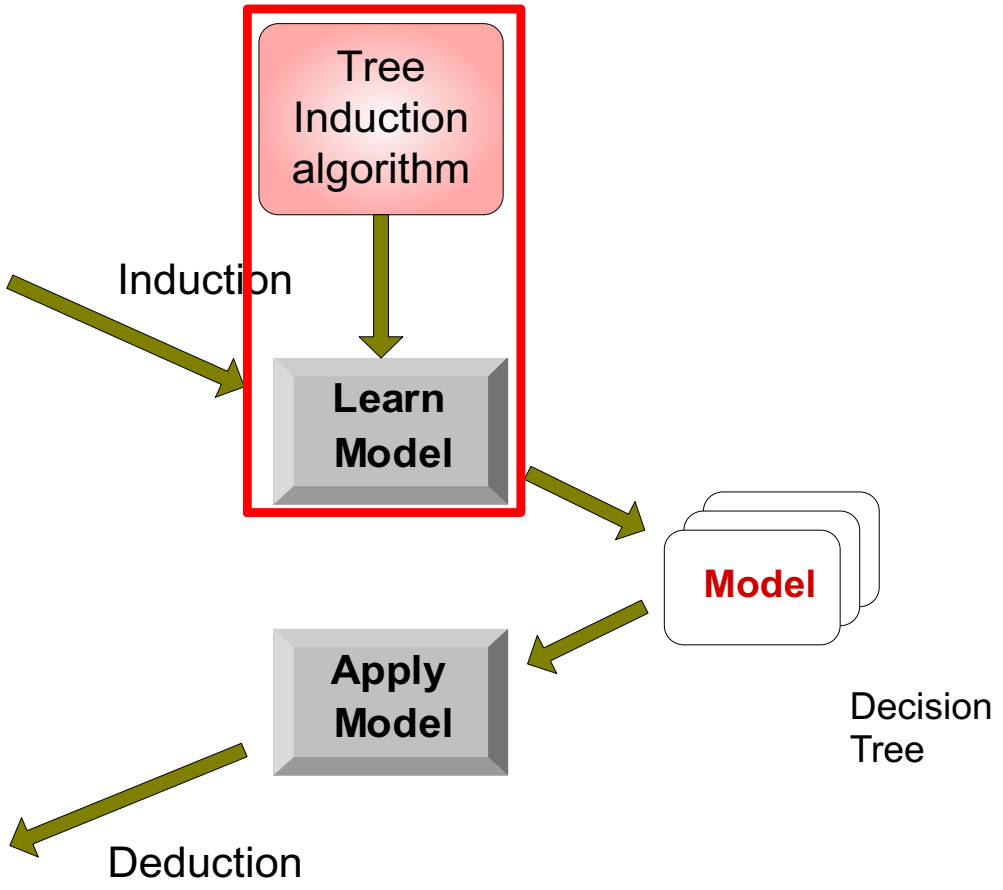
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Decision Tree Induction

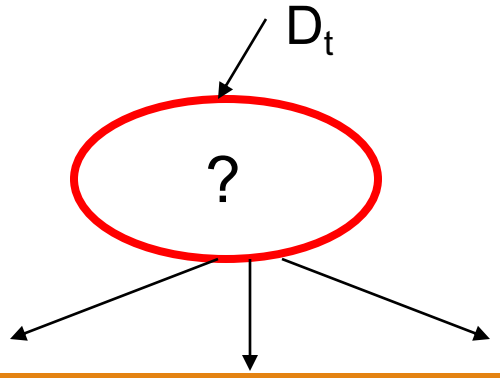
- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART (Classification And Regression Trees)
  - ID3 (Iterative Dichotomiser 3)
  - C4.5 (successor of ID3)
  - SLIQ (Supervised Learning In Quest,)
  - SPRINT (Scalable PaRallelizable INndution of decision Trees)

General Structure of Hunt's Algorithm

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

- General Procedure:
  - If D_t contains records that only belong the class y_t, then *t* is a **leaf** node labeled as y_t
  - If D_t contains records that belong to more than one class, use **an attribute test** to split the data into smaller subsets.
  - Recursively apply the procedure to each subset.

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

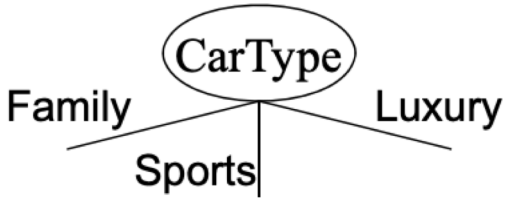


Design Issues of Decision Tree Induction

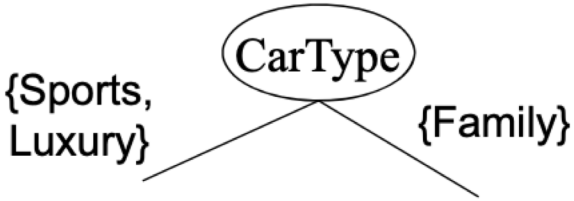
- **How** should training records be split?
  - *Which attribute* to use in a split node split?
    - How to determine the best split
  - How to specify the attribute *test condition*?
    - E.g. x < 1? Or x+y < 1 ?
  - Shall we use 2-way or *multi-way split*?
- How should the splitting procedure **stop**?
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

Splitting of nominal attributes

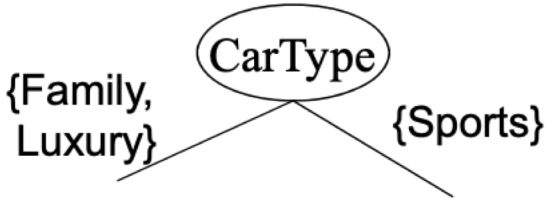
- Multi-way split: use as many partitions as distinct values



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



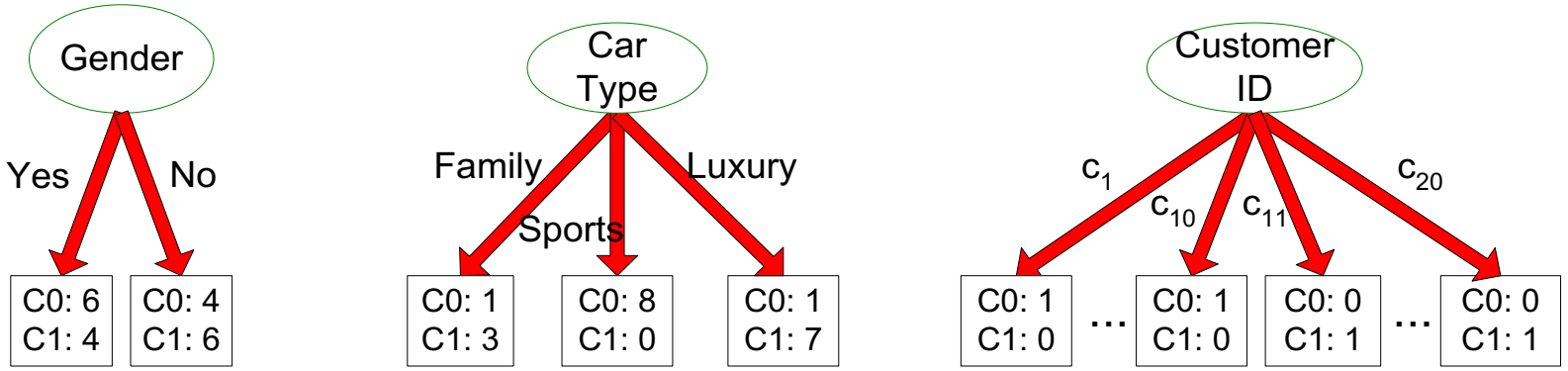
OR



How to determine the Best Split

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

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 purer, homogeneous class distribution are preferred

- Need a measure M of node impurity:

C0: 5 C1: 5

Non-homogeneous,
High degree of impurity

C0: 9 C1: 1

Homogeneous,
Low degree of impurity

Measures M of Node Impurity

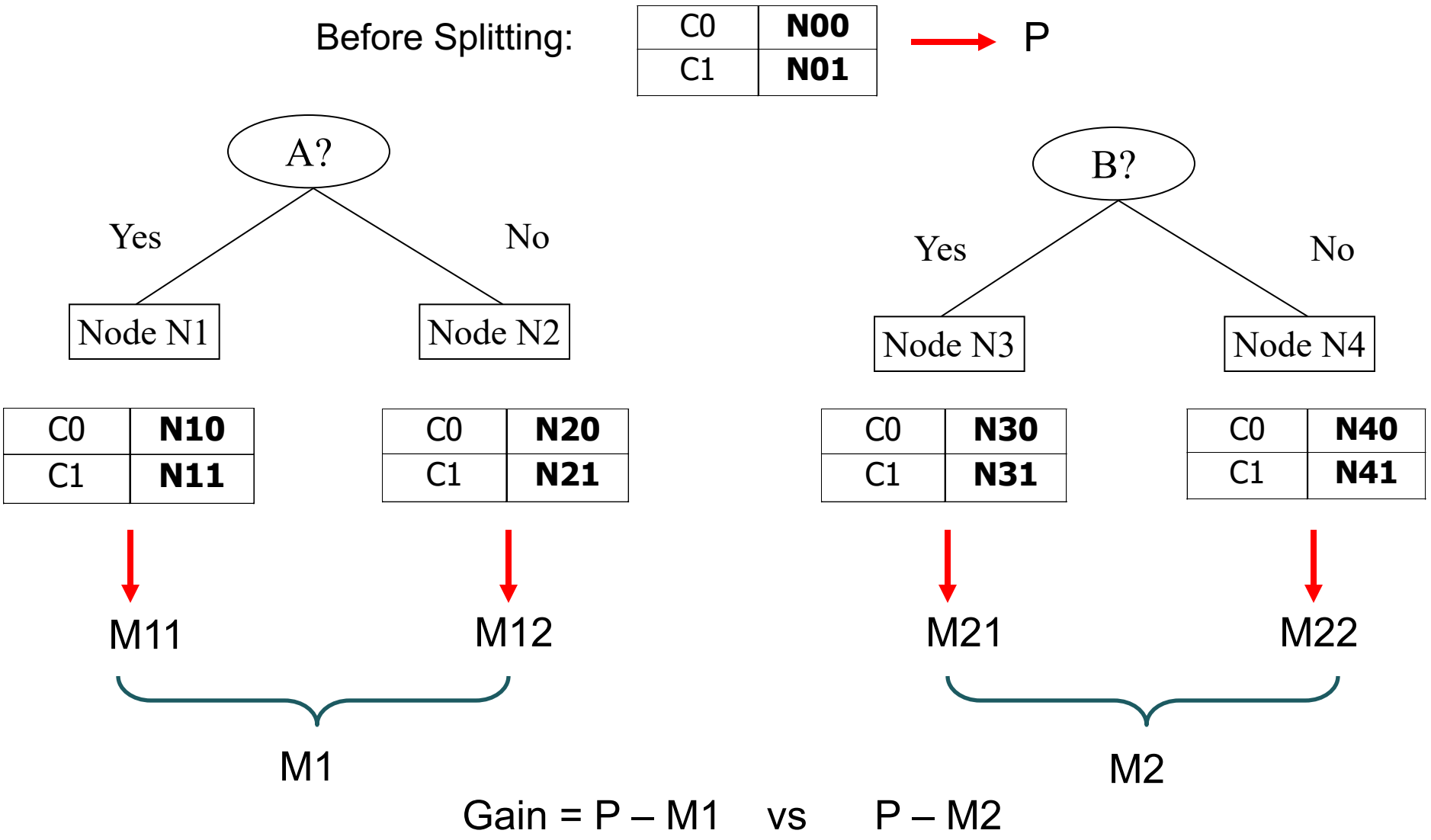
- Entropy
- Gini Index
- Misclassification error

Finding the Best Split

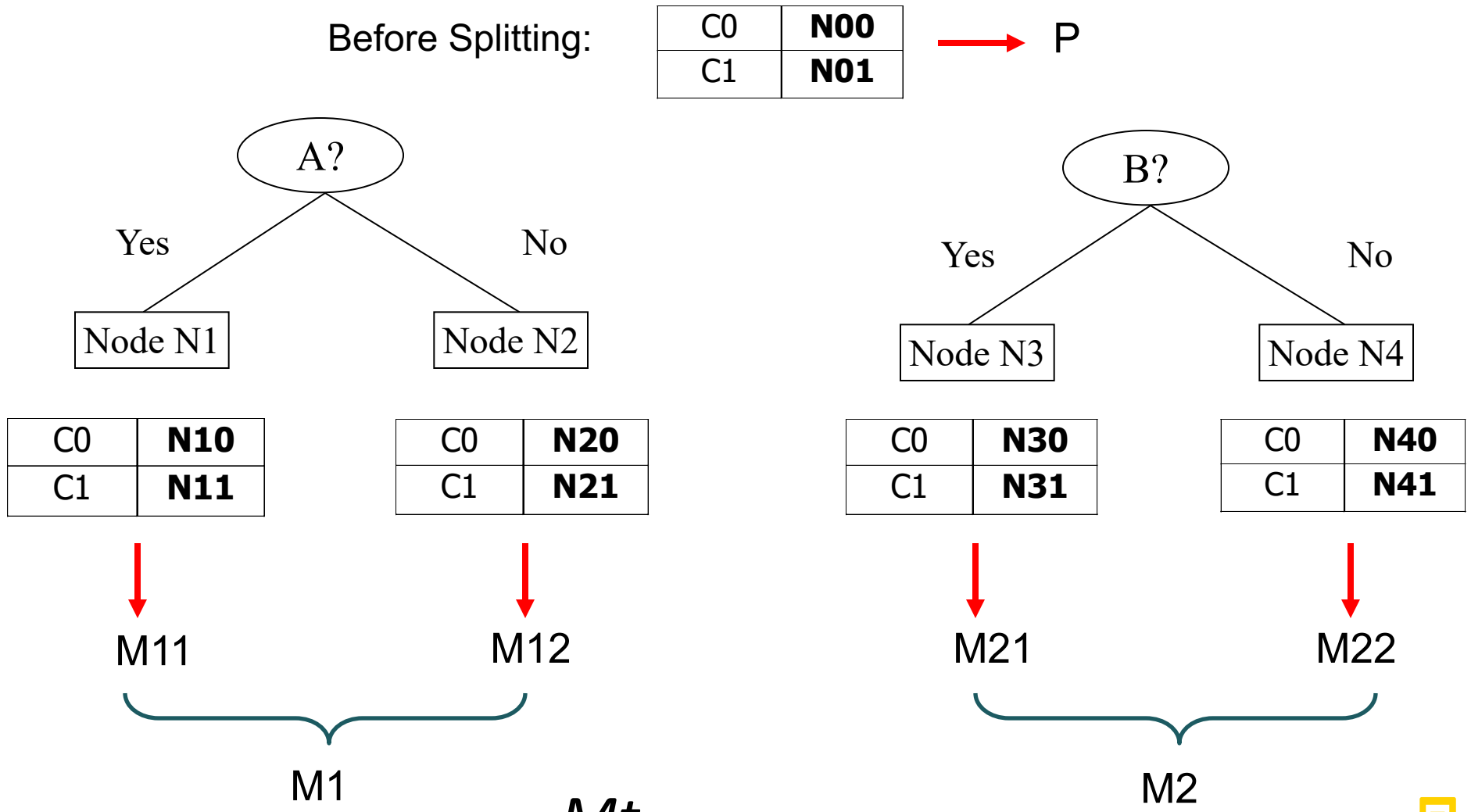
1. Compute impurity measure (P) **before** splitting
2. Compute impurity measure (M) **after** splitting
   - Compute impurity measure of each child node
   - M is the weighted impurity of children
3. Choose the attribute test condition that produces the **highest gain** or equivalently, lowest impurity measure after splitting (M)

$$\text{Gain} = P - M$$

Finding the Best Split



Finding the Best Split



$$Mt = Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

Measure of Impurity: Entropy

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

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

  - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
  - Minimum (0.0) when all records belong to one class, implying most information

- Entropy based computations are quite similar to the GINI index computations

Computing Entropy of a Single Node

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

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

Splitting based on Information Gain

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

- $GAIN_{split}$ measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
  - Used in ID3 and C4.5
- Disadvantage: bias toward attributes with large number of values
  - Large trees with many branches are preferred
  - What happens if there is an ID attribute?

Splitting based on Gain Ratio

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \qquad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

Parent Node, p is split into k partitions

n_i is the number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO).
  - **Higher entropy partitioning (large number of small partitions) is penalized**!
- Used in C4.5 algorithm
- Designed to overcome the disadvantage of Information Gain

Split information

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

A = 1	A= 2	A = 3	A = 4	*SplitINFO*
32	0	0	0	0
16	16	0	0	1
16	8	8	0	1.5
16	8	4	4	1.75
8	8	8	8	2

Other measures of impurity

- Gini Index for a given node *t* :

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

(NOTE: p(j | t) is the relative frequency of class *j* at node *t*)

- Classification error at a node *t (with classes i)*:

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

Computing Gini Index of a Single Node

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

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

Computing Error of a Single Node

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

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**

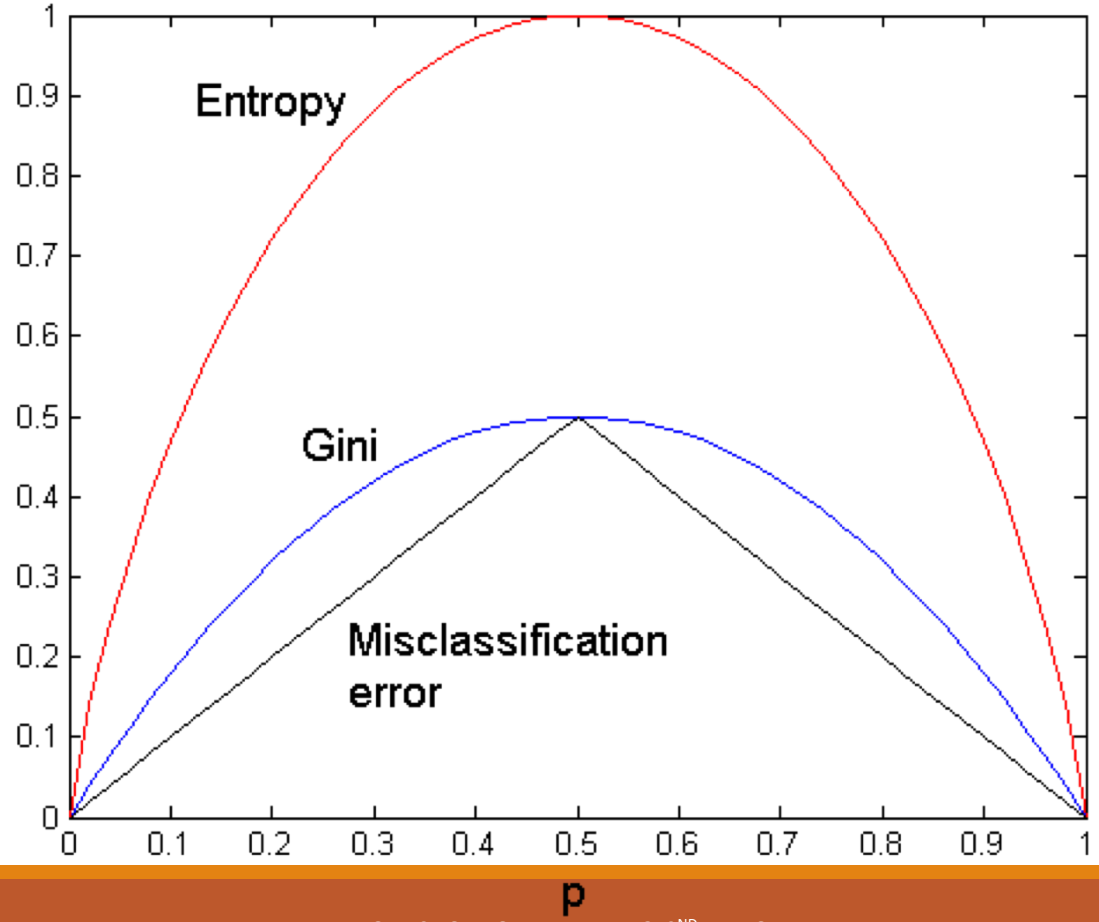
P(C1) = 2/6 P(C2) = 4/6

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

If there are children nodes, they will be weighted depending on how many instances they contain

Comparison among Impurity Measures

For a 2-class problem:



Practical issues

- Conclusion: decision trees are built by greedy search algorithms, guided by some heuristic that measures “impurity”
- In real-world applications we need also to consider
  - Continuous attributes
  - Missing values – handled during preprocessing
  - Improving computational efficiency
  - Overfitted trees

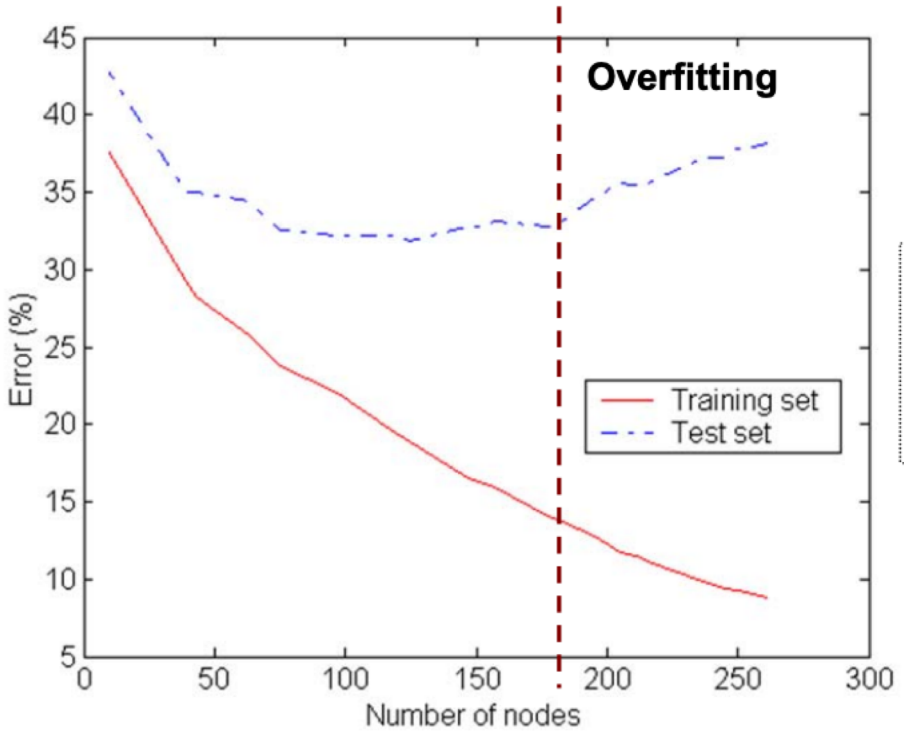
Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute

    Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering:
    - Static – discretize once at the beginning
    - Dynamic – repeat at each node [computationally intensive!]

  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more computationally intensive

Overfitting and underfitting



When the tree becomes too large, its test error rate begins increasing while its training error rate continues too decrease.

What causes overfitting?

- Noise

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

How to address underfitting

Pre-Pruning (Early Stopping Rule)

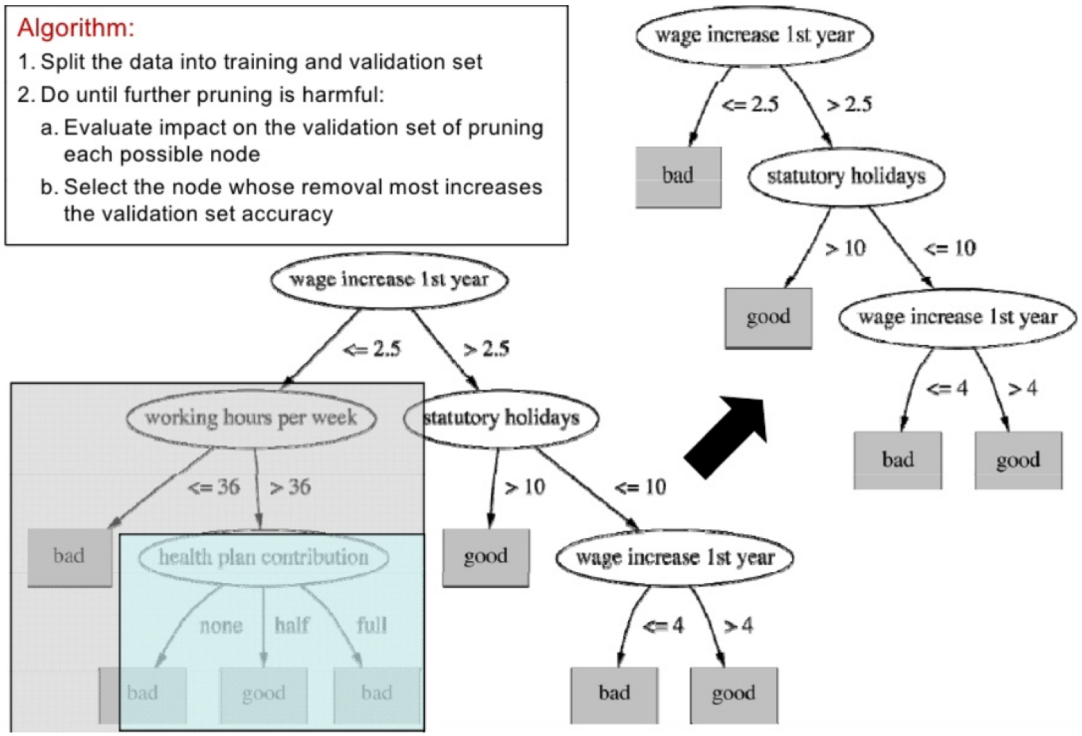
- Stop the algorithm before it becomes a fully-grown tree
  - Stop if **all instances belong to the same class**
  - Stop if **all the attribute values are the same**
- Early stopping conditions:
  - Stop if number of instances is less than some **user-specified threshold**
    - e.g. 5 to 10 records per node
  - Stop if class distribution of instances are independent of the available attributes (e.g., using Chi Sq test)
  - Stop if splitting the current node improves the **impurity measure** (e.g. Gini or information gain) below a given **threshold**

How to address overfitting

Post-pruning

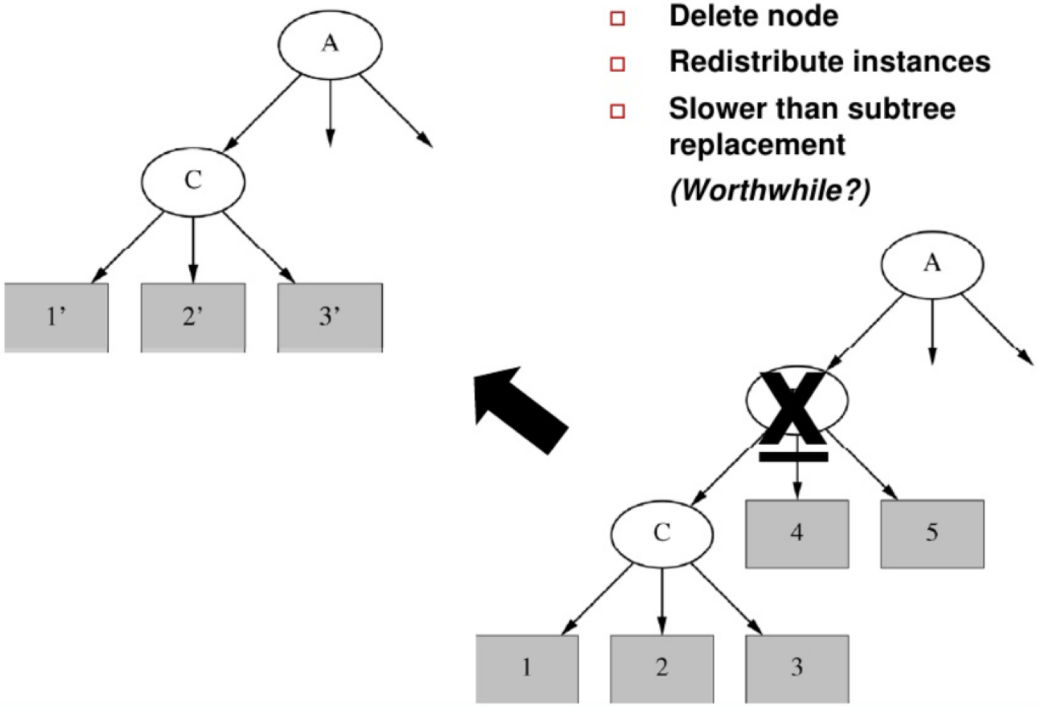
- Grow decision tree to its entirety
  - Trim the nodes of the decision tree in a bottom-up fashion
  - **Reduced-error pruning** *[remove subtrees if error does not increase]*
- Use a dataset not used in the training pruning set
  - Three data sets are needed: training set, test set, **pruning set**
- If test error in the pruning set improves after trimming then prune the tree
  - Post-pruning can be achieved in two ways:
    - **Sub-tree replacement:** selects a subtree and replaces it with a single leaf.
    - **Sub-tree raising:** selects a subtree and replaces it with the child one (ie, a "sub-subtree" is deleted)

Subtree replacement



https://www.slideshare.net/aorriols/lecture6-c45

Subtree raising



- Delete node
- Redistribute instances
- Slower than subtree replacement

(Worthwhile?)

https://www.slideshare.net/aorriols/lecture6-c45

ID3, C4.5, C5.0, CART

- Ross Quinlan
  - ID3 uses the Hunt's algorithm with information gain criterion and gain ratio
  - C4.5 (1994) improves ID3
    - Needs entire data to fit in memory
    - Handles missing attributes and continuous attributes
    - Performs tree post-pruning
  - C5.0 is the current commercial successor of C4.5
- Breiman et al.
  - CART builds multivariate decision (binary) trees, supports regression.

Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Robust to noise (especially when methods to avoid overfitting are employed)
  - Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)
- Disadvantages:
  - Space of possible decision trees is exponentially large. Greedy approaches are often unable to find the best tree.
  - Does not take into account interactions between attributes
  - Each decision boundary involves only a single attribute

Rule based methods

Example of a rule set

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Application of rules - covering

A rule r **covers** an instance x if the attributes of the instance satisfy the condition (LHS) of the rule

- *R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds*
- *R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes*
- *R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals*
- *R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles*
- *R5: (Live in Water = sometimes) → Amphibians*

The rule R1 **covers** a hawk ⇒ Class = Bird

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

From rules to decision tree

Consider the rule set

- Attributes A, B, C, and D can have values 1, 2, and 3

$$A = 1 \wedge B = 1 \rightarrow Class = Y$$
$$C = 1 \wedge D = 1 \rightarrow Class = Y$$
$$Otherwise, Class = N$$

How to represent it as a decision tree?

- The rules need a common attribute

$$A = 1 \wedge B = 1 \rightarrow Class = Y$$
$$A = 1 \wedge B = 2 \wedge C = 1 \wedge D = 1 \rightarrow Class = Y$$
$$A = 1 \wedge B = 3 \wedge C = 1 \wedge D = 1 \rightarrow Class = Y$$
$$A = 2 \wedge C = 1 \wedge D = 1 \rightarrow Class = Y$$
$$A = 3 \wedge C = 1 \wedge D = 1 \rightarrow Class = Y$$
$$Otherwise, Class = N$$

More verbose!

Building classification rules

- Direct Method
  - Extract rules directly from data
  - e.g.: RIPPER, Holte's 1R (OneR)
- Indirect Method
  - Extract rules from other classification models (e.g. decision trees, etc).
  - e.g: C4.5rules

A direct method: sequential covering

- Let E be the training set
  - Extract rules one class at a time

For each class C

1. Initialize set S with E
2. While S contains instances in class C
   3. Learn one rule R for class C
   4. Remove training records covered by the rule R

Goal: to create rules that cover many examples of a class C and none (or very few) of other classes

Direct method: RIPPER

- For 2-class problem, choose one of the classes as positive class, and the other as negative class
  - Learn rules for positive class
  - **Negative class will be default class**
- For multi-class problem
  - **Order** the classes according to **increasing class prevalence** (fraction of instances that belong to a particular class)
  - Learn the rule set for smallest class first, treat the rest as negative class
  - Repeat with next smallest class as positive class

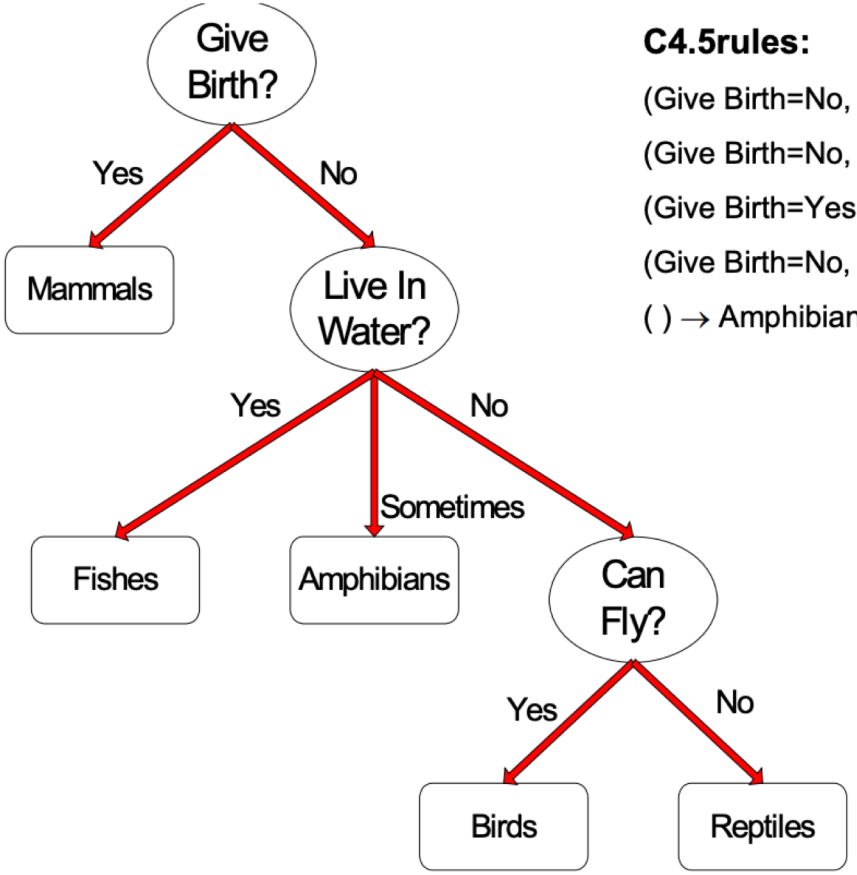
Direct method: RIPPER

- Start from empty rule
- Add conjuncts as long as they improve information gain (FOIL)
  - Stop when rule no longer covers negative examples
- Prune the rule immediately using **reduced error pruning**
  - Measure for pruning: W(R) = (p-n)/(p+n)
    - p: number of positive examples covered by the rule in the validation set
    - n: number of negative examples covered by the rule in the validation set
- Pruning starts from the last test added to the rule
  - May create rules that cover some negative examples (accuracy < 1)
  - A global optimization (pruning) strategy is also applied

Direct method: RIPPER

- Use sequential covering algorithm
  - Find the best rule that covers the current set of positive examples
  - Eliminate both positive and negative samples covered by the rule from the dataset
- Each time a rule is added to the rule set, compute the new description length
  - Stop adding new rules when the new description length is *d* bits longer than the smallest description length obtained so far.

C4.5 rules versus RIPPER



C4.5rules:

(Give Birth=No, Can Fly=Yes) → Birds

(Give Birth=No, Live in Water=Yes) → Fishes

(Give Birth=Yes) → Mammals

(Give Birth=No, Can Fly=No, Live in Water=No) → Reptiles

( ) → Amphibians

RIPPER:

(Live in Water=Yes) → Fishes

(Have Legs=No) → Reptiles

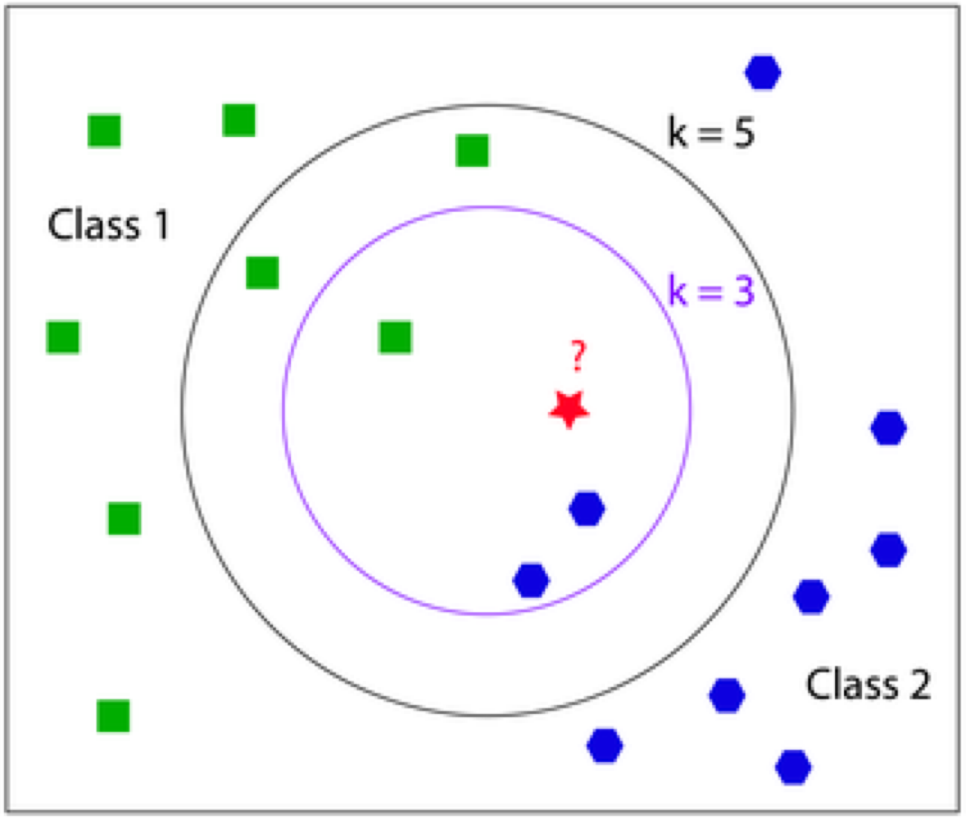
(Give Birth=No, Can Fly=No, Live In Water=No) → Reptiles

(Can Fly=Yes,Give Birth=No) → Birds

() → Mammals

Advantages

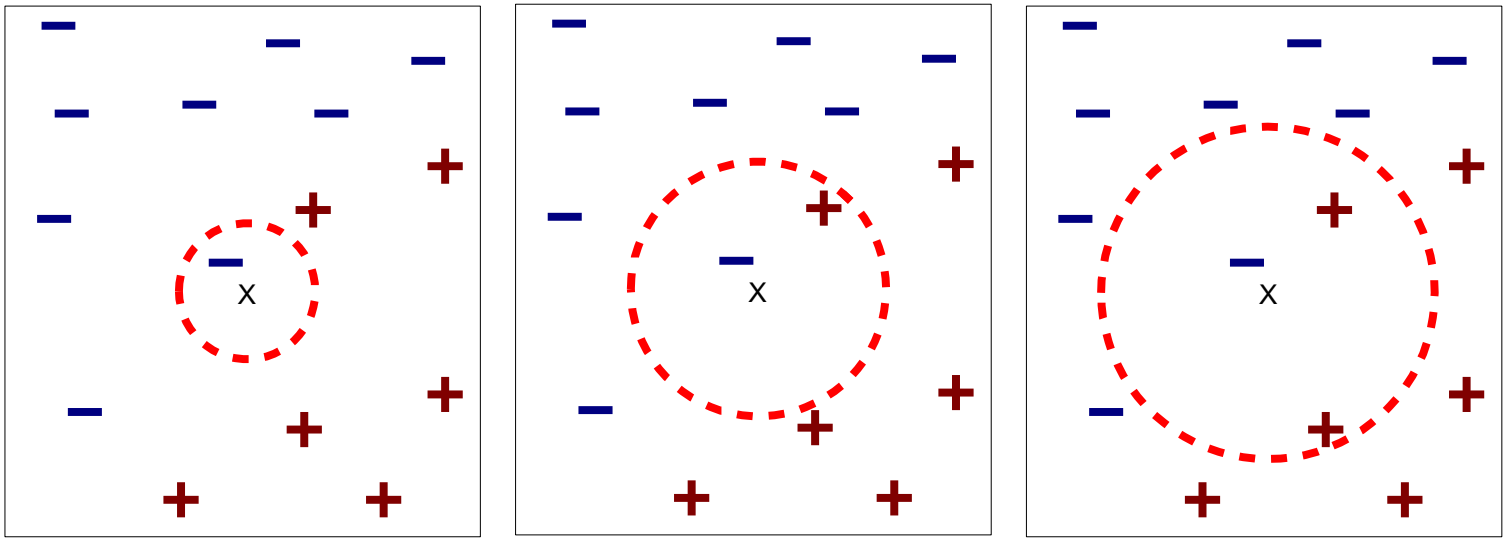
- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees
- Can easily handle missing values and numeric attributes



K-nearest neighbour

k-Nearest Neighbour

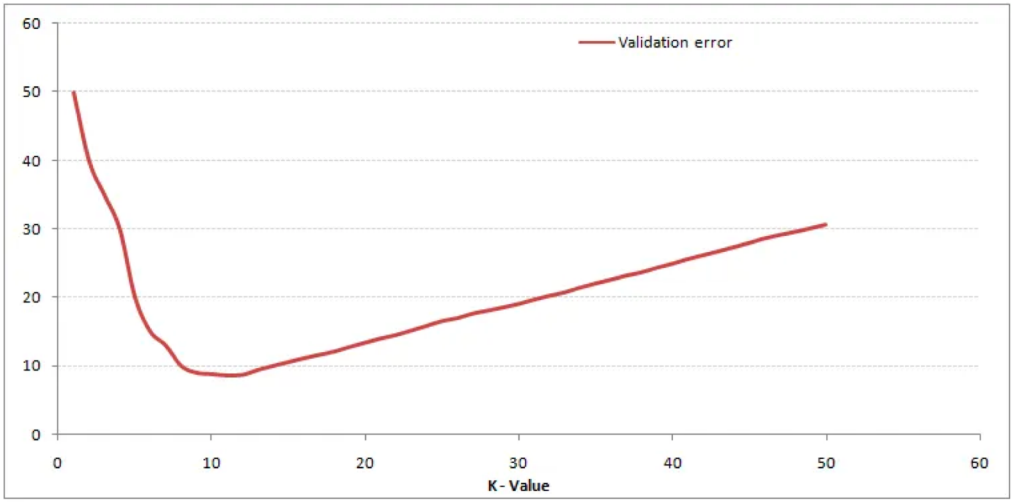
- Assign observation to nearest neighbour, or k-nearest neighbour (with majority voting)
- Euclidean distance (takes a while to compute for all pairs)



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

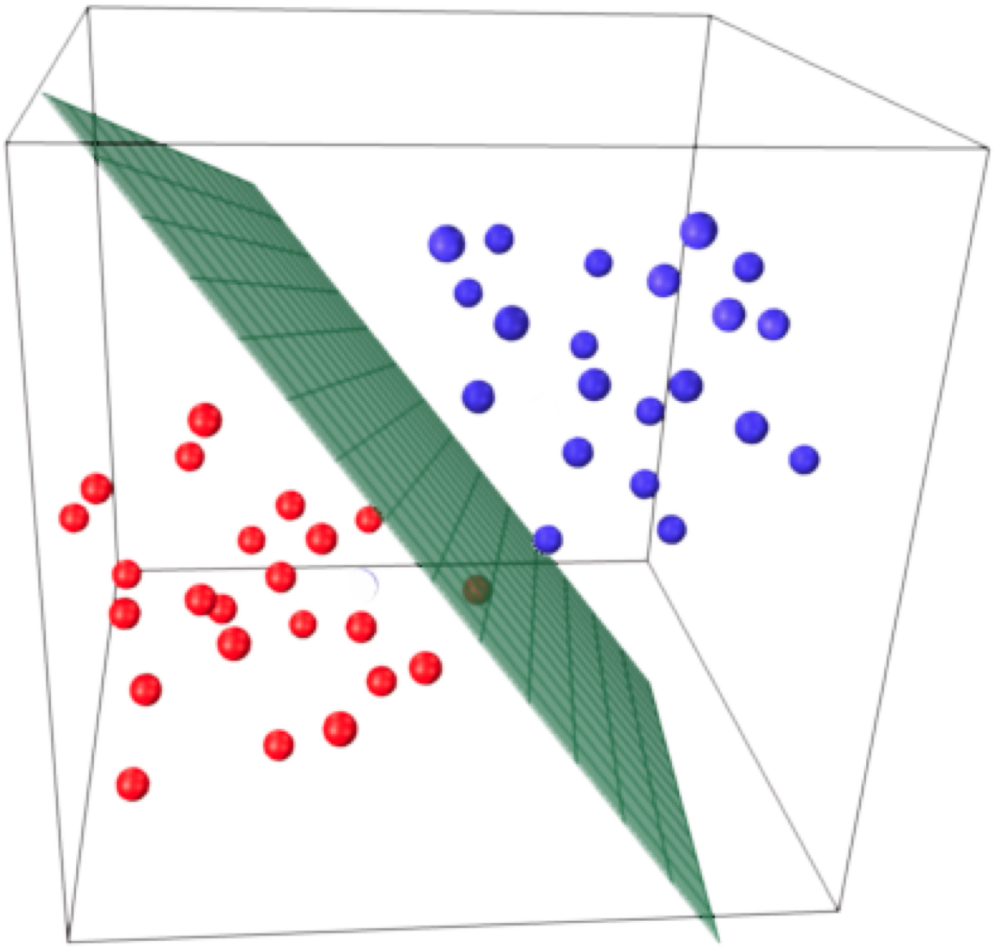
How to choose k

- Make this plot for your data:



Characteristics of k-NN

- Straightforward algorithm
  - Simply assign to nearest neighbour or majority voting
- Sensitive to value of k
  - Too small, overly sensitive to noise/outliers
  - Too large, neighbours include points from other classes
- Computationally expensive
  - Need to compute distances to all points (but can speed up)
- Space requirement
  - Need to store all data points

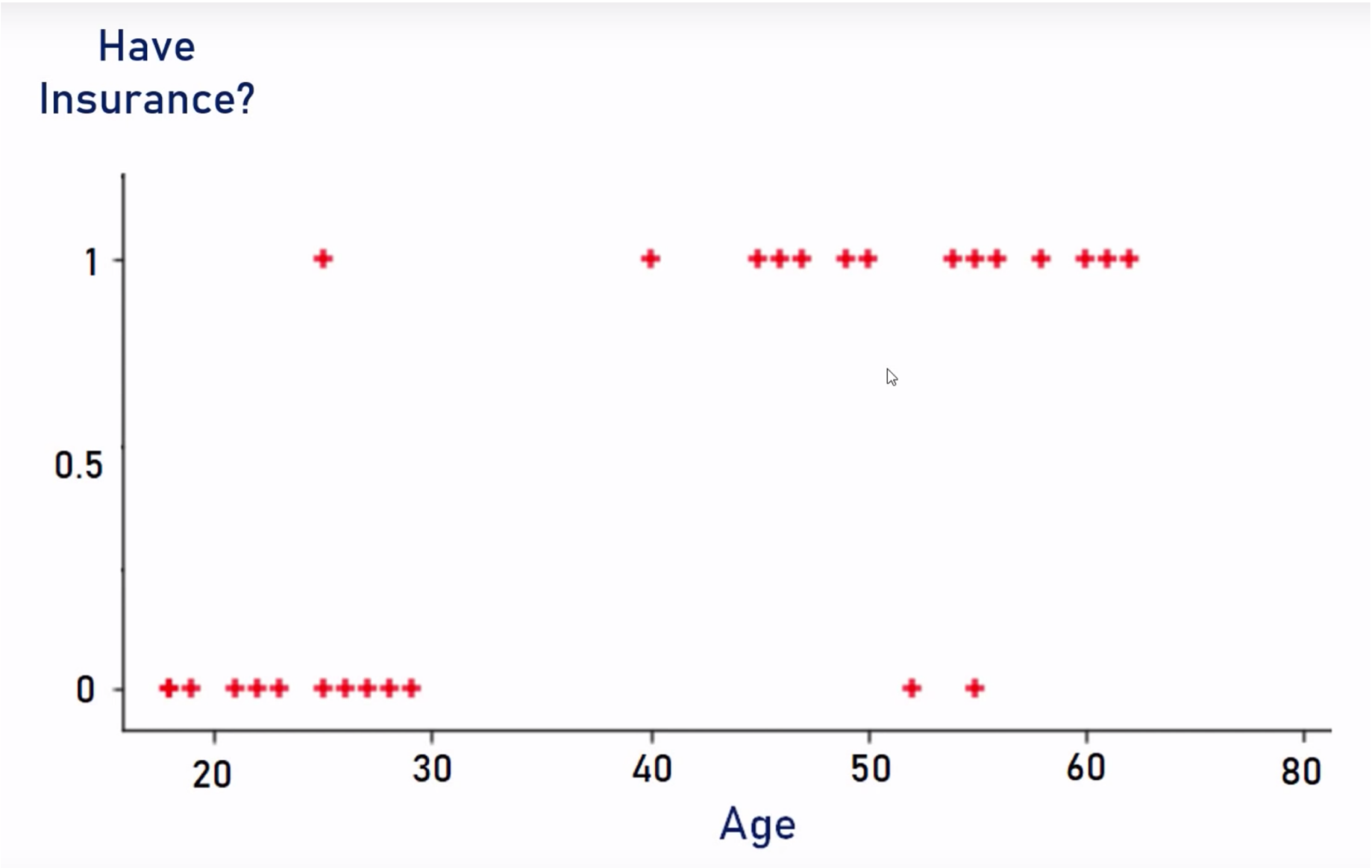


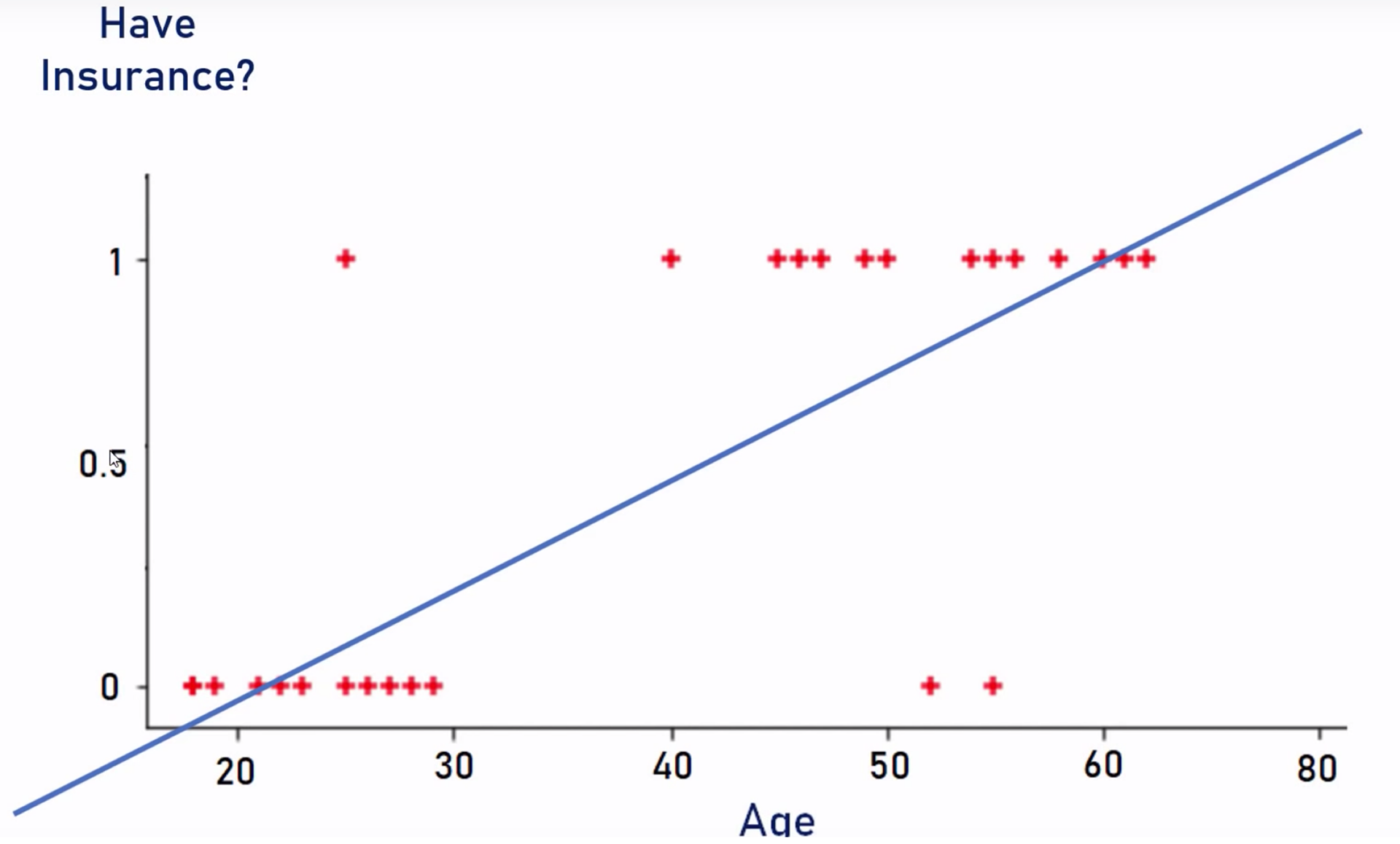
Logistic regression

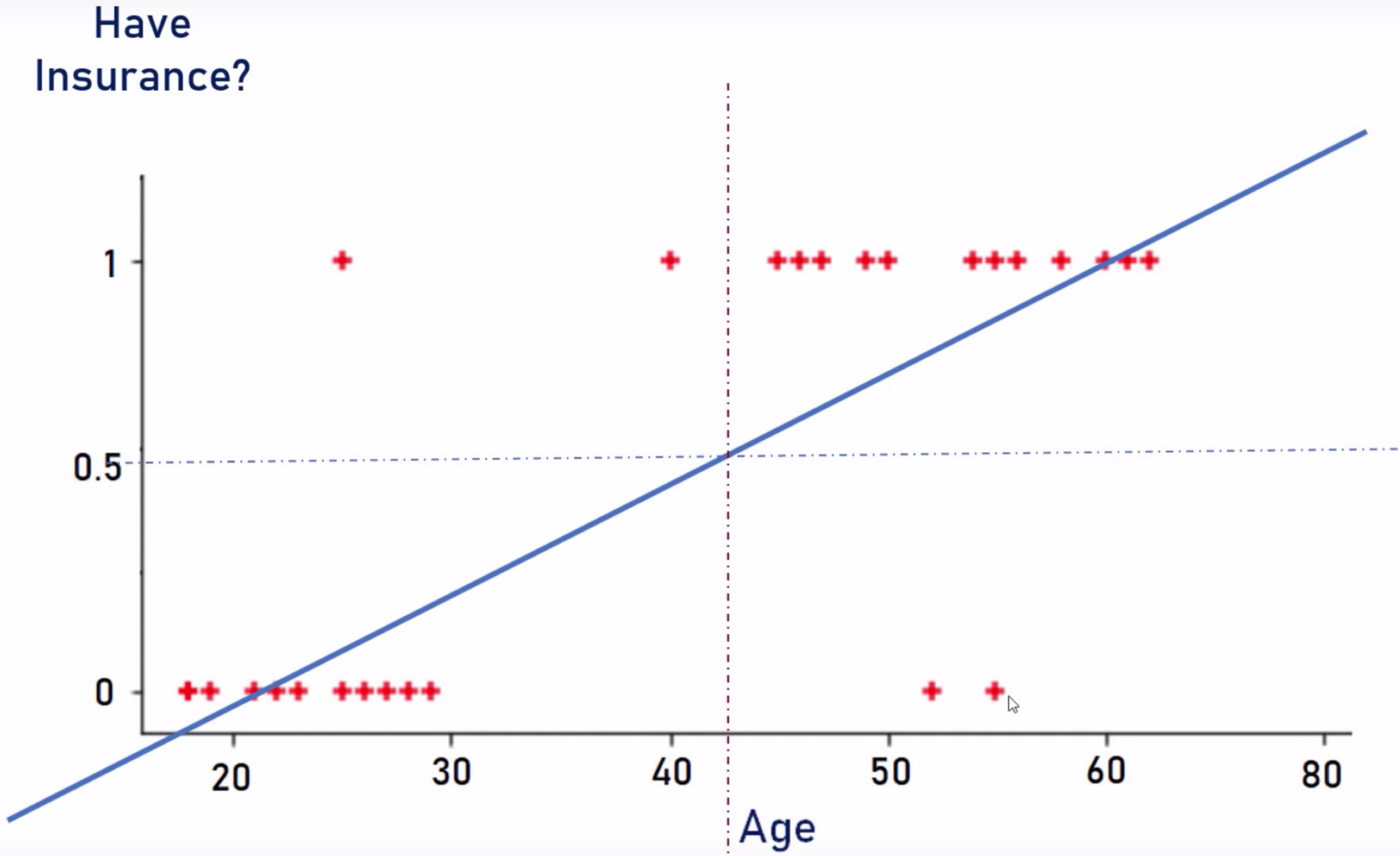
Binary classification

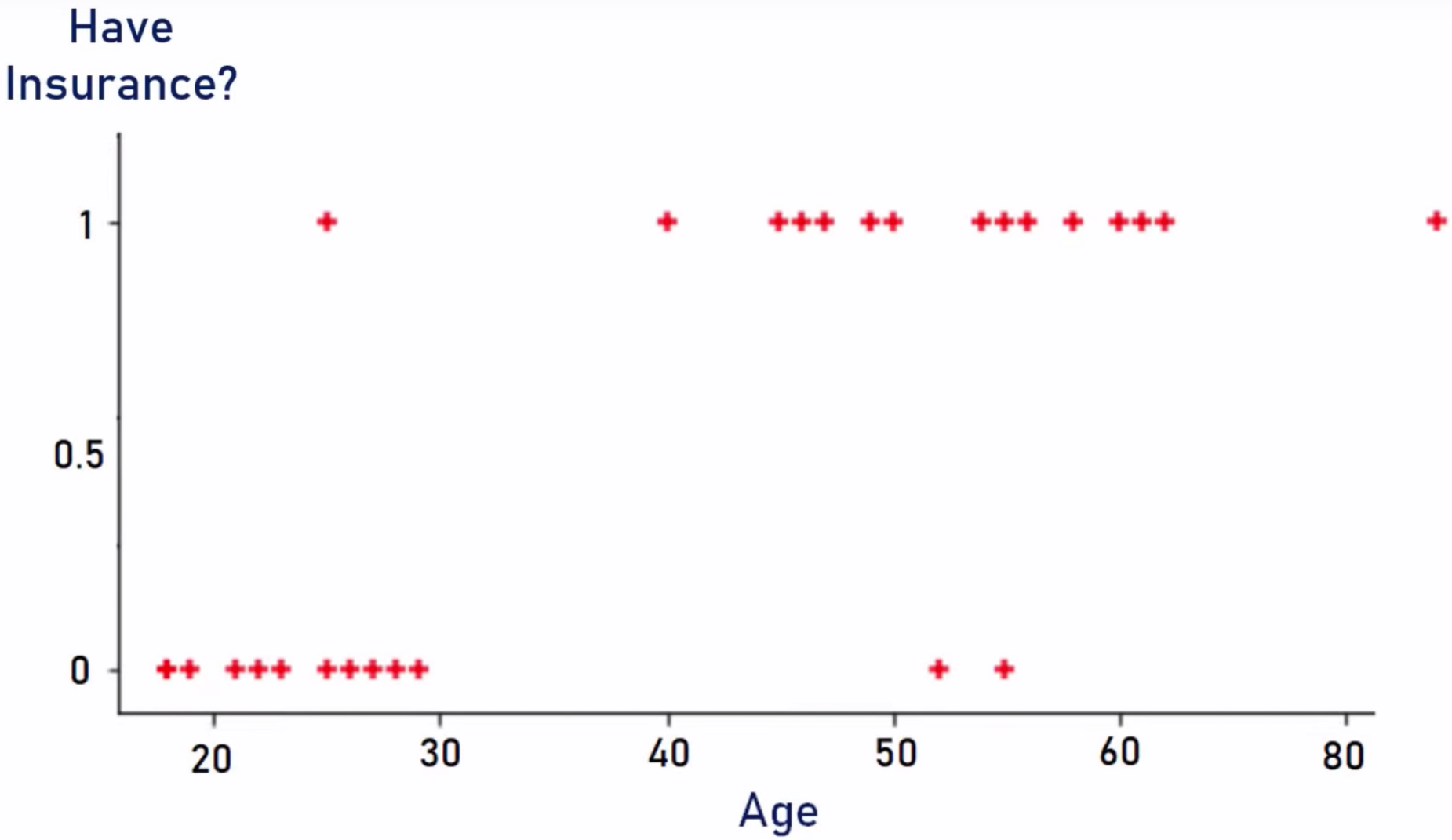
age	have_insurance
22	0
25	0
47	1
52	0
46	1
56	1
55	0
60	1
62	1
61	1
18	0
28	0
27	0
29	0
49	1

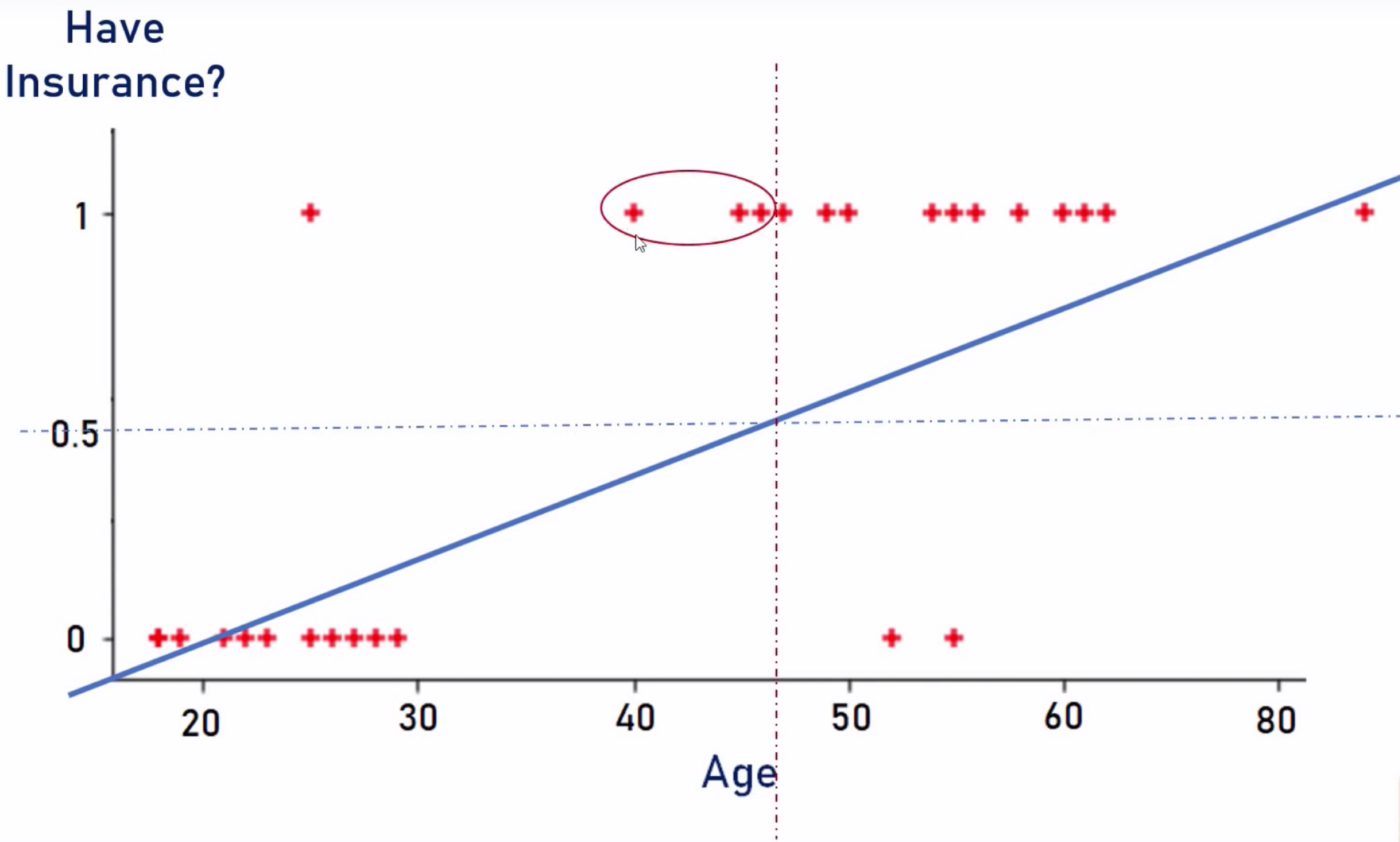
https://www.youtube.com/watch?v=zM4VZR0px8E

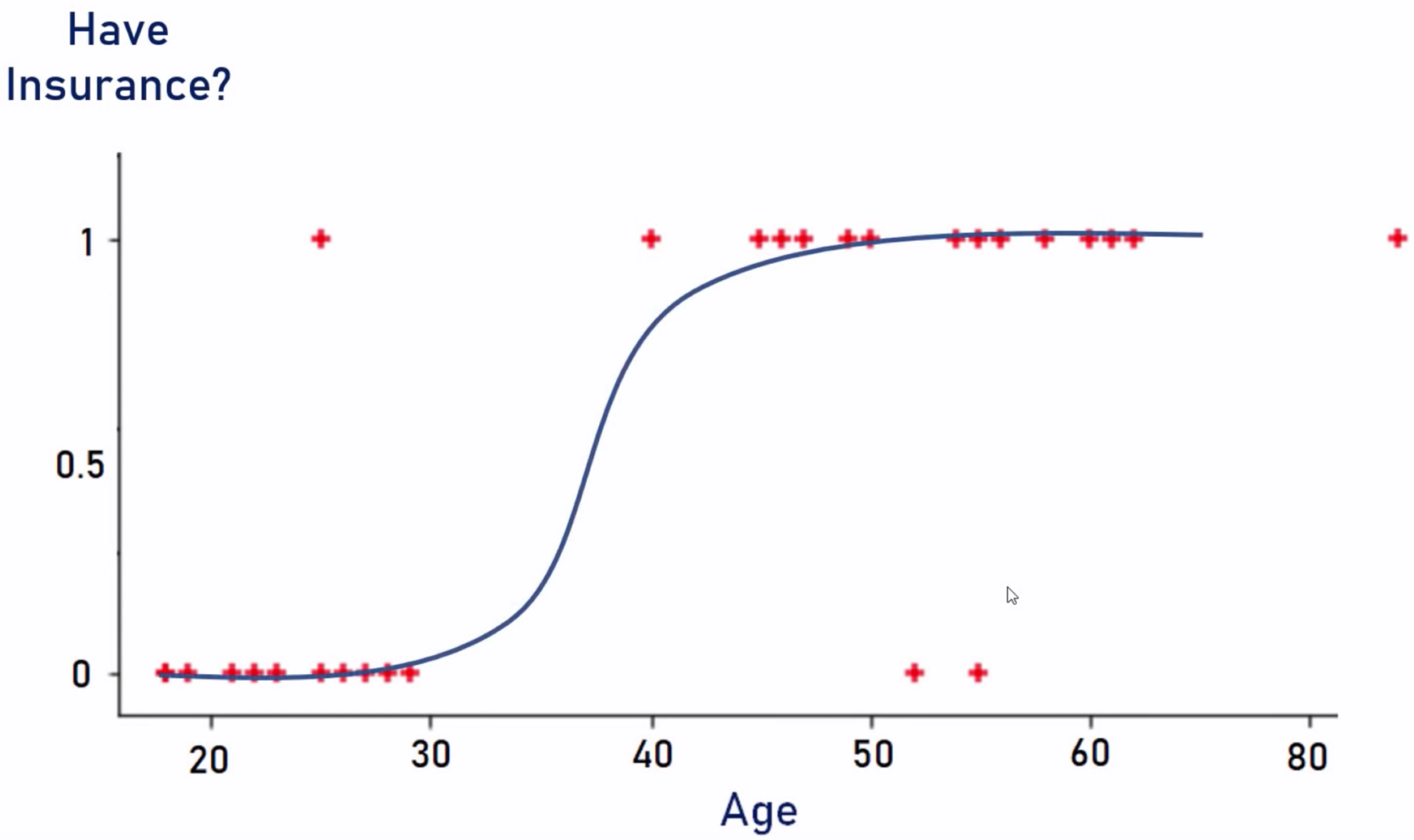

Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age


Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age

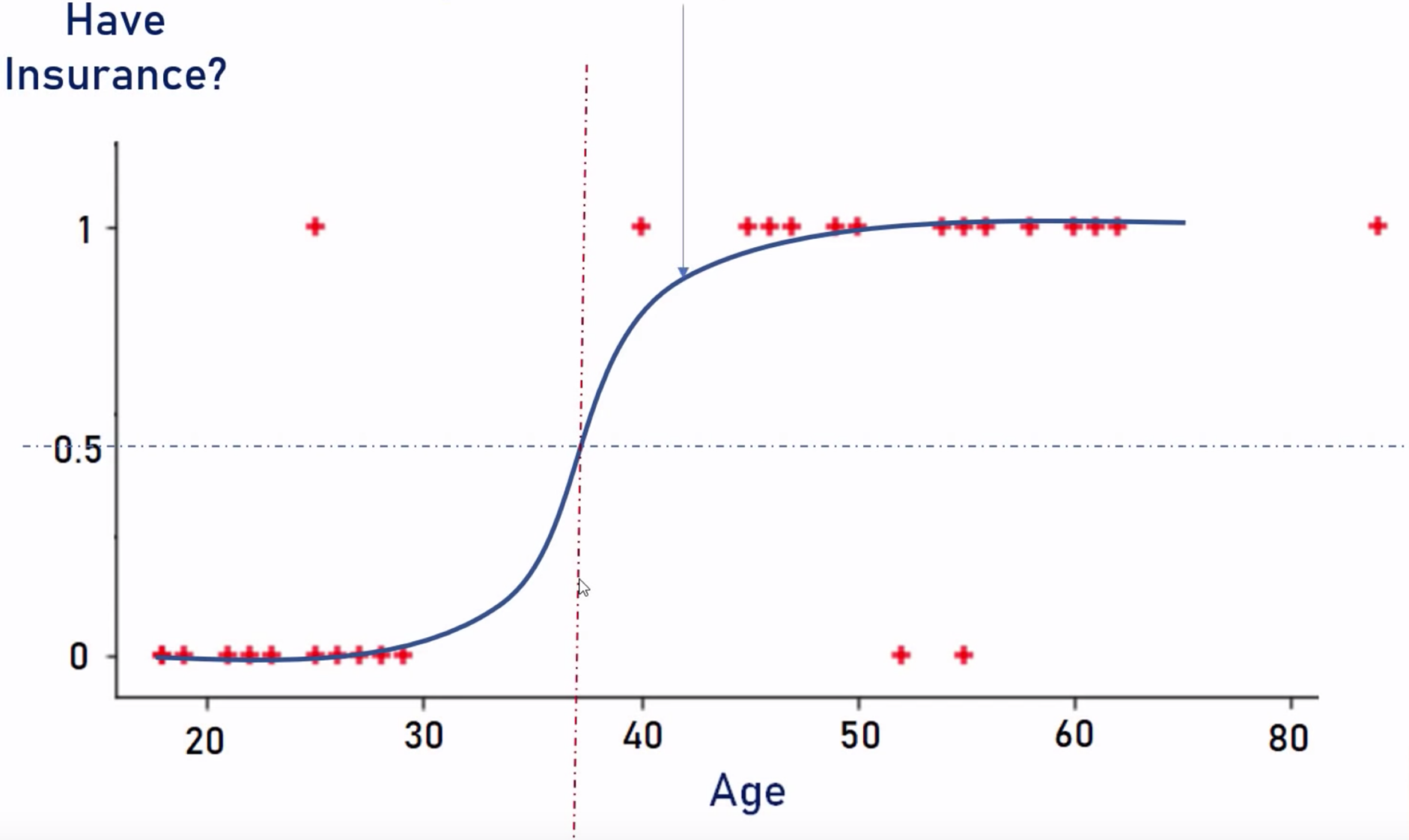

Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age


Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age


Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age

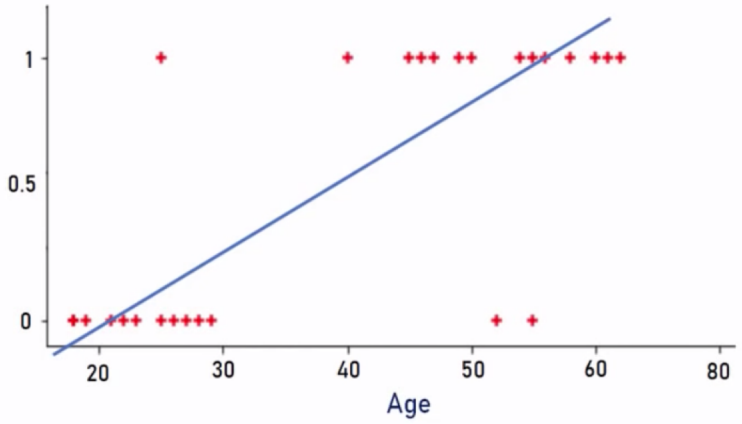

Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age

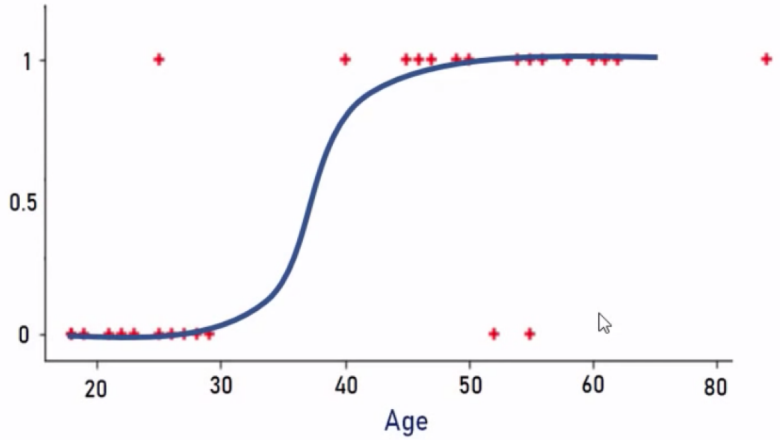
Sigmoid or Logit Function



$$y = m * x + b$$

$$y = \frac{1}{1 + e^{-(m*x+b)}}$$





Threshold value

- Always 0.5?

- No. Choose a good value based on confusion matrix

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Logit

Logistic regression (multiple variables):

- P(class is 1) = $\frac{1}{1+e^{-(m1*x1+m2*x2+\ldots+b)}}$
- P(class is 0) = **1 –** $\frac{1}{1+e^{-(m1*x1+m2*x2+\ldots+b)}}$

Logit:

Class is 1 when:

Odds = $\frac{\text{P (class is 1)}}{\text{P (class is 0)}} = e^{(m1*x1+m2*x2+\ldots+b)} > 1$

Logit(P): log (Odds) = $m1 * x1 + m2 * x2 + \ldots + b$ ~ linear regression

Deriving logit

$$\frac{P(\text{class } 1)}{P(\text{class } 0)} = \frac{\frac{1}{1+e^{-z}}}{1 - \frac{1}{1+e^{-z}}} = \frac{\frac{1}{1+e^{-z}}}{\frac{(1+e^{-z}) - 1}{1+e^{-z}}}$$

$$= \frac{\frac{1}{1+e^{-z}}}{\frac{e^{-z}}{1+e^{-z}}} = \frac{\frac{1}{1+e^{-z}}}{e^{-z}} \cdot (1+e^{-z}) = \frac{1}{e^{-z}} = e^{z}$$

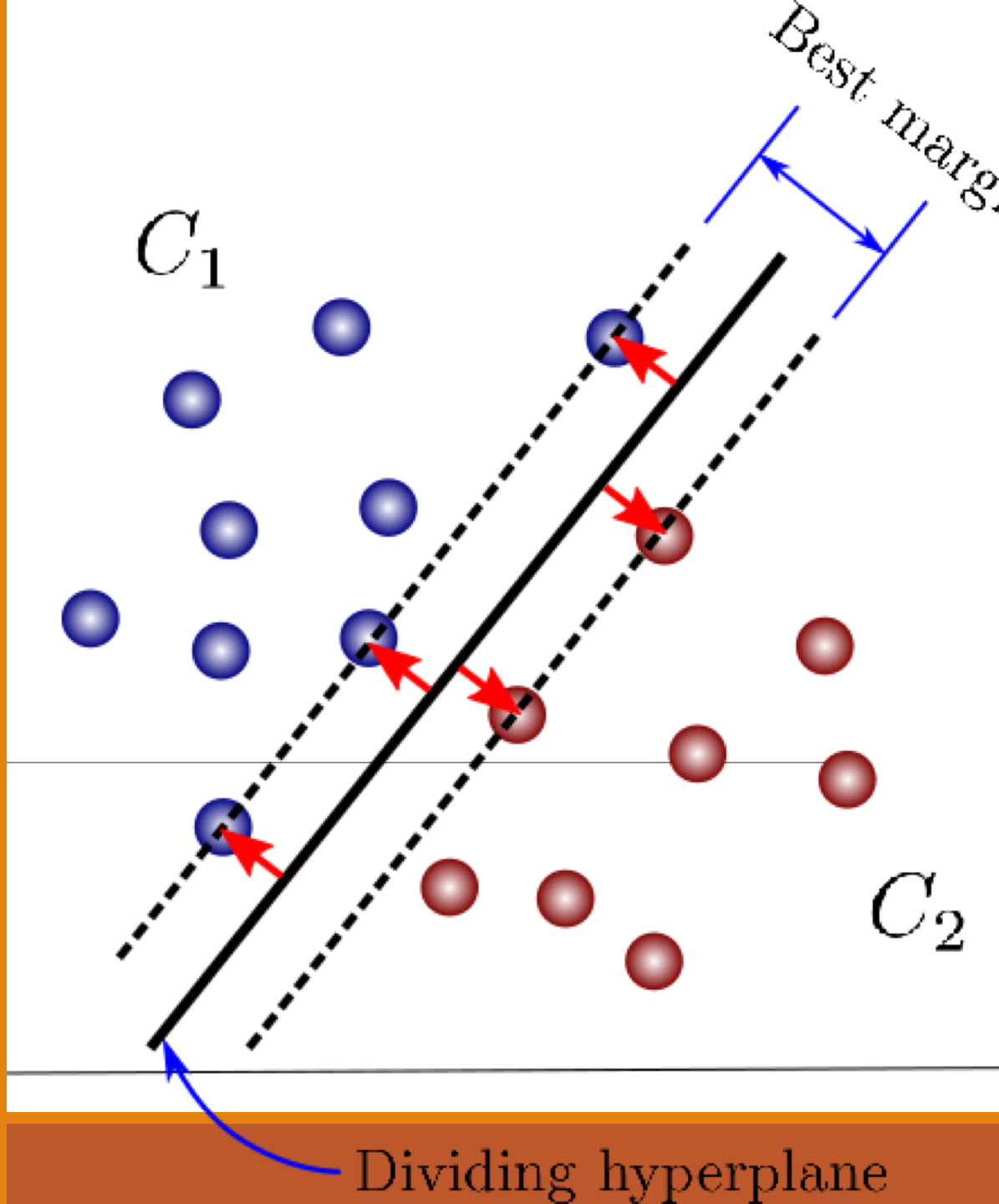
with $z = m_1 x_1 + m_2 x_2 + \dots b$

Multiclass classification

- In this case, a regression model is learned for each class, highest prediction = class.
- Disadvantage: independent predictions, models are built independent of each other.
- Example: cheap, expensive, car, tree
  If cars are expensive, and a car is detected, the model for expensive will not know there is a car detected.

Support vector machines

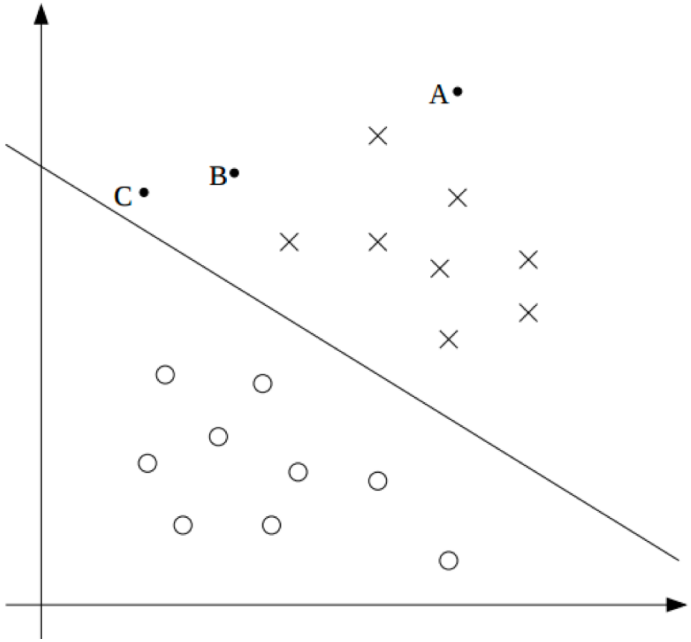
SVM



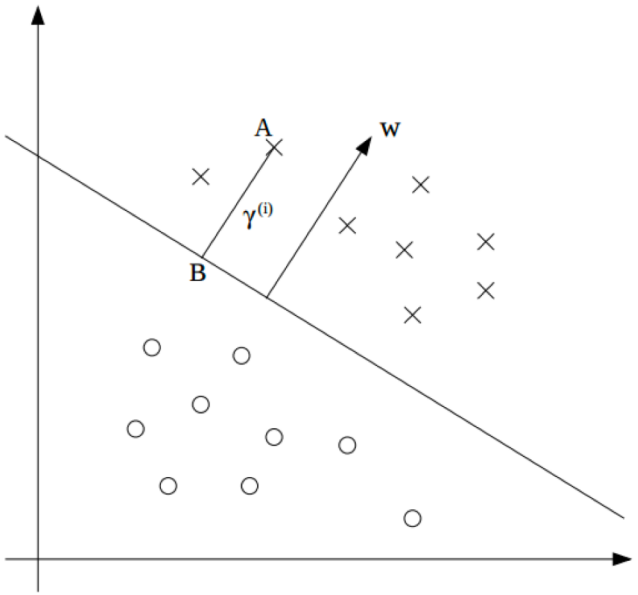
SVM

- A classifier derived from statistical learning theory by Vapnik, et al. in 1992
- Became famous as it performed as well as neural networks on the handwriting recognition task
- Widely used in:
  - Object detection & recognition
  - Content-based image retrieval
  - Text recognition
  - Biometrics
  - Speech recognition
  - ...

Margins & confidence

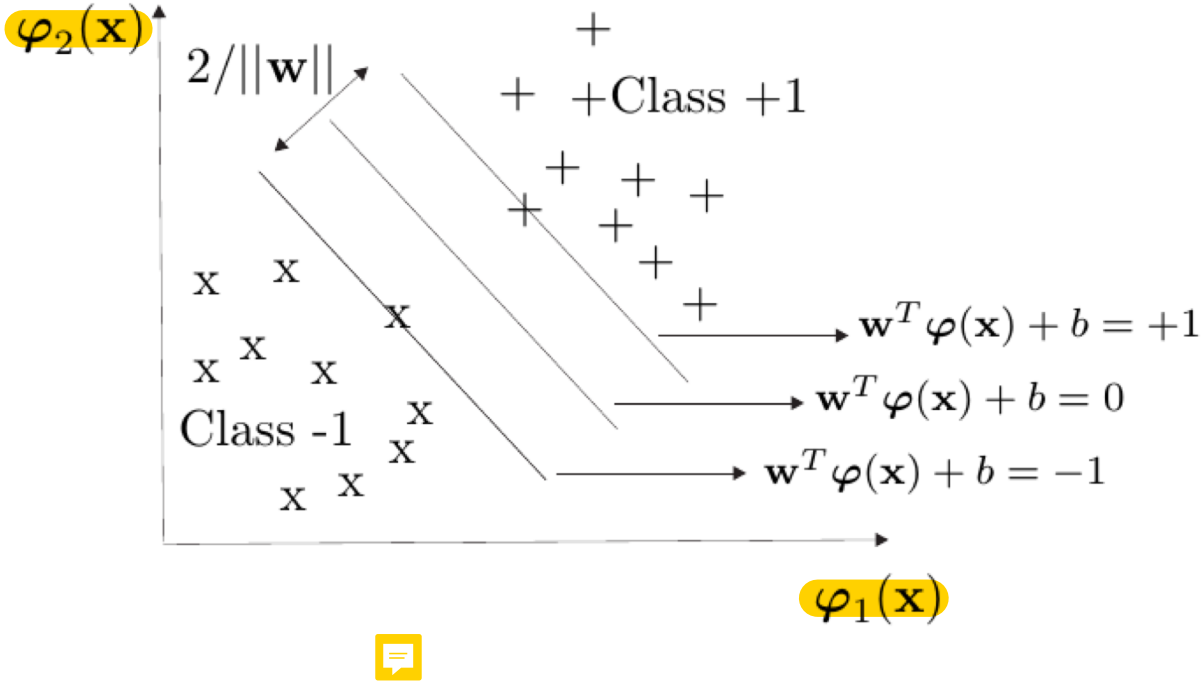


Geometric margins



(Ng, 2014)

Optimal margins



Optimal margins

Given a training set of N data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ with input data $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding binary class labels $y_i \in \{-1, +1\}$, the SVM classifier should fulfil following conditions. (vapnik, 1995):

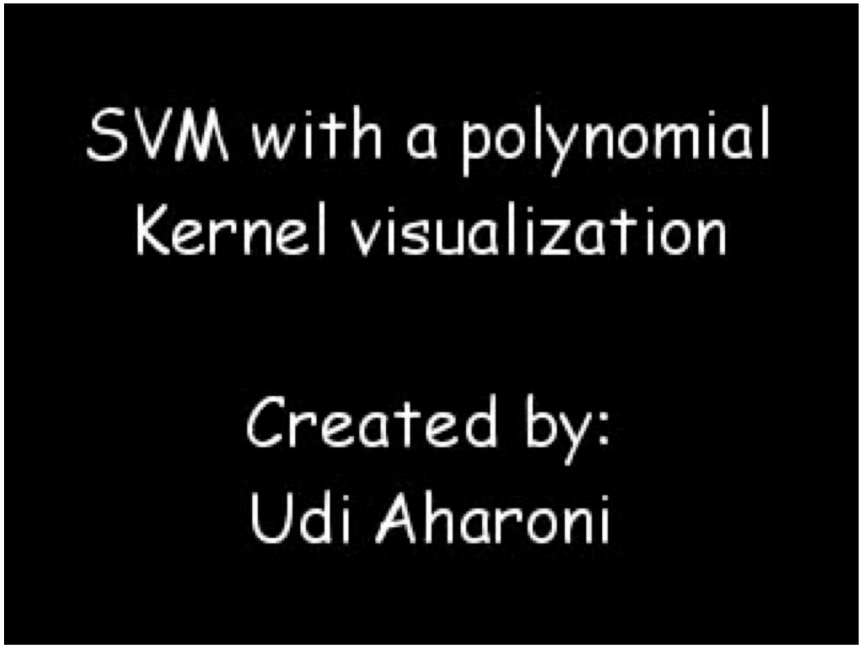
$$\begin{cases} \mathbf{w}^T \varphi(\mathbf{x}_i) + b \geq +1, & \text{if } \; y_i = +1 \\ \mathbf{w}^T \varphi(\mathbf{x}_i) + b \leq -1, & \text{if } \; y_i = -1 \end{cases} \tag{5}$$

which is equivalent to

$$y_i[\mathbf{w}^T \varphi(\mathbf{x}_i) + b] \geq 1, \quad i = 1, ..., N.$$

SVM

- What's the $\varphi(\cdot)$?
- What if data is non-linearly separable?



SVM

- $\varphi(\cdot)$ is a non-linear function that maps the input space to a high (possibly infinite) dimensional feature space
- Equation 5 constructs a hyperplane $\mathbf{w}^T \varphi(\mathbf{x}) + b = 0$ between the two classes
- Goal: maximize margin between both classes
  $\rightarrow$ minimizing $\mathbf{w}^T \mathbf{w}$

SVM (optional)

This convex optimization problem is defined as:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{J}(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \sum_{i=1}^{N} \xi_i \tag{7}$$

subject to

$$\begin{cases} y_i[\mathbf{w}^T\varphi(\mathbf{x}_i) + b] \geq 1 - \xi_i, & i = 1, ..., N \\ \xi_i \geq 0, & i = 1, ..., N. \end{cases} \tag{8}$$

- ξ_i are slack variables which are needed to allow misclassifications in the set of inequalities
- C is the regularisation coefficient $\rightarrow$ tuned

SVM (optional)

- The objective function (Eq. 7):
  - First part:
    - max. margin between both classes in the feature space
    - regularisation mechanism that penalizes large weights
  - Second part:
    - min. the misclassfication error

SVM (optional)

This leads to the following classifier (Cristianini & Shawe-Taylor, 2000):

$$y(\mathbf{x}) = \text{sign}[\sum_{i=1}^{N} \alpha_i \, y_i \, K(\mathbf{x}_i, \mathbf{x}) + b], \tag{9}$$

- The Lagrange multipliers α_i are determined by optimizing the dual problem
- Low-noise problems, many of the α_i will typically be equal to zero (sparseness property)
- Support vectors: the training observations corresponding to non-zero α_i

SVM

$K(\mathbf{x}_i, \mathbf{x}) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x})$ is taken with a positive definite kernel satisfying the Mercer theorem.

The following kernel functions $K(\cdot, \cdot)$ were used:

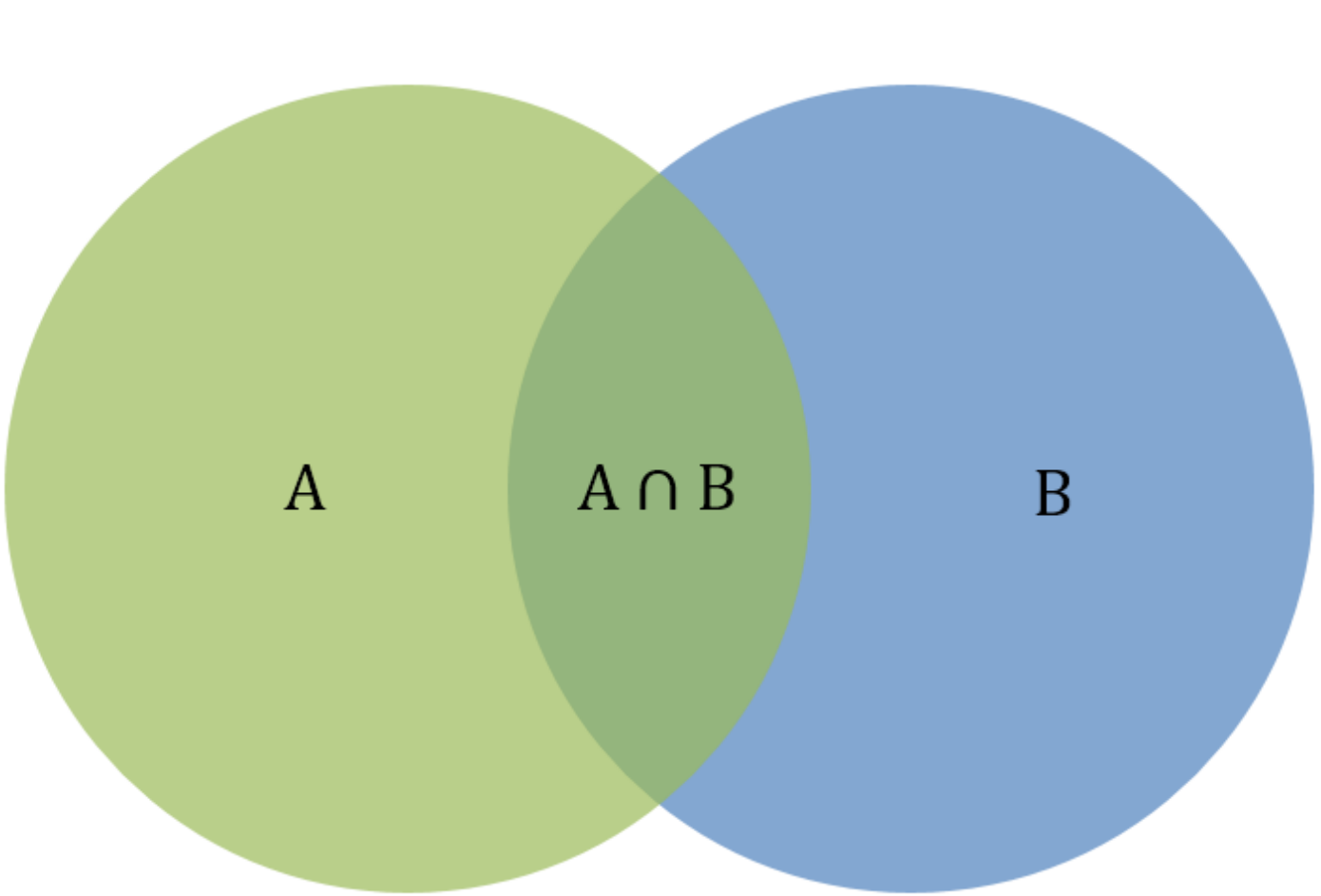
$$K(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x}_i^T \mathbf{x}/c)^d, \quad \text{(polynomial kernel)}$$
$$K(\mathbf{x}, \mathbf{x}_i) = \exp\{-\|\mathbf{x} - \mathbf{x}_i\|_2^2/\sigma^2\}, \quad \text{(RBF kernel)}$$

where d, c and σ are constants.

$\rightarrow$ black-box model

Tuning hyperparameters

- GridSearch
  - Grid is determined by the two input parameters
  - 2-fold cross validation on the initial grid
  - 10-fold cross validation on the best point and its adjacent point (based on the weighted AUC by class size)
  - If a better pair is found: repeat procedure on its neighbours
  - Stop: no better pair found or border reached



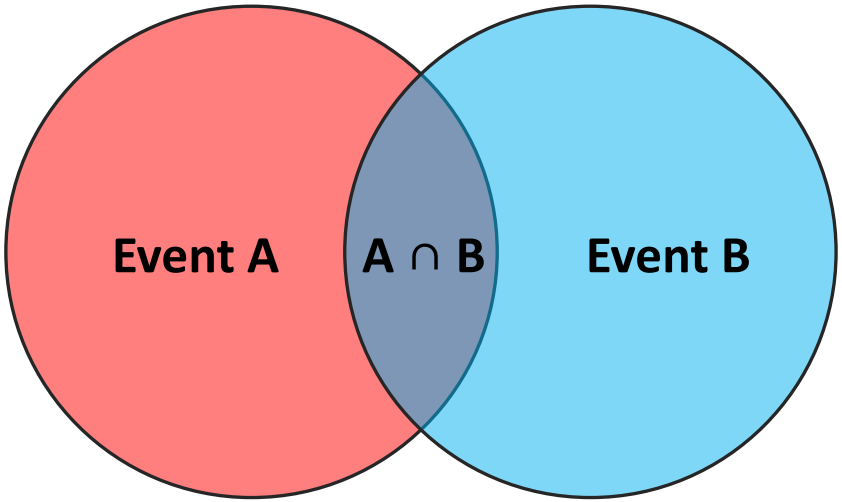
Naïve Bayes

Recap on Bayes Theorem

- Conditional Probability:

$$P(A \mid B) = \frac{P(A,B)}{P(B)}$$

$$P(B \mid A) = \frac{P(A,B)}{P(A)}$$



	Female	Male	Total
Teacher	8	12	20
Student	32	48	80
Total	40	60	100

$$P(Teacher \mid Male) = \frac{P(Teacher \cap Male)}{P(Male)} = 12/60 = 0.2$$

Recap on Bayes Theorem

- Conditional Probability:

$$P(A \mid B) = \frac{P(A,B)}{P(B)}$$

$$P(B \mid A) = \frac{P(A,B)}{P(A)}$$

$$P(A,B) = P(B \mid A)\, P(A)$$

- Bayes Theorem:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

Naïve Bayes

- In real world problem: multiple X-variables
- **Assumes the features are independent:**

$$P(A = k \mid X_1 \dots Xn) = \frac{P(X_1 \mid A = k) * P(X_2 \mid A = k) * \dots * P(X_n \mid A = k) * P(A = k)}{P(X_1) * P(X_2) * \dots * P(Xn)}$$

$$\text{Probability of Outcome} \mid Evidence = \frac{Probability\ of\ Likelyhood\ of\ evidence * Prior}{Probability\ of\ Evidence}$$

(Posterior Probability)

Naïve Bayes example

- Dataset with probabilities:

Fruit	Long (x1)	Sweet (x2)	Yellow (x3)
Orange	0	1	0
Banana	1	0	1
Banana	1	1	1
Other	1	1	0
..	..	..	..

- Aggregated to a counts table:

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/

Naïve Bayes example

- Objective: predict if a given fruit is Banana, Orange, or Other when only three features are know (long, sweet and yellow)
- Step 1: compute Prior probabilities for each class of fruits:
  - P(Y=Banana) = 500 / 1000 = 0.50
  - P(Y=Orange) = 300 / 1000 = 0.30
  - P(Y=Other) = 200 / 1000 = 0.20

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

Naïve Bayes example

- Step 2: Compute the probability of evidence that goes in the denominator.
  - P(x1=Long) = 500 / 1000 = 0.50
  - P(x2=Sweet) = 650 / 1000 = 0.65
  - P(x3=Yellow) = 800 / 1000 = 0.80

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

Naïve Bayes example

- Step 3: Compute the probability of likelihood of evidences that goes in the numerator.
  - P(x1=Long | Y=Banana) = 400 / 500 = 0.80
  - P(x2=Sweet | Y=Banana) = 350 / 500 = 0.70
  - P(x3=Yellow | Y=Banana) = 450 / 500 = 0.90

The overall probability of Likelihood of evidence for Banana = 0.8 * 0.7 * 0.9 = 0.504

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

Naïve Bayes example

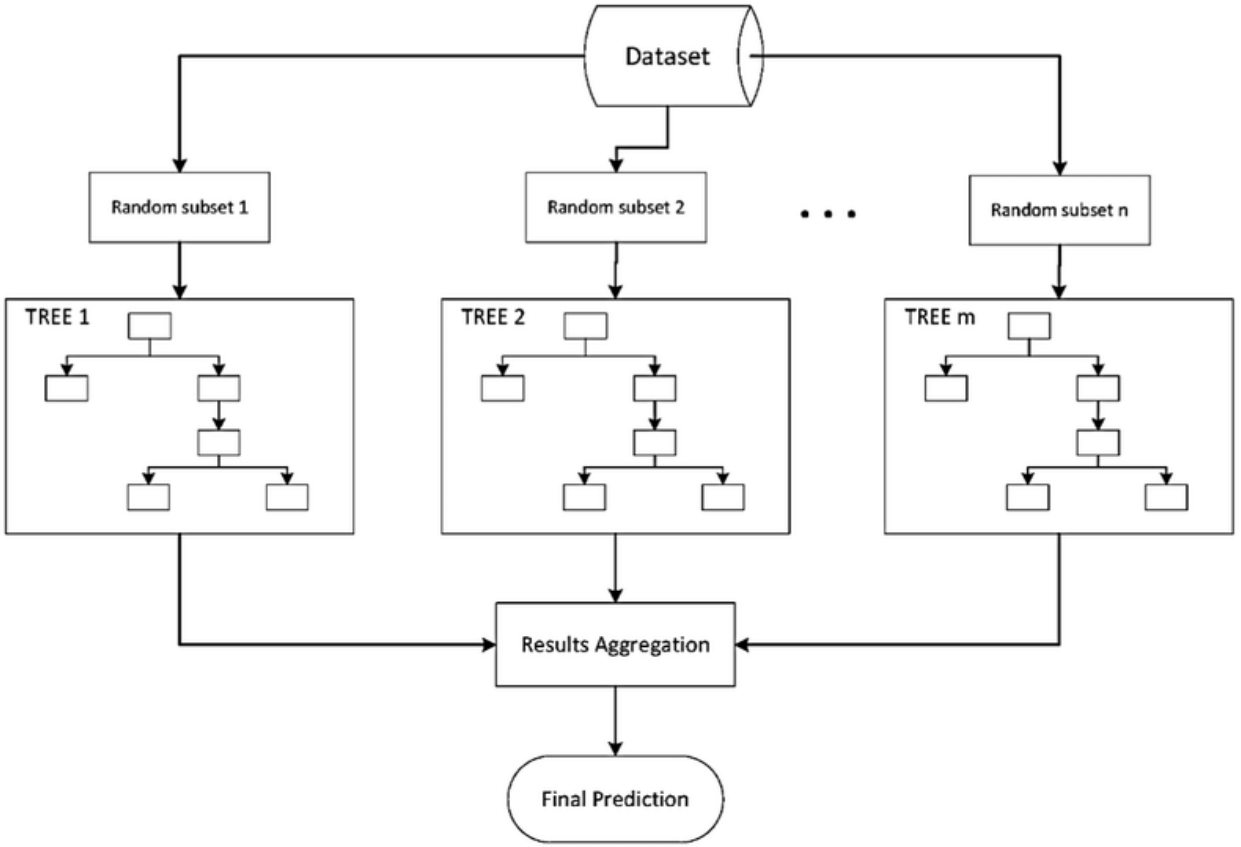
- Step 4: Substitute all the 3 equations into the Naive Bayes formula, to get the probability that it is a banana.
  - P(**Banana** | Long, Sweet and Yellow) = 0.252| P(Evidence). ← 0.504 * 0.5
  - P(Orange | Long, Sweet and Yellow) = 0 | P(Evidence)
  - P(Other | Long, Sweet and Yellow) =0.01875 | P(Evidence)

P(evidence) is the same for all, so optional to compute

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

Laplace Correction

- The value of P(Orange | Long, Sweet and Yellow) was zero in the above example, because, P(Long | Orange) was zero.
- When you have a model with many features, the entire probability will become zero because one of the feature's value was zero.
- To avoid this, we increase the count of the variable with zero to a small value (usually 1) in the numerator, so that the overall probability doesn't become zero.



Ensemble methods

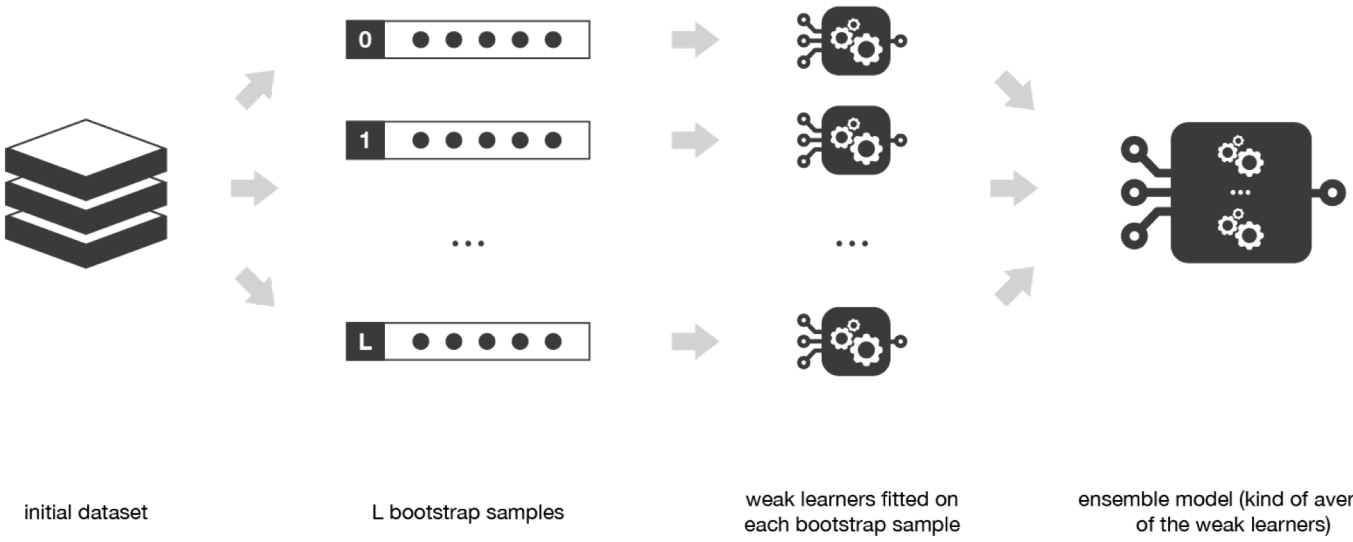
Ensemble Methods

- Construct a set of classifiers from the training data
- Predict class label of test records by combining the predictions made by multiple classifiers

Types of ensemble methods

- **Bagging:** considers homogeneous weak learners, learns them independently from each other in parallel and combines them following some kind of deterministic averaging process
- **Boosting:** considers homogeneous weak learners, learns them *sequentially* in a very adaptive way (a base model depends on the previous ones) and combines them following a deterministic strategy
- **Stacking:** considers *heterogeneous* weak learners, learns them in parallel and combines them by training a meta-model to output a prediction based on the different weak models predictions

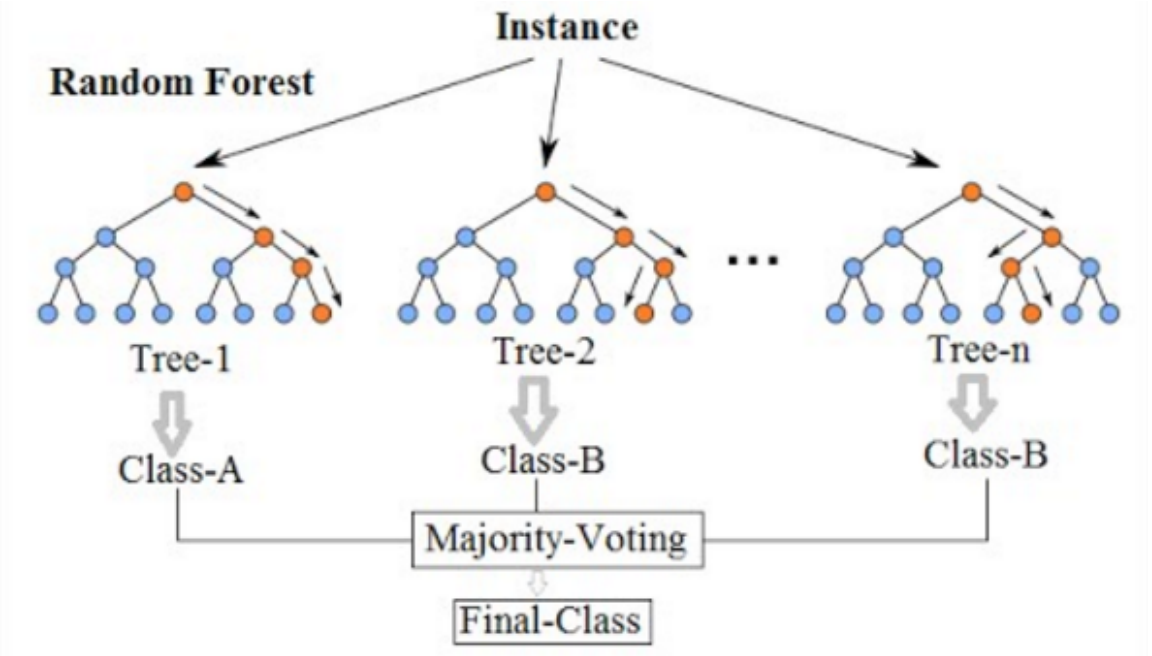
Bagging



https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

Random Forest

- Special *bagging* case
- An ensemble approach that trains and combines the results of multiple decision trees
- Sample over features



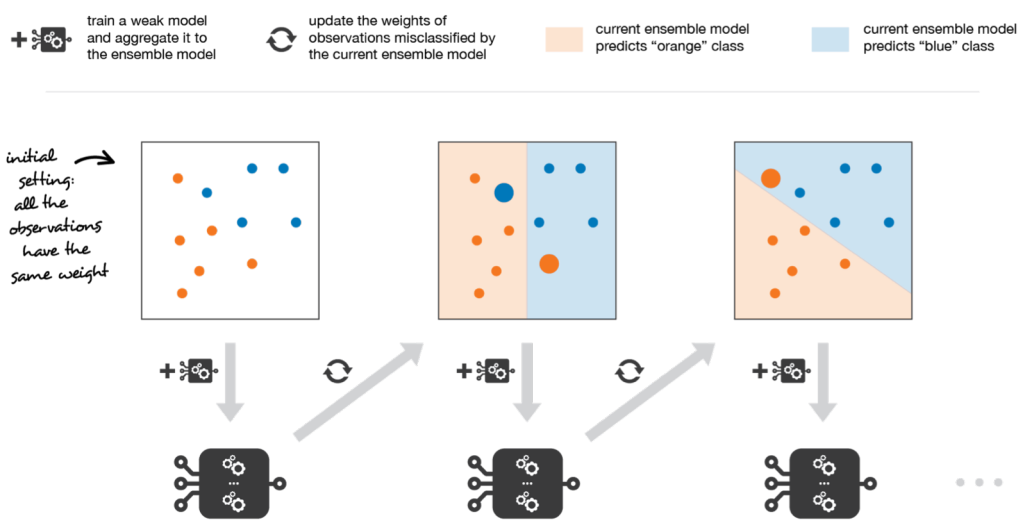
https://commons.wikimedia.org/wiki/File:Random_forest_diagram_complete.png

Characteristics of DT and RF

- Decision Trees
  - Inexpensive to construct tree
  - Results are easy to explain/interpret (for small trees!)
  - Prone to overfitting (if no pruning, early stopping)
- Random Forests
  - Multiple decision trees
    - Using different data sample (bagging) and random training features
  - Avoid overfitting
  - Improve accuracy

Boosting

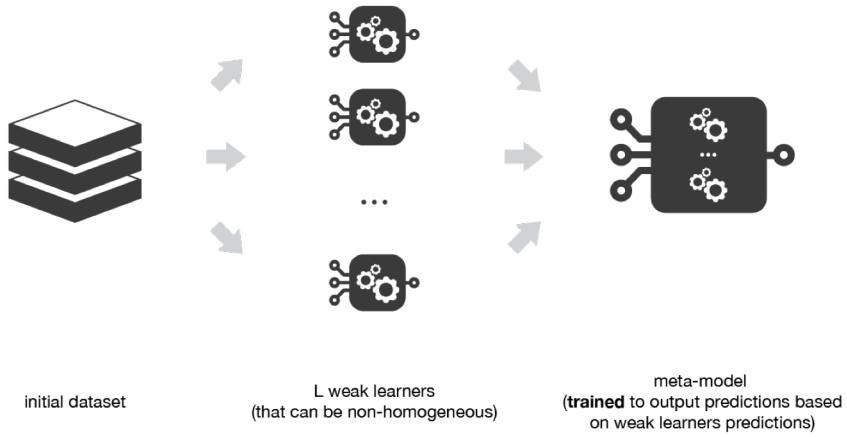
- Fit models iteratively such that the training of model at a given step depends on the models fitted at the previous steps
  - **Records that are wrongly classified will have their weights increased**
  - Records that are classified correctly will have their weights decreased
- Two popular techniques:
  - Gradient boosting
  - Adaptive boosting (AdaBoost) -> Fig. on right



Stacking

- learn several different weak learners and combine them by training a meta-model to output predictions based on the multiple predictions returned by these weak models.
- For example, we can choose as weak learners a KNN classifier, a logistic regression and a SVM, and decide to learn a neural network as meta-model.

Then, the neural network will take as inputs the outputs of our three weak learners and will learn to return final predictions based on it.

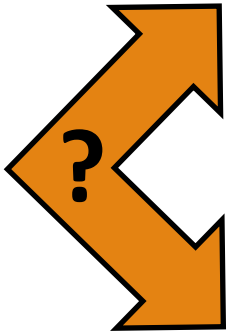


Evaluation

Training and Test Splits

- How do we determine which observation goes into the training and test sets?

Dataset
O1
O2
O3
O4
O5
O6
O7
O8
O9
O10
O11
O12
O13
O14
O15
O16

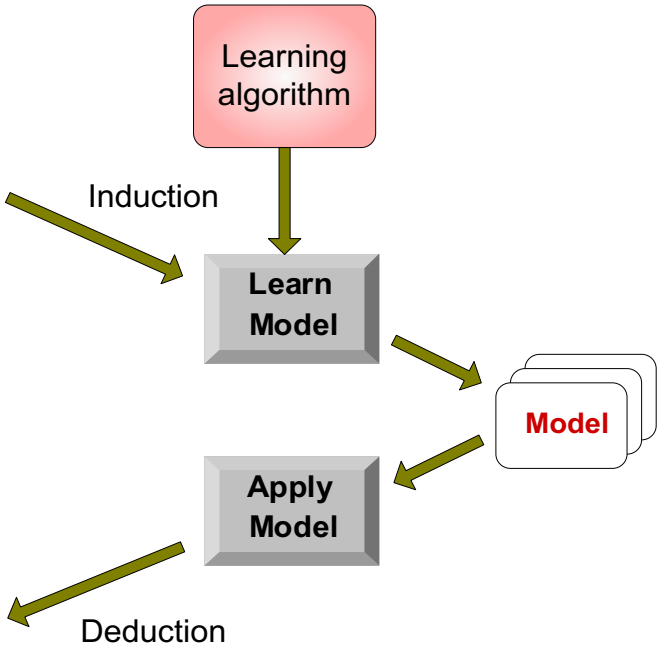


Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

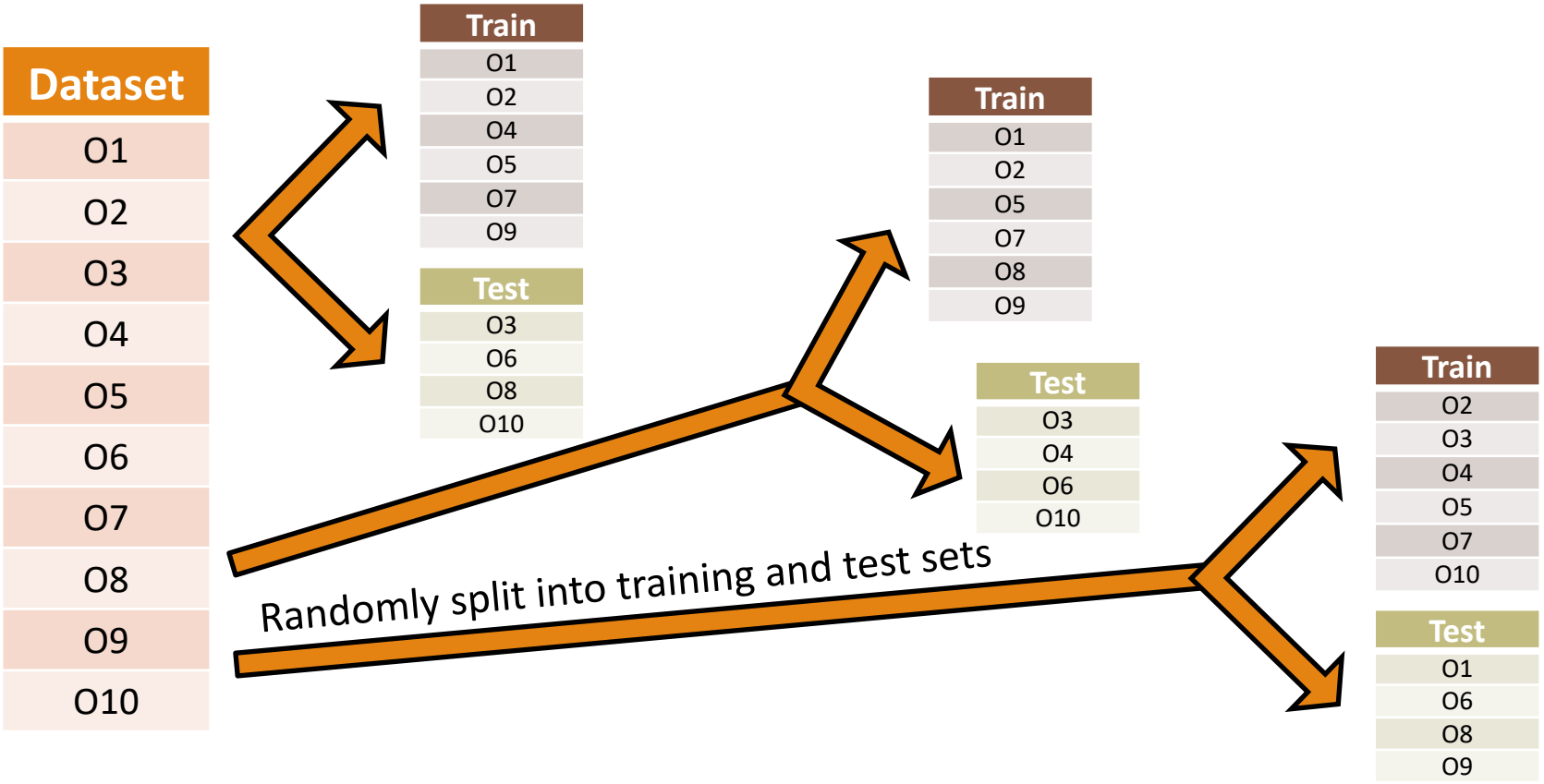
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



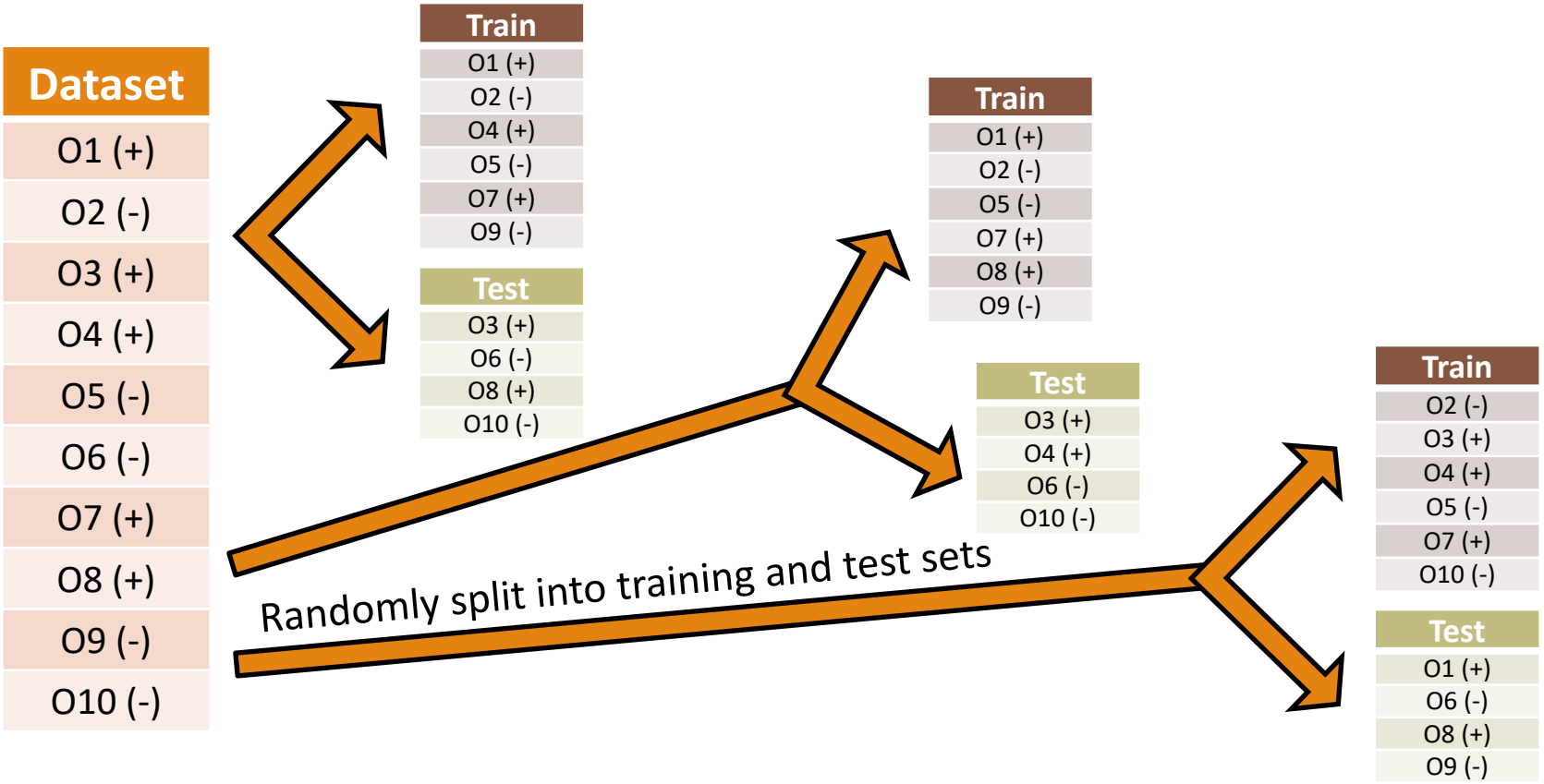
Random Resampling

- Generate training and test sets by randomly sampling from the dataset



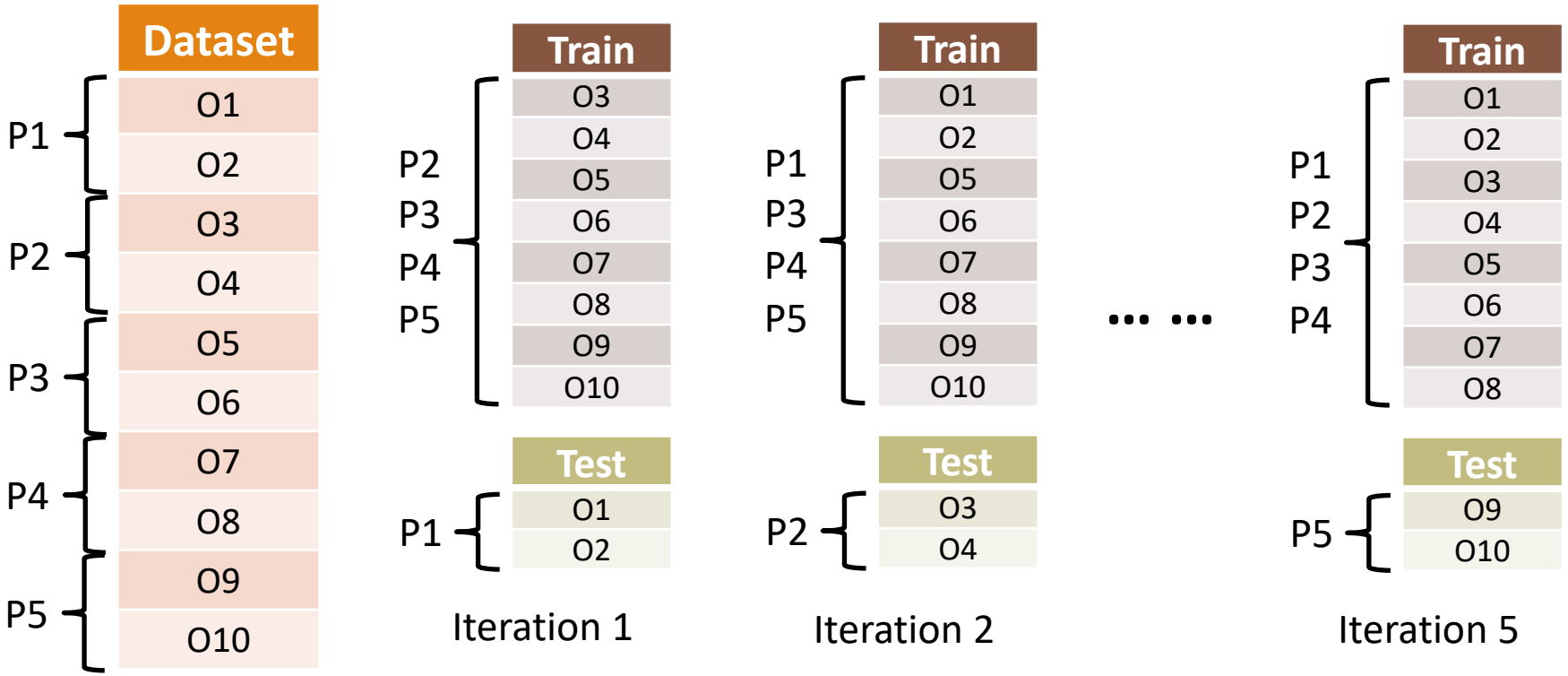
Stratified Sampling

- Similar to random sampling, but maintain proportion of class labels



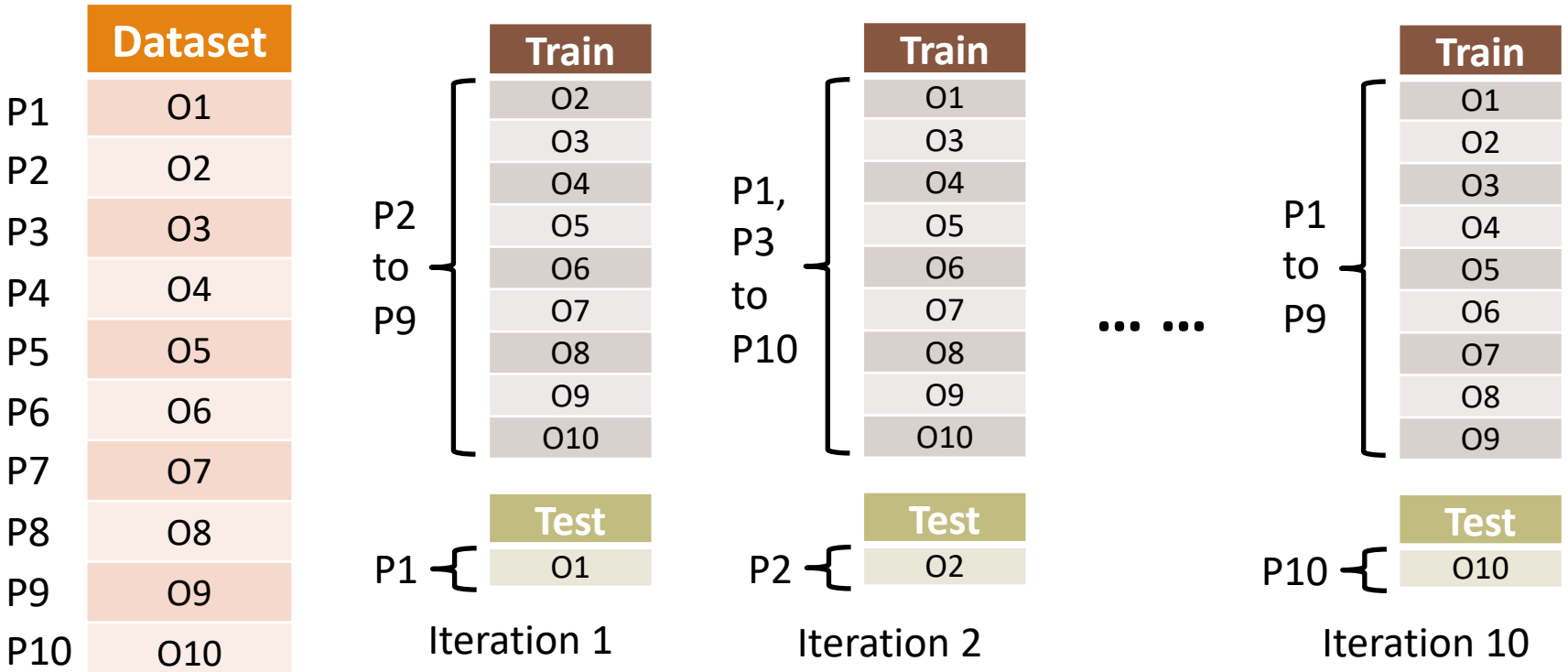
K-Fold Cross Validation

- Split dataset into *k* partitions, iteratively use one partition as test set



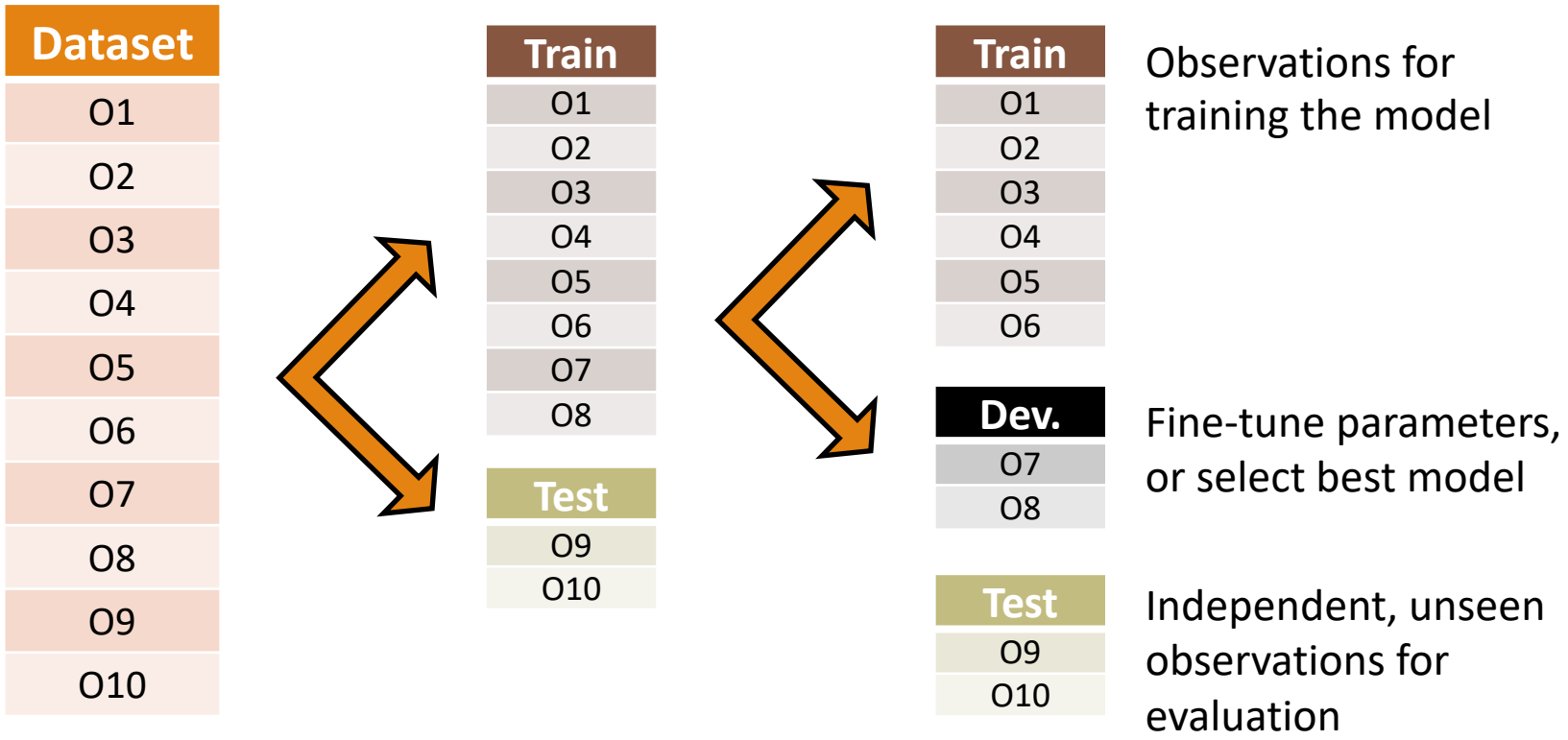
Leave-one-out Validation

- Special case of *k*-fold cross validation, where *k* = no. of observations



Development Set

- Used for tuning hyper-parameters or select a model among many



Confusion Matrix

- A confusion matrix allows us to understand the prediction results
  - E.g., predicting if a person has a cold (Yes) or not (No)
  - E.g., determining if a retrieved result is relevant (Yes) or not (No)

		Ground Truth (Relevant)	
		Yes	**No**
Classification (Retrieved)	**Yes**	TP	FP
	No	FN	TN

Accuracy

- Accuracy = % of correct classification, out of all observations

  = (TP + TN) / (TP + FP + TN + FN)

- Problem: unbalanced classes

		Ground Truth (Relevant)	
		Yes	**No**
Classification (Retrieved)	**Yes**	TP	FP
	No	FN	TN

Precision and Recall

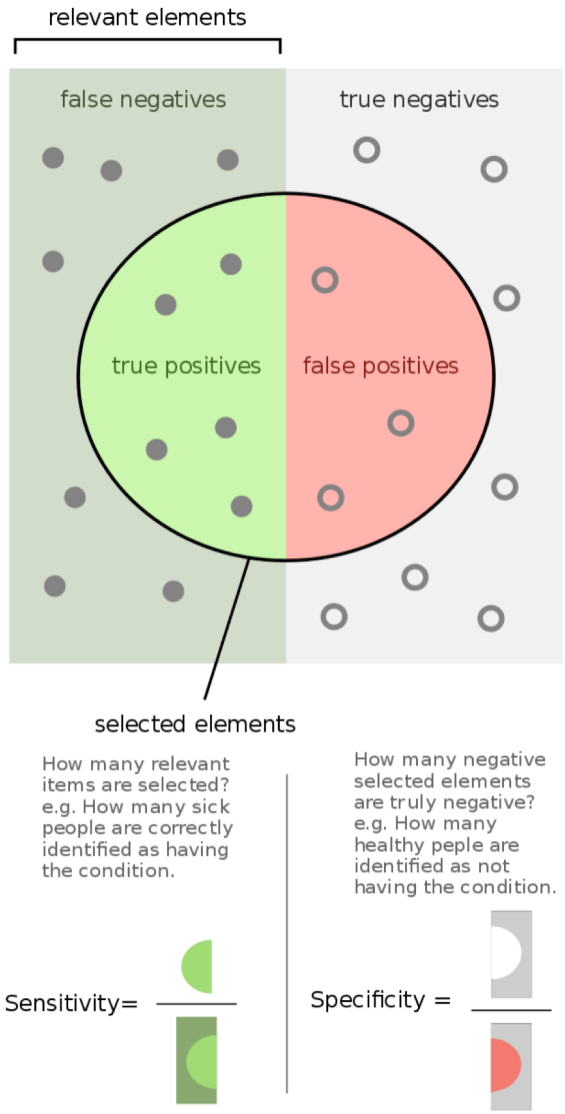
- Precision = % of classified (retrieved) items that are correct (relevant)

  = TP / (TP + FP)

- Recall = % of correct (relevant) items that are classified (retrieved) (aka specificity)

  = TP / (TP + FN)

		Ground Truth (Relevant)	
		Yes	**No**
Classification (Retrieved)	**Yes**	TP	FP
	No	FN	TN



https://en.wikipedia.org/wiki/Sensitivity_and_specificity

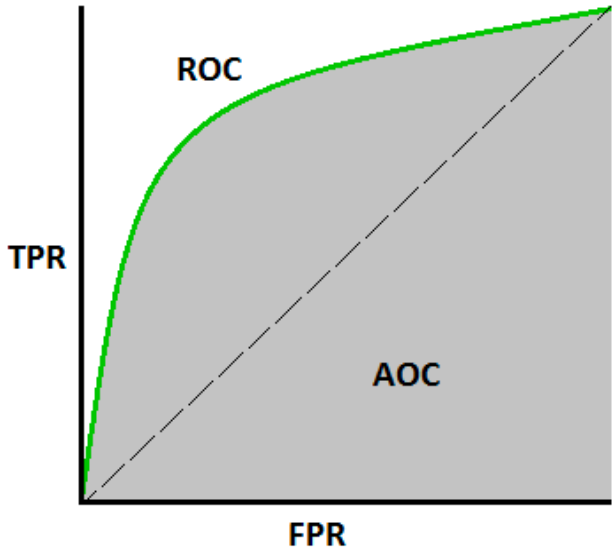
F1-score

- Problems with Precision and Recall
  - Trade-off between precision and recall
  - To get 100% recall, just retrieve everything

- $\text{F1 score} = \frac{2 \times Precision \times Recall}{Precision + Recall}$
  - i.e., Taking the harmonic mean of Precision and Recall

Receiver Operating Curve - ROC

- Random classification = diagonal line
- AOC or AUC: area under the curve: 0.5 for random, higher is better.

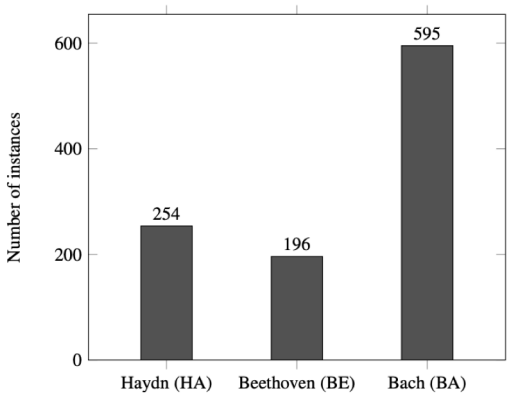


Classifying MIDI files per composer

Table 2 Performance of the models with 10-fold cross-validation

Method	Accuracy	AUC
C4.5 Decision tree	*80%*	*79%*
RIPPER ruleset	*81%*	*85%*
Logistic regression	*83%*	92%
Naive Bayes	*80%*	90%
Support vector machines	**86%**	**93%**

$p < 0.01$: italic, $p > 0.05$: bold, best: **bold**.



Class distribution

Table 9 Confusion matrix for support vector machines

a	b	c	classified as
204	26	24	a = HA
49	**127**	20	b = BE
22	10	**563**	c = BA

Book chapter: Herremans D., Martens D, Sörensen K., Meredith D.. 2015. Composer Classification Models for Music-Theory Building. Computational Music Analysis. Springer