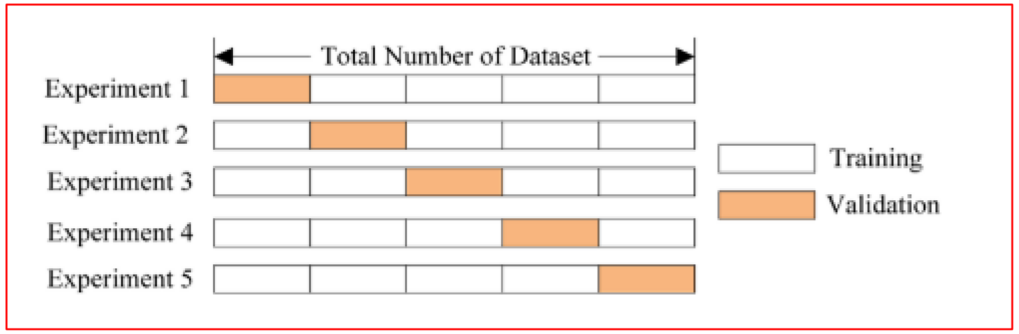
Introduction to Machine Learning Applications Part 2

K-Fold Cross-Validation



- In k-fold cross-validation, the samples are randomly partitioned into k roughly equal sized subsamples.
- Among the k sample sets, the first subset is retained as the validation data for testing the model and the held-out samples are used as training data to fit the model.
- The first subset is then return to the training data and the process repeats k times while each of the k subsets used exactly once as the validation data.
- The k resampled estimates of performance are aggregated and summarized to produce a single estimation

Feature Engineering

- features for representing *categorical data*, features for representing *text*, and features for representing *images*

- *Check the python script*

Classification

Classification maps data into predefined groups or classes

It is useful in

- ➢ Supervised learning
- ➢ Pattern recognition
- ➢ Prediction

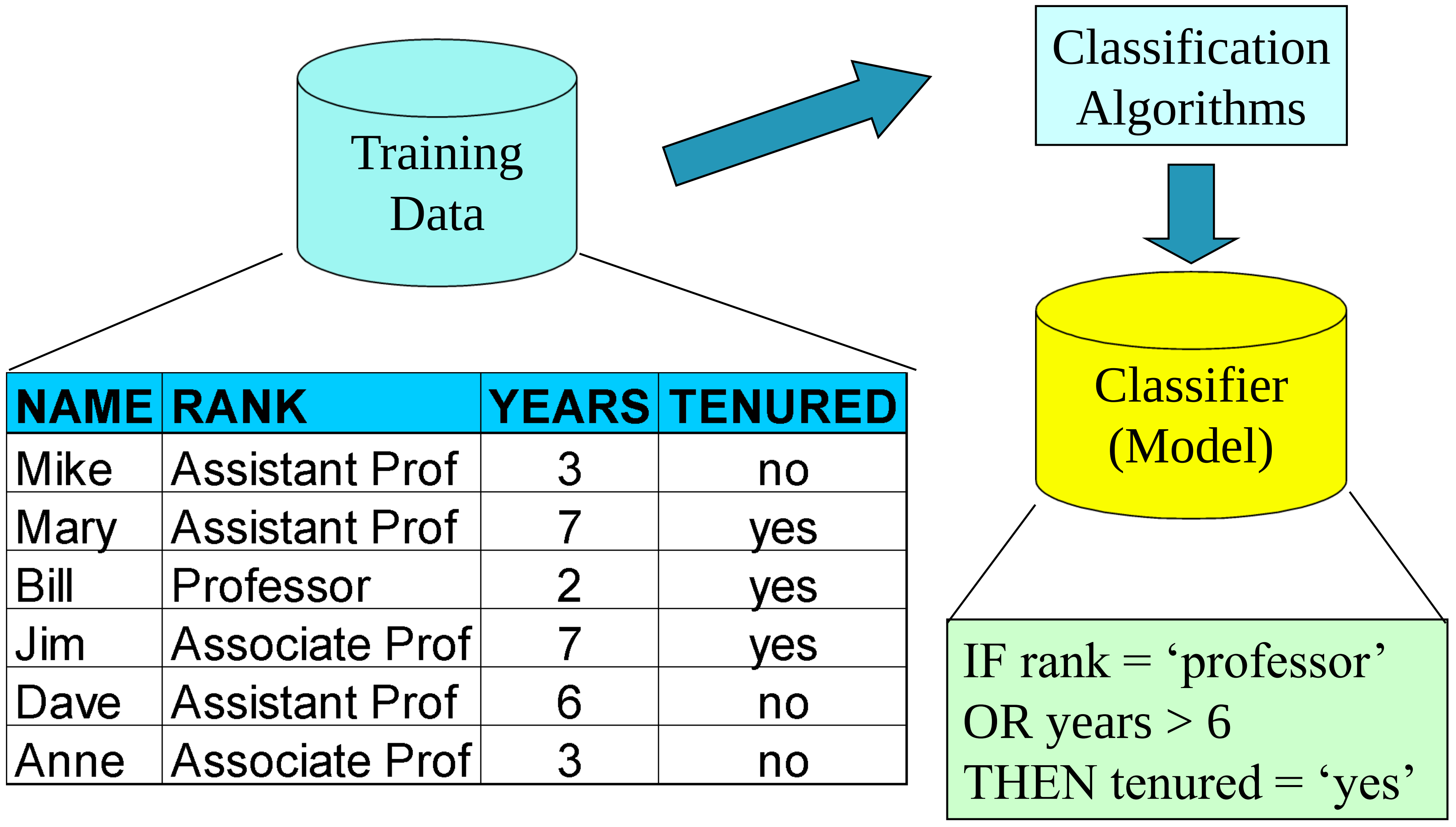
Classification Vs Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit approval
  - Target marketing
  - Medical diagnosis
  - Fraud detection

Classification Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

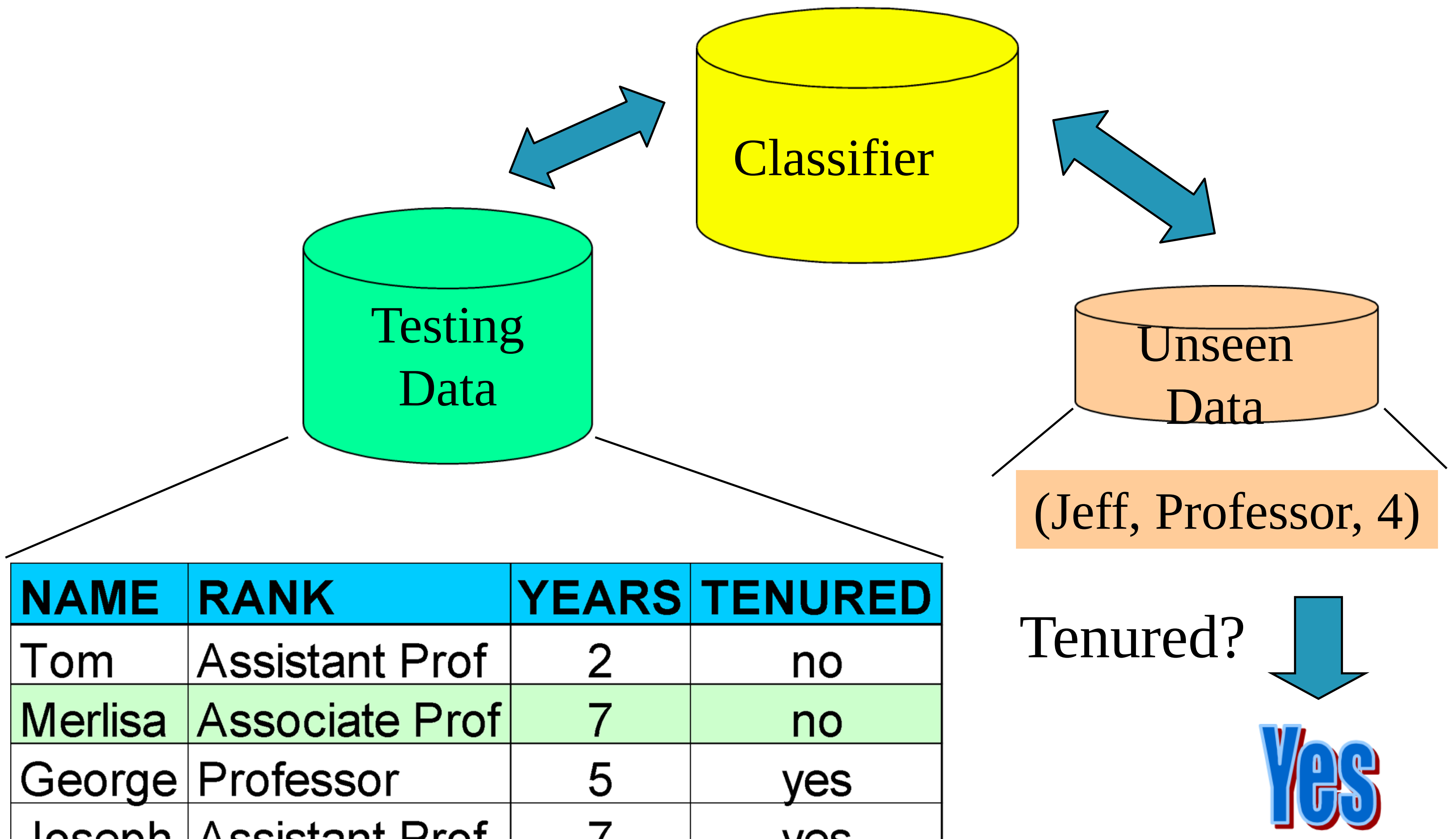
Step 1: Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Step 2: Model Usage



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Bayesian Classification

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes’ Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Supervised learning : Classification

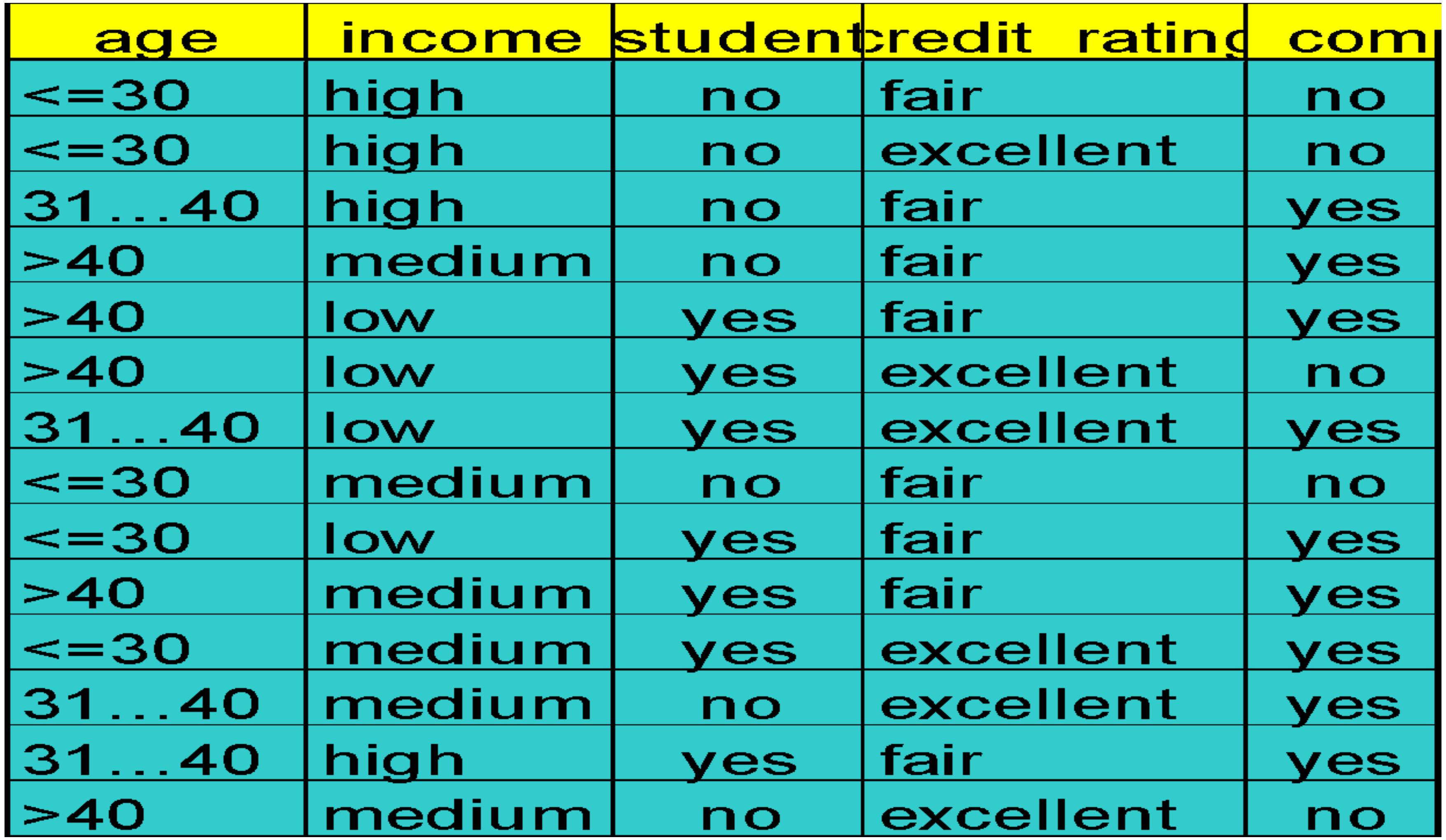
- Naive Bayes models are a group of extremely fast and simple classification algorithms
- often suitable for very high-dimensional datasets.
- Because they are so fast and have so few tunable parameters, they end up being very useful as a quick-and dirty baseline for a classification problem.
- This section will focus on an intuitive explanation of how naive Bayes classifiers work, followed by a couple examples of them in action on some datasets.

Contents

Bayesian Classification

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers

Example



age	income	student	credit ratin	com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

>40

Probability

- *Probability:* How likely something is to happen

- Probability of an event happening =

$$\frac{\text{Number of ways it can happen}}{\text{Total number of outcomes}}$$

Bayesian Theorem Basics

- Let **X** be a data sample ("*evidence*"): class label is unknown
- Let H be a *hypothesis* that X belongs to class C

$$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})}$$

- Classification is to determine P(H|**X**), the probability that the hypothesis holds given the observed data sample **X**
- P(H) (*prior probability*), the initial probability
  - E.g., **X** will buy computer, regardless of age, income, …
- P(**X**): probability that sample data is observed
- P(**X**|H) (*posteriori probability*), the probability of observing the sample **X**, given that the hypothesis holds
  - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Bayesian Theorm

- Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})}$$

- Informally, this can be written as

  posteriori = likelihood x prior/evidence

- ***Predicts X belongs to C_2 iff the probability $P(C_i|X)$ is the highest among all the $P(C_k|X)$ for all the k classes***
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Naiive Bayesian

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector **X** = (x_1, x_2, …, x_n)
- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal P(C_i|**X**)
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Training Data

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

Data sample
X = (age =31..40,
Income = low,
Student = yes
Credit_rating = excellent)

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

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

Test for X = (age <= 30 , income = medium, student = yes, credit_rating = fair)

- P(C_i): **P(buys_computer = “yes”) = 9/14 = 0.643** **P(buys_computer = “no”) = 5/14= 0.357**
- Compute P(X|C_i) for each class

P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222

P(age = “<=30” | buys_computer = “no”) = 3/5 = 0.6

P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444

P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4

P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667

P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2

P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667

P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

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

- **P(X|C_i) :** P(X|buys_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

  P(X|buys_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

P(X|C_i)*P(C_i) : P(X|buys_computer = “yes”) * P(buys_computer = “yes”) = 0.028

P(X|buys_computer = “no”) * P(buys_computer = “no”) = 0.007

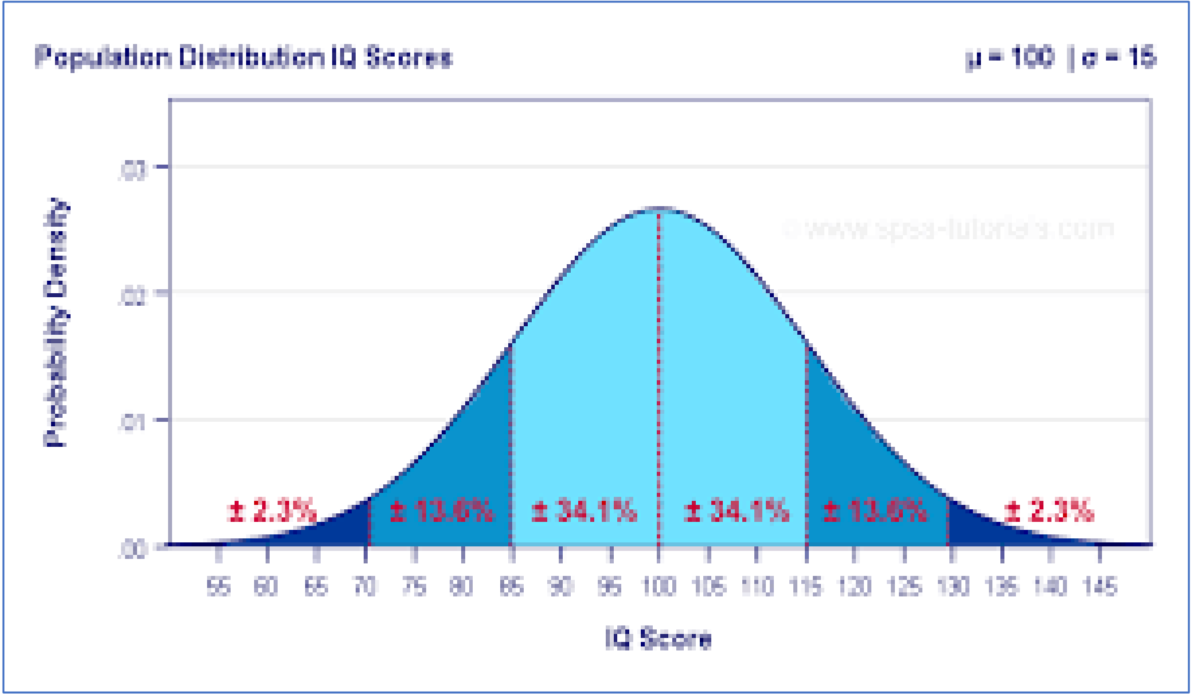
0.028>0.007 ..
Therefore, X belongs to class (“buys_computer = yes”)

GaussianNB

- GaussianNB implements the Gaussian Naive Bayes algorithm for classification.
- The likelihood of the features is assumed to be Gaussian type

Normal distribution, also known as the **Gaussian distribution**, is a probability **distribution** that is symmetric about the mean, showing that **data** near the mean are more frequent in occurrence than **data** far from the mean. In graph form, **normal distribution** will appear as a bell curve

For **example**, heights, blood pressure, measurement error, and IQ scores follow the **normal distribution**. It is also known as the **Gaussian distribution** and the bell curve.



Multinomial naive Bayes

- Multinomial naive Bayes the features are assumed to be generated from a simple multinomial distribution.
- The multinomial distribution describes the probability of observing **counts among a number of categories.**
- So Multinomial naive Bayes is most appropriate for features that represent counts or count rates.
- One place where multinomial naive Bayes is often used is in text classification, where the features are related to word counts or frequencies within the documents to be classified.

Example

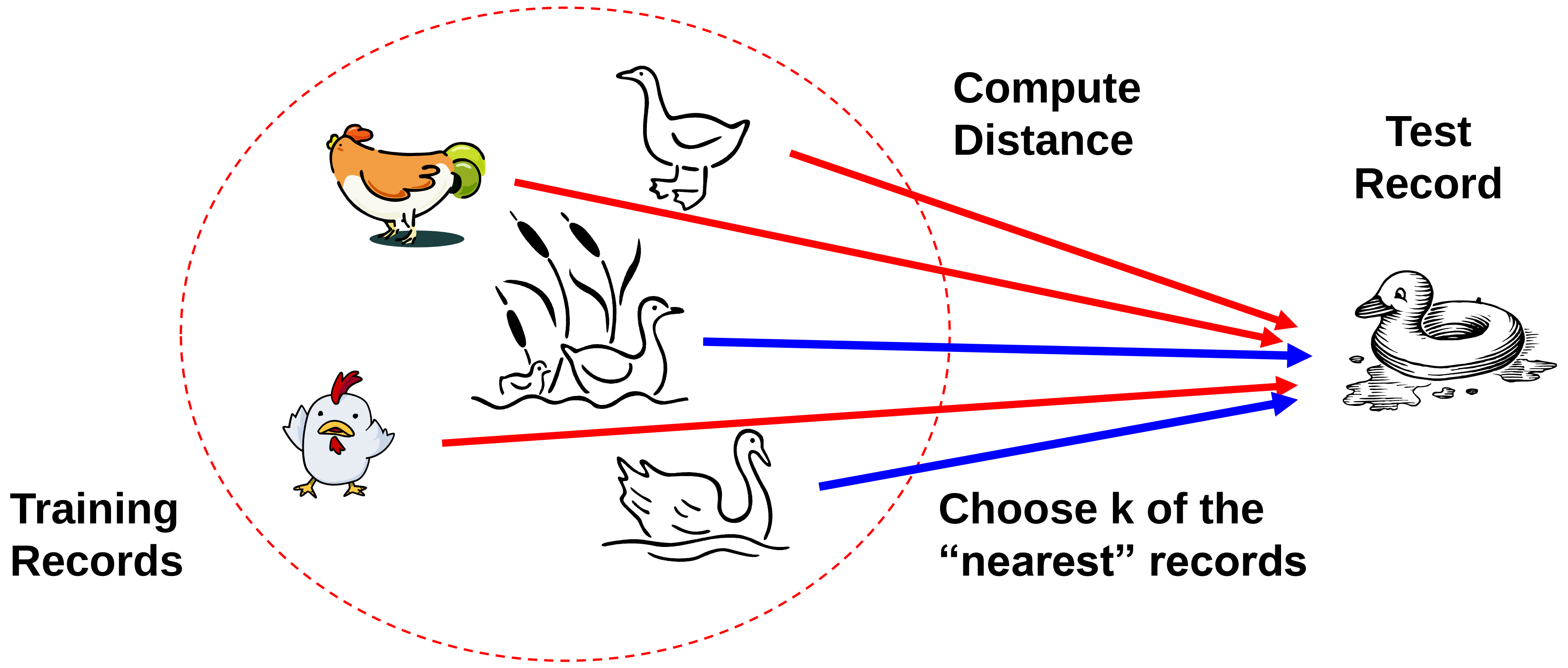
- Dataset Description: we will use the sparse word count features from the 20 Newsgroups corpus

from sklearn.datasets import fetch_20newsgroups

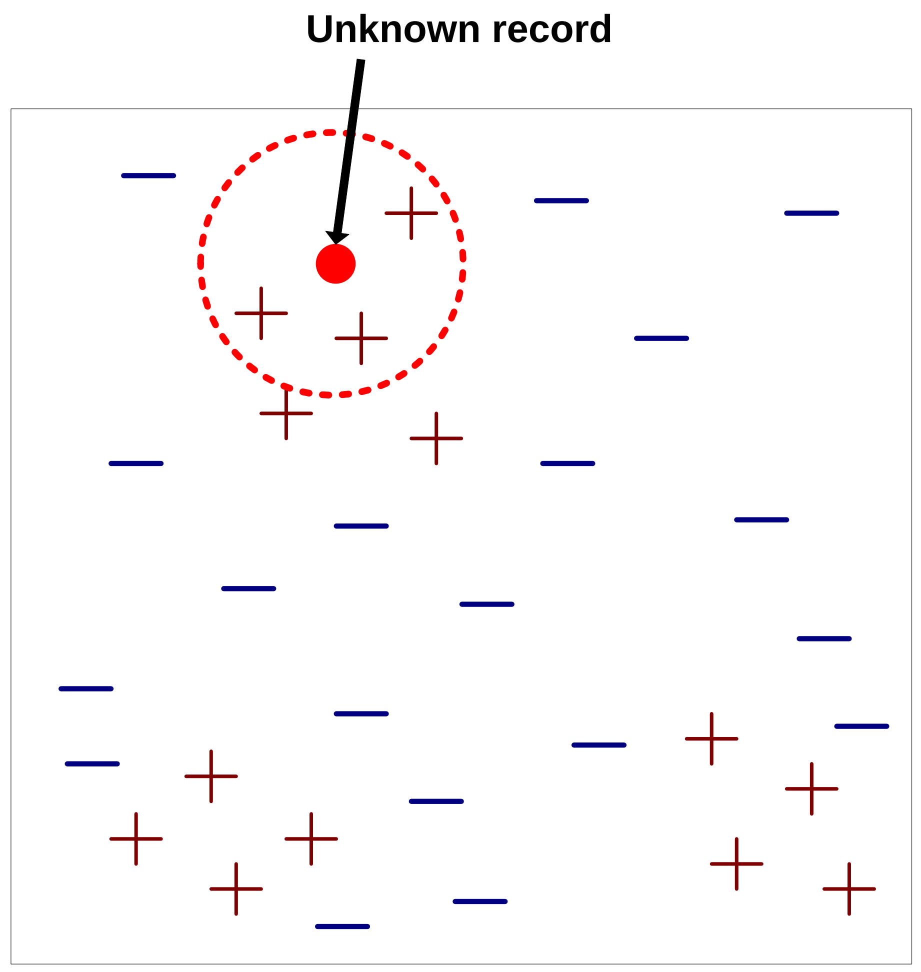
K Nearest neighbour

Nearest Neighbour Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



Nearest-Neighbor Classifiers



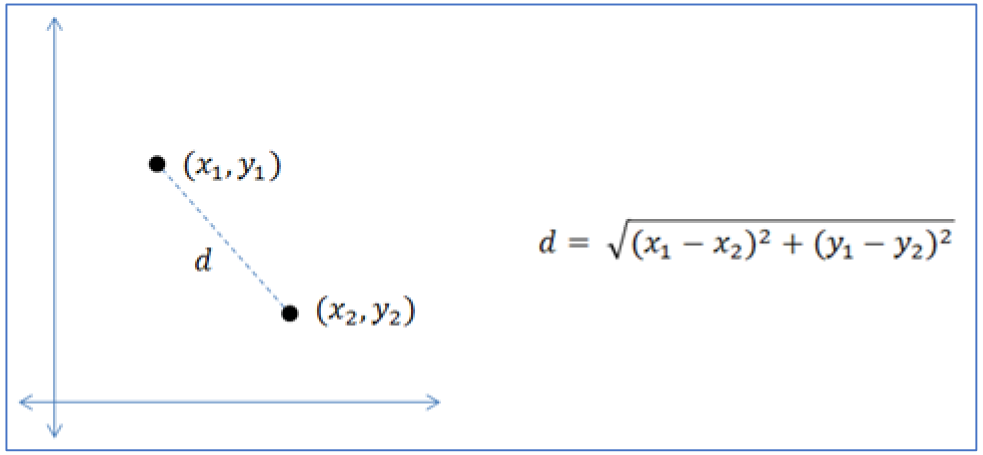
- Requires three things
  - The set of labeled records
  - Distance metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Nearest Neighbor Classification

- Compute Similarity between two points:
  - Example: Euclidean distance : minimum the distance , maximum the similarity

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_i (\boldsymbol{x}_i - \boldsymbol{y}_i)^2}$$



- Determine the class from nearest neighbor list
  - Take the majority vote of class labels among the k-nearest neighbors

Exercise: Consider the following data, find the class label for (6,6) with K=3

x	y	Class
2	4	N
4	6	N
4	4	P
4	2	N
6	4	N
6	2	P

1. Determine parameter K = number of nearest neighbors.

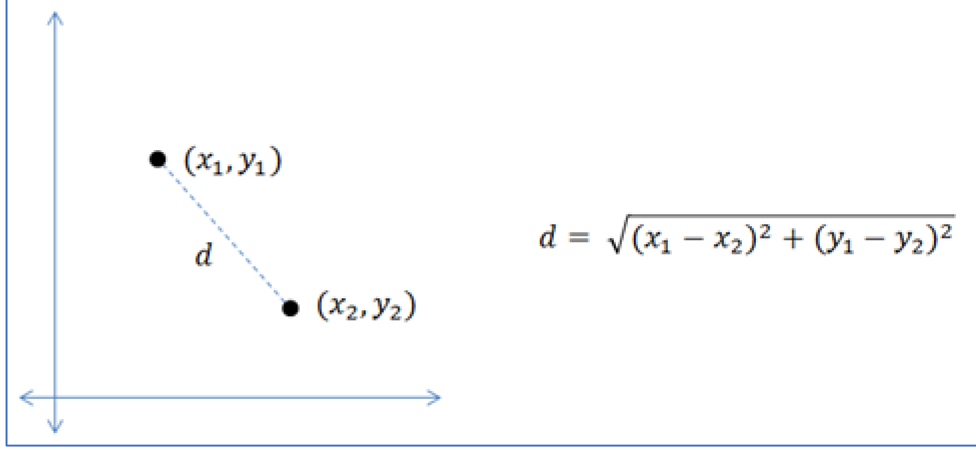
2. Calculate the distance between the query-instance and all the training samples.

3. Sort the distance and determine nearest neighbors based on the K-th minimum distance.

4. Gather the category of the nearest neighbors

5. Use a simple majority of the category of nearest neighbors as the prediction value of the query.

Example: Find the class label for (6,6) with K=3



Features		Target	D	E	F	G	H	I			
X	Y	Class	X-Xi	Y-Yi	D*D	E*E	F+G	sqrt(H)	k=3	Class Lable of Neighbor	Mode
2	4	N	-4	-2	16	4	20	4.472136			
4	6	N	-2	0	4	0	4	2	Consider	N	N
4	4	P	-2	-2	4	4	8	2.828427	Consider	P	
4	2	N	-2	-4	4	16	20	4.472136			
6	4	N	0	-2	0	4	4	2	Consider	N	
6	2	P	0	-4	0	16	16	4			

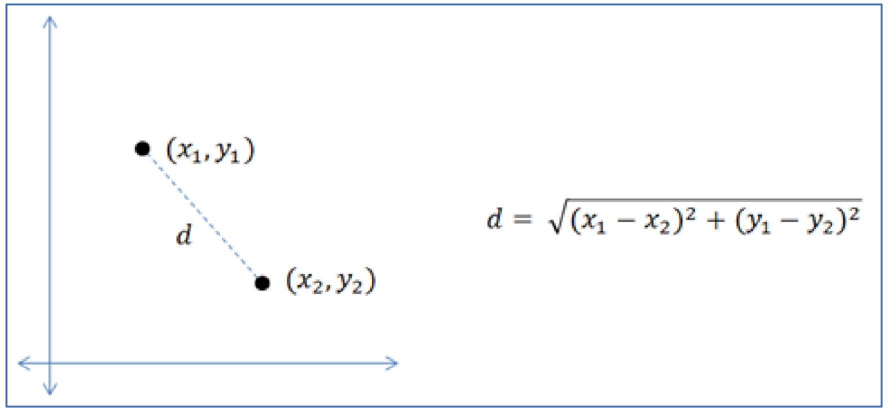
Example KNN

- Find weight of [5.50,26]

=mean(47,60,45)

=50.667

- K=3
- K=4

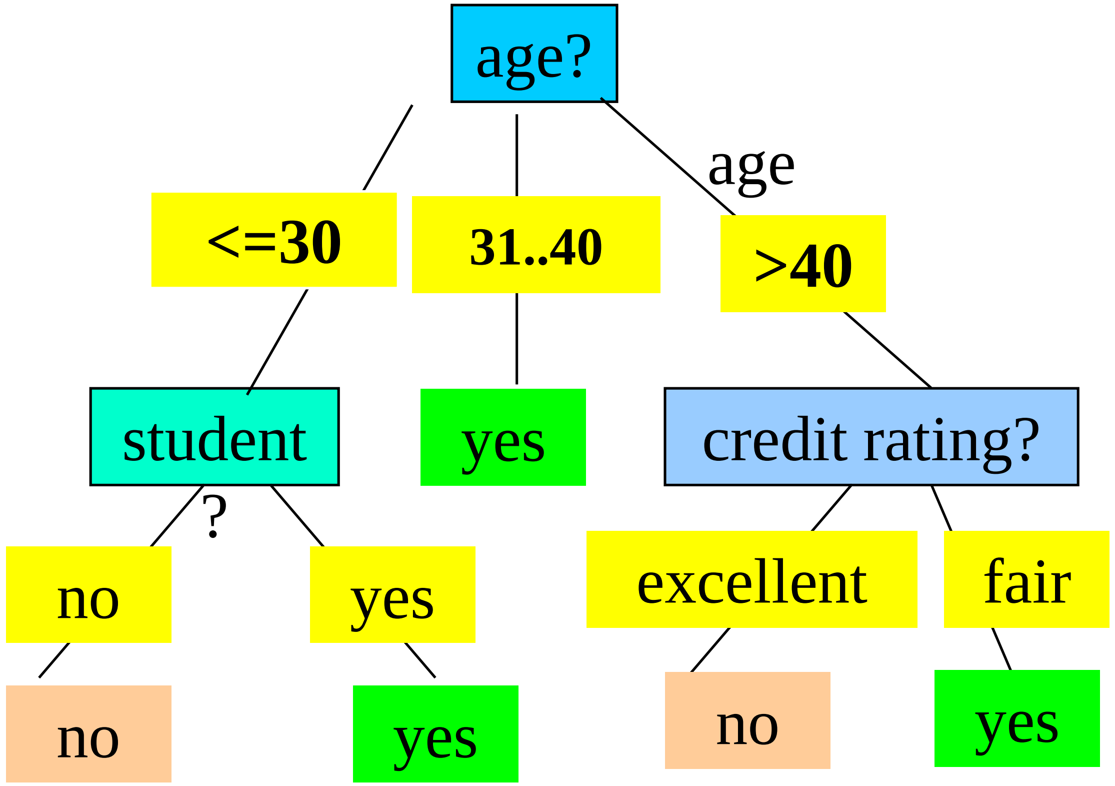


$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Height	Age	Weight	
5	45	77	19.065
5.11	26	47	0.39
5.6	30	55	4.0012
5.9	34	59	8.01
4.8	40	72	14.0175
5.8	36	60	10.0045
5.3	19	40	7.0029
5.8	28	60	2.0224
5.5	23	45	3
5.6	32	58	6.00083

Decision Tree

Decision Tree Classification



age	income	student	credit rating	buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Terminologies in DT

- Decision tree may be *n*-ary, $n \geq 2$ (In sk learn it is binary)
- There is a special node called root node.
- Internal nodes are test attribute/decision attribute
- leaf nodes are class labels
- Edges of a node represent the outcome for a value of the test node.
- Decision tree is not unique, as different ordering of internal nodes can give different decision tree

Rule Extraction From decision Tree

- Rules are easier to understand than large trees
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive

Example: Rule extraction from our *buys_computer* decision-tree

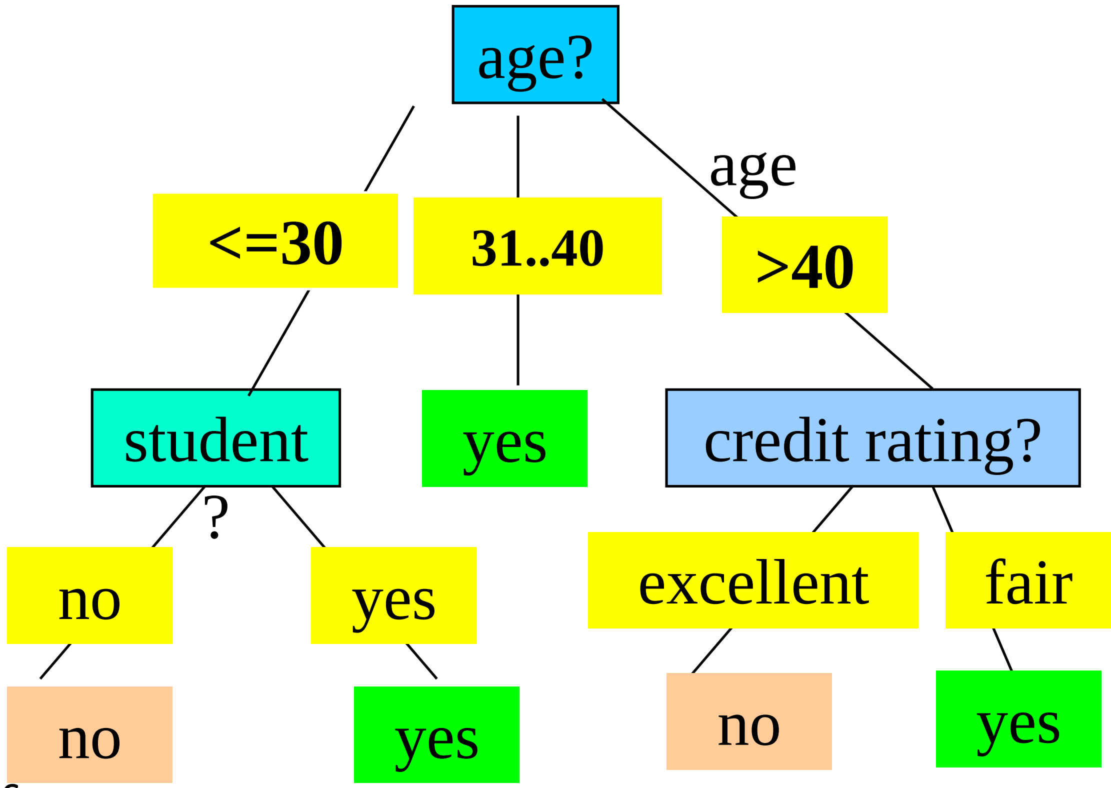
IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*

IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*

IF *age* = mid-age THEN *buys_computer* = *yes*

IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *yes*

IF *age* = young AND *credit_rating* = *fair* THEN *buys_computer* = *no*



Decision Tree Algorithm

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Test Attribute Selection

- **Select the attribute with the highest information gain**
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i, D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute Selection : Information gain

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

Class P: buys_computer = “yes”

Class N: buys_computer = “no”

1 : Calculate Entropy for Class Labels

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940$$

2 : Calculate Information of Each Attribute (one by one)

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

$\frac{5}{14}I(2,3)$ means “age <=30” has 5 out of 14 samples, with 2 yes’es and 3 no’s.

3 : Calculate Gain of Each Attribute (one by one)

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

$$Gain(income) = 0.029$$

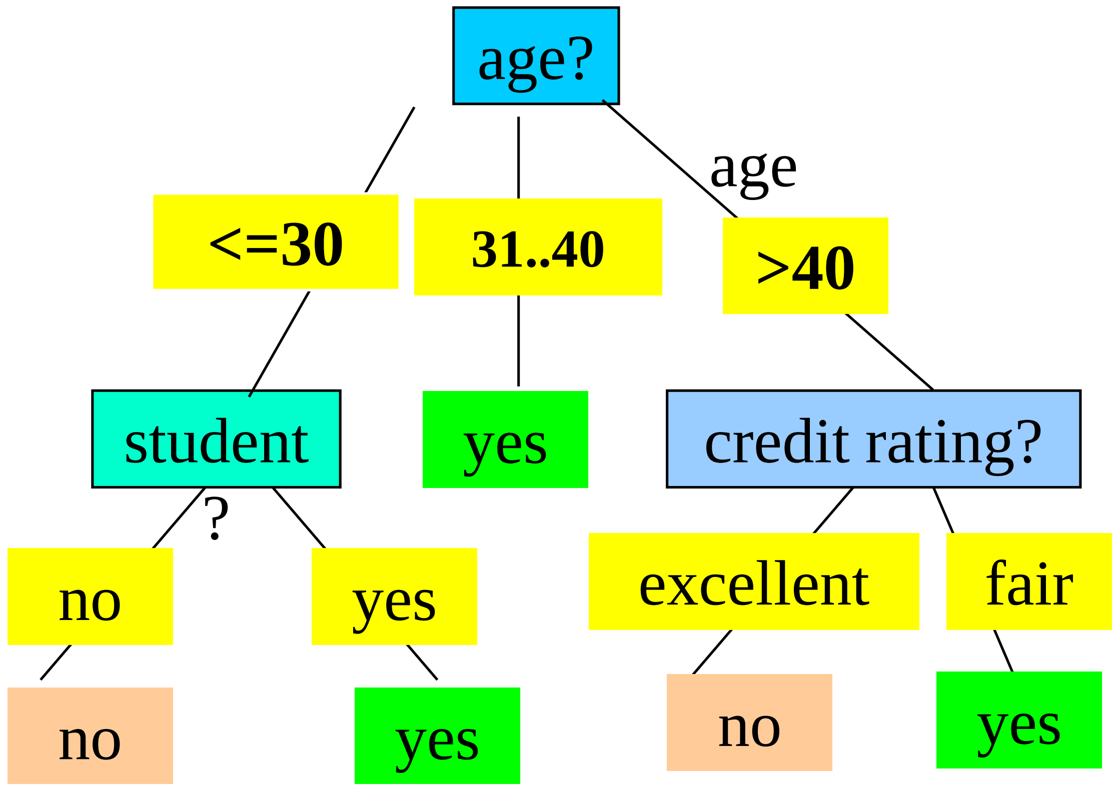
$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

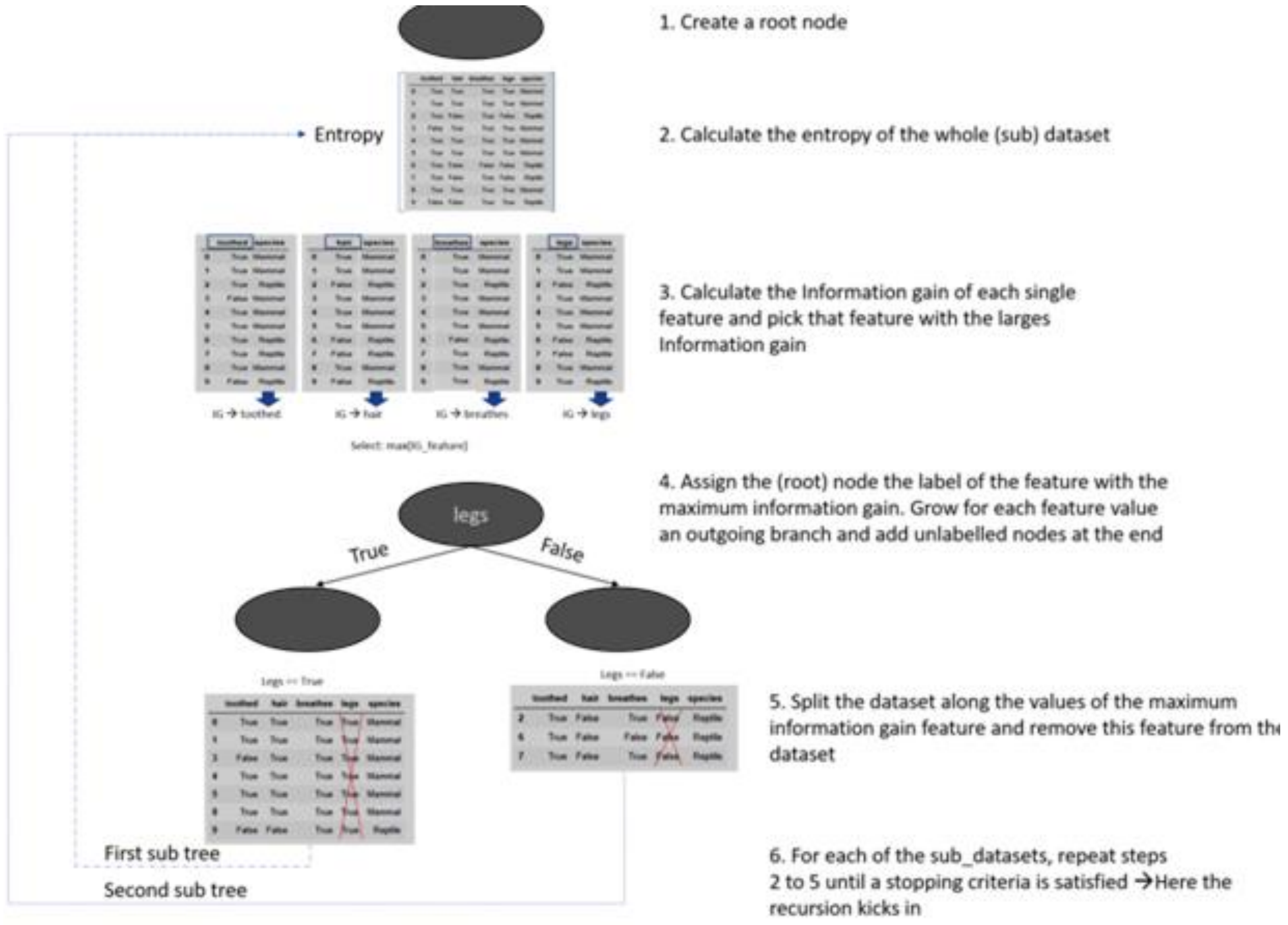
4: Select Attribute with max gain as a root attribute

Gain(age)> any other gain. so age is selected as root node

Final DT of the buys_computer dataset



Example : Usage of Information Gain and Entropy in DT Creation



1. Create a root node

2. Calculate the entropy of the whole (sub) dataset

3. Calculate the Information gain of each single feature and pick that feature with the larges Information gain

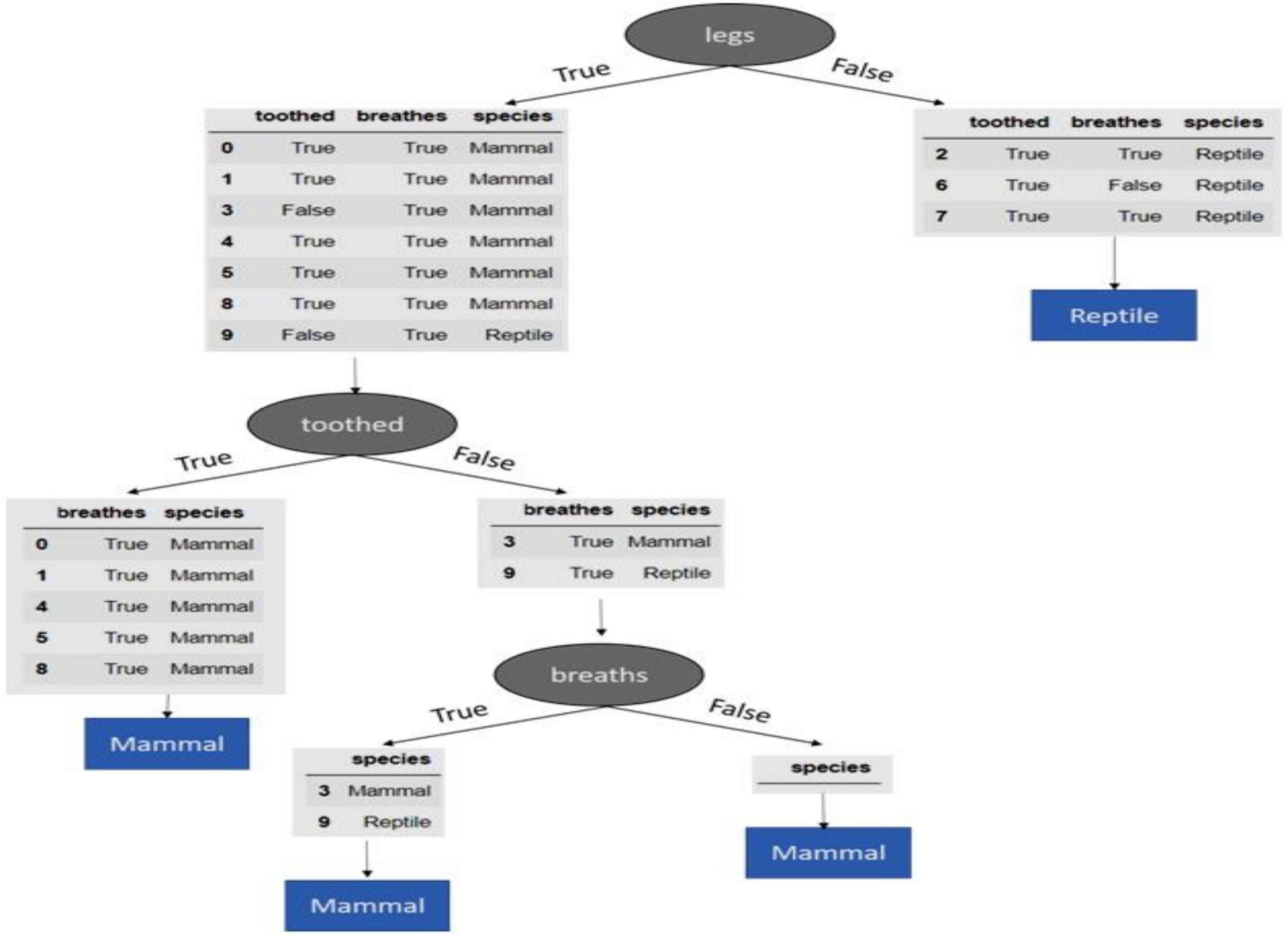
4. Assign the (root) node the label of the feature with the maximum information gain. Grow for each feature value an outgoing branch and add unlabelled nodes at the end

5. Split the dataset along the values of the maximum information gain feature and remove this feature from th dataset

6. For each of the sub_datasets, repeat steps 2 to 5 until a stopping criteria is satisfied →Here the recursion kicks in

	toothed	hair	breathes	legs	species
0	True	True	True	True	Mammal
1	True	True	True	True	Mammal
2	True	False	True	False	Reptile
3	False	True	True	True	Mammal
4	True	True	True	True	Mammal
5	True	True	True	True	Mammal
6	True	False	False	False	Reptile
7	True	False	True	False	Reptile
8	True	True	True	True	Mammal
9	False	False	True	True	Reptile

DT for Given Example



	toothed	breathes	species
0	True	True	Mammal
1	True	True	Mammal
3	False	True	Mammal
4	True	True	Mammal
5	True	True	Mammal
8	True	True	Mammal
9	False	True	Reptile

	toothed	breathes	species
2	True	True	Reptile
6	True	False	Reptile
7	True	True	Reptile

	breathes	species
0	True	Mammal
1	True	Mammal
4	True	Mammal
5	True	Mammal
8	True	Mammal

	breathes	species
3	True	Mammal
9	True	Reptile

	species
3	Mammal
9	Reptile

species

Summary

- Classification is a form of data analysis that extracts models describing important data classes.
- Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods.

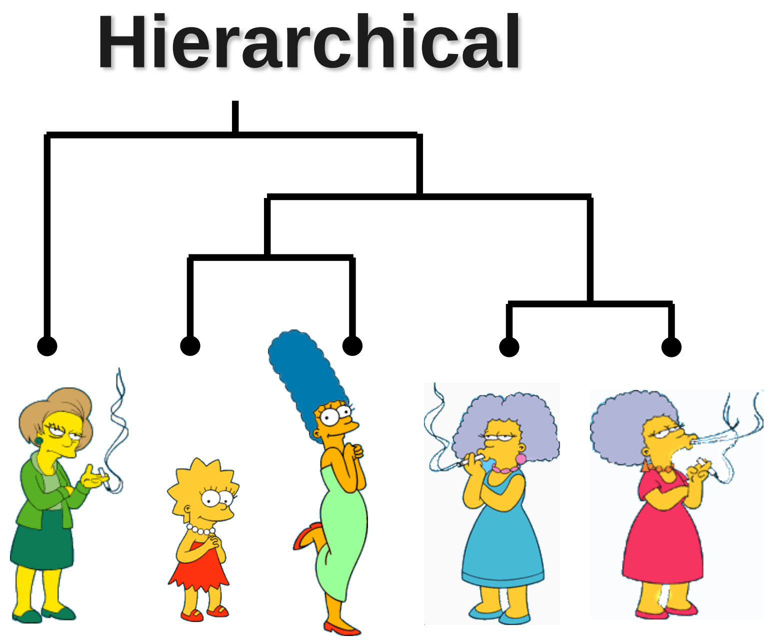
Unit IV Clustering

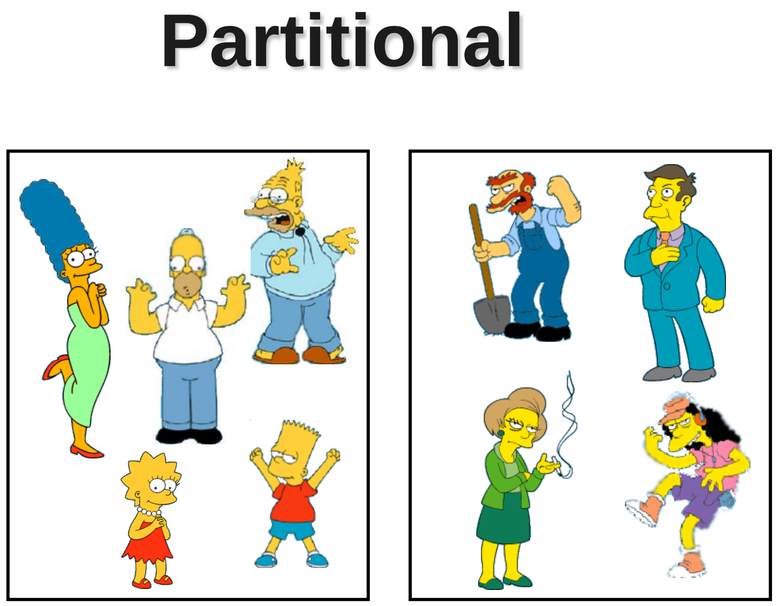
Clustering

- A process of organizing objects into groups such that data points in the same groups are similar to the data points in the same group.
- A cluster is a collection of objects where these objects are similar and dissimilar to the other cluster.

Two types of Clustering

- Partitional algorithms: Construct various partitions and then evaluate them by some criterion (we will see an example called BIRCH)
- Hierarchical algorithms: Create a hierarchical decomposition of the set of objects using some criterion





Types of Clustering

1. **Hierarchical algorithms**: these find successive clusters

   using previously established clusters.

   Agglomerative ("bottom-up"): Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters.

   Divisive ("top-down"): Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

1. **Partitional clustering:** Partitional algorithms determine all lusters at once. They include:

   - K-means and derivatives
   - Fuzzy *c*-means clustering
   - QT clustering algorithm

Dr. Vishwanath Karad MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Common Distance measures:

Distance measure will determine how the *similarity* of two elements is calculated and it will influence the shape of the clusters.

They include:

1. The Euclidean distance

$$d(x,y) = \sqrt[2]{\sum_{i=1}^{p} |x_i - y_i|^2}$$

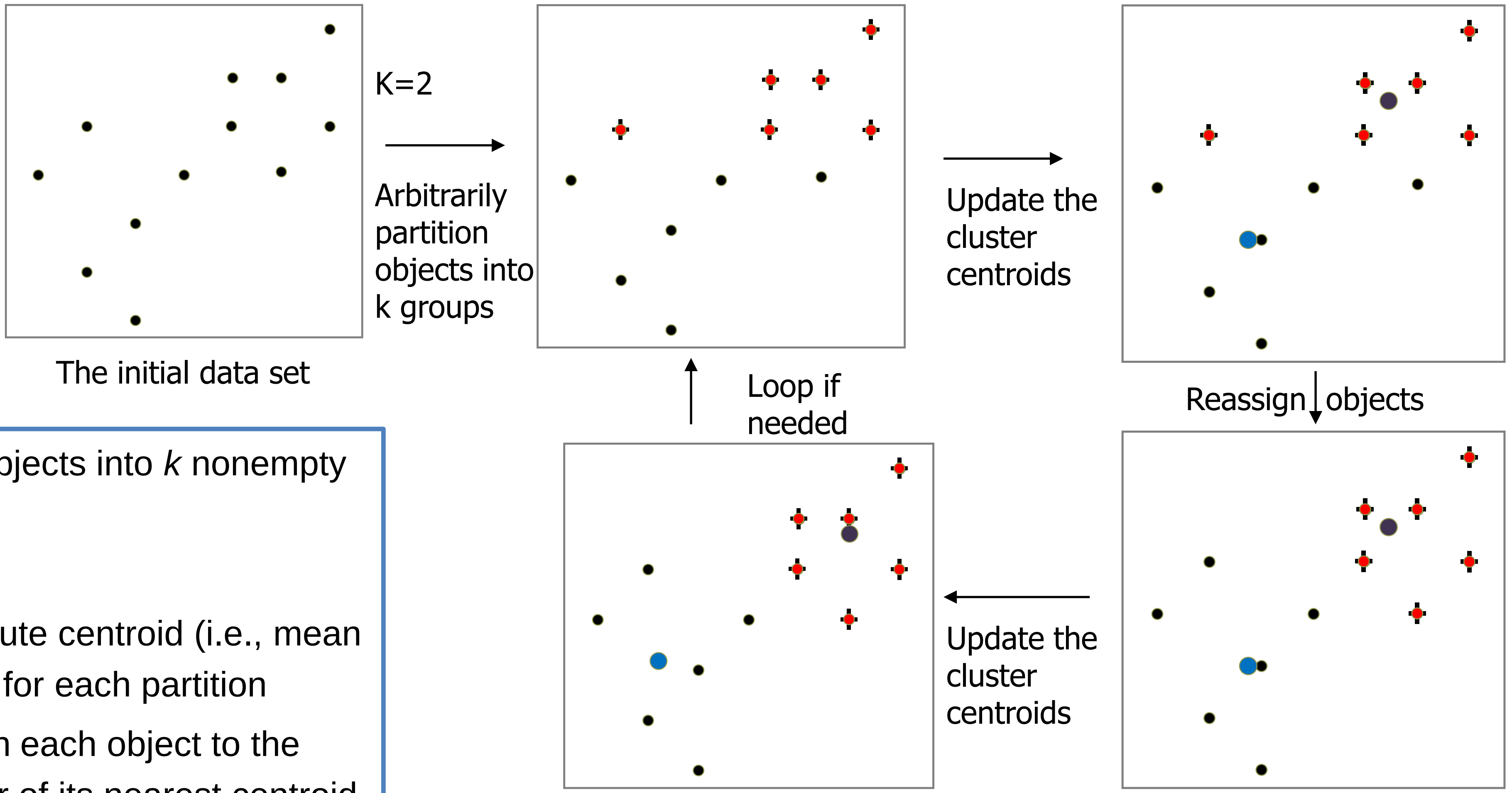
2. The Manhattan distance

$$d(x,y) = \sum_{i=1}^{p} |x_i - y_i|$$

Basic Cluster Analysis Methods

- Cluster Analysis: Basic Concepts
  - Group data so that object similarity is high within clusters but low across clusters
- Partitioning Methods
  - K-means and k-medoids algorithms and their refinements
- Hierarchical Methods
  - Agglomerative and divisive method, Birch, Cameleon
- Density-Based Methods
  - DBScan, Optics and DenCLu
- Grid-Based Methods
  - STING and CLIQUE (subspace clustering)
- Evaluation of Clustering
  - Assess clustering tendency, determine # of clusters, and measure clustering quality

K-Means Clustering



K=2
Arbitrarily partition objects into k groups
The initial data set
Update the cluster centroids
Reassign objects
Update the cluster centroids
Loop if needed

- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

Cluster the following eight points (with (x, y) representing locations) into three clusters:
A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)

Initial cluster centers are: A1(2, 10), A4(5, 8) and A7(1, 2).
The distance function between two points a = (x1, y1) and b = (x2, y2) is defined as-

$$P(a, b) = |x2 – x1| + |y2 – y1|$$

Given Points	**Distance from center (2, 10) of Cluster-01**	**Distance from center (5, 8) of Cluster-02**	**Distance from center (1, 2) of Cluster-03**	**Point belongs to Cluster**
A1(2, 10)	0	5	9	C1
A2(2, 5)	5	6	4	C3
A3(8, 4)	12	7	9	C2
A4(5, 8)	5	0	10	C2
A5(7, 5)	10	5	9	C2
A6(6, 4)	10	5	7	C2
A7(1, 2)	9	10	0	C3
A8(4, 9)	3	2	10	C2

Clusters at 1st Iteration

Cluster-01:

First cluster contains points-
- A1(2, 10)

Cluster-02:

Second cluster contains points-
- A3(8, 4)
- A4(5, 8)
- A5(7, 5)
- A6(6, 4)
- A8(4, 9)

Cluster-03:

Third cluster contains points-
- A2(2, 5)
- A7(1, 2)

Recompute Centroid

For Cluster-01:

- We have only one point A1(2, 10) in Cluster-01.
- So, cluster center remains the same.

For Cluster-02:

Center of Cluster-02
= ((8 + 5 + 7 + 6 + 4)/5, (4 + 8 + 5 + 4 + 9)/5)
= (6, 6)

For Cluster-03:

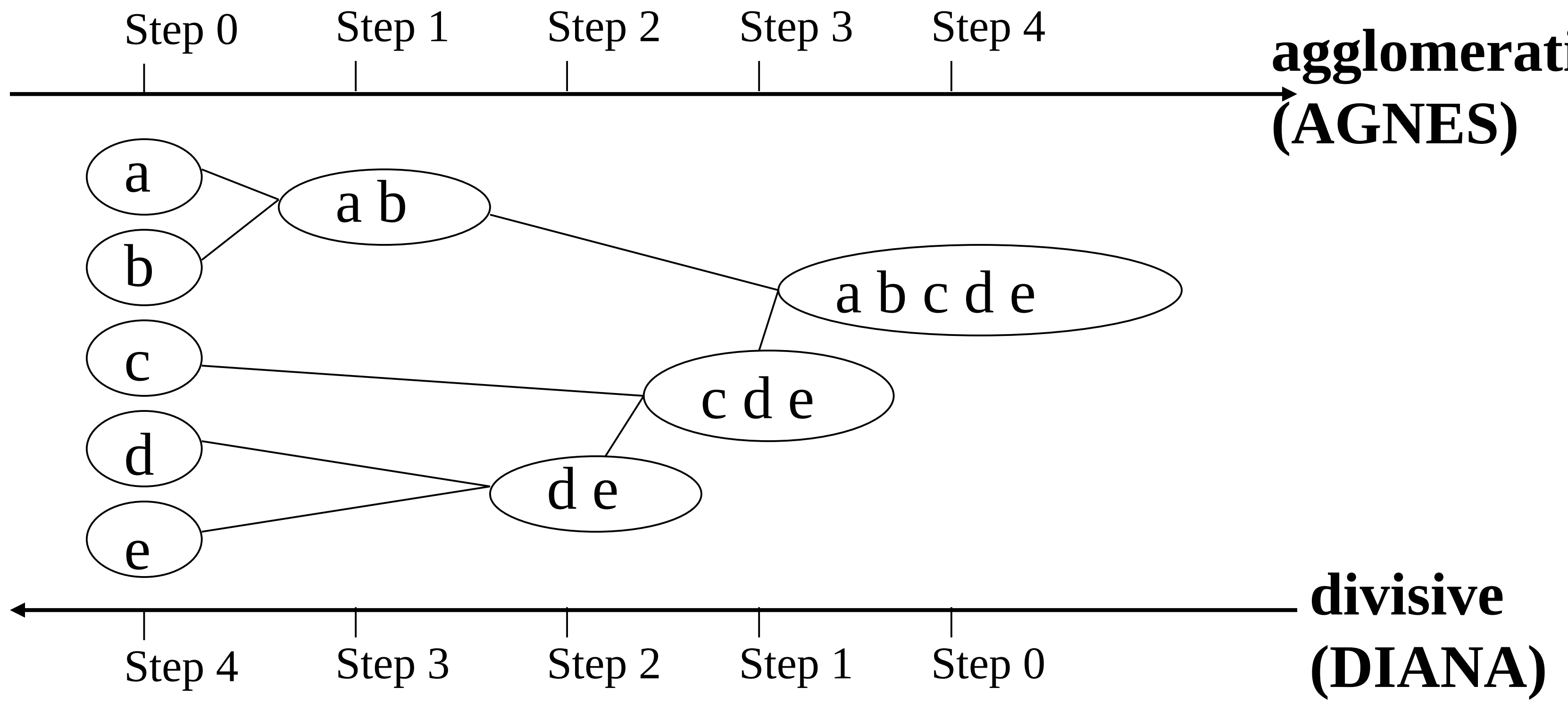
Center of Cluster-03
= ((2 + 1)/2, (5 + 2)/2)
= (1.5, 3.5)

Iteration 2

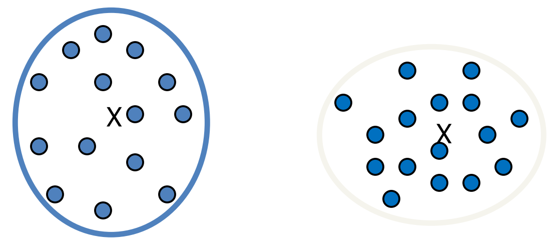
iven Points	Distance from center (2, 10) of Cluster-01	Distance from center (6, 6) of Cluster-02	Distance from center (1.5, 3.5) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	8	7	C1
A2(2, 5)	5	5	2	C3
A3(8, 4)	12	4	7	C2
A4(5, 8)	5	3	8	C2
A5(7, 5)	10	2	7	C2
A6(6, 4)	10	2	5	C2
A7(1, 2)	9	9	2	C3
A8(4, 9)	3	5	8	C1

Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition



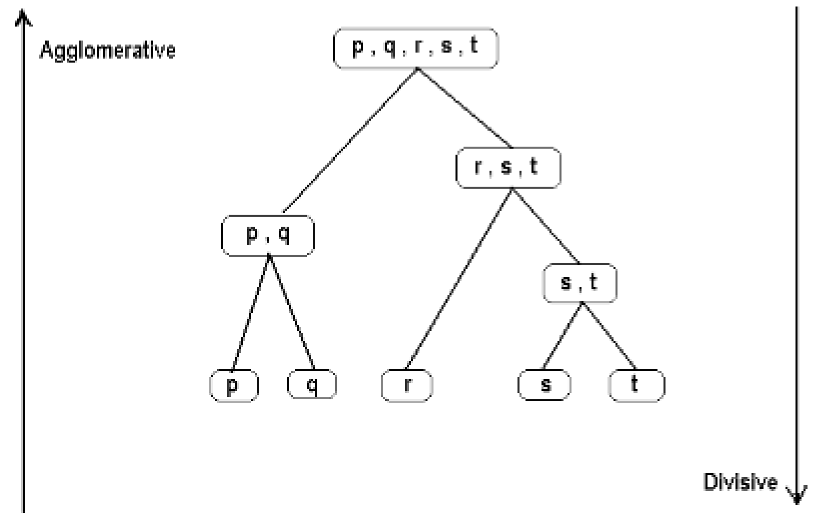
Distance between Clusters



- Single link: smallest distance between an element in one cluster and an element in the other, i.e., $\text{dist}(K_i, K_j) = \min(t_{ip}, t_{jq})$
- Complete link: largest distance between an element in one cluster and an element in the other, i.e., $\text{dist}(K_i, K_j) = \max(t_{ip}, t_{jq})$
- Average: avg distance between an element in one cluster and an element in the other, i.e., $\text{dist}(K_i, K_j) = \text{avg}(t_{ip}, t_{jq})$
- Centroid: distance between the centroids of two clusters, i.e., $\text{dist}(K_i, K_j) = \text{dist}(C_i, C_j)$
- Medoid: distance between the medoids of two clusters, i.e., $\text{dist}(K_i, K_j) = \text{dist}(M_i, M_j)$
  - Medoid: a chosen, centrally located object in the cluster

Clustering : Hierarchical

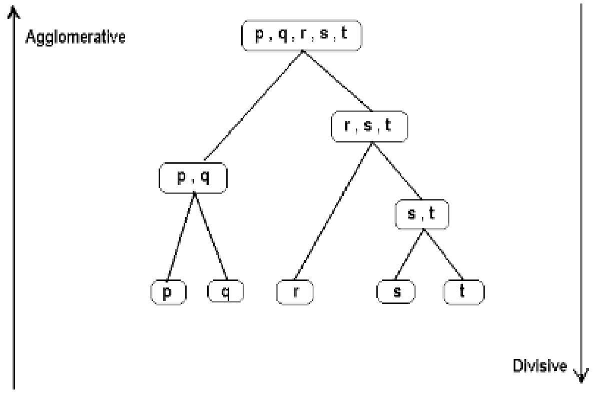
- Clustering algorithms that build nested clusters by merging or splitting them successively.
- This hierarchy of clusters is represented as a binary tree (or dendrogram).
- The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.



Hierarchical clustering approaches:

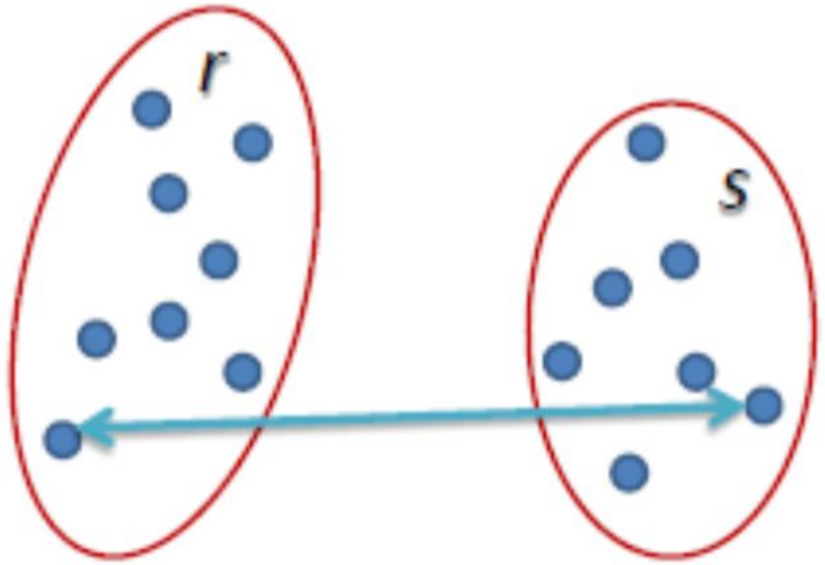
- Agglomerative: "bottom up" approach; each sample starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- Divisive: "top down" approach; all samples start in one cluster, and splits are performed recursively as one moves down the hierarchy

The Agglomerative Clustering



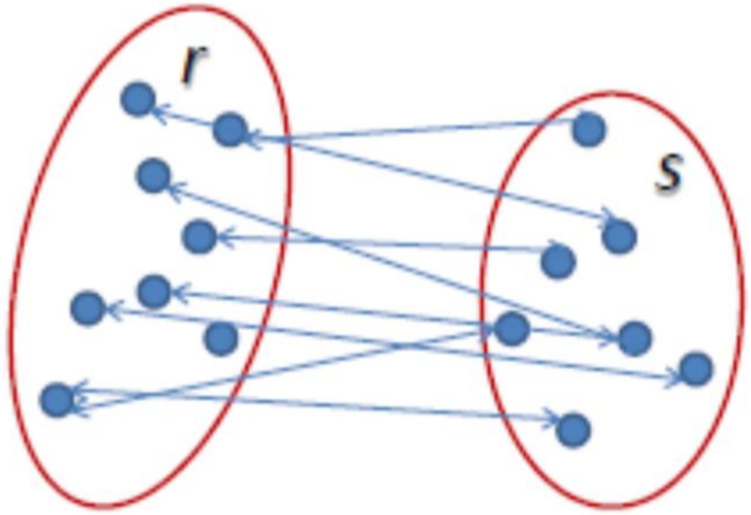
- object perform the bottom-up approach: it recursively merges the pair of clusters that minimally increases a given linkage distance
- The linkage parameter determines the metric used for the merge strategy (default value is 'ward')
- ward: minimizes the sum of squared differences within all clusters.
- complete: minimizes the maximum distance between samples of pairs of clusters
- average: minimizes the average of the distances between all samples of pairs of clusters.

Complete Linkage: The distance between two clusters is the longest distance between two points in each cluster



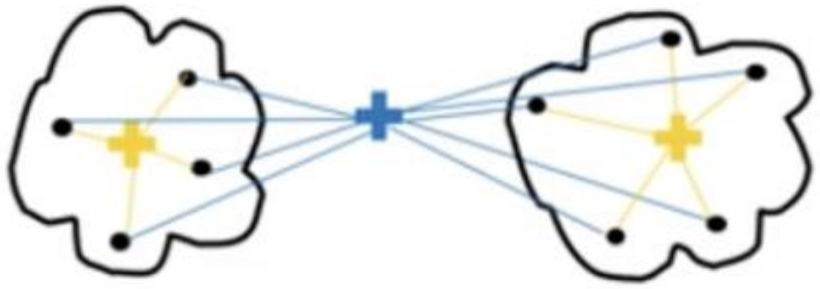
$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

Average Linkage: The distance between clusters is the average distance between each point in one cluster to every point in other cluster



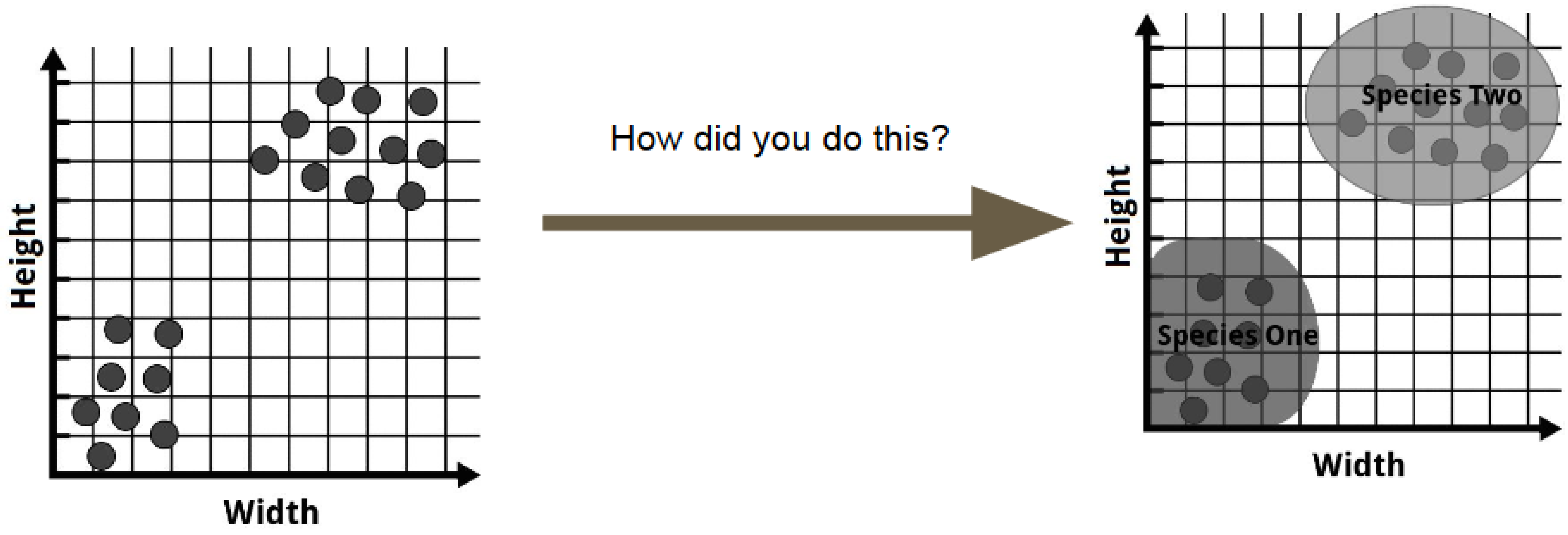
$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

Ward Linkage: The distance between clusters is the sum of squared differences within all clusters



DBSCAN

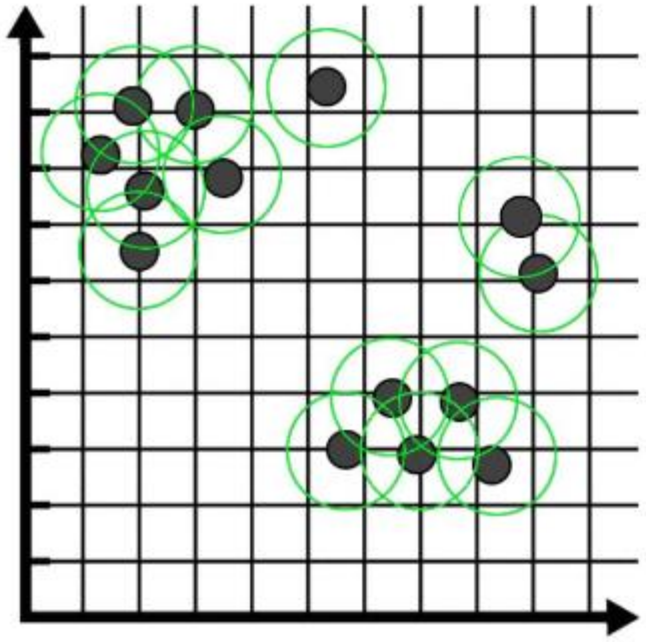
What was your thought process here?



1. You made the guess by looking at **how close these data points were to one another**.
2. Looking at the density (or closeness) of our observations is a common way to discover clusters in a dataset

DBSCAN: First Step

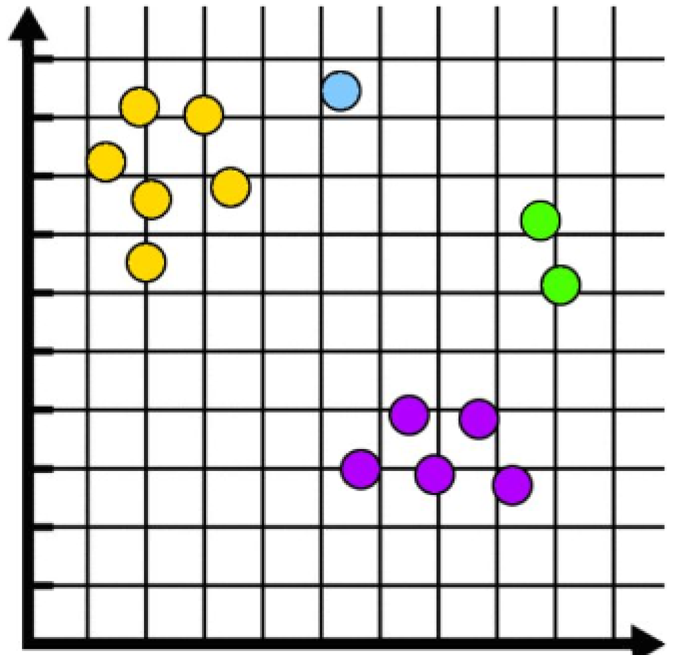
- Step 1: DBSCAN starts by identifying the neighboring observations of each observation within (a hyperparameter) some radius



Identifying radius of *some length* for each vector

DBSCAN: Second step

- Step 2: Any data point that is within the data point of radius of another data point are in the same cluster

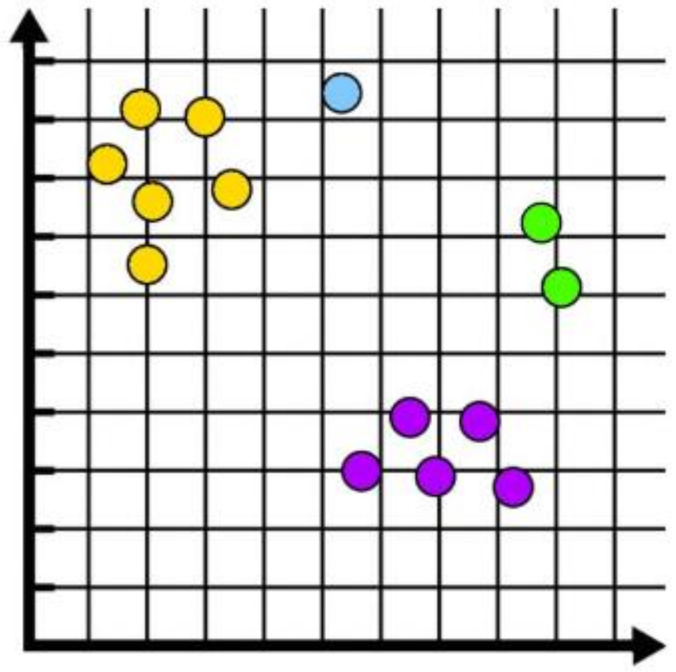


Observations that are within another observation radius are assigned to the same cluster.

Four clusters were found in total.

Contd..

- three observations which are noise and we end up creating two clusters entirely from these bad observations themselves.
- What should we do to avoid this?


Blue cluster and green cluster are formed from noise!!

DBSCAN Algorithm

- Step 1: Compute the neighborhood of all data points
- Step 2: Group the data points that have at least some specific number of points in their cluster.
- Edge points should be taken into consideration in Step 2

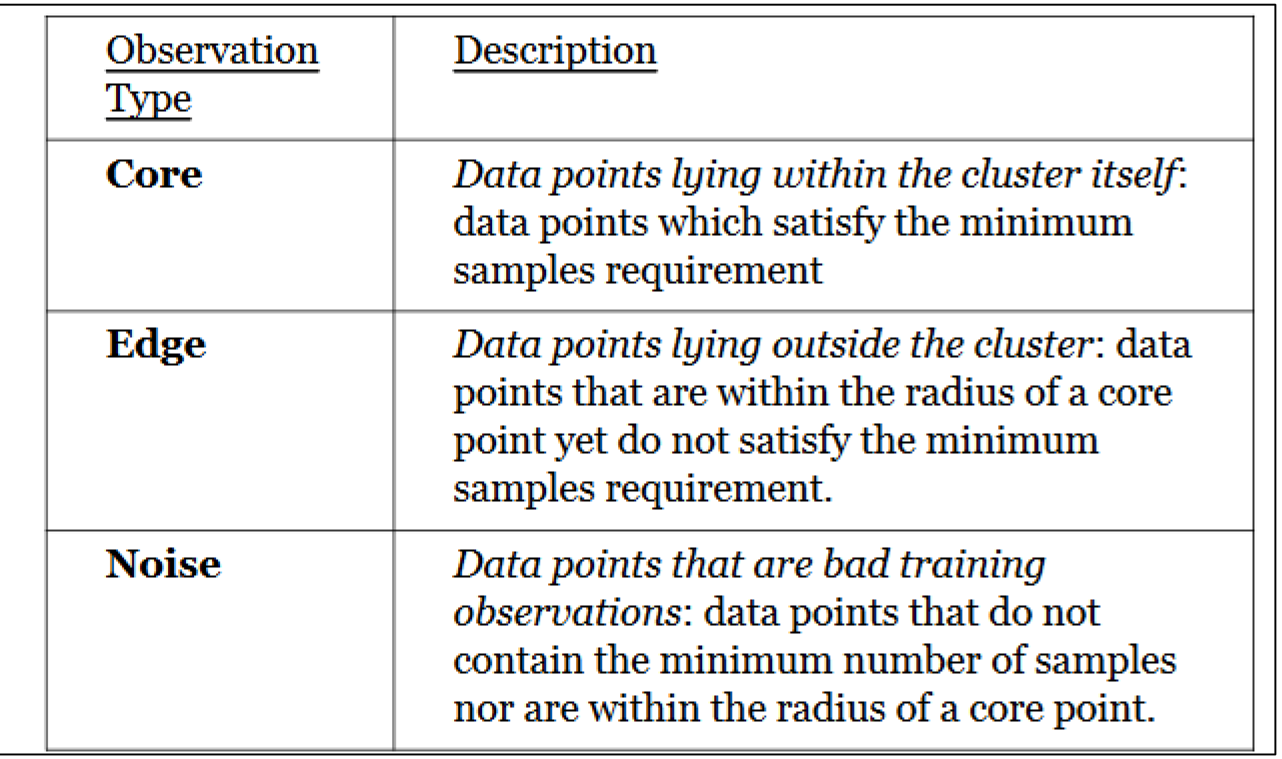
DBSCAN :Hyperparameter

Radius, *How far apart should observations be, to be considered in the same cluster?*

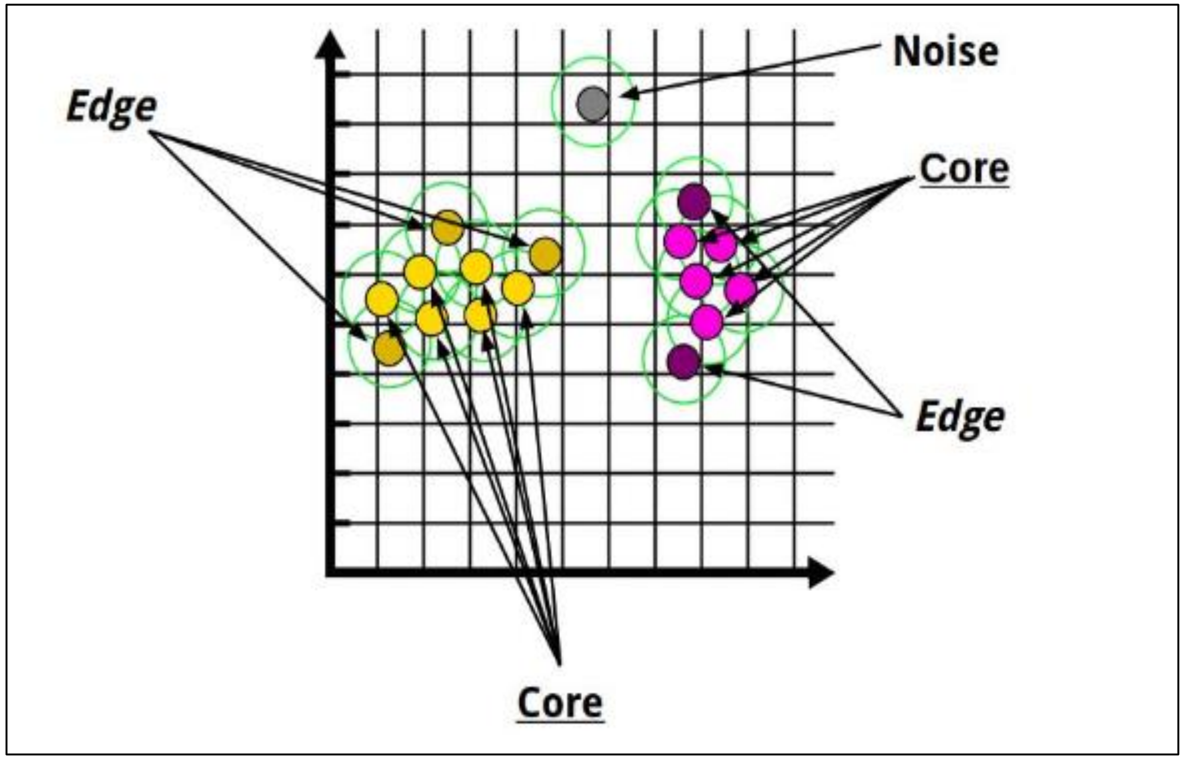
Minimum # of Samples, *How many observations should be in the radius of a data point?*

We have to manually choose these parameters using domain-specific knowledge related to the problem at hand

Type of Data Points in DBSCAN



Observation Type	Description
Core	*Data points lying within the cluster itself*: data points which satisfy the minimum samples requirement
Edge	*Data points lying outside the cluster*: data points that are within the radius of a core point yet do not satisfy the minimum samples requirement.
Noise	*Data points that are bad training observations*: data points that do not contain the minimum number of samples nor are within the radius of a core point.



Unit III

Ensemble classifiers, Bagging and Boosting, **Training** versus Testing Samples, Positive and Negative Class, **Confusion Matrix for Model Evaluation,** Model Selection, Implementation and Evaluation using Scikit-learn library

Confusion Matrix : Classifier Accuracy Measure

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

Example of Confusion Matrix:

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

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

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

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All

- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?
- Perfect score is 1.0

$$recall = \frac{TP}{TP + FN}$$

- ***F* measure (*F_1* or *F*-score)**: harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

- *Precision* = 90/230 = 39.13%   *Recall* = 90/300 = 30.00%

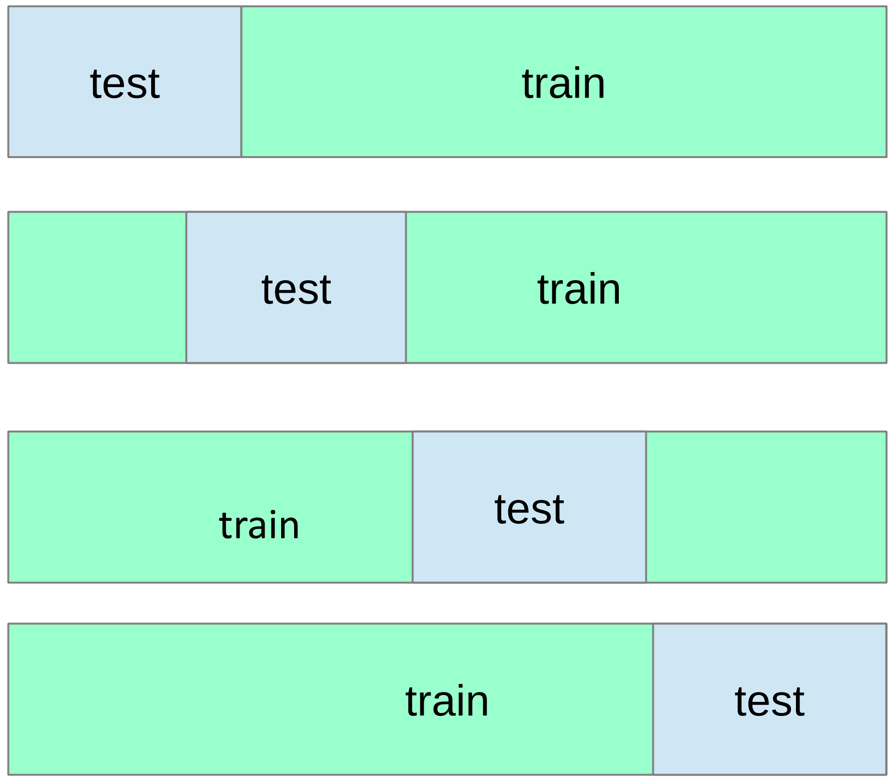
Actual class\Predicted class	C_1	¬ C_1
C_1	**True Positives (TP)**	**False Negatives (FN)**
¬ C_1	**False Positives (FP)**	**True Negatives (TN)**

Cross Validation

- To handle the Problem of Overfitting Cross Validation is required
- Data considered as train, test set may prove excellent accuracy with some validation sample(which cases already taken in to consideration while training), but poor accuracy with some other validation set
- Solution : Use Variety of samples for Training and Testing
- Cross Validation:
  - The data is divided in multiple ways, and then statistical methods are used to find the average accuracy.
  - This can be used to compare efficiency of different machine learning algorithms.

Cross Validation

- Data is divided into multiple chunks of test- train splits
- The average accuracy is then used.



Ensemble: Together

- Ensembles can give you a boost in accuracy on your dataset

- The goal of **ensemble methods** is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

Families of Ensemble Methods

- **Averaging methods**, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.[aggregate their individual predictions to form a final prediction]

  Examples: Bagging methods, Forests of randomized trees, …

- By contrast, in **boosting methods**, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.

- **Examples:** AdaBoost, Gradient Tree Boosting, …

Bagging

Bagging, is an Ensemble Learning technique that has two parts – i) Bootstrap and ii) Aggregation.

Bootstrap

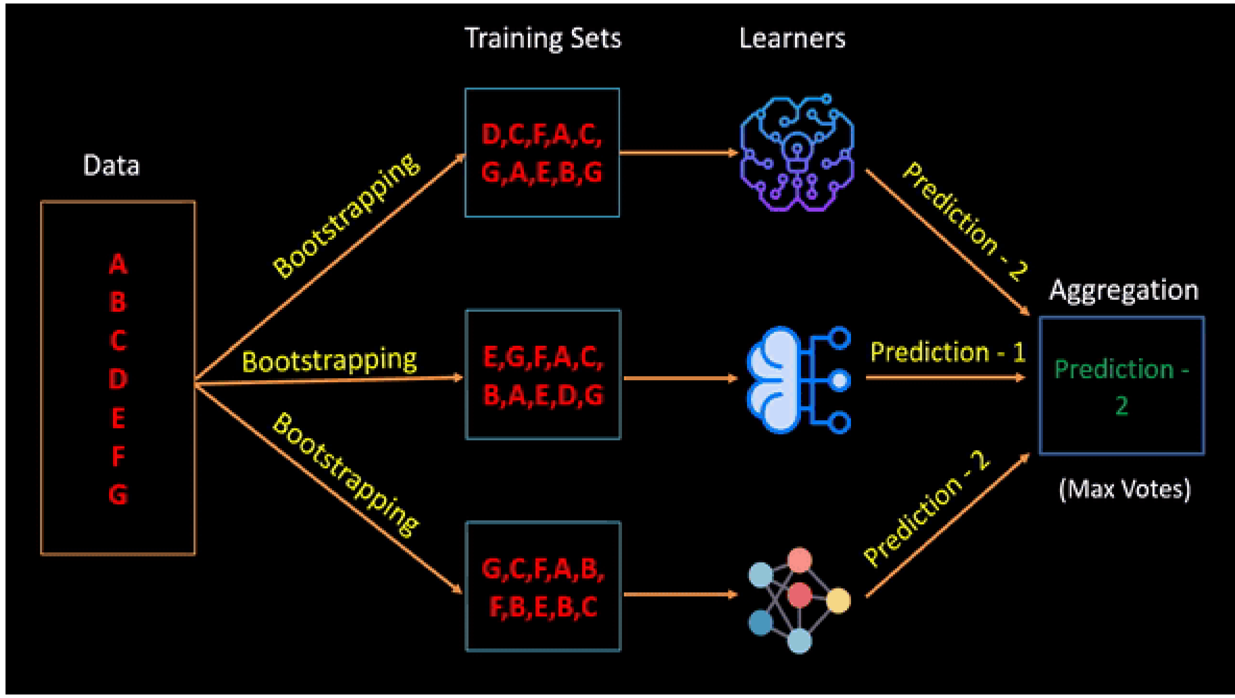
Bootstrap is a technique of random sampling of data with replacement. In the ensemble learning context, we prepare multiple training sets by this bootstrap method.

Just to give an example, if we have a data set (a,b,c,d,e,f,g,h,i,j) then we can get following training set using bootstrap – (a,f,d,j,h,c) (b,g,a,i,c,f) (i,d,c,e,a,d) (b,h,g,h,a,b)

Notice, that the same data can appear multiple times in the training set, since we are doing random sampling with replacement.

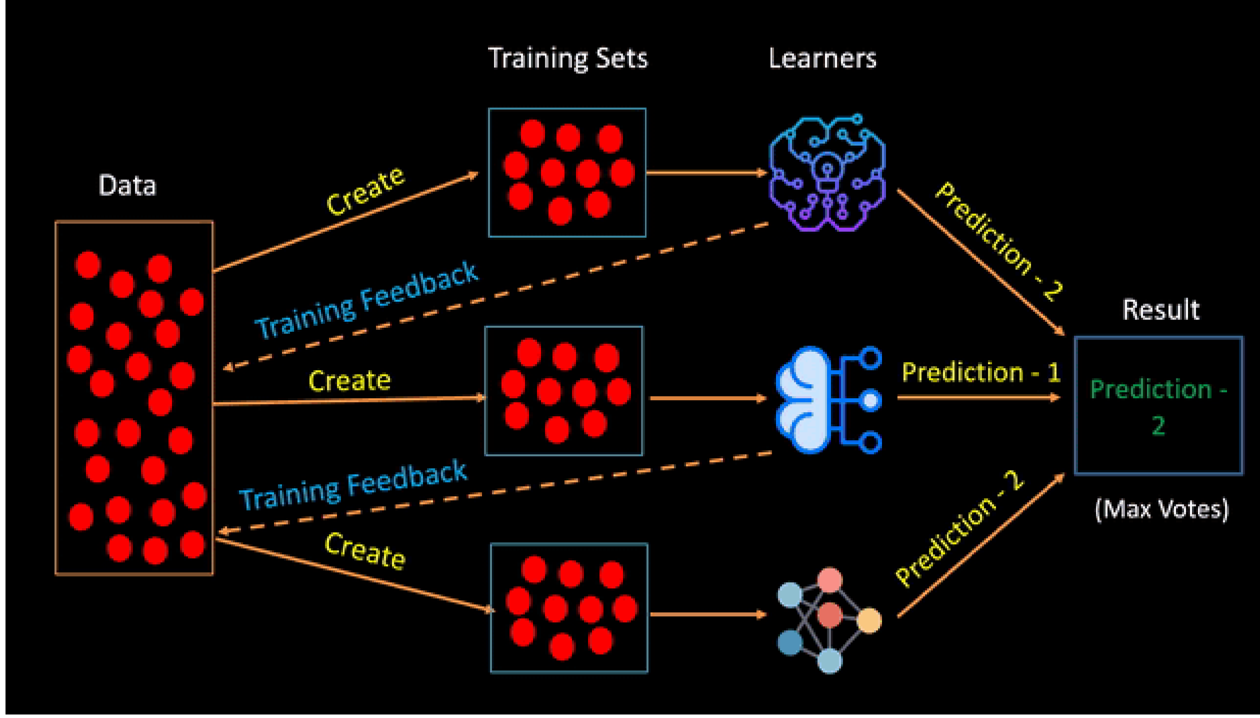
Aggregation

The different learners train on these various training sets that are obtained by bootstrapping method. The hypothesis of these individually trained learners are then aggregated to form a single consensus output.



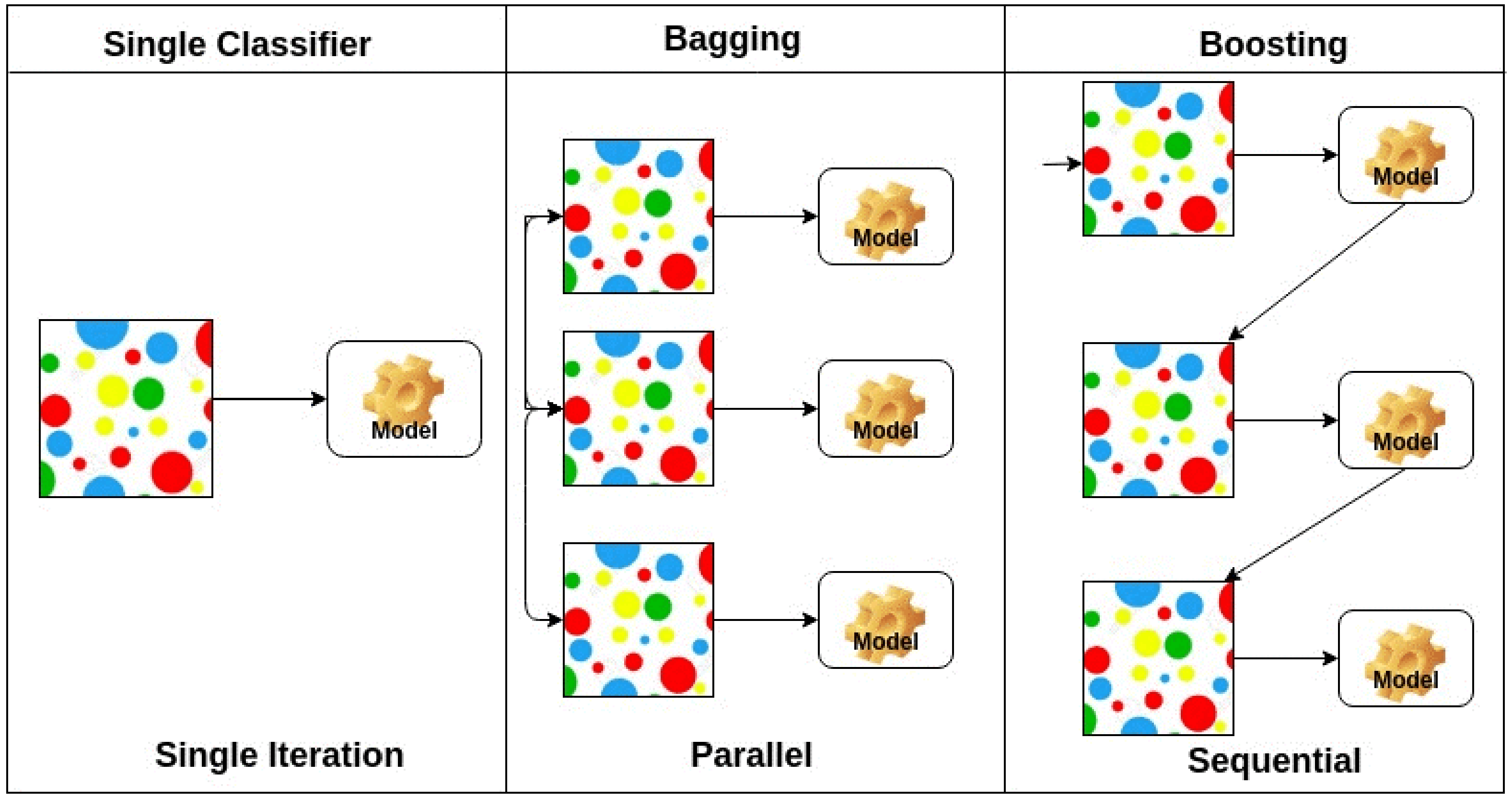
Boosting Algorithms

- Boosting ensemble algorithms creates a sequence of models that attempt to correct the mistakes of the models before them in the sequence.
- Once created, the models make predictions which may be weighted by their demonstrated accuracy and the results are combined to create a final output prediction.
- Most common boosting ensemble machine learning algorithms is:Adaboost



Adaboost

- Initially, Adaboost selects a training subset randomly.
- It iteratively trains the AdaBoost machine learning model by selecting the training set based on the accurate prediction of the last training.
- It assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification.
- Also, It assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight.
- This process iterate until the complete training data fits without any error or until reached to the specified maximum number of estimators.
- To classify, perform a "vote" across all of the learning algorithms you built.


Single Classifier
Bagging
Boosting
Model
Model
Model
Model
Model
Model
Model
Single Iteration
Parallel
Sequential

Thank You !!