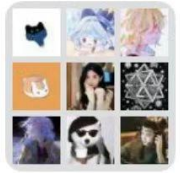
When AI meets life sciences
当人工智能遇上生命科学

2025 清华大学 AI创新实践课程

张强锋，清华大学

Sep 22, 2025



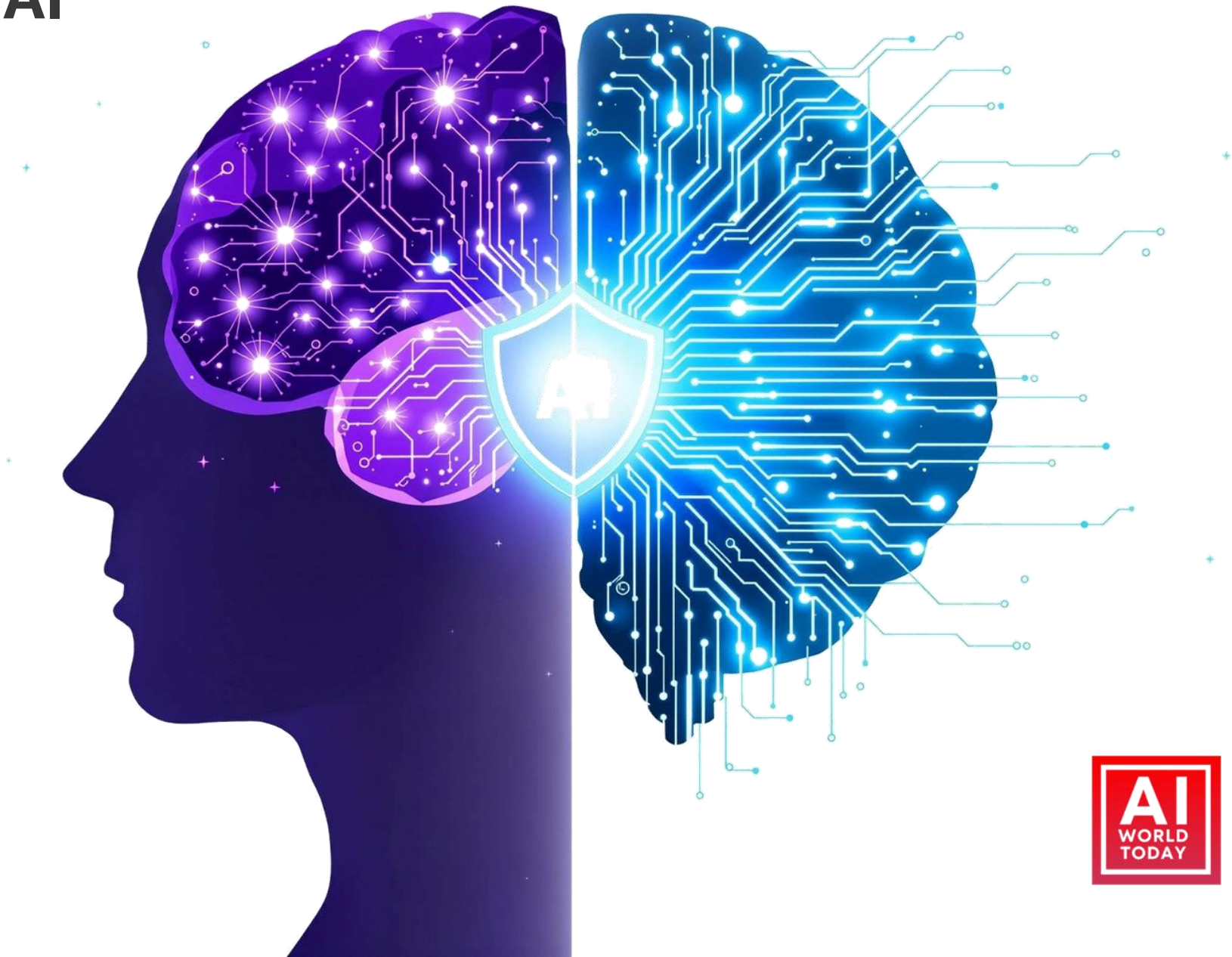
群聊：人工智能与生命科学

该二维码7天内(9月29日前)有效，重新进入将更新

Outline

- AI in a nutshell
- What is machine learning?
- Categories of machine learning and representative algorithms
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning
- Feature engineering
- Practical methodology
- Practice

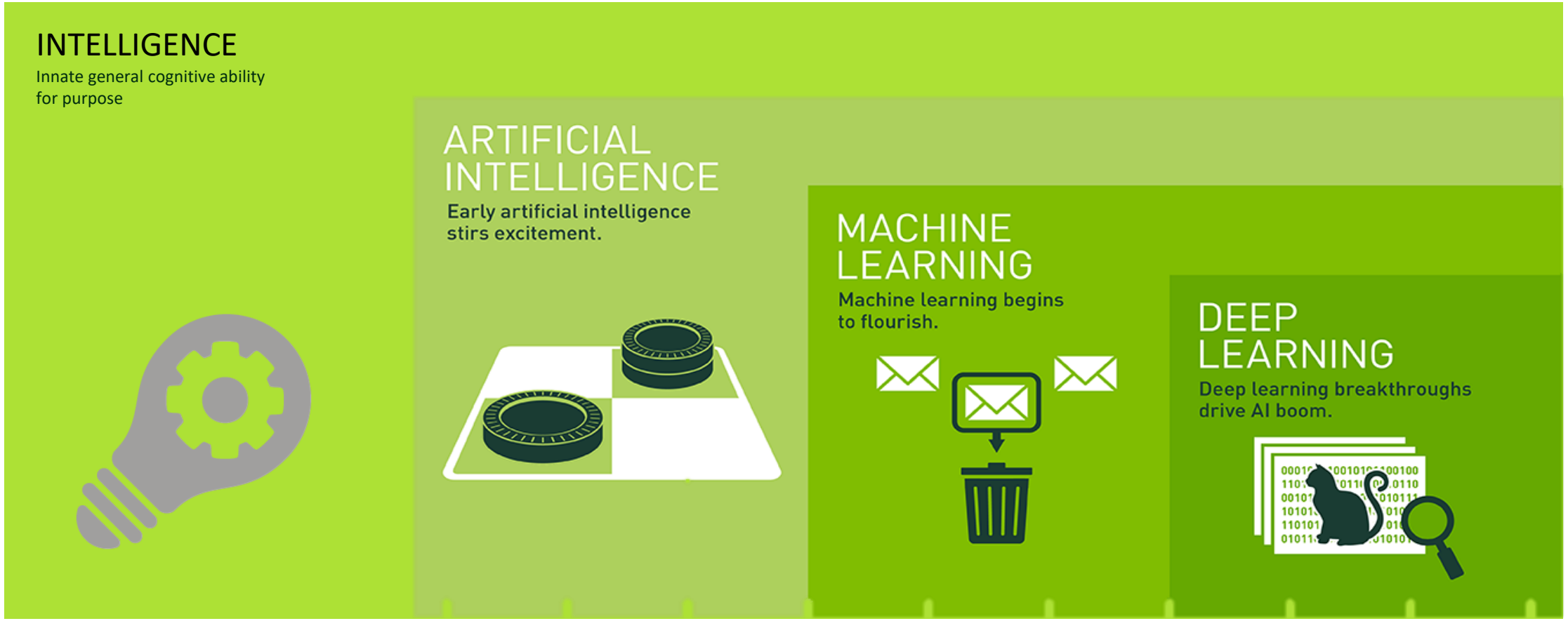
What is AI



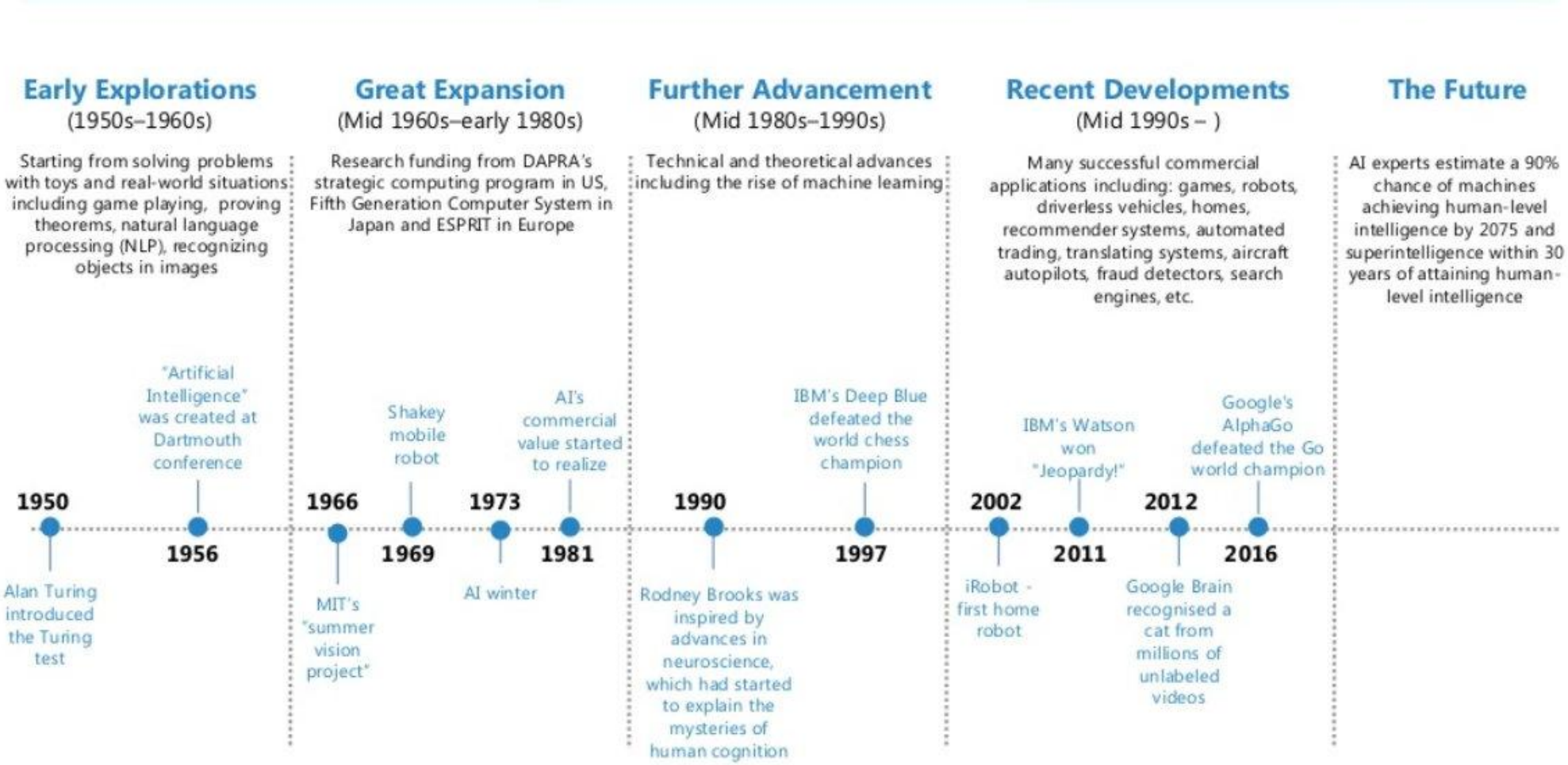
What is intelligence?

- Innate general cognitive ability. – Cyril Burt
  - perception, spatial cognition, time cognition, attention, memory, tool use, reasoning and "computation", problem solving and decision making, judgment and evaluation, comprehension, language, mind, abstraction, concepts, consciousness and free will
- The aggregate or global capacity of the individual to act purposefully, to think rationally, and to deal effectively with his environment. – David Wechsler
- Turing test for human-level intelligence?

Artificial intelligence & machine learning & deep learning



A brief history of AI development



Source: Artificial Intelligence by Giorgio Fumera from University of Cagliari | AI: 15 Key Moments in the Story of Artificial Intelligence by BBC | Future Progress in Artificial Intelligence: A Survey of Expert Opinion

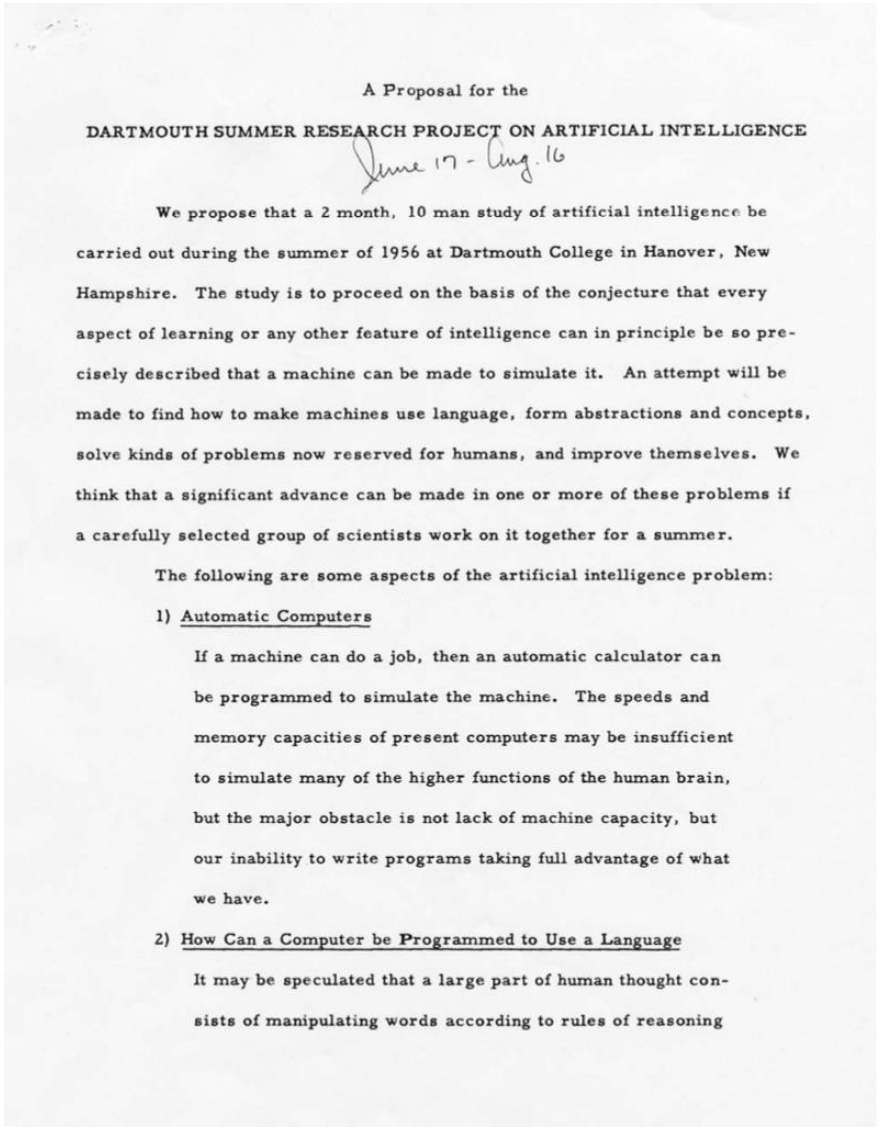
vertex

Alan Turing and Turing's test

Alan Mathison Turing
(23 June 1912 – 7 June 1954)



The Birth of AI, and The 1956 Dartmouth Conference



A Proposal for the

DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE

June 17 - Aug. 16

We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

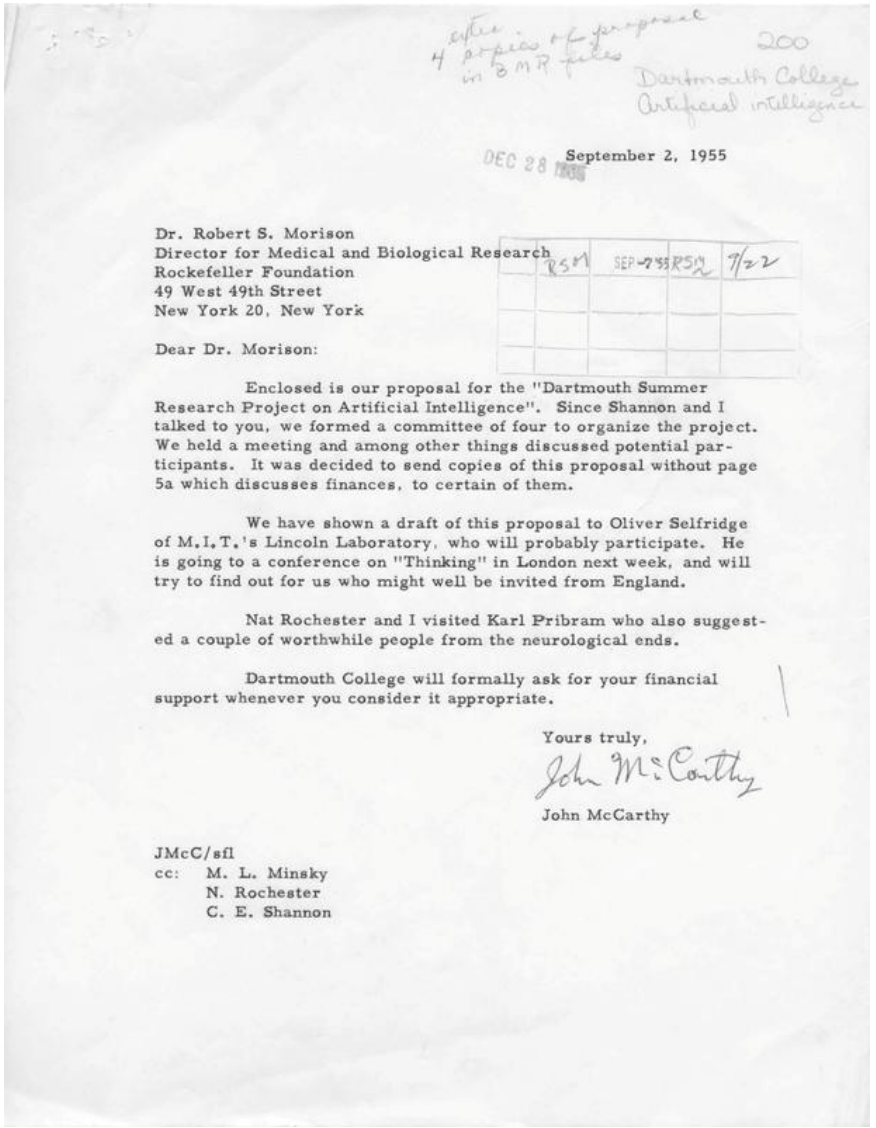
The following are some aspects of the artificial intelligence problem:

1) Automatic Computers

If a machine can do a job, then an automatic calculator can be programmed to simulate the machine. The speeds and memory capacities of present computers may be insufficient to simulate many of the higher functions of the human brain, but the major obstacle is not lack of machine capacity, but our inability to write programs taking full advantage of what we have.

2) How Can a Computer be Programmed to Use a Language

It may be speculated that a large part of human thought consists of manipulating words according to rules of reasoning



4 copies of proposal in 3 MR files

200

Dartmouth College
Artificial Intelligence

DEC 28

September 2, 1955

Dr. Robert S. Morison
Director for Medical and Biological Research
Rockefeller Foundation
49 West 49th Street
New York 20, New York

RSM SEP 7 '55 RSM 7/22

Dear Dr. Morison:

Enclosed is our proposal for the "Dartmouth Summer Research Project on Artificial Intelligence". Since Shannon and I talked to you, we formed a committee of four to organize the project. We held a meeting and among other things discussed potential participants. It was decided to send copies of this proposal without page 5a which discusses finances, to certain of them.

We have shown a draft of this proposal to Oliver Selfridge of M.I.T.'s Lincoln Laboratory, who will probably participate. He is going to a conference on "Thinking" in London next week, and will try to find out for us who might well be invited from England.

Nat Rochester and I visited Karl Pribram who also suggested a couple of worthwhile people from the neurological ends.

Dartmouth College will formally ask for your financial support whenever you consider it appropriate.

Yours truly,

John McCarthy

John McCarthy

JMcC/sfl
cc: M. L. Minsky
N. Rochester
C. E. Shannon

The founding fathers of AI



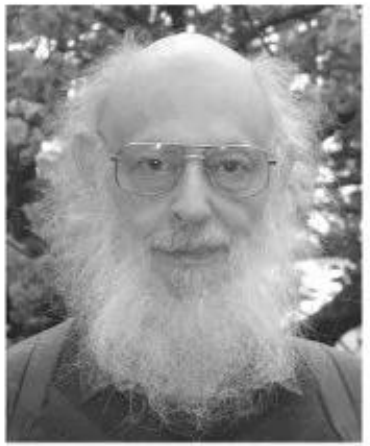
John MacCarthy



Marvin Minsky



Claude Shannon



Ray Solomonoff



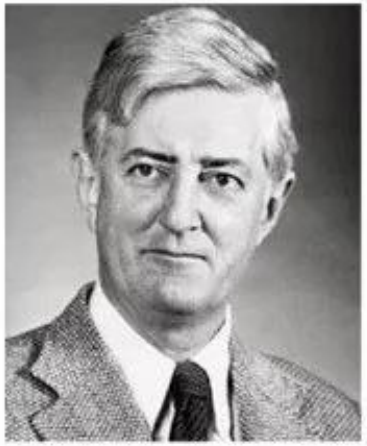
Alan Newell



Herbert Simon



Arthur Samuel



Oliver Selfridge



Nathaniel Rochester



Trenchard More

Major schools of AI

Symbolism

Symbolism, also known as **Logicism** or the **Psychological School**, is the "classical school" in the field of AI and dominated the period from the 1950s to the 1980s.

Core idea: Intelligence is based on **knowledge** represented by **symbols**. Intelligent acts like reasoning involve manipulating these symbols logically. It claims that with a complete symbolic system and rules, human rational thinking can be replicated.

Logical reasoning, knowledge graph, search algorithms

Expert systems, theorem proving, game

Connectionism

Connectionism experienced a revival after the 1980s and, with the breakthroughs in **deep learning** in the 2010s, has become the dominant mainstream approach in AI today. It is also known as the **Bionic School** or the **Physiological School**.

Core idea: Intelligence stems not from complex symbolic operations, but from **networks** of interconnected units (simulating biological neurons). Intelligent behavior emerges by **learning** from data through adjusting connection weights.

Machine learning / deep learning, graphical models / neural networks

Image processing, machine translation / natural language processing

Behaviorism

Behaviorism focuses not on the internal structure of intelligence (symbolic or neural), but rather on the **interaction** between an agent and its environment.

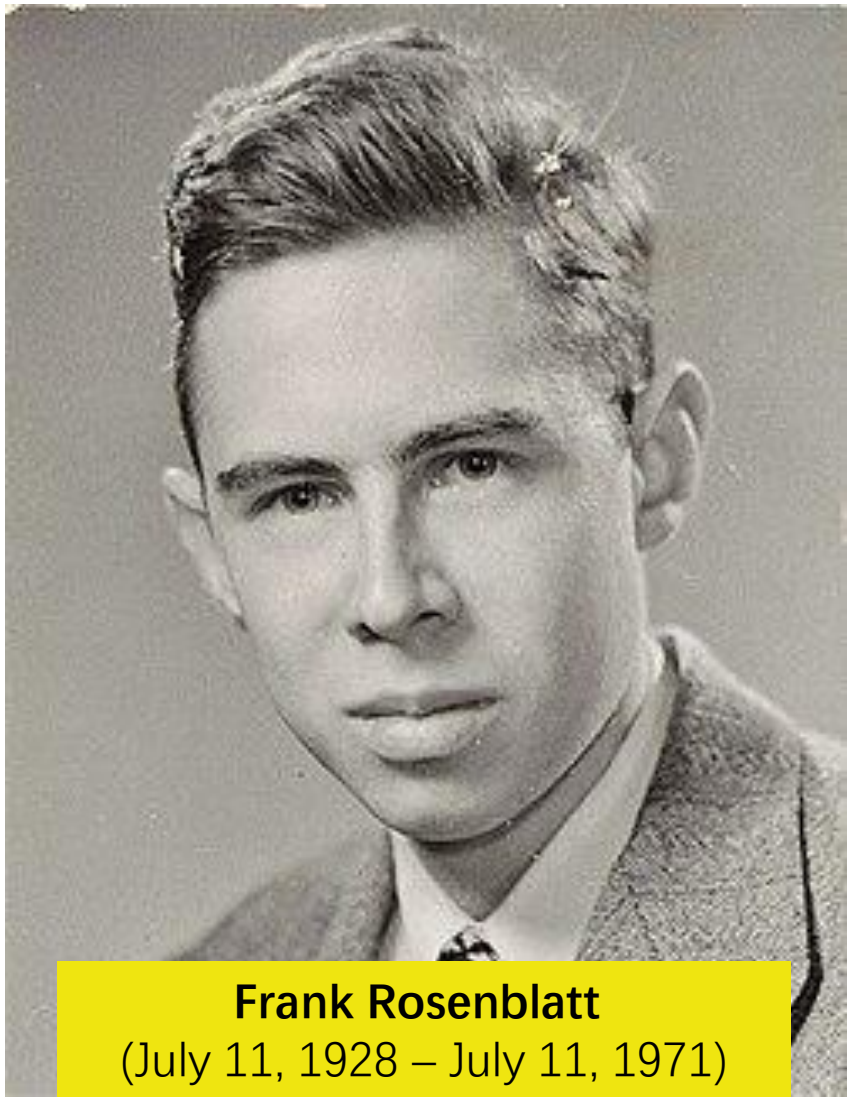
Core idea: Intelligence is demonstrated through interaction with the environment, receiving **feedback (rewards or punishments)**, and learning optimal behaviors through this process. It emphasizes "learning by doing."

Reinforcement learning

Game, self-driving cars, embodied intelligence

Other schools: Evolutionism, Bayesianism, …

Frank Rosenblatt and perceptron



Frank Rosenblatt
(July 11, 1928 – July 11, 1971)

Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN[1]

F. ROSENBLATT

Cornell Aeronautical Laboratory

If we are eventually to understand the capability of higher organisms for perceptual recognition, generalization, recall, and thinking, we must first have answers to three fundamental questions:

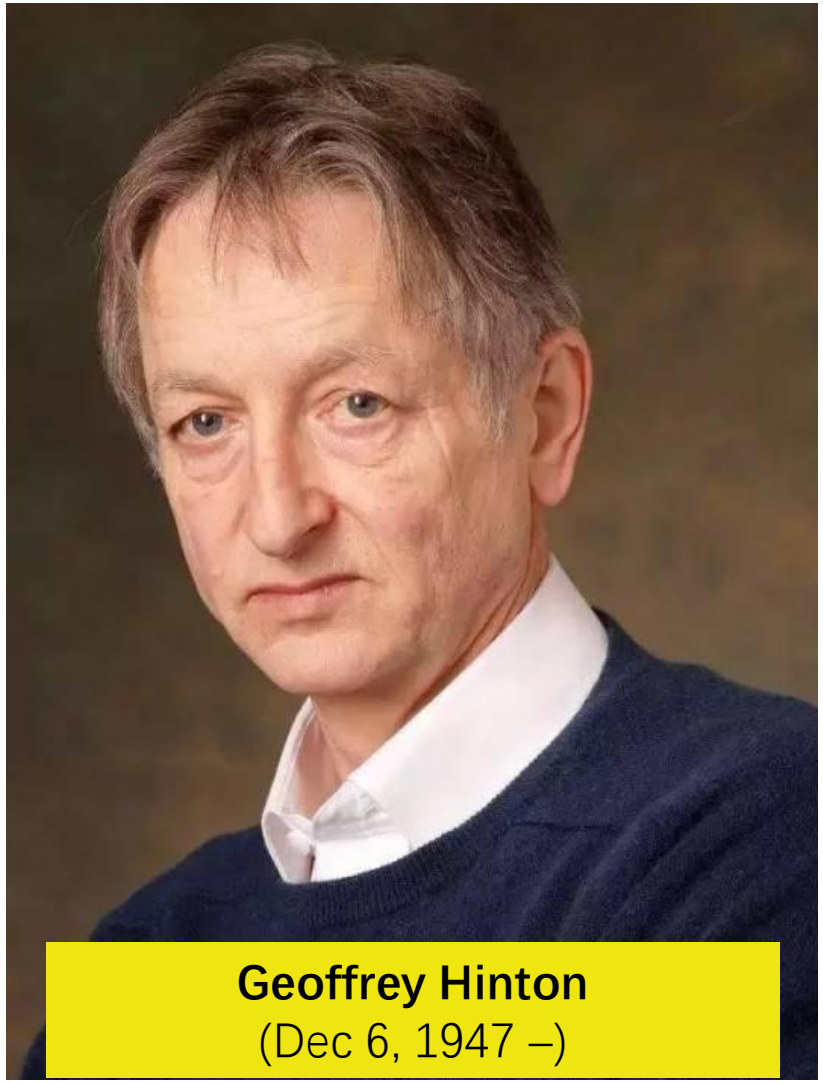
1. How is information about the physical world sensed, or detected, by the biological system?
2. In what form is information stored, or remembered?
3. How does information contained in storage, or in memory, influence recognition and behavior?

The first of these questions is in the province of sensory physiology, and is the only one for which appreciable understanding has been achieved. This article will be concerned primarily with the second and third questions, which are still subject to a vast amount of speculation, and where the few relevant facts currently supplied by neurophysiology have not yet been integrated into an acceptable theory.

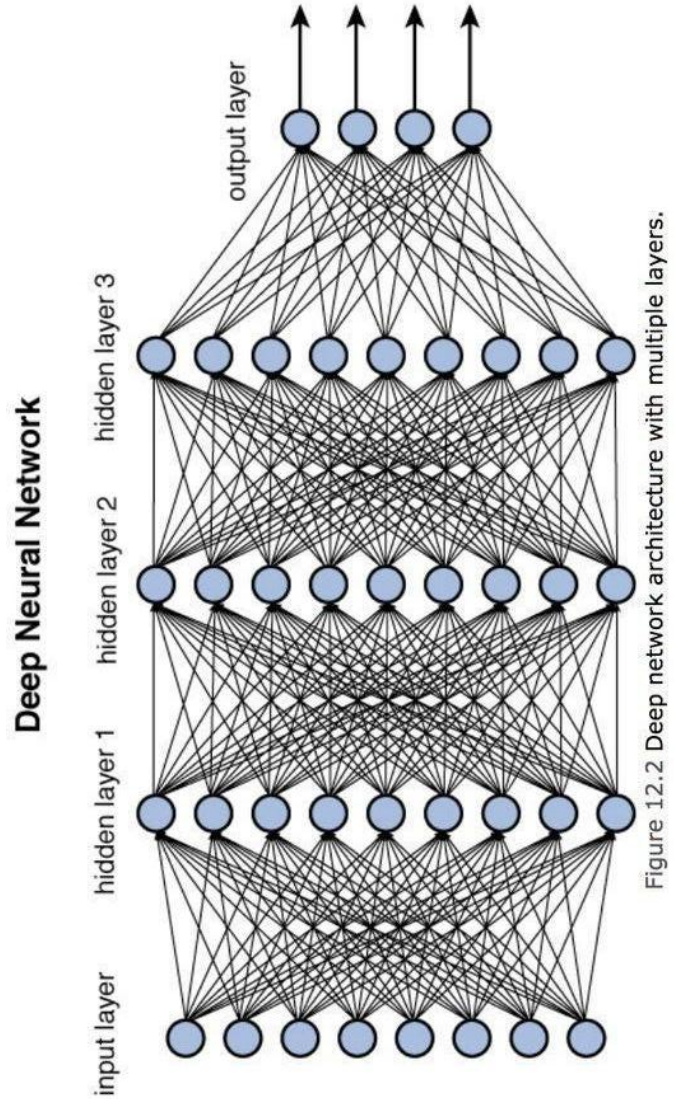
With regard to the second question, two alternative positions have been maintained. The first suggests that storage of sensory information is in the form of coded representations or images, with some sort of one-to-one mapping between the sensory stimulus and the stored pattern. According to this hypothesis, if one understood the code or "wiring diagram" of the nervous system, one should, in principle, be able to discover exactly what an organism remembers by reconstructing the original sensory patterns from the "memory traces" which they have left, much as we might develop a photographic negative, or translate the pattern of electrical charges in the "memory" of a digital computer. This hypothesis is appealing in its simplicity and ready intelligibility, and a large family of theoretical brain models has been developed around the idea of a coded, representational memory (**2**, **3**, **9**, **14**). The alternative approach, which stems from the tradition of British empiricism, hazards the guess that the images of stimuli may never really be recorded at all, and that the central nervous system simply acts as an intricate switching network, where retention takes the form of new connections, or pathways, between centers of activity. In many of the more recent developments of this position (Hebb's "cell assembly," and Hull's "cortical anticipatory goal response," for example) the "responses" which are associated to stimuli may be entirely contained within the CNS itself. In this case the response represents an "idea" rather than an action. The important feature of this approach is that there is never any simple mapping of the stimulus into memory, according to some code which would permit its later reconstruction. Whatever in-

[1] The development of this theory has been carried out at the Cornell Aeronautical Laboratory, Inc., under the sponsorship of the Office of Naval Research, Contract Nonr-2381(00). This article is primarily an adaptation of material reported in Ref. 15, which constitutes the first full report on the program.

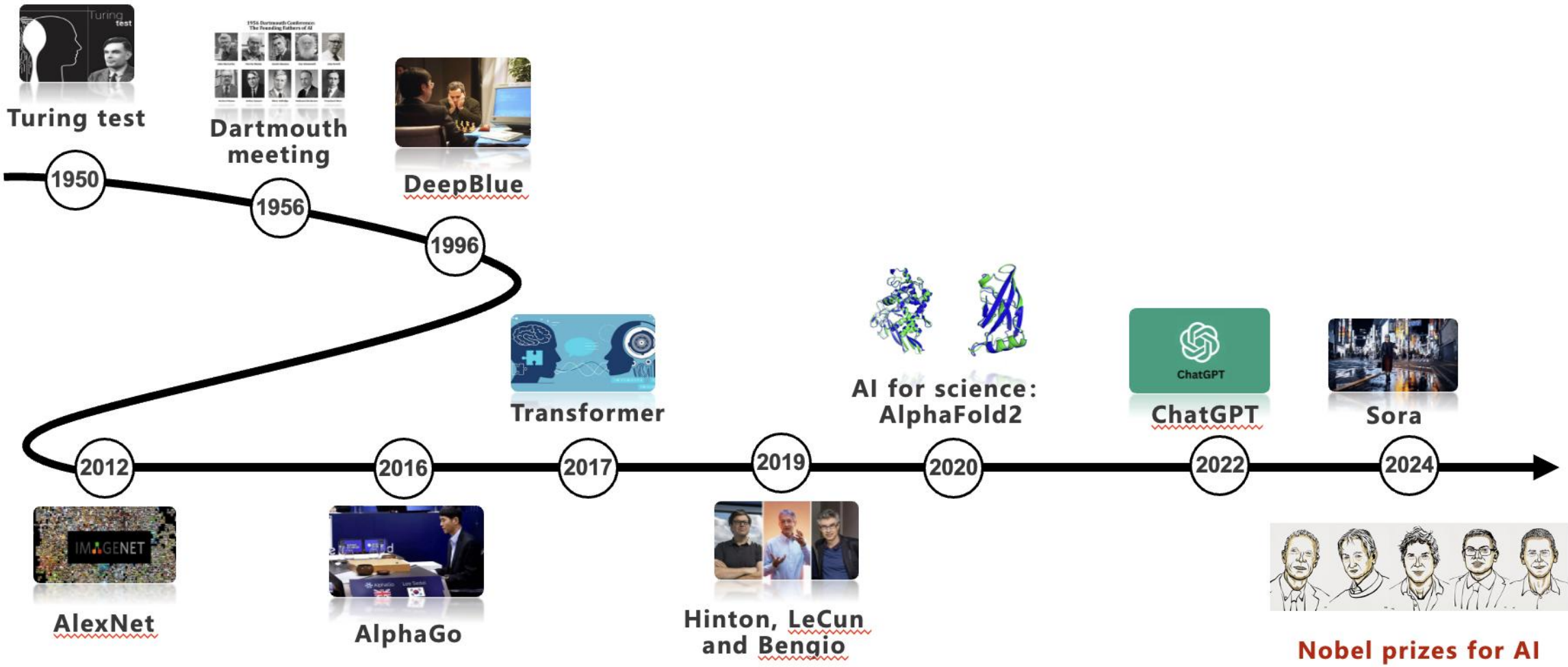
Geoffrey Hinton and deep neural networks



Geoffrey Hinton
(Dec 6, 1947 –)

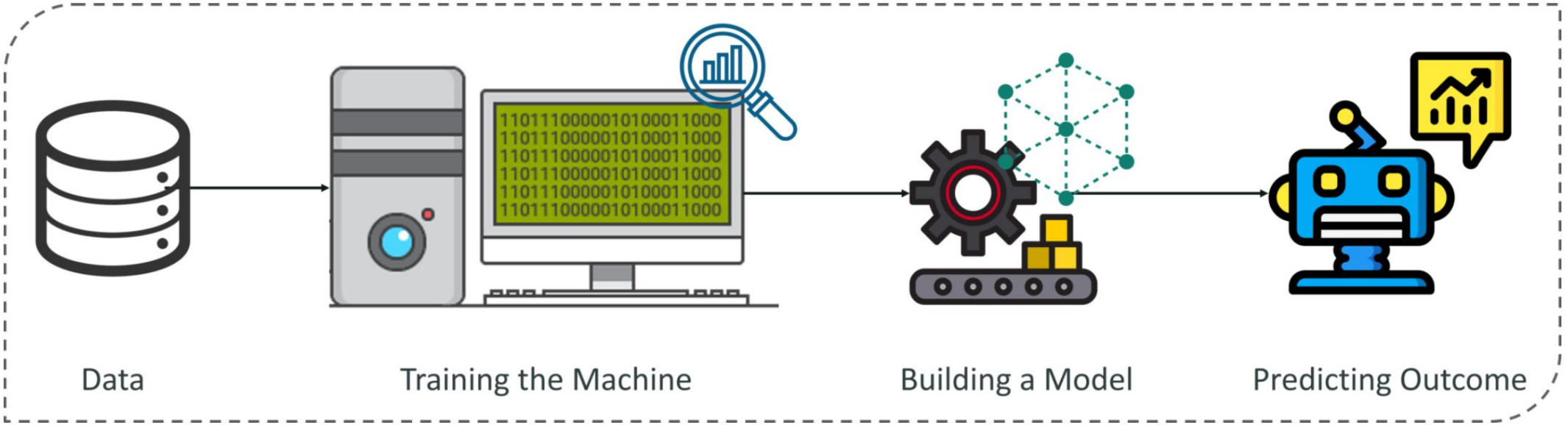


Recent breakthroughs in AI development



What is machine learning?

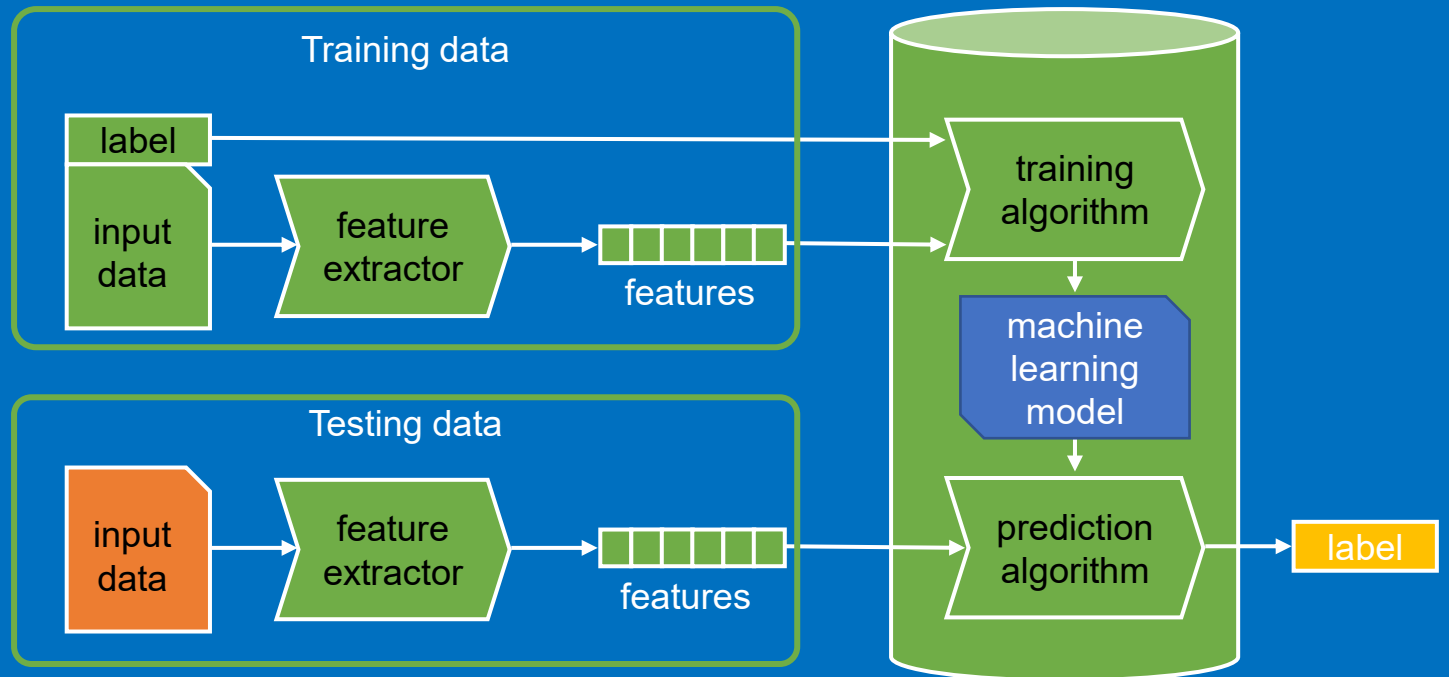
- **Machine learning (ML)** is a field of study in artificial intelligence concerned with the development and study of statistical algorithms that can **learn from data and generalize to unseen data**, and thus perform tasks **without explicit instructions**.



A working paradigm of machine learning

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

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	?



A simple example of machine learning

Problem: decide whether to wait for a table at a restaurant
Attributes:

- Alternate: is there an alternative restaurant nearby?
- Bar: is there a comfortable bar area to wait in?
- Fri/Sat: is today Friday or Saturday?
- Hungry: are we hungry?
- Patrons: number of people in the restaurant (None, Some, Full)
- Price: price range ($, $$, $$$)
- Raining: is it raining outside?
- Reservation: have we made a reservation?
- Type: kind of restaurant (French, Italian, Thai, Burger)
- WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Decision Tree

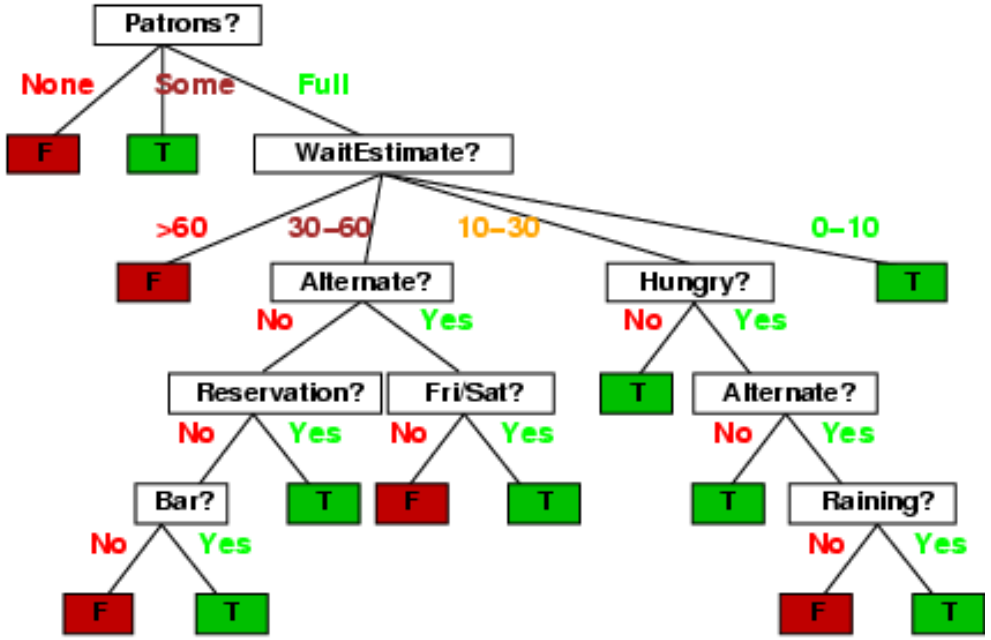
Goal: Learn induction rules from empirical data

- Disjunctive Normal Function!

  E.g., f = (A AND B) OR (NOT(A) AND D)

New Goal: Find the smallest decision tree consistent with the sample

- Unfortunately, this is provably intractable to do optimally
- Greedy heuristic search used in practice:
  - Select root node that is “best” in some sense
  - Partition data into subsets, depending on root attribute value
  - Recursively grow subtrees
  - Different termination criteria
    - For noiseless data, if all examples at a node have the same label, then declare it a leaf and backup
    - For noisy data it might not be possible to find a “pure” leaf using the given attributes



Core algorithm

Recursive algorithm for creating branches - CreateBranch()

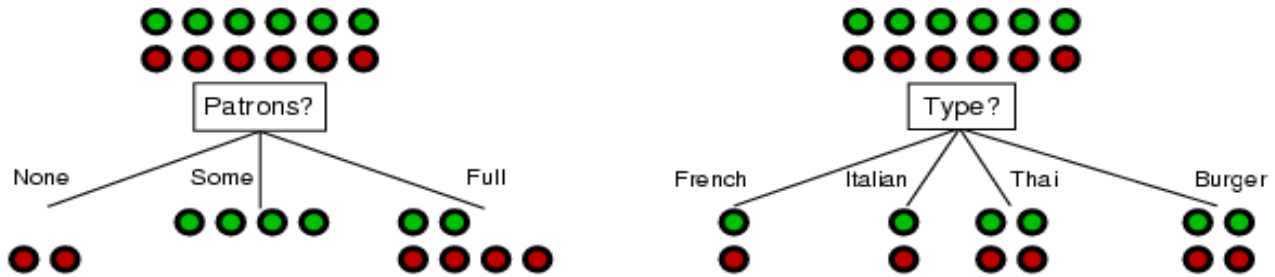
```
detect whether each item in the dataset belongs to the same category:
    if so
        return label of category;
    else
        search for the best attributes to classify each item in the dataset
        split the datasets to subsets
        for each subset:
            CreateBranch
        return branches
```

Decision Tree construction: picking an attribute

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

Patrons is a better choice?

- How can we quantify this?
- One approach would be to use the classification error E directly (greedily)
  - Empirically it is found that this works poorly
- Much better is to use information gain (next slides)



Picking an attribute: information gain

Entropy of a distribution: H(X) = E[I(X)]

= p(x1)I(x1)+p(x2)I(x2)+ p(x3)I(x3)+…

= -p(x1)logp(x1)-p(x2)logp(x2)-p(x3)logp(x3)-…

Information gain IG(A) = H(p) – H(p|A)

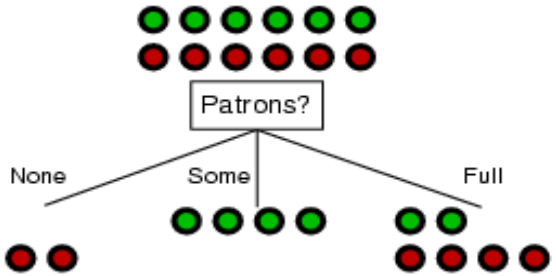
For the training set, *6 positives, 6 negatives, H(6/12, 6/12) = 1* bit

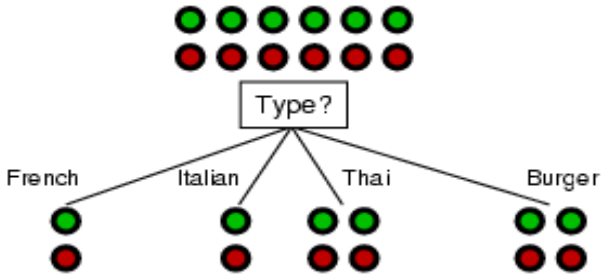
Consider the attributes *Patrons* and *Type:*

Patrons: *1- (2/12*H(0/2,2/2)+4/12*H(4/4,0/4)+6/12*H(2/6,4/6)) = 0.0541* bit

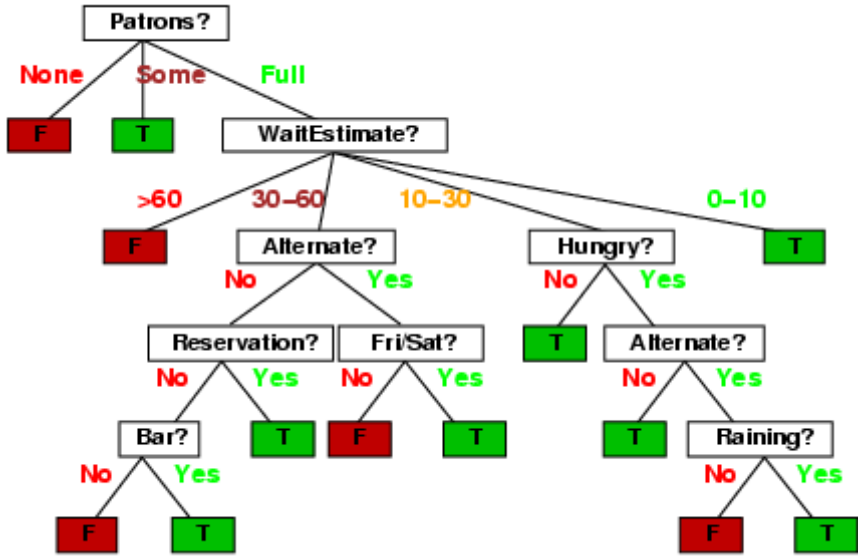
Type: *1- (2/12*H(1/2,1/2)+2/12*H(1/2,1/2)+4/12*H(2/4,2/4)+4/12*H(2/4,2/4)) = 0* bit

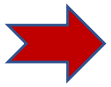
➔ *Patrons* is selected over *Type*!

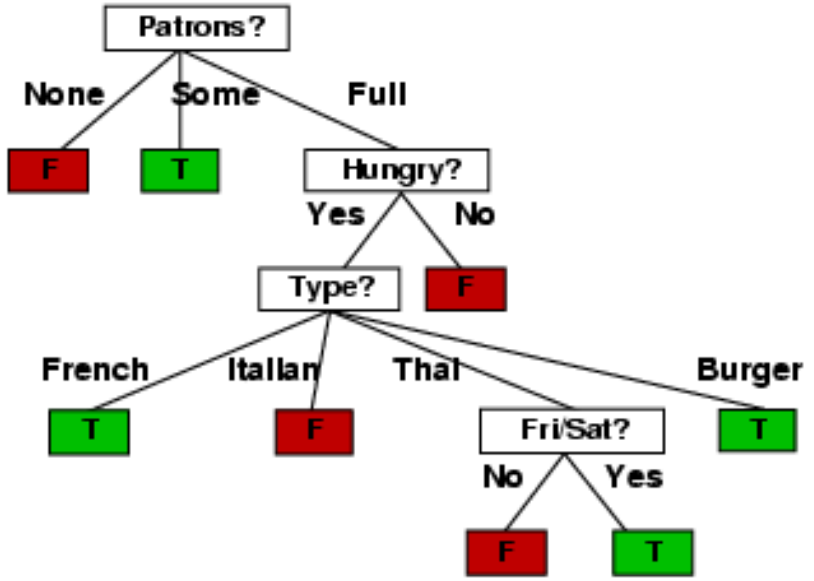




Decision Tree learned





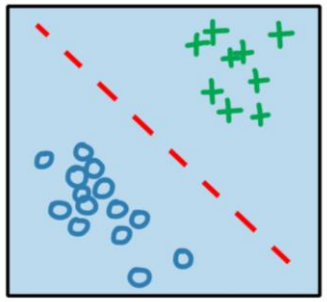


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	T	T	Some	$$$	F	T	French	0–10	T
X_2	T	T	F	T	Full	$	T	F	Thai	30–60	F
X_3	F	T	T	F	Full	$	F	T	Burger	30–60	T

Concepts in machine learning: lessons from the above example

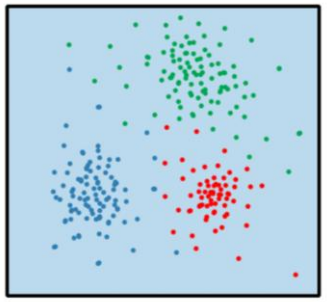
- Data collection (x, y)
  - Positive reference set
  - Negative reference set
  - Multi-class
  - Training/validation/test data
    - Feature/attribute collection
- Representations
- Models
  - Feature/attribute selection
- Performance
  - Overfitting

Types of machine learning



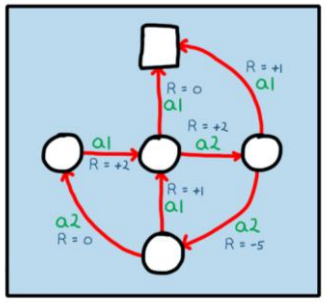
Supervised learning

- Regression
  - Linear
  - Random Forest
- Classification
  - Decision Tree
  - Gaussian Naive Bayes

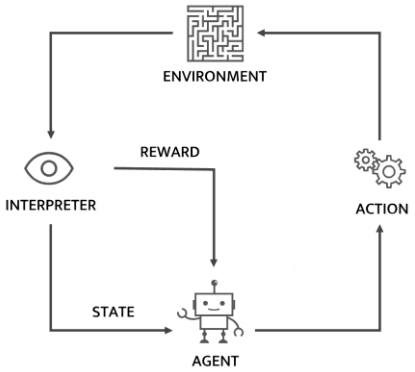


Unsupervised learning

- Clustering
  - Mean Shift
  - Hierarchical
- Dimensionality reduction
  - PCA
  - LDA



Reinforcement learning

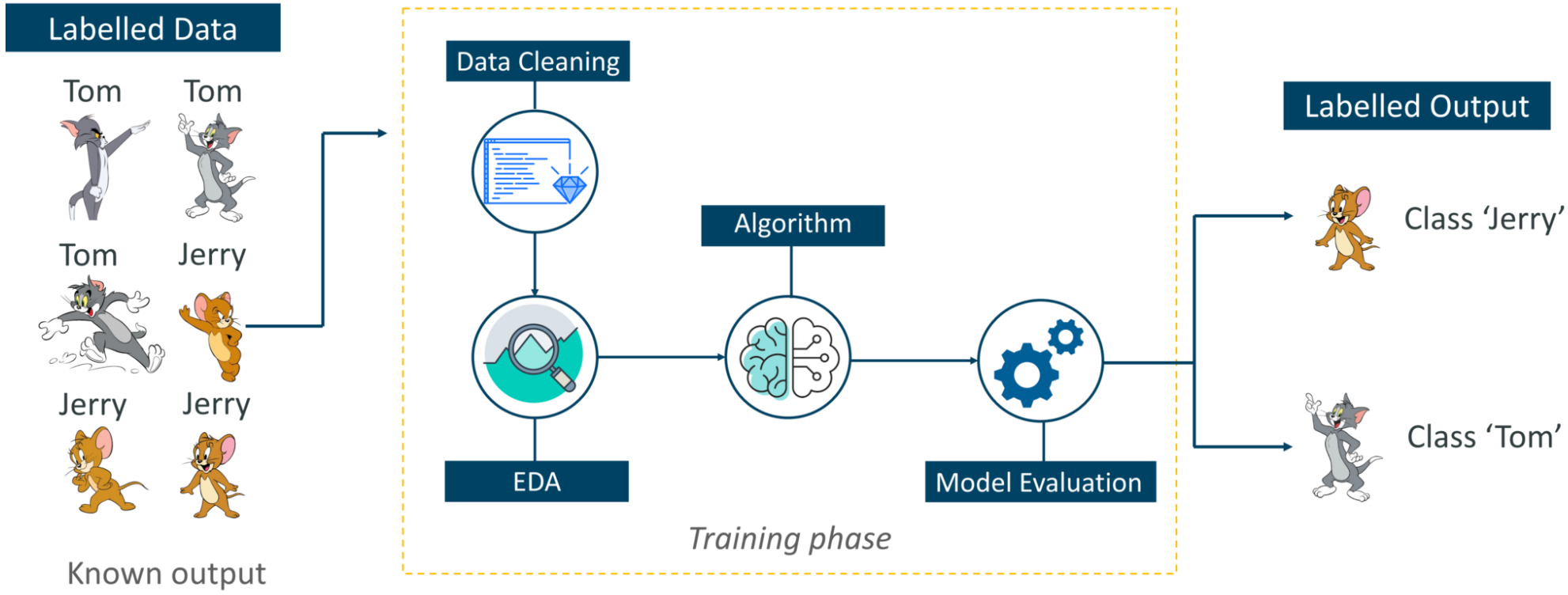


Type of problems solved using machine learning

	Supervised learning	Unsupervised learning
Discrete	Classification	Clustering
Continuous	Regression	Dimensionality reduction

Supervised learning

- **Supervised learning** is a type of machine learning paradigm where an algorithm learns to map input data to a specific output based on example **input-output pairs**.
- Commonly used for tasks like **classification** (discrete) and **regression** (continuous).



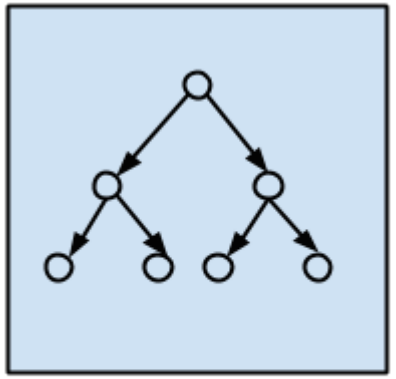
Classification

- Classification is the task of predicting the value of a categorical variable given some input variables (a.k.a. the features or “predictors”).

- Featured algorithms:
  - Naive Bayes
  - k-Nearest Neighbors (kNN)
  - Support Vector Machine (SVM)
  - Random Forest
  - …

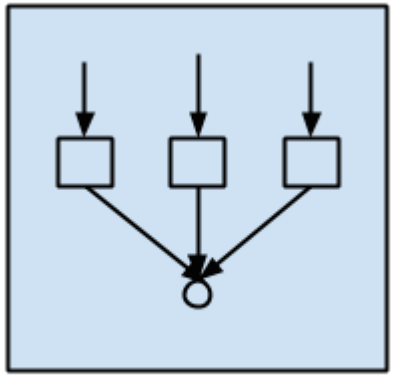
Random forest

- Decision tree methods construct a model of decisions made based on actual values of attributes in the data.



Decision Tree

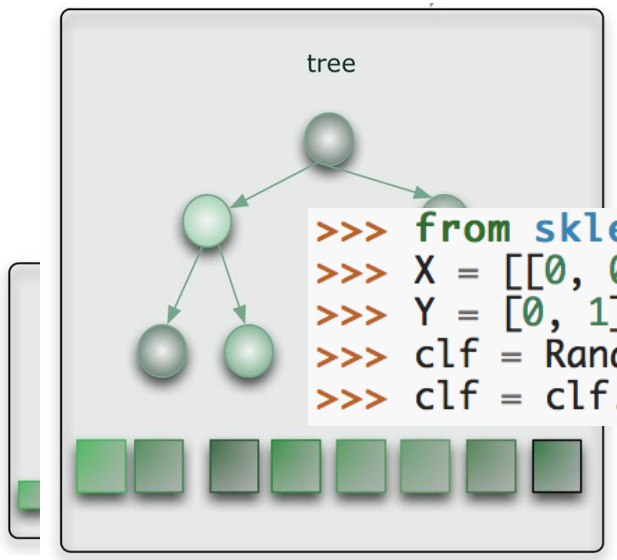
- Ensemble methods are models composed of multiple weaker models that are independently trained and whose predictions are combined in some way to make the overall prediction.



Ensemble Algorithms

Random forest

- Each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.
- Each tree is the best split among a random subset of the features.

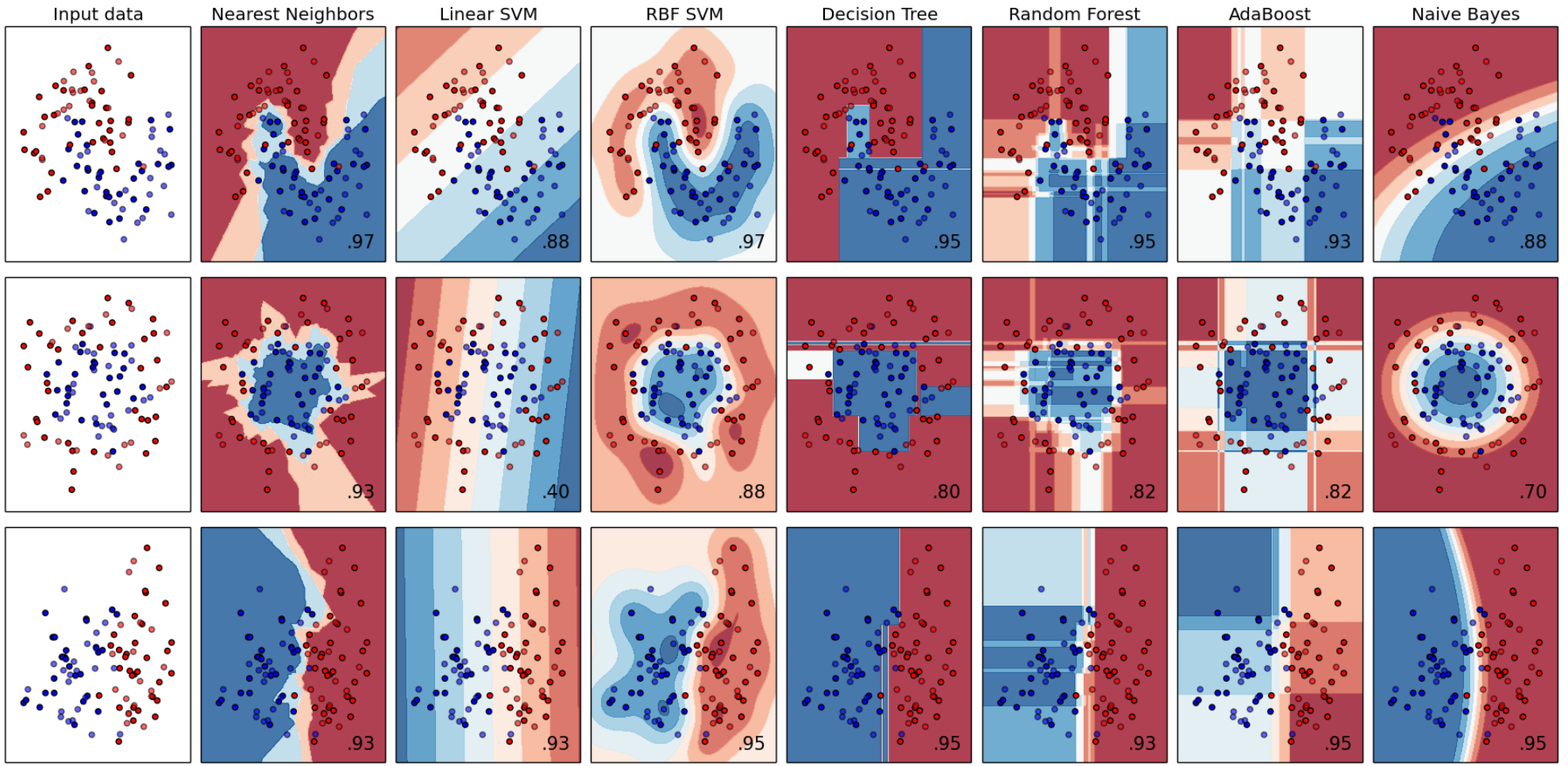


At each node:
choose some small subset of variables at random
find a variable (and a value for that variable) which optimizes the split

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = RandomForestClassifier(n_estimators=10)
>>> clf = clf.fit(X, Y)
```

Classifier comparison

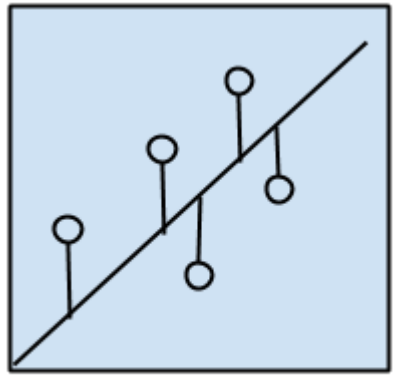
- All models are wrong, but some are useful. -G.E.P. Box



Regression

- Regression is the task of predicting the value of a continuously varying variable (e.g. a price, a temperature, a conversion rate...) given some input variables (a.k.a. the features, “predictors” or “regressors”)

- Featured algorithms:
  - Linear Regression
  - Stepwise Regression
  - …



Regression Algorithms

Linear Regression

- LinearRegression fits a linear model

$$y = w_0 + w_1 x_1 + \cdots + w_p x_p$$

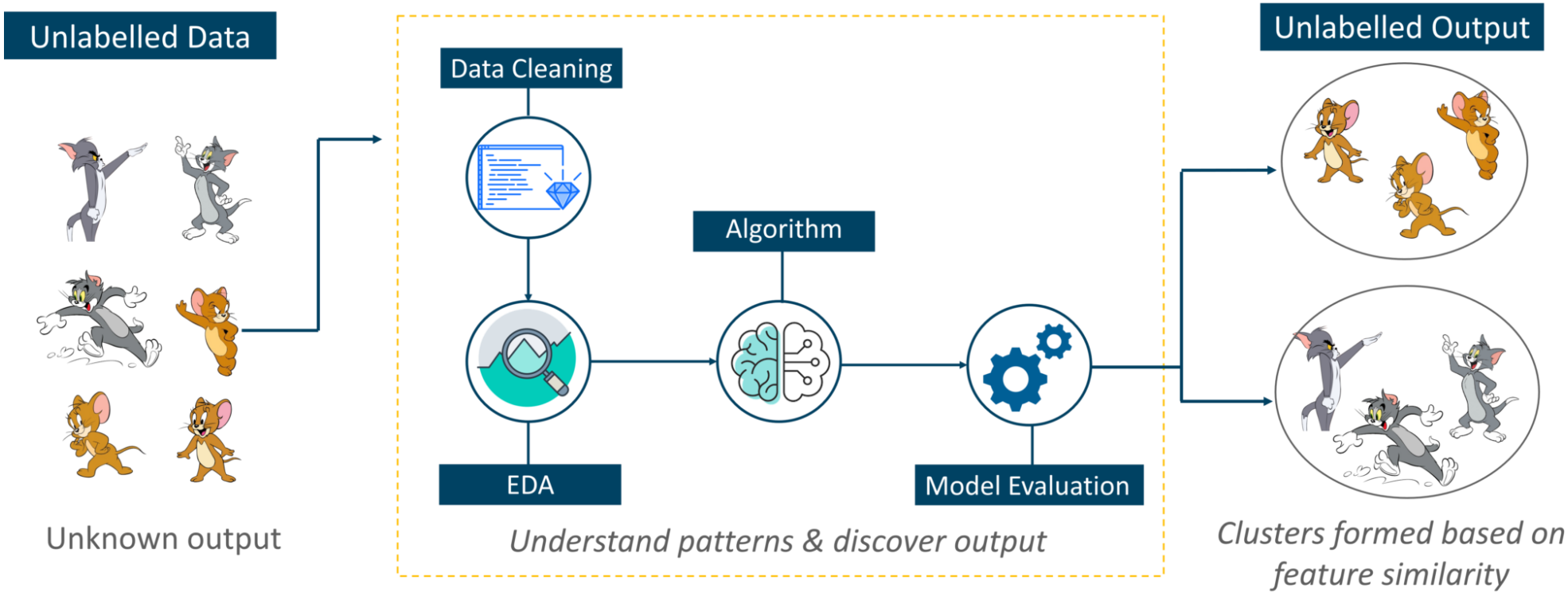
with coefficients $w = (w_1, \ldots, \quad)^{\mathrm{T}}$ to minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation. Mathematically it solves a problem of the form:

$$\min_w \left\| X_w - y \right\|_2^2$$

```
>>> from sklearn import linear_model
>>> reg = linear_model.LinearRegression()
>>> reg.fit ([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
>>> reg.coef_
array([ 0.5,  0.5])
```

Unsupervised learning

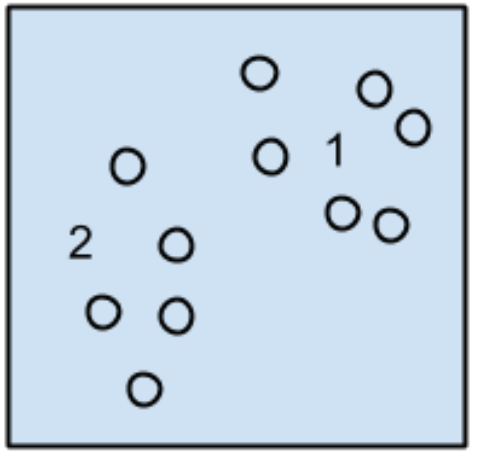
- **Unsupervised learning** is a framework in machine learning where algorithms learn patterns exclusively from **unlabeled data**.
- Commonly used for tasks like **clustering** (discrete) and **dimensionality reduction** (continuous).



Clustering

- Clustering is the task of gathering samples into groups of similar samples according to some predefined similarity or dissimilarity measure (such as the Euclidean distance).

- Featured algorithms:
  - k-Means
  - Gaussian mixture models (GMM)
  - Hierarchical Clustering
  - Spectral Clustering
  - …



Clustering Algorithms

k-Means

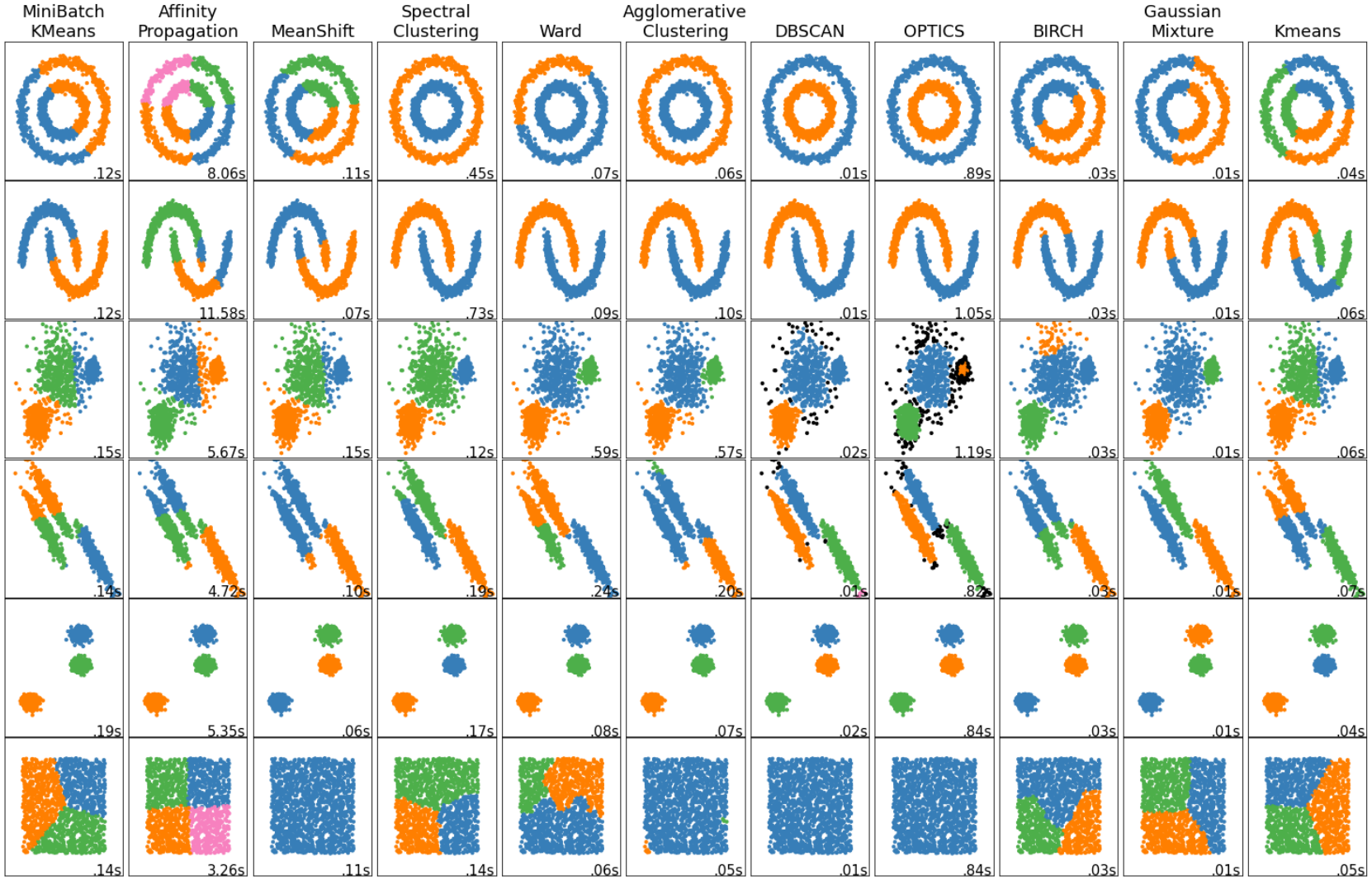
- Find cluster centers and assign data points to the nearest center, minimizing within-cluster distance

$$\underset{s}{arg\min} \sum_{i=1}^{k} \sum_{x \in s_i} \|x - \mu_i\|$$

  - Initially guess cluster centers
  - Assign each data point to the nearest cluster
  - Calculate new cluster centers

```
>>> from sklearn.cluster import KMeans
>>> import numpy as np
>>> X = np.array([[1, 2], [1, 4], [1, 0],
...               [4, 2], [4, 4], [4, 0]])
>>> kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
>>> kmeans.labels_
array([0, 0, 0, 1, 1, 1], dtype=int32)
>>> kmeans.predict([[0, 0], [4, 4]])
array([0, 1], dtype=int32)
>>> kmeans.cluster_centers_
array([[ 1.,  2.],
       [ 4.,  2.]])
```

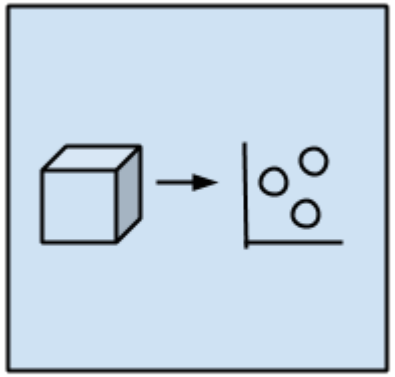
Clustering algorithms comparison



Dimensionality reduction

- Dimensionality reduction is the task of deriving a set of new artificial features that is smaller than the original feature set while retaining most of the variance of the original data.

- Featured algorithms:
  - Principal Component Analysis (PCA)
  - Isomap
  - Multidimensional Scaling (MDS)
  - t-distributed Stochastic Neighbor Embedding (t-SNE)
  - …

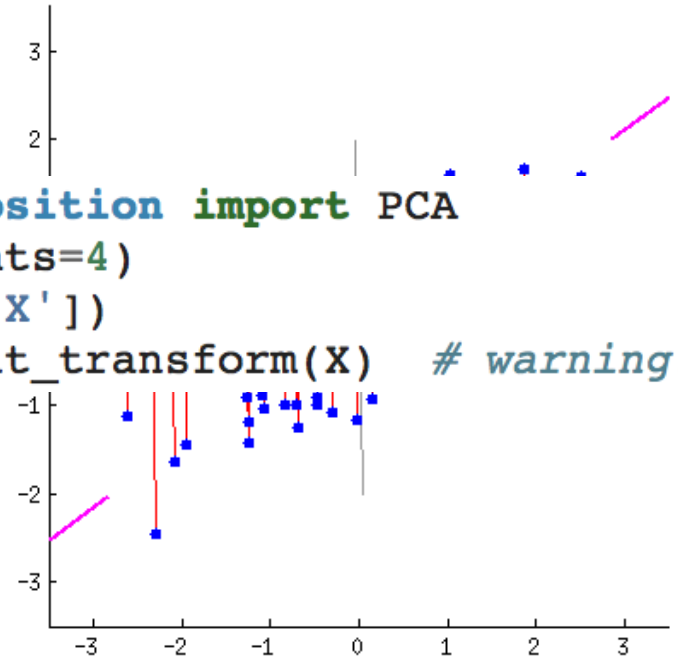


Dimensional Reduction Algorithms

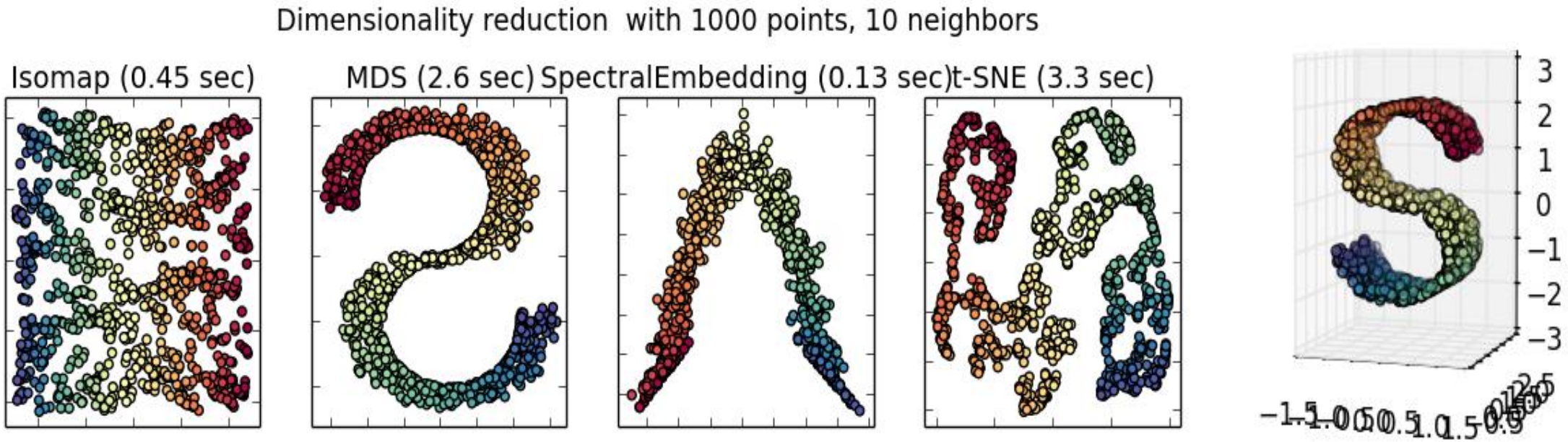
Principal Component Analysis

- Project the data onto an orthogonal linear space
  - Maximize the variance of the projected data
  - Minimize the distance between the data and projections

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=4)
>>> X = normalize(data['X'])
>>> X_projected = pca.fit_transform(X)   # warning: this takes a long time!
```

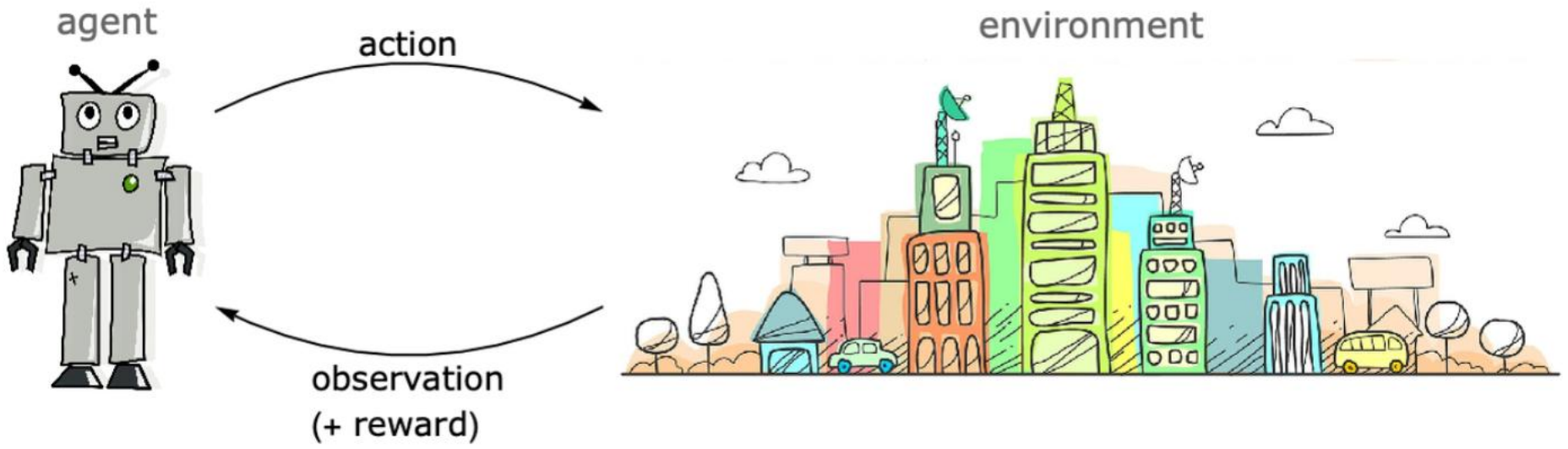


Dimensionality reduction algorithms

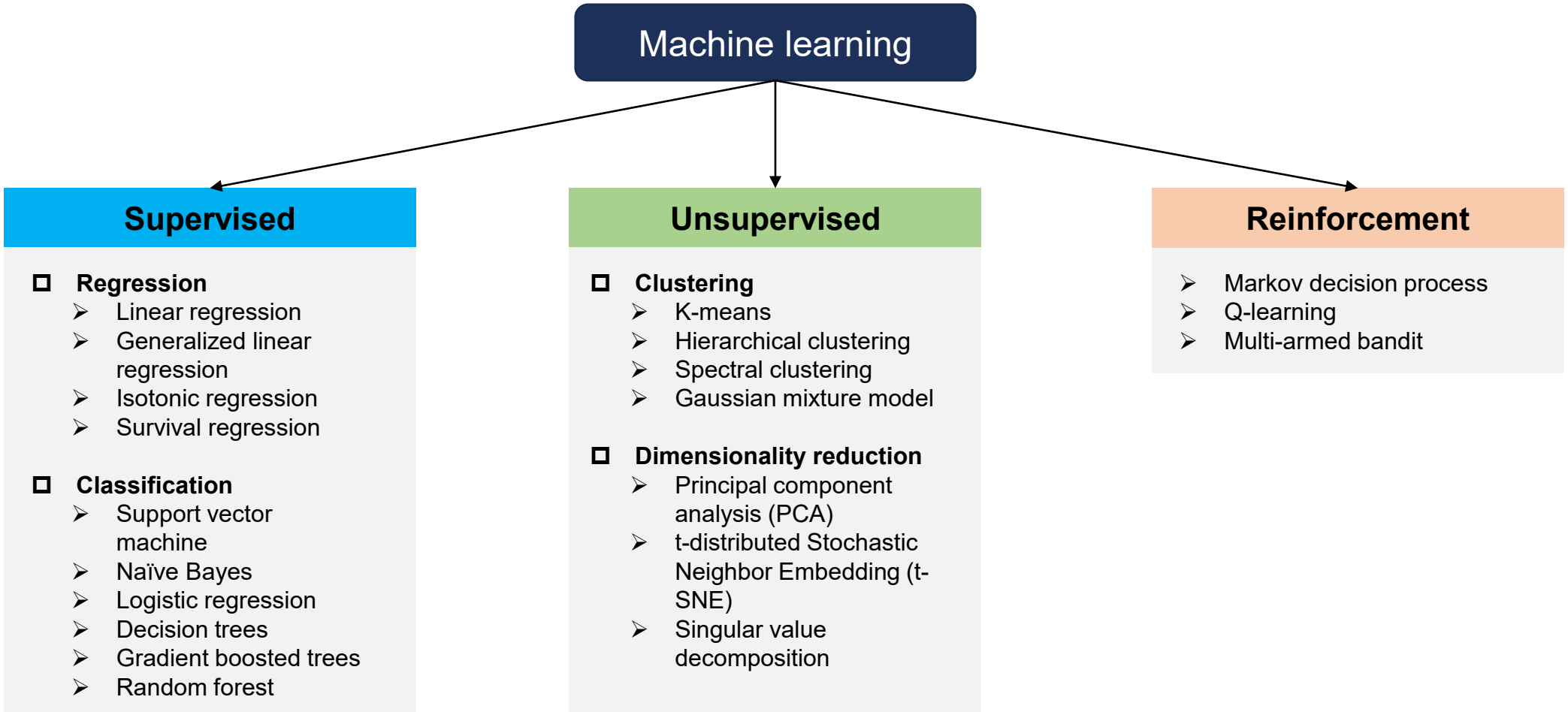


Reinforcement learning

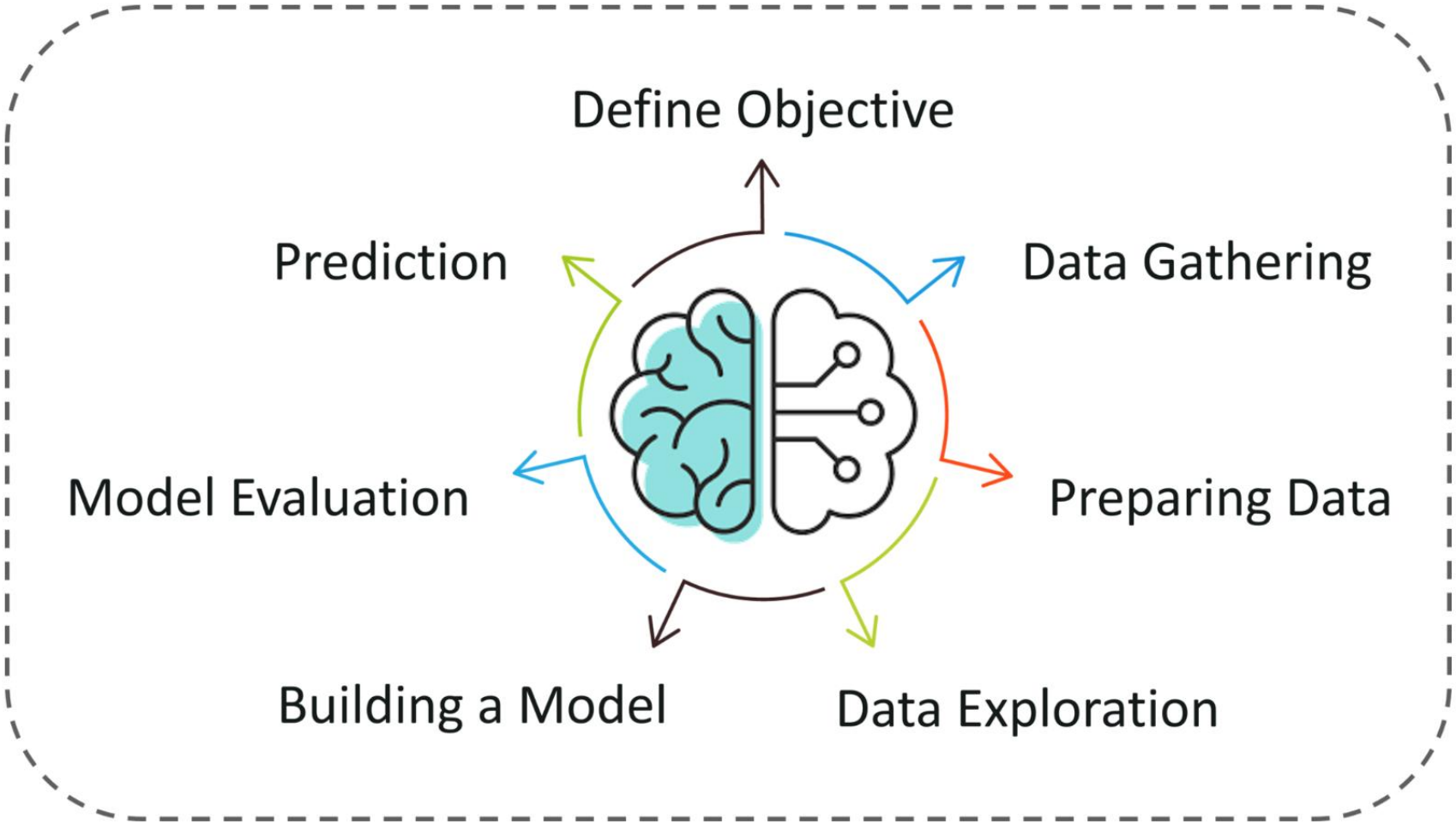
- **Reinforcement learning (RL)** is an interdisciplinary area of machine learning and optimal control concerned with how an intelligent agent should take actions in a dynamic environment in order to maximize a reward signal.



Summary of machine learning methods



Machine learning life cycle

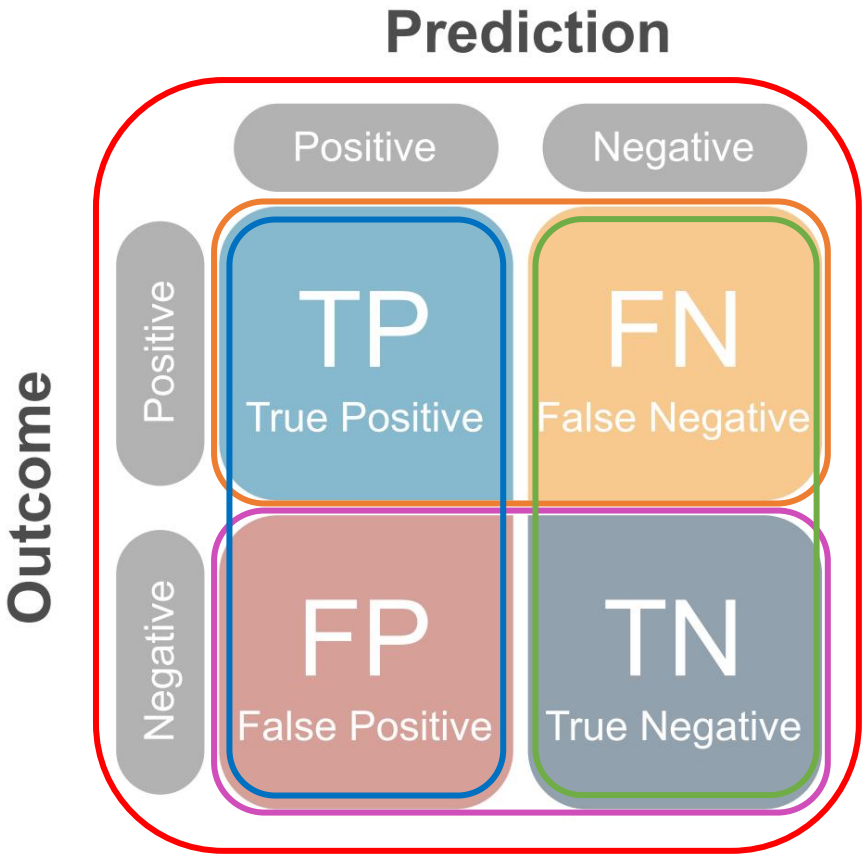


Practical methodology

- Evaluation metrics
  - Cross validation
  - Training and loss function
- Overfitting and generalization
  - Regularization
- Feature engineering

Confusion matrix and related evaluation metrics

- **Confusion matrix**: a table that summarizes the number of correct and incorrect predictions of a **classification** model.



$$Recall\ (or\ TPR, sensitivity) = \frac{TP}{TP + FN}$$

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

$$Precision\ (or\ PPV) = \frac{TP}{TP + FP}$$

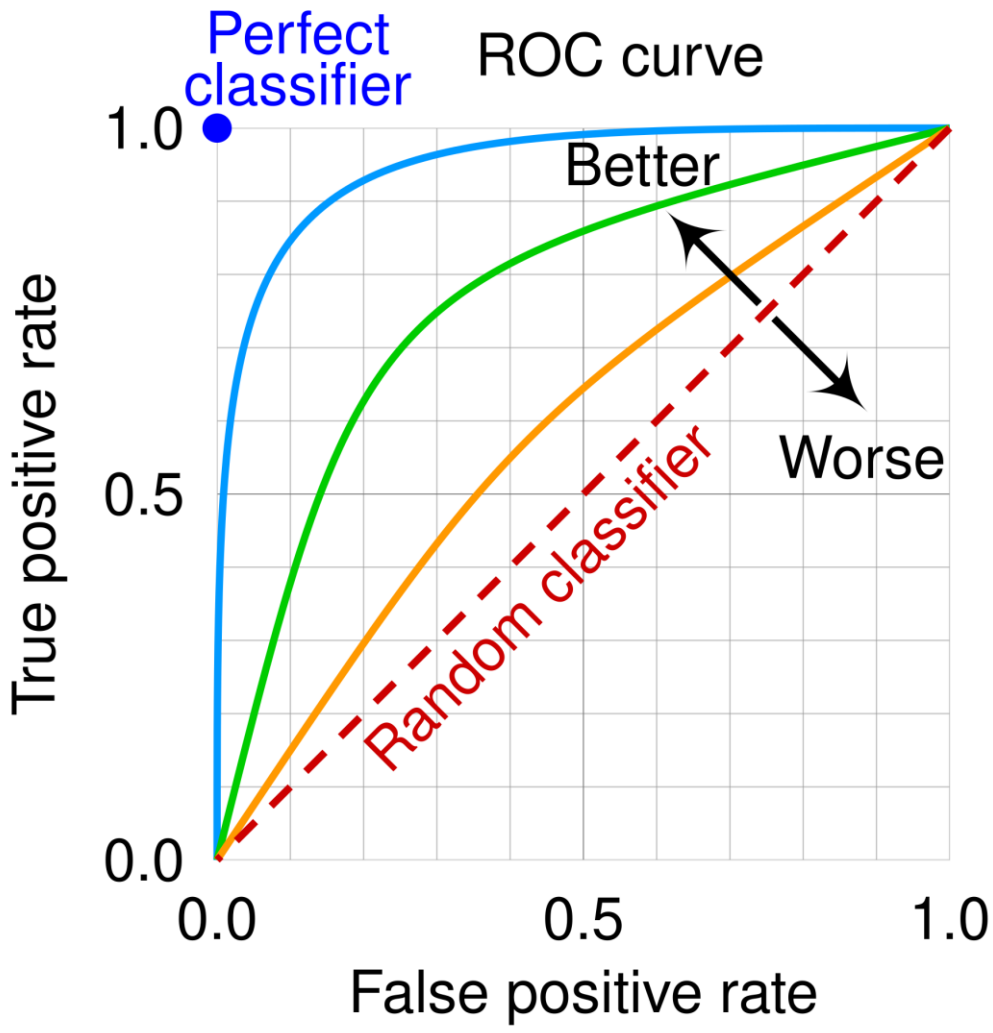
$$NPV = \frac{TN}{TN + FN}$$

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

$$\mathrm{F}_1 score = \frac{2Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$

ROC and AUC

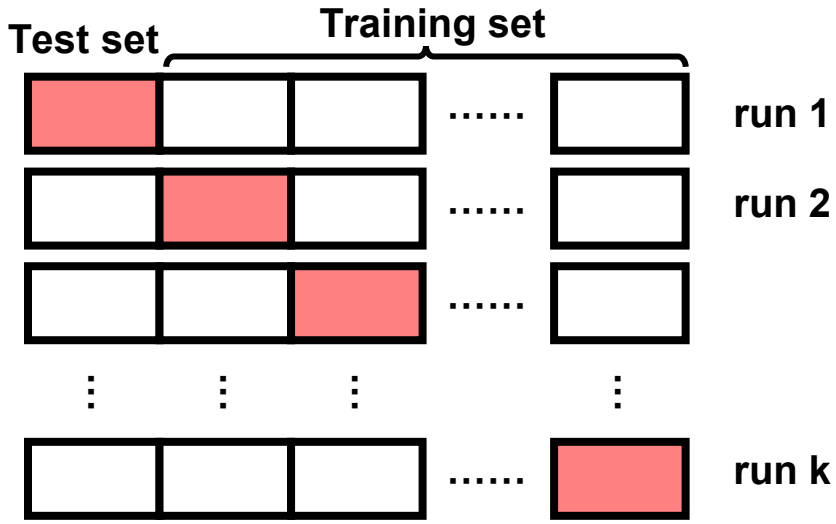
- Evaluate a model's quality **across all possible thresholds.**
- **ROC** (Receiver-operating characteristic curve): a visual representation of model performance across all thresholds. The ROC curve is drawn by calculating the true positive rate (TPR) and false positive rate (FPR) at every possible threshold
- **AUC**: area under the ROC curve



Cross validation

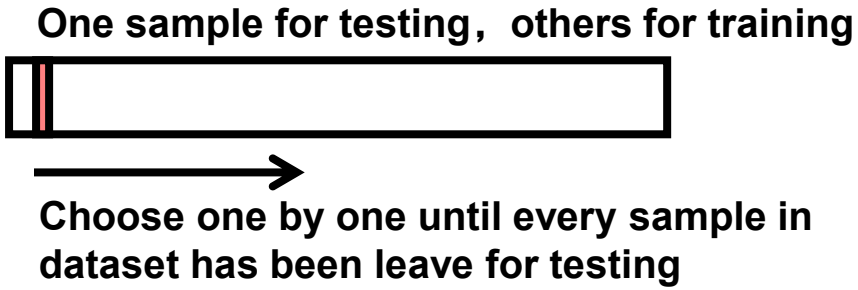
K-fold Cross-Validation

- A machine learning model should be validated on an independent test set
- Cross-validation creates k groups of data, use k-1 for training, one for testing (k << N).
- The error of Cross-Validation is the average of all Cross-Validation models on their corresponding test set.

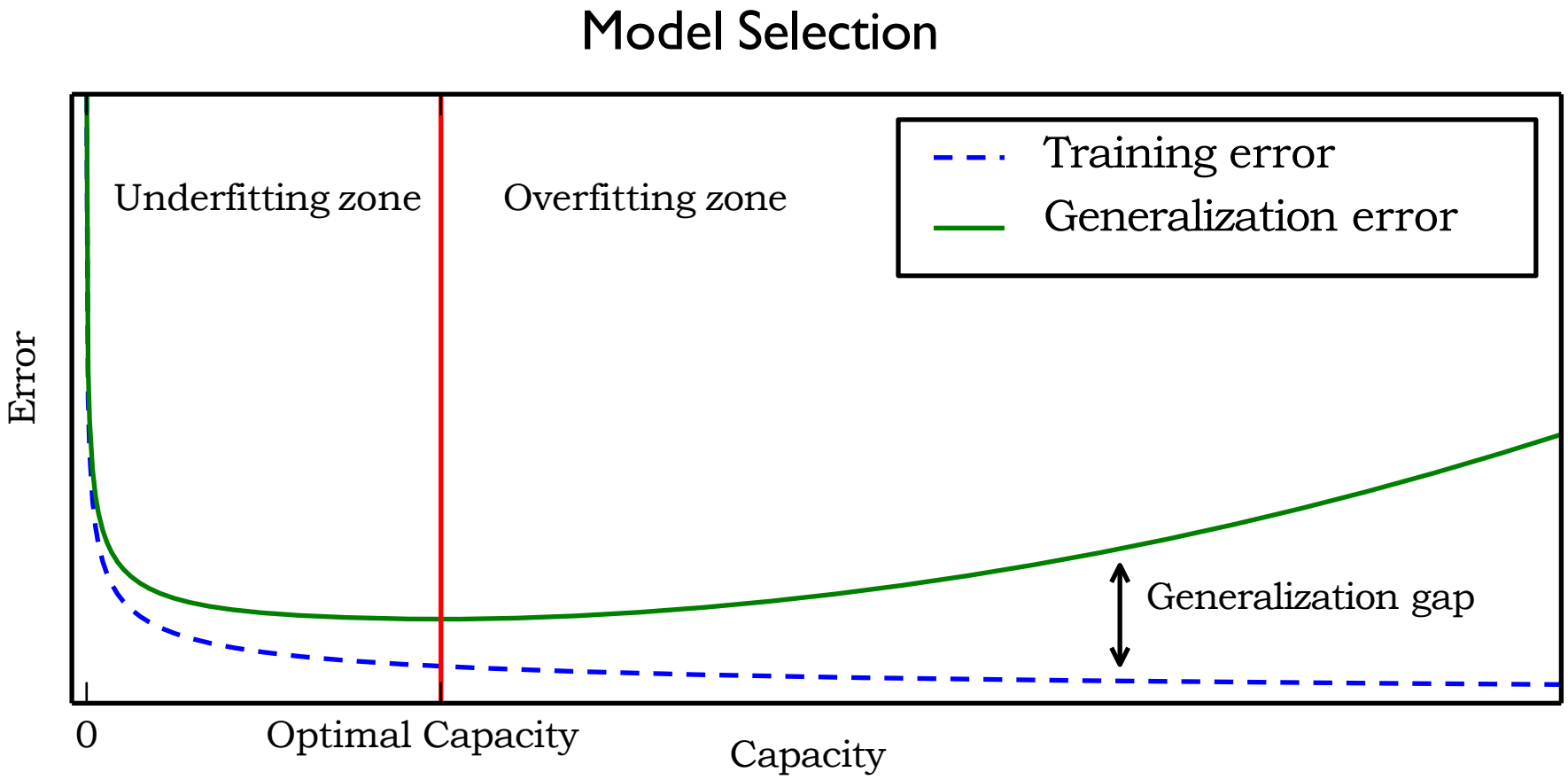


Leave-one-out Cross-Validation

- It is extreme case in K-fold Cross-Validation
- k is the number of all samples. (k = N)

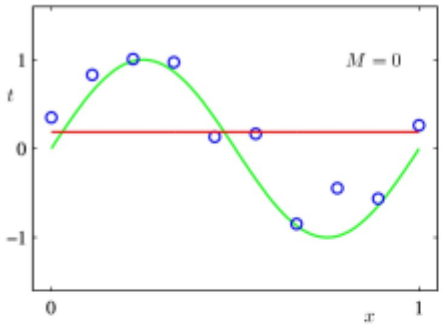


Overfitting and underfitting

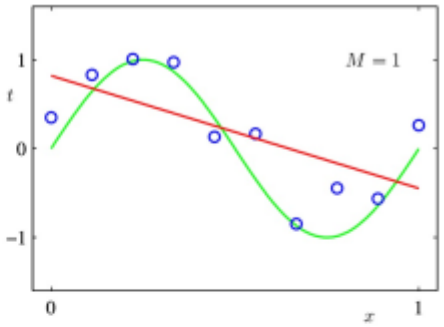


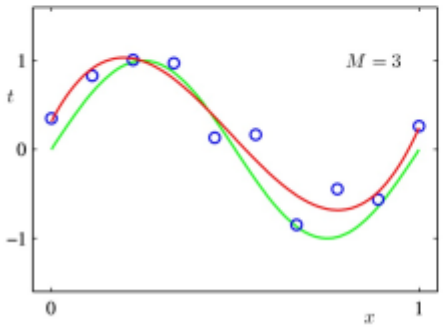
Regularization

$y = w_0$

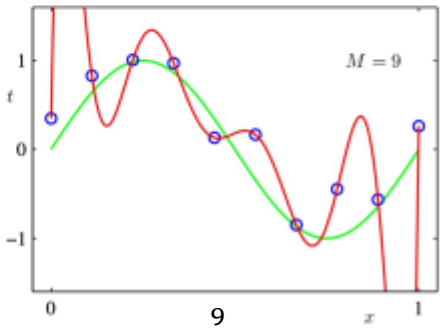


$y = w_0 + w_1 x$





$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$



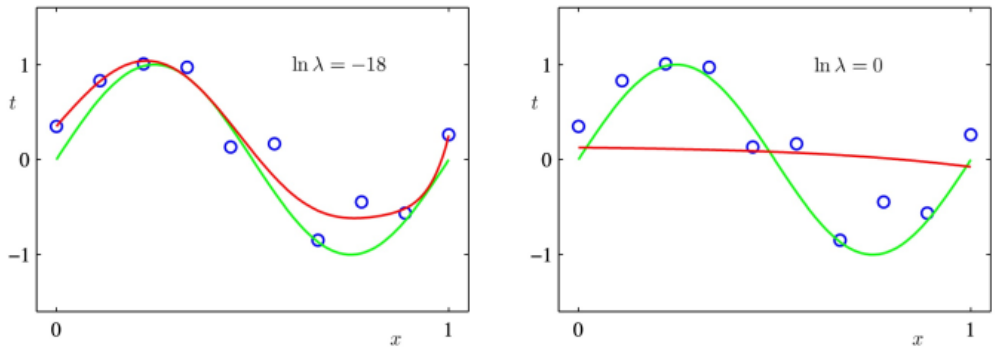
$y = \sum_{i=0}^{9} w_i x^i$

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Parameters is too large !

M = 9 → error(w) = 0: but this is overfitting

Lasso and ridge



	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Fits for $M = 9$

Adding a regularization term

$$E(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - wx_i)^2 + \lambda\phi(w)$$

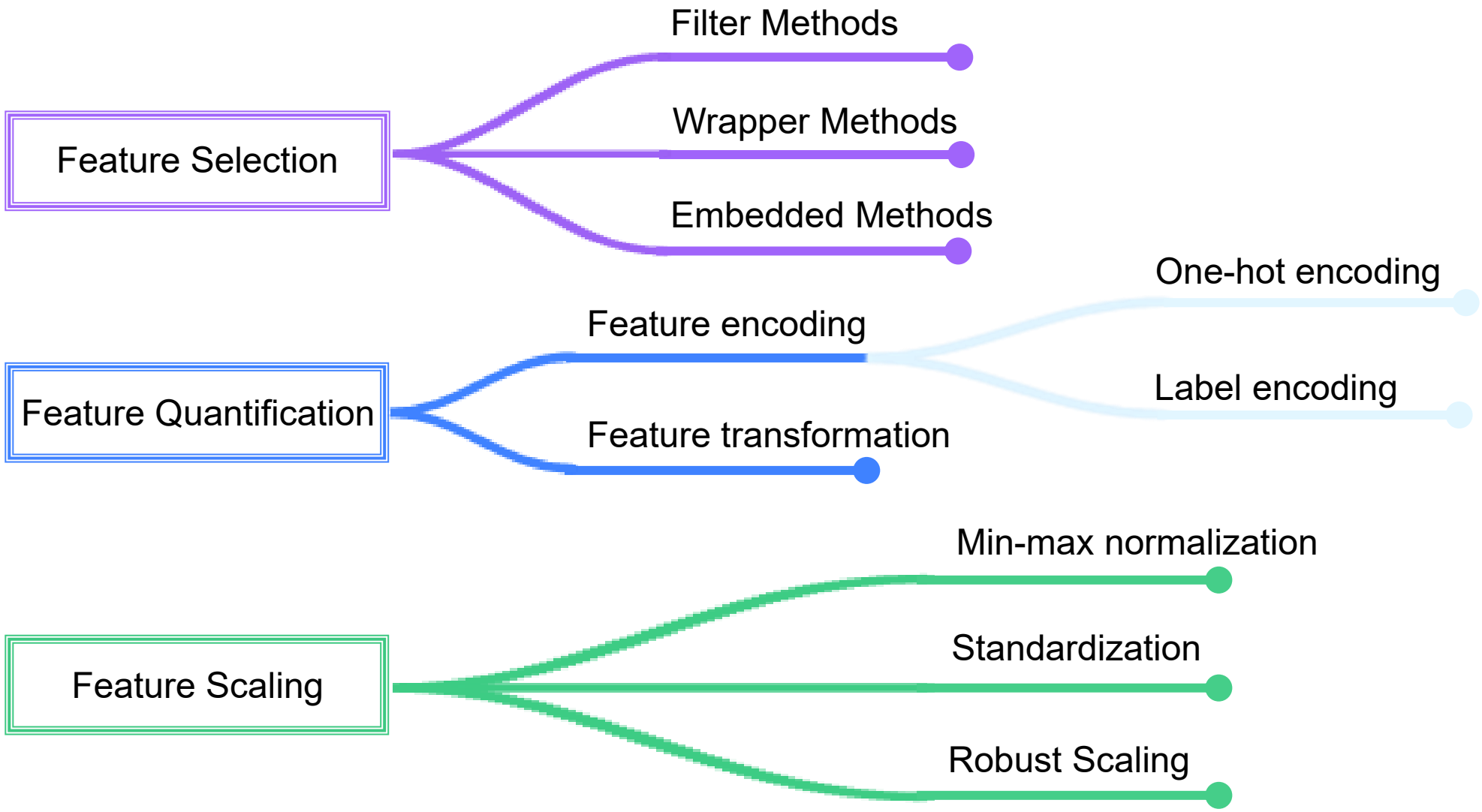
Lasso regression:

$$E(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - wx_i)^2 + \lambda\|w\|_1$$

Ridge regression:

$$E(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - wx_i)^2 + \lambda\|w\|_2^2$$

Feature engineering



Feature of a car

Feature selection | **Feature transform** | **Feature encoding**



car

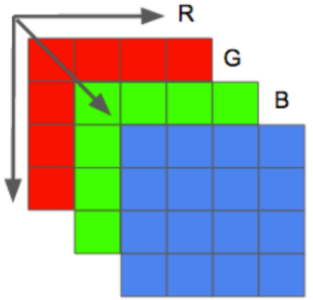
?



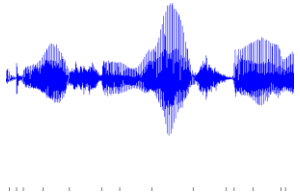
Image of appearance



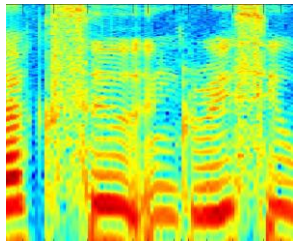
vision features



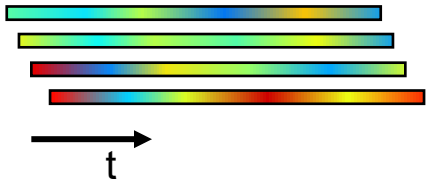
?



Audio of engine



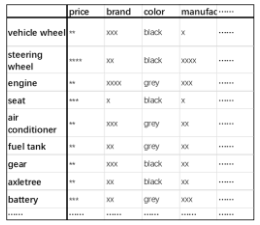
audio features



?

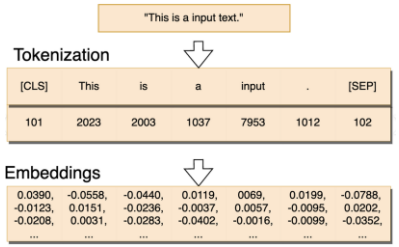


Object of components



	price	brand	color	manufac
vehicle wheel	**	xxx	black	x
steering wheel	****	xx	black	xxxx
engine	**	xxxx	grey	xxx
seat	***	x	black	x
air conditioner	**	xxx	grey	xx
fuel tank	**	xx	grey	xx
gear	**	xxx	black	xx
axletree	**	xx	black	xx
battery	***	xx	grey	xxx

texture features



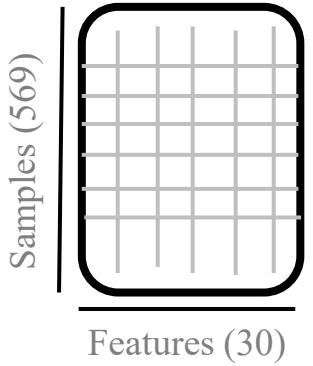
Q&A

How to construct a decision tree: predict breast cancer

	mean		error		worst	
	min	max	min	max	min	max
radius	6.981	28.110	0.112	2.873	7.930	36.040
texture	9.710	39.280	0.360	4.885	12.020	49.540
perimeter	43.790	188.500	0.757	21.980	50.410	251.200
area	143.500	2501.000	6.802	542.200	185.200	4254.000
smoothness	0.053	0.163	0.002	0.031	0.071	0.223
compactness	0.019	0.345	0.002	0.135	0.027	1.058
concavity	0.000	0.427	0.000	0.396	0.000	1.252
concave points	0.000	0.201	0.000	0.053	0.000	0.291
symmetry	0.106	0.304	0.008	0.079	0.156	0.664
fractal dimension	0.050	0.097	0.001	0.030	0.055	0.208

```
cancer.feature_names
```

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```




```
cancer.target_names
```

```
array(['malignant', 'benign'], dtype='<U9')
```



Step1: prepare our datasets

```
from sklearn.datasets import load_breast_cancer
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

Import the Python packages, and you can implement AI algorithms more quickly!

```
cancer=load_breast_cancer()    # Import datasets
X_train, X_test, y_train, y_test=train_test_split(cancer.data, cancer.target,
    stratify=cancer.target, random_state=42)
print(X_train.shape)
print(X_test.shape)
```

Input data: samples * features

Labels, 1 or 0

Split the datasets to training and test sets.

```
(426, 30)
(143, 30)
```

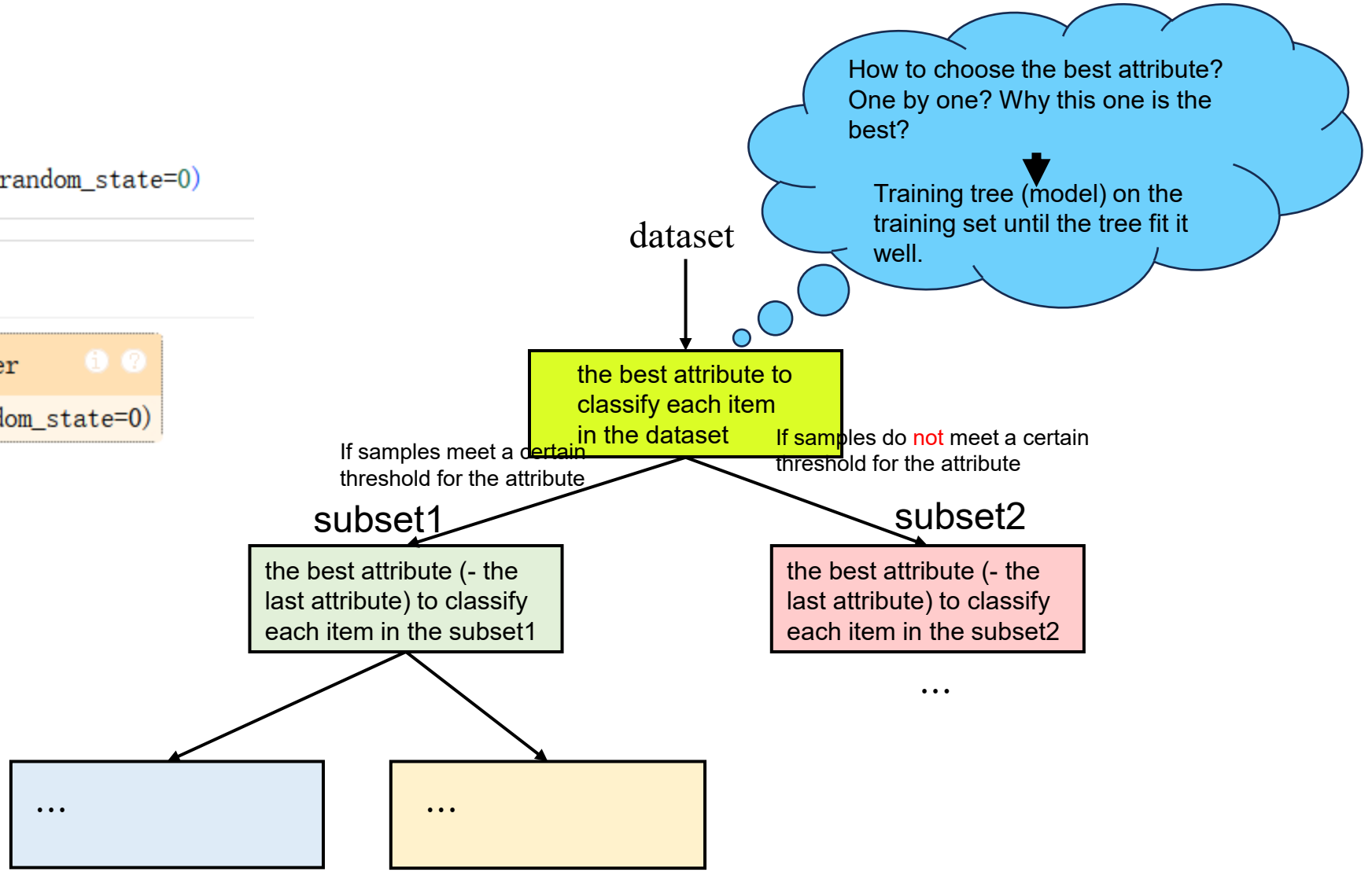
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points		
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0	malignant
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0	malignant
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0	malignant
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0	malignant
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0	malignant

Step2: construct Decision Tree

```
clf=DecisionTreeClassifier(random_state=0)
```

```
clf
```

```
DecisionTreeClassifier(random_state=0)
```



Step3: train and predict

```
clf.fit(X_train, y_train)      # Training tree (model) on the training set until the tree fit it well
print("accuracy in the train set: {:.3f}".format(
        clf.score(X_train, y_train)))
print("accuracy in the test set: {:.3f}".format(
        clf.score(X_test, y_test)))
```

```
accuracy in the train set: 1.000
accuracy in the test set: 0.937
```



It seems to indicate that the performance on the test set is much worse than that on the training set. Why?

0	malignant
1	benign

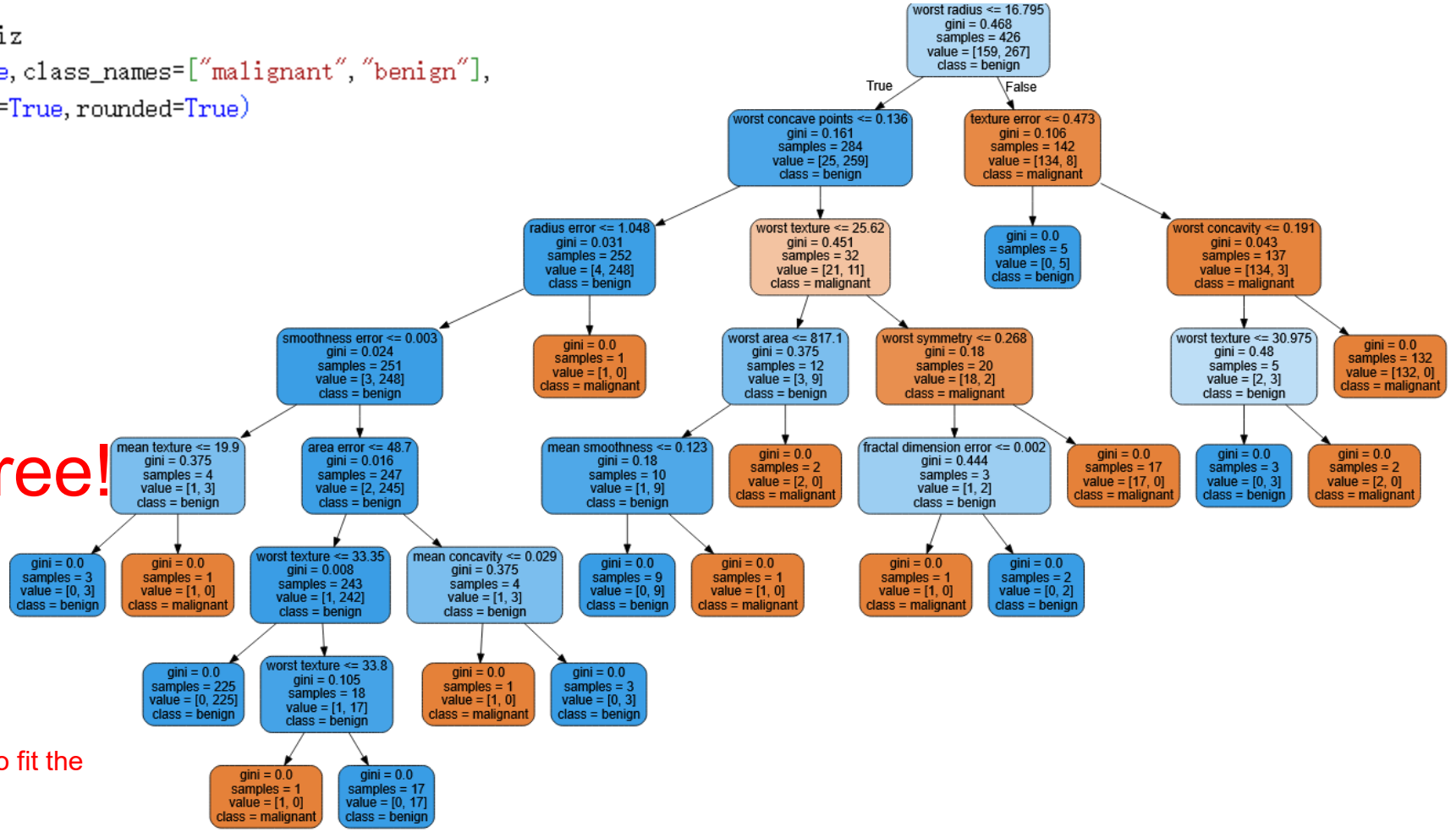
```
print(clf.predict(X_test))
print(y_test)
```

Predict on the test set using the tree

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

Step4: visualize tree

```
from sklearn.tree import export_graphviz
dot_data=export_graphviz(clf, out_file=None, class_names=["malignant", "benign"],
feature_names=cancer.feature_names, filled=True, rounded=True)
import graphviz
graphviz.Source(dot_data)
```



Oh! It is a so big tree!

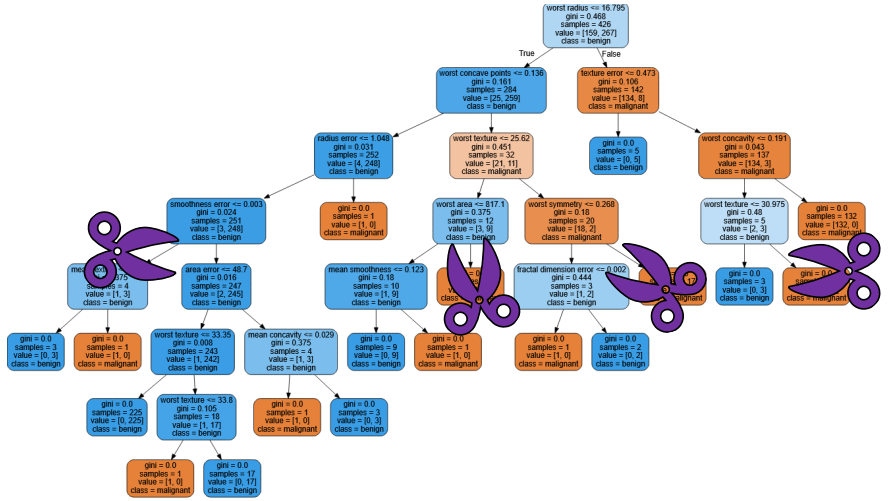
Maybe we choose too much parameters to fit the training set, so this tree is overfitting.

Step5: pruning

```
tree=DecisionTreeClassifier(max_depth=4,random_state=0)
tree.fit(X_train,y_train)
print("accuracy in the train set: {:.3f}".format(
      tree.score(X_train,y_train)))
print("accuracy in the test set: {:.3f}".format(
      tree.score(X_test,y_test)))
```

```
accuracy in the train set: 0.988
accuracy in the test set: 0.951
```

Test is better



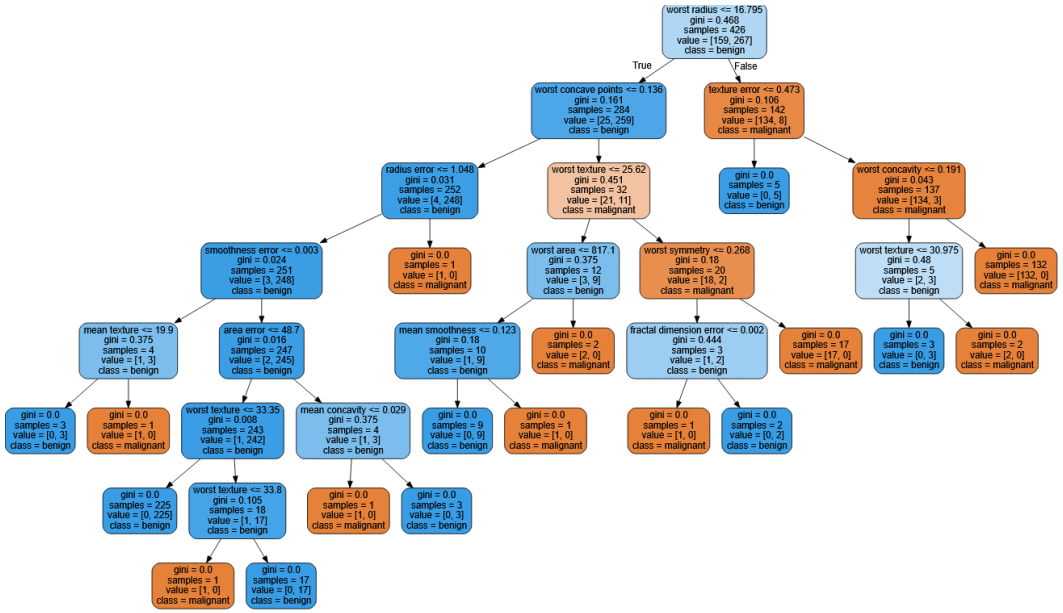
Occam's Razor(奥卡姆剃刀) → Try simple classifiers first

Better to have **smart features** and **simple classifiers** than simple features and smart classifiers



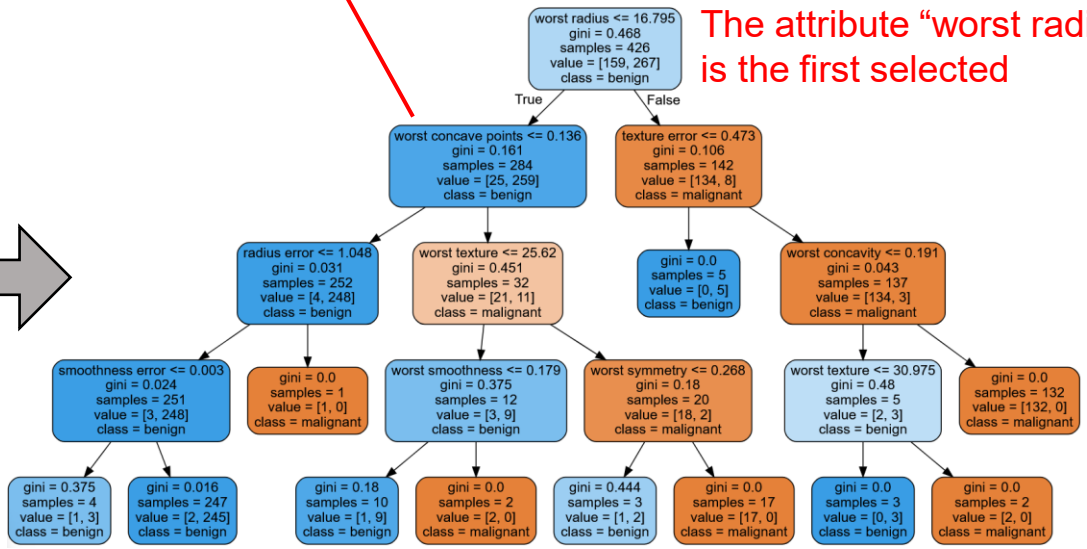
In our life, we also need to prune crops and cash crops to maximize their production capacity.

Step5: pruning



The attribute “worst concave points” and “texture error” is the second selected

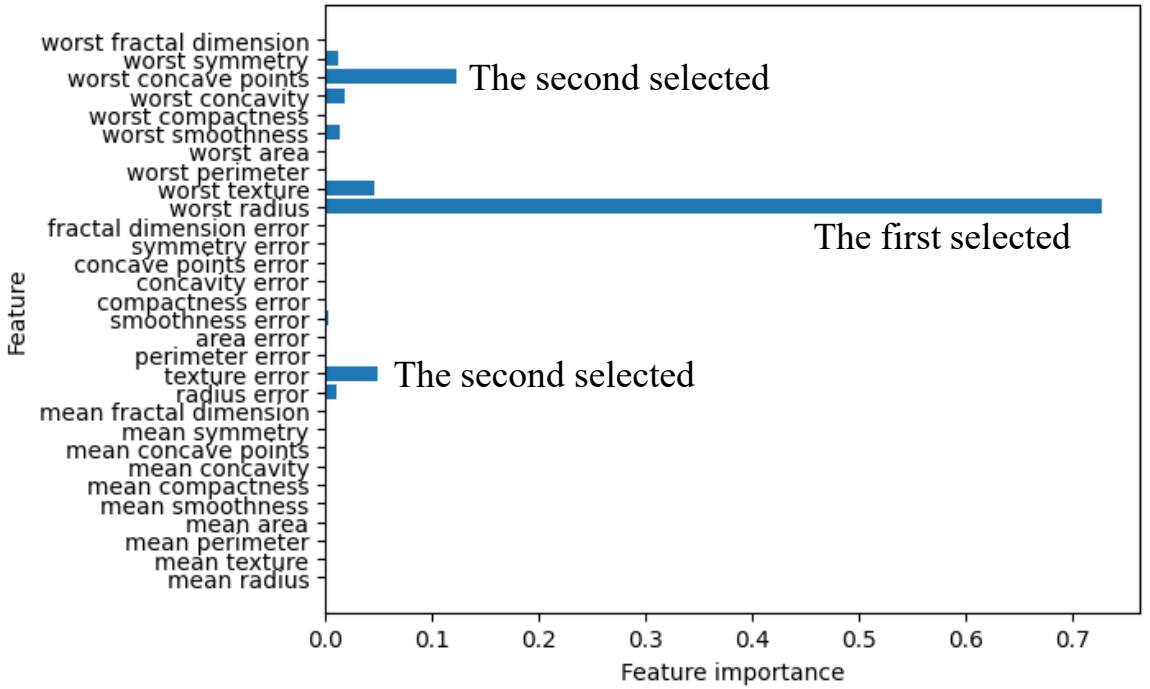
The attribute “worst radius” is the first selected



Step6: how to select the attributes/features

```
import matplotlib.pyplot as plt
import numpy as np
n_features=cancer.data.shape[1]
plt.barh(range(n_features),tree.feature_importances_,align='center')
plt.yticks(np.arange(n_features),cancer.feature_names)
plt.xlabel("Feature importance")
plt.ylabel("Feature")
```

This feature importance is correlated with the previous ***information gain***.



Step7: ensemble to random forest

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
tree=DecisionTreeClassifier(max_depth=4, random_state=0)
model=BaggingClassifier(tree, n_estimators=100, max_samples=0.8, random_state=1)
model.fit(X_train, y_train)
```

BaggingClassifier

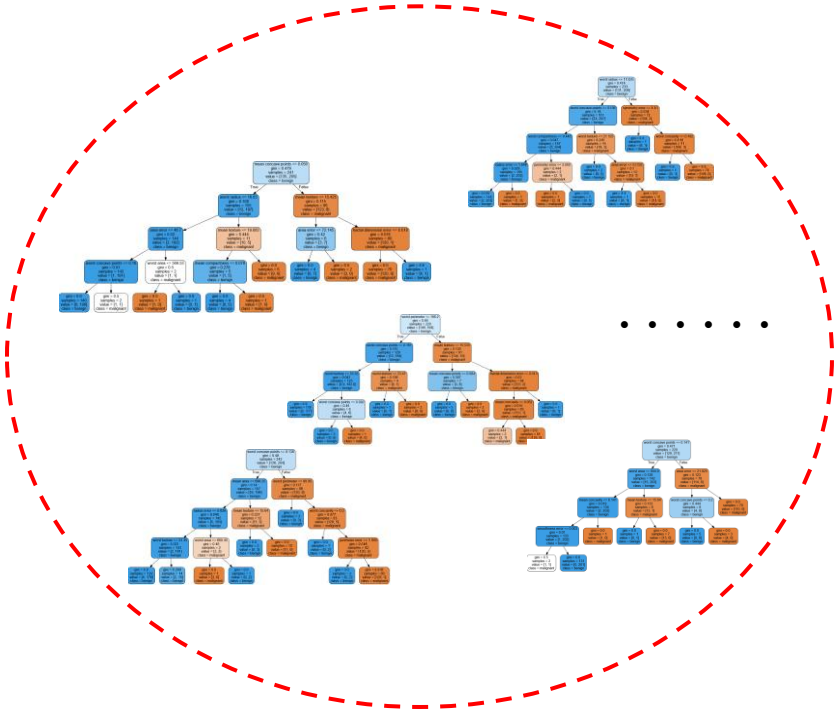
estimator: DecisionTreeClassifier

DecisionTreeClassifier

```
print("accuracy in the train set: {:.3f}".format(
      model.score(X_train, y_train)))
print("accuracy in the test set: {:.3f}".format(
      model.score(X_test, y_test)))
```

```
accuracy in the train set: 0.993
accuracy in the test set: 0.951
```

Training set and test set are also better!



A large number of trees formed a forest



The strength of an individual is limited,
but the power of **the collective** can achieve greatness!