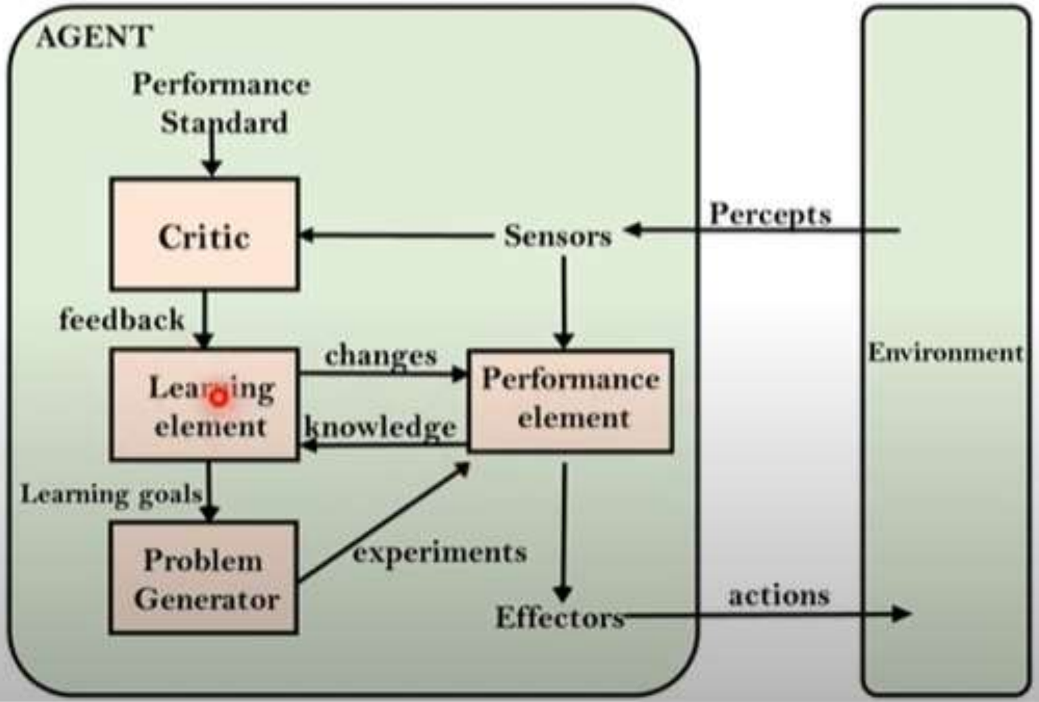
Learning

- Learning is Agent’s percepts should be used for acting,
- It also used for improving the agent's ability to act in the future.
- Learning takes place as the agent observes, its interactions with the world and its own decision-making processes.

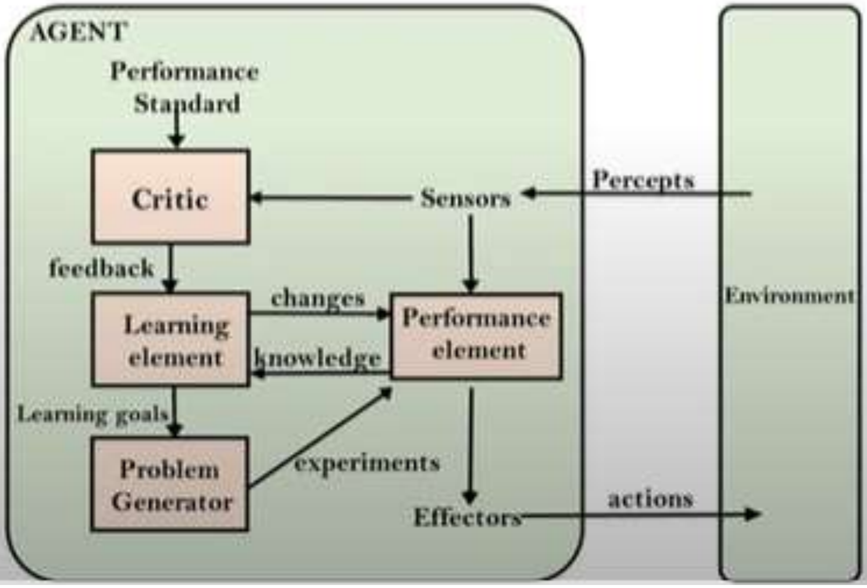
Forms of Learning

- Learning Agent can be thought of as containing a **Performance Element,** that decides, what actions to take, and a **learning element** that modifies the performance element so that it makes better decisions.



Learning Agent

- The design of a learning element is affected by three major issues:
  - Which *components* of the performance element are to be learned.
  - What *feedback* is available to learn these components.
  - What *representation* is used for the components.



Components of Learning Agents

- The components of learning agents include the following

1. A direct mapping from conditions on the current state to actions.
2. A means to infer relevant properties of the world from the percept sequence.
3. Information about the way the world evolves and about the results of possible actions the agent can take.
4. Utility information indicating the desirability of world states.
5. Action-value information indicating the desirability of actions.
6. Goals that describe classes of states whose achievement maximizes the agent's utility.

Automatic Taxi Driver

- Learning Agents' components can be learned from appropriate feedback.
- An agent training to become a taxi driver.
- Every time the instructor gives a command "Brake!", the agent can learn a condition-action rule, when we apply brake (component 1).
- By seeing many camera images, it can learn to recognize the objects on road (2).
- By trying actions and observing the results, for example, braking hard on a wet road, it can learn the effects of its actions (3).
- If there is no tip from passengers, but they shaken up during the trip, then it can learn a useful component of its overall utility function (4).
- The *type of feedback* available for learning is usually the most important factor in determining the nature of the learning problem that the agent faces.

Popular Machine Learning Algorithms

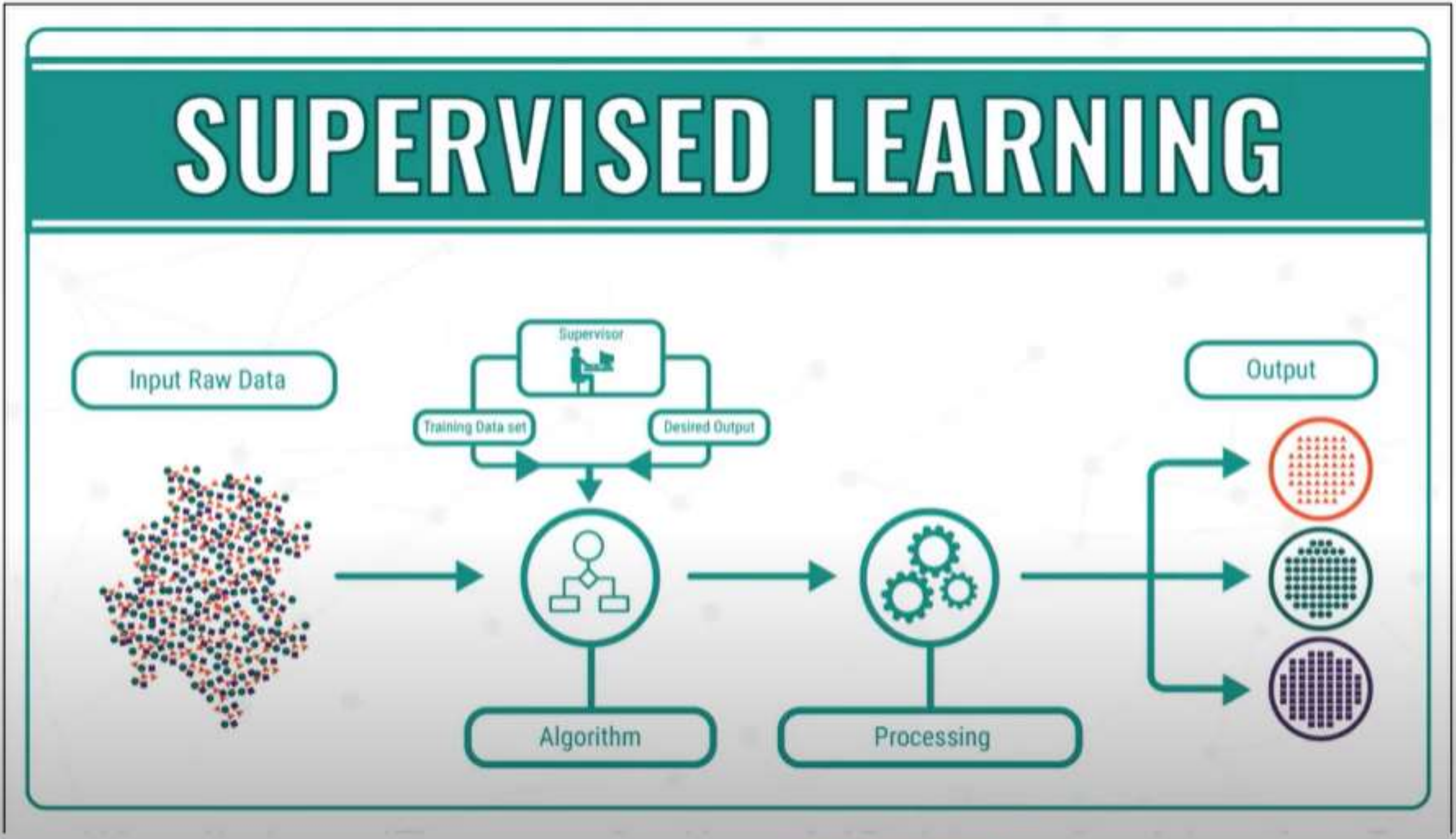
- The field of machine learning usually distinguishes three cases:
  - **Supervised Learning**
  - **Unsupervised Learning**
  - **Reinforcement learning**

Types of learning

- Any situation in which both the inputs and outputs of a component can be perceived is called **supervised learning**.
- In learning the condition-action component, the agent receives some evaluation of its action but is not told the correct action. This is called **reinforcement learning**;
- Learning when there is no hint at all about the correct outputs is called **unsupervised learning.**

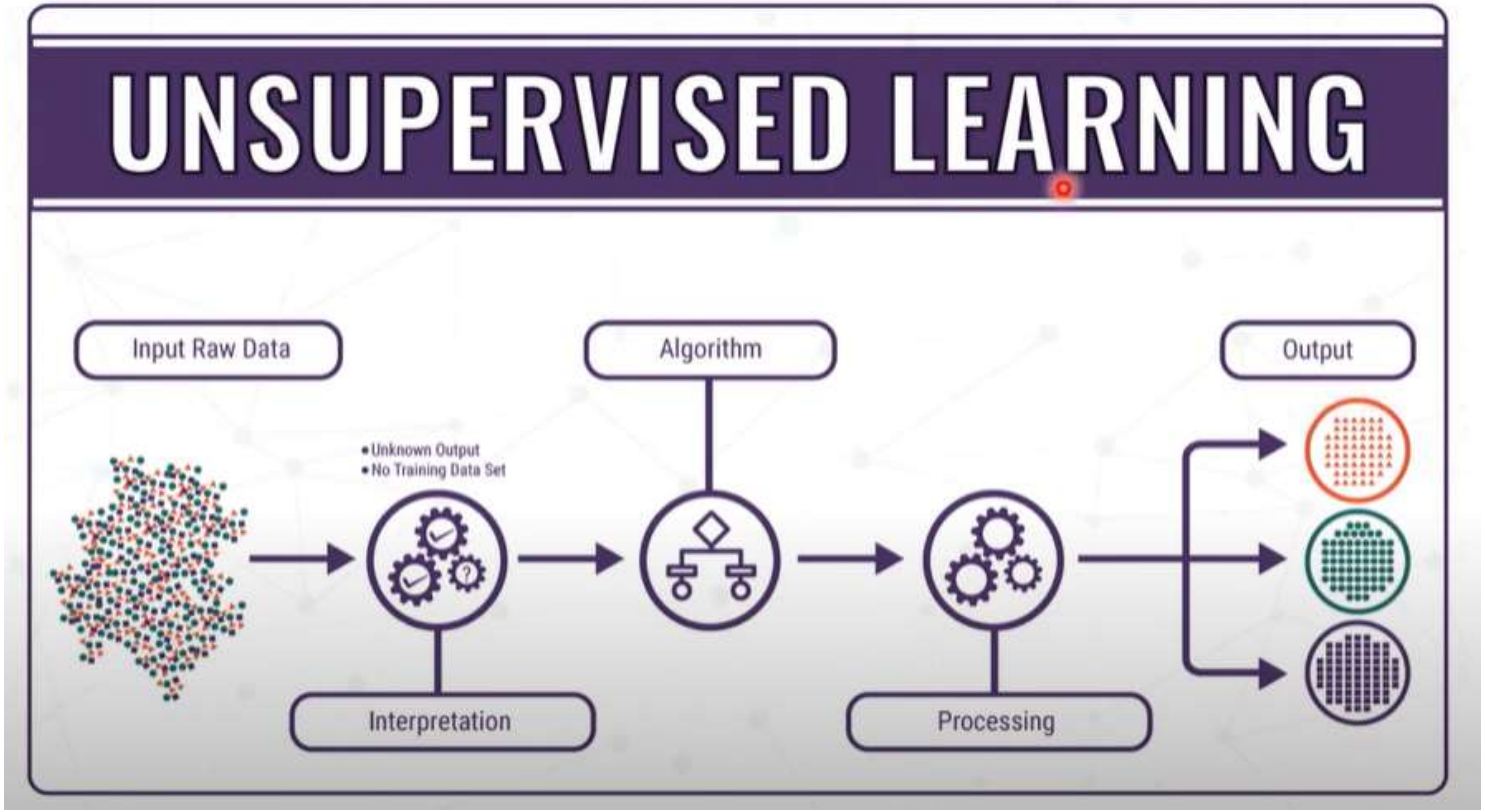
Supervised Learning

- **Supervised Learning -** the algorithm learns on a labeled dataset, providing an answer key that the algorithm can use to evaluate its accuracy on training data.


SUPERVISED LEARNING
Input Raw Data
Supervisor
Training Data set
Desired Output
Output
Algorithm
Processing

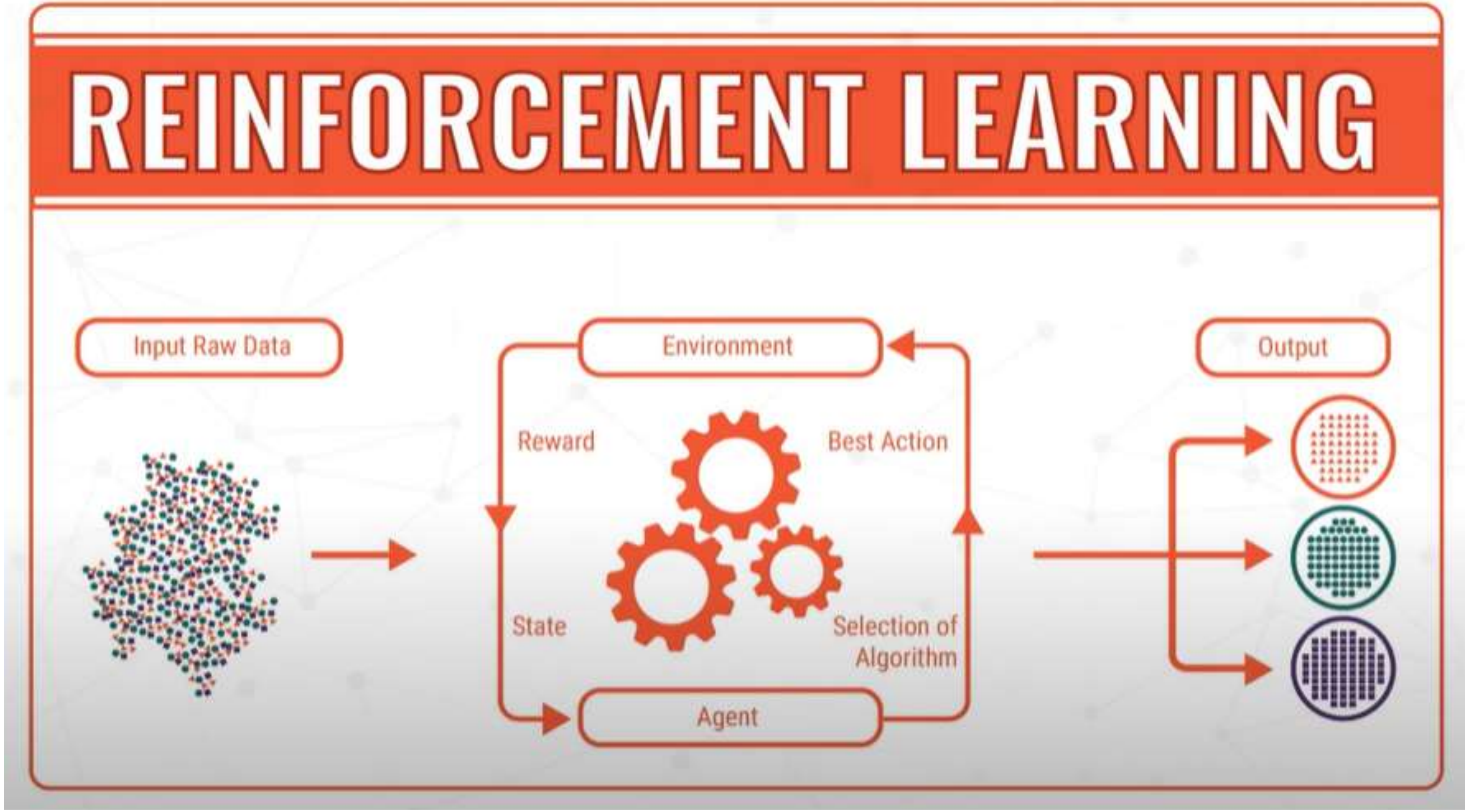
Unsupervised Learning

- **Unsupervised Learning -** provides unlabeled data, the algorithm tries to make sense of by extracting features and patterns on its own.


UNSUPERVISED LEARNING
Input Raw Data
Algorithm
Output
Unknown Output
No Training Data Set
Interpretation
Processing

Reinforcement learning

- **Reinforcement learning** - is a type of dynamic programming that trains algorithms using a system of reward and punishment.


REINFORCEMENT LEARNING
Input Raw Data
Environment
Output
Reward
Best Action
State
Selection of
Algorithm
Agent

Need for Inductive Learning

- There are basically two methods for knowledge extraction firstly from domain experts and then with machine learning.
- For a very large amount of data, the domain experts are not very useful and reliable.
- So we move towards the machine learning approach for this work.

Inductive Learning

- Also called as Deterministic Supervised Learning
- In this first input x, (the verified value) given to a function f, and the output is f(x).
- Then we can give different set of inputs (raw inputs) to the same function f, and verify the output f(x).
- By using the outputs we generate (learn) the rules.

- Inductive learning, also known as discovery learning, is a process where the learner discovers rules by observing examples.
- We can often work out rules for ourselves by observing examples. If there is a pattern; then record it.
- We then apply the rule in different situations to see if it works.
- With inductive language learning, tasks are designed specifically to guide the learner and assist them in discovering a rule.

- Inductive learning: system tries to make a “general rule” from a set of observed instances.
- Example:
- Mango -> f(Mango) - > sweet (e1)
- Banana -> f(Banana) -> sweet (e2)
- ...
- Fruits -> f(Fruits) -> sweet (general rule)
- Supervised learning:
  - Learning algorithm is given the correct value of the function for particular inputs, (verified output)
  - Then changes its representation of the function to try to match the information provided by the feedback.

Example

- Suppose an example set having attributes - Place type, weather, location, decision and seven examples.
- Our task is to generate a set of rules that under what condition what is the decision.

EXAMPLE NO.	PLACE TYPE	WEATHER	LOCATION	DECISION
I)	hilly	winter	kullu	Yes
II)	mountain	windy	Mumbai	No
III)	mountain	windy	Shimla	Yes
IV)	beach	windy	Mumbai	No
V)	beach	warm	goa	Yes
VI)	beach	windy	goa	No
VII)	beach	warm	Shimla	Yes

step 1

subset 1

S.NO	PLACE TYPE	WEATHER	LOCATION	DECISION
1	hilly	winter	kullu	Yes
2	mountain	windy	Shimla	Yes
3	beach	warm	goa	Yes
4	beach	warm	Shimla	Yes

subset 2

S.NO	PLACE TYPE	WEATHER	LOCATION	DECISION
5	mountain	windy	Mumbai	No
6	beach	windy	Mumbai	No
7	beach	windy	goa	No

- **step (2-8)**
- **at iteration 1**
- row 3 & 4 column weather is selected and row 3 & 4 are marked.
  the rule is added to R, IF weather is warm then a decision is yes.
- **at iteration 2**
- row 1 column place type is selected and row 1 is marked.
  the rule is added to R, IF place type is hilly then the decision is yes.
- **at iteration 3**
- row 2 column location is selected and row 2 is marked.
  the rule is added to R IF location is Shimla then the decision is yes.
- **at iteration 4**
- row 5&6 column location is selected and row 5&6 are marked.
  the rule is added to R, IF location is Mumbai then a decision is no.
- **at iteration 5**
- row 7 column place type & the weather is selected and row 7 is marked.
  rule is added to R IF place type is beach AND weather is windy then the decision is no.

- **finally we get the rule set :-**
- **Rule Set**
- **Rule 1:** IF the weather is warm THEN the decision is yes.
- **Rule 2:** IF place type is hilly THEN the decision is yes.
- **Rule 3:** IF location is Shimla THEN the decision is yes.
- **Rule 4:** IF location is Mumbai THEN the decision is no.
- **Rule 5:** IF place type is beach AND the weather is windy THEN the decision is no.

Text Book Example

- **example** is a pair *(x, f* (x)),
- where x is the input and *f(x)* is the output of the function applied to *x*.
- The task of **pure inductive inference** (or **induction)** is this:
  - Given a collection of examples of *f,* return a function *h* that approximates f .
  - Where The function h is called a **hypothesis.**
  - A good hypothesis will **generalize** well-that is, will predict unseen examples correctly.
- This is the fundamental **problem of induction.**

examples are *(x,* f *(x))* pairs...

- Fitting a function of a single variable to some data points.
- The examples are *(x,* f *(x))* pairs, where both x and f (x) are real numbers.
- We choose the **hypothesis space** H, which are the set of hypotheses we will consider,-to be the set of polynomials of degree at most ***k***.

examples are *(x, f (x))* pairs...

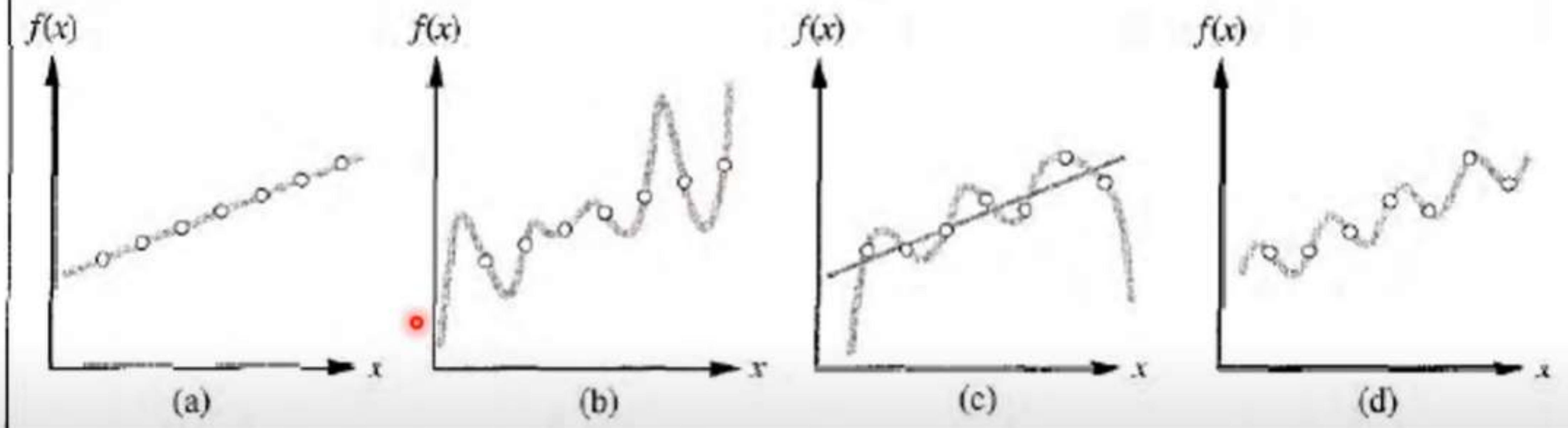


Figure 18.1 (a) Example $(\mathbf{x}, f(\mathbf{x}))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set that admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data set.

examples are *(x,* f *(x))* pairs…

- *There is a tradeoff between the expressiveness of a hypothesis space and the complexity of finding simple, consistent hypotheses within that space.*
- For example,
  - fitting straight lines to data is very easy;
  - fitting high-degree polynomials is harder; and
  - fitting Turing machines is very hard indeed because determining whether a given Turing machine is consistent with the data is not even decidable in general.
- Prefer a simple hypothesis spaces,
  - in which the resulting hypotheses may be simpler to use
  - It is faster to compute *h(x)* when h is a linear function than when it is an arbitrary Turing machine program.

Decision Tree

- A decision tree takes as input an object or situation described by a set of attributes and returns a "decision"-the predicted output value for the input.
- The input attributes can be discrete or continuous.
- we assume discrete inputs, Then output value can also be discrete or continuous;
- Learning a discrete-valued function is called classification learning;
- Learning a continuous function is called regression learning.
- We will concentrate on *Boolean* classification, wherein each example is classified as true (positive) or false (negative).

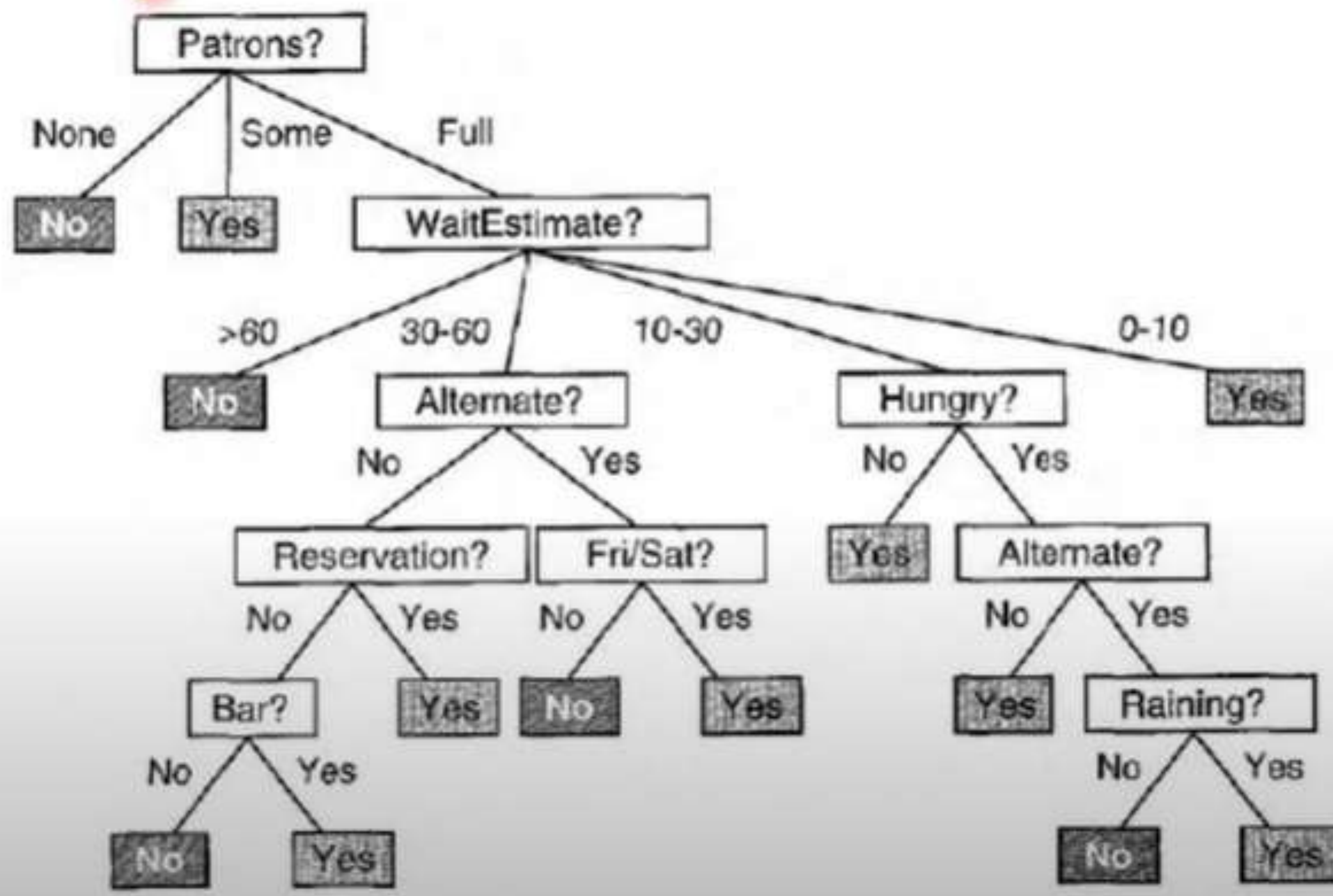
Decision Tree...

- A decision tree reaches its decision by performing a sequence of tests.
- Each internal node in the tree is to a test of the value of properties,
- The branches are labeled with the possible values of the test.
- Each leaf node specifies the value to be returned

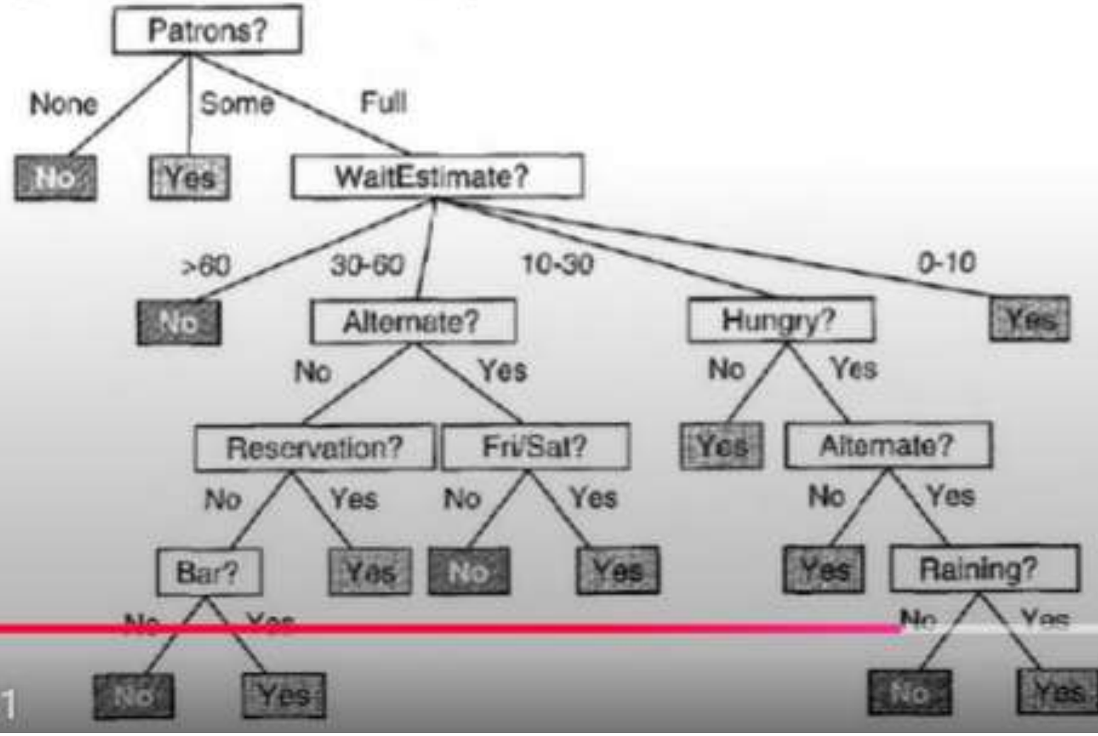
Example – Restaurant – wait for a table

- The aim is to learn the **goal predicate** WillWait.
- The attributes:
  - 1. Alternate: whether there is a suitable alternative restaurant nearby.
  - 2. Bar: whether the restaurant has a comfortable bar area to wait in.
  - 3. Fri/Sat: true on Fridays and Saturdays.
  - 4. *Hungry:* whether we are hungry.
  - 5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
  - 6. Price: the restaurant's price range ($, $$, $$$).
  - 7. Raining: whether it is raining outside.
  - 8. Reservation: whether we made a reservation.
  - 9. Type: the kind of restaurant (French, Italian, Thai, or burger).
  - 10. WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

Example – Restaurant – wait for a table…



- Notice that the tree does not use the Price and Type attributes, considering them to be irrelevant.
- Examples are processed by the tree starting at the root and following the appropriate branch until a leaf is reached.
- An example with Patrons = Full and WaitEstimate = 0-10 will be classified as positive (i.e., yes, we will wait for a table).



Choosing Attribute Tests

- The scheme used in decision tree learning for selecting attributes is designed to minimize the depth of the final tree.
- The idea is to pick the attribute that goes towards an exact classification of the examples.
- A perfect attribute divides the examples into sets that are all positive or all negative.
- The Patrons attribute is not perfect, but it is fairly good.

Assessing the performance of the learning algorithm

- A learning algorithm is good if it produces hypotheses, when it predict the classifications of unseen examples.
- we can assess the quality of a hypothesis by checking its predictions against the correct classification, is called as Test Set
- Then the learning algorithm will perform the following methodology
  - 1. Collect a large set of examples.
  - 2. Divide it into two disjoint sets: the **training set** and the **test set.**
  - **3.** Apply the learning algorithm to the training set, generating a hypothesis h.
  - 4. Measure the percentage of examples in the test set that are correctly classified by h.
  - 5. Repeat steps 2 to 4 for different sizes of training sets and different randomly selected training sets of each size.

Assessing the performance of the learning algorithm…

- The result of this procedure is a set of data, that can be processed to give the average prediction quality as a function of size of training set.
- This function can be plotted on a graph, is called the **learning curve** for the algorithm on the particular domain

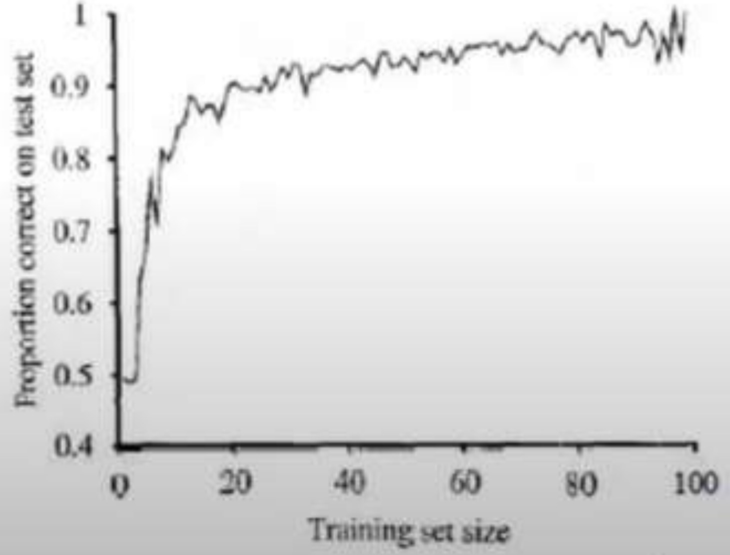


Figure 18.7 A learning curve for the decision tree algorithm on 100 randomly generated examples in the restaurant domain. The graph summarizes 20 trials.

Problems in Decision Trees

- Missing data
- Multivalued attributes
- Continuous and integer-valued input attributes
- Continuous-valued output attributes

Ensemble Learning in Machine Learning

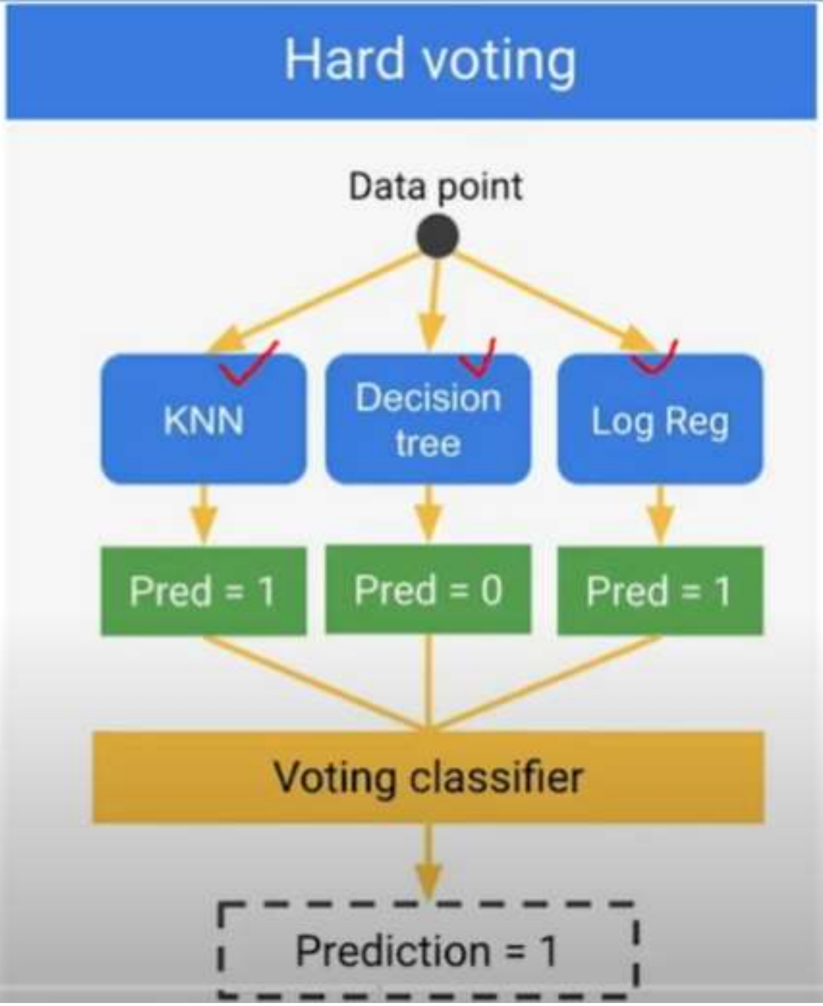
- Ensemble learning is a supervised learning technique used in machine learning to improve overall performance by combining the predictions from multiple models.

Ensemble Learning - Types of Ensemble Methods

- Voting (Averaging)
- Bootstrap aggregation (bagging)
- Random Forests
- Boosting
- Stacked Generalization (Blending)

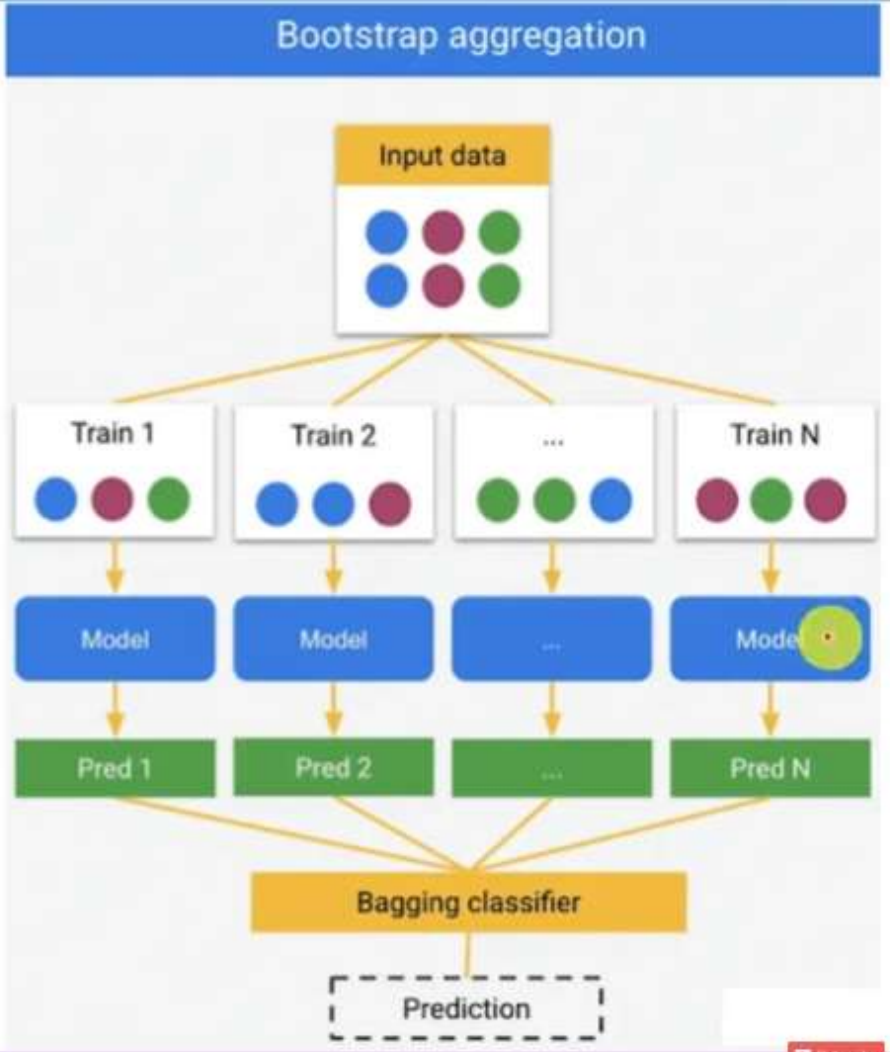
Ensemble Learning – Voting (Averaging)

- Voting is an ensemble machine learning algorithm that involves making a prediction that is the average (regression) or the sum (classification) of multiple machine learning models.



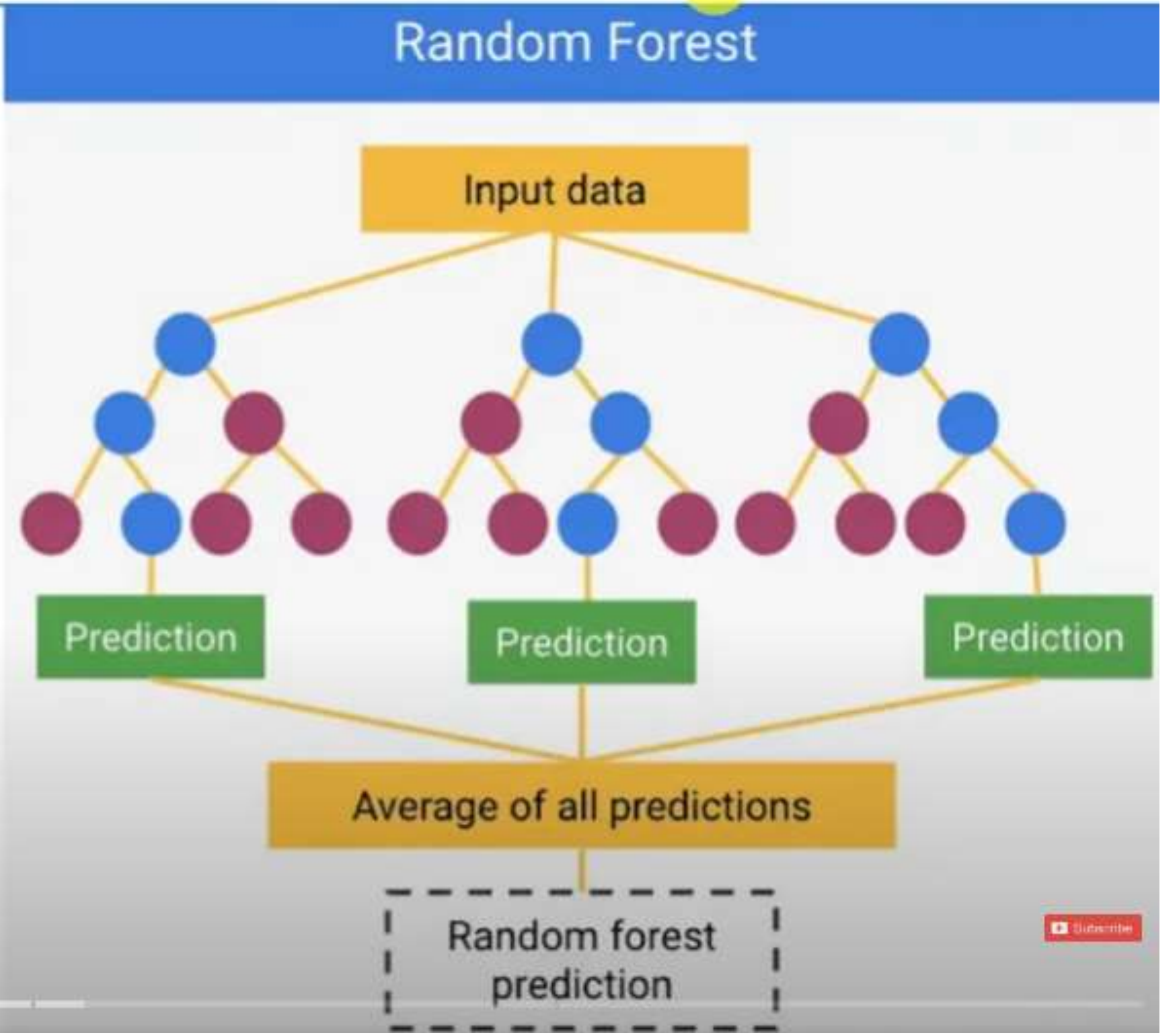
Ensemble Learning – Bootstrap aggregation (bagging)

- Bootstrap Aggregating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms like classification and regression.
- It decreases the variance and helps to avoid overfitting.
- It is usually applied to decision tree methods.
- Bagging is a special case of the model averaging approach.



Ensemble Learning – Random Forest

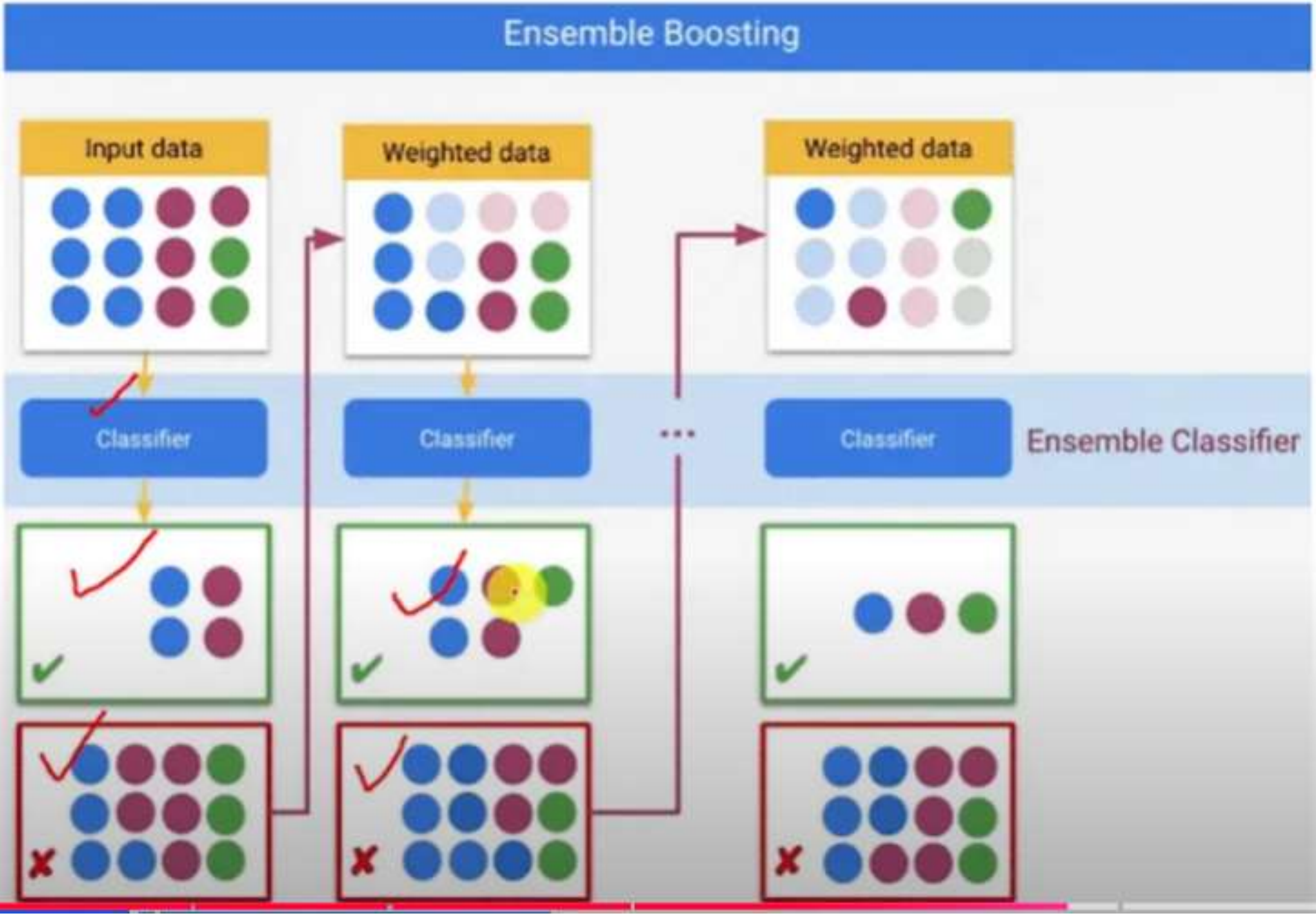
- Random forest is a commonly-used machine learning algorithm.
- A random forest is an ensemble learning method where multiple decision trees are constructed and then they are merged to get a more accurate prediction.



Ensemble Learning – Boosting

- Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers.
- It is done by building a model by using weak models in series.
- Firstly, a model is built from the training data.
- Then the second model is built which tries to correct the errors present in the first model.
- This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models is added.

Ensemble Learning – Boosting

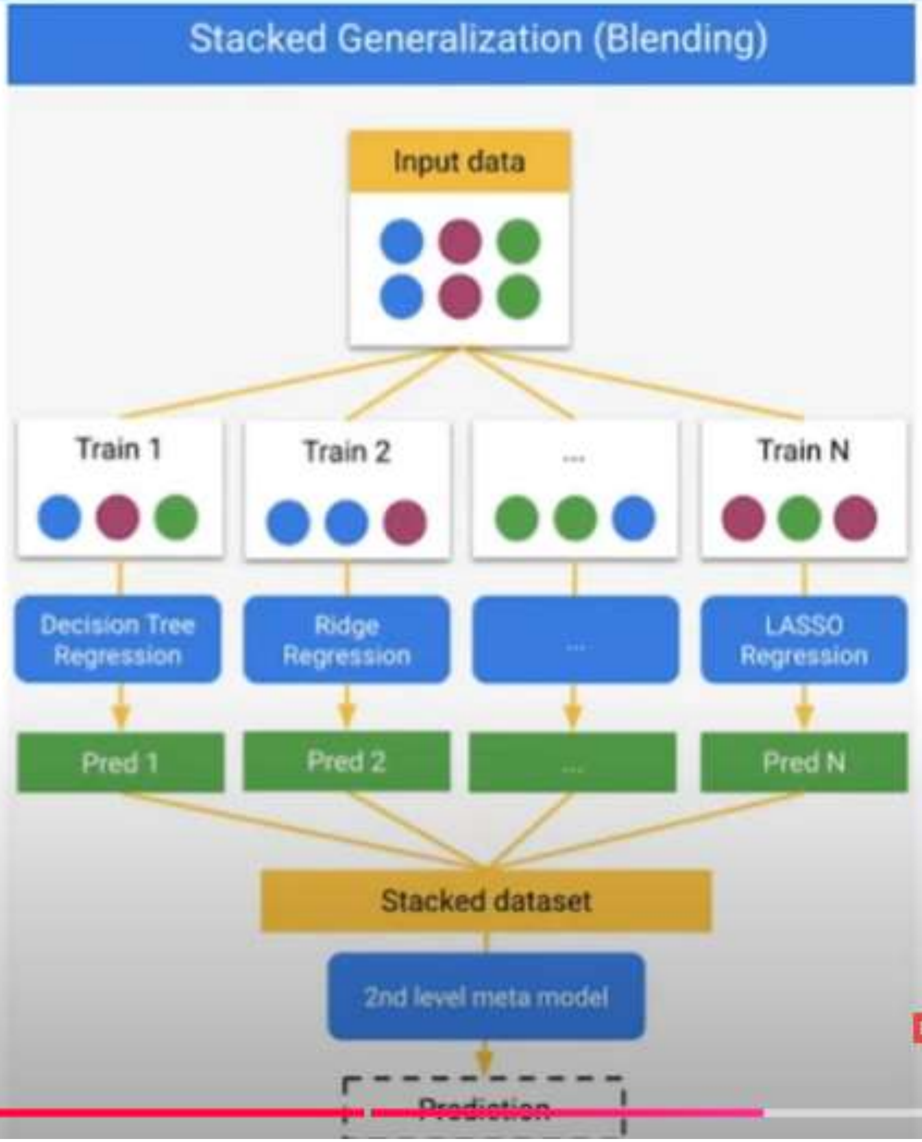


Ensemble Learning – Stacked Generalization (Blending)

- Stacking, Blending and and Stacked Generalization are all the same thing with different names. It is a kind of ensemble learning.
- In traditional ensemble learning, we have multiple classifiers trying to fit to a training set to approximate the target function.
- Since each classifier will have its own output, we will need to find a combining mechanism to combine the results.
- This can be through voting (majority wins), weighted voting (some classifier has more authority than the others), averaging the results, etc.
- This is the traditional way of ensemble learning.

Ensemble Learning – Stacked Generalization (Blending)

- In stacking, the combining mechanism is that the output of the classifiers (Level 0 classifiers) will be used as training data for another classifier (Level 1 classifier) to approximate the same target function.
- Basically, you let the Level 1 classifier to figure out the combining mechanism.



Iterative Deepening Search (IDS)

- The Iterative Deepening Depth First Search is simply called as iterative deepening search (IDS)
- Combine the benefits of depth-first and breadth-first search to finds the best solution.
- It gradually increase the limit from 0,1,2 and so on until reaches the goal.
- It will terminate when the depth limit reaches d, depth of the shallowest goal node, with success message.

Iterative Deepening Search (IDS)…

- May seem wasteful because states are generated multiple times
- but actually not very costly, because nodes at the bottom level are generated only once.
- The overhead of these multiple expansions is small, because most of the nodes are towards leaves (bottom) of the search tree:
  - thus, the nodes that are evaluated several times (towards top of tree) are in relatively small number.
- Iterative depending is the preferred uninformed search method when the search space is large and the depth of the solution is unknown

Performance measure of IDS

- Combines the best of breadth-first and depth-first search strategies
- Completeness: Yes
- Time complexity: $O(b^d)$

  b – branching factor
  d – depth
- Space complexity: $O(bd)$
- Optimality: Yes, if step cost = 1