AIML_Module2

Topics:

1. Informed Search Strategies:
    a. **Greedy best-first search,**
    b. **A*search,**
    c. **Heuristic functions**.
2. Introduction to Machine Learning,
3. Understanding Data

2. 1 Informed Search Strategies

Informed Search: Informed search is a search strategy that utilizes problem-specific knowledge, to find solutions more efficiently. Informed search methods make use of heuristics and evaluation functions to guide the search towards more promising paths.

Heuristic Function(h(n)): It is a heuristic function that provides an estimate of the cost from the current node to the goal node. This heuristic is admissible if it never overestimates the true cost to reach the goal. In other words, **h(n)** is always less than or equal to the actual cost.

Actual cost function(g(n)): It is Cost of the path from the **start node to node n**. It represents the actual cost incurred to reach the current node from the initial node. For the initial node (start node), **g(n)** is usually 0.

Evaluation Function (f(n)): The evaluation function, denoted as **f(n),** is the total estimated cost of the cheapest path from the start node to the goal node that passes through node n. It is the sum of g(n) and h(n): **f(n) = g(n) + h(n).**

f(n) represents the priority of a node. Nodes with lower f(n) values are explored first, making the algorithm prioritize paths that are likely to be more efficient.

2.1.a Greedy Best First Search Algorithm

Greedy best first search tries to expand the node that is closest to the goal to lead to a solution quickly. It evaluates the nodes by using just the heuristic function i.e. for Greedy best first search, f(n) = h(n).

Let's explore the application of this method to route-finding challenges in Romania for the map given in figure 3.2.

We will employ the straight-line distance heuristic, denoted as h_{SLD}. Specifically, for our destination in **Bucharest**, we require knowledge of the straight-line distances to Bucharest, as illustrated in Figure 3.22.

As an illustration, consider h_{SLD}(In(Arad)) = 366. It's important to note that the values of h_{SLD} cannot be derived directly from the problem description. Furthermore, understanding that h_{SLD} correlates with actual road distances and serves as a valuable heuristic requires a certain level of experience.

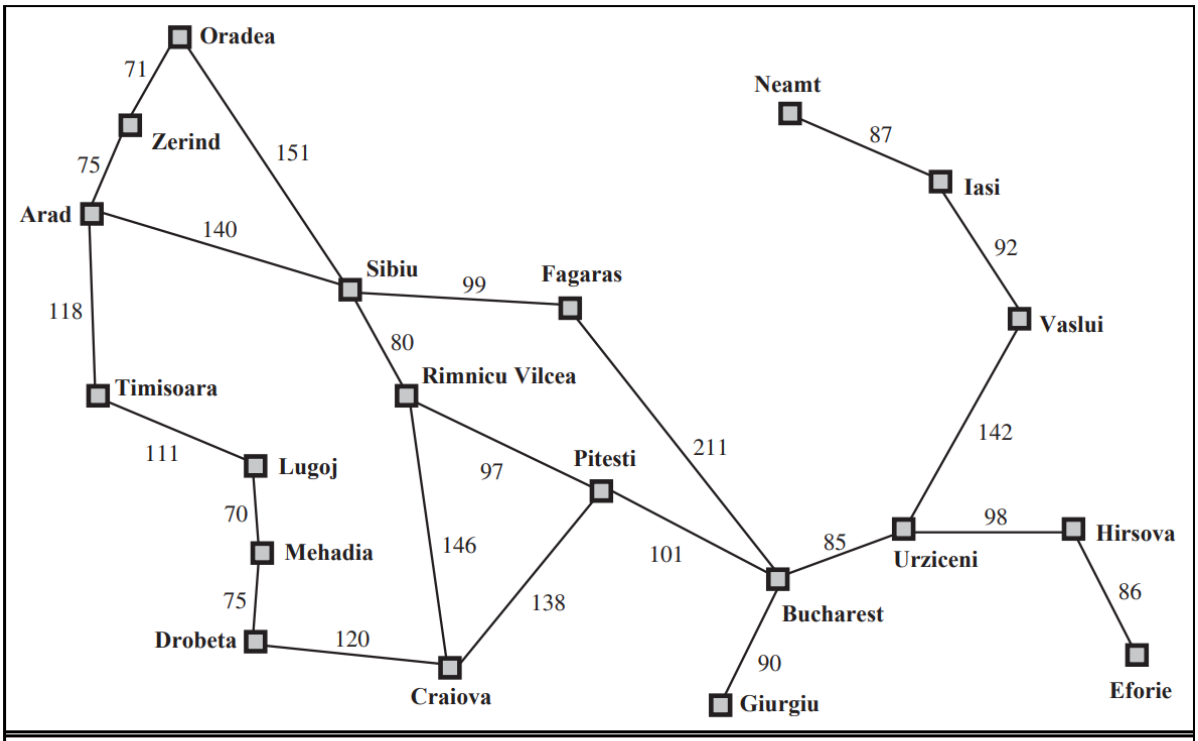


Figure 3.2 A simplified road map of part of Romania.

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Figure 3.22 Values of h_{SLD}—straight-line distances to Bucharest.

Following Figure shows the progress of a greedy best-first search using h_{SLD} to find a path from Arad to Bucharest.

The first node to be expanded from Arad will be Sibiu because it is closer to Bucharest than either Zerind or Timisoara. The next node to be expanded will be Fagaras because it is closest. Fagaras in turn generates Bucharest, which is the goal.

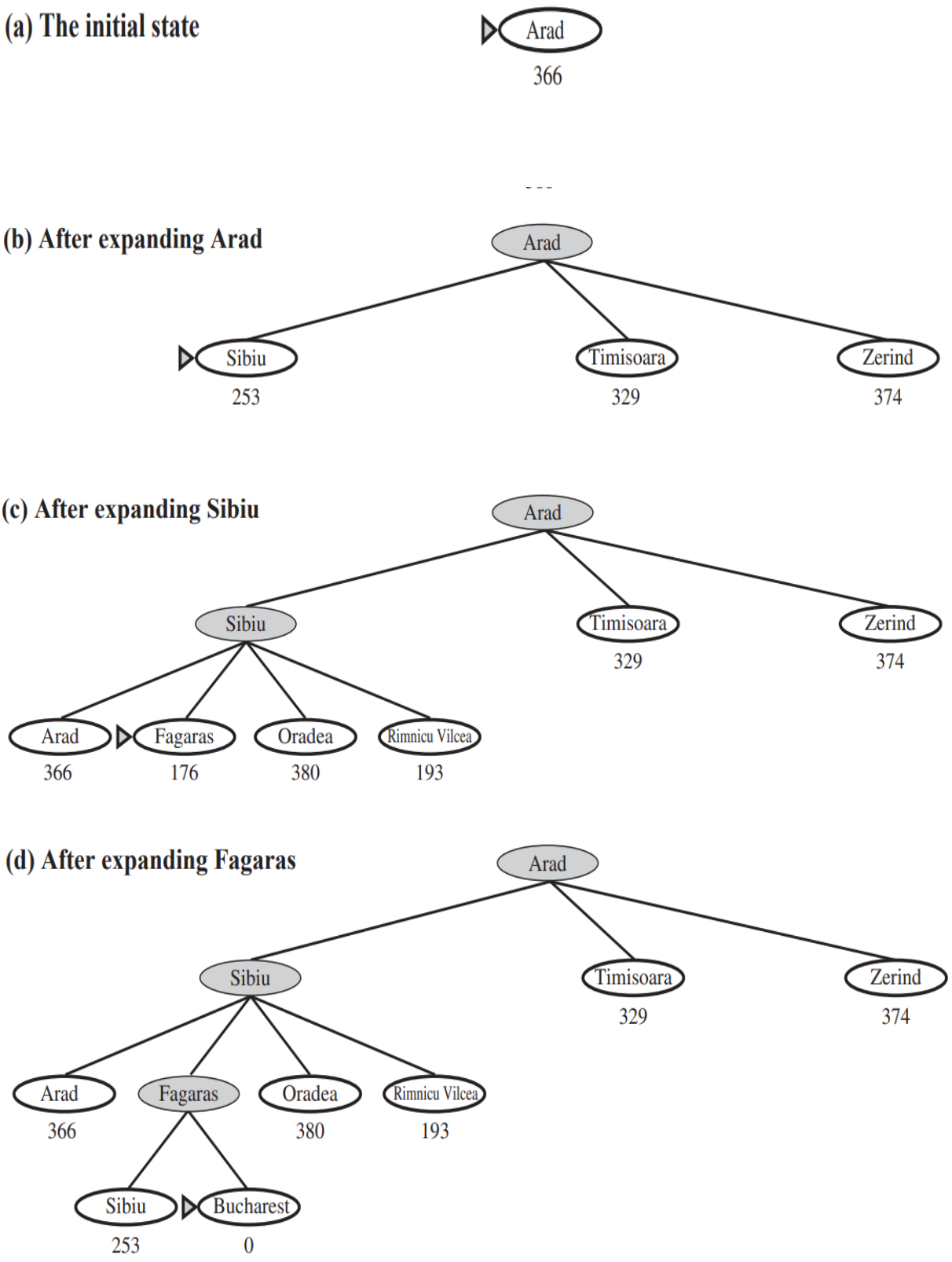
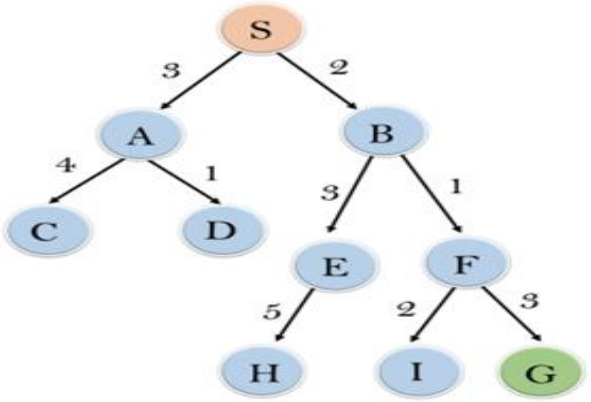


Figure: Stages in a greedy best-first tree search for Bucharest with the straight-line distance heuristic h_{SLD} . Nodes are labelled with their h-values.

Best first search algorithm:

Step 1: Place the starting node into the OPEN list.
Step 2: If the OPEN list is empty, Stop and return failure.
Step 3: Remove the node n, from the OPEN list which has the lowest value of h(n), and places it in the CLOSED list.
Step 4: Expand the node n, and generate the successors of node n.
Step 5: Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.
Step 6: For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.
Step 7: Return to Step 2.

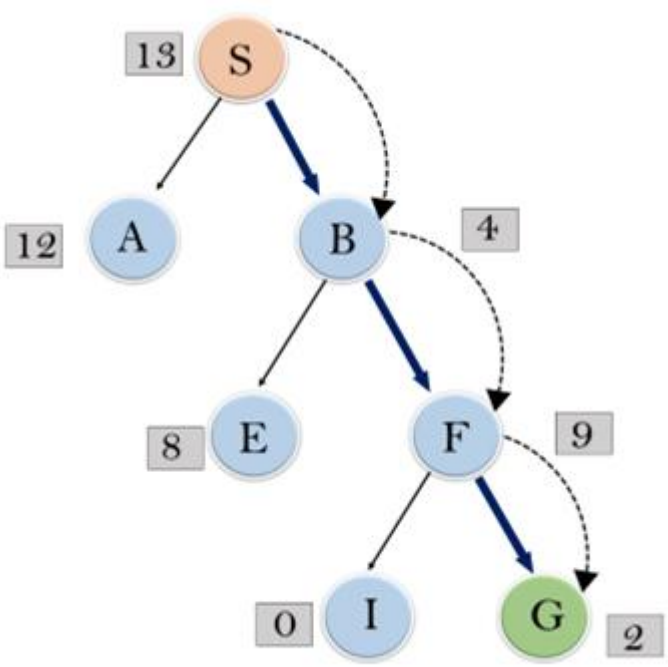
Example: Consider the tree and heuristic values given below:



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

In this search example, we are using two lists which are **OPEN** and **CLOSED** Lists. Following are the iteration for traversing the above example.

Step	OPEN List	CLOSED List	Details
Initialization	[A, B]	[S]	
Iteration1: Expand B	[A]	[S,B]	h(B)<h(A)
Iteration2: Expand F	[E,F,A] [E,A]	[S,B] [S,B,F]	H(F)<H(E),H(A)
Iteration3: Visit Goal G	[I,G,E,A] [I,E,A]	[S,B,F] [S,B,F,G]	H(G)< H(E),H(A),H(I)



Hence the final solution path will be: **S----> B----->F----> G**

Time Complexity: The worst case time complexity of Greedy best first search is $O(b^m)$.

Space Complexity: The worst case space complexity of Greedy best first search is $O(b^m)$. Where, m is the maximum depth of the search space.

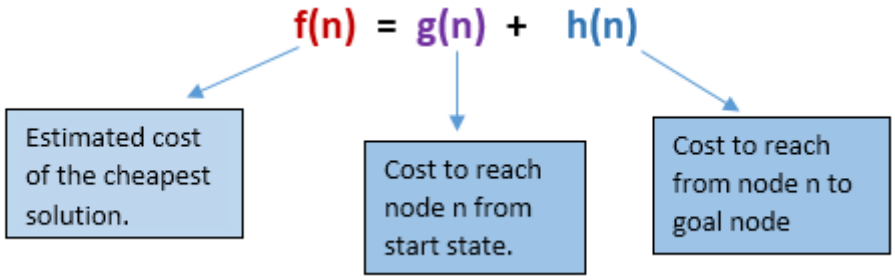
Complete: Greedy best-first search is also incomplete, even if the given state space is finite.

Optimal: Greedy best first search algorithm is not optimal.

2.1.b A* Search Algorithm

A* search is the most commonly known form of best-first search. It uses heuristic function h(n), and cost to reach the node n from the start state g(n). A* search algorithm finds the shortest path through the search space using the heuristic function. This search algorithm expands less search tree and provides optimal result faster.

In A* search algorithm, we use search heuristic as well as the cost to reach the node. Hence, we can combine both costs as following, and this sum is called as a **fitness number**.



Algorithm of A* search:

Step1: Place the starting node in the OPEN list.

Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.

Step 3: Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node then return success and stop, otherwise

Step 4: Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into Open list.

Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.

Step 6: Return to Step 2.

Advantages:

- A* search algorithm is the best algorithm than other search algorithms.
- A* search algorithm is optimal and complete.
- This algorithm can solve very complex problems.

Disadvantages:

- It does not always produce the shortest path as it mostly based on heuristics and approximation.
- A* search algorithm has some complexity issues.
- The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

Example1: Let's explore the application of this method to route-finding challenges in Romania for the map given in figure 3.2 .

We will employ the straight-line distance heuristic, denoted as h_{SLD}. Specifically, for our destination in Bucharest, we require knowledge of the straight-line distances to Bucharest, as illustrated in Figure 3.22.

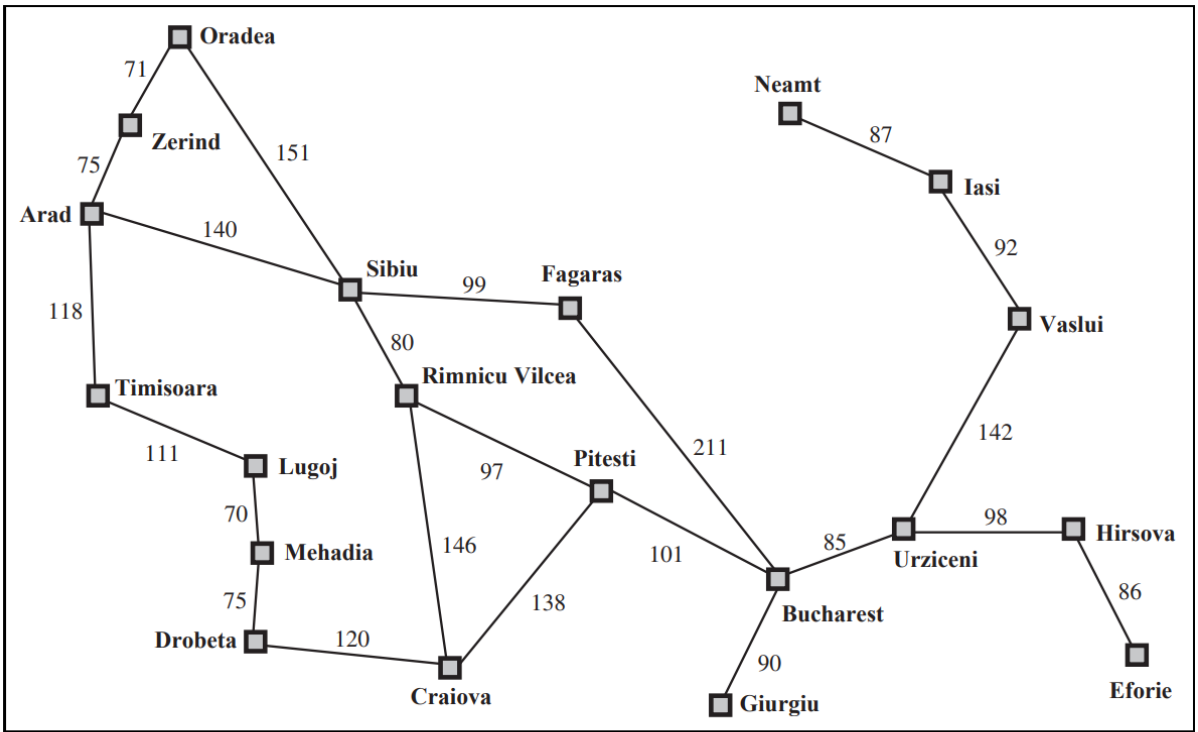
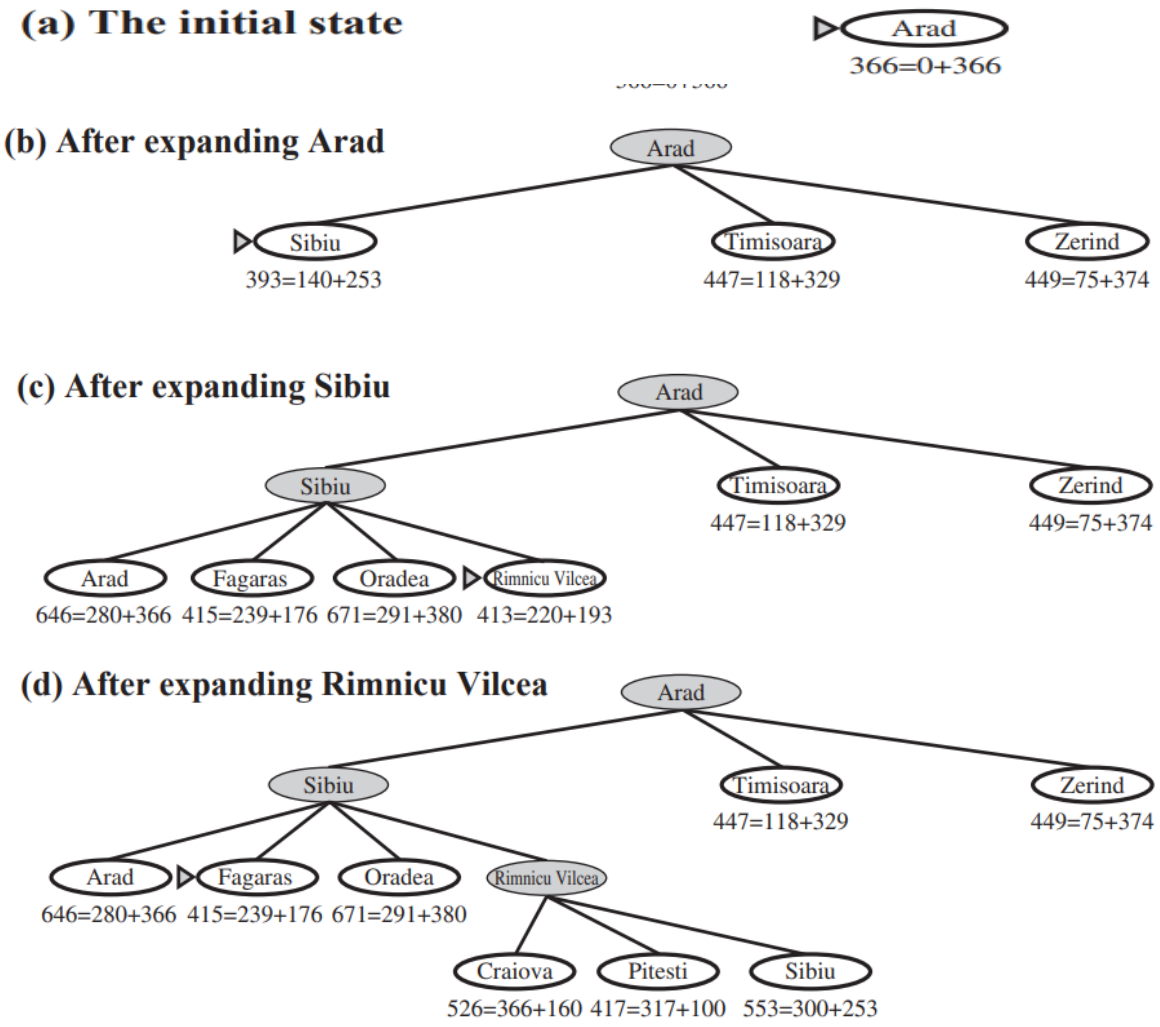


Figure 3.2 A simplified road map of part of Romania.

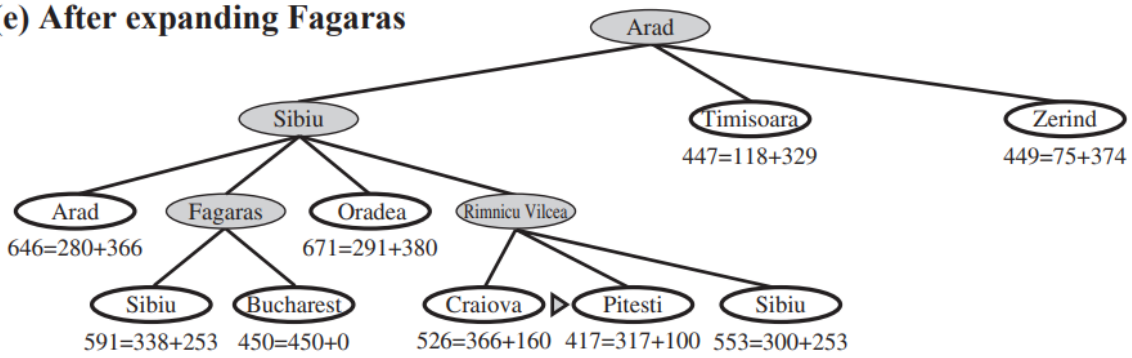
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Figure 3.22 Values of h_{SLD}—straight-line distances to Bucharest.

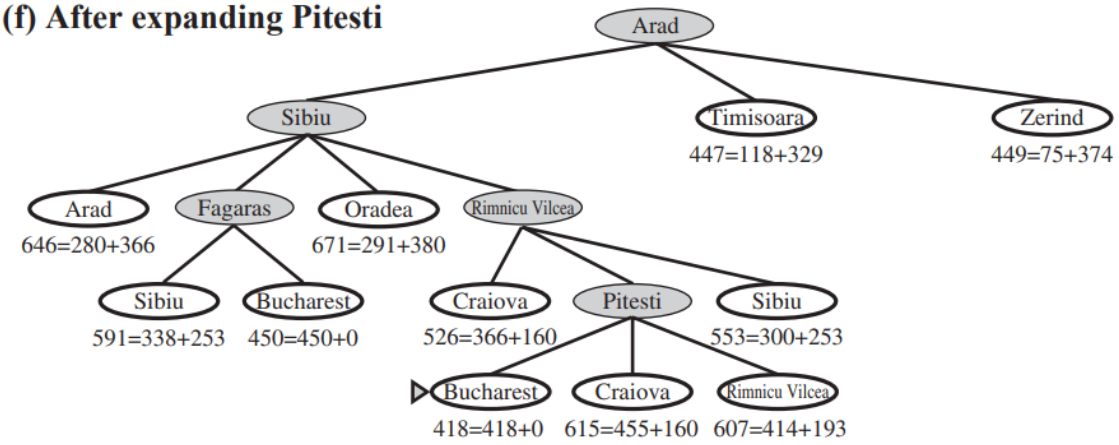
Figure below illustrates the Stages in **an A∗ search for Bucharest**. Nodes are labelled with f = g + h. The h values are the straight-line distances to Bucharest



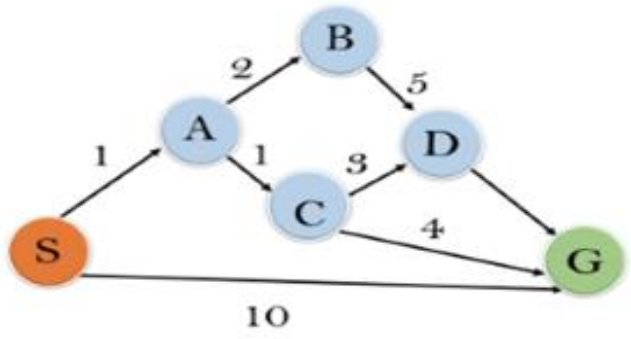
(e) After expanding Fagaras



(f) After expanding Pitesti

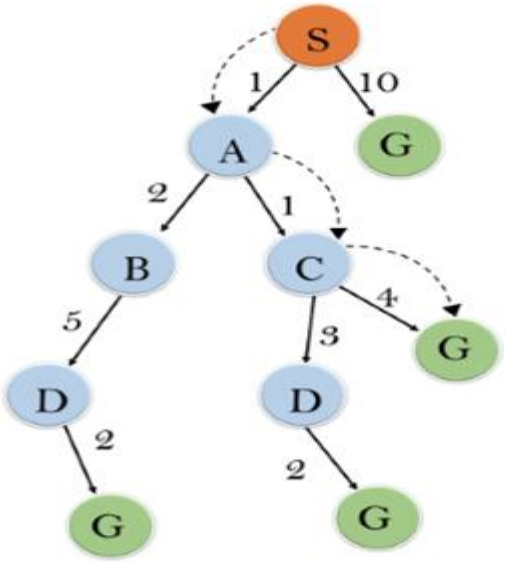


Example 2: In this example, we will traverse the given graph using the A* algorithm. The heuristic value of all states is given in the below table so we will calculate the f(n) of each state using the formula f(n)= g(n) + h(n), where g(n) is the cost to reach any node from start state. Here we will use OPEN and CLOSED list**.**



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

Solution:



Initialization: {(S, 5)}
Iteration1: {(S--> A, 4), (S-->G, 10)}
Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}
Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}
Iteration 4 will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.

2.1.b.1 Points to remember:

A* Algorithm and First-Occurrence Path: The A* algorithm is designed to return the first path it finds from the start node to the goal node. Once a valid path is discovered, A* terminates its search, making it more efficient in scenarios where finding a single optimal solution is sufficient.

Quality of Heuristic: The effectiveness of the A* algorithm is significantly influenced by the quality of the heuristic function **h(n))**. A well-designed heuristic provides a good estimate of the remaining cost from a given node to the goal, guiding **A*** to explore more promising paths and enhancing its efficiency.

Node Expansion Condition: A* algorithm expands nodes based on the evaluation function f(n)=g(n)+h(n), where:
g(n) is the cost of the path from the start node to node n,
h(n) is the heuristic estimate of the cost from node n to the goal node.

The algorithm expands nodes that satisfy the condition f(n)≤M, where M is a specified threshold or maximum cost. This condition ensures that A* explores nodes within a predefined cost limit, allowing for efficient pathfinding without exhaustively searching the entire space.

Complete: A* algorithm is complete as long as:

- Branching factor is finite.
- Cost at every action is fixed.

2.1.b.2 Conditions for optimality: Admissibility and consistency:

A* search algorithm is optimal if it follows below two conditions:

Admissible:

The first condition requires for optimality is that **h(n)** should be an admissible heuristic for **A* tree** search. An admissible heuristic is one that never overestimates the cost to reach the goal. Because g(n) is the actual cost to reach n along the current path, and f(n) = g(n) + h(n), we have as an immediate consequence that f(n) never overestimates the true cost of a solution along the current path through n. If the heuristic function is admissible, then A* tree search will always find the least cost path.

Consistency (or sometimes monotonicity):

Second required condition is consistency for only **A* graph**-search. A heuristic h(n) is consistent if, for every node n and every successor n ′ of n generated by any action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n ′ plus the estimated cost of reaching the goal from n ′ : $h(n) \leq c(n, a, n') + h(n')$.

This is a form of the general **triangle inequality**, which stipulates that each side of a triangle cannot be longer than the sum of the other two sides. Here, the triangle is formed by n, n ′ , and the goal **Gn** closest to n.

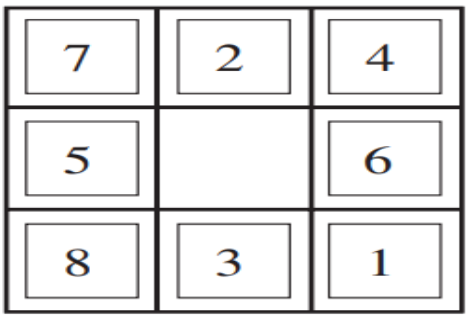
Time Complexity: The time complexity of A* search algorithm depends on heuristic function, and the number of nodes expanded is exponential to the depth of solution d. So the time complexity is $O(b^d)$, where b is the branching factor.

Space Complexity: The space complexity of A* search algorithm is **$O(b^d)$**

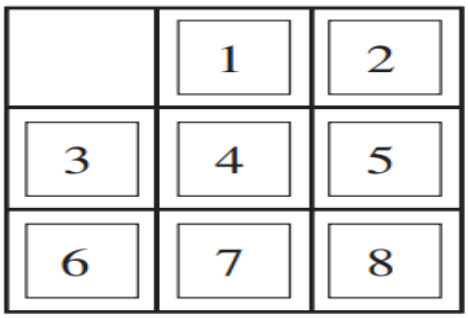
2.1.c Heuristics Functions

Heuristic Functions **h(n)** guide search algorithms by estimating the cost or distance to a goal state from the current state (n).

Consider the 8-puzzle game. The object of the 8 puzzle is to slide the title horizontally or vertically into the empty space until the configuration matches the goal configuration.



Start State



Goal State

Figure illustrates the A typical instance of the 8-puzzle. The solution is 26 steps long.

The average solution cost for a randomly generated 8-puzzle instance is about 22 steps. The branching factor is about 3. (When the empty tile is in the middle, four moves are possible; when it is in a corner, two; and when it is along an edge, three.) This means that an exhaustive tree search to depth 22 would look at about $3^{22} \approx 3.1 \times 10^{10}$ states.

The two commonly used candidates for 8 puzzles are as follows:

h1 = the number of misplaced tiles. For Figure, all of the eight tiles are out of position, so the start state would have h1 = 8. h1 is an admissible heuristic because it is clear that any tile that is out of place must be moved at least once

h2 = the sum of the distances of the tiles from their goal positions. Because tiles cannot move along diagonals, the distance we will count is the sum of the horizontal and vertical distances. This is sometimes called the city block distance or Manhattan distance. h2 is also admissible because all any move can do is move one tile one step closer to the goal. Tiles 1 to 8 in the start state give a Manhattan distance of h2 = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18 .

As expected, neither of these overestimates the true solution cost, which is 26. The performance of heuristic search algorithms depends on the quality of the heuristic function. One can sometimes construct good heuristics by relaxing the problem definition, by storing precomputed solution costs for subproblems in a pattern database, or by learning from experience with the problem class.

1. The effect of heuristic accuracy on performance:

Experimentally it is determined that h_2 is better than h_1. That is for any node n, $h_2(n) \geq h_1(n)$. This implies that h_2 dominate h_1. Domination translates directly into efficiency. A^* using h_2 will never expand more nodes than A^* using h_1.

2. Generating admissible heuristics from relaxed problems:

A problem with fewer restrictions on the actions is called a relaxed problem. The state-space graph of the relaxed problem is a super graph of the original state space because the removal of restrictions creates added edges in the graph.
Because the relaxed problem adds edges to the state space, any optimal solution in the original problem is, by definition, also a solution in the relaxed problem; but the relaxed problem may have better solutions if the added edges provide short cuts. Hence, the cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem.

For example, if the 8-puzzle actions are described as

- A tile can move from square **A** to square **B** if
- **A** is horizontally or vertically adjacent to **B** and **B** is blank,

we can generate three relaxed problems by removing one or both of the conditions:

a) A tile can move from square A to square B if A is adjacent to B.
b) A tile can move from square A to square B if B is blank.
c) A tile can move from square A to square B.

If a collection of admissible heuristics h1 . . . hm is available for a problem and none of them dominates any of the others, which should we choose? As it turns out, we need not make a choice. We can have the best of all worlds, by defining $h(n) = \max\{h1(n), \ldots, hm(n)\}$

3.Generating admissible heuristics from subproblems: Pattern databases:

Admissible heuristics can also be derived from the solution cost of a subproblem of a given problem. For example, Figure below shows a subproblem of the 8-puzzle instance.

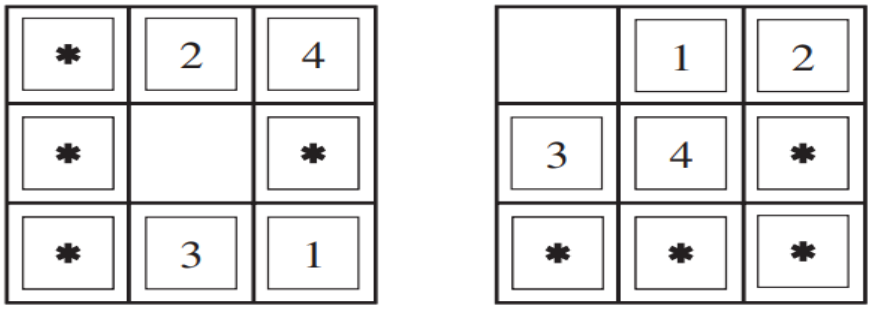


Fig: A subproblem of the 8-puzzle instance. The task is to get tiles 1, 2, 3, and 4 into their correct positions, without worrying about what happens to the other tiles.

The subproblem involves getting tiles 1, 2, 3, 4 into their correct positions. Clearly, the cost of the optimal solution of this subproblem is a lower bound on the cost of the complete problem. It turns out to be more accurate than Manhattan distance in some case.

The idea behind pattern databases is to store these exact solution costs for every possible subproblem instance—in our example, every possible configuration of the four tiles and the blank. (The locations of the other four tiles are irrelevant for the purposes of solving the subproblem, but moves of those tiles do count toward the cost.) Then we compute an admissible heuristic h_{DB} for each complete state encountered during a search simply by looking up the corresponding subproblem configuration in the database.

4 Learning heuristics from experience:

A heuristic function, denoted as h(n), aims to approximate the solution cost starting from the state represented by node **n**. One of the strategies is to learning from practical experiences. In this context, "experience" refers to solving numerous instances of problems like 8-puzzles.

In each optimal solution to an 8-puzzle, valuable examples emerge, comprising a state along the solution path and the actual cost of reaching the solution from that particular point.

Employing these examples, a learning algorithm can be employed to generate a heuristic function, **h(n),** with the potential to predict solution costs for other states encountered during the search process.
Inductive learning methods are most effective when provided with relevant features of a state for predicting its value, rather than relying solely on the raw state description.

For instance, a feature like "**number of misplaced tiles**" (**$x_1(n)$**) can be useful in predicting the distance of a state from the goal in an 8-puzzle. By gathering statistics from randomly generated 8-puzzle configurations and their actual solution costs, one can use these features to predict h(n).

Multiple features, such as **$x_2(n)$** representing the "number of pairs of adjacent tiles that are not adjacent in the goal state," can be combined using a linear combination approach:

$$h(n) = c_1x_1(n) + c_2x_2(n).$$

The constants (**c1** and **c2**) are adjusted to achieve the best fit with the actual data on solution costs. It is expected that both **c_1** and **c_2** are positive, as misplaced tiles and incorrect adjacent pairs make the problem more challenging. While this heuristic satisfies the condition **h(n) = 0** for goal states, it may not necessarily be both admissible and consistent.

2.2 Introduction to Machine Learning

2.2.1 Need of Machine Learning:

The need for machine learning arises from the growing complexity and volume of data stored in various archives within an organization. Traditional methods of data analysis and manual processing are often insufficient to extract meaningful insights from the vast amounts of information available. Machine learning offers a set of powerful tools and techniques to address this challenge. Let's explore how machine learning fulfills the specific needs mentioned:

1. Extracting Knowledge from Data:

Challenge: Organizations accumulate large amounts of data in various forms, such as customer information, transaction records, and operational data.
Machine Learning Solution: Algorithms can be trained to automatically analyze and identify patterns, trends, and correlations within this data, helping extract valuable knowledge and insights.

2. Facilitating Decision Making:

Challenge: Decision-makers need accurate and timely information to make informed choices.
Machine Learning Solution: By processing and analyzing data, machine learning models can provide predictions and recommendations, enabling better decision-making based on data-driven insights.

3. Useful for Designing New Products:

Challenge: Designing products that meet the evolving needs of customers requires a deep understanding of market trends and user preferences.
Machine Learning Solution: Machine learning algorithms can analyze historical data, customer feedback, and market trends to identify patterns and preferences, aiding in the design of new products that are more likely to be successful in the market.

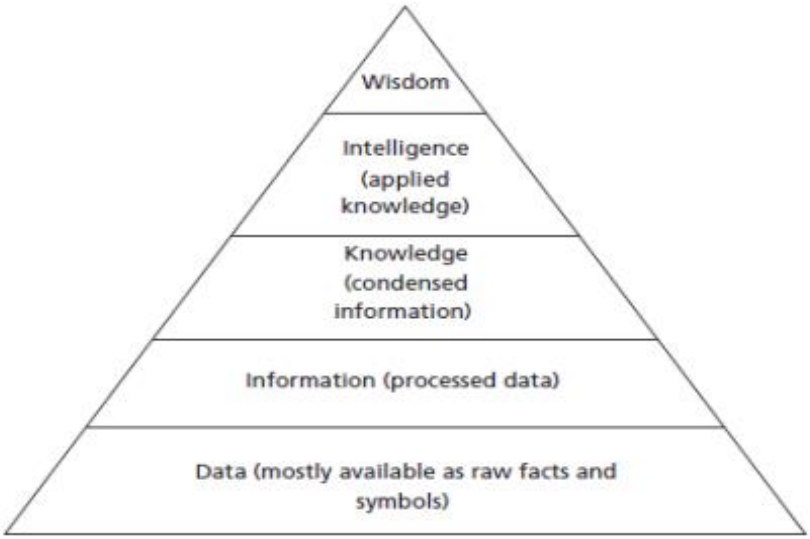
4. Improving Business Processes:

Challenge: Inefficient business processes can lead to increased costs and reduced productivity.
Machine Learning Solution: Machine learning can optimize and automate various aspects of business processes by identifying bottlenecks, predicting maintenance needs, and suggesting improvements, leading to increased efficiency and cost savings.

5. Developing Decision Support Systems:

Challenge: Decision-makers often need comprehensive and organized information to make strategic choices.
Machine Learning Solution: Machine learning plays a crucial role in developing decision support systems that can analyze diverse data sources, provide relevant insights, and assist decision-makers in understanding complex scenarios.

2.2.2 Knowledge Pyramid:



The knowledge pyramid is a conceptual framework that illustrates the hierarchical relationships between different levels of information processing and understanding.

It typically consists of the following levels:

Data, Information, Knowledge, Intelligence, and Wisdom. Let's explore each level with examples:

Data:

Data are raw facts and figures without context or meaning.

Example: A list of numbers (e.g., 1, 5, 7, 3) or a series of characters (e.g., A, B, C, D) would be considered data. Without further context, these values lack significance.

Information:

Information is processed and organized data that conveys meaning.

Example: If we take the numbers from the previous example (1, 5, 7, 3) and label them as the scores of students in a test, we now have information. This information provides context and meaning to the raw data.

Knowledge:

Knowledge involves understanding and applying information to solve problems or make decisions.

Example: If we analyze the test scores (information) and draw conclusions, such as identifying trends, strengths, weaknesses, or correlations, we are moving into the realm of knowledge. Knowledge represents a deeper level of understanding derived from information.

Intelligence:

Intelligence involves the ability to learn from experiences, adapt to new situations, and apply knowledge to solve problems.

Example: An intelligent system, such as a machine learning algorithm, can analyze vast datasets, learn patterns, and make predictions. For instance, predicting future student performance based on historical data requires an intelligent system that can discern underlying trends and relationships.

Wisdom:

Wisdom is the highest level and involves making **sound judgments** and decisions based on a deep understanding of principles and values.

Example: Wisdom involves the ability to apply knowledge and intelligence in a way that reflects ethical considerations, long-term consequences, and a broader understanding of the human condition. A wise decision, for example, might

involve not just optimizing a business process for short-term gains but considering the well-being of employees and the impact on the community.

2.2.3 Popularity of Machine Learning

The popularity of machine learning can be attributed to several factors, and the mentioned facts play a significant role in its widespread adoption. Let's explore each of these factors:

High Volume of Available Data:

In today's digital age, there is an unprecedented amount of data generated and collected by organizations. Traditional methods of data analysis may struggle to handle such large volumes effectively. Machine learning excels in processing and extracting valuable insights from massive datasets. The more data available, the better machine learning models can be trained, leading to more accurate predictions and outcomes.

Cost of Storage Reduction:

The cost of storing data has significantly decreased over the years. This reduction in storage costs allows organizations to accumulate and retain vast amounts of data without incurring prohibitive expenses. Machine learning relies on having access to diverse and extensive datasets for training robust models. The affordability of storage facilitates the collection and retention of the data necessary for effective machine learning applications.

Availability of Complex Algorithms:

Machine learning algorithms form the backbone of the technology's capabilities. The field has seen significant advancements, leading to the development of sophisticated and powerful algorithms. These algorithms can handle complex patterns, relationships, and decision-making tasks. The availability of these advanced algorithms empowers developers and data scientists to tackle a wide range of problems, from image recognition and natural language processing to predictive analytics and recommendation systems.

2.2.4 What is Machine Learning?

“Machine Learning is the field of study that gives the computers ability to learn without being programmed”. [Arthur Samuel]

Here the input Data is used to **develop intelligent models**. The models will be used to predict new inputs. The aim of ML is **to learn a model or set of rules** from the given data set automatically so that it can predict the unknown data correctly.

Tom Mitchell, a computer scientist and professor at Carnegie Mellon University, provided a widely cited and influential definition of machine learning in his book titled "Machine Learning" (1997). The definition is as follows:

"*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.*"

Let's break down the components of this definition:

Tasks (T):

These are the specific activities or problems that the machine learning system is designed to perform. Tasks can range from image recognition and language translation to playing games or making predictions.

Experience (E):

Experience refers to the data or information that the machine learning system is exposed to. This data serves as the basis for the system to learn patterns, relationships, and rules.

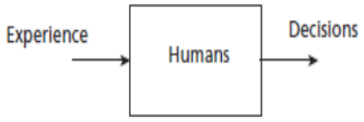
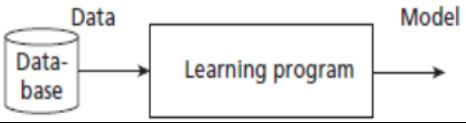
Performance Measure (P):

This represents the metric used to evaluate how well the machine learning system is performing the tasks in T. The goal is to improve performance over time as the system gains more experience.

In essence, **Mitchell's** definition emphasizes the idea that a machine learning system learns from experience by improving its performance on a specific set of tasks. The learning process involves the system adapting to patterns and information within the provided data to enhance its ability to perform the designated tasks more effectively.

2.2.5 Learning Systems

Learning systems, in a broad sense, refer to systems or mechanisms that have the capability to acquire knowledge or skills through experience or instruction. This term can encompass various types of learning, including human learning and machine learning.

Aspect	Human Learning Systems	Machine Learning Systems
		
Nature of Learning	Humans learn through sensory perception, cognition, and social interactions.	Machines learn through algorithms and statistical models. They analyze data, identify patterns, and adjust parameters to make predictions or decisions.
Learning Process	Involves complex cognitive processes including perception, memory, reasoning, and decision-making.	Typically involves training a model on a dataset, where it learns patterns and relationships in the data. The learning process is iterative with parameter adjustments.
Adaptability	Exhibits a high degree of adaptability, can transfer knowledge across diverse situations, and generalize information.	Specialized for specific tasks and may struggle with generalization to new, unseen scenarios. Relies on the data it was trained on.
Explainability	Can articulate reasoning behind decisions and actions. Learning is accompanied by a rich understanding of cause-and-effect relationships.	Some models, especially complex ones, are considered "black boxes" as their decision-making processes can be challenging to interpret or explain.
Innateness	Born with innate cognitive abilities and a capacity for learning. Can acquire complex skills and knowledge without explicit instruction.	Requires explicit programming or training data to learn. Lacks innate cognitive abilities and relies on information provided during training.
Biological Basis	Deeply rooted in the structure and function of the brain. Neurological processes, synaptic plasticity, and cognitive functions contribute to the learning process.	Based on mathematical models, algorithms, and computational processes. Inspired by statistical methods and optimization techniques.

2.2.6 What is a Model?

In machine learning, a model refers to the mathematical or computational representation of a real-world process or system. It is a set of rules, algorithms, or parameters that a machine learning algorithm uses to make predictions or decisions based on input data. The goal of building a model is to capture patterns, relationships, and insights from data, allowing the model to generalize and make predictions on new, unseen data.

Here are some key components of a machine learning model:

Input Data:
The data that the model uses to make predictions or decisions. This data is typically divided into features, which are the input variables used by the model.

Parameters:
These are the internal variables of the model that are learned from the training data. The model is trained to adjust its parameters to minimize the difference between its predictions and the actual outcomes in the training data.

Output or Prediction:
The result produced by the model based on the input data. The output could be a classification (e.g., spam or not spam), a regression value (e.g., predicting house prices), or some other type of prediction.

Training:
The process of feeding the model with labeled training data, allowing it to learn and adjust its parameters to make accurate predictions. During training, the model aims to minimize the difference between its predictions and the actual outcomes.

Testing/Evaluation:
After training, the model is evaluated on new, unseen data to assess its performance and generalization ability. This is important to ensure that the model can make accurate predictions on data it has not seen before.

2.2.7 Types of Models:

A Model can be any one of the following:

1. Mathematical Equation
2. Relational diagrams like Graphs/Trees
3. Logical if/else rules
4. Groupings called clusters

1. Mathematical Equation:

Example: In linear regression, the model can be expressed as a mathematical equation, such as $y = mx + b$, where 'y' is the output, 'x' is the input feature, 'm' is the slope, and 'b' is the intercept. The model learns the values of 'm' and 'b' during training to predict 'y' based on new 'x' values.

2. Relational Diagrams like Graphs/Trees:

Example: Decision trees are a graphical representation of if/else conditions. Each node in the tree represents a decision based on a specific feature, leading to subsequent nodes or leaves. For instance, a decision tree for classifying whether an email is spam might involve nodes such as "Is the subject line misleading?" and "Does the email contain suspicious links?"

3.Logical if/else Rules:

Example: In a rule-based system, logical if/else rules guide the decision-making process. For spam detection, a rule might be "if the sender is unknown and the email contains certain keywords, then classify it as spam." These rules collectively form the model's decision logic.

4. Groupings called Clusters:

Example: Clustering models, like K-Means, group similar data points together. In customer segmentation, the model might identify clusters of customers based on their purchasing behavior. Customers with similar buying patterns form distinct clusters, enabling targeted marketing strategies.

In essence, a machine learning model can adopt the guise of a mathematical equation, a graphical structure, a set of logical rules, or even data groupings. The choice of model depends on the nature of the problem and the characteristics of the data, with the overarching goal of making informed predictions, classifications, or uncovering patterns in the given data.

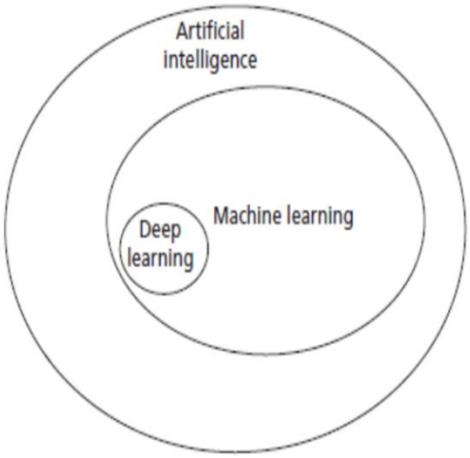
2.2.8 Relationship of ML with Other Fields:

1. **AI, ML and DL**:

AI is a broad concept that refers to the development of computer systems capable of performing tasks that typically require human intelligence. These tasks include problem-solving, understanding natural language, recognizing patterns, and making decisions.

ML is a subset of AI that focuses on the development of algorithms that enable computers to learn patterns and make predictions or decisions without explicit programming. Instead of being explicitly programmed, ML systems learn from data.

DL is a subfield of ML that involves neural networks with multiple layers (deep neural networks). It aims to mimic the structure and function of the human brain, allowing the model to automatically learn hierarchical representations of data.



Relationship:

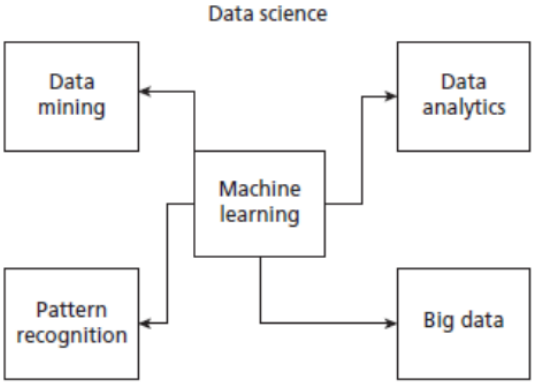
AI and ML: AI is the broader concept that encompasses ML. ML is a subset of AI, representing a specific approach to achieving artificial intelligence—by allowing machines to learn from data.

ML and DL: ML includes traditional algorithms as well as deep learning algorithms. Deep learning is a specialized form of ML that specifically deals with neural networks containing multiple layers, enabling more sophisticated feature learning.

2. Machine Learning and Data Science:

Machine Learning (ML), Data Science, Data Mining, Data Analytics, and Big Data are interconnected fields that often overlap but serve different purposes within the broader domain of handling and extracting value from data. Data science is an “umbrella term” covering from data collection to data analysis.

Data Science is a multidisciplinary field that involves using scientific methods, processes, algorithms, and systems to extract knowledge and insights from structured and unstructured data. Data scientists work with various tools and techniques, including statistics, machine learning, and domain-specific knowledge, to analyze and interpret complex datasets.



Machine Learning is a subset of artificial intelligence that focuses on the development of algorithms and models that enable computers to learn from data without being explicitly programmed. ML is used within data science for predictive modeling, pattern recognition, and decision-making based on historical data.

Data Mining involves the process of discovering patterns and knowledge from large datasets. It includes techniques from statistics, machine learning, and database systems. Data mining is often used to uncover hidden patterns and relationships in data, contributing to the predictive modeling aspect of data science.

Data Analytics involves examining, cleaning, transforming, and modeling data to draw conclusions, make decisions, and support decision-making processes. While data analytics is a broader term, it often includes statistical analysis, descriptive analytics, and, in some cases, predictive analytics, which aligns with the goals of data science.

Big Data refers to extremely large and complex datasets that cannot be easily managed, processed, or analyzed with traditional data processing tools.

The processing and analysis of Big Data require specialized technologies and approaches, such as distributed computing, parallel processing, and storage solutions.

Characteristics of Big Data

1. **Volume:** Big Data involves massive amounts of data that can be in petabytes or exabytes.
2. **Velocity:** Describes the speed at which data is generated, collected, and processed.
3. **Variety:** Encompasses the different types of data sources and formats. Big Data often includes structured data (e.g., databases), unstructured data (e.g., text, images, videos), and semi-structured data (e.g., JSON or XML files).
4. **Veracity:** Refers to the quality and reliability of the data. Big Data sources can be messy and may include inaccuracies, inconsistencies, and errors.
5. **Value:** Focuses on the importance of turning data into value. The ultimate goal of working with Big Data is to extract meaningful insights and value from the massive amounts of data collected.

Relationships:

Data Science and Machine Learning: ML is a key component of data science, providing tools and techniques for building predictive models and extracting patterns from data.

Data Science and Data Mining: Data mining contributes to the exploratory and predictive analysis stages in data science, aiding in uncovering hidden patterns and trends.

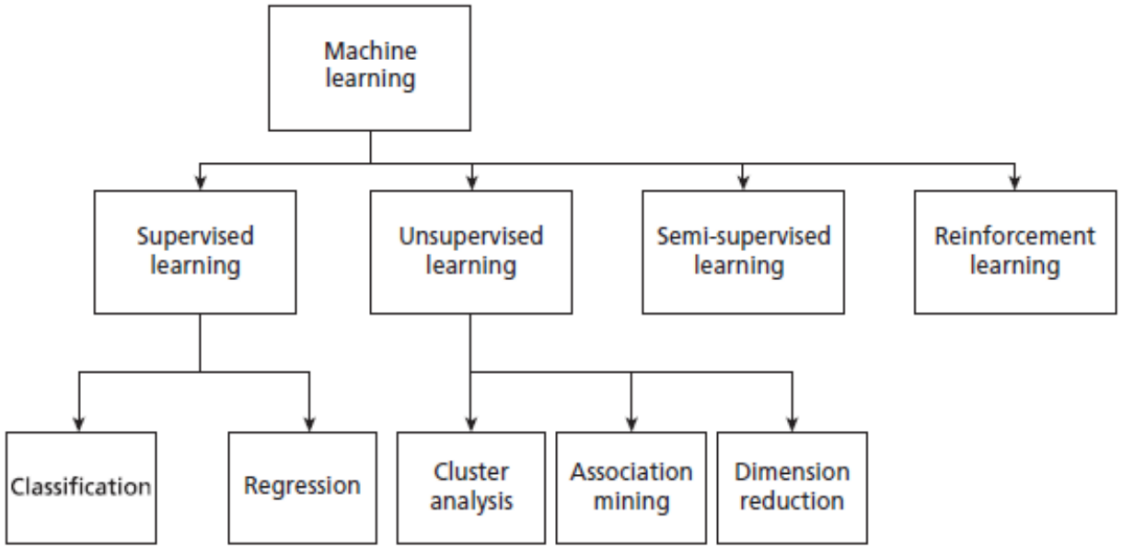
Data Science and Data Analytics: Data analytics is an integral part of data science, involving various techniques for examining and interpreting data to inform decision-making.

Big Data and Data Science: Big data technologies are often used in data science to handle and process large volumes of data efficiently, enabling the analysis of massive datasets.

Big Data and Data Analytics: Big data technologies are crucial for handling the volume, velocity, and variety of data in data analytics processes, especially when dealing with large-scale datasets.

2.2.9 Types of Machine Learning

Machine learning can be broadly categorized into three main types based on the learning styles and approaches: supervised learning, unsupervised learning, and reinforcement learning. Each type has its unique characteristics and applications.



1. Supervised Learning:

In supervised learning, the algorithm is trained on a labeled dataset, where the input data is paired with corresponding output labels. The goal is for the algorithm to learn the mapping from input to output, making predictions or classifications on new, unseen data.

Examples:

Classification: Predicting whether an email is spam or not.

Regression: Predicting house prices based on features like size and location.

2. Unsupervised Learning:

Unsupervised learning involves training algorithms on unlabeled data, and the system tries to find patterns, relationships, or structures in the data without explicit guidance. It is about discovering the inherent structure of the data.

Examples:

Clustering: Grouping similar customers based on their purchasing behavior.

Dimensionality Reduction: Reducing the number of features in a dataset while retaining important information.

3. Reinforcement Learning:

Reinforcement learning involves an agent interacting with an environment and learning to make decisions by receiving feedback in the form of rewards or penalties. The agent aims to maximize the cumulative reward over time.

Examples:

Game Playing: Learning to play games like chess or Go.

Robotics: Training a robot to perform tasks in a dynamic environment.

4. Semi-Supervised Learning:

Semi-supervised learning is a combination of supervised and unsupervised learning. It involves training a model on a dataset that contains both labeled and unlabeled data. This approach is useful when obtaining a fully labeled dataset is expensive or time-consuming.

Example: Training a model to recognize handwritten digits with a small set of labeled examples and a larger set of unlabeled examples.

5. Self-Supervised Learning:

Self-supervised learning is a type of unsupervised learning where the model generates its own labels from the input data. It involves creating tasks that the model can solve without external annotations.

Example: Training a model to predict missing parts of an image by learning from the surrounding context.

6. Transfer Learning:

Transfer learning involves training a **model on one task** and then transferring its knowledge to **another related task**. This can save computational resources and improve performance, especially when the amount of labeled data for the target task is limited.

Example: Pre-training a model on a large dataset for image classification and then fine-tuning it on a smaller dataset for a specific classification task.

These types of machine learning cater to different scenarios and problem domains, allowing practitioners to choose the most suitable approach based on the characteristics of the data and the goals of the task at hand.

2.2.10 Labeled and Unlabeled Data

Labeled and unlabeled data refer to the presence or absence of target labels (output values) associated with the input data in a machine learning dataset.

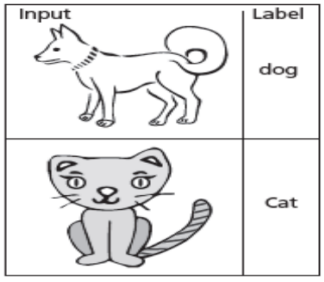
Labeled Data: Labeled data is a dataset in which each example (data point) is paired with its corresponding output or target label. In supervised learning, the algorithm uses these labels during training to learn the relationship between the input features and the target variable.

Example: In a dataset for spam email detection, each email is labeled as either "spam" or "not spam."

Table 1.1: Iris Flower Dataset

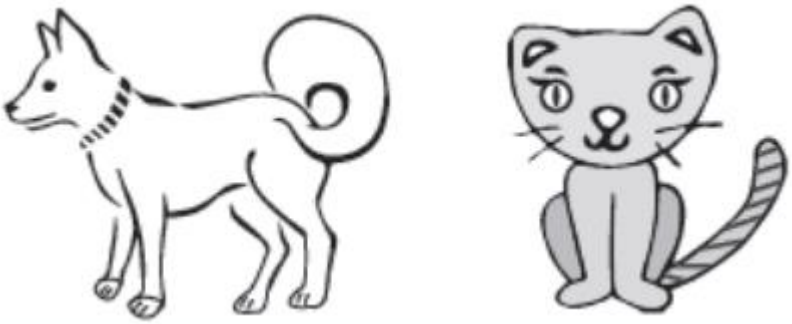
S.No.	Length of Petal	Width of Petal	Length of Sepal	Width of Sepal	Class
1.	5.5	4.2	1.4	0.2	Setosa
2.	7	3.2	4.7	1.4	Versicolor
3.	7.3	2.9	6.3	1.8	Virginica

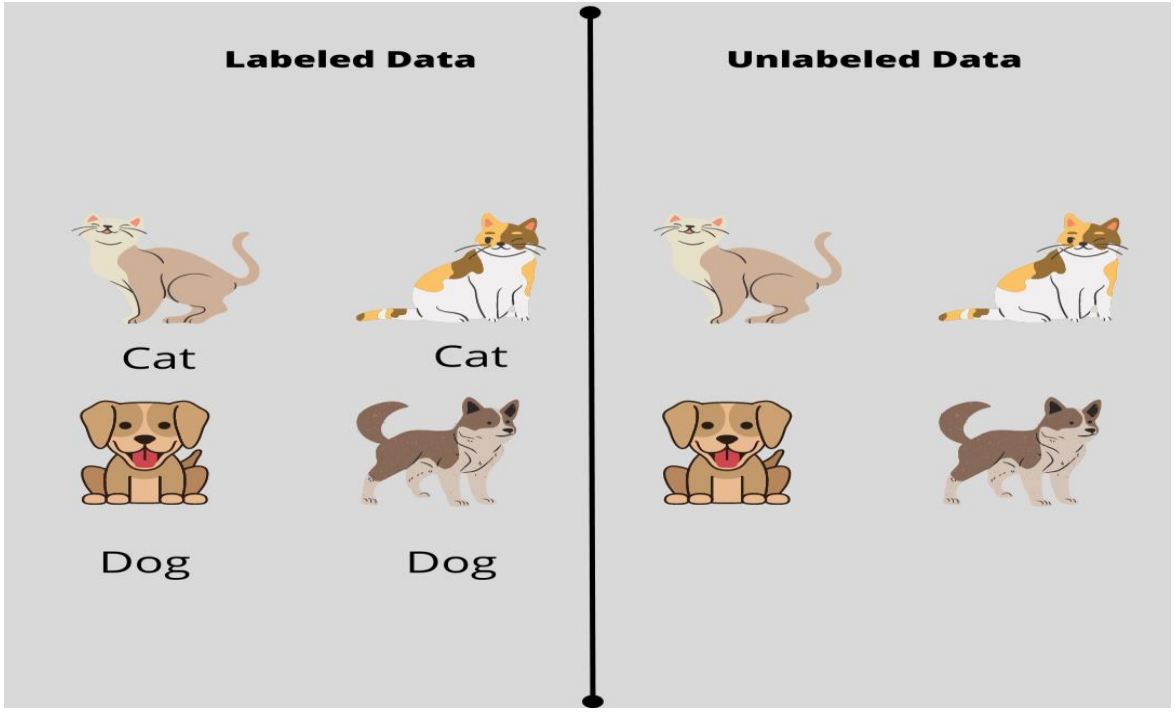
A dataset need not be always numbers. It can be images or video frames. Deep neural networks can handle images with labels. In the following Figure 1.6, the deep neural network takes images of dogs and cats with labels for classification.



Unlabeled Data: Unlabeled data is a dataset in which the input examples are provided without corresponding output labels. This type of data is often used in unsupervised learning, where the algorithm aims to find patterns, relationships, or structures in the data without explicit guidance from labeled examples.

Example: An image dataset with pictures of various animals but without labels indicating the specific type of each animal.





2.2.11 Supervised Learning

Supervised learning is a type of machine learning where the algorithm is trained on a labeled dataset, meaning that the input data used for training is paired with corresponding output labels. The goal of supervised learning is to learn a mapping from input features to the target output by generalizing patterns from the labeled training data. Once trained, the model can make predictions or classifications on new, unseen data.

Here are the key components and characteristics of supervised learning:

Labeled Data:

- **Training Data**: The labeled dataset used for training consists of input-output pairs. Each input example is associated with a known output label.
- **Example:** In a supervised learning problem for image recognition, each image in the training dataset is labeled with the class it belongs to (e.g., cat, dog, car).

Input Features:

- **Features:** The input to the algorithm consists of features or attributes that describe each example. These features are used to make predictions about the output.
- **Example:** For a supervised learning model predicting house prices, the input features might include the number of bedrooms, square footage, and location of a house.

Output Labels:

- **Target Variable**: The output label, also known as the target variable, is what the algorithm aims to predict. It represents the desired outcome or classification for a given input example.
- **Example**: In a supervised learning task for spam detection, the output labels are "spam" or "not spam" for each email.

Training Process:

- Learning from Examples: During the training phase, the algorithm learns from the labeled examples in the training dataset. It adjusts its internal parameters or model structure to minimize the difference between its predictions and the true output labels.
- Objective Function: The algorithm typically optimizes an objective function, such as minimizing the mean squared error for regression tasks or minimizing the cross-entropy loss for classification tasks.

Prediction on New Data:

- **Generalization**: After training, the supervised learning model is capable of making predictions on new, unseen data. It generalizes from the patterns learned during training to make accurate predictions on similar examples.

- **Evaluation**: The model's performance is often evaluated on a separate dataset, called the test set, to assess how well it generalizes to new, unseen examples.

2.2.12 Types of Supervised Learning:

Classification: In classification tasks, the goal is to assign input data to specific categories or classes. Examples include spam detection, image recognition, or sentiment analysis.

Regression: In regression tasks, the goal is to predict a continuous output variable. Examples include predicting house prices, temperature, or stock prices.

A. Classification

Classification is a type of supervised learning in machine learning where the goal is to predict the category or class label of an input example. In classification tasks, the algorithm is trained on a labeled dataset, where each example is associated with a specific class or category. Once trained, the model can then classify new, unseen data into predefined categories based on the patterns it learned during training.

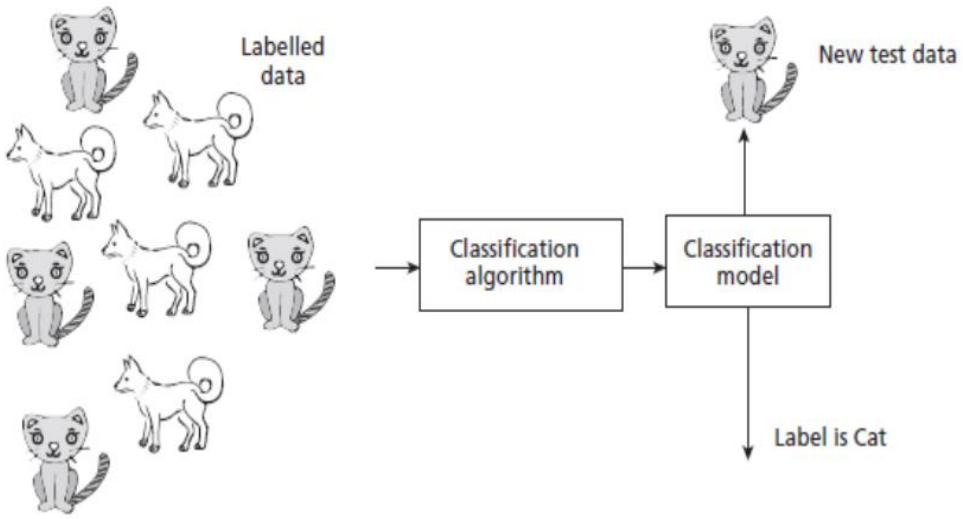
Types of Classification:

Binary Classification:
The task involves classifying examples into two classes (e.g., spam or not spam, positive or negative).

Multiclass Classification:
The task involves classifying examples into more than two classes (e.g., classifying animals into categories like cats, dogs, and birds).

Evaluation Metrics:
Classification models are evaluated using metrics such as *accuracy, precision, recall, F1 score, and confusion matrix*. These metrics assess the model's ability to correctly classify examples and manage trade-offs between different types of classification errors.

Example: Classification of Cat and Dog



Example: Email Spam Classification:
Input Features: Various features of an email, such as sender, subject, and content.
Classes: "Spam" or "Not Spam."
Training Data: Labeled examples of emails, with each labeled as either spam or not spam.
Objective: Predict whether a new email is spam or not based on its features.

Common classification algorithms include:

- Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machines (SVM)
- k-Nearest Neighbors (k-NN)
- Naive Bayes
- Neural Networks

Classification is a fundamental and widely used technique in machine learning, with applications in various domains such as image recognition, document classification, medical diagnosis, and more. The choice of the algorithm depends on the characteristics of the data and the specific requirements of the classification task at hand.

B. Regression

Regression is a type of supervised learning in machine learning that focuses on predicting a continuous output variable (also known as the response or dependent variable) based on one or more input features (independent variables). The goal of regression is to establish a relationship or mapping between the input features and the continuous output, allowing the algorithm to make predictions on new, unseen data.

Types of Regression:

1. **Simple Linear Regression**: Involves a single input feature to predict a continuous output.
2. **Multiple Linear Regression**: Involves multiple input features to predict a continuous output.
3. **Polynomial Regression**: Extends linear regression by including polynomial terms, allowing the model to capture more complex relationships.

Evaluation Metrics: Regression models are evaluated using metrics such as **mean squared error (MSE), mean absolute error (MAE), and R-squared**. These metrics quantify the difference between the predicted values and the actual values in the test dataset.

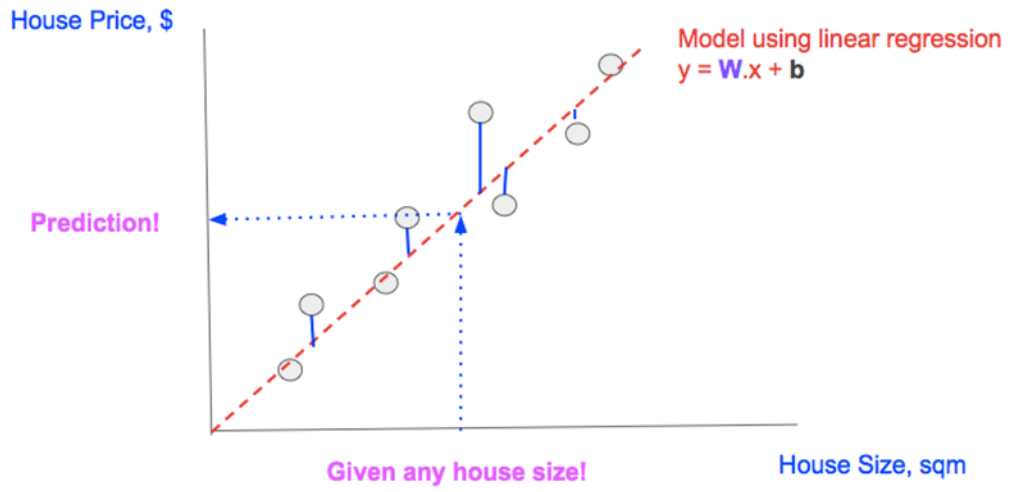
Example: Housing Price Prediction:

Input Features: Features such as square footage, number of bedrooms, and location of a house.

Output Variable: The continuous output variable is the price of the house.

Training Data: Labeled examples of houses, where each example includes features and the corresponding sale price.

Objective: Predict the price of a new house based on its features.



Regression Algorithms: Common regression algorithms include:

- Linear Regression
- Ridge Regression
- Lasso Regression
- Decision Trees for Regression
- Random Forests for Regression
- Support Vector Machines (SVM) for Regression
- Neural Networks for Regression

Regression is applied in various domains, including finance, economics, healthcare, and engineering, where predicting a numerical value is essential for decision-making. The choice of the regression algorithm depends on the nature of the data and the specific requirements of the regression task.

Example 1 : Predicting weekly Product Sales using Regression

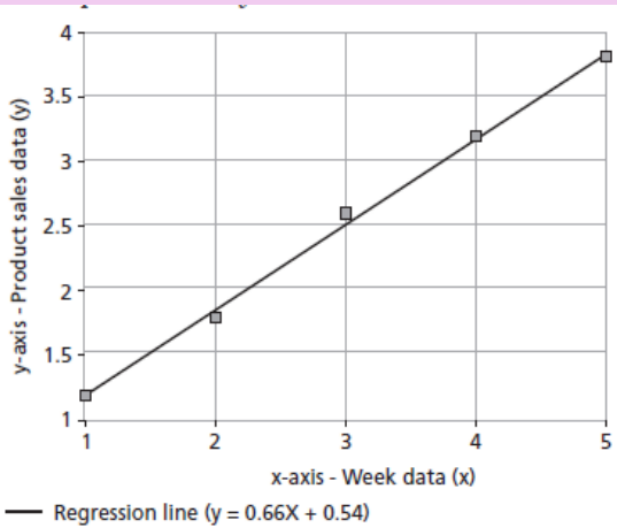
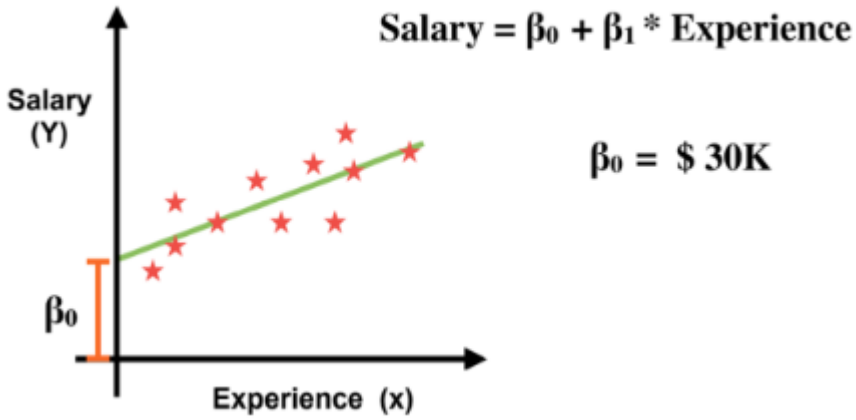


Figure 1.8: A Regression Model of the Form $y = ax + b$

product sales = 0.66 × Week + 0.54.

Example 2: Predicting Salary based on Experience using Linear Regression



2.2.13 Unsupervised Learning

Unsupervised learning is a type of machine learning where the algorithm is given data without explicit instructions on what to do with it. The system tries to learn the patterns and the structure from the data without labeled responses to guide the learning process. Unsupervised learning is often used for exploratory data analysis and pattern discovery.

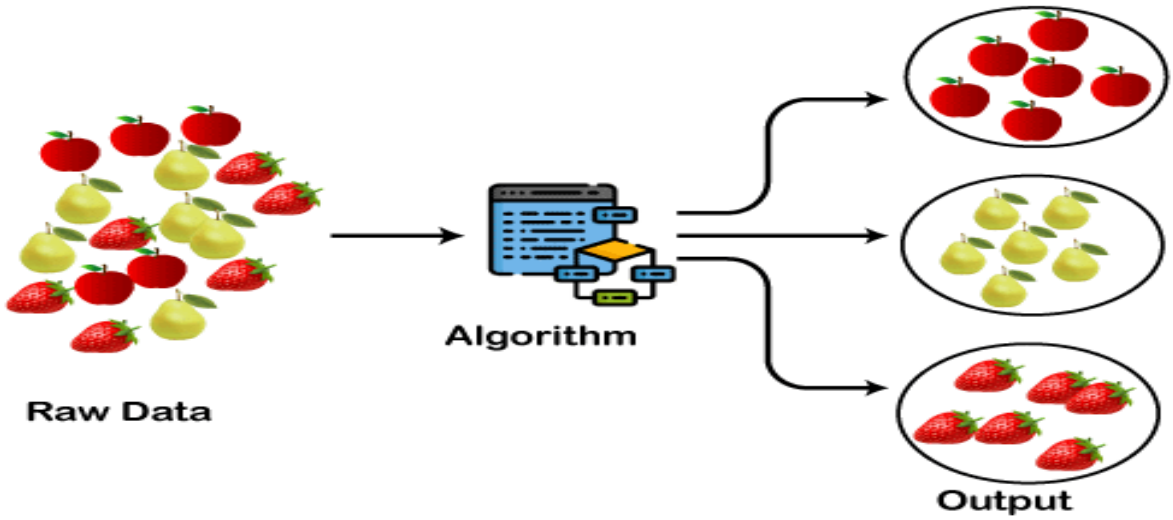
Unsupervised learning is particularly useful in scenarios where labeled data is scarce or unavailable. It allows the algorithm to discover hidden patterns and structures in the data without explicit guidance. The choice of algorithm depends on the specific task, characteristics of the data, and the goals of the analysis.

There are several types of unsupervised learning, with **clustering and dimensionality reduction** being prominent approaches:

A. Clustering:

Clustering involves grouping similar data points together based on some notion of similarity or distance. The goal is to identify natural groupings or clusters in the data.

Example 1:



Example 2:

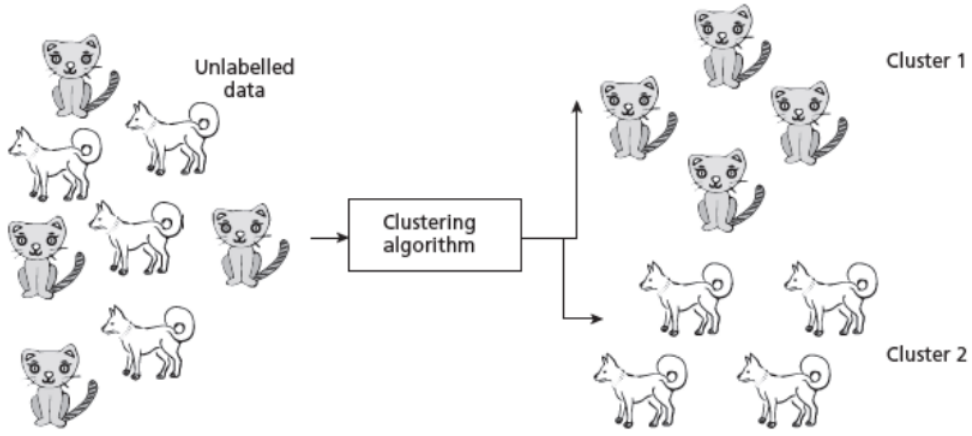


Figure 1.9: An Example Clustering Scheme

Types:

- **K-Means Clustering**: Divides data into k clusters based on the mean values of data points.
- **Hierarchical Clustering**: Forms a tree-like hierarchy of clusters, revealing nested relationships.
- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**: Identifies dense regions of data points, forming clusters separated by areas of lower point density.

B. Dimensionality Reduction:

Dimensionality reduction techniques aim to reduce the number of features or variables in the data while retaining essential information. This can help in visualizing and understanding complex datasets and mitigating the curse of dimensionality.

Types:

- **Principal Component Analysis (PCA)**: Finds orthogonal axes (principal components) along which the data varies the most and projects the data onto these components.
- **t-Distributed Stochastic Neighbor Embedding (t-SNE)**: Focuses on preserving local structures in high-dimensional data, often used for visualizing clusters in lower-dimensional space.
- **Autoencoders:** Neural network-based models that learn a compressed representation of the input data.

C. Association:

Association rule learning discovers interesting relationships, dependencies, or associations between variables in large datasets. It is commonly used in market basket analysis.

Examples:

- **Apriori Algorithm**: Identifies frequent itemsets in a transactional database and generates association rules.
- **Eclat Algorithm**: Similar to Apriori but focuses on identifying frequent itemsets without candidate generation.

D. Generative Modeling:

Generative models learn the underlying distribution of the data and can generate new, similar data samples.

Examples:

Generative Adversarial Networks (GANs): Consist of a generator and a discriminator network that are trained adversarially to generate realistic data samples.

Variational Autoencoders (VAEs): Learn a probabilistic mapping from the input space to a latent space and can generate new samples by sampling from the latent space.

E. Density Estimation:

Density estimation techniques aim to model the underlying probability distribution of the data.

Examples:

- **Kernel Density Estimation (KDE):** Estimates the probability density function by placing a kernel at each data point.
- **Gaussian Mixture Models (GMMs):** Model the distribution of data points as a mixture of multiple Gaussian distributions.

Unsupervised learning techniques are versatile and find applications in various domains, including data exploration, pattern discovery, and preprocessing for subsequent supervised learning tasks. The choice of the technique depends on the specific goals and characteristics of the data at hand.

Supervised Vs Unsupervised Learning

This table summarizes the key differences between supervised and unsupervised learning in terms of their definitions, objectives, training data, outputs, examples, evaluation metrics, types, applications, and example algorithms

Sl. No	Supervised Learning	Unsupervised Learning
1	**Definition:**	**Definition:**
	In supervised learning, the algorithm is trained on a labeled dataset, where	In unsupervised learning, the algorithm is trained on an unlabeled dataset, and there are no predefined output labels.

	each example has a corresponding output label.	
2	**Objective:**	**Objective:**
	The primary objective is to learn a mapping between input features and output labels, enabling the algorithm to make predictions on new, unseen data.	The goal is to explore the inherent patterns, relationships, or structures within the data without explicit guidance from labeled examples.
3	**Training Data:**	**Training Data:**
	Requires a labeled dataset for training, where each example includes input features and a corresponding output label.	Utilizes an unlabeled dataset for training, where input examples do not have associated output labels.
4	**Output:**	**Output:**
	The output of a supervised learning model is a prediction or classification label for a given input example.	The output of an unsupervised learning model may include clusters, reduced-dimension representations, or discovered patterns in the data.
5	**Examples:**	**Examples:**
	Examples of supervised learning tasks include image classification, spam detection, and predicting house prices.	Examples of unsupervised learning tasks include clustering similar customer behaviors, dimensionality reduction for visualization, and discovering associations in data.
6	**Evaluation Metrics:**	**Evaluation Metrics:**
	Common evaluation metrics include accuracy, precision, recall, F1 score, and confusion matrix.	Evaluation metrics depend on the specific unsupervised learning task and may include measures like silhouette score for clustering or reconstruction error for dimensionality reduction.
7	**Types:**	**Types:**
	Types include classification, regression, and tasks where the algorithm learns to map input features to predefined output labels.	Types include clustering, dimensionality reduction, and tasks focused on discovering patterns or associations within the data.
8	**Applications:**	**Applications:**
	Applied in scenarios where the goal is to predict an outcome based on labeled examples, such as in medical diagnosis or sentiment analysis.	Applied when exploring data structures, identifying groups of similar data points, or reducing the dimensionality of high-dimensional data for visualization.
9	**Example Algorithms:**	**Example Algorithms:**
	Examples include logistic regression, decision trees, support vector machines, and neural networks.	Examples include k-means clustering, principal component analysis (PCA), and association rule learning algorithms like Apriori.

2.2.14 Semi Supervised Learning

Semi-supervised learning is a type of machine learning where the algorithm is trained on a dataset that contains both labeled and unlabeled data. In other words, only a subset of the training data has explicit labels, while the majority of the data is unlabeled. Semi-supervised learning aims to leverage the information from both labeled and unlabeled data to improve the performance of the model.

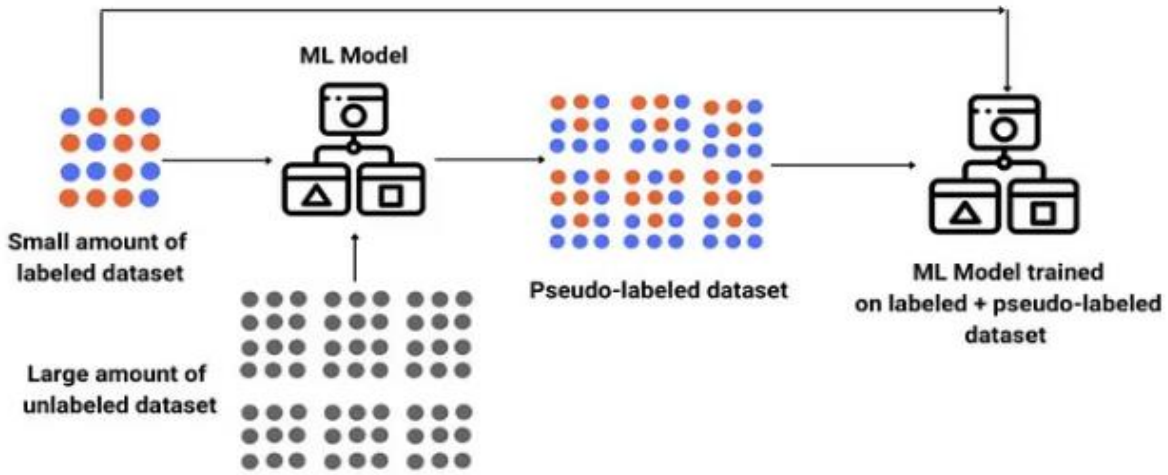


Image From Google

Example 1:

In image classification, if a model is trained on a dataset of labeled images of cats and dogs, but has a large set of unlabeled images, semi-supervised learning can be applied by incorporating both labeled and unlabeled images to improve the model's ability to generalize to new images.

Image From Google

Example 2:

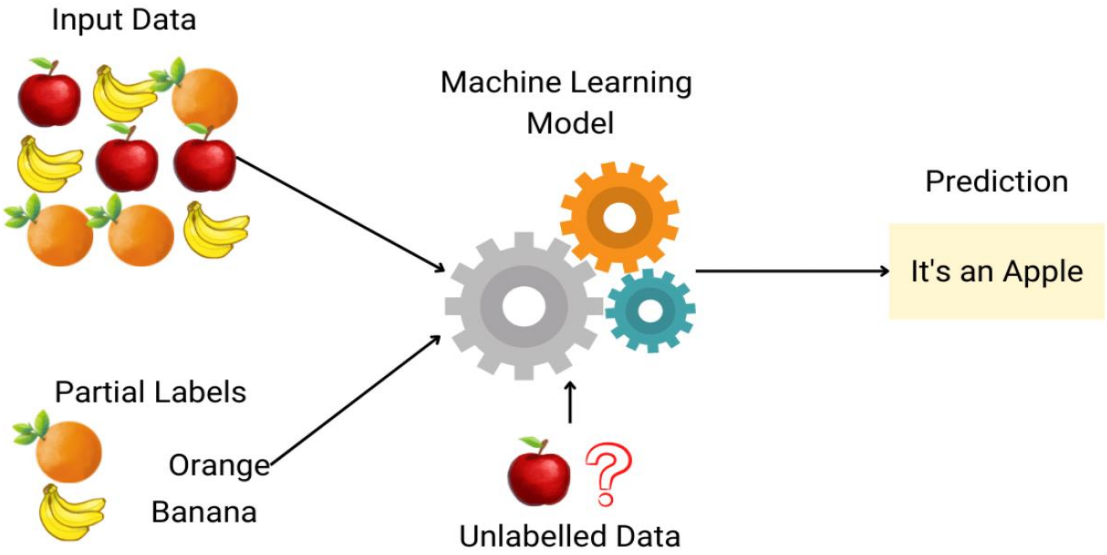


Image From Google

2.2.15 Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning paradigm where an agent learns to make decisions by interacting with an environment. The agent takes actions, receives feedback in the form of rewards or penalties, and learns to optimize its behavior over time to achieve a specific goal. In RL, the agent is not explicitly told which actions to take but discovers the optimal strategy through trial and error.

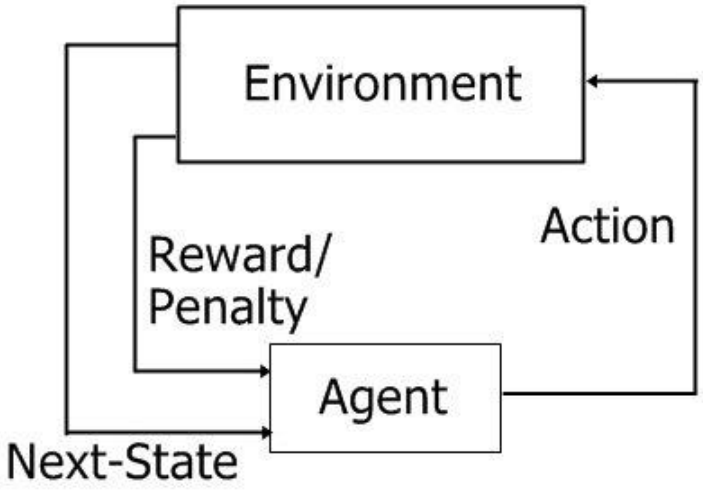


Image From Google

Example:

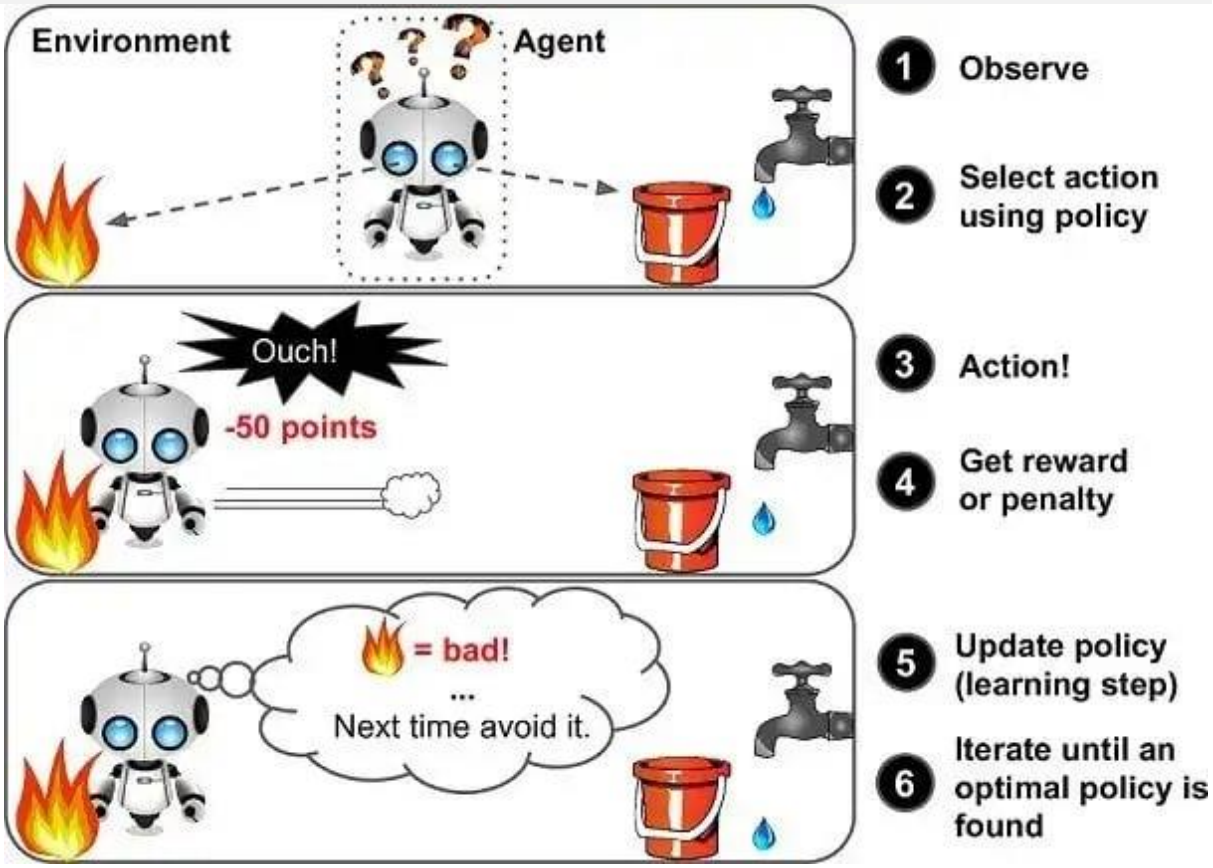


Image From Google

Another Example: Negative Feedback

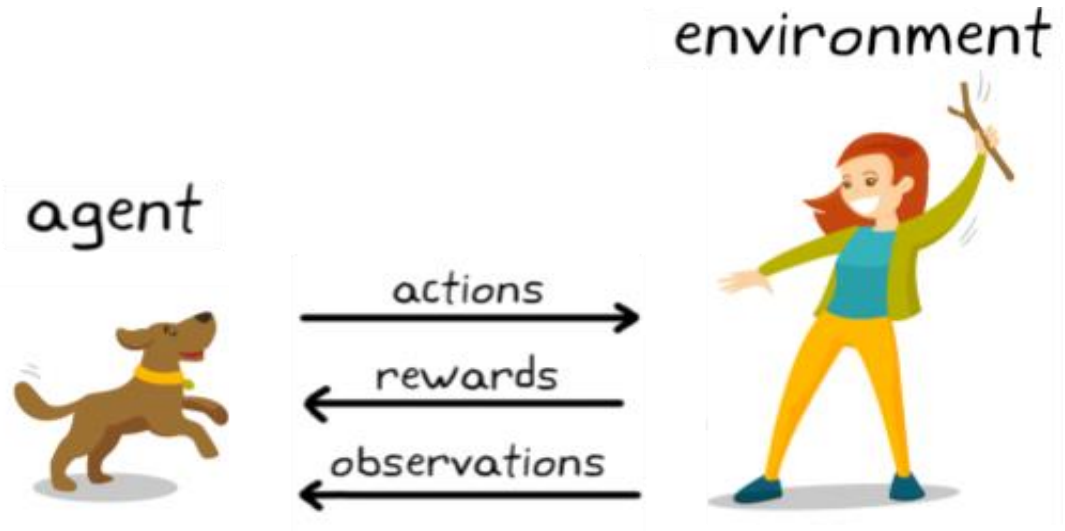


Image From Google

Another Example :

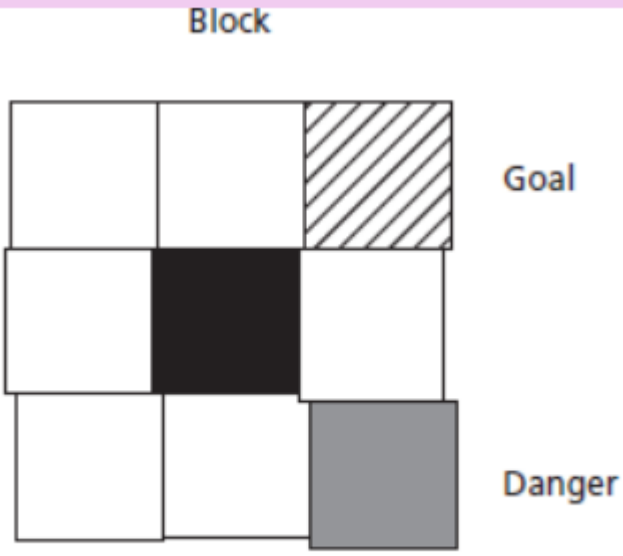


Figure 1.10: A Grid game

2.2.16 Challenges of Machine Learning

1. ILL-POSED PROBLEMS – PROBLEMS WHOSE SPECIFICATIONS ARE NOT CLEAR
2. HUGE DATA
3. HUGE COMPUTATION POWER
4. COMPLEXITY OF ALGORITHMS
5. BIAS-VARIANCE

Ill-posed Problems: Ill-posed problems are challenging in machine learning as their specifications are ambiguous or incomplete, making it difficult to formulate a precise solution. The lack of clear problem definitions can hinder the development of effective models.

Huge Data: Dealing with massive datasets poses challenges in terms of storage, processing, and computational efficiency. It requires scalable algorithms and infrastructure to extract meaningful patterns and insights from large volumes of data.

Huge Computation Power: The need for significant computational power arises when handling complex models or large datasets. High computational requirements can limit the accessibility and affordability of certain machine learning approaches.

Complexity of Algorithms: The complexity of machine learning algorithms, especially in deep learning, can lead to challenges in training and optimization. Understanding and managing the intricate details of complex algorithms are crucial for effective implementation.

Bias-Variance: Balancing the trade-off between bias and variance is a fundamental challenge. High bias may lead to oversimplified models, while high variance may result in overfitting. Striking the right balance is essential for creating models that generalize well to unseen data.

2.2.17 Machine Learning Process

The machine learning process is a series of interconnected steps, commencing with a profound understanding of the business problem, progressing through data exploration and preprocessing, model training and evaluation, and the deployment of a fine-tuned solution.

Collaboration between domain experts and data scientists is pivotal, ensuring that each phase aligns cohesively with business objectives and delivers meaningful value.

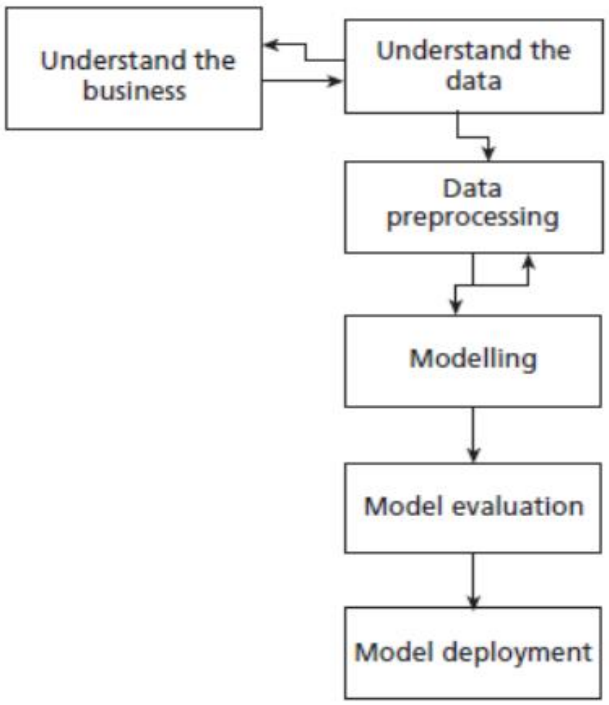


Figure 1.11: A Machine Learning/Data Mining Process

Understanding the Business Problem:
In this initial step, it's crucial to comprehend the business problem at hand, including its goals, challenges, and desired outcomes. Collaborating with stakeholders and domain experts helps in defining the scope and objectives of the machine learning project.

Understanding Data:
This stage involves exploring and gaining insights into the dataset that will be used for training and testing the machine learning model. Understanding the data includes examining its structure, identifying relevant features, and assessing the quality of information available for analysis.

Data Preprocessing:
Data preprocessing is the cleaning and transformation phase where the raw data is refined to ensure it is suitable for analysis. This includes handling missing values, removing outliers, scaling features, and encoding categorical variables, preparing the data for effective model training.

Data Modeling:
Data modeling entails selecting an appropriate machine learning algorithm and training it on the preprocessed dataset. The model learns patterns and relationships from the data to make predictions or classifications, with the choice of algorithm depending on the nature of the problem (e.g., regression, classification).

Model Evaluation:
After training the model, it is essential to evaluate its performance using metrics relevant to the specific problem, such as accuracy, precision, recall, or mean squared error. Model evaluation helps ensure that the algorithm meets the specified criteria and generalizes well to new, unseen data.

Model Deployment:
Once the model has been trained and evaluated successfully, it is deployed into a real-world environment. Deployment involves integrating the model into existing systems or applications, allowing it to make predictions on new data and contribute value to the intended business use case. Continuous monitoring and updates may be necessary to maintain optimal performance.

2.2.18 Applications of Machine Learning

Machine learning has diverse applications across various industries, contributing to advancements and efficiency improvements. Some notable machine learning applications include:

Sl. No	Problem Domain	Applications
1	**Business**	- Predictive Analytics for Sales and Marketing/ Predicting the banckruptcy of a business firm
2	**Banking**	- Credit Scoring/ Prediction of bank loan defaulters and detecting credit card frauds
3	**Image Processing**	- Object Detection/ Image search engines, object identification, image classification and generating synthetic images
4	**Audio/Voice**	- Speech Recognition/ Chatbots like Alexa, Microsoft Cortana. Developing chatbots for customer support, speech to text and text to voice
5	**Telecommunication**	- Network Optimization/ Trend analysis and identification of bogus calls, fraudulent calls and the callers.
6	**Games**	- AI-driven Game Characters/ Game programs for Chess, GO and Video games
7	**NLP**	- Chatbots for Customer Support/ Google Translation, Text Summarization and sentiment and analysis
8	**Web Analysis and Services**	- Personalized Content Recommendations/ Identification of Access patterns, detection of e-mail spams, viruses, personalized web services, search engines like Google, detection of promotion of user websites
9	**Medicine**	- Diagnostic Systems/ Prediction of disease
10	**Multimedia and Security**	- Facial Recognition/ Face Recognition/Identification biometric projects like identification of a person from a large image or video database
11	**Environmental Monitoring**	- Climate Modeling
12	**Human Resources**	- Recruitment Automation

These examples showcase the diverse applications of machine learning in different problem domains, from enhancing business strategies to revolutionizing healthcare and optimizing various processes in different industries. The list is not exhaustive, and machine learning continues to find innovative applications across a wide range of domains.

2.3 Understanding Data

Under Development