

Fill in the blanks

- i) Artificial intelligence is mimicking \_\_\_\_\_.
- ii) Bayesian network is \_\_\_\_\_ graph.
- iii) A frame represents \_\_\_\_\_.
- iv) A semantic network is an irregular graph that has \_\_\_\_\_ in vertices and \_\_\_\_\_ on arcs

Here are the answers to your questions:

i) Artificial intelligence is mimicking **human intelligence**<sup>12345</sup>. ii) Bayesian network is a **directed acyclic** graph<sup>678</sup>. iii) A frame represents **a concept or an instance; it is a record-like structure which consists of a collection of attributes and its values to describe an entity in the world**<sup>91011</sup>. iv) A semantic network is an irregular graph that has **concepts** in vertices and **relations** on arcs<sup>121314</sup>.

## **What are the types of agents? Describe the structure of agents and their architecture.**

Sure, let's start with the types of agents. Based on their degree of perceived intelligence and capability, agents can be grouped into five classes<sup>12345</sup>:

1. **Simple Reflex Agent:** These agents take decisions based on the current percepts and ignore the rest of the percept history. They only succeed in a fully observable environment<sup>1</sup>.
2. **Model-based Reflex Agent:** These agents can work in a partially observable environment. They have two important factors: a model, which is knowledge about "how things happen in the world," and an internal state, which is a representation of the current state based on percept history<sup>1</sup>.
3. **Goal-based Agents:** These agents expand the capabilities of the model-based agent by having "goal" information. They choose an action so that they can achieve the goal<sup>1</sup>.
4. **Utility-based Agents:** These agents act based not only on goals but also on the best way to achieve the goal. They provide a measure of success at a given state<sup>1</sup>.
5. **Learning Agents:** A learning agent in AI is the type of agent which can learn from its past experiences, or it has learning capabilities<sup>1</sup>.

Now, let's discuss the structure of agents. An agent is anything that can perceive its environment through sensors and act upon that environment through actuators<sup>2</sup>. The structure of an agent can be viewed as Agent = Architecture + Agent

[Program](#)<sup>26</sup>. Here, the architecture is the machinery that the agent executes on. [It could be a device with sensors and actuators, for example, a robotic car, a camera, or a PC](#)<sup>26</sup>. An agent program is an implementation of an agent function. [An agent function is a map from the percept sequence \(history of all that an agent has perceived to date\) to an action](#)<sup>26</sup>.

The architecture of an agent is the building block for creating an agent much like creating an object in a class. The agent architecture is the brain of the agent as it determines how the knowledge/information is represented in the agent. [It also determines the action the agent should take based on its underlying reasoning/interpretation mechanism](#)<sup>7</sup>.

I hope this helps! Let me know if you have any other questions.

## **Specify the task environment using PEAS.**

[Sure, let's define a task environment using the PEAS \(Performance measure, Environment, Actuators, Sensors\) framework](#)<sup>1234</sup>:

1. **Performance Measure:** This is the criterion an AI agent uses to evaluate its actions. The performance measure can vary according to the specific application. [For instance, in a recommendation system, the performance measure might be user satisfaction or click-through rates](#)<sup>2</sup>.
2. **Environment:** This refers to the external context where the AI agent operates. [This encompasses all the factors and variables the agent cannot directly control but needs to interact with to achieve its objectives](#)<sup>2</sup>.
3. **Actuators:** [These are the mechanisms enabling the agent to interact with the environment](#)<sup>12</sup>.
4. **Sensors:** [These provide the agent with the means to perceive and gather information](#)<sup>12</sup>.

Let's take an example of an autonomous car:

- **Performance Measure:** Safety, obeying traffic rules, comfort of the passengers, reaching the destination in time, etc.
- **Environment:** Roads, traffic signals, pedestrians, other vehicles, weather conditions, etc.
- **Actuators:** Steering wheel, accelerator, brake, horn, etc.
- **Sensors:** Cameras, radar, GPS, speedometer, odometer, etc.

This is how we specify a task environment using the PEAS framework. Let me know if you have any other questions!

**Write a comparison between following. (Any Two) (Write any three points of differentiations only)**

**a) Breadth First Search and Depth First Search**

**b) Depth Limited Search and Depth First Iterative Deepening**

**c) Branch & bound and A\* search**

**d) Forward and backward chaining**

<https://www.geeksforgeeks.org/difference-between-bfs-and-dfs/>

<https://www.geeksforgeeks.org/difference-between-backward-and-forward-chaining/>

Sure, here are some key differences between Branch & Bound (BnB) and A\* search algorithms<sup>1234</sup>:

1. **Search Strategy:** BnB explores the search space exhaustively by branching on variables and estimating bounds on the solutions that can be attained in the subproblems<sup>2</sup>. On the other hand, A\* search uses both path cost and heuristic values to estimate the total path cost of going from a start node to a goal via a particular path<sup>1</sup>.
2. **Priority Determination:** For BnB, the priority is solely based on the cost from the start state to the goal state<sup>3</sup>. Conversely, for A\*, the priority is defined by the sum of the cost from the start state to the current state and the heuristic estimate of the cost to reach the goal state<sup>1</sup>.
3. **Optimality:** If A\* returns a solution, that solution is guaranteed to be optimal, as long as the branching factor is finite, arc costs are non-negative, and the heuristic used is an underestimate of the length of the shortest path from a node to a goal node<sup>1</sup>. BnB, on the other hand, does not inherently guarantee optimality unless combined with other strategies<sup>2</sup>.

I hope this helps! Let me know if you have any other questions.

Sure, here are some key differences between Depth Limited Search (DLS) and Depth First Iterative Deepening (DFID):

1. **Depth Limit:** DLS is an algorithm that performs depth-first search up to a specified limit, meaning it searches only up to a limited depth<sup>1</sup>. On the other hand, DFID is an algorithm that combines the benefits of Breadth-First Search's completeness and Depth-First Search's space efficiency<sup>2</sup>. It does

this by performing a depth-first search to a certain “depth limit”, and keeps increasing the depth limit until the goal is found<sup>2</sup>.

2. **Efficiency:** DLS can be less efficient than DFID because it might not reach the goal if the depth limit is less than the depth at which the goal node is located<sup>1</sup>. DFID, however, is more efficient because it ensures that the shallowest goal node (least depth) is found first<sup>2</sup>.
3. **Completeness:** DLS is not complete, i.e., it does not guarantee that a solution will be found if one exists, especially when the depth limit is less than the depth of the goal<sup>1</sup>. DFID, on the other hand, is complete for finite state spaces as it covers all nodes at a particular depth before moving to the next level<sup>2</sup>.

I hope this helps! Let me know if you have any other questions.

**Consider the water jug problem: you are given 2 jugs, a 4 gallon and 3-gallon jugs. Neither has any measuring mark in it. There is a pub that can be used to fill the jugs with the water. How can you get exactly 2-gallon of water into a 4-gallon of jug? State the production rules for water jug problem.**

Sure, here's how you can get exactly 2 gallons of water into the 4-gallon jug:

1. Fill the 3-gallon jug completely.
2. Pour the 3 gallons from the 3-gallon jug into the 4-gallon jug.
3. Fill the 3-gallon jug again.
4. Pour the water from the 3-gallon jug into the 4-gallon jug until it's full. Now you have 2 gallons left in the 3-gallon jug.
5. Empty the 4-gallon jug.
6. Pour the 2 gallons from the 3-gallon jug into the 4-gallon jug. Now you have exactly 2 gallons in the 4-gallon jug.

Now, let's state the production rules for the water jug problem:

1. **Rule 1:** If the 4-gallon jug is not full and the 3-gallon jug is not empty, fill the 4-gallon jug from the 3-gallon jug.
2. **Rule 2:** If the 4-gallon jug is not full, fill it completely from the pub.
3. **Rule 3:** If the 3-gallon jug is not empty and the 4-gallon jug is full, empty the 4-gallon jug.
4. **Rule 4:** If the 3-gallon jug is not empty, empty it.

5. **Rule 5:** If the 3-gallon jug is empty, fill it completely from the pub.
6. **Rule 6:** If the 4-gallon jug is full and the 3-gallon jug is not empty, empty the 3-gallon jug.
7. **Rule 7:** If the 4-gallon jug is full, empty it.

These rules can be applied in different orders to solve different instances of the water jug problem. In our case, we used a specific sequence of these rules to get exactly 2 gallons of water into the 4-gallon jug. Let me know if you have any other questions!

**Explain Alpha beta pruning with appropriate example**

<https://www.mygreatlearning.com/blog/alpha-beta-pruning-in-ai/>

**Explain the concept of minimax algorithm with suitable example**

<https://tutorialforbeginner.com/mini-max-algorithm-in-ai>

**Solve the following 15-puzzle problem Using branch and bound technique, up to 3 iterations.**

<https://www.geeksforgeeks.org/8-puzzle-problem-using-branch-and-bound/>

**Using the suggested abbreviations (the capitalized words), translate each of the following into the language of predicate logic.**

- i) All HORSES and COWS are FARM animals.
- ii) All CATS and DOGS make EXCELLENT pets.
- iii) RAINY days and MONDAYS always get me DOWN.
- iv) CATS and DOGS are the only SUITABLE pets.
- v) The only PERSONS INSIDE are MEMBERS and GUESTS.
- vi) The only CATS and DOGS that are SUITABLE pets are the ones that have been HOUSE-trained.
- vii) CATS and DOGS are the only ANIMALS that are SUITABLE pets.

### **viii) CATS and DOGS that have RABIES are not SUITABLE pets.**

Here are the translations of the given sentences into the language of predicate logic:

i)  $\forall x (HORSE(x) \vee COW(x) \rightarrow FARM(x))$  ii)  $\forall x (CAT(x) \vee DOG(x) \rightarrow EXCELLENT(x))$  iii)  $\forall x (RAINY(x) \vee MONDAY(x) \rightarrow DOWN(x))$  iv)  $\forall x (SUITABLE(x) \rightarrow (CAT(x) \vee DOG(x)))$  v)  $\forall x (INSIDE(x) \rightarrow (MEMBER(x) \vee GUEST(x)))$  vi)  $\forall x ((CAT(x) \vee DOG(x)) \wedge SUITABLE(x) \rightarrow HOUSE-TRAINED(x))$  vii)  $\forall x (SUITABLE(x) \rightarrow (CAT(x) \vee DOG(x)))$  viii)  $\forall x ((CAT(x) \vee DOG(x)) \wedge RABIES(x) \rightarrow \neg SUITABLE(x))$

Please note that these translations assume that the domain of discourse includes all relevant entities (e.g., all animals for the first statement). Let me know if you have any other questions!

### **Consider the following probabilities and find the complete probability distribution over $P(\text{fever, cold, flu, malaria})$**

$$P(\neg \text{fever} \mid \text{cold}, \neg \text{flu}, \neg \text{malaria}) = 0.6$$

$$P(\neg \text{fever} \mid \neg \text{cold}, \text{flu}, \neg \text{malaria}) = 0.2$$

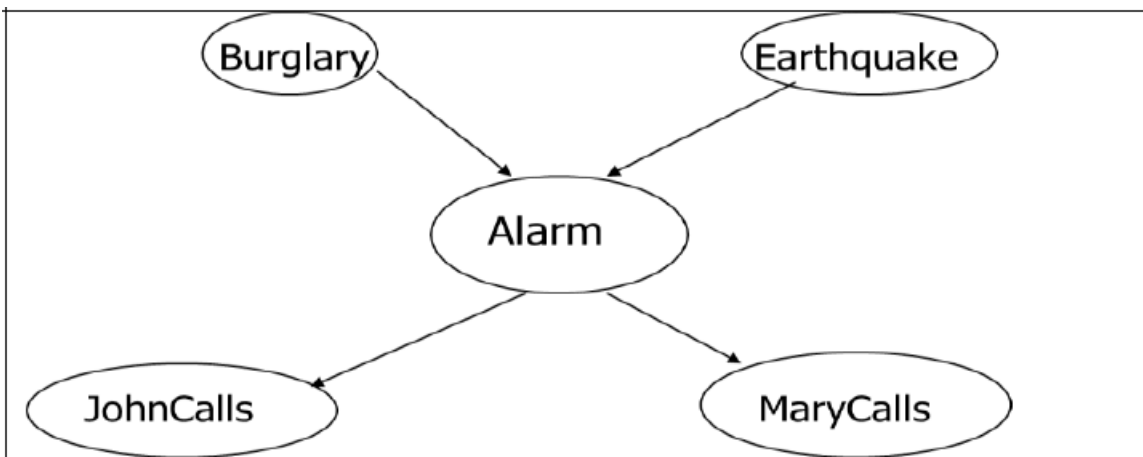
$$P(\neg \text{fever} \mid \neg \text{cold}, \neg \text{flu}, \text{malaria}) = 0.1$$

The given probabilities are conditional probabilities of not having a fever given certain conditions. However, to find the complete probability distribution over  $P(\text{fever, cold, flu, malaria})$ , we would need more information.

The complete probability distribution would be a table that lists the probability of every possible combination of fever, cold, flu, and malaria. This would be a total of  $2^4 = 16$  combinations, since each of the 4 variables (fever, cold, flu, malaria) can either be present or not present.

For example, one entry in the distribution would be  $P(\text{fever, cold, flu, malaria})$ , the probability of having all four conditions. Another entry would be  $P(\neg \text{fever}, \neg \text{cold}, \neg \text{flu}, \neg \text{malaria})$ , the probability of having none of the conditions.

Without additional information, such as the marginal probabilities of each condition and their dependencies, it's not possible to compute the complete probability distribution. If you have more information, I'd be happy to help further! Let me know if you have any other questions.



Consider the above Bayesian network of probability, in this context explain types of inferences.

- i) Diagnosis inferences
- ii) Causal inferences
- iii) Intercausal inferences
- iv) Mixed inferences

**Draw a frame system for hotel room. (consider the room with your own assumptions.**

**“Moti, the white dog on the porch, eats a bone.”**

**Draw a conceptual graph for above knowledgebase.**

Consider the following Rommmate world,

- i) Convert sentences in FOL
  - ii) Convert the knowledgebase in CNF
  - iii) Conclude that “Sprinklers are on” ? (Use resolution )
- Your roommate is wet
  - If your roommate is wet, it is because of rain, sprinklers, or both
  - If your roommate is wet because of sprinklers, the sprinklers must be on
  - If your roommate is wet because of rain, your roommate must not be carrying the umbrella
  - The umbrella is not in the umbrella holder
  - If the umbrella is not in the umbrella holder, either you must be carrying the umbrella, or your roommate must be carrying the umbrella
  - You are not carrying the umbrella

Let’s denote the following predicates:

- $Wet(Roommate)$  : Your roommate is wet.
- $Rain$  : It is raining.

- $\text{SprinklersOn}$  : The sprinklers are on.
- $\text{CarryingUmbrella}(\text{Roommate})$  : Your roommate is carrying the umbrella.
- $\text{UmbrellaInHolder}$  : The umbrella is in the holder.
- $\text{CarryingUmbrella}(\text{You})$  : You are carrying the umbrella.

i) The sentences in First Order Logic (FOL) would be:

1.  $\text{Wet}(\text{Roommate})$
2.  $\text{Wet}(\text{Roommate}) \leftrightarrow (\text{Rain} \vee \text{SprinklersOn})$
3.  $(\text{Wet}(\text{Roommate}) \wedge \text{SprinklersOn}) \rightarrow \text{SprinklersOn}$
4.  $(\text{Wet}(\text{Roommate}) \wedge \text{Rain}) \rightarrow \neg \text{CarryingUmbrella}(\text{Roommate})$
5.  $\neg \text{UmbrellaInHolder}$
6.  $\neg \text{UmbrellaInHolder} \rightarrow (\text{CarryingUmbrella}(\text{You}) \vee \text{CarryingUmbrella}(\text{Roommate}))$
7.  $\neg \text{CarryingUmbrella}(\text{You})$

ii) The knowledge base in Conjunctive Normal Form (CNF) would be:

1.  $\text{Wet}(\text{Roommate})$
2.  $\neg \text{Wet}(\text{Roommate}) \vee \text{Rain} \vee \text{SprinklersOn}$
3.  $\neg \text{Wet}(\text{Roommate}) \vee \neg \text{SprinklersOn} \vee \text{SprinklersOn}$
4.  $\neg \text{Wet}(\text{Roommate}) \vee \neg \text{Rain} \vee \neg \text{CarryingUmbrella}(\text{Roommate})$
5.  $\neg \text{UmbrellaInHolder}$
6.  $\text{UmbrellaInHolder} \vee \text{CarryingUmbrella}(\text{You}) \vee \text{CarryingUmbrella}(\text{Roommate})$
7.  $\neg \text{CarryingUmbrella}(\text{You})$

iii) To conclude that “Sprinklers are on” using resolution, we add the negation of what we want to prove to our knowledge base and try to resolve to an empty clause:

Add  $\neg \text{SprinklersOn}$  to the knowledge base. Now, we can resolve clauses 2 and 8 to get  $\neg \text{Wet}(\text{Roommate}) \vee \text{Rain}$ . Then, resolve this with clause 1 to get  $\text{Rain}$ .

Next, resolve  $\text{Rain}$  with clause 4 to get  $\neg \text{CarryingUmbrella}(\text{Roommate})$ . Resolve this with clause 6 to get  $\text{UmbrellaInHolder} \vee \text{CarryingUmbrella}(\text{You})$ . Finally, resolve this with clause 5 and clause 7 to get an empty clause.

Since we derived an empty clause, it means our assumption ( $\neg \text{SprinklersOn}$ ) led to a contradiction. Therefore, the original statement (“Sprinklers are on”) must be true in this world.

I hope this helps! Let me know if you have any other questions.

**Give three computer applications for which machine learning approaches**



**seem appropriate and three for which they seem inappropriate.**

Machine learning can be highly effective in applications where patterns can be learned from large amounts of data and where it's difficult to explicitly program the desired behavior. Here are three examples:

1. **Recommendation Systems:** Machine learning is great for recommendation systems (like those used by Amazon or Netflix) because it can analyze a user's past behavior and the behavior of similar users to suggest products or movies that the user might like.
2. **Speech Recognition:** Machine learning algorithms can learn to understand spoken language by being trained on large amounts of data where the spoken words are matched to written words.
3. **Image Recognition:** Machine learning can be used to identify objects, people, or even emotions in images. This is used in a wide range of applications, from tagging friends on social media to identifying tumors in medical imaging.

However, machine learning might not be the best approach in the following scenarios:

1. **Simple Rule-Based Systems:** If a task can be accomplished by following a set of simple rules, then explicit programming might be more appropriate. For example, a system to calculate tax based on income doesn't require machine learning.
2. **Predicting Rare Events:** Machine learning models often struggle with predicting rare events because they don't have enough data about these events to learn from. For example, predicting earthquakes or other natural disasters is currently beyond the scope of machine learning.
3. **When Interpretability is Key:** In situations where it's important to understand why a decision was made, machine learning might not be the best choice. For example, in legal or medical decisions, it's often important to understand the reasoning behind a decision, which can be difficult with some machine learning models.

Evaluate dimension reduction using PCA on the given dataset and calculate the Eigen values.

	X1	X2	X3	X4
X	4	8	13	7
	11	4	5	14

**Distinguish between supervised learning and Reinforcement learning. Illustrate with an example.**

Sure, here are some key differences between Supervised Learning and Reinforcement Learning:

1. **Learning Method:** In Supervised Learning, the learning model is presented with an input and desired output. [It learns by example<sup>1</sup>](#). [In contrast, in Reinforcement Learning, the learning agent interacts directly with an environment and learns from the consequences of its actions<sup>12</sup>](#).
2. **Tasks:** Supervised Learning has two main tasks called Regression and Classification<sup>1</sup>. [In contrast, Reinforcement Learning has different tasks, such as exploitation or exploration, Markov's decision processes, Policy Learning, Deep Learning, and value learning<sup>1</sup>](#).
3. **Data Dependency:** [In Supervised Learning, each example will have a pair of input objects and an output with desired values<sup>1</sup>](#). In contrast, in Reinforcement Learning, the agent makes an observation for every time period "t" and receives a reward for every observation. [The goal is to collect as many rewards as possible to make more observations<sup>1</sup>](#).

Now, let's illustrate with examples:

- **Supervised Learning Example:** An example of Supervised Learning is spam email detection. The model is trained with a set of emails (inputs) and their labels (outputs: spam or not spam). [The model learns to classify new emails based on this training<sup>3</sup>](#).
- **Reinforcement Learning Example:** An example of Reinforcement Learning is a chess game. The agent (the player) interacts with the environment (the game) by making moves. [The agent receives a reward or penalty based on the outcome of the game \(win or lose\), and it learns to make better moves over time<sup>2</sup>](#).

I hope this helps! Let me know if you have any other questions.

## What do you mean by 'Vanishing Gradient'? In which NN it occurs?

The 'Vanishing Gradient' problem is a challenge encountered when training artificial neural networks with gradient-based learning methods and backpropagation<sup>1</sup>. During each iteration of training, each of the neural network's weights receives an update proportional to the partial derivative of the error function with respect to the current weight<sup>1</sup>. The issue arises when the gradients are very small, effectively preventing the weight from changing its value<sup>1</sup>. In the worst case, this may completely stop the neural network from further training<sup>1</sup>.

This problem is often attributed to the choice of activation functions and the architecture of the neural network<sup>2</sup>. Activation functions like the sigmoid or hyperbolic tangent (tanh) have gradients that are in the range of 0 to 0.25 for sigmoid and -1 to 1 for tanh<sup>2</sup>. When these activation functions are used in deep networks, the gradients of the loss function with respect to the parameters can become very small, effectively preventing the weights from changing their values during training<sup>2</sup>.

The Vanishing Gradient problem is particularly prevalent in deep feedforward and recurrent neural networks<sup>23</sup>. In Recurrent Neural Networks (RNNs), this problem occurs when the RNN is trying to learn long-term dependencies<sup>4</sup>. In deep feedforward networks, the gradients of the loss function with respect to the parameters can become very small, effectively preventing the weights from changing their values during training<sup>2</sup>.

## When does regularization come into play in Machine Learning?

Regularization is a technique used in machine learning to prevent overfitting<sup>12</sup>. Overfitting occurs when a model performs well on the training data but poorly on unseen data, such as a test set or validation set<sup>12</sup>. This is often due to the model learning the noise in the training data, rather than the underlying patterns<sup>1</sup>.

Regularization comes into play when you have a complex model that is overfitting the training data<sup>12</sup>. By adding a penalty term to the loss function, regularization discourages learning a more complex or flexible model, thus avoiding overfitting<sup>12</sup>. The penalty term reduces the values of the model parameters, which in turn reduces the complexity of the model<sup>12</sup>.

Commonly used regularization techniques include Lasso (L1) and Ridge (L2) regularization<sup>1</sup>. Lasso tends to result in sparse solutions, effectively performing

variable selection, while Ridge regularization reduces the coefficients but does not set them to zero<sup>1</sup>.

For example, in a linear regression model, without regularization, the aim is to minimize the residual sum of squares (RSS). When regularization is applied, the aim is to minimize not just the RSS but also a penalty term that is a function of the magnitude of the coefficients<sup>12</sup>.

In summary, regularization is used in machine learning to prevent overfitting, reduce model complexity, and improve the model's performance on unseen data<sup>12</sup>.  
Let me know if you have any other questions!

## **Can Random Forest Algorithm be used both for Continuous and Categorical Target Variables? What do you mean by Bagging**

Yes, the Random Forest algorithm can be used for both continuous and categorical target variables<sup>1</sup>. In a random forest, which is a combination of decision trees, the classification model refers to the categorical dependent variable, and the regression model refers to the numeric or continuous dependent variable<sup>1</sup>.

Bagging, short for bootstrap aggregating, is a machine learning ensemble method<sup>2</sup>. It involves training multiple models on different subsets of the training data and combining their predictions<sup>2</sup>. This technique helps to reduce overfitting and improve the accuracy of the model<sup>2</sup>. In bagging, a random sample of data in a training set is selected with replacement, meaning that the individual data points can be chosen more than once<sup>3</sup>. After numerous data samples are generated, these models are trained independently<sup>4</sup>. For example, the average or majority of these predictions yield a correct estimate depending on the type of task - regression or classification<sup>4</sup>. Bagging is also known as Bootstrap aggregating<sup>4</sup>.

### **i) Compare between Inductive and deductive Methods**

Sure, here are some key differences between Inductive and Deductive methods:

1. **Direction of Reasoning:** Inductive reasoning moves from specific observations to broad generalizations, while deductive reasoning works the other way around, moving from general premises to specific conclusions<sup>12</sup>.
2. **Purpose:** Inductive reasoning aims at developing a theory, while deductive reasoning aims at testing an existing theory<sup>1</sup>.
3. **Approach:** Inductive reasoning is a bottom-up approach that starts with specific observations and ends with a general conclusion. On the other hand, deductive reasoning is a top-down approach that starts with a general theory or premise and ends with a specific conclusion<sup>12</sup>.

For example, if you observe that the sun rises every morning, and from this observation, you conclude that the sun will rise tomorrow morning, you are using inductive reasoning. [On the other hand, if you start with the general premise that all birds can fly, and you know that a penguin is a bird, so you conclude that a penguin can fly, you are using deductive reasoning<sup>12</sup>.](#)

I hope this helps! Let me know if you have any other questions.

**Suppose 10000 patients get tested for flu; out of them. 9000 are actually-**

**) healthy and 1000 are actually sick. For the sick people, a test was positive**

**for 620 and negative for 380. For the healthy people, (the same test was**

**positive for 180 and negative for 8820.**

**Construct a confusion matrix for the data and compute the**

**° Prevalence, Accuracy**

**• Misclassification rate, Sensitivity**

**• Specificity. False Positive Rate (FPR):**

**False Negative Rate. Precision.**

**• Negative Predictive Value (NPV)**

Sure, let's start by constructing the confusion matrix:

	Predicted Sick	Predicted Healthy
Actually Sick	620	380

	Predicted Sick	Predicted Healthy
Actually Healthy	180	8820

Now, let's compute the requested metrics:

1. **Prevalence** is the proportion of the population that has the condition (in this case, the flu). It's calculated as the number of actual positive cases divided by the total number of cases. So,  $\text{Prevalence} = 1000 / 10000 = 0.1$  or 10%.
2. **Accuracy** is the proportion of true results (both true positives and true negatives) in the population. It's calculated as  $(\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$ . So,  $\text{Accuracy} = (620 + 8820) / 10000 = 0.944$  or 94.4%.
3. **Misclassification Rate** (or Error Rate) is the proportion of false results in the population. It's calculated as  $(\text{FP} + \text{FN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$ . So,  $\text{Misclassification Rate} = (180 + 380) / 10000 = 0.056$  or 5.6%.
4. **Sensitivity** (or True Positive Rate or Recall) is the proportion of actual positive cases that were identified correctly. It's calculated as  $\text{TP} / (\text{TP} + \text{FN})$ . So,  $\text{Sensitivity} = 620 / 1000 = 0.62$  or 62%.
5. **Specificity** (or True Negative Rate) is the proportion of actual negative cases that were identified correctly. It's calculated as  $\text{TN} / (\text{TN} + \text{FP})$ . So,  $\text{Specificity} = 8820 / 9000 = 0.98$  or 98%.
6. **False Positive Rate (FPR)** is the proportion of actual negative cases that were incorrectly identified as positive. It's calculated as  $\text{FP} / (\text{FP} + \text{TN})$ . So,  $\text{FPR} = 180 / 9000 = 0.02$  or 2%.
7. **False Negative Rate (FNR)** is the proportion of actual positive cases that were incorrectly identified as negative. It's calculated as  $\text{FN} / (\text{FN} + \text{TP})$ . So,  $\text{FNR} = 380 / 1000 = 0.38$  or 38%.
8. **Precision** (or Positive Predictive Value) is the proportion of predicted positive cases that were correct. It's calculated as  $\text{TP} / (\text{TP} + \text{FP})$ . So,  $\text{Precision} = 620 / (620 + 180) = 0.775$  or 77.5%.
9. **Negative Predictive Value (NPV)** is the proportion of predicted negative cases that were correct. It's calculated as  $\text{TN} / (\text{TN} + \text{FN})$ . So,  $\text{NPV} = 8820 / (8820 + 380) = 0.958$  or 95.8%.

I hope this helps! Let me know if you have any other questions.

i) K-Means can be used to solve

ii) In K-Means. K stands for

iii) K-means clustering aims to partition  $n$  observations into  $k$  clusters in

which each observation belongs to the cluster with the nearest

iv) ----- algorithm has similarity with K-Means?

v) The goal for K-Means cost function is to squared error

function where error function represents distance between data points and

cluster centroid.

ii) How to decide the optimal number of  $K$  in the K mean algorithm?

i) K-Means is a popular unsupervised learning algorithm used for clustering. [It can be applied to data that has a smaller number of dimensions, is numeric, and is continuous<sup>1</sup>. It's typically used in scenarios where you want to make groups of similar things from a randomly distributed collection of things<sup>1</sup>. Some use cases include document classification<sup>1</sup>, and delivery store optimization<sup>1</sup>.](#)

ii) [In K-Means, the 'K' stands for the number of clusters we're trying to identify<sup>234</sup>.](#)

iii) [K-means clustering aims to partition  \$n\$  observations into  \$k\$  clusters in which each observation belongs to the cluster with the nearest mean \(cluster centers or cluster centroid\), serving as a prototype of the cluster<sup>5678</sup>.](#)

iv) The K-Nearest Neighbors (KNN) algorithm has some similarity with K-Means. Both are iterative algorithms. [In K-Means, we need to iteratively choose centroids and assign points to different clusters<sup>9</sup>. Another algorithm similar to K-Means is the Fuzzy c-means algorithm, where objects can belong to more than one cluster<sup>10</sup>.](#)

v) [The goal for the K-Means cost function is to minimize the total squared error function, where the error function represents the squared Euclidean distance between data points and the cluster centroid<sup>11121314</sup>.](#)

vi) [To decide the optimal number of  \$K\$  in the K-Means algorithm, methods like the Elbow Method and the Silhouette Method are commonly used<sup>1516</sup>. The Elbow Method involves running K-Means with different  \$K\$  values and choosing the  \$K\$  for which the Within-Cluster-Sum of Squared Errors \(WCSS\) first starts to diminish<sup>15</sup>.](#)

The Silhouette Method measures how similar a point is to its own cluster compared to other clusters<sup>16</sup>.

## **What are the Steps of Multivariate Regression analysis?**

Multivariate regression analysis involves several steps<sup>1234</sup>:

1. **Select the Features:** Identify the independent variables (features) that are believed to influence the dependent variable<sup>1</sup>.
2. **Normalize the Features:** Scale the features to a certain range (often 0-1) to make the analysis easier<sup>1</sup>. This can be done by changing the value of each feature<sup>1</sup>.
3. **Select Loss Function and Formulate a Hypothesis:** A hypothesis is a predicted value of the response variable, represented by  $h(x)$ <sup>1</sup>. The loss function defines the cost for wrongly predicting the hypothesis<sup>2</sup>. It should be as small as possible<sup>2</sup>.
4. **Minimize the Loss Function:** Use an optimization algorithm (like Gradient Descent) to adjust the parameters of the hypothesis in order to minimize the loss function<sup>2</sup>.
5. **Test the Hypothesis:** The hypothesis function is then tested over a test set to check its correctness and efficiency<sup>2</sup>.

Remember, the goal of multivariate regression is to model the relationship between multiple independent variables and the dependent variable<sup>1</sup>. This technique is commonly used in machine learning, economics, science, and medical research studies<sup>1</sup>.

**Design the suitable model to predict the word in the following sentence.**

**Rahul eats chicken almost every day it should not be hard to sa\ that he is -**

**His sister Lata how ever is lover of sushi and Udon Noodles that means Lata's favorite Cuisine is----- .**

To predict the missing words in the sentences, we can use a language model. Language models are a type of statistical model for generating human language. They are a key component in larger models for Machine Translation, Speech Recognition, Reading Comprehension, and others.



A suitable model for this task could be a Recurrent Neural Network (RNN) or more specifically, a type of RNN called Long Short-Term Memory (LSTM). These models are capable of learning the context from the input sentences and predicting the next word in the sequence.

For example, given the sentence “Rahul eats chicken almost every day it should not be hard to say that he is -”, the model might predict “non-vegetarian” or “a meat-lover” as the missing word, based on the context that Rahul eats chicken almost every day.

Similarly, for the sentence “His sister Lata however is a lover of sushi and Udon Noodles that means Lata’s favorite Cuisine is -”, the model might predict “Japanese” as the missing word, given that sushi and Udon noodles are part of Japanese cuisine.

Please note that the accuracy of the predictions would depend on the quality and quantity of the training data provided to the model. The model needs to be trained on a large corpus of text so that it can learn the patterns and structures in the language. Let me know if you have any other questions!

## **What is Gaussian Mixture Model and When to use?**

A Gaussian Mixture Model (GMM) is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters<sup>12</sup>. It’s a type of mixture model used in statistics, that does not require that the observed dataset needs to identify the subpopulation to which an individual observation belongs<sup>3</sup>. GMM assigns probabilities to data points, allowing them to belong to multiple clusters simultaneously<sup>4</sup>.

In real life, many datasets can be modeled by Gaussian Distribution (Univariate or Multivariate). So it is quite natural and intuitive to assume that the clusters come from different Gaussian Distributions<sup>1</sup>. Or in other words, it tried to model the dataset as a mixture of several Gaussian Distributions<sup>1</sup>.

GMM can be used to model different data types and distributions. It can deal with data that has several peaks or modes, non-spherical clusters, and various modes<sup>5</sup>. The GMM is robust to outliers and can be used for both density estimation and clustering applications<sup>5</sup>. Picture segmentation and anomaly detection can both benefit from it<sup>5</sup>.

However, Gaussian mixture models perform best on datasets where all the features are normally distributed<sup>6</sup>. They should be used for datasets that have multiple interesting features but no specific outcome variable you want to predict<sup>6</sup>. When

[your features are not roughly normally distributed or are categorical, it's better to avoid using Gaussian mixture models<sup>6</sup>.](#)

**Justify with suitable steps that Google Translate app is best example of Natural Language processing (NLP).**

<https://blogs.commonsgorgetown.edu/cctp-607-spring2019/2019/02/27/natural-language-processing-google-translate/>

Google Translate is indeed a prime example of an application that utilizes Natural Language Processing (NLP). Here's how it exemplifies NLP:

1. **Language Understanding:** Google Translate uses NLP to understand the syntax, semantics, and context of the input language<sup>1</sup>. It can accurately interpret the meaning of words and phrases in different languages<sup>1</sup>.
2. **Translation:** Google Translate uses machine learning algorithms to translate text from one language to another<sup>1</sup>. It uses a technique called sequence-to-sequence modeling, which allows the algorithm to convert a sequence of words from one language to another<sup>2</sup>.
3. **Language Generation:** After translation, Google Translate generates text in the target language that maintains the original meaning while following the grammatical rules of the target language<sup>1</sup>.
4. **Continuous Learning:** Google Translate continually learns and improves its translations by analyzing user corrections and new documents<sup>1</sup>.
5. **Handling Ambiguity:** Google Translate uses the context of the surrounding words to handle the ambiguity that often arises in language<sup>1</sup>.
6. **Speech Recognition and Synthesis:** In addition to text translation, Google Translate also offers speech-to-text and text-to-speech features, converting spoken words into text and vice versa<sup>3</sup>.
7. **Multilingual Support:** Google Translate supports over 100 languages, demonstrating its robustness and wide applicability<sup>3</sup>.

These features make Google Translate a comprehensive NLP application that covers a wide range of NLP tasks

**Solve 8 puzzle problems using the steepest ascent hill climbing. Use the misplaced tiles heuristic.**

[https://www.youtube.com/watch?v=Rm3cWbB03KM&ab\\_channel=MuhammadUmarFarooq](https://www.youtube.com/watch?v=Rm3cWbB03KM&ab_channel=MuhammadUmarFarooq)

**Define Artificial Intelligence laid out along two dimensions covering all four categories or approaches of AI.**

**Also, Explain each approach of AI definitions briefly.**

Artificial Intelligence (AI) can be defined along two dimensions<sup>1234</sup>:

1. **Thinking vs. Acting:** This dimension refers to whether the AI is designed to think like a human (cognitive modeling) or to act like a human (Turing Test approach). The former involves understanding human cognition and the latter involves creating machines that behave indistinguishably from a human<sup>5</sup>.
2. **Human-like vs. Rational:** This dimension refers to whether the AI is designed to replicate human behavior (with all its flaws) or to behave in an ideal, rational manner. The former involves creating systems that reproduce human behavior, while the latter involves creating systems that make the most rational decision given the available information<sup>5</sup>.

Based on these dimensions, there are four main approaches to AI:

1. **Thinking Humanly:** This approach involves cognitive modeling and requires scientific theories from cognitive science. It aims to build systems that mimic the human thought process<sup>5</sup>.
2. **Thinking Rationally:** This approach involves the use of logic and reasoning. It aims to build systems that mimic the rational thinking process, making decisions based on logical inference or optimization of some criteria<sup>5</sup>.
3. **Acting Humanly:** This approach involves passing the Turing Test and requires the system to interact like a human. It aims to build systems that can not only solve problems but also exhibit behavior that a human observer would deem “intelligent”, such as understanding natural language, learning from experience, and even displaying emotions<sup>5</sup>.
4. **Acting Rationally:** This approach involves rational agents and is also known as the “intelligent agent” approach. It aims to build systems that act rationally, meaning they do the right thing given their perceptual inputs and the goals they have been set<sup>5</sup>.

Each of these approaches has its own strengths and weaknesses, and the choice between them often depends on the specific problem at hand<sup>5</sup>.

Apply Uninformed Search algorithms on a Binary tree(branching factor=2) where nodes are in the sequence of A to Z, where M is the only Goal.

Draw the portion of the state space for states A to Z. Stop whenever you reach goal state 'M'. List the order in which nodes will be visited for:

1- BFS      2- DFS      3-Iterative Deepening

Compare these algorithms with respect to evaluation criteria to measure the performance of search algorithms

Given Confusion metrics for cancer prediction example, for 100 patients. Calculate 1-Precision, 2- Recall, 3- F1-score, and also write the interpretation of each metric

		Actual (True) Values	
		Cancer	No Cancer
Predicted Values	Cancer	45	18
	No Cancer	12	25

## State the drawbacks of the hill Climbing algorithm

The Hill Climbing algorithm, while simple and easy to implement, has several limitations<sup>1234</sup>:

1. **Local Optima:** The algorithm can get stuck in local optima, meaning it may not find the global optimum of the problem<sup>134</sup>. This happens when the algorithm reaches a peak in the search space where all neighboring states have lower values, but it's not the highest possible peak<sup>1</sup>.
2. **Lack of Exploration:** Hill Climbing does not explore the search space very thoroughly, which can limit its ability to find better solutions<sup>23</sup>. It only considers immediate neighbors and does not look ahead beyond them<sup>23</sup>.
3. **Ridges:** These are sequences of local maxima, making it difficult for the algorithm to navigate<sup>1</sup>.

4. **Plateaux:** This is a flat state-space region. As there is no uphill to go, the algorithm often gets lost in the plateau<sup>1</sup>.
5. **Initial Solution Sensitivity:** The algorithm is sensitive to the choice of the initial solution, and a poor initial solution may result in a poor final solution<sup>2</sup>.

To overcome these limitations, many variants of the Hill Climbing algorithm have been developed, such as Stochastic Hill Climbing, First-Choice Climbing, and Random-restart hill climbing<sup>1</sup>.

**Give a representation in First order predicate logic of the following propositions, in the form of Horn clauses;**

**also in this case, first of all clearly define the domain of discourse and the**

**predicate, function, and constant symbols you intend to use.**

**(a) Cows, pigs and horses are mammals**

**(b) The child of a horse is a horse**

**(c) Bluebeard is a horse**

**(d) Bluebeard is Charlie's father**

**(e) Child and father are inverse relations**

**(f) Every mammal has a father**

**Prove that the proposition "Charlie is a mammal" is entailed by the above**

**propositions, using the backward chaining inference algorithm.**

Let's define the domain of discourse and the predicate, function, and constant symbols:

- Domain of Discourse: All living organisms.
- Predicate Symbols:  $Mammal(x)$ ,  $Horse(x)$ ,  $Cow(x)$ ,  $Pig(x)$ ,  $Child(x, y)$ ,  $Father(x, y)$ .

- Constant Symbols: Bluebeard, Charlie.

Now, let's represent the given propositions in First Order Predicate Logic (FOPL):

(a)  $\forall x (Cow(x) \vee Pig(x) \vee Horse(x) \rightarrow Mammal(x))$  (b)  $\forall x \forall y (Child(y, x) \wedge Horse(x) \rightarrow Horse(y))$  (c)  $Horse(Bluebeard)$  (d)  $Father(Bluebeard, Charlie)$  (e)  $\forall x \forall y (Child(x, y) \leftrightarrow Father(y, x))$  (f)  $\forall x (Mammal(x) \rightarrow \exists y Father(y, x))$

To convert these into Horn clauses, we need to ensure each clause has at most one positive literal:

(a)  $\neg Cow(x) \vee \neg Pig(x) \vee \neg Horse(x) \vee Mammal(x)$  (b)  $\neg Child(y, x) \vee \neg Horse(x) \vee Horse(y)$  (c)  $Horse(Bluebeard)$  (d)  $Father(Bluebeard, Charlie)$  (e)  $\neg Child(x, y) \vee Father(y, x)$  and  $\neg Father(y, x) \vee Child(x, y)$  (f)  $\neg Mammal(x) \vee Father(F, x)$  (where  $F$  is a new function symbol representing the father of  $x$ )

Now, let's prove that the proposition "Charlie is a mammal" is entailed by the above propositions using the backward chaining inference algorithm:

1. Start with the goal  $Mammal(Charlie)$ .
2. From (f), we can infer  $Father(F, Charlie)$ .
3. From (d), we know  $Father(Bluebeard, Charlie)$ , so  $F$  must be Bluebeard.
4. From (c), we know  $Horse(Bluebeard)$ .
5. From (b), since Bluebeard is a horse and Charlie is a child of Bluebeard, we can infer  $Horse(Charlie)$ .
6. From (a), since Charlie is a horse, we can infer  $Mammal(Charlie)$ .

So, the proposition "Charlie is a mammal" is indeed entailed by the given propositions. Let me know if you have any other questions!

**Consider EnjoySport learning task.**

**Given that the attribute Sky has three possible values, and that AirTemp,**

**Humidity, Wind, Water, and Forecast each have two possible values**

**Consider the above learning task and Define the following**

**1- Instance Space,**

**2- Hypothesis space,**

**3- Version Space**

Sure, let's define these terms in the context of the EnjoySport learning task:

1. **Instance Space:** The instance space is the set of all possible instances that can be represented by the given attributes. In this case, each instance is a specific day described by the attributes Sky, AirTemp, Humidity, Wind, Water, and Forecast. Since Sky has three possible values and the other attributes each have two possible values, the size of the instance space would be  $3 * 2^5 = 96$ . So, the instance space consists of these 96 different combinations of attribute values.
2. **Hypothesis Space:** The hypothesis space is the set of all hypotheses that can be represented by the learning algorithm. In the context of EnjoySport, a hypothesis is a statement about which combinations of attribute values would result in a day being suitable for playing the sport. If we're using a conjunctive representation (i.e., a day is suitable for the sport if and only if certain conditions on the attributes are met), then the hypothesis space would consist of all possible conjunctions of attribute values.
3. **Version Space:** The version space is the subset of the hypothesis space that is consistent with the observed training examples. In other words, it's the set of all hypotheses in the hypothesis space that correctly classify the training examples. As more training examples are observed, the version space typically becomes smaller, converging on the true hypothesis.

I hope this helps! Let me know if you have any other questions.

## MSE OCT 22-23

Explain the Properties of the Task environment with suitable examples of each.

We can identify fairly small number of dimensions along which task environments can be categorized.

### 1] Fully observable Vs Partially Observable.

If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is fully observable. A task environment is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action, relevance depends on the performance measure. Fully observable environments are convenient because the agent need not maintain any internal state to keep track of

the world. An environment might be partially observable because of noisy and inaccurate sensors missing from the sensor data.

**Example:** A vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares.

## 2] Deterministic Vs stochastic.

If the next state of the environment is completely determined by the current state and the action executed by the agent then we say the environment is deterministic otherwise it is stochastic.

## 3] Episodic Vs Sequential.

In an episodic task environment, the agent experience is divided into atomic episodes each episode consists of the agent perceiving and then performing a single action. Crucially, the next episode does not depend on the actions taken in previous episodes. In episodic environment, the choice of action in each episode depends only on the episode itself. Many classification tasks are episodes.

**Example:** An agent that has to spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions moreover, the current decision doesn't affect whether the next part is defective. In sequential environments, the current decision could affect all future decisions.

**Example:** Chess and taxi driving are sequential, in both cases, short term actions can have long term consequences.

Episodic environment are much simpler than sequential environments because the agent does not need to think ahead.

## 4] Static Vs Dynamics.

If the environment can change while an agent is deliberating then we say the environment is dynamic for that agent otherwise, it is static. Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action, nor need it worry about the passage of time. Dynamic environments are continuously asking the agent what it wants to do, if it has not decided yet, that counts as deciding to do nothing. If the environment itself does not change with the passage of time but the agents performance score does, then we say the environment is semi dynamic.

**Example:** Taxi driving is clearly dynamic, the other cars and the taxi itself keep moving while the driving algorithm dithers about what to do next. Chess when played with a clock is semi dynamic crossword, puzzles are static.

## 5] Discrete Vs Continuous.

The discrete/continuous distinction can be applied to the state of the environment, to the way time is handled and to the percepts and actions of the agent.

**Example:** A discrete state environment such as a chess game has a finite no of distinct states chess also has a discrete set of percepts and actions.



Taxi driving is a continuous state and continuous time problem: the speed and location of the taxi and of the other vehicles sweep through a range of continuous values and so smoothly over time. Taxi driving actions are also continuous. Input from digital cameras is discrete.

#### **6] Single Agent Vs Multi Agent.**

The distinction between single agent and multi agent environment may seem simple enough.

**Example:** An agent solving a crossword puzzle by itself is clearly in a single agent chess is in a two agents environment. Chess is a competitive multi agent environment. In a taxi driving environment, avoiding collisions maximizes the performance measure. Of all agents, so it is partially cooperative multi agent environment.

**Describe the following agents with a neat diagram**

**1- Utility-based agent**

**2- Learning agent.**

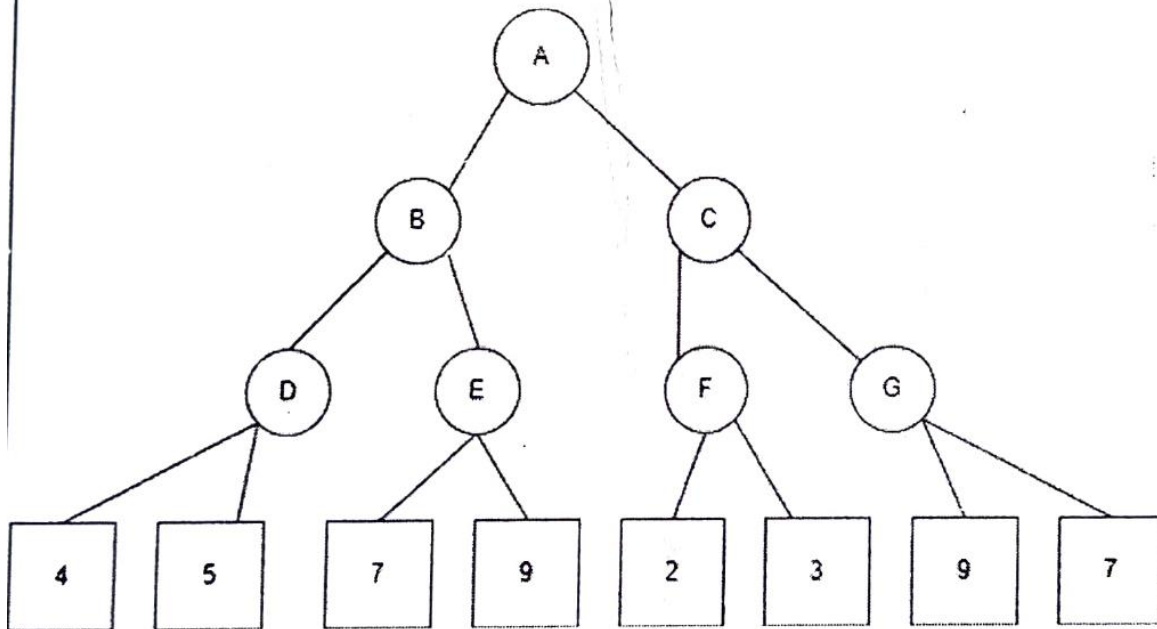
---

<https://www.geeksforgeeks.org/agents-artificial-intelligence/>

**Describe different criteria that are used to evaluate the performance of searching algorithms.**

**Describe the different criteria that are used to evaluate the performance of searching algorithms**

3- Given the utility values at the leaf node, apply the alpha-beta pruning algorithm for the figure below. Find the alpha value, beta value, and node value of each node. Also, show the pruning criteria applied along with pruned nodes and subtrees.



**What do you understand by payoff?**

[https://www.youtube.com/watch?v=4IEl4vDkML8&ab\\_channel=WilliamSpaniel](https://www.youtube.com/watch?v=4IEl4vDkML8&ab_channel=WilliamSpaniel)

**Describe the different criteria that are used to evaluate the performance of searching algorithms**

Sure, the performance of search algorithms can be evaluated based on several criteria<sup>123</sup>:

1. **Time Complexity**: This is the computational complexity that describes the amount of computational time taken by an algorithm to run, as a function of the size of the input to the program<sup>1</sup>.
2. **Space Complexity**: This refers to the amount of memory space that an algorithm needs to execute<sup>1</sup>.
3. **Optimality**: This measures whether the algorithm is able to find the best solution to the problem<sup>1</sup>.
4. **Completeness**: This checks if the algorithm is able to find a solution if one exists<sup>1</sup>.

5. **Branching Factor:** This is the maximum number of successors of any state of the search tree<sup>2</sup>.
6. **Path Length:** This is the minimal length of the path in the state space between the initial and a goal state<sup>2</sup>.
7. **Depth:** This is the depth of the shallowest goal node in the search tree<sup>2</sup>.

Each of these criteria provides a different perspective on the performance of the algorithm, and they are often used in combination to fully evaluate the effectiveness of a search algorithm.

## What is adversarial search?

<https://www.javatpoint.com/ai-adversarial-search>

Adversarial search is a type of search in artificial intelligence where we examine the problem that arises when we try to plan ahead in the world and other agents are planning against us<sup>12</sup>. This situation usually occurs in game playing, where more than one agent is searching for the solution in the same search space<sup>12</sup>. Each agent needs to consider the action of other agents and the effect of that action on their performance<sup>12</sup>.

Adversarial searches, often known as games, are modeled as a search problem and a heuristic evaluation function<sup>12</sup>. These are the two main factors which help to model and solve games in AI<sup>12</sup>.

### Types of Games in AI<sup>12</sup>:

- **Deterministic games:** These games follow a strict pattern and set of rules for the games, and there is no randomness associated with them. Examples are chess, Checkers, Go, tic-tac-toe, etc.
- **Non-deterministic games:** These games have various unpredictable events and have a factor of chance or luck. This factor of chance or luck is introduced by either dice or cards. These are random, and each action response is not fixed. Such games are also called as stochastic games. Example: Backgammon, Monopoly, Poker, etc.
- **Perfect information games:** In these games, agents can look into the complete board. Agents have all the information about the game, and they can see each other moves also. Examples are Chess, Checkers, Go, etc.
- **Imperfect information games:** In these games, agents do not have all information about the game and not aware with what's going on, such as tic-tac-toe, Battleship, blind, Bridge, etc.

In adversarial search, we often discuss deterministic games, fully observable environment, zero-sum, and where each agent acts alternatively<sup>12</sup>. In a **Zero-Sum**

Game, each agent's gain or loss of utility is exactly balanced by the losses or gains of utility of another agent<sup>12</sup>. One player of the game tries to maximize one single value, while the other player tries to minimize it<sup>12</sup>. Each move by one player in the game is called as ply<sup>12</sup>. Chess and tic-tac-toe are examples of a Zero-sum game<sup>12</sup>.

**What do you mean by Multicollinearity? Identify the regularization technique which deals with multicollinearity.**

Multicollinearity is a statistical concept where several independent variables in a model are correlated<sup>1</sup>. It is a phenomenon in which one predictor variable in a multiple regression model can be perfectly predicted from the others<sup>2</sup>. This can lead to less reliable statistical inferences as it can be hard to determine how the independent variables influence the dependent variable individually<sup>1</sup>.

Regularization techniques are used to address the issue of multicollinearity in linear regression<sup>3</sup>. Specifically, **Ridge Regression**, also known as **L2 regularization**, is one of several types of regularization for linear regression models<sup>4</sup>. It corrects for multicollinearity in regression analysis<sup>4</sup>. This is useful when developing machine learning models that have a large number of parameters, particularly if those parameters also have high weights<sup>4</sup>. Another regularization technique that can be used is **Lasso Regression**, also known as **L1 regularization**<sup>5</sup>. These techniques implement a means to control regression coefficients, decrease the variance, and reduce sample error to tackle multicollinearity<sup>6</sup>.

**What are the requirements of clustering algorithms.**

Clustering algorithms have several key requirements:

1. **Scalability:** Some clustering algorithms work well on small data sets, but a large database can include millions of objects. Highly scalable clustering algorithms are required<sup>123</sup>.
2. **Ability to deal with different types of attributes:** Some algorithms are designed to cluster interval-based (numerical) data. However, applications may require clustering several types of data, including binary, categorical (nominal), and ordinal data, or a combination of these data types<sup>123</sup>.
3. **Discovery of clusters with arbitrary shape:** Some clustering algorithms determine clusters based on Euclidean or Manhattan distance measures. These algorithms tend to discover spherical clusters with the same size and density. But, a cluster can be of any shape<sup>12</sup>.
4. **Minimal requirements for domain knowledge to determine input parameters:** Some clustering algorithms require users to input specific

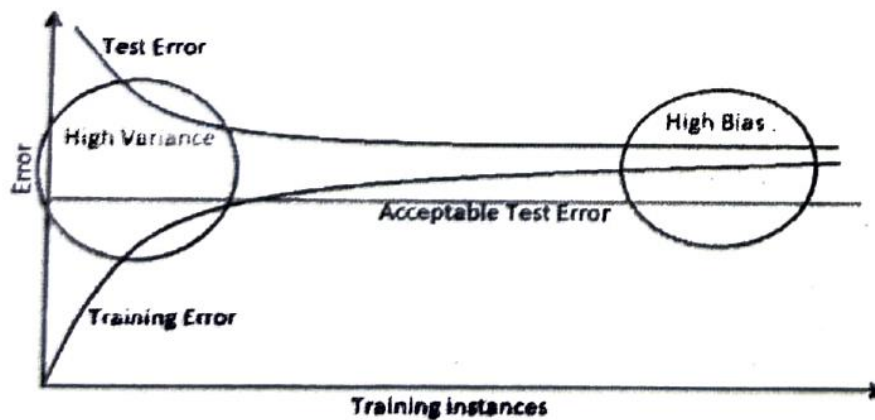
parameters in cluster analysis (including the number of desired clusters).  
[The clustering results can be sensitive to input parameters<sup>12</sup>.](#)

5. **Ability to deal with noisy data:** Most real-world databases include outliers or missing, unknown, or erroneous data. [Some clustering algorithms are sensitive to such data and can lead to clusters of poor quality<sup>123</sup>.](#)
6. **[Incremental clustering and insensitivity to the order of input records:](#)** [Some clustering algorithms cannot include newly inserted data into current clustering structures and instead must decide a new clustering from scratch<sup>12</sup>.](#)
7. **High dimensionality:** A database or a data warehouse can include multiple dimensions or attributes. [Some clustering algorithms are good at managing low-dimensional data, but finding clusters of data objects in high-dimensional space is complex<sup>123</sup>.](#)

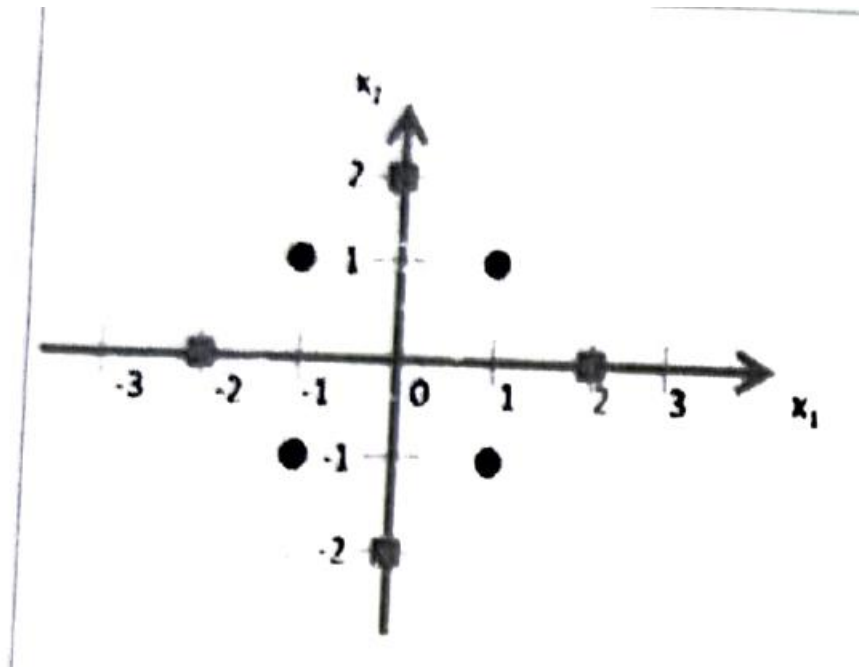
Find linear regression equation for the following two sets of data:

x	2	4	6	8
y	3	7	5	10

How to identify High Variance or High Bias from given figure



Apply the suitable SVM on the following datapoints



What are the characteristics of the problems suited for decision tree learning?

Build the decision tree for the Boolean functions

$$A \vee [B \wedge C]$$

$$A \text{ XOR } B$$

$$[A \wedge B] \vee [C \wedge D]$$

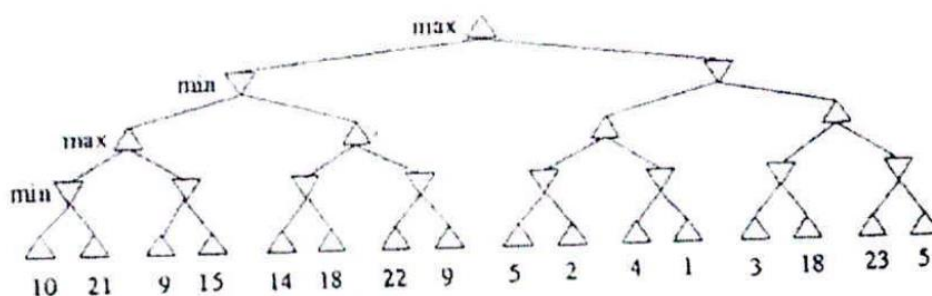


Apply the Naïve Bay's classifier on the following dataset.

Weather	Car	Class
Sunny	Working	Go-out
Rainy	Broken	Go-out
Sunny	Working	Go-out
Sunny	Working	Go-out
Sunny	Working	Go-out
Rainy	Broken	Stay-home
Rainy	Broken	Stay-home
Sunny	Working	Stay-home
Sunny	Broken	Stay-home
Rainy	Broken	Stay-home

Aug 2023

Suggest the optimal decision for the given game tree using Alpha-beta Pruning. At each point, show the range of possible values for each node i.e.  $\alpha$ ,  $\beta$ , and pruned nodes along with pruning criteria.



Represent the following sentences in first order logic, use a consistent vocabulary?

(which you must define):



- a. Some students took French in spring 2001.**
- b. Every student who takes French passes it.**
- c. Only one student took Greek in spring 2001.**
- d. The best score in Greek is always higher than the best score in French.**
- e. Every person who buys a policy is smart.**
- f. No person buys an expensive policy.**

- $\text{Student}(x)$ :  $x$  is a student.
- $\text{Took}(x, y, z)$ :  $x$  took  $y$  in  $z$ .
- $\text{Passes}(x, y)$ :  $x$  passes  $y$ .
- $\text{Greek}(x)$ :  $x$  is Greek.
- $\text{French}(x)$ :  $x$  is French.
- $\text{BestScore}(x, y)$ :  $x$  has the best score in  $y$ .
- $\text{HigherThan}(x, y)$ :  $x$  is higher than  $y$ .
- $\text{Person}(x)$ :  $x$  is a person.
- $\text{Buys}(x, y)$ :  $x$  buys  $y$ .
- $\text{Smart}(x)$ :  $x$  is smart.
- $\text{Expensive}(x)$ :  $x$  is expensive.

Now, let's represent the sentences in first-order logic:

a. Some students took French in spring 2001:

$\exists x(\text{Student}(x) \wedge \text{Took}(x, \text{French}, \text{Spring2001}))$

b. Every student who takes French passes it:

$\forall x((\text{Student}(x) \wedge \text{Takes}(x, \text{French})) \rightarrow \text{Passes}(x, \text{French}))$

c. Only one student took Greek in spring 2001:

$\exists x(\text{Student}(x) \wedge \text{Took}(x, \text{Greek}, \text{Spring2001}) \wedge \forall y((\text{Student}(y) \wedge \text{Took}(y, \text{Greek}, \text{Spring2001})) \rightarrow (x = y)))$

d. The best score in Greek is always higher than the best score in French:

$\forall x \forall y((\text{BestScore}(x, \text{Greek}) \wedge \text{BestScore}(y, \text{French})) \rightarrow \text{HigherThan}(x, y))$

e. Every person who buys a policy is smart:

$\forall x((\text{Person}(x) \wedge \text{Buys}(x, \text{Policy})) \rightarrow \text{Smart}(x))$

f. No person buys an expensive policy:

$\forall x((\text{Person}(x) \wedge \text{Expensive}(\text{Policy})) \rightarrow \neg \text{Buys}(x, \text{Policy}))$

**For each of the following activities, give PEAS description of the task**

**environment and characterize it in terms of properties of task environment.**

**1- Knitting a Sweater 2- Performing High Jump 3- Practicing tennis against wall**

Sure, let's define the PEAS (Performance measure, Environment, Actuators, Sensors) for each activity and characterize them in terms of properties of the task environment:

**1. Knitting a Sweater**

- Performance Measure: Quality and correctness of the pattern, speed of knitting, efficient use of yarn.
- Environment: The knitting area, including the knitting needles, yarn, and pattern instructions.
- Actuators: Hands (to manipulate the needles and yarn).
- Sensors: Eyes (to follow the pattern and check the knitting), touch (to feel the texture and tension of the yarn).
- Properties: Fully Observable, Deterministic, Sequential, Single-agent, Static, Discrete.

**2. Performing High Jump**

- Performance Measure: Height of the jump, correct form, safe landing.
- Environment: The track and field area, including the run-up track, the bar, and the landing area.
- Actuators: Legs (for running and jumping), arms (for maintaining balance and form).
- Sensors: Eyes (to judge the distance and height), proprioception (to sense body position and movement), vestibular system (for balance).
- Properties: Fully Observable, Stochastic, Sequential, Single-agent, Dynamic, Continuous.

**3. Practicing Tennis Against a Wall**

- Performance Measure: Accuracy of shots, consistency of returns, variety of strokes practiced.

- Environment: The practice area, including the wall, the tennis racket, and the ball.
- Actuators: Arms (to swing the racket), legs (for positioning and movement).
- Sensors: Eyes (to track the ball), touch (to feel the racket grip), proprioception (to sense body position and movement).
- Properties: Fully Observable, Stochastic, Sequential, Single-agent, Dynamic, Continuous.

These characterizations can help in designing an artificial agent to perform these tasks. They provide a framework for understanding what the agent needs to be able to sense and do, and what it will be evaluated on. The properties of the task environment further help in deciding the type of algorithms or methods that can be used for these tasks. For example, deterministic environments can often be solved with search or planning algorithms, while stochastic environments may require probabilistic reasoning or learning algorithms. Similarly, static environments allow the agent to take its time to decide on an action, while dynamic environments require real-time decision making. Discrete environments usually have a finite number of states and actions, while continuous environments require dealing with ranges of values.

**Describe Learning agent model with suitable diagram.**

<https://www.geeksforgeeks.org/agents-artificial-intelligence/>

Find the most cost-effective path to reach the final state from initial state using A\* Algorithm.

Consider  $g(n)$  = Depth of node and  $h(n)$  = Number of misplaced tiles

2	8	3
1	6	4
7		5

Initial State

1	2	3
8		4
7	6	5

Final State

For the following English statements choose the correct FOL statements. And give Justification for the incorrect statement stating the reason.

**a. Paris and Marseilles are both in France.**

- (i)  $\text{In}(\text{Paris} \wedge \text{Marseilles}, \text{France})$ .
- (ii)  $\text{In}(\text{Paris}, \text{France}) \wedge \text{In}(\text{Marseilles}, \text{France})$ .
- (iii)  $\text{In}(\text{Paris}, \text{France}) \vee \text{In}(\text{Marseilles}, \text{France})$ .

**b. There is a country that borders both Iraq and Pakistan.**

- (i)  $\exists c \text{ Country}(c) \wedge \text{Border}(c, \text{Iraq}) \wedge \text{Border}(c, \text{Pakistan})$ .
- (ii)  $\exists c \text{ Country}(c) \Rightarrow [\text{Border}(c, \text{Iraq}) \wedge \text{Border}(c, \text{Pakistan})]$ .
- (iii)  $[\exists c \text{ Country}(c)] \Rightarrow [\text{Border}(c, \text{Iraq}) \wedge \text{Border}(c, \text{Pakistan})]$ .
- (iv)  $\exists c \text{ Border}(\text{Country}(c), \text{Iraq} \wedge \text{Pakistan})$ .

**c. All countries that border Ecuador are in South America.**

- (i)  $\forall c \text{ Country}(c) \wedge \text{Border}(c, \text{Ecuador}) \Rightarrow \text{In}(c, \text{SouthAmerica})$ .
- (ii)  $\forall c \text{ Country}(c) \Rightarrow [\text{Border}(c, \text{Ecuador}) \Rightarrow \text{In}(c, \text{SouthAmerica})]$ .
- (iii)  $\forall c [\text{Country}(c) \Rightarrow \text{Border}(c, \text{Ecuador})] \Rightarrow \text{In}(c, \text{SouthAmerica})$ .
- (iv)  $\forall c \text{ Country}(c) \wedge \text{Border}(c, \text{Ecuador}) \wedge \text{In}(c, \text{SouthAmerica})$ .

**d. No region in South America borders any region in Europe.**

- (i)  $\neg[\exists c, d \text{ In}(c, \text{SouthAmerica}) \wedge \text{In}(d, \text{Europe}) \wedge \text{Borders}(c, d)]$ .
- (ii)  $\forall c, d [\text{In}(c, \text{SouthAmerica}) \wedge \text{In}(d, \text{Europe})] \Rightarrow \neg \text{Borders}(c, d)$ .

- (iii)  $\neg \forall c \text{ In}(c, \text{SouthAmerica}) \Rightarrow \exists d \text{ In}(d, \text{Europe}) \wedge \neg \text{Borders}(c, d)$ .

- (iv)  $\forall c \text{ In}(c, \text{SouthAmerica}) \Rightarrow \forall d \text{ In}(d, \text{Europe}) \Rightarrow \neg \text{Borders}(c, d)$ .

**e. No two adjacent countries have the same map color.**

- (i)  $\forall x, y \neg \text{Country}(x) \vee \neg \text{Country}(y) \vee \neg \text{Borders}(x, y) \vee \neg (\text{MapColor}(x) = \text{MapColor}(y))$ .
- (ii)  $\forall x, y (\text{Country}(x) \wedge \text{Country}(y) \wedge \text{Borders}(x, y) \wedge \neg(x = y)) \Rightarrow \neg (\text{MapColor}(x) = \text{MapColor}(y))$ .
- (iii)  $\forall x, y \text{ Country}(x) \wedge \text{Country}(y) \wedge \text{Borders}(x, y) \wedge \neg (\text{MapColor}(x) = \text{MapColor}(y))$ .
- (iv)  $\forall x, y (\text{Country}(x) \wedge \text{Country}(y) \wedge \text{Borders}(x, y)) \Rightarrow \text{MapColor}(x) \neq \text{MapColor}(y)$ .

Give a sequence of S and G boundary sets computed by applying the CANDIDATE-ELIMINATION algorithm if it is given the sequence of examples in the order in which they appear on the table.

STATUS	FLOOR	DEPT.	OFFICE SIZE	RECYCLING BIN?
Faculty	four	cs	medium	yes
Faculty	four	ee	medium	yes
Student	four	cs	small	no
Faculty	five	cs	medium	yes

**Apply K means algorithm on given data set for K=3,**

**use C1(2),C2(6) and C3(15) as initial cluster centers.**

**Data :{2,3,6,7,8,9,12,15,18,22}**

Sure, let's apply the K-means algorithm on the given data set with K=3 and initial cluster centers C1(2), C2(6), and C3(15).

The K-means algorithm involves the following steps:

1. Initialize the centroids. In this case, they are given as C1(2), C2(6), and C3(15).
2. Assign each data point to the nearest centroid. This forms K clusters.
3. Recalculate the centroid of each cluster.
4. Repeat steps 2 and 3 until the centroids do not change significantly or a maximum number of iterations is reached.

Let's apply these steps to your data:

Data: {2,3,6,7,8,9,12,15,18,22}

### Iteration 1:

- Initial Centroids: C1(2), C2(6), C3(15)
- Assign each data point to the nearest centroid:
  - Cluster 1: {2,3}
  - Cluster 2: {6,7,8,9}
  - Cluster 3: {12,15,18,22}
- Recalculate centroids:
  - New C1:  $(2+3)/2 = 2.5$
  - New C2:  $(6+7+8+9)/4 = 7.5$
  - New C3:  $(12+15+18+22)/4 = 16.75$

### Iteration 2:

- Current Centroids: C1(2.5), C2(7.5), C3(16.75)
- Assign each data point to the nearest centroid:
  - Cluster 1: {2,3}
  - Cluster 2: {6,7,8,9,12}
  - Cluster 3: {15,18,22}
- Recalculate centroids:
  - New C1:  $(2+3)/2 = 2.5$  (no change)
  - New C2:  $(6+7+8+9+12)/5 = 8.4$
  - New C3:  $(15+18+22)/3 = 18.33$

### Iteration 3:

- Current Centroids: C1(2.5), C2(8.4), C3(18.33)
- Assign each data point to the nearest centroid:
  - Cluster 1: {2,3,6}
  - Cluster 2: {7,8,9,12}
  - Cluster 3: {15,18,22}
- Recalculate centroids:
  - New C1:  $(2+3+6)/3 = 3.67$
  - New C2:  $(7+8+9+12)/4 = 9$
  - New C3:  $(15+18+22)/3 = 18.33$  (no change)

### Iteration 4:

- Current Centroids: C1(3.67), C2(9), C3(18.33)
- Assign each data point to the nearest centroid:
  - Cluster 1: {2,3,6}
  - Cluster 2: {7,8,9,12}
  - Cluster 3: {15,18,22}
- Recalculate centroids:
  - New C1:  $(2+3+6)/3 = 3.67$  (no change)

- New C2:  $(7+8+9+12)/4 = 9$  (no change)
- New C3:  $(15+18+22)/3 = 18.33$  (no change)

At this point, the centroids have not changed after an iteration, so we can conclude that the algorithm has converged. The final clusters are:

- Cluster 1: {2,3,6} with centroid at 3.67
- Cluster 2: {7,8,9,12} with centroid at 9
- Cluster 3: {15,18,22} with centroid at 18.33

5(a) Apply Naive Bayesian Classification algorithm on a given dataset. Sugar is the target label. The target label has two possibilities 0 and 1. 0 means the diabetic patient has no sugar and 1 means the diabetic patient has sugar (1- He is a diabetic patient).

Predict the response for the person with BMI=43.6, Age=40 .

BMI	Age	Sugar
33.6	50	1
26.6	30	0
23.4	40	0
43.1	67	0
35.3	23	1
35.9	67	1
36.7	45	1
25.7	46	0
23.3	29	0
31	56	1

## Describe various properties of task environments

Sure, task environments in AI have several key properties:



1. **Fully Observable vs Partially Observable:** If an agent's sensors can access the complete state of the environment at each point in time, it is a fully observable environment. Otherwise, it is partially observable<sup>1</sup>.
2. **Deterministic vs Stochastic:** If the current state of the agent uniquely determines the next state, the environment is deterministic. A stochastic environment is random and cannot be completely determined by the agent<sup>1</sup>.
3. **Competitive vs Collaborative:** In a competitive environment, an agent competes against another agent to optimize the output. In a collaborative environment, multiple agents cooperate to produce the desired output<sup>1</sup>.
4. **Single-agent vs Multi-agent:** A single-agent environment consists of only one agent, while a multi-agent environment involves more than one agent<sup>1</sup>.
5. **Static vs Dynamic:** A static environment does not change over time, while a dynamic environment constantly changes when the agent is performing some action<sup>1</sup>.
6. **Discrete vs Continuous:** If an environment consists of a finite number of actions to obtain the output, it is a discrete environment<sup>1</sup>.
7. **Episodic vs Sequential:** In an episodic environment, the episodes are independent of each other. The action performed in one episode doesn't affect subsequent episodes. In a sequential environment, the current action could affect all future decisions<sup>1</sup>.

Obtain the equation of regression line for the following data. Calculate value of correlation coefficient. Estimate glucose level for age=55 using the regression line equation.

Age (X)	43	21	25	42	57	59
Glucose Level(Y)	99	65	79	75	87	81

## Compare Bias and Variance with suitable examples.

Sure, let's compare bias and variance in the context of machine learning:

### Bias:

- Bias is a type of error that occurs due to wrong assumptions about data<sup>1</sup>.
- It is the inability of the model to capture the true relationship between the data points<sup>2</sup>.



- [High bias means more assumptions are taken to build the target function, leading to an underfitting model<sup>1</sup>.](#)
- [For example, a linear regression model may have a high bias if the data has a non-linear relationship<sup>1</sup>.](#)
- [Ways to reduce high bias include using a more complex model, increasing the number of features, and reducing regularization of the model<sup>1</sup>.](#)

### **Variance:**

- [Variance describes the situation in which the model accounts for the variations in the data as well as the noise<sup>3</sup>.](#)
- [It measures the model's sensitivity to fluctuations in the data<sup>4</sup>.](#)
- [High variance means the model is highly sensitive to variations in training data, leading to an overfitting model<sup>1</sup>.](#)
- [For example, if a model is allowed to view the data too many times, it will learn very well for only that data<sup>4</sup>.](#)
- [Ways to reduce high variance include using simpler models, reducing the number of features, and increasing regularization of the model<sup>1</sup>.](#)

In summary, bias and variance are two types of errors in machine learning that need to be balanced. [A model with high bias oversimplifies the data and misses relevant relations between features and target outputs \(underfitting\), while a model with high variance pays too much attention to the noise in the training data \(overfitting\)<sup>1</sup>.](#)

### **How Support Vector Machine are used as Classification Model.**

[Support Vector Machine \(SVM\) is a powerful supervised machine learning algorithm used for classification<sup>12345</sup>.](#) Here's how it works:

1. **Training:** [SVM takes a set of labeled training data and tries to find the optimal hyperplane which separates the different classes in the feature space<sup>1</sup>. The dimension of the hyperplane depends upon the number of features<sup>1</sup>.](#)
2. **Classification:** [Once the model is trained, it can classify new instances based on which side of the hyperplane they fall on<sup>1</sup>. The decision function of SVM depends on some subset of the training data, called the support vectors<sup>2</sup>.](#)
3. **Handling Outliers:** [SVM has the characteristic to ignore outliers and finds the best hyperplane that maximizes the margin<sup>1</sup>.](#)
4. **Kernel Trick:** [SVMs can handle non-linear classification using what is known as the kernel trick. The kernel trick involves transforming the data into a higher-dimensional space where it is easier to classify with a linear decision boundary<sup>1</sup>.](#)

5. **Multidimensional Space:** SVMs are effective in high dimensional spaces and cases where the number of dimensions is greater than the number of samples<sup>2</sup>.
6. **Memory Efficient:** SVMs use a subset of training points in the decision function (called support vectors), so it is also memory efficient<sup>2</sup>.

SVMs can be used for a variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection<sup>1</sup>. They are adaptable and efficient in a variety of applications because they can manage high-dimensional data and nonlinear relationships<sup>1</sup>.

**ESE DEC 22**

## **Why is it necessary for AI to think rationally?**

Rational thinking is crucial in Artificial Intelligence (AI) for several reasons:

1. **Decision Making:** Rationality allows AI agents to make decisions similar to humans<sup>1</sup>. It helps AI systems to perform expected behaviors to maximize their performance measure<sup>2</sup>.
2. **Predicted Behaviors and Outcomes:** Rationality is concerned with the predicted behaviors and outcomes based on the agent's perceptions<sup>2</sup>. It enables AI to respond appropriately to different situations<sup>2</sup>.
3. **Collecting Valuable Knowledge:** An important aspect of rationality is taking activities intending to collect valuable knowledge<sup>2</sup>. This helps AI to learn and improve over time<sup>2</sup>.
4. **Autonomous Action:** Rationality is a key principle in AI development because it allows us to create agents that can act independently<sup>3</sup>. This autonomy is crucial for AI to meet its design objectives<sup>3</sup>.
5. **Human-like Thinking:** When we design AI systems that are rational, we are essentially teaching them how to think like humans<sup>1</sup>. This is particularly important in applications where AI interacts directly with humans<sup>1</sup>.

In summary, rationality in AI is about making reasonable, sensible decisions that maximize the success of the AI's objectives, based on its perceptions and knowledge<sup>213</sup>.

## **Can there be more than one agent function? Why?**

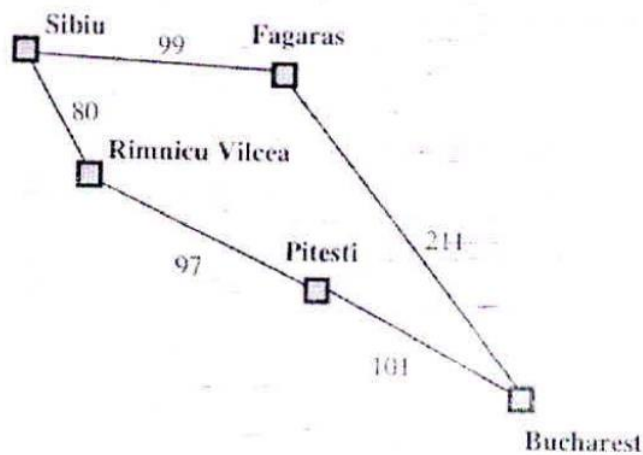
Yes, there can be more than one agent program that implements a given agent function<sup>12</sup>. An agent function defines the mapping from percept sequences to

actions, and there can be multiple ways to achieve the same mapping<sup>2</sup>. Different agent programs can use different algorithms or strategies to achieve the same goal<sup>2</sup>.

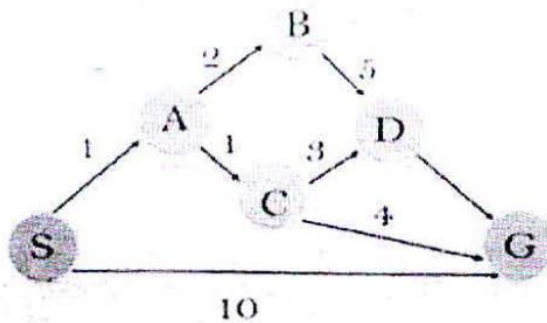
For example, the agent function of navigation may be implemented by agent programs that approach this problem through stored tables vs. a map plus a general-purpose search<sup>1</sup>. These different implementations can still fulfill the same agent function, i.e., navigation, but they do so through different methods<sup>1</sup>.

So, multiple agent programs can exist for a single agent function, each offering a different approach to solving the problem at hand<sup>12</sup>.

Find the path from Sibiu to Bucharest using uniform cost search algorithm.



Traverse the given graph using A\* algorithm. The heuristic values for all states is given in the table. S is the start state and G is goal state. Find cost at each node? Find the optimal path? How many iterations are needed to reach the goal state?



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

Solve the following 8 Puzzle problem using A\* algorithm. The heuristic used is no of misplaced tiles. Give State Space at each level and path from Start to goal state.

Start state  $g=0$ ,  $h=3$ ,  $f=g+h=3$

1	2	3
	4	6
7	5	8

Goal state  $h=0$

1	2	3
4	5	6
7	8	

**Write PEAS for vacuum cleaning ROBOT.**

Sure, let's define the PEAS (Performance measure, Environment, Actuators, Sensors) for a vacuum cleaning robot:

1. **Performance Measure:** The robot's performance can be measured by how clean the floor is after it has finished its task. This could include factors like the amount of dust and debris collected, the coverage of the floor area, the time taken to clean, and the efficiency of battery usage.
2. **Environment:** The environment is the area that the robot is cleaning. This could be a room or a set of rooms within a house or an office. The environment includes all the furniture, walls, stairs, and also the dirt and debris that needs to be cleaned up.
3. **Actuators:** The robot's actuators could include the wheels or tracks that allow it to move around, the vacuum and brushes that collect the dust and debris, and possibly a bin or bag where the dust and debris are stored.
4. **Sensors:** The robot's sensors could include cameras or infrared sensors that help it navigate around the room and avoid obstacles, sensors that detect the level of dirt or dust on the floor, and sensors that monitor the level of dust in the collection bin and the battery level.

In terms of properties of the task environment, the vacuum cleaning task is partially observable (the robot may not see the whole room at once), deterministic (the outcome is directly influenced by the robot's actions), sequential (the robot's actions over time will determine the cleanliness of the room), dynamic (the environment can change as the robot is cleaning or if there are people or pets moving around), discrete (the robot can break down its task into distinct steps like move forward, turn, start/stop vacuuming), and single-agent (assuming the robot is working alone). However, if there are people or pets in the room that can interfere with the robot, it could be considered a multi-agent environment.

Find the overall impulse response of the system if following given systems are connected in cascade.

$$h_1(n) = \left(\frac{1}{4}\right)^n u(n-1), \quad h_2(n) = \left[1 + \left(\frac{1}{2}\right)^n\right] u(n)$$

Comment on stability of the overall system.



Given

$$H(z) = \frac{10z}{z - 0.5}$$

(1) Find the response of the system to the input

$$x[n] = 10 - 5 \sin(0.2\pi n) + 20 \cos(0.4\pi n + 0.5\pi).$$

(2) Identify the filter based on passband of the filter

A Linear – time invariant system is characterized by the system function:

$$H(z) = \frac{3 - 4z^{-1}}{1 - 3.5z^{-1} + 1.5z^{-2}}$$

1. Specify ROC of  $H(z)$  and determine  $h(n)$  for following conditions:

- (i) The system is stable.
- (ii) The system is causal.
- (iii) The system is anti-causal

2. Find the difference equation of above system.

3. Show Parallel realization diagram of system using minimum number of unit delay blocks.

Frequency response of FIR filter is given below:

$$H(e^{j\omega}) = e^{-3j\omega} (2 + 1.8 \cos 3\omega + 1.2 \cos 2\omega + 0.5 \cos \omega)$$

Find: (i) Impulse Response of the filter.

(ii) Response of the filter to the input  $x[n] = (0.5)^n u[n]$ .

**OR**

Contradict Or justify the following statement.

- 1) Impulse Invariant method is not suitable for LPF and HPF.

- 2) Frequency warping is needed to perform in BLT technique but not in Impulse Invariance Technique (IIT).

Impulse response of desired Linear Phase High Pass FIR filter is given below-

$$h_d[n] = \left[ \frac{\sin[(n-2)\pi]}{[(n-2)\pi]} - \frac{7}{10} \frac{\sin((n-2)0.7\pi)}{(n-2)0.7\pi} \right]$$

(1) Truncate the values of  $h_d[n]$  using following window function and obtain realizable finite length  $h[n]$ .

$$w[n] = \left[ 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \right]$$

(2) Plot Magnitude & Phase spectrum of the designed filter.

A Digital Butterworth is required to meet the following specifications:

$$0.85 \leq |H(e^{j\omega})| \leq 1.0 \quad \text{for} \quad 0 \leq \omega \leq 0.22\pi$$

$$|H(e^{j\omega})| \leq 0.25 \quad \text{for} \quad 0.72\pi \leq \omega \leq \pi$$

$$F_s = 1 \text{ KHz.}$$

1. Design a filter using BLT
2. Draw filter realization diagram
3. What is the value of magnitude response of the designed filter at  $\omega = 0$  and  $\omega = \pi$



## Differentiate between Informed and uninformed search technique?

Sure, let's differentiate between Informed and Uninformed search techniques in AI:

### Informed Search:

- [Informed search algorithms have information on the goal state which helps in more efficient searching<sup>1</sup>.](#)
- [This information is obtained by a function that estimates how close a state is to the goal state<sup>1</sup>.](#)
- [Examples of informed search algorithms include A\\* search, Best-First search, and Greedy search<sup>1</sup>.](#)
- [Key features of informed search algorithms in AI include the use of heuristics, efficiency, goal-directedness, cost-based estimates, prioritization, and potentially guaranteeing an optimal solution<sup>1</sup>.](#)

### Uninformed Search:

- [Uninformed search algorithms have no additional information on the goal node other than the one provided in the problem definition<sup>1</sup>.](#)
- [The plans to reach the goal state from the start state differ only by the order and length of actions<sup>1</sup>.](#)
- [Examples of uninformed search algorithms include Breadth-First search \(BFS\), Depth-First search \(DFS\), and Depth-Limited search<sup>1</sup>.](#)
- [Uninformed search algorithms are often used as a starting point for more complex, informed search algorithms or as a way to explore the search space in simple problems<sup>1</sup>.](#)

In summary, the choice between informed and uninformed search depends on the problem and the available domain knowledge. [Informed search algorithms are more efficient when you have relevant information, while uninformed searches are more suitable for situations where you lack such information or need to ensure completeness<sup>2</sup>.](#)