CUET PG 2026 Data Science/AI Question Paper with Solutions

Time Allowed :1 Hour 30 Minutes	Maximum Marks :300	Total Questions :75

General Instructions

Read the following instructions very carefully and strictly follow them:

- The Question Paper consists of 75 Multiple Choice Questions (MCQs).
- Total marks for the exam is 300.
- The total duration is 90 minutes; a, timer on the screen will display the remaining time, and the exam will automatically submit when the time reaches zero.
- For every correct answer, 4 marks (+4) will be awarded to the candidate, and a 1 mark (-1) will be deducted for every incorrect answer.
- Question papers are available in both English and Hindi.

1. What is the time complexity of building a heap from an unsorted array of size n?
(A) $O(n \log n)$
(B) $O(n)$
(C) $O(\log n)$
(D) $O(n^2)$

Correct Answer: (B) $O(n)$

Solution:
Step 1: Understanding the Question:
The question asks for the time complexity of the standard algorithm to convert an unsorted array of n elements into a binary heap.
This is typically done using the bottom-up heap construction method, often called Floyd's algorithm or the 'build-heap' algorithm.

Step 2: Key Formula or Approach:
The approach is to start from the last non-leaf node and call the 'heapify' function on each node up to the root.
The cost of 'heapify' on a node is proportional to its height, h.
The total time complexity is the sum of the costs of 'heapify' for all non-leaf nodes.
The number of nodes at height h in a complete binary tree of n nodes is approximately $\frac{n}{2^{h+1}}$.
The total cost $T(n)$ is given by the summation:

$$T(n) = \sum_{h=0}^{\log_2 n} \left(\frac{n}{2^{h+1}} \right) \cdot O(h)$$

Step 3: Detailed Explanation:
Let's analyze the summation for the total work done.

$$T(n) \approx \sum_{h=0}^{\log n} \frac{n}{2^{h+1}} \cdot c \cdot h$$

Where c is a constant factor from the $O(h)$ complexity of heapify.

$$T(n) = \frac{cn}{2} \sum_{h=0}^{\log n} \frac{h}{2^h}$$

The summation part $\sum_{h=0}^{\infty} \frac{h}{2^h}$ is a known series that converges to a constant value (specifically, 2).
Since the sum is bounded by a constant, the total complexity is proportional to n.

$$T(n) = O(n)$$

This linear time complexity arises because most nodes are near the bottom of the tree (leaves), and the 'heapify' operation on them is very cheap (height is small).

Step 4: Final Answer:
The time complexity of building a heap from an unsorted array of size n using the standard bottom-up approach is $O(n)$.

Quick Tip

A common mistake is to think building a heap takes $O(n \log n)$ by calling insert n times.
While inserting one element is $O(\log n)$, the optimized 'build-heap' algorithm is much faster at $O(n)$ because it performs less work on the numerous nodes near the bottom of the tree.

---

2. In Operating Systems, which condition ensures that a process cannot be pre-empted from its resource until it completes its task?
(A) Mutual Exclusion
(B) Hold and Wait
(C) No Preemption
(D) Circular Wait

Correct Answer: (C) No Preemption

Solution:
Step 1: Understanding the Question:
The question asks to identify the specific condition for deadlock that describes a scenario where a resource cannot be forcibly taken away from a process that holds it.

Step 2: Detailed Explanation:
For a deadlock to occur in an operating system, four conditions must be met simultaneously. Let's review them:

- **Mutual Exclusion:** At least one resource must be held in a non-shareable mode. Only one process at a time can use the resource.
- **Hold and Wait:** A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.
- **No Preemption:** A resource can only be released voluntarily by the process holding it, after that process has completed its task. It cannot be forcibly taken away.
- **Circular Wait:** A set of processes $\{P_0, P_1, \ldots, P_n\}$ must exist such that P_0 is waiting for a resource held by P_1, P_1 is waiting for a resource held by P_2, ..., and P_n is waiting for a resource held by P_0.

The condition described in the question, "a process cannot be preempted from its resource until it completes its task," directly matches the definition of the No Preemption condition.

Step 3: Final Answer:
Therefore, the condition that ensures a resource cannot be forcibly taken from a process is **No Preemption**.

> Quick Tip
>
> Use the mnemonic **MHNC** to remember the four necessary conditions for deadlock:
> **M**utual Exclusion
> **H**old and Wait
> **N**o Preemption
> **C**ircular Wait

---

3. If a relation is both symmetric and antisymmetric, what must be true about the elements in that relation?
(A) All elements must be comparable
(B) The relation must be transitive
(C) Only pairs of the form (a, a) can exist
(D) The relation must contain all ordered pairs

Correct Answer: (C) Only pairs of the form (a, a) can exist

Solution:
Step 1: Understanding the Question:

The question asks about the nature of a relation R that satisfies both the symmetric and antisymmetric properties simultaneously.

Step 2: Key Formula or Approach:
We need to use the formal definitions of symmetric and antisymmetric relations.

- **Symmetric:** For all a, b, if $(a, b) \in R$, then $(b, a) \in R$.
- **Antisymmetric:** For all a, b, if $(a, b) \in R$ and $(b, a) \in R$, then $a = b$.

Step 3: Detailed Explanation:
Let's assume an arbitrary pair (a, b) is in the relation R.

1. Because the relation is **symmetric**, if $(a, b) \in R$, it must be that $(b, a) \in R$.
2. Now we have both $(a, b) \in R$ and $(b, a) \in R$.
3. Because the relation is also **antisymmetric**, the presence of both (a, b) and (b, a) in R implies that $a = b$.

This means that for any pair (a, b) in the relation, it must be the case that $a = b$. Therefore, the only pairs that can exist in such a relation are of the form (a, a).

Step 4: Final Answer:
If a relation is both symmetric and antisymmetric, any pair (a, b) in the relation must satisfy $a = b$. Thus, only pairs of the form (a, a) can exist.

> Quick Tip
>
> Remember this special case: A relation that is both **symmetric** and **antisymmetric** can only contain pairs where the elements are equal to each other.
> Any pair with distinct elements, like $(1, 2)$, would violate the condition.

---

4. Which activation function is zero-centered and ranges between -1 and 1?
(A) Sigmoid
(B) ReLU
(C) Tanh
(D) Softmax

Correct Answer: (C) Tanh

Solution:
Step 1: Understanding the Question:
The question asks to identify a specific activation function based on two properties: its output range is $(-1, 1)$ and its output is centered around zero.

Step 2: Detailed Explanation:
Let's examine the properties of each activation function listed in the options:

- **Sigmoid:** The sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$, has an output range of $(0, 1)$. It is not zero-centered; its outputs are always positive.

- **ReLU (Rectified Linear Unit):** The ReLU function, $f(x) = \max(0, x)$, has an output range of $[0, \infty)$. It is not zero-centered.

- **Tanh (Hyperbolic Tangent):** The Tanh function, $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, has an output range of $(-1, 1)$. Also, $\tanh(0) = 0$, which means its output is centered around zero. This matches the question's criteria.

- **Softmax:** The Softmax function is typically used in the output layer for multi-class classification. It converts a vector of numbers into a probability distribution, where each value is in the range $(0, 1)$ and the sum of all values is 1. It is not zero-centered in the way Tanh is.

Step 3: Final Answer:
The Tanh function is the only one among the options that is both zero-centered and has an output range between -1 and 1.

> Quick Tip
>
> Think of **Tanh** as a scaled and shifted version of the Sigmoid function.
> While Sigmoid's range is $(0, 1)$, Tanh's range is $(-1, 1)$.
> Being zero-centered often helps neural networks converge faster during training.

---

5. What is the determinant of an identity matrix of any order n?
(A) 0
(B) 1
(C) n
(D) -1

Correct Answer: (B) 1

Solution:
Step 1: Understanding the Question:

The question asks for the determinant of an identity matrix, denoted as I_n, for any size $n \times n$.

Step 2: Key Formula or Approach:
A key property of determinants is that for any upper triangular, lower triangular, or diagonal matrix, the determinant is simply the product of the elements on its main diagonal.

Step 3: Detailed Explanation:
An identity matrix I_n is a special type of diagonal matrix. Its main diagonal consists entirely of 1s, and all other elements are 0s.
For example, the 3×3 identity matrix is:

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

To find its determinant, we apply the rule for diagonal matrices and multiply the elements on the main diagonal.

$$\det(I_n) = \underbrace{1 \times 1 \times 1 \times \cdots \times 1}_{n \text{ times}}$$

The product of any number of 1s is always 1.

$$\det(I_n) = 1$$

Step 4: Final Answer:
The determinant of an identity matrix of any order n is always 1.

> Quick Tip
>
> The determinant of any **diagonal matrix** is the product of its diagonal entries.
> Since an identity matrix has only 1s on its diagonal, the product is always 1, regardless of the matrix size.

---

6. In Linear Regression, what is the primary goal of the Ordinary Least Squares (OLS) method?
(A) Maximize the variance of predictions
(B) Minimize the sum of squared residuals
(C) Maximize the correlation between variables
(D) Minimize the number of features

Correct Answer: (B) Minimize the sum of squared residuals

Solution:
Step 1: Understanding the Question:
The question asks for the main objective of the Ordinary Least Squares (OLS) method, which

is the most common technique for fitting a linear regression model.

Step 2: Key Formula or Approach:
In linear regression, we try to find a line (or hyperplane) that best fits the data. The model is of the form $\hat{y} = \beta_0 + \beta_1 x$.
The difference between the actual value y_i and the predicted value $\hat{y}_i$ is called the residual, $e_i = y_i - \hat{y}_i$.
The OLS method aims to find the coefficients (β_0, β_1) that minimize the Sum of Squared Residuals (SSR), also known as the Residual Sum of Squares (RSS).

$$\text{SSR} = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Step 3: Detailed Explanation:
The goal of OLS is to find the line that is "closest" to all the data points simultaneously.
The "distance" from each point to the line is measured by the residual. Some residuals will be positive (point is above the line) and some negative (point is below the line).
To prevent positive and negative errors from canceling each other out, we square them. Squaring also has the desirable property of penalizing larger errors more heavily.
By minimizing the sum of these squared residuals, OLS finds the unique best-fitting line for the given data.

Step 4: Final Answer:
The primary goal of the OLS method is to minimize the sum of squared residuals.

Quick Tip

Remember the name: **Ordinary Least Squares**.
It literally means finding the line that makes the sum of the **squares** of the errors the **least** possible.
This directly points to minimizing the sum of squared residuals.

---

7. Which data structure is used by the system to implement recursion?
(A) Queue
(B) Stack
(C) Linked List
(D) Heap

Correct Answer: (B) Stack

Solution:
Step 1: Understanding the Question:
This question asks about the underlying data structure that computer systems use to manage the execution of recursive function calls.

Step 2: Detailed Explanation:
Recursion is a process where a function calls itself. To manage these calls, the system needs to keep track of the state of each active function call (its parameters, local variables, and the return address).
This is managed using a region of memory called the **call stack**.

- When a function is called (including a recursive call), a new block of memory, called a **stack frame** or **activation record**, is created for it.
- This stack frame is **pushed** onto the top of the call stack.
- When the function finishes execution and returns, its stack frame is **popped** off the stack, and control returns to the address stored in the frame below it.

This mechanism works on a **Last-In, First-Out (LIFO)** principle, because the last function to be called is the first one to return. A Stack is a data structure that operates on the LIFO principle.

Step 3: Final Answer:
The data structure that follows the LIFO principle and is used by the system to implement recursion is the **Stack**.

> Quick Tip
>
> Think of recursion like a stack of plates. You add a new plate (*function call*) to the top. To get to a plate at the bottom, you must first remove all the plates on top of it (*functions must return in the reverse order they were called*). This is a perfect analogy for a **Stack**.

---

8. A high variance in a machine learning model is a primary indicator of which problem?
(A) Underfitting
(B) Overfitting
(C) High bias
(D) Data normalization

Correct Answer: (B) Overfitting

Solution:
Step 1: Understanding the Question:
The question asks to identify the machine learning problem associated with a model that exhibits high variance.

Step 2: Detailed Explanation:
In machine learning, the performance of a model is often analyzed through the lens of the bias-variance tradeoff.

- **Bias** is the error from erroneous assumptions in the learning algorithm. High bias can cause a model to miss relevant relations between features and target outputs. This is known as **underfitting**. The model is too simple.

- **Variance** is the error from sensitivity to small fluctuations in the training set. High variance means the model pays too much attention to the training data, including its noise and random fluctuations.

A model with high variance performs very well on the data it was trained on but fails to generalize to new, unseen data. This phenomenon, where the model essentially "memorizes" the training data instead of learning the underlying pattern, is called **overfitting**. The model is too complex for the data.

Step 3: Final Answer:
A high variance in a machine learning model is a primary indicator of **overfitting**.

Quick Tip

A simple way to remember the relationship is:
High Bias $\rightarrow$ **Underfitting** (Model is too simple and "biased" by its assumptions).
High Variance $\rightarrow$ **Overfitting** (Model "varies" too much with the training data and is too complex).

---

9. In a Binary Search Tree, which traversal method produces the elements in non-decreasing order?
(A) Preorder Traversal
(B) Inorder Traversal
(C) Postorder Traversal
(D) Level Order Traversal

Correct Answer: (B) Inorder Traversal

Solution:
Step 1: Understanding the Question:
The question asks which tree traversal algorithm, when applied to a Binary Search Tree (BST), will visit the nodes in sorted (non-decreasing) order.

Step 2: Key Formula or Approach:
The defining property of a Binary Search Tree is that for any given node:

- All values in its left subtree are less than or equal to the node's value.
- All values in its right subtree are greater than or equal to the node's value.

Let's review the common traversal methods:

- **Preorder:** Root → Left → Right
- **Inorder:** Left → Root → Right
- **Postorder:** Left → Right → Root

Step 3: Detailed Explanation:
By combining the BST property with the Inorder traversal algorithm, we can see why it produces a sorted list.
The Inorder traversal algorithm is defined as:

1. Recursively traverse the left subtree.
2. Visit the root node.
3. Recursively traverse the right subtree.

Following this on a BST, the algorithm first visits all the smaller elements (in the left subtree), then the current element (the root), and finally all the larger elements (in the right subtree). This process, when applied recursively down the tree, naturally results in visiting all nodes in ascending order.

Step 4: Final Answer:
The traversal method that produces elements in non-decreasing order in a BST is the **Inorder Traversal**.

> Quick Tip
>
> A simple mnemonic for this is: **In**order traversal on a BST gives the elements **in order** (sorted).
> This is a fundamental and frequently tested property of Binary Search Trees.

---

10. What is the probability of getting a sum of 9 when two fair dice are rolled simultaneously?
(A) $\frac{1}{12}$
(B) $\frac{1}{9}$

(C) $\frac{1}{8}$
(D) $\frac{1}{6}$

Correct Answer: (B) $\frac{1}{9}$

Solution:
Step 1: Understanding the Question:
The question asks for the probability of a specific event: the sum of the outcomes of rolling two fair six-sided dice is exactly 9.

Step 2: Key Formula or Approach:
The formula for probability is:

$$P(\text{Event}) = \frac{\text{Number of Favorable Outcomes}}{\text{Total Number of Possible Outcomes}}$$

Step 3: Detailed Explanation:
1. Calculate the Total Number of Possible Outcomes:
Each die has 6 possible outcomes $\{1, 2, 3, 4, 5, 6\}$.
When two dice are rolled, the total number of combinations is $6 \times 6 = 36$.

2. Identify the Favorable Outcomes:
We need to find all pairs of outcomes (d_1, d_2) such that $d_1 + d_2 = 9$.
Listing them out:

- If the first die is 3, the second must be 6: $(3, 6)$
- If the first die is 4, the second must be 5: $(4, 5)$
- If the first die is 5, the second must be 4: $(5, 4)$
- If the first die is 6, the second must be 3: $(6, 3)$

There are a total of 4 favorable outcomes.

3. Calculate the Probability:
Using the probability formula:

$$P(\text{sum} = 9) = \frac{4}{36}$$

Simplifying the fraction:

$$\frac{4}{36} = \frac{1}{9}$$

Step 4: Final Answer:
The probability of getting a sum of 9 is $\frac{1}{9}$.

Quick Tip

For two-dice problems, always remember the total number of outcomes is 36.
It can be helpful to visualize a 6×6 grid of all possible sums. The sums equal to 9 form a diagonal line in this grid.

---

11. Which layer of the OSI model is responsible for IP addressing and routing?
(A) Data Link Layer
(B) Network Layer
(C) Transport Layer
(D) Session Layer

Correct Answer: (B) Network Layer

Solution:
Step 1: Understanding the Question:
The question asks to identify the layer in the seven-layer OSI (Open Systems Interconnection) model that handles logical addressing (like IP addresses) and the process of routing data packets between different networks.

Step 2: Detailed Explanation:
Let's review the primary responsibilities of the relevant OSI layers:

- **Data Link Layer (Layer 2):** Responsible for node-to-node data transfer on the same physical link. It deals with physical addressing (MAC addresses) and is concerned with frames, not packets across networks.

- **Network Layer (Layer 3):** This layer is responsible for packet forwarding, which includes routing through different routers. It manages logical addressing (e.g., IPv4, IPv6 addresses) to identify hosts on the internetwork and determines the best path for data to travel from source to destination. Routers operate at this layer.

- **Transport Layer (Layer 4):** Responsible for end-to-end communication between hosts. It provides services like segmentation, flow control, and error control. Protocols like TCP and UDP operate here. It uses port numbers but relies on the Network Layer for IP addressing and routing.

- **Session Layer (Layer 5):** Manages sessions between applications, handling session establishment, maintenance, and termination.

The tasks of IP addressing and routing are the core functions of the Network Layer.

Step 3: Final Answer:
The layer of the OSI model responsible for IP addressing and routing is the **Network Layer**.

> Quick Tip
>
> Remember this key association: **Network Layer = Routers and IP Addresses**. Think of the Network Layer as the "postal service" of the internet. It reads the logical address (IP address) on a package (packet) and decides the route it should take to reach its destination.

---

12. How many edges are there in a complete graph with n vertices?
(A) $n(n-1)$
(B) $\frac{n(n-1)}{2}$
(C) n^2
(D) $\frac{n(n+1)}{2}$

Correct Answer: (B) $\frac{n(n-1)}{2}$

Solution:
Step 1: Understanding the Question:
The question asks for the formula to calculate the number of edges in a complete graph, denoted K_n, which has n vertices. A complete graph is one where every vertex is connected to every other vertex by a unique edge.

Step 2: Key Formula or Approach:
We can solve this from two perspectives: using combinations or using the handshaking lemma.
Approach 1 (Combinations):
An edge is formed by connecting a pair of vertices. The problem is equivalent to finding how many unique pairs of vertices we can choose from a set of n vertices. This is a combination problem.
The number of ways to choose 2 vertices from n is given by "n choose 2":

$$\binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$$

Step 3: Detailed Explanation:
Approach 2 (Handshaking Lemma):
1. In a complete graph with n vertices, each vertex is connected to all other $n-1$ vertices. Therefore, the degree of each vertex is $n-1$.
2. The sum of the degrees of all vertices in the graph is the number of vertices multiplied by the degree of each vertex: Sum of degrees $= n \times (n-1)$.
3. The Handshaking Lemma states that the sum of degrees is equal to twice the number of edges ($2E$).

$$2E = n(n-1)$$

4. To find the number of edges E, we divide by 2:

$$E = \frac{n(n-1)}{2}$$

Step 4: Final Answer:
The number of edges in a complete graph with n vertices is $\frac{n(n-1)}{2}$.

> Quick Tip
>
> Think of it as a "handshake problem": If n people are in a room and everyone shakes hands with everyone else, how many handshakes occur?
> This is the same as choosing 2 people out of n for a handshake, which is $\binom{n}{2} = \frac{n(n-1)}{2}$.
> This is a very common formula in graph theory and combinatorics.

---

13. In SQL, which clause is used to filter the results of an aggregate function?
(A) WHERE
(B) GROUP BY
(C) HAVING
(D) ORDER BY

Correct Answer: (C) HAVING

Solution:
Step 1: Understanding the Question:
The question asks for the specific SQL clause used to apply a filter based on the output of an aggregate function like `COUNT()`, `SUM()`, `AVG()`, etc.

Step 2: Detailed Explanation:
In SQL, filtering can happen at two different stages of a query's execution:

- **Filtering before grouping (`WHERE` clause):**
  The `WHERE` clause is used to filter individual rows from a table *before* any grouping or aggregation is performed. You cannot use an aggregate function in a `WHERE` clause because the aggregation has not happened yet.
  Example: `SELECT * FROM sales WHERE amount > 100;`

- **Filtering after grouping (`HAVING` clause):**
  The `HAVING` clause is specifically designed to filter groups *after* the `GROUP BY` clause has created them and the aggregate functions have been computed. This is where you place conditions on aggregate results.
  Example: `SELECT department, SUM(salary) FROM employees GROUP BY department HAVING SUM(salary) > 500000;`

The `GROUP BY` clause is used to create the groups, and the `ORDER BY` clause is used to sort the final result set. Neither is used for filtering aggregates.

Step 3: Final Answer:
The SQL clause used to filter the results of an aggregate function is **HAVING**.

> Quick Tip
>
> Remember the order of execution in a SQL query: `FROM` → `WHERE` → `GROUP BY` → **HAVING** → `SELECT` → `ORDER BY`.
> **WHERE** filters rows.
> **HAVING** filters groups.
> You can't use `HAVING` without `GROUP BY` (with few exceptions in some SQL dialects).

---

14. What is the value of $(AB)^T$ in matrix algebra?
(A) $A^T B^T$
(B) $B^T A^T$
(C) AB
(D) $A^T B$

Correct Answer: (B) $B^T A^T$

Solution:
Step 1: Understanding the Question:
The question asks for the correct formula for the transpose of the product of two matrices, A and B.

Step 2: Key Formula or Approach:
This question tests a fundamental property of matrix transposes known as the "reversal rule" for products. The rule states that the transpose of a product of matrices is the product of their transposes in the reverse order.

$$(AB)^T = B^T A^T$$

Step 3: Detailed Explanation:
Let's consider why the order must be reversed. This can be understood by looking at the element in the i-th row and j-th column of the matrices.
The element at position (i, j) of the transpose of a matrix M, denoted $(M^T)_{ij}$, is the same as the element at position (j, i) of the original matrix M, i.e., $(M^T)_{ij} = M_{ji}$.
So, the element at position (i, j) of $(AB)^T$ is the element at position (j, i) of AB.
The (j, i)-th element of AB is the dot product of the j-th row of A and the i-th column of B.
Now, consider $B^T A^T$. The (i, j)-th element of this product is the dot product of the i-th row of B^T and the j-th column of A^T.
The i-th row of B^T is the i-th column of B, and the j-th column of A^T is the j-th row of A.
Thus, the (i, j)-th element of $B^T A^T$ is the dot product of the j-th row of A and the i-th column

of B, which is exactly the same as the (i, j)-th element of $(AB)^T$.

Step 4: Final Answer:
The value of $(AB)^T$ in matrix algebra is $B^T A^T$.

> Quick Tip
>
> This is often called the "socks and shoes" property. To get dressed, you put on socks first, then shoes. To get undressed, you must reverse the order: take off shoes first, then socks.
> Similarly, for the transpose of a product: $(AB)^T = B^T A^T$. The order is reversed.

---

15. Which ensemble technique reduces variance by training multiple trees on different subsets of data?
(A) Boosting
(B) Bagging
(C) Stacking
(D) Gradient Descent

Correct Answer: (B) Bagging

Solution:
Step 1: Understanding the Question:
The question describes an ensemble machine learning technique and asks for its name. The key features described are: 1) it reduces variance, 2) it trains multiple models (trees), and 3) it uses different subsets of data for each model.

Step 2: Detailed Explanation:
Let's analyze the options:

- **Boosting:** This is an ensemble technique where models are trained sequentially. Each subsequent model focuses on correcting the errors made by its predecessor. The primary goal of boosting is to reduce **bias**, not variance.
- **Bagging (Bootstrap Aggregating):** This technique involves creating multiple random subsets of the original training data with replacement (this is called bootstrapping). A separate model (often a decision tree) is trained independently on each subset. The final prediction is made by aggregating the predictions of all models (e.g., by voting or averaging). This process of averaging over multiple models trained on different data samples is highly effective at reducing the model's **variance** and preventing overfitting. The Random Forest algorithm is a well-known implementation of Bagging.

- **Stacking:** This is an ensemble technique that combines heterogeneous models by training a "meta-model" to learn how to best combine the predictions from several base models.

- **Gradient Descent:** This is an optimization algorithm used to train a single model by minimizing a loss function. It is not an ensemble technique.

The description in the question perfectly matches the definition of Bagging.

Step 3: Final Answer:
The ensemble technique that reduces variance by training multiple trees on different subsets of data is **Bagging**.

> Quick Tip
>
> Remember the main goal of the two most popular ensemble methods:
> **Bagging** $\rightarrow$ Reduces **Variance**. Models are trained in parallel.
> **Boosting** $\rightarrow$ Reduces **Bias**. Models are trained sequentially.
> Random Forest is a prime example of Bagging.
