**Question 1.1: Write the Answer to those Questions.**

**Q) what is the difference between static and dynamic variables in python?**

**Static Variables:**

- Variables that are shared across all instances of a class.
- Defined at the class level and have the same value for all objects of the class.

**Example:**

```
class MyClass:
  static_var = 10  # Static variable
obj1 = MyClass()
obj2 = MyClass()
print(obj1.static_var)  # Output: 10
print(obj2.static_var)  # Output: 10
```

**Dynamic Variables:**

- Variables that are instance-specific.
- Defined inside a method (usually __init__) and can have different values for different instances.

**Example:**

```
class MyClass:
  def __init__(self, value):
    self.dynamic_var = value  # Dynamic variable
obj1 = MyClass(10)
obj2 = MyClass(20)
print(obj1.dynamic_var)  # Output: 10
print(obj2.dynamic_var)  # Output: 20
```

---

**Q) Explain the purpose of 'pop','popitem','clear()' in a dictionary with suitable examples.**

**pop(key):**

- Removes the item with the specified key and returns its value.
- Raises a KeyError if the key doesn't exist unless a default value is provided

**Example:**

```
 my_dict = {'a': 1, 'b': 2, 'c': 3}
value = my_dict.pop('b')  # Removes key 'b' and returns its value
print(value)  # Output: 2
print(my_dict)  # Output: {'a': 1, 'c': 3}
```

**popitem()**:

- Removes and returns the last key-value pair as a tuple.
- Raises a KeyError if the dictionary is empty

**Example:**

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
item = my_dict.popitem()  # Removes the last item ('c', 3)
print(item)  # Output: ('c', 3)
print(my_dict)  # Output: {'a': 1, 'b': 2}
```

**clear()**:

- Removes all items from the dictionary, leaving it empty.

**Example:**

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
my_dict.clear()  # Clears all items from the dictionary
print(my_dict)  # Output: {}
```

---

**Q) what do you mean by FrozenSet? Explain it with suitable example.**

A frozenset in Python is like a set, but it is **i**mmutable, meaning once created, its elements cannot be changed, added, or removed. It's useful when you need a collection of unique elements that should remain constant.

**Example:**

```
my_set = frozenset([1, 2, 3, 4])
print(my_set)  # Output: frozenset({1, 2, 3, 4})
# Trying to modify the frozenset will raise an error
```

---

**Q) Differentiate between mutal and immutable data types in python and give examples of mutable and immutable data types.**

**Mutable Data Types:**

- Can be changed or modified after creation without creating a new object.

- Examples: list, dictionary, set.

**Example:**
```
my_list = [1, 2, 3]
my_list[0] = 10  # Modifying the list
print(my_list)  # Output: [10, 2, 3]
```

---

**Immutable Data Types:**
- Cannot be changed or modified after creation. If you attempt to change it, a new object is created instead.
- Examples: int, float, string, tuple, frozenset

**Example:**
```
my_string = "hello"
# my_string[0] = "H"  # This will raise an error as strings are immutable
new_string = my_string.replace("h", "H")  # Creates a new string
print(new_string)  # Output: "Hello"
```

---

**Q) What is __init__?Explain with an example.**

__init__ is a special method in Python classes that is used for initializing objects. It is called automatically when a new instance of a class is created.

**Example:**
```
class MyClass:
    def __init__(self, name):
        self.name = name  # Initialize the object's name attribute
obj = MyClass("John")
print(obj.name)  # Output: John
```

---

**Q ) What is docstring in Python? Explain with an example.**

A docstring is a string literal used to document functions, methods, classes, or modules. It helps describe what the code does.

**Example:**
```
def add(a, b):
    """
    This function adds two numbers.
    :param a: First number
    :param b: Second number
    :return: Sum of a and b
    """
    return a + b
print(add.__doc__)  # Output: This function adds two numbers.
```

---

**Q) What are unit tests in Python?**

Unit tests are small tests that check individual units of code (like functions or methods) for correctness. Python's unittest framework is commonly used to create and run these tests.

**Example:**
```
import unittest
def add(a, b):
    return a + b
class TestAddFunction(unittest.TestCase):
    def test_add(self):
        self.assertEqual(add(2, 3), 5)
if __name__ == "__main__":
    unittest.main()
```

---

**Q) What is break, continue and pass in Python?**

**break**: Exits the current loop completely.
**continue**: Skips the current iteration and moves to the next iteration of the loop.
**pass**: Does nothing and is used as a placeholder in code.

**Example:**
```
for i in range(5):
    if i == 3:
        break  # Exits the loop when i is 3
    print(i)

for i in range(5):
    if i == 3:
```

```
    continue  # Skips the iteration when i is
print(i)


for i in range(5):
  if i == 3:
    pass  # Does nothing
  print(i)
```

---

**Q) What is the use of self in Python? •**

**'self '** refers to the instance of the class. It allows access to the instance variables and methods within the class. It must be the first parameter of instance methods.
self refers to the instance of the class. It allows access to the instance variables and methods within the class. It must be the first parameter of instance methods.

**Example:**

```
class MyClass:
  def __init__(self, name):
    self.name = name  # Using 'self' to refer to the instance's 'name' attribute

  def greet(self):
    return f"Hello, {self.name}!"

obj = MyClass("Alice")
print(obj.greet())  # Output: Hello, Alice!
```

---

**Q) What are global, protected and private attributes in Python?**

**Global Attributes**: These are variables defined at the top level of a module, outside of any class or function. They can be accessed globally throughout the module.
**Protected Attributes**: These are attributes that are meant to be accessed within the class and its subclasses. In Python, they are indicated by a single underscore _. However, they can still be accessed from outside (this is a convention, not enforcement).
**Private Attributes**: These attributes are meant to be used only inside the class. They are indicated by a double underscore __. Python internally changes the name of the attribute to prevent accidental access (name mangling).

**Example:**

```
class MyClass:
  global_var = 10  # Global attribute

  def __init__(self):
    self._protected_var = 20  # Protected attribute
    self.__private_var = 30   # Private attribute

obj = MyClass()
print(obj._protected_var)  # Accessible, but discouraged (Output: 20)
# print(obj.__private_var)  # Will raise an AttributeError
print(obj._MyClass__private_var)  # Accessing private var via name mangling (Output: 30)
```

---

**Q) What are modules and packages in Python?**

**Modules**: A module is simply a Python file (with a .py extension) that contains Python code, such as functions, classes, or variables. Modules allow code reuse.
**Packages**: A package is a collection of modules organized in directories. It often contains an __init__.py file. Packages allow a hierarchical structure for organizing modules.

**Example:**

```
# Module (my_module.py)
def greet(name):
  return f"Hello, {name}!"    # Package structure:
# my_package/
#    __init__.py
#    my_module.py
```

---

**Q) What are lists and tuples? What is the key difference between the two?**

- Lists: Mutable sequences in Python, which means their elements can be changed (added, removed, modified).
- Tuples: Immutable sequences, meaning once a tuple is created, its elements cannot be changed.

Key Difference: Lists are mutable, and tuples are immutable.

**Example:**
```
my_list = [1, 2, 3]
my_tuple = (1, 2, 3)

my_list[0] = 10  # Modifying a list (allowed)
# my_tuple[0] = 10  # Trying to modify a tuple (raises an error)
```

---

**Q) What is an Interpreted language & dynamically typed language?Write 5 differences between them.**

**Interpreted Language:** In an interpreted language, the code is executed line by line by an interpreter at runtime.
**Dynamically Typed Language**: In a dynamically typed language, the type of a variable is determined at runtime, not in advance.

| Interpreted Language | Dynamically Typed Language |
|---|---|
| Executes code line by line | Variable types are determined at runtime |
| Slower execution speed compared to compiled languages | No need to declare variable types |
| No separate compilation step required | Can lead to runtime errors due to type mismatches |
| Easier for debugging | Variables can change types dynamically |
| Examples: Python, Ruby | Examples: Python, JavaScript |

---

**Q) What are Dict and List comprehensions?**

**List Comprehension:** A concise way to create lists based on existing lists.
**Dict Comprehension:** A concise way to create dictionaries based on existing dictionaries or other iterables.

**Example:**
```
# List Comprehension
squares = [x**2 for x in range(5)]
print(squares)  # Output: [0, 1, 4, 9, 16]

# Dict Comprehension
square_dict = {x: x**2 for x in range(5)}
print(square_dict)  # Output: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

---

**Q) What are decorators in Python? Explain it with an example.Write down its use cases.**

**Decorators:** Functions that modify the behavior of other functions or methods. They allow you to "wrap" another function in order to add functionality before or after the wrapped function runs.

**Use Cases**:

- Logging
- Authentication
- Caching
- Timing execution

**Example:**
```
def my_decorator(func):
  def wrapper():
    print("Something before the function")
    func()
    print("Something after the function")
  return wrapper

@my_decorator
def say_hello():
  print("Hello!")

say_hello()
```

---

**Q) How is memory managed in Python?**

Memory management in Python is handled by the Python Memory Manager. It uses a combination of:

- Reference Counting: Python keeps track of how many references exist for each object. When an object's reference count drops to zero, the memory is deallocated.
- Garbage Collection: Python uses a cyclic garbage collector to detect and clean up objects involved in reference cycles (where objects reference each other, preventing their reference counts from reaching zero).

**Example:**
```
a = [1, 2, 3]
```

```
b = a  # Reference count for the list increases to 2
del a  # Reference count decreases to 1  # When the reference count drops to 0, memory is freed
```

---

**Q) What is lambda in Python? Why is it used?**

lambda is used to create small, anonymous functions in Python. It is often used when you need a simple function for a short period of time, especially as an argument to higher-order functions like map(), filter(), or sorted().
**Syntax:** 'lambda arguments: expression'

**Example:**

```
add = lambda x, y: x + y
print(add(2, 3))  # Output: 5
```

---

**Q) Explain split() and join() functions in Python?**

**split():** This method splits a string into a list of substrings based on a delimiter (e.g., space, comma).
**join():** This method joins elements of an iterable (like a list) into a string, using a specified separator.

**Example:**

```
# split()
text = "hello world"
words = text.split()  # Default delimiter is space
print(words)  # Output: ['hello', 'world']

# join()
words = ['hello', 'world']
sentence = " ".join(words)  # Joins with space as a separator
print(sentence)  # Output: hello world
```

---

**Q) What are iterators, iterable & generators in Python?**

**Iterable:** Any object capable of returning its members one at a time. Examples include lists, tuples, and strings.
**Iterator:** An object representing a stream of data; it returns data one element at a time when called with next(). Iterators are created using iterables.
**Generator:** A special type of iterator defined using a function and the yield keyword. Generators produce items lazily, meaning they generate items only when requested.

**Example:**

```
# Iterable
my_list = [1, 2, 3]

# Iterator
my_iter = iter(my_list)
print(next(my_iter))  # Output: 1

# Generator
def my_gen():
    for i in range(3):
        yield i

gen = my_gen()
print(next(gen))  # Output: 0
```

---

**Q) What is the difference between xrange and range in Python?**

**range()** returns a list of numbers.
**xrange()** returns an iterator (more memory efficient).

**Example:**

```
# range(5) -> [0, 1, 2, 3, 4]
# xrange(5) -> xrange(0, 5)
```

---

**Q) Pillars of Oops.**

**Encapsulation:** Bundling data and methods that operate on the data into a single unit (class) and restricting direct access to some of the object's components.
**Inheritance:** A mechanism by which one class inherits the attributes and methods of another class.
**Polymorphism:** The ability to use a single interface for different data types or classes (e.g., method overriding). Abstraction: Hiding the complex implementation details and showing only the essential features.

**Example:**

```
class Animal:
    def sound(self):
        return "Some sound"
```

```
class Dog(Animal):
  def sound(self):
    return "Bark"

dog = Dog()
print(dog.sound())  # Output: Bark
```

---

**Q)How will you check if a class is a child of another class?**

You can use the built-in function issubclass() to check if a class is a child (subclass) of another class.

**Example:**

```
class Animal:
  pass
class Dog(Animal):
  pass
print(issubclass(Dog, Animal))  # Output: True
```

---

**Q) How does inheritance work in python? Explain all types of inheritance with an example.**

Inheritance allows a class to inherit attributes and methods from another class. There are different types of inheritance:

- Single Inheritance: Inherits from one parent class.
- Multiple Inheritance: Inherits from more than one parent class.
- Multilevel Inheritance: A class inherits from a class, which itself inherits from another class.
- Hierarchical Inheritance: Multiple classes inherit from one parent class.
- Hybrid Inheritance: A combination of more than one type of inheritance.

**Example:**

```
# Single Inheritance
class Parent:
  def show(self):
    return "Parent"

class Child(Parent):
  pass

c = Child()
print(c.show())  # Output: Parent

# Multiple Inheritance
class Father:
  def sound(self):
    return "Father sound"

class Mother:
  def sound(self):
    return "Mother sound"

class Child(Father, Mother):
  pass

c = Child()
print(c.sound())  # Output: Father sound (MRO: Method Resolution Order)
```

---

**Q) What is encapsulation? Explain it with an example.**

Encapsulation is the concept of wrapping data and the methods that operate on that data within a single class and controlling access to the data by making attributes private or protected.

**Example:**

```
class Person:
  def __init__(self, name):
    self.__name = name  # Private attribute

  def get_name(self):
    return self.__name  # Accessor method to get the private attribute

p = Person("Alice")
print(p.get_name())  # Output: Alice
# print(p.__name)  # This will raise an AttributeError
```

---

**Q) What is polymorphism? Explain it with an example.**

Polymorphism allows objects of different classes to be treated as objects of a common base class. It enables using the same method name for different types of objects.

**Example:**

```
class Cat:
  def sound(self):
    return "Meow"

class Dog:
  def sound(self):
    return "Bark"

def animal_sound(animal):
  print(animal.sound())

cat = Cat()
dog = Dog()

animal_sound(cat)  # Output: Meow
animal_sound(dog)  # Output: Bark
```

---

**Question 1. 2. Which of the following identifier names are invalid and why? a) Serial_no. b) 1st_Room c) Hundred$d) Total_Marks e) total-Marks f) Total Marks g) True h) _Percentag**

**Invalid Identifiers:**

**b) 1st_Room**
Invalid. Identifiers cannot start with a number. In Python, identifiers must begin with a letter (A-Z, a-z) or an underscore (_).

**e) total-Marks**
Invalid. Identifiers cannot contain hyphens (-). The hyphen is interpreted as a subtraction operator in Python, so it cannot be used in an identifier.

**f) Total Marks**
Invalid. Identifiers cannot contain spaces. Identifiers must be a continuous sequence of characters without spaces.

**g) True**
Invalid. True is a reserved keyword in Python and cannot be used as an identifier.

---

**Question 1. 3. name = ["Mohan", "dash", "karam", "chandra", "gandhi","Bapu"] do the following operations in this list;**

**a) add an element "freedom_fighter" in this list at the Oth index.**

```
Code: name.insert(0, "freedom_fighter")
      Print(name)
```

**b) find the output of the following, and explain how?**

```
name = ["freedomFighter", "Bapuji", "Mohan" "dash", "karam", "chandra", "gandhi"]
length1=len((name[-len (name)+1:-1:2]))
length2=len((name[-len (name)+1:-1]))
print(length1+length2)
```

**Output:** 7

**Explanation:**

length1 = len(name[-len(name)+1:-1:2])
This slices the list from index 1 to -1, with a step of 2.

name[-len(name)+1:-1:2] results in:
['Bapuji', 'karam', 'gandhi']
The length of this sublist is 3, so length1 = 3.
length2 = len(name[-len(name)+1:-1])
This slices the list from index 1 to -1, without skipping any elements.

name[-len(name)+1:-1] results in:
['Bapuji', 'Mohandash', 'karam', 'chandra']
The length of this sublist is 4, so length2 = 4.

**c) Add two more elements, ["Netaji", "Bose"], at the end of the list:**

**code:**

```
name.extend(["Netaji", "Bose"])
print(name)
```

**d) What will be the value of temp?**

```
name = ["Bapuji", "dash", "karam", "chandra", "gandi", "Mohan"]
temp = name[-1]  # "Mohan"
name[-1] = name[0]  # name[-1] = "Bapuji"
name[0] = temp  # name[0] = "Mohan"
print(name)
```

Value of **Temp :** temp = "Mohan"  # Since we store the last element before the swap.

---

**Question 1.4. Find the output of the following.**

```
animal = ['Human','cat','mat', 'cat','rat', 'Human', 'Lion']
print(animal.count('Human'))
print(animal.index('rat'))
print(len(animal))
```

**Final Output:**

```
2  # Output of animal.count('Human')
4  # Output of animal.index('rat')
7  # Output of len(animal)
```

---

**20) What do you mean by Measure of Central Tendency and Measures of Dispersion. How it can be calculated.**

**Measure of Central Tendency** refers to the statistical measure that identifies a single value as representative of an entire distribution. It includes the mean, median, and mode.

- Mean: Average of all values.
- Median: Middle value when data is ordered.
- Mode: Most frequent value.

**Measures of Dispersion** refers to the spread of the data around the central tendency. It includes range, variance, and standard deviation.

- Range: Difference between maximum and minimum values.
- Variance: Average of squared differences from the mean.
- Standard Deviation: Square root of variance.

---

**21) What do you mean by skewness. Explain its types.Use graph to show.**

Skewness refers to the asymmetry in the distribution of data.

- **Positive skew (Right skew):** Long tail on the right side.
- **Negative skew (Left skew):** Long tail on the left side.

---

**22) Explain PROBABILITY MASS FUNCTION (PMF) and PROBABILITY DENSITY FUNCTION (PDF). and what is the difference between them?**

- Probability Mass Function (PMF): Used for discrete random variables; gives the probability of each outcome.
- Probability Density Function (PDF): Used for continuous random variables; describes the relative likelihood of a value.

Difference:

- PMF deals with discrete values, while PDF deals with continuous ranges.
-

---

**23) What is correlation. Explain its type in details.what are the methods of determining correlation.**

- Correlation measures the strength and direction of a relationship between two variables. Types include:
  - Positive Correlation: Both variables move in the same direction.
  - Negative Correlation: Variables move in opposite directions.
  - No Correlation: No relationship.

Methods of Determining Correlation:

- Pearson Correlation Coefficient: Measures linear relationships.
- Spearman's Rank Correlation: Measures monotonic relationships.

---

**25) Discuss the 4 differences between correlation and regression.**

**Direction:** Correlation measures the strength and direction of the relationship between two variables. Regression predicts the value of one variable based on the value of another.
**Dependency:** Correlation treats both variables symmetrically, while in regression, one variable is dependent, and the other is independent.
**Unit of Measure:** Correlation is dimensionless (no units). Regression has units because it predicts actual values.
**Linearity:** Correlation measures linear relationships. Regression assumes a linear relationship between the variables.

---

**26) Find the most likely price at Delhi corresponding to the price of Rs. 70 at Agra from the following data: Coefficient of correlation between the prices of the two places +0.8.**

- The formula to use is: y=mx+by = mx + by=mx+b, where mmm is the slope, and bbb is the intercept.
- Given the correlation coefficient r=0.8r = 0.8r=0.8, the most likely price in Delhi is calculated by setting up the regression equation.

For Rs. 70 at Agra and r=0.8r = 0.8r=0.8, we approximate that Delhi's price would be similar to Agra's due to the strong correlation, i.e., likely around Rs. 70.

---

**27) In a partially destroyed laboratory record of an analysis of correlation data, the following results only are legible: Variance of x = 9, Regression equations are: (i) 8x-10y = -66; (ii) 40x-18y214. What are (a) the mean values of x and y, (b) the coefficient of correlation between x and y, (c) the σ of y.**

**Final Answers:**

- **(a) Mean values: $\bar{x}$=13\bar{x} =** 13$\bar{x}$=13, $\bar{y}$=17\bar{y} = 17$\bar{y}$=17
- **(b) Coefficient of correlation:** r=0.6r = 0.6r=0.6
- **(c) Standard deviation of yyy: σy=4\sigma_y = 4σy=**4
-

---

**28) What is Normal Distribution? What are the four Assumptions of Normal Distribution? Explain in detail.**

**Definition:** A probability distribution that is symmetric about the mean, showing that data near the mean are more frequent in occurrence.
**Assumptions:**

1. The data is continuous.
2. The distribution is symmetric around the mean.
3. The mean, median, and mode are equal.
4. The curve follows the 68-95-99.7 rule (standard deviations from the mean).

---

**29) Write all the characteristics or Properties of the Normal Distribution Curve.**

**Characteristics of Normal Distribution:**

- Symmetrical bell-shaped curve.
- Mean, median, and mode are equal.
- The area under the curve represents probability.
- The curve never touches the x-axis.
- 68% of data is within 1 standard deviation, 95% within 2, and 99.7% within 3.

---

**30) Which of the following options are correct about Normal Distribution Curve. (a) Within a range 0.6745 of a on both sides the middle 50% of the observations occur i,e. mean ±0.67450 covers 50% area 25% on each side. (b) Mean #IS.D. (le.μ = 10) covers 68.268% area, 34.134 % area lies on either side of the mean. (c) Mean ±2S.D. (ie. μ 20) covers 95.45% area, 47.725% area lies on either side of the mean. (d) Mean ±3 S.D. (ie. μ ±30) covers 99.73% area, 49.856% area lies on the either side of the mean. (e) Only 0.27% area is outside the range μ ±30.**

- (a) True: 0.6745 standard deviations cover 50% of observations.
- (b) True: Mean ±1 S.D. covers 68.268% area.
- (c) True: Mean ±2 S.D. covers 95.45% area.
- (d) True: Mean ±3 S.D. covers 99.73% area.
- (e) True: Only 0.27% area is outside μ ±3 S.D.

---

**31) The mean of a distribution is 60 with a standard deviation of 10. Assuming that the distribution is normal, what percentage of items be (i) between 60 and 72, (ii) between 50 and 60, (iii) beyond 72 and (iv) between 70 and 80?**

Given mean = 60, S.D. = 10, and assuming normal distribution:

- (i) Between 60 and 72: 34.13%
- (ii) Between 50 and 60: 34.13%
- (iii) Beyond 72: 15.87%
- (iv) Between 70 and 80: Approx. 13.59%

---

**32) 15000 students sat for an examination. The mean marks was 49 and the distribution of marks had a standard deviation of 6. Assuming that the marks were normally distributed what proportion of students scored (a) more than 55 marks, (b) more than 70 marks**

(a) More than 55 marks: 15.87%
(b) More than 70 marks: Very small, close to 0.13%.

---

**33) If the height of 500 students are normally distributed with mean 65 inch and standard deviation 5 inch. How many students have height: a) greater than 70 inch. b) between 60 and 70 inch.**

(a) Greater than 70 inches: Approx. 16%.
(b) Between 60 and 70 inches: Approx. 68%.

---

**34) What is the statistical hypothesis? Explain the errors in hypothesis testing.b) Explain the Sample. What are Large Samples & Small Samples?**

**Definition:** A hypothesis is a statement that can be tested statistically.
**Errors in Hypothesis Testing:**

- **Type I Error:** Rejecting a true null hypothesis.

- **Type II Error:** Accepting a false null hypothesis.

**Sample:** A sample is a subset of a population. Large samples typically have n>30n > 30n>30; small samples have n≤30n

---

**35) A random sample of size 25 from a population gives the sample standard derivation to be 9.0. Test the hypothesis that the population standard derivation is 10.5. Hint(Use chi-square distribution).**

**Hypotheses:**

- Null Hypothesis ($H_0$): The population standard deviation is 10.5.
  - H0:σ=10.5H_0: \sigma = 10.5H0:σ=10.5
- Alternative Hypothesis ($H_1$): The population standard deviation is not 10.5.
  - H1:σ≠10.5H_1: \sigma \neq 10.5H1:σ=10.5

**Step 1: Formula for Chi-Square Test**

The chi-square statistic for testing a hypothesis about the population variance is given by:

χ2=(n−1)·s2σ02\chi^2 = \frac{(n-1) \cdot s^2}{\sigma_0^2}χ2=σ02(n−1)·s2

Where:

- nnn = sample size
- s2s^2s2 = sample variance
- σ02\sigma_0^2σ02 = hypothesized population variance

**Step 2: Calculate the Chi-Square Test Statistic**

The degrees of freedom for the chi-square distribution is n−1=25−1=24n - 1 = 25 - 1 = 24n−1=25−1=24.

- Sample variance: s2=9.02=81s^2 = 9.0^2 = 81s2=9.02=81
- Hypothesized population variance: σ02=10.52=110.25\sigma_0^2 = 10.5^2 = 110.25σ02=10.52=110.25

Now, calculate the chi-square statistic:

χ2=(25−1)·81110.25=24·81110.25=1944110.25≈17.63\chi^2 = \frac{(25 - 1) \cdot 81}{110.25} = \frac{24 \cdot 81}{110.25} = \frac{1944}{110.25} \approx 17.63χ2=110.25(25−1)·81=110.2524·81=110.251944≈17.63

**Step 3: Determine the Critical Values**

For a two-tailed test with a significance level of α=0.05\alpha = 0.05α=0.05 and df=24df = 24df=24, we find the critical chi-square values from the chi-square distribution table.

- Critical value for the lower tail (χα/2,242\chi^2_{\alpha/2, 24}χα/2,242): χ0.025,242=12.401\chi^2_{0.025, 24} = 12.401χ0.025,242=12.401
- Critical value for the upper tail (χ1−α/2,242\chi^2_{1-\alpha/2, 24}χ1−α/2,242): χ0.975,242=39.364\chi^2_{0.975, 24} = 39.364χ0.975,242 =39.364

**Step 4: Conclusion**

We compare the calculated chi-square statistic χ2=17.63\chi^2 = 17.63χ2=17.63 with the critical values.

- If χ2\chi^2χ2 falls between the critical values, we fail to reject the null hypothesis.
- If χ2\chi^2χ2 is outside the critical values, we reject the null hypothesis.

In this case:

12.401<17.63<39.36412.401 < 17.63 < 39.36412.401<17.63<39.364

Since the test statistic falls between the critical values, we fail to reject the null hypothesis.

**Conclusion:**

At the α=0.05\alpha = 0.05α=0.05 significance level, there is not enough evidence to reject the null hypothesis. Hence, we conclude that the population standard deviation is likely to be 10.5.

---

**37) 100 students of a PWIOI obtained the following grades in Data Science paper: Grade: (A, B, C, D, E) Total Frequency : [15, 17, 30, 22, 16, 100] Using the x 2 test, examine the hypothesis that the distribution of grades is uniform.**

**Step 1: State the Hypotheses**

Null Hypothesis ($H_0$): The distribution of grades is uniform.
Alternative Hypothesis ($H_1$): The distribution of grades is not uniform.

**Step 2: Calculate Expected Frequencies**

If the distribution is uniform, each grade should have an equal frequency. Since there are 5 grades (A, B, C, D, E) and 100 students, the expected frequency for each grade is:

Ei=Total studentsNumber of grades=1005=20E_i = \frac{\text{Total students}}{\text{Number of grades}} = \frac{100}{5} = 20Ei =Number of gradesTotal students=5100=20

Thus, the expected frequencies EiE_iEi for each grade are: [20, 20, 20, 20, 20].

**Step 3: Calculate the Chi-Square Statistic**

The chi-square statistic is calculated using the formula:

χ2=∑(Oi−Ei)2Ei\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}χ2=∑Ei(Oi−Ei)2

Where:

- OiO_iOi = observed frequency for grade iii
- EiE_iEi = expected frequency for grade iii

Calculation:

For each grade, calculate the individual chi-square contributions:

- For grade A: (15−20)220=2520=1.25\frac{(15 - 20)^2}{20} = \frac{25}{20} = 1.2520(15−20)2=2025=1.25
- For grade B: (17−20)220=920=0.45\frac{(17 - 20)^2}{20} = \frac{9}{20} = 0.4520(17−20)2=209=0.45
- For grade C: (30−20)220=10020=5.00\frac{(30 - 20)^2}{20} = \frac{100}{20} = 5.0020(30−20)2=20100=5.00
- For grade D: (22−20)220=420=0.20\frac{(22 - 20)^2}{20} = \frac{4}{20} = 0.2020(22−20)2=204=0.20
- For grade E: (16−20)220=1620=0.80\frac{(16 - 20)^2}{20} = \frac{16}{20} = 0.8020(16−20)2=2016=0.80

Now sum these contributions:

χ2=1.25+0.45+5.00+0.20+0.80=7.70\chi^2 = 1.25 + 0.45 + 5.00 + 0.20 + 0.80 = 7.70χ2=1.25+0.45+5.00+0.20+0.80=7.70

**Step 4: Determine the Critical Value**

Degrees of freedom (df) = number of categories - 1 = 5 - 1 = 4.

Using a chi-square distribution table at α=0.05\alpha = 0.05α=0.05 and df=4df = 4df=4, the critical value of χ2\chi^2χ2 is approximately 9.488.

**Step 5: Conclusion**

Calculated Compare the calculated chi-square value with the critical value:

- $\chi 2 = 7.70 \backslash chi^2 = 7.70 \chi 2 = 7.70$
- Critical $\chi 2 = 9.488 \backslash chi^2 = 9.488 \chi 2 = 9.488$

Since $7.70 < 9.488 7.70 < 9.488 7.70 < 9.488$, we fail to reject the null hypothesis.
Conclusion:
At the 0.05 significance level, there is not enough evidence to reject the hypothesis that the distribution of grades is uniform. Thus, the grades appear to be uniformly distributed across the students.

---

**39.How would you create a basic Flask route that displays "Hello, World!" on the homepage?**

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def home():
  return "Hello, World!"
if __name__ == '__main__':
  app.run(debug=True)
```

In this example:

- We import Flask, create an instance of it, and define a route (@app.route('/')) for the homepage ("/").
- The home() function returns "Hello, World!" when this route is accessed.

---

**40.Explain how to set up a Flask application to handle form submissions using POST requests.**

**HTML Form:**

```
<form action="/submit" method="POST">
 <input type="text" name="username" placeholder="Enter Username">
 <input type="submit" value="Submit">
</form>
```

**Flask Code:**

```
from flask import Flask, request, render_template
app = Flask(__name__)
@app.route('/submit', methods=['GET', 'POST'])
def submit():
  if request.method == 'POST':
    username = request.form['username']
    return f"Hello, {username}!"
  return render_template('form.html')
if __name__ == '__main__':
  app.run(debug=True)
```

Here:

- The form submits data via POST to the /submit route.
- Flask's request.form retrieves form data from the POST request.

---

**41. Write a Flask route that accepts a parameter in the URL and displays it on the page.**

```
from flask import Flask
app = Flask(__name__)
@app.route('/greet/<name>')
def greet(name):
  return f"Hello, {name}!"
if __name__ == '__main__':
  app.run(debug=True)
```

---

**42.How can you implement user authentication in a Flask application?**

**Steps:**

1. Install Flask-Login: pip install flask-login.
2. Set up user authentication using Flask-Login.

**Example:**

```
from flask import Flask, render_template, redirect, url_for, request
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user

app = Flask(__name__)
app.secret_key = 'your_secret_key'

login_manager = LoginManager()
login_manager.init_app(app)

class User(UserMixin):
  def __init__(self, id):
```

```
        self.id = id

users = {'user1': 'password1'} # Example users

@login_manager.user_loader
def load_user(user_id):
    return User(user_id)

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if username in users and users[username] == password:
            login_user(User(username))
            return redirect(url_for('protected'))
    return render_template('login.html')

@app.route('/protected')
@login_required
def protected():
    return "You are logged in!"

@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))

if __name__ == '__main__':
    app.run(debug=True)
```

This code uses Flask-Login for managing user sessions and checking authentication.
login handles the login form, /protected is a route protected by authentication.

---

**43.Describe the process of connecting a Flask app to a SQLite database using SQLAlchemy.**

**Steps:**

1. Install SQLAlchemy: pip install flask-sqlalchemy.
2. Configure Flask to use SQLite and SQLAlchemy.

**Example:**

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)

if __name__ == '__main__':
    db.create_all()  # This will create the tables in the database
    app.run(debug=True)
```

---

**44.How would you create a RESTful API endpoint in Flask that returns JSON data?**

```
from flask import Flask, jsonify
app = Flask(__name__)

@app.route('/api/data', methods=['GET'])
def get_data():
    data = {'name': 'John', 'age': 30}
    return jsonify(data)

if __name__ == '__main__':
    app.run(debug=True)
```

**Here:**

- The /api/data endpoint returns JSON data using Flask's jsonify function.

---

**45.Explain how to use Flask-WTF to create and validate forms in a Flask application.**

1. Install Flask-WTF: pip install flask-wtf.
2. Set up a form using Flask-WTF.

**Example:**

```
from flask import Flask, render_template
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired

app = Flask(__name__)
app.secret_key = 'secret_key'

class MyForm(FlaskForm):
  name = StringField('Name', validators=[DataRequired()])
  submit = SubmitField('Submit')

@app.route('/', methods=['GET', 'POST'])
def index():
  form = MyForm()
  if form.validate_on_submit():
    return f'Hello, {form.name.data}!'
  return render_template('form.html', form=form)

if __name__ == '__main__':
  app.run(debug=True)
```

---

**46.How can you implement file uploads in a Flask application?**

**HTML Form:**

```
<form action="/upload" method="POST" enctype="multipart/form-data">
  <input type="file" name="file">
  <input type="submit">
</form>
```

**Flask Code:**

```
from flask import Flask, request
app = Flask(__name__)

@app.route('/upload', methods=['POST'])
def upload_file():
  if 'file' not in request.files:
    return 'No file part'
  file = request.files['file']
  if file.filename == '':
    return 'No selected file'
  file.save(f"./uploads/{file.filename}")
  return 'File successfully uploaded!'

if __name__ == '__main__':
  app.run(debug=True)
```

---

**47.Describe the steps to create a Flask blueprint and why you might use one.**

**Why Use Blueprints?**

- Modularization: Organize code by grouping related routes and logic.
- Reusability: Reuse blueprint components across different applications.
- Maintainability: Manage large applications by dividing code into smaller, manageable parts.
- Extensibility: Blueprints can be registered multiple times in different configurations.

1. Create a Blueprint
2. Define Routes Within the Blueprint
3. Register the Blueprint in the Flask App
4. Blueprints for Larger Applications
5. Example: Auth Blueprint

**Advantages of Using Blueprints**

- **Code Organization:** Break down your app by functionality (e.g., auth, admin, user, etc.).
- **Scalability:** As your application grows, you can keep each part modular and avoid a large monolithic codebase.
- **Reusability:** Blueprints can be reused across multiple applications, helping you avoid repeating code.
- **Easier Collaboration:** In a team environment, developers can work on different blueprints independently.

**48.How would you deploy a Flask application to a production server using Gunicorn and Nginx?**

**Steps:**

1. Install Gunicorn: pip install gunicorn.
2. Run Flask with Gunicorn: gunicorn app:app.
3. Set up Nginx as a reverse proxy to Gunicorn by configuring an Nginx server block to forward requests to the Gunicorn server.

**49. Make a fully functional web application using flask, Mangodb. Signup, Signin page. And after successfully login.Say hello Geeks message at webpage.**

To create a web app with Flask and MongoDB:

1. Install Flask-PyMongo: pip install flask-pymongo.
2. Set up a basic signup and signin system, connecting to MongoDB for user storage.

```
from flask import Flask, request, render_template, redirect, url_for
from flask_pymongo import PyMongo

app = Flask(__name__)
app.config["MONGO_URI"] = "mongodb://localhost:27017/mydatabase"
mongo = PyMongo(app)

@app.route('/signup', methods=['POST', 'GET'])
def signup():
  if request.method == 'POST':
    username = request.form['username']
    password = request.form['password']
    mongo.db.users.insert_one({'username': username, 'password': password})
    return redirect(url_for('signin'))
  return render_template('signup.html')

@app.route('/signin', methods=['POST', 'GET'])
def signin():
  if request.method == 'POST':
    username = request.form['username']
    user = mongo.db.users.find_one({'username': username})
    if user and user['password'] == request.form['password']:
      return "Hello Geeks!"
  return render_template('signin.html')

if __name__ == '__main__':
  app.run(debug=True)
```

**50)Machine Learning:**

**Q) What is the difference between Series & Dataframes.**

**Difference Between Series & DataFrames**

- Series: A Series is a one-dimensional labeled array capable of holding any data type (integer, string, float, etc.). It is like a single column in a table, where each value has an associated label (index).
- DataFrame: A DataFrame is a two-dimensional labeled data structure with columns of potentially different types. It is equivalent to a table in a relational database or a spreadsheet with rows and columns.

**Key Differences:**

- Dimensionality: Series is 1-dimensional, DataFrame is 2-dimensional.
- Use case: Series represents a single column of data, while DataFrame represents multiple columns (a table).

**Q) Create a database name Travel_Planner in mysql, and create a table name bookings in that which having attributes (user_id INT, flight_id INT,hotel_id INT, activity_id INT,booking_date DATE) .fill with some dummy value.Now you have to read the content of this table using pandas as dataframe.Show the output.**

**SQL Code:**

```
--Creating database and table in MySQL
CREATE DATABASE Travel_Planner;
USE Travel_Planner;

CREATE TABLE bookings (
  user_id INT,
  flight_id INT,
  hotel_id INT,
  activity_id INT,
  booking_date DATE
);
```

```
-- Insert some dummy values
INSERT INTO bookings (user_id, flight_id, hotel_id, activity_id, booking_date)
VALUES
  (1, 101, 201, 301, '2024-01-01'),
  (2, 102, 202, 302, '2024-02-01'),
  (3, 103, 203, 303, '2024-03-01');
```

**Pandas Code:**

```
import pandas as pd
import mysql.connector

# Connect to MySQL database
conn = mysql.connector.connect(
  host="localhost",
  user="your_username",
  password="your_password",
  database="Travel_Planner"
)

# Query to select all data from bookings table
query = "SELECT * FROM bookings"

# Read SQL query into a DataFrame
df = pd.read_sql(query, conn)

# Print the DataFrame
print(df)

# Close the connection
conn.close()
```

**Output:**

| | user_id | flight_id | hotel_id | activity_id | booking_date |
|---|---|---|---|---|---|
| 0 | 1 | 101 | 201 | 301 | 2024-01-01 |
| 1 | 2 | 102 | 202 | 302 | 2024-02-01 |
| 2 | 3 | 103 | 203 | 303 | 2024-03-01 |

---

**Q) Difference between loc and iloc.**

**loc:** Label-based indexing. It is used to access rows and columns by their labels or boolean arrays.
- Example: df.loc[2] accesses the row with index label 2.

**iloc:** Integer-based (positional) indexing. It is used to access rows and columns by their positions (index numbers).
- Example: df.iloc[2] accesses the third row by position.

---

**Q) What is the difference between supervised and unsupervised learning?**

**Supervised Learning:**
- Data includes both input features and corresponding output labels.
- Goal: Learn a mapping from inputs to the output labels.
- Examples: Regression, classification (e.g., predicting house prices, classifying emails as spam).

**Unsupervised Learning:**
- Data includes only input features without any output labels.
- Goal: Discover hidden patterns or structure in the data.
- Examples: Clustering, dimensionality reduction (e.g., segmenting customers, reducing feature dimensions).

---

**Q) Explain the bias-variance tradeoff.**

- **Bias:** Error due to overly simplistic assumptions in the model. High bias leads to underfitting (model fails to capture the underlying data patterns).
- **Variance:** Error due to excessive sensitivity to small fluctuations in the training data. High variance leads to overfitting (model learns noise rather than the actual pattern).

**Tradeoff**: The goal is to find the right balance between bias and variance. Increasing model complexity reduces bias but increases variance, while simplifying the model reduces variance but increases bias.

---

**Q) What are precision and recall? How are they different from accuracy?**

**Precision:** The ratio of true positive predictions to the total number of positive predictions. It answers the question: "How many of the predicted positives were actually positive?"
**Recall:** The ratio of true positive predictions to the total number of actual positives. It answers the question: "How many of the actual positives were correctly predicted
**Accuracy:** The ratio of correct predictions (both true positives and true negatives) to the total number of predictions.

**Q) What is overfitting and how can it be prevented?**

**Overfitting:** Occurs when a model performs well on the training data but poorly on unseen test data. The model becomes too complex, capturing noise and fluctuations in the training data rather than the underlying trend.
**Prevention Methods:**

- Regularization: Techniques like L1/L2 regularization add a penalty for complexity in the model.
- Cross-Validation: Splitting the data into multiple training and validation sets to ensure the model generalizes well.
- Simplification: Reducing the number of features or parameters to make the model simpler.
- Pruning (for Decision Trees): Removing parts of the tree that do not provide significant power to the model.
- Early Stopping: In iterative algorithms like neural networks, stopping training once the model's performance on the validation data begins to deteriorate.

---

**Q) Explain the concept of cross-validation.**

Cross-validation is a technique used to assess how a machine learning model generalizes to an independent dataset. It involves splitting the data into multiple subsets, training the model on some subsets (training set), and validating it on the remaining subsets (validation set). The most common form is k-fold cross-validation, where the data is split into *k* equal-sized folds, and the model is trained and validated *k* times, with each fold being used as the validation set once.

---

**Q) What is the difference between a classification and a regression problem?**

**Classification:** Involves predicting discrete labels or categories. The output variable is categorical (e.g., spam/not spam, 0/1).
**Regression:** Involves predicting continuous numerical values. The output variable is continuous (e.g., predicting house prices).

---

**Q) Explain the concept of ensemble learning.**

**Ensemble learning** is a technique that combines multiple models (often called "weak learners") to produce a better-performing model (a "strong learner"). The goal is to reduce model variance or bias and improve prediction accuracy. Common ensemble methods include bagging (e.g., Random Forest) and boosting (e.g., XGBoost).

---

**Q) What is gradient descent and how does it work?**

**Gradient descent** is an optimization algorithm used to minimize a function (e.g., a cost function in machine learning). It works by iteratively adjusting the model parameters in the opposite direction of the gradient (slope) of the cost function with respect to the parameters, leading to the minimum value.

---

**Q) Describe the difference between batch gradient descent and stochastic gradient descent.**

**Batch Gradient Descent:** Calculates the gradient using the entire training dataset. It updates the parameters once per iteration based on the average gradient.
**Stochastic Gradient Descent (SGD):** Calculates the gradient using one randomly chosen data point at a time. It updates parameters for each training example, which makes it faster but noisier.

---

**Q) What is the curse of dimensionality in machine learning?**

The curse of dimensionality refers to the phenomenon where the feature space becomes exponentially larger as the number of features increases. This leads to overfitting and poor model performance due to the sparsity of data in high-dimensional spaces.

---

**Q) Explain the difference between L1 and 12 regularization.**

**L1 Regularization (Lasso):** Adds the absolute values of the coefficients as a penalty term to the loss function. It can lead to sparse models (some coefficients being exactly zero).
**L2 Regularization (Ridge):** Adds the squared values of the coefficients as a penalty term. It tends to shrink coefficients towards zero but rarely results in zero coefficients.

---

**Q) What is a confusion matrix and how is it used?**

A confusion matrix is a table used to evaluate the performance of a classification model. It shows the number of true positives, false positives, true negatives, and false negatives, providing insight into model accuracy, precision, recall, and more.

---

**Q) Define AUC-ROC curve.**

The AUC-ROC curve is a graphical representation of a classifier's performance. The ROC curve plots the true positive rate (sensitivity) against the false positive rate, while AUC (Area Under the Curve) measures the overall performance, where a higher value indicates a better model.

---

**Q) Explain the k-nearest neighbors algorithm.**

K-Nearest Neighbors (KNN) is a simple, non-parametric classification algorithm that assigns a class label based on the majority class among the *k* closest data points (neighbors) to a query point, usually measured by Euclidean distance.

---

**Q) Explain the basic concept of a Support Vector Machine (SVM).**

An SVM is a supervised learning algorithm used for classification and regression. It finds the optimal hyperplane that separates classes in the feature space with the maximum margin between the nearest points of the classes (support vectors).

---

**Q) How does the kernel trick work in SVM?**

The kernel trick allows SVM to work efficiently in high-dimensional spaces by transforming the original input space into a higher-dimensional space where the data becomes linearly separable. Common kernels include linear, polynomial, and radial basis function (RBF).

---

**Q) What are the different types of kernels used in SVM ond when would you use each?**

**Linear Kernel:** Used when the data is linearly separable.
**Polynomial Kernel:** Used for polynomial relationships between features.
**Radial Basis Function (RBF) Kernel**: Useful for non-linear problems with more complex relationships.).

---

**Q) What is the hyperplane in SVM and how is it determined?**

The hyperplane is a decision boundary that separates different classes. In SVM, it is determined by maximizing the margin between the nearest points (support vectors) from each class.

---

**Q) What are the pros and cons of using a Support Vector Machine (SVM)?**

Effective in high-dimensional spaces.
Works well with clear margin separation. Cons:
Not effective when classes overlap.
Computationally intensive for large datasets.

---

**Q) Explain the difference between a hard margin and a soft margin SVM.**

**Hard Margin:** The model enforces that all points are correctly classified with no tolerance for misclassification. Works only with perfectly separable data.
**Soft Margin:** Allows some misclassifications to handle noisy data and improve generalization.

---

**Q) Describe the process of constructing a decision tree.**

The decision tree construction involves recursively splitting the data based on features that result in the largest information gain (or lowest impurity) until the data cannot be split further or a stopping criterion is met.

---

**Q) Describe the working principle of a decision tree.**

A decision tree works by splitting the data at each node based on a feature that maximizes the separation of the classes (in classification) or reduces variance (in regression). This creates a hierarchical tree structure that leads to predictions.

---

**Q) What is information gain and how is it used in decision trees?**

Information gain measures how well a feature separates the data based on the target variable. It is calculated as the reduction in entropy or impurity after a dataset is split on an attribute. It is used to decide which feature to split on at each node.

---

**Q) What are the advantages and disadvantages of decision trees?**

**Advantages:**
Easy to interpret and visualize.
Can handle both categorical and numerical data.
**Disadvantages:**
Prone to overfitting.
Sensitive to small changes in the data.

**Q) Explain Gini impurity and its role in decision trees.**

Gini impurity measures the likelihood of an incorrect classification of a randomly chosen element from the dataset. It is used in decision trees to evaluate splits by selecting the one that minimizes impurity.

**Q) How do random forests improve upon decision trees?**

Random forests improve upon decision trees by building an ensemble of decision trees on random subsets of data and features. The final prediction is made by averaging (regression) or majority voting (classification), reducing overfitting.

**Q) How does a random forest algorithm work?**

A random forest builds multiple decision trees using random samples from the data. Each tree is built using a random subset of features. The final prediction is the aggregate of predictions from all trees.

**Q) What is bootstrapping in the context of random forests?**

Bootstrapping is a statistical technique where multiple subsets of data are sampled (with replacement) from the original data set. In random forests, each decision tree is trained on a bootstrap sample, leading to model diversity.

**Q) Explain the concept of feature importance in random forests.**

**Feature importance** in random forests measures how much a feature contributes to the prediction. It is calculated by looking at how much a feature reduces impurity across all trees.

**Q) What are the key hyperparameters of a random forest and how do they affect the model?**

**Number of trees (n_estimators):** More trees generally lead to better performance but increase computation time.
**Max depth:** Controls the maximum depth of trees to prevent overfitting.
**Min samples split:** The minimum number of samples required to split a node, controlling the tree's complexity.

**Q) Describe the logistic regression model and its assumptions.**

Logistic regression is used for binary classification. It models the probability that an observation belongs to a particular class using the logistic (sigmoid) function. Key assumptions include:

- Linearity between features and log-odds.
- Independent observations.

**Q) How does logistic regression handle binary classification problems?**

Logistic regression applies the sigmoid function to the linear combination of features to output probabilities, which are then thresholded (e.g., at 0.5) to classify observations into two classes.

**Q) What is the sigmoid function and how is it used in logistic regression?**

The sigmoid function maps any real-valued number to the range (0, 1). It is used in logistic regression to output the probability of a sample belonging to the positive class.

**Q) Explain the concept of the cost function in logistic regression.**

The **cost function** in logistic regression, often the **log-loss** or binary cross-entropy, measures the difference between the predicted probabilities and the actual labels. The model is trained to minimize this cost.

**Q) How can logistic regression be extended to handle multiclass classification?**

Multinomial logistic regression (also called softmax regression) can handle more than two classes by generalizing the sigmoid function into the softmax function, which outputs a probability distribution over multiple classes.

**Q) What is the difference between L1 and 12 regularization in logistic regression?**

**L1 Regularization (Lasso):** Adds an absolute value of coefficients to the cost function. Leads to sparse models with some coefficients being zero.
**L2 Regularization (Ridge):** Adds squared coefficients to the cost function. Shrinks coefficients but does not set them to zero.

---

**Q) What is XGBoost and how does it differ from other boosting algorithms?**

XGBoost is a high-performance implementation of gradient boosting that is optimized for speed and efficiency. It improves upon other boosting algorithms with regularization, handling missing data, and parallel processing.

---

**Q) Explain the concept of boosting in the context of ensemble learning.**

Boosting is an ensemble technique where models are trained sequentially, with each new model focusing on correcting the errors made by previous models. The final model is a weighted sum of the weak learners.

---

**Q) How does XGBoost handle missing values?**

XGBoost automatically learns how to handle missing values by assigning them to the optimal branch in the decision tree, ensuring that missing values do not negatively impact performance.

---

**Q) What are the key hyperparameters in XGBoost and how do they affect model performance.**

**Learning rate (eta):** Controls how much each tree contributes to the final prediction.
**Max depth:** Limits the depth of trees to prevent overfitting.
**n_estimators:** The number of trees in the model.
**subsample:** The fraction of samples used to build each tree.

---

**Q) Describe the process of gradient boosting in XGBoost.**

Gradient boosting builds trees sequentially, with each tree trying to minimize the residual errors of the previous trees. In XGBoost, regularization and other enhancements make the process faster and more accurate.

---

**Q) What are the advantages and disadvantages of using XGBoost?**

High accuracy and performance.
Handles missing values well.
Efficient with large datasets. Disadvantages:
Computationally intensive.
Sensitive to hyperparameters.

---