Assignments

1. What is the difference between static and dynamic variables in Python?

Static variables are variables that are shared among all instances of a class. They are defined within a class but outside any instance methods and can be accessed using the class name. Dynamic variables, on the other hand, are instance-specific variables. They are defined within methods and belong to the individual instance of the class.

Example:

```
class MyClass:
    static_var = 0  # Static variable

    def __init__(self, var):
        self.dynamic_var = var  # Dynamic variable

obj1 = MyClass(5)
obj2 = MyClass(10)

print(MyClass.static_var)  # Output: 0
print(obj1.dynamic_var)    # Output: 5
print(obj2.dynamic_var)    # Output: 10
```

2. Explain the purpose of `pop()`, `popitem()`, `clear()` in a dictionary with suitable examples.

- `pop(key)`: Removes the specified key and returns the corresponding value.
- `popitem()`: Removes and returns an arbitrary (key, value) pair as a tuple.
- `clear()`: Removes all items from the dictionary.

Examples:

```
my_dict = {'a': 1, 'b': 2, 'c': 3}

# pop()
value = my_dict.pop('b')
```

```
print(value)        # Output: 2
print(my_dict)      # Output: {'a': 1, 'c': 3}

# popitem()
item = my_dict.popitem()
print(item)         # Output: ('c', 3)
print(my_dict)      # Output: {'a': 1}

# clear()
my_dict.clear()
print(my_dict)      # Output: {}
```

3. What do you mean by FrozenSet? Explain it with suitable examples.

A `frozenset` is an immutable version of a Python set. Elements of a frozenset cannot be modified after creation, which makes them hashable and able to be used as keys in a dictionary or as elements of another set.

Example:

```
frozen_set = frozenset([1, 2, 3, 4])
print(frozen_set)  # Output: frozenset({1, 2, 3, 4})

# Attempting to add or remove elements will raise an error
# frozen_set.add(5)  # AttributeError: 'frozenset' object has no attribute 'add'
```

4. Differentiate between mutable and immutable data types in Python and give examples of mutable and immutable data types.

- **Mutable data types** can be changed after creation. Examples include lists, dictionaries, and sets.
- **Immutable data types** cannot be changed after creation. Examples include strings, tuples, and frozensets.

Examples:

```
# Mutable
my_list = [1, 2, 3]
my_list.append(4)
print(my_list)  # Output: [1, 2, 3, 4]
```

```
my_dict = {'a': 1, 'b': 2}
my_dict['c'] = 3
print(my_dict)  # Output: {'a': 1, 'b': 2, 'c': 3}

# Immutable
my_str = "hello"
# my_str[0] = 'H'  # TypeError: 'str' object does not support item assignment

my_tuple = (1, 2, 3)
# my_tuple[0] = 0  # TypeError: 'tuple' object does not support item assignment
```

5. What is `__init__`? Explain with an example.

`__init__` is a special method in Python classes, known as the constructor. It is automatically called when a new instance of a class is created and is used to initialize the instance's attributes.

Example:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

p1 = Person("John", 30)
print(p1.name)  # Output: John
print(p1.age)   # Output: 30
```

6. What is docstring in Python? Explain with an example.

A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition. It is used to document the object and can be accessed using the `__doc__` attribute.

Example:

```
def my_function():
    """This is a docstring for my_function."""
    pass
```

```
print(my_function.__doc__)  # Output: This is a docstring for my_function.
```

7. What are unit tests in Python?

Unit tests are a method of testing individual units of source code to determine if they are fit for use. In Python, the `unittest` module is used to create and run unit tests.

Example:

```
import unittest

def add(a, b):
    return a + b

class TestAddFunction(unittest.TestCase):
    def test_add(self):
        self.assertEqual(add(1, 2), 3)
        self.assertEqual(add(-1, 1), 0)

if __name__ == '__main__':
    unittest.main()
```

8. What is break, continue and pass in Python?

- `break`: Terminates the loop containing it.
- `continue`: Skips the current iteration and proceeds to the next iteration of the loop.
- `pass`: Does nothing and is used as a placeholder.

Examples:

```
# break
for i in range(5):
    if i == 3:
        break
    print(i)  # Output: 0 1 2

# continue
for i in range(5):
    if i == 3:
        continue
    print(i)  # Output: 0 1 2 4

# pass
```

```
for i in range(5):
    if i == 3:
        pass  # Does nothing
    print(i)  # Output: 0 1 2 3 4
```

9. What is the use of self in Python?

`self` is a reference to the current instance of the class. It is used to access variables that belong to the class.

Example:

```
class MyClass:
    def __init__(self, value):
        self.value = value

    def print_value(self):
        print(self.value)

obj = MyClass(10)
obj.print_value()  # Output: 10
```

10. What are global, protected, and private attributes in Python?

- **Global attributes**: Accessible from anywhere in the code.
- **Protected attributes**: Prefixed with a single underscore (e.g., `_var`) and are intended for internal use in the class and its subclasses.
- **Private attributes**: Prefixed with a double underscore (e.g., `__var`) and are intended for internal use within the class only.

Example:

```
class MyClass:
    def __init__(self):
        self.public_var = 1
        self._protected_var = 2
        self.__private_var = 3

obj = MyClass()
print(obj.public_var)       # Output: 1
print(obj._protected_var)   # Output: 2
# print(obj.__private_var)  # AttributeError: 'MyClass' object has no attribute
'__private_var'
```

```
print(obj._MyClass__private_var)  # Output: 3 (name mangling)
```

11. What are modules and packages in Python?

- **Module**: A single file containing Python code that can be imported and used in other Python files.
- **Package**: A directory containing multiple modules and an `__init__.py` file, allowing it to be treated as a single unit.

Example:
```python
# my_module.py
def my_function():
    print("Hello from my_module")

# main.py
import my_module
my_module.my_function()  # Output: Hello from my_module
```

12. What are lists and tuples? What is the key difference between the two?

- **List**: A mutable, ordered collection of items. Items can be added, removed, or changed.
- **Tuple**: An immutable, ordered collection of items. Items cannot be changed after creation.

Example:

```
# List
my_list = [1, 2, 3]
my_list.append(4)
print(my_list)  # Output: [1, 2, 3, 4]

# Tuple
my_tuple = (1, 2, 3)
# my_tuple[0] = 0  # TypeError: 'tuple' object does not support item assignment
```

13. What is an interpreted language and dynamically typed language? Write 5 differences between them.

- **Interpreted language**: A language where the code is executed line by line by an interpreter.
- **Dynamically typed language**: A language where the type of a variable is determined at runtime.

Differences:
1. **Compilation**: Interpreted languages are not compiled into machine code; dynamically typed languages can be either compiled or interpreted.
2. **Error Detection**: In interpreted languages, errors are detected at runtime; in dynamically typed languages, type errors are detected at runtime.
3. **Performance**: Interpreted languages are generally slower than compiled languages; dynamically typed languages may have performance overhead due to runtime type checking.
4. **Flexibility**: Interpreted languages allow for interactive coding and quick testing; dynamically typed languages offer flexibility in variable usage.
5. **Type Declaration**: Interpreted languages do not require type declarations; dynamically typed languages do not require explicit type declarations.

```
x = 10
x = "Hello"
```

14. What are Dict and List comprehensions?

- **List comprehension**: A concise way to create lists using a single line of code.
- **Dict comprehension**: A concise way to create dictionaries using a single line of code.

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

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

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

Decorators are a way to modify or enhance the behavior of functions or methods. They allow you to wrap another function to extend its behavior without permanently modifying it.

Example:
```python
def my_decorator(func):
    def wrapper():
        print("Something is happening before the function is called.")
        func()
        print("Something is happening after the function is called.")
    return wrapper

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

say_hello()
# Output:
# Something is happening before the function is called.
# Hello!
# Something is happening after the function is called.
```

Use cases:
- Logging
- Access control and authentication
- Caching
- Validation
- Timing functions

16. How is memory managed in Python?

Memory management in Python involves a private heap containing all Python objects and data structures. The management of this private heap is ensured internally by the Python memory manager. Python uses reference counting and garbage collection to manage memory.

Example:
```python
import sys

a = []
```

```
print(sys.getrefcount(a))  # Output: 2 (one reference from `a`, one from getrefcount argument)

b = a
print(sys.getrefcount(a))  # Output: 3 (additional reference from `b`)
```

17. What is lambda in Python? Why is it used?

A `lambda` function is a small anonymous function defined with the `lambda` keyword. It can take any number of arguments but can only have one expression. It is used for creating small, throwaway functions without the need for formal function definitions.

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

18. Explain `split()` and `join()` functions in Python?

- `split()`: Splits a string into a list where each word is a list item.
- `join()`: Joins the elements of a list into a single string with a specified separator.

Examples:

```
# split()
text = "Hello world"
words = text.split()
print(words)  # Output: ['Hello', 'world']

# join()
words = ['Hello', 'world']
text = ' '.join(words)
print(text)  # Output: Hello world
```

19. What are iterators, iterable & generators in Python?

- **Iterable**: An object capable of returning its members one at a time, such as lists, tuples, and strings.
- **Iterator**: An object representing a stream of data; it returns the next item in the sequence when the `__next__()` method is called.

- **Generator**: A special type of iterator created using a function with `yield` statements, which allows iteration over a sequence of values.

Examples:

```
# Iterable
my_list = [1, 2, 3]
for item in my_list:
    print(item)

# Iterator
my_iterator = iter(my_list)
print(next(my_iterator))  # Output: 1
print(next(my_iterator))  # Output: 2

# Generator
def my_generator():
    yield 1
    yield 2
    yield 3

for value in my_generator():
    print(value)  # Output: 1 2 3
```

20. What is the difference between `xrange` and `range` in Python?

In Python 2, `range()` returns a list, while `xrange()` returns an iterator that generates the numbers on demand. In Python 3, `xrange()` is removed, and `range()` behaves like `xrange()` from Python 2, returning an immutable sequence type.

Example in Python 2:

```python
# range()
print(range(5))  # Output: [0, 1, 2, 3, 4]

# xrange()
for i in xrange(5):
    print(i)  # Output: 0 1 2 3 4
```

21. Pillars of OOPs.

The four main pillars of Object-Oriented Programming (OOP) are:

1. **Encapsulation**: Bundling the data and methods that operate on the data within one unit (class).
2. **Abstraction**: Hiding the complex implementation details and showing only the necessary features.
3. **Inheritance**: Mechanism by which one class can inherit the attributes and methods of another class.
4. **Polymorphism**: The ability to take many forms; allows methods to do different things based on the object it is acting upon.

22. How will you check if a class is a child of another class?

You can use the `issubclass()` function to check if a class is a subclass of another class.

Example:
```python
class Parent:
    pass

class Child(Parent):
    pass

print(issubclass(Child, Parent))  # Output: True
```

23. 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. The types of inheritance in Python are:
- **Single Inheritance**: A child class inherits from one parent class.
- **Multiple Inheritance**: A child class inherits from more than one parent class.
- **Multilevel Inheritance**: A child class inherits from a parent class, which in turn inherits from another class.
- **Hierarchical Inheritance**: Multiple child classes inherit from one parent class.
- **Hybrid Inheritance**: A combination of two or more types of inheritance.

Examples:
```python
# Single Inheritance
class Parent:
    def func1(self):
        print("Parent")

class Child(Parent):
```

```
    def func2(self):
        print("Child")

obj = Child()
obj.func1()  # Output: Parent

# Multiple Inheritance
class Mother:
    def func1(self):
        print("Mother")

class Father:
    def func2(self):
        print("Father")

class Child(Mother, Father):
    def func3(self):
        print("Child")

obj = Child()
obj.func1()  # Output: Mother
obj.func2()  # Output: Father

# Multilevel Inheritance
class GrandParent:
    def func1(self):
        print("GrandParent")

class Parent(GrandParent):
    def func2(self):
        print("Parent")

class Child(Parent):
    def func3(self):
        print("Child")

obj = Child()
obj.func1()  # Output: GrandParent

# Hierarchical Inheritance
class Parent:
    def func1(self):
        print("Parent")
```

```
class Child1(Parent):
    def func2(self):
        print("Child1")

class Child2(Parent):
    def func3(self):
        print("Child2")

obj1 = Child1()
obj2 = Child2()
obj1.func1()  # Output: Parent
obj2.func1()  # Output: Parent

# Hybrid Inheritance
class Base1:
    def func1(self):
        print("Base1")

class Base2:
    def func2(self):
        print("Base2")

class Derived1(Base1, Base2):
    def func3(self):
        print("Derived1")

class Derived2(Derived1):
    def func4(self):
        print("Derived2")

obj = Derived2()
obj.func1()  # Output: Base1
obj.func2()  # Output: Base2
```

24. What is encapsulation? Explain it with an example.

Encapsulation is the wrapping of data and methods into a single unit, typically a class. It restricts direct access to some of an object's components, which can prevent accidental modification of data.

Example:
```python
class MyClass:
```

```
    def __init__(self, value):
        self.__hidden = value  # Private variable

    def get_hidden(self):
        return self.__hidden

    def set_hidden(self, value):
        self.__hidden = value

obj = MyClass(10)
print(obj.get_hidden())  # Output: 10
obj.set_hidden(20)
print(obj.get_hidden())  # Output: 20
```

25. What is polymorphism? Explain it with an example.

Polymorphism allows methods to do different things based on

the object it is acting upon. In Python, it can be achieved through method overriding and operator overloading.

Example:

```
class Bird:
    def sound(self):
        print("Chirp")

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

def make_sound(animal):
    animal.sound()

bird = Bird()
dog = Dog()

make_sound(bird)  # Output: Chirp
make_sound(dog)   # Output: Bark
```

Question 1.2: Which of the following identifier names are invalid and why?

a) **Serial_no.**
- **Valid**: This identifier name follows the rules of identifiers in Python. It starts with a letter and includes only alphanumeric characters and underscores.

b) **1st_Room**
- **Invalid**: In Python, identifiers cannot start with a digit. This identifier starts with '1'.

c) **Hundred$**
- **Invalid**: Identifiers in Python can only include letters, digits, and underscores. The dollar sign ('$') is not allowed.

d) **Total_Marks**
- **Valid**: This identifier name is valid because it starts with a letter and includes only alphanumeric characters and underscores.

e) **total-Marks**
- **Invalid**: Identifiers in Python cannot include hyphens ('-'). They can only include letters, digits, and underscores.

f) **Total Marks**
- **Invalid**: Identifiers cannot include spaces. Identifiers should be a single continuous string of characters.

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

h) **_Percentag**
- **Valid**: This identifier name is valid because it starts with an underscore and includes only alphanumeric characters and underscores.

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

Indicates the central point of a data set, commonly using mean, median, and mode.

Measures of Dispersion: Describe the spread of data around the central point, using range, variance, and standard deviation.

- **Mean:** Sum of all values divided by the number of values.

- **Median:** Middle value when data is sorted.
- **Mode:** Most frequent value.
- **Range:** Difference between the highest and lowest values.
- **Variance:** Average of the squared differences from the mean.
- **Standard Deviation:** Square root of variance.

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

Skewness: Measures the asymmetry of the probability distribution of a real-valued random variable.

- **Positive Skew (Right-skewed):** Tail on the right side is longer.
- **Negative Skew (Left-skewed):** Tail on the left side is longer.

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

Probability Mass Function (PMF): Defines the probability distribution for a discrete random variable.

Probability Density Function (PDF): Defines the probability distribution for a continuous random variable.

Difference: PMF applies to discrete variables, while PDF applies to continuous variables.

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

Correlation: Measures the strength and direction of the linear relationship between two variables.

- **Positive Correlation:** Variables increase together.
- **Negative Correlation:** One variable increases while the other decreases.
- **No Correlation:** No linear relationship.

Methods: Pearson correlation, Spearman rank correlation, Kendall's tau.

Question 24: 24. Calculate coefficient of correlation between the marks obtained by 10 students in Accountancy and statistics:

Calculate using Karl Pearson's method:

r=n(∑xy)−(∑x)(∑y)[n∑x2−(∑x)2][n∑y2−(∑y)2]r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n \sum x^2 - (\sum x)^2][n \sum y^2 - (\sum y)^2]}}r=[n∑x2−(∑x)2][n∑y2−(∑y)2]n(∑xy)−(∑x)(∑y)

Question 25. Discuss the 4 differences between correlation and regression.

1. **Purpose:** Correlation measures the relationship; regression predicts values.
2. **Symmetry:** Correlation is symmetric; regression is not.
3. **Units:** Correlation is unitless; regression depends on units.
4. **Causality:** Regression implies causation; correlation does not.

Question 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.

Price at Delhi=0.8×70=56\text{Price at Delhi} = 0.8 \times 70 = 56Price at Delhi=0.8×70=56

Question 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 − 18y = 214. What are (a) the

mean values of x and y, (b) the coefficient of correlation between x and y, (c) the σ of y.

Given regression equations:

1. 8x−10y=−668x - 10y = -668x−10y=−66
2. 40x−18y=21440x - 18y = 21440x−18y=214

Solve for mean values of xxx and yyy:

x=−66+10y8x = \frac{-66 + 10y}{8}x=8−66+10y y=40x−21418y = \frac{40x - 214}{18}y=1840x−214

Question 28: What is Normal Distribution? What are the four Assumptions of Normal Distribution? Explain in detail.

Assumptions:

1. Data is continuous.
2. Symmetrical distribution.
3. Mean, median, mode are equal.
4. Follows empirical rule.

Question 29:Write all the characteristics or Properties of the Normal Distribution Curve.

1. Symmetrical about the mean.
2. Mean, median, and mode are equal.
3. Asymptotic to the horizontal axis.
4. Area under curve equals 1.

Question 30: Which of the following options are correct about Normal Distribution Curve.

(a) Within a range 0.6745 of σ on both sides the middle 50% of the observations occur i,e. mean ±0.6745σ

covers 50% area 25% on each side.

(b) Mean ±1S.D. (i,e.μ ± 1σ) covers 68.268% area, 34.134 % area lies on either side of the mean.

(c) Mean ±2S.D. (i,e. μ ± 2σ) covers 95.45% area, 47.725% area lies on either side of the mean.

(d) Mean ±3 S.D. (i,e. μ ±3σ) 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 ±3σ.

(a) Correct (b) Correct (c) Correct (d) Correct (e) Correct

Question 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, Standard Deviation = 10

(i) Between 60 and 72: $Z = \frac{72 - 60}{10} = 1.2$ Using Z-tables, area = 0.3849 (38.49%)

(ii) Between 50 and 60: $Z = \frac{50 - 60}{10} = -1$ Using Z-tables, area = 0.3413 (34.13%)

(iii) Beyond 72: Area beyond Z = 1.2 is $1 - 0.8849 = 0.1151$ (11.51%)

(iv) Between 70 and 80: $Z = \frac{70 - 60}{10} = 1$ $Z = \frac{80 - 60}{10} = 2$ Using Z-tables, area = 0.3413 - 0.4772 = 0.1359 (13.59%)

Question 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

Given: Mean = 49, Standard Deviation = 6

(a) More than 55 marks: $Z = \frac{55 - 49}{6} = 1$ Using Z-tables, area = 0.8413 Proportion = $1 - 0.8413 = 0.1587$ (15.87%)

(b) More than 70 marks: $Z = \frac{70 - 49}{6} = 3.5$ Using Z-tables, area = 0.9997 Proportion = $1 - 0.9997 = 0.0003$ (0.03%)

Question 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.

Given: Mean = 65 inches, Standard Deviation = 5 inches

(a) Greater than 70 inches: Z=70−655=1Z = \frac{70 - 65}{5} = 1Z=570−65=1 Using Z-tables, area = 0.8413 Proportion = 1−0.8413=0.15871 - 0.8413 = 0.15871−0.8413=0.1587 Number of students = 0.1587×500=790.1587 \times 500 = 790.1587×500=79

(b) Between 60 and 70 inches: Z1=60−655=−1Z_1 = \frac{60 - 65}{5} = -1Z1=560−65=−1 Z2=70−655=1Z_2 = \frac{70 - 65}{5} = 1Z2=570−65=1 Using Z-tables, area = 0.3413 + 0.3413 = 0.6826 Number of students = 0.6826×500=3410.6826 \times 500 = 3410.6826×500=341

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

Statistical Hypothesis: A claim or assertion about a population parameter.

- **Null Hypothesis (H0):** No effect or difference.
- **Alternative Hypothesis (H1):** Some effect or difference.

Errors in Hypothesis Testing:

- **Type I Error:** Rejecting H0 when it is true.
- **Type II Error:** Not rejecting H0 when it is false.

Sample:

- **Large Samples:** n > 30, Central Limit Theorem applies.
- **Small Samples:** n ≤ 30, use t-distribution.

35. 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).

Answer:

1. Null Hypothesis H0H_0H0: Population standard deviation σ=10.5\sigma = 10.5σ=10.5
2. Alternative Hypothesis HaH_aHa: Population standard deviation σ≠10.5\sigma \neq 10.5σ=10.5

$\chi^2 = \frac{(n - 1)s^2}{\sigma^2} = \frac{(25 - 1) \cdot 9^2}{10.5^2} = \frac{24 \cdot 81}{110.25} = 17.63$

Degrees of freedom $df = 24$.

Using chi-square tables, compare χ^2 value with critical values for $\alpha = 0.05$.

37. 100 students of a PW IOI obtained the following grades in Data Science paper. Grade: [A, B, C, D, E], Total Frequency: [15, 17, 30, 22, 16, 100]. Using the χ2 test, examine the hypothesis that the distribution of grades is uniform.

Answer:

1. Null Hypothesis H_0: Grades are uniformly distributed.
2. Expected frequency for each grade: $\frac{100}{5} = 20$

$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} = \frac{(15-20)^2}{20} + \frac{(17-20)^2}{20} + \frac{(30-20)^2}{20} + \frac{(22-20)^2}{20} + \frac{(16-20)^2}{20} = 2.5 + 0.45 + 5 + 0.2 + 0.8 = 8.95$

Degrees of freedom $df = 4$. Compare χ^2 value with critical value for $\alpha = 0.05$.

38. ANOVA Test

Answer: ANOVA (Analysis of Variance) is used to compare means of three or more samples to understand if at least one sample mean is significantly different. It examines the variance within groups and between groups.

Machine Learning

Q1: What is the difference between Series & Dataframes?

A1: A Series is a one-dimensional labeled array in pandas that can hold data of any type, while a DataFrame is a two-dimensional labeled data structure with columns of potentially different types. A DataFrame can be thought of as a collection of Series objects.

Q2: Create a database named Travel_Planner in MySQL, and create a table named bookings in it which having attributes (user_id INT, flight_id INT, hotel_id INT, activity_id INT, booking_date DATE). Fill with some dummy values. Now you have to read the content of this table using pandas as dataframe. Show the output.

A2: Here's how you can create the database, table, insert dummy values, and read it using pandas:

```
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 INTO bookings VALUES
(1, 101, 201, 301, '2024-07-01'),
(2, 102, 202, 302, '2024-07-02'),
(3, 103, 203, 303, '2024-07-03');
```

To read this table using pandas:

```
import pandas as pd
import mysql.connector

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

df = pd.read_sql("SELECT * FROM bookings", conn)
```

print(df)

This will output the contents of the bookings table as a pandas DataFrame.

Q3: Difference between loc and iloc?

A3: In pandas, loc is label-based indexing, while iloc is integer-based indexing. loc is used to access rows and columns by their labels, while iloc is used to access rows and columns by their integer positions.

Q4: What is the difference between supervised and unsupervised learning?

A4: Supervised learning uses labeled data to train models, where the algorithm learns to map input data to known output labels. Unsupervised learning works with unlabeled data, trying to find patterns or structures within the data without predefined outputs.

Q5: Explain the bias-variance tradeoff.

A5: The bias-variance tradeoff is a fundamental concept in machine learning. Bias refers to the error due to overly simplistic assumptions in the learning algorithm, while variance is the error due to the model's sensitivity to small fluctuations in the training set. High bias can cause underfitting, while high variance can lead to overfitting. The goal is to find the right balance between the two to create a model that generalizes well.

Q6: What are precision and recall? How are they different from accuracy?

A6: Precision is the ratio of true positives to all predicted positives, while recall is the ratio of true positives to all actual positives. Accuracy is the ratio of correct predictions to total predictions. Precision and recall are particularly useful for imbalanced datasets, where accuracy alone might be misleading.

Q7: What is overfitting and how can it be prevented?

A7: Overfitting occurs when a model learns the training data too well, including its noise and fluctuations, leading to poor generalization on new data. It can be prevented through techniques like cross-validation, regularization, increasing training data, feature selection, and using simpler models.

Q8: Explain the concept of cross-validation.

A8: Cross-validation is a technique used to assess how well a model will generalize to an independent dataset. It involves partitioning the data into subsets, training the model on a

subset, and validating it on the remaining data. This process is repeated multiple times with different partitions to reduce variability.

Q9: What is the difference between a classification and a regression problem?

A9: Classification problems involve predicting a discrete class label or category, while regression problems involve predicting a continuous numerical value. For example, predicting whether an email is spam or not is classification, while predicting house prices is regression.

Q10: Explain the concept of ensemble learning.

A10: Ensemble learning combines multiple models to create a more powerful predictive model. It works on the principle that a group of weak learners can come together to form a strong learner. Common ensemble methods include bagging (e.g., Random Forests), boosting (e.g., AdaBoost, XGBoost), and stacking.

Certainly. I'll provide answers to all the remaining questions:

Q11: What is gradient descent and how does it work?

A11: Gradient descent is an optimization algorithm used to minimize a function by iteratively moving in the direction of steepest descent. In machine learning, it's used to find the minimum of the cost function. The algorithm calculates the gradient of the cost function with respect to each parameter, then updates the parameters in the opposite direction of the gradient.

Q12: Describe the difference between batch gradient descent and stochastic gradient descent.

A12: Batch gradient descent uses the entire dataset to compute the gradient at each iteration, while stochastic gradient descent (SGD) uses a single random data point. Mini-batch gradient descent, a compromise between the two, uses a small random subset of data. SGD is faster and can escape local minima more easily, but it's noisier. Batch gradient descent is more stable but slower and can get stuck in local minima.

Q13: What is the curse of dimensionality in machine learning?

A13: The curse of dimensionality refers to various phenomena that arise when analyzing data in high-dimensional spaces that do not occur in low-dimensional settings. As the number of features increases, the amount of data needed to generalize accurately grows exponentially. This can lead to overfitting, increased computational complexity, and decreased model performance.

Q14: Explain the difference between L1 and L2 regularization.

A14: L1 (Lasso) and L2 (Ridge) regularization are techniques to prevent overfitting:
- L1 adds the absolute value of the coefficients to the loss function, leading to sparse models (feature selection).
- L2 adds the squared magnitude of coefficients, which doesn't lead to sparsity but prevents any feature from having a very large weight.

Q15: What is a confusion matrix and how is it used?

A15: A confusion matrix is a table used to evaluate the performance of a classification model. It shows the counts of true positives, true negatives, false positives, and false negatives. From this, various metrics like accuracy, precision, recall, and F1-score can be calculated.

Q16: Define AUC-ROC curve.

A16: The AUC-ROC (Area Under the Curve - Receiver Operating Characteristic) curve is a performance metric for binary classification problems. The ROC curve plots the True Positive Rate against the False Positive Rate at various threshold settings. The AUC represents the degree of separability, with 1 being perfect classification.

Q17: Explain the k-nearest neighbours algorithm.

A17: K-Nearest Neighbors (KNN) is a simple, instance-based learning algorithm. For classification, it assigns the majority class of the k nearest neighbors to a new data point. For regression, it averages the values of the k nearest neighbors. The choice of k and the distance metric are crucial parameters.

Q18: Explain the basic concept of a Support Vector Machine (SVM).

A18: SVM is a supervised learning algorithm used for classification and regression. It aims to find the hyperplane that best separates classes in a high-dimensional space. SVMs maximize the margin between this hyperplane and the nearest data points of any class, called support vectors.

Q19: How does the kernel trick work in SVM?

A19: The kernel trick allows SVMs to operate in a high-dimensional feature space without explicitly computing the coordinates of the data in that space. It uses kernel functions to compute the inner products between pairs of data points in the feature space, enabling non-linear classification.

Q20: What are the different types of kernels used in SVM and when would you use each?

A20: Common SVM kernels include:
- Linear: For linearly separable data
- Polynomial: For non-linear boundaries with more complex relationships
- Radial Basis Function (RBF): For non-linear boundaries with circular or elliptical decision regions
- Sigmoid: Similar to a neural network activation function

The choice depends on the data's characteristics and the problem at hand.

Q21: What is the hyperplane in SVM and how is it determined?

A21: In SVM, the hyperplane is the decision boundary that separates different classes. It's determined by maximizing the margin between the closest points (support vectors) of different classes. The optimal hyperplane is the one that provides the largest minimum distance to the training examples.

Q22: What are the pros and cons of using a Support Vector Machine (SVM)?
A22:
Pros:
- Effective in high-dimensional spaces
- Memory efficient
- Versatile through different kernel functions
Cons:
- Not suitable for large datasets due to high training time
- Sensitive to noisy data
- Doesn't directly provide probability estimates

Q23: Explain the difference between a hard margin and a soft margin SVM.

A23: Hard margin SVM assumes data is linearly separable and doesn't allow any misclassifications. Soft margin SVM allows some misclassifications, introducing a slack variable to handle non-linearly separable data. Soft margin is more practical for real-world, noisy datasets.

Q24: Describe the process of constructing a decision tree.

A24: Decision tree construction involves:
1. Selecting the best attribute to split the data
2. Splitting the data based on that attribute
3. Repeating the process for each child node
4. Stopping when a termination condition is met (e.g., max depth, min samples per leaf)

The best attribute is typically chosen using metrics like Gini impurity or information gain.

Q25: Describe the working principle of a decision tree.

A25: A decision tree works by recursively partitioning the data based on feature values. Starting from the root, it asks a series of questions about the features, following the appropriate branch based on the answers. This process continues until a leaf node is reached, which provides the prediction.

Q26: What is information gain and how is it used in decision trees?
A26: Information gain measures the reduction in entropy after a dataset is split on an attribute. It's used in decision trees to select the most informative feature for splitting at each node. The attribute with the highest information gain is chosen to split the data.

Q27: Explain Gini impurity and its role in decision trees.

A27: Gini impurity measures the probability of incorrectly classifying a randomly chosen element if it were randomly labeled according to the distribution of labels in the subset. In decision trees, it's used as a criterion to minimize the probability of misclassification. The attribute that results in the lowest Gini impurity is chosen for splitting.

Q28: What are the advantages and disadvantages of decision trees?

A28:
Advantages:
- Easy to understand and interpret
- Requires little data preparation
- Can handle both numerical and categorical data
Disadvantages:
- Can create overly complex trees that don't generalize well
- Can be unstable (small changes in data can result in a very different tree)
- Biased toward features with more levels (in classification)

Q29: How do random forests improve upon decision trees?
A29: Random forests improve upon decision trees by:
- Using ensemble learning (combining multiple trees)
- Implementing bagging (bootstrap aggregating)
- Introducing random feature selection at each node
These techniques reduce overfitting, improve generalization, and increase robustness to noise.

Q30: How does a random forest algorithm work?

A30: Random forest works by:
1. Creating multiple decision trees using bootstrap samples of the data
2. At each node, considering only a random subset of features for splitting
3. For classification, using majority voting of trees for the final prediction
4. For regression, averaging the predictions of all trees

Q31: What is bootstrapping in the context of random forests?

A31: Bootstrapping in random forests involves creating multiple subsets of the original dataset by randomly sampling with replacement. Each of these subsets (bootstrap samples) is used to train a different decision tree in the forest. This process introduces variability among the trees.

Q32: Explain the concept of feature importance in random forests.

A32: Feature importance in random forests measures how much each feature contributes to the predictions. It's typically calculated by measuring the average reduction in impurity (Gini impurity or entropy) across all trees in the forest that results from splits on that feature.

Q33: What are the key hyperparameters of a random forest and how do they affect the model?

A33: Key hyperparameters include:
- Number of trees: More trees generally improve performance but increase computation time
- Max depth: Controls the depth of trees, affecting model complexity
- Min samples split/leaf: Affects the minimum size of nodes, controlling overfitting
- Max features: Number of features to consider when looking for the best split

Q34: Describe the logistic regression model and its assumptions.

A34: Logistic regression is a statistical method for predicting a binary outcome. It models the probability of an outcome using the logistic function. Assumptions include:
- Binary or ordinal dependent variable
- Independence of observations
- Little or no multicollinearity among independent variables
- Linearity of independent variables and log odds

Q35: How does logistic regression handle binary classification problems?

A35: Logistic regression handles binary classification by estimating the probability that an instance belongs to a particular class. It uses the logistic function to transform the output of a linear equation into a value between 0 and 1, which can be interpreted as a probability. A threshold (usually 0.5) is then applied to make the final classification.

Q36: What is the sigmoid function and how is it used in logistic regression?

A36: The sigmoid function, also known as the logistic function, is an S-shaped curve that maps any real-valued number to a value between 0 and 1. In logistic regression, it's used to transform the linear combination of features into a probability value. The function is defined as f(x) = 1 / (1 + e^(-x)).

Q37: Explain the concept of the cost function in logistic regression.

A37: The cost function in logistic regression measures how well the model's predictions match the actual outcomes. It's typically the negative log-likelihood function. The goal is to minimize this cost function during training, usually using optimization algorithms like gradient descent.

Q38: How can logistic regression be extended to handle multiclass classification?
A38: Logistic regression can be extended to multiclass classification using techniques like:
- One-vs-Rest (OvR): Train a binary classifier for each class against all others
- One-vs-One (OvO): Train a binary classifier for every pair of classes
- Softmax Regression: A direct extension that uses the softmax function to output probabilities for each class

Q39: What is the difference between L1 and L2 regularization in logistic regression?
A39: In logistic regression:
- L1 regularization (Lasso) adds the absolute value of coefficients to the cost function, promoting sparsity (some coefficients become exactly zero)
- L2 regularization (Ridge) adds the squared magnitude of coefficients, which doesn't lead to sparsity but prevents any coefficient from becoming excessively large

Q40: What is XGBoost and how does it differ from other boosting algorithms?
A40: XGBoost (Extreme Gradient Boosting) is an optimized distributed gradient boosting library. It differs from other boosting algorithms by:
- Using a more regularized model formalization to control overfitting
- Implementing parallel processing
- Having built-in cross-validation and handling of missing values
- Using a unique sparsity-aware split-finding technique

Q41: Explain the concept of boosting in the context of ensemble learning.
A41: Boosting is an ensemble technique where weak learners are combined to create a strong learner. It works by training models sequentially, with each new model focusing on the errors of the previous ones. Each model is given a weight based on its performance, and the final prediction is a weighted combination of all models.

Q42: How does XGBoost handle missing values?

A42: XGBoost handles missing values by learning the best direction (left or right child) to go in the presence of missing values at each tree split. During training, it tries both directions for missing values and selects the one that leads to the best gain. This approach allows XGBoost to handle missing data without imputation.

Q43: What are the key hyperparameters in XGBoost and how do they affect model performance?
A43: Key XGBoost hyperparameters include:
- learning_rate: Controls the contribution of each tree
- max_depth: Maximum depth of trees
- n_estimators: Number of boosting rounds
- subsample: Fraction of samples used for training each tree
- colsample_bytree: Fraction of features used for training each tree
These parameters affect model complexity, training speed, and the balance between bias and variance.

Q44: Describe the process of gradient boosting in XGBoost.
A44: XGBoost's gradient boosting process involves:
1. Starting with an initial simple model
2. Calculating the residuals (errors) of this model
3. Training a new model to predict these residuals
4. Adding this new model to the ensemble, weighted by the learning rate
5. Repeating steps 2-4 for a specified number of iterations
Each iteration aims to correct the errors of the previous ensemble.

Q45: What are the advantages and disadvantages of using XGBoost?
A45:
Advantages:
- High performance and fast execution speed
- Good handling of missing values
- Built-in regularization and cross-validation
- Flexibility (can be used for regression, classification, and ranking)
Disadvantages:
- Can be prone to overfitting if not properly tuned
- Less interpretable than simpler models
- Requires careful hyperparameter tuning
- Computationally intensive for very large datasets