Q 1.1:

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

Ans:

|  |  |
| --- | --- |
| Static Variable | Dynamic Variable |
| 1. Scope: Static variables are associated with the class itself, not with any specific instance of the class. They are shared among all instances of the class.  2. Declaration: Declared within the class but outside any method, typically at the beginning of the class definition.  3. Access: Accessed using the class name or through any instance of the class.  4. Modification: Modifying a static variable affects all instances of the class.  5. Example:  class MyClass:  static\_var = 10   def \_\_init\_\_(self):  self.instance\_var = 5 | 1. Scope: Dynamic variables are associated with specific instances of a class. Each instance has its own copy of the variable.  2. Declaration: Declared within the constructor (\_\_init\_\_) method or any other instance method.  3. Access: Accessed using the instance of the class.  4. Modification: Modifying a dynamic variable affects only the specific instance where the modification is made.  5. Example:  class MyClass:  def \_\_init\_\_(self):  self.dynamic\_var = 10 |

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

Ans:

Pop(): Removes a key-value pair and returns the value of the key, as long as the key is specified. The syntax for pop() is dictionary.pop(key, default). If the key is present, the method returns the value associated with the deleted key-value pair. If the key is not present, the method returns the default value specified, or KeyError if no default value is specified. For example,

fruits = ['apple', 'banana', 'cherry']

print(fruits.pop(1))

print(fruits)

Output:

banana

['apple', 'cherry']

popitem(): Removes and returns the last key-value pair inserted into the dictionary as a tuple. The syntax for popitem() is dictionary.popitem(). For example,

car = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

print(car.popitem())

print(car)

Output:

('year', 1964)

{'brand': 'Ford', 'model': 'Mustang'}

clear(): Removes all items from the dictionary. The syntax for clear() is dict.clear(). The method doesn't take any parameters or return any value. For example,

d = {1: "sun", 2: "mon"};

d.clear()

Output:

d = {}

* *What do you mean by FrozenSet? Explain it with suitable example?*

Ans: In Python, a frozenset is an immutable version of a set. It is a built-in data type that represents an unordered collection of unique elements, just like a set, but once created, its elements cannot be added or removed.

Key characteristics of a frozenset:

Immutable: Elements cannot be added, removed, or modified after creation.

Unordered: Elements have no specific order.

Unique: Duplicate elements are not allowed.

Iterable: You can loop through the elements using a for loop.

Hashable: Can be used as a dictionary key or as an element of another set.

Example:

mylist = ['apple', 'banana', 'cherry']

x = frozenset(mylist)

print(x)

Output:

frozenset({'cherry', 'apple', 'banana'})

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

*immutable data types.*

Ans:

|  |  |  |
| --- | --- | --- |
|  | **Mutable datatypes** | **Immutable Datatypes** |
| Definition | Data type whose values can be changed after creation. | Data types whose values can’t be changed or altered. |
| Memory Location | Retains the same memory location even after the content is modified. | Any modification results in a new object and new memory location |
| Example | List, Dictionaries, Set | Strings, Types, Integer |
| Performance | It is memory-efficient, as no new objects are created for frequent changes. | It might be faster in some scenarios as there’s no need to track changes. |
| Thread-Safety | Not inherently thread-safe. Concurrent modification can lead to unpredictable results. | They are inherently thread-safe due to their unchangeable nature. |
| Use-cases | When you need to modify, add, or remove existing data frequently. | When you want to ensure data remains consistent and unaltered. |

* *What is \_\_init\_\_? Explain with an example.*

Ans: In Python, the \_\_init\_\_ method is a special method, also known as a constructor, that is automatically called when an object is created from a class. It is used to initialize the object's attributes with initial values.

Purpose: The \_\_init\_\_ method sets the initial state of an object by assigning values to its attributes.

Example:

class Person:  
 def \_\_init\_\_(self, name, age):  
 self.name = name  
 self.age = age  
 person1 = Person("Alice", 30)  
 print(person1.name)  
 print(person1.age)

Output:  
Alice  
30

* *Wat is docstring in Python? Explain with an example.*

Ans: In Python, a docstring is a string literal that appears as the first statement in a module, function, class, or method definition. It is used to document the purpose, usage, and behavior of that particular piece of code.

Purpose: Docstrings provide documentation that explains what the code does, making it easier for other developers (and yourself in the future) to understand and use it.

Format: Docstrings are enclosed in triple quotes (""") and placed immediately after the declaration of the module, function, class, or method.

Example:

def func():

    """My Function"""

    return 'Supriyo Sarkar'

print(func.\_\_doc\_\_)

Output:

My Function

* *What are unit tests in Python?*

Ans: Unit tests in Python are a way to test individual pieces of code (units) in isolation to ensure they are working correctly. These units are typically functions or methods. Unit testing helps catch bugs early in the development process, ensures that code changes don't break existing functionality, and promotes better code design.

Example:

import unittest

class TestAbsFunction(unittest.TestCase):

    def test\_positive\_number(self):

        self.assertEqual(abs(10), 10)

    def test\_negative\_number(self):

        self.assertEqual(abs(-10), 10)

    def test\_zero(self):

        self.assertEqual(abs(0), 0)

if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

Output:

E

======================================================================

ERROR: /root/ (unittest.loader.\_FailedTest)

----------------------------------------------------------------------

AttributeError: module '\_\_main\_\_' has no attribute '/root/'

----------------------------------------------------------------------

Ran 1 test in 0.012s

FAILED (errors=1)

An exception has occurred, use %tb to see the full traceback.

SystemExit: True

/usr/local/lib/python3.10/dist-packages/IPython/core/interactiveshell.py:3561: UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.

warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)

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

Ans:

Break: Terminates the loop entirely and continues with the next statement after the loop.

When you want to stop the loop prematurely if a certain condition is met.

Continue: Skips the rest of the current iteration and moves to the next iteration.

When you want to skip certain elements in the loop based on a condition.

Pass: A null operation that does nothing. It's a placeholder statement.

When a statement is syntactically required, but you don't want any code to execute.

Example:

lst = []

for i in range(10):

    if i == 5:

        break  # Exit the loop when i equals 5

    lst.append(i)

print(f"lst: {lst}")

lst1 = []

for i in range(10):

    if i % 2 == 0:

        continue  # Skip even numbers

    lst1.append(i)

print(f"lst1: {lst1}")

lst2 = []

for i in range(10):

    if i == 5:

        pass  # Do nothing when i equals 5

    else:

        lst2.append(i)

print(f"lst2: {lst2}")

Output:

lst: [0, 1, 2, 3, 4]

lst1: [1, 3, 5, 7, 9]

lst2: [0, 1, 2, 3, 4, 6, 7, 8, 9]

* *What is the use of self in Python?*

Ans: In Python, self is a reference to the instance of the class. It is used to access the attributes and methods of the class within its own methods.

Here's a breakdown of its use:

Accessing attributes: self allows you to access and modify instance variables (attributes) specific to each object created from the class.

Calling methods: self lets you call other methods defined within the same class.

Constructor (\_\_init\_\_) method: The \_\_init\_\_ method is a special method used to initialize the instance variables when an object is created. self is used here to bind the values passed as arguments to the corresponding instance variables.

Example:

class Details:

  def \_\_init\_\_(self, name, age):

    self.name = name

    self.age = age

  def abc(self):

    return f"Name: {self.name}, Age: {self.age}"

std = Details("Supriyo", 32)

print(std.abc())

Output:

Name: Supriyo, Age: 32

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

Ans: In Python, global, protected, and private attributes are used to control the visibility and accessibility of class attributes.

Global attributes:

Global Attributes are defined outside of a class and can be accessed from anywhere in the code, including inside classes.

Protected attributes:

Protected attributes are denoted with a single underscore (\_) before the attribute name and are intended to be accessed only within the class and its subclasses.

Private attributes:

Private attributes are denoted with double underscores (\_\_) before the attribute name and are intended to be accessed only within the class in which they are defined.

Example:

class tour\_details:

    def \_\_init\_\_(self, tourist, place, cost):

            self.tourist = tourist

            self.\_place = place

            self.\_\_cost = cost

    def get\_private\_data(self): # since cost is privated, it needs to be accessed through this method.

        print(f"cost to travel for {self.tourist} to {self.\_place} is {self.\_\_cost}")

class place\_cost(tour\_details):

    def getUnpublicData(self):

        print(f"place: {self.\_place}")

        print(f"cost: {self.\_\_cost}")

# error will occur in cost print as inheritence allows only public and protected data, not private data.

t1 = tour\_details("Sujoy", "Digha", 12500)

t1a = place\_cost("bittu", "manali", 45000)

t1.get\_private\_data() # this will be printed.

print(t1.tourist) # this will be printed.

print(t1.\_place) # this will be printed.

print(t1.\_\_cost) # this will occur error.

t1a.getUnpublicData()

Output:

cost to travel for Sujoy to Digha is 12500

Sujoy

Digha

**AttributeError** Traceback (most recent call last)

Cell **In[13],** [**line 4**](vscode-notebook-cell:?execution_count=13&line=4)

[2](vscode-notebook-cell:?execution_count=13&line=2) print(t1.tourist)

[3](vscode-notebook-cell:?execution_count=13&line=3) print(t1.\_place)

**---->** [**4**](vscode-notebook-cell:?execution_count=13&line=4) **print(t1.\_\_cost)**

**AttributeError**: 'tour\_details' object has no attribute '\_\_cost'

**---------------------------------------------------------------------------**

**AttributeError** Traceback (most recent call last)

Cell **In[14],** [**line 1**](vscode-notebook-cell:?execution_count=14&line=1)

**---->** [**1**](vscode-notebook-cell:?execution_count=14&line=1) **t1a.getUnpublicData()**

**Cell In[11],** [**line 12**](vscode-notebook-cell:?execution_count=11&line=12)

[10](vscode-notebook-cell:?execution_count=11&line=10) def getUnpublicData(self):

[11](vscode-notebook-cell:?execution_count=11&line=11) print(f"place: {self.\_place}")

**--->** [**12**](vscode-notebook-cell:?execution_count=11&line=12) **print(f"cost: {**self.\_\_cost}")

**AttributeError**: 'place\_cost' object has no attribute '\_place\_cost\_\_cost'

* *What are modules and packages in Python?*

Ans:

Modules:

Individual files that contain Python code, such as functions, classes, and variables. Modules have a .py extension and are executable files. They can be imported and used by other programs, and their contents can be accessed using dot notation. For example, you can import a module called demo\_module using the command import demo\_module. You can then call a function within that module using the command demo\_module.myModule("Math").

Packages:

Directories that contain multiple modules or subpackages. Packages provide a higher level of organization for code by grouping related modules, which can help create more structured and maintainable projects. Packages require an empty or initialization \_\_init\_\_.py file to be considered a package.

Example:

Module –

math: Provides mathematical functions like sqrt, sin, cos, pi, etc.

datetime: For working with dates and times.

os: Provides functions for interacting with the operating system, like creating files, directories.

Packages –

NumPy: Provides support for arrays and matrices, along with functions for numerical operations, linear algebra, and random number generation. Essential for scientific and engineering computations.

Pandas: Offers flexible data structures like DataFrames, making data manipulation and analysis easy and intuitive. Widely used for data science and machine learning tasks.

Matplotlib: A powerful library for creating visualizations, ranging from simple plots to complex figures. Helps in understanding data patterns and presenting insights.

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

Ans: In Python, lists and tuples are both built-in data structures that store and manipulate collections of data. They are both ordered and sequential data types that can store any kind of data type.

Example:

a = [4, 8, 5]

b = (9, 7, 5)

print(type(a), type(b))

Output:

<class 'list'> <class 'tuple'>

Key differences:

Mutability-

Lists are mutable, meaning their elements can be modified after creation, while tuples are immutable, meaning their elements cannot be changed once they are created.

Use cases-

Lists are best suited for situations where data needs to be modified frequently or where the order of the elements matters, while tuples are best suited for situations where the data needs to be accessed frequently but not modified, and the size of the collection is fixed and known in advance.

Memory efficiency-

Tuples are more memory efficient than lists.

Functionality-

Tuples have limited functionality compared to lists. For example, tuples do not have the append() method for adding elements, which is a list-specific operation.

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

Ans:

Interpreter language: An interpreted language is a programming language that executes instructions directly during runtime without first compiling the program into machine language. In an interpreted language, the source code is read and executed line by line by an interpreter, rather than being translated into machine code beforehand by the target machine.

Dynamically typed language: Dynamically typed languages are programming languages that assign variables a type at runtime based on the variable's value at that time. This means that type checking happens during execution time, or when the program is running. In these languages, programmers don't need to specify the type of variable a function is accepting or returning.

Difference between interpreter and dynamically typed language:

An interpreter is a computer program that executes instructions in a programming language, while dynamically typed languages are programming languages that use interpreters to assign variable types at runtime.

* *What are Dict and List comprehensions?*

Ans: List comprehensions: Create new lists by applying an expression to each item in an iterable (like a list, tuple, or string) and filtering the items based on a condition. For example, x = [i for i in range(10)] or x = [i for i in range(10) if i > 5]. List comprehensions are more compact and readable than for loops, and they can use functions and complex expressions.

Dictionary comprehensions: Create new dictionaries by specifying key-value pairs using expressions. For example, square = {num: num \*\*2} for num in a}. Dictionary comprehensions need two expressions separated with a colon, followed by the usual “for” and “if” clauses. The resulting key and value elements are inserted in the new dictionary in the order they are produced.

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

Ans: In Python, decorators are a powerful and versatile tool that allow you to modify the behavior of functions or classes without directly changing their source code. They essentially wrap around the original function, adding extra functionality before or after the function is called.

Example:

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:

Decorators can be used to add logging functionality to functions. This is useful for debugging and tracking the execution flow of your program.

Timing:

Decorators can measure the execution time of a function. This can help you identify performance bottlenecks in your code.

Caching:

Decorators can cache the results of a function, so that it doesn't need to be recomputed every time it's called. This can significantly improve the performance of your application.

Authentication and Authorization:

Decorators can be used to check if a user has the necessary permissions to access a certain resource.

Input Validation:

Decorators can be used to validate the input arguments to a function. This can help you prevent errors and improve the robustness of your code.

* *How is memory managed in Python?*

Ans: Python manages memory automatically, freeing developers from the complexities of manual memory allocation and deallocation. It achieves this primarily through two key mechanisms:

1. Reference Counting:

Every object in Python has a reference count, which keeps track of how many variables or data structures are pointing to it.

When an object's reference count reaches zero, meaning nothing is referencing it anymore, Python's garbage collector automatically deallocates the memory occupied by that object.

2. Garbage Collection:

Python's garbage collector runs periodically to detect and reclaim memory from objects that are no longer reachable.

It uses a generational garbage collection algorithm, which divides objects into generations based on their age. Younger generations are collected more frequently than older ones, as they tend to have a higher turnover rate.

Circular references, where two or more objects reference each other, creating a cycle, can pose a challenge for reference counting. Python's garbage collector employs a cycle detection algorithm to identify and collect these circular references.

Other Memory Management Techniques in Python:

Memory Pooling:

Python uses memory pools to allocate memory efficiently for small objects. This avoids the overhead of requesting memory from the operating system for every object creation.

Generators and Iterators:

These constructs allow you to process data one element at a time, instead of loading the entire dataset into memory. This is especially useful for large datasets.

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

Ans: In Python, a lambda function is an anonymous function that can be defined in a single line using the lambda keyword instead of def. Lambda functions are useful for short-term functions or when a function needs to be used only once. They can accept any number of arguments but can only have one expression.  
Here's what makes lambda functions useful:

Simple expressions: They're efficient for creating functions that contain simple expressions, typically a single line of code.

Passing as arguments: They're most useful when used as an argument to another function.

Processing objects: Lambda functions can process objects in a dictionary using functional programming constructs like map, reduce, and filter. This can help write clean and maintainable code.

Example:

lst = [33, 3, 22, 2, 11, 1]

sorted(filter(lambda x: x > 10, lst))

Output:

[11, 22, 33]

* *Explain split() and join() functions in Python.*

Ans:

split()

The split() method breaks a string into a list of substrings based on a specified delimiter. If no delimiter is specified, it defaults to whitespace (spaces, tabs, newlines).

join()

The join() method is the opposite of split(). It takes an iterable (like a list or tuple) and joins its elements into a single string, using a specified separator.

Example:

a = "Apple, Banana, Grapes"

b = a.split(',')

print(b)

c = ",".join(i for i in b)

print(c)

Output:

['Apple', ' Banana', ' Grapes']

Apple, Banana, Grapes

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

Ans: Iterable:

An object that can be iterated upon, meaning you can loop through its elements. Examples include lists, tuples, dictionaries, sets, and strings. They implement the \_\_iter\_\_() method, which returns an iterator object.

Iterator:

An object that represents a stream of data. It implements the \_\_next\_\_() method, which returns the next value in the sequence. When there are no more values, it raises a StopIteration exception.

Generator:

A special type of function that generates a sequence of values on-the-fly, rather than returning them all at once. It uses the yield keyword to return each value. Generators are also iterators, meaning you can use them in for loops and with the next() function.

Example:

# Iterable

my\_list = [1, 2, 3]

# Iterator

my\_iter = iter(my\_list)

print(next(my\_iter))

print(next(my\_iter))

# Generator

def my\_gen(n):

for i in range(n):

yield i

gen = my\_gen(3)

print(next(gen))

print(next(gen))

Output:

1  
2  
0  
1

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

Ans:

|  |  |  |
| --- | --- | --- |
|  | Range() | Xrange() |
|  |
| **Return type** | It gives a list of integers | It gives a generator object |
| **Memory Utilization** | It gives a list of elements and takes more memory. | It takes less memory less range () |
| **Efficiency** | Its execution time is slow. | Its execution time is fast. |

* *Pillars o Oops.*

Ans:

Encapsulation

Wraps up everything about a particular thing into a defined object with features and behaviors. For example, if you want to add a zoom-in feature to an open camera function, you can update the function without worrying about anything outside of it.

Inheritance

Defines a class for something and then lets other classes inherit common features from it. For example, you can define a class for accounts and then let checking and savings accounts inherit common features.

Polymorphism

Treats the same object as different things depending on how it's needed at different times.

* *How will you check if a class is a child o another class?*

Ans: In Python, you can use the built-in function issubclass() to determine if a class is a subclass of another. It returns True if the class is a subclass, and False if it isn't. For example, issubclass(bool, int) returns True because bool is a subclass of int.

You can also use the built-in function isinstance() to check if an object or instance is a subclass of a specified class.

In inheritance, a child class gets its properties from its parent class. Child classes inherit their parent class's methods, so they can use those methods in programs.

Example:

print("Float is the subclass of str:", issubclass(float,str))

print("Bool is the subclass of int:", issubclass(bool,int))

print("int is the subclass of float:",issubclass(int,float))

import collections

print('collections.defaultdict is the subclass of dict: ', issubclass(collections.defaultdict, dict))

Output:

Float is the subclass of str: False

Bool is the subclass of int: True

int is the subclass of float: False

collections.defaultdict is the subclass of dict: True

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

Ans: Inheritance is a fundamental concept in object-oriented programming (OOP) that allows you to create a new class (child class) based on an existing class (parent class). The child class inherits all the attributes and methods of the parent class, enabling code reusability and creating a hierarchical relationship between classes.

Single Inheritance:

class Animal:

def speak(self):

print("Animal speaks")

class Dog(Animal):

def bark(self):

print("Dog barks")

d = Dog()

d.speak() # Output: Animal speaks

d.bark() # Output: Dog barks

Multiple Inheritance:

class Flyer:

def fly(self):

print("Flying")

class Swimmer:

def swim(self):

print("Swimming")

class Duck(Flyer, Swimmer):

pass

d = Duck()

d.fly() # Output: Flying

d.swim() # Output: Swimming

Multilevel Inheritance:

class Vehicle:

def start(self):

print("Vehicle started")

class Car(Vehicle):

def drive(self):

print("Car driving")

class SportsCar(Car):

def turbo(self):

print("Turbo boost!")

sc = SportsCar()

sc.start() # Output: Vehicle started

sc.drive() # Output: Car driving

sc.turbo() # Output: Turbo boost!

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

Ans: Encapsulation is a software technique that bundles data with the methods that operate on it, and may also limit direct access to some of that data. It's a way to prevent external code from being concerned with an object's internal workings.

Hiding data

In a class, encapsulation can be used to hide the values or state of a structured data object. For example, in a class called "Employee", you might use encapsulation to specify the access of data members like "Name" and "Age".

Organizing code

Encapsulation encourages programmers to put all the code that's concerned with a certain set of data in the same class, which can make it easier for other programmers to understand.

Information hiding

Encapsulation can also be called "information hiding" because it conceals the details of an object's internal processes. For example, in a car, the engine encapsulates its internal workings, so you interact with it through the car's controls without accessing the engine directly.

Encapsulation is supported by all object-oriented programming (OOP) systems, but it's not unique to OOP. Implementations of abstract data types, modules, and libraries also offer encapsulation.

Example:

class Student:

   def \_\_init\_\_(self, name, marks):

        self.\_\_name = name

        self.\_\_marks = marks

   def studentdata(self): # method defined to retrive encapsulated data passively.

        print (f"Name: {self.\_\_name} marks: {self.\_\_marks}")

s1 = Student("Supriyo", 50)

s2 = Student("Arijit", 65)

s1.studentdata()

s2.studentdata()

print (f"Name: {s1.\_\_name} marks: {s1.\_\_marks}") # s1 will not find the \_\_name and \_\_marks attribute due to encapsulation.

print (f"Name: {s2.\_\_name} marks: {s2.\_\_marks}") # s2 will not find the \_\_name and \_\_marks attribute due to encapsulation.

Output:

Name: Supriyo marks: 50

Name: Arijit marks: 65

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

Ans: The word polymorphism means having many forms. In programming, polymorphism means the same function name (but different signatures) being used for different types. The key difference is the data types and number of arguments used in function.

Example:

class Student:

    def \_\_init\_\_(self, name, surname, roll):

        self.name = name

        self.surname = surname

        self.roll = roll

    def display\_name(self):

        print(self.name)

    def display\_surname(self):

        print(self.surname)

class Teacher:

    def \_\_init\_\_(self, name, surname, subject):

        self.name = name

        self.surname = surname

        self.subject = subject

    def display\_name(self):

        print(self.name)

    def display\_surname(self):

        print(self.surname)

class Place:

    def \_\_init\_\_(self, name, country):

        self.name = name

        self.country = country

    def display\_name(self):

        print(self.name)

    def display\_country(self):

        print(self.country)

stud1 = Student("Hirak", "Das", 21)

teacher1 = Teacher("Sandipan", "Banerjee", "Chemistrty")

place1 = Place("Birati", "India")

cl\_list = [stud1, teacher1, place1]

for n in cl\_list:

    n.display\_name()

Output:

Hirak

Sandipan

Birati

*Q 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*

Ans: Serial\_no. – the dot operator (.) is used to access attributes and methods of an object. Cannot be used in identifier name.

Hundred$ - Special character cannot be used in identifier name.

total-Marks - ‘\_’ is allowed in identifier name, not ‘-‘.

Total Marks - spacebar splits identifier name into two. Cannot be used as a single one.

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

*calculated.*

Ans: Measures of Central Tendency:

Central Tendencies in Statistics are the numerical values that are used to represent mid-value or central value a large collection of numerical data. These obtained numerical values are called central or average valuesin [Statistics](https://www.geeksforgeeks.org/statistics/). A centralor average value of any statistical data or series is the value of that variable that is representative of the entire data or its associated frequency distribution. Such a value is of great significance because it depicts the nature or characteristics of the entire data, which is otherwise very difficult to observe.

The representative value of a data set, generally the central value or the most occurring value that gives a general idea of the whole data set is called Measure of Central Tendency. Some of the most commonly used measures of central tendency are:

Mean, Median & Mode.

Mean (x̅ or μ)**:** The mean, or arithmetic average, is calculated by summing all the values in a dataset and dividing by the total number of values. It’s sensitive to outliers and is commonly used when the data is symmetrically distributed.

Median (M): The median is the middle value when the dataset is arranged in ascending or descending order. If there’s an even number of values, it’s the average of the two middle values. The median is robust to outliers and is often used when the data is skewed.

Mode (Z)**:** The mode is the value that occurs most frequently in the dataset. Unlike the mean and median, the mode can be applied to both numerical and categorical data. It’s useful for identifying the most common value in a dataset.

Measures of Dispersion:

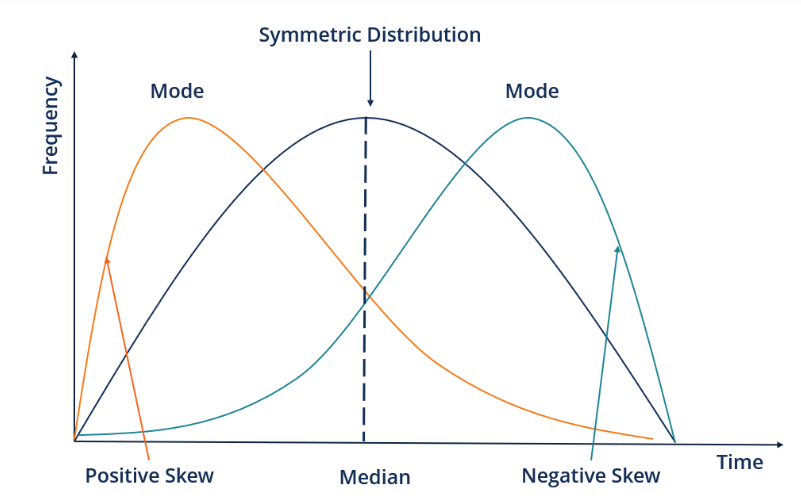
Measures of dispersion in statistics describe the spread of data and how likely numerical data is to vary around an average value. A good measure of dispersion should be easy to calculate and understand, based on all observations, and not affected by extreme values or sampling fluctuations. Here are some measures of dispersion:

* Standard deviation: Estimates how far data points are from the mean value. It's the square root of the mean of the squared deviation of all values in a series from the arithmetic mean.
* Mean deviation: Calculates the average absolute deviation of values in a dataset from their mean. Deviations can be positive or negative, and are calculated as the absolute difference between the value and the mean.
* Range: The difference between the highest and lowest data values.
* Variance: The mean squared difference between each data point and the mean.
* Coefficient of variation: A dimensionless variable that's usually expressed as a proportion. It helps compare two data sets based on their degree of variance.
* Quartile deviation: A statistic that measures dispersion.
* Interquartile range (IQR): The range between the first and third quartiles in a dataset.

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

Ans: Skewness is a measure of asymmetry or distortion of symmetric distribution. It measures the deviation of the given distribution of a [random variable](https://corporatefinanceinstitute.com/resources/data-science/random-variable/) from a symmetric distribution, such as normal distribution. A normal distribution is without any skewness, as it is symmetrical on both sides. Hence, a curve is regarded as skewed if it is shifted towards the right or the left.

* Skewness measures the deviation of a random variable’s given distribution from the normal distribution, which is symmetrical on both sides.
* A given distribution can be either be skewed to the left or the right. Skewness risk occurs when a symmetric distribution is applied to the skewed data.
* Investors take note of skewness while assessing investments’ return distribution since extreme data points are also considered.



**Types of Skewness**

1. Positive Skewness

If the given distribution is shifted to the left and with its tail on the right side, it is a positively skewed distribution. It is also called the right-skewed distribution. A tail is referred to as the tapering of the curve differently from the data points on the other side.

As the name suggests, a positively skewed distribution assumes a skewness value of more than zero. Since the skewness of the given distribution is on the right, the mean value is greater than the [median](https://corporatefinanceinstitute.com/resources/?topics=111064) and moves towards the right, and the mode occurs at the highest frequency of the distribution.

2. Negative Skewness

If the given distribution is shifted to the right and with its tail on the left side, it is a negatively skewed distribution. It is also called a left-skewed distribution. The skewness value of any distribution showing a negative skew is always less than zero. The skewness of the given distribution is on the left; hence, the mean value is less than the median and moves towards the left, and the [mode](https://corporatefinanceinstitute.com/resources/data-science/mode/) occurs at the highest frequency of the distribution.

**Formula**: Skewness = 3(mean-median)/standard deviation.

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

Ans: PROBABILITY MASS FUNCTION (PMF)

The Probability Mass function is defined on all the values of R, where it takes all the arguments of any [real number](https://byjus.com/maths/real-numbers/). It doesn’t belong to the value of X when the argument value equals to zero and when the argument belongs to x, the value of PMF should be positive.

The Probability Mass Function (PMF) is also called a probability function or frequencyfunction which characterizes the distribution of a discrete random variable. Let X be a discrete random variable of a function, then the probability mass function of a [random variable](https://byjus.com/maths/random-variable/) X is given by

Px (x) = P( X=x ), For all x belongs to the range of X

It is noted that the probability function should fall on the condition:

* Px(x) ≥ 0 and
* ∑xϵRange(x) Px(x) = 1

Here the Range(X) is a countable set and it can be written as { x1, x2, x3, ….}. This means that the random variable X takes the value x1, x2, x3, ….

These can also be stated as explained below.

The probability mass function P(X = x) = f(x) of a discrete random variable is a function that satisfies the following properties:

* P(X = x) = f(x) > 0; if x ∈ Range of x that supports
* ∑𝑥𝜖𝑅𝑎𝑛𝑔𝑒 𝑜𝑓𝑥𝑓(𝑥)=1
* 𝑃(𝑋𝜖𝐴)=∑𝑥𝜖𝐴𝑓(𝑥)

PROBABILITY DENSITY FUNCTION (PDF)

The Probability Density Function(PDF) defines the probability function representing the density of a continuous random variable lying between a specific range of values. In other words, the probability density function produces the likelihood of values of the continuous random variable. Sometimes it is also called a [probability distribution](https://byjus.com/maths/probability-distribution/) function or just a probability function. However, this function is stated in many other sources as the function over a broad set of values. Often it is referred to as cumulative distribution function or sometimes as [probability mass function](https://byjus.com/maths/probability-mass-function/)(PMF). However, the actual truth is PDF (probability density function) is defined for continuous random variables, whereas PMF (probability mass function) is defined for discrete random variables.

In the case of a continuous random variable, the probability taken by X on some given value x is always 0. In this case, if we find P(X = x), it does not work. Instead of this, we must calculate the probability of X lying in an interval (a, b). Now, we have to figure it for P(a< X< b), and we can calculate this using the formula of PDF. The Probability density function formula is given as,

𝑃(𝑎<𝑋<𝑏)=∫𝑎𝑏𝑓(𝑥) dx

Or

𝑃(𝑎≤𝑋≤𝑏)=∫𝑎𝑏𝑓(𝑥) dx

This is because, when X is continuous, we can ignore  the endpoints of intervals while finding probabilities of continuous random variables. That means, for any constants a and b,

P(a ≤ X ≤ b) = P(a < X ≤ b) = P(a ≤ X < b) = P(a < X < b).

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

Ans: Correlation refers to a process for establishing the relationships between two variables. You learned a way to get a general idea about whether or not two variables are related, is to plot them on a “[scatter plot](https://byjus.com/maths/scatter-plot/)”. While there are many measures of association for variables which are measured at the ordinal or higher level of measurement, correlation is the most commonly used approach.

In statistics, Correlation studies and measures the direction and extent of relationship among variables, so the correlation measures co-variation, not causation. Therefore, we should never interpret correlation as implying cause and effect relation. For example, there exists a correlation between two variables X and Y, which means the value of one variable is found to change in one direction, the value of the other variable is found to change either in the same direction (i.e. positive change) or in the opposite direction (i.e. negative change). Furthermore, if the correlation exists, it is linear, i.e. we can represent the relative movement of the two variables by drawing a straight line on graph paper.

Correlation Coefficient

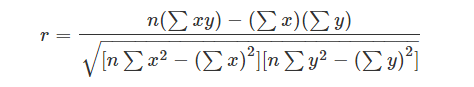
The correlation coefficient, r, is a summary measure that describes the extent of the statistical relationship between two interval or ratio level variables. The correlation coefficient is scaled so that it is always between -1 and +1. When r is close to 0 this means that there is little relationship between the variables and the farther away from 0 r is, in either the positive or negative direction, the greater the relationship between the two variables.

Types of Correlation:

* Positive Correlation – when the values of the two variables move in the same direction so that an increase/decrease in the value of one variable is followed by an increase/decrease in the value of the other variable.
* Negative Correlation – when the values of the two variables move in the opposite direction so that an increase/decrease in the value of one variable is followed by decrease/increase in the value of the other variable.
* No Correlation – when there is no linear dependence or no relation between the two variables.

Pearson Correlation Coefficient Formula

The most common formula is the Pearson Correlation coefficient used for linear dependency between the data sets. The value of the coefficient lies between -1 to +1. When the coefficient comes down to zero, then the data is considered as not related. While, if we get the value of +1, then the data are positively correlated, and -1 has a negative correlation.



Where n = Quantity of Information

Σx = Total of the First Variable Value

Σy = Total of the Second Variable Value

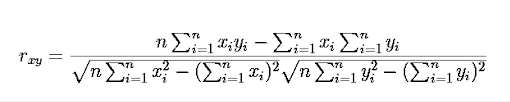
Σxy = Sum of the Product of first & Second Value

Σx2 = Sum of the Squares of the First Value

Σy2 = Sum of the Squares of the Second Value

Linear Correlation Coefficient Formula

The formula for the linear correlation coefficient is given by;



Sample Correlation Coefficient Formula

The formula is given by:

rxy = Sxy/SxSy

Where Sx and Sy are the sample standard deviations, and Sxy is the sample covariance.

Population Correlation Coefficient Formula

The population correlation coefficient uses σx and σy as the population standard deviations and σxy as the population covariance.

rxy = σxy/σxσy.

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

Ans:

|  |  |  |
| --- | --- | --- |
| Basis for Comparison | Correlation | Regression |
| Meaning | Correlation is the statistical measure that determines the association or co-relationship between two variables. | Regression describes how to numerically relate an independent variable to dependant variable. |
| Usage | To represent a linear relationship between two variables. | To fit the best line and to estimate one variable based on another. |
| Objective | To find a numerical value expressing the relationship between variables. | To estimate values of random variables  on the basis of values of fixed variables. |
| Indicates | Correlation coefficients extent to which two variable move together. | Regression indicates the impact of a change of unit on the estimated variable  In the known variable. |

*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.*

Ans:

The coefficient of correlation between the prices of the two places is +0.8 

Mean price at Delhi=Rs.65Mean price at Delhi equals Rs. 65

Mean price at Delhi=Rs.65 

Mean price at Agra=Rs.67Mean price at Agra equals Rs. 67

Mean price at Agra=Rs.67 

Standard deviation of prices at Delhi=Rs.2.5Standard deviation of prices at Delhi equals Rs. 2.5

Standard deviation of prices at Delhi=Rs.2.5.

y− y¯=r (σy/σx) (x−x¯)

𝑦−𝑦=𝑟(𝜎𝑦/𝜎𝑥)(𝑥−𝑥)

y−67=0.8⋅3.52.5(x−65)y

𝑦−67=0.8⋅3.52.5(𝑥−65)

y−67=1.12(x−65)y



70=1.12x−6.870

70=1.12𝑥−6.8

1.12x=76.81.

1.12𝑥=76.8

x=76.8 / 1.12

x≈68.57

The most likely price at Delhi corresponding to the price of Rs. 68.57.

*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.*

Ans: var x = 9

Std x =

8x – 10y = -66

10 y = 8x + 66

Y = 0.8x + 6.6

Coreg x = 0.8

40x – 18y = 214

40x = 18y + 214

X = 0.45y + 5.35

Coreg y = 0.45

Corr coef (x, y) = sqrt(0.8 x 0.45)

= sqrt(0.36)

= 0.6

Solving regression equations, we have mean(y) = 17, mean(x) = 13

Coreg y = Corr coef (x, y) \* (std x / std y)

0.45 = 0.6 \* (3 / std y)

Std y = (0.6 \* 3)/ 0.45

= 4

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

Ans:

Normal distribution, also known as the Gaussian distribution, is a [probability distribution](https://www.investopedia.com/terms/p/probabilitydistribution.asp) that is symmetric about the mean, showing that data near the mean are more frequent in occurrence than data far from the mean. The normal distribution appears as a "[bell curve](https://www.investopedia.com/terms/b/bell-curve.asp)" when graphed.

The normal distribution is the most common type of distribution assumed in technical stock market analysis. The standard [normal distribution has two parameters](https://www.investopedia.com/articles/active-trading/092914/normal-distribution-table-explained.asp): the mean and the standard deviation. In a normal distribution, [mean](https://www.investopedia.com/terms/m/mean.asp) (average), [median](https://www.investopedia.com/terms/m/median.asp) (midpoint), and [mode](https://www.investopedia.com/terms/m/mode.asp) (most frequent observation) are equal. These values represent the peak or highest point. The distribution then falls symmetrically around the mean, the width of which is defined by the [standard deviation](https://www.investopedia.com/terms/s/standarddeviation.asp).

The normal distribution model is key to the [Central Limit Theorem](https://www.investopedia.com/terms/c/central_limit_theorem.asp) (CLT) which states that averages calculated from independent, identically distributed random variables have approximately normal distributions, regardless of the type of distribution from which the variables are sampled.1

Boston University. "[The Central Limit Theorem](https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_probability/BS704_Probability12.html)."

The normal distribution is one type of [symmetrical distribution](https://www.investopedia.com/terms/s/symmetrical-distribution.asp). Symmetrical distributions occur when a dividing line produces two mirror images. Not all symmetrical distributions are normal since some data could appear as two humps or a series of hills in addition to the bell curve that indicates a normal distribution.

A normal distribution is a continuous probability distribution that's symmetric, unimodal, and asymptotic, with an equal mean, median, and mode. It can be described by four moments: mean, standard deviation, skewness, and kurtosis. The statistical properties of normal distributions are important for parametric statistical tests that rely on assumptions of normality. In a normal distribution, the mean is zero and the standard deviation is 1. It has zero skew and a kurtosis of 3. Normal distributions are symmetrical, but not all symmetrical distributions are normal.

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

Ans: The normal distribution curve has many properties, including:

* Bell shape: The curve is always bell-shaped
* Symmetry: The curve is symmetric around the mean
* Unimodality: The curve has a single peak value
* Measures of central tendency: The mean, median, and mode are all equal
* Probability density function: The curve provides a probability that a random variable falls within a certain range, in this case, a deviation from the mean
* Standard deviation: The curve includes a standard deviation parameter, which measures how spread out the curve is. Approximately 68% of outcomes are within one standard deviation of the mean, 95% are within two standard deviations, and 99.7% are within three standard deviations
* Empirical rule: Also known as the 68-95-99.7 rule, this property can be used to approximate relative frequencies and probabilities for certain intervals of the curve
* Central limit theorem: This property states that under certain conditions, the distribution of a sum of many independent variables is approximately normal
* Skewness: The curve has no skewness because it is symmetrical on both sides.

*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.

Ans: All options are correct about Normal Distribution Curve.

*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?*

Ans:

X = 72

Mean(x) = 60

Std(x) = 10

Z = (72 – 60)/10

= 1.2

Percentile = 0.8849

i.e. (88.49 – 50)% = 38.49 % items will be between 60 and 72

X = 50

Mean(x) = 60

Std(x) = 10

Z = (50 – 60)/10

= -1

Percentile = 0.1586

i.e. (50 – 15.86)% = 34.14 % items will be between 50 and 60

data beyond 72 = (100- 88.49)% = 11.51 %

Z70 = 1; percentile = 84.13%

Z80 = 2; percentile = 97.72%

Items between 70 and 80 = 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.

Ans:

Mean(m) = 49

Std(m) = 6

Z55 = 1, percentile = 84.13%

Therefore, students scored beyond 55 = 15000 x (100 – 84.13) % = 2380

Z70 = 3, percentile = 99.86 %

Therefore, students scored beyond 55 = 15000 x (100 – 99.86) % = 21

*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.*

Ans:

Mean(h) = 65

Std(h) = 5

Z70 = (70 - 65) / 5 = 1; percentile = 84.13 %

Therefore heights beyond 70 inch = 500 x (100 – 84.13) % = 79

Z60 = (60 - 65) / 5 = -1; percentile = 15.86 %

Students between 60 to 70 inch = 500 x (84.13 – 15.86) % = 341

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

Ans:

A statistical hypothesis test is a method of statistical inference used to decide whether the data sufficiently support a particular hypothesis. A statistical hypothesis test typically involves a calculation of a [test statistic](https://en.wikipedia.org/wiki/Test_statistic). Then a decision is made, either by comparing the test statistic to a [critical value](https://en.wikipedia.org/wiki/Critical_value_(statistics)) or equivalently by evaluating a [*p*-value](https://en.wikipedia.org/wiki/P-value) computed from the test statistic. Roughly 100 [specialized statistical tests](https://en.wikipedia.org/wiki/List_of_statistical_tests) have been defined.

Errors in Hypothesis testing:

Errors in hypothesis testing can occur when a researcher rejects a true null hypothesis or fails to reject a false one. These errors are called Type I and Type II errors, respectively:

* Type I error

Occurs when a researcher wrongly rejects the null hypothesis, thinking they found a significant effect when there isn't one. This is also known as a false positive. The probability of making a Type I error is represented by alpha, α.

* Type II error

Occurs when a researcher wrongly fails to reject the null hypothesis, missing a significant effect that is really there. This is also known as a false negative. The probability of making a Type II error is represented by beta, β.

Sample:

A sample is a representative part of a larger group, or a finite part of a statistical population that's studied to learn about the whole. In research, the sample size is the number of observations used to estimate a population. Sample size is important because it ensures the sample represents the population.

Large samples

* More accurately represent the population
* Provide more accurate results
* Can make small differences statistically significant, even if clinically insignificant
* Can be unethical
* Can make it difficult to interpret significance tests

Small samples

* Can have insufficient statistical power to answer the primary research question
* Can undermine the validity of a study
* Can be important for reaching valid conclusions in certain situations.

The appropriate sample size depends on the context. Large samples are only necessary for studies with highly variable outcomes, or when a small effect size needs to be detected. However, very large samples can mislead researchers and clinicians, which can lead to poor treatment decisions.

Some say that a sample size of 30–60 is considered small. Others say that a sample size of less than 5,000, or if the sample size is 20% or more of the population, is considered small.

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

Ans:

1. Import Flask:

Import the Flask class from the flask module.

2. Create an app instance:

Create an instance of the Flask class. The \_\_name\_\_ argument is a special variable that refers to the name of the current module.

3. Define a route:

Use the @app.route() decorator to define a route for the homepage (/). The function below the decorator will be executed when the homepage is accessed.

4. Return a response:

The index() function returns the string "Hello, World!", which will be displayed on the homepage.

5. Run the app:

The if \_\_name\_\_ == "\_\_main\_\_": block ensures that the app is only run when the script is executed directly. The app.run(debug=True) starts the Flask development server in debug mode, which allows for automatic reloading and error reporting.

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

Ans:

@app.route("/", methods=["GET", "POST"]): This decorator defines a route for the root URL (/) that accepts both GET and POST requests.

if request.method == "POST":: This checks if the request method is POST, indicating a form submission.

request.form.get("name"): This extracts the value of the "name" field from the form data.

render\_template("form.html"): This renders the form template, which is an HTML file containing the form.

app.run(debug=True): This starts the Flask development server with debugging enabled.

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

Ans:

1. Install Flask-Login

pip install Flask-Login

2. Define a User model to store user information, typically including username, password hash, and potentially other details.

from flask\_login import UserMixin

from werkzeug.security import generate\_password\_hash, check\_password\_hash

class User(UserMixin):

def \_\_init\_\_(self, id, username, password):

self.id = id

self.username = username

self.password = password

def check\_password(self, password):

return check\_password\_hash(self.password, password)

3. Configure Flask-Login:

Initialize the LoginManager and define a user loader function to retrieve a user object based on their ID.

from flask import Flask

from flask\_login import LoginManager

app = Flask(\_\_name\_\_)

login\_manager = LoginManager()

login\_manager.init\_app(app)

@login\_manager.user\_loader

def load\_user(user\_id):

# Load user from database based on user\_id

return User.get(user\_id)

4. Implement Login and Registration Routes:

Create routes for handling user login and registration, using forms to collect user credentials.

from flask import render\_template, request, redirect, url\_for

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

user = User.get\_by\_username(username)

if user and user.check\_password(password):

login\_user(user)

return redirect(url\_for('index'))

return render\_template('login.html')

@app.route('/register', methods=['GET', 'POST'])

def register():

# Handle registration form submission

5. Use the @login\_required decorator to restrict access to certain routes to authenticated users.

from flask\_login import login\_required

@app.route('/profile')

@login\_required

def profile():

return render\_template('profile.html')

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

Ans:

Install necessary packages:

pip install Flask SQLAlchemy

Create a Flask app and configure SQLAlchemy:

from flask import Flask

from flask\_sqlalchemy import SQLAlchemy

app = Flask(\_\_name\_\_)

Configure the database URI

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///database.db' # Use 'sqlite:///:memory:' for an in-memory database

# Create a SQLAlchemy instance

db = SQLAlchemy(app)

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

Ans:

To create a RESTful API endpoint in Flask that returns JSON data, you can use the Flask-RESTful library:

1. Import libraries: In a new Python file, import Flask, API, and Resource from flask\_restful.
2. Register app: Register the web app as an app variable and as an API object.
3. Create resource class: Define a class that returns JSON, such as ReturnJSON.
4. Return JSON: In the get method, return a dictionary with the JSON response.
5. Add resource: Use the add\_resource method to add the resource class to the API.

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

Ans:

Flask-WTF is an extension that makes creating and handling forms in Flask applications easier. It leverages the powerful WTForms library to provide a simple and secure way to manage user input.

1. Installation:

Install Flask-WTF using pip:

2. Creating a Form:

Create a form class by inheriting from FlaskForm. Define the form fields using WTForms field classes like StringField, IntegerField, PasswordField, etc. Add validators to enforce specific rules on the input data.

3. Rendering the Form in a Template:

Pass the form object to your template and render it using the form helper.

4. Validating Form Data:

Use the validate\_on\_submit() method to validate form data on submission. If validation fails, error messages are automatically displayed in the template.

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

Ans:

1. HTML Form:

Create an HTML form with the enctype="multipart/form-data" attribute. This allows the form to handle file uploads. Include an <input type="file"> element for the user to select the file.

<form method="POST" action="/upload" enctype="multipart/form-data">

<input type="file" name="file">

<button type="submit">Upload</button>

</form>

2. Flask Route:

Create a Flask route to handle the form submission. Use the request.files object to access the uploaded file.

from flask import Flask, request, render\_template

import os

app = Flask(\_\_name\_\_)

UPLOAD\_FOLDER = 'uploads'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

@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'

if file:

file.save(os.path.join(app.config['UPLOAD\_FOLDER'], file.filename))

return 'File uploaded successfully'

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

Ans:

1. Import the Blueprint class:

from flask import Blueprint

2. Create a Blueprint object:

my\_blueprint = Blueprint("my\_blueprint", \_\_name\_\_)

* "my\_blueprint": The name of your blueprint.
* "\_\_name\_\_": The name of the module where the blueprint is defined.

3. Define routes and functions:

@my\_blueprint.route("/")

def home():

return "Hello, World!"

4. Register the blueprint with the Flask application:

app.register\_blueprint(my\_blueprint)

Reasons to use Flask Blueprints:

* Organize your application:

Break down your application into smaller, reusable components, making it easier to maintain and scale.

* Simplify large applications:

Manage complex applications by grouping related views and functions together.

* Support common patterns:

Implement common patterns across applications or within a single application.

* Enable extension registration:

Provide a central location for Flask extensions to register operations on your application.

* Reuse code:

Easily reuse blueprint components across different applications.

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

Ans:

Deploying a Flask app to a production server using Gunicorn and Nginx involves setting up a virtual environment, installing dependencies, and configuring Nginx. Here's a general overview of the steps:

* Set up: Create a non-root user and a directory for the Flask app. Set up a virtual environment.
* Install dependencies: Install Python, pip, Flask, and Gunicorn. Install any other libraries the app needs.
* Configure Gunicorn: Create a systemd service and activate it.
* Configure Nginx: Set up Nginx to act as a reverse proxy and static files server.
* Permissions: Grant necessary permissions.

50. Machine Learning:

What is the difference between Series & Dataframes.

Ans:

* Dimensionality

Series are one-dimensional, while DataFrames are two-dimensional.

* Data alignment

Series align data based on index labels, while DataFrames can align data along both rows and columns.

* Data types

Series can only hold one data type at a time, while DataFrames can hold multiple data types in different columns.

* Functionality

DataFrames are more versatile than Series, making them useful for tasks like merging datasets or performing operations across multiple dimensions.

* Structure

Series are similar to NumPy arrays and contain a sequence of values and an associated index. DataFrames can be thought of as a collection of Series, where each column is a Series.

Difference between loc and iloc.

Ans:

|  |  |
| --- | --- |
| loc() | iloc() |
| Label-based data selector | Index-based data selector |
| Indices should be sorted in order, or loc[ ] will only select the mentioned indices when slicing | Indices need not be sorted in order when slicing |
| Indices should be numerical, else slicing cannot be done | Indices can be numerical or categorical |
| The end index is included during slicing | The end index is excluded during slicing |
| Accepts bool series or list in conditions | Only accepts bool list in conditions |

What is the difference between supervised and unsupervised learning?

Ans:

* Data

Supervised learning models have a baseline understanding of the correct output values, while unsupervised models don't.

* Human intervention

Supervised learning involves a human "teacher" who provides labeled examples. Unsupervised learning operates autonomously, without human supervision.

* Goal

Supervised learning aims to predict outcomes for new data, while unsupervised learning aims to gain insights from large amounts of new data.

* Applications

Supervised learning is used for email spam classification, image recognition, and stock price prediction. Unsupervised learning is used for organizing data archives, building recommendation systems, and grouping customers based on purchasing behaviors.

* Model complexity

Supervised learning models are often calculated using simple tools like Python or R. Unsupervised learning models can be computationally complex.

Explain the bias-variance tradeoff.

Ans:

If the algorithm is too simple (hypothesis with linear equation) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree equation) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well. Well, there is something between both of these conditions, known as a Trade-off or Bias Variance Trade-off. This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can’t be more complex and less complex at the same time.

What are precision and recall? How are hey different from accuracy?

Ans:

* Precision: Measures how accurate the model is at predicting a specific category. It indicates the proportion of correctly predicted positive instances.
* Recall: Measures how well the model identifies all actual positive instances. It indicates the proportion of actual positive instances that the model identifies correctly.

Accuracy shows how often a classification ML model is correct overall. Precision shows how often an ML model is correct when predicting the target class. Recall shows whether an ML model can find all objects of the target class. Consider the class balance and costs of different errors when choosing the suitable metric.

What is overfitting and how can it be prevented?

Ans:

Overfitting is a modeling error that occurs when a model is too closely aligned with a limited set of data. This makes the model only relevant to the initial data set, and not to any other data sets.

* Regularization

Reduces a model's complexity by adding a penalty term to the loss function. L1 regularization is particularly useful for identifying and focusing on the most essential features in a dataset.

* Data augmentation

Artificially increases the size and diversity of a dataset by applying transformations like rotation, cropping, or flipping to the existing data.

* Cross-validation

Splits the dataset into subsets, trains the model on different combinations of those subsets, and evaluates its performance on the remaining subsets. This technique helps identify potential overfitting.

* Use more complete training data

The dataset should cover the full range of inputs that the model is expected to handle.

* Ensembling

Creates an ensemble model by aggregating multiple models. Well-known ensemble methods include bagging and boosting.

Explain the concept of cross-validation.

Ans:

Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. We can also say that it is a technique to check how a statistical model generalizes to an independent dataset.

In [machine learning](https://www.javatpoint.com/machine-learning), there is always the need to test the stability of the model. It means based only on the training dataset; we can't fit our model on the training dataset. For this purpose, we reserve a particular sample of the dataset, which was not part of the training dataset. After that, we test our model on that sample before deployment, and this complete process comes under cross-validation. This is something different from the general train-test split.

Validation Set Approach

We divide our input dataset into a training set and test or validation set in the validation set approach. Both the subsets are given 50% of the dataset.

But it has one of the big disadvantages that we are just using a 50% dataset to train our model, so the model may miss out to capture important information of the dataset. It also tends to give the underfitted model.

Leave-P-out cross-validation

In this approach, the p datasets are left out of the training data. It means, if there are total n datapoints in the original input dataset, then n-p data points will be used as the training dataset and the p data points as the validation set. This complete process is repeated for all the samples, and the average error is calculated to know the effectiveness of the model.

There is a disadvantage of this technique; that is, it can be computationally difficult for the large p.

Leave one out cross-validation

This method is similar to the leave-p-out cross-validation, but instead of p, we need to take 1 dataset out of training. It means, in this approach, for each learning set, only one datapoint is reserved, and the remaining dataset is used to train the model. This process repeats for each datapoint. Hence for n samples, we get n different training set and n test set. It has the following features:

In this approach, the bias is minimum as all the data points are used.

The process is executed for n times; hence execution time is high.

This approach leads to high variation in testing the effectiveness of the model as we iteratively check against one data point.

K-Fold Cross-Validation

K-fold cross-validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called folds. For each learning set, the prediction function uses k-1 folds, and the rest of the folds are used for the test set. This approach is a very popular CV approach because it is easy to understand, and the output is less biased than other methods.

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

Ans:

* Classification

Predicts discrete categories or labels, like "spam" or "not spam". The output is usually a class or label from a predefined set of options.

* Regression

Predicts continuous numerical values, like age, income, or temperature. The output is a real-valued number that can range within a set limit.

Goal

Classification assigns input data to predefined categories, while regression establishes a relationship between input variables and the output.

Common output values

Classification results in labels like "yes" or "no", while regression results in numbers like salaries or sales figures.

Examples

Classification can be used for cancer detection or spam detection, while regression can be used for weather forecasting or predicting market trends.

Algorithms

Classification uses algorithms like logistic regression and K-nearest neighbours, while regression uses algorithms like simple linear regression and decision tree regression.

Explain the concept of ensemble learning.

Ans:

Ensemble learning is a [machine learning](https://www.ibm.com/topics/machine-learning) technique that aggregates two or more learners (e.g. [regression](https://www.ibm.com/topics/linear-regression) models, [neural networks](https://www.ibm.com/topics/neural-networks)) in order to produce better predictions. In other words, an ensemble model combines several individual models to produce more accurate predictions than a single model alone.

-Parallelmethods train each base learner apart from the others of the others. Per its name, then, parallel ensembles train base learners in parallel and independent of one another.

- Sequential methods train a new base learner so that it minimizes errors made by the previous model trained in the preceding step. In other words, sequential methods construct base models sequentially in stages.

[Bagging](https://www.ibm.com/topics/bagging) is a homogenous parallel method sometimes called *bootstrap aggregating*. It uses modified replicates of a given training data set to train multiple base learners with the same training algorithm. Scikit-learn’s ensemble module in Python contains functions for implementing bagging, such as BaggingClassifier.

Stacking, or stacked generalization, is a heterogenous parallel method that exemplifies what is known as meta-learning. Meta-learning consists of training a meta-learner from the output of multiple base learners. Stacking specifically trains several base learners from the same dataset using a different training algorithm for each learner. Each base learner makes predictions on an unseen dataset. These first model predictions are then compiled and used to train a final model, being the meta-model.

[Boosting](https://www.ibm.com/topics/boosting) algorithms are a sequential ensemble method. Boosting has many variations, but they all follow the same general procedure. Boosting trains a learner on some initial dataset, *d*. The resultant learner is typically weak, misclassifying many samples in the dataset. Much like bagging, boosting then samples instances from the initial dataset to create a new dataset .

- Adaptive boosting (AdaBoost) weights model errors. That is, when creating a new iteration of a dataset for training the next learner, AdaBoost adds weights to the previous learner’s misclassified samples, causing the next learner to prioritize those misclassified samples.

What is gradient descent and how does it work?

Ans:

Gradient Descent is known as one of the most commonly used optimization algorithms to train machine learning models by means of minimizing errors between actual and expected results. Further, gradient descent is also used to train Neural Networks.

In mathematical terminology, Optimization algorithm refers to the task of minimizing/maximizing an objective function f(x) parameterized by x. Similarly, in machine learning, optimization is the task of minimizing the cost function parameterized by the model's parameters. The main objective of gradient descent is to minimize the convex function using iteration of parameter updates. Once these machine learning models are optimized, these models can be used as powerful tools for Artificial Intelligence and various computer science applications.

* If we move towards a negative gradient or away from the gradient of the function at the current point, it will give the local minimum of that function.
* Whenever we move towards a positive gradient or towards the gradient of the function at the current point, we will get the local maximum of that function.

This entire procedure is known as Gradient Ascent, which is also known as steepest descent. The main objective of using a gradient descent algorithm is to minimize the cost function using iteration***.*** To achieve this goal, it performs two steps iteratively:

* Calculates the first-order derivative of the function to compute the gradient or slope of that function.
* Move away from the direction of the gradient, which means slope increased from the current point by alpha times, where Alpha is defined as Learning Rate. It is a tuning parameter in the optimization process which helps to decide the length of the steps.

The cost function is defined as the measurement of difference or error between actual values and expected values at the current position and present in the form of a single real number. It helps to increase and improve machine learning efficiency by providing feedback to this model so that it can minimize error and find the local or global minimum. Further, it continuously iterates along the direction of the negative gradient until the cost function approaches zero. At this steepest descent point, the model will stop learning further. Although cost function and loss function are considered synonymous, also there is a minor difference between them. The slight difference between the loss function and the cost function is about the error within the training of machine learning models, as loss function refers to the error of one training example, while a cost function calculates the average error across an entire training set.

The cost function is calculated after making a hypothesis with initial parameters and modifying these parameters using gradient descent algorithms over known data to reduce the cost function.

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

Ans:

Batch gradient descent (BGD) and stochastic gradient descent (SGD) are both optimization algorithms used in machine learning that differ in the number of samples used to calculate the gradient. BGD is better suited for small datasets, while SGD is better for large ones.

* Dataset usage:

BGD uses the entire training dataset in each iteration to calculate the cost function's gradient. SGD uses a single training example or a small subset of examples.

* Parameter updates:

BGD doesn't change the model until it assesses every training sample in a cycle, or training epoch. SGD updates the parameters after each observation, or iteration.

* Speed:

SGD can be faster than BGD, especially as the dataset increases in size.

* Memory:

BGD requires more memory because it needs to store the entire dataset for each iteration.

* Convergence:

BGD has more predictable convergence because it uses the average gradient of the entire dataset.

* Accuracy:

SGD may not provide the exact solution due to its random nature, but it can provide the best approximate solution.

* Stability:

SGD updates are noisy and have high variance, which can make the optimization process less stable.

What is the curse of dimensionality in machine learning?

Ans:

The curse of dimensionality is a common issue in machine learning that refers to the challenges that arise when analyzing and organizing data in high-dimensional spaces. As the number of features or dimensions in a dataset increases, the amount of data needed to generalize accurately grows exponentially. This can make it difficult or impossible for machines to understand and analyze the data, making it hard to find patterns or make accurate predictions.

Here are some ways the curse of dimensionality can manifest:

* Data becomes more spread out

In high-dimensional spaces, most points are very far apart from each other, making it difficult to work with the data.

* Data organization strategies are inefficient

In high-dimensional data, all objects appear to be sparse and dissimilar in many ways, which prevents common data organization strategies from being efficient.

* Distance-based algorithms fail

For example, k-Nearest Neighbor (KNN) algorithms tend to fail when the number of dimensions in the data is very high.

Explain the difference between L1 and L2 regularization.

Ans:

|  |  |
| --- | --- |
| **L1 Regularization** | **L2 Regularization** |
| The penalty term is based on the absolute values of the model's parameters. | The penalty term is based on the squares of the model's parameters. |
| Produces sparse solutions (some parameters are shrunk towards zero). | Produces non-sparse solutions (all parameters are used by the model). |
| Sensitive to outliers. | Robust to outliers. |
| Selects a subset of the most important features. | All features are used by the model. |
| Optimization is non-convex. | Optimization is convex. |
| The penalty term is less sensitive to correlated features. | The penalty term is more sensitive to correlated features. |
| Useful when dealing with high-dimensional data with many correlated features. | Useful when dealing with high-dimensional data with many correlated features and when the goal is to have a less complex model. |
| Also known as Lasso regularization. | Also known as Ridge regularization. |

What is a confusion matrix and how is it used?

Ans:

A confusion matrix is a table that summarizes the performance of a classification model in machine learning. It's also known as an error matrix. The matrix compares predicted values against actual values for a dataset, and it's based on the concepts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

The matrix has two dimensions, with identical sets of classes in both dimensions. Each row represents instances in an actual class, while each column represents instances in a predicted class. The diagonal of the matrix represents all instances that are correctly predicted.

A confusion matrix can help you:

* Assess accuracy and effectiveness: It provides a granular view of a model's performance across different classes.
* Identify which classes are most often misplaced: It shows the number of correctly and wrongly classified data.
* Calculate other model performance metrics: You can use it to calculate metrics such as precision and recall. For example, precision measures how many of the predicted positive cases were correct.

Define AUC-ROC curve.

Ans:

AUC-ROC stands for "Area Under the Receiver Operating Characteristic Curve". It's a performance metric used in machine learning to evaluate binary classification models. The AUC-ROC curve is a probability curve that plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at different classification thresholds. The AUC measures the area under the ROC curve, which ranges from 0 to 1.

The AUC-ROC curve is useful for:

* Measuring performance

It measures how well a model performs across all possible classification thresholds.

* Comparing classifiers

It can help compare the performance of different classifiers for the same problem.

* Determining classification thresholds

It can help determine a classification threshold that's appropriate for a specific problem. For example, if a classifier is predicting disease, you might allow more false positives to ensure that no true positives are missed.

A score of 0.5 on the AUC-ROC curve indicates random guessing, while a score of 1 indicates perfect performance. A score above 0.8 is generally considered good, and a score above 0.9 is considered great. However, the usefulness of the model depends on the specific problem and use case, so the score should be interpreted in context with other classification quality metrics.

Explain the k-nearest neighbors algorithm.

Ans:

The K-Nearest Neighbors (KNN) algorithm is a popular machine learning technique used for classification and regression tasks. It relies on the idea that similar data points tend to have similar labels or values.

During the training phase, the KNN algorithm stores the entire training dataset as a reference. When making predictions, it calculates the distance between the input data point and all the training examples, using a chosen distance metric such as Euclidean distance.

Next, the algorithm identifies the K nearest neighbors to the input data point based on their distances. In the case of classification, the algorithm assigns the most common class label among the K neighbors as the predicted label for the input data point. For regression, it calculates the average or weighted average of the target values of the K neighbors to predict the value for the input data point.

The KNN algorithm is straightforward and easy to understand, making it a popular choice in various domains. However, its performance can be affected by the choice of K and the distance metric, so careful parameter tuning is necessary for optimal results.

When Do We Use the KNN Algorithm:

KNN Algorithm can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique, we generally look at 3 important aspects:

1. Ease of interpreting output

2. Calculation time

3. Predictive Power

Breaking It Down – Pseudo Code of KNN

We can implement a KNN model by following the below steps:

1. Load the data
2. Initialise the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
   * Calculate the distance between test data and each row of training dataset. Here we will use Euclidean distance as our distance metric since it’s the most popular method. The other distance function or metrics that can be used are Manhattan distance, Minkowski distance, Chebyshev, cosine, etc. If there are categorical variables, hamming distance can be used.
   * Sort the calculated distances in ascending order based on distance values
   * Get top k rows from the sorted array
   * Get the most frequent class of these rows
   * Return the predicted class

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

Ans:

A Support Vector Machine (SVM) is a supervised machine learning algorithm that analyzes data for classification and regression. SVMs are known for finding a hyperplane that maximizes the margin, or distance, between two classes of data points. This makes them resilient to noisy data and less likely to overfit to training data.  
Here's how SVMs work:

* Training: SVMs are trained on sets of labeled data for each category.
* Finding the hyperplane: SVMs search for a line or hyperplane that maximizes the distance between the closest data points from opposite classes.
* Classifying new data: SVMs can categorize new text and label unseen data points based on the hyperplane.

SVMs have several advantages, including:

* Faster speeds: Compared to newer algorithms like neural networks, SVMs are faster.
* Better performance with limited samples: SVMs perform well with datasets of just a few thousand samples, making them suitable for text classification.
* Non-linear classification: SVMs can perform non-linear classification using the “kernel trick” to map inputs to higher-dimensional spaces.

SVMs have many applications, including:

* Facial detection and recognition

SVMs can distinguish facial structures and expressions like happy, sad, or angry.

* Bioinformatics

SVMs can be used for cancer classification, diabetes prediction, and protein classification.

* Text classification

SVMs can be used to arrange text and find important information.

* Anomaly detection

SVMs can be used to detect anomalies in power datasets and smart environments.

How does the kernel trick work in SVM?

Ans:

The kernel trick is a key component of Support Vector Machines (SVMs) that helps them handle non-linear data. It works by mapping input data into a higher-dimensional space using a kernel function, then using a linear separator to divide the classes. The kernel trick is computationally efficient because it avoids directly calculating the coordinates in the higher space.

Here's how the kernel trick works:

1. 1. Represent data with pairwise similarity comparisons

The kernel function has a special mathematical property that allows it to represent the data set X as an n x n kernel matrix of pairwise similarity comparisons. The entries (i, j) are defined by the kernel function: k(xi, xj).

1. 2. Map data into a higher-dimensional space

The kernel trick uses the kernel function to map the input data points into a higher-dimensional space where it's easier to separate the two classes.

1. 3. Find the best boundary

The SVM then finds the best possible boundary that can separate the two classes of data points with maximum margin, also known as the hyperplane.

The kernel trick allows you to bypass the need for specifying a non-linear transformation explicitly. Instead, you specify a "kernel" function that directly describes how each point relates to each other.

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

Ans:

The type of kernel used in a Support Vector Machine (SVM) depends on the nature of the data and the task at hand. Here are some common types of kernels used in SVMs:

* Linear kernel

The simplest kernel, it draws straight lines to separate data. It's good for data that's linearly separable, meaning it can be separated by a single line. It's also commonly used when a data set has a large number of features.

* Polynomial kernel

Adds curves to linear separation, allowing it to handle more complex data. It's used when data isn't linearly separable, but it's not used as often in practice because it's not as computationally efficient as other kernels and its predictions aren't as accurate.

* Radial Basis Function (RBF) kernel

Flexible and can handle all sorts of shapes, including indefinite dimensional spaces. It's a general-purpose kernel that's often used in SVM classification when there's no prior knowledge about the data.

* Sigmoid kernel

Works for data with no clear shapes, using a hyperbolic tangent function.

* ANOVA radial basis kernel

Performs well in multidimensional regression problems, similar to the Gaussian and Laplacian kernels.

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

Ans:

In a support vector machine (SVM) algorithm, a hyperplane is a decision boundary that separates data points of different classes in a feature space. The dimension of the hyperplane depends on the number of input features in the dataset:

* 2 input features: The hyperplane is a line
* 3 input features: The hyperplane is a two-dimensional plane
* The goal of an SVM is to find the hyperplane that best separates the different classes. The optimal hyperplane maximizes the distance between the class bounding hyperplanes, also called supporting hyperplanes. The bounding hyperplanes are parallel to the optimal hyperplane and are defined by the training samples that are closest to the boundary, called the support vectors. The distance between the support vector and hyperplane is called the margin.
* Careful hyperplanning can help improve the accuracy and generalizability of the model. For example, SVMs can be used to determine whether a digital image has been tampered with, contaminated, or is pure.

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

Ans:

Support Vector Machines (SVMs) are machine learning algorithms that can handle high-dimensional data and perform well with small datasets. However, they can be computationally expensive and slow with large datasets. Here's a breakdown of the pros and cons:

Pros

* High-dimensional data: SVMs can handle high-dimensional data well, such as text and image data, and perform better than logistic regression.
* Small datasets: SVMs can perform well with small datasets.
* Unstructured data: SVMs can work well with unstructured and semi-structured data like text and images.
* Overfitting: SVMs are less prone to overfitting compared to decision trees.
* Global optimum: SVMs only have one global optimum, unlike other machine learning algorithms that can get stuck in local optima.
* Hyperparameters: SVMs don't require many hyperparameters to choose from.
* Memory efficiency: SVMs use a subset of the training set in the decision function, making them memory efficient.

Cons

* Big data: SVMs can be slow and consume a lot of memory when dealing with large datasets with many features.
* Noise and outliers: SVMs are sensitive to noise and outliers.
* Nonlinear SVMs: Nonlinear SVMs can be slow.
* Tuning hyperparameters: SVMs require tuning for different hyperparameters.
* Overlapping target classes: SVMs don't perform well when data sets have overlapping target classes.
* Kernel selection: Choosing the right kernel can be challenging.
* Computational intensity: SVMs can be computationally intensive.
* Algorithm complexity: SVM algorithms can be complex.
* Model overfitting: SVMs can overfit the model.

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

Ans:

Hard margin and soft margin Support Vector Machines (SVMs) are classifier models that differ in how they separate data:

* Hard margin SVM

Aims for perfect separation between two classes, with no misclassification. It works best for linearly separable data and maximizes the margin between the decision boundary and data points. However, hard margins are sensitive to outliers and can overfit.

* Soft margin SVM

Allows some misclassification, with a penalty for non-separable samples. It uses slack variables to introduce some misclassification, resulting in a wider margin and better generalization to unseen data. Soft margins are useful when a linear boundary isn't possible or when some misclassification is desired.

Describe the process of constructing a decision tree.

Ans:

Step 1: Identify the problem

Every [decision tree](https://miro.com/diagramming/what-is-a-decision-tree-diagram/) begins with a clear understanding of the problem at hand. Identify the goals and objectives, as well as the key variables and factors that will influence the decision. This step lays the foundation for the entire analysis. The more precise your problem definition, the better your decision tree will serve you.

Step 2: Begin to structure the decision tree

Once the problem is well-defined, the next step is to begin [creating the decision tree](https://miro.com/diagramming/decision-tree/). The tree starts with a decision node from which branches extend, representing different options. Further nodes are added to represent the potential outcomes of those options.

At this stage you won't have a clear idea as to how far each branch will extend, but by laying out the groundwork you'll begin to get a visual sense of how the decision tree will evolve. Add notes and other documents to serve as a reference as you continue to build you your decision tree.

Step 3: Identify decision alternatives

Continue building out the decision tree by listing all possible alternatives or courses of action available. Add these as branches stemming from the central decision node.

These alternatives represent the different paths or choices that can be taken in the decision-making process. Ensure that you include a comprehensive range of options and potential outcomes for each.

Step 4: Estimate payoffs or costs

Assign payoffs or costs to each outcome. These values represent the impact or consequences of each outcome on the overall decision. Consider both quantitative and qualitative factors when estimating payoffs or costs.

Step 5: Assign probabilities

Assigning probabilities to each potential outcome is crucial. These probabilities can be derived from historical data, market research, or expert judgment. They represent the likelihood of each outcome occurring, providing a quantitative basis for your decision-making process.

Step 6: Determine the potential outcomes

Each outcome has a value attached to it. This could be the potential financial gain or loss, the impact on customers, or any other metric that matters to your decision. These outcomes should cover both the positive and negative aspects, as well as any uncertainties or risks involved. Multiply each outcome value by its probability to calculate the expected value of each decision path.

Step 7: Analyze and select the best decision

Now comes the analysis. By adding up the expected values of each decision path, you can identify the most promising option. This decision point provides the highest expected value, giving you a data-driven recommendation for your strategic decision.

Step 6: Review and update the decision tree

Decision trees are not set in stone. As new information becomes available or circumstances change, your decision tree should evolve. You can perform a sensitivity analysis at this stage by testing key assumptions, probabilities, or payoffs. This step helps identify the robustness of the chosen decision and provides insights into potential areas of uncertainty or risk.

Describe the working principle of a decision tree.

Ans:

A decision treeis a flowchart-like structure used to make decisions or predictions. It consists of nodes representing decisions or tests on attributes, branches representing the outcome of these decisions, and leaf nodes representing final outcomes or predictions. Each internal node corresponds to a test on an attribute, each branch corresponds to the result of the test, and each leaf node corresponds to a class label or a continuous value.

Structure of a Decision Tree

1. Root Node: Represents the entire dataset and the initial decision to be made.
2. Internal Nodes: Represent decisions or tests on attributes. Each internal node has one or more branches.
3. Branches: Represent the outcome of a decision or test, leading to another node.
4. Leaf Nodes: Represent the final decision or prediction. No further splits occur at these nodes.

The process of creating a decision tree involves:

1. Selecting the Best Attribute: Using a metric like Gini impurity, entropy, or information gain, the best attribute to split the data is selected.
2. Splitting the Dataset: The dataset is split into subsets based on the selected attribute.
3. Repeating the Process: The process is repeated recursively for each subset, creating a new internal node or leaf node until a stopping criterion is met (e.g., all instances in a node belong to the same class or a predefined depth is reached).

What is information gain and how is it used in decision trees.

Ans:

In decision trees, information gain (IG) measures how much a feature contributes to accurate predictions by quantifying how well it separates training examples. IG calculates the reduction in entropy, or uncertainty, of the target variable after a specific feature is known.

Explanation

* Entropy: A metric that measures the impurity or randomness in data.
* Information gain: The difference between entropy before and after splitting the dataset based on a specific attribute.

When building a decision tree, the goal is to find the attribute with the highest IG and lowest entropy. Features with higher IG are more informative and result in nodes with more homogenous classes. IG helps determine which feature to use to split data at each internal node of the tree.

The ID3 (Iterative Dichotomiser) decision tree algorithm uses IG.

Explain Gini impurity and its role in decision trees.

Ans:

Gini impurity is a measurement in decision tree algorithms that quantifies a dataset's disorder or impurity level. It helps identify optimal splits for nodes, which can lead to more accurate predictions and homogeneous subsets.

* Explanation

Gini impurity is a number between 0–0.5 that indicates the likelihood of misclassifying new, random data. It's a variation of the entropy measure used in decision trees. A lower Gini impurity indicates a better split, or a lower chance of misclassification. Gini impurity is at its minimum (zero) when all cases in a node belong to the same target category.

* Role

Gini impurity helps determine how a dataset's features should split nodes to form the tree. It measures how often a randomly chosen element of a set would be incorrectly labeled if it were labeled randomly and independently according to the set's label distribution.

What are the advantages and disadvantages of decision trees?

Ans:

Decision trees are a simple and powerful machine learning algorithm used for classification and regression tasks. They have a hierarchical structure and use a divide and conquer approach to classify records. Here's a summary of their advantages and disadvantages:

Advantages

* + Interpretability: Easy to understand and visualize, with a visual representation that mirrors human decision-making.
  + Versatility: Can handle both classification and regression tasks, and can capture non-linear relationships between features and target variables.
  + Non-parametric: Doesn't require data normalization or scaling.
  + Handles missing values: Can handle unbalanced data and missing values in the data.
  + Little data preparation: Requires little data preparation.

Disadvantages

* + Overfitting: Can overfit the training data, especially if the tree is too complex. This can lead to poor generalization to new data.
  + Instability: Can be prone to instability.
  + High-dimensional data: Not well-suited for high-dimensional data or large datasets.
  + Sensitivity: Sensitive to small variations.
  + Biased learning: Can be susceptible to biased learning.

How do random forests improve upon decision trees?

Ans:

Random forests can improve on decision trees in several ways, including:

* Overfitting

Random forests are less likely to overfit than decision trees because they combine the predictions of multiple trees, which reduces the variance of the model. Decision trees can become too complex and overfit, which can lead to poor performance on unseen data.

* Data sampling

Random forests sample data multiple times before generating a prediction, so small changes in data don't significantly impact the model's prediction.

* Robustness

Random forests are more robust and generalized when performing on new data.

How does a random forest algorithm work?

Ans:

The random forest algorithm is a machine learning classifier that uses multiple decision trees to improve the accuracy of predictions for a given dataset. It works by randomly selecting features and observations to build several decision trees, then averaging the results. The algorithm is based on the concept of ensemble learning, which combines multiple classifiers to solve complex problems.

Here are the steps the random forest algorithm takes:

1. Generate the random forest: Choose a random subset of data points from the training set and create decision trees for those subsets. Repeat these steps to create the desired number of decision trees.
2. Make predictions: For each decision tree, find predictions for new data points.
3. Combine the results: Assign the new data points to the category with the most votes from the predictions of the individual trees.

The more trees in the random forest, the higher its accuracy and problem-solving ability. However, building too many trees can slow down computation.

What is bootstrapping in the context of random forests?

Ans:

The Bootstrap

Bootstrapping[[1]](https://www.quantstart.com/articles/bootstrap-aggregation-random-forests-and-boosted-trees/#ref-efron1979) is a statistical resampling technique that involves random sampling of a dataset with replacement. It is often used as a means of quantifying the uncertainty associated with a machine learning model.

For quantitative finance purposes bootstrapping is extremely useful since it allows generation of new samples from a population without having to go and collect additional "training data". In quantitative finance applications it is often impossible to generate more data in the case of financial asset pricing series as there is only one "history" to sample from.

The idea is to repeatedly sample data with replacement from the original training set in order to produce multiple separate training sets. These are then used to allow "meta-learner" or "ensemble" methods to reduce the variance of their predictions, thus greatly improving their predictive performance.

Bootstrap Aggregation

One of the main drawbacks of DTs is that they suffer from being high-variance estimators. This means that the addition of a small number of extra training observations can dramatically alter the prediction performance of a learned tree, despite the training data not changing to any great extent.

This is in contrast to a low-variance estimator such as linear regression, which is not hugely sensitive to the addition of extra points–at least those that are relatively close to the remaining points.

One way to mitigate against this problem is to utilise a concept known as bootstrap aggregation or bagging. The idea is to combine multiple leaners (such as DTs), which are all fitted on separate bootstrapped samples and average their predictions in order to reduce the overall variance of these predictions.

Explain the concept of feature importance in random forests.

Ans:

In Random Forests, feature importance can help you understand how each feature contributes to the model's accuracy. Features that are ranked highly have a significant influence on the model's decision-making. Feature importance can be used for:

* Data comprehension: Understanding the relationship between the features and the target variable
* Model improvement: Reducing the dimensionality of the model
* Feature selection: Selecting a subset of relevant features for use in building a model

There are several techniques for calculating feature importance in Random Forests, including:

* Built-in feature importance

This method uses the model's internal calculations to measure feature importance by computing the average over all trees in the forest. It's available in the scikit-learn implementation of the Random Forest.

* Permutation feature importance

This method assesses the significance of each feature independently by measuring the change in the model's performance when the feature's values are randomly shuffled. The permutation is done after the model is trained, using an out-of-sample dataset. This technique overcomes limitations of the impurity-based feature importance, such as bias toward high-cardinality features.

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

Ans:

Random forest algorithms have several hyperparameters that control the model's training behavior and complexity, and can significantly impact its performance. These hyperparameters are set before training begins. Some key hyperparameters include:

* Number of trees (n\_estimators)

The number of decision trees in the forest. Increasing the number of trees can improve accuracy and reduce overfitting. However, too many trees can make the model slow and ineffective for real-time predictions.

* Maximum depth of trees (max\_depth)

The maximum depth of each decision tree. Fine-tuning this parameter doesn't usually impact the forest's performance significantly.

* Minimum samples per leaf node (min\_samples\_leaf)

The minimum number of samples a leaf node must contain. This can help reduce overfitting when there are many parameters.

* Maximum number of features (max\_features)

The number of features considered when splitting a node in a decision tree.

* Criterion

The measure used to evaluate the quality of splits, such as Gini impurity or entropy.

* Splitting rule

The rule used to split trees during construction.

Describe the logistic regression model and its assumptions.

Ans:

Logistic regression is a statistical model and data analysis technique that uses mathematics to predict the likelihood of an event or class. It's a non-linear transformation of linear regression that's commonly used for binary regression problems, where the output is a binary value (0 or 1). For example, logistic regression can be used to predict whether an email is spam or not spam, or whether a patient is healthy or not.

In logistic regression, the model estimates the parameters of a logistic model that models the log-odds of an event as a linear combination of one or more independent variables. The log-odds are then converted to probabilities using the logistic function, which constrains the probabilities to lie between 0 and 1. The output probability is then used to assign the data to a category based on a chosen threshold.

Logistic regression is useful for classification problems, such as in cyber security for attack detection, or in banking and other financial institutions for fraud detection. It can also be used in medicine to predict the likelihood of disease or illness for a given population.

Logistic regression is a statistical model that makes certain assumptions about the data:

* Independence

Observations are independent of each other and not the result of repetitive measurements. This means that no individual should be measured more than once.

* Linearity

There is a linear relationship between the log odds of the independent and dependent variables.

* No multicollinearity

There is not a lot of correlation between the independent variables. Multicollinearity can reduce the precision of the estimated coefficients, which can weaken the model's statistical power.

* Large sample size

A large number of samples is needed to get accurate estimates of the coefficients.

* No outliers

Extreme values or outliers in the data are not expected to affect the results of the logistic regression.

* Binary or dichotomous response variable

The dependent variable can only take on two possible outcomes, such as pass/fail, male/female, or malignant/benign.

How does logistic regression handle binary classification problems?

Ans:

The logistic regression model transforms the [linear regression](https://www.geeksforgeeks.org/ml-linear-regression/) function continuous value output into categorical value output using a sigmoid function, which maps any real-valued set of independent variables input into a value between 0 and 1. This function is known as the logistic function.

Let the independent input features be:

 𝑋=[𝑥11 …𝑥1𝑚𝑥21 …𝑥2𝑚 ⋮⋱ ⋮ 𝑥𝑛1 …𝑥𝑛𝑚]*X*=​*x*11​ *x*21​  ⋮*xn*1​ ​……⋱ …​*x*1*m*​*x*2*m*​⋮ *xnm*​​​

 and the dependent variable is Y having only binary value i.e. 0 or 1.

𝑌={0 if 𝐶𝑙𝑎𝑠𝑠11 if 𝐶𝑙𝑎𝑠𝑠2*Y*={01​ if *Class*1 if *Class*2​

then, apply the multi-linear function to the input variables X.

𝑧=(∑𝑖=1𝑛𝑤𝑖𝑥𝑖)+𝑏*z*=(∑*i*=1*n*​*wi*​*xi*​)+*b*

Here 𝑥𝑖*xi*​ is the ith observation of X, 𝑤𝑖=[𝑤1,𝑤2,𝑤3,⋯,𝑤𝑚]*wi*​=[*w*1​,*w*2​,*w*3​,⋯,*wm*​] is the weights or Coefficient, and b is the bias term also known as intercept. simply this can be represented as the dot product of weight and bias.

𝑧=𝑤⋅𝑋+𝑏*z*=*w*⋅*X*+*b*

whatever we discussed above is the [linear regression](https://www.geeksforgeeks.org/ml-linear-regression/).

What is the sigmoid #unction and ho is it used in logistic regression.

Ans:

We use the [sigmoid function](https://www.geeksforgeeks.org/derivative-of-the-sigmoid-function/) where the input will be z and we find the probability between 0 and 1. i.e. predicted y.

𝜎(𝑧)=11+𝑒−𝑧*σ*(*z*)=1+*e*−*z*1​

*Sigmoid function*

As shown above, the figure sigmoid function converts the continuous variable data into the [probability](https://www.geeksforgeeks.org/probability-gq/) i.e. between 0 and 1.

* 𝜎(𝑧):  *σ*(*z*)  tends towards 1 as 𝑧→∞*z*→∞
* 𝜎(𝑧):   *σ*(*z*)  tends towards 0 as 𝑧→−∞*z*→−∞
* 𝜎(𝑧) :  *σ*(*z*)  is always bounded between 0 and 1

where the probability of being a class can be measured as:

𝑃(𝑦=1)=𝜎(𝑧)𝑃(𝑦=0)=1−𝜎(𝑧)*P*(*y*=1)=*σ*(*z*)*P*(*y*=0)=1−*σ*(*z*).

Explain the concept of the cost function in logistic regression.

Ans:

A [cost function](https://www.geeksforgeeks.org/what-is-cost-function/)is a mathematical function that calculates the difference between the target actual values (ground truth) and the values predicted by the model. A function that assesses a machine learning model’s performance also referred to as a loss function or objective function. Usually, the objective of a machine learning algorithm is to reduce the error or output of cost function.

Log loss and Cost function for Logistic Regression

One of the popular metrics to evaluate models for classification by using probabilities is log loss.

F=−∑i=1Myilog(pθ(xi))+(1−yi)log(1−pθ(xi))F=−∑i=1Myilog⁡(p𝜃(xi))+(1−yi)log⁡(1−p𝜃(xi))

The cost function can be written as

F(θ)=1n∑i=1n12[pθ(xi)−Yi]2F(𝜃)=1n∑i=1n12[p𝜃(xi)−Yi]2

For Logistic Regression,

pθ(x)=g(θTx)p𝜃(x)=g(𝜃Tx)

The above equation leads to a non−convex function that acts as the cost function. The cost function logistic regression is log loss and is summarized below.

cost(pθ(x),(y))=(−log(pθ(x))ify=1 −log(1−pθ(x))ify=0)

How can logistic regression be extended to handle multiclass classification?

Ans:

Logistic regression can be extended to handle multiclass classification problems using techniques like multinomial logistic regression or softmax regression. These extensions enable logistic regression to classify instances into multiple classes by estimating the probabilities of each class.

Here's how to apply logistic regression to multiclass problems:

* Build logistic regression for each class: To determine the probability of an observation belonging to a class.
* Predict the class with the highest probability: For each data point.
* Normalize probabilities: So they sum to 1.

Multinomial logistic regression is an extension of logistic regression that uses cross-entropy loss to predict a multinomial probability distribution. The cross-entropy function measures the distance between the calculated probabilities and the target one-hot-encoding matrix. The distance value is smaller for the right target class and larger for the wrong target class.

Logistic regression models estimated by maximizing the log-likelihood objective function can be biased towards the majority class in imbalanced data.

Logistic regression models can be improved through methods like data preprocessing, feature scaling, and regularization.

What is the difference between L1 and L2 regularization in logistic regression?

Ans:

L1 (Lasso) and L2 (Ridge) regularization are both used in logistic regression to reduce model complexity and improve generalization by penalizing coefficients. The main difference is how they penalize the weights, with L1 penalizing the absolute value sum and L2 penalizing the squared value sum.

* Sparsity: L1 regularization produces sparse models by driving some coefficients to zero, which is like automatic feature selection. L2 regularization shrinks all coefficients evenly, but not necessarily to zero.
* Feature selection: L1 regularization has built-in feature selection, making it useful for high-dimensional datasets. L2 regularization doesn't perform feature selection.
* Outliers: L1 regularization is robust to outliers, while L2 is not.
* Multicollinearity: L2 regularization helps with multicollinearity and model stability.
* Solution type: L1 regularization produces sparse solutions, while L2 regularization produces non-sparse solutions.

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

Ans:

Regularization

XGBoost incorporates various regularization techniques to prevent overfitting and improve the generalization capability of the models.

Regularization methods such as L1 and L2 regularization (also known as Lasso and Ridge regularization) penalize complex models, helping to control model complexity and reduce overfitting.

Tree Pruning

XGBoost employs tree pruning algorithms to control the size of decision trees, reducing overfitting and improving computational efficiency.

Pruning techniques such as depth-based pruning and weight-based pruning remove unnecessary branches from decision trees, leading to more compact and efficient models.

Handling Missing Values

XGBoost provides built-in support for handling missing values in the dataset during training and prediction.

It automatically learns the best imputation strategy for missing values, reducing the need for manual preprocessing and imputation techniques.

Flexibility

XGBoost offers flexibility in terms of customization and parameter tuning, allowing users to fine-tune the model according to specific use cases and objectives.

It supports various objective functions and evaluation metrics, enabling users to optimize the model for different types of tasks, such as classification, regression, and ranking.

Feature Importance

XGBoost provides insights into feature importance, allowing users to understand the relative importance of different features in the dataset.

Feature importance scores generated by XGBoost help identify key predictors and guide feature selection and model interpretation.

Explain the concept of boosting in the context of ensemble learning.

Ans:

Boosting is a machine learning technique in ensemble learning that involves training multiple weak learners sequentially to produce a strong learner or ensemble model. The base models are typically weak learners, but when combined, they can produce a model with better prediction accuracy and performance. Boosting can also help avoid underfitting by reducing bias.

Boosting consists of multiple steps, including:

1. Training base models
2. Assigning weights to samples
3. Passing weighted data to the next base model
4. Repeating steps 2 and 3 until the data is fitted well enough

There are two meta-algorithms in boosting that differentiate how the base models are aggregated: Adaptive Boosting (AdaBoost) and Gradient Boosting. Other boosting techniques include:

* Stochastic Gradient Boosting

An extension of gradient boosting that introduces randomness during tree construction to prevent overfitting

* eXtreme Gradient Boosting (XGBoost)

An efficient boosting technique that checks every bit of the data values in a database.

How does XGBoost handle missing values?

Ans:

XGBoost in a MICE Framework

XGBoost is super popular tree-based algorithm due to its speed, versatility, and out-of-the-box accuracy. There’s a phenomenal explanation in the comments, but for this post you can just think of XGBoost as a black box that takes in predictors and outputs an estimate of our missing data.

MICE stands for Multiple Imputation by Chained Equations.

The way MICE works is it creates *M* copies of the data. Then it sequentially goes through the columns in the first copied dataset (M1 in figure 4) and uses a linear model to predict the missing values. The predictors are all other variables in the row. MICE then repeats this process for the rest of the *M* datasets, resulting in M complete datasets.

From there, we take the mean of the value at each index for all *M* datasets and these averages become our final imputed dataset.

What are the key hyperparameters in XGBoost and how do they affect model performance?

Ans:

General Approach for XGBoost Parameters Tuning

We will use an approach similar to that of GBM here. The various steps to be performed are:

* Choose a relatively high learning rate. Generally, a learning rate of 0.1 works, but somewhere between 0.05 to 0.3 should work for different problems. Determine the optimum number of trees for this learning rate. XGBoost has a very useful function called “cv” which performs cross-validation at each boosting iteration and thus returns the optimum number of trees required.
* Tune tree-specific parameters ( max\_depth, min\_child\_weight, gamma, subsample, colsample\_bytree) for the decided learning rate and the number of trees. Note that we can choose different parameters to define a tree, and I’ll take up an example here.
* Tune regularization parameters (lambda, alpha) for xgboost, which can help reduce model complexity and enhance performance.
* Lower the learning rate and decide the optimal parameters.

Let us look at a more detailed step-by-step approach.

Step 1: Fix Learning Rate and Number of Estimators for Tuning Tree-Based Parameters.

In order to decide on boosting parameters, we need to set some initial values of other parameters. Let’s take the following values:

* max\_depth = 5: This should be between 3-10. I’ve started with 5, but you can choose a different number as well. 4-6 can be good starting points.
* min\_child\_weight = 1: A smaller value is chosen because it is a highly imbalanced class problem, and leaf nodes can have smaller size groups.
* gamma = 0: A smaller value like 0.1-0.2 can also be chosen for starting. This will, anyways, be tuned later.
* subsample, colsample\_bytree = 0.8: This is a commonly used start value. Typical values range between 0.5-0.9.
* scale\_pos\_weight = 1: Because of high-class imbalance.

Please note that all the above are just initial estimates and will be tuned later. Let’s take the default learning rate of 0.1 here and check the optimum number of trees using the cv function of xgboost. The function defined above will do it for us.

Step2: Tune max\_depth and min\_child\_weight

We tune these first as they will have the highest impact on the model outcome. To start with, let’s set wider ranges, and then we will perform another iteration for smaller ranges.

Step3: Tune gamma

Now let’s tune the gamma value using the parameters already tuned above. Gamma can take various values, but I’ll check for 5 values here. You can go into more precise values.

Step4: Tune subsample and colsample\_bytree

The next step would be to try different subsample and colsample\_bytree values.

Step5: Tuning Regularization Parameters

The next step is to apply regularization to reduce overfitting. However, many people don’t use this parameter much as gamma provides a substantial way of controlling complexity. But we should always try it. I’ll tune the ‘reg\_alpha’ value here and leave it up to you to try different values of ‘reg\_lambda’.

Step6: Reducing the Learning Rate

Lastly, we should lower the learning rate and add more trees. Let’s use the cv function of XGBoost classifier to do the job again.

Describe the process of gradient boosting in XGBoost.

Ans:

Gradient boosting is a machine learning ensemble technique that uses a series of models to minimize prediction error. In XGBoost, the process of gradient boosting involves the following steps:

1. Build a base model: Predict the target variable using an initial model, F0.
2. Calculate pseudo residuals: Calculate the difference between the observed value and the predicted value, or the residual (y−F0).
3. Build a model on the residuals: Fit a new model, h1 to the residuals from the previous step.
4. Combine the models: Combine F0 h1 create a boosted version of F0 called F1. The mean squared error of F1 should be lower than that of F0.

.

What are the advantages and disadvantages of using XGBoost?

Ans:

XGBoost is a machine learning algorithm that's fast to learn, easy to tune, and can rank variables to simplify model interpretation. However, it also has some disadvantages:

* Data imbalance

XGBoost's classification may not be ideal when dealing with imbalanced data.

* Overfitting

XGBoost can overfit in high-dimensional data and may have reduced accuracy with small datasets.

* Low-dimensional data

XGBoost's efficiency and house price prediction may be limited with low-dimensional input spaces.

Here are some of XGBoost's advantages:

* Regularization

XGBoost offers adjustable L1 and L2 regularization parameters to balance model performance and generalization.

* Prediction accuracy

XGBoost's optimized decision tree model can be highly accurate, even with smaller datasets. It may outperform other models, like artificial neural networks and support vector regression.