PYTHON

Question 1.1: Write the Answer to these questions.
Note: Give at least one example for each of the questions.

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

   Key points about static and dynamic variables in Python:

   Static Variable:

   - Scope: Belongs to the class itself, not to individual instances.
   - Access: Accessed using the class name, not an object instance.
   - Use case: Useful for keeping track of a value that should be consistent across all objects of a class, like a counter

   Dynamic Variable:
   - Scope: Belongs to an individual object instance.
   - Access: Accessed through an object instance.
   - Use case: Most common type of variable, used to store unique data specific to each
   - object.

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

   The pop() method removes the specified item from the dictionary and returns value of the removed item.

```
Example
The value of the removed item is the return value of the pop() method:
student = {
  "name": "abc",
  "class": "VI",
  "section": 'A',
  "marks":56
}

x = student.pop("class")
print("data type of x:",type(x))

print(x)
print(student)
Output:
data type of x: <class 'str'>
VI
```

```
{'name': 'abc', 'section': 'A', 'marks': 56}
```

The popitem() method removes the item that was last inserted into the dictionary. The removed item is the return value of the popitem() method, as a tuple.

```
Example:
student = {
  "name": "abc",
  "class": "VI",
  "section": 'A',
  "marks":56
}
x = student.popitem()
print("data type of x:",type(x))
print(x)
print(student)

Output:
data type of x: <class 'tuple'>
('marks', 56)
{'name': 'abc', 'class': 'VI', 'section': 'A'}
```

The clear() method removes all the elements from a dictionary.

```
Example:
student = {
  "name": "abc",
  "class": "VI",
  "section": 'A',
  "marks":56
}
student.clear()
print(student)
Output:
{}
```

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

frozenset() is a built in function in Python.
The frozenset() function returns an unchangeable frozenset object (which is like a set object, only unchangeable).

```
Example
mylist = ['apple', 'banana', 'cherry']
print("old list:",mylist)
```

```
mylist[1]='Mulberry'
print("New list:",mylist)
x = frozenset(mylist)
print("frozen list:",x)
print(type(x))
```

Output:

```
old list: ['apple', 'banana', 'cherry']
New list: ['apple', 'Mulberry', 'cherry']
frozen list: frozenset({'Mulberry', 'cherry', 'apple'})
<class 'frozenset'>
```

Note:If we try to change the value of the frozenset, it results in an error like below:

```
mylist = ['apple', 'banana', 'cherry']
print("old list:",mylist)
mylist[1]='Mulberry'
print("New list:",mylist)

x = frozenset(mylist)
print("frozen list:",x)
print(type(x))
x[1]='pineapple'
Traceback (most recent call last):
```

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

	Mutable	Immutable
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, Tuples, Integer, float, bool and unicode
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.

5. What is __init__? Explain with an example.

   Python is an object-oriented language. Whenever we create the object of our class, the constructor of our class is called first. There is a method called __init__() for this task in Python. This method is called automatically whenever a new object of a class is created.
   __init__ is a magic method called **Dunder** Methods (These are special methods whose invocation happens internally, and they start and end with double underscores).

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

   Docstrings in Python are string literals that appear right after the definition of a function, method, class, or module. They are used to document what the object does. Enclosed in triple quotes, docstrings can be one line or multiline and are accessible via the .__doc__ attribute of the object.

   Example1:

```
def add(a, b):
    """Add two numbers and return the result."""
    return a + b

print(add.__doc__)  # Output: Add two numbers and return the result.
```

7. What are unit tests in Python

   Unit tests are segments of code written to test other pieces of code, typically a single function or method, that we refer to as a unit. They are a very important part of the software development process, as they help to ensure that code works as intended and catch bugs early on. Also, testing is a best practice that can save time and money by finding and fixing issues before they cause major problems.

8. What is break, continue and pass in Python

   Python supports the following control statements:
   - Break statement
   - Continue statement
   - Pass statement

   Loop control statements change execution from their normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

   Break Statement in Python
   The break statement in Python is used to terminate the loop or statement in which it is present. After that, the control will pass to the statements that are present after the break statement, if

available. If the break statement is present in the nested loop, then it terminates only those loops which contain the break statement.

Continue Statement in Python

Continue is also a loop control statement just like the break statement. continue statement is opposite to that of the break statement, instead of terminating the loop, it forces to execute the next iteration of the loop. As the name suggests the continue statement forces the loop to continue or execute the next iteration. When the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped and the next iteration of the loop will begin.

Pass Statement in Python

As the name suggests pass statement simply does nothing. The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute. It is like a null operation, as nothing will happen if it is executed. Pass statements can also be used for writing empty loops. Pass is also used for empty control statements, functions, and classes

9. What is the use of self in Python

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

A Class in Python has three types of access modifiers:

- Public/ Global Access Modifier: Public methods and fields can be accessed directly by any class.
- Protected Access Modifier: Protected methods and fields can be accessed within the same class it is declared and its subclass. a single underscore "_" is used to describe a protected data member or method of the class.
- Private Access Modifier: Private methods and fields can be only accessed within the same class it is declared. Private access modifier is the most secure access modifier. Data members of a class are declared private by adding a double underscore '__' symbol before the data member of that class.

11. What are modules and packages in Python

The module is a simple Python file that contains collections of functions and global variables and with having a .py extension file. It is an executable file.
Examples of modules:

1. Datetime
2. Regex
3. Random etc.

The package is a simple directory having collections of modules. This directory contains Python modules and also having __init__.py file by which the interpreter interprets it as a Package. The package is simply a namespace. The package also contains sub-packages inside it.
Examples of Packages:

1. Numpy
2. Pandas

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

A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item. Just as strings are defined as characters between quotes, lists are defined by having values between square brackets [ ].
Ex: L=[1,'Reena',23,45,67,78]

A tuple in Python is an immutable ordered collection of elements. Tuples are similar to lists, but unlike lists, they cannot be changed after their creation (i.e., they are immutable). Tuples can hold elements of different data types.

The main characteristics of tuples are being ordered, heterogeneous and immutable. A tuple is created by placing all the items inside parentheses (), separated by commas. A tuple can have any number of items and they can be of different data types.

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

14. What are Dict and List comprehensions

Dictionary Comprehension is an elegant and concise way to create dictionaries.
Example:

```
square_dict = {num: num*num for num in range(1, 11)}
print(square_dict)
```

List comprehension is a way to create lists using a concise syntax. It allows us to generate a new list by applying an expression to each item in an existing iterable (such as a list or range). This helps us to write cleaner, more readable code compared to traditional looping techniques.
For example, if we have a list of integers and want to create a new list containing the square of each element, we can easily achieve this using list comprehension.

Example:

```
a = [2,3,4,5]
res = [val ** 2 for val in a]
print(res)
```

Output

```
[4, 9, 16, 25]
```

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

Decorators are one of the most helpful and powerful tools of Python. These are used to modify the behavior of the function. Decorators provide the flexibility to wrap another function to expand the working of wrapped function, without permanently modifying it.

In Decorators, functions are passed as an argument into another function and then called inside the wrapper function.

16. How is memory managed in Python

Python's memory management system automatically allocates and manages memory for programs to run efficiently. The Python memory manager handles this process by using a private heap to store all Python objects and data structures.

Memory management techniques

- **Dynamic allocation**

  Memory is allocated as the program runs, allowing the program to reuse and release memory.

- **Garbage collection**

  The Python memory manager frees memory that's no longer needed by identifying and removing unreachable objects.

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

Python Lambda Functions are anonymous functions means that the function is without a name. As we already know the def keyword is used to define a normal function in Python. Similarly, the lambda keyword is used to define an anonymous function in Python.

```
s1 = 'Python Programming'
s2 = lambda func: func.upper()
print(s2(s1))
```

Output

```
PYTHON PROGRAMMING
```

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

split() is used to split a string
join() is used to join a string.
These functions allow us to easily break a string into smaller parts and then reassemble those parts into a new string.

```
# Split into words
```

```
words = a.split()
print(words)
 # Join with a hyphen
b = "-".join(words)
print(b )
```

```
Output:
['Hello,', 'how', 'are', 'you?']
Hello,-how-are-you?
```

19. What are iterators, iterable & generators in Python

An iterator in Python is an object that holds a sequence of values and provide sequential traversal through a collection of items such as lists, tuples and dictionaries. The Python iterators object is initialized using the iter() method. It uses the next() method for iteration.

Iterables are objects that can return an iterator. These include built-in data structures like lists, dictionaries, and sets. Essentially, an iterable is anything you can loop over using a for loop. An iterable implements the __iter__() method, which is expected to return an iterator object.

A generator function is a special type of function that returns an iterator object. Instead of using return to send back a single value, generator functions use yield to produce a series of results over time. This allows the function to generate values and pause its execution after each yield, maintaining its state between iterations.

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

range()	**xrange()**
Returns a list of integers.	Returns a generator object.
Execution speed is slower	Execution speed is faster.
Takes more memory as it keeps the entire list of elements in memory.	Takes less memory as it keeps only one element at a time in memory.
All arithmetic operations can be performed as it returns a list.	Such operations cannot be performed on xrange().

21. Pillars of Oops.

There are four main pillars in object-oriented programming (OOP). They are

Encapsulation
Inheritance
Polymorphism
Abstraction

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

Python issubclass() is a built-in function used to check if a class is a subclass of another class or not. This function returns True if the given class is the subclass of the given class else it returns False. We can determine class inheritance in Python by using issubclass().

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

Output:

```
Float is the subclass of str: False
```

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

Inheritance in Python is a concept in which the child class acquires the properties and can access all the data members and methods defined in the parent class. A child class can also provide its specific implementation to the methods of the parent class.

Inheritance is an important aspect of the object-oriented programming. It provides code reusability to the program because we can use an existing class to create a new class instead of creating it from scratch. All properties, data members and methods can be re-used.

Types of inheritance:
- Single inheritance
- Multiple inheritance
- Multi-level inheritance
- Hierarchical inheritance

Single Inheritance: Single inheritance is the simplest type of inheritance where a class inherits from a single superclass.

```
## Parent Class
class Animal:
   def speak(self):
      print("  Animal Speaking")

##Child class 'Dog' inherits the base class 'Animal'
class Dog(Animal):
   def bark(self):
      print("  Dog Barking")

d = Dog()
d.bark()
d.speak()
```

Multiple Inheritance: Multiple inheritance is when a class can inherit from more than one parent class.

```
## Parent Class 1
```

```
class Sum:

    def Add(self,a,b):
        return a+b

## Parent Class 2
class Product:
    def Multiply(self,a,b):
        return a*b

## Child Class inherits from both parent classes 'Sum' and 'Product'
class MathOperations(Sum,Product):
    def Divide(self,a,b):
        return a/b
```

Multilevel Inheritance: Multilevel inheritance refers to a scenario where a subclass is derived from a derived class.

```
class Grandparent:
    def greet(self):
        print('Hello from the Grandparent class!')

class Parent(Grandparent):
    pass

class Child(Parent):
    pass
```

Hierarchical Inheritance: Hierarchical inheritance is when one class serves as a superclass (base class) for more than one subclass.

```
class Parent:
    def greet(self):
        print('Hello from the Parent class!')

class Child1(Parent):
    pass

class Child2(Parent):
    pass
```

24. What is encapsulation? Explain it with an example

Encapsulation in Python describes the concept of bundling data and methods within a single unit. A class is an example of encapsulation as it binds all the data members (instance variables) and methods into a single unit.

Encapsulation is one of the key concepts of object-oriented programming. It is used to restrict access to methods and variables directly and can prevent the accidental modification of data. To prevent accidental change, an object’s variable can only be changed by an object’s method. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class.

```
class Student:
  def __init__(self, name, age):

    # private member
    self.name = name
    self.__age = age

  # getter method
  def get_age(self):
    return self.__age

  # setter method
  def set_age(self, age):
    self.__age = age

## Driver Code
stud = Student('Vijay', 50)

# Invoke getter method to retrieve the age
print('Name:', stud.name, stud.get_age())

# Invoke setter method to change the age with the new value
stud.set_age(60)

# Invoke getter method again to retrieving the new age set
print('Name:', stud.name, stud.get_age())
```

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

Polymorphism is a foundational concept in programming that allows entities like methods or operators to behave differently based on the type of data they are handling.

Polymorphism is one of the pillars of object-oriented programming. Polymorphism gives a program the ability to redefine methods for derived classes.

In OOP, polymorphism allows methods in different classes to share the same name but perform distinct tasks. This is achieved through inheritance and interface design. Polymorphism complements other OOP principles like inheritance (sharing behavior) and encapsulation (hiding complexity) to create robust and modular applications.

```
class Parent():

    # Constructor
    def __init__(self):
        self.value = "Inside Parent"

    # Parent's show method
    def show(self):
        print(self.value)

# Defining child class
class Child(Parent):

    # Constructor
    def __init__(self):
        super().__init__()  # Call parent constructor
        self.value = "Inside Child"

    # Child's show method
    def show(self):
        print(self.value)

# Driver's code
obj1 = Parent()
obj2 = Child()

obj1.show()  # Prints "Inside Parent"
obj2.show()  # Prints "Inside Child"
```

50. Machine Learning

1. What is the difference between Series & Dataframes.
   In Pandas,
   Series is a 1-dimensional array holding column of a DataFrame.
   DataFrame is a 2-dimensional array that can hold a dataset in the form of a table.

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

3. Difference between loc and iloc.
   loc is used to locate a row in pandas DataFrame using labels.
   iloc is used to locate a row in DataFrame using integer location indexing. This can be used when we don’t know the labels of the rows.

4. What is the difference between supervised and unsupervised learning?

   In supervised learning, the machine is trained on a set of labeled data, which means the input data is paired with the desired output. The machine then learns to predict the output for new input data. Supervised learning is often used for tasks such as classification, regression, and object detection.

   In unsupervised learning, the machine is trained on a set of unlabeled data, which means that the input data is not paired with the desired output. The machine then learns to find patterns and relationships in the data. Unsupervised learning is often used for tasks such as clustering, dimensionality reduction, and anomaly detection.

5. Explain the bias-variance tradeoff.

   The bias–variance tradeoff is a central problem in supervised learning. Ideally, one wants to choose a more generalized model where there is low bias and low variance. Unfortunately, it is typically impossible to do both simultaneously. Low-bias results in overfitting the model, i.e testing error is high and High-bias results in underfitting the model i.e training error is high.

   In statistics and machine learning, the bias-variance tradeoff describes the relationship between a model's complexity, the accuracy of its predictions and how well it can make predictions on previously unseen data that were not used to train the model.

6. What are precision and recall? How are they different from accuracy?
   Precision and recall are derived from the confusion matrix, the metrics used to measure the performance of classification models.

   Precision is a measure of how accurate a model's positive predictions are.

   Precision = TP / TP+FP

   Recall (Also known as Sensitivity) measures how good the model is at predicting positives. In other words, of all the positive cases, what percentage are predicted positive?

   Recall = TP / TP+FN

   Accuracy is used to measure the performance of the model. It measures how often the model is correct.

   Accuracy = TP+TN / TP+TN+FP+FN

7. What is overfitting and how can it be prevented?
   Overfitting in machine learning is a behavior of the model where the model performs well on the training data but performs badly on the unseen data or testing data.

   Some of the ways to prevent overfitting are -
   - Regularization - is a collection of training/optimization techniques that seek to reduce overfitting.
   - Feature Selection - Selecting the relevant features,
   - Cross-Validation - Experimenting with different arrangements of the same data to build different models of the same algorithm.
   - Data Augmentation - Data augmentation is a machine learning technique that changes the sample data slightly every time the model processes it.
   - Ensembling - is a machine learning technique that works by combining predictions from two or more separate models.

8. Explain the concept of cross-validation.
   Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds, using one of these folds as a validation set, and training the model on the remaining folds. This process is repeated multiple times, each time using a different fold as the validation set. Finally, the results from each validation step are averaged to produce a more robust estimate of the model's performance. Cross validation is an important step in the machine learning process and helps to ensure that the model selected for deployment is robust and generalizes well to new data.

9. What is the difference between a classification and a regression problem?
   Classification problem involves prediction of discrete labels or categories.
   Regression problem involves prediction of continuous numeric values.

10. Explain the concept of ensemble learning.

In the real world, a single model may not give the desired prediction results. Instead, Ensemble learning technique is used. Ensemble learning is a machine learning technique that combines the predictions from multiple individual models to obtain a better predictive performance than any single model.

In this, several individual base models are fitted to learn from the same data and produce an aggregation of output based on which a final decision is taken. These base models can be machine learning algorithms such as decision trees (mostly used), linear models, support vector machines (SVM) or any other model that is capable of making predictions.

11. What is gradient descent and how does it work?
    Gradient Descent is a fundamental optimization algorithm in machine learning used to minimize the cost or loss function during model training.

    - It iteratively adjusts model parameters by moving in the direction of the steepest decrease in the cost function.
    - The algorithm calculates gradients, representing the partial derivatives of the cost function concerning each parameter.

    These gradients guide the updates, ensuring convergence towards the optimal parameter values that yield the lowest possible cost.

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

    Batch gradient descent computes the gradient using the entire training dataset in each iteration. Stochastic gradient descent, computes the gradient using only a single training example or a small subset of examples in each iteration.

    Batch gradient descent takes longer to converge since it computes the gradient using the entire training dataset in each iteration. Stochastic gradient descent, on the other hand, can converge faster since it updates the model parameters after processing each example, which can lead to faster convergence.

    Batch gradient descent is more accurate since it computes the gradient using the entire training dataset. Stochastic gradient descent, on the other hand, can be less accurate since it computes the gradient using a subset of examples, which can introduce more noise and variance in the gradient estimate.

    Batch gradient descent requires more computation and memory since it needs to process the entire training dataset in each iteration. Stochastic gradient descent, on the other hand, requires less computation and memory since it only needs to process a single example or a small subset of examples in each iteration.

Stochastic gradient descent is more suitable for optimizing non-convex functions since it can escape local minima and find the global minimum. Batch gradient descent, on the other hand, can get stuck in local minima.

13. What is the curse of dimensionality in machine learning?

    The curse of dimensionality is the phenomenon where the efficiency and effectiveness of algorithms deteriorate as the dimensionality of the data increases. In high-dimensional spaces, data points become sparse, making it challenging to learn meaningful patterns or relationships due to the vast amount of data required to adequately sample the space.

14. Explain the difference between L1 and L2 regularization.

    Both L1 and L2 regularization methods are used to prevent overfitting of the machine learning model.

    L1 regularization introduces a penalty term into the model's loss function based on the absolute values of the model's parameters. The size of a penalty term is controlled by the hyperparameter lambda, which regulates the L1 regularization's regularization strength. As lambda rises, more parameters will be lowered to zero, improving regularization.

    On the other hand, L2 regularization introduces a penalty term into the model's loss function based on the squares of the model's parameters. The goal of L2 regularization is to keep the model's parameter sizes short and prevent oversizing. A hyperparameter called lambda that controls the regularization's intensity also controls the size of the penalty term. The parameters will be smaller and the regularization will be stronger the greater the lambda.

15. What is a confusion matrix and how is it used?

    Confusion matrix is a matrix (table with rows and columns) that evaluates the performance of a classification model.
    With the help of the confusion matrix, we can calculate different parameters of the model such as Accuracy, Precision, Sensitivity(Recall), Specificity, and the F-score.

    This table is also used to assess what errors (Type I and Type II) were made by the model. The rows represent the actual classes the outcomes should have been. While the columns represent the predictions that the model has made. Using this table, it is easy to see which predictions are wrong.

16. Define AUC-ROC curve.

L1 Regularization (Lasso)	L2 Regularization (Ridge)
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.

The AUC-ROC curve, or Area Under the Receiver Operating Characteristic curve, is commonly used in machine learning to assess the ability of a binary classification model to distinguish between two classes, typically the positive class and the negative class.

ROC stands for Receiver Operating Characteristics curve is the graphical representation of the effectiveness of the binary classification model. It plots the true positive rate (TPR) vs the false positive rate (FPR) at different classification thresholds.

AUC stands for the Area Under the Curve, curve represents the area under the ROC curve.

As both TPR and FPR range between 0 to 1, So, the area will always lie between 0 and 1, and A greater value of AUC denotes better model performance.

17. Explain the k-nearest neighbors algorithm.

The k-nearest neighbors (KNN) algorithm is a supervised machine learning method used for both classification and regression problems. It operates on the principle of similarity, where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.

Steps in algorithm:

- Initialize the optimal k value - i.e the number of nearest neighbors to consider.
- Calculate the distance of the new datapoint w.r.t all data points. The distance is calculated using Euclidean or Manhattan distance.

- Sort the distance.
- Based on the k value, find the k-nearest neighbors of the new data point.
- For classification, the class or value of the data point is determined by the majority vote of the k-nearest neighbors.
  For regression, the class or value of the data point is determined by the average of the k-nearest neighbors.

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

    The basic concept of SVM lies in finding the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible.

    The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane and so on. It becomes difficult to imagine when the number of features exceeds three.

19. How does the kernel trick work in SVM?

    The kernel trick enables SVMs to solve non-linear classification problems by implicitly mapping the input data to a higher-dimensional feature space.

    The kernel trick relies on the inner products of vectors. For SVMs, the decision function is based on the dot products of vectors within the input space. Kernel functions replace these dot products with a nonlinear function that computes a dot product in a higher-dimensional space. Importantly, the computation of this dot product via the kernel function does not require explicit knowledge of the coordinates in the higher space, thus saving computational resources and time.

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

    Different types of kernels used in SVM and when to use each are -
    1. Linear kernel - If the data is distinguishable, we can use linear kernel
    2. Polynomial kernel - Maps inputs into a polynomial feature space, enhancing the classifier’s ability to capture interactions between features.
    3. Radial basis function (rbf) kernel - If the data is the form of radial function.
    4. Sigmoid kernel - is used for binary classification problems.

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

    The hyperplane in SVM is a decision boundary that divides the input space into two or more regions, each corresponding to a different class or output label.
    In a 2D space, a hyperplane is a straight line that divides the space into two halves.

In a 3D space, however, a hyperplane is a plane that divides the space into two halves. Meanwhile in higher-dimensional spaces, a hyperplane is a subspace of one dimension less than the input space.

In support vector machines (SVMs), the maximum margin hyperplane is found by solving a convex optimization problem that seeks to maximize the margin while also ensuring that all data points are classified correctly.

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

    Pros:
    - SVM classifiers perform well in high-dimensional space and have excellent accuracy. SVM classifiers require less memory because they only use a portion of the training data.
    - SVM performs reasonably well when there is a large gap between classes.
    - The kernel trick is the real strength of SVM. With an appropriate kernel function, we can solve any complex problem.
    - SVM uses memory effectively.

    Cons:
    - SVM requires a long training time, as a result, it is not practical for large datasets.
    - The inability of SVM classifiers to handle overlapping classes is another drawback.
    - When the data set contains more noise, such as overlapping target classes, SVM does not perform as well.
    - The SVM will perform poorly when the number of features for each data point is greater than the number of training data samples.

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

    In hard margin SVM, all data points are correctly classified. There is no overlapping of data points.

    In soft margin, some data points are allowed to be misclassified. This is represented as C - the number of data points ready to be misclassified so that the best fit hyperplane is not overfitting. C becomes a regularization parameter.

24. Describe the process of constructing a decision tree.

    Process of constructing a decision tree involves the following steps:
    Step 1: Recursive binary splitting/partitioning data into smaller subset based on the values of different features.
    Step 2: Select the best feature to split the data at each internal node, based on certain criteria such as Information gain or Gini impurity.
    Step 3: Apply the split.

Step 4: Repeat the process for the subset obtained.
Step 5: Continue the process until every node is a pure node/stopping criteria is reached.
Step 6: Majority value in the leaf node will be the prediction.

25. Describe the working principle of a decision tree.

    Decision trees in machine learning are used for non-linear data. Constructing a decision tree starts at the root node. Data is split into subsets based on a criteria such as Information gain or Gini impurity. At every node, a decision is being made to split the data further until a pure node is reached.

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

    Information gain is a measure used to determine which feature should be used to split the data at each internal node of the decision tree. It is a measure to define the degree of randomness/disorganization in a system. Whichever feature has maximum
    Information Gain, that feature will be used to split.

    Information Gain is calculated based on Entropy which is a measure of impurity.
    IG = Entropy(Parent) - Entropy(child combined)

27. Explain Gini impurity and its role in decision trees.

    Gini impurity is one of the impurity measures used to split the dataset in a decision tree. It measures inequality or impurity of a distribution of the data.

    - It is calculated summing the squared probabilities of each outcome in a distribution and subtracting the result from 1.
    - A lower Gini Index indicates a more homogeneous or pure distribution, while a higher Gini Index indicates a more heterogeneous or impure distribution.

    In decision trees, the Gini Index is used to evaluate the quality of a split by measuring the difference between the impurity of the parent node and the weighted impurity of the child nodes.

28. What are the advantages and disadvantages of decision trees?

    Advantages of decision trees:
    - Easy to understand and interpret.
    - Captures non-linear relationships.
    - Handles both numerical and categorical data without requiring extensive preprocessing.
    - Provides insights into feature importance for decision-making.
    - Handles missing values and outliers without significant impact.
    - Applicable to both classification and regression tasks.

Disadvantages of decision trees:

- Prone to overfitting
- Sensitivity to small changes in data, limited generalization if training data is not representative.
- Decision Tree classifiers can be biased if the training data is highly dominated by certain classes.
- Classification and Regression Tree (CART) follows a greedy algorithm that finds only locally optimal solutions at each node in the tree.
- Computationally expensive on large datasets.
- Complex calculations on large datasets.

29. How do random forests improve upon decision trees?

    Decision Tree is a greedy algorithm, due to the nature of its greediness, it often leads to overfitting and hence model leads to high variance.

    Using random forest which is a bagging technique, we can reduce this variance. In a random forest, all base learners are decision trees. Each base learner in a random forest is trained with a random subset of rows and features, so each subset trains a different model. One feature which might be causing overfitting in one tree may not cause overfitting in another tree and hence overall, a random forest reduces the variance. This way, random forests improve the performance compared to a single model decision tree.

30. How does a random forest algorithm work?

    Random Forest works in two-phase first is to create the random forest by combining N decision trees, and second is to make predictions for each tree created in the first phase.

    Algorithm works with the following steps -
    Step1: Select random K data points from the training set.
    Step2: Build the decision trees associated with the selected data points (subsets).
    Step3: Choose the number N for decision trees that need to be built.
    Step4: Repeat step1 & step2.
    Step5: For new data points, find the predictions of each decision tree and assign the
    new data points to the category that wins the majority votes.

    For classification tasks, the final prediction is determined by the mode (most frequent prediction) across all the trees. In regression tasks, the average of the individual tree predictions is taken.

31. What is bootstrapping in the context of random forests?

    Bootstrap in a random forest means that instead of training the model on all the observations, each tree of the random forest is trained on a subset of the observations. This method describes creating multiple samples from the original dataset, allowing instances to be sampled with replacement. This results in different subsets of data for each decision tree, introducing variability in the training process and making the model more robust.

32. Explain the concept of feature importance in random forests.

    The importance of features in Random Forest models reveals how each feature contributes to the accuracy of the model. Features that are ranked highly have a significant influence on the model's decision-making, improving its performance.

    Feature importance has the following advantages on the model training:
    - Enhanced model performance: By identifying the most influential features, we can prioritize them during model training, leading to more accurate predictions.
    - Faster training times: Focusing on the most relevant features streamlines the training process, saving time and computational resources.
    - Reduced overfitting: By focusing on important features, we can prevent the model from becoming overly reliant on specific data points.

    Hence features are selected based on Information Gain in each decision tree in random forest.

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

    Below are the list of most important hyperparameters of a random forest that can be tuned to get a better trained model and improved prediction from the model.

    max_depth: The maximum depth of the tree - meaning the longest path between the root node and the leaf node.
    n_estimators: This is the number of trees in the forest.
    criterion: The function to measure the quality of a split.
    max_sample: This determines the fraction of the original dataset that is given to any individual tree.
    max_features: This is the number of features to consider when looking for the best split.

34. Describe the logistic regression model and its assumptions.

    Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation. The model delivers a binary outcome limited to two possible outcomes: yes/no, 0/1, or true/false.

    Logistic regression uses a sigmoid function that takes input as independent variables and produces a probability value between 0 and 1.

For example, we have two classes Class 0 and Class 1 if the value of the logistic function for an input is greater than 0.5 (threshold value) then it belongs to Class 1 otherwise it belongs to Class 0.

Assumptions:

1. Binary logistic regression requires the dependent variable to be binary and ordinal logistic regression requires the dependent variable to be ordinal.
2. Logistic regression requires the observations to be independent of each other. In other words, the observations should not come from repeated measurements or matched data.
3. Logistic regression requires there to be little or no multicollinearity among the independent variables.
4. Logistic regression assumes linearity of independent variables and log odds of the dependent variable. Although this analysis does not require the dependent and independent variables to be related linearly, it requires that the independent variables are linearly related to the log odds of the dependent variable.
5. Logistic regression typically requires a large sample size.

35. How does logistic regression handle binary classification problems?

Logistics regression uses the sigmoid function to return the probability of a label.
Instead of fitting a regression line, we fit an “S” shaped logistic function, which predicts two maximum values (0 or 1). It can be either Yes or No, True or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

36. What is the sigmoid function and how is it used in logistic regression?

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the “S” form.
- The S-form curve is called the Sigmoid function or the logistic function.

In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

37. Explain the concept of the cost function in logistic regression.

Cost function in logistic regression is referred to as log loss function.

38. How can logistic regression be extended to handle multiclass classification?

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

    L2 regularization induces convexity in the cost function by adding a quadratic penalty term, which makes the cost function smoother and easier to optimize. L1 regularization, on the other hand, does not induce convexity but can still lead to sparse solutions.

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

    XGBoost is a type of boosting method which uses a sequence of weak learner decision trees to make a prediction. XGBoost is an optimized version of the gradient boosting algorithm.

    One of the main advantages of XGBoost is its ability to find the best balance between making accurate predictions and keeping things simple. It does this by using regularized objective functions.

    Following are the other reasons that make XGBoost a more efficient algorithm compared to other boosting techniques.
    - XGBoost has its own library.
    - Supports parallel computation to make it a fast algorithm
    - Good with speed and accuracy
    - Works very well with large datasets.

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

    Boosting is an ensemble learning method that involves training homogenous (decision trees) weak learners sequentially such that a base model depends on the previously fitted base models. All these base learners are then combined in a very adaptive way to obtain an ensemble model.

    In boosting, the ensemble model is the weighted sum of all constituent base learners.

42. How does XGBoost handle missing values?

    XGBoost can handle missing values internally. When it encounters a missing value at a node, it tries both left and right splits and learns the path leading to a higher loss for each node. This process is repeated when working on the testing data.

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

    XGBoost hyperparameters are divided as -
    General parameters, **Booster parameters** and **learning task parameters**. Some of the key hyperparameters are listed below:

eta - It is analogous to learning rate in GBM. It is the step size shrinkage used in updates to prevent overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative. It makes the model more robust by shrinking the weights on each step. range : [0,1]. Typical final values : 0.01-0.2.

gamma - A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split. It makes the algorithm conservative. The values can vary depending on the loss function and should be tuned. The larger gamma is, the more conservative the algorithm will be.

max_depth - Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. XGBoost aggressively consumes memory when training a deep tree. Should be tuned using CV. Typical values: 3-10.

lambda - L2 regularization term on weights (analogous to Ridge regression). This is used to handle the regularization part of XGBoost. Increasing this value will make the model more conservative.

alpha - L1 regularization term on weights (analogous to Lasso regression). It can be used in case of very high dimensionality so that the algorithm runs faster when implemented. Increasing this value will make the model more conservative.

tree_method - The tree construction algorithm used in XGBoost. XGBoost supports approx, hist and gpu_hist for distributed training. Experimental support for external memory is available for approx and gpu_hist.

scale_pos_weight - It controls the balance of positive and negative weights. It is useful for imbalanced classes. A value greater than 0 should be used in case of high class imbalance as it helps in faster convergence. A typical value to consider: sum(negative instances) / sum(positive instances).

44. Describe the process of gradient boosting in XGBoost.

    Gradient Boosting is a powerful machine learning technique that is popular for its ability to create highly accurate prediction models. This technique builds models in stages, focusing on errors from previous stages. At its core, Gradient Boosting creates a prediction model as an ensemble of weak prediction models, which individually make very few assumptions about the data and are typically simple decision trees. When a decision tree is the weak learner, the resulting algorithm is known as a gradient-boosted tree.

    Despite their name, weak learners are the strength of Gradient Boosting. These simple models are leveraged by the algorithm and combined to create a single, strong predictive model.

45. What are the advantages and disadvantages of using XGBoost?

Advantages:

High accuracy: XGBoost is known for its high accuracy, making it a popular choice for machine learning tasks that require high precision.

Speed: XGBoost is designed to be fast and efficient, even for large datasets. It is optimized for both single- and multi-core processing, making it an excellent choice for tasks that require fast predictions.

Regularization: XGBoost includes regularization techniques that help to prevent overfitting. It uses a combination of L1 and L2 regularization to reduce the complexity of the model, resulting in more robust and accurate predictions.

Flexibility: XGBoost is a flexible algorithm that can be used for a variety of machine-learning tasks, including classification, regression, and ranking.

Disadvantages:

Complexity: XGBoost is a complex algorithm that requires some degree of technical expertise to implement and optimize effectively. It can be challenging to configure and tune the many hyperparameters that are involved, which can make it time-consuming to work with.

Overfitting: While XGBoost includes regularization techniques to prevent overfitting, it is still possible for the algorithm to overfit the training data. This can lead to inaccurate predictions of new data.

Memory usage: XGBoost can be memory-intensive, especially for large datasets. This can make it challenging to run on computers with limited memory, leading to slower performance.

Lack of transparency: XGBoost has often been considered a "black box" algorithm, which means that it can be difficult to interpret and understand how it arrives at its predictions. This can make it challenging to troubleshoot and fine-tune.