Q 1.1 Write the answer to these questions.

Note: Give at least one example of the questions.

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

Ans :-

**Static Variable** – Variables are shared among all instances or class.

Eg. When obj1 and obj2 are created, static\_var becomes 2, and this value is the same for both instances.

**Dynamic Variable** – variables are specific to each instance.

Eg. obj1.dynamic\_var is initially 10, and obj2.dynamic\_var is 20.

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

Ans :-

**Pop** - This method is used to remove a key-value pair from the dictionary based on the specified key and return the value associated with that key.

Eg. Dictionary.pop(‘key’)

**Popitem()** – Removes and returns the last inserted key-value pair as a tuple from the dictionary.

Eg. Dictionary.popitem()

**Clear()** - removes all key-value pairs from the dictionary.

Eg. Dictionary.clear()

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

Ans :-

It is an immutable version of python set object. The frozenset() function returns a unchangeable frozenset object(which is like a set object, only unchangeable)

Eg. Frozenset(iterable)

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

Ans:-

**Mutable data types** – we can change their values after creation.

Eg. List, dictionaries, sets

**Immutable data types** – We cannot change their values after creation.

Eg. Tuples, strings.

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

Ans.

**\_\_init\_\_** is a special method in python, referred to as the ‘constructor’ of a class. It is automatically called when a new instance of a class is created.

The first parameter of ‘\_\_init\_\_’ is always ‘self’, which refers to the instance being created.

Eg. class Person: def \_\_init\_\_(self, name, age): # Initialize instance attributes self.name = name self.age = age

vi. What is docstring in python ? Explain with an example.

Ans.

**Docstring**: This stands for documentation string, which are not just comments. We enclose the docstings in triple quotation marks.

Eg. ‘’’ anything ‘’’

vii. What are unit tests in python ?

Ans.

Unit tests in a python are a fundamental practice for ensuring the correctness of individual components of your code. The unittest framework is a powerful tool provided by python to create and run these tests, helping to maintain code quality and realibility throughout the development process.

Advantages – 1. Easy bug detection 2. Refactoring Confidence 3. Documentation

viii. What is break, continue and pass in python?

Ans.

**Break** – The loop is terminated when criteria is fulfilled and control is passed to the subsequent statement.

**Continue** – When a specified criteria is fulfilled, the control is moved to the start of the loop, allowing some parts of loop currently in execution to be skipped.

**Pass** – This is used when code block is syntactically correct but don’t want to run it. When it is running nothing takes place.

viii. What is the use of self in python ?

Ans.

Self is class instance. This is explicitly supplied as the initial argument in python.

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

Eg. class MyClass:

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

self.name = name

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

Ans.

**Global Attributes** – This are defined outside of any class or function and can be accessed throughout the entire module. Global variables are accessible from any part of code within the module.

Eg. global\_var = "I am global"

class MyClass:

def print\_global(self):

print(global\_var)

# Accessing the global variable within a class method

obj = MyClass()

obj.print\_global()

**Protected Attributes** – This are indicated by a single underscore(‘\_’) prefix. These attributes are intended to be accessed within the class and its subclasses but not from outside the class.

Eg. class MyClass:

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

self.\_name = name

# Protected attribute

self.age = age

**Private Attributes** – This are indicated by double underscore(‘\_\_’) prefix. These attributes are intended to be accessed only within class where they are defined and are not accessible directly from outside the class.

Eg. class MyClass:

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

self.\_\_name = name

# Private attribute

self.age = age

obj = MyClass("Bob", 40)

print(obj.\_Myclass\_\_name)

x. What are modules and packages in python ?

Ans.

**Module** – A Single python file(‘.py’) containining functions, classes or variables that can be imported and reused in other scripts.

**Package** – A directory containing multiple modules and an ‘\_\_init\_\_’.py file. Packages allow for better organization and structure of larger projects, enabling to grp related mdoules together.

Xi. What are list and tuples? What is the key difference between the two ?

Ans.

**List** -list is a mutable, ordered collection of items in python. We can add, remove or change items after the list has been created. List are defined using [].

**Tuples** – A tuple is an immutable, ordered collection of items in python. once tuple is created, its elements cannot be changes, added or removed. Tuples are defined using ().

Key difference :

1. Mutability – lists are mutable, while tuples are immutable.
2. Syntax - Lists defined using square brackets ‘[]’, while tuples defined using parantheses ‘()’.
3. Performance – lists are slower as compare to tuples because of its mutability.
4. Use cases – lists used when we need a collection of items that can be modified, tuples used when we need a collection of items that should not be modified.

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

Ans.

**Interpreted Language** – It is a type of programming language where most of the instructions are executed directly by an interpreter rather than being compiled into machine code.

Execution is generally slower compared to compiled languages. Easier to test and debug since code can be executed immediately after writing.

Eg. Python, ruby, php, js.

**Dynamically typed language** – It is a type of programming language where variable types are determined at runtime rather than at compile time.

More flexible and can lead to quicker development since you don’t need to specify types. Type-related errors are generally discovered during program execution rather than an compile time.

5 differences between both -

1. **Execution method** – In Interpreted language, code is executed line by line by an interpreter. Eg. Python. In Dynamically typed language, type checking is done at runtime, regardless of whether the language is interpreted or compiled. Eg. Python
2. **Type checking** – In Interpreted language, type checking can be either dynamic or static, depending on the language. Eg. Python, java . In dynamically typed language, types are checked at runtime, allowing variables to change types during execution.
3. **Performance** – Interpreted language have generally slower execution speed dur to line-by-line interpretation. Eg. Python. Dynamically typed language may or may not impact performance based on whether it’s interpreted or compiled. Eg.Ruby.

Xiii. What are Dict and List comprehensions ?

Ans.

**List Comprehension** – It creates lists in a concise and readable way, using a single line of code with optional filtering. it uses square brackets []. It is used when we need to create lists from existing iterables with optional filtering and transformation.

**Dict Comprehension** – It creates dictionaries in a similar concise manner, generating both keys and values based on a given expression and optional filtering. It uses curely braces {}. It is used when we need to create dictionaries, where we generate both keys and values based on some logic.

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

Ans.

**Decorators** in python are essentially functions that add functionality to an existing function in python without changing the structure of function itself.

Syntax - @dcorator name

Eg. def my\_decorator(func):

def wrapper():

print("Something is happening before the function is called.")

func() # Call the original function

print("Something is happening after the function is called.")

return wrapper

# Apply the decorator using the @ syntax

@my\_decorator

def say\_hello():

print("Hello!")

# Call the decorated function

say\_hello()

Use cases -

1. Logging – to log information before and after function execution
2. Authentication and Authorization
3. Performance Measurement
4. Caching

Xv. How is memory managed in python ?

Ans. In python memory management is handled automatically through combination of mechanisms.

1. Automatic Memory Management: It automatically reclaims memory from objects that are no longer in user.
2. Reference Counting: Each object in python has a reference count that count that tracks how many references point to the object. When reference counts drops to zero, the memory is deallocated.
3. Cyclic Garbage Collection: Python uses a cyclic garbage collector to detect and clean up these cycles.

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

Ans.

It is a type of nameless function. This method in python can take as many parameters as we want but in a single statement.

Syntax. Lambda arg: exp

We use lambda functions because

1. Conciseness – Lambda functions provide a consise way to define simple functions inline without having to write a full function defination.
2. Convenience - Useful in scenarios where a short function is required.
3. Anonymous functions – Ideal for situations where a function is used temporarily, making it unnecessary to formally define a named function.

Xvii. Explain split() and join () functions in python ?

Ans.

**Split()** - This method divides a string into a list of substrings based on a specified delimiter or whitespace, allowing for optional limit on number of splits. If no delimiter is specified, it splits on any whitepsace by default.

Syntax – string.split(separator, maxsplit)

‘separator’ - It is optional. The delimater string on which to split. The default is whitespace.

‘maxsplit’ - It is optional. Specifies the max. No. Of splits. The default is –1, which means ‘all occurrences’.

Eg. # Splitting a string into a list of words

text = "Python is great"

words = text.split()

print(words)

# Output: ['Python', 'is', 'great']

# Splitting a string by a specific delimiter

data = "apple,banana,cherry"

items = data.split(',')

print(items)

# Output: ['apple', 'banana', 'cherry']

**join()** - This method is used to concatenate a list of strings into a single string, with a specified separator between each element.

Syntx – separator.join(iterable)

‘separator’ - a string that will be inserted between each element of the ‘iterable’.

‘iterable’ - An iterable containing strings to be joined.

Eg. words = ['Python', 'is', 'great']

sentence = ' '.join(words)

print(sentence)

# Output: 'Python is great'

Eg 2. # Joining a list with a specific delimiter

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

data = ','.join(items)

print(data)

# Output: 'apple,banana,cherry'

Xix. What are iterator, iterable and generators in python ?  
Ans.

* **Iterable**: An object that can return an iterator (e.g., list, tuple, string). It provides an \_\_iter\_\_() method or \_\_getitem\_\_() method.
* **Iterator**: An object that represents a stream of data and implements \_\_iter\_\_() and \_\_next\_\_() methods. It is obtained from an iterable and can be used to access elements sequentially.
* **Generator**: A type of iterator created using a function with yield statements. It generates values on the fly and does not store them, making it efficient for handling large datasets or streams of data.

XX. What is the difference between xrange and range in python ?  
Ans.

range – In python3, ‘range’ generates a sequence of numbers as an immutable sequence type. It is designed to handle large ranges efficiently. Available in python 3.x

Xrange – In python2, ‘xrange’ was used to generate a sequence of numbers in memory-efficient way. Available in python 2.x, removed in python 3.x

XXI. Pillars of oops.  
Ans.   
1. Encapsulation

2. Polymorphism

3. Abstraction

4. Incapsulation. these are main pillars of oops.

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

To check if a class is a child (subclass) of another class in Python, you can use the built-in issubclass() function. This function returns True if a class is a subclass of another class, and False otherwise.

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

**Inheritance** in python allows one class to inherit attributes and methods from another class, facilitating code reuse and establishing a hierarchical relationship between classes.

Types of Inheritance :-   
1. Single Inheritance – It occurs when a class inherits from another single class.  
Eg. class Animal:   
 def speak(self):   
 print("Animal speaks")

class Dog(Animal): # Dog inherits from Animal   
 def bark(self):   
 print("Dog barks")   
 # Usage  
 dog = Dog()   
 dog.speak() # Inherited method   
 dog.bark() # Child class method

2. Multiple Inheritance – It allows a class to inherit from more than one parent class.

3. Multilevel Inheritance – It involves a hierarchy where a class is derived from another class, which is itself derived from another class.

4. Hierarchical Inheritance – It occurs when multiple classes inherit from a single parent class.

XIV. What is encapsulation ? Explain it with an example.  
Ans.   
**Encapsulation** – It is used to restrict access to methods and variables.  
In encapsulation, the variables of class will be hidden from other classes, can be accessed only through the methods of their current class.

Eg.

class person:  
 def \_\_init\_\_(self, name):  
 self.\_\_name = name  
 def name(self):  
 print(f’the name is {self.\_\_name}’)  
dev = person(‘deva’)  
dev.name()

dev.\_person\_\_name = ‘sam’

dev.name()

XV. What is polymorphism ? Explain it with an example.  
Ans.

**Polymorphism** :- It means having many forms. Polymorphism means the same name being used for different types.

Eg. If their are 4 diff. Animals like cat, hourse, cow, dog.

Question 1. 2. Which of the following identifier names are invalid and why?  
 a) Serial\_no.  
 b) 1st\_Room  
 c) Hundred  
 d) Total\_Marks   
 e) total-Marks  
 f) Total Marks  
 g) True  
 h) \_Percentag

Ans. Below are invalid Identifier -   
 b. 1st\_Room : Invalid, Identifies cannot start with a digit.

e.. total-Marks : Invalid, most programming languages do not allow haphens (‘-’) in identifies.

f. Total Marks : Invalid, Spaces are not allowed in identifiers.

g. True: Invalid, In python ‘True’ is a reserved Keyword and cannot be used as an identifier.

Q.1.3 to 49 - This question solved in google colab.

Q50. **Machine Learning**

**i.** What is the difference between Series and Dataframes.

Ans.  
 **Series** – It is a one-dimensional array-like structure that can hold any data type(integers, floats, strings, etc).  
 Eg. A list of Numbers

**DataFrame** – It is two-dimensional, tabular data structure that can hold any data type.

Eg. A table of data.

Series is typically used for single columns of data, while DataFrame’s used for entire datasets.

ii. Crease a database name Travel\_Planner in mysql ,and create a table name bookings in that which having attributes (user\_id INT, flight\_id INT, hotel\_id INT, activity\_id INT, booking\_date DATE) .fill with some dummy value. Now you have to read he content of this able using pandas as dataframe. Show he output  
Ans.  
 CREATE DATABASE Travel\_planner;  
 USE Travel\_Planner;

CREATE TABLE bookings (  
 user\_id INT,  
 flight\_id INT,  
 hotel\_id INT,  
 activity\_id INT,  
 booking\_date DATE  
 );

INSERT INTO bookings (user\_id, flight\_id, hotel\_id, activity\_id, booking\_date)  
 VALUES

(1, 101, 201, 301, ‘2023-08-01');

iii. Difference between loc and iloc.

Ans.

**iloc (Integer Location)** :-  
 used for integer-location based indexing.

It requires integer indices to select rows and columns.

Usage – use integer positions to select data.

**loc (label location)** :-  
 used for label-based indexing.  
 It requires labels to select rows and columns.  
 Usage – use label positions to select data.

iv. what is the difference between supervised and unsupervised learning?  
Ans.  
 **Supervised Learning** : It involves training a model based on a labelled dataset. In this machine learns the relationship between inputs and output.   
 Algorithms of SL: Linear regression, Logistic regression, SVM, DT etc.

**Unsupervised Learning** : it involves training a model based on a unlabeled dataset.  
It discover hidden patterns or structure in data.

Algorithms Of USL: K-means, PCA, Hierarchical Clustering.

v. Explain the bias-variance tradeoff.  
Ans.  
 **Bias-variance tradeoff**: It describes the balance between two sources of error that affect the performance of predictive models: bias (Training error) and variance(testing error).   
 It 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 model.   
In order to reduce high bias – use some more dp’s, more feature engineering, use some other ml algorithms.

vi. What are precision and recall ? How are they different from accuracy ?  
Ans.

**Precision** : It is the ratio of correctly predicted positive observations to the total predicted positive observations.

Precision=True Positives (TP) / (True Positives(TP) + False Positives(FP))

High precision indicates a low number of false positives.

Eg. Spam detection.

**Recall** : It is ratio of correctly predicted positive observations to all observations that are actually positive.

Recall = True Positives (TP) / (True Positives(TP) + False Negative(FN))

High recall indicates a low number of false negatives.

Eg. Disease screening

Accuracy focuses on the overall correctness of the model but can be misleading in imblanced datasets. That removes with the help of precision and recall.

vii. What is overfitting and how can it be prevented ?  
Ans.   
**Overfitting** is a common problem in machine learning where a model learns the details and noise in the training data to the some extent (zero error) due to which it negatively impacts the performance of the model on new, unseen data.   
**characterics of overfitting** :  
1. Excellent Performance on training data.  
2. Poor performance on test data.  
3. High model complexity.

**How to prevent Overfitting** :

1. Cross validation   
2. Adding more training data.  
3. Ensemble Methods.  
4. More training data.

viii. Explain the concept of cross-validation.  
Ans.  
 **Cross-validation** : - It is a technique used in ML to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set, and training the model on remaining folds.  
 The goal is to obtain a more reliable estimate of a model’s performance by assessing it on multiple subsets of the data, rather than relying on a single train-test split.

Types of Cross validation

1. K-fold cross validation
2. Leave one-out cross validation
3. Stratified k-fold cross validation
4. Time series cross validation

Ix. What is the difference between a classification and a regression problem ?  
Ans.

**Classification** :  
 objective – predict categorical labels or classes.  
 Output – Discrete values eg. 0 or 1 , pass/fail , spam or not spam  
 metrics – Accuracy, Precision, recall, F1 score  
 Eg. Email Spam detection, image classification.

**Regression**:

Objective – predict continuous numerical values.  
 output – Continuous values (e.g. house price)  
 metrics – Mean Absolute Error(MAE), Mean Squared Error(MSE), r-square  
 Eg. Predicting house prices, forecasting sales.

x. Explain the concept of ensemble learning.  
Ans.   
 **Ensemble Learning** : It is a machine learning technique where multiple models are combined to improve overall performance.   
 **Key concepts**:

Combining models : Ensemble methods combine the predictions of several models to make a final decision.  
 Diversit – The base models shoud be diverse, meaning they should make different errors to complement each other.

Techniques :

* **Bagging** (Bootstrap Aggregating): Trains multiple models on different subsets of the data (e.g., Random Forests).
* **Boosting**: Trains models sequentially, each correcting the errors of the previous ones (e.g., AdaBoost, Gradient Boosting).
* **Stacking**: Combines predictions of multiple models using a meta-model to make the final prediction.

Xi. What is gradient descent and how does it work ?  
Ans.  
 **Gradient descent** is optimization algorithm which is commonly used to train machine learning models. It trains ml models by minimizing errors between predicted and actual results.   
It works by calculating the gradient(slope) of the function at the current parameters and updating them in opposite direction of the gradient. The process continues until the algorithm converges to a minimum, typically point where the loss is smallest.  
 update rule,

θnew =θold −α⋅∇θ J(θ),  
 where, θnew - updated parameters,  
 θold - current parameters,  
 α - Learning rate,  
 ∇θ J(θ) - It is the gradient of the loss function with respect to the parameters.

Learning rate is crucial. If it’s too large, the algorithm may overshoot the minimum and fail to converge. If it’s too small, convergence can be very slow. It should be optimal.

Xii. Describe the difference between batch gradient descent and stochastic gradient desent.  
Ans.

**Batch Gradient Descent** :- In this, the gradient(slope) is calculated using the entire dataset. This means that for each iteration, we sum up the errors over all data points before updating the parameters.  
 It uses entire dataset for calculation due to which it is more accurate, leading to a smoother and more stable convergence path.   
 It is slow due to large dataset. Also it requires more memory to store.  
 It leads to steady path towards convergence but may take longer to reach the minimum due to the cost of each iteration.

**Stochastic Gradient Descent** :- In this, gradient is computed using only one randomly selected data point at each iteration.   
 It is less accurate because it’s based on a single data point, leading to a noisier, more erratic convergence path.  
 It is fast also it requires much less memory since one one data point processed at a time.  
 It leads to unsteady path towards convergence, due to which it also make harder to precisely converge to the minimum.

Xiii. What is curse of dimensionality in machine learning ?  
Ans.  
 **Curse of dimensionality** :- It refers to the challenges that arise when analyzing and organizing data in high-dimensional spaces. In simply, With increase in no. Of features the performance of model decreases.

To remove curse of dimensionality there are two methods :  
 I) **Feature Selection** – correlation, VIF, p-value.  
 II) **Feature Extraction** – PCA (Dimensionality reduction technique).

Xiv. Explain the difference between L1 and L2 regularization.  
Ans.   
 **L1 Regularization(Lasso)** - It adds the sum of the absolute values of the coefficients to the loss function.

L1 = CF + λ . Summation( |slope(m)|)

Where λ - regularization parameter that controls the strength of the penalty.

Lasso removes the insignificant feature. The main aim is to only select only necessary features.

**L2 Regularization(Ridge)** - it adds the sum of the squared values of the coefficients to the loss function.  
 L2 = CF + λ.summation(slope(m))^2

Ridge reduces the overfitting(by reducing the coefficient)

xv. What is a confusion matrix and how it is used ?  
Ans.  
  **Confusion Matrix** : It used to evaluate the performance of a classification model. It shows the number of correct and incorrect predictions

**Structure** :  
 True Positives(TP) : correctly predicted positive cases.  
 False Positives(FP) : Incorrectly predicted negative cases.  
 True Negatives (TN): correctly predicted negative cases.  
 False Negative(FN) : Incorrectly predicted negative cases.

It helps to calculate key metrics like accuracy, precision, recall, and F1 score. It provides insights into where the model is making errors, allowing for better model evaluation.

XVI. Define AUC-ROC curve.  
Ans.  
 **AUC-ROC** curve is graphical representation used to evaluate the performance of a binary classification model.  
 **Components**:  
 **Roc curve** : plots the true positive rate against false positive rate at various threshold settings.  
 **Auc(area under the curve)** : measures the entire two-dimensional area underneath the ROC curve.  
 **Usage:**  
 AUC value – ranges from 0 to 1. A higher AUC indicates a better-performing model.

XVII. Explain the k-nearest neighbors algorithm.  
Ans.  
 **K-Nearest Neighbors(KNN)** : It is a simple, non-parametric classification and regression algorithm.  
 How it works:  
 1. Distance Calculation – Given a new data point, the algorithm calculates the distance between this point and all other pts. in training dataset.  
 2. Find Neighbors – identifies the ‘k’ closest data points to new point.

3. Prediction :  
 Classification: The new data point is assigned to the class that is most common among its 'k' nearest neighbors.

Regression: The value for the new data point is the average of the values of its 'k' nearest neighbors.

Advantages – Easy to understand.

XViii. Explain the basic concept of a Support vector Machine (SVM) .  
Ans.  
 **SVM(Support Vector Machine)** - It is a supervised learning algorithm used for classification and regression tasks. It works by finding optimal hyperplane that separates different classes in feature space.

**Basic Concept**:  
 1. Hyperplane – It is a decision boundary that separates different classes.

2. Margin – It is distance between the hyperplane and nearest data points from each class. These nearest points are called Support vectors.  
 3. Optimal Hyperplane – The hyperplane with the maximum margin is considered the optimal hyperplane.  
 4. Kernel trick - For non-linearly separable data, SVM uses kernel functions to transform the data into a higher-dimensional space where a linear hyperplane can be used for separation.  
  
 The data points closest to the hyperplane are crucial in defining the optimal hyperplane.

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

Ans:  
 **Hyperplane in SVM** – It is the decision boundary that separates different classes in the feature space. It divides the data into distinct classes.  
 **How is it determines** ?  
 1. Objective – The objective is to find the optimal hyperplane that maximizes the margin between the two classes. The margin is distance between the hyperplane and nearest data points from each class.

2. formula -  
 w.x + b = 0,  
 where,   
 w = normal vector  
 x = feature vector  
 b = bias term  
 3. Maximizing the margin: The SVM algorithm maximizes the margin 2/||w|| between the hyperplane and support vectors.  
 4. Support vectors : Only the points closest to the hyperplane influence its position and orientation.  
 5. Kernel Trick : If data is not linear then we use kernel trick.

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

Ans.   
 **Pros of SVM** :  
 1. It is effective when the number of features is greater than the number of samples, and they can handle high-dimensional data well.  
 2. SVM have regalarization parameters(C and the kernel) that help prevent overfitting.  
 3. SVMs perform well when there is a clear margin of separtion between classes.  
 4. Memory efficient.

**Cons of SVM**:  
 1. Svm can be slow to train, especially with large datasets because it requires solving complex optimization problem.  
 2. Sensitive to parameter selection  
 3. For very large datasets, the training time can be long, and the algorithm may struggle with efficiency.  
 4. It is less effective on noisy data.

XXI. Explain the difference between a hard margin and a soft margin.  
Ans.  
 **Hard Margin SVM** : In this, all data points be perfectly separated by the hyperplane, with no misclassification. This means that all points must lie on correct side of the margin, with no data points on the wrong side.  
Use case : It’s used when data is linearly separable.  
Limitations : Hard margin SVM is very sensitive to outliers. Even a single outlier can prevent the algorithm from finding valid hyperplane.

**Soft margin SVM** : In this, some points can be on the wrong side of the hyperplane. The algorithm aims to find a balance between maximizing the margin and minimizing classification errors.  
User case : It used when the data is not perfectly separable.

Soft Margin is not sensitive to outliers.

XXII. Describe the process of constructing a decision tree.  
Ans.  
 1. Recursive binary splitting/partioning the data into smaller subnet.  
 2. Select the feature to split [Information gain].  
 3. Apply the split  
 4. Repeat the process for the subset obtained.  
 5. Continue the process until every node is a pure node.  
 6. Majority value in the leaf node will be prediction.

XXIII. Describe the working principle of a decision tree.  
Ans.  
 **Working principle of a decision tree**   
 1. The algorithm begins at the root node, which represents the entire dataset. At each node, it selects the feature that best splits the data based on a criterion like Gini impurity or Info. Gain for classification or mean squared error(MSE) for regression.  
 2. The dataset is split into two or more homogeneous sets based on the selected feature and its values.  
 3. Each subset of the data becomes a child node of the current node.  
 4. The recursion stops at the leaf nodes, where the final decision is made. For classification, each leaf node is assigned the majority class of the data points it contains. For regression, the leaf node predicts the average of the values of the data points.

5. To make a prediction for a new data point.

XIV. What is information gain and how is it used in decision trees ?  
Ans.  
 **Information Gain** : It is a measure to define degree of randomness in a system.   
It measures how much splitting on a particular feature reduces uncertainty about the class labels.  
If randomness decreases, purity increases then we can say that Information Gain.

Entropy = -p1.log2(p1) - p2.log2(p2)  
 where p1 and p2 are proportions of the two classes.

IG = Entropy (before split) - Entropy(after split)

**Uses of IG in decision tree** :

During the construction of DT, Info. Gain is used to select the feature that best splits the data at each node. The feature with highest IG is chosen for the split because it results in the purest subsets.

XXV. Explain Gini Impurity and its role in decision trees.  
Ans.  
 **Gini Impurity** :- It is used to measure the impurity of a dataset at a node in a decision tree. Gini impurity measures how mixed the classes are in dataset.

**Gini Impurity = 1 - ((Pa)^2 + (Pb)^2)**  
 where, Pa and Pb are proportions of class A and class B.  
  
 **Role in Decision trees**:  
 Splitting Criterion : It is used to determine which feature to split the data on at each node. The feature that results in the lowest gini impurity after the split is chosen for the split.  
 Minimizing Impurity: The goal at each node is to minimize the gini impurity.

XXVI. What are the advantages and disadvantages of decision trees ?  
Ans.  
 **Advantages of Decision tree**:  
 1. Decision trees are easy to visualize, making them useful for explaining the decision making process.  
 2. They can work with both types of data numerical and categorical.  
 3. No need for feature scaling.  
 4. Missing values do not affect the process significantly.  
 5. They can be used for both classification and regression tasks.

**Disadvantages of Decision Tree**:  
 1. Prone to overfitting.  
 2. Unstable with small variation in data.  
 3. Biased with Imbalanced data.  
 4. Complex trees can be difficult to interpret.

XXVII. How do random forests improve upon decision trees ?  
Ans.   
 1. **Reduces Overfitting** :   
It build multiple decision trees on different subsets of the data and average their predictions. This ensemble approach reduces overfitting by smoothing out the predictions.  
 2. **Increases Stability** : By the averaging the results of many tree, random forests are less sensitive to small variations in the data, making the model more robust and stable compared to individual decision trees.  
 3. **Feature Randomization**: During the construction of each tree, random forests randomly select a subset of features to split on at each node.

XXVIII. How does a random forest algorithm work ?  
Ans.

A **Random Forest** algorithm works by creating an ensemble of multiple decision trees and combining their predictions to improve accuracy and robustness.

**Working principle**  
 1. Bootstrap Sampling : The algorithm generates multiple subsets of the original training data using a process called bootstrap sampling. Each subset is used to build a different decision tree.  
 2. Feature Randomization : For each split in a tree, a random subset of features is selected, and the best feature from this subset is used to split the data.  
 3. Tree construction : Each decision tree is grown to its full depth without pruning, but because of the random sampling of data and features, each tree is different.  
 4. Prediction Aggregation : For classification tasks, the prediction of all the trees are combines using majority voting.

XXIX. what is bootstrapping in the context of random forests?  
Ans.   
 **Bootstrapping** – It is technique of creating multiple subsets of the original dataset by sampling with replacement.   
 **Key points**:  
 some data points may appear multiple times in a bootstrap sample, while others may be omitted.  
 This process introduces variability among the trees, helping to reduce overfitting and improve the model's robustness.

XXX. Explain the concept of feature importance in random forests ?  
Ans.  
 **Feature Importance** : It is technique used to measure the contribution of each feature in predicting the target variable.   
 **Key concept** :   
 1. Calculation: Feature importance is typically calculated by averaging the decrease in impurity (e.g., Gini impurity) across all the trees in the forest whenever a feature is used to make a split.  
 2. Ranking : features that cause larger decreases in impurity are considered more important, and the model ranks them accordingly.  
 3. Interpretation : Features with higher importance scores are more critical to the model's predictions, while those with lower scores may have less influence.

XXXI. what are the key hyperparameters of a random forest and how do they affect the model?  
Ans.   
 Key hyperparameters of random forest are :  
1. **n\_estimators** : The no. Of trees in the forest. Increasing it generally improves model accuracy but also increases computational cost.  
2. **max\_depth** : The maximum depth of each tree. Limiting depth prevents overfitting, making the model more generalizable.  
3. **min\_samples\_split** : The minimum no. Of samples required to split a node.  
4. **min\_samples\_leaf** : The minimum number of samples required to be at a leaf node.  
5. **max\_features** : The number of features to consider when looking for the best split.  
  
XXXII. Desribe the logistic regression model and its assumptions.  
Ans.  
 **Logistic Regression** : It is used for binary classification tasks where the goal is to predict the probability that an instance belongs to a given class or not.

**Assumptions :**   
1. Independent observations : Each observation is independent of the other. Meaning there is no correlation between any input variables.  
2. Binary dependent variables : It takes the assumption that the dependent variable must be binary, meaning it can take only two values.  
3. There should be no outliers in the dataset.  
4. The relationship between the independent variables and log odds of the dependent variables should be linear.  
4. The sample size is sufficiently large.

XXXIII. How does logistic regression handle binary classifiation problems ?  
Ans.   
 Logistic regression handles binary classification problems by modeling the probability that an input belongs to one of two classes(e.g 0 or 1).  **It does this by applying the logistic function to a linear combination of the input features.**

If the probability is above a certain threshold(usually 0.5) the model classifies the input as belonging to one class.(e.g 1).  
If the probability is below the threshold, the model classifies it as the other class (e.g. 0)

XXXIV. what is the sigmoid function and how is it used in logistic regression?  
Ans.   
 **Sigmoid function**: It is used to convert the linear combination of input features into a probability. It also known as the logistic function. it converts the valued number into value between 0 and 1. it is defined as:   
 σ(x) = 1 / (1+e^-x)

**How it work in logistic regression**:   
1. Linear combination : logistic regression first computes a linear combination of the input features, often represented as z=β0 +β1 x1 +β2 x2 +⋯+βn xn , where β are the coefficients.  
2. Sigmoid Application : sigmoid function is then applied to this linear combination, transforming it into probability value between 0 and 1.   
3. Classification : The output of sigmoid function represents probability that the input belongs to positive class or negative class.

XXXV. Explain the concept of the cost function in logistic regression.  
Ans.   
 The **cost function** measures the difference between the predicted probabilities and the actual binary outcomes (0 or 1).   
 **Log Loss** : this cost function is commonly used in logistic regression.

CF = −[ylog(hθ (x))+(1−y)log(1−hθ (x))]  
 Where,   
 hθ (x) - predicted probability of positive class (sigmoid function)  
 y is the actual label (0 or 1).

**Purpose**:  
 Minimizing the cost function during training ensures that model’s predicted probabilities align as closely as possible with the actual outcomes, leading to accurate classifications.

XXXVI. How can logistic regression be extended to handle multiclass classification?  
Ans.  
 It can be extended to handle multiclass classification using two main approaches:  
1. **One vs Rest(OVR)** : This approach involves creating a separate binary classifier for each class. Each classifier is trained to distinguish one class from all the others. During prediction, each classifier outputs a probability for its class, and the class with the highest probability is chosen as the final prediction.

2. **Softmax Regression** : This method generalizes logistic regression for multiclass classification by using the softmax function. The softmax function computes the probabilities for all classes simultaneously, ensuring that the probabilities sum to 1. The class with the highest probability is selected as the prediction.

XXXVII. what is the difference between L1 and L2 regularization in logistic regression?  
Ans.  
 **L1 Regularization(lasso)** : Adds the absolute value of the coefficients to the loss function. some coefficients may be reduces to exactly zero, effectively performing feature selection by keeping only the most important features.  
 Can lead to simpler models by eliminating some features entirely.  
 **L2 Regularization(Ridge)** : Adds the square of the coefficients to the loss function. This discourages large coefficients, but unlike L1, it tends to reduces coefficients more uniformly rather than driving them to zero.  
 generally reduces all the coefficients but keep them non-zero, leading to smoother models with less variance.

XXXIX. What is XGBoost and how does it differ from other boosting algorithms ?  
Ans.   
 **XGBoost(Extreme Gradient Boosting)** is a powerful, efficient implementation of the gradient boosting algorithm, designed for speed and performance in supervised learning tasks, particularly classification and regression.  
 **Key differences from other boosting algorithms**:  
 **1. Performance and speed** : XGBoost is optimized for speed and efficiency, with features like parallel processing, tree pruning and regularization, making it faster and more scalable than traditional boosting algorithms.  
 **2. Regularization** : XGBoost includes L1 and L2 regularization terms, which help prevent overfitting.  
 **3. Handling missing data** : It has built-in methods for handling missing data, allowing it to learn from incomplete datasets without requiring imputation.  
 XGBoost offering better accuracy, faster training times and more flexibility compared to other boosting algorithms.

XL. Explain the concept of boosting in the context of ensemble learning.  
Ans.  
 **Boosting** : It is an ensemble learning technique that improves model accuracy by combining multiple weak learners into strong learner.  
 **Key points** :  
 **1. Sequential Learning** : Models are built one after another, with each new model focusing on correcting the errors made by the previous ones.  
 **2. Weight Adjustment** : During each iteration, the data points that were misclassified by the current model are given higher weights, so the next model pays more attention to them.  
 **3. Model combination** : The final prediction is a weighted combination of all the weak learners, resulting in a more accurate and robust model.

Boosting Improves the accuracy of the final model. Boosting can be applied to various types of base models (e.g. decision trees, linear models)

XLI. How does XGBoost handle missing values ?  
Ans.  
 It handle missing values efficiently during the training process by automatically learning the best direction(left to right) to take at each tree split when a missing value is encountered.  
 **Key points** :   
 **1. Sparsity-Aware Algorithm** : It uses a sparsity-aware split finding algorithm that optimizes for missing values without requiring explicit imputation.  
 **2. Optimal Split Direction** : During tree construction, XGBoost determines the optimal direction for missing values by considering the gain for both possible directions and choosing the one that improves the model the most.

XLII. what are the key hyperparameters in XGBoost and how do they affect model performance ?  
Ans.  
 **Key hyperparameters in XGBoost are**:   
 1. n\_estimators : number of boosting trees in the ensemble. More trees generally improve accuracy but increase the risk of overfitting.  
 2. learning\_rate : step size used in updating weights after each boosting round.   
 3. Max\_depth : maximum depth of each tree. Deeper trees capture more complex patterns but can lead to overfitting.  
 4. Min\_child\_weight : Minimum sum of instance weights needed in a child.

XLIII. Describe the process of gradient boosting in XGBoost.  
Ans.   
 **Gradient Boosting in XGBoost** is an iterative process where each new model is trained to correct the errors of the previous models.

1. Start with an initial prediction.  
2. Calculate the difference between actual and predicted values.  
3. Train a new model to predict these difference.  
4. Add new model’s predictions to the previous predictions, scaled by learning rate.  
5. Continue this process for a specified no. Of iterations.  
6. Combine all model’s predictions to make the final prediction.

XLIV. what are the advantages and disadvantages of using XGBoost?  
Ans.  
 **Advantages of XGBoost** :  
 1. **High accuracy** as compare to other algorithm.  
 2. **Fast Training**: Optimized for speed with parallel processing.  
 3. **Regularization** : Includes L1 and L2 regularization to reduce overfitting.  
 4. **Handles Missing Data** : Automatically handles missing values.  
 5. **Scalability** : Efficient with large datasets.

**Disadvantages of XGBoost** :   
 1. Requires extensive hyperparameter tuning.  
 2. There is risk of overfitting if not carefully managed..  
 3. High memory usage – resource intensive, especially with large datasets.  
 4. Longer training time on small data.  
 5. Less Interpretability.