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?**

Python doesn't have static variables in the traditional sense like other languages (e.g., C++). However, you can achieve similar behavior using class attributes or global variables.

Class attributes: Shared among all instances of a class.

class MyClass:

class\_variable = 10

obj1 = MyClass

print(obj1.class\_variable)

Global variables: Defined outside of any function or class, accessible from anywhere

global\_var = 5

def my\_function():

print(global\_var)

my\_function()

1. **Explain the purpose of in a dictionary with suitable examples.**

Keys in a dictionary are unique identifiers used to access corresponding values. They must be immutable data types

person = {"name": "Shubhajit", "age": 23, "city": "Kolkata"}

print(person["name"]) # Output: Shubhajit

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

A frozenset is an immutable version of a set. Once created, its elements cannot be changed

my\_set = frozenset([1, 2, 3])

print(my\_set)

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

**Mutable:** Values can be changed after creation.

Examples: lists, dictionaries, sets

my\_list = [1, 2, 3]

my\_list[0] = 0 # Modifies the list

**Immutable:** Values cannot be changed after creation.

Examples: numbers, strings, tuples

my\_string = "hello"

my\_string[0] = "H" # Raises a TypeError

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

The \_\_init\_\_ method is a constructor in Python classes. It's called when an object of the class is created to initialize its attributes.

Example :

class Person:

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

self.name = name

self.age = age

person1 = Person("Shubhajit", 25)

print(person1.name) # Output: Shubhajit

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

Docstrings are strings placed at the beginning of a module, class, or function to provide documentation about their purpose and usage.

Example:

def my\_function(x, y):

"""

Adds two numbers together.

Args:

x: The first number.

y: The second number.

Returns:

The sum of x and y.

"""

return x + y

1. **What are unit tests in Python?**

Unit tests verify the correctness of individual code units (functions, methods). They help ensure code quality and reliability.

Python has libraries like unittest and pytest for writing unit tests

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

**break:** Exits the current loop.

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

**pass:** Does nothing, often used as a placeholder.

Example:

for i in range(5):

if i == 3:

break

print(i) # Output: 0 1 2

1. **What is the use Of self in Python?**

self refers to the instance of the class itself within a method. It's used to access and modify the object's attributes.

Example:

class MyClass:

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

self.value = value

def my\_method(self):

print(self.value) # Accessing the instance's attribute

obj = MyClass(10)

obj.my\_method() # Output: 10

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

**Global:** Accessible from anywhere.

**Protected:** Accessible within the class and its subclasses (conventionally prefixed with an underscore).

**Private:** Accessible only within the class (conventionally prefixed with double underscores).

**Example:**

class MyClass:

public\_var = 10

\_protected\_var = 20

\_\_private\_var = 30

def my\_method(self):

print(self.\_\_private\_var) # Accessible within the class

1. **What are modules and packages in Python?**

**Modules:** Single Python files containing definitions and statements.

**Packages:** Directories containing multiple modules, organized hierarchically.

**Example:**

# module1.py

def my\_function():

print("Hello from module1")

# package/module2.py

def another\_function():

print("Hello from module2 in package")

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

**Lists:** Ordered, mutable collections of elements.

**Tuples:** Ordered, immutable collections of elements

**Example:**

my\_list = [1, 2, 3]

my\_list[0] = 0 # Modifiable

my\_tuple = (4, 5, 6)

my\_tuple[0] = 4 # Raises a TypeError

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

**Interpreted:** Code is executed line by line at runtime. Ex- C, C++ , java

**Dynamically typed:** Data types are determined at runtime. EX- python,javascript

1. **What are Dict and List comprehensions?**

**Dict comprehensions:** Create dictionaries using concise syntax.

**Example:**

squares = {x: x\*\*2 for x in range(5)}

**List comprehensions:** Create lists using concise syntax.

**Example:**

even\_numbers = [x for x in range(10) if x % 2 == 0]

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

Decorators are functions that modify other functions. They add functionality without changing the original function's code.

Example:  
def decorator\_function(func):

def wrapper(\*args, \*\*kwargs):

print("Before function call")

result = func(\*args, \*\*kwargs)

print("After function call")

return result

return wrapper

@decorator\_function

def my\_function():

print("Inside function")

1. **How is memory managed in Python?**

Python uses garbage collection to manage memory automatically. It reclaims memory occupied by unused objects

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

Lambda functions are small, anonymous functions that can take any number of arguments but can only have one expression.

Example:

Double=lambda x:x\*2

a =Double(3)

print(a) #output- 6

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

**split():** Splits a string into a list of substrings based on a delimiter.

Example:

text = "apple,banana,orange"

fruits = text.split(",")

**join():** Joins elements of a list into a string using a delimiter.

**Example:**

fruits = ["apple", "banana", "orange"]

text = ",".join(fruits)

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

**Iterables:** Objects that can be iterated over (e.g., lists, tuples, strings).

**Iterators:** Objects that produce the next value with the next() function.

**Generators:** Functions that return an iterator.

**Example:**

my\_list = [1, 2, 3]

my\_iterator = iter(my\_list)

print(next(my\_iterator)) # Output: 1

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

**xrange** (Python 2): Returns a generator object, more memory-efficient for large ranges.

**range** (Python 3): Returns a range object, similar to xrange but more flexible.

1. Pillars of Oops.

**Encapsulation:** Bundling data (attributes) and methods that operate on that data within a single unit (class).

**Inheritance:** Creating new classes (subclasses) based on existing classes (superclasses), inheriting their attributes and methods.

**Polymorphism:** Objects of different classes can be treated as if they were of the same type.

**Abstraction:** Focusing on essential features while hiding implementation details.

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

you can check if a class is a child of another class using the built-in issubclass() function. This function returns True if the first argument is a subclass (direct or indirect) of the second argument, and False otherwise.

Example:

class Parent:

pass

class Child(Parent):

pass

print(issubclass(Child, Parent)) # True

print(issubclass(Parent, Child)) # False

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

Inheritance allows one class (the child or subclass) to inherit attributes and methods from another class (the parent or superclass). Python supports several types of inheritance:

**Single Inheritance**: A class inherits from one parent class.

class Parent:

def parent\_method(self):

print("This is the parent method")

class Child(Parent):

def child\_method(self):

print("This is the child method")

c = Child()

c.parent\_method() # This is the parent method

c.child\_method() # This is the child method

**Multiple Inheritance**: A class inherits from more than one parent class.

Example:

class Parent1:

def method1(self):

print("This is method1 from Parent1")

class Parent2:

def method2(self):

print("This is method2 from Parent2")

class Child(Parent1, Parent2):

pass

c = Child()

c.method1() # This is method1 from Parent1

c.method2() # This is method2 from Parent2

**Multilevel Inheritance**: A class inherits from a parent class, which in turn inherits from another class.

Example:

class Grandparent:

def grandparent\_method(self):

print("This is the grandparent method")

class Parent(Grandparent):

def parent\_method(self):

print("This is the parent method")

class Child(Parent):

def child\_method(self):

print("This is the child method")

c = Child()

c.grandparent\_method() # This is the grandparent method

c.parent\_method() # This is the parent method

c.child\_method() # This is the child method

**Hierarchical Inheritance**: Multiple classes inherit from the same parent class.

Example:

class Parent:

def parent\_method(self):

print("This is the parent method")

class Child1(Parent):

def child1\_method(self):

print("This is the child1 method")

class Child2(Parent):

def child2\_method(self):

print("This is the child2 method")

c1 = Child1()

c2 = Child2()

c1.parent\_method() # This is the parent method

c1.child1\_method() # This is the child1 method

c2.parent\_method() # This is the parent method

c2.child2\_method() # This is the child2 method

**Hybrid Inheritance**: A combination of two or more types of inheritance.

Example:

class Base:

def base\_method(self):

print("This is the base method")

class Parent1(Base):

def parent1\_method(self):

print("This is the parent1 method")

class Parent2(Base):

def parent2\_method(self):

print("This is the parent2 method")

class Child(Parent1, Parent2):

def child\_method(self):

print("This is the child method")

c = Child()

c.base\_method() # This is the base method

c.parent1\_method() # This is the parent1 method

c.parent2\_method() # This is the parent2 method

c.child\_method() # This is the child method

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

Encapsulation is a concept where the internal representation of an object is hidden from the outside. It is achieved by making the data members private and providing public methods to access and modify them.

Example:

class EncapsulatedObject:

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

self.\_\_value = value # private attribute

def get\_value(self):

return self.\_\_value

def set\_value(self, value):

self.\_\_value = value

obj = EncapsulatedObject(10)

print(obj.get\_value()) # 10

obj.set\_value(20)

print(obj.get\_value()) # 20

# Direct access will result in an error

# print(obj.\_\_value) # AttributeError: 'EncapsulatedObject' object has no attribute '\_\_value'

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

Polymorphism allows objects of different classes to be treated as objects of a common superclass. It provides a way to perform a single action in different forms.

Example:

class Animal:

def speak(self):

pass

class Dog(Animal):

def speak(self):

return "Woof!"

class Cat(Animal):

def speak(self):

return "Meow!"

def make\_animal\_speak(animal):

print(animal.speak())

dog = Dog()

cat = Cat()

make\_animal\_speak(dog) # Woof!

make\_animal\_speak(cat) # Meow!

**Question l. 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) \_ Percentage**

Answer:

* **Serial \_ no.**

**Invalid**: Spaces are not allowed in identifier names.

* **1st\_Room**

**Invalid**: Identifiers cannot begin with a digit.

* **Hundred$**

**Invalid**: The dollar sign $ is not allowed in identifier names.

* **Total\_Marks**

**Valid**: This is a valid identifier. It starts with a letter and contains only letters, digits, and underscores.

* **total—Marks**

**Invalid**: The — (em dash) is not allowed in identifier names. Only underscores \_ are allowed as special characters.

* **Total Marks**

**Invalid**: Spaces are not allowed in identifier names.

* **True**

**Invalid**: True is a reserved keyword in Python, representing the boolean value True.

* **\_Percentage**

**Valid**: This is a valid identifier. Identifiers can start with an underscore and contain only letters, digits, and underscores.

**Question 1.3.**

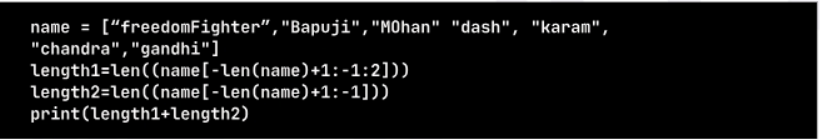
**name = [“mohan”,”dash”,”karam”,”Chandra”,”Gandhi”,”bapu”]**

**do the following operations in this list;**

**a) add an element “freedom \_ fighter" in this list at the 0th index.**

Answer: name.insert(0,”freedom\_fighter”)

**b) find the output of the following vand explain how?**



Answer: 8

**c) add two more elements in the name at the end of the list.**

Answer:

name.append(“element 1”)

name.append(“element 2”)

**d) what will be the value of temp:**

name = CBapuji", "dashi •karame,

name[Ol-temp

print(name)

Answer: “bapu”

**Question 1.4.**

**Find the output Of the following.**

animal =[ ‘Human’,’cat’,’mat’,’cat’,’rat’,’Human’,’Lion']

print(animal.count(‘Human’)) #output -- 2

print(animal.index(‘Cat’)) #output -- 1

print(len(animal)) #output – 7

**Question 1.5**

**tuple1=(10,20,"apple",3.4,'a',["master","ji"],("sita","geeta",22),**

        [{"roll\_no":1},{"name":"navneet"}])

1. print(len(tuple1))

output- 8

1. print(tuple1[-1][-1]["name"])

output- navneet

1. fetch the value of roll\_no from the tuple

output - print(tuple1[-1][0]["roll\_no"])

1. print(tuple1[-3][1])

output- ji

1. fetch the element ‘22’ from the tuple

output- print(tuple1[6][2])

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

**Measures of Central Tendency** are statistical metrics that describe the center or typical value of a dataset. They provide a summary measure that represents the entire distribution.

1. **Mean**: The average of all data points.
   * **Calculation**: Mean=sum(Xi)/N
2. **Median**: The middle value when the data points are sorted in ascending order. If there is an even number of observations, it is the average of the two middle values.
   * **Calculation**: Sort the data and find the middle value(s).
3. **Mode**: The value that appears most frequently in the dataset.
   * **Calculation**: Identify the most frequent value(s).

**Measures of Dispersion** describe the spread or variability of a dataset. They show how much the data points differ from the central measure.

1. **Range**: The difference between the maximum and minimum values.
   * **Calculation**: Range=Max−Min
2. **Variance**: The average of the squared differences from the mean, indicating the dispersion around the mean.
   * **Calculation**: Variance=(sum(Xi – mean)^2) /N
3. **Standard Deviation**: The square root of the variance, providing a measure of the average distance from the mean.
   * **Calculation**: Standard Deviation=root under Variance

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

**Skewness** measures the asymmetry of the distribution of a dataset around its mean. It provides insights into the shape of the distribution.

1. **Positive Skew (Right Skew)**: The tail on the right side of the distribution is longer or fatter. Most values are concentrated on the left.
   * **Example Graph**: A graph with a long right tail.
2. **Negative Skew (Left Skew)**: The tail on the left side of the distribution is longer or fatter. Most values are concentrated on the right.
   * **Example Graph**: A graph with a long left tail.
3. **No Skew (Symmetric Distribution)**: The distribution is symmetrical, with tails on both sides being approximately equal.
   * **Example Graph**: A bell-shaped curve (normal distribution).

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

**Probability Mass Function (PMF)**:

* **Definition**: A PMF is used for discrete random variables and gives the probability of each possible value the variable can take.
* **Properties**: P(X=x) where x is a specific value.
* **Example**: The PMF of rolling a fair die gives P(X=k)=1/6 for k=1

**Probability Density Function (PDF)**:

* **Definition**: A PDF is used for continuous random variables and describes the likelihood of a variable falling within a particular range.
* **Properties**: The area under the PDF curve over a range gives the probability of the variable falling within that range.
* **Example**: The PDF of a normal distribution is a bell-shaped curve where the total area under the curve is 1.

**Difference**:

* **PMF** is used for discrete variables and provides probabilities directly.
* **PDF** is used for continuous variables and provides a density; probabilities are obtained by integrating the PDF over an interval.

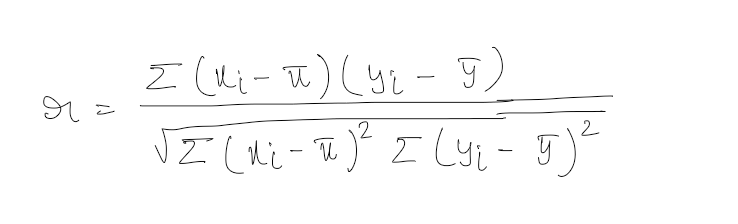
**23. What is correlation. Explain its type in details.what are the methods of determining correlation**

**Correlation** measures the strength and direction of the linear relationship between two variables. It ranges from -1 to 1.

1. **Positive Correlation**: As one variable increases, the other also increases. The correlation coefficient is between 0 and 1.
   * **Example**: Height and weight in adults.
2. **Negative Correlation**: As one variable increases, the other decreases. The correlation coefficient is between -1 and 0.
   * **Example**: Amount of gas in a tank and the distance traveled.
3. **No Correlation**: No linear relationship between the variables. The correlation coefficient is around 0.
   * **Example**: Shoe size and intelligence.

**Methods of Determining Correlation**:

1. **Pearson Correlation Coefficient**: Measures linear correlation. Calculated as:



1. **Spearman’s Rank Correlation Coefficient**: Measures the strength and direction of the monotonic relationship between two variables.
   * **Calculation**: Based on ranked values rather than actual values.
2. **Kendall’s Tau**: Measures correlation between two variables by comparing the number of concordant and discordant pairs.
   * **Calculation**: Uses ranks of values to compute correlation.

Each method is useful depending on the nature of the data and the type of relationship being investigated.

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

**Correlation** and **Regression** are both statistical techniques used to analyze the relationship between variables, but they have distinct purposes and characteristics. Here are four key differences:

1. **Purpose:**
   * **Correlation:** Measures the strength and direction of the linear relationship between two variables. It tells us whether and how strongly the variables are related but does not imply causation.
   * **Regression:** Predicts the value of one variable based on the value of another variable. It provides a model to estimate the dependent variable from the independent variable(s) and implies a causal relationship.
2. **Output:**
   * **Correlation:** Results in a correlation coefficient (e.g., Pearson’s r) which ranges from -1 to 1. This coefficient quantifies the strength and direction of the relationship.
   * **Regression:** Results in a regression equation (e.g., Y=a+bXY = a + bXY=a+bX) which allows for prediction of the dependent variable. It also provides regression coefficients and statistical significance.
3. **Modeling:**
   * **Correlation:** Does not involve creating a model for prediction. It simply describes the degree to which two variables move together.
   * **Regression:** Involves creating a predictive model. It specifies how changes in the independent variable are expected to affect the dependent variable.
4. **Assumptions:**
   * **Correlation:** Assumes linearity in the relationship and does not require assumptions about the distribution of the variables.
   * **Regression:** Assumes linearity, normality of residuals, homoscedasticity (constant variance of residuals), and independence of residuals.
5. **Find the most likely price at Delhi corresponding to the price of Rs. 70 at Agra from the following data: Coefficient of correlation between the prices of the two places +0.8.**

Given:

Coefficient of correlation (r) = +0.8

Price at Agra (X) = Rs. 70

To find the most likely price at Delhi (Y), we need to use the formula for regression predictions. However, additional information such as means and standard deviations would be necessary for precise calculation. Without the exact values for means and standard deviations, a simplified approach assumes linear relationship:

Y=meany + r. SDy/SDx.(X-meanx)

In the absence of actual means and standard deviations, if we assume that the mean prices are equal and standard deviations are also equal, the most likely price can be approximated using the given correlation directly:

Most Likely Price at Delhi≈meanY+r⋅(Price at Agra−meanX)

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

**Normal Distribution** is a continuous probability distribution that is symmetrical around its mean, with most of the observations clustering around the central peak and probabilities tapering off equally in both directions from the center.

**Four Assumptions:**

1. **Symmetry:** The distribution is symmetrical around the mean. The mean, median, and mode are all equal.
2. **Bell-shaped Curve:** The distribution forms a bell-shaped curve. It has a single peak at the mean.
3. **Mean and Standard Deviation:** The distribution is defined by its mean (μ) and standard deviation (σ). The mean determines the center, while the standard deviation determines the spread.
4. **Asymptotic:** The tails of the distribution approach the horizontal axis but never touch it. They extend infinitely in both directions.

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

1. **Bell-shaped Curve:** The normal distribution curve is bell-shaped, centered around the mean.
2. **Symmetry:** The curve is symmetric about the mean. The left and right sides are mirror images of each other.
3. **Mean, Median, and Mode Equality:** In a normal distribution, the mean, median, and mode are all equal and located at the center of the distribution.
4. **68-95-99.7 Rule:** Approximately 68% of the data falls within one standard deviation of the mean, 95% within two standard deviations, and 99.7% within three standard deviations.
5. **Asymptotic:** The tails of the distribution approach the horizontal axis but never touch it.
6. **Area under the Curve:** The total area under the normal distribution curve is equal to 1, representing the total probability.

Each of these properties and assumptions is fundamental for statistical analysis using normal distributions and is widely applied in various fields including finance, science, and engineering.

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

(a) Within a range 0.6745 of o on both sides the middle 50% of the observations occur i,e. mean \*0.67450 covers 50% area 25% on each side. 🡪**Incorrect**

(b) Mean ±IS.D. (ite.p ± Io) covers 68.268% area, 34.134 % area lies on either side of the mean. 🡪 **correct**

(c) Mean ±2S.D. (i,e. p 20) covers 95.45% area, 47.725% area lies on either side of the mean. 🡪 **correct**

(d) Mean S.D. (i,e. p \*30) covers 99.73% area, 49.856% area lies on the either side of the mean. 🡪 in**correct**

(e) Only 0.27% area is outside the range \*30. 🡪 **correct**

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

**(a) Statistical Hypothesis:**

A **statistical hypothesis** is a statement or assumption about a population parameter. It can be tested using statistical methods to determine if there's enough evidence to reject the null hypothesis in favor of the alternative hypothesis. There are two types of hypotheses:

* **Null Hypothesis (H0):** The hypothesis that there is no effect or no difference, and any observed effect is due to sampling variability.
* **Alternative Hypothesis (H1 or Ha):** The hypothesis that there is an effect or a difference.

**Errors in Hypothesis Testing:**

1. **Type I Error (α):** Rejecting the null hypothesis when it is actually true. Also known as a false positive.
2. **Type II Error (β):** Failing to reject the null hypothesis when the alternative hypothesis is actually true. Also known as a false negative.

**(b) Sample Explanation:**

A **sample** is a subset of a population used to make inferences about the population.

**Large Samples vs. Small Samples:**

* **Large Samples:** Typically, samples with a size greater than 30. Large samples provide more accurate estimates of population parameters and tend to follow the Central Limit Theorem, which ensures that the sampling distribution of the mean will be approximately normal regardless of the population distribution.
* **Small Samples:** Typically, samples with a size of 30 or less. Small samples may not accurately represent the population and can be more influenced by outliers or variability. The normality assumption may not hold, and different statistical methods or corrections might be necessary.

**Machine Learning**

**1.What is the difference between Series & Dataframes?**

**Series**: A one-dimensional labeled array capable of holding any data type (integers, strings, floating-point numbers, Python objects, etc.). It is similar to a column in an Excel spreadsheet or a database table.

**DataFrame**: A two-dimensional, size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). It is similar to a table in a relational database or an Excel spreadsheet.

**2.** **Create a database name Travel\_Planner in mysql ,and  create a table name bookings in that which having attributes (user\_id INT,  flight\_id INT,hotel\_id INT, activity\_id INT,booking\_date DATE) .fill with some dummy value .Now you have to read the content of this table using pandas as dataframe.Show the output**

import pandas as pd

import mysql.connector

Establish connection

conn = mysql.connector.connect(

host="localhost",

user="your\_username",

password="your\_password",

database="Travel\_Planner"

)

query = "SELECT FROM bookings"

df = pd.read\_sql(query, conn)

print(df)

conn.close()

**3. Difference between loc and iloc?**

-loc: Access a group of rows and columns by labels or a boolean array. It is label-based indexing.

Example: df.loc[0, 'user\_id'] or df.loc[0:2, ['user\_id', 'flight\_id']].

iloc: Access a group of rows and columns by integer position(s). It is integer-location based indexing.

Example: df.iloc[0, 0] or df.iloc[0:2, [0, 1]].

**4. Difference between Supervised and Unsupervised Learning**

Supervised Learning: The model is trained on a labeled dataset, meaning each training example is paired with an output label. Examples include classification and regression.

Example Algorithms: Linear Regression, Support Vector Machines, Neural Networks.

Unsupervised Learning: The model is trained on data without labeled responses. The goal is to infer the natural structure present within a set of data points. Examples include clustering and dimensionality reduction.

Example Algorithms: K-Means Clustering, Principal Component Analysis (PCA).

**5. Bias-Variance Tradeoff**

Bias: Error due to overly simplistic assumptions in the learning algorithm. High bias can cause the model to miss relevant relations between features and target outputs (underfitting).

Variance: Error due to too much complexity in the learning algorithm. High variance can cause the model to model the random noise in the training data instead of the intended outputs (overfitting).

The tradeoff is about finding the right balance where both bias and variance are minimized to improve the model's performance on unseen data.

**6. What are precision and recall? How are they different from accuracy?**

Precision, recall, and accuracy are metrics used to evaluate the performance of classification models, particularly in contexts where the classes are imbalanced or when different types of errors have different implications.

Differences Between Precision, Recall, and Accuracy

* **Precision vs. Recall**:
  + **Precision** focuses on the quality of positive predictions, asking: "Of all the positive predictions I made, how many were actually correct?"
  + **Recall** focuses on the completeness of positive predictions, asking: "Of all the actual positives, how many did I successfully identify?"
* **Accuracy**:
  + Accuracy measures the overall correctness of the model and is given by the ratio of correctly classified instances (both positive and negative) to the total instances.
  + Accuracy does not differentiate between false positives and false negatives, which can be a limitation in cases of class imbalance.

**7. What is overfitting and how can it be prevented?**

Overfitting: A modeling error that occurs when a model is too complex, capturing noise in the training data rather than the intended outputs. This results in poor generalization to new data.

Prevention Techniques:

Cross-Validation: To ensure the model generalizes well.

Regularization: Techniques like L1 (Lasso) or L2 (Ridge) to penalize larger coefficients.

Pruning: Reducing the size of the model, especially in decision trees.

Early Stopping: Stopping training when performance on a validation set starts to degrade.

Gather More Data: More data can help the model learn better.

**8. Explain the concept Of cross-validation.**

Cross-validation is a technique used to evaluate the performance of a machine learning model on unseen data. It involves splitting the dataset into multiple subsets, training the model on some subsets, and validating it on the remaining subsets.

**9. What is the difference between a classification and a regression problem?**

Classification and regression are two fundamental types of supervised learning problems in machine learning. They differ primarily in the nature of the output variable they predict.

Regression problems aim to predict a continuous numerical value based on input features.

**10. Explain the concept of ensemble learning.**

Combines multiple models to improve performance. The idea is to leverage the strengths of different models to produce a more robust model.

**11. What is gradient descent and how does it work?**

Gradient descent is an optimization algorithm used to minimize the cost function in machine learning models and neural networks. It is fundamental for training models by iteratively adjusting the model parameters to find the values that minimize the error.

**How Gradient Descent Works**

1. **Initialize Parameters**: Start with initial values for the model parameters (weights). These values can be random or zero.
2. **Compute Gradient**: Calculate the gradient (partial derivatives) of the cost function with respect to each parameter. The gradient indicates the direction of the steepest ascent in the cost function.
3. **Update Parameters**: Adjust the parameters in the direction opposite to the gradient to minimize the cost function. The size of the adjustment is controlled by the learning rate.
4. **Repeat**: Repeat the process of computing the gradient and updating the parameters until the cost function converges to a minimum value or until a predefined number of iterations is reached.

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

Batch Gradient Descent: Uses the entire dataset to compute the gradient of the cost function.

- Pros: Stable and converges smoothly.

- Cons: Computationally expensive for large datasets.

Stochastic Gradient Descent (SGD): Uses one sample at a time to update the parameters.

- Pros: Faster and handles large datasets well.

- Cons: Noisy updates, which can lead to convergence issues.

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

Refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces. Problems include increased computational cost, sparsity, and overfitting

**14. Explain the difference between LI and L2 regularization.**

L1 Regularization (Lasso Regularization)

L1 regularization adds a penalty equal to the absolute value of the magnitude of coefficients to the loss function. It is also known as Lasso (Least Absolute Shrinkage and Selection Operator) regularization.

L2 Regularization (Ridge Regularization)

L2 regularization adds a penalty equal to the square of the magnitude of coefficients to the loss function. It is also known as Ridge regularization.

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

A confusion matrix is a tool used to evaluate the performance of a classification algorithm. It provides a summary of the classification results by showing the number of true and false predictions made by the model. The confusion matrix helps to understand how well the model is performing and where it might be making errors.

**16. Define AUC-ROC curve.**

The AUC-ROC curve is a performance measurement tool for classification models, particularly useful for evaluating binary classification problems. It combines two important metrics: the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC).

**17. Explain the k-nearest neighbors algorithm.**

The k-Nearest Neighbors (k-NN) algorithm is a simple, yet powerful, machine learning algorithm used for classification and regression tasks. It is a type of instance-based learning or lazy learning where the model makes predictions based on the similarity between the data points.

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

A **Support Vector Machine (SVM)** is a powerful supervised learning algorithm used primarily for classification tasks, but it can also be adapted for regression. The core idea of SVM is to find a hyperplane that best separates different classes in the feature space.

**19. How does the kernel trick work in SVM?**

The **kernel trick** is a powerful technique used in Support Vector Machines (SVMs) to handle non-linearly separable data by transforming it into a higher-dimensional space where a linear separation is possible. This trick allows SVMs to find complex decision boundaries without explicitly computing the coordinates in the higher-dimensional space.

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

In Support Vector Machines (SVM), different types of kernel functions are used to transform the data into a higher-dimensional space, where it may be easier to find a separating hyperplane. Here’s a breakdown of the common kernel functions used in SVMs, along with their characteristics and typical use cases:

* + 1. **Linear Kernel**
    2. **Polynomial Kernel**
    3. Radial Basis Function (RBF) Kernel
    4. Sigmoid Kernel

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

In Support Vector Machines (SVM), the **hyperplane** is a crucial concept that represents the decision boundary separating different classes in the feature space.

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

**Pros**

1. **Effective in High-Dimensional Spaces**: SVMs are particularly effective when the number of dimensions is greater than the number of samples.
2. **Versatility**: SVMs can be used for both classification and regression tasks. They can also handle both linear and non-linear data.
3. **Robustness to Overfitting**: SVMs use regularization parameters (like C) to avoid overfitting, especially in high-dimensional space.
4. **Clear Margins of Separation**: SVMs maximize the margin between the data points and the decision boundary, which can result in a more robust model.
5. **Effective with a Clear Margin of Separation**: They work well when the classes are clearly separable and there's a large margin.
6. **Kernel Trick**: The kernel trick allows SVMs to solve non-linear classification problems by implicitly mapping the input features into high-dimensional feature spaces.

**Cons**

1. **Memory-Intensive**: SVMs require a lot of memory for larger datasets as they store all support vectors from the training data.
2. **Training Time**: The training time can be quite slow for large datasets. This is especially true for non-linear kernels.
3. **Complexity in Choosing Kernel Functions**: Selecting the right kernel function and tuning its parameters can be complex and computationally expensive.
4. **Not Suitable for Large Datasets**: SVMs are not suitable for very large datasets due to their high training time and memory usage.
5. **Difficult to Interpret**: The resulting model from an SVM can be difficult to interpret, especially when non-linear kernels are used.
6. **Sensitivity to Parameters**: SVMs require careful tuning of parameters such as the regularization parameter (C) and the kernel parameters (e.g., gamma for the RBF kernel).

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

**Hard Margin SVM**

**Definition**: A hard margin SVM strictly enforces that all training data points must be correctly classified and that there must be no data points within the margin.

**Soft Margin SVM**

**Definition**: A soft margin SVM allows some data points to be within the margin or misclassified. It introduces a trade-off between maximizing the margin and minimizing the classification error.

**24. Describe the process of constructing a decision tree.**

**Data Preparation**:

Collect and preprocess data (handle missing values, encode categorical variables, etc.).

**Choose Splitting Metric**:

Select a metric (Gini impurity, entropy, variance reduction, etc.).

**Build the Tree**:

Start with the entire dataset as the root node.

For each node:

Evaluate all possible splits for each feature.

Choose the best split based on the splitting metric.

Create child nodes and assign data subsets to them.

Repeat until stopping criteria are met (e.g., max depth, min samples per node).

**Stopping Criteria**:

Define criteria such as maximum depth, minimum samples per node, or minimum information gain.

**Pruning (Optional)**:

Post-prune the tree to reduce overfitting using techniques like cost complexity pruning.

**Validation**:

Evaluate the tree's performance using a validation set or cross-validation.

Tune hyperparameters to optimize performance.

**25. Describe the working principle of a decision tree.**

A decision tree splits data into subsets based on feature values to create a tree-like structure for making decisions. Internal nodes represent tests on attributes, branches are outcomes of tests, and leaf nodes represent class labels or continuous values.

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

Information gain measures how well an attribute separates the training data into classes. It is calculated as the difference between the entropy of the dataset before and after the split. The attribute with the highest information gain is selected for the split.

1. **Explain Gini impurity and its role in decision trees.**

Gini impurity measures the probability of a randomly chosen element being incorrectly classified if it was randomly labeled according to the distribution of labels in the subset. Lower Gini impurity indicates a better split.

1. **What are the advantages and disadvantages of decision trees?**

**Advantages**:

* Easy to understand and interpret.
* Handles both numerical and categorical data.
* Requires little data preprocessing.

**Disadvantages**:

* Prone to overfitting.
* Can be unstable with small variations in data.
* Greedy algorithm might not find the optimal tree.

1. **How do random forests improve upon decision trees?**

Random forests improve decision trees by averaging multiple trees to reduce overfitting and increase accuracy. They use bootstrapping and feature randomness to create diverse trees.

1. **How does a random forest algorithm work?**

 Create multiple decision trees using bootstrapped samples of the data.

 For each split, consider a random subset of features.

 Aggregate the predictions of all trees (majority vote for classification, average for regression).

1. **What is bootstrapping in the context of random forests?**

Bootstrapping involves creating multiple datasets by sampling with replacement from the original dataset, ensuring diversity among the trees.

1. **Explain the concept of feature importance in random forests.**

Feature importance is measured by how much a feature reduces impurity across the forest. Features with higher importance scores are more influential in making predictions.

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

 **Number of Trees**: More trees generally improve performance but increase computation time.

 **Max Depth**: Limits the depth of trees to prevent overfitting.

 **Min Samples Split/Leaf**: Minimum number of samples required to split a node or to be a leaf node.

 **Max Features**: Number of features considered for splitting a node.

1. **Describe the logistic regression model and its assumptions.**

Logistic regression models the probability of a binary outcome using a logistic function. It assumes linearity between predictors and the log-odds of the outcome, and independence of observations.

1. **How does logistic regression handle binary classification problems?**

It predicts probabilities using the sigmoid function, which maps any real-valued number to a value between 0 and 1.

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

The sigmoid function is used to transform the linear output of a logistic regression model into a probability: σ(x)=1/(1+e^-x)

1. **Explain the concept of the cost function in logistic regression.**

The cost function (log-loss) measures the difference between predicted probabilities and actual class labels, and is minimized during training.

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

Handled using one-vs-rest (OvR) or softmax regression, where the model predicts the probability distribution over multiple classes.

1. **What is the difference between LI and 12 regularization in logistic regression?**

**L1 Regularization**: Adds the absolute value of coefficients to the loss function, promoting sparsity.

**L2 Regularization**: Adds the squared value of coefficients to the loss function, promoting small weights.

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

XGBoost is an optimized gradient boosting algorithm known for its high performance and speed. It differs from other boosting algorithms by incorporating regularization, handling missing values natively, and employing a more efficient tree-building algorithm.

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

Boosting combines weak learners (typically decision trees) sequentially, with each new model focusing on the errors of the previous ones to improve overall performance.

1. **How does XGBoost handle missing values?**

XGBoost can automatically learn the best direction to handle missing values in its tree-building process.

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

 **Learning Rate**: Controls the contribution of each tree.

 **Number of Trees**: Total number of boosting rounds.

 **Max Depth**: Maximum depth of each tree.

 **Subsample**: Fraction of samples used for training each tree.

 **Colsample\_bytree**: Fraction of features used for training each tree

1. **Describe the process of gradient boosting in XGBoost.**

Involves adding new models that predict the residuals (errors) of prior models, using gradient descent to minimize the loss function.

1. **What are the advantages and disadvantages of using XGBoost?**

**Advantages**:

* High performance and speed.
* Handles missing values and large datasets efficiently.
* Regularization to prevent overfitting.

**Disadvantages**:

* Can be complex to tune.
* Computationally intensive for very large datasets.