Question 1.1:
1:STATIC VARIABLES: static variables are variables that are assigned outside the methods in the class and are accessible in all the methods throughout the class.
DYNAMIC VARIABLES: dynamic variables are those variables which are created within the methods in the class.

2: “pop()”: This method removes a specified key from the dictionary and returns its associated value.
Example:

```
dict1={'shivani':1,'dushyant':2,'nikhil':3}
dict1.pop('shivani')
#output: 1
```

“popitem()”: this method removes the last key value pair.
Example:

```
dict1={'shivani':1,'dushyant':2,'nikhil':3}
dict1.popitem()
#output: (“nikhil”,3)
```

“clear()”: this method removes all the key value pair and left dictionary empty.
Example:

```
dict1={'shivani':1,'dushyant':2,'nikhil':3}
dict1.clear()
#output : {}
```

3: “frozenset”: frozensets are type of sets that are immutable in nature.
Example:

```
 frozen_set=frozenset({"apple","shivani","neha","harsh"
print(frozen_set)#output:frozenset({‘neha’,’shivani’,
 ‘apple’,’harsh’
frozen_set[3]=”Harsh” #output: TypeError:’frozenset’ object does not support item assignment
```

4: “mutable’’: mutable data types are those data types in which we can change the elements after once they are created.
Example: sets, array
“Immutable” : immutable data types are those data types in which we cannot change the elements after once they are created.
Example: list ,dictionary

5: “__init__” : __init__ is the constructor in the class is used to initialize the instances in the class and when we create the
new instance then it will call itself and assign the values.
Example:

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

6: docstring : docstrings are the multiline comments which are written in the code to make it easier to understand to other people.
Example :

```
Def sum(num1,num2):
“”” this is the function named sum
here it is summing up two numbers
```

```
 and return the output “””
Return num1+num2
```

7: “Unit tests”: unit tests in python are used to test the code if it is performing as expected or not or to fix the error or the bugs in it.
8: “break”: break is the statement used in loops to stop the iteration where the given condition meets.
Example:

```
  for i in range(9):
     If i==6:
        break
```

“Continue”: continue is the statement used in loops to escape the current iteration output and moving to the next iteration.
Example:

```
 for i in range(9):
     If i==7:
        continue
```

“Pass”: pass statement is used to pass the loop to the next iteration without affecting the output.

```
 for i in range(9):
     If i==5:
        pass
```

8: “self”: self is the parameter used in __init__ method to access the attributes in the class.
Example:

```
class Students:
 def __init__(self):
```

9: “global”: global attribute can be accessed by any function or class within the same module.
Example:

```
class Itsme:
 def __init__(self):
 self.its_shiv = “ hello myself shivani”
```

“Protected”:protected attribute are the attributes which can be accessed within the class or can can be accessed outside the class
 If you know the name of the attribute.
example:

```
class Itsme:
 def __init__(self):
 self._its_shiv = “ hello myself shivani”
```

“Private”: private attribute can be accessed within the class only.
Example:

```
class Itsme:
 def __init__(self):
 self.__its_shiv = “ hello myself shivani”
```

 10: “module”: module is file containing functions, classes, and variables in it.
Example:
Scipy, matplotlib, pandas

“Packages”: A package is the way to organize the related modules.
Example: __init__py

11: “list”: lists are the mutable collection of elements that can be string ,integer or float etc.
Example:

```
list1=[1,2.4,’apple’, ‘’pass’’: dsghgd]
```

“Tuple”: tuple is a collection of elements which can be of any data type such as string,float ,dictionary etc. and it is immutable in nature.
Example:
tuple1=(5,2.5, “shivani”, “key1’’:hkt43gfd)

12: interpreted language: An interpreted language is a programming language that does not need to be compiled into machine code before it is executed.
Example: Python
Dynamically typed language: dynamically typed language is the language in which the programmer has to specify the data type of a variable at the time of its declaration.
Example: c++
Difference between interpreted language and dynamically typed language:-
Interpreted language:

- Do not require compilation before execution
- Slower execution speed
- Error handling at runtime
- Type checking at runtime
- Automatic memory management
- Platform-independent
- Faster development speed
- More manual debugging
- Less optimization

Dynamically typed language:

- Require compilation before execution
- Faster execution speed
- Error handling at compile-time
- Type checking at compile-time
- Manual memory management
- Platform-dependent

13: Dictionary comprehension: Dict comprehensions are a concise way to create dictionaries in Python.
Example: [a**4 for a in range(9)]
List comprehension:List comprehensions are a concise way to create lists in Python.
Example: {a:a**2 for a in range(9)}
14:
15: when we create an object python allocates memory for this object and deallocates the memory if no longer needed.
Example:

```
array1=("apple","shivani","neha","harsh")
array1
del array1 # output= NameError name 'array1' is not defined
```

16: lambda is a python function which is used to define short or one-time function in a single expression.
Example:

```
div=lambda x,y:x//y
print(div(8,3))# output=2
```

17: “split”: split is used to split the string into the list of substring separated by the given separator .

```
str1="hello , my self shivani, How have you been"
str1.split()
#output =['hello', ',', 'my', 'self', 'shivani,', 'How', 'have', 'you',
'been']
```

“Join”: join is used to join the list of strings into a single string.

```
str2=['hello', ',', 'my', 'self', 'shivani,', 'How', 'have', 'you',
'been']
"".join(str2)
#output=‘hello,myselfshivani,Howhaveyoubeen’
```

18: “Iterable”: An iterable are the objects that can be iterated over.
Example: ‘iter()’
“Iterator”: An iterator are the objects that keeps track of its position in sequence.
Example:

“Generators”: generators are the iterators that are used to generate a sequence of values without storing it in the memory at all once.
Example: ‘yield()’

19: “xrange” : xrange is used to generate the iterables in the given range start:end:step in old version of python.
Example:

```
shivi=xrange(5)# python(2.x)
print(shivi)#output= 0 1 2 3 4
```

“Range”: range is used to generate iterables sequences in the given range in latest version of python.
Example:

```
for i in range(5):
  print(i)
# output=
0
1
2
3
4
```

20: pillar of OOPs: encapsulation, inheritance, abstraction, polymorphism
Example:
21: we will check using “issubclass” function if a class is a child class or not.
Example:

```
class Pushpa:
    pass
class shivi(Pushpa):
    pass
print(issubclass(shivi, Pushpa))  # Output: True
```

22: Inheritance: inheritance is a fundamental concept of OOPs in which the subclass/child class inherit the properties of its parent class.
There are 5 types of inheritance:
“Single inheritance”: child class inherits from a single parent class.

```
class Fish:
  def smell(self):
    print("this smells fishy")
class Sheldon(Fish):
  def sound(self):
    print("baby sheldon")
my_fish=Sheldon()
my_fish.smell()
# output= this smells fishy
```

“Multiple inheritance”: In this inheritance child class inherits from multiple parent classes.
Example:

```
class Ocean:
  def species(self):
    print("This animal live in water ")
class Fish:
  def smell(self):
    print("this smells fishy")
class Sheldon(Ocean,Fish):
  def sound(self):
    print("baby sheldon")
my_fish=Sheldon()
my_fish.species()  #output=This animal live in water
```

“Multi level inheritance”: in this inheritance child class inherits from a parent class that itself inherits from another parent class.
Example:

```
class Ocean:
  def species(self):
    print("This animal live in water ")
class Fish(Ocean):
  def smell(self):
    print("this smells fishy")
class Sheldon(Fish):
  def sound(self):
    print("baby sheldon")
my_fish=Sheldon()
my_fish.species()
my_fish.smell() #output=This animal live in water
this smells fishy
```

“Hierarchical inheritance”: in this inheritance multiple child classes inherit from a single parent.
Example:

```
class Ocean:
  def species(self):
    print("This animal live in water ")
class Fish(Ocean):
  def smell(self):
    print("this smells fishy")
class Sheldon(Ocean):
  def sound(self):
    print("baby sheldon")
my_fish=Sheldon()
fish_2=Fish()
my_fish.species()
fish_2.species()
# output: This animal live in water
This animal live in water
```

“Hybrid inheritance”: this inheritance is combination of multiple inheritance and multilevel inheritance.
Example:

```
class Ocean:
  def aquatic_species(self):
    print("This animal live in water ")
class Land:
  def terrestrial_species(self):
    print("this animal live on land")
```

```
class Both_land_water(Ocean,Land):
  def amphibians_species(self):
    print(" this species live in both water and land")
class Fish(Ocean):
  def aqua(self):
    print("Fishes live in water ")
class Human(Land):
  def terres(self):
    print("humans live on land")
class Frog(Both_land_water):
  def amphi(self):
    print("frogs live in both water and land")
obj1=Frog()
obj1.aquatic_species()
obj2=Human()
obj2.terrestrial_species()
#output= This animal live in water
this animal live on land
```

23: “encapsulation”: encapsulation means hiding something .
Example:

```
class Love:
  def __init__(self,name,age,salary):
    self.name=name
    self.age=age
    self.__salary=salary
  def tell(self):
    print("His name is",self.name,"and his age is",
self.age,self.__salary)
obj1=Love('shivi',20,50000)
obj1.salary()
output= ---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-63-3498b297cb80> in <cell line: 1>()
----> 1 obj1.salary()

AttributeError: 'Love' object has no attribute 'salary'
```

24: “polymorphism”: polymorphism is fundamental concept of OOPs where objects of different classes have same functionalities
and child class is overriding the properties of parent class.
Example:

```
class group:
  def mathematics(self):
    print("this is mathematics arrangement period")
class Group(group):
  def mathematics(self):
    print("this is mathematics period")
obj1=Group()
obj1.mathematics()
#output=
this is mathematics period
```

Question: 1.2
 According to the rules in python to create a variable :
a) Serial_no.= valid because we can create a variable name starting with alphabets.
b) 1st_Room= invalid because we cannot create a variable name starting with numerical value.
c) Hundred$= invalid because variable name cannot have a special character in it.
d) Total_Marks= valid because following all the rule of python to name a variable.
e) total-Marks= invalid because do not follow the rules of naming the python variable.
f) Total Marks= invalid because variable name cannot have space in it.
g) True= invalid because it is built in keyword in python and we cannot use it to name the variable.
h) _Percentag= valid because we can start a variable name with an alphabet or underscore.

Question:1.3
a)

```
name = ["Mohan", "dash", "karam", "chandra","gandhi","Bapu"]
name.insert(0,"freedom_fighter")
name
#output=
['freedom_fighter', 'Mohan', 'dash', 'karam', 'chandra', 'gandhi', 'Bapu']
```

b)

```
name = ["freedomFighter","Bapuji","MOhan" "dash", "karam",
"chandra","gandhi"]
length1=len((name[-len(name)+1:-1:2]))
length2=len((name[-len(name)+1:-1]))
```

```
print(length1+length2)
#output=6
# explanation: the list named name is given and it has a length of 6, for
length1 indexing is starting
 from -5   and ending at index -1 and the step size is 2 so in length1
there will be index -5,-2 so length1 will be 2 for length 2 indexing is
starting from -5 and ending at index -1 with a step size of 1
So in length2 there will be index -5,-4,-3,-2 so the length of length2
will be  4. The final result will be 6.
```

c)

```
name = ["freedomFighter","Bapuji","MOhan" "dash", "karam",
"chandra","gandhi"]
name.extend(["NetaJi","Bose"])
name
#output=['freedomFighter',
 'Bapuji',
 'MOhandash',
 'karam',
 'chandra',
 'gandhi',
 'NetaJi',
 'Bose']
```

d)

```
name = ["Bapuji", "dash", "karam", "chandra","gandi","Mohan"]
temp=name[-1]
name[-1]=name[0]
name[0]=temp
print(name)
 output:
name = ["Mohan", "dash", "karam", "chandra","gandi","Bapuji"]
```

Question 1.4:

```
animal = ['Human','cat','mat','cat','rat','Human', 'Lion']
print(animal.count('Human'))
print(animal.index('rat'))
print(len(animal))

Output:
2
4
```

7

Question 1.5: Question 1.5.
tuple1=(10,20,"Apple",3.4,'a',["master","ji"],("sita","geeta",22),[{"roll_no"N1},
{"name"N"Navneet"}])

a) 8
b) “Navneet”
c)

```
tuple1=(10,20,"Apple",3.4,'a',["master","ji"],("sita","geeta",22),[{"roll_
no":1},
{"name":"Navneet"}])
print(tuple1[7][0]["roll_no"])
#output= 1
```

d) “Ji”

e)

```
tuple1=(10,20,"Apple",3.4,'a',["master","ji"],("sita","geeta",22),[{"roll_
no":1},
{"name":"Navneet"}])
print(tuple1[6][2])
```

Question 1.6:

```
color = input("Enter the color of the signal (RED/Yellow/Green):
").strip()

if color == "RED":
    print("Stop")
elif color == "Yellow":
    print("stay")
elif color == "Green":
    print("Go")
else:
    print("Invalid color. Please enter traffic light colour")

#output=
 Enter the color of the signal (RED/Yellow/Green):  RED
Stop
```

Question 1.7:

```
def add(num1,num2):
  return num1 +num2
def subtract(num1,num2):
  return num1-num2
def multiply(num1,num2):
  return num1*num2
def divide(num1,num2):
  return num1/num2
print(add(8,2))
print(subtract(9,5))
print(multiply(6,3))
divide(9,3)
#output=
10
4
18
3.0
```

Question: 1.8

```
input1=int(input("enter the first number:"))
input2=int(input("enter the second number:"))
input3=int(input("enter the third number:"))
result= input1 if input1>input2 and input1>input3 else input2 if
input2>input1 and input2>input3 else input3
print("the larger number is:" ,result)
#output=
enter the first number:87
enter the second number:65
enter the third number:67
the larger number is: 87
```

Question:1.9

```
whole_number=int(input("enter the whole number"))
i=1
while i <=whole_number:
  if whole_number%i==0:
    print(i)
  i+=1
#output=enter the whole number 56
```

```
1
2
4
7
8
14
28
56
```

Question:1.10

```
summation=0
while True:
  num=int(input("enter the number"))

  if num>0:
    summation+=num
    print(summation)

  else:
    break

#output=
enter the number 56
56
enter the number 67
123
enter the number-1
```

Question: 1.11
Question: 1.12
1:

```
subjects=['mathematics','english','hindi','science','social science']
marks=[]
for i in subjects:
  mark=int(input("enter your marks:"))
  if 0 <= mark <100:
    marks.append(mark)
print(marks)
```

```
#output=
enter your marks: 67
enter your marks:56
enter your marks:89
enter your marks:95
enter your marks:92
[67, 56, 89, 95, 92]
```

2:

```
subjects=['mathematics','english','hindi','science','social science']
marks=[]
for i in subjects:
  mark=int(input("enter your marks:"))
  if 0 <= mark <100:
    marks.append(mark)
print(marks)
sum=0
for i in marks:
  sum+=i
print(sum)
percentage=(sum/500)*100
print(percentage)
#output=
enter your marks: 56
enter your marks:89
enter your marks:86
enter your marks:54
enter your marks:56
[56, 89, 86, 54, 56]
341
68.2
```

3:

```
subjects=['mathematics','english','hindi','science','social science']
marks=[]
for i in subjects:
  mark=int(input("enter your marks:"))
  if 0 <= mark <100:
    marks.append(mark)
print("marks:",marks)
sum=0
for i in marks:
  sum+=i
print("total:",sum)
percentage=(sum/500)*100
print("percentage:",percentage)
if percentage>85:
  print("grade:","A")
elif 75<= percentage <85:
   print("grade:","B")
elif 50<= percentage <75:
   print("grade:","C")

elif 30<= percentage<50:
  print("grade:","D")
else:
  print("reappear")

#output:
enter your marks: 89
enter your marks:9
enter your marks:92
enter your marks:95
enter your marks:91
marks: [89, 9, 92, 95, 91]
total: 376
percentage: 75.2
grade: B
```

Question:1.13

```
wavelength=int(input("enter the wavelength"))
if 400<=wavelength<440:
  print("violet")
elif 440<=wavelength<460:
  print("indigo")
elif 460<=wavelength<500:
  print("blue")
elif 500<=wavelength<570:
  print("green")
elif 570<=wavelength<590:
  print("yellow")
elif 590<=wavelength<620:
  print("orange")
elif 620<=wavelength<720:
  print("red")
#output=
enter the wavelength 560
green
```

Question:1.14
a)

```
mass_earth = 5.972e24
mass_moon = 7.34767309e22
mass_sun =1.989e30
distance_earth_sun=1.496e11
distance_moon_earth=3.844e8
gravitational_force_sun_earth=
6.674*10**-11*(mass_earth*mass_sun)/distance_earth_sun**2
gravitational_force_sun_earth
#output=
3.5422368558580452e+22
```

b)

```
mass_earth = 5.972e24
mass_moon = 7.34767309e22
mass_sun =1.989e30
distance_earth_sun=1.496e11
distance_moon_earth=3.844e8
gravitational_force_moon_earth=
6.674*10**-11*(mass_earth*mass_moon)/distance_moon_earth**2
gravitational_force_moon_earth
#output=
1.9819334566450407e+20
```

c)

```
mass_earth = 5.972e24
mass_moon = 7.34767309e22
mass_sun =1.989e30
distance_earth_sun=1.496e11
distance_moon_earth=3.844e8
gravitational_force_sun_earth=
6.674*10**-11*(mass_earth*mass_sun)/distance_earth_sun**2
print(gravitational_force_sun_earth)
gravitational_force_moon_earth=
6.674*10**-11*(mass_earth*mass_moon)/distance_moon_earth**2
print(gravitational_force_moon_earth)
if gravitational_force_sun_earth>gravitational_force_moon_earth:
  print("gravitational force between sun and earth is greater than
gravitational force between moon and earth")
else:
  print("gravitational force between moon and earth is greater than
gravitational force between sun and earth")
```

d)

```
mass_earth = 5.972e24
mass_moon = 7.34767309e22
mass_sun =1.989e30
distance_earth_sun=1.496e11
distance_moon_earth=3.844e8
gravitational_force_sun_earth=
6.674*10**-11*(mass_earth*mass_sun)/distance_earth_sun**2
print(gravitational_force_sun_earth)
gravitational_force_moon_earth=
6.674*10**-11*(mass_earth*mass_moon)/distance_moon_earth**2
print(gravitational_force_moon_earth)
moon_acceleration=gravitational_force_moon_earth/mass_moon
print(moon_acceleration)
earth_acceleration=gravitational_force_moon_earth/mass_earth
print(earth_acceleration)
if earth_acceleration>moon_acceleration:
  print(" earth is faster than moon")
else:
  print("moon is faster than earth")
#output=
3.5422368558580452e+22
1.9819334566450407e+20
0.002697362052405955
3.3187097398610864e-05
moon is faster than earth
```

Question: 2

```
class Students:
  def __init__(self,name,age,roll_number):
    self.__name=name
    self.__age=age
    self.__roll_number=roll_number
  def get_name(self):
    return self.__name
  def get_age(self):
    return self.__age
  def get_roll_number(self):
    return self.__roll_number
  def set_name(self,name):
    self.__name=name
  def set_age(self,age):
    self.__age=age
  def set_roll_number(self,roll_number):
    self.__roll_number=roll_number


obj1=Students("shivani",20, 617)
obj1.get_name()
obj1.set_name("shivi")
obj1.get_name()
#output=
shivi
```

Question: 3

```
class LibraryBook:
  def __init__(self,bookname,author,availability_status):
    self.__bookname=bookname
    self.__author=author
    self.__availability_status=availability_status

  def borrow_book(self):
    if self.__availability_status==True:
      print("book is available")
      self.__availability_status=False
      return self.__bookname, self.__author

    else:
      print("book is not available")
      return None
  def return_book(self):
    self.__availability_status=True
    print("book is returned")
    return None

obj1=LibraryBook('python for datascience','oreilly',True)
obj2=LibraryBook('statistics for datascience','oreilly',True)
obj3=LibraryBook('data science','oreilly',True)
objs=[obj1,obj2,obj3]
for obj in objs:
  print(obj.borrow_book())
  print(obj.return_book())
print(obj1.borrow_book())
print(obj2.borrow_book())
print(obj3.borrow_book())
#output=
book is available
('python for datascience', 'oreilly')
book is returned
None
book is available
('statistics for datascience', 'oreilly')
book is returned
None
book is available
('data science', 'oreilly')
book is returned
None
book is available
```

```
('python for datascience', 'oreilly')
book is available
('statistics for datascience', 'oreilly')
book is available
('data science', 'oreilly')
```

Question:4

```

```

Question: 5

```
class Animal:
  def make_sound(self):
    print(" animal is making sound")
class Dog(Animal):
  def make_sound(self):
    print("dog barks")
class Cat(Animal):
  def make_sound(self):
    print(" cat meows")
objt1=Cat()
objt1.make_sound()
objt2=Dog()
objt2.make_sound()
#output=
cat meows
dog barks
```

Question:6
Question:7

```
class Room:
  def __init__(self,room_number,room_type,rate,availability):
    self._room_number=room_number
    self._room_type=room_type
    self._rate=rate
    self.__availability=availability
  def room_identification_number(self,room_number):
    self.__room_number
  def book_room(self):
    pass
  def check_in_guest(self):
    pass
  def check_out_guest(self):
    pass
class SuiteRoom(Room):
  def __init__(self,room_number,room_type,rate,availability,curtains):
    super().__init__(room_number,room_type,rate,availability)
    self._curtains=curtains
class StandardRoom(Room):
  def
__init__(self,room_number,room_type,rate,availability,airConditioner):
    super().__init__(room_number,room_type,rate,availability)
    self._airConditioner=airConditioner
obj1=SuiteRoom(142,'suite',1200,True,'available')
obj2=StandardRoom(126,"standard",2500,True,'AC available')
print(obj1._curtains)
print(obj1._rate)
 #output=
available
1200
```

Question:8

```
class Member:
  member_list=[]
  def __init__(self,name,age,membership_type,membership_status):
    self._name=name
    self._age=age
    self._membership_type=membership_type
    self.__membership_status=membership_status

  def new_member(self,name,age,membership_type,membership_status):
    new_member = Member(name, age, membership_type, membership_status)
    Member.member_list.append(new_member)
    return  Member.member_list

  def renew_membership(self,membership_status):
    self._membership_status=membership_status

  def cancel_membership(self,membership_status):
    self._membership_status=None
  def Identification_number(self,id_number):
    self.__id_number=id_number

  def show_details(self):
    return [self._name,self._age,self._membership_type]

class FamilyMember(Member):
  def __init__(self,name,age,membership_type,membership_status,relation):
    super().__init__(name,age,membership_type,membership_status)
    self.__relation=relation
class IndividualMember(Member):
  def __init__(self,name,age,membership_type,membership_status,gender):
    super().__init__(name, age, membership_type, membership_status)
    self.__gender=gender

  for i in  Member.member_list :
    i.show_details()
    print()

obj1=IndividualMember('shivi',20,'personal',True,'female')
print(obj1._name)
obj2=Member('neha',18,'elite',True)
obj2._age

#output=
shivi

18
```

Question:9

```
class Event:
  list_of_events=[]
  def __init__(self,name,date,time,location,list_of_attendees):
    self.name=name
    self.date=date
    self.time=time
    self.location=location
    self.__list_of_attendees=list_of_attendees

  def show(self):
    return Event.list_of_events

  def create_new_event(self,name,date,time,location,list_of_attendees):
      new_event=Event(name,date,time,location,list_of_attendees)
      Event.list_of_events.append(new_event)
      return Event.list_of_events

  def add_attendees(self,new_attendee):
      self.__list_of_attendees += new_attendee

  def remove_attendees(self,extra_attendee):
      self.__list_of_attendees.remove(extra_attendee)

  def total_attendees(self):
      return self.__list_of_attendees
  def __repr__(self):
      return f"Event('{self.name}', '{self.date}', '{self.time}',
'{self.location}', {self.__list_of_attendees})"

class PrivateEvent(Event):
  def
__init__(self,name,date,time,location,list_of_attendees,identification_num
ber,event_type):
    super().__init__(name,date,time,location,list_of_attendees)
    self.__identification_number=identification_number
    self.event_type=event_type

class PublicEvent(Event):
  def
__init__(self,name,date,time,location,list_of_attendees,identification_num
ber,event_type):
    super().__init__(name,date,time,location,list_of_attendees)
    self.__identification_number=identification_number
    self.event_type=event_type
```

```
obj1=PrivateEvent('haldi ceremony','28 dec
2023','9am','moradabad',250,112233,'private')
obj2=Event("marriage",'29 dec 2023','10pm','moradabad',1000)
obj3=obj2.create_new_event("mehndi",'29 dec 2023','6pm','moradabad',200)
obj4=Event("marriage",'29 dec 2023','10pm','moradabad',1000)
obj1.add_attendees(100)
obj2.add_attendees(100)
print(obj2.total_attendees())
print(obj1.total_attendees())
for events in Event.list_of_events:
  print(events)
#output=
1100
350
Event('mehndi', '29 dec 2023', '6pm', 'moradabad', 200)
```

Question:10

```
class Flight:
  def
__init__(self,flight_number,departure_airport,arrival_airport,departure_ti
me,arrival_time,available_seats):
    self._flight_number=flight_number
    self._departue_airport=departure_airport
    self._arrival_airport=arrival_airport
    self._deoarture_time=departure_time
    self._arrival_time=arrival_time
    self.__available_seats=available_seats
  def book_seats(self,number):
    if self.__available_seats>0:
      self.__available_seats-=number
      return "seat is available"

    else:
        print("seats are not available")
  def cancel_reservation(self,number):
     self.__available_seats+=number
     return f"reservation cancelled for {number} seats"

  def remaining_seats(self):
    return self.__available_seats
```

```
class DomesticFlight(Flight):
  def
__init__(self,flight_number,departure_airport,arrival_airport,departure_ti
me,arrival_time,available_seats,Food):

super().__init__(flight_number,departure_airport,arrival_airport,departure
_time,arrival_time,available_seats)
    self.Food=Food
class InternationalFlight(Flight):
  def
__init__(self,flight_number,departure_airport,arrival_airport,departure_ti
me,arrival_time,available_seats,Food,Blanket,Wifi):

super().__init__(flight_number,departure_airport,arrival_airport,departure
_time,arrival_time,available_seats)
    self.Food=Food
    self.Blanket=Blanket
    self.Wifi=Wifi

flight1=DomesticFlight(234,'delhi','bangalore','6am','11am',100,'yes')
print(flight1.remaining_seats())
print(flight1.book_seats(5))
print(flight1.remaining_seats())
print(flight1.cancel_reservation(2))

#output=
100
seat is available
95
reservation cancelled for 2 seats
```

Question:11

```
# this module name is constants.py
PI = 3.14159265359
E = 2.71828182846



SPEED_OF_LIGHT = 299792458
golden_ratio=1.61803
Eulers_number=2.71828
```

```
#main.py
import os ,sys
from os.path import dirname,join,abspath
parent_dir_path=abspath(join(dirname(__file__),'..'))
sys.path.insert(0,parent_dir_path)

from assignment_questions  import constants
print('the value of PI is :',constants.PI)
print('the speed of  light is :',constants.SPEED_OF_LIGHT)
print("the value of golden ratio:",constants.golden_ratio)
#output:
[Running] python -u "/config/workspace/.vscode/acces_module/main.py"
the value of PI is : 3.14159265359
the speed of  light is : 299792458
the value of golden ratio: 1.61803

[Done] exited with code=0 in 0.037 seconds
```

Question:12

```
# this module name is calculator.py
def addition(*args):
    return sum(args)
def subtract(x,y):
    return x-y
def multiply(x,y):
    return x*y
def division(x,y):
    return x/y
```

```
#main.py
import os ,sys
from os.path import dirname,join,abspath
parent_dir_path=abspath(join(dirname(__file__),'..'))
sys.path.insert(0,parent_dir_path)

from assignment_questions  import calculator
print("sum of the values:",calculator.addition(2,4,9))
print(" difference  between numbers",calculator.subtract(4,6))
print(" product of the values:",calculator.multiply(2,8))
print('division value :',calculator.division(9,3))
```

```
#output=
[Running] python -u "/config/workspace/.vscode/acces_module/main.py"
sum of the values: 15
 difference  between numbers -2
 product of the values: 16
division value : 3.0

[Done] exited with code=0 in 0.04 seconds
```

Question:13

```python
#order_processing module
def take_orders(number_of_products):
    if number_of_products>0:
        print(f"deliver {number_of_products} products to the customer ")
    else:
        print(f" cancel order for {abs(number_of_products)} products")
    return  'thankyou for trusting our brand'
```

```python
#product_mangement module
import os,sys
from os.path import dirname,join,abspath
parent_path=abspath(join(dirname(__file__),'..'))
sys.path.insert(0,parent_path)
from ecommerce import order_processing
print(order_processing.take_orders(50))
print(order_processing.take_orders(-10))
```

```
[Running] python -u
"/config/workspace/.vscode/ecommerce/product_management.py"
deliver 50 products to the customer
thankyou for trusting our brand
 cancel order for 10 products
thankyou for trusting our brand

[Done] exited with code=0 in 0.04 seconds
```

Question:14

```python
#string_utils module
rev_string=[]
```

```
def reverse(string):
    rev_string.append(string[::-1])
    return rev_string
def make_capital(string):
    return string.capitalize()
```

```
#main.py module to import sting_utils module
import os,sys
from os.path import dirname,join,abspath
shivi=abspath(join(dirname(__file__),'..'))
sys.path.insert(0,shivi)
from assignment_questions import string_utils
print(" capitalizing string :",string_utils.make_capital('shivani
virang'))
print('revrsing the given string :',string_utils.reverse("shivani"))
#output:
[Running] python -u "/config/workspace/.vscode/acces_module/main1.py"
 capitalizing string : Shivani virang
revrsing the given string : ['inavihs']

[Done] exited with code=0 in 0.035 seconds
```

Question:15

```
# file_operations module
def create_file(text_paragraphs):
    with open('paragraphs.txt','w') as file:
      file.write(text_paragraphs)
      file.close()
      return file
def read_file():
    with open('paragraphs.txt','r') as file:
     return file.read()

def add_info_to_file(new_data):
    with open('paragraphs.txt','a') as file:
      file.write(new_data)
      file.close()
    return file
```

```
import os,sys
from os.path import dirname,join,abspath
file_path=abspath(join(dirname(__file__),'..'))
sys.path.insert(0,file_path)
from  assignment_questions import file_operations
print("creating the file :",file_operations.create_file('hello my name is
shivani virang'))
print('reading the text file :',file_operations.read_file())
print("appending new data to the file:",file_operations.add_info_to_file('
\n I want to become a data scientist'))
print(' updated file content:',file_operations.read_file())
#output:
[Running] python -u "/config/workspace/.vscode/acces_module/main2.py"
creating the file : <_io.TextIOWrapper name='paragraphs.txt' mode='w'
encoding='UTF-8'>
reading the text file : hello my name is shivani virang
appending new data to the file: <_io.TextIOWrapper name='paragraphs.txt'
mode='a' encoding='UTF-8'>
 updated file content: hello my name is shivani virang
 I want to become a data scientist

[Done] exited with code=0 in 0.039 seconds
```

Question:16

```
data=open('employees.txt','w')
data.write(str({'name':'shivani','age':20,'salary':40000})+'\n')
data.close()
data=open('employees.txt','a')
data.write(str({'name':'nikhil','age':21,'salary':100000}))
data.close()
data=open('employees.txt','r')
print(data.read())
#output:
{'name': 'shivani', 'age': 20, 'salary': 40000}
{'name': 'nikhil', 'age': 21, 'salary': 100000}
```

Question:17

```
# file_operations module
text_paragraphs='''The rapid advancement of technology has revolutionized
the way we live and work.
```

```
 With the rise of artificial intelligence, machine learning, and the
Internet of Things (IoT),
we are witnessing a seismic shift in the tech landscape. From smart homes
to autonomous vehicles,
technology is transforming every aspect of our lives. The proliferation of
cloud computing, big data,
and cybersecurity has enabled businesses to operate more efficiently and
securely. Moreover,
the emergence of 5G networks and quantum computing is poised to unlock new
for innovation and growth. As technology continues to evolve at an
exponential rate, it is essential
individuals and organizations to stay ahead of the curve and harness its
power to drive progress and prosperity'''
with open('inventory.txt','w') as file:
  file.write(text_paragraphs)
  file.close()

def read_file():
    with open('inventory.txt','r') as file:
     return file.read()
```

```
import os,sys
from os.path import dirname,join,abspath
file_path=abspath(join(dirname(__file__),'..'))
sys.path.insert(0,file_path)
from  assignment_questions import file_operations
print('reading the text file :',file_operations.read_file())
#output:
[Running] python -u "/config/workspace/.vscode/acces_module/main2.py"
reading the text file : The rapid advancement of technology has
revolutionized the way we live and work.
 With the rise of artificial intelligence, machine learning, and the
Internet of Things (IoT),
we are witnessing a seismic shift in the tech landscape. From smart homes
to autonomous vehicles,
technology is transforming every aspect of our lives. The proliferation of
cloud computing, big data,
and cybersecurity has enabled businesses to operate more efficiently and
securely. Moreover,
the emergence of 5G networks and quantum computing is poised to unlock new
for innovation and growth. As technology continues to evolve at an
exponential rate, it is essential
individuals and organizations to stay ahead of the curve and harness its
power to drive progress and prosperity

[Done] exited with code=0 in 0.043 seconds
```

Question:18

```
# file_operations module
expenses_list=[{'car':700000,'grocery':20000,'clothes':10000,'makeup and
skincare':5000}]
with open('expenses.txt','w') as file:
  file.write(str(expenses_list))


def read_file():
    with open('expenses.txt','r') as file:
     return file.read()
def calculate_expenses():
  total_expenses=0
  for i in expenses_list:
    for key,value in i.items():
      total_expenses+=value
    return total_expenses
```

```
import os,sys
from os.path import dirname,join,abspath
file_path=abspath(join(dirname(__file__),'..'))
sys.path.insert(0,file_path)
from  assignment_questions import file_operations
print('reading the text file :',file_operations.read_file())
print('total_expenses :',file_operations.calculate_expenses())
#output:
[Running] python -u "/config/workspace/.vscode/acces_module/main2.py"
reading the text file : [{'car': 700000, 'grocery': 20000, 'clothes':
10000, 'makeup and skincare': 5000}]
total_expenses : 735000

[Done] exited with code=0 in 0.04 seconds
```

Question:19

```
paragraph='''The rapid advancement of technology has revolutionized the
 way we live and work. With the rise of artificial intelligence, machine
learning,
 and the Internet of Things (IoT), we are witnessing a seismic shift in
the tech landscape.'''

with open('paragraph.txt','w') as file:
    file.write(paragraph)
    file.close()

def read_file():
    with open('paragraph.txt','r') as file:
     return file.read()
```

```
import os,sys
from os.path import dirname,join,abspath
path=abspath(join(dirname(__file__),'..'))

sys.path.insert(0,path)

from assignment_questions import paragraph
print('reading the file:', paragraph.read_file())
#output:
[Running] python -u "/config/workspace/.vscode/acces_module/main3.py"
reading the file: The rapid advancement of technology has revolutionized
the
 way we live and work. With the rise of artificial intelligence, machine
learning,
 and the Internet of Things (IoT), we are witnessing a seismic shift in
the tech landscape.

[Done] exited with code=0 in 0.043 seconds
```

Question:20

Measure of central tendency: measure of central tendency is a single value that represents the whole data.

The most common measures of central tendency are:
1: mean: sum of all possible values/number of possible values
2:mode:
For odd:The median is the middle value. (n+1)/2
For even:The median is the average of the two middle values. n/2
3: the mode is the value that appears most frequently in the dataset.

Measure of dispersion: measure of dispersion is the spread of data.
The most common measures of dispersion are:
1: Range: max-min
2: variance: variance(σ2)=n∑i=1 to n (xi−μ)2
3: standard deviation: The standard deviation is the square root of the variance.
4:interquartile Range: interquartile range is the difference between the first quartile and the third quartile. IQR=Q3-Q1

Question: 21
Skewness is a statistical measure that describes the asymmetry or lack of symmetry in the distribution of the data.
Types of skewness:

Positive skewness:A distribution is positively skewed when the tail on the right side of the distribution is longer or fatter than the left side.

Negative skewness:A distribution is negatively skewed when the tail on the left side of the distribution is longer or fatter than the right side.
Zero skewness:A distribution is said to have zero skewness when it is perfectly symmetrical, meaning the left and right sides of the distribution are mirror images of each other.

Question:22
PMF: A Probability Mass Function is a function that describes the probability of each distinct value of a discrete random variable.
PDF:A Probability Density Function is a function that describes the probability density of a continuous random variable.
Difference between PDF and PMF:
Pmf:
- PMF is used for discrete random variables.
- PMF assigns a probability to each distinct value of a discrete random variable.
- PMF is summed over all possible values of the discrete random variable.
- PMF is often represented as a bar graph or histogram.

Pdf:
- PDF is used for continuous random variables.
- PDF describes the probability density of a continuous random variable.
- PDF is integrated over the entire range of the continuous random variable.
- PDF is represented as a continuous curve.

Question:23

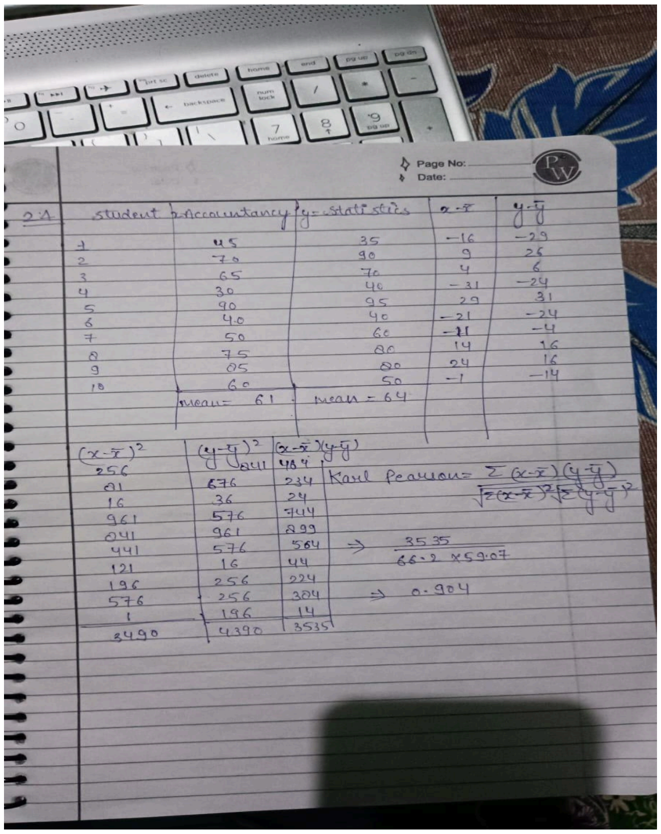
Correlation is a statistical measure that describes the strength and direction of the linear relationship between two continuous variables, X and Y. It measures how well the changes in one variable are related to the changes in another variable.
Types of correlation:

- Positive correlation:A positive correlation occurs when both variables tend to increase or decrease together.
- Negative correlation:A negative correlation occurs when one variable tends to increase while the other variable tends to decrease.
- No correlation: No correlation occurs when there is no linear relationship between the two variables.

Methods to calculate correlation:

- Pearson correlation coefficient
- Scatter plate
- Covariance
- Rank correlation
- Spearman's rank correlation coefficient
- Kendall tau's coefficient

Question:24



Page No:
Date:

24.

student	x=Accountancy	y=statistics	$x-\bar{x}$	$y-\bar{y}$
1	45	35	−16	−29
2	70	90	9	26
3	65	70	4	6
4	30	40	−31	−24
5	90	95	29	31
6	40	40	−21	−24
7	50	60	−11	−4
8	75	80	14	16
9	85	80	24	16
10	60	50	−1	−14
	mean= 61	mean = 64		

$(x-\bar{x})^2$	$(y-\bar{y})^2$	$(x-\bar{x})(y-\bar{y})$
256	841	484
81	676	234
16	36	24
961	576	744
841	961	899
441	576	504
121	16	44
196	256	224
576	256	384
1	196	14
3490	4390	3535

Karl Pearson= $\dfrac{\Sigma (x-\bar{x})(y-\bar{y})}{\sqrt{\Sigma(x-\bar{x})^2}\sqrt{\Sigma(y-\bar{y})^2}}$

$\Rightarrow \dfrac{3535}{66.2 \times 59.07}$

$\Rightarrow 0.904$

Question:25
Differences between correlation and regression:
Correlation:

- The primary purpose of correlation is to measure the strength and direction of the linear relationship between two continuous variables.
- Correlation analysis is a two way analysis.

- Correlation analysis assumes that both variables are normally distributed and that the relationship between them is linear.

Regression:

- The purpose of regression is to establish a mathematical relationship between two or more variables.
- Regression analysis is one way analysis.
- Regression analysis assumes that the dependent variable is normally distributed and the relationship between both dependent and independent variables is linear.

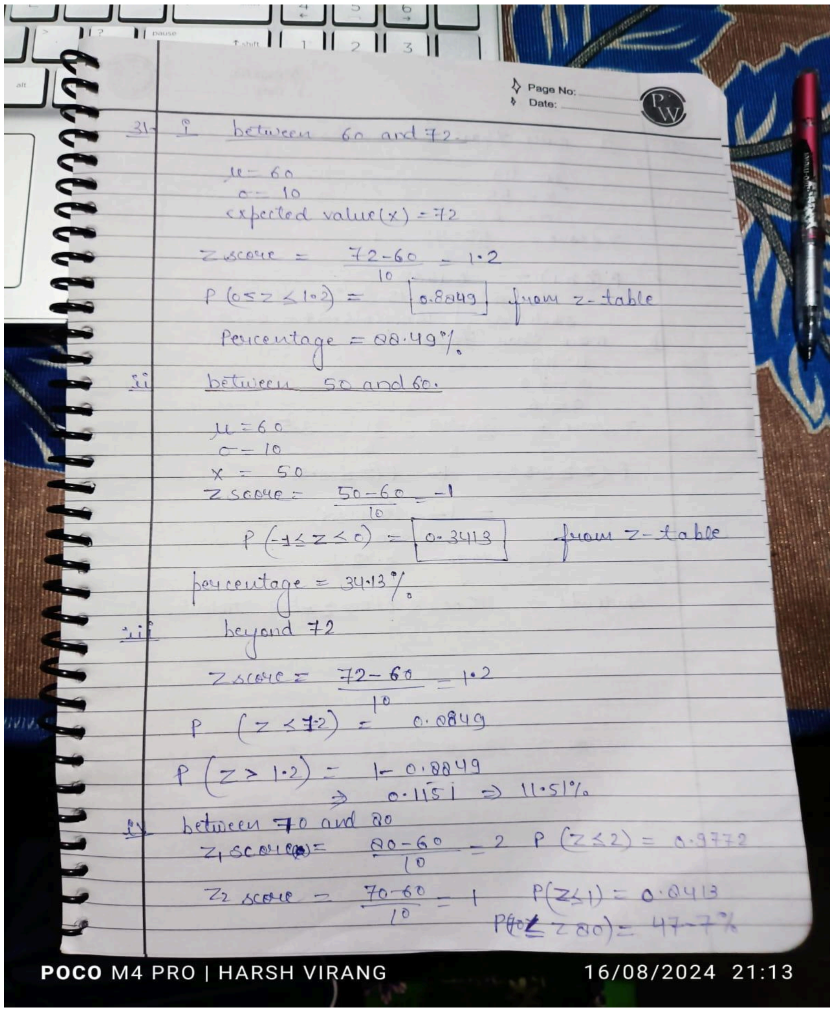
Question:26

Question:30

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

(e) Only 0.27% area is outside the range μ ±3σ.

Question:31



31. i between 60 and 72.

$\mu = 60$

$\sigma = 10$

expected value$(x) = 72$

z score $= \frac{72-60}{10} = 1.2$

$P(0 \leq z \leq 1.2) = \boxed{0.8849}$ from z-table

Percentage $= 88.49\%$.

ii between 50 and 60.

$\mu = 60$

$\sigma = 10$

$x = 50$

z score $= \frac{50-60}{10} = -1$

$P(-1 \leq z \leq 0) = \boxed{0.3413}$ from z-table

percentage $= 34.13\%$.

iii beyond 72

z score $= \frac{72-60}{10} = 1.2$

$P(z \leq 1.2) = 0.8849$

$P(z > 1.2) = 1 - 0.8849$

$\Rightarrow 0.1151 \Rightarrow 11.51\%$

iv between 70 and 80

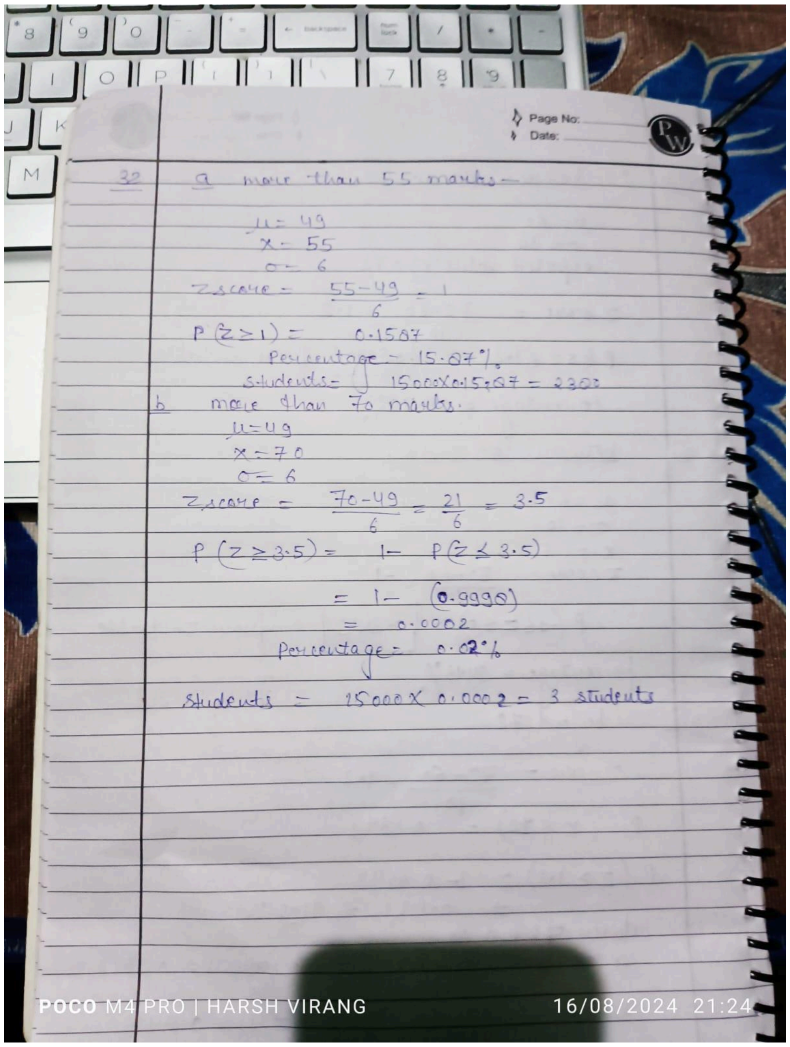
z_1 score(80) $= \frac{80-60}{10} = 2$ $P(z \leq 2) = 0.9772$

z_2 score $= \frac{70-60}{10} = 1$ $P(z \leq 1) = 0.8413$

$P(70 \leq z\ 80) = 47.7\%$

POCO M4 PRO | HARSH VIRANG 16/08/2024 21:13

Question:32


Page No:
Date:

32 a more than 55 marks –

$\mu = 49$
$x = 55$
$\sigma = 6$

Zscore = $\frac{55-49}{6} = 1$

$P(Z \geq 1) = 0.1587$

Percentage = 15.87%
students = 15000×0.1587 = 2380

b more than 70 marks.

$\mu = 49$
$x = 70$
$\sigma = 6$

Zscore = $\frac{70-49}{6} = \frac{21}{6} = 3.5$

$P(Z \geq 3.5) = 1 - P(Z \leq 3.5)$

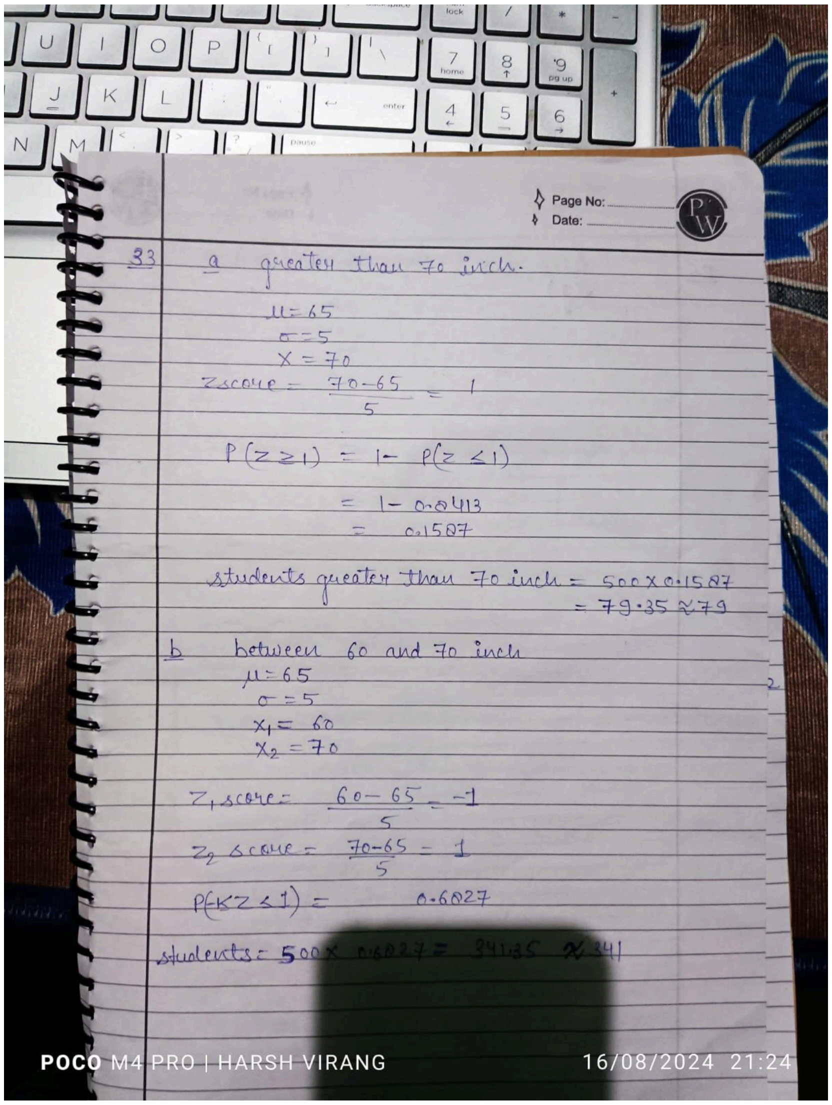
$= 1 - (0.9998)$
$= 0.0002$

Percentage = 0.02%

Students = 15000 × 0.0002 = 3 students

POCO M4 PRO | HARSH VIRANG 16/08/2024 21:24

Question:33


Page No:
Date:

33 a greater than 70 inch.

$\mu = 65$
$\sigma = 5$
$X = 70$

Zscore = $\frac{70-65}{5} = 1$

$P(Z \geq 1) = 1 - P(Z \leq 1)$
$= 1 - 0.8413$
$= 0.1587$

students greater than 70 inch = 500×0.1587
$= 79.35 \approx 79$

b between 60 and 70 inch

$\mu = 65$
$\sigma = 5$
$x_1 = 60$
$x_2 = 70$

Z_1 score = $\frac{60-65}{5} = -1$

Z_2 score = $\frac{70-65}{5} = 1$

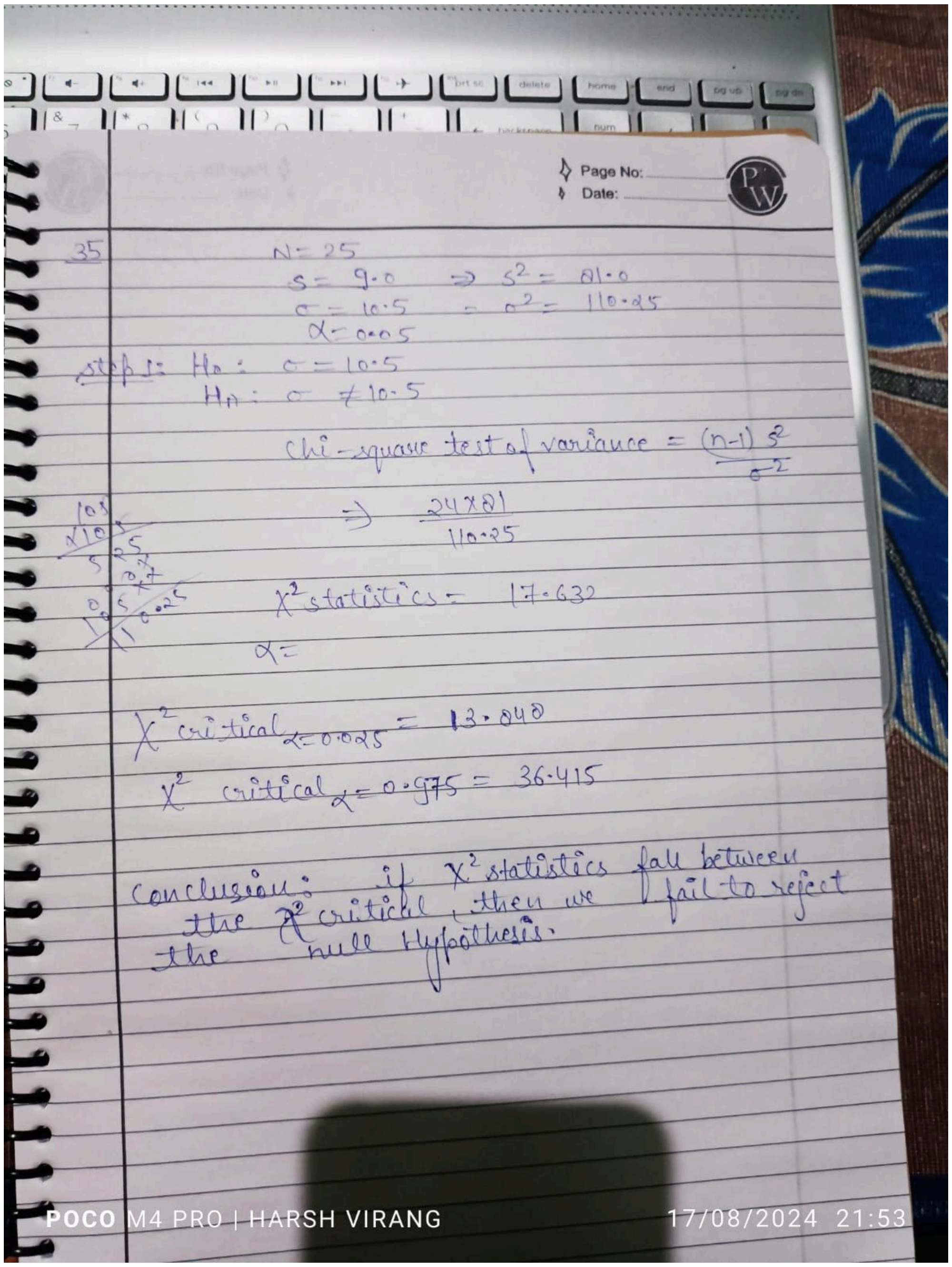
$P(-1 \leq Z \leq 1) = 0.6827$

students = $500 \times 0.6827 = 341.35 \approx 341$

POCO M4 PRO | HARSH VIRANG 16/08/2024 21:24

Question: 34
A statistical hypothesis is a statement or assumption about a population parameter (such as the mean, proportion, or variance) that can be tested using statistical methods.
Types of hypothesis:
- Null hypothesis
- Alternate hypothesis

Errors in hypothesis testing:
Type 1 error: type 1 error occured when null hypothesis is rejected when its actually True.
Type 2 error:type 2 error occured when null hypothesis is not rejected when its actually False
Question:35



Page No:
Date:

35

$N = 25$

$s = 9.0 \Rightarrow s^2 = 81.0$

$\sigma = 10.5 \Rightarrow \sigma^2 = 110.25$

$\alpha = 0.05$

step 1: $H_0 : \sigma = 10.5$

$H_A : \sigma \neq 10.5$

Chi-square test of variance $= \frac{(n-1)\, s^2}{\sigma^2}$

$\Rightarrow \frac{24 \times 81}{110.25}$

[illegible]

χ^2 statistics $= 17.632$

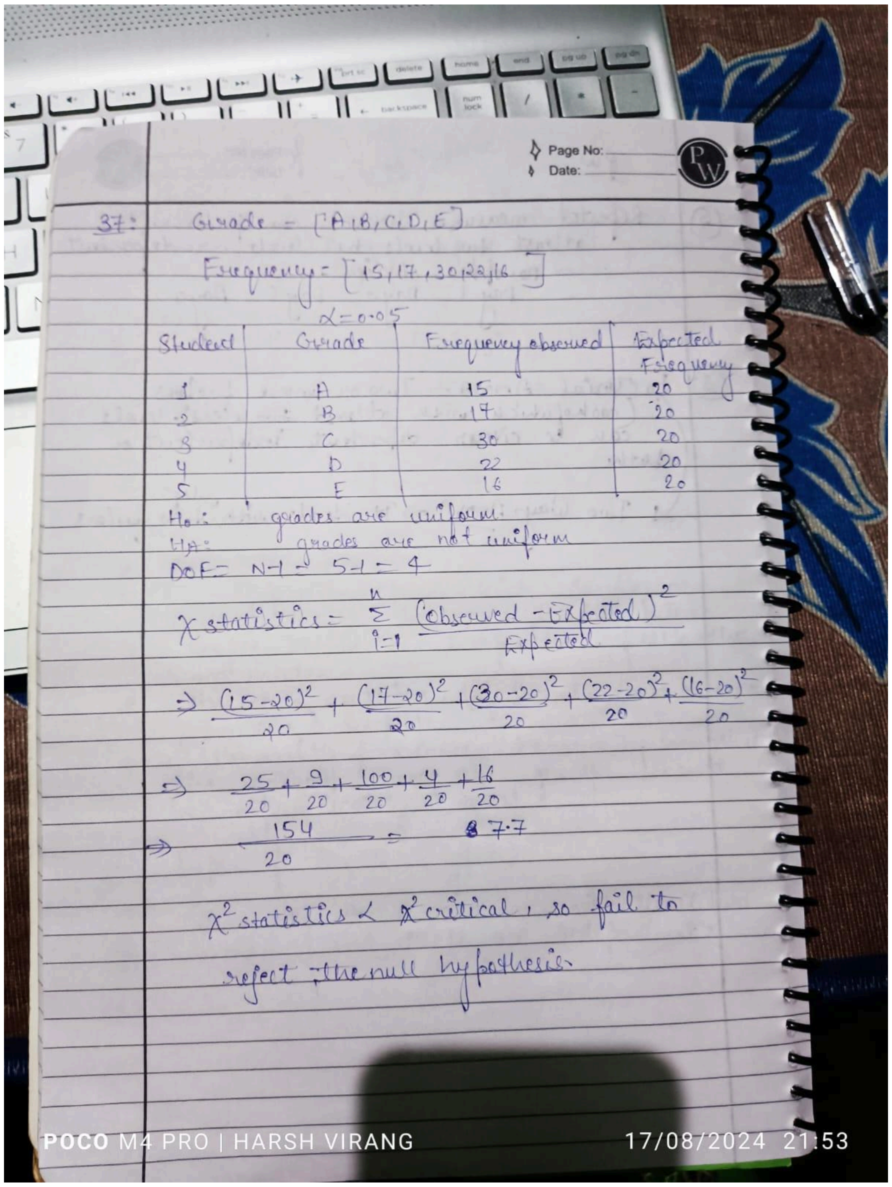
$\alpha =$

χ^2 critical$_{\alpha = 0.025} = 13.848$

χ^2 critical$_{\alpha = 0.975} = 36.415$

Conclusion: if χ^2 statistics fall between the χ^2 critical, then we fail to reject the null hypothesis.

Question:37



Page No:
Date:

37: Grade = [A, B, C, D, E]

Frequency = [15, 17, 30, 22, 16]

$\alpha = 0.05$

Student	Grade	Frequency observed	Expected Frequency
1	A	15	20
2	B	17	20
3	C	30	20
4	D	22	20
5	E	16	20

H_0: grades are uniform

H_A: grades are not uniform

DOF = N-1 = 5-1 = 4

$$\chi \text{ statistics} = \sum_{i=1}^{n} \frac{(\text{Observed} - \text{Expected})^2}{\text{Expected}}$$

$$\Rightarrow \frac{(15-20)^2}{20} + \frac{(17-20)^2}{20} + \frac{(30-20)^2}{20} + \frac{(22-20)^2}{20} + \frac{(16-20)^2}{20}$$

$$\Rightarrow \frac{25}{20} + \frac{9}{20} + \frac{100}{20} + \frac{4}{20} + \frac{16}{20}$$

$$\Rightarrow \frac{154}{20} = \cancel{8}\,7.7$$

χ^2 statistics < χ^2 critical, so fail to reject the null hypothesis.

Question:39

```
from flask import Flask

app = Flask(__name__)

@app.route('/homepage')
def homepage():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(debug=True,port=5001)
#output=
* Serving Flask app '__main__'
```

```
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a
production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5001 # after clicking on this link =>
Hello, World!
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
```

.

Question:40

```
from flask import Flask, render_template,url_for,redirect,request
app= Flask(__name__)

@app.route("/")
def home():
    return render_template("home.html")

@app.route('/login', methods=[ 'GET','POST'])
def login():
  if request.method=='POST':
    username=request.form.get('username')
    password=request.form.get('password')
    return redirect(url_for('home'))
  return render_template('login.html')


if __name__== "__main__":
    app.run(host='0.0.0.0' ,debug=True,port=5001)
```

Question:41

```
from flask import Flask, render_template,url_for,redirect,request
app= Flask(__name__)

@app.route("/")
def home():
    return "this is the home page"
@app.route("/about/<name>")
def about(name):
  return f"it's all about {name}"
```

```
if __name__== "__main__":
    app.run(host='0.0.0.0' ,debug=True,port=5001)
#output:
127.0.0.1:5001
this is the home page
127.0.0.1:5001/about/shivani
it's all about shivani
```

Question:42

```
from flask import Flask, render_template,url_for,redirect,request,abort
app= Flask(__name__)

@app.route("/")
def home():
    return " welcome !!!"

@app.route('/login/<username>', methods=[ 'GET','POST'])
def login(username):
  if username=='admin':
    return f"welcome to the website {username}"
  else:
    return abort(400)


if __name__== "__main__":
    app.run(host='0.0.0.0' ,debug=True,port=6001)
#output:
127.0.0.1:6001
welcome !!!
127.0.0.1:6001/login/admin
welcome to the website admin
400 Bad Request

Bad Request

The browser (or proxy) sent a request that this server could not
understand.
```

Question:43

Question:44

```
from flask import Flask, jsonify

app = Flask(__name__)


data = [
    {"id": 1, "name": "John", "age": 30},
    {"id": 2, "name": "Alice", "age": 25},
    {"id": 3, "name": "Bob", "age": 40}
]

@app.route('/api/users', methods=['GET'])
def get_users():
    return jsonify({'users': data})

if __name__ == '__main__':
    app.run(debug=True)
```

Question:45

```
!pip install Flask-WTF
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired

class MyForm(FlaskForm):
    name = StringField('Name', validators=[DataRequired()])
    email = StringField('Email', validators=[DataRequired()])
    submit = SubmitField('Submit')

from flask import render_template
from myapp.forms import MyForm

@app.route('/form', methods=['GET', 'POST'])
def form_view():
    form = MyForm()
    return render_template('form.html', form=form)
```

```python
@app.route('/form', methods=['GET', 'POST'])
def form_view():
    form = MyForm()
    if form.validate_on_submit():
        name = form.name.data
        email = form.email.data

        return redirect(url_for('success'))
    return render_template('form.html', form=form)
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form action="" method="post">
        {{ form.hidden_tag() }}
        {{ form.name.label }} {{ form.name(size=20) }}
        {{ form.email.label }} {{ form.email(size=20) }}
        {{ form.submit() }}
    </form>
</body>
</html>
```

Question: 46

```python
from flask import request
import os

@app.route('/upload', methods=['POST'])
def upload_file():
    if request.method == 'POST':
        file = request.files['file']
        filename = file.filename
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        return 'File uploaded successfully!'
    return 'Invalid request'
app.config['UPLOAD_FOLDER'] = '/path/to/upload/folder'
```

```
ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'])

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS
```

Question:50
1:
Difference between Series and Dataframe:
Both series and dataframe are types of data structures in python,whereas series is a one-dimensional data structure and dataframe is two-dimensional data structure.

2:
3: ‘loc’: loc method is used to select the values based on the index labels, it takes both label-based and boolean indexing.
‘iloc’: iloc method is used to select data values on the basis of integer index.

4:
In supervised learning model is trained on the labels data whereas in unsupervised model id trained on the unlabeled data.
5:
The bias-variance tradeoff is a fundamental concept in machine learning.
The bias-variance tradeoff is a delicate balance between two types of errors that occur in machine learning models: bias and variance.
6:
Precision, recall, and accuracy are three fundamental metrics used to evaluate the performance of machine learning models.
Precision:
Precision measures the proportion of true positives (correctly predicted instances) among all positive predictions made by the model.
Recall:
Recall measures the proportion of true positives among all actual positive instances in the dataset.
Accuracy:
Accuracy measures the proportion of correctly classified instances among all instances in the dataset.
7:
Overfitting occurs when a model is too complex and learns the noise in the training data, rather than the underlying patterns. As a result, the model performs well on the training data but poorly on new, unseen data. This is because the model has learned to fit the training data too closely, including the noise and random fluctuations.

- Hyperparameter tuning is one of the method to prevent overfitting.

8:

Cross-validation is a technique used to evaluate the performance of a machine learning model by training and testing it on multiple subsets of the data. The goal is to assess the model's ability to generalize to new, unseen data.

- Overfitting: Models can overfit the training data, meaning they perform well on the training data but poorly on new data. Cross-validation helps detect overfitting.
- Model selection: Cross-validation helps choose the best model among multiple candidates by evaluating their performance on unseen data.
- Hyperparameter tuning: Cross-validation is used to tune hyperparameters, such as the learning rate or regularization strength, to optimize the model's performance.

9: Classification is a type of supervised learning problem where the goal is to predict a categorical label whereas Regression is a type of supervised learning problem where the goal is to predict a continuous or numerical value.

10:Ensemble learning is a powerful technique in machine learning that combines the predictions of multiple base models to produce a more accurate and robust prediction model
11:
Gradient Descent is a popular optimization algorithm used to minimize the loss function in machine learning and it calculates the gradient of the loss with respect to each parameter and the predicted output using the current parameters.it also calculates
 the loss (or cost) between the predicted output and the actual output using a loss function.
 12: Both Batch Gradient Descent and Stochastic Gradient Descent are optimization algorithms used to minimize the loss function in machine learning.

13:The Curse of Dimensionality is a phenomenon that occurs when dealing with high-dimensional data in machine learning.
14: L1 regularization adds a penalty term to the loss function, proportional to the absolute value of the model's weights whereas L2 regularization adds a penalty term to the loss function, proportional to the square of the model's weights.
15:
A Confusion Matrix is a table used to evaluate the performance of a machine learning model or a classification algorithm.

- True Positives : The number of actual positive instances that are correctly predicted as positive.
- True Negatives : The number of actual negative instances that are correctly predicted as negative.
- False Positives : The number of actual negative instances that are incorrectly predicted as positive.
- False Negatives: The number of actual positive instances that are incorrectly predicted as negative.

16:
The AUC-ROC Curve, also known as the Receiver Operating Characteristic Curve, is a graphical representation of the performance of a binary classification model at different classification thresholds. It's a powerful tool for evaluating and comparing the performance of machine learning models.
17:
The k-Nearest Neighbors algorithm is a popular supervised learning algorithm used for classification and regression tasks.

A distance metric is used to calculate the distance between the input features of the training data and a new, unseen data point.The algorithm finds the k most similar data points to the new data point based on the distance metric.
18:
A Support Vector Machine is a powerful supervised learning algorithm used for classification and regression tasks. It's a popular choice among machine learning practitioners due to its ability to handle high-dimensional data, non-linear relationships, and noisy datasets.
The basic concept of an SVM is to find a decision boundary that maximally separates the classes in the feature space.
19:
The kernel trick is a mathematical technique used in Support Vector Machines (SVMs) to transform the original feature space into a higher-dimensional space, where it's easier to find a linear hyperplane that separates the classes.
The kernel function takes two input vectors and returns a scalar value that represents the similarity between them. By computing the kernel matrix, which is a matrix of dot products between all pairs of data points, the SVM algorithm can operate in the feature space without explicitly computing the coordinates of the data points. This allows the algorithm to handle non-linear relationships between the input features and target outputs, as well as high-dimensional data, without incurring the computational cost of explicitly computing the coordinates in the higher-dimensional space. The kernel trick is a powerful technique that enables SVMs to achieve high accuracy and flexibility in a wide range of applications, from image classification to text analysis.

20:
There are several types of kernels that can be used in Support Vector Machines:
Linear kernel:The linear kernel is the simplest kernel and is defined as the dot product of two vectors.
When we have to use this type of kernel:

- Data is linearly separable
- Number of features is small
- Computational resources are limited

Polynomial kernel:The polynomial kernel is a non-linear kernel that maps the data to a higher-dimensional space using a polynomial transformation.
When we have to use this type of kernel:

- Data is not linearly separable
- Number of features is moderate
- Degree of nonlinearity is moderate

Radial Basis Function Kernel: The radial basis function kernel is a non-linear kernel that maps the data to a higher-dimensional space using a Gaussian distribution.
When we have to use this type of kernel:

- Data is not linearly separable
- Number of features is large
- Relationship between features is complex

Sigmoid Kernel: The sigmoid kernel is a non-linear kernel that maps the data to a higher-dimensional space using a sigmoid function.
When we have to use this type of kernel:

- Data is not linearly separable
- Number of features is moderate

- Relationship between features is non-linear

Gaussian Kernel: The Gaussian kernel is a non-linear kernel that maps the data to a higher-dimensional space using a Gaussian distribution.
When we have to use this type of kernel:
- Data is not linearly separable
- Number of features is large
- Relationship between features is complex

Laplacian Kernel: The Laplacian kernel is a non-linear kernel that maps the data to a higher-dimensional space using a Laplacian distribution.
When we have to use this type of kernel:
- Data is not linearly separable
- Number of features is large
- Relationship between features is complex

Cosine Kernel: The cosine kernel is a non-linear kernel that maps the data to a higher-dimensional space using a cosine function.
When we have to use this type of kernel:
- Data is text-based
- Number of features is large
- Relationship between features is non-linear

21:
In Support Vector Machines , a hyperplane is a decision boundary that separates the data into different classes or regions. In the context of SVMs, a hyperplane is a line or a plane that maximally separates the classes in the feature space.
The hyperplane in SVMs is determined by finding the optimal separating hyperplane that maximally separates the classes in the feature space.

22:
Here are some pros and cons of using a Support Vector Machine (SVM):
Pros:
1. High accuracy: SVMs are known for their high accuracy in classification and regression tasks, especially when the data is linearly or non-linearly separable.
2. Robust to noise and outliers: SVMs are robust to noisy or outlier data points, as they focus on the margin between classes rather than individual data points.
3. Flexibility in kernel choice: SVMs allow for different kernel functions to be used, which enables them to handle various types of data and relationships.
4. Handling high-dimensional data: SVMs can handle high-dimensional data with a small number of samples, making them suitable for text classification, image recognition, and other applications.

Cons:
1. Computational complexity: Training an SVM can be computationally expensive, especially for large datasets.
2. Sensitive to parameter tuning: The performance of an SVM is highly dependent on the choice of kernel, regularization parameter, and other hyperparameters, which can be time-consuming to tune.

3. Not suitable for multi-class problems: SVMs are primarily designed for binary classification problems and can be extended to multi-class problems using techniques like one-vs-all or one-vs-one, but this can lead to increased computational complexity.
4. Not suitable for imbalanced datasets: SVMs can be biased towards the majority class in imbalanced datasets, leading to poor performance on the minority class.

23:
In Support Vector Machines (SVMs), the margin is the distance between the decision boundary (hyperplane) and the closest data points of each class.
Hard margin: the goal is to find a hyperplane that separates the classes with the maximum margin, without allowing any misclassifications. This means that all data points must be on the correct side of the hyperplane, and the margin is the distance between the hyperplane and the closest data points of each class.
Soft margin: the goal is to find a hyperplane that separates the classes with the maximum margin, while allowing some misclassifications. This means that some data points can be on the wrong side of the hyperplane, and the margin is the distance between the hyperplane and the closest data points of each class, minus the slack variables.

24:
Step 1: Data Preparation:
- Collect and preprocess the data, including feature scaling, handling missing values, and encoding categorical variables.
- Split the data into training and testing sets (e.g., 80% for training and 20% for testing).

Step 2: Root Node Selection:
- Select the root node of the decision tree, which represents the entire dataset.
- Calculate the entropy or Gini impurity of the target variable (response variable) for the entire dataset.

Step 3: Feature Selection:
- Select the best feature to split the data at the current node.
- Calculate the information gain or Gini gain for each feature.
- Choose the feature with the highest information gain or Gini gain.

Step 4: Splitting:
- Split the data into two subsets based on the selected feature and a split point (e.g., a threshold value).
- Calculate the entropy or Gini impurity of the target variable for each subset.

Step 5: Recursion:
- Recursively apply steps 3-4 to each subset until a stopping criterion is met (e.g., a maximum depth, a minimum number of samples, or a minimum improvement in entropy or Gini impurity).

Step 6: Leaf Node Creation:
- Create a leaf node for each subset that meets the stopping criterion.
- Assign a class label or predicted value to each leaf node based on the majority class or the average value of the target variable.

Step 7: Model Evaluation
- Evaluate the performance of the decision tree using metrics such as accuracy, precision, recall, F1 score, and ROC-AUC.
- Use the testing set to evaluate the model's performance

25:
The working principles of decision tree are:

1. The decision tree algorithm starts with the root node, which contains all the training data points.
2. The algorithm selects a feature to split the data at the root node and calculates the best split point.
3. The data is split into two subsets, one for each child node, based on the split point.
4. The algorithm recursively applies the splitting process to each child node until a stopping criterion is met (e.g., a maximum depth or a minimum number of samples).
5. A leaf node is created for each subset of data that meets the stopping criterion.
6. The class label or predicted value is assigned to each leaf node based on the majority class or the average value of the target variable.
7. When a new data point is input into the decision tree, it starts at the root node and traverses the tree by making decisions at each node based on the feature values.
8. The traversal continues until a leaf node is reached, which provides the predicted class label or value.

26:
Information gain is a fundamental concept in decision trees that measures the reduction in impurity or uncertainty in the data after splitting it based on a particular feature.

Step 1: Calculate Information Gain for Each Feature
At each node, the decision tree algorithm calculates the information gain for each feature in the dataset. The information gain is calculated as the difference between the entropy of the parent node and the weighted sum of the entropies of the child nodes.

Step 2: Select the Feature with the Highest Information Gain
The feature with the highest information gain is selected as the splitting feature for the current node. This is because the feature with the highest information gain is the one that best separates the classes or reduces the impurity of the data.

Step 3: Split the Data
The data is split into two subsets based on the selected feature and the split point. The split point is the value of the feature that maximizes the information gain.

Step 4: Recursively Apply the Process
The decision tree algorithm recursively applies the process to each child node until a stopping criterion is met, such as:
- A maximum depth is reached
- All instances in a node belong to the same class
- The node contains fewer than a minimum number of instances

Step 5: Create a Leaf Node
When a stopping criterion is met, a leaf node is created, and the class label or predicted value is assigned to the node based on the majority class or the average value of the target variable.

27: Gini impurity is a measure of the impurity or uncertainty of a node in a decision tree. It is a metric used to evaluate the quality of a split in a decision tree.
Gini impurity plays a crucial role in decision trees in the following ways:
1. Splitting criterion: Gini impurity is used as a splitting criterion to determine the best feature to split the data at each node. The feature that results in the lowest Gini impurity is chosen as the splitting feature.

2. Node impurity: Gini impurity is used to measure the impurity of each node in the decision tree. The node with the lowest Gini impurity is considered the most pure node.
3. Stopping criterion: Gini impurity is used as a stopping criterion to determine when to stop splitting the data. If the Gini impurity of a node is below a certain threshold, the node is considered pure, and splitting stops.
4. Feature selection: Gini impurity is used to evaluate the importance of each feature in the dataset. Features with higher Gini impurity are considered more important for classification.

28:
Advantages of Decision Trees:

1. Easy to Interpret: Decision trees are very easy to understand and interpret, even for non-technical stakeholders. The tree-like structure and the rules for splitting the data make it easy to visualize and understand the decision-making process.
2. Handling Missing Values: Decision trees can handle missing values in the data, which is a common problem in real-world datasets.

Disadvantages of Decision Trees:

1. Overfitting: Decision trees can suffer from overfitting, especially when the tree is deep and has many nodes. This means that the tree is too complex and fits the noise in the training data rather than the underlying pattern.
2. Greedy Algorithm: Decision trees use a greedy algorithm to select the best feature to split the data, which can lead to suboptimal solutions.

29:
Random forests are an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of predictions.

1. Reduced Overfitting: Decision trees can suffer from overfitting, especially when the tree is deep and has many nodes. Random forests reduce overfitting by averaging the predictions of multiple trees, which helps to smooth out the noise in the data.
2. Improved Accuracy: Random forests can improve the accuracy of predictions by combining the strengths of multiple trees. Each tree in the forest is trained on a random subset of features and samples, which helps to reduce the correlation between trees and improve overall performance.
3. Handling High-Dimensional Data: Random forests can handle high-dimensional data with a large number of features, which can be challenging for decision trees. By randomly selecting features for each tree, random forests can reduce the curse of dimensionality.
4. Robustness to Noise and Outliers: Random forests are more robust to noise and outliers in the data, as the ensemble effect helps to reduce the impact of individual noisy or outlier samples.

30:

bootstrapping refers to the process of randomly sampling the training data with replacement to create a new dataset for each decision tree in the forest. This is also known as bootstrap aggregating or bagging.

31:
Feature Importance in Random Forests:
Feature importance is a measure of how important each feature is in a random forest model. It helps identify the most relevant features that contribute to the model's predictions. In random

forests, feature importance is calculated for each feature in the dataset, and it's a crucial aspect of model interpretation and feature selection.

32:
Key Hyperparameters of a Random Forest:
Random forests have several hyperparameters that can be tuned to improve the model's performance. Here are the key hyperparameters and their effects on the model:

1. Number of Trees:
    - Controls the number of decision trees in the forest.
    - Increasing the number of trees can improve the model's accuracy and reduce overfitting, but it also increases computational cost and memory usage.
2. Maximum Depth of Trees:
    - Controls the maximum depth of each decision tree.
    - Increasing the maximum depth can improve the model's accuracy, but it also increases the risk of overfitting.
3. Number of Features to Consider at Each Split:
    - Controls the number of features to consider at each split in each decision tree.
    - Increasing the number of features can improve the model's accuracy, but it also increases the risk of overfitting.
4. Minimum Sample Size for Splitting:
    - Controls the minimum number of samples required to split an internal node.
    - Decreasing the minimum sample size can improve the model's accuracy, but it also increases the risk of overfitting.
5. Minimum Sample Size for Leaf Node:
    - Controls the minimum number of samples required to be at a leaf node.
    - Decreasing the minimum sample size can improve the model's accuracy, but it also increases the risk of overfitting.
6. Bootstrap Sampling:
    - Controls whether to use bootstrap sampling to create the training dataset for each tree.
    - Enabling bootstrap sampling can improve the model's accuracy and reduce overfitting.
7. Random State:
    - Controls the random seed used to shuffle the data and create the trees.
    - Setting a fixed random state can ensure reproducibility of the model.

How hyperparameters of random forest affects the model:

1. Overfitting vs. Underfitting: Increasing the number of trees, maximum depth, and number of features can lead to overfitting, while decreasing these hyperparameters can lead to underfitting.
2. Model Complexity: Increasing the maximum depth and number of features can increase the model's complexity, making it more prone to overfitting.
3. Computational Cost: Increasing the number of trees and maximum depth can increase the computational cost and memory usage.
4. Model Interpretability: Decreasing the maximum depth and number of features can improve model interpretability, as it reduces the complexity of the decision trees.
5. Feature Importance: The number of features to consider at each split can affect the feature importance values, as it changes the way the model selects features for splitting.

33:
Logistic regression is a type of regression analysis used for predicting the outcome of a categorical dependent variable, based on one or more predictor variables.

For logistic regression to be valid, the following assumptions should be met:

1. Linearity: The relationship between the predictor variables and the log-odds of the dependent variable should be linear.
2. Independence: Each observation should be independent of the others.
3. Homoscedasticity: The variance of the residuals should be constant across all levels of the predictor variables.
4. No or little multicollinearity: The predictor variables should not be highly correlated with each other.
5. No outliers: The data should not contain outliers that can affect the model's performance.
6. Binary dependent variable: The dependent variable should be binary (0 or 1, yes or no, etc.).
7. Large sample size: The sample size should be sufficiently large to ensure reliable estimates of the model parameters.

34:
Logistic regression is a popular machine learning algorithm for binary classification problems, where the goal is to predict the probability of an event occurring
Step 1: Data Preparation

- Collect and preprocess the data, including feature scaling and encoding categorical variables.
- Split the data into training and testing sets (e.g., 80% for training and 20% for testing).

Step 2: Model Formulation

- Define the logistic regression model as:

Step 3: Model Estimation

- The goal is to find the values of the coefficients that maximize the likelihood of observing the training data.

Step 4: Probability Estimation

- Use the estimated model parameters to calculate the probability of the event occurring for each observation in the training data:

Step 5: Model Evaluation

- Evaluate the performance of the model using metrics such as:
  - Accuracy
  - Precision
  - Recall
  - F1-score
  - ROC-AUC
- Use techniques such as cross-validation to evaluate the model's performance on unseen data.

Step 6: Prediction

- Use the trained model to make predictions on new, unseen data.
- Calculate the probability of the event occurring for each new observation.

Step 7: Classification

- Apply a threshold to the predicted probabilities to obtain binary classifications.

Step 8: Model Refining

- Refine the model by:
  - Feature engineering: adding or removing features to improve the model's performance.
  - Hyperparameter tuning: adjusting the model's hyperparameters to improve its performance.
  - Regularization: adding penalties to the model to prevent overfitting.

35:

The sigmoid function, also known as the logistic function, is a mathematical function that maps any real-valued number to a value between 0 and 1.
Here are the steps to describe how Sigmoid is used in Logistic Regression:

Step 1: Data Preparation

- Collect and preprocess the dataset, including feature scaling and encoding categorical variables.
- Split the dataset into training and testing sets.

Step 2: Model Initialization

- Set the learning rate and other hyperparameters.

Step 3: Forward Propagation

- For each training example , compute the linear combination of the input features and model parameters:

Step 4: Sigmoid Activation

- Apply the sigmoid function to the linear combination .

Step 5: Probability Interpretation

- Interpret the output as the probability of the positive class given the input features .

Step 6: Cost Function Calculation

- Calculate the cost function, typically using the binary cross-entropy loss.

Step 7: Backpropagation

- Compute the gradient of the cost function with respect to the model parameters.

Step 8: Model Update

- Update the model parameters using the gradients and the learning rate.

Step 9: Model Evaluation

- Use the trained model to make predictions on the testing set.
- Evaluate the model's performance using metrics such as accuracy, precision, recall, and F1-score.

36:
In logistic regression, the cost function (also known as the loss function or objective function) plays a crucial role in training the model to make accurate predictions. The cost function measures the difference between the model's predictions and the true labels, and its goal is to minimize this difference.

37:
38:
In logistic regression, regularization is a technique used to prevent overfitting by adding a penalty term to the loss function. The two most popular types of regularization are L1 and L2 regularization. While both methods aim to reduce overfitting, they differ in their approach and effects on the model.
L1 regularization, also known as Lasso (Least Absolute Shrinkage and Selection Operator), adds a term to the loss function that is proportional to the absolute value of the model's weights.

L2 regularization, also known as Ridge regression, adds a term to the loss function that is proportional to the square of the model's weights.
39:
XGBoost (Extreme Gradient Boosting) is a popular open-source implementation of the gradient boosting algorithm. It is a decision-tree-based ensemble learning method that is widely used for classification and regression tasks. XGBoost is known for its high performance, scalability, and interpretability.

- Gradient Boosting Machine (GBM): XGBoost is more efficient and scalable than GBM, and it provides better performance on large datasets.
- LightGBM: XGBoost is more accurate and efficient than LightGBM, and it provides better support for categorical features.
- CatBoost: XGBoost is more scalable and efficient than CatBoost, and it provides better support for missing values.

40:
Boosting is a popular ensemble learning technique that combines multiple weak models to create a strong predictive model.
Boosting is a family of algorithms that combines multiple base models to create a more accurate and robust predictive model. The idea is to iteratively train base models on the same dataset, with each subsequent model focusing on the mistakes made by the previous model. This process continues until a desired level of accuracy is achieved or a stopping criterion is met.
41:
XGBoost has a built-in mechanism to handle missing values, which is based on the concept of "missing value treatment". When XGBoost encounters a missing value, it uses the following strategies to handle it:
Default direction: XGBoost assumes that the missing value is in the default direction, which is the direction of the majority class or the mean value of the feature.
Imputation: XGBoost can impute missing values using the mean or median value of the feature.
Split-based imputation: XGBoost can also impute missing values based on the split point of the decision tree. This means that the missing value is imputed based on the value of the feature that is most correlated with the target variable.

41:
XGBoost is a powerful and popular gradient boosting algorithm that is widely used in machine learning. Hyperparameters play a crucial role in tuning the performance of XGBoost models.

1. learning_rate (eta): Learning rate determines how quickly the model learns from the data. A high learning rate can lead to overfitting, while a low learning rate can lead to underfitting. (Default: 0.3)
2. max_depth: Maximum depth of the decision tree. A higher value can lead to overfitting, while a lower value can lead to underfitting. (Default: 6)
3. min_child_weight: Minimum weight of the child nodes. A higher value can lead to underfitting, while a lower value can lead to overfitting. (Default: 1)
4. subsample: Fraction of instances to be used for training. A lower value can help reduce overfitting, while a higher value can lead to underfitting. (Default: 1)
5. colsample_bytree: Fraction of features to be used for training. A lower value can help reduce overfitting, while a higher value can lead to underfitting. (Default: 1)
6. gamma: Regularization parameter that controls the complexity of the model. A higher value can lead to underfitting, while a lower value can lead to overfitting. (Default: 0)
7. reg_alpha: L1 regularization parameter that controls the sparsity of the model. A higher value can lead to underfitting, while a lower value can lead to overfitting. (Default: 0)
8. reg_lambda: L2 regularization parameter that controls the complexity of the model. A higher value can lead to underfitting, while a lower value can lead to overfitting. (Default: 1)
9. n_estimators: Number of decision trees in the ensemble. A higher value can lead to overfitting, while a lower value can lead to underfitting. (Default: 100)
10. objective: Objective function to be optimized. Common objectives include regression, binary classification, and multi-class classification.

Effects of Hyperparameters on Model Performance:

1. Learning rate: A high learning rate can lead to overfitting, while a low learning rate can lead to underfitting.
2. Max depth: A high max depth can lead to overfitting, while a low max depth can lead to underfitting.
3. Min child weight: A high min child weight can lead to underfitting, while a low min child weight can lead to overfitting.
4. Subsample: A low subsample can help reduce overfitting, while a high subsample can lead to underfitting.
5. Colsample bytree : A low colsample bytree can help reduce overfitting, while a high colsample bytree can lead to underfitting.
6. Gamma: A high gamma can lead to underfitting, while a low gamma can lead to overfitting.
7. Reg alpha: A high reg alpha can lead to underfitting, while a low reg alpha can lead to overfitting.
8. Reg lambda: A high reg lambda can lead to underfitting, while a low reg lambda can lead to overfitting.
9. N estimators: A high n estimator can lead to overfitting, while a low n estimator can lead to underfitting.

42:
Gradient boosting is a powerful machine learning algorithm that is widely used in XGBoost.

Step 1: Initialize the Model:

The gradient boosting process starts with an initial model, which is typically a simple model such as a decision tree or a linear model. The initial model is used to make predictions on the training data.

Step 2: Calculate the Residuals:
The residuals are calculated as the difference between the actual target values and the predicted values from the initial model. The residuals represent the error or the "leftover" information that the initial model was unable to capture.

Step 3: Create a New Decision Tree:
A new decision tree is created to fit the residuals from the previous step. The decision tree is trained on the residuals, and its goal is to minimize the loss function.

Step 4: Calculate the Gradient:
The gradient is calculated as the partial derivative of the loss function with respect to the predicted values. The gradient represents the direction of the steepest descent, which is used to update the model.

Step 5: Update the Model:
The model is updated by adding the new decision tree to the ensemble. The predicted values from the new decision tree are added to the predicted values from the previous model, weighted by a learning rate.

Step 6: Repeat Steps 2-5:
Steps 2-5 are repeated multiple times, with each iteration adding a new decision tree to the ensemble. The process continues until a stopping criterion is reached, such as a maximum number of iterations or a minimum improvement in the loss function.

Step 7: Make Predictions:
Once the gradient boosting process is complete, the final model is used to make predictions on new, unseen data.
43:

1. High Accuracy: XGBoost is known for its high accuracy and performance in various machine learning competitions and real-world applications.
2. Handling Large Datasets: XGBoost can handle large datasets with millions of rows and features, making it a great choice for big data analytics.
3. Speed: XGBoost is highly optimized for speed and can train models quickly, even on large datasets.
4. Parallel Processing: XGBoost uses parallel processing, which makes it scalable and efficient for distributed computing environments.
5. Handling Missing Values: XGBoost can handle missing values in the data, which is a common problem in many real-world datasets.

Disadvantages of Using XGBoost

1. Overfitting: XGBoost can suffer from overfitting, especially when the model is complex and the training dataset is small.
2. Computational Cost: Training an XGBoost model can be computationally expensive, especially for large datasets.
3. Hyperparameter Tuning: XGBoost has a large number of hyperparameters, which can make it difficult to tune the model for optimal performance.
4. Not Suitable for Small Datasets: XGBoost is not suitable for small datasets, as it can be computationally expensive and may not provide good results.

5. Not Suitable for High-Dimensional Data: XGBoost can struggle with high-dimensional data, as it can lead to overfitting and poor performance.