Pimpri-Chinchwad Educational Trust's
Pimpri-Chinchwad College of Engineering and Research, Ravet, Pune An Autonomous Institute | NBA Accredited (4 UG Programs) | NAAC A++ Accredited | An ISO 21001:2018 Certified IQAC PCCOER

DEPARTMENT OF COMPUTER ENGINEERING

LAB MANUAL

LABORATORY PRACTICE III

BE COMPUTER

(2019 Course)

Vision of the Institute:

To be a premier institute of technical education and research to serve the need of society and all the stakeholders.

Mission of the Institute:

To establish state-of-the-art facilities to create an environment resulting in individuals who are technically sound having professionalism, research and innovative aptitude with high moral and ethical values.

Vision of the Computer Department:

To strive for excellence in the field of Computer Engineering and Research through Creative Problem Solving related to societal needs.

Mission of the Computer Department:

1. *Establish strong fundamentals, domain knowledge and skills among the students with analytical thinking, conceptual knowledge, social awareness and expertise in the latest tools & technologies to serve industrial demands.*
2. *Establish leadership skills, team spirit and high ethical values among the students to serve industrial demands and societal needs.*
3. *Guide students towards Research and Development, and a willingness to learn by connecting themselves to the global society.*

Program Specific Outcomes (PSO)

Program Specific Outcomes (PSO)
A graduate of the Computer Engineering Program will demonstrate- **PSO1**: Professional Skills- The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality project. **PSO2:** Problem-Solving Skills- The ability to understand, analyze and develop computer programs in the areas related to algorithms, software testing, application software, web design, data analytics, IOT and networking for efficient design of computer-based systems. **PSO3:** Successful Career and Entrepreneurship- The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur, and a zest for higher studies and to generate IPR & Deliver a quality project.

Program Outcomes

Students are expected to know and be able –

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet t h e specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: The problems that cannot be solved by straightforward application of knowledge, theories and techniques applicable to the engineering discipline that may not have a unique solution. For example, a design problem can be solved in many ways and lead to multiple possible solutions that require consideration of appropriate constraints/requirements not explicitly given in the problem statement. (like: cost, power requirement, durability, product life, etc.) which need to be defined (modeled) within appropriate mathematical framework. that often require use of modern computational concepts and tools.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

	Pimpri-Chinchwad Educational Trust's Pimpri-Chinchwad College of Engineering and Research, Ravet, Pune An Autonomous Institute | NBA Accredited (4 UG Programs) | NAAC A++ Accredited | An ISO 21001:2018 Certified IQAC PCCOER	
Academic Year: 2025-26	**Course Objectives and Outcomes**	**Tem** – I

Department: Computer Engineering **Class – BE**
Subject Name: Laboratory Practice III **Subject Code: [**410246**]**

Course Objectives:

1. Learn effect of data preprocessing on the performance of machine learning algorithms
2. Develop in depth understanding for implementation of the regression models.
3. Implement and evaluate supervised and unsupervised machine learning algorithms.
4. Analyze performance of an algorithm.
5. Learn how to implement algorithms that follow algorithm design strategies namely divide and conquer, greedy, dynamic programming, backtracking, branch and bound.
6. Understand and explore the working of Blockchain technology and its applications.

Course Outcomes:

	Statements
C406.1	**Apply** preprocessing techniques on datasets.
C406.2	**Implement** and **evaluate** linear regression and random forest regression models.
C406.3	**Apply** and **evaluate** classification and clustering techniques.
C406.4	**Analyze** performance of an algorithm.
C406.5	**Implement** an algorithm that follows one of the following algorithm design strategies: divide and conquer, greedy, dynamic programming, backtracking, branch and bound.
C406.6	**Interpret** the basic concepts in Blockchain technology and its applications

CO-PO Mapping:

CO\PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C406.1	3	3	3	1	2	1	-	1	2	-	2	3	3	3	2
C406.2	3	3	3	2	2	1	-	1	2	-	2	3	3	3	2
C406.3	3	3	3	2	2	2	-	1	2	-	2	3	3	3	2
C406.4	3	2	2	-	1	-	-	1	2	-	2	2	3	3	1
C406.5	3	2	3	-	1	-	-	1	2	-	-	2	3	3	1
C406.6	3	3	2	2	2	-	-	1	2	-	-	2	2	3	3
AVERAGE:	**3.00**	**2.67**	**2.67**	**1.75**	**1.67**	**1.33**		**1.00**	**2.00**		**2.00**	**2.50**	**2.83**	**3.00**	**1.83**

Group A: Design and Analysis of Algorithms

Assignment No: 1

Title: Write a program to calculate Fibonacci numbers and find its step count.

Objective: Students should be able to perform non-recursive and recursive programs to calculate Fibonacci numbers and analyze time and space complexity of non-recursive and recursive program.

Theory:

Introduction to Fibonacci numbers

- The Fibonacci series, named after Italian mathematician Leonardo Pisano Bogollo, later known as Fibonacci, is a series (sum) formed by Fibonacci numbers denoted as Fn. The numbers in Fibonacci sequence are given as: 0, 1, 1, 2, 3, 5, 8, 13, 21, 38, . . .
- In a Fibonacci series, every term is the sum of the preceding two terms, starting from 0 and 1 as first and second terms. In some old references, the term '0' might be omitted.

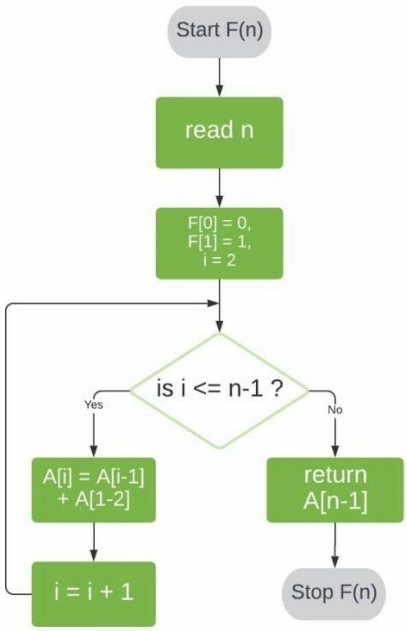
What is the Fibonacci Series?

- The **Fibonacci series** is obtained by taking the sum of the previous two numbers in the series, given that the first and second terms are 0 and 1, respectively.
- In mathematical terms, the sequence Fn of Fibonacci numbers is defined by the recurrence relation $F_n = F_{n-1} + F_{n-2}$ with seed values $F_0 = 0$ and $F_1 = 1$.

The following are different methods to get the nth Fibonacci number.

Method 1: Iterative Algorithm (Use Non-recursion)

- A simple method that is a direct recursive implementation of mathematical recurrence relation is given above. First, we'll store 0 and 1 in F[0] and F[1], respectively.
- Next, we'll iterate through array positions 2 to n-1. At each position i, we store the sum of the two preceding array values in F[i].
- Finally, we return the value of F[n-1], giving us the number at position n in the sequence.
- Here's a visual representation of this process:



 # Pseudo code to display the Fibonacci sequence up to n-th term

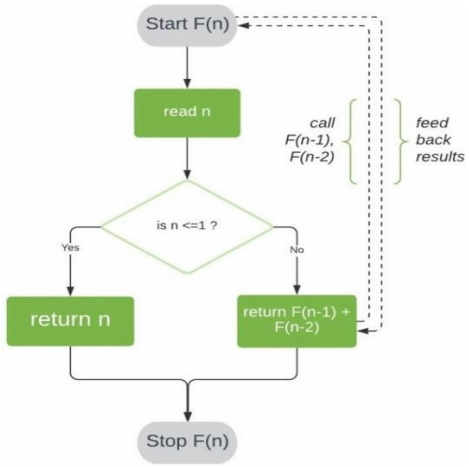
```
1   Algorithm Fibonacci(n)
2   // Compute the nth Fibonacci number.
3   {
4       if (n ≤ 1) then
5           write (n);
6       else
7       {
8           fnm2 := 0; fnm1 := 1;
9           for i := 2 to n do
10          {
11              fn := fnm1 + fnm2;
12              fnm2 := fnm1; fnm1 := fn;
13          }
14          write (fn);
15      }
16  }
```

Time and Space Complexity of Space Optimized Method

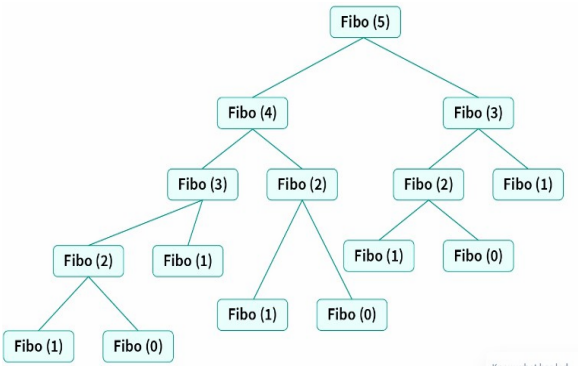
- The time complexity of the Fibonacci series is T (N) i.e, linear. We have to find the sum of two terms and it is repeated n times depending on the value of n.
- The space complexity of the Fibonacci series using dynamic programming is O (1).

Method 2: Recursive Algorithm:

- In recursion, the function calls itself until the base condition is met. To evaluate $F(n)$ for $n > 1$, we can reduce our problem into two smaller problems of the same kind: $F(n-1)$ and $F(n-2)$.
- Here's a visual representation of this algorithm:



- Here, the function rec_Fibonacci ( ) makes a call to itself. This can be easily understood by the below illustration:




```
Algorithm  rec_Fibonacci(n)

if (n <= 1)
return n;
else
return rFibonacci(n - 1) + rFibonacci(n - 2);
```

Analysis

$T(n) = T(n-1) + T(n-2) + c$
$= 2T(n-1) + c$ //from the approximation $T(n-1) \sim T(n-2)$
$= 2*(2T(n-2) + c) + c$
$= 4T(n-2) + 3c$
$= 8T(n-3) + 7c$
$= 2^k * T(n - k) + (2^k - 1)*c$
Let's find the value of k for which: $n - k = 0$ k
$= n$
$T(n) = 2^n * T(0) + (2^n - 1)*c = 2^n * (1 + c) – c = 2^n$

Time and Space Complexity

- The time complexity of the above code is T(2^N), i.e., exponential.
- The Space complexity of the above code is O(N) for a recursive series.

Using dynamic programming

- In dynamic programming, we will store all the previously calculated values of the Fibonacci numbers in **an array**. We know that the array is zero-indexed. Therefore the Fibonacci numbers are stored in the n-1 array index. For example, the 2nd Fibonacci number is stored in the 1st index of the array.

Time Complexity and Space Complexity of Dynamic Programming

- The time complexity of the above code is **T(N), i.e., linear**. We have to find the sum of two terms, and it is repeated n times depending on the value of n.
- The space complexity of the above code is **O(N)**.

Space Optimized Method

- In the previous method, we created an array to store the Fibonacci numbers but, as we need only the last two numbers to find the next one, it is **space-consuming** to store all the previously calculated numbers. Therefore we will use the space-optimized method, where just the previous two numbers are stored and using which the next number is found.

Time Complexity and Space Complexity of Space Optimized Method

- The time complexity of the Fibonacci series is **T(N) i.e., linear**. We have to find the sum of two terms, and it is repeated n times depending on the value of n.
- The space complexity of the Fibonacci series using dynamic programming is **O(1)**.

Method	Time complexity	Space complexity
Using recursion	T(n) = T(n-1) + T(n-2)	O(n)
Using DP	O(n)	O(1)
Space optimization of DP	O(n)	O(1)
Using the power of matrix method	O(n)	O(1)
Optimized matrix method	O(log n)	O(log n)
Recursive method in O(log n)time	O(log n)	O(n)
Using direct formula	O(log n)	O(1)
DP using memorization	O(n)	O(1)

Applications of Fibonacci Series

- The Fibonacci series finds application in different fields in our day-to-day lives. The differentpatterns found in a varied number of fields from nature, to music, and to the human body follow the Fibonacci series.

Some of the applications of the series are given as,

- It is used in the grouping of numbers and used to study different other special mathematical sequences.
- It finds application in Coding (computer algorithms, distributed systems, etc). For example, Fibonacci series are important in the computational run-time analysis of Euclid's algorithm, used for determining the GCF of two integers.
- It is applied in numerous fields of science like quantum mechanics, cryptography, etc.
- In finance market trading, Fibonacci retracement levels are widely used in technical analysis.

Conclusion: In this way concept of Huffman Encoding is explored using greedy method.

Viva questions:

1. What is the Fibonacci sequence of numbers?
2. How do the Fibonacci work?
3. What is the Golden Ratio?
4. What is the Fibonacci Search technique?
5. What is the real application for Fibonacci series

Assignment Questions:

1. Write algorithm/pseudo code to calculate Fibonacci numbers using non-recursive and recursive method.
2. Explain time and space complexity of the recursive implementation to calculate Fibonacci numbers.
3. Explain time and space complexity of the non-recursive implementation to calculate Fibonacci numbers.
4. Write a recurrence relation to find the nth Fibonacci number and solve it.
5. Explain improvement over the purely recursive version for iterative computation of Fibonacci numbers.(Refer page no 109 – Parag Dave Text book)
6. Explain different techniques/algorithms to make Fibonacci sequence algorithm more efficient/ optimized.

Assignment No: 2

Title: Write a program to implement Huffman Encoding using a greedy strategy.

Objective: Students should be able to understand and solve Huffman Encoding and analyze time and space complexity using a greedy strategy.

Theory:

What is a Greedy Method?

- A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.
- The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach.
- This algorithm may not produce the best result for all the problems. It's because it always goes for the local best choice to produce the global best result.

Advantages of Greedy Approach

- The algorithm is easier to describe.
- This algorithm can perform better than other algorithms (but, not in all cases).

Disadvantages of Greedy Approach

- As mentioned earlier, the greedy algorithm doesn't always produce the optimal solution. This is the major disadvantage of the algorithm
- For example, suppose we want to find the longest path in the graph below from root to leaf.

Greedy Algorithm

1. To begin with, the solution set (containing answers) is empty.
2. At each step, an item is added to the solution set until a solution is reached.
3. If the solution set is feasible, the current item is kept.
4. Else, the item is rejected and never considered again.

Huffman Encoding

- Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters.

- The most frequent character gets the smallest code and the least frequent character gets the largest code. The variable-length codes assigned to input characters are Prefix Codes means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bit stream.

There are mainly two major parts in Huffman Coding

1) Build a Huffman Tree from input characters.
2) Traverse the Huffman Tree and assign codes to characters.

Steps to build Huffman Tree

Input is array of unique characters along with their frequency of occurrences and output is Huffman Tree.

1. Create a leaf node for each unique character and build a min heap of all leaf nodes (Min Heap is used as a priority queue. The value of frequency field is used to compare two nodes in min heap.

Initially, the least frequent character is at root)

2. Extract two nodes with the minimum frequency from the min heap.

3. Create a new internal node with frequency equal to the sum of the two nodes frequencies. Make the first extracted node as its left child and the other extracted node as its right child. Add this node to the min heap.

4. Repeat steps#2 and #3 until the heap contains only one node. The remaining node is the root node and the tree is complete.

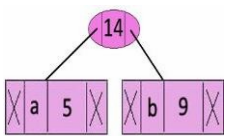
Example:

Let us understand the algorithm with an example:

character	Frequency
a	5
b	9
c	12
d	13
e	16
f	45

Step 1: Build a min heap that contains 6 nodes where each node represents root of a tree with single node.

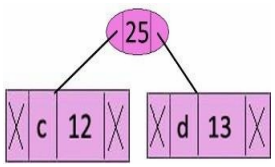
Step 2 : Extract two minimum frequency nodes from min heap. Add a new internal node with frequency 5 + 9 = 14.



Now min heap contains 5 nodes where 4 nodes are roots of trees with single element each, and one heap node is root of tree with 3 elements,

Character	Frequency
c	12
d	13
Internal Node	14
e	16
f	45

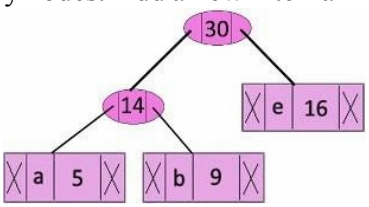
Step 3: Extract two minimum frequency nodes from heap. Add a new internal node with frequency 12 + 13 = 25



Now min heap contains 4 nodes where 2 nodes are roots of trees with single element each, and two heap nodes are root of tree with more than one nodes.

Character	Frequency
Internal Node	14
e	16
Internal Node	25
f	45

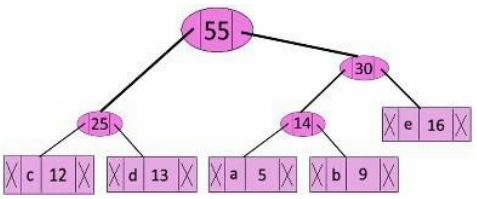
Step 4*:* Extract two minimum frequency nodes. Add a new internal node with frequency 14 + 16 = 30



Now min heap contains 3 nodes.

Character	Frequency
Internal Node	25
Internal Node	30
f	45

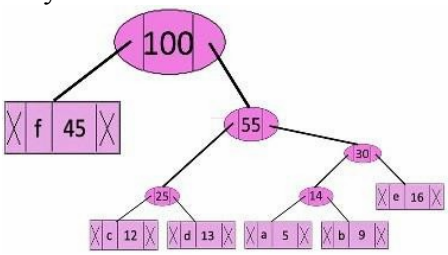
Step 5*:* Extract two minimum frequency nodes. Add a new internal node with frequency 25 + 30 = 55



Now min heap contains 2 nodes.

Character	Frequency
f	45
Internal Node	55

Step 6*:* Extract two minimum frequency nodes. Add a new internal node with frequency 45 + 55 = 100

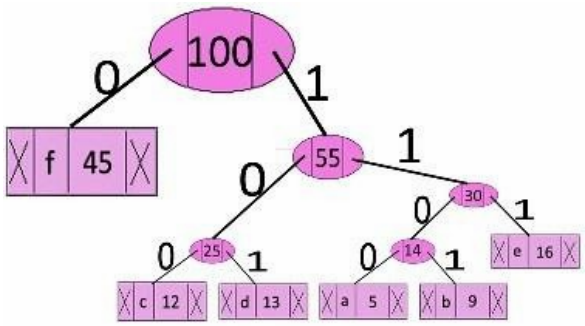


Now min heap contains only one node.
Character Frequency
Internal Node 100.

Since the heap contains only one node, the algorithm stops here.

Steps to print codes from Huffman Tree:

Traverse the tree formed starting from the root. Maintain an auxiliary array. While moving to the left child, write 0 to the array. While moving to the right child, write 1 to the array. Print the array when a leaf node is encountered.



The codes are as follows:

Character	code-word
f	0
c	100
d	101
a	1100
b	1101
e	111

Algorithm for Huffman code

Input:-Number of message with frequency count.
Output: - Huffman merge tree.

1. Begin
2. Let Q be the priority queue,
3. Q= {initialize priority queue with frequencies of all symbol or message}
4. Repeat n-1 times
5. Create a new node Z
6. X=extract_min(Q)
7. Y=extract_min(Q)

8. Frequency(Z) =Frequency(X) +Frequency(y);
9. Insert (Z, Q)
10. End repeat
11. Return (extract_min(Q))
12. End.

Time Complexity-

O(nlogn) where n is the number of unique characters. If there are n nodes, extractMin() is called 2*(n – 1) times. extractMin() takes O(logn) time as it calles minHeapify(). So, overall complexity is O(nlogn). Thus, Overall time complexity of Huffman Coding becomes O(nlogn). If the input array is sorted, there exists a linear time algorithm.

Conclusion: In this way concept of Huffman Encoding is explored using greedy method.

Viva Questions:

1. What is Huffman Encoding?
2. How many bits may be required for encoding the message ‘mississippi’?
3. Which tree is used in Huffman encoding?Give one Example
4. Why Huffman coding is lossless compression?

Assignment Questions:

1. Write algorithm/pseudo code for Huffman Encoding.
2. Explain complexity analysis of Huffman Encoding using a greedy strategy.
3. Explain the purpose of Huffman coding? What are the basic principles of Huffman coding?
4. Find Huffman code for each symbol in following text : *ABCCDEBABFFBACBEBDFAAAABCDEEDCCBFEBFCAEFBACBEBDFAAAABC*
5. How do you calculate average code length in Huffman coding? Explain with example.
6. How are Huffman coding bits calculated? Explain with example.

Assignment No: 3

Title: Write a program to solve a fractional Knapsack problem using a greedy method.

Objective: To analyze time and space complexity of fractional Knapsack problem using a greedy method.

Theory:

Knapsack Problem

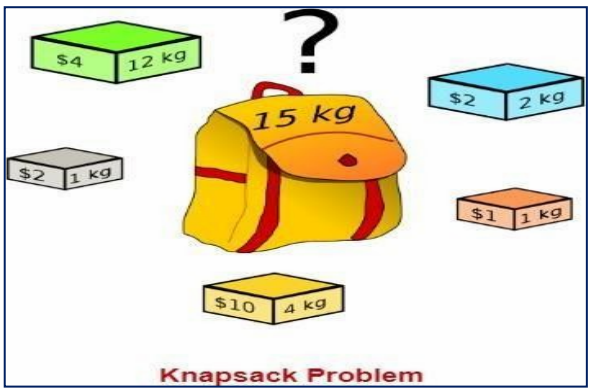
You are given the following-

- A knapsack (kind of shoulder bag) with limited weight capacity.
- Few items each having some weight and value.

The problem states-

Which items should be placed into the knapsack such that-

- The value or profit obtained by putting the items into the knapsack is maximum And the weight limit of the knapsack does not exceed.



Knapsack Problem Variants

Knapsack problem has the following two variants-

1. Fractional Knapsack Problem
2. 0/1 Knapsack Problem

Fractional Knapsack Problem

In Fractional Knapsack Problem,

- As the name suggests, items are divisible here.
- We can even put the fraction of any item into the knapsack if taking the complete item is not possible.
- It is solved using the Greedy Method.

Fractional Knapsack Problem Using Greedy Method

Fractional knapsack problem is solved using greedy method in the following steps-

Step-01: For each item, compute its value / weight ratio.

Step-02: Arrange all the items in decreasing order of their value / weight ratio.

Step-03: Start putting the items into the knapsack beginning from the item with the highest ratio. Put as many items as you can into the knapsack.

Example:

Find the optimal solution for the fractional knapsack problem making use of greedy approach. Consider-

n = 5

w = 60 kg

(w1, w2, w3, w4, w5) = (5, 10, 15, 22, 25)

(b1, b2, b3, b4, b5) = (30, 40, 45, 77, 90)

Solution-

Step-01:

Compute the value / weight ratio for each item-

Items	Weight	Value	Ratio
1	5	30	6
2	10	40	4
3	15	45	3
4	22	77	3.5
5	25	90	3.6

Step-02:

Sort all the items in decreasing order of their value / weight ratio-

I1 I2 I5 I4 I3

(6) (4) (3.6) (3.5) (3)

Step-03:

Start filling the knapsack by putting the items into it one by one.

Knapsack Weight	Items in Knapsack	Cost
60	∅	0
55	I1	30
45	I1, I2	70
20	I1, I2, I5	160

Now,

- Knapsack weight left to be filled is 20 kg but item-4 has a weight of 22 kg.
- Since in fractional knapsack problem, even the fraction of any item can be taken.
- So, knapsack will contain the following items-

< I1 , I2 , I5 , (20/22) I4 >

Total cost of the knapsack

= 160 + (20/27) x 77

= 160 + 70

= 230 units

Algorithm- Fractional knapsack

- **Greedy-fractional-knapsack (*w, v, W*)**

```
1.  for i =1 to n
2.    do x[i] =0
3.  weight = 0
4.  while weight < W
5.    do i = best remaining item
6.    if weight + w[i] ≤ W
7.      then x[i] = 1
8.      weight = weight + w[i]
9.    else
10.     x[i] = (w - weight) / w[i]
11.     weight = W
12. return x
```

Time Complexity-

- The main time taking step is the sorting of all items in decreasing order of their value / weight ratio.
- If the items are already arranged in the required order, then while loop takes O(n) time.
- The average time complexity of Quick Sort is O(nlogn).
- Therefore, total time taken including the sort is O(nlogn).

Conclusion: In this way concept of Fractional Knapsack is explained using greedy method.

Viva Questions:

1. What is Greedy Approach?
2. Explain concept of fractional knapsack
3. Difference between Fractional and 0/1 Knapsack.

Assignment Questions:

1. Explain greedy knapsack problem with example
2. Compute the optimal solution for knapsack problem using greedy method N=3, M= 20, (p1,p2,p3)= (25,24,15), (w1,w2,w3) =(18,15,10)
3. Discuss the solution for knapsack problem using branch and bound technique with example.
4. Write algorithm/pseudo code for fractional Knapsack problem using a greedy method.
5. Discuss in details time and space complexity of fractional Knapsack problem.
6. What is the time complexity of greedy knapsack if you have the objects already sorted in descending order of profit weight?
7. Why is a Greedy approach not necessarily optimal for 0-1 knapsack?

Assignment No: 4

Title: Write a program to solve a 0-1 Knapsack problem using dynamic programming strategy.

Objective: To analyze time and space complexity of 0-1 Knapsack problem using dynamic programming.

Theory:

What is Dynamic Programming?

- Dynamic Programming is also used in optimization problems. Like divide-and-conquer method, Dynamic Programming solves problems by combining the solutions of subproblems.
- Dynamic Programming algorithm solves each sub-problem just once and then saves its answer in a table, thereby avoiding the work of re-computing the answer every time.
- Two main properties of a problem suggest that the given problem can be solved using Dynamic Programming. These properties are **overlapping sub-problems and optimal substructure**.
- Dynamic Programming also combines solutions to sub-problems. It is mainly used where the solution of one sub-problem is needed repeatedly. The computed solutions are stored in a table, so that these don't have to be re-computed. Hence, this technique is needed where overlapping sub-problem exists.
- For example, Binary Search does not have overlapping sub-problem. Whereas recursive program of Fibonacci numbers have many overlapping sub-problems.

Steps of Dynamic Programming Approach

Dynamic Programming algorithm is designed using the following four steps −

- Characterize the structure of an optimal solution.
- Recursively define the value of an optimal solution.
- Compute the value of an optimal solution, typically in a bottom-up fashion.
- Construct an optimal solution from the computed information.

Applications of Dynamic Programming Approach

- Matrix Chain Multiplication
- Longest Common Subsequence

- Travelling Salesman Problem

0/1 Knapsack Problem-

In 0/1 Knapsack Problem,

- As the name suggests, items are indivisible here.
- We can not take a fraction of any item.
- We have to either take an item completely or leave it completely.
- It is solved using a dynamic programming approach.

0/1 Knapsack Problem Using Greedy Method-

Consider-

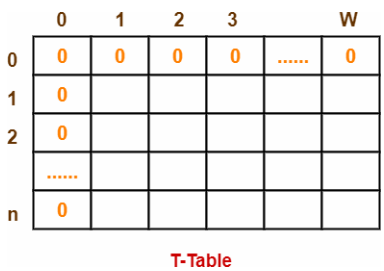
- Knapsack weight capacity = w
- Number of items each having some weight and value = n

0/1 knapsack problem is solved using dynamic programming in the following steps-

Step-01:

- Draw a table say 'T' with (n+1) number of rows and (w+1) number of columns.
- Fill all the boxes of 0th row and 0th column with zeroes as shown-

Step-02:



T-Table

Start filling the table row wise top to

bottom from left to right.Use the

following formula-

$$T(i, j) = \max \{ T(i-1, j), value_i + T(i-1, j - weight_i) \}$$

Here, T(i , j) = maximum value of the selected items if we can take items 1 to i and have weight restrictionsof j.

- This step leads to completely filling the table.
- Then, value of the last box represents the maximum possible value that can be put into the knapsack.

Step-03:

- To identify the items that must be put into the knapsack to obtain that maximum profit,
- Consider the last column of the table.
- Start scanning the entries from bottom to top.
- On encountering an entry whose value is not same as the value stored in the entry immediatelyabove it, mark the row label of that entry.
- After all the entries are scanned, the marked labels represent the items that must be put into theknapsack

Problem

For the given set of items and knapsack capacity = 5 kg, find the optimal solution for the 0/1 knapsackproblem making use of a dynamic programming approach.

Item	Weight	Value
1	2	3
2	3	4
3	4	5
4	5	6

n = 4

w = 5 kg

(w1, w2, w3, w4) = (2, 3, 4, 5)

(b1, b2, b3, b4) = (3, 4, 5, 6)

Solution-

Given

- Knapsack capacity (w) = 5 kg
- Number of items (n) = 4

Step-01:

Draw a table say 'T' with (n+1) = 4 + 1 = 5 number of rows and (w+1) = 5 + 1 = 6 number of columns. Fill all the boxes of 0th row and 0th column with 0

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					

T-Table

Step-02: Start filling the table row wise top to bottom from left to right using the formula-

$$T(i, j) = \max \{ T(i-1, j), value_i + T(i-1, j - weight_i) \}$$

Finding T(1,1)-

We have,

- i = 1
- j = 1
- $(value)_i = (value)_1 = 3$
- $(weight)_i = (weight)_1 = 2$

Substituting the values, we get-

T(1,1) = max { T(1-1 , 1) , 3 + T(1-1 , 1-2) }

T(1,1) = max { T(0,1) , 3 + T(0,-1) }

T(1,1) = T(0,1) { Ignore T(0,-1) }

T(1,1) = 0

Finding T(1,2)-

We have,

- i = 1
- j = 2
- $(value)_i = (value)_1 = 3$
- $(weight)_i = (weight)_1 = 2$

Substituting the values, we get-

T(1,2) = max { T(1-1 , 2) , 3 + T(1-1 , 2-2) }

T(1,2) = max { T(0,2) , 3 + T(0,0) }

T(1,2) = max {0 , 3+0}

T(1,2) = 3

Finding T(1,3)-

We have,

- i = 1
- j = 3
- $(value)_i = (value)_1 = 3$
- $(weight)_i = (weight)_1 = 2$

Substituting the values, we get-

T(1,3) = max { T(1-1 , 3) , 3 + T(1-1 , 3-2) }

T(1,3) = max { T(0,3) , 3 + T(0,1) }

T(1,3) = max {0 , 3+0}

T(1,3) = 3

Finding T(1,4)-

We have,

- i = 1
- j = 4
- $(value)_i = (value)_1 = 3$
- $(weight)_i = (weight)_1 = 2$

Substituting the values, we get-

T(1,4) = max { T(1-1 , 4) , 3 + T(1-1 , 4-2) }

T(1,4) = max { T(0,4) , 3 + T(0,2) }

T(1,4) = max {0 , 3+0}

T(1,4) = 3

Finding T(1,5)-

We have,

- i = 1
- j = 5
- $(value)_i = (value)_1 = 3$
- $(weight)_i = (weight)_1 = 2$

Substituting the values, we get-

T(1,5) = max { T(1-1 , 5) , 3 + T(1-1 , 5-2) }

T(1,5) = max { T(0,5) , 3 + T(0,3) }

T(1,5) = max {0 , 3+0}

T(1,5) = 3

Finding T(2,1)-

We have,

- i = 2
- j = 1
- $(value)_i = (value)_2 = 4$
- (weight)i = (weight)2 = 3

Substituting the values, we get-

T(2,1) = max { T(2-1 , 1) , 4 + T(2-1 , 1-3) }

T(2,1) = max { T(1,1) , 4 + T(1,-2) }

T(2,1) = T(1,1) { Ignore T(1,-2) }

T(2,1) = 0

Finding T(2,2)-

We have,

- i = 2
- j = 2
- $(value)_i = (value)_2 = 4$
- $(weight)_i = (weight)_2 = 3$

Substituting the values, we get-

T(2,2) = max { T(2-1 , 2) , 4 + T(2-1 , 2-3) }

T(2,2) = max { T(1,2) , 4 + T(1,-1) }

T(2,2) = T(1,2) { Ignore T(1,-1) }

T(2,2) = 3

Finding T(2,3)-

We have,

- i = 2
- j = 3
- $(value)_i = (value)_2 = 4$
- $(weight)_i = (weight)_2 = 3$

Substituting the values, we get-

T(2,3) = max { T(2-1 , 3) , 4 + T(2-1 , 3-3) }

T(2,3) = max { T(1,3) , 4 + T(1,0) }

T(2,3) = max { 3 , 4+0 }

T(2,3) = 4

Similarly, compute all the entries. After all the entries are computed and filled in the table, we get the following table-

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

T-Table

- The last entry represents the maximum possible value that can be put into the knapsack.
- So, maximum possible value that can be put into the knapsack = 7.

Identifying Items To Be Put Into Knapsack

Following Step-04,

- We mark the rows labeled "1" and "2".
- Thus, items that must be put into the knapsack to obtain the maximum value 7 are **Item-1 and Item-2**

Time Complexity-

- Each entry of the table requires constant time θ(1) for its computation.
- It takes θ(nw) time to fill (n+1)(w+1) table entries.
- It takes θ(n) time for tracing the solution since tracing process traces the n rows.
- Thus, overall θ(nw) time is taken to solve 0/1 knapsack problem using dynamic programming

Conclusion: In this way we have explored Concept of 0/1 Knapsack using Dynamic approach.

Viva Questions:

1. What is Dynamic Approach?
2. Explain concept of 0/1 knapsack
3. Difference between Dynamic and Branch and Bound Approach.Which is best?

Assignment Questions:

1. Explain 0-1 Knapsack problem with example
2. Compute the optimal solution for knapsack problem using 0-1 Knapsack Weights: {3, 4, 6, 5} Profits: {2, 3, 1, 4}, weight of the knapsack is 8 kg, number of items is 4.
3. Write algorithm/pseudo code for 0-1 Knapsack problem using a greedy method.
4. Discuss in details time and space complexity of 0-1 Knapsack problem.
5. Can knapsack be done in polynomial time? Explain.
6. Discuss: 0-1 knapsack problem is an NP-Complete problem.

Assignment No: 5

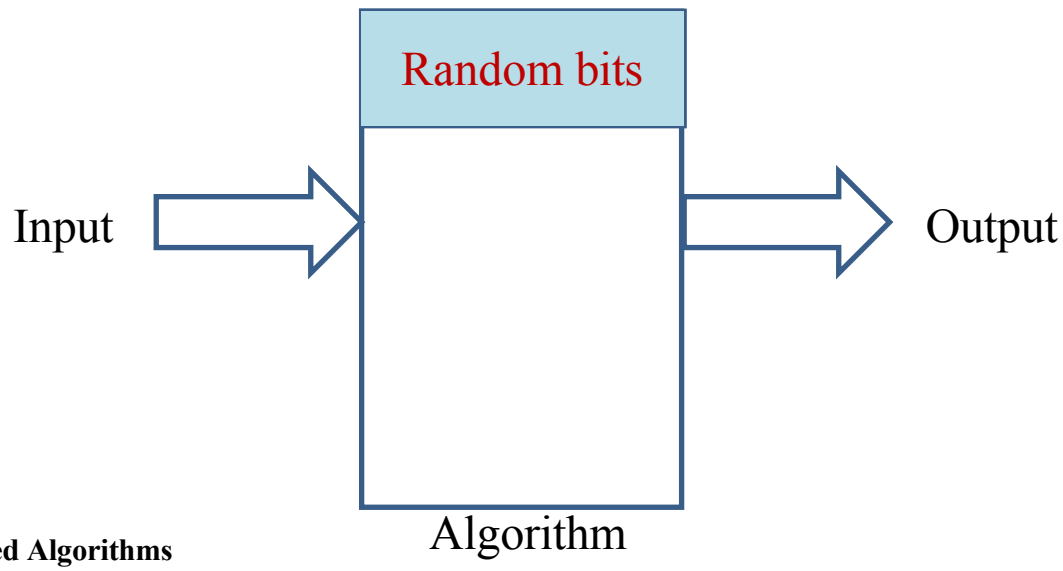
Title: Write a program for analysis of quick sort by using deterministic and randomized variant.

Objective: To analyze time and space complexity of quick sort by using deterministic and randomized variant.

Theory:

What is a Randomized Algorithm?

- An algorithm that uses random numbers to decide what to do next anywhere in its logic is called Randomized Algorithm..
- For example, in Randomized Quick Sort, we use random number to pick the next pivot (or we randomly shuffle the array).
- Typically, this randomness is used to reduce time complexity or space complexity in other standard algorithms.
- Randomized algorithm for a problem is usually simpler and **more efficient** than its deterministic counterpart.
- The output or the running time are functions of the input and random bits chosen .



Types of Randomized Algorithms

1. **Las Vegas Algorithms**

- These algorithms always produce correct or optimum result.
- Time complexity of these algorithms is based on a random value and time complexity is evaluated as expected value.
- For example, Randomized Quicksort always sorts an input array and expected worst case time complexity of Quicksort is O(nLogn).
- A Las Vegas algorithm fails with some probability, but we can tell when it fails. In particular, we can run it again until it succeeds, which means that we can eventually succeed with probability 1.
- Alternatively, we can think of a Las Vegas algorithm as an algorithm that runs for an

unpredictable amount of time but always succeeds

2. Monte Carlo Algorithms

- Produce correct or optimum result with some probability.
- These algorithms have deterministic running time and it is generally easier to find out worst case time complexity.
- For example Karger's Algorithm produces minimum cut with probability greater than or equal to $1/n^2$ (n is number of vertices) and has worst case time complexity as O(E).
- A Monte Carlo algorithm fails with some probability, but we can't tell when it fails.
- The polynomial equality-testing algorithm in an example of a Monte Carlo algorithm

Randomized Quick Sort Algorithm:

Randomized-Partition(*A, p, r*)

1. $i \leftarrow Random(p, r)$
2. exchange $A[r] \leftrightarrow A[i]$
3. **return Partition**(A, p, r)

Randomized-Quicksort(*A, p, r*)

1. **if** $p < r$

Applications of Randomized Algorithms

- Randomized algorithms have huge applications in Cryptography.
- Load Balancing.
- Number-Theoretic Applications: Primality Testing
- Data Structures: Hashing, Sorting, Searching, Order Statistics and Computational Geometry.
- Algebraic identities: Polynomial and matrix identity verification. Interactive proof systems.
- Mathematical programming: Faster algorithms for linear programming, Rounding linear program solutions to integer program solutions

Analysis of Randomized Quick sort

The running time of quicksort depends mostly on the number of comparisons performed in all calls to the Randomized-Partition routine. Let X denote the random variable counting the number of

comparisons in all calls to Randomized-Partition.

Let zi denote the i-th smallest element of A[1..n].

Thus A[1..n] sorted is <z1, z2, ... , zn >.

Let Zij = {zi, ..., zj} denote the set of elements between zi and zj, including these elements. Xij =

I{ zi is compared to zj}.

Thus, Xij is an indicator random variable for the event that the i-th smallest and the j-th smallest elements of A are compared in an execution of quicksort.

Number of Comparisons

Since each pair of elements is compared at most once by quicksort, the number X of comparisons is given by

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

Therefore, the expected number of comparisons is

$$\mathrm{E}[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathrm{E}[X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr[z_i \text{ is compared to } z_j]$$

Expected Number of Comparisons

$$\begin{aligned} \mathrm{E}[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\ &= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \\ &< \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} \\ &= \sum_{i=1}^{n-1} O(\log n) \\ &= O(n \log n) \end{aligned}$$

- It follows that the expected running time of Randomized- Quicksort is **O(n log n).**
- It is unlikely that this algorithm will choose a terribly unbalanced partition each time, so the performance is very good almost all the time.

Conclusion: In this way we have explored Concept of quick sort by using deterministic and randomized variant.

Assignment Questions:

1. Explain difference between quick sort and randomized quick sort? What is deterministic quick sort?
2. Comment on why randomized quick sort is more preferable to normal quick sort? Write pseudo code for randomized quick sort.
3. Discuss in details complexity analysis of randomized quick sort. What is overall Time

Complexity in Worst Case? Comment on stability of the same.

4. Discuss advantages of randomized algorithm? How to analyze Randomized Algorithms?
5. Which is better randomized quick sort and/or merge sort? Why?

Assignment No: 6 Mini Project

Title:

Mini Project - Write a program to implement matrix multiplication. Also implement multithreaded matrix multiplication with either one thread per row or one thread per cell. Analyze and compare their performance.

Mini Project - Implement merge sort and multithreaded merge sort. Compare time required by both the algorithms. Also analyze the performance of each algorithm for the best case and the worst case.

Mini Project - Implement the Naive string matching algorithm and Rabin-Karp algorithm for string matching. Observe difference in working of both the algorithms for the same input.

Mini Project - Different exact and approximation algorithms for Travelling-Sales-Person Problem

Objective: To implement and analyze performance of algorithm.

Theory:

1. Introduction to problems
2. Approach used to solve problem (Introduction with example)
3. Algorithm/ Pseudo code of problems
4. Complexity Analyze for all cases
5. Implementation of project with output

Group B: Machine Learning Assignment

No: 1

Title: Predict the price of the Uber ride from a given pickup point to the agreed drop-off location.
Perform following tasks:

1. **Pre-process the dataset.**
2. **Identify outliers.**
3. **Check the correlation.**
4. **Implement linear regression and random forest regression models.**
5. **Evaluate the models and compare their respective scores like R2, RMSE, etc.**

Dataset Description:
The project is about on world's largest taxi company Uber Inc. In this project, we're looking to predict the fare for their future transactional cases. Uber delivers service to lakhs of customers daily. Now it becomes really important to manage their data properly to come up with new business ideas to get best results. Eventually, it becomes really important to estimate the fare prices accurately.

Link for Dataset: https://www.kaggle.com/datasets/yasserh/uber-fares-dataset

Objective of the Assignment:

- Students should be able to preprocess dataset and identify outliers, to check correlation and Implement linear regression and random forest regression models. Evaluate them with respective scores like R2, RMSE etc.

Prerequisite:

1. Basic knowledge of Python
2. Concept of preprocessing data
3. Basic knowledge of Data Science and Big Data Analytics.

Contents of the Theory:

- Data Preprocessing
- Linear regression
- Random forest regression models
- Box Plot
- Outliers

- Haversine
- Mathplotlib
- Mean Squared Error

Data Preprocessing:

- Data preprocessing is a process of preparing the raw data and making it suitable for a machinelearning model. It is the rst and crucial step while creating a machine learning model.

- When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

Why do we need Data Preprocessing?

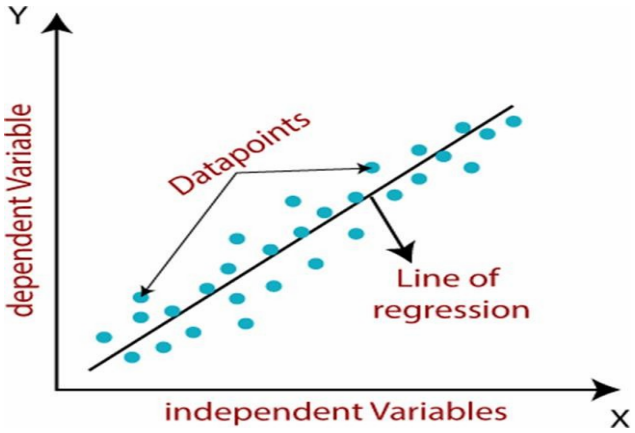
- A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy.

Linear Regression:

- Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

- Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it nds how the value of the dependent variable is changing according to the value of the independent variable.

- The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



Random Forest Regression Models:

- Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of combining multiple classifiers to solve a complexproblem and to improve the performance of the model.

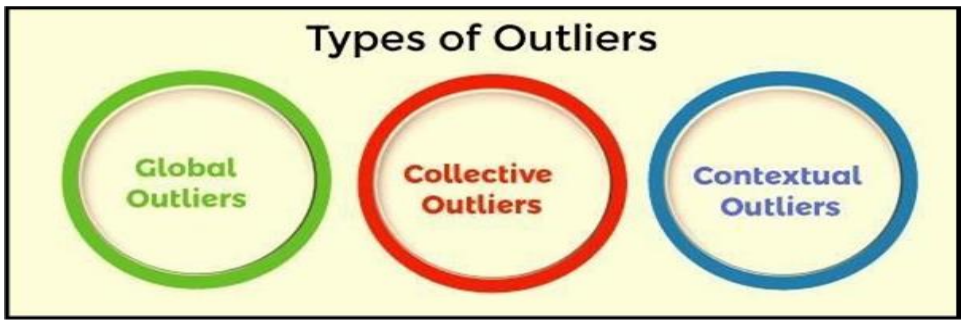
- As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.** "Instead of relying on one decision tree, the random forest takesthe

- Prediction from each tree and based on the majority votes of predictions, and it predicts the nal output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem ofover fitting.

Outlier:

- The major thing about the outliers is what you do with them. If you are going to analyze any task to analyze data sets, you will always have some assumptions based on how this data is generated. If you need some data points that are likely to contain some form of error, then these are de nitely outliers, and depending on the context, you want to overcome those errors. The data mining process involves the analysis and prediction of data that the data holds. In 1969, Grubbs introduced the rst definition of outliers.

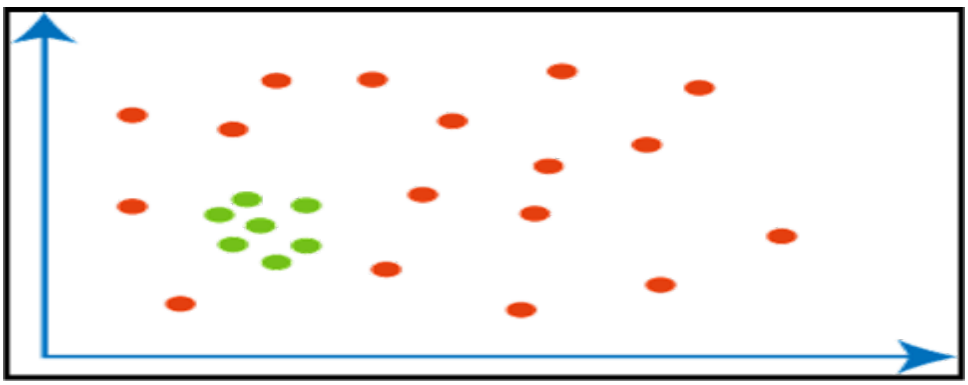


Global Outliers

- Global outliers are also called point outliers. Global outliers are taken as the simplest form of outliers. When data points deviate from all the rest of the data points in a given data set, it is known as the global outlier. In most cases, all the outlier detection procedures are targeted to determine the global outliers. The green data point is the global outlier.

Collective Outliers

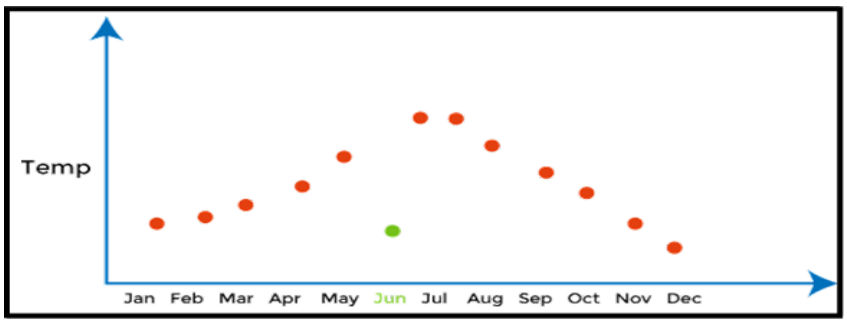
- In a given set of data, when a group of data points deviates from the rest of the data set is called collective outliers. Here, the particular set of data objects may not be outliers, but when you consider the data objects as a whole, they may behave as outliers. To identify the types of different outliers, you need to go through background information about the relationship between the behaviors of outliers shown by different data objects. For example, in an Intrusion Detection System, the DOS package from one system to another is taken as normal behavior. Therefore, if this happens with the various computers simultaneously, it is considered abnormal behavior, and as a whole, they are called collective outliers. The green data points asa whole represent the collective outlier.



Contextual Outliers

- As the name suggests, "Contextual" means this outlier introduced within a context. For example, in the speech recognition technique, the single background noise. Contextual outliers are also known as Conditional outliers. These types of outliers

happen if a data object deviates from the other data points because of any specific condition in a given data set. As we know, there are two types of attributes of objects of data: contextual attributes and behavioral attributes. Contextual outlier analysis enables the users to examine outliers in different contexts and conditions, which can be useful in various applications. For example, A temperature reading of 45 degrees Celsius may behave as an outlier in a rainy season. Still, it will behave like a normal data point in the context of a summer season. In the given diagram, a green dot representing the low-temperature value in June is a contextual outlier since the same value in December isnot an outlier.



Haversine:

- The Haversine formula calculates the shortest distance between two points on a sphere using their latitudes and longitudes measured along the surface. It is important for use in navigation.

Matplotlib:

- Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi- platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

- One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Mean Squared Error:

- The Mean Squared Error (MSE) or Mean Squared Deviation (MSD)of an estimator measures the average of error squares i.e. the average squared difference between the estimated values and true value. It is a risk function, corresponding to the expected value of the squared error loss. It is always non – negative and values close to zero are better. The MSE is the second moment of the error (about the

origin) and thus incorporates both the variance of the estimator and its bias.

Conclusion:

In this way we have explored Concept correlation and implement linear regression and random forest regression models.

Assignment Questions:

1. What is data preprocessing?
2. De ne Outliers?
3. What is Linear Regression?
4. What is Random Forest Algorithm?
5. Explain: pandas, numpy?

Assignment no:2

Title: Classify the email using the binary classification method. Email Spam detection has two states: a) Normal State – Not Spam, b) Abnormal State – Spam. Use K-Nearest Neighbors and Support Vector Machine for classification. Analyze their performance.

Dataset Description:

The csv file contains 5172 rows, each row for each email. There are 3002 columns. The first column indicates Email name. The name has been set with numbers and not recipients' name to protect privacy. The last column has the labels for prediction: 1for spam, 0 for not spam. The remaining 3000 columns are the 3000 most common words in all the emails, after excluding the non-alphabetical characters/words. For each row, the count of each word (column) in that email (row) is stored in the respective cells. Thus, information regarding all 5172 emails are stored in a compact data frame rather than as separate text files.

Link : https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv

Objective of the Assignment:

Students should be able to classify email using the binary Classification and implement email spam detection technique by using K-Nearest Neighbors and Support Vector Machine algorithm.

Prerequisite:

1. Basic knowledge of Python
2. Concept of K-Nearest Neighbors and Support Vector Machine for classification.

Contents of the Theory:

1. Data Preprocessing
2. Binary Classification
3. K-Nearest Neighbours
4. Support Vector Machine
5. Train, Test and Split Procedure

Data Preprocessing:

- Data preprocessing is a process of preparing the raw data and making it suitable for a machinelearning model. It is the rst and crucial step while creating a machine learning model.

- When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

Why do we need Data Preprocessing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machinelearning model.

It involves below steps:

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scalin

Assignment no:3

Title: Given a bank customer, build a neural network-based classifier that can determine whether they will leave or not in the next 6 months.
Dataset Description: The case study is from an open-source dataset from Kaggle.
The dataset contains 10,000 sample points with 14 distinct features such as Customer Id, Credit Score, Geography, Gender, Age, Tenure, Balance, etc.

Link to the Kaggle project: https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling
Perform following steps:
1. Read the dataset.
2. Distinguish the feature and target set and divide the data set into training and test sets.
3. Normalize the train and test data.
4. Initialize and build the model. Identify the points of improvement and implement the same.
5. Print the accuracy score and confusion matrix (5 points).

Objective of the Assignment:

Students should be able to distinguish the feature and target set and divide the data set into training and test sets and normalize them and students should build the model on the basis of that.

Prerequisite:

- Basic knowledge of Python
- Concept of Confusion Matrix

Contents of the Theory:

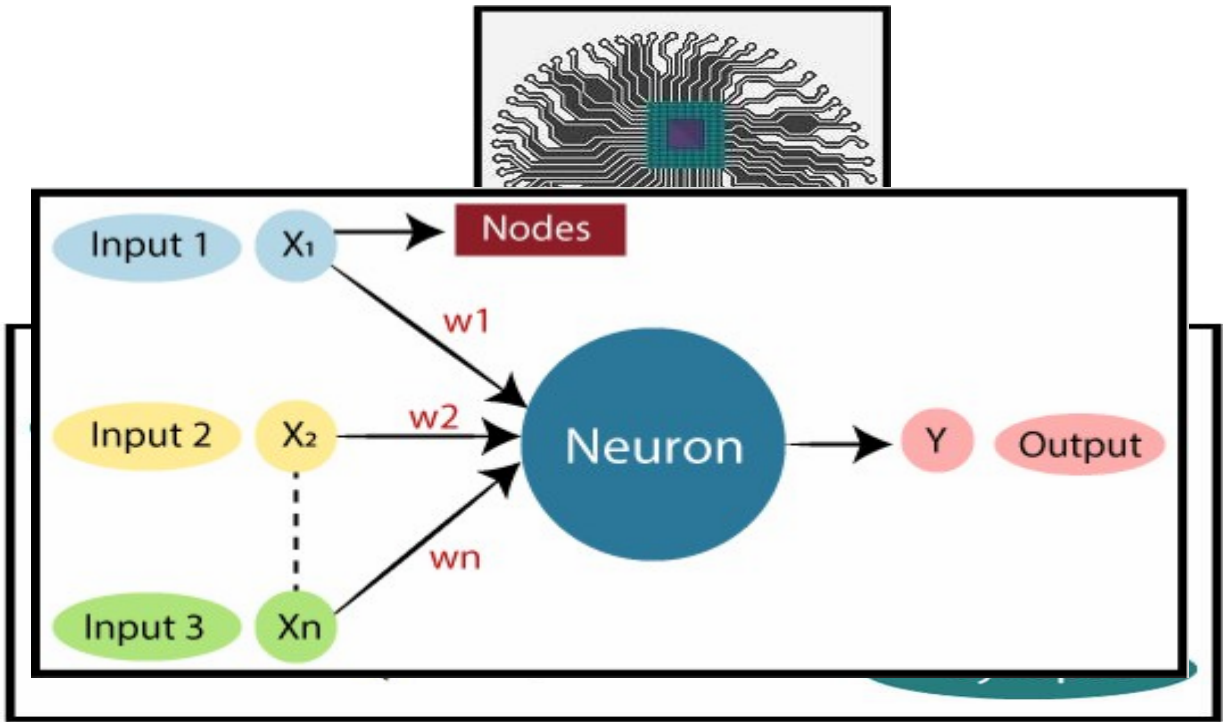
1. Artificial Neural Network
2. Keras

3. Tensor flow
4. Normalization
5. Confusion Matrix

1. Artificial Neural Network:

- The term "Artificial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.

The



given gure illustrates the typical diagram of Biological Neural Network. The

typical Artificial Neural Network looks something like the given gure.

Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus

represents Nodes, synapse represents Weights, and Axon represents Output.

Relationship between Biological neural network and artificial neural network:

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

An Artificial Neural Networking the eld of Artificial intelligence where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network's designed by programming computers to behave simply like interconnected brain cells.

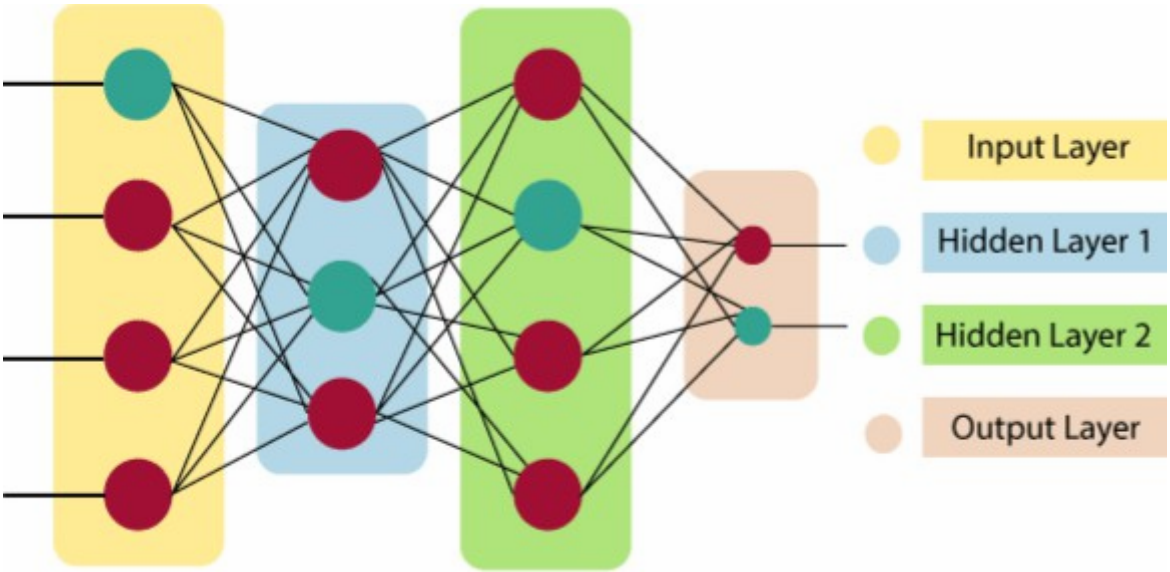
There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallels. We can say that the human brain is made up of incredibly amazing parallel processors.

We can understand the artificial neural network with an example, consider an example of a digital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," then we get "Off" in output. Here the output depends upon input. Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons in our brain, which are "learning."

The architecture of an artificial neural network:

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. In order to de ne a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Let's us look at various types of layers available in an artificial neural network.

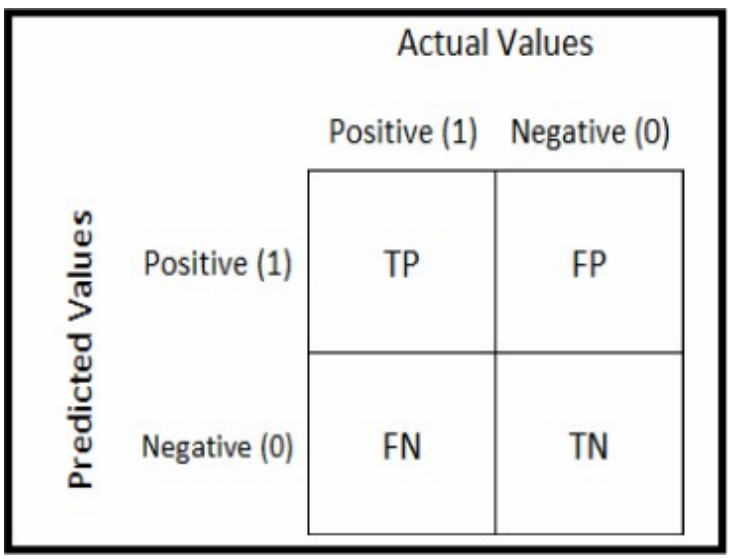
Artificial Neural Network primarily consists of three layers:



The hidden layer presents in-between input and output layers. It performs all the calculations to and hidden features and patterns. Confusion Matrix:

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an error matrix. Some features of Confusion matrix are given below:

● For the 2 prediction classes of classifiers, the matrix is of 2*2 table, for 3 classes, it is3*3 table, and so on.
● The matrix is divided into two dimensions that are predicted values and actual values along with the total number of predictions.
● Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.
● It looks like the below table:



The above table has the following cases:

● True Negative: Model has given prediction No, and the real or actual value was also No.

● True Positive: The model has predicted yes, and the actual value was also true.

● False Negative: The model has predicted no, but the actual value was Yes, it is also called as Type- II error.
● False Positive: The model has predicted Yes, but the actual value was No. It is also called a Type-I error.

Need for Confusion Matrix in Machine learning

● It evaluates the performance of the classification models, when they make predictions on test data, and tells how good our classification model is.
● It not only tells the error made by the classifiers but also the type of errors such as it is either type- I or type-II error.
● With the help of the confusion matrix, we can calculate the different parameters for the model, such as accuracy, precision, etc.

Example: We can understand the confusion matrix using an example.

Suppose we are trying to create a model that can predict the result for the disease that is either a person has that disease or not. So, the confusion matrix for this is given as:

n = 100	Actual: No	Actual: Yes	
Predicted: No	TN: 65	FP: 3	68
Predicted: Yes	FN: 8	TP: 24	32
	73	27	

From the above example, we can conclude that:

● The table is given for the two-class classifier, which has two predictions "Yes" and "NO." Here, Yes defines that patient has the disease, and No defines that patient does not has that disease.
● The classifier has made a total of100 predictions. Out of 100 predictions, 89 are true predictions, and11 are incorrect predictions.
● The model has given prediction "yes" for 32 times, and "No" for 68 times. Whereas the actual "Yes"

was 27, and actual "No" was 73 times.

Calculations using Confusion Matrix:

We can perform various calculations for the model, such as the model's accuracy, using this matrix. These calculations are given below:

● Classification Accuracy: It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classical predictions made by the classifiers. The formula is given below:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

● Misclassification rate: It is also termed as Error rate, and it denotes how often the model gives the wrong predictions. The value of error rate can be calculated as the number of incorrect predictions to all number of the predictions made by the classifier. The formula is given below.

$$\text{Error rate} = \frac{FP+FN}{TP+FP+FN+TN}$$

● Precision: It can be de need as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many of them were actually true. It can be calculate using the below formula

$$\text{Precision} = \frac{TP}{TP+FP}$$

● Recall: It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

$$\text{Recall} = \frac{TP}{TP+FN}$$

● F-measure: If two models have low precision and high recall or vice versa, it is di cult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate the recall and

precision at the same time. The F-score is maximum if the recall is equal to the precision. It can be calculated using the below formula:

$$\text{F-measure} = \frac{2*Recall*Precision}{Recall+Precision}$$

Other important terms used in Confusion Matrix:

● Null Error rate: It defines how often our model would be incorrect if it always predicted the majority class. As per the accuracy paradox, it is said that "the best classic error rate than the null error rate." has a higher

● ROC Curve: The ROC is a graph displaying a classifier's performance for all possible thresholds. The graph is plotted between the true positive rate (on the Y-axis) and the false Positive rate (on the x- axis).

Conclusion:

In this way we build a neural network-based classic the next 6 months Assignment

Questions:

1) What is Normalization?
2) What is Standardization?
3) Explain Confusion Matrix?
4) De ne the following: Classification Accuracy, Misclassification Rate, Precision.
5) One Example of Confusion Matrix?

Practical: 4

Title: Implement K-Nearest Neighbors algorithm on diabetes.csv dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset.

Dataset link : https://www.kaggle.com/datasets/abdallamahgoub/diabetes

Dataset Description:

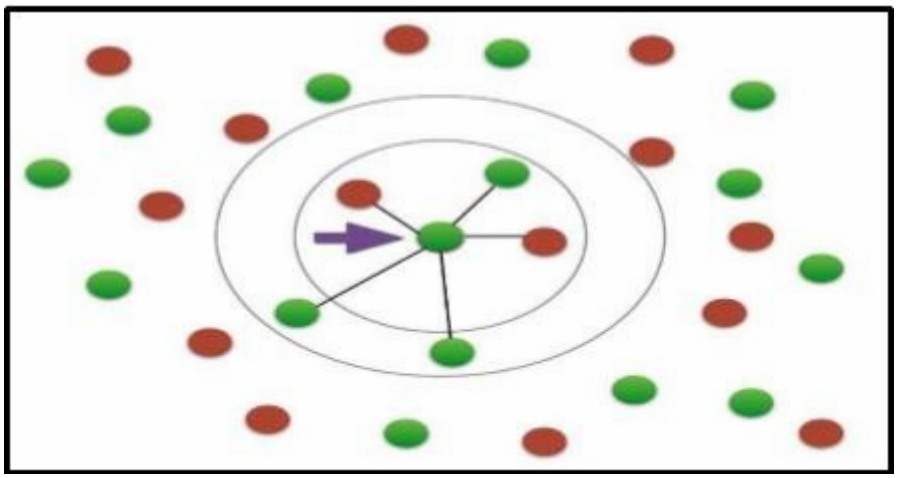
We will try to build a machine learning model to accurately predict whether or not the patients in the dataset have diabetes or not? The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Objective of the Assignment:

Students should be able to preprocess dataset and identify outliers, to check correlation and implement KNN algorithm and random forest classification models. Evaluate them with respective scores like confusion matrix, accuracy score, mean_squared_error, r2_score, roc_auc_score, roc_curve etc.

Prerequisite:

1. Basic knowledge of Python
2. Concept of Confusion Matrix
3. Concept of roc_auc curve.
4. Concept of Random Forest and KNN algorithms



K-Nearest-Neighbors (k-NN) is a supervised machine learning model. Supervised learning is when a model learns from data that is already labeled. A supervised learning model takes in a set of input objects and output values. The model then trains on that data to learn how to map the inputs to the desired output so it can learn to make predictions on unseen data.

kNN models work by taking a data point and looking at the 'k' closest labeled data points. The data point is then assigned the label of the majority of the 'k' closest points. For example, if k = 5, and 3 of points are 'green' and 2 are 'red', then the data point in question would be labeled 'green', since 'green' is the majority (as shown in the above graph). Scikit-learn is a machine learning library for Python. In this tutorial, we will build a k-NN model using Scikit- learn to predict whether or not a patient has diabetes.

Reading in the training data:

For our k-NN model, the first step is to read in the data we will use as input. For this example, we are Using the diabetes dataset. To start, we will use Pandas to read in the data. I will not go into detail on Pandas, but it is a library you should become familiar with if you're looking to dive further into data Science and machine learning.

import pandas as pd#read in the data using pandas
df = pd.read_csv('data/diabetes_data.csv')#check data has been read in properly df.head()

	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age	diabetes
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Next, let's see how much data we have. We will call the 'shape' function on our data frame to see how many rows and columns there are in our data. The rows indicate the number of patients and the columns indicate the number of features (age, weight, etc.) in the dataset for each patient.

#check number of rows and columns in dataset
df.shape
Op → (768,9)

We can see that we have 768 rows of data (potential diabetes patients) and 9 columns (8 input features and 1 target output).

Split up the dataset into inputs and targets

Now let's split up our dataset into inputs (X) and our target (y). Our input will be every column except 'Diabetes' because 'diabetes' is what we will be attempting to predict. Therefore, 'diabetes' will be our Target.

We will use pandas 'drop' function to drop the column 'diabetes' from our data frame and store it in the variable 'X'. This will be our input.

#create a dataframe with all training data except the target column
X = df.drop(columns=['diabetes'])#check that the target variable has been removed X.head()

	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

We will insert the 'diabetes' column of our dataset into our target variable (y).

#separate target values
y = df['diabetes'].values#view target values
y[0:5]

```
array([1, 0, 1, 0, 1])
```

Split the dataset into train and test data

Now we will split the dataset into training data and testing data. The training data is the data that
The model will learn from. The testing data is the data we will use to see how well the model performs On unseen data.

Scikit-learn has a function we can use called 'train_test_split' that makes it easy for us to split our dataset into training and testing data.

from sklearn.model_selection import train_test_split#split dataset into train and test data X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)

'train_test_split' takes in 5 parameters. The first two parameters are the input and target data we split up earlier. Next, we will set 'test_size' to 0.2. This means that 20% of all the data will be used for testing, which leaves 80% of the data as training data for the model to learn from. Setting 'random_state' to 1 ensures that we get the same split each time so we can reproduce our results.

Setting 'stratify' to y makes our training split represent the proportion of each value in the y variable. For example, in our dataset, if 25% of patients have diabetes and 75% don't have diabetes, setting 'stratify' to y will ensure that the random split has 25% of patients with diabetes and 75% of patients without diabetes.

Building and training the model

Next, we have to build the model. Here is the code:
from sklearn.neighbors
import KNeighborsClassifier# Create KNN classifier
knn = KNeighborsClassifier(n_neighbors = 3)# Fit the classifier to the data knn.fit(X_train,y_train)

First, we will create a new k-NN classifier and set 'n_neighbors' to 3. To recap, this means that if at least 2 out of the 3 nearest points to an new data point are patients without diabetes, then the new data point will be labeled as 'no diabetes', and vice versa. In other words, a new data point is labeled with by majority from the 3 nearest points.

We have set 'n_neighbors' to 3 as a starting point. We will go into more detail below on how to better select a value for 'n_neighbors' so that the model can improve its performance.

Next, we need to train the model. In order to train our new model, we will use the 'fit' function and pass in our training data as parameters to fit our model to the training data.

Testing the model

Once the model is trained, we can use the 'predict' function on our model to make predictions on our test data. As seen when inspecting 'y' earlier, 0 indicates that the patient does not have diabetes and 1 indicates that the patient does have diabetes. To save space, we will only show print the first 5 predictions of our test set.

#show first 5 model predictions on the test data
knn.predict(X_test)[0:5]

```
array([0, 0, 0, 0, 1])
```

We can see that the model predicted 'no diabetes' for the first 4 patients in the test set and 'has diabetes' for the 5th patient.

Now let's see how our accurate our model is on the full test set. To do this, we will use the 'score' function and pass in our test input and target data to see how well our model predictions match up to the actual results.

#check accuracy of our model on the test data
knn.score(X_test, y_test)

```
0.6688311688311688
```

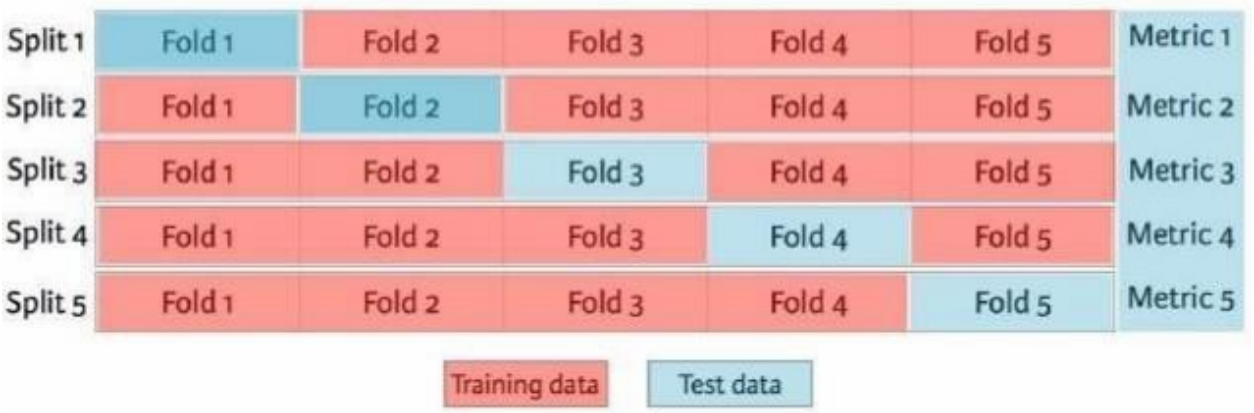
Our model has an accuracy of approximately 66.88%. It's a good start, but we will see how we can Increase model performance below.

Congrats! You have now built an amazing k-NN model!

K-Fold Cross-Validation

Cross-validation is when the dataset is randomly split up into 'k' groups. One of the groups is used as The test set and the rest are used as the training set. The model is trained on the training set and Scored on the test set. Then the process is repeated until each unique group has been used as the test Set.

For example, for 5-fold cross validation, the dataset would be split into 5 groups, and the model would be trained and tested 5 separate times so each group would get a chance to be the test set. This can be seen in the graph below.



4- Fold cross validation

The train-test-split method we used in earlier is called 'holdout'. Cross-validation is better than using the holdout method because the holdout method score is dependent on how the data is split into train and test sets. Cross-validation gives the model an opportunity to test on multiple splits so we can get a better idea on how the model will perform on unseen data.

In order to train and test our model using cross-validation, we will use the 'cross_val_score' function with a cross-validation value of 5. 'cross_val_score' takes in our k-NN model and our data as parameters. Then it splits our data into 5 groups and fits and scores our data 5 separate times, recording the accuracy score in an array each time. We will save the accuracy scores in the 'cv_scores' variable.

To find the average of the 5 scores, we will use numpy's mean function, passing in 'cv_score'. Numpy is a useful math library in Python.

```
from sklearn.model_selection import cross_val_score
import numpy as np#create a new KNN model
knn_cv = KNeighborsClassifier(n_neighbors=3)#train model with cv of 5
cv_scores = cross_val_score(knn_cv, X, y, cv=5)#print each cv score (accuracy) and average them
print(cv_scores)
print('cv_scores mean:{}'.format(np.mean(cv_scores)))
```

Bharati Vidyapeeth's College Of Engineering Lavale Pune.

Using cross-validation, our mean score is about 71.36%. This is a more accurate representation of How our model will perform on unseen data than our earlier testing using the holdout method.

Hyper tuning model parameters using GridSearchCV

When built our initial k-NN model, we set the parameter ‘n_neighbors’ to 3 as a starting point with no real logic behind that choice.

Hyper tuning parameters is when you go through a process to find the optimal parameters for your Model to improve accuracy. In our case, we will use GridSearchCV to find the optimal value for ‘n_neighbors’.

GridSearchCV works by training our model multiple times on a range of parameters that we specify. That way, we can test our model with each parameter and figure out the optimal values to get the best accuracy results.

For our model, we will specify a range of values for ‘n_neighbors’ in order to see which value works best for our model. To do this, we will create a dictionary, setting ‘n_neighbors’ as the key and using numpy to create an array of values from 1 to 24.

Our new model using grid search will take in a new k-NN classifier, our param_grid and a cross-validation value of 5 in order to find the optimal value for ‘n_neighbors’.

from sklearn.model_selection import GridSearchCV#create new a knn model
knn2 = KNeighborsClassifier()#create a dictionary of all values we want to test for n_neighbors
param_grid = {‘n_neighbors’: np.arange(1, 25)}#use gridsearch to test all values for n_neighbors
knn_gscv = GridSearchCV(knn2, param_grid, cv=5)#fit model to data knn_gscv.fit(X, y)

After training, we can check which of our values for ‘n_neighbors’ that we tested performed the best. To do this, we will call ‘best_params_’ on our model.

#check top performing n_neighbors value
knn_gscv.best_params_

```
{'n_neighbors': 14}
```

We can see that 14 is the optimal value for ‘n_neighbors’. We can use the ‘best_score_’ function to check the accuracy of our model when ‘n_neighbors’ is 14. ‘best_score_’ outputs the mean accuracy of the scores obtained through cross validation.

#check mean score for the top performing value of n_neighbors
knn_gscv.best_score_

```
0.7578125
```

Conclusion:

In this way we build a neural network-based classifier that can determine whether they will leave or not in the next 6 months.

Assignment Questions

1. What is the KNN Algorithm?
2. KNN is parametric algorithm or non-parametric algorithm explained it why?
3. What is space and time complexity of the KNN Algorithm?
4. Why is KNN Algorithm known as Lazy Learner?
5. What are Support Vector Machines (SVMs)?
6. What is the basic principle of a Support Vector Machine?
7. What are hard margin and soft Margin SVMs?
8. What do you mean by Hinge loss in SVM?

Assignment no:5

Title of the Assignment: Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method.

Dataset Description:

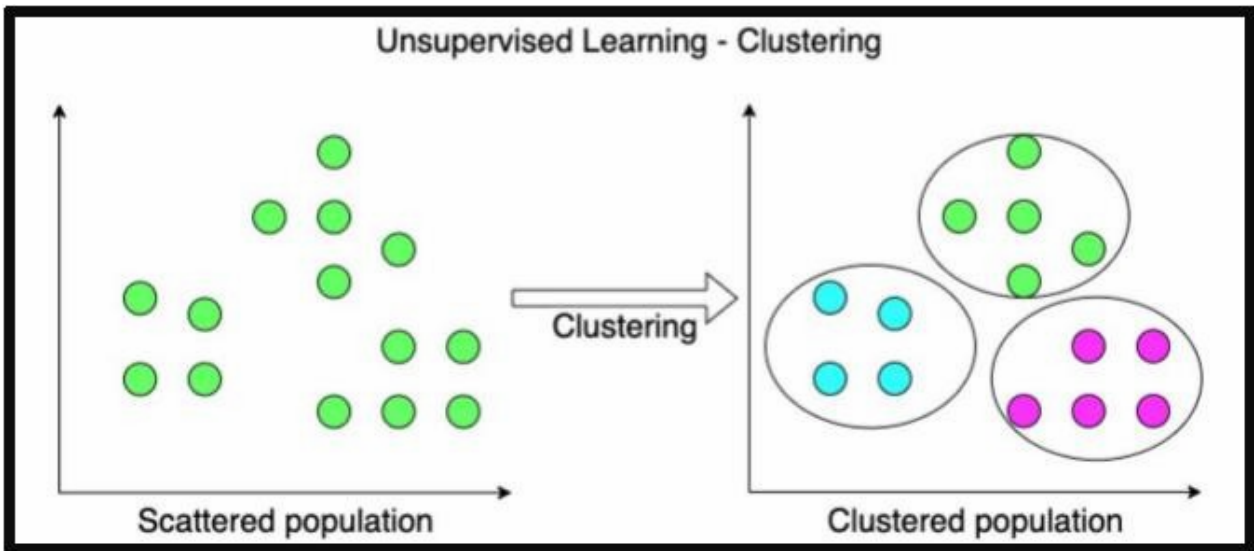
The data includes the following features: 1. Customer ID 2. Customer Gender 3. Customer Age 4. Annual Income of the customer (in Thousand Dollars) 5. Spending score of the customer (based on customer behavior and spending nature)

Objective of the Assignment:
Students should able to understand how to use unsupervised learning to segment different-different clusters or groups and used to them to train your model to predict future things.

Prerequisite:

1. Knowledge of Python
2. Unsupervised learning
3. Clustering
4. Elbow method

Clustering algorithms try to find natural clusters in data, the various aspects of how the algorithms to cluster data can be tuned and modified. Clustering is based on the principle that items within the same cluster must be similar to each other. The data is grouped in such a way that related elements are close to each other.



Diverse and different types of data are subdivided into smaller groups.

Uses of Clustering Marketing: In the field of marketing, clustering can be used to identify various customer groups with existing customer data. Based on that, customers can be provided with discounts, offers, promo codes etc.
Real Estate:

Clustering can be used to understand and divide various property locations based on value and importance. Clustering algorithms can process through the data and identify various groups of property on the basis of probable price.
Book Store and Library management: Libraries and Bookstores can use Clustering to better manage the book database. With proper book ordering, better operations can be implemented.
Document Analysis: Often, we need to group together various research texts and documents according to similarity. And in such cases, we don't have any labels. Manually labelling large amounts of data is also not possible. Using clustering, the algorithm can process the text and group it into different themes.

These are some of the interesting use cases of clustering.

K-Means Clustering K-Means clustering is an unsupervised machine learning algorithm that divides the given data into the given number of clusters. Here, the "K" is the given number of predefined clusters, that need to be created.

It is a centroid based algorithm in which each cluster is associated with a centroid. The main idea is to reduce the distance between the data points and their respective cluster centroid.

The algorithm takes raw unlabeled data as an input and divides the dataset into clusters and the process is repeated until the best clusters are found.

K-Means is very easy and simple to implement. It is highly scalable, can be applied to both small and

large datasets. There is, however, a problem with choosing the number of clusters or K. Also, with the increase in dimensions, stability decreases. But, overall K Means is a simple and robust algorithm that makes clustering very easy.

```
#Importing the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

The necessary libraries are imported.

```
#Reading the excel file
data=pd.read_excel("Mall_Customers.xlsx")
```

The data is read. I will share a link to the entire code and excel data at the end of the article. The

data has 200 entries, that is data from 200 customers.

```
data.head()
```

So let us have a look at the data.

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

data.corr()

	CustomerID	Age	Annual Income (k$)	Spending Score
CustomerID	1.000000	-0.026763	0.977548	0.013835
Age	-0.026763	1.000000	-0.012398	-0.327227
Annual Income (k$)	0.977548	-0.012398	1.000000	0.009903
Spending Score (1-100)	0.013835	-0.327227	0.009903	1.000000

The data seems to be interesting. Let us look at the data distribution.

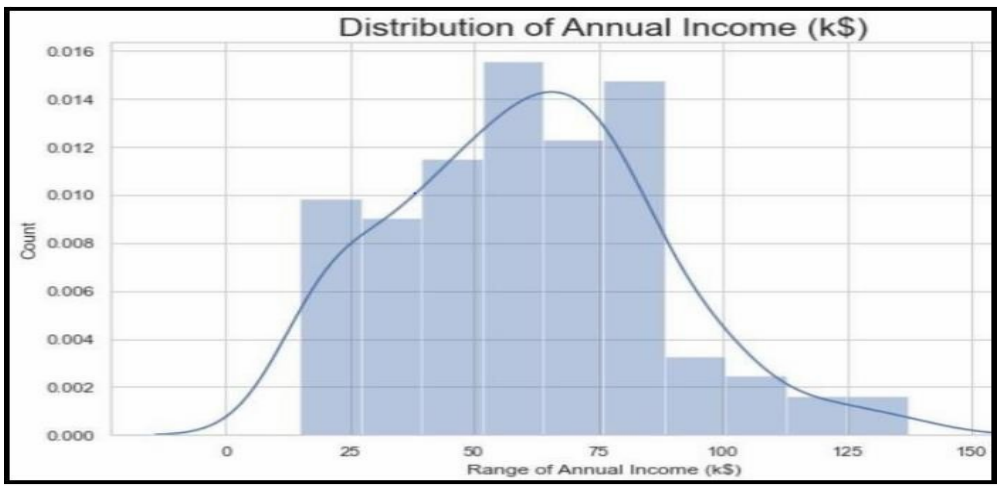
Annual Income Distribution:

#Distribution of Annnual Income

plt.figure(figsize=(10, 6))

sns.set(style = 'whitegrid')

sns.distplot(data['Annual Income (k$)']) plt.title('Distribution of Annual Income (k$)', fontsize = 20) plt.xlabel('Range of Annual Income (k$)') plt.ylabel('Count')



#Importing KMeans from sklearn

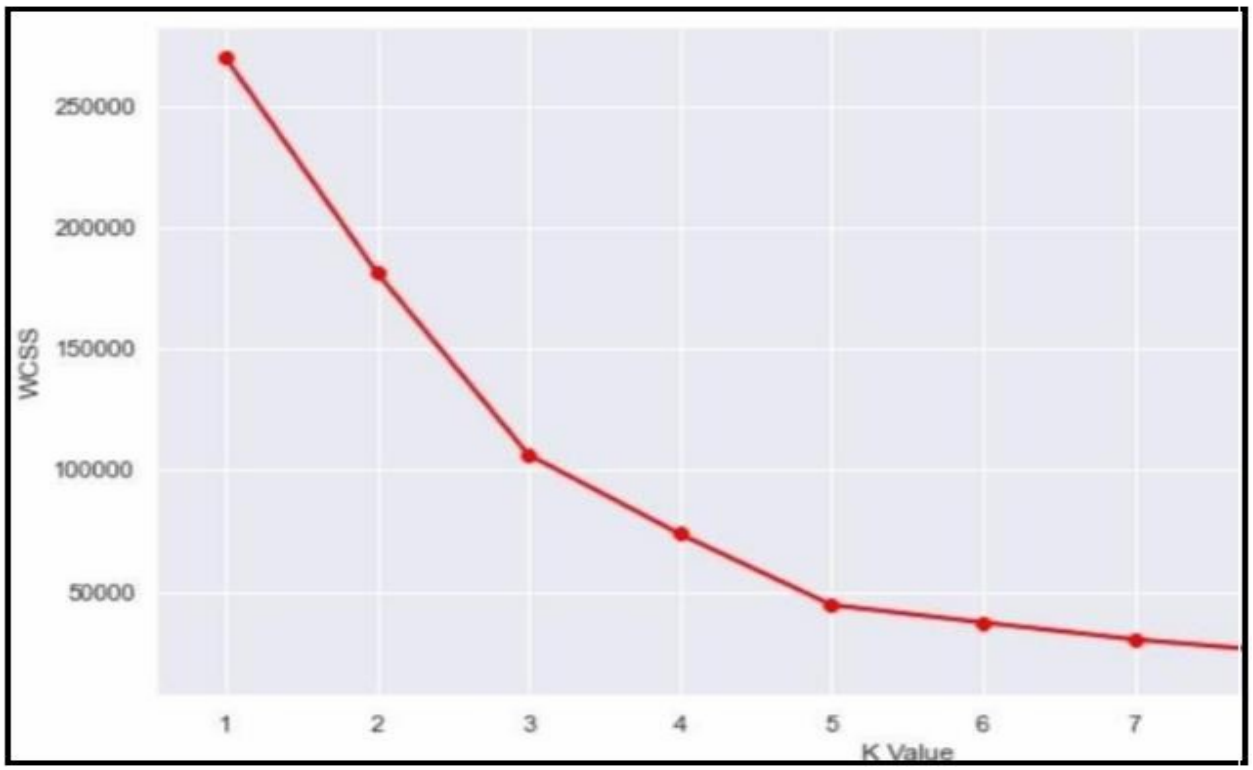
from sklearn.cluster import KMeans

Now we calculate the Within Cluster Sum of Squared Errors (WSS) for different values of k. Next, we choose the k for which WSS first starts to diminish. This value of K gives us the best number of clusters to make from the raw data.

wcss=[]

```
for i in range(1,11):
km=KMeans(n_clusters=i)
km.fit(X)
wcss.append(km.inertia_)
#The elbow curve
plt.figure(figsize=(12,6))
plt.plot(range(1,11),wcss)
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")
plt.xlabel("K Value")
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS") plt.show()
```

The plot:



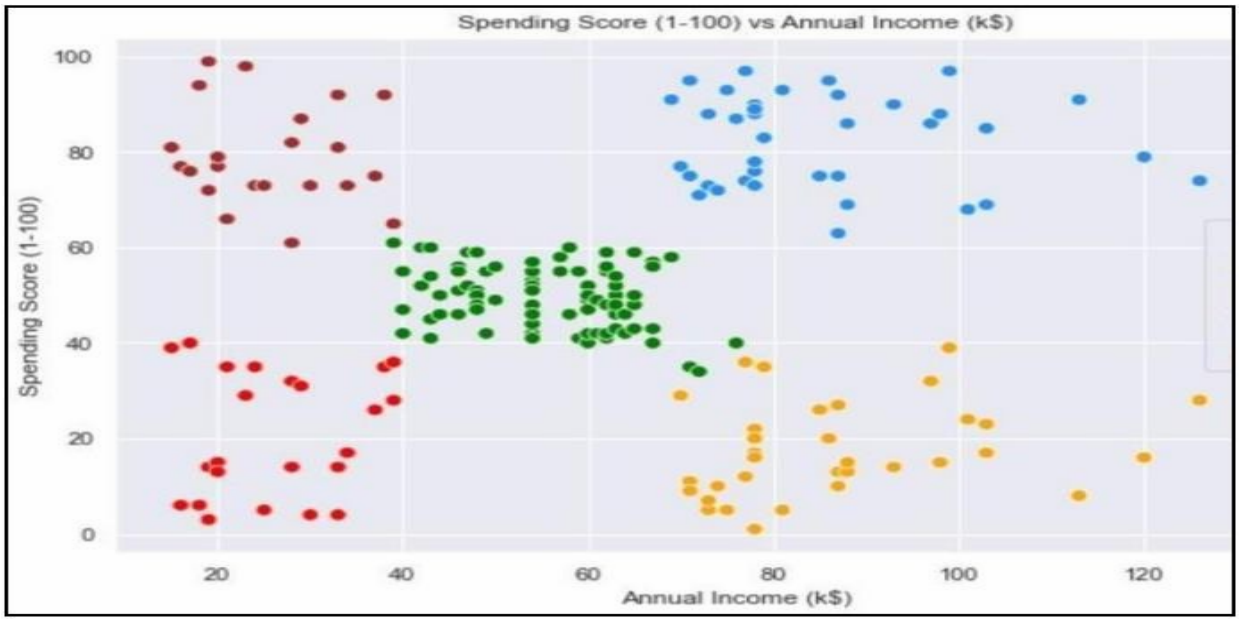
This is known as the elbow graph, the x-axis being the number of clusters, the number of clusters is taken at the elbow joint point. This point is the point where making clusters is most relevant as here the value of WCSS suddenly stops decreasing. Here in the graph, after 5 the drop is minimal, so we take 5 to be the number of clusters.

```
#Taking 5 clusters
km1=KMeans(n_clusters=5)
#Fitting the input data
km1.fit(X)
#predicting the labels of the input data
y=km1.predict(X)
#adding the labels to a column named label
df1["label"] = y
#The new dataframe with the clustering done
df1.head()
```

The labels added to the data.

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	label
0	1	Male	19	15	39	4
1	2	Male	21	15	81	2
2	3	Female	20	16	6	4
3	4	Female	23	16	77	2
4	5	Female	31	17	40	4

```
#Scatterplot of the clusters
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',hue="label",
palette=['green','orange','brown','dodgerblue','red'], legend='full',data = df1 ,s = 60 )
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending  Score  (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.show()
```



We can clearly see that 5 different clusters have been formed from the data. The red cluster is the customers with the least income and least spending score, similarly, the blue cluster is the customers with the most income and most spending score.

k-Means Clustering on the basis of 3D data Now, we shall be working on 3 types of data. Apart from the spending score and annual income of customers, we shall also take in the age of the customers.

#Taking the features

X2=df2[["Age","Annual Income (k$)","Spending Score (1-100)"]]

#Now we calculate the Within Cluster Sum of Squared Errors (WSS) for different values of k. wcss = []

for k in range(1,11):

kmeans = KMeans(n_clusters=k, init="k-means++")

kmeans.fit(X2)

wcss.append(kmeans.inertia_)

plt.figure(figsize=(12,6))

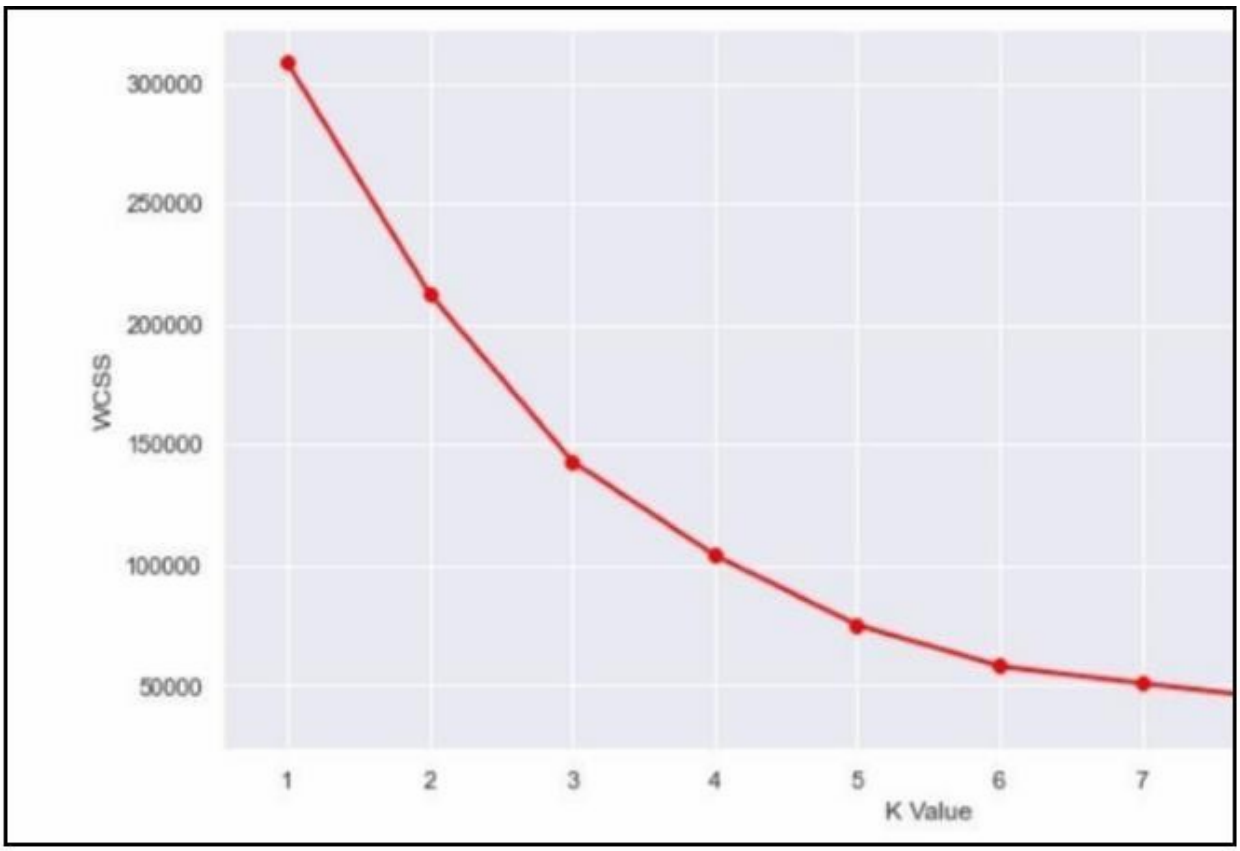
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")

plt.xlabel("K Value")

plt.xticks(np.arange(1,11,1))

plt.ylabel("WCSS") plt.show()

The WCSS curve.


300000
250000
200000
150000
100000
50000
WCSS
1
2
3
4
5
6
7
K Value

Group C

Assignment no:1

Title of the Assignment: Installation of MetaMask and study spending Ether per transaction

Objective of the Assignment:
Students should be able to learn new technology such as metamask.Its application and implementations

Prerequisite:

1. Basic knowledge of cryptocurrency
2. Basic knowledge of distributed computing concept
3. Working of blockchain

Introduction to Blockchain

- Blockchain can be described as a data structure that holds transactional records and while ensuring security, transparency, and decentralization. You can also think of it as a chain or records stored in the forms of blocks which are controlled by no single authority.

- A blockchain is a distributed ledger that is completely open to any and everyone on the network. Once an information is stored on a blockchain, it is extremely difficult to change or alter it.

- Each transaction on a blockchain is secured with a digital signature that proves its authenticity. Due to the use of encryption and digital signatures, the data stored on the blockchain is tamper-proof and cannot be changed.

- Blockchain technology allows all the network participants to reach an agreement, commonly known as consensus. All the data stored on a blockchain is recorded digitally and has a common history which is available for all the network participants. This way, the chances of any fraudulent activity or duplication of transactions is eliminated without the need of a third-party.

Blockchain Features

The following features make the revolutionary technology of blockchain stand out:

Decentralized

- Blockchains are decentralized in nature meaning that no single person or group holds the authority of the overall network. While everybody in the network has the copy of the distributed ledger with them, no one can modify it on his or her own. This unique feature of blockchain allows transparency and security while giving power to the users.

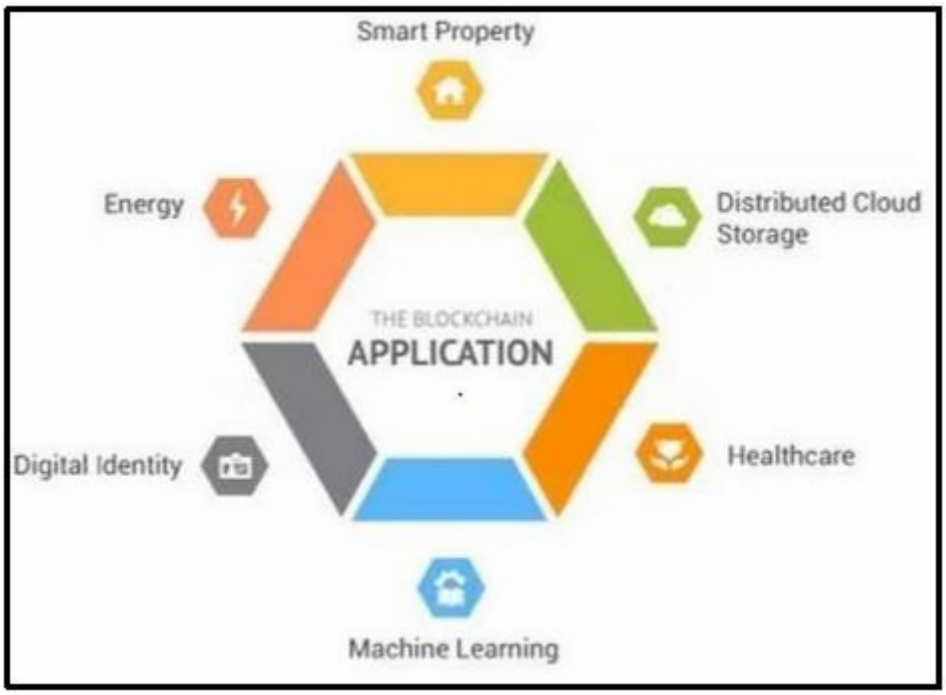
Immutable

- The immutability property of a blockchain refers to the fact that any data once written on the blockchain cannot be changed. To understand immutability, consider sending email as an example. Once you send an email to a bunch of people, you cannot take it back. In order to find a way around, you'll have to ask all the recipients to delete your email which is pretty tedious. This is how immutability works.

Tamper-Proof

- With the property of immutability embedded in blockchains, it becomes easier to detect tampering of any data. Blockchains are considered tamper-proof as any change in even one single block can be detected and addressed smoothly. There are two key ways of detecting tampering namely, hashes and blocks.

Popular Applications of Blockchain Technology



Benefits of Blockchain Technology:

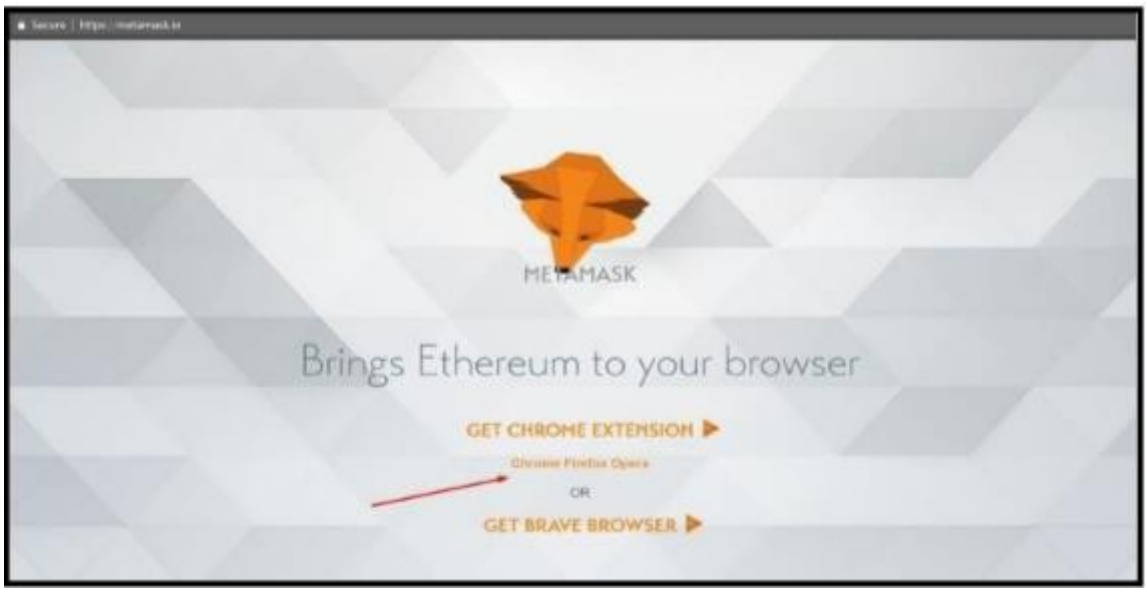
- Time-saving: No central Authority verification needed for settlements making the process faster and cheaper.
- Cost-saving: A Blockchain network reduces expenses in several ways. No need for third-party verification. Participants can share assets directly. Intermediaries are reduced. Transaction efforts are minimized as every participant has a copy of shared ledger.
- Tighter security: No one can temper with Blockchain Data as it is shared among millions of participants. The system is safe against cybercrimes and Fraud.
- In finance market trading, Fibonacci retracement levels are widely used in technical analysis.

How to use MetaMask: A step by step guide

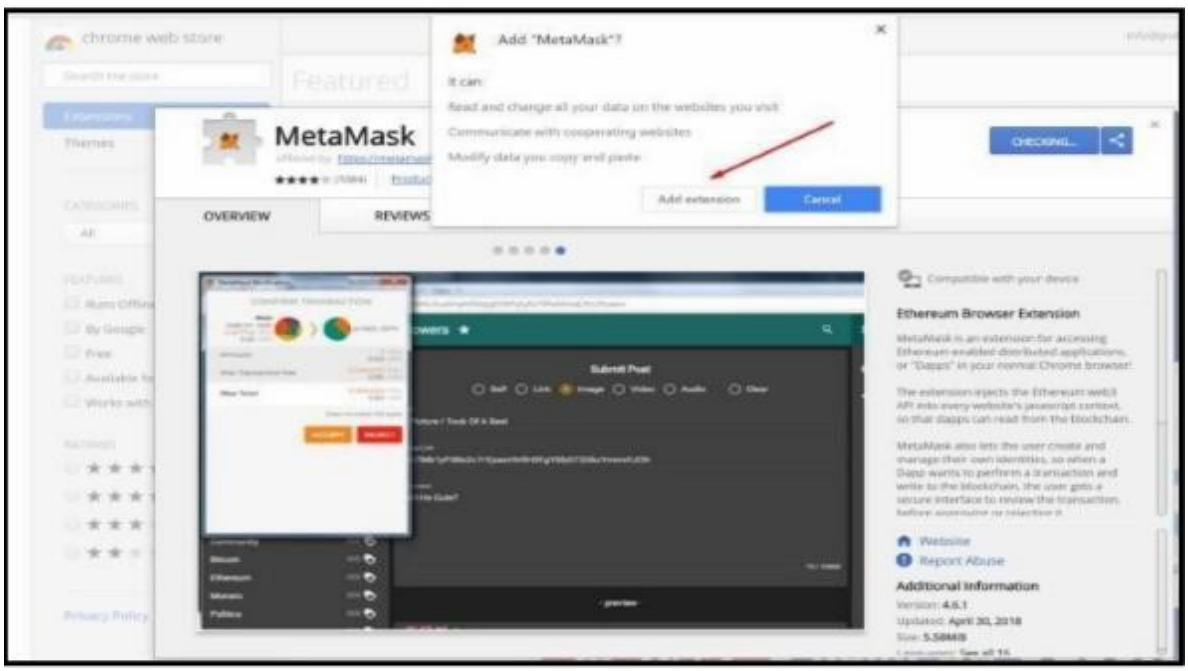
MetaMask is one of the most popular browser extensions that serves as a way of storing your Ethereum and other ERC-20 Tokens. The extension is free and secure, allowing web applications to read and interact with Ethereum's blockchain.

Step 1. Install MetaMask on your browser.

To create a new wallet, you have to install the extension first. Depending on your browser, there are different marketplaces to find it. Most browsers have MetaMask on their stores, so it's not that hard to see it, but either way, here they are Chrome, Firefox, and Opera.



- **Click on Install Meta Mask as a Google Chrome extension.**
- **Click Add to Chrome.**
- **Click Add Extension.**



It's as easy as that to install the extension on your browser, continue reading the next step to figure out how to create an account.

Step 2. Create an account.

- Click on the extension icon in the upper right corner to open MetaMask.
- To install the latest version and be up to date, click Try it now.
- Click Continue.
- You will be prompted to create a new password. Click Create.

Proceed by clicking Next and accept the Terms of Use.

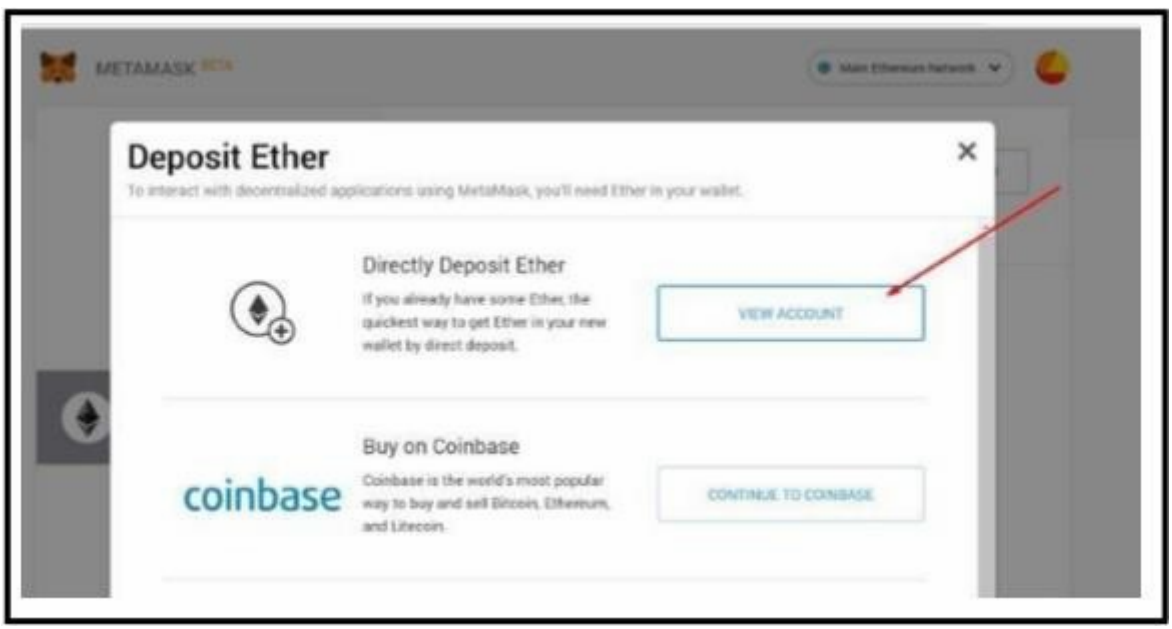
Click Reveal Secret Words. There you will see a 12 words seed phrase. This is really important and usually not a good idea to store digitally, so take your time and write it down.

- Verify your secret phrase by selecting the previously generated phrase in order. Click Confirm.

And that's it; now you have created your MetaMask account successfully. A new Ethereum wallet address has just been created for you. It's waiting for you to deposit funds, and if you want to learn how to do that, look at the next step below.

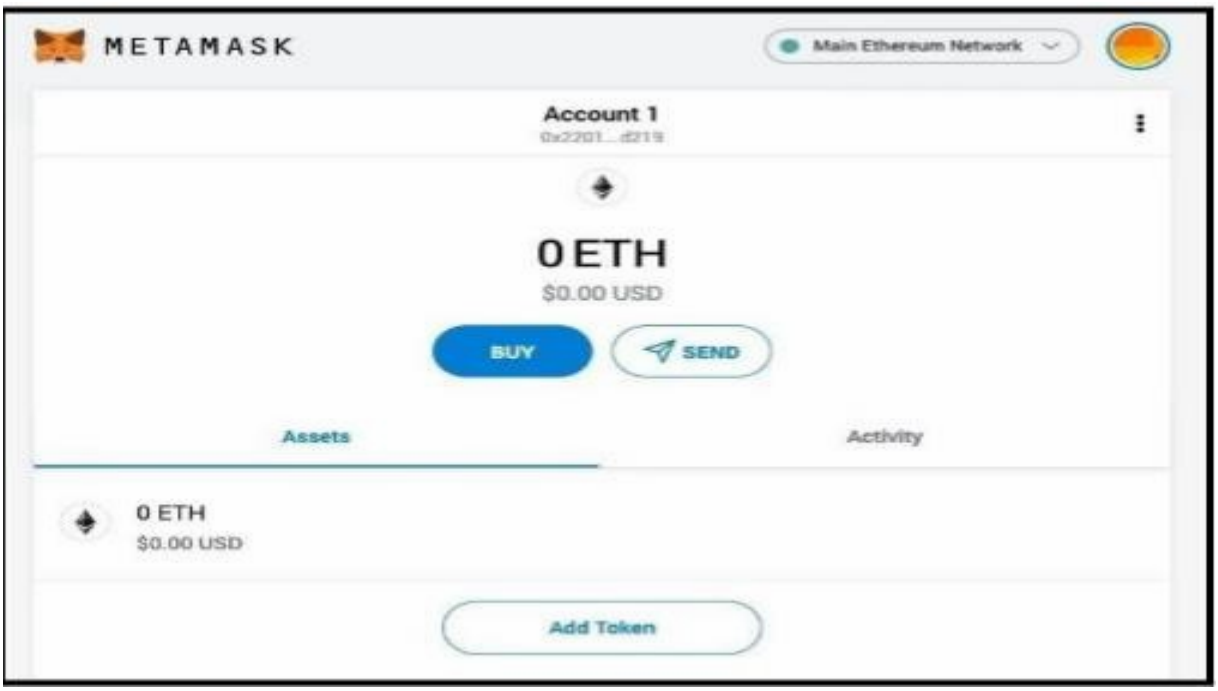
Step 3. Depositing funds.

- Click on View Account.



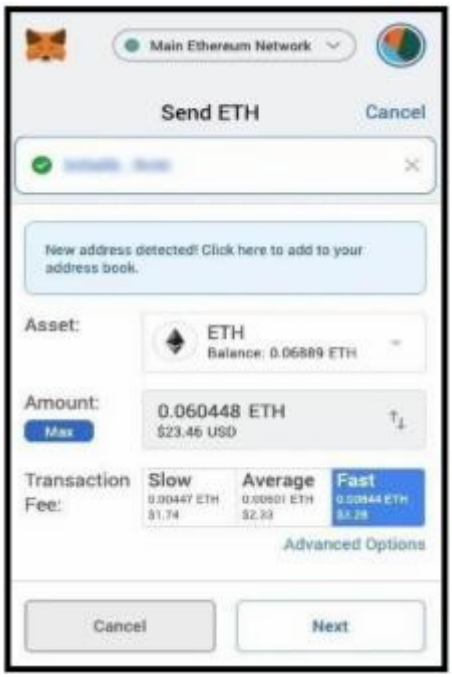
You can now see your public address and share it with other people. There are some methods to buy coins offered by Meta Mask, but you can do it differently as well; you just need your address.

If you ever get logged out, you'll be able to log back in again by clicking the MetaMask icon, which will have been added to your web browser (usually found next to the URL bar).



You can now access your list of assets in the 'Assets' tab and view your transaction history in the 'Activity' tab.

- Sending crypto is as simple as clicking the 'Send' button, entering the recipient address and amount to send, and selecting a transaction fee. You can also manually adjust the transaction fee using the 'Advanced Options' button, using information from ETH Gas Station or similar platforms to choose a more acceptable gas price.
- After clicking 'Next', you will then be able to either confirm or reject the transaction on the subsequent page.



- To use MetaMask to interact with a dapp or smart contract, you'll usually need to find a 'Connect to Wallet' button or similar element on the platform you are trying to use. After clicking this, you should then see a prompt asking whether you want to let the dapp connect to your wallet.

What advantages does MetaMask have?

- Popular - It is commonly used, so users only need one plugin to access a wide range of dapps.
- Simple - Instead of managing private keys, users just need to remember a list of words, and transactions are signed on their behalf.
- Saves space - Users don't have to download the Ethereum blockchain, as MetaMask sends requests to nodes outside of the user's computer.
- Integrated - Dapps are designed to work with MetaMask, so it becomes much easier to send Ether in and out.

Conclusion- In this way we have explored Concept Block chain and metamask wallet for transaction of digital currency

Assignment Question

1. What Are the Different Types of Blockchain Technology?
2. What Are the Key Features/Properties of Blockchain?
3. What Type of Records You Can Keep in A Blockchain?
4. What is the difference between Ethereum and Bitcoin?

5. What are Merkle Trees? Explain their concept.
6. What is Double Spending in transaction operation
7. Give real-life use cases of blockchain.

Assignment no:2

Title of the Assignment: Create your own wallet using Metamask for crypto transactions Objective of the Assignment:

Students should be able to learn about crypto currencies and learn how transaction done by using different digital currency.

Prerequisite:

1. Basic knowledge of crypto currency
2. Basic knowledge of distributed computing concept
3. Working of blockchain

Contents for Theory:

1. Crypto currency
2. Transaction Wallets
3. Ether transaction

Introduction to Crypto currency

- Crypto currency is a digital payment system that doesn't rely on banks to verify transactions. It's a peer-to-peer system that can enable anyone anywhere to send and receive payments. Instead of being physical money carried around and exchanged in the real world, crypto currency payments exist purely as digital entries to an online database describing specific transactions. When you transfer crypto currency funds, the transactions are recorded in a public ledger. Crypto currency is stored in digital wallets.

- Crypto currency received its name because it uses encryption to verify transactions. This means advanced coding is involved in storing and transmitting crypto currency data between wallets and to public ledgers. The aim of encryption is to provide security and safety.

- The first crypto currency was Bit coin, which was founded in 2009 and remains the best known today. Much of the interest in crypto currencies is to trade for profit, with speculators at times driving prices skyward.

How does crypto currency work?

- Crypto currencies run on a distributed public ledger called blockchain, a record of all transactions updated and held by currency holders.
- Units of crypto currency are created through a process called mining, which involves using computer power to solve complicated mathematical problems that generate coins. Users can also buy the currencies from brokers, then store and spend them using cryptographic wallets.
- If you own crypto currency, you don't own anything tangible. What you own is a key that allows you to move a record or a unit of measure from one person to another without a trusted third party.
- Although Bit coin has been around since 2009, crypto currencies and applications of blockchain technology are still emerging in financial terms, and more uses are expected in the future. Transactions including bonds, stocks, and other financial assets could eventually be

traded using the technology.

Cryptocurrency examples

There are thousands of cryptocurrencies. Some of the best known include:

- **Bitcoin:**

  Founded in 2009, Bitcoin was the first cryptocurrency and is still the most commonly traded. The currency was developed by Satoshi Nakamoto – widely believed to be a pseudonym for an individual or group of people whose precise identity remains unknown.

- **Ethereum:**

  Developed in 2015, Ethereum is a blockchain platform with its own cryptocurrency, called Ether (ETH) or Ethereum. It is the most popular cryptocurrency after Bitcoin.

- **Litecoin:**

  This currency is most similar to bitcoin but has moved more quickly to develop new innovations, including faster payments and processes to allow more transactions.

- **Ripple:**

  Ripple is a distributed ledger system that was founded in 2012. Ripple can be used to track different kinds of transactions, not just cryptocurrency. The company behind it has worked with various banks and financial institutions.

- Non-Bitcoin cryptocurrencies are collectively known as “altcoins” to distinguish them from the original.

How to store crypto currency

Once you have purchased crypto currency, you need to store it safely to protect it from hacks or theft. Usually, crypto currency is stored in crypto wallets, which are physical devices or online software used to store the private keys to your crypto currencies securely. Some exchanges provide wallet services, making it easy for you to store directly through the platform. However, not all exchanges or brokers automatically provide wallet services for you.

There are different wallet providers to choose from. The terms “hot wallet” and “cold wallet” are used:

- Hot wallet storage: "hot wallets" refer to crypto storage that uses online software to protect the private keys to your assets.
- Cold wallet storage: Unlike hot wallets, cold wallets (also known as hardware wallets) rely on offline electronic devices to securely store your private keys.

Conclusion- In this way we have explored Concept Crypto currency and learn how transactions are done using digital currency

Assignment Question

1. What is Bit coin?
2. What are the biggest four common crypto currency scams?
3. Explain How safe are money e-transfers?
4. What is crypto jacking and how does it work?

Assignment no:3

Title of the Assignment:

Write a smart contract on a test network, for Bank account of a customer for following operations:

- **Deposit money**
- **Withdraw Money**

• Show balance

Objective of the Assignment:

Students should be able to learn new technology such as metamask.Its application and implementations.

Prerequisite:

1. Basic knowledge of crypto currency
2. Basic knowledge of distributed computing concept
3. Working of blockchain.

Contents for Theory:

The contract will allow deposits from any account, and can be trusted to allow withdrawals only by accounts that have sufficient funds to cover the requested withdrawal. This post assumes that you are comfortable with the ether-handling concepts introduced in our post, Writing a Contract That Handles Ether.

That post demonstrated how to restrict ether withdrawals to an "owner's" account. It did this by persistently storing the owner account's address, and then comparing it to the msg.sender value for any withdrawal attempt. Here's a slightly simplified version of that smart contract, which allows anybody to deposit money, but only allows the owner to make withdrawals: pragma solidity ^0.4.19;

```
contract TipJar {

address owner; // current owner of the contract function

TipJar() public { owner = msg.sender; }

function withdraw() public { require(owner == msg.sender); msg.sender.transfer(address(this).balance); }

function deposit(uint256 amount) public payable { require(msg.value == amount);
}

function getBalance() public view returns (uint256) { return address(this).balance;
}
}
```

I am going to generalize this contract to keep track of ether deposits based on the account address of the depositor, and then only allow that same account to make withdrawals of that ether. To do this, we need a way keep track of account balances for each depositing account—a mapping from accounts to

balances. Fortunately, Solidity provides a ready-made mapping data type that can map account addresses to integers, which will make this bookkeeping job quite simple. (This mapping structure is much more general key/value mapping than just addresses to integers, but that's all we need here.) Here's the code to accept deposits and track account balances:

```
pragma solidity ^0.4.19;
contract Bank {
mapping(address => uint256) public balanceOf; // balances, indexed by addresses

function deposit(uint256 amount) public payable { require(msg.value == amount); balanceOf[msg.sender]

+= amount; // adjust the account's balance
}
}
```

Here are the new concepts in the code above:

- mapping(address => uint256) public balanceOf; declares a persistent public variable, balanceOf, that is a mapping from account addresses to 256-bit unsigned integers. Those integers will represent the current balance of ether stored by the contract on behalf of the corresponding address.
- Mappings can be indexed just like arrays/lists/dictionaries/tables in most modern programming languages.
- The value of a missing mapping value is 0. Therefore, we can trust that the beginning balance for all account addresses will effectively be zero prior to the first deposit.

It's important to note that balance Of keeps track of the ether balances assigned to each account, but it does not actually move any ether anywhere. The bank contract's ether balance is the sum of all the balances of all accounts—only balances Of tracks how much of that is assigned to each account. Note also that this contract doesn't need a constructor. There is no persistent state to initialize other than the balance of mapping, which already provides default values of 0. Given the balance Of mapping from account addresses to ether amounts, the remaining code for a fully- functional bank contract is pretty small. I'll simply add a withdrawal function:

bank.sol

```
pragma solidity ^0.4.19;

contract Bank {

   mapping(address => uint256) public balanceOf;  // balances, indexed by addresses

   function deposit(uint256 amount) public payable {
      require(msg.value == amount);
      balanceOf[msg.sender] += amount;     // adjust the account's balance
   }

   function withdraw(uint256 amount) public {
      require(amount <= balanceOf[msg.sender]);
      balanceOf[msg.sender] -= amount;
      msg.sender.transfer(amount);
   }
}
```

The code above demonstrates the following:

- The require(amount <= balances[msg.sender]) checks to make sure the sender has sufficient funds to cover the requested withdrawal. If not, then the transaction aborts without making any state changes or ether transfers.

- The balanceOf mapping must be updated to reflect the lowered residual amount after the withdrawal.

- The funds must be sent to the sender requesting the withdrawal. In the withdraw() function above, it is very important to adjust balanceOf[msg.sender] before transferring ether to avoid an exploitable vulnerability. The reason is specific to smart contracts and the fact that a transfer to a smart contract executes code in that smart contract. (The essentials of Ethereum transactions are discussed in How Ethereum Transactions Work.)

Now, suppose that the code in withdraw () did not adjust balance Of[msg.sender] before making the transfer and suppose that msg.sender was a malicious smart contract. Upon receiving the transfer—handled by msg.sender's fallback function—that malicious contract could initiate another withdrawal from the banking contract. When the banking contract handles this second withdrawal request, it would have already transferred ether for the original withdrawal, but it would not have an updated balance, so it would allow this second withdrawal!

This vulnerability is called a "reentrancy" bug because it happens when a smart contract invokes code in a different smart contract that then calls back into the original, thereby reentering the exploitable contract. For this reason, it's essential to always make sure a contract's internal state is fully updated before it potentially invokes code in another smart contract. (And, it's essential to remember that every transfer to a smart contract executes that contract's code.)

To avoid this sort of reentrancy bug, follow the "Checks-Effects-Interactions pattern" as described in the Solidity documentation. The withdraw() function above is an example of implementing this pattern

Assignment no:4

Title of the Assignment:

Write a survey report on types of Blockchains and its real time use cases.

Objective of the Assignment:

Students should be able to learn new technology such as metamask.Its application and implementations

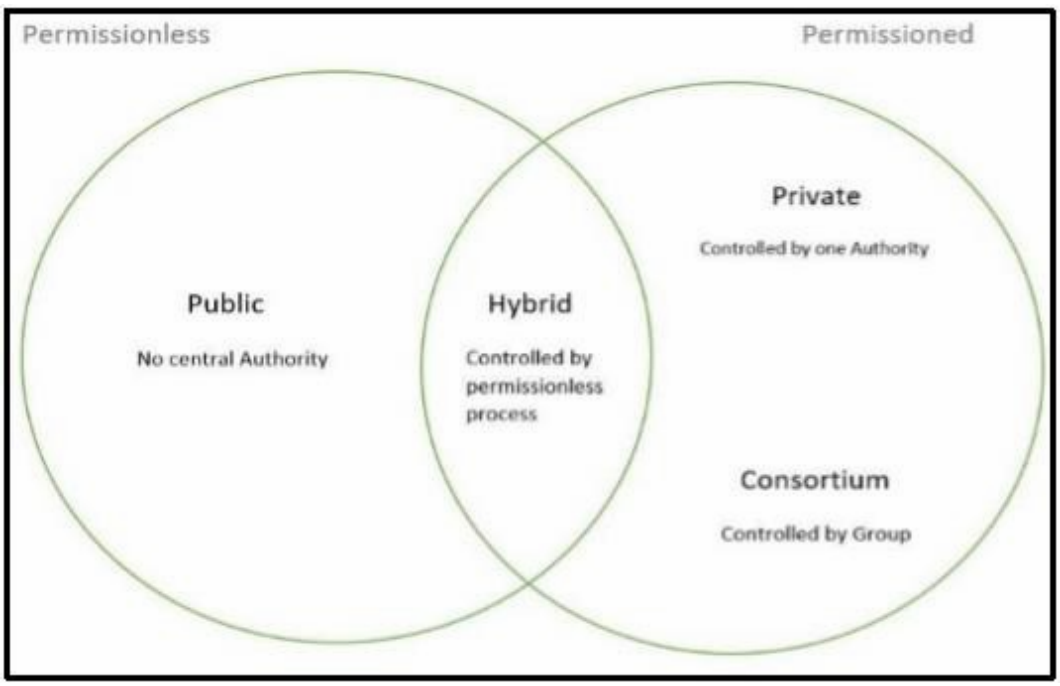
Prerequisite:

1. Basic knowledge of cryptocurrency
2. Basic knowledge of distributed computing concept
3. Working of blockchain

Contents for Theory:

There are 4 types of blockchain:

1. Public Blockchain.
2. Private Blockchain.
3. Hybrid Blockchain.
4. Consortium Blockchain



1. **Public Blockchain**

- These blockchains are completely open to following the idea of decentralization. They don't have any restrictions, anyone having a computer and internet can participate in the network.

- As the name is public this blockchain is open to the public, which means it is not owned by anyone. Anyone having internet and a computer with good hardware can participate in this public blockchain. All the computer in the network hold the copy of other nodes or block present in the network In this public blockchain, we can also perform verification of transactions or records.

Advantages

- Trustable: There are algorithms to detect no fraud. Participants need not worry about the other nodes in the network

- Secure: This blockchain is large in size as it is open to the public. In a large size, there is greater distribution of records

- Anonymous Nature: It is a secure platform to make your transaction properly at the same time, you are not required to reveal your name and identity in order to participate.

- Decentralized: There is no single platform that maintains the network, instead every user has a copy of the ledger.

Disadvantages

- Processing: The rate of the transaction process is very slow, due to its large size. Verification of each node is a very time-consuming process.

- Energy Consumption: Proof of work is high energy-consuming. It requires good computer hardware to participate in the network Acceptance: No central authority is there so governments are facing the issue to implement the technology faster.

- Use Cases: Public Blockchain is secured with proof of work or proof of stake they can be used to displace traditional financial systems. The more advanced side of this blockchain is the smart contract that enabled this blockchain to support decentralization. Examples of public blockchain are Bitcoin, Ethereum.

2. Private Blockchain

- These blockchains are not as decentralized as the public blockchain only selected nodes can participate in the process, making it more secure than the others.
- These are not as open as a public blockchain.
- They are open to some authorized users only.
- These blockchains are operated in a closed network. In this few people are allowed to participate in a network within a company/organization.

Advantages:

- Speed: The rate of the transaction is high, due to its small size. Verification of each node is less time- consuming.
- Scalability: We can modify the scalability. The size of the network can be decided manually.
- Privacy: It has increased the level of privacy for confidentiality reasons as the businesses required.
- Balanced: It is more balanced as only some user has the access to the transaction which improves the performance of the network.

Disadvantages:

- Security- The number of nodes in this type is limited so chances of manipulation are there. These blockchains are more vulnerable.
- Centralized- Trust building is one of the main disadvantages due to its central nature. Organizations can use this for malpractices.
- Count- Since there are few nodes if nodes go offline the entire system of blockchain can be endangered. Use Cases: With proper security and maintenance, this blockchain is a great asset to secure information without exposing it to the public eye. Therefore companies use them for internal auditing, voting, and asset management. An example of private blockchains is Hyperledger, Corda.

3. **Hybrid Blockchain**

- It is the mixed content of the private and public blockchain, where some part is controlled by some organization and other makes are made visible as a public blockchain.
- It is a combination of both public and private blockchain.
- Permission-based and permission less systems are used. User access information via smart contracts Even a primary entity owns a hybrid blockchain it cannot alter the transaction.

Advantages:

- Ecosystem: Most advantageous thing about this blockchain is its hybrid nature. It cannot be hacked as 51% of users don't have access to the network
- Cost: Transactions are cheap as only a few nodes verify the transaction. All the nodes don't carry the verification hence less computational cost.
- Architecture: It is highly customizable and still maintains integrity, security, and transparency.
- Operations: It can choose the participants in the blockchain and decide which transaction can be made public.

Disadvantages:

- Efficiency: Not everyone is in the position to implement a hybrid Blockchain. The organization also faces some difficulty in terms of efficiency in maintenance.
- Transparency: There is a possibility that someone can hide information from the user. If someone wants to get access through a hybrid blockchain it depends on the organization whether they will give or not.
- Ecosystem: Due to its closed ecosystem this blockchain lacks the incentives for network participation.
- Use Case: It provides a greater solution to the health care industry, government, real estate, and financial companies. It provides a remedy where data is to be accessed publicly but needs to be shielded privately. Examples of Hybrid Blockchain are Ripple network and XRP token.

4. Consortium Blockchain

- It is a creative approach that solves the needs of the organization.
- This blockchain validates the transaction and also initiates or receives transactions. Also known as Federated Blockchain.
- This is an innovative method to solve the organization's needs. Some part is public and some part is private. In this type, more than one organization manages the blockchain.

Advantages:

- Speed: A limited number of users make verification fast. The high speed makes this more usable for organizations. Authority: Multiple organizations can take part and make it decentralized at every level. Decentralized authority, makes it more secure.
- Privacy: The information of the checked blocks is unknown to the public view. but any member belonging to the blockchain can access it.
- Flexible: There is much divergence in the flexibility of the blockchain. Since it is not a very large decision can be taken faster.

Disadvantages:

- Approval: All the members approve the protocol making it less flexible. Since one or more organizations are involved there can be differences in the vision of interest.
- Transparency: It can be hacked if the organization becomes corrupt. Organizations may hide information from the users.
- Vulnerability: If few nodes are getting compromised there is a greater chance of vulnerability in this blockchain
- Use Cases: It has high potential in businesses, banks, and other payment processors. Food tracking of the organizations frequently collaborates with their sectors making it a federated solution ideal for their use. Examples of consortium Blockchain are Tender mint and Multichannel.

Conclusion-In this way we have explored types of blockchain and its applications in real time.