SHREE CHANAKYA EDUCATION SOCIETY'S

INDIRA COLLEGE OF ENGINEERING AND MANAGEMENT

Approved By AICTE New Delhi, DTE (MS) and Affiliated to Pune University (Id-No. PU/PN/Engg/282/2007)

Department of Computer Engineering

410246: Laboratory Practice-III Lab
(B. E. Computer) 2019 Course

LABORATORY MANUAL
(w. e. f. July 2024)

Subject Teacher: - Prof. Minal Pravin Jungare

INDIRA COLLEGE OF ENGINEERING AND MANAGEMENT,PARANDWADI		
Department of Computer Engineering		
Class: B.E	**Laboratory Practice III**	**Semester**: VII
Practical Plan		
Practical hours per week: 04		
Notes : 1. Set-A (DAA) &Set-B (ML) & Set-C (Blockchain) assignments are covered with one mini project each. 2.Operating System recommended :- 64-bit Open source Linux or its derivative 3.Programming tools recommended: - Open Source Programming tool like G++/GCC , /Java/Python/Sharp/Apex		

Assig. No.	Planned Date	Broad Topics to be Covered	Name of Experiment	Actual Date of Conduction
SET-A (Design & Analysis of Algorithm)				
1		Recursion and Non-recursive functions	Write a program non-recursive and recursive program to calculate Fibonacci numbers and analyze their time and space complexity.	
2		Greedy Strategy	Write a program to implement Huffman Encoding using a greedy strategy.	
3		Knapsack Problem	Write a program to solve a fractional Knapsack problem using a greedy method.	
4		dynamic programming or branch and bound strategy.	Write a program to solve a 0-1 Knapsack problem using dynamic programming or branch and bound strategy.	
5		Backtracking technique	Design n-Queens matrix having first Queen placed. Use backtracking to place remaining Queens to generate the final n-queen‘s matrix.	

6		Multithreading and single thread comparison	**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.	
7		linear regression and random forest regression models.	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 link: https://www.kaggle.com/datasets/yasserh/uber-fares-dataset	
8		K-Nearest Neighbors and Support Vector Machine for classification.	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 link: The emails.csv dataset on the Kaggle https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv	
9		Gradient Descent Algorithm	Implement Gradient Descent Algorithm to find the local minima of a function. For example, find the local minima of the function $y=(x+3)^2$ starting from the point x=2.	
10		confusion matrix, accuracy, error rate, precision and recall	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	
11		K-Means clustering/ hierarchical clustering	Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method. Dataset link: https://www.kaggle.com/datasets/kyanyoga/sample-sales-data	

12		Prediction on future stock price	**Mini Project -** Use the following dataset to analyze ups and downs in the market and predict future stock price returns based on Indian Market data from 2000 to 2020. Dataset Link: https://www.kaggle.com/datasets/sagara9595/stock-data	
13		MetaMask and Ehter	Installation of MetaMask and study spending Ether per transaction.	
14		Wallet and Crypto transactions	Create your own wallet using Metamask for crypto transactions.	
15		smart contract	Write a smart contract on a test network, for Bank account of a customer for following operations: □ Deposit money □ Withdraw Money □ Show balance	
16		Solidity & Ethereum	Write a program in solidity to create Student data. Use the following constructs: □ Structures □ Arrays □ Fallback Deploy this as smart contract on Ethereum and Observe the transaction fee and Gas values.	
17		Case Study of Blockchains	Write a survey report on types of Blockchains and its real time use cases.	
18		Blockchain & de-centralized app	**Mini Project -** Develop a Blockchain based application dApp (de-centralized app) for e-voting system.	

Experiment No: 1

Title: Write a program non-recursive and recursive program to calculate Fibonacci numbers and analyze their time and space complexity.

Aim:

Implement a program for non-recursive and recursive version of calculating Fibonacci number.

Prerequisites:

- Concept of programming and Data Structure.

Objectives:

- Student should be able to implement program non-recursive and recursive program to calculate Fibonacci numbers.

Theory:

The Fibonacci Sequence is the series of numbers:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
The next number is found by adding up the two numbers before it:

- the 2 is found by adding the two numbers before it (1+1),
- the 3 is found by adding the two numbers before it (1+2),
- the 5 is (2+3),
- and so on!

The Fibonacci Sequence can be written as a "Rule".
First, the terms are numbered from 0 onwards like this:

n =	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	...
x_n =	0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	...

So term number 6 is called x_6 (which equals 8).

The general mathematical formulation of Fibonacci series is:

$$F(n) = \begin{cases} 0 & , n = 0 \\ 1 & , n = 1 \\ F(n-1) + F(n-2) & , n > 1 \end{cases}$$

Algorithm for Iterative Fibonacci Series:

The iterative approach is the dynamic programming approach. It makes use of a loop to perform the addition of the previous two terms. The algorithm for the iterative approach will be:

Procedure Iterative_Fibonacci(n):
```
   int f0, f1, fib
   f0 = 0
   f1 = 1
   display f0, f1
   for int i := 1 to n:
      fib := f0 + f1
      f0 := f1
      f1 := fib
      display fib
   END for loop
END Iterative_Fibonacci
```

Code:
```
void fibonacciIterative(){
                int F,L,V;
                F=0;
                L=1;
                cout<<"\n";
                if(n<=2)
                        cout<<F<<" "<<L;
                else
                {
                        cout<<F<<" "<<L<<" ";
                        for (int i=3;i<n;i++)
                        {
                                V=F+L;
                                cout<<V<<" ";
                                F=L;
                                L=V;
                        }
                }
        }
```

Analysis of Algorithm:
In an algorithm and code, a for loop is used to print first ‘n’ Fibonacci numbers and the loop iterates ‘n’ number of times. Moreover, the remaining statements are executed exactly once and so the overall time complexity of the algorithm is O(n).

Algorithm for Recursive Fibonacci Series:

We can also use recursion to generate the Fibonacci series.

$$\begin{cases} T(0)=1, \\ T(1)=1, \\ T(n)=T(n-1)+T(n-2)+4 \end{cases}$$

$T(n)$ represents the fib function.
$T(n-1)$ and T(n-2) represent the recursive calls to fib(n-1) fib(n-2),
whereas the '4' represents the basic operations in the code.
Lastly, T(1) = 1 and T(0) = 1represent the base-cases.

The pseudo-code for the recursive method is:

```
Procedure Recursive_Fibonacci(n)
   if(n = 0 or n=1) then
      return n
   return Recursive_Fibonacci(n-1) + Recursive_Fibonacci(n-2)
END Recursive_Fibonacci
```

Code:

```
int fiboRec(int n)
                {
                        if((n==1)||(n==0))
                                return(n);
                        else
                                return(fiboRec(n-1)+fiboRec(n-2));
                }
```

Analysis of Algorithm:
T(n) = T(n-1) + T(n-2) + c

To solve this recurrence, we can simplify it by approximating that $T(n-1) \approx T(n-2)$.
The time that is taken to calculate T(n-1)$T(n-1)$ is *greater* than the time taken to calculate T(n-2)$T(n-2)$, so this approximation gives us an *upper bound* on the total time complexity:
T(n) = 2T(n-2)+c
T(n) = 2(2T(n-2)+c)+c
= 4T(n-4)+3c
= 4(2T(n-3)+c)+3c
= 8T(n-6)+7c
.
.
.
$T(n) = 2^kT(n-2k)+(2^k-1)c$----------(1)

where k is the number of times that the equation was substituted into.
T(n) = 2T(n-1)+c when $k=1$

$T(n) = 4T(n-2)+3c$ when $k=2$

$T(n) = 8T(n-3)+7c$ when $ck=3$

…

At terminating condition,

$n-2k = 0$

equating it to n we get,

$k=n/2$ ----------(2)

placing value of k in equation (1) we get,

$T(n) = 2^{n/2} T(0)+(2^{n/2} -1)c$

But $T(0) = 1$

So the above equation will be rewritten as,

$T(n) = 2^{n/2} +(2^{n/2} -1)c$

$= 2^{n/2} +c2^{n/2} -c$

$= (1+c) 2^{n/2}-c$

Considering asymptotic notion, the time required will be $O(2^{n/2})$

This means that the fib(n) function takes *exponential* time, which isn't very efficient.

Facilities: Fedora Operating Systems, C/C++/Java/Python

Application:

The algorithmis used in Stock Market price prediction and other similar applications.

Input:

The value of 'n' for which the series is to be generated.

Output:

The output is theFibonacci series up to 'n'.

Conclusion:

The implementation of Fibonacci series algorithm is done using recursion and non-recursive version. The analysis of algorithm is done using mathematical procedure.

Questions:

1. What is Fibonacci series?
2. What is algorithmic analysis of algorithm?
3. What is big O, Ω and Θ notations of algorithmic analysis?
4. What are the applications of Fibonacci series?
5. Can you apply the code optimization on both the versions of the algorithms? Justify the answer.

Experiment No: A2

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

Aim: Implementation of Huffman Encoding using a greedy strategy.

Prerequisites:

Concept of binary tree and greedy strategy.

Objectives:

Student should be able to implement Huffman Encoding using a greedy strategy.

Theory:

Weighted Binary Tree: (HUFFMAN TREE)

Consider the binary tree with 'n' internal nodes and we are given a set of n+1 positive weights as $q_1, q_2, \ldots q_{n+1}$. Now exactly one of these weights can be attached to each of the n+1 external nodes of binary tree. The tree with positive weights attached as external nodes in place of null nodes is called as ***weighted binary tree***.

The *weighted external path length* of such binary tree can be given as

$E=\Sigma\ q_i.k_i$

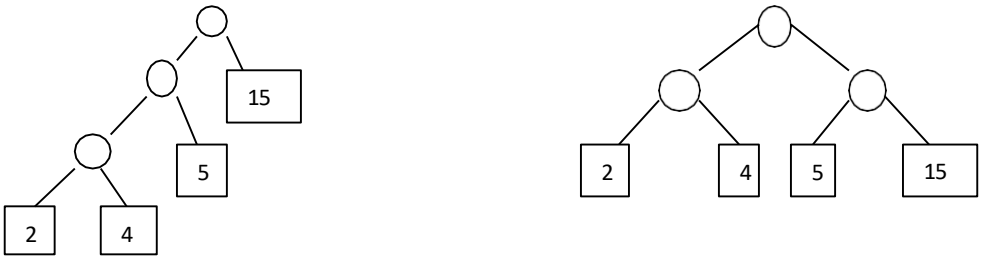
$1 \leq k \leq n$

where,

q→ Weight.

k→Distance from root node to the external node.

Consider the example in which n=3 and the four weights are given as $q_1=15$, $q_2=2$, $q_3=4$, $q_4=5$. The two possible trees would be as shown in Figure-8.4 below.



(a) (b)

Figure-1: Weighted binary tree

The weighed external path length of both the trees is given by

Tree-(a):

$E= 2 \cdot 3 + 4 \cdot 3 + 5 \cdot 2 + 15 \cdot 1 = 43$

Tree-(b):

$E= 2 \cdot 2 + 4 \cdot 2 + 5 \cdot 2 + 15 \cdot 2 = 52$

The binary tree with minimal weighted path length has application in various areas like:

1) To determine optimal merge pattern using 2-way merge.
2) To obtained optimal set of codes for given set of messages M1, M2……Mn+1 using Huffman code.

Optimal merge patter with 2-way merge

Here the optimal merge patterns are obtained for n+1 runs using 2-way merge. Consider four records $q_1=15$, $q_2=2$, $q_3=4$, $q_4=5$. Now in each run R_i, if we are to merge q_i records. If we have

respectively n and m records in two runs then merging will take O (n + m) time. Now let us merge the four records one by one.

R_1 has record q_1=15.
R_2 has record q_2=2.
R_3 has record q_3=4.
R_4 has record q_4=5.

Now we can merge R_2 and R_3 with time proportional to q_2+q_3.
Then we will merge the above result with R_4 with time proportional to $\{(q_2+q_3)+q_4\}$.
At last we merge the above result with R_1 with time proportional to $\{q_2+q_3+q_4+q_1\}$.

The total time of merge is $(q_2+q_3) + \{(q_2+q_3)+q_4\} + \{q_2+q_3+q_4+q_1\}$ which is optimal and the merge result will be the skewed binary tree with optimal or minimal weighted external path length as shown in Figure-1 (a).

In general if an external node R_i is at distance ki from the root of the merge tree, then the cost of merge will be proportional to

$\Sigma\ q_i.k_i$, which is weighted external path length.

Huffman Code using Huffman algorithm

Huffman algorithm is used to obtained optimal set of codes for given set of messages M1, M2……Mn+1. Each message or symbol can be transmitted as an encoded binary string and can be decoded back to original message using same binary tree, also called as decode tree, at receiving end. Decode tree is binary tree in which the external nodes represent the message. For a binary tree if we interpret left branch as zero (0) and right branch as one (1) then the path from root to external node will give us the binary encoded bit patter for a message. Consider the message list consisting messages M1, M2, M3, M4. Now using Huffman coding scheme we can represent a binary coded tree for message list as shown in Figure-2 below.

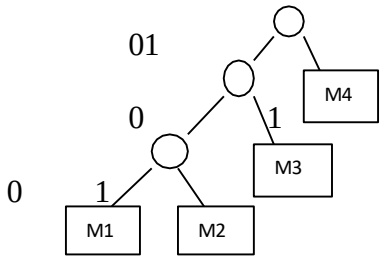


Figure-2: Huffman codes as Binary Decode tree

In above tree the codes 000, 001, 01 and 1 corresponds to the message M1, M2, M3 and M4 respectively. These codes are called as Huffman codes. The cost of decoding of code is proportional to the number of bits in the code. This number is equal to the distance of the corresponding external node from the root node. If message M_i is transmitted with frequency q_i then expected decoding time can be given as

$$T=\Sigma\ q_i \cdot d_i$$

$$1 \leq k \leq n+1$$

Where, d_i is the distance of the external node for message M_i from the root node.

This time can be minimized using Huffman algorithm which represents the given message list as binary tree with minimal weighted external path length. This algorithm is given below. The node structure for storing the weights can be represented using C-language structure as given below.

typedef struct Node

```
{
        struct Node *LC; // Left Child link
int weight; // weight of given word or symbol or code
 struct Node *RC; // Right Child link
}TREE;
```

Algorithm HUFFMAN (L, n)

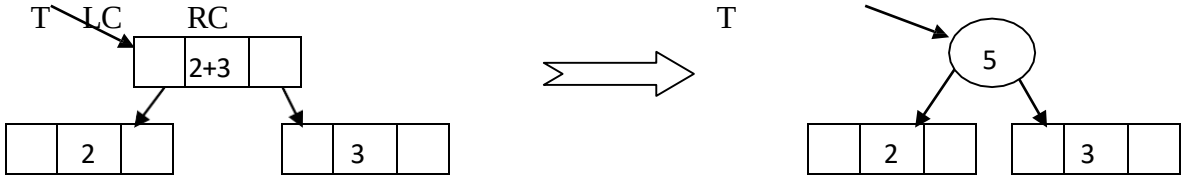
In this algorithm the ascending order priority List L of n single node binary tree is used for construction of decode tree. The sub-algorithm MIN finds a tree in L with minimum weight and removes it from L. The sub-algorithm INSERT adds a new tree to the list L. The HUFFMAN will iterate for n-1 number of passes. The pointer T is of TEE type.

```
1) for (i=1; i< n; i++) //loop n-1 times
2) {
3)    T= (TREE*) malloc(sizeof(TREE)); // allocate memory each time for new node to hold weight
4)    T->LC= MIN(L);//get first minimum tree node and remove it from L.
5)    T->RC=MIN(L); //get second minimum tree node and remove it from L.
6)    T->weight=T->LC->weight + T->RC->weight; // add these weights to T.
7)    INSERT(L,T);// add this new tree node to L.
8) }
```

Let us now try one example to illustrate the working of above algorithm. We have been given a list L of weights q_1=2, q_2=3, q_3=5, q_4=7, q_5=9 and q_6=13. The stepwise working of HUFFMAN is as given below.

I) L = {2, 3, 5, 7, 9, 13}

Remove first minimum 2 from L and attach it to LC of T. Then remove next minimum 3from L and attach it to RC of L as shown in figure below.

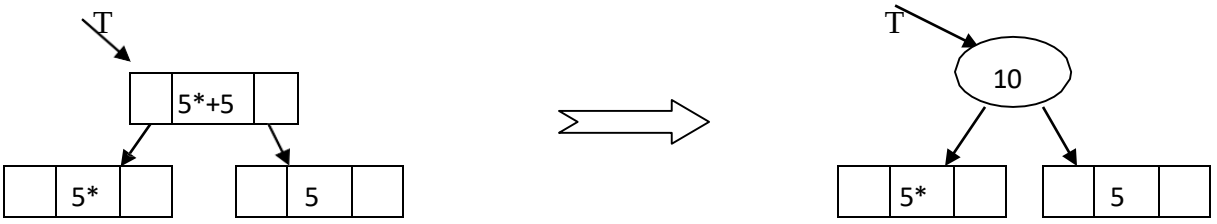


Now add 5* to the list L.

L = {5*, 5, 7, 9, 13}

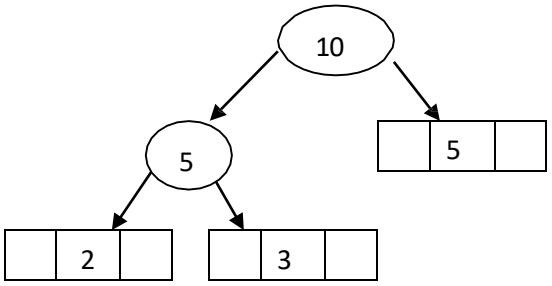
II) L = {5*, 5, 7, 9, 13}

Now remove 5* and then 5 from L and attach them as LC and RC of newly created T as shown in figure below.



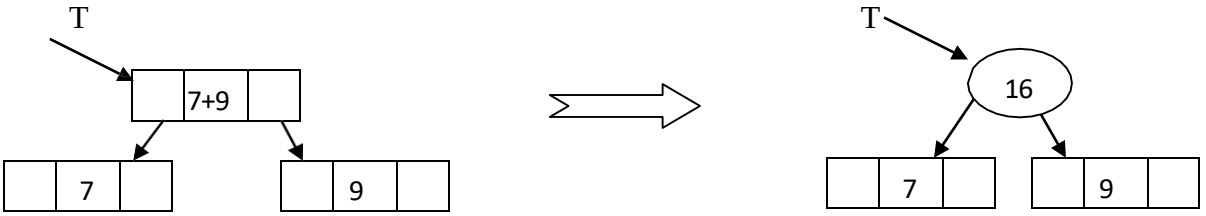
Now add 10* to the list L.

L = {7, 9, 10*, 13}



III) L = { 7, 9, 10*, 13}

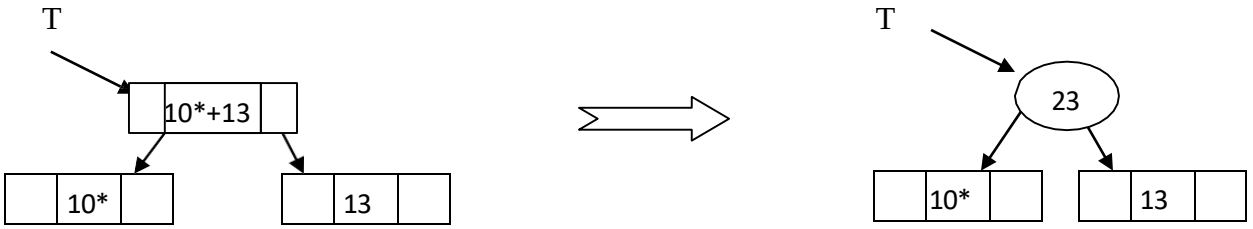
Now remove 7 and then 9 from L and attach them as LC and RC of newly created T as shown in figure below.



Now add 15* to the list L.
L = {10*, 13, 16*}

IV) L = {10*, 13, 16*}

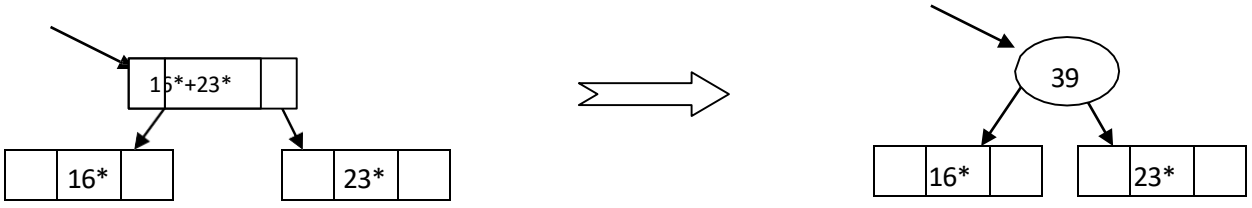
Now remove 10 and then 13 from L and attach them as LC and RC of newly created T as shown in figure below.



Now add 23* to the list L.
L = {16*, 23*}

V) L = {16*, 23*}

Now remove 16* and then 23* from L and attach them as LC and RC of newly created T as shown in figure below.



Now add 39* to the list L.
L = {39*}

Now the algorithm terminates as it has iterated for 5 times for given list of 6 elements. The final tree will look like binary tree as shown in Figure-8.6 below.

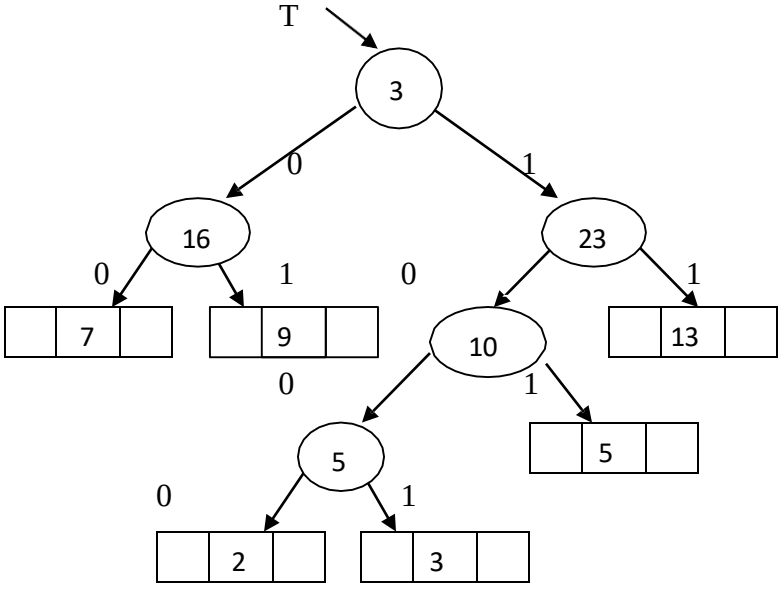


Figure-8.6: Binary tree using Huffman algorithm

Now let us calculate the weighted external path length for this tree.

$$E= 2 \cdot 4 + 3 \cdot 4 + 5 \cdot 3 + 13 \cdot 2 + 7 \cdot 2 + 9 \cdot 2 = 93$$

Code:

Analysis of HUFFMAN algorithm

The main for loop iterates for n-1 times. If list is maintained as an ascending order priority linked list then the MIN and INSERT will take O(n) time. So the time complexity of algorithm will be n(n-1), i.e. $O(n^2)$.

But if the least L is maintained as heap then each call to MIN and INSERT will require $O(\log_2 n)$ time. Hence the asymptotic computing time for the algorithm is $O(n\log_2 n)$.

HUFFMAN code and Prefix property

The Huffman codes have prefix property, i.e. we can decode strings of 0's and 1's by repeatedly deleting prefixes of the string that are codes of characters. Now consider the symbols with their frequency of occurrences as a=.02, b=.03, c= .05, d=.07, e=.07, f=.13. Now if we apply HUFFMAN algorithm to this list, the binary tree generated will look like Figure-8.6. The Table-8.3 gives the Huffman codes generated for given symbols.

Symbols	**Probability**	**Huffman Code**
a	.02	1000
b	.03	1001
c	.05	101
d	.07	00
e	.09	01
f	.13	11

Table-1: Huffman Codes for given symbols

Consider the strings of 0's and 1's as 011100101100111. Decoding of this string is given in Table-2.

Input	Prefix-Encode	Action	Output
01 11 00 101 1001 11	01-e	Delete prefix 01	e
11 00 101 1001 11	11-f	Delete prefix 11	ef
00 101 1001 11	00-d	Delete prefix 00	efd
101 1001 11	101-c	Delete prefix 101	efdc
1001 11	1001-b	Delete prefix 1001	efdcb
11	11-f	Delete prefix 11	efdcbf
-		Final Encoded string	efdcbf

Table-2: Encoding of string 01 11 00 101 1001 11 using Huffman codes of Table-1.

Facilities: Fedora Operating Systems, C/C++/Java/Python

Application:

Huffman coding Algorithm is used in various diversified areas of research, computer network, graph and cryptography.

Input:

Symbols with weights in sequence.

Output:

The Huffman encoding tree structure.

Conclusion:

The Huffman algorithm generates the binary tree with each symbol having unique encoding scheme.

Questions:

1. What is Huffam algorithm?
2. What is Huffam encoding?
3. What is the time complexity of Huffman algorithm?
4. What are the applications of Huffam coding/algorithm?

Experiment No: A3

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

Prerequisites:
- Concept of data structure like binary tree, priority queue and linked list.

Objectives:
Student should be able to implement fractional Knapsack problem using a greedy method.

Theory:

Knapsack Problem
Given a set of items, each with a weight and a value, determine a subset of items to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.
The knapsack problem is in combinatorial optimization problem.
A thief is robbing a store and can carry a maximal weight of W into his knapsack. There are n items available in the store and weight of i^{th} item is w_i and its profit is p_i. What items should the thief take?
In this context, the items should be selected in such a way that the thief will carry those items for which he will gain maximum profit. Hence, the objective of the thief is to maximize the profit.
Based on the nature of the items, Knapsack problems are categorized as

- Fractional Knapsack
- 0-1 Knapsack

Fractional Knapsack
In this case, items can be broken into smaller pieces, hence the thief can select fractions of items.
According to the problem statement,

- There are **n** items in the store
- Weight of $\mathbf{i^{th}}$ item wi>0
- Profit for $\mathbf{i^{th}}$ item pi>0 and
- Capacity of the Knapsack is **W**

In this version of Knapsack problem, items can be broken into smaller pieces. So, the thief may take only a fraction x_i of i^{th} item.

$$0 \leqslant xi \leqslant 1$$

The i^{th} item contributes the weight xi.wi to the total weight in the knapsack and profit xi.pi to the total profit.

Hence, the objective of this algorithm is to

$$\text{maximize} \sum_{n=1}^{n} (xi.pi)$$

subject to constraint,

$$\sum_{n=1}^{n} (xi.wi) \leqslant W$$

It is clear that an optimal solution must fill the knapsack exactly, otherwise we could add a fraction of one of the remaining items and increase the overall profit.

Thus, an optimal solution can be obtained by

$$\sum_{n=1}^{n} (xi.wi) = W$$

In this context, first we need to sort those items according to the value ofpiwi, so that (pi+1)/(wi+1) ≤ pi/wi.

Here, ***x*** is an array to store the fraction of items.

Algorithm: Greedy-Fractional-Knapsack (w[1..n], p[1..n], W)

```
for i = 1 to n
  do x[i] = 0

weight = 0

for i = 1 to n
  if weight + w[i] ≤ W then
     x[i] = 1
     weight = weight + w[i]
  else
     x[i] = (W - weight) / w[i]
     weight = W
break
return x
```

Analysis of Algorithm:

If the provided items are already sorted into a decreasing order of pi/wi, then the forloop calculating profit takes a time in ***O(n)***.

If items are not sorted then the sorting will take ***O(n logn)*** and the profit calculation will be taking ***O(n)***.

Therefore, the total time including the sort is ***O(n logn) + O(n)***. Thus, asymptotically the overall time taken by the algorithm is ***O(n logn)***.

Example:
Let us consider that the capacity of the knapsack ***W* = 60** and the list of provided items are shown in the following table –

Item	A	B	C	D
Profit	280	100	120	120
Weight	40	10	20	24
Ratio (pi/wi)	7	10	6	5

As the provided items are not sorted based on pi/wi. After sorting, the items are as shown in the following table.

Item	B	A	C	D
Profit	100	280	120	120
Weight	10	40	20	24
Ratio (pi/wi)	10	7	6	5

Solution
After sorting all the items according to pi/wi. First all of ***B*** is chosen as weight of ***B*** is less than the capacity of the knapsack. Next, item ***A*** is chosen, as the available capacity of the knapsack is greater than the weight of ***A***. Now, ***C*** is chosen as the next item. However, the whole item cannot be chosen as the remaining capacity of the knapsack is less than the weight of ***C***.
Hence, fraction of ***C*** (i.e. (60 − 50)/20) is chosen.
Now, the capacity of the Knapsack is equal to the selected items. Hence, no more item can be selected.
The total weight of the selected items is **10 + 40 + 20 * (10/20) = 60**
And the total profit is **100 + 280 + 120 * (10/20) = 380 + 60 = 440**
This is the optimal solution. We cannot gain more profit selecting any different combination of items.

C++ Code:

```
static int fractionalKS(FractionalKS Item[],int n,int W)
        {
                int i,finalValue,curWeight;

                //sort Items with their profit/weight ratio
                insertionSort(Item,n);
                finalValue=0;
                curWeight=W;
                for (i=0;i<n;i++)
                {
                        //If adding Item won't overflow,
                        //add it completely
                        if (Item[i].weight <= curWeight)
                        {
                                curWeight -= Item[i].weight;
                                finalValue += Item[i].profit;
                        }
                        // If we can't add current Item,
                        // add fractional part of it
                        else
                        {
                                finalValue += Item[i].profit * curWeight / Item[i].weight;
                                break;
                        }
                }

                return finalValue;
        }
```

Facilities: Fedora Operating Systems,C++

Application:

In many cases of resource allocation along with some constraint, the problem can be derived in a similar way of Knapsack problem. Following is a set of examples.

- Finding the least wasteful way to cut raw materials
- portfolio optimization
- Cutting stock problems

Input:

The n items with corresponding profit and weight.

Output:

The filled knapsack up to its capacity with maximum profit value.

Conclusion:

The fractional knapsack algorithm is implemented and test with different input values of profit & weights for ‘n’ items.

Questions:

1. What is knapsack problem?
2. What is fractional knapsack problem?
3. What are the application of fractional knapsack?
4. What are various methods used to solve knapsack problem other than greedy method?
5. What is greedy method of solving knapsack problem?

Experiment No:A4

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

Aim: Implement 0-1 Knapsack problem using branch and bound strategy.

Prerequisites:

- Concept of data structure like binary tree, priority queue and linked list.

Objectives:

Student should be able to implement 0-1 Knapsack problem using branch and bound strategy.

Theory:

Knapsack Problem

Given a set of items, each with a weight and a value, determine a subset of items to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

The knapsack problem is in combinatorial optimization problem.

A thief is robbing a store and can carry a maximal weight of W into his knapsack. There are n items available in the store and weight of i^{th} item is w_i and its profit is p_i. What items should the thief take?

In this context, the items should be selected in such a way that the thief will carry those items for which he will gain maximum profit. Hence, the objective of the thief is to maximize the profit.

Based on the nature of the items, Knapsack problems are categorized as

- Fractional Knapsack
- 0-1 Knapsack

0-1 Knapsack

any different combination of items.

Algorithm 01Knapsack()

1. Sort all items in decreasing order of ratio of value per unit weight so that an upper bound can be computed using Greedy Approach.
2. Initialize maximum profit, maxProfit = 0
3. Create an empty queue, Q.
4. Create a dummy node of decision tree and enqueue it to Q. Profit and weight of dummy node are 0.
5. Do following while Q is not empty.
    - Extract an item from Q. Let the extracted item be u.
    - Compute profit of next level node. If the profit is more than maxProfit, then update maxProfit.
    - Compute bound of next level node. If bound is more than maxProfit, then add next level node to Q.

- Consider the case when next level node is not considered as part of solution and add a node to queue with level as next, but weight and profit without considering next level nodes.

Analysis of Algorithm:
If the provided items are already sorted into a decreasing order of pi/wi, then the for loop calculating profit takes a time in ***O(n)***.

If items are not sorted then the sorting will take ***O(n logn)*** and the profit calculation using **Queue as data structure**will be taking ***O(n)***.

Therefore, the total time including the sort is ***O(n logn) + O(n)***. Thus, asymptotically the overall time taken by the algorithm is ***O(n logn)***.

Facilities: Fedora Operating Systems, C++

Application:
In many cases of resource allocation along with some constraint, the problem can be derived in a similar way of Knapsack problem. Following is a set of examples.

- Finding the least wasteful way to cut raw materials
- portfolio optimization
- Cutting stock problems

Input:
The n items with corresponding profit and weight.

Output:
The filled knapsack up to its capacity with maximum profit value.

Conclusion:
The fractional knapsack algorithm is implemented and test with different input values of profit & weights for ‘n’ items.

Questions:
1. What is knapsack problem?
2. What is 0-1 knapsack problem?
3. What are the application of 0-1 knapsack?
4. What are various methods used to solve knapsack problem other than greedy method?
5. What is branch and bound method of solving knapsack problem?

Assignment No: 5

Title of the Assignment: Design n-Queens matrix having first Queen placed. Use backtracking to place remaining Queens to generate the final n-queen's matrix.

Objective of the Assignment: Students should be able to understand and solve n-Queen Problem,and understand basics of Backtracking

Prerequisite:

1. Basic of Python or Java Programming
2. Concept of backtracking method
3. N-Queen Problem

Contents for Theory:

1. **Introduction to Backtracking**
2. **N-Queen Problem**

Introduction to Backtracking

- Many problems are difficult to solve algorithmically. Backtracking makes it possible to solve at least some large instances of difficult combinatorial problems.

Suppose we have to make a series of decisions among various choices, where

- We don't have enough information to know what to choose
- Each decision leads to a new set of choices.
- Some sequence of choices (more than one choices) may be a solution to your problem.

What is backtracking?

Backtracking is finding the solution of a problem whereby the solution depends on the previous steps taken.

For example, in a maze problem, the solution depends on all the steps you take one-by-one. If any of those steps is wrong, then it will not lead us to the solution. In a maze problem, we first choose a path and continue moving along it. But once we understand that the particular path is incorrect, then we just come back and change it. This is what backtracking basically is.

In backtracking, we first take a step and then we see if this step taken is correct or not i.e., whether it will give a correct answer or not. And if it doesn't, then we just come back and change our first step. In general, this is accomplished by recursion. Thus, in backtracking, we first start with a partial sub-solution of the problem (which may or may not lead us to the solution) and then check if we can proceed further with this sub-solution or not. If not, then we just come back and change it.

Thus, the general steps of backtracking are:

- start with a sub-solution
- check if this sub-solution will lead to the solution or not
- If not, then come back and change the sub-solution and continue again

Applications of Backtracking:

- N Queens Problem
- Sum of subsets problem
- Graph coloring
- Hamiltonian cycles.

N queens on NxN chessboard

One of the most common examples of the backtracking is to arrange N queens on an NxN chessboard such that no queen can strike down any other queen. A queen can attack horizontally, vertically, or diagonally. The solution to this problem is also attempted in a similar way. We first place the first queen anywhere arbitrarily and then place the next queen in any of the safe places. We continue this process until the number of unplaced queens becomes zero (a solution is found) or no safe place is left. If no safe place is left, then we change the position of the previously placed queen.

N-Queens Problem:

A classic combinational problem is to place n queens on a n*n chess board so that no two attack, i.,e no two queens are on the same row, column or diagonal.

What is the N Queen Problem?

N Queen problem is the classical Example of backtracking. N-Queen problem is defined as, “given N x N chess board, arrange N queens in such a way that no two queens attack each other by being in the same row, column or diagonal”.

- For N = 1, this is a trivial case. For N = 2 and N = 3, a solution is not possible. So we start with N = 4 and we will generalize it for N queens.

If we take n=4then the problem is called the 4 queens problem. If we take n=8 then the problem is called the 8 queens problem.

Algorithm

1) Start in the leftmost column

2) If all queens are place return true

3) Try all rows in the current column.

Do following for every tried row.

a) If the queen can be placed safely in this row then mark this [row, column] as part of the solution and recursively check if placing queen here leads to a solution.

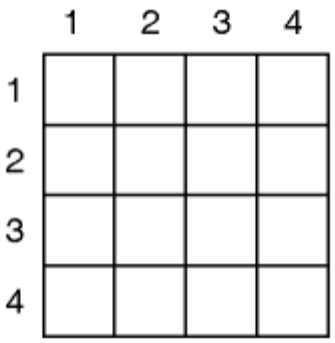
b) If placing the queen in [row, column] leads to a solution then return true.

c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.

4) If all rows have been tried and nothing worked,return false to trigger backtracking.

4-Queen Problem

Problem 1 : Given 4 x 4 chessboard, arrange four queens in a way, such that no two queens attack each other. That is, no two queens are placed in the same row, column, or diagonal.



4 x 4 Chessboard

- We have to arrange four queens, Q1, Q2, Q3 and Q4 in 4 x 4 chess board. We will put with queen in ith row. Let us start with position (1, 1). Q1 is the only queen, so there is no issue. partial solution is <1>
- We cannot place Q2 at positions (2, 1) or (2, 2). Position (2, 3) is acceptable. the partial solution is <1, 3>.
- Next, Q3 cannot be placed in position (3, 1) as Q1 attacks her. And it cannot be placed at (3, 2), (3, 3) or (3, 4) as Q2 attacks her. There is no way to put Q3 in the third row. Hence, the algorithm backtracks and goes back to the previous solution and readjusts the position of queen Q2. Q2 is moved from positions (2, 3) to
  (2, 4). Partial solution is <1, 4>
- Now, Q3 can be placed at position (3, 2). Partial solution is <1, 4, 3>.
- Queen Q4 cannot be placed anywhere in row four. So again, backtrack to the previous solution and readjust the position of Q3. Q3 cannot be placed on (3, 3) or(3, 4). So the algorithm backtracks even further.
- All possible choices for Q2 are already explored, hence the algorithm goes back to partial solution <1> and moves the queen Q1 from (1, 1) to (1, 2). And this process continues until a solution is found

  All possible solutions for 4-queen are shown in fig (a) & fig. (b)

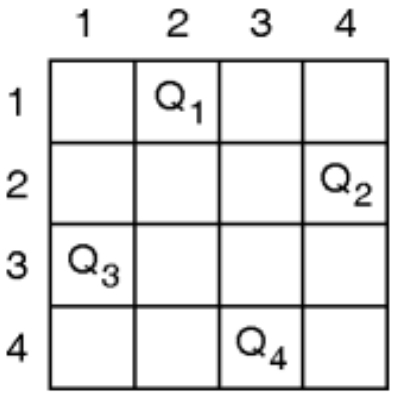


Fig. (a): Solution – 1

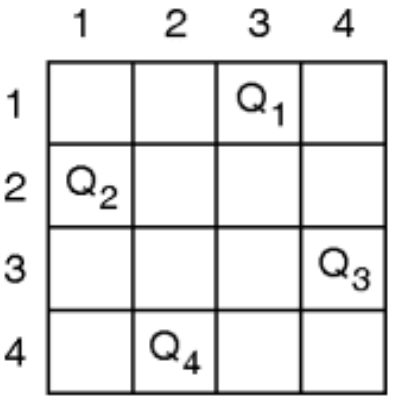
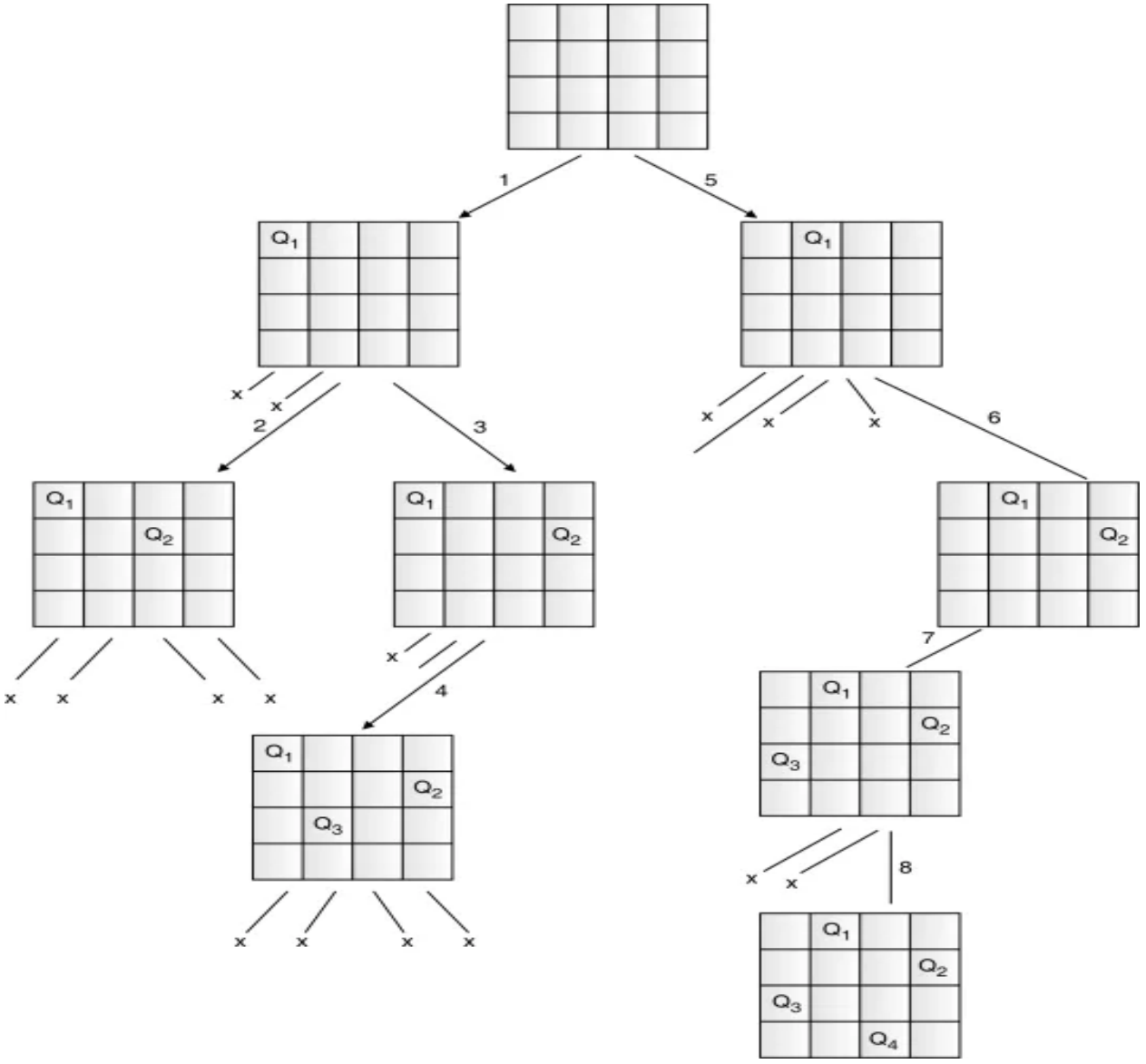


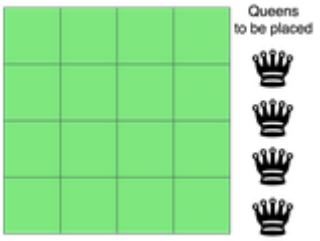
Fig. (b): Solution – 2

Fig. (d) describes thebacktracking sequence for the 4-queen problem.

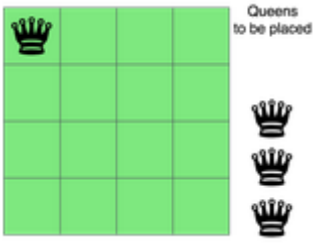


The solution of the 4-queen problem can be seen as four tuples (x_1, x_2, x_3, x_4), where x_i represents the column number of queen Q_i. Two possible solutions for the 4-queen problem are (2, 4, 1, 3) and (3, 1, 4, 2).

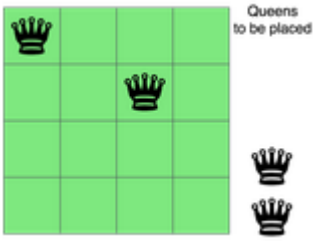
Explanation :

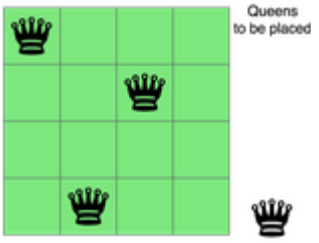


The above picture shows an NxN chessboard and we have to place N queens on it. So, we will start by placing the first queen.



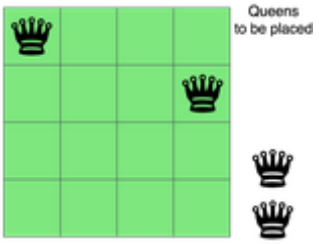
Now, the second step is to place the second queen in a safe position and then the third queen.



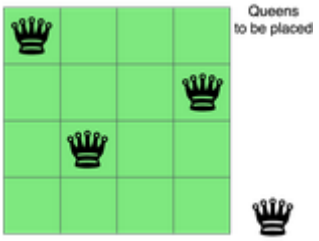


Now, you can see that there is no safe place where we can put the last queen. So, we will just change the position of the previous queen. And this is backtracking.

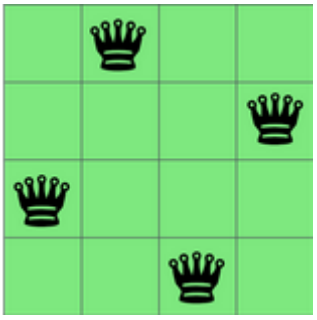
Also, there is no other position where we can place the third queen so we will go back one more step and change the position of the second queen.



And now we will place the third queen again in a safe position until we find a solution.

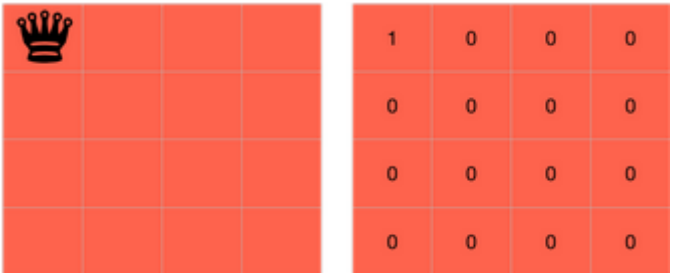


We will continue this process and finally, we will get the solution as shown below.

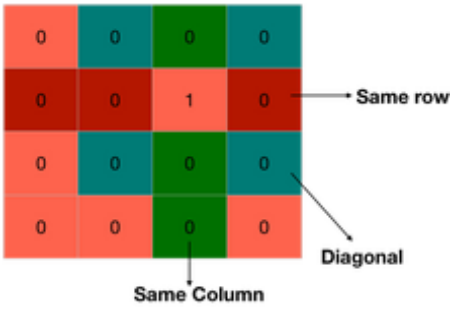


We need to check if a cell (i, j) is under attack or not. For that, we will pass these two in our function along with the chessboard and its size - IS-ATTACK(i, j, board, N).

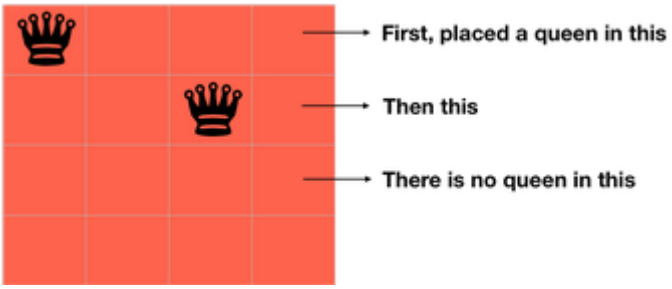
If there is a queen in a cell of the chessboard, then its value will be 1, otherwise, 0.



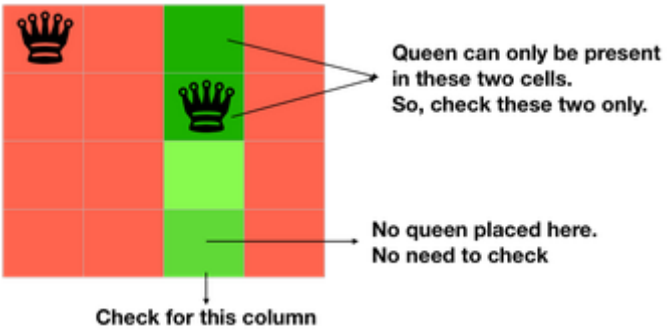
The cell (i,j) will be under attack in three condition - if there is any other queen in row i, if there is any other queen in the column j or if there is any queen in the diagonals.



We are already proceeding row-wise, so we know that all the rows above the current row(i) are filled but not the current row and thus, there is no need to check for row i.



We can check for the column j by changing k from 1 to i-1 in board[k][j] because only the rows from 1 to i-1 are filled.

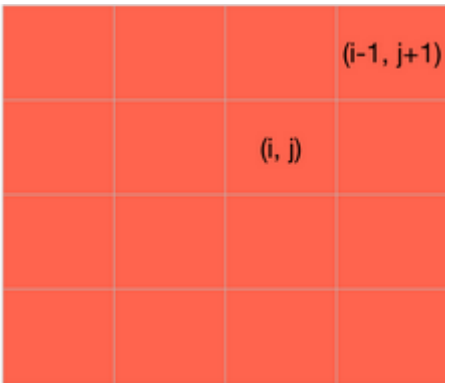


for k in 1 to i-1

if board[k][j]==1

return TRUE

Now, we need to check for the diagonal. We know that all the rows below the row i are empty, so we need to check only for the diagonal elements which above the row i. If we are on the cell (i, j), then decreasing the value of i and increasing the value of j will make us traverse over the diagonal on the right side, above the row i



k = i-1 l =

j+1

while k>=1 and l<=N if

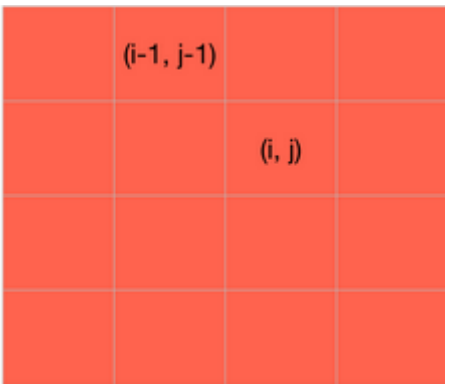
board[k][l] == 1 return

TRUE

k=k-1

l=l+1

Also if we reduce both the values of i and j of cell (i, j) by 1, we will traverse over the left diagonal, above the row i.



k = i-1 l =

j-1

while k>=1

and l>=1

if board[k][l] == 1 return

TRUE

k=k-1 l=l-

1

At last, we will return false as it will be return true is not returned by the above statements and the cell (i,j) is safe.

We can write the entire code as:

```
IS-ATTACK(i,j, board, N)
   //checking in the column j
     for k in 1 to i-1
          if board[k][j]==1
              return TRUE
   //Checking upper right diagonal
    K= i-1
    l = j+1
    while k>=1 and l<=N
     if board[k][l]==1
          return TRUE
        k=k+1
        l=l+1
   // checking
   upper left
```

diagonal k

= i-1

l = j-1

while

k>=1 and

l>=1 if

board[k][l]

== 1 return

TRUE

k=k-1 l=l-

1

return

FALSE

Now, let's write the real code involving backtracking to solve the N Queen problem.

Our function will take the row, number of queens, size of the board and the board itself - N-QUEEN(row, n, N, board).

If the number of queens is 0, then we have already placed all the queens. if

n==0

return TRUE

Otherwise, we will iterate over each cell of the board in the row passed to the function and for each cell, we will check if we can place the queen in that cell or not. We can't place the queen in a cell if it is under attack.

```
for j in 1 to N

  if !IS-ATTACK(row, j, board, N)

    board[row][j] = 1
```

After placing the queen in the cell, we will check if we are able to place the next queen with this arrangement or not. If not, then we will choose a different position for the current queen.

```
for j in 1 to N

    if N-QUEEN(row+1, n-1, N, board)

        return TRUE

    board[row][j] = 0
```

if N-QUEEN(row+1, n-1, N, board) - We are placing the rest of the queens with the current arrangement. Also, since all the rows up to 'row' are occupied, so we will start from 'row+1'. If this returns true, then we are successful in placing all the queen, if not, then we have to change the position of our current queen. So, we are leaving the current cell board[row][j] = 0 and then iteration will find another place for the queen and this is backtracking.

Take a note that we have already covered the base case - if n==0 → return TRUE. It means when all queens will be placed correctly, then N-QUEEN(row, 0, N, board) will be called and this will return true.

At last, if true is not returned, then we didn't find any way, so we will return false. N-

QUEEN(row, n, N, board)

...

```
   return FALSE
```

N-QUEEN(row, n, N, board)

if n==0

return TRUE

for j in 1 to N

if !IS-ATTACK(row, j, board, N)

```
        board[row][j] = 1

if N-QUEEN(row+1, n-1, N, board)

return TRUE

board[row][j] = 0 //backtracking, changing current decision

    return FALSE
```

Conclusion- In this way we have explored Concept of Backtracking method and solve n-Queen problem using backtracking method

Assignment Question

1. **What is backtracking? Give the general Procedure.**
2. **Give the problem statement of the n-queens problem. Explain the solution**
3. **Write an algorithm for N-queens problem using backtracking?**
4. **Why it is applicable to N=4 and N=8 only?**

Reference link

- https://www.codesdope.com/blog/article/backtracking-explanation-and-n-queens-problem/
- https://www.codesdope.com/course/algorithms-backtracking/\
- https://codecrucks.com/n-queen-problem/

GROUP- B

Assignment No:1

Title of the Assignment: 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:

1. Data Preprocessing
2. Linear regression
3. Random forest regression models
4. Box Plot
5. Outliers
6. Haversine
7. Mathplotlib
8. Mean Squared Error

Data Preprocessing:

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first 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 machine learning model.

It involves below steps:

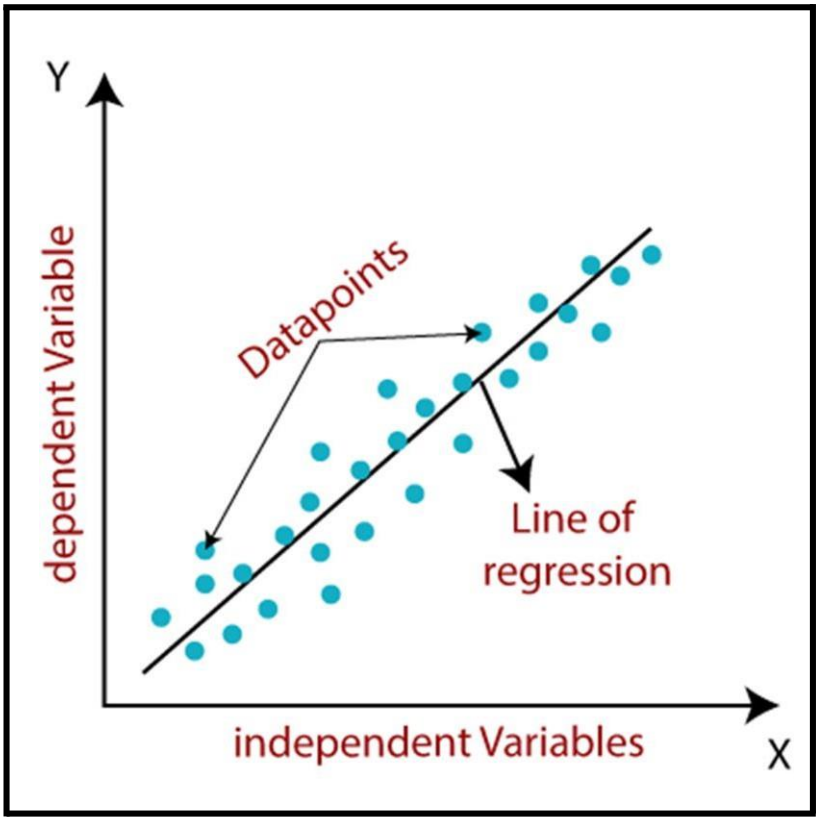
- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

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 finds 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 complex problem 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 takes the

prediction from each tree and based on the majority votes of predictions, and it predicts the
final
output.

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

Boxplot:

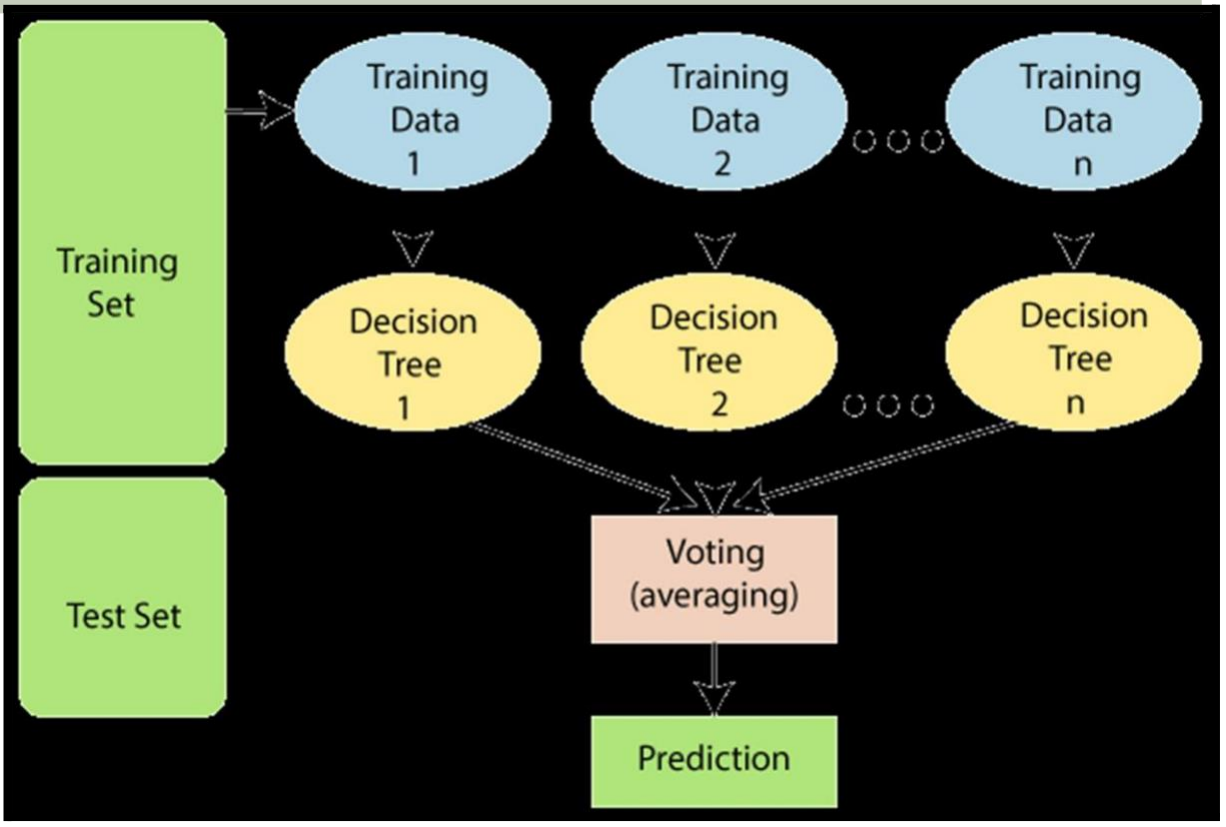
Boxplots are a measure of how well data is distributed across a data set. This divides the data set into three quartiles. This graph represents the minimum, maximum, average, first quartile, and the third quartile in the data set. Boxplot is also useful in comparing the distribution of data in a data set by drawing a boxplot for each of them.

R provides a boxplot() function to create a boxplot. There is the following syntax of boxplot()
function
:

boxplot(x, data, notch, varwidth, names, main)

Here
,

S.No	Parameter	Description



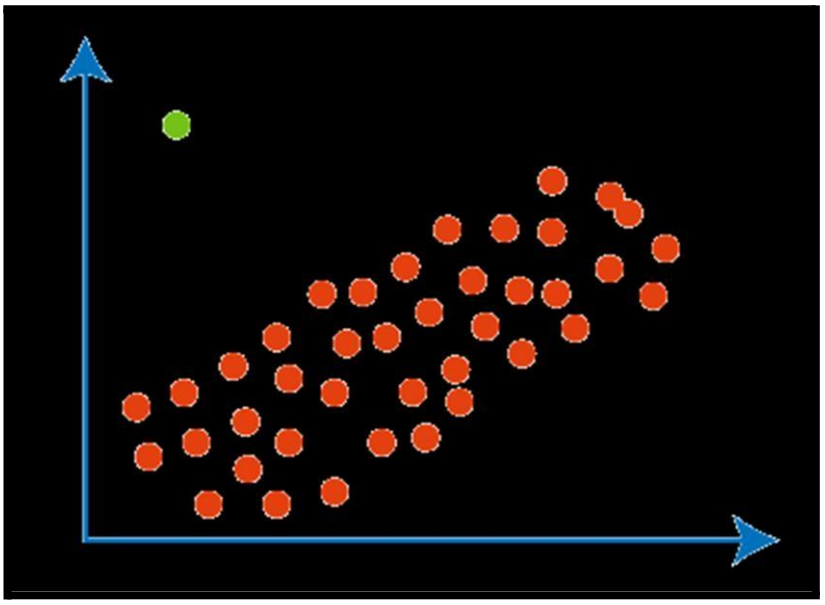
1.	x	It is a vector or a formula.
2.	data	It is the data frame.
3.	notch	It is a logical value set as true to draw a notch.
4.	varwidth	It is also a logical value set as true to draw the width of the box same as the sample size.
5.	names	It is the group of labels that will be printed under each boxplot.
6.	main	It is used to give a title to the graph.

Outliers:

As the name suggests, "outliers" refer to the data points that exist outside of what is to be expected. 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 find some data points that are likely to contain some form of error, then these are definitely 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 first definition of outliers.

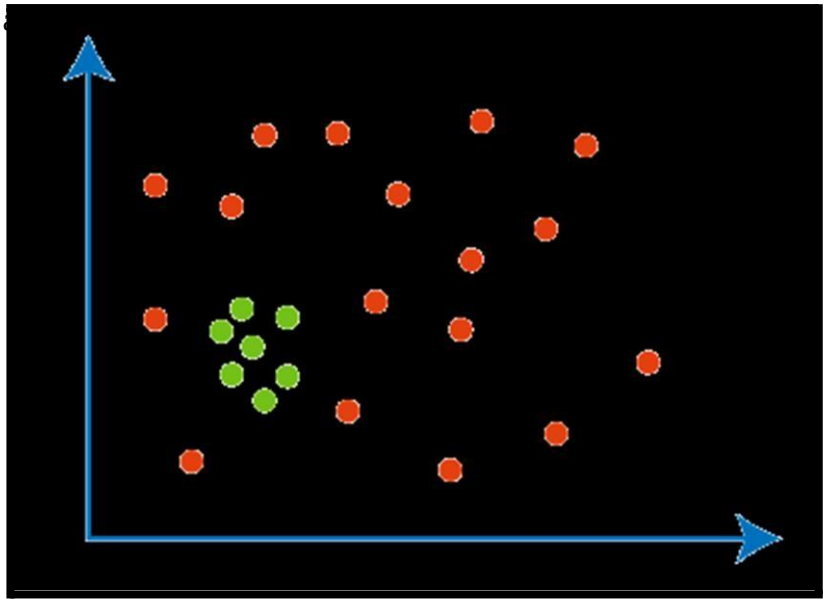
Global Outliers

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.



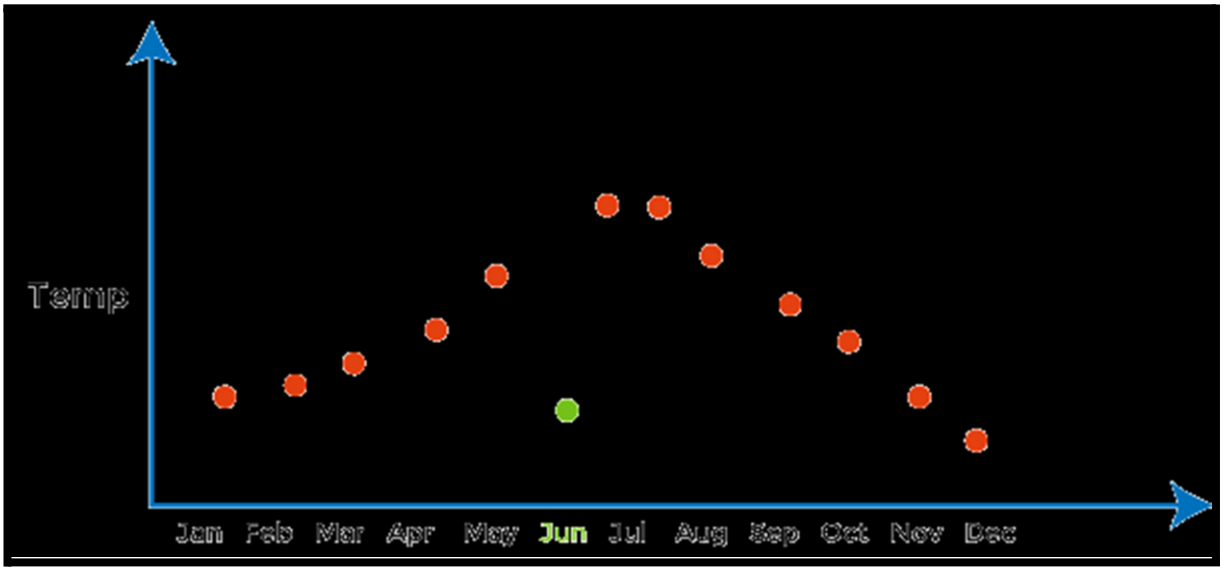
Collective Outliers

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 behavior 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 computer simultaneously, it is considered abnormal behavior, and as a whole, they are called collective outliers. The green data points as



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 speci□c 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 is not 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 of the Assignment: 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:

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 *k*nowledge 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 machine learning model. It is the first 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 machine learning 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 scaling

Binary Classification

Binary classification refers to those classification tasks that have two class labels.

Examples include:

- Email spam detection (spam or not).
- Churn prediction (churn or not).
- Conversion prediction (buy or not).

Typically, binary classification tasks involve one class that is the normal state and another class that is the abnormal state.

For example "*not spam*" is the normal state and "*spam*" is the abnormal state. Another example is "*cancer not detected*" is the normal state of a task that involves a medical test and "*cancer detected*" is the abnormal state.

The class for the normal state is assigned the class label 0 and the class with the abnormal state is assigned the class label 1.

It is common to model a binary classification task with a model that predicts a Bernoulli probability distxribution for each example.

The Bernoulli distribution is a discrete probability distribution that covers a case where an event will have a binary outcome as either a 0 or 1. For classification, this means that the model predicts a probability of an example belonging to class 1, or the abnormal state.

Popular algorithms that can be used for binary classification include:

- Logistic Regression

- k-Nearest Neighbors
- Decision Trees
- Support Vector Machine
- Naive Bayes

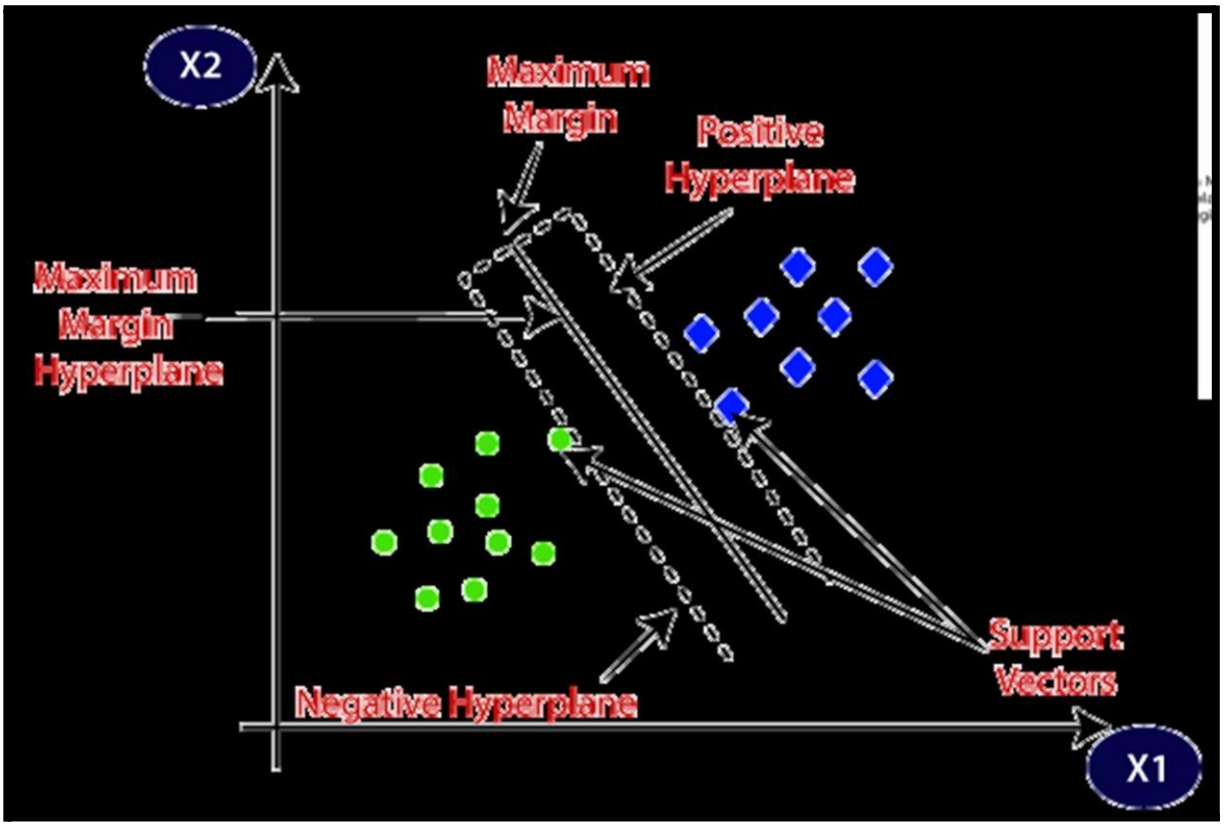
Some algorithms are specifically designed for binary classification and do not natively support more than two classes; examples include Logistic Regression and Support Vector Machines.

Support Vector Machine:

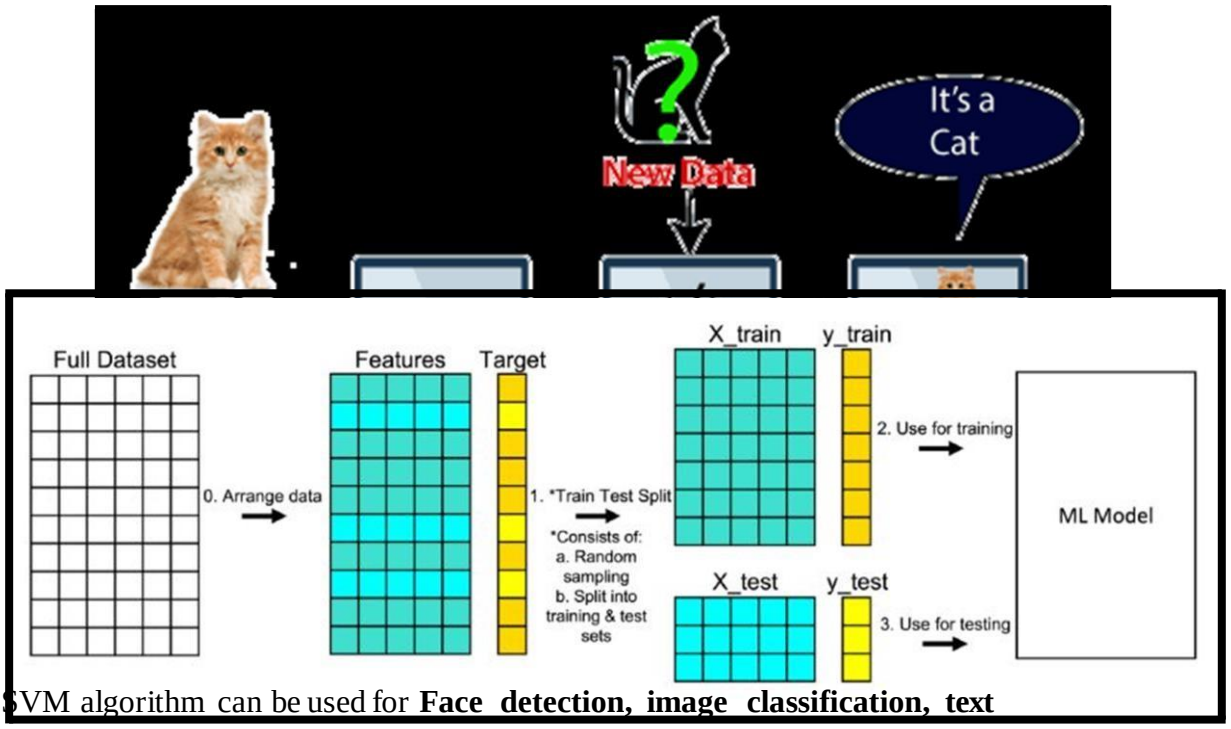
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using adecision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for **Face detection, image classification, text categorization,**
etc.

Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Train, Test, Split Procedure:

Train test split is a model validation procedure that allows you to simulate how a model would perform on new/unseen data. Here is how the procedure works:

1. ARRANGE THE DATA

Make sure your data is arranged into a format acceptable for train test split. In scikit-learn, this consists of separating your full data set into "Features" and "Target."

2. SPLIT THE DATA

Split the data set into two pieces — a training set and a testing set. This consists of random sampling without replacement about 75 percent of the rows (you can vary this) and putting them into your training set. The remaining 25 percent is put into your test set. Note that the colors in "Features" and "Target" indicate where their data will go ("X_train," "X_test," "y_train," "y_test") for a particular train test split.

3. TRAIN THE MODEL

Train the model on the training set. This is "X_train" and "y_train" in the image.

4. TEST THE MODEL

Test the model on the testing set ("X_test" and "y_test" in the image) and evaluate the performance.

Conclusion:

In this way we have explored Concept of Email Spam detection by using binary classification.

Assignment Questions:

1. What is Binary Classification?
2. Explain Support Vector Machine?
3. Explain K-Nearest Neighbour algorithm for Machine Learning?

Assignment No:3

—---

Title of the Assignment: 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 CustomerId, CreditScore, Geography, Gender, Age, Tenure, Balance, etc.

Link for Dataset:https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling

Perform the 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:

1. Basic knowledge of Python
2. Concept of Confusion Matrix

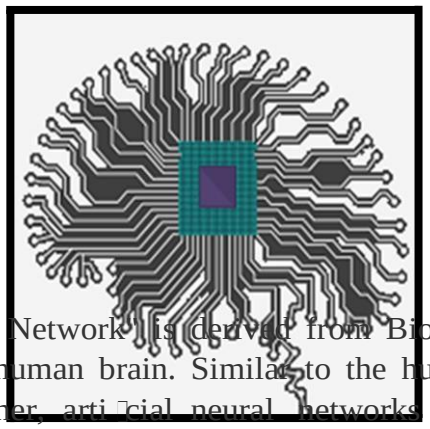
x

Contents of the Theory:

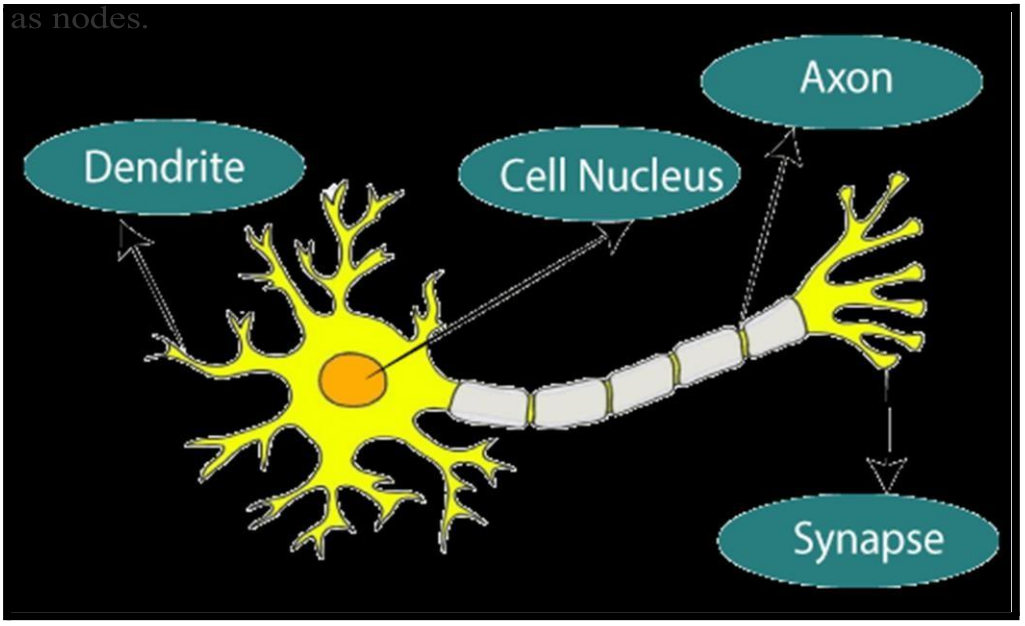
1. Artificial Neural Network
2. Keras
3. tensorflow

4. Normalization
5. Confusion Matrix

Artificial Neural Network:

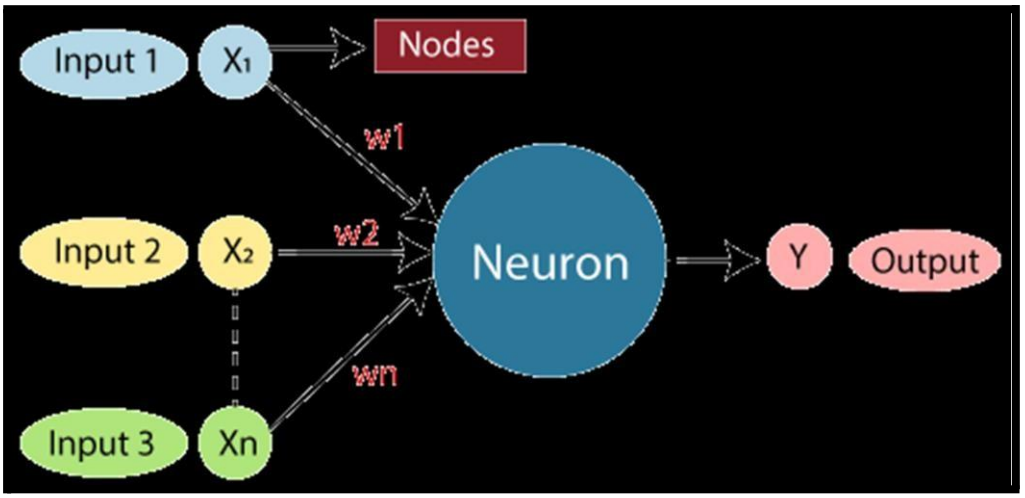


The term "Arti cial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, arti cial neural networks also have neurons that are interconnected to one another in various layers of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



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

The typical Arti cial 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

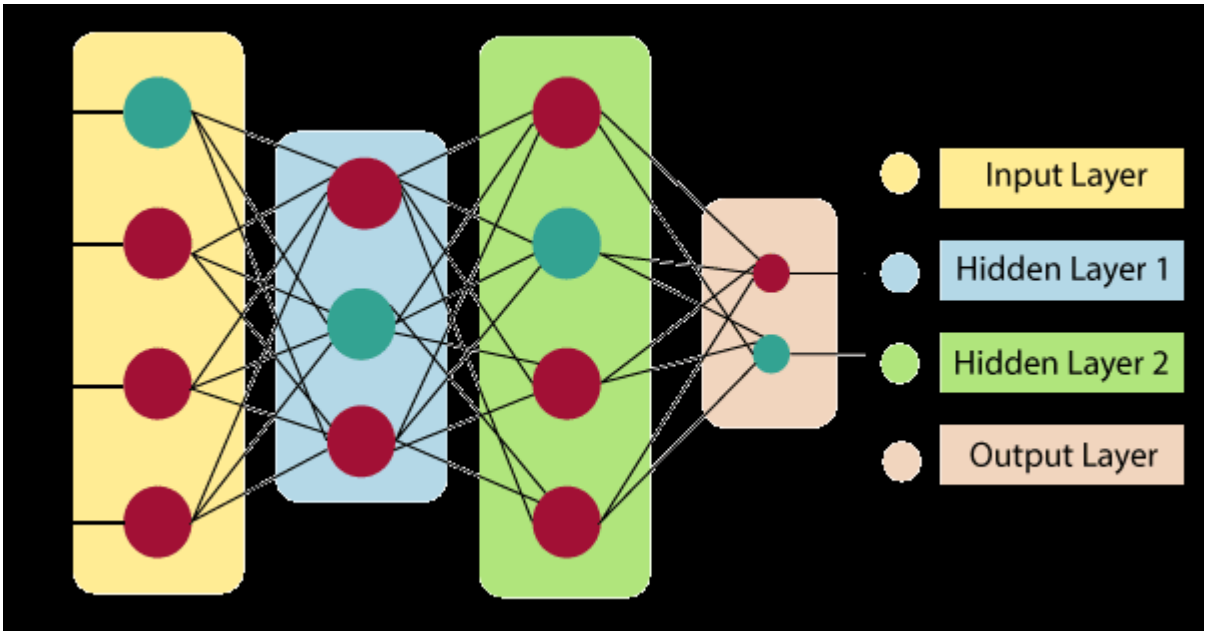
An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming 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 parallelly. 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 define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Lets us look at various types of layers available in an artificial neural network.

Artificial Neural Network primarily consists of three layers:



Input Layer:

As the name suggests, it accepts inputs in several different formats provided by the programmer.

Hidden Layer:

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

Output Layer:

The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} Wi * Xi + b$$

It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available

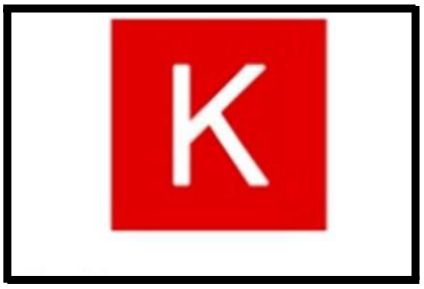
that can be applied upon the sort of task we are performing.

Keras:

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.

It cannot handle low-level computations, so it makes use of the**Backend**library to resolve it. The backend library act as a high-level API wrapper for the low-level API, which lets it run on TensorFlow, CNTK, or Theano.

Initially, it had over 4800 contributors during its launch, which now has gone up to 250,000 developers. It has a 2X growth ever since every year it has grown. Big companies like Microsoft, Google, NVIDIA, and Amazon have actively contributed to the development of Keras. It has an amazing industry interaction, and it is used in the development of popular
rms likes Net ix, Uber, Google, Expedia,
etc.



Tensor ow:

TensorFlow is a Google product, which is one of the most famous deep learning tools widely used in the research area of machine learning and deep neural network. It came into the market on 9th November 2015 under the Apache License 2.0. It is built in such a way that it can easily run on multiple CPUs and GPUs as well as on mobile operating systems. It consists of various wrappers in distinct languages such as Java, C++, or Python.

Normalization:

Normalization is a scaling technique in Machine Learning applied during data preparation to change the values of numeric columns in the dataset to use a common scale. It is not necessary for all datasets in a model. It is required only when features of machine learning models have different ranges.

Mathematically, we can calculate normalization with the below

formula: Xn = (X - Xminimum) / (Xmaximum - Xminimum)

Where,

- Xn = Value of Normalization
- Xmaximum = Maximum value of a feature
- Xminimum = Minimum value of a feature

Example: Let's assume we have a model dataset having maximum and minimum values of feature as mentioned above. To normalize the machine learning model, values are shifted and rescaled so their range can vary between 0 and 1. This technique is also known as Min-Max scaling. In this scaling technique, we will change the feature values as follows:

Case1-If the value of X is minimum, the value of Numerator will be 0; hence Normalization will also be 0.

Xn = (X - Xminimum) / (Xmaximum - Xminimum) ---------- formula

Put X =Xminimum in above formula, we get;

Xn = Xminimum- Xminimum/ (Xmaximum -

Xminimum) Xn = 0

Case2-If the value of X is maximum, then the value of the numerator is equal to the denominator; hence Normalization will be 1.

Xn = (X - Xminimum) / (Xmaximum -

Xminimum) Put X =Xmaximum in above formula, we

get;

Xn = Xmaximum - Xminimum/ (Xmaximum -

Xminimum) Xn = 1

Case3-On the other hand, if the value of X is neither maximum nor minimum, then values of normalization will also be between 0 and 1.

Hence, Normalization can be de ned as a scaling method where values are shifted and rescaled to maintain their ranges between 0 and 1, or in other words; it can be referred to as Min-Max scaling technique.

Normalization techniques in Machine Learning

Although there are so many feature normalization techniques in Machine Learning, few of them are most frequently used. These are as follows: ●**Min-Max Scaling:**This technique is also referred to as scaling. As we have already discussed above, the Min-Max scaling method helps the dataset to shift and rescale the values of their attributes, so they end up ranging between 0 and 1.

- **Standardization scaling:**

Standardization scaling is also known as**Z-score**normalization, in which values are centered around the mean with a unit standard deviation, which means the attribute becomes zero and the resultant distribution has a unit standard deviation. Mathematically, we can calculate the standardization by subtracting the feature value from the mean and dividing it by standard deviation.

Hence, standardization can be expressed as follows:

$$X' = \frac{X - \mu}{\sigma}$$

Here,**μ**represents the mean of feature value, and**σ**represents the standard deviation of feature values.

However, unlike Min-Max scaling technique, feature values are not restricted to a speci c range in the standardization technique.

This technique is helpful for various machine learning algorithms that use distance measures such as**KNN, K-means clustering, and Principal component analysis**, etc. Further, it is also important that the model is built on assumptions and data is normally distributed.

When to use Normalization or Standardization?

Which is suitable for our machine learning model, Normalization or Standardization? This is probably a big confusion among all data scientists as well as machine learning engineers. Although both terms have the almost same meaning choice of using normalization or standardization will depend on your problem and the algorithm you are using in models.

1. Normalization is a transformation technique that helps to improve the performance as well as the accuracy of your model better. Normalization of a machine learning model is useful when you don't know feature distribution exactly. In other words, the feature distribution of data does not follow a**Gaussian**(bell curve) distribution. Normalization must

have an abounding range, so if you have outliers in data, they will be affected by

Normalizatio
n.

Further, it is also useful for data having variable scaling techniques such as**KNN, arti cial neural network**s. Hence, you can't use assumptions for the distribution of data.

2. Standardization in the machine learning model is useful when you are exactly aware of the feature distribution of data or, in other words, your data follows a Gaussian distribution. However, this does not have to be necessarily true. Unlike Normalization, Standardization does not necessarily have a bounding range, so if you have outliers in your data, they will not be affected by Standardization.

Further, it is also useful when data has variable dimensions and techniques such as**linear regression, logistic regression, and linear discriminant analysis**.

Example: Let's understand an experiment where we have a dataset having two attributes, i.e., age and salary. Where the age ranges from 0 to 80 years old, and the income varies from 0 to
75,000 dollars or more. Income is assumed to be 1,000 times that of age. As a result, the ranges of these two attributes are much different from one another.

Because of its bigger value, the attributed income will organically in uence the conclusion more when we undertake further analysis, such as multivariate linear regression. However, this does not necessarily imply that it is a better predictor. As a result, we normalize the data so that all of the variables are in the same range.

Further, it is also helpful for the prediction of credit risk scores where normalization is applied to all numeric data except the class column. It uses the**tanh transformation** technique, which converts all numeric features into values of range between 0 to 1.

Confusion Matrix:

The confusion matrix is a matrix used to determine the performance of the classi cation 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 classi ers, the matrix is of 2*2 table, for 3 classes, it is
  3*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 of **100 predictions**. Out of 100 predictions, **89 are true predictions**, and **11 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 classifier to all number of 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 defines 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 defined 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 calculated 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 difficult 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 classifier has a higher error rate than the null error rate.*"
- **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 a neural network-based classifier that can determine whether they will leave or not in the next 6 months

Assignment Questions:

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

Assignment No:4

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

Dataset Description: This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes

Link for Dataset: https://www.kaggle.com/datasets/abdallamahgoub/diabetes

Objective of the Assignment:
Students should be able to learn the concept of K-Nearest Neighbours and Confusion Matrix

Prerequisite:
1. Basic knowledge of Python
2. Concept of Confusion Matrix
3. Concept of K-Nearest Neighbour

Contents of the Theory:

1. K-Nearest Neighbour
2. Confusion Matrix
3. Scikit learn

K-Nearest Neighbour:

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

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: Sample Sales Data, Order Info, Sales, Customer, Shipping, etc., Used for Segmentation, Customer Analytics, Clustering and More. Inspired for retail analytics. This was originally used for Pentaho DI Kettle, But I found the set could be useful for Sales Simulation training.

Originally Written by María Carina Roldán, Pentaho Community Member, BI consultant (Assert Solutions), Argentina. This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License. Modified by Gus Segura June 2014.

Link for Dataset: https://www.kaggle.com/datasets/kyanyoga/sample-sales-data

Objective of the Assignment:
Students should be able to learn the concept of K-Means clustering and Hierarchical clustering

Prerequisite:
1. Basic knowledge of Python
2. Concept of K-Means Clustering
3. Concept of Hierarchical Clustering

Contents of the Theory:

1. K-Means Clustering
2. Hierarchical Clustering
3. Principal Component Analysis(PCA)
4. Elbow Method

K-Means Clustering Algorithm

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

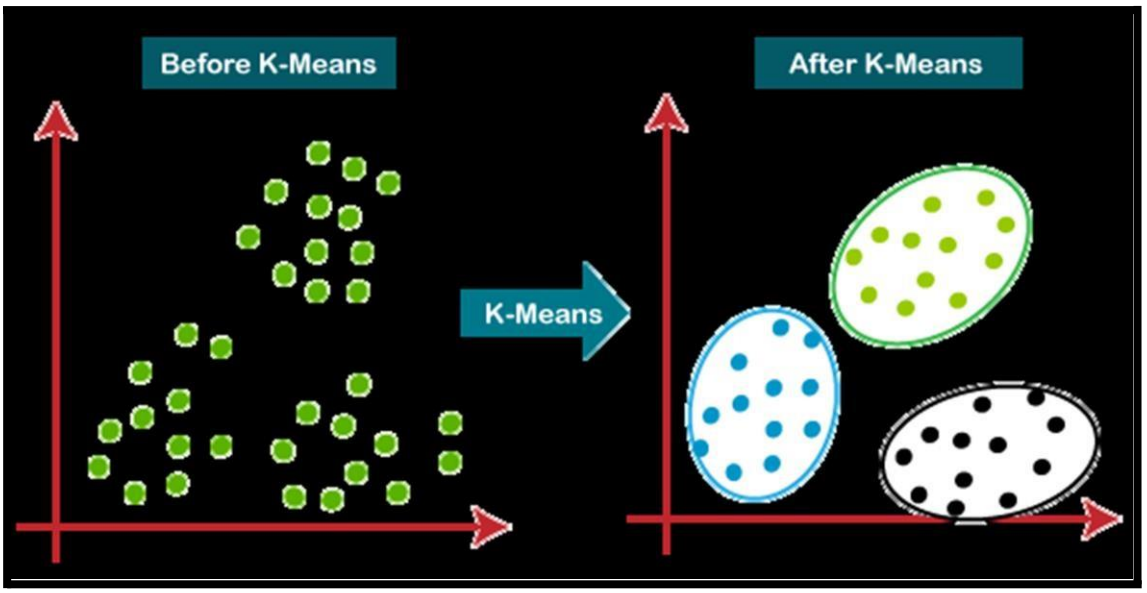
The k-means clustering

algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps: Step-1: Select the number K to decide the number of clusters.
Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

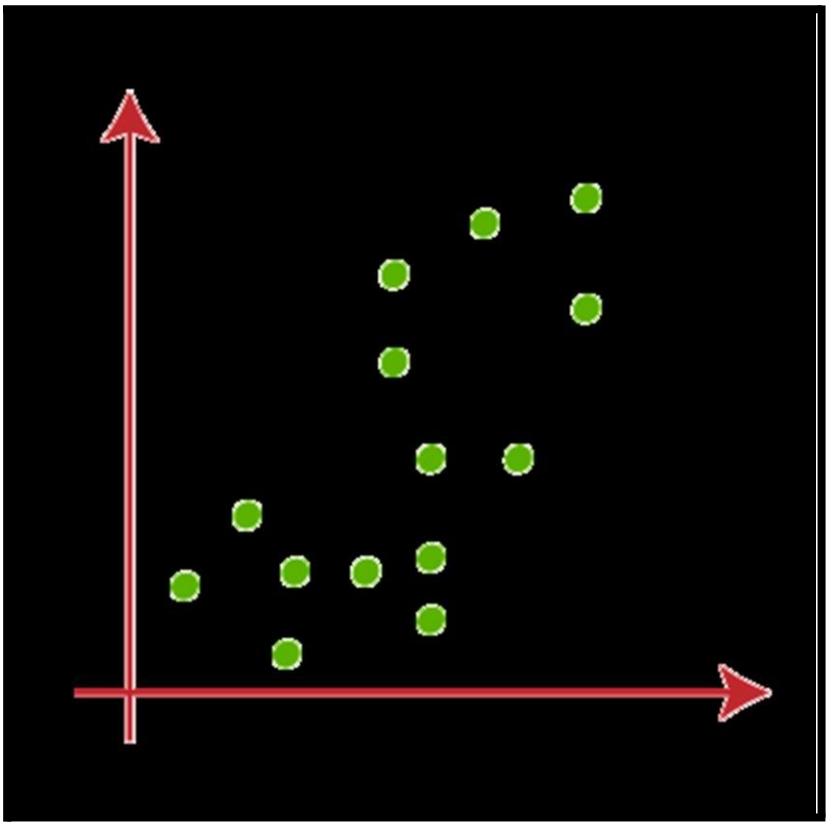
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

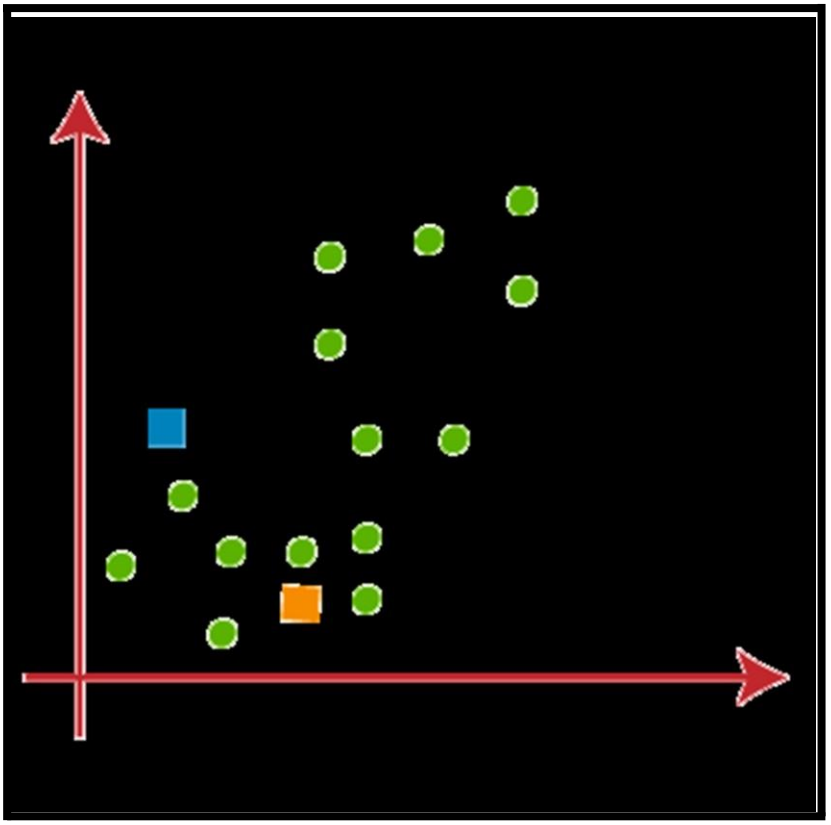
Step-7: The model is ready.

Let's understand the above steps by considering the visual plots:

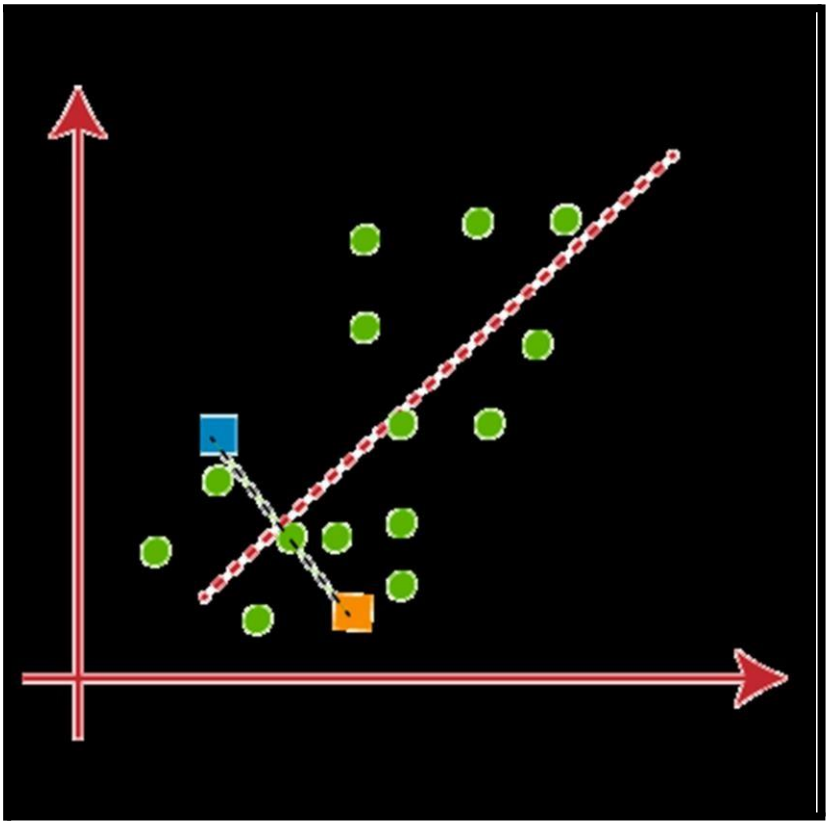
Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



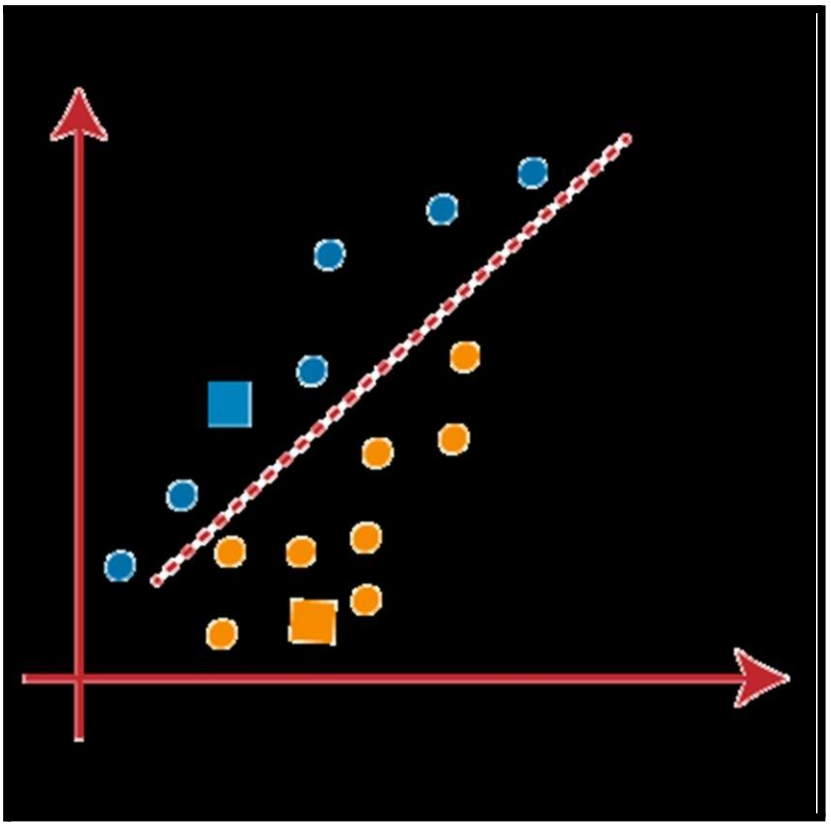
- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset or any other point. So, here we are selecting the below two points as k points, which are not the part of our dataset. Consider the below image



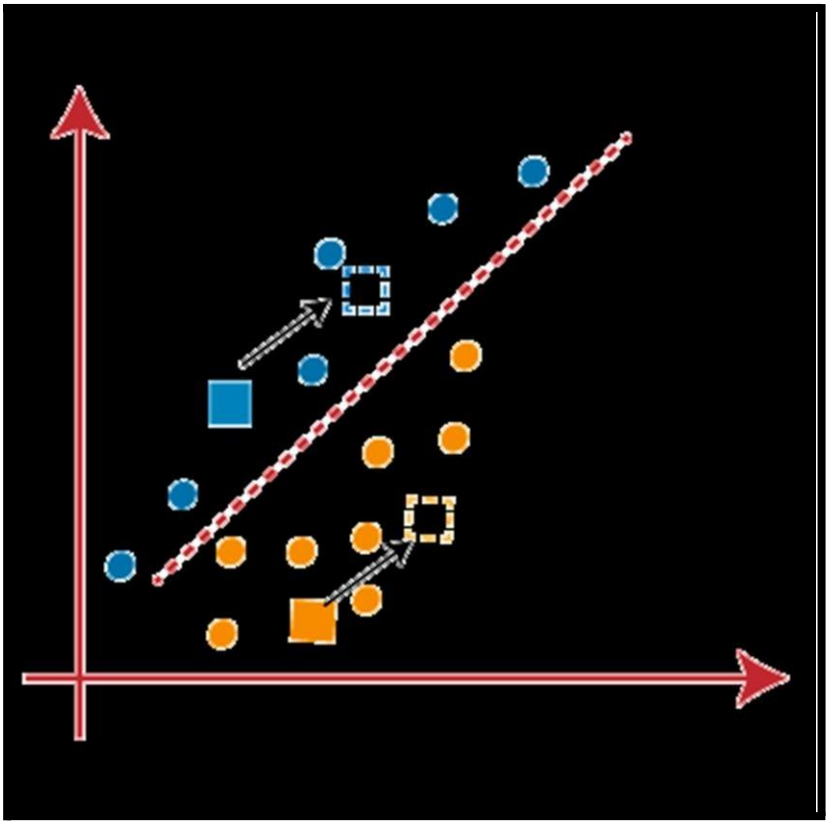
Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:



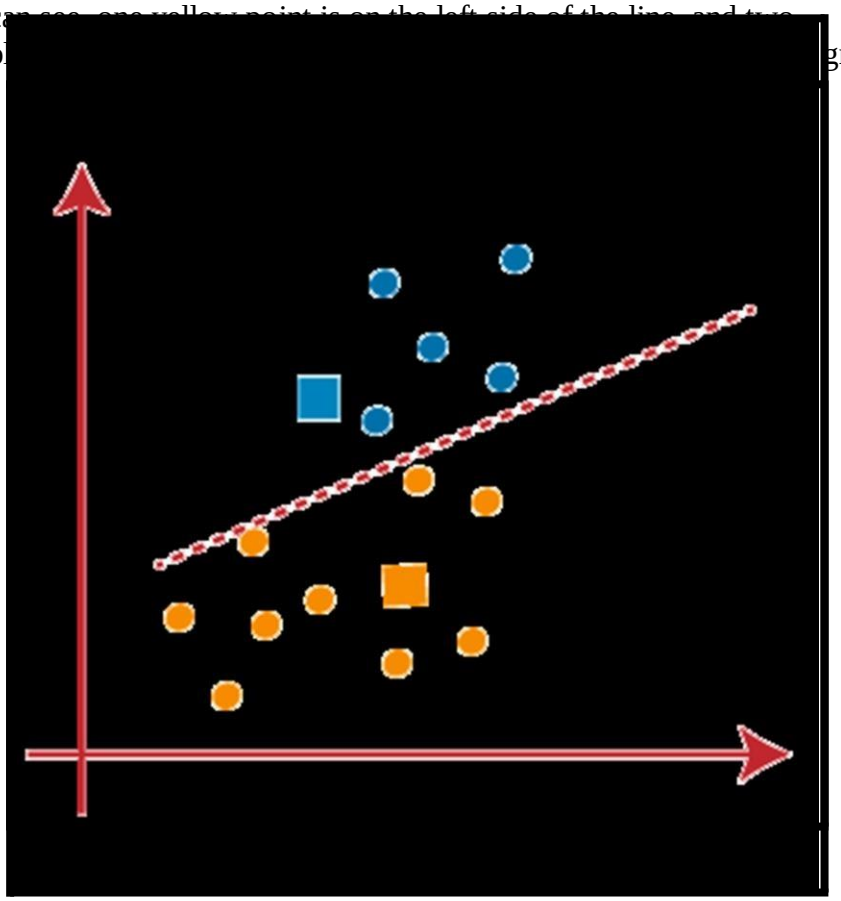
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:

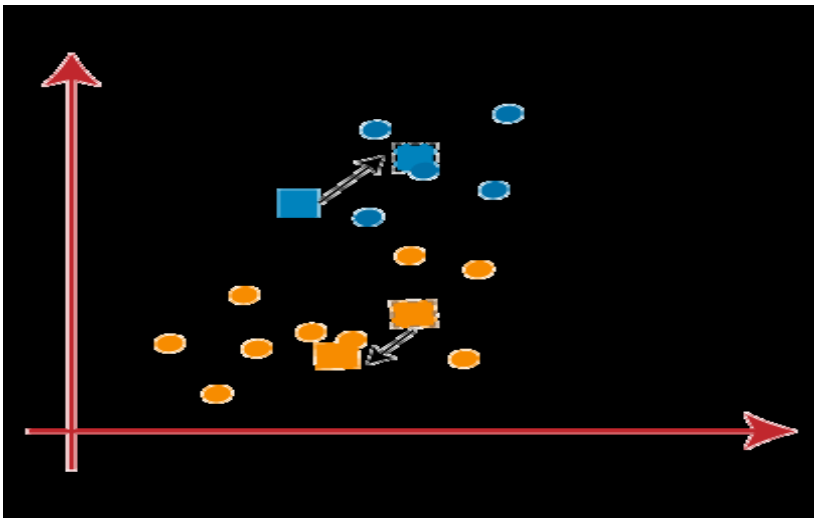


Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below imag From the above image, we ca see, one yellow point is on the left side of the line, and two b gned to new centroids.



As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

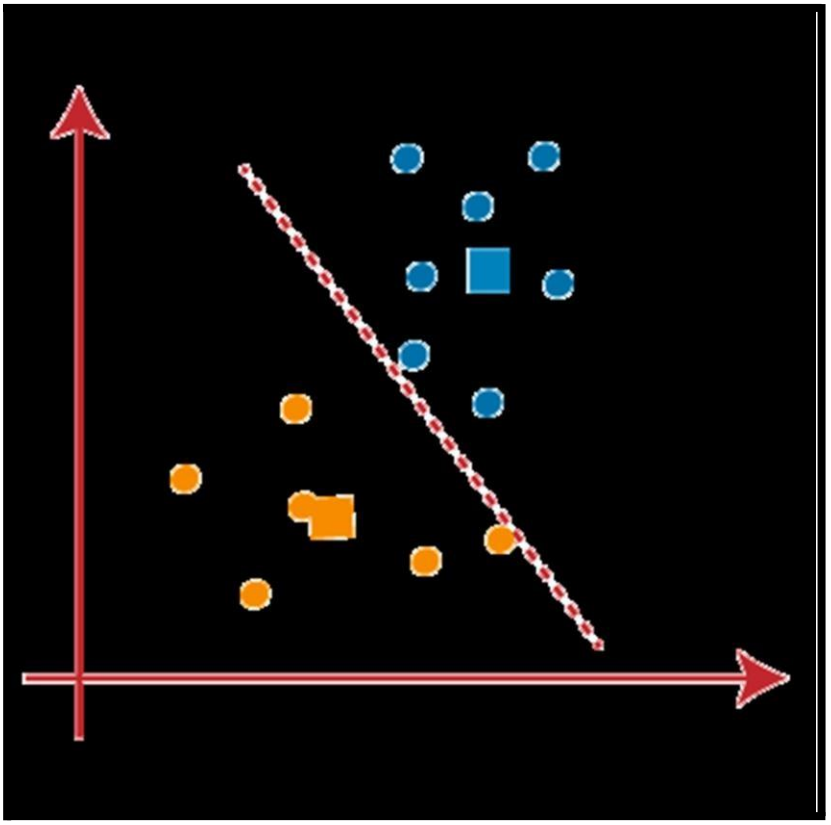
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



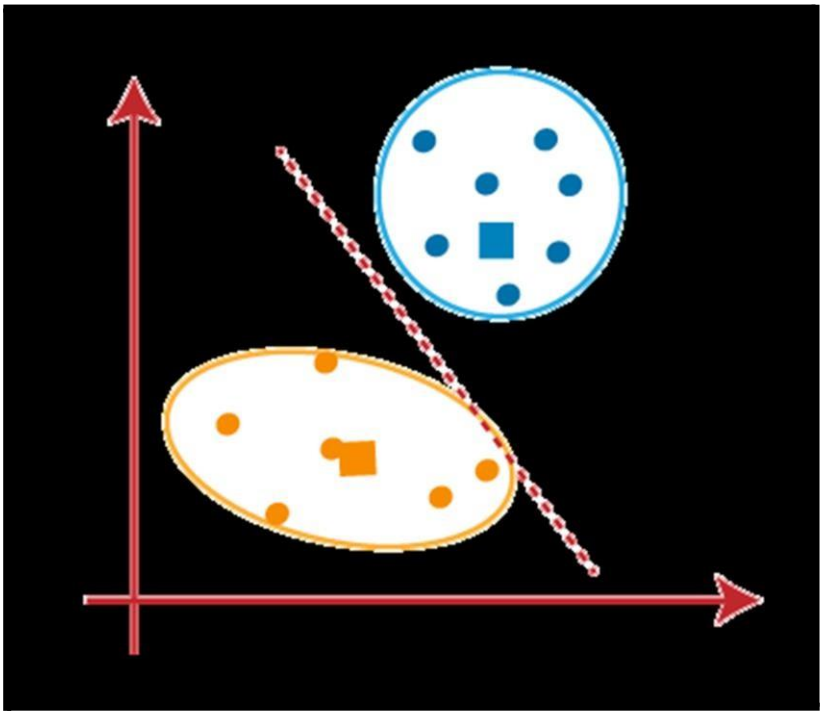
e:

As we got the new centroids so again will draw the median line and reassign the data points.

So, the image will be:



We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:



As our model is ready, so we can now remove the assumed centroids, and the two final clusters will be as shown in the below image:

HierarchicalClustering: Hierarchical clustering is another unsupervised machine learning algorithm, which is used to
group the unlabeled datasets into a cluster and also known as **hierarchical cluster analysis** or HCA.

In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.

Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

The hierarchical clustering technique has two approaches:

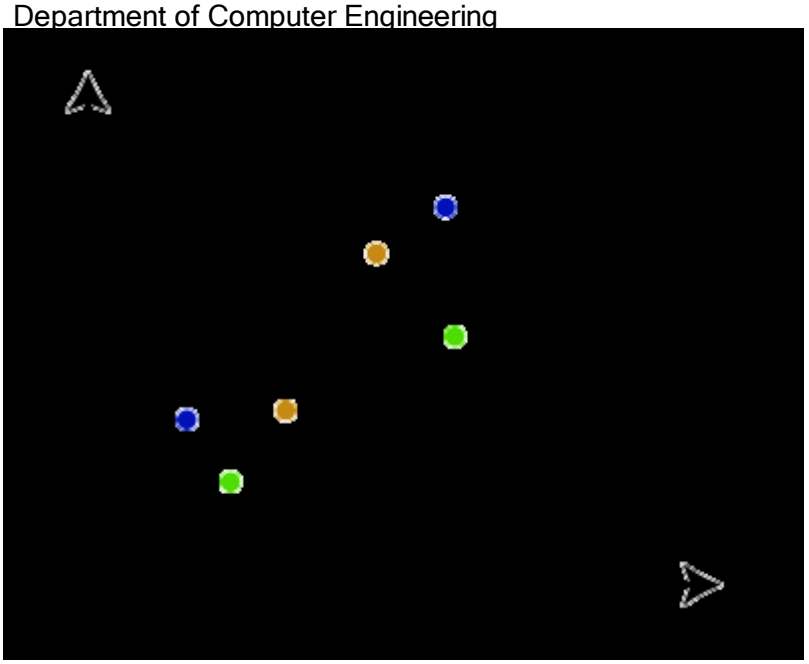
Agglomerative Hierarchical clustering:
The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the **bottom-up approach**. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

This hierarchy of clusters is represented in the form of the dendrogram.

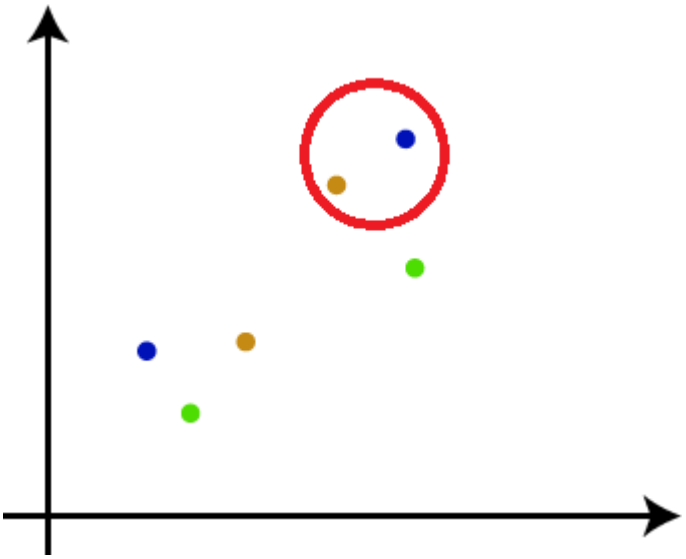
How the Agglomerative Hierarchical clustering Work?

The working of the AHC algorithm can be explained using the below steps:

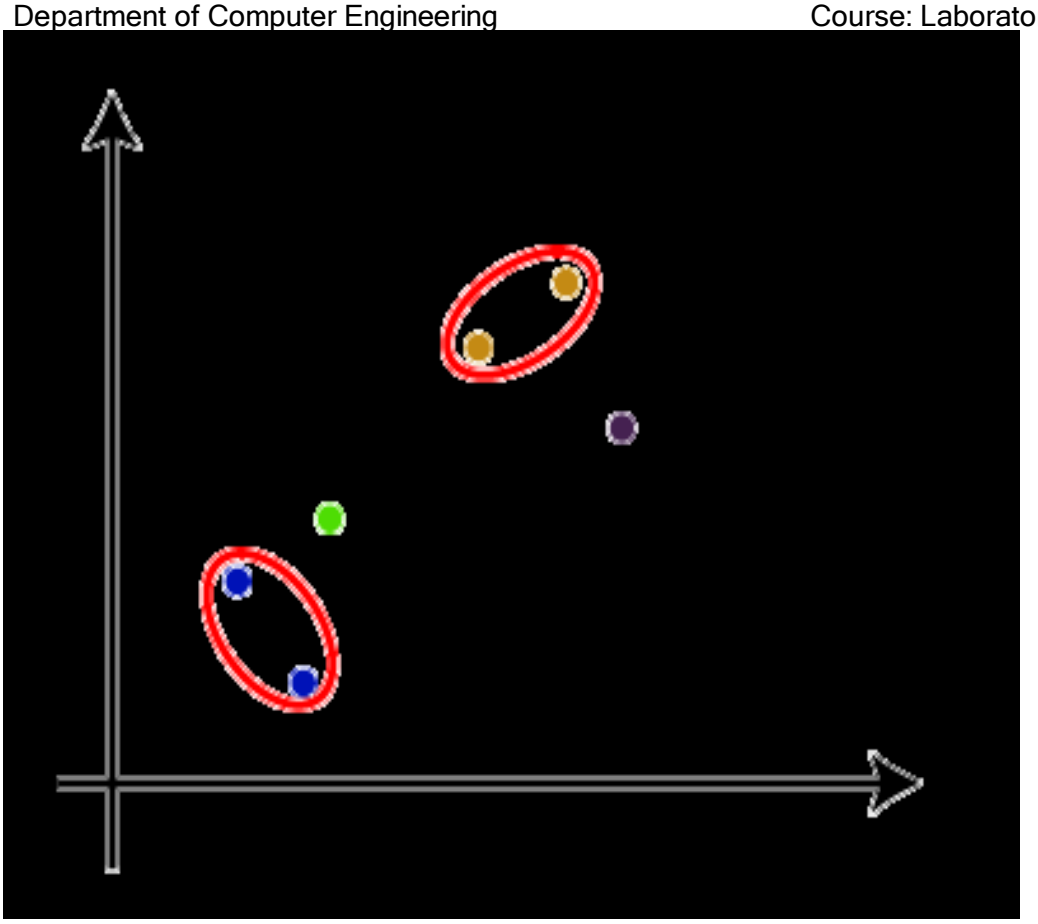
Step-1: Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.



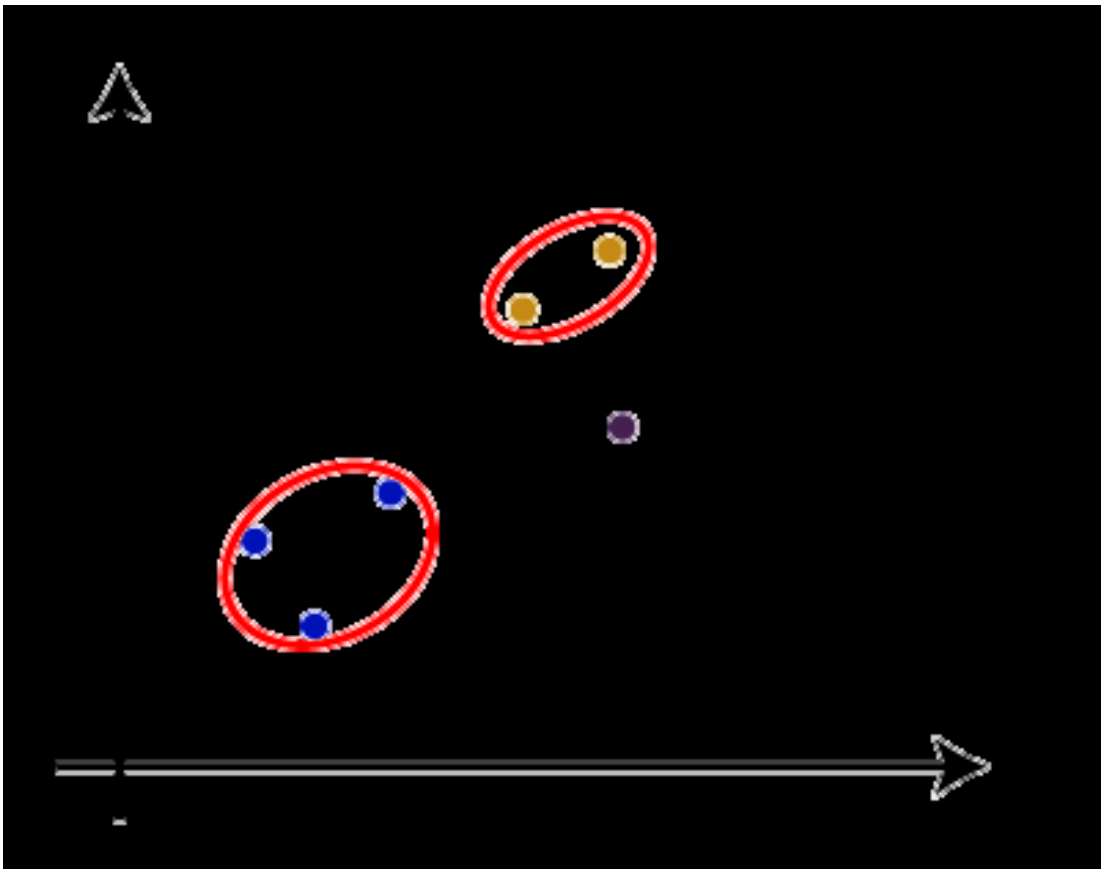
Step-2: Take two closest data points or clusters and merge them to form one cluster.
So, there will now be N-1 clusters.

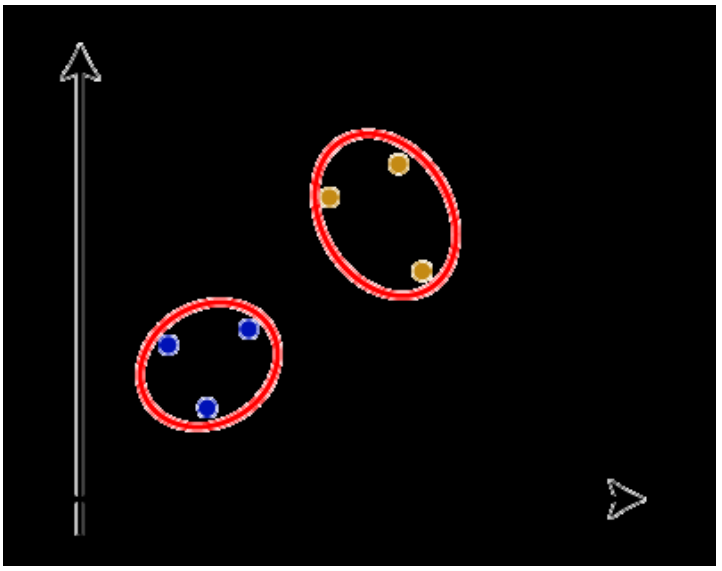


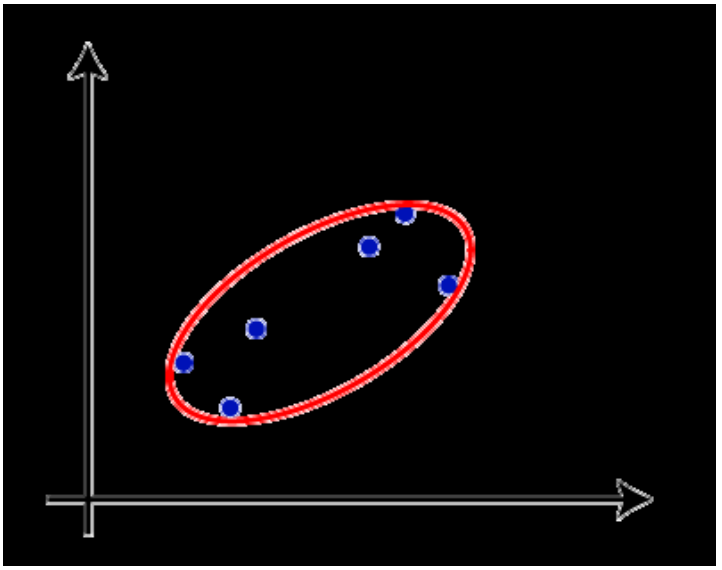
- **Step-3**: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.



Step-4: Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:





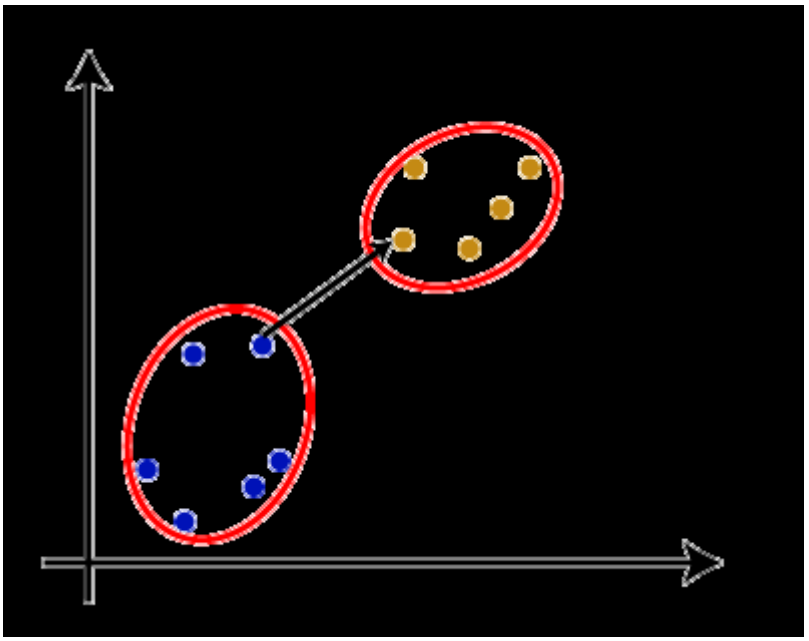


- **Step-5:** Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

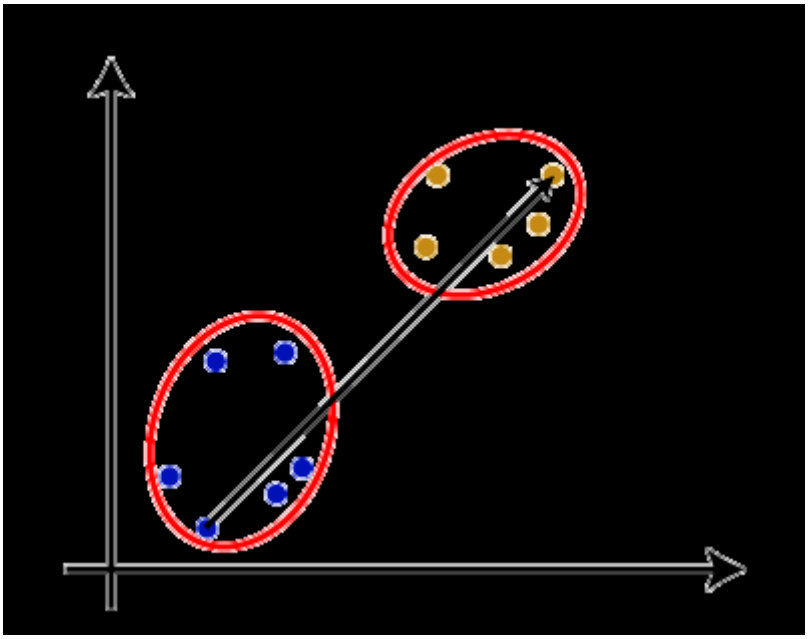
Measure for the distance between two clusters

As we have seen, the **closest distance** between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called **Linkage methods**. Some of the popular linkage methods are given below:

1. **Single Linkage:** It is the Shortest Distance between the closest points of the clusters. Consider the below image:

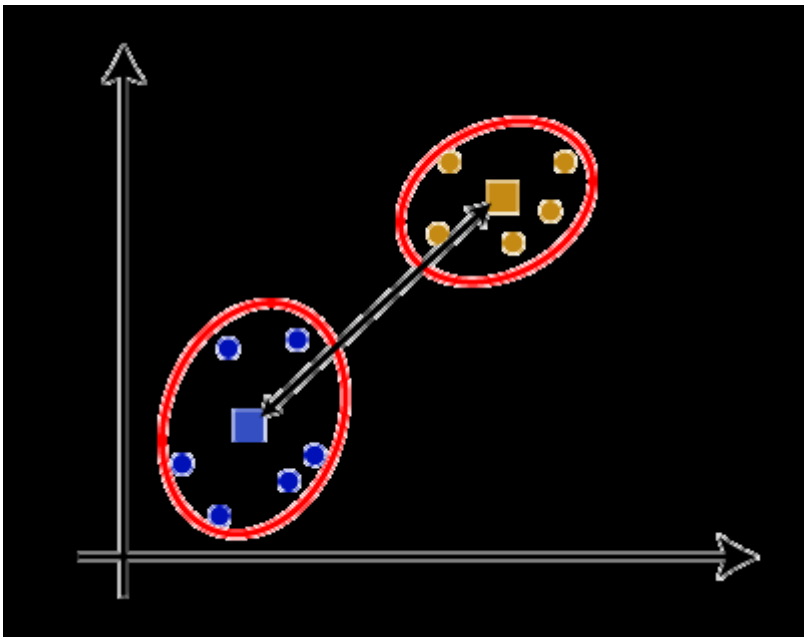


2. **Complete Linkage:** It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



3. **Average Linkage:** It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.

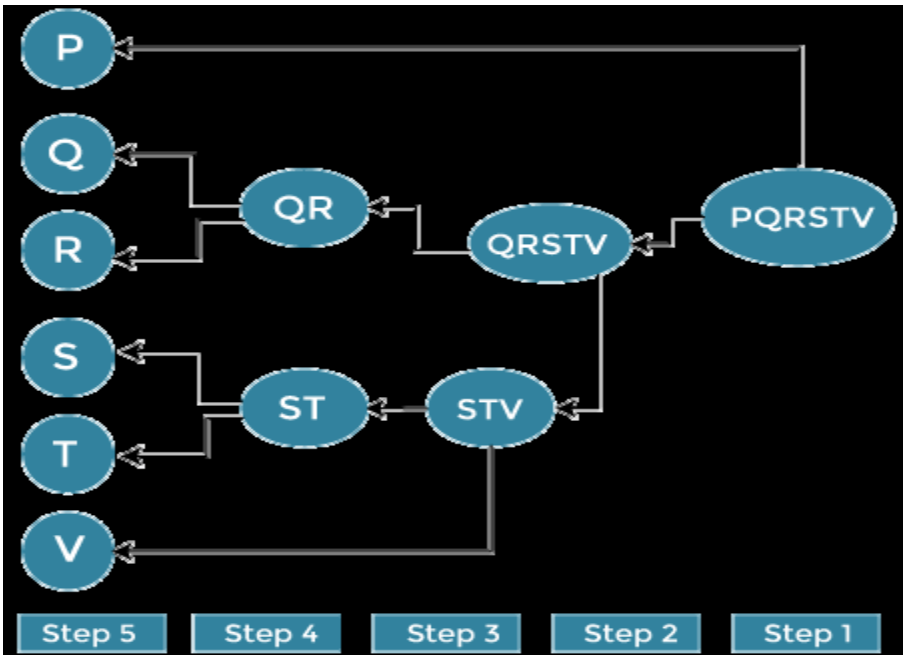
4. **Centroid Linkage:** It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:



From the above-given approaches, we can apply any of them according to the type of problem or business requirement.

Divise Hierarchical Clustering:

Divisive hierarchical clustering is exactly the opposite of Agglomerative Hierarchical clustering. In Divisive Hierarchical clustering, all the data points are considered an individual cluster, and in every iteration, the data points that are not similar are separated from the cluster. The separated data points are treated as an individual cluster. Finally, we are left with N clusters.



Principal Component Analysis:

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning

. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the **Principal Components**. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.*** It is a feature extraction technique, so it contains the important variables and drops the least important variable.

The PCA algorithm is based on some mathematical concepts such as:

- Variance and Covariance
- Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

- **Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- **Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors:** If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Steps for PCA Algorithm

1. **Getting the dataset**
Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.
2. **Representing data into a structure**

Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.

3. **Standardizing the data**

In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.

If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.

4. **Calculating the Covariance of Z**

To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.

5. **Calculating the Eigen Values and Eigen Vectors**

Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.

6. **Sorting the Eigen Vectors**

In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.

7. **Calculating the new features Or Principal Components**

Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other.

8. **Remove less or unimportant features from the new dataset.**

The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out

Elbow Method

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

WCSS= ∑Pi in Cluster1 distance(Pi C1)2 +∑Pi in Cluster2distance(Pi C2)2+∑Pi in CLuster3 distance(Pi C3)2

In the above formula of WCSS,

∑Pi in Cluster1 distance(Pi C1)2: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

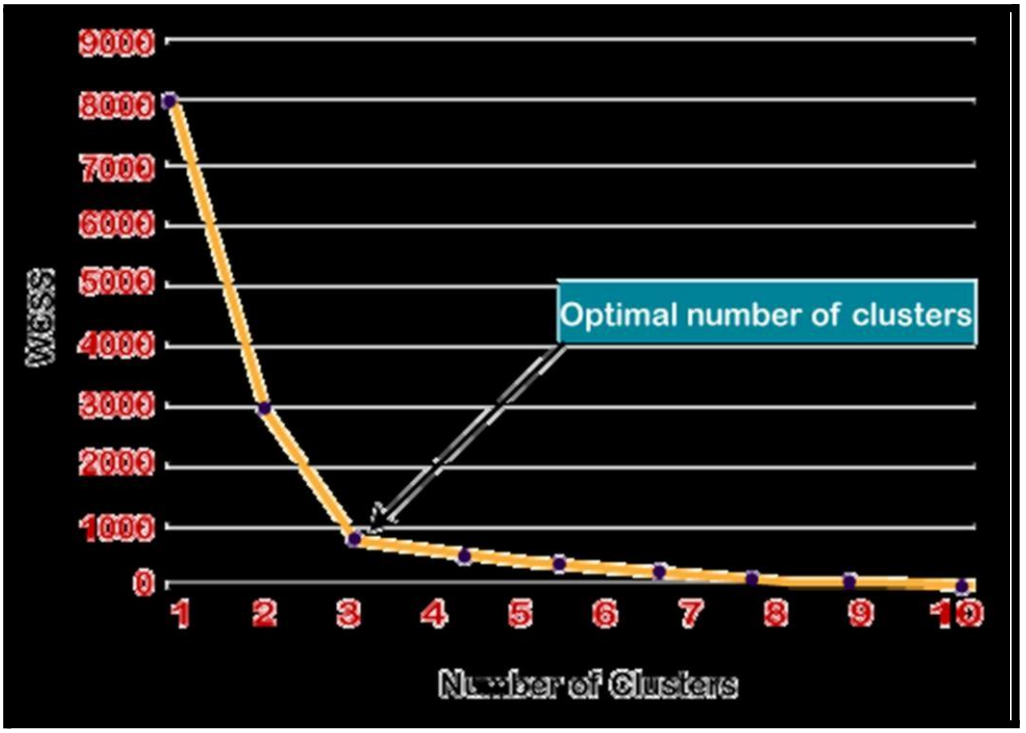
To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

To find the optimal value of clusters, the elbow method follows the below steps:

- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).

- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method. The graph for the elbow method looks like the below image:



Conclusion:

In this we have successfully implemented the above assignment.

Assignment Questions:

1) Explain K-Means Clustering?
2) Explain Hierarchical Clustering?
3) Explain Principal Component Analysis?
4) Explain Elbow Method?
5) Define Eigenvectors?

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

- -------- Contents for Theory:

1. **Introduction Blockchain**
2. **Cryptocurrency**
3. **Transaction Wallets**
4. **Ether transaction**
5. **Installation Process of Metamask**

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.

- **Peer-to-Peer Network**

  With the use of Blockchain, the interaction between two parties through a peer-to-peer model is easily accomplished without the requirement of any third party. Blockchain uses P2P protocol which allows all the network participants to hold an identical copy of transactions, enabling approval through a machine consensus. For example, if you wish to make any transaction from one part of the world to another, you can do that with blockchain all by yourself within a few seconds. Moreover, any interruptions or extra charges will not be deducted in the transfer.

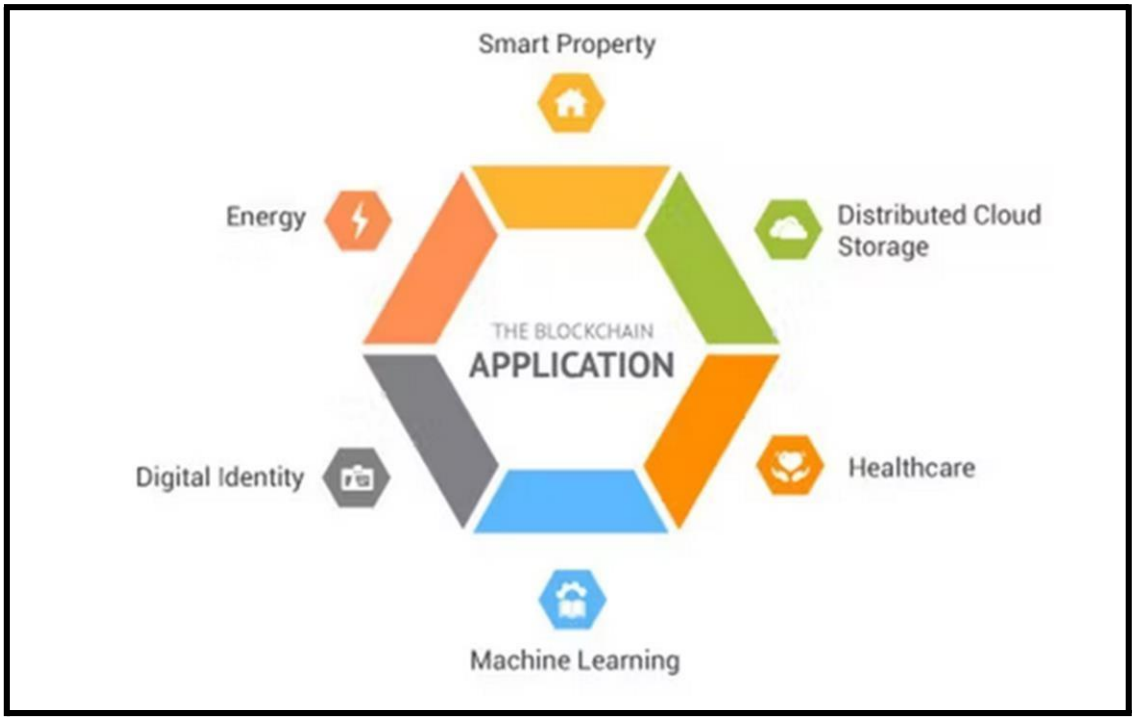
Immutable

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:

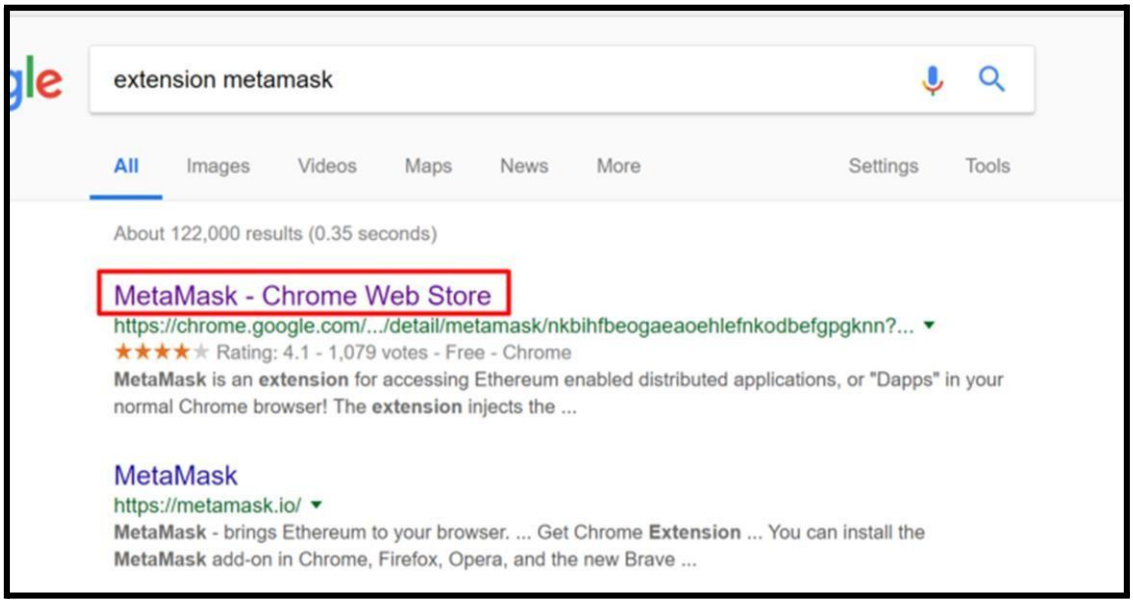
- **Time-saving:** No central Authority verification needed for settlements making the process faster and cheaper.
- **Cost-saving:** A Blockchain network reduces expenses in several ways. No need for third-party verification. Participants can share assets directly. Intermediaries are reduced. Transaction 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.

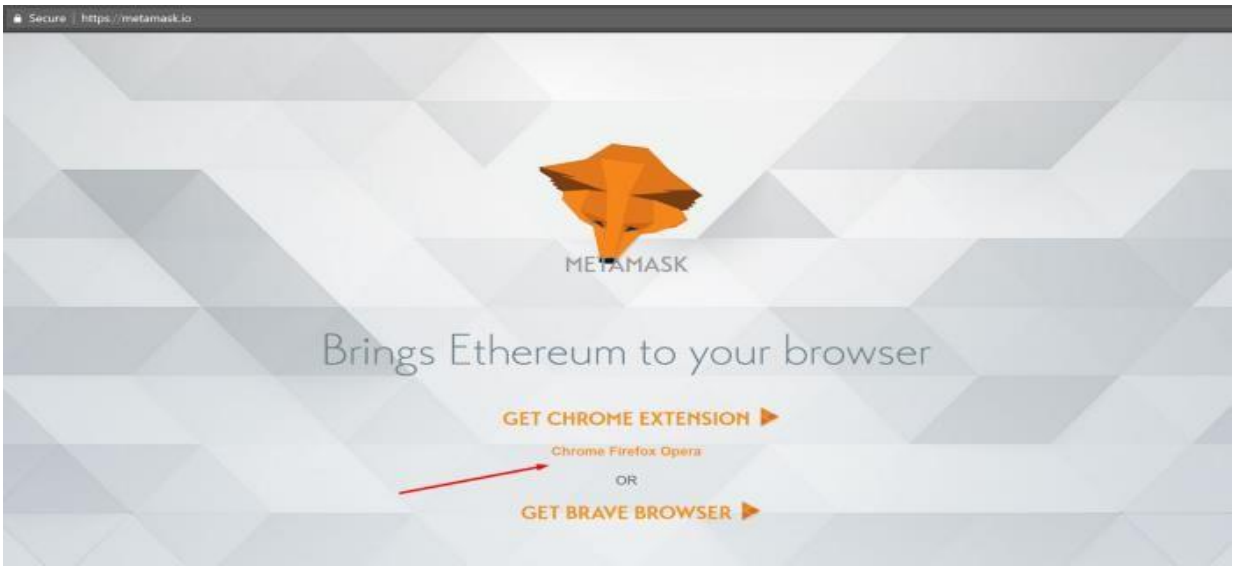
How to use MetaMask: A step by step guide

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.

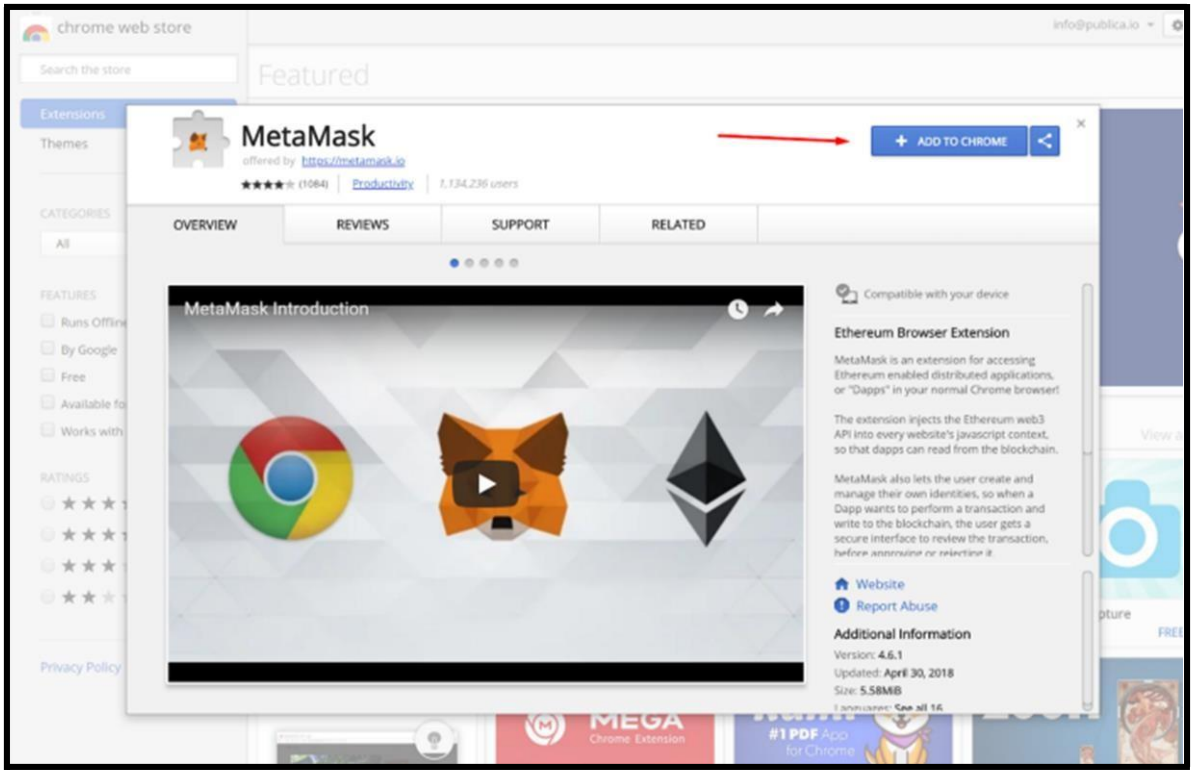
Step 1. Install MetaMask on your browser.

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 MetaMask** as a Google Chrome extension.
- Click **Add to Chrome**.
- Click **Add Extension**.



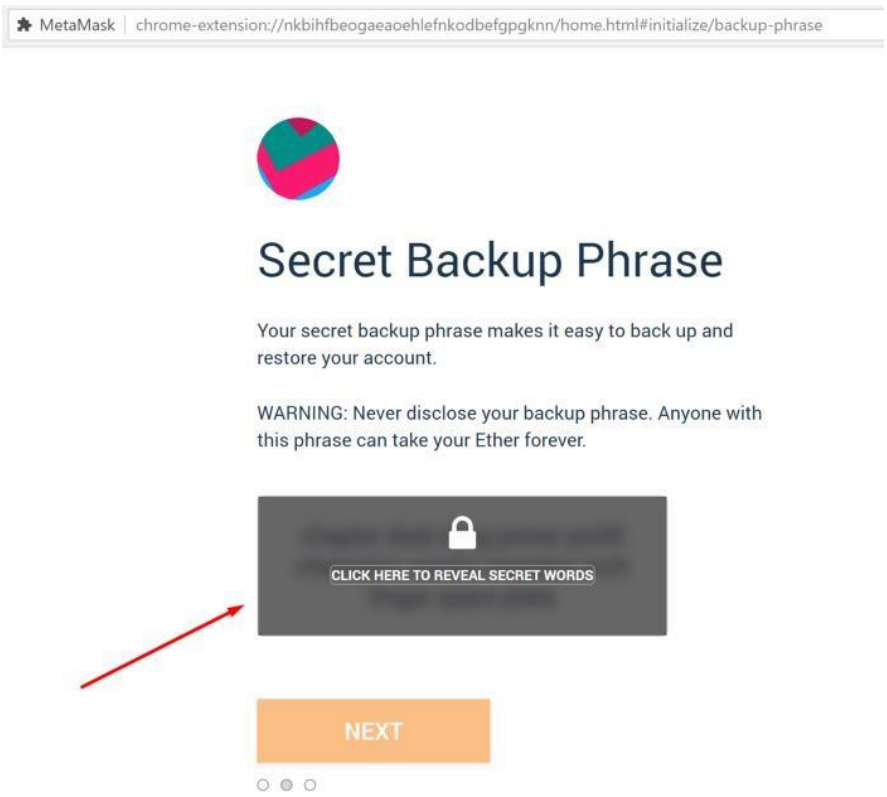
and it's as easy as that to install the extension on your browser, continue reading the next step to figure out how to create an account.

Step 2. Create an account.

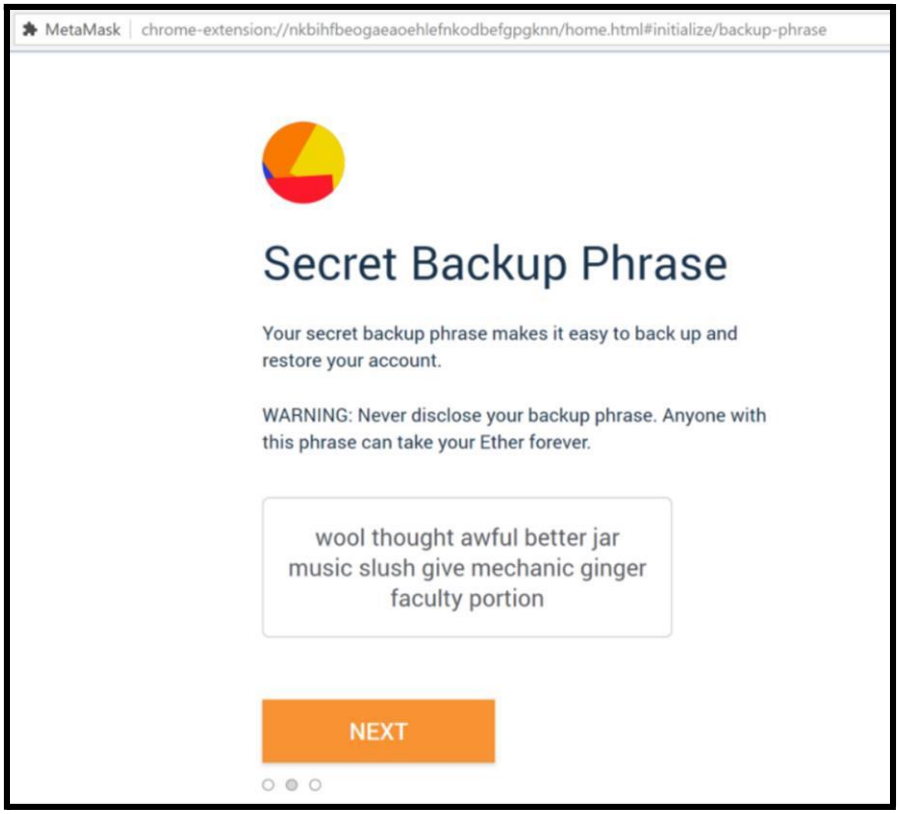
- Click on the extension icon in the upper right corner to open MetaMask.
- To install the latest version and be up to date, **click Try it now**.
- **Click Continue**.
- You will be prompted to create a new password. **Click Create**.

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 reall mportant and usually not a good idea to store digitally, so take your time and write it do


MetaMask
chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#initialize/backup-phrase
Secret Backup Phrase
Your secret backup phrase makes it easy to back up and restore your account.
WARNING: Never disclose your backup phrase. Anyone with this phrase can take your Ether forever.
CLICK HERE TO REVEAL SECRET WORDS
NEXT

wn



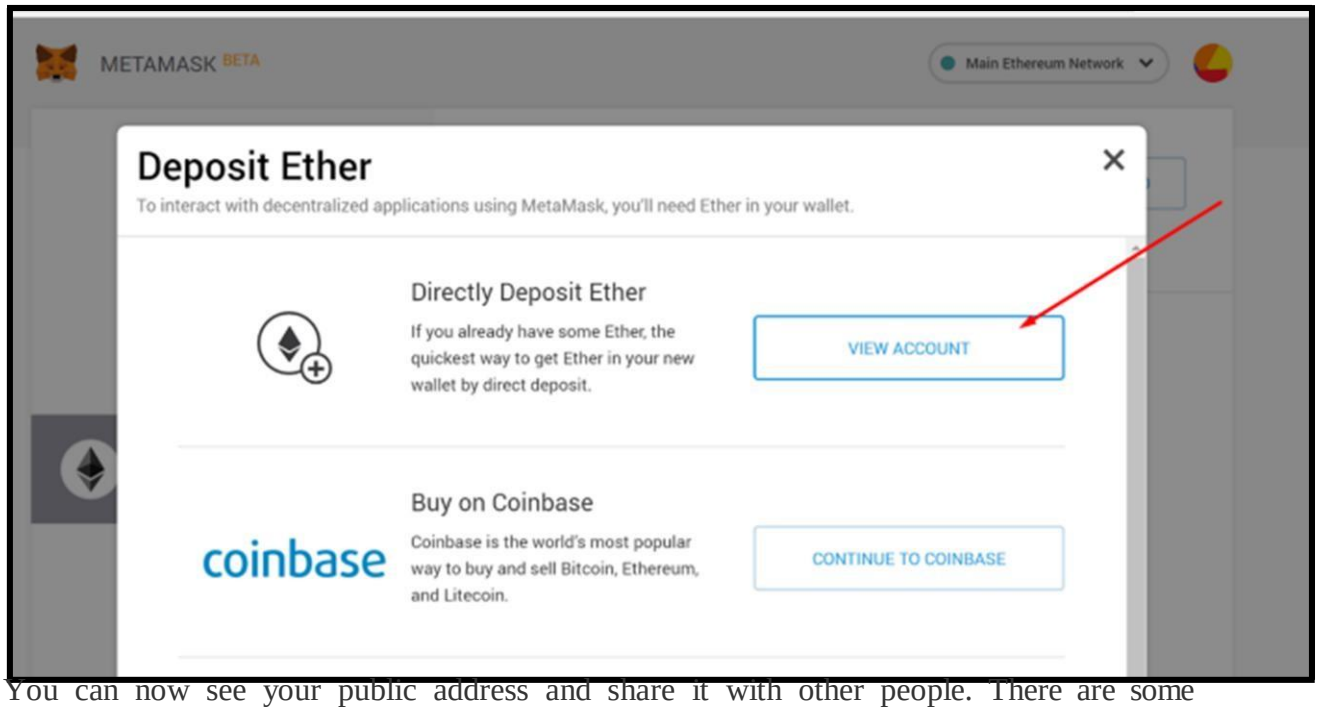
- Verify your secret phrase by selecting the previously generated phrase in order. **Click Confirm**.

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.

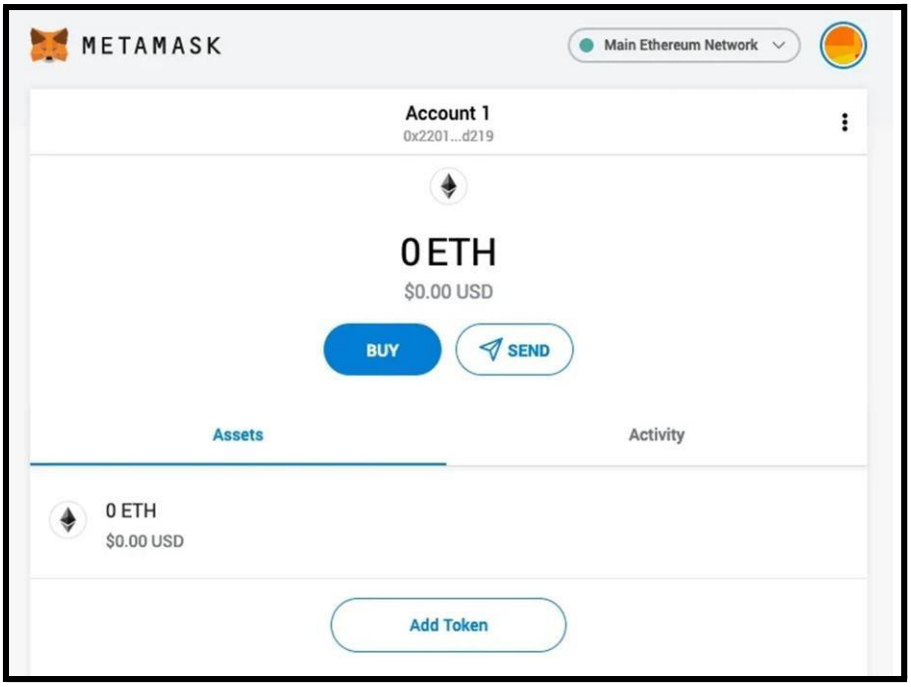
Step 3. Depositing funds.

- Click on **View Account**.

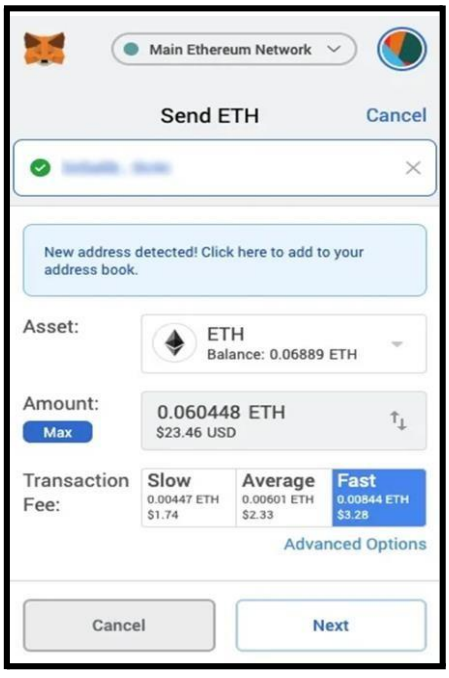


You can now see your public address and share it with other people. There are some methods to buy coins offered by MetaMask, but you can do it differently as well; you just need your address.

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 Blockchain and metamat 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.**

Reference link

- https://hackernoon.com/blockchain-technology-explained-introduction-meaning-and-applications-edbd6759a2b2
- https://levelup.gitconnected.com/how-to-use-metamask-a-step-by-step-guide-f380a3943fb1
- https://decrypt.co/resources/metamask

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 cryptocurrencies and learn how transaction done by using different digital currency

Prerequisite:

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

Contents for Theory:

1. **Cryptocurrency**
2. **Transaction Wallets**
3. **Ether transaction**

Introduction to Cryptocurrency

- Cryptocurrency 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, cryptocurrency payments exist purely as digital entries to an online database describing specific transactions. When you transfer cryptocurrency funds, the transactions are recorded in a public ledger. Cryptocurrency is stored in digital wallets.
- Cryptocurrency received its name because it uses encryption to verify transactions. This means advanced coding is involved in storing and transmitting cryptocurrency data between wallets and to public ledgers. The aim of encryption is to provide security and safety.
- The first cryptocurrency was Bitcoin, which was founded in 2009 and remains the best known today. Much of the interest in cryptocurrencies is to trade for profit, with speculators at times driving prices skyward.

How does cryptocurrency work?

- Cryptocurrencies run on a distributed public ledger called blockchain, a record of all transactions updated and held by currency holders.
- Units of cryptocurrency 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 cryptocurrency, 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 Bitcoin has been around since 2009, cryptocurrencies 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 cryptocurrency

- Once you have purchased cryptocurrency, you need to store it safely to protect it from hacks or theft. Usually, cryptocurrency is stored in crypto wallets, which are physical devices or online software used to store the private keys to your cryptocurrencies 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 Cryptocurrency and learn how transactions are done using digital currency

Assignment Question

1. **What is Bitcoin?**
2. **What Are the biggest Four common cryptocurrency scams**
3. **Explain How safe are money e-transfers?**
4. **What is cryptojacking and how does it work?**

Reference link

- https://www.kaspersky.com/resource-center/definitions/what-is-cryptocurrency

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 about smart contract for banking application and able to perform some operation like Deposit money Withdraw Money Show balance

Prerequisite:

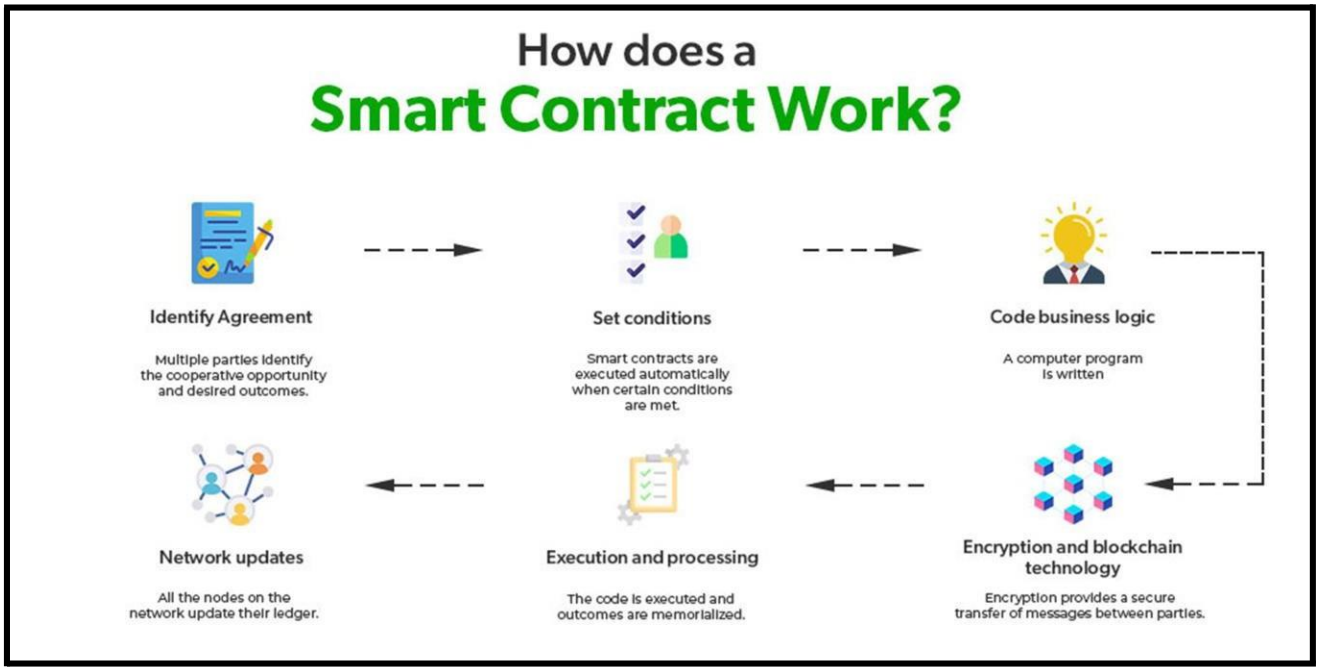
1. Basic knowledge of smart contract
2. Remix IDE
3. Basic knowledge of solidity

Contents for Theory:

1. **Smart contract**
2. **Remix IDE**
3. **Solidity Basics**
4. **Ether transaction**

Introduction to Smart Contract

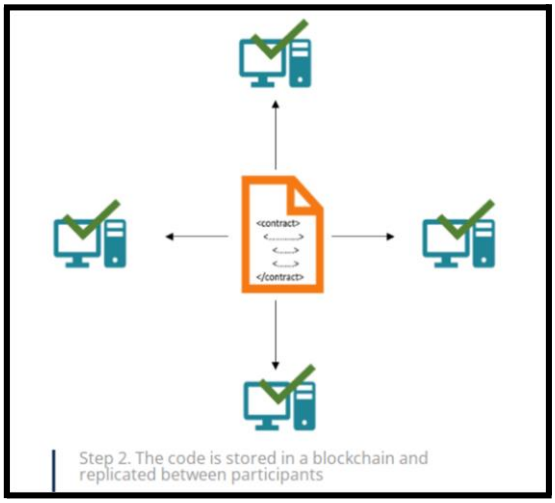
- Smart contracts are self-executing lines of code with the terms of an agreement between buyer and seller automatically verified and executed via a computer network.
- Nick Szabo, an American computer scientist who invented a virtual currency called "Bit Gold" in 1998,1 defined smart contracts as computerized transaction protocols that execute terms of a contract.2
- Smart contracts deployed to blockchains render transactions traceable, transparent, and irreversible.



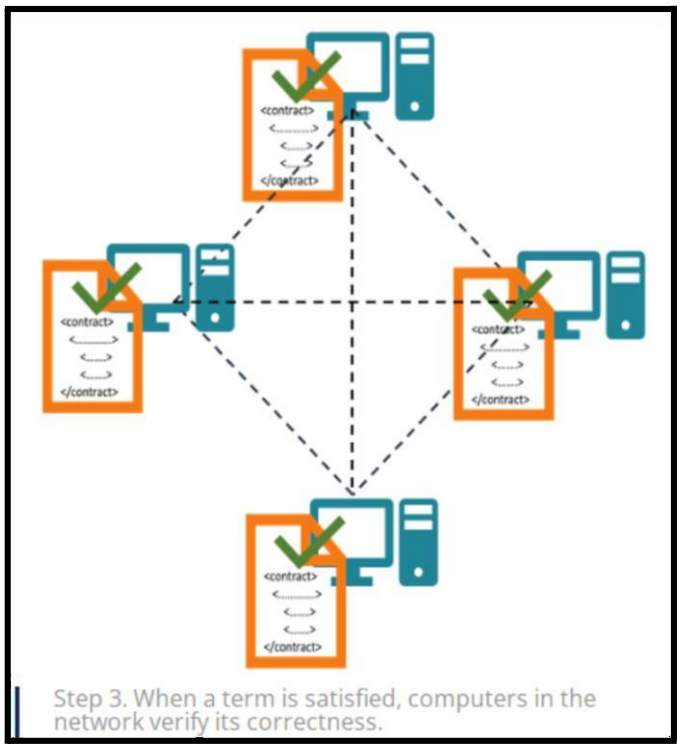
How does Smart Contract work?

- First, the contractual parties should determine the terms of the contract. After the contractual terms are finalized, they are translated into programming code. Basically, the code represents a number of different conditional statements that describe the possible scenarios of a future transaction.

- When the code is created, it is stored in the blockchain network and is replicated among the participants in the blockchain.



- Then, the code is run and executed by all computers in the network. If a term of the contract is satisfied and it is verified by all participants of the blockchain network, then the relevant transaction is executed.

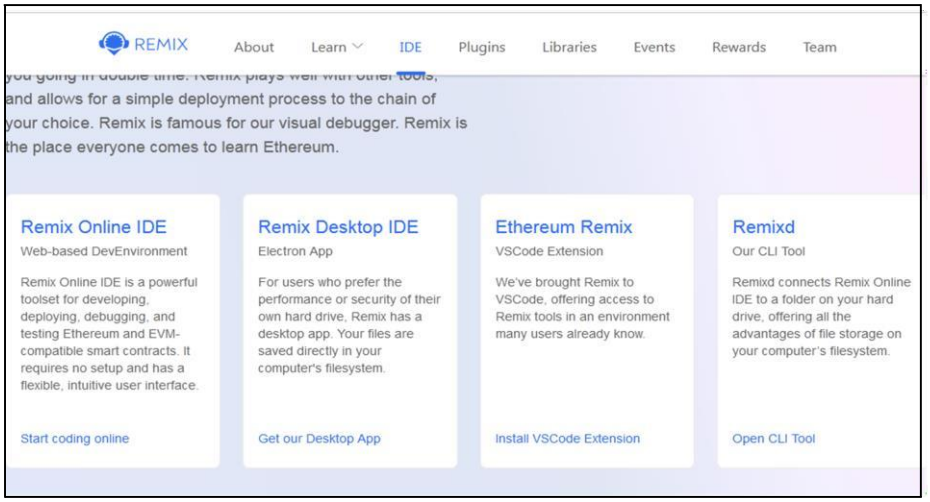


Remix IDE : Installation steps

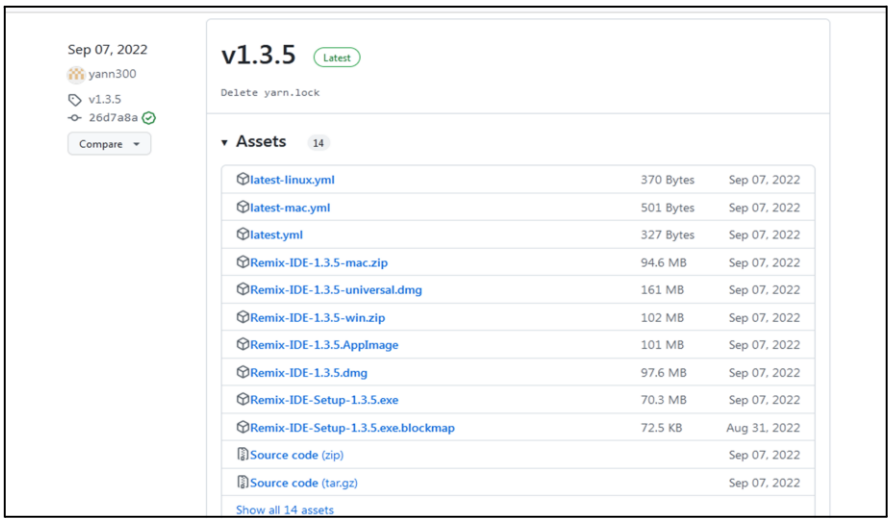
- First and foremost, head on over to the release page of the official Remix Desktop repository

and grab the binary suited for your host system. This will be the .exe file for Windows users, the .dmg for macOS and the .deb for Debian-derivative GNU/Linux systems. For Ubuntu and other AppImage setups, the .AppImage will be what you are looking for.

- Go to https://remix-project.org/
- Go to IDE option .Select Desktop IDE



- Select The file As per your choice



- Install Executable file,by double clicking on file(Applicable to .exe file Windows OS)

What is Solidity?

- Solidity is a rather simple language deliberately created for a simplistic approach to tackle real world solutions. Gavin Wood initially proposed it in August of 2014. Several developers

of the Ethereum chain such as Christian Reitwiessner, Alex Beregszaszi, Liana Husikyan, Yoichi Hirai and many more contributed to creating the language. The Solidity language can be executed on the Ethereum platform, that is a primary Virtual Machine implementing the blockchain network to develop decentralized public ledgers to create smart contract systems.

Solidity Basics

- To get started with the language and learn the basics let's dive into coding. We will begin by understanding the syntax and general data types, along with the variable data types. Solidity supports the generic value types, namely:
- Booleans: Returns value as either true or false. The logical operators returning Boolean data types are as follows:
- ! Logical negation
- && logical conjunction, "and"
- || logical disjunction, "or"
- == equality
- != inequality

Integers: Solidity supports int/unit for both signed and unsigned integers respectively. These storage allocations can be of various sizes. Keywords such as uint8 and uint256 can be used to allocate a storage size of 8 bits to 256 bits respectively. By default, the allocation is 256 bits. That is, uint and int can be used in place of uint256 and int256. The operators compatible with integer data types are:

- Comparisons: <=, <, ==, !=, >=, >. These are used to evaluate to bool.
- Bit operators: &, |, ^ bitwise exclusive 'or', ~ bitwise negation, "not".
- Arithmetic operators: +, -, unary -, unary +, *, /, % remainder, ** exponentiation, << left shift, >> right shift.

The EVM returns a Runtime Exception when the modulus operator is applied to the zero of a "divide by zero" operation.

Address: An address can hold a 20 byte value that is equivalent to the size of an Ethereum address. These address types are backed up with members that serve as the contract base.

String Literals: String literals can be represented using either single or double quotes (for example, "foo" or 'bar'). Unlike in the C language, string literals in Solidity do imply trailing value zeroes. For instance, “bar” will represent a three byte element instead of four. Similarly, in the case of integer literals, the literals are convertible inherently using the corresponding fit, that is, byte or string.

Modifier: In a smart contract, modifiers are used to ensure the coherence of the conditions defined before executing the code.

Solidity provides basic arrays, enums, operators, and hash values to create a data structure known as **“mappings**.” These mappings are used to return values associated with a given storage location. An Array is a contiguous memory allocation of a size defined by the programmer where if the size is initialized as K, and the type of element is instantiated as T, the array can be written as T[k].

Arrays can also be dynamically instantiated using the notation uint[][6]. Here the notation initializes a dynamic array with six contiguous memory allocations. Similarly, a two dimensional array can be initialized as arr[2][4], where the two indices point towards the dimensions of the matrix.

We will begin our programming venture with a simple structure of a contract. Consider the following code:

```
pragma solidity^0.4.0;
contract StorageBasic {
  uint storedValue;
  function set(uint var) {
     storedValue= var;
  }
  function get() constant returns (uint) {
   return storedValue;
  }
}
```

- The word “**Pragma**” refers to the instructions given to a compiler to sequentially execute the source code.
- Solidity is statically typed language. Therefore, each variable type irrespective of their scope

can be instantiated at compile time. These elementary types can be further combined to create

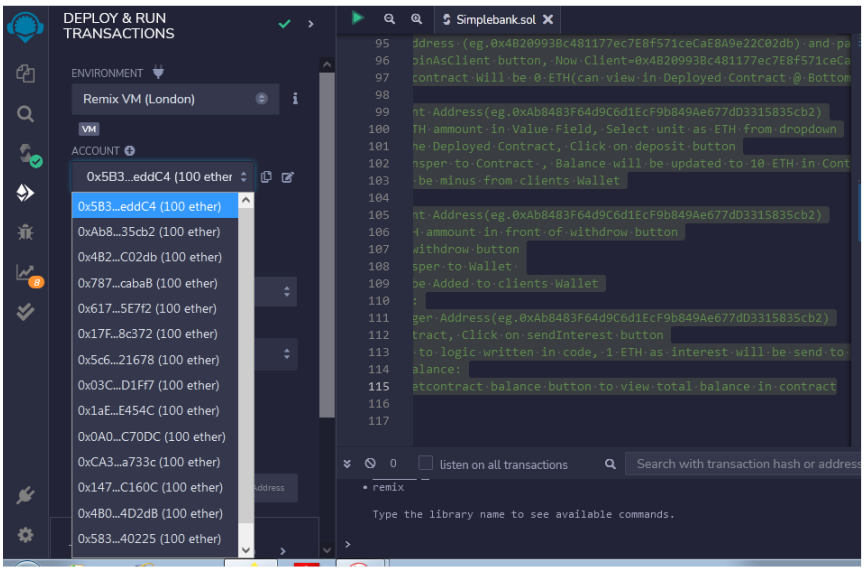
complex data types. These complex data types then synchronize with each other according to their respective preferences.

Steps to Run Banking Application

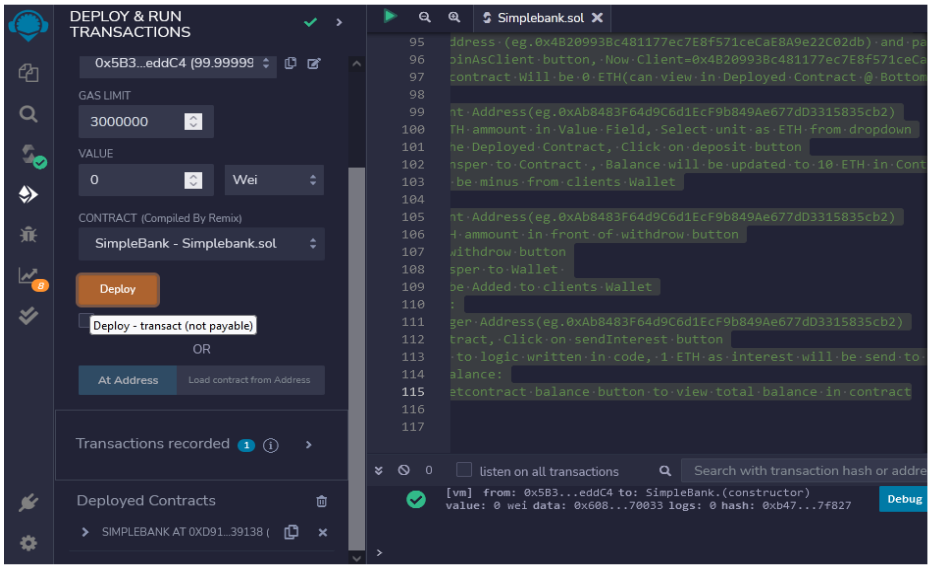
1.//Output steps

//Initially All Account has 100 fake ether

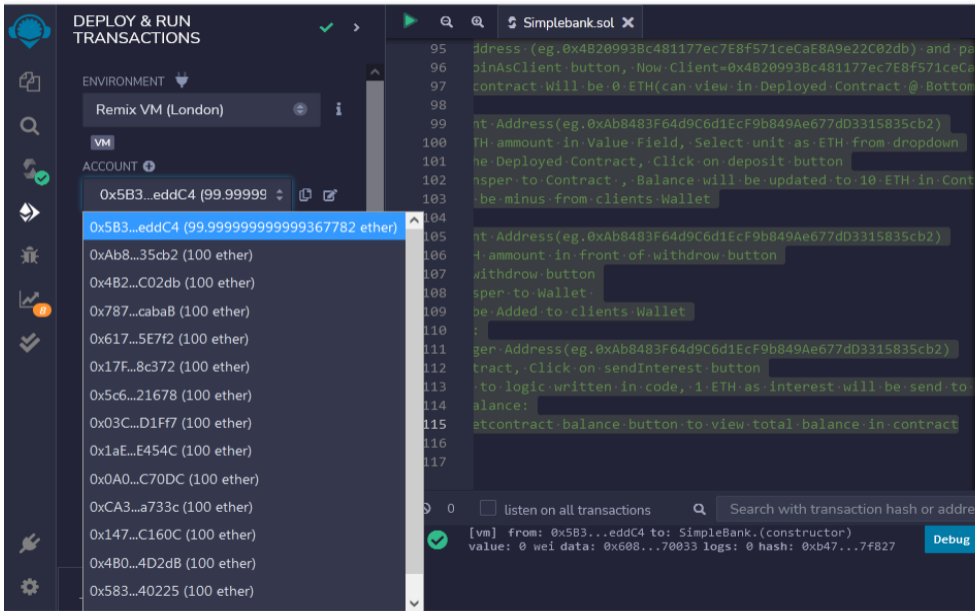
//Step 1: Select first Address (eg.0x5B38Da6a701c568545dCfcB03FcB875f56beddC4)



//Step 2: Click on Deploy button(Contract Created,Can view under Deployed Contract)



//After deploying contract 100 ETH turns to 99.99999. ... ETH

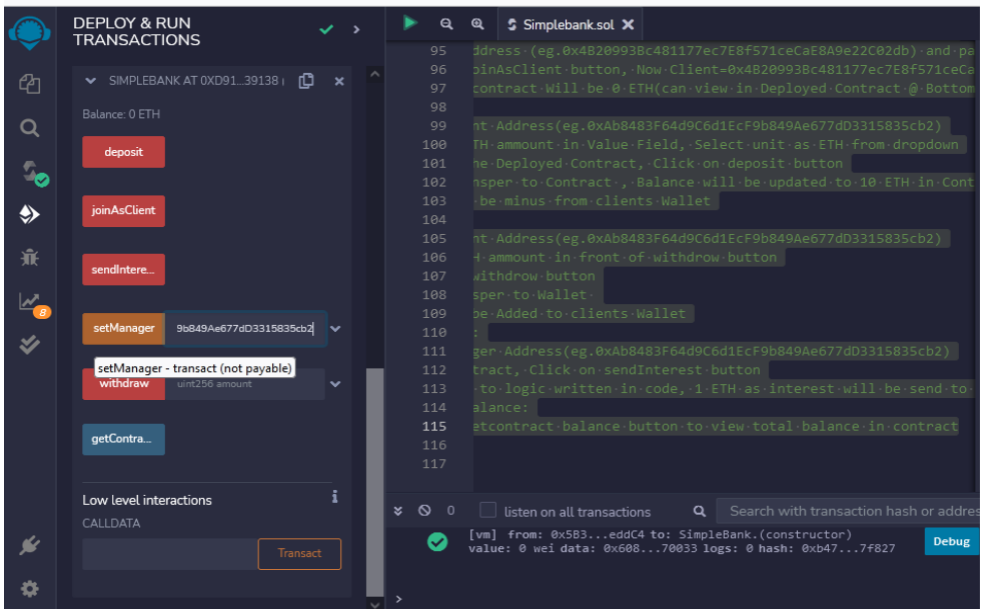


//Step 3: Set Manager: Follow Following instructions

// i.Select Onother Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)

// ii.Copy this address (eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2) and paste it in contract, infront of set Manager button

// iii. click on set manager button, Now Manager=0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2



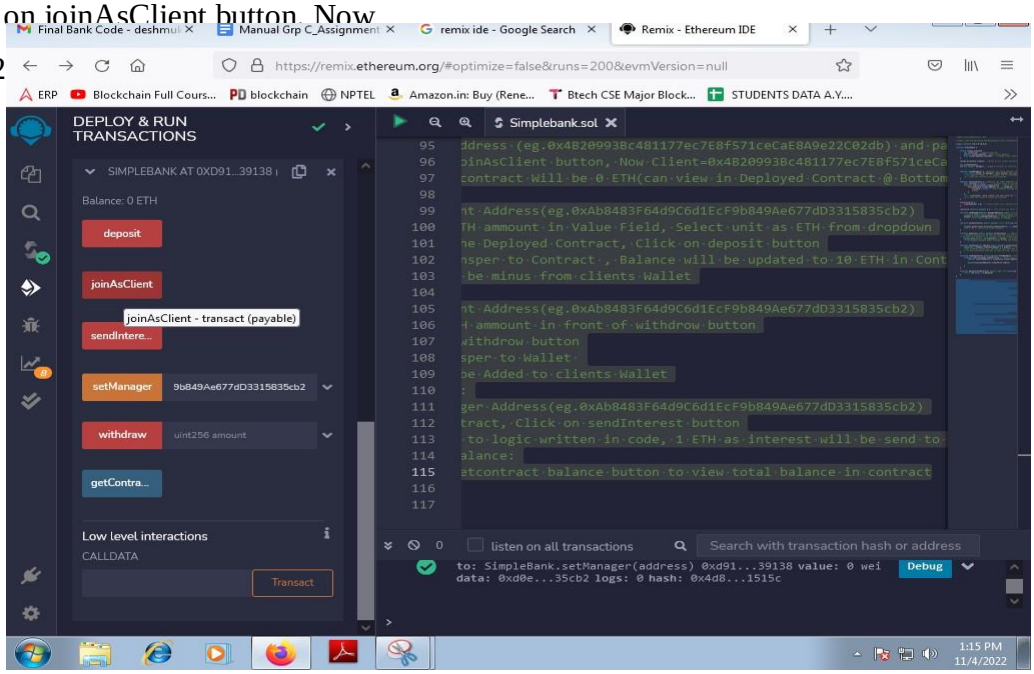
//Step 4: join as Client: Follow Following instructions

// i.Select Onother Address(eg.0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db)

// ii.Copy this address (eg.0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db) and paste it in

contract, infront of joinAsClient button

// iii.click on joinAsClient button. Now Client=0x4B2



//Initially Balance in contract Will be 0 ETH(can view in Deployed Contract @ Bottom)

//Step 5: Deposit:

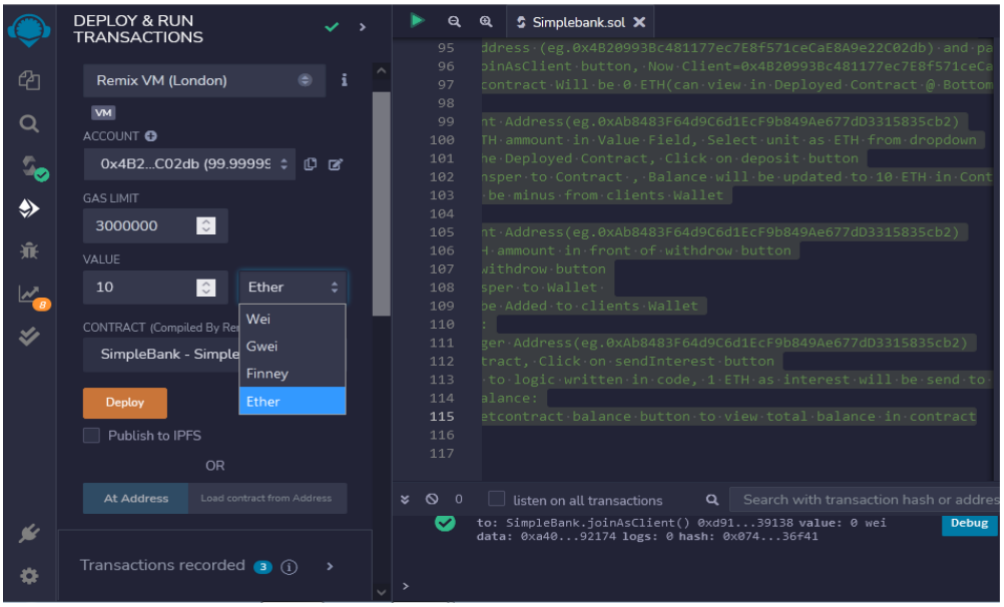
// i.Select Client Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)

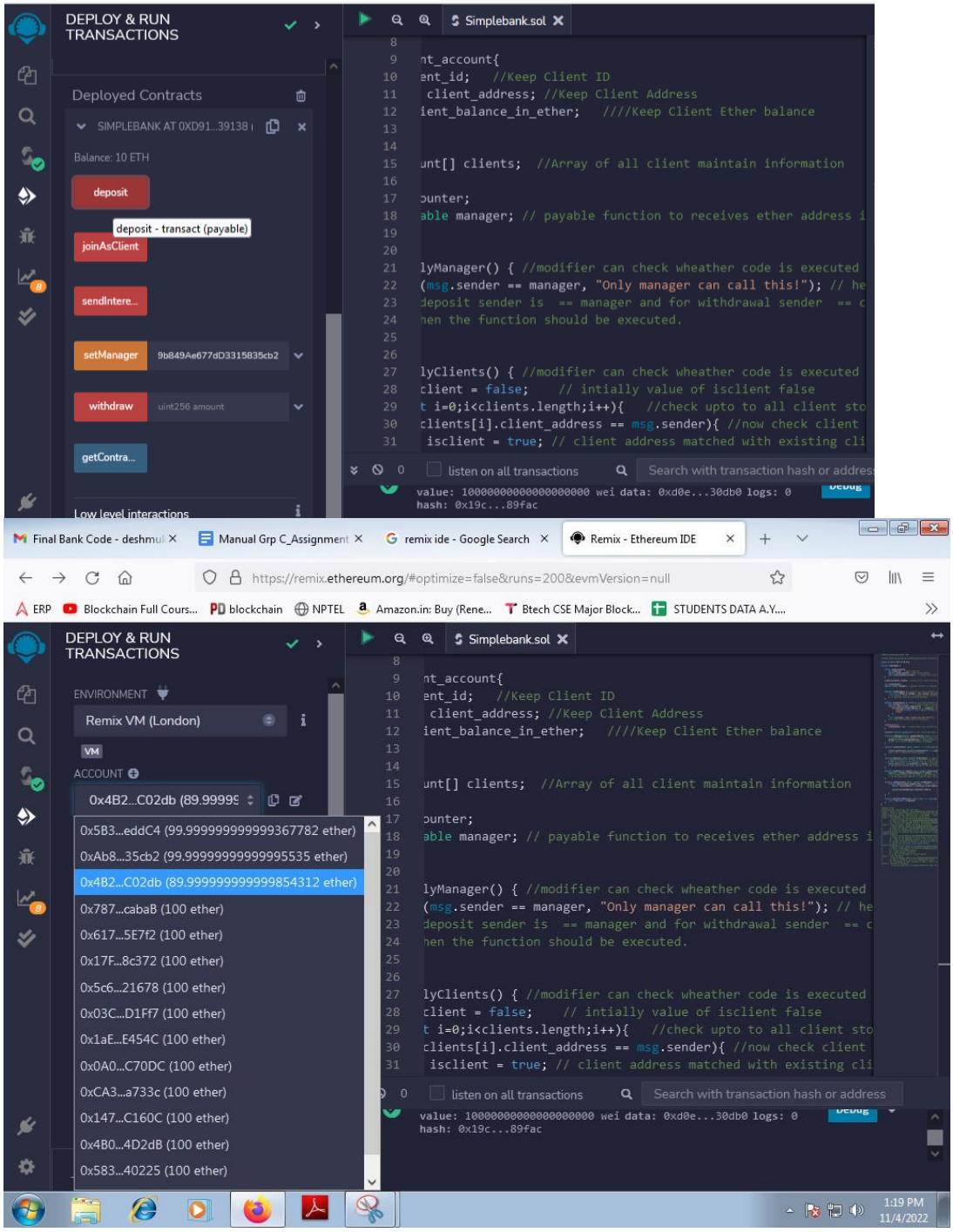
// ii. Enter 10 ETH ammount in Value Field, Select unit as ETH from dropdown

// iii. Come to the Deployed Contract, Click on deposit button

// iv. 10 ETH transper to Contract , Balance will be updated to 10 ETH in Contract

// V. 10 ETH will be minus from clients Wallet





//Step 6: Withdraw:

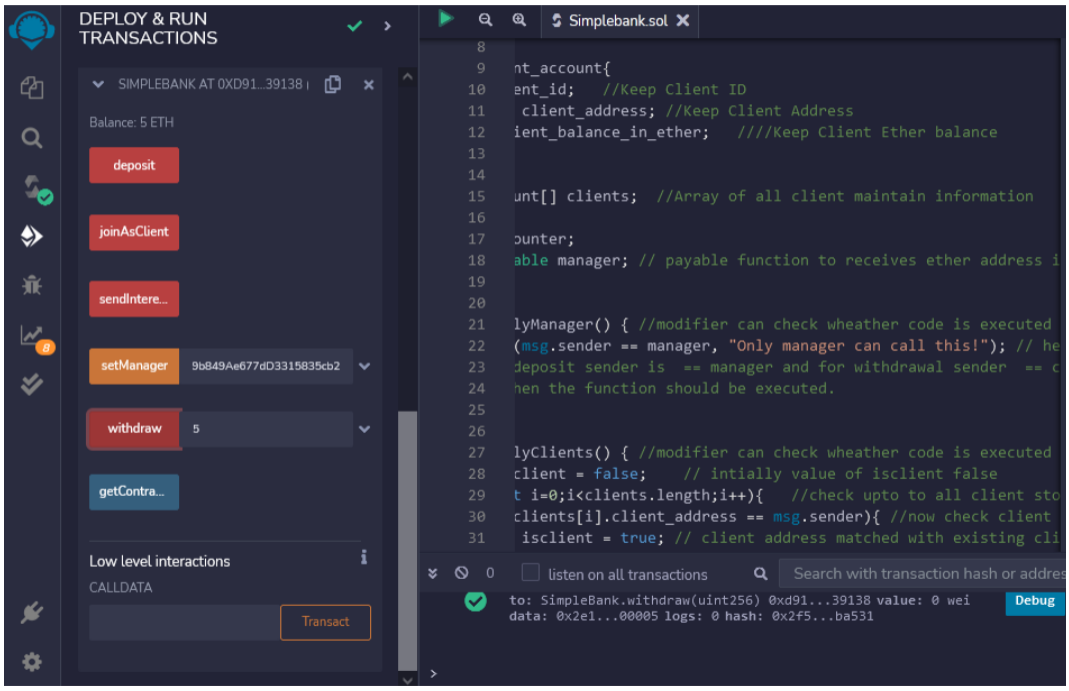
// i.Select Client Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)

// ii. Enter 5 ETH ammount in front of withdrow button

// iii. Click on withdrow button

// iv. 5 ETH transper to Wallet

// V. 5 ETH will be Added to clients Wallet

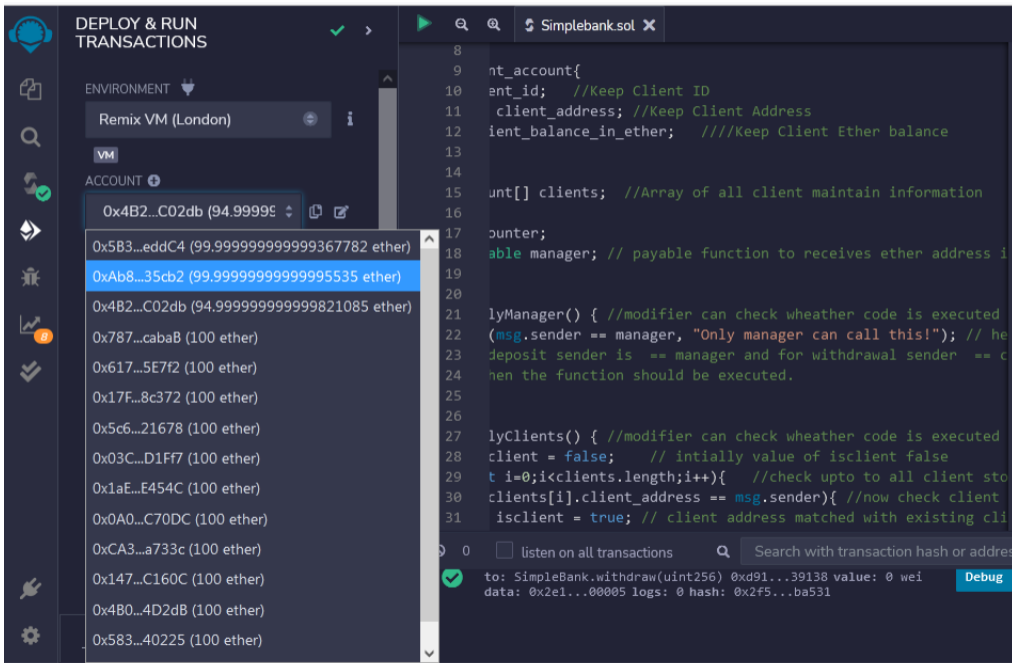


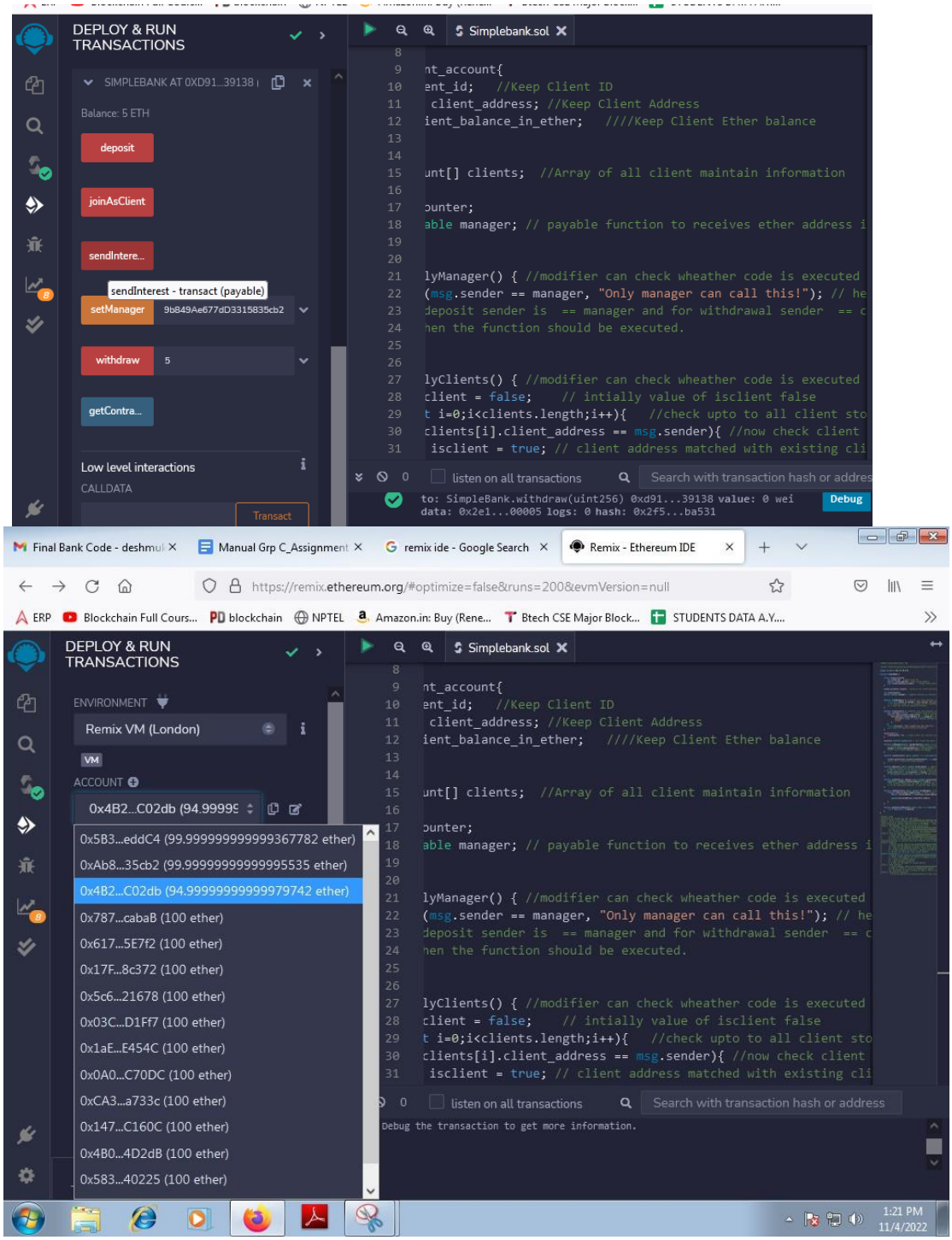
//Step 7: Send Interest:

// i.Select Manager Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)

// ii.Come to contract, Click on sendInterest button

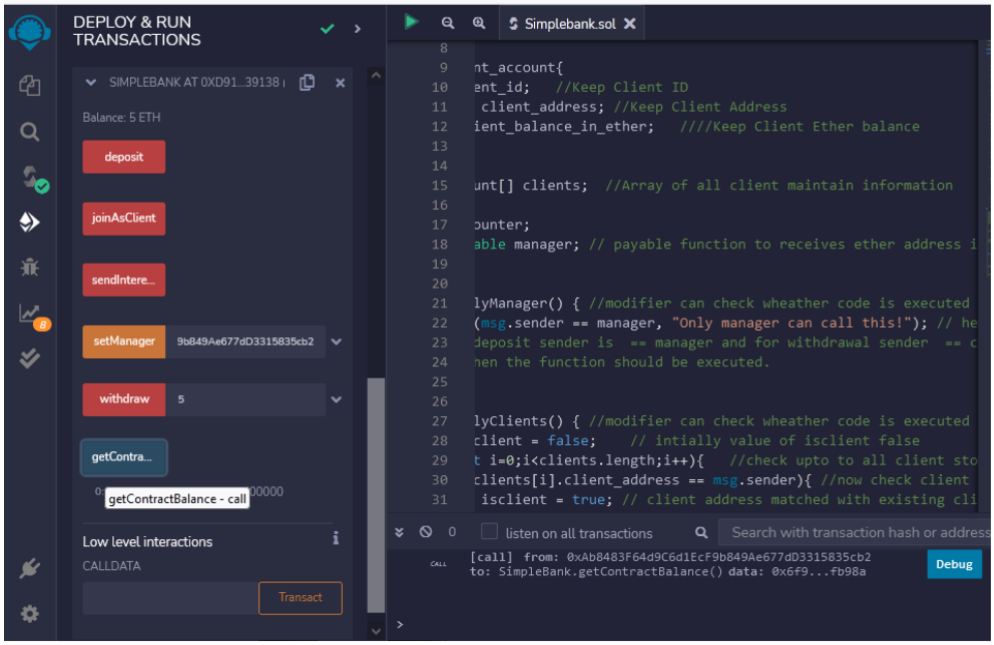
// iii. According to logic written in code, 1 ETH as interest will be send to Client Wallet





//Step 8: getContract Balance:

// i.Click on get contract balance button to view total balance in contract



Assignment Question

1. **What is Solidity?**
2. **What are the main differences between Solidity and other programming languages like Python, Java, or C++?**
3. **What is EVM bytecode?**
4. **What are the differences between Ethereum and blockchain and bitcoin?**

Reference link

- https://www.simplilearn.com/solidity-interview-questions-article

Code :

//SPDX-License-Identifier: MIT

```
//https://betterprogramming.pub/developing-a-smart-contract-by-using-re mix-ide-81ff6f44ba2f

pragma solidity >=0.7.0 <0.9.0;

contract SimpleBank {

        struct client_account{
          int client_id;            //Keep Client ID
          address client_address; //Keep Client Address
          uint client_balance_in_ether;              ////Keep Client Ether balance
      }

      client_account[] clients;            //Array of all client maintain information

      int clientCounter;
      address payable manager; // payable function to receives ether address is datatype it is 20 byte hash address public key


      modifier onlyManager() { //modifier can check wheather code is executed accouding to condition for manager side
            require(msg.sender == manager, "Only manager can call this!");
// here sender is manager in this case
            // for deposit sender is            == manager and for withdrawal sender == client
            _; // when the function should be executed.
      }

      modifier onlyClients() { //modifier can check wheather code is executed accouding to condition for client side
            bool isclient = false;            // intially value of isclient false for(uint
            i=0;i<clients.length;i++){          //check upto to all client store in array
                  if(clients[i].client_address == msg.sender){ //now check client address matched with sender only that client intiate transaction
```

```
                    isclient = true; // client address matched with existing client address
        in bank database isclient value updated true.
                    break;
                }
```

```
            }
            require(isclient, "Only clients can call this!"); // isclient true here so allowed call
the transaction.
            _; // when the function should be executed.
        }

        constructor() {
            clientCounter = 0;            // those client join contract assign there
ID intially it set 0
        }

        receive() external payable { } // this allows the smart contract to receive ether

        function setManager(address managerAddress) public returns(string memory){
//setManager method will be used to set the manager address to variables
            // string memory store address of manager account instead of store data
            manager = payable(managerAddress);// managerAddress is consumed as a
parameter and cast as payable to provide sending ether.
            return ""; // return payable address of manager
        }

        function joinAsClient() public payable returns(string memory){
//joinAsClient method will be used to make sure the client joins the contract.

            clients.push(client_account(clientCounter++, msg.sender,
address(msg.sender).balance)); // push() array method to add items into a storage array.
            return ""; // return all client details
        }

        function deposit() public payable onlyClients{                    // deposit ==
client to contract by onlyclient
            //deposit method will be used to send ETH from the client account to the
contract.
        // We want this method to be callable only by clients who've joined the contract, so the
onlyClient modifier is used for this restriction.
            payable(address(this)).transfer(msg.value); //transfer methods belongs to the
contract, and it's dedicated to sending an indicated amount of ETH between addresses.
```

```
        // The payable keyword makes receipt of the ETH transfer possible so the
amount of ETH indicated in the msg.value will be transferred to the contract address.
    }

    function withdraw(uint amount) public payable onlyClients{ //
withdraw == contract to client by onlyclient payable(msg.sender).transfer(amount * 1
        ether); // The address
of the sender( ie contract ) is held in the msg.sender variable.
        //The withdraw method will be used to send ETH from the contract to the client
account. It sends the unit of ETH indicated in the amount parameter, from the contract to
the client who sent the transaction. We want this method to be callable only by clients
who've joined the contract either,
        // so the onlyClient modifier is used for this restriction.
    }

    function sendInterest() public payable onlyManager{ //The sendInterest method
will be used to send ETH as interest from the contract to all clients. can called by
only manager
        for(uint i=0;i<clients.length;i++){ // check client in database address
            initialAddress = clients[i].client_address; //
check client address

            payable(initialAddress).transfer(1 ether);

        }
    }

    function getContractBalance() public view returns(uint){
//getContractBalance method will be used to get the balance of the contract we
deployed.
        return address(this).balance;
    }
}

//Output steps
//Initially All Acount has 100 fake ether
//Step 1: Select first Address
(eg.0x5B38Da6a701c568545dCfcB03FcB875f56beddC4)
//Step 2: Click on Deploy button(Contract Created,Can view under
```

```
Deployed Contract)
//After deploying contract 100 ETH turns to 99.99999.....ETH
```

```
//Step 3: Set Manager: Follow Following instructions
//              i.Select Onother
Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)
//              ii.Copy this address
(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2) and paste it in contract,
infront of set Manager button
//              iii. click on set manager button, Now
Manager=0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
//Step 4: join as Client: Follow Following instructions
//              i.Select Onother
Address(eg.0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db)
//              ii.Copy this address
(eg.0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db) and paste it in contract,
infront of joinAsClient button
//              iii.click on joinAsClient button, Now
Client=0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
//Initially Balanve in contract Will be 0 ETH(can view in Deployed
Contract @ Bottom)
//Step 5: Deposit:
//              i.Select Client
Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)
//              ii. Enter 10 ETH ammount in Value Field, Select unit as ETH
from dropdown
//              iii. Come to the Deployed Contract, Click on deposit button
//              iv. 10 ETH transper to Contract , Balance will be updated to
10 ETH in Contract
//              V. 10 ETH will be minus from clients Wallet
//Step 6: Withdraw:
//              i.Select Client
Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)
//              ii. Enter 5 ETH ammount in front of withdrow button
//              iii. Click on withdrow button
//              iv. 5 ETH transper to Wallet
//              V. 5 ETH will be Added to clients Wallet
//Step 7: Send Interest:
//              i.Select Manager
```

```
Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)
//          ii.Come to contract, Click on sendInterest button
//          iii. According to logic written in code, 1 ETH as interest will be send to Client
Wallet
//Step 8: getContract Balance:
//              i.Click on getcontract balance button to view total balance in contract
```

Assignment No: 4

Title of the Assignment: Write a program in solidity to create Student data. Use the following constructs:

- Structures
- Arrays
- Fallback

Deploy this as smart contract on Ethereum and Observe the transaction fee and Gas values.

Objective of the Assignment: Students should be able to learn about Solidity. Its datatypes and implementations.

Prerequisite:

1. Basic Programming Logic
2. Basic knowledge of Solidity

Contents for Theory:

1. **Solidity - Arrays**
2. **Solidity - Structures**
3. **Solidity – Fallback**
4. **Implementation**

1. Solidity – Arrays :

Arrays are data structures that store the fixed collection of elements of the same data types in which each and every element has a specific location called index. Instead of creating numerous individual variables of the same type, we just declare one array of the required size and store the elements in the array and can be accessed using the index. In Solidity, an array can be of fixed size or dynamic size. Arrays have a continuous memory location, where the lowest index corresponds to the first element while the highest represents the last.

Creating an Array

To declare an array in Solidity, the data type of the elements and the number of elements should be specified. The size of the array must be a positive integer and data type should be a valid Solidity type

Syntax:

<data type> <array name>[size] = <initialization>

Fixed-size Arrays :

The size of the array should be predefined. The total number of elements should not exceed the size of the array. If the size of the array is not specified then the array of enough size is created which is enough to hold the initialization.

Dynamic Array:

The size of the array is not predefined when it is declared. As the elements are added the size of array changes and at the runtime, the size of the array will be determined.

Array Operations :

1. Accessing Array Elements:

The elements of the array are accessed by using the index. If you want to access ith element then you have to access (i-1)th index.

2. Length of Array:

Length of the array is used to check the number of elements present in an array. The size of the memory array is fixed when they are declared, while in case the dynamic array is defined at runtime so for manipulation length is required.

3. Push:

Push is used when a new element is to be added in a dynamic array. The new element is always added at the last position of the array.

4. Pop:

Pop is used when the last element of the array is to be removed in any dynamic array.

Solidity – Structures :

Structs in Solidity allows you to create more complicated data types that have multiple properties. You can define your own type by creating a **struct**.

They are useful for grouping together related data.

Structs can be declared outside of a contract and imported in another contract. Generally, it is used to represent a record. To define a structure *struct* keyword is used, which creates a new data type.

Syntax:

```
struct <structure_name> {
   <data type> variable_1;
   <data type> variable_2;
}
```

For accessing any element of the structure, 'dot operator' is used, which separates the struct variable and the element we wish to access. To define the variable of structure data type structure name is used.

3. Solidity – Fallback :

The solidity fallback function is executed if none of the other functions match the

function identifier or no data was provided with the function call. Only one unnamed function can be assigned to a contract and it is executed whenever the contract receives plain Ether without any data. To receive Ether and add it to the total balance of the contract, the fallback function must be marked payable. **If no such function exists, the contract cannot receive Ether through regular transactions and will throw an exception.**

Properties of a fallback function:

1. Has no name or arguments.
2. If it is not marked **payable**, the contract will throw an exception if it receives plain ether without data.
3. Can not return anything.
4. Can be defined once per contract.
5. It is also executed if the caller meant to call a function that is not available
6. It is mandatory to mark it external.
7. It is limited to 2300 gas when called by another function. It is so for as to make this function call as cheap as possible.

4. Implementation

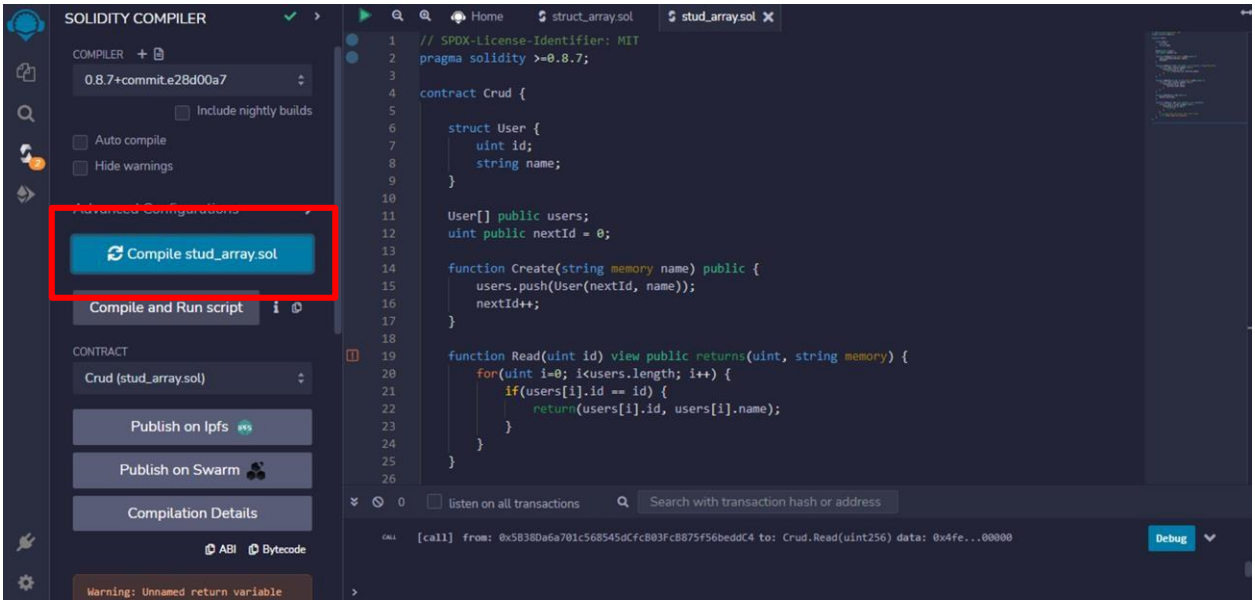
Code -

```
pragma solidity ^0.5.0;
contract Crud {
    struct User {
        uint id;
        string name;
    }
    User[] public users;
    uint public nextId = 0;
    function Create(string memory name) public {
        users.push(User(nextId, name));
        nextId++;
    }
    function Read(uint id) view public returns(uint, string memory) {
        for(uint i=0; i<users.length; i++) {
            if(users[i].id == id) {
                return(users[i].id, users[i].name);
            }
        }
    }
    function Update(uint id, string memory name) public {
```

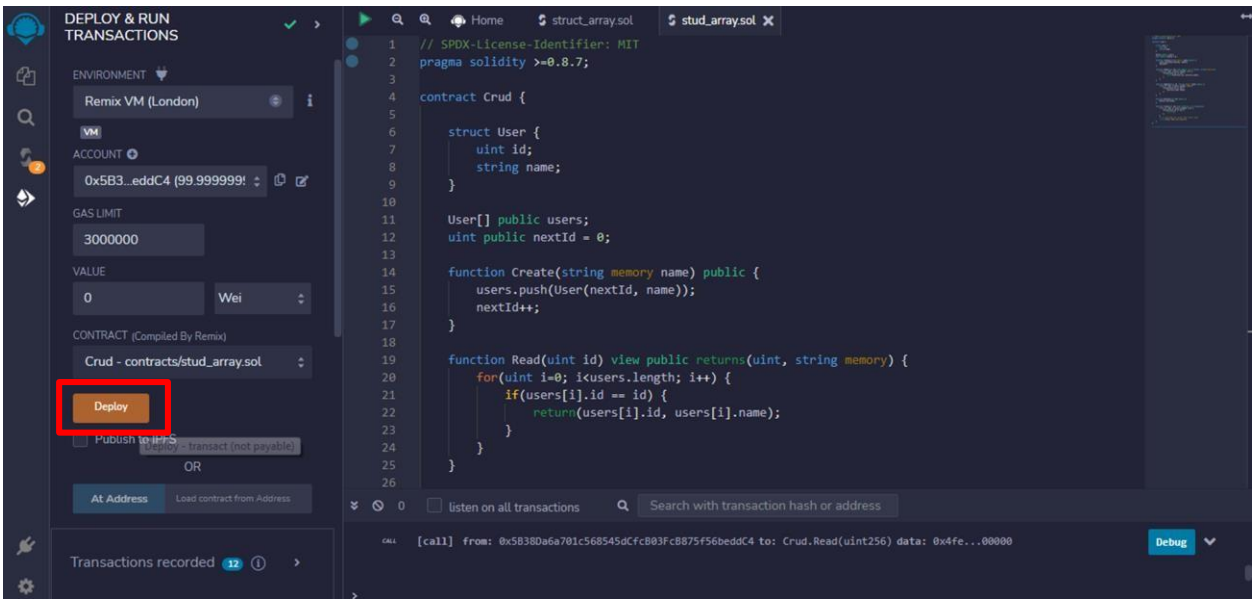
```
                for(uint i=0; i<users.length; i++) {
                    if(users[i].id == id) {
                        users[i].name =name;
                    }
                }
        }
        function Delete(uint id) public {
            delete users[id];
        }
        function find(uint id) view internal returns(uint) {
            for(uint i=0; i< users.length; i++) {
                if(users[i].id == id) {
                    return i;
                }
            }
            // if user does not exist then revert back
            revert("User does not exist");
        }
    }
```

Output –

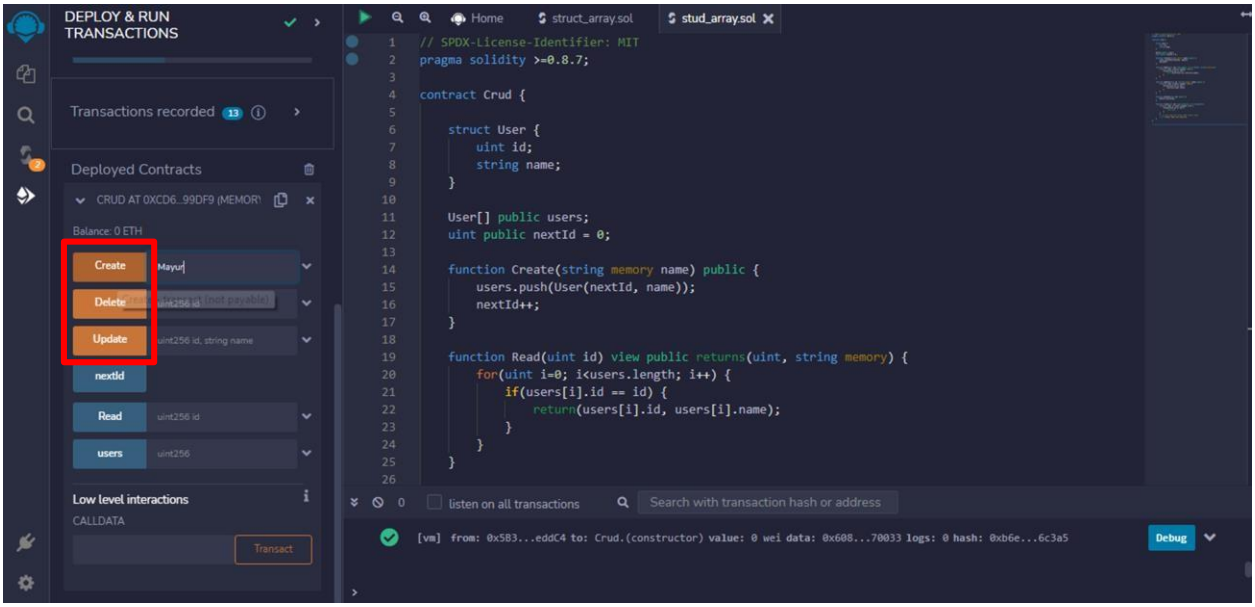
Step 1 – Compile the program by clicking on compile button.



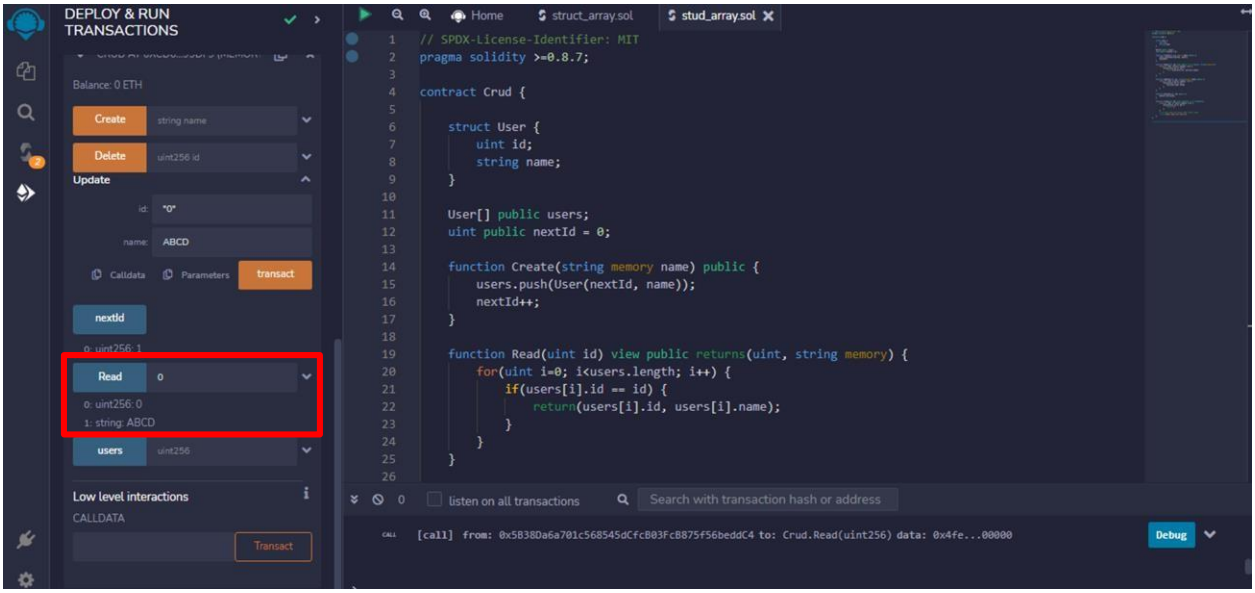
Step 2 – After the Successful compilation, Deploy the contract to see the output.



Step 3 - Now you can check the output. You can insert, update and delete the student data using your smart contract.



Step 4 – After entering your data, you can read the data using ID.



Conclusion-

In this way we have created array, structure and used fallback function in solidity.

Assignment Question:

1. **What is fixed array and dynamic array in solidity?**
2. **What is Array in solidity ?**
3. **What is structure in solidity? Define its syntax.**
4. **What is fallback function ?**

Reference link

- https://www.geeksforgeeks.org/solidity-arrays/?ref=lbp
- tutorialspoint.com/solidity/solidity_structs.htm

Assignment No : 5

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:

Public Blockchain.

Private Blockchain.

Hybrid Blockchain.

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 computerin 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 thenetwork Secure: This blockchain is large in size as it is open to the public. In a large size, there is greater distribution of recordsAnonymous Nature: It is a secure platform to make your transaction properly at the same time, you are notrequired 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 theledger.
Disadvantages:

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

Energy Consumption: Proof of work is high energy-consuming. It requires good computer hardware toparticipate in thenetwork

Acceptance: No central authority is there so governments are facing the issue to implement the technologyfaster.

Use Cases: Public Blockchain is secured with proof of work or proof of stake they can be used to displace traditionalfinancial systems. The more advanced side of this blockchain is the smart contract that enabledthis 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 inthe process,making it more secure than the others.

These are not as open as a public blockchain.
Theyare 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 hasincreased 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 theperformance of thenetwork.

Disadvantages:

Security- The number of nodes in this type is limited so chances of manipulation are there. Theseblockchains are morevulnerable.

Centralized- Trust building is one of the main disadvantages due to its central nature. Organizations can usethis formalpractices.

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 thepublic eye. Therefore companies use them for internal auditing, voting, and assetmanagement. An example of privateblockchains is Hyperledger, Corda.

3. Hybrid Blockchain
It is the mixed content of the private and public blockchain, where some part is controlled by someorganization andother makes are made visible as a public blockchain.

It is a combination of both public and private blockchain.
Permission-based and permissionless 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 usersdon’t have access to the network

Cost: Transactions are cheap as only a few nodes verify the transaction. All the nodes don’t carry theverification henceless computational cost.

Architecture: It is highly customizable and still maintains integrity, security, and transparency.
Operations: It canchoose the participants in the blockchain and decide which transaction can be madepublic.

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 Tendermint and Multichain.

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