

STES's
SINHGAD COLLEGE OF ENGINEERING
Vadgaon(Bk), Pune

Department of Computer Engineering



LABORATORY MANUAL

2022-23

LABORATORY PRACTICE-III
BE-COMPUTER ENGINEERING
SEMESTER-I

Subject Code:410246

TEACHING SCHEME

Lectures: 4Hrs/Week

Practical: 2 Hrs/Week

EXAMINATION SCHEME

Practical: 50 Marks

Term Work: 50 Marks

-: Name of Faculty:-

Prof.P.M.Kamde

Prof. S.P.Bolane

Prof. A.M.Karanjkar

Prof. M.A. Khade

Prof. A.V.Dirgule

INDEX

GROUP A: DESIGN & ANALYSIS ALGORITHM

Any 5 assignments and 1 mini project are mandatory.

Sr.No.	Title	Page Number
1	Write a program to calculate Fibonacci numbers and find its step count.	
2	Write a program to implement Job sequencing with deadlines using a greedy method.	
3	Write a program to implement Huffman Encoding using a greedy strategy.	
4	Write a program to solve a fractional Knapsack problem using a greedy method.	
5	Write a program to solve a 0-1 Knapsack problem using dynamic programming or branch and bound strategy.	
6	Design 8-Queens matrix having first Queen placed. Use backtracking to place remaining Queens to generate the final 8-queen's matrix.	
7	Write a program for analysis of quick sort by using deterministic and randomized variant.	

8	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.	

OR

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

OR

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

OR

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

GROUP B: MACHINE LEARNING

Any 5 assignments and 1 mini project are mandatory.

Sr.No.	Title	Page Number
1	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.	
2	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	
3	Given a bank customer, build a neural network-based classifier that can determine whether they will leave or not in the next 6 months. Dataset Description: The case study is from an open-source dataset from Kaggle. The dataset contains 10,000 sample points with 14 distinct features such as Customer Id, Credit Score, Geography, Gender, Age, Tenure, Balance, etc.	
4	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 $x=2$.	
5	Implement K-Nearest Neighbors algorithm on diabetes.csv dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset.	
6	Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method.	

7	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..	

OR

	Build a machine learning model that predicts the type of people who survived the Titanic shipwreck using passenger data (i.e. name, age, gender, socio-economic class, etc.).	
--	---	--

OR

	Develop an application for signature identification by creating your own dataset of your college student.	
--	---	--

GROUP C: BLOCKCHAIN TECHNOLOGY

Any 5 assignments and 1 mini project are mandatory.

Sr.No.	Title	Page Number
1	Installation of Metamask and study spending Ether per transaction	
2	Create your own wallet using Metamask for crypto transactions.	
3	Write a smart contract on a test network, for Bank account of a customer for following operations: -Deposit money -Withdraw Money -Show balance	

4	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.	
5	Write a survey report on types of Blockchains and its real time use cases.	
6	Write a program to create a Business Network using Hyperledger.	

7	Mini Project	
	Create a dApp (de-centralized app) for e-voting system	

OR

	Develop a Blockchain based application for transparent and genuine charity	
--	--	--

OR

	Develop a Blockchain based application for health related medical records	
--	---	--

OR

	Develop a Blockchain based application for mental health	
--	--	--

GROUP A: DESIGN AND ANALYSIS ALGORITHM

Assignment No.	1 (GROUP A)
Title	Write a program to calculate Fibonacci numbers and find its step count.
Subject	Laboratory Practice-III
Class	B.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No: 1

Title: Calculate Fibonacci numbers and find its step count.

Problem Statement: Develop a program in C++ to create a Fibonacci Series.

Prerequisites: Design and analysis algorithm.

Objectives: Implement non-recursive and recursive program to calculate Fibonacci numbers find step count.

Outcome: To understand the implementation of Fibonacci numbers and analyze and find its step Count.

Theory:

Fibonacci Series: Fibonacci series is defined as a sequence of numbers in which the first two numbers are 1 and 1, or 0 and 1, depending on the selected beginning point of the sequence, and each subsequent number is the sum of the previous two. So, in this series, the n^{th} term is the sum of $(n-1)^{\text{th}}$ term and $(n-2)^{\text{th}}$ term. **Fibonacci Series in C++:** In case of fibonacci series, next number is the sum of previous two numbers for example 0, 1, 1, 2, 3, 5, 8, 13, 21 etc. The first two numbers of fibonacci series are 0 and 1.

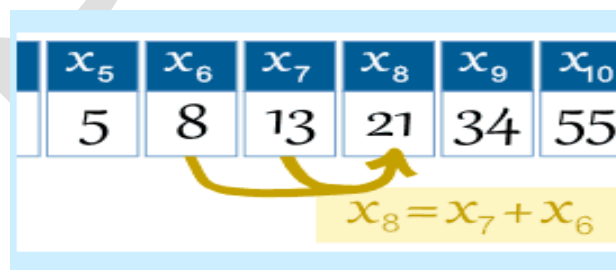
There are two ways to write the fibonacci series program:

- Fibonacci Series without recursion
- Fibonacci Series using recursion

Generate Fibonacci Series?

Mathematically, the n^{th} term of the Fibonacci series can be represented as:

$$t_n = t_{n-1} + t_{n-2}$$



The Fibonacci numbers upto certain term can be represented as: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144....

Fibonacci Series in C++ without Recursion

```
#include <iostream>
using namespace std;
int main() {
    int n1=0,n2=1,n3,i,number;
    cout<<"Enter the number of elements: ";
    cin>>number;
    cout<<n1<<" "<<n2<<" "; //printing 0 and 1
    for(i=2;i<number;++i) //loop starts from 2 because 0 and 1 are already printed
    {
        n3=n1+n2;
        cout<<n3<<" ";
        n1=n2;
        n2=n3;
    }
    return 0;
}
```

Output:

Enter the number of elements: 10
0 1 1 2 3 5 8 13 21 34

Fibonacci series using recursion in C++

```
#include<iostream>
using namespace std;
void printFibonacci(int n)
{
    static int n1=0, n2=1, n3;
    if(n>0){
        n3 = n1 + n2;
        n1 = n2;
        n2 = n3;
    }
}
```



```
cout<<n3<<" ";
    printFibonacci(n-1);
}
}
int main(){
    int n;
    cout<<"Enter the number of elements: ";
    cin>>n;
    cout<<"Fibonacci Series: ";
    cout<<"0 "<<"1 ";
    printFibonacci(n-2); //n-2 because 2 numbers are already printed
    return 0;
}
```

Output:

Enter the number of elements: 15

Fibonacci Series: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

Facilities:

Linux Operating Systems, G++

Algorithm:

Step-1: Start

Step-2: Declare variables i, a,b , show

Step-3: Initialize the variables, a=0, b=1, and show =0

Step-4: Enter the number of terms of Fibonacci series to be printed

Print First two terms of series

Step-5: Use loop for the following steps

-> show=a+b

-> a=b

-> b=show

-> increase value of i each time by 1

-> print the value of show

Step-6: End

Input:

Take the term number as an input. Say it is 10

Output:

Enter the number of elements: 14

Fibonacci Series: 0 1 1 2 3 5 8 13 21 34 55 89 144 233

Enter number of terms: 10

10th Fibonacci Terms:55

Conclusion:

Hence, we have studied concept of Fibonacci Series.

Questions:

1. What is Fibonacci Series example?
2. What are the first 10 Fibonacci series number?
3. Explain Fibonacci search technique.?
4. What is Fibonacci series formula?

Assignment No.	2 (GROUP A)
Title	Write a program to implement Job sequencing with deadlines using a greedy method.
Subject	Laboratory Practice-III
Class	B.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No: 2

Title: Job sequencing with deadlines using a greedy method.

Problem Statement: Implement and find the maximum profit job sequencing with deadlines using a greedy method.

Prerequisites: Design and analysis algorithm.

Objectives: The objective is to determine the sequence of performing jobs such that we can minimize the total cost/time. Implement job sequence based on deadline to earn max profit.

Outcome: To understand the implementation of job sequencing.

Theory:

Job scheduling is the problem of scheduling jobs out of a set of N jobs on a single processor which maximizes profit as much as possible. Consider N jobs, each taking unit time for execution. Each job is having some profit and deadline associated with it. The job is feasible only if it can be finished on or before its deadline. A feasible solution is a subset of N jobs such that each job can be completed on or before its deadline. An optimal solution is a solution with maximum profit. The simple and inefficient solution is to generate all subsets of the given set of jobs and find the feasible set that maximizes the profit. For N jobs, there exist 2^N schedules, so this brute force approach runs in $O(2^N)$ time.

Complexity Analysis of Job Scheduling

Simple greedy algorithm spends most of the time looking for the latest slot a job can use. On average, N jobs search $N/2$ slots. This would take $O(N^2)$ time. However, with the use of set data structure (find and union), the algorithm runs nearly in $O(N)$ time.

Examples

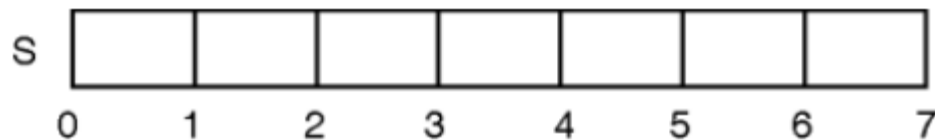
Problem: Solve the following instance of “job scheduling with deadlines” problem : $n = 7$, profits $(p_1, p_2, p_3, p_4, p_5, p_6, p_7) = (3, 5, 20, 18, 1, 6, 30)$ and deadlines $(d_1, d_2, d_3, d_4, d_5, d_6, d_7) = (1, 3, 4, 3, 2, 1, 2)$. Schedule the jobs in such a way to get maximum profit.

Given that,

Jobs	J_1	J_2	J_3	J_4	J_5	J_6	J_7
Profit	3	5	20	18	1	6	30
Deadline	1	3	4	3	2	1	2

Sort all jobs in descending order of profit.

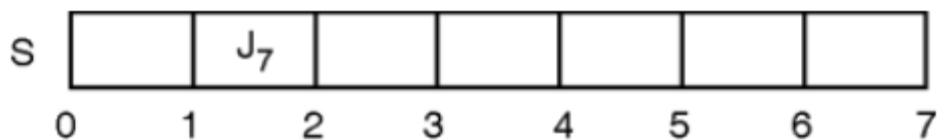
So, $P = (30, 20, 18, 6, 5, 3, 1)$, $J = (J_7, J_3, J_4, J_6, J_2, J_1, J_5)$ and $D = (2, 4, 3, 1, 3, 1, 2)$. We shall select one by one job from the list of sorted jobs J , and check if it satisfies the deadline. If so, schedule the job in the latest free slot. If no such slot is found, skip the current job and process the next one. Initially,



Profit of scheduled jobs, $SP = 0$

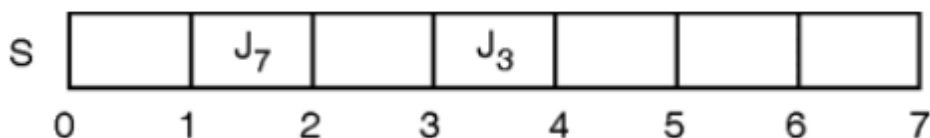
Iteration 1:

Deadline for job J_7 is 2. Slot 2 ($t = 1$ to $t = 2$) is free, so schedule it in slot 2. Solution set $S = \{J_7\}$, and Profit $SP = \{30\}$



Iteration 2:

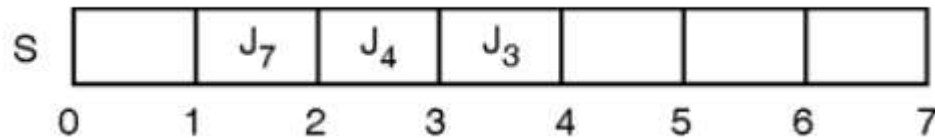
Deadline for job J_3 is 4. Slot 4 ($t = 3$ to $t = 4$) is free, so schedule it in slot 4. Solution set $S = \{J_7, J_3\}$, and Profit $SP = \{30, 20\}$



Iteration 3:

Deadline for job J_4 is 3. Slot 3 ($t = 2$ to $t = 3$) is free, so schedule it in slot 3.

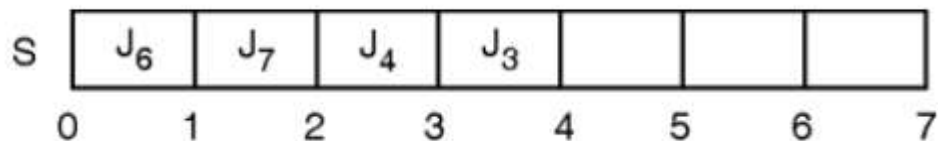
Solution set $S = \{J_7, J_3, J_4\}$, and Profit $SP = \{30, 20, 18\}$

**Iteration 4:**

Deadline for job J_6 is 1. Slot 1 ($t = 0$ to $t = 1$) is free, so schedule it in slot 1.

Solution set $S = \{J_7, J_3, J_4, J_6\}$, and Profit

$SP = \{30, 20, 18, 6\}$



First, all four slots are occupied and none of the remaining jobs has deadline lesser than 4. So none of the remaining jobs can be scheduled. Thus, with the greedy approach, we will be able to schedule four jobs $\{J_7, J_3, J_4, J_6\}$, which give a profit of $(30 + 20 + 18 + 6) = 74$ units.

```
#include<iostream>
#include<algorithm>
using namespace std;
// A structure to represent a job
struct Job
{
    char id;
    int dead;
    int profit;
};
// This function is used for sorting all the jobs according to the profit
bool compare(Job a, Job b)
{
    return (a.profit > b.profit);
}
```

```
void jobschedule (Job arr[], int n)
{
    // Sort all jobs according to decreasing order of profit
    sort(arr, arr+n, compare);
    int result[n]; // To store result
    bool slot[n];
    // Initialize all slots to be free
    for (int i=0; i<n; i++)
        slot[i] = false;
    for (int i=0; i<n; i++)
    {
        // Find a free slot for this job (Note that we start
        // from the last possible slot)
        for (int j=min(n, arr[i].dead)-1; j>=0; j--)
        {
            // Free slot found
            if (slot[j]==false)
            {
                result[j] = i; // Add this job to result
                slot[j] = true; // Make this slot occupied
                break;
            }
        }
    }
    // Print the result
    for (int i=0; i<n; i++)
        if (slot[i])
            cout << arr[result[i]].id << " ";
}

int main()
{
    Job arr[] = { {'a', 2, 20}, {'b', 2, 15}, {'c', 1, 10}, {'d', 3, 5}, {'e', 3, 1} };
    int n = 5;
    cout << "maximum profit sequence of jobs is-->";
    jobschedule(arr, n);
}
```

Output:

maximum profit sequence of jobs is-->b a d

Facilities:

Linux Operating Systems, G++

Algorithm:**Algorithm: Job-Sequencing-With-Deadline (D, J, n, k)**

$D(0) := J(0) := 0$

$k := 1$

$J(1) := 1$ // means first job is selected

for $i = 2 \dots n$ do

$r := k$

 while $D(J(r)) > D(i)$ and $D(J(r)) \neq r$ do

$r := r - 1$

 if $D(J(r)) \leq D(i)$ and $D(i) > r$ then

 for $l = k \dots r + 1$ by -1 do

$J(l + 1) := J(l)$

$J(r + 1) := i$

$k := k + 1$

Analysis

In this algorithm, we are using two loops, one is within another. Hence, the complexity of this algorithm is $O(n^2)O(n^2)$.

Input: Five Jobs with following deadlines and profits

JobID Deadline Profit

a	2	100
b	1	19
c	2	27
d	1	25
e	3	15

Output:

Following is maximum profit sequence of jobs

Sequence of jobs: c a e

Maximum Profit: 142

Conclusion:

Hence, we have successfully studied concept of job sequencing with deadlines using greedy method.

Questions:

1. What is job sequencing with deadlines using greedy method?
2. Which algorithm technique is used to solve the job scheduling?
3. How do you implement job sequencing with deadline.
4. What is the technique used to solve job sequencing with deadlines?
5. What is time complexity of job sequencing with deadlines using greedy method?

Assignment No.	3 (GROUP A)
Title	Huffman Encoding using a greedy strategy.
Subject	Laboratory Practice-III
Class	B.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No: 3

Title: Huffman Encoding using a greedy strategy.

Problem Statement: Implement and find the characters with the frequencies Huffman encoding using greedy strategy.

Prerequisites: Design and analysis algorithm.

Objectives: The objective is to Implement Huffman Encoding using greedy method.

Outcome: To understand the implementation of Huffman Encoding using greedy method.

Theory:

Huffman coding is a greedy algorithm frequently used for lossless data compression. Therefore, a basic principle of Huffman coding is to compress and encode the text or the data depending on the frequency of the characters in the text.

The basic idea behind the Huffman coding algorithm is to assign the variable-length codes to input characters of text based on the frequencies of the corresponding character. So, the most frequent character gets the smallest code, and the least frequent character is assigned with the largest code.

Here, the codes assigned to the characters are termed prefix codes which means that the code assigned to one character is not the prefix of the code assigned to any other character. Using this technique, Huffman coding ensures that there is no ambiguity when decoding the generated bit stream.

Huffman encoding implements the following steps.

- It assigns a variable-length code to all the given characters.
- The code length of a character depends on how frequently it occurs in the given text or string.
- A character gets the smallest code if it frequently occurs.
- A character gets the largest code if it least occurs.

Huffman coding follows a **prefix rule** that prevents ambiguity while decoding. The rule also ensures that the code assigned to the character is not treated as a prefix of the code assigned to any other character.

There are the following two major steps involved in Huffman coding:

- First, construct a **Huffman tree** from the given input string or characters or text.
- Assign, a Huffman code to each character by traversing over the tree.

Suppose, we have to encode string **abracadabra**. Determine the following:

- Huffman code for All the characters
- Average code length for the given String
- Length of the encoded string

(i) Huffman Code for All the Characters

In order to determine the code for each character, first, we construct a **Huffman tree**.

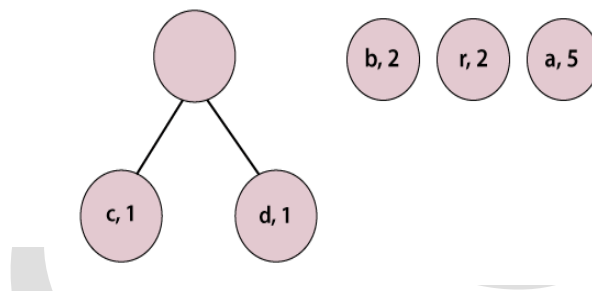
Step 1: Make pairs of characters and their frequencies.

(a, 5), (b, 2), (c, 1), (d, 1), (r, 2)

Step 2: Sort pairs with respect to frequency, we get:

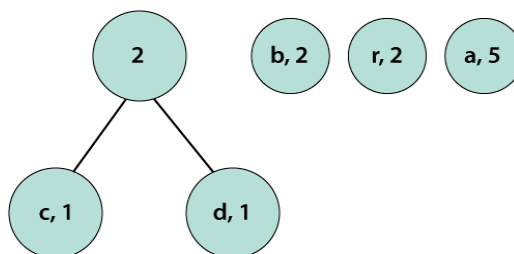
(c, 1), (d, 1), (b, 2), (r, 2), (a, 5)

Step 3: Pick the first two characters and join them under a parent node.

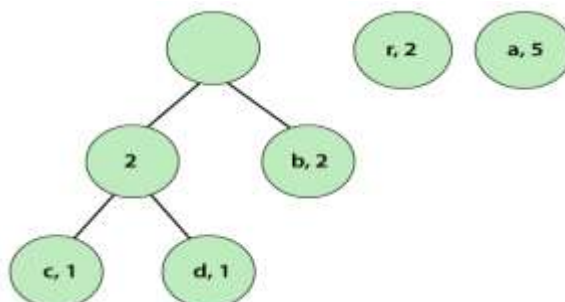


We observe that a parent node does not have a frequency so, we must assign a frequency to it. The parent node frequency will be the sum of its child nodes (left and right) i.e. $1+1=2$.

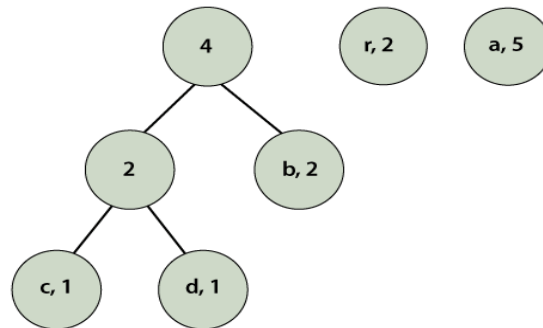
Step 4: Repeat Steps 2 and 3 until, we get a single tree.



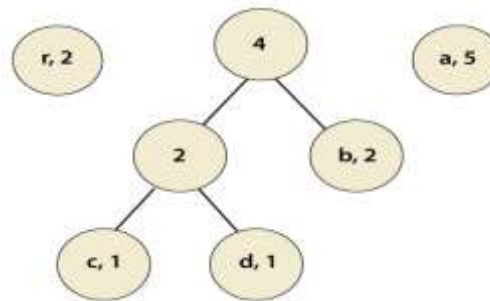
We observe that the pairs are already in a sorted (by step 2) manner. Again, pick the first two pairs and join them.



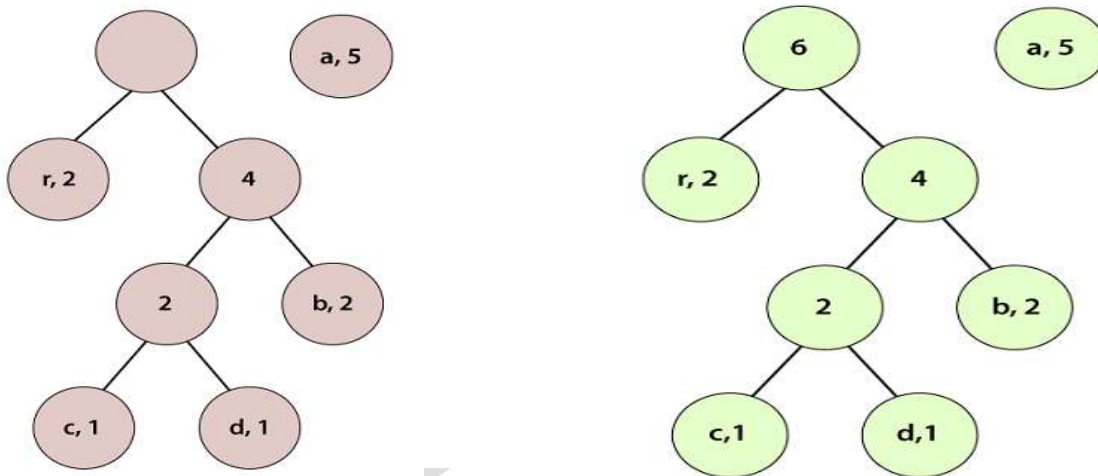
We observe that a parent node does not have a frequency so, we must assign a frequency to it. The parent node frequency will be the sum of its child nodes (left and right) i.e. $2+2=4$.



Again, we check if the pairs are in a sorted manner or not. At this step, we need to sort the pairs.

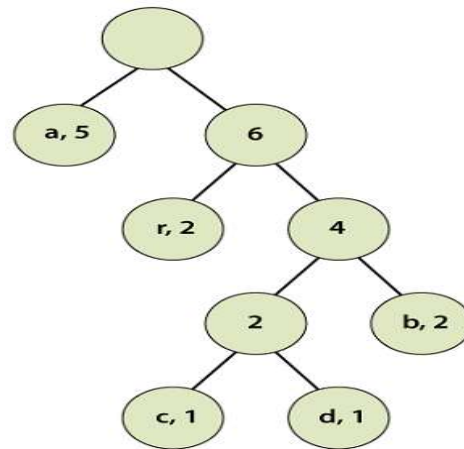
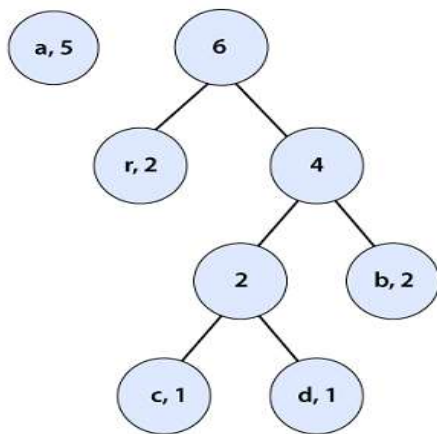


According to step 3, pick the first two pairs and join them, we get:



We observe that a parent node does not have a frequency so, we must assign a frequency to it. The parent node frequency will be the sum of its child nodes (left and right) i.e. $2+4=6$.

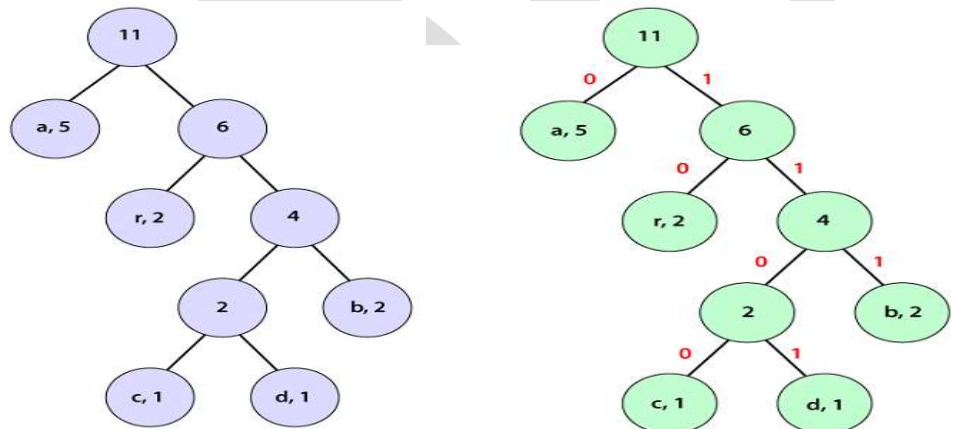
Again, we check if the pairs are in a sorted manner or not. At this step, we need to sort the pairs. After sorting the tree looks like the following:



According to step 3, pick the first two pairs and join them, we get:

We observe that a parent node does not have a frequency so, we must assign a frequency to it. The parent node frequency will be the sum of its child nodes (left and right) i.e. $5+6=11$.

Therefore, we get a single tree.



At last, we will find the code for each character with the help of the above tree. Assign a weight to each edge. Note that each **left edge-weighted is 0** and the **right edge-weighted is 1**.

We observe that input characters are only presented in the leaf nodes and the internal nodes have null values. In order to find the Huffman code for each character, traverse over the Huffman tree from the root node to the leaf node of that particular character for which we want to find code. The table describes the code and code length for each character.

Character	Frequency	Code	Code Length
A	5	0	1
B	2	111	3
C	1	1100	4
D	1	1101	4
R	2	10	2

Algorithm:

```

Huffman (C)
n=|C|
Q=C
for i=1 to n-1
  do
    z=allocate_Node()
    x=left[z]=Extract_Min(Q)
    y=right[z]=Extract_Min(Q)
    f[z]=f[x]+f[y]
  Insert(Q,z)
return Extract_Min(Q)

```

Time Complexity-

The time complexity analysis of Huffman Coding is as follows-

- extract Min() is called $2 * (n-1)$ times if there are n nodes.
- As extract Min() calls min Heapify(), it takes $O(\log n)$ time.

Thus, Overall time complexity of Huffman Coding becomes **$O(n \log n)$** .

Facilities:

Linux Operating Systems, G++

Input:

Character	Frequencies
A	5
B	1
C	6
D	3

Output:

Character	Huffman Code
C	0
B	100
D	101
A	11

Conclusion:

Hence, we have successfully studied concept of Huffman encoding using greedy method.

Questions:

1. How do you encode with Huffman?
2. Why do we use Huffman encoding?
3. What is time complexity of Huffman encoding.
4. What is Huffman encoding explain with example?

Assignment No.	4 (GROUP A)
Title	Solve a fractional Knapsack problem using a greedy method
Subject	Laboratory Practice-III
Class	B.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No: 4

Title: Solve a fractional Knapsack problem using a greedy method

Problem Statement: Given a set of N items each having value V with weight W and the total capacity of a knapsack. The task is to find the maximal value of fractions of items that can fit into the knapsack.

Prerequisites: Design and analysis algorithm

Objective: The objective is to fill the knapsack of some given volume with different such that value of selected items is maximized.

Outcome: To understand the implementation of greedy method using fractional knapsack problem.

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. It appears as a subproblem in many, more complex mathematical models of real-world problems. One general approach to difficult problems is to identify the most restrictive constraint, ignore the others, solve a knapsack problem, and somehow adjust the solution to satisfy the ignored constraints. The fractional knapsack problem is also one of the techniques which are used to solve the knapsack problem. In fractional knapsack, the items are broken in order to maximize the profit. The problem in which we break the item is known as a Fractional knapsack problem.

Greedy approach: In Greedy approach, we calculate the ratio of profit/weight, and accordingly, we will select the item. The item with the highest ratio would be selected first.

Fractional Knapsack Problem Solution in C++ and Java

The same approach we are using in our program.

- We have taken an array of structures named **Item**.
- Each **Item** has **value & weight**.
- We are calculating **density= value/weight** for each item and sorting the *items* array in the order of decreasing density.
- We add values from the top of the array to **totalValue** until the bag is full i.e. **totalValue<=W** (where W is Knapsack weight).

There are basically three approaches to solve the problem:

- The first approach is to select the item based on the maximum profit.
- The second approach is to select the item based on the minimum weight.
- The third approach is to calculate the ratio of profit/weight.

Consider the below example:

Objects: 1 2 3 4 5 6 7

Profit (P): 10 15 7 8 9 4

Weight(w): 1 3 5 4 1 3 2

W (Weight of the knapsack): 15

n (no of items): 7

First approach:

Object	Profit	Weight	Remaining weight
3	15	5	$15 - 5 = 10$
2	10	3	$10 - 3 = 7$
6	9	3	$7 - 3 = 4$
5	8	1	$4 - 1 = 3$
7	$7 * \frac{3}{4} = 5.25$	3	$3 - 3 = 0$

The total profit would be equal to $(15 + 10 + 9 + 8 + 5.25) = 47.25$

Second approach:

The second approach is to select the item based on the minimum weight.

Object	Profit	Weight	Remaining weight
1	5	1	$15 - 1 = 14$
5	7	1	$14 - 1 = 13$
7	4	2	$13 - 2 = 11$
2	10	3	$11 - 3 = 8$

6	9	3	$8 - 3 = 5$
4	7	4	$5 - 4 = 1$
3	$15 * 1/5 = 3$	1	$1 - 1 = 0$

In this case, the total profit would be equal to $(5 + 7 + 4 + 10 + 9 + 7 + 3) = 46$

Third approach:

In the third approach, we will calculate the ratio of profit/weight.

Objects: 1 2 3 4 5 6 7

Profit (P): 5 10 15 7 8 9 4

Weight(w): 1 3 5 4 1 3 2

In this case, we first calculate the profit/weight ratio.

Object 1: $5/1 = 5$

Object 2: $10/3 = 3.33$

Object 3: $15/5 = 3$

Object 4: $7/4 = 1.7$

Object 5: $8/1 = 8$

Object 6: $9/3 = 3$

Object 7: $4/2 = 2$

P:w: 5 3.3 3 1.7 8 3 2

In this approach, we will select the objects based on the maximum profit/weight ratio. Since the P/W of object 5 is maximum so we select object 5.

Object	Profit	Weight	Remaining weight
5	8	1	$15 - 1 = 14$
1	5	1	$14 - 1 = 13$
2	10	3	$13 - 3 = 10$
3	15	5	$10 - 5 = 5$
6	9	3	$5 - 3 = 2$
7	4	2	$2 - 2 = 0$

As we can observe in the above table that the remaining weight is zero which means that the knapsack is full. We cannot add more objects in the knapsack. Therefore, the total profit would be equal to $(8 + 5 + 10 + 15 + 9 + 4)$, i.e., 51.

In the first approach, the maximum profit is 47.25. The maximum profit in the second approach is 46. The maximum profit in the third approach is 51. Therefore, we can say that the third approach, i.e., maximum profit/weight ratio is the best approach among all the approaches.

The following is the sample example of a Fractional knapsack problem in c++ programming :

```
// C++ program to solve fractional Knapsack Problem
#include <bits/stdc++.h>
using namespace std;
// Structure for an item which stores weight and corresponding value of Item
struct Item {
    int value, weight;
    // Constructor
    Item(int value, int weight)
    {
        this->value = value;
        this->weight = weight;
    }
};
// Comparison function to sort Item according to val/weight ratio
```

```
bool cmp(struct Item a, struct Item b)
{
    double r1 = (double)a.value / (double)a.weight;
    double r2 = (double)b.value / (double)b.weight;
    return r1 > r2;
}

double fractionalKnapsack(int W, struct Item arr[], int N)
{
    sort(arr, arr + N, cmp);
    double finalvalue = 0.0; // Result (value in Knapsack)
    for (int i = 0; i < N; i++) {
        // If adding Item won't overflow, add it completely
        if (arr[i].weight <= W) {
            W -= arr[i].weight;
            finalvalue += arr[i].value;
        }
        else {
            finalvalue += arr[i].value * ((double)W / (double)arr[i].weight);
            break;
        }
    }
    return finalvalue;
}

// Driver's code
int main()
{
    int W = 50; // Weight of knapsack
    Item arr[] = { { 60, 10 }, { 100, 20 }, { 120, 30 } };
    int N = sizeof(arr) / sizeof(arr[0]);
    // Function call
    cout << "Maximum value we can obtain = "
        << fractionalKnapsack(W, arr, N);
    return 0;
}
```

Output:

Maximum value we can obtain = 240

Facilities:

Linux Operating Systems, G++

Algorithm:**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
```

Input: A[] = { {60, 10} , {100, 20}, {120, 30} }, Total_capacity = 50

Output:

Enter total 3 items values and weight

Enter 1 value 60

Enter 1 weight 10

Enter 2 value 100

Enter 2 weight 20

Enter 3 value 120

Enter 3 weight 30

Entered data

Values: 60 100 120

Weight: 10 20 30

Enter knapsack weight

50

Total value in bag 50

---max value for 50 weight is 240

Process returned 0(0x0) execution time:17.376 s

Conclusion:

Hence, we have successfully studied concept of knapsack problem using greedy method.

Questions:

1. How do you solve fractional knapsack?
2. What is the time complexity of fractional knapsack problem?
3. Which approach is best in knapsack problems?
4. What is fractional knapsack problem? Explain with example?
5. what are the two ways to solve fractional knapsack problem?

Assignment No.	5 (GROUP A)
Title	Write a program to solve a 0-1 Knapsack problem using dynamic programming or branch and bound strategy
Subject	Laboratory Practice-III
Class	
Roll No.	
Date	
Signature	

Assignment No: 5

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

Problem Statement: We are given a set of n objects which each have a value v_i and a weight w_i .

Prerequisites: Design and analysis algorithm

Objective: Implement 0/1 Knapsack problem using dynamic programming or branch and bound strategy

Outcome: To understand the implementation of dynamic programming or branch and bound Strategy using 0/1 knapsack problem.

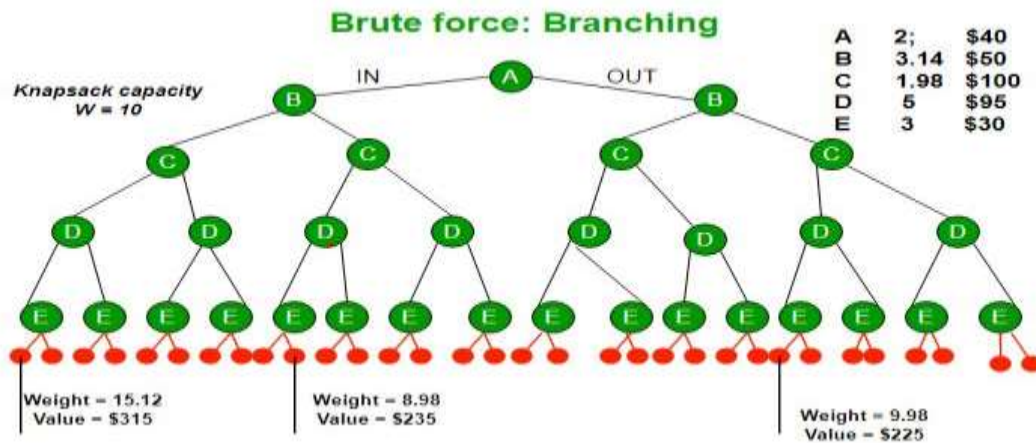
Theory:

Branch and bound is an algorithm design paradigm which is generally used for solving combinatorial optimization problems. These problems typically exponential in terms of time complexity and may require exploring all possible permutations in worst case.

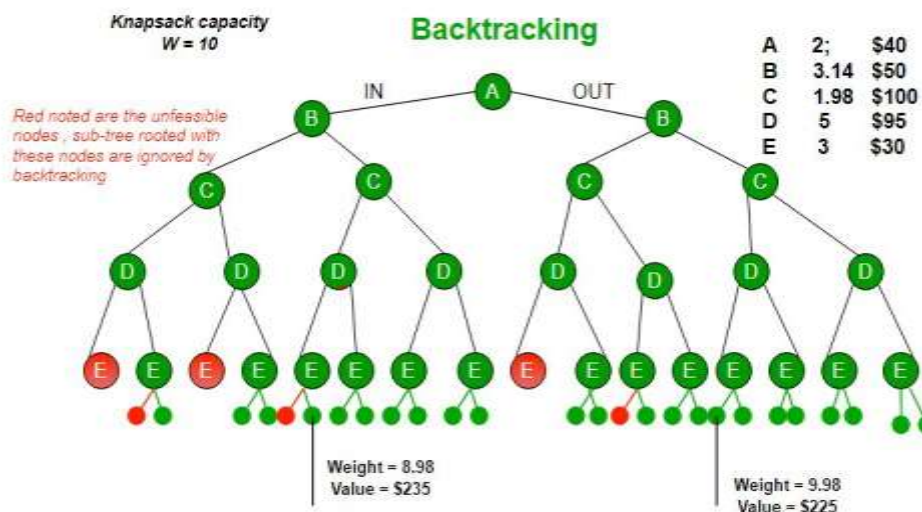
Let us consider below 0/1 Knapsack problem to understand Branch and Bound. Given two integer arrays **val[0..n-1]** and **wt[0..n-1]** that represent values and weights associated with n items respectively.

Find out the maximum value subset of **val []** such that sum of the weights of this subset is smaller than or equal to Knapsack capacity W . Let us explore all approaches for this problem.

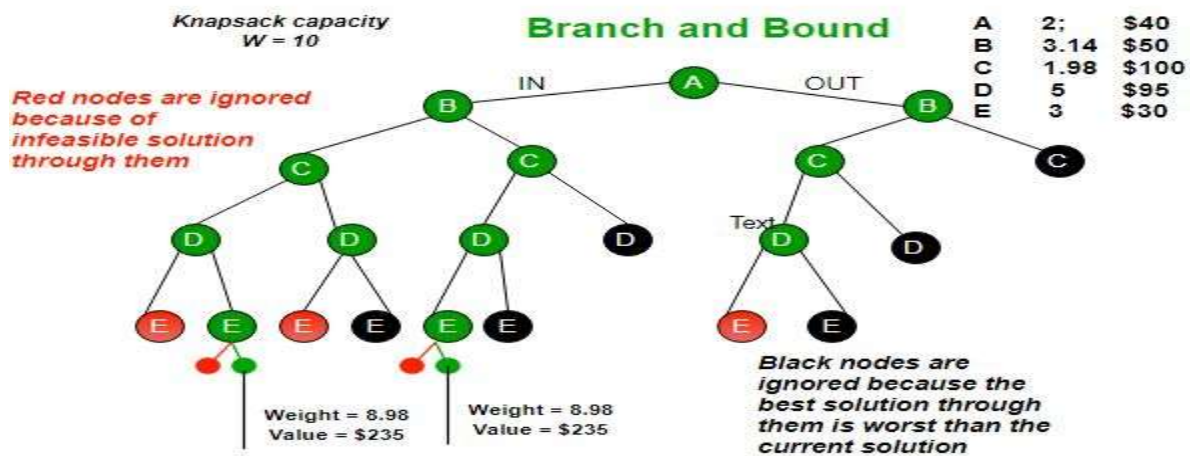
1. A **Greedy approach** is to pick the items in decreasing order of value per unit weight. The Greedy approach works only for [fractional knapsack](#) problem and may not produce correct result for [0/1 knapsack](#).
2. We can use [Dynamic Programming \(DP\) for 0/1 Knapsack problem](#). In DP, we use a 2D table of size $n \times W$. The **DP Solution doesn't work if item weights are not integers**.
3. Since DP solution doesn't always work, a solution is to use **Brute Force**. With n items, there are 2^n solutions to be generated, check each to see if they satisfy the constraint, save maximum solution that satisfies constraint. This solution can be expressed as **tree**.



We can use **Backtracking** to optimize the Brute Force solution. In the tree representation, we can do DFS of tree. If we reach a point where a solution no longer is feasible, there is no need to continue exploring. In the given example, backtracking would be much more effective if we had even more items or a smaller knapsack capacity.



Branch and Bound The backtracking based solution works better than brute force by ignoring infeasible solutions. We can do better (than backtracking) if we know a bound on best possible solution subtree rooted with every node. If the best in subtree is worse than current best, we can simply ignore this node and its subtrees. So we compute bound (best solution) for every node and compare the bound with current best solution before exploring the node. Example bounds used in below diagram are, **A** down can give \$315, **B** down can \$275, **C** down can \$225, **D** down can \$125 and **E** down can \$30. In the [next article](#), we have discussed the process to get these bounds.



Branch and bound is very useful technique for searching a solution but in worst case, we need to fully calculate the entire tree. At best, we only need to fully calculate one path through the tree and prune the rest of it.

Following is C++ implementation of above idea.

```
#include <iostream.h>
#include <conio.h>
using namespace std;
struct Item
{
    float weight;
    int value;
};
struct Node
{
    int level, profit, bound;
    float weight;
};
bool cmp(Item a, Item b)
{
    double r1 = (double)a.value / a.weight;
    double r2 = (double)b.value / b.weight;
    return r1 > r2;
}

int bound(Node u, int n, int W, Item arr[])
```

```
{
    if (u.weight >= W)
        return 0;
    int profit_bound = u.profit;
    int j = u.level + 1;
    int totweight = u.weight;
    while ((j < n) && (totweight + arr[j].weight <= W))
    {
        totweight += arr[j].weight;
        profit_bound += arr[j].value;
        j++;
    }
    if (j < n)
        profit_bound += (W - totweight) * arr[j].value / arr[j].weight;
    return profit_bound;
}
```

Facilities:

Linux Operating Systems, G++

Algorithm:

- Step 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.
- Step 2: Initialize maximum profit, max Profit = 0
- Step 3: Create an empty queue, Q.
- Step 4: Create a dummy node of decision tree and enqueue it to Q. Profit and weight of dummy node are 0.
- Step 5: Do following while Q is not empty.
- Step 6: Extract an item from Q. Let the extracted item be u.
- Step 7: Compute profit of next level node. If the profit is more than max Profit, then update Max Profit.
- Step 8: Compute bound of next level node. If bound is more than max Profit, then add next level node to Q.
- Step 9: 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.

Input:

enter the number of items 4

enter the weight and profit of the item 1:2 12
enter the weight and profit of the item 2:1 10
enter the weight and profit of the item 3:3 20
enter the weight and profit of the item 4:2 15
enter the capacity of the knapsack5

Output:

The table is

0	0	0	0	0	0
0	0	12	12	12	12
0	10	12	22	22	22
0	10	12	22	30	32
0	10	15	25	30	37

The maximum profit is= 37

The most valuable subset is :{ item 4: item 2: item 1 :}

Conclusion: We can see that the branch and bound method will give the solution to the problem by exploring just 1 path in the best case.

Questions:

1. How do you solve 0 1 knapsack problem using branch and bound?
2. What is the time complexity of 0 to 1 knapsack problem using branch and & bound method?
3. What is branch and bound algorithm example?
4. Which problem is solved by using branch and bound method?
5. What is 0/1 knapsack problem with example?

Assignment No.	6 (GROUP A)
Title	Design 8-Queens matrix having first Queen placed. Use backtracking to place remaining Queens to generate the final 8-queen's matrix.
Subject	Laboratory Practice-III
Class	B.E. C.E
Roll No.	
Date	
Signature	

Assignment No: 6

Title: Design 8-Queens matrix having first Queen placed. Use backtracking to place remaining Queens to generate the final 8-queen's matrix.

Problem Statement: This problem is to find an arrangement of N queens on a chess board, such that no queen can attack any other queens on the board.

Prerequisites: Design and analysis algorithm

Objective: Implement 8 Queens using backtracking.

Outcome: To understand the implementation of 8 Queens using backtracking.

Theory:

Backtracking:

In backtracking, we start with one possible move out of many available moves. We then try to solve the problem. If we are able to solve the problem with the selected move then we will print the solution. Else we will backtrack and select some other move and try to solve it. If none of the moves works out we claim that there is no solution for the problem.

8 Queen Problem:

This problem is commonly seen for $N=4$ and $N=8$. Let's look at an example where $N=8$

Before solving the problem, you need to know about the movement of the queen in chess. A queen can move any number of steps in any direction. The only constraint is that it can't change its direction while it's moving. One thing that is clear by looking at the queen's movement is that no two queens can be in the same row or column. N - Queens problem is to place n - queens in such a manner on an $n \times n$ chessboard that no queens attack each other by being in the same row, column or diagonal. 8-queens problem A classic combinatorial problem is to place 8 queens on a 8×8 chess board so that no two attack, i.e no two queens are to the same row, column or diagonal. Now, we will solve 8 queen's problem by using similar procedure adapted for 4 queen's problem. The algorithm of 8 queen's problem can be obtained by placing $n=8$, in N queen's algorithm. If two queens are placed at positions (i,j) and (k,l). They are on the same diagonal only

if $i-j=k-l$ (1)

or $i+j=k+l$ (2).

From (1) and (2) implies $j-l=i-k$ and $j-l=k-i$

Two queens lie on the same diagonal if $|j-l|=|i-k|$

The solution of 8 queens problem can be obtained similar to the solution of 4 queens. problem. $X_1=3, X_2=6, X_3=2, X_4=7, X_5=1, X_6=4, X_7=8, X_8=5$,

Place (k, i) returns a Boolean value that is true if the kth queen can be placed in column i. It tests both whether i is distinct from all previous costs x_1, x_2, \dots, x_{k-1} and whether there is no other queen on the same diagonal. Using place, we give a precise solution to then n- queens problem.

The solution can be shown as

Q							
			Q				
	Q						
				Q			
		Q					
					Q		
						Q	

no solution with this configuration, hence we backtrack and try again

// C++ program 8 Queen Programming

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#define N 8
using namespace std;
/* print solution */
void printSolution(int board[N][N])
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
            cout<<board[i][j]<<" ";
        cout<<endl;
    }
}
```



```
    }  
}  
/* check if a queen can be placed on board[row][col]*/  
bool isSafe(int board[N][N], int row, int col)  
{  
    int i, j;  
    for (i = 0; i < col; i++)  
    {  
        if (board[row][i])  
            return false;  
    }  
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)  
    {  
        if (board[i][j])  
            return false;  
    }  
    for (i = row, j = col; j >= 0 && i < N; i++, j--)  
    {  
        if (board[i][j])  
            return false;  
    }  
    return true;  
}  
/*solve N Queen problem */  
bool solveNQUtil(int board[N][N], int col)  
{  
    if (col >= N)  
        return true;  
    for (int i = 0; i < N; i++)  
    {  
        if ( isSafe(board, i, col) )  
        {  
            board[i][col] = 1;  
            if (solveNQUtil(board, col + 1) == true)
```

```
        return true;
        board[i][col] = 0;
    }
}
return false;
}
/* solves the N Queen problem using Backtracking.*/
bool solveNQ()
{
    int board[N][N] = {0};
    if (solveNQUtil(board, 0) == false)
    {
        cout<<"Solution does not exist"<<endl;
        return false;
    }
    printSolution(board);
    return true;
}
// Main
int main()
{
    solveNQ();
    return 0;
}
```

Time Complexity Analysis

- the isPossible method takes $O(n)$ time
- for each invocation of loop in nQueen Helper, it runs for $O(n)$ time
- the is Possible condition is present in the loop and also calls n Queen Helper which is recursive

adding this up, the recurrence relation is:

$$T(n) = O(n^2) + n * T(n-1)$$

solving the above recurrence by iteration or recursion tree, the time complexity of the nQueen problem is $= O(N!)$

Facilities:

Linux Operating Systems, G++

Algorithm:**START**

1. begin from the leftmost column
2. if all the queens are placed,
return true/ print configuration
3. check for all rows in the current column
 - a) if queen placed safely, mark row and column; and recursively check if we approach in the current configuration, do we obtain a solution or not
 - b) if placing yields a solution, return true
 - c) if placing does not yield a solution, unmark and try other rows
4. if all rows tried and solution not obtained, return false and backtrack

END

Input: The size of a chess board. Generally, it is 8. as (8 x 8 is the size of a chess board)

Output: The matrix that represents in which row and column the N Queens can be placed. If the solution does not exist, it will return false.

```
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
```

In this output, the value 1 indicates the correct place for the queens.

Conclusion: Hence, we have successfully studied concept of 8-Queens matrix having first Queen placed Using backtracking.

Questions:

1. How do you solve 8 queen problem using backtracking?
2. What is 8 queen problem with example?
3. Which algorithm is used to solve 8 queen's problem?
4. How do you solve 8 queen's problem backtracking?
5. What is 8 queen problem explain with algorithm in detail in DAA?

Assignment No.	7 (GROUP A)
Title	Write a program for analysis of quick sort by using deterministic and randomized variant.
Subject	Laboratory Practice-III
Class	
Roll No.	
Date	
Signature	

Assignment No: 7

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

Problem Statement: Write a function should recursively sort the subarray using deterministic and randomized variant.

Prerequisites: Design and analysis algorithm

Objective: Implement deterministic Quick sort.

Outcome: To understand the implementation of quick sort by using deterministic and randomized variant.

Theory:**Deterministic Quicksort**

If Deterministic Quicksort is applied to an array of length n whose entries are already sorted, then this algorithm uses $\Omega(n^2)$ steps.

It comes down to how the pivots are chosen. In a deterministic quicksort, the pivots are chosen by either always choosing the pivot at the same relative index such as the first, last, or middle element or by using the median of any number of predetermined element choices. For instance, a common method is to choose the median of first, last, and middle elements as the pivot. Even with the median-of-3 method I just described, certain datasets can easily give $O(N^2)$ time complexity. An example dataset is the so-called organ pipes set of data:

array = [1,2,3,4,5,6,7,8,9,10,9,8,7,6,5,4,3,2,1]

Partition the array, then recursively sort the pieces before and after the pivot element.

Quicksort(A,p,r)

if $p < r$ then

q := Partition(A,p,r); // rearrange A[p..r] in place

Quicksort(A, p,q-1);

Quicksort(A,p+1,r);

Divide and-Conquer

The design of Quicksort is based on the divide-and-conquer paradigm.

a) Divide: Partition the array A[p..r] into two (possibly empty) subarrays A[p..q-1] and A[q+1,r]

such that

- $A[x] \leq A[q]$ for all x in $[p..q-1]$

- $A[x] > A[q]$ for all x in $[q+1..r]$

b) Conquer: Recursively sort $A[p..q-1]$ and $A[q+1..r]$

c) Combine: nothing to do here

Randomized Quicksort

We expect good average case behavior if all input permutations are equally likely, but what if it is not? To get better performance on sorted or nearly sorted data -- and to foil our adversary! -- we can randomize the algorithm to get the same effect as if the input data were random.

Instead of explicitly permuting the input data (which is expensive), randomization can be accomplished trivially by **random sampling** of one of the array elements as the pivot. If we swap the selected item with the last element, the existing PARTITION procedure applies:

Algorithm:

```
partition(arr[], lo, hi)
```

```
    pivot = arr[hi]
```

```
    i = lo    // place for swapping
```

```
    for j := lo to hi - 1 do
```

```
        if arr[j] <= pivot then
```

```
            swap arr[i] with arr[j]
```

```
            i = i + 1
```

```
    swap arr[i] with arr[hi]
```

```
    return i
```

```
partition_r(arr[], lo, hi)
```

```
    r = Random Number from lo to hi
```

```
    Swap arr[r] and arr[hi]
```

```
    return partition(arr, lo, hi)
```

```
quicksort(arr[], lo, hi)
```

```
    if lo < hi
```

```
        p = partition_r(arr, lo, hi)
```

```
        quicksort(arr, lo, p-1)
```

```
        quicksort(arr, p+1, hi)
```

Randomized Quicksort Analysis: The analysis assumes that all elements are unique, but with some work can be generalized to remove this assumption (Problem 7-2 in the text).

Worst Case The previous analysis was pretty convincing, but was based on an assumption about the worst case. This analysis proves that our selection of the worst case was correct, and also shows something interesting: we can solve a recurrence relation with a "max" term in it!

PARTITION produces two subproblems, totaling size $n-1$. Suppose the partition takes place at index q . The recurrence for the worst case always selects the maximum cost among all possible ways of splitting the array (i.e., it always picks the worst possible q):

$$T(n) = \max_{0 \leq q \leq n-1} (T(q) + T(n-q-1)) + \Theta(n)$$

Based on the informal analysis, we guess $T(n) \leq cn^2$ for some c . Substitute this guess into the recurrence:

$$\begin{aligned} T(n) &\leq \max_{0 \leq q \leq n-1} (cq^2 + c(n-q-1)^2) + \Theta(n) \\ &= c \cdot \max_{0 \leq q \leq n-1} (q^2 + (n-q-1)^2) + \Theta(n) \end{aligned}$$

The maximum value of $q^2 + (n-q-1)^2$ occurs when q is either 0 or $n-1$ (the second derivative is positive), and has value $(n-1)^2$ in either case:

$$\begin{aligned} \max_{0 \leq q \leq n-1} (q^2 + (n-q-1)^2) &\leq (n-1)^2 \\ &= n^2 - 2n + 1 \end{aligned}$$

Substituting this back into the recurrence:

$$\begin{aligned} T(n) &\leq cn^2 - c(2n-1) + \Theta(n) \\ &\leq cn^2, \end{aligned}$$

We can pick c so that $c(2n-1)$ dominates $\Theta(n)$. Therefore, the worst case running time is $O(n^2)$. One can also show that the recurrence is $\Omega(n^2)$, so worst case is $\Theta(n^2)$.

//Sample Example

//C++ program implementation of Quicksort.

```
#include <bits/stdc++.h>
using namespace std;
//rearrange the elements to get the actual pivot index
int partition(int arr[], int low, int high, int pivot){
    int PIndex = low;
    for(int i=low;i<=high;i++) {
        if(arr[i]<=pivot) {
            swap(arr[PIndex],arr[i]);
            PIndex++;
        }
    }
}
```



```
    }  
  }  
  PIndex--;  
  return PIndex;  
}  
void quickSort(int arr[], int low, int high){  
    if(low < high) {  
        int pivot = arr[high];  
        int PIndex = partition(arr, low, high, pivot);  
        quickSort(arr, low, PIndex-1);  
        quickSort(arr, PIndex+1, high);  
    }  
}  
int main()  
{  
    int arr[7]={6,3,9,5,2,8,7};  
    int n=7;  
    quickSort(arr, 0 , n-1);  
    cout<<"The sorted array is: ";  
    for( int i = 0 ; i < n; i++){  
        cout<< arr[i]<<" ";  
    }  
    return 0;  
}
```

Output:

The sorted array is: 2 3 5 6 7 8 9

Facilities:

Linux Operating Systems, G++

Time Complexity:

If all elements of the given array are same then that is the worst case for the randomized quicksort. And time complexity of worst case of **quicksort is $O(n^2)$** that is proven already

Input:

Before Sorting

4 2 8 3 1 5 7 11 6

Output:

After Sorting

1 2 3 4 5 6 7 8 11

Conclusion: Hence, we have studied and implement of analysis of quick sort by using deterministic and randomized variant.

Questions:

1. What are randomized algorithms write and explain randomized quicksort?
2. What is a the analysis of randomized quick sort?
3. What is the difference between quick sort and randomized quick sort?
4. What is meant by randomized algorithm?
5. What is the difference between quick sort and randomized quick sort analyze the running time complexity of randomized quick sort for worst case?

Assignment No.	8 (GROUP A) (Mini Project)
Title	<p>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.</p> <p>OR</p> <p>Implement merge sort and multithreaded merge sort. Compare time required by both the algorithms. Also analyze the performance of each algorithm for the best case and the worst case.</p> <p>OR</p> <p>Implement the Naive string matching algorithm and Rabin-Karp algorithm for string matching. Observe difference in working of both the algorithms for the same input.</p> <p>OR</p> <p>Develop a application for signature identification by creating your own dataset of your college student.</p>
Subject	
Class	
Roll No.	
Date	
Signature	

GROUP B: MACHINE LEARNING

Assignment No.	1 (GROUP B)
Title	<p>Predict the price of the Uber ride from a given pickup point to the agreed drop-off location. Perform following tasks:</p> <ol style="list-style-type: none">1. Pre-process the dataset.2. Identify outliers.3. Check the correlation.4. Implement linear regression and random forest regression models. <p>Evaluate the models and compare their respective scores like R², RMSE, etc. Dataset link: https://www.kaggle.com/datasets/yasserh/uber-fares-dataset</p>
Subject	Laboratory Practice-III
Class	
Roll No.	
Date	
Signature	

Assignment No: 1

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

Problem Statement: Develop a Python program to implement Predict the price of the Uber ride from a given pickup point to the agreed drop-off location.

Prerequisites: LR and Random Forest Algorithm From DSBDA

Objectives

Understand the Dataset & cleanup (if required).

Build Regression models to predict the fare price of uber ride.

Also evaluate the models & compare their respective scores like R2, RMSE, etc.

Theory:

The Uber Datasets

We will perform data analysis on two types of rider data from Uber. The first dataset contains information about the rides taken by one particular user, and the second contains similar details about the rides taken by Uber users in two cities.

Uber Personal Data

[This Kaggle Uber dataset](#) contains information about 1155 rides of a single Uber user. The features include the trip date, source, destination, distance traveled, and purpose of the trip.

This [dataset](#) is a good starting point for performing basic EDA. Based on this, we might also be able to generate some insights by relating the data to real-world events and user habits.

Algorithm:

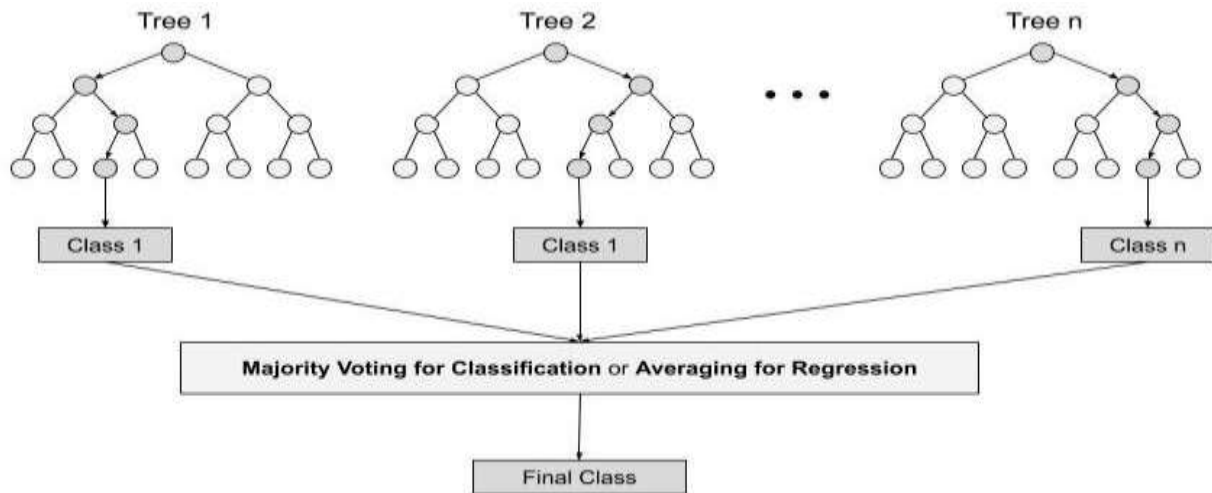
Steps involved in random forest algorithm:

Step 1: In Random forest n number of random records are taken from the data set having k number of records.

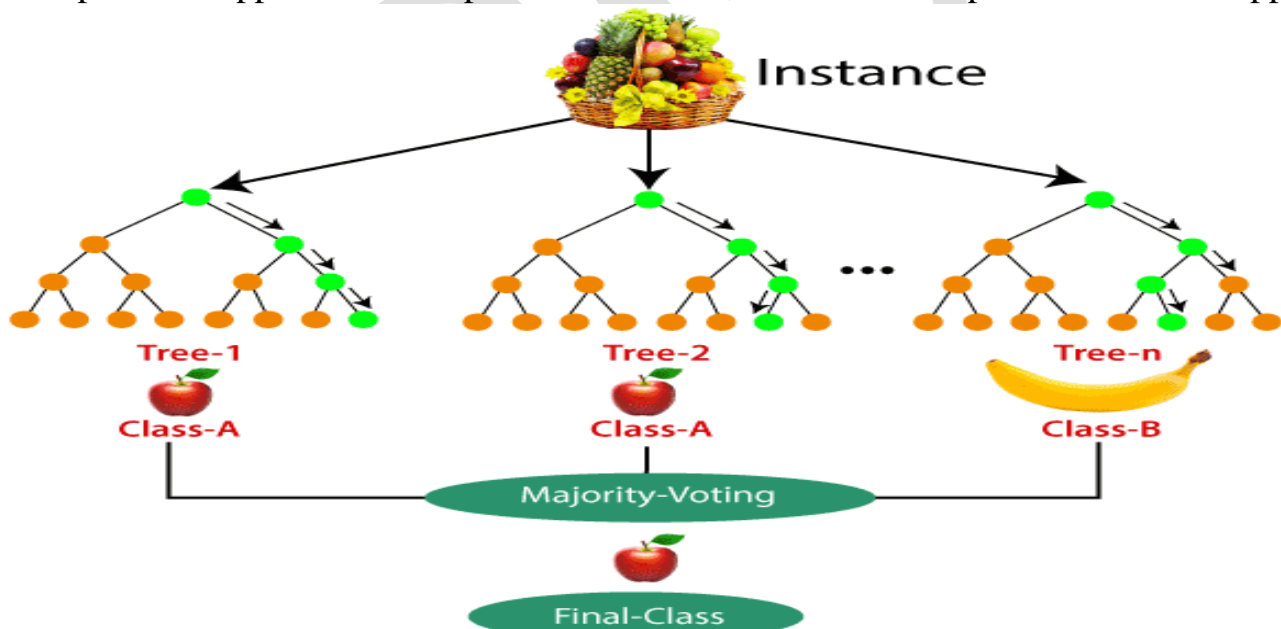
Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.



For example: consider the fruit basket as the data as shown in the figure below. Now n number of samples are taken from the fruit basket and an individual decision tree is constructed for each sample. Each decision tree will generate an output as shown in the figure. The final output is considered based on majority voting. In the below figure you can see that the majority decision tree gives output as an apple when compared to a banana, so the final output is taken as an apple.



Important Features of Random Forest

1. Diversity- Not all attributes/variables/features are considered while making an individual tree, each tree is different.

2. Immune to the curse of dimensionality- Since each tree does not consider all the features, the feature space is reduced.
3. Parallelization-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.
4. Train-Test split- In a random forest we don't have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
5. Stability- Stability arises because the result is based on majority voting/ averaging.

In this Assignment, 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.

We aim to solve the problem statement by creating a plan of action, Here are some of the necessary steps:

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>

Exploratory Data Analysis on Uber Data

We will use the data of a single Uber user for the uploaded on Kaggle . First, we import the necessary Python libraries.

```
1 import pandas as pd
2 import numpy as np
3 import datetime
4 import matplotlib
5 import matplotlib.pyplot as plt
6 import seaborn as sns
7 import calendar
```

Guidelines:-

- 1) #Importing the required libraries
import pandas as pd


```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
2)#importing the dataset
df = pd.read_csv("uber.csv")
```

1) Pre-process the dataset.

```
df.head()
df.info() #To get the required information of the dataset
df.columns #TO get number of columns in the dataset
df = df.drop(['Unnamed: 0', 'key'], axis= 1) #To drop unnamed column as it isn't required
df.head()
df.shape #To get the total (Rows,Columns)
df.dtypes #To get the type of each column
df.info()
df.describe() #To get statistics of each columns
```

2)Filling Missing values

```
df.isnull().sum()
df['dropoff_latitude'].fillna(value=df['dropoff_latitude'].mean(),inplace = True)
df['dropoff_longitude'].fillna(value=df['dropoff_longitude'].median(),inplace = True)
df.isnull().sum()
df.dtypes
```

3) Column pickup_datetime is in wrong format (Object). Convert it to DateTime Format

```
df.pickup_datetime = pd.to_datetime(df.pickup_datetime, errors='coerce')
df.dtypes
```

4)To segregate each time of date and time

```
df= df.assign(hour = df.pickup_datetime.dt.hour,
              day= df.pickup_datetime.dt.day,
              month = df.pickup_datetime.dt.month,
              year = df.pickup_datetime.dt.year,
              dayofweek = df.pickup_datetime.dt.dayofweek)
df.head()
```

```
# drop the column 'pickup_daetime' using drop()
# 'axis = 1' drops the specified column
```

```
df = df.drop('pickup_datetime',axis=1)
df.head()
df.dtypes
```

5) Checking outliers and filling them

```
df.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15,20)) #Boxplot to check the outliers
```

#Using the InterQuartile Range to fill the values

#Using the InterQuartile Range to fill the values

```
def remove_outlier(df1 , col):
    Q1 = df1[col].quantile(0.25)
    Q3 = df1[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_whisker = Q1-1.5*IQR
    upper_whisker = Q3+1.5*IQR
    df[col] = np.clip(df1[col] , lower_whisker , upper_whisker)
    return df1
```

```
def treat_outliers_all(df1 , col_list):
```

```
    for c in col_list:
```

```
        df1 = remove_outlier(df , c)
```

```
    return df1
```

```
df = treat_outliers_all(df , df.iloc[:, 0::])
```

```
df.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15,20)) #Boxplot shows that dataset is free from outliers
```

Distance Calculation

#pip install haversine

import haversine as hs #Calculate the distance using Haversine to calculate the distance between to points. Can't use Eucladian as it is for flat surface.

```
travel_dist = []
```

```
for pos in range(len(df['pickup_longitude'])):
```

```
    long1,lati1,long2,lati2 = [df['pickup_longitude'][pos],df['pickup_latitude'][pos],df['dropoff_longitude'][pos],df['dropoff_latitude'][pos]]
```

```
    loc1=(lati1,long1)
```

```
    loc2=(lati2,long2)
```

```
    c = hs.haversine(loc1,loc2)
```

```
travel_dist.append(c)
print(travel_dist)
df['dist_travel_km'] = travel_dist
df.head()

#Uber doesn't travel over 130 kms so minimize the distance
df= df.loc[(df.dist_travel_km >= 1) | (df.dist_travel_km <= 130)]
print("Remaining observastions in the dataset:", df.shape)

#Finding inccorect latitude (Less than or greater than 90) and longitude (greater than or less than
180) incorrect_coordinates = df.loc[(df.pickup_latitude > 90) |(df.pickup_latitude < -90) |
(df.dropoff_latitude > 90) |(df.dropoff_latitude < -90) |
(df.pickup_longitude > 180) |(df.pickup_longitude < -180) |
(df.dropoff_longitude > 90) |(df.dropoff_longitude < -90)
]
df.drop(incorrect_coordinates, inplace = True, errors = 'ignore')
df.head()
df.isnull().sum()
sns.heatmap(df.isnull()) #Free for null values
corr = df.corr() #Function to find the correlation
corr
fig,axis = plt.subplots(figsize = (10,6))
sns.heatmap(df.corr(),annot = True) #Correlation Heatmap (Light values means highly correlated
)
```

6)Dividing the dataset into feature and target value

```
x = df[['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_latitude','passenger_coun
t','hour','day','month','year','dayofweek','dist_travel_km']]
y = df['fare_amount']
```

7) Dividing the dataset into training and testing dataset

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.33)
```

8)Linear Regression

```
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
```

```
regression.fit(X_train,y_train)
regression.coef_ #To find the linear coefficient
regression.intercept_ #To find the linear intercept
prediction = regression.predict(X_test) #To predict the target values
print(prediction)
y_test
```

9)Metrics Evaluation using R2, Mean Squared Error, Root Mean Squared Error

```
from sklearn.metrics import r2_score
r2_score(y_test,prediction)
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test,prediction)
MSE
RMSE = np.sqrt(MSE)
RMSE
3.156187085348032
```

10)Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100) #Here n_estimators means number of trees you
want to build before making the prediction
rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
y_pred
array([ 5.714 , 10.285 , 12.68 , ..., 6.338 , 19.4685, 7.712 ])
```

11) Metrics evaluation for Random Forest

```
R2_Random = r2_score(y_test,y_pred)
R2_Random
MSE_Random = mean_squared_error(y_test,y_pred)
MSE_Random
RMSE_Random = np.sqrt(MSE_Random)
RMSE_Random
```

Facilities:

Google Colab,Jupyter notebook

Input: <https://www.kaggle.com/datasets/yasserh/uber-fares-dataset> uber dataset file in .csv format

Output:

Confusion Matrix showing Final Results of accuracy comparison

Conclusion:

Thus we have studied uber fare price data analysis for LR and Random Forest Successfully.

Questions:

- i. Compare the accuracy for RF and LR
- ii. Justify is the accuracy difference between both the models.

Assignment No.	2 (GROUP B)
Title	Classify the email using the binary classification method. Email Spam detection has two states: a) Normal State – Not Spam, b) Abnormal State – Spam. Use K-Nearest Neighbors and Support Vector Machine for classification. Analyze their performance.
Subject	
Class	
Roll No.	
Date	
Signature	

Assignment No: 2

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

Problem Statement: Develop a Python program to implement Classify the email using the binary classification method.

Prerequisites: SVM,K-Nearest algorithm

Objectives:

Understand the Dataset & cleanup (if required).

Build models to predict the class as spam/not spam

Also evaluate the models & compare their respective scores like R2, RMSE, etc.

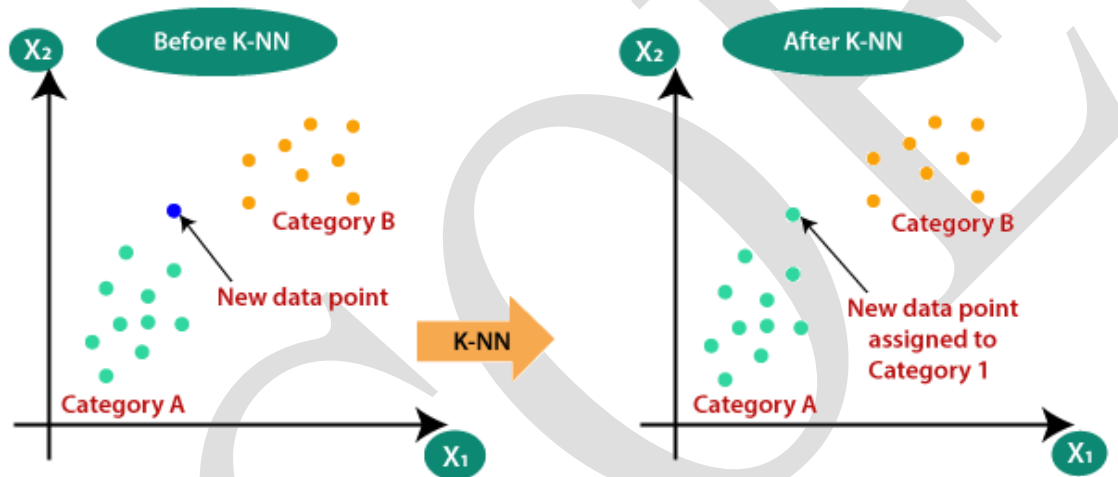
Theory:

- 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.

Why do we need a K-NN Algorithm?

- Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x_1 , so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

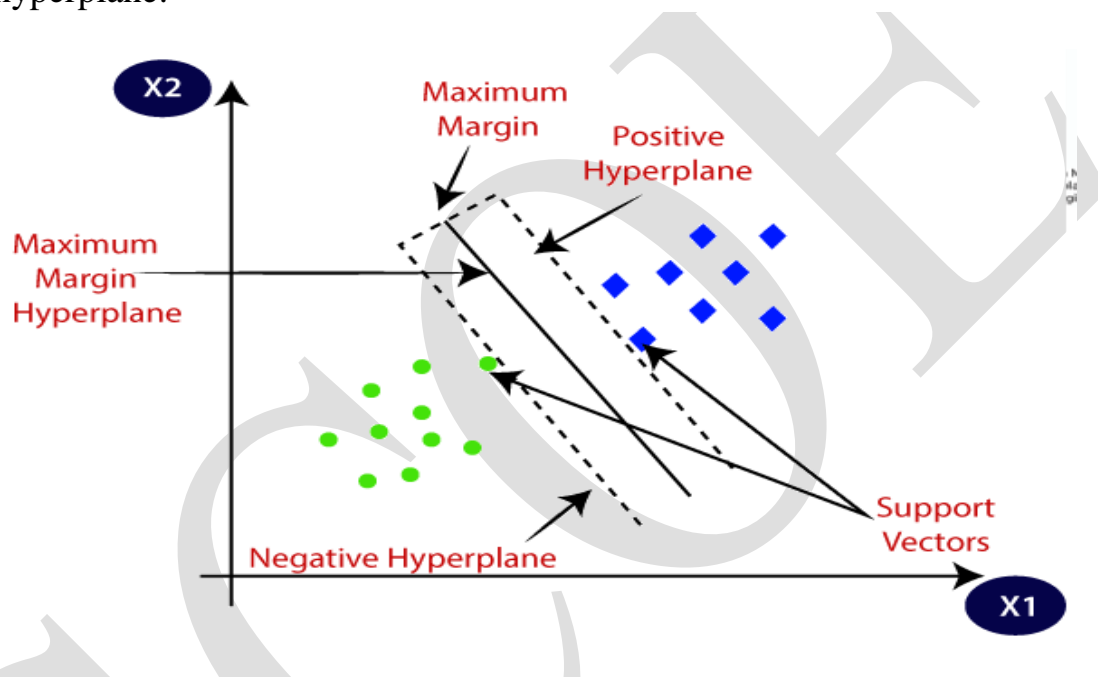
Step-6: Our model is ready.

Support Vector Machine Algorithm

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 a decision boundary or hyperplane:



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.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane. We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Algorithm:

Steps to implement the K-NN /SVM algorithm:

- Data Pre-processing step
- Fitting the K-NN algorithm/SVM to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

Sample Code

1)Preprocessing removing null values if any and Training testing split

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn import metrics
df=pd.read_csv('emails.csv')
df.head()
df.columns
```

```
df.isnull().sum()
df.dropna(inplace = True)
df.drop(['Email No.'],axis=1,inplace=True)
X = df.drop(['Prediction'],axis = 1)
y = df['Prediction']
from sklearn.preprocessing import scale
X = scale(X)
# split into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

2)Apply KNN Classifier

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7)
```

```
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Prediction",y_pred)
print("KNN accuracy = ",metrics.accuracy_score(y_test,y_pred))
print("Confusion matrix",metrics.confusion_matrix(y_test,y_pred))
```

3)Apply SVM Classifier

```
# cost C = 1
model = SVC(C = 1)
# fit
model.fit(X_train, y_train)
# predict
y_pred = model.predict(X_test)
metrics.confusion_matrix(y_true=y_test, y_pred=y_pred)
print("SVM accuracy = ",metrics.accuracy_score(y_test,y_pred))
```

Facilities:

Google colab,Jupyter notebook

Input:

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

Email dataset in csv File Format

Output:

- 1)Data preprocessing visuals if any
- 2)Confusion matrix of accuracy

Conclusion: Thus we have studied KNN and SVM algorithm successfully.

Questions:

- 1.What do you mean by KNN algorithm?
- 2.What do you mean by SVM.
- 3.Compare both the algorithms and justify the accuracy?

Assignment No.	3 (GROUP B)
Title	Given a bank customer, build a neural network-based classifier that can determine whether they will leave or not in the next 6 months.
Subject	
Class	
Roll No.	
Date	
Signature	

Assignment No: 3

Title: Given a bank customer, build a neural network-based classifier that can determine whether they will leave or not in the next 6 months.

Problem Statement: Develop a Python program to implement bank customer, build a neural network-based classifier that can determine whether they will leave or not in the next 6 months.

Prerequisites: Neural Networks

Objectives: Understand the Dataset & cleanup (if required).
Build models to predict whether employee will leave or not
Also evaluate the models & compare their respective scores like R2, RMSE, etc

Theory:

Neural Network:

The main purpose of a neural network is to try to find the relationship between features in a data set., and it consists of a set of algorithms that mimic the work of the human brain. A “neuron” in a neural network is a mathematical function that collects and classifies information according to a specific architecture.

Classification:

Classification problem involves predicting if something belongs to one class or not. In other words, while doing it we try to see something is one thing or another.

Types of Classification

- Suppose that you want to predict if a person has diabetes or not. If you are facing this kind of situation, there are two possibilities, right? That is called **Binary Classification**.
- Suppose that you want to identify if a photo is of a toy, a person, or a cat, right? this is called **Multi-class Classification** because there are more than two options.
- Suppose you want to decide that which categories should be assigned to an article. If so, it is called **Multi-label Classification**, because one article could have more than one category assigned. Let's take our explanation through this article. We may assign categories like “Deep Learning, Tensor Flow, Classification” etc. to this article

Algorithm:

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

Steps to implement**1) Importing header files and csv file**

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt #Importing the libraries
df = pd.read_csv("Churn_Modelling.csv")
```

2) Preprocessing:

```
df.head()
df.shape
df.describe()
df.isnull()
df.isnull().sum()
df.info()
df.dtypes
df.columns
df = df.drop(['RowNumber', 'Surname', 'CustomerId'], axis= 1) #Dropping the unnecessary columns
df.head()
```

3) Visualization:

```
def visualization(x, y, xlabel):
    plt.figure(figsize=(10,5))
    plt.hist([x, y], color=['red', 'green'], label = ['exit', 'not_exit'])
    plt.xlabel(xlabel, fontsize=20)
    plt.ylabel("No. of customers", fontsize=20)
    plt.legend()
df_churn_exited = df[df['Exited']==1]['Tenure']
df_churn_not_exited = df[df['Exited']==0]['Tenure']
```

```
visualization(df_churn_exited, df_churn_not_exited, "Tenure")
df_churn_exited2 = df[df['Exited']==1]['Age']
df_churn_not_exited2 = df[df['Exited']==0]['Age']
visualization(df_churn_exited2, df_churn_not_exited2, "Age")
```

4)Converting the Categorical Variables:

```
X = df[['CreditScore','Gender','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary']]
states = pd.get_dummies(df['Geography'],drop_first = True)
gender = pd.get_dummies(df['Gender'],drop_first = True)
df = pd.concat([df,gender,states], axis = 1)
```

5) Splitting the training and testing Dataset

```
df.head()
X = df[['CreditScore','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary','Male','Germany','Spain']]
y = df['Exited']
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.30)
```

6) Normalizing the values with mean as 0 and Standard Deviation as 1

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_train
X_test
```

7)Building the Classifier Model using Keras

```
import keras #Keras is the wrapper on the top of tensorflow
#Can use Tensorflow as well but won't be able to understand the errors initially.
from keras.models import Sequential #To create sequential neural network
from keras.layers import Dense #To create hidden layers
classifier = Sequential()
#To add the layers
#Dense helps to construct the neurons
#Input Dimension means we have 11 features
# Units is to create the hidden layers
```



```
#Uniform helps to distribute the weight uniformly
classifier.add(Dense(activation = "relu",input_dim = 11,units = 6,kernel_initializer = "uniform"))
classifier.add(Dense(activation = "relu",units = 6,kernel_initializer = "uniform")) #Adding second hidden layers
classifier.add(Dense(activation = "sigmoid",units = 1,kernel_initializer = "uniform")) #Final neuron will be having sigmoid function
classifier.compile(optimizer="adam",loss = 'binary_crossentropy',metrics = ['accuracy']) #To compile the Artificial Neural Network. Used Binary crossentropy as we just have only two output
classifier.summary() #3 layers created. 6 neurons in 1st,6neurons in 2nd layer and 1 neuron in last
classifier.fit(X_train,y_train,batch_size=10,epochs=50) #Fitting the ANN to training dataset
y_pred =classifier.predict(X_test)
y_pred = (y_pred > 0.5) #Predicting the result
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm = confusion_matrix(y_test,y_pred)
cm
accuracy = accuracy_score(y_test,y_pred)
accuracy
plt.figure(figsize = (10,7))
sns.heatmap(cm,annot = True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
print(classification_report(y_test,y_pred))
```

Facilities:

GoogleColab,Jupyter notebook

Input: <https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling> .csv file containing bank employee data

Output:

- 1)Data preprocessing visuals if any
- 2)Confusion matrix of accuracy

Conclusion: Thus we have studied Neural Networks successfully.

Questions:

1. Explain neural Networks
2. How this model will work if go with tensor flow
3. Why neural networks for this kind of data justify

Assignment No.	4
Title	Implement Gradient Descent Algorithm to find the local minima of a function
Subject	Laboratory Practice-III
Class	B.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No: 4

Title: Implement Gradient Descent Algorithm.

Problem Statement: Develop a program in Python to create a Gradient Descent.

Prerequisites: Linear Regression, Logistic Regression, Neural Network.

Objectives: Implement Gradient Descent Algorithm to find the local minima of a function

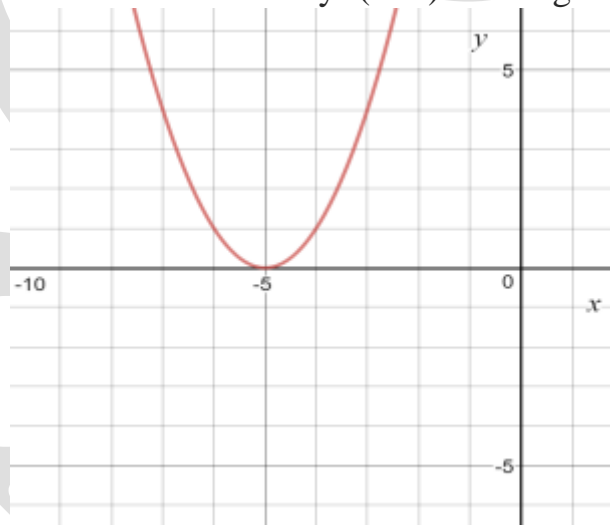
Outcome: To understand the implementation of Gradient Descent Algorithm to find the local minima of a function

Theory:

It is an optimization algorithm to find the minimum of a function. We start with a random point on the function and move in the **negative direction** of the **gradient of the function** to reach the **local/global minima**.

Example by hand :

Question : Find the local minima of the function $y=(x+5)^2$ starting from the point $x=3$



Solution : We know the answer just by looking at the graph. $y = (x+5)^2$ reaches its minimum value when $x = -5$ (i.e. when $x=-5$, $y=0$). Hence $x=-5$ is the local and global minima of the function.

Now, let's see how to obtain the same numerically using gradient descent.

Step 1 : Initialize $x = 3$. Then, find the gradient of the function, $dy/dx = 2*(x+5)$.

Step 2 : Move in the direction of the negative of the gradient ([Why?](#)). But wait, how much to move? For that, we require a learning rate. Let us assume the **learning rate** $\rightarrow 0.01$

Step 3 : Let's perform 2 iterations of gradient descent

Step 4 : We can observe that the X value is slowly decreasing and should converge to -5 (the local minima). However, how many iterations should we perform?

Let us set a precision variable in our algorithm which calculates the difference between two consecutive "x" values. If the difference between x values from 2 consecutive iterations is lesser than the precision we set, stop the algorithm !

Initialize Parameters :

$$X_0 = 3$$

$$\text{Learning rate} = 0.01$$

$$\frac{dy}{dx} = \frac{d}{dx}(x+5)^2 = 2*(x+5)$$

Iteration 1 :

$$X_1 = X_0 - (\text{learning rate}) * \left(\frac{dy}{dx}\right)$$

$$X_1 = 3 - (0.01) * (2 * (3 + 5)) = 2.84$$

Iteration 2 :

$$X_2 = X_1 - (\text{learning rate}) * \left(\frac{dy}{dx}\right)$$

$$X_2 = 2.84 - (0.01) * (2 * (2.84 + 5)) = 2.6832$$

Step 4 : We can observe that the X value is slowly decreasing and should converge to -5 (the local minima). However, how many iterations should we perform?

Let us set a precision variable in our algorithm which calculates the difference between two consecutive “x” values . If the difference between x values from 2 consecutive iterations is lesser than the precision we set, stop the algorithm !

Gradient descent in Python :

Step 1 : Initialize parameters

```
cur_x = 3 # The algorithm starts at x=3
rate = 0.01 # Learning rate
precision = 0.000001 #This tells us when to stop the algorithm
previous_step_size = 1 #
max_iters = 10000 # maximum number of iterations
iters = 0 #iteration counter
df = lambda x: 2*(x+5) #Gradient of our function
```

Step 2 : Run a loop to perform gradient descent :

i. Stop loop when difference between x values from 2 consecutive iterations is less than 0.000001 or when number of iterations exceeds 10,000

```
while previous_step_size > precision and iters < max_iters:
    prev_x = cur_x #Store current x value in prev_x
    cur_x = cur_x - rate * df(prev_x) #Grad descent
    previous_step_size = abs(cur_x - prev_x) #Change in x
    iters = iters+1 #iteration count
    print("Iteration",iters,"\nX value is",cur_x) #Print iterations
    print("The local minimum occurs at", cur_x)
```

Facilities:

GoogleColab,Jupyter notebook

Input:

To find the local minima of the given function from its starting point.

Output:

Finds the optimal solution by taking a step in the direction of the maximum rate of decrease of the function.

Conclusion:

Hence, we have successfully studied implementation of Gradient Descent Algorithm successfully.

Questions:

1. Find the local minima of the function $y=(x+3)^2$ starting from the point $x=2$.
2. What is the idea behind Gradient Descent?
3. Compare the Batch Gradient Descent and Stochastic Gradient Descent.
4. What are the types of Gradient Descent?

Assignment No.	5
Title	Implement K-Nearest Neighbors algorithm
Subject	Laboratory Practice-III
Class	B.E. (C.E.)
Roll No.	
Date	
Signature	

Assignment No: 5

Title: Implement K-Nearest Neighbors algorithm

Problem Statement: Develop a Python program to implement K-Nearest Neighbors algorithm

Prerequisites: Supervised Machine Learning, Confusion Matrix

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

Outcome: To understand the implementation of K-Nearest Neighbors algorithm and analyze and find its steps.

Theory

k Nearest Neighbours algorithm

In machine learning, k Nearest Neighbours or kNN is the simplest of all machine learning algorithms. It is a non-parametric algorithm used for classification and regression tasks. Non-parametric means there is no assumption required for data distribution. So, kNN does not require any underlying assumption to be made. In both classification and regression tasks, the input consists of the k closest training examples in the feature space. The output depends upon whether kNN is used for classification or regression purposes.

- In kNN classification, the output is a class membership. The given data point is classified based on the majority of type of its neighbours. The data point is assigned to the most frequent class among its k nearest neighbours. Usually k is a small positive integer. If k=1, then the data point is simply assigned to the class of that single nearest neighbour.
- In kNN regression, the output is simply some property value for the object. This value is the average of the values of k nearest neighbours.

kNN is a type of instance-based learning or lazy learning. Lazy learning means it does not require any training data points for model generation. All training data will be used in the testing phase. This makes training faster and testing slower and costlier. So, the testing phase requires more time and memory resources.

In kNN, the neighbours are taken from a set of objects for which the class or the object property value is known. This can be thought of as the training set for the kNN algorithm, though no explicit training step is required. In both classification and regression kNN algorithm, we can assign weight to the contributions of the neighbours. So, nearest neighbours contribute more to the average than the more distant ones.

k Nearest Neighbours intuition

The kNN algorithm intuition is very simple to understand. It simply calculates the distance between a sample data point and all the other training data points. The distance can be Euclidean distance or Manhattan distance. Then, it selects the k nearest data points where k can be any integer. Finally, it assigns the sample data point to the class to which the majority of the k data points belong.

Now, we will see kNN algorithm in action. Suppose, we have a dataset with two variables which are classified as Red and Blue.

In kNN algorithm, k is the number of nearest neighbours. Generally, k is an odd number because it helps to decide the majority of the class. When $k=1$, then the algorithm is known as the nearest neighbour algorithm.

Now, we want to classify a new data point X into Blue class or Red class. Suppose the value of k is 3. The kNN algorithm starts by calculating the distance between X and all the other data points. It then finds the 3 nearest points with least distance to point X.

In the final step of the kNN algorithm, we assign the new data point X to the majority of the class of the 3 nearest points. If 2 of the 3 nearest points belong to the class Red while 1 belong to the class Blue, then we classify the new data point as Red.

How to decide the number of neighbours in kNN

While building the kNN classifier model, one question that come to my mind is what should be the value of nearest neighbours (k) that yields highest accuracy. This is a very important question because the classification accuracy depends upon our choice of k.

The number of neighbours (k) in kNN is a parameter that we need to select at the time of model building. Selecting the optimal value of k in kNN is the most critical problem. A small value of k means that noise will have higher influence on the result. So, probability of overfitting is very high. A large value of k makes it computationally expensive in terms of time to build the kNN model. Also, a large value of k will have a smoother decision boundary which means lower variance but higher bias.

The data scientists choose an odd value of k if the number of classes is even. We can apply the elbow method to select the value of k. To optimize the results, we can use Cross Validation technique. Using the cross-validation technique, we can test the kNN algorithm with different values of k. The model which gives good accuracy can be considered to be an optimal choice. It depends on individual cases and at times best process is to run through each possible value of k and test our result.

Example:

#Loading the dataset

```
diabetes_data = pd.read_csv('./input/diabetes.csv')
```

#Print the first 5 rows of the dataframe.

```
diabetes_data.head()
```

Out[2]:

	Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Confusion Matrix

The confusion matrix is a technique used for summarizing the performance of a classification algorithm i.e. it has binary outputs.

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

```
#import confusion_matrix
from sklearn.metrics import confusion_matrix
#let us get the predictions using the classifier we had fit above
y_pred = knn.predict(X_test)
confusion_matrix(y_test,y_pred)
pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
Out[29]:
```

Predicted	0	1	All
True			
0	142	25	167
1	35	54	89
All	177	79	256

```
In [30]:
y_pred = knn.predict(X_test)
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
p = sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu", fmt='g')
```

```
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
Out[30]:
Text(0.5,12.5,'Predicted label')
```

2. Classification Report

Report which includes Precision, Recall and F1-Score.

Precision Score

TP – True Positives

FP – False Positives

Precision – Accuracy of positive predictions.

Precision = $TP / (TP + FP)$

Recall Score

FN – False Negatives

Recall(sensitivity or true positive rate): Fraction of positives that were correctly identified.

Recall = $TP / (TP + FN)$

F1 Score

F1 Score (aka F-Score or F-Measure) – A helpful metric for comparing two classifiers.

F1 Score takes into account precision and the recall.

It is created by finding the the harmonic mean of precision and recall.

$F1 = 2 \times (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Precision = $TP / TP + FP$

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes. The question recall answers is: Of all the passengers that truly survived, how many did we label? A recall greater than 0.5 is good.

Recall = $TP / TP + FN$

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the

cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

$$F1 \text{ Score} = 2(\text{Recall Precision}) / (\text{Recall} + \text{Precision})$$

In [31]:

```
#import classification_report
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

	precision	recall	f1	-score	support
0	0.80	0.85	0.83		167
1	0.68	0.61	0.64		89
micro avg	0.77	0.77	0.77		256
macro avg	0.74	0.73	0.73		256
weighted avg	0.76	0.77	0.76		256

Accuracy:

ROC (Receiver Operating Characteristic) Curve tells us about how good the model can distinguish between two things (e.g If a patient has a disease or no). Better models can accurately distinguish between the two. Whereas, a poor model will have difficulties in distinguishing between the two

```
from sklearn.metrics import roc_curve
y_pred_proba = knn.predict_proba(X_test)[:,-1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
```

In [33]:

```
plt.plot([0,1],[0,1], 'k--')
plt.plot(fpr,tpr, label='Knn')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('Knn(n_neighbors=11) ROC curve')
plt.show()
```

In [34]:

```
#Area under ROC curve
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,y_pred_proba)
```

Out[34]:

0.8193500639171096

Facilities:

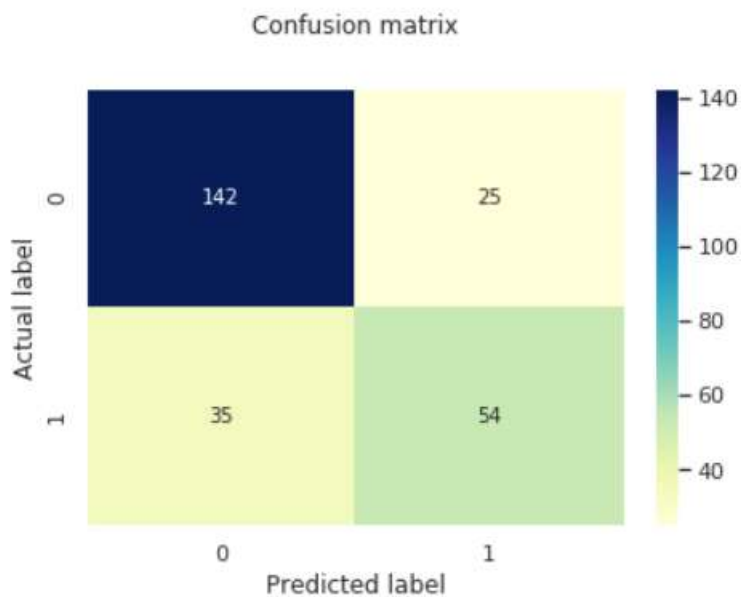
GoogleColab,Jupiter notebook

Input:

<https://www.kaggle.com/datasets/abdallamahgoub/diabetes>

Output:

- 1)Data preprocessing visuals if any
- 2)Confusion matrix of accuracy

**Conclusion:**

Hence, we have successfully studied implementation of K-NN Algorithm successfully.

Questions:

1. When do we use K-NN algorithm?
2. Would you use K-NN for large datasets?
3. What are advantages and disadvantages of K-NN algorithm?
4. Explain some cases where K-NN clustering fails to give good results

Assignment No.	6
Title	Implement K-Means clustering/ hierarchical clustering
Subject	Laboratory Practice-III
Class	B.E.(C.E.)
Roll No.	
Date	
Signature	

Assignment No: 6

Title: Implement K-Means clustering/ hierarchical clustering

Problem Statement: Develop a Python program to implement K-Means clustering/ hierarchical clustering

Prerequisites: Unsupervised Machine Learning, Clusterization.

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

Outcome: To understand the implementation of K-Means clustering/ hierarchical clustering algorithm and analyze and its steps.

Theory

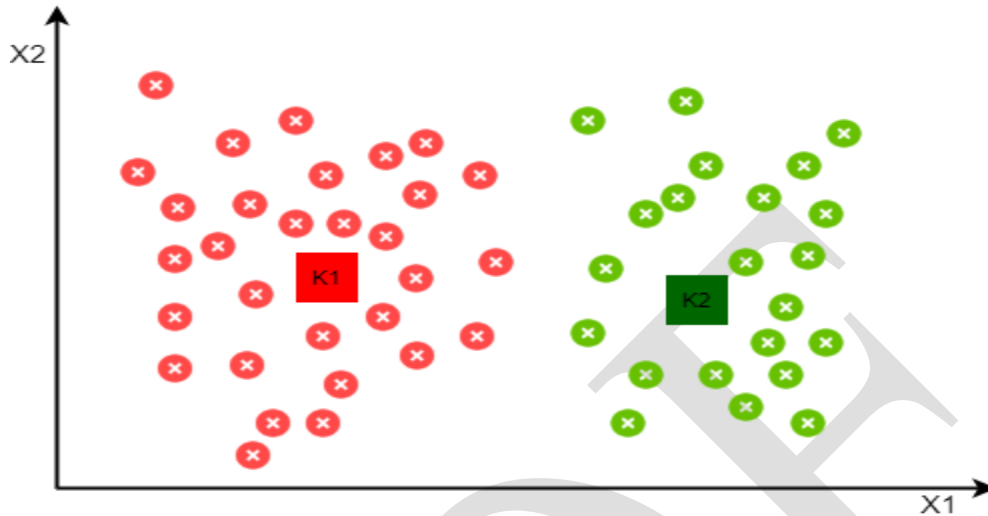
K-Means Clustering

K-Means clustering is the most popular unsupervised machine learning algorithm. K-Means clustering is used to find intrinsic groups within the unlabelled dataset and draw inferences from them. In this kernel, I implement K-Means clustering Uses

- **Search engine:** Search engine, groups results together using clustering algorithm
- **Customer segmentation:** K-mean clustering can be used to create customer clusters based on demographic information, geographical information and behavioral data.
- **Social network analysis:** To find groups of people with specific interest to direct the personalized ads.
- **Data center:** To organize the computer clusters in data center.
- **Inventory management:** Create inventory clusters based on sales number and manufacturing capacity

Inner Working of K-Means Clustering

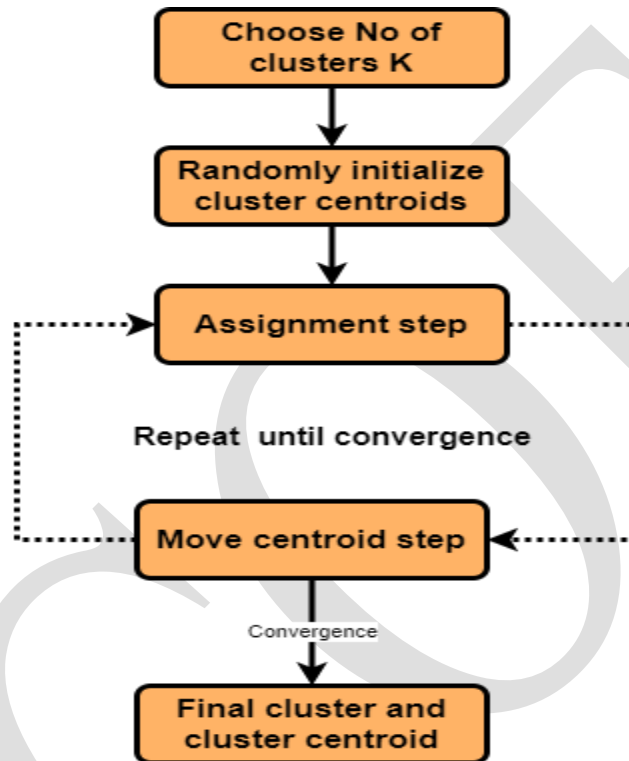
K-means is often referred to as Lloyd's algorithm. It is one of the most popular clustering algorithm. Refer below plot where there are two clusters ($K=2$) one is of red data points and another one of green data points.



So how does K-Means algorithm find the clusters of the data points without any label? Below steps will explain the inner working of K-Means algorithm.

- First step is to finalize the number of clusters you want to identify in your data. This is the "K" in K-means clustering.
- Now randomly initialize the points equal to the number of clusters K. 'Cluster Centroid' is the terminology used to refer these points.
- Note that centroid means center point of given dataset, but initially these points are at random location, but at the end when K-Means algorithm will converge they will be at the center of their respective cluster.
- Once cluster centroids are defined, K-means algorithm will go through each data point from given data and depending on that points closeness to cluster centroid, it will assign the data point to the cluster centroid. This is called as 'Assignment Step'.
- In order to move the cluster centroids from random location to their respective group, K-means algorithm will find the mean of each data point assigned to the cluster centroid and move the respective centroid to the mean value location. This is called as 'Move Centroid Step'.
- Note that during 'Move Centroid Step' data points can get reassigned from one cluster to another as centroid position change.
- Now repeat the assignment and move centroid steps till cluster centroid position don't change. K-means algorithm will converge when we get the unchanged position of cluster centroids.

- Once K-means algorithm is converged, data point assigned to respective centroid will represent the respective cluster.
- During cluster assignment step if we found a centroid who has no data point associated with it, then it's better to remove it.



Since we have to randomly pick the cluster centroids, its initialization may affect the final outcome of the clustering. In case our initialization is not correct, then K-Means algorithm may form a cluster with few points only. Such situation is referred as 'centroid random initialization trap' and it may cause algorithm to get stuck at local optima.

Check below plots, where for same dataset, we end up getting different clusters depending on initial position of cluster centroids. Gray color squares represent the initial positions of centroids and red, green and blue squares represent the final position of centroids.

Example:

```
kmeans= KMeans(n_clusters = 5, random_state = 42)
# Compute k-means clustering
kmeans.fit(X)
```

```
# Compute cluster centers and predict cluster index for each sample.
pred = kmeans.predict(X)
pred
```

Once cluster centroids are defined, K-means algorithm will go through each data point from given data and depending on that points closeness to cluster centroid, it will assign the data point to the cluster centroid. This is called as 'Assignment Step'.

In order to move the cluster centroids from random location to their respective group, K-means algorithm will find the mean of each data point assigned to the cluster centroid and move the respective centroid to the mean value location. This is called as 'Move Centroid Step'

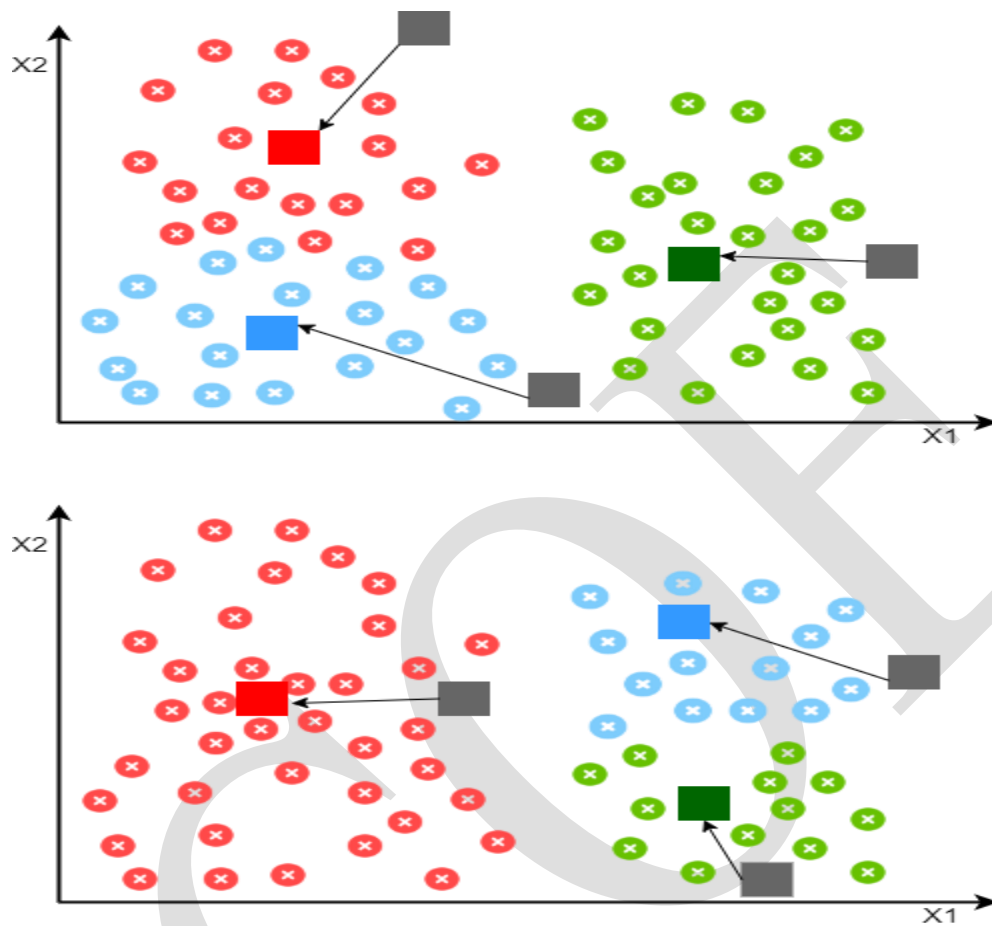
```
def plot_k_means_progress(centroid_history,n_clusters, centroid_sets, cluster_color):
    """
    This function will plot the path taken by the centroids
    I/P:
    * centroid_history: 2D array of centroids. Each element represent the centroid coordinate.
      If there are 5 clusters then first set contains initial cluster coordinates
      (i.e. first 5 elements) and then k_means loop will keep appending new cluster coordinates
      for each iteration
    * n_clusters: Total number of clusters to find
    * centroid_sets: At the start we set random values as our first centroid set. K-Means loop
      will keep adding
      new centroid sets to centroid_history. Since we are plotting the path of centroid locations, c
      centroid set value
      will be K-Means loop iteration number plus 1 for initial centroid set.
      So its value will be from 2 to K-Means loops max iter plus 1
    * cluster_color: Just to have same line and cluster color
    O/P: Plot the centroid path
    """
    c_x = [] # To store centroid X coordinated
    c_y=[] # To store the centroid Y coordinates
    for i in range(0, n_clusters):
        cluster_index = 0
        for j in range(0, centroid_sets):
            c_x = np.append(c_x, centroid_history[:,0][i + cluster_index])
```

```

c_y = np.append(c_y, centroid_history[:,1][i + cluster_index])
cluster_index = cluster_index + n_clusters
# if there are 5 clusters then first set contains initial cluster coordinates and then k_means loop will keep appending new cluster coordinates for each iteration
plt.plot(c_x, c_y, c= cluster_color['c_' + str(i)], linestyle='--')
# Reset coordinate arrays to avoid continuous lines
c_x = []
c_y=[]
In [10]:
linkcode
plt.figure(figsize=(10,6))

# Random Initialization of Centroids
plt.scatter(df['Annual Income (k$)'],df['Spending Score (1-100)'])
initial_centroid = np.array([[10, 2], [50,100], [130,20], [50,15], [140,100]])
plt.scatter(initial_centroid[:,0], initial_centroid[:, 1],s = 200, c = 'red', label = 'Random Centroid', marker='*')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.legend()
plt.title('Random Initialization of Centroids')
# K-Means loop of assignment and move centroid steps
centroid_history = []
centroid_history = initial_centroid
#
cluster_color= {'c_0':'brown','c_1':'green','c_2':'blue','c_3':'purple','c_4':'orange'}
n_clusters = 5
for i in range(1,6):
    kmeans= KMeans(n_clusters, init= initial_centroid, n_init= 1, max_iter= i, random_state = 42) #n_init= 1 since our init parameter is array

```



Random Initialization Guidelines

To avoid random initialization trap, follow below guidelines for random initialization.

- Number of cluster centroids should be less than number of training examples
- To avoid local optima issue, try to do multiple random initialization of centroids.
- Multiple random initialization technique is more effective when we have a small number of clusters.
- Similarly for large number of clusters, few random initialization are sufficient

Choosing The Number of Clusters

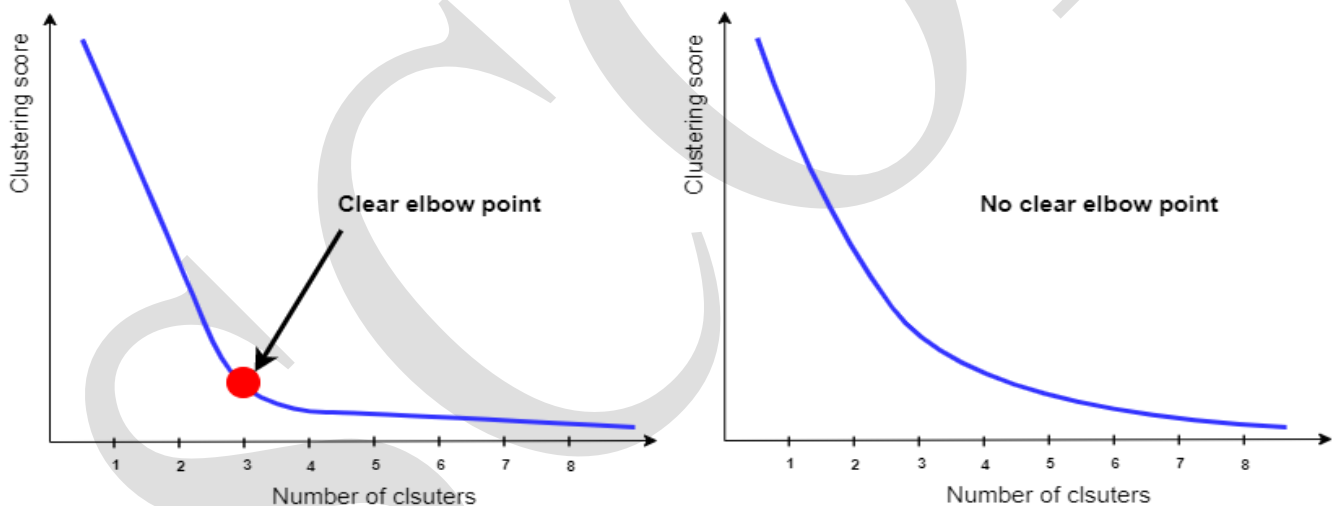
So using random initialization we can avoid the local optima issue, but to choose how many clusters to look for in a given data we can use below methods.

Visualization

To find the number of clusters manually by data visualization is one of the most common method. Domain knowledge and proper understanding of given data also help to make more informed decisions. Since its manual exercise there is always a scope for ambiguous observations, in such cases we can also use 'Elbow Method'

Elbow Method

In Elbow method we run the K-Means algorithm multiple times over a loop, with an increasing number of cluster choice(say from 1 to 10) and then plotting a clustering score as a function of the number of clusters. Clustering score is nothing but sum of squared distances of samples to their closest cluster center. Elbow is the point on the plot where clustering score (distortion) slows down, and the value of cluster at that point gives us the optimum number of clusters to have. But sometimes we don't get clear elbow point on the plot, in such cases its very hard to finalize the number of clusters.



Advantages

- One of the simplest algorithm to understand
- Since it uses simple computations it is relatively efficient
- Gives better results when there is less data overlapping

Disadvantages

- Number of clusters need to be defined by user

- Doesn't work well in case of overlapping data
- Unable to handle the noisy data and outliers
- Algorithm fails for non-linear data set to find intrinsic groups within the dataset that display the same status_type behaviour.

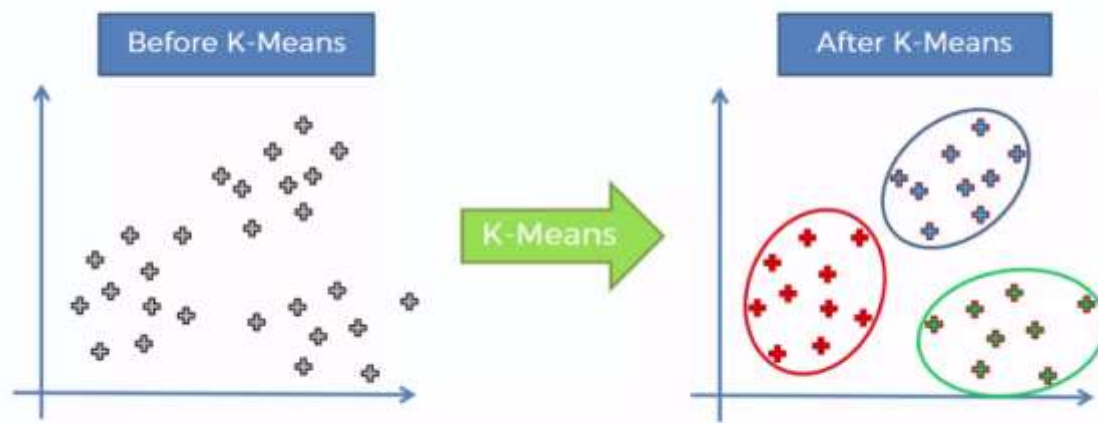
Facilities: GoogleColab,Jupiter notebook

Input : <https://www.kaggle.com/datasets/kyanyoga/sample-sales-data>

Output:

- 1)Data preprocessing visuals if any
- 2)Compute the centroids

K-Means



Conclusion:

Hence, we have successfully studied implementation of K-Means Algorithm successfully.

Questions:

1. What is Clustering?
2. How is Clustering an Unsupervised Learning Problem?
3. Properties of Clusters.
4. Applications of Clustering in Real-World Scenarios?
5. Compare Hierarchical Clustering and K-Means Clustering.
6. Explain some cases where k-Means clustering fails to give good results.

SCOE

Assignment No.	7 (GROUP B)
Title	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.. OR Build a machine learning model that predicts the type of people who survived the Titanic shipwreck using passenger data (i.e. name, age, gender, socio-economic class, etc.). OR Develop a application for signature identification by creating your own dataset of your college student
Subject	
Class	
Roll No.	
Date	
Signature	

Assignment No: 7

Title:

Problem Statement:

Prerequisites:

Objectives:

Theory:

Facilities:

Google Colab, Jupiter notebook

Algorithm:

Input:

Output:

Conclusion:

Questions:

GROUP C: BLOCKCHAIN TECHNOLOGY

Assignment No.	1 (GROUP C)
Title	Installation of Metamask and study spending Ether per transaction.
Subject	Laboratory Practice-III
Class	
Roll No.	
Date	
Signature	

Assignment No: 1

Title: Installation of Metamask and study spending Ether per transaction.

Problem Statement: Install the extension step of Metamask on browser and spending ether per transaction.

Prerequisites: Blockchain Technology

Objectives: Understand and explore the working of Blockchain Technology and its applications.

Outcome: Interpret the basic concepts in Blockchain technology and its applications.

Theory:

MetaMask:

Blockchain offers privacy, transparency, and immutability. You will be powered to use applications, transact anywhere, and do a lot more without anyone watching (read Google, Governments). But there are various blockchains, each one coded for a different purpose. However, Ethereum, a gigantic decentralized ecosystem, is for the masses. And MetaMask is a free, open-source, hot wallet to get you rolling with Ethereum.

Ease of use

Starting with MetaMask is easy, quick, and anonymous. You don't even need an email address. Just set up a password and remember (and store) the secret recovery phrase, and you're done.

Security

Your information is encrypted in your browser that nobody has access to. In the event of a lost password, you have the 12-word secret recovery phase (also called a seed phrase) for recovery. Notably, it's essential to keep the seed phrase safe, as even MetaMask has no information about it. Once lost, it can't be retrieved.

Built-In Crypto Store

If you're wondering, no, you can't buy Bitcoin with MetaMask. It only supports Ether and other Ether-related tokens, including the famous ERC-20 tokens. Cryptocurrencies (excluding Ether) on Ethereum are built as ERC-20 tokens.

Backup and Restore

MetaMask stores your information locally. So, in case you switch browsers or machines, you can restore your MetaMask wallet with your secret recovery phrase.

Installing MetaMask on Chrome

Go to the [download page](#), choose your platform, and hit the download button at the button. It will auto-detect the browser. But for iOS and Android, select your platform to get the links or search in the respective app stores.

MetaMask Browser Extension

Visit the download page and Click **Install MetaMask for Chrome** to visit the MetaMask extension page on the chrome web store.



Install and pin that to make it available in your browser toolbar. This process is similar to installing any [browser extension on chrome](#).

MetaMask Wallet Registration

Finally, you will be greeted with this. Now click **Get Started** to begin the registration process.



Welcome to MetaMask



Connecting you to Ethereum and the Decentralized Web.

We're happy to see you.

Get Started

The subsequent screen will give you the option to **Import Wallet**, or you can **Create a Wallet**. Click the latter option to register a new one.

New to MetaMask?

 No, I already have a Secret Recovery Phrase Import your existing wallet using a Secret Recovery Phrase <p>Import wallet</p>	 Yes, let's get set up! This will create a new wallet and Secret Recovery Phrase <p>Create a Wallet</p>
---	--

Next, you can choose to share some anonymous data with MetaMask to jump on to the following screen. You can click **No Thanks** for complete peace of mind—it won't hamper your user experience with MetaMask. Select accordingly and proceed.



Help us improve MetaMask

MetaMask would like to gather usage data to better understand how our users interact with the extension. This data will be used to continually improve the usability and user experience of our product and the Ethereum ecosystem.

MetaMask will..

- ✓ Always allow you to opt-out via Settings
- ✓ Send anonymized click & pageview events

- ✗ **Never** collect keys, addresses, transactions, balances, hashes, or any personal information
- ✗ **Never** collect your full IP address
- ✗ **Never** sell data for profit. Ever!

No Thanks

I Agree

Now Create a strong password on the following screen.



METAMASK

< Back

Create Password

New password (min 8 chars)

.....

Confirm password

.....



I have read and agree to the [Terms of Use](#)

Create

Next, you'll see the MetaMask team trying to emphasize the importance of the secret recovery phrase with a short video. Better watch it and click **Next**. Now it's time to save the secret recovery phrase. You can uncover the secret phrase and download it as a text file. Then click **Next** to proceed.



Secret Recovery Phrase

Your Secret Recovery Phrase makes it easy to back up and restore your account.

WARNING: Never disclose your Secret Recovery Phrase. Anyone with this phrase can take your Ether forever.



Tips:

Store this phrase in a password manager like 1Password.

Write this phrase on a piece of paper and store in a secure location. If you want even more security, write it down on multiple pieces of paper and store each in 2 - 3 different locations.

Memorize this phrase.

Download this Secret Recovery Phrase and keep it stored safely on an external encrypted hard drive or storage medium.



MetaMask goes to lengths to ensure you remember this important phrase as they don't store it for you. So you see a word puzzle to rebuild that phrase, exactly the way it was. Unless you get it right, the Confirm button at the end won't get activated, and you can't proceed. And if you've missed remembering or saving it, there is a back button under the MetaMask icon at the top. That will take you to the previous page to download and memorize it.

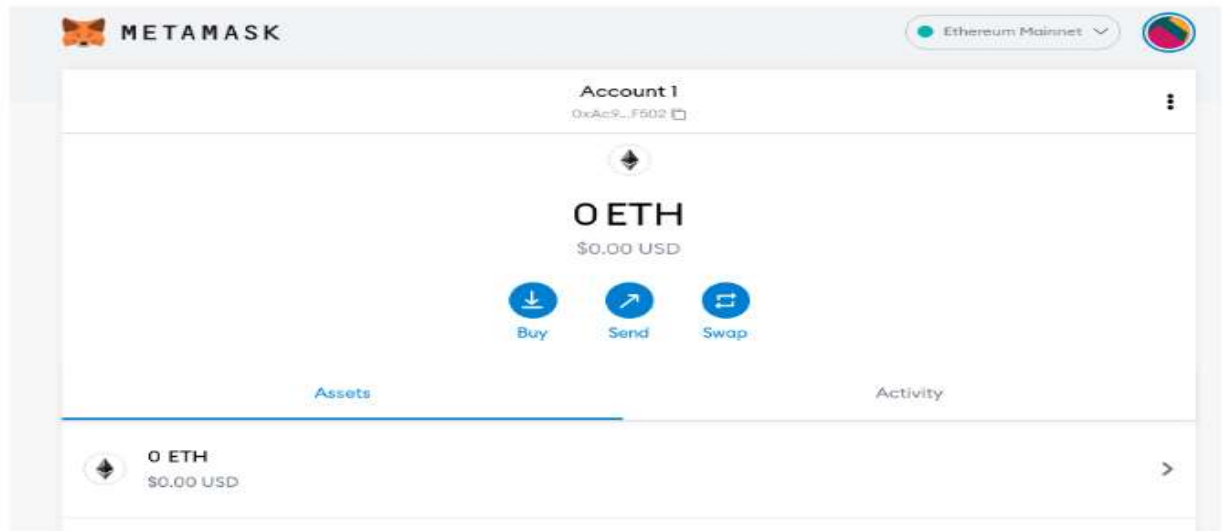


Confirm your Secret Recovery Phrase

Please select each phrase in order to make sure it is correct.



Once you complete this, the Confirm button will come to life. Click that, and you will see the congratulations message on the last screen of the registration process. Finally, click the **All Done** on the congratulations page to enter into your wallet dashboard.



Conclusion: Hence, we have successfully installation step of metamask.

Questions:

- b. What is metamask? Explain in detail.
- c. What is blockchain?
- d. How to install metamask on browser?
- e. How to create your password on metamask.

Assignment No.	2 (GROUP C)
Title	Create your own wallet using Metamask for crypto transactions
Subject	Laboratory Practice-III
Class	
Roll No.	
Date	
Signature	

Assignment No: 2

Title: Create your own wallet using Metamask for crypto transactions

Problem Statement: Generate your own wallet using metamask for crypto on web browser.

Prerequisites: Blockchain Technology

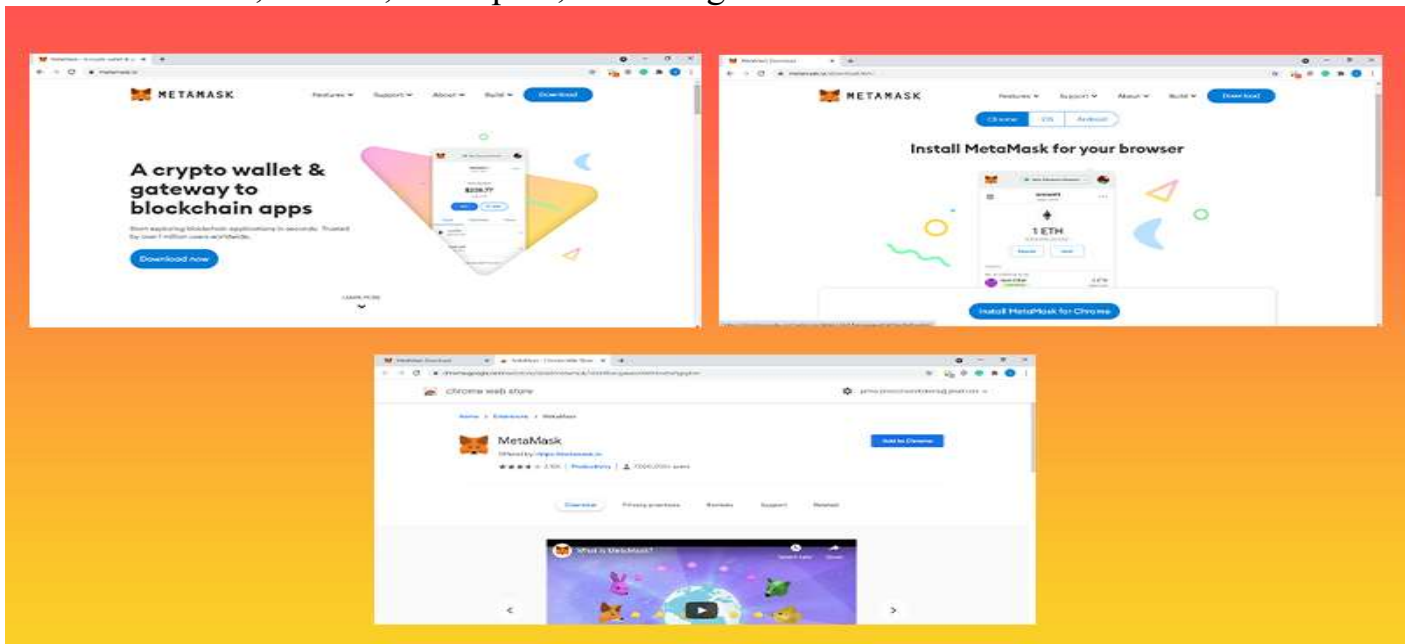
Objectives: Implement Own wallet using Metamask for crypto transactions.

Outcome: Interpret the basic concepts in Blockchain technology and its applications.

Theory:

Metamask: MetaMask is a popular browser extension that allows one to store Ethereum and other ERC-20 tokens. The free and secure extension allows web apps to read and interact with Ethereum's blockchain. MetaMask is a hot wallet, where you can store your cryptocurrencies online. Users can send cryptocurrencies in conventional transactions and interact with many decentralized apps using this web browser plugin. You can follow these simple steps to create MetaMask wallet.

Step 1: Install MetaMask in the browser: MetaMask can be downloaded through the project's official website or from an app store. Since previously app stores had phoney MetaMask programs, it is safer to download from the official website. MetaMask is now available in most browsers' stores like Chrome, Firefox, and Opera, so finding it isn't difficult.



Although, if the links above don't work for you, simply type MetaMask Extension into your preferred search engine.

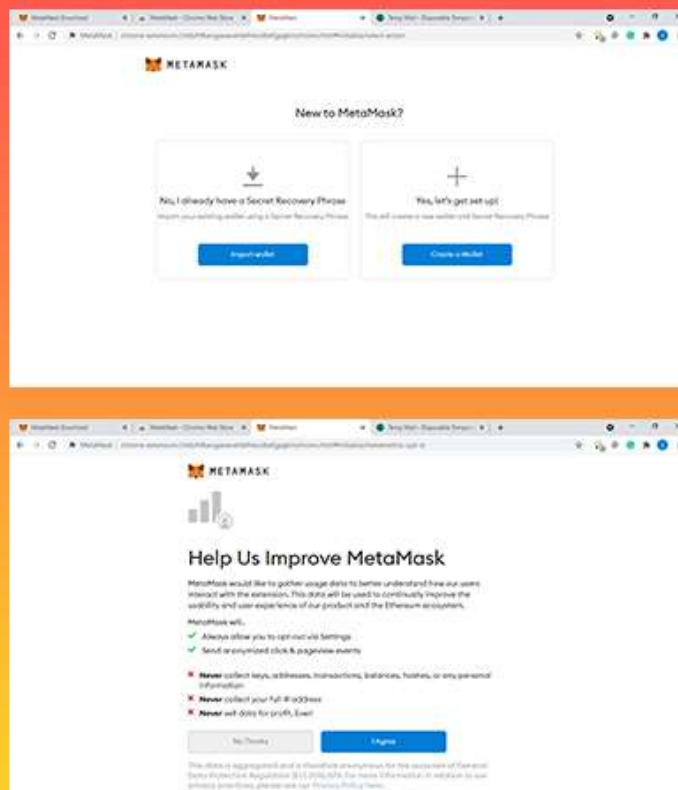
After installing MetaMask

- Click on **add to chrome**
- Then click **add extension**

Step 2: Create Your wallet

Now it's time to create your wallet.

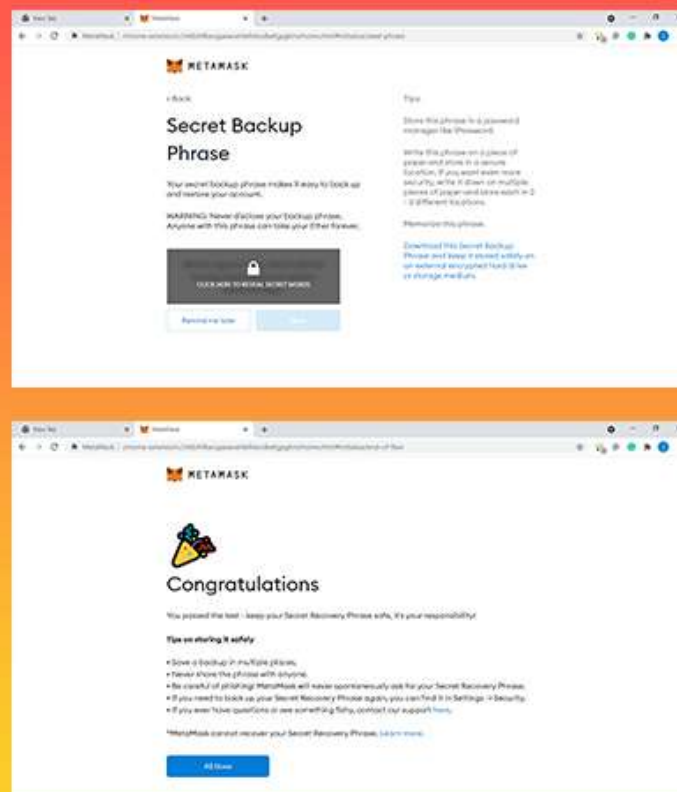
- To open MetaMask, click the extension icon in the upper right corner.
- Click Try it now to get the most recent version and stay up to date.
- Then, press the Enter key to continue.
- You'll be asked to choose a new password. Select Create from the drop-down menu.



Continue by **clicking Next** and agreeing to the Terms of Service.

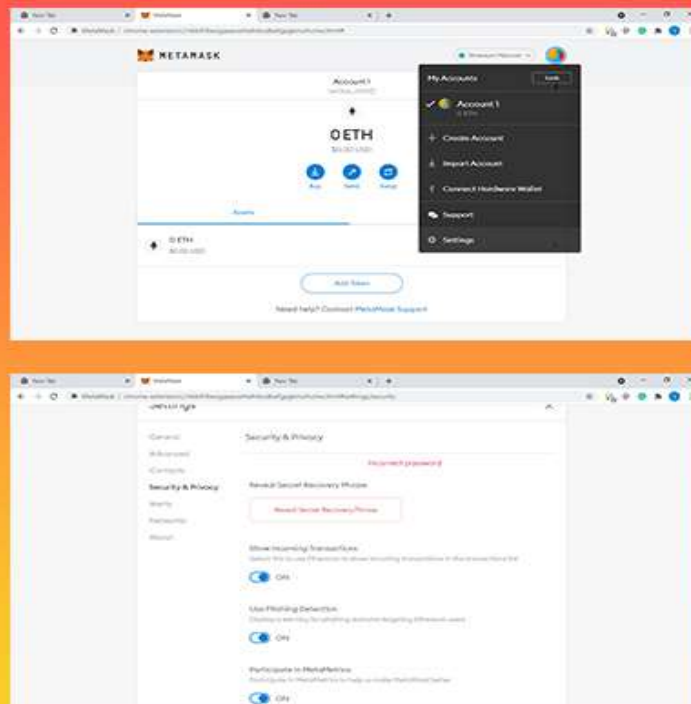
Step 3: Store Your Private Key

- Select Reveal Secret Words from the drop-down menu. There is a 12-word seed phrase there. This is crucial information that should not be stored digitally, so take your time and write it down.
- Then to verify, click confirm.
- Always keep your Private key safe, if you lose it you cannot access your wallet.
- If anyone somehow gains your private key, they can have complete access to your money.
- MetaMask never asks user's seed phrases and operates no Google Doc-based support.



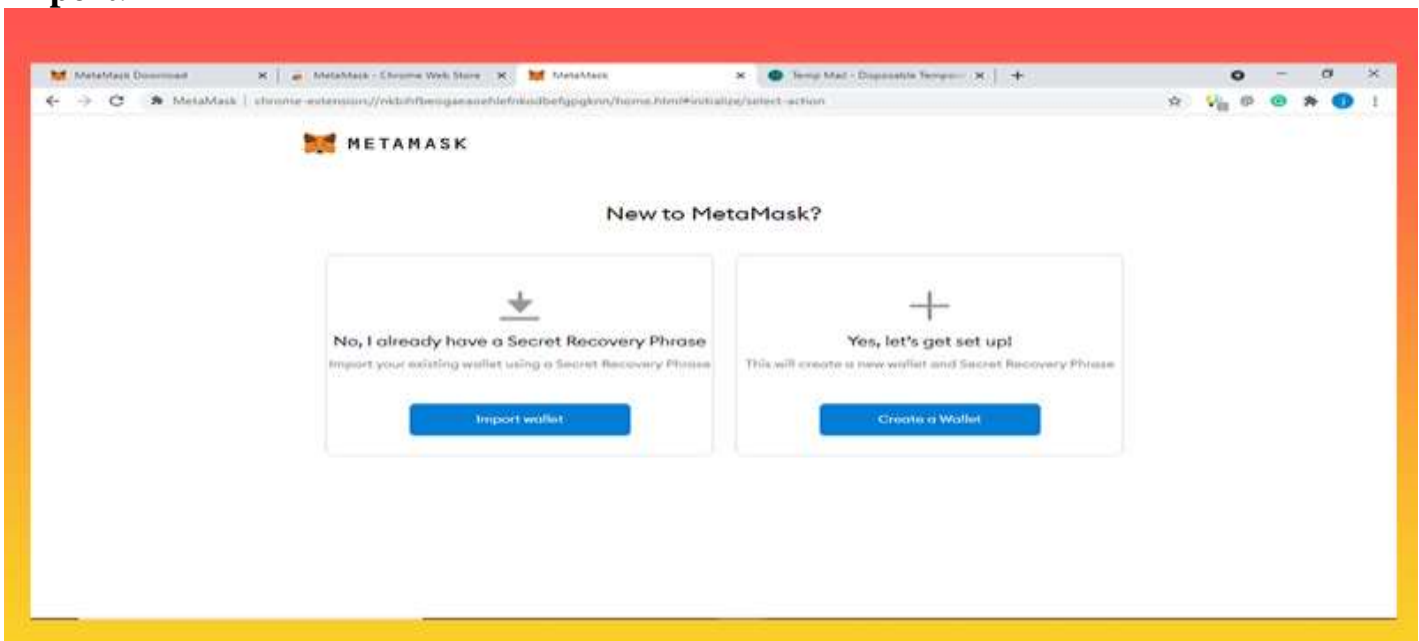
Use the seed phrase to backup and restore your wallet: If you change your browser or computer and want to reconnect with your wallet, this is how you can do it. To begin, you must locate your seed phrase, which you should have previously saved.

- Select the account icon from the drop-down menu. Then select Settings.
- Click Reveal Seed Words once you find it.
- Type in your password.



Now write down the Secret Seed Phrase somewhere, preferably not on your computer. Now that you've completed the backup, all you have to do is discover how to restore the data.

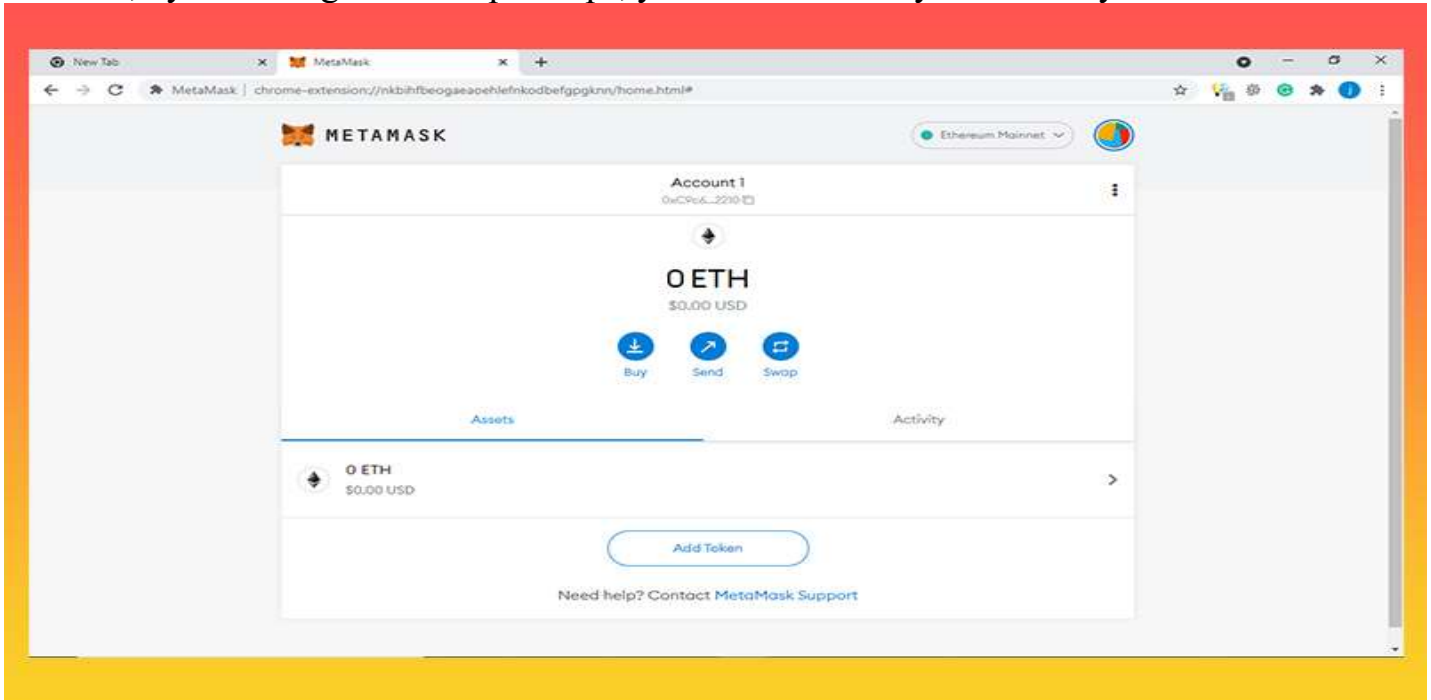
Step 4: Import wallet on Metamask : Using the account seed phrase, open MetaMask and **click Import**.



Type your Seed Phrase.

- Now, make a new, secure password.
- Select Restore from the menu.

That's it, by following these simple steps, you've successfully recovered your account.



In terms of wallet management, now you are fully prepared to take on the crypto world.

Conclusion: Hence, we have successfully studied MetaMask is gateway to the decentralised web, thus knowing how to properly create own wallet in metamask.

Questions:

1. What is metamask?
2. How to create your own wallet metamask?
3. What is crypto transactions.

Assignment No.	3 (GROUP C)
Title	Write a smart contract on a test network, for Bank account of a customer for following operations: -Deposit money -Withdraw Money -Show balance
Subject	Laboratory Practice-III
Class	
Roll No.	
Date	
Signature	

Assignment No: 3

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

Problem Statement: how to write a simple, but complete, smart contract in Solidity that acts like a bank account of a customer.

Prerequisites: Blockchain Technology.

Objective: Implement smart contract on a test network for bank account customer.

Outcome: To understand the implementation for bank account Customer using smart contract on test network.

Theory:

Smart Contract:

Smart contracts are computer programs or protocols for automated transactions that are stored on a blockchain and run in response to meeting certain conditions. In other words, smart contracts automate the execution of agreements so that all participants can ascertain the outcome as soon as possible without the involvement of an intermediary or time delay.

- Smart contracts are self-executing contracts in which the contents of the buyer-seller agreement are inscribed directly into lines of code.

Benefits of Smart Contracts:

Accuracy, Speed, and Efficiency

- The contract is immediately executed when a condition is met.
- Because smart contracts are digital and automated, there is no paperwork to deal with, and
- No time was spent correcting errors that can occur when filling out documentation by hand.

Trust and Transparency

- There's no need to worry about information being tampered with for personal gain because there's no third party engaged and
- Encrypted transaction logs are exchanged among participants.

Security

- Because blockchain transaction records are encrypted, they are extremely difficult to hack.
- Furthermore, because each entry on a distributed ledger is linked to the entries before and after it, hackers would have to change the entire chain to change a single record.

Savings

- Smart contracts eliminate the need for intermediaries to conduct transactions, as well as the time delays and fees that come with them.

Smart Contracts Work:

A smart contract is a sort of program that encodes business logic and operates on a dedicated virtual machine embedded in a blockchain or other distributed ledger.

Step 1: Business teams collaborate with developers to define their criteria for the smart contract's desired behavior in response to certain events or circumstances.

Step 2: Conditions such as payment authorization, shipment receipt, or a utility meter reading threshold are examples of simple events.

Step 3: More complex operations, such as determining the value of a derivative financial instrument, or automatically releasing an insurance payment, might be encoded using more sophisticated logic.

Step 4: The developers then use a smart contract writing platform to create and test the logic. After the application is written, it is sent to a separate team for security testing.

Step 5: An internal expert or a company that specializes in vetting smart contract security could be used.

Step 6: The contract is then deployed on an existing blockchain or other distributed ledger infrastructure once it has been authorized.

Step 7: The smart contract is configured to listen for event updates from an "oracle," which is effectively a cryptographically secure streaming data source, once it has been deployed.

Step 8: Once it obtains the necessary combination of events from one or more oracles, the smart contract executes.

Sample Code:

Wallets are just like your bank account, through which we can receive money, send money, and can check the balance.

Deposit money into the create account:

```
// pragma is the directive through  
// which we write the smart contract  
pragma solidity 0.6.0;
```

```
// Creating a contract  
contract wallet{
```

```
    address payable public Owner;
```

```
// mapping is created for mapping
// address=>uint for transaction
mapping(address=>uint) Amount;

// Defining a constructor
constructor() public payable{
// msg.sender is the address of the
// person who has currently deployed contract
    Owner = msg.sender
    Amount[Owner] = msg.value;
}
modifier onlyOwner(){
    require(Owner == msg.sender);
}

function sendMoney (address payable receiver, uint amount)
public payable onlyOwner
{
    require( receiver.balance>0);
    require(amount>0);
    Amount[Owner] -= amount;
    Amount[receiver] += amount;
}
function ReceiveMoney() public payable{
}

function CheckBalance_contractAccount()
public view onlyOwner returns(uint){
    return address(this).balance;
}
function CheckBalance()
public view onlyOwner returns(uint){
    return Amount[Owner];
}
}
```

Withdrawals and Account Balances:

Given the balance Of mapping from account addresses to ether amounts, the remaining code for a fully-functional bank contract is pretty small. simply add a withdrawal function:

```
bank.sol
pragma solidity ^0.4.19;

contract Bank {

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

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

- The `require(amount <= balances[msg.sender])` checks to make sure the sender has sufficient funds to cover the requested withdrawal. If not, then the transaction aborts without making any state changes or ether transfers.
- The `balanceOf` mapping must be updated to reflect the lowered residual amount after the withdrawal.
- The funds must be sent to the sender requesting the withdrawal.

Conclusion: Hence, we have successfully studied smart contract on test network.

Questions:

1. What is smart contract?
2. What are benefits of Smart contract?
3. How do smart contract works?

Assignment No.	4 (GROUP C)
Title	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.
Subject	Laboratory Practice-III
Class	
Roll No.	
Date	
Signature	

Assignment No: 4

Title: Write a program in solidity to create Student data. and Deploy this as smart contract on Ethereum and Observe the transaction fee and Gas values

Problem Statement: create Student data and follow the Structures Arrays Fallback Deploy this as smart contract on Ethereum and Observe the transaction fee and Gas values.

Prerequisites: Blockchain Technology

Objectives: Implement solidity programming and smart contract on Ethereum and observe the transaction fee and Gas values

Outcomes: To learn the concept of smart contract on Ethereum and observe the transaction fee and gas values.

Theory:

Ethereum:

Ethereum is a decentralized blockchain designed to be highly secure, fault-tolerant, and programmable. Ethereum blockchain is a choice for many developers and businesses. As said programmable, the main task of Ethereum is to securely execute and verify the application code known as smart contracts. Ethereum helps to build native scripting language(solidity) and EVM.

Overview of Smart Contracts:

A smart contract is a small program that runs on an Ethereum blockchain. Once the smart contract is deployed on the Ethereum blockchain, it cannot be changed. To deploy the smart contract to Ethereum, you must pay the ether (ETH) cost. Understand it as a digital agreement that builds trust and allows both parties to agree on a particular set of conditions that cannot be tampered with.

To understand the need for a smart contract, suppose there was one grocery shop, and ram went to buy some groceries. He purchased the groceries for 500 rupees and kept on debt that would pay the money next month when he returned, so the shopkeeper jotted down his purchase in his ledger. In between the period somehow shopkeeper changed 500 to 600 and when next month ram went to pay the money, the shopkeeper has demanded 600 INR and ram has no proof to show that he has only bought 500 INR so in this case, smart contracts play an essential role which prevents both the parties to tamper the agreement and only gets terminate when all the conditions satisfy after the deal.

Ethereum Blockchain Platform executes Smart Contracts:

Ethereum Virtual Machine (EVM)

The purpose of EVM is to serve as a runtime environment for smart contracts built on Ethereum. Consider it as a global supercomputer that executes all the smart contracts.

As the name indicates, Ethereum Virtual Machine is not physical but a virtual machine. The functionality of EVM is restricted to virtual machines; for example, it cannot make delayed calls on the internet or produce random numbers. Therefore, it is considered a simple state machine. Writing programs in assembly language do not make any sense, so, Ethereum required a programming language for the EVM.

Gas

In the Ethereum Virtual Machine, gas is a measurement unit used for assigning fees to each transaction with a smart contract. Each computation happening in the EVM needs some amount of gas. The more complex the computation is, the more the gas is required to run the smart contracts.

Transaction fee = Total gas used*gas price

Solidity:

Solidity is a smart contract programming language on Ethereum. Developed on the top of the EVM, it is similar to the object-oriented programming language that uses class and methods. It allows you to perform arbitrary computations, but it is used to send and receive tokens and store states. When it comes to syntax, Solidity is greatly influenced by C++, Python, and Javascript so that developers can understand its syntax quickly.

Solidity Programming

Solidity is object-oriented, high-level statically-typed programming language used to create smart contracts. Solidity programming looks similar to Javascript, but there are a lot of differences between both languages. In solidity, you need to compile the program first, while in Javascript, you can run the program directly in your browser or by using Node JS. Solidity is a Javascript-like language developed specifically for creating smart contracts. It is typed statically and supports libraries, inheritance and complex user-defined types. Solidity compiler converts code into EVM bytecode which is sent to the Ethereum network as a deployment transaction.

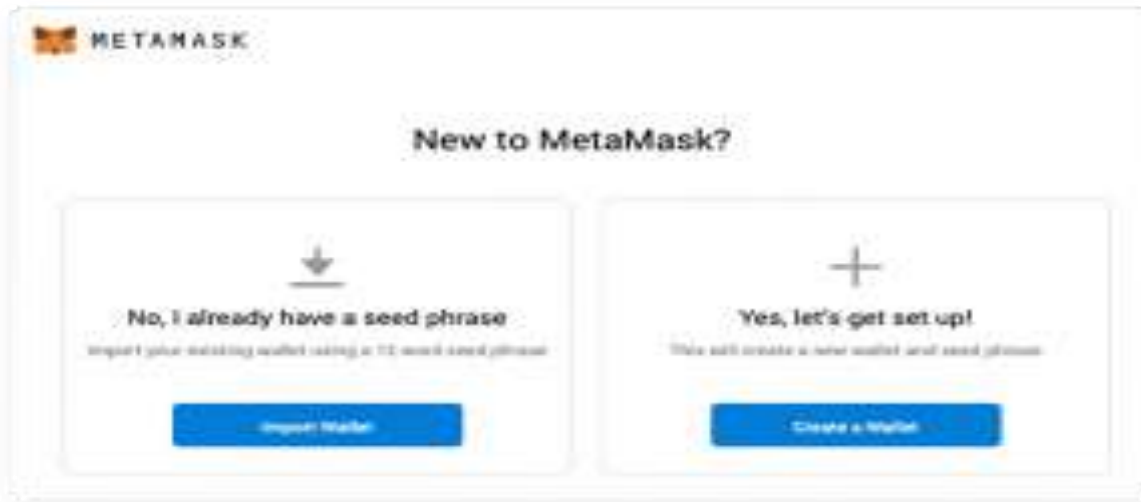
Here's a step-by-step guide to creating and deploying Ethereum Smart Contracts with Solidity

Installing Prerequisites

Meta-mask Chrome Extension

MetaMask acts both as an Ethereum browser and a wallet. It allows you to interact with smart contracts and dApps on the web without downloading the blockchain or installing any software. You only need to add MetaMask as a Chrome Extension, create a wallet and submit Ether. Though MetaMask is currently available for Google Chrome browser, it is expected to launch for Firefox

too in the coming years. Download MetaMask chrome extension before you start writing smart contracts. Once it is downloaded and added as a Chrome extension, you can either import an already created wallet or create a new wallet. You must have some ethers in your Ethereum wallet to deploy Ethereum smart contract on the network.



Steps to develop an Ethereum Smart Contract

Step 1: Create a wallet at meta-mask

Install MetaMask in your Chrome browser and enable it. Once it is installed, click on its icon on the top right of the browser page. Clicking on it will open it in a new tab of the browser. Click on “Create Wallet” and agree to the terms and conditions by clicking “I agree” to proceed further. It will ask you to create a password. After you create a password, it will send you a secret backup phrase used for backing up and restoring the account. Do not disclose it or share it with someone, as this phrase can take away your Ethers.

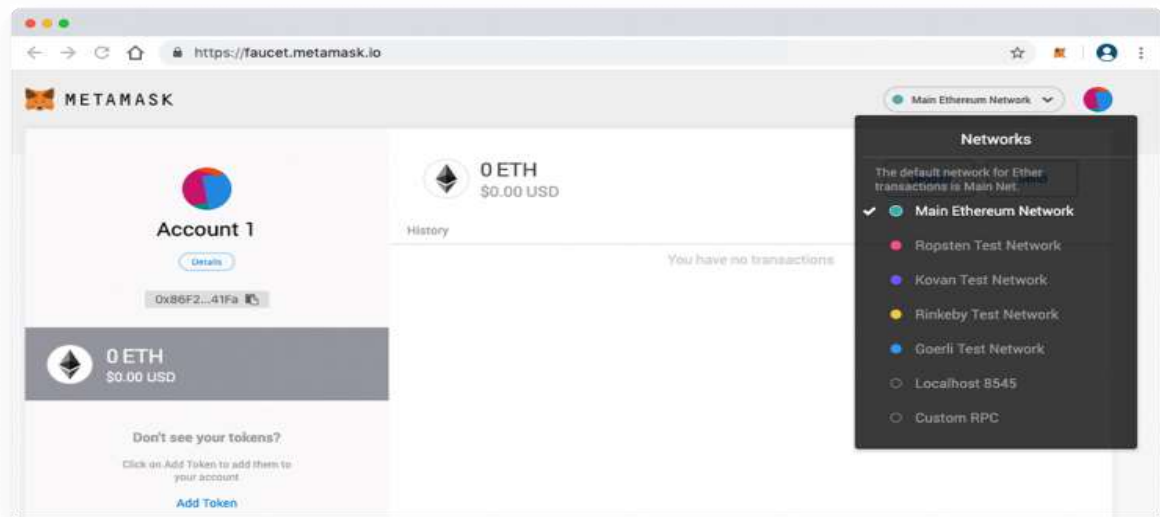


The next step is to ensure that you are in the “Main Ethereum Network.” If you find a checkmark next to “Main Ethereum Network”, you are in the right place.

Step 2: Select any one test network

You might also find the following test networks in your MetaMask wallet:

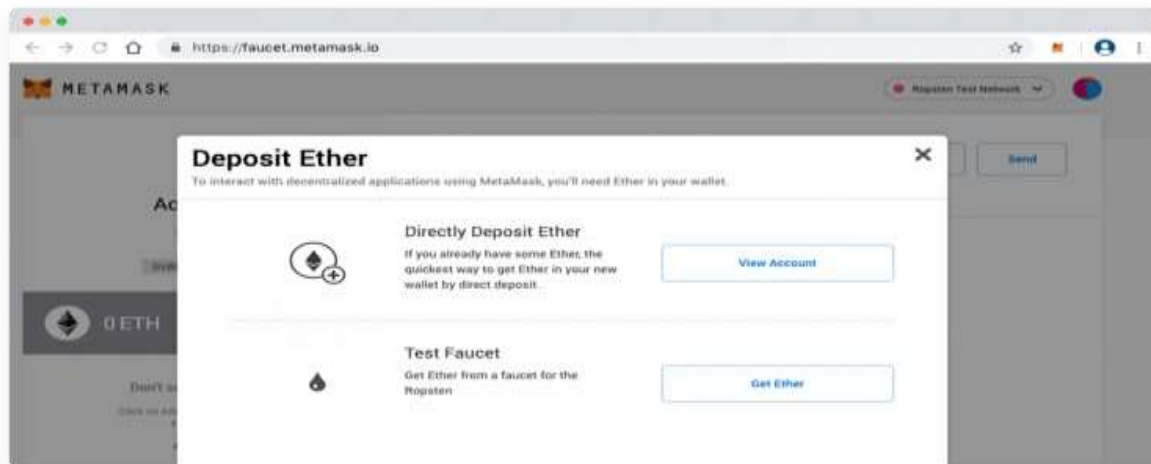
- Robsten Test Network
- Kovan Test Network
- Rinkeby Test Network
- Goerli Test Network



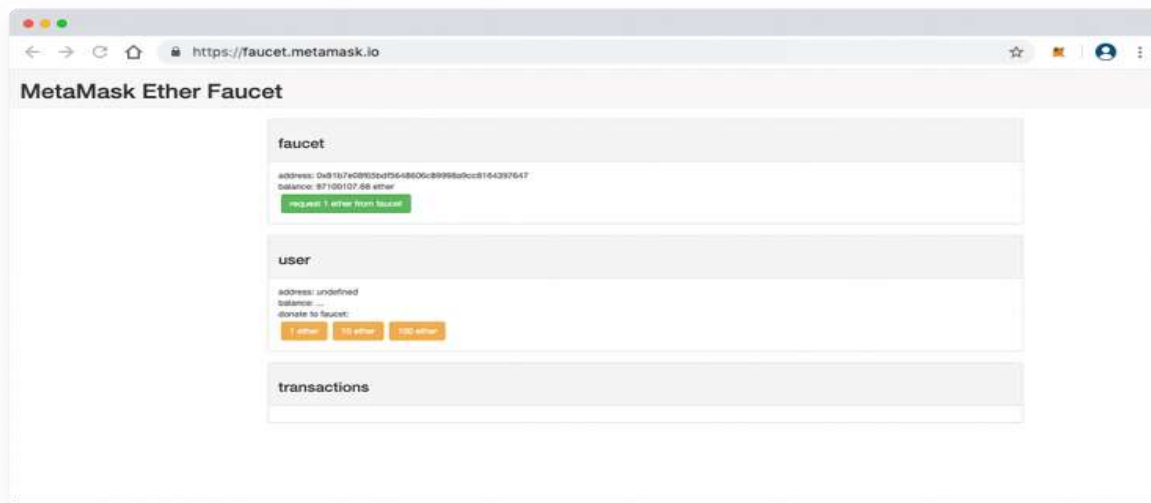
- The above networks are for testing purposes only; note that these networks' ethers have no real value

Step 3: Add some dummy Ethers to your wallet

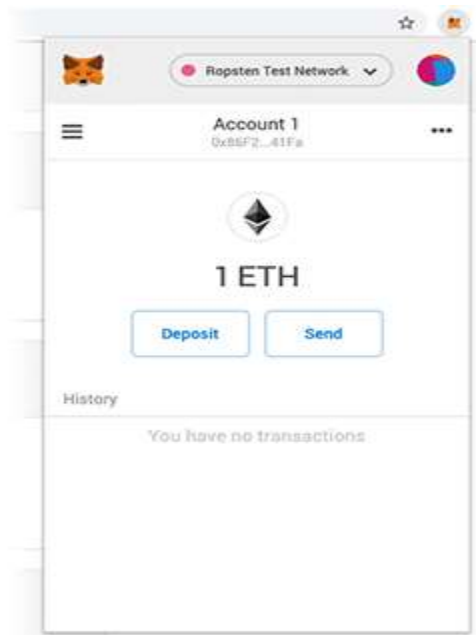
In case you want to test the smart contract, you must have some dummy ethers in your MetaMask wallet.



For example, if you want to test a contract using the Robsten test network, select it and you will find 0 ETH as the initial balance in your account. To add dummy ethers, click on the “Deposit” and “Get Ether” buttons under Test Faucet. To proceed, you need to click “request one ether from the faucet,” and 1 ETH will be added to your wallet. You can add as many Ethers you want to the test network.



For example, I have added 1 ETH in this scenario.



Once the dummy ethers are added to the wallet, you can start writing smart contracts on the Remix Browser IDE in the Solidity programming language.

Step 4: Use editor remix to write the smart contract in Solidity

We will use Remix Browser IDE to write our Solidity code. The remix is the best option for writing smart contracts as it comes with a handful of features and offers a comprehensive development experience.

Step 5: Create a .sol extension file

Open Remix Browser and click on the plus icon on the top left side, next to the browser to create a .sol extension file.

Step 6: A sample Code

Arrays in Solidity

The array is a special data structure used to create a list of similar type values. The array can be of fixed size and dynamic-sized. With the help of index elements can be accessed easily. below is a sample code to create, and access a fixed-sized array in solidity.

```
pragma solidity >= 0.5.0 < 0.9.0;
contract Array {
    uint [4] public arr = [10, 20, 30, 40];
    function setter(uint index, uint value) public {
        arr[index] = value;
    }
}
```

```
function length() public view returns(uint) {  
    return arr.length;  
}  
}
```

You can compile and deploy the code to try changing the array elements with an index and printing the array length.

Creating Dynamic Array

A dynamic array is an array where we can insert any number of elements and delete the details easily using an index. So solidity has functions like push and pops like python, making it easy to create a dynamic array. Below is a code using which you can create a dynamic array. After writing code, compile and deploy the code by visiting the deploy section in the left-side navigation bar. After that, try inserting and deleting some elements from an array.

```
pragma solidity >= 0.5.0 < 0.9.0;  
contract Array {  
    uint [] public arr;  
    function PushElement(uint item) public {  
        arr.push(item);  
    }  
    function Length() public view returns(uint) {  
        return arr.length;  
    }  
    function PopElement() public {  
        arr.pop();  
    }  
}
```

Structure in Solidity

The structure is a user-defined data type that stores more than one data member of different data types. As in array, we can only store elements of the same data type, but in structure, you can keep elements of different data types used to create multiple collections. The structure can be made outside and inside the contract storage, and the Structure keyword can be used to declare the form. The structure is storage type, meaning we use it in-store only, and if we want to use it in function, then we need to use the memory keyword as we do in the case of a string.

```
pragma solidity >= 0.5.0 < 0.9.0;  
struct Student {  
    uint rollNo;  
    string name;  
}
```

```
contract Demo {
    Student public s1;
    constructor(uint _rollNo, string memory _name) {
        s1.rollNo = _rollNo;
        s1.name = _name;
    }
    // to change the value we have to implement a setter function
    function changeValue(uint _rollNo, string memory _name) public {
        Student memory new_student = Student( {
            rollNo : _rollNo,
            name : _name
        });
        s1 = new_student;
    }
}
```

Fallback:

```
pragma solidity ^0.4.0;
// Creating a contract
contract fback
{
    // Declaring the state variable
    uint x;
    // Mapping of addresses to their balances
    mapping(address => uint) balance;
    // Creating a constructor
    constructor() public
    {
        // Set x to default
        // value of 10
        x=10;
    }
    // Creating a function
    function SetX(uint _x) public returns(bool)
    {
        // Set x to the
        // value sent
        x=_x;
    }
}
```

```
        return true;
    }

    // This fallback function
    // will keep all the Ether
    function() public payable
    {
        balance[msg.sender] += msg.value;
    }
}

// Creating the sender contract
contract Sender
{
    function transfer() public payable
    {
        // Address of Fback contract
        address _receiver =
            0xbcD310867F1b74142c2f5776404b6bd97165FA56;

        // Transfers 100 Eth to above contract
        _receiver.transfer(100);
    }
}
```

Create a Smart Contract with CRUD Functionality

We have excellent theoretical and hands-on practical knowledge about solidity, and now you can create a primary smart contract like hello world, getter, and setter contracts. So it's a great time to try making some functional smart contracts, and the best way to try all the things in one code is to create one program that performs all CRUD operations.

A sample smart contract code to create ERC20 tokens

```
pragma solidity ^0.4.0;
import "./ERC20.sol";
contract myToken is ERC20{
    mapping(address =>uint256) public amount;
    uint256 totalAmount;
    string tokenName;
    string tokenSymbol;
    uint256 decimal;
```

```
constructor() public{
totalAmount = 10000 * 10**18;
amount[msg.sender]=totalAmount;
tokenName="Mytoken";
tokenSymbol="Mytoken";
decimal=18;
}
function totalSupply() public view returns(uint256){
return totalAmount;
}
function balanceOf(address to_who) public view
returns(uint256){
return amount[to_who];
}
function transfer(address to_a,uint256 _value) public
returns(bool){
require(_value<=amount[msg.sender]);
amount[msg.sender]=amount[msg.sender]-_value;
amount[to_a]=amount[to_a]+_value;
return true;
}
}
```

Step 7: Deploy your contract

Deploy the smart contract at the Ethereum test network by pressing the deploy button at the Remix window's right-hand side. Wait until the transaction is complete.

After the transaction commits successfully, the address of the smart contract would be visible at the right-hand side of the remix window. At first, all the ERC20 tokens will be stored in the wallet of a user who is deploying the smart contract.

To check the tokens in your wallet, go to the metamask window, click add tokens, enter the smart contract address and click ok. You would be able to see the number of tokens there.

Steps to deploy Ethereum Smart Contracts

- To make your smart contract live, switch to the main ethereum network at metamask
- Add some real ethers.
- Now again, deploy your smart contract using remix as mentioned in the above steps.
- When a smart contract is deployed successfully, visit <http://www.etherscan.io> and search your smart contract address there. Select your smart contract.
- Now you need to verify your smart contract here, click “verify the contract.”

- Copy your smart contract code and paste it at Etherscan. Select the same compiler version that you selected at remix to compile your code.
- Check “optimization” to Yes, if you had selected optimization at remix; otherwise, select No.
- Click Verify.
- It will take a few minutes and your smart contract will be live if no issue occurs.
- You can now run your smart contract methods at Etherscan.

Conclusion: Hence, we have successfully studied Solidity is an object-oriented high-level programming language for creating a smart contract that runs on the Ethereum blockchain. We have learned about the smart contract and its creation using solidity programming.

Questions:

1. What is solidity programming.
2. What are the step Steps to develop an Ethereum Smart Contract
3. How to deploy Ethereum Smart Contract.
4. How to define array structure and fallback.
5. What is Ethereum? Explain in detail?

Assignment No.	5 (GROUP C)
Title	Write a survey report on types of Blockchains and its real time use cases.
Subject	Laboratory Practice-III
Class	
Roll No.	
Date	
Signature	

Assignment No: 5

Title: survey report on types of Blockchains and its real time use cases.

Problem Statement: Write and survey report on blockchain technology

Prerequisites: Blockchain Technology

Objectives: The objective is to explore deep into blockchain concepts, types, its real time use cases. Survey aggregates all the core concepts of blockchain technologies for future researchers and readers who are initiating their studies in the particular technology.

Outcomes: To understand the of blockchain concepts and implementation of real time use cases.

Theory:

Blockchain Concepts:

Blockchain can be defined as an immutable distributed digital ledger, which is secured using advanced cryptography, replicated among the peer nodes in the peer-to-peer network, and uses consensus mechanism to agree upon the transaction log, whereas control is decentralized. With this definition, paper identifies following concepts as the core concepts to unwrap the meaning of blockchain—immutable, distributed, digital ledger, cryptography, peer-to-peer network, consensus mechanism, decentralization. In accounting, a ledger is a place to record and store all the transactions with regard to an entity. A digital ledger could be a computer file, or database, or even distributed database like blockchain, where transactions are recorded electronically. Blockchain transaction ledger is pretty unique to other ledgers in a manner, which ensures that transaction log is computationally impractical to change, as long as honest nodes in the network control the majority of CPU power, thus making it immutable. The origins of ledger can be traced back to over 5000 years ago in Mesopotamia. The Earliest and simplest form of recording transactions is called single entry accounting, which enters transactions into a list to keep track of adding or deducting assets. The single entry accounting was managed by owners or family members, as this kind of recordings are error-prone as well as difficult to track down, when recorded fraudulently. Double entry accounting added a clear strategy to identify and remove errors, where there are two entries recorded against each transaction, so that the ledger is balanced all the time. Grigg proposed triple entry accounting in 2005, an alternative to traditional double entry accounting, which secures transactions using cryptography in order to make it difficult to change. Blockchain implements triple entry accounting concept to permanently store transactions in blockchain, ensuring that the sender has authority to execute non-reversible transactions using public-key cryptography.

Cryptography can be defined as techniques used for secure communication to protect confidential information, in the presence of adversaries. Blockchain uses concepts from public key cryptosystems to verify the authority of the user to execute transactions, and cryptographic hash functions to achieve consensus between network nodes on blockchain data. The use of public key cryptosystems to provide digital signatures was suggested by Diffie and Hellman. Digital signatures, whether based on public key cryptosystems, conventional encryption functions, on probabilistic computations, or other techniques share several important properties in common—such as an easier way for the sender to generate the personal digital signature, convenient way for receiver to verify the sender of the message, but must be impossible to generate someone else's digital signature by others. In public key cryptography, there exists two keys called public and private and a function or cypher algorithm to encrypt the original text into a ciphertext using the private encryption key. Sender or owner generates the public–private key pair and keeps the private key as the confidential key to encrypt information; public key is distributed to anyone to verify that the information is digitally signed by the original owner. This public key cryptography technique is used in blockchain to verify the ownership of coins or tokens, whenever transferring coins or tokens. One another important concept used in blockchain to secure its data integrity is cryptographic hash function—a one-way function that maps strings of arbitrary size into a bit string of fixed size called hash using a mathematical algorithm. An algorithm required for blockchain hash functions has three main properties—same input should always result in with the same output hash, given the hash no algorithm could produce the original input, small changes in input results in completely different output hash. Bitcoin uses SHA-256 hash function, whereas Ethereum uses Ethash, and Litecoin uses Script when hashing its block data.

Blockchain Types

According to our survey findings, blockchains can be categorized into two main types namely **permissionless blockchains** and **permissioned blockchains**.

Permissionless Blockchains

Permissionless blockchains do not enforce any restrictions on its nodes; anyone can openly read data, inspect data, and participate in validation and writing of the data in accordance with the consensus protocol of the particular blockchain. Bitcoin, Ethereum and many other cryptocurrencies run on permissionless blockchains. These blockchains are considered fully decentralized and secured using advanced cryptography, whereas economic incentives are provided for users who work to keep the integrity of the network. The transactions are completely irreversible on a permissionless blockchain by its design, meaning once confirmed by its nodes the blockchain transactions cannot be reversed. Due to the security considerations and strict restrictions, transaction throughput of a permissionless blockchain is comparatively lesser

than one of a permissioned blockchain. Permissionless blockchains are fully decentralized and transparent.

Permissioned Blockchains:

Permissioned blockchains restrict the writing access for a limited set of participants, and a consensus mechanism is used to validate the writing of data among its privileged participants. Read access could either be open to anyone or closed to the public based on the requirement of the permissioned blockchain. This type of blockchains has evolved as an alternative to initial permissionless blockchains, to address the requirement for running blockchain technology among a set of known and identifiable participants that have to be explicitly responsible to the blockchain network, while participants need not be fully trusting each other. The permissioned blockchains are mainly useful for business and social applications, which requires blockchain distributed ledger technology without the need of a in centifying cryptocurrency. Based on the read access mentioned, permissioned blockchains are further divided as open and closed—open permissioned blockchains are partially decentralized, anyone can read its data, whereas closed permissioned blockchains are fully centralized, data is visible only to the participants.

We thoroughly believe blockchain technology is rather necessary only for permissionless blockchains, and open permissioned blockchains. Closed permissioned blockchains can be argued as restricted distributed databases which are facelifted with the blockchain term. The initial idea of introducing blockchain concept was to remove centralization and add transparency to everyone to read and update its data. Open permissioned blockchains mostly adhere to this principle of transparency even though somewhat centralized in writing its data and could be useful for applications such as identity systems, academic certification systems, where anyone can read its data but only a certain set of participants are privileged to write the data into blockchain. Closed permissioned blockchains are fully centralized and also not transparent to anyone, dismantling the core concept of a blockchain. Therefore, these blockchains can be replaced with distributed database systems with restrictions implemented on top of it. For example, a supply chain management system for a private organization can be implemented without the concepts of blockchain. In order to support our argument on closed permissioned blockchains, we have presented a characteristic comparison of different blockchain types compared with restricted distributed database system. The comparison shows that all of the characteristics in closed permissioned blockchains are comparatively similar to that of restricted database systems. In addition to this categorization, there is also another blockchain categorization called public, consortium, and private blockchains. In simple terms, public blockchains are permissionless blockchains, whereas consortium and private blockchains fall into permissioned blockchains.

Real Time blockchain use cases

Blockchain technology's core characteristics include decentralization, transparency, immutability, and automation. These elements can be applied to various industries, creating a multitude of use cases. Here are what we believe to be the most pertinent blockchain use cases for enterprises, institutions, and governments.

Capital Markets

For capital markets, blockchain unlocks easier, cheaper, and faster access to capital. It reduces the barriers to issuance and enables peer-to-peer trading, faster and more transparent settlement and clearing, reduced costs, decreased counterparty risks, and streamlined auditing and compliance.

Central Bank Digital Currencies CBDC

CBDCs are a digital form of central bank money that offers central banks unique advantages at the retail and wholesale levels, including increased financial access for individual customers and a more efficient infrastructure for interbank settlements.

Decentralized Finance (DeFi)

Decentralized finance—DeFi—refers to the shift from traditional, centralized financial systems to peer-to-peer finance enabled by decentralized technologies built on Ethereum. Millions are building and participating in this new economic system that is setting new standards for financial access, opportunity, and trust.

Digital Identity

A blockchain-based digital identity system provides a unified, interoperable, and tamper-proof infrastructure with key benefits to enterprises, users, and IoT management systems. The solution protects against theft and provides individuals greater sovereignty over their data.

Finance

Financial services struggle with archaic operational processes, slow payment settlements, limited transparency, and security vulnerabilities. Blockchain enhances the efficient digitization of financial instruments, which increases liquidity, lowers cost of capital, and reduces counterparty risk.

Energy and Sustainability

Oil and gas companies suffer from siloed infrastructures and a lack of transparency, efficiency, and optimization. Enterprise-grade blockchain solutions can significantly increase process efficiencies and reduce costs associated with oil and gas operations and distribution.

Global Trade and Commerce

Major trading companies and consortiums are recognizing the transformative impact of blockchain in operating global supply chains, managing trade finance, and unlocking new business models. ConsenSys' blockchain products offer secure digitization, enabling the tokenization of existing documents, letters of credit, and more.

Government and the Public Sector

Ethereum blockchain technology allows governments to build trust, improve accountability and responsiveness, increase efficiency, reduce costs, and create high-performing government functions with more secure, agile, and cost-effective structures.

Healthcare and the Life Sciences

Blockchain-based healthcare solutions will enable faster, more efficient, and more secure medical data management and medical supply tracking. This could significantly improve patient care, facilitate the advancement to medical discoveries, and ensure the authenticity of drugs circulating global markets.

Real Estate

Enterprise Ethereum enables the digitization of assets and financial instruments. This enhances fractionalization of ownership, expanded access to global markets, increased liquidity, and democratized access to real estate investment opportunities.

Supply Chain Management

Existing global supply chains are inefficient, poorly tracked, and oftentimes exploitative. Blockchain can facilitate accurate asset tracking, enhanced licensing of services, products, and software, and transparency into the provenance of consumer goods, from sourcing to the point of consumption

Conclusion: Hence, we have successfully studied and survey report on Blockchain technology

Questions:

- 1.What is blockchain concepts.
- 2.What are types of blockchain technology? Explain in detail.
- 3.What are real time blockchain use cases.

Assignment No: 6

Title: Write a program to create a Business Network using Hyperledger

Problem Statement: Write and creating and deploying business network using Hyperledger.

Prerequisites: Blockchain Technology

Objectives: Implement a business network using Hyperledger.

Outcomes: To understand the implementation of business network using Hyperledger.

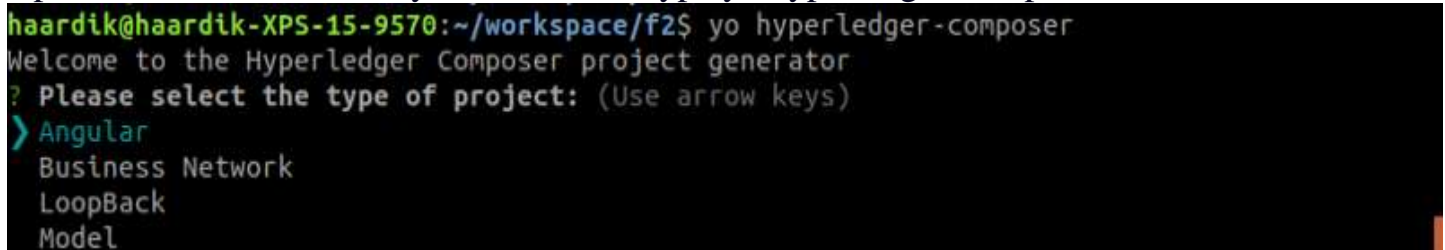
Theory:

Creating and deploying our business network

Remember the packages yo and generator-hyperledger-composer we installed earlier? yo provides us a generator ecosystem where generators are plugins which can be run with the yo command. This is used to set up boilerplate sample applications for various projects. generator-hyperledger-composer is the yo generator we will be using as it contains specs to generate boilerplate business networks among other things.

1. Generating a business network

Open terminal in a directory of choice and type yo hyperledger-composer



```
haardik@haardik-XPS-15-9570:~/workspace/f2$ yo hyperledger-composer
Welcome to the Hyperledger Composer project generator
? Please select the type of project: (Use arrow keys)
> Angular
  Business Network
  LoopBack
  Model
```

You'll be greeted with something similar to the above. Select Business Network and name it cards-trading-network as shown below:


```
haardik@haardik-XPS-15-9570:~/workspace/f2$ yo hyperledger-composer
Welcome to the Hyperledger Composer project generator
? Please select the type of project: Business Network
You can run this generator using: 'yo hyperledger-composer:businessnetwork'
Welcome to the business network generator
? Business network name: cards-trading-network
? Description: A Hyperledger Fabric network to trade cards between permissioned participants
? Author name: Haardik Haardik
? Author email: haardikk21@gmail.com
? License: Apache-2.0
? Namespace: org.example.biznet
? Do you want to generate an empty template network? Yes: generate an empty template network
  create package.json
  create README.md
  create models/org.example.biznet.cto
  create permissions.acl
  create .eslintrc.yml
haardik@haardik-XPS-15-9570:~/workspace/f2$
```

2. Modeling our business network

The first and most important step towards making a business network is identifying the resources present. We have four resource types in the modeling language:

- Assets
- Participants
- Transactions
- Events

For our cards-trading-network, we will define an asset type Trading Card, a participant type Trader, a transaction Trade Card and an event Trade Notification.

Go ahead and open the generated files in a code editor of choice. Open up org.example.biznet.cto which is the modeling file. Delete all the code present in it as we're gonna rewrite it (except for the namespace declaration).

This contains the specification for our asset Trading Card. All assets and participants need to have a unique identifier for them which we specify in the code, and in our case, it's card ID

Also, our asset has a Game Type card Type property which is based off the enumerator defined below. Enums are used to specify a type which can have up to N possible values, but nothing else.

In our example, no Trading Card can have a card Type other than Baseball, Football, or Cricket

Now, to specify our Trader participant resource type, add the following code in the modeling file

This is relatively simpler and quite easy to understand. We have a participant type Trader and they're uniquely identified by their trader Ids.

Now, we need to add a reference to our Trading Cards to have a reference pointing to their owner so we know who the card belongs to. To do this, add the following line inside your Trading Card asset:

--> Trader owner

so that the code looks like this:

This is the first time we've used --> and you must be wondering what this is. This is a relationship pointer. o and --> are how we differentiate between a resource's own properties vs a relationship to another resource type. Since the owner is a Trader which is a participant in the network, we want a reference to that Trader directly, and that's exactly what --> does.

. Generating a Business Network Archive (BNA)

Now that all the coding is done, it's time to make an archive file for our business network so we can deploy it on our local Fabric runtime. To do this, open Terminal in your project directory and type this: `composer archive create --sourceType dir --sourceName .`

This command tells Hyperledger Composer we want to build a BNA from a directory which is our current root folder.

```
haardik@haardik-GT72S-6QE:~/Desktop/workspace/hyperledger-tutorial/cards-trading-network$ composer archive create --sourceType dir --sourceName .
Creating Business Network Archive

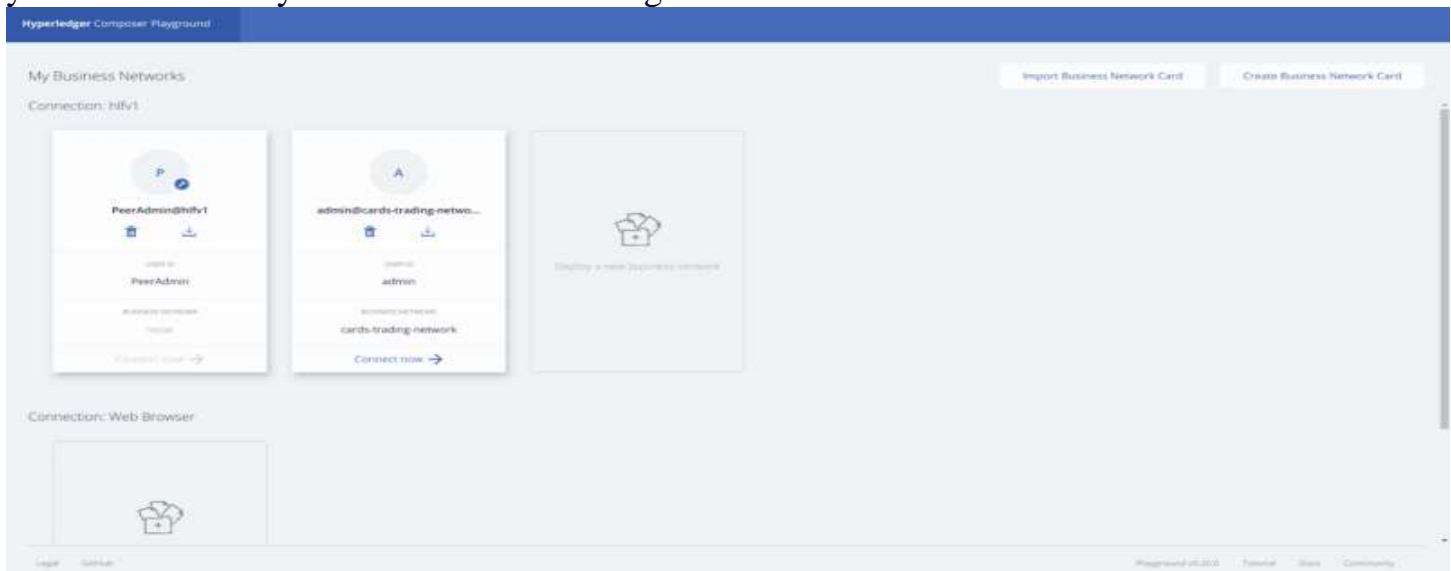
Looking for package.json of Business Network Definition
  Input directory: /home/haardik/Desktop/workspace/hyperledger-tutorial/cards-trading-network
Found:
  Description: A Hyperledger Fabric network to trade cards between permissioned participants
  Name: cards-trading-network
  Identifier: cards-trading-network@0.0.1

Written Business Network Definition Archive file to
  Output file: cards-trading-network@0.0.1.bna

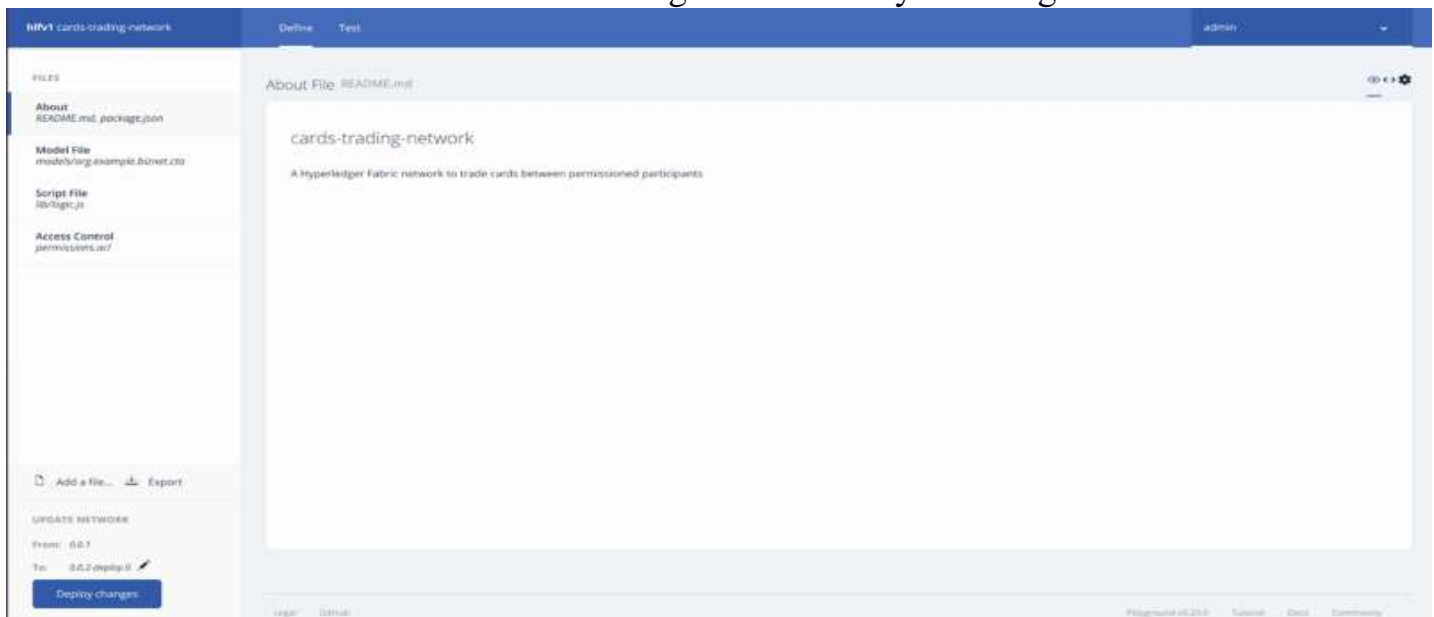
Command succeeded
```

Testing our Business Network

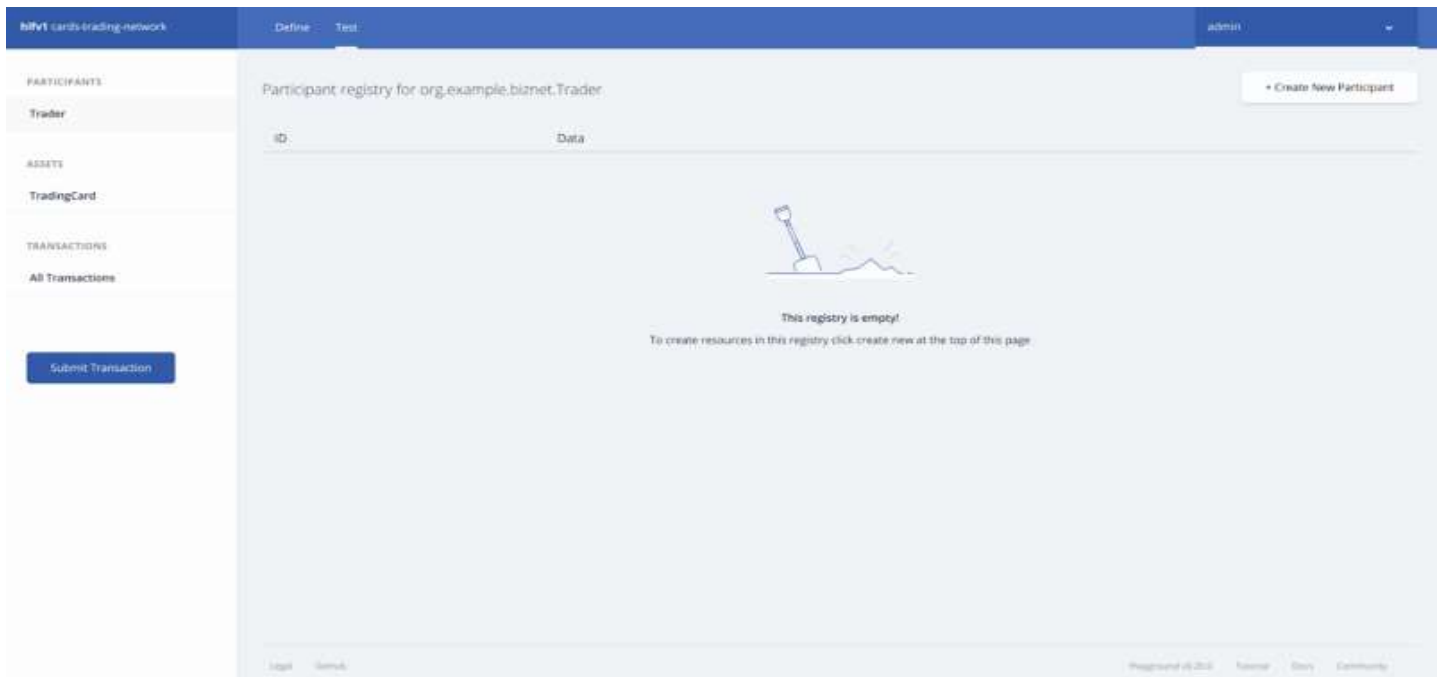
Now that our network is up and running on Fabric, we can start Composer Playground to interact with it. To do this, type `composer-playground` in Terminal and open up <http://localhost:8080/> in your browser and you should see something similar to this:



Press **Connect Now** for `admin@cards-trading-network` and you'll be greeted with this screen:



The **Define** page is where we can make changes to our code, deploy those changes to upgrade our network, and export business network archives. Head over to the **Test** page from the top menu, and you'll see this:



Select Trader from Participants, click on **Create New Participant** near the top right, and make a new Trader similar to this:

Create New Participant

In registry: **org.example.biznet.Trader**

JSON Data Preview

```
1  {
2    "$class": "org.example.biznet.Trader",
3    "traderId": "1",
4    "traderName": "Haardik"
5  }
```

Conclusion:

Hence, we have studied and implement of analysis of Business Network using Hyperledger

Questions:

- 1.How do you set up a Hyperledger Fabric network?
- 2.Which command is used for generating the business network?
- 3.How do you implement Hyperledger?
- 4.What can be included in a business network?
- 5.What are the three components that make up Hyperledger composer?

Assignment No.	7 (GROUP C) (Mini Project)
Title	Create a dApp (de-centralized app) for e-voting system OR Develop a Blockchain based application for transparent and genuine charity OR Develop a Blockchain based application for health related medical records OR Develop a Blockchain based application for mental health
Subject	Laboratory Practice-III
Class	
Roll No.	
Date	
Signature	

Assignment No: 7

Title:

Problem Statement:

Prerequisites:

Objectives:

Theory:

Facilities:

Meta-mask Google Extension

Algorithm:

Input:

Output:

Conclusion:

Questions: