Assignment No: 1

Title of the Assignment: Write a program non-recursive and recursive program to calculate Fibonacci numbers and analyze their time and space complexity.
Objective of the Assignment: Students should be able to perform non-recursive and recursive programs to calculate Fibonacci numbers and analyze their time and space complexity
Prerequisite:
1. Basic of Python or Java Programming

2. Concept of Recursive and Non-recursive functions

3. Execution flow of calculate Fibonacci numbers

4. Basic of Time and Space complexity

Contents for Theory:
1. Introduction to Fibonacci numbers
2. Time and Space complexity
Introduction to Fibonacci numbers
● The Fibonacci series, named after Italian mathematician Leonardo Pisano Bogollo, later known as Fibonacci, is a series (sum) formed by Fibonacci numbers denoted as Fn. The numbers in Fibonacci sequence are given as: 0, 1, 1, 2, 3, 5, 8, 13, 21, 38, . . .

● In a Fibonacci series, every term is the sum of the preceding two terms, starting from 0 and 1 as first and second terms. In some old references, the term '0' might be omitted.

What is the Fibonacci Series?
● The Fibonacci series is the sequence of numbers (also called Fibonacci numbers), where every number is the sum of the preceding two numbers, such that the first two terms are '0' and '1'.

● In some older versions of the series, the term '0' might be omitted. A Fibonacci series can thus be given as, 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, . . . It can be thus be observed that every term can be calculated by adding the two terms before it.

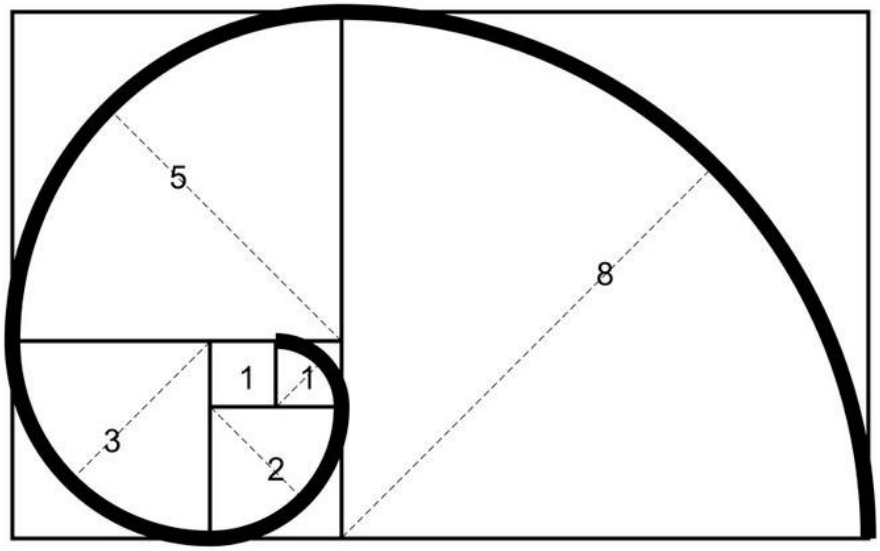
● Given the first term, F0 and second term, F1 as '0' and '1', the third term here can be given as, F2 = 0 + 1 = 1

Similarly,

F3 = 1 + 1 = 2

F4 = 2 + 1 = 3

Given a number n, print n-th Fibonacci Number



Fibonacci Sequence Formula

The Fibonacci sequence of numbers “Fn” is defined using the recursive relation with the seed values F0=0 and F1=1:
Fn = Fn-1+Fn-2
Here, the sequence is defined using two different parts, such as kick-off and recursive relation.
The kick-off part is F0=0 and F1=1.
The recursive relation part is Fn = Fn-1+Fn-2.
It is noted that the sequence starts with 0 rather than 1. So, F5 should be the 6th term of the sequence.

Examples:

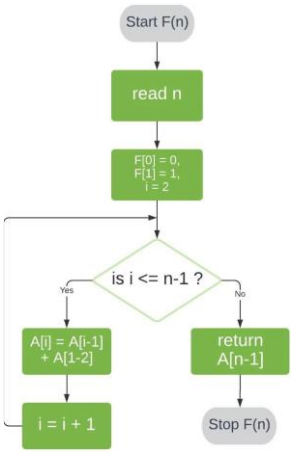
Input : n = 2 Output : 1 Input : n = 9 Output : 34
The list of Fibonacci numbers are calculated as follows:

Fn	Fibonacci Number
0	0
1	1
2	1
3	2
4	3
5	5

6	8
7	13
8	21
9	34
… and so on	… and so on

Method 1 (Use Non-recursion)

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



# Program to display the Fibonacci sequence up to n-th term nterms = int(input("How many terms? "))
# first two terms n1, n2 = 0, 1
count = 0
# check if the number of terms is valid if nterms <= 0:
print("Please enter a positive integer") # if there is only one term, return n1
elif nterms == 1:
print("Fibonacci sequence upto",nterms,":") print(n1)
# generate fibonacci sequence
else:
print("Fibonacci sequence:") while count < nterms:
print(n1)
nth = n1 + n2
# update values n1 = n2
n2 = nth count += 1

Output

How many terms? 7 Fibonacci sequence:

0
1
1
2
3
5
8

Time and Space Complexity of Space Optimized Method
● The time complexity of the Fibonacci series is **T(N) i.e, linear**. We have to find the sum of two terms and it is repeated n times depending on the value of n.

● The space complexity of the Fibonacci series using dynamic programming is **O(1)**.

Time Complexity and Space Complexity of Dynamic Programming
● The time complexity of the above code is **T(N) i.e, linear**. We have to find the sum of two terms and it is repeated n times depending on the value of n.

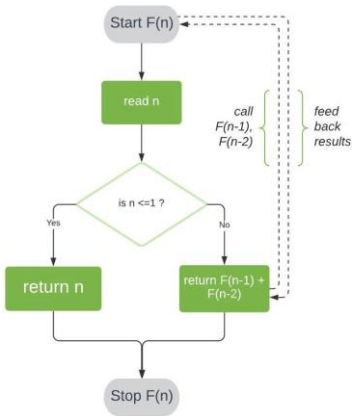
The space complexity of the above code is **O(N)**.

Method 2 (Use Recursion)

Let's start by defining F(n) as the function that returns the value of Fn.
To evaluate *F*(*n*) for *n* > 1, we can reduce our problem into two smaller problems of the same kind: *F*(*n*-1) and *F*(*n*-2). We can further reduce *F*(*n*-1) and *F*(*n*-2) to *F*((*n*-1)-1) and *F*((*n*-1)-2); and *F*((*n*-2)-1) and *F*((*n*-2)-2), respectively.
If we repeat this reduction, we'll eventually reach our known base cases and, thereby, obtain a solution to *F*(*n*).
Employing this logic, our algorithm for *F*(*n*) will have two steps:
1. Check if $n \leq 1$. If so, return *n*.

2. Check if $n > 1$. If so, call our function *F* with inputs *n*-1 and *n*-2, and return the sum of the two results.

Here's a visual representation of this algorithm:



Python program to display the Fibonacci sequence def recur_fibo(n):
if n <= 1: return n
else:
return(recur_fibo(n-1) + recur_fibo(n-2)) nterms = 7
check if the number of terms is valid if nterms <= 0:
print("Plese enter a positive integer") else:
print("Fibonacci sequence:") for i in range(nterms):

```
print(recur_fibo(i))
```
Output
```
Fibonacci sequence:
0
1
1
2
3
5
8
```

Time and Space Complexity

● The time complexity of the above code is **T(2^N) i.e, exponential**.

The Space complexity of the above code is **O(N) for a recursive series.**

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

Applications of Fibonacci Series

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

● It is used in the grouping of numbers and used to study different other special mathematical sequences.

● It finds application in Coding (computer algorithms, distributed systems, etc). For example, Fibonacci series are important in the computational run-time analysis of Euclid's algorithm, used for determining the GCF of two integers.

● It is applied in numerous fields of science like quantum mechanics, cryptography, etc.

● In finance market trading, Fibonacci retracement levels are widely used in technical analysis.

Conclusion- In this way we have explored Concept of Fibonacci series using recursive and non recursive method and also learn time and space complexity

Reference link

● https://www.scaler.com/topics/fibonacci-series-in-c/
● https://www.baeldung.com/cs/fibonacci-computational-complexity

Fibonacci Numbers (Non-recursive)

```
import java.util.Scanner;
public class Main{
   public static void fib(int n){
      int a=0;
      int b=1;
      for(int i=0;i<n;i++){
         System.out.print(a+" ");
         int c=a+b;
         a=b;
         b=c;
      }
   }
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      int n=sc.nextInt();
      fib(n);
   }
}
```

#OUTPUT:

```
PS C:\Users\hp\OneDrive\Desktop\JAVA> javac Main.java
PS C:\Users\hp\OneDrive\Desktop\JAVA> java Main.java
10
0 1 1 2 3 5 8 13 21 34
PS C:\Users\hp\OneDrive\Desktop\JAVA>
```

Fibonacci Numbers (recursive)

```
import java.util.Scanner;
public class Main{
   public static int fib(int n){
      if(n==0 || n==1){
         return n;
      }
      return fib(n-1)+fib(n-2);
   }
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      int n=sc.nextInt();
      for(int i=0;i<n;i++){
         System.out.print(fib(i)+" ");
      }
   }
}
```

OUTPUT:

```
PS C:\Users\hp\OneDrive\Desktop\JAVA> java Main.java
10
0 1 1 2 3 5 8 13 21 34
PS C:\Users\hp\OneDrive\Desktop\JAVA> java Main.java
15
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
PS C:\Users\hp\OneDrive\Desktop\JAVA>
```

Assignment No: 2

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

Objective of the Assignment: Students should be able to understand and solve Huffman Encoding using greedy method

Prerequisite:
1. Basic of Python or Java Programming

2. Concept of Greedy method

3. Huffman Encoding concept

Contents for Theory:
1. Greedy Method
2. Huffman Encoding
3. Example solved using huffman encoding

What is a Greedy Method?
● A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.
● The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach.
● This algorithm may not produce the best result for all the problems. It's because it always goes for the local best choice to produce the global best result.
Advantages of Greedy Approach
● The algorithm is easier to describe.
● This algorithm can perform better than other algorithms (but, not in all cases).
Drawback of Greedy Approach
● As mentioned earlier, the greedy algorithm doesn't always produce the optimal solution. This is the major disadvantage of the algorithm
● For example, suppose we want to find the longest path in the graph below from root to leaf.

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

Huffman Encoding
● Huffman Coding is a technique of compressing data to reduce its size without losing any of the details. It was first developed by David Huffman.
● Huffman Coding is generally useful to compress the data in which there are frequently occurring
characters.
● Huffman Coding is a famous Greedy Algorithm.
● It is used for the lossless compression of data.
● It uses variable length encoding.

● It assigns variable length code to all the characters.
● The code length of a character depends on how frequently it occurs in the given text.
● The character which occurs most frequently gets the smallest code.
● The character which occurs least frequently gets the largest code.
● It is also known as Huffman Encoding.

Prefix Rule-
● Huffman Coding implements a rule known as a prefix rule.
● This is to prevent the ambiguities while decoding.
● It ensures that the code assigned to any character is not a prefix of the code assigned to any other character

Major Steps in Huffman Coding-
There are two major steps in Huffman Coding-
1. Building a Huffman Tree from the input characters.
2. Assigning code to the characters by traversing the Huffman Tree.

How does Huffman Coding work?
Suppose the string below is to be sent over a network.

Each character occupies 8 bits. There are a total of 15 characters in the above string. Thus, a total of 8 * 15 = 120 bits are required to send this string.

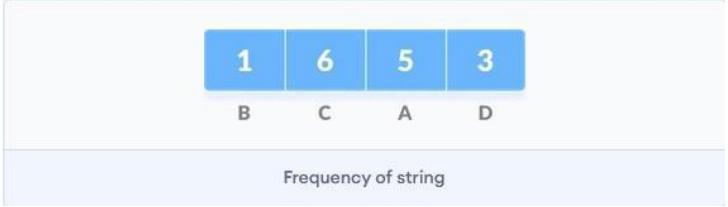
● Using the Huffman Coding technique, we can compress the string to a smaller size.

Huffman coding first creates a tree using the frequencies of the character and then generates code for each character.

● Once the data is encoded, it has to be decoded. Decoding is done using the same tree.

● Huffman Coding prevents any ambiguity in the decoding process using the concept of prefix code ie. a code associated with a character should not be present in the prefix of any other code. The tree created above helps in maintaining the property.

● Huffman coding is done with the help of the following steps.
1. Calculate the frequency of each character in the string.

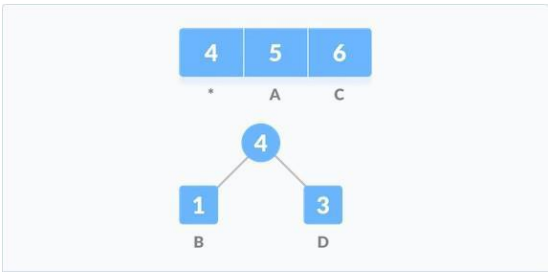


2. Sort the characters in increasing order of the frequency. These are stored in a priority queue Q.

Characters sorted according to the frequency

3. Make each unique character as a leaf node.

4. Create an empty node z. Assign the minimum frequency to the left child of z and assign the second minimum frequency to the right child of z. Set the value of the z as the sum of the above two minimum frequencies.

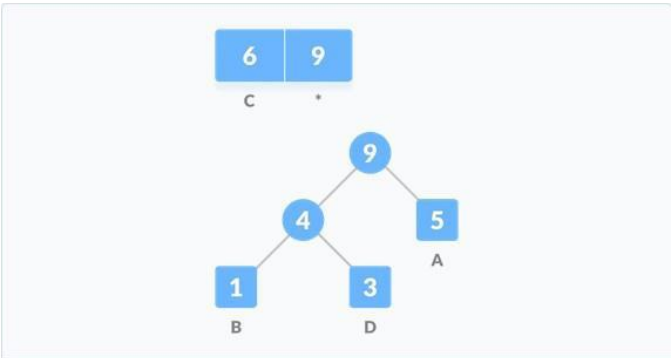


Getting the sum of the least numbers

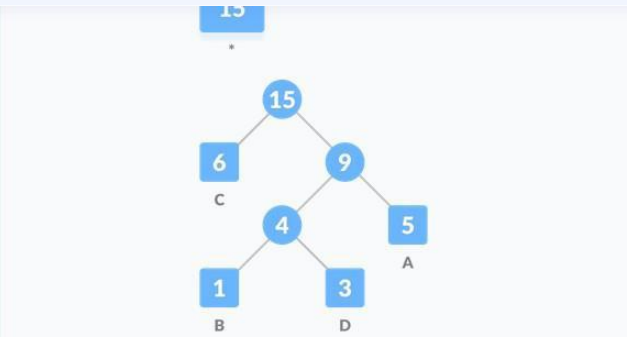
5. Remove these two minimum frequencies from Q and add the sum into the list of frequencies (* denote the internal nodes in the figure above).

6. Insert node z into the tree.

7. Repeat steps 3 to 5 for all the characters.

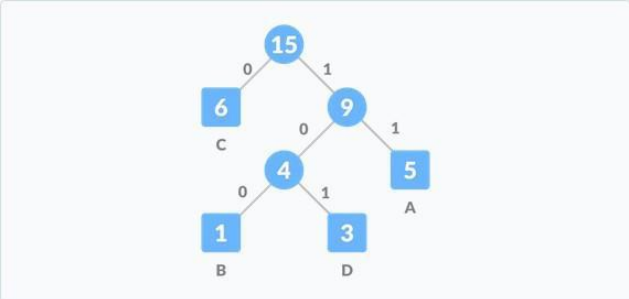


Repeat steps 3 to 5 for all the characters.



Repeat steps 3 to 5 for all the characters.

8. For each non-leaf node, assign 0 to the left edge and 1 to the right edge



Assign 0 to the left edge and 1 to the right edge

For sending the above string over a network, we have to send the tree as well as the above compressed-code. The total size is given by the table below.

Character	Frequency	Code	Size
A	5	11	5*2 = 10
B	1	100	1*3 = 3
C	6	0	6*1 = 6
D	3	101	3*3 = 9
4 * 8 = 32 bits	15 bits		28 bits

Without encoding, the total size of the string was 120 bits. After encoding the size is reduced to 32
+ 15 + 28 = 75.

Example:

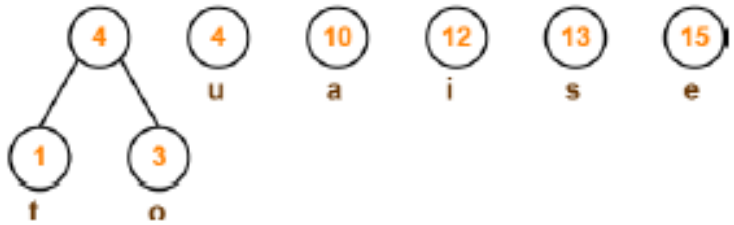
A file contains the following characters with the frequencies as shown. If Huffman Coding is used for data compression, determine-

1. Huffman Code for each character
2. Average code length
3. Length of Huffman encoded message (in bits)

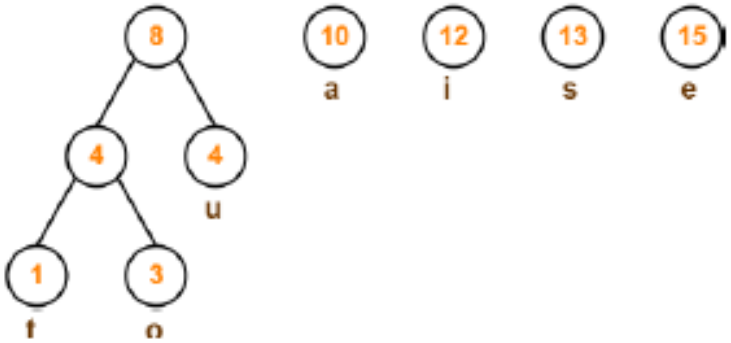
Characters	Frequencies
a	10
e	15
i	12
o	3
u	4
s	13
t	1

Step-01:

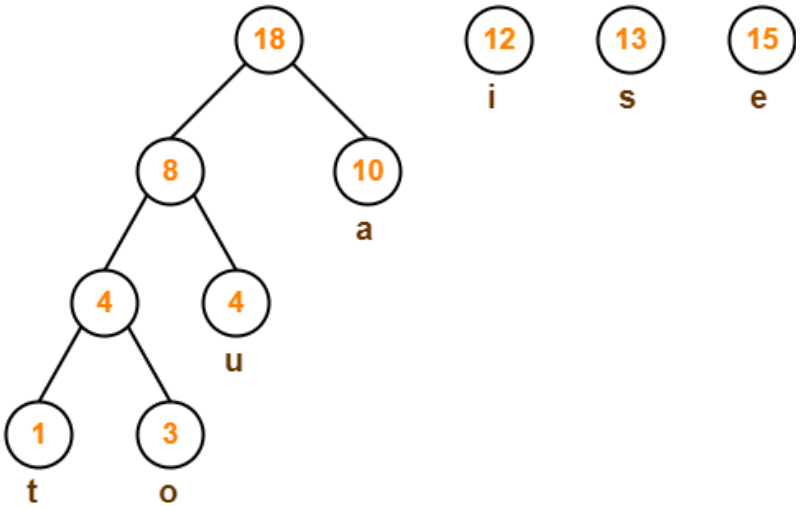
Step-02:



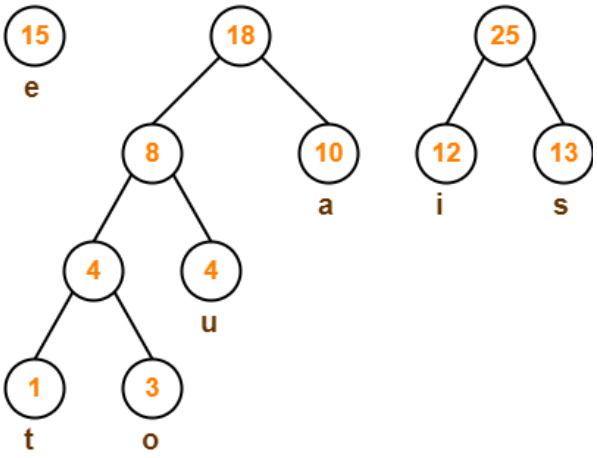
Step-03:

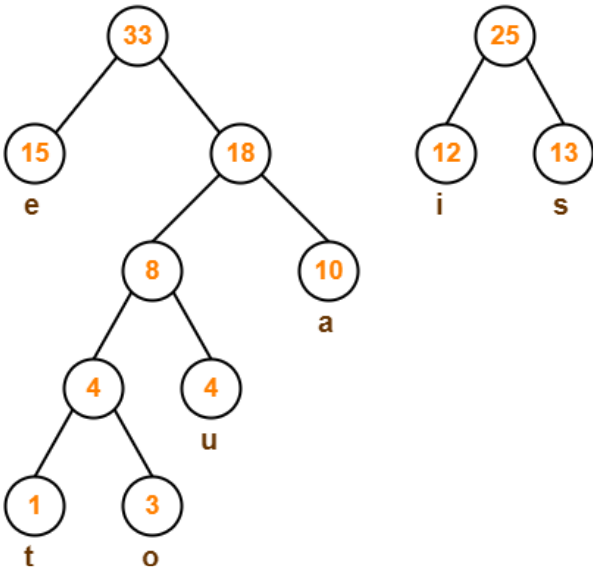


Step-04:

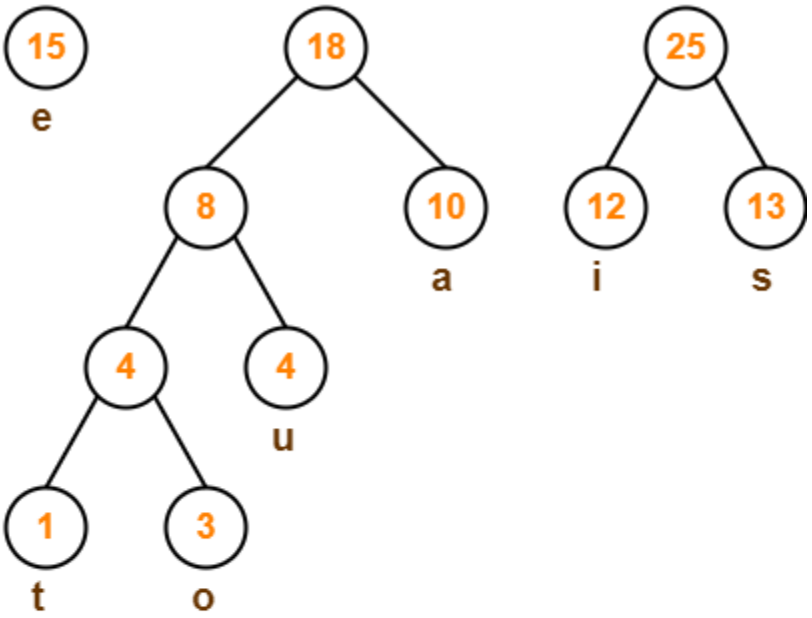


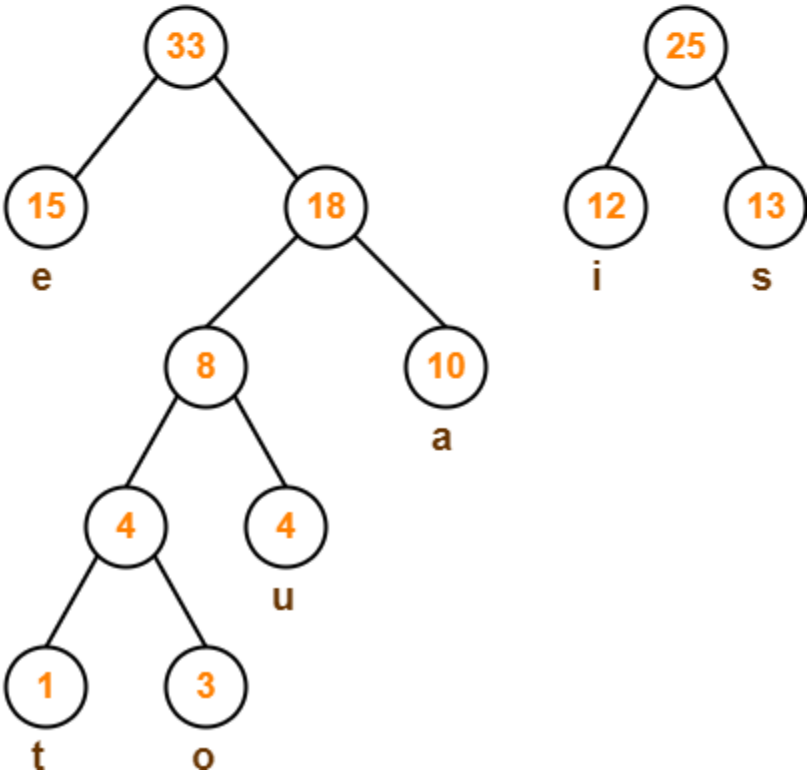
Step-06:


15
e
18
8
10
a
4
4
u
1
3
t
o
25
12
13
i
s

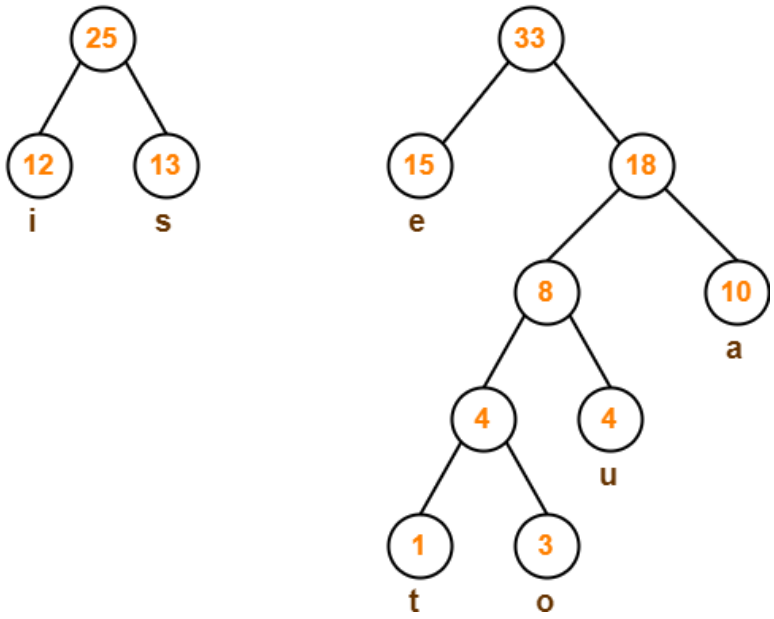

33
15
e
18
8
10
a
4
4
u
1
3
t
o
25
12
13
i
s

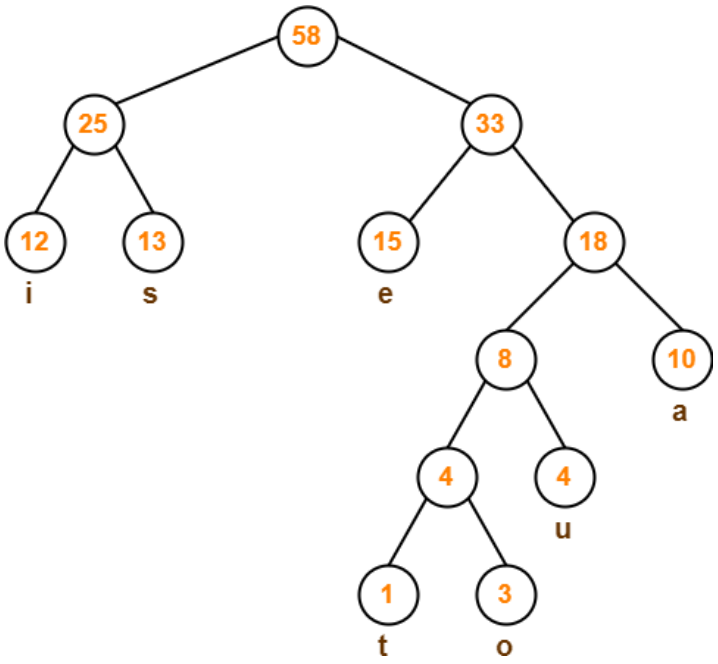
Step-06:

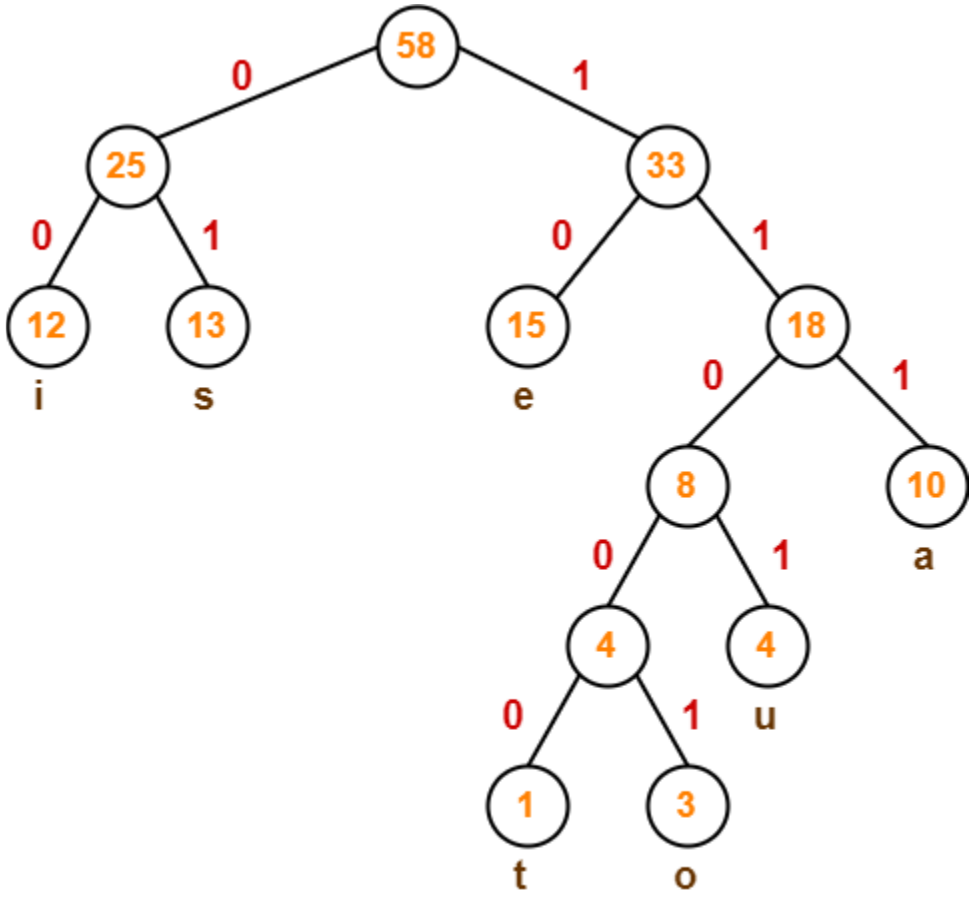

15
e
18
8
10
a
4
4
u
1
3
t
o
25
12
13
i
s


33
15
e
18
8
10
a
4
4
u
1
3
t
o
25
12
13
i
s

Step-07:


25
12
13
i
s
33
15
18
e
8
10
a
4
4
u
1
3
t
o


58
25
33
12
13
15
18
i
s
e
8
10
a
4
4
u
1
3
t
o



Huffman Tree

To write Huffman Code for any character, traverse the Huffman Tree from root node to the leaf node of that character. Following this rule, the Huffman Code for each character is-
a = 111
e = 10
i = 00
o = 11001
u = 1101
s = 01
t = 11000

Time Complexity-
The time complexity analysis of Huffman Coding is as follows-
- extractMin( ) is called 2 x (n-1) times if there are n nodes.
- As extractMin( ) calls minHeapify( ), it takes O(logn) time.

Thus, Overall time complexity of Huffman Coding becomes **O(nlogn)**.

Conclusion- In this way we have explored Concept ofHuffman Encoding using greedy method

Reference link

- https://towardsdatascience.com/huffman-encoding-python-implementation-8448c3654328
- https://www.programiz.com/dsa/huffman-coding#cpp-code
- https://www.gatevidyalay.com/tag/huffman-coding-example-ppt/

Huffman Coding (PROGRAM)

```
import java.util.PriorityQueue;
import java.util.Scanner;
import java.util.Comparator;

class Huffman {

	public static void printCode(HuffmanNode root, String s)
	{

		if (root.left
				== null
			&& root.right
				== null
			&& Character.isLetter(root.c)) {

			System.out.println(root.c + ":" + s);

			return;
		}

		printCode(root.left, s + "0");
		printCode(root.right, s + "1");
	}

	public static void main(String[] args)
	{

		Scanner s = new Scanner(System.in);

		int n = 6;
		char[] charArray = { 'a', 'b', 'c', 'd', 'e', 'f' };
		int[] charfreq = { 5, 9, 12, 13, 16, 45 };

		PriorityQueue<HuffmanNode> q
			= new PriorityQueue<HuffmanNode>(n, new MyComparator());

		for (int i = 0; i < n; i++) {

			HuffmanNode hn = new HuffmanNode();

			hn.c = charArray[i];
			hn.data = charfreq[i];

			hn.left = null;
			hn.right = null;

			q.add(hn);
```

```
                }

                HuffmanNode root = null;

                while (q.size() > 1) {

                        HuffmanNode x = q.peek();
                        q.poll();

                        HuffmanNode y = q.peek();
                        q.poll();

                        HuffmanNode f = new HuffmanNode();

                        f.data = x.data + y.data;
                        f.c = '-';

                        f.left = x;

                        f.right = y;

                        root = f;

                        q.add(f);
                }

                printCode(root, "");
        }
}

class HuffmanNode {

        int data;
        char c;

        HuffmanNode left;
        HuffmanNode right;
}

class MyComparator implements Comparator<HuffmanNode> {
        public int compare(HuffmanNode x, HuffmanNode y)
        {

                return x.data - y.data;
        }
}
```

OUTPUT:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\hp\OneDrive\Desktop\JAVA> javac Huffman.java
PS C:\Users\hp\OneDrive\Desktop\JAVA> java Huffman.java
f:0
c:100
d:101
a:1100
b:1101
e:111
PS C:\Users\hp\OneDrive\Desktop\JAVA>
```

Assignment No: 3

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

Objective of the Assignment: Students should be able to understand and solve fractional Knapsack problems using a greedy method.

Prerequisite:
1. Basic of Python or Java Programming
2. Concept of Greedy method
3. fractional Knapsack problem

Contents for Theory:
1. Greedy Method
2. Fractional Knapsack problem
3. Example solved using fractional Knapsack problem

What is a Greedy Method?
● A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.

● The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach.

● This algorithm may not produce the best result for all the problems. It's because it always goes for the local best choice to produce the global best result.

Advantages of Greedy Approach
● The algorithm is **easier to describe**.

● This algorithm can **perform better** than other algorithms (but, not in all cases).

Drawback of Greedy Approach
● As mentioned earlier, the greedy algorithm doesn't always produce the optimal solution. This is the major disadvantage of the algorithm

● For example, suppose we want to find the longest path in the graph below from root to leaf.

Greedy Algorithm
1. To begin with, the solution set (containing answers) is empty.

2. At each step, an item is added to the solution set until a solution is reached.

3. If the solution set is feasible, the current item is kept.

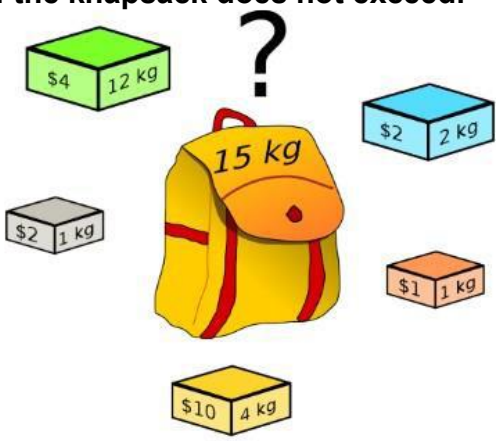
4. Else, the item is rejected and never considered again.

Knapsack Problem

You are given the following-
● A knapsack (kind of shoulder bag) with limited weight capacity.

● Few items each having some weight and value.
The problem states-
Which items should be placed into the knapsack such that-
● **The value or profit obtained by putting the items into the knapsack is maximum.**

● **And the weight limit of the knapsack does not exceed.**



Knapsack Problem

Knapsack Problem Variants

Knapsack problem has the following two variants-
1. Fractional Knapsack Problem
2. 0/1 Knapsack Problem

Fractional Knapsack Problem-

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

Fractional Knapsack Problem Using Greedy Method-

Fractional knapsack problem is solved using greedy method in the following steps-
Step-01:
For each item, compute its value / weight ratio.
Step-02:
Arrange all the items in decreasing order of their value / weight ratio.
Step-03:
Start putting the items into the knapsack beginning from the item with the highest ratio. Put as many items as you can into the knapsack.

Problem-

For the given set of items and knapsack capacity = 60 kg, find the optimal solution for the fractional knapsack problem making use of greedy approach.

Item	Weight	Value
1	5	30
2	10	40
3	15	45
4	22	77
5	25	90

n = 5

w = 60 kg

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

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

Solution-

Step-01:

Compute the value / weight ratio for each item-

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

Step-02:

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

I1 I2 I5 I4 I3

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

Step-03:

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

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

Now,

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

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

Total cost of the knapsack

= 160 + (20/22) x 77

= 160 + 70

= 230 units

Time Complexity-

● The main time taking step is the sorting of all items in decreasing order of their value / weight ratio.

● If the items are already arranged in the required order, then while loop takes O(n) time.

● **The average time complexity of** Quick Sort is O(nlogn).

● **Therefore, total time taken including the sort is O(nlogn).**

Conclusion-In this way we have explored Concept of Fractional Knapsack using greedy method

Reference link

- https://www.gatevidyalay.com/fractional-knapsack-problem-using-greedy-approach/

Fractional KnapSack Problem

```
import java.util.Arrays;
import java.util.Comparator;
public class fractionalKnapsack {
    public static void main(String[] args) {
        int val[]={60, 100, 120};
        int weight[]={10, 20, 30};
        int capacity=50;

        double ratio[][]=new double[val.length][2];
        // 0th col==idx;  1st col==ratios;
        for(int i=0;i<val.length;i++){
            ratio[i][0]=i;
            ratio[i][1]=val[i]/(double)weight[i];
        }

        Arrays.sort(ratio, Comparator.comparingDouble(o -> o[1]));

        int finalValue=0;
        for(int i=ratio.length-1;i>=0;i--){
            int idx=(int)ratio[i][0];
            if(capacity >= weight[idx]){ // include full
                finalValue=finalValue+val[idx];
                capacity=capacity-weight[idx];
            }else{
                //include fractional
                finalValue +=(ratio[i][1]*capacity);
                capacity=0;
                break;
            }
        }
```

```
        System.out.println("final Value: "+ finalValue);
    }
}
//OUTPUT
PS C:\Users\hp\OneDrive\Desktop\JAVA> java fractionalKnapsack.java
final Value: 240
PS C:\Users\hp\OneDrive\Desktop\JAVA>
```

Assignment No: 4

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

Objective of the Assignment: Students should be able to understand and solve 0-1 Knapsack problem using dynamic programming

Prerequisite:
1. Basic of Python or Java Programming
2. Concept of Dynamic Programming
3. 0/1 Knapsack problem

Contents for Theory:
1. Greedy Method
2. 0/1 Knapsack problem
3. Example solved using 0/1 Knapsack problem

What is Dynamic Programming?
● Dynamic Programming is also used in optimization problems. Like divide-and-conquer method, Dynamic Programming solves problems by combining the solutions of subproblems.

● Dynamic Programming algorithm solves each sub-problem just once and then saves its answer in a table, thereby avoiding the work of re-computing the answer every time.

● Two main properties of a problem suggest that the given problem can be solved using Dynamic Programming. These properties are **overlapping sub-problems and optimal substructure**.

● Dynamic Programming also combines solutions to sub-problems. It is mainly used where the solution of one sub-problem is needed repeatedly. The computed solutions are stored in a table, so that these don't have to be re-computed. Hence, this technique is needed where overlapping sub-problem exists.

● For example, Binary Search does not have overlapping sub-problem. Whereas recursive program of Fibonacci numbers have many overlapping sub-problems.

Steps of Dynamic Programming Approach
Dynamic Programming algorithm is designed using the following four steps −
● Characterize the structure of an optimal solution.
● Recursively define the value of an optimal solution.
● Compute the value of an optimal solution, typically in a bottom-up fashion.
● Construct an optimal solution from the computed information.

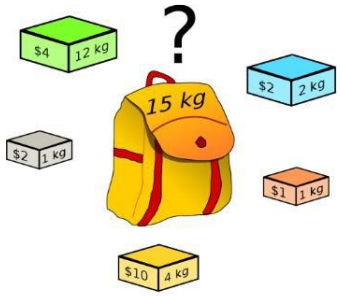
Applications of Dynamic Programming Approach
● Matrix Chain Multiplication
● Longest Common Subsequence
● Travelling Salesman Problem

Knapsack Problem
You are given the following-
● A knapsack (kind of shoulder bag) with limited weight capacity.

● **Few items each having some weight and value.**

The problem states-
Which items should be placed into the knapsack such that-
● **The value or profit obtained by putting the items into the knapsack is maximum.**

● And the weight limit of the knapsack does not exceed.



Knapsack Problem

Knapsack Problem Variants

Knapsack problem has the following two variants-
1. Fractional Knapsack Problem
2. 0/1 Knapsack Problem

0/1 Knapsack Problem-

In 0/1 Knapsack Problem,
● As the name suggests, items are indivisible here.
● We can not take a fraction of any item.
● We have to either take an item completely or leave it completely.
● It is solved using a dynamic programming approach.

0/1 Knapsack Problem Using Greedy Method-

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

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

Step-01:

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

	0	1	2	3	.....	W
0	0	0	0	0	.....	0
1	0					
2	0					
	.....					
n	0					

T-Table

Step-02:

Start filling the table row wise top to bottom from left to right. Use the following formula-

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

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

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

Step-03:

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

Problem-.

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

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

n = 4

w = 5 kg

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

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

Solution-

Given

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

Step-01:

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

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

T-Table

Step-02:

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

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

Finding T(1,1)-

We have,

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

Substituting the values, we get-

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

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

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

$T(1,1) = 0$

Finding T(1,2)-

We have,

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

Substituting the values, we get-

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

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

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

T(1,2) = 3

Finding T(1,3)-

We have,

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

Substituting the values, we get-

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

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

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

T(1,3) = 3

Finding T(1,4)-

We have,

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

Substituting the values, we get-

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

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

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

T(1,4) = 3

Finding T(1,5)-

We have,

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

Substituting the values, we get-

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

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

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

T(1,5) = 3

Finding T(2,1)-

We have,

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

Substituting the values, we get-

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

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

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

T(2,1) = 0

Finding T(2,2)-

We have,

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

Substituting the values, we get-

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

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

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

T(2,2) = 3

Finding T(2,3)-

We have,

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

Substituting the values, we get-

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

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

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

T(2,3) = 4

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

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

T-Table

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

Identifying Items To Be Put Into Knapsack

Following Step-04,
- We mark the rows labelled “1” and “2”.
- Thus, items that must be put into the knapsack to obtain the maximum value 7 are-

Item-1 and Item-2

Time Complexity-

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

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

Reference link
- https://www.gatevidyalay.com/0-1-knapsack-problem-using-dynamic-programming-approach/
- https://www.youtube.com/watch?v=mMhC9vuA-70

0/1 KnapSack Problem

```
class knapsack01 {

	static int max(int a, int b)
	{
		return (a > b) ? a : b;
	}

	static int knapSack(int W, int wt[],
						int val[], int n)
	{
		int i, w;
		int K[][] = new int[n + 1][W + 1];

		for (i = 0; i <= n; i++)
		{
			for (w = 0; w <= W; w++)
			{
				if (i == 0 || w == 0)
					K[i][w] = 0;
				else if (wt[i - 1] <= w)
					K[i][w]
						= max(val[i - 1]
						+ K[i - 1][w - wt[i - 1]],
						K[i - 1][w]);
				else
					K[i][w] = K[i - 1][w];
			}
		}

		return K[n][W];
	}

	// Driver code
	public static void main(String args[])
	{
		int val[] = new int[] { 60, 100, 120 };
		int wt[] = new int[] { 10, 20, 30 };
		int W = 50;
		int n = val.length;
		System.out.println(knapSack(W, wt, val, n));
	}
}
```

OUTPUT:

```
PS C:\Users\hp\OneDrive\Desktop\JAVA> javac knapsack01.java
PS C:\Users\hp\OneDrive\Desktop\JAVA> java knapsack01.java
220
```

Assignment No: 5

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

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

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

Contents for Theory:
1. Introduction to Backtracking
2. N-Queen Problem

Introduction to Backtracking

● Many problems are difficult to solve algorithmically. Backtracking makes it possible to solve at least some large instances of difficult combinatorial problems. Suppose we have to make a series of decisions among various choices, where

● We don't have enough information to know what to choose

● Each decision leads to a new set of choices.

● Some sequence of choices (more than one choices) may be a solution to your problem.

What is backtracking?

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

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

solution) and then check if we can proceed further with this sub-solution or not. If not, then we just come back and change it.

Thus, the general steps of backtracking are:

● start with a sub-solution

● check if this sub-solution will lead to the solution or not

● If not, then come back and change the sub-solution and continue again

Applications of Backtracking:

● N Queens Problem

● Sum of subsets problem

● Graph coloring

● Hamiltonian cycles.

N queens on NxN chessboard

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

N-Queens Problem:

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

What is the N Queen Problem?

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

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

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

1) Start in the leftmost column

2) If all queens are place return true

3) Try all rows in the current column

Do following for every tried row.

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

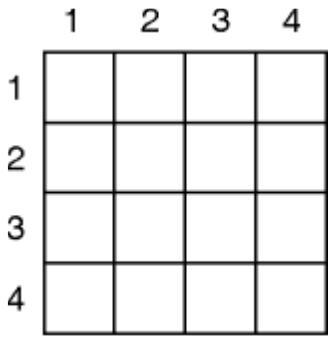
b) If placing the queen in [row, column] leads to a solution then return true.

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

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

4- Queen Problem

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



4 x 4 Chessboard

● We have to arrange four queens, Q1, Q2, Q3 and Q4 in 4 x 4 chess board. We will put with queen in ith row. Let us start with position (1, 1). Q1 is the only queen, so there is no issue. partial solution is <1>

● We cannot place Q2 at positions (2, 1) or (2, 2). Position (2, 3) is acceptable. the partial solution is <1, 3>.

● Next, Q3 cannot be placed in position (3, 1) as Q1 attacks her. And it cannot be placed at (3, 2), (3, 3) or (3, 4) as Q2 attacks her. There is no way to put Q3 in the third row. Hence, the algorithm

backtracks and goes back to the previous solution and readjusts the position of queen Q2. Q2 is moved from positions (2, 3) to (2, 4). Partial solution is <1, 4>

● Now, Q3 can be placed at position (3, 2). Partial solution is <1, 4, 3>.

● Queen Q4 cannot be placed anywhere in row four. So again, backtrack to the previous solution and readjust the position of Q3. Q3 cannot be placed on (3, 3) or(3, 4). So the algorithm backtracks even further.

● All possible choices for Q2 are already explored, hence the algorithm goes back to partial solution <1> and moves the queen Q1 from (1, 1) to (1, 2). And this process continues until a solution is found.

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

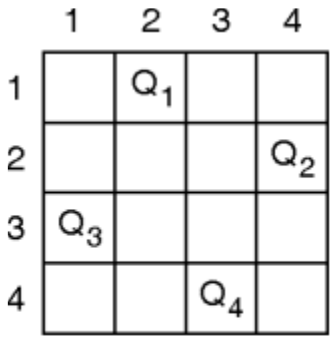


Fig. (a): Solution – 1

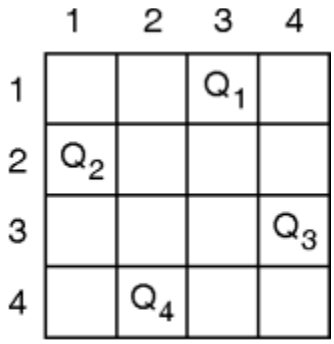
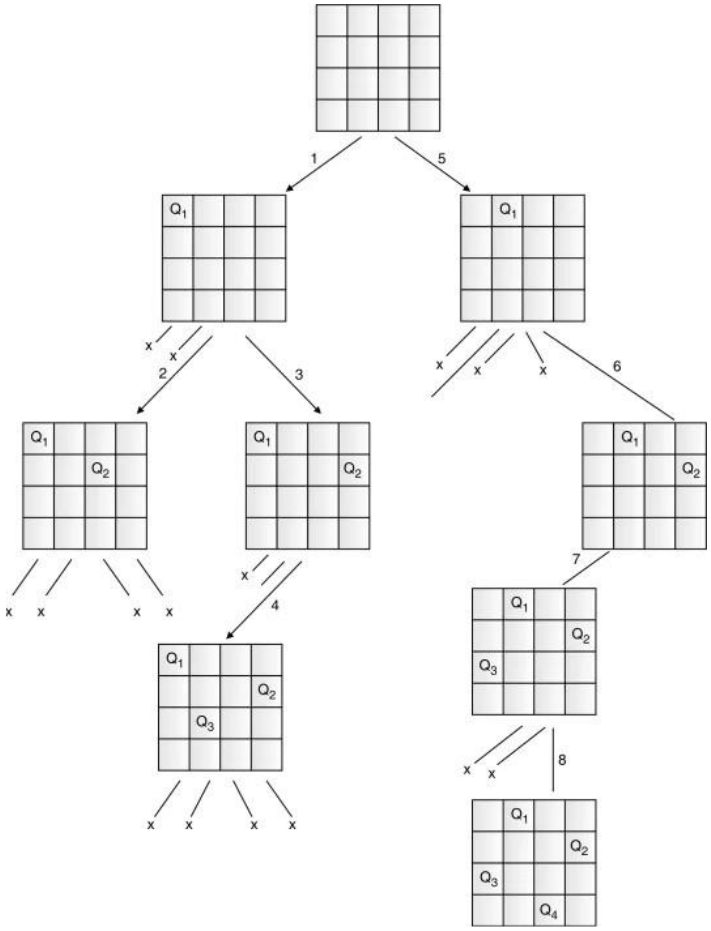


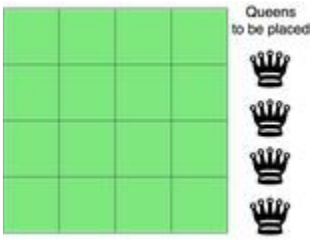
Fig. (b): Solution – 2

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

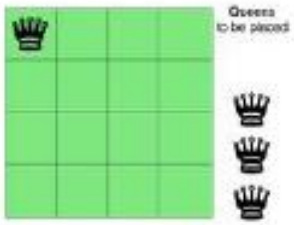


The solution of the 4-queen problem can be seen as four tuples (x_1, x_2, x_3, x_4), where x_i represents the column number of queen Q_i. Two possible solutions for the 4-queen problem are (2, 4, 1, 3) and (3, 1, 4, 2).

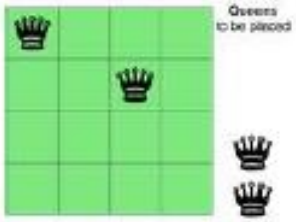
Explanation :

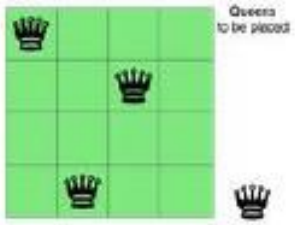


The above picture shows an NxN chessboard and we have to place N queens on it. So, we will start by placing the first queen.



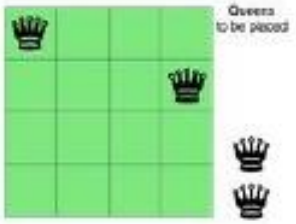
Now, the second step is to place the second queen in a safe position and then the third queen.



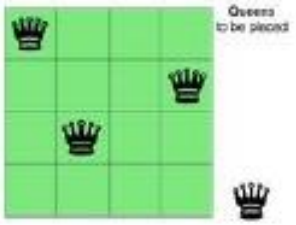


Now, you can see that there is no safe place where we can put the last queen. So, we will just change the position of the previous queen. And this is backtracking.

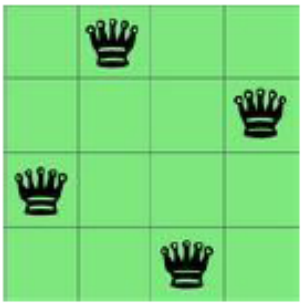
Also, there is no other position where we can place the third queen so we will go back one more step and change the position of the second queen.



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

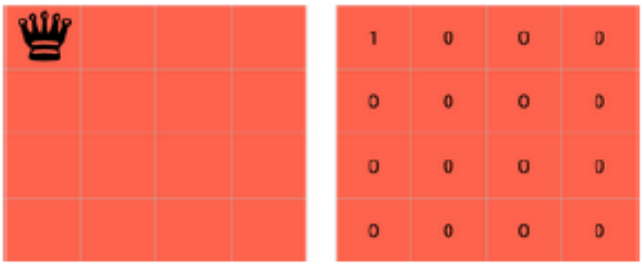


We will continue this process and finally, we will get the solution as shown below.

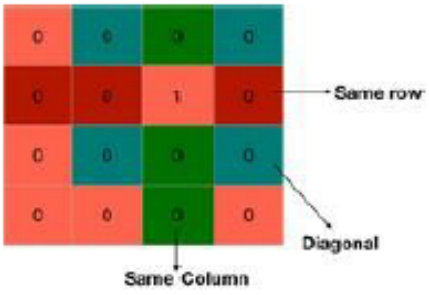


We need to check if a cell (i, j) is under attack or not. For that, we will pass these two in our function along with the chessboard and its size - IS-ATTACK(i, j, board, N).

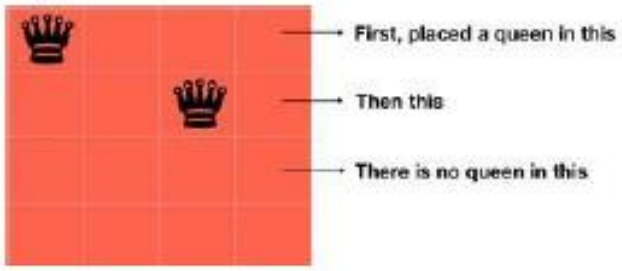
If there is a queen in a cell of the chessboard, then its value will be 1, otherwise, 0.



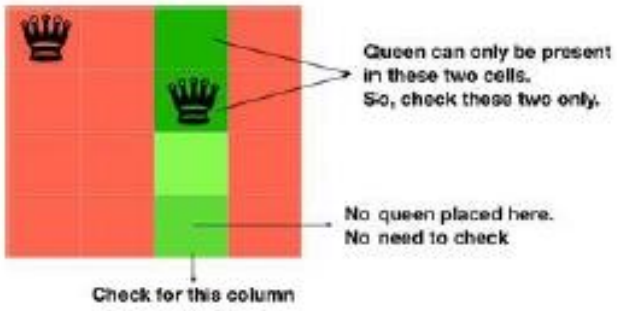
The cell (i,j) will be under attack in three condition - if there is any other queen in row i, if there is any other queen in the column j or if there is any queen in the diagonals.



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



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



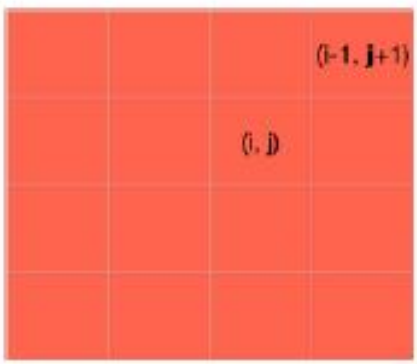
for k in 1 to i-1

if board[k][j]==1

return TRUE

Now, we need to check for the diagonal. We know that all the rows below the row i are empty, so we need to check only for the diagonal elements which above the row i.

If we are on the cell (i, j), then decreasing the value of i and increasing the value of j will make us traverse over the diagonal on the right side, above the row i.



k = i-1

l = j+1

while k>=1 and l<=N

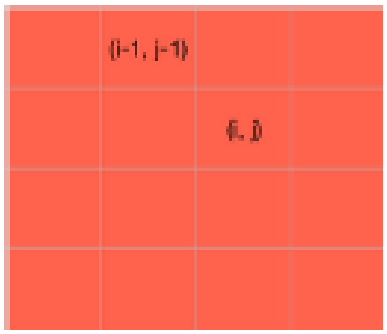
 if board[k][l] == 1

 return TRUE

 k=k-1

 l=l+1

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



k = i-1

l = j-1

while k>=1 and l>=1

 if board[k][l] == 1

 return TRUE

 k=k-1

 l=l-1

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

We can write the entire code as:

```
IS-ATTACK(i, j, board, N)
 // checking in the column j
 for k in 1 to i-1
  if board[k][j]==1
   return TRUE

 // checking upper right diagonal
 k = i-1
 l = j+1
 while k>=1 and l<=N
  if board[k][l] == 1
   return TRUE
  k=k+1
  l=l+1

 // checking upper left diagonal
 k = i-1
 l = j-1
```

```
while k>=1 and l>=1
  if board[k][l] == 1
    return TRUE
  k=k-1
  l=l-1

return FALSE
```

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

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

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

```
if n==0
  return TRUE
```

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

```
for j in 1 to N
  if !IS-ATTACK(row, j, board, N)
    board[row][j] = 1
```

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

```
for j in 1 to N
  ...
```

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

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

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

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

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

```
N-QUEEN(row, n, N, board)
..
 return FALSE
```

```
N-QUEEN(row, n, N, board)
 if n==0
  return TRUE

 for j in 1 to N
  if !IS-ATTACK(row, j, board, N)
   board[row][j] = 1
```

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

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

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

Reference link

- https://www.codesdope.com/blog/article/backtracking-explanation-and-n-queens-problem/

N-Queen Problem

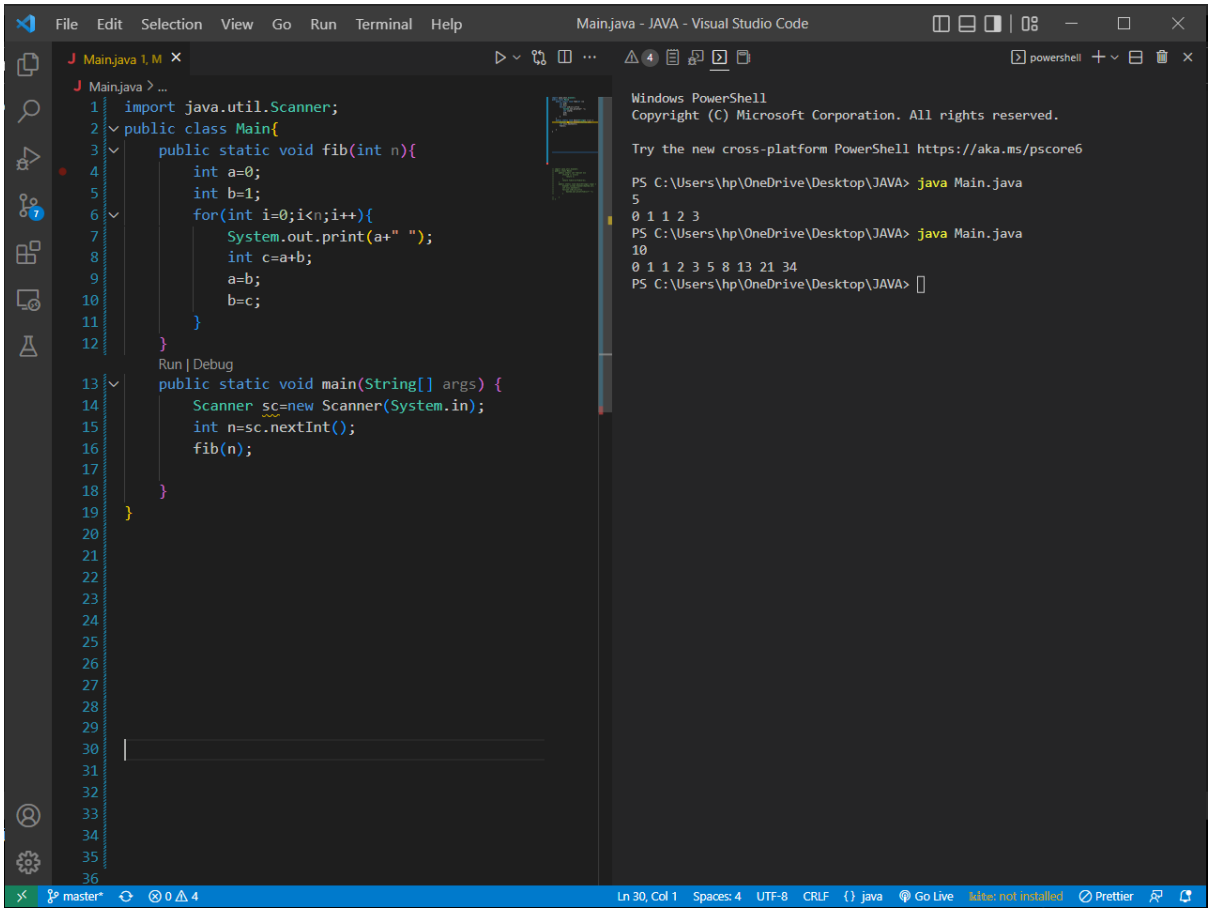
```
public class NQueenProblem {

	final int N = 4;


	void printSolution(int board[][])
	{
		for (int i = 0; i < N; i++) {
			for (int j = 0; j < N; j++)
				System.out.print(" " + board[i][j]
												+ " ");
			System.out.println();
		}
	}


	boolean isSafe(int board[][], int row, int col)
	{
		int i, j;


		for (i = 0; i < col; i++)
			if (board[row][i] == 1)
				return false;


		for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
			if (board[i][j] == 1)
				return false;


		for (i = row, j = col; j >= 0 && i < N; i++, j--)
			if (board[i][j] == 1)
				return false;


		return true;
```

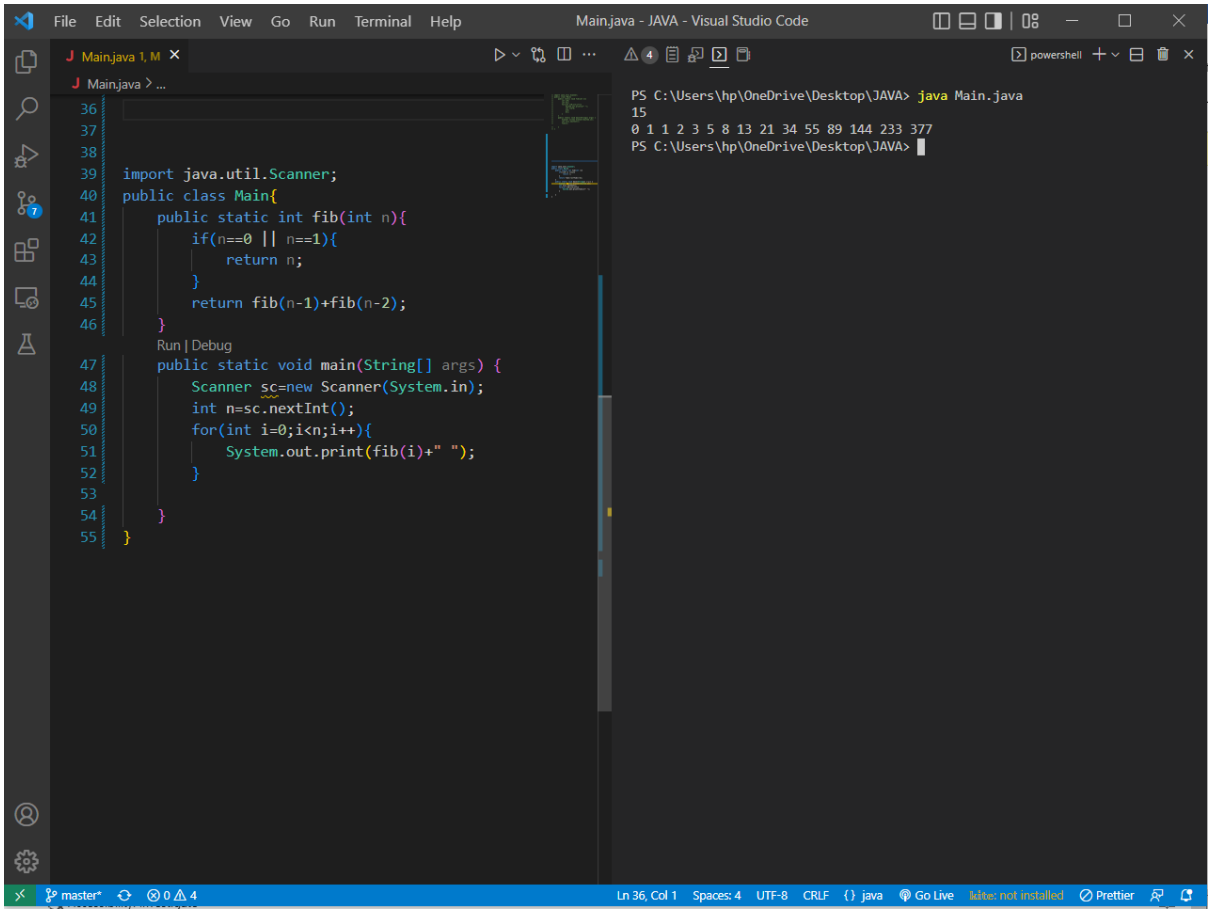
```
}

boolean solveNQUtil(int board[][], 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; // BACKTRACK
		}
	}

	return false;
}

boolean solveNQ()
{
	int board[][] = { { 0, 0, 0, 0 },
					{ 0, 0, 0, 0 },
					{ 0, 0, 0, 0 },
					{ 0, 0, 0, 0 } };
```

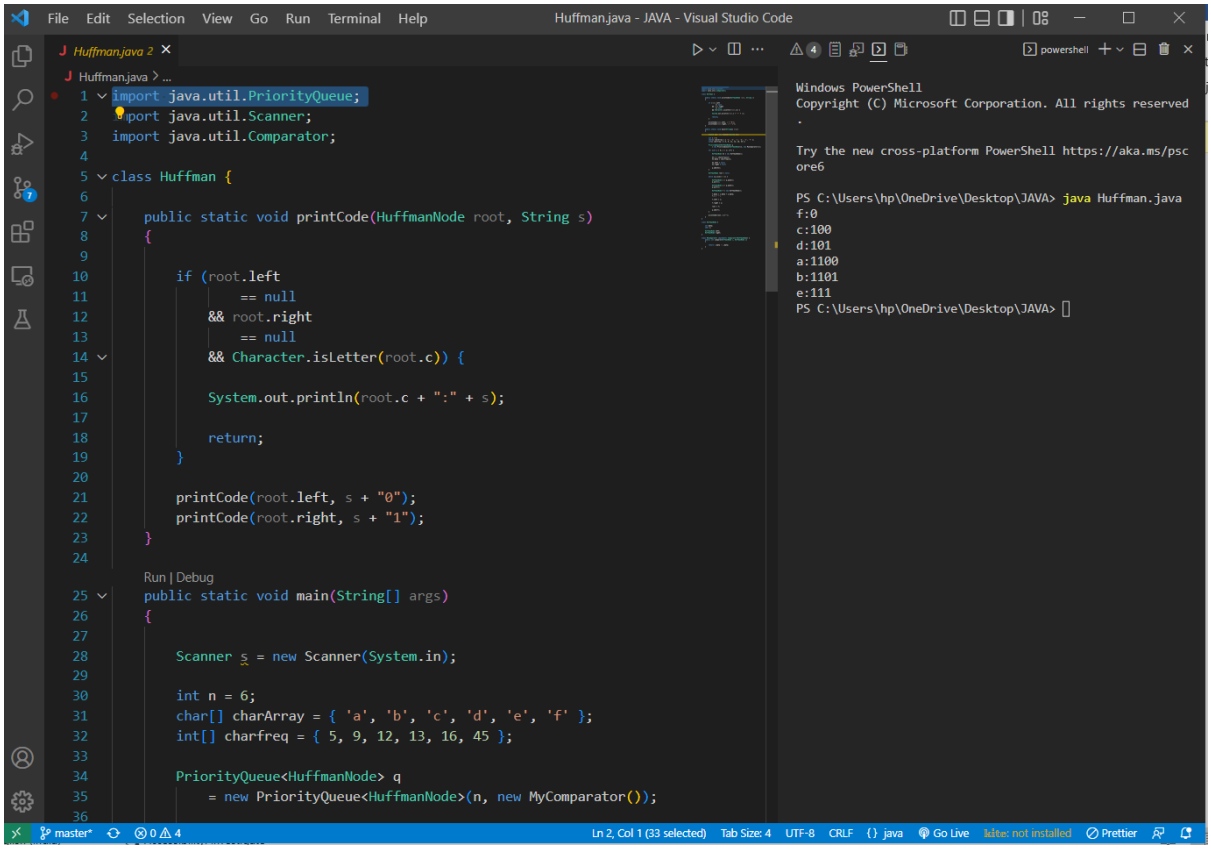
```
                if (solveNQUtil(board, 0) == false) {
                        System.out.print("Solution does not exist");
                        return false;
                }

                printSolution(board);
                return true;
        }

        public static void main(String args[])
        {
                NQueenProblem Queen = new NQueenProblem();
                Queen.solveNQ();
        }
}
```

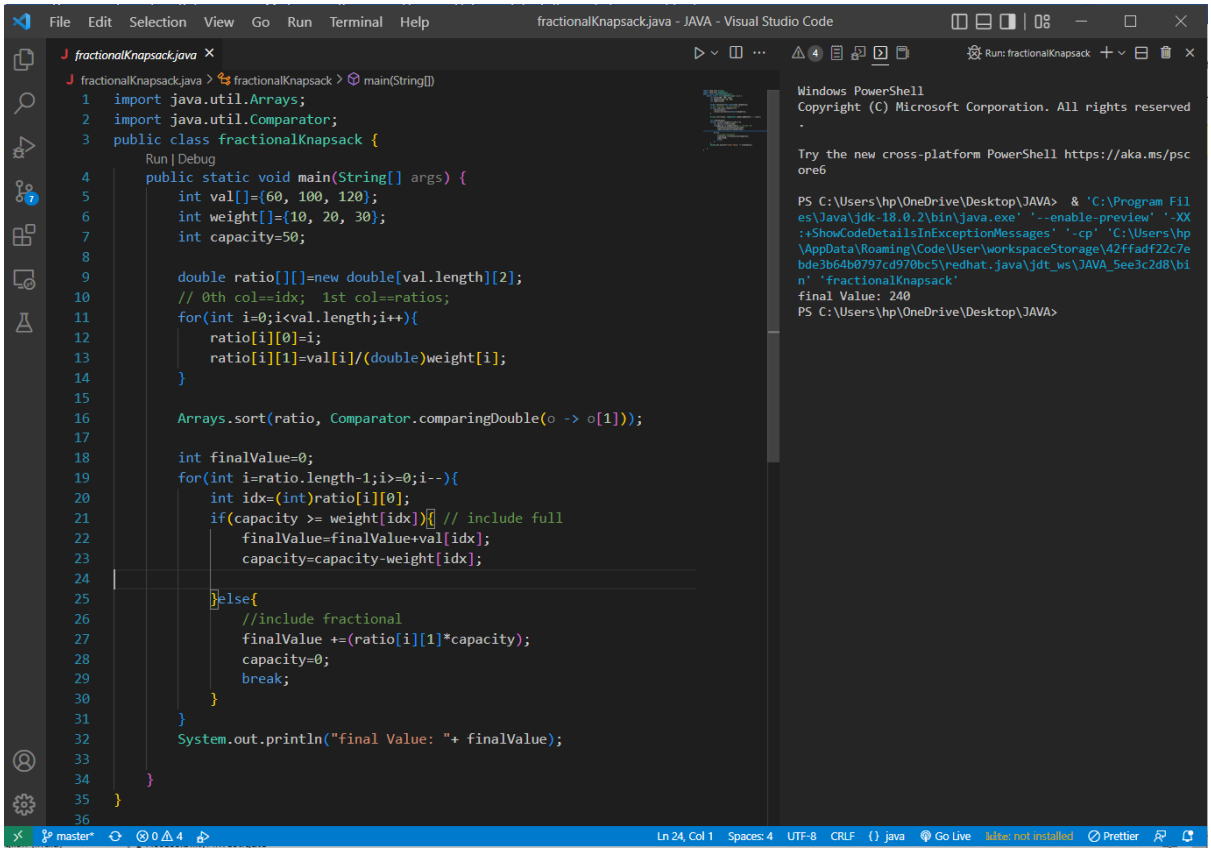
output:

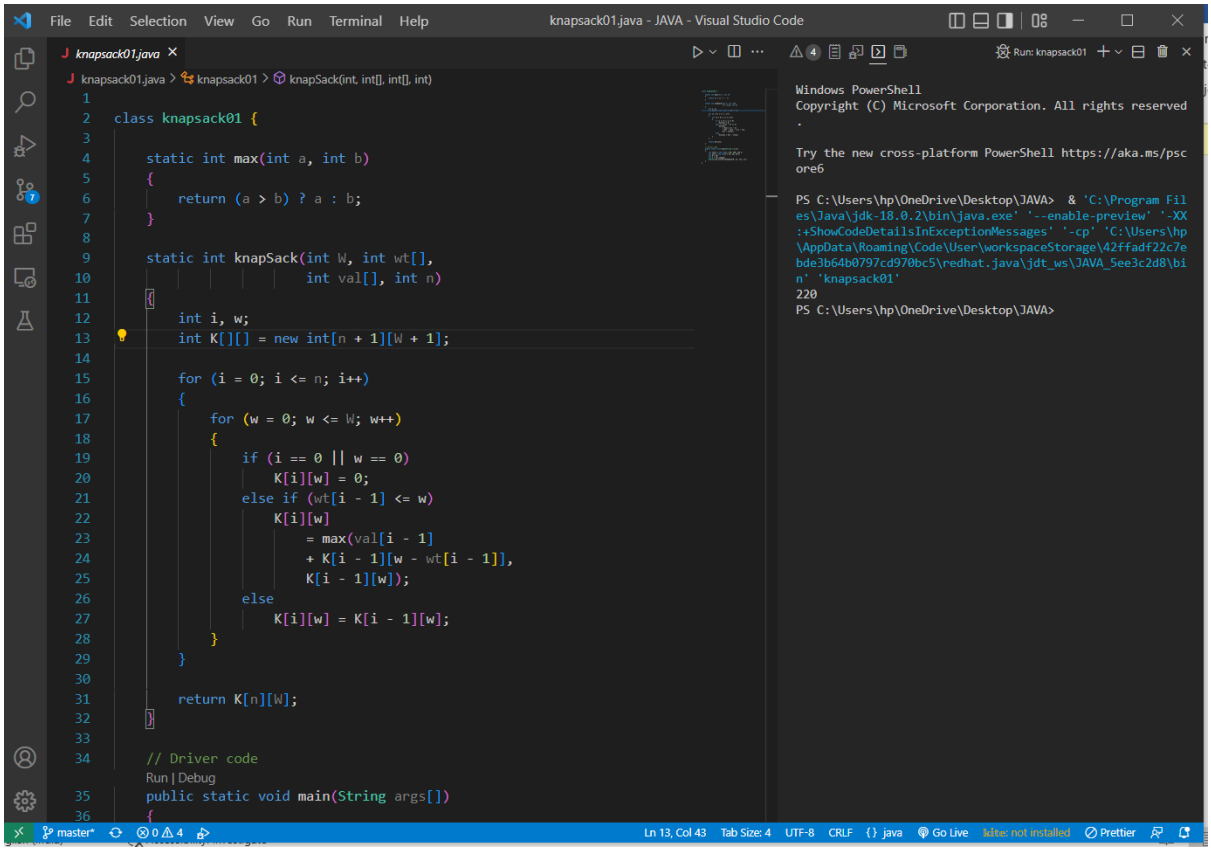
```
PS C:\Users\hp\OneDrive\Desktop\JAVA> javac NQueenProblem.java
PS C:\Users\hp\OneDrive\Desktop\JAVA> java NQueenProblem.java
 0  0  1  0
 1  0  0  0
 0  0  0  1
 0  1  0  0
PS C:\Users\hp\OneDrive\Desktop\JAVA>
```

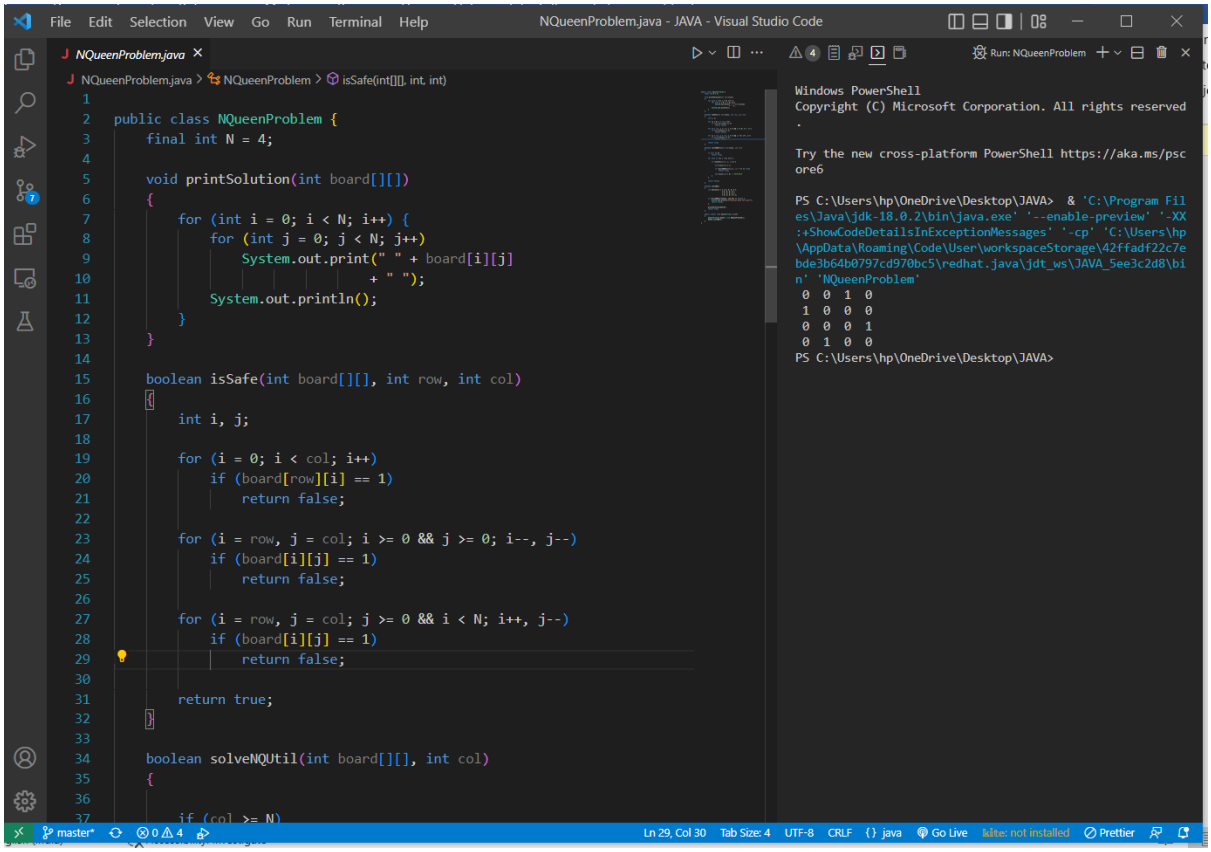

File Edit Selection View Go Run Terminal Help
Main.java - JAVA - Visual Studio Code
Main.java 1, M
Main.java > ...
import java.util.Scanner;
public class Main{
public static void fib(int n){
int a=0;
int b=1;
for(int i=0;i<n;i++){
System.out.print(a+" ");
int c=a+b;
a=b;
b=c;
}
}
Run | Debug
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
fib(n);
}
}
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\hp\OneDrive\Desktop\JAVA> java Main.java
5
0 1 1 2 3
PS C:\Users\hp\OneDrive\Desktop\JAVA> java Main.java
10
0 1 1 2 3 5 8 13 21 34
PS C:\Users\hp\OneDrive\Desktop\JAVA>
master*
Ln 30, Col 1
Spaces: 4
UTF-8
CRLF
{} java
Go Live
Prettier


File Edit Selection View Go Run Terminal Help
Main.java - JAVA - Visual Studio Code
Main.java 1, M
Main.java > ...
36
37
38
39 import java.util.Scanner;
40 public class Main{
41 public static int fib(int n){
42 if(n==0 || n==1){
43 return n;
44 }
45 return fib(n-1)+fib(n-2);
46 }
Run | Debug
47 public static void main(String[] args) {
48 Scanner sc=new Scanner(System.in);
49 int n=sc.nextInt();
50 for(int i=0;i<n;i++){
51 System.out.print(fib(i)+" ");
52 }
53
54 }
55 }
powershell
PS C:\Users\hp\OneDrive\Desktop\JAVA> java Main.java
15
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
PS C:\Users\hp\OneDrive\Desktop\JAVA>
master*
0 4
Ln 36, Col 1
Spaces: 4
UTF-8
CRLF
{} java
Go Live
Tabnine: not installed
Prettier


File Edit Selection View Go Run Terminal Help
Huffman.java - JAVA - Visual Studio Code
Huffman.java 2
Huffman.java > ...
import java.util.PriorityQueue;
import java.util.Scanner;
import java.util.Comparator;
class Huffman {
public static void printCode(HuffmanNode root, String s)
{
if (root.left
== null
&& root.right
== null
&& Character.isLetter(root.c)) {
System.out.println(root.c + ":" + s);
return;
}
printCode(root.left, s + "0");
printCode(root.right, s + "1");
}
Run | Debug
public static void main(String[] args)
{
Scanner s = new Scanner(System.in);
int n = 6;
char[] charArray = { 'a', 'b', 'c', 'd', 'e', 'f' };
int[] charfreq = { 5, 9, 12, 13, 16, 45 };
PriorityQueue<HuffmanNode> q
= new PriorityQueue<HuffmanNode>(n, new MyComparator());
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved
.
Try the new cross-platform PowerShell https://aka.ms/psc
ore6
PS C:\Users\hp\OneDrive\Desktop\JAVA> java Huffman.java
f:0
c:100
d:101
a:1100
b:1101
e:111
PS C:\Users\hp\OneDrive\Desktop\JAVA>
master*
Ln 2, Col 1 (33 selected)
Tab Size: 4
UTF-8
CRLF
java
Go Live
Prettier


File Edit Selection View Go Run Terminal Help
fractionalKnapsack.java - JAVA - Visual Studio Code
fractionalKnapsack.java
fractionalKnapsack.java > fractionalKnapsack > main(String[])
Run: fractionalKnapsack
import java.util.Arrays;
import java.util.Comparator;
public class fractionalKnapsack {
Run | Debug
public static void main(String[] args) {
int val[]={60, 100, 120};
int weight[]={10, 20, 30};
int capacity=50;
double ratio[][]=new double[val.length][2];
// 0th col==idx; 1st col==ratios;
for(int i=0;i<val.length;i++){
ratio[i][0]=i;
ratio[i][1]=val[i]/(double)weight[i];
}
Arrays.sort(ratio, Comparator.comparingDouble(o -> o[1]));
int finalValue=0;
for(int i=ratio.length-1;i>=0;i--){
int idx=(int)ratio[i][0];
if(capacity >= weight[idx]){ // include full
finalValue=finalValue+val[idx];
capacity=capacity-weight[idx];
}else{
//include fractional
finalValue +=(ratio[i][1]*capacity);
capacity=0;
break;
}
}
System.out.println("final Value: "+ finalValue);
}
}
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\hp\OneDrive\Desktop\JAVA> & 'C:\Program Files\Java\jdk-18.0.2\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\hp\AppData\Roaming\Code\User\workspaceStorage\42ffadf22c7ebde3b64b0797cd970bc5\redhat.java\jdt_ws\JAVA_5ee3c2d8\bin' 'fractionalKnapsack'
final Value: 240
PS C:\Users\hp\OneDrive\Desktop\JAVA>
master*
Ln 24, Col 1
Spaces: 4
UTF-8
CRLF
java
Go Live
Prettier


File Edit Selection View Go Run Terminal Help
knapsack01.java - JAVA - Visual Studio Code
knapsack01.java
knapsack01.java > knapsack01 > knapSack(int, int[], int[], int)
class knapsack01 {
static int max(int a, int b)
{
return (a > b) ? a : b;
}
static int knapSack(int W, int wt[],
int val[], int n)
{
int i, w;
int K[][] = new int[n + 1][W + 1];
for (i = 0; i <= n; i++)
{
for (w = 0; w <= W; w++)
{
if (i == 0 || w == 0)
K[i][w] = 0;
else if (wt[i - 1] <= w)
K[i][w]
= max(val[i - 1]
+ K[i - 1][w - wt[i - 1]],
K[i - 1][w]);
else
K[i][w] = K[i - 1][w];
}
}
return K[n][W];
}
// Driver code
Run | Debug
public static void main(String args[])
{
Run: knapsack01
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\hp\OneDrive\Desktop\JAVA> & 'C:\Program Files\Java\jdk-18.0.2\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\hp\AppData\Roaming\Code\User\workspaceStorage\42ffadf22c7ebde3b64b0797cd970bc5\redhat.java\jdt_ws\JAVA_5ee3c2d8\bin' 'knapsack01'
220
PS C:\Users\hp\OneDrive\Desktop\JAVA>
master*
Ln 13, Col 43
Tab Size: 4
UTF-8
CRLF
java
Go Live
Prettier


File Edit Selection View Go Run Terminal Help
NQueenProblem.java - JAVA - Visual Studio Code
NQueenProblem.java
NQueenProblem.java > NQueenProblem > isSafe(int[][], int, int)
public class NQueenProblem {
final int N = 4;
void printSolution(int board[][])
{
for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++)
System.out.print(" " + board[i][j]
+ " ");
System.out.println();
}
}
boolean isSafe(int board[][], int row, int col)
{
int i, j;
for (i = 0; i < col; i++)
if (board[row][i] == 1)
return false;
for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
if (board[i][j] == 1)
return false;
for (i = row, j = col; j >= 0 && i < N; i++, j--)
if (board[i][j] == 1)
return false;
return true;
}
boolean solveNQUtil(int board[][], int col)
{
if (col >= N)
Run: NQueenProblem
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved
.
Try the new cross-platform PowerShell https://aka.ms/psc
ore6
PS C:\Users\hp\OneDrive\Desktop\JAVA> & 'C:\Program Fil
es\Java\jdk-18.0.2\bin\java.exe' '--enable-preview' '-XX
:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\hp
\AppData\Roaming\Code\User\workspaceStorage\42ffadf22c7e
bde3b64b0797cd970bc5\redhat.java\jdt_ws\JAVA_5ee3c2d8\bi
n' 'NQueenProblem'
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
PS C:\Users\hp\OneDrive\Desktop\JAVA>
master*
Ln 29, Col 30
Tab Size: 4
UTF-8
CRLF
java
Go Live
Prettier

Assignment No : 6

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

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

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

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

Assignment:

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

Prerequisite:

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

Contents of the Theory:

1. Data Preprocessing
2. Linear regression
3. Random forest regression models

4. Box Plot
5. Outliers
6. Haversine
7. Mathplotlib
8. Mean Squared Error

Data Preprocessing:

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

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

Why do we need Data Preprocessing?

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

It involves below steps:

● Getting the dataset

● Importing libraries

● Importing datasets

● Finding Missing Data

● Encoding Categorical Data

● Splitting dataset into training and test set

● Feature scaling

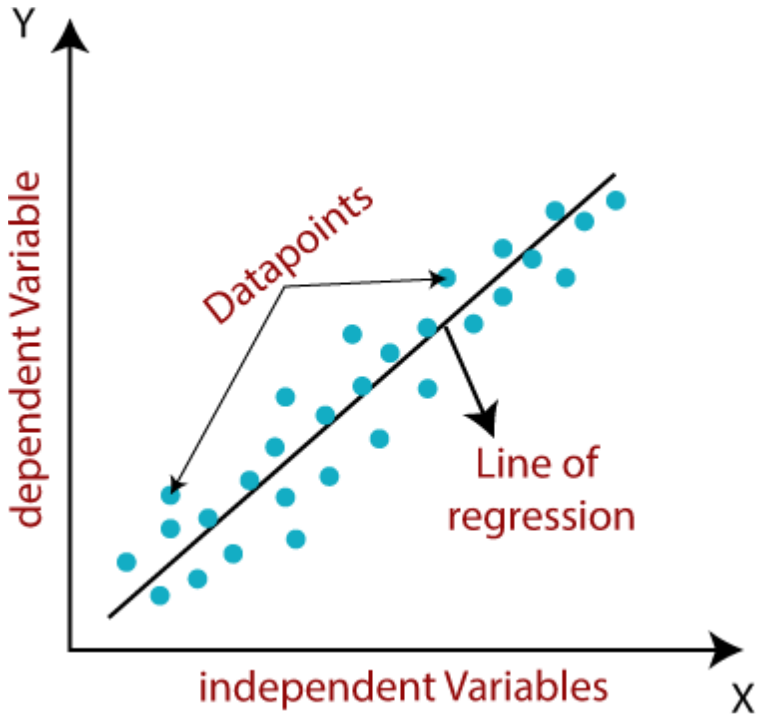
Linear Regression:

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

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the

linear relationship, which means itnds how the value of the dependent variable is changing according to the value of the independent variable.

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



Random Forest Regression Models:

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

As the name suggests,***"Random Forest is a classier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."***Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

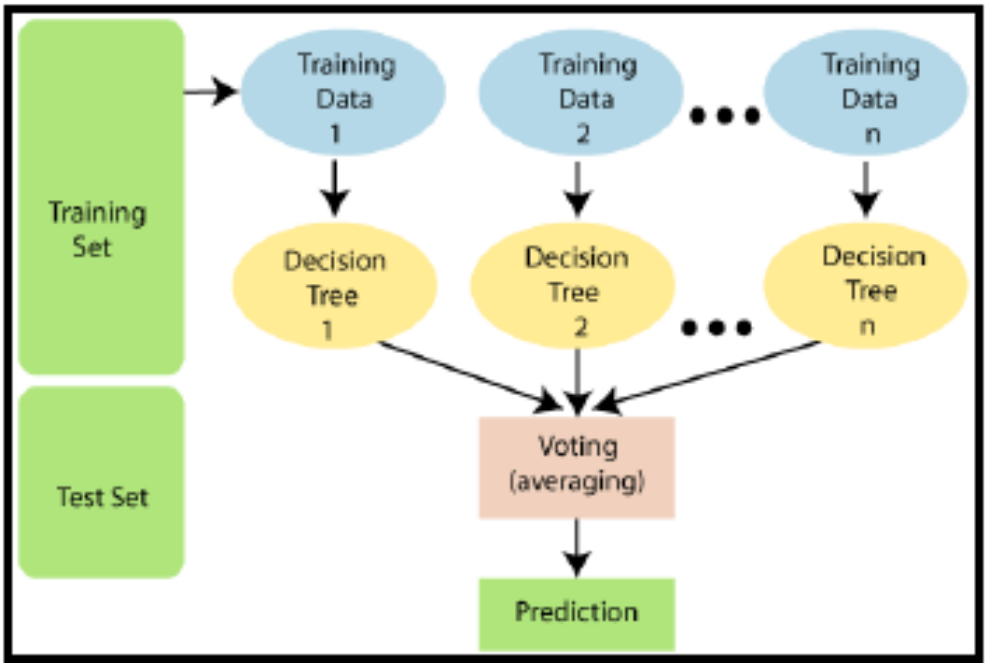
The greater number of trees in the forest leads to higher accuracy and prevents the problem of overftting.

Boxplot:

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

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

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



Conclusion:

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

Assignment No : 7

Title of the Assignment: Classify the email using the binary classification method. Email Spam detection has two states:

a) Normal State – Not Spam,

b) Abnormal State – Spam.

Use K-Nearest Neighbors and Support Vector Machine for classification. Analyze their performance.

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

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

Objective of the Assignment:

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

Prerequisite:

1. Basic knowledge of Python

2. Concept of K-Nearest Neighbors and Support Vector Machine for classification.

Contents of the Theory:

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

Data Preprocessing:

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

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

Why do we need Data Preprocessing?

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

It involves below steps:

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

Conclusion : In this way we have explored Concept of classification using KNN and SVM

Assignment No : 8

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

Dataset Description: The case study is from an open-source dataset from Kaggle. The dataset contains 10,000 sample points with 14 distinct features such as CustomerId, CreditScore, Geography, Gender, Age, Tenure, Balance, etc.

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

Perform the following steps:

1. Read the dataset.
2. Distinguish the feature and target set and divide the data set into training and test sets.
3. Normalize the train and test data.
4. Initialize and build the model. Identify the points of improvement and implement the same.
5. Print the accuracy score and confusion matrix (5 points).

Objective of the Assignment:

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

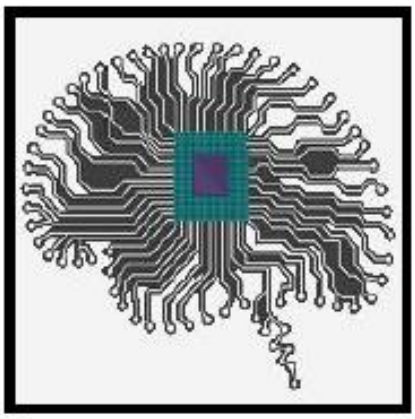
Prerequisite:

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

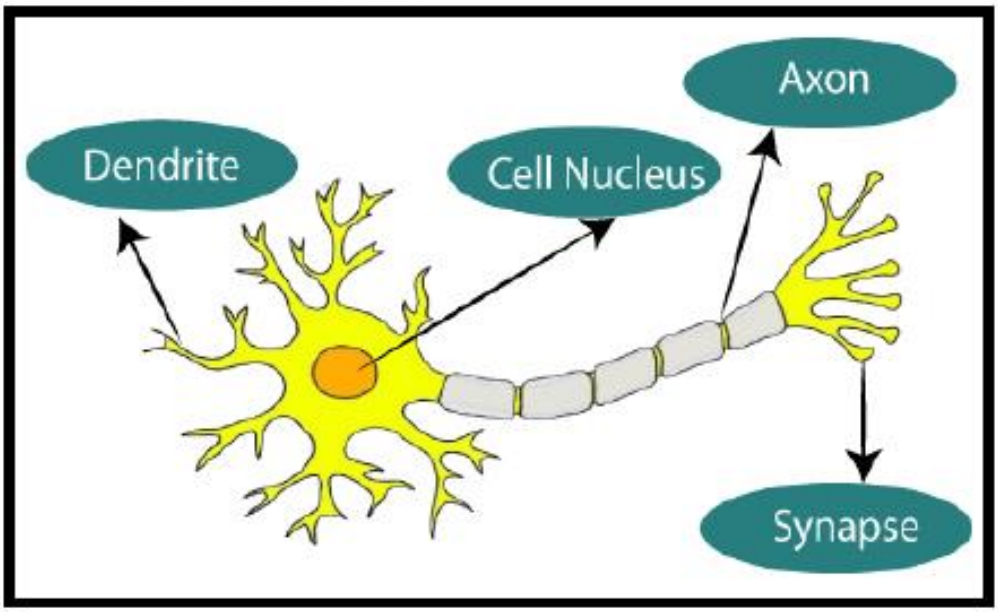
Contents of the Theory:

1. Artificial Neural Network
2. Keras
3. tensorflow
4. Normalization
5. Confusion Matrix

Artificial Neural Network:

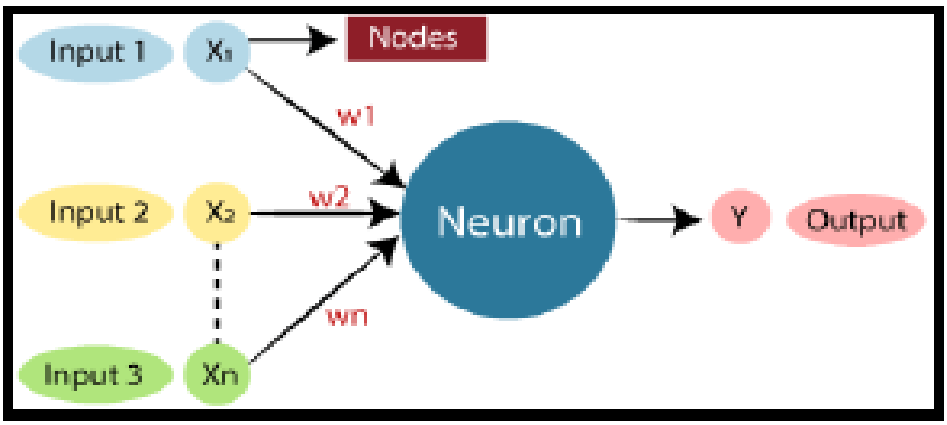


The term "Arti cial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, arti cial neural networks also have neurons that are interconnected to one another in various layers of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



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

The typical Arti cial Neural Network looks something like the given gure.



Dendrites from Biological Neural Network represent inputs in Arti cial Neural Networks, cellnucleus represents Nodes, synapse represents Weights, and Axon represents Output.

Relationship between Biological neural network and arti cial neural network:

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

AnArti cial Neural Networkin the eld ofArti cial intelligencewhere it attempts to mimic the network of neurons makes up a human brain so that computers will have an option tounderstand things and make decisions in a human-like manner. The arti cial neural networkis designed by programming computers to behave simply like interconnected brain cells.

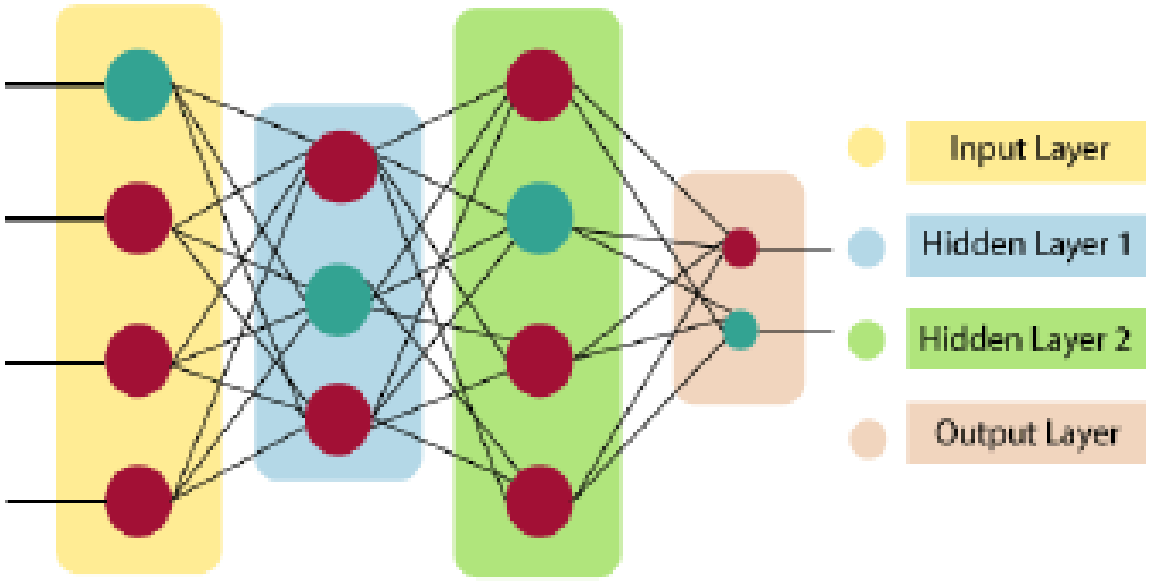
There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly.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

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

The architecture of an artificial neural network:

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

Artificial Neural Network primarily consists of three layers:



Input Layer:

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

Hidden Layer:

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

Output Layer:

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

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

$$\sum_{i=1}^{n} W_i * X_i + b$$

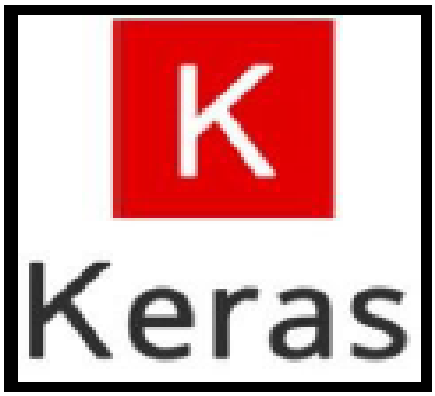
It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that canbe applied upon the sort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

Keras:

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

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

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



Tensor□ow:

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

Normalization:

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

Mathematically, we can calculate normalization with the below formula:Xn = (X -

Xminimum) / (Xmaximum - Xminimum)

Where,

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

Conclusion:

In this way we build a a neural network-based classier that can determine whether they will leave

Assignment No: 9

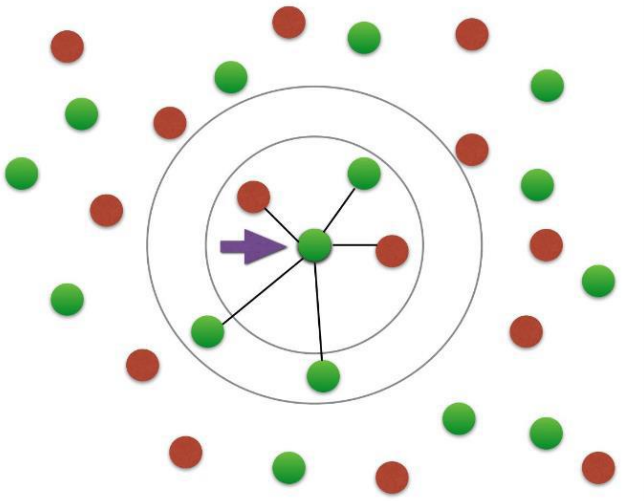
Title of the Assignment: Implement K-Nearest Neighbors algorithm on diabetes.csv dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset.

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

Link for Dataset: Diabetes predication system with KNN algorithm | Kaggle

Objective of the Assignment:
Students should be able to preprocess dataset and identify outliers, to check correlation and implement KNN algorithm and random forest classification models. Evaluate them with respective scores like confusion_matrix, accuracy_score, mean_squared_error, r2_score, roc_auc_score, roc_curve etc.

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



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

k-NN models work by taking a data point and looking at the 'k' closest labeled data points. The data point is then assigned the label of the majority of the 'k' closest points.

For example, if k = 5, and 3 of points are 'green' and 2 are 'red', then the data point in question would be labeled 'green', since 'green' is the majority (as shown in the above graph).

Scikit-learn is a machine learning library for Python. In this tutorial, we will build a k-NN model using Scikit-learn to predict whether or not a patient has diabetes.

Reading in the training data

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

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

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

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

#check number of rows and columns in dataset df.shape Op (768,9) We can see that we have 768 rows of data (potential diabetes patients) and 9 columns (8 input features and 1 target output).

Split up the dataset into inputs and targets Now let's split up our dataset into inputs (X) and our target (y). Our input will be every column except 'diabetes' because 'diabetes' is what we will be attempting to predict. Therefore, 'diabetes' will be our target. We will use pandas 'drop' function to drop the column 'diabetes' from our dataframe and store it in the variable 'X'. This will be our input. #create a dataframe with all training data except the target column X = df.drop(columns=['diabetes'])#check that the target variable has been removed X.head()

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

We will insert the ‘diabetes’ column of our dataset into our target variable (y). #separate target values y = df[‘diabetes’].values#view target values y[0:5]

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

Split the dataset into train and test data Now we will split the dataset into into training data and testing data. The training data is the data that the model will learn from. The testing data is the data we will use to see how well the model performs on unseen data. Scikit-learn has a function we can use called ‘train_test_split’ that makes it easy for us to split our dataset into training and testing data. from sklearn.model_selection import train_test_split#split dataset into train and test data X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y) ‘train_test_split’ takes in 5 parameters. The first two parameters are the input and target data we split up earlier. Next, we will set ‘test_size’ to 0.2. This means that 20% of all the data will be used for testing, which leaves 80% of the data as training data for the model to learn from. Setting ‘random_state’ to 1 ensures that we get the same split each time so we can reproduce our results. Setting ‘stratify’ to y makes our training split represent the proportion of each value in the y variable. For example, in our dataset, if 25% of patients have diabetes and 75% don’t have diabetes, setting ‘stratify’ to y will ensure that the random split has 25% of patients with diabetes and 75% of patients without diabetes.

Building and training the model

Next, we have to build the model.

Here is the code:

from sklearn.neighbors import KNeighborsClassifier # Create KNN classifier

knn = KNeighborsClassifier(n_neighbors = 3)# Fit the classifier to the data knn.fit(X_train,y_train)

First, we will create a new k-NN classifier and set ‘n_neighbors’ to 3. To recap, this means that if at least 2 out of the 3 nearest points to an new data point are patients without diabetes, then the new data point will be labeled as ‘no diabetes’, and vice versa. In other words, a new data point is labeled with by majority from the 3 nearest points. We have set ‘n_neighbors’ to 3 as a starting point. We will go into more detail below on how to better select a value for ‘n_neighbors’ so that the model can improve its performance.

Next, we need to train the model. In order to train our new model, we will use the ‘fit’ function and pass in our training data as parameters to fit our model to the training data. Testing the model Once the model is trained, we can use the ‘predict’ function on our model to make predictions on our test data. As seen when inspecting ‘y’ earlier, 0 indicates that the patient does not have diabetes and 1 indicates that the patient does have diabetes. To save space, we will only show print the first 5 predictions of our test set.

#show first 5 model predictions on the test data

knn.predict(X_test)[0:5]

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

We can see that the model predicted ‘no diabetes’ for the first 4 patients in the test set and ‘has diabetes’ for the 5th patient. Now let’s see how our accurate our model is on the full test set. To do this, we will use the ‘score’ function and pass in our test input and target data to see how well our model predictions match up to

the actual results.

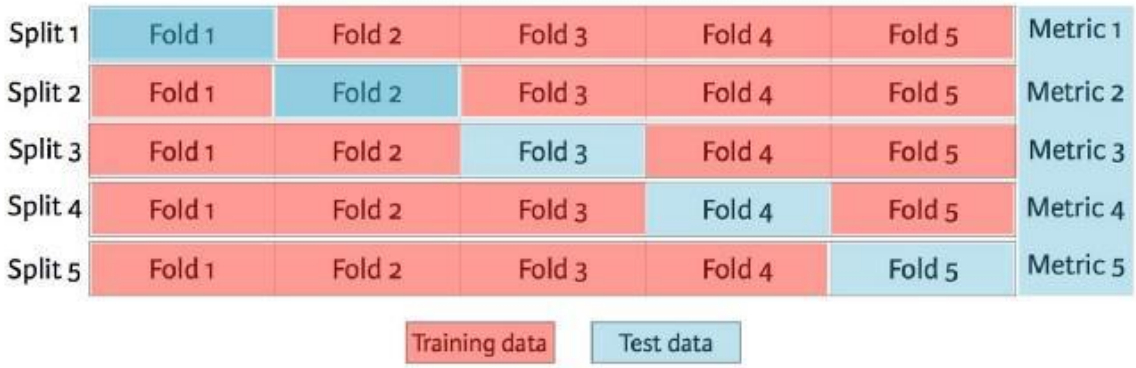
#check accuracy of our model on the test data

knn.score(X_test, y_test)

```
0.6688311688311688
```

Our model has an accuracy of approximately 66.88%. It’s a good start, but we will see how we can increase model performance below. Congrats! You have now built an amazing k-NN model! k-Fold Cross-Validation Cross-validation is when the dataset is randomly split up into ‘k’ groups.

One of the groups is used as the test set and the rest are used as the training set. The model is trained on the training set and scored on the test set. Then the process is repeated until each unique group as been used as the test set. For example, for 5-fold cross validation, the dataset would be split into 5 groups, and the model would be trained and tested 5 separate times so each group would get a chance to be the test set. This can be seen in the graph below.



The train-test-split method we used in earlier is called 'holdout'. Cross-validation is better than using the holdout method because the holdout method score is dependent on how the data is split into train and test sets. Cross-validation gives the model an opportunity to test on multiple splits so we can get a better idea on how the model will perform on unseen data. In order to train and test our model using cross-validation, we will use the 'cross_val_score' function with a cross-validation value of 5. 'cross_val_score' takes in our k-NN model and our data as parameters. Then it splits our data into 5 groups and fits and scores our data 5 seperate times, recording the accuracy score in an array each time. We will save the accuracy scores in the 'cv_scores' variable. To find the average of the 5 scores, we will use numpy's mean function, passing in 'cv_score'. Numpy is a useful math library in Python.

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

```
[ 0.68181818  0.69480519  0.75324675  0.75163399  0.68627451]
cv_scores mean:0.7135557253204311
```

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

Hypertuning model parameters using GridSearchCV When built our initial k-NN model, we set the parameter ‘n_neighbors’ to 3 as a starting point with no real logic behind that choice. Hypertuning parameters is when you go through a process to find the optimal parameters for your model to improve accuracy. In our case, we will use GridSearchCV to find the optimal value for ‘n_neighbors’. GridSearchCV works by training our model multiple times on a range of parameters that we specify. That way, we can test our model with each parameter and figure out the optimal values to get the best accuracy results. For our model, we will specify a range of values for ‘n_neighbors’ in order to see which value works best for our model. To do this, we will create a dictionary, setting ‘n_neighbors’ as the key and using numpy to create an array of values from 1 to 24. Our new model using grid search will take in a new k-NN classifier, our param_grid and a cross-

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

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

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

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

```
0.7578125
```

By using grid search to find the optimal parameter for our model, we have improved our model accuracy by over 4%!

Conclusion:

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

Assignment No : 10

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

Dataset Description: The data includes the following features:

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

Objective of the Assignment:

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

Prerequisite:

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

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

Uses of Clustering Marketing: In the field of marketing, clustering can be used to identify various customer groups with existing customer data. Based on that, customers can be provided with discounts, offers, promo codes etc. Real Estate: Clustering can be used to understand and divide various property locations based on value and importance. Clustering algorithms can process through the data and identify various groups of property on the basis of probable price. BookStore and Library management: Libraries and Bookstores can use Clustering to better manage the book

database. With proper book ordering, better operations can be implemented. Document Analysis: Often, we need to group together various research texts and documents according to similarity. And in such cases, we don't have any labels. Manually labelling large amounts of data is also not possible. Using clustering, the algorithm can process the text and group it into different themes.

These are some of the interesting use cases of clustering. K-Means Clustering K-Means clustering is an unsupervised machine learning algorithm that divides the given data into the given number of clusters. Here, the "K" is the given number of predefined clusters, that need to be created. It is a centroid based algorithm in which each cluster is associated with a centroid. The main idea is to reduce the distance between the data points and their respective cluster centroid. The algorithm takes raw unlabelled data as an input and divides the dataset into clusters and the process is repeated until the best clusters are found. K-Means is very easy and simple to implement. It is highly scalable, can be applied to both small and large datasets. There is, however, a problem with choosing the number of clusters or K. Also, with the increase in dimensions, stability decreases. But, overall K Means is a simple and robust algorithm that makes clustering very easy.

Conclusion : Thus we have studied clustering algorithm.

MINI PROJECT

Title of Project:

Objectives:

Input:

Output:

Implementation Code & Result:

Conclusion:

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

1. Pre-process the dataset.
2. Identify outliers.
3. Check the correlation.
4. Implement linear regression and random forest regression models.
5. Evaluate the models and compare their respective scores like R2, RMSE, etc. Dataset link: https://www.kaggle.com/datasets/yasserh/uber-fares-dataset

In [1]:

```
#Importing the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
#importing the dataset
df  = pd.read_csv("uber.csv")
```

1. Pre-process the dataset.

In [3]:

```
df.head()
```

Out[3]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40

In [4]:

```
df.info() #To get the required information of the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Unnamed: 0         200000 non-null  int64
 1   key                200000 non-null  object
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
```

```
 6   dropoff_longitude  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
```

In [5]:

```
df.columns #TO get number of columns in the dataset
```

Out[5]:

```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

In [6]:

```
df = df.drop(['Unnamed: 0', 'key'], axis= 1) #To drop unnamed column as it isn't requirec
```

In [7]:

```
df.head()
```

Out[7]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	**7.5**	**2015-05-07 19:52:06 UTC**	**-73.999817**	**40.738354**	**-73.999512**	**40.723217**	**1**
1	**7.7**	**2009-07-17 20:04:56 UTC**	**-73.994355**	**40.728225**	**-73.994710**	**40.750325**	**1**
2	**12.9**	**2009-08-24 21:45:00 UTC**	**-74.005043**	**40.740770**	**-73.962565**	**40.772647**	**1**
3	**5.3**	**2009-06-26 08:22:21 UTC**	**-73.976124**	**40.790844**	**-73.965316**	**40.803349**	**3**
4	**16.0**	**2014-08-28 17:47:00 UTC**	**-73.925023**	**40.744085**	**-73.973082**	**40.761247**	**5**

In [8]:

```
df.shape #To get the total (Rows,Columns)
```

Out[8]:

```
(200000, 7)
```

In [9]:

```
df.dtypes #To get the type of each column
```

Out[9]:

```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
dtype: object
```

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 7 columns):
```

```
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   fare_amount        200000 non-null  float64
 1   pickup_datetime    200000 non-null  object 
 2   pickup_longitude   200000 non-null  float64
 3   pickup_latitude    200000 non-null  float64
 4   dropoff_longitude  199999 non-null  float64
 5   dropoff_latitude   199999 non-null  float64
 6   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 10.7+ MB
```

In [11]:

```
df.describe() #To get statistics of each columns
```

Out[11]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	200000.000000	200000.000000	200000.000000	199999.000000	199999.000000	200000.000000
mean	11.359955	-72.527638	39.935885	-72.525292	39.923890	1.684535
std	9.901776	11.437787	7.720539	13.117408	6.794829	1.385997
min	-52.000000	-1340.648410	-74.015515	-3356.666300	-881.985513	0.000000
25%	6.000000	-73.992065	40.734796	-73.991407	40.733823	1.000000
50%	8.500000	-73.981823	40.752592	-73.980093	40.753042	1.000000
75%	12.500000	-73.967154	40.767158	-73.963658	40.768001	2.000000
max	499.000000	57.418457	1644.421482	1153.572603	872.697628	208.000000

Filling Missing values

In [12]:

```
df.isnull().sum()
```

Out[12]:

```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    1
dropoff_latitude     1
passenger_count      0
dtype: int64
```

In [13]:

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

In [14]:

```
df.isnull().sum()
```

Out[14]:

```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
```

In [15]:

```
df.dtypes
```

Out[15]:

```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
dtype: object
```

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

In [16]:

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

In [17]:

```
df.dtypes
```

Out[17]:

```
fare_amount                        float64
pickup_datetime      datetime64[ns, UTC]
pickup_longitude                   float64
pickup_latitude                    float64
dropoff_longitude                  float64
dropoff_latitude                   float64
passenger_count                      int64
dtype: object
```

To segregate each time of date and time

In [18]:

```
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)
```

In [19]:

```
df.head()
```

Out[19]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hour
0	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-73.999512	40.723217	1	19
1	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-73.994710	40.750325	1	20
2	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-73.962565	40.772647	1	21
3	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-73.965316	40.803349	3	8
4	16.0	2014-08-28 17:47:00+00:00	-73.925023	40.744085	-73.973082	40.761247	5	17

In [20]:

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

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

In [21]:

```
df.head()
```

Out[21]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hour	day	month	ye
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1	19	7	5	20
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1	20	17	7	20
2	12.9	-74.005043	40.740770	-73.962565	40.772647	1	21	24	8	20
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3	8	26	6	20
4	16.0	-73.925023	40.744085	-73.973082	40.761247	5	17	28	8	20

In [22]:

```
df.dtypes
```

Out[22]:

```
fare_amount          float64
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
hour                   int64
day                    int64
month                  int64
year                   int64
dayofweek              int64
dtype: object
```

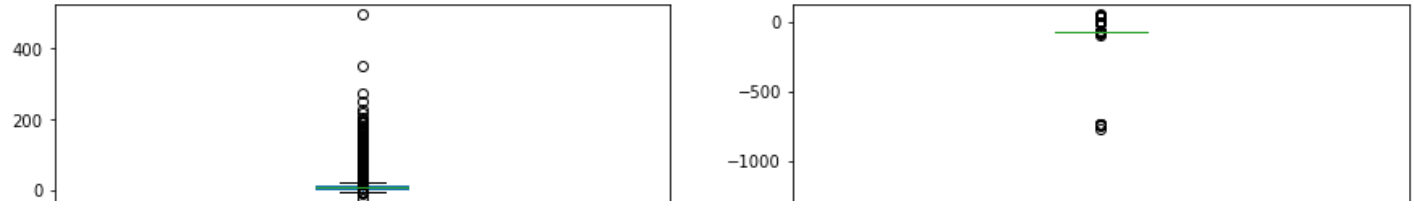
Checking outliers and filling them

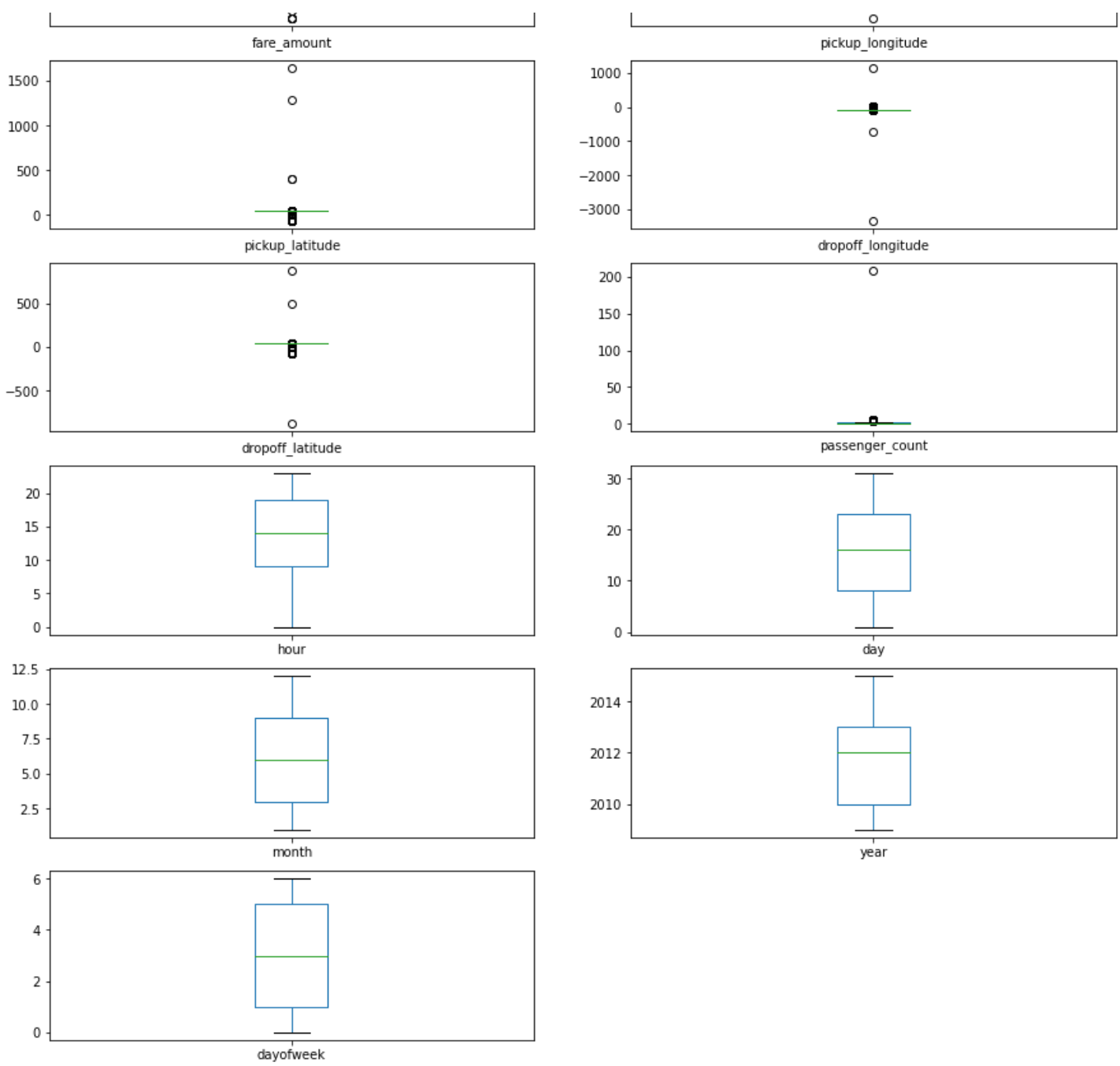
In [23]:

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

Out[23]:

```
fare_amount            AxesSubplot(0.125,0.787927;0.352273x0.0920732)
pickup_longitude     AxesSubplot(0.547727,0.787927;0.352273x0.0920732)
pickup_latitude        AxesSubplot(0.125,0.677439;0.352273x0.0920732)
dropoff_longitude    AxesSubplot(0.547727,0.677439;0.352273x0.0920732)
dropoff_latitude       AxesSubplot(0.125,0.566951;0.352273x0.0920732)
passenger_count      AxesSubplot(0.547727,0.566951;0.352273x0.0920732)
hour                   AxesSubplot(0.125,0.456463;0.352273x0.0920732)
day                  AxesSubplot(0.547727,0.456463;0.352273x0.0920732)
month                  AxesSubplot(0.125,0.345976;0.352273x0.0920732)
year                 AxesSubplot(0.547727,0.345976;0.352273x0.0920732)
dayofweek              AxesSubplot(0.125,0.235488;0.352273x0.0920732)
dtype: object
```





In [24]:

```
#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
```

In [25]:

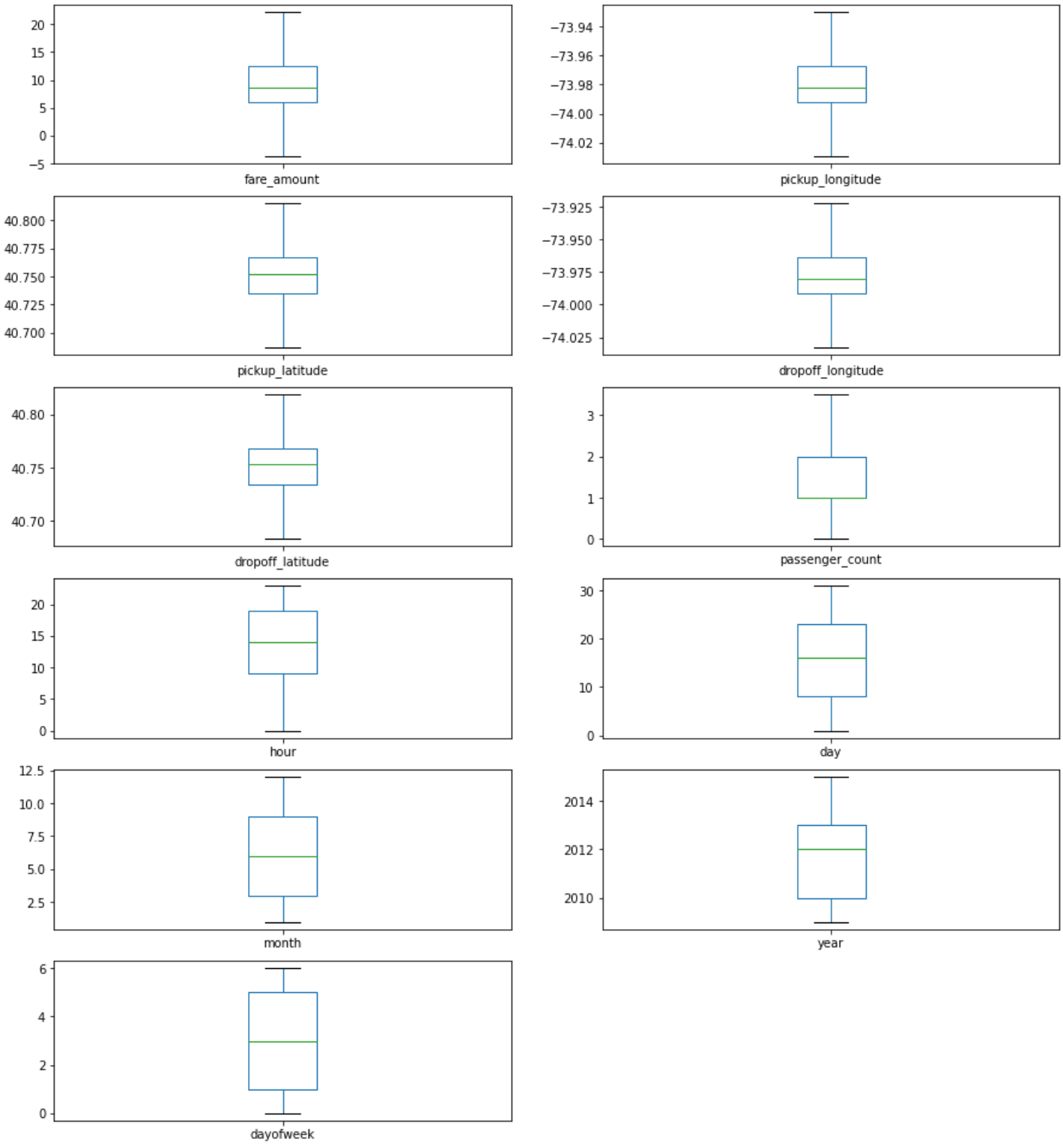
```
df = treat_outliers_all(df , df.iloc[: , 0::])
```

In [26]:

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

Out[26]:

```
fare_amount             AxesSubplot(0.125,0.787927;0.352273x0.0920732)
pickup_longitude     AxesSubplot(0.547727,0.787927;0.352273x0.0920732)
pickup_latitude         AxesSubplot(0.125,0.677439;0.352273x0.0920732)
dropoff_longitude    AxesSubplot(0.547727,0.677439;0.352273x0.0920732)
dropoff_latitude        AxesSubplot(0.125,0.566951;0.352273x0.0920732)
passenger_count      AxesSubplot(0.547727,0.566951;0.352273x0.0920732)
hour                    AxesSubplot(0.125,0.456463;0.352273x0.0920732)
day                  AxesSubplot(0.547727,0.456463;0.352273x0.0920732)
month                   AxesSubplot(0.125,0.345976;0.352273x0.0920732)
year                 AxesSubplot(0.547727,0.345976;0.352273x0.0920732)
dayofweek               AxesSubplot(0.125,0.235488;0.352273x0.0920732)
dtype: object
```



In [27]:

```
#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()
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

Out[27]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hour	day	month	ye
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1.0	19	7	5	20
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1.0	20	17	7	20
2	12.9	-74.005043	40.740770	-73.962565	40.772647	1.0	21	24	8	20
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3.0	8	26	6	20
4	16.0	-73.929786	40.744085	-73.973082	40.761247	3.5	17	28	8	20

In [28]:

```
#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)
```

```
Remaining observastions in the dataset: (200000, 12)
```

In [29]:

```
#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)
                                ]
```

In [30]:

```
df.drop(incorrect_coordinates, inplace = True, errors = 'ignore')
```

In [31]:

```
df.head()
```

Out[31]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hour	day	month	ye
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1.0	19	7	5	20
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1.0	20	17	7	20

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hour	day	month	ye
2	12.9	-74.016048	40.711303	-73.979268	40.782004	1.0	21	28	8	20
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3.0	8	26	6	20
4	16.0	-73.929786	40.744085	-73.973082	40.761247	3.5	17	28	8	20

In [32]:

```
df.isnull().sum()
```

Out[32]:

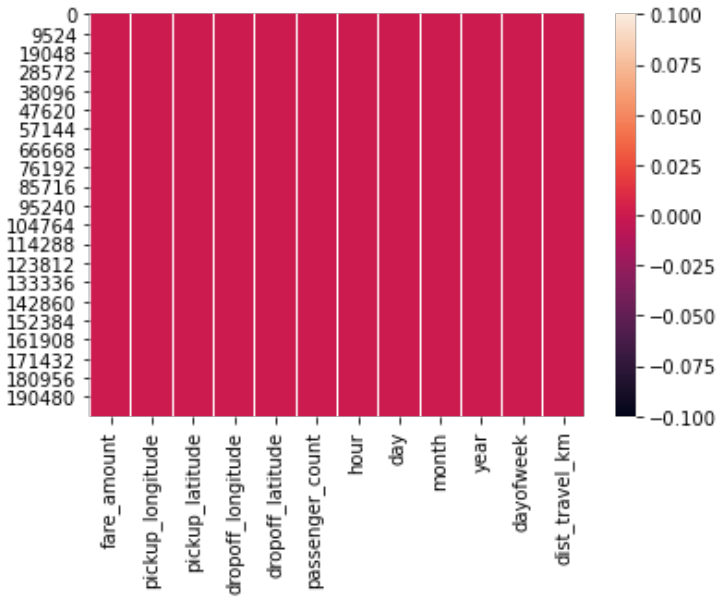
```
fare_amount          0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
hour                 0
day                  0
month                0
year                 0
dayofweek            0
dist_travel_km       0
dtype: int64
```

In [33]:

```
sns.heatmap(df.isnull()) #Free for null values
```

Out[33]:

```
<AxesSubplot:>
```



In [34]:

```
corr = df.corr() #Function to find the correlation
```

In [35]:

```
corr
```

Out[35]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	ho
fare_amount	1.000000	0.154069	-0.110842	0.218675	-0.125898	0.015778	0.0236
pickup_longitude	0.154069	1.000000	0.259497	0.425619	0.073290	-0.013213	0.0115

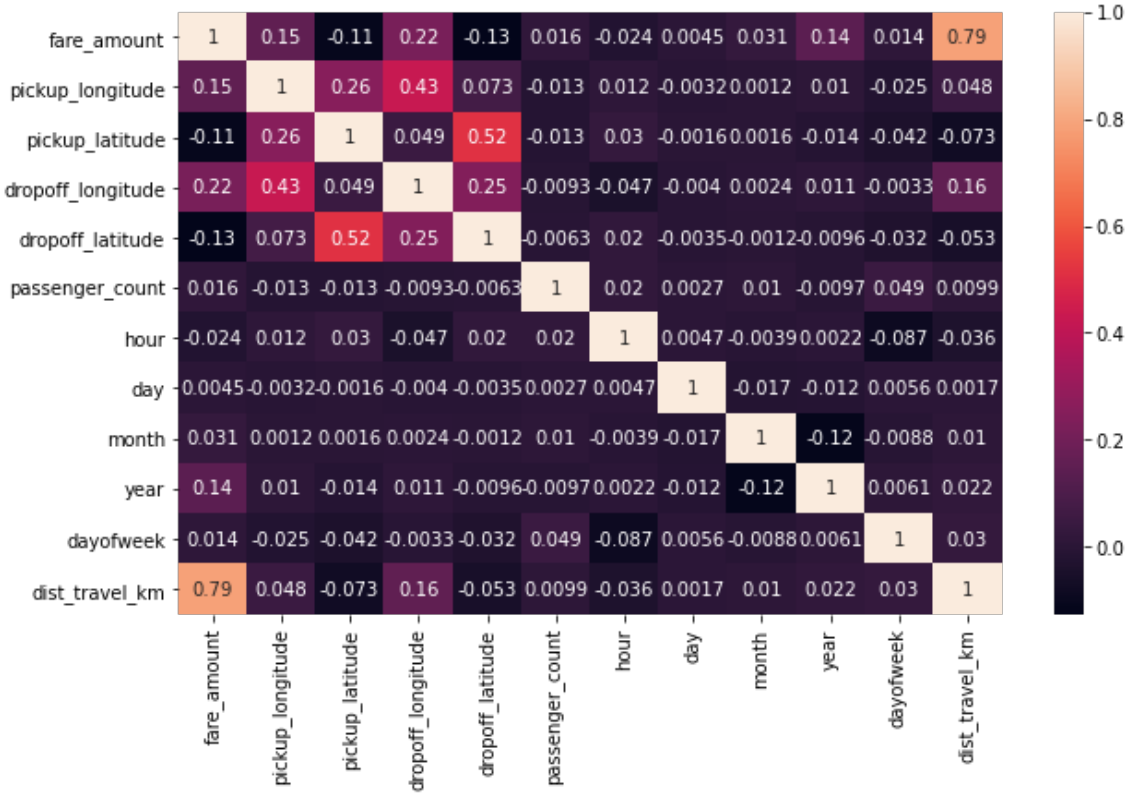
	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hc
pickup_latitude	-0.110842	-0.289497	1.000000	-0.048889	0.515714	-0.012889	0.0296
dropoff_longitude	0.218675	0.425619	0.048889	1.000000	0.245667	-0.009303	0.0465
dropoff_latitude	-0.125898	0.073290	0.515714	0.245667	1.000000	-0.006308	0.0197
passenger_count	0.015778	-0.013213	-0.012889	-0.009303	-0.006308	1.000000	0.0202
hour	-0.023623	0.011579	0.029681	-0.046558	0.019783	0.020274	1.0000
day	0.004534	-0.003204	-0.001553	-0.004007	-0.003479	0.002712	0.0046
month	0.030817	0.001169	0.001562	0.002391	-0.001193	0.010351	0.0039
year	0.141277	0.010198	-0.014243	0.011346	-0.009603	-0.009749	0.0021
dayofweek	0.013652	-0.024652	-0.042310	-0.003336	-0.031919	0.048550	0.0869
dist_travel_km	0.786385	0.048446	-0.073362	0.155191	-0.052701	0.009884	0.0357

In [36]:

```
fig,axis = plt.subplots(figsize = (10,6))
sns.heatmap(df.corr(),annot = True) #Correlation Heatmap (Light values means highly corr
elated)
```

Out[36]:

<AxesSubplot:>



Dividing the dataset into feature and target values

In [182]:

```
x = df[['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_latitude','pass
enger_count','hour','day','month','year','dayofweek','dist_travel_km']]
```

In [183]:

```
y = df['fare_amount']
```

Dividing the dataset into training and testing dataset

In [184]:

```
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)
```

Linear Regression

In [185]:

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

In [186]:

```
regression.fit(X_train,y_train)
```

Out[186]:

```
LinearRegression()
```

In [80]:

```
regression.intercept_ #To find the linear intercept
```

Out[80]:

```
2640.1356169149753
```

In [187]:

```
regression.coef_ #To find the linear coeeficient
```

Out[187]:

```
array([ 2.54805415e+01, -7.18365435e+00,  1.96232986e+01, -1.79401980e+01,
        5.48472723e-02,  5.32910041e-03,  4.05930990e-03,  5.74261856e-02,
        3.66574831e-01, -3.03753790e-02,  1.84233728e+00])
```

In [188]:

```
prediction = regression.predict(X_test) #To predict the target values
```

In [189]:

```
print(prediction)
```

```
[ 5.47848314 10.11016249 12.19490542 ...  7.11952609 20.2482979
  8.82791961]
```

In [190]:

```
y_test
```

Out[190]:

```
155740     4.90
47070     10.00
116192    14.50
164589     6.50
154309    11.30
          ...
76552      7.70
27926     10.90
38972      6.50
```

```
38972       6.50
120341     22.25
178449      8.10
Name: fare_amount, Length: 66000, dtype: float64
```

Metrics Evaluation using R2, Mean Squared Error, Root Mean Sqared Error

In [191]:

```
from sklearn.metrics import r2_score
```

In [192]:

```
r2_score(y_test,prediction)
```

Out[192]:

```
0.6651880468683617
```

In [193]:

```
from sklearn.metrics import mean_squared_error
```

In [194]:

```
MSE = mean_squared_error(y_test,prediction)
```

In [195]:

```
MSE
```

Out[195]:

```
9.961516917717704
```

In [196]:

```
RMSE = np.sqrt(MSE)
```

In [197]:

```
RMSE
```

Out[197]:

```
3.156187085348032
```

Random Forest Regression

In [198]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [199]:

```
rf = RandomForestRegressor(n_estimators=100) #Here n_estimators means number of trees you want to build before making the prediction
```

In [200]:

```
rf.fit(X_train,y_train)
```

Out[200]:

```
RandomForestRegressor()
```

In [201]:

```
y_pred = rf.predict(X_test)
```

In [202]:

```
y_pred
```

Out[202]:

```
array([ 5.714 , 10.285 , 12.68  , ...,  6.338 , 19.4685,  7.712 ])
```

Metrics evaluatin for Random Forest

In [210]:

```
R2_Random = r2_score(y_test,y_pred)
```

In [211]:

```
R2_Random
```

Out[211]:

```
0.7948374920410631
```

In [205]:

```
MSE_Random = mean_squared_error(y_test,y_pred)
```

In [206]:

```
MSE_Random
```

Out[206]:

```
6.104112397417331
```

In [207]:

```
RMSE_Random = np.sqrt(MSE_Random)
```

In [208]:

```
RMSE_Random
```

Out[208]:

```
2.4706501972997574
```

Assignment 2

1. **Classify the email using the binary classification method. Email Spam detection has two states: a) Normal State – Not Spam, b) Abnormal State – Spam. Use K-Nearest Neighbors and Support Vector Machine for classification. Analyze their performance. Dataset link: The emails.csv dataset on the Kaggle https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv**

In [19]:

```
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
```

In [20]:

```
df=pd.read_csv('emails.csv')
```

In [21]:

```
df.head()
```

Out[21]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	P
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	0	0	0	0	0	
1	Email 2	8	13	24	6	6	2	102	1	27	...	0	0	0	0	0	0	0	1	0	
2	Email 3	0	0	1	0	0	0	8	0	0	...	0	0	0	0	0	0	0	0	0	
3	Email 4	0	5	22	0	5	1	51	2	10	...	0	0	0	0	0	0	0	0	0	
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0	0	1	0	

5 rows × 3002 columns

In [22]:

```
df.columns
```

Out[22]:

```
Index(['Email No.', 'the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou',
       ...
       'connevey', 'jay', 'valued', 'lay', 'infrastructure', 'military',
       'allowing', 'ff', 'dry', 'Prediction'],
      dtype='object', length=3002)
```

In [23]:

```
df.isnull().sum()
```

Out[23]:

```
Email No.     0
the           0
to            0
ect           0
and           0
             ..
military      0
allowing      0
ff            0
dry           0
Prediction    0
Length: 3002, dtype: int64
```

In [24]:

```
df.dropna(inplace = True)
```

In [25]:

```
df.drop(['Email No.'],axis=1,inplace=True)
X = df.drop(['Prediction'],axis = 1)
y = df['Prediction']
```

In [26]:

```
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)
```

KNN classifier

In [35]:

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

knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

In [36]:

```
print("Prediction",y_pred)
```

```
Prediction [0 0 1 ... 1 1 1]
```

In [37]:

```
print("KNN accuracy = ",metrics.accuracy_score(y_test,y_pred))
```

```
KNN accuracy =  0.8009020618556701
```

In [39]:

```
print("Confusion matrix",metrics.confusion_matrix(y_test,y_pred))
```

```
Confusion matrix [[804 293]
 [ 16 439]]
```

SVM classifier

In [27]:

```
# cost C = 1
model = SVC(C = 1)
```

```
# fit
model.fit(X_train, y_train)

# predict
y_pred = model.predict(X_test)
```

In [28]:

```
metrics.confusion_matrix(y_true=y_test, y_pred=y_pred)
```

Out[28]:

```
array([[1091,    6],
       [  90,  365]])
```

In [29]:

```
print("SVM accuracy = ",metrics.accuracy_score(y_test,y_pred))
```

```
SVM accuracy =  0.9381443298969072
```

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

Dataset Description: The case study is from an open-source dataset from Kaggle. The dataset contains 10,000 sample points with 14 distinct features such as CustomerId, CreditScore, Geography, Gender, Age, Tenure, Balance, etc. Link to the Kaggle project: https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling Perform following steps:

1. **Read the dataset.**
2. **Distinguish the feature and target set and divide the data set into training and test sets.**
3. **Normalize the train and test data.**
4. **Initialize and build the model. Identify the points of improvement and implement the same.**
5. **Print the accuracy score and confusion matrix.**

In [46]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt #Importing the libraries
```

In [47]:

```
df = pd.read_csv("Churn_Modelling.csv")
```

Preprocessing.

In [48]:

```
df.head()
```

Out[48]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	

In [49]:

```
df.shape
```

Out[49]:

```
(10000, 14)
```

In [50]:

```
df.describe()
```

Out[50]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	Is

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	Is
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	

In [51]:

```
df.isnull()
```

Out[51]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrC
0	False	False	False	False	False	False	False	False	False	False	Fa
1	False	False	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	False	Fa
3	False	False	False	False	False	False	False	False	False	False	Fa
4	False	False	False	False	False	False	False	False	False	False	Fa
...	...	...	...	...	...	...	...	...	...	...	
9995	False	False	False	False	False	False	False	False	False	False	Fa
9996	False	False	False	False	False	False	False	False	False	False	Fa
9997	False	False	False	False	False	False	False	False	False	False	Fa
9998	False	False	False	False	False	False	False	False	False	False	Fa
9999	False	False	False	False	False	False	False	False	False	False	Fa

10000 rows × 14 columns

In [52]:

```
df.isnull().sum()
```

Out[52]:

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [53]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
```

```
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [54]:

```
df.dtypes
```

Out[54]:

```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

In [55]:

```
df.columns
```

Out[55]:

```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [56]:

```
df = df.drop(['RowNumber', 'Surname', 'CustomerId'], axis= 1) #Dropping the unnecessary columns
```

In [57]:

```
df.head()
```

Out[57]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	France	Female	42	2	0.00	1	1	1	101348.88
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	699	France	Female	39	1	0.00	2	0	0	93826.63
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10

Visualization

In [101]:

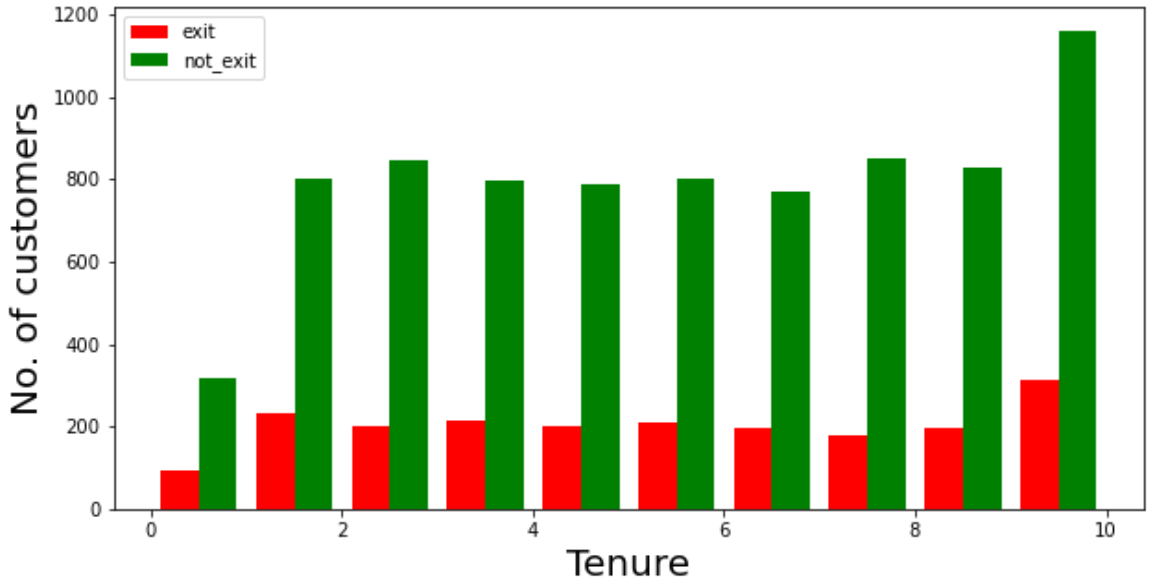
```
def visualization(x, y, xlabel):
    plt.figure(figsize=(10,5))
    plt.hist([x, y], color=['red', 'green'], label = ['exit', 'not_exit'])
    plt.xlabel(xlabel,fontsize=20)
    plt.ylabel("No. of customers", fontsize=20)
    plt.legend()
```

In [102]:

```
df_churn_exited = df[df['Exited']==1]['Tenure']
df_churn_not_exited = df[df['Exited']==0]['Tenure']
```

In [103]:

```
visualization(df_churn_exited, df_churn_not_exited, "Tenure")
```

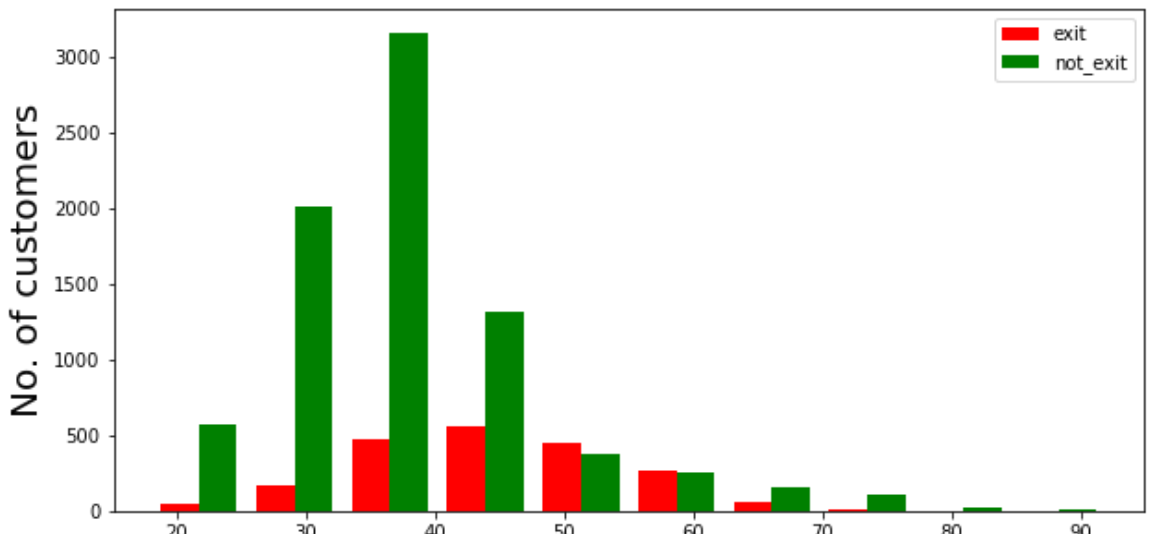


In [105]:

```
df_churn_exited2 = df[df['Exited']==1]['Age']
df_churn_not_exited2 = df[df['Exited']==0]['Age']
```

In [106]:

```
visualization(df_churn_exited2, df_churn_not_exited2, "Age")
```



Converting the Categorical Variables

In [59]:

```
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)
```

In [61]:

```
df = pd.concat([df,gender,states], axis = 1)
```

Splitting the training and testing Dataset

In [62]:

```
df.head()
```

Out[62]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	France	Female	42	2	0.00	1	1	1	101348.88
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	699	France	Female	39	1	0.00	2	0	0	93826.63
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10

In [63]:

```
X = df[['CreditScore','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary','Male','Germany','Spain']]
```

In [64]:

```
y = df['Exited']
```

In [65]:

```
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)
```

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

In [66]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [67]:

```
X_train  = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [68]:

```
X_train
```

Out[68]:

```
array([[ 4.56838557e-01, -9.45594735e-01,  1.58341939e-03, ...,
         9.13181783e-01, -5.81969145e-01, -5.73611200e-01],
       [-2.07591864e-02, -2.77416637e-01,  3.47956411e-01, ...,
        -1.09507222e+00, -5.81969145e-01,  1.74334114e+00],
       [-1.66115021e-01,  1.82257167e+00, -1.38390855e+00, ...,
        -1.09507222e+00, -5.81969145e-01, -5.73611200e-01],
       ...,
       [-3.63383654e-01, -4.68324665e-01,  1.73344838e+00, ...,
         9.13181783e-01, -5.81969145e-01, -5.73611200e-01],
       [ 4.67221117e-01, -1.42286480e+00,  1.38707539e+00, ...,
         9.13181783e-01, -5.81969145e-01,  1.74334114e+00],
       [-8.82511636e-01,  2.95307447e-01, -6.91162564e-01, ...,
         9.13181783e-01, -5.81969145e-01, -5.73611200e-01]])
```

In [69]:

```
X_test
```

Out[69]:

```
array([[ 3.63395520e-01,  1.99853433e-01,  1.58341939e-03, ...,
         9.13181783e-01, -5.81969145e-01, -5.73611200e-01],
       [-4.15243057e-02,  4.86215475e-01,  1.58341939e-03, ...,
        -1.09507222e+00, -5.81969145e-01,  1.74334114e+00],
       [-1.87923736e+00, -3.72870651e-01, -1.38390855e+00, ...,
         9.13181783e-01, -5.81969145e-01, -5.73611200e-01],
       ...,
       [-6.02182526e-01, -5.63778679e-01, -1.73028154e+00, ...,
        -1.09507222e+00, -5.81969145e-01, -5.73611200e-01],
       [ 1.51585964e+00, -6.59232693e-01,  1.73344838e+00, ...,
         9.13181783e-01, -5.81969145e-01, -5.73611200e-01],
       [-5.19122049e-01,  1.04399419e-01,  1.73344838e+00, ...,
         9.13181783e-01, -5.81969145e-01, -5.73611200e-01]])
```

Building the Classifier Model using Keras

In [70]:

```
import keras #Keras is the wrapper on the top of tenserflow
#Can use Tenserflow as well but won't be able to understand the errors initially.
```

In [71]:

```
from keras.models import Sequential #To create sequential neural network
from keras.layers import Dense #To create hidden layers
```

In [72]:

```
classifier = Sequential()
```

In [74]:

```
#To add the layers
#Dense helps to contruct 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"))
```

In [75]:

```
classifier.add(Dense(activation = "relu",units = 6,kernel_initializer = "uniform"))    #A
dding second hidden layers
```

In [76]:

```
classifier.add(Dense(activation = "sigmoid",units = 1,kernel_initializer = "uniform")) #
Final neuron will be having siigmoid function
```

In [77]:

```
classifier.compile(optimizer="adam",loss = 'binary_crossentropy',metrics = ['accuracy'])
#To compile the Artificial Neural Network. Ussed Binary crossentropy as we just have only
two output
```

In [79]:

```
classifier.summary() #3 layers created. 6 neurons in 1st,6neurons in 2nd layer and 1 neur
on in last
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 6)                 72
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 42
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 7
=================================================================
Total params: 121
Trainable params: 121
Non-trainable params: 0
_________________________________________________________________
```

In [89]:

```
classifier.fit(X_train,y_train,batch_size=10,epochs=50) #Fitting the ANN to training data
set
```

```
Epoch 1/50
700/700 [==============================] - 0s 674us/step - loss: 0.4293 - accuracy: 0.794
7
Epoch 2/50
700/700 [==============================] - 0s 647us/step - loss: 0.4239 - accuracy: 0.794
7
Epoch 3/50
700/700 [==============================] - 0s 657us/step - loss: 0.4203 - accuracy: 0.806
7
Epoch 4/50
700/700 [==============================] - 0s 664us/step - loss: 0.4167 - accuracy: 0.826
0
Epoch 5/50
700/700 [==============================] - 0s 674us/step - loss: 0.4153 - accuracy: 0.828
7
Epoch 6/50
700/700 [==============================] - 0s 653us/step - loss: 0.4137 - accuracy: 0.831
0
Epoch 7/50
700/700 [==============================] - 0s 658us/step - loss: 0.4125 - accuracy: 0.831
7
Epoch 8/50
700/700 [==============================] - 1s 842us/step - loss: 0.4116 - accuracy: 0.830
6
Epoch 9/50
700/700 [==============================] - 0s 671us/step - loss: 0.4103 - accuracy: 0.833
1
Epoch 10/50
700/700 [==============================] - 0s 682us/step - loss: 0.4100 - accuracy: 0.832
6
Epoch 11/50
700/700 [==============================] - 0s 690us/step - loss: 0.4093 - accuracy: 0.833
7
Epoch 12/50
700/700 [==============================] - 0s 688us/step - loss: 0.4087 - accuracy: 0.833
9
```

```
Epoch 13/50
700/700 [==============================] - 0s 675us/step - loss: 0.4081 - accuracy: 0.834
1
Epoch 14/50
700/700 [==============================] - 1s 722us/step - loss: 0.4071 - accuracy: 0.833
1
Epoch 15/50
700/700 [==============================] - 1s 811us/step - loss: 0.4065 - accuracy: 0.834
1
Epoch 16/50
700/700 [==============================] - 0s 711us/step - loss: 0.4056 - accuracy: 0.835
6
Epoch 17/50
700/700 [==============================] - 0s 702us/step - loss: 0.4046 - accuracy: 0.836
6
Epoch 18/50
700/700 [==============================] - 0s 688us/step - loss: 0.4035 - accuracy: 0.834
3
Epoch 19/50
700/700 [==============================] - 1s 715us/step - loss: 0.4024 - accuracy: 0.836
3
Epoch 20/50
700/700 [==============================] - 0s 714us/step - loss: 0.4020 - accuracy: 0.833
7
Epoch 21/50
700/700 [==============================] - 0s 705us/step - loss: 0.4010 - accuracy: 0.837
4
Epoch 22/50
700/700 [==============================] - 1s 720us/step - loss: 0.4003 - accuracy: 0.837
0
Epoch 23/50
700/700 [==============================] - 0s 692us/step - loss: 0.3993 - accuracy: 0.837
4
Epoch 24/50
700/700 [==============================] - 0s 709us/step - loss: 0.3990 - accuracy: 0.835
6
Epoch 25/50
700/700 [==============================] - 1s 871us/step - loss: 0.3984 - accuracy: 0.836
6
Epoch 26/50
700/700 [==============================] - 1s 719us/step - loss: 0.3984 - accuracy: 0.836
7
Epoch 27/50
700/700 [==============================] - 1s 719us/step - loss: 0.3980 - accuracy: 0.836
6
Epoch 28/50
700/700 [==============================] - 0s 695us/step - loss: 0.3981 - accuracy: 0.836
6
Epoch 29/50
700/700 [==============================] - 0s 667us/step - loss: 0.3976 - accuracy: 0.837
4
Epoch 30/50
700/700 [==============================] - 0s 669us/step - loss: 0.3972 - accuracy: 0.837
3
Epoch 31/50
700/700 [==============================] - 0s 670us/step - loss: 0.3970 - accuracy: 0.837
0
Epoch 32/50
700/700 [==============================] - 1s 720us/step - loss: 0.3972 - accuracy: 0.837
6
Epoch 33/50
700/700 [==============================] - 0s 675us/step - loss: 0.3965 - accuracy: 0.836
7
Epoch 34/50
700/700 [==============================] - 0s 680us/step - loss: 0.3961 - accuracy: 0.836
4
Epoch 35/50
700/700 [==============================] - 0s 685us/step - loss: 0.3962 - accuracy: 0.837
9
Epoch 36/50
700/700 [==============================] - 1s 771us/step - loss: 0.3960 - accuracy: 0.837
0
```

```
Epoch 37/50
700/700 [==============================] - 1s 1ms/step - loss: 0.3963 - accuracy: 0.8366
Epoch 38/50
700/700 [==============================] - 1s 764us/step - loss: 0.3962 - accuracy: 0.837
3
Epoch 39/50
700/700 [==============================] - 1s 823us/step - loss: 0.3950 - accuracy: 0.838
4
Epoch 40/50
700/700 [==============================] - 1s 759us/step - loss: 0.3956 - accuracy: 0.836
1
Epoch 41/50
700/700 [==============================] - 1s 773us/step - loss: 0.3949 - accuracy: 0.836
6
Epoch 42/50
700/700 [==============================] - 0s 695us/step - loss: 0.3953 - accuracy: 0.836
9
Epoch 43/50
700/700 [==============================] - 0s 701us/step - loss: 0.3952 - accuracy: 0.836
9
Epoch 44/50
700/700 [==============================] - 0s 707us/step - loss: 0.3952 - accuracy: 0.836
6
Epoch 45/50
700/700 [==============================] - 0s 680us/step - loss: 0.3955 - accuracy: 0.837
6
Epoch 46/50
700/700 [==============================] - 0s 665us/step - loss: 0.3947 - accuracy: 0.837
3
Epoch 47/50
700/700 [==============================] - 0s 708us/step - loss: 0.3947 - accuracy: 0.837
1
Epoch 48/50
700/700 [==============================] - 0s 681us/step - loss: 0.3944 - accuracy: 0.837
1
Epoch 49/50
700/700 [==============================] - 0s 678us/step - loss: 0.3947 - accuracy: 0.838
3
Epoch 50/50
700/700 [==============================] - 1s 869us/step - loss: 0.3944 - accuracy: 0.837
0
```

Out[89]:

```
<tensorflow.python.keras.callbacks.History at 0x1fb1eb93df0>
```

In [90]:

```
y_pred =classifier.predict(X_test)
y_pred = (y_pred > 0.5) #Predicting the result
```

In [97]:

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
```

In [92]:

```
cm = confusion_matrix(y_test,y_pred)
```

In [93]:

```
cm
```

Out[93]:

```
array([[2328,   72],
       [ 425,  175]], dtype=int64)
```

In [94]:

```
accuracy = accuracy_score(y_test,y_pred)
```

In [95]:

```
accuracy
```

Out[95]:

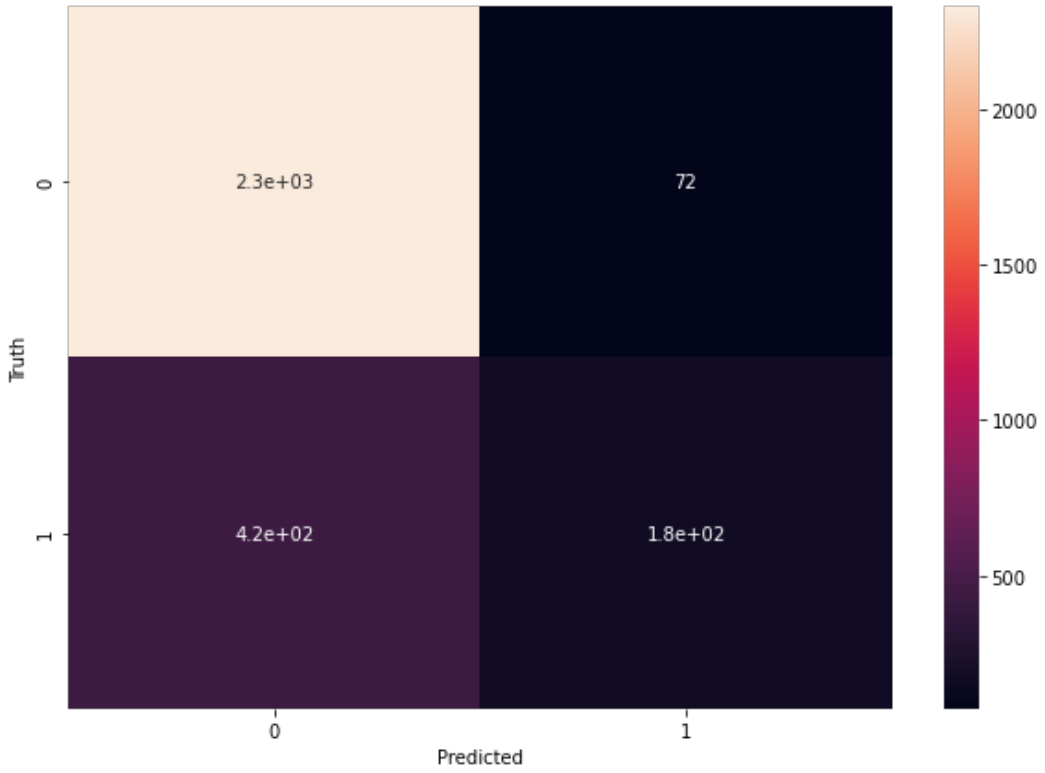
```
0.8343333333333334
```

In [98]:

```
plt.figure(figsize = (10,7))
sns.heatmap(cm,annot = True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[98]:

```
Text(69.0, 0.5, 'Truth')
```



In [100]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.97      0.90      2400
           1       0.71      0.29      0.41       600

    accuracy                           0.83      3000
   macro avg       0.78      0.63      0.66      3000
weighted avg       0.82      0.83      0.81      3000
```

In [ ]:

```

```

Assignment 5

KNN algorithm on diabetes dataset

In [1]:

```
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
```

In [2]:

```
df=pd.read_csv('diabetes.csv')
```

In [3]:

```
df.columns
```

Out[3]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'Pedigree', 'Age', 'Outcome'],
      dtype='object')
```

Check for null values. If present remove null values from the dataset

In [4]:

```
df.isnull().sum()
```

Out[4]:

```
Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
Pedigree         0
Age              0
Outcome          0
dtype: int64
```

In [ ]:

```

```

Outcome is the label/target, other columns are features

In [7]:

```
X = df.drop('Outcome',axis = 1)
y = df['Outcome']
```

In [8]:

```
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)
```

In [9]:

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

knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

In [17]:

```
print("Confusion matrix: ")
cs = metrics.confusion_matrix(y_test,y_pred)
print(cs)
```

```
Confusion matrix:
[[123  28]
 [ 37  43]]
```

In [12]:

```
print("Acccuracy ",metrics.accuracy_score(y_test,y_pred))
```

```
Acccuracy  0.7186147186147186
```

Classification error rate: proportion of instances misclassified over the whole set of instances. Error rate is calculated as the total number of two incorrect predictions (FN + FP) divided by the total number of a dataset (examples in the dataset.

Also error_rate = 1- accuracy

In [29]:

```
total_misclassified = cs[0,1] + cs[1,0]
print(total_misclassified)
total_examples = cs[0,0]+cs[0,1]+cs[1,0]+cs[1,1]
print(total_examples)
print("Error rate",total_misclassified/total_examples)
print("Error rate ",1-metrics.accuracy_score(y_test,y_pred))
```

```
65
231
Error rate 0.2813852813852814
Error rate  0.2813852813852814
```

In [13]:

```
print("Precision score",metrics.precision_score(y_test,y_pred))
```

```
Precision score 0.6056338028169014
```

In [14]:

```
print("Recall score ",metrics.recall_score(y_test,y_pred))
```

```
Recall score  0.5375
```

In [15]:

```
print("Classification report ",metrics.classification_report(y_test,y_pred))
```

```
Classification report                precision    recall  f1-score   support

           0       0.77      0.81      0.79       151
           1       0.61      0.54      0.57        80

    accuracy                           0.72       231
   macro avg       0.69      0.68      0.68       231
```

weighted avg	0.71	0.72	0.71	231

In [4]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [5]:

```
df = pd.read_csv("sales_data_sample.csv")
```

In [6]:

```
df.head()
```

Out[6]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONT
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shipped	1	
1	10121	34	81.35	5	2765.90	5/7/2003 0:00	Shipped	2	
2	10134	41	94.74	2	3884.34	7/1/2003 0:00	Shipped	3	
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shipped	3	
4	10159	49	100.00	14	5205.27	10/10/2003 0:00	Shipped	4	

5 rows × 25 columns

In [7]:

```
df.dtypes
```

Out[7]:

```
ORDERNUMBER           int64
QUANTITYORDERED       int64
PRICEEACH           float64
ORDERLINENUMBER       int64
SALES               float64
ORDERDATE            object
STATUS               object
QTR_ID                int64
MONTH_ID              int64
YEAR_ID               int64
PRODUCTLINE          object
MSRP                  int64
PRODUCTCODE          object
CUSTOMERNAME         object
PHONE                object
ADDRESSLINE1         object
ADDRESSLINE2         object
CITY                 object
STATE                object
POSTALCODE           object
COUNTRY              object
TERRITORY            object
CONTACTLASTNAME      object
CONTACTFIRSTNAME     object
DEALSIZE             object
dtype: object
```

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
ORDERNUMBER            0
QUANTITYORDERED        0
PRICEEACH              0
ORDERLINENUMBER        0
SALES                  0
ORDERDATE              0
STATUS                 0
QTR_ID                 0
MONTH_ID               0
YEAR_ID                0
PRODUCTLINE            0
MSRP                   0
PRODUCTCODE            0
CUSTOMERNAME           0
PHONE                  0
ADDRESSLINE1           0
ADDRESSLINE2        2521
CITY                   0
STATE               1486
POSTALCODE            76
COUNTRY                0
TERRITORY           1074
CONTACTLASTNAME        0
CONTACTFIRSTNAME       0
DEALSIZE               0
dtype: int64
```

In [9]:

```
df.info()
```

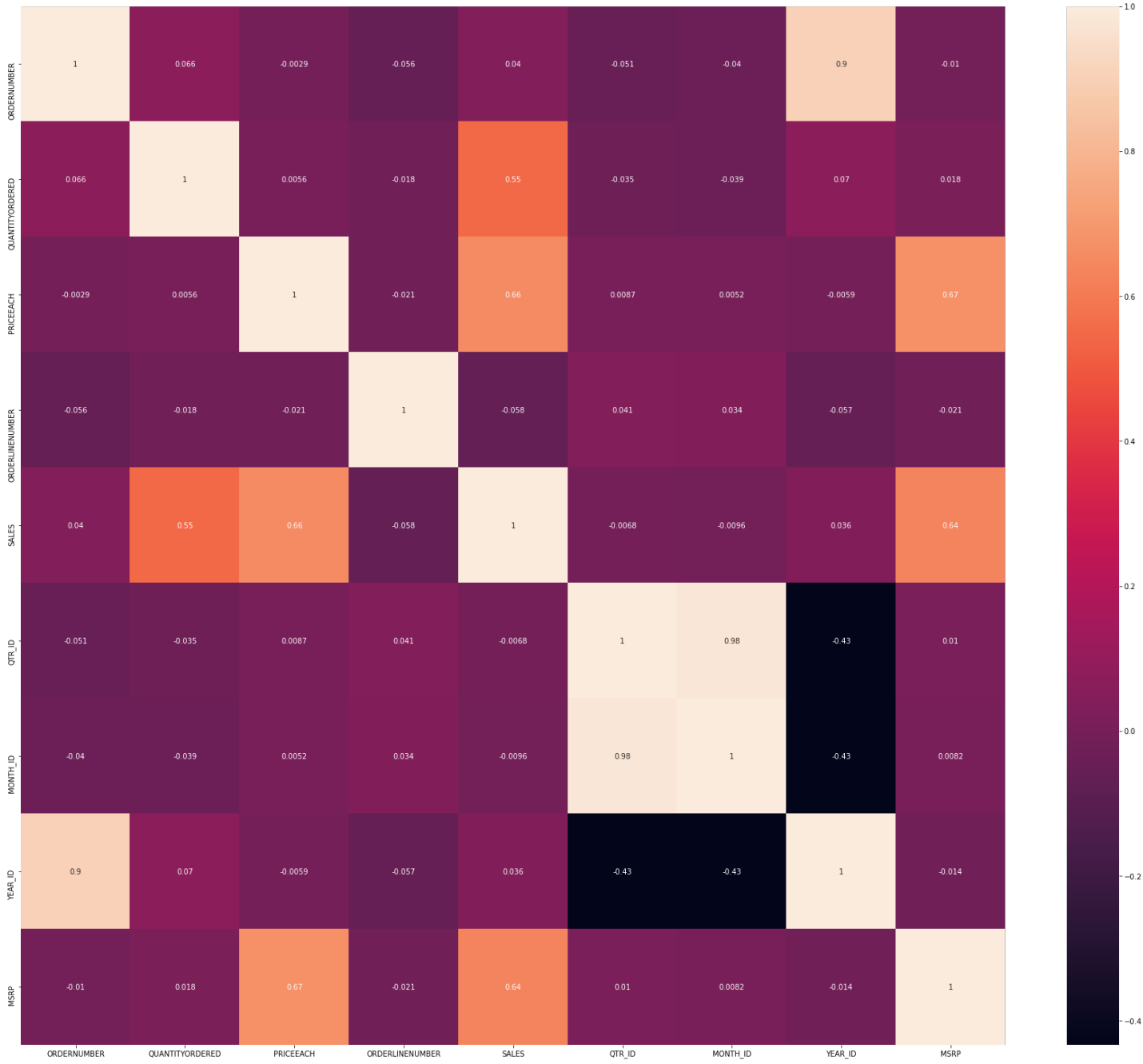
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ORDERNUMBER       2823 non-null   int64  
 1   QUANTITYORDERED   2823 non-null   int64  
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64  
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object 
 6   STATUS            2823 non-null   object 
 7   QTR_ID            2823 non-null   int64  
 8   MONTH_ID          2823 non-null   int64  
 9   YEAR_ID           2823 non-null   int64  
 10  PRODUCTLINE       2823 non-null   object 
 11  MSRP              2823 non-null   int64  
 12  PRODUCTCODE       2823 non-null   object 
 13  CUSTOMERNAME      2823 non-null   object 
 14  PHONE             2823 non-null   object 
 15  ADDRESSLINE1      2823 non-null   object 
 16  ADDRESSLINE2      302 non-null    object 
 17  CITY              2823 non-null   object 
 18  STATE             1337 non-null   object 
 19  POSTALCODE        2747 non-null   object 
 20  COUNTRY           2823 non-null   object 
 21  TERRITORY         1749 non-null   object 
 22  CONTACTLASTNAME   2823 non-null   object 
 23  CONTACTFIRSTNAME  2823 non-null   object 
 24  DEALSIZE          2823 non-null   object 
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

In [10]:

```
plt.figure(figsize = (30,26))
sns.heatmap(df.corr(),annot = True)
```

Out[10]:

<AxesSubplot:>



In [11]:

```
df_drop  = ['ADDRESSLINE1', 'ADDRESSLINE2', 'STATUS','POSTALCODE', 'CITY', 'TERRITORY',
'PHONE', 'STATE', 'CONTACTFIRSTNAME', 'CONTACTLASTNAME', 'CUSTOMERNAME', 'ORDERNUMBER']
df = df.drop(df_drop, axis=1)
```

In [12]:

```
df.head()
```

Out[12]:

	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	QTR_ID	MONTH_ID	YEAR_ID	PRODUCTL
0	30	95.70	2	2871.00	2/24/2003 0:00	1	2	2003	Motorcy
1	34	81.35	5	2765.90	5/7/2003 0:00	2	5	2003	Motorcy
2	41	94.74	2	3884.34	7/1/2003 0:00	3	7	2003	Motorcy
3	45	83.26	6	3746.70	8/25/2003 0:00	3	8	2003	Motorcy

	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	QTR_ID	MONTH_ID	YEAR_ID	PRODUCTLI
4	49	100.00	14	5205.27	10/10/2003 0:00	4	10	2003	Motorcy

In [13]:

```
df.shape
```

Out[13]:

```
(2823, 13)
```

In [14]:

```
df.isnull().sum()
```

Out[14]:

```
QUANTITYORDERED    0
PRICEEACH          0
ORDERLINENUMBER    0
SALES              0
ORDERDATE          0
QTR_ID             0
MONTH_ID           0
YEAR_ID            0
PRODUCTLINE        0
MSRP               0
PRODUCTCODE        0
COUNTRY            0
DEALSIZE           0
dtype: int64
```

In [15]:

```
df.dtypes
```

Out[15]:

```
QUANTITYORDERED      int64
PRICEEACH          float64
ORDERLINENUMBER      int64
SALES              float64
ORDERDATE           object
QTR_ID               int64
MONTH_ID             int64
YEAR_ID              int64
PRODUCTLINE         object
MSRP                 int64
PRODUCTCODE         object
COUNTRY             object
DEALSIZE            object
dtype: object
```

In [ ]:

```

```

In [16]:

```
country = pd.get_dummies(df['COUNTRY'])
productline = pd.get_dummies(df['PRODUCTLINE'])
Dealsize = pd.get_dummies(df['DEALSIZE'])
```

In [17]:

```
df = pd.concat([df,country,productline,Dealsize], axis = 1)
```

In [18]:

```
df.head()
```

Out[18]:

	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	QTR_ID	MONTH_ID	YEAR_ID	PRODUCTI
0	30	95.70	2	2871.00	2/24/2003 0:00	1	2	2003	Motorcy
1	34	81.35	5	2765.90	5/7/2003 0:00	2	5	2003	Motorcy
2	41	94.74	2	3884.34	7/1/2003 0:00	3	7	2003	Motorcy
3	45	83.26	6	3746.70	8/25/2003 0:00	3	8	2003	Motorcy
4	49	100.00	14	5205.27	10/10/2003 0:00	4	10	2003	Motorcy

5 rows × 42 columns

In [19]:

```
df_drop  = ['COUNTRY','PRODUCTLINE','DEALSIZE']
df = df.drop(df_drop, axis=1)
```

In [20]:

```
df.dtypes
```

Out[20]:

```
QUANTITYORDERED       int64
PRICEEACH           float64
ORDERLINENUMBER       int64
SALES               float64
ORDERDATE            object
QTR_ID                int64
MONTH_ID              int64
YEAR_ID               int64
MSRP                  int64
PRODUCTCODE          object
Australia             uint8
Austria               uint8
Belgium               uint8
Canada                uint8
Denmark               uint8
Finland               uint8
France                uint8
Germany               uint8
Ireland               uint8
Italy                 uint8
Japan                 uint8
Norway                uint8
Philippines           uint8
Singapore             uint8
Spain                 uint8
Sweden                uint8
Switzerland           uint8
UK                    uint8
USA                   uint8
Classic Cars          uint8
Motorcycles           uint8
Planes                uint8
Ships                 uint8
Trains                uint8
Trucks and Buses      uint8
Vintage Cars          uint8
Large                 uint8
Medium                uint8
Small                 uint8
dtype: object
```

dtype: object

In [21]:

```
df['PRODUCTCODE'] = pd.Categorical(df['PRODUCTCODE']).codes
```

In [22]:

```
df.dtypes
```

Out[22]:

```
QUANTITYORDERED       int64
PRICEEACH           float64
ORDERLINENUMBER       int64
SALES               float64
ORDERDATE            object
QTR_ID                int64
MONTH_ID              int64
YEAR_ID               int64
MSRP                  int64
PRODUCTCODE            int8
Australia             uint8
Austria               uint8
Belgium               uint8
Canada                uint8
Denmark               uint8
Finland               uint8
France                uint8
Germany               uint8
Ireland               uint8
Italy                 uint8
Japan                 uint8
Norway                uint8
Philippines           uint8
Singapore             uint8
Spain                 uint8
Sweden                uint8
Switzerland           uint8
UK                    uint8
USA                   uint8
Classic Cars          uint8
Motorcycles           uint8
Planes                uint8
Ships                 uint8
Trains                uint8
Trucks and Buses      uint8
Vintage Cars          uint8
Large                 uint8
Medium                uint8
Small                 uint8
dtype: object
```

In [23]:

```
df.drop('ORDERDATE', axis=1, inplace=True)
```

In [24]:

```
df.dtypes
```

Out[24]:

```
QUANTITYORDERED       int64
PRICEEACH           float64
ORDERLINENUMBER       int64
SALES               float64
QTR_ID                int64
MONTH_ID              int64
YEAR_ID               int64
MSRP                  int64
PRODUCTCODE            int8
Australia             uint8
```

```
Austria                uint8
Belgium                uint8
Canada                 uint8
Denmark                uint8
Finland                uint8
France                 uint8
Germany                uint8
Ireland                uint8
Italy                  uint8
Japan                  uint8
Norway                 uint8
Philippines            uint8
Singapore              uint8
Spain                  uint8
Sweden                 uint8
Switzerland            uint8
UK                     uint8
USA                    uint8
Classic Cars           uint8
Motorcycles            uint8
Planes                 uint8
Ships                  uint8
Trains                 uint8
Trucks and Buses       uint8
Vintage Cars           uint8
Large                  uint8
Medium                 uint8
Small                  uint8
dtype: object
```

In [25]:

```
from sklearn.cluster import KMeans
```

In [26]:

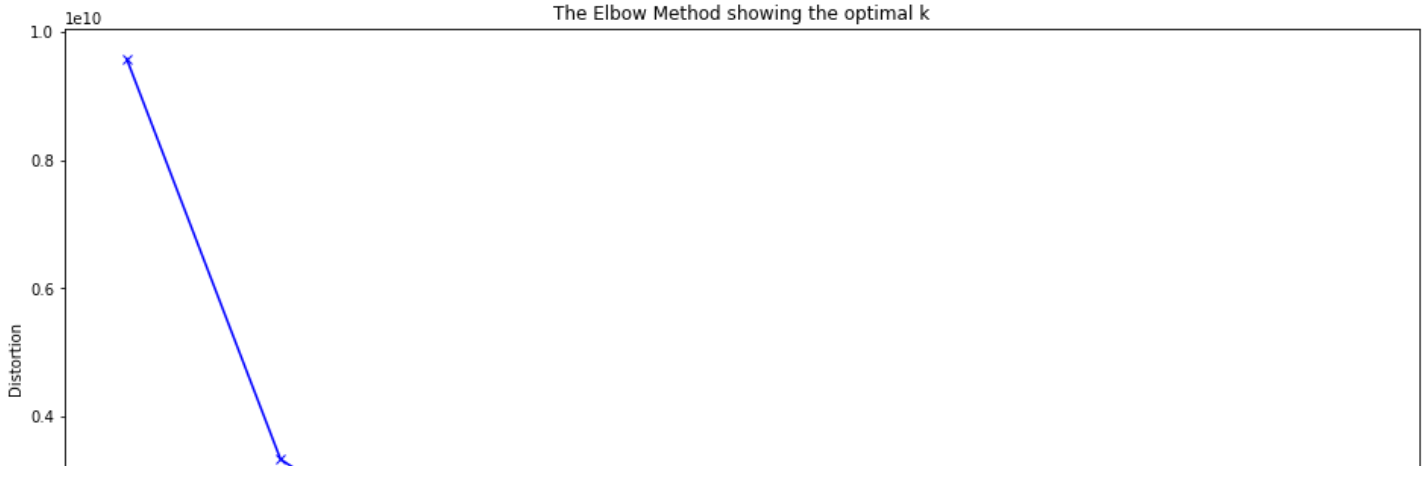
```
WCSS = [] # Withhin Cluster Sum of Squares from the centroid
```

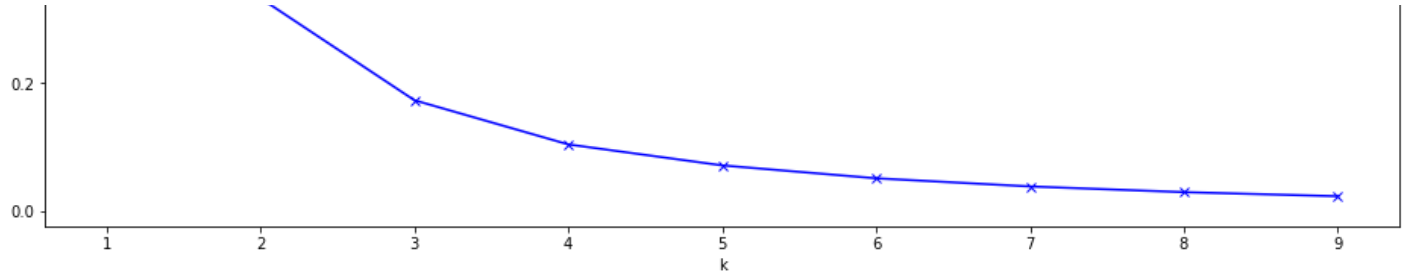
In [27]:

```
distortions = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(df)
    distortions.append(kmeanModel.inertia_)
```

In [28]:

```
plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```





In [29]:

```
kmeanModel = KMeans(n_clusters=3)
y_kmeans = kmeanModel.fit_predict
```

In [30]:

```
plt.scatter(df['y'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, t
olerance)
   3079             try:
-> 3080                 return self._engine.get_loc(casted_key)
   3081             except KeyError as err:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_i
tem()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_i
tem()

KeyError: 'y'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
<ipython-input-30-00540c767b35> in <module>
----> 1 plt.scatter(df['y'])

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   3022             if self.columns.nlevels > 1:
   3023                 return self._getitem_multilevel(key)
-> 3024             indexer = self.columns.get_loc(key)
   3025             if is_integer(indexer):
   3026                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, t
olerance)
   3080                 return self._engine.get_loc(casted_key)
   3081             except KeyError as err:
-> 3082                 raise KeyError(key) from err
   3083
   3084         if tolerance is not None:

KeyError: 'y'
```

In [ ]:

```
print(y_kmeans)
```

In [ ]:

```
plt.figure(figsize = (30,26))
sns.heatmap(df.corr(),annot = True)
```

In [ ]:

```
pip install yellowbrick
```

In [ ]:

```
from yellowbrick.cluster import KElbowVisualizer
```

In [ ]:

```
model = KMeans()
visualizer = KElbowVisualizer(model,k=(1,0),timings = False)
visualizer.fit(df)
visualizer.show()
```

In [ ]:

```

```

In [ ]:

```

```

In [31]:

```
df.head()
```

Out[31]:

	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	QTR_ID	MONTH_ID	YEAR_ID	MSRP	PRODUCTCODE
0	30	95.70	2	2871.00	1	2	2003	95	0
1	34	81.35	5	2765.90	2	5	2003	95	0
2	41	94.74	2	3884.34	3	7	2003	95	0
3	45	83.26	6	3746.70	3	8	2003	95	0
4	49	100.00	14	5205.27	4	10	2003	95	0

5 rows × 38 columns

In [32]:

```
from sklearn.preprocessing import Normalizer
```

In [33]:

```
df_scaled = Normalizer(df)
```

In [34]:

```
df_x = pd.DataFrame(df_scaled,columns = df.columns )
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-34-7343a6fbcd9a> in <module>
----> 1 df_x = pd.DataFrame(df_scaled,columns = df.columns )

~\anaconda3\lib\site-packages\pandas\core\frame.py in __init__(self, data, index, columns
, dtype, copy)
    588         else:
    589             if index is None or columns is None:
--> 590                 raise ValueError("DataFrame constructor not properly called!")
    591
    592             if not dtype:

ValueError: DataFrame constructor not properly called!
```

In [ ]:

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

In [198]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#Importing the required libraries.
```

In [199]:

```
from sklearn.cluster import KMeans, k_means #For clustering
from sklearn.decomposition import PCA #Linear Dimensionality reduction.
```

In [200]:

```
df = pd.read_csv("sales_data_sample.csv") #Loading the dataset.
```

Preprocessing

In [201]:

```
df.head()
```

Out[201]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONT
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shipped	1	
1	10121	34	81.35	5	2765.90	5/7/2003 0:00	Shipped	2	
2	10134	41	94.74	2	3884.34	7/1/2003 0:00	Shipped	3	
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shipped	3	
4	10159	49	100.00	14	5205.27	10/10/2003 0:00	Shipped	4	

5 rows × 25 columns

In [202]:

```
df.shape
```

Out[202]:

```
(2823, 25)
```

In [203]:

```
df.describe()
```

Out[203]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	QTR_ID	MONTH_ID

count	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	2823.SALES	2823.QTR_ID	2823.MONTH_ID	28
mean	10258.725115	35.092809	83.658544	6.466171	3553.889072	2.717676	7.092455	20
std	92.085478	9.741443	20.174277	4.225841	1841.865106	1.203878	3.656633	
min	10100.000000	6.000000	26.880000	1.000000	482.130000	1.000000	1.000000	20
25%	10180.000000	27.000000	68.860000	3.000000	2203.430000	2.000000	4.000000	20
50%	10262.000000	35.000000	95.700000	6.000000	3184.800000	3.000000	8.000000	20
75%	10333.500000	43.000000	100.000000	9.000000	4508.000000	4.000000	11.000000	20
max	10425.000000	97.000000	100.000000	18.000000	14082.800000	4.000000	12.000000	20

In [204]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ORDERNUMBER       2823 non-null   int64  
 1   QUANTITYORDERED   2823 non-null   int64  
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64  
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object 
 6   STATUS            2823 non-null   object 
 7   QTR_ID            2823 non-null   int64  
 8   MONTH_ID          2823 non-null   int64  
 9   YEAR_ID           2823 non-null   int64  
 10  PRODUCTLINE       2823 non-null   object 
 11  MSRP              2823 non-null   int64  
 12  PRODUCTCODE       2823 non-null   object 
 13  CUSTOMERNAME      2823 non-null   object 
 14  PHONE             2823 non-null   object 
 15  ADDRESSLINE1      2823 non-null   object 
 16  ADDRESSLINE2      302 non-null    object 
 17  CITY              2823 non-null   object 
 18  STATE             1337 non-null   object 
 19  POSTALCODE        2747 non-null   object 
 20  COUNTRY           2823 non-null   object 
 21  TERRITORY         1749 non-null   object 
 22  CONTACTLASTNAME   2823 non-null   object 
 23  CONTACTFIRSTNAME  2823 non-null   object 
 24  DEALSIZE          2823 non-null   object 
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

In [205]:

```
df.isnull().sum()
```

Out[205]:

```
ORDERNUMBER          0
QUANTITYORDERED      0
PRICEEACH            0
ORDERLINENUMBER      0
SALES                0
ORDERDATE            0
STATUS               0
QTR_ID               0
MONTH_ID             0
YEAR_ID              0
PRODUCTLINE          0
MSRP                 0
PRODUCTCODE          0
CUSTOMERNAME         0
PHONE                0
```

```
PHONE                0
ADDRESSLINE1         0
ADDRESSLINE2      2521
CITY                 0
STATE             1486
POSTALCODE          76
COUNTRY              0
TERRITORY         1074
CONTACTLASTNAME      0
CONTACTFIRSTNAME     0
DEALSIZE             0
dtype: int64
```

In [206]:

```
df.dtypes
```

Out[206]:

```
ORDERNUMBER           int64
QUANTITYORDERED       int64
PRICEEACH           float64
ORDERLINENUMBER       int64
SALES               float64
ORDERDATE            object
STATUS               object
QTR_ID                int64
MONTH_ID              int64
YEAR_ID               int64
PRODUCTLINE          object
MSRP                  int64
PRODUCTCODE          object
CUSTOMERNAME         object
PHONE                object
ADDRESSLINE1         object
ADDRESSLINE2         object
CITY                 object
STATE                object
POSTALCODE           object
COUNTRY              object
TERRITORY            object
CONTACTLASTNAME      object
CONTACTFIRSTNAME     object
DEALSIZE             object
dtype: object
```

In [207]:

```
df_drop  = ['ADDRESSLINE1', 'ADDRESSLINE2', 'STATUS','POSTALCODE', 'CITY', 'TERRITORY',
'PHONE', 'STATE', 'CONTACTFIRSTNAME', 'CONTACTLASTNAME', 'CUSTOMERNAME', 'ORDERNUMBER']
df = df.drop(df_drop, axis=1) #Dropping the categorical uneccessary columns along with c
olumns having null values. Can't fill the null values are there are alot of null values.
```

In [208]:

```
df.isnull().sum()
```

Out[208]:

```
QUANTITYORDERED    0
PRICEEACH          0
ORDERLINENUMBER    0
SALES              0
ORDERDATE          0
QTR_ID             0
MONTH_ID           0
YEAR_ID            0
PRODUCTLINE        0
MSRP               0
PRODUCTCODE        0
COUNTRY            0
DEALSIZE           0
dtype: int64
```

In [209]:

```
df.dtypes
```

Out[209]:

```
QUANTITYORDERED      int64
PRICEEACH          float64
ORDERLINENUMBER      int64
SALES              float64
ORDERDATE           object
QTR_ID               int64
MONTH_ID             int64
YEAR_ID              int64
PRODUCTLINE         object
MSRP                 int64
PRODUCTCODE         object
COUNTRY             object
DEALSIZE            object
dtype: object
```

In [ ]:

```
# Checking the categorical columns.
```

In [210]:

```
df['COUNTRY'].unique()
```

Out[210]:

```
array(['USA', 'France', 'Norway', 'Australia', 'Finland', 'Austria', 'UK',
       'Spain', 'Sweden', 'Singapore', 'Canada', 'Japan', 'Italy',
       'Denmark', 'Belgium', 'Philippines', 'Germany', 'Switzerland',
       'Ireland'], dtype=object)
```

In [211]:

```
df['PRODUCTLINE'].unique()
```

Out[211]:

```
array(['Motorcycles', 'Classic Cars', 'Trucks and Buses', 'Vintage Cars',
       'Planes', 'Ships', 'Trains'], dtype=object)
```

In [212]:

```
df['DEALSIZE'].unique()
```

Out[212]:

```
array(['Small', 'Medium', 'Large'], dtype=object)
```

In [213]:

```
productline = pd.get_dummies(df['PRODUCTLINE']) #Converting the categorical columns.
Dealsize = pd.get_dummies(df['DEALSIZE'])
```

In [214]:

```
df = pd.concat([df,productline,Dealsize], axis = 1)
```

In [215]:

```
df_drop  = ['COUNTRY','PRODUCTLINE','DEALSIZE'] #Dropping Country too as there are alot o
f countries.
df = df.drop(df_drop, axis=1)
```

In [216]:

```
df['PRODUCTCODE'] = pd.Categorical(df['PRODUCTCODE']).codes #Converting the datatype.
```

In [217]:

```
df.drop('ORDERDATE', axis=1, inplace=True) #Dropping the Orderdate as Month is already in
cluded.
```

In [218]:

```
df.dtypes #All the datatypes are converted into numeric
```

Out[218]:

```
QUANTITYORDERED       int64
PRICEEACH           float64
ORDERLINENUMBER       int64
SALES               float64
QTR_ID                int64
MONTH_ID              int64
YEAR_ID               int64
MSRP                  int64
PRODUCTCODE            int8
Classic Cars          uint8
Motorcycles           uint8
Planes                uint8
Ships                 uint8
Trains                uint8
Trucks and Buses      uint8
Vintage Cars          uint8
Large                 uint8
Medium                uint8
Small                 uint8
dtype: object
```

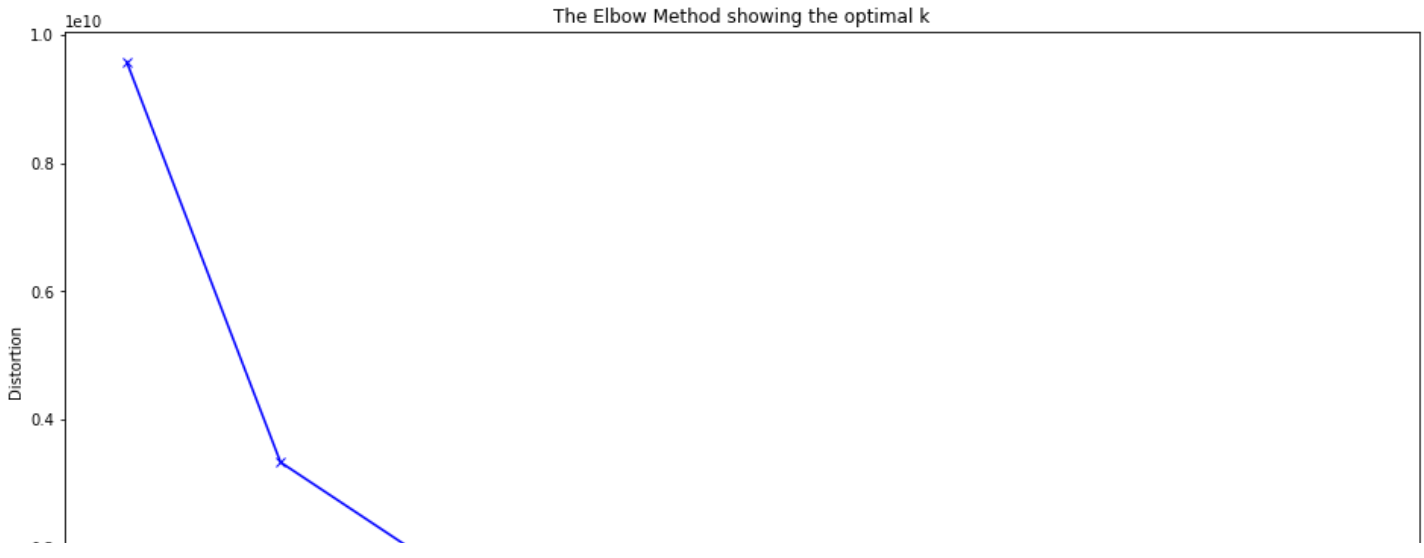
Plotting the Elbow Plot to determine the number of clusters.

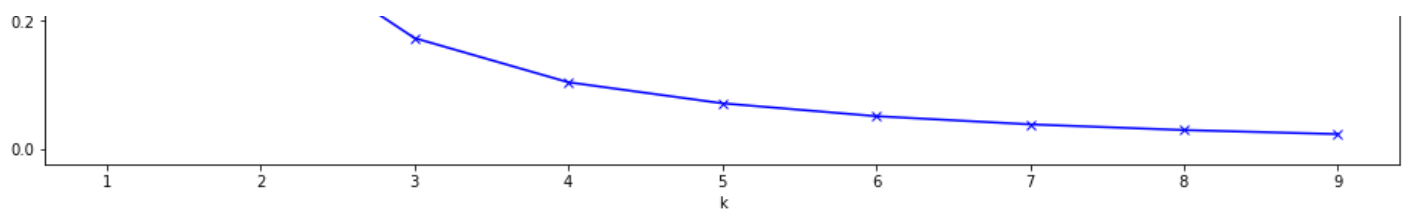
In [219]:

```
distortions = [] # Within Cluster Sum of Squares from the centroid
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(df)
    distortions.append(kmeanModel.inertia_)   #Appeding the intertia to the Distortions
```

In [220]:

```
plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```





As the number of k increases Inertia decreases.

Observations: A Elbow can be observed at 3 and after that the curve decreases gradually.

In [221]:

```
X_train = df.values #Returns a numpy array.
```

In [222]:

```
X_train.shape
```

Out[222]:

```
(2823, 19)
```

In [223]:

```
model = KMeans(n_clusters=3,random_state=2) #Number of cluster = 3
model = model.fit(X_train) #Fitting the values to create a model.
predictions = model.predict(X_train) #Predicting the cluster values (0,1,or 2)
```

In [225]:

```
unique,counts = np.unique(predictions,return_counts=True)
```

In [226]:

```
counts = counts.reshape(1,3)
```

In [227]:

```
counts_df = pd.DataFrame(counts,columns=['Cluster1','Cluster2','Cluster3'])
```

In [228]:

```
counts_df.head()
```

Out[228]:

	Cluster1	Cluster2	Cluster3
0	1083	1367	373

Visualization

In [229]:

```
pca = PCA(n_components=2) #Converting all the features into 2 columns to make it easy to visualize using Principal COmponent Analysis.
```

In [230]:

```
reduced_X = pd.DataFrame(pca.fit_transform(X_train),columns=['PCA1','PCA2']) #Creating a DataFrame.
```

In [231]:

```
reduced_X.head()
```

Out[231]:

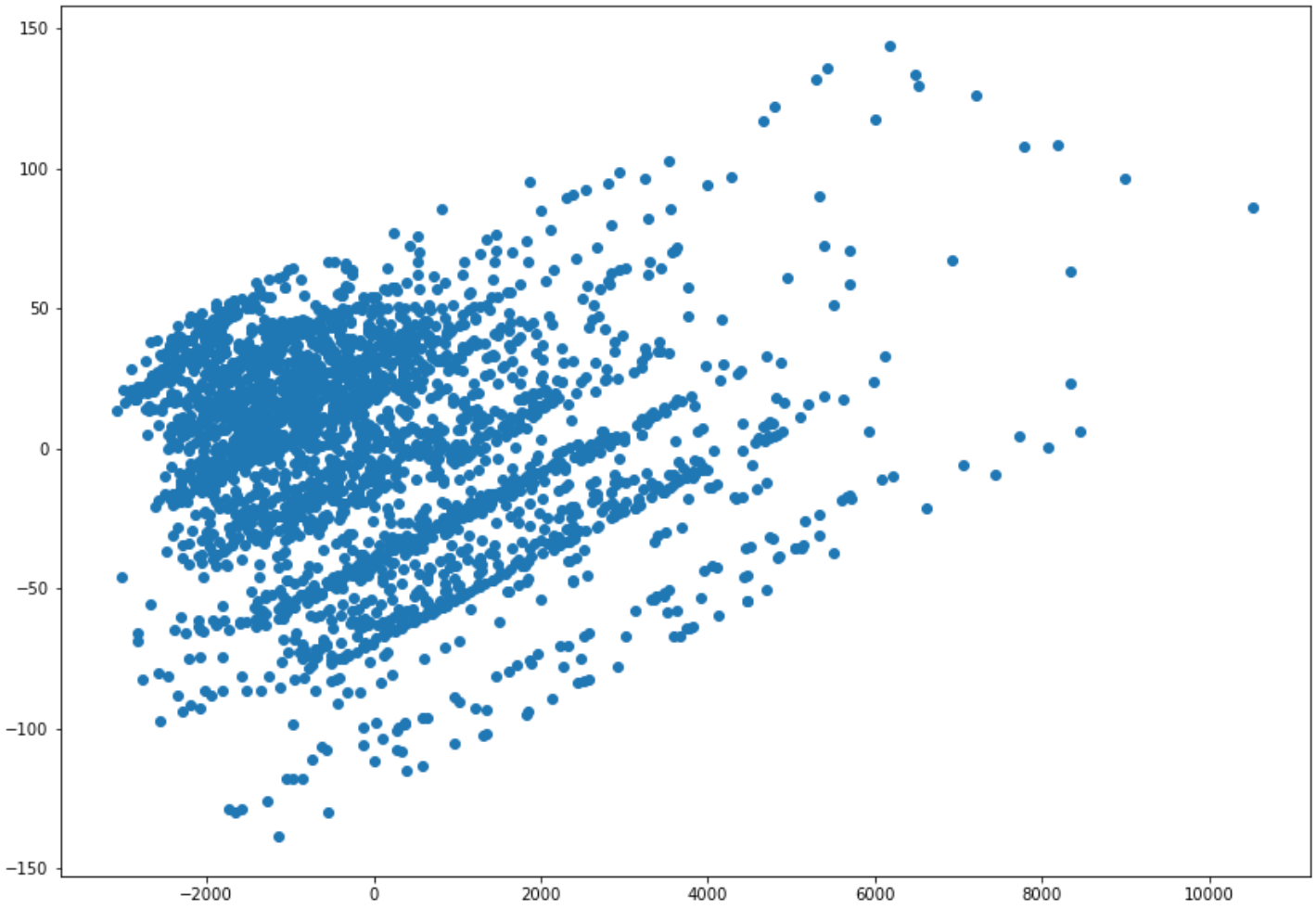
	PCA1	PCA2
0	**-682.488323**	**-42.819535**
1	**-787.665502**	**-41.694991**
2	**330.732170**	**-26.481208**
3	**193.040232**	**-26.285766**
4	**1651.532874**	**-6.891196**

In [232]:

```
#Plotting the normal Scatter Plot
plt.figure(figsize=(14,10))
plt.scatter(reduced_X['PCA1'],reduced_X['PCA2'])
```

Out[232]:

```
<matplotlib.collections.PathCollection at 0x218dc747880>
```



In [233]:

```
model.cluster_centers_ #Finding the centriods. (3 Centriods in total. Each Array contains a centroids for particular feature )
```

Out[233]:

```
array([[ 3.72031394e+01,  9.52120960e+01,  6.44967682e+00,
         4.13868425e+03,  2.72022161e+00,  7.09879963e+00,
         2.00379409e+03,  1.13248384e+02,  5.04469067e+01,
         3.74884580e-01,  1.15420129e-01,  9.41828255e-02,
         8.21791320e-02,  1.84672207e-02,  1.16343490e-01,
         1.98522622e-01,  2.08166817e-17,  1.00000000e+00,
        -6.66133815e-16],
```

```
       [ 3.08302853e+01,  7.00755230e+01,  6.67300658e+00,
         2.12409474e+03,  2.71762985e+00,  7.09509876e+00,
         2.00381127e+03,  7.84784199e+01,  6.24871982e+01,
         2.64813460e-01,  1.21433797e-01,  1.29480614e-01,
         1.00219459e-01,  3.87710315e-02,  9.21726408e-02,
         2.53108998e-01,  6.93889390e-18,  6.21799561e-02,
         9.37820044e-01],
       [ 4.45871314e+01,  9.98931099e+01,  5.75603217e+00,
         7.09596863e+03,  2.71045576e+00,  7.06434316e+00,
         2.00389008e+03,  1.45823056e+02,  3.14959786e+01,
         5.33512064e-01,  1.07238606e-01,  7.23860590e-02,
         2.14477212e-02,  1.07238606e-02,  1.31367292e-01,
         1.23324397e-01,  4.20911528e-01,  5.79088472e-01,
         5.55111512e-17]])
```

In [234]:

```
reduced_centers = pca.transform(model.cluster_centers_) #Transforming the centroids into 3 in x and y coordinates
```

In [235]:

```
reduced_centers
```

Out[235]:

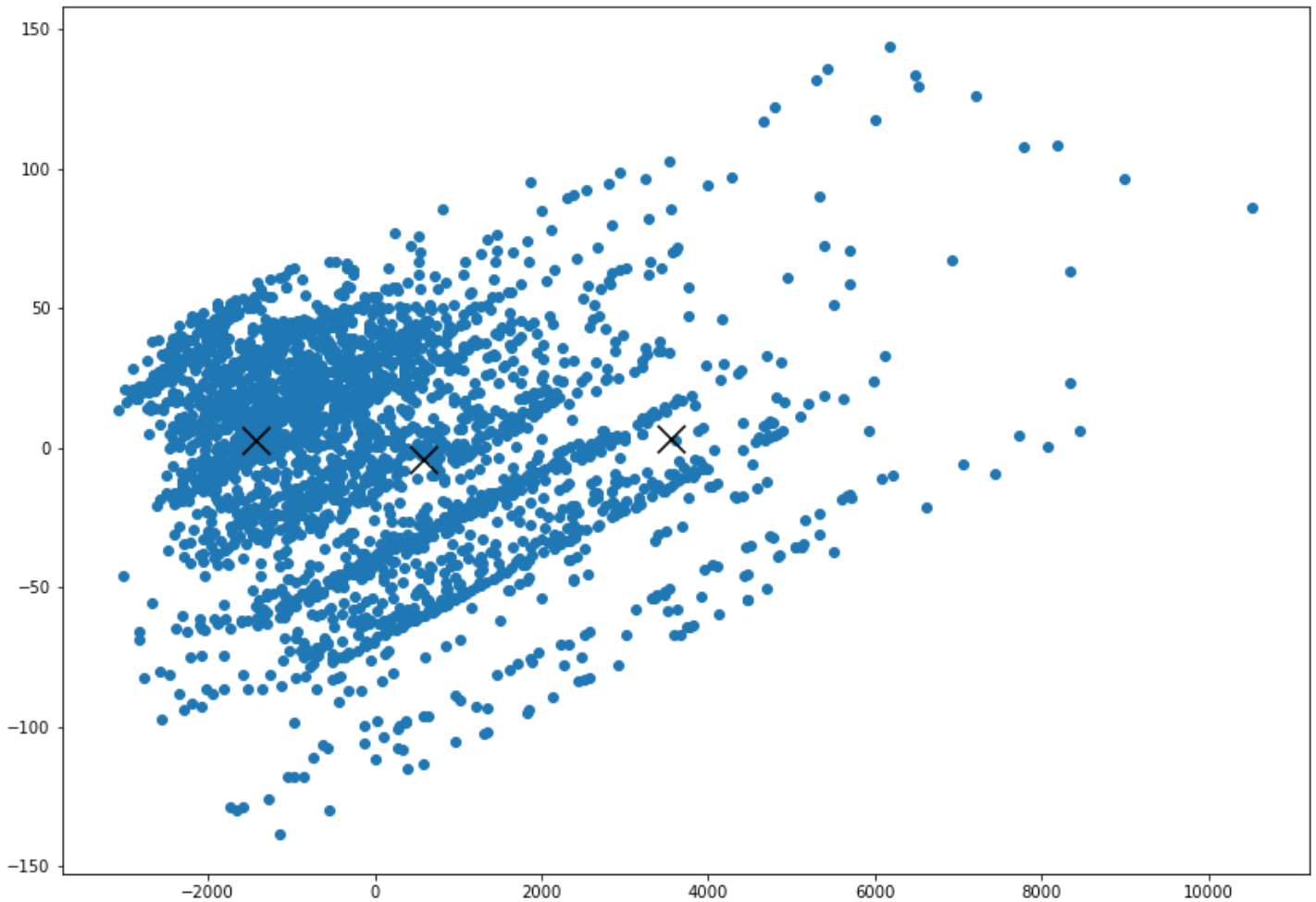
```
array([[ 5.84994044e+02, -4.36786931e+00],
       [-1.43005891e+03,  2.60041009e+00],
       [ 3.54247180e+03,  3.15185487e+00]])
```

In [236]:

```
plt.figure(figsize=(14,10))
plt.scatter(reduced_X['PCA1'],reduced_X['PCA2'])
plt.scatter(reduced_centers[:,0],reduced_centers[:,1],color='black',marker='x',s=300) #Plotting the centriods
```

Out[236]:

```
<matplotlib.collections.PathCollection at 0x218deb6e220>
```



In [237]:

```
reduced_X['Clusters'] = predictions #Adding the Clusters to the reduced dataframe.
```

In [238]:

```
reduced_X.head()
```

Out[238]:

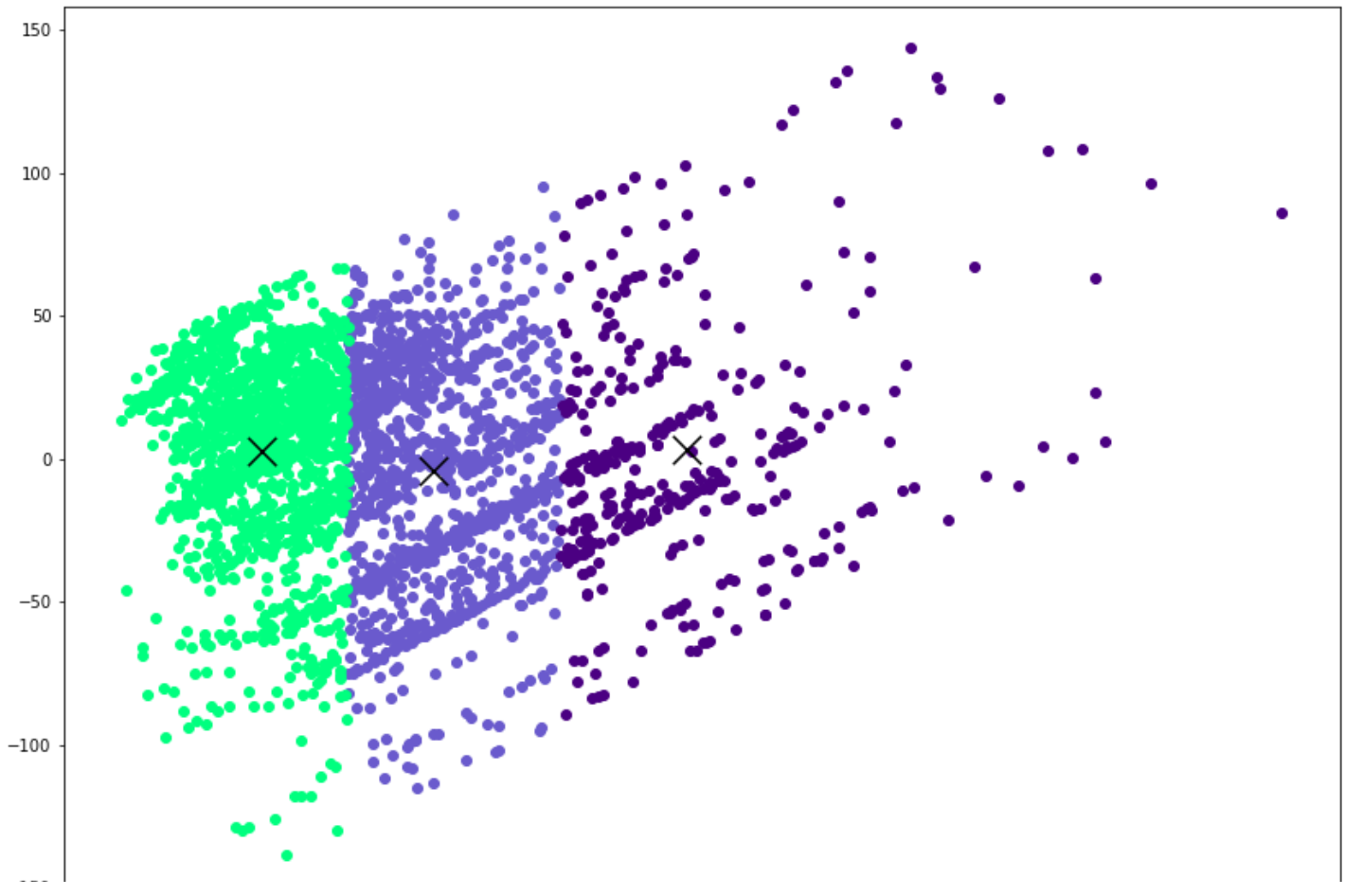
	PCA1	PCA2	Clusters
0	**-682.488323**	**-42.819535**	**1**
1	**-787.665502**	**-41.694991**	**1**
2	**330.732170**	**-26.481208**	**0**
3	**193.040232**	**-26.285766**	**0**
4	**1651.532874**	**-6.891196**	**0**

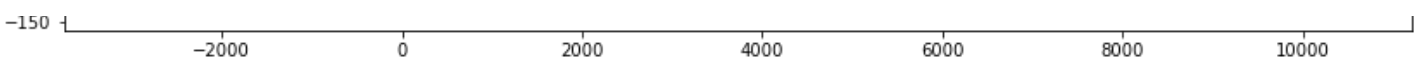
In [239]:

```
#Plotting the clusters
plt.figure(figsize=(14,10))
#                    taking the cluster number and first column           taking the sa
me cluster number and second column      Assigning the color
plt.scatter(reduced_X[reduced_X['Clusters'] == 0].loc[:,'PCA1'],reduced_X[reduced_X['Clus
ters'] == 0].loc[:,'PCA2'],color='slateblue')
plt.scatter(reduced_X[reduced_X['Clusters'] == 1].loc[:,'PCA1'],reduced_X[reduced_X['Clus
ters'] == 1].loc[:,'PCA2'],color='springgreen')
plt.scatter(reduced_X[reduced_X['Clusters'] == 2].loc[:,'PCA1'],reduced_X[reduced_X['Clus
ters'] == 2].loc[:,'PCA2'],color='indigo')


plt.scatter(reduced_centers[:,0],reduced_centers[:,1],color='black',marker='x',s=300)
```

Out[239]:

```
<matplotlib.collections.PathCollection at 0x218dce9e1f0>
```





In [ ]:

Titanic - Machine Learning from Disaster

The Challenge The sinking of the Titanic is one of the most infamous shipwrecks in history.

On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

In this noobook we will go step by step to understand:

1. Getting the data and the librairies
2. Cleaning the data and handel the missing values in differeny ways
3. preprocessing the data
4. quick analysis, visualization on the data to get better understand
5. traing the data on different machine learninig models and make predicitions

+ Code + Text

Will use 5 different machine learning models to train the data and make prediciotn,here are the models will be used:

1. k-nearest neighbors
2. Decision Tree Classifier
3. RandomForestClassifier
4. LogisticRegression
5. Super Vector Machine (svm)

Well, first import libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.optimize as opt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import jaccard_score
import itertools
```

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import svm
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


from google.colab import files
train = files.upload()
```

Choose Files train.csv

- **train.csv**(text/csv) - 61194 bytes, last modified: 12/11/2019 - 100% done

```
Saving train.csv to train.csv
```

```
test = files.upload()
```

Choose Files test.csv

- **test.csv**(text/csv) - 28629 bytes, last modified: 12/11/2019 - 100% done

```
Saving test.csv to test.csv
```

Get the Trian and Test data into DataFrame

```
train_full= pd.read_csv('train.csv')
test_full= pd.read_csv('test.csv')


train_full.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs	female	38.0	1	0	PC 17599	71.

```
test_full.head()
```

```
test_full.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN

```
print('Number of columns in train data :',train_full.shape[1])
print('Number of rows train data :',train_full.shape[0])
print('-'*50)
print('Number of columns in test data :',test_full.shape[1])
print('Number of rows in test data :',test_full.shape[0])
```

```
Number of columns in train data : 12
Number of rows train data : 891
--------------------------------------------------
Number of columns in test data : 11
Number of rows in test data : 418
```

Look for missing values in Train data

```
train_full.isnull().sum().sort_values(ascending=False)
```

```
Cabin          687
Age            177
Embarked         2
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
SibSp            0
Parch            0
Ticket           0
Fare             0
dtype: int64
```

Look for missing values in Test data

```
test_full.isnull().sum().sort_values(ascending=False)
```

```
Cabin          327
Age             86
Fare             1
PassengerId      0
Pclass           0
Name             0
Sex              0
```

```
SibSp           0
Parch           0
Ticket          0
Embarked        0
dtype: int64
```

Looking for Percentage of missing value in both data

```
precent_null_train = train_full.isnull().sum().sum()/np.product(train_full.shape)*100
precent_null_test = test_full.isnull().sum().sum()/np.product(test_full.shape)*100

print('Percetnage of null values in train data: ',precent_null_train)
print('-'*60)
print('Percetnage of null values in test data: ',precent_null_test)
```

```
Percetnage of null values in train data:  8.099513655069211
------------------------------------------------------------
Percetnage of null values in test data:  9.00391474554154
```

```
precent_null_train_cabin= train_full['Cabin'].isnull().sum()/train_full.shape[0]*100
precent_null_test_cabin= test_full['Cabin'].isnull().sum()/test_full.shape[0]*100
print('Percetnage of null values in column Cabin in train data: ',precent_null_train_cabin
print('-'*60)
print('Percetnage of null values in column Cabin in test data: ',precent_null_test_cabin)
```

```
Percetnage of null values in column Cabin in train data:  77.10437710437711
------------------------------------------------------------
Percetnage of null values in column Cabin in test data:  78.22966507177034
```

As we see here 77% of data in Cabin column are missing, so drop this column since it doesnt provide usefull information

```
#drop cabin column
train_full.drop('Cabin',axis=1,inplace=True)
test_full.drop('Cabin',axis=1,inplace=True)


#drop ticket column
train_full.drop('Ticket',axis=1,inplace=True)
test_full.drop('Ticket',axis=1,inplace=True)


#fill missing value in Age column with mean of ages in train data
age_mean= train_full['Age'].mean()
train_full.Age.fillna(round(age_mean),inplace=True)

#fill missing value in Age column with mean of ages in train data
age_mean_test= test_full['Age'].mean()
test_full.Age.fillna(round(age_mean_test),inplace=True)


#Filling missing value in Embarked column with previous value
train_full.Embarked.fillna(method='bfill',inplace=True)
```

```
#Filling missing value in Fare column with previous value
test_full.Fare.fillna(method='bfill',inplace=True)


#looking for duplicated rows in both data
train_full.duplicated().sum(),test_full.duplicated().sum()
```

```
(0, 0)
```

Good no duplicated values in both dataset

```
#quick statistics describtion on the data
train_full.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	F
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000
mean	446.000000	0.383838	2.308642	29.758889	0.523008	0.381594	32.204
std	257.353842	0.486592	0.836071	13.002570	1.102743	0.806057	49.693
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910
50%	446.000000	0.000000	3.000000	30.000000	0.000000	0.000000	14.454
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329

In age column min age is 0.42, so we need to normlize age data

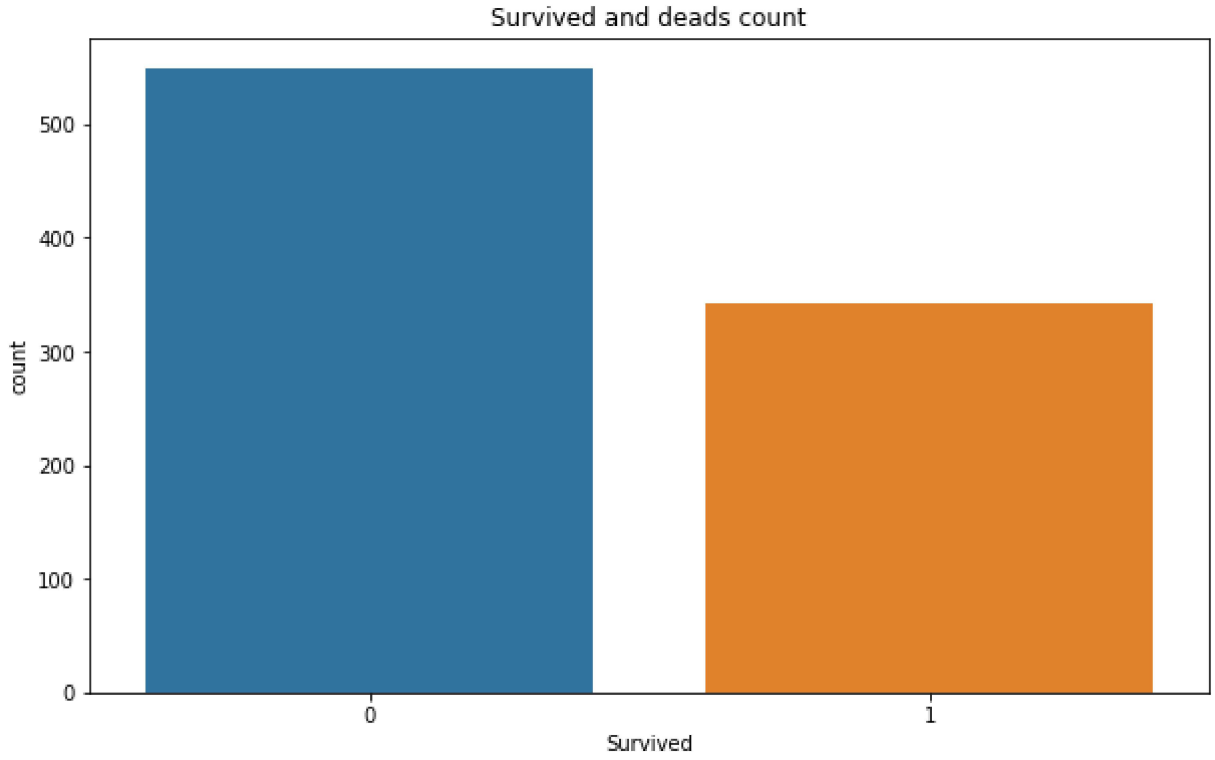
```
#Looking for all value under 1, multiplay *100 and covert all column to int
age_index = train_full[train_full['Age']<1].index
train_full.Age.loc[age_index]= train_full.Age.loc[age_index]*100
train_full['Age'] = train_full['Age'] .astype(int)
```

EDA

Numbers of Surviveds and deads

```
plt.figure(figsize=(10,6))
sns.countplot(train_full.Survived).set_title('Survived and deads count')
plt.show();

dead= round(train_full.query('Survived == 0')['Survived'].count()/train_full.shape[0]*100)
```

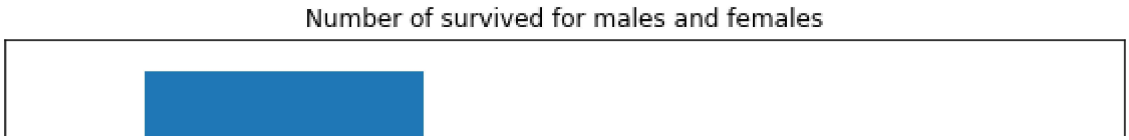
```
survived= 100-dead
print('Percentage of dead: ',dead,'%')
print('Percentage of survived:',survived,'%')
```




```
Percentage of dead:  62 %
Percentage of survived: 38 %
```

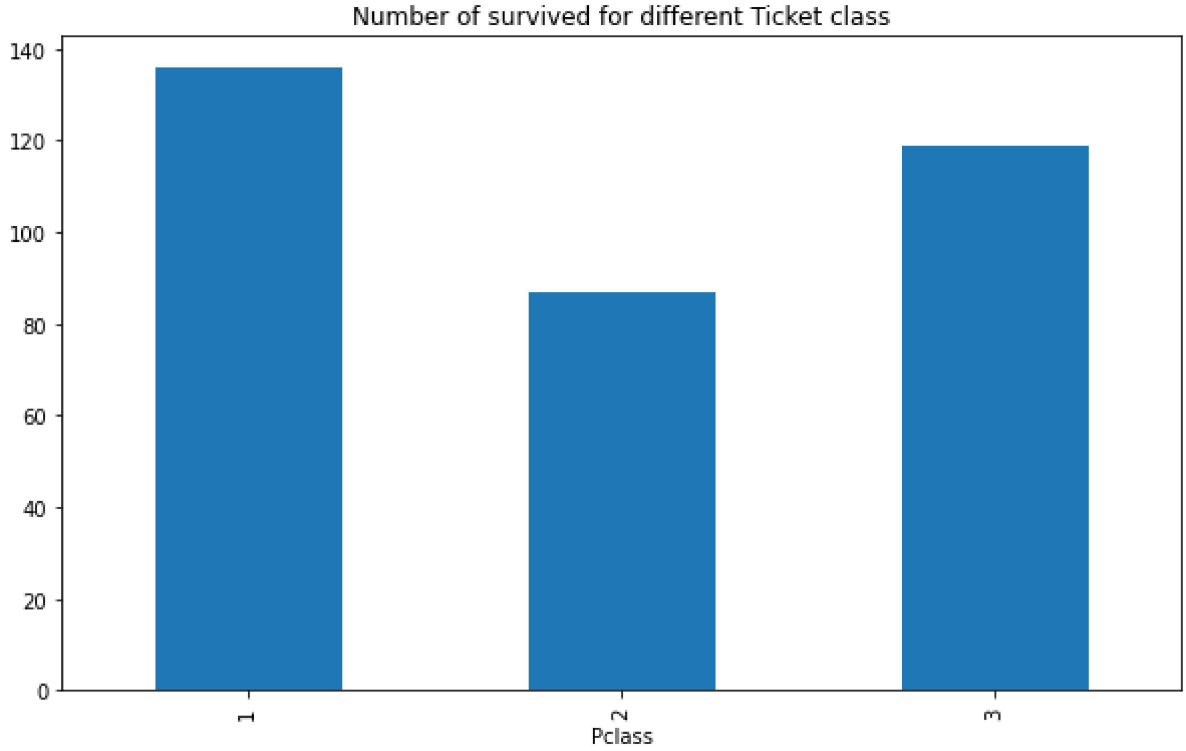
What are most survived males or females?

```
train_full.groupby('Sex').sum()['Survived'].plot(kind='bar',figsize=(10,6))
plt.title('Number of survived for males and females')
plt.show();
```



Number of survived based on classes?

```
train_full.groupby('Pclass').sum()['Survived'].plot(kind='bar',figsize=(10,6))
plt.title('Number of survived for different Ticket class');
plt.show();
```



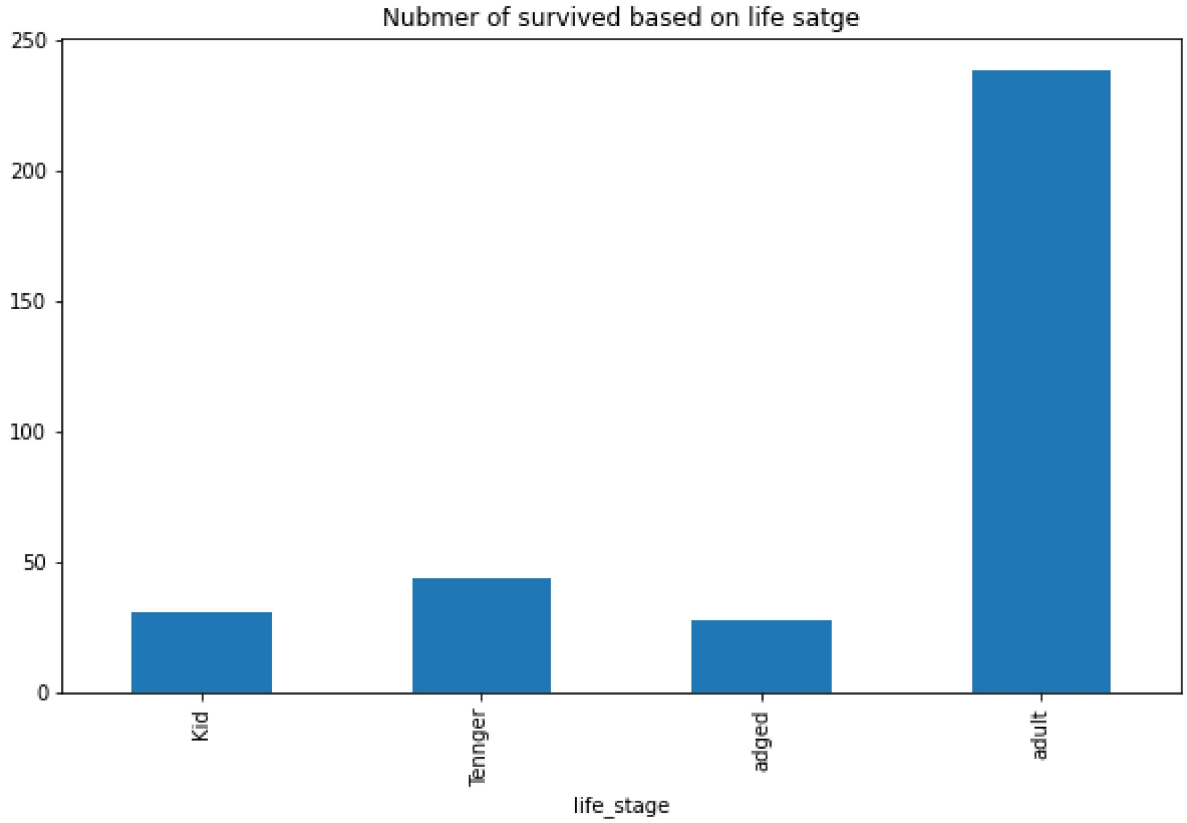
Number of survived based on life stage

```
#divided age to for  categgory kid, , tennger, adult and adged to look for wthic category
train_full['life_stage']=0

for i in train_full['Age'].index:
    if train_full['Age'].loc[i] in list(range(1,11)):
        train_full['life_stage'].loc[i]= 'Kid'

    if train_full['Age'].loc[i] in list(range(11,21)):
        train_full['life_stage'].loc[i]= 'Tennger'

    if train_full['Age'].loc[i] in list(range(21,51)):
        train_full['life_stage'].loc[i]= 'adult'

    if train_full['Age'].loc[i] in list(range(51,120)):
        train_full['life_stage'].loc[i]= 'adged'
```

here we added a new column represnt life stage based on adges

Number of survived based on life stage

```
train_full.groupby('life_stage').sum()['Survived'].plot(kind='bar',figsize=(10,6))
plt.title('Nubmer of survived based on life satge')
plt.show();
```



Adult is the most life stage survived

Looking for unqiue values in categorical columns

```
train_full[['Survived','Pclass','Sex','SibSp','Parch','Embarked','life_stage']].nunique()
```

```
Survived      2
Pclass        3
Sex           2
SibSp         7
Parch         7
Embarked      3
life_stage    4
dtype: int64
```

```
train_full.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          891 non-null    int64  
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Fare         891 non-null    float64
 9   Embarked     891 non-null    object 
 10  life_stage   891 non-null    object 
dtypes: float64(1), int64(6), object(4)
memory usage: 76.7+ KB
```

```
#This function is used to encode categorical column with ONE HOT ENCODING APPROCHE
def oh_encoding(X):
    # Select categorical columns
    categorical_cols = X.select_dtypes(include='object')

    # Select numerical columns
    numerical_cols = X.select_dtypes(exclude='object')


    # Apply one-hot encoder to each column with categorical data
    OH_encoder = OneHotEncoder(handle_unknown='ignore', sparse=False)
    OH_X_full = pd.DataFrame(OH_encoder.fit_transform(categorical_cols))

    # One-hot encoding removed index; put it back
    OH_X_full.index = categorical_cols.index

    # Add one-hot encoded columns to numerical features
    OH_X = pd.concat([numerical_cols, OH_X_full], axis=1)
    return OH_X


#select target column
y = train_full['Survived']

#select indempendant columns
X = ['Pclass','Sex','Age','SibSp','Parch','Fare','Embarked']

#get indempendant column for train data set
X_full = train_full[X].copy()

#get indempendant column for test data set
X_test = test_full[X].copy()



#Use train test split approche
X_train, X_valid, y_train, y_valid = train_test_split(oh_encoding(X_full), y,train_size=0.
```

```
#function to build confusion_matrix,
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

K-Nearest Neighbors

K-Nearest Neighbors is a supervised learning algorithm. Where the data is 'trained' with data points corresponding to their classification. To predict the class of a given data point, it takes into account the classes of the 'K' nearest data points and chooses the class in which the majority of the 'K' nearest data points belong to as the predicted class.

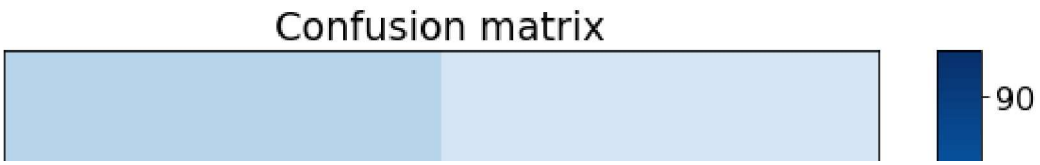
```
#function to find the best result of accuracy score for KNeighborsClassifier
def get_as_dtr(n_neighbors, train_X, val_X, train_y, val_y):
    model = KNeighborsClassifier(n_neighbors=n_neighbors)
    model.fit(train_X, train_y)
    preds_val = model.predict(val_X)
```

```
        ae = accuracy_score(val_y, preds_val)
        return(ae)
    # loop to find the ideal n_neighbors size
    scores = {k_size: get_as_dtr(k_size, X_train, X_valid, y_train, y_valid) for k_size in ran
    best_k_size = max(scores, key=scores.get)

    ae_decision = scores[best_k_size]
    print('Best result at max n_neighbors : ',best_k_size)
    print('accuracy score for KNeighborsClassifier: ',ae_decision)
```

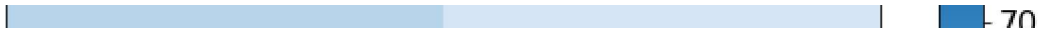
```
Best result at max n_neighbors :  7
accuracy score for KNeighborsClassifier:  0.7430167597765364
```

```
    #build knn model for best score
    kn_model = KNeighborsClassifier(n_neighbors=best_k_size)
    kn_model.fit(X_train, y_train)
    preds_kn = kn_model.predict(X_valid)


    # Compute confusion matrix
    cnf_matrix = confusion_matrix(y_valid,preds_kn, labels=[1,0])
    np.set_printoptions(precision=2)

    # Plot non-normalized confusion matrix
    plt.figure(figsize=(10,8))
    plt.rcParams['font.size'] = '16'
    plot_confusion_matrix(cnf_matrix, classes=['Survived=1','Survived=0'],normalize= False,  t
```

```
Confusion matrix, without normalization
[[39 30]
 [16 94]]
```



DecisionTreeClassifier

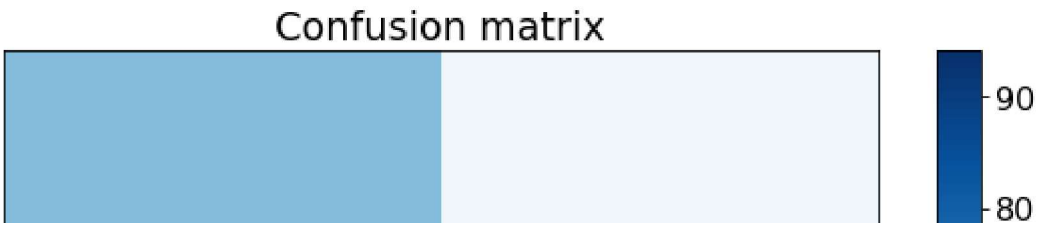



```
dtc_model = DecisionTreeClassifier(criterion="entropy", max_depth = 3)
dtc_model.fit(X_train, y_train)
prediction_dtc = dtc_model.predict(X_valid)

print('accuracy score for DecisionTreeClassifier: ',accuracy_score(y_valid,prediction_dtc)
```

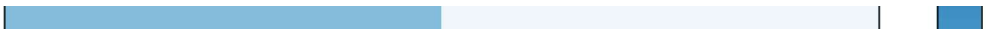
```
accuracy score for DecisionTreeClassifier:  0.8044692737430168
```

```
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_valid,prediction_dtc, labels=[1,0])
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure(figsize=(10,8))
plt.rcParams['font.size'] = '16'
plot_confusion_matrix(cnf_matrix, classes=['Survived=1','Survived=0'],normalize= False,  t
```

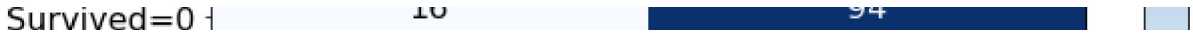
```
Confusion matrix, without normalization
[[50 19]
 [16 94]]
```



Random Forest Classifier



```
#function to find resturn the result of accuracy score for KNeighborsClassifier
def get_rfc(n_estimators, train_X, val_X, train_y, val_y):
    model = model = RandomForestClassifier(n_estimators=n_estimators, max_depth=5, random_
    model.fit(train_X, train_y)
    preds_val = model.predict(val_X)
    ae = accuracy_score(val_y, preds_val)
    return(ae)
```

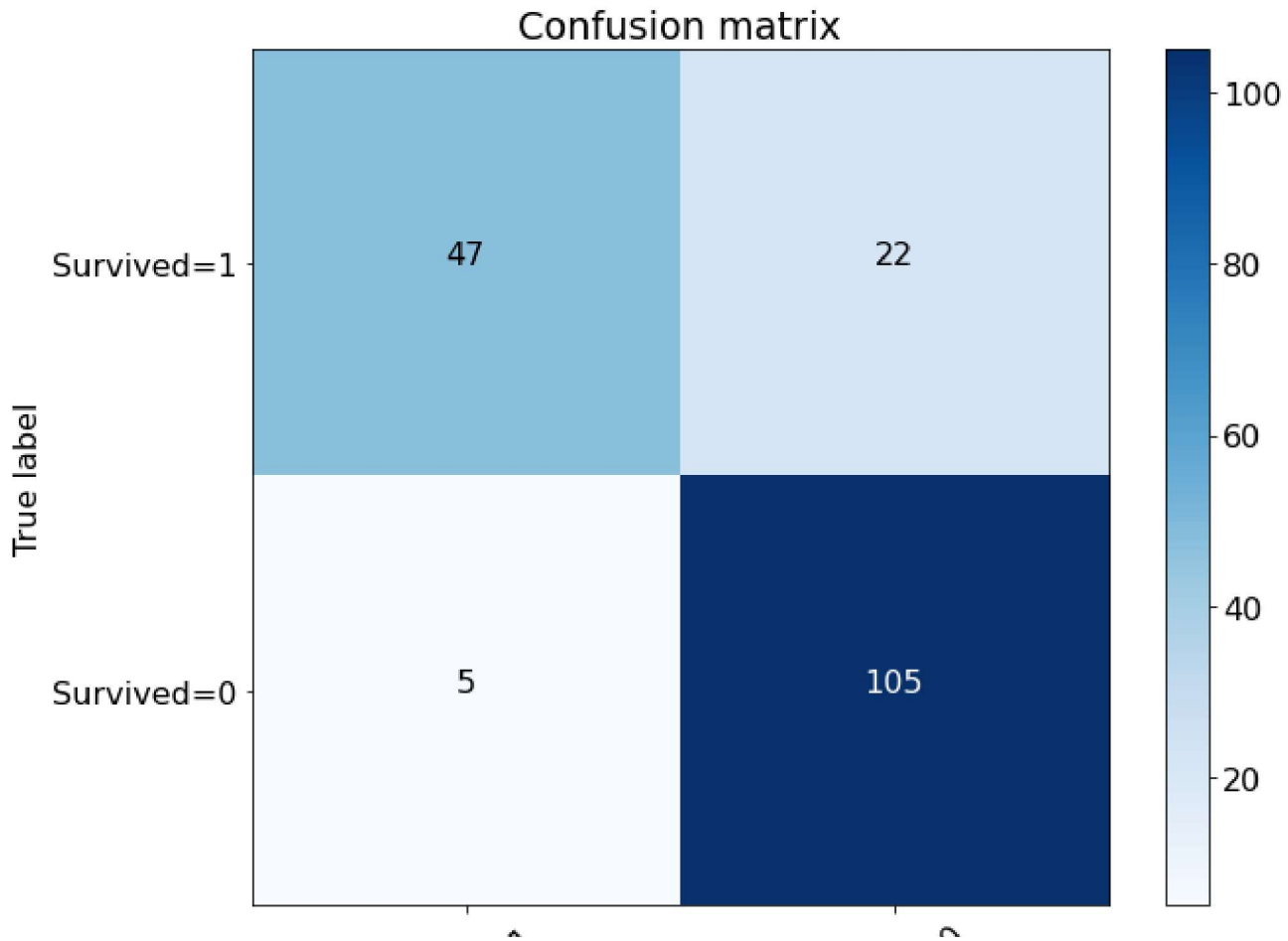



```
# loop to find the ideal n_neighbors size
scores = {n_size: get_rfc(n_size, X_train, X_valid, y_train, y_valid) for n_size in range(
best_n_size = max(scores, key=scores.get)

best_score = scores[best_n_size]
print('Best result at max n_neighbors : ',best_n_size)
print('accuracy score for Random Forest Classifier: ',best_score)
```

```
Best result at max n_neighbors :  16
accuracy score for Random Forest Classifier:  0.8491620111731844
```

```
#Build RandomForestClassifier for best scoore at best n_estimators
model_rfc = RandomForestClassifier(n_estimators=best_n_size, max_depth=5, random_state=1)
model_rfc.fit(X_train, y_train)
preds_rfc = model_rfc.predict(X_valid)


# Compute confusion matrix
cnf_matrix = confusion_matrix(y_valid,preds_rfc, labels=[1,0])
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure(figsize=(10,8))
plt.rcParams['font.size'] = '16'
plot_confusion_matrix(cnf_matrix, classes=['Survived=1','Survived=0'],normalize= False,  t
```

```
Confusion matrix, without normalization
[[ 47  22]
 [  5 105]]
```



Logistic Regression

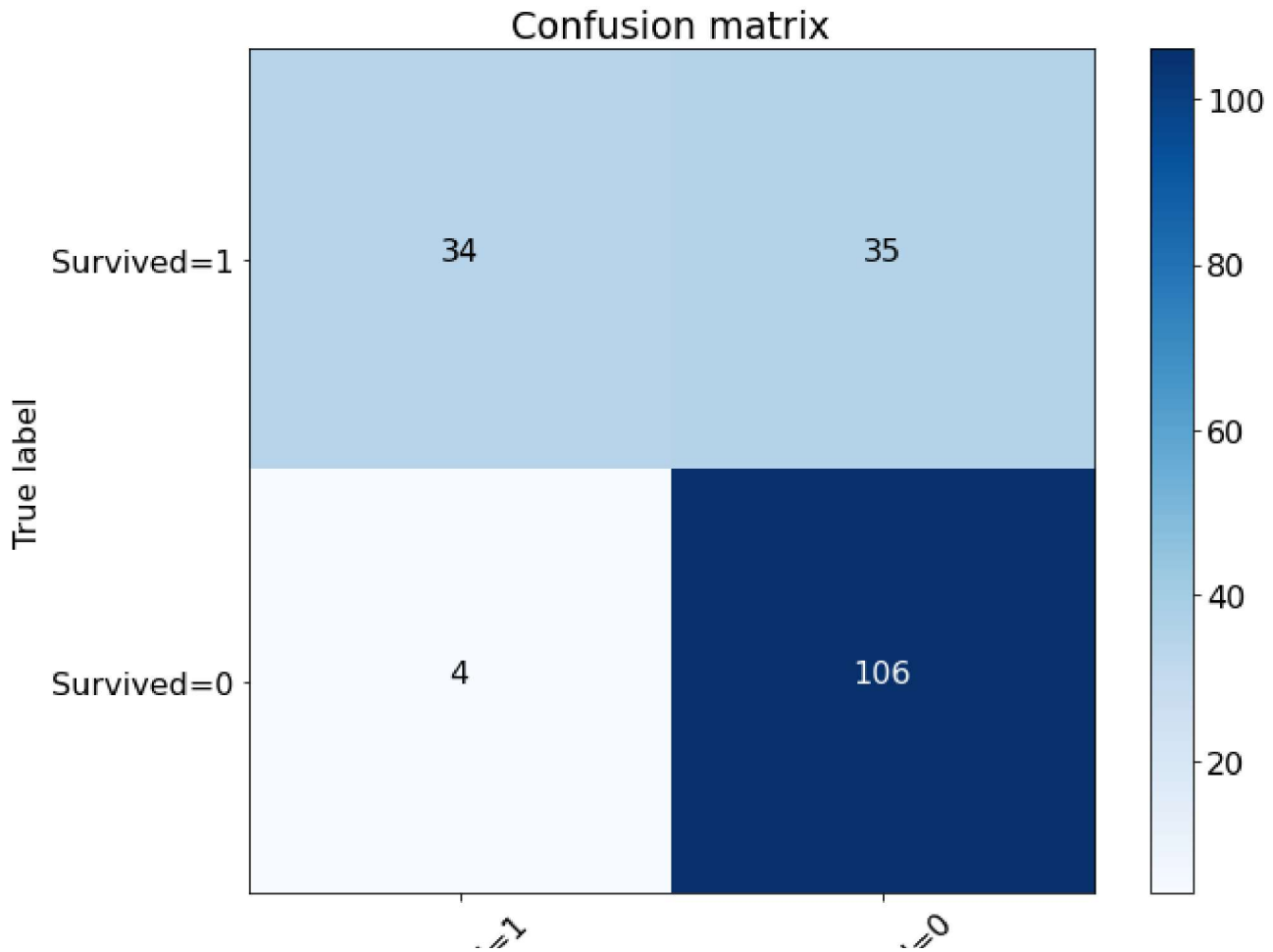
Predicted label

```
LR_model = LogisticRegression(C=0.01, solver='liblinear')
LR_model.fit(X_train,y_train)
prediction_lr = LR_model.predict(X_valid)
print('accuracy score for Logistic Regression',accuracy_score(y_valid,prediction_lr))
```

```
accuracy score for Logistic Regression 0.7821229050279329
```

```
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_valid,prediction_lr, labels=[1,0])
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure(figsize=(10,8))
plt.rcParams['font.size'] = '16'
plot_confusion_matrix(cnf_matrix, classes=['Survived=1','Survived=0'],normalize= False,  t
```

```
Confusion matrix, without normalization
[[ 34  35]
 [  4 106]]
```



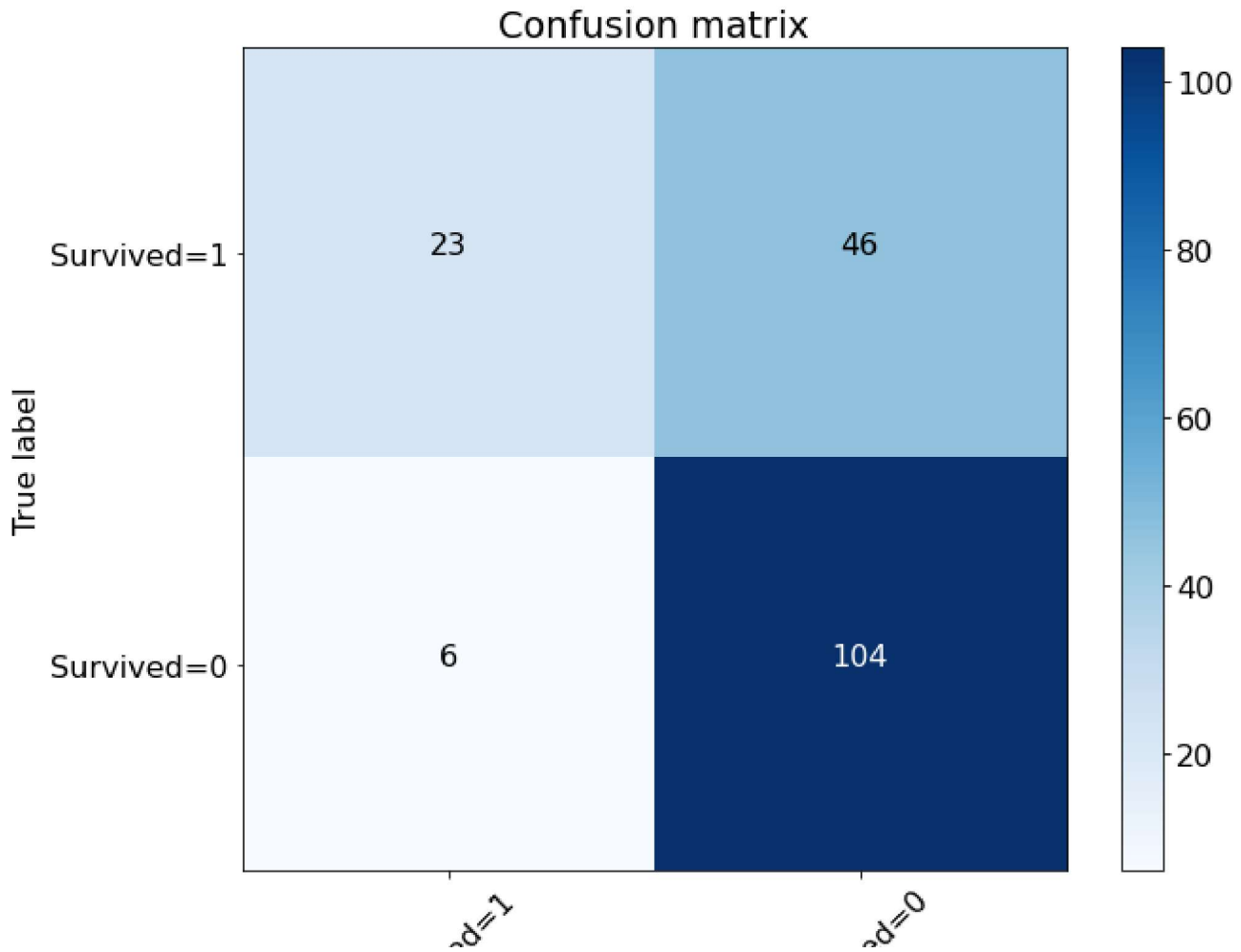
Support Vector Machines

```
svm_model = svm.SVC(kernel='rbf')
svm_model.fit(X_train, y_train)
predicition_svm = svm_model.predict(X_valid)
print('accuracy score for Support Vector Machines',accuracy_score(y_valid,predicition_svm)
```

```
accuracy score for Support Vector Machines 0.7094972067039106
```

```
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_valid,predicition_svm, labels=[1,0])
np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plt.figure(figsize=(10,8))
plt.rcParams['font.size'] = '16'
plot_confusion_matrix(cnf_matrix, classes=['Survived=1','Survived=0'],normalize= False,  t
```

```
Confusion matrix, without normalization
[[ 23  46]
 [  6 104]]
```



Predictions Scores For All Five Models

As we seen all model give similiar Acuuracy score, so lets use some diffrent aproches to look for predicition score

```
accu_preds = [preds_kn,prediction_dtc,preds_rfc,prediction_lr,predicition_svm]

accu_df = pd.DataFrame(index=['K-Nearest Neighbors',
                              'DecisionTreeClassifier',
                              'Random Forest Classifier',
                              'Logistic Regression',
                              'Support Vector Machines'])

accu_df['Acuuracy_score']=[accuracy_score(y_valid,i) for i in accu_preds]
accu_df['F1_score']=[f1_score(y_valid, i, average='weighted')  for i in accu_preds]
accu_df['Jaccard_score']=[jaccard_score(y_valid, i,pos_label=0)  for i in accu_preds]
accu_df
```

	Acuuracy_score	F1_score	Jaccard_score
K-Nearest Neighbors	0.743017	0.736197	0.671429
DecisionTreeClassifier	0.804469	0.803612	0.728682
Random Forest Classifier	0.849162	0.843976	0.795455

As wee see from above table, Almost all values are the samaliar, so lets use Random Forest Classifier to submit the compition

```
model= RandomForestClassifier(n_estimators=best_n_size, max_depth=5, random_state=1)

# Preprocessing of training data, fit model
model.fit(oh_encoding(X_full), y)

# Preprocessing of validation data, get predictions
predictions = model.predict(oh_encoding(X_test))

output = pd.DataFrame({'PassengerId': test_full.PassengerId, 'Survived': predictions})
output.to_csv('submission.csv', index=False)
print("Your submission was successfully saved!")
```

```
Your submission was successfully saved!
```

Mini Project

Title of Project : 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.).

Dataset Link: https://www.kaggle.com/competitions/titanic/data

Objectives :

- To predict the survival of passengers based off a set of data.
- To use SciKit-Learn library in Python, including
  a. Building decision tree
  b. Building Random Forests
  c. Using Support Vector Machine
  d. Using Logistic Regression algorithm
- To apply machine learning algorithms, and analyze results.

Result :

Therefore the accuracy of the models are:

a) K-Nearest Neigbour - 0.70
b) Random Forest Classifier - 0.84
c) Decision Tree Classifier - 0.80

Conclusion :

Thus, we have concluded that Random Forest Classifier predicts the outcome of Titanic Dataset most accurately.

We have created a machine learning model that predicts the survival of passengers based on Demographic data.

Assignment No: 11

11.1 Title

Assignment based on installation Metamask

11.2 Problem Definition:

Installation of Metamask and study spending Ether per transaction.

11.4 Software Requirements:

Mozilla Firefox

11.6 Learning Objectives:

Learn Installation of Metamask and study spending Ether per transaction.

11.7 Outcomes:

After completion of this assignment students are able to understand the Installation of Metamask and study spending Ether per transaction.

11.8 Theory Concepts:

IntroductiontoBlockchain

- Blockchain can be described as a data structure that holds transactional records andwhile ensuring security, transparency, and decentralization. You can also think of it asa chain or records stored in the forms of blocks which are controlled by no singleauthority.
- Ablockchainisadistributedledgerthatiscompletelyopentoanyandeveryoneonthe network. Once information is stored on a blockchain, it is extremely difficult tochange or alters it.
- Each transaction on a blockchain is secured with a digital signature that proves itsauthenticity. Due to the use of encryption and digital signatures, the data stored on theblockchainis tamper-proof and cannot be changed.
- Blockchaintechnologyallowsallthe network participants to reach an agreement, commonlyknownasconsensus.Allthedatastoredonablockchainis recordeddigitally and has a common history which is available for all the network participants. This way, the chances of any fraudulent activity or duplication of transactions iseliminatedwithout the need ofa third-party.

BlockchainFeatures

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

- **Decentralized**

  Blockchains are decentralized in nature meaning that no single person or group holdsthe authority of the overall network. While everybody in the network has the copy ofthe distributed ledger with them, no one can modify it on his or her own. This uniquefeatureofblockchainallowstransparencyand security while giving power to theusers.

- **Peer-to-PeerNetwork**

  With the use of Blockchain, the interaction between two parties through a peer-to-peermodel is easily accomplished without the requirement of any third party. Blockchainuses P2P protocol which allows all the network participants to hold an identical copyof transactions, enabling approval through a machine consensus. For example, if youwish to make any transaction from one part of the world to another, you can do thatwithblockchain all by yourself within a few seconds. Moreover, any interruptions orextracharges will notbe deducted in thetransfer.

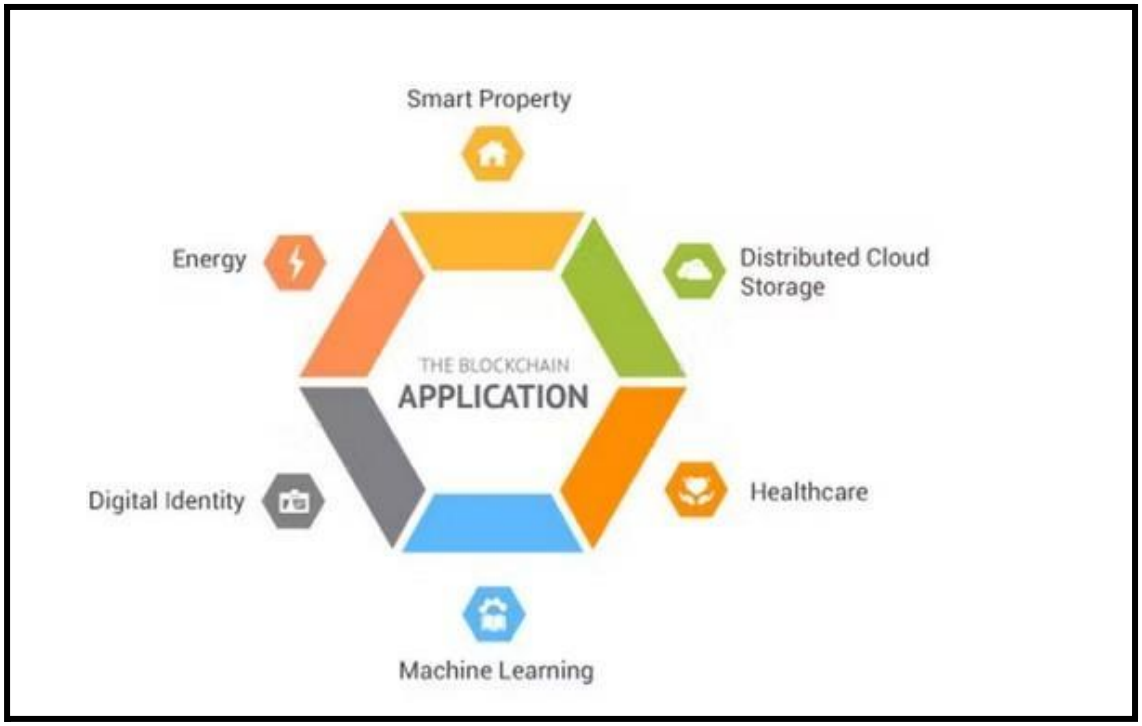
- **Immutable**

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

- **Tamper-Proof**

  Withthepropertyofimmutability embedded in blockchains, it becomes easier todetect tampering of any data. Blockchains are considered tamper-proof as any changein even one single block can be detected and addressed smoothly. There are two keywaysof detecting tamperingnamely, hashes and blocks.

PopularApplicationsofBlockchainTechnology



Benefits of BlockchainTechnology:

- **Time-saving:** No central Authority verification needed for settlements making theprocessfaster and cheaper.
- **Cost-saving:** A Blockchain network reduces expenses in several ways. No need for third partyverification.Participantscanshareassetsdirectly.Intermediariesarereduced.Transactioneffortsareminimizedaseveryparticipanthasacopyofsharedledger.

Tighter security: Noone can temper with Block chain Data as it is shared among millions of participants. The system is safe against cybercrimes and Fraud.In finance market trading, Fibonacci retracement levels are widely used in technicalanalysis.

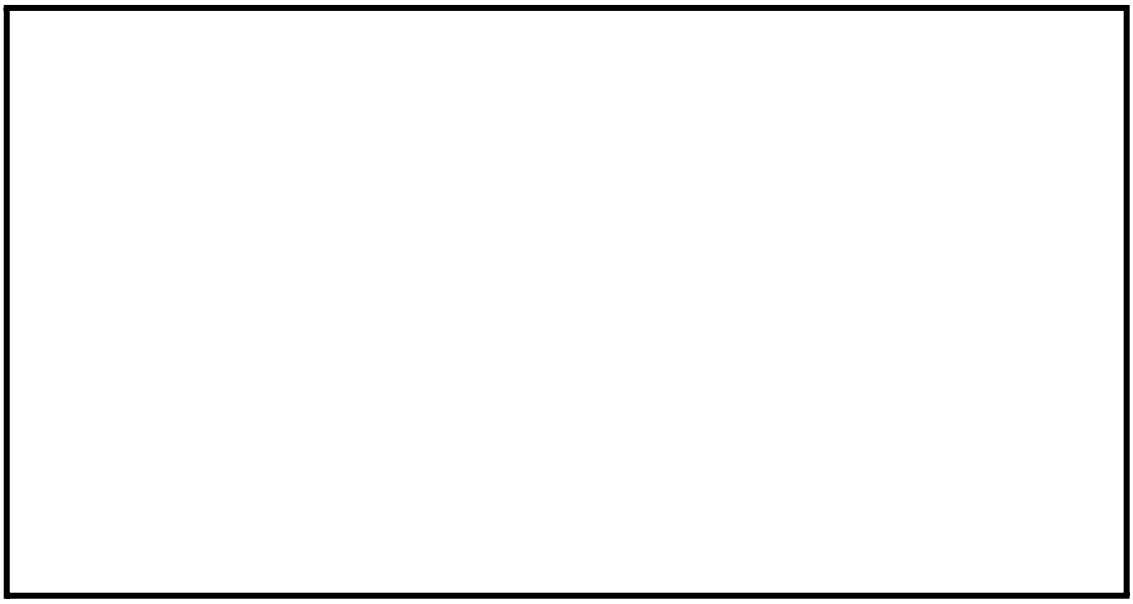
11.9 Implementation & screenshot

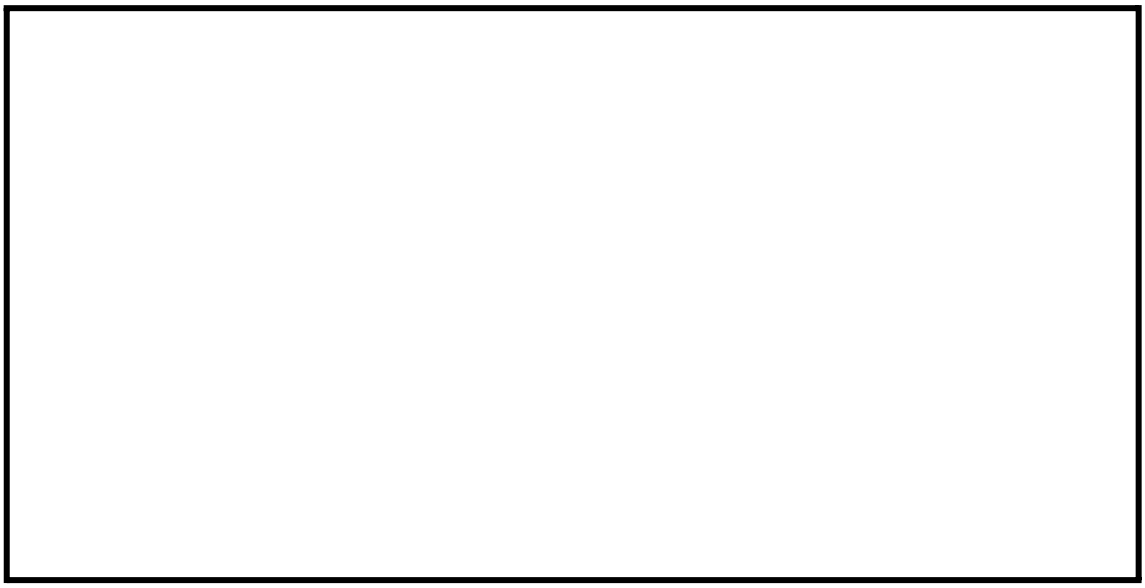
How to use MetaMask:A step by step guide

MetaMask is one of the most popular browser extensions that serves as a way of storing yourEthereumandotherERC20Tokens.Theextensionisfreeandsecure,allowingwebapplicationsto read and interactwithEthereum'sblockchain.

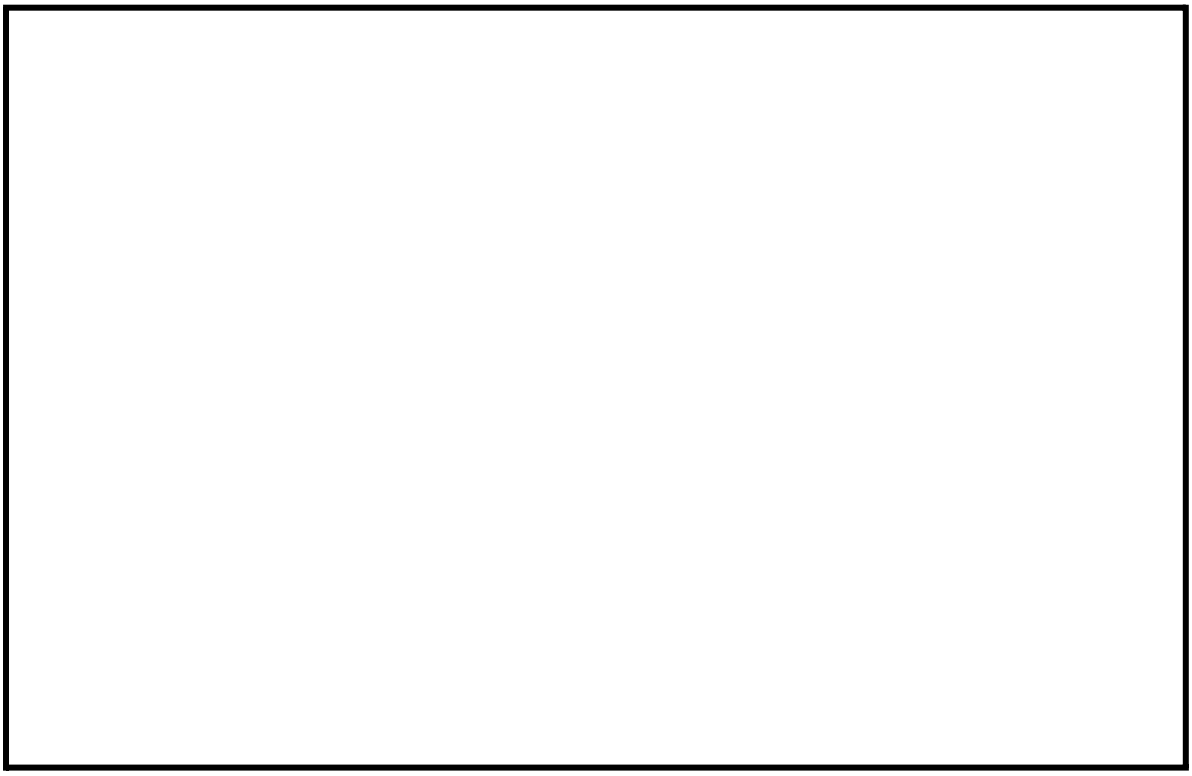
Step11.InstallMetaMaskonyourbrowser.

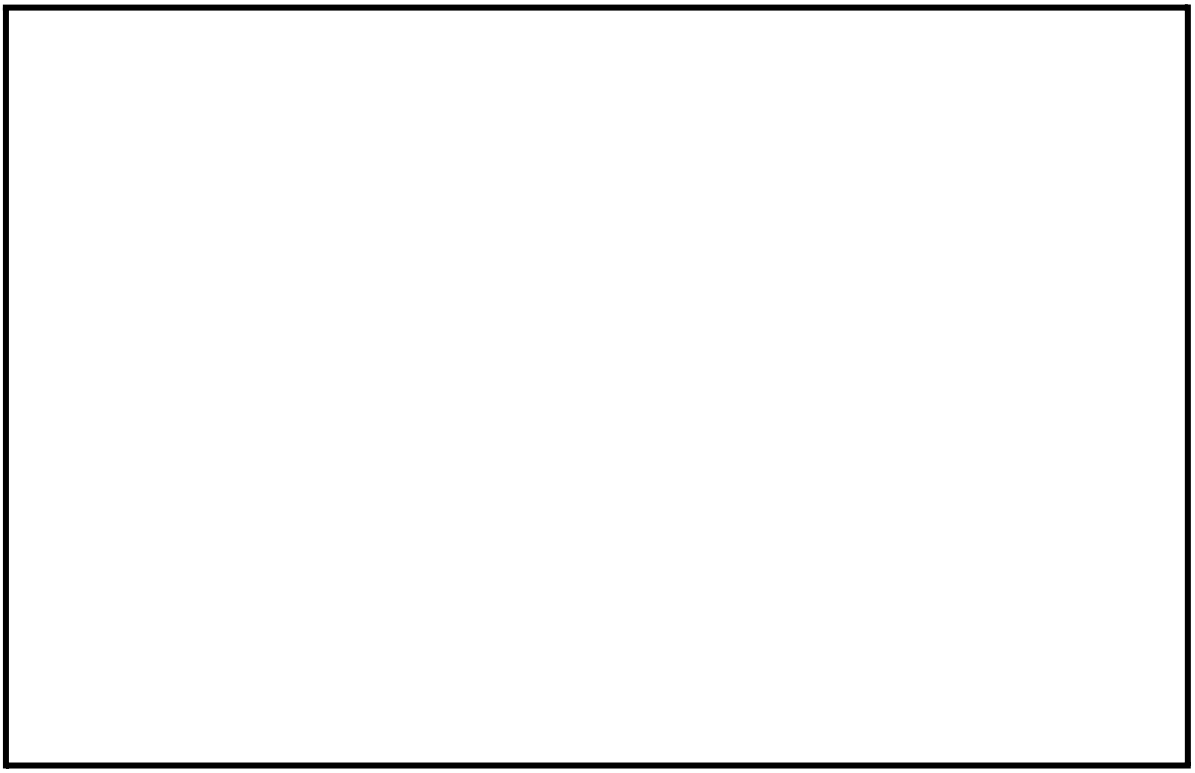
To create a new wallet, you have to install the extension first. Depending on your browser, there are different marketplaces to find it. Most browsers have MetaMask on their stores, soit'snotthathardtoseeit, buteitherway,herethey areChrome,Firefox,andOpera.





- Click on **Install MetaMask**as a Google Chrome extension.
- Click**AddtoChrome**.
- Click **Add Extension**.



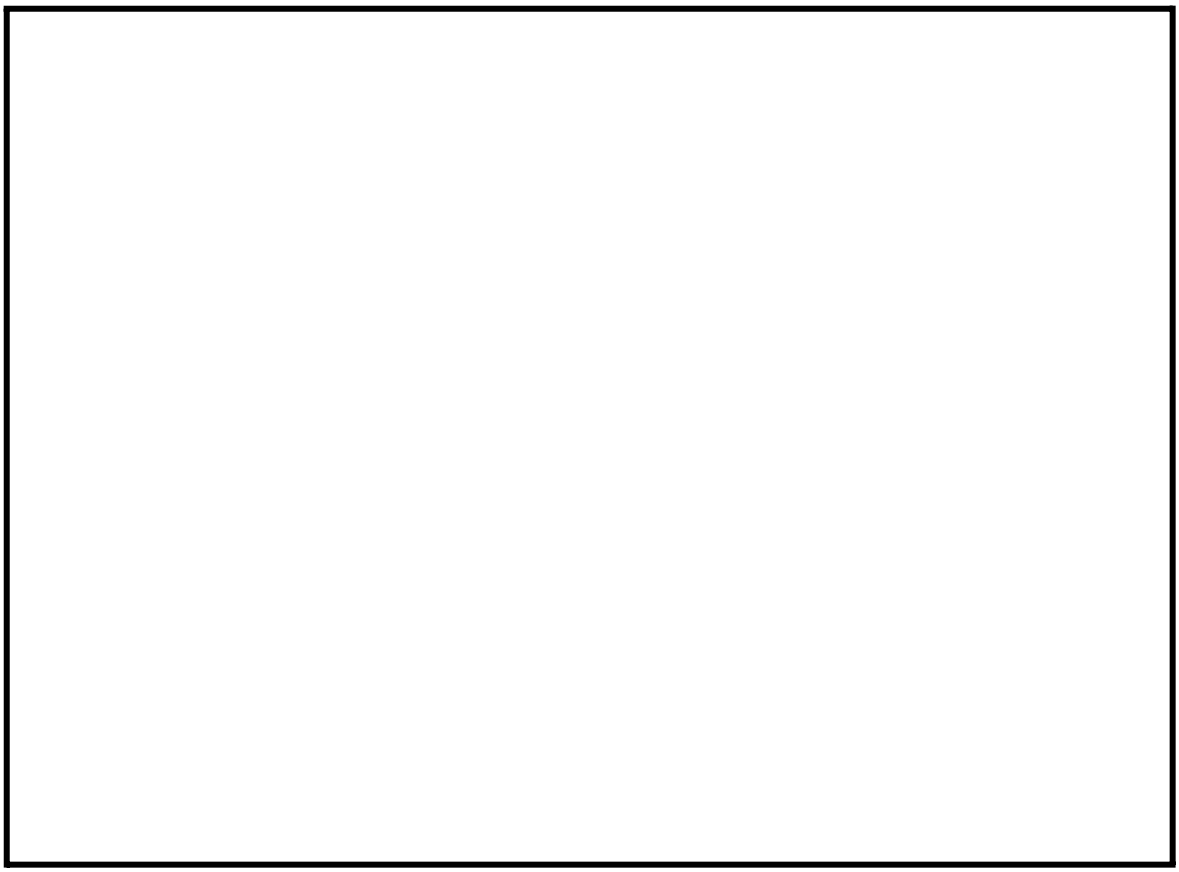


aseasyasthattoinstalltheextensiononyourbrowser,continuereadingthenextstepto figure out how to create an account.

Step12.Createanaccount.

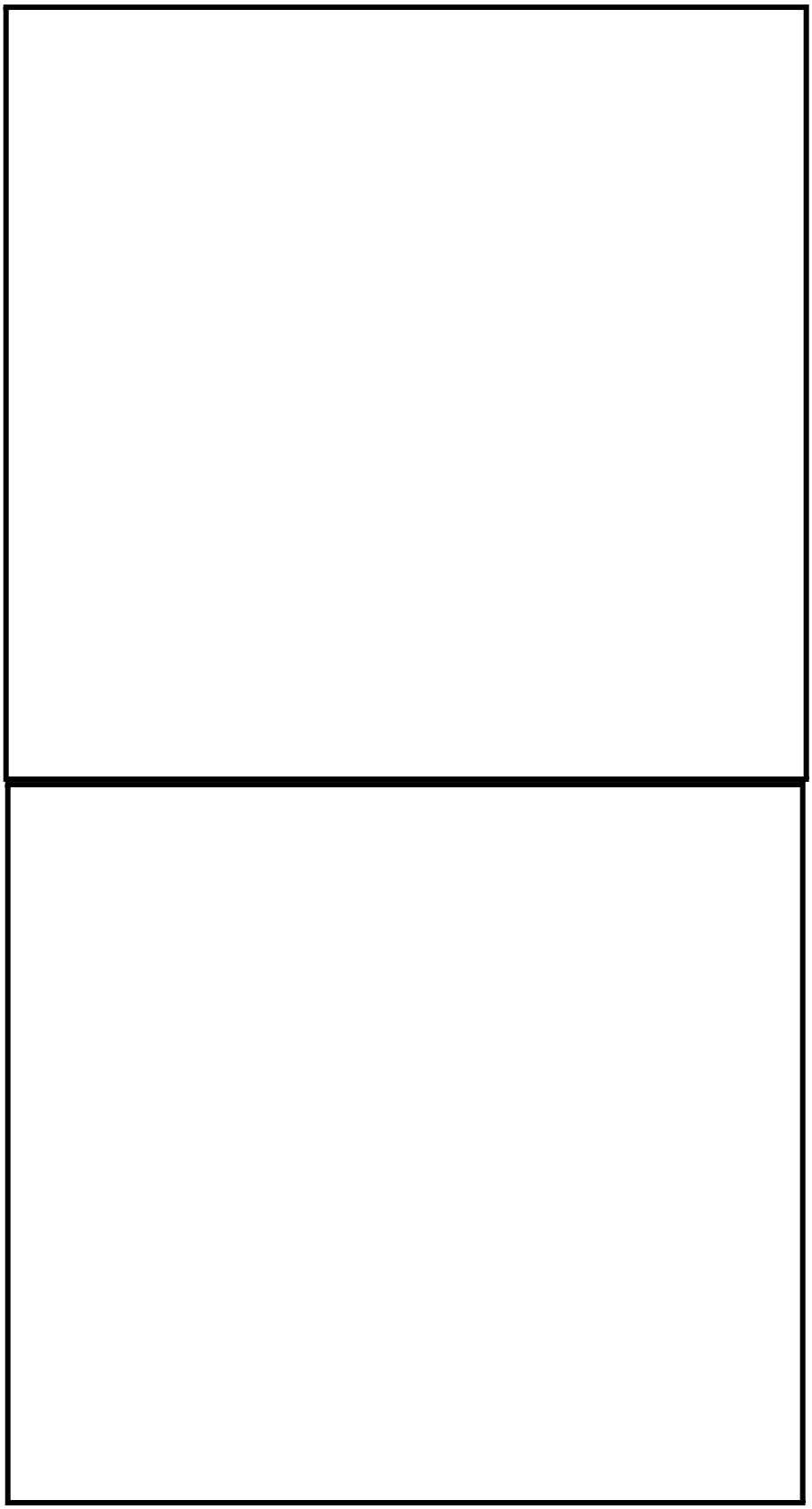
- Click on the extension icon in the upper right corner to open MetaMask.

- Toinstallthelatestversionandbeuptodate, **clickTryitnow**.
- **Click Continue**.
- Youwillbepromptedtocreateanewpassword.**ClickCreate**.



- Proceedby**clickingNext**andacceptttheTermsofUse.

ClickRevealSecretWords.Thereyouwillseea12wordsseedphrase.Thisisreallyimportantandus uallynotagood ideatostoredigitally,sotake yourtimeandwriteitdown



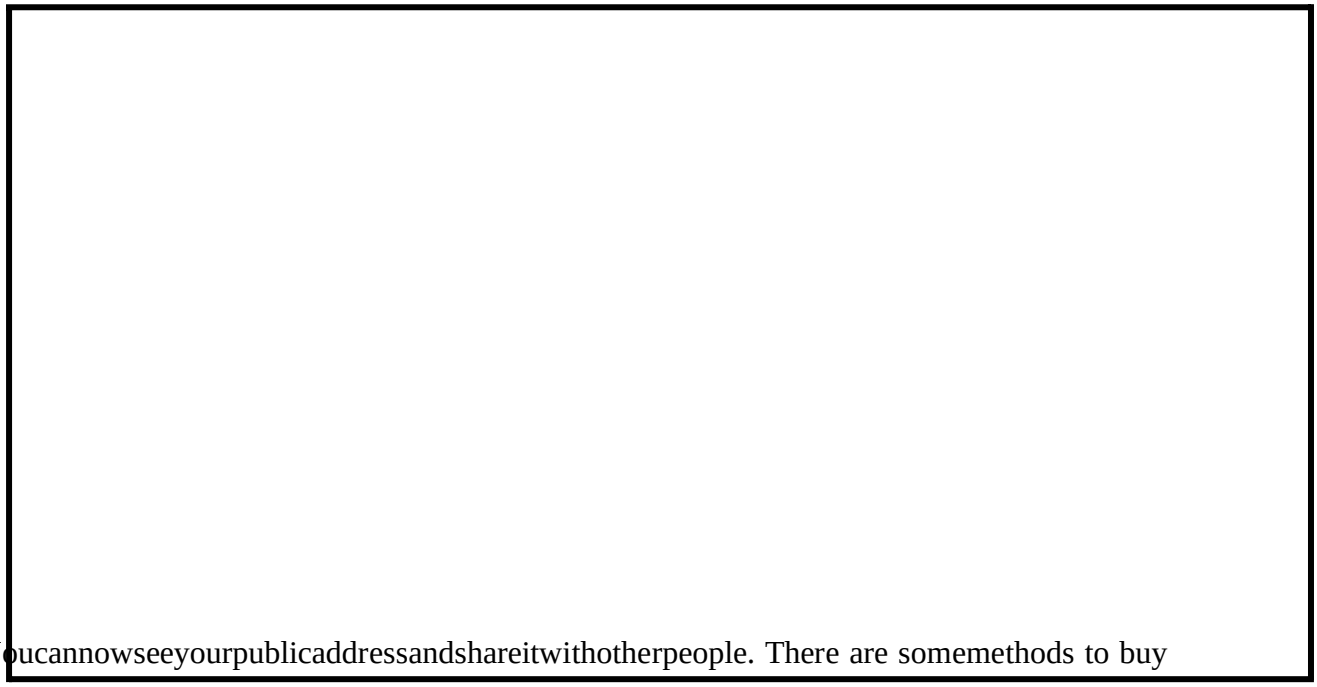
- Verifyyoursecretphrasebyselectingthepreviouslygeneratedphraseinorder.**ClickConfirm**.

Andthat’sit; now youhavecreatedyourMetaMaskaccountsuccessfully.AnewEthereumwallet

Addresshasjustbeencreatedforyou.It’swaitingforyoutodepositfunds,andifyouwanttolearnhowto do that, look at thenext step below.

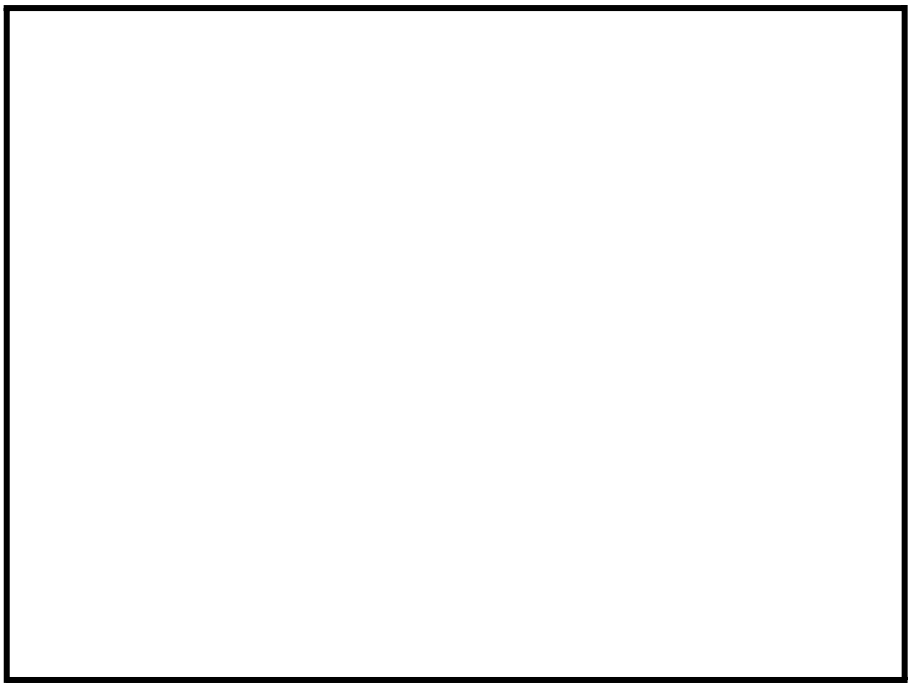
Step13.Depositingfunds.

- Click on**ViewAccount**.



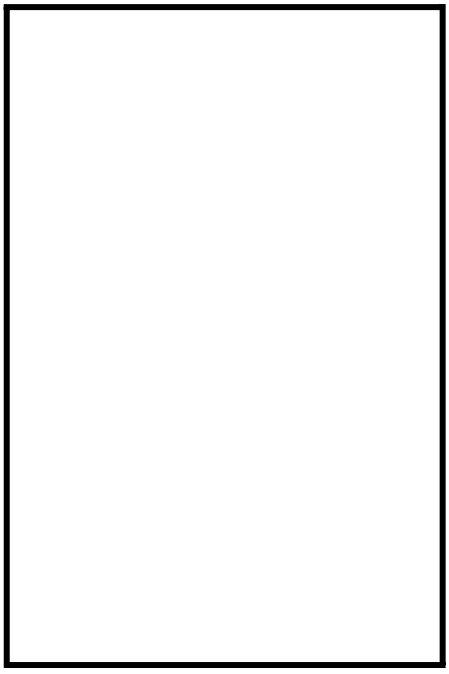
Youcannowseeyourpublicaddressandshareitwithotherpeople. There are somemethods to buy coins offered by MetaMask, but you can do it differently as well; you justneed your address.

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



Youcannowaccessyourlistofassetsinthe‘Assets’tabandviewyourtransactionhistoryinthe‘Activity’tab.

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



- To use MetaMask to interact with a dapp or smart contract, you'll usually need to find a'ConnecttoWallet'buttonorsimilarelement on theplatform you aretrying to use.Afterclicking this, you should then see a prompt asking whether you want to let the dapp
  Connect to your wallet.

What advantages does MetaMask have?

- **Popular** - It is commonly used, so users only need one plugin to access a wide range ofdapps.
- **Simple** - Instead of managing private keys, users just need to remember a list of words, andtransactions are signed on their behalf.
- **Savesspace**- Users don'thavetodownloadtheEthereumblockchain,asMetaMasksendsrequeststo nodes outsideof the user's computer.
- **Integrated** - Dapps are designed to work with MetaMask, so it becomes much easier to sendEther in and out.

Conclusion- InthiswaywehaveexploredConceptBlockchainandmetamaskwalletfortransaction of digital currency.

Referencelink

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

Assignment No. 12

12.1 Title

Assignment based on Metamask

12.2 Problem Definition:

Create your own wallet using Metamask for crypto transactions.

12.3 Software Requirements:

Mozilla Firefox

12.4 Learning Objectives:

Learn Metamaskwalletfor crypto transactions

12.5 Outcomes:

After completion of this assignment students are able to understand the How to create metamask wallet.

12.6 Theory Concepts:

MetaMask is a plug-in Ethereum crypto wallet for Chrome onboard users. Available as a browserextension and as a mobile app, MetaMask equips us with a key vault, secure login, and tokenwallet—everything we need to manage our digital assets. MetaMask provides the simplest yetmost secure way to connect to blockchain-based applications.

Available as a browser extension and as a mobile app, MetaMask equips you with a key vault, secure login, token wallet, and token exchange—everything you need to manage your digital assets.

MetaMask provides the simplest yet most secure way to connect to blockchain-based applications. You are always in control when interacting on the new decentralized web.

MetaMask generates passwords and keys on your device, so only you have access to your accounts and data. You always choose what to share and what to keep private.

MetaMask provides an essential utility for blockchain newcomers, token traders, crypto gamers, and developers.

12.6.1 Features

Swap: Swap tokens directly from your desktop or mobile wallet. The Swaps feature combines data from

decentralized exchange aggregators, market makers, and DEXs, to ensure you get the very best price with the lowest network fees.

Swaps ensures that you always have access to the largest selection of tokens and the most competitive prices, by providing prices from multiple aggregators and individual market makers in one place.

Gas fee: A **gas fee is a blockchain transaction fee**, paid to network validators for their services to the blockchain. After EIP-1559 launched, we've been observing the market and adapting our estimations and UI to account for the dramatic price spikes that still exist on the EthereumMainnet due to popular NFT drops and other projects.

- Low" (previously "Low"): is much lower than market prices and it allows a user to pay a lower fee when they are willing to wait a longer time. It allows you to wait a longer period and skip the price spikes (i.e. save money). Note that this setting is based on past trends, which means we can never be sure the transaction goes through. If you require a transaction to go through, this may not be the right setting for you.

- Market"(previously "Medium"): reflects market prices.

- "Aggressive"(previously "High"): is much higher compared to market prices. It allows you to set a really high max fee and priority fee to increase the likelihood of your transaction being successful if you're expecting to participate in a gas war.

Buy crypto: Buying cryptocurrency with MetaMask has never been easier. Just head on over to the "Buy" button to get started. The improved user experience shows you the best quotes based on the token you want to buy, your region, and the provider that is available to you.With a funded MetaMask wallet, you're ready to explore Web3 on your own terms.

12.6.2 MetaMask Setup

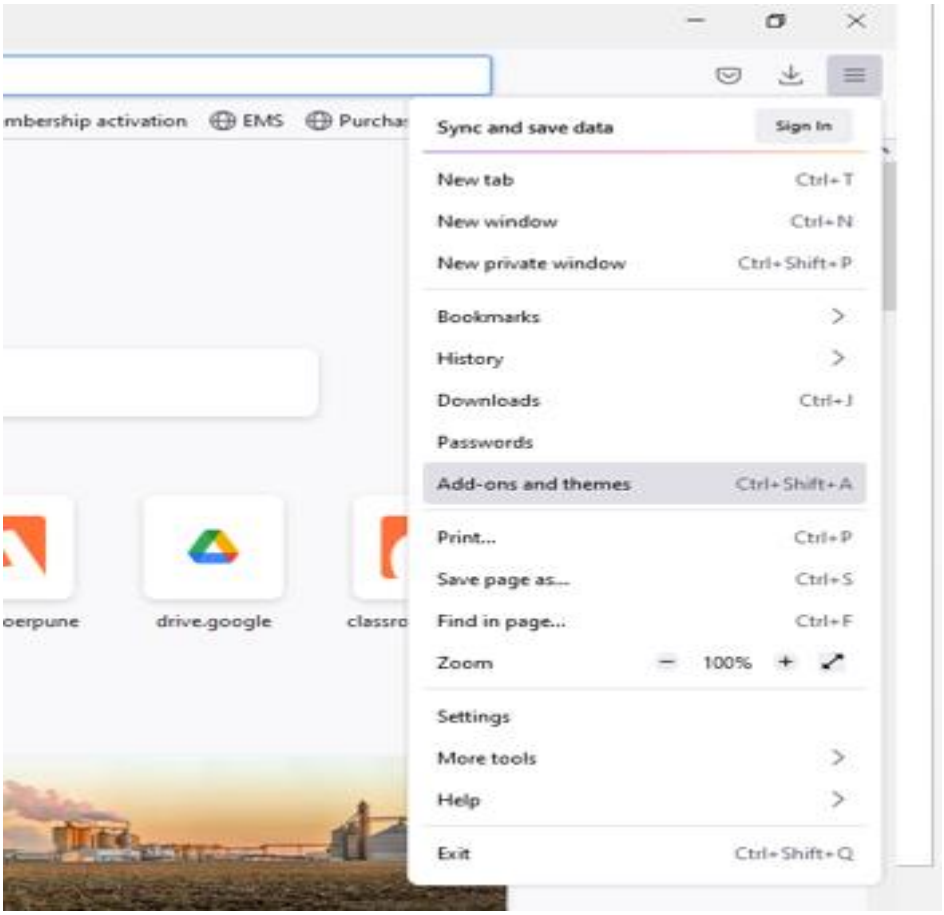
Complete information and study guide about MetaMask can be found at its official website metamask.io. We need to choose the right browser (Chrome is recommended) and follow its installation instruction. When we are creating a new MetaMask account, here are some key pointswe need to pay attention to.First of all, creating a new strong password is extremely important because it encrypts private key.

12.6.3 Make A Transaction

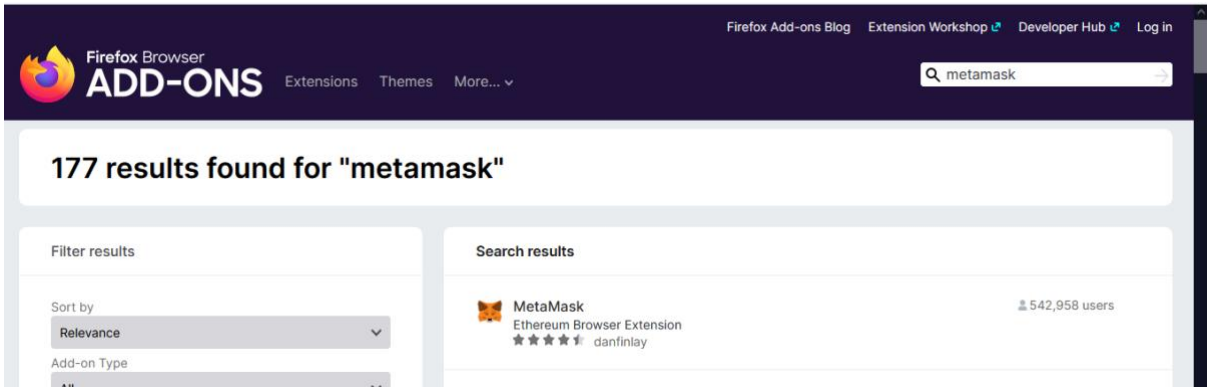
We are going to make a transaction between our accounts. We only have one default account right now, but we definitely need to create more accounts.By clicking the top-right account picture in MetaMask interface, we can see 'Create Account'button. By clicking that button and entering the account name, we will switch back to the accountwhere we deposited Ether, click 'Send', enter an amount of Ether, select an account we want totransfer to and choose the speed to send Ether. Depending on the speed we choose, thetransactionusually takes about 15 - 30 seconds.While we are waiting for it to be completed, we can find this transaction in 'Queue' (or in 'History'if the transaction is finished). By clicking 'View on Etherscan', we will find transaction detailsincluding Transaction Hash, Status, Block, Timestamp, From, To, Transaction Value andTransaction Fee.

12.7Implementation & Output screenshot

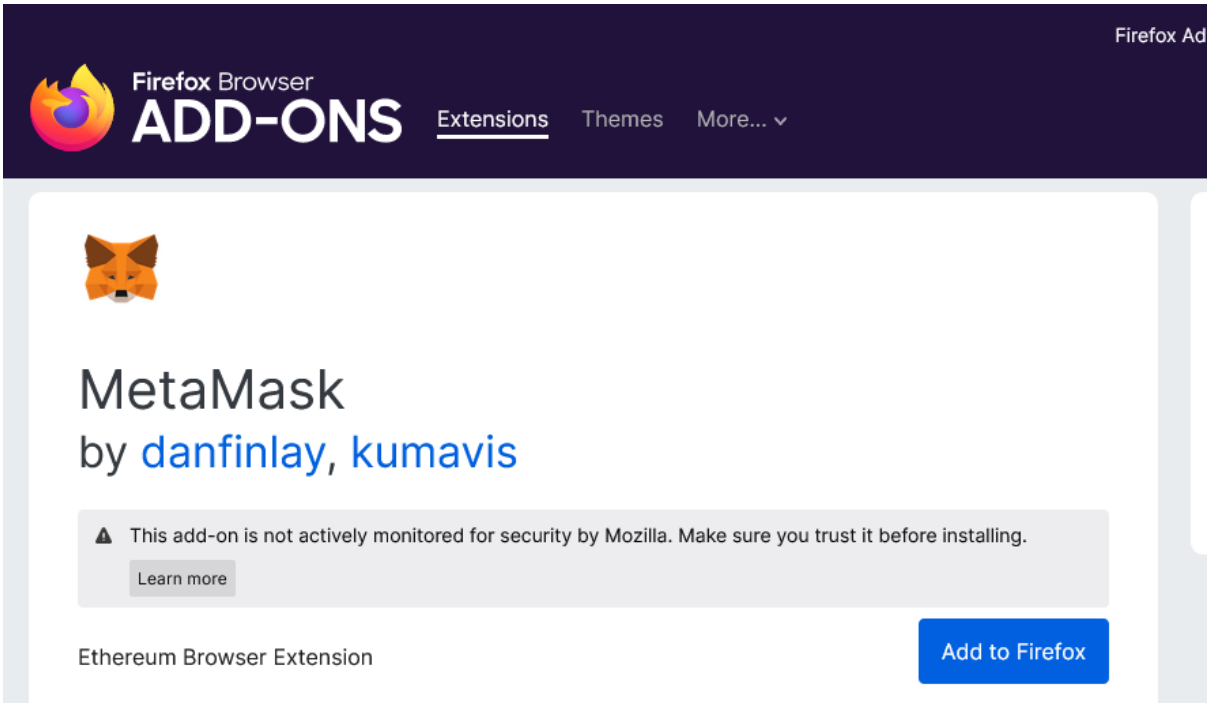
Step 1:- Open Mozilla Firefox click on application menu then click on add-ons



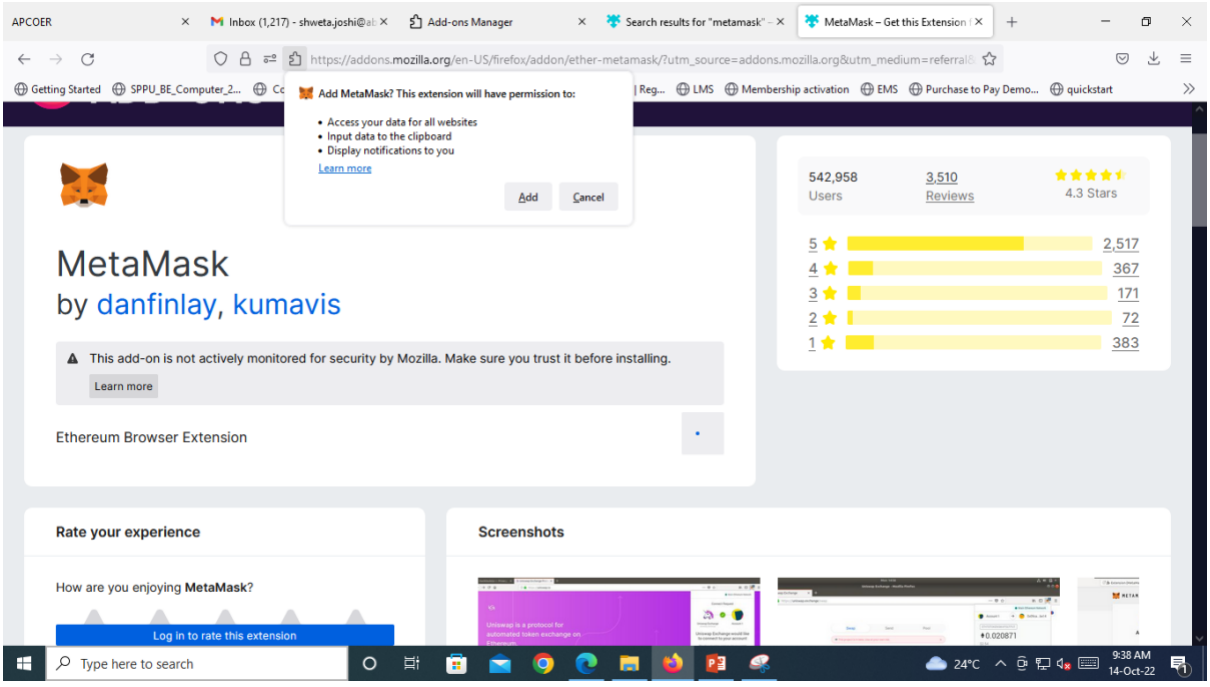
Step 2:- Type metamask& select metamask browser extension

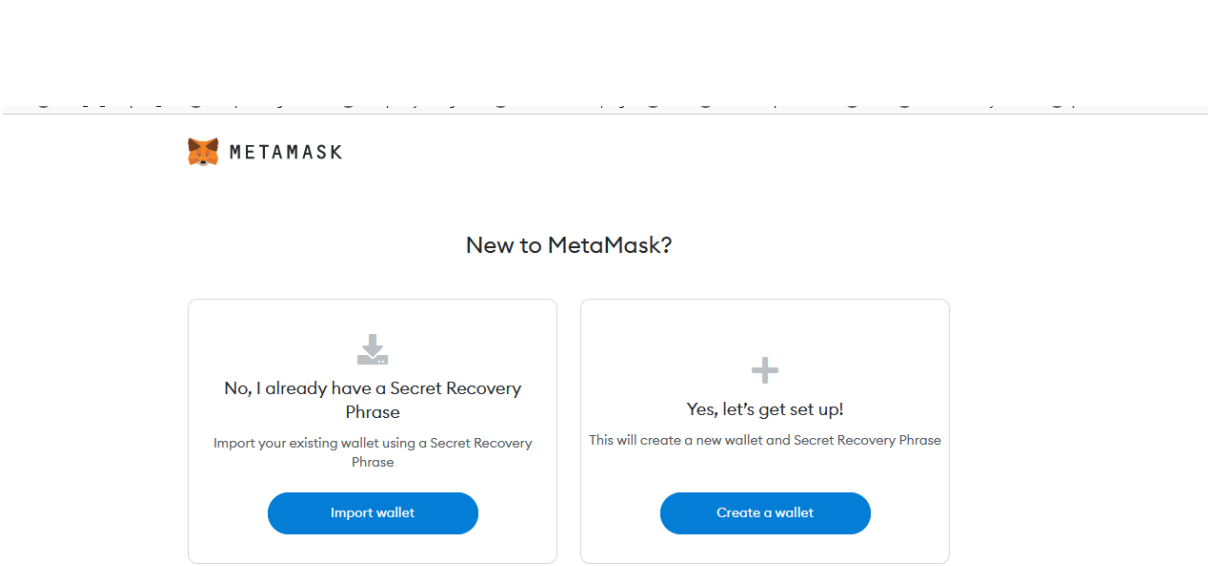


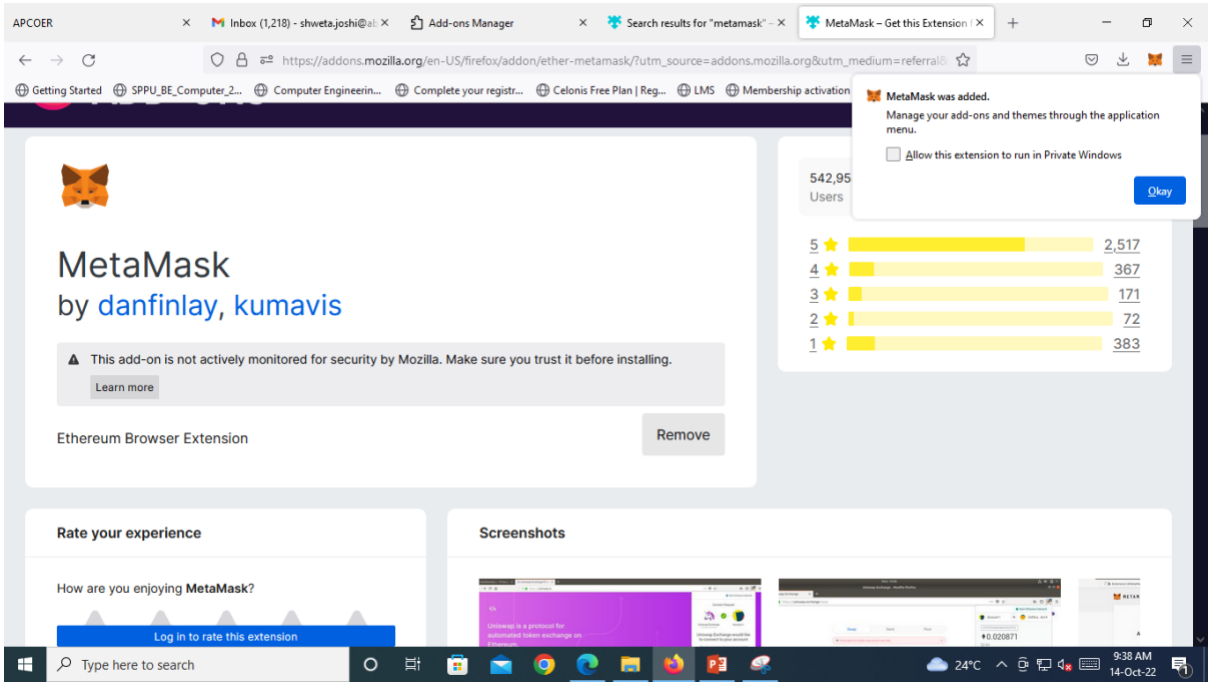
Step 3:- Click on add to Firefox



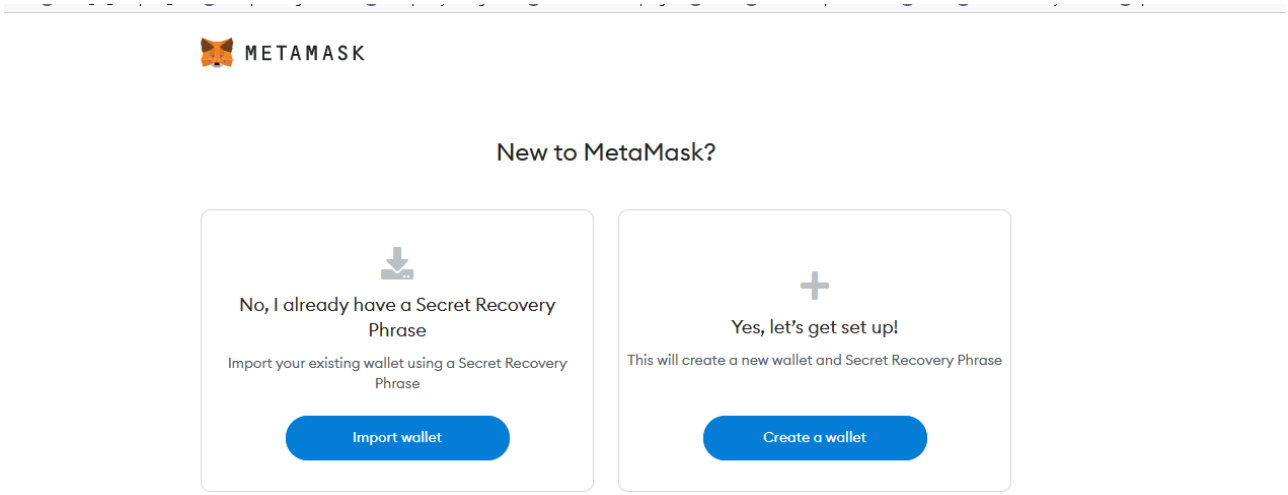
Step 4:- Click on add & ok



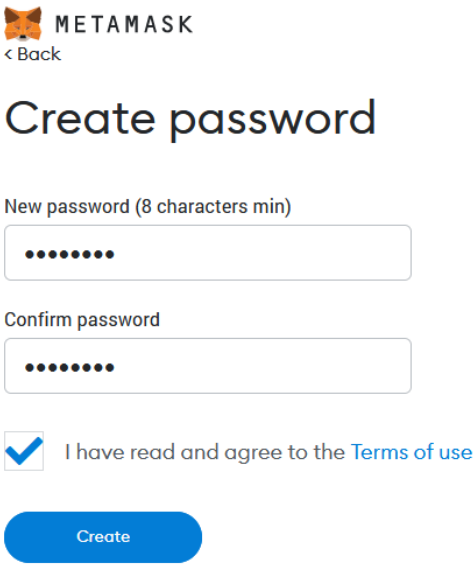

METAMASK
New to MetaMask?
No, I already have a Secret Recovery Phrase
Import your existing wallet using a Secret Recovery Phrase
Import wallet
Yes, let's get set up!
This will create a new wallet and Secret Recovery Phrase
Create a wallet

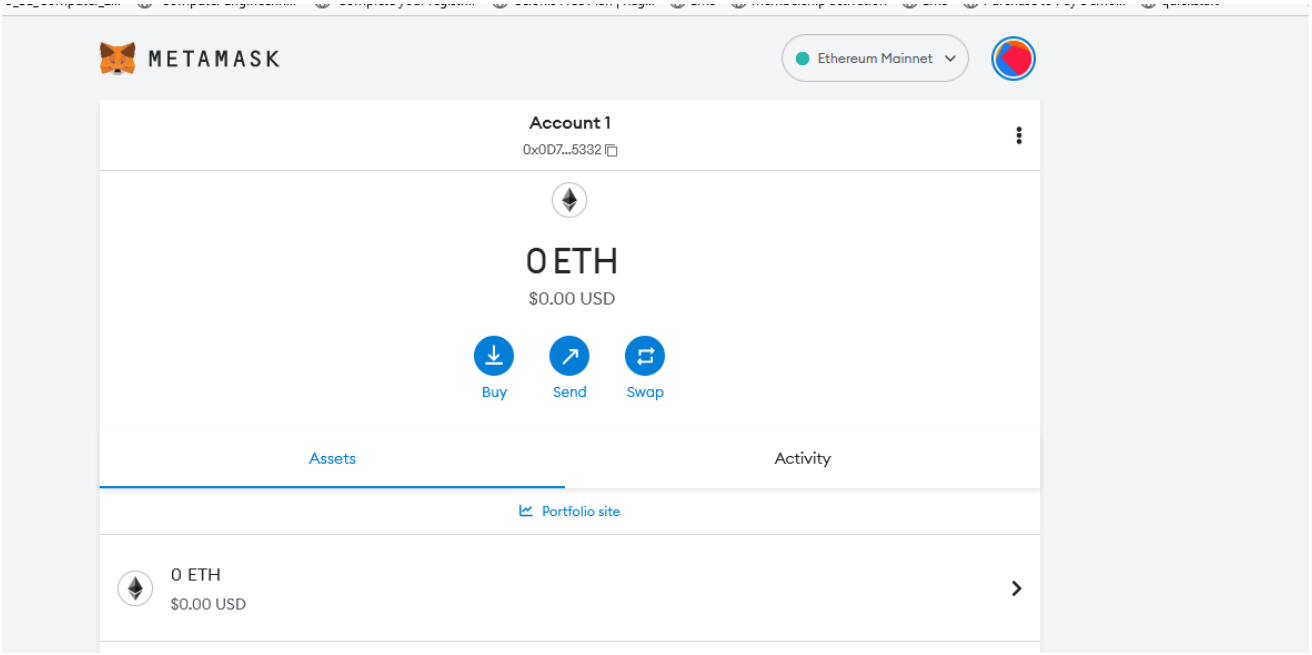


Step 5:- Create wallet



Step 6:- after create wallet, create password





12.8 Conclusion

Thus we learn how to create metamask wallet.

References:-

1. https://metamask.io/
2. MetamaskYoutube Channel (https://www.youtube.com/watch?v=GNPz-Dv5BjM)

Assignment No. 13

13.1 Title

Assignment based on Smart contract

13.2 Problem Definition:

Write a smart contract on a test network, for Bank account of a customer for following operations:
· `Deposit money`
· `Withdraw Money`
· `Show balance`

13.3 Prerequisite:

Solidity language

13.4 Software Requirements:

Remix IDE

13.5 Learning Objectives:

Learn How to write smart contract.

13.6 Outcomes:

After completion of this assignment students are able write smart contract for bank account.

13.7 Theory Concepts:

First of all, we need to understand the differences between a paper contract and a smart contractand the

reason why smart contracts become increasingly popular and important in recent years. Acontract, by definition, is a written or spoken (mostly written) law-enforced agreement containingthe rights and duties of the parties. Because most of business contracts are complicated and tricky,the parties need to hire professional agents or lawyers for protecting their own rights. However, ifwe hire those professionals every time we sign contracts, it is going to be extremely costly andinefficient. Smart contracts perfectly solve this by working on 'If-Then' principle and also asescrow services. All participants need to put their money, ownership right or other tradable assetsinto smart contracts before any successful transaction. As long as all participating parties meet therequirement, smart contracts will simultaneously distribute stored assets to recipients and thedistribution process will be witnessed and verified by the nodes on Ethereum network.There are a couple of languages we can use to program smart contract. Solidity, an object-orientedand high-level language, is by far the most famous and well maintained one. We can use Solidityto create various smart contracts which can be used in different scenarios, including voting, blindauctions and safe remote purchase. In this lab, we will discuss the semantics and syntax of Solidity with specific explanation, examples and practices. If you want to find more information aboutSolidity, please check its official website at:solidity.readthedocs.io.After deciding the coding language, we need to pick an appropriate compiler. Among variouscompilers like Visual Code Studio, we will use Remix IDE in this and following labs because itcan be directly accessed from browser where we can test, debug and deploy smart contracts withoutany installation. Remix can be reached at its official website: http://remix.ethereum.org/.

By clicking 'File Explorers' at the left side of Remix interface, we can see the list of smart contracts(or empty list) under 'Browser'. Clicking '+' next to 'Browser', we will start to compile our firstcontract by entering its name 'MyFirstContract.sol' (all solidity files need to add '.sol' in the endof the file name). Before compiling smart contracts, we need to choose the right version of compiler. Full list ofSolidity versions displays under 'Compiler' in 'Solidity Compiler' tab. We are going to switch toany version starting with '0.6.x' (we mainly use 0.6.10+commit.00c0fcaf in this and the followinglabs). Because Solidity made some breaking changes in every big version update, if we use version0.7.x, 0.15.x, 0.14.x or even lower versions, Remix might show some errors on our codes.
The sample of our first smart contract 'MyFirstContract' is in 'LAB 2 Assignment.txt'. You needto copy and paste this contract into Solidity file which is created earlier. The lab also provides thetutorial video 'My First Contract.mov'. You need to check if the deployment process of yourcontract is similar with the video.

13.7.1 Smart contract

A *smart contract* is a computer program or a transaction protocol that is intended to automatically execute, control, or document legally relevant events and actions according to the terms of a contract or an agreement. The main contribution of smart contract is making the blockchain applications programmable, and it brought the blockchain applications beyond just transfer currency. It makes the verification of the terms of any agreement automatic. Thus all operations, which depend on a condition, can be figured out without any middleman or third party's intervention.
Smart contracts provide the following benefits:

- **Transparency:** Since smart contracts are deployed to the Ethereum network like a transaction after compile, these contracts are fully accessible and visible to all the relevant parties.
- **Security:** The conditions and environments of where to keep smart contracts are a secure place, much as cryptocurrencies have.
- **Trust:** By keeping smart contracts in a place that is as secure as cryptocurrencies, the secure, autonomous, and transparent nature of blockchain can be provided for smart contracts too. Knowing

that the possibility of making manipulations on information placed in a block is very low helps us to trust.

- **Speed:** Smart contracts live on the internet and run on software code. Thus it verifies transactions very fast. This speed can save many wasted hours caused by humans.
- **Saving:** Smart contracts eliminate the need for having a middleman or third-party authority to determine what to do according to the terms of the agreement. Thus, they save wasted money besides wasted time, too. That means there is no need for lawyers, banks, witnesses, and any other intermediaries.

13.7.2 Solidity Language Description

Solidity is an object-oriented, high-level language for implementing smart contracts, and it is designed to target the Ethereum Virtual Machine (EVM), managing the behavior of accounts within the Ethereum state. It is a contract-oriented language, which means that smart contracts are responsible for storing all of the programming logic that transacts with the blockchain.
With Solidity, any contract can be developed for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.
Among several methods (i.e., solc-js**,** a Node.js framework, or up-to-date Docker builds) to compile the Solidity code and compile it, Remix IDEis a powerful, open source tool that helps you write Solidity contracts straight from the browser without installing anything. Written in JavaScript, Remix supports both usage in the browser and locally. Remix also supports testing, debugging, and deploying smart contracts, and much more.
The following elements that of a smart contract are good to understand the general structure of a smart contract.

Declaring a contract

The Solidity version and the contract name must be declared by the following.

```
pragma solidity ^0.14.24;contract MyContract {
   // do something
}
```

Importing other source files

Ethereum Solidity supports import statements that are very similar to those available in JavaScript, although Solidity does not know the concept of a “default export.”

```
import "filename";----- OR -----import * as symbolName from "filename";
```

Comments

Just like any other language, single-line and multi-line comments are possible in Solidity.

```
// This is a single-line comment.----- OR -----/*
  This is a
  multi-line comment
*/
```

Variables

Solidity supports the following variables:

- **state variables** —permanently stored in contract storage
- **local variables** — presented till the function is executing
- **global variables** —special variables that exist in the global namespace and are used to get information about the blockchain

```
pragma solidity …contract HelloWorld{
uintstoredData;     // state variable
```

```
function getResult() public view returns(uint){
uint a = 1;     // local variable
}
}
```

Variable scope

Solidity supports the following types of visibilities that are common for both functions and variables

- **public** —public functions or variables that can be called internally or through messages
- **internal** — functions and variables only available to the current contract and derived contracts
- **private** — only available to the current contract and not derived contracts
- **external** — can just be used to functions to be called from other contracts and transactions

Data types

The following types are also called *value types* because variables of these types will always be passed by value.

- **boolean**— `bool`; the possible values are constants, i.e., true or false
- **integer** —`int/uint`; signed and unsigned integers of various sizes
- **string** —`string`; string literals are written with either double or single-quotes "foo" or 'bar'
- **address** — `address/address payable` holds a 20-byte value, the size of an Ethereum address, etc.

Address types

Address types also have members and serve as a base for all contracts. There are two types of address: `address` holds a 20-byte value, and `address payable` is the same as `address` above but just has `transfer` and `send` members. The distinction between `address`/`address payable` is that `address payable`is an address you can send ether to, while a plain `address`cannot be sent Ether.
Implicit conversions from `address payable` to `address` are allowed, whereas conversions from `address` to `address payable` are not allowed. Address type has several members:

- `balance` returns the balance of given address (in units of wei)
- `transfer` sends a given amount to a given address (in units of wei)
- `send` sends a given amount of wei to a given address (not a safe alternative for transfer)

Data structures

The following types are specialized in Solidity:

- **Struct**— `struct` Solidity provides a way to define new types in the form of structs. Structs are custom-defined types that can group several variables.
- **Array** — `[]` Arrays can have a compile-time fixed size, or they can have a dynamic size.
- **Mapping** — `mapping(k => v)` Mappings can be seen as hash tables which are virtually initialized such that every possible key exists and is mapped to a value whose byte-representation is all zeros: a type's default value, etc.

Data locations

All reference types contain information on where they are stored, and there are three areas to store them:

- **Storage** —State variable, local variables of struct, array, and mapping: Each account has its own storage, which is persistent between function calls and transactions.
- **Memory** —Function arguments, which a contract obtains a freshly cleared instance for each message call. It holds temporary values and gets erased between (external) function calls and is cheaper to use.
- **Stack** — Small local variables. The EVM is not a register machine but a stack machine, so all computations are performed on a data area called the *stack*. It holds small local variables and is almost free to use, but can only hold a limited amount of values.

Functions

In a function, `<parameter types>` can be empty, but `<return types>` must be filled.

- **Internal functions** can only be called inside the current contract.
- **External functions** consist of an address and a function signature, and they can be passed and returned from an external function.

function (<parameter types>) {internal|external} [pure|view|payable] [returns (<return types>)]

Type of functions

`pure` ensures that they do not change the state (it can be converted to view).
`view` does not read from state or modify the state.
`payable` makes it so ether can be sent in the function.
`fallback` does not have any arguments and does not return anything, but it is especially used for sending Ether, writing to storage, and creating a contract.
Functions can be set as `view` and `pure` to restrict reading and modifying of the state.

Method overloading

Function overloading occurs when several functions in a contract have the same name but differing arguments.

```
function sayHi(string n1) public pure returns (string out) {
  out = "Hi " + n1 + " !" ;
}function sayHi(string n1, string n2) public pure returns (string out) {
  out = "Hi " + n1 + " and " + n2 + " !" ;
}
```

Modifiers

Modifiers are defined to change the way functions work in Solidity. By using modifiers, the condition, which is checked before the function execution, can be set.

```
pragma solidity ^0.14.24;contract MyContract {
  constructor() public { owner = msg.sender; }
  address payable owner;   modifier onlyOwner {
    require(msg.sender == owner, "Only owner can call this function.");
    _;    // When the owner calls this function, it executes. Otherwise, it throws an exception.
}   function close() public onlyOwner {
    // do something
  }
}
```

13.8Implementation

We can start to develop a smart contract that includes all of the functionalities and capabilities Solidity presents, or at least as much as we can. There are several methods to developing a smart contract. Of these methods, we'll use Remix IDE, a powerful open-source tool that provides the ability to develop a smart contract from a browser.

First, let's open Remix IDE from a browser and create a new file with the extension `.sol`. We'll use this for our Solidity code here.

Declaring the Contract

We're supposed to declare a contract beside the Solidity version we want to use.

```
pragma solidity ^0.6.6;contract BankContract { }
```

Define the State Variables, Data Types, and Data Structures

We create a client object to keep the client's information, which will join the contract by using the struct element. It keeps the client's ID, address, and balance in the contract. Then we create an array in the client_account type to keep the information of all of our clients.

```
pragma solidity ^0.6.6;contract BankContract {
structclient_account{
intclient_id;
      address client_address;
uintclient_balance_in_ether;
   }    client_account[] clients;
}
```

We're supposed to assign an ID to each client whenever they join the contract, so we define an int counter and set it to 0 in the constructor of the contract.

```
pragma solidity ^0.6.6;contract BankContract {
structclient_account{
intclient_id;
      address client_address;
uintclient_balance_in_ether;
   }    client_account[] clients;    int clientCounter;
   constructor() public{
clientCounter = 0;
   }
}
```

Also, we need to define an `address` variable for the manager and a `mapping` to keep the last interest date of each client.

```
address payable manager;
    mapping(address =>uint) public interestDate;
```

Both methods will check the people who call the relevant method and which of the modifiers is used. One of them

determines whether the sender is the manager, and the other one determines whether the sender is a client.

```
modifier onlyManager() {
        require(msg.sender == manager, "Only manager can call!");
        _;
    }

    modifier onlyClients() {
        bool isClient = false;
        for(uinti=0;i<clients.length;i++){
            if(clients[i].client_address == msg.sender){
isClient = true;
                break;
            }
        }
        require(isclient, "Only clients can call!");
        _;
    }
}
```

Receive ether from the clients as a deposit,

```
receive() external payable { }
}
```

Develop the Methods

The `setManager` method will be used to set the manager address to variables we've defined. The `managerAddress` is consumed as a parameter and cast as payable to provide sending ether.

```
functionsetManager(address managerAddress) public returns(string memory){
     manager = payable(managerAddress);
     return "";
}
```

The `joinAsClient` method will be used to make sure the client joins the contract. Whenever a client joins the contact, their interest date will be set, and the client information will be added to the `client` array.

```
functionjoinAsClient() public payable returns(string memory){
interestDate[msg.sender] = now;
clients.push(client_account(clientCounter++, msg.sender, address(msg.sender).balance));
     return "";
}
```

The `deposit` method will be used to send ETH from the client account to the contract. We want this method to be callable only by clients who've joined the contract, so the `onlyClient` modifier is used for this restriction

```
function deposit() public payable onlyClients{
     payable(address(this)).transfer(msg.value);
}
```

The `withdraw` method will be used to send ETH from the contract to the client account.

```
function withdraw(uint amount) public payable onlyClients{
msg.sender.transfer(amount * 1 ether);
}
```

The `sendInterest` method will be used to send ETH as interest from the contract to all clients. We want this method to be callable only by the manager, so the `onlyManager` modifier is used for this restriction.

```
function sendInterest() public payable onlyManager{
     for(uinti=0;i<clients.length;i++){
          address initialAddress = clients[i].client_address;
uintlastInterestDate = interestDate[initialAddress];
          if(now <lastInterestDate + 10 seconds){
```

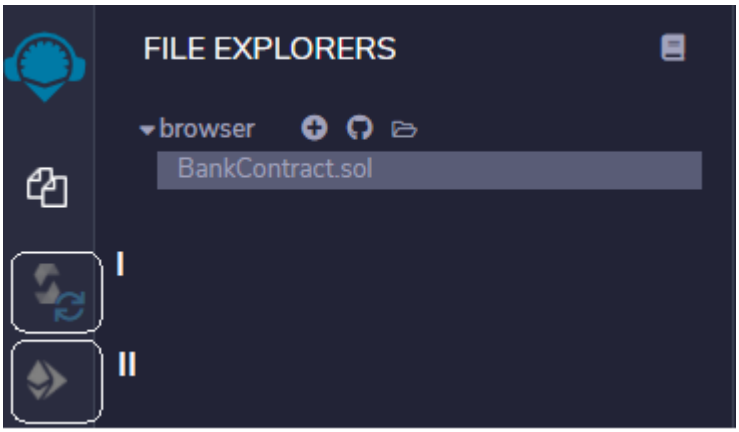
```
                revert("It's just been less than 10 seconds!");
            }
            payable(initialAddress).transfer(1 ether);
interestDate[initialAddress] = now;
        }
}
```

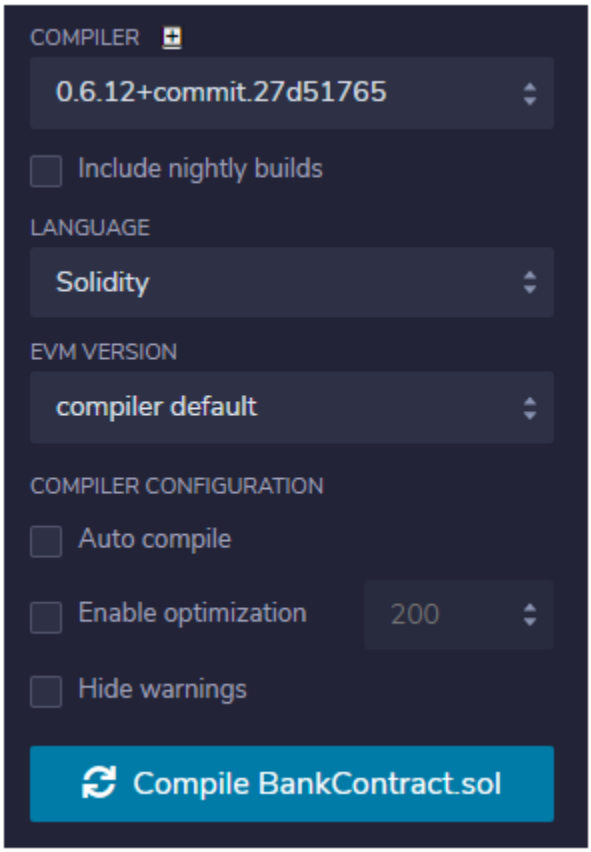
The `getContractBalance` method will be used to get the balance of the contract we deployed.

```
functiongetContractBalance() public view returns(uint){
     return address(this).balance;
}
```

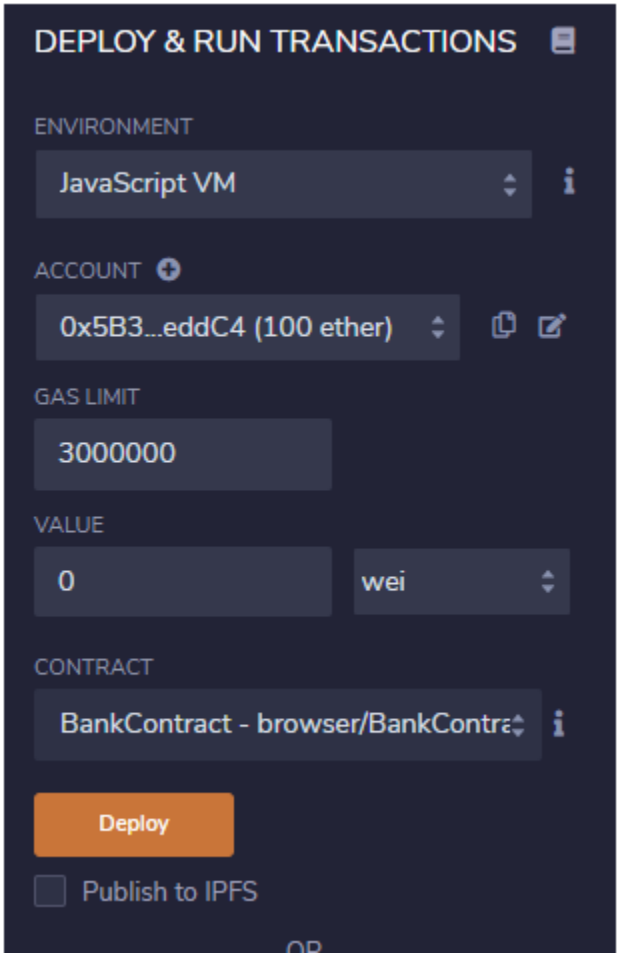
Compile the smart contract

After finishing the development of the smart contract, we'll compile it onto the Remix IDE.





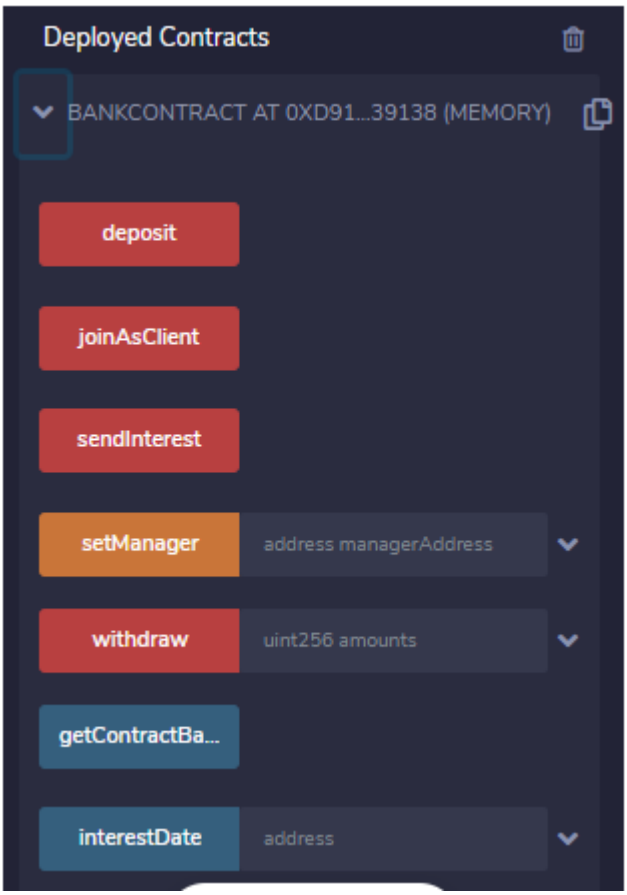
Once we've compiled the smart contract without any problems, we're ready to continue with deploying the smart contract and running the transactions.



we'll deploy our contract on the JavaScript VM environment, so we'll select it among the following environments.

Run the Transactions

Now, we're ready to call the functions that compound the smart contract developed. When we expand the relevant contract in the Deployed Contract subsection, the methods developed appear.



13.9 Conclusion:-we have successfully developed a smart contract that allows for the implementation of all of the elements that compound Solidity as much as we can. Then we compiled and deployed it on the Remix IDE. Finally, we tested all of the functionalities it presents through the methods it has by simulating a process.

References:-https://betterprogramming.pub/developing-a-smart-contract-by-using-remix-ide-81ff6f44ba2f

Assignment No:-14

14.1 Title

Assignment based on write a survey report

14.2 Problem Definition:

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

14.3 Learning Objectives:

Learn How to write survey report

14.4 Outcomes:

After completion of this assignment students are able write survey report on types of Blockchains and its real time use cases.

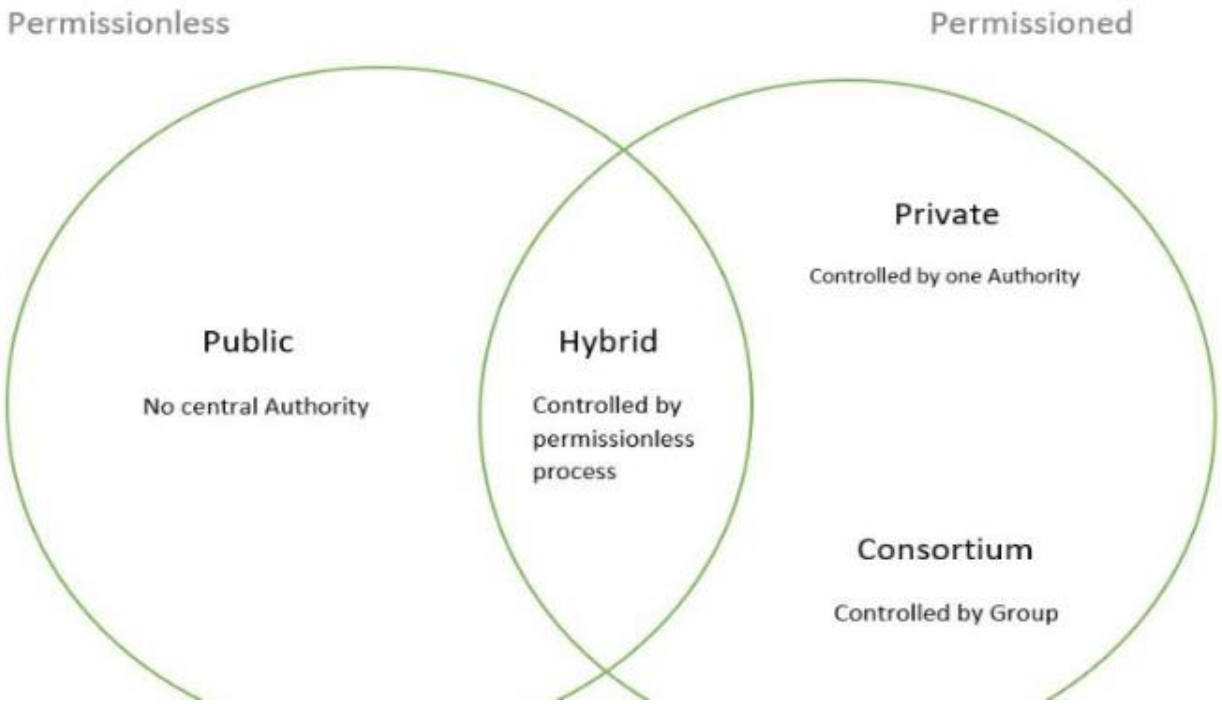
14.5 Theory Concepts:

Blockchain has gained massive popularity in the past decade, thanks to Bitcoin. However, Blockchain use cases have expanded way beyond cryptocurrency now. Blockchain technology is emerging as a game-changer for multiple industries, including BFSI, healthcare, education, real estate, supply chain & logistics, and IoT, to name a few.

Blockchain's popularity and increasing industrial applications are skyrocketing because of its innate qualities the decentralized and distributed ledger is immutable and completely transparent. There's no need for a centralized authority to manage a Blockchain network. All the peers in the network verify the information stored in the Blockchain ledger. The bottom line – it is highly secure and reliable with hardly any point of entry for attackers.

Essentially, these qualities of Blockchain technology make it one of the most revolutionary technological tools of the 21st century. Being a tamper-proof ledger, Blockchain technology offers many possible use cases for businesses across all industries. According to the latest data, the global Blockchain market is projected to grow at a CAGR of 67.3% to reach a value of US$ 39.7 billion between 2020–20215.

Types of blockchain



Public Blockchain

Theseblockchains are completely open to following the idea of decentralization. They don't have anyrestrictions, anyone having a computer and internet can participate in the network**.**As the name is public this blockchain is open to the public, which means it is not owned by anyone.Anyone having internet and a computer with good hardware can participate in this public blockchain.All the computer in the network hold the copy of other nodes or block present in the networkIn this public blockchain, we can also perform verification of transactions or records.

Advantages:

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

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

Anonymous Nature: It is a secure platform to make your transaction properly at the same time; you are notrequired to reveal your name and identity in order to participate.

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

Disadvantages:

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

Energy Consumption: Proof of work is high energy-consuming. It requires good computer hardware to participate in the network

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

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

PrivateBlockchain

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

Advantages:

Speed: The rate of the transaction is high, due to its small size. Verification of each node is less time-consuming.

Scalability: We can modify the scalability. The size of the network can be decided manually.

Privacy: It has increased the level of privacy for confidentiality reasons as the businesses required.

Balanced: It is more balanced as only some user has the access to the transaction which improves the performance of the network.

Disadvantages:

Security- The number of nodes in this type is limited so chances of manipulation are there. These blockchains are more vulnerable.

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

Count- Since there are few nodes if nodes go offline the entire system of blockchain can be endangered.

Use Cases: With proper security and maintenance, this blockchain is a great asset to secure information without exposing it to the public eye. Therefore companies use them for internal auditing, voting, and asset management. An example of private blockchains is Hyperledger, Corda.

HybridBlockchain

It is the mixed content of the private and public blockchain, where some part is controlled by someorganization and other makes are made visible as a public blockchain.It is a combination of both public and private

blockchain.Permission-based and permissionless systems are used.User access information via smart contractsEven a primary entity owns a hybrid blockchain it cannot alter the transaction

Advantages:
Ecosystem: Most advantageous thing about this blockchain is its hybrid nature. It cannot be hacked as 51%of users don't have access to the network.

Cost: Transactions are cheap as only a few nodes verify the transaction. All the nodes don't carry the verification hence less computational cost.

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

Disadvantages:
Efficiency: Not everyone is in the position to implement a hybrid Blockchain. The organization also faces some difficulty in terms of efficiency in maintenance.

Transparency: There is a possibility that someone can hide information from the user. If someone wants to get access through a hybrid blockchain it depends on the organization whether they will give or not.

Ecosystem: Due to its closed ecosystem this blockchain lacks the incentives for network participation.
Use Case: It provides a greater solution to the health care industry, government, real estate, and financial companies. It provides a remedy where data is to be accessed publicly but needs to be shielded privately. Examples of Hybrid Blockchain are Ripple network and XRP token.

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

Advantages:
Speed: A limited number of users make verification fast. The high speed makes this more usable for organizations.

Authority: Multiple organizations can take part and make it decentralized at every level. Decentralized authority, makes it more secure.

Privacy: The information of the checked blocks is unknown to the public view. but any member belonging to the blockchain can access it.

Flexible: There is much divergence in the flexibility of the blockchain. Since it is not a very large decision canbe taken faster.

Disadvantages:
Approval: All the members approve the protocol making it less flexible. Since one or more organizations are involved there can be differences in the vision of interest.

Transparency: It can be hacked if the organization becomes corrupt. Organizations may hide information from the users.

Vulnerability: If few nodes are getting compromised there is a greater chance of vulnerability in this blockchain

Use Cases: It has high potential in businesses, banks, and other payment processors. Food tracking of the organizations frequently collaborates with their sectors making it a federated solution ideal for their use. Examples of consortium Blockchain are Tendermint and Multichain.

Blockchain Technology Use Cases

11. Smart Contracts

Smart contracts Blockchain-based contracts enforced in real-time. They are created as an agreement between two or more parties without the involvement of any intermediary. The contract exists across a distributed and decentralized Blockchain network. Smart contracts are now a staple in healthcare, real estate, and even for government agencies.

Use cases:

BurstIQ (healthcare)

BurstIQ used Big Data-based smart contracts to facilitate the transfer of sensitive medical data between patients and doctors. These contracts specify clear outlines and parameters for data sharing. They contain personalized health plans and other relevant details for individual patients.

12. Internet of Things (IoT)

The Internet of Things (IoT) industry is growing rapidly with billions of connected devices. The latest forecasts suggest that by 2030, there'll be 50 billion devices in use globally. As this number continues to grow, it will increase vulnerabilities as hackers can easily breach your data through a single connected device. By integrating Blockchain technology in IoT devices, the possibility of data breaches can be reduced to a great extent.

Use cases:

HYPR (IoT/Cybersecurity)

HYPR takes cybersecurity to the next level by combining smartphone technology with the highly secure FIDO token. This is the secret behind its True Passwordless Authentication feature. Through its decentralized credential solutions and biometric authentication, HYPR makes IoT devices tamper-proof.

13. Money Transfer

Money transfer and payment processing are the most excellent Blockchain technology use cases. Blockchain tech enables lightning-fast transactions in real-time. This has already transformed the BFSI sector for good as it saves both time and money (mostly eliminates transaction fees charged by banks/financial institutions).

Use cases:

Circle (FinTech)

Circle uses USD Coin (USDC), the fastest-growing regulated Stablecoin, to help individuals run and establish their internet business. The platform offers around seven cryptocurrencies (Bitcoin, Monero, Zcash, etc.). Every month, Circle handles over $2 billion in cryptocurrency investments and exchanges.

14.Digital Media

Digital media companies are burdened with many challenges like data privacy, piracy of intellectual property, royalty payments, and copyright infringement, among other issues. By incorporating Blockchain technology into the digital media infrastructure, companies can protect their intellectual property, maintain data integrity, target the right customers, and ensure that artists receive their royalty payments in due time.

Use cases:

MadHive (Media & Advertising)

MadHive is a New York bases advertising agency that delivers full-stack solutions to digital marketers. You can use MadHive to monitor, record, and create reports on customertarget="_blank" rel="nofollow" activity and stores all the information in a private Blockchain ledger. By offering real-time data monitoring and audience analytics reports, MadHive allows marketers to understand their target audiences better.

15.Education:

Blockchain technology can be seamlessly integrated into control systems and document storage. The key benefit of this **blockchain use case** is the incapability to modify data saved into the system. The information can be added but can't be overwritten. Moreover, the document's legality can be easily confirmed because everybody can observe who wrote it and when.

Comparatively, fewer people have used the benefit of blockchain in education. One of the pioneers is the University of Nicosia in Cyprus. This university used blockchain technology to store certificates and degrees. Moreover, it is the first university to accept cryptocurrencies as payment for tuition.

6.Medical Field:

The use of blockchain technology in the healthcare industry proves to be a lifesaver, especially in terms of data processing. The reason is the medical business has long been exploited. Since medical data is massive, blockchain technology is useful for a small amount of information.

One of the best**blockchain use cases** in the medical field is the Healthureum system. Based on blockchain technology, it works as a tool to control and account for the data concerning modifications in the medical records. It means that the answer was for data to be saved outside the blocks, whereas the links directing to huge files will be saved on the blockchain.

7.Entertainment:

One of the most interesting and prevailing**blockchainusecases**is entertainment. Blockchain technology prevents gambling sanctions in several jurisdictions due to cryptocurrency's non-recognition as property or cash. Furthermore, blockchain has raised the curiosity of the music industry.

Blockchain technology can store data on all transactions, encrypted data on finance, ownership rights, and smart contracts too. For example, Stem is one of the**blockchain use cases** in the entertainment sector. It is a blockchain-centred payment and distribution platform for audiovisual products. It helps you publish material, handle contracts, and do payments from a single place.

Case Studies:

Here are the **blockchainusecases** in entertainment:

- Plague Hunters: It is a free strategy game with an inbuilt Ethereum marketplace that uses NFT transactions for purchasing and selling weapons and the hunters.
- Beyond the Void: This game uses Ethereum'sblockchain to permit players to purchase, sell, and sell "cosmetic in-game items" via NFT transactions.

8. Real Estate:

The real estate market faces many issues, specifically connecting buyers and sellers. Currently, buyers need to meet the seller or brokers to finalize a deal. The duration can range from a few months to a year.

Furthermore, the paperwork can be a tedious task. It demands a lot of effort to complete the paperwork, which can still be impacted by incorrect data input. There are challenges while verifying the property's ownership. Moreover, frauds can happen in several ways, costing the buyer money.

14.6 Conclusion:we have successfullyWrite a survey report on types of Blockchains and its real time use cases.

Assignment No:-15

15.1 Title

Assignment based on hyper ledger.

15.2 Problem Definition: Write a program to create a Business Network using hyper ledger.

15.3 Prerequisite:

11. Basic knowledge of cryptocurrency
12.Basic knowledge of distributed computing concept
13. Working of blockchain

15.4 Software Requirements:
RemixIDE

Theory:
Hyperledger Composer is an extensive, open development toolset and framework to make developing blockchain applications easier. The primary goal is to accelerate time to value, and make it easier to integrate your blockchain applications with the existing business systems.

· You can use Composer to rapidly develop use cases and deploy a blockchain solution in days.
· Composer allows you to model your business network and integrate existing systems and data with your blockchain applications.

· Hyperledger Composer supports the existing Hyperledger Fabric blockchain infrastructure and runtime.

· Hyperleder Composer generate business network archive (bna) file which you can deploy on existing Hyperledger Fabric network

You can use Hyperledger Composer to model business network, containing your existing assets and the transactions related to them.

Key Concepts of
Hyperledger
Composer

11. Blockchain State Storage: It stores all transaction that happens in your hyperledger composer application. It stores transaction in Hyperledger fabric network.

12. Connection Profiles: Connection Profiles to configuration JSON file which help composer to connect to Hyperledger Fabric. You can find Connection Profile JSON file in user's home directory.

13. Assets: Assets are tangible or intangible goods, services, or property, and are stored in registries. Assets

can represent almost anything in a business network, for example, a house for sale, the sale listing, the land registry certificate for that house. Assets must have a unique identifier, but other than that, they can contain whatever properties you define.

14. Participants: Participants are members of a business network. They may own assets and submit transactions. Participant must have an identifier and can have any other properties.

15. Identities and ID cards: Participants can be associated with an identity. ID cards are a combination of an identity, a connection profile, and metadata. ID cards simplify the process of connecting to a business network.

6. Transactions: Transactions are the mechanism by which participants interact with assets. Transaction processing logic you can define in JavaScript and you can also emit event for transaction.

7. Queries: Queries are used to return data about the blockchain world-state. Queries are defined within a business network, and can include variable parameters for simple customisation. By using queries, data can be easily extracted from your blockchain network. Queries are sent by using the Hyperledger Composer API.

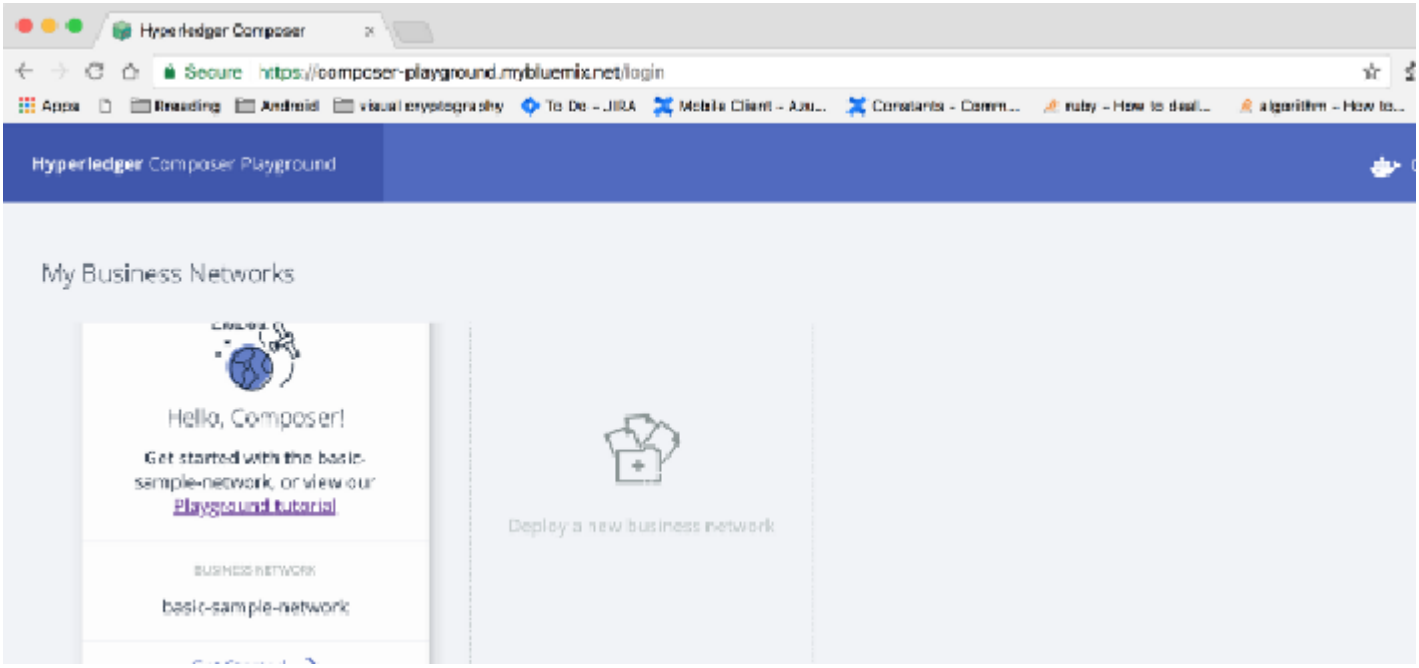
8. Events: Events are defined in the model file. Once events have been defined, they can be emitted by transaction processor functions to indicate to external systems that something of importance has happened to the ledger.

9. Access Control: Hyperledger is enterprise blockchain and access control is core feature of any business blockchain. Using Access Control rules you can define who can do what in Business networks. The access control language is rich enough to capture sophisticated conditions.

10. Historian registry: The historian is a specialised registry which records successful transactions, including the participants and identities that submitted them. The historian stores transactions as HistorianRecord assets, which are defined in the Hyperledger Composer system namespace.

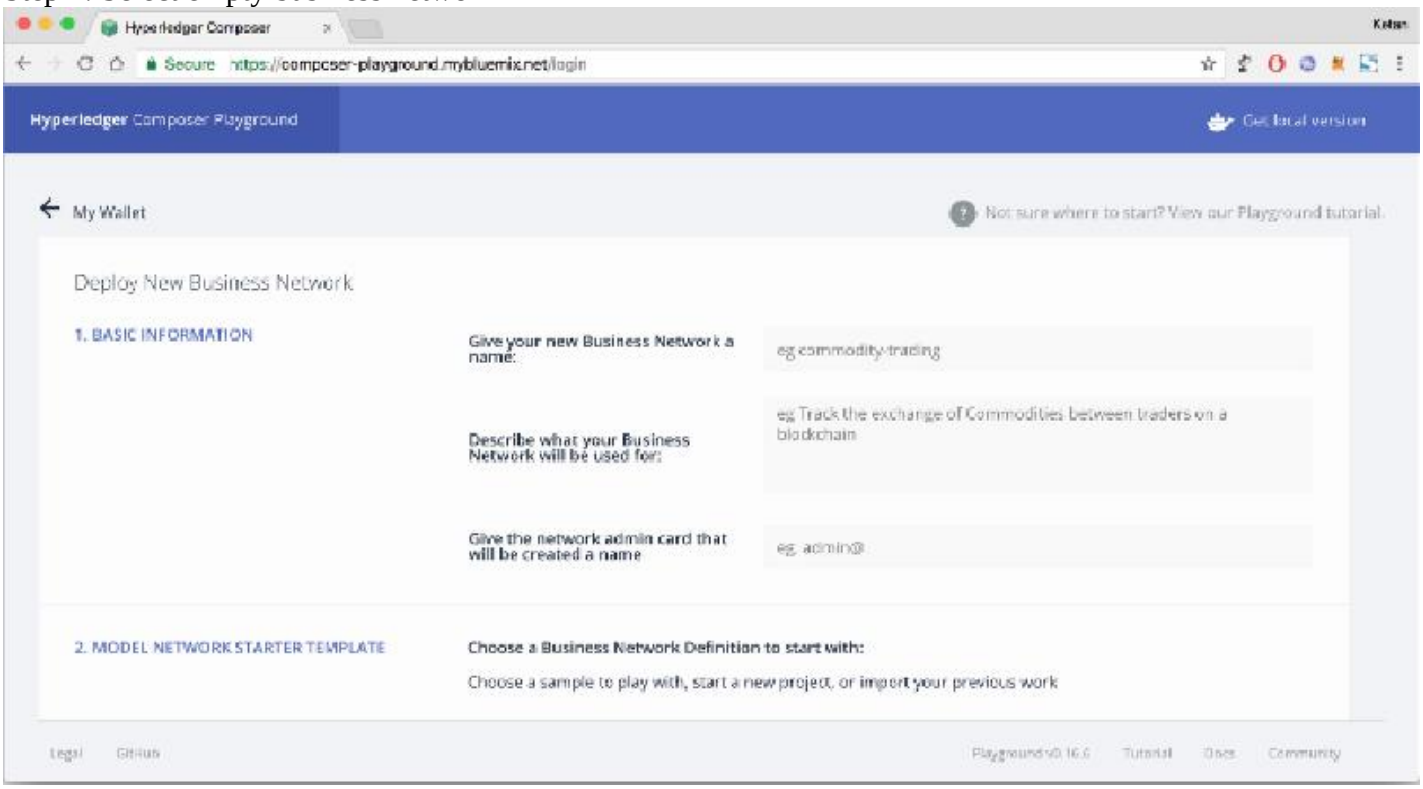
Let’s create first
Hyperledger
Composer Application

Step 1: Start Hyperledger Composer Online version of Local. Click on Deploy a new business network

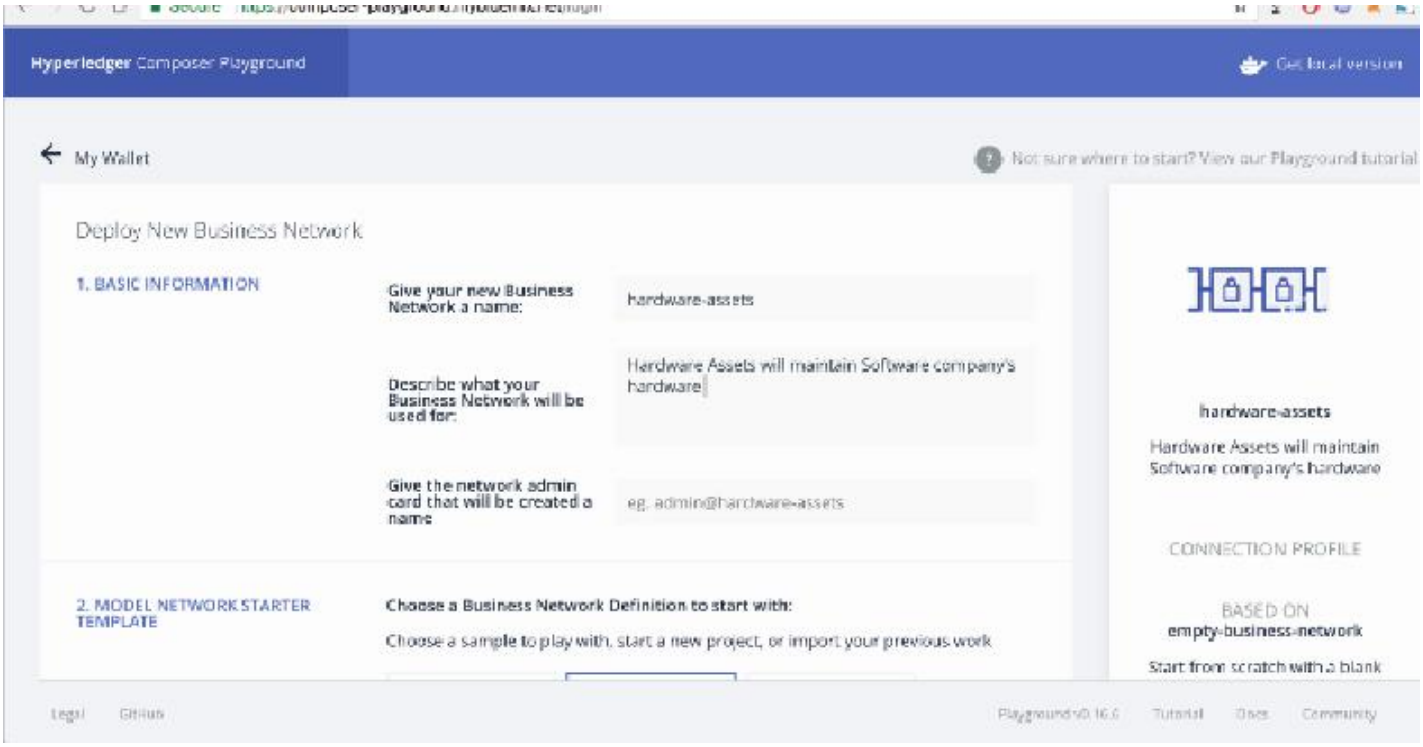


Hyperledger Composer Playground Online version

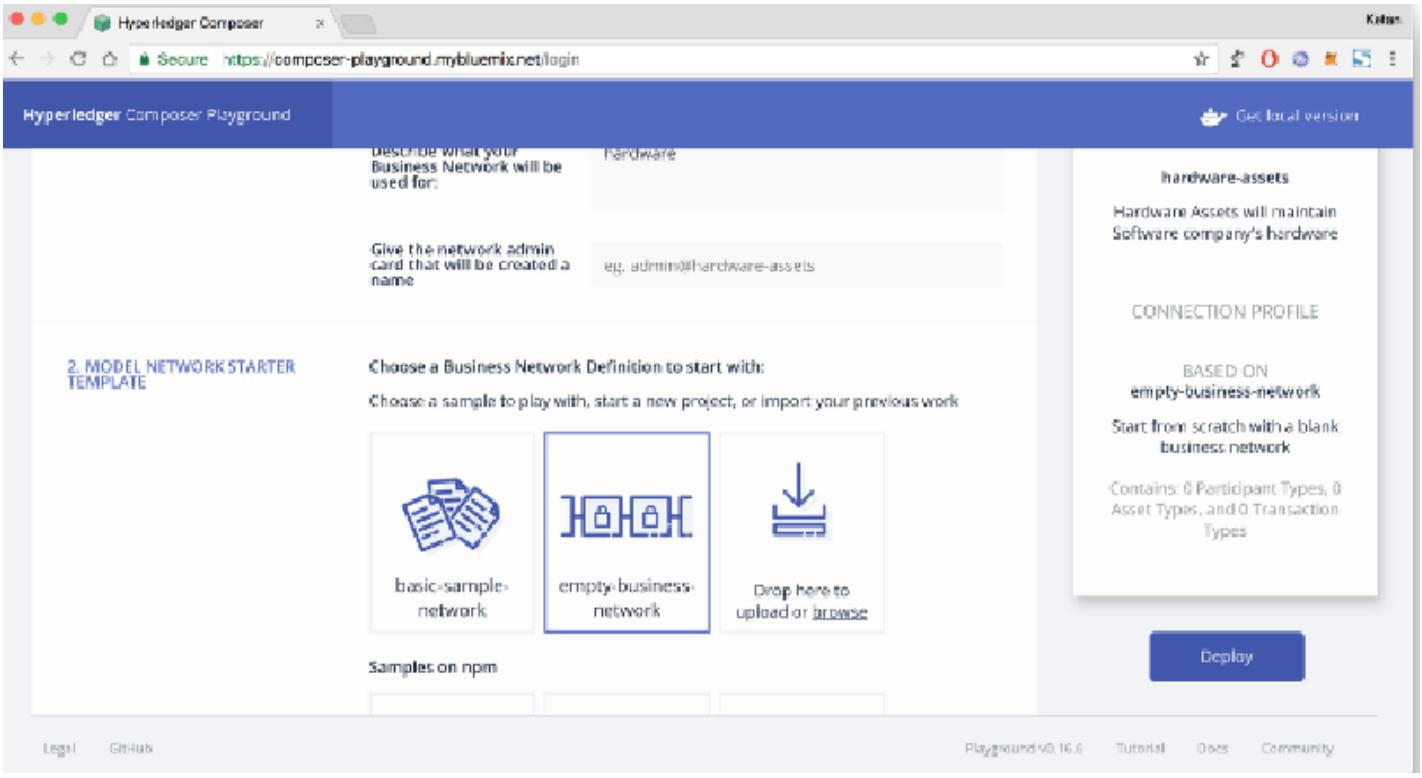
Step 2: Select empty business network



Step 3: Fill basic information, select empty business network and click “deploy” button from right pannel

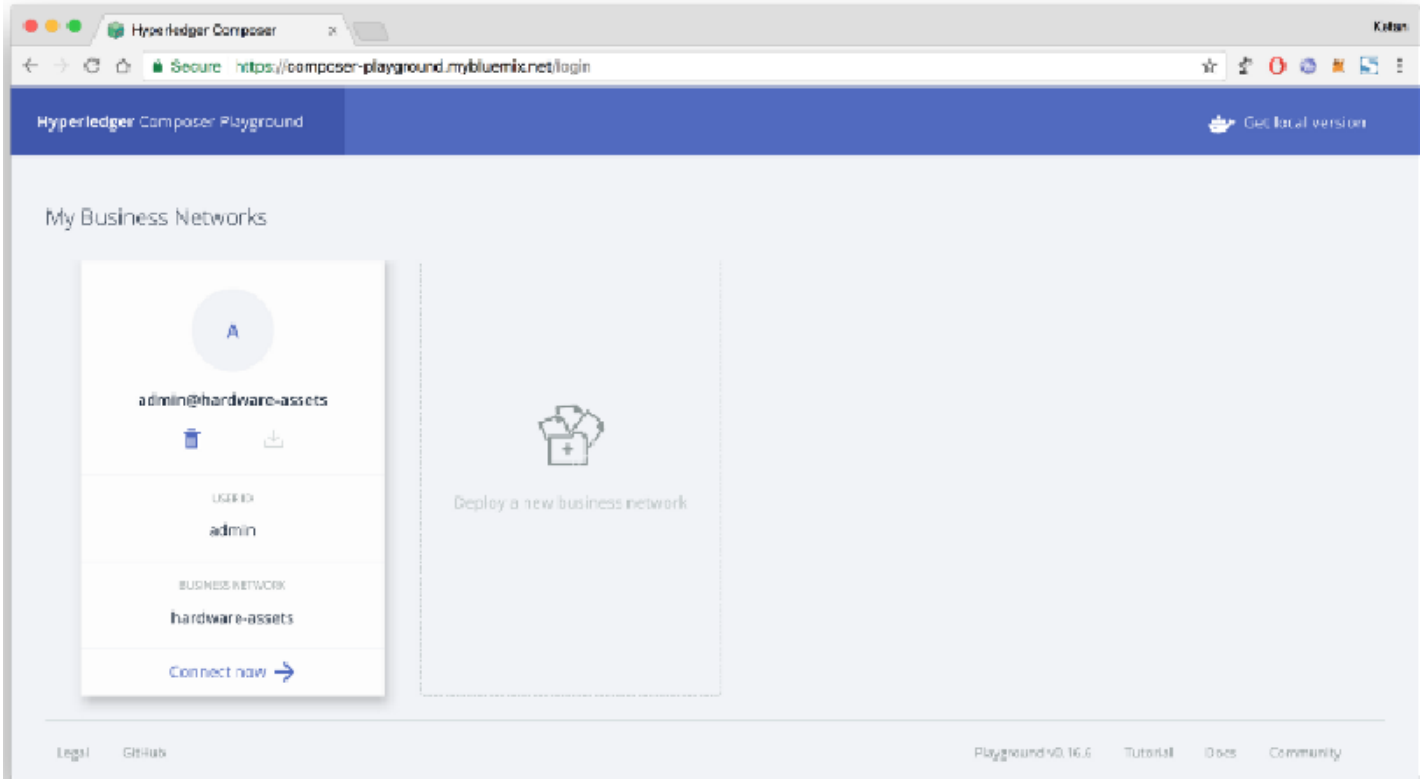


Fill basic information

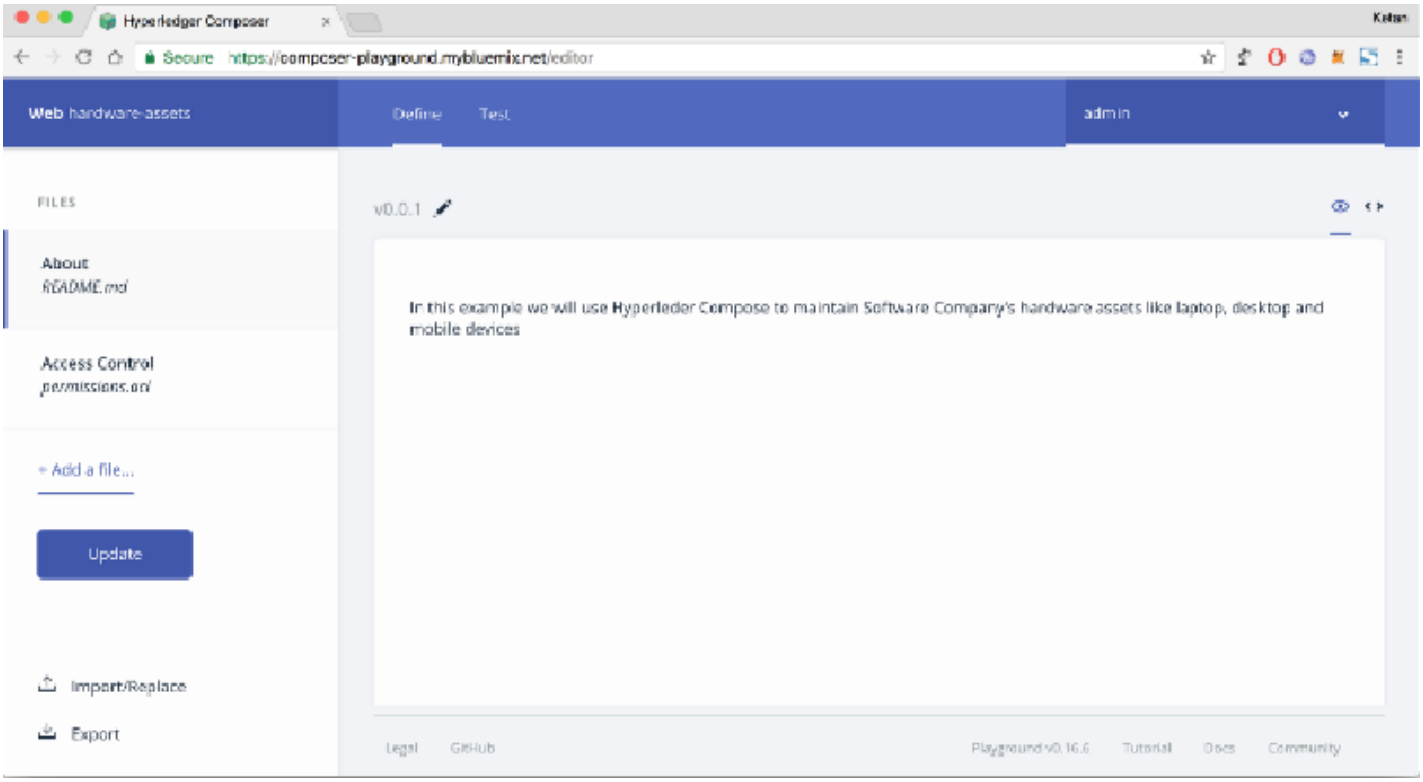


Select empty business network

Step 4: Connect to “hardware-assets” business network that we have just deployed

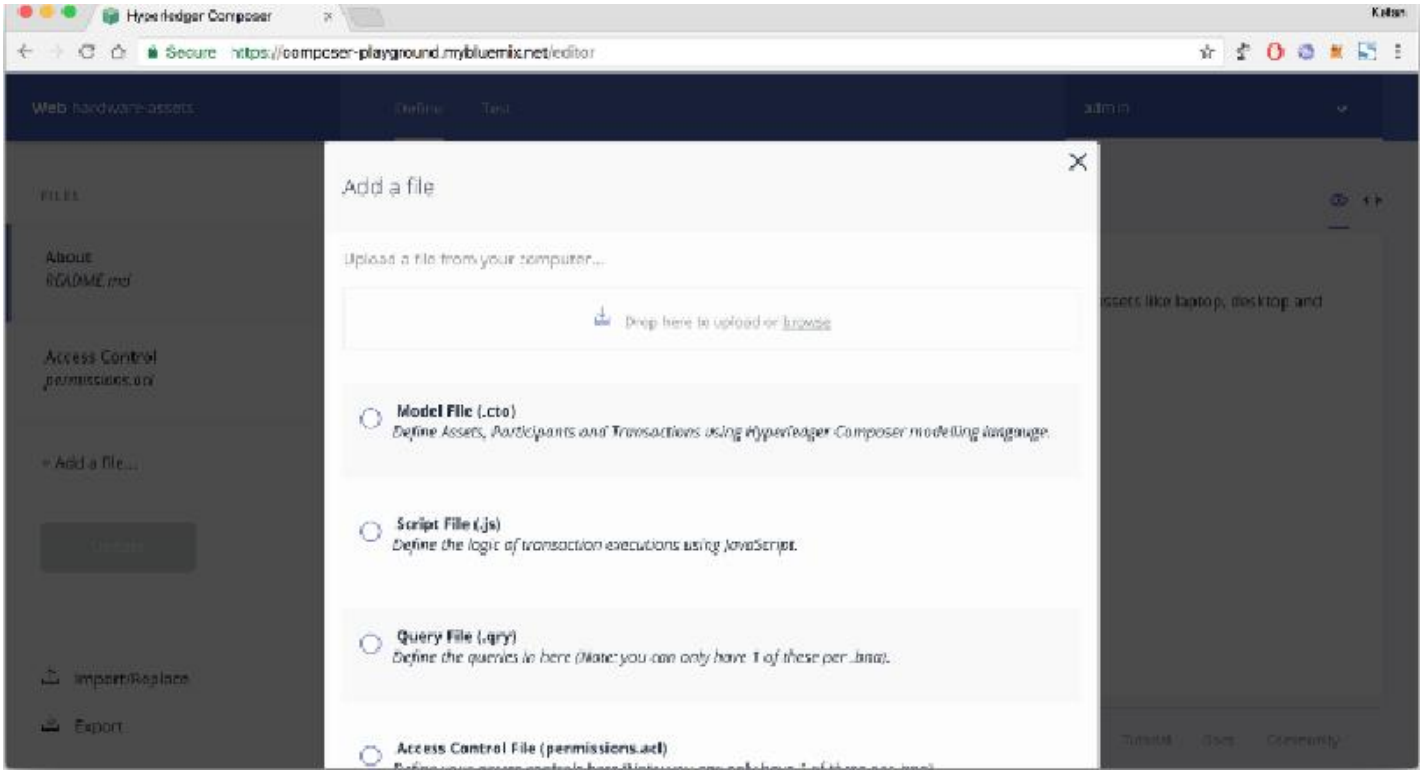


Click on “connect now” button



Inside hardware-assets business network.

Step 5: Click on “+Add a file...” from left panel and select “model file (.cto)”



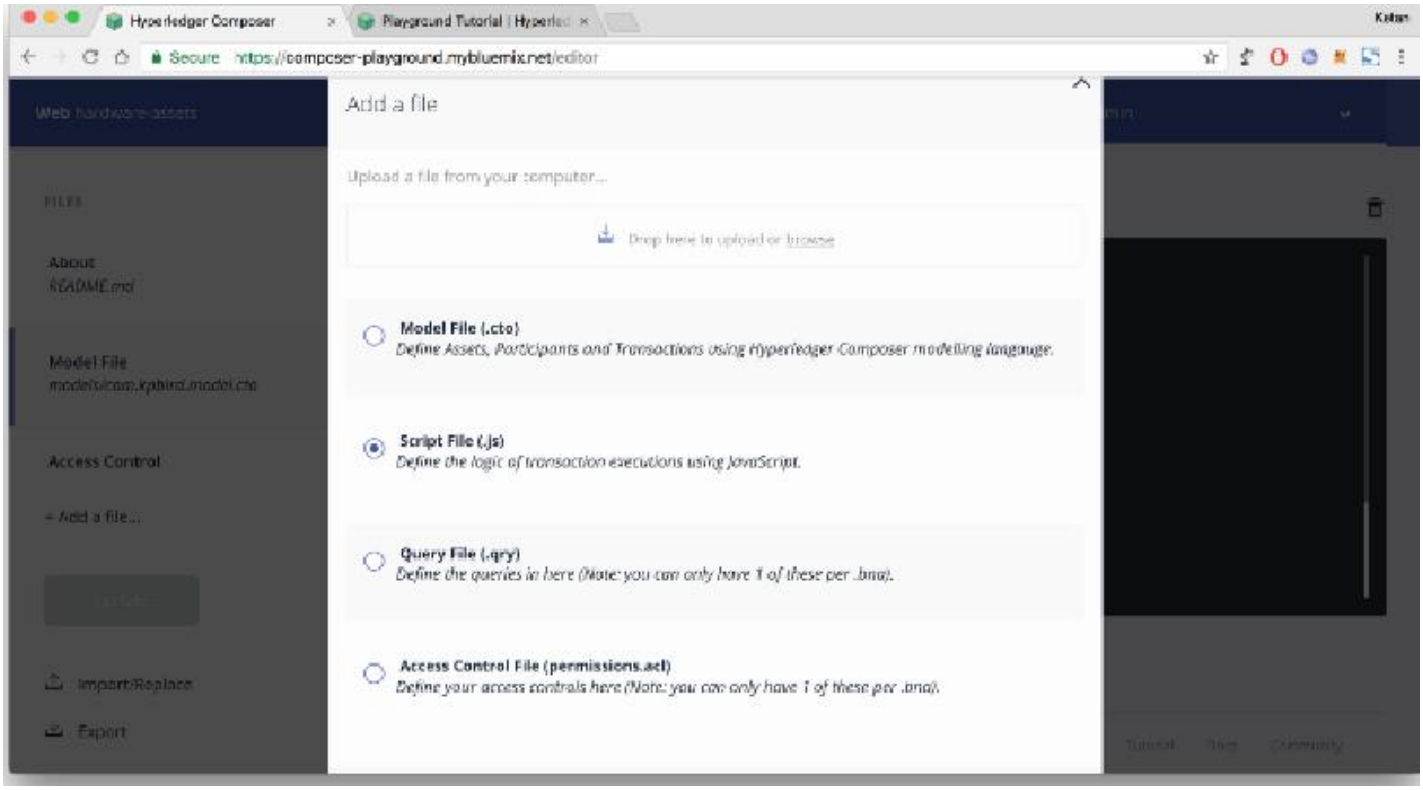
Write following code in model file. Model file contain asset in our case it's hardware, participant in our case participants are employee of organisation and transaction as Allocate hardware to employee. Each model has extra properties. Make sure your have proper and unique namespace. In this example I am using "com.kpbird" as namespace. You can access all models using this namespace i.e. com.kpbird.Hardware, com.kpbird.Employee

```
/**
* Hardware model
*/namespace com.kpbirdasset Hardware identified by hardwareId {
o String hardwareId
o String name
o String type
o String description
o Double quantity
→ Employee owner
}
participant Employee identified by employeeId {
o String employeeId
o String firstName
o String lastName
}
transaction Allocate {
→ Hardware hardware
→ Employee newOwner
}
```

Hyperledger modeling language

reference: https://hyperledger.github.io/composer/reference/cto_language.html

Step 6: Click on "+Add a file..." from left panel and select "script file (*.js)"



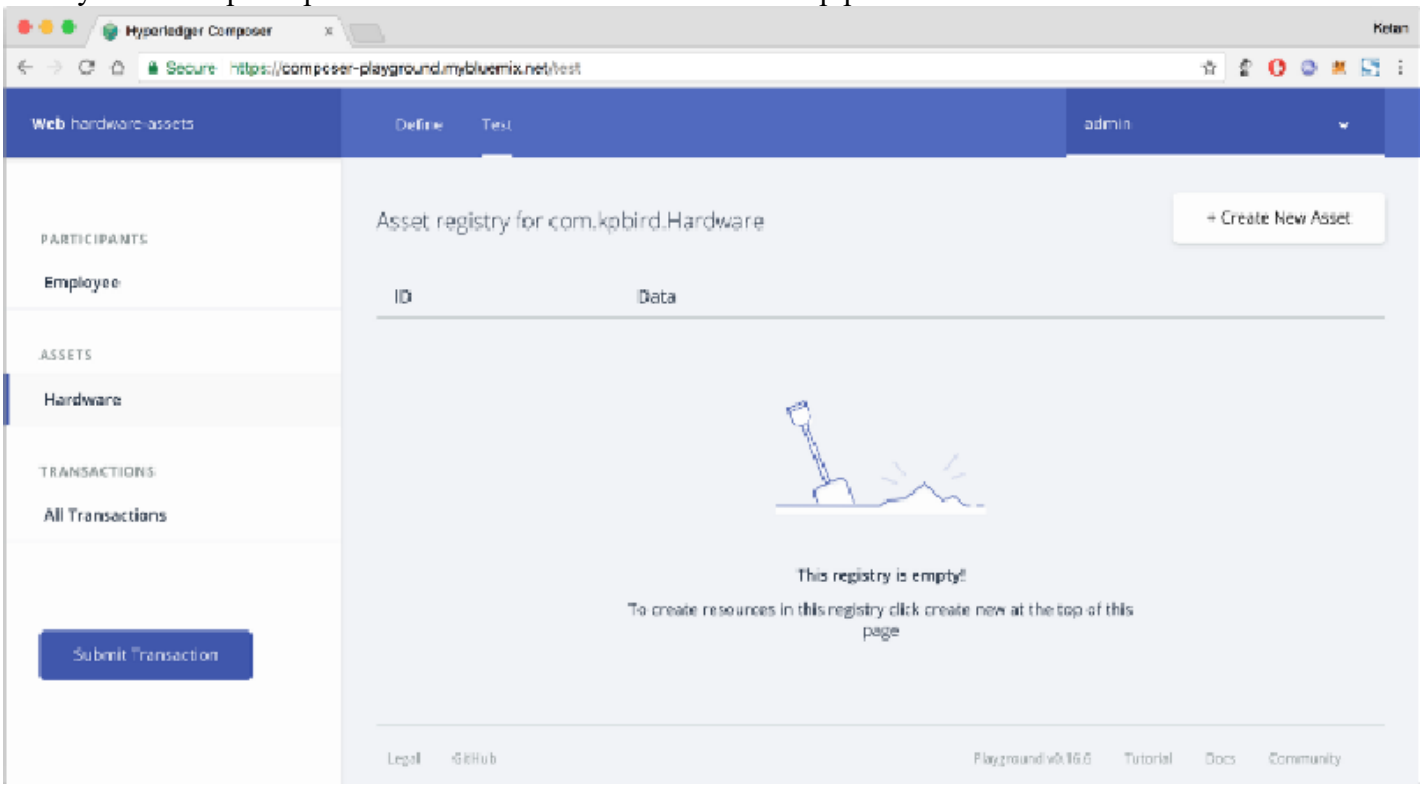
Write following code in Script File. In Script we can define transaction processing logic. In our case we want to allocate hardware to the employee so, we will update owner of hardware. Make sure about annotation above functions @params and @transaction

```
/**
* Track the trade of a commodity from one trader to another
* @param {com.kpbird.Allocate} trade — the trade to be processed
* @transaction
*/
function allocateHardware(allocate) {
allocate.hardware.owner = allocate.newOwner;
return getAssetRegistry('com.kpbird.Hardware')
.then(function (assetRegistry) {
return assetRegistry.update(allocate.hardware);
});
}
```

Step 7: permissions.acl file sample is already available, Add following code in permissions.acl file.

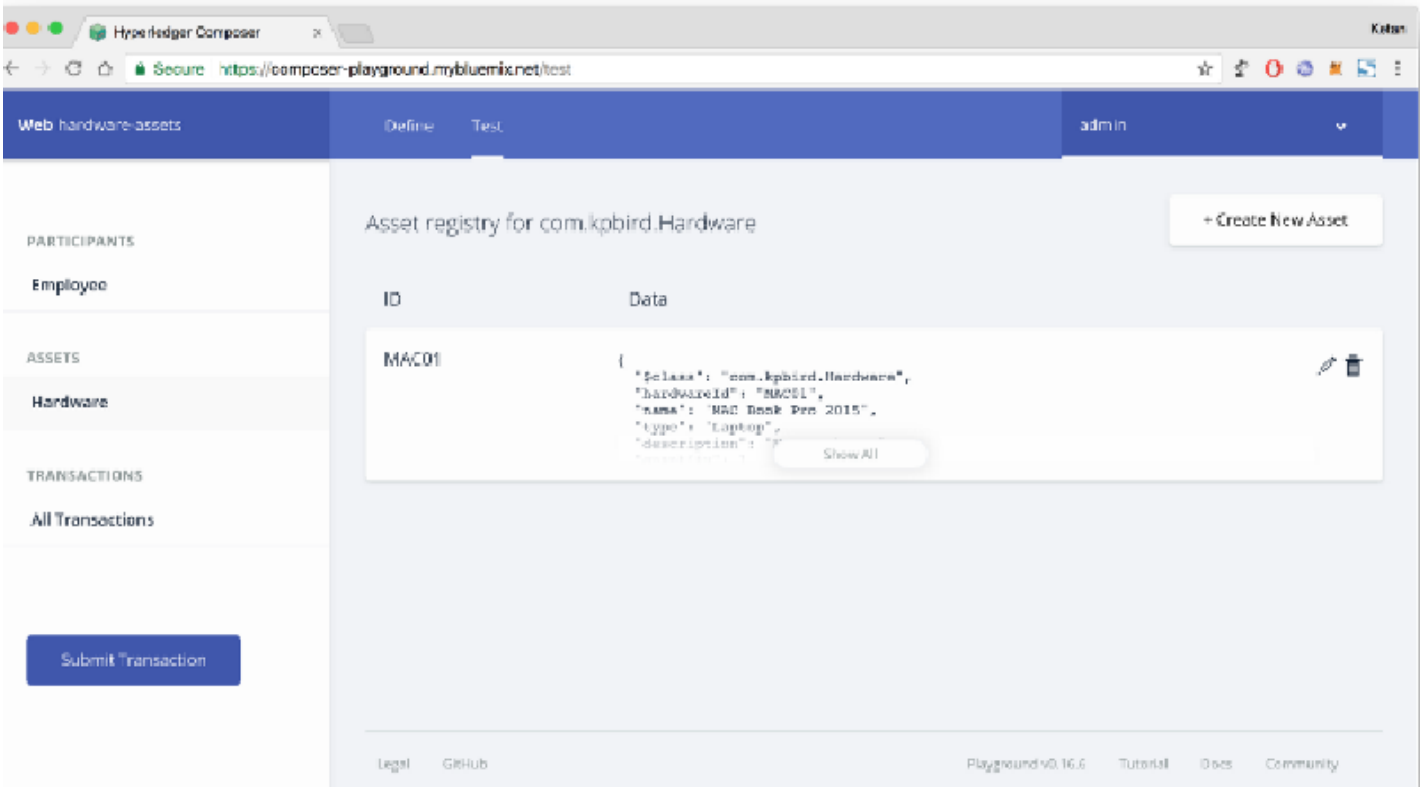
```
**
* New access control file
*/
rule AllAccess {
description: "AllAccess — grant everything to everybody."
participant: "ANY"
operation: ALL
resource: "com.kpbird.**"
action: ALLOW
}rule SystemACL{
```

description: “System ACL to permit all access”
participant: “org.hyperledger.composer.system.Participant”
operation: ALL
resource: “org.hyperledger.composer.system.**”
action: ALLOW
}
Step 8: Now, It’s time to test our hardware assets business network. Hyperledger composer gives “Test” facility from composer panel it self. Click on “Test” tab from top panel



Test feature of Hyperledger Composer

Step 9: Create Assets. Click on “Hardware” from left panel and click “+ Create New Assets” from right top corner and add following code. We will create Employee#01 in next step. Click on “Create New” button
{
“$class”: “com.kpbird.Hardware”,
“hardwareId”: “MAC01”,
“name”: “MAC Book Pro 2015”,
“type”: “Laptop”,
“description”: “Mac Book Pro”,
“quantity”: 1,
“owner”: “resource:com.kpbird.Employee#01”
}



After adding Hardware assets

Steps 10: Let's create participants. Click "Employee" and click "+ Create New Participants" and add following

code. We will add two employees

{

"$class": "com.kpbird.Employee",

"employeeId": "01",

"firstName": "Ketan",

"lastName": "Parmar"

}

Click on "Create New" on dialog

{

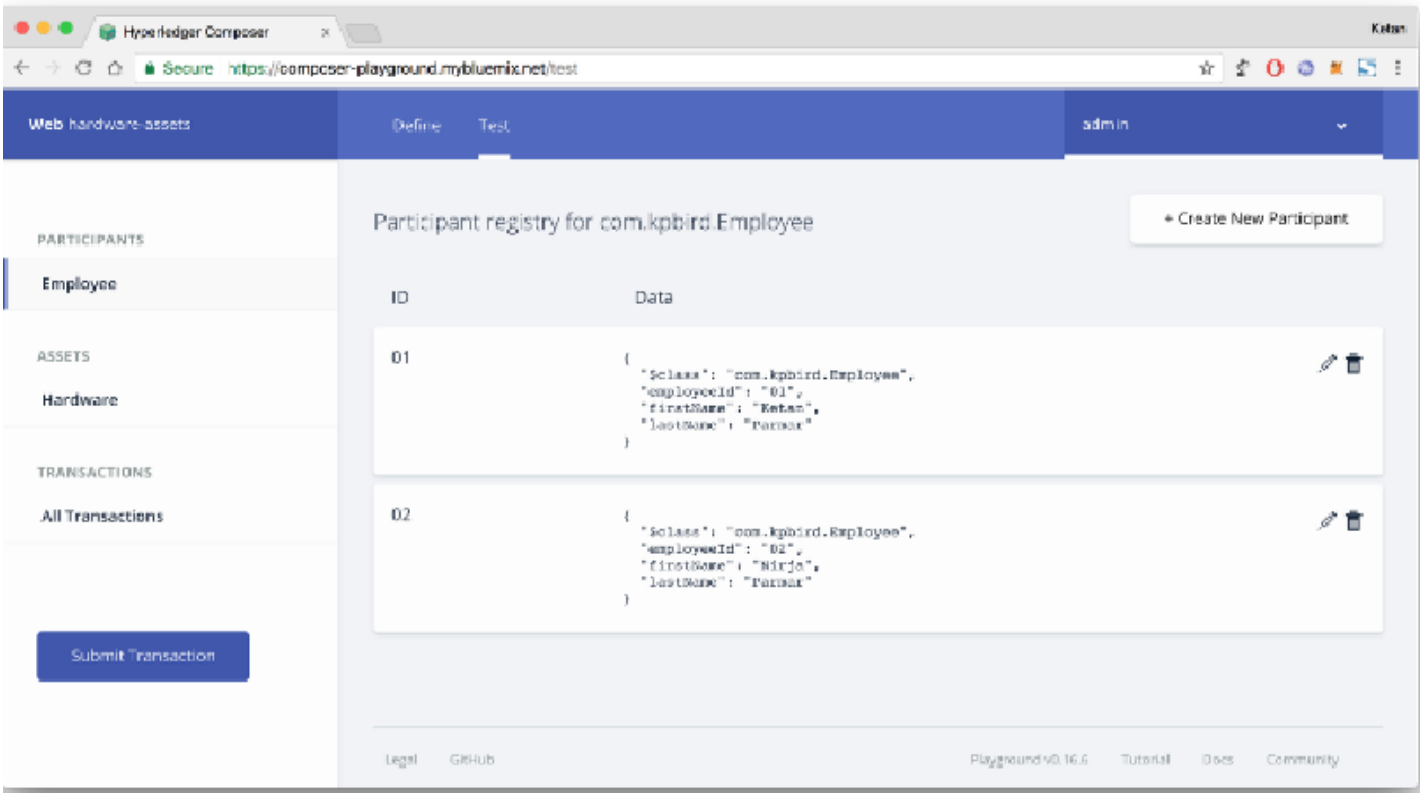
"$class": "com.kpbird.Employee",

"employeeId": "02",

"firstName": "Nirja",

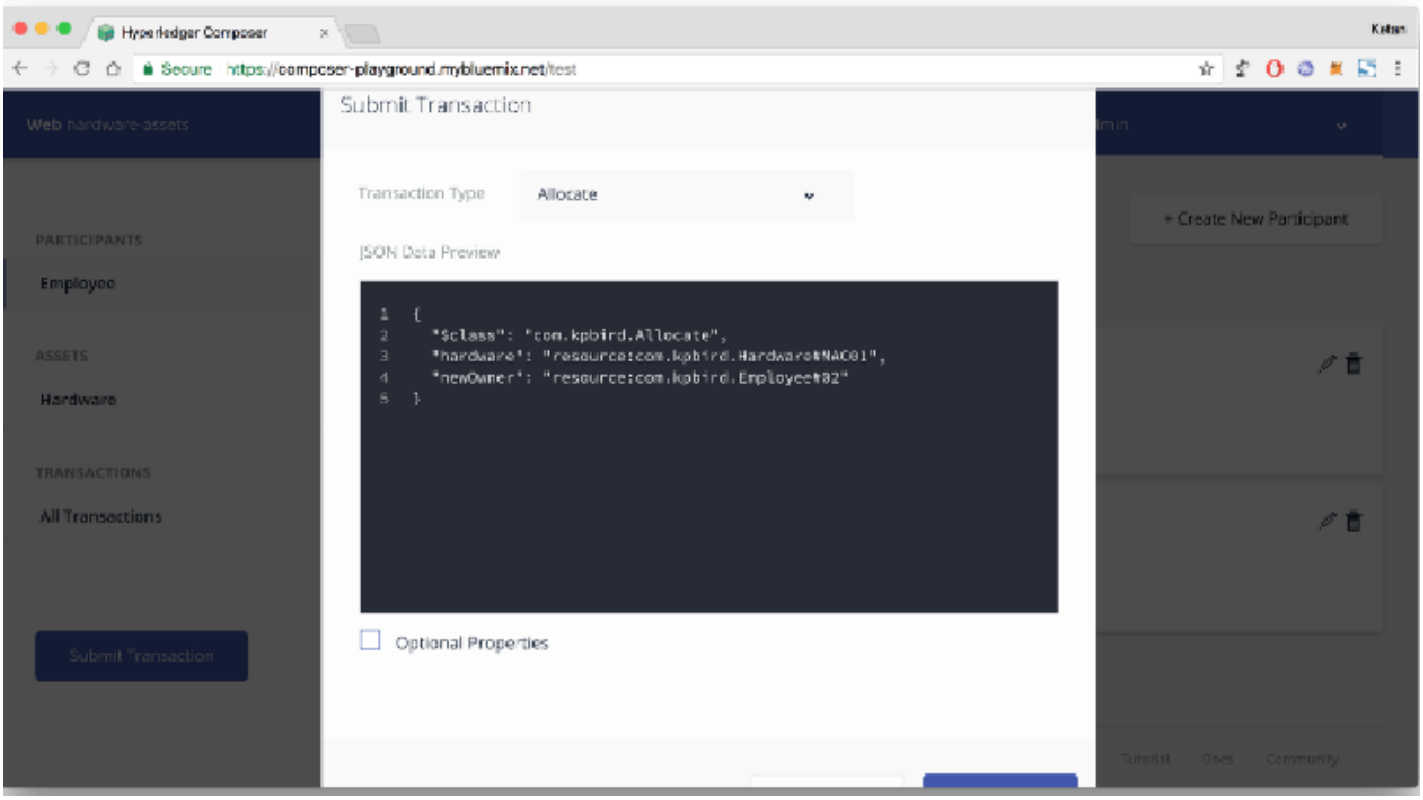
"lastName": "Parmar"

}



We have two employees

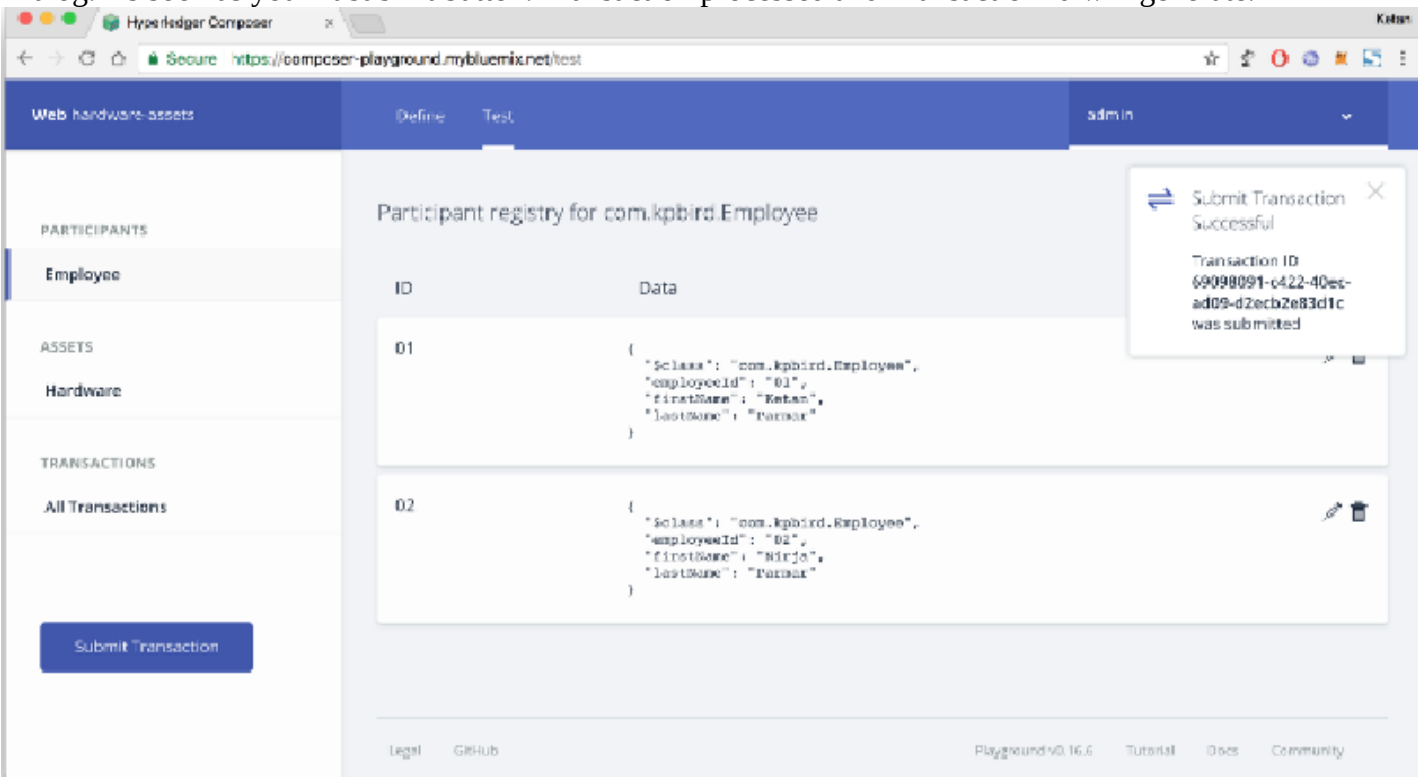
Step 11: It's time to do transaction, We will allocate Macbook Pro from Ketan (Employee#01) to Nirja (Employee#02). Click on "Submit Transaction" button from left panel. In Transaction dialog, We can see all transaction functions on top "Transaction Type" dropdown



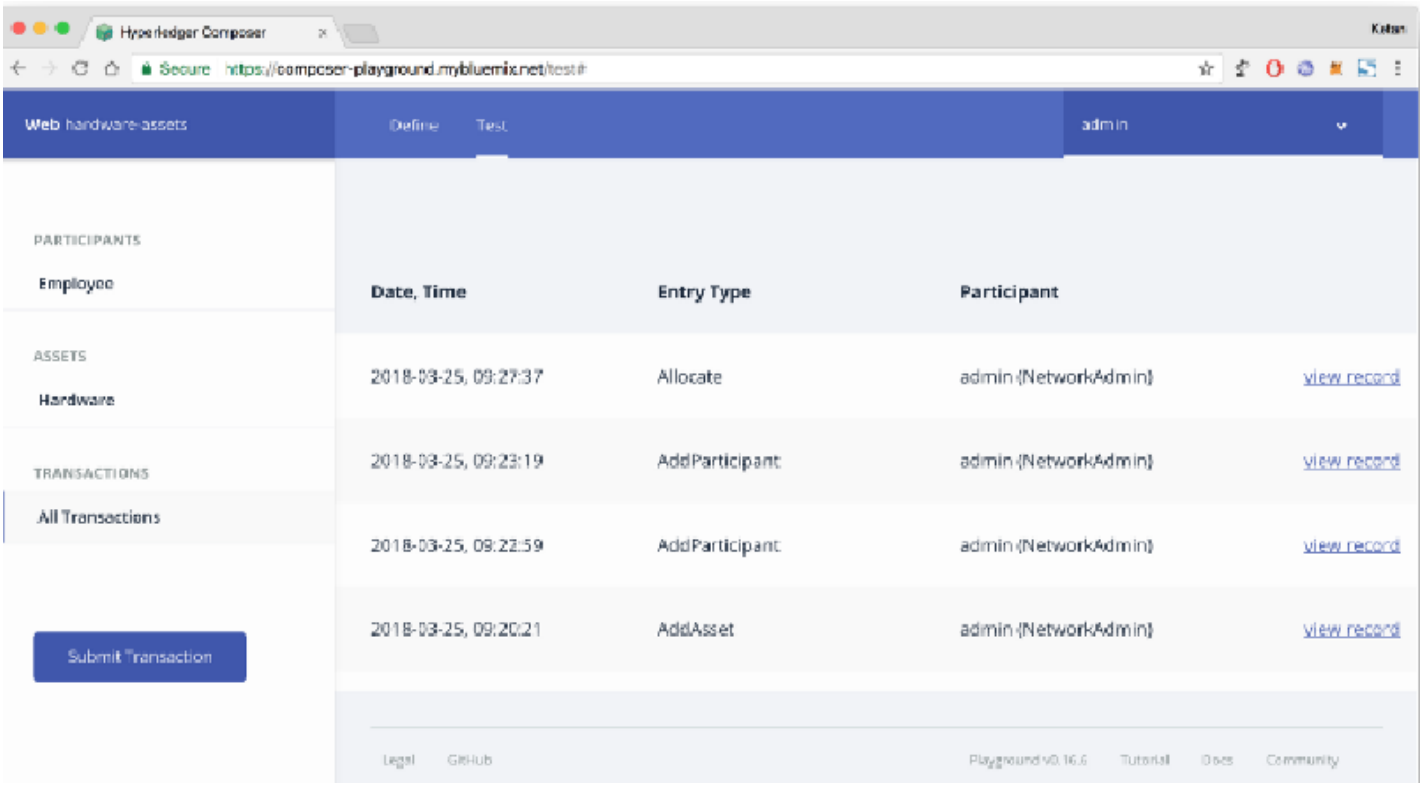
Submit Transaction Dialog

{

“$class”: “com.kpbird.Allocate”,
“hardware”: “resource:com.kpbird.Hardware#MAC01”,
“newOwner”: “resource:com.kpbird.Employee#02”
}
Now, We are allocating Mac01 to Employee 012. Click Submit button after update above JSON in Transaction
Dialog. As soon as you hit submit button. Transaction processed and Transaction Id will generate.

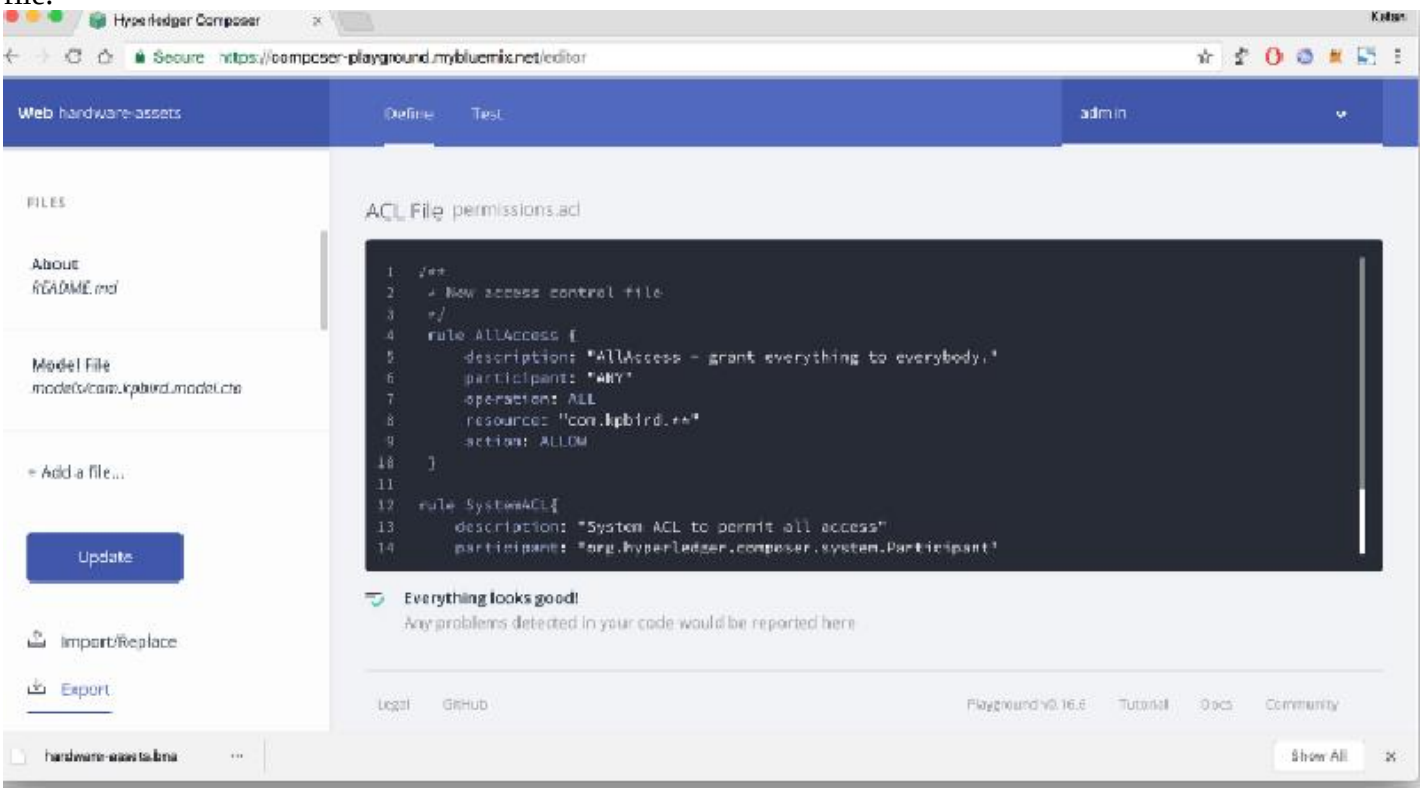


Step 12: Click on “All Transactions” from left panel to verify all transactions. In following screenshots you can see add assets, ass participants and allocation all operation are consider as transactions. “view records” will give us more information about transaction.



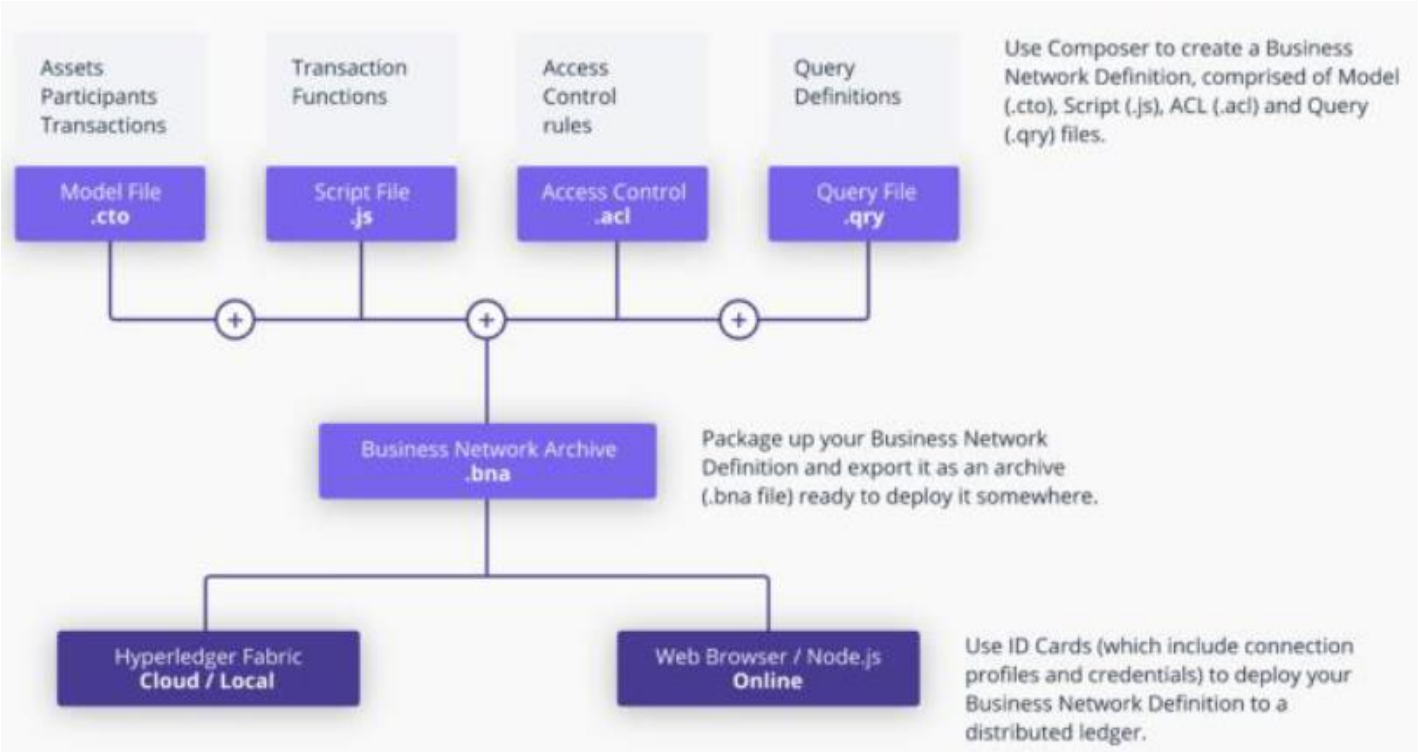
All Transactions

Step 13: Now, It's time to deploy "hardware-assets" business network to Hyperledger Fabric. Click on "Define" tab from top panel and click "Export" button from left panel. Export will create hardware-assets.bna file.



Download hardware-assets.bna file

.bna is Business Network Archive file which contains model, script, network access and query file



Step 14: Start Docker and run following commands from ~/fabric-tools directory

Install business network to Hyperledger Fabric, If business network is already installed you can use “update”

instead of “install”

```
$composer runtime install -c PeerAdmin@hlfv1 -n hardware-assets
Ketan-Parmar:fabric-tools ketan$ composer runtime install -c PeerAdmin@hlfv1 -n hardware-assets
✔ Installing runtime for business network hardware-assets. This may take a minute...

Command succeeded
```

Following command will deploy and start hardware-assets.bna file. Change hardware-assets.bna file before you execute following command. networkadmin.card file will generate in ~/fabric-tools directory from previous command.

```
$composer network start — card PeerAdmin@hlfv1 — networkAdmin admin — networkAdminEnrollSecret adminpw — archiveFile /Users/ketan/Downloads/hardware-assets.bna — file networkadmin.card
```

```
Ketan-Parmar:fabric-tools ketan$ composer network start --card PeerAdmin@hlfv1 --networkAdmin admin --networkAdminEnrollSecret adminpw --archiveFile /Users/ketan/Downloads/hardware-assets.bna --file networkadmin.card
Starting business network from archive: /Users/ketan/Downloads/hardware-assets.bna
Business network definition:
        Identifier: hardware-assets@0.0.1
        Description: Hardware Assets will maintain Software company's hardware

Processing these Network Admins:
        userName: admin

✔ Starting business network definition. This may take a minute...
Successfully created business network card:
        Filename: networkadmin.card

Command succeeded
```

To connect business network you need connection card. so we can import networkadmin.card using following command

$composer card import -f networkadmin.card

To make sure networkadmin.card successfully install you can list cards using following command

$composer card list

```
Ketan-Parmar:fabric-tools ketan$ composer card list
The following Business Network Cards are available:

Connection Profile: hlfv1
```

Card Name	UserId	Business Network
admin@hardware-assets	admin	hardware-assets
PeerAdmin@trade-network	PeerAdmin	trade-network
admin@trade-network	admin	trade-network
PeerAdmin@hlfv1	PeerAdmin	

```
Issue composer card list --name <Card Name> to get details a specific card

Command succeeded
```

Following command will make sure that our hardware-assets business network is successfully running in Hyperledger Fabric.

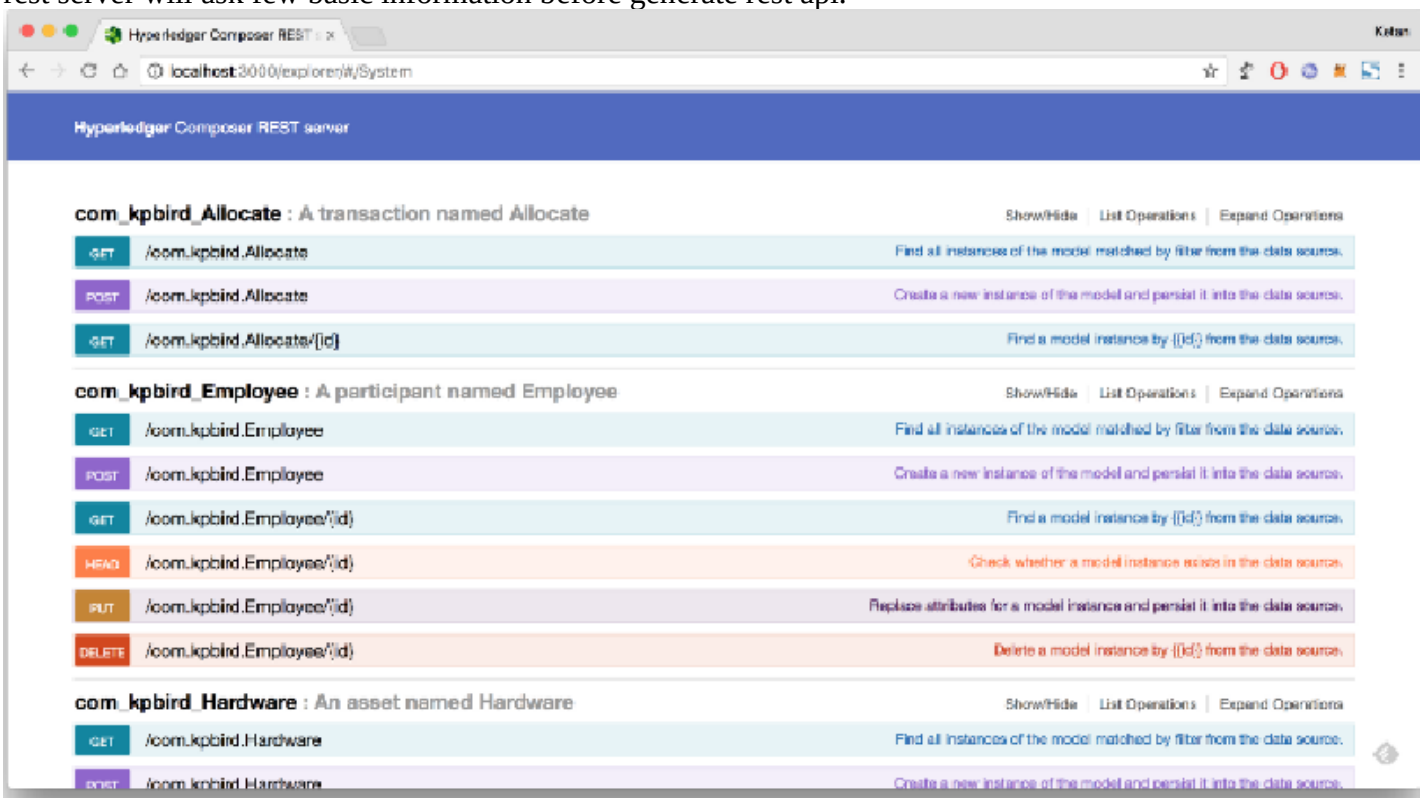
$composer network ping — card admin@hardware-assets

```
Ketan-Parmar:fabric-tools ketan$ composer network ping --card admin@hardware-assets
The connection to the network was successfully tested: hardware-assets
        version: 0.16.0
        participant: org.hyperledger.composer.system.NetworkAdmin#admin

Command succeeded
```

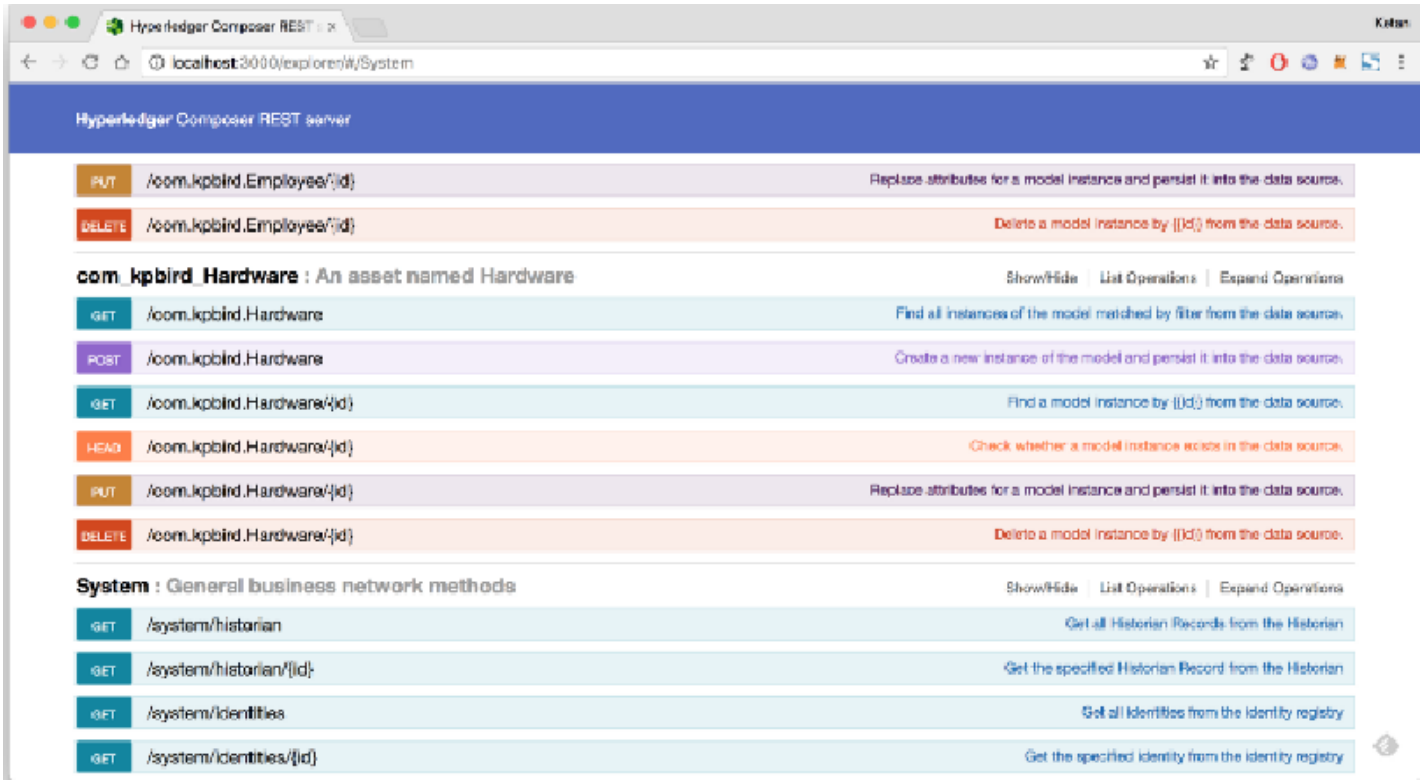
Now It's time to interact with REST API. To develop Web or Mobile Application we require REST API. you can run following command to generate REST API for hardware-assets business network.

```
$composer-rest-server
Ketan-Parmar:fabric-tools ketan$ composer-rest-server
? Enter the name of the business network card to use: admin@hardware-assets
? Specify if you want namespaces in the generated REST API: always use namespaces
? Specify if you want to enable authentication for the REST API using Passport: No
? Specify if you want to enable event publication over WebSockets: Yes
? Specify if you want to enable TLS security for the REST API: No

To restart the REST server using the same options, issue the following command:
   composer-rest-server -c admin@hardware-assets -n always -w true

Discovering types from business network definition ...
Discovered types from business network definition
Generating schemas for all types in business network definition ...
Generated schemas for all types in business network definition
Adding schemas for all types to Loopback ...
Added schemas for all types to Loopback
Web server listening at: http://localhost:3000
Browse your REST API at http://localhost:3000/explorer
```

rest server will ask few basic information before generate rest api.



REST API for our hardware assets



REST API methods for all operations

15.5 Conclusion: In this way we have learnt about hyperledger and its use case in business world.

Mini Project

Title:-Assignment based on Mini Project.
Problem definition: Create a dApp (de-centralized app) for e-voting system.

Software requirement:-RemixIDE Online

Learning Objective:-we are going to learn how to create a dApp (de-centralized app) for e-voting system.

Outcome: - You are able to cretae a dApp (de-centralized app) for e-voting system.

Theory: **Decentralized Voting Application (DApps)** which is built on Solidity Language. This Project showcases a lot of Solidity's features. It implements a voting contract. Of course, the main problem of electronic voting is how to prevent to assign the duplicate Vote.

Some Important Concepts are:
11. Contract: A contract is just like a class in Solidity which consists (its functions) and data (its state) that resides at a specific address on the Ethereum Blockchain. In each Contract, we can define State Variables, Methods, and Events, etc. A smart contract runs exactly as programmed without any possibility of downtime, censorship, fraud, and third-party interference.
12. Structure: The Structure is Collection of different type of Data Types same like C and C++, which is shown in the following example:

```
struct Voter{
        bool authorized;
        bool voted;
    }
```

13. Mapping: Mapping is just like Hash tables It stores the value based on key. They cannot be used as parameters or return parameters of contract functions that are publicly visible. You cannot iterate over mappings, i.e. you cannot enumerate their keys. It possible to implement a data structure on top of them and iterate over that.

Example:

```
mapping(address=>Voter) info;
```

14. Modifier: Modifiers are used to easily change the behavior of a function. They can automatically check conditions before executing the functions.

```
modifier ownerOn() {
      require(msg.sender==owner);
      _;
   }
function temaAF(address _address) public {
      require(!info[_address].voted, "already voted person"); //If already not vote
      require(info[_address].authorized, "You Have No Right for Vote");
      info[_address].voted = true;
      teamA++;
      totalVotes++;
    }
```

Explanation:

```
require(!info[_address].voted, "already voted person");
```

Firstly we need to verify that person is Voted Or Not. If Person is voted then stop to proceed in the code otherwise proceed rest of code.

E-Voting App
A simple E-voting Decentralised App using the Ethereum Block chain, Solidity and the MERN(MongoDB, Express.js, ReactJS, Node.js) stack.
Ethereum is an open source, public, block chain-based distributed computing platform and operating system featuring smart contract functionality.

About the D-App
The E-Voting app has 2 main users:

1. Admin
2. Voter

Admin can create an election and add candidates to the Ethereum Block chain
Users(Voters) can select an election and vote for a candidate of their choice

Dependencies

- Node.js
- npm
- React.js
- Web13.js
- Ganache-cli
- Truffle
- Solidity
- MongoDB
- Metamask

Getting Started

To deploy the Smart Contract

1. Install Ganache and create a workspace.
2. Install Truffle npm package globally by running npm install -g truffle.
3. Run truffle migrate --reset from the command line to deploy the smart contract to the blockchain.
4. Download Metamask Chrome extension for the browser to help interaction between the application and the blockchain.

To run react development server
cd blockchain
npm start

To run node server
cd server
npm run dev

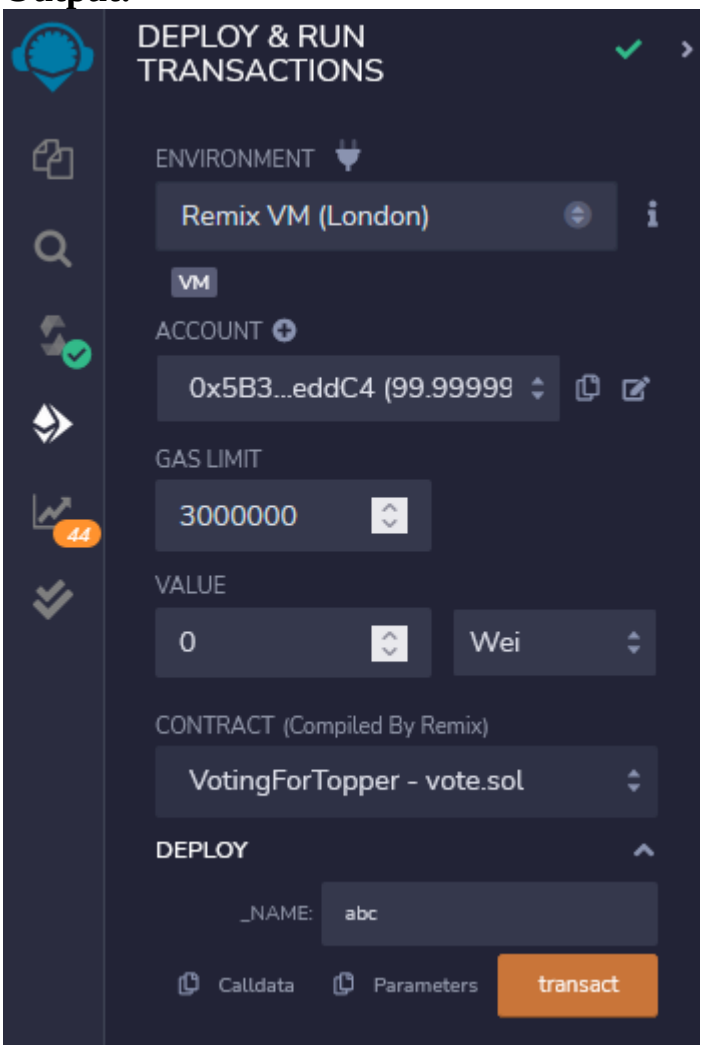
Implementation & result.

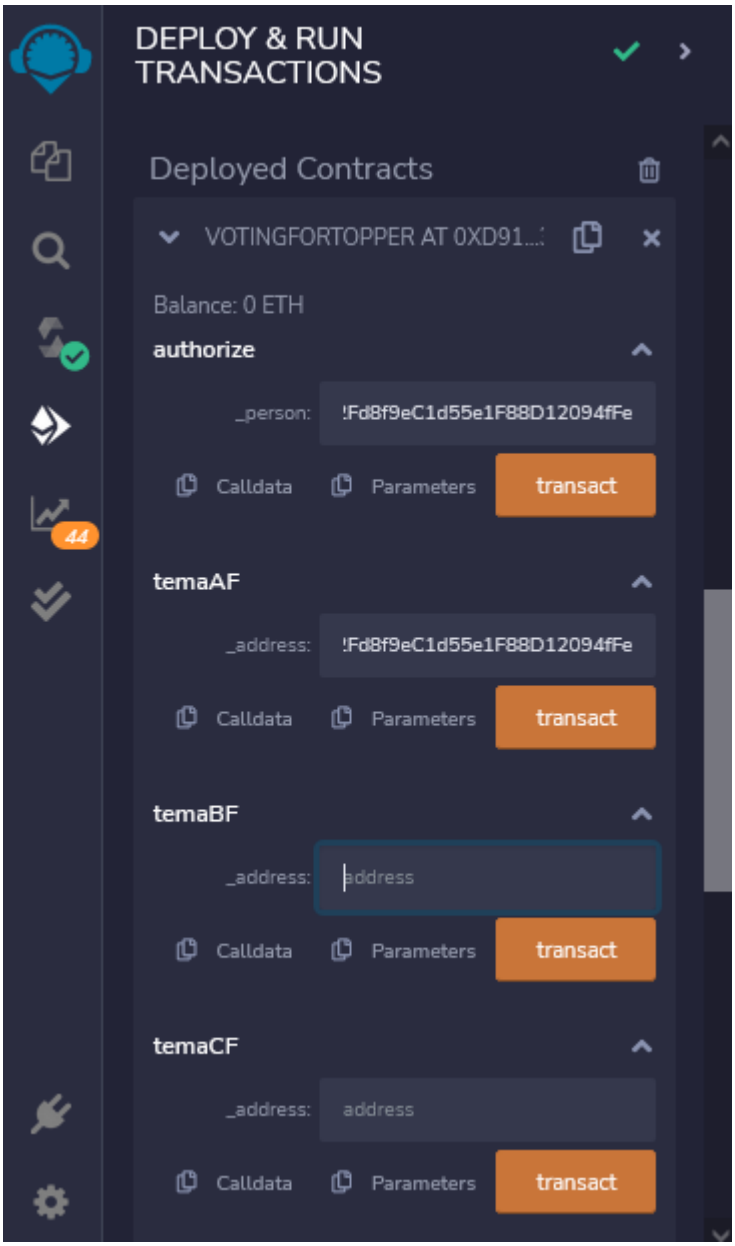
```
pragmasolidity0.6.6;

// Smart Contract for the Voting application
contractVotingForTopper{

      // Refer to the owner
      address  owner;

      // Declaring the public variable 'purpose'
      // to demonstrate the purpose of voting
      stringpublic purpose;

      // Defining a structure with boolean
      // variables authorized and voted
      struct Voter{
          bool authorized;
          bool voted;
      }

      // Declaring the unsigned integer
      // variables totalVotes, and for the
      //3 teams- A,B, and C
      uint  totalVotes;
```

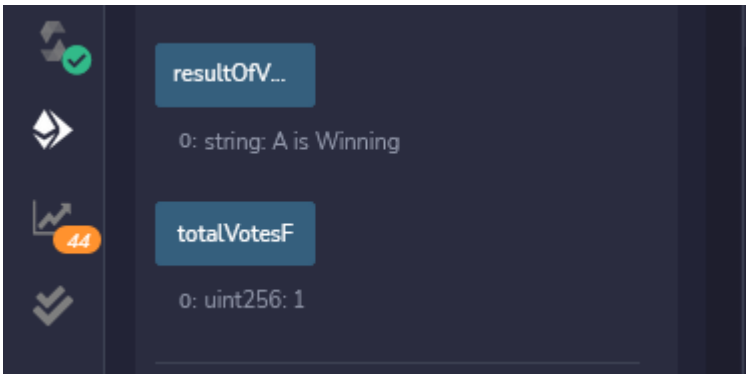
```
    uintteamA;
    uint  teamB;
    uintteamC;

    // Creating a mapping for the total Votes
    mapping(address=>Voter) info;

    // Defining a constructor indicating
    // the purpose of voting
    constructor(
    stringmemory _name)public{
      purpose= _name;
      owner=msg.sender;
    }

    // Defining a modifier to
    // verify the ownership
    modifierownerOn(){
      require(msg.sender==owner);
      _;
  }

    // Defining a function to verify
    // the person is voted or not
    function authorize(
    address _person)ownerOnpublic{
      info[_person].authorized=true;

  }

    // Defining a function to check and
    // skip the code if the person is already
    // voted else allow to vote and
    // calculate totalvotes for team A
    functiontemaAF(address _address)public{
      require(
      !info[_address].voted,
      "already voted person");
      require(
      info[_address].authorized,
      "You Have No Right for Vote");
      info[_address].voted =true;
      teamA++;
      totalVotes++;
    }

    // Defining a function to check
    // and skip the code if the person
    // is already voted else allow to vote
    // and calculate totalvotes for team B
    functiontemaBF(address _address)public{
     require(
     !info[_address].voted,
     "already voted person");
      require(
      info[_address].authorized,
      "You Have No Right for Vote");
```

```
        teamB++;
        info[_address].voted =true;
        totalVotes++;
    }

    // Defining a function to check
    // and skip the code if the person
    // is already voted else allow to vote
    // and calculate totalvotes for team C
    functiontemaCF(address _address)publicreturns(
    stringmemory){
        require(
        !info[_address].voted,
        "already voted person");
        require(
        info[_address].authorized,
        "You Have No Right for Vote");
        info[_address].voted =true;
        teamC++;
        totalVotes++;
        return("Thanks for Voting");
    }

    functiontotalVotesF()publicviewreturns(uint){
        returntotalVotes;
    }

    // Defining a function to announce
    // the result of voting and
    // the name of the winning team
    functionresultOfVoting()publicviewreturns(
    stringmemory){
        if(teamA>teamB){
            if(teamA>teamC){
                return"A is Winning";
            }
            elseif(teamC>teamA){
                return"C is Winning";}}
        elseif(teamB>teamC){
            return"B is Winning";
        }
        elseif(
        teamA==teamB&&teamA==teamC||teamB==teamC){
            return"No One is Winning";
        }
    }
}
```

Output:


DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (London)
VM
ACCOUNT
0x5B3...eddC4 (99.99999
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT (Compiled By Remix)
VotingForTopper - vote.sol
DEPLOY
_NAME:
abc
Calldata
Parameters
transact





Conclusion:-We have successfully Develop a Blockchain based application dApp (de-centralized app) for e-voting system.

Mini Project

Title of 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.

Objectives :

- To implement matrix multiplication using multithreading .
- To Analyze and compare the performance .

Input :

Matrix A
3 7 3 6
9 2 0 3
0 2 1 7
2 2 7 9

Matrix B
6 5 5 2
1 7 9 6
6 6 8 9
0 3 5 2

Output :

Multiplication of A and B

43 100 132 87

56 68 78 36

8 41 61 35

36 93 129 97

Implementation Code :

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 4
#define MAX_THREAD 4

int matA[MAX][MAX];
int matB[MAX][MAX];
int matC[MAX][MAX];
int step_i = 0;

void* multi(void* arg)
{
    int i = step_i++;

    for (int j = 0; j < MAX; j++)
      for (int k = 0; k < MAX; k++)
        matC[i][j] += matA[i][k] * matB[k][j];
}

int main()
{
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++) {
            matA[i][j] = rand() % 10;
            matB[i][j] = rand() % 10;
        }
    }

    cout << endl
         << "Matrix A" << endl;
```

```
for (int i = 0; i < MAX; i++) {
    for (int j = 0; j < MAX; j++)
        cout << matA[i][j] << " ";
    cout << endl;
}

cout << endl
     << "Matrix B" << endl;
for (int i = 0; i < MAX; i++) {
    for (int j = 0; j < MAX; j++)
        cout << matB[i][j] << " ";
    cout << endl;
}

pthread_t threads[MAX_THREAD];

for (int i = 0; i < MAX_THREAD; i++) {
    int* p;
    pthread_create(&threads[i], NULL, multi, (void*)(p));
}

for (int i = 0; i < MAX_THREAD; i++)
    pthread_join(threads[i], NULL);

cout << endl
     << "Multiplication of A and B" << endl;
for (int i = 0; i < MAX; i++) {
    for (int j = 0; j < MAX; j++)
        cout << matC[i][j] << " ";
    cout << endl;
}
return 0;}
```

Result :

```
main.cpp:18:1: warning: no return statement in function returning non-void [-Wreturn-type]
   18 | }
      | ^

Matrix A
3 7 3 6
9 2 0 3
0 2 1 7
2 2 7 9

Matrix B
6 5 5 2
1 7 9 6
6 6 8 9
0 3 5 2

Multiplication of A and B
43 100 132 87
56 68 78 36
8 41 61 35
56 93 129 97
```

Conclusion :

Thus, we have successfully created a Console based application for matrix multiplication using Multi-Threading. Also the analysis and comparison is evaluated.