Group A

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 Nonrecursive 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?

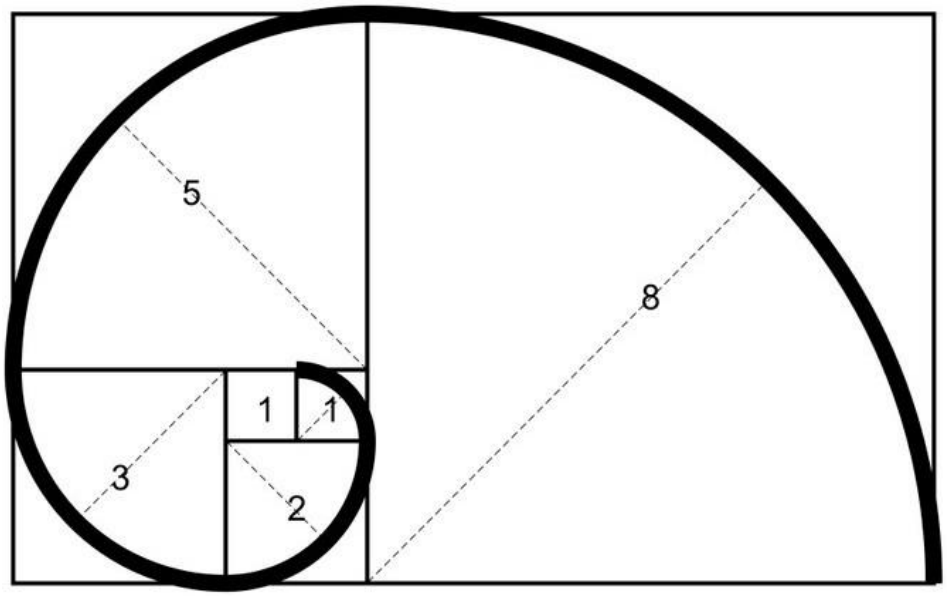
- The Fibonacci series is the sequence of numbers (also called Fibonacci numbers), where every number is the sum of the preceding two numbers, such that the first two terms are '0' and '1'.
- In some older versions of the series, the term '0' might be omitted. A Fibonacci series can thus be given as, 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:

F_n	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)

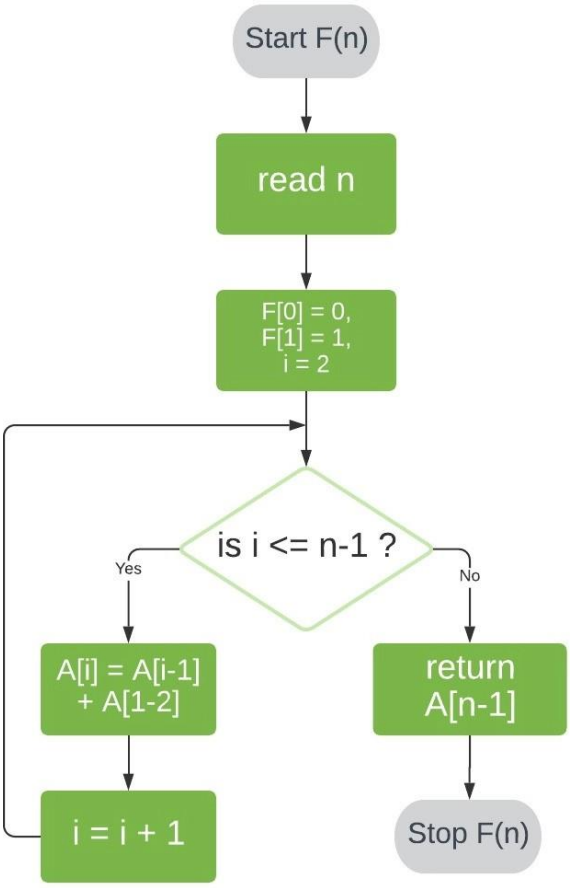
A simple method that is a direct recursive implementation of mathematical recurrence relation is given above.

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.

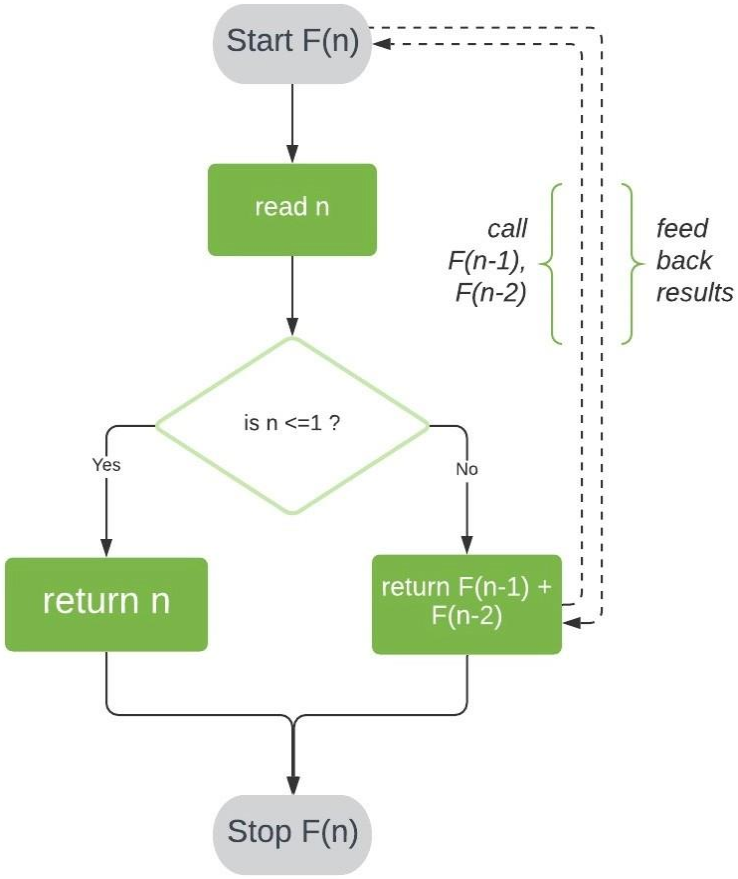
To evaluate *F*(*n*) for *n* > 1, we can reduce our problem into two smaller problems of the same kind: *F*(*n*-1) and *F*(*n*-2). We can further reduce *F*(*n*-1) and *F*(*n*-2) to *F*((*n*-1)-1) and *F*((*n*-1)-2); and *F*((*n*-2)-1) and *F*((*n*-2)-2), respectively.

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

Assignment Question

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

Reference link

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

Program

```
def recur_fibo(n):
   if n <= 1:
       return n
   else:
       return(recur_fibo(n-1) + recur_fibo(n-2))
nterms = int(input("How many terms? "))
if nterms <= 0:
   print("Plese enter a positive integer")
else:
   print("Fibonacci sequence:")
   for i in range(nterms):
       print(recur_fibo(i))
```

Group A

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.

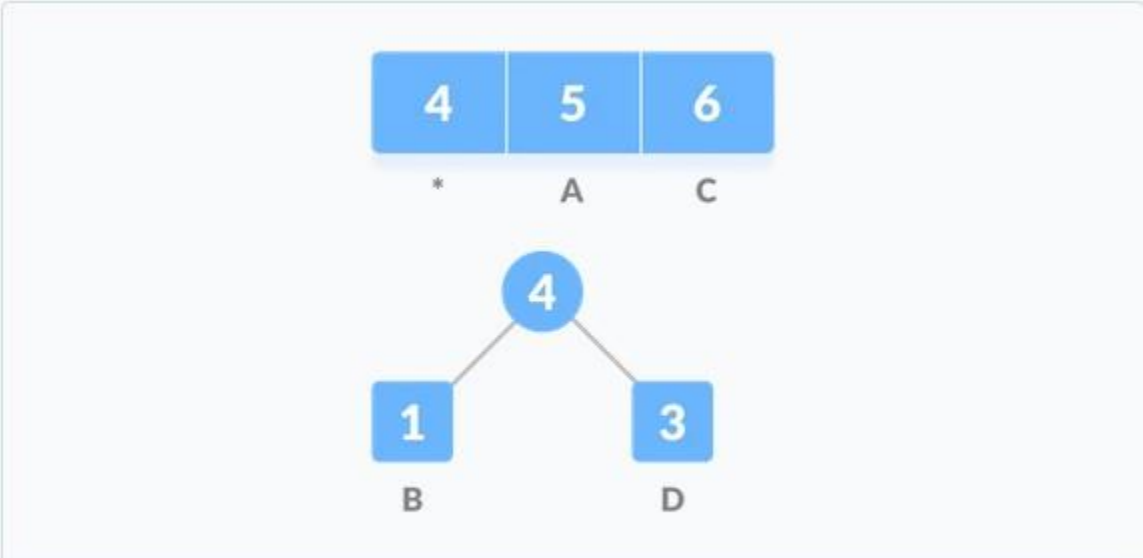
- Huffman coding first creates a tree using the frequencies of the character and then generates code for each character.
- Once the data is encoded, it has to be decoded. Decoding is done using the same tree.
- Huffman Coding prevents any ambiguity in the decoding process using the concept of **prefix 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.

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.

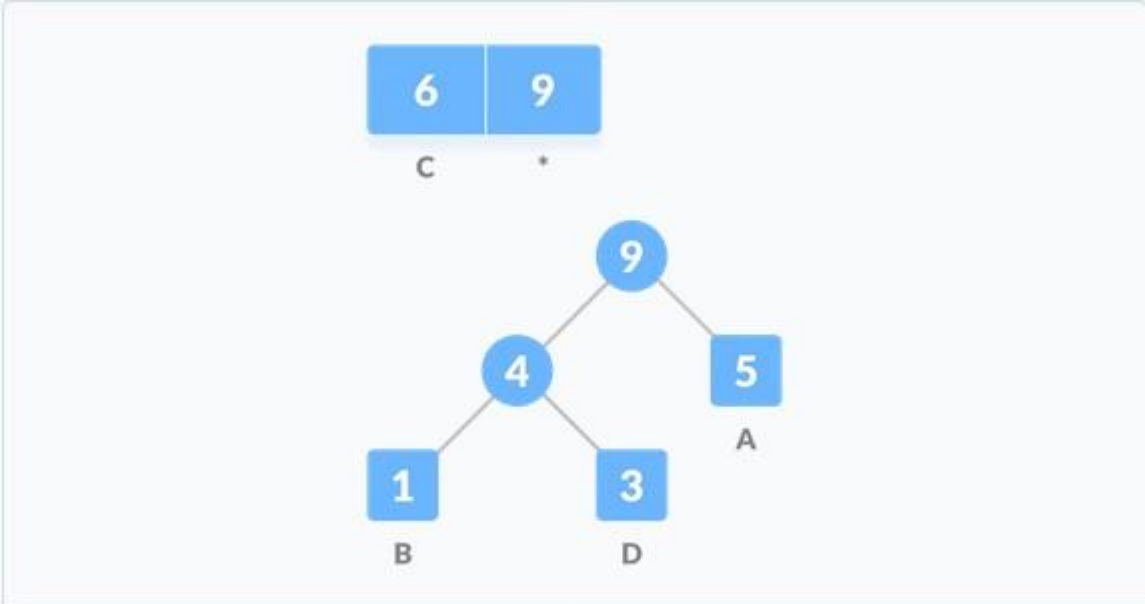


Getting the sum of the least numbers

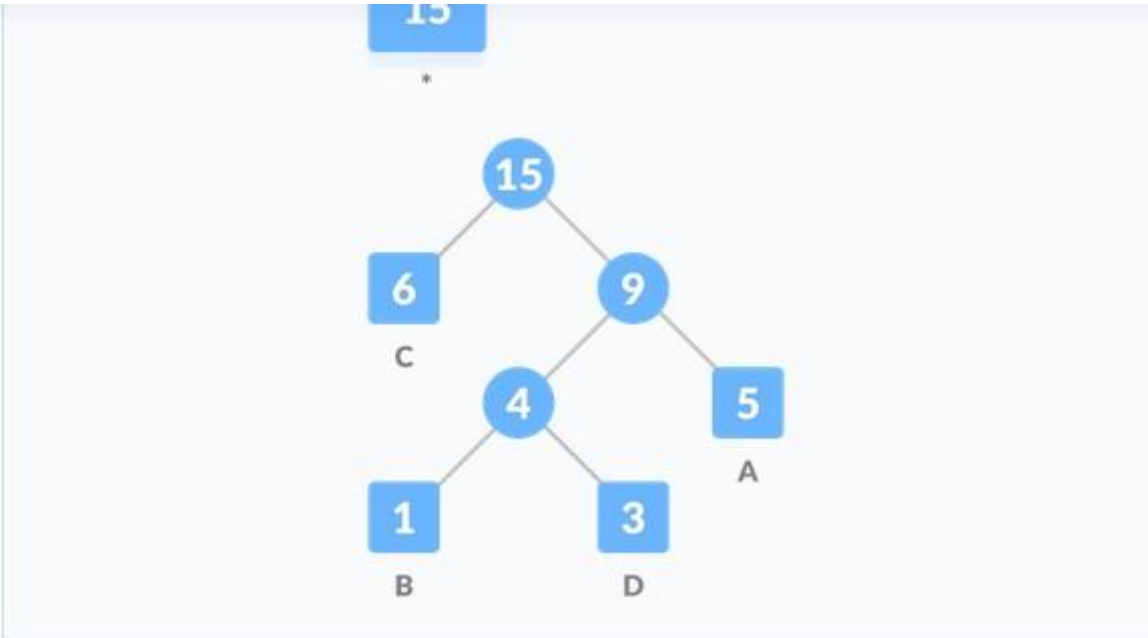
5. Remove these two minimum frequencies from Q and add the sum into the list of frequencies (* denote the internal nodes in the figure above).

6. Insert node z into the tree.

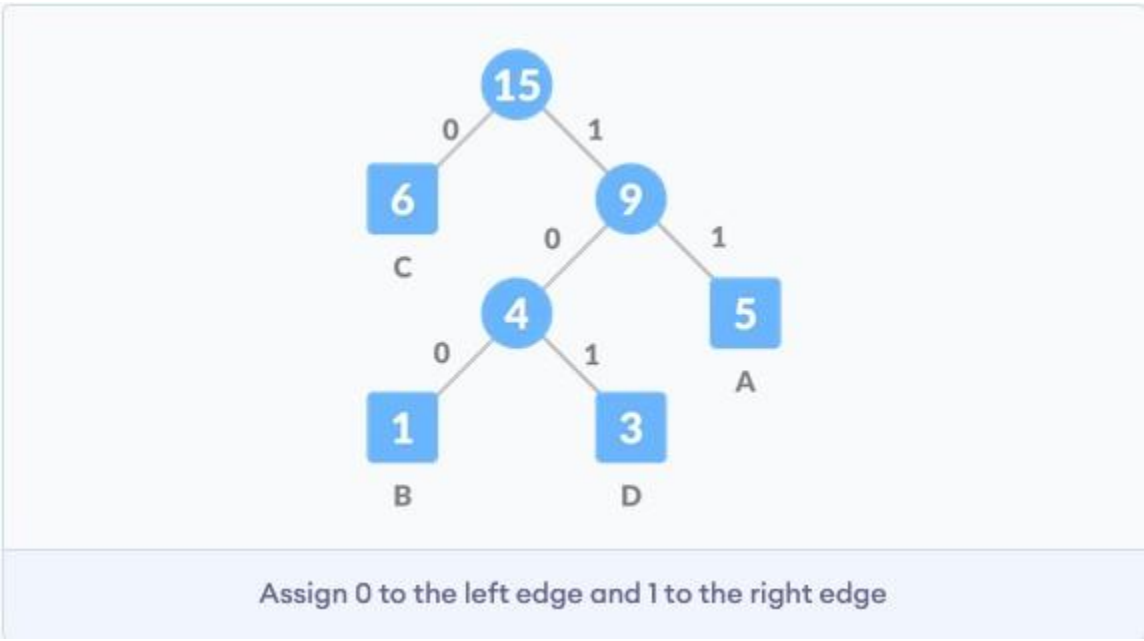
7. Repeat steps 3 to 5 for all the characters.



Repeat steps 3 to 5 for all the characters.



8. For each non-leaf node, 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:

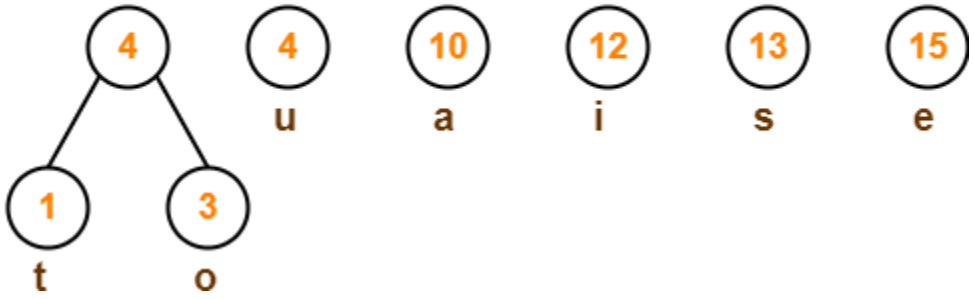
A file contains the following characters with the frequencies as shown. If Huffman Coding is used for data compression, determine-

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

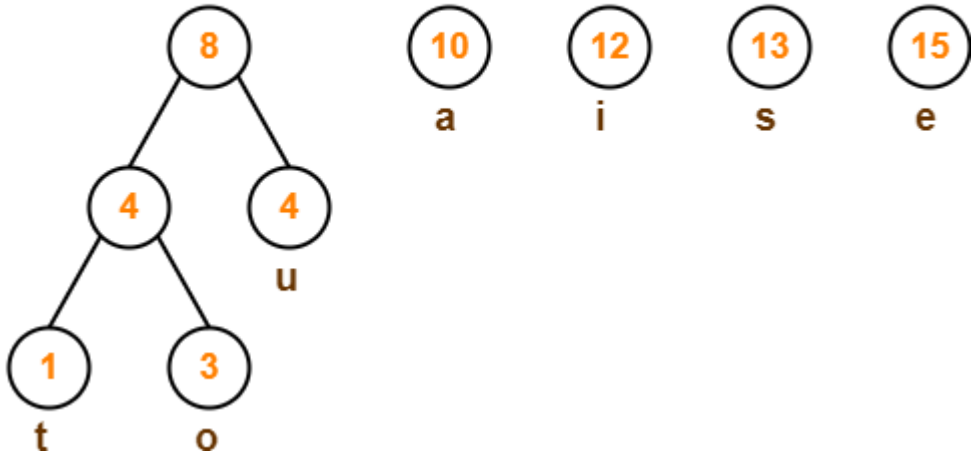
Characters	Frequencies
a	10
e	15
i	12
o	3
u	4
s	13
t	1

Step-01:

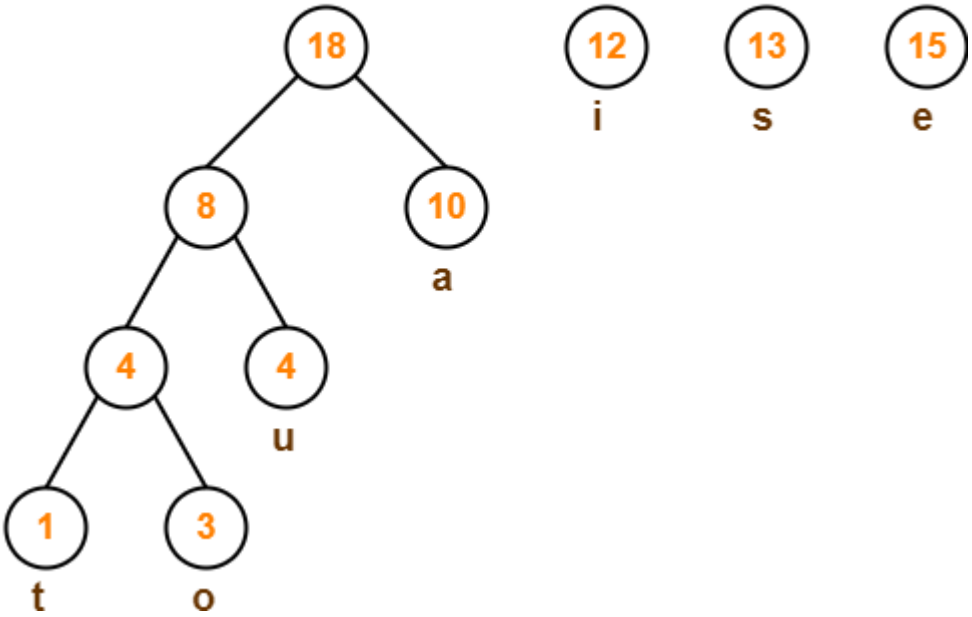
Step-02:



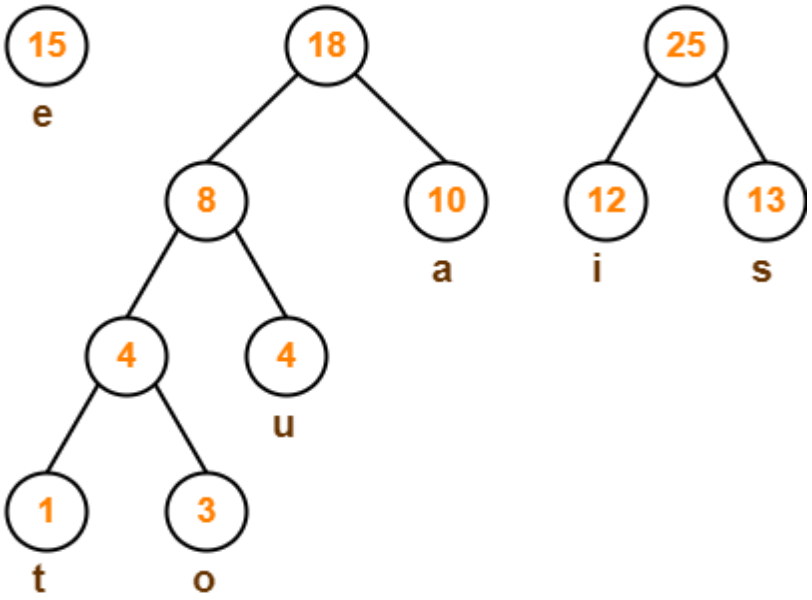
Step-03:

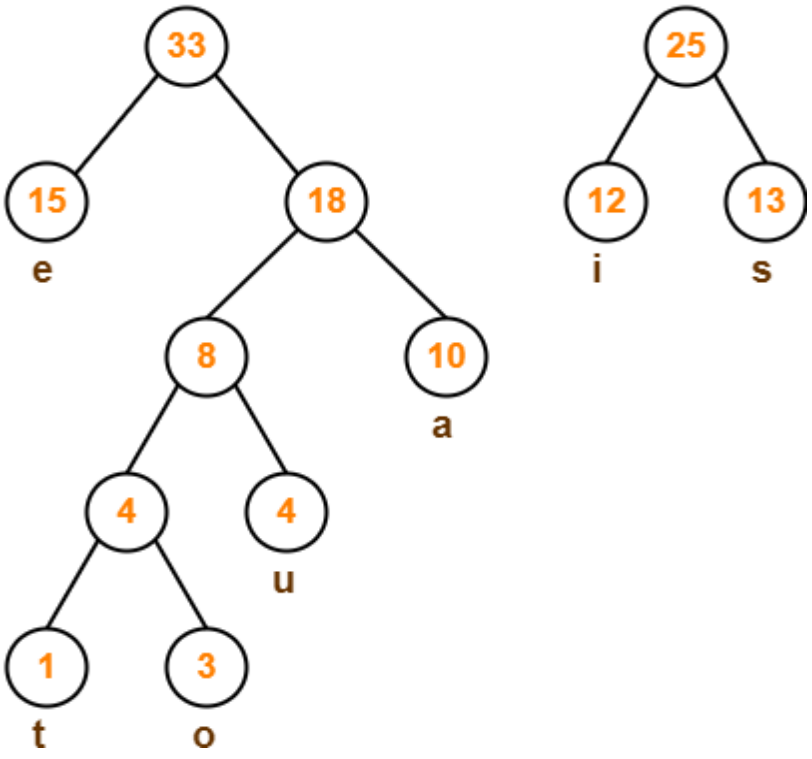


Step-04:

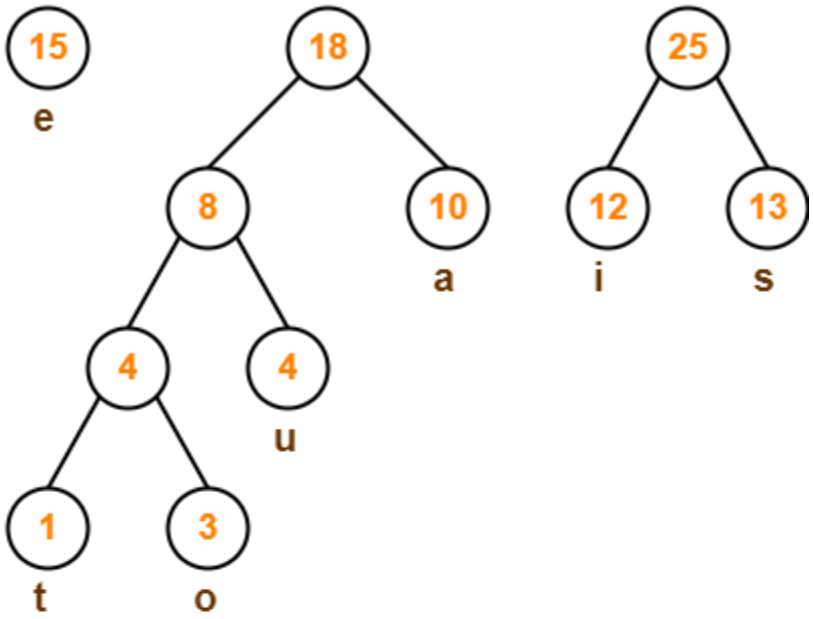


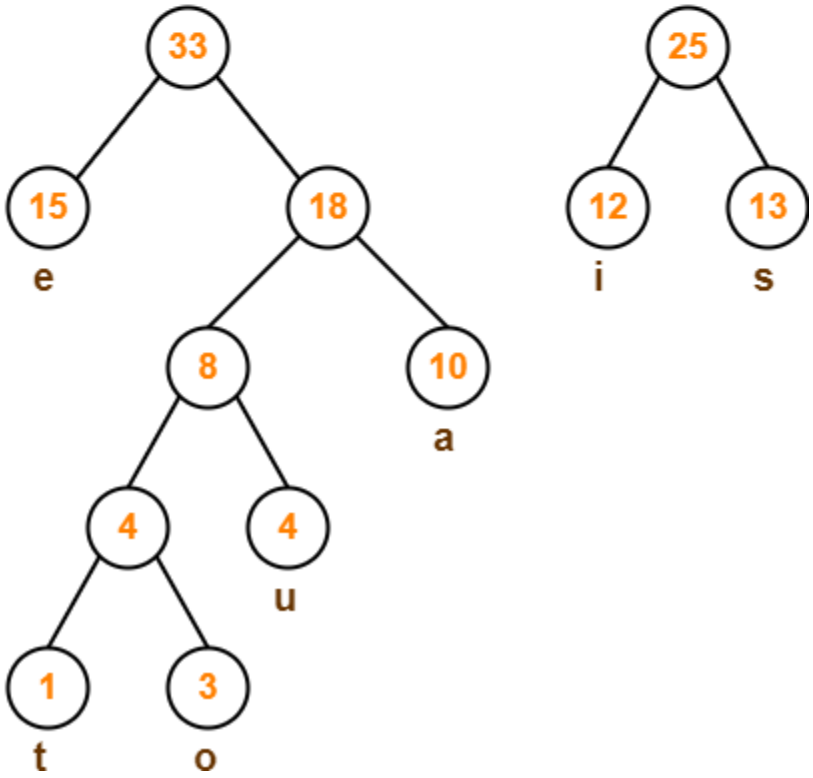
Step-06:


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

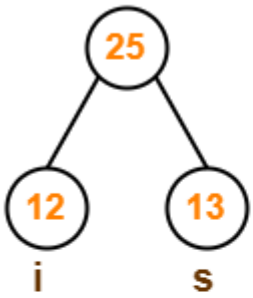

33
15
e
18
8
10
a
25
12
13
i
s
4
4
u
1
3
t
o

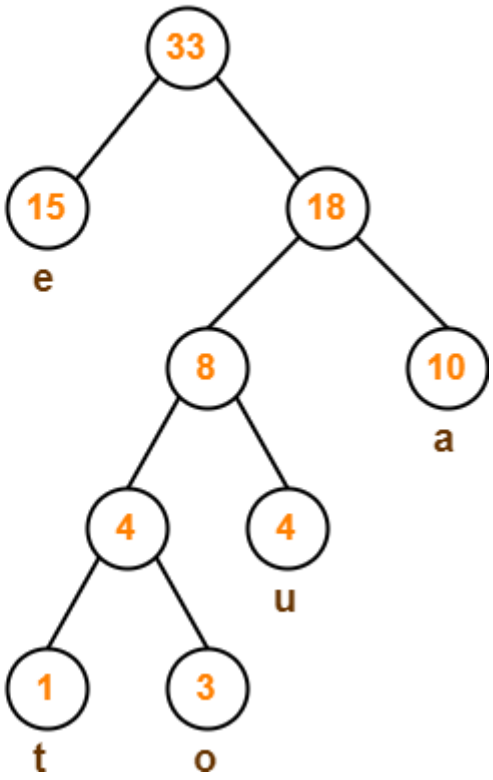
Step-06:

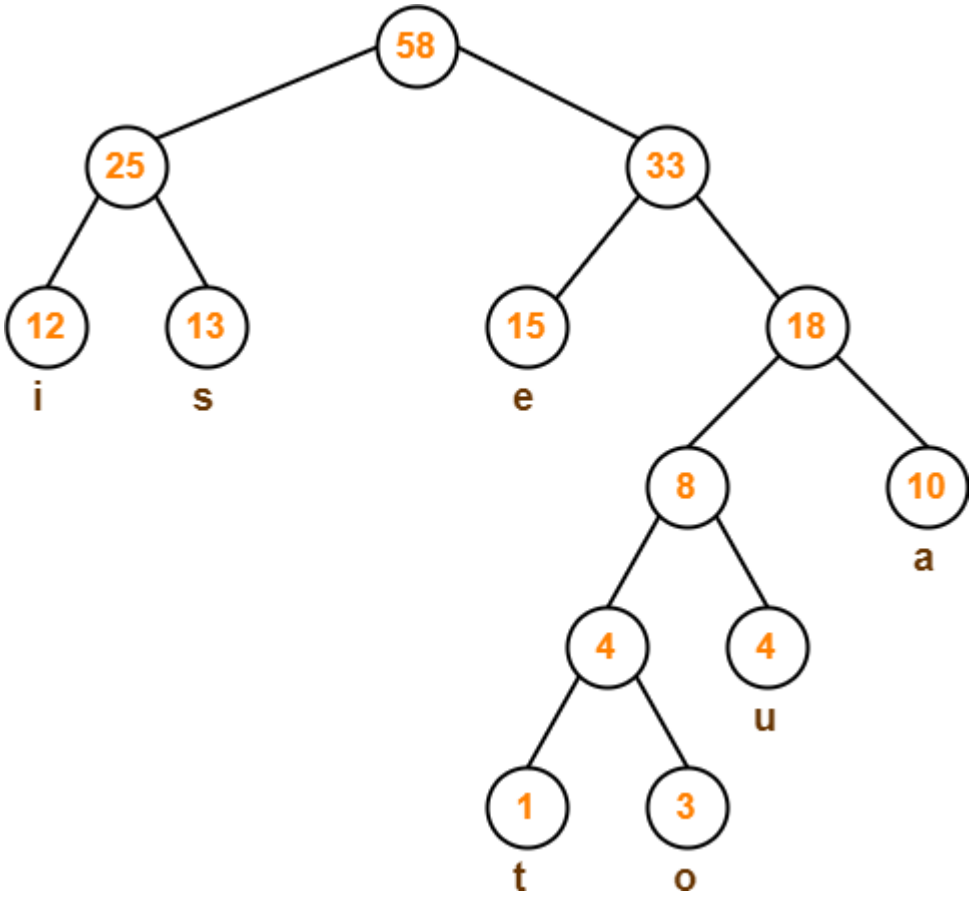

15
e
18
8
10
a
25
12
13
i
s
4
4
u
1
3
t
o


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

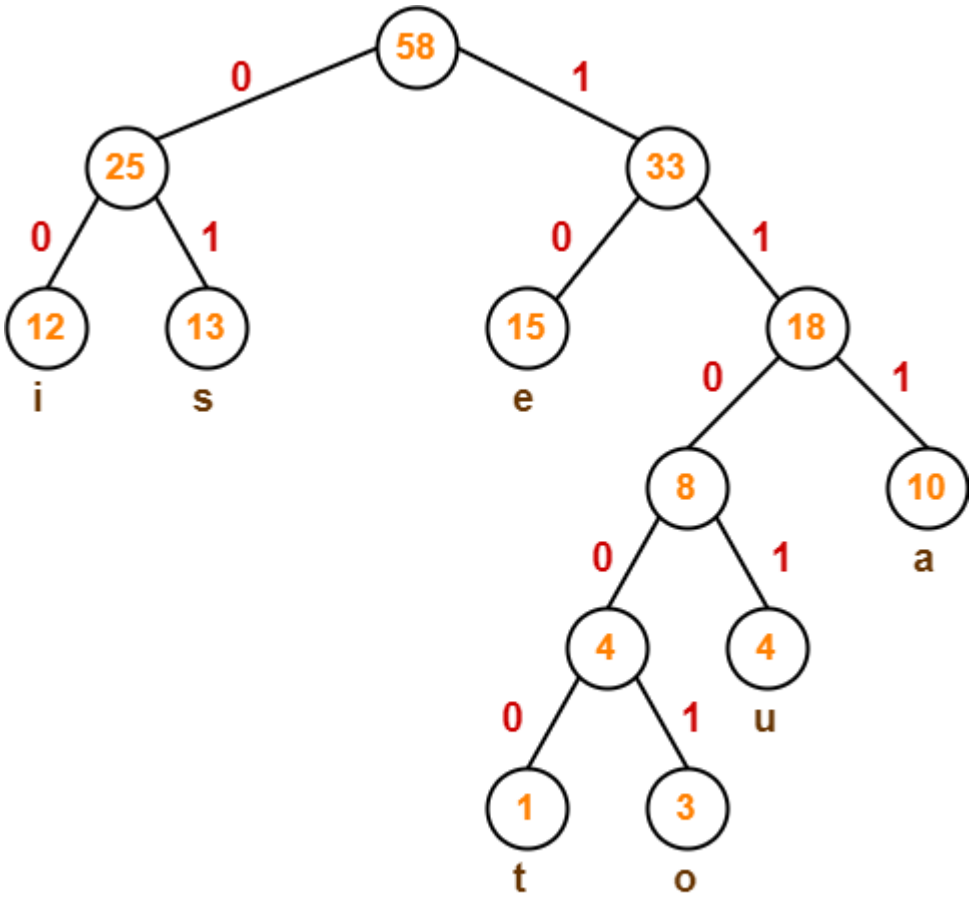
Step-07:







After assigning weight to all the edges, the modified Huffman Tree is-



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

Assignment Question

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

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/

Program

```
string = 'BCAADDDCCACACAC'
class NodeTree(object):

    def __init__(self, left=None, right=None):
        self.left = left
        self.right = right

    def children(self):
        return (self.left, self.right)

    def nodes(self):
        return (self.left, self.right)

    def __str__(self):
        return '%s_%s' % (self.left, self.right)


def huffman_code_tree(node, left=True, binString=''):
    if type(node) is str:
        return {node: binString}
    (l, r) = node.children()
    d = dict()
    d.update(huffman_code_tree(l, True, binString + '0'))
    d.update(huffman_code_tree(r, False, binString + '1'))
    return d

freq = {}
for c in string:
    if c in freq:
        freq[c] += 1
    else:
        freq[c] = 1

freq = sorted(freq.items(), key=lambda x: x[1], reverse=True)

nodes = freq

while len(nodes) > 1:
    (key1, c1) = nodes[-1]
    (key2, c2) = nodes[-2]
    nodes = nodes[:-2]
    node = NodeTree(key1, key2)
    nodes.append((node, c1 + c2))
```

```
    nodes = sorted(nodes, key=lambda x: x[1], reverse=True)

huffmanCode = huffman_code_tree(nodes[0][0])

print(' Char | Huffman code ')
print('----------------------')
for (char, frequency) in freq:
    print(' %-4r |%12s' % (char, huffmanCode[char]))
```

Group A

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-

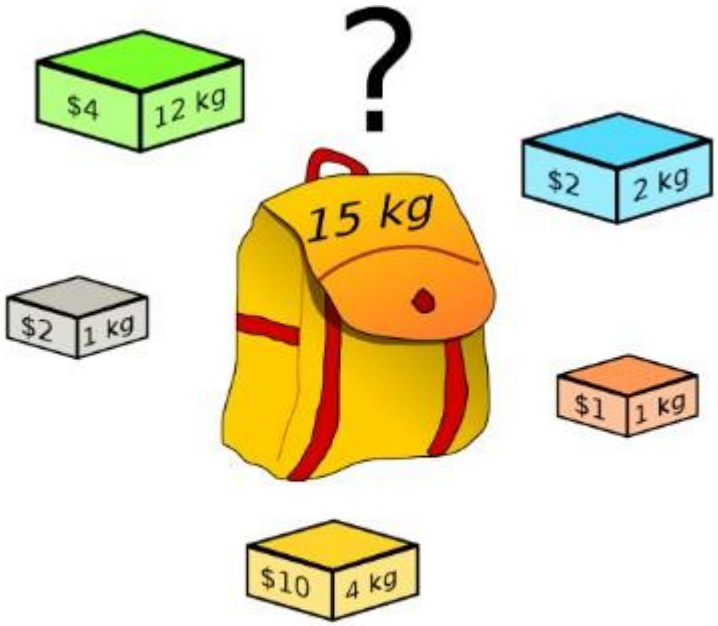
- A knapsack (kind of shoulder bag) with limited weight capacity.

- 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

Assignment Question

1. **What is Greedy Approach?**
2. **Explain concept of fractional knapsack**
3. **Difference between Fractional and 0/1 Knapsack**
4. **Solve one example based on Fractional knapsack(Other than Manual)**

Reference link

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

Program

```
class Item:
    def __init__(self, value, weight):
        self.value = value
        self.weight = weight

def fractionalKnapsack(W, arr):

    arr.sort(key=lambda x: (x.value/x.weight), reverse=True)

    finalvalue = 0.0

    for item in arr:
        if item.weight <= W:
            W -= item.weight
            finalvalue += item.value

        else:
            finalvalue += item.value * W / item.weight
            break

    return finalvalue


if __name__ == "__main__":

    W = 50
    arr = [Item(60, 10), Item(100, 20), Item(120, 30)]

    max_val = fractionalKnapsack(W, arr)
    print(max_val)
```

Group A

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

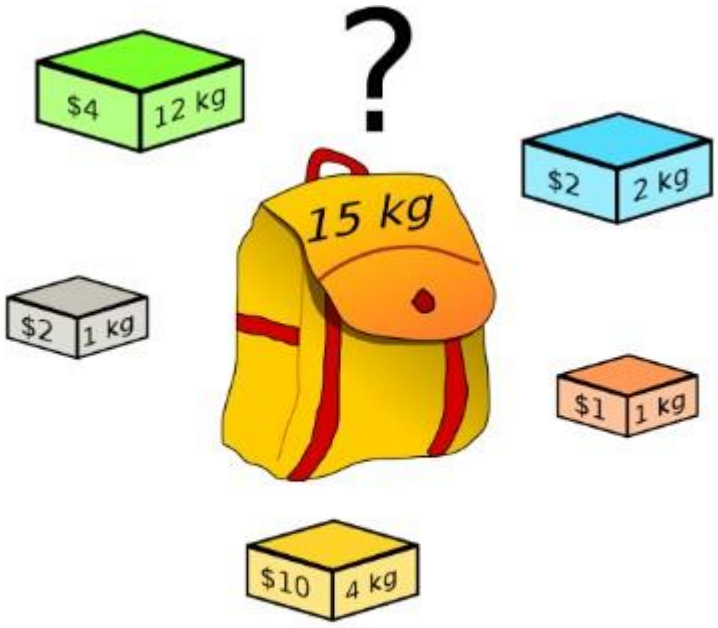
You are given the following-

- 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 0^{th} row and 0^{th} 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 0^{th} row and 0^{th} 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 approch

Assignment Question

1. **What is Dynamic Approach?**
2. **Explain concept of 0/1 knapsack**
3. **Difference between Dynamic and Branch and Bound Approach.Which is best?**
4. **Solve one example based on 0/1 knapsack(Other than Manual)**

Reference link

- https://www.gatevidyalay.com/0-1-knapsack-problem-using-dynamic-programming-approach/
- https://www.youtube.com/watch?v=mMhC9vuA-70
- https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_fractional_knapsack.htm

Program

```
def knapSack(W, wt, val, n):
    K = [[0 for x in range(W + 1)] for x in range(n + 1)]

    for i in range(n + 1):
        for w in range(W + 1):
            if i == 0 or w == 0:
                K[i][w] = 0
            elif 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]

val = [60, 100, 120]
wt = [10, 20, 30]
W = 50
n = len(val)
print(knapSack(W, wt, val, n))
```

Group A

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.

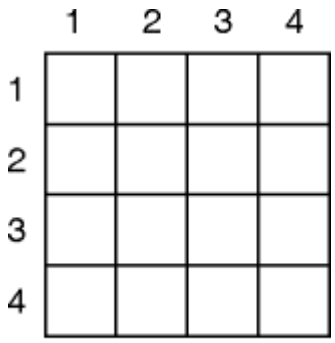
b) If placing the queen in [row, column] leads to a solution then return true.

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)

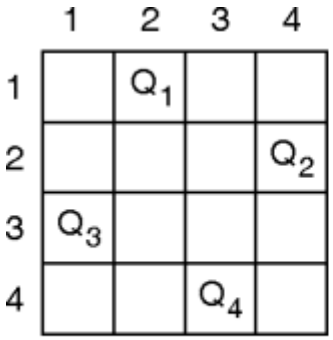


Fig. (a): Solution – 1

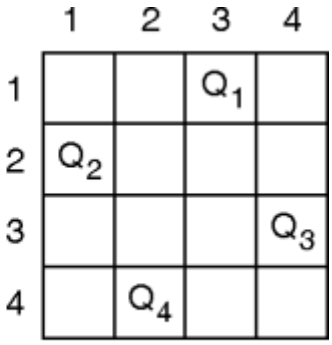
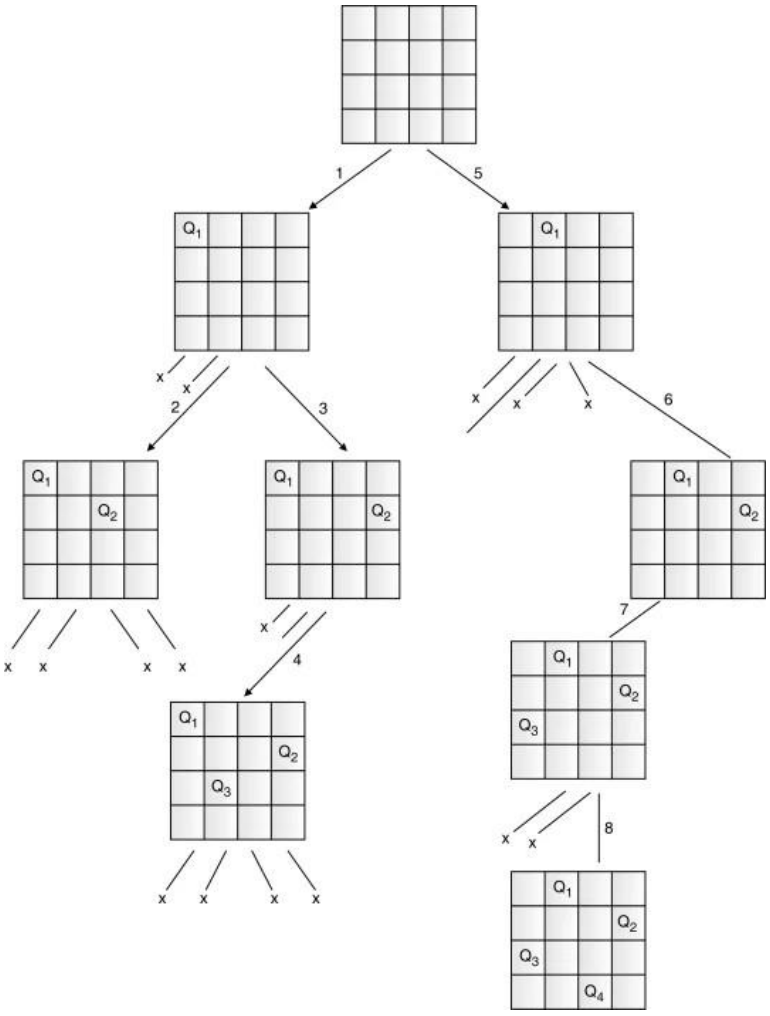


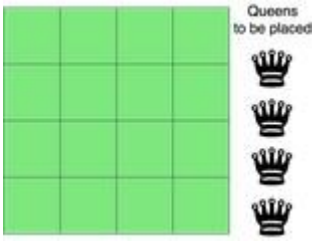
Fig. (b): Solution – 2

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

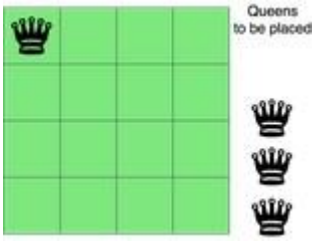


The solution of the 4-queen problem can be seen as four tuples (x_1, x_2, x_3, x_4), where x_i represents the column number of queen Q_i. Two possible solutions for the 4-queen problem are (2, 4, 1, 3) and (3, 1, 4, 2).

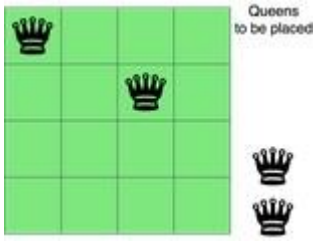
Explanation :

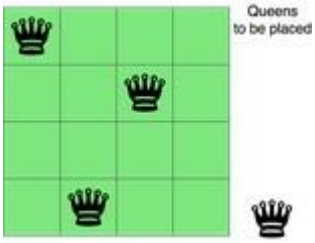


The above picture shows an NxN chessboard and we have to place N queens on it. So, we will start by placing the first queen.



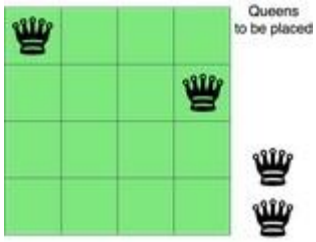
Now, the second step is to place the second queen in a safe position and then the third queen.



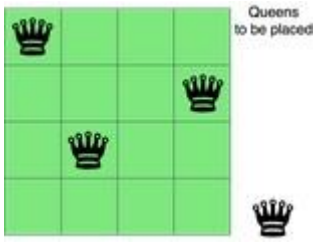


Now, you can see that there is no safe place where we can put the last queen. So, we will just change the position of the previous queen. And this is backtracking.

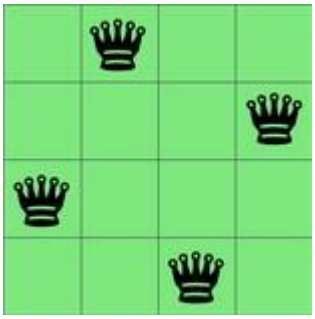
Also, there is no other position where we can place the third queen so we will go back one more step and change the position of the second queen.



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

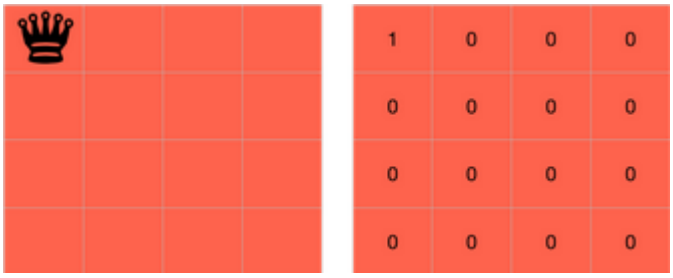


We will continue this process and finally, we will get the solution as shown below.

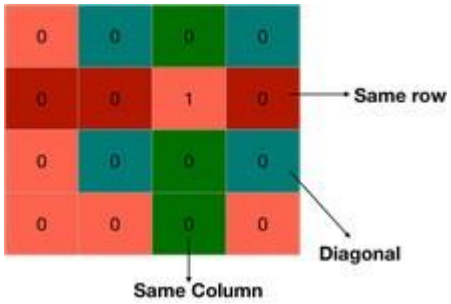


We need to check if a cell (i, j) is under attack or not. For that, we will pass these two in our function along with the chessboard and its size - IS-ATTACK(i, j, board, N).

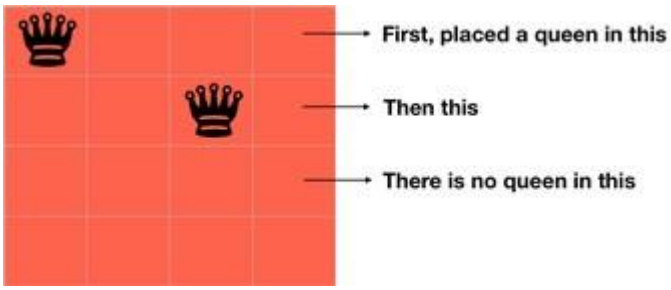
If there is a queen in a cell of the chessboard, then its value will be 1, otherwise, 0.



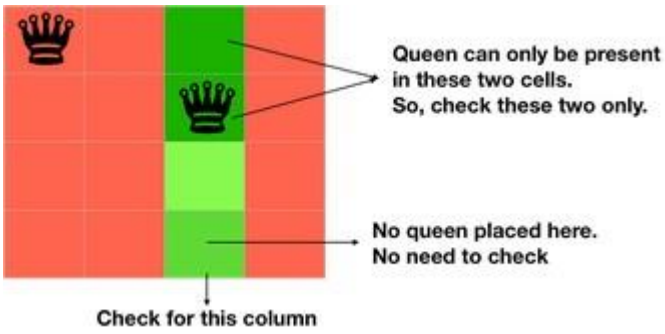
The cell (i,j) will be under attack in three condition - if there is any other queen in row i, if there is any other queen in the column j or if there is any queen in the diagonals.



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.



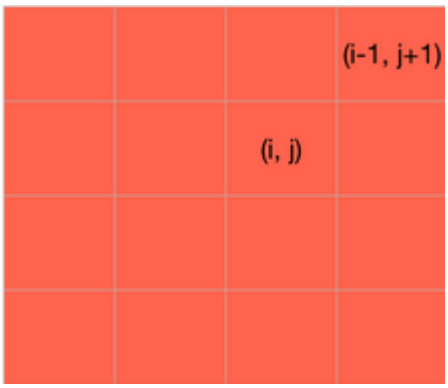
for k in 1 to i-1

if board[k][j]==1

return TRUE

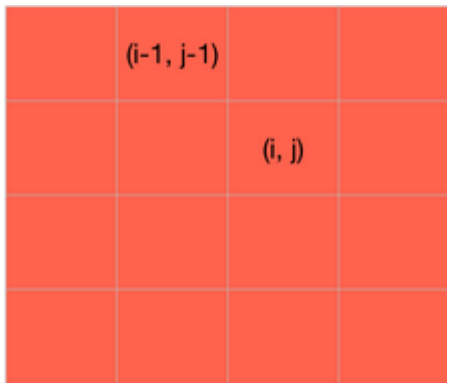
Now, we need to check for the diagonal. We know that all the rows below the row i are empty, so we need to check only for the diagonal elements which above the row i.

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

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

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

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

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

...

return FALSE

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

if n==0

return TRUE

for j in 1 to N

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

board[row][j] = 1

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

      return TRUE


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

return FALSE
```

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

Assignment Question

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

Reference link

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

Program

```
global N
N = 4

def printSolution(board):
    for i in range(N):
        for j in range(N):
            print (board[i][j],end=' ')
        print()



def isSafe(board, row, col):

    for i in range(col):
        if board[row][i] == 1:
            return False

    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    for i, j in zip(range(row, N, 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    return True

def solveNQUtil(board, col):

    if col >= N:
        return True


    for i in range(N):

        if isSafe(board, i, col):
            board[i][col] = 1

            if solveNQUtil(board, col + 1) == True:
                return True


            board[i][col] = 0

    return False

def solveNQ():
```

```
    board = [ [0, 0, 0, 0],
            [0, 0, 0, 0],
            [0, 0, 0, 0],
            [0, 0, 0, 0]
            ]

    if solveNQUtil(board, 0) == False:
        print ("Solution does not exist")
        return False

    printSolution(board)
    return True

solveNQ()
```

Group B

Assignment No:1

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

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

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

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

Objective of the Assignment:

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

Prerequisite:

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

Contents of the Theory:

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

Data Preprocessing:

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

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

Why do we need Data Preprocessing?

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

It involves below steps:

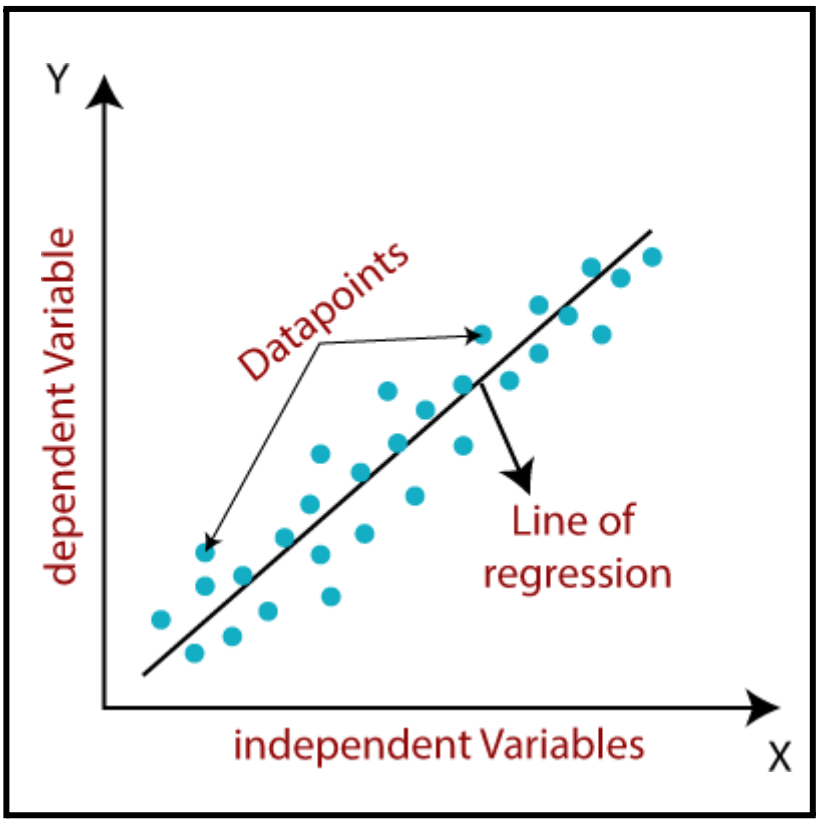
- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

Linear Regression:

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

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

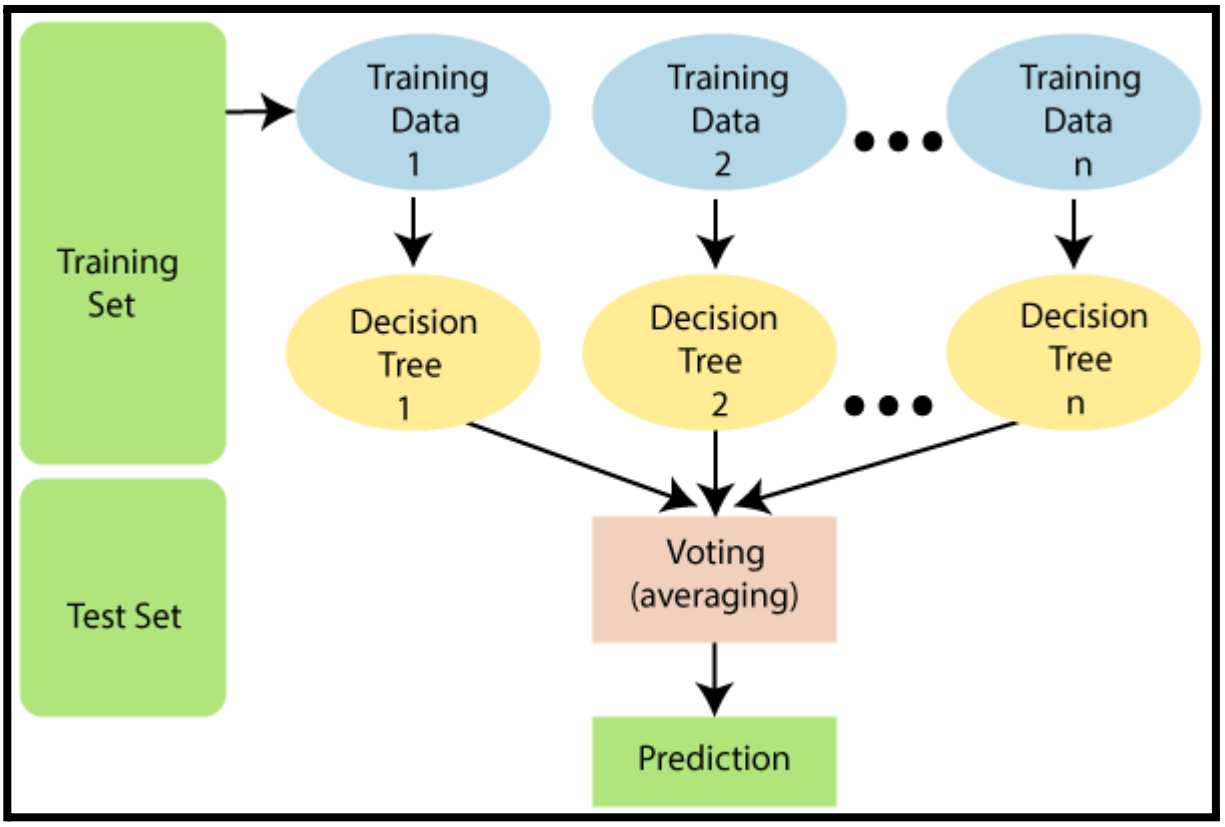
Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classi cation and Regression problems in ML. Itis based on the concept of**ensemble learning,**which is a process of*combining multiple classi ers to solve a complex problem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

As the name suggests,"***Random Forest is a classi er that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictiveaccuracy 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 overfitting.



Boxplot:

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

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

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

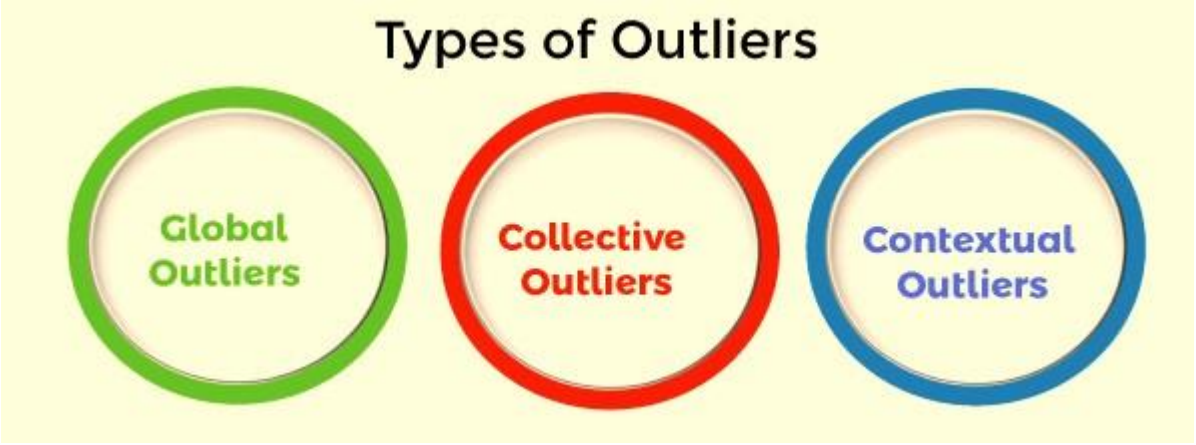
Here,

S.No	Parameter	Description

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

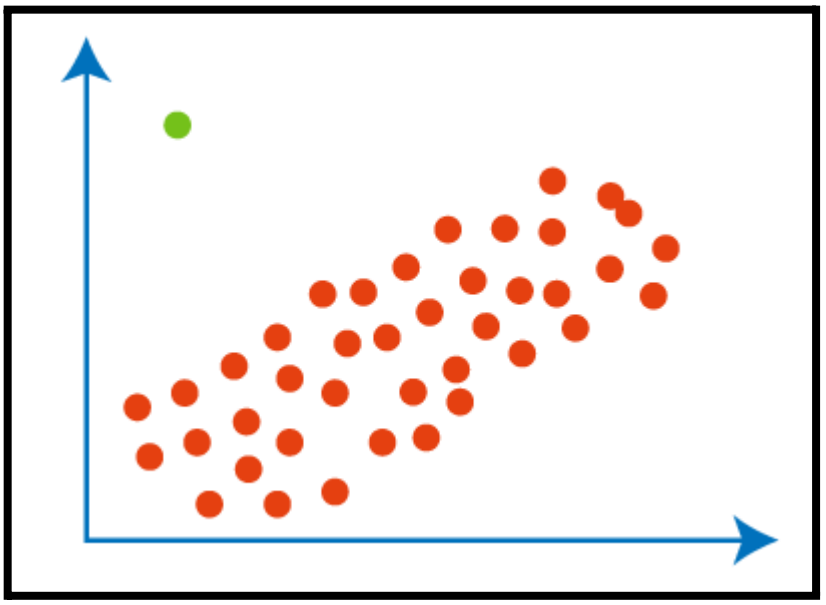
Outliers:

As the name suggests, "outliers" refer to the data points that exist outside of what is to be expected. The major thing about the outliers is what you do with them. If you are going to analyze any task to analyze data sets, you will always have some assumptions based on how this data is generated. If you find some data points that are likely to contain some form of error, then these are definitely outliers, and depending on the context, you want to overcome those errors. The data mining process involves the analysis and prediction of data that the data holds. In 1969, Grubbs introduced the first definition of outliers.



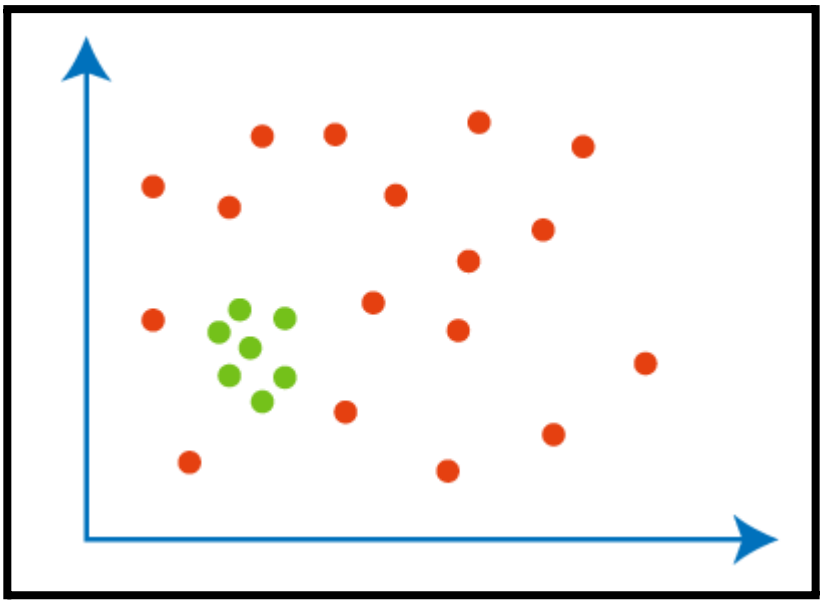
Global Outliers

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



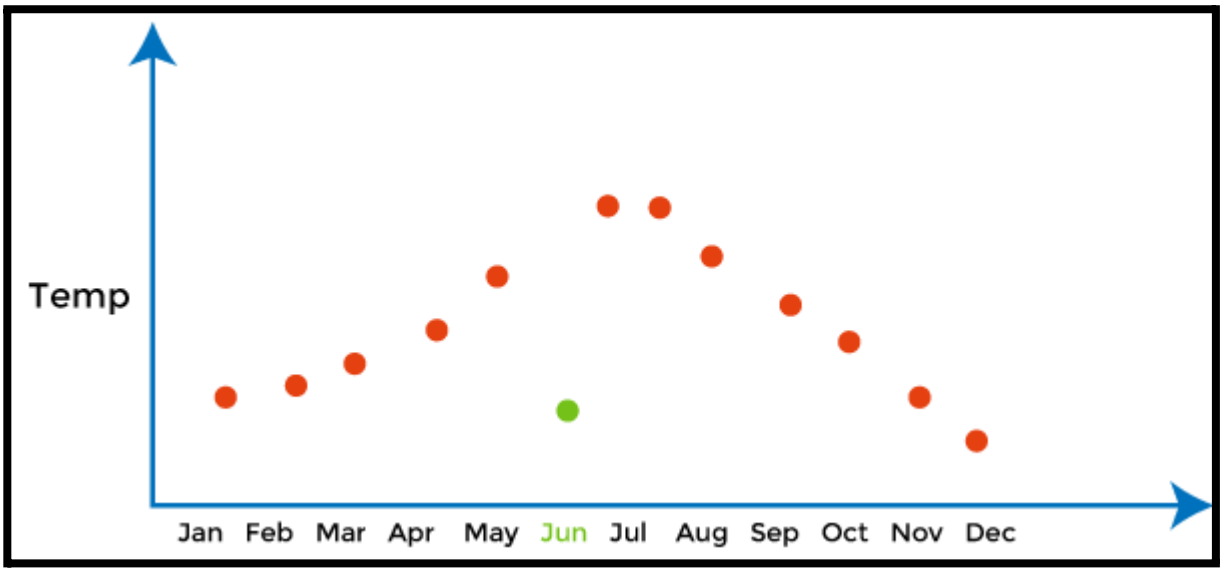
Collective Outliers

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



Contextual Outliers

As the name suggests, "Contextual" means this outlier introduced within a context. For example, in the speech recognition technique, the single background noise. Contextual outliers are also known as Conditional outliers. These types of outliers happen if a data object deviates from the other data points because of any speci□c condition in a given data set. As we know, there are two types of attributes of objects of data: contextual attributes and behavioral attributes. Contextual outlier analysis enables the users to examine outliers in different contexts and conditions, which can be useful in various applications. For example, A temperature reading of 45 degrees Celsius may behave as an outlier in a rainy season. Still, it will behave like a normal data point in the context of a summer season. In the given diagram, a green dot representing the low-temperature value in June is a contextual outlier since the same value in December is not an outlier.



Haversine:

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

Matplotlib:

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

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

Mean Squared Error;

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

Conclusion:

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

Assignment Questions:

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

Program

#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 (https://www.kaggle.com/datasets/yasserh/uber-fares-dataset)

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

In [ ]:
```
#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_latitude	pa
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.723217	
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.750325	
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.772647	
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.803349	
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.761247	

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 required
```

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	day	mon
0	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-73.999512	40.723217	1	19	7	
1	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-73.994710	40.750325	1	20	17	
2	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-73.962565	40.772647	1	21	24	
3	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-73.965316	40.803349	3	8	26	
4	16.0	2014-08-28 17:47:00+00:00	-73.925023	40.744085	-73.973082	40.761247	5	17	28	

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	year	dayofwe
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1	19	7	5	2015	
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1	20	17	7	2009	
2	12.9	-74.005043	40.740770	-73.962565	40.772647	1	21	24	8	2009	
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3	8	26	6	2009	
4	16.0	-73.925023	40.744085	-73.973082	40.761247	5	17	28	8	2014	

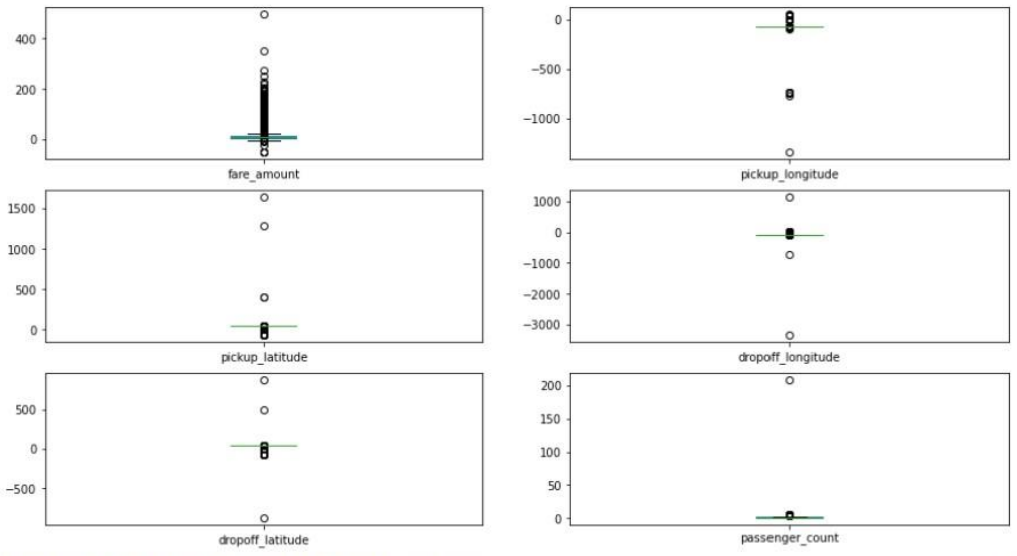
In [22]:
```
df.dtypes
```

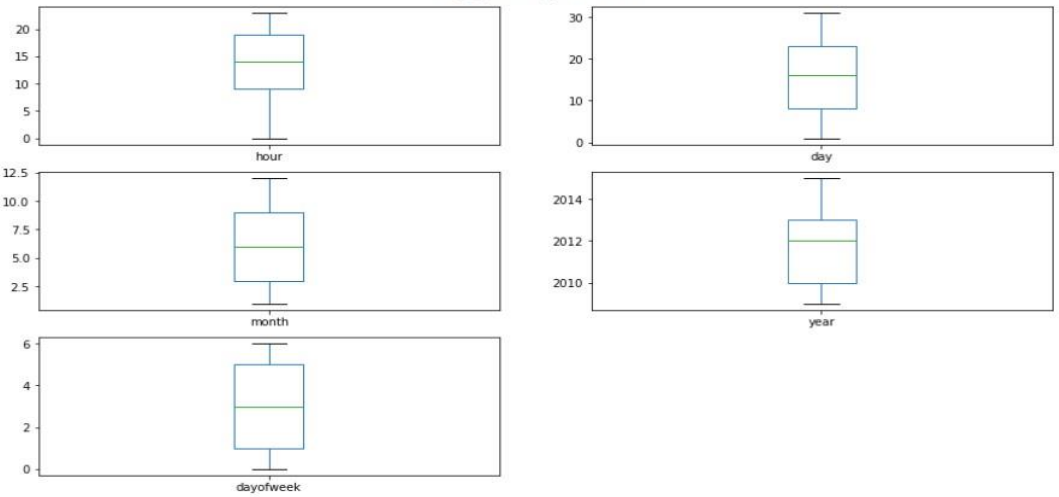
Out[22]:
```
fare_amount          float64
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
hour                   int64
day                    int64
month                  int64
year                   int64
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 the 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
```



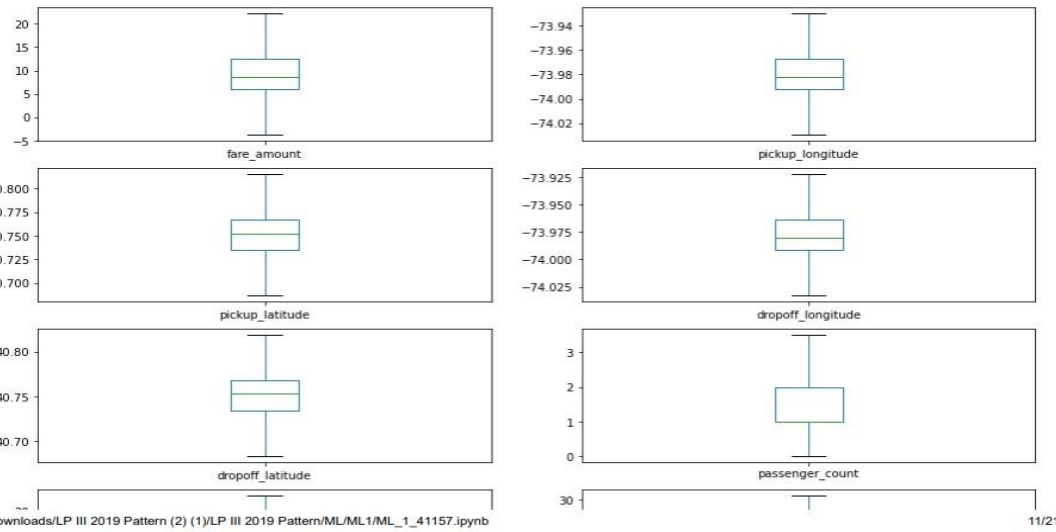


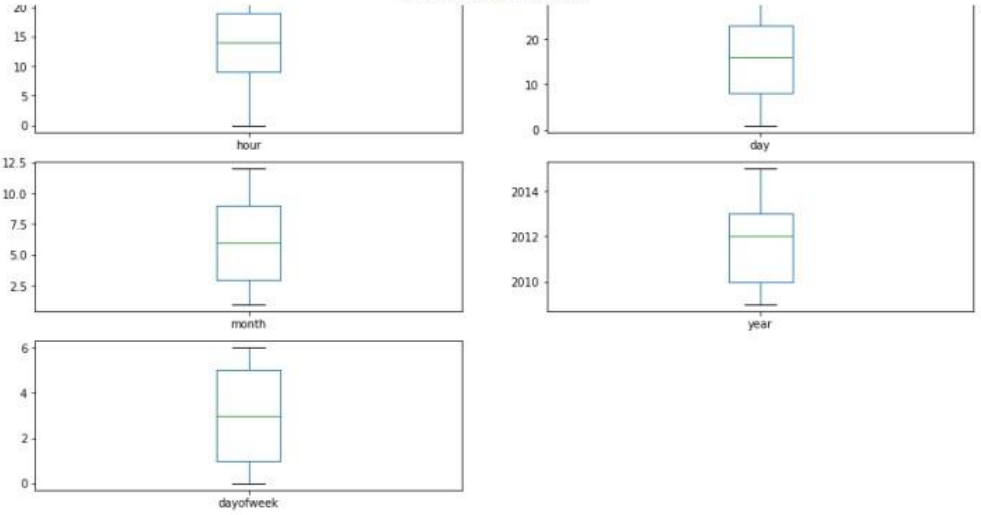
In [24]:
```
#Using the InterQuartile Range to fill the values
def remove_outlier(df1 , col):
    Q1 = df1[col].quantile(0.25)
    Q3 = df1[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_whisker = Q1-1.5*IQR
    upper_whisker = Q3+1.5*IQR
    df[col] = np.clip(df1[col] , lower_whisker , upper_whisker)
    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 ou
```

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
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'
        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	year	dayofwe
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1.0	19	7	5	2015	
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1.0	20	17	7	2009	
2	12.9	-74.005043	40.740770	-73.962565	40.772647	1.0	21	24	8	2009	
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3.0	8	26	6	2009	
4	16.0	-73.929786	40.744085	-73.973082	40.761247	3.5	17	28	8	2014	

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]:

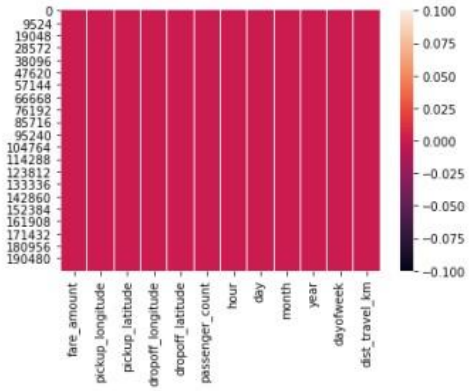
	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hour	day	month	year	dayofwe
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1.0	19	7	5	2015	
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1.0	20	17	7	2009	
2	12.9	-74.005043	40.740770	-73.962565	40.772647	1.0	21	24	8	2009	
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3.0	8	26	6	2009	
4	16.0	-73.929786	40.744085	-73.973082	40.761247	3.5	17	28	8	2014	

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
```

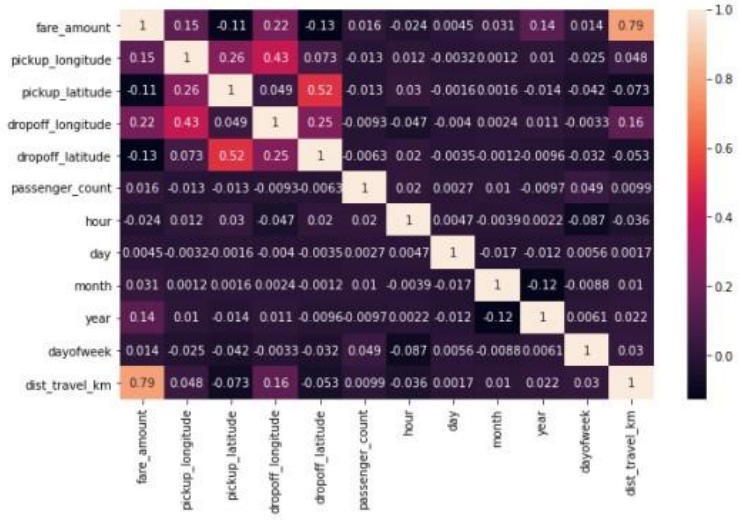
In [35]:
```
corr
```

Out[35]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hour	d
fare_amount	1.000000	0.154069	-0.110842	0.218675	-0.125898	0.015778	-0.023623	0.0045
pickup_longitude	0.154069	1.000000	0.259497	0.425619	0.073290	-0.013213	0.011579	-0.0032
pickup_latitude	-0.110842	0.259497	1.000000	0.048889	0.515714	-0.012889	0.029681	-0.0015
dropoff_longitude	0.218675	0.425619	0.048889	1.000000	0.245667	-0.009303	-0.046558	-0.0040
dropoff_latitude	-0.125898	0.073290	0.515714	0.245667	1.000000	-0.006308	0.019783	-0.0034
passenger_count	0.015778	-0.013213	-0.012889	-0.009303	-0.006308	1.000000	0.020274	0.0027
hour	-0.023623	0.011579	0.029681	-0.046558	0.019783	0.020274	1.000000	0.0046
day	0.004534	-0.003204	-0.001553	-0.004007	-0.003479	0.002712	0.004677	1.0000
month	0.030817	0.001169	0.001562	0.002391	-0.001193	0.010351	-0.003926	-0.0173
year	0.141277	0.010198	-0.014243	0.011346	-0.009603	-0.009749	0.002156	-0.0121
dayofweek	0.013652	-0.024652	-0.042310	-0.003336	-0.031919	0.048550	-0.086947	0.0056
dist_travel_km	0.786385	0.048446	-0.073362	0.155191	-0.052701	0.009884	-0.035708	0.0017

In [36]:
```
fig,axis = plt.subplots(figsize = (10,6))
sns.heatmap(df.corr(),annot = True) #Correlation Heatmap (Light values means highly correlated)
```

Out[36]: <AxesSubplot:>



Dividing the dataset into feature and target values

In [182]:
```
x = df[['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_latitude','passenger_count','hour','da
```

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
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 m
```

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
```

Group B

Assignment No:2

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

Dataset 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 the first and crucial step while creating a machine learning model.

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

Why do we need Data Preprocessing?

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

It involves below steps:

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

Assignment 2

2. Classify the email using the binary classification method. Email Spam detection has two states: a) Normal State – Not Spam, b) Abnormal State – Spam. Use K-Nearest Neighbors and Support Vector Machine for classification. Analyze their performance. Dataset link: The emails.csv dataset on the Kaggle https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv (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	Prediction
0	Email 1	0	0	1	0	0	0	2	0	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	0
2	Email 3	0	0	1	0	0	0	8	0	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	0
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0	0	1	0	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
```

Group B

Assignment No:3

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

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

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

Perform the following steps:

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

Objective of the Assignment:

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

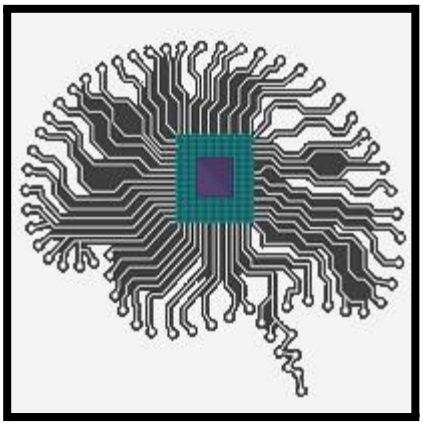
Prerequisite:

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

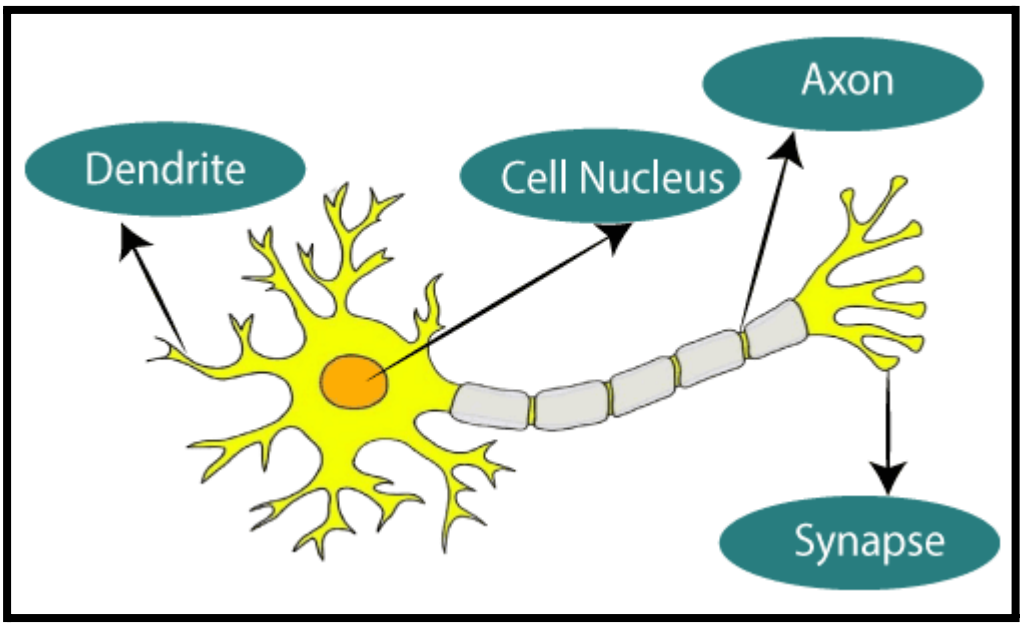
Contents of the Theory:

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

Artificial Neural Network:

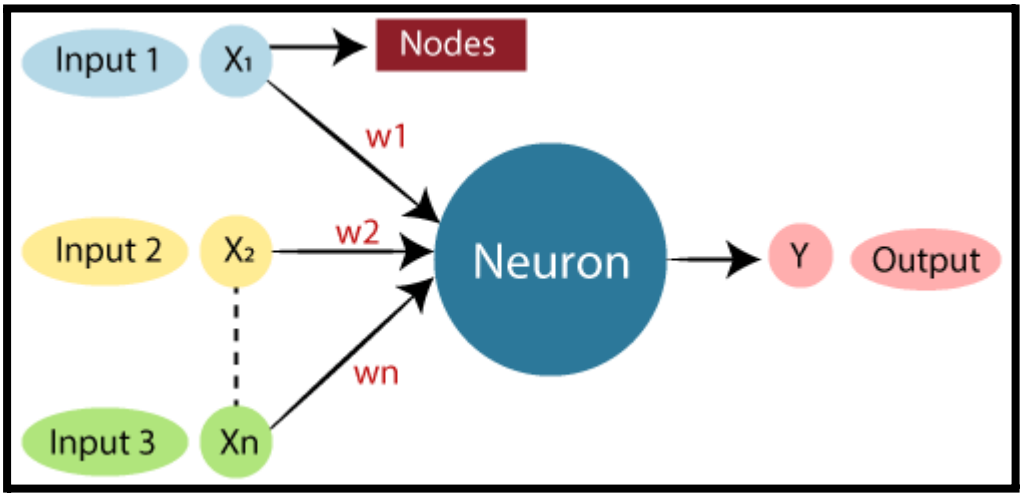


The term "Arti cial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, arti cial neural networks also have neurons that are interconnected to one another in various layers of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

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

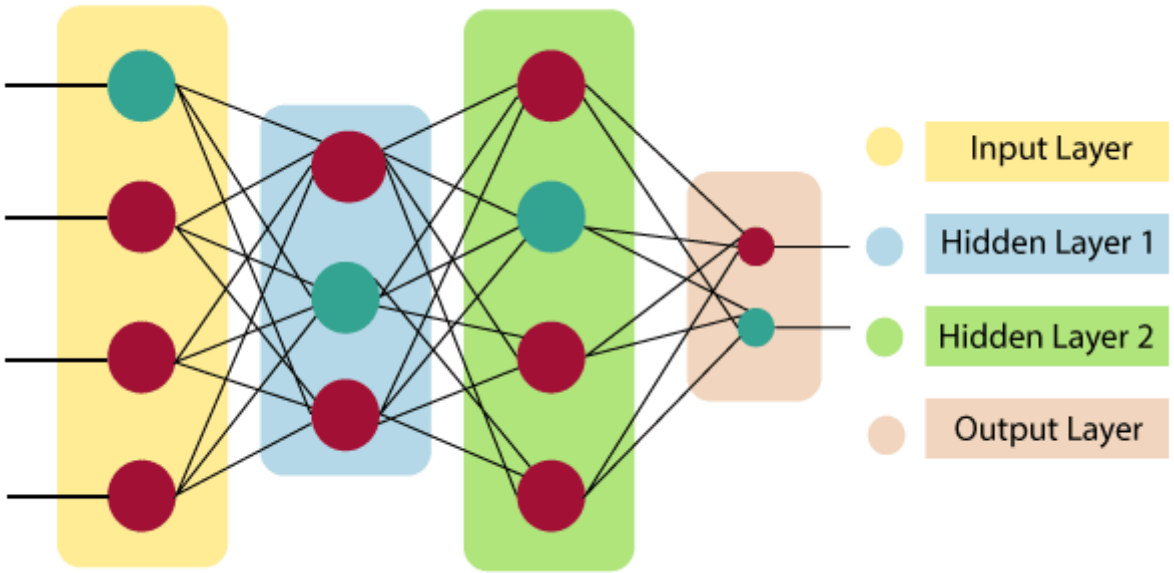
There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly.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

We can understand the artificial neural network with an example, consider an example of a digital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," thenwe 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 nd hidden features and patterns.

Output Layer:

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

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

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

It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should re or not. Only those who are red 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 rms likes Net ix, Uber, Google, Expedia, etc.

Tensor□ow:

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

Normalization:

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

Mathematically, we can calculate normalization with the below formula:

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

Where,

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

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

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

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

Put X =Xminimum in above formula, we get;

Xn = Xminimum- Xminimum/ (Xmaximum - Xminimum)

Xn = 0

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

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

Put X =Xmaximum in above formula, we get;

Xn = Xmaximum - Xminimum/ (Xmaximum - Xminimum)

Xn = 1

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

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

Normalization techniques in Machine Learning

Although there are so many feature normalization techniques in Machine Learning, few of them are most frequently used. These are as follows:

- **Min-Max Scaling:**This technique is also referred to as scaling. As we have already discussed above, the Min-Max scaling method helps the dataset to shift and rescale the values of their attributes, so they end up ranging between 0 and 1.
- **Standardization scaling:**

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

Hence, standardization can be expressed as follows:

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

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

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

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

When to use Normalization or Standardization?

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

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

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

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

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

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

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

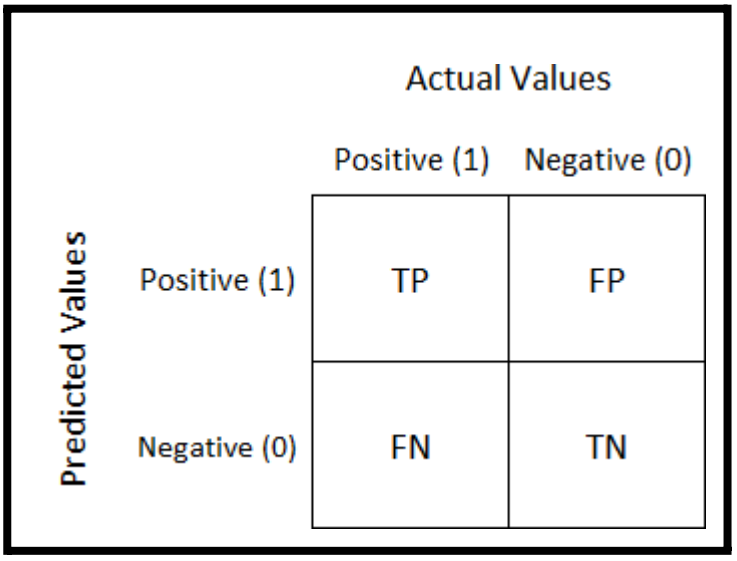
Because of its bigger value, the attributed income will organically influence the conclusion more when we undertake further analysis, such as multivariate linear regression. However, this does not necessarily imply that it is a better predictor. As a result, we normalize the data so that all of the variables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

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

Confusion Matrix:

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

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



The above table has the following cases:

- **True Negative:**Model has given prediction No, and the real or actual value was also No.
- **True Positive:**The model has predicted yes, and the actual value was also true.
- **False Negative:**The model has predicted no, but the actual value was Yes, it is also called as**Type-II error**.
- **False Positive:**The model has predicted Yes, but the actual value was No. It is also called a**Type-I error.**

Need for Confusion Matrix in Machine learning

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

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

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

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

From the above example, we can conclude that:

- The table is given for the two-class classi er, which has two predictions "Yes" and "NO." Here, Yes de nes that patient has the disease, and No de nes that patient doesnot has that disease.
- The classi er has made a total of**100 predictions**. Out of 100 predictions,**89 are true predictions**, and**11 are incorrect predictions**.
- The model has given prediction "yes" for 32 times, and "No" for 68 times. Whereas the actual "Yes" was 27, and actual "No" was 73 times.

Calculations using Confusion Matrix:

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

- **Classi cation Accuracy:**It is one of the important parameters to determine the accuracy of the classi cation problems. It de nes how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classi er to all number of predictions made by the classi ers. The formula is given below:

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

- **Misclassi cation rate:**It is also termed as Error rate, and it de nes how often the model gives the wrong predictions. The value of error rate can be calculated as the number of incorrect predictions to all number of the predictions made by the classi er. The formula is given below:

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

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

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

- **Recall:**It is de□ned as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

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

- **F-measure:**If two models have low precision and high recall or vice versa, it is di□cult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision. It can be calculated using the below formula:

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

Other important terms used in Confusion Matrix:

- **Null Error rate:**It de□nes how often our model would be incorrect if it always predicted the majority class. As per the accuracy paradox, it is said that "*the best classi□er has a higher error rate than the null error rate.*"
- **ROC Curve:**The ROC is a graph displaying a classi□er's performance for all possible thresholds. The graph is plotted between the true positive rate (on the Y-axis) and the false Positive rate (on the x-axis).

Conclusion:

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

Assignment Questions:

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

Program

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 (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	IsActiveMe
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	

In [49]:
```
df.shape
```

Out[49]:
```
(10000, 14)
```

In [50]:
```
df.describe()
```

Out[50]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMembe
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.00000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.51510
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.49979
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.00000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.00000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.00000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.00000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.00000

In [51]:
```
df.isnull()
```

Out[51]:

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

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]:

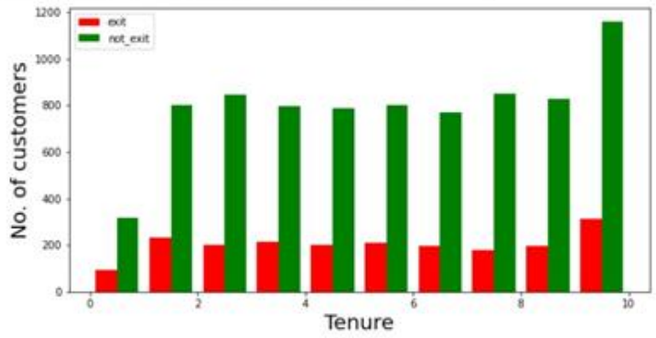
	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63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

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()
```

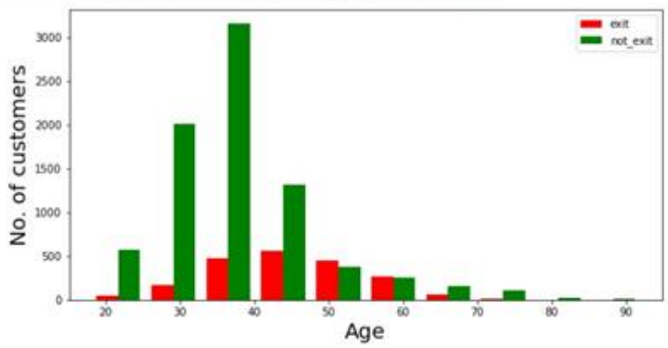
In [102]:
```
df_churn_exited = df[df['Exited']==1]['Tenure']
df_churn_not_exited = df[df['Exited']==0]['Tenure']
```

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','EstimatedS
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	Exited	Male
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1	0
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1	0
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0	0

In [63]:
```
X = df[['CreditScore','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary','
```

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"))   #Adding second hidden laye
```

In [76]:
```
classifier.add(Dense(activation = "sigmoid",units = 1,kernel_initializer = "uniform")) #Final neuron will be hav
```

In [77]:
```
classifier.compile(optimizer="adam",loss = 'binary_crossentropy',metrics = ['accuracy']) #To compile the Artific
```

In [79]:
```
classifier.summary() #3 layers created. 6 neurons in 1st,6neurons in 2nd layer and 1 neuron 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 dataset
```

```
Epoch 1/50
700/700 [==============================] - 0s 674us/step - loss: 0.4293 - accuracy: 0.7947
Epoch 2/50
700/700 [==============================] - 0s 647us/step - loss: 0.4239 - accuracy: 0.7947
Epoch 3/50
700/700 [==============================] - 0s 657us/step - loss: 0.4203 - accuracy: 0.8067
Epoch 4/50
700/700 [==============================] - 0s 664us/step - loss: 0.4167 - accuracy: 0.8260
Epoch 5/50
700/700 [==============================] - 0s 674us/step - loss: 0.4153 - accuracy: 0.8287
Epoch 6/50
700/700 [==============================] - 0s 653us/step - loss: 0.4137 - accuracy: 0.8310
Epoch 7/50
700/700 [==============================] - 0s 658us/step - loss: 0.4125 - accuracy: 0.8317
Epoch 8/50
700/700 [==============================] - 1s 842us/step - loss: 0.4116 - accuracy: 0.8306
Epoch 9/50
700/700 [==============================] - 0s 671us/step - loss: 0.4103 - accuracy: 0.8331
Epoch 10/50
700/700 [==============================] - 0s 682us/step - loss: 0.4100 - accuracy: 0.8326
Epoch 11/50
700/700 [==============================] - 0s 690us/step - loss: 0.4093 - accuracy: 0.8337
Epoch 12/50
700/700 [==============================] - 0s 688us/step - loss: 0.4087 - accuracy: 0.8339
Epoch 13/50
700/700 [==============================] - 0s 675us/step - loss: 0.4081 - accuracy: 0.8341
Epoch 14/50
700/700 [==============================] - 1s 722us/step - loss: 0.4071 - accuracy: 0.8331
Epoch 15/50
700/700 [==============================] - 1s 811us/step - loss: 0.4065 - accuracy: 0.8341
Epoch 16/50
700/700 [==============================] - 0s 711us/step - loss: 0.4056 - accuracy: 0.8356
Epoch 17/50
700/700 [==============================] - 0s 702us/step - loss: 0.4046 - accuracy: 0.8366
Epoch 18/50
700/700 [==============================] - 0s 688us/step - loss: 0.4035 - accuracy: 0.8343
Epoch 19/50
700/700 [==============================] - 1s 715us/step - loss: 0.4024 - accuracy: 0.8363
Epoch 20/50
700/700 [==============================] - 0s 714us/step - loss: 0.4020 - accuracy: 0.8337
Epoch 21/50
```

```
700/700 [==============================] - 0s 705us/step - loss: 0.4010 - accuracy: 0.8374
Epoch 22/50
700/700 [==============================] - 1s 720us/step - loss: 0.4003 - accuracy: 0.8370
Epoch 23/50
700/700 [==============================] - 0s 692us/step - loss: 0.3993 - accuracy: 0.8374
Epoch 24/50
700/700 [==============================] - 0s 709us/step - loss: 0.3990 - accuracy: 0.8356
Epoch 25/50
700/700 [==============================] - 1s 871us/step - loss: 0.3984 - accuracy: 0.8366
Epoch 26/50
700/700 [==============================] - 1s 719us/step - loss: 0.3984 - accuracy: 0.8367
Epoch 27/50
700/700 [==============================] - 1s 719us/step - loss: 0.3980 - accuracy: 0.8366
Epoch 28/50
700/700 [==============================] - 0s 695us/step - loss: 0.3981 - accuracy: 0.8366
Epoch 29/50
700/700 [==============================] - 0s 667us/step - loss: 0.3976 - accuracy: 0.8374
Epoch 30/50
700/700 [==============================] - 0s 669us/step - loss: 0.3972 - accuracy: 0.8373
Epoch 31/50
700/700 [==============================] - 0s 670us/step - loss: 0.3970 - accuracy: 0.8370
Epoch 32/50
700/700 [==============================] - 1s 720us/step - loss: 0.3972 - accuracy: 0.8376
Epoch 33/50
700/700 [==============================] - 0s 675us/step - loss: 0.3965 - accuracy: 0.8367
Epoch 34/50
700/700 [==============================] - 0s 680us/step - loss: 0.3961 - accuracy: 0.8364
Epoch 35/50
700/700 [==============================] - 0s 685us/step - loss: 0.3962 - accuracy: 0.8379
Epoch 36/50
700/700 [==============================] - 1s 771us/step - loss: 0.3960 - accuracy: 0.8370
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: 0.8373
Epoch 39/50
700/700 [==============================] - 1s 823us/step - loss: 0.3950 - accuracy: 0.8384
Epoch 40/50
700/700 [==============================] - 1s 759us/step - loss: 0.3956 - accuracy: 0.8361
Epoch 41/50
700/700 [==============================] - 1s 773us/step - loss: 0.3949 - accuracy: 0.8366
Epoch 42/50
700/700 [==============================] - 0s 695us/step - loss: 0.3953 - accuracy: 0.8369
```



```
Epoch 43/50
700/700 [==============================] - 0s 701us/step - loss: 0.3952 - accuracy: 0.8369
Epoch 44/50
700/700 [==============================] - 0s 707us/step - loss: 0.3952 - accuracy: 0.8366
Epoch 45/50
700/700 [==============================] - 0s 680us/step - loss: 0.3955 - accuracy: 0.8376
Epoch 46/50
700/700 [==============================] - 0s 665us/step - loss: 0.3947 - accuracy: 0.8373
Epoch 47/50
700/700 [==============================] - 0s 708us/step - loss: 0.3947 - accuracy: 0.8371
Epoch 48/50
700/700 [==============================] - 0s 681us/step - loss: 0.3944 - accuracy: 0.8371
Epoch 49/50
700/700 [==============================] - 0s 678us/step - loss: 0.3947 - accuracy: 0.8383
Epoch 50/50
700/700 [==============================] - 1s 869us/step - loss: 0.3944 - accuracy: 0.8370
```

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)
```

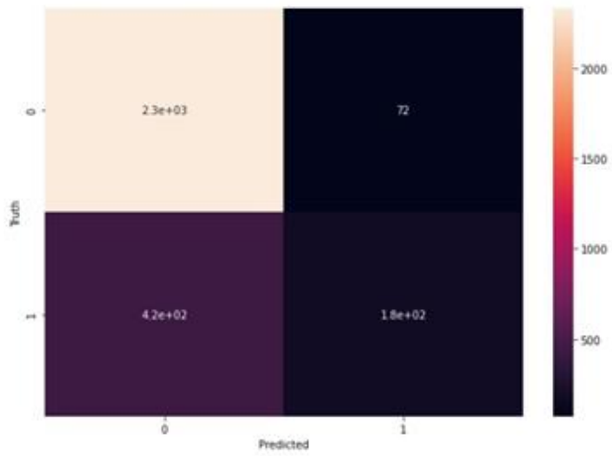
In [94]:
```
accuracy = accuracy_score(y_test,y_pred)
```

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 [ ]:

Group B

Assignment No:4

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

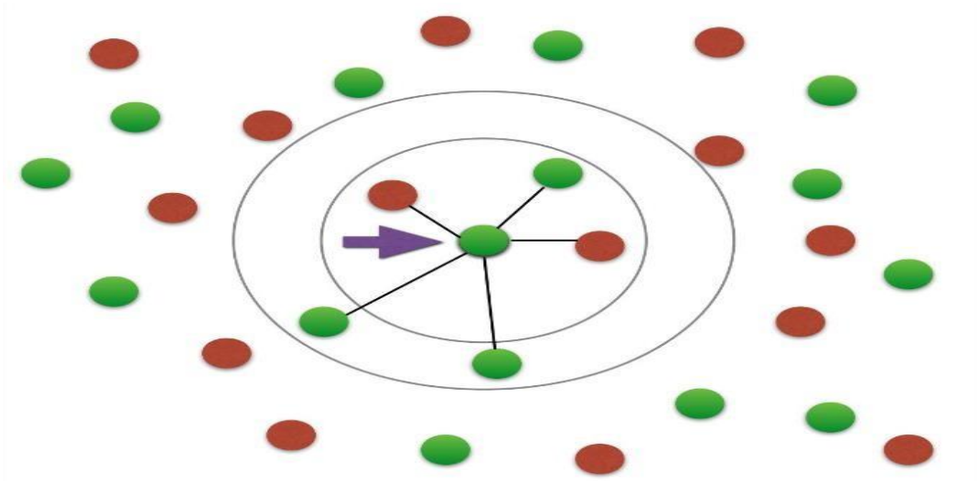
Dataset Description: We will try to build a machine learning model to accurately predict whether or not the patients in the dataset have diabetes or not?
The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies 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 tomake 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 isthen 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 belabeled '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 ourtarget.

We will use pandas 'drop' function to drop the column 'diabetes' from our dataframe and store it in thevariable '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
valuesy[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,
```

'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 patientswithout diabetes.

Building and training the model

Next, we have to build the model. Here is the code:

```
from sklearn.neighbors import KNeighborsClassifier# Create KNN
classifierknn = KNeighborsClassifier(n_neighbors = 3)# Fit the classifier
to the dataknn.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 labeledwith 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 andpass 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 and1 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
```

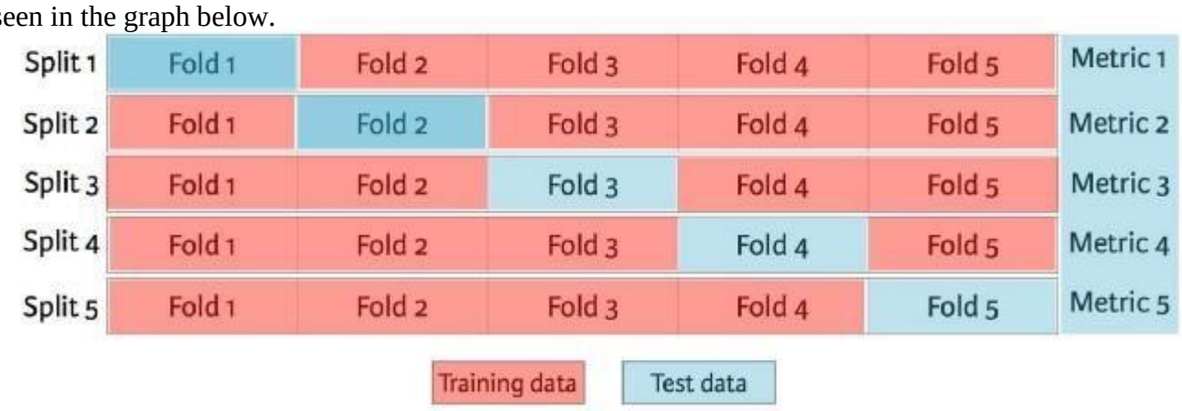
Our model has an accuracy of approximately 66.88%. It’s a good start, but we will see how we canincrease model performance below.

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 testset.

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 canbe seen in the graph below.



4-fold cross validation (image credit)

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 geta 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 ofhow 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 worksbest for our model. To do this, we will create a dictionary, setting ‘n_neighbors’ as the key and usingnumpy 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 accuracyof 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%!

Code :- https://www.kaggle.com/code/shrutimechlearn/step-by-step-diabetes-classification-knn-detailed

Conclusion:

In this way we build a a neural network-based classifier that can determine whether they willleave or not inthe next 6 months

Program

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
```

Practice-III

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
```

Group B

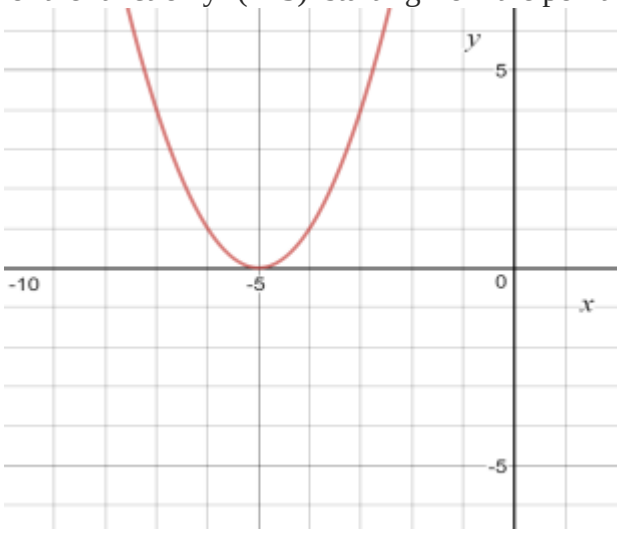
Assignment No:5

Title: Implement Gradient Descent Algorithm.

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

Example by hand :

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



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

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

Step 1 : Initialize x =3. Then, find the gradient of the function, dy/dx = 2*(x+5).
Step 2 : Move in the direction of the negative of the gradient (Why?). But wait, how much to move?For that, we require a learning rate. Let us assume the **learning rate → 0.01**
Step 3 : Let's perform 2 iterations of gradient descent
Step 4 : We can observe that the X value is slowly decreasing and should converge to -5 (the localminima). However, how many iterations should we perform?
Let us set a precision variable in our algorithm which calculates the difference between two consecutive "x" values . If the difference between x values from 2 consecutive iterations is lesser than the precision we set, stop the algorithm !

Step 4 : We can observe that the X value is slowly decreasing and should converge to -5 (the localminima).

However, how many iterations should we perfor?

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

Gradient descent in Python :

Step 1 : Initialize parameters

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

Step 2 : Run a loop to perform gradient descent :
i. Stop loop when difference between x values from 2 consecutive iterations is less than 0.000001or when number of iterations exceeds 10,000

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

iterationsprint("The local minimum occurs at", cur_x)

Facilities:

GoogleColab,Jupiter notebook

Input:

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

Output:

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

Conclusion:

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

Group C

Assignment No: 1

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

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

Prerequisite:

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

Contents for Theory:

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

Introduction to Blockchain

- Blockchain can be described as a data structure that holds transactional records and while ensuring security, transparency, and decentralization. You can also think of it as a chain or records stored in the forms of blocks which are controlled by no single authority.
- A blockchain is a distributed ledger that is completely open to any and everyone on the network. Once an information is stored on a blockchain, it is extremely difficult to change or alter it.
- Each transaction on a blockchain is secured with a digital signature that proves its authenticity. Due to the use of encryption and digital signatures, the data stored on the blockchain is tamper-proof and cannot be changed.
- Blockchain technology allows all the network participants to reach an agreement, commonly known as consensus. All the data stored on a blockchain is recorded digitally and has a common history which is available for all the network participants. This way, the chances of any fraudulent activity or duplication of transactions is eliminated without the need of a third-party.

Blockchain Features

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

- **Decentralized**

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

- **Peer-to-Peer Network**

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

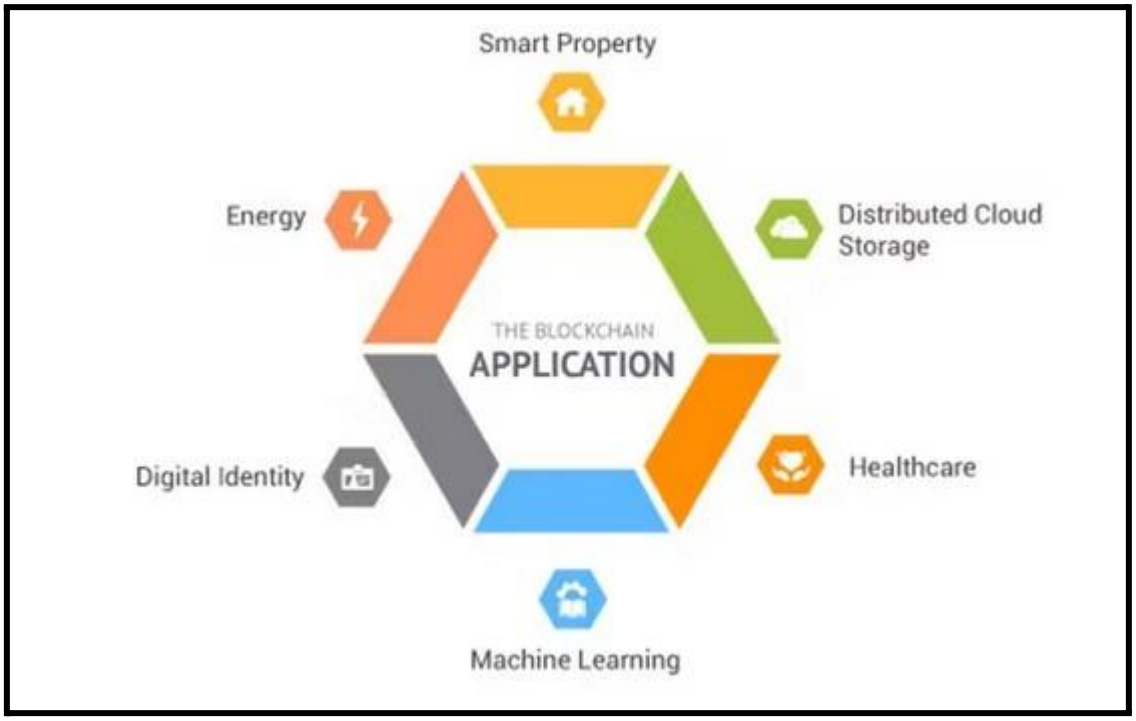
- **Immutable**

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

- **Tamper-Proof**

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

Popular Applications of Blockchain Technology



Benefits of Blockchain Technology:

- **Time-saving:** No central Authority verification needed for settlements making the process faster and cheaper.
- **Cost-saving:** A Blockchain network reduces expenses in several ways. No need for third-party verification. Participants can share assets directly. Intermediaries are reduced. Transaction efforts are minimized as every participant has a copy of shared ledger.
- **Tighter security:** No one can temper with Blockchain Data as it is shared among

millions of participants. The system is safe against cybercrimes and Fraud.

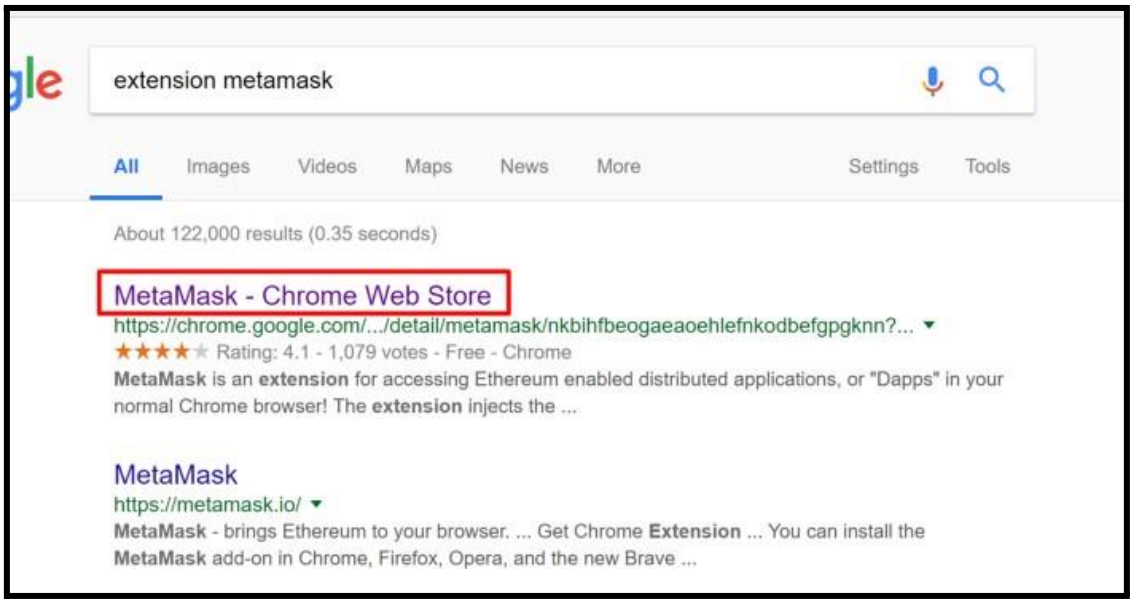
- In finance market trading, Fibonacci retracement levels are widely used in technical analysis.

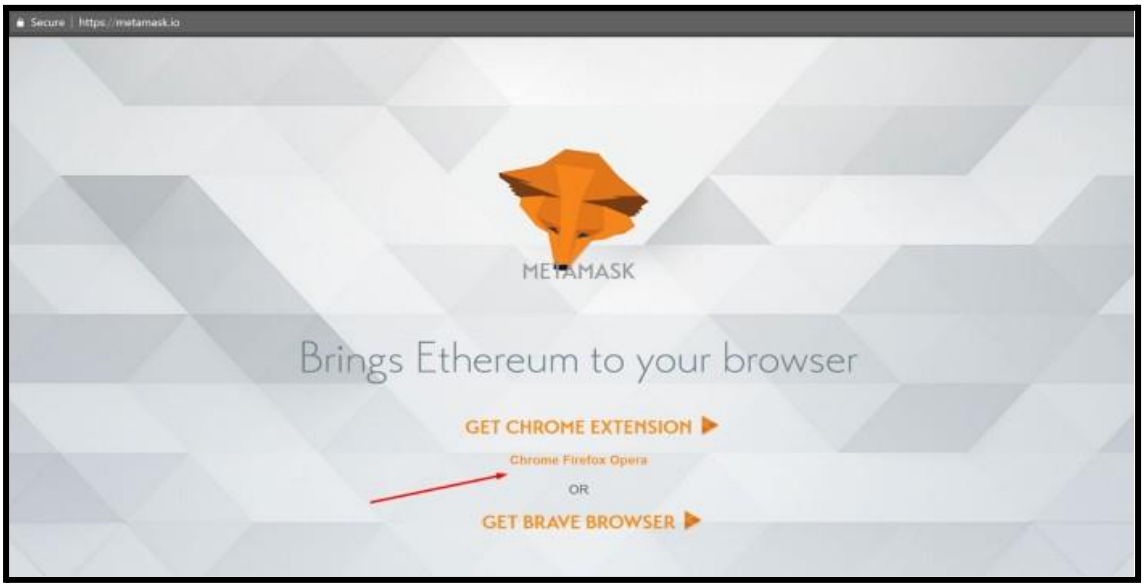
How to use MetaMask: A step by step guide

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

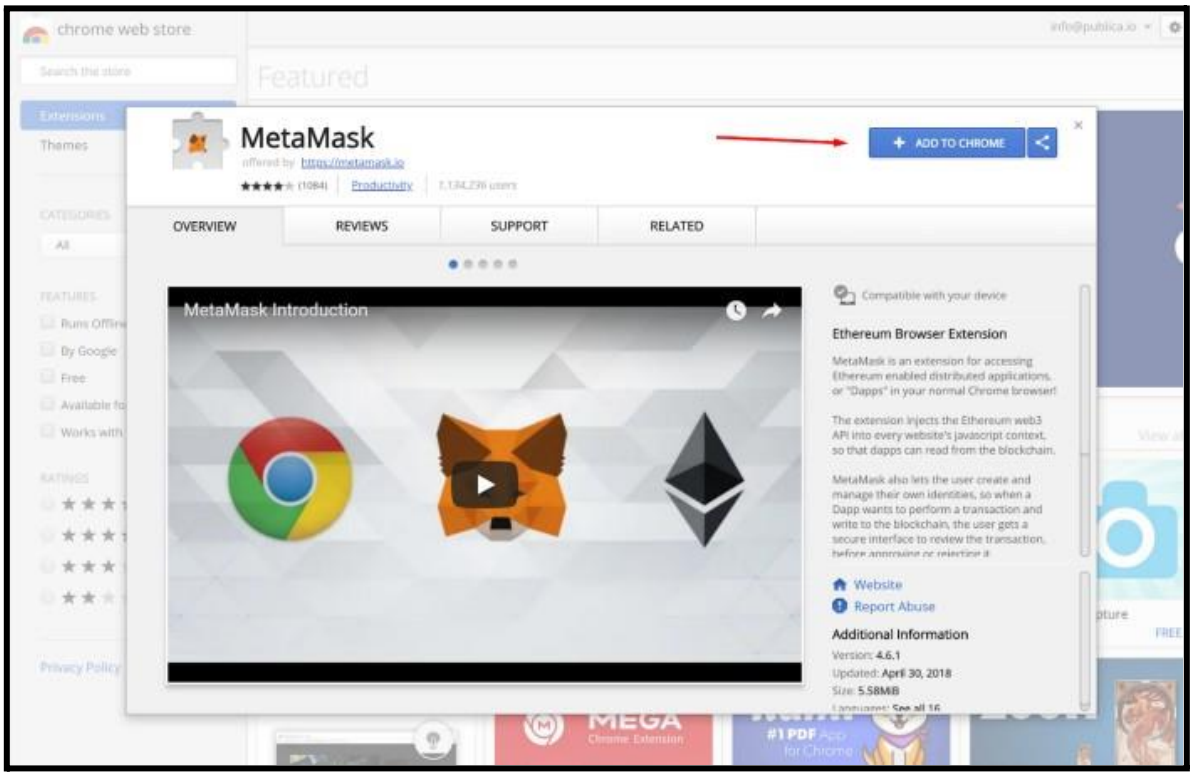
Step 1. Install MetaMask on your browser.

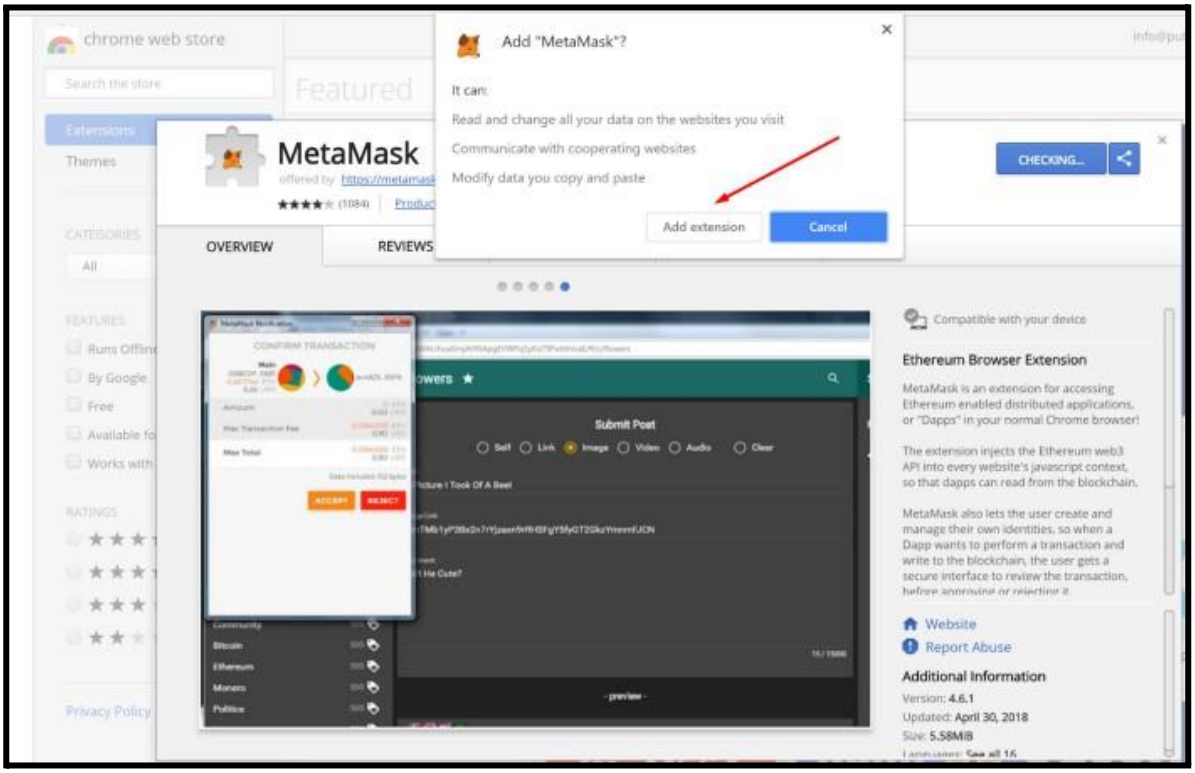
To create a new wallet, you have to install the extension first. Depending on your browser, there are different marketplaces to find it. Most browsers have MetaMask on their stores, so it's not that hard to see it, but either way, here they are Chrome, Firefox, and Opera.





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

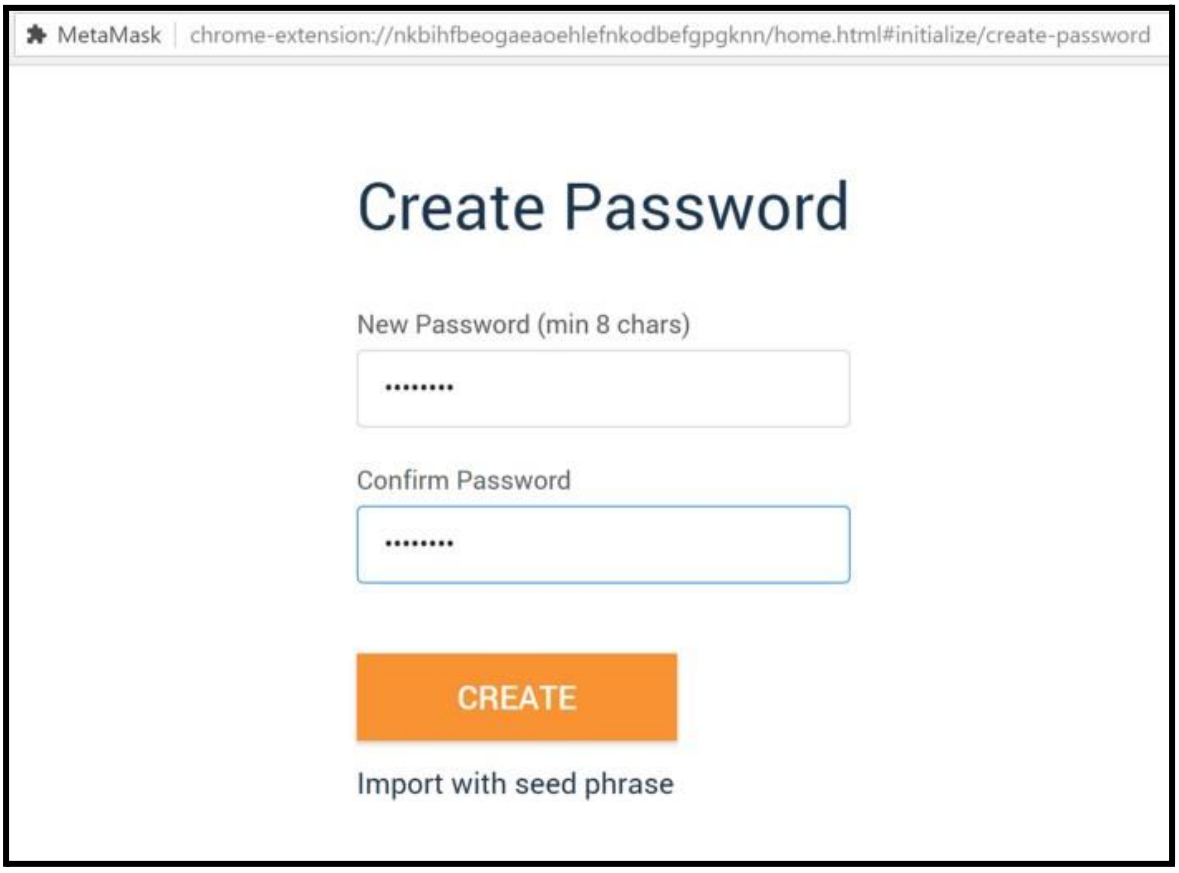




nd it's as easy as that to install the extension on your browser, continue reading the next step to figure out how to create an account.

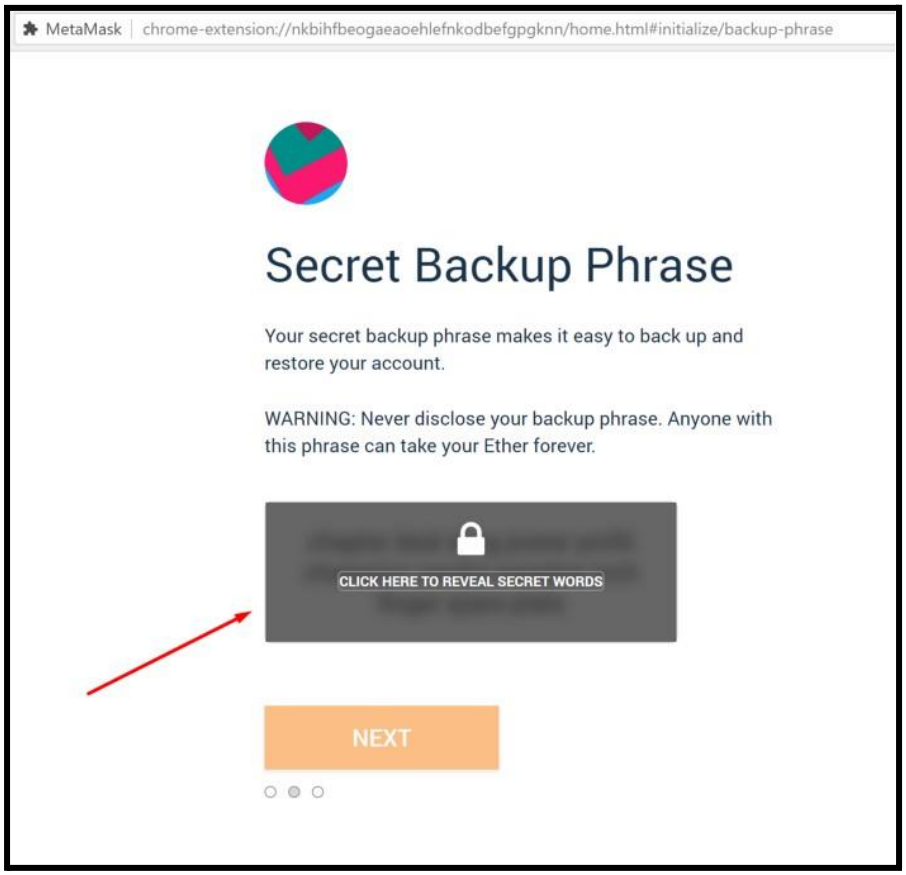
Step 2. Create an account.

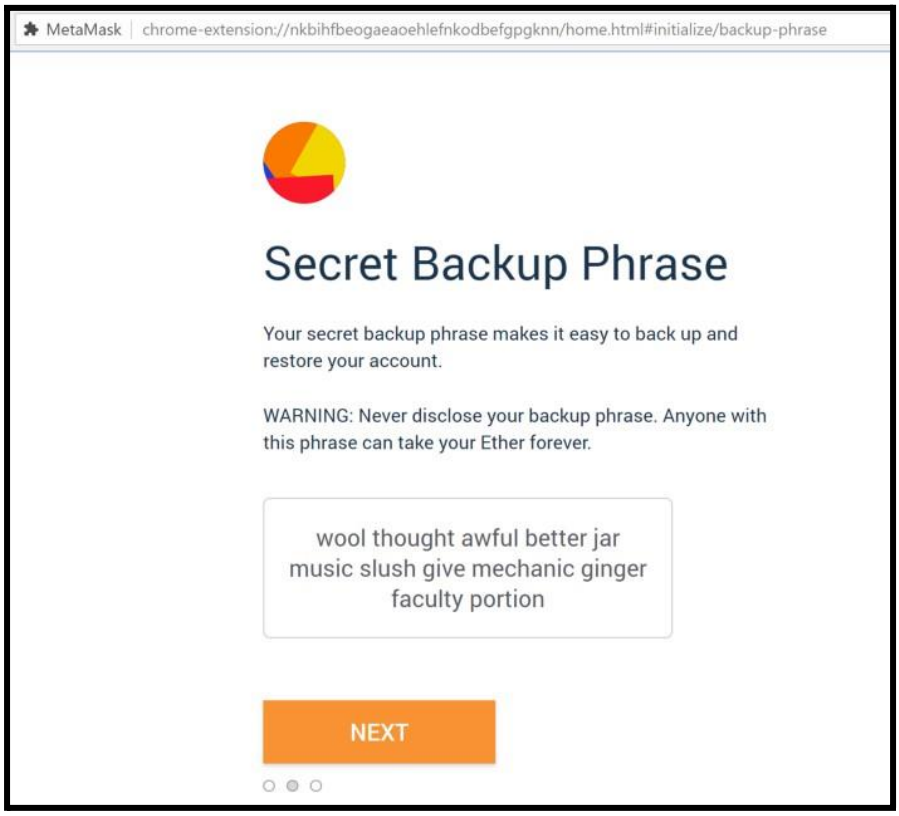
- Click on the extension icon in the upper right corner to open MetaMask.
- To install the latest version and be up to date, **click Try it now**.
- **Click Continue**.
- You will be prompted to create a new password. **Click Create**.



- Proceed by **clicking Next** and accept the Terms of Use.

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





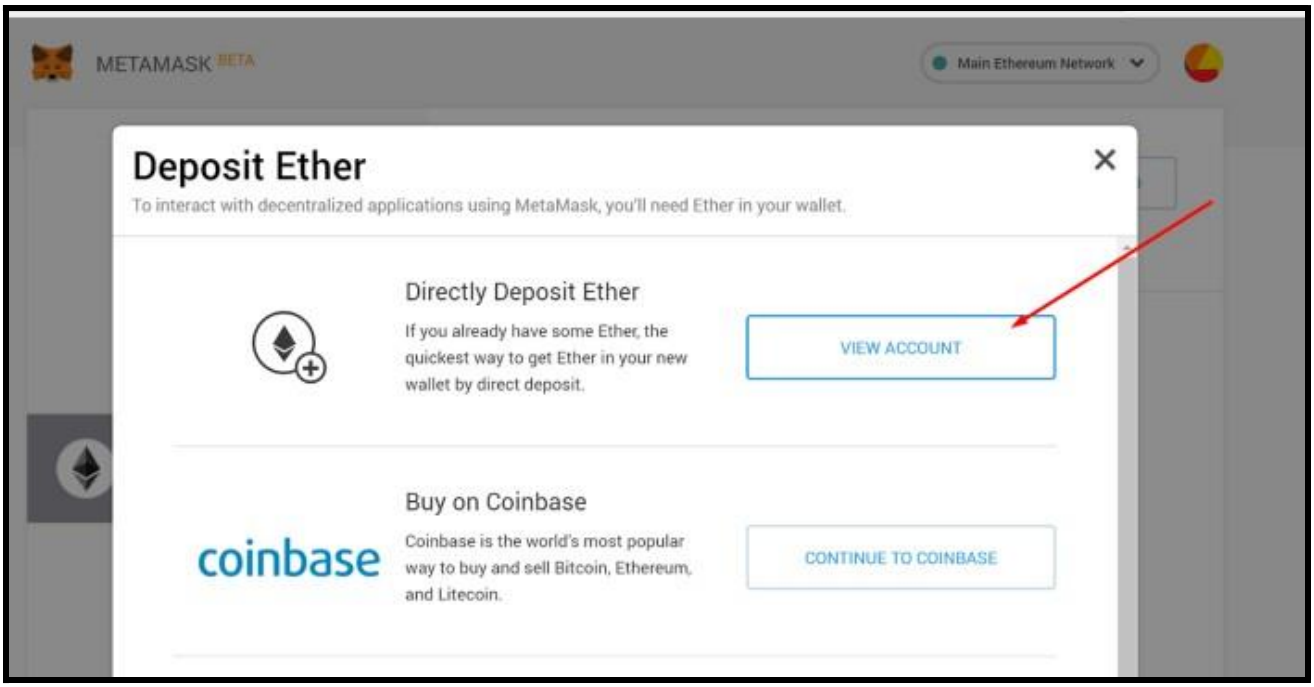
- Verify your secret phrase by selecting the previously generated phrase in order. **Click Confirm**.

And that's it; now you have created your MetaMask account successfully. A new Ethereum wallet

address has just been created for you. It's waiting for you to deposit funds, and if you want to learn how to do that, look at the next step below.

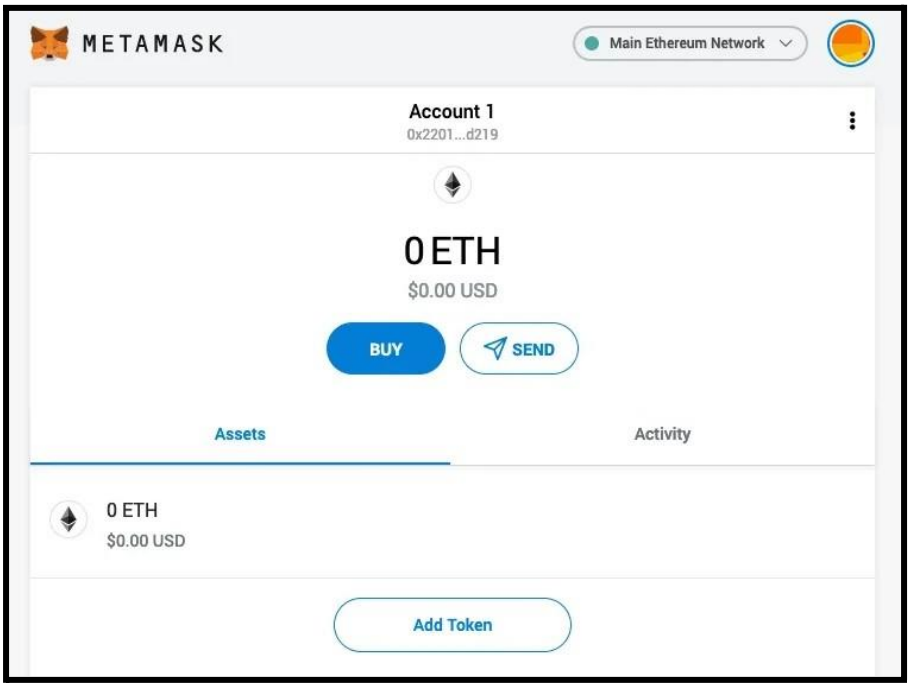
Step 3. Depositing funds.

- Click on **View Account**.



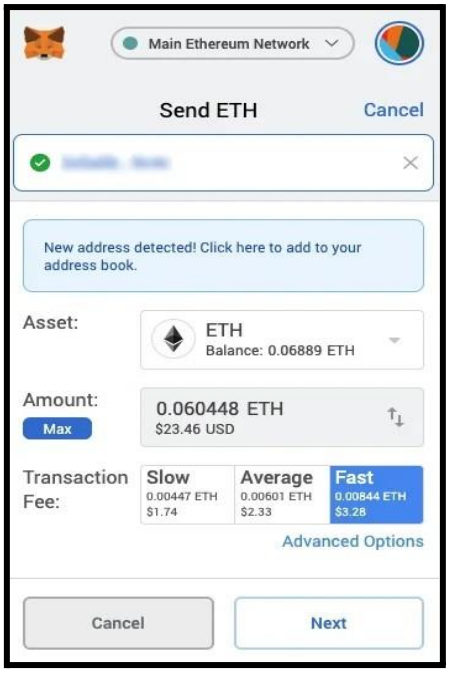
You can now see your public address and share it with other people. There are some methods to buy coins offered by MetaMask, but you can do it differently as well; you just need your address.

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



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

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



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

What advantages does MetaMask have?

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

Conclusion- In this way we have explored Concept Blockchain and metamat wallet for transaction of digital currency

Assignment Question

1. **What Are the Different Types of Blockchain Technology?**
2. **What Are the Key Features/Properties of Blockchain?**
3. **What Type of Records You Can Keep in A Blockchain?**
4. **What is the difference between Ethereum and Bitcoin?**
5. **What are Merkle Trees? Explain their concept.**
6. **What is Double Spending in transaction operation**
7. **Give real-life use cases of blockchain.**

Reference link

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

Group C

Assignment No: 2

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

Objective of the Assignment: Students should be able to learn about cryptocurrencies and learn how transaction done by using different digital currency

Prerequisite:

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

Contents for Theory:

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

Introduction to Cryptocurrency

- Cryptocurrency is a digital payment system that doesn't rely on banks to verify transactions. It's a peer-to-peer system that can enable anyone anywhere to send and receive payments. Instead of being physical money carried around and exchanged in the real world, cryptocurrency payments exist purely as digital entries to an online database describing specific transactions. When you transfer cryptocurrency funds, the transactions are recorded in a public ledger. Cryptocurrency is stored in digital wallets.
- Cryptocurrency received its name because it uses encryption to verify transactions. This means advanced coding is involved in storing and transmitting cryptocurrency data between wallets and to public ledgers. The aim of encryption is to provide security and safety.
- The first cryptocurrency was Bitcoin, which was founded in 2009 and remains the best known today. Much of the interest in cryptocurrencies is to trade for profit, with speculators at times driving prices skyward.

How does cryptocurrency work?

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

Cryptocurrency examples

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

- **Bitcoin:**

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

- **Ethereum:**

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

- **Litecoin:**

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

- **Ripple:**

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

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

How to store cryptocurrency

- Once you have purchased cryptocurrency, you need to store it safely to protect it from hacks or theft. Usually, cryptocurrency is stored in crypto wallets, which are physical devices or online software used to store the private keys to your cryptocurrencies securely. Some exchanges provide wallet services, making it easy for you to store directly through the platform. However, not all exchanges or brokers automatically provide wallet services for you.
- There are different wallet providers to choose from. The terms “hot wallet” and “cold wallet” are used:
- **Hot wallet storage:** "hot wallets" refer to crypto storage that uses online software to protect the private keys to your assets.
- **Cold wallet storage:** Unlike hot wallets, cold wallets (also known as hardware wallets) rely on offline electronic devices to securely store your private keys.

Conclusion- In this way we have explored Concept Cryptocurrency and learn howtransactions are done using digital currency

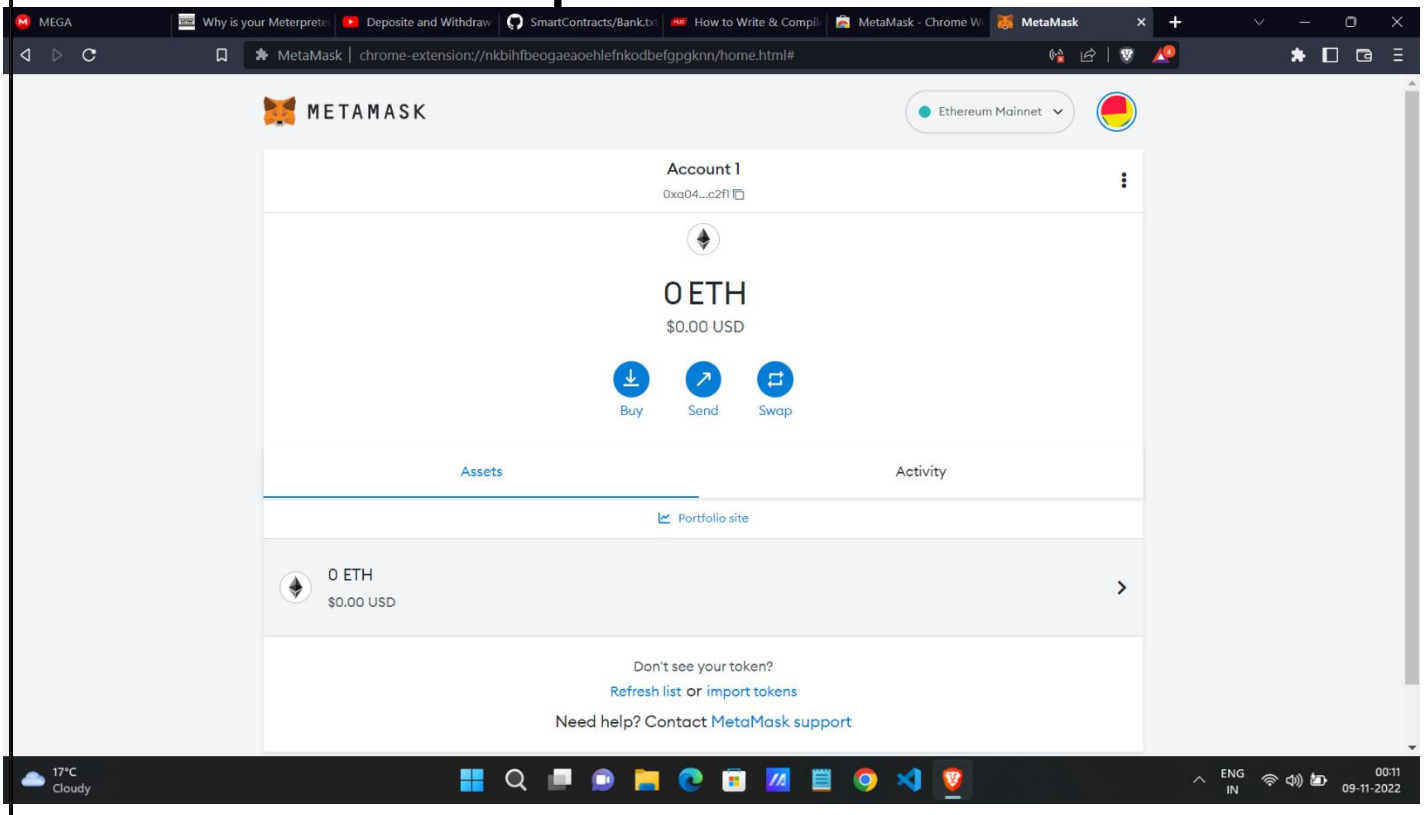
Assignment Question

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

Reference link

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

Output



Assignment No: 3

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

- Deposit money
- Withdraw Money
- Show balance

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

Prerequisite:

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

Contents for Theory:

The contract will allow deposits from any account, and can be trusted to allow withdrawals only by accountsthat have sufficient funds to cover the requested withdrawal.

This post assumes that you are comfortable with the ether-handling concepts introduced in our post, Writing a Contract That Handles Ether.

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

```
pragma solidity ^0.4.19;

contract TipJar {

   address owner;       // current owner of the contract

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

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

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

   function getBalance() public view returns (uint256)
```

```
        {return address(this).balance;
    }
}
```

I am going to generalize this contract to keep track of ether deposits based on the account address of the depositor, and then only allow that same account to make withdrawals of that ether. To do this, we need a way keep track of account balances for each depositing account—a mapping from accounts to balances. Fortunately, Solidity provides a ready-made mapping data type that can map account addresses to integers,

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

```
pragma solidity

^0.4.19;contract Bank {

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

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

        balanceOf[msg.sender] += amount;      // adjust the account's balance
    }
}
```

Here are the new concepts in the code above:

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

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

bank.sol

```
pragma solidity ^0.4.19;

contract Bank {

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

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

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

The code above demonstrates the following:

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

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

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

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

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

Program :

```
pragma solidity ^0.6.0;
contract MyBank
{

mapping(address=> uint ) private _balances;
address public owner;
```

```
 event LogDepositeMade(address accountHoder, uint amount );
constructor () public
{
  owner=msg.sender;
            emit LogDepositeMade(msg.sender, 1000);
}
   function deposite() public payable  returns (uint)
  {
  require ((_balances[msg.sender] + msg.value) >  _balances[msg.sender] && msg.sender!=address(0));
 _balances[msg.sender] += msg.value;
emit LogDepositeMade(msg.sender , msg.value);
return _balances[msg.sender];
 }
  function withdraw (uint withdrawAmount) public  returns (uint)
 {
        require (_balances[msg.sender] >= withdrawAmount);
    require(msg.sender!=address(0));
    require (_balances[msg.sender] > 0);
    _balances[msg.sender]-= withdrawAmount;
    msg.sender.transfer(withdrawAmount);
    emit LogDepositeMade(msg.sender , withdrawAmount);
    return _balances[msg.sender];
}
function viewBalance() public view returns (uint)
{
   return _balances[msg.sender];
}
}
```

Group C

Assignment No: 4

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

Problem Statement: Write and survey report on blockchain technology

Prerequisites: Blockchain Technology

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

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

Theory:

Blockchain Concepts:

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

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

Blockchain Types

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

Permissionless Blockchains

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

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

Permissioned Blockchains:

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

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

Real Time blockchain use cases

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

Capital Markets

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

Central Bank Digital Currencies CBDC

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

Decentralized Finance (DeFi)

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

Digital Identity

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

Finance

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

Energy and Sustainability

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

Global Trade and Commerce

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

Government and the Public Sector

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

Healthcare and the Life Sciences

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

Real Estate

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

Supply Chain Management

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

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

Group C

Assignment No: 5

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

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

Prerequisites: Blockchain Technology

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

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

Theory:

Ethereum:

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

Overview of Smart Contracts:

A smart contract is a small program that runs on an Ethereum blockchain. Once the smart contractis deployed on the Ethereum blockchain, it cannot be changed. To deploy the smart contract to Ethereum, you must pay the ether (ETH) cost. Understand it as a digital agreement that builds trustand allows both parties to agree on a particular set of conditions that cannot be tampered with.
To understand the need for a smart contract, suppose there was one grocery shop, and ram went to buy some groceries. He purchased the groceries for 500 rupees and kept on debt that would pay themoney next month when he returned, so the shopkeeper jotted down his purchase in his ledger. Inbetween the period somehow shopkeeper changed 500 to 600 and when next month ram went to pay the money, the shopkeeper has demanded 600 INR and ram has no proof to show that he has only bought 500 INR so in this case, smart contracts play an essential role which prevents both the parties to tamper the agreement and only gets terminate when all the conditions satisfy after the deal.

Ethereum Blockchain Platform executes Smart Contracts:

Ethereum Virtual Machine (EVM)

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

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

Gas

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

Solidity:

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

Solidity Programming

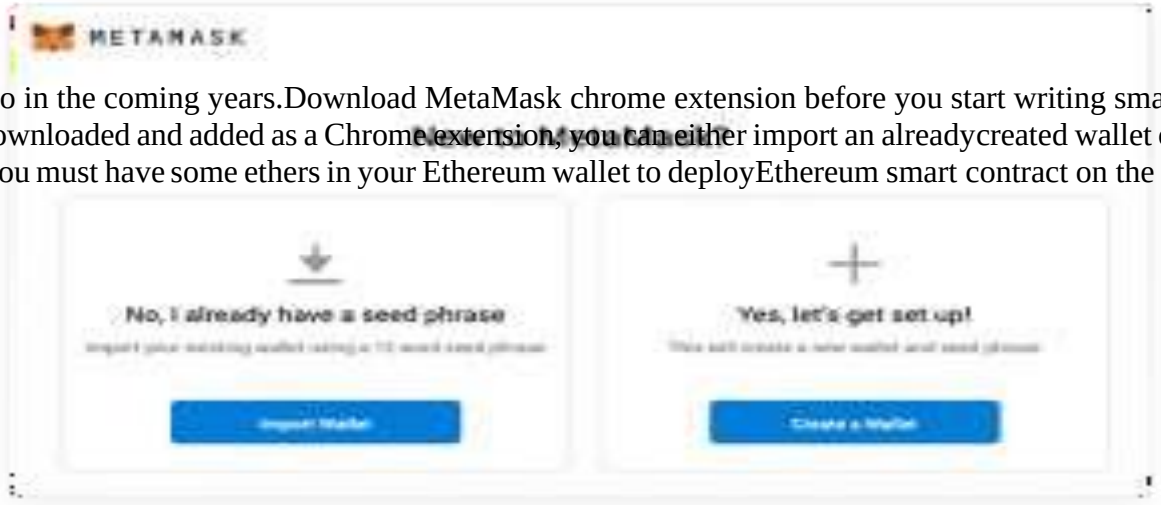
Solidity is object-oriented, high-level statically-typed programing language used to create smart contracts. Solidity programming looks similar to Javascript, but there are a lot of differences between both languages. In solidity, you need to compile the program first, while in Javascript, you can run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

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

Installing Prerequisites

Meta-mask Chrome Extension

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



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

Steps to develop an Ethereum Smart Contract

Step 1: Create a wallet at meta-mask

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

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

Step 2: Select any one test network

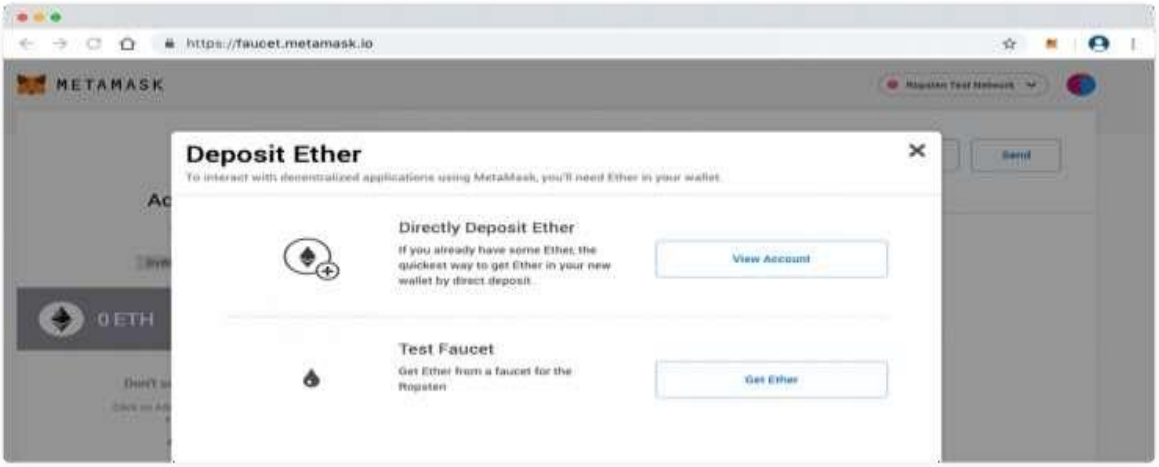
You might also find the following test networks in your MetaMask wallet:

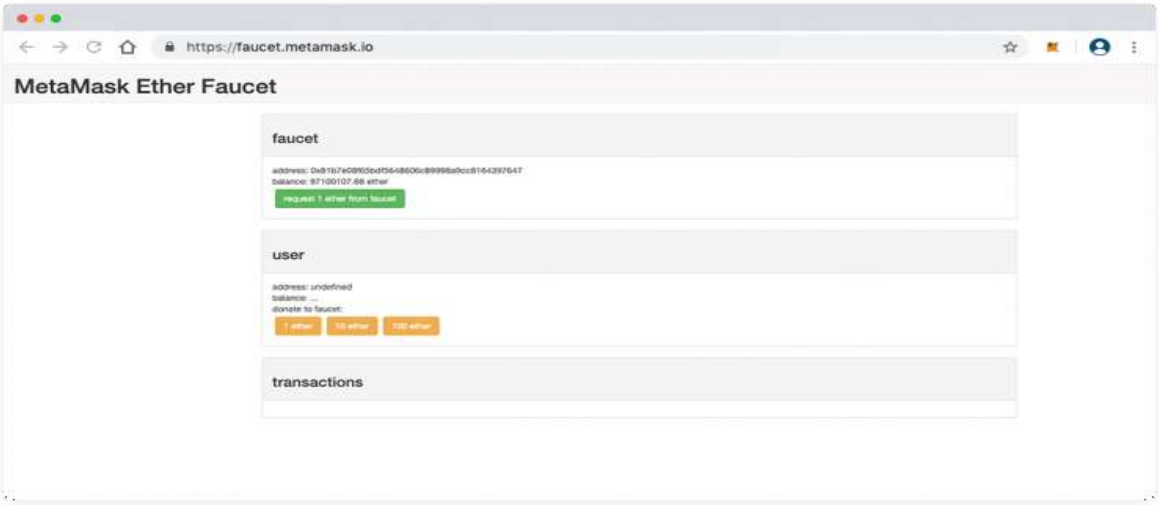
- Robsten Test Network
- Kovan Test Network
- Rinkeby Test Network
- Goerli Test Network

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

Step 3: Add some dummy Ethers to your wallet

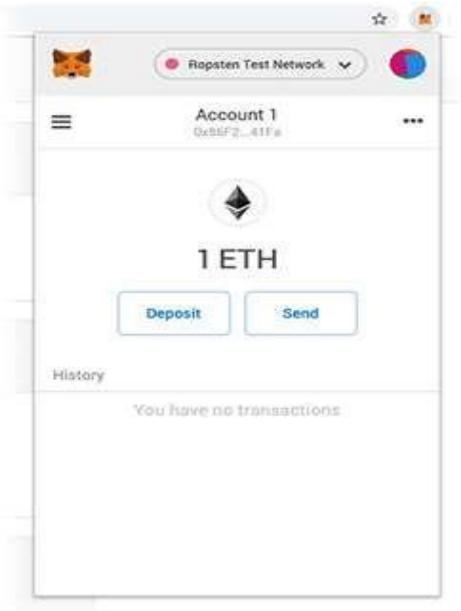
In case you want to test the smart contract, you must have some dummy ethers in your MetaMaskwallet.





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

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



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

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

We will use Remix Browser IDE to write our Solidity code. The remix is the best option for writingsmart contracts

as it comes with a handful of features and offers a comprehensive development experience.

Step 5: Create a .sol extension file

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

Step 6: A sample Code

Arrays in Solidity

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

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



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

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

Creating Dynamic Array

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

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

Structure in Solidity

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

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

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

Fallback:

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

```
            return true;
        }

        // This fallback function
        // will keep all the Ether
        function() public payable
        {
            balance[msg.sender] += msg.value;
        }
}
// Creating the sender contract
contract Sender
{
function transfer() public payable
{
        // Address of Fback contract
        address _receiver =
                    0xbcD310867F1b74142c2f5776404b6bd97165FA56;

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

Create a Smart Contract with CRUD Functionality

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

A sample smart contract code to create ERC20 tokens

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

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

Step 7: Deploy your contract

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

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

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

Steps to deploy Ethereum Smart Contracts

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

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

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