



Ajeenkya D Y Patil School Of Engineering, Lohegaon, Pune.

Savitribai Phule Pune University (SPPU)
Fourth Year of Computer Engineering (2019 Course)

410246: Laboratory Practice III

Term work: 50 Marks

Practical: 50 Marks

Design and Analysis of Algorithms (410241)

Machine Learning(410242)

Blockchain Technology(410243)

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 Non-recursive functions
 3. Execution flow of calculate Fibonacci numbers
 4. Basic of Time and Space complexity
-

Contents for Theory:

1. Introduction to Fibonacci numbers
2. Time and Space complexity

Introduction to Fibonacci numbers

- The Fibonacci series, named after Italian mathematician Leonardo Pisano Bogollo, later known as Fibonacci, is a series (sum) formed by Fibonacci numbers denoted as F_n . 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?

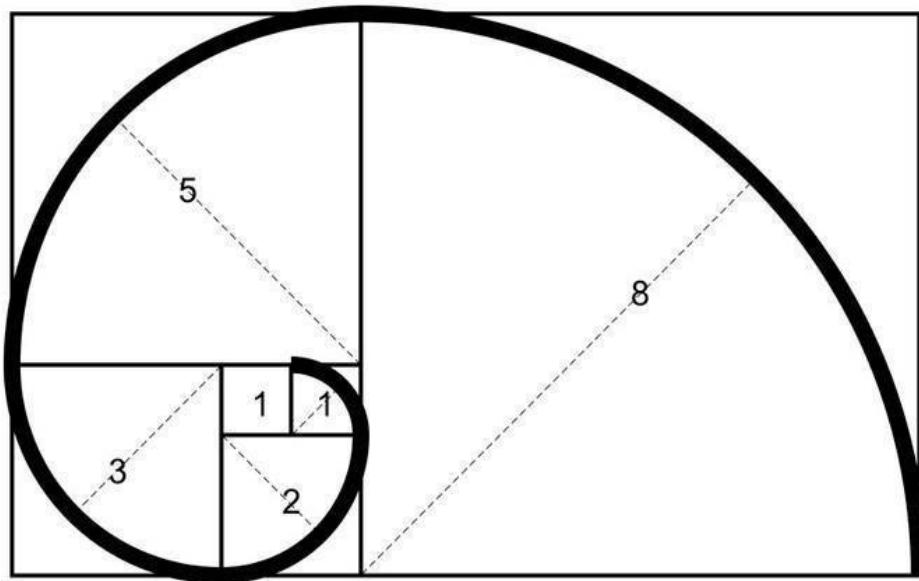
- The Fibonacci series is the sequence of numbers (also called Fibonacci numbers), where every number is the sum of the preceding two numbers, such that the first two terms are '0' and '1'.
- In some older versions of the series, the term '0' might be omitted. A Fibonacci series can thus be given as, 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, F_0 and second term, F_1 as '0' and '1', the third term here can be given as,
$$F_2 = 0 + 1 = 1$$

Similarly,

$$F_3 = 1 + 1 = 2$$

$$F_4 = 2 + 1 = 3$$

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:

$$F_n = F_{n-1} + F_{n-2}$$

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 $F_n = F_{n-1} + F_{n-2}$.

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)

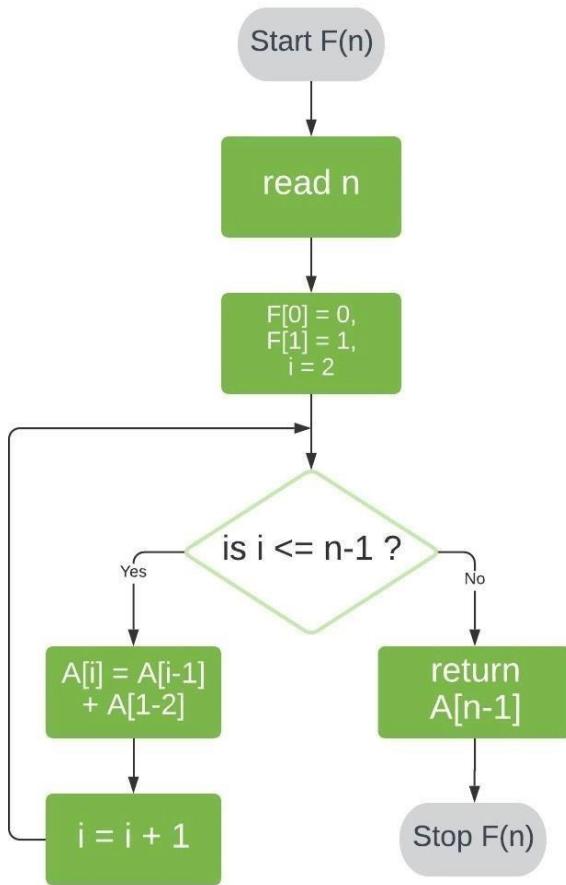
A simple method that is a direct recursive implementation of mathematical recurrence relation is given above.

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 F_n .

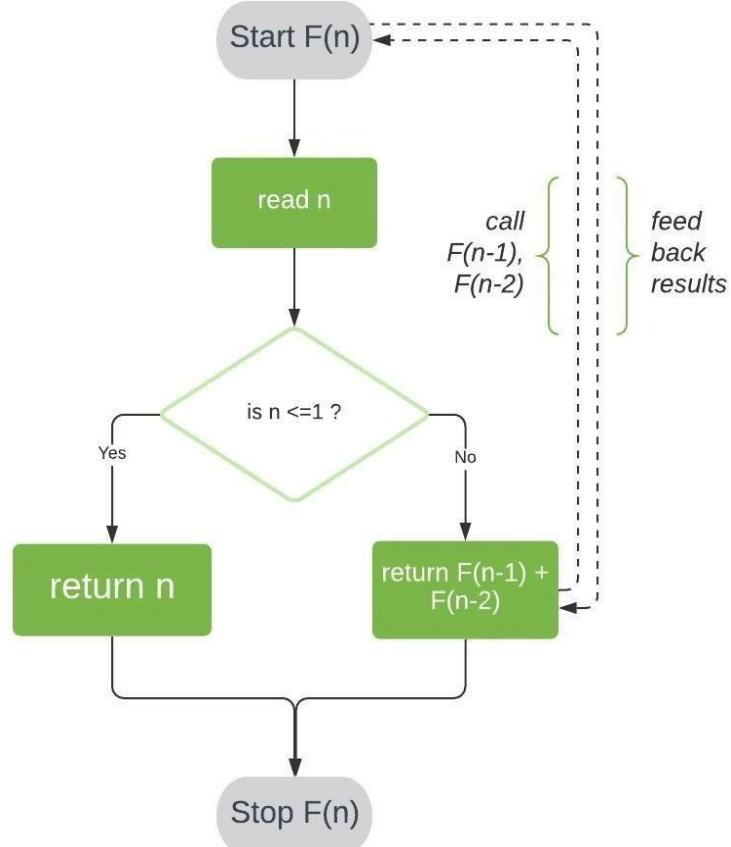
To evaluate $F(n)$ for $n > 1$, we can reduce our problem into two smaller problems of the same kind: $F(n-1)$ and $F(n-2)$. We can further reduce $F(n-1)$ and $F(n-2)$ to $F((n-1)-1)$ and $F((n-1)-2)$; and $F((n-2)-1)$ and $F((n-2)-2)$, respectively.

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("Please 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>

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.

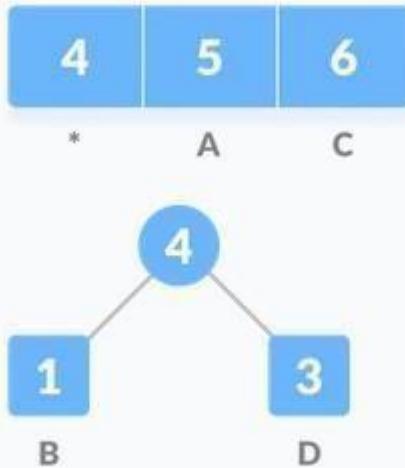


2. Sort the characters in increasing order of the frequency. These are stored in a priority queue Q.



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.

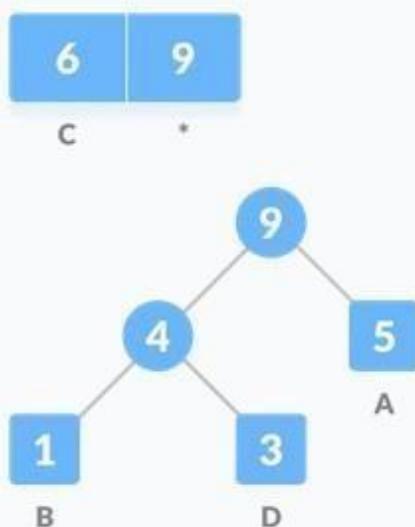


Getting the sum of the least numbers

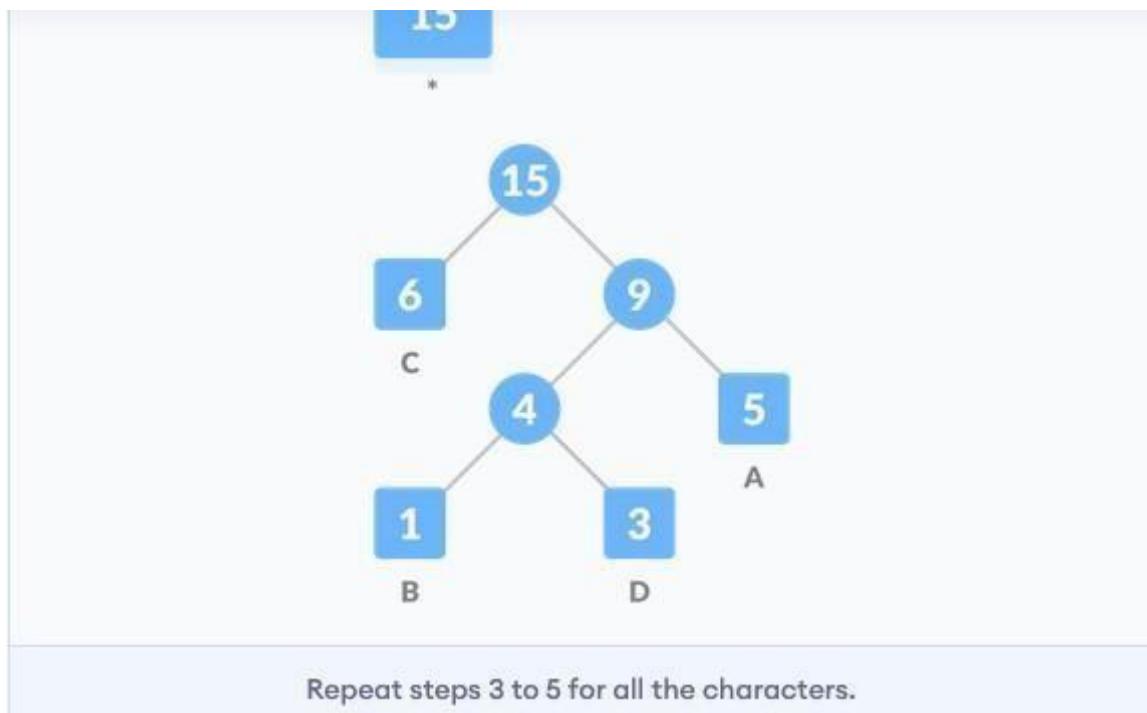
5. Remove these two minimum frequencies from Q and add the sum into the list of frequencies (* denote the internal nodes in the figure above).

6. Insert node z into the tree.

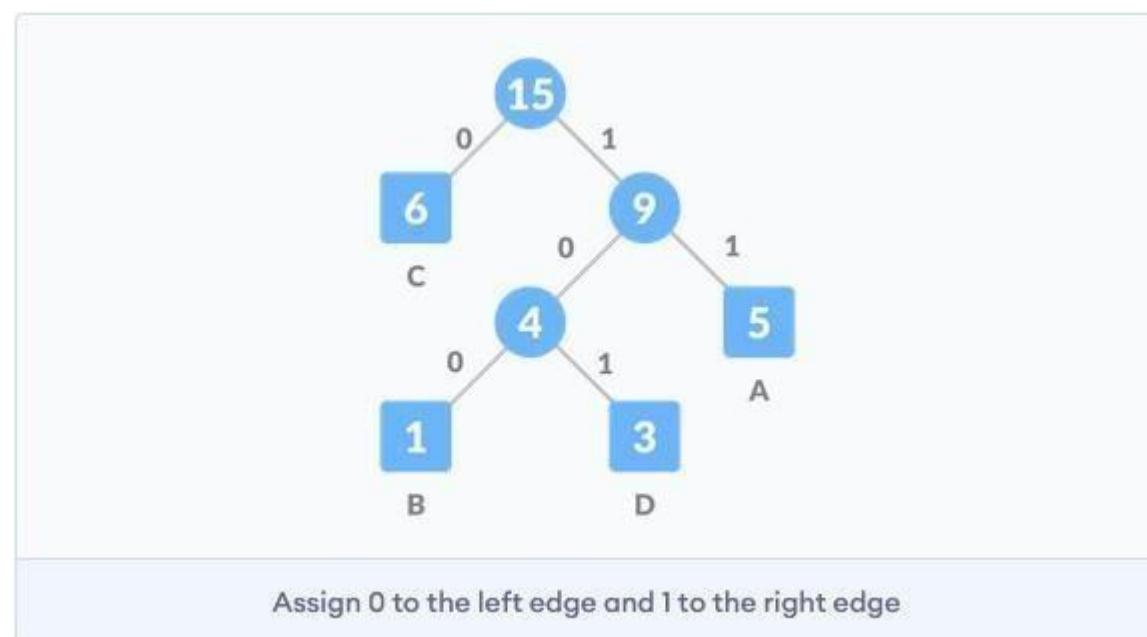
7. Repeat steps 3 to 5 for all the characters.



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

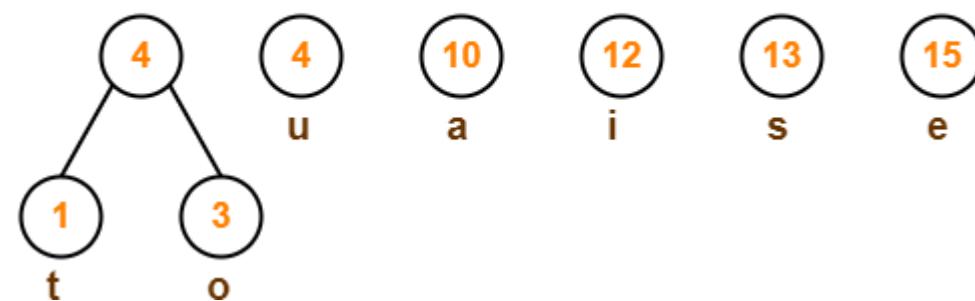
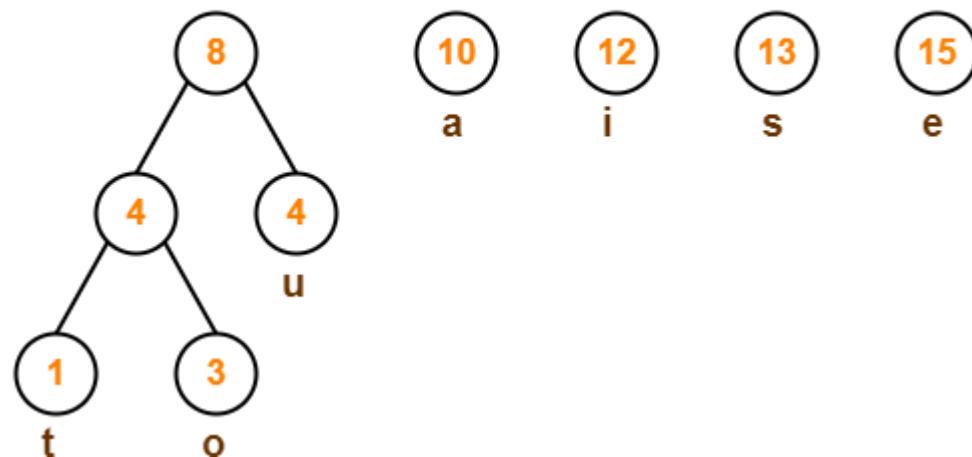
Without encoding, the total size of the string was 120 bits. After encoding the size is reduced to $32 + 15 + 28 = 75$.

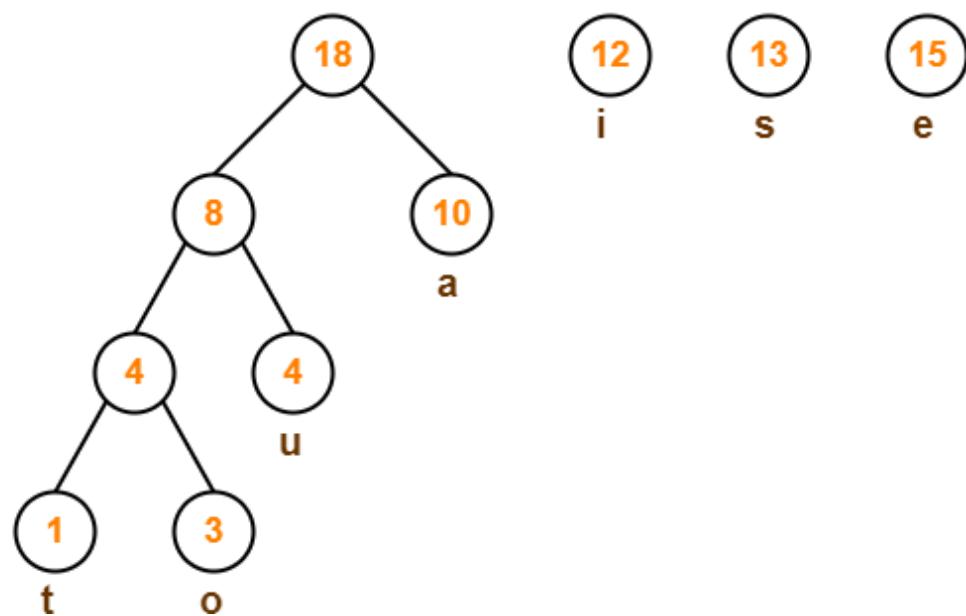
Example:

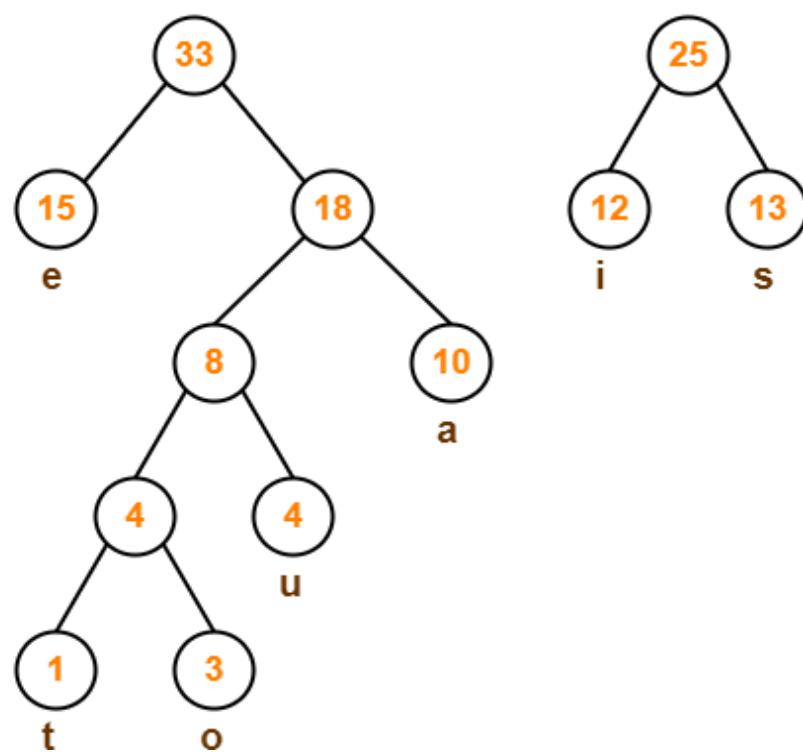
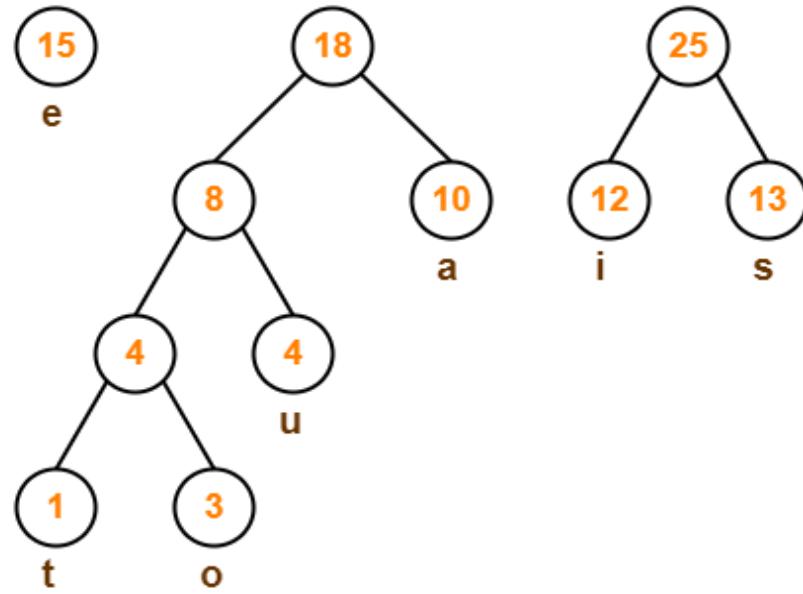
A file contains the following characters with the frequencies as shown. If Huffman Coding is used for data compression, determine-

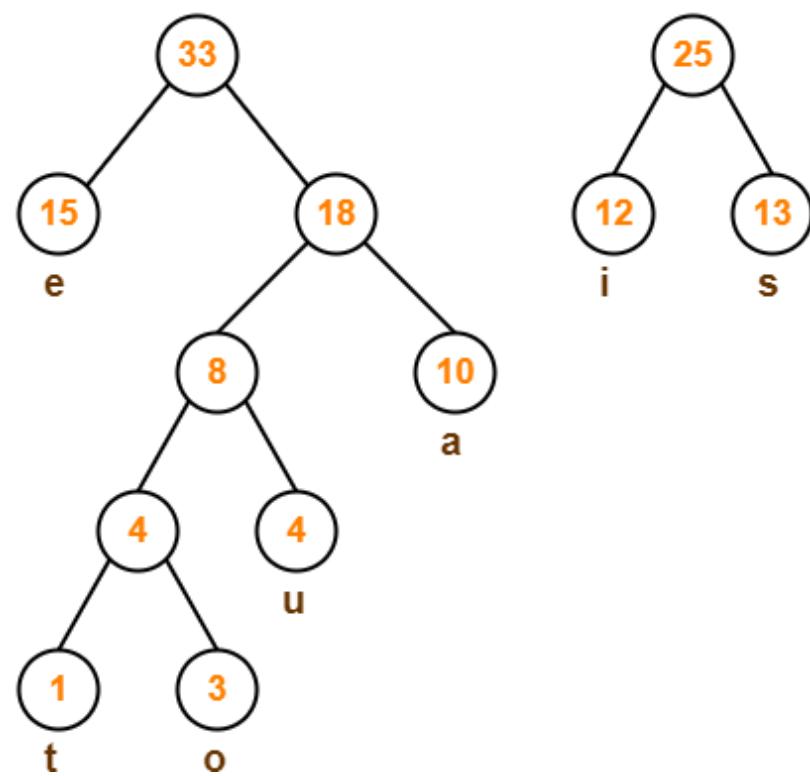
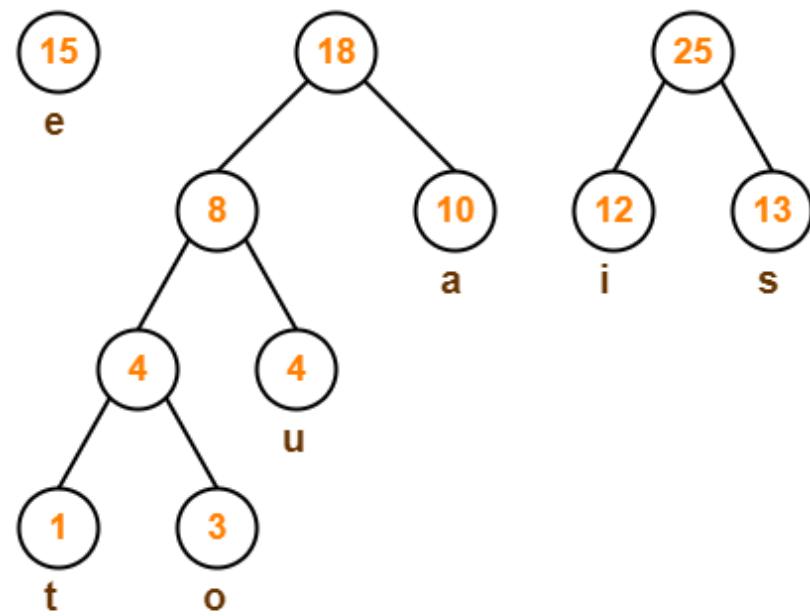
1. Huffman Code for each character
2. Average code length
3. Length of Huffman encoded message (in bits)

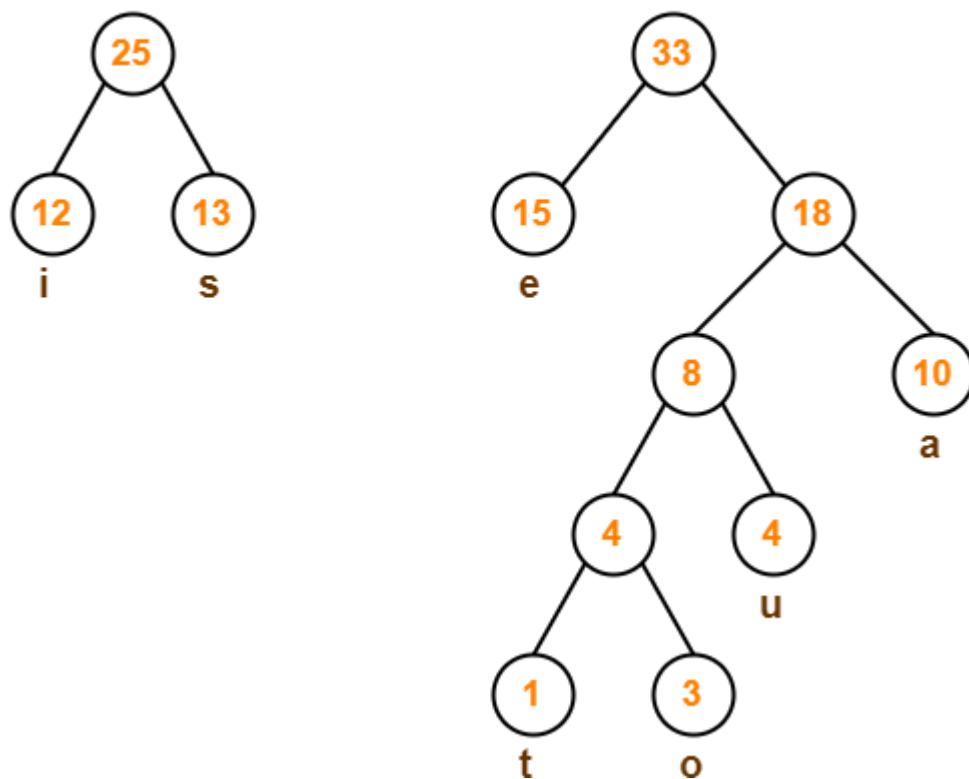
Characters	Frequencies
a	10
e	15
i	12
o	3
u	4
s	13
t	1

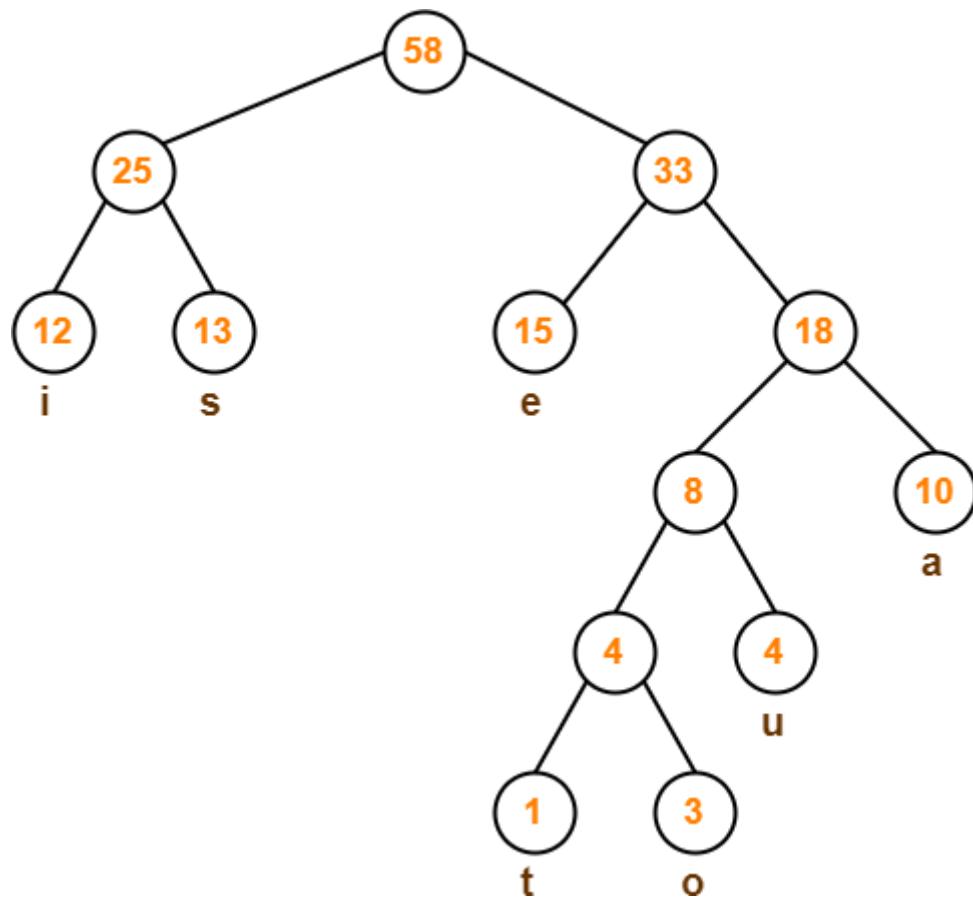
Step-01:**Step-02:****Step-03:**

Step-04:

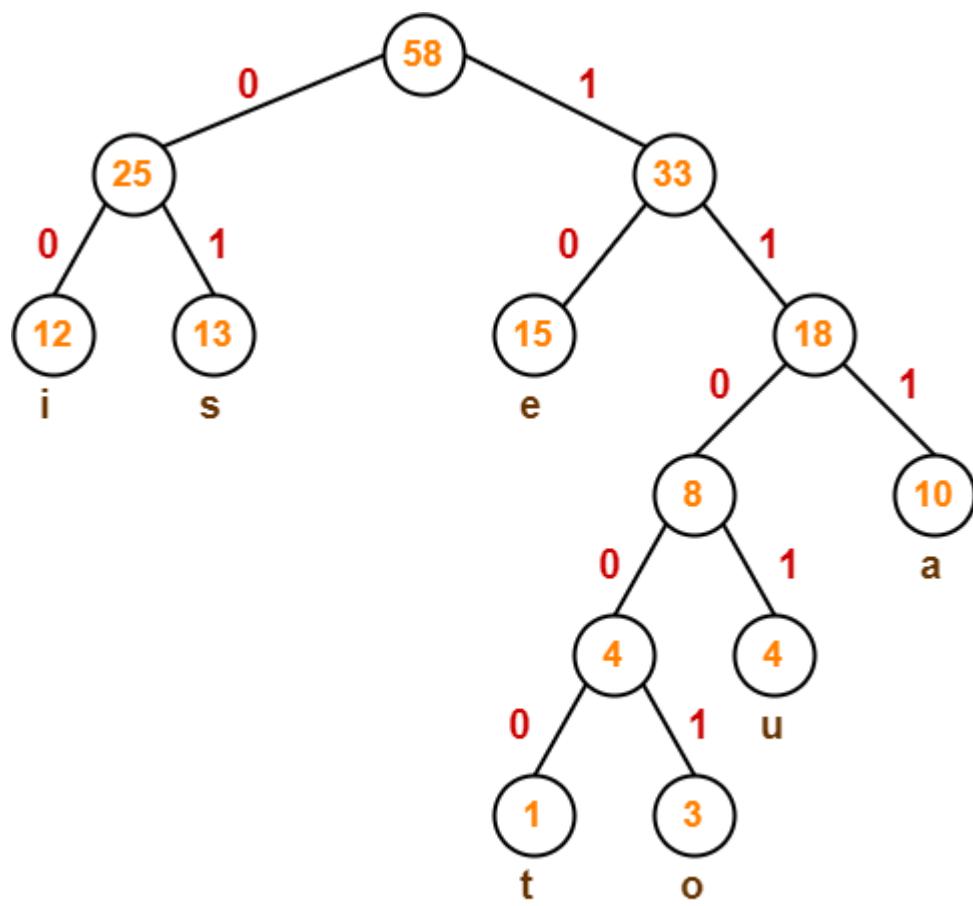
Step-06:

Step-06:

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 $2 \times (n-1)$ times if there are n nodes.
- As `extractMin()` calls `minHeapify()`, it takes $O(n\log n)$ time.

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

Code :-

```

class Node:
    def __init__(self, prob, symbol, left=None, right=None):
        # probability of symbol
        self.prob = prob

        # symbol
        self.symbol = symbol

        # left node
        self.left = left

        # right node
        self.right = right

        # tree direction (0/1)
        self.code = ''

    """ A helper function to calculate the probabilities of symbols in given data"""
def Calculate_Probability(data):
    symbols = dict()
    for element in data:
        if symbols.get(element) == None:
            symbols[element] = 1
        else:
            symbols[element] += 1
    return symbols

    """ A helper function to obtain the encoded output"""
def Output_Encoded(data, coding):
    encoding_output = []
    for c in data:
        print(coding[c], end = ' ')
        encoding_output.append(coding[c])

    string = ''.join([str(item) for item in encoding_output])
    return string

    """ A helper function to calculate the space difference between compressed and non compressed data"""
def Total_Gain(data, coding):
    before_compression = len(data) * 8 # total bit space to stor the data before compression
    after_compression = 0
    symbols = coding.keys()
    for symbol in symbols:
        count = data.count(symbol)
        after_compression += count * len(coding[symbol]) #calculate how many bit is required for each symbol to represent it
    print("Space usage before compression (in bits):", before_compression)
    print("Space usage after compression (in bits):", after_compression)

```

```

def Huffman_Encoding(data):
    symbol_with_probs = Calculate_Probability(data)
    symbols = symbol_with_probs.keys()
    probabilities = symbol_with_probs.values()
    print("symbols: ", symbols)
    print("probabilities: ", probabilities)

    nodes = []

    # converting symbols and probabilities into huffman tree nodes
    for symbol in symbols:
        nodes.append(Node(symbol_with_probs.get(symbol), symbol))

    while len(nodes) > 1:
        # sort all the nodes in ascending order based on their probability
        nodes = sorted(nodes, key=lambda x: x.prob)
        # for node in nodes:
        #     print(node.symbol, node.prob)

        # pick 2 smallest nodes
        right = nodes[0]
        left = nodes[1]

        left.code = 0
        right.code = 1

        # combine the 2 smallest nodes to create new node
        newNode = Node(left.prob+right.prob, left.symbol+right.symbol, left, right)

        nodes.remove(left)
        nodes.remove(right)
        nodes.append(newNode)

    huffman_encoding = Calculate_Codes(nodes[0])
    print(huffman_encoding)
    Total_Gain(data, huffman_encoding)
    encoded_output = Output_Encoded(data,huffman_encoding)
    print("Encoded output:", encoded_output)
    return encoded_output, nodes[0]

```

Output

Conclusion- In this way we have explored Concept of Huffman 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/>

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-

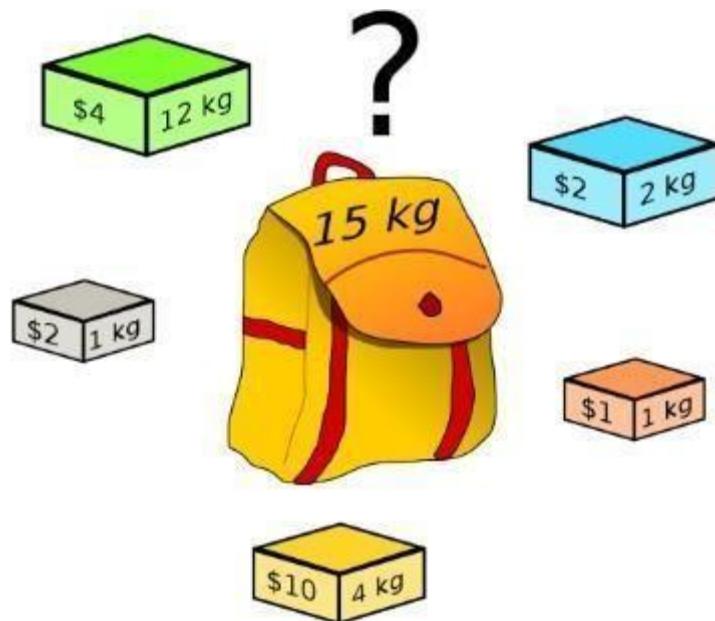
- A knapsack (kind of shoulder bag) with limited weight capacity.

- 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

$$n = 5$$

$$w = 60 \text{ kg}$$

$$(w_1, w_2, w_3, w_4, w_5) = (5, 10, 15, 22, 25)$$

$$(b_1, b_2, b_3, b_4, b_5) = (30, 40, 45, 77, 90)$$

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	\emptyset	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

$$= 160 + (20/22) \times 77$$

$$= 160 + 70$$

$$= 230 \text{ units}$$

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 $O(n\log n)$.
- Therefore, total time taken including the sort is $O(n\log n)$.

Code:-

```

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

def fractionalKnapsack(W, arr):

    # Sorting Item on basis of ratio
    arr.sort(key=lambda x: (x.value/x.weight), reverse=True)

    # Result(value in Knapsack)
    finalvalue = 0.0

    # Looping through all Items
    for item in arr:

        # If adding Item won't overflow,
        # add it completely
        if item.weight <= W:
            W -= item.weight
            finalvalue += item.value

        # If we can't add current Item,
        # add fractional part of it
        else:
            finalvalue += item.value * W / item.weight
            break

    # Returning final value
    return finalvalue

# Driver Code
if __name__ == "__main__":
    W = 50
    arr = [Item(60, 10), Item(100, 20), Item(120, 30)]

    # Function call
    max_val = fractionalKnapsack(W, arr)
    print(max_val)

```

Output

Maximum value we can obtain = 24

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/>

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

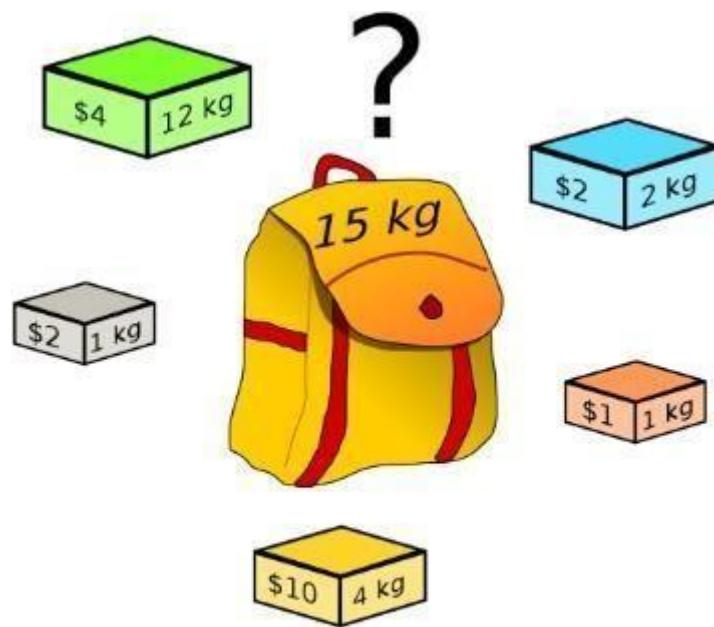
You are given the following-

- 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-

$$T(i, j) = \max \{ T(i-1, j), \text{value}_i + T(i-1, j - \text{weight}_i) \}$$

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

$$n = 4$$

$$w = 5 \text{ kg}$$

$$(w_1, w_2, w_3, w_4) = (2, 3, 4, 5)$$

$$(b_1, b_2, b_3, b_4) = (3, 4, 5, 6)$$

Solution-

Given

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

Step-01:

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

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

T-Table

Step-02:

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

$$T(i, j) = \max \{ T(i-1, j), \text{value}_i + T(i-1, j - \text{weight}_i) \}$$

Finding $T(1,1)$ -

We have,

- $i = 1$
- $j = 1$
- $(\text{value})_i = (\text{value})_1 = 3$
- $(\text{weight})_i = (\text{weight})_1 = 2$

Substituting the values, we get-

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

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

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

$$T(1,1) = 0$$

Finding T(1,2)-

We have,

- $i = 1$
- $j = 2$
- $(\text{value})_i = (\text{value})_1 = 3$
- $(\text{weight})_i = (\text{weight})_1 = 2$

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,

- $i = 1$
- $j = 3$
- $(\text{value})_i = (\text{value})_1 = 3$
- $(\text{weight})_i = (\text{weight})_1 = 2$

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,

- $i = 1$
- $j = 4$
- $(\text{value})_i = (\text{value})_1 = 3$
- $(\text{weight})_i = (\text{weight})_1 = 2$

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-

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

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

$$T(2,1) = T(1,1) \{ \text{Ignore } T(1,-2) \}$$

$$T(2,1) = 0$$

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-

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

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

$$T(2,2) = T(1,2) \{ \text{Ignore } T(1,-1) \}$$

$$T(2,2) = 3$$

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

Code :-

```
# code

# A Dynamic Programming based Python

# Program for 0-1 Knapsack problem

# Returns the maximum value that can

# be put in a knapsack of capacity W

def knapSack(W, wt, val, n):

    dp = [0 for i in range(W+1)]    # Making the dp array

    for i in range(1, n+1):    # taking first i elements

        for w in range(W, 0, -1): # starting from back, so that we also
        have data of

            # previous computation when taking i-1
            items

            if wt[i-1] <= w:

                # finding the maximum value

                dp[w] = max(dp[w], dp[w-wt[i-1]]+val[i-1])

    return dp[W]    # returning the maximum value of knapsack

# Driver code

val = [60, 100, 120]

wt = [10, 20, 30]

W = 50

n = len(val)

print(knapSack(W, wt, val, n))
```

Output
220

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

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/o-1-knapsack-problem-using-dynamic-programming-approach/>
- <https://www.youtube.com/watch?v=mMhC9vuA-7o>
- https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_fractional_knapsack.htm

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 $n \times n$ 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 $N \times N$ 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 $n=4$ then the problem is called the 4 queens problem.

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

Algorithm

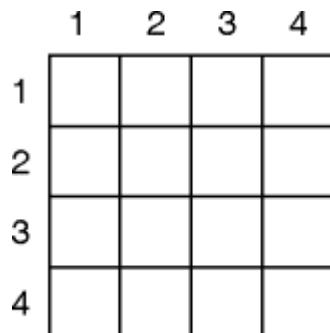
- 1) Start in the leftmost column
- 2) If all queens are place return true
- 3) Try all rows in the current column.

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 i th 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)

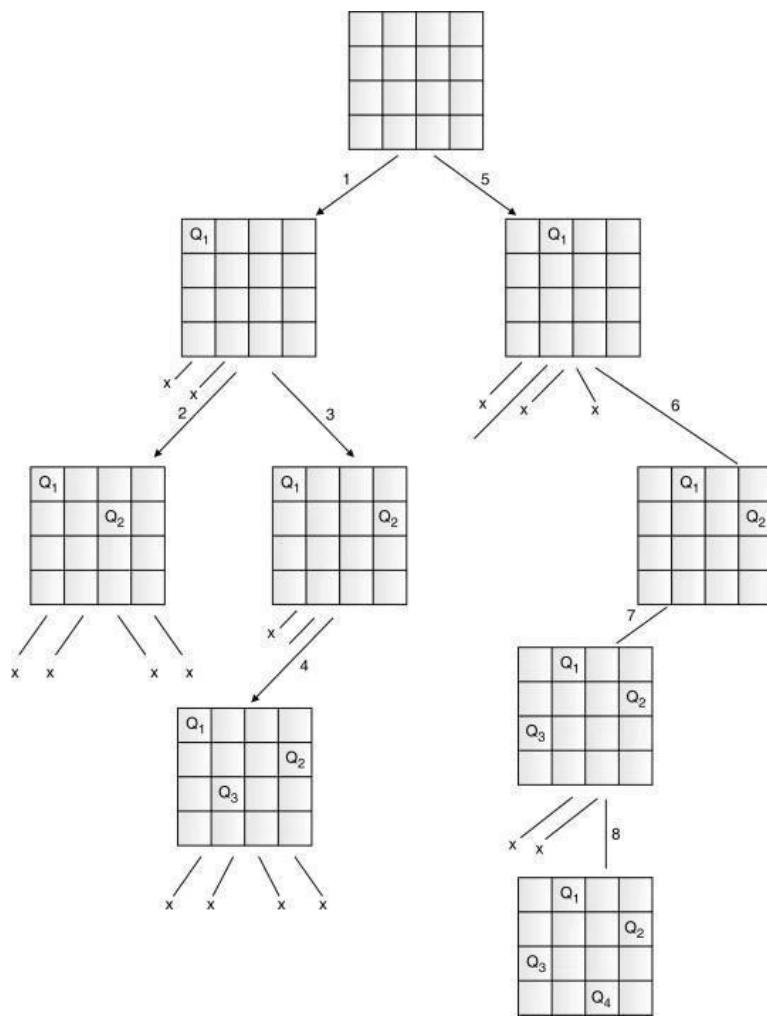
	1	2	3	4
1		Q ₁		
2				Q ₂
3	Q ₃			
4			Q ₄	

Fig. (a): Solution – 1

	1	2	3	4
1			Q ₁	
2	Q ₂			
3				Q ₃
4		Q ₄		

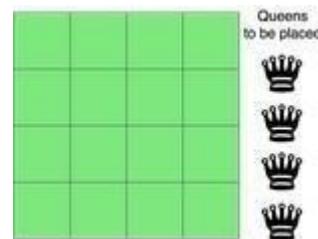
Fig. (b): Solution – 2

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

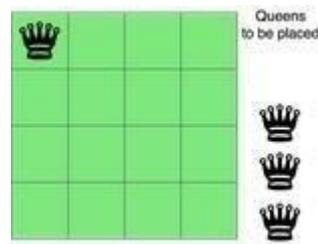


The solution of the 4-queen problem can be seen as four tuples (x_1, x_2, x_3, x_4) , where x_i represents the column number of queen Q_i . Two possible solutions for the 4-queen problem are $(2, 4, 1, 3)$ and $(3, 1, 4, 2)$.

Explanation :



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

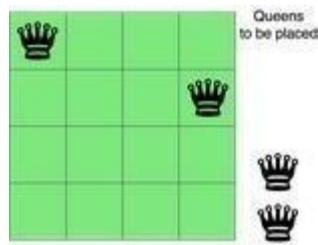


Now, the second step is to place the second queen in a safe position and then the third queen.

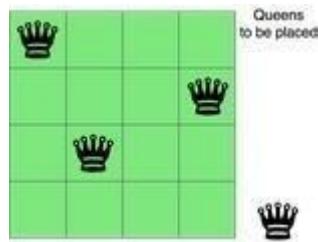


Now, you can see that there is no safe place where we can put the last queen. So, we will just change the position of the previous queen. And this is backtracking.

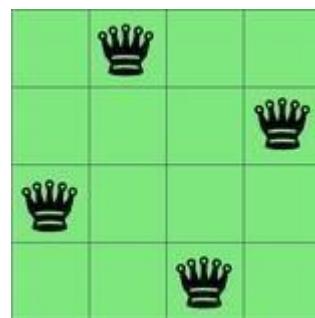
Also, there is no other position where we can place the third queen so we will go back one more step and change the position of the second queen.



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.

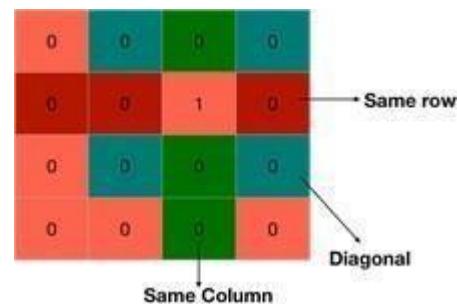


We need to check if a cell (i, j) is under attack or not. For that, we will pass these two in our function along with the chessboard and its size - IS-ATTACK(i, j, board, N).

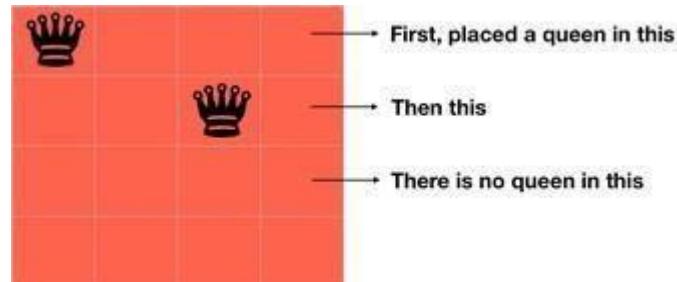
If there is a queen in a cell of the chessboard, then its value will be 1, otherwise, 0.

1	0	0	0
0	0	0	0
0	0	0	0

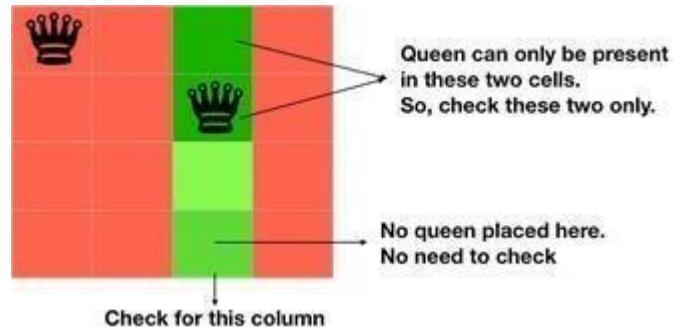
The cell (i,j) will be under attack in three condition - if there is any other queen in row i , if there is any other queen in the column j or if there is any queen in the diagonals.



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 $\text{board}[k][j]$ because only the rows from 1 to $i-1$ are filled.



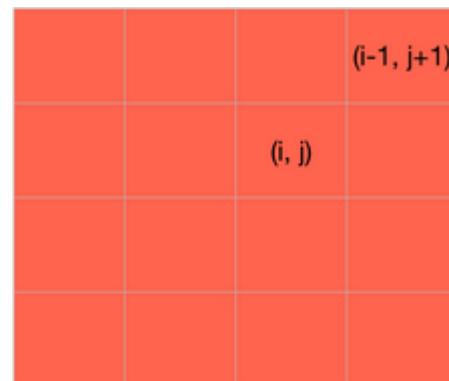
for k in 1 to $i-1$

```
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 $k \geq 1$ and $l \leq N$

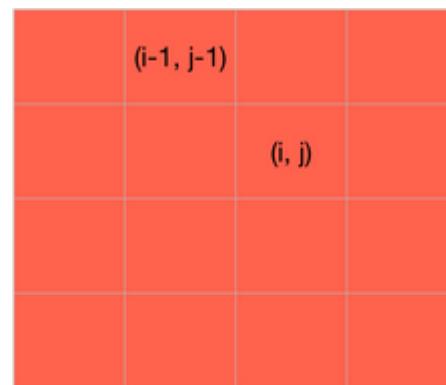
if $\text{board}[k][l] == 1$

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 $k \geq 1$ and $l \geq 1$

if $\text{board}[k][l] == 1$

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
```

Code :-

```
# Python3 program to solve N Queen
```

```
# Problem using backtracking
```

```
global N
```

```
N = 4
```

```
def printSolution(board):
```

```
    for i in range(N):
```

```
        for j in range(N):
```

```
            print(board[i][j], end = " ")
```

```
        print()
```

```
# A utility function to check if a queen can
```

```
# be placed on board[row][col]. Note that this
```

```
# function is called when "col" queens are
```

```
# already placed in columns from 0 to col -1.
```

```
# So we need to check only left side for
```

```
# attacking queens
```

```
def isSafe(board, row, col):
```

```
    # Check this row on left side
```

```
    for i in range(col):
```

```
        if board[row][i] == 1:
```

```
            return False
```

```
    # Check upper diagonal on left side
```

```
    for i, j in zip(range(row, -1, -1),
```

```
                    range(col, -1, -1)):
```

```
        if board[i][j] == 1:
```

```
            return False
```

```
    # Check lower diagonal on left side
```

```
    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):
```

```
    # base case: If all queens are placed
```

```
    # then return true
```

```
    if col >= N:
```

```
        return True
```

```
    # Consider this column and try placing
```

```
    # this queen in all rows one by one
```

```
    for i in range(N):
```

```
        if isSafe(board, i, col):
```

```
            # Place this queen in board[i][col]
```

```

board[i][col] = 1

# recur to place rest of the queens
if solveNQUtil(board, col + 1) == True:
    return True

# If placing queen in board[i][col]
# doesn't lead to a solution, then
# queen from board[i][col]
board[i][col] = 0

# if the queen can not be placed in any row in
# this column col then return false
return False

# This function solves the N Queen problem using
# Backtracking. It mainly uses solveNQUtil() to
# solve the problem. It returns false if queens
# cannot be placed, otherwise return true and
# placement of queens in the form of 1s.
# note that there may be more than one
# solutions, this function prints one of the
# feasible solutions.
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

# Driver Code
solveNQ()

```

Output:-

0	0	1	0
1	0	0	0
0	0	0	1
0	1	0	0

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/>

Assignment No : 6

MINI PROJECT 1

Theory :-

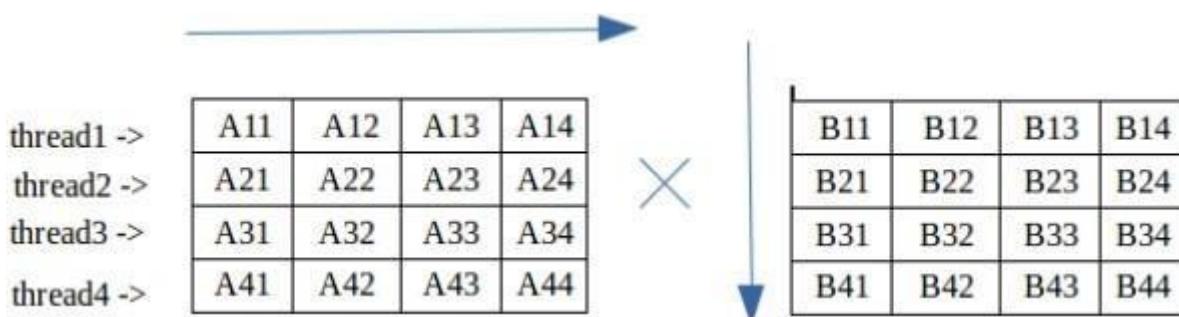
Multiplication of matrix does take time surely. Time complexity of matrix multiplication is $O(n^3)$ using normal matrix multiplication. And Strassen algorithm improves it and its time complexity is $O(n^{(2.8074)})$.

But, Is there any way to improve the performance of matrix multiplication using the normal method.

Multi-threading can be done to improve it. In multi-threading, instead of utilizing a single core of your processor, we utilizes all or more core to solve the problem.
We create different threads, each thread evaluating some part of matrix multiplication.

Depending upon the number of cores your processor has, you can create the number of threads required. Although you can create as many threads as you need, a better way is to create each thread for one core.

In second approach, we create a separate thread for each element in resultant matrix. Using `pthread_exit()` we return computed value from each thread which is collected by `pthread_join()`. This approach does not make use of any global variables.



Code :-

```

// CPP Program to multiply two matrix using pthreads
#include <bits/stdc++.h>
using namespace std;

// maximum size of matrix
#define MAX 4

// maximum number of threads
#define MAX_THREAD 4

int matA[MAX][MAX];
int matB[MAX][MAX];
int matC[MAX][MAX];
int step_i = 0;

void* multi(void* arg)
{
    int i = step_i++; // i denotes row number of resultant matC

    for (int j = 0; j < MAX; j++)
        for (int k = 0; k < MAX; k++)
            matC[i][j] += matA[i][k] * matB[k][j];
}

// Driver Code
int main()
{
    // Generating random values in matA and matB
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++) {
            matA[i][j] = rand() % 10;
            matB[i][j] = rand() % 10;
        }
    }
}

```

```
    matB[i][j] = rand() % 10;  
}
```

```
}

// Displaying matA
cout << endl
    << "Matrix A" << endl;
for (int i = 0; i < MAX; i++) {
    for (int j = 0; j < MAX; j++)
        cout << matA[i][j] << " ";
    cout << endl;
}

// Displaying matB
cout << endl
    << "Matrix B" << endl;
for (int i = 0; i < MAX; i++) {
    for (int j = 0; j < MAX; j++)
        cout << matB[i][j] << " ";
    cout << endl;
}

// declaring four threads
pthread_t threads[MAX_THREAD];

// Creating four threads, each evaluating its own part
for (int i = 0; i < MAX_THREAD; i++) {
    int* p;
    pthread_create(&threads[i], NULL, multi, (void*)(p));
}

// joining and waiting for all threads to complete
for (int i = 0; i < MAX_THREAD; i++)
    pthread_join(threads[i], NULL);

// Displaying the result matrix
cout << endl
    << "Multiplication of A and B" << endl;
for (int i = 0; i < MAX; i++) {
    for (int j = 0; j < MAX; j++)
        cout << matC[i][j] << " ";
    cout << endl;
}

return 0;
}
```

Group B

Assignment No : 7

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:

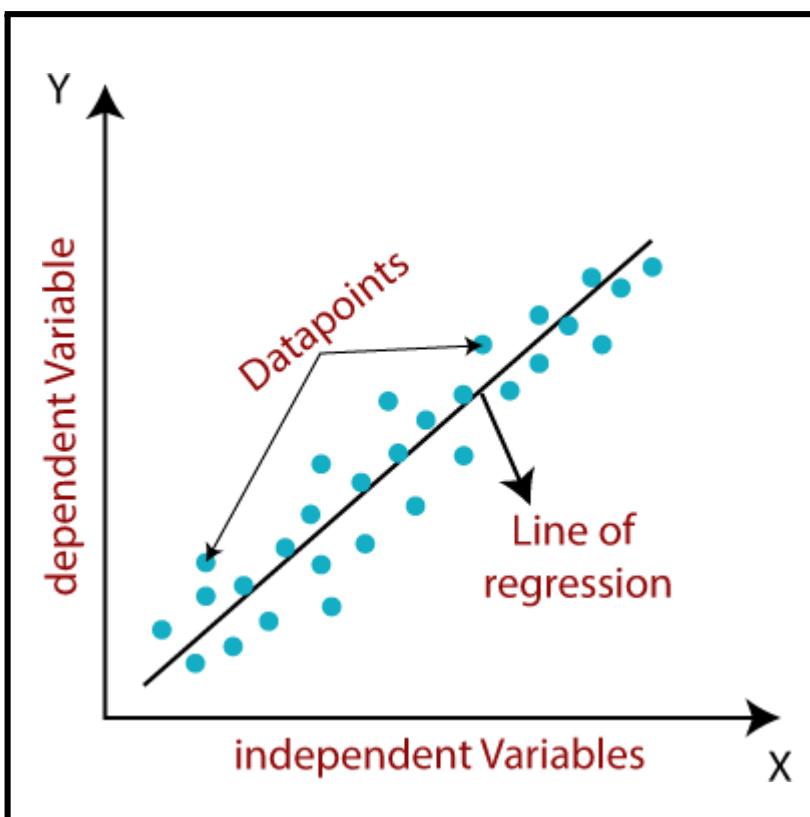
- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

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 (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

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



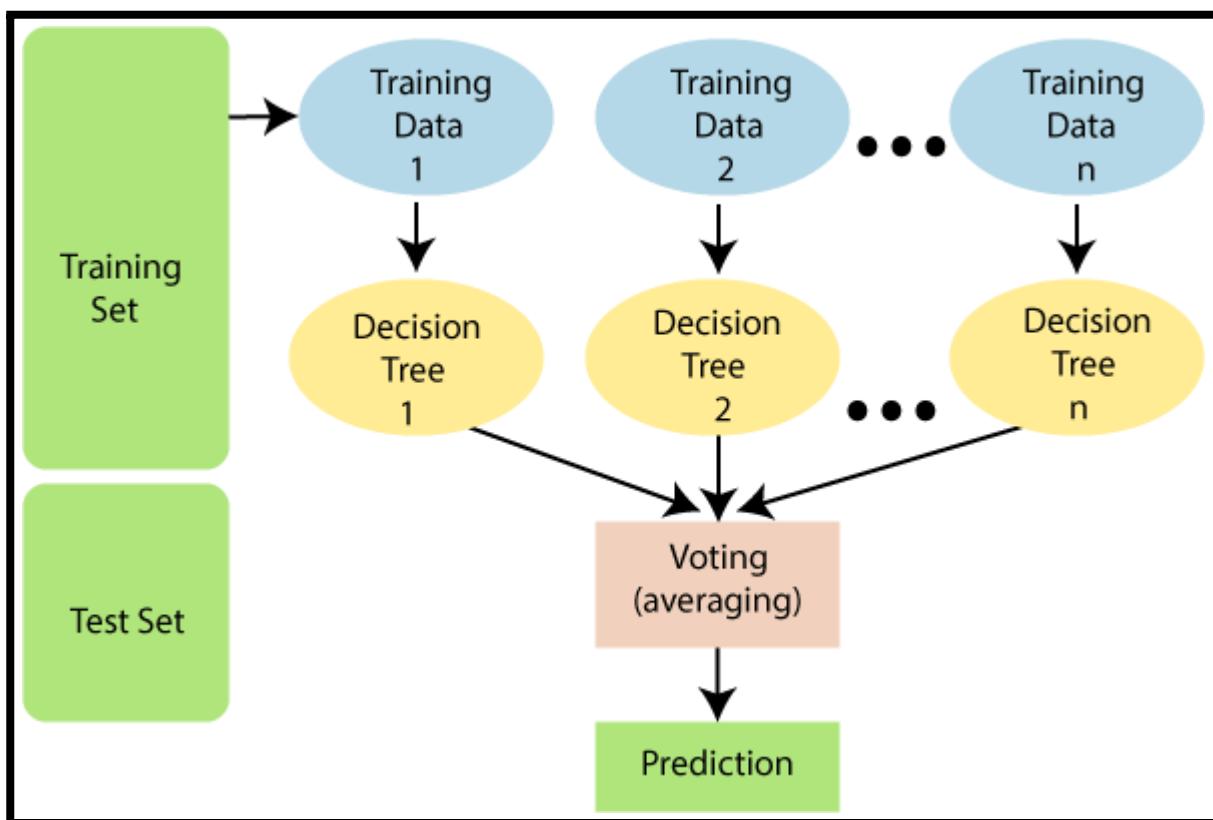
Random Forest Regression Models:

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

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

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

The greater number of trees in the forest leads to higher accuracy and prevents the problem of 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.N	Parameter	Description
o		

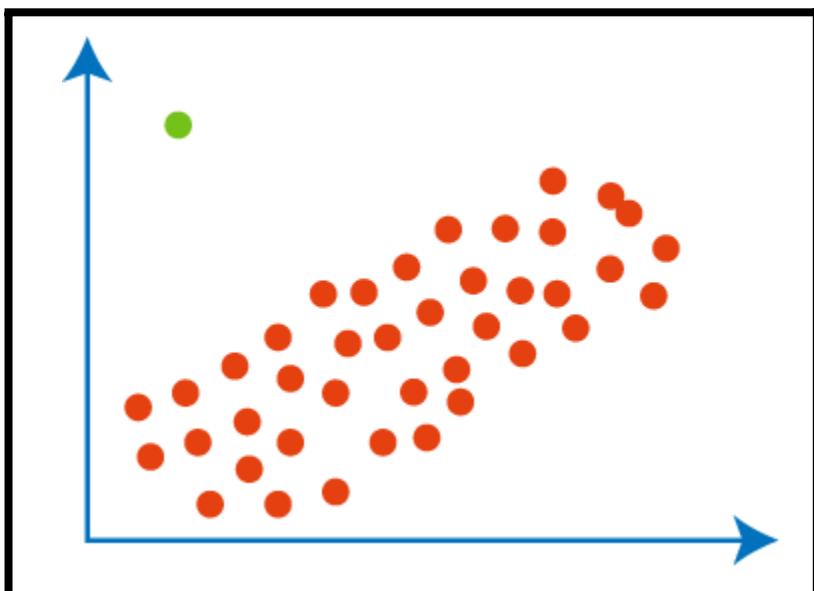
1.	x	It is a vector or a formula.
2.	data	It is the data frame.
3.	notch	It is a logical value set as true to draw a notch.
4.	varwidth	It is also a logical value set as true to draw the width of the boxes same as the sample size.
5.	names	It is the group of labels that will be printed under each boxplot.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

Outliers:

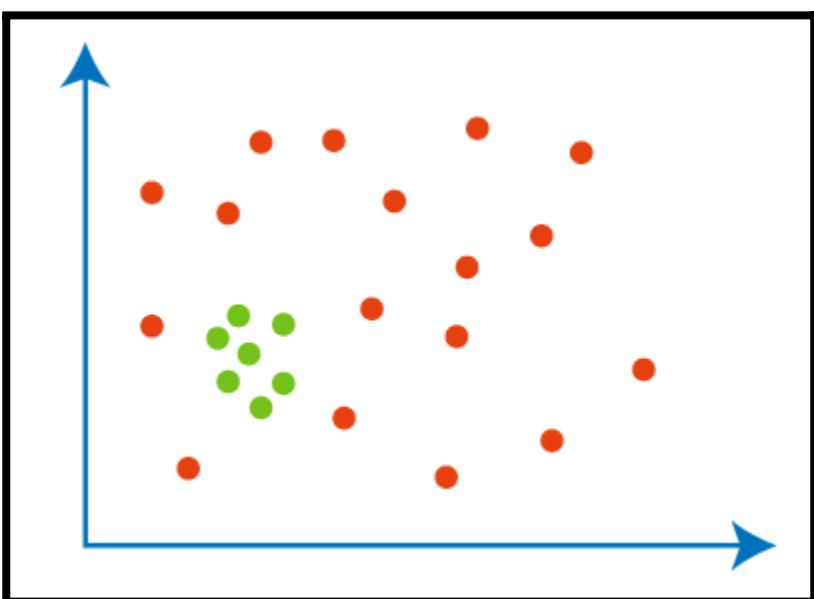
As the name suggests, "outliers" refer to the data points that exist outside of what is to be expected. The major thing about the outliers is what you do with them. If you are going to analyze any task to analyze data sets, you will always have some assumptions based on how this data is generated. If 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.

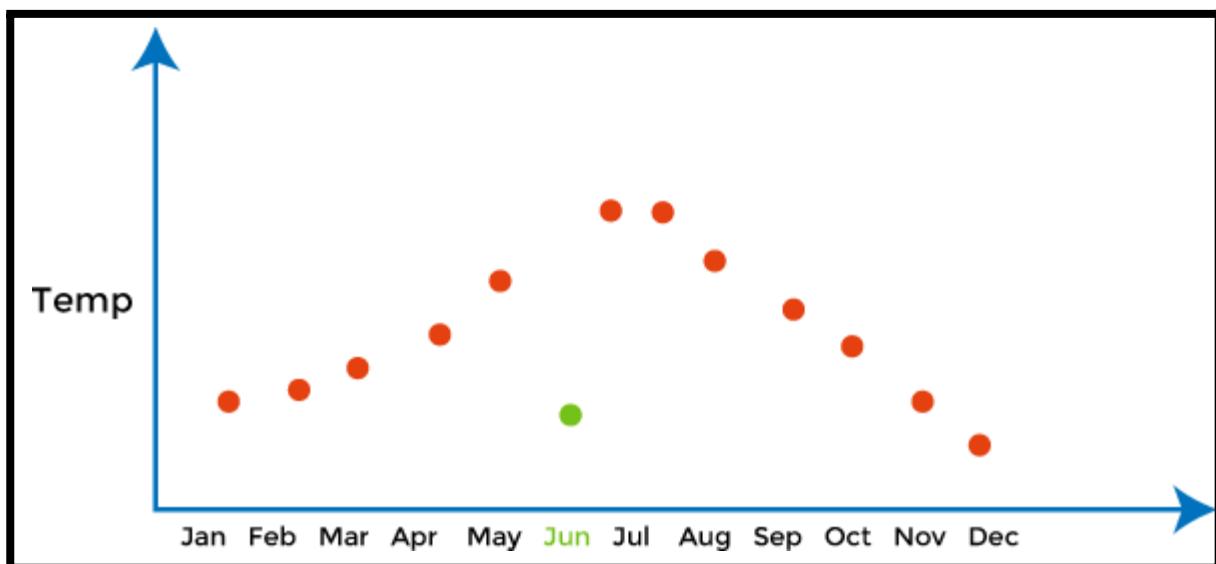
**Collective Outliers**

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

Code :- <https://www.kaggle.com/code/proxzima/uber-fare-price-prediction>

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. Define Outliers?
3. What is Linear Regression?
4. What is Random Forest Algorithm?
5. Explain: pandas, numpy?

Write-up	Correctness of Program	Documentation of Program	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
4	4	4	4	4	20	

Expected Date of Completion: Actual Date of Completion:

Group B
Assignment No : 8

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

Code :- <https://www.kaggle.com/code/mfaisalqureshi/email-spam-detection-98-accuracy/notebook>

Group B
Assignment No : 9

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:

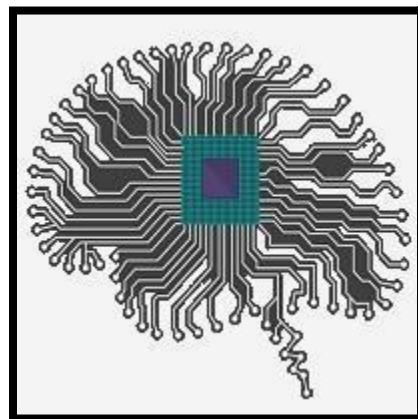
Students should be able to distinguish the feature and target set and divide the data set into training and test sets and normalize them and students should build the model on the basis of that.

Prerequisite:

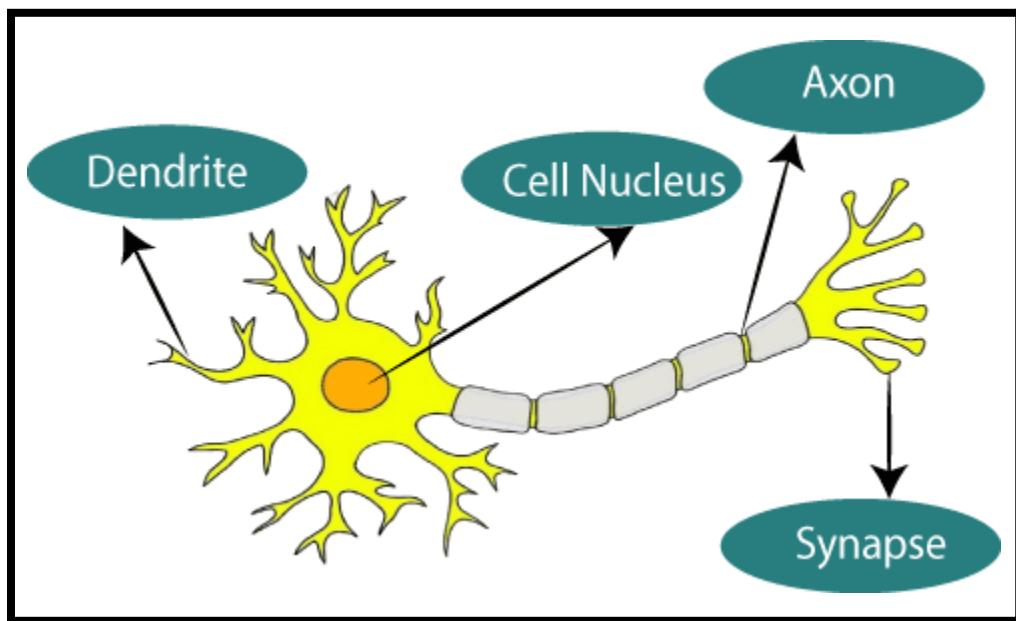
1. Basic knowledge of Python
2. Concept of Confusion Matrix

Contents of the Theory:

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

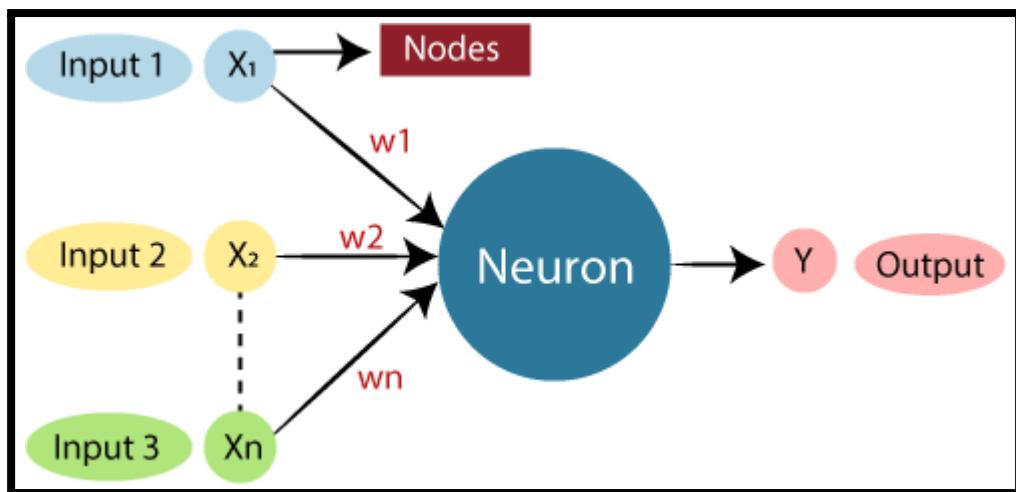
Artificial Neural Network:

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



The figure illustrates the typical diagram of Biological Neural Network.

The typical Artificial Neural Network looks something like the given figure.



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

Relationship between Biological neural network and artificial neural network:

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

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

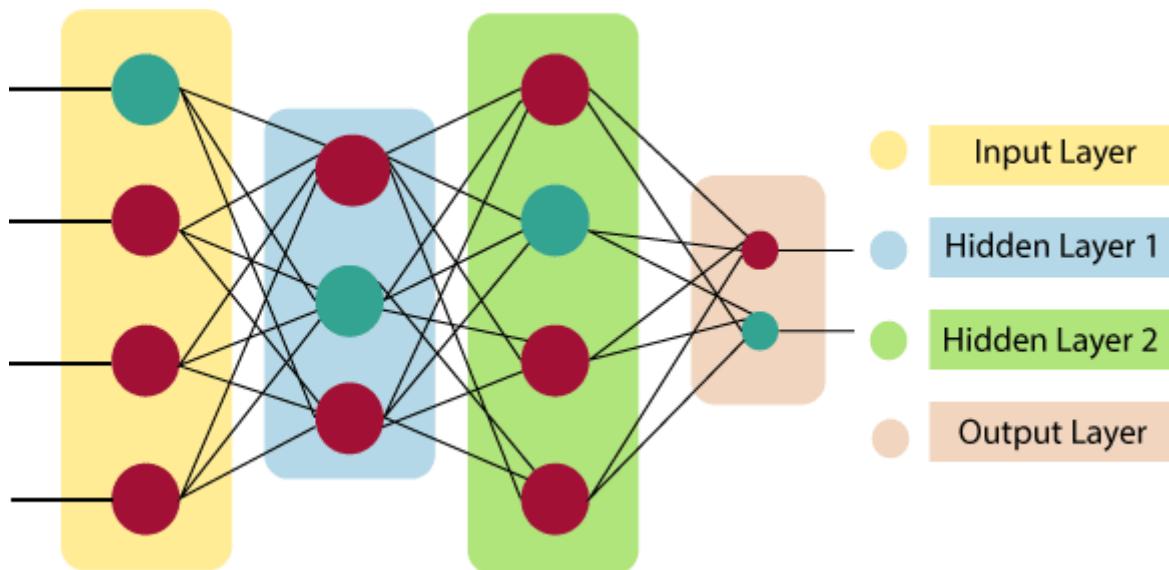
There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly.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

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

The architecture of an artificial neural network:

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Let's 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 and hidden features and patterns.

Output Layer:

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

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

$$\sum_{i=1}^n w_i * x_i + b$$

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

Keras:

Keras is an open-source high-level Neural Network library, which is written in Python and 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 acts as a high-level API wrapper for the low-level API, which lets it run on TensorFlow, CNTK, or Theano.

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

**TensorFlow:**

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: $X_n = (X -$

$$X_{\text{minimum}}) / (X_{\text{maximum}} - X_{\text{minimum}})$$

Where,

- X_n = Value of Normalization
- X_{maximum} = Maximum value of a feature
- X_{minimum} = 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.

$$X_n = (X - X_{\min}) / (X_{\max} - X_{\min}) \text{ ----- formula}$$

Put $X = X_{\min}$ in above formula, we get;

$$X_n = X_{\min} - X_{\min} / (X_{\max} - X_{\min}) X_n = 0$$

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

$$X_n = (X - X_{\min}) / (X_{\max} - X_{\min}) \text{ Put } X \\ = X_{\max} \text{ in above formula, we get;}$$

$$X_n = X_{\max} - X_{\min} / (X_{\max} - X_{\min}) X_n = 1$$

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

Normalization techniques in Machine Learning

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

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

- **Standardization scaling:**

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

Hence, standardization can be expressed as follows:

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

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

However, unlike Min-Max scaling technique, feature values are not restricted to a specific 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 networks**. 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 classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an **error matrix**. Some features of Confusion matrix are given below:

- For the 2 prediction classes of classifiers, the matrix is of 2*2 table, for 3 classes, it 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:

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

The above table has the following cases:

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

Need for Confusion Matrix in Machine learning

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

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

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

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

From the above example, we can conclude that:

- The table is given for the two-class classifier, which has two predictions "Yes" and "NO." Here, Yes defines that patient has the disease, and No defines that patient does not have that disease.
- The classifier has made a total of **100 predictions**. Out of 100 predictions, **89 are true predictions**, and **11 are incorrect predictions**.
- The model has given prediction "yes" for 32 times, and "No" for 68 times. Whereas the actual "Yes" was 27, and actual "No" was 73 times.

Calculations using Confusion Matrix:

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

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

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

- Misclassification rate:** It is also termed as Error rate, and it defines how often the model gives the wrong predictions. The value of error rate can be calculated as the number of incorrect

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

predictions to all number of the predictions made by the classifier. The formula is given below:

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

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

- **Recall:** It is defined as the out of total positive classes, how our model predicted correctly.

The recall must be as high as possible.

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

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

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

Other important terms used in Confusion Matrix:

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

Code :- <https://www.kaggle.com/code/jaysadguru00/starter-bank-customer-churn-modeling-6dbfe05e-a>

Conclusion:

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

Assignment Questions:

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

Group B

Assignment No : 10

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

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

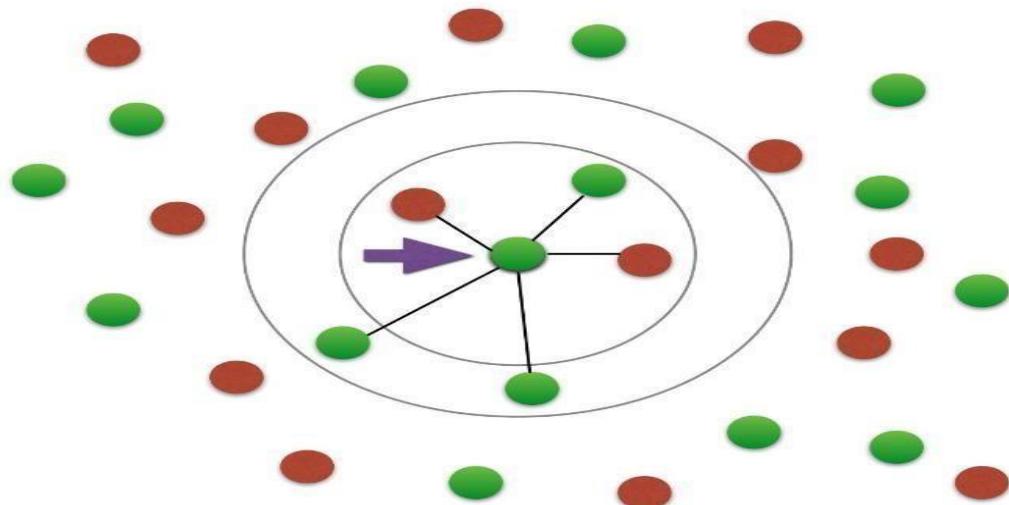
Link for Dataset: [Diabetes predication system with KNN algorithm | Kaggle](#)

Objective of the Assignment:

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

Prerequisite:

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



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

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

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

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

Reading in the training data

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

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

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

Next, let’s see how much data we have. We will call the ‘shape’ function on our dataframe to see how many rows and columns there are in our data. The rows indicate the number of patients and the columns indicate the number of features (age, weight, etc.) in the dataset for each patient. 1

```
#check number of rows and columns in dataset
df.shape
```

Op → (768,9)

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

Split up the dataset into inputs and targets

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

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

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

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

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

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

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

Split the dataset into train and test data

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

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

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

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

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

Building and training the model

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

```
from sklearn.neighbors import KNeighborsClassifier# Create KNN classifier  
knn = KNeighborsClassifier(n_neighbors = 3)# Fit the classifier to the data  
knn.fit(X_train,y_train)
```

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

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

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

Testing the model

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

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

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

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

Now let’s see how 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)
```

0.66883116883116878

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

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

k-Fold Cross-Validation

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

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

Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 1
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 2
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 3
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 4
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 5

Training data Test data

[5-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 get a better idea on how the model will perform on unseen data.

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

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

```
from sklearn.model_selection import cross_val_score
import numpy as np #create a new KNN model
knn_cv = KNeighborsClassifier(n_neighbors=3) #train model with cv of 5
cv_scores = cross_val_score(knn_cv, X, y, cv=5) #print each cv score (accuracy) and average them
print(cv_scores)
print('cv_scores mean:{}'.format(np.mean(cv_scores)))
[ 0.68181818  0.69480519  0.75324675  0.75163399  0.68627451]
cv_scores mean:0.7135557253204311
```

Bharati Vidyapeeth's College Of Engineering Lavale Pune.

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

Hypertuning model parameters using GridSearchCV

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

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

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

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

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

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

After training, we can check which of our values for ‘n_neighbors’ that we tested performed the best.

To do this, we will call ‘best_params_’ on our model.
#check top performing n_neighbors value
knn_gscv.best_params_

{'n_neighbors': 14}

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

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

0.7578125

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

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

Conclusion:

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

Assignment No : 11

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

Dataset Description: The data includes the following features:

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

Objective of the Assignment:

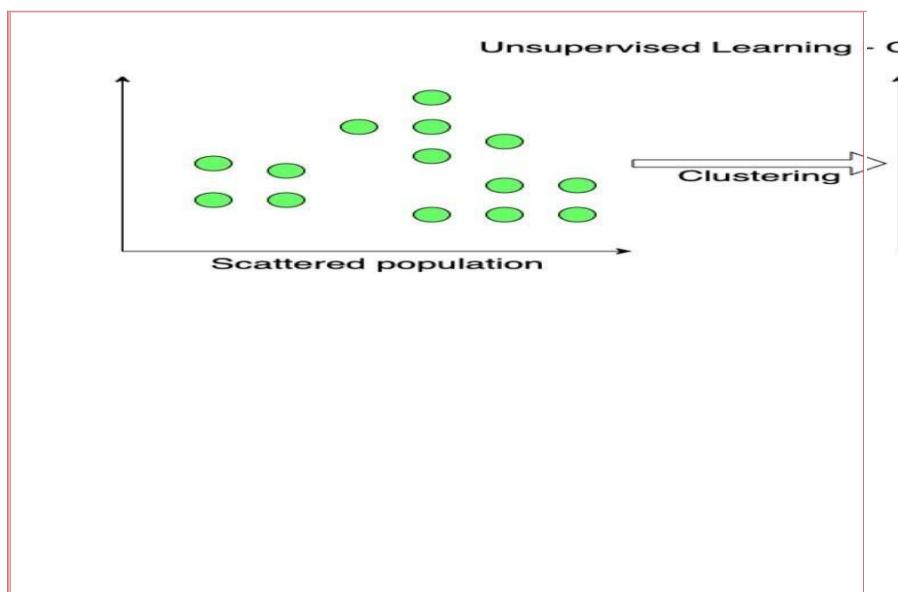
Students should able to understand how to use unsupervised learning to segment different-different clusters or groups and used to them to train your model to predict future things.

Prerequisite:

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

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

Bharati Vidyapeeth's College Of Engineering Lavale Pune.



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

Uses of Clustering

Marketing:

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

Real Estate:

Clustering can be used to understand and divide various property locations based on value and importance. Clustering algorithms can process through the data and identify various groups of property on the basis of probable price.

BookStore and Library management:

Libraries and Bookstores can use Clustering to better manage the book database. With proper book ordering, better operations can be implemented.

Document Analysis:

Often, we need to group together various research texts and documents according to similarity. And in such cases, we don't have any labels. Manually labelling large amounts of data is also not possible. Using clustering, the algorithm can process the text and group it into different themes.

These are some of the interesting use cases of clustering.

K-Means Clustering

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

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

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

K-Means is very easy and simple to implement. It is highly scalable, can be applied to both small and large datasets. There is, however, a problem with choosing the number of clusters or K. Also, with the increase in dimensions, stability decreases. But, overall K Means is a simple and robust algorithm that makes clustering very easy.

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

The necessary libraries are imported.

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

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

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

```
data.head()
```

So let us have a look at the data.

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

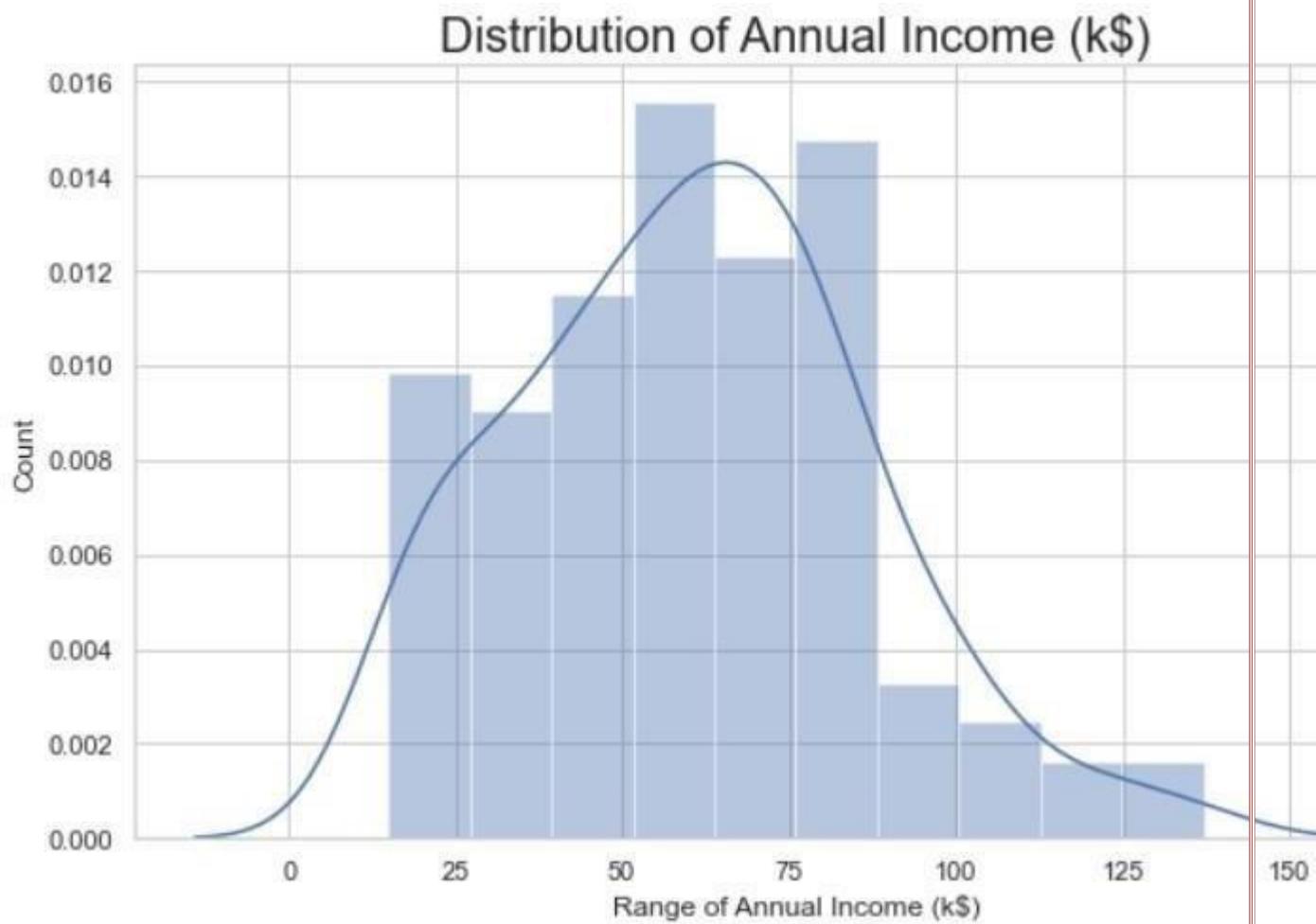
```
data.corr()
```

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

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

Annual Income Distribution:

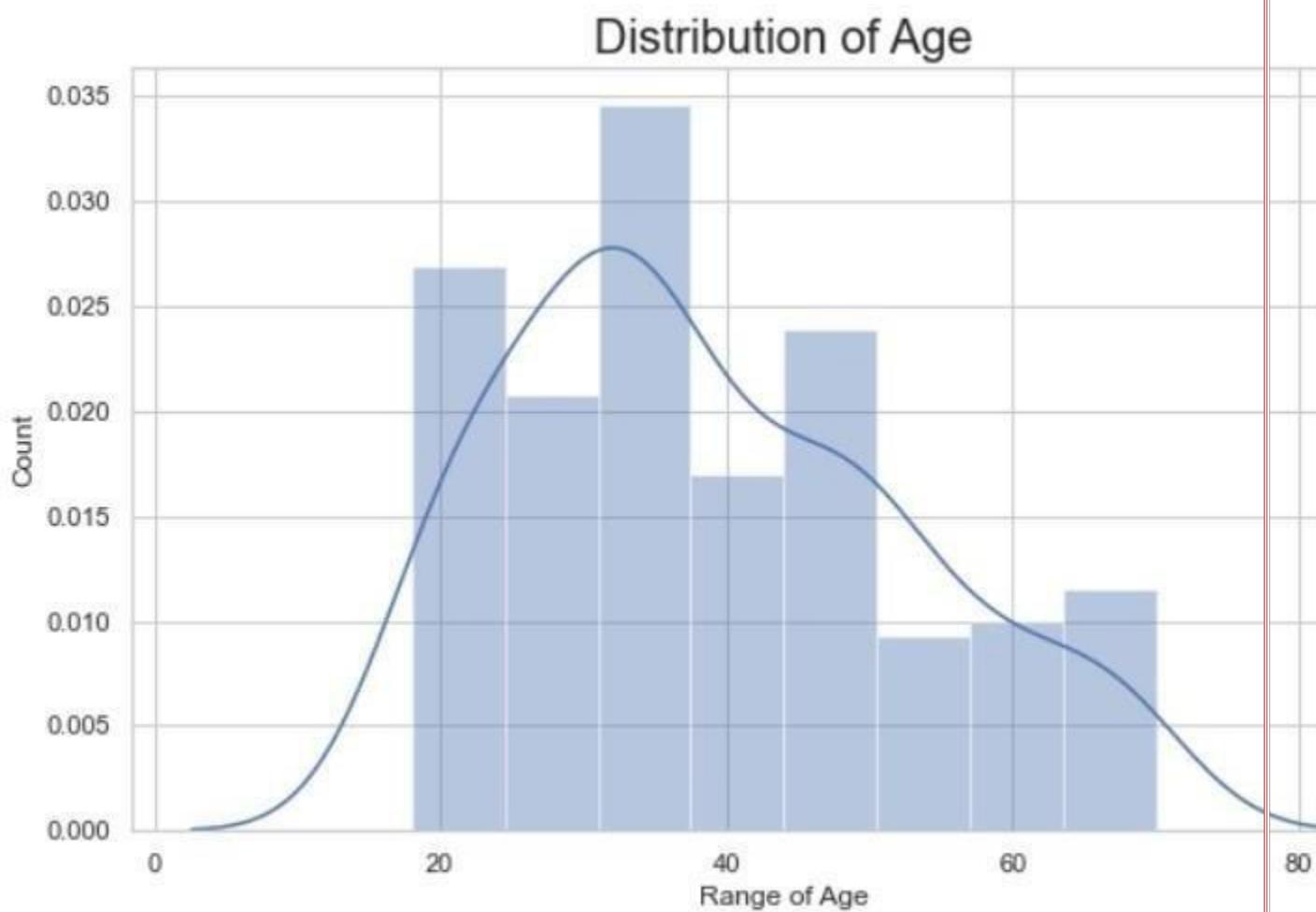
```
#Distribution of Annual Income
plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(data['Annual Income (k$)'])
plt.title('Distribution of Annual Income (k$)', fontsize = 20)
plt.xlabel('Range of Annual Income (k$)')
plt.ylabel('Count')
```



Most of the annual income falls between 50K to 85K.

Age Distribution:

```
#Distribution of age
plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(data['Age'])
plt.title('Distribution of Age', fontsize = 20)
plt.xlabel('Range of Age')
plt.ylabel('Count')
```



There are customers of a wide variety of ages.

Spending Score Distribution:

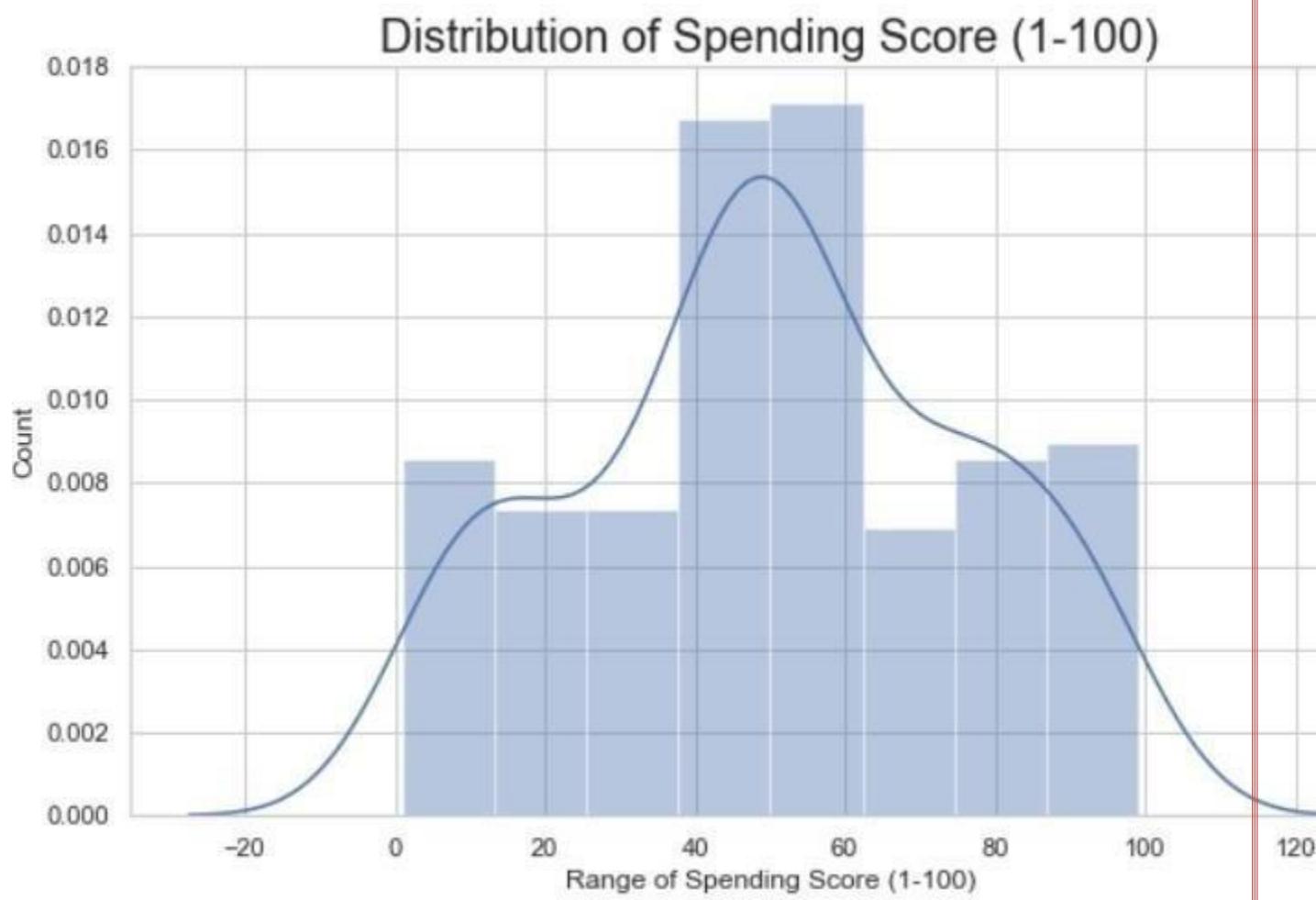
Earn Rewards by Writing and Sharing Data Science Knowledge



Learn | Write | Earn

Assured INR 2000 (\$26) for every published article! [Register Now](#)
#Distribution of spending score

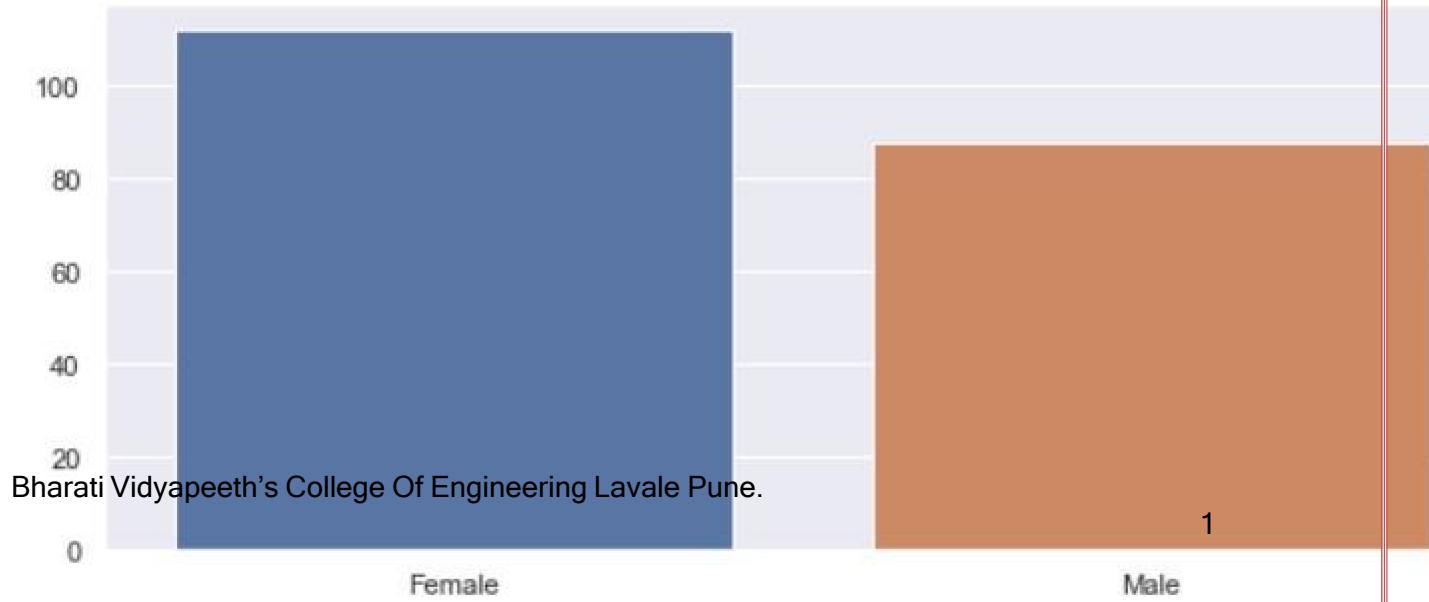
```
plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.distplot(data['Spending Score (1-100)'])
plt.title('Distribution of Spending Score (1-100)', fontsize = 20)
plt.xlabel('Range of Spending Score (1-100)')
plt.ylabel('Count')
```



The maximum spending score is in the range of 40 to 60.

Gender Analysis:

```
genders = data.Gender.value_counts()  
sns.set_style("darkgrid")  
plt.figure(figsize=(10,4))  
sns.barplot(x=genders.index, y=genders.values)  
plt.show()
```



More female customers than male.

I have made more visualizations. Do have a look at the GitHub link at the end to understand the data analysis and overall data exploration.

Clustering based on 2 features

Spending Score (1-100) vs Annual Income (k\$)

First, we work with two features only, annual income and spending score.

#We take just the Annual Income and Spending score

df1=data[["CustomerID", "Gender", "Age", "Annual Income (k\$)", "Spending Score (1-100)"]]

X=df1[["Annual Income (k\$)", "Spending Score (1-100)"]]

#The input data

X.head()

	Annual Income (k\$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40

#Scatterplot of the input data

```

plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)', data = X ,s = 60)
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.show()

```

The data does seem to hold some patterns.

```

#Importing KMeans from sklearn
from sklearn.cluster import KMeans

```

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

```

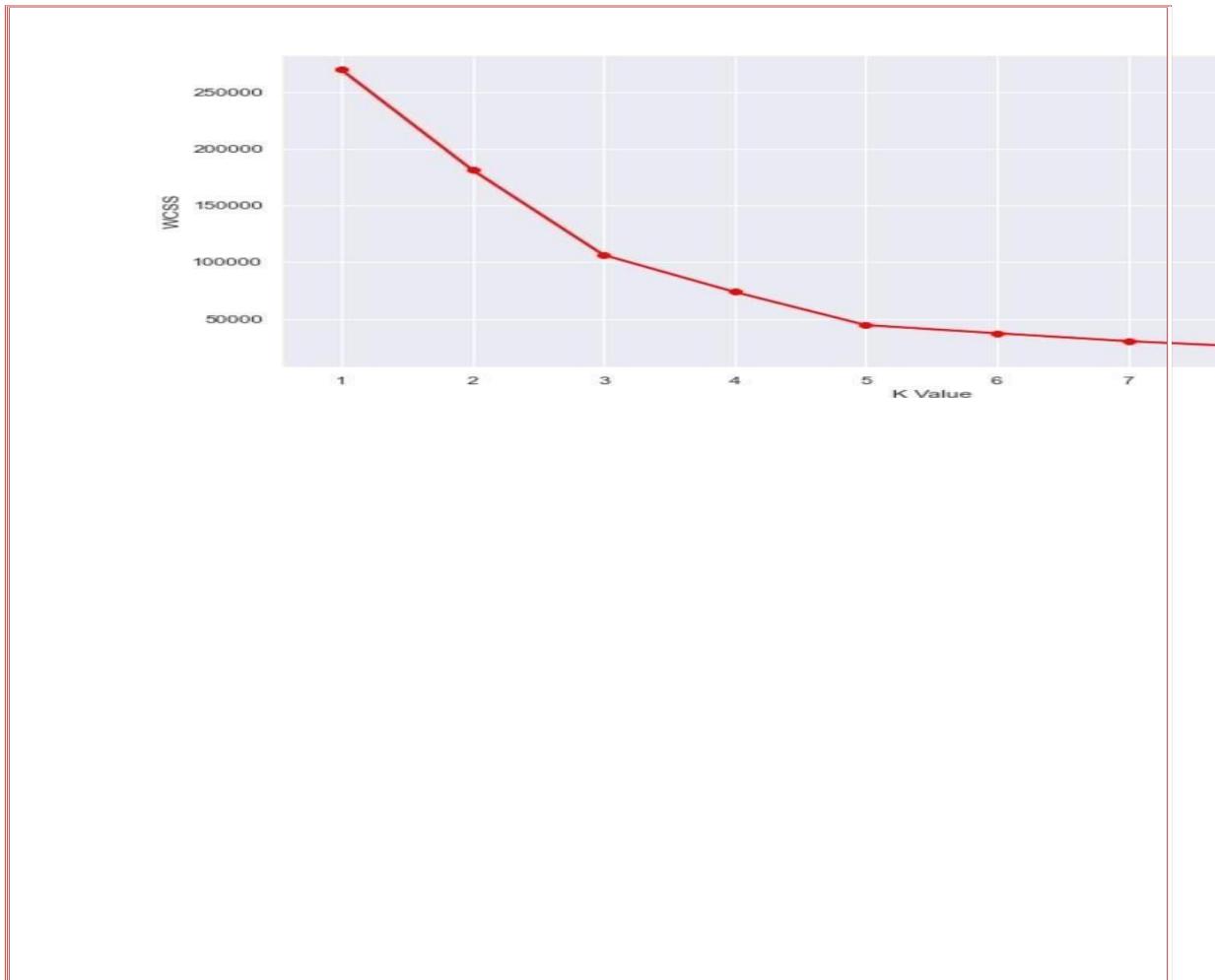
wcss=[]
for i in range(1,11):
    km=KMeans(n_clusters=i)
    km.fit(X)
    wcss.append(km.inertia_)

#The elbow curve
plt.figure(figsize=(12,6))
plt.plot(range(1,11),wcss)
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")
plt.xlabel("K Value")
plt.xticks(np.arange(1,11,1))

```

plt.show()

The plot:



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

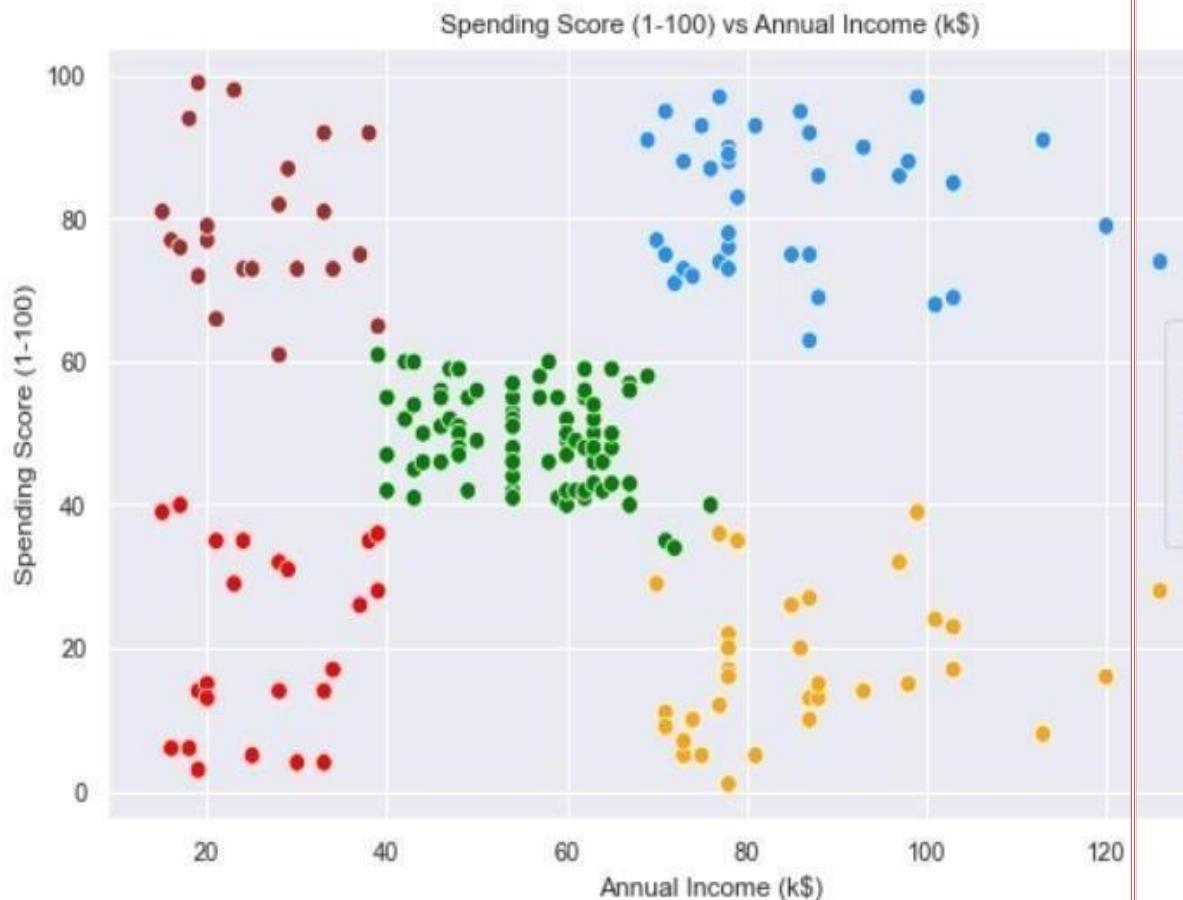
```
#Taking 5 clusters
km1=KMeans(n_clusters=5)
#Fitting the input data
km1.fit(X)
#predicting the labels of the input data
y=km1.predict(X)
#adding the labels to a column named label
df1["label"] = y
#The new dataframe with the clustering done
df1.head()
```

The labels added to the data.

Bharati Vidyapeeth's College Of Engineering Lavale Pune.

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

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



We can clearly see that 5 different clusters have been formed from the data.¹ The red cluster is the customers with the least income and least spending score, similarly, the blue cluster is the customers with the most income and most spending score.

k-Means Clustering on the basis of 3D data

Now, we shall be working on 3 types of data. Apart from the spending score and

annual income of customers, we shall also take in the age of the customers.

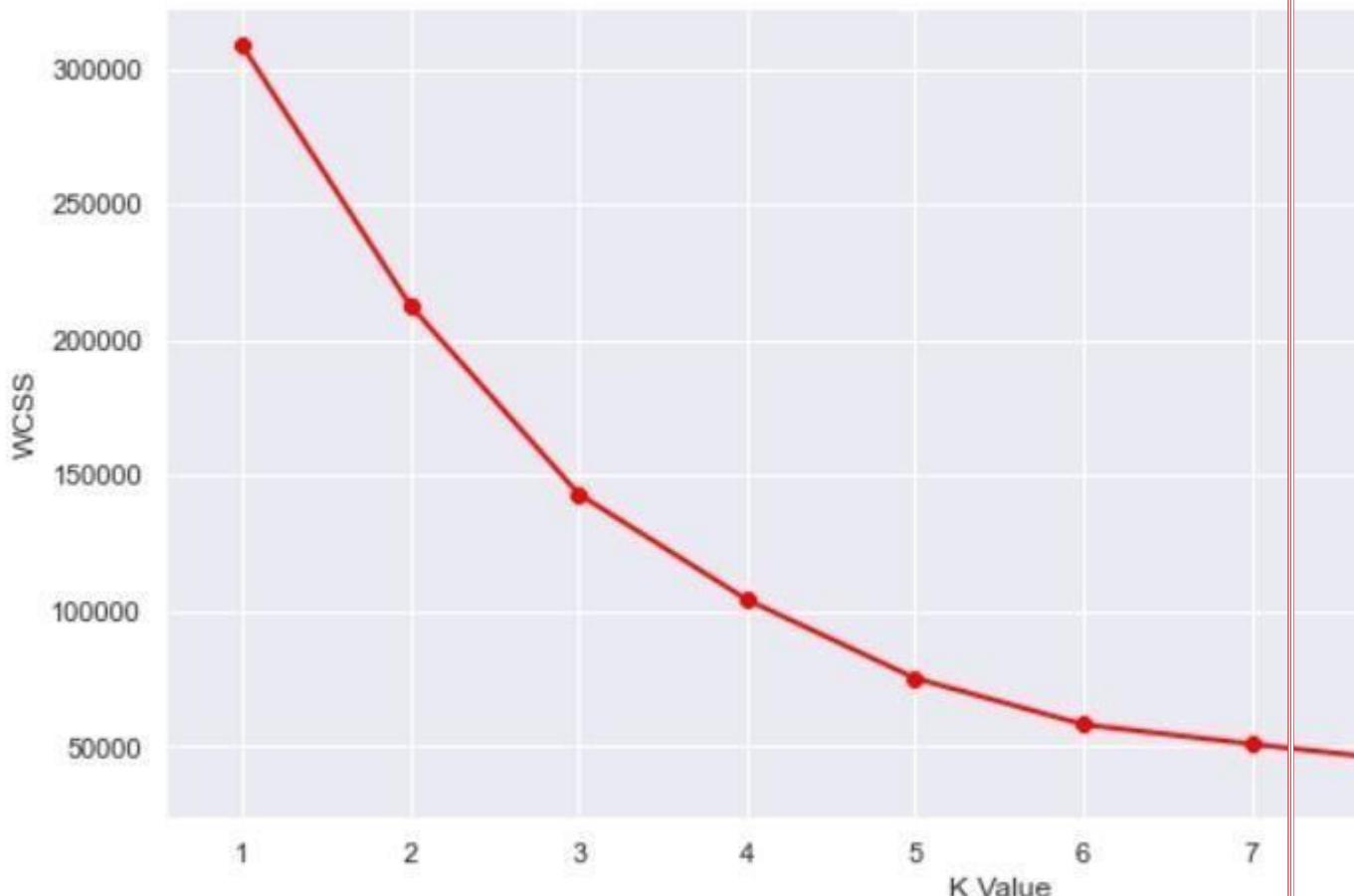
#Taking the features

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

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

```
wcss = []
for k in range(1,11):
    kmeans = KMeans(n_clusters=k, init="k-means++")
    kmeans.fit(X2)
    wcss.append(kmeans.inertia_)
plt.figure(figsize=(12,6))
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker = "8")
plt.xlabel("K Value")
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS")
plt.show()
```

The WCSS curve.



Here can assume that K=5 will be a good value.

#We choose the k for which WSS starts to diminish

```
km2 = KMeans(n_clusters=5)
```

```
y2 = km2.fit_predict(X2)
```

```
df2["label"] = y2
```

#The data with labels

```
df2.head()
```

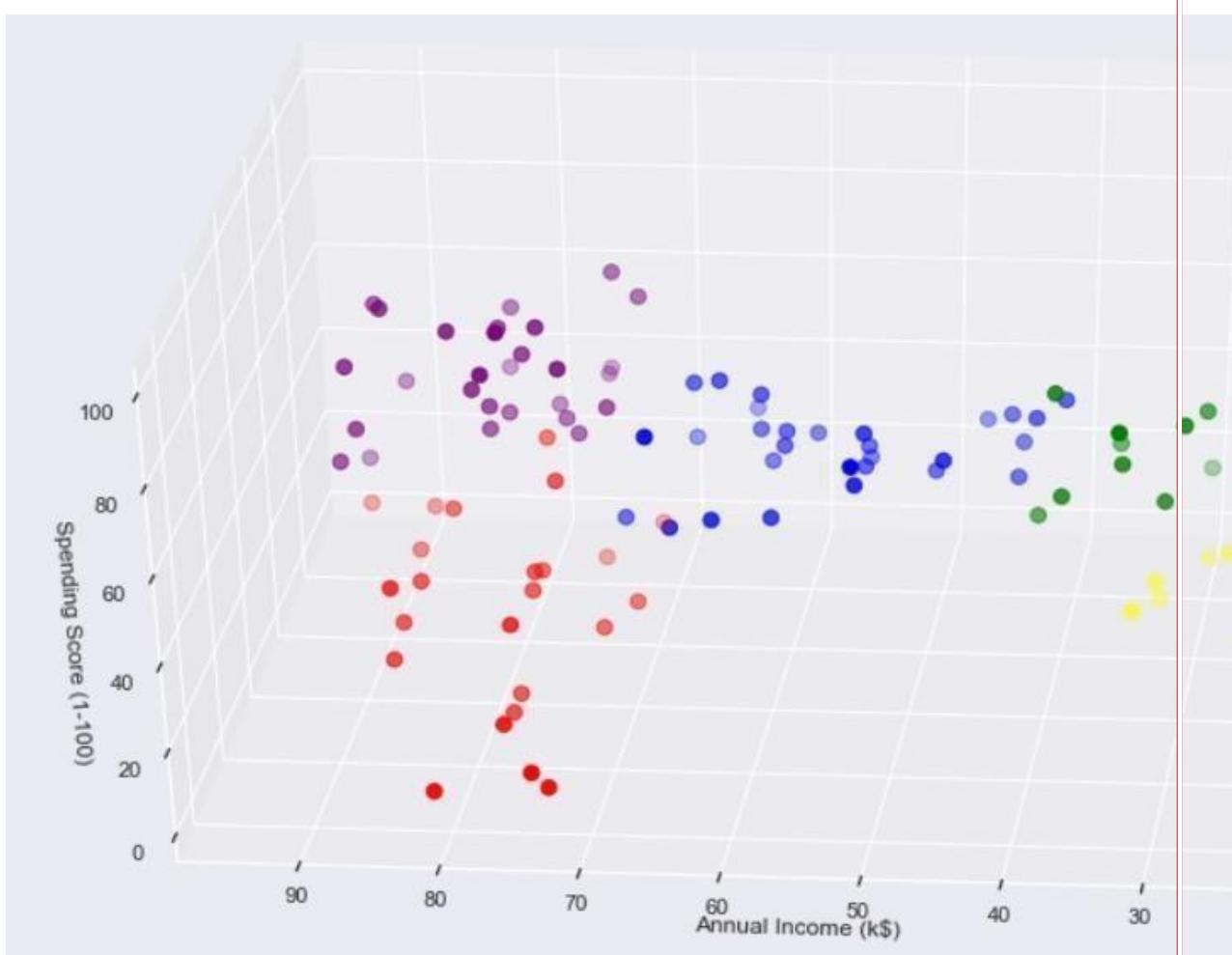
The data:

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

Now we plot it.

```
#3D Plot as we did the clustering on the basis of 3 input features
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df2.Age[df2.label == 0], df2["Annual Income (k$)"][df2.label == 0],
           df2["Spending Score (1-100)"][df2.label == 0], c='purple', s=60)
ax.scatter(df2.Age[df2.label == 1], df2["Annual Income (k$)"][df2.label == 1],
           df2["Spending Score (1-100)"][df2.label == 1], c='red', s=60)
ax.scatter(df2.Age[df2.label == 2], df2["Annual Income (k$)"][df2.label == 2],
           df2["Spending Score (1-100)"][df2.label == 2], c='blue', s=60)
ax.scatter(df2.Age[df2.label == 3], df2["Annual Income (k$)"][df2.label == 3],
           df2["Spending Score (1-100)"][df2.label == 3], c='green', s=60)
ax.scatter(df2.Age[df2.label == 4], df2["Annual Income (k$)"][df2.label == 4],
           df2["Spending Score (1-100)"][df2.label == 4], c='yellow', s=60)
ax.view_init(35, 185)
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
plt.show()
```

The output:



What we get is a 3D plot. Now, if we want to know the customer IDs, we can do that too.

```

cust1=df2[df2["label"]==1]
print('Number of customer in 1st group=', len(cust1))
print('They are -', cust1["CustomerID"].values)
print("-----")
cust2=df2[df2["label"]==2]
print('Number of customer in 2nd group=', len(cust2))
print('They are -', cust2["CustomerID"].values)
print("-----")
cust3=df2[df2["label"]==0]
print('Number of customer in 3rd group=', len(cust3))
print('They are -', cust3["CustomerID"].values)
print("-----")
cust4=df2[df2["label"]==3]
print('Number of customer in 4th group=', len(cust4))
print('They are -', cust4["CustomerID"].values)
print("-----")
cust5=df2[df2["label"]==4]
print('Number of customer in 5th group=', len(cust5))
print('They are -', cust5["CustomerID"].values)

```

```
print("-----")
```

The output we get:

Number of customer in 1st group= 24

They are - [129 131 135 137 139 141 145 147 149 151 153 155 157 159 161 163 165 167
169 171 173 175 177 179]

Number of the customer in 2nd group= 29

They are - [47 51 55 56 57 60 67 72 77 78 80 82 84 86 90 93 94 97
99 102 105 108 113 118 119 120 122 123 127]

Number of the customer in 3rd group= 28

They are - [124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158
160 162 164 166 168 170 172 174 176 178]

Number of the customer in 4th group= 22

They are - [2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 46]

Number of customer in 5th group= 12

They are - [3 7 9 11 13 15 23 25 31 33 35 37]

So, we used K-Means clustering to understand customer data. K-Means is a good clustering algorithm. Almost all the clusters have similar density. It is also fast and efficient in terms of computational cost.

Assignment No : 12

MINI PROJECT 2

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

Importing the Libraries

```
# linear algebra
import numpy as np

# data processing
import pandas as pd

# data visualization
import seaborn as sns
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style

# Algorithms
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.naive_bayes import GaussianNB
```

Getting the Data

```
test_df = pd.read_csv("test.csv")
train_df = pd.read_csv("train.csv")
```

Data Exploration/Analysis

```
train_df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId      891 non-null int64
Survived          891 non-null int64
Pclass             891 non-null int64
Name               891 non-null object
Sex                891 non-null object
Age                714 non-null float64
SibSp              891 non-null int64
Parch              891 non-null int64
Ticket             891 non-null object
Fare               891 non-null float64
Cabin              204 non-null object
Embarked           889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB

```

The training-set has 891 examples and 11 features + the target variable (survived). 2 of the features are floats, 5 are integers and 5 are objects. Below I have listed the features with a short description:

```

survival: Survival
PassengerId: Unique Id of a passenger.
pclass: Ticket class
sex: Sex
Age: Age in years
sibsp: # of siblings / spouses aboard the Titanic
parch: # of parents / children aboard the Titanic
ticket: Ticket number
fare: Passenger fare
cabin: Cabin number
embarked: Port of Embarkationtrain_df.describe()

```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Above we can see that **38% out of the training-set survived the Titanic**. We can also see that the passenger ages range from 0.4 to 80. On top of that we can already detect some features, that contain missing values, like the ‘Age’ feature.

```
train_df.head(8)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... ...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S

From the table above, we can note a few things. First of all, that we **need to convert a lot of features into numeric** ones later on, so that the machine learning algorithms can process them. Furthermore, we can see that the **features have widely different ranges**, that we will need to convert into roughly the same scale. We can also spot some more features, that contain missing values (NaN = not a number), that we need to deal with.

Let's take a more detailed look at what data is actually missing:

```
total = train_df.isnull().sum().sort_values(ascending=False)
percent_1 = train_df.isnull().sum()/train_df.isnull().count()*100
percent_2 = (round(percent_1, 1)).sort_values(ascending=False)
missing_data = pd.concat([total, percent_2], axis=1, keys=['Total',
    '%'])
missing_data.head(5)
```

	Total	%
Cabin	687	77.1
Age	177	19.9
Embarked	2	0.2
Fare	0	0.0
Ticket	0	0.0

The Embarked feature has only 2 missing values, which can easily be filled. It will be much more tricky, to deal with the 'Age' feature, which has 177 missing values. The 'Cabin' feature needs further investigation, but it looks like that we might want to drop it from the dataset, since 77 % of it are missing.

```
train_df.columns.values
array(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'], dtype=object)
```

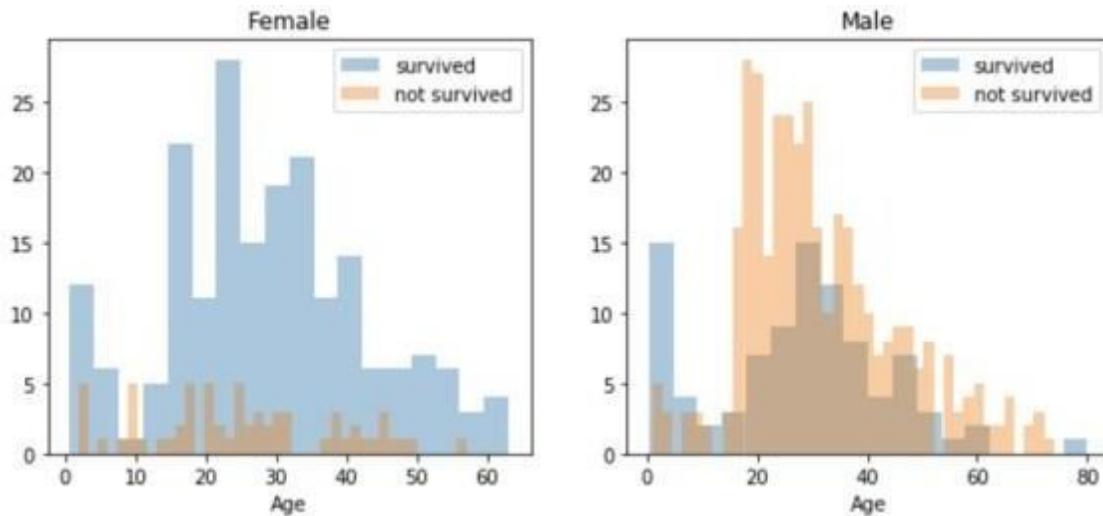
Above you can see the 11 features + the target variable (survived). **What features could contribute to a high survival rate ?**

To me it would make sense if everything except 'PassengerId', 'Ticket' and 'Name' would be correlated with a high survival rate.

1. Age and Sex:

```
survived = 'survived'
not_survived = 'not survived'
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 4))
women = train_df[train_df['Sex']=='female']
men = train_df[train_df['Sex']=='male']
ax = sns.distplot(women[women['Survived']==1].Age.dropna(), bins=18,
label = survived, ax = axes[0], kde =False)
ax = sns.distplot(women[women['Survived']==0].Age.dropna(), bins=40,
label = not_survived, ax = axes[0], kde =False)
ax.legend()
ax.set_title('Female')
ax = sns.distplot(men[men['Survived']==1].Age.dropna(), bins=18, label = survived, ax = axes[1], kde = False)
ax.set_title('Male')
```

```
ax = sns.distplot(men[men['Survived']==0].Age.dropna(), bins=40, label='not survived', ax=axes[1], kde=False)
ax.legend()
ax.set_title('Male')
```



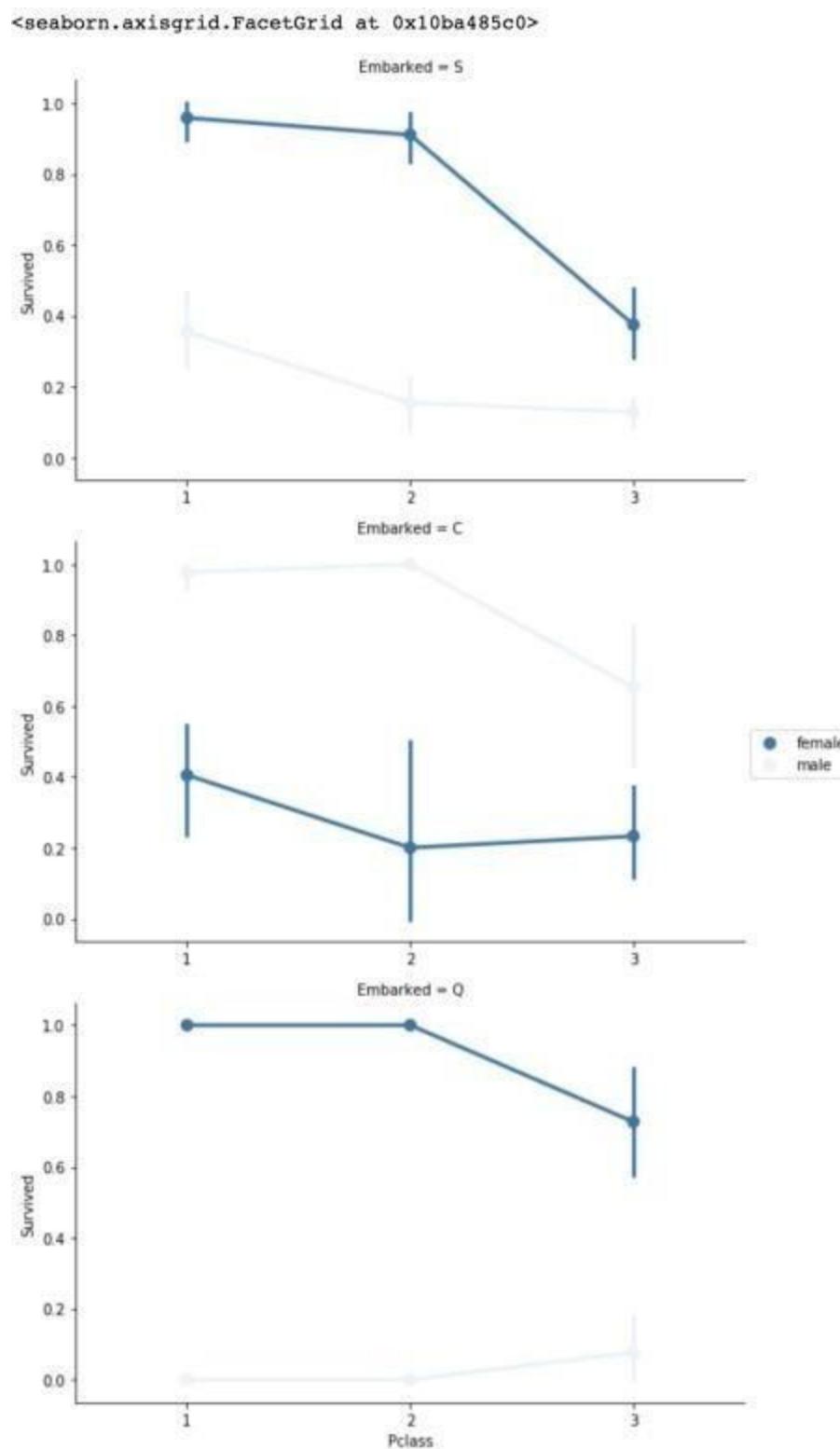
You can see that men have a high probability of survival when they are between 18 and 30 years old, which is also a little bit true for women but not fully. For women the survival chances are higher between 14 and 40.

For men the probability of survival is very low between the age of 5 and 18, but that isn't true for women. Another thing to note is that infants also have a little bit higher probability of survival.

Since there seem to be **certain ages, which have increased odds of survival** and because I want every feature to be roughly on the same scale, I will create age groups later on.

3. Embarked, Pclass and Sex:

```
FacetGrid = sns.FacetGrid(train_df, row='Embarked', size=4.5, aspect=1.6)
FacetGrid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex',
palette=None, order=None, hue_order=None )
FacetGrid.add_legend()
```



Embarked seems to be correlated with survival, depending on the gender.

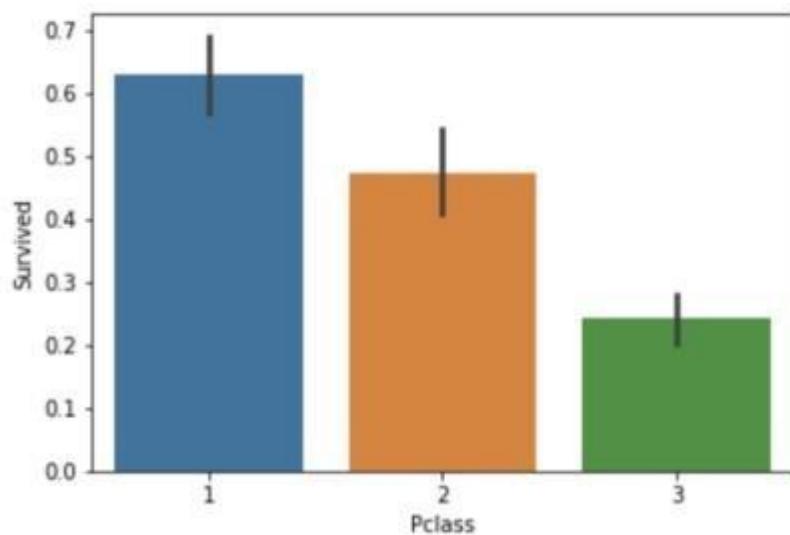
Women on port Q and on port S have a higher chance of survival. The inverse is true, if they are at port C. Men have a high survival probability if they are on port C, but a low probability if they are on port Q or S.

Pclass also seems to be correlated with survival. We will generate another plot of it below.

4. Pclass:

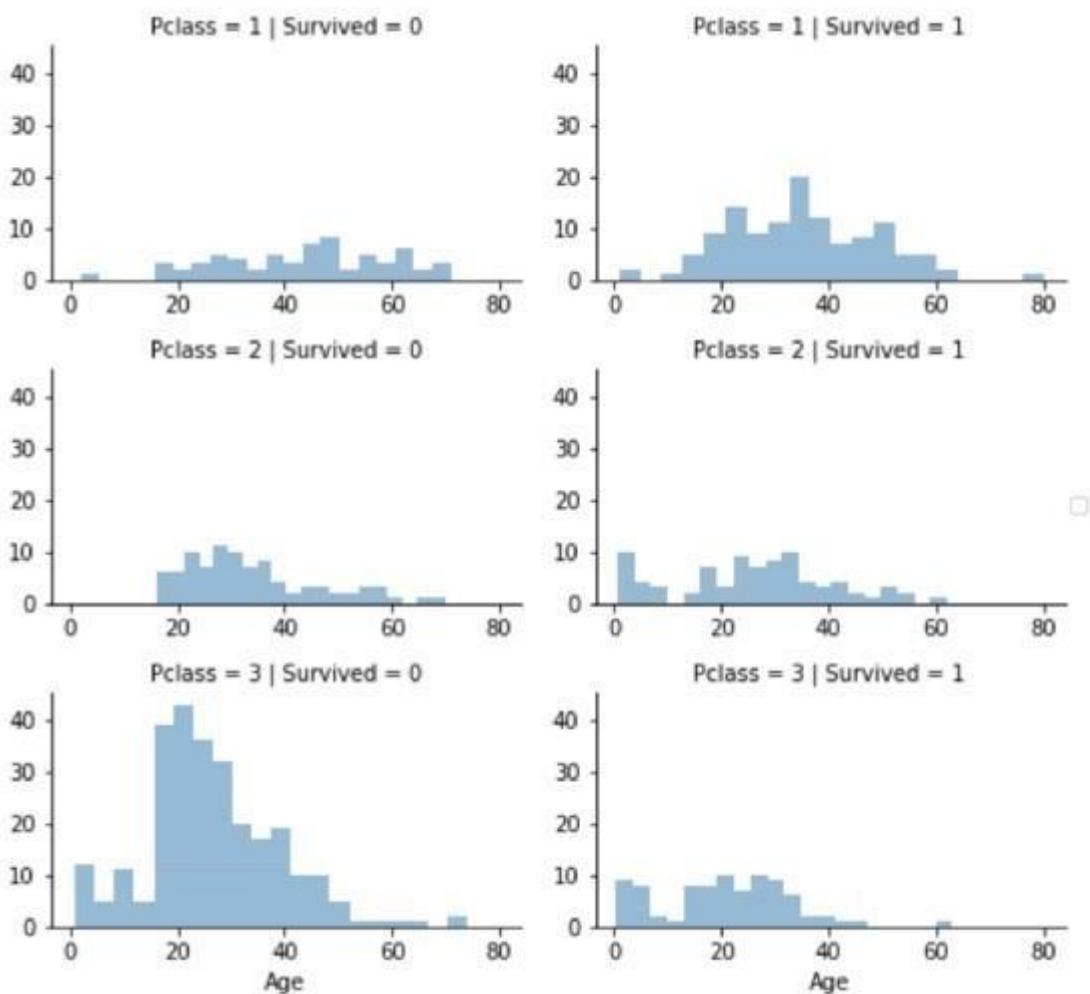
```
sns.barplot(x='Pclass', y='Survived', data=train_df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x10d1dc7b8>
```



Here we see clearly, that Pclass is contributing to a persons chance of survival, especially if this person is in class 1. We will create another pclass plot below.

```
grid = sns.FacetGrid(train_df, col='Survived', row='Pclass', size=2.2, aspect=1.6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend()
```



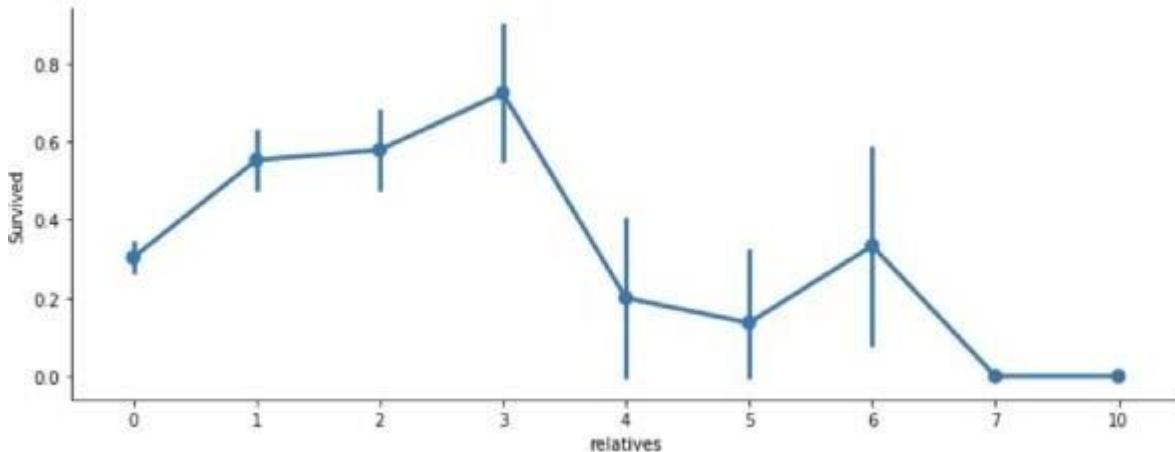
The plot above confirms our assumption about pclass 1, but we can also spot a high probability that a person in pclass 3 will not survive.

5. SibSp and Parch:

SibSp and Parch would make more sense as a combined feature, that shows the total number of relatives, a person has on the Titanic. I will create it below and also a feature that says if someone is not alone.

```
data = [train_df, test_df]
for dataset in data:
    dataset['relatives'] = dataset['SibSp'] + dataset['Parch']
    dataset.loc[dataset['relatives'] > 0, 'not_alone'] = 0
    dataset.loc[dataset['relatives'] == 0, 'not_alone'] = 1
    dataset['not_alone'] =
dataset['not alone'].astype(int)train df['not alone'].value counts()
1      537
0      354
Name: not alone, dtype: int64
```

```
axes = sns.factorplot('relatives','Survived',
                      data=train_df, aspect = 2.5, )
```



Here we can see that you had a high probability of survival with 1 to 3 relatives, but a lower one if you had less than 1 or more than 3 (except for some cases with 6 relatives).

Data Preprocessing

First, I will drop ‘PassengerId’ from the train set, because it does not contribute to a persons survival probability. I will not drop it from the test set, since it is required there for the submission.

```
train df = train df.drop(['PassengerId'], axis=1)
```

Missing Data:

Cabin:

As a reminder, we have to deal with Cabin (687), Embarked (2) and Age (177). First I thought, we have to delete the ‘Cabin’ variable but then I

found something interesting. A cabin number looks like 'C123' and the **letter refers to the deck**. Therefore we're going to extract these and create a new feature, that contains a persons deck. Afterwards we will convert the feature into a numeric variable. The missing values will be converted to zero. In the picture below you can see the actual decks of the titanic, ranging from A to G.

```
import re
deck = {"A": 1, "B": 2, "C": 3, "D": 4, "E": 5, "F": 6, "G": 7, "U": 8}
data = [train_df, test_df]

for dataset in data:
    dataset['Cabin'] = dataset['Cabin'].fillna("U0")
    dataset['Deck'] = dataset['Cabin'].map(lambda x: re.compile("([a-zA-Z]+)").search(x).group())
    dataset['Deck'] = dataset['Deck'].map(deck)
    dataset['Deck'] = dataset['Deck'].fillna(0)
    dataset['Deck'] = dataset['Deck'].astype(int) # we can now drop the
cabin feature
train_df = train_df.drop(['Cabin'], axis=1)
test_df = test_df.drop(['Cabin'], axis=1)
```

Age:

Now we can tackle the issue with the age features missing values. I will create an array that contains random numbers, which are computed based on the mean age value in regards to the standard deviation and is_null.

```
data = [train_df, test_df]

for dataset in data:
    mean = train_df["Age"].mean()
    std = test_df["Age"].std()
    is_null = dataset["Age"].isnull().sum()
    # compute random numbers between the mean, std and is_null
    rand_age = np.random.randint(mean - std, mean + std, size =
is_null)
    # fill NaN values in Age column with random values generated
    age_slice = dataset["Age"].copy()
    age_slice[np.isnan(age_slice)] = rand_age
    dataset["Age"] = age_slice
    dataset["Age"] =
train_df["Age"].astype(int)train_df["Age"].isnull().sum()
```

0

Bharati Vidyapeeth's College Of Engineering Lavale Pune.

3

Embarked:

Since the Embarked feature has only 2 missing values, we will just fill these with the most common one.

```
train_df['Embarked'].describe()
```

```

count      889
unique      3
top         S
freq      644
Name: Embarked, dtype: object

common_value = 'S'
data = [train_df, test_df]

for dataset in data:
    dataset['Embarked'] = dataset['Embarked'].fillna(common_value)

```

Converting Features:

```

train_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 columns):
Survived     891 non-null int64
Pclass       891 non-null int64
Name         891 non-null object
Sex          891 non-null object
Age          891 non-null int64
SibSp        891 non-null int64
Parch        891 non-null int64
Ticket       891 non-null object
Fare          891 non-null float64
Embarked     891 non-null object
relatives    891 non-null int64
not_alone    891 non-null int64
Deck          891 non-null int64
dtypes: float64(1), int64(8), object(4)
memory usage: 90.6+ KB

```

Above you can see that ‘Fare’ is a float and we have to deal with 4 categorical features: Name, Sex, Ticket and Embarked. Lets investigate and transfrom one after another.

Fare:

Converting “Fare” from float to int64, using the “astype()” function pandas provides:

```

data = [train_df, test_df]

for dataset in data:
    dataset['Fare'] = dataset['Fare'].fillna(0)
    dataset['Fare'] = dataset['Fare'].astype(int)

```

Name:

We will use the Name feature to extract the Titles from the Name, so that we can build a new feature out of that.

```

data = [train_df, test_df]
titles = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}

for dataset in data:
    # extract titles

```

```

dataset['Title'] = dataset.Name.str.extract(' ([A-Za-z]+)\.', expand=False)
# replace titles with a more common title or as Rare
dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess', 'Capt', 'Col', 'Don', 'Dr', \
                                             'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')
dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')
# convert titles into numbers
dataset['Title'] = dataset['Title'].map(titles)
# filling NaN with 0, to get safe
dataset['Title'] = dataset['Title'].fillna(0)
train_df = train_df.drop(['Name'], axis=1)
test_df = test_df.drop(['Name'], axis=1)

```

Sex:

Convert ‘Sex’ feature into numeric.

```

genders = {"male": 0, "female": 1}
data = [train_df, test_df]

for dataset in data:
    dataset['Sex'] = dataset['Sex'].map(genders)

```

Ticket:

```

train_df['Ticket'].describe()
count      891
unique     681
top       1601
freq        7
Name: Ticket, dtype: object

```

Since the Ticket attribute has 681 unique tickets, it will be a bit tricky to convert them into useful categories. So we will drop it from the dataset.

```

train_df = train_df.drop(['Ticket'], axis=1)
test_df = test_df.drop(['Ticket'], axis=1)

```

Embarked:

Convert ‘Embarked’ feature into numeric.

```

ports = {"S": 0, "C": 1, "Q": 2}
data = [train_df, test_df]

for dataset in data:
    dataset['Embarked'] = dataset['Embarked'].map(ports)

```

Creating Categories:

We will now create categories within the following features:

Age	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	relatives	not_alone	Deck	Title
Now we need to convert the 'age' feature. First we will convert it from float into integer. Then we will create the new 'AgeGroup' variable, by categorizing every age into a group. Note that it is important to place attention on how you form these groups, since you don't want for example that 80% of your data falls into group 1.	1	1	1	1	1	1	1	1	1	1	1	1
5 0	3	0	4	0	0	8	2	0	0	1	8	1
6 0	1	0	6	0	0	51	0	0	0	1	5	1
7 0	3	0	0	3	1	21	0	4	0	8	4	
8 1	3	1	3	0	2	11	0	2	0	8	3	
9 1	2	1	1	1	0	30	1	1	0	8	3	

```

= 3
    dataset.loc[ (dataset['Age'] > 27) & (dataset['Age'] <= 33), 'Age'] = 3
= 4
    dataset.loc[ (dataset['Age'] > 33) & (dataset['Age'] <= 40), 'Age'] = 4
= 5
    dataset.loc[ (dataset['Age'] > 40) & (dataset['Age'] <= 66), 'Age'] = 5
= 6
    dataset.loc[ dataset['Age'] > 66, 'Age'] = 6

# let's see how it's distributed train_df['Age'].value_counts()
4    165
6    158
5    147
3    129
2    124
1    100
0     68
Name: Age, dtype: int64

```

Fare:

For the 'Fare' feature, we need to do the same as with the 'Age' feature. But it isn't that easy, because if we cut the range of the fare values into a few equally big categories, 80% of the values would fall into the first category. Fortunately, we can use sklearn "qcut()" function, that we can use to see, how we can form the categories.

```
train_df.head(10)
```

```

data = [train_df, test_df]

for dataset in data:
    dataset.loc[ dataset['Fare'] <= 7.91, 'Fare' ] = 0
    dataset.loc[(dataset['Fare'] > 7.91) & (dataset['Fare'] <=
14.454), 'Fare' ] = 1
    dataset.loc[(dataset['Fare'] > 14.454) & (dataset['Fare'] <= 31),
'Fare' ] = 2
    dataset.loc[(dataset['Fare'] > 31) & (dataset['Fare'] <= 99),
'Fare' ] = 3
    dataset.loc[(dataset['Fare'] > 99) & (dataset['Fare'] <= 250),
'Fare' ] = 4
    dataset.loc[ dataset['Fare'] > 250, 'Fare' ] = 5
    dataset['Fare'] = dataset['Fare'].astype(int)

```

Creating new Features

I will add two new features to the dataset, that I compute out of other features.

1. Age times Class

```

data = [train_df, test_df]
for dataset in data:
    dataset['Age_Class']= dataset['Age']* dataset['Pclass']

```

2. Fare per Person

```

for dataset in data:
    dataset['Fare_Per_Person'] =
dataset['Fare']/(dataset['relatives']+1)
    dataset['Fare_Per_Person'] =
dataset['Fare_Per_Person'].astype(int)# Let's take a last look at the
training set, before we start training the models.
train_df.head(10)

```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	relatives	not_alone	Deck	Title	Age_Class	Fare_Per_Pers
0	0	3	0	2	1	0	0	0	1	0	8	1	6	0
1	1	1	1	5	1	0	3	1	1	0	3	3	5	1
2	1	3	1	3	0	0	0	0	0	1	8	2	9	0
3	1	1	1	5	1	0	3	0	1	0	3	3	5	1
4	0	3	0	5	0	0	1	0	0	1	8	1	15	1
5	0	3	0	4	0	0	1	2	0	1	8	1	12	1
6	0	1	0	6	0	0	3	0	0	1	5	1	6	3
7	0	3	0	0	3	1	2	0	4	0	8	4	0	0
8	1	3	1	3	0	2	1	0	2	0	8	3	9	0
9	1	2	1	1	1	0	2	1	1	0	8	3	2	1
10	1	3	1	0	1	1	2	0	2	0	7	2	0	0

Building Machine Learning Models

Now we will train several Machine Learning models and compare their results. Note that because the dataset does not provide labels for their testing-set, we need to use the predictions on the training set to compare the algorithms with each other. Later on, we will use cross validation.

```
X_train = train_df.drop("Survived", axis=1)
Y_train = train_df["Survived"]
X_test = test_df.drop("PassengerId", axis=1).copy()
```

Stochastic Gradient Descent (SGD):

```
sgd = linear_model.SGDClassifier(max_iter=5, tol=None)
sgd.fit(X_train, Y_train)
Y_pred = sgd.predict(X_test)

sgd.score(X_train, Y_train)

acc_sgd = round(sgd.score(X_train, Y_train) * 100, 2)
```

Random Forest:

```
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)

Y_prediction = random_forest.predict(X_test)

random_forest.score(X_train, Y_train)
acc_random_forest = round(random_forest.score(X_train, Y_train) * 100, 2)
```

Logistic Regression:

```
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)

Y_pred = logreg.predict(X_test)

acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
```

K Nearest Neighbor:

```
# KNN
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
```

Gaussian Naive Bayes:

```
gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
Y_pred = gaussian.predict(X_test)
acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
```

Perceptron:

```
perceptron = Perceptron(max_iter=5)
perceptron.fit(X_train, Y_train)

Y_pred = perceptron.predict(X_test)

acc_perceptron = round(perceptron.score(X_train, Y_train) * 100, 2)
```

Linear Support Vector Machine:

```
linear_svc = LinearSVC()
linear_svc.fit(X_train, Y_train)

Y_pred = linear_svc.predict(X_test)

acc_linear_svc = round(linear_svc.score(X_train, Y_train) * 100, 2)
```

Decision Tree

```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100, 2)
```

Which is the best Model ?

```
results = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'Perceptron',
              'Stochastic Gradient Decent',
              'Decision Tree'],
    'Score': [acc_linear_svc, acc_knn, acc_log,
              acc_random_forest, acc_gaussian, acc_perceptron,
              acc_sgd, acc_decision_tree]})

result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df.head(9)
```

Bharati Vidyapeeth's College Of Engineering Lavale Pune.

	Model
Score	
92.82	Random Forest
92.82	Decision Tree
87.32	KNN
81.14	Logistic Regression
80.81	Support Vector Machines
80.70	Perceptron
77.10	Naive Bayes
76.99	Stochastic Gradient Decent

As we can see, the Random Forest classifier goes on the first place. But first, let us check, how random-forest performs, when we use cross validation.

K-Fold Cross Validation:

K-Fold Cross Validation randomly splits the training data into **K subsets called folds**. Let's image we would split our data into 4 folds ($K = 4$). Our random forest model would be trained and evaluated 4 times, using a different fold for evaluation everytime, while it would be trained on the remaining 3 folds.

The image below shows the process, using 4 folds ($K = 4$). Every row represents one training + evaluation process. In the first row, the model get's trained on the first, second and third subset and evaluated on the fourth. In the second row, the model get's trained on the second, third and fourth subset and evaluated on the first. K-Fold Cross Validation repeats this process till every fold acted once as an evaluation fold.

Training	Training	Training	Evaluation
1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

The result of our K-Fold Cross Validation example would be an array that contains 4 different scores. We then need to compute the mean and the standard deviation for these scores.

The code below perform K-Fold Cross Validation on our random forest model, using 10 folds ($K = 10$). Therefore it outputs an array with 10 different scores.

```
from sklearn.model_selection import cross_val_score
rf = RandomForestClassifier(n_estimators=100)
scores = cross_val_score(rf, X_train, Y_train, cv=10, scoring =
"accuracy")
print("Scores:", scores)
print("Mean:", scores.mean())
print("Standard Deviation:", scores.std())
Scores: [ 0.76666667  0.82222222  0.7752809   0.82022472  0.85393258  0.86516854
 0.83146067  0.76404494  0.85393258  0.85227273]
Mean: 0.820520655998
Standard Deviation: 0.0367333665466
```

This looks much more realistic than before. Our model has a average accuracy of 82% with a standard deviation of 4 %. The standard deviation shows us, how precise the estimates are .

This means in our case that the accuracy of our model can differ $+ - 4\%$.

I think the accuracy is still really good and since random forest is an easy to use model, we will try to increase it's performance even further in the following section.

Random Forest

What is Random Forest ?

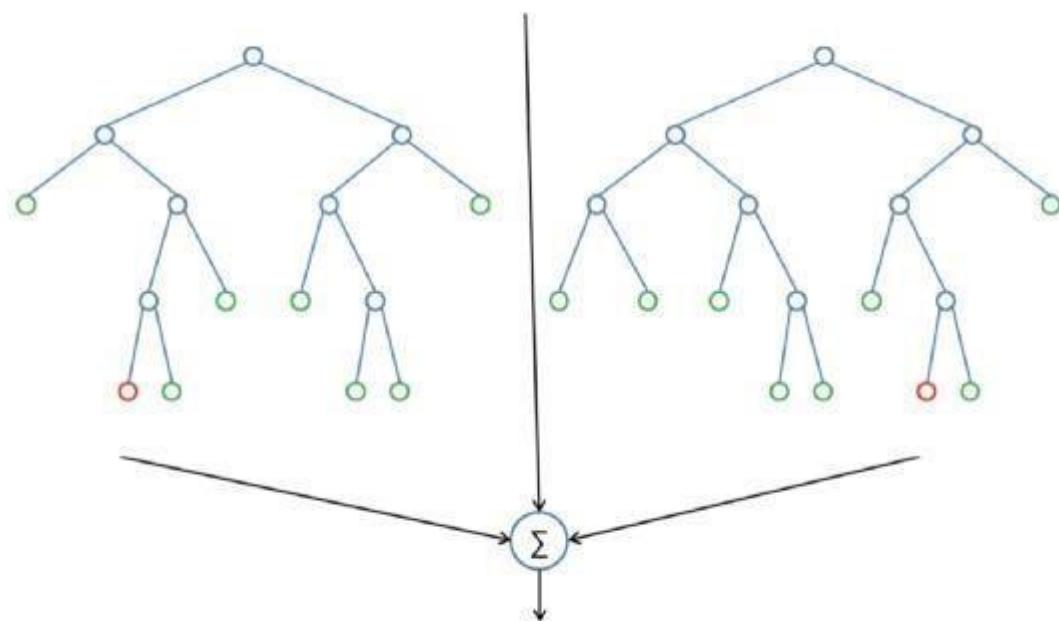
Random Forest is a supervised learning algorithm. Like you can already see from it's name, it creates a forest and makes it somehow random. The „forest“ it builds, is an ensemble of Decision Trees, most of the time trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. With a few exceptions a random-forest classifier has all the hyperparameters of a decision-tree classifier and also all the hyperparameters of a bagging classifier, to control the ensemble itself.

The random-forest algorithm brings extra randomness into the model, when it is growing the trees. Instead of searching for the best feature while splitting a node, it searches for the best feature among a random subset of features. This process creates a wide diversity, which generally results in a better model. Therefore when you are growing a tree in random forest, only a random subset of the features is considered for splitting a node. You can even make trees more random, by using random thresholds on top of it, for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

Below you can see how a random forest would look like with two trees:



4

Feature Importance

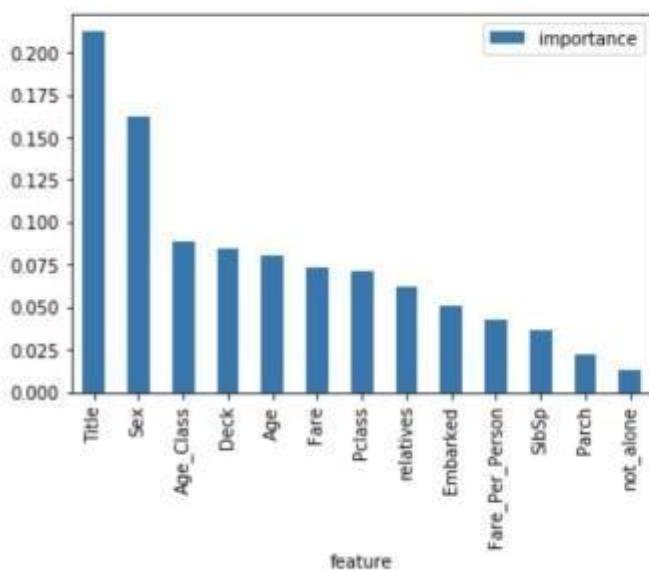
Another great quality of random forest is that they make it very easy to measure the relative importance of each feature. Sklearn measure a features importance by looking at how much the treee nodes, that use that

feature, reduce impurity on average (across all trees in the forest). It computes this score automaticall for each feature after training and scales the results so that the sum of all importances is equal to 1. We will acces this below:

```
importances =
pd.DataFrame({'feature':X_train.columns,'importance':np.round(random_f
orest.feature_importances_,3)})
importances =
importances.sort_values('importance',ascending=False).set_index('featu
re')importances.head(15)
```

	importance
feature	
Title	0.212
Sex	0.162
Age_Class	0.089
Deck	0.084
Age	0.080
Fare	0.073
Pclass	0.071
relatives	0.062
Embarked	0.051
Fare_Per_Person	0.043
SibSp	0.036
Parch	0.022
not_alone	0.013

```
importances.plot.bar()
<matplotlib.axes._subplots.AxesSubplot at 0x1a157c1e10>
```



Conclusion:

not_alone and Parch doesn't play a significant role in our random forest classifiers prediction process. Because of that I will drop them from the dataset and train the classifier again. We could also remove more or less

features, but this would need a more detailed investigation of the features effect on our model. But I think it's just fine to remove only Alone and Parch.

```
train_df = train_df.drop("not_alone", axis=1)
test_df = test_df.drop("not_alone", axis=1)

train_df = train_df.drop("Parch", axis=1)
test_df = test_df.drop("Parch", axis=1)
```

Training random forest again:

```
# Random Forest

random_forest = RandomForestClassifier(n_estimators=100, oob_score = True)
random_forest.fit(X_train, Y_train)
Y_prediction = random_forest.predict(X_test)

random_forest.score(X_train, Y_train)

acc_random_forest = round(random_forest.score(X_train, Y_train) * 100, 2)
print(round(acc_random_forest, 2), "%")
```

92.82%

Our random forest model predicts as good as it did before. A general rule is that, **the more features you have, the more likely your model will suffer from overfitting** and vice versa. But I think our data looks fine for now and hasn't too much features.

There is also another way to evaluate a random-forest classifier, which is probably much more accurate than the score we used before. What I am talking about is the **out-of-bag samples** to estimate the generalization accuracy. I will not go into details here about how it works. Just note that out-of-bag estimate is as accurate as using a test set of the same size as the training set. Therefore, using the out-of-bag error estimate removes the need for a set aside test set.

```
print("oob score:", round(random_forest.oob_score, 4)*100, "%")
```

oob score: 81.82 %

Now we can start tuning the hyperparameters of random forest.

Hyperparameter Tuning

Below you can see the code of the hyperparameter tuning for the parameters criterion, min_samples_leaf, min_samples_split and n_estimators.

I put this code into a markdown cell and not into a code cell, because it takes a long time to run it. Directly underneath it, I put a screenshot of the gridsearch's output.

```
param_grid = { "criterion" : ["gini", "entropy"], "min_samples_leaf" : [1, 5, 10, 25, 50, 70], "min_samples_split" : [2, 4, 10, 12, 16, 18, 25, 35], "n_estimators": [100, 400, 700, 1000, 1500]}from sklearn.model_selection import GridSearchCV, cross_val_scoref = RandomForestClassifier(n_estimators=100, max_features='auto', oob_score=True, random_state=1, n_jobs=-1)clf = GridSearchCV(estimator=rf, param_grid=param_grid, n_jobs=-1)clf.fit(X_train, Y_train)clf.best_params
```

```
{'criterion': 'gini',
'min_samples_leaf': 1,
'min_samples_split': 10,
'n_estimators': 100}
```

Test new Parameters:

```
# Random Forest
random_forest = RandomForestClassifier(criterion = "gini",
                                       min_samples_leaf = 1,
                                       min_samples_split = 10,
                                       n_estimators=100,
                                       max_features='auto',
                                       oob_score=True,
                                       random_state=1,
                                       n_jobs=-1)

random_forest.fit(X_train, Y_train)
Y_prediction = random_forest.predict(X_test)

random_forest.score(X_train, Y_train)

print("oob score:", round(random_forest.oob_score_, 4)*100, "%")
```

oob score: 83.05 %

Now that we have a proper model, we can start evaluating it's performance in a more accurate way. Previously we only used accuracy and the oob score, which is just another form of accuracy. The problem is just, that it's more complicated to evaluate a classification model than a regression model. We will talk about this in the following section.

Further Evaluation

Confusion Matrix:

```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
predictions = cross_val_predict(random_forest, X_train, Y_train, cv=3)
confusion_matrix(Y_train, predictions)

array([[488,  61],
       [ 95, 247]])
```

The first row is about the not-survived-predictions: **493 passengers were correctly classified as not survived** (called true negatives) and **56 where wrongly classified as not survived** (false positives).

The second row is about the survived-predictions: **93 passengers where wrongly classified as survived** (false negatives) and **249 where correctly classified as survived** (true positives).

A confusion matrix gives you a lot of information about how well your model does, but there's a way to get even more, like computing the classifier's precision.

Precision and Recall:

```
from sklearn.metrics import precision_score, recall_score

print("Precision:", precision_score(Y_train, predictions))
print("Recall:", recall_score(Y_train, predictions))
```

Precision: 0.801948051948

Recall: 0.722222222222

Our model predicts 81% of the time, a passengers survival correctly (precision). The recall tells us that it predicted the survival of 73 % of the people who actually survived.

F-Score

You can combine precision and recall into one score, which is called the F-score. The F-score is computed with the harmonic mean of precision and recall. Note that it assigns much more weight to low values. As a result of that, the classifier will only get a high F-score, if both recall and precision are high.

```
from sklearn.metrics import f1_score
f1_score(Y_train, predictions)
```

0.759999999999

There we have it, a 77 % F-score. The score is not that high, because we have a recall of 73%. But unfortunately the F-score is not perfect, because it favors classifiers that have a similar precision and recall. This is a problem, because you sometimes want a high precision and sometimes a high recall. The thing is that an increasing precision, sometimes results in an decreasing recall and vice versa (depending on the threshold). This is called the precision/recall tradeoff. We will discuss this in the following section.

Precision Recall Curve

For each person the Random Forest algorithm has to classify, it computes a probability based on a function and it classifies the person as survived (when the score is bigger than the threshold) or as not survived (when the score is smaller than the threshold). That's why the threshold plays an important part.

We will plot the precision and recall with the threshold using matplotlib:

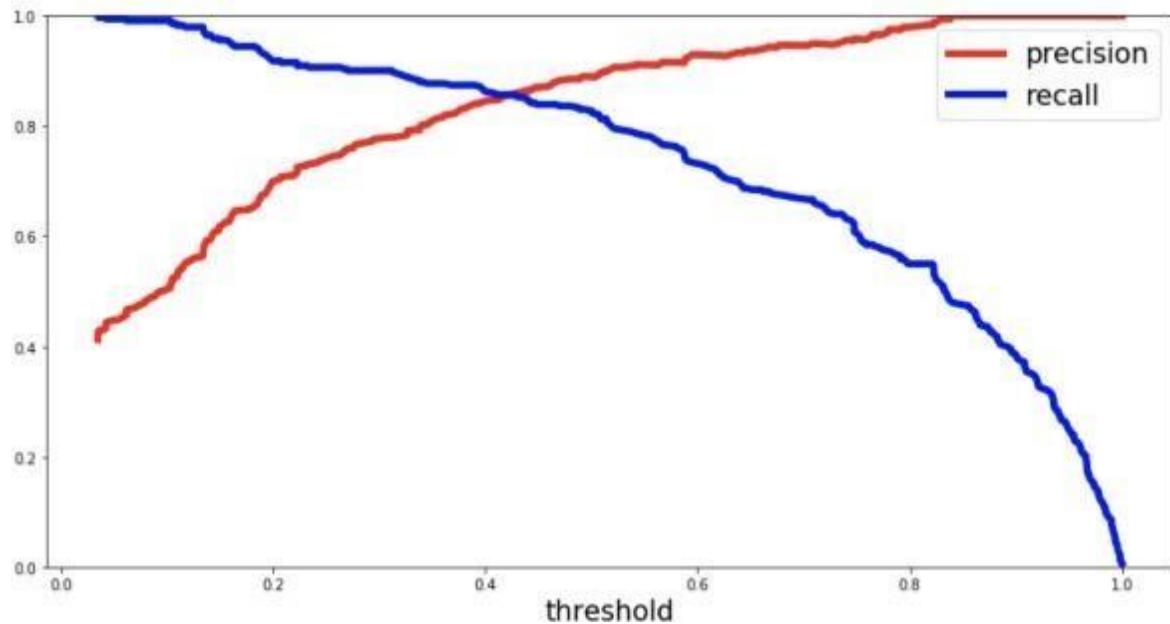
```
from sklearn.metrics import precision_recall_curve

# getting the probabilities of our predictions
y_scores = random_forest.predict_proba(X_train)
y_scores = y_scores[:,1]

precision, recall, threshold = precision_recall_curve(Y_train,
```

```
y_scores)def plot_precision_and_recall(precision, recall, threshold):
    plt.plot(threshold, precision[:-1], "r-", label="precision",
    linewidth=5)
    plt.plot(threshold, recall[:-1], "b", label="recall", linewidth=5)
    plt.xlabel("threshold", fontsize=19)
    plt.legend(loc="upper right", fontsize=19)
    plt.ylim([0, 1])

plt.figure(figsize=(14, 7))
plot_precision_and_recall(precision, recall, threshold)
plt.show()
```



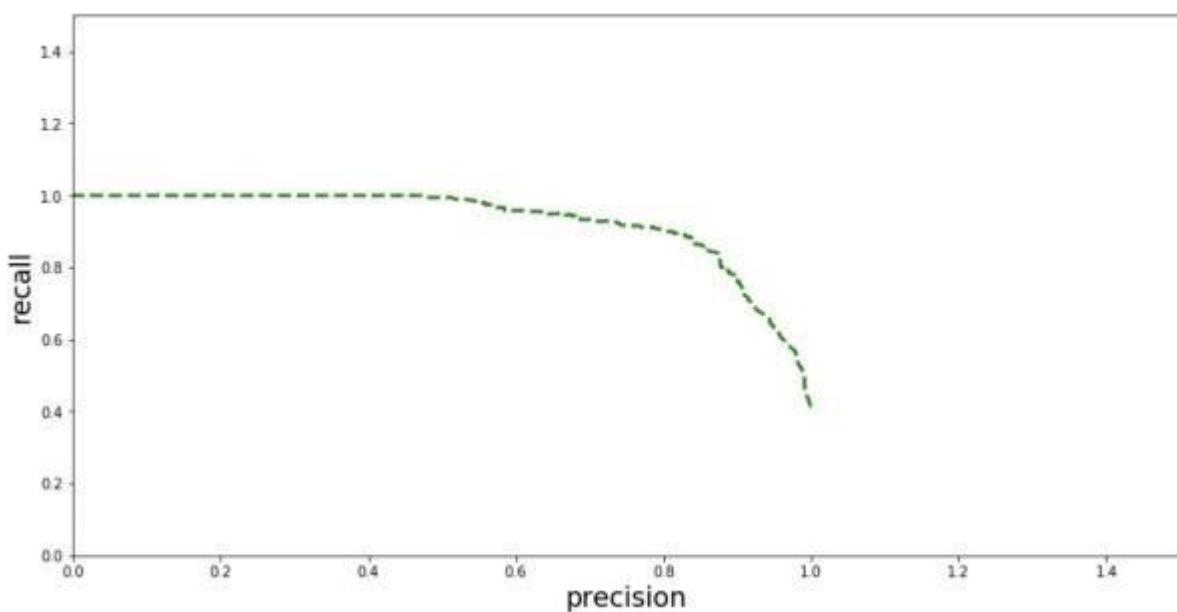
Above you can clearly see that the recall is falling off rapidly at a precision of around 85%. Because of that you may want to select the precision/recall tradeoff before that — maybe at around 75 %.

You are now able to choose a threshold, that gives you the best precision/recall tradeoff for your current machine learning problem. If you want for example a precision of 80%, you can easily look at the plots and see that you would need a threshold of around 0.4. Then you could train a model with exactly that threshold and would get the desired accuracy.

Another way is to plot the precision and recall against each other:

```
def plot_precision_vs_recall(precision, recall):
    plt.plot(recall, precision, "g--", linewidth=2.5)
    plt.ylabel("recall", fontsize=19)
    plt.xlabel("precision", fontsize=19)
    plt.axis([0, 1.5, 0, 1.5])

plt.figure(figsize=(14, 7))
plot_precision_vs_recall(precision, recall)
plt.show()
```

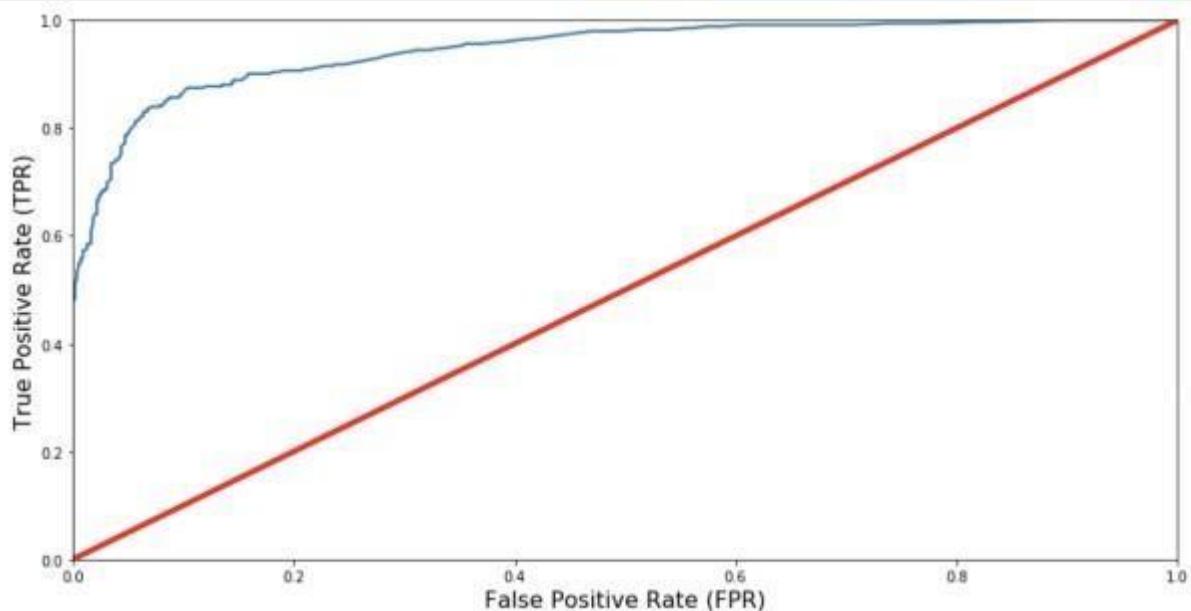


ROC AUC Curve

Another way to evaluate and compare your binary classifier is provided by the ROC AUC Curve. This curve plots the true positive rate (also called recall) against the false positive rate (ratio of incorrectly classified negative instances), instead of plotting the precision versus the recall.

```
from sklearn.metrics import roc_curve
# compute true positive rate and false positive rate
false_positive_rate, true_positive_rate, thresholds =
roc_curve(Y_train, y_scores) # plotting them against each other
def plot_roc_curve(false_positive_rate, true_positive_rate,
label=None):
    plt.plot(false_positive_rate, true_positive_rate, linewidth=2,
label=label)
    plt.plot([0, 1], [0, 1], 'r', linewidth=4)
    plt.axis([0, 1, 0, 1])
    plt.xlabel('False Positive Rate (FPR)', fontsize=16)
    plt.ylabel('True Positive Rate (TPR)', fontsize=16)

plt.figure(figsize=(14, 7))
plot_roc_curve(false_positive_rate, true_positive_rate)
plt.show()
```



The red line in the middle represents a purely random classifier (e.g a coin flip) and therefore your classifier should be as far away from it as possible. Our Random Forest model seems to do a good job.

Of course we also have a tradeoff here, because the classifier produces more false positives, the higher the true positive rate is.

ROC AUC Score

The ROC AUC Score is the corresponding score to the ROC AUC Curve. It is simply computed by measuring the area under the curve, which is called AUC.

A classifiers that is 100% correct, would have a ROC AUC Score of 1 and a completely random classifier would have a score of 0.5.

```
from sklearn.metrics import roc_auc_score
r_a_score = roc_auc_score(Y_train, y_scores)
print("ROC-AUC-Score:", r_a_score)
```

ROC_AUC_SCORE: 0.945067587

Group C

Assignment No : 13

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 of records stored in the form 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.

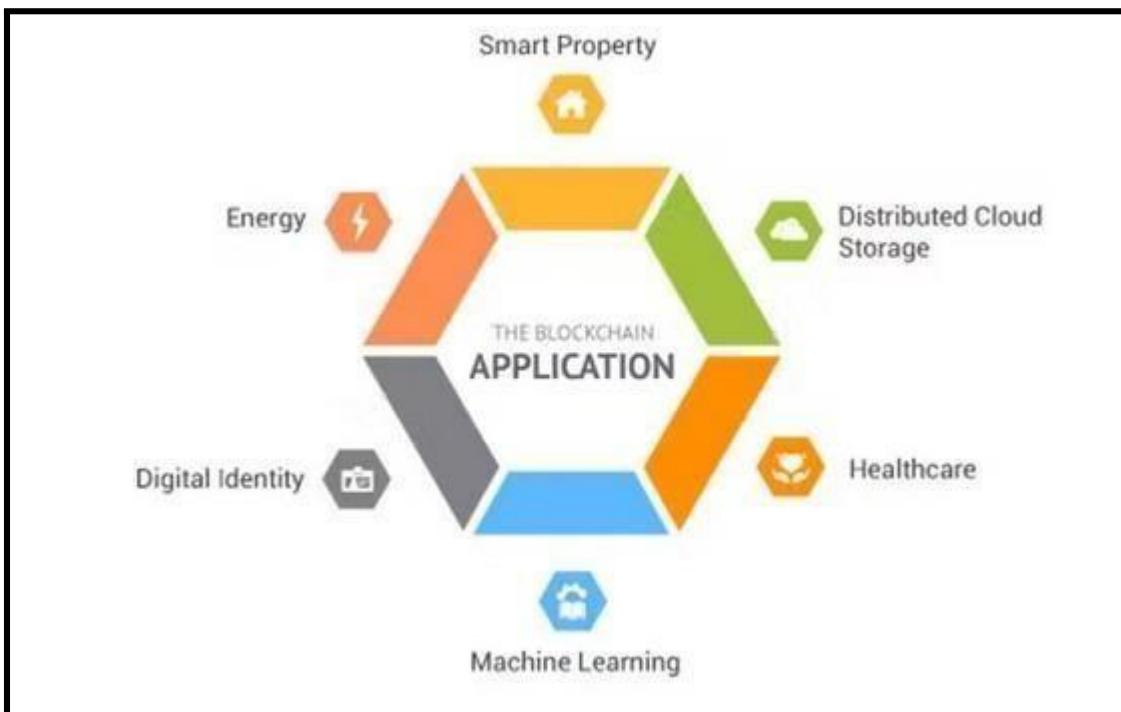
- **Immutable**

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.

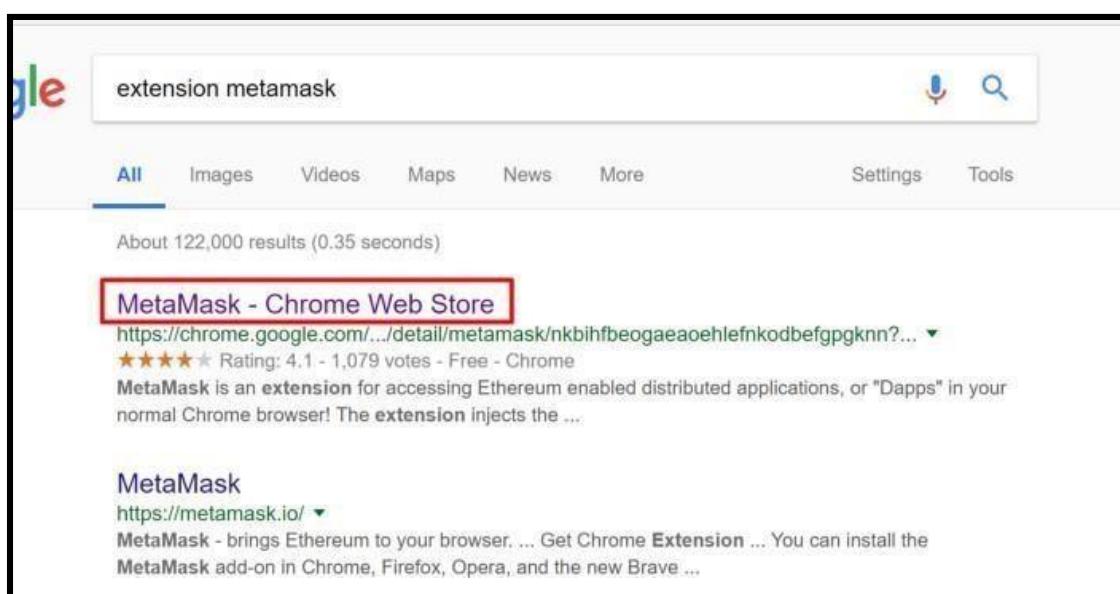
- In finance market trading, Fibonacci retracement levels are widely used in technical analysis.

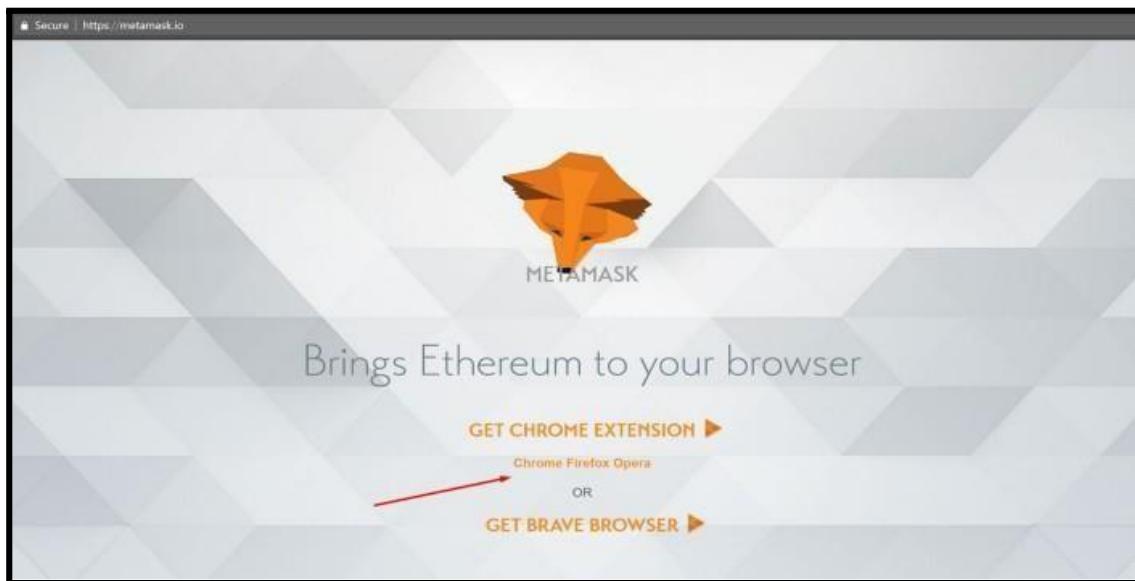
How to use MetaMask: A step by step guide

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.

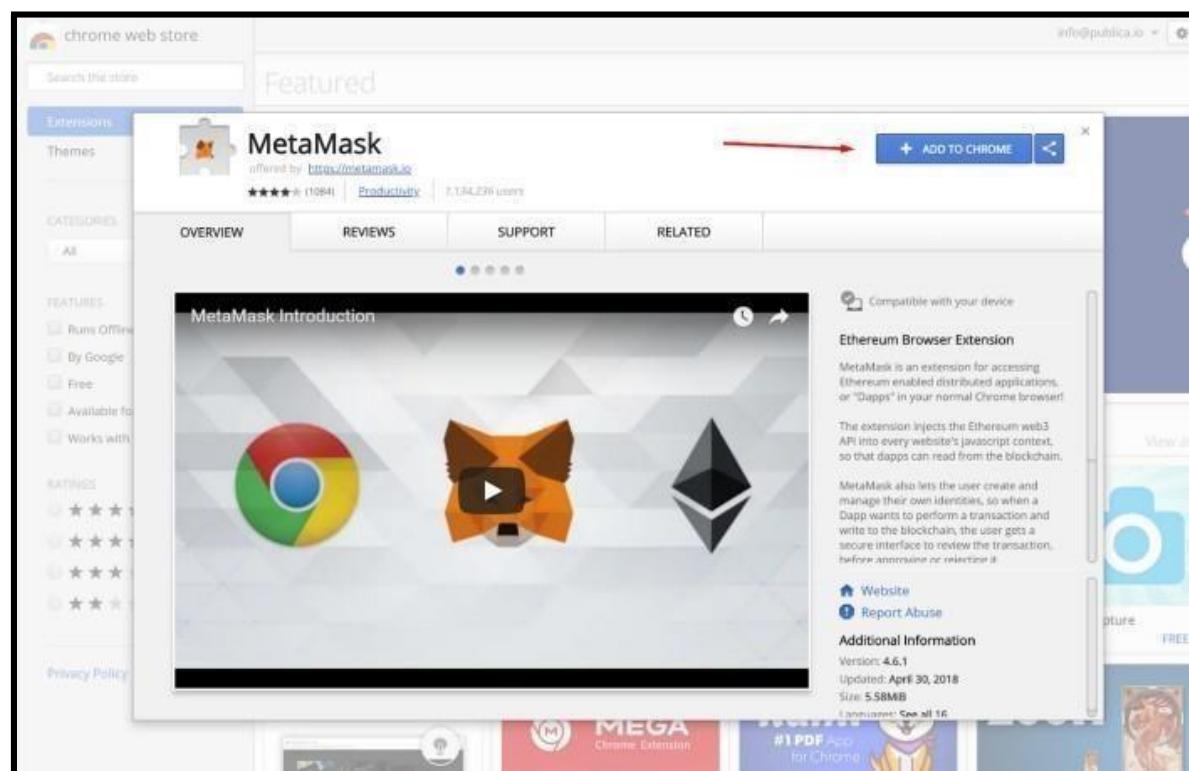
Step 1. Install MetaMask on your browser.

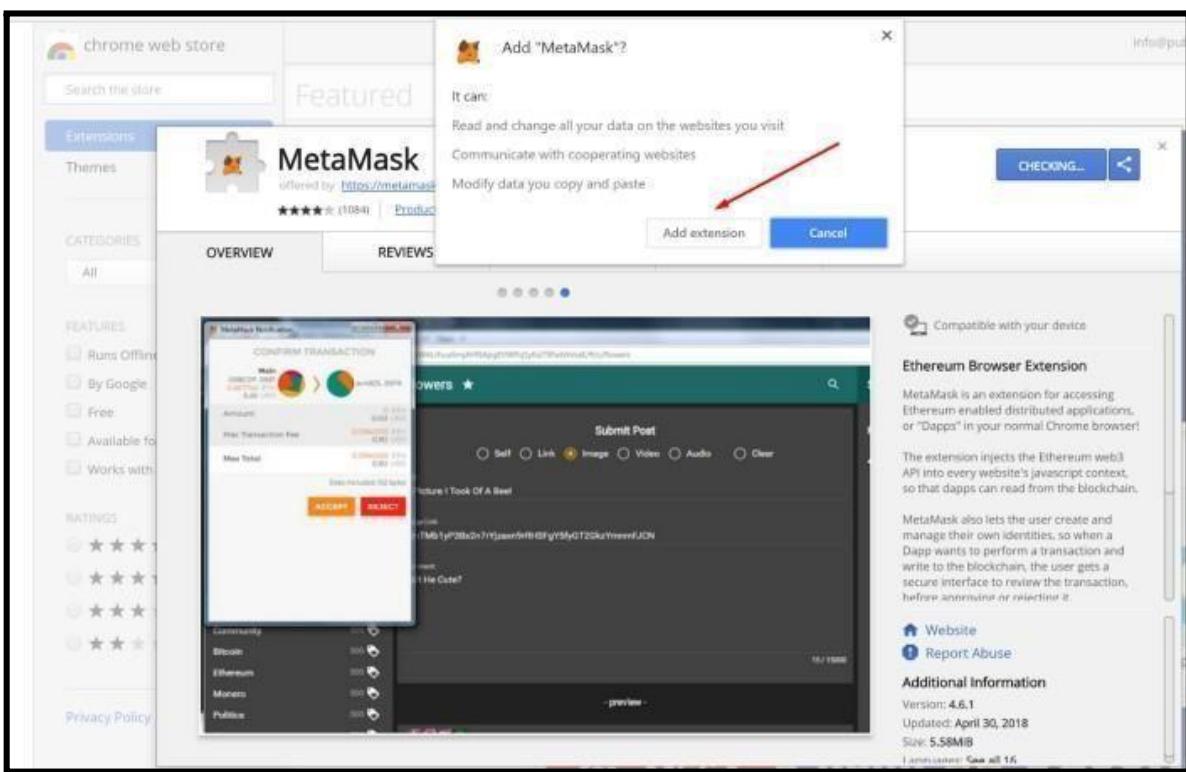
To create a new wallet, you have to install the extension first. Depending on your browser, there are different marketplaces to find it. Most browsers have MetaMask on their stores, so it's not that hard to see it, but either way, here they are [Chrome](#), [Firefox](#), and [Opera](#).





- 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.

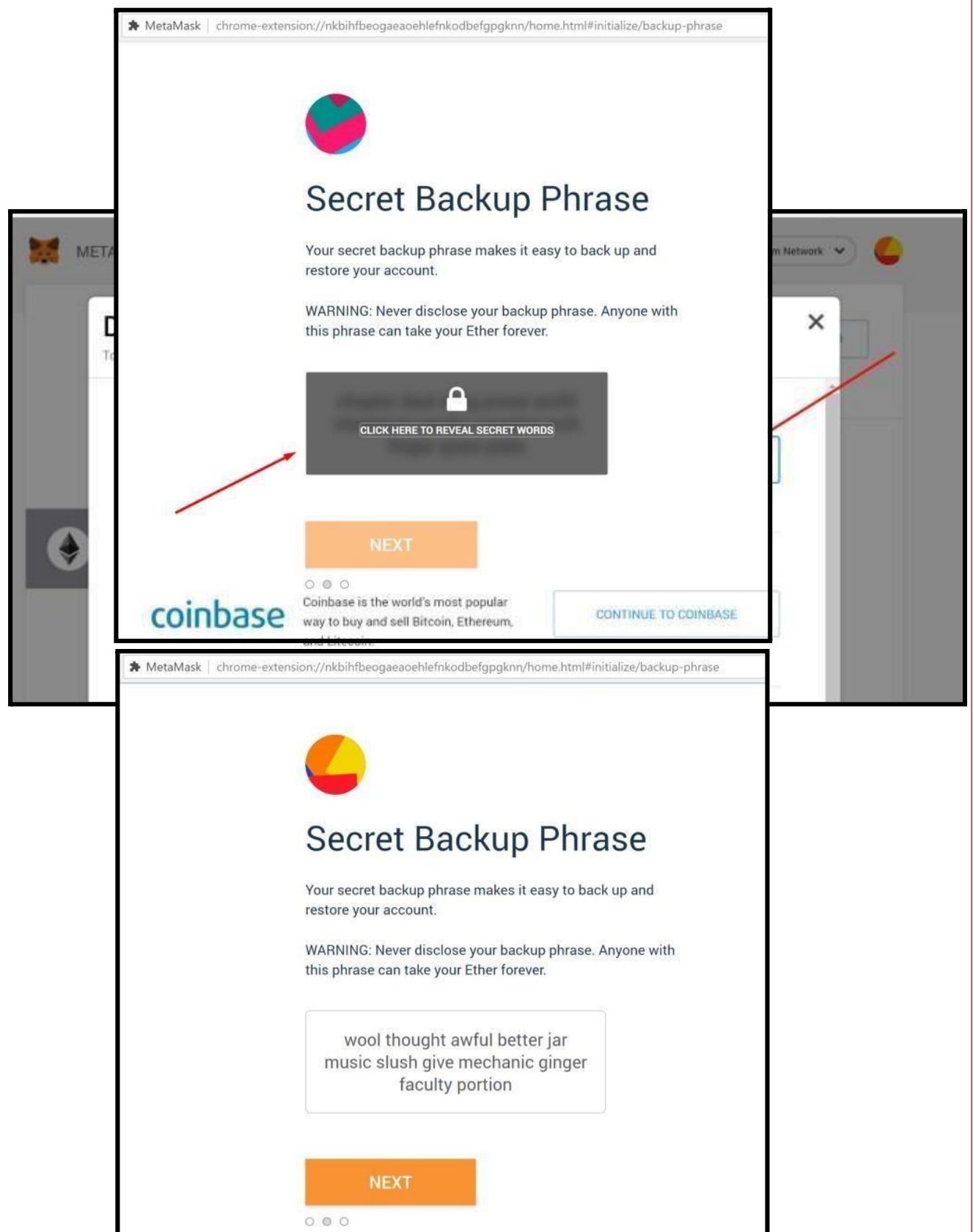
Step 2. Create an account.

- 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**.

The screenshot shows a web-based password creation interface for MetaMask. At the top, it says "MetaMask | chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#initialize/create-password". The main title is "Create Password". Below it, there are two input fields: "New Password (min 8 chars)" containing "....." and "Confirm Password" also containing ".....". A large orange button labeled "CREATE" is centered below the inputs. At the bottom left, there is a link "Import with seed phrase".

- 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.**

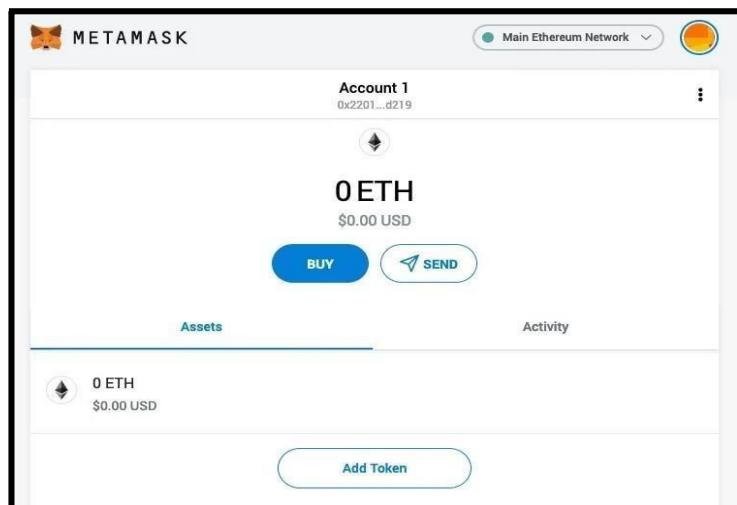
And that's it; now you have created your MetaMask account successfully. A new Ethereum wallet address has just been created for you. It's waiting for you to deposit funds, and if you want to learn how to do that, look at the next step below.

Step 3. Depositing funds.

- Click on **View Account**.

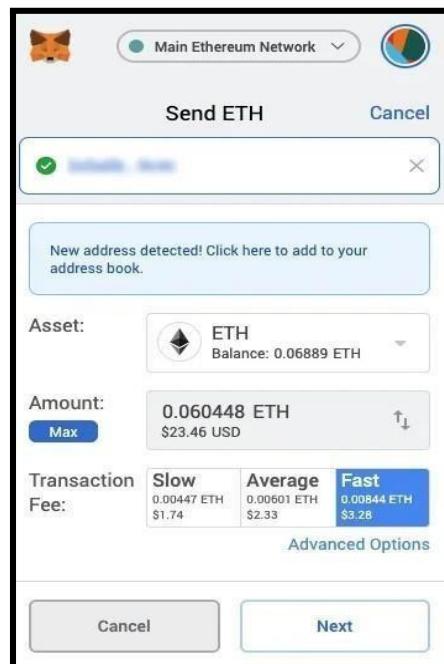
You can now see your public address and share it with other people. There are some methods to buy coins offered by MetaMask, but you can do it differently as well; you just need your address.

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>

Assignment No : 14

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

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

Prerequisite:

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

Contents for Theory:

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

Introduction to Cryptocurrency

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

How does cryptocurrency work?

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

Cryptocurrency examples

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

- **Bitcoin:**

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

- **Ethereum:**

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

- **Litecoin:**

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

- **Ripple:**

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

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

How to store cryptocurrency

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

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

Assignment Question

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

Reference link

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

Assignment No : 15

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 accounts that have sufficient funds to cover the requested withdrawal.

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

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

```
pragma solidity ^0.4.19;
```

```
contract TipJar {
    address owner; // current owner of the contract

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

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

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

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

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

which will make this bookkeeping job quite simple. (This mapping structure is much more general key/value mapping than just addresses to integers, but that's all we need here.)

Here's the code to accept deposits and track account balances:

```
pragma solidity ^0.4.19;
```

```
contract Bank {
```

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

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

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

```
}
```

Here are the new concepts in the code above:

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

It's important to note that `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 than the `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 funds to cover the requested withdrawal. If not, then the transaction aborts without making any state changes or ether transfers.
- The `balanceOf` mapping must be updated to reflect the lowered residual amount after the withdrawal.
- The funds must be sent to the sender requesting the withdrawal.

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

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

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

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

Assignment No : 16

Title of the Assignment: Write a survey report on types of Blockchains and its real time use cases.

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

Prerequisite:

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

Contents for Theory:

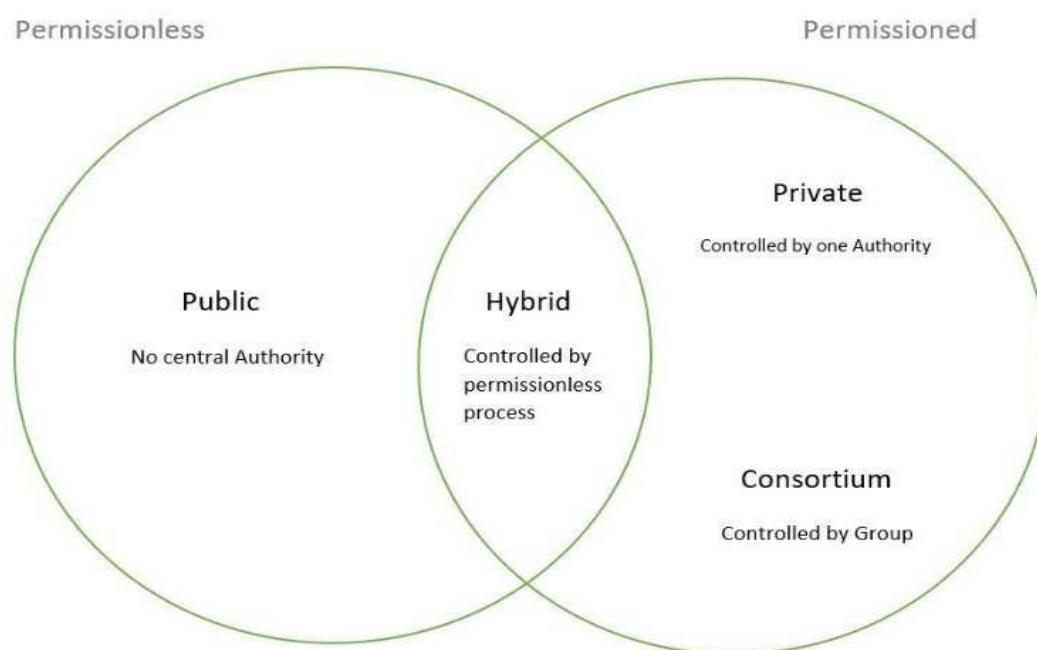
There are 4 types of blockchain:

Public Blockchain.

Private Blockchain.

Hybrid Blockchain.

Consortium Blockchain



1. Public Blockchain

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

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

Advantages:

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

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

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

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

Disadvantages:

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

Energy Consumption: Proof of work is high energy-consuming. It requires good computer hardware to participate in the network

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

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

2. Private Blockchain

These blockchains are not as decentralized as the public blockchain only selected nodes can participate in the process, making it more secure than the others.

These are not as open as a public blockchain.

They are open to some authorized users only.

These blockchains are operated in a closed network.

In this few people are allowed to participate in a network within a company/organization.

Advantages:

Speed: The rate of the transaction is high, due to its small size. Verification of each node is less time-consuming.

Scalability: We can modify the scalability. The size of the network can be decided manually.

Privacy: It has increased the level of privacy for confidentiality reasons as the businesses required.

Balanced: It is more balanced as only some user has the access to the transaction which improves the performance of the network.

Disadvantages:

Security- The number of nodes in this type is limited so chances of manipulation are there. These blockchains are more vulnerable.

Centralized- Trust building is one of the main disadvantages due to its central nature. Organizations can use this for malpractices.

Count- Since there are few nodes if nodes go offline the entire system of blockchain can be endangered.

Use Cases: With proper security and maintenance, this blockchain is a great asset to secure information without exposing it to the public eye. Therefore companies use them for internal auditing, voting, and asset management. An example of private blockchains is Hyperledger, Corda.

3. Hybrid Blockchain

It is the mixed content of the private and public blockchain, where some part is controlled by some organization and other makes are made visible as a public blockchain.

It is a combination of both public and private blockchain.

Permission-based and permissionless systems are used.

User access information via smart contracts

Even a primary entity owns a hybrid blockchain it cannot alter the transaction

Advantages:

Ecosystem: Most advantageous thing about this blockchain is its hybrid nature. It cannot be hacked as 51% of users don't have access to the network

Cost: Transactions are cheap as only a few nodes verify the transaction. All the nodes don't carry the verification hence less computational cost.

Architecture: It is highly customizable and still maintains integrity, security, and transparency.

Operations: It can choose the participants in the blockchain and decide which transaction can be made public.

Disadvantages:

Efficiency: Not everyone is in the position to implement a hybrid Blockchain. The organization also faces

some difficulty in terms of efficiency in maintenance.

Transparency: There is a possibility that someone can hide information from the user. If someone wants to get access through a hybrid blockchain it depends on the organization whether they will give or not.

Ecosystem: Due to its closed ecosystem this blockchain lacks the incentives for network participation.

Use Case: It provides a greater solution to the health care industry, government, real estate, and financial companies. It provides a remedy where data is to be accessed publicly but needs to be shielded privately.

Examples of Hybrid Blockchain are Ripple network and XRP token.

4. Consortium Blockchain

It is a creative approach that solves the needs of the organization. This blockchain validates the transaction and also initiates or receives transactions.

Also known as Federated Blockchain.

This is an innovative method to solve the organization's needs.

Some part is public and some part is private.

In this type, more than one organization manages the blockchain.

Advantages:

Speed: A limited number of users make verification fast. The high speed makes this more usable for organizations.

Authority: Multiple organizations can take part and make it decentralized at every level. Decentralized authority, makes it more secure.

Privacy: The information of the checked blocks is unknown to the public view. but any member belonging to the blockchain can access it.

Flexible: There is much divergence in the flexibility of the blockchain. Since it is not a very large decision can be taken faster.

Disadvantages:

Approval: All the members approve the protocol making it less flexible. Since one or more organizations are involved there can be differences in the vision of interest.

Transparency: It can be hacked if the organization becomes corrupt. Organizations may hide information from the users.

Vulnerability: If few nodes are getting compromised there is a greater chance of vulnerability in this blockchain

Use Cases: It has high potential in businesses, banks, and other payment processors. Food tracking of the organizations frequently collaborates with their sectors making it a federated solution ideal for their use.

Examples of consortium Blockchain are Tendermint and Multichain.

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

Assignment No : 17

Title of the Assignment: Write a program to create a Business Network using Hyperledger.

Objective of the Assignment: Students should be able to learn hyperledger .Its application and implementations

Prerequisite:

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

Contents for Theory:

Hyperledger Composer is an extensive, open development toolset and framework to make developing blockchain applications easier. The primary goal is to accelerate time to value, and make it easier to integrate your blockchain applications with the existing business systems.

- You can use Composer to rapidly develop use cases and deploy a blockchain solution in days.
- Composer allows you to model your business network and integrate existing systems and data with your blockchain applications.
- Hyperledger Composer supports the existing [Hyperledger Fabric blockchain](#) infrastructure and runtime.
- Hyperledger Composer generate business network archive (bna) file which you can deploy on existing Hyperledger Fabric network

You can use Hyperledger Composer to model business network, containing your existing assets and the transactions related to them

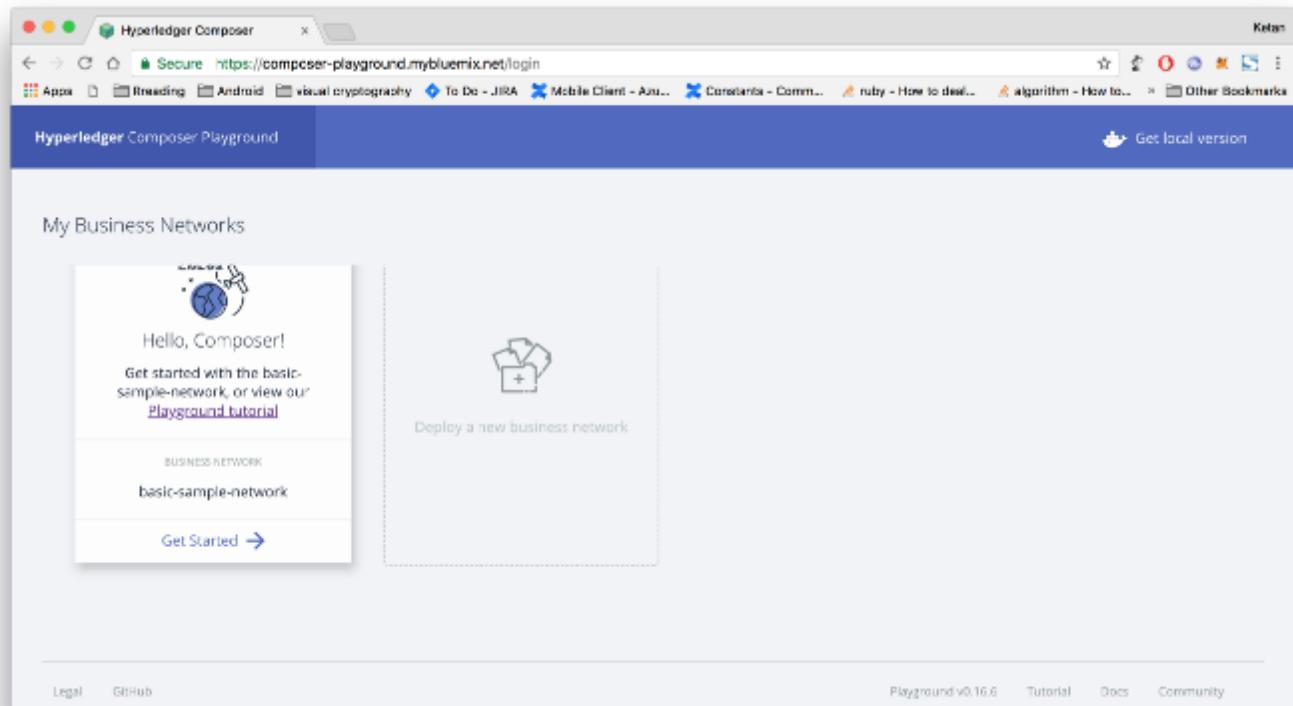
**Key Concepts of
Hyperledger
Composer**

1. Blockchain State Storage: It stores all transaction that happens in your hyperledger composer application.
It stores transaction in Hyperledger fabric network.

2. Connection Profiles: Connection Profiles to configuration JSON file which help composer to connect to Hyperledger Fabric. You can find Connection Profile JSON file in user's home directory.
3. Assets: Assets are tangible or intangible goods, services, or property, and are stored in registries. Assets can represent almost anything in a business network, for example, a house for sale, the sale listing, the land registry certificate for that house. Assets must have a unique identifier, but other than that, they can contain whatever properties you define.
4. Participants: Participants are members of a business network. They may own assets and submit transactions. Participant must have an identifier and can have any other properties.
5. Identities and ID cards: Participants can be associated with an identity. ID cards are a combination of an identity, a connection profile, and metadata. ID cards simplify the process of connecting to a business network.
6. Transactions: Transactions are the mechanism by which participants interact with assets. Transaction processing logic you can define in JavaScript and you can also emit event for transaction.
7. Queries: Queries are used to return data about the blockchain world-state. Queries are defined within a business network, and can include variable parameters for simple customisation. By using queries, data can be easily extracted from your blockchain network. Queries are sent by using the Hyperledger Composer API.
8. Events: Events are defined in the model file. Once events have been defined, they can be emitted by transaction processor functions to indicate to external systems that something of importance has happened to the ledger.
9. Access Control: Hyperledger is enterprise blockchain and access control is core feature of any business blockchain. Using Access Control rules you can define who can do what in Business networks. The access control language is rich enough to capture sophisticated conditions.
10. Historian registry: The historian is a specialised registry which records successful transactions, including the participants and identities that submitted them. The historian stores transactions as HistorianRecord assets, which are defined in the Hyperledger Composer system namespace.

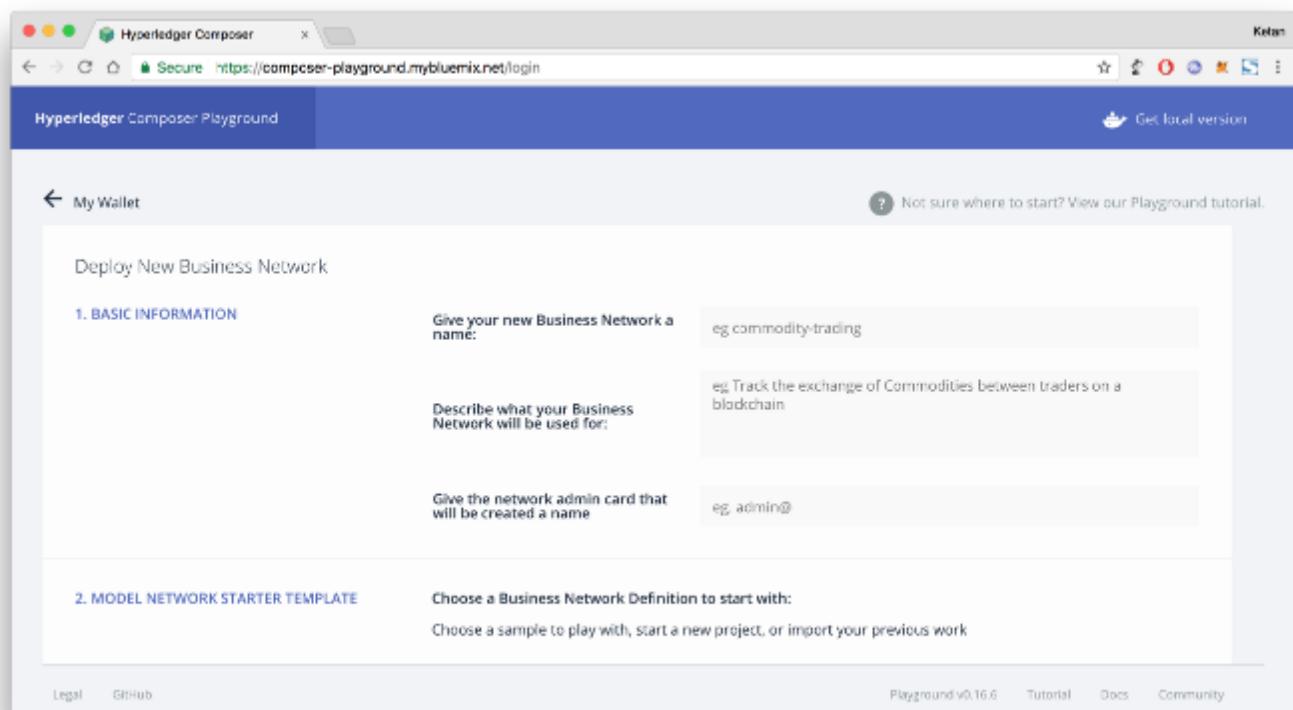
**Let's create first
Hyperledger
Composer Application**

Step 1: Start Hyperledger Composer Online version of Local. Click on Deploy a new business network



Hyperledger Composer Playground Online version

Step 2: Select empty business network



Step 3: Fill basic information, select empty business network and click “deploy” button from right pannel

The screenshot shows the Hyperledger Composer Playground interface for deploying a new business network. The main form is titled "Deploy New Business Network". It has two main sections: "1. BASIC INFORMATION" and "2. MODEL NETWORK STARTER TEMPLATE".

- 1. BASIC INFORMATION:**
 - "Give your new Business Network a name:" input field containing "hardware-assets".
 - "Describe what your Business Network will be used for:" input field containing "Hardware Assets will maintain Software company's hardware".
 - "Give the network admin card that will be created a name" input field containing "eg. admin@hardware-assets".
- 2. MODEL NETWORK STARTER TEMPLATE:**
 - "Choose a Business Network Definition to start with:" dropdown menu showing "empty-business-network" selected.
 - "Choose a sample to play with, start a new project, or import your previous work" section with options: "basic-sample-network" (with a document icon), "empty-business-network" (with a blue square icon), and a file upload area labeled "Drop here to upload or browse".

On the right side, there is a sidebar titled "hardware-assets" which displays the same information as the main form. Below the sidebar, a "CONNECTION PROFILE" section is shown, indicating it is "BASED ON empty-business-network" and "Start from scratch with a blank business network".

At the bottom of the page, there are links for "Legal", "GitHub", "Playground v0.16.6", "Tutorial", "Docs", and "Community".

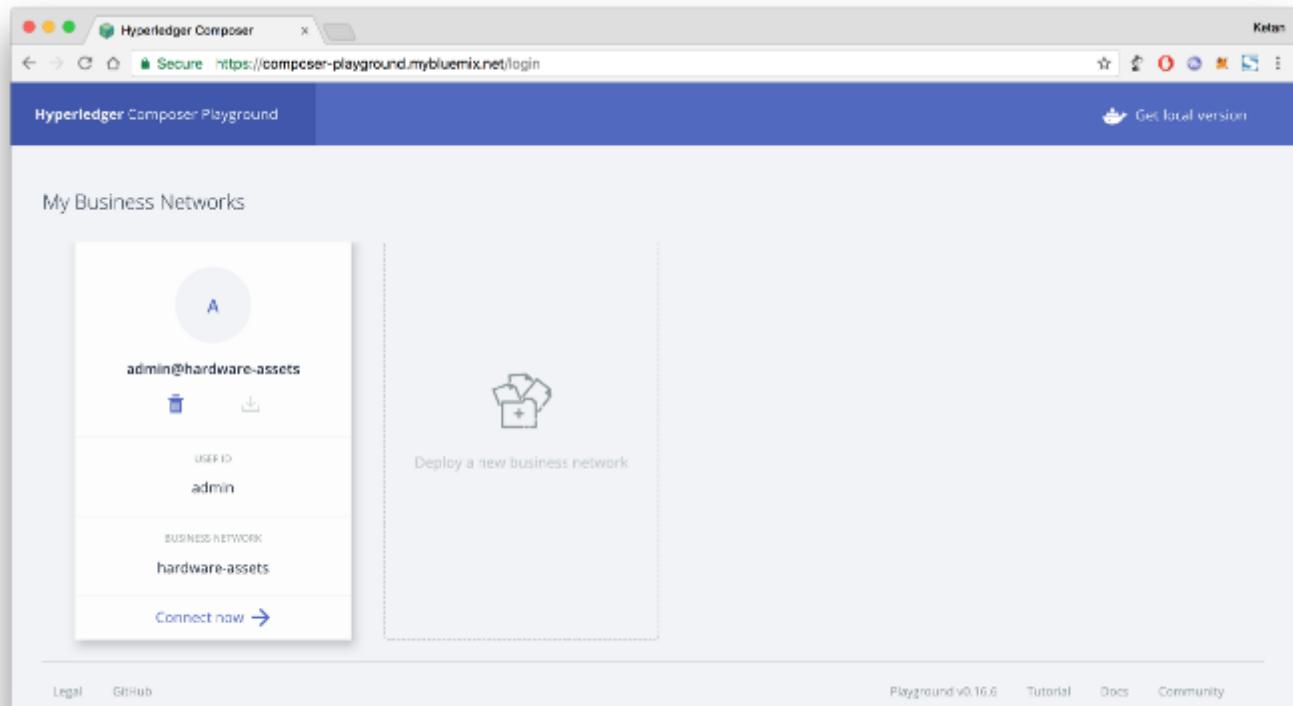
Fill basic information

This screenshot shows the same interface as the previous one, but with the "basic-sample-network" option selected in the "Choose a Business Network Definition to start with:" dropdown. The "empty-business-network" option is also visible in the list.

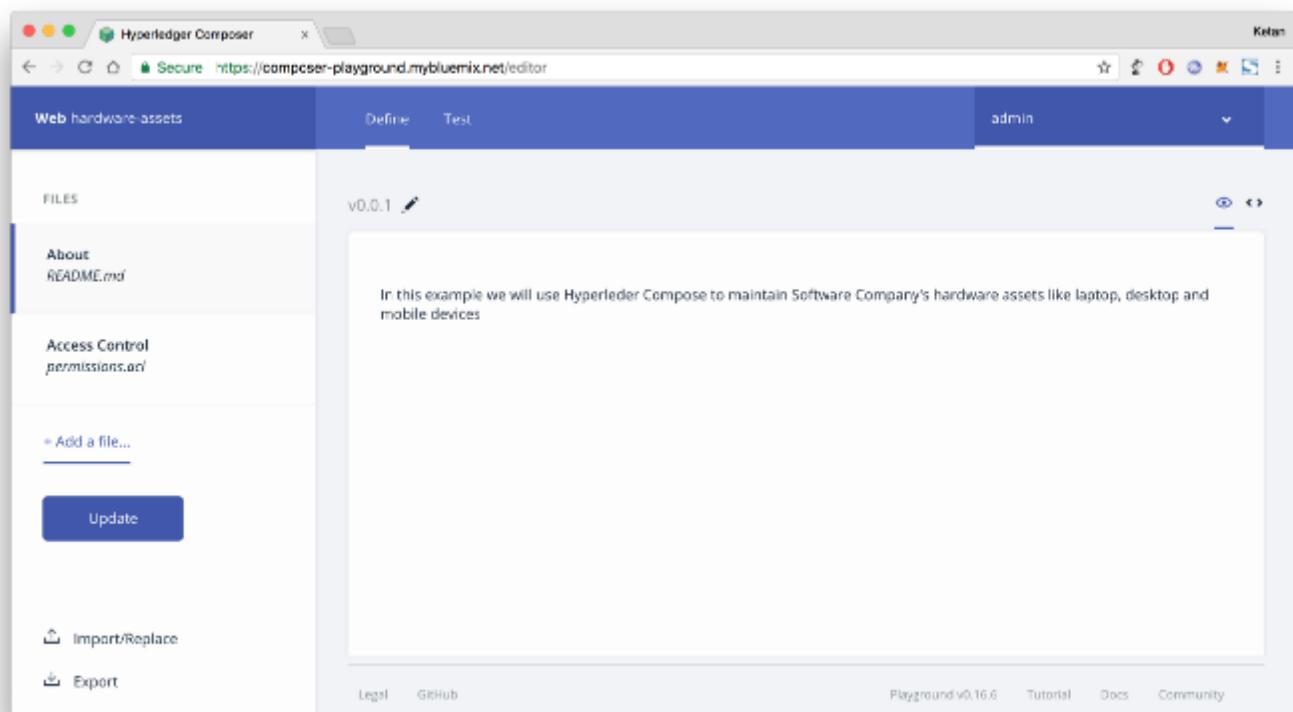
The right sidebar and connection profile details remain the same. At the bottom right, a large blue "Deploy" button is visible.

select empty business network

Step 4: Connect to “hardware-assets” business network that we have just deployed

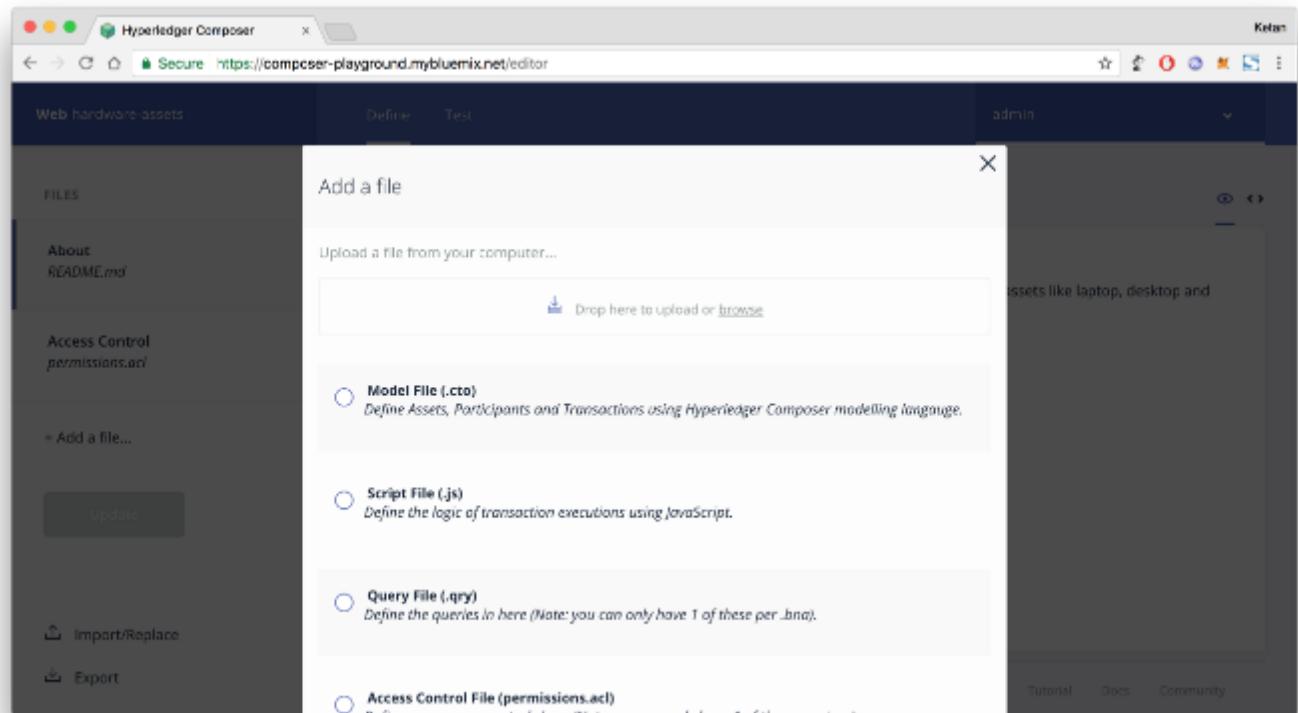


click on “connect now” button



Inside hardware-assets business network

Step 5: Click on “+Add a file...” from left panel and select “model file (.cto)”



Write following code in model file. Model file contain asset in our case it's hardware, participant in our case participants are employee of organisation and transaction as Allocate hardware to employee. Each model has extra properties. Make sure your have proper and unique namespace. In this example I am using "com.kpbird" as namespace. You can access all models using this namespace i.e. com.kpbird.Hardware,

```

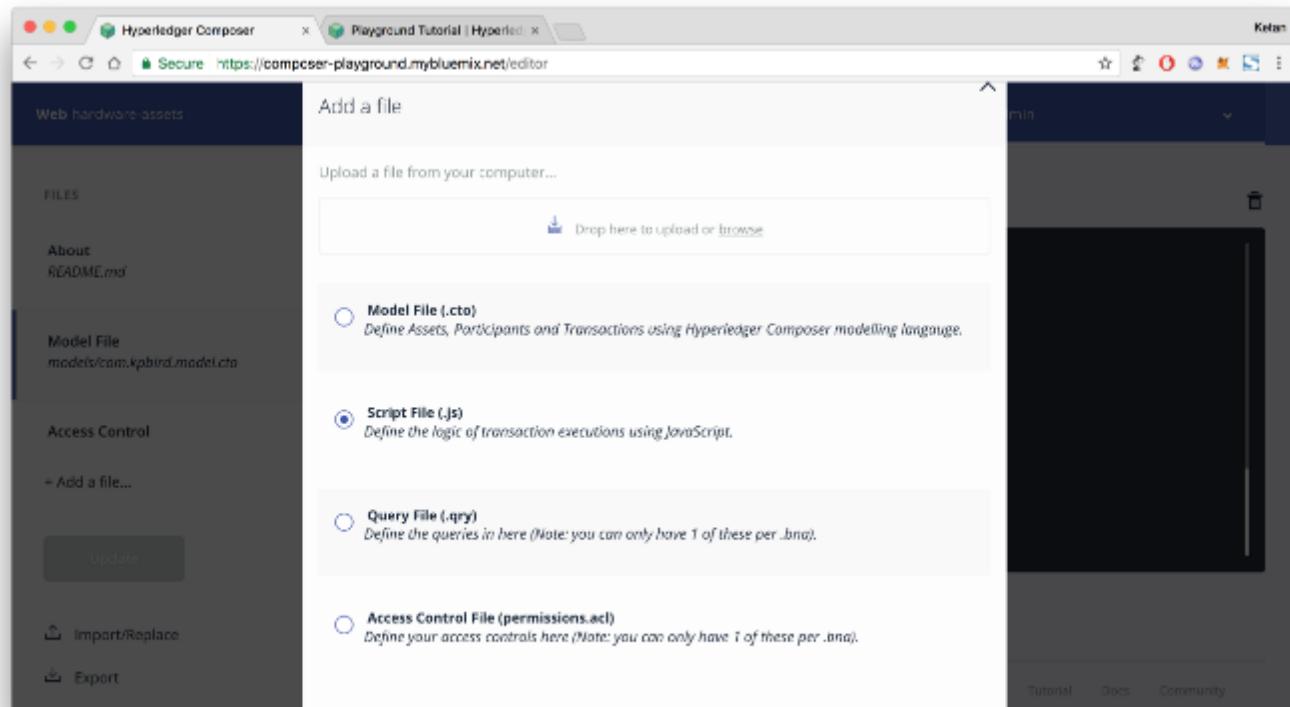
com.kpbird.Employee
/**
 * Hardware model
 */namespace com.kpbirdasset Hardware identified by hardwareId {
    o String hardwareId
    o String name
    o String type
    o String description
    o Double quantity
    → Employee owner
}
participant Employee identified by employeeId {
    o String employeeId
    o String firstName
    o String lastName
}
transaction Allocate {
    → Hardware hardware
    → Employee newOwner
}

```

Hyperledger modeling language

reference: https://hyperledger.github.io/composer/reference/cto_language.html

Step 6: Click on "+Add a file..." from left panel and select "script file (*.js)"



Write following code in Script File. In Script we can define transaction processing logic. In our case we want to allocate hardware to the employee so, we will update owner of hardware. Make sure about annotation above functions @params and @transaction

```
/**
 * Track the trade of a commodity from one trader to another
 * @param {com.kpbird.Allocate} trade – the trade to be processed
 * @transaction
 */
function allocateHardware(allocate) {
  allocate.hardware.owner = allocate.newOwner;
  return getAssetRegistry('com.kpbird.Hardware')
    .then(function(assetRegistry) {
      return assetRegistry.update(allocate.hardware);
    });
}
```

Hyperledger Composer Script file reference: https://hyperledger.github.io/composer/reference/js_scripts.html

Step 7: permissions.acl file sample is already available, Add following code in permissions.acl file.

```
/** 
 * New access control file
 */
rule AllAccess {
  description: "AllAccess – grant everything to everybody."
  participant: "ANY"
  operation: ALL
  resource: "com.kpbird.**"
  action: ALLOW
}
rule SystemACL{
  description: "System ACL to permit all access"
  participant: "org.hyperledger.composer.system.Participant"
  operation: ALL
  resource: "org.hyperledger.composer.system.**"
  action: ALLOW
}
```


Hyperledger Composer Access Control Language

reference: https://hyperledger.github.io/composer/reference/acl_language.html

Step 8: Now, It's time to test our hardware assets business network. Hyperledger composer gives "Test" facility from composer panel it self. Click on "Test" tab from top panel

The screenshot shows the Hyperledger Composer Test interface. The left sidebar has sections for PARTICIPANTS (Employee), ASSETS (Hardware selected), and TRANSACTIONS (All Transactions). A blue button at the bottom left says "Submit Transaction". The main area is titled "Asset registry for com.kpbird.Hardware" and contains a table with columns "ID" and "Data". A large, stylized wrench icon is centered below the table. A message at the bottom states "This registry is empty! To create resources in this registry click create new at the top of this page". The bottom right corner includes links for Legal, GitHub, Playground v0.16.6, Tutorial, Docs, and Community.

Test feature of Hyperledger Composer

Step 9: Create Assets. Click on "Hardware" from left panel and click "+ Create New Assets" from right top corner and add following code. We will create Employee#01 in next step. Click on "Create New" button

```
{  
  "$class": "com.kpbird.Hardware",  
  "hardwareId": "MAC01",  
  "name": "MAC Book Pro 2015",  
  "type": "Laptop",  
  "description": "Mac Book Pro",  
  "quantity": 1,  
  "owner": "resource:com.kpbird.Employee#01"  
}
```


The screenshot shows the Hyperledger Composer web interface. The left sidebar has tabs for PARTICIPANTS, ASSETS, and TRANSACTIONS, with ASSETS selected. Under ASSETS, 'Hardware' is selected. The main area is titled 'Asset registry for com.kpbird.Hardware'. It shows a table with one row for 'MAC01'. The 'Data' column for 'MAC01' displays the following JSON:

```
{
  "$class": "com.kpbird.Hardware",
  "hardwareId": "MAC01",
  "name": "MAC Book Pro 2015",
  "type": "laptop",
  "description": ""
}
```

Buttons for '+ Create New Asset' and edit/delete icons are visible. At the bottom, there are links for Legal, GitHub, and playground documentation.

After adding Hardware assets

Steps 10: Let's create participants. Click "Employee" and click "+ Create New Participants" and add following code. We will add two employees

```
{
  "$class": "com.kpbird.Employee",
  "employeeId": "01",
  "firstName": "Ketan",
  "lastName": "Parmar"
}
```

Click on "Create New" on dialog

```
{
  "$class": "com.kpbird.Employee",
  "employeeId": "02",
  "firstName": "Nirja",
  "lastName": "Parmar"
}
```

The screenshot shows the Hyperledger Composer web interface. The left sidebar has tabs for PARTICIPANTS, ASSETS, and TRANSACTIONS, with PARTICIPANTS selected. Under PARTICIPANTS, 'Employee' is selected. The main area is titled 'Participant registry for com.kpbird Employee'. It shows a table with two rows, '01' and '02'. The 'Data' column for '01' displays the following JSON:

```
{
  "$class": "com.kpbird.Employee",
  "employeeId": "01",
  "firstName": "Ketan",
  "lastName": "Parmar"
}
```

The 'Data' column for '02' displays the following JSON:

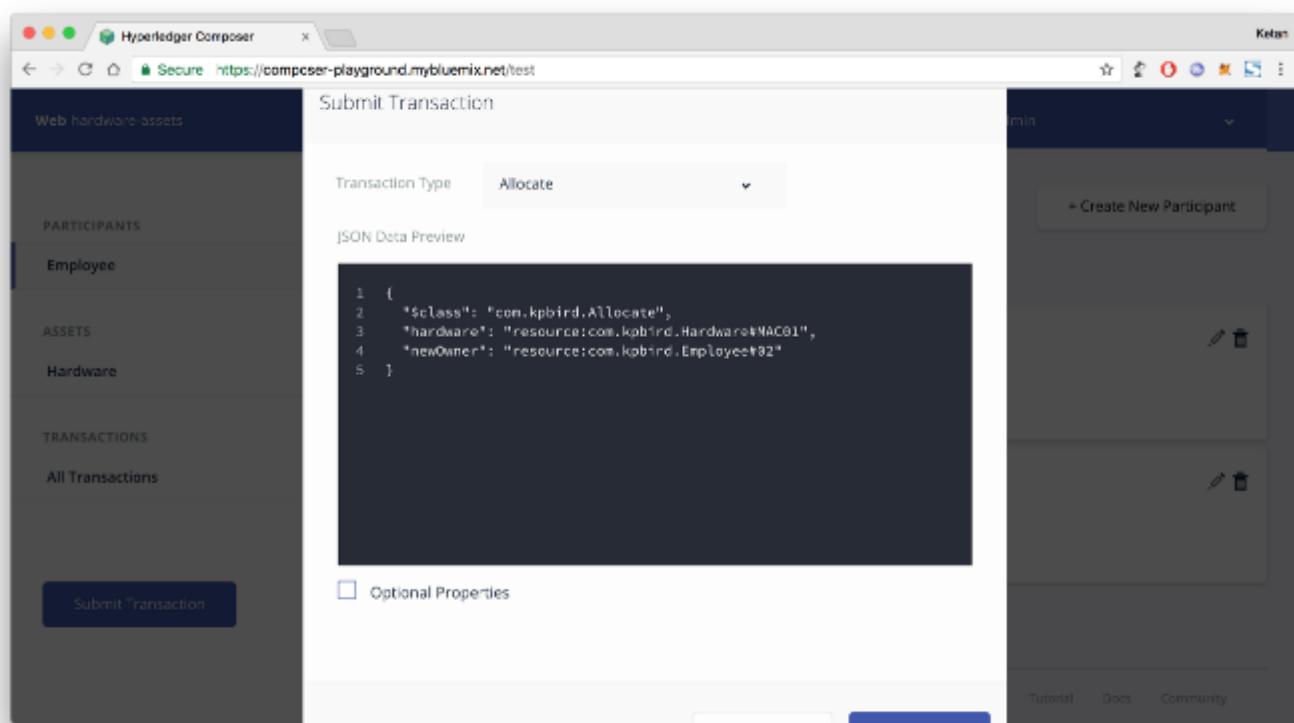
```
{
  "$class": "com.kpbird.Employee",
  "employeeId": "02",
  "firstName": "Nirja",
  "lastName": "Parmar"
}
```

Buttons for '+ Create New Participant' and edit/delete icons are visible. At the bottom, there are links for Legal, GitHub, and playground documentation.

We have two employees

AJEEKYA D Y PATIL SCHOOL OF ENGINEERING, LOHEGAON

Step 11: It's time to do transaction, We will allocate Macbook Pro from Ketan (Employee#01) to Nirja (Employee#02). Click on "Submit Transaction" button from left panel. In Transaction dialog, We can see all transaction functions on top "Transaction Type" dropdown.

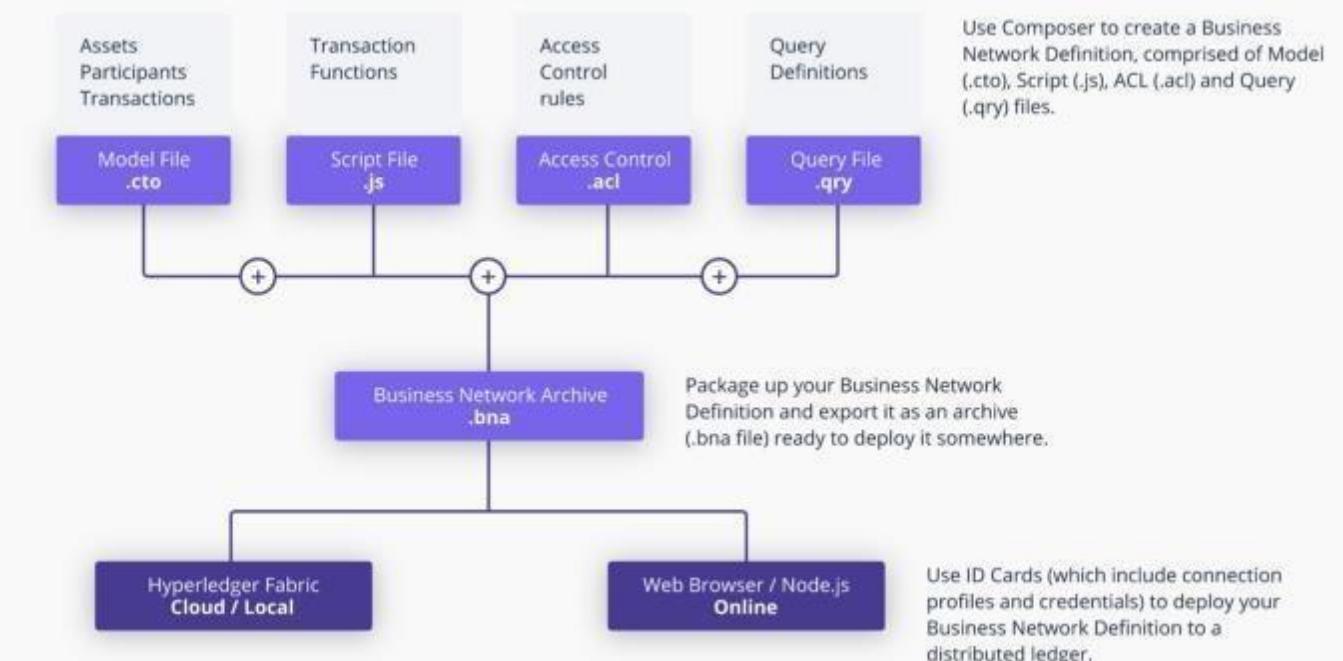


Submit Transaction Dialog

```
{
  "$class": "com.kpbird.Allocate",
  "hardware": "resource:com.kpbird.Hardware#MAC01",
  "newOwner": "resource:com.kpbird.Employee#02"
}
```

Now, We are allocating Mac01 to Employee 02. Click Submit button after update above JSON in Transaction Dialog. As soon as you hit submit button. Transaction processed and Transaction Id will generate.





source: <https://hyperledger.github.io/composer/introduction/introduction>

Step 14: Start Docker and run following commands from ~/fabric-tools directory

Install business network to Hyperledger Fabric, If business network is already installed you can use “update”

Step 13: Now, It's time to deploy “hardware-assets” business network to Hyperledger Fabric. Click on “Define” tab from top panel and click “Export” button from left panel. Export will create hardware-assets.bna file.



Following command will deploy and start hardware-assets.bna file. Change hardware-assets.bna file before you execute following command. networkadmin.card file will generate in ~/fabric-tools directory from previous command.

```
$composer network start --card PeerAdmin@hlfv1 --networkAdmin admin --networkAdminEnrollSecret adminpw --archiveFile /Users/ketan/Downloads/hardware-assets.bna --file networkadmin.card
```

```
Ketan-Parmar:fabric-tools ketan$ composer network start --card PeerAdmin@hlfv1 --networkAdmin admin --networkAdminEnrollSecret adminpw --archiveFile /Users/ketan/Downloads/hardware-assets.bna --file networkadmin.card
Starting business network from archive: /Users/ketan/Downloads/hardware-assets.bna
Business network definition:
  Identifier: hardware-assets@0.0.1
  Description: Hardware Assets will maintain Software company's hardware

Processing these Network Admins:
  userName: admin

✓ Starting business network definition. This may take a minute...
Successfully created business network card:
  Filename: networkadmin.card

Command succeeded
```

Download hardware-assets.bna file

To connect business network you need connection card. so we can import networkadmin.card using following .bna is Business Network Archive file which contains model, script, network access and query file

```
$composer card import -f networkadmin.card
```

To make sure networkadmin.card successfully install you can list cards using following command

```
$composer card list
```


Ketan-Parmar:fabric-tools ketan\$ composer card list
The following Business Network Cards are available:

Connection Profile: hlfv1

Card Name	UserId	Business Network
admin@hardware-assets	admin	hardware-assets
PeerAdmin@trade-network	PeerAdmin	trade-network
admin@trade-network	admin	trade-network
PeerAdmin@hlfv1	PeerAdmin	

Issue `composer card list --name <Card Name>` to get details a specific card

Command succeeded

Following command will make sure that our hardware-assets business network is successfully running in Hyperledger Fabric.

\$composer network ping – card admin@hardware-assets

[Ketan-Parmar:fabric-tools ketan\$ composer network ping --card admin@hardware-assets
The connection to the network was successfully tested: hardware-assets
version: 0.16.0
participant: org.hyperledger.composer.system.NetworkAdmin#admin

Command succeeded

Now It's time to interact with REST API. To develop Web or Mobile Application we require REST API. you can run following command to generate REST API for hardware-assets business network.

\$composer-rest-server

[Ketan-Parmar:fabric-tools ketan\$ composer-rest-server
? Enter the name of the business network card to use: admin@hardware-assets
? Specify if you want namespaces in the generated REST API: always use namespaces
? Specify if you want to enable authentication for the REST API using Passport: No
? Specify if you want to enable event publication over WebSockets: Yes
? Specify if you want to enable TLS security for the REST API: No

To restart the REST server using the same options, issue the following command:
`composer-rest-server -c admin@hardware-assets -n always -w true`

Discovering types from business network definition ...
Discovered types from business network definition
Generating schemas for all types in business network definition ...
Generated schemas for all types in business network definition
Adding schemas for all types to Loopback ...
Added schemas for all types to Loopback
Web server listening at: http://localhost:3000
Browse your REST API at http://localhost:3000/explorer

rest server will ask few basic information before generate rest api

The screenshot shows the Hyperledger Composer REST API documentation. It lists three main models: **com_kpbird_Allocate**, **com_kpbird_Employee**, and **com_kpbird_Hardware**. Each model has associated REST API operations:

- com_kpbird_Allocate**:
 - GET /com.kpbird.Allocate**: Find all instances of the model matched by filter from the data source.
 - POST /com.kpbird.Allocate**: Create a new instance of the model and persist it into the data source.
 - GET /com.kpbird.Allocate/{id}**: Find a model instance by {{id}} from the data source.
- com_kpbird_Employee**:
 - GET /com.kpbird.Employee**: Find all instances of the model matched by filter from the data source.
 - POST /com.kpbird.Employee**: Create a new instance of the model and persist it into the data source.
 - GET /com.kpbird.Employee/{id}**: Find a model instance by {{id}} from the data source.
 - HEAD /com.kpbird.Employee/{id}**: Check whether a model instance exists in the data source.
 - PUT /com.kpbird.Employee/{id}**: Replace attributes for a model instance and persist it into the data source.
 - DELETE /com.kpbird.Employee/{id}**: Delete a model instance by {{id}} from the data source.
- com_kpbird_Hardware**:
 - GET /com.kpbird.Hardware**: Find all instances of the model matched by filter from the data source.
 - POST /com.kpbird.Hardware**: Create a new instance of the model and persist it into the data source.

REST API for our hardware assets

The screenshot shows the Hyperledger Composer REST API documentation. It lists two main models: **com_kpbird_Employee** and **com_kpbird_Hardware**, and a section for **System**:

- com_kpbird_Employee**:
 - PUT /com.kpbird.Employee/{id}**: Replace attributes for a model instance and persist it into the data source.
 - DELETE /com.kpbird.Employee/{id}**: Delete a model instance by {{id}} from the data source.
- com_kpbird_Hardware**:
 - GET /com.kpbird.Hardware**: Find all instances of the model matched by filter from the data source.
 - POST /com.kpbird.Hardware**: Create a new instance of the model and persist it into the data source.
 - GET /com.kpbird.Hardware/{id}**: Find a model instance by {{id}} from the data source.
 - HEAD /com.kpbird.Hardware/{id}**: Check whether a model instance exists in the data source.
 - PUT /com.kpbird.Hardware/{id}**: Replace attributes for a model instance and persist it into the data source.
 - DELETE /com.kpbird.Hardware/{id}**: Delete a model instance by {{id}} from the data source.
- System** (General business network methods):
 - GET /system/historian**: Get all Historian Records from the Historian
 - GET /system/historian/{id}**: Get the specified Historian Record from the Historian
 - GET /system/identities**: Get all Identities from the Identity registry
 - GET /system/identities/{id}**: Get the specified identity from the Identity registry

REST API methods for all operations

Conclusion: In this way we have learnt about hyperledger and its use case in business world.

Assignment No : 18
MINI PROJECT 3

Code :- <https://github.com/sherwyn11/E-Voting-App>