



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF ENGINEERING & TECHNOLOGY

(Formerly SRM University, Under section 3 of UGC Act, 1956)

**S.R.M. NAGAR, KATTANKULATHUR –603 203, KANCHEEPURAM
DISTRICT**

SCHOOL OF COMPUTING

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEM

Course Code: 18CSE305J

Course Name: Artificial Intelligence

LAB REPORT

NAME: Arnav Kumar

REG.NO.: RA1911027010040

SECTION: N1

CSE – BIG DATA ANALYTICS

TABLE OF CONTENT

SR. NO.	NAME OF EXPERIMENT	PAGE NO.
1	TOY PROGRAM IMPLEMENTATION	3
2	AGENTS AND REAL WORLD PROBLEMS	8
3	CSP IMPLEMENTATION	11
4	IMPLEMENTATION OF BFS AND DFS IN AN APPLICATION AND CHECKING IT PERFORMANCE	18
5	IMPLEMENTATION OF INFORMED BFS AND A* - INFORMED SEARCH ALGORITHM	24
6	IMPLEMENTATION OF UNCERTAIN METHODS FOR AN APPLICATION FUZZY LOGIC AND DEMPSTER SHAFER THEORY	31
7	IMPLEMENTATION OF UNIFICATION AND RESOLUTION FOR REAL WORLD PROBLEMS	37
8	IMPLEMENTATION OF MACHINE LEARNING ALGORITHMS FOR AN APPLICATION	41
9	IMPLEMENTATION OF NLP PROGRAMS	48
10	IMPLEMENTATION OF DEEP LEARNING ALGORITHMS FOR AN APPLICATION	52

Date: 9/1/2022 Ex No:1	Title of the Lab TOY PROGRAM IMPLEMENTATION	Name: Arnav Kumar Registration Number: RA1911027010040 Section:N1 Lab Batch:2 Day Order:3
---	--	---

a) Camel and Banana

AIM:To solve, understand and implement camel and banana problem.

Description of the Concept or Problem given:-

A merchant has 3000 bananas and a camel. He wants to transport the maximum number of bananas to a destination which is 1000 km away, using only one camel as a mode of transportation. The camel cannot carry more than 1000 bananas at a time and eats a banana every km it travels. What is the maximum number of bananas that can be transferred to the destination using only camel (no other mode of transportation is allowed).

Manual Solution:-

The camel first takes 1000 bananas, moves 1km and drops 998 bananas there, moves back by 1km, carry 1000 more bananas, move forward by 1km drop 998 bananas there again. Then it goes back by 1km again, carries 1000 bananas and moves forward by 1km yet again. Now there are 2995 bananas in the first km mark, so it has eaten 5 bananas for this cycle. This cycle would continue until the camel eats 1000 bananas, i.e., until the 200km mark. Now that the camel has to carry only 2000 bananas in the cycle of movement the number of bananas eaten by the camel is reduced by 2 for each cycle. This will continue yet again till the camel has to carry only 1000 bananas, the camel would have travelled this way for 333km and 533km in total. After this point the camel can just move forward for the rest of the journey. So, the camel would have to travel 467 km with 1000 bananas, and at point B it would still have 533 bananas.

Program Implementation [Coding]:-

```
def camel_and_banana(total_bananas,load_capacity,distance):  
    lose=0  
    left_bananas=total_bananas  
    for i in range(distance):  
        while left_bananas>0:  
            left_bananas=left_bananas-load_capacity  
            if left_bananas==1:  
                lose=lose-1  
            lose=lose+2  
    lose=lose-1
```

```
    left_bananas=total_bananas-lose

    if left_bananas==0:
        break

    return left_bananas

total_bananas=int(input("Enter total numbers of bananas"))
load_capacity=int(input("Enter maximum load capacity of camel"))
distance=int(input("Enter distance to be covered"))
bananas_left=camel_and_banana(total_bananas,load_capacity,distance)

print("The maximum number of bananas that can be transferred to the destination using only
camel is:- ",bananas_left)
```

Screenshots of the Outputs

```
Enter total numbers of bananas3000
Enter maximum load capacity of camel1000
Enter distance to be covered1000

bananas_left=camel_and_banana(total_bananas,load_capacity,distance)
```

Result:- Implemented and verified camel and banana problem.

a) Water and Jug

AIM:-To solve, understand and implement water and jug problem.

Description of the Concept or Problem given:-

We have 3 water jugs. They can hold 12L, 8L, 5L respectively without marking. The initial state is (12,0,0) and the final state should be (6,6,0).

Manual Solution:-

- a) First, we pour 8L into the middle jug and transfer the rest 4L into the third jug from first jug,(0,8,4)
- b) Then we transfer the 8L from middle jug to the first jug, and transfer the 4L into the middle jug ,(8,4,0)
- c) Now transfer 5L from first jug to the third jug, (3,4,5)
- d) Then add the 4L from the third jug and fill up the middle jug completely. (3,8,1)
- e) Now transfer 8L into the first jug from the middle jug and transfer 1L from the third jug to the middle jug. (11,1,0)
- f) Add 5L from the first jug to the third jug. (6,1,5)

- g) Finally transfer the 5L from third jug to the middle jug and we have the final state.
(6,6,0)

Program Implementation [Coding]:-

```
initial_capacity=(12,8,5)
x=initial_capacity[0]
y=initial_capacity[1]
z=initial_capacity[2]
memory={}
ans=[]
def get_all_states(state):
    a=state[0]
    b=state[1]
    c=state[2]
    if(a==6 and b==6):
        ans.append(state)
        return True
    if((a,b,c) in memory):
        return False
    memory[(a,b,c)] = 1
    if(a>0):
        if(a+b<=y):
            if( get_all_states((0,a+b,c)) ):
                ans.append(state)
                return True
        else:
            if( get_all_states((a-(y-b), y, c)) ):
                ans.append(state)
                return True
    if(a+c<=z):
        if( get_all_states((0,b,a+c)) ):
            ans.append(state)
            return True
    else:
        if( get_all_states((a-(z-c), b, z)) ):
            ans.append(state)
```

```
        return True

if(b>0):
    if(a+b<=x):
        if( get_all_states((a+b, 0, c)) ):
            ans.append(state)
            return True
    else:
        if( get_all_states((x, b-(x-a), c)) ):
            ans.append(state)
            return True
    if(b+c<=z):
        if( get_all_states((a, 0, b+c)) ):
            ans.append(state)
            return True
    else:
        if( get_all_states((a, b-(z-c), z)) ):
            ans.append(state)
            return True
if(c>0):
    if(a+c<=x):
        if( get_all_states((a+c, b, 0)) ):
            ans.append(state)
            return True
    else:
        if( get_all_states((x, b, c-(x-a))) ):
            ans.append(state)
            return True
    if(b+c<=y):
        if( get_all_states((a, b+c, 0)) ):
            ans.append(state)
            return True
    else:
        if( get_all_states((a, y, c-(y-b))) ):
            ans.append(state)
            return True
```

```
        return False
initial_state=(12,0,0)
get_all_states(initial_state)
ans.reverse()
for i in ans:
    print(i)
```

Screenshots of the Outputs

```
(12, 0, 0)
(4, 8, 0)
(0, 8, 4)
(8, 0, 4)
(8, 4, 0)
(3, 4, 5)
(3, 8, 1)
(11, 0, 1)
(11, 1, 0)
(6, 1, 5)
(6, 6, 0)
```

Result:- Implemented and verified water and jug problem.

Signature of the Student

Arnav Kumar

Date: 17/1/2022 Ex No:2	Title of the Lab AGENTS AND REAL WORLD PROBLEMS	Name:Arnav Kumar Registration Number: RA1911027010040 Section:N1 Lab Batch:2 Day Order:3
--	--	--

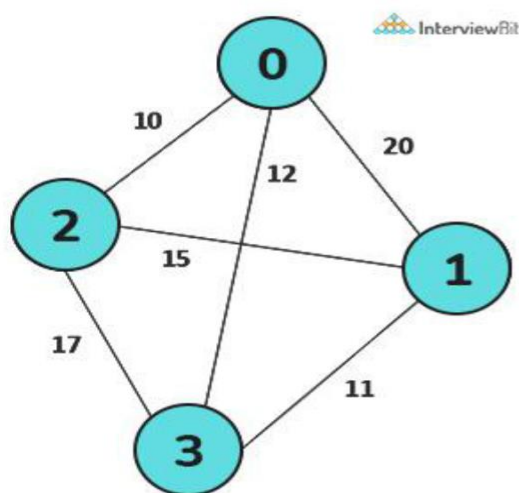
a)Traveling Salesman Problem

AIM: To implement Traveling Salesman Problem in Python

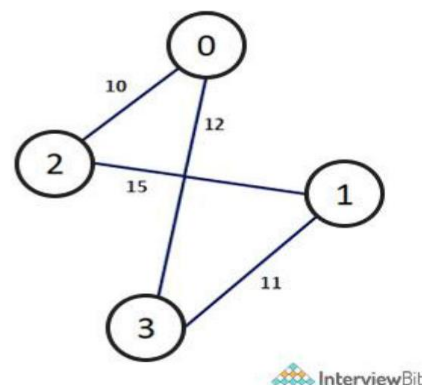
Description of the Concept or Problem given:-

Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.

Manual Solution:-



The Shortest Path Covering All The Nodes



From node 0 the shortest distance is 10, so move to node 2.

From node 2 the shortest distance is 15, so move to node 1.

From node 1 the shortest distance is 11, so move to node 3

From node 3 the shortest distance is 12, so move back to node 0.

So, in this case the shortest possible route would be of $10+15+11+12=48$

Program Implementation [Coding]:-

```

from sys import maxsize
from itertools import permutations
V = 4
def TSP(graph, s):
    vertex = []
    for i in range(V):
        if i != s:

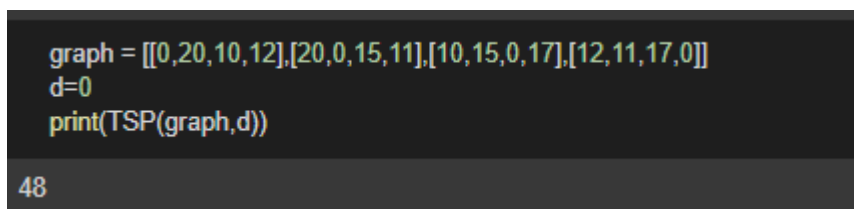
```



```
        vertex.append(i)
    min_path = maxsize
    next_permutation=permutations(vertex)
    for i in next_permutation:
        current_pathweight = 0
        k = s
        for j in i:
            current_pathweight += graph[k][j]
            k = j
        current_pathweight += graph[k][s]
        min_path = min(min_path, current_pathweight)
    return min_path

graph = [[0,20,10,12],[20,0,15,11],[10,15,0,17],[12,11,17,0]]
d=0
print(TSP(graph,d))
```

Screenshots of the Outputs



```
graph = [[0,20,10,12],[20,0,15,11],[10,15,0,17],[12,11,17,0]]
d=0
print(TSP(graph,d))

48
```

Result:- Implemented Travelling Salesman Problem in Python

b) Medical Diagnosis System

AIM: To implement Medical Diagnosis System in Python

Description of the Concept or Problem given:-

Medical diagnosis is basically a pattern classification phenomena: based on some input provided by a patient, an expert gives a conclusion on the basis of its knowledge, which is normally stored in a binary form, and finally the result is calculated i.e. either the patient suffering from a certain disease or not.

Manual Solution:-

The Medical System programs asks the patient a series of questions, to which patient replies in yes or no and then, the system uses if and elif conditions to diagnose the problem of the patient.

Program Implementation [Coding]:-

```
fever = input("Do you have a fever (y/n):- ")
rashes= input("Do you have a rash (y/n):- ")
runnyNose = input("Do you have a runny nose (y/n):-")
conjunctivitis= input("Do you have a conjunctivitis (y/n):-")
cough= input("Do you have a cough (y/n):-")
headAche=input("Do you have a head ache (y/n):-")
bodyAche= input("Do you have body ache (y/n):-")
chills= input("Do you have chills (y/n):-")
soreThroat= input("Do you have a sore throat (y/n):-")
sneezing= input("Do you have sneezing (y/n):-")
swollenGlands= input("Do you have swollen glands (y/n):-")
if fever=='y' and headAche== 'y' and runnyNose=='y' and rashes=='y':
    print("Result: You are diagnosed with German measles")
elif cough=="y" and sneezing=='y' and runnyNose=='y':
    print("Result: You are diagnosed with measles")
elif fever == 'y' and rashes== 'y' and bodyAche== 'y' and chills=="y":
    print("Result: You are diagnosed with chicken pox")
elif fever == 'y' and swollenGlands== 'y':
    print("Result: You are diagnosed with mumps")
elif fever == 'y' and headAche== 'y' and bodyAche=="y" and conjunctivitis=="y" and chills=="y"
    " and cough=="y" and runnyNose=="y" and soreThroat=="y":
    print("Result: You are diagnosed with flu")
```

Screenshots of the Outputs

```
Do you have a fever (y/n):- y
Do you have a rash (y/n):- n
Do you have a runny nose (y/n):-y
Do you have a conjunctivitis (y/n):-n
Do you have a cough (y/n):-y
Do you have a head ache (y/n):-y
Do you have body ache (y/n):-y
Do you have chills (y/n):-y
Do you have a sore throat (y/n):-y
Do you have sneezing (y/n):-y
Do you have swollen glands (y/n):-n
Result: You are diagnosed with measles
```

Result:- Implemented Medical Diagnosis system in Python

Signature of the Student

Arnav Kumar

Date: 07/02/22 Ex No: 3	Title of the Lab Constraint Satisfaction problem	Name: Arnav Kumar Registration Number: RA1911027010040 Section: N1 Lab Batch: 2 Day Order: 3
--	---	---

AIM: To build program to evaluate Cryptarithmic problem CROSS + ROAD = DANGER

Description of the Concept or Problem given :-

In this cryptarithmic problem, we got an expression where sum of 2 strings is equal to third string, our task is to allocate unique digits to every letter in such a way that the expression will get the arithmetically correct result.

Key Constraints for the problem:

1. Each letter or symbol represented only one and a unique digit (0-9)
2. When the digits replace letters or symbols, the resultant arithmetical operation must be correct.
3. There must be at most 10 unique letters in the problem
4. The letters at the beginning of each number should not correspond to 0

Manual Solution

- Starting from LHS, keep allocating the values to the new letters and give the resultant sum to the LHS result side (for the 1st time, if result length is 1 unit longer than the addition strings, pick the upper two digits ≥ 5)
- Keep allocating values to the letters and check if the sum is giving the expected output (prefer to give most occurring letters safest values accordingly)
- If the sum/ resultant number can't be allocated to the respective letter of result string, change the previous values accordingly, to get the desired result
- Stop once we reach the right end

Program Implementation [Coding]

```
#include <bits/stdc++.h>
using namespace std;

// vector stores 1 corresponding to index
// number which is already assigned
// to any char, otherwise stores 0
vector<int> use(10);

// structure to store char and its corresponding integer
struct node
{
```

```

    char c;
    int v;
};

// function check for correct solution
int check(node* nodeArr, const int count, string s1,
          string s2, string s3)
{
    int val1 = 0, val2 = 0, val3 = 0, m = 1, j, i;

    // calculate number corresponding to first string
    for (i = s1.length() - 1; i >= 0; i--)
    {
        char ch = s1[i];
        for (j = 0; j < count; j++)
            if (nodeArr[j].c == ch)
                break;

        val1 += m * nodeArr[j].v;
        m *= 10;
    }
    m = 1;

    // calculate number corresponding to second string
    for (i = s2.length() - 1; i >= 0; i--)
    {
        char ch = s2[i];
        for (j = 0; j < count; j++)
            if (nodeArr[j].c == ch)
                break;

        val2 += m * nodeArr[j].v;
        m *= 10;
    }
    m = 1;

    // calculate number corresponding to third string
    for (i = s3.length() - 1; i >= 0; i--)
    {
        char ch = s3[i];
        for (j = 0; j < count; j++)
            if (nodeArr[j].c == ch)
                break;

        val3 += m * nodeArr[j].v;
        m *= 10;
    }

    // sum of first two number equal to third return true
    if (val3 == (val1 + val2))
        return 1;

    // else return false
    return 0;
}

// Recursive function to check solution for all permutations
bool permutation(const int count, node* nodeArr, int n,
                 string s1, string s2, string s3)
{
    // Base case

```

```

    if (n == count - 1)
    {

        // check for all numbers not used yet
        for (int i = 0; i < 10; i++)
        {

            // if not used
            if (use[i] == 0)
            {

                // assign char at index n integer i
                nodeArr[n].v = i;

                // if solution found
                if (check(nodeArr, count, s1, s2, s3) == 1)
                {
                    cout << "\nSolution found: \n";
                    for (int j = 0; j < count; j++)
                        cout << nodeArr[j].c << " = "
                            << nodeArr[j].v << endl;
                    return true;
                }
            }
        }
        return false;
    }

    for (int i = 0; i < 10; i++)
    {

        // if ith integer not used yet
        if (use[i] == 0)
        {

            // assign char at index n integer i
            nodeArr[n].v = i;

            // mark it as not available for other char
            use[i] = 1;

            // call recursive function
            if (permutation(count, nodeArr, n + 1, s1, s2, s3))
                return true;

            // backtrack for all other possible solutions
            use[i] = 0;
        }
    }
    return false;
}

bool solveCryptographic(string s1, string s2,
                        string s3)
{
    // count to store number of unique char
    int count = 0;

    // Length of all three strings
    int l1 = s1.length();
    int l2 = s2.length();

```

```

    int l3 = s3.length();

    // vector to store frequency of each char
    vector<int> freq(26);

    for (int i = 0; i < l1; i++)
        ++freq[s1[i] - 'A'];

    for (int i = 0; i < l2; i++)
        ++freq[s2[i] - 'A'];

    for (int i = 0; i < l3; i++)
        ++freq[s3[i] - 'A'];

    // count number of unique char
    for (int i = 0; i < 26; i++)
        if (freq[i] > 0)
            count++;

    // solution not possible for count greater than 10
    if (count > 10)
    {
        cout << "Invalid strings";
        return 0;
    }

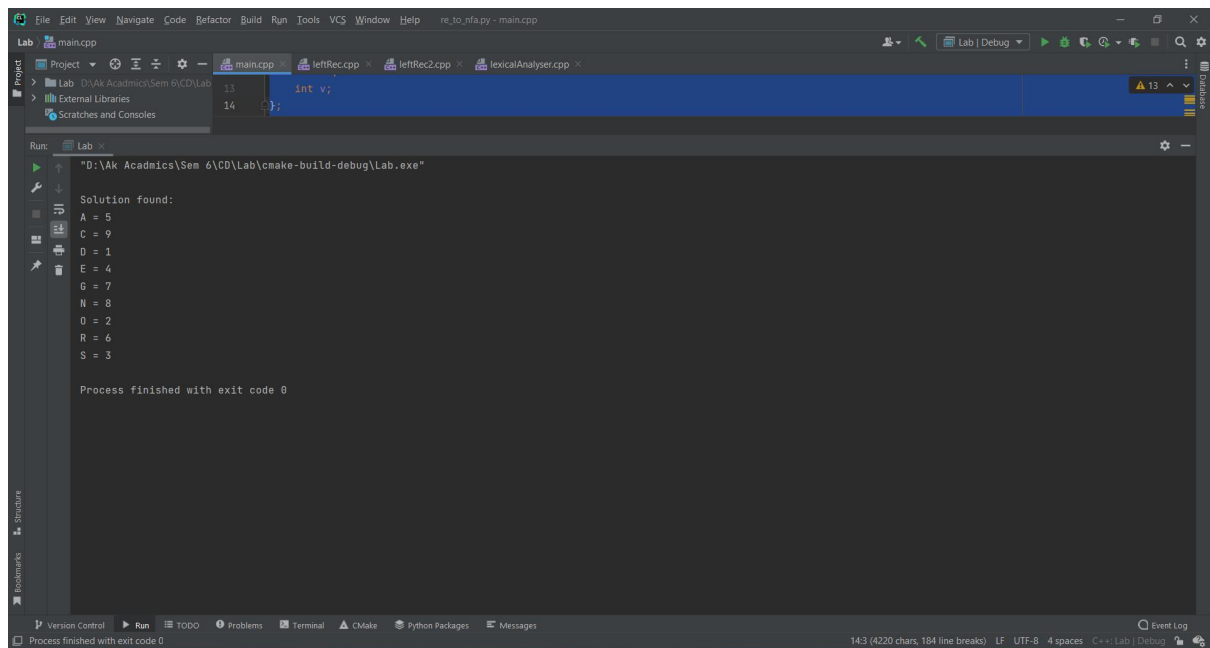
    // array of nodes
    node nodeArr[count];

    // store all unique char in nodeArr
    for (int i = 0, j = 0; i < 26; i++)
    {
        if (freq[i] > 0)
        {
            nodeArr[j].c = char(i + 'A');
            j++;
        }
    }
    return permutation(count, nodeArr, 0, s1, s2, s3);
}

// Driver function
int main()
{
    string s1 = "CROSS";
    string s2 = "ROADS";
    string s3 = "DANGER";

    if (solveCryptographic(s1, s2, s3) == false)
        cout << "No solution";
    return 0;
}

```



AIM: To implement the program for Map Colouring Problem.

Description of the Concept or Problem given :-

In this problem we are given an undirected graph, our task is to colour the graph in such a way that no two adjacent nodes have same colour, also we have to find minimum number of colours needed to colour all the nodes of the map.

Manual Solution

- Starting from the root node, colour the current node, and mark all the adjacent nodes as “not color1”
- Then go to all next nodes one by one and check if the colour1 is available:
 - If available colour it and continue
 - Else check for next colour (i.e. colour2, colour3....) and keep checking till we find an available colour, after that continue
 - After finding colour continue all next nodes once reached an end backtrack and continue for the previous node's next
- Once we cover all the next nodes of our root node we will print the output, as all the nodes are now coloured

Program Implementation [Coding]

```
# Adjacent Matrix
G = [[ 0, 1, 1, 0, 1, 0],
      [ 1, 0, 1, 1, 0, 1],
      [ 1, 1, 0, 1, 1, 0],
      [ 0, 1, 1, 0, 0, 1],
      [ 1, 0, 1, 0, 0, 1],
      [ 0, 1, 0, 1, 1, 0]]

# initiate the name of node.
node = "ABCDEF"
t_={}
for i in range(len(G)):
    t_[node[i]] = i

# count degree of all node.
degree = []
for i in range(len(G)):
    degree.append(sum(G[i]))

# initiate the possible color
colorDict = {}
for i in range(len(G)):
    colorDict[node[i]]=["Blue", "Red", "Yellow", "Green"]

# sort the node depends on the degree
sortedNode=[]
```



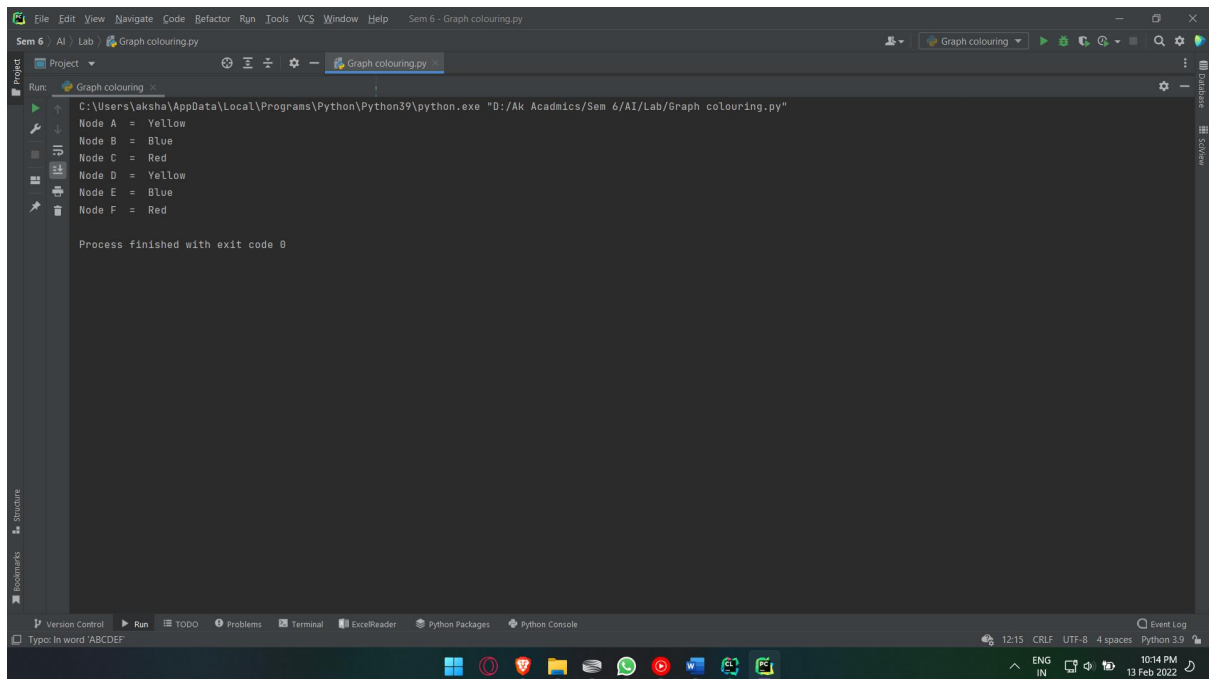
```
indeks = []

# use selection sort
for i in range(len(degree)):
    _max = 0
    j = 0
    for j in range(len(degree)):
        if j not in indeks:
            if degree[j] > _max:
                _max = degree[j]
                idx = j
    indeks.append(idx)
    sortedNode.append(node[idx])

# The main process
theSolution={}
for n in sortedNode:
    setTheColor = colorDict[n]
    theSolution[n] = setTheColor[0]
    adjacentNode = G[t_[n]]
    for j in range(len(adjacentNode)):
        if adjacentNode[j]==1 and (setTheColor[0] in colorDict[node[j]]):
            colorDict[node[j]].remove(setTheColor[0])

# Print the solution
for t,w in sorted(theSolution.items()):
    print("Node",t," = ",w)
```

Screenshots of the Outputs



Signature of the Student

Arnav Kumar

Date: Ex No: 4.1	Title of the Lab Implementation of BFS for finding nth level friend	Name: Arnav Kumar Registration Number: RA1911027010040 Section: N1 Lab Batch: 2 Day Order: 3
--------------------------------------	---	--

AIM:

To find nth-Level friends for a random person in the friend network using BFS.

Description of the Concept or Problem given:

Breadth First Search, being a level order traversal, would be quite efficient over Depth First Search for many business based insights, like the following: 1. Finding all the friends of all the people in the network 2. Finding all the mutual friends for a node in the network 3. Finding the shortest path between two people in the network 4. Finding the nth level friends for a person in the network.

Manual Solution:

We can find the path between two people by running a BFS algorithm, starting the traversal from one person in level order until we reach the other person or in a much optimised way we can run a bi-directional BFS from both the nodes until our search meet at some point and hence we conclude the path, whereas DFS being a depth wise traversal may run through many unnecessary sub-trees, unknowing of the fact that the friend could be on the first level itself. Moreover, in order to find friends at nth level, using BFS this could be done in much less time, as this traversal keeps account of all the nodes in each level.

Program Implementation [Coding]:

```
from collections import deque

graph = {}

queries = []

def accept_values():

    vertex_edge = [int(i) for i in input().strip().split(" ")]

    for i in range(vertex_edge[1]):
```

```

edge = [int(i) for i in input().strip().split(" ")]
try:
    graph[edge[0]].append(edge[1])
except:
    graph[edge[0]] = [edge[1]]
try:
    graph[edge[1]].append(edge[0])
except:
    graph[edge[1]] = [edge[0]]
number_of_queries = int(input())
for i in range(number_of_queries):
    queries.append([int(i) for i in input().strip().split(" ")])
for query in queries:
    bfs(query)

def bfs(query):
    counter = 0
    q = deque()
    q.append(query[0])
    visited = {query[0] : True}
    distance = {query[0] : 0}
    while q:
        popped = q.popleft()
        for neighbour in graph[popped]:
            if neighbour not in visited:
                visited[neighbour] = True

            if distance[popped] + 1 > query[1]:
                for key in distance:

```

```
        if distance[key] == query[1]:
            counter +=1
        print(counter)
        return
    else:
        distance[neighbour] = distance[popped] + 1
        q.append(neighbour)
    print(counter)

accept_values()
```

Screenshots of the Outputs:

```
0 61 21 32 43 64 5 -1 -11 2
4 6
```

Signature of the Student

Arnav Kumar

Date: Ex No: 4.2	Title of the Lab Implementation of Topological Sort using DFS	Name: Arnav Kumar Registration Number: RA1911027010040 Section: N1 Lab Batch: 2 Day Order: 3
--------------------------------------	---	--

AIM:

To find the Topological Sort of a graph using DFS.

Description of the Concept or Problem given:

Topological sort is an algorithm that takes a directed acyclic graph and returns the sequence of nodes where every node will appear before other nodes that it points to. Topological sorting for graphs is not applicable if the graph is not a Directed Acyclic Graph. In it, directed acyclic graph is the operation of arranging the nodes in the order in such a way that if there exists an edge (i,j) , i precedes j in the lists. A topological sort basically gives a sequence in which we should perform the job and helps us to check whether the graph consists of the cycle or not. Every graph can have more than one topological sorting possible.

Manual Solution:

The algorithm of the topological sort goes like this:

1. Identify the node that has no in-degree(no incoming edges) and select that node as the source node of the graph.
2. Delete the source node with zero in-degree and also delete all its outgoing edges from the graph. Insert the deleted vertex in the result array.
3. Update the in-degree of the adjacent nodes after deleting the outgoing edges.
4. Repeat step 1 to step 3 until the graph is empty.

The resulting array at the end of the process is called the topological ordering of the directed acyclic graph. If due to some reason, there are some nodes left but they have the incoming edges, that means that the graph is not an acyclic graph and topological ordering does not exist.

Program Implementation [Coding]:

```
from collections import defaultdict
```

```
class Graph:
```

```
    def __init__(self,n):
```

```
        self.graph = defaultdict(list)
```

```
        self.N = n
```

```
    def addEdge(self,m,n):
```

```
        self.graph[m].append(n)
```

```
    def sortUtil(self,n,visited,stack):
```

```
        visited[n] = True
```

```
        for element in self.graph[n]:
```

```
            if visited[element] == False:
```

```
                self.sortUtil(element,visited,stack)
```

```
        stack.insert(0,n)
```

```
    def topologicalSort(self):
```

```
        visited = [False]*self.N
```

```
        stack =[]
```

```
        for element in range(self.N):
```

```
            if visited[element] == False:
```

```
                self.sortUtil(element,visited,stack)
```

```
        print(stack)
```

```
graph = Graph(5)
```

```
graph.addEdge(0,1);
```

```
graph.addEdge(0,3);
```

```
graph.addEdge(1,2);  
graph.addEdge(2,3);  
graph.addEdge(2,4);  
graph.addEdge(3,4);  
  
print("The Topological Sort Of The Graph Is: ")  
  
graph.topologicalSort()
```

Screenshots of the Outputs:

```
The Topological Sort Of The Graph Is:  
[0, 1, 2, 3, 4]
```

Signature of the Student

Arnav Kumar

Date: Ex No: 5.1	Title of the Lab Implementation of Best First Algorithm	Name: Arnav Kumar Registration Number: RA1911027010040 Section: N1 Lab Batch: 2 Day Order: 3
--------------------------------------	---	--

AIM:

To implement the Best First Algorithm.

Description of the Concept or Problem given:

In BFS and DFS, when we are at a node, we can consider any of the adjacent as the next node. So both BFS and DFS blindly explore paths without considering any cost function. The idea of Best First Search is to use an evaluation function to decide which adjacent is most promising and then explore.

Manual Solution:

1. Define a list, OPEN, consisting solely of a single node, the start node, s.
2. IF the list is empty, return failure.
3. Remove from the list the node n with the best score (the node where f is the minimum), and move it to a list, CLOSED.
4. Expand node n.
5. IF any successor to n is the goal node, return success and the solution (by tracing the path from the goal node to s).
6. FOR each successor node: 1. apply the evaluation function, f, to the node. 2. IF the node has not been in either list, add it to OPEN.
7. Looping structure by sending the algorithm back to the second step.

Program Implementation [Coding]:

```
from queue import PriorityQueue
```

```
v = 14
```

```
graph = [[] for i in range(v)]
```



```
def best_first_search(source, target, n):
```

```
    visited = [0] * n
```

```
    visited[0] = True
```

```
    pq = PriorityQueue()
```

```
    pq.put((0, source))
```

```
    while pq.empty() == False:
```

```
        u = pq.get()[1]
```

```
        print(u, end=" ")
```

```
        if u == target:
```

```
            break
```

```
        for v, c in graph[u]:
```

```
            if visited[v] == False:
```

```
                visited[v] = True
```

```
                pq.put((c, v))
```

```
    print()
```

```
def addedge(x, y, cost):
```

```
    graph[x].append((y, cost))
```

```
    graph[y].append((x, cost))
```

```
adddedge(0, 1, 3)
```

```
adddedge(0, 2, 6)
```

```
adddedge(0, 3, 5)
```

```
adddedge(1, 4, 9)
```

```
adddedge(1, 5, 8)
```

```
adddedge(2, 6, 12)
```

```
adddedge(2, 7, 14)
```

```
adddedge(3, 8, 7)
```

```
adddedge(8, 9, 5)
```

```
addedge(8, 10, 6)
```

```
addedge(9, 11, 1)
```

```
addedge(9, 12, 10)
```

```
addedge(9, 13, 2)
```

```
source = 0
```

```
target = 9
```

```
best_first_search(source, target, v)
```

Screenshots of the Outputs:

0 1 3 2 8 9

Signature of the Student

Arnav Kumar

Date: Ex No: 5.2	Title of the Lab Implementation of A* Algorithm	Name:Arnav Kumar Registration Number: RA1911027010040 Section: N1 Lab Batch: 2 Day Order: 3
--------------------------------------	---	---

AIM:

To implement the A* Algorithm.

Description of the Concept or Problem given:

A* is an informed search algorithm, or a best-first search, meaning that it is formulated in terms of weighted graphs: starting from a specific starting node of a graph, it aims to find a path to the given goal node having the smallest cost (least distance travelled, shortest time, etc.). It does this by maintaining a tree of paths originating at the start node and extending those paths one edge at a time until its termination criterion is satisfied.

Manual Solution:

1. Define a list, OPEN, consisting solely of a single node, the start node, s.
2. IF the list is empty, return failure.
3. Remove from the list the node n with the best score (the node where f is the minimum), and move it to a list, CLOSED.
4. Expand node n.
5. IF any successor to n is the goal node, return success and the solution (by tracing the path from the goal node to s).
6. FOR each successor node: 1.apply the evaluation function, f, to the node. 2. IF the node has not been in either list, add it to OPEN.
7. Looping structure by sending the algorithm back to the second step.

Program Implementation [Coding]:

def aStarAlgo(start_node, stop_node):

 open_set = set(start_node)

 closed_set = set()

 g = {}

```
parents = {}
g[start_node] = 0
parents[start_node] = start_node

while len(open_set) > 0:
    n = None
    for v in open_set:
        if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):
            n = v

    if n == stop_node or Graph_nodes[n] == None:
        pass
    else:
        for (m, weight) in get_neighbors(n):
            if m not in open_set and m not in closed_set:
                open_set.add(m)
                parents[m] = n
                g[m] = g[n] + weight
            else:
                if g[m] > g[n] + weight:
                    g[m] = g[n] + weight
                    parents[m] = n

                if m in closed_set:
                    closed_set.remove(m)
                    open_set.add(m)

    if n == None:
        print('Path does not exist!')
        return None
```

```
if n == stop_node:
    path = []
    while parents[n] != n:
        path.append(n)
        n = parents[n]
    path.append(start_node)
    path.reverse()
    print('Path found: {}'.format(path))
    return path
```

```
open_set.remove(n)
closed_set.add(n)
print('Path does not exist!')
return None
```

```
def get_neighbors(v):
    if v in Graph_nodes:
        return Graph_nodes[v]
    else:
        return None
```

```
def heuristic(n):
    H_dist = {
        'A': 11,
        'B': 6,
        'C': 99,
        'D': 1,
        'E': 7,
```

```
'G': 0,  
}  
return H_dist[n]
```

```
Graph_nodes = {  
    'A': [('B', 2), ('E', 3)],  
    'B': [('C', 1), ('G', 9)],  
    'C': None,  
    'E': [('D', 6)],  
    'D': [('G', 1)],  
}  
aStarAlgo('A', 'G')
```

Screenshots of the Outputs:

Path found: ['A', 'E', 'D', 'G']

Signature of the Student

[ARNAV KUMAR]

Date: Ex No: 6.1	Title of the Lab Implementation of Fuzzy Logic	Name: Arnav Kumar Registration Number: RA1911027010040 Section: N1 Lab Batch: 2 Day Order: 3
--------------------------------------	--	--

AIM:

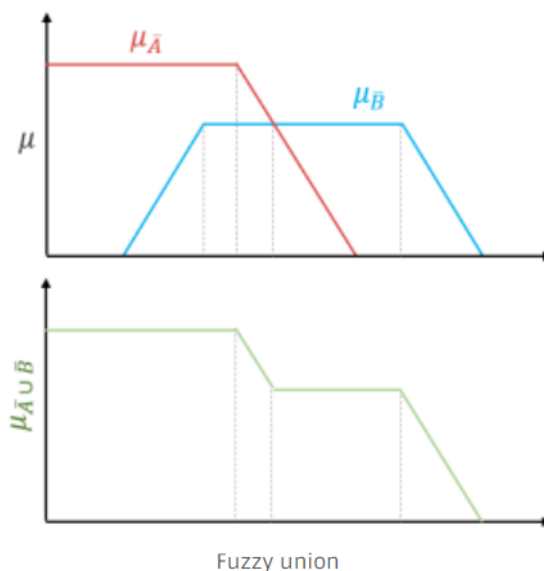
To implement Fuzzy Logic.

Description of the Concept or Problem given:

In case of union of crisp sets, we simply have to select repeated elements only once. In case of fuzzy sets, when there are common elements in both the fuzzy sets, we should select the element with **maximum membership value**.

The **union** of two fuzzy sets \underline{A} and \underline{B} is a fuzzy set \underline{C} , written as $\underline{C} = \underline{A} \cup \underline{B}$

Graphically we can represent union operation as follow. Red and Blue membership functions represents the fuzzy value for elements in set A and B, respectively. Wherever these fuzzy functions overlaps, we have to consider the point with maximum membership value.



Manual Solution:

1. Import matplotlib from the python library.
2. Initialize an array of numbers in two 2D arrays a and b which would act as sets of numbers.
3. Define two functions checkset() and set_and_mf_of_set() which will check whether the elements in the array contains more than one membership value or not, if not than return the set.
4. Initialize another set of numbers in two 2D array x and y with slightly different values and store them in a different variable and check whether they contain they contain membership value or not.

5. Find the union of arrays x and y using the fuzzy logic code of union for finding the union between the arrays and print them.
6. Plot the corresponding graphs of `set_and_mf_of_set(x)` and `set_and_mf_of_set(y)`.
7. The middle graph represents the union between the two sets of arrays.

Program Implementation [Coding]:

```
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (20,7)
a=[[1,0.0],[2,0.0],[3,0.2],[4,0.8],[5,1],[6,0.2],[7,0.0],[8,0.0]]
b=[[1,0.0],[2,0.1],[3,0.3],[4,1],[5,0.5],[6,0.1],[7,0.0],[8,0.0]]

print()
def checkset(a):
    r=0
    for i in range(len(a)):
        for j in range(i+1,len(a)):
            if a[i][0]==a[j][0]:
                r+=1
                break
    return r

def set_and_mf_of_set(a):
    p = checkset(a)
    if p == 0:
        set1=[]
        mfset=[]
        for i in range(len(a)):
            set1.append(a[i][0])
            mfset.append(a[i][1])
        return set1,mfset
    else:
        print("In Set at one element more than one MemberShip Value")

a = set_and_mf_of_set(a)
x=[[1,0.0],[2,0.0],[3,0.2],[4,0.8],[5,1],[6,0.2],[7,0.0],[8,0.0]]
y=[[1,0.0],[2,0.1],[3,0.3],[4,1],[5,0.8],[6,0.1],[7,0.0],[8,0]]
p,mfp = set_and_mf_of_set(x)
q,mfq = set_and_mf_of_set(y)

def Union(a,b):
    p = checkset(a)
    q = checkset(b)
    if p == 0 & q==0:
        union=[]
        if len(a) < len(b) :
            temp = a
            a = b
            b = temp
        for i in range (len(a)):
            for j in range(0,len(b)):
                if a[i][0] == b[j][0]:
                    if a[i][1] > b[j][1]:
                        union.append(a[i])
```



```

        else:
            union.append(b[j])
    else:
        if len(union)==0:
            union.append(a[i])
        else:
            p=0
            for k in range(0,len(union)):
                if union[k][0]==a[i][0]:
                    p+=1
                    if p==0:
                        union.append(a[i])

    return union
else:
    print("In Set at one element more than one MemberShip Value")

```

```

z=Union(y,x)
r,mfr = set_and_mf_of_set(z)

```

```

print(x)
print(y)
print("Union is",z)

```

```

plt.subplot(131)
plt.plot(p,mfp)
plt.plot(q,mfq)

```

```

plt.subplot(132)
plt.plot(r,mfr)

```

```

plt.subplot(133)

```

```

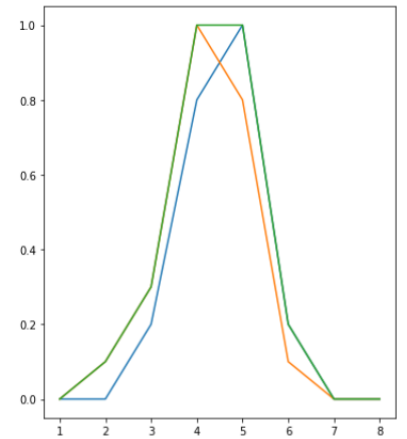
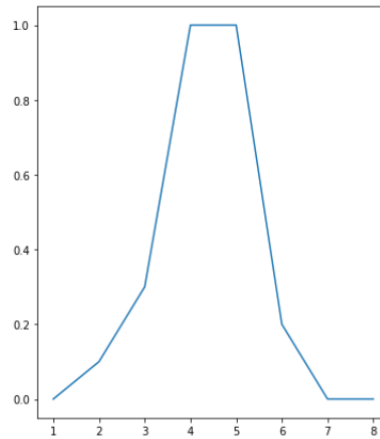
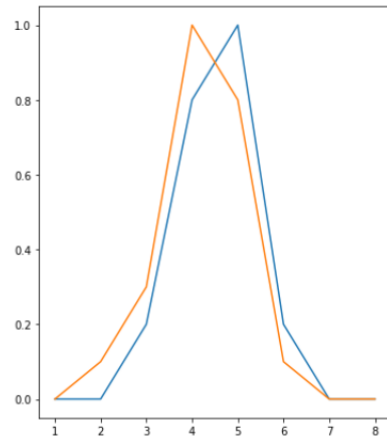
plt.plot(p,mfp)
plt.plot(q,mfq)
plt.plot(r,mfr)

```

Screenshots of the Outputs:

```
[[1, 0.0], [2, 0.0], [3, 0.2], [4, 0.8], [5, 1], [6, 0.2], [7, 0.0], [8, 0.0]]  
[[1, 0.0], [2, 0.1], [3, 0.3], [4, 1], [5, 0.8], [6, 0.1], [7, 0.0], [8, 0]]  
Union is [[1, 0.0], [2, 0.1], [3, 0.3], [4, 1], [5, 1], [6, 0.2], [7, 0.0], [8, 0.0]]
```

Out[1]: [<matplotlib.lines.Line2D at 0x187c7729910>]



Signature of the Student

Arnav Kumar

Date: Ex No: 6.2	Title of the Lab Implementation of Dempster Shafer Theory	Name: Arnav Kumar Registration Number: RA1911027010040 Section: N1 Lab Batch: 1 Day Order: 3
--	--	--

AIM:

To implement Dempster Shafer Theory.

Description of the Concept or Problem given:

Dempster-Shafer Theory is a mathematical theory of evidence, offers an alternative to traditional probabilistic theory for the mathematical representation of uncertainty. The significant innovation of this framework is that it allows for the allocation of a probability mass to sets or intervals as opposed to mutually exclusive singletons. D-S is a potentially valuable tool for the evaluation of risk and reliability in engineering applications when it is not possible to obtain a precise measurement from experiments, or when knowledge is obtained from expert elicitation. An important aspect of D-S theory is the combination of evidence obtained from multiple sources and the modelling of conflict between them.

Manual Solution:

1. **Mass functions** denoted by m : A mass function is in many respects the most fundamental belief representation and all other representations can be easily obtained from a mass function. Formally, a mass function is a mapping assigning a mass value to each hypothesis belongs to of the frame of discernment is the amount of belief strictly committed to hypothesis.
2. **Belief functions** denoted by bel : The total amount of belief committed to a hypothesis, including all subsets is denoted by $bel(A)$. The function is called a belief function. It can be directly computed from a mass function. A belief function is sometimes interpreted as de

fining a “lower bound” for an unknown probability function.

3. **Plausibility functions** denoted by pl: The plausibility is the amount of belief not strictly committed to the complement. It therefore expresses how plausible a hypothesis is, i.e., how much belief mass potentially supports. Whereas can be viewed as a lower bound for an unknown probability function. Under a lower and upper probability interpretation, the plausibility can be viewed as an “upper bound”.
4. **Commonality functions** denoted by q: The commonality states how much mass in total is committed to and all of the supersets with as its subset. The commonality therefore expresses how much mass potentially supports the entire set.
5. Test the various belief functions from the pyds MassFunction.
6. Print the MassFunction frame of discernment and the powerset.

Program Implementation [Coding]:

```
from Pyds import *
m1 = MassFunction({'a':0.4, 'b':0.2, 'ab':0.1, 'abc':0.3})
m2 = MassFunction({'b':0.5, 'c':0.2, 'ac':0.3, 'a':0.0})
print("m1:",m1)
print("m1: bpa of {'a','b'}=", m1['ab'])
print("m1: belief of {'a','b'}=", m1.bel('ab'))
print("m1: plausibility of {'a','b'}=", m1.pl('ab'))
print("m1: commonality of {'a','b'}=", m1.q('ab'))
print("m2:",m2)
print("m2: bpa of {'b'}=", m2['b'])
print("m2: belief of {'b'}=", m2.bel('b'))
print("m2: plausibility of {'b'}=", m2.pl('b'))
print("m2: commonality of {'b'}=", m2.q('b'))
m1_1 = MassFunction([(('a',), 0.4), (('b',), 0.2), (('a', 'b'), 0.1), (('a', 'b', 'c'), 0.3)])
if (m1_1==m1):
    print("m1_1 Equal to m1")
m1_2 = MassFunction([(('a', 0.4), ('b', 0.2), ('ab', 0.1), ('abc', 0.3)])]
if (m1_2==m1):
    print("m1_2 Equal to m1")
print("frame of discernment", m1.frame())
print("powerset of frame", list(m1.all()))
```

Screenshots of the Outputs:

```
m1: {'a':0.4; {'a', 'b', 'c':0.3; {'b':0.2; {'a', 'b':0.1}
m1: bpa of {'a','b'}= 0.1
m1: belief of {'a','b'}= 0.7000000000000001
m1: plausibility of {'a','b'}= 1.0
m1: commonality of {'a','b'}= 0.4
m2: {'b':0.5; {'a', 'c':0.3; {'c':0.2; {'a':0.0}
m2: bpa of {'b'}= 0.5
m2: belief of {'b'}= 0.5
m2: plausibility of {'b'}= 0.5
m2: commonality of {'b'}= 0.5
m1_1 Equal to m1
m1_2 Equal to m1
frame of discernment frozenset({'a', 'c', 'b'})
powerset of frame [frozenset(), frozenset({'a'}), frozenset({'c'}), frozenset({'b'}), frozenset({'a', 'c'}), frozenset({'a', 'b'}), frozenset({'b', 'c'}), frozenset({'a', 'b', 'c'})]
```

Signature of the Student

Arnav Kumar

Date: Ex No: 7.1	Title of the Lab Implementation of Unification in SWI Prolog	Name: Arnav Kumar Registration Number: RA1911027010040 Section: N1 Lab Batch: 2 Day Order: 3
--	---	--

AIM:

To implement Unification in SWI Prolog.

Description of the Concept or Problem given:

Prolog uses the unification technique, and it is a very general form of matching technique. In unification, one or more variables being given value to make the two call terms identical. This process is called binding the variables to values. For example, Prolog can unify the terms `cat(A)`, and `cat(mary)` by binding variable `A` to atom `mary` that means we are giving the value `mary` to variable `A`.

Manual Solution:

1. If `Y1` or `Y2` is a variable or constant, then:
 - a) If `Y1` , or `Y2` are identical, then return `NIL`.
 - b) Else if `Y1` is a variable,
 - a. then if `Y1`, occurs in `Y2`, then return `FAILURE`
 - b. Else return `{{ Y2,/Y1 }}`.
 - c) Else if `Y2` is a variable,
 - a. If `Y2` occurs in `Y1`, then return `FAILURE`,
 - b. Else return `{(Y1/Y2)}`.
 - d) Else return `FAILURE`.
2. If the initial Predicate symbol in `Y1`, and `Y2` are not same, then return `FAILURE`.
3. If `Y1` and `Y2` have a different number of arguments, then return `FAILURE`.
4. Set Substitution `set(SUBST)` to `NIL`.
5. For `i=1` to the number of elements in `Y1`.
 - a) Call Unify function with the `ith` element of `Y1`, and `ith` element of `Y2`, and put the result into `S`.
 - b) If `S=failure` then returns `Failure`
 - c) If `S != NIL` then do,
 - a. Apply `S` to the remainder of both `L1` and `L2`.
 - b. `SUBST = APPEND(S, SUBST)`.
6. Return `SUBST`.

Screenshots of the Outputs:

18CSC305J Artificial Intelligence Lab

```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.2)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- employees(_name(sid),_).
true.

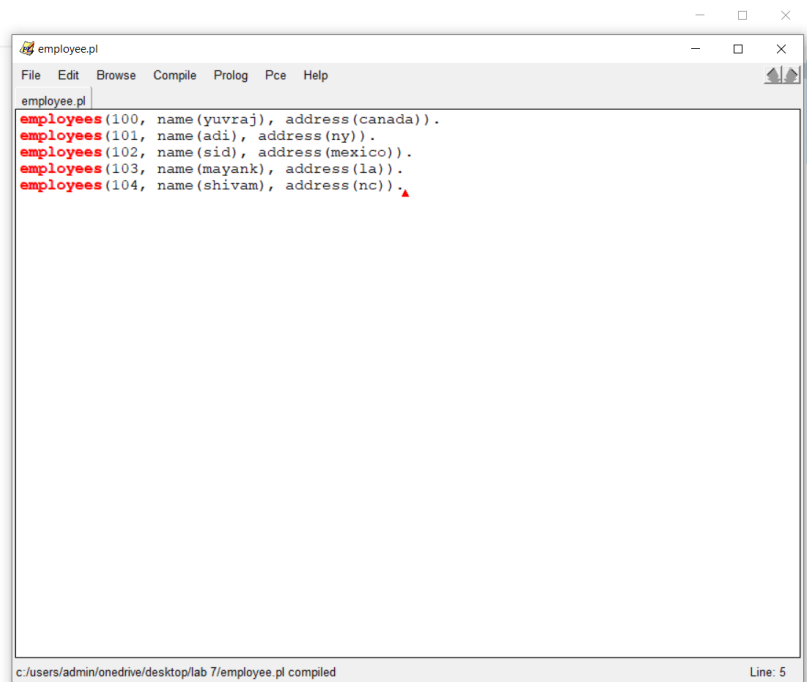
?- employees(X,name(sid),Y).
X = 102
Y = address(mexico).

?- employees(101,name(B),C).
B = adi
C = address(ny)
Unknown action: [] (h for help)
Action? .

?- employees(A,name(B),C).
A = 100
B = yuvraj
C = address(canada) .

?- employees(101,Name,Address).
Name = name(adi),
Address = address(ny).

?- employees(ID,Name,Address).
ID = 100,
Name = name(yuvraj),
Address = address(canada)
```



```
employee.pl
File Edit Browse Compile Prolog Pce Help

employee.pl
employees(100, name(yuvraj), address(canada)).
employees(101, name(adi), address(ny)).
employees(102, name(sid), address(mexico)).
employees(103, name(mayank), address(la)).
employees(104, name(shivam), address(nc)).
```

c:/users/admin/onedrive/desktop/lab 7/employee.pl compiled Line: 5

Signature of the Student

Arnav Kumar

Date: Ex No: 7.2	Title of the Lab Implementation of Resolution in SWI Prolog	Name: Arnav Kumar Registration Number: RA1911027010040 Section: N1 Lab Batch: 2 Day Order: 3
--	--	--

AIM:

To implement Resolution in SWI Prolog.

Description of the Concept or Problem given:

In simple words resolution is inference mechanism. Let's say we have clauses $m :- b.$ and $t :- p, m, z.$ So from that we can infer $t :- p, b, z.$ - that is called resolution. Means, when you resolve two clauses you get one new clause. Another easy example, we have two sentences (1) All women like shopping. (2) Olivia is a woman. Now we ask query 'Who likes shopping'. So, by resolving above sentences we can have one new sentence Olivia likes shopping.

Manual Solution:

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove (proof by contradiction)
4. Draw resolution graph (unification).

Screenshots of the Outputs:

The screenshot shows a Prolog environment with two windows. The left window, titled 'hobbies.pl', contains the following code:

```

person(ali,20).
person(bob,20).
person(cal,25).

hobby(ali,skiing).
hobby(bob,skiing).
hobby(cal,skiing).

friends(P1,P2):-
    hobby(P1,H),
    hobby(P2,H),
    P1\=P2,
    person(P1,A1),
    person(P2,A2),
    AD is abs(A2-A1),
    AD<=3.

```

The right window, titled 'SWI-Prolog (AMD64, Multi-threaded, version 8.4.2)', shows the execution trace for the query `?- friends(ali,bob).`. The trace indicates that the query is successful (true).

```

Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic), or ?- apropos(Word).

?- friends(ali,bob).
true.

?- trace.
true.

[trace] ?- friends(ali,bob).
Call: (10) friends(ali,bob) ? creep
Call: (11) hobby(ali,skiing) ? creep
Exit: (11) hobby(ali,skiing) ? creep
Call: (11) hobby(bob,skiing) ? creep
Exit: (11) hobby(bob,skiing) ? creep
Call: (11) ali\=bob ? creep
Exit: (11) ali\=bob ? creep
Call: (11) person(ali,20) ? creep
Exit: (11) person(ali,20) ? creep
Call: (11) person(bob,20) ? creep
Exit: (11) person(bob,20) ? creep
Call: (11) 26128 is abs(20-20) ? creep
Exit: (11) 0 is abs(20-20) ? creep
Call: (11) 0<=3 ? creep
Exit: (11) 0<=3 ? creep
Exit: (10) friends(ali,bob) ? creep
true.

[trace] ?- friends(ali,cal).
Call: (10) friends(ali,cal) ? creep
Call: (11) hobby(ali,skiing) ? creep
Exit: (11) hobby(ali,skiing) ? creep
Call: (11) hobby(cal,skiing) ? creep
Exit: (11) hobby(cal,skiing) ? creep
Call: (11) ali\=cal ? creep
Exit: (11) ali\=cal ? creep
Call: (11) person(ali,20) ? creep
Exit: (11) person(ali,20) ? creep
Call: (11) person(cal,25) ? creep
Exit: (11) person(cal,25) ? creep
Call: (11) 40092 is abs(25-20) ? creep
Exit: (11) 5 is abs(25-20) ? creep
Call: (11) 5<=3 ? creep
Exit: (11) 5<=3 ? creep
Fail: (11) friends(ali,cal) ? creep
false.

[trace] ?-

```

Signature of the Student

Arnav Kumar

Date: 7/4/2022 Ex No:8	Title of the Lab IMPLEMENTATION OF MACHINE LEARNING ALGORITHMS FOR AN APPLICATION	Name: Arnav Kumar Registration Number: RA1911027010040 Section: N1 Lab Batch: 2 Day Order: 3
---	--	--

AIM:- To study about supervised machine learning algorithms on a selected dataset and to Summarize the Performance measures for each algorithm.

For this Experiment, we have chosen mushrooms dataset where we will be applying Supervised Learning Algorithms for our Machine Learning model.

About Dataset:-

Data columns (total 23 columns):

#	Column	Non-Null Count	Dtype
---	-----	-----	----
0	class	8124 non-null	object
1	cap-shape	8124 non-null	object
2	cap-surface	8124 non-null	object
3	cap-color	8124 non-null	object
4	bruises	8124 non-null	object
5	odor	8124 non-null	object
6	gill-attachment	8124 non-null	object
7	gill-spacing	8124 non-null	object
8	gill-size	8124 non-null	object
9	gill-color	8124 non-null	object
10	stalk-shape	8124 non-null	object
11	stalk-root	8124 non-null	object
12	stalk-surface-above-ring	8124 non-null	object
13	stalk-surface-below-ring	8124 non-null	object
14	stalk-color-above-ring	8124 non-null	object
15	stalk-color-below-ring	8124 non-null	object
16	veil-type	8124 non-null	object
17	veil-color	8124 non-null	object
18	ring-number	8124 non-null	object
19	ring-type	8124 non-null	object
20	spore-print-color	8124 non-null	object
21	population	8124 non-null	object
22	habitat	8124 non-null	object

Algorithm and implementation:-

1.Support Vector Machine Algorithm

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

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

Implementation:-

from sklearn.svm we import SVC and we set parameter kernel='rbf', random_state=0, gamma=.10, C=1.0 and then fit the X_train and y_train. And find the accuracy of both test and train data

```
from sklearn.svm import SVC
Algorithm.append('Support Vector Classifier')
svm = SVC(kernel='rbf', random_state=0, gamma=.10, C=1.0)
svm.fit(X_train, y_train)
tr_sc1=svm.score(X_train, y_train)
ts_sc1=svm.score(X_test, y_test)
print('The accuracy of the SVM classifier on training data is {:.2f}'.format(tr_sc1))
print('The accuracy of the SVM classifier on test data is {:.2f}'.format(ts_sc1))
```

2. K-Nearest Neighbor(KNN)

o K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

o K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

o K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

Implementation:-

from sklearn.neighbors we import KNeighborsClassifier

and we set parameter n_neighbors = 7, p = 2, metric='minkowski and then fit the X_train and y_train. And find the accuracy of both test and train data

```
from sklearn.neighbors import KNeighborsClassifier
Algorithm.append('K-Nearest Neighbors')
knn = KNeighborsClassifier(n_neighbors = 7, p = 2, metric='minkowski')
```

```
knn.fit(X_train, y_train)
tr_sc2=knn.score(X_train, y_train)
ts_sc2=knn.score(X_test, y_test)
print('The accuracy of the Knn classifier on training data is {:.2f}'.format(tr_sc2))
print('The accuracy of the Knn classifier on test data is {:.2f}'.format(ts_sc2))
```

3. XGBoost

XGBoost stands for Extreme Gradient Boosting, which was proposed by the researchers at the University of Washington. It is a library written in C++ which optimizes the training for Gradient Boosting.

XGBoost is an implementation of Gradient Boosted decision trees. XGBoost models majorly dominate in many Kaggle Competitions.

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

Implementation:-

From xgboost we use XGBClassifier() to fit the X_train and y_train. And find the accuracy of both test and train data

```
import xgboost as xgb
Algorithm.append('XGBoost')
xgb_clf = xgb.XGBClassifier()
xgb_clf = xgb_clf.fit(X_train, y_train)
tr_sc3=xgb_clf.score(X_train, y_train)
ts_sc3=xgb_clf.score(X_test, y_test)
print('The accuracy of the XGBoost classifier on training data is {:.2f}'.format(tr_sc3))
print('The accuracy of the XGBoost classifier on test data is {:.2f}'.format(ts_sc3))
```

4. Decision Tree

- o Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- o In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- o The decisions or the test are performed on the basis of features of the given dataset.

Implementation:-

from sklearn we import tree

and we set parameter criterion='gini'

and then fit the X_train and y_train. And find the accuracy of both test and train data

```
from sklearn import tree
Algorithm.append('Decision Tree')
decision_tree = tree.DecisionTreeClassifier(criterion='gini')
decision_tree.fit(X_train, y_train)
tr_sc4=decision_tree.score(X_train, y_train)
ts_sc4=decision_tree.score(X_test, y_test)
print('The accuracy of the Decision Tree classifier on training data is {:.2f}'.format(tr_sc4))
print('The accuracy of the Decision Tree classifier on test data is {:.2f}'.format(ts_sc4))
```

5. Random Forest

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.

Implementation:-

from sklearn.ensemble we import RandomForestClassifier

and then fit the X_train and y_train. And find the accuracy of both test and train data

```
from sklearn.ensemble import RandomForestClassifier
Algorithm.append('Random Forest')
random_forest = RandomForestClassifier()
random_forest.fit(X_train, y_train)
tr_sc5=random_forest.score(X_train, y_train)
ts_sc5=random_forest.score(X_test, y_test)
print('The accuracy of the Random Forest classifier on training data is {:.2f}'.format(tr_sc5))
print('The accuracy of the Random Forest classifier on test data is {:.2f}'.format(ts_sc5))
```

6. Perceptron model

In Machine Learning and Artificial Intelligence, Perceptron is the most commonly used term for all folks. It is the primary step to learn Machine Learning and Deep Learning technologies, which consists of a set of weights, input values or scores, and a threshold. Perceptron is a building block of an Artificial Neural Network.

Implementation:-

from sklearn.linear_model we import Perceptron

and we set parameter max_iter = 40, eta0 = 0.1, random_state = 0

and then fit the X_train and y_train. And find the accuracy of both test and train data

```
from sklearn.linear_model import Perceptron
Algorithm.append('Perceptron model')
ppn = Perceptron(max_iter = 40, eta0 = 0.1, random_state = 0)
ppn.fit(X_train, y_train)
tr_sc6=ppn.score(X_train, y_train)
ts_sc6=ppn.score(X_test, y_test)
print('The accuracy of the Perceptron model classifier on training data is {:.2f}'.format(tr_sc6))
print('The accuracy of the Perceptron model classifier on test data is {:.2f}'.format(ts_sc6))
```

7. Logistic Regression

- o Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- o Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1.**

Implementation:-

from sklearn.linear_model we import LogisticRegression and we set parameter C = 1000.0, random_state = 0 and then fit the X_train and y_train. And find the accuracy of both test and train data

```
from sklearn.linear_model import LogisticRegression
Algorithm.append('Logistic Regression')
lr = LogisticRegression(C = 1000.0, random_state = 0 )
lr.fit(X_train, y_train)
tr_sc7=lr.score(X_train, y_train)
ts_sc7=lr.score(X_test, y_test)
print('The accuracy of the Logistic Regression classifier on training data is {:.2f}'.format(tr_sc7))
print('The accuracy of the Logistic Regression classifier on test data is {:.2f}'.format(ts_sc7))
```

8. Naive Bayes

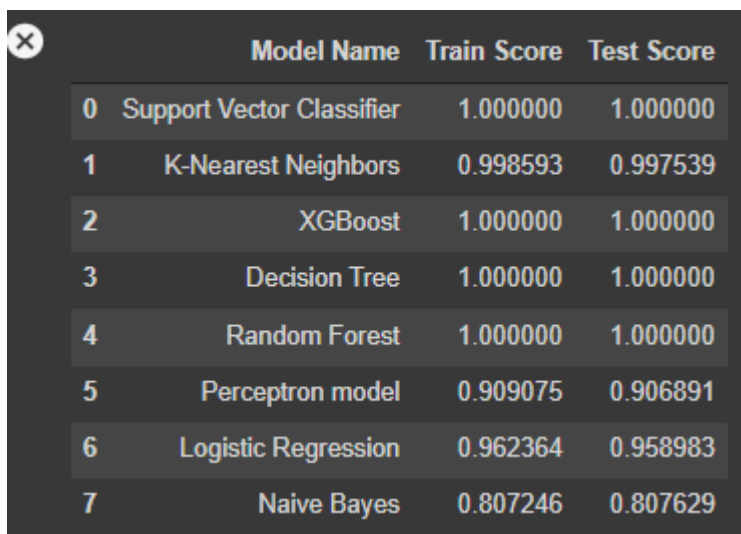
- o Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- o It is mainly used in *text classification* that includes a high-dimensional training dataset.
- o Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- o It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Implementation:-

from sklearn.naive_bayes we import MultinomialNB
and then use MultinomialNB() to fit the X_train and y_train. And find the accuracy of both test and train data

```
from sklearn.naive_bayes import MultinomialNB
Algorithm.append('Naive Bayes')
mnb = MultinomialNB()
mnb.fit(X_train, y_train)
tr_sc8=mnb.score(X_train, y_train)
ts_sc8=mnb.score(X_test, y_test)
print('The accuracy of the Naive Bayes classifier on training data is {:.2f}'.format(tr_sc8))
print('The accuracy of the Naive Bayes classifier on test data is {:.2f}'.format(ts_sc8))
```

CONCLUSION:-

A screenshot of a table with a dark background and light text. The table has four columns: an index column, a 'Model Name' column, a 'Train Score' column, and a 'Test Score' column. There are eight rows of data, indexed from 0 to 7. The models listed are Support Vector Classifier, K-Nearest Neighbors, XGBoost, Decision Tree, Random Forest, Perceptron model, Logistic Regression, and Naive Bayes. The scores are numerical values representing accuracy.

	Model Name	Train Score	Test Score
0	Support Vector Classifier	1.000000	1.000000
1	K-Nearest Neighbors	0.998593	0.997539
2	XGBoost	1.000000	1.000000
3	Decision Tree	1.000000	1.000000
4	Random Forest	1.000000	1.000000
5	Perceptron model	0.909075	0.906891
6	Logistic Regression	0.962364	0.958983
7	Naive Bayes	0.807246	0.807629

We can see that some of the algorithms have similar train and test scores.

The model accuracy of the Support Vector Classifier, XGBoost, Decision Tree and Random Forest are 100%. But this does not mean that the models are the best. Rather they are overfitted and will tend to yield a stereotypic result for any more variations in the dataset. So we will rule them out.

Apart from them, we can see that Perceptron Model ,Logistic Regression and Naïve Bayes has a the highest accuracy score around 90% which is a very good accuracy. This also has least probable changes of overfitting. So, the best model for the analysis would be Perceptron Model ,Logistic Regression and Naïve Bayes.

RESULT:-We have Summarized the Performance measures for each supervise machine learning algorithm.

Signature of the Student

Arnav Kumar

Date: 10/4/2022 Ex No:9	Title of the Lab IMPLEMENTATION OF NLP PROGRAMS	Name: Arnav Kumar Registration Number: RA1911027010040 Section:N1 Lab Batch:2 Day Order:3
--	--	---

AIM:- Implementation of NLP Programs in python

Predicting Similar Words from Dataset using NLP in Python.

Algorithm:-

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment.

Word2Vec:- The effectiveness of Word2Vec comes from its ability to group together vectors of similar words. Given a large enough dataset, Word2Vec can make strong estimates about a words meaning based on their occurrences in the text

Implementation:-

Import the necessary packages. First we scrap the data from the dataset which is a website. Then convert the paragraphs into words and removing the stop words. Then use Word2Vec to find the similar words of the desired word.

```
import bs4 as bs
```

```
import urllib.request
```

```
import re
```

```
import nltk
```

```
url="https://en.wikipedia.org/wiki/Natural_language_processing"
```

```
s_data=urllib.request.urlopen(url)
```

```
data=s_data .read()
```

```
p_data=bs.BeautifulSoup(data,'lxml')
```



```
paragraphs=p_data.find_all('p')

data_text=""
for p in paragraphs:
    data_text += p.text

try:
    import string
    from nltk.corpus import stopwords
    import nltk
except Exception as e:
    print(e)

class PreProcessText(object):
    def __init__(self):
        pass
    def __remove_punctuation(self,text):
        message=[]
        for x in text:
            if x in string.punctuation:
                pass
            else:
                message.append(x)
        message=''.join(message)

        return message

    def __remove_stopwords(self,text):
        words= []
        for x in text.split():

            if x.lower() in stopwords.words('english'):
```

```
        pass
    else:
        words.append(x)
    return words

def token_words(self,text=''):
    message=self.__remove_punctuation(text)
    words=self.__remove_stopwords(message)
    return words

import nltk
flag=nltk.download("stopwords")

if (flag=="False" or flag==False):
    print("Failed to Download Stop Words")
else:
    print("Downloaded Stop words ..... ")
    helper=PreProcessText()
    words=helper.token_words(text=data_text)

#words

"""#Training the model"""

#pip install gensim==3.8.3

#!pip install word2vec

from gensim.models import Word2Vec

#model=Word2Vec([words],min_count=1)
model=Word2Vec([words],size=200>window=7,min_count=2,workers=3)
```

```
vocabulary=model.wv.vocab
```

```
vocabulary
```

```
similar_words=model.wv.most_similar('deep')
```

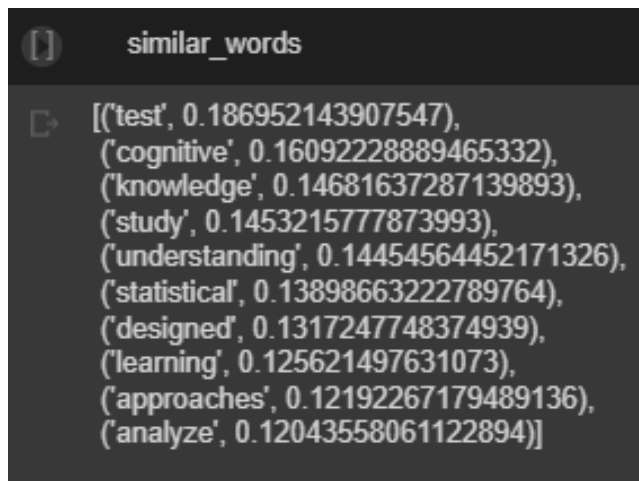
```
similar_words
```

```
"""#Reference:-
```

```
https://soumilshah1995.blogspot.com/2020/05/lets-build-simple-nlp-model-that.html
```

```
"""
```

OUTPUT:-



```
similar_words  
[('test', 0.186952143907547),  
 ('cognitive', 0.16092228889465332),  
 ('knowledge', 0.14681637287139893),  
 ('study', 0.1453215777873993),  
 ('understanding', 0.14454564452171326),  
 ('statistical', 0.13898663222789764),  
 ('designed', 0.1317247748374939),  
 ('learning', 0.125621497631073),  
 ('approaches', 0.12192267179489136),  
 ('analyze', 0.12043558061122894)]
```

RESULT:- Implemented a NLP program in Python to Predict the similar word of the give word.

Signature of the Student

Arnav Kumar

Date: 21/4/2022 Ex No:10	Title of the Lab IMPLEMENTATION OF DEEP LEARNING ALGORITHMS FOR AN APPLICATION	Name: Arnav Kumar Registration Number: RA1911027010040 Section:N1 Lab Batch:2 Day Order:3
---	---	---

AIM:- To implement Image classification using CNN (CIFAR10 dataset)

About Dataset:-

The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:

airplane



automobile



bird



cat



deer



dog



frog



horse



ship



truck



Implementation:-

We will use convolutional neural network for this image classification problem. First we will train a model using simple artificial neural network and then check how the performance looks like and then we will train a CNN and see how the model accuracy improves. This tutorial will help you understand why CNN is preferred over ANN for image classification.

Program:-

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np

(X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()
X_train.shape

X_test.shape

y_train.shape

y_train[:5]

y_train = y_train.reshape(-1,)
y_train[:5]

y_test = y_test.reshape(-1,)

classes =
["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

def plot_sample(X, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(X[index])
    plt.xlabel(classes[y[index]])

plot_sample(X_train, y_train, 0)

plot_sample(X_train, y_train, 1)

X_train = X_train / 255.0
X_test = X_test / 255.0

ann = models.Sequential([
    layers.Flatten(input_shape=(32,32,3)),
    layers.Dense(3000, activation='relu'),
```

```
        layers.Dense(1000, activation='relu'),
        layers.Dense(10, activation='softmax')
    ])

ann.compile(optimizer='SGD',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])

ann.fit(X_train, y_train, epochs=5)

from sklearn.metrics import confusion_matrix , classification_report
import numpy as np
y_pred = ann.predict(X_test)
y_pred_classes = [np.argmax(element) for element in y_pred]

print("Classification Report: \n", classification_report(y_test, y_pred_classes))

cnn = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

cnn.compile(optimizer='adam',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])

cnn.fit(X_train, y_train, epochs=10)

cnn.evaluate(X_test,y_test)

y_pred = cnn.predict(X_test)
y_pred[:5]

y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]

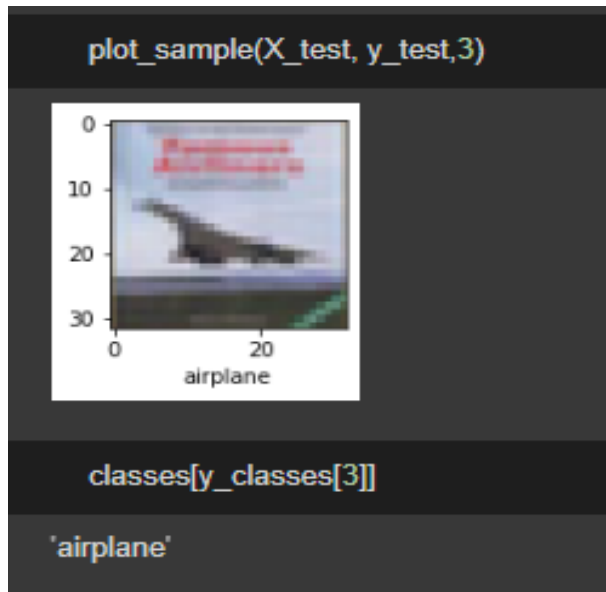
y_test[:5]

plot_sample(X_test, y_test,3)
```

```
classes[y_classes[3]]
```

```
classes[y_classes[3]]
```

OUTPUT:-



RESULT:- Successfully Implemented Image classification using CNN on CIFAR10 dataset.

Signature of the Student

Arnav Kumar