yeskay The BOSS

INDEX

NAME : Adhish. k STD : _____ SEC : _____ ROLL NO. : _____

S. No.	Date	Title	~~Page No.~~ marks	Teacher's Sign / Remarks
1)	31/07/24	Sample programs	9	
2)	7/08/24	Problem Statement Domain	10	
3)	14/8/24	N- Queen problem	10	
4)	4/9/24	Depth - First - Search.	10	
5)	11/9/24	To implement A*	10	
6.	11/9/24	To implmnt Ao*	10	
7	25/9/24	Desision tree	10	
8.	9/10/24	K-means	10	
9.	16/10/24	Artificial neural network	10	
10.	28/10/24	Minimax	10	
11.	30/10/24	Introduction to prolog	10	
12.	6/11/24	Prolog - family tree	10	

completed

Ex: 01

Date: 31/07/2024

PYTHON

SAMPLE PROGRAMS

USING GOOGLE COLLAB

1) def factorial (n):

return 1 if (n == 1 or n == 0) else n * factorial (n-1)

print ("factorial of", num, "is", factorial (num))

num = int (input ("Enter a number:"))

factorial (num)

Output

Enter a number: 6

720

2) import math

def isPerfectSquare (x):

s = int (math.sqrt (x))

return s*s == x

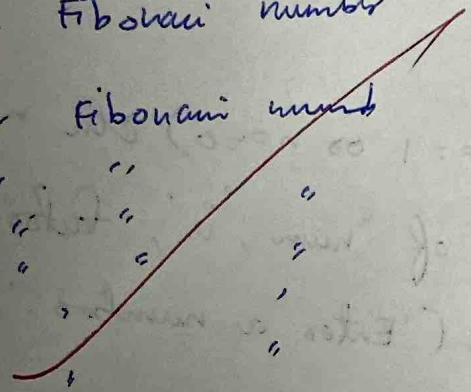
def isFibonacci (n):

return isPerfectSquare (5*n*n + 4) or isPerfectSquare (5*n*n - 4)

```
for i in range (1, 11):
    if (is Fibonaci (i) == True):
        print (i, "is a Fibonaci Number")
    else:
        print (i, "is a not Fibonnaci Number")
```

Output:

1 is a Fibonaci number
2 is a Fibonaci number
3 is a Fibonaci number
4 is a Fibonaci numb
5 " " " "
6 " " " "
7 " " "
8 " "
9 "



3) Finding largest array :

```
def largest (arr, n):
    max = arr [0]
    for i in range (1, n)
        if arr [i] > max :
            max = arr [i]
    return max

arr = [10, 324, 45, 90, 9808]
n = len (arr)
Ans = largest (arr, n)
print ("Largest in given array", Ans)
```

Output:

Largest in given array 9108

3) Sum of 2 numbers

```
num 1 = 15
num 2 = 12
Sum = num 1 + num 2
print ("sum of", num1, "and", num 2, "is", sum)
```

Output:

sum of 15 and 12 is 27

4) Find max of 2 numbers.

```
def maximum (a, b):
    if a >= b:
        return a
    else:
        return b

a = 2
b = 4
print (maximum (a, b))
```

Output

4

6) Armstrong Number

```
def power (x, y):
    if y == 0:
        return 1
    if y % 2 == 0:
        return power (x, y//2) * power (x, y//2)
    return x * power (x, y//2) * power (x, y//2)

def order (x):
    n = 0
    while (x != 0)
        n = n + 1
        x = x //10
        x = x //10
    return n

def isArmstrong (x):
    n = order (x)
    temp = x
    sum1 = 0
    while (temp != 0)
        r = temp % 10
        sum1 = sum1 + power (r, n)
                    //10
```

x = 153
print (isArmstrong (x))

x = 1253
print (isArmstrong (x))

Output:

True

False

⇒ Sum of array:

```
def _sum (arr):
    sum =0
    for i in arr:
        sum = sum + i
    return (sum)
if_name - == '_main' -:
arr =[12, 3, 4, 15]
h= len (arr)
ans = - sum (arr)
print ('sum of the array is', ans)
```

output:

sum of array is 34

Date : 14/3/24 Ex: 2

1) N-QUEEN PROBLEM:

~~global N~~

AIM: To solve N-Queen problem using Python.

Program:

```
def print_solution(board):
    for row in board:
        print(" ".join('Q' if col else '.' for
            col in row))

    print()
def is_safe(board, row, col, n):
    for i in range(col):
        if board[row][i]:
            return False

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

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

```
def solve_n_queen(board, col, n):
    if col >= n:
        return True
    for i in range(n):
        if is_safe(board, i, col, n):
            board[i][col] = True
            if solve_n_queen(board, col+1, n):
                return True
            board[i][col] = False
    return False

def n_queens(n):
    board = [[False]*n for _ in range(n)]
    if not solve_n_queen(board, 0, n):
        print("No solution exists")
    else:
        print_solution(board)

if __name__ == "__main__":
    try:
        n = int(input("Enter the value of N for the N-queens problem: "))
```

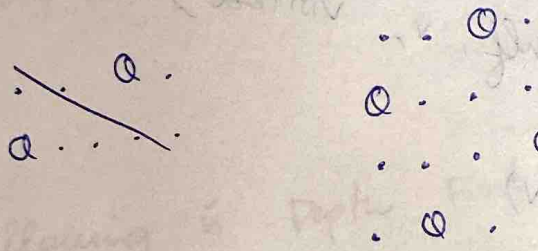
```
if n<1:
    print ("N must be a positive integer :)

else:
    n_queen (n)

except ValueError:
    print ("Invalid input. Please enter a
           positive integer.")
```

Output:

Enter the value of N for the N Queen problem : 4



2) ~~Depth - first - Search~~

Aim: ~~To implement~~ Depth - First - Search using python.

Result

RESULT:

Thus the above code is executed sucessfully

2) DEPTH - First - Search.

Program:

```
from collection import defaultdict

class Graph:
    def __init__(self):
        self.graph = defaultdict(list)

    def addEdge(self, u, v):
        self.graph[u].append(v)

    def DFSUtil(self, v, visited):
        visited.add(v)
        print(v, end=' ')
        for neighbour in self.graph[v]:
            if neighbour not in visited:
                self.DFSUtil(neighbour, visited)

    def DFS(self, v):
        visited = set()
        self.DFSUtil(v, visited)
```

```
if __name__ == "__main__":

    g = Graph()
    g.addEdge(0, 1)
    g.addEdge(0, 2)
    g.addEdge(1, 2)
    g.addEdge(0, 0)
    g.addEdge(2, 3)
    g.addEdge(3, 3)

    print("Following is Depth First Traversal
          (starting from vertex 2)")
    g.DFS(2)
```

Output:

Following is Depth First Traversal
(starting from vertex 2)

2 0 1 3

Result:

Thus the above program is executed sucessfully

11/9/2024

Aim: To implement AO* in Python

PROGRAM:

```
def cost (H, condition, weight=1):
    cost = {}
    if 'AND' in condition:
        AND_nodes = condition['AND']
        Path_A = 'AND'.join(AND_nodes)
        cost[Path_A] = sum(H(Node) + weight for node in AND_nodes)

    if 'OR' in condition:
        OR_nodes = condition['OR']
        path_B = 'OR'.join(OR_nodes)
        cost[Path_B] = min(H[node] + weight for node in OR_nodes)
    return cost

def update_cost(H, conditions, weight=1):
    least_cost = {}
    for key in sum(list(conditions.keys())):
        condition = conditions[key]
        c = cost(H, condition, weight)
        H[key] = min(c.values())
        least_cost[key] = c
```

```
def shortest_path (start, updated_cost, H):

    path = cost
    if start in updated_cost
    min_cost = min (updated_cost [start].values ())
    best_path = min (updated_cost [start], key
                = updated_cost [start].get)
    next_nodes = best_path.split(',')

if ~~len~~ (next_nodes) == 1:
    next_node = next_nodes [0]
    path += '←' + shortest_path (next_node,
                                 ~~updated~~ = [

if len (next_nodes) == 1:
    next_node = next_node [0]
    path += '←(' + shortest_path (node, updated_cost
                                  H)

else
    path += '←(' + best_path + ')'
    path += '[' + ','.join (shortest_path (node
                                w next_nodes)) + ']'

return path

H = {'A': -1, 'B': 5, 'C': 2, 'D': 4, 'E': 7,
     'F': 9, 'G': 3, 'H': 0, 'I': 0, 'J': 0}
```

Condition = {

'A' = { 'OR' : ['B'], 'AND' : ['C', 'D']}

'B' : { 'OR' : ['E'], 'AND' : ['C', 'D']}

'C' : { 'OR' : ['G'] : 'AND' ['H', 'I']},

'D' : { 'OR' : ['J']}

weight = 1

print ('Updated Cost :')

updated_cost = update_cost (H, conditions, weights)

print ('Updated Cost :')

updated_cost = updated_cost (H, Conditions, weight = 1)

print ('*' * 75)

print ('shortest path :\n', shortest_path ('A', updated_cost, H))

OUTPUT:

D: {'OR' : ['J']} >>> {'J' : 1}

C : {'OR' : ['G'], 'AND' : ['H', 'I']}

>>> {'H AND I' : 2, 'A' : 4}

B : {'OR' : ['E'], ['F']} >>> {'E OR F' : 3}

A : {'OR' : ['B'], 'AND' : ['C', 'D']}

>>> {'C AND D' : 5, 'B' : 7}

SHORTEST PATH:

A <- (C AND) [C <- (H AND J) [H+1] + D <- J]

RESULT:

Thus the above program is executed sucessfully.

2) To implement A* Algorithm:

AIM: To implement A* Algorithm

Program:

```
import math
import heapq

class cell:
    def __init__(self):
        self.parent_i = 0
        self.parent_j = 0
        self.g = float('inf')
        self.f = float('inf')
        self.h = 0

ROW = 9
COL = 10
```

```
path.append((row, col))
path.reverse()

for i in path:
    print("->", i, end=" ")

print()

def a_star_search(grid, src, dest):
    if not is_valid(src[0], src[1], dest):
        print("We are already at the
              destination")

    return.

cell_details = [[cell() for _ in range(col)]
                for _ in range(ROW)]

i = src[0]
j = src[1]
cell_details[i][j].f = 0
cell_details[i][j].g = 0
cell_details[i][j].h = 0
                    ; parent_i = i
                    . parent_j = j

open_list = []
found_dest = False.
```

trace_path (cell, details, dest)

found_dest = True

return

g_new = cell_details [i] [j] . g + 1.0

h_new = calculate_h_value (new_i, new_j, dest)

f_new = g_new + h_new.

if cell_details == float ('inf')

cell_details [new_i] [new_j] . f = f_new

" " . g = g_new

" " . h = h_new

" " . parent_i = i

" " . parent_j = j

def main ():

grid = [

[1, 0, 1, 1, 1, 0, 1, 1, 1]

[1, 1, 1, 0, 1, 1, 0, 1 0, 1, 1]

[1, 1, 1, 0, 1, 1, 0, 1 . 0, 1]

[0, 0, 1, 0, 1, 0, 0, 0, 0, 1]

[1, 0, 1, 0, 1, 1, 1, 0, 1, 0]

[1, 0, 0, 0, 0, 1, 0, 0, 0, 1]

[1, 0, 1, 1, 1, 0, 1, 1, 1]

[1, 1, 1, 0, 0, 1, 0, 0, 1]

src = [8, 0]

dest = [0, 0]

a_star_search (grid, src, dest)

if __name__ == "__main__":

main()

OUTPUT:

The destination cell is found

The path is

→ (8, 0) → (7, 0) → (6, 0) → (5, 0)

→ (4, 1) → (3, 2) → (2, 1) → (1, 0)

→ (0, 0)

RESULT:

Thus the above code is executed successfully

Ex: no: 7

Date: 25/1/24

IMPLEMENTATION OF DECISION TREE CLASSIFICATION TECHNIQUES

AIM: To implement a decision tree classification technique for gender classification using python.

Explanation:

Import tree from sklearn

Call the function DecisionTreeClassifier()

from tree

Assign values for x and y

Call the function predict for Predicting on the basis of given random values for each given feature.

Display output

PROGRAM:

```
import pandas as pd;
from sklearn.tree import DecisionTreeClassifier
```

```
data = {
    'Height' : [152, 151, 172, 185, 167, 180, 157, 180,
                164, 199],
    'Weight' : [45, 57, 72, 85, 68, 78, 22, 90, 66,
                88],
    'Gender' : ['Female', 'Female', 'Male',
                'Male', 'Female', 'Male', 'Female',
                'Male', 'Female', 'Male']
}

df = pd.DataFrame(data)
x = df['Height', 'Weight']
y = df['Gender']

classifier = DecisionTreeClassifier()
classifier.fit(x, y)
height = float(input("Enter height (in cm)
                     for prediction: "))
weight = float(input("Enter weight in kg for
                     prediction:"))

random_values = pd.DataFrame([[height,
weight]], columns=['Height', 'Weight'])
```

```
predicted_gender = classifier.predict(random_value)
print(f"predicted gender for height
    {height} cm and weight {weight} kg,
    {predicted_gender[0]}")
```

Output:

Enter height (in cm) for prediction: 169

Enter weight (in kg) for prediction: 61

predicted gender for height 169.0 cm and weight 61.0 kg - female

RESULT:

Thus the decision tree classification has been executed sucessfully.

Ex no: 8 DATE: 2/10/24

IMPLEMENTATION OF ARTIFICIAL NEURAL NETWORKS FOR AN APPLICATION USING PYTHON - REGRESSION

AIM:

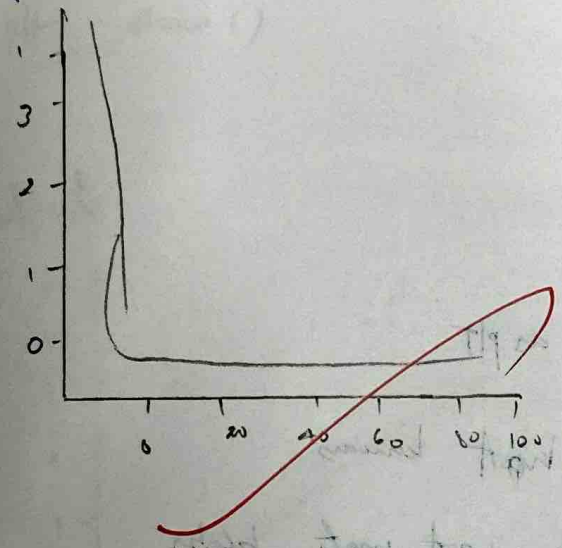
To implementing artificial neural networks for an application in Regression using Python.

PROGRAM:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import standard scal
import matplotlib.pyplot as plt
# Generating synthetic dataset for demonstration
# Replace this with your own dataset
x = np.random.rand(1000,3)
if = 3* x[:,0) + 2*x[:, 1]** 2+1.5 *np.sin(x
[:,2]*np.pi) + hp.random.normal
10000) Non linear relationship
```

```
label = 'Validation loss'
Plt.title ('Training and Validation loss')
plt.xlabel ('Epochs')
plt.ylabel ("Loss")
plt.legend()
plt.show()
```

output:



RESULT:

Thus the artificial neural networks for regression has been implemented manually.

Ex no : 9

DATE : 16/10/24

Implementation of clustering techniques k-means

Aim: To implement a k-means clustering technique using python language.

PROGRAM:

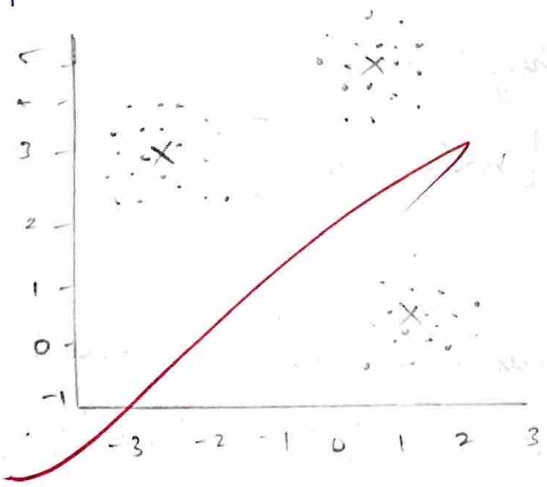
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import kmeans
from sklearn.datasets import make_blobs

X, y = make_blobs (n_sample = 300,
centers = 3, cluster_std = 0.60, random_state = 0)

k = 3
kmeans = kmeans (n_clusters = 4, random_state = 0)
y_kmeans = kmeans.fit_predict (x)
plt.figure (fig_size = (8,6))
plt.scatter (x [:,0], x [:,1] c = y_kmeans
's = 30, cmap = 'viridis', label = 'clusters')
```

```
plt.scatter(centers[:,0], centers[:,1], c='red',
s=200, alpha=0.75, marker='x',
label='centroids')
plt.title('K-means Clustering Results')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.show()
```

Output:



RESULT:

Thus the K-means cluster technique has been implemented manually

Ex : 10

Date : 20/12/24

MINIMAX ALGORITHM

AIM: To implement Minimax algorithm

Program:

```
import math
def minimax(depth, node_index, is_maximizer,
            scores, height):
    if depth == height:
        return scores[node_index]
    if is_maximizer:
        return max(minimax(depth+1, node_index*2,
            False, scores, height), minimax(depth+1,
            node_index*2+1, False, scores, height))
    else:
        return min(maximax(depth+1, node_index*2,
            True, scores, height), minimax
    (depth+1, node_index*2+1, True, scores, height))
def calculate_tree_height(num_leaves):
```

```
return math.log2(num_leaves)

scores = [3, 5, 6, 9, 1, 2, 0, -1]

tree_height = calculate_tree_height(len(scores))

optimal_score = minimax(0, 0, True, scores, tree_height)

print(f"The optimal score is {optimal_score}")
```

Output:

The optimal score is: 5

RESULT:

Thus the decision tree program has been executed successfully.

Ex: 11

date: 26/10/24

INTRODUCTION TO PROLOG

AIM: To learn prolog terminologies and write basic programs.

TERMINOLOGIES:

1) Atomic terms:

They are usually strings made up of lower and upper case letter, digits and the underscore, starting with a lowercase letter eg. dog ab_c_321

2) Variables:

They are strings of letter, digits and underscore, starting with a capital letter or an underscore Eg: Dog, Apple_420

3) Compound terms:

Compound terms are made up of a Prolog atom and a number of arguments enclosed in parenthesis and separated by commas eg: is_bigger(elephant, X), fg:(X,Y),7)

4) Fact: A fact is a predicate followed by a dot eg. bigger-animal (whale). like. is-beautiful

5) Rule
A rule consists of a head and body
eg is-smaller (X, Y, Yes-bigger (Y, X), aunt (Aunt, child). sister (Aunt, Aunt), parent (parent, child)

Program:

KB 1

woman (mia)

woman (jody)

woman (yolanda)

playsAguitar (jody)

party

Query 1: ? - woman (mia)

Query 2: ? - playsAguitar (mia)

Query 3: ? - party

Query 4: ? - concert

likes (john, boothing)

mammid (x, y) :- likes (x, y), likes (y, x)

friends (x, y) :- likes (x, y), likes (y, x)

Output :

? = -likes (dan, x)

k = sally

? = mammid (dan, sally)

True

? - mammid

false

KB 4

food (burgers)

food (sandwich)

food (pizza)

lunch (sandwich)

dinner (pizza)

meal (x) :- food (x)

Output :

?- food (pizza)

true

?- meal (x), lunch (x)

x = sandwitch

KBS

```
own (jack, car (bmw))
own (jim, car (civic))
own (ahmad, car (civic))
own (jane, car, (civic))
sedan (car, civic)
```

Output:

```
own (jack, car (bmw))
own (jim, car (civic))
own (ahmad, as (civic)
own (jane, car (civic))
sedan (car (bmw)
```

Output

```
? own (jim, x)
x = car (civic)
-own (jim-)
tom
```

RESULT:

Thus the bmw prolog program has been executed successfully.

Ex: 12 Date: 6/11/24

PROLOG - FAMILY TREE

AIM: To develop a family tree program using PROLOG with all possible facts, rules and queries.

PROGRAM:

Knowledge base:

male (peter)
male (john)
male (chris)
male (kumi)
female (betty)
female (jane)
female (lisa)
female (helen)
Parent of (chris, peter)
parent of (chris, betty)
parent of (kelvin, lisa)
parent of (jay, helen)

father (x,y) . male (y) . parent (x,y)

mother (x,y) - female (y), parent (x,y)

grand-father (x,y) :- male (y), parent of (x,z)

parent (z, y)

grand-mother (x, y) . female (y) . parent of (x,z)

parent (z, y)

brother (x,y) . male (y,), father (x,z), father

(y,w) . z = w (sister (x, y) grand (x, z)

father (x, z) , father (y, w) . z = w

Output:

male (peter)

true

father (chris-peter)

true

father (chris, katy)

false

mother (chris , x)

x = katy

brother (chris , katy)

false

RESULT:

Thus program for family tree program has been created successfully.