INDEX

Name: K. Bawesh karthik Age:

Class: Sec: Subject:

Address:

Emergency Contact :

Sl No	Particulars	Page No / marks.	
1	8 Queen problem	6	
2	Dfs	6	
3	Dfs – water jug problem	6	
4	A* search Algorithm	6	
5	Implementation of decision tree	6	
6	Implementation of Decision tree	8	
7	Implementation of Clustering technique	8	
8	Minmax Algorithm	10	
9	Introduction to prolog	10	
10	Prolog family tree	10	

Late submission

completed

Python programs

1. Simple Calculator

```
def add (x,y);
    return x+y
def subtract (x,y)
    return x-y
def multiply (x,y);
    return x*y
def division (x,y);
    if y==0
    return "Error: Division by zero"
else:
    return x/y
print ("Select operation:")
print ("1. Add")
print ("2. Subtract")
print ("3. multiply")
print ("4. Divide")
choice = input ("Enter choice (1/2/3/4):")
num1 = float (input ["Enter first number:"])
num 2 = float (input ("Enter second number:")
if choice == '1'
    print [num1, "+", num 2, "=", add (num1
                                       num2))
elif choice == '2'
    print (num 1, "-", num 2, "=", subtract,
                                   (num 1, num 2))
elif choice == '3'
    print (num, "*", num 2, "=", multiply
                                   (num), num2))
```

```
elif choice == '4':
    print(num1, "/", num2, "=", divide(num1, num2))
else:
    print("Invalid Input")
```

OUTPUT

1. Add
2. Subtract
3. Multiply
4. Divide

```
enter choice (1/2/3/4):1
enter first number:30
enter second number:50
30.0 + 50.0 = 80.0
```

2.)

```
def multiplylist(mylist):
    result = 1
    for x in mylist:
        result = result * x
    return result
list1 = [2,2,3]
list2 = [3,3,4]
print(multiplylist(list1))
print(multiplylist(list2))
```

Output

```
12
36
```

3.)

```
lst = ['iris', 'orchids', 'rose', 'lavender', 'Lily', 'carnations']
print ("original list is:", lst)
lst.remove('orchids')
print ("After deleting the item:", lst)
```

Output

original list is: ['iris' 'orchids' rose, lavender 'lily' 'carnations']

After deleting the item: ['iris', 'rose', lavender, 'lily', 'carnations']

4.)

```
test_list3 = [1, 4, 5, 3, 5]
test_list4 = [3, 5, 3, 2, 5]
test_list3 = test_list3 + test_list4
print ("concatenated list using +:" + str (test_list3))
```

OUTPUT

Concatenated list using +: [1, 4, 5, 5, 3, 3, 5, 2, 5]

5.)

```
list1 = [10, 20, 20, 4, 45, 43, 45, 99, 99]
list2 = list(set(list1))
list2.sort()
print ("second largest element is:", list2[-2])
```

OUTPUT

Second largest element is: 45

6.)

```
test_str = 'chan, Hi'
print ("The original string is : " + str(test_str))
x = set(test_str)
yes = []
for i in x:
    if (test_str.count(i) % 2 != 0):
        yes.append(i)
print ("The odd frequency characters are : "
    + str(yes))
```

OUTPUT

The original string is : chan, Hi

The odd frequency characters are :

['p', ' ', 'c', 'a', 'h', 'n', ',', 'H']

N - Queens problem

Aim:-

To execute the N- Queen problem using Python

Code:-

```
def is_safe (board, row, col, n):
    for i in range (col):
        if board [row] [i] == 1:
            return false
    for i, j in zip (range (row, -1, -1), range
    (col, -1, -1)):
        if board [i] [j] == 1:
            return false
    for i, j in zip (range (row, n), range
    (col, -1, -1)):
        if board [i] [j] == 1:
            return false
    return true
def solve - n - queens - util (board, col, n):
    if col >= n:
        return true
    for i in range(n)
        if is - safe (board, i, col, n):
            board [i] [col] = 1
            if solve - n - queens - util (board, (col+1, n):
```

return false

for i in range(N):

def is_safe(board, i, col, N):

board[i][col] = True

return

def solve_n_queens(N):

board = [[False] * N for _ in range(N)]

print("Solution for", N, "queens problem:")

solve_n_queens_util(board, 0, N)

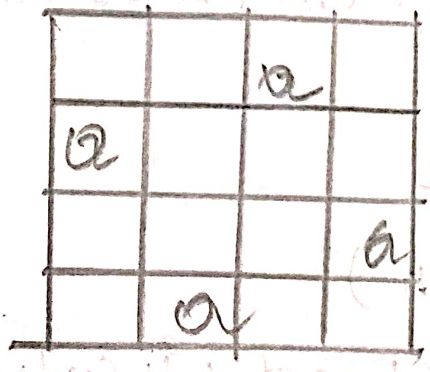
N_values = [4, 8]

for N in N_values:

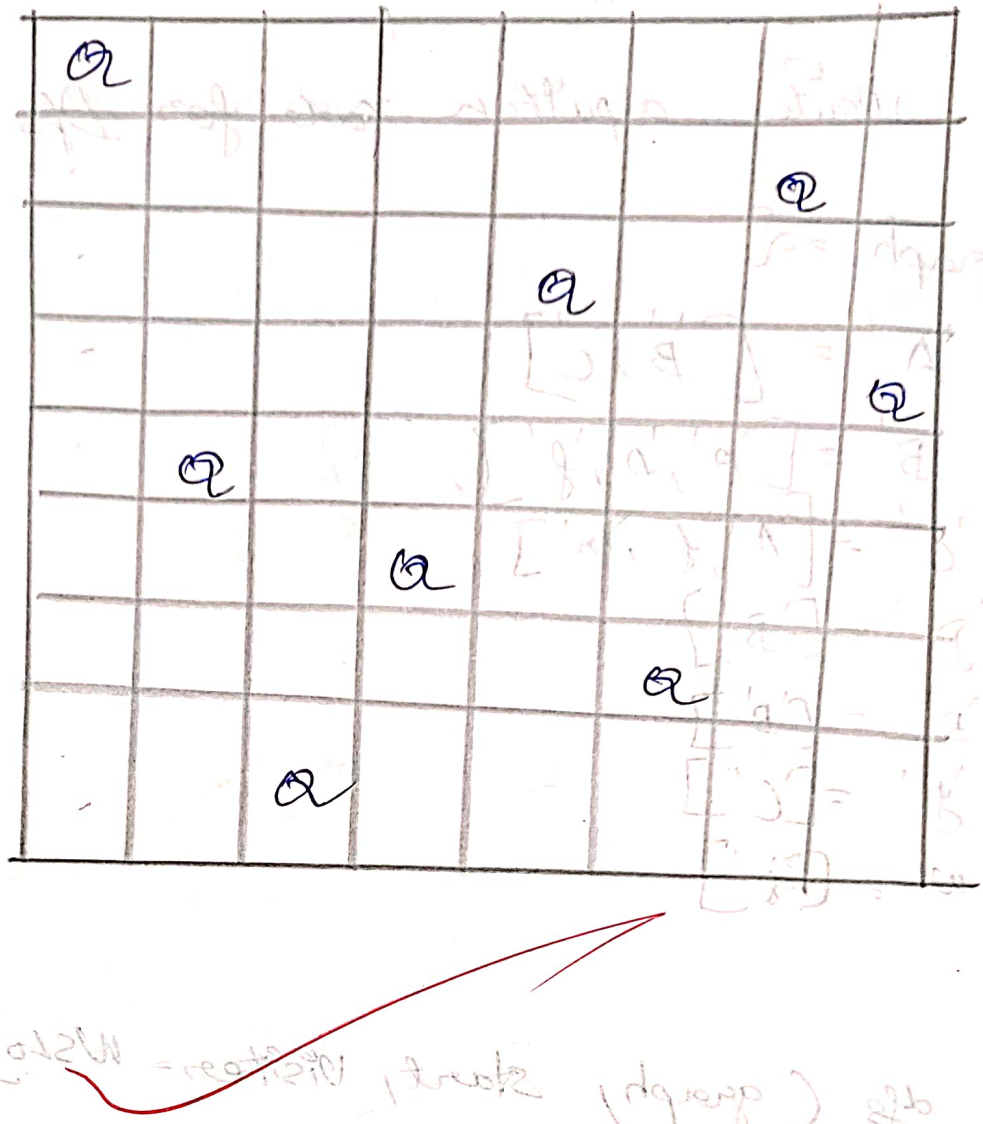
solve_n_queens(N)

print("\n" + "-" * 20 + "\n")

Output: Solution for 4 queens



solution for 8queen



Result

Thus the above python code executed successfully

Dept first search graph

Aim: Write a python code for Dfs.

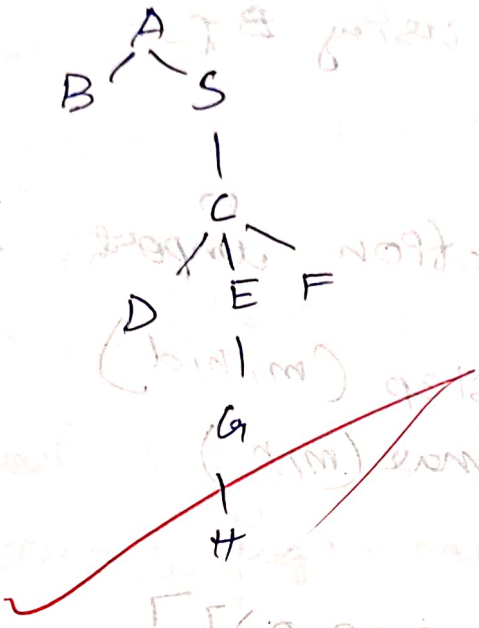
```
graph ={
    'A' = ['B', 'C']
    'B' =['A', 'D', 'f'],
    'C' =['A', 'f', 'G'],
    'D' =['B']
    'E' = ['B']
    'y' =['C']
    w = ['i']
}

Def dfs (graph, start, visited = NULL):
    if visited is rose:
    visited.add (start)
    print (start even ='')
    for (start, end='')
    for (neighbour in graph [start]:
    if height bor not in visited'

Df= (graph, neighbon, visited]
dfs (graph (neighbor, visited')
```

Output

A B D E F C F G



Result:-

Thus the above python code for dfs executed successfully.

Water Jug problem using BFS.

AIM:-

To write a python code water Jug Problem using BFS

Program:-

```
from collection import Deque
Def min_ step (m, n, d)
  if  d > max (m, n)
  return = 1

q = deque [[[0,0]]]
visited = [ [false] (n+1) for-in range(m+1)]
visited [0] [0] = True.
while q:
    jug1, jug2, steps = q . popleft ()
    if jug1 == = d or jug2 == = d
    return steps.
if not visited [m] [jug 2]:
visited [m] [jug2] = true
q. append ((m, jug 2, steps + 1))
if not visited [jug 1] [n]:
  visited [jug 1] [n] = True.
```

if not visited [0][jug2]:
visited [0][jug2] = True
q.append ((0, jug2, steps+1))
if not visited [jug1][0]:
visited [jug1][0] = True.
q.append ((jug1, 0, step+1))
Pour1to2 = min (jug1, n - jug2)
if not visited [jug1 = Pour1to2][jug2 + Pour1to2]:
visited [jug1 - Pour1to2][jug2 + Pour1to2]
q.append ((jug1 - Pour1to2, jug2 + Pour1to2, steps+1))
Pour2to1 = min (jug2, n - jug1)
if not visited [jug1 + Pour2to1][jug2 - Pour2to1]
visited [jug1 + Pour2to1][jug2 - Pour2to1]
q.append ((jug1 + Pour2to1, jug2 - Pour2to1, steps+1))
return -1
if __name__ == "__main__":
m, n, d = 4, 3, 2
print (minSteps(m, n, d))
Output
4

Result:-
Thus the above python code for Water Jug problem was executed Successfull. Water Jug Successfully.

A* Algorithm

Aim:-

To write a python code for A* algorithm

Program:-

```
Def star (start_node, Stop_node)
  open_set = set (start_node)
  closed_set = set()
  g = {}
    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 == none or g[v] + heuristic v)
                    < graph_nodes[n] == None:
  else:
    for (m, weight) in get_height (n):
    if m not in open_set and m not in
                                closed_set
    open_set.add[m]
    parent [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 parent[n] != n:
path.append(n)
n = Parents[n]
path.append(start_node)
path.reverse()
print("Path found: {}".format(path))
return

    open_set.remove(n)
    closed_set.add(n)
print("path does not exist!")
return none

Def get_neighbours(v):
if (v) in graph_nodes:
    return graph_nodes()
```

```
else
    return node
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)],
}
astar('A', 'G')
```

Output:

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

Result:

Thus the Python program for A* Algorithm was executed Sucessfully.

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 do function Decision tree classifier from tree
* Design value for * and Y
* Call de function predict for. predicting one the basis of given random values for each given features
* Display the output.

CODE:

```
Import pandas as pd.
from sklearn.tree import DecisionTreeClassifier
Data = {
'height': [150, 160, 170, 180, 165, 175, 155,
'weight' 185, 62, 72]
'weight': [50, 60, 70, 80, 65, 75, 55
```

'gender': ['female', 'female', 'male', 'male',

'female', 'male', 'female', 'male', 'Female',

'male']

25

df = pd.DataFrame(data)

x = df[['height', 'weight']]

y = df['gender']

```
Classifier = DecisionTreeClassifier()
classifier = fit (X,Y)
height = float input ("Enter height (cm on
                      your 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 ("Predicted gender for height (in cm)
in cm and weight of weight (kg (predicted)
gender [0]:")
```

Input + output

Enter height (in cm): 150

Enter weight (in kg) : 50

predicted gender for height 150 and weight 50.0 kg : female

Result:

Thus implementation of Decision tree classification technique is done and executed successfully

Implementation of clustering techniques

K - means

Aim

To implement a K near clustering techniques using python language.

Explanation:

* Import K-means from sklearn.cluster
* Assign X and Y
* Call the function K(variable)
* Performs Scatter operation and Display to output

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
num_points = int(input("Enter the number of data points"))
X = np.zeros((num_point, 2))
X = float(input("Enter "+ - coordinate for Data point "+i+": "))
x(i) = [X,Y]
num_clusters = int(input("Enter the number of clusters"))
```

Def process [x, num, Clustor max (. itas =100));

centroids = x (np. random choice [x. shape [0]

num - clustor, replace false)]

for i-in range (max - iters)

Distance = np linalg norm [x [im newtios] - centroids axis = 2))

labels = np orgmin (Distribution, axis=1)

new Centroids = np array [[x [labels = k] mean (axis =a) for k in range (num = clusters))]

if np ll centroids == new centroids):

break

Centroids = new- centroids

return labels, centroied

labels, centroids = ~~kmean~~ (x, num = clusters) .

print ("clusters lables = \n" labels).

print ("cluster. \n", centroids)

~~plt~~ figure figsize = (8,6)]

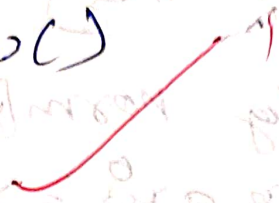
plt scater [x [:,0] x [:,1], c= lables

cmap- shield

marker= 'o', lables - 'Data points'')

plt. scatter ('K - mean distance (from scratch'))

```
plt.xlabel('feature1')
plt.ylabel('feature2')
plt.legend()
plt.grid(True)
plt.show()
```



Exp No : 8

AIM:-

To implement artificial Lealeal networks for an application is pegrasion using python.

Algorithm:

→ Generate Data Create sample Data with a linear relationship.

→ prepare Data : Scale the and output values

→ Split Dot : Divide the data int traing and testing sets.

→ Build models : Create an ANN with input, hidden and output

→ Train Model : fit model the training Data over fuesl epolles

→ Evaluate Model : Test the model using the testing test.

→ Visualize Result : Plot actual predicted Value to acess performance

CODE:-

```
import empty as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train
```

test split

```
from sklearn.preprocessing import StandardScaler
from tensorflow import keras
from tensorflow.keras import layers
np.random.seed(0)

y = A + B * np.random.randn(1000)
scale_X = StandardScaler()
scale_X = StandardScaler()
X_scale = scale_X.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size
                                   = 0.2, random_state=2)
model = keras.Sequential()
layers.Dense(64, activation='relu',
             input_shape=[X_train.shape[1]])
layers.Dense(64, activation='relu')
layers.Dense(1)
model.fit(X_train, y_train, epochs=100
          batch_size=32,
          verbose=0)
loss = model.evaluate(X_test, y_test)
print("Test loss:", loss)
y_pred_scaled = model.predict(X_test)
y_pred = scaler.inverse_transform(y_pred_scaled)
```

plt.title ("Actual vs predicted")

plt.xlabel ('x')

plt.ylabel ('y')

plt.legend()

plt.show()

	feature	Target
0	3.745401	7.846204
1	9.5043	16.343597
2	7.319939	15.400225
3	6.986555	13.194341
4	1.560186	4.239934

Epoch 1/100

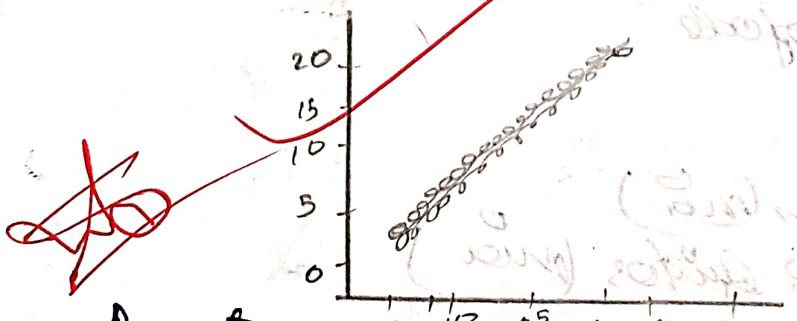
20/20 —— 2s 6ms/step loss: 4.436730

!

Epoch 100/100

20/20 —— 0s 4ms/step - loss: 34.2.248

Test score (MSE): 3.5400087222305297285



Result: Thus the above program executed successfully.

Introduction to Prolog

EXNO: 9

AIM:-

To Learn Prolog terminologies and write basic programs

TERMINOLOGIES:-

1. Atomic Terms:

Atomic terms are usually strings made up of lower and uppercase letters, digits and the underscore starting with a lowercase letter

Source Code:-

```
woman(mia).
woman(jody).
woman(yolanda).
plays(vincent, guitar).
party.

query 1: woman(mia).
query 2: plays(vincent, guitar).
query 3: party.
```

Query4 :- ? concert

output...

? woman (mia)

true

? plays Air Guitar (mia)

false

? concert.

ERROR. unknown procedure concert/0
DWIM could not correct. goal)

KB2:

happy (Yolanda)

listens2music (mia)

listens2music (yolanda) :- happy (yolanda)

plays AirGuitar (mia) :- listens2music (mia)

plays AirGuitar (yolanda) :- listens2music (yolanda)

Output

Query1 ? plays AirGuitar (mia)

true

Query2 plays AirGuitar (yolanda)

true

KB3

madden Sql:

likes (Sally, dan)

likes (python, britney)

list (john, britney)

married (X, Y) :- list (X, Y), list (Y)

friend (X, Y) :- likes (X, Y) list (Y, X)

output:

Query 1:- ? likes (dan, X)

V = sally.

Query 2. married (dan, Sally)

true

Query ? - married (John, britney)

RB4:-

food (burger)

food (sandwiches)

food (pizza)

lunch (sandwich)

dinner (pizza)

meal (X) :- food (X)

output:

Query 1 : ? food (pizza)

true

Query 2. ? meal (X) burger

Query 3: ? dinner (sandwich)

false:

KBS

query (jack, car(bmw))

query (john, car(chevy))

query (olivia, car(civic))

query (jane, car(chevy))

sedan (car(bmw))

sedan (car(civic))

sedan (car(chevy)).

output:

query -? owns (john, X)

X = car(chevy)

query 2? owns (john)

true

query 3 =? ebars (jane, X), sedan (z)

false

query 5: owns (jonex), tuned(x)

X = car(Chevy)

Result:

Thus to learn prolog terminologies of write program before executed successfully.

PROLOG FAMILY TREE

Exno:10

Aim:-

To Develop a family tree program using prolog write all possible facts, rules and queries

Source code:

Knowledge Base:

```
% facts
male(peter).
male(john).
male(chris).
male(kevin).
female(betty).
female(jenny).
female(ester).
female(ellen).
parent of(chris, peter).
parent of(chris, betty).
parent of(ester, peter).
parent of(ester, betty).
parent of(kevin, chris).
```

Parent of (levin, lisa)
Parent of (Jeny, John)

RULES:

1. Son parent
   son grand parent +1

father (X,Y) :- male (Y) parent of (X,Y)

mother (X,Y) :- female (Y) parent of (X,Y)

grand father (X,Y) :- male (Y) parent of (X,Z), parent of (Z,Z)

grand mother (X,Y) :- female (Y)
Parent of (X,Z) parent of (Z,Y)

brother (X,Y) :- male (Y), father (X,Z)
father (Y,W) :- =

sister (X,Y) female (Y), father (X,Z)
father (X,Z), father (Y,W) :- =

output:

2 male (X), parent of (X,Y)

y	x
peter	chris
peter	helen
john	jeny
chris	levin

? female (Y) parent of (X,Y)

X	Y
betty	chris
betty	jelen
lisa	kevin
chelen	jeny

? male (Y), parent of (X,Z) parent of (Z,Y)

Y	X	Z
peter	kevin	chris
peter	jeny	helen

? female (Y), parent (X,Z) parent of (Z,Y)

Y	X	Z
betty	kevin	chris
betty	jeny	helen

? - male (X) father (X,Z) father (Z,D)
Z = = D
procedure father (A,B) Does not exist

? female (Y), father (X,Z), father (Z,D)
Z = = D procedure father (A,B) does not exist

Result:-

Thus we have Declared a family program using prolog with all possible facts and related queries